OPTIMAL LINEAR COMBINATIONS OF NEURAL NETWORKS[1]

A Thesis
Submitted to the Faculty

of

Purdue University

by

Sherif Hashem

In Partial Fulfillment of the
Requirements for the Degree

of

Doctor of Philosophy

December 1993

©Copyright 1993 by Sherif Hashem.
Internet: shashem@ecn.purdue.edu

[1]TECHNICAL REPORT *SMS 94-4*, SCHOOL OF INDUSTRIAL ENGINEERING.

ACKNOWLEDGMENTS

I am indebted to my advisor, Professor Bruce Schmeiser, for his guidance and support throughout my graduate studies at Purdue. Bruce, your insights and discussions have been invaluable and I appreciate all what you did for me.

I am grateful to my co-advisor, Professor Yuehwern Yih, for many helpful discussions that enhanced the application perspective of the dissertation.

I would like to express my deepest appreciation to Dr. Tariq Samad for his encouragement and support. Tariq, thank you for your friendship.

I express my gratitude to the other advisory committee members: Professors Ronald Rardin and Manoel Tenorio for their helpful comments.

I acknowledge the support of David Ross Fellowship and Purdue Research Foundation research grant 6901627 from Purdue University, and National Science Foundation grants DMS–8717799 and 9358158-DDM.

A special acknowledgement is devoted to the Sensor and System Development Center (SSDC) — currently named Honeywell Technology Center — Honeywell Inc., Minneapolis, MN, where I spent one year as a Research Intern. Beside the broad exposure to many real-world neural networks applications, the interactions with the research staff at SSDC made my internship a unique experience and had a large impact on the direction of my dissertation. In addition to Dr. Tariq Samad, I thank Anoop Mathur for many valuable discussions and suggestions.

To my wife, Fatma, your understanding and encouragement helped me throughout this dissertation. You and my lovely daughters, Mariam and Marwa, fill my life with joy and happiness. I am really grateful to all of you.

To my parents, your lifelong support and inspiration have always strengthened me through many tough times, God bless you.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES ix

ABSTRACT x

1. INTRODUCTION 1

1.1 Motivation 2
1.1.1 Neural network based models 2
1.1.1.1 Model construction 2
1.1.1.2 Some design concerns 2
1.1.1.3 Automating parameter selection 3
1.1.2 Why to combine neural networks? 4
1.1.3 When to combine neural networks? 4
1.1.4 How to interpret combining neural networks? 5
1.1.5 Why use linear methods for combining? 5
1.2 Literature Review on Combining Methods 5
1.2.1 Forecasting literature 7
1.2.2 Neural networks literature 8
1.2.3 Other related literature 8
1.3 Related Publications 9
1.4 Remarks 9

2. OPTIMALLY COMBINING NEURAL NETWORKS 10

2.1 General Problem of Optimally Combining Neural Networks 10
2.2 Linear Combination of Neural Networks 10
2.3 Optimal Linear Combination (OLC) of Neural Networks 11
2.4 MSE-OLC of Neural Networks 11
2.4.1 MSE-OLC combination-weights 12
2.4.1.1 Unconstrained MSE-OLC with a constant term . . . 12
2.4.1.2 Constrained MSE-OLC with a constant term 13
2.4.1.3 Unconstrained MSE-OLC without a constant term . 13
2.4.1.4 Constrained MSE-OLC without a constant term . . . 14

Page

2.4.2 MSE-OLC and ordinary least squares regression 14
2.4.3 Alternate expressions for the constrained MSE-OLC combination-weights 15
2.4.3.1 Constrained MSE-OLC with a constant term 15
2.4.3.2 Constrained MSE-OLC without a constant term . . . 16
2.5 Convex MSE-OLC 17
2.6 Multi-Output MSE-OLC Problem 17

3. ESTIMATING MSE-OLC COMBINATION-WEIGHTS 18

3.1 MSE-OLC Combination-Weights Estimation Problem 18
3.2 Ordinary Least Squares Estimators of the MSE-OLC Combination-Weights 18
3.2.1 Unconstrained MSE-OLC with a constant term 19
3.2.2 Constrained MSE-OLC with a constant term 20
3.2.3 Unconstrained MSE-OLC without a constant term 20
3.2.4 Constrained MSE-OLC without a constant term 21
3.3 Alternate Estimators for the Constrained MSE-OLC Combination-Weights 21
3.3.1 Constrained MSE-OLC with a constant term 21
3.3.2 Constrained MSE-OLC without a constant term 21
3.4 Example 1 22

4. A PRODUCT ALLOCATION PROBLEM 27

4.1 Problem Description 27
4.2 Model Structure 27
4.3 Input-Output Representations 28
4.4 Neural Network Topologies 28
4.5 Neural Networks Training 29
4.6 Results of U-OLC of the Trained Networks 29
4.7 Discussion 32
4.8 Conclusions 33

5. APPROXIMATING A FUNCTION AND ITS DERIVATIVES USING MSE-OLC OF NEURAL NETWORKS 34

5.1 Sensitivity Analysis for Neural Networks with Differentiable Activation Functions 34
5.1.1 Expressions for neural networks output derivatives 35
5.1.1.1 First-order output derivatives 36

Page

5.1.1.2 Second-order output derivatives 36
5.1.2 Example 2 37
5.2 Improving the Accuracy of Approximating a Function and Its Derivatives by Using MSE-OLC of Neural Networks 39
5.2.1 Example 1 (Continued) 40

6. COLLINEARITY AND THE ROBUSTNESS OF THE MSE-OLC 45

6.1 The Robustness of the MSE-OLC 45
6.1.1 Definition 45
6.1.2 Testing the robustness of the MSE-OLC 46
6.2 Collinearity 46
6.2.1 Definition 46
6.2.2 Collinearity and correlation 46
6.2.3 Ill effects of collinearity 47
6.2.3.1 Computational ill-effects 47
6.2.3.2 Statistical ill-effects 48
6.2.3.3 Additional remarks 48
6.3 Collinearity and Estimating the MSE-OLC Combination-Weights . . 48
6.4 Collinearity Detection 50
6.4.1 Common methods for collinearity detection 50
6.4.2 BKW's collinearity diagnostics 51
6.5 Harmful Collinearity 54
6.5.1 Example 3 54
6.5.2 Example 4 56
6.5.3 Conclusions 57
6.6 How to Determine that an Existing Collinearity is Harmful? 57
6.6.1 Some common approaches 58
6.6.2 A cross-validation approach for detecting harmful collinearity 59

7. METHODS FOR IMPROVING THE ROBUSTNESS OF MSE-OLC 60

7.1 Common Methods for Treating Harmful Collinearity 60
7.2 Improving the Robustness of MSE-OLC by Restricting the Combination-Weights 62
7.2.1 Example 3 (continued) 63
7.3 Improving the Robustness of MSE-OLC by Proper Selection of the Neural Networks 64
7.3.1 Algorithm A 66
7.3.2 Algorithm B 67

Page

7.3.3 Algorithm C 69
7.3.4 Algorithm D 70
7.3.5 Algorithm E 71
7.3.6 Algorithm K 73
7.3.7 Example 4 (continued) 73
7.3.8 Example 1 (continued) 74
7.3.9 Modification to the algorithms 74
7.3.9.1 Example 3 (continued) 75
7.3.9.2 Example 4 (continued) 76
7.3.10 Conclusions 76

8. EMPIRICAL STUDY 77

8.1 Part I: Effect of the Size and Quality of the Combination Data 77
8.1.1 Training the neural networks 77
8.1.2 Combining the neural networks 78
8.1.3 Part I—Level 1 78
8.1.4 Part I—Level 2 80
8.1.4.1 Example 5 81
8.1.5 Part I—Level 3 84
8.1.6 Part I—Level 4 84
8.2 Part II: Effect of the Size and Quality of Data on Training and Combining Neural Networks 87
8.2.1 Part II—Level 1 87
8.2.1.1 Example 6 89
8.2.2 Part II—Level 2 91
8.2.3 Part II—Level 3 91
8.2.4 Part II—Level 4 94
8.3 Main Conclusions of the Empirical Study 94

9. SUMMARY, CONCLUSIONS, RECOMMENDATIONS, AND FUTURE DIRECTIONS 97

9.1 Summary 97
9.2 Conclusions 98
9.3 Recommendations 99
9.4 Future Directions 100

LIST OF REFERENCES 101

Page

APPENDIX MSE-OPTIMAL WEIGHTS FOR LINEAR COMBINATIONS 110

VITA 116

LIST OF TABLES

Table		Page
4.1	Total MSEs of best trained NNs obtained after 1000 iterations	29
4.2	Total MSEs of U-OLC of three NNs trained for 1000 iterations	30
4.3	Total MSEs of U-OLC of nine NNs trained for 1000 iterations	31
4.4	Total MSEs of best trained NNs obtained after 100000 iterations	32
8.1	Part I: level 1: (a) Original algorithms. (b) Modified algorithms.	79
8.2	Part I: level 2 :(a) Original algorithms. (b) Modified algorithms.	82
8.3	Part I: level 3: (a) Original algorithms. (b) Modified algorithms.	85
8.4	Part I: level 4: (a) Original algorithms. (b) Modified algorithms.	86
8.5	Part II: level 1: (a) Original algorithms. (b) Modified algorithms.	88
8.6	Part II: level 2: (a) Original algorithms. (b) Modified algorithms.	92
8.7	Part II: level 3: (a) Original algorithms. (b) Modified algorithms.	93
8.8	Part II: level 4: (a) Original algorithms. (b) Modified algorithms.	95

LIST OF FIGURES

Figure		Page
1.1	Linear combination of the outputs of p trained neural networks	6
3.1	The function $r_1(X)$ and the approximations obtained using the unconstrained MSE-OLC and using NN4.	24
3.2	The function $r_1(X)$ and the approximations obtained using the simple averaging of the outputs of the six NNs and using NN3.	25
5.1	The function $r_2(x)$ and the approximation obtained using the NN. . . .	38
5.2	The first-order derivative $r_2'(x)$ and the approximation obtained using the NN.	38
5.3	The second-order derivative $r_2''(x)$ and the approximation obtained using the NN.	39
5.4	The first-order derivative $r_1'(X)$ and the approximations obtained using the unconstrained MSE-OLC and using NN3.	41
5.5	The first-order derivative $r_1'(X)$ and the approximations obtained using the simple averaging of the outputs of the six NNs and using NN4. . . .	42
5.6	The second-order derivative $r_1''(X)$ and the approximations obtained using the unconstrained MSE-OLC and using NN3.	43
5.7	The second-order derivative $r_1''(X)$ and the approximations obtained using the simple averaging of the outputs of the six NNs and using NN4. . . .	44
8.1	The function $r_3(X)$ and the approximations obtained using the best NN, the simple averaging, and U-OLC 2	83
8.2	The function $r_3(X)$ and the approximations obtained using the best NN, the simple averaging, and U-OLC 2	90

ABSTRACT

Neural network (NN) based modeling often involves trying multiple networks with different architectures, learning techniques, and training parameters in order to achieve "acceptable" model accuracy. Typically, one of the trained networks is chosen as "best," while the rest are discarded.

In this dissertation, using optimal linear combinations (OLCs) of the corresponding outputs of a number of NNs is proposed as an alternative to using a single network. Modeling accuracy is measured by mean squared error (MSE) with respect to the distribution of random inputs to the NNs. Optimality is defined by minimizing the MSE, with the resultant combination referred to as MSE-OLC.

MSE-OLCs are investigated for four cases: allowing (or not) a constant term in the combination and requiring (or not) the combination-weights to sum to one. In each case, deriving the MSE-OLC is straightforward and the optimal combination-weights are simple, requiring modest matrix manipulations.

In practice, the optimal combination-weights need to be estimated from observed data: observed inputs, the corresponding true responses, and the corresponding outputs of each component network. Given the data, computing the estimated optimal combination-weights is straightforward.

Collinearity among the outputs and/or the approximation errors of the component NNs sometimes degrades the generalization ability of the estimated MSE-OLC. To improve generalization in the presence of degrading collinearity, six algorithms for selecting subsets of the NNs for the MSE-OLC are developed and tested.

Several examples, including a real-world problem and an empirical study, are discussed. The examples illustrate the importance of addressing collinearity and demonstrate significant improvements in model accuracy as a result of employing MSE-OLCs supported by the NN selection algorithms.

1. INTRODUCTION

The objective of this dissertation is to present and evaluate the use of optimal linear combinations of a number of trained neural networks to integrate the knowledge acquired by the component networks, and hence improve model accuracy.

Optimal linear combinations (OLCs) of neural networks (NNs) are constructed by forming weighted sums of the corresponding outputs of the networks. The combination-weights are selected to minimize the mean squared error (MSE) with respect to the distribution of random inputs to the NNs, a criterion commonly used by both the statistics and the neural network communities. The resultant optimal linear combinations are referred to as MSE-OLCs.

Constructing MSE-OLCs is straightforward. Expressions for the (MSE-)optimal combination-weights are obtained in closed-form and require modest computational effort, mainly simple matrix manipulations. In practice, the optimal combination-weights need to be estimated from observed data, which makes the resultant MSE-OLC prone to data problems, especially collinearity (linear dependency).

Since the component NNs are trained to approximate the same physical quantity (or quantities), collinearity among the outputs and/or the approximation errors of the component networks can sometimes undermine the robustness (generalization ability) of the MSE-OLCs. Six algorithms are developed to improve the robustness of the resultant MSE-OLC by properly selecting the component NNs included in the combination. An empirical study is conducted to examine the merits of MSE-OLC and to evaluate the effectiveness of the six algorithms in improving robustness.

This dissertation focuses mainly on function approximation or regression problems. However, MSE-OLCs are also applicable to classification settings with predefined classes, since the decision to join a given class can be represented by the probability of belonging to that class. In such a setting, the classification problem can be formulated as a function approximation problem.
The dissertation is divided into two main parts:

The first part, which includes Chapters 2–5, investigates the merits of MSE-OLC. Four cases of MSE-OLCs are discussed and nine closed-form expressions for the optimal combination-weights are presented in Chapter 2. The estimation of the optimal combination-weights is discussed in Chapter 3. The MSE-OLC approach is employed in constructing a NN-based model that aids in solving a product allocation problem in Chapter 4. The impact of MSE-OLC on improving the accuracy of approximating a function and its derivatives is investigated in Chapter 5.

The second part, which includes Chapters 6–8, investigates the robustness of the MSE-OLCs. The ill effects of collinearity on the robustness of the MSE-OLCs are examined in Chapter 6. Six algorithms for improving the robustness of the MSE-OLC by the proper selection of the component networks, are proposed in Chapter 7. An empirical study that explores the merits of the MSE-OLCs and the effectiveness of the proposed algorithms in improving robustness, is presented in Chapter 8.

Beside the development of a theoretical framework for constructing MSE-OLCs of NNs, the main contribution of this dissertation is the introduction of a framework for testing and improving the robustness of MSE-OLCs.

In the rest of this chapter, the motivation behind this dissertation is discussed and the related literature is reviewed.

1.1 Motivation

1.1.1 Neural network based models

Artificial neural networks (NNs) are widely applied to a variety of practical problems [52, 96]. The areas of applications include control [75, 80, 83, 108], signal processing [61], pattern recognition [71, 73], forecasting [106], modeling chemical processes [11, 62, 78] and manufacturing processes [3, 104]. Many success stories have been reported. However, many concerns about the construction and the use of NN-based models have also been raised.

1.1.1.1 Model construction

The problem of constructing a NN-based model for a data-generating process may be defined by:

> Given a set of observed data, construct a NN-based model that *adequately* approximates the underlying (data-generating) process.

While several measures of the adequacy of the model can be used, the most widely used measure for function approximation and regression applications is the MSE.

1.1.1.2 Some design concerns

The complexity of the design phase, which includes selecting an "appropriate" network topology and an "efficient" training scheme, is a major obstacle that limits the use of NNs by many potential users [2, 28, 30, 54, 97]. There are many degrees of freedom in selecting the topological and the training parameters, which may significantly affect the accuracy of the resultant model as well as the training time. Parameter selection is often the result of rules of thumb combined with trial and error [44, 70]. At the end of the training process, a number of trained networks is produced, then typically one of them is chosen, based on some optimality criterion,

while the rest are discarded. The complexity of the design phase combined with the uncertainty in its outcome may lead to limited success in some applications [20] and sometimes even frustration [84].

The accuracy of NN-based models may vary dramatically depending on many factors, including the network topology and the learning technique. This dependency has been investigated in the literature by a number of researchers. For a learning algorithm such as Error Backpropagation, which is also known as the Generalized Delta Rule [87, 92, 107], the choice of the initial connection-weights for the NN may significantly affect the learning convergence [26, 64, 65]. The choice of the activation function(s) to use in a network may influence the learning speed [59]. NNs with sigmoid activation functions as well as those with Gaussian activation functions are capable of approximating unknown mappings arbitrary well under some mild conditions [56, 46]. However, this does not help in deciding which type of activation is more appropriate to use, for a given problem, in terms of the learning speed, the accuracy, and compactness of the resultant model. A mixture of different activation functions within the same NN may yield the best performance [29]. Similar concerns have been raised regarding the choice of network topology [23], and the learning technique [15, 58].

1.1.1.3 Automating parameter selection

Many approaches have been developed to automate the process of constructing NN-based models.

Harp et al. [45] use genetic algorithms [37] for designing application-specific NNs. They use a genetic algorithm to evolve appropriate network structures and values of learning parameters. Whitley et al. [110] discuss the use of genetic algorithms to optimize the connection-weights in feedforward neural networks, and to discover novel architectures in the form of connectivity patterns for neural networks that learn using error propagation. Schaffer et al. [93] survey techniques that are based on genetic algorithms to evolve network parameters and learning rules.

Fahlman and Lebiere [29] and Tenorio [101] present learning algorithms that construct the network topology during training, guided by a measure of performance.

Techniques such as cross-validation [44], incremental learning [1], and network pruning and weight decay [52, pages 156–158] often (at least partially) automate the process of obtaining a trained network by considering jointly the choice of architecture and method of training.

These approaches have substantially helped in providing solutions to the problem of constructing efficient NN-based models. However, constructing an "acceptable" model, as many optimization problems, may require setting up some initial conditions and additional local tuning. Whether this process is automated or performed interactively with the guidance of a human supervisor, often a number of trained NNs is produced. Typically, these NNs are scanned to select the best trained network based on some performance measure.

Instead of picking just a single NN as best, we propose combining the trained NNs using MSE-OLCs. Combining the trained NNs may help integrate the knowledge acquired by the component NNs, and thus improves the resultant model accuracy.

1.1.2 Why to combine neural networks?

In the literature on combining forecasts, Makridakis and Winkler [69] conclude:

> Thus, combining forecasts seems to be a reasonable practical alternative when, as often the case, a "true" model of the data-generating process or a single best forecasting method cannot be or is not, for whatever reason, identified.

This statement is equally applicable to combining NNs. As universal approximators [46, 56, 86], NNs are often employed with the understanding that they are "generic" modeling tools and that they belong to a class of non-parametric regression methods [33, 109]. In such cases, there is no reason to believe that one of the trained NNs represents the "true" model of the underlying (data-generating) process.

1.1.3 When to combine neural networks?

The need for constructing combinations of trained networks may arise in many situations, for instance:

- A situation where a variety of networks structures, activation functions, and/or learning parameter values is tried to come up with the best possible model.
- When the learning scheme used in constructing the NN-based model generates many trained networks, as the case of using genetic algorithms [45, 110]. In this case, training is started with a population of networks and new trained individuals (NNs) are produced in each generation. Thus, during training, many good individuals may be selected and stored for use in constructing MSE-OLCs.
- For complex problems, instead of constructing one large network, one may try constructing a number of smaller networks. These networks may be trained separately and then combined. In case of the existence of separate processors, training the networks in parallel helps in utilizing the available hardware resources efficiently. Running the networks may be performed in parallel as well.

In the first two situations, the selection of trained networks for constructing MSE-OLCs may require little or no additional computational costs. However, in the last situation, the trade-off between using one large network as opposed to using several small ones remains an open question, since there is a valid concern that the "small" component NNs may not be adequate for the modeling task. In any case, the computational cost associated with calculating the optimal combination-weights of the MSE-OLC is fairly modest, as shown in Chapters 2 and 3. Moreover, the robustness of the MSE-OLC may be easily tested, as discussed in Sections 6.1.2 and 6.6.2.

1.1.4 How to interpret combining neural networks?

From a NN perspective, combining the outputs of a number of trained NNs is similar to creating a large NN in which the trained NNs are subnetworks operating in parallel, and the combination-weights are the connection-weights of the output layer (Figure 1.1). For a given input, $\vec{x}$, the output of the combined model, $\widetilde{y}$, is the weighted sum of the corresponding outputs of the component NNs, $y_j, j = 1, \ldots, p$, and α_j's are the associated combination-weights. The fundamental difference between the two situations is that in the former situation (combining NNs), the connection-weights of the trained NNs are fixed and the combination-weights are computed by performing simple (fast) matrix manipulations, as discussed in Chapters 2 and 3. However, in case of training one large NN, there is a large number of parameters (weights) that need to be simultaneously estimated (trained). Thus, the training time may be longer, and also the risk of over-fitting to the data may be significant. Over-fitting to the training or combination data becomes a serious concern, especially when the number of parameters in the model becomes large compared to the cardinality of the data set used in estimating these parameters.

1.1.5 Why use linear methods for combining?

A question that immediately rises whenever linear methods are suggested is:

Why *linear*, aren't non-linear methods more general?

When combining NNs, one needs to keep in mind a fundamental issue: presumably all the NNs are trained to approximate the *same* physical quantity (or quantities). Thus, forming a weighted sum of the corresponding outputs of the NNs is directly comprehended. Beside their intuitive appeal, linear methods are often simpler to analyze and easier to implement than non-linear methods. The implementations of the four MSE-OLCs, discussed in Chapter 2, are straightforward and require modest computational effort that mainly involves a matrix inverse.

In the literature on combining forecasts, which is perhaps the most well developed and mature among the relevant literature, linear formulation is almost always adopted [14].

1.2 Literature Review on Combining Methods

The literature on combining methods is very rich and diverse. Clemen [18] traces the literature back to Laplace in 1818 [63]. Averaging a number of estimators is frequently compared to the individual estimators, and in many cases performs better [18, 33, 38].

Combining methods have been used in various modeling applications, and it would be a challenging task to survey all the literature on combining methods. In this section, a brief review on the relevant literature on combining methods is presented. Major points of resemblance, distinction, or dissension are highlighted. However,

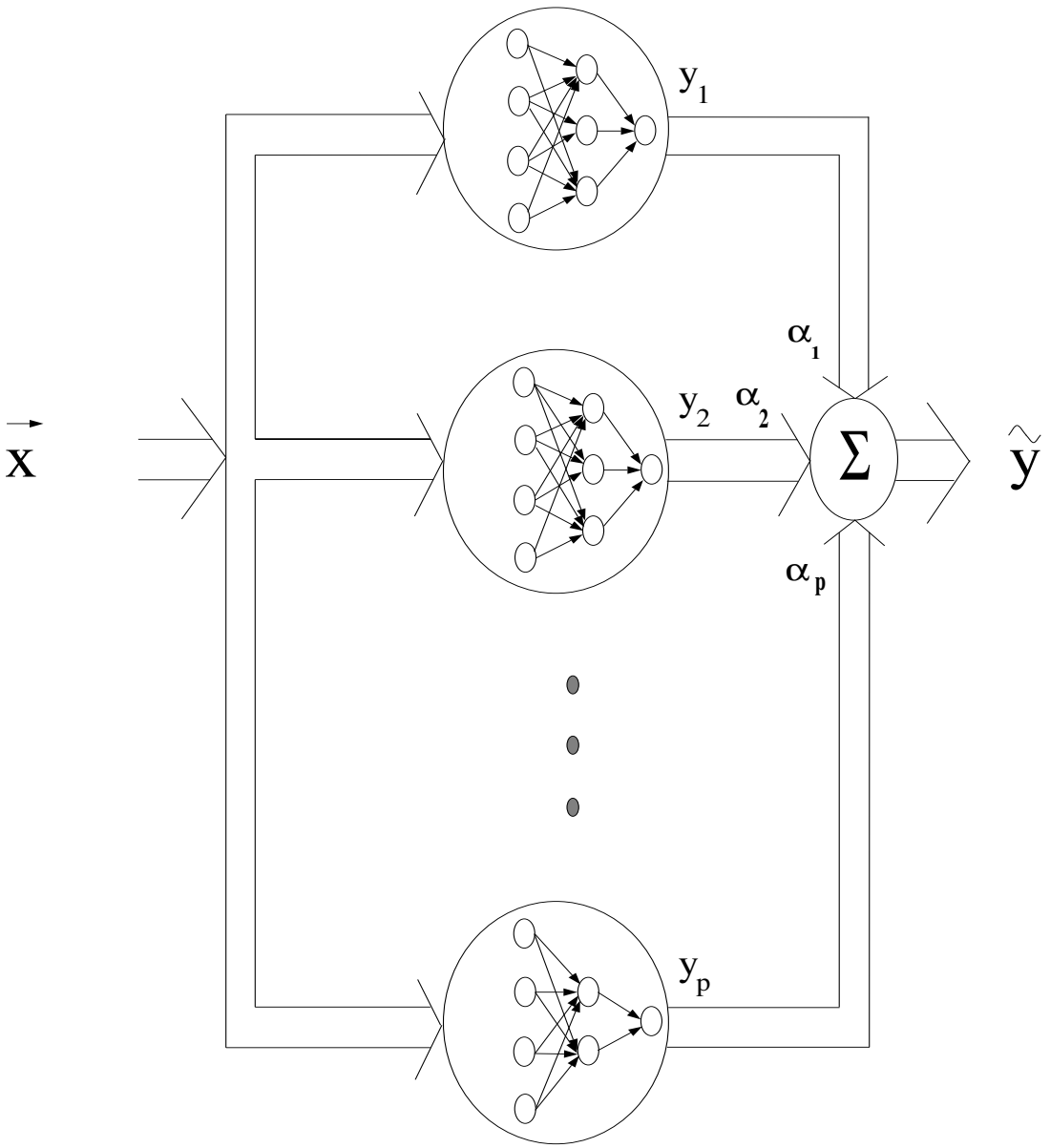

Figure 1.1 Linear combination of the outputs of p trained neural networks

many of the details are discussed in the appropriate sections of the dissertation, especially in Chapters 6 and 7.

1.2.1 Forecasting literature

Linear combinations of estimators have been used for over twenty-five years [38]. Clemen [18] cites more than 200 studies in his review of the literature related to combining forecasts, including contributions from forecasting, psychology, statistics, and management science literatures. These studies comprise over 2000 journal pages and 11 books, monographs, and theses. In 1989, twenty years after Bates and Granger [4] published their seminal paper entitled: "The Combination of Forecasts," the *Journal of Forecasting* dedicated a Special Issue on combining forecasts (Volume 8, Issue 3), and the *International Journal of Forecasting* dedicated a Special Section on combining forecasts (Volume 5, Issue 4).

The studies conducted by Bates and Granger [4] and Ried [90, 91] are considered to be the initial impetus for the current style of research in combining forecasts. Bates and Granger [4] derive an expression for combining pairs of forecasts. Their expression is extended by Ried [91] to the combination of several forecasts. Following Ried [91], if f is the $n \times 1$ vector of separate forecasts (unbiased), the optimal combined forecasts f_c is given as

$$f_c = (e^t S^{-1} f)/(e^t S^{-1} e), \qquad [1.1]$$

where e is the $n \times 1$ unit vector, S is the $n \times n$ forecast error covariance matrix. Equation 1.1 is similar to the MSE-OLC discussed in Section 2.4.3.2.

Granger and Ramanathan [39] show the equivalence between Equation 1.1 and constrained regression (discussed in Section 2.4.2). They also advocate the use of the unconstrained form (discussed in Section 2.4.1.1), and show that it (theoretically) yields a lower MSE.

Numerous empirical studies have been conducted to examine the effectiveness of combining forecasts in improving forecast accuracy (for example see [82]). Besides, two forecasting competitions, the "M-competition" and the "M2-competition", which compare the forecasting accuracy of major time series methods, demonstrate the merits of combining methods. In the M-competition[67], seven experts in each of 24 major time series methods forecast up to 1001 series for six up to eighteen time horizons. Among the methods tried in the competition are two methods based on combining individual forecasts: the simple averaging (equal combination-weights) and the weighted averaging. Both methods perform quite well and result in robust performance [67, pages 289–295]. The simple averaging has the best average ranking, with no other method even close in terms of average ranking. The M2-Competition [68] involves 29 actual series distributed to five forecasters who have had the chance to ask for additional information from the collaborating companies that are supplying the data. Two combining methods based on simple averaging are used. The results of the competition confirm that "combining of forecasters does better than the individual forecasters in the great majority of cases."

1.2.2 Neural networks literature

The idea of using a collection of trained NNs (or NN ensembles) instead of simply using the best NN has been proposed in the context of classification problems by Hansen and Salamon [44], Cooper [21], Mani [70], Baxt [5], Alpaydin [2], and Benediktsson et al. [10].

Hansen and Salamon [44] suggest training a group of NNs of the same architecture but initialized with different connection-weights. Then, a screened subset of the trained NNs is used in making the final classification decision by some voting scheme.

Cooper [21] suggests constructing a multi-neural network system in which a number of NNs independently compete to accomplish the required classification task. The multi-network system learns from experience which networks have the most effective separators and those determine the final classification.

Mani [70] suggests training a portfolio of NNs, of possibly different topologies, using a variety of learning techniques. He also sketches an approach for lowering the variance of the decision of the NNs using portfolio theory [72].

In the field of medical diagnosis, Baxt [5] trains two networks separately on data sampled from populations of different likelihoods in order to simplify the training and improve model accuracy.

Alpaydin [2] proposes building several possible models instead of one, train them independently on the same task, and take a vote over their responses.

Benediktsson et al. [10] propose a new architecture called the parallel consensual neural network that is based on statistical consensus theory. The architecture consists of several stage networks whose outputs are combined to make a decision.

In the absence of a "true" model, these methods employ several networks to solve the classification task independently. Then, a final vote is constructed by making use of the individual votes. The resultant improvement in the classification quality is achieved by avoiding classification error through the inclusion of more than one network in making the final decision.

Perrone [88] and Perrone and Cooper [89] present a general ensemble method (GEM) of constructing improved regression estimates. They employ a constrained MSE-optimal linear combination (similar to Equation 1.1) to combine a number of trained NNs. Perrone [88] demonstrates the efficiency of this ensemble method on three real-world classification and time series prediction tasks.

1.2.3 Other related literature

The literature on combining methods is fairly diverse and is scattered among numerous application areas. The literature on combining forecasts, discussed in Section 1.2.1 is perhaps the most comprehensive and mature. Other areas of application that are not mentioned above include combining probability distributions [34], stochastic simulation [36, 99], and estimation theory [41].

1.3 Related Publications

The following chapters extend research published by the author:

- Chapters 2 and 3 extend the two types of MSE-OLCs discussed in [48] by constructing two additional types of MSE-OLCs as well as providing alternate expressions.
- Chapter 4 contains an extended discussion on the product allocation problem in [50].
- Chapter 5 integrates the methods developed in [47, 49].

1.4 Remarks

Some important remarks regarding the discussions in this dissertation are:

- Unless stated otherwise, the mean squared error (MSE) is the measure for evaluating the performance of a single NN or a combination of NNs.
- The class of neural networks investigated in this dissertation is the class of multilayer feedforward networks [52, pages 115–162]. No further assumptions regarding the network architecture or the type of learning are required.
- To evaluate the effectiveness of an MSE-OLC, its performance is compared to those of the two most popular alternatives: the simple averaging (equal combination-weights) and the (apparent) best NN. While the former method, simple averaging, does not require any data for estimation, the latter method, best NN, does. The best NN to fit the training data is not necessarily the *true* best among a number of trained NNs. Unless the *true* function is known, a common practice is to test the trained NNs on a data set separate from the training data set in order to select the best performer (as discussed in Section 6.6.2).
- As indicated in Section 1.2, the literature on combining methods is scattered across numerous research fields, which makes unintentional replication or reinvention of results inevitable. By placing the literature review in Section 1.2 and by referencing related research, we hope to have had identified the major contributors to this rich literature. And most important, we hope to have avoided unnecessary repetitions.

2. OPTIMALLY COMBINING NEURAL NETWORKS

We define the general problem of combining a number of trained neural networks in Section 2.1. We then formulate the problem of constructing optimal linear combinations (OLCs) of neural networks in Sections 2.2 and 2.3. Optimality is defined by minimizing the mean squared error (MSE) with respect to the distribution of random inputs.

Closed-form expressions for the optimal combination-weights of four types of single-output MSE-OLCs are presented in Section 2.4. Convex MSE-OLCs are briefly discussed in Section 2.5. An approach to the multi-output MSE-OLC problem is presented in Section 2.6.

2.1 General Problem of Optimally Combining Neural Networks

Definition:

> Given p trained neural networks (NNs), the problem is to construct a function of the corresponding outputs of the NNs based on a given optimality criterion.

Although various functions can be used in combining the component NNs, for reasons discussed in Section 1.1.5, we focus on linear combinations.

The optimality criterion adopted in this dissertation is minimizing the mean squared error (MSE) with respect to the distribution of random inputs, a criterion commonly used by both the neural network and statistics communities.

2.2 Linear Combination of Neural Networks

The mapping being approximated — by the component networks — may be a multi-input-multi-output mapping. However, for the sake of clarity of the analysis and the derivations, we focus on constructing MSE-OLC of single-outputs at first. The more general multi-output MSE-OLC problem is discussed in Section 2.6.

A trained NN accepts a vector-valued input $\vec{x}$ and returns a scalar output (response) $y(\vec{x})$. The approximation error is $\delta(\vec{x}) = r(\vec{x}) - y(\vec{x})$, where $r(\vec{x})$ is the true answer (the response of the real system) for $\vec{x}$.

A linear combination of the outputs of p NNs returns the scalar output $\tilde{y}(\vec{x};\vec{\alpha}) = \sum_{j=1}^{p} \alpha_j y_j(\vec{x})$, with corresponding error $\tilde{\delta}(\vec{x};\vec{\alpha}) = r(\vec{x}) - \tilde{y}(\vec{x};\vec{\alpha})$; where $y_j(\vec{x})$ is the

output of the jth network, and α_j[1] is the combination-weight associated with $y_j(\vec{x})$; $j = 1, \ldots, p$. This definition of $\tilde{y}(\vec{x}; \vec{\alpha})$ may be extended to include a constant term, $\alpha_0 y_0(\vec{x})$, where $y_0(\vec{x}) = 1$. This term allows for correcting any (statistical) bias in $y_j(\vec{x})$; $j = 1, \ldots, p$. Thus, $\tilde{y}(\vec{x}; \vec{\alpha})$ is given by

$$\begin{aligned} \tilde{y}(\vec{x}; \vec{\alpha}) &= \sum_{j=0}^{p} \alpha_j y_j(\vec{x}) \\ &= \vec{\alpha}^t \vec{y}(\vec{x}) \,, \end{aligned} \qquad [2.1]$$

where $\vec{\alpha}$ and $\vec{y}(\vec{x})$ are $(p+1) \times 1$ vectors.

The problem is to find good values for the combination-weights $\alpha_0, \alpha_1, \ldots, \alpha_p$. One approach is to select one of the p networks as best, say NN_b, set $\alpha_b = 1$ and set the other combination-weights to zero. Using a single network has the advantage of simplicity, but the disadvantage of ignoring the (possibly) useful information in the other $p-1$ networks. Another approach, which is widely used by the forecasting community [18], is to use equal combination-weights (simple averaging). Simple averaging is straightforward but assumes that all the component networks are equally good.

2.3 Optimal Linear Combination (OLC) of Neural Networks

Think of the input $\vec{x}$ as an observation of a random variable $\vec{X}$ from a (usually unknown) multivariate distribution function $F_{\vec{X}}$. Then, the real response is the random variable $r(\vec{X})$, the output of the jth network is the random variable $y_j(\vec{X})$, and the associated approximation error is the random variable $\delta_j(\vec{X})$; $j = 1, \ldots, p$. The linear-combination output is the random variable $\tilde{y}(\vec{X}; \vec{\alpha}) = \sum_{j=0}^{p} \alpha_j y_j(\vec{X})$, and the linear-combination error is the random variable $\tilde{\delta}(\vec{X}; \vec{\alpha}) = r(\vec{X}) - \tilde{y}(\vec{X}; \vec{\alpha})$.

The optimal linear combination (OLC) is defined by the optimal combination-weights vector $\vec{\alpha}^* = (\alpha_0^*, \alpha_1^*, \ldots, \alpha_p^*)$ that minimizes the expected loss

$$\int_{\mathcal{S}} \ell(\tilde{\delta}(\vec{X}; \vec{\alpha}))\, dF_{\vec{X}} \,,$$

where $\mathcal{S}$ is the support of $F_{\vec{X}}$ and ℓ is a loss-function.

2.4 MSE-OLC of Neural Networks

Although various loss functions could be pursued, in this dissertation attention is restricted to squared-error loss, $\ell(\tilde{\delta}) = \tilde{\delta}^2$. The objective is then to minimize the mean squared error (MSE),

$$\mathrm{MSE}(\tilde{y}(\vec{X}; \vec{\alpha})) = \mathrm{E}\left[(\tilde{\delta}(\vec{X}; \vec{\alpha}))^2\right] \qquad [2.2]$$

[1] Notice that α_j's are not functions of $\vec{x}$.

where E denotes expected value with respect to $F_{\vec{X}}$. The resultant linear combination is referred to as the MSE-optimal linear combination (MSE-OLC). Thus, the MSE-OLC is defined by the optimal combination-weights vector $\vec{\alpha}^* = (\alpha_0^*, \alpha_1^*, \ldots, \alpha_p^*)$ that minimizes MSE$(\tilde{y}(\vec{X}; \vec{\alpha}))$.

Beside the general MSE-OLC problem described above, three special MSE-OLC problems are also considered. The variations among the four MSE-OLC problems are in the inclusion (or exclusion) of the constant term and/or constraining the sum of the combination-weights to unity. In practice, the decision to apply any of these four MSE-OLCs may be influenced by the desired properties of the MSE-OLC. However, the general MSE-OLC (theoretically) yields the minimal MSE amongst the four MSE-OLC forms.

In Section 2.4.1, the four MSE-OLC problems are formulated and expressions for the associated optimal combination-weights are presented. The expressions for the corresponding (minimal) MSE are also presented. In the three special MSE-OLC problems, the optimal combination-weights and the corresponding MSE are expressed in terms of those of the general MSE-OLC in addition to some extra (correction) terms.

The relation between the MSE-OLC and the ordinary least squares (OLS) regression is highlighted in Section 2.4.2. Thus, alternate expressions for optimal combination-weights of the three special MSE-OLCs are obtained.

Another alternate expressions for optimal combination-weights of the two constrained forms of the MSE-OLC are presented in Section 2.4.3.

2.4.1 MSE-OLC combination-weights

From the MSE-OLC problem in Section 2.4, three other MSE-OLCs problems are derived and considered. The variations among the four MSE-OLC problems are in the inclusion (or exclusion) of the constant term, $\alpha_0 y_0(\vec{X})$, defined in Section 2.2, and/or constraining the combination-weights; $\alpha_j, j = 1, \ldots, p$; to sum to one.

The inclusion of the constant term helps in correcting for (possible) bias in the component NNs. Constraining the combination-weights to sum to one, which is referred to as *weighted average*, may be desirable in some applications since the outputs of the NNs, $y_j(\vec{X})$; $j = 1, \ldots, p$, are approximating the same quantity $r(\vec{X})$.

2.4.1.1 Unconstrained MSE-OLC with a constant term

Consider the problem

$$\text{P1}: \quad \min_{\vec{\alpha}} \text{MSE}(\tilde{y}(\vec{X}; \vec{\alpha})).$$

Differentiating the MSE with respect to $\vec{\alpha}$ leads to the unconstrained optimal combination-weights vector,

$$\vec{\alpha}_{(1)} = \Psi^{-1}\vec{U}\,, \qquad [2.3]$$

where $\Psi = [\psi_{ij}] = [\mathrm{E}\langle y_i(\vec{X})\, y_j(\vec{X})\rangle]$ is a $(p+1) \times (p+1)$ matrix, and $\vec{U} = [u_i] = [\mathrm{E}\langle r(\vec{X})\, y_i(\vec{X})\rangle]$ is a $(p+1) \times 1$ vector. The corresponding (minimal) MSE is

$$\mathrm{MSE}^{(1)} = \mathrm{E}(r^2(\vec{X})) - \vec{U}^t\, \Psi^{-1}\, \vec{U}. \qquad [2.4]$$

Equations 2.3 and 2.4 are derived in the appendix.

2.4.1.2 Constrained MSE-OLC with a constant term

Consider the case of constraining the combination-weights, $\alpha_1, \ldots, \alpha_p$ to sum to one

$$\mathrm{P2}: \quad \min_{\vec{\alpha}} \mathrm{MSE}(\tilde{y}(\vec{X}; \vec{\alpha})), \qquad \text{s.t. } \vec{\alpha}^t\, \vec{1}_z = 1\,,$$

where $\vec{1}_z$ is a vector of proper dimension with the first component equals to zero and the remaining components equal to 1.

Solving the Lagrangian equivalent of P2 leads to the optimal combination-weights vector

$$\vec{\alpha}_{(2)} = \vec{\alpha}_{(1)} - \beta_{(2)}\, \Psi^{-1}\, \vec{1}_z\,, \qquad [2.5]$$

where

$$\beta_{(2)} = \frac{-1 + \vec{1}_z^t\, \Psi^{-1}\, \vec{U}}{\vec{1}_z^t\, \Psi^{-1}\, \vec{1}_z}\,.$$

The corresponding (minimal) MSE is

$$\mathrm{MSE}^{(2)} = \mathrm{MSE}^{(1)} + \beta_{(2)}^2\, \vec{1}_z^t\, \Psi^{-1}\, \vec{1}_z\,. \qquad [2.6]$$

The second term in the expression of $\mathrm{MSE}^{(2)}$ can be easily shown to be non-negative. It reflects the cost (increase in MSE) as a result of constraining the sum of the combination-weights to unity. Equations 2.5 and 2.6 are derived in the appendix.

2.4.1.3 Unconstrained MSE-OLC without a constant term

Consider the problem

$$\mathrm{P3}: \quad \min_{\vec{\alpha}} \mathrm{MSE}(\tilde{y}(\vec{X}; \vec{\alpha})), \qquad \text{s.t. } \vec{\alpha}^t\, \vec{\vartheta}_z = 0\,,$$

where $\vec{\vartheta}_z$ is a vector of proper dimension with the first component equals to 1 and the remaining components equal to zero.

Solving the Lagrangian equivalent of P3 leads to the optimal combination-weights vector

$$\vec{\alpha}_{(3)} = \vec{\alpha}_{(1)} - \beta_{(3)}\, \Psi^{-1}\, \vec{\vartheta}_z\,, \qquad [2.7]$$

where

$$\beta_{(3)} = \frac{\vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{U}}{\vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{\vartheta}_z}\,.$$

The corresponding (minimal) MSE is

$$\mathrm{MSE}^{(3)} = \mathrm{MSE}^{(1)} + \beta_{(3)}^2\ \vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{\vartheta}_z\,. \qquad [2.8]$$

The second term in the expression of $\mathrm{MSE}^{(3)}$ can be easily shown to be non-negative. It reflects the cost (increase in MSE) as a result of not using the constant term, α_0, in the combination. Equations 2.7 and 2.8 are derived in the appendix.

2.4.1.4 Constrained MSE-OLC without a constant term

In this case, α_0 is set to zero and the sum of the remaining combination-weights is constrained to unity. Hence, if the y_j's are unbiased (in a statistical sense), then $\tilde{y}$ will also be unbiased.

$$\mathrm{P4}: \quad \min_{\vec{\alpha}}\ \mathrm{MSE}(\tilde{y}(\vec{X};\vec{\alpha})), \qquad \text{s.t. } \vec{\alpha}^{\,t}\,\vec{\vartheta}_z = 0, \text{and } \vec{\alpha}^{\,t}\,\vec{1}_z = 1\,.$$

Solving the Lagrangian equivalent of P4 leads to the optimal combination-weights vector

$$\vec{\alpha}_{(4)} = \vec{\alpha}_{(1)} - \beta_{(4a)}\ \Psi^{-1}\ \vec{\vartheta}_z - \beta_{(4b)}\ \Psi^{-1}\ \vec{1}_z\,, \qquad [2.9]$$

where

$$\beta_{(4a)} = \frac{\vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{U} - \beta_{(4b)}\ \vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{1}_z}{\vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{\vartheta}_z}$$

and

$$\beta_{(4b)} = \frac{(\vec{1}_z^{\,t}\ \Psi^{-1}\ \vec{U} - 1)(\vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{\vartheta}_z) - (\vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{U})(\vec{1}_z^{\,t}\ \Psi^{-1}\ \vec{\vartheta}_z)}{(\vec{1}_z^{\,t}\ \Psi^{-1}\ \vec{1}_z)(\vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{\vartheta}_z) - (\vec{\vartheta}_z^{\,t}\ \Psi^{-1}\ \vec{1}_z)^2}\,.$$

The corresponding (minimal) MSE is

$$\mathrm{MSE}^{(4)} = \mathrm{MSE}^{(1)} + \beta_{(4a)}^2\,(\vec{\vartheta}_z^{\,t}\,\Psi^{-1}\,\vec{\vartheta}_z) + \beta_{(4b)}^2\,(\vec{1}_z^{\,t}\,\Psi^{-1}\,\vec{1}_z) + 2\,\beta_{(4a)}\,\beta_{(4b)}(\vec{\vartheta}_z^{\,t}\,\Psi^{-1}\,\vec{1}_z). \quad [2.10]$$

Each of the three last terms in the expression of $\mathrm{MSE}^{(4)}$ can be easily shown to be non-negative. Thus, their sum reflects the cost (increase in MSE) of not using the constant term, α_0, and at the same time restricting the remaining combination-weights to sum to unity. Equations 2.9 and 2.10 are derived in the appendix.

2.4.2 MSE-OLC and ordinary least squares regression

The MSE-OLC problems in Section 2.4.1 are equivalent to ordinary least squares (OLS) regression. This equivalence is intuitive and provides alternate expressions for the optimal combination-weights in the three MSE-OLC problems described in

Sections 2.4.1.2–2.4.1.4. It also facilitates studying the properties of the estimators of the combination-weights, as discussed in Chapter 3.

The MSE-OLC problem P1 in Section 2.4.1.1 is equivalent to regressing $r(\vec{X})$ against $y_i(\vec{X}); i = 1, \ldots, p$, with an intercept term. Thus, the optimal combination-weights in Equation 2.3 are equal to the ordinary least squares (OLS) regression coefficients. Problem P3 is equivalent to the same OLS problem but with no intercept term. Hence, the optimal combination-weights may be (alternatively) obtained using

$$\vec{\alpha}_{(3)} = \acute{\Psi}^{-1} \acute{\vec{U}} \,, \qquad [2.11]$$

where $\acute{\Psi} = [\acute{\psi}_{ij}] = [\mathrm{E}\langle y_i(\vec{X})\, y_j(\vec{X})\rangle], (i, j > 0)$, is a $p \times p$ matrix, and $\acute{\vec{U}} = [\acute{u}_i] = [\mathrm{E}\langle r(\vec{X})\, y_i(\vec{X})\rangle], (i > 0)$, is a $p \times 1$ vector.

Similarly, the constrained MSE-OLC problem in Section 2.4.1.2, P2, is equivalent to regressing $(r(\vec{X}) - y_c(\vec{X}))$ against $(y_i(\vec{X}) - y_c(\vec{X}))$, for some $c \in \{1, \ldots, p\}$; $i = 1, \ldots, p$, $i \neq c$; with an intercept term. Thus, the constrained optimal-weights vector may be (alternatively) obtained using

$$\vec{\alpha}_{(2)} = \acute{\Psi}^{-1} \acute{\vec{U}} \,, \qquad [2.12]$$

where $\acute{\Psi} = [\acute{\psi}_{ij}] = [\mathrm{E}\langle (y_i(\vec{X}) - \mathcal{I}_{(i>0)}\, y_c(\vec{X}))\,(y_j(\vec{X}) - \mathcal{I}_{(j>0)}\, y_c(\vec{X}))\rangle]$ is a $p \times p$ matrix, $\mathcal{I}_{(i>0)}$ is an indicator variable that is equal to unity for $(i > 0)$ and to zero otherwise, and $\acute{\vec{U}} = [\acute{u}_i] = [\mathrm{E}\langle (r(\vec{X}) - y_c(\vec{X}))\,(y_i(\vec{X}) - \mathcal{I}_{(i>0)}\, y_c(\vec{X}))\rangle]$ is a $p \times 1$ vector. Problem P4 is equivalent to the same OLS problem but with no intercept term. Hence, the optimal combination-weights may be (alternatively) obtained using

$$\vec{\alpha}_{(4)} = \acute{\Psi}^{-1} \acute{\vec{U}} \,, \qquad [2.13]$$

where $\acute{\Psi} = [\acute{\psi}_{ij}] = [\mathrm{E}\langle (y_i(\vec{X}) - y_c(\vec{X}))\,(y_j(\vec{X}) - y_c(\vec{X}))\rangle], (i, j > 0)$, is a $(p-1) \times (p-1)$ matrix, and $\acute{\vec{U}} = [\acute{u}_i] = [\mathrm{E}\langle (r(\vec{X}) - y_c(\vec{X}))\,(y_i(\vec{X}) - y_c(\vec{X}))\rangle], (i > 0)$, is a $(p-1) \times 1$ vector.

2.4.3 Alternate expressions for the constrained MSE-OLC combination-weights

In Sections 2.4.1.2 and 2.4.1.4, the constrained optimal combination-weights are expressed as functions of the outputs of the NNs, $y_i(\vec{X}); i = 1, \ldots, p$. Alternatively, for Problems P2 and P4, the optimal combination-weights may be expressed in terms of the neural networks approximation errors, $\delta_j(\vec{X})$; $j = 1, \ldots, p$.

2.4.3.1 Constrained MSE-OLC with a constant term

From Equation 2.2, the constrained MSE-OLC, P2, is equivalent to:

$$\mathrm{P2}' : \quad \min_{\vec{\alpha}} \mathrm{E}\left[(\tilde{\delta}(\vec{X}; \vec{\alpha}))^2\right], \qquad \text{s.t. } \vec{\alpha}^t\, \vec{1}_z = 1 \,.$$

where

$$\begin{aligned}\tilde{\delta}(\vec{X};\vec{\alpha}) &= r(\vec{X}) - \tilde{y}(\vec{X};\vec{\alpha}) \\ &= r(\vec{X})(\vec{\alpha}^t\,\vec{1}_z) - \vec{\alpha}^t\,\vec{y}(\vec{X}) \\ &= \vec{\alpha}^t\,(r(\vec{X})\,\vec{1}_z - \vec{y}(\vec{X})) \\ &= \vec{\alpha}^t\,\vec{\delta}(\vec{X}),\end{aligned}$$

$\vec{\delta}(\vec{X})$ is a $(p+1)\times 1$ vector, with $\delta_0(\vec{X}) = -y_0(\vec{X}) = -1$, and $\delta_i(\vec{X}) = r(\vec{X}) - y_i(\vec{X});\ i = 1,\ldots,p$.

Solving the Lagrangian equivalent of P2′ leads to the optimal combination-weights vector

$$\vec{\alpha}_{(2)} = \frac{\Omega^{-1}\,\vec{1}_z}{\vec{1}_z^t\,\Omega^{-1}\,\vec{1}_z}\,, \qquad [2.14]$$

where $\Omega = [\omega_{ij}] = [\mathrm{E}\langle\delta_i(\vec{X})\,\delta_j(\vec{X})\rangle]$ is a $(p+1)\times(p+1)$ matrix. The corresponding (minimal) MSE is

$$\mathrm{MSE}^{(2)} = \frac{1}{\vec{1}_z^t\,\Omega^{-1}\,\vec{1}_z}\,. \qquad [2.15]$$

Equations 2.14 and 2.15 are derived in the appendix.

2.4.3.2 Constrained MSE-OLC without a constant term

From Equation 2.2, the constrained MSE-OLC problem P4 is equivalent to

$$\mathrm{P4}'\!: \quad \min_{\vec{\alpha}}\ \mathrm{E}\left[(\tilde{\delta}(\vec{X};\vec{\alpha}))^2\right], \qquad \text{s.t. } \vec{\alpha}^t\,\vec{\vartheta}_z = 0 \text{ and } \vec{\alpha}^t\,\vec{1}_z = 1\,.$$

Solving the Lagrangian equivalent of P4′ leads to the optimal combination-weights vector

$$\vec{\alpha}_{(4)} = (\,0,\ \vec{\alpha''}^{\,t}_{(4)}\,)^t\,, \qquad [2.16]$$

where

$$\vec{\alpha''}_{(4)} = \frac{\Omega''^{-1}\,\vec{1}}{\vec{1}^t\,\Omega''^{-1}\,\vec{1}}\,,$$

$\vec{1}$ is a vector of proper dimension with all components equal 1, and $\Omega'' = [\omega''_{ij}] = [\mathrm{E}\langle\delta_i(\vec{X})\,\delta_j(\vec{X})\rangle], (i,j>0)$, is a $p\times p$ matrix. The corresponding (minimal) MSE is

$$\mathrm{MSE}^{(4)} = \frac{1}{\vec{1}^t\,\Omega''^{-1}\,\vec{1}}\,. \qquad [2.17]$$

Equations 2.16 and 2.17 are derived in the appendix.

2.5 Convex MSE-OLC

In the four MSE-OLC problems discussed in Section 2.4, the signs of the optimal combination-weights are not restricted. Although allowing the combination-weights to assume positive or negative values permits the maximal reduction in MSE, in some contexts having negative combination-weights may be undesirable [4, 13, 19].

Even in the two constrained MSE-OLC problems, P2 and P4, in Sections 2.4.1.2 and 2.4.1.4), negative combination-weights make the combination non-convex, and thus allow the combined model to produce an output outside the range of the p networks. Moreover, as Bunn [13] points out, large positive and negative combination-weights can occur, which obviously require careful estimation if the theoretical advantage of the MSE-OLC is to be obtained in practice.

In the context of combining forecasts, Clemen and Winkler [19] suggest constraining the combined forecast to fall between the lowest and highest forecasts. A combined forecast lower (higher) than the lowest (highest) forecast can be modified by setting it equal to the lowest (highest) forecast.

A more formal, yet more computationally demanding, solution to the non-convexity problem is to constrain the combination-weights to lie between zero and one. However, in this case there may not be a closed-form expression for the optimal combination-weights, but they can be computed numerically.

2.6 Multi-Output MSE-OLC Problem

One approach to the multi-output case is to compute an optimal combination-weights vector for each output separately. Such approach is straightforward and minimizes the total MSE for multi-input-multi-output mappings, and thus may be adequate in many applications. This approach is adopted in Chapter 4 where each component network has six inputs and four outputs.

In other contexts, a multivariate analysis may be more appropriate, but is not considered here.

3. ESTIMATING MSE-OLC COMBINATION-WEIGHTS

In Chapter 2, several expressions for MSE-OLC combination-weights are presented. These expressions are based on expected values taken with respect to the multivariate distribution function $F_{\vec{x}}$ of the inputs to the neural networks. In practice, one seldom knows $F_{\vec{x}}$. Thus, $\Psi, \vec{U}, \acute{\Psi}, \acute{\vec{U}}, \Omega,$ and Ω'' in Equations 2.3–2.17 need to be estimated.

In this chapter, the problem of estimating the MSE-OLC combination-weights is treated. The use of ordinary least squares estimators (OLS) for estimating the MSE-OLC combination-weights is discussed. Expressions for computing OLS estimates of the MSE-OLC combination-weights for the four MSE-OLC problems in Section 2.4.1 are presented.

Except for P1 — the unconstrained MSE-OLC with a constant term problem — the MSE-OLC combination-weights expressions given in Section 2.4.1 will not be used as bases for estimating the combination-weights. While these expressions provide valuable insight about the relationships between the the optimal combination-weights and the corresponding MSE, for a given case, and those of P1 which (theoretically) yields the minimal MSE, they are not as efficient computationally as the expressions based on the OLS estimators or the alternate expressions in Section 2.4.3. For the two constrained MSE-OLC problems, P2 and P4, estimators based on the alternate expressions of the optimal combination-weights given in Sections 2.4.3.1 and 2.4.3.2 are presented. An example that illustrates significant improvement in approximation accuracy as a result of using MSE-OLCs is also presented.

3.1 MSE-OLC Combination-Weights Estimation Problem

Definition:

> Given a set $\mathcal{K}$ of observed data, estimate the MSE-OLC combination-weights, where $\mathcal{K} = \{k_j : k_j = (\vec{x_j}, r(\vec{x_j}), \vec{y}(\vec{x_j})),\ j = 1, \ldots, \kappa\}$ and $\vec{x_j}$'s are independently sampled from $F_{\vec{x}}$.

3.2 Ordinary Least Squares Estimators of the MSE-OLC Combination-Weights

From Section 2.4.2, the equivalence relation between the MSE-OLC combination-weights and the OLS regression coefficients allows the use of the OLS estimators

in estimating the MSE-OLC combination-weights. OLS estimators are also used by Granger and Ramanathan [39] in estimating the optimal combination-weights for combining forecasts. The analysis of the OLS estimators is straightforward and may provide quality measures for the MSE-OLC, as discussed in Section 3.4.

3.2.1 Unconstrained MSE-OLC with a constant term

For the MSE-OLC Problem P1 in Section 2.4.1.1, the equivalent regression model is

$$r(\vec{X}) = \alpha_0 + \sum_{j=1}^{p} \alpha_j \, y_j(\vec{X}) + \varepsilon \,, \qquad [3.1]$$

where ε is a random error with zero mean and variance σ^2.

The OLS estimator of $\vec{\alpha}_{(1)}$ is

$$\widehat{\vec{\alpha}_{(1)}} = \hat{\Psi}^{-1} \hat{\vec{U}} \,, \qquad [3.2]$$

where

$$\hat{\Psi} = [\widehat{\psi_{ij}}] = [\sum_{k=1}^{\kappa} (y_i(\vec{X}_k) \, y_j(\vec{X}_k))/\kappa]$$

and

$$\hat{\vec{U}} = [\widehat{u_i}] = [\sum_{k=1}^{\kappa} (r(\vec{X}_k) \, y_i(\vec{X}_k))/\kappa].$$

Assuming that the observed errors, ε_k, are uncorrelated, $\widehat{\vec{\alpha}_{(1)}}$ is unbiased and have the minimum variance among all unbiased estimators of $\vec{\alpha}_{(1)}$ [81, page 39].

Moreover, assuming that ε is normally distributed, then $\widehat{\vec{\alpha}_{(1)}}$ has a multivariate normal distribution with mean $\vec{\alpha}_{(1)}$ and covariance matrix $\sigma^2 \, \Psi^{-1}$. An estimate of the covariance matrix of $\widehat{\vec{\alpha}_{(1)}}$ may be obtained by substituting $\hat{\Psi}$ for Ψ, and substituting the unbiased estimator of σ^2, namely the MSE. Hence, one can obtain estimates of the standard deviations of the estimates of the MSE-OLC combination-weights as well as estimates of their pairwise correlations. Also joint confidence regions and tests of statistical significance for $\widehat{\vec{\alpha}_{(1)}}$ can be easily constructed [81, pages 238–247].

Descriptive measures of the association between $r(\vec{X})$ and $\tilde{y}(\vec{X})$, such as the coefficient of multiple determination, $\mathcal{R}^2$, and the adjusted coefficient of multiple determination, $\mathcal{R}_a^2$, may be computed for the combined model [81, page 241]. $\mathcal{R}^2$ measures the proportionate reduction in the total variation in $r(\vec{X})$ associated with the use of the set of variables $y_j(\vec{X}), j = 1, \ldots, p$, while $\mathcal{R}_a^2$ is an adjustment to $\mathcal{R}^2$ that penalizes for the excessive use of regression parameters in the model.

The normality assumption for ε is justifiable in many situations, because the error term often represents the effects of factors missing from the model as well as (additive) measurement noise associated with $r(\vec{X})$. These random effects have a degree of mutual independence and hence the composite error term ε representing all

these factors would tend to comply with the central limit theorem, and thus the error term distribution would approach normality [81, pages 49,70].

The above discussion also applies to the OLS estimators of $\vec{\alpha}_{(2)}$, $\vec{\alpha}_{(3)}$, and $\vec{\alpha}_{(4)}$ presented in Section 3.2.2–3.2.4.

3.2.2 Constrained MSE-OLC with a constant term

For the MSE-OLC Problem P2 in Section 2.4.1.2, the equivalent regression model is

$$\acute{r}(\vec{X}) = \alpha_0 + \sum_{j=1, j\neq c}^{p} \alpha_j \, \acute{y}_j(\vec{X}) + \varepsilon, \qquad [3.3]$$

where $\acute{r}(\vec{X}) = r(\vec{X}) - y_c(\vec{X})$, $\acute{y}_j(\vec{X}) = y_j(\vec{X}) - y_c(\vec{X})$, for some $c \in \{1, \ldots, p\}$, $j = 1, \ldots, p, j \neq c$.

The OLS estimator of $\vec{\alpha}_{(2)}$ is

$$\widehat{\vec{\alpha}_{(2)}} = \hat{\acute{\Psi}}^{-1} \hat{\vec{U}}, \qquad [3.4]$$

where

$$\hat{\acute{\Psi}}^{-1} = [\widehat{\acute{\psi}_{ij}}] = [\sum_{k=1}^{\kappa} (\acute{y}_i(\vec{X}_k) \, \acute{y}_j(\vec{X}_k))/\kappa],$$

$\acute{y}_0(\vec{X}_k) = 1$, and

$$\hat{\vec{U}} = [\widehat{\acute{u}_i}] = [\sum_{k=1}^{\kappa} (r(\vec{X}_k) \, \acute{y}_i(\vec{X}_k))/\kappa].$$

3.2.3 Unconstrained MSE-OLC without a constant term

For the MSE-OLC Problem P3 in Section 2.4.1.3, the equivalent regression model is

$$r(\vec{X}) = \sum_{j=1}^{p} \alpha_j \, y_j(\vec{X}) + \varepsilon. \qquad [3.5]$$

The OLS estimator of $\vec{\alpha}_{(3)}$ is

$$\widehat{\vec{\alpha}_{(3)}} = \hat{\acute{\Psi}}^{-1} \hat{\vec{U}}, \qquad [3.6]$$

where

$$\hat{\acute{\Psi}} = [\widehat{\acute{\psi}_{ij}}] = [\sum_{k=1}^{\kappa} (y_i(\vec{X}_k) \, y_j(\vec{X}_k))/\kappa], (i, j > 0),$$

and

$$\hat{\vec{U}} = [\widehat{\acute{u}_i}] = [\sum_{k=1}^{\kappa} (r(\vec{X}_k) \, y_i(\vec{X}_k))/\kappa], (i > 0).$$

3.2.4 Constrained MSE-OLC without a constant term

For the MSE-OLC Problem P4 in Section 2.4.1.4, the equivalent regression model is

$$\acute{r}(\vec{X}) = \sum_{j=1, j\neq c}^{p} \alpha_j \, \acute{y}_j(\vec{X}) + \varepsilon \,, \qquad [3.7]$$

where $\acute{r}(\vec{X}) = r(\vec{X}) - y_c(\vec{X})$, $\acute{y}_j(\vec{X}) = y_j(\vec{X}) - y_c(\vec{X})$, for some $c \in \{1, \ldots, p\}$, $j = 1, \ldots, p, j \neq c$.

The OLS estimator of $\vec{\alpha}_{(4)}$ is

$$\widehat{\vec{\alpha}_{(4)}} = \hat{\acute{\Psi}}^{-1} \hat{\vec{U}} \,, \qquad [3.8]$$

where

$$\hat{\acute{\Psi}}^{-1} = [\widehat{\acute{\psi}_{ij}}] = [\sum_{k=1}^{\kappa} (\acute{y}_i(\vec{X}_k) \, \acute{y}_j(\vec{X}_k))/\kappa], (i, j > 0),$$

and

$$\hat{\vec{U}} = [\widehat{\acute{u}_i}] = [\sum_{k=1}^{\kappa} (r(\vec{X}_k) \, \acute{y}_i(\vec{X}_k))/\kappa], (i > 0).$$

3.3 Alternate Estimators for the Constrained MSE-OLC Combination-Weights

For reasons discussed in Section 6.3, we also consider alternate estimators for the optimal combination-weights of the constrained MSE-OLCs. These estimators are based on the expressions presented in Section 2.4.3.

3.3.1 Constrained MSE-OLC with a constant term

For the constrained MSE-OLC Problem P2 in Section 2.4.1.2, and based on the expression for MSE-OLC combination-weighs presented in Section 2.4.3.1, one may estimate the MSE-OLC combination-weights using

$$\widehat{\vec{\alpha}_{(2)}} = \frac{\hat{\Omega}^{-1} \, \vec{1}_z}{\vec{1}_z^t \, \hat{\Omega}^{-1} \, \vec{1}_z} \,, \qquad [3.9]$$

where

$$\hat{\Omega} = [\widehat{\omega_{ij}}] = [\sum_{k=1}^{\kappa} (\delta_i(\vec{X}_k) \, \delta_j(\vec{X}_k))/\kappa].$$

3.3.2 Constrained MSE-OLC without a constant term

For the constrained MSE-OLC Problem P4 in Section 2.4.1.4, and based on the expression for MSE-OLC combination-weighs presented in Section 2.4.3.2, one may estimate the MSE-OLC combination-weights using

$$\widehat{\vec{\alpha}_{(4)}} = (\,0,\ \widehat{\vec{\alpha''}^{t}_{(4)}}\,)^t, \qquad [3.10]$$

where

$$\widehat{\vec{\alpha''}_{(4)}} = \frac{\widehat{\Omega''}^{-1}\,\vec{1}}{\vec{1}^{\,t}\,\widehat{\Omega''}^{-1}\,\vec{1}}\ ,$$

and

$$\widehat{\Omega''} = [\widehat{\omega''_{ij}}] = [\sum_{k=1}^{\kappa}(\delta_i(\vec{X}_k)\,\delta_j(\vec{X}_k))/\kappa].$$

3.4 Example 1

Consider the problem of approximating the single-input-single-output function

$$r_1(X) = 0.02\,(12 + 3X - 3.5X^2 + 7.2X^3)\,(1 + \cos 4\pi X)\,(1 + 0.8\sin 3\pi X),$$

over the interval $[0, 1]$, reported in [79]. The range of $r_1(X)$ is [0, 0.9).

We use three 2-hidden-layers NNs with 5 hidden units in each hidden layer (NN1, NN2, and NN3); and three 1-hidden-layer NNs with 10 hidden units (NN4, NN5, and NN6). Each network has one input unit and one output unit. The activation function for the hidden units as well as the output units is the logistic sigmoid function $g(s) = (1+e^{-s})^{-1}$. The networks are initialized with independent random connection-weights uniformly distributed in [–0.3, 0.3], then trained using the Error-Backpropagation algorithm. A set of 200 independent uniformly distributed points is used in training all the networks and in estimating the optimal combination-weights as well. Except for the structural differences and the different initial connection-weights, the six networks are trained in the same manner.

Using Equation 3.2 yields an estimated unconstrained optimal combination-weights vector $(0.0003, 0.125, -0.195, 0.639, 0.781, -0.665, 0.315)^t$, with an estimated standard deviation vector $(0.0002, 0.031, 0.024, 0.020, 0.031, 0.077, 0.072)^t$. Thus, the constant term does not appear to be statistically significant (at a 0.10 level of significance) since the associated two-sided P-value [81, page 12] is about 0.15. However, the other combination-weights are statistically significant with associated two-sided P-values less than 0.001.

Using Equation 3.6, which omits the constant, yields the estimated unconstrained optimal combination-weights vector $(0., 0.126, -0.195, 0.639, 0.779, -0.660, 0.312)^t$, with an estimated standard deviation vector $(0., 0.031, 0.024, 0.020, 0.031, 0.077, 0.072)^t$. $\mathcal{R} = \mathcal{R}_a = .9999$, which indicates that the combined model reduces the total variation in $r_1(X)$ to almost zero.

Similarly, using Equation 3.4 yields an estimated constrained optimal combination-weights vector $(0.0003, 0.125, -0.194, 0.638, 0.781, -0.665, 0.315)^t$, with an estimated standard deviation vector $(0.0002, 0.031, 0.024, 0.020, 0.031, 0.077, 0.072)^t$. Thus, the constant term does not appear to be statistically significant (at a 0.10 level of significance) since the associated two-sided P-value is about 0.15. However, the other

combination-weights are statistically significant with associated two-sided P-values less than 0.001.

Using Equation 3.8, which omits the constant, yields an estimated constrained optimal combination-weights vector $(0., 0.128, -0.197, 0.640, 0.777, -0.657, 0.309)^t$, with an estimated standard deviation vector $(0., 0.031, 0.024, 0.020, 0.031, 0.077, 0.072)^t$. $\mathcal{R} = \mathcal{R}_a = .9999$, which indicates that the combined model reduces the total variation in $r_1(X)$ to almost zero.

The unconstrained MSE-OLC results in a *true*[1] MSE of 0.000018; 87% less than that produced by NN4, the true best NN to approximate $r_1(X)$; and 95% less than that of the simple averaging of the outputs of the NNs. For the constrained MSE-OLC, the above results differ by about 1–3%. Such a small difference arises because unconstrained optimal combination-weights for accurate component networks (such as seen in this example) tend to automatically sum to one, with the constant term not being statistically significant. For less-accurate component networks the sum of the unconstrained optimal combination-weights may be far from being unity, as discussed in Sections 8.1.4.1 and 8.2.1. In such cases, one may expect the results of the unconstrained and the constrained MSE-OLCs to be significantly different.

These results illustrate that MSE-OLCs can dramatically improve the accuracy of a neural network based model. The computational effort associated with estimating the MSE-OLC combination-weights involves only simple matrix manipulations, as shown in Sections 3.2 and 3.3, which are modest if compared to requiring extra NN training effort to achieve a similar improvement in accuracy.

Figure 3.1 shows the approximation obtained using the unconstrained MSE-OLC against that obtained using NN4, the true best NN. The unconstrained MSE-OLC improves the quality of the fit, especially near the far ends (tails) of $r_1(X)$.

Figure 3.2 shows the approximations obtained using the simple averaging of the outputs of the six NNs, and from using NN3 (the true best NN to approximate the first- and second-order derivatives of $r_1(X)$ as discussed later in Section 5.2.1).

Figures 3.1 and 3.2 demonstrate that the unconstrained MSE-OLC yields a superior fit to $r_1(X)$ compared to using a single best NN or the simple averaging of the six NNs.

An interesting observation in this example is that the unconstrained MSE-OLC yields combination-weights that (almost) sum to unity. This is a common behavior in the cases where the component NNs are "well-trained[2]," as discussed later in Section 8.2.1. In this example, the root mean squared (RMS) error associated with the best NN is 0.011 and that associated with the worst NN is 0.030, which show that the component NNs are well-trained. This is also evident from Figures 3.1 and 3.2.

Another interesting observation is that the constant term is not statistically significant (not statistically different from zero). This has to do with the biasedness of

[1] "True" means computed relative to the *true* (known) response function.

[2] "Well-trained" means that the outputs of these networks are already very close to $r(\vec{X})$, which results in small approximation errors associated with the component NNs.

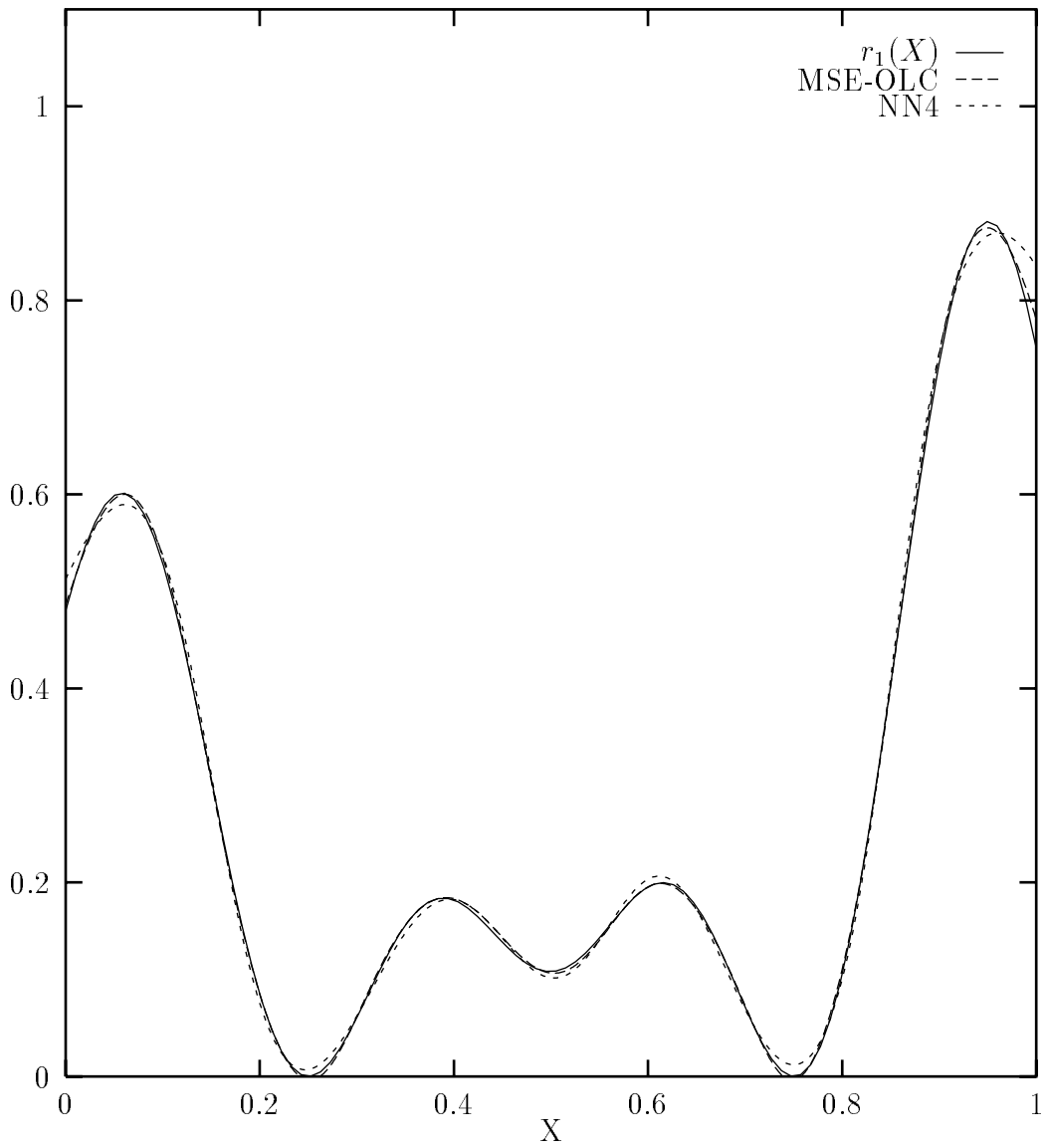

Figure 3.1 The function $r_1(X)$ and the approximations obtained using the unconstrained MSE-OLC and using NN4.

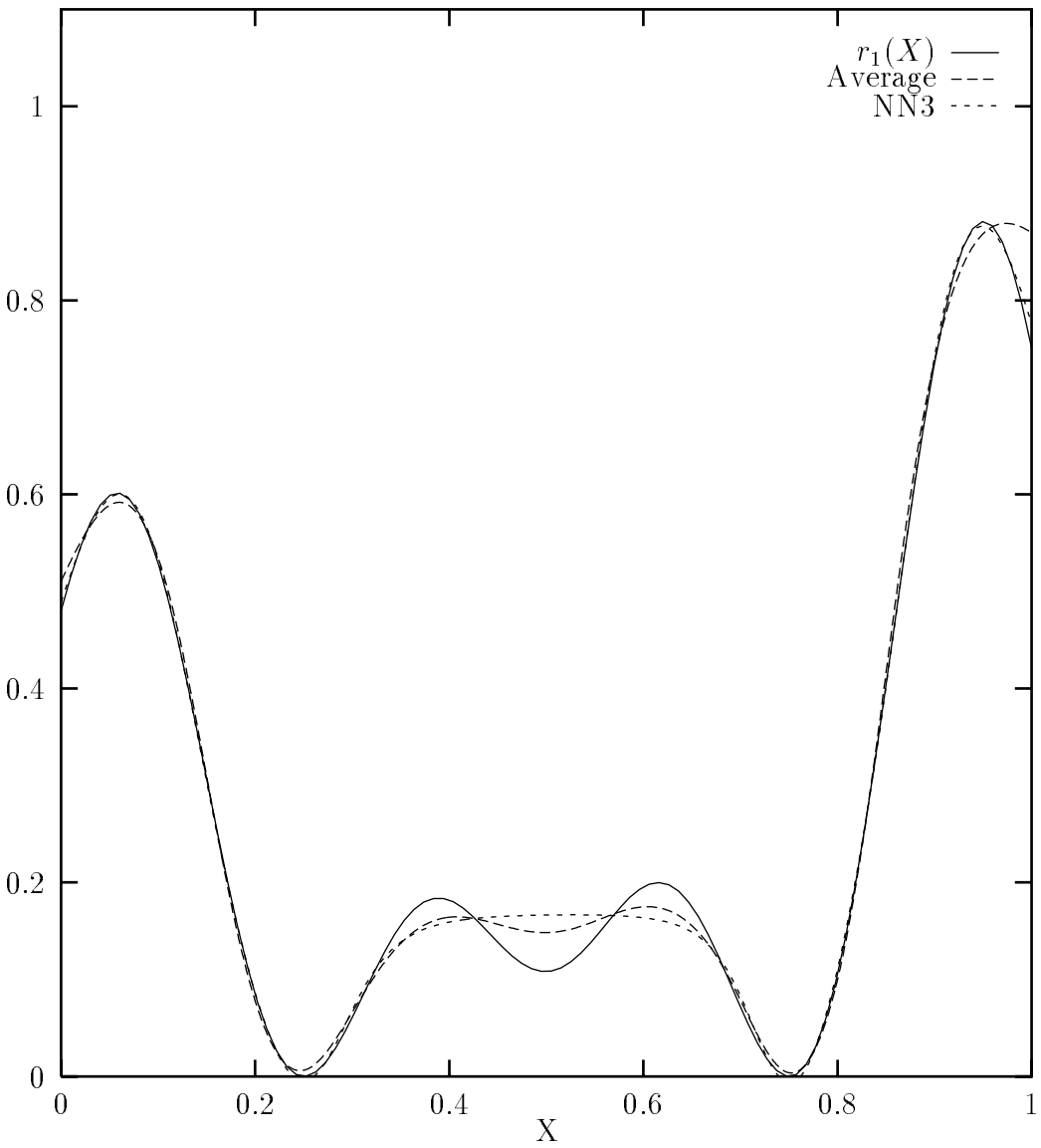

Figure 3.2 The function $r_1(X)$ and the approximations obtained using the simple averaging of the outputs of the six NNs and using NN3.

the component NNs. Well-trained NNs may have insignificant bias and as a result there may be no need for the constant term.

4. A PRODUCT ALLOCATION PROBLEM

In this chapter, unconstrained MSE-OLCs (U-OLCs) are used in constructing a NN-based model that aids in solving a product allocation problem.

4.1 Problem Description

We consider the allocation problem of a product made by an Indiana manufacturing company. Due to the workmanship requirements, the production capacity is limited. Meanwhile, the daily demand is typically two to three times larger than the available supply. The product is made and then distributed to 40 Customer Service Centers (CSCs) spread across the United States.

The problem is to allocate the available supply among the demanding CSCs in a manner that partially satisfies their demand, while following some guidelines and constraints. Among these guidelines and constraints are the capacity of the distribution trucks (98 units/truck), the priority of each CSC, the frequency and size of historical demands of each CSC, and the geographic location of each CSC. These constraints make the allocation problem fairly complex. Currently, the daily allocation schedule is planned manually and requires approximately four hours. The present human scheduler has been working for the company for a long time and her performance is deemed satisfactory by the company. In the last few years, the company tried unsuccessfully to use a variety of approaches to reduce the time and effort associated with the daily allocation schedule.

As an interim step, we investigate the use of a NN-based model, that is trained to mimic the human scheduler, to produce initial daily schedules that the human scheduler can improve on, or perhaps integrate with other approaches. Thus, we hope to reduce the time and effort associated with the daily allocation process.

4.2 Model Structure

The data for creating the NN model consist of the allocation schedules for 42 consecutive[1] working days. For each day, the demand by each CSC as well as the supply allocated (by the human scheduler) to that CSC are given. The total daily demand ranges between 1093 and 3093 units, and the available daily supply ranges

[1]According to the scheduler, given the daily demand, the time dependency among the individual daily schedules may be ignored.

between 637 and 1127 units. The data are split randomly into a training data set of 30 days and a testing data set of 12 days.

The number of available data patterns compared to the number of connection-weights in the NN is often a serious concern during training. If the latter number is larger than the former one, the NN may tend to fit to the data rather than to the underlying process, a problem usually referred to as overfitting.

Given the relatively small data set available for creating the model, the allocation problem is partitioned into a number of (similar) smaller problems, each requiring a relatively small NN. The 40 CSCs are partitioned into 4 groups, and once the daily supply to a given group is determined by a NN, it may be reallocated among the CSCs in that group by other NNs. This hierarchical approach requires NNs much smaller than the case of creating one NN to handle the 40 CSCs in one level. Partitioning the problem among a number of small NNs may also speed up the training and may result in a better model accuracy. The resultant allocation subproblems are similar; hence, we here focus on the first subproblem in the hierarchy, allocating the available supply among the 4 groups.

The individual demands from each group, the total demand, and the available supply are inputs to the model. The output of the model is the supply to be allocated to each group. Thus, the model has 6 inputs and 4 outputs. The total MSE is used as a measure of the accuracy of the NN model (as explained in Section 2.6).

4.3 Input-Output Representations

The following three different data representations are investigated:

1. All inputs and outputs are expressed in number of units.

2. The inputs are expressed in number of units, while the outputs (individual supplies) are expressed as percentages of the available supply.

3. The individual demands as well as the individual supplies are expressed as percentages of the total demand and available supply (respectively). The total demand and available supply are in number of units.

4.4 Neural Network Topologies

Since the NN topology may influence its approximation capability, three different network topologies are considered:

- 6–3–4 NN: network with one hidden layer that contains 3 hidden units;

- 6–4–4 NN: network with one hidden layer that contains 4 hidden units;

- 6–3–2–4 NN: network with two hidden layers that contain 3 and 2 hidden units (respectively).

The activation function for the hidden units as well as the output units is the logistic sigmoid function $g(s) = (1 + e^{-s})^{-1}$.

4.5 Neural Networks Training

The networks are initialized with independent random connection-weights generated uniformly from the interval [–0.3, 0.3]. Since the initial connection-weights may affect the convergence of the training as well as the accuracy of the resultant network, for every input-output representation, three networks (replications) of each of the three topologies — initialized with independent random connection-weights — are tried. The networks are trained using the Error Backpropagation algorithm with a learning rate of 0.01 for 1000 iterations with the connection-weights being updated after each training pattern. At the end of training, the NN that yields the best performance (among the nine NNs) in terms of the total MSE on the training data[2] is selected. The total MSEs on the training and testing data are shown in Table 4.1.

Table 4.1 Total MSEs of best trained NNs obtained after 1000 iterations

Input-output Representation	Training Data Total MSE	Testing Data Total MSE
A	6303	8228
B	0.0066	0.0093
C	0.0064	0.0086

4.6 Results of U-OLC of the Trained Networks

The merits of the U-OLC are investigated by measuring its impact on: 1) The number of required replications of NNs from each topology, 2) The accuracy of the resultant model, and 3) The required training time to achieve a given model accuracy. The following discussion covers these three factors.

[2]With "small" training budgets, there is little chance for overfitting to the training data. Actually, in this example the best NNs on the training data are also the best on the testing data.

1. The number of required replications of NNs from each topology:

   For each input-output representation, the U-OLC of the three NNs (one from each of the three topologies) is constructed. The training data are used in estimating the optimal combination-weights. The performance of the U-OLC is then evaluated on the training data as well as the testing data, and the results are summarized in Table 4.2. In all the three input-output representations, the U-OLC yields significant reductions in the total MSE over both the training and testing data compared to the best NN among the NNs included in the combination. These reductions are between 15 % and 16 % for the training data, and between 20 % and 22 % for the testing data. Since only the training data are used in estimating the optimal combination-weights for the U-OLC, the comparable accuracy improvements achieved by the combined model on the training and testing data sets suggest that the U-OLC generalizes well.

   Table 4.2 Total MSEs of U-OLC of three NNs trained for 1000 iterations

Input-output Representation	Training Data Total MSE	Testing Data Total MSE
A	5643	6869
B	0.0060	0.0079
C	0.0057	0.0072

   Moreover, comparing the results in Table 4.2 with the results shown in Table 4.1, the U-OLC of only three NNs — out of the nine available NNs — outperforms the best NN amongst the nine NNs in all the three input-output representations. The reductions in the total MSE are between 9 % and 11 % for the training data, and between 15 % and 17 % for the testing data. This result suggests that using U-OLC may eliminate the need for additional replications from the three NN topologies.

2. The accuracy of the resultant model:

   For each of the three input-output representations, the U-OLC of the available nine trained networks is constructed. The performances of the U-OLCs on both

the training and the testing data are summarized in Table 4.3. Compared to the performance of the single best network (Table 4.1), the U-OLCs of the nine trained networks (Table 4.3) reduce the total MSE by 27 % to 44 % on the training data, and by 29 % to 42 % on the testing data. Since only the training data are used in estimating the optimal combination-weights for the U-OLC, the comparable accuracy improvements achieved by the combined model on the training and testing data sets suggest that the U-OLC generalizes well.

Table 4.3 Total MSEs of U-OLC of nine NNs trained for 1000 iterations

Input-output Representation	Training Data Total MSE	Testing Data Total MSE
A	3501	4761
B	0.0043	0.0054
C	0.0047	0.0061

3. The required training time to achieve a given model accuracy:

   The training budget is extended to 100000 iterations, with the final outcome of training being the NN that yields the lowest MSE on the testing data within the allowed budget[3]. The performances of the best[4] NNs[5] obtained with the extended training budgets are evaluated and summarized in Table 4.4.

   Extending the training budget has resulted in better individual NNs, as evident from comparing the performances of the best NNs in Tables 4.1 and 4.4.

   Despite of the shorter training period, the U-OLCs of nine NNs trained for 1000 iterations outperform the best NNs obtained with a training budget of 100000

[3]With a "large" training budget, there is a valid concern of overfitting to the training data. Thus, the performance of the NNs over the testing data may be monitored to detect the occurrence of such overfitting. However, using such procedure gives the resultant NNs some (unfair) advantage over those trained with the 1000 iteration budget, not only because the formers are trained longer, but also since they have benefited from the testing data.

[4]One best NN is chosen from among the nine trained NNs in each case.

[5]The best NNs result from 13000 to 76000 iterations out of the allowed 100000 iterations.

Table 4.4 Total MSEs of best trained NNs obtained after 100000 iterations

Input-output Representation	Training Data Total MSE	Testing Data Total MSE
A	3782	5270
B	0.0038	0.0068
C	0.0048	0.0060

iterations on both the training and the testing data, in representations A and B. In representation C, the U-OLC of the nine networks — trained for 1000 iterations — performs as good as the best NN obtained with a training budget of 100000. These results suggest that using U-OLC may eliminate the need for excessive training to achieve a given model accuracy.

4.7 Discussion

In all the three input-output representations (A, B, C), the U-OLC of the nine trained networks yields better model accuracy than the corresponding best NNs (Tables 4.1 and 4.3). Hence, we focus on the analysis of the U-OLC models.

From Table 4.3, representation B yields lower total MSE compared to representation C on both the training data and the testing data. Moreover, converting the NN outputs in representations B to be in terms of number of units instead of percentages of the available supply, the total MSE on the training data is 3639, and on the testing data is 4455. Thus, representation B performs better than representation A in terms of the total MSE on the testing data. This makes representation B the best performer among the three representations, since the performance on the testing data is often used as a measure of out-of-sample performance (robustness) of the model.

Another performance measure, of special practical value, is the frequency of the approximation error, $\tilde{\delta}$, exceeding a certain tolerance level. Since the capacity of the distribution trucks is 98 units, this value is chosen to be the tolerance level. For a given output, the number of times that $\tilde{\delta}$ exceeds the above tolerance is divided by the total number of data points, yielding the desired error frequency. This performance

measure indicates the frequency that the NN schedule is different from the human schedule by one or more truck load, for a given group (out of the 4 groups of CSCs). For representation B, the error frequencies for the four outputs (groups) are between 3% and 17% on the training data; and between 0% and 25% on the testing data. These results indicate that the NN-based model produces allocation schedules close to those produced by the human scheduler.

4.8 Conclusions

Our pilot study shows that NNs may be successfully employed to capture allocation patterns and relationships that influence product allocation decisions.

U-OLC is a straightforward and effective method for integrating the knowledge acquired by a number of trained networks. Since multiple trained networks are often available as byproducts of the modeling process, the additional computational effort required to create an U-OLC is essentially that of estimating the optimal combination-weights, which is mainly a matrix inverse. The gain in accuracy, compared to using the best network, is substantial in our example. Moreover, the reduction in the training time required to achieve such accuracy is dramatic.

5. APPROXIMATING A FUNCTION AND ITS DERIVATIVES USING MSE-OLC OF NEURAL NETWORKS

In this chapter, we obtain approximations of the first- and second-order derivatives of a function from a NN trained to approximate the function values. Derivative information obtained from a trained NN may provide valuable insight about the underlying (data-generating) process. In Section 5.1, expressions for (systematically) computing the first- and second-order derivatives from a trained NN are derived. In Section 5.2, the use of MSE-OLC in improving the accuracy of the approximations of the derivatives of a function from a NN-based model is investigated.

5.1 Sensitivity Analysis for Neural Networks with Differentiable Activation Functions

As universal approximators, NNs offer a systematic approach for modeling industrial processes, especially processes which are otherwise hard to analyze in full detail. Many success stories are reported in the literature [11, 78, 3], yet there are some cases with fairly limited success [20, 97].

One of the key problems that affect the success of process modeling is the ability to extract information about the model structure and the relationships among its inputs and outputs from the trained NN [54, 97]. Such information is essential for model validation and for process optimization and control. Moreover, in some cases where the original process is not well understood [3, 78], this information can be employed as a basis for analyzing the process and for determining the most significant factors that affect it.

One approach to deal with this problem is proposed by Klimasauskas [60] who uses first-order sensitivity analysis, based on the perturbation of one input variable at a time and monitoring the outputs' variations.

We propose computing the function derivatives directly from the trained NN as a basis for the sensitivity analysis. In Section 5.1.1, expressions to compute the first- and second-order output derivatives (sensitivities) with respect to the NN inputs for a general multilayer feedforward NN with differentiable activation functions are derived. The derivatives are computed in a systematic manner, starting at the output layer and then moving towards the input layer, similar to the backward propagation of errors in the Error Backpropagation learning. Expressions for higher-order derivatives may be derived in the same manner. This approach is more efficient for computing the

output derivatives than using perturbation, as it yields closed-form expressions that can be systematically computed.

The first- and second-order derivatives obtained from a NN that is trained on the process response, may be used as approximations of the gradient vector and the Hessian matrix of the process response. Thus, the output derivatives can be bases for inference about input-output relationships. Moreover, a NN-based process model can easily support efficient implementations of Newton and Quasi-Newton methods for process optimization [35, pages 105–107].

5.1.1 Expressions for neural networks output derivatives

In this section, we calculate the output derivatives with respect to the inputs for a trained NN. The derivatives are computed one layer at a time, in a systematic manner, starting from the output layer and proceeding backwards toward the input layer. The expressions of the first- and second-order derivatives are given in Sections 5.1.1.1 and 5.1.1.2, respectively. This method may be easily extended to obtain higher-order derivatives, if desired.

The activation functions are assumed to be differentiable. For computational convenience, all the activation functions are assumed to be of the same form, the logistic sigmoid function $g(s) = (1 + e^{-s})^{-1}$. We also assume that there are no bypassing connections in the network (i.e., direct interconnections can only exist from one layer to the succeeding one). However, while the first assumption about the differentiability of the activation function is fundamental, the latter assumptions are not, and the method discussed here can easily be modified to deal with such variations.

The capability of multilayer feedforward neural networks to approximate an unknown mapping $f : \mathcal{R}^n \rightarrow \mathcal{R}$ arbitrarily well has been investigated by Cybenko [22], Funahashi [31], Hecht-Nielson [51], and Hornik et al. [56]. Moreover, Hornik et al. [57] show that multilayer feedforward networks with as few as a single hidden layer and an appropriately smooth hidden layer activation function are capable of arbitrarily accurate approximation to an arbitrary function and its derivatives. This fundamental result provides the necessary theoretical foundation for the output sensitivity analysis discussed here.

For a feedforward NN consisting of an input layer, N hidden layers and an output layer, the input vector is introduced to the input layer (layer 0), which transfers it to the first hidden layer (layer 1). A weighted sum of the NN inputs is then computed at each processing element PE (neuron) in the first hidden layer, based on the connection-weights between the input layer and the first hidden layer. This sum is used to compute the output of every PE (the "PE value") by applying the activation function $g(s)$. Then, the PEs in the first hidden layer pass their values to the subsequent layers and finally to the output layer (layer N+1). The values of the PEs in the hidden layers and the output layer are calculated in the same manner as in the first hidden layer using the connection-weights and $g(s)$.

5.1.1.1 First-order output derivatives

First-order output derivatives are computed by applying a simple backward chaining partial differentiation rule. First, the output derivatives with respect to the values of the PEs of layer N are calculated. Backward chaining is then employed to calculate the output derivatives with respect to the network input variables. This is done as follows:

- For the PEs in layer N:

$$\frac{\partial O_k}{\partial h_i^N} = \frac{\partial O_k}{\partial net_k^{N+1}} \cdot \frac{\partial net_k^{N+1}}{\partial h_i^N} = O_k\,(1 - O_k) \cdot w_{ik}^N, \qquad \forall\, i,k.$$

- For the PEs in the remaining hidden layers (layer j, $j = N-1, \ldots, 1$):

$$\frac{\partial O_k}{\partial h_i^j} = \sum_l \frac{\partial O_k}{\partial h_l^{j+1}} \cdot \frac{\partial h_l^{j+1}}{\partial net_l^{j+1}} \cdot \frac{\partial net_l^{j+1}}{\partial h_i^j} = \sum_l \frac{\partial O_k}{\partial h_l^{j+1}} \cdot h_l^{j+1}\,(1 - h_l^{j+1}) \cdot w_{il}^j, \qquad \forall\, i,k.$$

- For the input layer (layer 0):

$$\frac{\partial O_k}{\partial I_i} = \sum_l \frac{\partial O_k}{\partial h_l^1} \cdot \frac{\partial h_l^1}{\partial net_l^1} \cdot \frac{\partial net_l^1}{\partial I_i} = \sum_l \frac{\partial O_k}{\partial h_l^1} \cdot h_l^1\,(1 - h_l^1) \cdot w_{il}^0, \qquad \forall\, i,k.$$

where O_k : output of the kth PE in the output layer (layer N+1),
h_i^j : output of the ith PE in layer j, $j = 1, \ldots, N$,
I_i : ith input to the network,
net_i^j : weighted sum of the inputs to the ith PE in the jth layer, $j = 1, \ldots, N+1$,
w_{ik}^j : connection-weight between the ith PE in layer j and the kth PE in layer $j+1$, $j = 0, \ldots, N$.

5.1.1.2 Second-order output derivatives

The second-order partial derivatives of the network outputs, with respect to the inputs, are calculated using a backward chaining rule similar to that used for the first-order derivatives in Section 5.1.1.1.

- For the PEs in layer N:

$$\begin{aligned}
\frac{\partial^2 O_k}{\partial h_i^N \partial h_j^N} &= \frac{\partial}{\partial h_i^N}\left(O_k\,(1 - O_k) \cdot w_{jk}^N\right) \\
&= \frac{\partial O_k}{\partial h_i^N} \cdot (1 - 2O_k) \cdot w_{jk}^N \\
&= O_k\,(1 - O_k)\,(1 - 2O_k) \cdot w_{jk}^N \cdot w_{ik}^N \qquad \cdots\cdots\cdots\cdots\cdots \forall\, i,j,k.
\end{aligned}$$

- For the PEs in the remaining hidden layers (layer τ, $\tau = N-1, \ldots, 1$):

$$\frac{\partial^2 O_k}{\partial h_i^\tau \partial h_j^\tau} = \frac{\partial}{\partial h_i^\tau}\left(\sum_l \frac{\partial O_k}{\partial h_l^{\tau+1}} h_l^{\tau+1}\left(1-h_l^{\tau+1}\right) w_{jl}^\tau\right) =$$

$$\sum_l w_{jl}^\tau \left[\left(1-2h_l^{\tau+1}\right) h_l^{\tau+1}\left(1-h_l^{\tau+1}\right) w_{il}^\tau \frac{\partial O_k}{\partial h_l^{\tau+1}} + h_l^{\tau+1}\left(1-h_l^{\tau+1}\right) \frac{\partial}{\partial h_i^\tau}\left(\frac{\partial O_k}{\partial h_l^{\tau+1}}\right)\right]$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \forall\, i,j,k.$$

$$\text{where} \quad \frac{\partial}{\partial h_i^\tau}\left(\frac{\partial O_k}{\partial h_l^{\tau+1}}\right) = \sum_\beta \left(\frac{\partial^2 O_k}{\partial h_\beta^{\tau+1} \partial h_l^{\tau+1}}\right)\left(\frac{\partial h_\beta^{\tau+1}}{\partial h_i^\tau}\right)$$

$$= \sum_\beta \left(\frac{\partial^2 O_k}{\partial h_\beta^{\tau+1} \partial h_l^{\tau+1}}\right)\left(h_\beta^{\tau+1}\left(1-h_\beta^{\tau+1}\right) w_{i\beta}^\tau\right)$$

- For the input layer (layer 0):

$$\frac{\partial^2 O_k}{\partial I_i \partial I_j} = \sum_l w_{jl}^0 \left[(1-2h_l^1) h_l^1 (1-h_l^1) w_{il}^0 \left(\frac{\partial O_k}{\partial h_l^1}\right) + h_l^1 (1-h_l^1)\left(\frac{\partial^2 O_k}{\partial I_i \partial h_l^1}\right)\right]$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \forall\, i,j,k.$$

$$\text{where} \quad \frac{\partial^2 O_k}{\partial I_i \partial h_l^1} = \sum_\beta \left(\frac{\partial^2 O_k}{\partial h_\beta^1 \partial h_l^1}\right)\left(h_\beta^1 (1-h_\beta^1) w_{i\beta}^0\right)$$

Higher-order derivatives may be obtained in a similar fashion. However, for many practical applications, the first- and second-order derivatives are the most commonly used.

5.1.2 Example 2

A NN with a single hidden layer that has three PEs is used in approximating $r_2(X) = \sin(4X)$, $X \in [-1, 1]$. The Error Backpropagation algorithm is used for training the network with a training data set of 100 uniformly distributed points. Another data set of 50 uniformly distributed points is used for testing the NN.

For the trained NN, the resultant root mean squared (RMS) errors on the training and the testing data sets are 0.08 and 0.10 respectively. Figures 5.1, 5.2, and 5.3 show the function, $r_2(X)$, and its first- and second-order derivatives and the approximations obtained from the trained NN (respectively). Figures 5.1–5.3 show that the approximations obtained from the trained NN are close to the actual values of the given function and its derivatives.

The most accurate approximation is obtained for the function values, while the approximations of the first- and second-order derivatives of the function are of increasingly (as a result of the the magnification of the approximation errors) lower accuracy . The Error Backpropagation algorithm minimizes an error function defined on the difference between the NN's output and the desired response, $r_2(X)$,

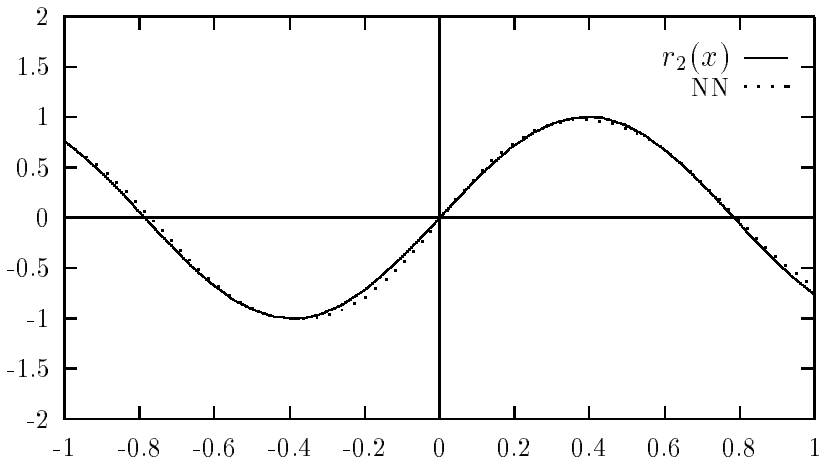

Figure 5.1 The function $r_2(x)$ and the approximation obtained using the NN.

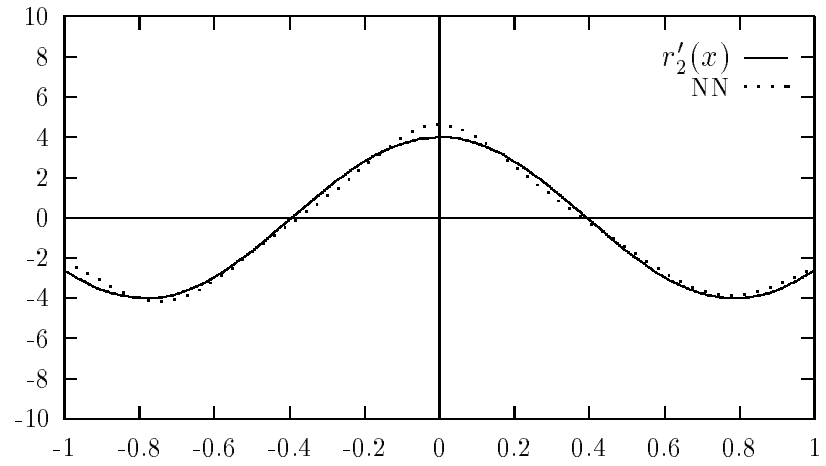

Figure 5.2 The first-order derivative $r_2'(x)$ and the approximation obtained using the NN.

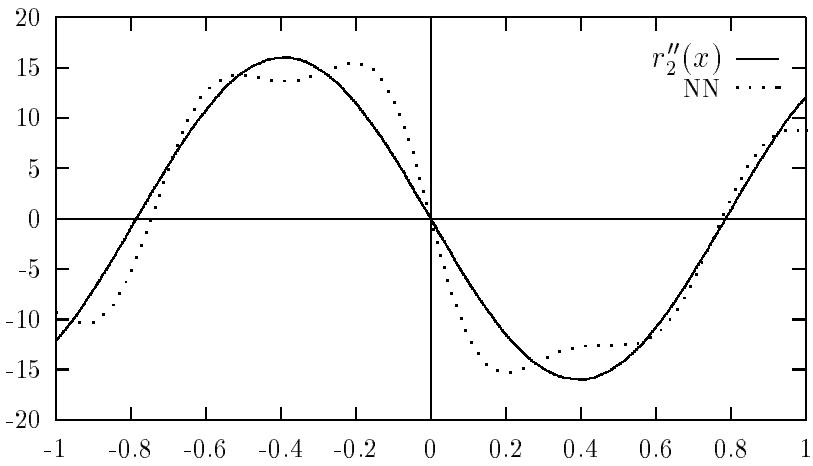

Figure 5.3 The second-order derivative $r_2''(x)$ and the approximation obtained using the NN.

which does not necessarily yield the best approximating set of connection-weights for approximating the derivatives of $r_2(X)$. However, it seems that a trained NN that adequately approximates a given function will approximate the derivatives of that function well.

5.2 Improving the Accuracy of Approximating a Function and Its Derivatives by Using MSE-OLC of Neural Networks

To examine the accuracy improvement as a result of using MSE-OLCs of NNs to approximate a function and its derivatives, an illustrative example is presented and discussed in this section.

As mentioned in Section 5.1.2, the approximations of the function derivatives are often of (increasingly) lower accuracies compared to the approximation of the function values. This is intuitive because learning algorithms often focus on minimizing the error in approximating the function, which need not minimize the errors in approximating the derivatives. Moreover, if a number of NNs are trained to approximate a given function, the best NN to approximate the function, need not be the best NN to approximate the derivatives of that function, as demonstrated by the following example.

5.2.1 Example 1 (Continued)

Consider the problem of approximating the function

$$r_1(X) = 0.02(12 + 3X - 3.5X^2 + 7.2X^3)(1 + \cos 4\pi X)(1 + 0.8 \sin 3\pi X)$$

over the interval $[0, 1]$, with the trained NNs from Section 3.4. The approximations of the first- and the second-order derivatives of $r_1(X)$ are computed using the expressions derived in Section 5.1.1.

The unconstrained MSE-OLC, obtained in Section 3.4, yields *true* MSEs of 0.10 and 133.3 in approximating $r_1'(X)$ and $r_1''(X)$, the first- and second-order derivatives of $r_1(X)$ respectively. These values are 70% and 65% less than the corresponding MSEs produced by NN3, the true best NN to approximate the first- and second-order derivatives, respectively. Moreover, the MSEs produced by using the unconstrained MSE-OLC are 86% and 74% less than the MSEs produced by using the simple averaging of the outputs of the NNs, respectively.

An interesting outcome is that NN4, which is the true best NN to approximate $r_1(X)$, is not the true best NN to approximate $r_1'(X)$ or $r_1''(X)$. This suggests that there may be an added value for combining the NNs — rather than keeping only a single best NN — that extends beyond improving the approximation accuracy of the function to the approximation accuracy of its derivatives.

Figures 5.4–5.7 show the approximations obtained from NN3, NN4, the simple averaging of the NNs, and the unconstrained MSE-OLC plotted against the first- and second-order derivatives of $r_1(X)$, respectively. The approximation of the function value obtained by NN3 is shown in Figure 3.2.

Figures 5.4 and 5.6 show that the approximations of $r_1'(X)$ and $r_1''(X)$ (respectively) obtained using NN3 are fairly inaccurate in the interval $(0.4, 0.6)$. Averaging the outputs of the six NNs (simple averaging) helps improve accuracy, as shown in Figures 5.5 and 5.7. On the other hand, NN4 yields much better approximation accuracies in that region, while performing poorly near the tail ends of $r_1'(X)$ and $r_1''(X)$. The unconstrained MSE-OLC yields the best performance almost everywhere.

The dramatic reduction in the MSEs in approximating $r_1(X), r_1'(X)$, and $r_1''(X)$ as a result of using the MSE-OLC, as well as the visual comparison of the resultant approximations in Figures 5.4–5.7, strongly indicate that using MSE-OLC of the trained NNs can significantly improve the the accuracy of NN-based models, compared to using the single best NN, or using the simple averaging of the outputs of the NNs.

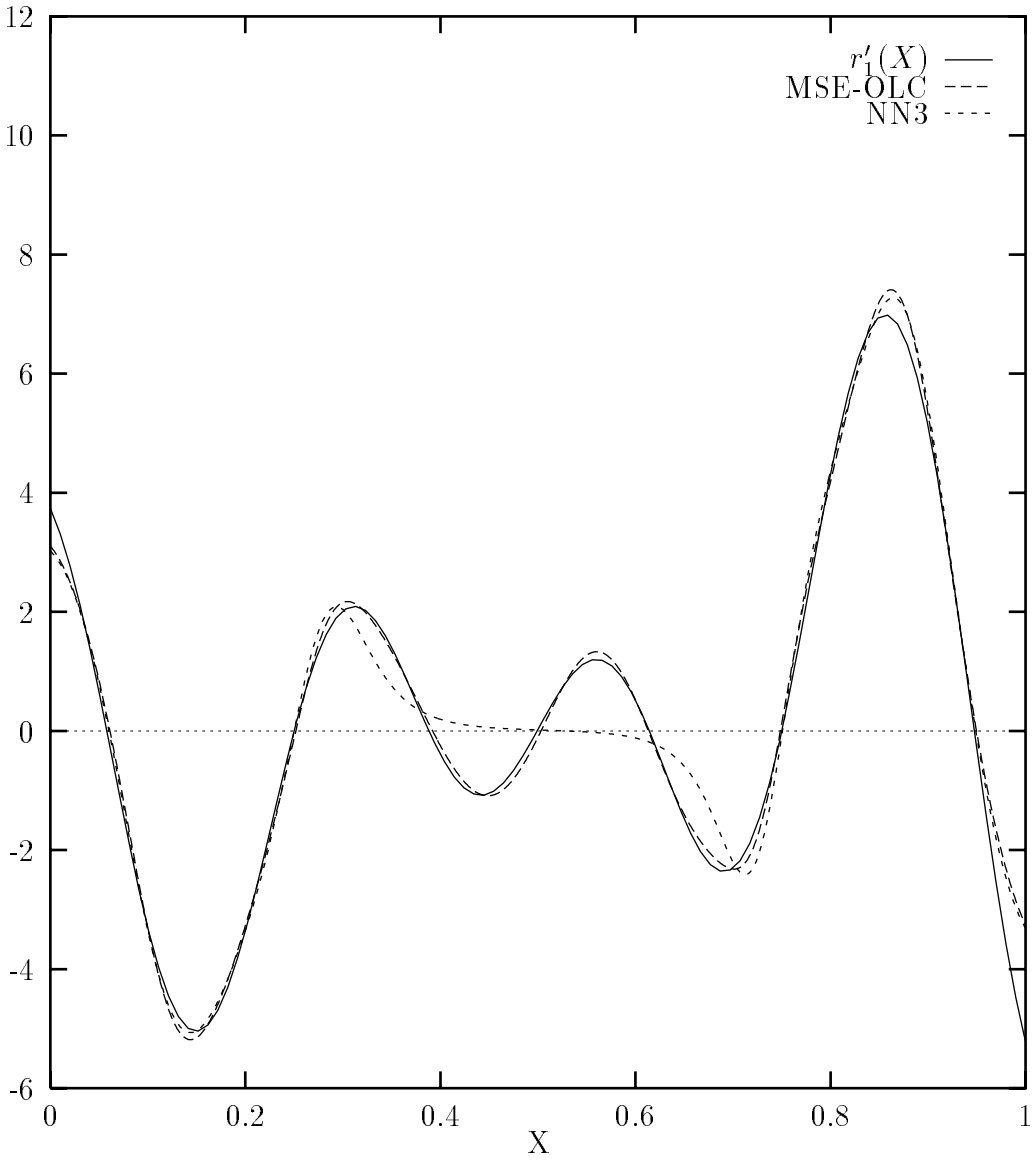

Figure 5.4 The first-order derivative $r_1'(X)$ and the approximations obtained using the unconstrained MSE-OLC and using NN3.

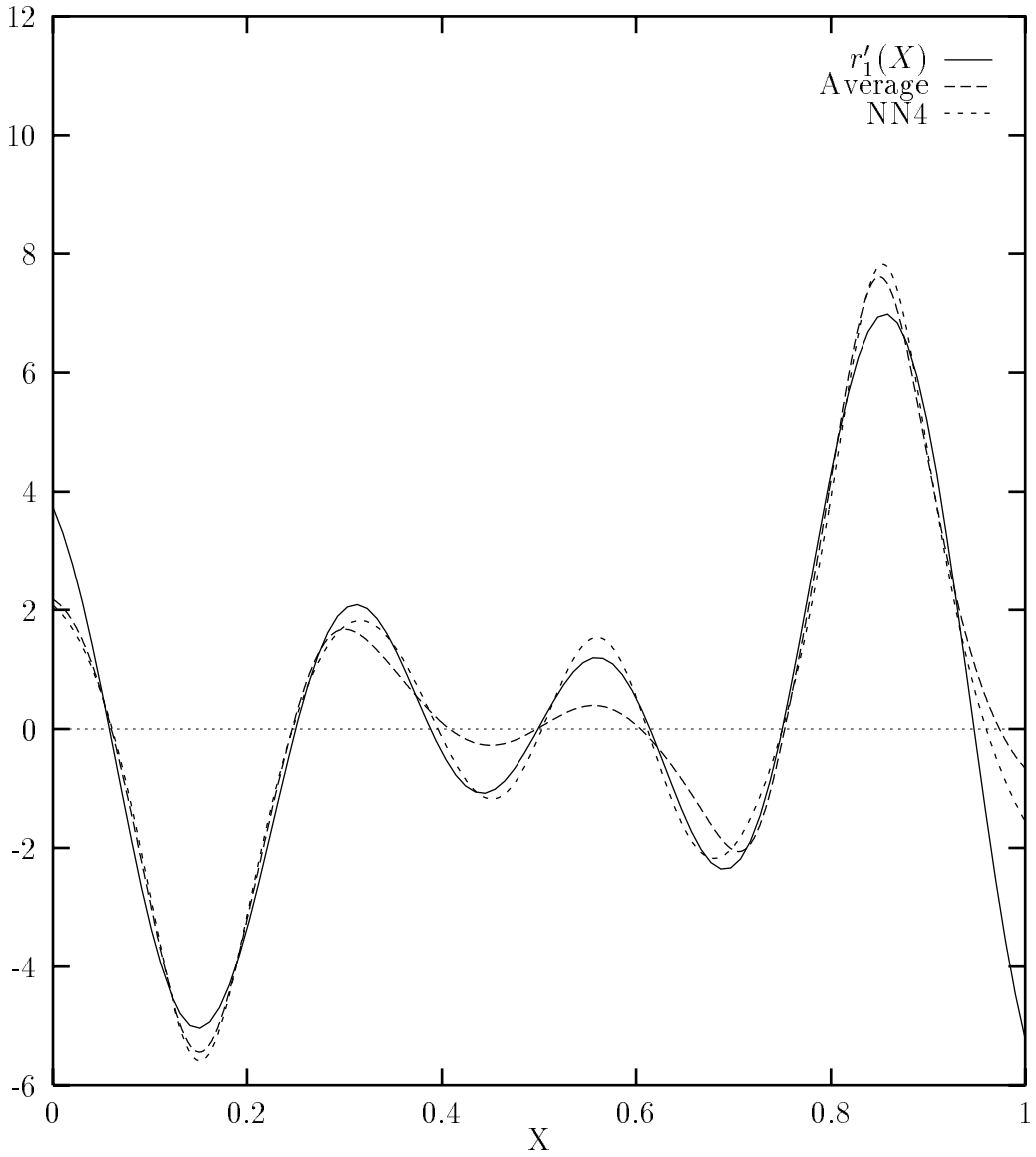

Figure 5.5 The first-order derivative $r_1'(X)$ and the approximations obtained using the simple averaging of the outputs of the six NNs and using NN4.

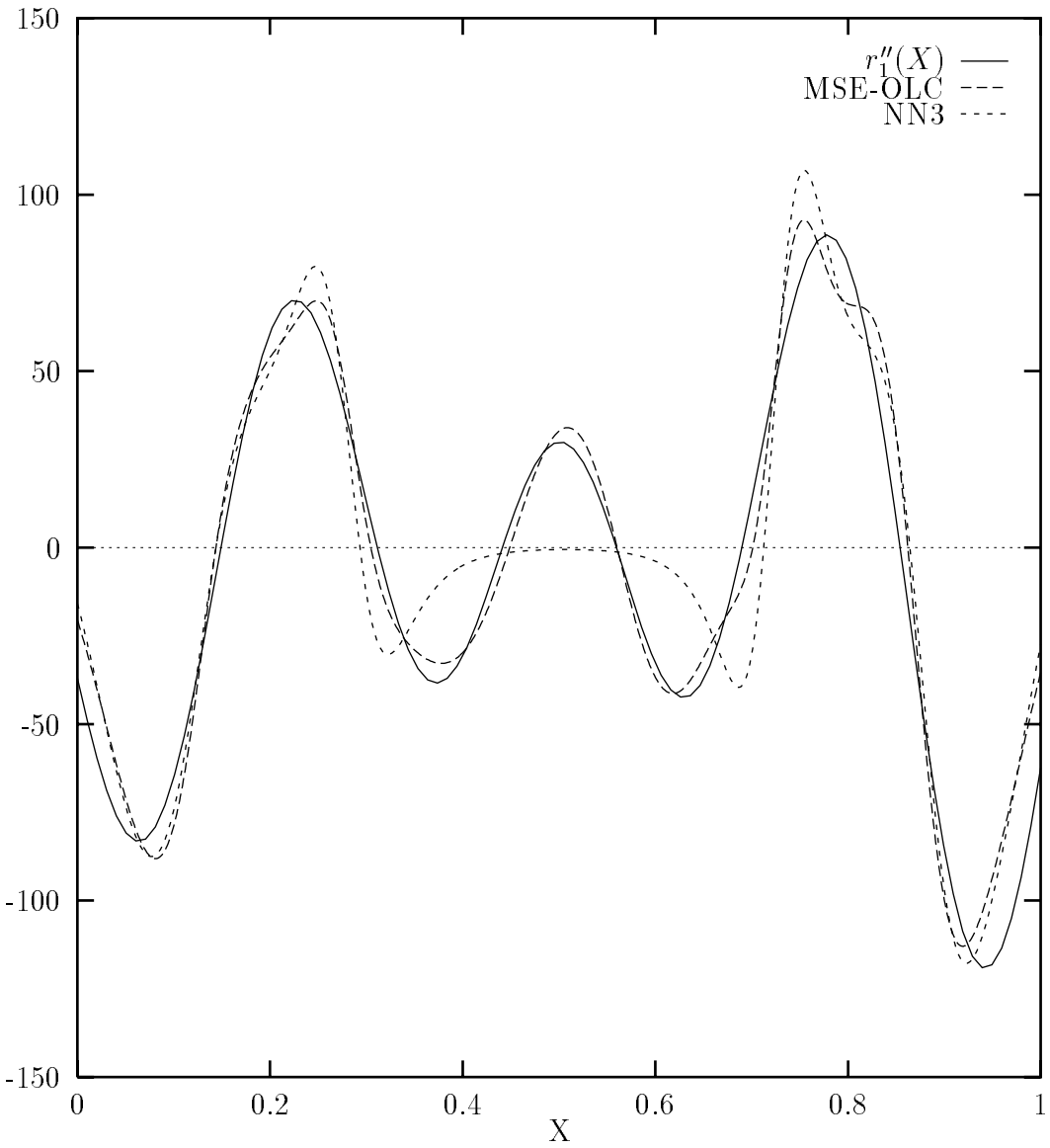

Figure 5.6 The second-order derivative $r_1''(X)$ and the approximations obtained using the unconstrained MSE-OLC and using NN3.

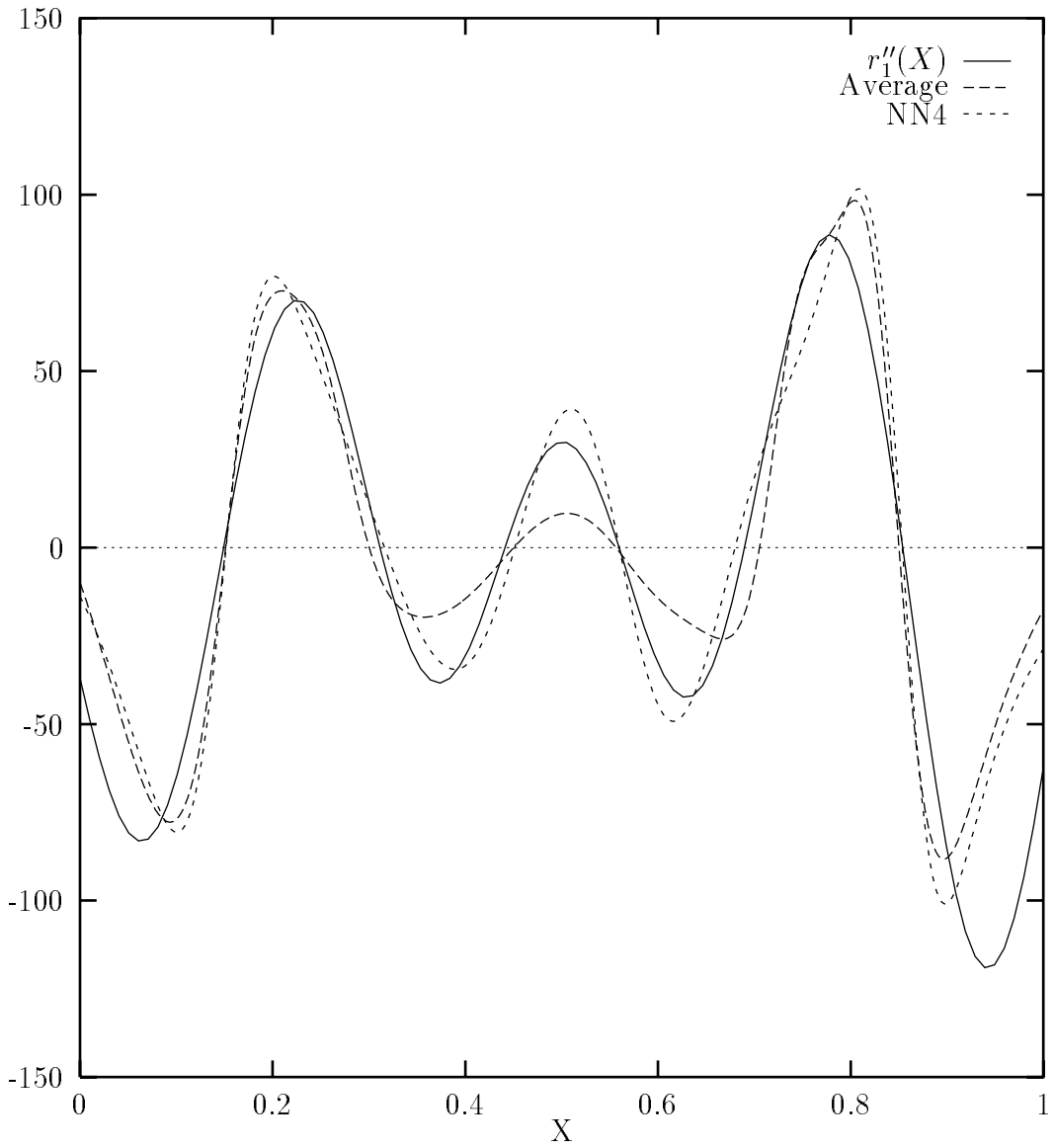

Figure 5.7 The second-order derivative $r_1''(X)$ and the approximations obtained using the simple averaging of the outputs of the six NNs and using NN4.

6. COLLINEARITY AND THE ROBUSTNESS OF THE MSE-OLC

In the forecasting literature, the computational and statistical ill-effects of collinearity are blamed for undermining the robustness (generalization ability) of OLCs [14, 19, 40, 74, 95, 111]. Likewise, in the literature on combining neural networks, Perrone and Cooper [89] point to the potential problems of ill-conditioned correlation matrices as a result of collinearity.

By construction, the unconstrained MSE-OLC with a constant term (theoretically) yields the minimal MSE among the four MSE-OLCs discussed in Chapter 2. In practice, the MSE-OLC combination-weights need to be estimated using (real-world) data. Hence, there is no guarantee that the estimated unconstrained MSE-OLC with a constant term will still be superior to other MSE-OLCs which impose restrictions on the combination-weights and/or exclude the constant term [14, 17, 55, 102].

The robustness of the MSE-OLC is investigated in Section 6.1. A discussion of collinearity and its ill effects is presented in Section 6.2. The impact of collinearity on estimating the optimal combination-weights is studied in Section 6.3. Methods for collinearity detection are discussed in Section 6.4. A discussion on harmful collinearity and two illustrative examples are included in Section 6.5. In Section 6.6, some methods for detecting the harmful effects of existing collinearity are briefly discussed. Then a method based on cross-validation is introduced.

6.1 The Robustness of the MSE-OLC

6.1.1 Definition

The *robustness* of an MSE-OLC stands for the invariance of the performance of the estimated MSE-OLC across observations sampled from the same distribution from which the data, used in estimating the combination-weights, are sampled. For the four MSE-OLCs defined in Chapter 2, this distribution is $F_{\vec{X}}$. Robustness is often referred to by the phrases "generalization ability" or "out-of-sample performance." In this dissertation, these three phrases are used interchangeably.

By construction, the estimated MSE-OLC minimizes the MSE over the data set $\mathcal{K}$ used in estimating the optimal combination-weights in Chapter 3. The concern is whether (or not) the estimated MSE-OLC will perform on a *new* (out-of-sample) data set as well as it performs on $\mathcal{K}$.

6.1.2 Testing the robustness of the MSE-OLC

A simple test for the robustness of the estimated MSE-OLC is to compute the resultant MSE over a testing data set, disjoint from $\mathcal{K}$, but sampled from the same distribution, $F_{\vec{X}}$. This MSE may be used as a measure of the performance of the estimated MSE-OLC on future observations, and indicates whether or not the estimated MSE-OLC is robust. This testing strategy is straightforward and is similar to the strategy often adopted in testing the generalization ability of a (single) NN [27, 66, 76, 106]. Likewise, in the literature on combining forecasts, many advocate the use of out-of-sample testing of the combination [24, 25, 32, 42, 55]. This test may be integrated in a more comprehensive framework, that attempts to correct for *harmful*[1] collinearity problems arising from the data, in order to construct more robust MSE-OLCs.

Collinearity and its impact on OLS estimation are discussed in Section 6.3. Methods to improve the robustness of the MSE-OLC based on collinearity diagnosis are introduced in Chapter 7.

6.2 Collinearity

6.2.1 Definition

Belsley [8, page 19] gives a precise definition for collinearity:

> Literally, two variates are collinear if the data vectors representing them lie on the same line, that is, in a subspace of dimension 1. More generally, k variates are collinear, or linearly dependent, if one of the vectors that represents them is in an exact linear combination of the others, that is, if the k vectors lie in a subspace of dimension less than k.

An alternate (quantitative) definition for collinearity is given in Section 6.4.2.

6.2.2 Collinearity and correlation

The correlation between two variables (variates) is defined by the expected value of the normalized product of the variables centered around their corresponding means (i.e. mean-centered), where *normalized* stands for normalization with respect to the standard deviations of the two variables, respectively.

Belsley [8, pages 26–27] indicates that while a high correlation coefficient between two explanatory (regressor) variables can indeed point to possible collinearity problems, the absence of high correlations cannot be viewed as evidence of the absence of collinearity problems, and that a high correlation implies collinearity, but the converse is not true. Belsley [8, pages 370–372] shows that k variables can be perfectly collinear and still have no absolute pairwise correlation between any two of them that exceeds $1/(k-1)$.

[1] "Harmful" collinearity refers to collinearity that undermines the robustness of the combination.

At this point in the discussion, we'd like to draw attention to two fundamental issues:

1. Collinearity and correlation are not the same thing [8, pages 20,26]. Hence, special diagnosis need to be applied to detect the presence of collinearity, (possibly) in addition to estimating the pairwise correlations among the variables being studied.

2. Not all collinearities harm the estimates of the combination-weights [8, pages 72–74], nor the robustness of the MSE-OLC[2]. Thus, beside looking for a diagnostic tool to detect the presence of collinearity, one also needs to look for an appropriate measure of the harmfulness of such collinearity.

6.2.3 Ill effects of collinearity

The ill effects of collinearity may be classified as computational and statistical ill-effects.

6.2.3.1 Computational ill-effects

The problem of solving a linear system of equations, and likewise matrix inversion, may be highly affected by the existence of collinearity among the system variables. Exact collinearity (linear dependency) results in a singular matrix. In less extreme cases, collinearity may cause the system to be near singular, and thus the solution may be highly sensitive to changes in the elements of the system.

For a linear system $Az = c$, the condition number of the matrix A provides a measure of the potential sensitivity of a solution. The condition number gives a magnification factor by which imprecision in the data can be blown up to produce even greater imprecision in the solution [8, page 71], see also [100]. A linear system is well-conditioned or ill-conditioned according to whether the condition number of its matrix is small or large [16, page 48].

For the OLS formulation of the MSE-OLC problems discussed in Section 3.2, the round-off errors in inverting the matrices which contain the y_j's (the outputs of the component NNs) may be particularly large due to the presence of collinearity [81, pages 377–378]. Likewise, there may be large round-off errors in inverting the matrices which contain the δ_j's (the approximation errors of the component NNs) in Section 3.3. Subsequently, such round-off errors are magnified when calculating the estimates of the optimal combination-weights or making any subsequent calculation.

[2]Notice that the definition of *harmful* in Section 6.1.2, is different from Belsley's definition in [8, pages 72–74]. Unless stated otherwise, the former definition will be followed in all subsequent discussions.

6.2.3.2 Statistical ill-effects

Statistically, the presence of collinearity among the regressor variables results in high correlations between the regression coefficients [81, page 275]. Thus, the common interpretation of regression coefficients as measuring the expected sensitivity of the response variable to varying the corresponding regressor variable is not fully applicable when collinearity exists [81, page 385].

Collinearity can cause the estimated variance of the OLS estimates to be high [53, pages 521–522]. Hence, the estimated regression coefficients may tend to vary widely from one sample to the next [81, pages 384–385], which may undermine the robustness of the MSE-OLC. Moreover, the inflated variances are quite harmful to the use of regression as a base for hypothesis testing and estimation [8, pages 71–72]. For instance, the estimated regression coefficients individually may not be statistically significant despite the existence of a definite (physical) relation between the associated regressor variables and the response variable [81, pages 382–383].

6.2.3.3 Additional remarks

1. Collinearity may also occur between the constant term and one (or more) of the regressor variables [8, page 22–24].

2. The presence of collinearity does not necessarily mean than the resultant model is not a good fit [81, page 384]. That is to say, not all collinearity need to be harmful, as long as a regression algorithm is used that does not blow up in the presence of highly collinear data [8, page 73].

3. In less extreme cases, the collinearity problem may be mitigated by low noise in the generation of the response variable [8, page 73]. Since the variance of the OLS coefficients is a function of σ^2 (defined in Section 3.2), a sufficiently small σ^2 may offset the effect of collinearity in inflating the estimated variances of the OLS coefficients.

6.3 Collinearity and Estimating the MSE-OLC Combination-Weights

In Section 3.2, the four MSE-OLC problems presented in Section 2.3 are shown to be equivalent to OLS regression, where the regressor variables are $y_j(\vec{X})$'s, the outputs of the p trained NNs, or a function of these outputs. Since the NNs are individually trained to approximate the same response, $r(\vec{X})$, one can expect the correlation between the $y_j(\vec{X})$'s to be fairly (positively) high. The inherent high (positive) correlations between the y_j's[3] make estimating the MSE-OLC prone to collinearity problems (Section 6.2.2). As a result, the robustness of the MSE-OLC may be affected [14, 40].

[3]We write y_j instead of $y_j(\vec{X})$ for simplicity. The same applied to δ_j in subsequent discussions.

The alternate expressions presented in Section 3.3 may (appear to) be less vulnerable to the ill effects of collinearity than those in Section 3.2, since the former expressions rely on the approximation errors of the component NNs, δ_j's, instead of the y_j's [14]. However, the correlations between the δ_j's may also be high (and positive). In combining forecasts, Bunn [14] argues that the errors from each model are surprises common to all, much of which we must accept as being unpredictable a priori, thereby giving a largely positive element to their correlation. Unlike forecasting models, which are usually constructed based on a priori understanding of the underlying process beside fitting to the data, NN models learn about the underlying process directly (solely) from the training data. Nevertheless, Bunn's argument may still apply. Bunn [13, 14] also argues that if there is large positive correlation between the forecast errors, no great gains for combining can be expected. Furthermore, if an unstable optimizing approach is used in this case, the results could be much worse than that from a simple policy of equal combination-weights (the simple averaging) or even that of selecting the apparently best model.

In Example 1, the pairwise correlations between the outputs of the six trained NNs range from 0.997 to above 0.999, and thus are fairly high. In addition, the pairwise correlations between the δ_j's of the six trained NNs range from -0.021 to 0.986. However, the unconstrained MSE-OLC reduces the *true*[4] MSE in the function value by 87% and 95% compared to the best NN and the simple averaging, respectively (Section 3.4). Moreover, the unconstrained MSE-OLC also reduces the *true* MSEs in the first- and second-order derivatives of the function by 70% and 65% (respectively) compared to the best NN to approximate the derivatives, and by 86% and 74% (respectively) compared to the simple averaging (Section 5.2.1). Thus, extremely high positive correlations between the y_j's, which are associated with positive (except for one pairwise correlation $= -0.021$) correlations between the δ_j's do not necessarily eliminate the benefit of combining nor result in harmful collinearity.

From the above discussion, it is evident that for the successful employment of the MSE-OLC we need to answer the following three questions:

1. How to detect the presence of collinearity and identify the (regressor) variables associated with it?

2. How to determine that an existing collinearity is harmful?

3. How to deal with harmful collinearities in order to improve the robustness of the MSE-OLC?

The first question is addressed in Section 6.4. Some methods for detecting the presence of collinearity are highlighted, with an emphasis on a collinearity diagnosis developed by Belsley et al. [9] and revised in [8].

Testing the robustness of the MSE-OLC in Section 6.1.2 offers a good practical answer for the second question. Other measures for the harmfulness of an existing collinearity are discussed in Section 6.6. The third question is addressed in Chapter 7.

[4] "True" refers to the MSE with respect to the *true* function, since it is known in this example.

6.4 Collinearity Detection

In the regression literature, several procedures have been developed for collinearity detection. Belsley [8, pages 26–37] discusses the main classes of these procedures and points to their strengths and weaknesses. In Section 6.4.1, some of the widely used procedures are examined. In Section 6.4.2, the collinearity diagnostic developed by Belsley et al. [9] and revised in [8] is discussed and its main aspects are briefly analyzed.

6.4.1 Common methods for collinearity detection

- Examine the correlation matrix of the regressor variables: This may be the most popular method for detecting collinearity [13, 14, 40, 89, 95, 111]. While high pairwise correlations can point out to possible collinearity problems, as discussed in Section 6.2.2, the absence of high correlations does not necessarily imply the absence of collinearity (see also [53, page 523]). Moreover, in the presence of several co-existing collinearities, pairwise correlations can not point to the variables involved in each one of these collinearities [8, page 27].

  Although examining the correlation matrix can detect possible collinearity problems, it can neither be used as a conclusive evidence for the absence of collinearity, nor can it identify the variables involved in the individual collinearities in case of several co-existing collinearities.

- Examine the variance inflation factors (VIFs) [81, pages 391–393] and [53, pages 521–523]. The VIFs are the diagonal elements of the inverse of the correlation matrix of the regressor variables. The VIFs measure how much the variances of the estimated regression coefficients are inflated as compared to when the regressor variables are not collinear. A VIF, VIF_k, is equal to 1 when the kth regressor variable is linearly independent of the remaining regressor variables. When the kth regressor variable has perfect linear association with other regressor variables, VIF_k becomes unbounded (infinitely large). The largest VIF among the regressor variables is often used as an indicator of the severity of collinearity. Furthermore, Stewart [100] shows that the largest VIF bounds the condition number of the matrix of the regressor variables from below.

  However, like pairwise correlations, VIFs is as sufficient but not necessary condition to collinearity. Also VIFs are not able to diagnose a number of co-existing collinearities [8, pages 27–30].

- Examine the determinant of the correlation matrix of the regressor variables [53, page 523]. When the regressor variables are linearly independent the determinant of the correlation matrix equal to one. A perfect collinear relation among the regressor variables makes the determinant of the correlation matrix equal to zero.

However, like pairwise correlations or VIFs, this method cannot determine the number of co-existing collinearities [8, page 30], nor can it identify the variables involved in each collinearity.

- Do all-subsets regression on the regressor variables. This is a straightforward method aimed at discovering all possible linear dependencies among the regressor variables by brute-force examination. Beside being computationally expensive, the presence of co-existing collinearities is capable of causing this procedure to misfire diagnostically [8, pages 30–31].

- Examine the eigenvalues and eigenvectors (principal components) of the correlation matrix of the regression variables [53, page 523] and [8, pages 35–37]. The presence of "small" eigenvalues indicates the existence of collinearity. The ratio between the largest and the smallest eigenvalues can also be used as a measure for collinearity.

  However, this method cannot be used to identify the variables involved in an existing collinearity. Belsley [8, pages 36–37] provides an example that shows that for an eigenvector corresponding to a small eigenvalue, relatively large eigenvector elements may indicate the involvement of a variable in a collinear relation, but relatively small eigenvector elements cannot be relied upon to show the absence of involvement.

6.4.2 BKW's collinearity diagnostics

In Section 6.4.1, several methods for detecting collinearity are discussed. These methods, although possessing some potential benefits, are shown to be inadequate for the collinearity diagnosis task defined in Question 1 in Section 6.3.

Belsley, Kuth, and Welsch [9] and Belsley [6, 7, 8] have developed diagnostics for explicit measurement of the severity of collinearity. These diagnostics are capable of determining the existence of multiple collinearities and identifying the variables involved in each collinearity as well. We refer to these diagnostics as the BKW diagnostics.

Belsley [8] gives illuminating discussions, which include valuable geometric and analytic considerations, to support the BKW collinearity diagnostics and to illustrate how and why it delivers what it promises: detecting the presence of collinearities and identifying the variables involved in each individual collinearity. The BKW collinearity diagnostics employ "condition indexes" [8, pages 55–56] to detect the existence of collinearities and to determine their number and strength. Then, the variables involved in each collinearity are identified by using the "variance-decomposition proportions." For completeness, condition indexes and variance-decomposition proportions are defined below.

Condition indexes: The condition indexes, η_k, of an $a \times b$ matrix Q are defined by

$$\eta_k \stackrel{\text{def}}{=} \frac{\mu_{\max}}{\mu_k}, \quad k = 1, \ldots, b,$$

where $\mu_k, k = 1, \ldots, b$, are the singular values [12, page 81] of Q. In theory, there will be exactly as many zero singular values as the number of exact linear dependencies among the columns of Q [8, pages 45–46]. In practice, the presence of a strong linear dependency results in a small singular value and, consequently, a large associated condition index.

Belsley [8, pages 40–56] discusses the use of the condition indexes of the "scaled" matrix of regressor variables, which may include a column of ones (before scaling) corresponding to the constant term in the regression, to detect the presence of collinear relations among the regressor variables, and to determine the number of such collinear relations. "Scaling" or "column scaling" of the matrix here means to scale the columns to have unit length — in the Euclidean sense (see [8, pages 65–67, 171–175] for a justification of column scaling). The largest condition index that is associated with the smallest singular value defines the scaled condition number of the matrix [8, pages 52–54]. This scaled condition number provides a measure of the potential sensitivity of the solution of a linear system of exact equations to changes in the data as discussed in Section 6.2.3.1. A similar result is true for a solution of an inexact system of equations, such as the regression equations [8, pages 54–55]. Experimental results [8, pages 79–127] show that weak collinearities are associated with condition indexes around 5–10, whereas moderate to strong collinearities are associated with condition indexes of 30–100.

Variance-decomposition proportions: As stated in Section 3.1, the covariance matrix of the OLS estimators of the optimal combination-weights, in the unconstrained MSE-OLC with a constant term, is $\sigma^2 \Psi^{-1}$. For the other three MSE-OLC problems, $\acute{\Psi}$ is used instead of Ψ.

The singular value decomposition (SVD) of the associated matrix of regressor variables, Q say, may be defined by $Q = LDV^t$, where $L^tL = V^tV = I_b$ and D is a $b \times b$ diagonal matrix with the diagonal elements equal to the singular values of Q. L is an $a \times b$ column orthogonal matrix, and V is a $b \times b$ column and row orthogonal matrix [8, pages 42–43].

Using this SVD of Q, the covariance matrix of the OLS estimators of the combination-weights may be written as

$$\mathrm{Cov}(\widehat{\vec{\alpha}}) = \sigma^2 V D^{-2} V^t. \qquad [6.1]$$

Thus, the variance of the kth regression coefficient, $\widehat{\alpha_k}$ is

$$\mathrm{var}(\widehat{\alpha_k}) = \sigma^2 \sum_j \frac{v_{kj}^2}{\mu_j^2}, \qquad [6.2]$$

where $V = [v_{ij}]$, and μ_j's are the singular values of Q.

In Equation 6.2, $\mathrm{var}(\widehat{\alpha_k})$ is expressed as a sum of terms (components), each of which is associated with only one of the singular values of Q. The singular

values appear in the denominators of the terms, and so a relatively small singular value results in a relatively large term. Belsley [8, page 58] defines the (k, j)th variance-decomposition proportion as the proportion of the variance of the kth regression coefficient associated with the jth component of its decomposition in Equation 6.2. Thus,

$$\phi_{kj} \stackrel{\text{def}}{=} \frac{v_{kj}^2}{\mu_j^2} \quad \text{and} \quad \phi_k \stackrel{\text{def}}{=} \sum_{j=1}^{b} \phi_{kj}, \qquad k = 1, \ldots, b, \tag{6.3}$$

and the variance-decomposition proportions are

$$\pi_{jk} \stackrel{\text{def}}{=} \frac{\phi_{kj}}{\phi_k}, \quad k, j = 1, \ldots, b. \tag{6.4}$$

Hence, for every singular value, μ_j, there is a corresponding condition index, η_j, and variance-decomposition proportions $\pi_{jk}, k = 1, \ldots, b$.

Belsley [8, page 59–70] discusses some of the properties of the condition indexes and the variance-decomposition proportions. As mentioned earlier, associated with each linear dependency (collinearity) is one small singular value that results in an associated large condition index. By definition, the sum of the variance-decomposition proportions associated with the variance of each regression coefficient, $\text{var}(\widehat{\alpha_k})$, is one. A matrix Q with mutually orthogonal columns results in one and only one non-zero variance-decomposition proportion associated with each $\text{var}(\widehat{\alpha_k})$ or with each condition index. On the other hand, if (only) two columns of Q, l and e say, are (strongly) linearly dependent, there will be one large condition index, μ_f say, with π_{fl} and π_{fe} being near unity, while the remaining variance-decomposition proportions associated with μ_f being near zero. The existence of the near linear-dependency between the two columns, indicated by a large condition index, results in a relatively large contribution to the variance of the regression coefficients associated with these columns, as reflected by the associated variance-decomposition proportions.

Based on the condition indexes and the variance-decomposition proportions, Belsley [8, page 67] suggests the following double conditions for diagnosing the presence of degrading collinearity:

1. A scaled condition index judged to be high.

2. High scaled variance-decomposition proportions for two or more estimated regression coefficients variances.

These two conditions provide an alternative definition to collinearity (beside the definition given in Section 6.2.1). In other words, collinearity exists when the above two conditions are met.

The number of scaled condition indexes deemed to be large (say greater than 30) indicates the number of co-existing linear dependencies, and the magnitude of these large scaled condition indexes provides a measure of their relative "tightness."

Furthermore, for a large condition index, the associated large variance-decomposition proportions (say greater than 0.5) identifies the variables involved in that near linear dependency, and the magnitude of these large proportions provides a measure of the degree to which the corresponding regression estimates have been degraded by the presence of that near linear dependency.

For detailed discussion and analysis of the BKW collinearity diagnostics, the reader may refer to [8]. Belsley [8, pages 128–163] provides an excellent summary that includes some experimental results as well as valuable guidelines to handle co-existing and simultaneous near linear dependencies.

6.5 Harmful Collinearity

Example 1, discussed in Sections 3.4 and 5.2.1, demonstrates the benefit of using MSE-OLC in significantly reducing the MSE in approximating the function as well as its first- and second-order derivatives.

However, as mentioned in Section 6.3, the pairwise correlation among the y_j's in Example 1 are fairly high, ranging from 0.997 to above 0.999. Moreover, the correlations among the δ_j's are mostly positive, with most of them (10 out of 15) being more than 0.5. These high (positive) correlations raise valid concerns about the computational ill-effects of collinearity [89], as well as the statistical ill-effects [14, 19, 40, 74, 95, 111].

However, high positive correlations, in themselves, cannot be taken as a conclusive evidence of the harmfulness of existing collinearity, but merely as a "warning" that collinearity exists and may harm the robustness of the MSE-OLC. Before discussing methods for detecting the harmful effects of existing collinearity, two examples are presented to demonstrate that such harmful effects really exist, and that collinearity can severely undermine the robustness of the MSE-OLC.

6.5.1 Example 3

Consider approximating the function $r_3(X) = \sin[2\,\pi\,(1-X)^2]$, where $X \in [0,1]$.The range of $r_3(X)$ is $[-1,1]$. Two 1–3–1 NNs (NN1 and NN2), two 1–2–2–1 NNs (NN3 and NN4), and two 1–4–1 NNs (NN5 and NN6) are initialized with independent random connection-weights uniformly distributed in [–0.3, 0.3]. The activation function for the hidden units as well as the output units is the logistic sigmoid function $g(s) = (1+e^{-s})^{-1}$. The NNs are trained using the Error Backpropagation algorithm with a learning rate of 0.25 for 5000 iterations. The training data set consists of 10 uniformly distributed independent points.

NN3, the *true*[5] best NN, yields an MSE of 0.09 on the training data. The *true* MSE corresponding to NN3 is 0.46. The simple averaging of the outputs of the six NNs yields an MSE of 0.10 on the training data and a true MSE of 0.68.

[5] "True" means relative to the true (known) response function.

Using the training data to estimate the optimal combination-weights, the unconstrained MSE-OLC with a constant term reduces the MSE on the training data set to almost zero (up to six decimal places). However, it yields a true MSE of 91, that is about 19457% larger than the true MSE produced by NN3, and about 13330% larger than the true MSE produced by the simple averaging, clearly indicating that the MSE-OLC can cause a disaster if applied "blindly," that is without proper assessment of its robustness. The MSE on the training data is listed only for completeness, since the true measure of performance and robustness is true MSE obtained relative to the true (known) function, $r_3(X)$.

An interesting observation in the above MSE-OLC is that the two-sided P-values of all the regression coefficients (including the constant term) are less than 0.035. In fact, the two-sided P-values of six out of the seven regression coefficients are less than 0.001. Thus, all the individual regression coefficients are statistically significant at a level of significance of 0.05. Hence, the statistical significance of the optimal combination-weights may not be an adequate measure of the robustness of the MSE-OLC. This conclusion comes in agreement with Belsley's [6] argument on using the usual t-statistic, $t = b_k/s_{b_k}$, where b_k is the OLS estimator and s_{b_k} is the estimator of the standard deviation of b_k, for testing the statistical significance of b_k. Belsley argues that: "while low t's may indicate (data) weaknesses, high t's need not indicate their absence."

Another important observation is that the scaled condition number of the matrix $\widehat{\Psi}$ (defined in Section 3.2.1) equals 813000, which is astronomical according to Section 6.4.2. Moreover, the scaled condition number of the matrix $\widehat{\Omega''}$ (defined in Section 3.3.2) is 96972, which is also astronomical. These high values indicate the presence of very strong collinearity among the outputs of the NNs as well as among their approximation errors.

A possible reason for the lack of robustness of the MSE-OLC, in this example, is the small number of data points used in combining, or in other words the small number of degrees of freedom[6] in the regression model. Indeed, increasing the number of data used points for combining the six NNs by 5 points (uniformly distributed and independent), results in an MSE-OLC that yields a true MSE of 0.68 (down from 91), which is about 45% larger than the true MSE produced by NN3, but is equal to the true MSE produced by the simple averaging.

Thus, with the five extra points, a dramatic improvement in the robustness of the MSE-OLC is achieved. The new optimal combination-weights are all statistically significant with associated two-sided P-values almost equal to zero. Moreover, the optimal combination-weights have adequate signal-to-noise[7] according to Belsley's [6] test for harmful collinearity and other forms of weak data. Thus, such tests are not sufficient to conclude that an existing collinearity is harmless, according to the

[6]Degrees of freedom = number of data points – number of parameters in the regression model.

[7]With a test size of 0.05 at an adequacy level of 0.999999, and a test size of 0.01 at an adequacy level of 0.999. Extrapolations from the tables given in [6] are used since the smallest number of degrees of freedom in these tables is 10, while in this example there are only 8 degrees of freedom.

definition of *harmful* in Section 6.2.2. The scaled condition number of $\hat{\Psi}$ has dropped by a factor of five to 171619, but is still considered very high. Also, the scaled condition number of $\widehat{\Omega''}$ has dropped by a factor of five to 18006, and is also still considered very high. These reductions in the scaled condition numbers confirm that the collinearity has been reduced by introducing new observations, which explains the improvement in the robustness of the MSE-OLC.

Acquiring more data, whenever possible, is one of the most effective means for "breaking up" the collinearity in the data (e.g. [8, page 297] and [40]), as discussed in Section 7.1. In Section 7.3, some algorithms for improving the robustness of the MSE-OLC by reducing collinearity, for a given (fixed) set of data, are introduced.

One may suspect that if the number of degrees of freedom in the regression model is large (in the original MSE-OLC), then the problem of robustness ceases to exist. Having more data aids in obtaining a better approximation to the true function both in terms of the individual NNs and in terms of the OLC. Moreover, including more data helps breakup existing collinearity. However, there is no guarantee that having more degrees of freedom in the original MSE-OLC would necessarily result in a robust MSE-OLC. In this example, even with the extra five points, the performance of the MSE-OLC is worse than that of the best NN. Example 4 further illustrates this fundamental issue.

6.5.2 Example 4

Consider approximating $r_3(X)$ in Example 3, given a training data set of 20 uniformly distributed independent points. The true response is corrupted with 10% Gaussian noise, that is an $N[0., (0.2)^2]$ is added to the true response $r_3(X)$ in the training data.

Six NNs, of the same topologies as in Example 3, are initialized with independent random connection-weights uniformly distributed in [–0.3, 0.3]. Then the networks are trained using the Error Backpropagation algorithm with a learning rate of 0.25 for 5000 iterations.

NN6, the true best NN, yields a true MSE of 0.033, and the simple averaging yields a true MSE of 0.078. Using the training data to estimate the optimal combination-weights, the unconstrained MSE-OLC with a constant term yields a true MSE of 0.938, which is 2772% more than that produced by NN6, and 1107% more than that produced by the simple averaging. Hence, the robustness of MSE-OLC is deemed unacceptable.

Although the number of degrees of freedom in the regression model used in estimating the optimal combination-weights is 13, which is more than four times those in the original MSE-OLC in Example 3, the consequencies of blindly applying the MSE-OLC, without proper assessment of its robustness, are the same.

The scaled condition number of the matrix $\hat{\Psi}$ is 31924, and that of the matrix $\widehat{\Omega''}$ is 1105. Both scaled condition numbers are high, indicating the presence of strong collinearity.

However, unlike in Example 3, the two-sided P-values of the regression coefficients are more than 0.26 suggesting that all the regression coefficients are not statistically significant at a level of significance of 0.05. Although such individual testing of the significance of the regression coefficients is of limited value in presence of strong collinearity as explained in Section 6.2.3.2, yet it may be used as evidence for the existence of severe collinearity ([81, pages 278–282] and [53, page 523]).

Increasing the number of data points used in combining the six NNs by an extra 10 uniformly distributed independent points, corrupted with the same level of noise as the original 20 points, results in an MSE-OLC that yields a true MSE of 0.100 (down from 0.938), which is much better than that produced by the original MSE-OLC. However, the resultant MSE is still 206% larger than that of NN6, and 29% larger than that of the simple averaging. Hence, even when increasing the number of degrees of freedom in the regression model to 23, the MSE-OLC is still suffering from the ill effects of collinearity. Indeed, the scaled condition number of $\widehat{\Psi}$ is 20263 and that of $\widehat{\Omega''}$ is 805. Both scaled condition numbers are significantly smaller than those of the original MSE-OLC, yet are still considered high.

6.5.3 Conclusions

Examples 3 and 4 illustrate some of the important aspects of the relation between collinearity and the robustness of the MSE-OLC. Among these aspects are

1. The presence of collinearity may severely undermine the robustness of the MSE-OLC.

2. Including extra data points in the construction of the MSE-OLC can help improve the robustness of the MSE-OLC by breaking up the collinearity among the regressor variables. However, having an "adequate" number of degrees of freedom in the regression model associated with the MSE-OLC, by itself, does not guarantee the robustness of the MSE-OLC.

3. Collinearity can affect the power of the conventional statistical tests for significance of the optimal combination-weights. However, even if all the optimal combination-weights are statistically significant or, for that matter, possess adequate signal-to-noise ratio, this does not necessarily imply that collinearity is "harmless" nor does it necessarily imply that the resultant MSE-OLC is robust.

6.6 How to Determine that an Existing Collinearity is Harmful?

First, some of the common approaches for determining harmful collinearity are discussed. Then, a cross-validation approach that attempts to measure the direct impact of collinearity on the robustness of the MSE-OLC is presented and discussed.

6.6.1 Some common approaches

There are many controversial approaches to determine whether (or not) an existing collinearity is harmful. Among these approaches are

- In the statistics literature, many consider the existence of collinearity, by itself, harmful. Hines and Montgomery [53, pages 522–523] report that some authors consider VIFs that exceed 10 as indication of problems due to collinearity, while other authors consider VIFs that exceed 4 or 5 as fairly high. Neter et al. [81, pages 391–393] state that VIFs that exceed 10 may often be taken as an indication that collinearity may be unduly influencing the OLS estimates.

  In the literature on combining forecasts, Guerard and Clemen [40] consider correlations above 0.8, scaled condition numbers around (or exceeding) 30, and/or VIFs around (or exceeding) 5, as indicators of possible collinearity problems.

  However, as Belsley [8, pages 206–207] explains, the ill effects of collinearity may be counteracted by a sufficiently small error variance σ (defined in Section 3.2.1), so that not all collinearity need to be harmful. Smith and Campbell [98] state that

  > The essential problem with VIF and similar measures is that they ignore the parameters while trying to assess the information given by the data. Clearly, an evaluation of the strength of the data depends on the scale and nature of the parameters. One cannot label a variance or a confidence interval (or, even worse, a part of the variance) as large or small without knowing what the parameter is and how much precision is required in the estimate of the parameter. In particular, seemingly large variance may be quite satisfactory if the parameter is very large, if one has strong a priori information about the parameter, or if the parameter is uninteresting (perhaps because the associated variable will be constant during the forecast period). A meaningful assessment will require a well-defined loss function that must necessarily depend on the particular problem being examined.

- Belsley [6] and [8, pages 205–244] develops and discusses a test to assess the presence of harmful[8] collinearity and other forms of weak data, based on a Signal-to-Noise parameter associated with the OLS estimators. For an OLS estimator b_k, the signal-to-noise (s/n) parameter is $\tau \stackrel{\text{def}}{=} \beta_k/\sigma_{b_k}$, where β_k is a regression parameter, and σ_{b_k} is the standard deviation of b_k. Belsley [6] shows that this test for adequate s/n is more useful than the conventional tests of hypothesis, since the latter may be less accurate (unpowerful) in the presence of collinearity.

[8] According to Belsley [6], inadequate s/n together with collinearity defines harmful collinearity.

However, as discussed in Example 3, the presence of harmful[9] collinearity can go undetected by either types of tests, although such collinearity may have significantly undermined the robustness of the MSE-OLC.

6.6.2 A cross-validation approach for detecting harmful collinearity

As defined in Section 6.1.2, harmful collinearity results in the lack of robustness of the MSE-OLC. A straightforward approach to determine the harmfulness of an existing collinearity is by testing the robustness of the MSE-OLC. The simple test for the robustness of the MSE-OLC presented in Section 6.1.2 is fit for this task.

By construction, the MSE-OLC results in the smallest MSE on the combination data set, $\mathcal{K}$, compared to the best NN among the component NNs, and to the simple averaging of the corresponding outputs of the NNs in the combination. The robustness of the resultant MSE-OLC may be tested by comparing its performance to that of the best NN and the simple averaging on a different data set, referred to as the cross-validation data set. If the MSE-OLC is still the best performer on the cross-validation test, then one may conclude that it is robust. Otherwise, one may look for corrective measures to improve the robustness of the MSE-OLC. Asymptotically, as the size of the cross-validation set increases, this test measures the *true* robustness of the MSE-OLC.

Some important issues concerning the application of this cross-validation approach are

- According to the definition of *robustness* in Section 6.1.1, the data in the cross-validation set need to be sampled from $F_{\vec{X}}$.

- In practice, $F_{\vec{X}}$ is often unknown, and only a set of observed data, $\mathcal{K}$, is available for constructing the MSE-OLC. In such cases, assuming that the data are independent and equally likely, the observed data set may be split into an estimation data set, $\mathcal{K}_1$, and a cross-validation data set, $\mathcal{K}_2$. Such splitting is at the expense of reducing the data used in estimating the optimal combination-weights. Meanwhile, $\mathcal{K}_2$ needs to be sufficiently large in order to accurately test the robustness of the MSE-OLC.

- The notion of "best" NN needs a more precise definition. In practice, one does not know which of the trained NNs is the *true* best. There is no reason to believe that the best NN on the training data set will be the true best NN. In fact, a NN that overfits the most to the training data will have the lowest MSE among a number of trained NNs. A consistent estimator of the true best is the best performer among the trained NNs on the cross-validation set, $\mathcal{K}_2$. Asymptotically, as the number of data points in $\mathcal{K}_2$ increases, this estimator yields the true best NN.

[9] According to the definition given in Section 6.1.2.

7. METHODS FOR IMPROVING THE ROBUSTNESS OF MSE-OLC

The discussions in Chapter 6 highlight the role of harmful collinearity in undermining the robustness of MSE-OLCs. In this chapter, methods for improving the robustness of the MSE-OLC, by treating harmful collinearity, are investigated. In Section 7.1, some of the common approaches for treating harmful collinearity are discussed. A method for improving the robustness of MSE-OLC by restricting the combination-weights is investigated in Section 7.2. In Section 7.3, six algorithms for improving the robustness of MSE-OLC by the proper selection of the NNs in the combination are introduced.

7.1 Common Methods for Treating Harmful Collinearity

There is no consensus among statisticians on how to treat harmful collinearity, very much like the controversy on how to determine the presence of harmful collinearity in the first place. When collinearity harms the estimation of the regression coefficients or undermines the robustness of the regression model, then the method used for treating such collinearity needs to take into consideration the context of the particular problem being examined.

Several remedial measures for collinearity have been developed and investigated. These methods include:

- Introducing new data in order to breakup the collinearity pattern [81, page 394], [8, page 297], [40], and [53, page 523]. Examples 3 and 4, in Section 6.5, illustrate that the robustness of the MSE-OLC may be significantly improved by introducing new data.

  Unfortunately, this method may be limited by the ability or the cost of acquiring extra data in practice [8, page 297]. According to the MSE-OLC problem statement in Section 3.1, an underlying assumption is that the data set $\mathcal{K}$ is the only available source for combination data. Thus, the option of acquiring more data, as a remedial measure for harmful collinearity, is no longer valid.

- Restrict the use of the fitted regression model to inferences for values of the regressor variables which follow the same pattern of collinearity [81, pages 393–394].

- Express the regressor variables in the form of deviations from the mean [77, page 554], which is known as "mean-centering." While mean-centering can

sometimes help in reducing collinearity among the first-, second-, and higher-order terms of a given regressor variable [81, page 394], Belsley [8, pages 175–191] illustrates that mean-centering is ineffective in removing ill conditioning from a given basic data set.

- Using biased estimation techniques to improve the efficiency of estimating the regression coefficients (the optimal combination-weights). In the presence of collinearity, using a biased estimation procedure, such as ridge regression [81, pages 394–400] or latent root regression [105], one is essentially trading the introduction of small bias in the estimates with the reduction of their variance [103, pages 452–460]. For a critique on using biased regression methods in practice, refer to [98].

  In the literature on combining forecasts, Guerard and Clemen [40] indicate that, while the use of latent root regression produces more efficient estimates of the combination-weights compared to the OLS estimates, their out-of-sample forecasting performances are comparable.

- Using robust estimation [43] of the combination-weights, such as minimizing the absolute deviations. Hallman and Kamstra [42] argue that, while the OLS estimators are asymptotically the best linear unbiased estimators, robust estimation techniques may be more appropriate than OLS estimation for small samples.

- In the literature on combining forecasts, Bunn [13] suggests that assuming independence among the errors of the combined forecasts, that is assuming zero off-diagonal elements in the covariance matrix of the errors of the forecasts, may result in a robust combination over small samples. That is, to disregard the estimates of the correlations among the forecast errors for small samples.

  Bunn's suggestion comes in agreement with a finding of a large empirical study conducted by Newbold and Granger [82], in which combining methods that ignore correlation are more successful than those methods that attempt to take account of correlation. Clemen and Winkler [19] also demonstrate that, without sufficient data to estimate the variances precisely, imposing the independence assumption can indeed improve the performance of the combined model.

  However, this method is ad hoc in nature, and in practice there may be no "clear-cut method" for determining whether or not a given data set is "sufficiently" large.

- The introduction of prior information using Bayesian analysis. Belsley [8, pages 297–301] advocates the use of Bayes-like procedures for introducing the experimenter's subjective prior information as a treatment for collinearity problems. Palm and Zellner [85] also advocate the use of Bayesian approaches to combining forecasts even if little information is available.

However, since NNs are usually employed as universal approximators that acquire their knowledge (solely) from the data, Bayesian approaches may be of limited use.

- One or several regressor variables may be dropped from the model in order to lessen the collinearity [81, page 394] and [77, page 554], especially when some of the regressor variables contribute redundant information [94, page 466–467].

  This method is not recommended for regression models in which the regressor variables represent distinct physical variables [8, pages 297, 301–304]. However, in the case of the MSE-OLCs of NNs, the component NNs are essentially approximations of the *same* physical variable, $r(\vec{X})$. Hence, dropping *some* of the collinear regressor variables can be justified. Moreover, in some cases of NN-based modeling, a large number of trained NNs is produced during training. Consequently, there would be a high risk of overfitting to the combination data (lack of robustness), if all the available NNs are to be included in the MSE-OLC.

  Six algorithms for selecting which NNs to drop and which NNs to include in an MSE-OLC, are developed and discussed in Section 7.3. An empirical study that demonstrates the effectiveness of this approach in improving the robustness of the MSE-OLC is presented in Chapter 8.

7.2 Improving the Robustness of MSE-OLC by Restricting the Combination-Weights

Among the four MSE-OLCs discussed in Section 2.4.1, the unconstrained MSE-OLC with a constant term yields the theoretical minimal MSE, and thus is the best choice if one has perfect information. In other words, if the associated optimal combination-weights can be estimated accurately, then one should use the unconstrained MSE-OLC with a constant term.

In practice, the accuracy of estimating the optimal combination-weights is affected by many factors, including the presence of harmful collinearity. Thus, the unconstrained MSE-OLC with a constant term may not always be the best choice.

In the literature on combining forecasts, Clemen [17] suggests that it may indeed be appropriate to restrict the combination-weights, or to require no constant term, or both, if the restricted combination results in a more efficient (robust) forecast. Trenkler and Liski [102] further support Clemen's results. Holden and Peel [55] agree with Clemen on the need to constrain the combination-weights to sum to unity, but emphasize the role of the constant term in minimizing the within-sample squared prediction errors. Hence, the constant term may correct for bias in the component forecasts. Bunn [14] argues that while the unconstrained model should give the better fit to past data, constraints can improve the robustness of the combination in forecasting.

To examine the effectiveness of restricting the combination-weights to sum to unity and/or removing the constant term from the combination, let us re-examine the MSE-OLC in Example 3:

7.2.1 Example 3 (continued)

The original unconstrained MSE-OLC with a constant term, constructed in Section 6.5.1, yields an almost zero MSE on the (10 points) training data set and a true MSE of 91. Considering the other three forms of the MSE-OLCs in Section 2.4.1 (presented in order):

- The constrained MSE-OLC with a constant term:
  Using the training data to estimate the optimal combination-weights results in an MSE-OLC that yields an MSE of 0.000054 on the training data, which is larger than the MSE resulting from the unconstrained MSE-OLC with a constant term, as expected. However, the constrained MSE-OLC yields a true MSE of 8.2, which is 91% less than that of the unconstrained MSE-OLC. Thus, constraining the combination-weights may significantly improve the robustness of the MSE-OLC.

  A close inspection of the constrained MSE-OLC reveals that the scaled condition number of the associated covariance matrix, $\hat{\Psi}$ (defined in Section 3.3.1), equals 660, which is indeed much smaller than that of the covariance matrix associated with the unconstrained MSE-OLC with a constant term (the latter is 813000). Thus, the collinearity associated with the constrained MSE-OLC is much less severe than that associated with the unconstrained MSE-OLC (in this example).

  The constrained MSE-OLC is still worse than the best NN and the simple averaging of the six NNs, with an MSE that is about 1663% larger than that of the best NN, and about 1111% larger than that of the simple averaging.

- The unconstrained MSE-OLC without a constant term:
  Using the training data to estimate the optimal combination-weights results in an MSE-OLC that yields an MSE of 0.000046 on the training data, which is larger than the MSE resulting from the unconstrained MSE-OLC with a constant term, as expected. However, the unconstrained MSE-OLC without a constant term yields a true MSE of 20.4, which is 78% less than when the constant term is included in the combination. Moreover, the scaled condition number of the covariance matrix of the regressor variables has dropped from 813000 to 164942 as a result of the exclusion of the constant term. Thus, the exclusion of the constant term may lead to a more robust MSE-OLC.

  A close inspection of the collinearity structure associated with the unconstrained MSE-OLC, with the constant term included, reveals that the constant term is involved in the strongest collinearity among the regressor variables. This explains why dropping the constant term from the combination improves its

robustness. However, the current unconstrained MSE-OLC is still worse than the best NN and the simple averaging of the six NNs, with an MSE that is about 4308% larger than that of the best NN, and about 2927% larger than that of the simple averaging.

- The constrained MSE-OLC without a constant term:
  Using the training data to estimate the optimal combination-weights results in an MSE-OLC that yields an MSE of 0.000057 on the training data, which is larger than the MSE resulting from the unconstrained MSE-OLC with a constant term, as expected. However, the constrained MSE-OLC without a constant term yields a true MSE of 4.1, which is about 96% less than the unconstrained MSE-OLC with a constant term, and 50% less than the constrained MSE-OLC with a constant term. The scaled condition number of the associated covariance matrix of the regressor variables is 80, the is the smallest among the four MSE-OLCs. Thus, the robustness of the MSE-OLC has improved dramatically by dropping the constant and constraining the combination-weights to sum to unity.

  Like in the unconstrained case, the collinearity structure associated with the constrained MSE-OLC, with the constant term included, reveals that the constant term is involved in the strongest collinearity among the regressor variables. This explains why dropping the constant term from the combination improves its robustness.

  However, the resultant true MSE is still about 774% larger than that of the best NN, and about 500% larger than that of the simple averaging.

Thus, in this example, constraining the combination-weights to sum to unity and excluding the constant term from the combination yields the most robust MSE-OLC (so far). However, the performance of the current best MSE-OLC is far from being acceptable.

7.3 Improving the Robustness of MSE-OLC by Proper Selection of the Neural Networks

In this section, we introduce an approach for improving the robustness of the MSE-OLC through the proper selection of the NNs, guided by diagnostics of the collinearity among the y_j's (the outputs of the NNs) and/or among the δ_j's (the approximation errors of the NNs). Six algorithms based on this approach are developed.

These six selection algorithms rely on

1. Using BKW collinearity diagnostics, discussed in Section 6.4.2, to detect the presence of collinearity among the y_j's and/or among the δ_j's, to learn about the structure and relative strength of existing collinearities, and to identify the NNs involved in each collinearity.

2. Splitting the combination data set, $\mathcal{K}$, into an estimation data set, $\mathcal{K}_1$, and a cross-validation data set, $\mathcal{K}_2$. Then, the cross-validation approach, outlined in Section 6.6.2, is used for measuring robustness (the out-of-sample performance).

The information obtained from BKW collinearity diagnostics and from the cross-validation is utilized in deciding which NNs need to be dropped from the combination and which NNs stay in the combination.

A common feature of these algorithms is that they are "greedy," in the sense that they target the strongest collinearity. Once the NNs involved in the strongest collinearity are identified, the algorithms attempt to breakup this collinearity by dropping the *worst performer* from the combination. The *worst performer* is defined to be the NN that yields the largest MSE on $\mathcal{K}_2$. In any case, the algorithms NEVER drop the best NN from the combination.

The inputs to all the algorithms are

- The p trained NNs.
- The estimation data set $\mathcal{K}_1$.
- The cross-validation data set $\mathcal{K}_2$.

The algorithms sense the danger (harm) of existing collinearity in different manners. That is, they employ different criteria in deciding when to start dropping NNs from the combination and when to stop.

In employing these algorithms, one needs to keep in mind that the NNs may carry different information (knowledge). Hence, the more NNs that can be salvaged (included in the final combination), the better. The only reason for excluding some NNs (or likewise the constant term) from the MSE-OLC is the presence of harmful collinearity.

As illustrated in Section 7.2, constrained MSE-OLCs (C-OLCs) may be more robust than unconstrained MSE-OLCs (U-OLCs), especially for small samples. Thus, some of the algorithms make use of C-OLCs to improve robustness in the cases where the robustness of the U-OLC with a constant term is deemed "unsatisfactory".

The algorithms may be classified based on the type of OLCs they employ, which in turn affects their reliance on the collinearity diagnostics. Hence, the algorithms may be classified as

- Algorithms that employ U-OLCs:
  These are Algorithm A, Algorithm B, and Algorithm C. According to the formulations in Sections 3.2.1 and 3.2.3, the regressor variables in the regression model associated with U-OLCs are the y_j's. Thus, these algorithms rely mainly on diagnosing the collinearity among the y_j's. However, Algorithms B and C also rely on diagnosing the collinearity among the δ_j's as a secondary source of information.

- Algorithms that employ C-OLCs:
  These are Algorithm D, Algorithm E, and Algorithm K. Since the formulations in Sections 3.3.1 and 3.3.2 involve the δ_j's, these algorithms rely on diagnosing the collinearity among the δ_j's.

In all six algorithms, the performance of the simple averaging of the outputs of the component NNs and that of best NN, measured on a cross-validation set $\mathcal{K}_2$, are taken as a yard-stick to measure the robustness of the resultant combination. Upon termination, if the best combination that an algorithm produces by using its the selection procedure yields an inferior performance (on $\mathcal{K}_2$) to either the best NN or the simple averaging, then the algorithm selects its final outcome to be the best performer of the latter two.

Based on the dynamics that control their action against NNs deemed collinear, the algorithms may be classified into

- Conservative algorithms:
  Algorithms A, B, and D are considered conservative. These algorithms only drop NNs from the combination whenever the current MSE-OLC is deemed inferior to using the best NN or the simple averaging, as determined by their relative performance on the $\mathcal{K}_2$.

- Opportunistic algorithms:
  Algorithms C and E are considered opportunistic, since they always attempt to improve on an existing combination, even if its performance on $\mathcal{K}_2$ is better than those of the best NN and the simple averaging.

- Over-protective algorithm:
  Algorithm K is over-protective in the sense that it attempts to fight collinearity if it is associated with a scaled condition number deemed to be unacceptably high, even if there is no "detectable" harm as a result of the collinearity.

7.3.1 Algorithm A

As mentioned earlier, Algorithm A employs U-OLCs and relies solely on the information provided by the BKW diagnostics of collinearity among the y_j's.

Algorithm A proceeds as follows:

1. Determine the MSE of the best NN and of the simple averaging of the p NNs on $\mathcal{K}_2$.

2. Consider all the p NNs for the combination.

3. Form the U-OLC of all the considered NNs, including a constant term (unless a decision to exclude the constant has been taken earlier), using $\mathcal{K}_1$ to estimate the optimal combination-weights.

4. Determine the MSE of the U-OLC (from step 3) on $\mathcal{K}_2$.

5. If the U-OLC yields the lowest MSE on $\mathcal{K}_2$ compared to the best NN and the simple averaging, then STOP and return the current U-OLC.

6. Construct $\mathcal{G}_1$, the set of NNs involved in the strongest collinearity among the y_j's in the current combination, using BKW diagnostics.

7. If $\mathcal{G}_1$ has two or more elements, then

   - If there are more than two NNs in the current combination or if the constant term is not involved in the strongest collinearity, then drop the worst performer in $\mathcal{G}_1$ from the combination. Go to Step 8.
   - Else if the constant is involved in the strongest collinearity, then drop it from the combination.
   - Else STOP and return the best performer between the best NN and the simple averaging.

   Else ($\mathcal{G}_1$ has one element):

   - If this NN is the best NN, then
     - If the constant term has already been dropped from the combination (this indicates that there is no significant collinearity and that the largest condition index is associated mainly with the best NN), then STOP and return the best performer between the best NN and the simple averaging.
     - Else drop the constant term from the combination.
   - Else drop the NN in $\mathcal{G}_1$ from the combination.

8. If there are more than one NN left, then go to Step 3.
   Else STOP and return the best performer between the best NN and the simple averaging.

7.3.2 Algorithm B

As indicated earlier, Algorithm B employs U-OLCs and relies on the information provided by the BKW diagnostics of collinearity among y_j's and among δ_j's.

Algorithm B proceeds as follows:

1. Determine the MSE of the best NN and of the simple averaging of the p NNs on $\mathcal{K}_2$.

2. Consider all the p NNs for the combination.

3. Form the U-OLC of all the considered NNs, including a constant term (unless a decision to exclude the constant has been taken earlier), using $\mathcal{K}_1$ to estimate the optimal combination-weights.

4. Determine the MSE of the U-OLC (from step 3) on $\mathcal{K}_2$.

5. If the U-OLC yields the lowest MSE on $\mathcal{K}_2$ compared to the best NN and the simple averaging, then STOP and return the current U-OLC.

6. Construct $\mathcal{G}_1$, the set of NNs involved in the strongest collinearity among the y_j's in the current combination, using BKW diagnostics.

7. Construct $\mathcal{G}_2$, the set of NNs involved in the strongest collinearity among the δ_j's in the current combination, using BKW diagnostics. In case of the existence of collinearities that compete [1] with a strongest collinearity that has a scaled condition index deemed to be high, append $\mathcal{G}_2$ with the NNs that are involved in the closest competing collinearity.

8. If there are only two NNs in the current combination, then

   - If the strongest collinearity among the y_j's involves the constant term, then drop the constant term and go to Step 3.
   - Else STOP and return the best performer between the best NN and the simple averaging.

9. Construct $\mathcal{G}_{12}$, the intersection between the sets $\mathcal{G}_1$ and $\mathcal{G}_2$ [2].

10. If $\mathcal{G}_{12}$ is empty, then

    - If $\mathcal{G}_1$ has two or more elements, then
      - If there are more than two NNs in the current combination, then drop the worst performer in $\mathcal{G}_1$ from the combination. Go to Step 11.
      - Else if the constant is involved in the strongest collinearity, then drop it from the combination.
      - Else STOP and return the best performer between the best NN and the simple averaging.
    - Else ($\mathcal{G}_1$ has one element):
      - If this NN is the best NN, then
        * If the constant term has already been dropped from the combination (this indicates that there is no significant collinearity and that the largest condition index is associated mainly with the best NN), then STOP and return the best performer between the best NN and the simple averaging.

[1] According to Belsley [8, page 131–132], two or more collinearities are competing when they have scaled condition indexes of roughly the same order of magnitudes.

[2] The reason for the inclusion of the NNs involved in closest competing collinearity in $\mathcal{G}_2$ is to reduce the chance of an empty $\mathcal{G}_{12}$. A collinearity, that occurs simultaneously with the strongest collinearity, can confuse the role of some of the NNs that are common to both collinearities [8, page 117].

        * Else drop the constant term from the combination.
      - Else drop the NN in $\mathcal{G}_1$ from the combination.

  Else if $\mathcal{G}_{12}$ has only one element, then

  - If $\mathcal{G}_1$ has one element, then
    - If this NN is the best NN, then
      * If the constant term has already been dropped from the combination (this indicates that there is no significant collinearity and that the largest condition index is associated mainly with the best NN), then STOP and return the best performer between the best NN and the simple averaging.
      * Else drop the constant term from the combination.
    - Else drop the NN in $\mathcal{G}_1$ from the combination.
  - Else:
    - If the NN in $\mathcal{G}_{12}$ is the best NN then drop the worst NN in $\mathcal{G}_1$ from the combination.
    - Else drop the NN in $\mathcal{G}_{12}$ from the combination.

  Else Drop the worst NN in $\mathcal{G}_{12}$ from the combination.

11. If there are more than one NN left, then go to Step 3.
    Else STOP and return the best performer between the best NN and the simple averaging.

7.3.3 Algorithm C

As indicated earlier, Algorithm C employs U-OLCs and relies on the information provided by the BKW diagnostics of collinearity among y_j's and among δ_j's.

Like Algorithm B, Algorithm C drops NNs from a combination whenever the performance of the combination (measured by the MSE on $\mathcal{K}_2$) is worse than that of the best NN or the simple averaging. However, even when the performance of the combination is (or becomes) better than that of the best NN and the simple averaging, Algorithm C proceeds on with dropping more and more NNs as long as the performance of the combination is improving on $\mathcal{K}_2$.

Algorithm C proceeds exactly like Algorithm B, except for:

- Step 5 is replaced by

  If {this is not the first execution (of Step 5), AND the previous[3] U-OLC yields the lowest MSE on $\mathcal{K}_2$ compared to the best NN and

[3]During the first execution of Step 5, the previous U-OLC is set to the U-OLC from Step 3.

the simple averaging, AND the current[4] U-OLC is worse than the previous U-OLC }, then STOP and return the previous U-OLC.

- The statement, "STOP and return the best performer between the best NN and the simple averaging," is replaced by "STOP and return the best performer among the previous U-OLC, the best NN, and the simple averaging."

7.3.4 Algorithm D

As mentioned earlier, Algorithm D employs C-OLCs and relies solely on the information provided by the BKW diagnostic of collinearity among the δ_j's.

Algorithm D is identical to Algorithm A except for two features:

- When the robustness of the U-OLC of all the p NNs is deemed "unsatisfactory," Algorithm D adopts C-OLC instead of U-OLC in the subsequent steps.
- Instead of relying on a collinearity diagnosis for the y_j's, Algorithm D relies on diagnosing the collinearity among the δ_j's. Thus, instead of the set $\mathcal{G}_1$, a set $\mathcal{G}_2$ of all the NNs involved in the strongest collinearity among the δ_j's is used.

Algorithm D proceeds as follows:

1. Determine the MSE of the best NN and of the simple averaging of the p NNs on $\mathcal{K}_2$.
2. Consider all the p NNs for the combination.
3. If this is the first execution (of this step), then
   - Form the U-OLC of all the NNs, including a constant term, using $\mathcal{K}_1$ to estimate the optimal combination-weights.
   - Determine the MSE of the U-OLC on $\mathcal{K}_2$.
   - If the U-OLC yields the lowest MSE on $\mathcal{K}_2$ compared to the best NN and the simple averaging, then STOP and return the current U-OLC.

   Else
   - Form the C-OLC of all the NNs, including a constant term (unless a decision to exclude the constant has been taken earlier), using $\mathcal{K}_1$ to estimate the optimal combination-weights.
   - Determine the MSE of the C-OLC on $\mathcal{K}_2$.
   - If the C-OLC yields the lowest MSE on $\mathcal{K}_2$ compared to the best NN and the simple averaging, then STOP and return the current C-OLC.

[4]During the first execution of Step 5, the current U-OLC may be arbitrarily set, since this If-statement will always be false.

4. Construct $\mathcal{G}_2$, the set of NNs involved in the strongest collinearity among the δ_j's in the current combination, using BKW diagnostics.

5. If $\mathcal{G}_2$ has two or more elements, then

   - If there are more than two NNs in the current combination or if the constant term is not involved in the strongest collinearity, then drop the worst performer in $\mathcal{G}_2$ from the combination. Go to Step 6.
   - Else if the constant is involved in the strongest collinearity, then drop it from the combination.
   - Else STOP and return the best performer between the best NN and the simple averaging.

   Else ($\mathcal{G}_2$ has one element):

   - If this NN is the best NN, then
     - If the constant term has already been dropped from the combination (this indicates that there is no significant collinearity and that the largest condition index is associated mainly with the best NN), then STOP and return the best performer between the best NN and the simple averaging.
     - Else drop the constant term from the combination.
   - Else drop the NN in $\mathcal{G}_2$ from the combination.

6. If there are more than one NN left, then go to Step 3.
   Else STOP and return the best performer between the best NN and the simple averaging.

7.3.5 Algorithm E

As indicated earlier, Algorithm E employs C-OLCs and relies on the information provided by the BKW diagnostics of collinearity among the δ_j's.

Algorithm E is identical to Algorithm C except for two features:

- When the robustness of the U-OLC of all the NNs is deemed "unsatisfactory," Algorithm E adopts C-OLC instead of U-OLC in the subsequent steps.

- Instead of relying on a collinearity diagnosis for both the y_j's and the δ_j's, Algorithm E relies only on diagnosing the collinearity among the δ_j's. Thus, instead of the using the sets $\mathcal{G}_1$ and $\mathcal{G}_{12}$, a set $\mathcal{G}_2$ of all the NNs involved in the strongest collinearity among the δ_j's, is used. Notice that the definition of the set $\mathcal{G}_2$ in Algorithms D and E is different from that in Algorithms B and C.

Algorithm E proceeds as follows:

1. Determine the MSE of the best NN and of the simple averaging of the p NNs on $\mathcal{K}_2$.

2. Consider all the p NNs for the combination.

3. If this is the first execution (of this step), then

   - Form the U-OLC of all the NNs, including a constant term, using $\mathcal{K}_1$ to estimate the optimal combination-weights.
   - Determine the MSE of the U-OLC on $\mathcal{K}_2$.

   Else

   - Form the C-OLC of all the NNs, including a constant term (unless a decision to exclude the constant has been taken earlier), using $\mathcal{K}_1$ to estimate the optimal combination-weights.
   - Determine the MSE of the C-OLC on $\mathcal{K}_2$.

4. If {this is not the first execution (of this step), AND the previous[5] OLC yields the lowest MSE on $\mathcal{K}_2$ compared to the best NN and the simple averaging, AND the current[6] C-OLC is worse than the previous OLC }, then STOP and return the previous OLC.

5. Construct $\mathcal{G}_2$, the set of NNs involved in the strongest collinearity among the y_j's in the current combination, using BKW diagnostics.

6. If there are only two NNs in the current combination, then

   - If the strongest collinearity among the y_j's involves the constant term, then drop the constant term and go to Step 3.
   - Else STOP and return the best performer among the previous OLC, the best NN, and the simple averaging.

7. If $\mathcal{G}_2$ has two or more elements, then

   - Drop the worst performer in $\mathcal{G}_2$ from the combination.
   - Go to Step 8.

   Else ($\mathcal{G}_2$ has one element):

   - If this NN is the best NN, then

[5] During the first execution of Step 4, the previous OLC is set to the U-OLC from Step 3.

[6] During the first execution of Step 4, the current C-OLC may be arbitrarily set, since this If-statement will always be false.

  - If the constant term has already been dropped from the combination (this indicates that there is no significant collinearity and that the largest condition index is associated mainly with the best NN), then STOP and return the best performer among the previous OLC, the best NN, and the simple averaging.
  - Else drop the constant term from the combination.
- Else drop the NN in $\mathcal{G}_2$ from the combination.

8. If there are more than one NN left, then go to Step 3.
   Else STOP and return the best performer among the previous OLC, the best NN and the simple averaging.

7.3.6 Algorithm K

As indicated earlier, Algorithm K employs C-OLCs and relies on the information provided by the BKW diagnostics of collinearity among δ_j's.

Algorithm K is identical to Algorithm E except that Step 4 is replaced by

> If {this is not the first execution (of this step), AND the scaled condition number of the associated covariance matrix is deemed acceptable[7] AND the previous[8] OLC yields the lowest MSE on $\mathcal{K}_2$ compared to the best NN and the simple averaging, AND the current[9] C-OLC is worse than the previous OLC }, then STOP and return the previous OLC.

Thus, Algorithm K is intolerant of collinearity that results in an "unacceptably" large scaled condition number of the involved covariance matrix. Regardless of whether or not such collinearity causes detectable[10] deterioration in the robustness of the MSE-OLC.

7.3.7 Example 4 (continued)

Algorithms A, B, C, D, E, and K are applied to construct MSE-OLCs of the six NNs in Example 4 (Section 6.5.2). The training set (20 points) is used as the estimation set, $\mathcal{K}_1$, and the extra set of 10 points is used as the cross-validation set, $\mathcal{K}_2$.

The results obtained from the algorithms are

[7] An arbitrary threshold value for the scaled condition number may be assigned here. A scaled condition number is deemed acceptable only if it is below that threshold. A very large threshold value makes Algorithm K behaves like Algorithm E, while a very small threshold value makes it more intolerable to collinearity. We suggest a value between 100 and 1000 depending on the desired behavior of Algorithm K. In the examples discussed in this dissertation, the threshold is set to 500.

[8] During the first execution of Step 4, the previous OLC is set to the U-OLC from Step 3.

[9] During the first execution of Step 4, the current C-OLC may be arbitrarily set, since this If-statement will always be false.

[10] Detectable by measuring the performance on $\mathcal{K}_2$, which is the normal practice of all algorithms.

- Algorithms B and C have only kept NN3, NN4, and NN6 in the combination while the remaining three NNs are dropped out. The resultant true MSE is 0.0107 down from 0.100 when all the six NNs are included in the combination and all the available 30 points are used in estimating the associated optimal combination-weights.

  The true MSE resulting from applying Algorithms B or C is 67% less than that of NN6, the best NN, and about 86% less than that of the simple averaging. This indicates the Algorithms B and C can dramatically improve the robustness of the MSE-OLC.

- Algorithms A, D, E, and K recommend using the best NN instead of combining. However, this is not a bad choice since the best NN turns out to have a lower true MSE compared to the simple averaging.

These results indicate that by taking measures to detect and treat harmful collinearity, all the algorithms have successfully improved the robustness of the MSE-OLC. The resultant combinations are at least as good as the best performer between NN6 and the simple averaging.

The fact that some of the algorithms have succeeded in finding a combination better than using the best NN or the simple averaging, while others recommend either of the latter choices, shows that the path that an algorithm takes while dropping NNs, can sometimes lead to a dead-end, and consequently the algorithm has to choose between the best NN and the simple averaging.

7.3.8 Example 1 (continued)

In Sections 3.4 and 5.2.1, the U-OLC of the six trained NNs in Example 1 has resulted in a dramatic improvement in the model accuracy. However, as discussed in Section 6.3, the pairwise correlations among the y_j's are near unity, while the pairwise correlations among the δ_j's are mostly positive and above 0.5. Such high correlations may raise some concerns about the robustness of the U-OLC, and whether the exclusion of some of the collinear NNs may produce better results.

Using a cross-validation set of 25 uniformly distributed independent data points, all the algorithms conclude that the robustness of the U-OLC is satisfactory and thus no network needs to be dropped from the combination. Such performance is very pleasing, since we would like the selection algorithms to keep as many networks as possible in the combination when there is no harmful collinearity.

7.3.9 Modification to the algorithms

A key method for treating collinearity that is discussed in Section 7.1 is to introduce new data to the combination. Evidence from Examples 3 and 4, discussed in Sections 6.5.1 and 6.5.2 respectively, strongly support the use of extra data whenever available.

Using a cross-validation set to measure out-of-sample performance (robustness) of the combinations examined by the algorithms is critical to avoid being misled by the performance on the estimation data. However, once an algorithm finishes selecting which NNs to combine, re-integrating the two data sets, $\mathcal{K}_1$ and $\mathcal{K}_2$, for the estimation of the optimal combination-weights, in the final estimation step, may result in a better performance. To examine the validity of this last statement, consider the following examples.

7.3.9.1 Example 3 (continued)

In Sections 6.5.1 and 7.2.1, several approaches for constructing MSE-OLCs of the six trained NNs, generated in this example, are discussed. None of these approaches yields performance comparable to that of NN3, the best NN, or that of the simple averaging.

The (unmodified versions) of Algorithms A, B, C, D, E, and K result in

- Algorithms A, B, and C recommend using NN3 instead of combining.
- Algorithm D has selected NN3, NN4, and NN6, and the resultant C-OLC, with a constant term included, yields a *true* MSE of 0.525, which is about 77% less than the true MSE resulting from the C-OLC (with a constant term) that includes all the six NNs and uses all the 15 available points in estimating the optimal combination-weights. The resultant MSE is also about 22% less than the MSE resulting from the U-OLC (with a constant term) that includes all the six NNs and uses all the available 15 points, examined in Section 6.5.1. The resultant MSE is about 23% less than that of the simple averaging, however, is about 12% larger than that of NN3.
- Algorithms E and K have selected NN3 and NN6, and the resultant C-OLC, with a constant term, yields a *true* MSE of 0.535, which is of the same order of magnitude as that resulting from the combination constructed by Algorithm D. Thus, all the comments on Algorithm D apply here.

The modified versions of Algorithms A, B, C, D, E, and K result in:

- Algorithms A, B, and C recommend using NN3. Since there is no estimation of optimal combination-weights involved, there is no difference between the unmodified and the modified versions.
- Algorithm D: using $\mathcal{K}$ in estimating the optimal combination-weights in the final step yields a true MSE of 0.037, which is about 93% less than that resulting from the unmodified version.

  The resultant true MSE is about 92% less than that produced by NN3, and about 95% less than that produced by simple averaging. This shows a dramatic improvement in the robustness of the MSE-OLC.

- Algorithms E and K: using $\mathcal{K}$ in estimating the optimal combination-weights in the final step yields a true MSE of 0.034, which is of the same order of magnitude as that resulting from the combination constructed by the modified version of Algorithm D. Thus, all the comments on the modified version of Algorithm D apply here.

From these results, the modified versions of the algorithms appear to be superior to the unmodified versions. Besides, all the algorithms result in robust MSE-OLCs.

7.3.9.2 Example 4 (continued)

In Section 7.3.7, the unmodified versions of the algorithms have been examined. To measure the impact of extending the estimation set to include all the available data in the last step, the modified versions of the algorithms are tried.

The results of applying the modified versions of the algorithms are

- Algorithms B and C: using $\mathcal{K}$ in estimating the optimal combination-weights in the final step yields a true MSE of 0.0087, which is about 19% less than that resulting from the unmodified version.

  The resultant true MSE is about 73% less than that produced by NN3, and about 89% less than that produced by simple averaging. This shows a dramatic improvement in the robustness of the MSE-OLC.

- Algorithms A, D, E, and K recommend using the best NN. Since there is no estimation of optimal combination-weights involved, there is no difference between the unmodified and the modified versions.

These results support the findings in Example 3 in Section 7.3.9.1. That is, the modified versions of the algorithms appear to be superior to the unmodified versions. Besides, all the algorithms result in robust MSE-OLCs.

7.3.10 Conclusions

The algorithms developed in this section demonstrate great aptitude for dealing with collinearity problems that undermine the robustness of MSE-OLCs.

Modifying the algorithms to include all the available data in the final estimation step of the optimal combination-weights appears to be significantly better than excluding the cross-validation data.

In order to further examine the merits of all the algorithms, an empirical study is conducted in Chapter 8.

8. EMPIRICAL STUDY

In this chapter, we present an empirical study to investigate the merits of MSE-OLCs and the effectiveness of the selection algorithms, developed in Section 7.3, in improving the robustness of the MSE-OLC.

The empirical study consists of two parts. The first part, presented in Section 8.1, examines the effect of the size and quality[1] of the combination data set, $\mathcal{K}$, on the robustness of the MSE-OLC and the effectiveness of the algorithms. The second part, presented in Section 8.2 examines the effects of the size and quality of available data on training the NNs, as well as, on the robustness of the MSE-OLC, and on the effectiveness of the algorithms. The main conclusions of the empirical study are summarized in Section 8.3.

In the empirical study, the performance of a given combination is expressed in terms of the percentage reduction (or increase) in the true[2] MSE compared to the best[3] NN and compared to the simple averaging of all the NNs.

8.1 Part I: Effect of the Size and Quality of the Combination Data

Consider approximating the function $r_3(X) = \sin[2\,\pi\,(1-X)^2]$, where $X \in [0,1]$. The range of $r_3(X)$ is $[-1,1]$.

8.1.1 Training the neural networks

Ten independent training sets, $\mathcal{T}_1, \ldots, \mathcal{T}_{10}$, each having 10 uniformly distributed independent points, are generated.

Ten experiments (replications) are then conducted. Each replication uses one separate $\mathcal{T}_j$, and the same copy of six NNs (NN1, NN2, NN3, NN4, NN5, and NN6) that are initialized with independent random connection-weights uniformly distributed in [–0.3, 0.3]. NN1 and NN2 are 1–3–1 NNs, NN3 and NN4 are 1–2–2–1 NNs, NN5 and NN6 are 1–4–1 NNs. The activation function for the hidden units as well as the output units is the logistic sigmoid function $g(s) = (1+e^{-s})^{-1}$.

In each replication, the six NNs are trained using $\mathcal{T}_j$, as a common training set for all the NNs. Training is carried out for 2000 iterations using the Error Backpropagation algorithm with a learning rate of 0.25.

[1] Quality is related to the amount of additive noise in the observed $r(\vec{X})$, in $\mathcal{K}$.

[2] "True" means that it is computed relative to the true (known) response function.

[3] As defined in Section 6.6.2.

Since a separate $\mathcal{T}_j$ is used for each replication, ten different sets (groups) of the six NNs are produced at the end of the training process; a total of sixty trained NNs.

8.1.2 Combining the neural networks

To examine the effect of the size and the quality of the combination data set, $\mathcal{K}$, four levels of testing are tried in a two-factors full-factorial experimental design fashion. The sizes and/or the quality of the combination data sets are varied across the four levels. Levels 2 and 4 have more data than Levels 1 and 3. Levels 1 and 2 have combination data sets that are not corrupted with additive noise, while the combination data used in Levels 3 and 4 are corrupted with 10% additive Gaussian noise, that is an $N[0., (0.2)^2]$ is added to the true response $r_3(X)$.

8.1.3 Part I—Level 1

Ten independent data sets, $\mathcal{C}_1, \ldots, \mathcal{C}_{10}$, each having 5 uniformly distributed independent points, are generated. Each set is assigned to a different replication and is used for cross-validation as discussed in Sections 6.6.2 and 7.3.9. The same $\mathcal{T}_j$ used for training the NNs in a given replication is used for estimating the optimal combination-weights. Thus, for the jth replication; $\mathcal{K}_1 = \mathcal{T}_j$, and $\mathcal{K}_2 = \mathcal{C}_j; j = 1, \ldots, 10$.

In each replication, two U-OLCs with constant terms are constructed. One of the U-OLCs, U-OLC 1, uses only the associated $\mathcal{K}_1$ for estimating the optimal combination-weights, while the second U-OLC, U-OLC 2, merges $\mathcal{K}_1$ and $\mathcal{K}_2$ into one set used for estimating the optimal combination-weights.

In each replication, the original and the modified versions of the six algorithms, discussed in Section 7.3, are tried.

The performance of each U-OLC and the individual algorithms is averaged across the ten replications, and the results are presented in Table 8.1. Part (a) of Table 8.1 shows the mean performance of U-OLC 1 and the original versions of the algorithms. Part (b) of Table 8.1 shows the mean performance of U-OLC 2 and the modified versions of the algorithms. The standard error in the mean performance is shown in brackets to the right of the mean performance.

Some important remarks on the results shown in Table 8.1 are

1. Employing the six algorithms in combining the trained NNs results in a significant and dramatic improvement in the model accuracy. While U-OLC 2 yields the best performance among all the examined combining procedures, employing OLC methods without proper assessment of the robustness of the resulting combination can have serious consequencies (as demonstrated by U-OLC 1 in Table 8.1 (a), see also the discussions on Examples 3 and 4 in Chapter 7).

2. U-OLC 1 results in an astronomical mean percentage increase in the true MSE compared to the best NN and the simple averaging. In nine out of the ten replications, applying U-OLC 1 results in a significant percentage reduction in MSE that is above 40% compared to the best NN, and above 28% compared to

Table 8.1 Part I: level 1: (a) Original algorithms. (b) Modified algorithms.

	Mean % reduction in MSE compared to best NN	Mean % reduction in MSE compared to simple averaging
U-OLC 1	-23277 (23369)	-45874 (45966)
Algorithm A	89 (7)	83 (12)
Algorithm B	89 (7)	83 (12)
Algorithm C	87 (7)	81 (11)
Algorithm D	89 (7)	83 (12)
Algorithm E	92 (5)	87 (10)
Algorithm K	41 (7)	45 (7)

(a)

	Mean % reduction in MSE compared to best NN	Mean % reduction in MSE compared to simple averaging
U-OLC 2	99 (1)	99 (1)
Algorithm A	94 (5)	89 (10)
Algorithm B	94 (5)	89 (10)
Algorithm C	92 (5)	88 (10)
Algorithm D	94 (5)	89 (10)
Algorithm E	93 (5)	88 (10)
Algorithm K	44 (8)	48 (8)

(b)

the simple averaging. However, in the remaining replication, U-OLC 1 results in an astronomical increase in the true MSE compared to the best NN and the simple averaging. Such a catastrophic situation is a result of overfitting to the data used in estimating the optimal combination-weights in the presence of harmful collinearity, a situation that can be detected by using cross-validation, as discussed in Section 6.6.2. This situation is similar to the situations in Examples 3 and 4 discussed in Chapter 7.

Although the original versions of the six algorithms use the same estimation data set as U-OLC 1, their mean relative performances, compared to the best NN and the simple averaging, are dramatically better than the mean performance of U-OLC 1.

3. The high standard error in the mean performance of U-OLC 1, compared to the value of the mean performance, reflects the high variability in the performance of U-OLC 1 across the ten replications. Hence, without proper assessment of the robustness of the MSE-OLC, its performance becomes vulnerable to data problems, especially collinearity problems.

4. Comparing the performance of U-OLC 1 to the performance of U-OLC 2 reveals that including new data into the combination helps in the treatment of harmful collinearity.

5. Comparing the performance of the modified versions of the individual algorithms to those of the original versions reveals that including all the available data in the last estimation step is beneficial. The performance of the individual modified versions is better than that of the corresponding original version for all the algorithms.

6. U-OLC 2 performs (slightly) better than the modified versions of Algorithms A, B, C, D, and E, which indicates that in applying the algorithms, one trades-off possible improvement in robustness when harmful collinearity is detected, with possible loss of the benefits from utilizing all the NNs in the combination.

7. Algorithm K performs modestly compared to Algorithms A, B, C, D, and E. Thus, indicating that being over-protective in treating collinear NNs can result in dropping NNs that carry useful information from the combination.

8.1.4 Part I—Level 2

In this level, the same trained NNs in Level 1 are considered. However, more data points are made available for combining the NNs (compared to Level 1). Ten independent data sets, $\mathcal{K}_1$, each having 20 uniformly distributed independent points, are generated. Also, ten independent data sets, $\mathcal{K}_2$, each having 10 uniformly distributed independent points, are generated. A pair of $\mathcal{K}_1$ and $\mathcal{K}_2$ is assigned to each replication.

For every replication, U-OLC 1 and U-OLC 2 are constructed in the same manner as in Section 8.1.3. Also, the original and modified versions of the six algorithms are tried. The results are shown in Table 8.2.

Some important remarks on the results shown in Table 8.2 are

1. All the algorithms as well as U-OLC 1 and U-OLC 2 have substantially reduced the MSE compared to the best NNs and the simple averaging.

2. Algorithm K performs modestly compared to the other algorithms due to its over-protective nature, as pointed out in Section 8.1.3.

3. The modified versions of the algorithms perform slightly better than the original versions.

4. The percentage reductions in the *true* MSE compared to the best NN and compared are above 99.8% for the two U-OLCs and both the original and the modified versions of all the algorithms except for Algorithm K. These are very high mean percentage reductions in the true MSE, and thus deserve a closer inspection. One of the replications in Level 2 is studied in the following example.

8.1.4.1 Example 5

In one of the replications from the ten replications examined in Level 2, the resultant MSE from using the best NN is 0.2190, and that from using the simple averaging is 0.2409. The resultant MSE from using U-OLC 2 is 0.000017, which is a dramatic reduction compared to the best NN and the simple averaging.

Figure 8.1 shows $r_3(X)$ plotted against the approximations resulting from using the best NN, the simple averaging, and using U-OLC 2, in this example.

Both the resultant MSE and Figure 8.1 confirm that U-OLC 2 has dramatically improved the accuracy of the NN based model.

Unlike Example 1, Section 3.4, the NNs in this example are not well-trained. This is evident from the graph of the best NN as well as from its resulting RMS, which is 0.47. Also, unlike Example 1, where the optimal combination-weights sum to unity and the constant term is statistically insignificant, the optimal combination-weights in this example sum to -70 with a statistically significant constant term equal to 16. This indicates that the U-OLC 2 has used the trained NNs as *crude* building blocks (or bases) to model $r_3(X)$ on its own. Obviously, the U-OLC 2 has succeeded in this task.

Thus, beside adding the "final touches" to a set of NNs that are well-trained, as in Example 1, the MSE-OLC is also capable of successfully creating a very good model using a set of NNs that are not well-trained.

Table 8.2 Part I: level 2 :(a) Original algorithms. (b) Modified algorithms.

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 1	99.8	(0.1)	99.8	(0.1)
Algorithm A	99.8	(0.1)	99.8	(0.1)
Algorithm B	99.8	(0.1)	99.8	(0.1)
Algorithm C	99.8	(0.1)	99.8	(0.1)
Algorithm D	99.8	(0.1)	99.8	(0.1)
Algorithm E	99.8	(0.1)	99.8	(0.1)
Algorithm K	52.8	(7.4)	58.8	(6.1)

(a)

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 2	99.9	(0.1)	100	(0)
Algorithm A	99.9	(0.1)	100	(0)
Algorithm B	99.9	(0.1)	100	(0)
Algorithm C	99.9	(0.1)	100	(0)
Algorithm D	99.9	(0.1)	100	(0)
Algorithm E	99.9	(0.1)	100	(0)
Algorithm K	53.4	(7.7)	59	(6)

(b)

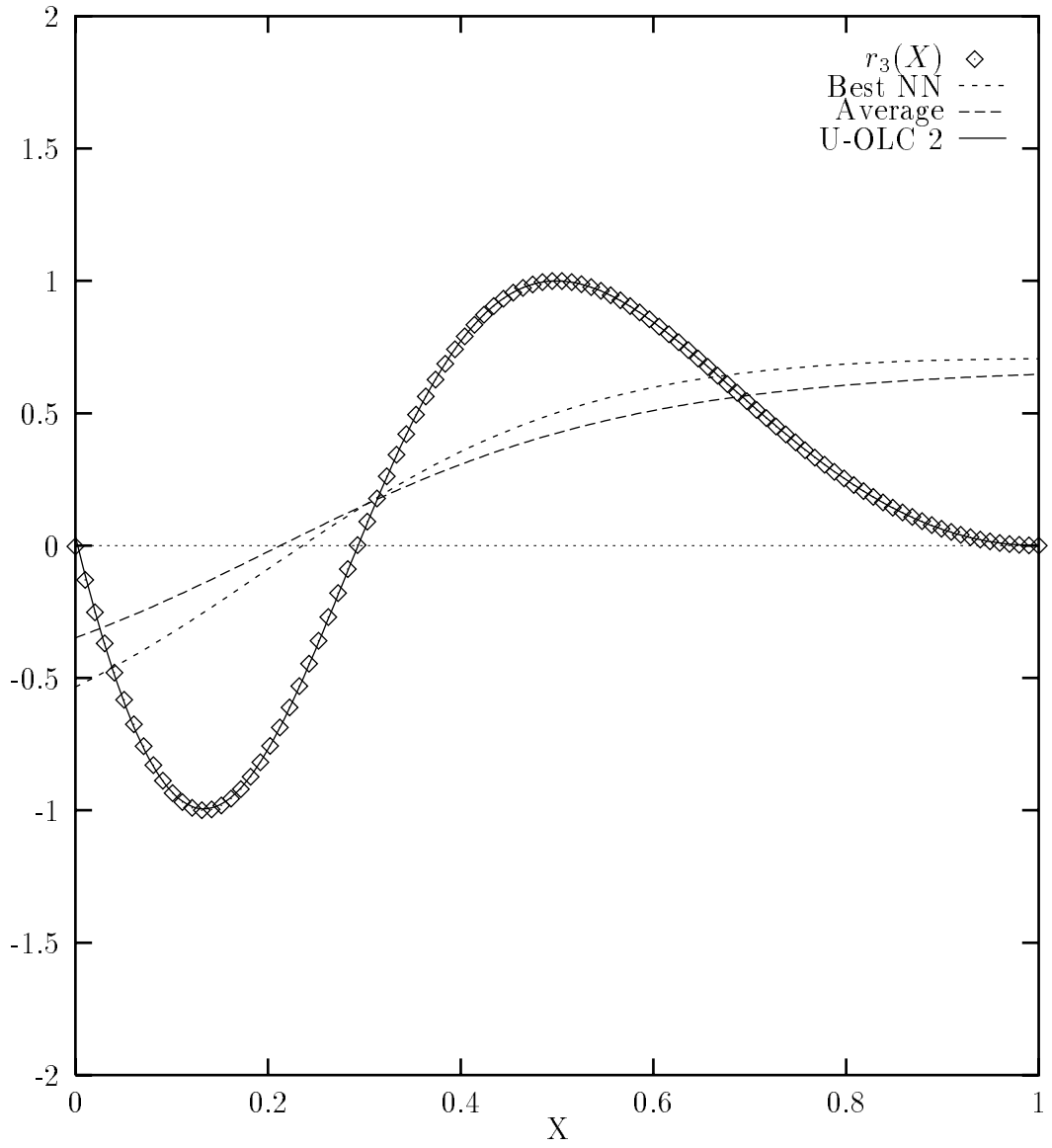

Figure 8.1 The function $r_3(X)$ and the approximations obtained using the best NN, the simple averaging, and U-OLC 2

8.1.5 Part I—Level 3

In this level, the same trained NNs in Level 1 are considered. Also, the combination data are identical to those used in Level 1, except that they are corrupted with 10% additive Gaussian noise, that is an $N[0., (0.2)^2]$ is added to the true response $r_3(X)$.

The two U-OLCs and the original and modified versions of the six algorithms are tried. The results are shown in Table 8.3.

Some important remarks on the results shown in Table 8.3 are

1. The presence of noise in the combination data significantly reduces the performances of the two U-OLCs and the original and modified versions of all the algorithms, except Algorithm K, compared to the results obtained using the uncorrupted version of the combination data in Table 8.1.

2. Although the performance of Algorithm K is (almost) unaffected by the presence of noise, it is still modest compared to the performances of the other algorithms.

3. Both the original and the modified versions of the algorithms yield significant mean percentage reductions in the true MSE compared to the best NN and the simple average. However, both U-OLCs fail to do the same. This suggests that the algorithms are more robust, in the presence of noise, compared to the U-OLCs.

4. Comparing the performance of the modified versions of the individual algorithms to those of the original versions reveals that including all the available data in the last estimation step is beneficial for almost all the algorithms, especially the conservative ones. Only Algorithm C, which is the best performer among the original versions, suffers a minor drop in its performance when more data are included in the final estimation step.

8.1.6 Part I—Level 4

In this level, the same trained NNs in Level 1 are considered. Also, the available combination data are identical to those used in Level 2, except that they are corrupted with 10% additive Gaussian noise, that is an $N[0., (0.2)^2]$ is added to the true response $r_3(X)$.

The two U-OLCs and the original and modified versions of the six algorithms are tried. The results are shown in Table 8.4.

Some important remarks on the results shown in Table 8.4 are

1. The presence of noise in the combination data significantly reduces the performances of the two U-OLCs and the original and modified versions of all the algorithms, except Algorithm K, compared to the results obtained using the uncorrupted version of the combination data, in Table 8.2. This confirms the first remark in Section 8.1.5.

Table 8.3 Part I: level 3: (a) Original algorithms. (b) Modified algorithms.

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 1	-1297295	(1297340)	-2551690	(2551737)
Algorithm A	52	(17)	48	(21)
Algorithm B	52	(17)	48	(21)
Algorithm C	70	(11)	70	(12)
Algorithm D	50	(16)	47	(21)
Algorithm E	67	(10)	67	(11)
Algorithm K	37	(10)	41	(10)

(a)

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 2	-166	(245)	-401	(484)
Algorithm A	75	(7)	73	(10)
Algorithm B	75	(7)	73	(10)
Algorithm C	68	(11)	68	(12)
Algorithm D	74	(7)	72	(10)
Algorithm E	69	(10)	68	(11)
Algorithm K	41	(11)	44	(11)

(b)

Table 8.4 Part I: level 4: (a) Original algorithms. (b) Modified algorithms.

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 1	63	(16)	67	(15)
Algorithm A	71	(16)	74	(14)
Algorithm B	70	(16)	73	(14)
Algorithm C	77	(14)	82	(9)
Algorithm D	71	(16)	74	(14)
Algorithm E	80	(14)	86	(9)
Algorithm K	60	(8)	63	(8)

(a)

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 2	74	(20)	83	(12)
Algorithm A	75	(20)	83	(12)
Algorithm B	74	(20)	83	(12)
Algorithm C	72	(20)	80	(12)
Algorithm D	74	(20)	83	(12)
Algorithm E	75	(20)	83	(12)
Algorithm K	60	(8)	63	(8)

(b)

2. Although the performance of Algorithm K has actually improved in the presence of noise, it is still modest compared to the performances of the other algorithms.

   This confirms the second remark in Section 8.1.5.

3. Compared to the results in Level 3, having more data available for combining the NNs has dramatically improved the performances of the two U-OLCs. The performances of the original versions of all the algorithms as well as the modified versions of Algorithm K have also significantly improved. However, the changes in the performances of the modified versions of all the algorithms, except for Algorithm K, are less significant.

4. Both the original and the modified versions of the algorithms yield dramatic improvement in the model accuracy compared to using the best NN and the simple averaging.

8.2 Part II: Effect of the Size and Quality of Data on Training and Combining Neural Networks

In this part, there are also four levels analogous to Levels 1–4 in Part I. Each level employs the same sets of $\mathcal{K}_1$'s and $\mathcal{K}_2$'s as the corresponding level in Part I. The two differences between Parts I and II are in training the NNs in the four levels:

- NN Training Time:
  The NNs employed in Part II are initially generated identical to those used in Part I. However, they are trained for 5000 iterations instead of only 2000 iterations in Part I.

- NN Training Data:
  In a given level in Part II, the NNs are trained using the ten sets of $\mathcal{K}_1$'s assigned to that level. Unlike Part I, where in all the four levels, the NNs are trained using the same ten $\mathcal{T}_j$ sets. Thus, in Part II, a total of 240 different trained NNs are produced.

8.2.1 Part II—Level 1

This level is identical to Level 1 in Part I, except that the NNs are trained longer before the combinations are constructed.

For each replication, U-OLC 1 and U-OLC 2 are constructed in the same manner as in Section 8.1.3. Also, the original and modified versions of the six algorithms are tried. The results are shown in Table 8.5.

Some important remarks on the results shown in Table 8.5 are

1. Both the original and the modified versions of all the algorithms dramatically improve the NN based model accuracy compared to using the best NN and the simple averaging.

Table 8.5 Part II: level 1: (a) Original algorithms. (b) Modified algorithms.

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 1	-2003	(1944)	-1303	(1337)
Algorithm A	78	(11)	87	(7)
Algorithm B	78	(11)	87	(7)
Algorithm C	77	(11)	87	(7)
Algorithm D	71	(15)	84	(8)
Algorithm E	70	(15)	83	(8)
Algorithm K	48	(14)	70	(8)

(a)

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 2	81	(14)	87	(10)
Algorithm A	81	(11)	89	(7)
Algorithm B	81	(11)	89	(7)
Algorithm C	81	(11)	88	(7)
Algorithm D	87	(8)	94	(4)
Algorithm E	86	(8)	93	(4)
Algorithm K	63	(11)	79	(6)

(b)

2. The performance of the modified version of the individual algorithms is generally better than that of the original version. This improvement confirms that the inclusion of all the available data in the last estimation step is beneficial.

3. Compared to the results in Table 8.1, U-OLC 1 appears to do significantly better. This suggests that having better trained component NNs reduces the risk of a catastrophic MSE-OLC.

4. Although the algorithms tend to sacrifice the inclusion of collinear NNs in the combination to reduce the ill effects of collinearity, the modified versions of Algorithms D and E significantly outperformed U-OLC2, where the latter is already a very good performer compared to the best NN and the simple averaging.

In five out of the ten replications examined in this level, each of the above combining procedures results in optimal combination-weights that almost sum to unity with the constant term being close to zero. In the five remaining replications, the absolute values of the sums of the optimal combination-weights are below 15 and the absolute values of the constant terms are below 6. However, in Level 1 in Part I, the absolute values of the sums of the optimal combination-weights are usually higher by orders of magnitude. Also, the absolute values of the constant terms are usually far from being zero. Based on the discussions in Sections 3.4 and 8.1.4, and keeping in mind that the NNs in Part II are trained for a relatively longer period of time, one may deduce that the OLCs in Part II are adding "final touches" on a set of well-trained NNs, while in Part I the OLCs are more involved in modeling the combination data since the NNs are not well-trained. To further support this claim, consider the following example

8.2.1.1 Example 6

Consider the six NNs treated in Example 5 in Section 8.1.4.1. In Part II, these NNs are trained for an extra 3000 iterations under the same training conditions as in Part I. However, the combination data sets are the ones used in Part I Level 1, and not the larger ones used in Part I Level 2.

The resultant *true* MSE from using the best NN is 0.0445 (down from 0.2190 in Example 5), and that from using the simple averaging is 0.0723 (down from 0.2409 in Example 5). Likewise, the MSEs resultant from all the remaining NNs are lower than those in Example 5. Thus, the NNs in this example are trained better than those in Example 5.

The resultant true MSE from using U-OLC 2 is 0.000168, which is a dramatic reduction compared to that resulting from using the best NN or the simple averaging. Figure 8.2 shows $r_3(X)$ plotted against the approximations resulting from using the best NN, the simple averaging, and using U-OLC 2 in this example. Both the resultant MSE and Figure 8.2 confirm that the MSE-OLC has dramatically improved the model accuracy.

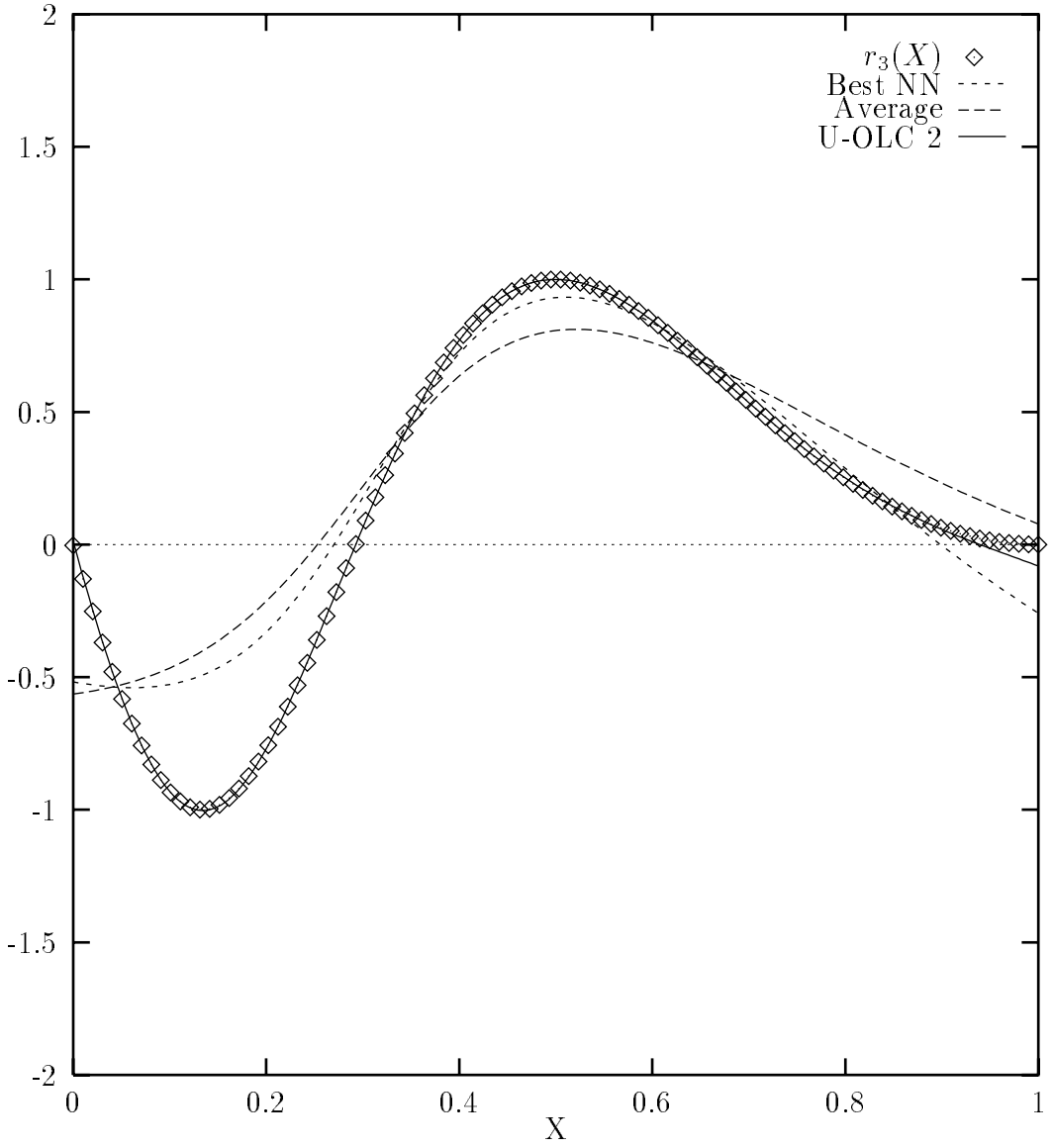

Figure 8.2 The function $r_3(X)$ and the approximations obtained using the best NN, the simple averaging, and U-OLC 2

Comparing Figures 8.1 and 8.2 and from the computed MSEs, the approximation accuracies of the best trained NN and the simple averaging of the six NNs in this example are much better than those in Example 5.

The sum of the optimal combination-weights is 1.0 (compared to -70 in Example 5), and the the constant term is -0.029 (compared to 16 in Example 5). Thus, while in both Examples, U-OLC 2 dramatically reduces the MSE compared to the best NN and the simple averaging, yet its roles in these two examples are apparently different. The changes in the magnitudes of sum of the optimal combination-weights and the constant term reveal the shift in the role of U-OLC 2 as the component NNs become well-trained.

8.2.2 Part II—Level 2

This level uses the same combination data as Level 2 in Part I. However, as mentioned earlier, the NNs are trained using different data sets and for a longer time.

For each replication, U-OLC 1 and U-OLC 2 are constructed in the same manner as in Section 8.1.4. Also, the original and modified versions of the six algorithms are tried. The results are shown in Table 8.6.

Some important remarks on the results shown in Table 8.6 are

1. Remarks 1–3 on Level 2 in Part I in Section 8.1.4 also apply here.

2. UOLC-1 slightly outperforms the original versions of the algorithms, and also U-OLC 2 outperforms the modified versions of the algorithms. This demonstrates that the U-OLCs are not necessarily inferior to the OLCs produced by the algorithms, especially in the presence "adequate[4]" combination data are available. The main concern with employing the U-OLCs is their robustness, especially in the presence of harmful collinearity.

8.2.3 Part II—Level 3

This level uses the same combination data as Level 3 in Part I. However, as mentioned earlier, the NNs are trained using different data sets and for a longer time.

For each replication, U-OLC 1 and U-OLC 2 are constructed in the same manner as in Section 8.1.5. Also, the original and modified versions of the six algorithms are tried. The results are shown in Table 8.7.

Some important remarks on the results shown in Table 8.7 are

1. The presence of noise in the combination data significantly reduces the performances of the two U-OLCs and the original and modified versions of all the algorithms, except Algorithm K, compared to the results obtained using the uncorrupted version of the combination data, in Table 8.5.

[4]Adequate in number and quality.

Table 8.6 Part II: level 2: (a) Original algorithms. (b) Modified algorithms.

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 1	54	(12)	69	(7)
Algorithm A	46	(14)	63	(10)
Algorithm B	46	(14)	63	(10)
Algorithm C	49	(13)	64	(9)
Algorithm D	46	(14)	63	(9)
Algorithm E	48	(13)	64	(9)
Algorithm K	17	(8)	39	(7)

(a)

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 2	70	(8)	79	(5)
Algorithm A	56	(13)	68	(10)
Algorithm B	56	(13)	68	(10)
Algorithm C	49	(13)	68	(10)
Algorithm D	57	(12)	69	(9)
Algorithm E	54	(12)	68	(9)
Algorithm K	24	(10)	44	(8)

(b)

Table 8.7 Part II: level 3: (a) Original algorithms. (b) Modified algorithms.

	Mean % reduction in MSE compared to best NN	Mean % reduction in MSE compared to simple averaging
U-OLC 1	-83416 (83163)	-56994 (56881)
Algorithm A	31 (13)	48 (9)
Algorithm B	31 (13)	48 (9)
Algorithm C	39 (11)	55 (8)
Algorithm D	33 (13)	49 (9)
Algorithm E	40 (11)	54 (8)
Algorithm K	45 (10)	57 (8)

(a)

	Mean % reduction in MSE compared to best NN	Mean % reduction in MSE compared to simple averaging
U-OLC 2	-678 (749)	-437 (512)
Algorithm A	44 (13)	57 (9)
Algorithm B	45 (13)	58 (10)
Algorithm C	40 (12)	53 (9)
Algorithm D	52 (11)	62 (9)
Algorithm E	49 (11)	60 (8)
Algorithm K	51 (11)	62 (9)

(b)

2. Remarks 2 and 3 on the results of Level 3 in Part I in Section 8.1.5 also apply here.

3. Algorithm K demonstrates a significant robustness to noise. It is the best performer among the original versions. Moreover, its modified version almost ties with the best performer among the modified versions, that is also the best performer among all the considered combining procedures.

8.2.4 Part II—Level 4

This level uses the same combination data as Level 4 in Part I. However, as mentioned earlier, the NNs are trained using different data sets and for a longer time.

For each replication, U-OLC 1 and U-OLC 2 are constructed in the same manner as in Section 8.1.6. Also, the original and modified versions of the six algorithms are tried. The results are shown in Table 8.8.

Some important remarks on the results shown in Table 8.8 are

1. The presence of noise in the combination data significantly reduces the performances of the two U-OLCs and the original and modified versions of all the algorithms.

   However, while the two U-OLCs perform poorly compared to the best NN and the simple averaging, all the OLCs produced by the original and the modified versions of the selection algorithms outperform the best NN and the simple averaging.

2. Algorithms A and B, which have a minor difference in their strategies, ended up being the worst and the best algorithms, respectively, both among the original and the modified versions of the algorithms. The only difference between the two algorithms is that Algorithm B considers the collinearity among the δ_j's (the approximation errors of the NNs) as a secondary source of information beside the collinearity among the y_j's (the outputs of the NNs), while Algorithm A relies only on diagnosing the latter type of collinearity. This indicates the value of analyzing both types of collinearities before deciding which of the collinear NNs to exclude from the combination.

8.3 Main Conclusions of the Empirical Study

In the empirical study conducted in Sections 8.1 and 8.2, two U-OLCs and the two versions of the six algorithms developed in Section 7.3 are considered. Each of these fourteen procedures is tried in combining NNs a total of 80 times. These combinations involve a total of 300 different trained NNs. The training and combination data consist of: 40 independent data sets containing a total of 300 uniformly distributed independent points; and 40 data sets generated from the former ones by adding 10% independent Gaussian noise to the true response $r_3(X)$.

Table 8.8 Part II: level 4: (a) Original algorithms. (b) Modified algorithms.

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 1	-264	(279)	-94	(114)
Algorithm A	18	(10)	32	(7)
Algorithm B	36	(11)	44	(10)
Algorithm C	34	(10)	42	(9)
Algorithm D	25	(11)	38	(8)
Algorithm E	22	(10)	35	(7)
Algorithm K	17	(8)	31	(6)

(a)

	Mean % reduction in MSE compared to best NN		Mean % reduction in MSE compared to simple averaging	
U-OLC 2	2	(27)	22	(20)
Algorithm A	15	(11)	31	(9)
Algorithm B	34	(12)	42	(11)
Algorithm C	31	(12)	40	(10)
Algorithm D	23	(12)	37	(10)
Algorithm E	21	(11)	35	(9)
Algorithm K	21	(10)	35	(8)

(b)

The main conclusions of the empirical study conducted in Sections 8.1 and 8.2 are

1. Although the U-OLC 2 yields the theoretical minimal MSE among all the combination methods, in practice the robustness of the estimated U-OLC 2 may be seriously undermined due to the presence of harmful collinearity. However, if adequate[5] combination data are available, the U-OLC 2 meets the theoretical expectations, as evident from Tables 8.1, 8.2, and 8.6.

2. Both the original and the modified versions of the six algorithms developed in Section 7.3 yield significant mean percentage reductions in the true MSE compared to using the best NN and the simple averaging. Compared to the performance of the two U-OLCs, these algorithms significantly increase the robustness of the MSE-OLC for a given (fixed) data set.

3. Comparing the performance of the modified versions of the individual algorithms to those of the original versions reveals that including all the available data in the last estimation step is beneficial. Almost always, the individual modified versions of the algorithms yields better performance compared to the corresponding original versions.

4. Due to the over-protective nature of Algorithm K, it performs modestly compared to the other algorithms. However, it is less sensitive to noisy data.

5. No algorithm can be considered a clear winner among the six algorithms. In general, the algorithms yield close results, with the exception of Algorithm K. Also, every algorithm is a top-winner in at least one level in the empirical study.

   In practice, and especially when the estimated U-OLC is deemed to have poor robustness, some or all of the modified versions of the algorithms need to be tried. Since the true function is generally unknown, the relative performance of the algorithms, as well as that of the U-OLC, may be measured using a testing data set, separate from the combination data set, in order to choose a winner.

6. The OLCs appear to have a role that is different from just adding some "final touches" on a set of trained NNs. The discussions in Sections 3.4, 8.1.4, and 8.2.1 reveal that OLCs become more involved in modeling the combination data when the NNs are not well-trained. As a result the sum of the optimal combination-weights becomes significantly different from unity, and the constant term becomes significantly different from zero. This is the case in almost all the replications in Part I of the empirical study, as well as in some of the replications in Part II, where the NNs are trained longer. However, in most of the replications in Part II, the sum of the optimal combination-weights is near unity and the constant term is near zero, indicating that the NNs are (relatively) well-trained.

[5] Adequate in number and quality.

9. SUMMARY, CONCLUSIONS, RECOMMENDATIONS, AND FUTURE DIRECTIONS

9.1 Summary

In this dissertation, we develop a framework for constructing robust MSE-optimal linear combinations (MSE-OLCs) of neural networks. The merits of MSE-OLCs are investigated, with several benefits and risk-factors highlighted. We introduce six selection algorithms to enhance the robustness of the estimated MSE-OLCs. Supported by these algorithms, the estimated MSE-OLCs yield dramatic improvements in model accuracy over a variety of situations, which include real-world as well as simulated (Monte Carlo) data.

The framework has benefited from a series of development and investigation steps that can be summerized as follows:

- In Chapter 2, nine closed-form expressions for the MSE-optimal combination-weights are derived for four types of MSE-OLCs. These MSE-OLCs include the unconstrained MSE-OLC with a constant term, which yields the theoretical minimal MSE among the four types. The other three types constrain the sum of the combination-weights to unity and (or) require no constant term.

- In Chapter 3, ordinary least squares estimators as well as alternate estimators are considered for estimating the optimal combination-weights from observed data.

- In Chapter 4, a pilot study utilizes unconstrained MSE-OLCs in constructing a neural network based scheduler that aids in generating daily allocation schedules for a manufacturing company. The study illustrates that MSE-OLCs can improve model accuracy without requiring excessive training or extensive exploration of the topology space of the considered networks through replications.

- In Chapter 5, Example 1 illustrates that the improvement in model accuracy as a result of employing MSE-OLCs extends beyond approximating the function values to approximating higher-order derivatives.

- In Chapters 6, the harmful effects of collinearity on the generalization ability (robustness) of the estimated MSE-OLCs are investigated.

  Since the component networks approximate the same physical quantity (or quantities), their corresponding outputs are highly (positively) correlated, which

often result in severe collinearity among them. Moreover, collinearity may also exist among the errors of the component networks. Such collinearity sometimes undermine the robustness of the estimated MSE-OLCs.

To detect the presence of collinearity, some common methods are evaluated and collinearity diagnostics developed by Belsley et al. [9], referred to as BKW diagnostics, are adopted for detecting the presence of collinearity and determining the involved networks.

Collinearity does not necessarily harm the robustness of estimated MSE-OLCs. Thus, we use a method based on cross-validation to identify harmful collinearity.

- In Chapter 7, we propose an approach for improving the robustness of estimated MSE-OLCs by the selection of the component networks. This selection approach utilizes the BKW collinearity diagnostics and the cross-validation method for testing the robustness of the resultant estimated MSE-OLC. Based on the selection approach, six algorithms are developed.

- In Chapter 8, an empirical study investigates the merits of MSE-OLCs and the effectiveness of the six selection algorithms. We use an experimental-design approach to examine the effects of the quantity and the quality of the observed data on the results of the study. The study confirms the potential benefits of the MSE-OLCs in improving model accuracy both for the case of well-trained as well as the case of poorly trained component networks. Furthermore, the study demonstrates the effectiveness of the six selection algorithms in improving the robustness of the estimated MSE-OLCs.

9.2 Conclusions

The main conclusions of this dissertation are:

- MSE-OLCs can significantly improve model accuracy, and may substitute for excessive training and (or) extensive exploration of the topology space of the considered networks through replications, which are often required to achieve a target model accuracy.

- MSE-OLCs are straightforward and require modest computational effort for estimating the optimal combination-weights.

- The improvement in model accuracy as a result of employing MSE-OLCs extends beyond approximating the function values to approximating higher-order derivatives. Moreover, since the best network to approximate a certain high-order derivative may be different from the best network to approximate the function, combining a number of trained network may help integrate the knowledge acquired by the component networks, rather than picking just one single network as best and discarding the rest.

- The effectiveness of MSE-OLC is not dependent on the accuracy of the component networks. Significant improvement in model accuracy may be achieved for both well-trained component networks as well as for poorly trained component networks. The latter result suggests that poorly trained networks can be used by MSE-OLCs as bases for regression.

- For well-trained component networks, the unconstrained optimal combination-weights tend to automatically sum to one, while the constant term tends to zero. This result is intuitive since well-trained component networks tend to be close to the approximated quantity, $r(\vec{X})$, and also tend to be unbiased.

- Collinearity among the component networks sometimes harm the robustness of the estimated MSE-OLC. Thus, proper collinearity diagnosis as well as corrective measures may be needed.

- Proper selection of the component networks, by utilizing collinearity diagnostics and cross-validation methods, can significantly improve the robustness of the resulting MSE-OLC.

9.3 Recommendations

In a practical situation, where it is desired to construct a neural network based model for a data-generating process from a given set of observed data, we recommend the following procedure for combining a number of trained networks[1]:

- Construct unconstrained MSE-OLC with a constant term using the training data. Also, construct the simple average of the trained networks and identify the best network (using the cross-validation data set).

- Compare the performance of the MSE-OLC on the cross-validation data with those of the best network and the simple averaging.

- If the MSE-OLC significantly outperforms the best network and the simple averaging, then there may be no need for trying the selection algorithms. Otherwise, try using some or all the six algorithms (modified versions) and pick the best performer on the cross-validation set.

[1]We assume that during training, the observed data set is split into: a training data set, a cross-validation data set (typically used in premature termination of training to avoid overfitting to the training data), and a testing data set.

9.4 Future Directions

Future directions to this dissertation include:

- Combining NNs generated by different learning techniques. A wide variety of learning algorithms and network architectures are currently available [28, 29, 45, 46, 86, 93]. Combining NNs generated using different learning techniques is of special value, since the collinearity among the errors of such networks may be small. This point leads to a very important research question: how to make the component NNs "different" or "less collinear" in order to increase the benefit of combining?

- Creating hybrid systems that include NNs and other types of models, such as polynomial regression models which are fairly popular in process modeling applications. No special extensions are required for this step since the framework developed here is readily applicable to such situations.

- Investigate the use of convex MSE-OLCs discussed in Section 2.5. Although the unconstrained MSE-OLC with constant term yields the theoretical minimal MSE, convex MSE-OLCs may be of value in some applications where negative combination-weights are undesirable [4, 13, 19].

- Study the effect of the presence of influential observations [8, pages 245–270] in the data on the robustness of the MSE-OLCs. Also, investigate methods for reducing collinearity through controlled data sampling as well as controlled splitting of the combination data into estimation and cross-validation data sets.

- Apply MSE-OLCs to combine recurrent NNs[2] [52, pages 172–176].

- Consider other methods for combining such as robust methods [42, 43].

[2] This extension has been suggested through personal communication with B. Kehoe, Department of Physics, California State University - Fresno.

LIST OF REFERENCES

[1] E. Alpaydin. GAL: Networks that grow when they learn and shrink when they forget. Technical Report 91–032, International Computer Science Institute, Berkeley, CA, May 1991.

[2] E. Alpaydin. Multiple networks for function learning. In *Proceedings of the 1993 IEEE International Conference on Neural Networks*, pages I:9–14. IEEE, Apr. 1993.

[3] C. W. Anderson, J. A. Franklin, and R. S. Sutton. Learning a nonlinear model of a manufacturing process using multilayer connectionist networks. In *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, pages 404–409. IEEE, 1990.

[4] J. M. Bates and C. W. J. Granger. The combination of forecasts. *Operational Research Quarterly*, 20(4):451–468, 1969.

[5] W. G. Baxt. Improving the accuracy of an artificial neural network using multiple differently trained networks. *Neural Computation*, 4:772–780, 1992.

[6] D. A. Belsley. Assessing the presence of harmful collinearity and other forms of weak data through a test for signal-to-noise. *Journal of Econometrics*, 20:211–253, 1982.

[7] D. A. Belsley. Collinearity and forecasting. *Journal of Forecasting*, 3:183–196, 1984.

[8] D. A. Belsley. *Conditioning Diagnostics: Collinearity and Weak Data in Regression.* John Wiley & Sons, New York, 1991.

[9] D. A. Belsley, E. Kuth, and R. E. Welsch. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity.* John Wiley & Sons, New York, 1980.

[10] J. A. Benediktsson, J. R. Sveinsson, O. K. Ersoy, and P. H. Swain. Parallel consensual neural networks. In *Proceedings of the 1993 IEEE International Conference on Neural Networks*, pages I:27–32. IEEE, Apr. 1993.

[11] N. V. Bhat, P. A. Minderman, Jr., T. McAvoy, and N. S. Wang. Modeling chemical process systems via neural computation. *IEEE Control Systems Magazine*, 80:24–30, Apr. 1990.

[12] E. K. Blum. *Numerical Analysis and Computation: Theory and Practice.* Addison-Wesley, Massachusetts, 1972.

[13] D. W. Bunn. Statistical efficiency in the linear combination of forecasts. *International Journal of Forecasting*, 1:151–163, 1985.

[14] D. W. Bunn. Forecasting with more than one model. *Journal of Forecasting*, 8:161–166, 1989.

[15] M. Caudill. Avoiding the great backpropagation trap. *AI Expert*, 6(7):29–35, July 1991.

[16] P. G. Ciarlet. *Introduction to Numerical Linear Algebra and Optimisation.* Cambridge University Press, New York, 1989.

[17] R. T. Clemen. Linear constraints and the efficiency of combined forecasts. *Journal of Forecasting*, 5:31–38, 1986.

[18] R. T. Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5:559–583, 1989.

[19] R. T. Clemen and R. L. Winkler. Combining economic forecasts. *Journal of Business & Economic Statistics*, 4(1):39–46, Jan. 1986.

[20] D. Cook and R. Shannon. A sensitivity analysis of a backpropagation neural network for manufacturing process parameters. *Journal of Intelligent Manufacturing*, 2:155–163, 1991.

[21] L. Cooper. Hybrid neural network architectures: Equilibrium systems that pay attention. In R. J. Mammone and Y. Y. Zeevi, editors, *Neural Networks Theory and Applications*, pages 81–96. Academic Press, 1991.

[22] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[23] J. De Villiers and E. Barnard. Backpropagation neural nets with one and two hidden layers. *IEEE Transactions on Neural Networks*, 4(1):136–141, 1993.

[24] F. X. Diebold. Forecast combination and encompassing: Reconciling two divergent literatures. *International Journal of Forecasting*, 5:589–592, 1989.

[25] F. X. Diebold and P. Pauly. Structural change and the combination of forecasts. *Journal of Forecasting*, 6:21–40, 1987.

[26] G. P. Drago and S. Ridella. Statistically controlled activation weight initialization SCAWI. *IEEE Transactions on Neural Networks*, 3(4):627–631, 1992.

[27] H. Drucker and Y. Le Cun. Double backpropagation increasing generalization performance. In *Proceedings of the 1991 International Joint Conference on Neural Networks in Seattle*, pages II:145–150. IEEE, 1991.

[28] S. E. Fahlman. Faster-learning variations on back-propagation: An empirical study. In *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kaufman, 1988.

[29] S. E. Fahlman and C. Lebiere. The cascade-correlation architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 524–532. Morgan Kaufman, 1990.

[30] L. A. Feldkamp, G. V. Puskorius, L. I. Davis Jr., and F. Yuan. Strategies and issues in applications of neural networks. In *Proceedings of the 1992 International Joint Conference on Neural Networks in Baltimore*, pages IV:304–309. IEEE, 1992.

[31] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[32] E. S. Gardner Jr. and S. Makridakis. The future of forecasting. *International Journal of Forecasting*, 4:325–330, 1988.

[33] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[34] C. Genest and J. V. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–148, 1986.

[35] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981.

[36] P. W. Glynn and D. L. Iglehart. The optimal linear combination of control variates in the presence of asymptotically negligible bias. *Naval Research Logistics*, 36:683–692, 1989.

[37] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[38] C. W. J. Granger. Combining forecasts — twenty years later. *Journal of Forecasting*, 8:167–173, 1989.

[39] C. W. J. Granger and R. Ramanathan. Improved methods of combining forecasts. *Journal of Forecasting*, 3:197–204, 1984.

[40] J. B. Guerard Jr. and R. T. Clemen. Collinearity and the use of latent root regression for combining GNP forecasts. *Journal of Forecasting*, 8:231–238, 1989.

[41] E. B. Hall, A. E. Wessel, and G. L. Wise. Some aspects of fusion in estimation theory. *IEEE Transactions on Information Theory*, 37(2):420–422, Mar. 1991.

[42] J. Hallman and M. Kamstra. Combining algorithms based on robust estimation techniques and co-integrating restrictions. *Journal of Forecasting*, 8:189–198, 1989.

[43] F. P. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley & Sons, New York, 1986.

[44] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[45] S. A. Harp, T. Samad, and A. Guha. Designing application-specific neural networks using the genetic algorithm. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 447–454. Morgan Kaufman, 1990.

[46] E. J. Hartman and J. D. Keeler. Layered neural networks with Gaussian hidden units as universal approximators. *Neural Computation*, 2:210–215, 1990.

[47] S. Hashem. Sensitivity analysis for feedforward artificial neural networks with differentiable activation functions. In *Proceedings of the 1992 International Joint Conference on Neural Networks in Baltimore*, pages I:419–424. IEEE, 1992.

[48] S. Hashem and B. Schmeiser. Improving model accuracy using optimal linear combinations of trained neural networks. Technical Report SMS92–16, School of Industrial Engineering, Purdue University, 1992.

[49] S. Hashem and B. Schmeiser. Approximating a function and its derivatives using MSE-optimal linear combinations of trained feedforward neural networks. In *Proceedings of the 1993 World Congress on Neural Networks*, pages I:617–620, New Jersey, 1993. Lawrence Erlbaum Associates.

[50] S. Hashem, Y. Yih, and B. Schmeiser. An efficient model for product allocation using optimal combinations of neural networks. In C. Dagli, L. I. Burke, B. R. Fernández, and J. Ghosh, editors, *Intelligent Engineering Systems through Artificial Neural Networks*, volume 3, pages 669–674. ASME Press, 1993.

[51] R. Hecht-Nielson. Theory of the backpropagation neural network. In *Proceedings of the 1989 International Joint Conference on Neural Networks*, pages I:593–605. IEEE, 1989.

[52] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[53] W. W. Hines and D. C. Montgomery. *Probability and Statistics in Engineering and Management Science.* John Wiley & Sons, 1990.

[54] D. Hodouin, J. Thibault, and F. Flament. Artificial neural networks: An emerging technique to model and control mineral processing plants. In *120th Annual Meeting.* Society for Mining, Metallurgy and Exploration Inc., Feb. 1991.

[55] K. Holden and D. A. Peel. Unbiasedness, efficiency and the combination of economic forecasts. *Journal of Forecasting*, 8:175–188, 1989.

[56] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–368, 1989.

[57] K. Hornik, M. Stinchcombe, and H. White. Approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560, 1990.

[58] J. T. Hsiung, W. Suewatanakul, and D. M. Himmelblau. Should back propagation be replaced by more effective optimization algorithms. In *Proceedings of the 1991 International Joint Conference on Neural Networks in Seattle*, pages I:353–356. IEEE, 1991.

[59] B. L. Kalman and S. C. Kwasny. Why tanh: Choosing a sigmoidal function. In *Proceedings of the 1992 International Joint Conference on Neural Networks in Baltimore*, pages IV:578–581. IEEE, 1992.

[60] C. Klimasauskas. Neural nets tell why. *Dr. Dobb's Journal*, pages 16–24, Apr. 1991.

[61] B. Kosko. *Neural Networks for Signal Processing.* Prentice Hall, 1992.

[62] J.-M. Lambert and R. Hecht-Nielsen. Application of feedforward and recurrent neural networks to chemical plant predictive modeling. In *Proceedings of the 1991 International Joint Conference on Neural Networks in Seattle*, pages I:373–378. IEEE, 1991.

[63] P. d. Laplace. *Deuxième Supplément a la Théorie Analytique des Probabilités.* Courcier, Paris, 1818. Reprinted (1847) in *Oeuvers Complètes de Laplace,* Vol. 7 (Paris, Gauthier-Villars), 531–580.

[64] H. Lari-Najafi, M. Nasiruddin, and T. Samad. Effect of initial weights on back-propagation and its variations. In *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, pages 218–219. IEEE, 1989.

[65] Y. Lee, S.-H. Oh, and M. W. Kim. The effect of initial weights on premature saturation in back-propagation learning. In *Proceedings of the 1991 International Joint Conference on Neural Networks in Seattle*, pages I:765–770. IEEE, 1991.

[66] E. Levin, N. Tishby, and S. A. Solla. A statistical approach to learning and generalization in layered neural networks. *Proceedings of the IEEE*, 78(10):1568–1574, Oct. 1990.

[67] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. *The Forecasting Accuracy of Major Time Series Methods*. John Wiley & Sons, New York, 1984.

[68] S. Makridakis, C. Chatfield, M. Hibon, M. Lawrence, T. Mills, K. Ord, and L. F. Simmons. The M2-Competition: A real-time judgmentally based forecasting study. *International Journal of Forecasting*, 9:5–22, 1993.

[69] S. Makridakis and R. L. Winkler. Averages of forecasts: Some empirical results. *Management Science*, 29(9):987–996, 1983.

[70] G. Mani. Lowering variance of decisions by using artificial neural networks portfolios. *Neural Computation*, 3:484–486, 1991.

[71] K. A. Marko, J. James, J. Dosdall, and J. Murphy. Automotive control system diagnostics using neural nets for rapid pattern classification of large data sets. In *Proceedings of the 1989 International Joint Conference on Neural Networks*, pages II:13–16. IEEE, 1989.

[72] H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[73] G. L. Martin and J. A. Pittman. Recognizing hand-printed letters and digits using backpropagation learning. *Neural Computation*, 3:258–267, 1991.

[74] L. Menezes and D. Bunn. Specification of predictive distribution from a combination of forecasts. *Methods of Operations Research*, 64:397–405, 1991.

[75] W. T. Miller, III, R. S. Sutton, and P. J. Werbos. *Neural Networks for Control*. MIT Press, 1990.

[76] N. Morgan and H. Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 630–637. Morgan Kaufman, 1990.

[77] H. Moskowitz and G. P. Wright. *Statistics for Management and Economics*. Charles Merrill Publishing Company, Ohio, 1985.

[78] F. Nadi, A. M. Agogino, and D. A. Hodges. Use of influence diagrams and neural networks in modeling LPCVD. In *Proceedings of the IEEE/SEMI International Semiconductor Manufacturing Science Symposium*, pages 111–112, May 1990.

[79] A. Namatame and Y. Kimata. Improving the generalising capabilities of a back-propagation network. *The International Journal of Neural Networks Research & Applications*, 1(2):86–94, 1989.

[80] K. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[81] J. Neter, W. Wasserman, and M. H. Kutner. *Applied Linear Statistical Models*. Irwin, Homewood, IL, 1985.

[82] P. Newbold and C. W. J. Granger. Experience with forecasting univariate time series and the combination of forecasts. *Journal of Royal Statistical Society A*, 137:131–165, 1974.

[83] D. H. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, pages 18–23, Apr. 1990.

[84] J. B. Orris and H. R. Feeser. Using neural networks for exploratory data analysis. Presentation at the 1990 ORSA/TIMS joint national meeting in Philadelphia, College of Business Administration, Butler University, 4600 Sunset Avenue, Indianapolis, IN, Oct. 1990.

[85] F. C. Palm and A. Zellner. To combine or not to combine? Issues of combining forecasts. *Journal of Forecasting*, 11:687–701, 1992.

[86] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3:246–257, 1991.

[87] D. Parker. Learning logic. Invention report S81–64, File 1, Office of Technology Licensing, Stanford University, CA, 1982.

[88] M. P. Perrone. *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization*. PhD thesis, Department of Physics, Brown University, May 1993.

[89] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*. Chapman & Hall, 1993. Forthcoming.

[90] D. J. Ried. Combining three estimates of gross domestic product. *Economica*, 35:431–444, 1968.

[91] D. J. Ried. *A Comparative Study of Time Series Prediction Techniques on Economic Data.* PhD thesis, University of Nottingham, Nottingham, UK, 1969.

[92] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, chapter 8, pages 318–328. MIT Press, Cambridge, MA, 1986.

[93] J. D. Schaffer, D. Whitley, and L. J. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In L. D. Whitley and J. D. Schaffer, editors, *Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 1–37. IEEE, 1992.

[94] R. L. Scheaffer and J. T. McClave. *Probability and Statistics for Engineers.* PWS-KENT Publishing Company, Boston, 1990.

[95] D. C. Schmittlein, J. Kim, and D. G. Morrison. Combining forecasts: Operational adjustments to theoretically optimal rules. *Management Science*, 36(9):1044–1056, Sept. 1990.

[96] P. K. Simpson. *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations.* Pergamon Press, New York, 1990.

[97] A. Smith and C. H. Dagli. Backpropagation neural network approaches to process control: Evaluation and comparison. Working Paper Series 90–22–47, Department of Engineering Management, University of Missouri-Rolla, Nov. 1990.

[98] G. Smith and F. Campbell. A critique of some ridge regression methods. *Journal of the American Statistical Association*, 75(369):74–103, Mar. 1980.

[99] W. T. Song and B. Schmeiser. Minimal-MSE linear combinations of variance estimators of the sample mean. In M. Abrams, P. Haigh, and J. Comfort, editors, *Proceedings of the 1988 Winter Simulation Conference*, pages 414–421, 1988.

[100] G. W. Stewart. Collinearity and least squares regression. *Statistical Science*, 2(1):68–100, 1987.

[101] M. F. Tenorio. Topology synthesis networks: Self organization of structure and weight adjustment as a learning paradigm. *Parallel Computing*, 14:363–380, 1990.

[102] G. Trenkler and E. P. Liski. Linear constraints and the efficiency of combined forecasts. *Journal of Forecasting*, 5:197–202, 1986.

[103] R. E. Walpole and R. H. Myers. *Probability and Statistics for Engineers and Scientists.* Macmillan Publishing Company, New York, 1989.

[104] J. Wang and Y. Yih. Using neural networks to select control strategy for automated storage and retrieval systems (AS/RS). Research Memorandum 92–9, School of Industrial Engineering, Purdue University, May 1992.

[105] J. T. Webster, R. F. Gunst, and R. L. Mason. Latent root regression analysis. *Technometrics*, 16(4):513–522, Nov. 1974.

[106] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight-elimination with application to forecasting. In R. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 875–882. Morgan Kaufman, 1991.

[107] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[108] P. Werbos. Backpropagation and neurocontrol: A review and prospectus. In *Proceedings of the 1989 International Joint Conference on Neural Networks*, pages I:209–216. IEEE, 1989.

[109] H. White. Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. *Neural Networks*, 3:535–549, 1990.

[110] D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing*, 14:347–361, 1990.

[111] R. L. Winkler and R. T. Clemen. Sensitivity of weights in combining forecasts. *Operations Research*, 40(3):609–614, May-June 1992.

APPENDIX
MSE-OPTIMAL WEIGHTS FOR LINEAR COMBINATIONS

A.1 Unconstrained MSE-OLC with a Constant Term

Consider the problem

$$\text{P1}: \quad \min_{\vec{\alpha}} \text{MSE}(\tilde{y}(\vec{X};\vec{\alpha})).$$

A.1.1 Solution:

$$\begin{aligned}
\text{MSE}(\tilde{y}(\vec{X};\vec{\alpha})) &= \text{E}(\tilde{\delta}(\vec{X};\vec{\alpha}))^2 = \text{E}\langle r(\vec{X}) - \vec{\alpha}^t\,\vec{y}(\vec{X})\rangle^2 \\
&= \text{E}\langle r^2(\vec{X}) - 2\,r(\vec{X})\,\vec{\alpha}^t\,\vec{y}(\vec{X}) + (\vec{\alpha}^t\,\vec{y}(\vec{X}))^2\rangle \\
&= \text{E}(r^2(\vec{X})) - 2\,\vec{\alpha}^t\,\vec{U} + \vec{\alpha}^t\,\Psi\,\vec{\alpha}\,,
\end{aligned}$$

where $\vec{U} = [u_i] = [\text{E}\langle r(\vec{X})y_i(\vec{X})\rangle]$ is a $(p+1)\times 1$ vector and $\Psi = [\psi_{ij}] = [\text{E}\langle y_i(\vec{X})y_j(\vec{X})\rangle]$ is a $(p+1)\times(p+1)$ matrix.

Taking the derivative of the MSE with respect to $\vec{\alpha}$, then equating it with zero, the resultant optimal-weights vector is

$$\vec{\alpha}_{(1)} = \Psi^{-1}\vec{U},$$

and the corresponding (minimum) MSE is

$$\begin{aligned}
\text{MSE}^{(1)} &= \text{E}\langle r(\vec{X}) - \vec{U}^t\,\Psi^{-1}\,\vec{y}(\vec{X})\rangle^2 \\
&= \text{E}(r^2(\vec{X})) - \vec{U}^t\,\Psi^{-1}\,\vec{U}\,.
\end{aligned}$$

A.2 Constrained MSE-OLC with a Constant Term

Consider the problem

$$\text{P2}: \quad \min_{\vec{\alpha}} \text{MSE}(\tilde{y}(\vec{X};\vec{\alpha})), \qquad \text{s.t. } \vec{\alpha}^t\,\vec{1}_z = 1\,,$$

where $\vec{1}_z$ is a vector of proper dimension with the first component equal to zero while the remaining components equal to one.

A.2.1 Solution:

Forming the Lagrangian function, $\mathcal{L}$, for $P2$

$$\begin{aligned}\mathcal{L} &= \mathrm{E}\langle r(\vec{X}) - \vec{\alpha}^t\,\vec{y}(\vec{X})\rangle^2 + 2\,\beta(\vec{\alpha}^t\,\vec{1}_z - 1) \\ &= \mathrm{E}(r^2(\vec{X})) - 2\,\vec{\alpha}^t\,\vec{U} + \vec{\alpha}^t\,\Psi\,\vec{\alpha} + 2\,\beta(\vec{\alpha}^t\,\vec{1}_z - 1)\,.\end{aligned}$$

Differentiating $\mathcal{L}$ w.r.t. $\vec{\alpha}$,

$$\frac{\partial\mathcal{L}}{\partial\vec{\alpha}} = -2\,\vec{U} + 2\,\Psi\,\vec{\alpha} + 2\,\beta\,\vec{1}_z = 0\,.$$

Thus,

$$\vec{\alpha}_{(2)} = \Psi^{-1}(\vec{U} - \beta\,\vec{1}_z)\,.$$

Premultiplying by $\vec{1}_z^t$, and using the condition, $\vec{\alpha}^t\,\vec{1}_z = \vec{1}_z^t\,\vec{\alpha} = 1$, yields

$$\beta_{(2)} = \frac{-1 + \vec{1}_z^t\,\Psi^{-1}\,\vec{U}}{\vec{1}_z^t\,\Psi^{-1}\,\vec{1}_z}\,.$$

Thus,

$$\vec{\alpha}_{(2)} = \vec{\alpha}_{(1)} - \beta_{(2)}\,\Psi^{-1}\,\vec{1}_z\,,$$

and

$$\begin{aligned}\mathrm{MSE}^{(2)} &= \mathrm{E}\langle r(\vec{X}) - \vec{\alpha}_{(2)}^t\,\vec{y}(\vec{X})\rangle^2 \\ &= \mathrm{E}\langle r(\vec{X}) - (\vec{\alpha}_{(1)}^t - \beta_{(2)}\,\vec{1}_z^t\,\Psi^{-1})\,\vec{y}(\vec{X})\rangle^2 \\ &= \mathrm{E}\langle (r(\vec{X}) - \vec{\alpha}_{(1)}^t\,\vec{y}(\vec{X})) + (\beta_{(2)}\,\vec{1}_z^t\,\Psi^{-1}\,\vec{y}(\vec{X}))\rangle^2 \\ &= \mathrm{MSE}^{(1)} + \beta_{(2)}^2\,\vec{1}_z^t\,\Psi^{-1}\,\vec{1}_z\,.\end{aligned}$$

A.2.2 Alternate Solution:

P2 is equivalent to

$$\mathrm{P2}':\quad \min_{\vec{\alpha}}\ \mathrm{E}(\tilde{\delta}(\vec{X};\vec{\alpha}))^2, \qquad \text{s.t. } \vec{\alpha}^t\,\vec{1}_z = 1\,.$$

Since

$$\begin{aligned}\tilde{\delta}(\vec{X};\vec{\alpha}) &= r(\vec{X}) - \tilde{y}(\vec{X};\vec{\alpha}) \\ &= r(\vec{X})(\vec{\alpha}^t\,\vec{1}_z) - \vec{\alpha}^t\,\vec{y}(\vec{X}) \\ &= \vec{\alpha}^t\,(r(\vec{X})\,\vec{1}_z - \vec{y}(\vec{X})) \\ &= \vec{\alpha}^t\,\vec{\delta}(\vec{X})\,,\end{aligned}$$

where $\vec{\delta}(\vec{X})$ is a $(p+1)\times 1$ vector, with $\delta_0(\vec{X}) = -y_0(\vec{X}) = -1$, and $\delta_i(\vec{X}) = r(\vec{X}) - y_i(\vec{X});\ (i > 0)$. Therefore, $\mathrm{E}(\tilde{\delta}(\vec{X};\vec{\alpha}))^2 = \mathrm{E}\langle\vec{\alpha}^t\,\vec{\delta}(\vec{X})\rangle^2 = \vec{\alpha}^t\,\Omega\,\vec{\alpha}$, where $\Omega = [\omega_{ij}] = [\mathrm{E}\langle\delta_i(\vec{X})\,\delta_j(\vec{X})\rangle]$ is a $(p+1)\times(p+1)$ matrix.

Forming the Lagrangian function, $\mathcal{L}$, for P2$'$

$$\mathcal{L} = \vec{\alpha}^t \, \Omega \, \vec{\alpha} + 2 \, \lambda \, (\vec{\alpha}^t \, \vec{1}_z - 1) \, .$$

Differentiating $\mathcal{L}$ w.r.t. $\vec{\alpha}$,

$$\frac{\partial \mathcal{L}}{\partial \vec{\alpha}} = 2 \, \Omega \, \vec{\alpha} + 2 \, \lambda \vec{1}_z = 0 \, ,$$

yields $\vec{\alpha}_{(2)} = -\lambda \Omega^{-1} \, \vec{1}_z$. Premultiplying by $\vec{1}_z^t$, and using the condition, $\vec{\alpha}^t \, \vec{1}_z = \vec{1}_z^t \, \vec{\alpha} = 1$, yields

$$\lambda_{(2)} = \frac{-1}{\vec{1}_z^t \, \Omega^{-1} \, \vec{1}_z} \, .$$

Thus,

$$\vec{\alpha}_{(2)} = \frac{\Omega^{-1} \, \vec{1}_z}{\vec{1}_z^t \, \Omega^{-1} \, \vec{1}_z} \, ,$$

and

$$\mathrm{MSE}^{(2)} = \frac{1}{\vec{1}_z^t \, \Omega^{-1} \, \vec{1}_z} \, .$$

A.3 Unconstrained MSE-OLC without a Constant Term

Consider the problem

$$\mathrm{P3}: \quad \min_{\vec{\alpha}} \; \mathrm{MSE}(\tilde{y}(\vec{X}; \vec{\alpha})), \qquad \text{s.t.} \;\; \vec{\alpha}^t \, \vec{\vartheta}_z = 0 \, ,$$

where $\vec{\vartheta}_z$ is a vector of proper dimension with the first component equal to one, while the remaining components equal to zero.

A.3.1 Solution:

Forming the Lagrangian function, $\mathcal{L}$, for $P3$

$$\begin{aligned} \mathcal{L} &= \mathrm{E}\langle r(\vec{X}) - \vec{\alpha}^t \, \vec{y}(\vec{X}) \rangle^2 + 2 \, \beta (\vec{\alpha}^t \, \vec{\vartheta}_z) \\ &= \mathrm{E}(r^2(\vec{X})) - 2 \, \vec{\alpha}^t \, \vec{U} + \vec{\alpha}^t \, \Psi \, \vec{\alpha} + 2 \, \beta (\vec{\alpha}^t \, \vec{\vartheta}_z) \, . \end{aligned}$$

Differentiating $\mathcal{L}$ w.r.t. $\vec{\alpha}$,

$$\frac{\partial \mathcal{L}}{\partial \vec{\alpha}} = -2 \, \vec{U} + 2 \, \Psi \, \vec{\alpha} + 2 \, \beta \, \vec{\vartheta}_z = 0 \, .$$

Thus,

$$\vec{\alpha}_{(3)} = \Psi^{-1} (\vec{U} - \beta \, \vec{\vartheta}_z) \, .$$

Premultiplying by $\vec{\vartheta}_z^t$, and using the condition, $\vec{\alpha}^t \, \vec{\vartheta}_z = \vec{\vartheta}_z^t \, \vec{\alpha} = 0$, yields

$$\beta_{(3)} = \frac{\vec{\vartheta}_z^t \, \Psi^{-1} \, \vec{U}}{\vec{\vartheta}_z^t \, \Psi^{-1} \, \vec{\vartheta}_z} \, .$$

Thus,

$$\vec{\alpha}_{(3)} = \vec{\alpha}_{(1)} - \beta_{(3)}\, \Psi^{-1}\, \vec{\vartheta}_z\,,$$

and

$$\begin{aligned}
\mathrm{MSE}^{(3)} &= \mathrm{E}\langle r(\vec{X}) - \vec{\alpha}^t_{(3)}\, \vec{y}(\vec{X})\rangle^2 \\
&= \mathrm{E}\langle r(\vec{X}) - (\vec{\alpha}^t_{(1)} - \beta_{(3)}\, \vec{\vartheta}^t_z\, \Psi^{-1})\, \vec{y}(\vec{X})\rangle^2 \\
&= \mathrm{E}\langle (r(\vec{X}) - \vec{\alpha}^t_{(1)}\, \vec{y}(\vec{X})) + (\beta_{(3)}\, \vec{\vartheta}^t_z\, \Psi^{-1}\, \vec{y}(\vec{X}))\rangle^2 \\
&= \mathrm{MSE}^{(1)} + \beta^2_{(3)}\, \vec{\vartheta}^t_z\, \Psi^{-1}\, \vec{\vartheta}_z\,.
\end{aligned}$$

A.4 Constrained MSE-OLC without a Constant Term

Consider the problem

$$\mathrm{P4}: \quad \min_{\vec{\alpha}}\ \mathrm{MSE}(\tilde{y}(\vec{X};\vec{\alpha})), \qquad \text{s.t. } \vec{\alpha}^t\, \vec{\vartheta}_z = 0 \text{and } \vec{\alpha}^t\, \vec{1}_z = 1\,.$$

A.4.1 Solution:

Forming the Lagrangian function, $\mathcal{L}$, for $P4$

$$\begin{aligned}
\mathcal{L} &= \mathrm{E}\langle r(\vec{X}) - \vec{\alpha}^t\, \vec{y}(\vec{X})\rangle^2 + 2\,\beta_a(\vec{\alpha}^t\, \vec{\vartheta}_z) + 2\,\beta_b(\vec{\alpha}^t\, \vec{1}_z - 1) \\
&= \mathrm{E}(r^2(\vec{X})) - 2\,\vec{\alpha}^t\, \vec{U} + \vec{\alpha}^t\, \Psi\, \vec{\alpha} + 2\,\beta_a(\vec{\alpha}^t\, \vec{\vartheta}_z) + 2\,\beta_b(\vec{\alpha}^t\, \vec{1}_z - 1)\,.
\end{aligned}$$

Differentiating $\mathcal{L}$ w.r.t. $\vec{\alpha}$,

$$\frac{\partial \mathcal{L}}{\partial \vec{\alpha}} = -2\,\vec{U} + 2\,\Psi\, \vec{\alpha} + 2\,\beta_a\, \vec{\vartheta}_z + 2\,\beta_b\, \vec{1}_z = 0\,.$$

Thus,

$$\begin{aligned}
\vec{\alpha}_{(4)} &= \Psi^{-1}(\vec{U} - \beta_{(4a)}\, \vec{\vartheta}_z - \beta_{(4b)}\, \vec{1}_z) \\
&= \vec{\alpha}_{(1)} - \beta_{(4a)}\, \Psi^{-1}\, \vec{\vartheta}_z - \beta_{(4b)}\, \Psi^{-1}\, \vec{1}_z\,.
\end{aligned}$$

Premultiplying $\vec{\alpha}_{(4)}$ by $\vec{\vartheta}^t_z$, and using the condition, $\vec{\alpha}^t\, \vec{\vartheta}_z = \vec{\vartheta}^t_z\, \vec{\alpha} = 0$, yields

$$\beta_{(4a)} = \frac{\vec{\vartheta}^t_z\, \Psi^{-1}\, \vec{U} - \beta_{(4b)}\, \vec{\vartheta}^t_z\, \Psi^{-1}\, \vec{1}_z}{\vec{\vartheta}^t_z\, \Psi^{-1}\, \vec{\vartheta}_z}\,.$$

Premultiplying $\vec{\alpha}_{(4)}$ by $\vec{1}^t_z$, and using the condition, $\vec{\alpha}^t\, \vec{1}_z = \vec{1}^t_z\, \vec{\alpha} = 1$, yields

$$\beta_{(4b)} = \frac{(\vec{1}^t_z\, \Psi^{-1}\, \vec{U} - 1)(\vec{\vartheta}^t_z\, \Psi^{-1}\, \vec{\vartheta}_z) - (\vec{\vartheta}^t_z\, \Psi^{-1}\, \vec{U})(\vec{1}^t_z\, \Psi^{-1}\, \vec{\vartheta}_z)}{(\vec{1}^t_z\, \Psi^{-1}\, \vec{1}_z)(\vec{\vartheta}^t_z\, \Psi^{-1}\, \vec{\vartheta}_z) - (\vec{\vartheta}^t_z\, \Psi^{-1}\, \vec{1}_z)^2}\,.$$

Hence,

$$\begin{aligned}
\mathrm{MSE}^{(4)} &= \mathrm{E}\langle r(\vec{X}) - \vec{\alpha}_{(4)}^{t}\,\vec{y}(\vec{X})\rangle^2 \\
&= \mathrm{E}\langle r(\vec{X}) - (\vec{\alpha}_{(1)} - \beta_{(4a)}\,\Psi^{-1}\,\vec{\vartheta}_z - \beta_{(4b)}\,\Psi^{-1}\,\vec{1}_z)^t\,\vec{y}(\vec{X})\rangle^2 \\
&= \mathrm{E}\langle (r(\vec{X}) - \vec{\alpha}_{(1)}^{t}\,\vec{y}(\vec{X})) + (\beta_{(4a)}\,\vec{\vartheta}_z^{t}\,\Psi^{-1}\,\vec{y}(\vec{X}) + \beta_{(4b)}\,\vec{1}_z^{t}\,\Psi^{-1}\,\vec{y}(\vec{X}))\rangle^2 \\
&= \mathrm{MSE}^{(1)} + \beta_{(4a)}^2\,(\vec{\vartheta}_z^{t}\,\Psi^{-1}\,\vec{\vartheta}_z) + \beta_{(4b)}^2\,(\vec{1}_z^{t}\,\Psi^{-1}\,\vec{1}_z) + 2\,\beta_{(4a)}\,\beta_{(4b)}(\vec{\vartheta}_z^{t}\,\Psi^{-1}\,\vec{1}_z)\,.
\end{aligned}$$

Note that using the above expressions, it can be easily shown that $\beta_{(4a)}$ and $\beta_{(4b)}$ can expressed as a function of $\beta_{(2)}$ and $\beta_{(3)}$ as follows:

$$\beta_{(4a)} = \mathcal{M}\left\langle \beta_{(3)} - \beta_{(2)}\,\frac{(\vec{1}_z^{t}\,\Psi^{-1}\,\vec{\vartheta}_z)}{(\vec{\vartheta}_z^{t}\,\Psi^{-1}\,\vec{\vartheta}_z)} \right\rangle,$$

and

$$\beta_{(4b)} = \mathcal{M}\left\langle \beta_{(2)} - \beta_{(3)}\,\frac{(\vec{1}_z^{t}\,\Psi^{-1}\,\vec{\vartheta}_z)}{(\vec{1}_z^{t}\,\Psi^{-1}\,\vec{1}_z)} \right\rangle,$$

where

$$\mathcal{M} = \frac{(\vec{\vartheta}_z^{t}\,\Psi^{-1}\,\vec{\vartheta}_z)(\vec{1}_z^{t}\,\Psi^{-1}\,\vec{1}_z)}{(\vec{\vartheta}_z^{t}\,\Psi^{-1}\,\vec{\vartheta}_z)(\vec{1}_z^{t}\,\Psi^{-1}\,\vec{1}_z) - (\vec{\vartheta}_z^{t}\,\Psi^{-1}\,\vec{1}_z)^2}\,.$$

A.4.2 Alternate Solution:

P4 is equivalent to

$$\mathrm{P4}': \quad \min_{\vec{\alpha}}\ \mathrm{E}(\tilde{\delta}(\vec{X};\vec{\alpha}))^2, \qquad \text{s.t. } \vec{\alpha}^{t}\,\vec{\vartheta}_z = 0 \text{and } \vec{\alpha}^{t}\,\vec{1}_z = 1\,.$$

Since

$$\begin{aligned}
\tilde{\delta}(\vec{X};\vec{\alpha}) &= r(\vec{X}) - \tilde{y}(\vec{X};\vec{\alpha}) \\
&= r(\vec{X})(\vec{\alpha}^{t}\,\vec{1}_z) - \vec{\alpha}^{t}\,\vec{y}(\vec{X}) \\
&= \vec{\alpha}^{t}\,(r(\vec{X})\,\vec{1}_z - \vec{y}(\vec{X})) \\
&= \vec{\alpha}^{t}\,\vec{\delta}(\vec{X})\,,
\end{aligned}$$

where $\vec{\delta}(\vec{X})$ is a $(p+1)\times 1$ vector, with $\delta_0(\vec{X}) = -y_0(\vec{X}) = -1$, and $\delta_i(\vec{X}) = r(\vec{X}) - y_i(\vec{X});\ (i>0)$. The condition $(\vec{\alpha}^{t}\,\vec{\vartheta}_z = 0) \equiv (\alpha_0 = 0)$ reduces the effective dimentionality of $\vec{\delta}(\vec{X})$ in P4′ (by one). Therefore, keeping in mind that $\alpha_0 = 0$, P4′ is equivalent to P4″,

$$\mathrm{P4}'': \quad \min_{\vec{\alpha''}}\ \mathrm{E}\langle \vec{\alpha''}^{t}\,\vec{\delta''}(\vec{X})\rangle^2, \qquad \text{s.t. } \vec{\alpha''}^{t}\,\vec{1} = 1\,,$$

where $\vec{1}$ is a vector of proper dimension with all components equal to one, $\vec{\alpha''}$ and $\vec{\delta''}(\vec{X})$ are $p\times 1$ vectors, and $\vec{\delta''}(\vec{X}) = [\delta_i] = [r(\vec{X}) - y_i(\vec{X})];\ (i>0)$.

Since $\mathrm{E}\langle \vec{\alpha''}^{\,t}\, \vec{\delta''}(\vec{X})\rangle^2 = \vec{\alpha''}^{\,t}\, \Omega''\, \vec{\alpha''}$, where $\Omega'' = [\omega''_{ij}] = [\mathrm{E}\langle \delta_i(\vec{X})\, \delta_j(\vec{X})\rangle], (i, j > 0)$, is a $p \times p$ matrix. Forming the Lagrangian function, $\mathcal{L}$, for P4″

$$\mathcal{L} = \vec{\alpha''}^{\,t}\, \Omega''\, \vec{\alpha''} + 2\,\lambda\,(\vec{\alpha''}^{\,t}\, \vec{1} - 1)\,,$$

then differentiating $\mathcal{L}$ w.r.t. $\vec{\alpha''}$,

$$\frac{\partial \mathcal{L}}{\partial \vec{\alpha''}} = 2\,\Omega''\, \vec{\alpha''} + 2\,\lambda \vec{1} = 0\,,$$

yields $\vec{\alpha''}_{(4)} = -\lambda\, \Omega''^{-1}\, \vec{1}$. Premultiplying by $\vec{1}^t$, and using the condition, $\vec{\alpha''}^{\,t}\, \vec{1} = \vec{1}^t\, \vec{\alpha''} = 1$, yields

$$\lambda_{(4)} = \frac{-1}{\vec{1}^t\, \Omega''^{-1}\, \vec{1}}\,,$$

and

$$\vec{\alpha''}_{(4)} = \frac{\Omega''^{-1}\, \vec{1}}{\vec{1}^t\, \Omega''^{-1}\, \vec{1}}\,.$$

Thus,

$$\vec{\alpha}_{(4)} = (\,0,\, \vec{\alpha''}^{\,t}_{(4)}\,)^t\,,$$

and

$$\mathrm{MSE}^{(4)} = \frac{1}{\vec{1}^t\, \Omega''^{-1}\, \vec{1}}\,.$$

VITA

Sherif Hashem was born on December 27, 1963, in Cairo, Egypt. He earned a B.Sc. degree in Electronic and Communication Engineering in 1985, and an M.Sc. degree in Engineering Mathematics and Physics in 1988, both from the Faculty of Engineering, Cairo University, Giza, Egypt. In 1993, he earned a Ph.D. degree in Industrial Engineering from Purdue University, West Lafayette, Indiana. His research interests include neural networks, operations research, system modeling and simulation, applied regression analysis, forecasting, and genetic algorithms. Sherif is currently a Postdoctoral Research Fellow at Pacific Northwest Laboratory, P.O. Box 999, MSIN K1–87, Richland, WA 99352, USA. Internet: s_hashem@pnl.gov

Selected Publications

- Hashem, S., Schmeiser, B., & Y. Yih (1993). Optimal Linear Combinations of Neural Networks: An Overview. *Tech. Rep. SMS93–19*, School of Industrial Engineering, Purdue University. (*Proceedings of the 1994 IEEE International Conference in Neural Networks*, forthcoming.)

- Hashem, S., Y. Yih, & B. Schmeiser (1993). An Efficient Model for Product Allocation using Optimal Combinations of Neural Networks. In *Intelligent Engineering Systems through Artificial Neural Networks*, Vol. 3, C. Dagli, L. Burke, B. Fernández & J. Ghosh (Eds.), ASME Press, pp. 669–674.

- Hashem, S., & B. Schmeiser (1993). Approximating a Function and its Derivatives using MSE-Optimal Linear Combinations of Trained Feedforward Neural Networks, *Proceedings of the World Congress on Neural Networks,* Lawrence Erlbaum Associates, New Jersey, Vol. 1, pp. 617–620.

- Hashem, S., & B. Schmeiser (1992). Improving Model Accuracy Using Optimal Linear Combinations of Trained Neural Networks, *Tech. Rep. SMS92–16*, School of Industrial Engineering, Purdue University. (*IEEE Transactions on Neural Networks*, forthcoming.)

- Hashem, S. (1992). Sensitivity Analysis for Feedforward Artificial Neural Networks with Differentiable Activation Functions, *Proceedings of the 1992 International Joint Conference on Neural Networks* in Baltimore, IEEE, New Jersey, Vol. I, pp. 419–424.