Methods for addressing some practical issues in MLP regression and their application to modelling curl in papermaking.

Andrew J. Myles

Thesis submitted for the degree of
Doctor of Philosophy
The University of Edinburgh
September 1997

Abstract

Over the last decade the multilayer perceptron (MLP) artificial neural network (ANN) has been applied increasingly to nonlinear modelling problems in fields such as process control and machine vision. Nonlinear modelling is also a problem which has been studied extensively by statisticians for several decades, and in recent years several people have pointed out that standard MLP and statistical regression methods are in fact very closely related.

This is a useful observation because MLP modelling has traditionally been a somewhat trial and error empirical process. Identifying the similarity between MLP and regression methods thus offers the possibility that the large body of existing statistical theory and practice may be used to improve our understanding and use of the MLP.

This thesis adopts this approach to examining two important practical problems in MLP regression. These are: the use of robust estimators to improve the fit, particularly when the training data contains outliers, and prediction error estimation for MLP model complexity selection. The investigations into robust MLP regression discovered that only simple robust estimators are likely to be useful in most MLP regression problems. Though more sophisticated estimators have previously been suggested for this task, it is shown why these are in fact unsuited to this. Estimating prediction error is a particularly important problem in MLP regression. The investigations into estimating prediction error yielded a fast method for estimating prediction error by cross-validation and also examined its limitations. This method is particularly useful when the amount of training data is limited.

The primary motivation for investigating these two issues was the desire to use the MLP to model a phenomenon known as curl in papermaking, and to use this model to improve the yield of a papercoating process. Only a limited amount of data was available for this task, and it was suspected that the data included several gross errors. Since these are general problems in MLP regression, the techniques devised here have wide applicability and importance.

Declaration

I declare that this thesis and all work presented herein has been completed by myself and, except where indicated otherwise, is based entirely on my own research.

Andrew J. Myles

Acknowledgements

There are many people who I would like to thank for their assistance and support during this work:

- Alan Murray and Robin Wallace for starting this project, allowing me to undertake this work, and for their supervision.
- John Barnard, Gordon Smith, and the other members of the coating plant at Tullis Russell & Co. Ltd., Markinch, for their assistance and sponsorship of this project.
- To the past and present members of the neural group for various suggestions, advice and entertainment, and particularly to Mike, Pete and Robin[1], for their various pearls-of-wisdom shared over the years.
- Robin and Mike again for extensive proof-reading and suggestions.
- And finally, and predictably, to my parents, who were clearly losing faith in me ever submitting, but didn't force the point...

[1] Ranked by age, not necessarily by authority...

Table of Contents

Guide to figure locations

Guide to table locations

Chapter 1

General introduction and thesis overview

1.1. Artificial neural networks

Over the last decade, much interest has developed in applying artificial neural network (ANN) methods to solving difficult problems in many fields such as machine vision, medical condition monitoring, market forecasting and process control[1-3].

ANNs are information processing systems based loosely upon the structure of the brain, which thereby attempt to emulate its abilities to process complex information such as speech and vision, and finding complex patterns in data[4-7]. These tasks are difficult to emulate using conventional algorithmic methods simply because they are so complicated that suitable algorithms are not known. Much of the interest in neural networks has been stimulated by the demonstrated ability of simple ANNs to solve many of these types of problems without the need for a detailed understanding of the problem.

1.2. Function fitting and classification

However, despite the impressive range of problems for which ANN solutions have been developed, the basic problem addressed by the ANN in many of these is simply that of modelling nonlinear relationships between sets of variables. Indeed, two basic types of nonlinear modelling problem which are fundamental to many larger problems in process control, signal processing, machine vision and general data analysis are:

- Classification problems where the decision boundaries to be estimated are nonlinear curves or surfaces.
- Function fitting (regression) problems where a nonlinear function must be estimated from a set of noisy samples of this function.

This thesis examines some issues in nonlinear function fitting using a particular ANN known as the multilayer perceptron (MLP). The MLP and its variants are arguably among the simplest and most widely used ANNs[3], and have provided useful solutions to many important nonlinear modelling and prediction problems in fields such as:

- Modelling and controlling complex nonlinear industrial processes[8-14].

- Predicting nonlinear time-series[15] such as the behaviour of financial markets[16-18] and variations in the demand for electric power[19-21].

1.3. The original project goals

Given the interest and many reported successes in using MLPs to model and control industrial processes, the original aim of this project was to examine whether they could be used to model a phenomenon known as *curl* in papermaking.

Curl is a paper quality that cannot be measured during manufacture, and the primary project goal was to examine whether the curl measured after manufacture could be modelled as a function of other variables which are measured during manufacture. Given such a curl model, it would be possible to estimate, and hence control, curl on line from measurements of these other quantities. In process engineering, this technique is known as inferential estimation, and is one area in which the MLP is becoming used quite widely[8].

The process of fitting a MLP model to a given data set is known as training, and traditionally a least squares estimation method known as backpropagation has been used for this task. At the beginning of this project it was assumed that this method would be suitable for developing the curl model, and the project thus aimed to investigate issues such as

- integrating the MLP curl model into the production process, and
- updating the model to accommodate changes in the process over time,

rather than basic issues in the function learning task itself.

1.4. Problems, investigations and project evolution

However, once the curl modelling data became available, it quickly became apparent that much of this data could not be used to develop the MLP curl model. This was because many of the data records had missing entries, and standard MLP training methods such as basic backpropagation cannot use such data.

As methods for approaching this problem were examined, further modelling issues were identified which were considered pertinent not only to the problem of modelling curl, but to MLP function fitting problems in general. These issues were:

- What effect can outliers in the data have on the MLP fit to the data, and what can be done to minimise any possible adverse changes they may cause in the fit?
- What effect do a type of influential data, known as *leverage* points in classical regression, have in MLP regression, and how can such data be diagnosed?

- How can the MLP generalisation ability be estimated when there is too little data available to use conventional split sample validation methods? Estimation of the generalisation ability is central to the important problem of selecting an appropriate MLP for a given modelling problem.

Though only the last of these issues has received much attention in the MLP literature, the problem of function fitting with noisy data is well-known to statisticians as a *regression* problem, and all these issues have been studied extensively in the regression literature.

Regression modelling is essentially concerned with the common problem of estimating a function given a set of noisy samples of this function. This task is clearly similar to those to which the MLP is commonly applied, and while investigating the above problems it was in fact noted that the MLP and its training methods were simply a type of regression model and model fitting methods. Once this was appreciated, it was realised that a wide range of existing regression knowledge and practice could thus be applied to tackling these problems in MLP modelling.

The main aim of this project thus shifted towards investigating these issues in this manner. Once these general issues were investigated, it was felt that this would provide a important tool-box of methods based on existing regression theory and practice, which could be applied not only to the curl modelling problem, but to practical MLP modelling problems in general.

1.5. Structure of the thesis

Though the issues listed in the previous section share many common features, they also represent a number of distinct areas of statistical research. Consequently, it would not be possible to provide one coherent review chapter which covers all the relevant background material, and so this has been distributed amongst the appropriate chapters.

Since most of the techniques used in this thesis were drawn from the regression literature, chapter two reviews the basic regression concepts and terminology which are important to understanding the MLP, and on which the work presented in the later chapters is based. The MLP is introduced as a type of regression model and some of the more important issues in its use, particularly overfitting and underfitting, are reviewed.

Underfitting and overfitting are important problems which must be considered when using MLP regression. In chapter three some regularisation methods for preventing overfitting are reviewed, and a simple modification of one of these methods is examined briefly. The main reason why regularisation methods were examined in some depth was that the techniques described in chapter six are more convenient to use when training is performed with regularisation.

In chapter four the curl modelling problem is revisited and described in more detail. The particular reason for this review is to discuss why the MLP was chosen for the curl model and also to discuss the data deficiencies which lead to the investigations which are described in the chapters five and six.

Chapter five examines the issue of training MLP regression models with data which contains outliers. The standard least squares training method which is used in MLP regression can be shown to be sensitive to outliers, and this chapter examines issues in the selection and use of alternative robust training methods which are less sensitive to outlying data. In classical regression, a type of influential points known as leverage points can cause difficulties for some robust regression methods, and the effects of these points in MLP regression is also considered.

Chapter six begins with a closer examination of leverage in MLP regression. The original aim of this investigation was to ascertain leverage could be used to diagnose local overfitting in MLP regression. However, it soon became apparent that leverage could also be used for the important task of estimating the generalisation ability of fitted MLP models using standard leave-one-out cross-validation. This is a particularly useful technique for problems where the amount of data is limited, such as the curl modelling problem. A fast cross-validation method based on the fit leverages, and conditions for its performance, are discussed using the curl modelling problem.

Earlier chapters having examined a range of methods for practical MLP regression, chapter seven returns to the curl modelling problem and the application of these methods to this problem.

Finally, chapter eight provides a summary of the thesis and states its final conclusions and suggestions for future work.

1.6. Contributions to knowledge

The main contributions to knowledge contained in this work are:

- In chapter three, some minor contributions are made towards extending current understanding and use of roughness penalties in MLP regression. This work was limited simply by the realisation that there was too little time to investigate it in the depth required.
- Chapter five makes several major contributions towards current understanding of the practical issues involved in applying classical robust regression methods and ideas to MLP regression. These techniques are important when dealing with real data which may contain gross errors and other outliers.
- Chapter six makes two contributions on the issue of overfitting, the most important of which is in the area of estimating generalisation ability using a fast and

approximate method for computing the leave-one-out cross-validation score for the MLP regression models considered in this thesis. This method is of particular importance in problems where training data are limited. A minor contribution is also made in illustrating the diagnostic uses of leverage in MLP regression.

Chapter 2

Background to regression modelling and the MLP

2.1. Introduction

Chapter one stated that a regression modelling approach was adopted for developing the MLP curl model. The primary reasons for this were:

- The function fitting problems to which MLP methods are applied are generally examples of regression modelling problems, and fit neatly into existing regression theory.
- It was felt that many important problems, such as how to deal with outliers, had not been addressed adequately in the MLP literature. These are not just specific to the curl modelling problem, but are general regression problems. These problems have, however, received much attention in the regression literature, and it would clearly be sensible to use well-understood existing methods as a starting point for tackling these problem in MLP regression.

Indeed, in recent years, many neural networks researchers have realised that existing statistical theory offers answers to many important problems in MLP modelling, and this is now an area of much research activity[22-28]. This chapter:

- reviews the basic regression concepts necessary to understand the material in later chapters,
- introduces the MLP as a regression model, and
- discusses some of the important basic issues in MLP regression.

The chapter is organised as follows:

Section 2 introduces the regression problem and the 2 issues which primarily determine what models and methods are applicable for a given problem type.

Section 3 reviews basic concepts in parametric regression, least squares fitting of parametric models and the difficulties of specifying the model functional form.

Section 4 reviews basic nonparametric regression concepts, focusing primarily on kernel regression to illustrate the idea of local smoothing.

Section 5 examines the important issues of under- and overfitting, using examples from both parametric and nonparametric regression. Under- and overfitting are important problems in MLP regression.

Section 6 discusses a problem known as Bellman's curse, which relates the amount of data required to estimate a regression function to the number of variables involved in the regression problem.

Section 7 summarises the key advantages of parametric and nonparametric models in preparation for discussing the advantages of using the MLP for regression.

Section 8 introduces the MLP regression model, and reviews the MLP, MLP training methods and some practical problems in training MLPs.

Section 9 focuses in depth on the issue of under- and overfitting in MLP regression, and how to minimise them when fitting the MLP.

2.2. Basic regression concepts and terminology

Finding and expressing systematic relationships between two or more noisy variables is a common problem in science and other fields, and it is this problem that regression modelling methods address. Figure 2.1 shows a simple example of such a problem involving only two variables, x and y:

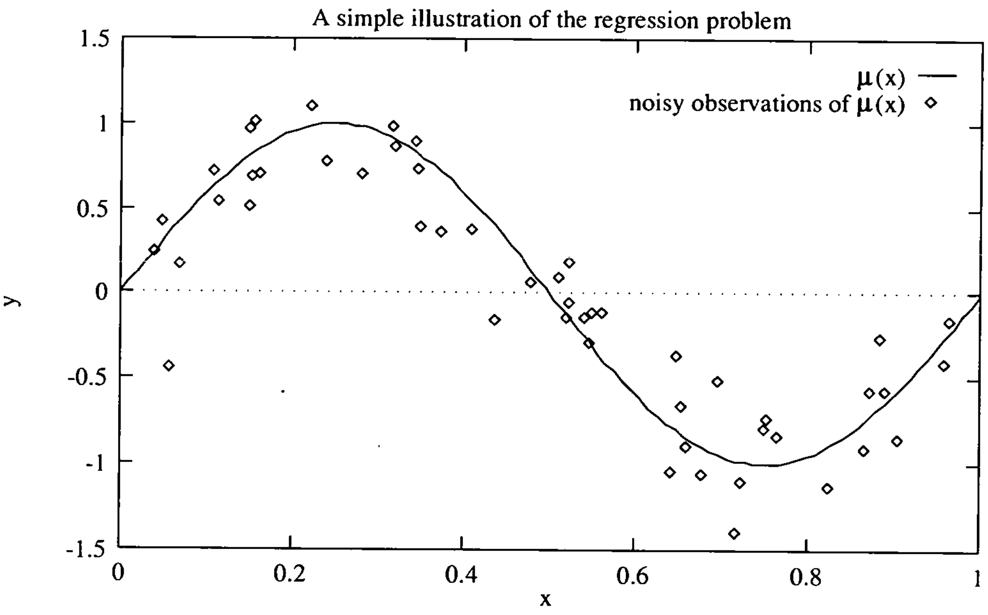


Figure 2.1: Illustration of a simple, two variable regression problem. The problem is to estimate $\mu(x)$ given the set of noisy observations of this function shown.

Here, the typical value (this will be made more precise shortly) of y depends upon x according to the *regression function* $\mu(x)$. However, one or both variables are subject to random variations which result in their observed values being scattered about this function. Since y depends systematically on x, it is usually called a *dependent* or *response* variable, and x is called an *independent* or *predictor* variable.

Given a set of data such as that shown in figure 2.1, it is often desirable to estimate the regression function (if any) which describes the systematic relationship between them. For example, knowledge of this function may be useful when examining the nature of the relationship between these variables, or it may be used to predict the response which is likely to be observed for given values of the predictor variables[29].

The aim of regression modelling methods is to estimate a regression function given a set of noisy observations of this function[29, 30]. Many methods have been developed for this task, and the choice of an appropriate method is dictated mainly by two considerations:

- How should the regression function be expressed? For example, it may be known that the regression function can be approximated closely by a specific type of function such as an exponential decay curve, or little may be known about the shape of this function other than its general properties, such as smoothness.
- How do the random variations, or *errors*, affect the observations of $\mu(x)$? The answer to this question determines the type of estimation (fitting) method which should be used in conjunction with the chosen regression model.

The next sections expand upon these issues and discuss the trade-offs involved in choosing a particular model type and fitting method. This discussion also serves to introduce further basic issues, notably underfitting and overfitting, which are of great importance in MLP regression[31].

2.2.1. How the error structure affects the regression problem

Before considering what sort of models can be used for the regression function, the important issue of how the random errors affect the data will be considered. The error structure largely dictates what methods are used to fit a model to the data, and failure to consider this may give a poor fit.

This thesis is concerned only with method based on the *fixed design*[30] or *fixed regressor*[32] model. This assumes that random errors affect only the response variable, so that the data have the form

$$y = \mu(x_1, \cdots, x_q) + \varepsilon = \mu(\underset{\sim}{x}) + \varepsilon \tag{2.1}$$

where ε is the error, $\underset{\sim}{x}$ is the vector $(x_1 \cdots x_q)^T$ and the regression function is shown as a function of q predictor variables for the sake of generality.

It is also necessary to consider what probability distribution best describes the distribution of the response errors. Since this question raises many issues, it is not discussed here, but at more appropriate points throughout this thesis.

Figure 2.2 show the fixed design model with Gaussian errors which was used to generate the data shown in figure 2.1.

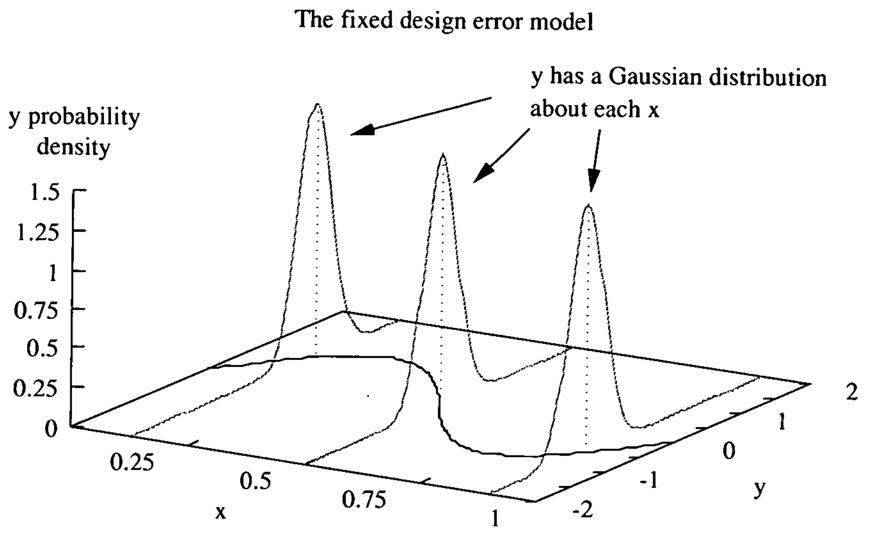


Figure 2.2: Example of a fixed design error model. Only the response variable is subject to random errors, which follow the distribution N(0, 0.25).

The fixed design assumption may seem too simple for real data, and techniques known as errors-in-variables methods[32] extend the error structure to include errors in the predictor variables. However, fixed design methods were used here because,

- most regression methods assume fixed design errors, and so the widest range of existing methods can be adopted for MLP regression when this error structure is assumed, and
- the fixed design method can actually be quite accurate for many problems, such as modelling industrial processes where most of the output variation arises within the process, which does not affect the inputs[33].

The regression function under the fixed design assumption

Assuming fixed design errors, a formal definition of a regression function can be given. If each error distribution has zero mean, then the regression function at any $\underline{x}$ is the expected value of the response[30, 34]. If many data are collected at a given $\underline{x}$, then averaging their responses will give a value close to $\mu(\underline{x})$ (c.f. figure 2.2).

2.3. Parametric models, nonparametric models, fitting and residuals

Most regression models can be classified as either parametric or nonparametric models; the difference between them being how the estimate of the regression function, or *fit*, is expressed mathematically.

Using either model type, the goal of regression modelling is to find a fit which minimises[2] the differences between the fit values and the data, or *residuals*, $\{y_i - f(\underline{x}_i)\}$, where $f(\underline{x}_i)$ is the value of the fit at $\underline{x}_i$. If the fit is a good estimate of the regression function, then the fit residuals are also good estimates of the response errors, $\{\varepsilon_i\}$. Examining the residuals is a good method for assessing the adequacy of a regression fit, and some examples of this are given in chapter seven[35].

Understanding the relative merits of the parametric and nonparametric approaches is essential for understanding the merits of using the MLP for regression.

2.3.1. Parametric models

Linear regression[35, 36] gives a good illustration of the parametric modelling approach. In linear regression, it is assumed that the shape of the regression function is known (linear), and only the model slopes and y-axis intercept (the model *parameters*) must be estimated to fit the model to the data. Formally, the linear model is

$$f(\underline{x}; \underline{\theta}) = \theta_0 + x_1\theta_1 + \cdots + x_q\theta_q \tag{2.2}$$

when $\underline{\theta}$ is the vector $(\theta_0 \cdots \theta_q)^T$ and it is assumed that

$$\mu(\underline{x}) = f(\underline{x}; \underline{\Theta}) \tag{2.3}$$

for some unknown value of the model parameters, $\underline{\Theta}$. The problem is then to find a good estimate of $\underline{\Theta}$, so that the model fits the data well.

Clearly the linear model is not appropriate for all problems, and any function can be used as a parametric model, provided that suitable parameters are included to allow it to be fitted to the data. For example, the models $\theta_1 \sin(\theta_2 x)$ and $\theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$ can both fit the 'N'-shaped regression function in figure 2.1, and so are more appropriate than a linear model.

2.3.2. Fitting parametric models using least squares

The most well-known and widely-used method for fitting parametric models is least squares (LS) estimation[32, 37]. In LS estimation, the model is fitted to the data by finding the parameter values, $\hat{\underline{\theta}}$, which minimise the sum-of-squares error,

[2] Subject to avoiding an effect known as overfitting, which is described shortly.

$$SSE = \sum_{i=1}^{n} |y_i - f(\underline{x}_i; \underline{\theta})|^2 \qquad (2.4)$$

where n is the number of data. The SSE is the sum of squared fit residuals, and so minimising the SSE attempts to improve how accurately the regression function is estimated by the fit by reducing the measured misfit between the model and the data.

Statistical aspects of LS estimation (error distribution)

Though LS estimation can be used without knowledge of the response error distribution, the method is most appropriate when the errors are independent and identically distributed variates from a Gaussian distribution.

If this the case, then it can be shown that the SSE is proportional to the negative log-likelihood of the parameters, and hence the LS estimates, $\hat{\underline{\theta}}$, are the maximum likelihood estimates (MLEs) of the unknown $\underline{\Theta}$[32].

MLE is an important estimation method in statistics because MLEs often have various desirable statistical properties (MLE is reviewed in appendix A). The properties of LS estimates, and other fitting methods, are examined further in chapter five.

2.3.3. The parametric model specification problem

A major difficulty with parametric modelling is that the shape of the regression function to be estimated is often unknown, making it very difficult to specify a suitable functional form for the model[38]. While visual inspection may suggest a suitable model function when there are only a few variables, this soon becomes a very difficult problem in higher-dimensions[39].

In such situations, nonparametric regression techniques are more suitable. These are discussed now.

2.4. Nonparametric regression

For regression problems where the shape of $\mu(\underline{x})$ is unknown and a suitable parametric model cannot be specified easily, a flexible method which can adapt to the shape of any systematic trends within the data is clearly desirable. This is the purpose of nonparametric methods[30, 38].

The basic principle underlying the nonparametric approach is to locally smooth the data to reduce the response error variations and hence recover the shape of $\mu(\underline{x})$[30, 40].

2.4.1. Kernel regression as an illustration of nonparametric smoothing

Figure 2.3 shows a typical example of local smoothing using a nonparametric method know as kernel regression. This example is discussed in some detail both to illustrate the principle of local smoothing, and because some later chapters use kernel regression analogies to explain phenomena in MLP regression.

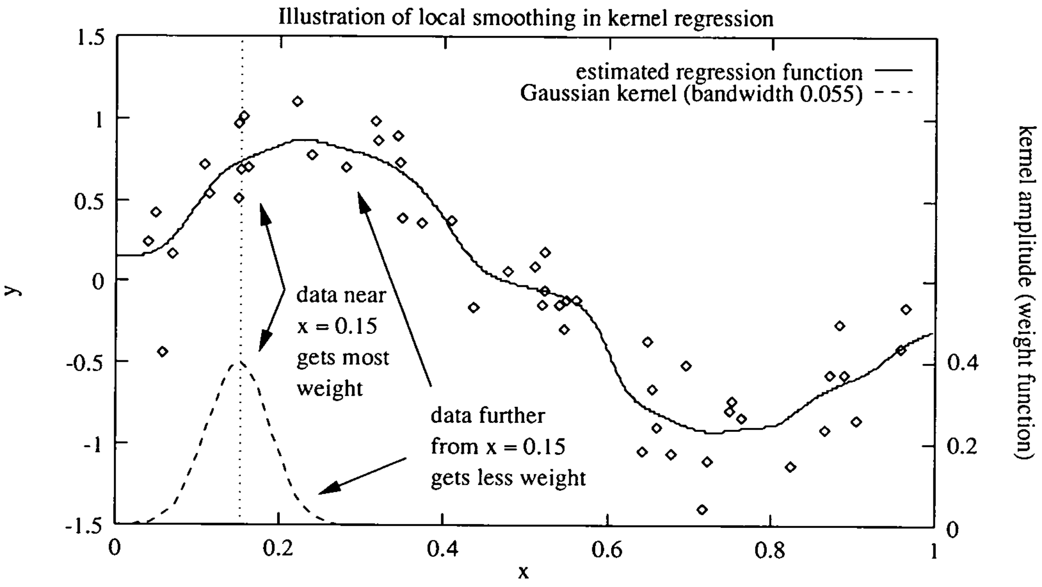


Figure 2.3: Illustration of local smoothing in nonparametric regression. $\mu(x = 0.15)$ is estimated averaging the weighted response data, where the weights are given by the dashed kernel function.

Figure 2.3 shows the kernel regression estimate of $\mu(x)$, and also details how $\mu(0.15)$ was estimated. Assuming $\mu(x)$ to be smooth, then the values of this function should be similar for all x in the vicinity of $x = 0.15$. Thus the regression function can be estimated by averaging the responses for these data, which reduces the random error variation[30, 41]. The data closest to $x = 0.15$ provide the best estimates of $\mu(0.15)$, and so the average is weighted towards these cases. In this example, the average is weighted according to the Gaussian *kernel function* shown centred at $x = 0.15$.

Formally, the kernel estimate of the regression function at x is the weighted average

$$\hat{y} = \frac{\sum_{i=1}^{n} W\left(\frac{x - x_i}{\lambda}\right) y_i}{\sum_{i=1}^{n} W\left(\frac{x - x_i}{\lambda}\right)} \tag{2.5}$$

where $W(\alpha)$ is the kernel function and $\hat{y}$ is a notation for the regression function estimate. The parameter λ controls the kernel bandwidth; that is, it controls how rapidly the kernel function decays as the distance $x - x_i$ increases. Wide bandwidth kernels smooth over a wide area, while narrow bandwidth kernels average only the data in narrow neighbourhood around x. The consequences of using a bandwidth that is too wide or narrow are examined in section 5.

Though the above description made no explicit statement about the fitting method used by kernel regression, this type of kernel regression is in fact a LS estimator. This is because taking the sample mean of the weighted response data is a LS estimator (see appendix A).

2.4.2. Other nonparametric methods

There are many other nonparametric methods in addition to kernel averaging. For example, cubic regression splines are another popular nonparametric estimation method where $\mu(x)$ is approximated by joining a series of cubic polynomial segments at their endpoints, subject to some smoothness constraints on how they are joined. This and other nonparametric methods are discussed in[30, 38, 42-48].

2.5. Underfitting and overfitting

Underfitting and overfitting can be serious problems when using parametric or nonparametric regression, though they are much easier problems to cure in nonparametric regression. Under- and overfitting are often serious problems in MLP regression, and are now discussed.

Underfitting

Underfitting occurs when the regression model cannot capture the shape of the regression function to be estimated. Figure 2.4 illustrates this concept using parametric and nonparametric regression examples, fitted to the data shown in figure 2.1.

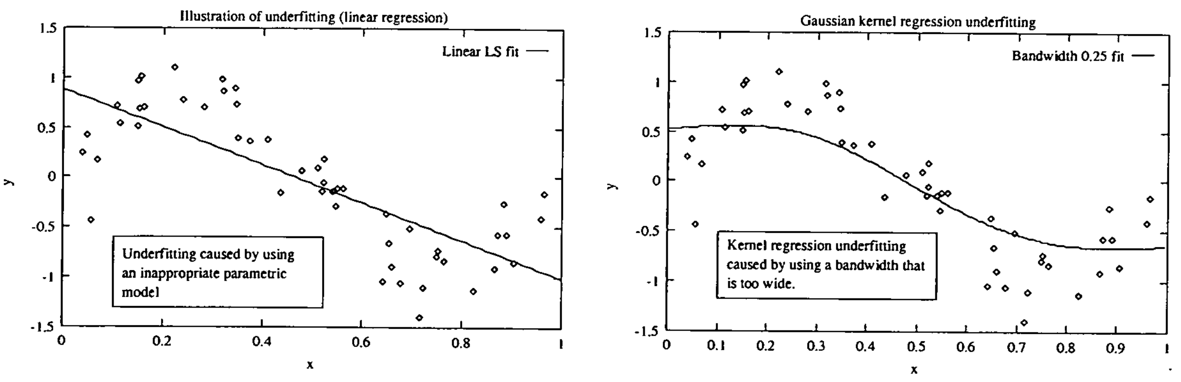


Figure 2.4: Underfitting in parametric (left graph) and nonparametric (right graph) regression

The parametric linear model obviously cannot represent accurately the regression function because it cannot capture the nonlinear relationship between the variables. In this sense, the model can be considered too simple to represent the regression function, and is thus said to be underfitted. In the kernel regression example, the use of an excessively wide bandwidth kernel has not only averaged-out the errors very effectively, but has also averaged-out the trends in the data.

Avoiding underfitting can be a very difficult problem in parametric regression because of the difficulties that finding a suitable model function to fit high dimensional regression functions can pose. Underfitting is a much easier problem to solve in nonparametric regression, because simply reducing the width of the smoothing neighbourhood (the kernel bandwidth in kernel regression) reduces the underfitting.

Statistically, underfitting is an example of estimation bias because the fit cannot converge to the regression function that it attempts to estimate no matter how much data is available to estimate this function[43]. That is, there will always be a systematic deviation between the fit and the data because the model simply cannot fit the trends in the data.

Overfitting

Overfitting is essentially the opposite problem to underfitting, and occurs when the model can fit not only the regression function, but also the random variations of the data due to the response errors. Figure 2.5 illustrates overfitting using kernel regression.

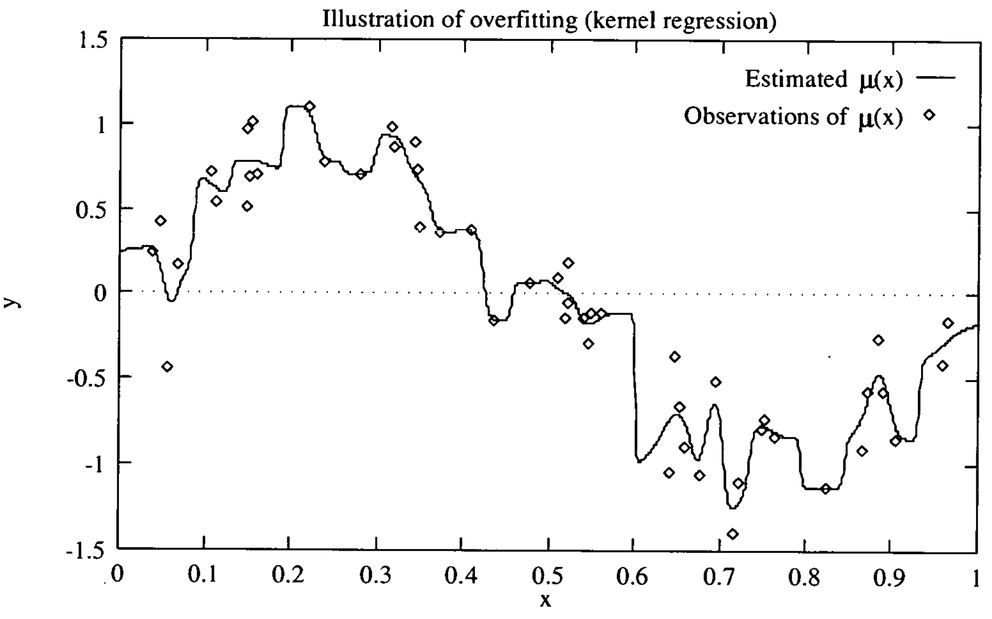


Figure 2.5: Illustration of overfitting in kernel regression

The regression fit shown in figure 2.5 follows the shape of the regression function well, but is a poor estimate of the regression function because it also follows the random data variations. This occurred because a narrow kernel was used, which does not allow for averaging of many data, and hence smoothing of the random errors.

Overfitting in parametric regression is illustrated later using the MLP. The absence of a MLP example here does not mean that it is not a serious problem.

Overfitting can be reduced in nonparametric regression simply by widening the smoothing neighbourhood. However, avoiding overfitting in parametric regression without causing underfitting instead is a difficult problem. Again, this is due to the difficulties of specifying a suitable parametric model form for complex, high-dimensional regression functions.

Statistically, overfitting is an example of high-variance estimation[43]. This is a result of the oscillations in the fit, such as those visible in figure 2.5. These oscillations are sensitive to the response error values, and so re-estimating the regression function using data with different errors will give different oscillations. This variation of the fit between different data-sets is the source of the high variance.

2.5.1. Trading-off underfitting bias versus overfitting variance

Obtaining the best possible fit to the data requires that underfitting and overfitting are minimised. However, reducing bias due to underfitting, by using a narrower kernel for example, always causes an increase in the fit variance. Conversely, reducing overfitting (variance) by using a wider bandwidth always increases the fit bias. This is known as the bias-variance trade-off or bias-variance dilemma[49].

Techniques for assessing when the best combination of low bias and variance have been achieved are discussed later in the context of MLP regression.

2.6. Bellman's curse of dimensionality

The final basic regression issue considered here relates to the amount of data required to estimate a given regression function, and is known as *Bellman's curse of dimensionality.*

Accurate estimation of complex regression functions, such as functions with many minima and maxima, requires many data to define these features accurately. A good analogy to this situation can be drawn with the Nyquist sampling theorem in signal processing, which states that a signal must be sampled at a rate greater than twice the highest frequency components in the signal if these components (the fine detail) are not to be lost due to aliasing.

However, as the number of predictor variables increases, the number of observations of the regression function required to maintain dense sampling of this function rises exponentially[29, 50]. For example, suppose ten uniformly spaced samples of a function of one variable are taken. In order to obtain a similar sampling density for a function of two variables, 100 samples would be required, distributed evenly over a fixed grid. For a function of three variables, 1000 samples over a cube would be required, and so on.

The consequence of Bellman's curse is that it is almost always difficult to estimate high-dimensional regression functions accurately, because of the vast amount of data that is required to represent the features of all but the simplest functions. The exponential increase in the number of data required to estimate functions accurately as their dimension increases is Bellman's 'curse' of high dimensionality.

MLP regression is often applied to high-dimensional data, and so Bellman's curse is a common problem in MLP regression.

2.7. A review and comparison of the regression approaches

Before introducing the MLP and discussing its useful features as a regression model, it is worth reviewing some of the relative merits of the parametric and nonparametric approaches.

Strengths of the nonparametric approach

The obvious weakness of the parametric approach is the difficult problem of specifying a suitable model for the regression function when the shape of the regression function is unknown. A bad model choice can lead to serious under- or overfitting. While nonparametric methods do require suitable smoothing parameter values (such as kernel bandwidths) to avoid serious under- or overfitting, choosing appropriate values for them is a much simpler problem than trying to specify a complex parametric model.

An additional advantage of nonparametric methods is that they do not require any model parameters to be estimated. For parametric models involving many nonlinear parameters this can be a difficult and very computer-time consuming task.

Strengths of the parametric approach

A particularly useful characteristic of parametric models in fields such as process modelling is that they provide a concise, functional summary of the regression function. To see why this is useful, note that once the model parameters have been estimated then $\mu(\underline{x})$ can be estimated at any point without the need to store the data used to compute $\underline{\hat{\theta}}$. In contrast, nonparametric methods must consult this data every time $\mu(\underline{x})$ is to be estimated, and this may require a significant amount of data storage space. In addition, for large amounts of data involving many variables, the time required to search this data to find the data which lie within the smoothing neighbourhood may be significant.

Parametric models are also useful if features of the regression function such as minima and maxima are to be found, or if this function is to be differentiated or integrated. With parametric models, these problems often can be addressed efficiently by applying appropriate standard analytic methods to the fitted model.

Combining the advantages of both approaches

While the nonparametric approach is most useful for estimating unknown nonlinear regression functions, the parametric approach also offers some practical benefits. Clearly it would be useful in some applications to combine the flexible function approximation abilities of nonparametric methods with the convenience of the parametric model. The curl modelling problem was considered to be one such application, and the MLP is one type of regression model that provides this combination.

2.8. The multilayer perceptron (MLP)

The remaining sections of this chapter now introduce and discuss the MLP as a type of parametric regression model. The issues discussed are:

- the MLP and its relationship to the regression concepts reviewed in the previous sections of this chapter,
- training (fitting) methods for MLPs, and
- underfitting and overfitting in MLP regression, and how these are usually avoided.

Avoidance of under- and overfitting is widely considered to be one of the most important problems in MLP regression[31], and provided the motivation for much of the work presented in chapter six. Consequently, this issue is discussed in most detail.

2.8.1. MLP architecture and universal function approximation

There are very many types of ANN[2], though only a few such as the MLP and radial basis functions are appropriate for regression modelling[51, 52]. Of these two, the MLP is arguably the most widely used.

The name 'multilayer perceptron' arises from the fact that a MLP is comprised of interconnected layers of simple mathematical models of the neuron (the basic brain cell). These neuron models are sometimes known as perceptrons. The neurons can be connected in many possible ways, and the combination of the neuron-types and interconnectivity in a given MLP define its *architecture.*

Figure 2.6 shows the MLP architecture used for the work presented in this thesis. The arrows show the direction of data flow through the MLP, and since all arrows point from the input layer to the output, this is known as a *feedforward* MLP.

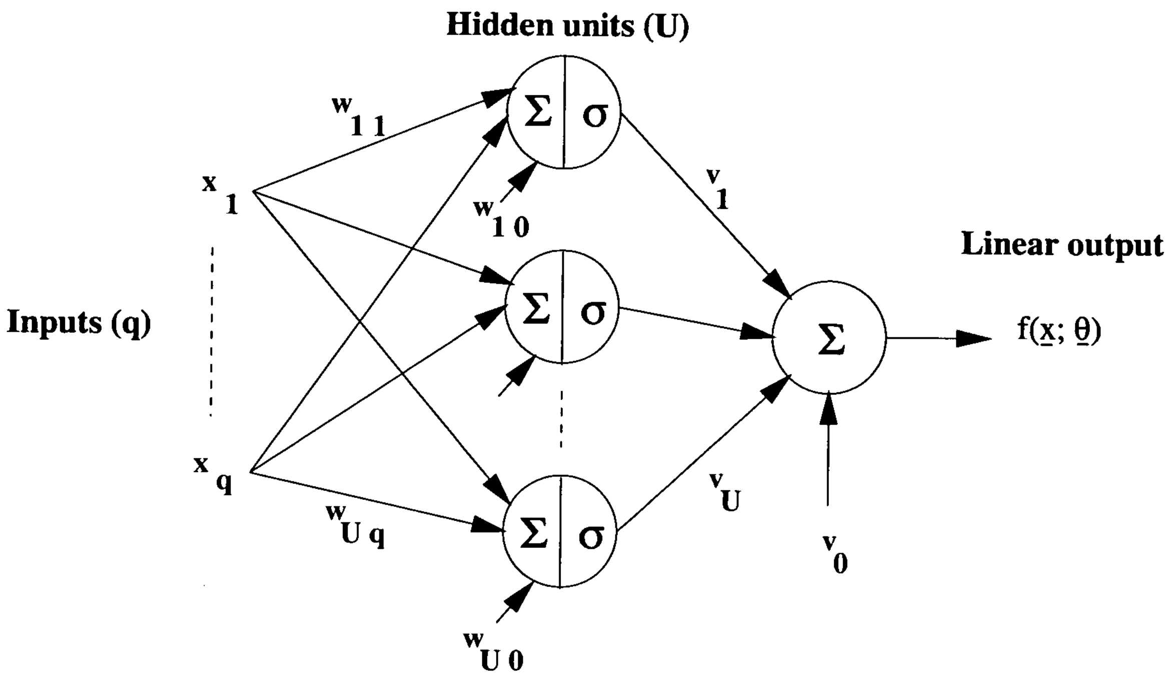


Figure 2.6: A feedforward MLP with one hidden layer of nonlinear neurons and a linear output neuron.

This MLP implements the parametric model

$$f(\underline{x};\underline{\theta}) = v_0 + \sum_{u=1}^{U} v_u \sigma(\underline{w}_u^T \underline{x} + w_{u0}) \tag{2.6}$$

where the parameter vector is comprised of the input and output weights, $\underline{w}_u$ and v_u, and the input and output biases, w_{u0} and v_0. When presented with a vector of inputs, $\underline{x}$, each hidden unit computes an *activation* which is a weighted sum of the inputs to which a constant bias is added[3]. These activations are transformed using the logistic sigmoid function

$$\sigma(\alpha) = \frac{1}{1+e^{-\alpha}} \tag{2.7}$$

to form the hidden unit outputs, and the final MLP output, $f(\underline{x};\underline{\theta})$, is a weighted sum of the hidden outputs. In summary: each hidden unit computes a nonlinear function of a linear combination of the inputs, and these are added to form the MLP output.

[3] Here bias refers to the intentional addition of an offset, such as the fit y-axis intercept, v_0. It should not be confused with the use of bias as an unwanted offset in an estimator's value.

Universal function approximation using the MLP

Much of the interest in using MLPs for regression can be traced to the fact that they are universal function approximators[51, 53]. This means that a there is always a MLP (which may require infinitely-many hidden units) that can approximate any smooth, continuous function exactly.

While few problems need infinitely-many hidden units, MLPs with even a single layer comprised of a few hidden units can still approximate a surprisingly wide range of functions well. This curve-fitting flexibility makes MLP regression competitive with nonparametric regression techniques for problems where the shape of the regression function is not known[24, 49].

Much of the success of the MLP in applications such as process control can be attributed to their combining of a compact parametric model form with the function-fitting flexibility of nonparametric methods.

2.8.2. MLP training (fitting) methods and practical problems

The process of estimating the parameters which fit the MLP to the data is usually called training. The methods used for training affect many aspects of how MLPs are used. In particular, understanding how MLPs are trained is necessary to understand how a technique known as early stopping works.

This review of basic MLP techniques thus concludes with a brief review of common training methods.

Fitting the MLP to the data

Training a MLP is conceptually no different from fitting a linear regression model; both require parameters to be estimated to fit the model to the data. The main difficulty in MLP regression is that the SSE is a complex, nonlinear function of the parameters, and so it is not possible to derive analytically the parameter values which minimise the SSE. Instead, iterative optimisation methods are used to find the SSE minimum[54-56].

Training by batch backpropagation

The first, and still commonly used, MLP training algorithm was a simple gradient descent method known as backpropagation[4], or usually just backprop[57]. Backpropagation searches for a SSE minimum by taking successive small steps in the direction in which the SSE function decreases most rapidly. The parameter update rule is

$$\underline{\theta}_{n+1} = \underline{\theta}_n - \eta \nabla_{\underline{\theta}} SSE(\underline{\theta}_n) \tag{2.8}$$

where the learning rate parameter, η, controls the step size made on each update, and is normally fixed in backpropagation.

This version of backpropagation is known as *batch* training because all the training data must be used to compute the SSE gradient direction before any parameter update is made. A common variant known as *stochastic* or *on-line* backpropagation uses a different gradient search method, where the components of the gradient direction due to individual training data are used to make the parameter updates[58]. This method is more suitable for time-series modelling than regression work, and so is not discussed further.

Faster training methods

Backpropagation is a rather unstable optimisation method, and this can cause its user much frustration[58].

If η is large and the descent path stumbles into a region where the SSE drops rapidly for small changes of the parameters, then the next training step changes the parameter values by a large amount. This can result in the new parameter estimates being further from the SSE minimum than the previous estimates, and so the minimum is not found.

This problem can be overcome by using a smaller learning rate, but training often becomes extremely slow, as very many of the small parameter updates may be needed to reach the SSE minimum. In practice, much time may be wasted simply tinkering with the learning parameter to achieve stability or to accelerate slow convergence.

Due to these problems of stability and convergence time, fast, stable optimisation methods such as conjugate gradient methods with line- searching and Newton-type methods have become increasingly popular for training MLPs[54, 58-61]. During this project, both conjugate gradient methods and the Nelder and Mead downhill simplex method[55, 62] were used, and both gave far faster training than backpropagation, with the further advantage of stability.

Local minima: a practical training problem

Since the SSE is a complex nonlinear function of the parameters, it often has several local minima in addition to a global minimum. All iterative optimisation methods may become trapped in these local minima, leading to poor fits to the training data.

If a poor fit is obtained, and it is suspected that this is due to a local minimum rather than underfitting, then the simplest method for addressing this problem is to train several MLPs using different random starting parameters values. This increases the likelihood of finding a SSE minimum (not necessarily global) which gives a good fit to the data,

[4] Strictly, backpropagation refers only to the version of the chain rule that is used to calculate the gradient. However, the term is often used to refer to the training process itself[23].

though training many MLPs does increase the time required to develop the MLP regression model.

Hidden unit saturation: a practical training problem

Another practical problem is known as hidden unit saturation. Saturation occurs when most or all of the hidden unit activations become large, forcing their outputs to be close to 0 or 1 for sigmoidal hidden units[63]. This almost binary behaviour of the hidden units usually gives a poor fit to the data.

Saturation is a difficult situation to escape from once entered, because the SSE gradient becomes very small, and so training slows dramatically. Fortunately, saturation can usually be avoided by scaling all predictor and response variables to small ranges near zero. Commonly used transforms include translating and scaling to fit the range $[0, 1]$ and standardising by subtracting the mean and dividing by the standard deviation. The need for, and possible consequences, of saturation and scaling are discussed at appropriate points in later chapters.

2.9. Underfitting and overfitting in MLP regression

Avoiding under- and overfitting is one of the most important problems in MLP regression[24], and so the remaining sections of this chapter discuss this problem, and some methods for addressing it, in some depth.

The cause of under- and overfitting

The complexity of the functions which can be approximated by a given MLP increases as more hidden units are used. If a MLP contains too few hidden units to approximate the regression function being estimated, then it cannot fit the data well no matter how much training data is available. This is underfitting in MLP regression.

Conversely, a MLP with many hidden units may be able to fit not only any regression function within the data, but also spurious trends which are due to the random response errors, and are not part of the regression function. This is overfitting in MLP regression.

Finding the right fit complexity to avoid under- and overfitting

Under- and overfitting can be avoided by using a MLP with enough hidden units to avoid underfitting, but no so many units as to allow overfitting[5]. However, since the shape of the regression function is usually unknown, it is not possible to specify a suitable number

[5] Other methods for complexity control are discussed later and in the chapter three. However, the goal of all complexity control methods is the same, and so this discussion considers only varying the number of hidden units for controlling the fit complexity.

of hidden units in advance of fitting the MLP. Instead, it is usual to search for signs of under- or overfitting after training, and then to re-train with more or fewer hidden units respectively.

The training data cannot be used to detect under- or overfitting

One method which cannot be used to assess whether under- or overfitting has occurred is to compare how well MLPs with different numbers of hidden units fit the training data. Figure 2.7 illustrates this point. Here, sets of 10 MLPs with random initial parameter values and 1 to 6 hidden units have been trained using 25 data drawn randomly from the data shown in figure 2.1. One reason for using only half of the data in figure 2.1 is to encourage overfitting by having sparse training data, and another important reason for holding-back some of the data is discussed in the next section.

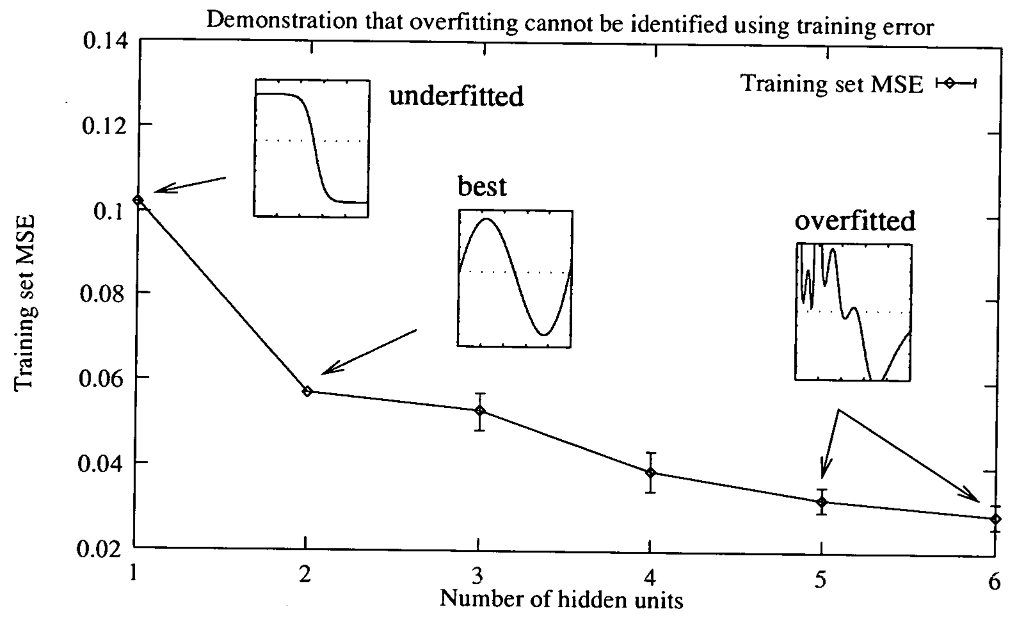


Figure 2.7: Mean square fit error versus number of hidden units for MLP fits to 25 of the data shown in figure 2.1. Error-bars are ± 1 standard deviation of the training errors obtained for the 10 different MLP fits with each number of hidden units.

The sub-figures within figure 2.7 show the typical fit shapes obtained using different numbers of hidden units. It is clear that only the MLP with 2 hidden units provides a good estimate of the regression function. However, this cannot be ascertained by comparing how well the MLPs fit the training data, because overfitting caused by increasing the number of hidden units always gives a lower training error. Here it can be seen that the error decreases monotonically as the number of hidden units increases, and this trend gives no indication that underfitting occurs with less than 2 hidden units or that overfitting occurs with more than 2.

Some other method clearly is required to identify under- and overfitting so that the right number of hidden units can be chosen for a given problem. This is normally achieved by comparing the MLP *generalisation abilities*, which is discussed now.

2.9.1. Generalisation ability and prediction error

Generalisation ability refers to the how well a trained MLP fits the regression function represented within its training data. If the fit gives accurate predictions of the regression function for all $\underline{x}$ of interest, then the MLP is said to show good generalisation ability.

Both underfitting and overfitting cause poor generalisation ability, the former due to the bias in the fit and the latter due to the wild fit-oscillations which are characteristic of overfitting (see figure 2.7 for example).

To determine how many hidden units are needed to obtain a MLP fit with good generalisation ability, it is first necessary to quantify this property. In both classical and MLP regression, generalisation ability is quantified using the *expected prediction error* of the fitted model[64]. Using the MSE as an example error measure, the expected prediction error is

$$\textit{Expected prediction error} = E\left[\,|y - \hat{y}|^2\right] \tag{2.9}$$

where the expectation is over all possible noisy observations of the regression function, and $\hat{y}$ is a common short-hand notation for the fitted model's regression estimate, $f(\underline{x}; \hat{\underline{\theta}})$.

The expected prediction error is minimised for a perfect fit to the regression function, and increases as the fit deviates further from this function. One easy way to confirm this is to note that when there is an infinite amount of training data, then a MLP with a suitable number of hidden units[49, 53] can estimate the regression function exactly. If this MLP is trained using LS estimation, then the SSE divided n tends to (2.9), and so (2.9) is minimised for a perfect fit.

In practice, the joint distribution of $\underline{x}$ and y is unknown (if it was known, the regression function could be estimated exactly by computing the expected y for any given $\underline{x}$), and so the expected prediction error must be estimated. If N new data become available after the MLP is fitted, then one estimate of the expected prediction error is

$$PE = \frac{1}{N}\sum_{v=1}^{N} |y_v - \hat{y}_v|^2 \tag{2.10}$$

which is simply an estimate of (2.9) based upon a finite sample.

Thus new data can be used to estimate the generalisation ability of MLP regression models by measuring how well the fitted model predicts this data. Note that this test data cannot also be used for training, as the prediction error can then be driven to zero by overfitting to this data during training. Rather than collecting new test data after training the MLP, it is usual to collect one set of data, but to reserve some of this data solely for the purpose of testing the MLP generalisation ability after training. This technique is used very widely, and is now discussed.

2.9.2. Data splitting

The most widely used method for estimating prediction error in MLP regression is data splitting[65, 66], which is also known as the hold-out method in the MLP literature[67].

Data splitting simulates model testing with new data by excluding some of the available data from training, and using this data to estimate the prediction error after the MLP is trained. Typically, the available data are split into two, or possibly three, data sets:

- A *training* set. This is used to train the MLP.
- A *validation* set. This is used to estimate generalisation ability for the purpose of deciding which of two MLPs has the best generalisation ability.
- An optional *test* set. This data is used to provide a final estimate of the generalisation ability once the model complexity has been tuned using the validation set.

Separate validation and test sets should be used because the validation error may be an over-optimistic estimate of the true generalisation ability. This is because the MLP complexity is determined by varying the MLP fit complexity and then selecting the MLP which fits the validation data best, and so this process is biased toward selecting models which may happen to overfit the validation data[66, 67]. The test set is thus used to provide an estimate of the generalisation ability using data which has not been used to train the MLP or to tune its complexity. In practice, however, limited data availability often results in the validation-set being used both to tune the MLP complexity and to estimate the final generalisation ability.

Figure 2.8 shows how data splitting can be used to find the number of hidden units which minimises under- and overfitting for the MLPs used in the previous example (figure 2.7).

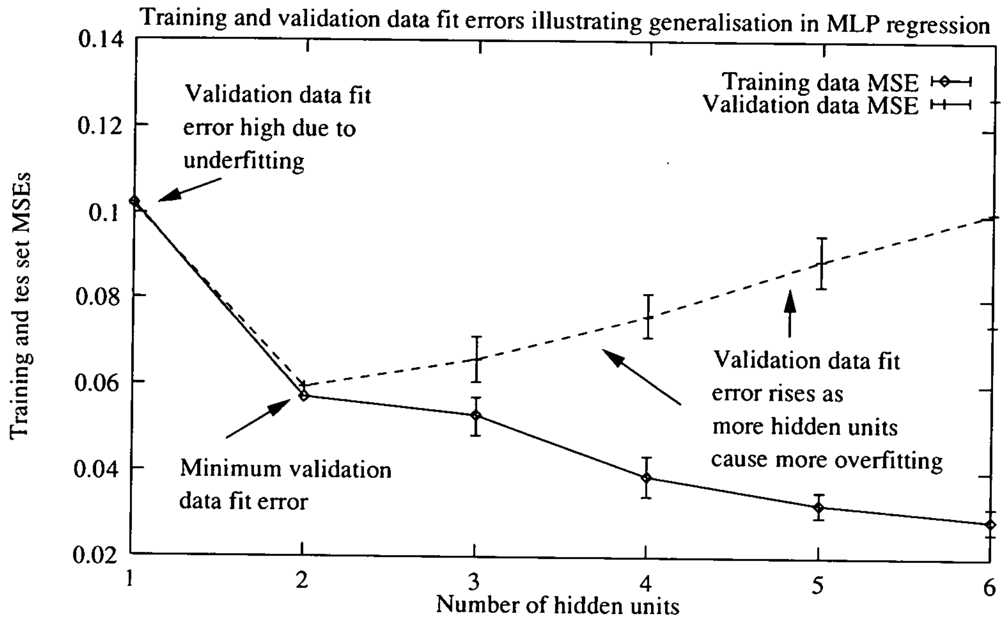


Figure 2.8: Training and validation data MSEs for the MLPs of figure 2.7. Each point is the average MSE for 10 MLP fits, and the error-bars are ± 1 standard deviation of the MSEs for these 10 MLPs.

This figure shows 2 error curves,

- the training data MSE shows how well the MLP fits the 25 training data, and
- the validation data MSE shows how well the MLP fits the 25 data which were held-back.

Though the training errors provide no indication of which MLPs are underfitted or overfitted, the validation data estimates of the prediction error show clearly that the best generalisation ability (i.e. least under- and overfitting) occurs for the MLPs with 2 hidden units.

2.9.3. Practical data splitting issues

While the simplicity of data splitting is very appealing, there are 2 practical issues to consider when using this method. These are

- deciding how much of the data to reserve for validation and testing, and
- ensuring that the validation data is not all concentrated in one small region.

Deciding how much data to reserve for validation

Deciding how much data to reserve for training and how much to reserve for validation (and testing, if a test set is to be used) is a contentious issue. This section discusses the problems this poses, and some methods used to address them.

Reserving too little validation data does not allow thorough testing of the MLP fit. This can result in overfitting or underfitting passing un-noticed where there is no validation data to detect their occurrence. Additionally, the estimates of the prediction error may be quite variable due to the small sample size[66, 68], and can change significantly depending on which portion of the data is used for model validation. If the validation errors are considered too variable to be reliable, then there is little point in using such a small validation set.

However, using too much of the available data for validation and testing may not leave enough data in the training set to estimate the regression function accurately[66, 69]. This can be particularly serious when the model involves many predictor variables, since Bellman's curse means that the regression function will be sampled sparsely, and so may be quite poorly defined by the data. Reducing the amount of training data further for validation and testing may remove completely some feature of this function.

A compromise must be made between reserving too little or too much data for validation and testing. In practice, typically 10 to 20 percent of the data is reserved for model validation; though this is a somewhat heuristic rule, and can vary appreciably between practitioners, and with whether or not test data is also to be reserved. Some recent theoretical results[68, 69] suggest how much validation data to reserve, though no experimental confirmations of these results have been reported yet.

Obtaining maximum validation set coverage

Once the validation-set size is determined, the next issue to address is ensuring that the validation data covers the same predictor variable range as the training data: that is, ensuring that the validation data is not concentrated in one or two small regions, but tests the model as extensively as possible over the domain of interest. Random splitting is used commonly in MLP regression, and does appear to work quite well in practice. However, it has been suggested that random splitting can often give poor validation coverage, and that systematic splitting methods should be used[65, 66]. This approach does not appear to have been examined in MLP regression to date, but certainly merits further investigation. It was not investigated here because the curl data was ordered in a way which meant that reasonably good systematic coverage could be obtained easily.

2.9.4. Complexity control using the method of early stopping

The final MLP regression technique reviewed here is the widely-used method of early stopping. Early stopping is an alternative to adjusting the number of hidden units when trying to avoid under- and overfitting[31].

When using early stopping, a large number of hidden units is normally used, so that underfitting is unlikely to be a problem. To then avoid overfitting, the validation error is monitored continuously during training, and training is stopped when the lowest validation error occurs[67, 70, 71].

The principle on which early stopping is based is as follows. In the initial stages of training, it is assumed that the MLP first learns the gross trends in the data, that is the regression function, and that overfitting to the finer error detail occurs only after prolonged training. Consequently, the validation error drops during the early stages of training when the regression function is being learned, but increases when overfitting begins. Training is thus stopped when a rising validation error signals the onset of overfitting[69, 70].

Figure 2.9 illustrates the use of early stopping. This figure which shows the variation in the validation error during training for one of the MLPs with 6 hidden units from the previous example (see figures 2.7 and 2.8).

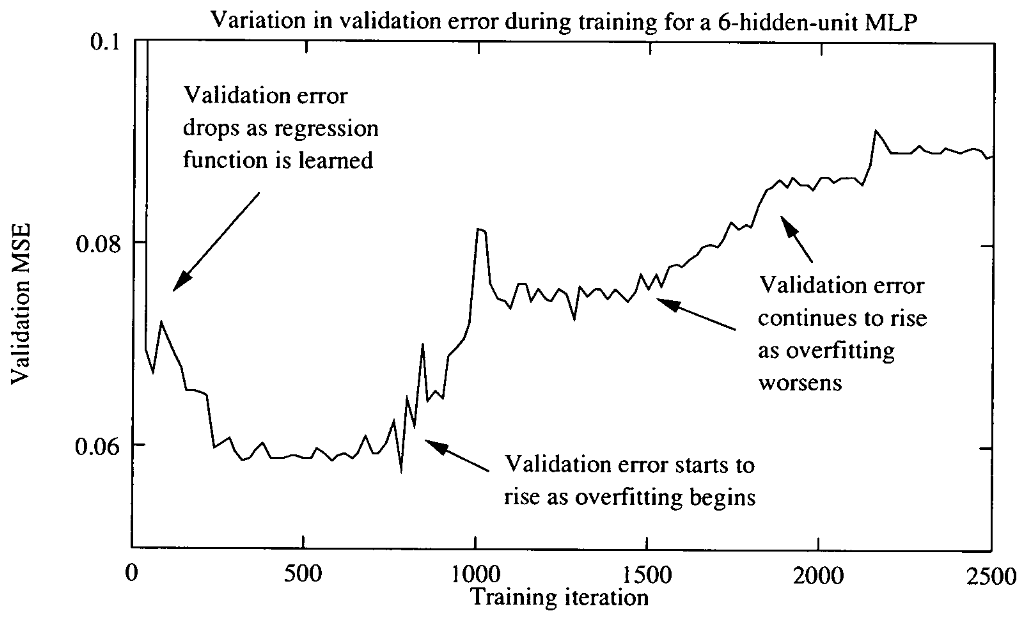


Figure 2.9: Variation of the validation error during training for one of the MLPs with 6 hidden units used in the previous examples. The error decreases at first as the regression function is learned, and then increases as overfitting begins.

In figure 2.9, the best generalisation (lowest prediction error) occurs between 300 to 700 training steps, and further training leads to degradation of the generalisation ability due to overfitting. Note that the lowest validation error obtained in this case is similar to that obtained previously using 2 hidden units (see figure 2.8). This illustrates an important reason for using early stopping, namely that a good fit can be obtained using a wide number of hidden units, and so less searching is required to find the optimal number of units. A further reason, as some examples in later chapters show, is that it is possible to obtain a lower validation error using early stopping than may be obtained for any combination of hidden units and training to completion.

Issues to consider when using early stopping

This review of early stopping concludes with a brief review of some known problems with this method[22, 41]. In most cases, these problems are in fact of little practical concern.

One problem is that the validation error may exhibit several minima during training, the first of which may not give the lowest validation error. Thus stopping training when the first minimum is encountered may cause underfitting. Figure 2.9 shows one of these early minima near 50 training steps. The simple solution to this problem is to continue training until all minima have been found, or until it is unlikely that no new lowest minimum will be found. While this will increase the MLP training time, training times with early stopping tend to be short anyway (because of the early stopping) compared to varying the number of hidden units and training the MLP to completion.

Another issue with early stopping is that there is no reason why the regression function must be learned well before early stopping occurs. Indeed, if training starts from overfitting, then further training leads only to worse overfitting. In practice, this particular problem is somewhat unlikely due to the use of small initial parameter values to avoid hidden unit saturation during training. However, using very many hidden units does increase the likelihood of rapid overfitting occurring before a good fit is learned. Consequently, different numbers of hidden units should still be tried when using early stopping, though as shown earlier, finding the optimal number of hidden units is less critical then when early stopping is not used.

A final, mostly theoretical, problem with early stopping is that it is difficult to analyse the statistical behaviour of the parameter estimates obtained using this method. This is simply because the path followed by the parameters toward a given minimum of the training error function depends upon the initial parameter values, and hence 2 different start points may lead to quite different parameter estimates depending on how the minimum is approached. A more practical aspect of this problem is that since the path followed towards the training error minimum depends on the starting parameter values, one MLP may give a good fit while an identical MLP trained using different starting conditions may not. In addition to the problem of avoiding local minima, this is another reason why MLPs are often trained several times with different starting parameters.

2.10. Summary

This chapter has reviewed the basic regression concepts and terminology used in the remaining chapters of this thesis.

Review of general regression topics

The regression problem was introduced, including a review of how

- the error structure, and
- how much is known about the shape of the regression function

affect the choice of methods for estimating a regression function.

Both parametric and nonparametric regression methods were reviewed. In terms of MLP regression, the key issues reviewed in this chapter are

- what under- and overfitting are, and how they result from fitting models to the data which are too simple or complex, and
- that the problems of under- and overfitting are more easily solved in nonparametric regression.

Avoiding under- and overfitting in parametric regression is a difficult problem because this requires a suitable parametric model to be specified for the regression function, but the shape of the regression function is usually unknown.

Review of MLP regression topics

The MLP was introduced as a flexible, nonlinear parametric regression model. One of the primary reasons for using the MLP is that it can fit a wide range of functions, and MLP regression can hence rival the flexibility of nonparametric methods for problems where the shape of the regression function is unknown.

As is also the case for nonparametric methods, careful complexity control is required to avoid serious underfitting or overfitting when using MLP regression. If either of these occurs, then the fit will be a poor estimate of the regression function. Optimising the fit complexity to minimise the degree of under- and overfitting requires

- methods for measuring or estimating the fit prediction error, and
- methods for controlling the complexity, so that the prediction error can be minimised.

Data splitting was reviewed as a popular method for estimating the prediction error, and varying the number of hidden units and early stopping were reviewed as two possible methods for controlling the fit complexity.

Chapter 3

Regularisation and smoothing

3.1. Introduction

As discussed in chapter two, under-fitting and overfitting are among the most important problems to consider when training MLPs[31]. Complexity control to prevent overfitting was achieved either by reducing the number of hidden units or by using early stopping. This chapter continues the discussion of how to control the fit complexity by looking at some examples of another method of complexity control: regularisation penalties.

One advantage of using regularisation is that this method can allow more sophisticated complexity control than the methods used in chapter two. However, the most important property of regularisation in the context of this project is that when it is used to prevent overfitting, training can continue until the training error function is minimised (i.e. no early stopping). This is necessary when using a technique described later, and it was the desire to use this technique for the curl modelling problem which stimulated this examination of regularisation methods.

Chapter overview

Two types of regularisation penalty are examined in this chapter: roughness penalties based on splines, and weight decay. The chapter is structured as follows.

Section 2 gives an overview of how regularisation is implemented in MLP training, and the principles underlying regularisation.

Section 3 looks at roughness penalties for smoothing MLP fits. Roughness penalties have received a little attention in the MLP regression literature, and interest in them continues to grow. They offer an intuitively appealing approach to smoothing, and so it was decided to develop them further with a view to using them for the curl problem. This investigation discovered a number of practical limitations on their use for MLP regression. Noise jittering methods are closely related to roughness penalty regularisation and so are compared with the roughness penalty approach.

Section 4 provides a review of weight decay smoothing penalties.

Section 5 discusses a problem which was uncovered while examining weight decay, namely how linearly transforming the data can affect the shape of fit. A minor change to the standard weight decay penalty is suggested to give invariance to translations of the training data when fitting the MLP. A simple example illustrates how this gives more intuitively appealing smoothing behaviour than standard weight decay.

3.2. What is regularisation?

Regularisation includes various techniques used to stabilise the solutions of numerically unstable problems known as *ill-posed* problems[72]. In ill-posed statistical estimation problems, the estimates are sensitive to small perturbations of the data, and this causes high estimate variances. Regularisation places additional constraints on the values of the estimates which limit their variances. This variance reduction is usually obtained at the cost of some added estimate bias[73]. By adjusting the amount of regularisation used in a given problem, the trade-off between estimate bias and variance can be controlled.

MLP overfitting gives fits which have low bias but are sensitive to perturbations of the training data. In fact, MLP overfitting is an example of a problem which is ill-posed through using an overparameterised model; there is too little data to assign unique values to the parameter estimates. Reducing the number of hidden units attacks this problem directly by reducing the number of parameters to be estimated during training. In contrast, by constraining the MLP parameter estimate values, regularisation reduces the *effective number of parameters*[74] or *degrees of freedom*[75] available for fitting without actually reducing the MLP size.

Implementing regularisation in MLP regression

The commonest method of implementing regularisation in MLP regression is to add a term to the training error function which penalises the learning of fits which are too complex. For example, if the SSE is used to measure fit error, then the regularised error function has the form

$$Error = SSE(\underline{\theta}) + \lambda J(\underline{\theta}) \tag{3.11}$$

where the value of the penalty $J(\underline{\theta})$ increases with increasing fit complexity and its contribution to the total error is controlled by the non-negative parameter λ. In the statistical literature, this is often known as the penalised likelihood [75] or penalised distance[76] method.

The fit complexity is controlled by the value of λ. A small λ suppresses the effect of the regularisation penalty in (3.11) and so biases training towards minimising the SSE. If λ is very small then there is little complexity control and so overfitting may occur. Conversely, a large λ biases training toward minimising the complexity rather than the

SSE. If λ is too large then the complexity penalty may prohibit the SSE from being reduced to a low enough level to allow a good fit to the data: that is, underfitting can occur.

Two questions which using regularisation thus raises are:

- what sort of penalty function should be used, and
- what value of λ should be used?

Choosing a penalty function

Ideally, the penalty function should be chosen to incorporate any prior knowledge about the shape of the regression function. For example, if it is known that this function is monotone, then the penalty should strongly penalise non-monotonic fits.

In practice, however, there is often very little or no prior knowledge about the shape of the regression function. Consequently, it may be necessary to try various penalties to obtain the best generalisation performance. This is the reason why 2 different penalty types are examined here.

Setting the value of λ

The value of λ is normally chosen to minimise the fit prediction error. This usually requires the MLP to be fitted for various values of λ and the fit prediction error estimated. The fit with the lowest error is then used as the regression function estimate.

To avoid underfitting even when λ is very small, a relatively large number of hidden units should be used when using regularisation to control the fit complexity.

3.2.1. Why use regularisation for fit complexity control?

One advantage regularisation offers over adjusting the number of hidden units is that very fine complexity control can be exercised by making very small changes in λ. Another advantage over early stopping and varying the number of hidden units is that the form of $J(\underline{\theta})$ can be tailored to enforce properties of the fit such as smoothness. Such properties usually cannot be enforced easily by controlling the number of hidden units or using early stopping (see[31] for some examples using early stopping).

My main reason for using regularisation for the curl modelling problem, however, was to aid the use of the technique discussed in chapter six. This technique requires training to continue until the error function is minimised, thus prohibiting the use of early stopping. The error function (3.11) can, however, be minimised when the regularisation penalty is used to prevent overfitting.

3.2.2. MLP regularisation using roughness and weight decay penalties

Various penalties have been used in the MLP literature to control the fit complexity[77-79]. The two types which are considered in this thesis are roughness penalties and weight decay penalties.

Roughness penalties have received some attention in the MLP literature[79-82], but are not yet used widely. I felt, however, that these offer a more intuitive and extensible approach to smoothing than weight decay, and so decided to investigate them further. This investigation was further motivated by the fact that such penalties have already been examined in detail in the regression literature as the basis of smoothing spline regression, and so a useful range of existing theory and experience could be drawn upon.

Weight decay is by far the most popular penalty for complexity control. It has been discussed widely in the literature[77, 78, 83, 84] and thus will not be discussed in depth in this chapter. The main discussion of weight decay in the second half of this chapter instead focuses on a suggested minor change to a standard weight decay penalty to remove its dependency on where the data origin is placed.

3.3. Roughness penalties and splines

Overfitted MLPs exhibit high *roughness* in order to fit closely as many of the training data as possible. Here, roughness refers to the fact the fit must deviate far from linearity to do this. Visually, this is apparent as wild oscillation of the fit between local maxima and minima. Mathematically, this is manifest as large values for the second and higher order derivatives of the fit[85-87].

One way to smooth the fit to prevent overfitting is to penalise the learning of fits with large high order derivatives. This idea has been used previously by numerical analysts and statisticians as the basis for smoothing spline regression[30, 38, 43, 86, 87]. Curiously, the large existing literature on splines has been neglected in many studies of roughness penalties for MLP regression. However, the spline literature provides a rich source of ideas for roughness penalties, as well as providing much guidance about their properties. The spline penalties are also quite similar to other penalties used previously for MLP regression[80-82], and so an understanding of spline penalties can also be used to predict the likely properties of these other penalties.

While investigating the use of spline-based penalties for smoothing MLP fits, important practical issues surrounding the use of these penalties became apparent. This discussion focuses mainly on these practical issues and their implications for the use of roughness penalties.

Why use MLPs like smoothing splines?

Given that various methods are known for constructing cubic splines, one point to address before proceeding is to justify the use of the MLP as seemingly yet another method for constructing splines.

Firstly, it should be noted that a MLP with a limited number of hidden units may only be able to approximate the true spline smooth, and so the MLP and spline fits may differ significantly. However, there is no obvious reason why the MLP fit should necessarily be better in such cases, and any differences could be reduced by using more hidden units.

A more practical point is that smoothing splines require many segments when there are many data, and a simple MLP may provide a more compact and convenient approximation to the true smoothing spline in some problems. Regression splines are another alternative which require fewer segments than smoothing splines, but these require the number of knots, and their positions, to be determined, which can be a difficult problem[43, 48].

Thus there is some practical justification for investigating spline penalties for MLP regression.

3.3.1. Two dimensional cubic splines and MLP regularisation

The simplest splines are two dimensional cubic smoothing splines. This section provides a brief overview of the properties of these splines as a prelude to using them in roughness penalties.

Two dimensional LS cubic smoothing splines are functions which minimise the error (3.11) when

$$J(\underline{\theta}) = \int_a^b \left| \frac{d^2 f(x;\underline{\theta})}{dx^2} \right|^2 dx \qquad (3.12)$$

integrated over the range of the predictor variable[6]. The cubic smoothing spline is unique for a given λ[38], and is comprised of cubic polynomial segments between the data, joined so that the first and second derivatives of the spline are continuous[30, 46, 87]. This segmented polynomial construction makes splines very flexible curve fitting tools, which can fit features which single polynomials cannot, such as plateaus and other flat regions[89]. The spline smoothness is controlled by varying λ; increasing λ forces the second derivatives of the spline to zero and so drives the smooth towards the LS linear regression line.

[6] This is also seen with limits of $\pm\infty$ to avoid boundary problems in theory[88]. Natural splines, which are linear outside the interval $[a, b]$, satisfy both forms[38, 43].

Cubic spline-based penalties for MLP regression

The penalty (3.12) leads to a useful, intuitive and theoretically well-understood type of smoothing, and so it was decided to investigate its use for smoothing MLP fits. Unfortunately, a closed-form expression for the value of (3.12) cannot be derived for MLPs with sigmoidal hidden units. Thus an approximation of the integral is required for training, and the most obvious candidate is the numerical quadrature formula

$$J(\underline{\theta}) = \Delta x \sum_{s=0}^{s=\frac{b-a}{\Delta x}} \left| \frac{d^2 f(x;\underline{\theta})}{dx^2} \right|^2_{x_a+s\Delta x} . \tag{3.13}$$

In practice the constant Δx can be absorbed into λ by defining $\lambda_{new} = \lambda_{old}\Delta x$, and thus omitted from (3.13), giving the discrete cubic spline penalty

$$J(\underline{\theta}) = \sum_{s=0}^{s=\frac{b-a}{\Delta x}} \left| \frac{d^2 f(x;\underline{\theta})}{dx^2} \right|^2_{x_a+s\Delta x} . \tag{3.14}$$

This penalty was used to generate the spline regularised examples and measured roughnesses which are presented throughout this thesis.

The derivatives in (3.14) can be computed by deriving a mathematical expression for the MLP fit derivative, or by using the finite difference approximation

$$\frac{d^2 f(x;\underline{\theta})}{dx^2} \approx \frac{f(x-\Delta x;\underline{\theta}) - 2f(x;\underline{\theta}) + f(x+\Delta x;\underline{\theta})}{\Delta x^2} . \tag{3.15}$$

Finite difference spline smoothing is discussed in[86, 90, 91]. Faster training is possible with the finite difference penalty if many of the fit values required for computing the SSE can be re-used in (3.15).

3.3.2. Practical issues in the use of the two dimensional penalty

There are two issues to consider when using the spline penalty, namely

- how many smoothing points should be used, and
- whether to use analytic or finite derivatives.

The issues, and some possible solutions to the serious problems posed by the first issue, are discussed now.

Practical point 1: setting the interval width, Δx

The most important and difficult problem is choosing a suitable width for the interval between the smoothing points, Δx.

If narrow intervals are used then there are many smoothing points, and this can increase the MLP training time considerably. However, if wide intervals are used, then overfitting may occur within the intervals. This is demonstrated by figure 3.1, which shows two fits obtained using MLPs with 4 hidden units. One fit was obtained using 10 equi-spaced smoothing points in the interval $0 \le x \le 0.9$ (giving $\Delta x = 0.1$) and the other fit was obtained using 19 points (giving $\Delta x = 0.05$). The λ values have the ratio 10/19 in an attempt to give the same $\lambda J(\underline{\theta})$ value, and hence fit, in both cases.

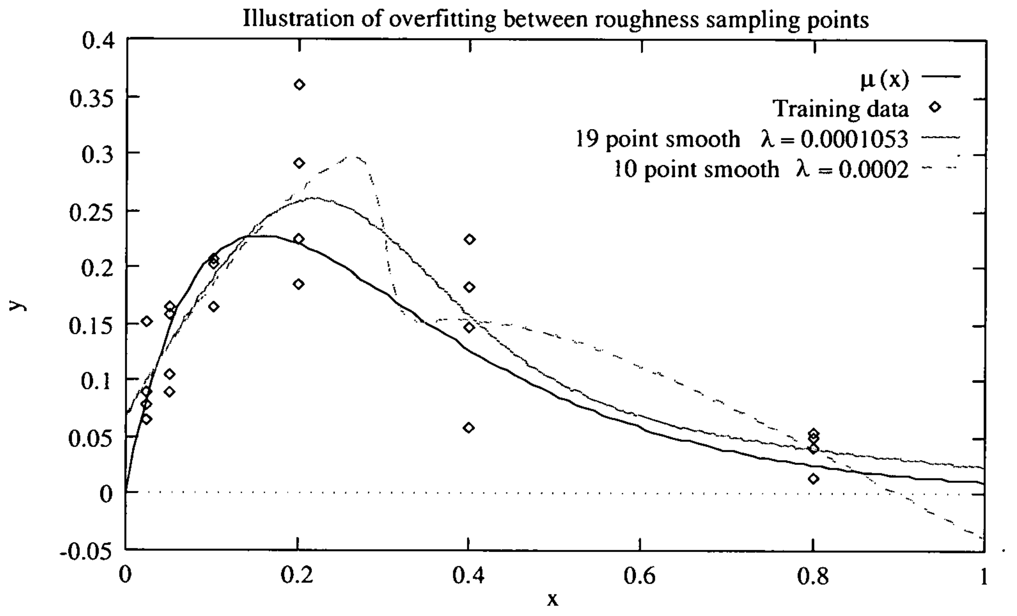


Figure 3.1: Demonstration of how overfitting may occur when using too few smoothing points. The fit obtained using 10 points shows two kinks between the smoothing points at 0.2, 0.3 and 0.4.

The fit obtained using 10 smoothing points shows two kinks which lie between smoothing points at $x = 0.2, 0.3$ and 0.4. These kinks thus do not contribute to the measured fit roughness, and so are not suppressed during training. This problem is also considered by Minai[81], and arises because measuring the fit roughness using a finite set of sample points will miss any overfitting between these points. The likelihood of this increases as the interval Δx becomes wider. When the number of smoothing points was increased to 19, the fit kinks disappeared, and did not appear even when training was restarted with various initial weights and biases.

Since the minimum Δx required to avoid overfitting depends on the number of hidden units, the function to be estimated and the training data, it is difficult to suggest any general rules which may be used to estimate a suitable Δx prior to training. Hence, attention focussed on finding ways for detecting overfitting after training, so that remedial action could be taken.

While visual inspection can be used in two and three dimensional problems, spotting overfitting this way may be difficult for higher dimensional fits. One method for identifying overfitting in all dimensions is to use a validation set. However, this reduces the amount of training data.

A very effective method for detecting overfitting is to reduce Δx after training and to re-compute the roughness to search for overfitting within the original smoothing intervals. For example, if Δx is halved after training, then the fit roughness measured using (3.14) should approximately double, because almost twice as many derivative samples will be summed. A larger increase indicates that overfitting may have occurred. This is illustrated in table 3.1, which shows the roughnesses measured at the end of training and after training using half the interval width for the fits shown in figure 3.1.

Δx	0.1	0.05	0.025	ratio
10-point fit	34.6 (train)	208 (after)	-	6.01
19-point fit	-	352 (train)	704 (after)	2.00

Table 3.1: Comparison of the measured roughnesses at the end of training with those measured using half the smoothing interval width after training. The roughness for the overfitted MLP increases by 3 times more than would be expected if no overfitting occurred.

Using half interval widths, the roughness doubles as expected for the 19-point fit, but increases by a factor of 6 for the 10-point fit. This large increase indicates that high fit curvature has been found within the original intervals, and hence that overfitting has occurred.

Compared to visual inspection for detecting overfitting, this method has the advantage of being easily extended to higher dimensional problems when suitable roughness penalties are used. It also has the advantage over the use of a validation set in that the fit can be tested even in regions where little or no data exist. For example, reserving a validation set in the above example could not have detected the overfitting near $x = 0.3$ because there is no data available to test this region.

Computer time requirements versus smoothing interval width

If overfitting is detected after training, then the MLP must be re-trained using a narrower smoothing interval. This process may have to be repeated several times, which can result in long model development times. Starting with a very narrow interval clearly reduces the chance of overfitting and hence the need for re-training, but requires a long training time. It is desirable to develop methods which can prevent overfitting during training while using the largest possible Δx to reduce the required computer time.

One possible method (which has not been tested) for addressing this problem is to combine the roughness penalty with a weight decay penalty. Weight decay smoothers do not sample the fit roughness and hence do not have the associated problem of overfitting between smoothing points. Thus relatively wide intervals could be used if some weight decay is added to discourage overfitting between the smoothing points. Other possible approaches to this problem include

- increasing the number of smoothing points once weight decay has been used to prevent initial overfitting, and
- adapting the smoothing interval based on occasional narrower interval (and hence computer-intensive) roughness measurements during training.

A further possibility is to search for an alternative to the sigmoid curve for the hidden unit transfer function, for which (3.12) can be integrated exactly without the need for computer-intensive numerical approximations. This is probably the most sensible approach to this problem, because it eliminates completely the need to choose a suitable smoothing interval width and to use computer-intensive numerical integration methods. New results using this approach are given in[92].

Practical issue 2: analytic versus finite difference derivatives

The second issue concerns the choice between the use of analytic derivatives or finite difference derivatives for the roughness penalty. If overfitting occurs, then the values of these derivatives may differ significantly, leading to different estimates of the fit roughness. This is shown in table 3.2, which gives the roughnesses for the previous example measured using finite differences.

Δx	0.1	0.05	0.025	ratio
10-point fit	212 (train)	2361 (after)	-	11.1
19-point fit	-	339 (train)	697 (after)	2.06

Table 3.2: Roughnesses for the fits of figure 3.1 computed using the finite difference derivative approximation.

Here it can be seen that the 10-point-fit roughnesses are significantly larger than those given in table 3.1, while the 19-point-fit roughnesses are quite similar. The reason the analytic derivatives are lower in the 10-point case is because the spline penalty has forced the fit to be almost linear except at the kinks, and the roughness measured at over these linear regions is very low. In contrast, the finite difference derivatives effectively measure the curvature of the parabola which passes through the points used to compute the derivative. If these points do not lie on a straight line, then the roughness will be high. Hence kinks in the fit can be detected even when no roughness measurements are taken at the kink positions.

This result suggests that using finite difference derivatives is superior to using analytic derivatives for avoiding overfitting between the roughness sample points. However, problems could also occur if finite derivatives are used during training. For example, a situation may arise where the analytic derivatives are high, but where the roughness sample points lie on a smoother curve. Though the finite difference variant was not studied in sufficient depth to observe such behaviour, this would simply be analogous to the well-known problem of aliasing in sampled data systems[93].

Usefully, comparing the analytic and finite difference derivatives offers another means of detecting overfitting; they differ when overfitting occurs. However, further work is required to assess if one method is generally better than the other in terms of avoiding overfitting during training and computer time requirements.

3.3.3. Overall assessment of the two dimensional penalty

Overall, the two dimensional spline-based penalty was found to be a useful penalty which gave good, smooth fits in a number of simulation studies. It has also been discussed in depth in the numerical analysis and regression literature, and so the basic properties of such penalties are well-understood.

However, the practical issues surrounding the trade-off between finding a suitable maximum smoothing interval width and the considerable amount of computer time that may be required when using too narrow intervals must be addressed if this penalty is to receive wider acceptance in MLP regression. These issues were not addressed here because other, more important issues merited attention.

There are also more significant issues to be addressed if the spline-based smoothing penalties are to be used in problems involving many predictor variables, including the curl modelling problem. These issues are discussed now.

3.3.4. Spline-based penalties in more than two dimensions

So far, the spline smoothing penalty has been considered only for problems involving one predictor variable. However, most problems to which MLPs are applied involve many predictor variables.

One interpretation of (3.12) is that it provides a measure of the bending energy in a thin straight rod[7] when deflected, and this suggests a method for extending the smoothing penalty to higher dimensions. For a thin plate in two dimensions, the equivalent energy

[7] Notably a draftsman's spline, from where the term was first adopted.

measure is[94]

$$J_2(\underline{\theta}) = \int\limits_{plate} \left| \frac{\partial^2 f(x_1, x_2; \underline{\theta})}{\partial x_1^2} \right|^2 + 2\left| \frac{\partial^2 f(x_1, x_2; \underline{\theta})}{\partial x_1 \partial x_2} \right|^2 + \left| \frac{\partial^2 f(x_1, x_2; \underline{\theta})}{\partial x_2^2} \right|^2 dx_1 dx_2 \quad .(3.16)$$

Smoothing splines obtained using this penalty are thus called thin-plate splines. In higher dimensional problems the general thin-plate penalty is[38, 86]

$$J_m(\underline{\theta}) = \sum_{\alpha_1 + \cdots + \alpha_q = m} \frac{m!}{\alpha_1! \cdots \alpha_q!} \int\limits_{-\infty}^{\infty} \left| \frac{\partial^m f(\underline{x}; \underline{\theta})}{\partial x_1^{\alpha_1} \cdots \partial x_q^{\alpha_m}} \right|^2 d\underline{x} \qquad (3.17)$$

where $0 \le \alpha_i \le m$, q is the number of predictor variables and $2m > q$. This last requirement is technically necessary to ensure that the smooth is well defined, and the fit may not be smooth if it is violated[86].

Unfortunately, high dimensional MLP roughness penalties based on (3.17) are very computationally-intensive to use for two reasons

- the number of terms in the sum grows combinatorially as q, and thus m, increase, and
- the number of smoothing points required to fill the space of interest grows exponentially with its dimension; another manifestation of Bellman's curse. Note that this exponential increase is due to the increase in dimension, and is not related to the issue of using enough smoothing points to prevent overfitting of the type discussed previously.

From experience, the thin-plate penalty can become computationally demanding for use in 3-dimensional problems which require many smoothing points, and will most likely be impractical without specialised computer resources[8] for almost all MLP regression problems where q is greater than about 5.

Since the curl modelling problem involves many predictor variables, spline-based roughness penalties clearly cannot be used, unless

- the number of variables (i.e dimensionality) can be reduced dramatically, or
- less computationally-demanding, high-dimensional penalties can be found.

Given roughness penalties' intuitive appeal and utility in low-dimensional problems, it is worth considering means for increasing their useful dimensionality.

[8] Assuming 1996 high-end workstation speeds.

Some possible approaches to higher dimensional roughness regularisation

One possible approach to the dimensionality problem which has often been associated with smoothing splines is to limit the regression model to the class of generalised additive models[43, 48, 95, 96]. An additive regression model has the form

$$f(\underset{\sim}{x}) = f_1(x_1) + \cdots + f_q(x_q) + constant \tag{3.18}$$

and so reduces the overall regression problem from $q+1$ dimensions to a one-dimensional and q two-dimensional problems by disallowing interactions between the variables. This may of course lead to some model bias, but it is generally hoped that this will be less than the bias from which all smoothers suffer in high-dimensional problems[38, 95, 96], and will be an acceptable price for reducing the problem to manageable dimensions. The individual functions in (3.18) can be implemented using separate MLPs, or the MLP architecture can be modified to prevent interactions between predictor variables by ensuring that only one input leads to any hidden unit[24]. The roughness penalty for an additive model is simply the sum of the individual penalties for each function. In combining the penalties, each term can be weighted by an individual regularisation parameter if optimisation of the individual parameters is considered feasible[86, 95]. If simple interactions between some variables are desired then these can be regularised by adding suitable low-dimensional thin-plate roughness penalty terms[97].

Another means of reducing dimensionality is to approximate high-dimensional functions using sums of products of lower-dimensional functions. There is an example of this approach using spline basis functions in[98], and it is also the principle on which tensor product splines are based[41].

Though successful application of these methods to MLP regression will require some research to address their practical issues, the usefulness of these approaches has been demonstrated previously in the regression literature, and there is clearly much scope for employing roughness penalties in reasonably high-dimensional problems if some architectural constraints are placed upon the types of regression functions which can be realised by the MLP.

Finally, recent work[92] which has examined the use of non-sigmoidal hidden unit transfer functions provides very promising possibilities for using spline-like smoothing penalties for high-dimensional problems without the need for computer-intensive numerical evaluation of the thin-plate penalty integral.

3.3.5. The Tikhonov roughness penalty

The first roughness penalty used for MLP regression was Bishop's Tikhonov penalty[79, 80]. This is hence the roughness penalty that other MLP researchers are most likely to be familiar with and use. This section thus compares the Tikhonov penalty with the spline penalty discussed in the previous sections.

Bishop's penalty derives its name from its origins in a method used by Tikhonov and Arsenin for solving ill-posed problems in numerical analysis[72]. The general form of the penalty is

$$J(\underline{\theta}) = \sum_{s=1}^{S} p(x_s) \left| \frac{d^2 f(x_s; \underline{\theta})}{dx^2} \right|^2 \tag{3.19}$$

which can be considered a generalisation of the discrete cubic spline penalty where a weighting function $p(x)$ has been introduced. To allow re-use of fit values computed for the SSE, Bishop suggests that $p(x)$ should be the empirical density function of the training data: that is, the training data should also be used as smoothing points[79]. In contrast, the spline penalties discussed previously assume that the smoothing points are distributed uniformly over the region of interest, irrespective of the training data distribution. This comparison of the Tikhonov and spline roughness penalties focuses mainly on this difference.

Smoothing regions of low density data

One disadvantage of weighting the smoothing penalty according to the data density is that the smoothing intervals will be widest in regions of low-density data, possibly resulting in overfitting manifest as kinks in the fit between the smoothing points. It could be argued that this overfitting would be relatively unimportant, since it occurs in regions where there is little data. However, sparse data in a given region does not always mean that the region is unimportant; for example the measurements might be the most expensive to make, and thus important. Even if the region does not have special significance, sharp ridges and valleys in the regression surface resulting from overfitting may cause problems if, for example, the MLP is used as part of a control system which uses gradient-based optimisation methods to find minima of the regression fit.

Smoothing regions of high density data

It seems intuitively plausible, however, that weighting the smoothing penalty according to the data density could be advantageous in regions where the data is most dense. As the local training data density increases, the local fit contribution to the SSE also increases because more squared fit errors are summed. If the local smoothing is not increased to compensate for this, then the SSE may dominate in determining the local fit, possibly causing local overfitting.

Cubic splines do, however, adapt very well to non-uniform data[87] and so such compensation may not be necessary. This was confirmed in simple comparisons of the Tikhonov and spline penalties with non-uniform artificial data, where little significant differences were noted between the quality of the fits.

Using the Tikhonov penalty in higher dimensions

The final comparison of the Tikhonov and spline penalties focuses on their applicability to multidimensional regression problems. Bishop suggests that the Tikhonov penalty can be used in problems of arbitrary dimensions simply by adding the derivative with respect to each MLP input to the penalty[79]. However, this will result in the additive model spline penalty instead of the thin-plate penalty, assuming the smoothing mesh is uniform. Thus overfitting due to interactions between variables may not be suppressed when using the Tikhonov penalty in high dimensional problems.

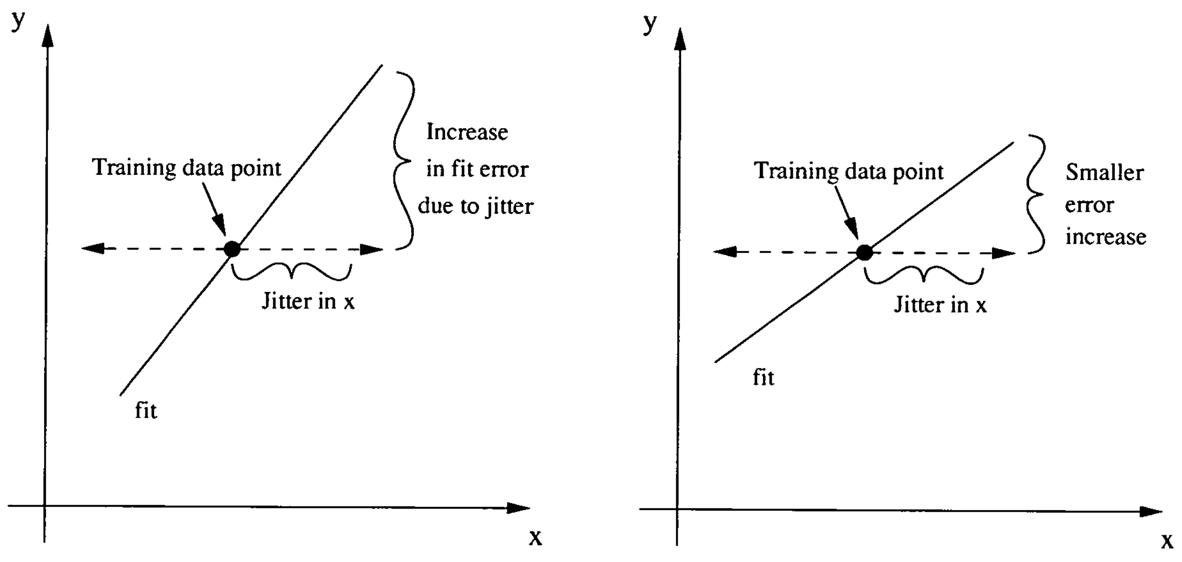
Conclusions on the Tikhonov penalty

The Tikhonov penalty appears to have shortcomings which limit its applicability to data sets with regions of low density data and to problems involving more than one predictor variable. For these reasons, this penalty was not used for any of the work reported in this thesis.

3.3.6. Roughness penalties and noise jittering

Jittering is the addition of random noise to the predictor variables, and has enjoyed considerable success and popularity in neural network research as a method for avoiding overfitting[77, 99, 100]. New noise is usually added on each training cycle, though a larger set of training data with pre-computed noise can also be used. It has been shown that this technique is closely related to training using a specific type of roughness penalty[100, 101].

An intuitive illustration of how jittering smooths MLP fits is shown in figure 3.2. Perturbing the point in the left hand example in this figure results in a large increase in the fit error, while perturbing the point in the right hand example by the same amount causes a smaller change in the error. Thus the average error when jittering is higher for fits with steep gradients, and so jittering forces the learning of fits with low gradients: that is, fits parallel to the line (or plane) on which the predictor variables lie. Complexity control is achieved by varying the jitter variance; larger perturbations cause larger increases in the error, and so result in more smoothing.



Jitter causes a large increase in the fit error when the fit gradient is large.

Increase in error is less when the fit has a smaller gradient.

Figure 3.2: Illustration of how jittering favours fits with low gradients, leading to a 'levelling'-type smoother.

An asymptotic analysis (i.e. valid only for suitably-large training-data sets) of jittering by Bishop[67, 101] shows that training with jitter is similar to training with the roughness penalty

$$J(\underline{\theta}) = \frac{\sigma^2}{2}\sum_{i=1}^{N}\left[\left(\frac{df(x_i;\underline{\theta})}{dx}\right)^2 + \frac{1}{2}(y_i - f(x_i;\underline{\theta}))\frac{d^2 f(x;\underline{\theta})}{dx^2}\right] \tag{3.20}$$

where N is the number of training data and σ^2 is the variance of the jitter noise, which is varied to adjust the smoothing in the same way as λ is when using regularisation. Bishop further argues that the first term will dominate under certain conditions[9], giving the penalty

$$J(\underline{\theta}) = \frac{\sigma^2}{2}\sum_{i=1}^{N}\left(\frac{df(x_i;\underline{\theta})}{dx}\right)^2 . \tag{3.21}$$

For uniformly spaced training data, this is a discrete *linear spline* penalty[47].

[9] An notes that these conditions will not be valid in general, except for very large data sets[100].

Comparison of jittering with roughness regularisation

Though analyses of jittering have been shown it to be similar to training with a roughness penalty, it has some undesirable characteristics.

One problem with jittering is that the type of smoothing obtained depends both on the probability density function of the jitter and also on the choice of fit error function[100]. The SSE is not an appropriate measure of fit error for all problems, and so various fit error functions may be tried in practice[102]. If different error functions are tried, then the type of smoothing applied to the fit will also vary when using jitter. Hence it is difficult to vary the fit error function while maintaining the same type of smoothing. In contrast, the fit error function and smoothing penalty can be chosen independently when regularisation is used.

A second problem with jittering, which it shares with the Tikhonov penalty, is that smoothing is localised around the training data. Thus the least smoothing is applied in regions of sparse data, but this is where overfitting is most likely and hence where fit smoothing is wanted most.

A third problem with jittering is that training with jitter does not lead to a minimum of any training error function. Instead, the training error drops until eventually it oscillates noisily around a level set by the competition between (over)fitting the data more closely to reduce the fit error and the jitter smoothing. The main reason for using regularisation for the curl modelling problem was to allow use of a technique which requires the training error to be minimised. This technique thus cannot be used if jittering is used.

3.3.7. Spline-based roughness penalties: conclusions

The first sections of this chapter have

- examined spline-based roughness penalties for MLP regression,
- examined the relationship between these penalties and a similar penalty that has been discussed in the existing MLP literature, and
- discussed some trade-offs between using the related methods of roughness penalty regularisation and noise jittering for smoothing.

The main results drawn from this work and the experience gained with using roughness penalties is that they are useful smoothing penalties, but a number of practical issues must be addressed before they can be used easily in all but simple, low-dimensional problems. These issues are

- choosing a sampling interval that is small enough to avoid overfitting, but not so small as to require much computer time to train the MLP, and

- simplifying the penalty for use in high dimensional problems without requiring an exponential increase in the number of sample points with dimension.

Some possible solutions to these problems have been suggested, but were not examined here because other more important issues demanded attention. It was thus decided to use only weight decay for the curl modelling problem.

Other recent work has shown further promising ways of addressing the computational problems associated with roughness penalties.

3.4. Smoothing using weight decay penalties

Since roughness penalties cannot be applied easily to problems involving many predictor variables, it was decided to use only weight decay for the curl modelling problem. Weight decay is used commonly for two reasons, namely

- smoothing the MLP fit to prevent overfitting[78, 83], or
- for weight elimination. Weight elimination attempts to minimise during training the values of any weights or biases which do not contribute significantly to the overall fit. These redundant parameters can be removed after training[84, 103, 104].

This discussion is concerned only with smoothing.

How does weight decay smoothing work?

Weight decay penalises the sizes of the MLP weights and biases during training. This smooths the MLP fit because

- small input-to-hidden-layer weights give small hidden unit activations. Small activations limit the hidden unit output range to the almost linear central section of the sigmoid transfer curve, hence limiting the amount of nonlinearity that the fit can exhibit.
- Small hidden-layer-to-output weights limit the amount by which the MLP output can change when a hidden unit's output changes between its maximum (1 for sigmoids) and minimum (0 for sigmoids) values. This suppresses the formation of sudden, large jumps in the fit.

While these effects do smooth the fit, the exact nature of the smoothing is not as intuitive as it is with the roughness penalties discussed previously. This is one reason why roughness penalty approaches are receiving more interest for smoothing MLP fits.

3.4.1. The weight decay penalty function

The most widely used weight decay smoothing penalty is the squared Euclidean length of the parameter vector,

$$J(\underline{\theta}) = \underline{\theta}^T \underline{\theta} = |\underline{\theta}|_2^2 \ . \tag{3.22}$$

Other distance measures can be used but the Euclidean length is most popular, mostly because

- it is similar to the sum-of-squares fit error expression, and
- its gradient is easy to compute, which simplifies the programming of weight decay training routines and reduces the amount of computer time required to train the MLP.

Weight decay, ridge regression and ill-conditioning

Weight decay is not unique to MLP regression but is used quite widely in classical regression under the name ridge regression[105-107]. The motivation for using the penalty (3.22) in ridge regression is not, however, to smooth the fit. Indeed, ridge estimation is commonly used in linear regression, where the fit is already smooth.

In ridge regression, the penalty is used to improve the numerical stability of the parameter estimates when certain types of ill-conditioning arise. Weight decay has the same effect in MLP regression, and some later examples show how this side-effect of weight decay can be useful.

3.5. Using regularisation with transformed training data

One question that arose concerning the use of weight decay was how linearly transforming the training data could affect the shape of the fit when using weight decay. Training data are often scaled and translated in MLP regression in an attempt to avoid hidden unit saturation during training. The predictor variables may also be rotated about their origin when using some data preprocessing methods such as principal components analysis[108, 109].

It is known that linear ridge regression is not invariant to linear transformation of the training data[110, 111]. This means that if the data are scaled, shifted or rotated and the model fitted, then applying the inverse transformations to the fit usually will not give the same fit that is obtained by fitting directly to the untransformed data[112]. One effect of this is that transforming the data may result in better estimation of the underlying regression function, or it may make the fit worse. For this reason it is usually recommended that the raw data are not transformed without good reason[111].

In MLP regression the training data usually must be transformed to avoid saturation, and so it was decided to examine how this could affect the fit when using weight decay or roughness penalty smoothing. Given that the transformations applied to avoid saturation are usually quite arbitrary, it was decided that invariance to as many transformations as possible would be desirable. This would eliminate the unfortunate choice of a bad transform as a culprit when trying when trying to diagnose why a good fit cannot be obtained.

How should linear transformations affect the MLP fit?

Translating or rotating the training data does not affect the shape of the regression function which this data describes. Thus, it is intuitively desirable that translating or rotating the training data should result in a similarly rotated or translated version of the original fit when using smoothing penalties.

It is also clear that invariance to arbitrary scaling cannot be expected when using smoothing penalties. This is because compressing or expanding the data by different amounts in different directions qualitatively changes the surface shape needed to fit this data well. Thus, the fit cannot be expected to follow arbitrary data scalings unless the regularisation penalty is also changed to reflect these scalings.

The next sections examine how rotation and translation affect fits obtained using roughness penalties or weight decay, to check whether they are invariant under these transformations. It is shown that standard weight decay is not invariant to translations, but that invariance can be obtained by making a simple change to the penalty. An example follows to show how translation invariance gives more intuitively appealing smoothing behaviour.

3.5.1. Effect of rotating the predictor variables

This section considers how rotating the predictor variables rigidly about the data origin affects the fit when using roughness penalties or weight decay. It is shown that both the thin-plate roughness penalty and weight decay penalty are invariant to this transformation, and thus satisfy one of the properties expected of these penalties.

Rotation and roughness penalties

The value of the thin-plate spline penalty does not change as the fit is rotated rigidly about the data origin[86]. Thus training a MLP with rotated data and thin-plate regularisation (with rotated smoothing data) results in a rotated version of the fit which is obtained using the original, un-rotated data.

Rotational invariance is not shared by the additive spline or Tikhonov penalties, however. This is not surprising given that the additive model (3.18) and the multivariable Tikhonov penalty assume that the regression function can be expressed a sum of functions which are aligned along the predictor variable axes. Rotating the axes thus changes the functions which must be learned by the additive MLP, and it may be impossible to learn the same fit after rotation.

Rotation and weight decay

The weight decay penalty also is invariant to rigid rotations of the predictor variables about the origin. To see why, first note that to keep the same fit shape after the rotation, the hidden unit activations must have the same values which they had before rotation. This can be achieved by applying an equal counter-rotation to the input-to-hidden-weight vector for each hidden unit, to nullify the effect of the predictor variable rotation on the activations. Rotating the weight vectors does not change their Euclidean lengths, however, and so the value of the weight decay penalty is unchanged. Hence a rotated copy of the original fit is obtained when training directly with the rotated data.

3.5.2. Effects of translating the predictor and response data

This section considers how translating the data origin affects the fit when using roughness penalty or weight decay smoothing.

Translation and roughness penalties

All roughness penalties discussed in this chapter are inherently invariant to translations of the training data. This arises from the fact that they are based only on various derivatives of the fit, which do not depend on its position relative to any arbitrary origin.

Translation and weight decay

Ridge regression is not invariant to data translations[110-112], hence neither is MLP regression when weight decay is used. This can be concluded by noting that a MLP with linear hidden units instead of sigmoidal hidden units is equivalent to a linear regression model, and that training this MLP using weight decay is thus equivalent to linear ridge regression.

In terms of the weight decay penalty, the fit can be translated by varying only the bias weights. However, changing the values of these weights changes the weight decay penalty value, and this is why the fit is not invariant to arbitrary data translations.

3.5.3. Modifying weight decay for translation invariance

A simple modification can be made to the standard weight decay penalty to achieve translation invariance. The change is to remove all bias parameters from the weight decay penalty; that is, to use only the weight vector Euclidean length for the weight decay penalty. The method has also been suggested recently for MLP regression in[67] and for ridge regression[110].

Removing the bias parameters from the penalty gives translation invariance because:

- To shift the MLP output along the y-axis, only the output bias must be changed. Thus removing this parameter from the weight decay penalty removes the associated variation in the penalty value.
- The MLP fit can be shifted to any position in the $\underline{x}$-plane by changing the values of the hidden-unit bias parameters. Thus removing these parameters from the weight decay penalty removes the associated variation in the penalty value.

A further motivation for the modified weight decay penalty

A further reason for removing the bias parameters from the weight decay penalty is that there exist functionally equivalent MLPs (i.e. that give exactly the same fit) which have different bias vector Euclidean lengths but the same weight vector lengths[113, 114]. There is no obvious reason why a MLP with a long bias vector should be penalised more than a functionally equivalent MLP with a shorter bias vector. If two MLPs give the same fit, then the value of any penalty used to regulate the smoothness of that fit should also be identical for both MLPs. This is not true when standard weight decay is used, but is true for the modified version.

3.5.4. Comparison of the old and new weight decay penalties

Here is an illustration of how standard weight decay's lack of translation invariance can give undesirable fitting behaviour, and how the modified weight decay penalty gives better behaviour.

Description of the experiment

41 training data were created by sampling the function

$$\mu(x) = \frac{x}{2}\sin(3\pi x) \tag{3.23}$$

at intervals of 0.05 over the domain -1 to 1 inclusive. For clarity and simplicity, no response errors were added to the data.

To examine how translating this data along the x-axis could affect a MLP fit to this data when using weight decay, two further training sets were created by translating the data in the positive x-axis direction by 1.25 and 2.5 units respectively. The first of these translations is representative of the sort of differences in data position which may result from the various transformations that are often used to prevent saturation. The second is slightly more extreme, and was used simply to observe how sensitive standard weight decay can be to the data origin position.

MLPs with 10 hidden units were fitted to the three data sets using standard and modified weight decay with λ in the range 0 to 1. Figure 3.3 shows some typical results obtained using standard weight decay. Here the fits obtained with the shifted data have been shifted back along the x-axis so that they are superimposed over the fit to the unshifted data. The value of λ was chosen to give slight oversmoothing with the unshifted data. This was done to indicate that the chosen value gives roughly the amount of smoothing that would be useful in practice; the lack of response errors making overfitting unlikely even for $\lambda = 0$.

Since the training data are symmetric about their centre point on the x-axis, symmetric fits would be expected in all cases. However, in figure 3.3 only the fit to the unshifted data is symmetric. For the shifted data sets, the smoothing increases as x increases, giving oversmoothing at the right-hand side of the fit. This occurs because the hidden units which generate the fit in this region need larger bias values to translate their sigmoid transfer functions up the x-axis. However, the larger biases increase the value of the weight decay penalty, and so their input weights are reduced to compensate, causing the smoothing to increase with x.

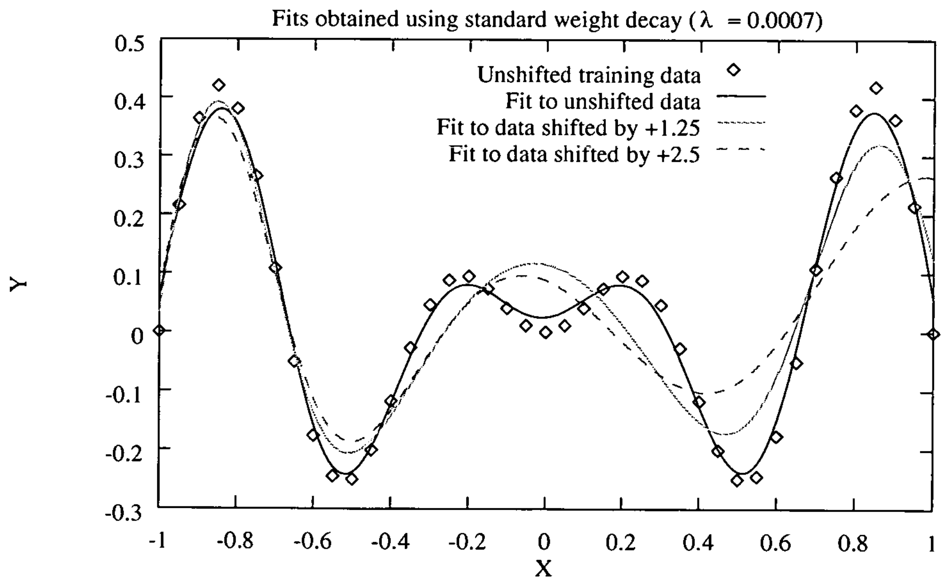


Figure 3.3: MLP fits obtained using the three training sets and standard weight decay with $\lambda = 0.0007$.

Figure 3.4 shows the fits obtained using the modified weight decay penalty. The three MLP fits are now identical when translated back along the x-axis. Thus the smoothing is now independent of the data origin, which is the type of behaviour that would be expected of a smoother.

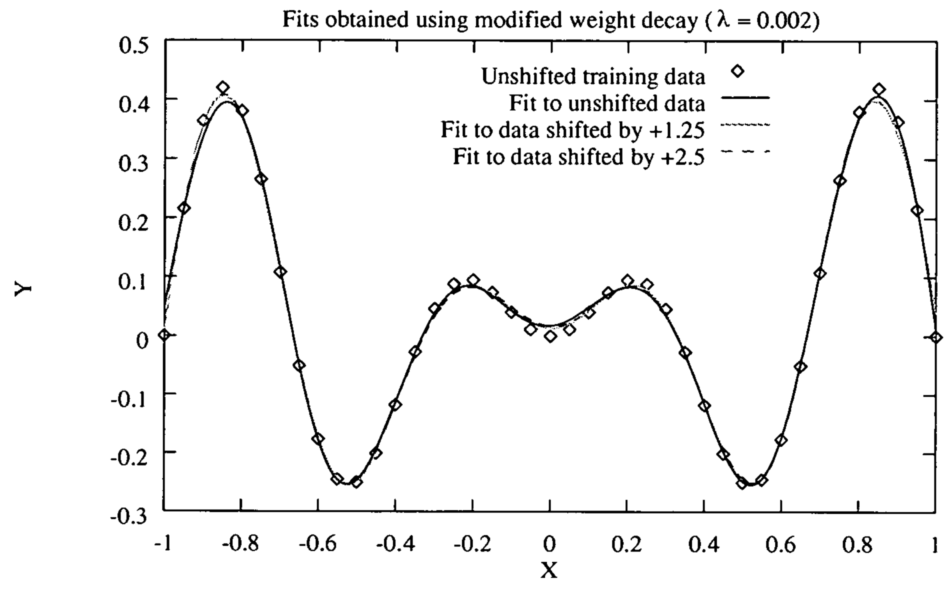


Figure 3.4: MLP fits obtained using the three training sets and modified weight decay with $\lambda = 0.0002$.

3.6. Chapter summary and conclusions

This chapter has considered regularisation methods because of their importance in aiding the use of a technique which is described in chapter six. The main issues examined were:

- use of MLP regularisation penalties based on spline roughness penalties,
- how scaling, rotating or translating the training data may affect the MLP fit when regularisation is used during training.

Roughness penalties

Roughness penalties are useful in low-dimensional problems and give a more direct, and hence intuitively appealing, approach to smoothing than weight decay. However, some practical issues must be addressed if roughness penalties are to gain wider use:

- How to choose a suitable smoothing interval without allowing overfitting or requiring very long computer run times.
- Whether to use analytic or finite difference derivatives for the roughness penalty.
- How to use roughness penalties for problems involving more than a few predictor variables.

The last of these issues is particularly important for the use of roughness penalties for the curl modelling problem. Since the available project time did not allow this issue to be examined in the depth necessary, roughness penalties were not used for developing the curl model. They are, however, used in some of the simulation studies presented in later chapters, where their computational demands could be kept under control.

It is hoped that sufficient basic issues have been uncovered, and possible avenues of further research indicated, to stimulate further work in the area of roughness penalties for MLP regression.

Regularisation and data transformation

When using smoothing penalties, the shape of the fit should not depend on whether the data are rotated or the data origin shifted. It was shown that fits obtained using standard weight decay do depend on where the data origin is located. It was shown that removing the bias terms from the weight decay penalty removed the dependence of the fit shape of the data origin position. A simple example showed how this translation invariance gives a more intuitively appealing smoothing behaviour.

Relevance to curl modelling problem

In terms of the curl modelling problem, the overall conclusion of the work presented here is that only the standard and modified weight decay could be used for smoothing the curl fit. Roughness penalties cannot be used until the issues concerning how to apply these penalties to high-dimensional problems are addressed.

Chapter 4

Background to the project modelling problem

4.1. Introduction

This chapter describes the curl modelling problem which provided the original motivation for this project. In particular, it discusses the characteristics of the data which were provided for this project in more detail than was given in chapter one, and describes the deficiencies of this data which led to the investigations which are presented in the next chapters.

4.2. Coated paper production

Tullis Russell is a Scottish papermaking company which manufactures a wide range of coated papers in weights ranging from light cards to heavy boards. This type of paper is most familiar as the smooth, glossy paper commonly used to make calendars, cards, posters and, to cite an example specific to Tullis Russell, the cover of the UK Yellow Pages telephone directory. Figure 4.1 overleaf shows a block diagram of the coating process which Tullis Russell use to produce much of their coated paper.

In this process, each side of the paper is coated by applying a surface layer of a coating mix which is then dried and smoothed. The mix itself is a thick slurry usually comprised of chalk, latex, binders and possibly colourings; and the exact composition can be varied to control qualities of the finished paper such as its grammage, smoothness and gloss. Once the coatings have been dried and polished, the paper is cooled and then reeled up for storage.

4.3. Curl

One characteristic that the coated paper may exhibit is a tendency for sheets of the paper to form curved surfaces rather than lying naturally flat, as illustrated in figure 4.2. This deformation is known as *curl*, and is caused by differences in the contractions of each side of the sheet as the coatings and base paper dry[115].

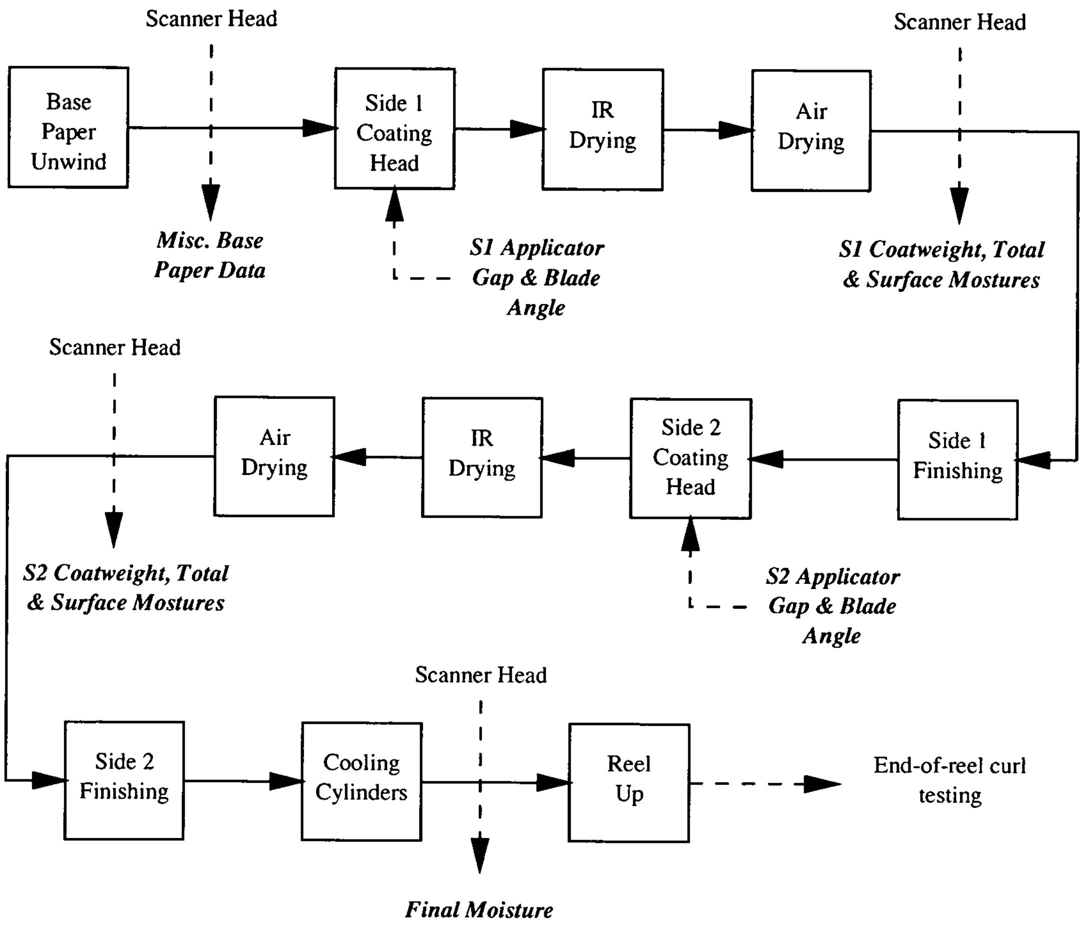


Figure 4.1: Block diagram of the Tullis Russell Truflo coater. Some possible predictor variables for a curl model are shown in bold italics.

(a) Curl-free paper lies naturally flat.

(b) Paper exhbiting curl forms a curved surface and will lie flat only when restrained.

Figure 4.2: A very simple illustration of curl. More complex types of curl, such as twisting, are also possible[116].

Curl is an undesirable paper quality for many reasons, some of which are:

- It reduces the aesthetic appeal of the paper for use in products such as brochures and magazine covers.
- It is the commonest cause of sheet feeding problems in non-impact printing processes such as photocopying and laser printing[117].

Consequently, paper exhibiting high levels of curl has no market value and must be scrapped. Since scrapping the paper results in the loss of its manufacturing costs such as the processing time and energy, it is clearly desirable to control the coating process to prevent the production of paper with unacceptably large curls.

4.3.1. Measurement and control of curl

One obvious approach to keeping curl within acceptable limits during manufacture would be to use a closed-loop control system where

- curl is measured continuously as the coated paper is produced, and
- the machine settings are promptly adjusted in an attempt to minimise curl while maintaining other desired paper quantities such as grammage and gloss.

However, standard curl measurement techniques cannot be used to measure curl continuously during manufacture[115]. These methods require a paper sample to be dried under controlled conditions to allow any inherent curl to develop, and this prohibits on line curl measurement since:

- Removing test samples will create unwanted holes in the final product, and puncturing the moving web (paper sheet) may cause it to break. If the web breaks then production will be halted while the broken material is removed and the coating machine is restarted.
- Even if a sample could be taken, the drying time causes a significant delay before the curl can be measured. During this time a large amount of paper may be coated, and this must be scrapped if the test sample curl is unacceptably high.

4.3.2. Curl measurement and control at Truflo

Since it is not possible to measure curl during manufacture, Tullis Russell currently assess curl after manufacture by taking only one test sample from the end of each coated reel. If the curl is too high then the entire reel is scrapped, and various heuristic rules are applied to establish which machine settings should be adjusted to reduce the curl for the next reel. For example, if the paper curls toward side one, then the side two surface moisture or coatweight may be increased in an attempt to reduce this curl.

Though experienced machine crews can apply these rules with some success, there are some obvious problems with this method:

- Since each curl measurement can be used to test only one change in the machine settings, several reels may be wasted until curl is reduced to acceptable levels.
- Effective use of the heuristic rules is clearly dependent on the machine crews' skill and experience.

- If production is to be switched to a different grade of paper, such as a heavier board, then the heuristic control process must be restarted.

Due to the large scale of production involved, the cost of the reels wasted while trying to bring curl under control can be very significant. Consequently, Tullis Russell are interested in investigating methods for assessing curl which can be used on line and by inexperienced operators.

4.3.3. Assessing curl on line using inferential estimation

One method which may be applied to the problem of assessing curl on line is *inferential estimation*[8]. Here, it is assumed that the measured curl can be related to the values of other quantities which can be measured on line, such as the paper moistures and machine settings. If a model of this relationship can be obtained, then this model can be used to estimate curl on line from these other quantities.

Some semi-empirical curl models have been reported previously[118], but applying these directly to the Tullis Russell process is very difficult because they are quite specific to their own processes. Thus it was decided to investigate whether a MLP could be used to model empirically the relationship between curl and other process variables for the Tullis Russell coating process. The MLP was chosen for this task because it was suspected that a nonlinear model would be required, and this is confirmed later in chapter seven.

Obviously, other methods such as kernel regression could also be applied to this problem, and the MLP was chosen simply because the research group knew very little about these methods when the project was proposed. As more was learned about these methods during the project, no compelling reasons to use them in preference to the MLP could be found, and there was insufficient time to try them in addition to the MLP. A parametric model was also considered desirable since standard function optimisation techniques could then be used to find machine settings which minimise the predicted curl.

4.4. Data problems and modelling issues

At the beginning of this project, it was assumed that standard training methods such as LS backpropagation could be used to train the MLP curl model. However, when the first set of process data became available, it was found that most of this data could not in fact be used with these training methods due to missing entries in many of the reel logs.

One obvious solution to this problem is to simply discard those records with missing entries, and to train the MLP with the remaining data. However, since the data set contained a very limited number of reel logs (318) and a large number of candidate predictor variables (29), it was desirable to retain as much data as possible to minimise dimensionality related problems. Unfortunately, once the incomplete cases were discarded, typically only 100 to 125 complete cases remained; the exact number depending on the choice of predictor variables.

Linear regression analysis of this data indicated that curl did appear to be related to some of the variables, but there was insufficient data to tell whether this relationship was significantly nonlinear.

The problem of missing data has, however, been studied quite extensively in the statistical literature[105, 119-121]. Consequently, it was decided to investigate whether these methods could be used to fill-in, or *impute*, the missing data entries and thus increase the amount of usable data.

Since many of these early investigations were rather naive, mostly due to initial unfamiliarity with MLP and regression methods, the details of these will not be discussed here. However, one question which arose when using unconditional mean imputation[120] to estimate the values of the missing predictors was whether or not this simple method was introducing many incorrect values into the data, and whether these could adversely affect the quality of the MLP fit.

Again, an answer to this question was found in the statistical literature, namely that least squares parameter estimates may in fact be very sensitive to such gross errors in the data[122]. Indeed, Tresp *et al.* have recently demonstrated a problem where the use of simple mean imputation led to poorer generalisation than when the missing data was discarded; and this was due to the number of grossly incorrect values introduced into the data[123]. However, the most disturbing aspect of the discovery that least squares estimates may be sensitive to outliers was that these may already be present in the complete data, because of operator data logging errors for example. It should be noted that the data with the largest response errors need not have the largest or lowest response values, and so discarding these data may not remove any or all gross errors which may present.

Thus the problems of training with missing data entries and possible gross errors in the data were identified as important to the curl modelling problem. Since these are general issues of data deficiency rather than being specific to this particular problem, it was thus decided that they should be investigated further in the context of MLP regression.

4.5. The second data set and project directions

At this time a second data set became available, and after an initial examination of this data it was decided not to pursue the problem of missing data further. The reason for this was that though this data contained only 504 reel logs, there were fewer missing entries than in the first set, and upwards of 356 complete records could be recovered.

Though this is obviously still a rather limited number of data, linear regression analysis of the complete data indicated that curl was nonlinearly related to some of the predictor variables. The results of this analysis will be discussed in more detail in chapter seven, but from these it was estimated that there was sufficient complete data to merit attempting to model this relationship using the MLP.

It should also be noted here that the complete data from the first and second data sets were not combined to form a larger data set. The reasons for this were:

- The data in the first set was suspected to be of a low quality due to variations between machine operators when measuring variables such as the paper curl. Many of the measurement procedures were improved to reduce such variations before the second data set was collected.
- The second set contained four more predictor variables which were considered pertinent to the problem of modelling curl.
- The first data set contained data from many paper grades and provided very little data for each grade. The second set, however, concentrated on a few of the more important lighter grades, which also exhibit the highest curl.

Thus though the first data set was eventually discarded because of the problems which it posed, it did bring several practical training issues to my attention which I felt should be investigated before attempting to use the second data set. Hence chapters five and six discuss the findings of these investigations before returning to the issue of developing the curl model in chapter seven.

Chapter 5

Outliers and robust MLP regression

5.1. Introduction

So far, this thesis has considered only LS training (with optional regularisation) for MLPs. The reason for this is that LS is by far the most widely used estimation method in classical and MLP regression. However, a problem with LS estimation is that the fit can be very sensitive to one or more bad outliers in the data. This can result in a few observations pulling the fit away from the fit suggested by the majority of the data. Concern was thus raised about using LS training for the curl modelling problem because it was suspected that the Tullis Russell data may contain several gross errors.

It is very difficult to identify and remove outliers by examining scatterplots of multidimensional data. A much more effective method for tackling problems where outliers are expected is to use *robust* estimators, which are much less outlier-sensitive than LS. Though these are becoming increasingly available in statistical software packages as their importance and benefits become more appreciated, few ANN researchers have considered the issue of outliers in MLP regression or the application of robust estimators to this problem. Those who have considered this problem often do not provide many guidelines to aid others in the practical use of these estimators in MLP regression.

It was thus decided to investigate some of the important issues in choosing and using these estimators for MLP regression. While the main aim of this work was to determine which robust estimators would be most suitable for the curl modelling problem, the issues examined are quite general, and the work presented in this chapter can be applied to any MLP regression problem where outliers are expected.

This chapter is structured as follows:

- Sections 3 and 4 introduce important basic robust estimation concepts which are used throughout the chapter.
- Section 5 reviews the robust estimators considered in this project. Practical issues pertinent to their implementation and ease of use for MLP regression are also considered here.

- Section 6 introduces the two issues in robust MLP regression which were examined, namely efficiency and the use of high-breakdown estimators, and explains why these are important.
- Section 7 examines the efficiency of different estimators for MLP regression. It is shown that simple robust estimators are likely to give the best fits in MLP regression, and that most sophisticated estimators may actually give poorer estimates of the regression function. This section also discusses how under- and overfitting affect robust training methods, an issue which has caused some confusion in the robust training literature.
- Sections 8 and 9 examine the effects of high leverage outliers and the role of high-breakdown estimators in MLP regression. It is shown that these estimators are not suitable for MLP regression, despite contrary claims in the literature.

5.2. The popularity and properties of least squares estimation

LS estimation is used very widely in regression because LS estimates are often amenable to relatively easy theoretical analysis and exhibit some important optimal properties[124]:

- For models which are linear in their parameters, closed-form expressions can be derived for the LS parameter estimates. The estimates can thus be computed directly without needing to use computer-intensive iterative optimisation methods, such as those required to train MLPs.
- The ability to solve linear LS problems directly simplifies the development of inferential methods for linear LS problems and also nonlinear problems where linear approximations are possible[32, 125-127].
- When the errors are independent Gaussian deviates and the model is unbiased, the LS parameter estimates are then ML estimates, which have various optimal properties (see appendix A).
- The Gauss-Markov theorem shows that the LS parameter estimates have the lowest variance among all possible linear estimators irrespective of the actual error distribution[124, 128, 129].

In MLP regression, relatively little attention has been given to the statistical properties of LS training methods until recently, when interest has focussed on the similarities between ANN and statistical methods. Here, the popularity of LS estimation appears to be due simply to its use in the earliest derivations of the backpropagation training method equations[57], and to a reputation for being a useful general fitting method. This lack of interest in the statistical properties of LS estimation and alternative estimators is probably the main reason why the effects of outliers have not received much attention in MLP regression.

5.3. Outliers and LS estimation

What are outliers?

In regression, outliers are data which have unusually large response errors and hence lie far from the majority of the data and the regression function. Their effect in LS regression is to pull the fit away from the trends suggested by the majority of the data, as figure 5.1 illustrates using a linear regression example.

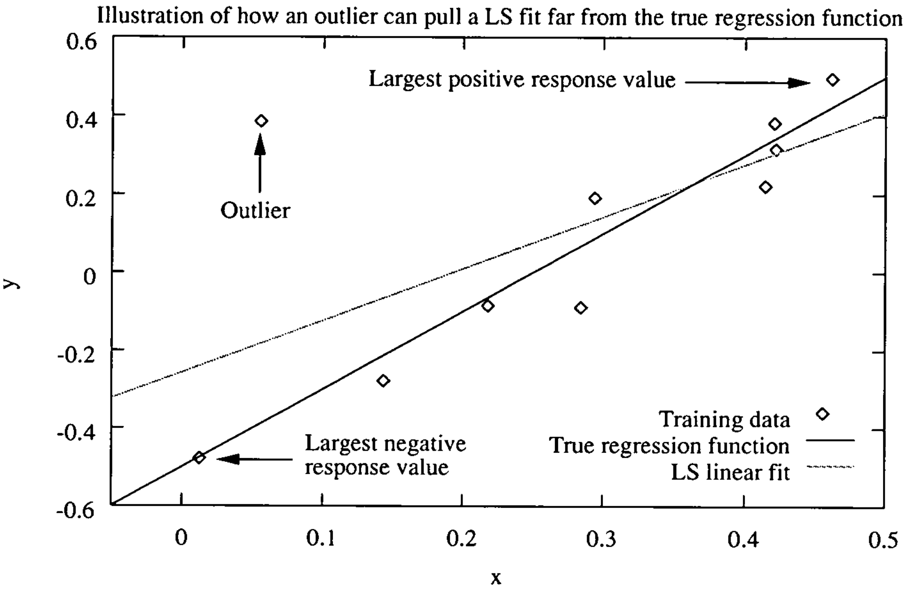


Figure 5.1: Illustration of a LS regression fit pulled away (upwards) from the general data trend by a single outlier. Outlier identification is easy in this example (by inspection), but can be a very difficult problem when there are many predictor variables[130].

How do outliers arise?

Outliers occur more commonly than is usually appreciated, even in supposedly high-quality data sets[129]. The two mechanisms which usually give rise to outliers are[131, 132],

- the error probability density function (pdf) is naturally long-tailed, making extreme data more likely than would be expected under, say, a Gaussian error distribution, and
- gross errors may be introduced while measuring or logging the data. This can be considered a special case of a long-tailed error distribution where the gross errors increase the tail lengths of a shorter-tailed natural error distribution.

Outliers generated by the second mechanism are clearly bad data, while those generated by the first are in fact genuine data. However, irrespective of which mechanism generates the outliers, LS estimates are sensitive to these distant data.

How can outliers be handled?

One obvious approach to this problem is to inspect the data and to remove suspected outliers before applying LS. However, this procedure has two serious shortcomings,

- it is very tedious for large data sets, and
- it can be very difficult to identify outliers in multidimensional data[41, 129]. As figure 5.1 shows, the data with the largest response values are not necessarily outliers in regression problems[130].

A different and generally superior approach to the outlier problem is to fit the model using a *robust* estimator[129, 133-137]. Two advantages of this approach are:

- Robust estimators are less sensitive to outliers and thus require little or no manual effort to identify and remove outliers. This can save the data analyst considerable time when trying to suppress the effects of any outliers.
- Some robust estimators can yield better estimates than applying LS to a censored data set. This is because they are more *efficient* estimators.

5.4. What are robust and resistant estimators?

Two of the most basic and important concepts in robust estimation are *resistance* and statistical *robustness*, particularly robustness of efficiency. Understanding these concepts is necessary to understand what most robust estimators aim to achieve, and hence to understand the aims of most of the work described in this chapter.

5.4.1. Resistant estimators and estimator breakdown

An estimator is said to be resistant if its value does not change by an arbitrarily large amount when one (or more) of the data to which the estimator is applied changes by an arbitrarily large amount.

To see why resistance is important when considering the effect of outliers on an estimator's value, consider the sample mean, which is the LS estimator (see[102] or appendix A),

$$\bar{y} = \hat{\theta} = \underset{\theta}{argmin} \sum_{i=1}^{n} |y_i - \theta|^2 = \frac{1}{n}\sum_{i=1}^{n} y_i \ . \tag{5.24}$$

This estimator is not resistant, as seen by noting that making any one of the y_i arbitrarily large causes the sample mean also to become arbitrarily large. Thus a single large outlier can pull the sample mean far from the value it would have if the outlier was not present.

In contrast, the sample median, which is the least absolute deviations (LAD) estimator

$$\tilde{y} = \hat{\theta} = \underset{\theta}{argmin} \sum_{i=1}^{n} |y_i - \theta| \ , \tag{5.25}$$

is resistant. This is because the value of the median depends on only the central values of the numerically ranked data, and so does not change if the most extreme data move further away from the median. At least half of the data (either all the data above or below the median) must change to affect the value of the median, and so the median is said to have a large-sample breakdown-point of 50%.

Resistance is fundamentally important when considering how outliers may affect a given estimator. If an estimator is not resistant, then it is likely that a large outlier will shift its value far from the true value of its estimand. LS estimation is not resistant, due essentially to the fact that squaring the fit errors increases the relative contribution of the largest errors to the SSE, and so the fit is pulled towards the outliers when minimising the SSE during fitting.

The influence function of an estimator

An important tool for assessing an estimator's resistance is its influence function (IF). This describes how much its value changes by when the data to which it is applied are perturbed, by creating outliers for example.

Understanding the IF in MLP regression aids understanding of how high leverage outliers affect MLP fits (examined in section 9) and also how sensitive the fit is to perturbations of the data (important in chapter six). However, since most of the work in this chapter can be understood and used without detailed understanding of the IF in MLP regression, this material is discussed in appendix B.

5.4.2. Statistical robustness and robustness of efficiency

Knowing that two estimators are resistant does not tell whether one of them may be more sensitive to certain data perturbations; for example, its value may undergo a large (but bounded) change for only a small perturbation of the data. The concept of how sensitive an estimator is to such changes is encapsulated in the idea of statistical robustness.

Statistical robustness

An estimator is robust if its useful properties, such as low bias or variance, are not sensitive to violations of any assumptions underlying its use[133, 138]. For example, LS estimation often has optimal properties, such as low estimate variance, when applied to Gaussian data, but is not *distributionally robust* because these properties diminish quickly if the data distribution tail lengths are increased slightly.

Since outliers increase distribution tail lengths, such distributional robustness is the type of robustness usually sought in robust estimation. The estimator property of most interest is usually efficiency, which is now described.

Estimator sampling variance and efficiency

Efficiency is linked strongly to the idea of estimator variance. The concept of fit variance in MLP regression was introduced in chapter two, where it was stated that overfitting causes the fit to be sensitive to small perturbation of the data.

However, the variance of a MLP fit depends not only on how complex the MLP is, but also on the estimator used to train the MLP. To illustrate how variance depends on the estimator, and to introduce the idea of efficiency, consider the sample mean and median again. Since overfitting is not possible with these estimators, their variance is solely a property of the estimator.

Since the value of the sample mean, sample median, or indeed any estimator, depends upon the particular random data set to which it is applied, this value is itself a random variable. Thus if a given estimator is computed many times using different data sets drawn from the population of interest, then a collection of random estimator values will be obtained. Ideally these should all be close to the true value of the parameter being estimated, and their limiting distribution as the number of estimates tends to infinity is known as the *sampling distribution* of the estimator[139-142]. This concept is illustrated in figure 5.2, which shows estimates of the sampling distributions of the sample mean and the sample median, computed using 500 million data sets each comprised of 30 observations from the Gaussian distribution N(0,1).

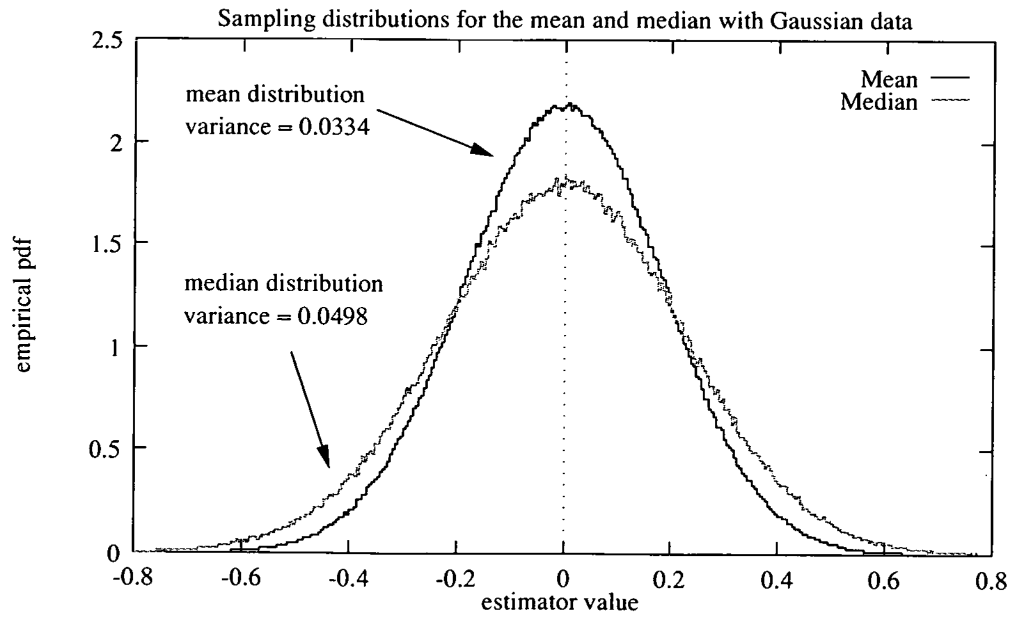


Figure 5.2: Examples of sampling distributions for the mean and median.

The most significant feature of figure 5.2 is that the sampling distribution of the mean has a smaller variance than that of the median. Consequently, the sample mean will generally (but not always) be closer to the unknown population mean than the sample median will be, and so is the preferred estimator for estimating the mean of Gaussian data.

Estimator efficiency and robustness of efficiency

Low estimation variance is desirable because low-variance estimators will usually give the most accurate estimate of the quantity being estimated. Estimators are thus often compared in terms of their relative efficiency[139, 142],

$$relative\ efficiency = \frac{var(T_1(F))}{var(T_2(F))} \tag{5.26}$$

where estimator T_2 has the higher sampling variance[10]. Efficiency depends on the distribution of the data to which the estimators are applied, and it is assumed that appropriate estimators are being compared. One reason why ML estimation is so important is that it often gives the most efficient estimator possible for the specific problem where ML estimation occurs[142]. For example, the sample mean is the ML estimator of the mean for Gaussian data, which explains why figure 5.2 shows the mean to have a lower sampling variance than the median.

The variances shown on figure 5.1 show the median to be only 67% as efficient as the mean. However, the median becomes the more efficient estimator when the data are drawn from a long-tailed distribution, such as the Laplace or Cauchy distributions. In the context of robust estimation, it is desirable to find and use estimators which are efficient both for Gaussian data and for data from long-tailed distributions which are likely to be representative of those caused by outlier contamination. This ensures that the estimator performs well for 'well-behaved' data, but that unexpected increases in the tail-lengths due to outliers do not compromise the efficiency badly[129, 144]. Such estimators are said to show good robustness of efficiency.

5.5. Robust estimators and regression

Having introduced the basic concepts of resistance and robustness of efficiency, it is now possible to discuss the simple robust estimators examined during this project and their relative merits.

[10] For biased estimators, the MSE is usually used instead of the variance[140, 143]. Sampling variances are often computed using Monte Carlo methods when they cannot be computed using theory.

5.5.1. M-estimators

M-estimators are arguably the most widely used robust estimators. They were first investigated by Huber as a generalisation of ML estimation, from which they derive their name[137, 144]. Most M-estimators attempt to maintain the highest possible minimum efficiency over a range of data distributions, usually including long-tailed distributions. This minimax approach attempts to provide the best possible efficiency under the assumption of a worst-case error distribution.

In regression, M-estimates of the true parameter values are given by

$$\hat{\underline{\theta}} = \underset{\underline{\theta}}{argmin} \sum_{i=1}^{n} \rho\left(\frac{y_i - f(\underline{x}_i; \underline{\theta})}{\hat{s}}\right) \tag{5.27}$$

where the function $\rho(\varepsilon)$ defines the estimator, and the purpose of the auxiliary scale estimator, $\hat{s}$, will be explained shortly.

The simplest M-estimators considered in this thesis are L_p-Norm estimators[124, 145]. These are obtained by minimising

$$\sum_{i=1}^{n} | y_i - f(\underline{x}_i; \underline{\theta}) |^p \tag{5.28}$$

and require no scale estimate. This estimator family includes the LS ($p = 2$) and LAD ($p = 1$) estimators as special cases.

For M-estimators, it can be shown (see appendix B) that the IF is proportional to the estimator *score function*,

$$\psi(\varepsilon) = \frac{d\rho(\varepsilon)}{d\varepsilon} . \tag{5.29}$$

Since the IF must be bounded for resistance, this implies that (5.29) must also be bounded for resistant estimators. L_p-Norm estimators are thus not resistant when $p > 1$ because the score function always increases as ε increases. However, for p in the range 1.2 to 1.5, their efficiency may nevertheless be good for both Gaussian data and long-tailed distributions likely to be encountered in practice[124, 145-148]. For this reason, and also because they are among the simplest M-estimators to compute, these estimators are used quite widely despite their non-resistance.

In his original theses on M-estimators, Huber proposed the estimator defined by

$$\rho(\varepsilon) = \begin{cases} \dfrac{\varepsilon^2}{2} & |\varepsilon| \le k \\ k|\varepsilon| + \dfrac{k^2}{2} & |\varepsilon| > k \end{cases} \tag{5.30}$$

for k typically in the range 1 to 2[133, 137, 144]. For simple location estimation, Huber's estimator is resistant and has good efficiency over the family of contaminated normal

(CN) distributions, which give a good approximation of Gaussian data contaminated by some gross errors[122]. The CN distribution with 100α percent contamination at scale factor β has the cumulative distribution function (CDF)

$$(1-\alpha)\Phi(\varepsilon)+\alpha\Phi\left(\frac{\varepsilon}{\beta}\right) , \qquad (5.31)$$

which is a combination of a predominant Gaussian CDF ($\Phi(\varepsilon)$) with a fraction of a higher variance ($\beta > 1$) Gaussian to increase the tail lengths.

Scale equivariance

Unfortunately, most M-estimators (but not L_p-Norm estimators) are not scale equivariant because

$$\rho(k\varepsilon) \neq \rho(k)\rho(\varepsilon) \ . \qquad (5.32)$$

This is undesirable because it causes the estimates to depend on any arbitrary scaling of the data.

For example, consider fitting a MLP using Huber's estimator. If the response data are scaled uniformly to small values, so that all fit errors lie in the quadratic section of $\rho(\varepsilon)$, then LS fitting results. Conversely, if the data are scaled up so that all fit errors fall on the linear region, then LAD fitting results. Thus simple, arbitrary data scaling leads to different types of estimation.

To overcome this problem, an estimate of the error scale, $\hat{s}$, is introduced into the estimator definition (5.27) to compensate for any data scaling[129]. Various robust scale estimates[149, 150] can be used, and for this work the median absolute deviation from the median (MADEV),

$$MADEV(\varepsilon_1,\cdots,\varepsilon_n) = \frac{median\{\ |\varepsilon_i - median\{\varepsilon_i\}|\ \}}{0.6745} , \qquad (5.33)$$

was used. It is not necessary here to know the properties of this estimator, other than that it is very resistant to outliers.

Practical considerations when using scale estimates in MLP regression

M-estimation is complicated considerably by introducing the auxiliary scale estimate, because computing each of $\underline{\hat{\theta}}$ and $\hat{s}$ requires the value of the other to be known. Some solutions to this problem are proposed in[144, 151] and the method used here was to first compute $\hat{s}$ using the MLP random starting parameter values, and then use this $\hat{s}$ to improve the $\underline{\hat{\theta}}$ estimate by applying a few MLP training steps. Once new parameter estimates are obtained, $\hat{s}$ is computed again using the new parameters, and the process is repeated until $\underline{\hat{\theta}}$ and $\hat{s}$ converge.

In the work reported here, $\hat{s}$ was updated every time the search direction of the conjugate gradient method used to train the MLPs was reset. Updating $\hat{s}$ at other times is clearly possible, but was not investigated because it was felt that this would interfere with the construction of the conjugate search directions.

An important practical point was discovered when using Huber's estimator[11]. The auxiliary scale estimate is usually very large at the start of training, because the MLP does not fit the data well and so the fit errors are large. This causes many of the data, some of which may be outliers, to lie on the quadratic section of the Huber error function, and this can cause fast overfitting to the outliers. Once this has occurred, further training cannot correct it. The solution adopted here was to reduce the size of the quadratic section by multiplying the scale estimate by 0.01 for the first 20 scale estimate updates, though different problems will require different degrees of extra scaling. This approach effectively causes LAD estimation to be used at the start of training. An alternative approach that does not require empirical tweaking of the scale estimate simply uses the LAD fit as a starting point for Huber's method.

The main point however is that to avoid rapid overfitting, it is advisable to avoid any LS-type fitting until the main features of the regression function have been fitted. Overfitting is shown later to cause loss of estimator efficiency.

Other M-estimators

Many more M-estimators have been described in the literature[102, 129, 152], most of which are *redescending* estimators. These have IFs which return to zero for fit errors larger than a constant known as the rejection point, and so give no influence at all to data with large fit errors. While these can provide good efficiency over a range of long-tailed distributions, they were not considered here for two reasons:

- The LAD estimate (or a similar simple robust fit) is often required as the starting point for these estimators. This is because good data can lie in the rejection region at the start of training, and may never be fitted because they have no influence during training[129, 153]. Requiring a LAD starting point also complicates the issue of implementing stopped training, because this raises the question of when the initial LAD training should be stopped.
- It was suspected that redescending estimators may not give better fits than simpler robust estimators in many cases. This is discussed later, when the efficiencies of some M-estimators in MLP regression are compared.

[11] It is expected that this result applies to all M-estimators whose error functions are quadratic for small to medium-sized errors, though no other estimators of this type where tested here.

Training MLPs with redescending estimators is possible however; some simple examples are provided in[154].

5.5.2. L-estimators

L-estimates are linear (L-) combinations of order statistics. That is, they are obtained by ranking the data (or some function of the data) into numerical order and then averaging part of this sequence. Since averaging is a LS estimator (see appendix A), this corresponds to trimming-away some of the data and then applying LS estimation to the data which remain. In L-estimation, the trimming is performed with the intent of removing outliers from the data.

L-estimation is often not as efficient as M-estimation because trimming data removes information[132]. However, some L-estimators have other advantages in linear regression which are discussed later.

L-estimation for regression

The definition of a regression L-estimator is confused somewhat by the fact that there are two widely used but different definitions. Conditional-quantile type estimators are obtained by averaging fits obtained using various LAD-like estimators[136, 151, 155]. They were not examined during this project, but it is suspected that their major properties will be similar to those of the LAD estimator, which was examined.

The L-estimators considered here are of the data-trimming type[130, 151, 155].

Least-trimmed-squares regression

Least-trimmed-squares (LTS) estimates are obtained by discarding the data which give the largest fit errors and then applying LS to the remaining data[155]. This clearly requires some care to avoid trimming too many or too few data[132, 148, 156], though this issue is not considered in depth here.

Note that LTS estimation is not the same as discarding the data with the largest response values. At each training step, LTS trims only the data which lie furthest from the fit and, as shown earlier, these need not be the data with extreme response values.

Least-median-of-squares estimation for MLP regression

Least-median-of-squares (LMS[12]) estimation[130] is a close relative of LTS where the parameters estimates are chosen to minimise the median squared error (MedSE),

$$MedSE = median\left\{ (y_i - f(\underline{x}_i; \underline{\theta}))^2 \right\} . \qquad (5.34)$$

LMS estimation was devised to tackle the problem of multiple high-leverage data in linear regression, an issue which is examined later in the context of MLP regression.

Practical implementation of LTS training

Training MLPs using LTS presents some difficulties because

- The LTS error surface has valleys with sharp (V-shaped) floors where the data being trimmed changes. Fast line-searching optimisation methods become trapped in valleys with V-shaped floors. Unless the search direction points exactly along the valley floor, the search gradient points predominantly towards the opposite valley wall, causing successive searches to oscillate between the valley walls while making little progress down the valley floor.
- Minimising the LTS error function using line-searching requires any necessary changes in which data are selected for trimming to be detected during the line search. The extra sorting and searching required to implement this increases the training time dramatically.

Using backpropagation-type gradient descent avoids these problems because it does not use line-searching. However, a small learning rate is then required to keep the method stable, and this was found to result in very long training times even for simple regression problems.

It was thus decided to investigate techniques for using the faster line-searching training methods. By changing which data are trimmed only after each line-search completes, sharp valley floors cannot be encountered, though the true LTS error function is no-longer being minimised. To check whether this approach would eventually lead to the true LTS estimates, downhill simplex optimisation[55, 62] was also used for training. The similar results obtained using both methods when the trim was small (less than 15% or so of the training data) suggested that this line-searching approach was suitable for training MLPs using LTS estimation. It is shown later that large trims are not useful for training MLPs, and so the training method performance with large trims is not important.

5.6. Investigations into robust training of MLPs

Existing works on using robust methods for training MLPs[21, 154, 156-159] focus almost exclusively on resistance and demonstrating that gross outliers can be resisted by using a robust training method. While this is useful in bringing the issue of outliers to the attention of other users of MLP regression, often little practical guidance is given

[12] Cautionary note: This should not be confused with the use of LS estimation in control theory and signal processing, where it is often called least mean squares (LMS) estimation.

concerning issues such as choosing what estimator to use. Indeed, it is shown here that some robust estimators which have been suggested for training MLPs should not be used at all for this purpose.

The work performed here extends existing robust MLP regression work by examining two important issues in particular:

- What affects the efficiency of different estimators in MLP regression?
- Many linear regression problems require special estimators known as high-breakdown estimators. Are these estimators also necessary, or even useful, for MLP regression?

Estimator efficiency in MLP regression

Examining efficiency is important for the same reason that it is important in robust linear regression, namely to aid the choice of a suitable estimator for a given problem. The use of quite sophisticated estimators, such as re-descending M-estimators, is often suggested for MLP regression[21, 154]. These are considerably more difficult to use than simpler estimators, such as L_p-Norm estimators. They also require more computation, resulting in longer training times. If little improvement in efficiency can be expected in practice when using these estimators, then simpler estimators should be used.

Studying efficiency in MLP regression also gave important insight into how the fit complexity and complexity control used during training affect robustness. This has been a source of some confusion and misadvice in some previous work, where it has been thought that using only complexity control methods, such as weight decay, is sufficient to control the affects of outliers[159]. This work shows that complexity control is important, but that robust methods are still essential when the data contains outliers.

Using high-breakdown estimators for MLP regression

High-breakdown estimators, such as the LMS estimator, are particularly important for linear regression problems where the data may contain *high-leverage* outliers[130]. These can strongly influence the fit even when using some simple robust estimators such as L_p-Norm estimators and Huber's estimator. This work examined whether these outliers posed as serious a problem in MLP regression as they do in linear regression, and whether high breakdown estimators are also necessary in MLP regression.

5.7. Estimator efficiency in MLP regression

This work examined the factors that affect efficiency in MLP regression.

An efficiency definition for MLP regression

The definition of efficiency used commonly in linear and polynomial regression is not well-suited to describing efficiency in MLP regression. Thus the first issue to be considered was how to define efficiency in MLP regression.

Efficiency in linear regression is usually defined on a per-parameter basis as the ratio of the variances or MSEs of the parameter estimates (see[147] for example). Two reasons for using this definition are:

- the parameter values are the estimands of the fitting process and so any theory relevant to the estimation procedure used (e.g the asymptotic normality of MLEs) applies directly to them, and
- sometimes the parameter values are of more interest than the fit itself. For example, knowing the sign of a parameter may be very important.

In MLP regression the parameter estimates are usually of little interest; instead it is almost always the fit that is of interest. It is thus more appropriate to use an efficiency definition based on the sampling variability of the fit instead of the variability of the parameter estimates. Figure 5.3 illustrates what is meant by the sampling variability of the fit. This figure shows 30 estimates of a sine-wave function obtained by applying LS and LAD MLP regression to 30 different training sets with contaminated normal errors. The LAD fits have the lowest variability in the sense that they cluster most tightly around the regression function. This means that LAD fitting will usually give a more accurate fit for this regression problem, and so is the more efficient estimator.

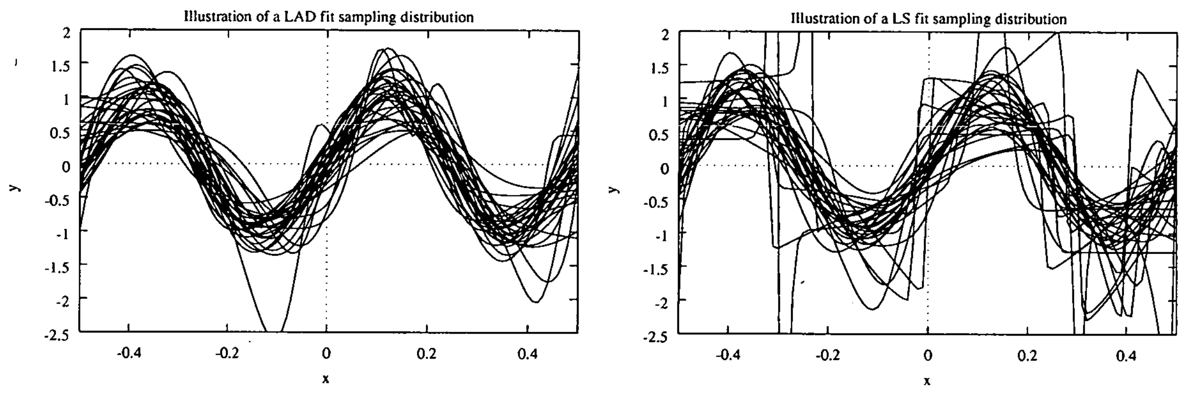


Figure 5.3: Estimates of fit sampling distributions for a simple regression problem. The graph on the left shows 30 fits obtained using LAD fitting, the graph on the right shows 30 fits obtained using LS fitting. The same 30 training sets were used in both cases.

In this work, combined fit bias and variability were measured using the mean square fit error over the domain of interest for the regression fit. The relative efficiency of different estimators is then

$$efficiency = \frac{MSE(\mathit{fit\ obtained\ using\ estimator\ A})}{MSE(\mathit{fit\ obtained\ using\ estimator\ B})}\ .$$

where the MSE is the expected fit MSE over all possible training data sets. Other fit error measures, such as the mean absolute error, could also be used. However, using the mean absolute and median square errors gave little qualitative difference in the results and conclusions of the efficiency experiments, and so only the MSE is used here.

A further motivation for using the fit variability to assess efficiency is that this definition applies directly to nonparametric regression. Thus direct comparison of MLP and nonparametric methods is possible, though not reported here.

Efficiency study overview and investigative method

Five estimators were compared in the efficiency study: LAD, $L_{1.2}$, $L_{1.5}$, LS and Huber's estimator with $k = 1.5$. The LTS and LMS estimators were not included because a separate study, discussed later, found these to be inappropriate for MLP regression. Since it is not possible to derive the expected fit MSE for most MLP regression problems, Monte Carlo simulation was used to estimate the MSEs, and hence relative efficiencies.

Each estimator was used to estimate some 2- and 3-dimensional regression functions. Using low-dimensional functions allowed the fits to be visualised to aid diagnosis of any strange results, and to allow the simulations to be performed in a reasonable amount of computer time. A 5-dimensional function was used to confirm the expected behaviour of the estimators for higher-dimensional problems.

For each function, 3 different error distributions were used to generate the training sets. These were the Gaussian, Laplace and a contaminated normal distribution. The difficulty of estimating the regression function was varied either by varying the amount of training data (and hence the data sparseness) or by varying the error distribution variance. Complexity control was accomplished by varying the number of hidden units and by early stopping.

For each combination of estimator, regression function, error distribution, problem difficulty and number of hidden units, 30 MLPs were trained. This was considered sufficient to obtain reasonably accurate estimates of the fit sampling distribution while allowing all simulations to be performed in a reasonable time.

5.7.1. Some experimental results and discussion

This section discusses one of the Monte Carlo experiments in detail to present the key results and conclusions of the efficiency study. Further, confirmatory results are presented in appendix C.

The results discussed in this section were obtained using the regression function[98]

$$\mu(x_1, x_2) = \exp(-4x_1x_2 - x_1^2 - 3x_2^2) \tag{5.36}$$

sampled over the domain $-1 \le x_1, x_2 \le 1$ on regular 10-by-10 and 7-by-7 grids. This scheme gave training sets containing 100 and 49 data; the smaller data set contains less information about $\mu(x_1, x_2)$, and hence makes avoidance of under- and overfitting more difficult. For each sampling scheme, 3 training sets were created by adding response errors from

- a Gaussian distribution with mean 0 and variance 0.4,
- a Laplace distribution with mean 0 and variance 0.4, and
- a contaminated Gaussian distribution comprised of a Gaussian distribution with mean 0 and variance 0.4, 10% contaminated at scale factor 5 by drawing every 10th error from a Gaussian distribution with mean 0 and variance 2.

Results from Gaussian errors experiments and discussion

Figures 5.4 and 5.5 show the final (complete training) and best (early stopping) fit MSEs obtained for the training data with Gaussian response errors. The points plotted are the means of the 27 smallest fit MSEs out of the total 30 fits for each estimator. The error-bars shown are ±1 standard deviation for each mean.

The 10% error trimming was used to reduce the sensitivity of the results to occasional large MSEs resulting from hidden unit saturation or chronic overfitting. All estimators were found to suffer equally from these problems, and so no advantage is given to any estimator by this trimming procedure.

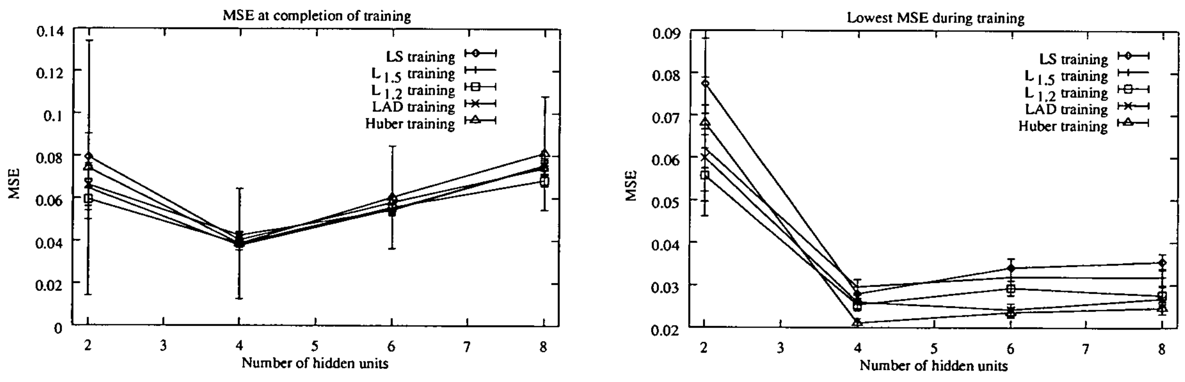


Figure 5.4: Estimator MSEs for the fits obtained using 100 training data with Gaussian errors. The left graph shows MSE at the completion of training, the right graph shows the best early stopping MSEs.

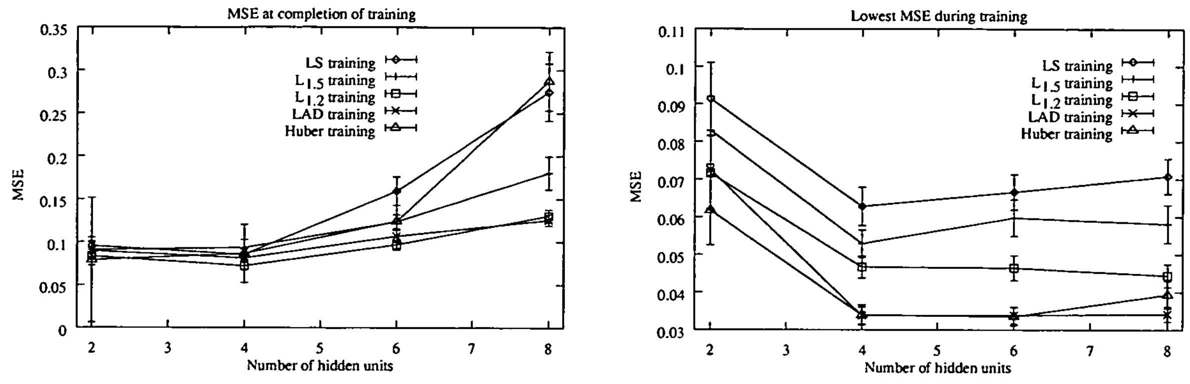


Figure 5.5: Estimator MSEs for the fits obtained using 49 training data with Gaussian errors. The left graph shows MSE at the completion of training, the right graph shows the best early stopping MSEs.

Previous investigations of efficiency in linear regression found LS estimation more efficient than other L_p-Norm or M-estimators when the response errors are Gaussian[145-147]. This result is unsurprising given that LS estimation gives ML parameter estimates when the errors are Gaussian.

For the MLP fits, however, LS estimation gave the largest fit MSEs, and is hence the least efficient estimator. Figures 5.4 and 5.5 show that the LAD and Huber estimators are more efficient (lower MSEs), particularly when early stopping is used.

This raises the question of why LS estimation was least efficient for this problem. The answer is that in previous linear regression work, the test function was also linear, and so under- and overfitting were not problems. Thus the efficiency was determined only by how well the estimator was matched to the response error distribution. In MLP regression, however, the fit MSE depends not only on how well the estimator is suited to the error distribution, but also on how much under- or overfitting occurs. The latter can affect the efficiency quite significantly.

For example, figure 5.4 shows that when early stopping is used, even the least efficient estimator when using 4 hidden units is significantly better than the best estimator when using 2 hidden units. This is because 2 hidden units are not enough to model the regression function well and so underfitting occurs. The MSE is dominated by the fit bias caused by underfitting.

Similarly, both figures 5.4 and 5.5 show that when training to completion, the MSEs increase as the number of hidden units increases above 4. This is because overfitting becomes worse as more hidden units are used. The resulting increase in fit variance increases the fit MSE, and so minimising overfitting is clearly a more important issue than choosing any particular estimator.

The key result illustrated by this example is that avoiding under- and overfitting can be as important as choosing an appropriate estimator for achieving the lowest fit MSE. The LAD and Huber estimators were more efficient than the LS estimator because overfitting occurs more slowly when using robust training methods[13]. When using early stopping, slower overfitting allows a better fit to be obtained before overfitting begins. This is shown clearly in figure 5.5, where the MSEs decrease as the L_p-estimator used becomes more robust ($p \to 1$). This is exactly the opposite result to that obtained in previous linear regression studies. Here, the rate at which overfitting occurs to the sparse data set is more important than how well the estimator is matched to the response error distribution.

The similarity of the Huber and LAD results is because, as explained earlier, the scaling method used with the Huber estimator results in LAD estimation being used for the first 20 conjugate gradient constructions during training. Without reducing the Huber scale factor as recommended earlier, the Huber efficiency dropped to being similar to that of the $L_{1.5}$ estimator, because overfitting occurred quickly for the data lying on the quadratic section of the error function at the start of training.

In other Monte Carlo experiments (such as the one using the sine function data used earlier to illustrate efficiency), the LS estimator efficiency was comparable to that of the LAD and Huber estimators only when there were many training data to define the regression function very well, and hence minimise the problems of under- and overfitting. Since MLP regression is often applied to sparse data sets, where under- and overfitting can be serious problems, this makes the issue of how they affect the efficiency very important.

Results from Laplacian errors experiments and discussion

Figures 5.6 and 5.7 show the end-of-training and early stopping MSEs obtained for the training data with Laplacian response errors.

[13] The likely reason for this is that robust training is not so disposed towards eliminating large fit errors as LS is. I am grateful to Duane DeSimio and Warren Sarle for some useful personal communications confirming this observation.

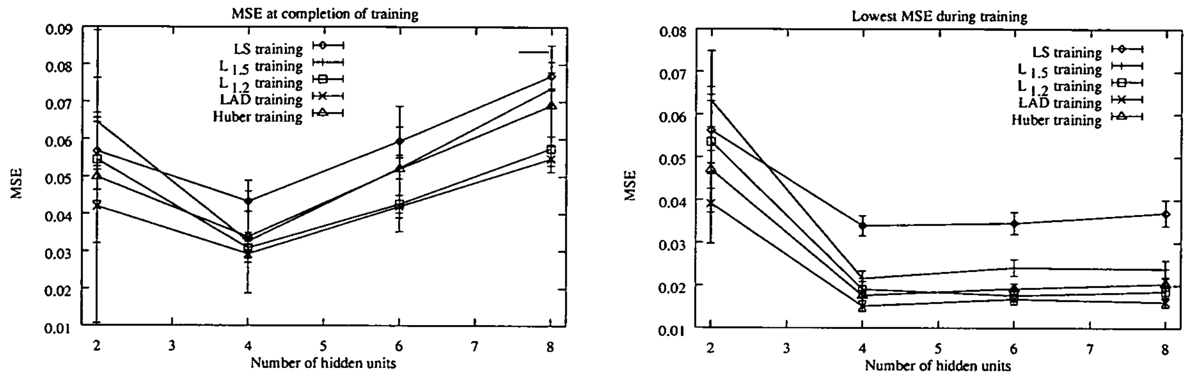


Figure 5.6: Estimator MSEs for the fits obtained using 100 training data with Laplacian errors. The left graph shows MSE at the completion of training, the right graph shows the best early stopping MSEs.

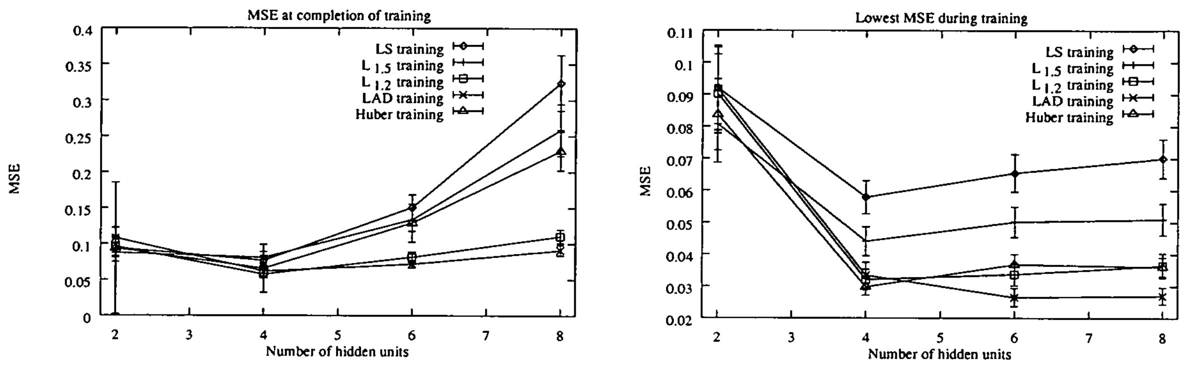


Figure 5.7: Estimator MSEs for the fits obtained using 49 training data with Laplacian errors. The left graph shows MSE at the completion of training, the right graph shows the best early stopping MSEs.

These results are similar to those obtained with Gaussian response errors. The main difference is that the LS estimator has become noticeably less efficient than the other estimators both when training to completion and using early stopping. This is expected, because the longer tails of the Laplacian error distribution increase the importance of using an estimator that is efficient with long-tailed data to fit the MLP, in addition to minimising under- and overfitting.

Results from contaminated normal errors experiments and discussion

Figures 5.8 and 5.9 show the end-of-training and early stopping MSEs obtained for the training data with contaminated normal response errors.

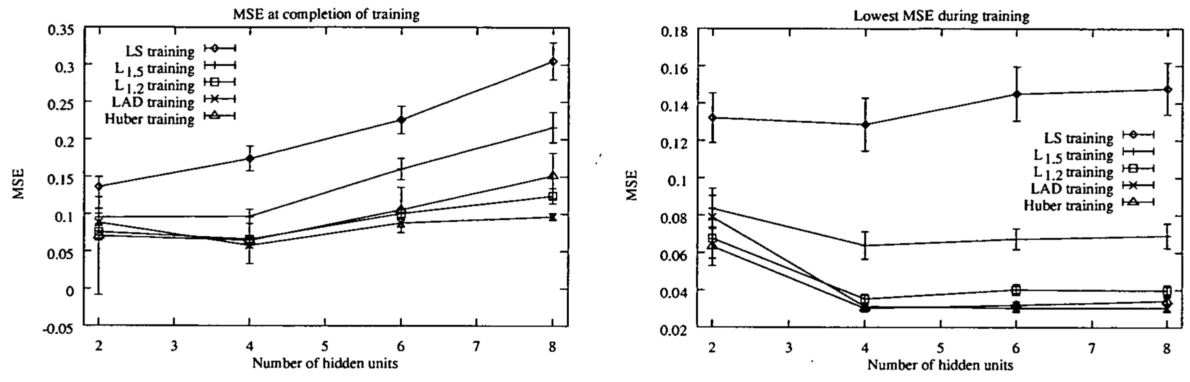


Figure 5.8: Estimator MSEs for the fits obtained using 100 training data with contaminated normal errors. The left graph shows MSE at the completion of training, the right graph shows the best early stopping MSEs.

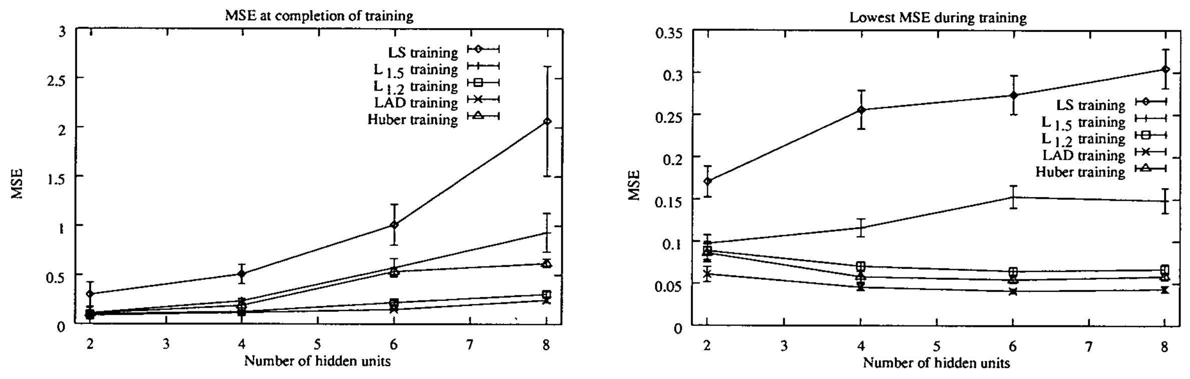


Figure 5.9: Estimator MSEs for the fits obtained using 49 training data with contaminated normal errors. The left graph shows MSE at the completion of training, the right graph shows the best early stopping MSEs.

Again the results are qualitatively similar to those obtained with Gaussian and Laplacian errors, but now both the LS and $L_{1.5}$ estimators are much less efficient than the other estimators. This is because the contaminated normal error distribution has the longest tails of the 3 error distributions used here, and so now using both a very robust estimator and good complexity control are necessary to obtain a food fit.

5.7.2. Efficiency study conclusions

The results of the Monte Carlo efficiency studies are that,

- as expected, robust estimators can be much more efficient than LS estimation when the error distribution has long tails, but
- the estimator efficiency is also influenced strongly by under- and overfitting.

The second result is the key result of this work in terms of deciding which types of robust estimators to use for MLP regression.

Some previous works have suggested that sophisticated M-estimators, notably redescending estimators, should be used to obtain the best fit when using robust training methods[21, 154]. This recommendation appears to be based on the fact that these estimators can be very efficient for location estimation, where under- and overfitting are not issues.

However, the results of this work suggest that in MLP regression, once a simple robust estimator is used to resist any gross outliers, then limiting the degree of under- and overfitting is likely to be the most important problem when trying to maximise efficiency. In fact, since most redescending estimators have quadratic error functions for small- to medium-sized fit errors, overfitting is likely to be a considerable problem when using them, as overfitting was found to occur quickly for LS-like estimators.

In summary, the results given here suggest that whenever some under- or overfitting is unavoidable (which is very often the case in MLP regression), then simple robust estimators are likely to be as, if not more, efficient than more sophisticated M-estimators. Given this, and their advantage of being much simpler to compute, simple estimators such as L_p-Norm estimators appear to be most suitable for robust MLP regression.

As a consequence of these results, it was decided to use only L_p-Norm estimation for the curl modelling problem.

Future robust regression work

Limited time and computer facilities restricted the number and complexity of Monte Carlo experiments that could be performed. Using new, faster computers, future studies should include sparser, high-dimensional data-sets, as these

- present greater difficulties in avoiding under- and overfitting, and
- are more representative of the type of data to which MLP regression is usually applied. Using such data will hence confirm (or refute) the applicability of the present study's results to real regression problems.

Another issue requiring investigation is how outliers in the validation data may affect complexity control. The early stopping results given here used error-free data, both to

- limit the issues to be considered to training issues only, and
- to show the best possible efficiencies under the assumption that the best early stopping point (or best regularisation method) can be found.

In practice, the validation data is also likely to contain outliers, and so a robust error measure should be used for validation[45]. However, robust validation errors may be insensitive to overfitting, because a large error caused by overfitting is indistinguishable from an outlier[14]. Inability to identify overfitting clearly may make overfitting more likely, causing an efficiency loss.

During this project, use of the absolute fit error for validation did not appear to lead to serious overfitting, but no serious comparisons were made between validation using error-free and outlier-containing validation data to assess whether significant overfitting could occur. It was also found that monitoring the weight decay penalty value (even if not using WD for training) and the validation-set SSE can provide good indication of the onset of overfitting, as both errors increase rapidly at this point. However, no attempts were made to assess the general effectiveness of this method, and further examination of these issues is required.

5.8. Leverage in linear and MLP regression

A well-known and important problem in robust linear regression is that a single outlier can still strongly influence some robust estimators when the outlier occurs at an extreme value of one or more of the predictor variables. This is illustrated in figure 5.10, which shows breakdown of both a LS and a LAD linear regression fit to some data containing such an outlier.

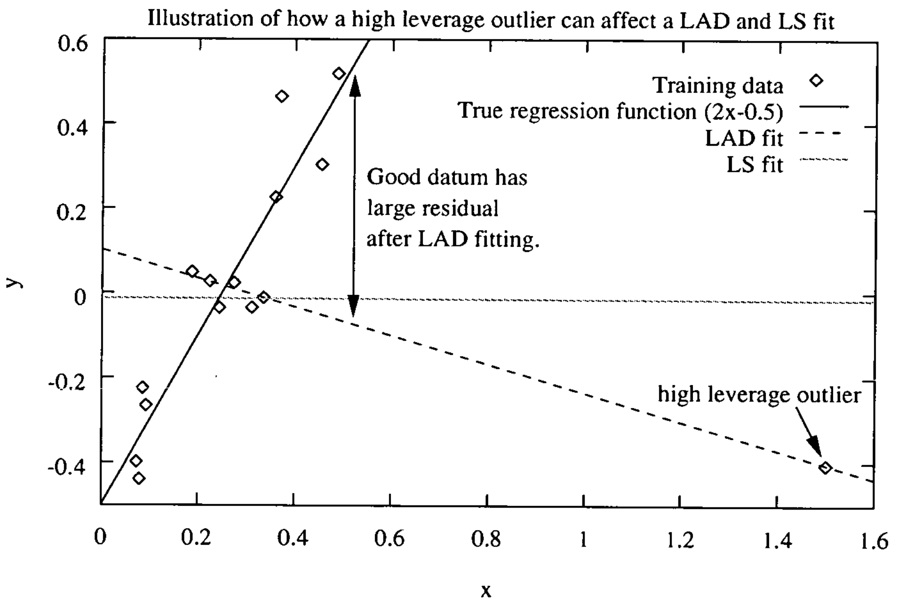


Figure 5.10: Demonstration of LS and LAD estimator breakdown due to a high leverage outlier in linear regression.

Although the LAD estimator has a bounded score function, the IF for the slope parameter also depends linearly on x (see appendix B for a proof), and so outliers with large x can still strongly influence the slope estimate. Such outliers are known as *high leverage*

[14] I am grateful to Warren Sarle for bringing my attention to this problem, and for useful discussion concerning possible ways of avoiding it.

outliers in analogy with how increasing the length of a lever (x) increases the torque (influence this point has in determining the slope of the fit) which can be applied.

A consequence of the influential nature of high leverage data is that the fit is forced to pass through or close to these data, as can be seen in figure 5.10. This makes spotting these outliers using the data residuals (fit errors) difficult, because the largest residuals can belong to good data.

To address this problem, various *high-breakdown* and *bounded-influence* estimators have been devised. Two examples of such estimators are the LMS and LTS estimators discussed earlier, which can resist high leverage outliers by trimming them from the data during training[130, 155, 160]. Other commonly-used estimators include generalised M-estimators (GM-estimators), which are based on standard M-estimators but additionally down-weight the error function contributions from any data which lie far from the centre of the predictor variable distribution[129, 153, 161, 162].

5.9. Leverage and high breakdown estimators in MLP regression

Given the serious problem that high leverage outliers pose in linear regression, namely their ability to dominate the overall fit, it was decided to investigate

- whether high leverage outliers posed the same problem in MLP regression, and
- whether high breakdown estimators would be necessary or useful for MLP regression.

This investigation was further motivated by reports in the literature[156, 158] in which LTS and LMS were found to be useful in MLP classification problems[15], and the suggestion that the good results obtained in some of this work were due to the ability of LMS and LTS to resist high leverage outliers[158].

5.9.1. Typical effect of high leverage outliers in MLP regression

It soon became apparent that high leverage outliers are not as serious a problem in MLP regression as they are in linear regression. In MLP regression, a high leverage outlier will typically have high local influence, causing localised overfitting, but will not control the overall fit.

The reason high leverage outliers typically distort the fit only in the region near the outlier is that, as explained in chapter two, MLP regression is similar to nonparametric

[15] Regression estimators are often used to train MLP classifiers. However, this generally will not work well unless the distribution of the data to be classified looks like a regression error distribution[25, 41]. It is thus difficult to compare results obtained from these experiments directly to MLP regression.

regression in terms of its curve-fitting flexibility. In nonparametric regression, the fit at any point is determined only by the data in the vicinity of that point. Since a high leverage outlier lies far from the other data, it will thus have little affect on the fit to most of the data. For example, a high leverage outlier will have little affect in kernel regression if it lies more than a few kernel bandwidths away from the other data.

Appendix B discusses in more depth the influence of high leverage outliers and their typical effect on the fit in MLP regression.

Can high-leverage outliers be ignored?

Though high leverage outliers will not affect the overall MLP fit, this does not mean that they do not present a problem in MLP regression. In the same way that they cannot affect the fit to most of the data, most of the data cannot affect the fit at the outlier either. This means that the fit near the outlier is determined almost completely by its own response value. If this datum is a genuine (response) outlier, then the fit will be pulled far away from the regression function even if robust training methods are used. It is thus necessary to identify these unreliable regions of the fit after training, and chapter six addresses this issue further.

5.9.2. Usefulness of high breakdown estimators for MLP regression

Given that high leverage outliers do not cause overall fit breakdown in MLP regression, this posed the question of whether the high breakdown estimators developed to address this problem in linear regression have any use in MLP regression. Some experiments which examined the performance of LTS for simple regression problems soon indicated that the answer to this question is 'no', as the next sections illustrate and discuss.

5.9.3. A simple regression problem illustrating the pitfalls of LTS

To demonstrate and explain why LMS, highly-trimmed LTS or other high breakdown estimators should not be used for MLP regression, consider the relatively simple problem of estimating the regression function for the motorcycle impact data shown in figure 5.11.

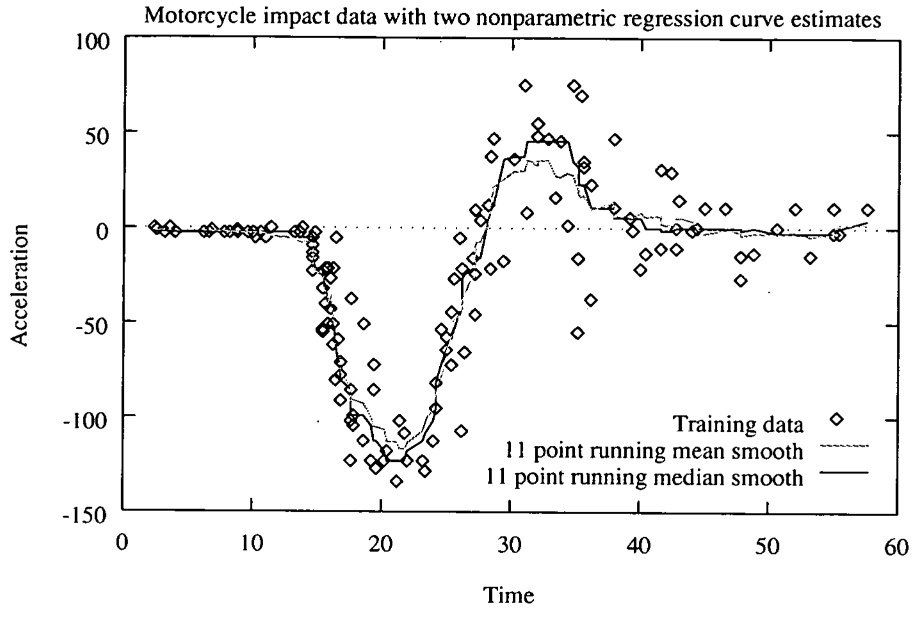


Figure 5.11: Whiplash acceleration versus time data for a motorcycle impact, with LS and LAD nonparametric estimates of the regression function.

Though the true regression function is obviously unknown here, this data is widely used to benchmark nonparametric regression methods[30, 87], and the 11-point running mean and running median shown in the figure correspond to typical LS and LAD estimates of this function. The main difference between these estimates is caused by the possible outliers below the peak near time 30 milliseconds, but they agree well everywhere else.

The regression function was next estimated by training MLPs with 3 to 6 hidden units using LTS estimation with trims of 20, 30, 40, 50, 60, 70 and 90. For each hidden unit and trim combination, 10 MLPs with different initial weight and bias values were trained to indicate the typical fit shape which would be obtained. Though some of the trims may seem quite large, it should be remembered that using LMS estimation would require all but one of the data to be trimmed at every training step (in this case, the datum with the 66th largest squared fit error). Overfitting was found to become quite bad when more than four hidden units were used, and so all results given in this section are for MLPs with four units.

For small trims of 20 to 40 or so, the MLP fits were very similar to the running median smooth. Figure 5.12 shows the 10 fits obtained using a trim of 30. Both variables have been standardised by subtracting the mean and dividing by twice the standard deviation, to prevent hidden unit saturation during training.

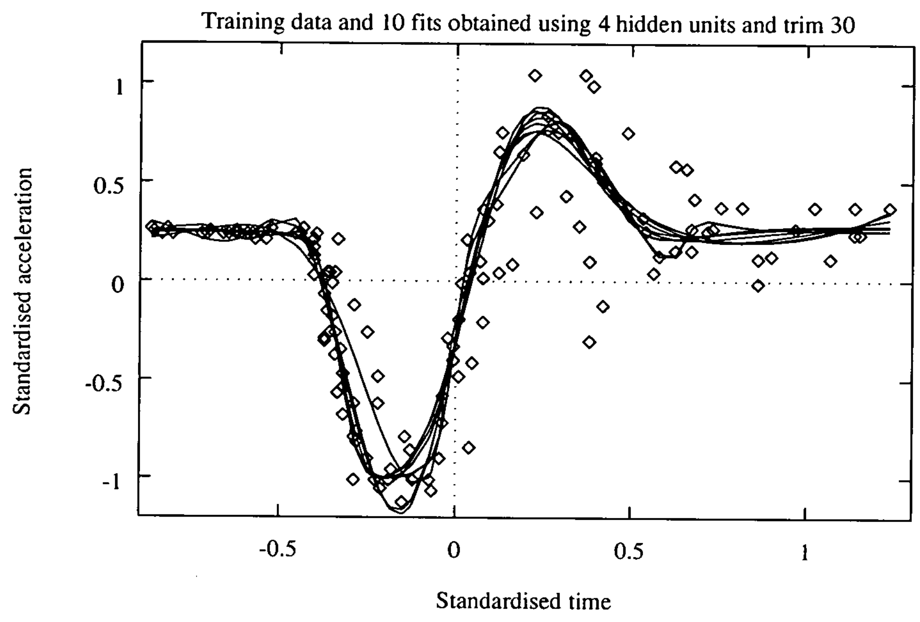


Figure 5.12: 10 MLP fits obtained using four hidden units and LTS estimation with a trim of 30. The fits match the running median fit shown in figure 5.11 very well.

As the trim increased however, the MLP fits began to flatten by pulling in from the data peak near time 0.25 and the trough near time -0.2. This is illustrated in figure 5.13, which shows the 10 fits obtained using a trim of 60.

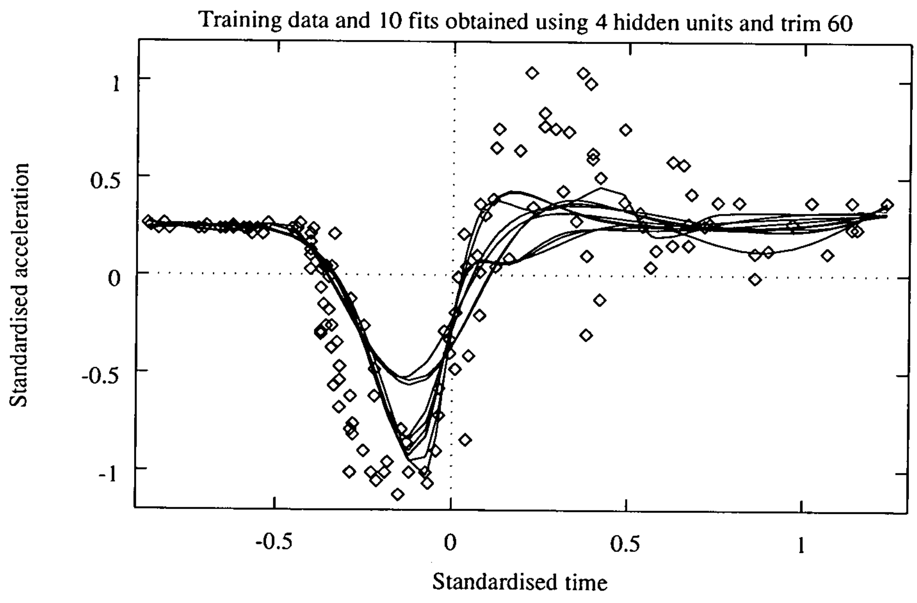


Figure 5.13: 10 MLP regression estimates obtained using four hidden units and LTS estimation with a trim of 60. The regression function is poorly estimated in all cases.

It can be seen clearly that the fit does not show a peak in the regression function near the standardised time of 0.25. Additionally, the minimum near time -0.2 has not been estimated well despite the fact that it is quite clearly defined by the data. As the trim increased towards 70 this minimum was estimated very poorly, as all the fits become very flat.

Discussion

The MLP fits obtained using lightly trimmed LTS compared well with the running median smooth, and also fits obtained using LAD estimation (not shown). However, LAD training has at least two advantages over LTS training:

- LTS training requires much more computer time because of the large amount of sorting required. The best sequential sorting algorithms have worst case time complexity of order $nlog(n)$, where n is the number of data, and so LTS training can be very slow for training sets comprised of more than a few hundred data[16].
- The test problem involved dense data and so complexity control was not a serious problem. As shown earlier, however, LS-like estimators require good complexity control to obtain good efficiency when training with sparse data.

As the trim was increased, increasingly poorer fits were obtained because too much good data was being trimmed at each training step, and training focussed on the remaining untrimmed data. This increased the difference in the fit errors between these groups of data, and hence caused this situation to worsen as the trim was biased more strongly towards the already low error data. A similar problem is discussed in[162-164], where the authors consider the use of high-breakdown estimators for fitting polynomials to good-quality data containing few or no outliers, and examining the residuals from high-breakdown linear fits for signs of systematic nonlinearity indicating the need for a nonlinear model. In all cases they concluded that high-breakdown estimators were inadequate for this task compared to simpler robust estimators, because they could reject too much data.

Basically, when using flexible regression methods in problems where the shape of the regression function is unknown, rejecting large amounts of data can cause some genuine trends within the data to be ignored during training. In the case of LMS regression, training can stop after only half of the data has been fitted reasonably well, even if the other half are not outliers.

[16] Significant speed-up may be possible by using the sorted error ordering at each training step and an initial ordering for the next step sort. However, even a time overhead of, say, 20% compared to LAD estimation can still be excessive when considering training runs requiring one or more weeks of computer time.

Some situations where high-breakdown estimators may focus on only a reduced sample of the training data include:

- Problems where the error distribution is heteroscedastic, in which case training may focus solely on the observations with the lowest error variances.
- Problems where the data are densely clustered in some regions. This may lead to the LMS estimator focusing on these regions, giving poor fits in other regions with sparser data.
- Problems where the regression function includes strong near-discontinuities, such as steps in the regression function. This could place groups of data far from the main data cluster, which would be completely trimmed and hence never fitted, even though they are not outliers, but indicate a genuine trend in the data.

In summary, despite previous recommendations for their use, strongly trimmed LTS, LMS and other high-breakdown estimators appear to be unsuitable for MLP regression.

5.10. Summary and conclusions

This chapter has looked at the use of robust regression methods for training MLPs with data which may contain outliers. The emphasis in this chapter has been towards practical issues which affect how to choose and use these estimators for MLP regression, with the mostly theoretical material placed in the appendices.

The two major issues examined were:

- What affects estimator efficiency in MLP regression, and hence the choice of appropriate estimators for MLP regression?
- Are high-breakdown estimators necessary, or even useful, in MLP regression?

Understanding these issues is important for choosing and using appropriate robust estimators for MLP regression.

Key efficiency study results

The key result of the Monte Carlo efficiency studies was the realisation that simple L_p-Norm estimators with $p \approx 1$ were likely to give as good, or even better, fits in most MLP regression problems as more sophisticated estimators such as re-descending M-estimators.

The reason for this is that the fit accuracy depends on both

- how well the estimator used to train the MLP is suited to the error distribution, such as resistance to outliers, and
- how well the fit complexity is controlled to avoid over and underfitting.

In the studies performed, it was found that once a simple robust estimator had been used to resist any gross outliers, avoidance of under- and overfitting quickly became the most important problem when trying to obtain the best fit. Overfitting was found to occur more slowly for the simple LAD-like estimators, and this had a greater affect on the fit accuracy than how well the LAD estimator was matched to the true response error distribution.

Choosing an estimator which is very well-matched to the error distribution is likely to be important only when overfitting and underfitting are not serious problems. This is unlikely in most MLP regression problems because MLP regression is commonly applied to sparse data, which makes avoidance of overfitting difficult.

The overall conclusion of this study was that L_p-Norm estimation with $p \approx 1$ will give good efficiency for many MLP regression problems, while also requiring less computer training time than more complicated estimators.

Using high breakdown estimators for MLP regression

This investigation was motivated by the importance of using high breakdown estimators in linear regression when high leverage outliers occurs.

The key results of this investigation were

- the demonstration that high breakdown estimators are not necessary in MLP regression because high leverage outliers can only cause local overfitting, and
- the demonstration that, despite previous recommendations for their use, high breakdown estimators should not be used for MLP regression.

High breakdown estimators were found to be unsuitable for MLP regression because they can ignore a large proportion of the training data. This can result in failure to fit important features of the regression function during training if the whole feature is rejected as a large group of outliers.

Overall conclusion

The overall conclusion of this work is that, when using robust estimators because data outliers are suspected, then simple robust estimators such as L_p-Norm estimators with $1 \leq p \leq 1.5$ are likely to give the best results in MLP regression. More sophisticated estimators are unlikely to give better fits because of the extra complexity control issues that their use involves.

It was thus decided only to use L_p-Norm estimation for the curl modelling problem.

Chapter 6

Using leverage to identify overfitting and cross-validate MLPs

6.1. Introduction

The previous chapter and appendix B considered the likely effects of training MLPs with data which contains high-leverage outliers. There it was shown that the fit usually will be very unreliable near these data because of local overfitting.

For most effective use of the fit for prediction, this unreliability must be signalled to the model user so that remedial action can be taken or suitable care exercised when using predictions from overfitted regions. Since wide confidence intervals indicate low certainty in the fit accuracy, one method for identifying local overfitting is to examine confidence intervals for the MLP fit, which can be obtained using bootstrapping[64, 165]. Bootstrapping is, however, very computer-intensive and so it was decided to investigate whether existing methods for computing leverage in linear regression could be extended to MLP regression. This is a much less computer-intensive approach to identifying overfitting, and so would speed-up model development.

While investigating the relationship between underfitting, overfitting and leverage, it was realised that the MLP leverages could be used to reduce greatly the amount of computation required to estimate the MLP generalisation ability using the method of leave-one-out cross-validation. The particular advantage of this technique over the data splitting method considered so far is that it allows all the available data to be used for training, and is thus very useful when working with a limited amount of data. The curl modelling problem a prime example of such a task, and provided the primary motivation for investigating this application of leverage in MLP regression.

Chapter structure

Sections 2 and 3 of this chapter examine linear estimators as a template for understanding leverage in general and also for illustrating the relationship between leverage and overfitting in MLP regression. The two methods which were investigated for computing MLP fit leverages are introduced and discussed.

In section 4, some simple examples are given which illustrate how leverage can be used to identify local overfitting. Some properties of the leverages and how these can affect the interpretation of the leverage values are also discussed. The advantages and disadvantages of using leverages to identify overfitted regions of the fit rather than using the computer-intensive bootstrapping method is also discussed.

Sections 5 to 10 discuss how to use leverage for fast cross-validation of MLP regression fits. An explanation of how this method works is given and two examples are given which illustrate the usefulness of this technique. The second example uses the curl data both to show a real application of the fast cross-validation estimator and to highlight and discuss practical issues to be considered when using this estimator.

6.2. Leverage in linear estimation and MLP regression

So far leverage has been discussed only in terms of outliers and the highly influential nature of high leverage outliers. In this section leverage is examined from the more general viewpoint of linear estimation as preparation for work presented in later sections. This approach gives clear insight into the relationship between leverage and overfitting in MLP regression.

6.2.1. Linear regression estimators

Linear regression estimators derive their name from the fact that the value of the fit at each training datum position, $\underline{x}_i$, is a linear sum of the response data,

$$\hat{y}_i = h_{i1}y_1 + \cdots + h_{ii}y_i + \cdots + h_{in}y_n , \tag{6.37}$$

where the values of the coefficients h_{ik} depend only on the predictor variable data, $\{\underline{x}_i\}$[38]. Common examples of linear estimators include LS kernel regression, where the coefficients in (6.37) are the normalised kernel weights, and LS linear regression. The following sections relate the values of the coefficients to local influence; for both LS linear[153] and kernel[30] regression $0 \leq h_{ik} \leq 1$ and

$$\sum_{k=1}^{n} h_{ik} = 1 . \tag{6.38}$$

6.2.2. Leverage and overfitting in linear estimation

In the context of the work presented here, the most important consequence of (6.37) is that if y_i is perturbed by Δy_i and the model re-fitted, then the fit at $\underline{x}_i$ changes by

$$\Delta\hat{y}_i = h_{ii}\Delta y_i . \tag{6.39}$$

If $h_{ii} \approx 1$ then the fit closely follows the perturbed datum. In chapter two it was stated that this is what happens when overfitting can occur in MLP, kernel and other types of flexible regression, and so a large h_{ii} can indicate local overfitting near $\underline{x}_i$.

In kernel regression for example, $h_{ii} \approx 1$ indicates that when the kernel centre is positioned near $\underline{x}_i$ the other training data lie far away in the kernel tails. The fit near $\underline{x}_i$ is thus determined almost totally by the single datum y_i. Indeed, it has been noted in the linear regression literature that[144, 153]

$$\textit{effective number of data determining the fit at } \underline{x}_i \approx \frac{1}{h_{ii}} .$$

If y_i is an outlier then the fit will be a poor estimate of the regression function. This problem cannot be addressed by using a robust estimator such as taking the median of the weighted kernel data instead of the average. This is because the sampling variances of the mean, median and all the other robust estimators discussed in chapter five become infinite as the number of data decreases to 1. Thus the fit will always have high variance (wide confidence intervals) where it is based on only one or two data, as the lack of data conveys little information about the regression function[38, 40]. When fitting to data, the fit at any point should ideally be determined by many data so that it is truly representative of general trends in the data, but not local random variations.

Nomenclature

In this thesis, the coefficients h_{ii} are called the fit leverages. This is because high leverage outliers in LS linear regression can be identified by the fact that they have large h_{ii} ($h_{ii} > 0.5$ is generally considered a high leverage)[144]. The coefficients are not conventionally called leverages in general linear estimation.

6.3. Leverage in MLP regression and nonlinear estimation

Following the intuitive relationship between leverage, overfitting and the fit confidence intervals in kernel regression outlined in the last section, it was realised that if similar leverages could be computed for MLP fits then this could provide a method for detecting, for example

- local overfitting at isolated data points such as high leverage outliers, or
- local overfitting when using the spline penalty described in chapter three, caused by using smoothing intervals which are too wide.

This ability to diagnose local overfitting would have several uses:

- It could provide a method for detecting overfitting which may not be otherwise detected when validation data is limited.
- When training with sparse data, even the fit with the lowest validation error may still be unreliable due to overfitting where the data is most sparse. This overfitting could be identified quickly after training by examining the fit leverages.

Leverage in nonlinear estimation.

In MLP regression it is not possible to write the fitted values in the form (6.37) where the coefficients depend only on $\{\underset{\sim}{x}_i\}$. However, for small response perturbations, it is possible to define leverages for which

$$\Delta \hat{y}_i \approx h_{ii} \Delta y_i \ . \tag{6.40}$$

A high leverage here indicates high sensitivity of the fit to perturbations of the data. Since this is a characteristic of overfitting, these leverages could thus be used to indicate, after training, where overfitting has occurred. Avoidance of overfitting is always an important problem in MLP regression, and so this would be a very useful tool.

The small perturbation limitation arises from the fact that the leverages depend on both $\{\underset{\sim}{x}_i\}$ and $\{y_i\}$ in nonlinear estimation, and so change as the response data is perturbed. Equation (6.40) gives accurate estimates of $\Delta \hat{y}_i$ only if the leverage remains approximately constant as y_i changes. The examples given later discuss some consequences of this limited range of leverage validity.

The next sections discuss two methods which can be used to compute leverages for MLP fits. Both methods assume that training has proceeded until the training error is minimised, and consequently I realised that they could not be used in conjunction with early stopping to prevent overfitting. Instead, either adjusting the number of hidden units or regularisation penalties of the type examined in chapter three must be used to prevent overfitting. In fact, it does not even seem possible to assess leverage sensibly if training is not complete. To see why, suppose that y_i is perturbed after early stopping is used to stop training. If training is then resumed to assess the change in the fit caused by this perturbation, then it is impossible to distinguish between change due to the perturbation and that resulting simply from further training. If training is completed first then any change in the fit is due only to the perturbation of y_i.

6.3.1. Tangent plane leverage

The tangent plane method assumes that the nonlinear regression model $f(\underset{\sim}{x}; \underset{\sim}{\theta})$ can be approximated by

$$f(\underset{\sim}{x}; \underset{\sim}{\theta}) \approx f(\underset{\sim}{x}; \hat{\underset{\sim}{\theta}}) + \nabla_{\underset{\sim}{\theta}} f(\underset{\sim}{x}; \hat{\underset{\sim}{\theta}})^T (\underset{\sim}{\theta} - \hat{\underset{\sim}{\theta}}) \tag{6.41}$$

for $\underset{\sim}{\theta}$ in a close enough vicinity of $\hat{\underset{\sim}{\theta}}$[127, 166, 167].

It is relatively simple to show (see appendix D) that if the model approximated by (6.41) is fitted using weight decay penalised LS then the leverages are the diagonal elements of the matrix

$$H = \hat{Z}\left(\hat{Z}^T \hat{Z} + \lambda I\right)^{-1} \hat{Z}^T \tag{6.42}$$

where I is the identity matrix and $\hat{Z}$ is the Jacobian matrix with $(i, j)^{th}$ element

$$\hat{z}_{ij} = \left. \frac{\partial f(\underline{x}_i; \underline{\theta})}{\partial \theta_j} \right|_{\underline{\theta}=\hat{\underline{\theta}}} . \tag{6.43}$$

Methods for deriving tangent plane leverages for models fitted using non-LS estimation are discussed in[168, 169].

6.3.2. Jacobian leverage

Another approach to estimating leverage in nonlinear regression is to compute the quantity

$$h_{ii} = \frac{\partial \hat{y}_i}{\partial y_i} \tag{6.44}$$

directly without assuming a tangent plane approximation. This is known as a Jacobian leverage[166, 170].

Methods for computing Jacobian leverages are discussed in[166, 171, 172]. These are considerably more complicated than the tangent plane method, however, and so will not be discussed in detail here.

The mean-shift perturbation method described in[171] has been used by Schall and Gonin[173] to derive an expression for the Jacobian leverage matrix for a nonlinear model fitted using L_p-Norm estimation with $p > 1$. Their work can be extended easily to MLP regression where training is performed using L_p-Norm estimation with a regularisation penalty, $J(\underline{\theta})$. The resulting Jacobian leverage matrix is

$$H_J = p(p-1)D\hat{Z}\left[p\left((p-1)\hat{Z}^T D^2 \hat{Z} - B_p\right) + \lambda \nabla^2_{\underline{\theta}} J\right]^{-1} \hat{Z}^T D \tag{6.45}$$

where D is the matrix

$$D = diag[|r_1|^{p/2-1}, \cdots, |r_n|^{p/2-1}] \tag{6.46}$$

with residual $r_i = y_i - \hat{y}_i$, and

$$B_p = \sum_{i=1}^{n} |r_i|^{p-2} r_i \nabla^2_{\underline{\theta}} f(\underline{x}_i; \hat{\underline{\theta}}) . \tag{6.47}$$

Only L_p-Norm training was examined in depth because this method was chosen in chapter five for developing the curl model. However, the mean-shift perturbation approach can be applied easily to other estimators.

6.3.3. How do tangent plane and Jacobian leverage differ?

If the tangent plane approximation is exact or if every training datum is interpolated by the fit, then $B_p = 0$ and the tangent plane and Jacobian leverage matrices are identical. However, if B_p cannot be neglected, then the tangent plane and Jacobian leverages can differ significantly[166]. Hence the question of which leverages may estimate most accurately changes in the fit cause by perturbing the data must be considered.

While the tangent plane method uses a linear approximation of the true regression model, the Jacobian leverages are based on a higher-order approximation which also accounts for some of the intrinsic nonlinearity of $f(\underline{x};\underline{\theta})$[172, 174]. Consequently, the Jacobian leverages would be expected to give more accurate estimates of how far the fit will shift when a response datum is perturbed. This was confirmed by experiment and so Jacobian leverages were used for most of the work reported here.

6.4. Investigating the relationship between leverage and overfitting

Various simulation studies with artificial training data were performed

- to check that the (rather complicated) leverage programs gave leverages which accurately predicted how a MLP fit would respond to small response data perturbations, and
- more importantly, to investigate the relationship between leverage and overfitting in MLP regression.

This section focuses on the second of these using two simple examples which illustrate how leverage can be used to identify local overfitting and to estimate approximately the extent of overfitting.

6.4.1. Illustrating the relationship between leverage and overfitting

Twelve training data were created by sampling the function

$$\mu(x) = 2x + 1 \tag{6.48}$$

at intervals of width 0.05 over the range $x = -0.2$ to $x = 0.35$ and adding a high leverage outlier at $(1, 0)$. Response errors were generated using the Gaussian distribution $N(0, 0.3)$.

MLPs with 8 hidden units were fitted to this data using LS training with the spline roughness penalty described in chapter three. The roughness was sampled at intervals of width $\Delta x = 0.025$ over the interval $x = -0.6$ to $x = 1.3$. Figures 6.1 and 6.2 show some MLP fits and their Jacobian leverages obtained for different smoothing parameter values, λ.

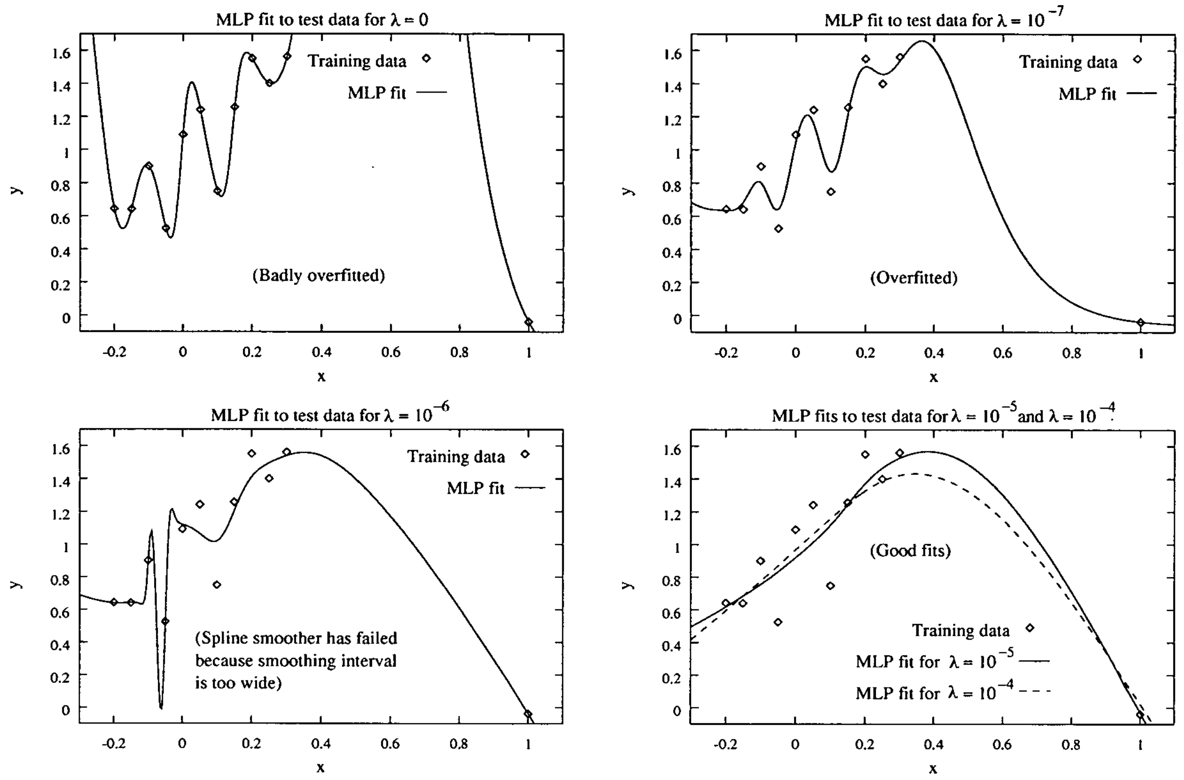


Figure 6.1: Training data and some MLP fits obtained using different amounts of smoothing. The fit interpolates every training data when $\lambda = 0$.

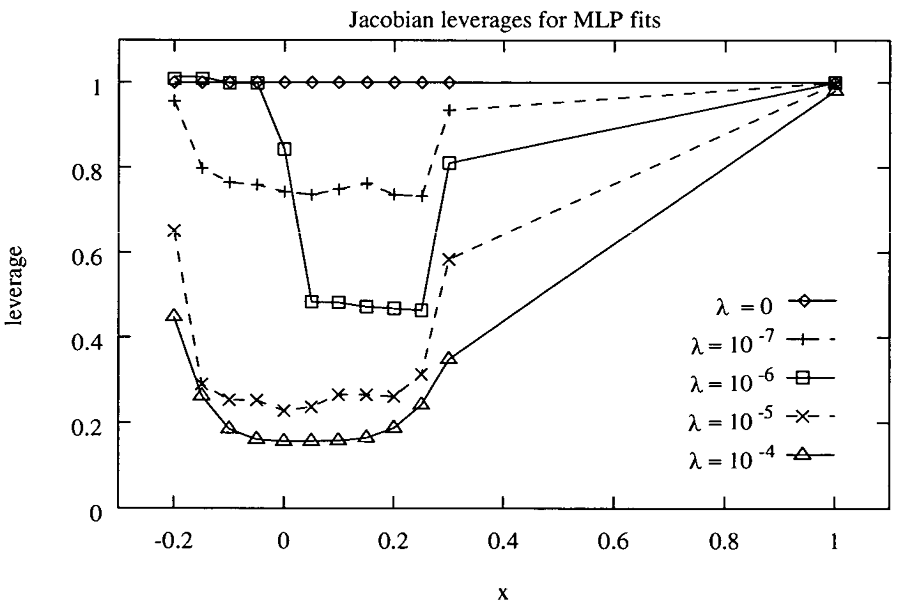


Figure 6.2: Leverages for the fits shown in figure 6.1. Curves have been drawn through the leverages to highlight trends.

When $\lambda = 0$ the MLP is badly overfitted. Even without viewing the fit, this could be deduced from the fact that all the leverages are 1. When all leverages are 1, the fit can follow small perturbations of any of the training data exactly. This occurs because the

MLP has sufficient flexibility to interpolate all the data, in the same way that a sufficiently complex polynomial could be forced to pass through each data point irrespective of their response values.

As λ increases, the overfitting is reduced and this is indicated by the decreasing leverages. The leverages for the data near $x = -0.2$ and $x = 0.35$ are higher than for the data which lie between them because they lie at the edges of the central data cluster, and so the fit is determined only from one side. The same effect is commonly seen in kernel regression, where the fit confidence intervals widen at the data edges[38, 40].

The leverage of the datum at $x = 1$ always remains near 1 indicating that all fits are locally overfitted in this region. As stated before, this is because the fit in this region is essentially determined by one datum and so must pass near it, even if it is an outlier. This overfitting may not be detected in practice even if data splitting is used because high leverage outliers, by definition, occur far away from any other data and so there may be no other data in the vicinity with which to test the fit. Even if validation data is available and gives a reasonably low prediction error estimate for the local fit, examining the leverages provides notification that the accuracy of the fit near $x = 1$ is still questionable.

The diagnostic usefulness of the leverages is further illustrated by the fit obtained with $\lambda = 10^{-6}$. This is badly overfitted for $x < 0$ because the spline smoothing interval is too wide to prevent overfitting. This failure is indicated clearly by the fact that the leverages are all close to 1 for this data, and this gives a further method for testing for this problem in addition to the methods discussed in chapter three.

This simple example has illustrated that leverages near 1 indicate local overfitting has occurred, and that the leverages can thus be used to identify unreliable regions of the fit.

What is a safe leverage and what should be done with high leverage data?

In the above example, identifying overfitting was easy because the relevant leverages had values close to 1. In practice, however, the fit which minimises the validation error may have a range of data leverages. This then raises the question of what constitutes an acceptably high leverage (i.e. safe from overfitting) and what is too high.

During this work, it was noted that

- leverages above 0.7 indicate that overfitting may have occurred, and
- leverages below 0.5 appear to be safe from overfitting, though some overfitting may still occur around data with leverages near 0.5.

While these limits were derived empirically, their values seem reasonable in that at least two data contribute to the local fit when the leverage is less than 0.5 while only just over 1 datum contributes when the leverage is greater than 0.7.

When high leverage data result after fitting, further investigation is required to determine if the high leverages are due to overfitting at a remote datum or, for example, because of a spline penalty failure as illustrated in the previous example when $\lambda = 10^{-6}$. In the former case, little can be done to reduce the leverage other than to gain more information (usually in the form of more training data) about the regression function near the high leverage data.

6.4.2. A further example illustrating the properties of leverages

The previous example illustrated how leverage can be used to diagnose possible local overfitting and hence locate unreliable sections of the MLP fit. This section gives a further example which illustrates how high leverage indicates low fit reliability. A comparison is also made between using leverages and bootstrapping to detect different types of fit unreliability. This example also highlights some properties of leverage in non-LS MLP regression, and how these affect the interpretation of the fit leverages.

A simple training data set was constructed by sampling the function

$$\mu(x) = x(x+1.5)(x-0.5) \tag{6.49}$$

at 25 random locations in the interval $x = -1.5$ to $x = 1.5$. Response errors from the distribution N(0, 0.25) were added to 20 of the data and errors from N(0, 1.5) were added to the remaining data to generate outliers. $L_{1.2}$ estimation was used to give resistance to outliers while fitting the MLP. The spline smoothing penalty was used to control the fit complexity with 31 sample points spaced at $\Delta x = 0.1$ in the interval $x = -1.5$ to $x = 1.5$. The smoothing parameter value $\lambda = 0.0005$ was chosen by increasing λ from zero until a visually good, smooth fit was obtained. In practice, the true regression function would, of course, be unknown and so λ would be set to minimise a prediction error estimate. However, the method used for choosing λ is not important here.

Figure 6.3 shows the MLP fit. The data with the largest leverages and their leverage values are also indicated in this figure. Though the fit approximates $\mu(x)$ well, the high leverages indicate that it is in fact quite poorly defined by the data in some regions. This is confirmed by figure 6.4, which shows 40 fits obtained by the method of bootstrapping points[64, 175]. This method approximates the sampling distribution of the regression fit, and so can be used to obtain confidence intervals for the fit. The fit confidence interval is narrowest where the bootstrap fits are most tightly clustered.

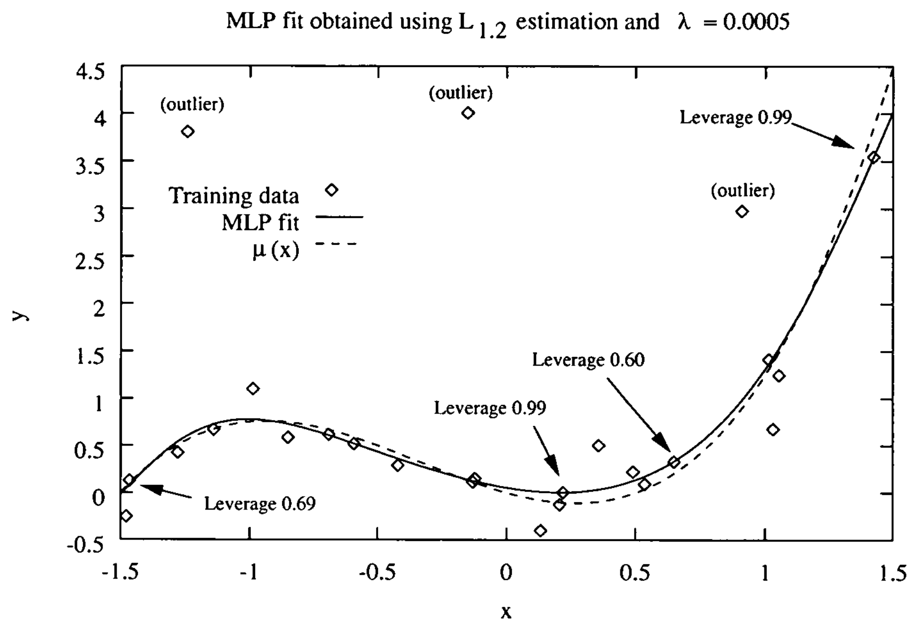


Figure 6.3: Training data, regression function and MLP fit for the second leverage example.

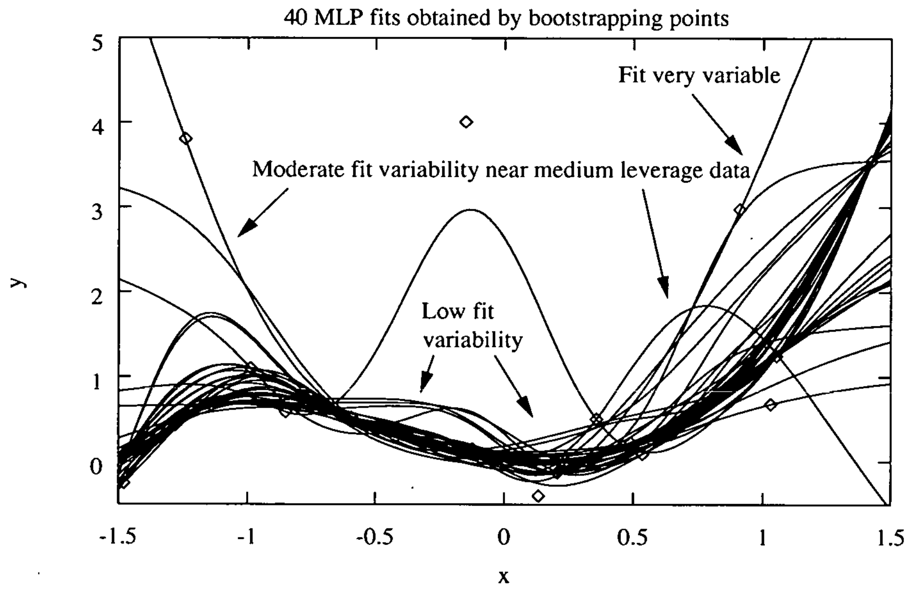


Figure 6.4: 40 fits obtained by bootstrapping points and re-fitting the MLP. The fit confidence interval is widest where the bootstrap fits are most variable.

Figure 6.4 shows that the fit is very unreliable near the high leverage datum near $x = 1.4$. In fact, the original fit at this point was good only because the datum had a low response error. When the response value was increased and decreased by 2 units and the MLP retrained, the fit followed the point. Thus, had the original response error been large, the fit would have given a poor estimate of the regression function here. The bootstrap fit variability is also high near the medium leverage data at $x = -1.5$ and $x = 0.6$. Both methods again indicate the high fit variability, and hence low reliability, in these regions.

While both bootstrapping and examination of leverages can be used to detect overfitting, examining leverages has two advantages:

- It is much less computer-intensive because repeated re-fitting is not required. Thus remedial action can be taken quickly to reduce overfitting if necessary.
- High leverages specifically indicate local overfitting while wide confidence intervals do not necessarily imply overfitting has occurred. For example, localised high response error variance will cause the fit variability to increase even if overfitting does not occur.

Thus, examining leverages provides a faster method for identifying overfitting during model development. As mentioned above, it does not, however, detect all causes of localised high fit variance, such as localised high response error variance. Thus, bootstrapping should be used after the model is developed if an overall assessment of fit reliability, such as confidence intervals, is required.

Properties of the leverages for non-LS fits

The above example also highlights an important relationship between the fit leverages and the change of value function (see appendix B). It can be seen that the fit at the high leverage datum near $x = 0.25$ is much less sensitive to response perturbations than the fit at the high leverage datum near $x = 1.4$. The reason for this discrepancy becomes apparent when the leverages are plotted against the fit residuals, see figure 6.5.

Figure 6.5 shows that the leverages are large only for those data with very small residuals. The reason for this is that L_p-Norm estimates obtained with $p < 2$ are most sensitive to the data with the smallest residuals. This can be seen by examining these estimators' change-of-value functions and noting that the largest shift sensitivity occurs for data with small residuals (see appendix B).

One consequence of this property of L_p-Norm estimators is that data with very small residuals always have high leverages when $p < 2$. If the fit passes close to a datum in a dense section of data then the leverage will be high even though the fit will not change much if a large perturbation is applied to this datum. In this case, the high leverage indicates high sensitivity only to very small perturbations, and should not cause concern about the local fit reliability. Data flagged as having high leverages when using L_p-Norm training with $p < 2$ should thus be perturbed and the MLP re-fitted to identify those data sensitive to large perturbations, such as high leverage outliers. Unlike bootstrapping however, only one or two re-fits will be necessary to confirm genuine high fit variability flagged by the fit leverages.

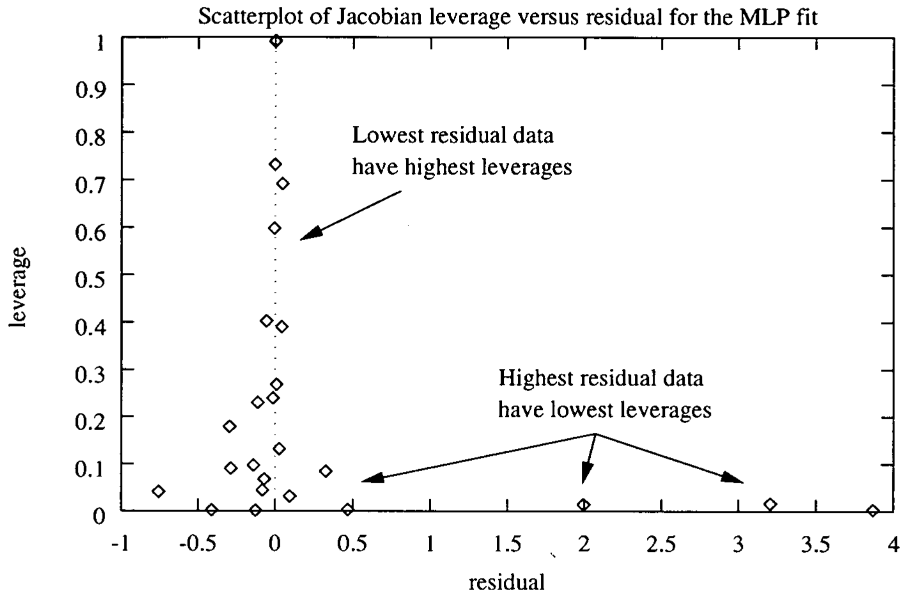


Figure 6.5: Scatterplot of leverage versus residual for the fit shown in figure 6.3.

This second example has again illustrated the use of leverage to identify unreliable regions of a MLP fit and has also shown

- how the behaviour of L_p-Norm estimator leverages can be related intuitively to their influence functions, and
- that this behaviour can cause high leverages in some reliable regions of the fit when $p < 2$. The genuinely unreliable can be identified by perturbing all data reported as having high leverages and then re-fitting the MLP.

6.5. Leave-one-out cross-validation for MLP regression

So far, only an intuitive relationship between leverage and overfitting has been presented. In practice, however, it is the effect of over- and underfitting on the fit prediction error estimate which is of most interest. It would thus be useful if this effect could be related to the leverages, and this is considered in the remaining sections of this chapter.

Using leverages for cross-validation

While examining the relationship between the fit leverages and complexity in MLP regression, it was realised that one very important possible use of leverage was in estimating the generalisation ability (prediction error) of a MLP fit using the method of leave-one-out cross-validation (CV).

Leave-one-out cross-validation, usually just called cross-validation, is a form of data splitting where only one test datum is reserved at any time[176-178]. This allows more data to be used for training than does conventional data splitting, and is thus preferable when the available data are distributed sparsely and it is wished to use as much data as possible for training[64, 179]. As explained in chapter two, this situation arises commonly when there are many predictor variables. The curl data is one example of such a data set, and for this reason it was decided to pursue further the possibility of using leverages for cross-validating MLP regression fits.

What is cross-validation?

Suppose $(\underline{x}_i, y_i)$ is reserved as single test datum and the MLP is trained using the remaining $n-1$ data. The L_p-Norm leave-one-out estimate of the fit prediction error is then

$$\textit{prediction error} = |y_i - \hat{y}_{(i)}|^p \tag{6.50}$$

where $\hat{y}_{(i)}$ denotes the estimate of the regression function at $\underline{x}_i$ given by the fit to the data set which excludes this datum. Reserving only one test case clearly will not provide a good estimate of the overall prediction error, and so leave-one-out CV requires this process to be repeated n times, once for each datum. The overall estimate of the prediction error is the average of the individual prediction errors

$$\textit{cross-validation error} = CV_1(\lambda) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_{(i)}|^p \tag{6.51}$$

and is called the cross-validation error or score. Here, λ is a quantity which controls the fit complexity, such as the number of hidden units or the value of a regularisation parameter. The cross-validation error is shown as a function of λ to emphasise the fact that we usually wish to find the value of λ that minimises the prediction error.

Though averaging errors based on a single test datum may seem an unlikely estimator of the prediction error, cross-validation does give statistically consistent prediction error estimates for a number of regression methods[176, 177, 180, 181]. The consistency of cross-validation in MLP regression is discussed in[179].

Computational aspects of cross-validation

When using cross-validation, none of the n MLPs trained for computing the cross-validation error are actually used as the final regression model. They are used only to estimate the prediction error for the MLP trained using all the available training data, and so $n+1$ MLPs must be trained overall. Leave-one-out CV is thus a very computer-intensive technique, and the time required to train these n additional MLPs may prohibit its use. This is especially true given that the process must be repeated for various values of λ when trying to minimise the CV error.

One way to reduce this time is to use k-fold CV[64, 182]. Here, the data are split into k roughly-equally-sized sets, where $k-1$ of these sets are used for training while one set is reserved for testing. The prediction error is estimated by leaving each set out in turn and by averaging the individual test set errors in the same manner as leave-one-out CV. If the data are relatively sparse, however, then it may be necessary to use a large k to maximise the number of training data, and so the speed-up will be limited.

6.6. A fast cross-validation error estimator for MLP regression

The main goal of the following work was to investigate a technique for greatly reducing the amount of computation required to compute the cross-validation error for a MLP, and also to examine the properties of the resulting estimator. The advantages of the approach described here are:

- It allows the number of training data to be maximised, because a validation set is not required to estimate the prediction error.
- It does not require additional validation MLPs to be trained, and so allows the cross-validation error to be computed very quickly after training.

6.6.1. Deriving the MLP cross-validation estimator

The technique used here to estimate the cross-validation error for MLPs is a direct extension of Craven and Wahba's fast method for cross-validating linear estimators[183]. Though this method gives the true CV error only for linear estimators, it was realised that it could be extended easily to give an estimate of the true CV error for MLP regression using the tangent plane or Jacobian leverages.

This section discusses the principle behind Craven and Wahba's method, to explain how it can be extend to MLP regression.

Craven and Wahba's fast cross-validation method

Consider a parametric regression fit obtained using the fit error function $E(\underline{\theta})$ with the regularisation penalty $J(\underline{\theta})$,

$$\textit{training error} \;=\; E(\underline{\theta}) + \lambda J(\underline{\theta}) \ . \tag{6.52}$$

Let $\hat{\underline{\theta}}_{(i)}$ denote the parameter values obtained when $(\underline{x}_i, y_i)$ is omitted from the training data. The crux of Craven and Wahba's exclusion lemma is to show that $\hat{\underline{\theta}}_{(i)}$ can also be obtained by replacing this datum with $(\underline{x}_i, \hat{y}_{(i)})$ rather than by excluding it from training; and this fact can be used to derive a useful expression for the CV error.

To see how Craven and Wahba's lemma works, consider the fit obtained when $(\underline{x}_i, y_i)$ is excluded from training. Assuming that training has proceeded until the training error is minimised, the gradient of this function must be zero at $\hat{\underline{\theta}}_{(i)}$,

$$\nabla_{\underline{\theta}} E(\hat{\underline{\theta}}_{(i)}) + \lambda \nabla_{\underline{\theta}} J(\hat{\underline{\theta}}_{(i)}) \;=\; 0 \ . \tag{6.53}$$

Now suppose that the *pseudo-datum* $(\underline{x}_i, \hat{y}_{(i)})$ is introduced into the training data and that training is restarted from $\hat{\underline{\theta}}_{(i)}$. If the regularisation penalty does not depend upon the training data, as is true for the weight decay and spline-type penalties discussed earlier, then introducing this new datum will not affect the value of this penalty or its gradient vector. Similarly, since this new datum is fitted exactly by the MLP, a lower fit error is not possible, and so the fit error gradient is still zero. Thus the overall gradient, (6.53), must be zero when training is restarted and so $\hat{\underline{\theta}}_{(i)}$ minimises the penalised error function both when $(\underline{x}_i, y_i)$ is excluded from training and when it is replaced with $(\underline{x}_i, \hat{y}_{(i)})$. Stated in a more useful form, this means that the same regression fit will be obtained for both sets of training data.

This relationship between exclusion and perturbation can be used to calculate the CV score using only a single fit to the complete training data set. Perturbing y_i by Δy_i will cause $\hat{y}_i$ to change by

$$\Delta \hat{y}_i = h_{ii} \Delta y_i \ . \tag{6.54}$$

Now for the fit obtained using all the data, consider the effect of perturbing y_i to $\hat{y}_{(i)}$. Craven and Wahba's lemma shows that if the model is re-fitted then the estimate of the regression function at this point must also become $\hat{y}_{(i)}$. Substituting these changes back into (6.54) gives

$$\hat{y}_{(i)} - \hat{y}_i = h_{ii}(\hat{y}_{(i)} - y_i) \ , \tag{6.55}$$

which can be re-written by adding y_i to each side and re-arranging as

$$y_i - \hat{y}_{(i)} = \frac{y_i - \hat{y}_i}{1 - h_{ii}} \ . \tag{6.56}$$

This expression can be substituted into (6.51) to calculate the L_p CV error directly

$$CV_1(\lambda) = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{1 - h_{ii}} \right|^p \tag{6.57}$$

without the need for n additional CV fits.

This formula is exact for linear estimators because the leverages depend only on $\{\underline{x}_i\}$, which are fixed, and so (6.54) is exact for any response perturbation. However, this does not disallow the use of this method for estimators for which (6.54) is only approximately true for all perturbations of interest. It was thus proposed that (6.57) could be used to compute the CV score for MLP regression estimators by using the tangent plane or Jacobian leverages[17]. The accuracy of this method for computing the true CV error

[17] Shortly after this work was completed, I discovered that similar work using tangent plane leverage had been reported by Wahba[184]. My work extends areas of Wahba's work by comparing the tangent plane leverages with the more accurate Jacobian leverages, and by considering practical issues in the use of this method for MLP regression and for non-LS regression.

would be limited by the accuracy of the MLP fit leverages for estimating changes in the fit due to large perturbations.

A brief note on using Bishop's Tikhonov penalty with the fast CV estimator

Since Craven and Wahba's method requires the gradient of the error function to be unchanged after introducing the pseudo-datum, the Tikhonov smoothing penalty cannot be used in the form presented in chapter three. The reason for this is that the weighting function will change as data are excluded from training, causing the penalty gradient to depend on the training data. If it is wished to use the Tikhonov penalty, then a fixed weighting function should be used, such as the empirical density function of the complete training data set.

6.7. Relationship to other work

Two other fast estimators of the cross-validation error have been reported in the MLP literature, namely Moody's generalised prediction error (GPE[74]) and Liu's use of one-step updating methods to compute the cross-validation score[185]. This section discusses these estimators and makes some simple comparisons between them and the leverage-based cross-validation error estimator.

The generalised prediction error

The GPE is a generalisation of Mallow's C_l statistic[186] which Moody obtained from a truncated Taylor series expansion of the mean square prediction error for a MLP regression model. The prediction error estimate is

$$GPE = E_{train} + 2\hat{\sigma}_{eff}^2 \frac{\hat{p}_{eff}}{n} \qquad (6.58)$$

where E_{train} is the training data SSE divided by the number of training data, n. p_{eff} is the trace of the Jacobian leverage matrix, which Moody calls the effective number of parameters in the MLP, in analogy with a corresponding term in the C_l statistic which is the number of model parameters.

Moody provides results from a simple experiment with controlled data which demonstrate the effectiveness of the GPE estimator[74]. However, this estimator does have some potential shortcomings.

One problem is that use of the GPE requires an estimate of the true response error variance, $\hat{\sigma}_{eff}^2$. Obtaining this may be difficult, such as when the fit is always biased due to an important predictor variable being absent from the data[64, 187].

Another issue is that the GPE uses the mean square training error and so is sensitive to outliers. This may lead to a MLP of the wrong complexity being used if the GPE is used for model complexity selection. It is not valid simply to replace E_{train} with a more robust error measure unless it can be shown that this results from truncating the Taylor series expansion of a proper robust prediction error estimate.

One-step estimation of the cross-validation error

Liu estimates the cross-validation error for a MLP regression model by first obtaining one-step estimates, $\underline{\hat{\theta}}_{(i)}$, of the MLP parameters when each datum is excluded from training, and then using these to compute the $\hat{y}_{(i)}$ required to compute the CV score. One-step estimates are so-called because they use only one iteration of a training algorithm to estimate the change in the parameter values rather than fully re-training each time a datum is excluded[188, 189].

Liu uses a slight variant of the Newton one-step estimator

$$\underline{\hat{\theta}}_{(i)} \approx \underline{\hat{\theta}} - \left[\nabla^2_{\underline{\theta}} E_{(i)}(\underline{\hat{\theta}})\right]^{-1} \nabla_{\underline{\theta}} E_{(i)}(\underline{\hat{\theta}}) \tag{6.59}$$

to compute the MLP cross-validation score, and has shown that the one-step estimate of the mean square prediction error is asymptotically equivalent to Moody's GPE as the number of training data increases[190]. $E_{(i)}$ is the value of the training error function when datum i is omitted.

Computing the cross-validation score using one-step estimation is more computer intensive than the fast CV method suggested here. However, there are many ways to accelerate this procedure, such as using the less accurate first-order Gauss-Newton method for the one-step estimation[188], using matrix inverse updating techniques to reduce the cost of computing the matrix inverses for each i[188, 189], and using the faster one-step estimators which can be derived when LS estimation is used[189, 191, 192].

Comparison of the different prediction error estimators

It is difficult to compare the properties of the GPE, Liu's estimator and the cross-validation estimator developed here from theory alone, and this issue was not considered sufficiently important to merit experimental investigation for now. However, for large n, it is possible to show a relationship between the GPE and a close relative of leave-one-out CV known as generalised CV (GCV)[38, 183].

The GCV estimate of the mean square prediction error is

$$GCV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{r_i}{1-\bar{h}}\right)^2 \tag{6.60}$$

which is similar to the CV estimate of this error, but here the individual leverages are replaced by the mean leverage

$$\bar{h} = \frac{tr(H_J)}{n}. \tag{6.61}$$

Using the MacClaurin expansion for $(1 - x)^{-1}$, the GCV score can be written as

$$GCV = \frac{1}{n}\sum_{i=1}^{n} r_i^2 \left(1 + \bar{h} + \bar{h}^2 + \cdots\right)^2 . \tag{6.62}$$

For a given MLP the mean leverage decreases as n increases[18] and (6.62) can be further simplified for small $\bar{h}$ to

$$GCV \approx \frac{1}{n}\sum_{i=1}^{n} r_i^2 + \frac{2\bar{h}}{n}\sum_{i=1}^{n} r_i^2. \tag{6.63}$$

The first term in (6.63) is the mean training set SSE, which also provides an estimate of the noise variance, σ^2, under the assumption that the fit is unbiased. Substituting these into (6.63) and re-arranging yields

$$GCV \approx \frac{SSE}{n} + 2\sigma^2 \frac{tr(H_J)}{n} \tag{6.64}$$

which is Moody's GPE with the noise variance estimated from the fit.

This relationship clearly cannot be used to directly compare Moody's GPE and the CV estimator without further understanding of the relationship between the GCV score and the CV score for MLPs. However, GCV and CV often behave similarly for linear estimators[38], and this suggests that the CV and GPE error estimates may behave similarly for some problems.

6.8. An experiment to test the validity of the fast CV estimator

Some simple tests were performed to check that the fast CV estimator could provide reasonably accurate estimates of the true leave-one-out CV error. This section discusses one of these experiments.

The regression function to be modelled is based on Duncan's chemical reaction rate model[193]

$$\mu(x) = 0.90235\left(e^{-4.2324x} - e^{-8.8328x}\right). \tag{6.65}$$

A set of twenty four training data was generated by sampling the regression function four times at x = 0.025, 0.05, 0.1, 0.2, 0.4 and 0.8 and adding response errors from the Gaussian distribution N(0, 0.05). A MLP with four hidden units was trained to estimate $\mu(x)$ by using LS estimation with the spline regularisation penalty discussed in chapter three. The model curvature was sampled for the cubic spline penalty at twenty points

[18] Since a single datum exerts less influence on the fit as the number of data increases.

distributed uniformly from $x = 0$ to $x = 0.95$ at spacings of 0.05 and λ was set at 0.00002 since this gave a visually good, smooth fit. The training data, $\mu(x)$ and the MLP fit are shown in figure 6.6.

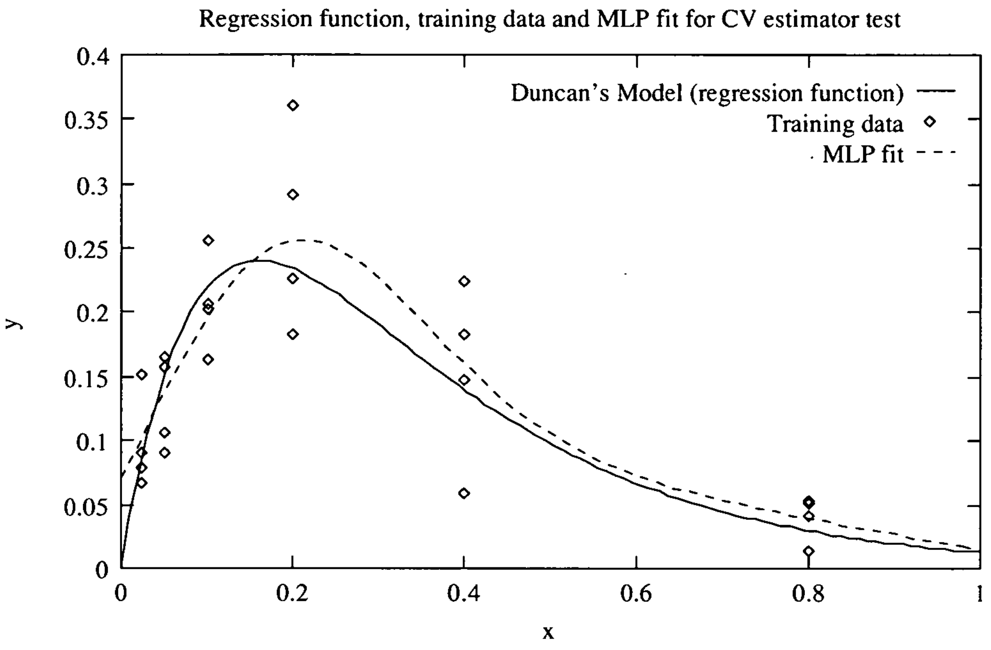


Figure 6.6: Training data and fit for the cross-validation estimator test. The fit was obtained using LS estimation with cubic spline regularisation.

The fit residuals and appropriate leverages were substituted into equation (6.51) with $p = 2$ to calculate both the Jacobian and tangent plane CV MSEs for the MLP. Since the data set and MLP model used in this test were small, it was computationally feasible to re-train the model the twenty-four times required to compute the true CV score. Table 6.1 gives the estimated and measured cross-validation errors, along with two true mean square prediction errors. The local prediction MSE is the MSE between the fit and the regression function at the points where the training data exist. The global prediction MSE is the error over the whole fit shown in figure 6.6.

Training set error	0.00199
Tangent Plane CV score	0.00303
Jacobian CV score	0.00305
True CV score	0.00306
Local true prediction MSE	0.00334
Global true prediction MSE	0.00377

Table 6.1: Training set MSE, true prediction MSE and different CV MS prediction error estimates for the cross-validation estimator test.

It can be seen that the true and estimated CV scores agree very well. The Jacobian estimate of the CV error is slightly more accurate than the tangent plane estimate, as would be expected, but the difference is small in this case.

It can also be seen that the CV errors give better estimates of the local and global prediction errors than the training error does. Testing that this is generally true, or that the CV error is a consistent estimate of the local prediction error would, of course, require Monte Carlo repetition of this experiment with changing response errors. The local error estimate is most accurate because it does not require the fit accuracy between the training data to be estimated.

This simple experiment confirms that the fast, approximate CV estimator can give accurate estimates of the true CV error and that the CV error can give a reasonable estimate of the true prediction error.

6.9. Application of the CV estimator to curl modelling

The previous example demonstrated the validity of the fast CV method for a relatively simple regression problem. This section describes the application of the technique to a more difficult, real problem: the development of the curl model.

The curl problem provides an excellent illustration of the usefulness of the fast CV estimator and also raises a number of practical issues. Details of the curl data are not required to understand this example.

MLPs with 4 hidden units were used to estimate the curl regression function. Since outliers were expected in the data, $L_{1.2}$ and $L_{1.5}$ estimation were used in addition to LS estimation to fit the MLPs. To prevent overfitting when training to completion, the standard weight decay penalty discussed in chapter three was used with λ = 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1, 2, 3. Local minima in the training error were encountered occasionally when using the smaller values of λ and so all MLPs were trained a further ten times with different initial parameters, to provide a representative result for each λ.

Only 318 training cases were available to fit the model. The data set included 28 predictor variables and hence is an example of a very sparse data set, for which it is desirable to use as much data as possible for training. Nevertheless, some means of testing validity the CV scores was required and so the data were split into sets of 253 training data and 64 validation data.

Results from the MLPs trained using LS estimation

Figure 6.7 shows the mean absolute training, validation and Jacobian CV errors for the MLPs trained using LS estimation with weight decay. The mean absolute error (L_1 error) was used instead of the mean square error because it is less sensitive to outliers. The errors shown in this figure are the mean errors for the 8 MLPS with the lowest prediction errors out of the 10 MLPs trained for each λ; trimming was performed to filter out occasional large errors from MLPs which had become trapped in poor local minima.

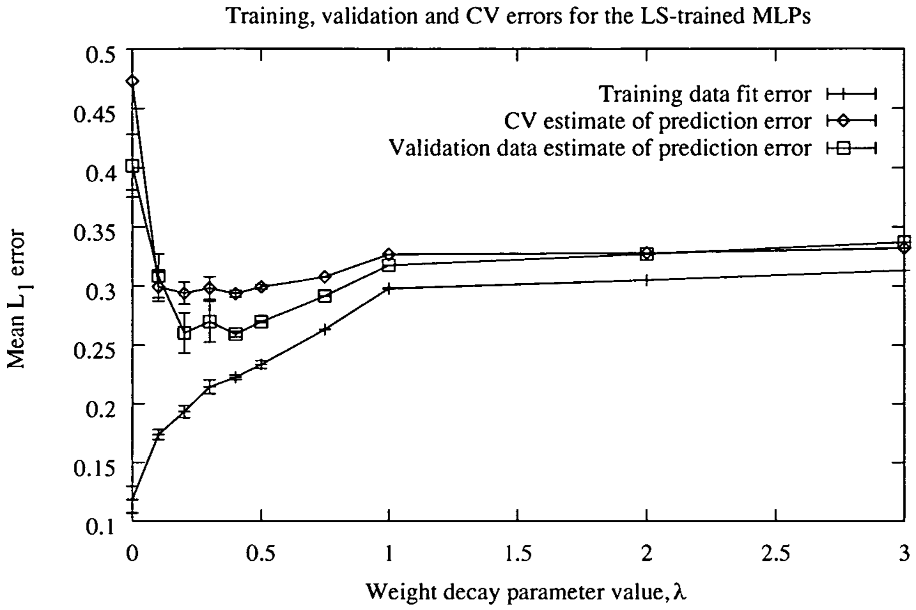


Figure 6.7: Mean absolute training, validation and CV errors for the MLPs trained using LS estimation with weight decay. The error-bars shown are the nominal 95% confidence intervals given by Student's-t for the 8 errors averaged for each λ.

Both the CV and validation data estimates of the prediction error indicate the same trends as λ is varied, and with both estimates the minimum occurs near $\lambda = 0.2$ to $\lambda = 0.4$. As expected, the training set error decreases monotonically as λ decreases (complexity increases) and the prediction errors show that the minimum training error at $\lambda = 0$ corresponds to overfitting.

The significant result of this experiment is that had there been insufficient data for a validation set, the prediction errors could still have been estimated using the fast cross-validation estimator and a suitable value for λ could be chosen. Thus the usefulness of the CV estimator for real problems with sparse data is confirmed.

The tangent plane leverages were also used to estimate the CV error, but did not give as good prediction error estimates as did the Jacobian leverages. Figure 6.8 shows the training, validation and tangent plane CV errors for the LS-trained MLPs.

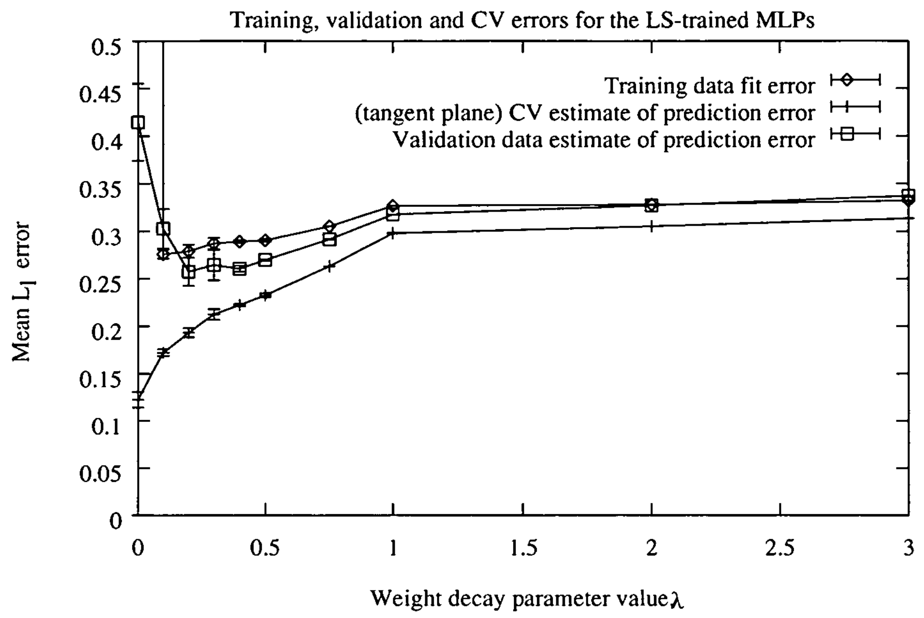


Figure 6.8: Mean absolute training, validation and tangent plane CV errors for the MLPs trained using LS estimation with weight decay.

The tangent plane CV errors estimate the magnitude of the prediction error reasonably well, but do not follow the shape of the validation error curve for small λ. The minimum CV error occurs for $\lambda = 0.1$, but the validation error curve indicates that overfitting occurs for this value of λ. This demonstrates the superiority of using Jacobian leverage instead of tangent plane leverage.

Results from the MLPs trained using $L_{1.5}$ and $L_{1.2}$ estimation

The results obtained using $L_{1.5}$ and $L_{1.2}$ estimation highlighted some weaknesses of the CV estimator, both in general and for L_p-Norm estimation in particular. The results of these experiments are presented in this section and the subsequent sections discuss the various practical issues which they raised.

Figure 6.9 shows the results obtained using $L_{1.5}$ estimation with weight decay. The CV error estimate follows the validation error quite closely and both errors are minimised for almost the same value of λ. Thus a MLP of suitable complexity could again have been selected had there been insufficient data for model validation. However, there is slight but noticeable downward bias in the CV error estimates for $\lambda < 0.4$.

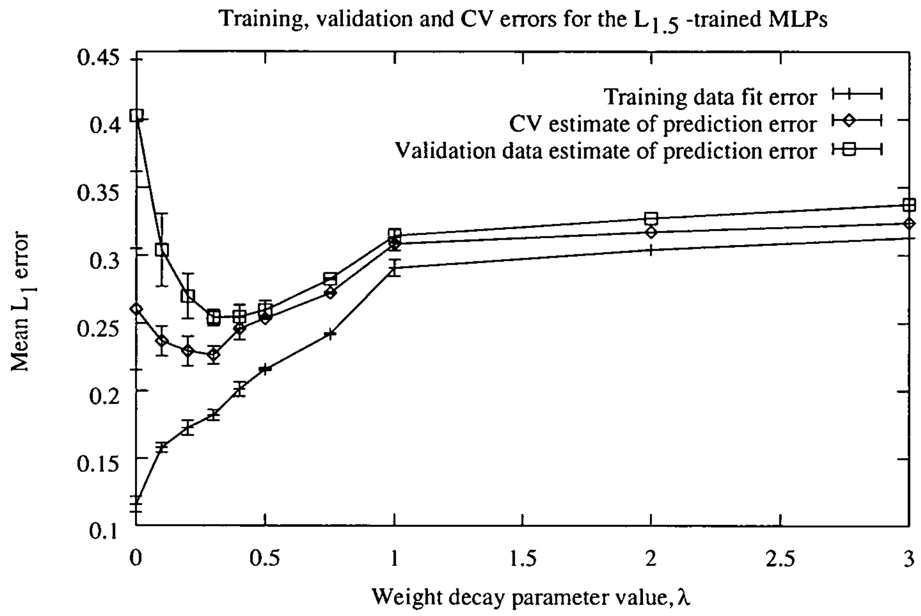


Figure 6.9: Mean absolute training, validation and CV errors for the MLPs trained using $L_{1.5}$ estimation with weight decay.

This downward bias for small λ became more pronounced when $L_{1.2}$ estimation was used, as shown in figure 6.10.

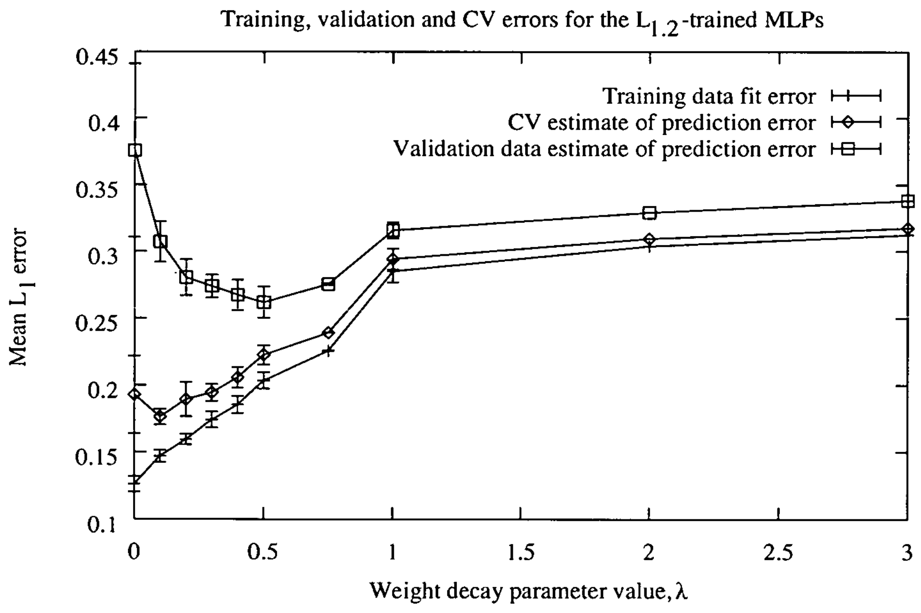


Figure 6.10: Mean absolute training, validation and CV errors for the MLPs trained using $L_{1.2}$ estimation with weight decay.

Here the CV error curve follows the training error curve more closely than the validation error curve and has a minimum near $\lambda = 0.1$. The validation error indicates, however, that the MLP is overfitted for this value of λ and so a MLP with too high complexity would have been chosen using the CV error.

It was not possible to compute the true CV errors for these MLPs because of the enormous amount of computer time that this would have required. Consequently, it was not possible to conclude whether the downward bias of the CV scores for the $L_{1.2}$- and $L_{1.5}$-trained MLPs is due to:

- CV being a biased estimator of the prediction error when using $L_{1.2}$ or $L_{1.5}$ estimation with weight decay.
- The approximations made by the fast CV estimator being invalid, resulting in the fast CV error being a poor estimate of the true CV error.

Further investigation suggested that the second of these issues was likely to be responsible for the fast CV error bias.

6.10. Limitations of the fast CV estimator

There are several circumstances under which the fast CV estimator gives inaccurate estimates of the true CV score, some of which occurred in the previous example using the curl data. These are:

- Training is not complete; that is the error function has not been minimised.
- The leverages are not valid over the range required by Craven and Wahba's method.
- Certain types of ill-conditioning occur.

These are discussed in the next sections, which also consider how their occurrence can be diagnosed and remedied if possible. It should be noted that the other prediction error estimators discussed earlier also suffer from these problems.

6.10.1. Training to completion

If training is not complete then the CV error estimate can be inaccurate because of:

- Bias in the fit due to the fact that the fitting process has not completed.
- The Jacobian and tangent plane leverages are incorrect due to violation of the assumption that $\underline{\hat{\theta}}$ minimises the training error.

In practice I found that the second of these is almost always the most important source of error when training is incomplete. Incomplete training often leads to spurious negative leverages and, more insidiously, leverages which look reasonable, but which differ significantly from those obtained when training is completed. The differences between these leverages can be quite large even when training is stopped close to the minimum of the error function.

Fortunately this problem can be solved easily by allowing training to complete. This can, however, require long training times for some problems, as was the case for the $L_{1.2}$-trained MLPs in the previous example when λ was small.

6.10.2. Leverage range of validity

When using the MLP leverages to compute the CV error, it is assumed that equation (6.54) gives exactly the change in the fit for all perturbations. However, this may not be true when large response and fit perturbations are required.

To see why this is so, suppose that a large response perturbation, Δy_i, is applied as a series of small perturbations and that the MLP is re-trained to give an updated leverage after every small perturbation. If the leverage increases after each re-fit then

$$\Delta\hat{y}_i = h_{ii}\Delta y_{ii} \tag{6.66}$$

underestimates the overall change in the fit because the leverage increase as the fit changes is not accounted for. Similarly, if the leverage decreases after each perturbation, then the overall fit perturbation will be overestimated by assuming that the leverage for the initial fit is valid over the full perturbation range.

Underestimation of $\Delta\hat{y}_i$ caused by leverage increase can explain the downward bias of the CV error estimates observed for the $L_{1.2}$- and $L_{1.5}$-trained MLPs in the previous example. Figure 6.11 shows a scatterplot of leverage versus residual for one of the LS-trained MLP curl models.

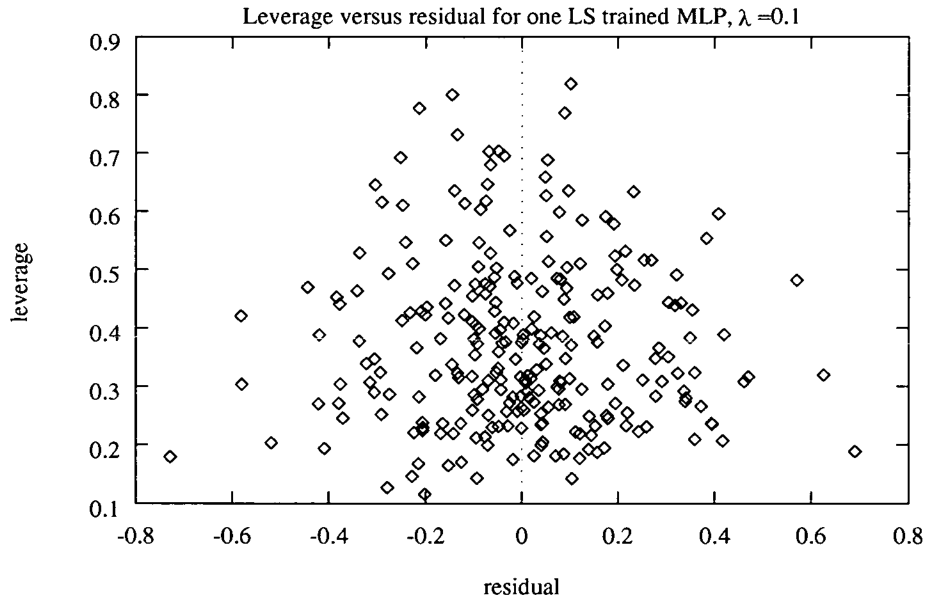


Figure 6.11: Scatterplot of leverage against residuals for one of the LS-trained MLPs with $\lambda = 0.1$. No strong relationship between the leverages and residuals is evident.

The typical LS leverages appear to become slightly smaller as the fit residual increases, but no strong relationship between leverage and residual is evident. Thus a given datum's leverage is not likely to change much as the response is perturbed towards the fit, and so the fast CV error estimator should estimate the true CV error reasonably well.

Figure 6.12 shows a scatterplot of leverage against residual for one of the $L_{1.2}$-trained MLPs.

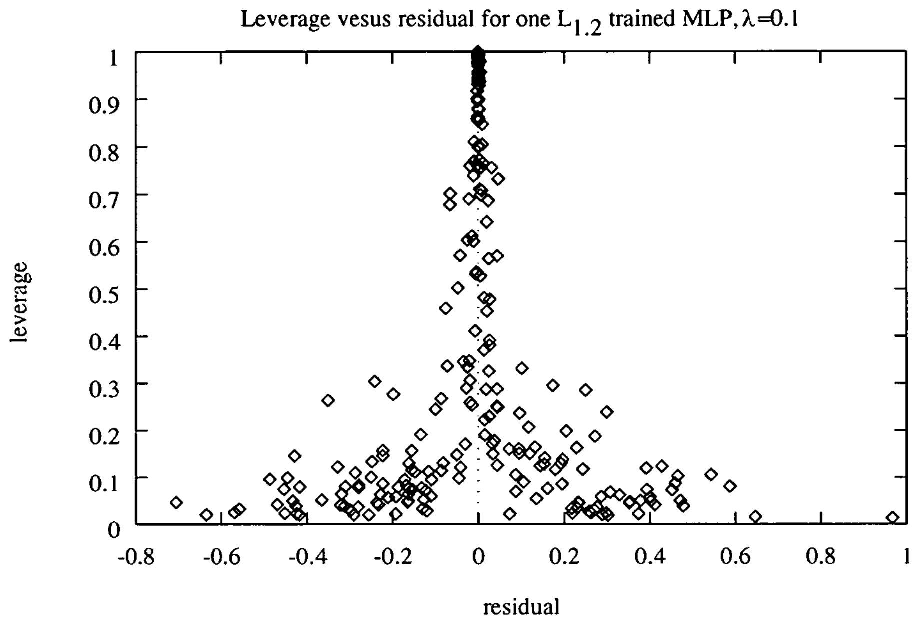


Figure 6.12: Scatterplot of leverage against residuals for one of the $L_{1.2}$-trained MLPs with $\lambda = 0.1$. Leverage depends strongly on the residuals, with the largest leverages occurring for the data with the smallest residuals.

The leverages now depend strongly on the fit residuals with the largest leverages occurring for the data with the smallest residuals. This is another example of the high sensitivity of L_p-Norm estimators to data with small residuals when $p < 2$.

Craven and Wahba's step of perturbing each datum towards the fit reduces the fit residual towards zero. As figure 6.12 shows, this increases the leverage as the residual decreases and so the fit leverage underestimates the effective leverage for this step. Since the leverages are too small to estimate the real change in the fit, the terms

$$\frac{1}{1 - h_{ii}} \tag{6.67}$$

in (6.57) are also too small. Consequently, the residuals are not inflated by large enough factors when computing the CV error and so it will be biased downwards.

If weight decay is used during training then the leverages decrease towards zero as λ increases. This has been confirmed by experiment and can be seen intuitively by noting that as λ increases the weight decay penalty dominates the training error and so the influence (and hence leverage) of the training data in determining the fit decreases[194]. Since (6.67) becomes more sensitive to errors in the leverage as $h_{ii} \rightarrow 1$, the bias caused by underestimating the effective leverage will be most severe for fits with larger typical leverages. Consequently, the largest downward CV error bias would be expected for smaller λ and this is what was observed.

Reducing the prediction error estimate bias for non-LS estimators

A conclusion that can be drawn from the work just described is that the fast CV estimator will be biased when using non-LS estimators. This is because only the LS estimator leverages are (to a first order) independent of the size of the residuals. Since it was desired to use both fast CV and robust training for the curl modelling problem, some methods to address this problem were considered:

- Investigate the relationship between leverage and the estimator shift sensitivity in the hope that this work might yield leverage correction factors suitable for cross-validation.
- Consider whether other prediction error estimators such as the GPE might perform better than the fast CV estimator for non-LS problems.
- Use least trimmed squares (LTS) estimation with a small trim to resist outliers.

The first of these options was not pursued because it could not be guaranteed that this work would prove fruitful in the time available.

The second option was also dismissed because neither the GPE or Liu's error estimator can be expected to perform much better that the fast CV estimator for non-LS estimators. This is because the accuracy of these estimators relies on the error function being locally quadratic near $\underline{\hat{\theta}}$. The GPE makes this assumption by truncating the Taylor expansion of the prediction error, and the Newton one-step estimator used by Liu is most accurate for near-quadratic error functions[189].

LTS leverages are relatively independent of the residual magnitudes because LTS estimation is effectively LS estimation using a reduced data set. However, as explained in chapter five, this method is more computer-intensive than L_p-Norm training and may also be more susceptible to overfitting. Thus additional care is required when using LTS estimation to achieve resistance to outliers and valid CV error estimates.

The issue of leverage range of validity and its affect on non-LS estimators clearly requires further work to assess which estimators and methods offer a good compromise between the accuracy of the fast CV estimate and other important issues such as computer time required to train the MLP and ease of avoiding under- and overfitting.

For the purpose of developing the curl model, the adequacy of the fast CV method when LS training is used was considered sufficient to make the method useful even when using non-LS training. This is because a good correlation between the LS CV and validation prediction error estimates gives a good indicator of whether the validation data is sufficient for validating non-LS trained MLPs. Thus the fast CV method can be used to address the difficult problem of deciding which portion of the data should be reserved for validation. Non-LS fitting can be used once the adequacy of the validation data has been confirmed.

6.10.3. Problems due to ill-conditioning

Ill-conditioning-related problems do not affect the validity of the fast CV estimator in the same way as does going beyond the leverage range of validity. They do however affect the number of significant digits which can be expected in the CV error estimate. In very ill-conditioned problems, the CV estimate can be useless (no significant digits) due to accumulation of numerical errors.

Two types of ill-conditioning were encountered regularly during this work. These are:

- Ill-conditioning due to overparameterisation (usually caused by using a MLP with too many hidden units).
- A type of small-residual ill-conditioning which is specific to the L_p-Norm estimators considered here.

These are discussed below. A simple technique for greatly improving the numerical condition of most MLP regression problems is also discussed.

Overparameterisation and nonidentifiability

A parametric model is overparameterised if it contains more parameters than are necessary to approximate the regression function being estimated[35]. A simple example is using LS estimation to fit a cubic polynomial to 3 data points; there is an infinite number of possible fits with the same error (zero in this case). Since the training error does not have a unique local minimum under these conditions, the model parameters are said to be nonidentifiable.

In near-nonidentifiable problems, where the error function has a wide, shallow minimum, the observed Fisher information matrix

$$\textit{Observed Fisher information} = \nabla^2_{\underline{\theta}} E(\hat{\underline{\theta}}) \tag{6.68}$$

is ill-conditioned[32, 195]. $E(\underline{\theta})$ is the training error function.

The inverse of the information matrix is the central term in the Jacobian leverage expression (6.45). If the information matrix is ill-conditioned then a serious loss of numerical precision may occur during its inversion. Ill-conditioning can be identified by examining the condition number of the information matrix[195-197]. Ill-conditioning due to identifiability problems can occur in MLP regression when overfitting occurs, when collinearities exist between the predictor variables, or when collinearities exist between the hidden unit outputs. A detailed discussion of sources of ill-conditioning in MLP regression is given in[198].

If the Jacobian leverages are affected by ill-conditioning then the tangent plane leverages will also be affected. This is because the tangent plane leverages are based on expected information while Jacobian leverages use observed information[166]. If one information matrix is ill-conditioned then the other is too.

Near division by zero in small residual problems

This type of ill-conditioning occurs for L_p-Norm estimation problems when $p < 2$ and some of the residuals are almost zero after training[199, 200]. For simplicity, the case where only one very small residual occurs will be considered first. It will also assumed that λ is zero and that B_p is negligible in (6.45).

When $p < 2$ the exponents of the diagonal elements of D in (6.46) are negative. If one residual is almost zero then the corresponding element in D is obtained by dividing by a small number and so will be very large relative to the other elements of this matrix. Making the substitution $Q = D\hat{Z}$ in (6.45), $Q^T Q$ must be nonsingular to allow the leverages to be computed. If the first residual, r_1, is almost zero then the first diagonal element of D may be very large and this will inflate the first row of Q. Using b and s to denote big and small matrix elements, Q can be written in the very approximate form

$$Q \approx \begin{pmatrix} b_1 & b_2 & \cdots & b_w \\ s & s & \cdots & s \\ . & . & \cdots & . \\ s & s & \cdots & s \end{pmatrix} \tag{6.69}$$

where w is the number of MLP weights and biases. Ignoring terms of order less than b^2,

$$Q^T Q \approx \begin{pmatrix} b_1\begin{pmatrix} b_1 \\ b_2 \\ . \\ b_w \end{pmatrix} & b_2\begin{pmatrix} b_1 \\ b_2 \\ . \\ b_w \end{pmatrix} & \begin{matrix} \cdots \\ \cdots \\ \cdots \\ \cdots \end{matrix} & b_w\begin{pmatrix} b_1 \\ b_2 \\ . \\ b_w \end{pmatrix} \end{pmatrix} \tag{6.70}$$

which is rank 1 and hence singular.

Clearly many of these assumptions will not be true in practice. However, it was found that when $p < 2$ and a near-zero residual occurred which was significantly smaller than the other residuals, the singular value decomposition (SVD) of the information matrix typically exhibited one singular value (SV) which was significantly larger than the other SVs. This is consistent with a tendency towards singularity with rank 1. With large data sets the effective rank is usually greater than 1 because there are small residuals of varying relative magnitude and so the largest SV is usually not significantly larger than the next largest SVs. The information matrix is still ill-conditioned, however, if the ratio of the largest and the smallest singular values is large.

An example of ill-conditioning and how to improve conditioning

Figure 6.13 shows the conditioning of the MLPs trained to estimate the curl function in the previous example. Each point in the graph shown is the median condition number for the 10 MLPs trained for each λ.

This graph illustrates both types of ill-conditioning discussed previously. All the MLPs are ill-conditioned when $\lambda = 0$ due to overfitting-induced parameter identification problems. The $L_{1.2}$-trained MLPs are also notably more ill-conditioned than the other MLPs due to small-residual-induced ill-conditioning. For the IEEE 754 double precision arithmetic used to train the MLPs and compute the leverages, there is a danger of losing all the significant leverage digits when the information matrix 2-Norm condition number is greater than about 10^{12}.

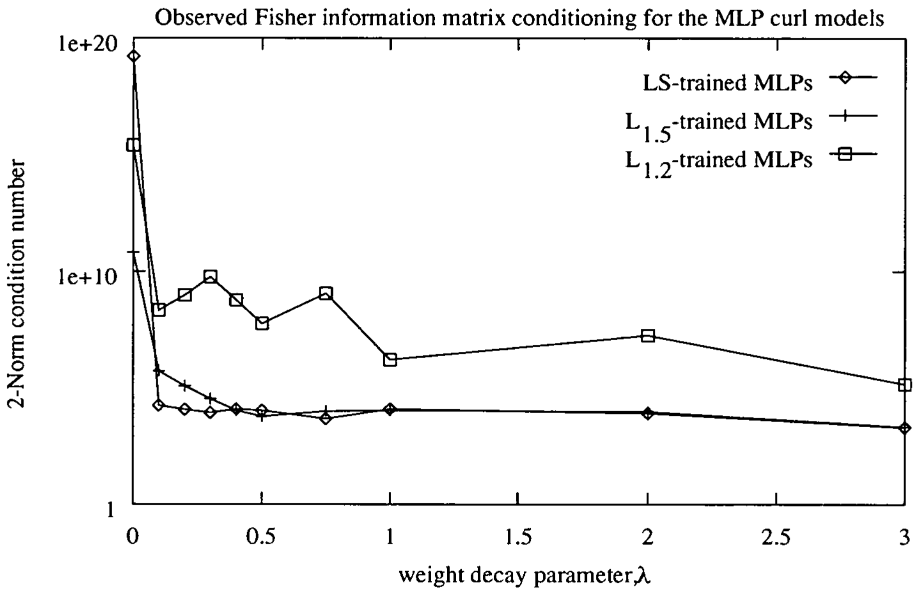


Figure 6.13: Median condition numbers for the MLPs trained to estimate the curl function.

Figure 6.13 also indicates a very useful property of weight decay, namely that using even a very small amount of weight decay (less than enough to cause smoothing or prevent overfitting) can improve the information matrix conditioning dramatically. This property of weight decay is well-known in its classical regression counterpart, ridge regression[106, 107]. Thus using a small amount weight decay should be considered even when using other methods for complexity control, such as the spline smoothing penalties discussed in chapter three. A further benefit of improving the information matrix conditioning is that this can speed-up training when using standard optimisation techniques such as Newton's method[198]. This reduces the time required to train the MLP to completion.

6.11. Summary and conclusions

The first sections of this chapter looked at

- how to compute leverages for MLP regression models, and
- how the fit leverages can be used to diagnose local overfitting.

In the absence of well-tested, non-computer-intensive methods for generating MLP fit confidence intervals, examining the leverages allows unreliable sections of the fit to be found quickly and suitable care to be exercised when using the fit to estimate the regression function in these regions.

The most important results of this work were discussed in the second part of the chapter, which looked at how the leverages could be used to reduce dramatically the amount of computation needed to compute the cross-validation estimate of the MLP generalisation ability. This method is preferable to data-splitting validation for problems involving sparse training data, such as the curl problem, because it allows all the available data to be used for training. Given that maximising the amount of training data is one way to address the ever-important problem of avoiding overfitting, the importance of the fast CV method is obvious.

The development of a simple curl model was used to illustrate both the usefulness of the fast CV estimator and also some situations where the estimator can give poor prediction error estimates, namely:

- When training is not complete.
- When the leverage range of validity is exceeded. This is a serious problem for most robust estimators.
- When ill-conditioning occurs.

Some methods for improving the CV method performance when these occur were also considered.

In the context of the curl modelling problem, the key result of this work was the demonstration that the fast CV method could be used to estimate the CV error when using LS training. Even though the problems associated with using the fast CV method with non-LS estimators have yet to be resolved adequately, a good correlation between the validation and CV errors when using LS training indicates that the validation data is likely to be adequate for estimating the prediction error for non-LS fits. The LS fast CV method thus provides a method for addressing the difficult problem of determining how much validation data to reserve.

Chapter 7

Developing the curl model

7.1. Introduction

The curl modelling problem was introduced in chapter four. This problem involves attempting to model the degree of curl exhibited by a coated paper as a function of other process variables measured during the coating process. In chapter four, was stated that some problems, such as how to deal with outliers in the training data, required investigation before developing the MLP curl model. These investigations and the further investigations which they stimulated were discussed in chapters five and six. This chapter now returns to the development of the curl model.

While it has been common practice in MLP regression to simply 'throw the data at a MLP and see what happens', a more systematic approach to model development was considered appropriate. The model development was structured as follows:

- Before performing any modelling, it was first necessary to select which process variables to include in the model as predictor variables. Section 2 discusses which variables where chosen and why.
- Before engaging in the relatively time-consuming process of developing a MLP curl model, it was considered prudent to check first for signs of a nonlinear regression function within the data. Section 3 discusses the preliminary modelling work which was performed to check that the MLP was an appropriate model for curl.
- Deciding how to split the available data into training, validation and test sets is a difficult problem. In section 4, the fast cross-validation method described in chapter six is used to address this problem.
- Section 5 discusses the development of the first MLP curl models.
- Section 6 discusses the testing of the MLP models, including preliminary work on the problem of identifying and removing irrelevant predictor variables from the models, and the results of field trials of one MLP curl model.
- Section 7 gives a summary and presents the conclusions of the modelling work.

7.2. Choosing the model variables

The paper-curl data contained records for 504 rolls of paper. A record contains 40 process variables, all of which were suspected to have some relationship to curl.

The large number of variables relative to the number of records means that this is very sparse data and, to reduce the risk of overfitting, it was decided to limit the number of variables used in the model. The most important issues which affected the choice of predictor variables for the first models were

- missing data,
- collinearity between predictor variables, and
- the use of nominal (non-numeric) values for some variables.

The next sections discuss how these issues influenced the choice of predictor variables.

Missing data

The most immediate variable selection problem was posed by missing data entries. Until methods for dealing effectively with missing data could examined, it was necessary to discard all records with any missing entries.

To maximise the amount of available training data, those variables which had most missing entries were considered for elimination. Unfortunately, the variables with most missing entries (155 missing per variable) were variables which were considered most likely to have a strong influence on curl. Hence it was necessary to discard many records in order to retain these variables in any curl models.

Collinearity between predictor variables

If the value of one predictor variable can be expressed accurately as a function of another variable, then these variables give essentially the same information about the regression function, and so one of them can be eliminated. Such variables are said to be collinear when their relationship is linear, though the term is used here to indicate any strong systematic relationship.

Detecting systematic nonlinear relationships between variables is difficult, but linear and nonlinear monotonic collinearity can be detected easily by examining the Pearson, Spearman and Kendall correlation coefficients[201] for all variable pairings.

Several of the predictor variable pairs had high correlations and were thus earmarked for elimination as collinear variables. However, on scatterplotting these variable pairs, it was decided to retain most of them for two reasons:

- many of the highly correlated variables still exhibited significant independent variability, and

- more commonly, most high correlations were due to the presence of two weakly-correlated data clusters.

Figure 7.1 shows two typical examples which illustrate these points. The left hand plot shows two variables which have a high overall Pearson's correlation, but are not strongly collinear for the larger moisture values. If curl depends on the difference between these variables (quite possible), then eliminating one variable may remove a significant amount of information about the regression function for high moistures. The right hand plot shows two variables which have a high overall Pearson's correlation, but which mostly reside in two weakly-correlated clusters (marked by dashed boxes). The high overall correlation arises simply because two clusters will always lie neatly on a straight line through their centres. Within each cluster there is little correlation between the variables, and so knowing the value of one variable does not allow the value of the other to be predicted accurately.

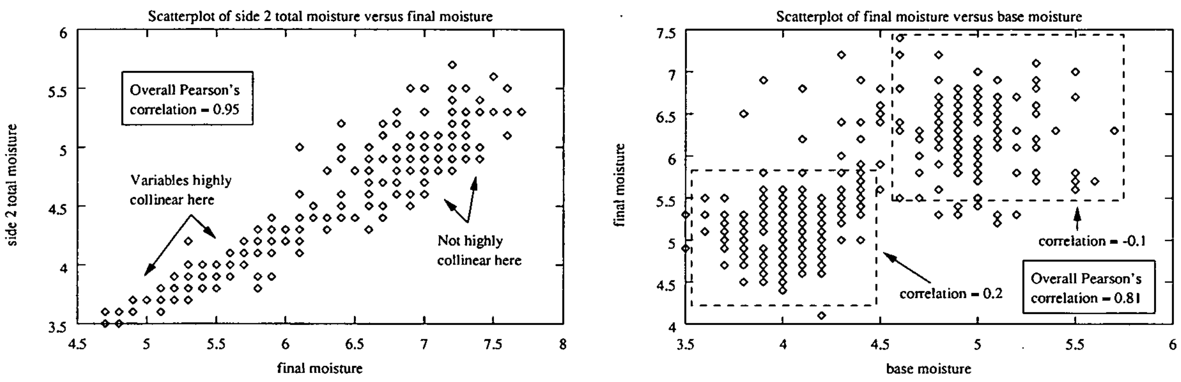


Figure 7.1: Illustrations of how some variables which have high overall correlation coefficients need not be highly correlated for all their values.

After examining the scatterplots for all the highly correlated variables, it was decided to omit only one variable (HW1 refiner setting) on the basis of very high collinearity (Pearson's correlation > 0.99) with another variable (HW2 refiner setting).

Nominal variables

For the first curl models, it was decided not to use the pulp composition data as predictor variables. The reasons for omitting these were

- there were many missing entries, with no record of whether this meant no pulp in the base composition, or un-logged pulp data, and
- these variables must be scored (assigned numerical values) for use in a MLP regression model. Use of a poor scoring method can increase dramatically the nonlinearity of, and hence difficulty of estimating, the regression function.

It was decided that it would be easier to investigate the incorporation of the pulp data into the model after an initial MLP model had been obtained.

Other nominal variables which were omitted were the hot-air and infra-red drier settings (values were 'on' or 'off'). While these were expected to have a significant affect on curl, their values were almost always the same, and so few records were available to assess the effect of changing these variables. It was also considered that more detailed data, such as drier temperature and air-flow would be useful, and so the use of the drier data was postponed until such data could be logged.

Data available after removing variables

After applying the above 3 procedures to eliminate variables, the data available for training comprised 317 complete records, each of which was comprised of 28 process variables. While the low ratio of the number of records to the number of variables may make avoiding overfitting difficult, it was decided not to attempt to remove any further variables until their importance could be assessed through initial model building; that is, to use a step-down variable elimination approach[202].

7.3. Justifying the use of the MLP for the curl model

As discussed in chapter four, it was decided to use the MLP to model curl because

- it was believed that curl would be a nonlinear function of the other measured process variables, and
- the research group was inexperienced in the use of other modelling techniques, such as kernel regression.

While it was believed that a nonlinear model would be required, it is appropriate to confirm this before attempting to develop a MLP model. Two questions asked in this regard were:

- Are there signs of any relationship between curl and the other variables?
- Is the MLP likely to model this relationship better than a simple linear regression model?

The need for an affirmative answer to the first question is obvious. If no signs of a relationship can be found then it is pointless trying to fit any sort of regression model to the data.

The second point is also important because if the regression function is almost linear, as was the case with the first data set which was eventually discarded, then there are good reasons why a MLP should not be used:

- The linear model is simpler and hence easier to interpret using informal methods such as inspection of the parameters, or the formal inference methods which are discussed in most introductory regression texts[35, 105].
- There is a risk of overfitting spurious nonlinearities in the data if the MLP is used.

In response to the second point, it could be argued that the fit complexity could be strongly constrained, using regularisation for example, to prevent overfitting. Two arguments against this are:

- The principle of parsimony: why should a complex model be used where a simpler model will suffice?
- If using the cross-validation method discussed in chapter six, excessive use of regularisation could cause overparameterisation-induced ill-conditioning problems.

7.3.1. Evidence for the need for a nonlinear model

This section answers the two questions posed in the previous section by verifying that data does suggest a systematic relationship between curl and the predictor variables, and that this relationship is nonlinear.

The techniques used are standard methods in linear regression. Both LS and LAD linear fits were used, the former because most formal inference methods assume LS fitting and the latter to check whether any outliers were affecting the LS fit significantly. All available data was used to fit the linear models, no validation and early stopping methods were used.

7.3.2. Confirming the presence of a regression function

An analysis of variance was performed for the LS fit to check the statistical significance of regression[141, 202]. Table 7.1 summarises the results of this test.

Source	SS error	Degrees of freedom	MS error	F-value
Total	92713	316	293	-
Error	49244	288	171	-
Model	43468	28	1552	9.07

Table 7.1: Analysis of variance for LS linear fit to curl data.

The 0.99 critical value for the F-distribution with (28, 288) degrees of freedom is approximately 1.8. The test F-value is much greater than this, giving better than 99% confidence that there is at least an approximately linear relationship between curl and one or more of the predictor variables.

The confidence level is, of course, not exact if the regression function is nonlinear, but this result nevertheless gives a strong indication that there is a systematic relationship between curl and the predictor variables.

Further investigation to determine whether a nonlinear MLP model was more appropriate for this relationship was hence justified. Before devising a more complex curl model however, it was appropriate to check whether a linear fit gave good enough curl estimates.

Quality of the linear fit

Figure 7.2 shows the LAD linear fit estimated curls versus the measured curls.

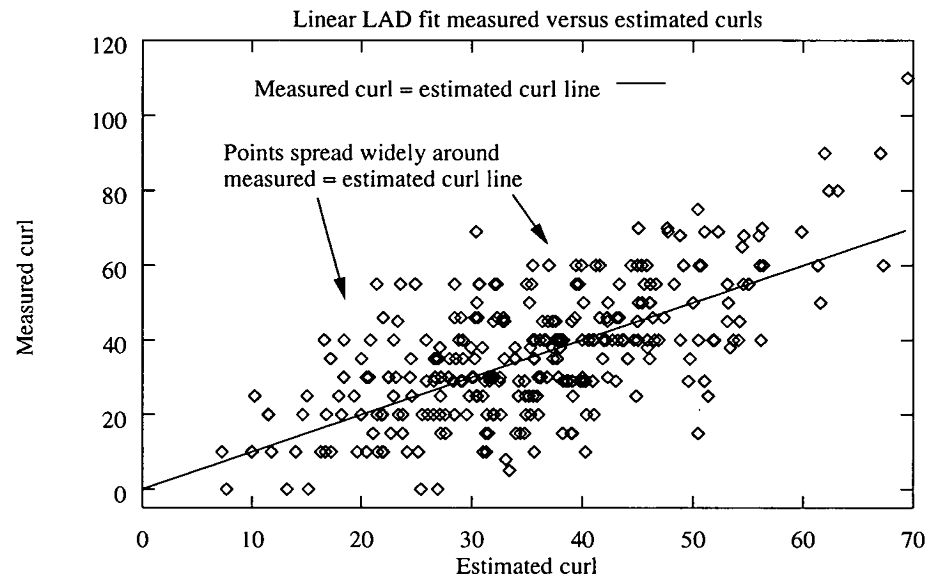


Figure 7.2: Plot of measured versus estimated curl for the LAD linear fit.

While the linear model seems to have captured a major trend in the data, the curl estimates are still spread quite widely around the line indicating where the measured curl is equal to the estimated curl. This wide variation limits the usefulness of the linear model for on-line curl estimation and control because figure 7.2 shows that paper with low curl may have a high predicted curl in some cases, and vice-versa.

There is thus justification for investigating whether a nonlinear MLP curl model will give better curl estimates.

7.3.3. Confirming that the regression function is nonlinear

Having confirmed a systematic relationship between curl and the predictor variables, the next step towards developing a MLP curl model was to check whether this relationship is nonlinear. This is necessary because the inaccuracy of the linear fit may in fact be due to important predictor variables being missing, a situation that cannot be remedied by using a nonlinear model instead.

The standard technique for identifying the need for a nonlinear regression model is to fit a linear model to the data and then to look for systematic nonlinear trends in the fit residuals[39, 128, 143, 203, 204].

Testing for overall nonlinear mis-specification

The regression function linearity was first assessed by scatterplotting the linear fit residuals against the predicted curls. Any statistically significant nonlinear trends on such plots are evidence that a nonlinear model is required.

Figures 7.3 and 7.4 (overleaf) show residual plots for the LS and LAD fits. These figures also show 41-point running mean smooths[44] of the plotted data with approximate 95% confidence intervals for the smooth. The parallel lines with slope -1 which can be seen on these figures are due to quantisation of the measured curls, and have no significance[205, 206].

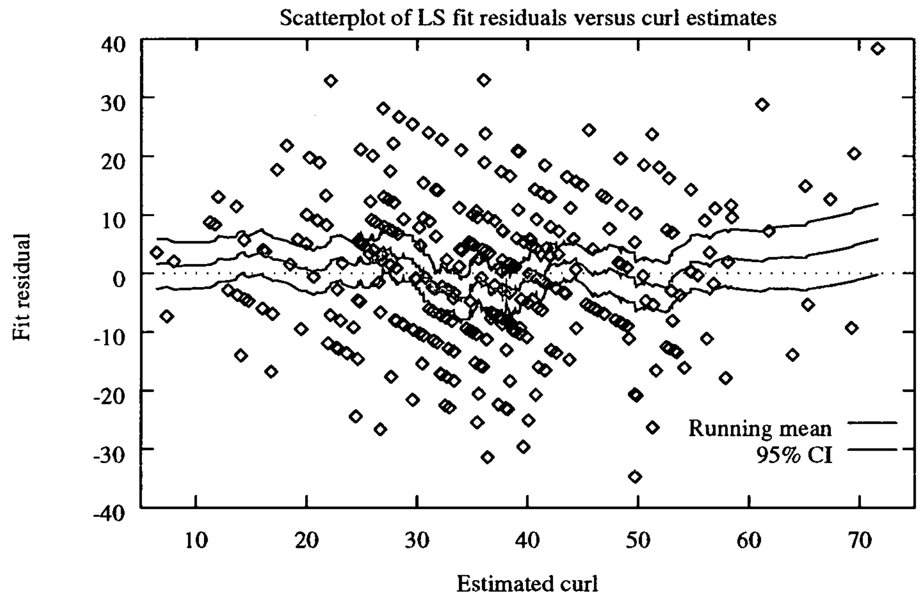


Figure 7.3: LS linear fit scatterplot of residuals versus estimated curl.

Both smooths show nonlinear trends with one or more minima near estimated curls of 30 to 40. These trends are statistically significant because the running means vary by more than the average confidence interval width (i.e. the means for different curl estimates are significantly different). Thus the trends are not likely to be random artifacts of the data, and so give good evidence that a nonlinear model is required.

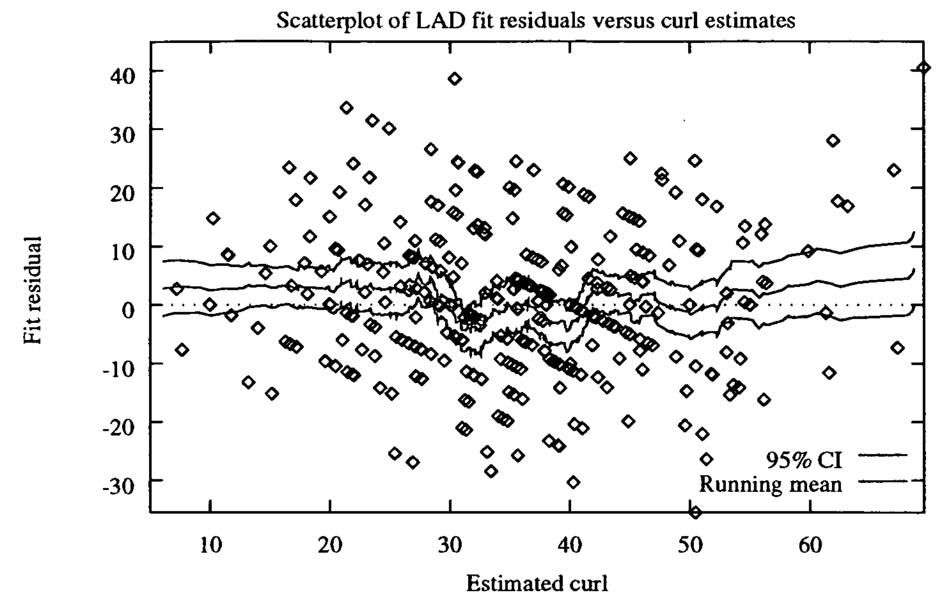


Figure 7.4: LAD linear fit scatterplot of residuals versus estimated curl.

Examining the nature of the nonlinearity further

The technique used in the last section indicated a slight but significant nonlinearity of the curl data regression function. Since that technique is insensitive to many types of nonlinearity, the residuals were also scatterplotted against the predictor variables to give a better indication and stronger evidence of the regression function nonlinearity.

Several of these plots showed statistically significant nonlinear trends in the residuals. Figure 7.5 (overleaf) shows some plots for the LS linear fit. The LAD fit gave similar plots.

The residual bins captioned 'A', 'B' and 'C' have means which alternate between being significantly greater than or less than zero. Thus the captions highlight statistically significant and also quite large bump- and bowl-shaped nonlinear trends in the residuals, giving strong evidence for the need for a nonlinear model.

Interactions between variables can make it difficult to assess the true shape of the regression function using plots such as those shown in figure 7.5. Thus either further analysis is required to specify a suitable parametric model, or a more flexible regression method, such as MLP regression, should be used. The use of the MLP is hence justified for the curl modelling problem.

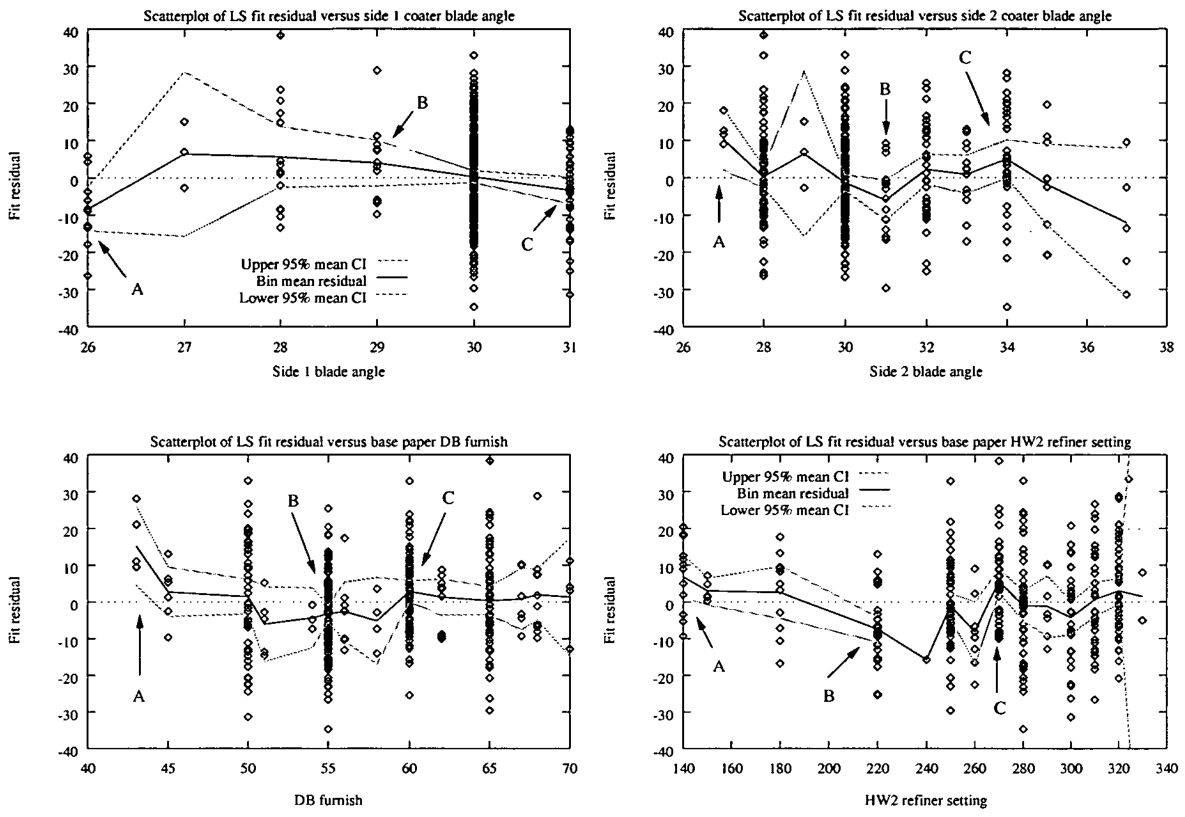


Figure 7.5: Plot of the LS linear fit residuals versus the coater blade angles and some refiner setting predictor variables.

7.4. Creating the MLP training and validation sets

Having decided to use a MLP to model curl, the first issue to be addressed was how to divide the data between training, validation and test sets.

Owing to the limited number of curl records, it was decided not to create a test set to estimate the final model performance. Instead, the curl model accuracy would be assessed by field-trials at the coating plant.

In creating the training and validation data sets, it was necessary to consider

- how to split the data to ensure that the smaller validation set covers the same predictor variable range as the training set, and
- how much data to reserve for model validation.

Splitting for good validation set coverage

Ensuring that the validation set covered the same range as the training set was relatively straightforward because the paper-reel logs were sorted by time of production. Thus long term changes in the machine settings vary smoothly between adjacent records, and so reserving, say, every 5th record for validation should ensure that these changes are sampled and represented in the validation data.

Deciding how much validation data to reserve

Since there are no agreed methods for deciding how much validation data to reserve, it was decided to try reserving 20% of the total data. To confirm that this was sufficient data to obtain useful estimates of the prediction error, the fast LS cross-validation method described in chapter six was also used to estimate the prediction error for some MLP fits. If the validation and prediction error estimates agree reasonably well, then this suggests that there is sufficient validation data.

Training and validation sets were created by reserving every 5th record for the validation set. The validation data sampling was started from the 2nd, 3rd and 5th records to give 3 different training and validation set pairs (called 'A', 'B' and 'C' respectively). For each pair, MLPs with 4 hidden units were used to estimate the curl regression function using LS fitting. Standard weight decay and the modified weight decay discussed in chapter three were used to control the fit complexity. For each value of the complexity control parameter, λ, 10 MLPs were trained and a 2-trimmed average of their prediction errors was taken as the estimate of the true prediction error. Trimmed averaging was used to prevent occasional large error estimates due to local minima from distorting the results.

Figure 7.6 shows average training error, validation and cross-validation errors for some of the MLP fits. The error-bars shown are 95% confidence intervals for each trimmed mean.

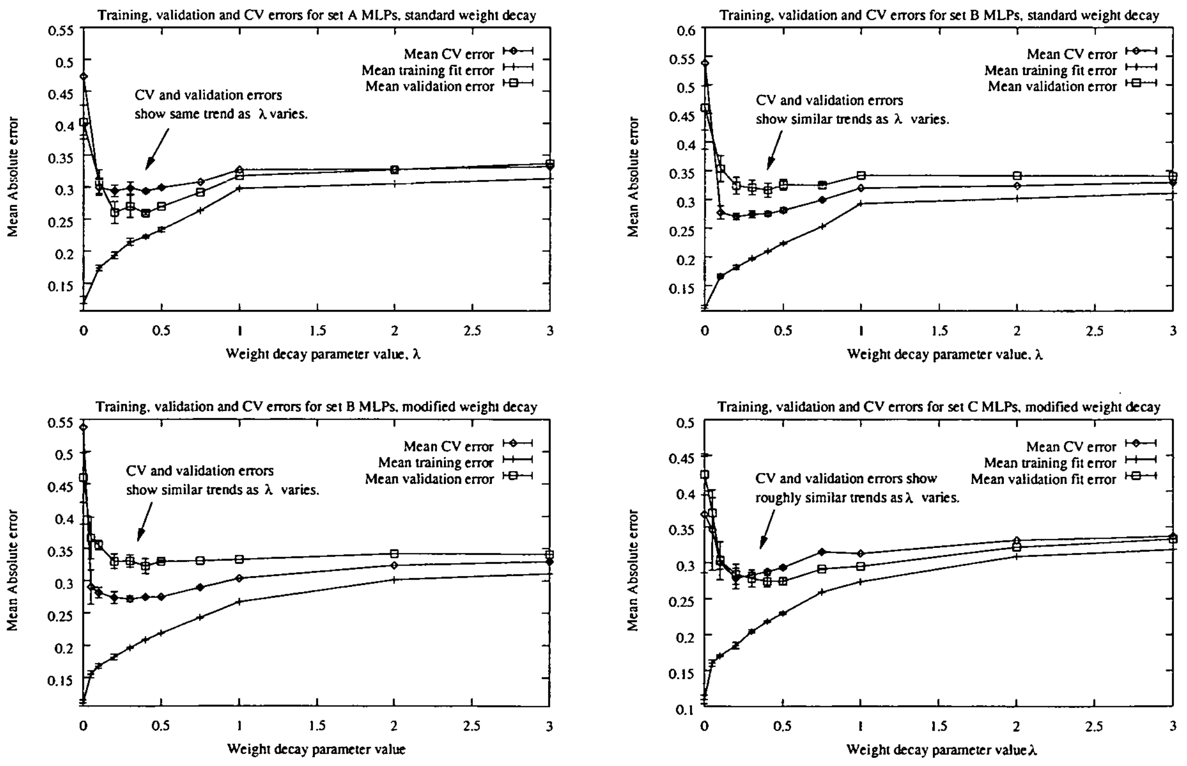


Figure 7.6: Mean training, validation and CV errors for the validation test.

Each graph in figure 7.6 shows similar trends in the validation and CV errors as λ varies. Though there is some variation between where the minima occur for each error curve, both curves show relatively flat minima and so a ± 0.1 variation in choosing the value of λ is not likely to cause serious under- or overfitting.

This good agreement between the CV and validation errors shows that reserving 20% of the total data for validation gives sufficient validation data to estimate the fit prediction error, necessary to avoid under- or overfitting of the curl model. Owing to the relatively large amount of computer time required to run the CV test, no attempt was made to try different validation set sizes such as 10%, 15% or 25% or the total data.

7.5. Developing the first MLP curl models

Training and validation sets 'A' were used to develop most of the first MLP curl models, and so this discussion is limited to results obtained using these data sets. As explained earlier, all 28 predictor variables were included in the first curl models. While this may lead to some overfitting, it was hoped that the importance of most variables could be assessed by including them, and that the least important variables then could be excluded from later models.

MLPs with 4, 6 and 8 hidden units were used to model the curl regression function. LS, $L_{1.5}$ and $L_{1.2}$ training were used for fitting and the fit complexity was controlled using standard and modified weight decay. Different numbers of hidden units were used because the smoothing given by both types of weight decay varies with the number of hidden units used, and so varying the number of units may be necessary to find the best fit. Early stopping was also used in conjunction with the weight decay to obtain the best fit.

As done previously when setting the validation set size, for each combination of fitting estimator, weight decay type, weight decay parameter value and number of hidden units, 10 MLPs were trained, and trimmed averaging of their validation errors was used to reduce the affect of any local minima on the results. Using many different training start points also increases the chance of obtaining a good fit when using early stopping.

Results from training to completion

This section discusses the results obtained by training all MLPs until the training error function had been minimised.

Similar validation error curves were obtained for all estimators and data sets. Figures 7.7, 7.8 and 7.9 show some typical results for MLPs with 4, 6 and 8 hidden units.

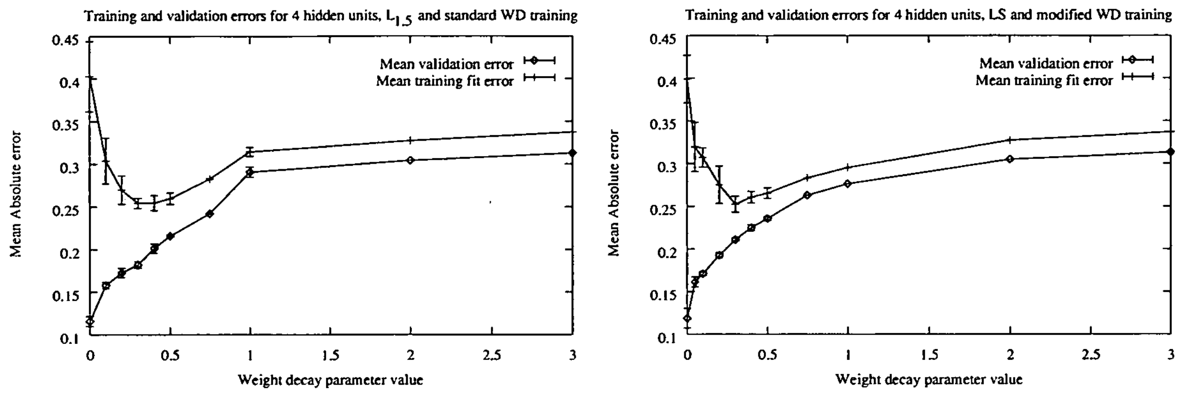


Figure 7.7: Trimmed mean training and validation fit errors for 2 of the sets of MLPs with 4 hidden units.

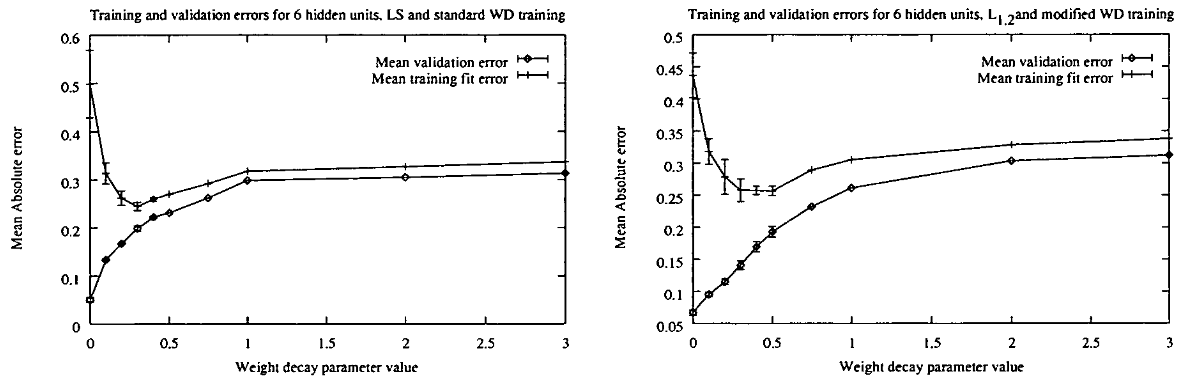


Figure 7.8: Trimmed mean training and validation fit errors for 2 of the sets of MLPs with 6 hidden units.

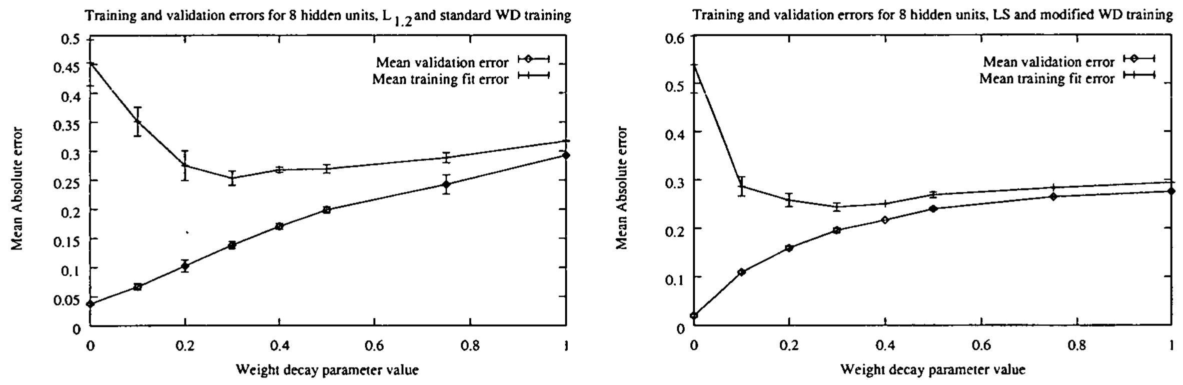


Figure 7.9: Trimmed mean training and validation fit errors for 2 of the sets of MLPs with 8 hidden units.

For all combinations of number of hidden units, training estimator and weight decay type, the minimum mean absolute validation error at completion of training was approximately 0.25. For reference, a LS linear fit to the data gave a validation error of 0.30. The fact that similar errors were obtained for LS, $L_{1.5}$ and $L_{1.2}$ training suggests that

- there are few outliers in the data, or
- there is still significant model bias which is dominating the validation errors for all estimators (this effect is discussed in chapter five), or
- the curl regression function is not very nonlinear and so overfitting is not a serious problem (see chapter five).

Results from early stopping

Since some overfitting is still possible when using weight decay, early stopping was also combined with weight decay to reduce overfitting. Table 7.2 gives the minimum validation errors (mean absolute error and MSE) obtained using early stopping and the conditions under which they were obtained. The validation errors for a linear LS fit to the training data are also included for comparison with the MLP errors.

Hidden units	Estimator	Weight decay type	λ value	MAE	MSE
4	$L_{1.5}$	standard	0.2	0.231	0.094
6	$L_{1.5}$	standard	0.1	0.210	0.081
8	$L_{1.5}$	modified	0.3	0.214	0.089
Linear	LS	N/A	N/A	0.306	0.141

Table 7.2: Lowest mean absolute and mean square validation errors obtained using weight decay with early stopping. The LS linear fit validation errors are also given for comparison.

Though $L_{1.5}$ training gave the lowest overall errors, the LS- and $L_{1.2}$-trained MLP validation errors were only slightly larger. Again, this suggests that outliers and overfitting are not serious problems, or that bias in the fit is dominating the validation errors. Testing for fit bias is discussed in the next section.

All MLP validation errors are significantly lower than those obtained for the LS linear model fit, showing that the MLPs give better predictions than the linear model in addition to fitting the training data better: that is, the MLPs are not overfitted.

7.6. Model testing and improvement

Having developed some initial curl models, the next steps attempted were

- to assess how well the best models fit the curl data and whether improvement could be expected by incorporating the unused pulp composition data, and
- to improve the models by identifying and removing predictor variables which have little affect on curl, and

- to provide Tullis Russell with some of these models for testing in production.

These are discussed in the next sections.

7.6.1. Assessing the model fit and adding the pulp variables

This work aimed to assess

- whether the MLPs fit the data well, or if significant model bias remains, and
- whether incorporating the pulp composition data is likely to improve the fit.

The method used to address both issues is that which was used earlier to assess the need for a nonlinear model, namely looking for systematic trends in the fit residuals. Any such trends indicate trends in the data which have not been captured by the MLP fit.

Assessing how well the MLPs fit the data

Figure 7.10 shows a plot of the measured curl versus the predicted curl for the best MLP fit with 6 hidden units. Compared to figure 7.2, the MLP fit is clearly better than the linear fit as the data clouds in this figure cluster more tightly around the line indicating perfect match between the measured and estimated curls (the axis scales differ from figure 7.2 because the MLP data was standardised to avoid saturation).

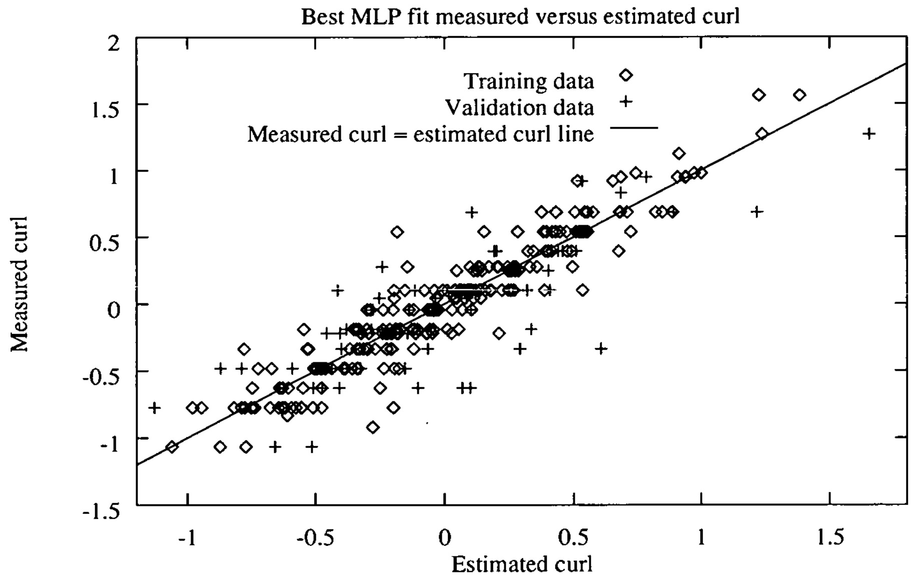


Figure 7.10: Measured versus estimated curl for the best MLP curl data fit.

Residual plotting also was used to check for any bias remaining in the best fits obtained using 4, 6 and 8 hidden units and early stopping. Figures 7.11, 7.12 and 7.13 show plots of the residuals from these fits versus some of the variables which were found earlier to be related nonlinearly to curl.

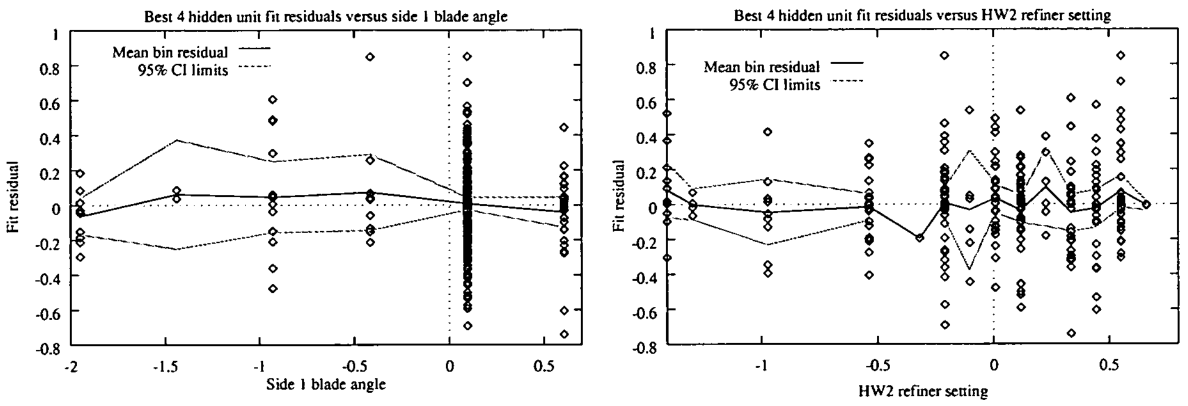


Figure 7.11: MLP fit residuals versus side 1 blade angle and HW2 refiner setting for the MLP with 4 hidden units

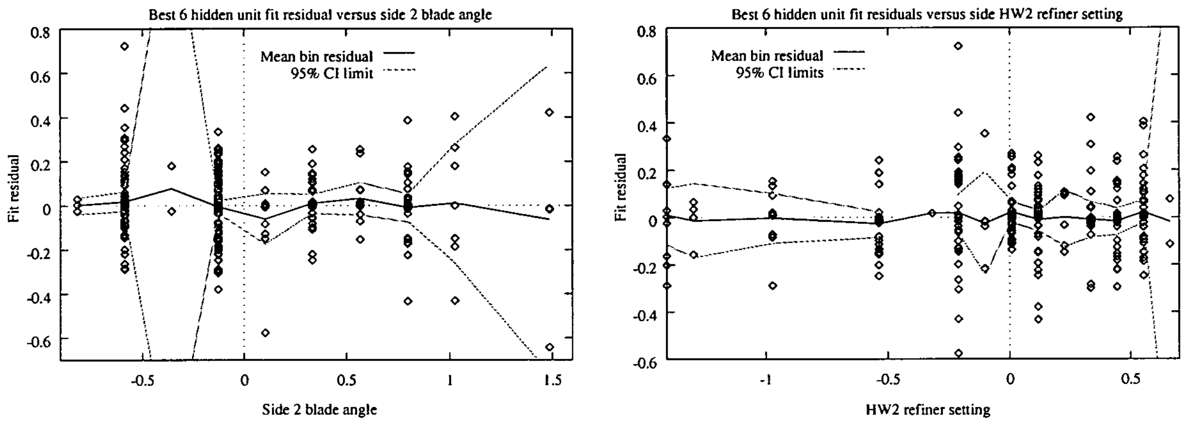


Figure 7.12: MLP fit residuals versus side 2 blade angle and HW2 refiner setting for the MLP with 6 hidden units

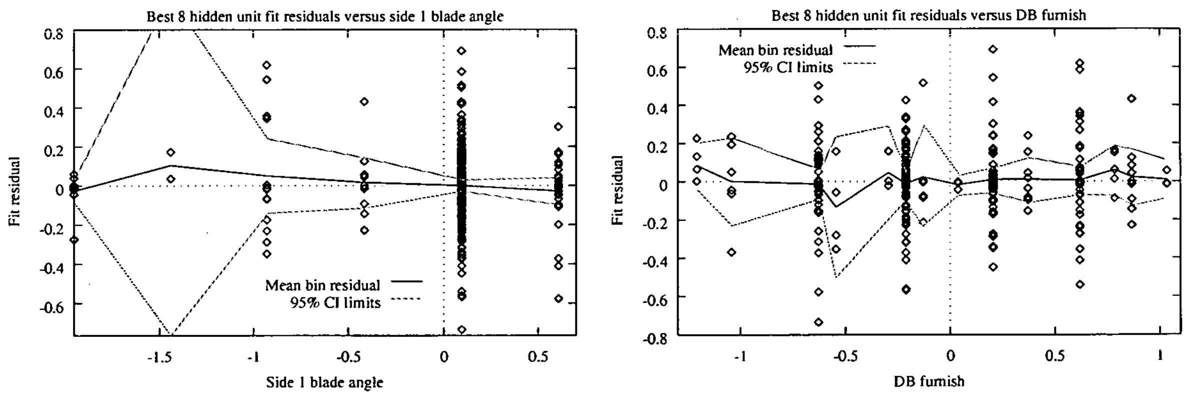


Figure 7.13: MLP fit residuals versus side 1 blade angle and DB furnish for the MLP with 8 hidden units

In all cases the nonlinear trends found earlier when assessing how well the linear model fitted the data have been reduced. This is seen in the fact that the lines joining the means in each data bin are straighter, and that the mean residuals are now closer to zero.

The lack of significant trends in the residual plots suggests that there is little bias in the fits, and so the MLP models have captured much of the nonlinear relationship between curl and the other process variables included in the model.

Incorporating the pulp data into the model

Though the last section showed that there was little or no fit bias, it was decided to confirm that the fits would not be improved much, if at all, by incorporating the pulp composition data into the model. Plotting the fit residuals against variables not included in the model provides a method for assessing whether these variables should be included in the model[35, 207].

The pulp wood types were assigned arbitrary numeric scores, and missing data entries were assumed to mean no pulp rather than missing data and were assigned the value zero. Figure 7.14 shows plots of the fit residuals versus 4 of the 5 pulp composition variables for the best MLP fit with 6 hidden units. The MLPs with 4 and 8 hidden units gave very similar plots. Note that the pulp data has been standardised, and so zero pulp scores do not correspond to the zero on the x-axis.

None of these plots show any significant non-random trends, and so adding the pulp variables to the model cannot be expected to improve the fit. The residual plot for hardwood pulp 1 (not shown) is very similar to the plot for softwood pulp 1.

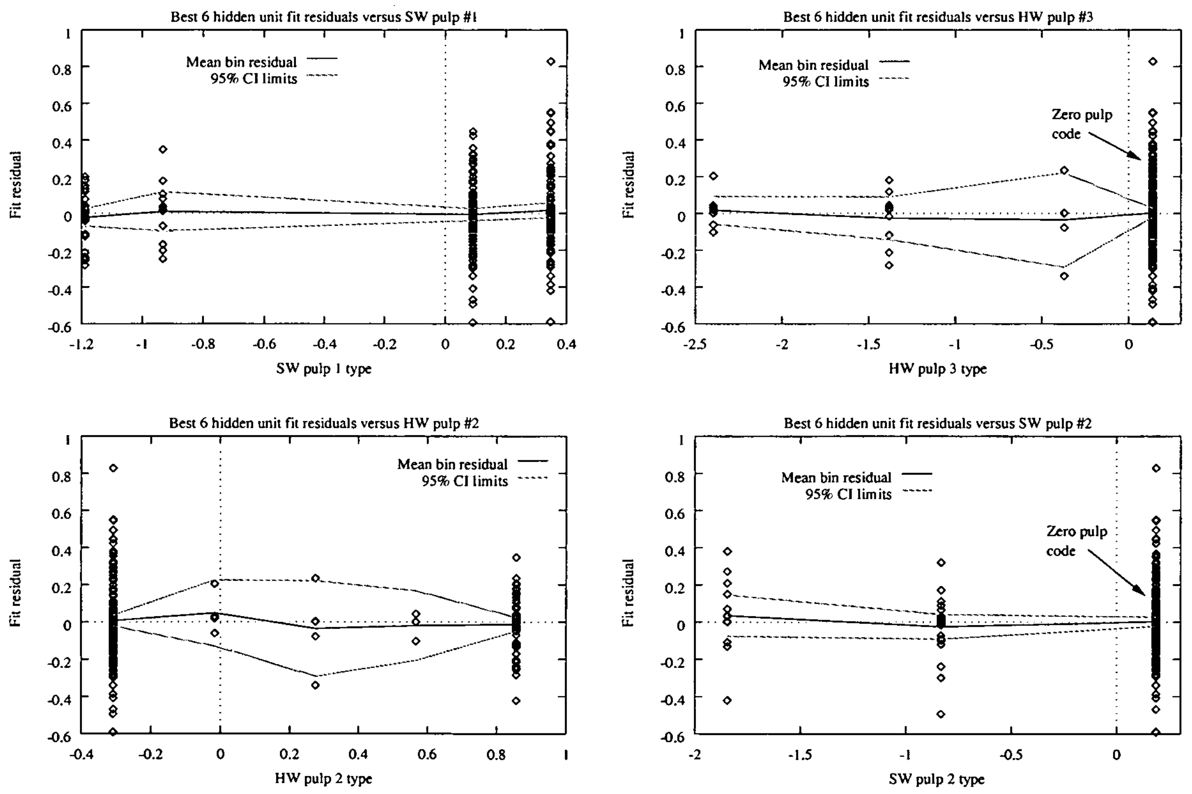


Figure 7.14: Fit residuals versus (arbitrary) pulp scores for 4 of the 5 pulp composition variables, best 6 hidden unit fit.

7.6.2. Eliminating irrelevant predictor variables

Having developed some MLP curl models which appeared to fit the data well, the final actions attempted were the identification of irrelevant predictor variables (i.e. variables which have little affect on curl) and their removal from the MLP model. Two reasons for doing this are

- since the MLP may overfit to irrelevant data, removing these data reduces the likelihood of overfitting, which may improve the fit slightly, and
- more importantly here, removing the irrelevant variables may give better insight as to what affects curl most, and hence how to best control curl on-line.

Variable selection is quite an empirical procedure in MLP regression, and so two methods were used to give better insight into which variables should be eliminated:

- weight decay and examination[103, 208], and
- saliency measures similar to those used by optimal brain damage methods[209].

Formal methods for eliminating variables in MLP regression based on linear-regression-hypothesis-testing methods have been suggested for MLP regression[210] but are rarely used. This is because a zero-valued hidden-unit-to-output weight causes nonidentifiability of the input weights to that unit, and hence these weights do not converge to any asymptotic distribution[23].

All results given in the following sections are for the MLP fits with 6 hidden units. Similar results were obtained however for the MLPs with 4 and 8 hidden units and for training and validation sets 'B' and 'C'.

Weight examination and results

If the predictor variables are standardised so that their values span approximately the same range, then the relative sizes of the input-to-hidden-layer indicate crudely how important each predictor variable is. A small weight indicates that the predictor variable makes little contribution to the hidden unit activation, and hence output. If the weight between that hidden unit and the output is also small, then the input variable has little affect on the fit via that hidden unit. If the weights leading from a given MLP input to all hidden units are small, then that predictor variable makes little contribution to the fit and is hence a candidate for elimination.

Figure 7.15 shows input-to-hidden-layer weight distributions for some of the MLP fits obtained using 6 hidden units; estimator and weight decay details are given in the title of each plot. There are 6 points for each predictor variable, corresponding to the average (over the 10 MLPs trained) weight leading to each of the 6 hidden units. The solid line shows the overall average input weight value for that variable.

The absolute input weight values were averaged because it is possible for 2 MLPs to give the same fit, but to have input weights with opposite signs[113, 114]. Thus the average weight value for an important input may be small if almost half of the fits have input weights of opposite signs (the weight signs depend on the choice of random initial weights at the start of training).

The weight distributions shown in figure 7.15 are for MLPs trained using slightly more weight decay than was found to give the best data fits. This is because assessing the importance of a predictor variable becomes easier as the amount of weight decay increases, because weight decay suppresses the sizes of unimportant weights and also provides some suppression of collinearity-induced weight-variance-inflation (c.f. ridge regression)[84, 202].

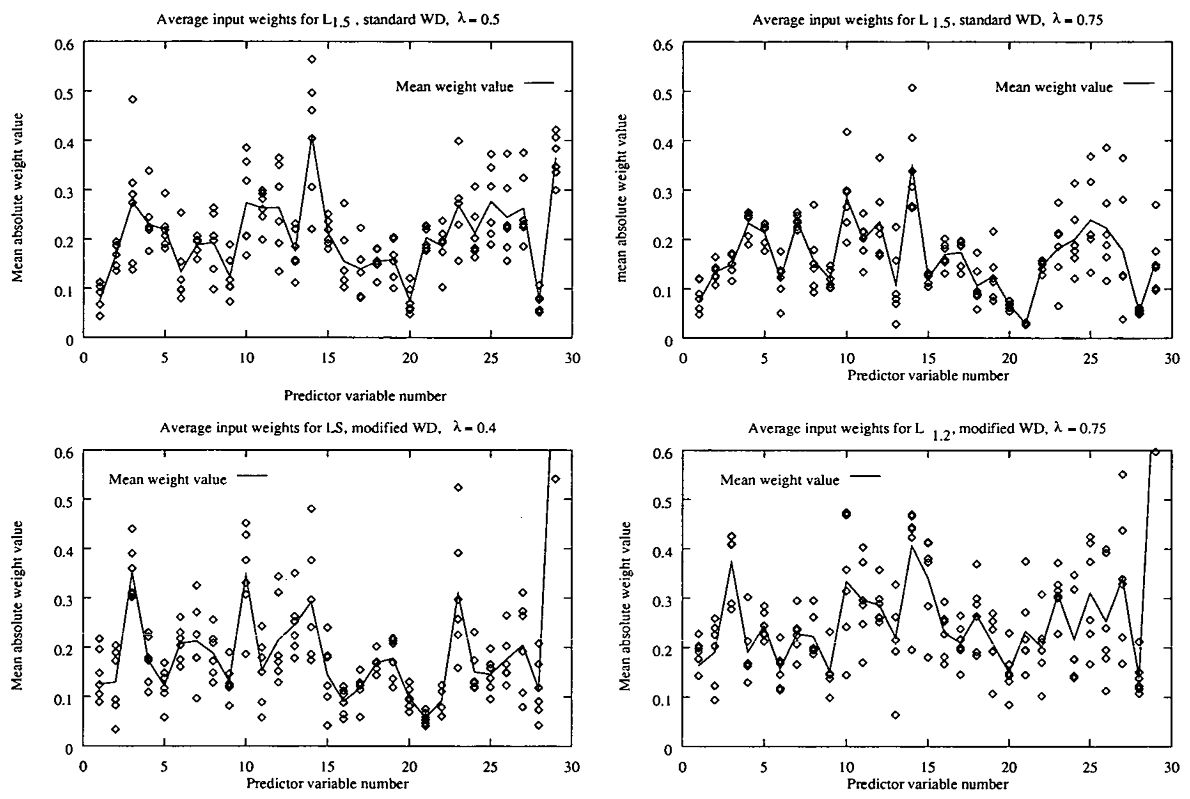


Figure 7.15: Average input weight distributions for some MLPs trained to completion. Predictor variable number 29 is the input bias (fixed value of 1).

For all MLPs giving reasonably good fits to the data, variables 1, 20 and 28 (grammage, top cobb and SW1 refiner setting) always show small weight values relative to the other input weights. Thus these variables were earmarked as the first to be eliminated. Variables 9 and 21 (final moisture and bottom cobb) also exhibited small input weights for many of the MLPs.

Saliency measurement and results

The second method used to assess predictor variable importance was to zero each predictor variable in turn and to note by how much this changes the training-data and validation-data fit errors (without re-training the MLPs). A similar idea is used commonly by MLP parameter pruning methods such as optimal brain damage, where is known as measuring the weight saliency[209]. If zeroing a variable causes little or no increase in the training and validation errors then it can be eliminated (which effectively fixes its value at zero).

Figure 7.16 shows the training and validation errors when each input is zeroed, for the same MLPs whose weights are shown in figure 7.15. The errors shown are the average training and validation set mean absolute fit errors for the 10 MLPs trained for each estimator and weight decay value. The errors shown for variable zero are the training and test errors with no variables zeroed (i.e. those obtained from training).

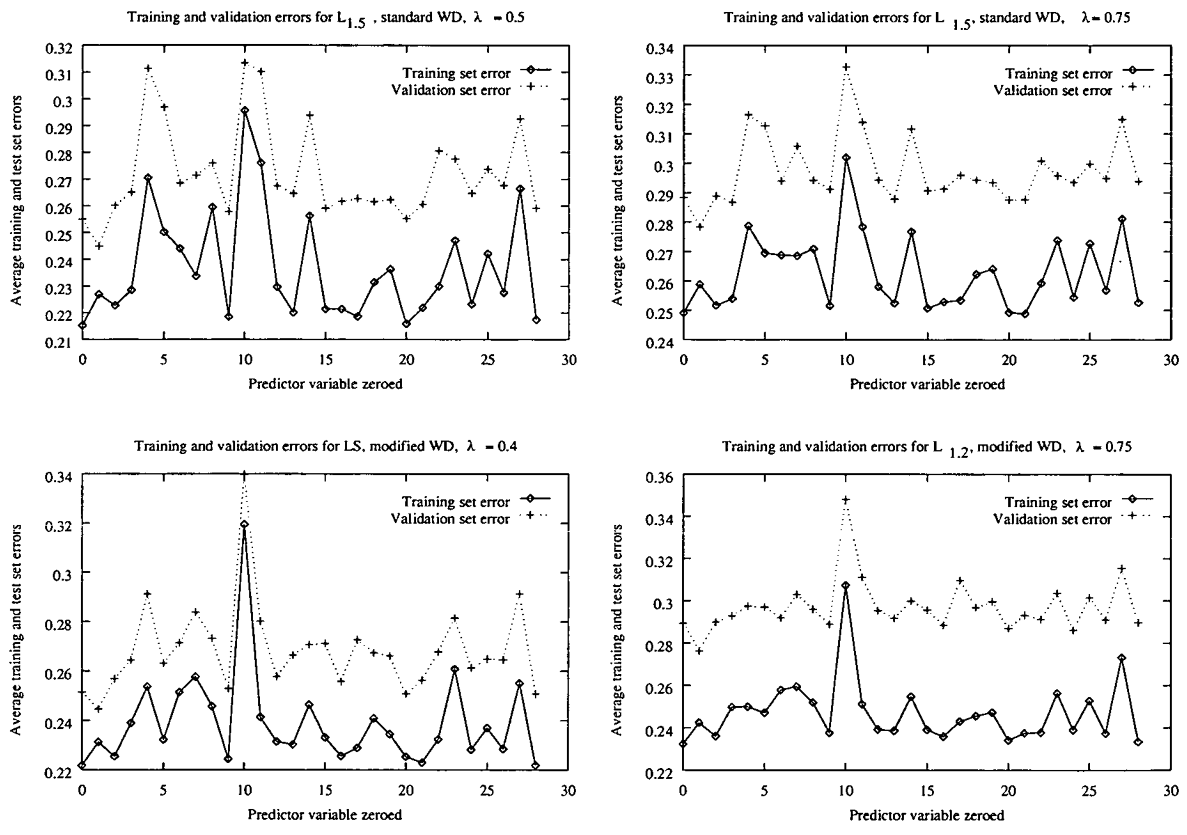


Figure 7.16: Average input saliencies for some of the MLPs trained to completion.

For all MLPs giving reasonably good fits to the data (including those not included in figure 7.16), zeroing variables 1, 9 and 28 caused at most a small increase in the training-data and validation-data fit errors. Since these were the variables which were also found to have small input weights, this was further evidence that the first elimination attempts should focus on these variables.

Other variables which generally gave small error increases were 9 and 21, which often had small input weights, and also 15, 16 and 17 (caliper and top and bottom smoothness).

It is not surprising that some variables such as grammage and caliper may have little affect on curl, because the curl data set covers only one grade of board (230 g/m^2 board), and so the grammage and caliper vary little between the reels.

It is also clear from the saliency results that variable 9 (base paper moisture) is very important for estimating the paper curl.

Deciding which variables to eliminate

From consideration of the results of weight examination and saliency assessment, it was decided to try 3 sets of variable elimination, starting with elimination of the variables which were most likely to be irrelevant (1, 20 and 28). The 3 data-set names and variables which were eliminated are given in table 7.3.

Data set	Variables eliminated
A1	1, 20, 28
A2	1, 9, 20, 21, 28
A3	1, 9, 15, 16, 17, 20, 21, 28

Table 7.3: Variable elimination combinations tried, based on weight examination and saliency assessment results.

Results of the variable elimination experiments

MLPs with 4 hidden units were trained to estimate the regression function for training sets 'A1' to 'A3'. Table 7.4 gives the lowest mean absolute validation errors obtained using early stopping.

Data set	Lowest validation error
A1	0.233
A2	0.225
A3	0.233

Table 7.4: Lowest mean absolute validation errors for the reduced variable data sets.

The validation errors are very similar to the lowest validation error obtained using all 28 predictor variables and 4 hidden units (which was 0.235). Thus it appears that weight examination and saliency assessment have identified successfully 8 predictor variables which are not important for estimating curl. The fit accuracy is confirmed by figure 7.17, which shows a plot of the measured curl versus estimated curl for the best fit obtained with 20 predictor variables. The curl estimates are still more tightly clustered around the line showing where the measured an predicted curls are equal than those given by the LAD linear fit, shown in figure 7.2.

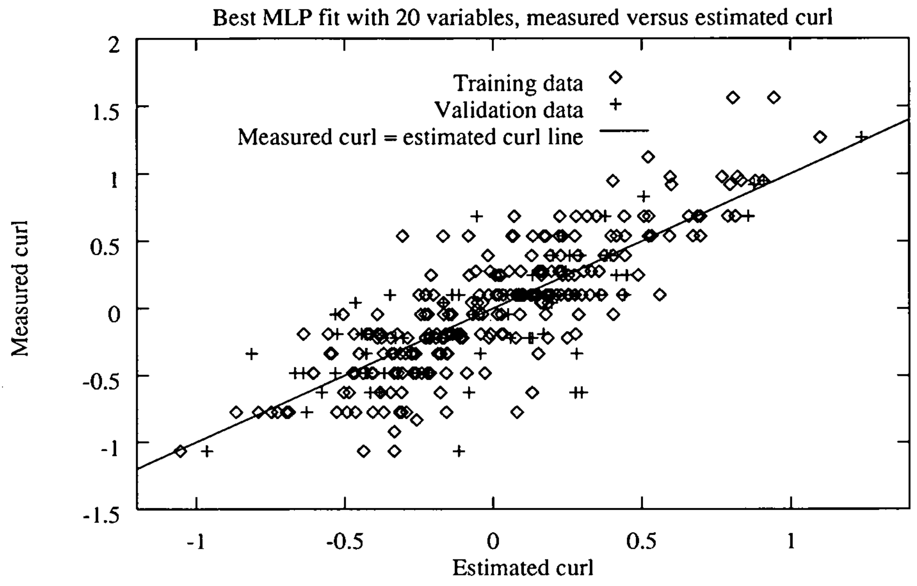


Figure 7.17: Measured versus estimated curl for the best MLP fit with 20 predictor variables.

Time did not permit further attempts at variable elimination. However, elimination of more predictor variables may be possible. For example, since curl depends on differences between the properties of the top and bottom sides of the paper, using differences of top and bottom side measurements as predictor variables may be an effective way of eliminating more variables. Work on this problem currently being undertaken by other members of the research group has confirmed that such an approach is effective in reducing the number of predictor variables, and hence giving more insight into how to control the coating process more effectively.

The 2 elimination methods used here can also be ineffective for identifying groups of irrelevant predictors which are moderately- to highly-collinear with each other. Thus this offers further possibility for finding more irrelevant predictor variables. Again, time did not permit investigation of this issue.

7.6.3. Results from field-testing a curl model

While variable elimination methods were being examined, Tullis Russell were given a MLP curl model for field-testing. The model provided was the best MLP fit obtained with 4 hidden units, and was tested using 227 new records.

Testing gave poor results, with the MLP curl model predicting the lowest curls for paper with the highest measured curls. Figure 7.18 shows a plot of estimated and measured curls provided by Tullis Russell. Pearson's correlation coefficient for the data in figure 7.18 is -0.215, indicating moderate anti-correlation between the estimated and measured curls.

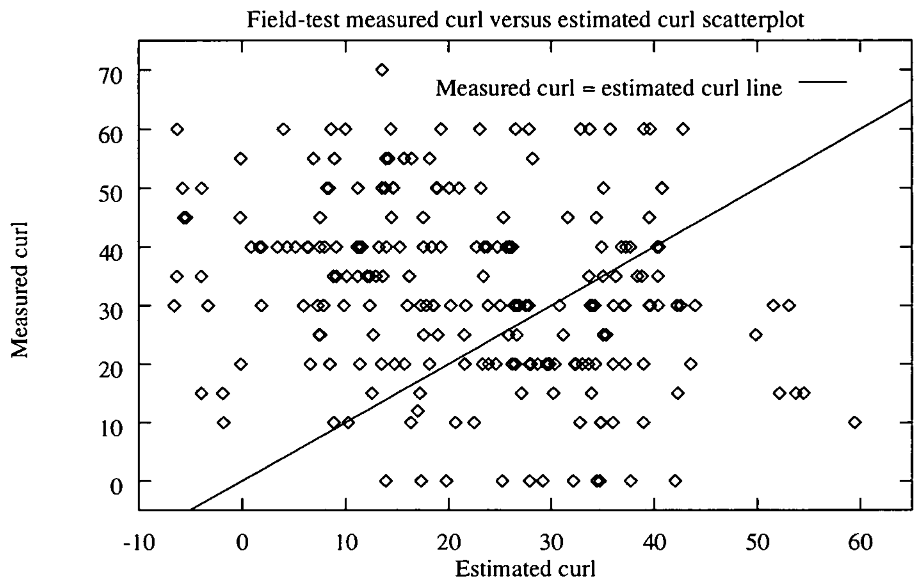


Figure 7.18: Measured versus estimated curl results from the MLP curl model field-test. The MLP estimates the measured curl very inaccurately.

It was not possible to determine why the MLP curl estimates were so poor because the new data set provided by Tullis Russell did not include 5 predictor variables used by the curl model (though these were not very important variables). However, the earlier assessment of the regression function linearity showed that a linear fit can give reasonable curl estimates. It was thus decided to compare linear fits to the old and new data to determine if the poor curl predictions were due to

- a poor MLP curl model, which would be the case if the linear fit predicted the new data curls more accurately, or
- a genuine difference between the regression functions for both data sets.

LS was used to fit linear regression models to the new data and to a subset of the old data which excluded the predictor variables missing in the new data. Both fits predicted the curl for their own data sets reasonably well, as shown in figure 7.19.

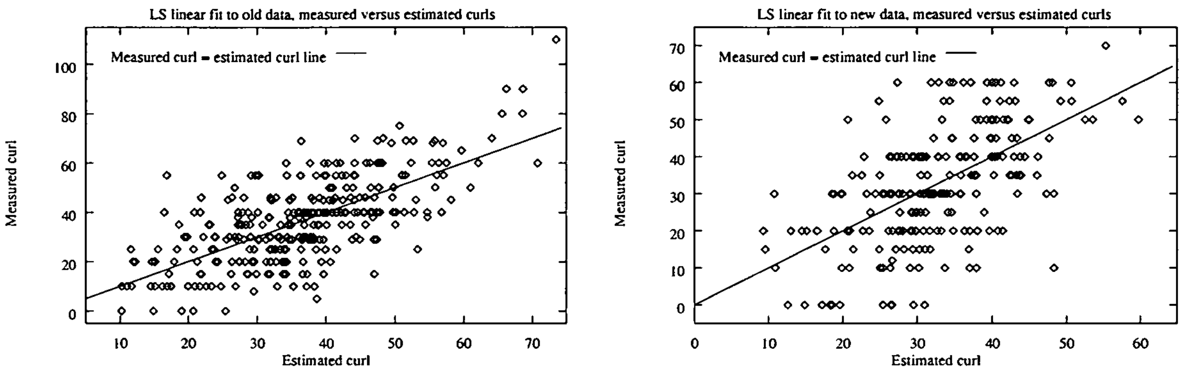


Figure 7.19: Measured versus estimated curl for linear fits to the MLP model development (old) and field-test (new) data sets.

Each fit was next used to predict the measured curl for the other data set. Figure 7.20 shows the results of this test. It is clear that neither model predicts well the curl for the other data, and in both cases the estimated curls are again anti-correlated with the measured curls.

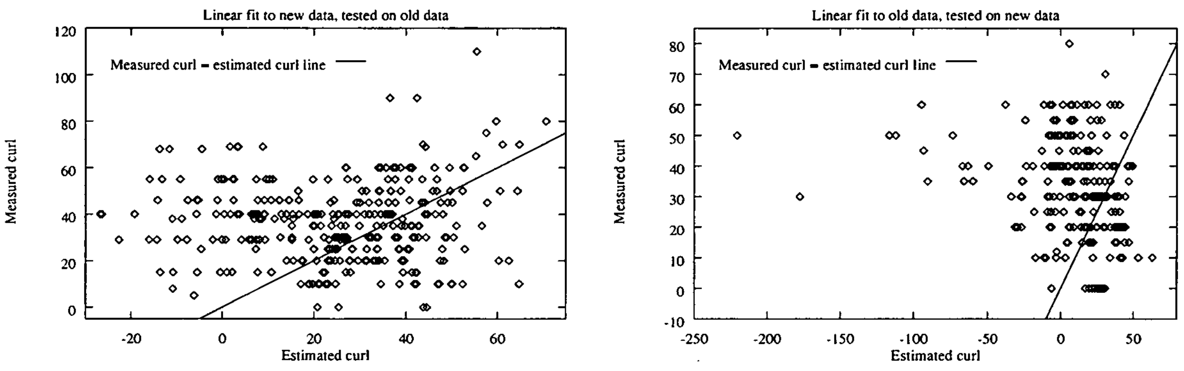


Figure 7.20: Measured versus estimated curl when using each linear fit to estimate curl for the other data set.

Since each fit estimates curl well for the data used to obtain the fit, but not for the other data, it was concluded that both data sets had quite different regression functions relating curl to the predictor variables. Thus is it not surprising that the MLP curl model provided poor curl estimates.

Since both data sets were collected around a year apart, it is suspected that the differences in the regression function are caused by changes in the coating process due to, for example

- machine wear and component drift, or
- changes in other variables which were constant in each data set but perhaps varied between them, such as a coating mix composition.

It was also noted that the new data had different base pulp composition codes, and it is not yet known whether changes in the base paper composition may be responsible for the different regression functions.

Further collection of data and investigation is clearly required to determine why the regression functions differed between the data sets. In particular

- if the differences are due to drift of the process, then future curl models may need regular updating to follow process drifts, and
- if the differences can be traced to other causes, such as a change in coating mix or base pulp composition, then these variables should be included in future models.

7.7. Future model improvements

The curl modelling project is being continued by other members of the research group. Planned improvements for the model include adding prediction intervals, so that each curl estimate is assigned a range of curl values which are likely to be observed. This will also raise a number of interesting issues when using the model for curl control, such as should the machine settings be adjusted to minimise the predicted curl if this actually causes an increase in the maximum possible curl (i.e. moving into a region with much wider prediction intervals).

7.8. Summary and conclusions

This chapter presented the work performed to develop a MLP curl model. A systematic approach was used, where the quality of the model fit was checked at each point, and the MLP was used only once it had been confirmed that it could be expected to give a better fit than a linear curl model.

Usefulness of the preliminary work

In terms of the preliminary work discussed in chapters five and six, the fast cross-validation method was found very useful for setting the size of the validation set. This method provided a faster methods for checking the validation set adequacy without having to try many different data splits. The use of robust methods actually turned out to be non-essential for the curl modelling problem. This was considered to be due to

- there being few gross outliers in the data, and
- overfitting not being a problem due to the fact that only a mildly nonlinear fit was required to estimate curl well.

However, this conclusion was reached only because robust methods were used. If only LS had been used, then doubt would have remained about whether some data were outliers.

Variable elimination methods

Some empirical variable elimination methods were also applied successfully to the problem of removing irrelevant predictor variables from the model. The most important reason for doing this here was to ease interpretation of which variables affect curl most, and hence how to best control the coating process for curl. Some possibilities for eliminating more variables were also discussed, and work on this problem is presently underway by other workers within the research group.

Field testing

The overall point of developing a curl model was to allow curl to be estimated, and hence controlled, on-line. Field testing of a curl model gave poor results, and on further analysis, it was considered that this was due to a major change in the way curl is affected by other process variables, rather than a failure of the MLP curl model. The reasons for this change are not yet know, but it may be due to process drift with time, or to hidden variables, such as changes in a coating mix, which changed between the collection of the model development and test data. This issue demands further research to ensure that this does not happen again, and that future curl models can predict curl for new paper accurately.

Overall conclusion

Overall, it was shown that curl could be modelled quite accurately using a simple MLP regression model. Consequently, future work should probably focus more closely on why the existing model failed on data collected many months later than the data used to develop the model, and how such events can be avoided in future.

Chapter 8

Summary and Conclusions

8.1. Project summary

The overall aim of this project was to examine whether a MLP could be used to model the relationship between paper curl measured after coating and other process variables measured during coating at Tullis Russell & Co. Ltd.'s Markinch coating plant. This model would then be used for on-line estimation and hence control of curl.

The data provided for modelling curl presented various problems, such as missing data and the possibility of outliers, and methods for dealing with these problems had not been examined in much depth in the existing MLP literature. However, it was appreciated that these problems could hinder the development of a curl model and were also general problems which may need to be addressed by many users of MLP modelling methods. It was thus decided that these issues should be examined first before focusing on the task of modelling curl.

The methods used to address the data deficiency problems drew heavily on existing methods in the statistical regression literature for dealing with these problems. This approach was adopted because

- it was realised that the MLP and its existing training methods were simply a type of parametric regression model and estimation methods, and
- the methods in the regression literature were already well researched and understood.

Most of the project time was devoted to examining practical issues in the application of existing robust regression methods to MLP regression, and the extension of cross-validation (CV) methods to MLP regression. Once these issues had been addressed, it was felt that sufficient knowledge had been gained to tackle real modelling problems, such as curl modelling, effectively.

Robust MLP regression: summary and conclusions

Outliers were considered to pose the most serious obstacle to the development of a good curl model, and so the application of existing robust regression methods to MLP regression was considered first.

The two key issues which were examined were

- what affects estimator efficiency in MLP regression, and
- whether high-breakdown estimators are necessary to minimise the affects of high leverage data in MLP regression.

These are key issues because the answers to these problems largely dictate which kinds of robust estimators to use for most problems.

Efficiency was found to depend as much on avoiding overfitting as matching the estimator used to fit the MLP as closely as possible to the error distribution. A consequence of this was that L_p-Norm estimation with $p \approx 1$ was likely to give the best results for many problems because in addition to resisting outliers, these estimators were found to be slow to overfit. The use of more sophisticated estimators which have LS behaviour for small errors actually gave poorer fits because the LS behaviour leads to fast overfitting, which can ruin the fit before the issue of optimising the training method to the error distribution becomes the key issue.

High leverage outliers are an important problem in linear regression, but not in MLP regression. Chapter five shown that they may lead to localised overfitting, but usually will not cause the sort of global fit breakdown that they can cause in linear regression. Thus the high breakdown estimators devised to address this problem in linear regression are not necessary for MLP regression. In fact, it was illustrated and explained that the use of these estimators may actually lead to very bad fits to the data, and hence that they should not be used at all for MLP regression.

The overall conclusion of this work was that, while the use of quite sophisticated robust estimators has been recommended previously in the MLP literature, there is in fact little reason to use them, and they may actually give poor fits compared to those obtained using simple robust estimators in many cases.

Overfitting and CV: summary and conclusions

Avoiding under- and overfitting is one of the most important and difficult problems which must be considered when using MLP regression. If either of these occurs then the fit cannot be expected to give good estimates of the regression function.

The work on robust MLP regression led to an examination of leverage in MLP regression, the primary goal of which was to allow localised overfitting due to high leverage data to be identified and reported. It was found that high fit leverages did indicate local overfitting, and hence could be used in addition to the use of a validation set to warn of overfitting.

The most important work presented here then examined how the leverages could be applied to the problem of estimating the fit prediction error by estimating the leave-one-out cross-validation error. The advantages of the method discussed here over direct implementation of leave-one-out or using a validation set to estimate the prediction error are

- it is much less computer-intensive to use than true leave-one-out CV, and
- unlike using a validation set, all data can be used for training, which reduces the likelihood of overfitting, and
- the problems associated with choosing the validation set size and coverage are avoided.

The fast CV method was shown to work well for a real regression problem (the curl modelling problem), though a number of limitations of the method were also discussed. In particular, the method does not work well unless LS training is used, which renders it incompatible with the use of robust training methods. This issue requires further work.

Other minor issues examined

Since the fast CV method requires the MLP to be trained to completion, early stopping cannot be used to control the fit complexity. Regularisation is the obvious method to use instead, and roughness penalties based on spline-smoothing were examined as one type of useful smoothing penalty. It was found that these have a number of practical difficulties, however, and that much work is required before they can be used for all but the simplest MLP regression problems.

Application to the curl modelling problem

The curl modelling problem actually presented fewer problems than first anticipated, mainly due to the fact that the regression function linking curl to the values of the other process variables was not strongly nonlinear.

There were few or no gross outliers in the data, and so robust methods did not give much more accurate fits to the data than conventional LS fitting. In this sense, the use of robust methods was unnecessary, but without having used them, it would have been more difficult to reach the firm conclusion that there were no gross outliers.

The fast CV method was useful, however, for confirming that the validation split used for the curl modelling problem gave a suitable validation set. Without this quick method of checking the validation split, it would have been necessary to compare many different splits, a very computer-intensive (and hence slow) task. Use of a validation set was necessary because the present version of the fast CV method does not work with robust training methods.

A good curl model was obtained quite quickly with the aid of the methods developed. The most serious problem now appears to be ensuring that the model accounts for all factors that affect curl, since model field testing showed there to be other factors affecting curl which are not accounted for by the current curl models.

8.2. Overall conclusions, practical guidance and future work

This thesis has examined methods for dealing with two important problems in MLP regression, namely dealing with outliers and estimating the fit prediction error (often called the generalisation ability in the MLP literature). When this project was started, there was little or no existing practical guidance on dealing with these problems, and in the case of robust methods, some of the existing advice has been demonstrated to be wrong.

Key-points summary of this work and recommendations for its use

The following bullet points summarise the key results of this work presented in this thesis and give some practical advice for those wishing to use this work.

- As in classical regression, robust training methods should be used in MLP regression when outliers may be present in the data.
- Simple L_p-Norm estimators with $1 \le p \le 1.25$ give good efficiency in MLP regression and are much easier to use than other common robust estimators. More sophisticated robust estimators require greater complexity control to avoid overfitting, and so overfitting usually occurs, giving worse fits than obtained using the simpler L_p-Norm estimators.
- Even when outliers are not suspected, a robust fit should be tried for 2 reasons. The first reason is to confirm where there really are no outliers influencing the LS fit. The second reason is because robust L_p-Norm estimators can give better fits than LS even when there are no outliers; again this is because overfitting occurs more slowly for these estimators.
- Even when using robust training, some data may still strongly influence the MLP fit; this may be due to overfitting at these data, or because they lie far from the other data and so completely determine the local fit. These influential data can cause the local fit to be a poor estimate of the regression function, and can be identified after training by their high leverages. If the high leverage is due to local overfitting, then better complexity control can be attempted to improve the fit.
- The fast CV method provides a method for validating MLP fits without the need to reserve validation data; however, this method presently works only for LS fitting. This is not a serious limitation as comparing the CV and validation errors for a LS

fit (even if it is affected by outliers in the data) provides a method for checking the adequacy of the validation data set. If the CV and validation errors agree then it is likely that the validation data is suitable for validating robust MLP fits.

- It is recommended that some validation data is reserved if possible when using the fast CV method since both methods have limitations, and comparing the CV and validation prediction errors provides a good 'double-check' of the prediction error estimate given by the other. If the estimates disagree badly, then further investigation is required to assess why they disagree.

The methods and results presented here are applicable to all MLP regression problems, not just curl modelling, and form part of a useful tool-box of methods for attacking future modelling problems. Other techniques are also required for this tool-box, such as methods for dealing with missing data and generating fit confidence intervals; there was no time to examine these in depth here, though there is currently much work in progress on these problems.

Further work

For the general techniques devised here, the most important issue in need of further work is the extension of the fast CV method work with non-LS training methods (notably, robust methods). In chapter six it was shown how the failure of the fast CV method could be related intuitively to the influence function of the estimator used for training, and this suggests one route to follow to tackle this problem.

Further work on robust MLP regression should examine methods for validating MLPs when there are outliers in the validation data. In chapter five, it was stated that although using robust validation errors appear to address this problem, this method may also be insensitive to overfitting, leading to MLPs of the wrong complexity being used. Some preliminary work on addressing this problem using multiple validation errors was discussed, but further work is required to assess how serious the problem is, and which methods are most effective for tackling it.

Further work in the curl modelling problem should firstly address why the existing model gave very poor curl estimates for new paper. Without addressing this issue, it is likely that future curl models will similarly fail if, for example, they lack some important predictor variable which caused the failure of the current model.

Other work on the curl modelling problem should address the construction of prediction intervals for the fit so that the machine crews can assess how trustworthy the curl predictions are. Some preliminary work on this problem is presently underway by the another curl modelling researcher within the research group.

Appendix A

Maximum likelihood estimation

A.1. Introduction

Maximum likelihood (ML) estimation is mentioned in chapters two and five in the context of least squares (LS) and robust MLP training methods. This appendix provides a brief review of the basic principles of this method and some of the important properties of ML estimates.

A.2. Estimating parameters of probability distributions

A probability density function (pdf) is often characterised by a number of parameters which specify the precise shape of this function[142]. For example, the univariate Gaussian density

$$P(Y = y;\, \mu, \sigma^2) \;=\; \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) \tag{A.1}$$

specifies a family of density curves where each member is characterised by a distinct population mean, μ, and variance, σ^2.

It is often that case that theory or experience suggests the type of distribution which describes the behaviour of a random variable, but not the values for the parameters of its pdf. Hence a common problem in statistics is that of estimating these values for a given random variable.

Since information about the true parameter values can be gained by observing outcomes of the random variable, parameter estimates are usually functions of these outcomes. Methods for deriving suitable estimating functions are discussed in most introductory mathematical statistics textbooks, and ML is one of the more important methods since it often leads to estimators with good statistical properties[142, 211].

A.3. The sample likelihood function

Suppose that the random variable Y is known or assumed to have a particular pdf of the form

$$P(Y = y) \;=\; f(y; \underline{\theta}) \tag{A.2}$$

where $\underline{\theta}$ represents the parameters which characterise this density function. If Y is observed several times giving the set of outcomes $\{y_1, \ldots, y_n\}$, and if these outcomes are independent, then their joint probability is simply

$$P(Y_1 = y_1, \ldots, Y_n = y_n) = \prod_{i=1}^{n} f(y_i; \underline{\theta}) \tag{A.3}$$

which follows from the basic properties of independent random variables[140].

Since this probability expresses how likely the observed outcomes are for different values of the parameters, it is known as the sample likelihood function, and is denoted $L(\underline{\theta})$.

A.4. Maximum likelihood estimation and log likelihood

Returning to the problem of estimating the unknown parameter values of Y, the likelihood function shows that the probability of a given set of outcomes depends on $\underline{\theta}$, and these outcomes will generally be more likely for some values of $\underline{\theta}$ than others.

Intuitively, then, it seems reasonable to use the parameter values which maximise the likelihood as the estimates of the unknown true parameter values, since these provide the most likely explanation of the data when the assumptions about the distribution of Y are correct. The estimates obtained in this manner are thus known as ML estimates (MLEs).

If the likelihood function is a smooth function of $\underline{\theta}$, then standard differential calculus can be used to find the parameter values which maximise this function. However, differentiation of products is often a tedious process, and so the sample log likelihood

$$l(\underline{\theta}) = \ln(L(\underline{\theta})) = \sum_{i=1}^{n} \ln(f(y_i; \underline{\theta}))$$

is usually maximised instead since sums are easier to differentiate. Note the parameter values which maximise the likelihood and log likelihood functions are identical because the logarithm function is monotonic.

A.5. Properties of MLEs

Though ML estimators can be justified intuitively, as shown above, the main reason why the ML method is important is that it commonly leads to estimators with various good, and often optimal, statistical properties. However, a discussion of these properties and the conditions under which they arise would be too long and complicated for this brief overview, and details can be found in[140, 142, 211, 212].

For the purpose of this review, it will simply be stated that the most important properties of ML estimators are their large sample properties. Under a few mild regularity conditions, they can be shown to have asymptotically Gaussian sampling distributions and to be consistent and asymptotically efficient estimators[213, 214]. Hence the ML method leads to estimators with good properties for sufficiently large samples, and sometimes for small samples too.

A.6. Example - LS estimation

A simple example will help to illustrate the principles of ML estimation, and also show the equivalence of the ML and LS estimates of the population mean of a Gaussian variable, which is stated in chapters two and five.

Suppose that Y is known or assumed to have the Gaussian distribution

$$P(Y = y; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) \tag{A.5}$$

where the values of μ and σ^2 are unknown and are to be estimated. The sample likelihood function is

$$L(\mu, \sigma^2) = \frac{1}{(\sigma\sqrt{2\pi})^n} \prod_{i=1}^{n} \exp\left(-\frac{(y_i-\mu)^2}{2\sigma^2}\right) \tag{A.6}$$

and this can be simplified using the multiplicative properties of exponentials to give

$$L(\mu, \sigma^2) = \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left(-\sum_{i=1}^{n} \frac{(y_i-\mu)^2}{2\sigma^2}\right) . \tag{A.7}$$

The log likelihood function is thus

$$l(\mu, \sigma^2) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2 \tag{A.8}$$

and it may be noted that the right hand term is proportional to the (negative) sum of squares error which is minimised in LS estimation problems[215]. Since the value of μ which maximises the sum in this term is independent of the value of σ^2, it can thus be seen that the ML estimate of μ is also the LS estimate of this parameter.

Differentiating $l(\mu, \sigma^2)$ with respect to μ and equating the gradient to zero to find the maximum gives

$$\frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - \hat{\mu}) = 0 \tag{A.9}$$

which can be re-arranged to give an algebraic expression for the estimator $\hat{\mu}$,

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} y_i \ . \tag{A.10}$$

Thus the ML estimator of the population mean of a Gaussian random variable is simply the sample mean. Repeating this process for the variance parameter yields the estimator

$$\hat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\mu})^2 \tag{A.11}$$

which is the sample second moment about $\hat{\mu}$.

A.7. Practical ML estimation

To complete this brief review of the ML method, this section outlines some of the problems which often must be addressed in practical ML estimation. All of these issues are relevant to the use of ML estimation to estimate MLP model parameters, and are discussed at various points throughout the thesis.

No closed form estimators

It is often impossible to derive closed form expressions for the estimators, such as those derived in the previous example. This problem usually occurs when the log likelihood is a complex nonlinear function of one or more of the parameters, and numerical methods such as iterative optimisation methods must then be used to maximise this function[55, 56, 212]. This may require large amounts of computer time.

Multiple likelihood maxima and minima

Another common problem is that the log likelihood function may have many local maxima in addition to the global maximum. Gradient ascent optimisation methods can often become trapped in these maxima, and hence fail to find the global maximum.

The presence of local maxima can often be detected by estimating the parameters a few times using different starting estimates, to see if these converge to different maxima. If many local maxima are present, then methods such as simulated annealing can be used to find the global maximum, or at least the vicinity of this maximum. These methods can be quite computer-intensive however.

The likelihood function may also exhibit local minima, and these should also be avoided. Fortunately, gradient-based maximisation methods will always move away from minima, and so it is unlikely that these will cause problems. If doubt arises however, the second derivative or Hessian matrix can be tested to confirm that a given stationary point is a maximum.

Estimate identifiability (uniqueness)

Under some conditions, the maximum of the likelihood may not occur at a point, but along the top of a flat ridge or plateau of this function. If this occurs then there are an infinite number of possible values for the estimates, and the asymptotic estimator properties no longer hold[213].

In regression, nonidentifiability is often due to the use of an overparameterised model, though it can also be caused by certain deficiencies of the data used to estimate the parameters. It is usually manifest as numerical ill-conditioning problems when computing the estimates[195].

Model validity

A final point which should be noted is that the optimal properties of ML estimators can depend quite strongly on the assumed distribution of the random variable Y. Some estimators which perform well for one distribution may be poor estimators for corresponding parameters of slightly different distributions. Further discussion of this issue can be found in the robust statistics literature[129, 144].

Note that in the regression context, the (inadvertent) use of a biased model will often upset the assumptions about the error distribution because the model bias leads to systematic trends in the residuals. This is discussed further in[32, 216].

Appendix B

The IF and influence in MLP regression

B.1. Introduction

This appendix looks at the influence function (IF) in MLP regression to aid understanding of some material presented in chapters five and six. Chapter five looks at robust MLP regression, where the IF plays a key role in determining an estimator's resistance to outliers. Since this is one of the most important uses of the IF, this appendix focuses mainly on examining the IF in MLP regression to assess resistance to outliers. In chapter six, the IF explains how training a MLP using different estimators gives different relationships between the fit leverages and residuals.

This appendix builds towards examining the IF in MLP robust regression as follows:

- A general IF is derived which covers all estimators of interest here.
- Some simple estimators are used to show how an estimator's IF can be used to determine if it is resistant to outliers. This review also looks at how the IF describes the estimator's sensitivity to small perturbations of the data, which is important in chapter six.
- The IF in simple linear regression is examined to show how the IF can be applied to regression. In particular, it is shown how leverage points can have high influence even when some robust estimators are used to fit the linear model.
- Having examined the IF in simple linear regression, the IF for MLP regression is examined. It is shown that robust MLP regression methods may be non-resistant when the training data contains high leverage outliers, but that it is difficult to assess the practical significance of this from theory alone. A simple example is used to show what is likely to happen in practice and discuss how important this is.

B.2. Derivation of a general influence function

In this section, a general IF is derived from which the IFs for all the types of estimators considered in this appendix can be derived easily.

Let the parametric model $f(\underline{x};\underline{\theta})$ be used to model the regression function for the population with cumulative distribution function $H(\underline{x}, y)$. If penalised M-estimation is used to fit the model, and in the limit that the amount of data used to train the MLP becomes infinitely large, the parameter values, $\underline{\Theta}_H$, are found by minimising

$$\int \rho(y - f(\underline{x}, \underline{\theta}))\ dH(\underline{x}, y) + \lambda J(\underline{\theta}) \tag{B.1}$$

where the function $\rho(z)$ defines the M-estimator as discussed in chapter five, and λJ is a regularisation penalty of the type described in chapter three. Minimising (B.1) is equivalent finding a suitable zero of the scores statistic,

$$\int \psi(y - f(\underline{x}, \underline{\theta}_H))\nabla_{\underline{\theta}} f(\underline{x}, \underline{\theta}_H)\ dH(\underline{x}, y) - \lambda \nabla_{\underline{\theta}} J(\underline{\theta}_H) \ = \ 0 \ . \tag{B.2}$$

where $\psi(z)$ is the derivative of $\rho(z)$ with respect to z. For simplicity, any scale parameter associated with the M-estimator is assumed to be a known constant and has been omitted.

Now let a point mass contaminant, $\delta_{\underline{x}_0, y_0}$, be added to $H(\underline{x}, y)$, giving the new distribution

$$G(\underline{x}, y) \ = \ H(\underline{x}, y) + \varepsilon \left(\delta_{\underline{x}_0, y_0} - H(\underline{x}, y) \right) \tag{B.3}$$

where ε determines the mixing ratio between the original distribution and the contaminant. The influence function

$$IF(\underline{x}_0, y_0; \rho, \lambda J, H) \ = \ \left. \frac{\partial \underline{\theta}_H}{\partial \varepsilon} \right|_{\varepsilon=0} \tag{B.4}$$

describes the change in $\underline{\theta}_H$ which is caused by adding an infinitesimal amount of contamination at $(\underline{x}_0, y_0)$. The estimator is resistant to outliers only if this change is bounded (i.e. the value of the IF is finite) no matter where the contaminant is placed[102, 129].

Re-fitting the model under $G(\underline{x}, y)$, the new parameter values, $\underline{\theta}_G$, must satisfy

$$\int \psi(y - f(\underline{x}, \underline{\theta}_G))\nabla_{\underline{\theta}} f(\underline{x}, \underline{\theta}_G)\ dG(\underline{x}, y) - \lambda \nabla_{\underline{\theta}} J(\underline{\theta}_G) \ = \ 0 \tag{B.5}$$

which can be expanded using (B.3) and re-arranged to give the equality

$$\begin{aligned} &\int \psi(y - f(\underline{x}, \underline{\theta}_G))\nabla_{\underline{\theta}} f(\underline{x}, \underline{\theta}_G)\ dH(\underline{x}, y) - \lambda \nabla_{\underline{\theta}} J(\underline{\theta}_G) \\ &= \ -\varepsilon \int \psi(y - f(\underline{x}, \underline{\theta}_G))\nabla_{\underline{\theta}} f(\underline{x}, \underline{\theta}_G)\ d\left(\delta_{\underline{x}_0, y_0} - H(\underline{x}, y) \right) . \end{aligned} \tag{B.6}$$

Differentiating with respect to ε on each side gives

$$\begin{aligned} &\int \Bigg(-\psi'(y - f(\underline{x}, \underline{\theta}_G))\nabla_{\underline{\theta}} f(\underline{x}, \underline{\theta}_G)\nabla_{\underline{\theta}} f(\underline{x}, \underline{\theta}_G)^T \frac{\partial \underline{\theta}_G}{\partial \varepsilon} + \\ &\psi(y - f(\underline{x}, \underline{\theta}_G))\nabla^2_{\underline{\theta}} f(\underline{x}, \underline{\theta}_G) \frac{\partial \underline{\theta}_G}{\partial \varepsilon} \Bigg) dH(\underline{x}, y) - \lambda \nabla^2_{\underline{\theta}} J(\underline{\theta}_G) \frac{\partial \underline{\theta}_G}{\partial \varepsilon} \end{aligned} \tag{B.7}$$

for the LHS of (B.6), and

$$\int \psi(y - f(\underline{x}, \underline{\theta}_G))\nabla_{\underline{\theta}} f(\underline{x}, \underline{\theta}_G)\ dH(\underline{x}, y) -$$

$$\psi(y_0 - f(\underset{\sim}{x}_0, \underset{\sim}{\theta}_G))\nabla_{\underset{\sim}{\theta}} f(\underset{\sim}{x}_0, \underset{\sim}{\theta}_G) + \varepsilon[\text{ other terms }] \tag{B.8}$$

for the RHS. Now letting $\varepsilon \rightarrow 0$ so that $G \rightarrow H$, (B.7) becomes

$$M \frac{\partial \underset{\sim}{\theta}_H}{\partial \varepsilon}\bigg|_{\varepsilon=0} \tag{B.9}$$

where M is the constant-valued matrix

$$\int \psi(y - f(\underset{\sim}{x}, \underset{\sim}{\theta}_H))\nabla^2_{\underset{\sim}{\theta}} f(\underset{\sim}{x}, \underset{\sim}{\theta}_H) - \psi'(y - f(\underset{\sim}{x}, \underset{\sim}{\theta}_H))\nabla_{\underset{\sim}{\theta}} f(\underset{\sim}{x}, \underset{\sim}{\theta}_H)\nabla_{\underset{\sim}{\theta}} f(\underset{\sim}{x}, \underset{\sim}{\theta}_H)^T \, dH(\underset{\sim}{x}, y)$$
$$-\lambda \nabla^2_{\underset{\sim}{\theta}} J(\underset{\sim}{\theta}_H) \tag{B.10}$$

and (B.8) can be further simplified using (B.2) to give

$$-\psi(y_0 - f(\underset{\sim}{x}_0, \underset{\sim}{\theta}_H))\nabla_{\underset{\sim}{\theta}} f(\underset{\sim}{x}_0, \underset{\sim}{\theta}_H) + \lambda \nabla_{\underset{\sim}{\theta}} J(\underset{\sim}{\theta}_H) \ . \tag{B.11}$$

Equating (B.9) and (B.11) and assuming M is non-singular, premultiplication by M^{-1} gives the IF

$$IF(\underset{\sim}{x}_0, y_0; \rho, \lambda J, H) \ = \ -\psi(y_0 - f(\underset{\sim}{x}_0, \underset{\sim}{\theta}_H))M^{-1}\nabla_{\underset{\sim}{\theta}} f(\underset{\sim}{x}_0, \underset{\sim}{\theta}_H) + \lambda M^{-1}\nabla_{\underset{\sim}{\theta}} J(\underset{\sim}{\theta}_H) \ . \tag{B.12}$$

Since the smoothing penalties described in chapter three do not depend on the training data, the second term in the IF is a constant. Thus this term is ignored henceforth. However, training with these penalties affects the shape of $f(\underset{\sim}{x}; \underset{\sim}{\theta}_H)$ and so they also affect the IF via the first term in (B.12).

B.3. Location estimators

Since the simplest IFs belong to single parameter location estimators, such as the mean and median, these will be used to show how the shape of an estimator's IF can be used to assess its resistance to outliers and sensitivity to small data perturbations. Only a brief overview is given here, and details can be found in[102, 129].

Single parameter location estimators[136, 214] can be considered a special case of regression where

$$f(\underset{\sim}{x}; \theta) \ = \ \theta \ , \tag{B.13}$$

that is, y does not depend on any $\underset{\sim}{x}$. Substituting this into (B.12) and ignoring the penalty term for now shows that a M-estimator's IF is proportional to its score function,

$$IF(y_0; \rho, H) \ \propto \ \psi(y_0 - \hat{\theta}) \ , \tag{B.14}$$

where $\underset{\sim}{\hat{\theta}}$ are the parameters estimated without the outlier.

Relationship between resistance and the IF

In the context of outlier resistance, the most important characteristic of the IF is its maximum absolute value,

$$\gamma^* = \sup_{y_0} |IF(y_0; \rho, H)| \tag{B.15}$$

which is known as the gross error sensitivity. This quantity is the largest (standardised) change in the estimator value that can be caused by introducing an outlier into the data. If γ^* is finite then an arbitrarily large outlier can only cause a limited change in the estimator value and so the estimator is resistant. It follows from (B.14) that the score function must be bounded for a resistant estimator. Figure B.1 illustrates this for the sample mean ($\psi(z) = z$, nonresistant) and the sample median ($\psi(z) = sgn(z)$, resistant).

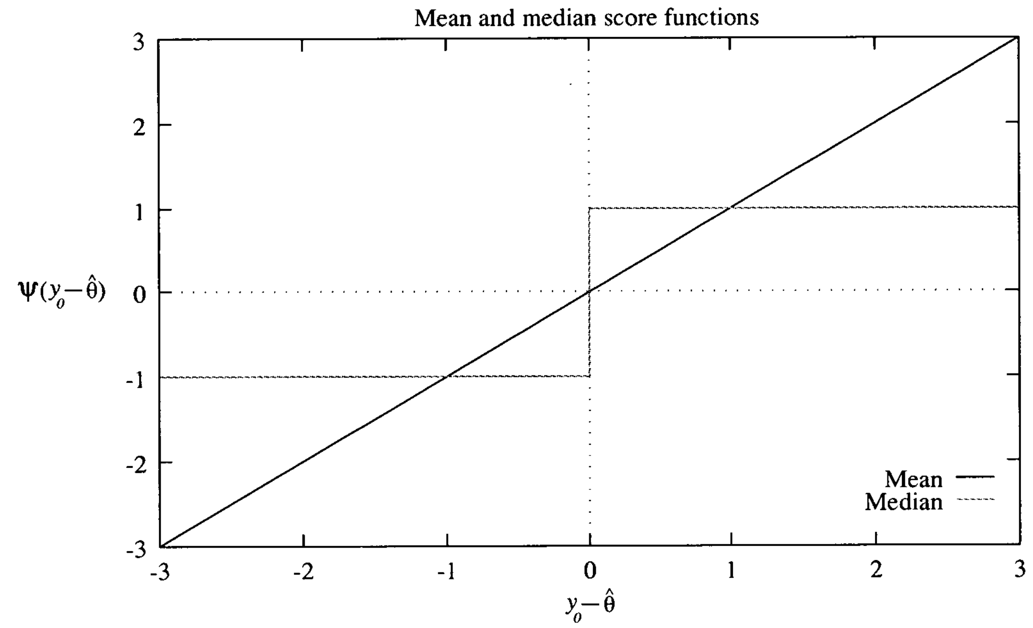


Figure B.1: Score functions for the sample mean and the sample median shown over the domain -3 to 3. The mean score function is unbounded.

Relationship between the IF and sensitivity to data perturbations

The derivative of the IF is known as the change-of-value function[102]. This function describes how sensitive the estimator value is to small perturbations of the data, such as those caused by numerical rounding. In chapter six the change of value function is used to explain how the fit leverages depend on the estimator used fit the MLP.

The maximum sensitivity is given by

$$\lambda^* = \sup_{\alpha \neq \beta} \frac{|IF(\alpha; \rho, H) - IF(\beta; \rho, H)|}{|\alpha - \beta|}$$

and known as the local shift sensitivity (lss). If the IF is continuous then the lss is the largest absolute value of its derivative. If the IF has discontinuities then the lss is infinite.

The sample mean and median provide good examples of how the change-of-value function describes an estimator's sensitivity to data perturbations. Figure B.1 shows that the sample mean IF is linear, and so the change-of-value function is a finite constant. This reflects the facts that

- perturbing any datum by a fixed amount causes the same change in the value of the mean no matter how far that datum lies from the mean (constant IF derivative), and
- the effect of perturbing only one datum is reduced as more data are averaged (finite lss).

In contrast, the median's change-of-value function is zero, except at $\hat{\theta}$, where it is infinite. This reflects that facts that

- the value of the sample median depends on only one or two data, surrounding $\hat{\theta}$, and so is insensitive to small perturbations of the other data (zero derivative except at $\hat{\theta}$), and
- because the median depends only on one or two data, its sensitivity to perturbations of these data does not decrease as the number of data increases (infinite lss).

B.4. Simple linear regression and leverage points

In problems involving more than one parameter, the IF is a vector quantity comprised of the individual influence functions for each parameter. To illustrate this, and to work towards examining the IF in MLP regression, consider first the simple linear regression model

$$f(x; \underline{\theta}) = \theta_0 + \theta_1 x \ . \tag{B.17}$$

For this model it can be shown easily that

$$IF \propto \psi(y_0 - f(x_0; \hat{\underline{\theta}})) \begin{pmatrix} 1 \\ x_0 \end{pmatrix} \tag{B.18}$$

by substituting (B.17) into (B.12) and ignoring λJ terms. The top component of this vector is the IF for the parameter θ_0 and the bottom component is the IF for θ_1.

Leverage and influence in linear regression

Note that the IF for the gradient parameter is proportional to x_0, and so now even if the score function is bounded (but non-zero), the influence of an outlier placed at an arbitrarily large x_0 may still be unbounded. Thus one of these ***high leverage*** outliers can strongly control the slope of the fitted line even if the majority of the data suggests a very different slope[19]. The fact the IF can depend on both $\underline{x}$ and y is an important issue in

[19] An example of this, which also illustrates why these are called high leverage outliers, is given in chapter five.

classical robust regression, and has motivated the development of various ***high breakdown*** estimators, which can resist one or more high leverage outliers[130]. Chapter five considers the usefulness of these estimators for training MLPs.

B.5. Influence in MLP regression

Having reviewed how

- an estimator's resistance can be related to the shape of its IF, and
- how influence in regression can depend on both $\underline{x}$ and y,

the remaining sections of this appendix look at the IF in MLP regression for a MLP of the type discussed in chapter two and used throughout this thesis. The aim of this examination is to determine if outliers, particularly high leverage outliers, can exhibit high influence in MLP regression, and if so, how seriously these outliers may affect how well the fit estimates the regression function.

For clarity, the IFs for the hidden to output and input to hidden weights will be examined separately. Ignoring any constant terms due to smoothing penalties,

$$IF \propto \psi(y_0 - f(\underline{x}_0, \hat{\underline{\theta}}))\nabla_{\underline{\theta}} f(\underline{x}_0, \hat{\underline{\theta}}) \ . \tag{B.19}$$

It will be assumed that a bounded $\psi(z)$ is used in an anticipation of outliers, and so the IF for a given parameter can have a large magnitude only if the associated component of $\nabla_{\underline{\theta}} f(\underline{x}_0; \hat{\underline{\theta}})$ has a large magnitude.

B.5.1. IFs for the hidden layer to outputs weights

For the weight v_u leading from hidden unit u to the linear output unit, the associated gradient component is

$$\left.\frac{\partial f(\underline{x}_0; \underline{\theta})}{\partial v_u}\right|_{\underline{\theta}=\hat{\underline{\theta}}} = \sigma_u \ , \tag{B.20}$$

where σ_u is the hidden unit output. For the sigmoidal type of hidden unit considered in this thesis, the output always lies between zero and one, and so the IFs for the hidden layer to output weights will always be bounded. Hence introducing large outliers into the training data can only cause limited changes in the hidden to output unit weights from those that would be obtained when training with the outlier-free data. These parameters are thus resistant to any outliers in the training data.

B.5.2. IFs for the input to hidden layer weights

The gradient component for the weight w_{uq} leading from input q to hidden unit u is

$$\left.\frac{\partial f(\underline{x}_0;\underline{\theta})}{\partial w_{uq}}\right|_{\underline{\theta}=\underline{\hat{\theta}}} = v_u \sigma_u' x_{0q} \ , \qquad \text{(B.21)}$$

where σ_u' is the derivative of the hidden unit output with respect to its activation. This output derivative depends implicitly on all the weights and inputs leading into the hidden unit, and this must be considered when determining the maximum possible influence.

Maximum influence when only one input becomes large

Liu notes that if sigmoidal hidden units are used, then robust MLP regression is resistant to a high leverage outlier caused by increasing (or decreasing) the value of only one input[151]. The reason for this is that a single sufficiently large input will dominate all hidden unit activations, and hence all hidden unit outputs and output derivatives. As the activation increases linearly, the output derivative goes to zero exponentially and so the product $\sigma_u' x_{0q}$ also goes to zero, thus bounding the IF. The IF may increase at first however, because the sigmoid derivative decays exponentially only for large activations (magnitude greater than about 2). Thus one question this raises is whether the maximum influence may still be large enough to affect the shape of the MLP fit seriously in some problems.

Maximum influence when two or more inputs are large

A further case which Liu does not consider is what happens if more than one input is increased (or decreased) to create a high leverage outlier. In this situation, any hidden unit output derivative can be kept constant (and non-zero) by changing the inputs in such a manner as to keep the hidden unit activation constant. Thus the IF for any input to hidden layer weight can be made arbitrarily large for some input variable combinations, and so the estimates of these weights are technically not resistant. Hence introducing large outliers into the training data could cause the weights to differ greatly from those which would be obtained when training with a similar data set without these outliers.

B.5.3. Interpreting the IF in MLP regression

The previous section demonstrated that outliers can still exhibit high influence in MLP robust regression when these occur at some extreme values of the predictor variables. However, assessing whether such outliers necessarily pose as serious a problem as they do in linear regression is difficult for two reasons.

Firstly, it must be remembered that the IF applies only to the model parameters, but it is usually the estimates of the regression function which are of primary interest in MLP regression[27]. To see why this makes assessing the importance of high leverage outliers difficult, recall that the output of a sigmoidal hidden unit always lies between zero and one. Thus large, outlier-induced changes in the hidden unit weights can only cause a bounded change in the MLP output. Whether this change is sufficient to render the model predictions worthless will depend on how many hidden units are affected, and what their overall contribution to the output is.

Secondly, it should also be noted that obtaining very high influence required the hidden unit activation to remain constant so that the output derivative did go not to zero. However, when the inputs are large then changing only one hidden weight slightly may increase the activation sufficiently that the output saturates and hence the output derivative becomes very small. Thus large weight changes are not necessarily required to reduce the influence of high leverage outliers, and so the fit may not change much as a result of introducing such data.

These difficulties in interpreting the effect of the parameter IFs on the fit are in fact quite general problems in regression when the fit is a nonlinear function of the parameters, and a more detailed discussion can be found in[217]. For this discussion, it is sufficient to conclude that high parameter influences are possible in MLP robust regression, but it is difficult to assess if and when these are likely to have an adverse effect on how well the fit estimates the regression function.

B.6. Example illustrating influence in MLP regression

The previous section showed that it is difficult to use the IF to assess the likely consequences of training a MLP with data containing high leverage outliers. Thus some simple experiments were conducted to investigate how such outliers may affect the fit in practice. This section discusses the results of one of these experiments and relates these results to the previous discussion of the IF. It also considers which results are most likely to important in practice.

Figure B.2 shows some samples of the regression function $7x(x+0.6)+1$, to which Gaussian response errors with standard deviation 0.25 have been added. Two outliers have also been introduced into the data, one at $x=-0.1$, and a second high leverage outlier at $x=1.2$. This figure also shows a quadratic polynomial fit to the data obtained using least absolute deviations (LAD) estimation in an attempt to resist any outliers in the data.[20] It is apparent that the fit is affected strongly by the high leverage outlier, to the

[20] LAD training, also known as L_1 estimation, is one of the robust regression methods discussed in chapter five.

extent of inverting the fitted parabola. This is confirmed by the fact that a much better fit is obtained when the high leverage outlier is removed, the LAD fitting method being able to limit the influence of the remaining low leverage outlier at $x = -0.1$.

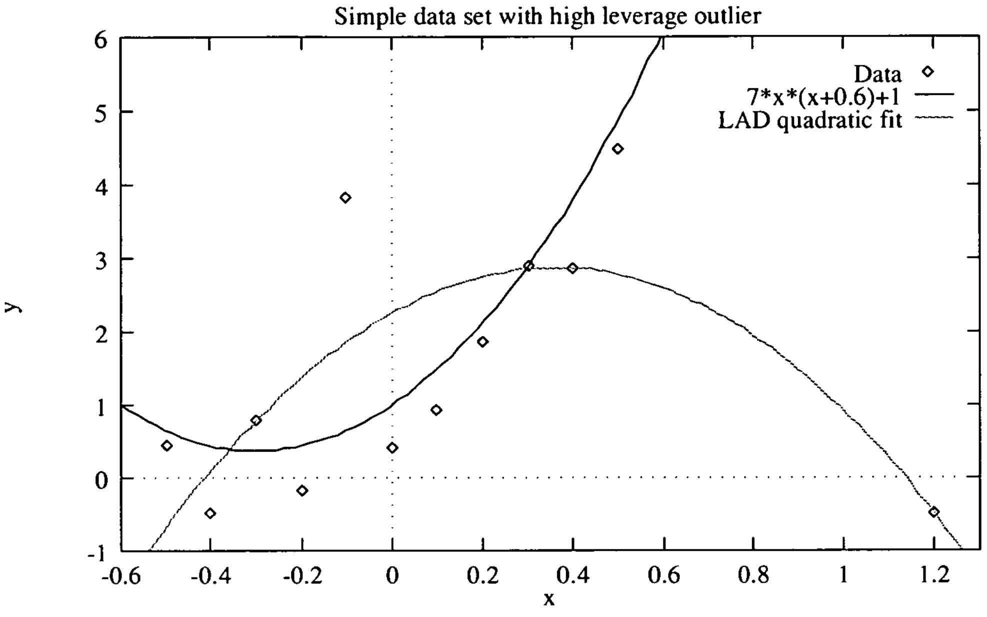


Figure B.2: The regression function and data for example one. A quadratic model fit to the data obtained by LAD estimation is also shown.

To investigate whether this outlier could cause similarly poor fits in MLP regression, MLPs with six hidden units were trained to estimate the regression function using LAD training. To control overfitting and to allow the effect of using a regularisation penalty to be considered, the spline penalty discussed in chapter three was used to control the fit complexity. The penalty was used with 77 roughness sampling points, spaced equally at steps of $\Delta x = 0.025$ over the interval $[-0.6, 1.3]$.

Figure B.3 shows some typical fits obtained using different amounts of spline regularisation. The amount of regularisation is controlled by the parameter λ in the manner described in chapter three, with larger λ forcing the fit closer to the linear regression fit.

The most interesting feature of this figure is that the outlier at $x = 1.2$ appears to be quite influential for all fits. This is evident from the fact that the all the fits are pulled towards this datum quite strongly, especially when λ is small.

The next sections discuss this behaviour in terms of the MLP IF, and also comment on which of the above fits are likely to be most important in practice.

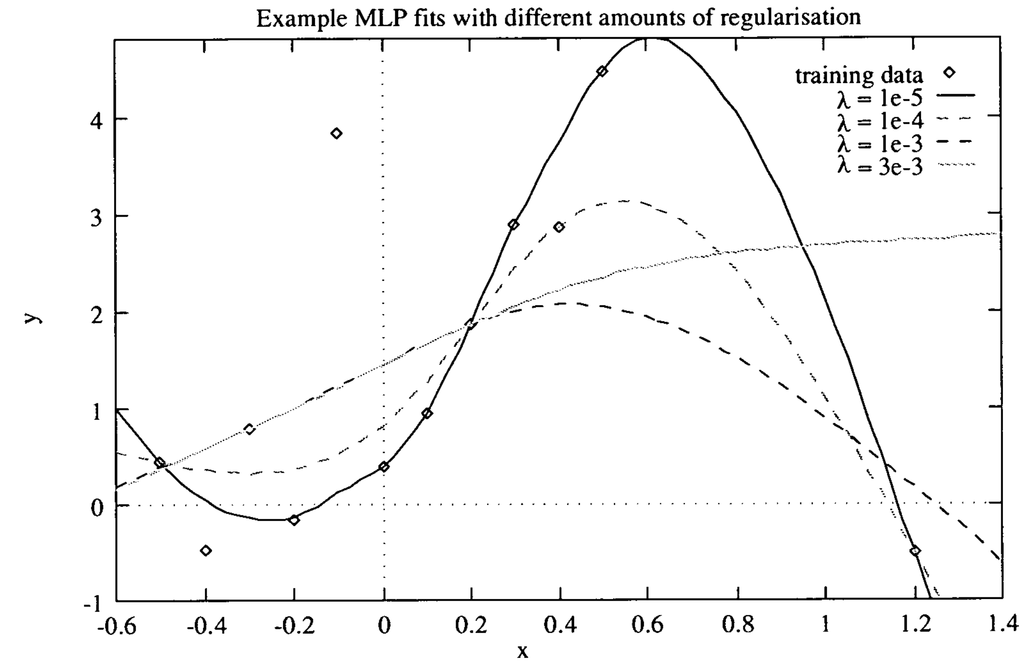


Figure B.3: Some typical MLP fits obtained using LAD training with different amounts of spline regularisation.

Large λ (very smooth fit)

When λ is large, the spline penalty forces the fit to be almost linear. This is achieved by constraining the input to hidden layer weights to small values so that the hidden unit activations are small. The results in only the approximately linear central section of the sigmoid curve being used to form the fit.

Forcing the hidden units to be linear over the range $x = -0.4$ to $x = 0.5$ also means, however, that the outlier at $x = 1.2$ is on, or near the end of, the linear section of the sigmoid transfer function. Thus the output derivative is large at the outlier, and so the outlier has a large influence on the input to hidden layer weights.

While this illustrates how a high leverage outlier can influence the overall fit in MLP regression, it is not likely to be important in practice. This is because forcing the fit to be almost linear will cause serious underfitting in most real problems, and so such fits will be rejected due to their large validation errors.

Small λ (less smooth fit)

For small λ, the fit divides into two distinct regions:

- a fit to the bulk of the data in the interval $[-0.5, 0.5]$, and
- the fit at the high leverage outlier.

As λ decreases, the fit to the bulk of the data improves (less underfitting) and the local influence of the high leverage outlier increases. The increase in influence causes the local fit near the outlier to be pulled closer to the outlier.

In terms of the MLP IF, reducing λ allows the hidden unit activations to increase so that the non-linear characteristics of some sigmoids can be used to improve the fit. This also means that their hidden unit activations will be large for the input $x = 1.2$, and so their outputs will be strongly saturated, reducing the influence of this datum. However, the same argument also applies to any hidden units which contribute to the fit at $x = 1.2$, namely the bulk of the data will have little influence on the values of the inputs weights leading from the MLP input to these units. Thus the fit near $x = 1.2$ is determined almost completely by the single outlier and so is forced to pass close to or through this point irrespective of its response value.

Since the fit to the majority of the data is good, the generalisation ability on a validation set is likely to be good, and so such fits are likely to be chosen by model selection procedures. However, this does not mean that high leverage data are not a cause for concern, as the next section explains.

Reliability of the fit near high leverage data

Since the fit near a high leverage datum is determined almost solely by this single datum, which may be a gross outlier, it must be considered extremely unreliable. The reason for this can be illustrated easily using kernel regression. In kernel regression, local domination of the fit by a single datum corresponds to all other data lying far into the tails of the kernel, and so receiving very small kernel weights. Thus the kernel fit at this location is essentially the average of only one datum, and so will have very wide confidence intervals (infinitely wide when the other data have absolutely no influence). This result would instantly give the data analyst reason to suspect that the fit may not be as good over all the regions of interest as the generalisation error estimated using a validation set may at first suggest.

Confidence intervals are not yet used widely in MLP regression, however, because they are more much difficult to generate than in kernel regression. Thus local overfitting near high leverage data may not be noticed in MLP regression. It was for the purpose of flagging such points to the data analyst that the investigation into estimating leverage in MLP regression discussed in chapter six was undertaken.

B.7. Summary

In this appendix, the IF was derived for a general nonlinear regression model and used to estimate the likely affects of outliers in robust MLP regression.

In particular, it was shown that high leverage outliers can still be very influential in MLP robust regression, but their influence will generally be limited to local control of the fit, rather than the type of global influence which causes breakdown in classical linear regression. This does not mean that these data do not pose serious problems in MLP regression. Since these data dominate the fit locally, the fit will be very poor in their vicinity if they lie far from the true regression function.

Since any data which strongly influence the fit either locally or globally are of interest to the data analyst, it is useful if such data be detected and flagged. Chapter six looks at one method for estimating leverages in MLP regression that can be used for this purpose.

Appendix C

Efficiency in MLP regression: additional results

C.1. Introduction

This appendix presents additional results from the Monte Carlo efficiency experiments described in chapter five. These results are given only to illustrate that the conclusions drawn from the single experiment discussed in chapter five do apply to other regression problems.

C.2. Additional results set A

The results discussed in this section were obtained using the regression function[98]

$$\begin{aligned}\mu(x_1, x_2) &= 1.37exp(-2.5((x_1 + 0.4)^2 + (x_2 + 0.4)^2)) \\ &+ 1.37exp(-2.5((x_1 - 0.4)^2 + (x_2 - 0.4)^2)) \qquad \text{(C.1)}\end{aligned}$$

sampled over the domain $-1 \le x_1, x_2 \le 1$ on a regular 10-by-10 grid. This is a harder function to estimate than used in chapter five. The response errors were drawn from

- a Gaussian distribution with mean 0 and variance 0.4,
- a Laplace distribution with mean 0 and variance 0.4, and
- a Gaussian distribution with mean 0 and variance 0.4 contaminated at the 10% level by a second Gaussian distribution with mean 0 and variance 2.

Results for data with Gaussian response errors

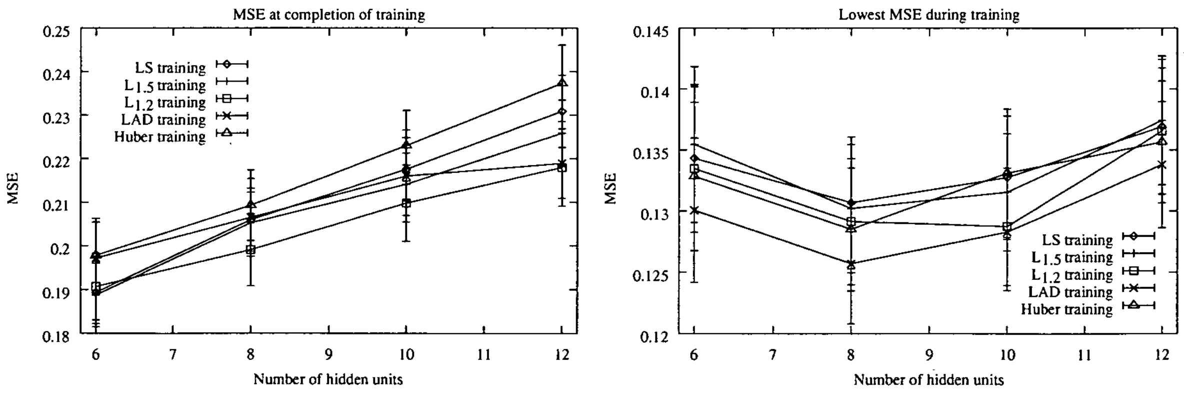


Figure C.1: MSE at end-of-training (left graph) and lowest MSE during training (right graph) for training data with Gaussian errors.

Results for data with Laplacian response errors

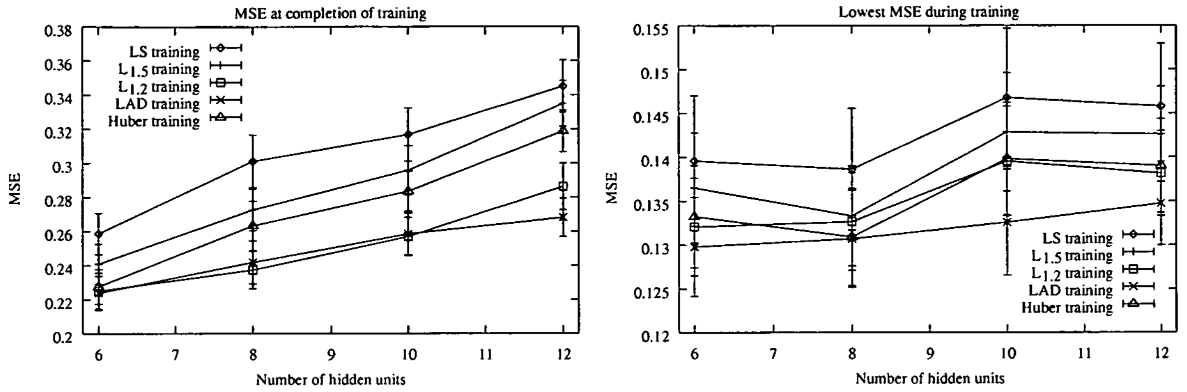


Figure C.2: MSE at end-of-training (left graph) and lowest MSE during training (right graph) for training data with Laplacian errors.

Results for data with contaminated normal response errors

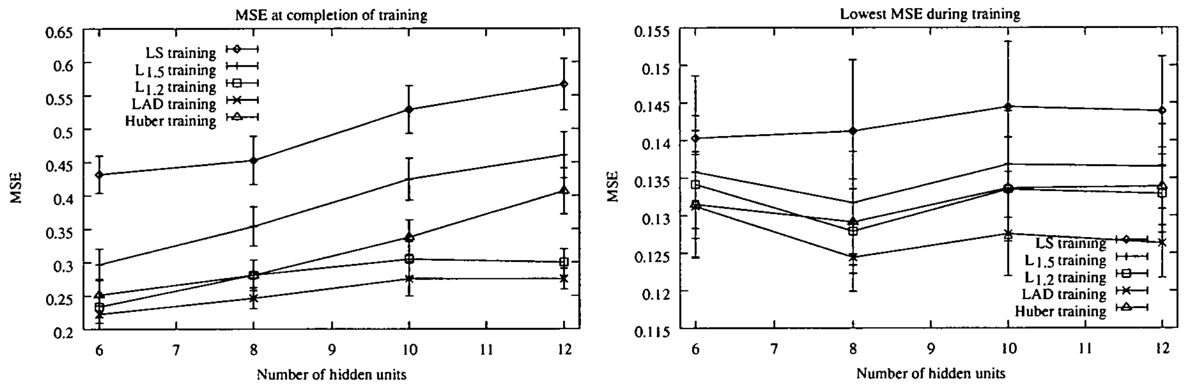


Figure C.3: MSE at end-of-training (left graph) and lowest MSE during training (right graph) for training data with contaminated normal errors.

Discussion of results

The trends shown by these results are similar to those shown in chapter five:

- As expected, the robust estimators give the lowest fit MSEs (i.e. they are the most efficient estimators) when the response error distribution has heavy tails.
- Good complexity control is required to achieve the lowest MSE (for example, compare the complete training and early stopping results).

As in chapter 5, the LAD, $L_{1.5}$ and Huber estimators give better efficiency than LS even when the response errors are Gaussian, though the wide confidence intervals give little significance to these results. It is again suspected that the lower MSEs are due to the slower overfitting that occurs when using these estimators.

C.3. Additional results set B

The results given in this section were obtained using the same regression function as used for the previous results, but with lower-variance response errors drawn from

- a Gaussian distribution with mean 0 and variance 0.2,
- a Laplace distribution with mean 0 and variance 0.2, and
- a Gaussian distribution with mean 0 and variance 0.2 contaminated at the 10% level by a second Gaussian distribution with mean 0 and variance 1.

Results for data with Gaussian response errors

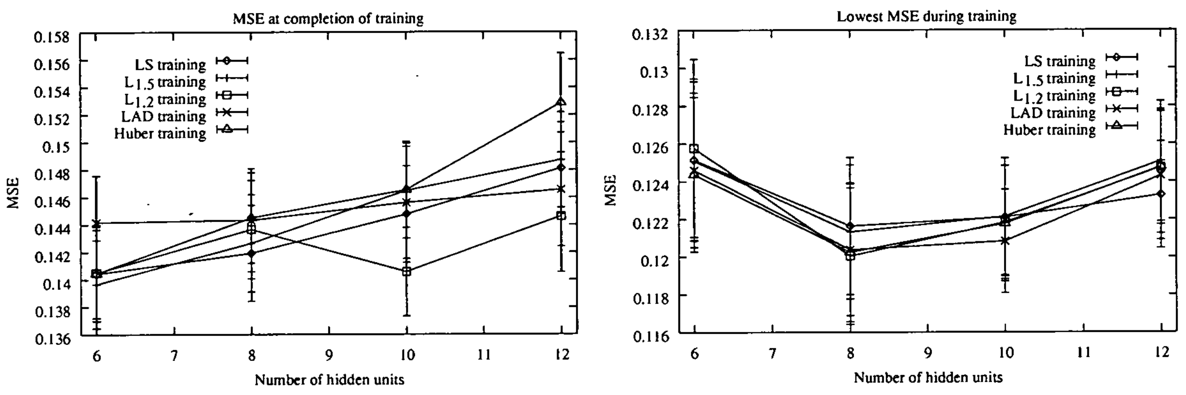


Figure C.4: MSE at end-of-training (left graph) and lowest MSE during training (right graph) for training data with Laplacian errors.

Results for data with Laplacian response errors

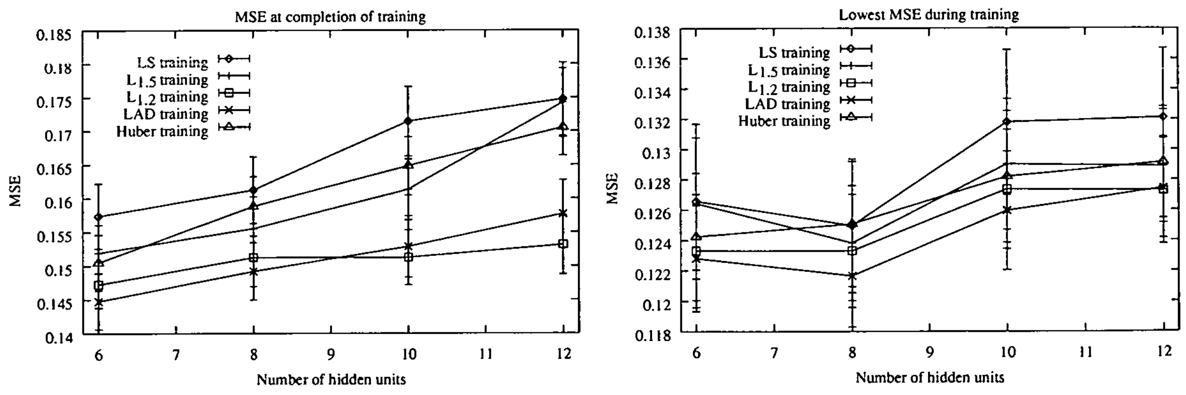
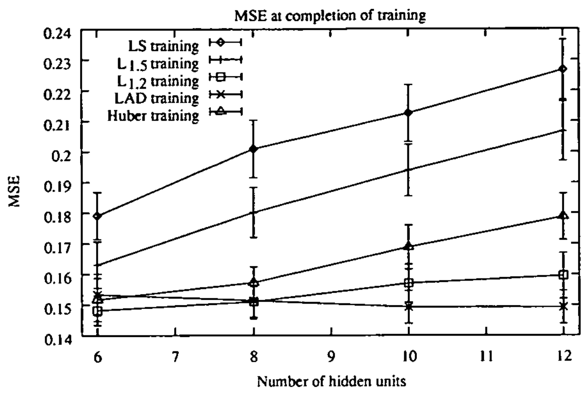


Figure C.5: MSE at end-of-training (left graph) and lowest MSE during training (right graph) for training data with Laplacian errors.

Results for data with contaminated normal response errors



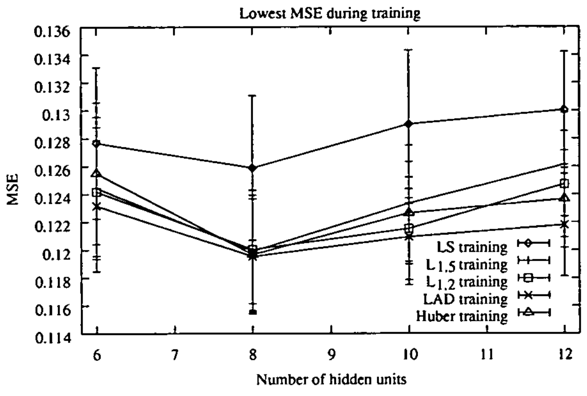


Figure C.6: MSE at end-of-training (left graph) and lowest MSE during training (right graph) for training data with contaminated normal errors.

Discussion of results

The trends shown in the above figures are very similar to those shown in section 2. The primary difference between these results is that now there is less MSE spread in each graph; it is easier for all estimators to achieve a good fit when the response errors are small.

C.4. Additional results set C

The results discussed in this section were obtained using the regression function[26]

$$\mu(x_1, x_2, x_3, x_4) = \frac{1}{20}\left(2.3log(x_1) + 0.6\sqrt{x_2} - \frac{1.7}{x_3} + 0.3x_4^2 + 3.2\log(x_1)\sqrt{x_2} + 0.8\frac{\log(x_1)}{x_3} - 1.9\frac{x_4^2}{x_3}\right) \quad (C.2)$$

This is quite a difficult regression function to estimate because of the interactions between the terms and the rapid increase in the function values for small x_3. Training data were generated by sampling the function 100 times at random positions within the interval $0 \le x_1, x_2, x_3, x_4 \le 1$. This limited number of training points gives sparse sampling of the regression function, and hence increases the difficulty of estimating it.

Response errors were drawn from

- a Gaussian distribution with mean 0 and variance 0.4,
- a Laplace distribution with mean 0 and variance 0.4, and
- a Gaussian distribution with mean 0 and variance 0.4 contaminated at the 10% level by a second Gaussian distribution with mean 0 and variance 1.

Results for data with Gaussian response errors

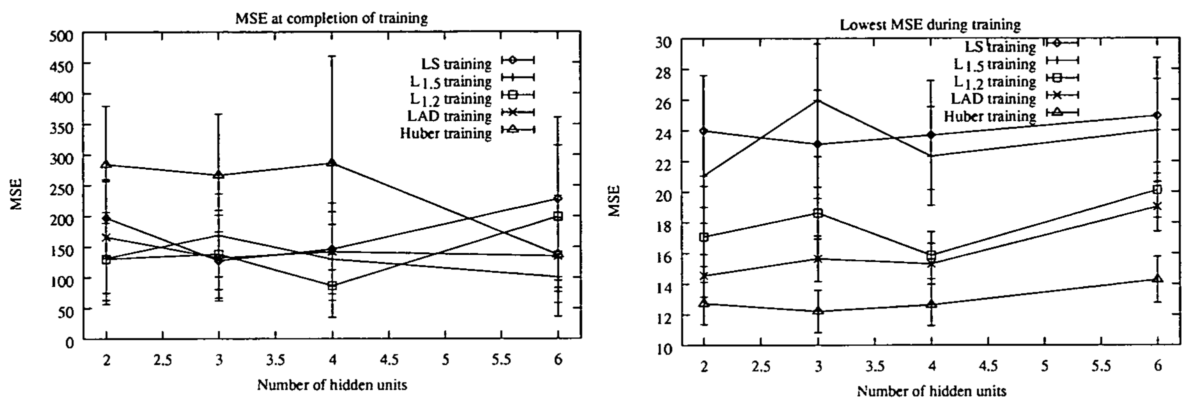


Figure C.7: MSE at end-of-training (left graph) and lowest MSE during training (right graph) for training data with Gaussian errors.

Results for data with Laplacian response errors

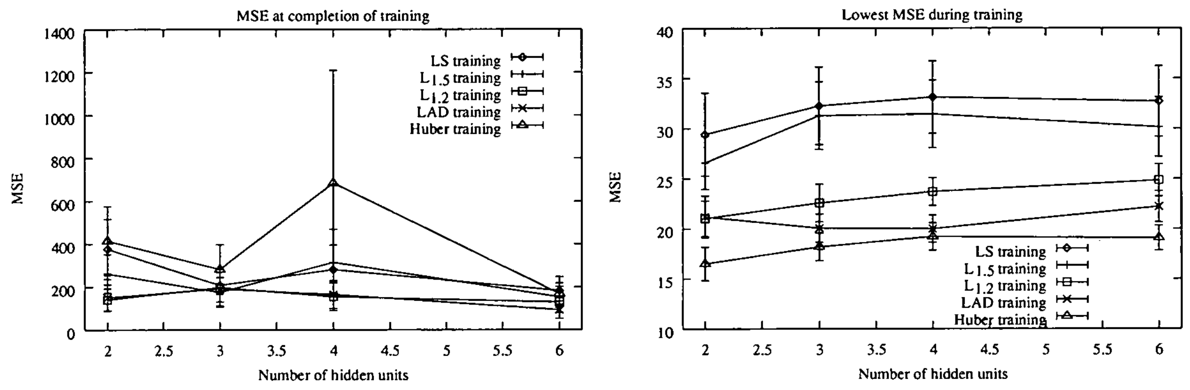


Figure C.8: MSE at end-of-training (left graph) and lowest MSE during training (right graph) for training data with Laplacian errors.

Results for data with contaminated normal response errors

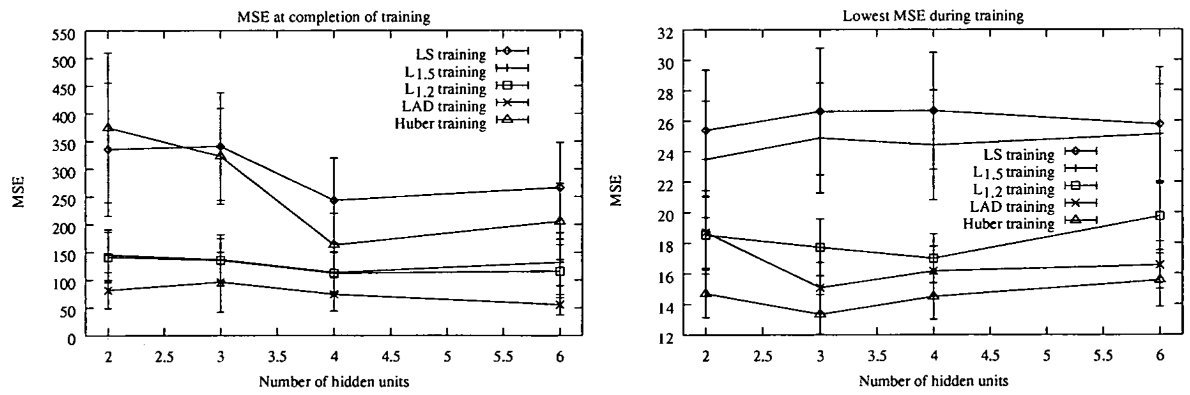


Figure C.9: MSE at end-of-training (left graph) and lowest MSE during training (right graph) for training data with contaminated normal errors.

Discussion of results

The difficulty of estimating this regression function is evident in the high MSEs for all estimators.

Again, the estimators show almost the same relative efficiencies as seen in chapter five and sections 2 and 3, with LS always being least efficient when using early stopping. The main difference from previous results is that Huber's estimator was slightly more efficient than the LAD estimator. The fits cannot be visualised because of the number of predictor variables, but some investigation suggested that the tendency of the Huber estimator to overfit more rapidly actually helps for this difficult problem, by allowing the peaks in the function near $x_3 = 0$ to be fitted quickly before overfitting to the errors occurs.

Plotting the median-square errors instead of the fit MSEs also indicated that the MSE was being dominated by a small proportion of the validation data. These are mostly likely the data with large response values near $x_3 = 0$. The median-square errors were much smaller than the MSEs, showing that most of the MSE value is due to fit errors from less than half of the data. The median-square errors otherwise showed the expected relative estimator efficiencies. Figure C.10 shows the median-square early stopping errors for the Laplacian data.

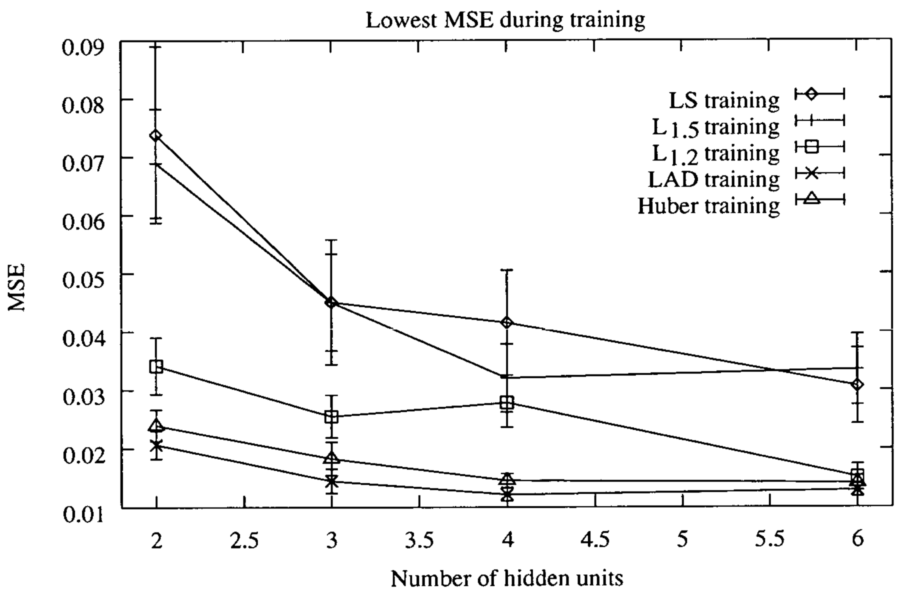


Figure C.10: Early stopping median-square errors for the Laplacian-error data. The median-square errors are much smaller than the MSEs.

Even for this complex, high-dimensional problems, the relative estimator efficiencies agree well with the results and conclusions of chapter five.

C.5. Summary

The results presented in this appendix confirm the general trends in estimator efficiency predicted in chapter five. In particular, it has again been shown that the rate at which overfitting occurs can affect efficiency as much as how well outliers can be resisted when the error distribution has long tails. This is why the most robust estimators (LAD, $L_{1.2}$, and Huber's) were more efficient than LS even when the errors were Gaussian.

Appendix D

MLP least squares tangent plane leverage

D.1. Introduction

This appendix contains a short derivation of the tangent plane leverage matrix given in chapter six for a MLP trained using least squares estimation with weight decay (LSWD). The leverages for weighted least squares (WLS) problems are also considered.

D.2. The tangent plane model and leverage

The tangent plane approximation[166, 172] for parametric models which are nonlinear in their parameters assumes that these models can be linearly approximated by

$$f(\underline{x};\underline{\theta}) \approx f(\underline{x};\hat{\underline{\theta}}_0)+\nabla_{\underline{\theta}} f(\underline{x};\hat{\underline{\theta}}_0)^T(\underline{\theta}-\hat{\underline{\theta}}_0) \tag{D.1}$$

for small parameter variations around the estimated parameters, $\hat{\underline{\theta}}_0$.

Suppose that the original vector of responses which was used to estimate $\hat{\underline{\theta}}_0$ is now perturbed to give a new set of response data, $\underline{Y} = (y_1 \cdots y_n)^T$. If the model is re-fitted using this data, then the new parameter estimates, $\hat{\underline{\theta}}$, must satisfy the LSWD criterion

$$\nabla_{\underline{\theta}}\left[(\underline{Y}-\hat{\underline{Y}})^T(\underline{Y}-\hat{\underline{Y}})+\lambda\hat{\underline{\theta}}^T\hat{\underline{\theta}}\right] = 0 \tag{D.2}$$

where $\hat{\underline{Y}} = (f(\underline{x}_1;\hat{\underline{\theta}}) \cdots f(\underline{x}_n;\hat{\underline{\theta}}))^T$. Expanding and simplifying (D.2) gives

$$-\nabla_{\underline{\theta}}\hat{\underline{Y}}^T\underline{Y}+\nabla_{\underline{\theta}}\hat{\underline{Y}}^T\hat{\underline{Y}}+\lambda\hat{\underline{\theta}} = 0 \tag{D.3}$$

and if $\hat{\underline{\theta}}$ is sufficiently close to $\hat{\underline{\theta}}_0$ for the approximation (D.1) to be valid, then

$$\hat{\underline{Y}} = \hat{\underline{Y}}_0+\hat{Z}(\hat{\underline{\theta}}-\hat{\underline{\theta}}_0) \tag{D.4}$$

where $\hat{\underline{Y}}_0$ is the vector of regression estimates obtained when $\underline{\theta}=\hat{\underline{\theta}}_0$ and $\hat{Z}$ is the Jacobian matrix with $(i, j)^{th}$ element

$$\hat{z}_{ij} = \left.\frac{\partial f(\underline{x}_i;\underline{\theta})}{\partial\theta_j}\right|_{\underline{\theta}=\hat{\underline{\theta}}_0}. \tag{D.5}$$

Substituting (D.4) into (D.3) gives

$$-\hat{Z}^T\underline{Y}+\hat{Z}^T\left(\hat{\underline{Y}}_0+Z(\hat{\underline{\theta}}-\hat{\underline{\theta}}_0)\right)+\lambda\hat{\underline{\theta}} = 0 \tag{D.6}$$

and this can be re-arranged to give the new parameter estimates,

$$\underline{\hat{\theta}} = \left(\hat{Z}^T\hat{Z}+\lambda I\right)^{-1}\hat{Z}^T\underline{Y}+\left(\hat{Z}^T\hat{Z}+\lambda I\right)^{-1}\hat{Z}^T\left(\hat{Z}\underline{\hat{\theta}}_0-\underline{\hat{Y}}_0\right). \tag{D.7}$$

The leverage matrix can now be found by substituting (D.7) back into (D.4). This gives

$$\underline{\hat{Y}} = \hat{Z}\left(\hat{Z}^T\hat{Z}+\lambda I\right)^{-1}\hat{Z}^T\underline{Y}+[\ \textit{other terms}\ ] \tag{D.8}$$

where the other terms do not depend on $\underline{Y}$. Thus perturbations of the response vector, $\Delta\underline{Y}$, will cause the regression estimates to change by

$$\Delta\underline{\hat{Y}} = \hat{Z}\left(\hat{Z}^T\hat{Z}+\lambda I\right)^{-1}\hat{Z}^T\Delta\underline{Y} \tag{D.9}$$

and so

$$H = \hat{Z}\left(\hat{Z}^T\hat{Z}+\lambda I\right)^{-1}\hat{Z}^T \tag{D.10}$$

is the tangent plane leverage matrix.

D.3. Leverage in weighted least squares estimation

WLS estimation is a variant of LS where the squared error for each observation is weighted by a constant[35, 36]. In matrix notation, the WLS estimation problem with weight decay can be written as

$$\nabla_{\underline{\theta}}\left[(\underline{Y}-\underline{\hat{Y}})^T W(\underline{Y}-\underline{\hat{Y}})+\lambda\underline{\hat{\theta}}^T\underline{\hat{\theta}}\right] = 0 \tag{D.11}$$

where W is the diagonal matrix of weights. Introducing this matrix adds little complexity to previous analysis, which can be repeated easily to give

$$H = W^{1/2}\hat{Z}\left(\hat{Z}^T W\hat{Z}+\lambda I\right)^{-1}\hat{Z}^T W^{1/2}. \tag{D.12}$$

Since many fast and numerically stable methods are known for computing LS estimates, one of the commonest uses of WLS is to compute the values of non-LS estimates using a process known as iteratively re-weighted LS (IRLS)[102, 133, 135, 169]. Here, W is defined to convert the non-LS problem into a LS problem, which is then solved to give a set of parameter estimates. These estimates are then used to modify W and the process alternates between estimating the parameters and updating W until both the weights and parameters converge.

For example, L_p-Norm estimation can be implemented using the weight matrix

$$W = \begin{pmatrix} |r_1|^{p-2} & 0 & \cdots & 0 \\ 0 & |r_2|^{p-2} & \cdots & . \\ . & . & \cdots & 0 \\ 0 & 0 & \cdots & |r_n|^{p-2} \end{pmatrix} \quad \text{(D.13)}$$

where $r_i = y_i - f(\underline{x}_i; \hat{\underline{\theta}})$ and $\hat{\underline{\theta}}$ is the current value of the parameter estimates.

Though IRLS is widely used for computing non-LS estimates, (D.12) generally cannot be used to obtain the correct leverages for these estimators. The reason for this is that the weights are not constants, but this is assumed in the derivation of (D.12). For example, comparing the WLS tangent plane matrix for L_p-Norm with the correct Jacobian matrix given in chapter six shows that the WLS matrix lacks important $p(p-1)$ factors on some terms.

Appendix E

References

References

1. D. Hammerstrom, "Neural networks at work", *IEEE Spectrum*, pp. 26-32, June 1993.
2. D. Hammerstrom, "Working with neural networks", *IEEE Spectrum*, pp. 46-53, July 1993.
3. R. Hecht-Nielsen, "Neurocomputing: picking the human brain", *IEEE Spectrum*, vol. 25, no. 3, pp. 36-41, 1988.
4. R.P. Lippmann, "An Introduction to Computing With Neural Nets", *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4-22, 1987.
5. R. Beale and T. Jackson, *Neural computing: an introduction,* IOP Publishing, 1990. ISBN 0-85274-262-2
6. V. Vemuri, *Artificial Neural Networks: An Introduction,* pp. 1-12, IEEE Computer Society Press, 1988. ISBN 0-8186-0855-2
7. T. Kohonen, "Artificial Neural Networks: Models, Paradigms, or Methods?", in *Artificial Neural Networks 2*, ed. I. Aleksander and J. Taylor, vol. 1, pp. 3-10, North-Holland, 1992. ISBN 0 444 89488 8
8. M.J. Willis, G.A. Montague, and A.J. Morris, "Modelling of industrial processes using artificial neural networks", *IEE Computing & Control Engineering Journal*, pp. 113-117, 1992.
9. K.S. Narendra and K. Parthasarthy, "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.
10. T. van der Walt, E. Barnard, and J. van Deventer, "Process Modelling with the Regression Network", *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 78-93, 1995.

11. G.S. May, "Manufacturing ICs the Neural Way", *IEEE Spectrum*, vol. 31, no. 4, 1994.

12. N.V. Bhat, P.A. Minderman, and T. McAvoy, "Modelling Chemical Process Systems via Neural Computation", *IEEE Control Systems*, pp. 24-30, April 1990.

13. A. Bulsari and H. Saxen, "System identification of a biochemical process using feed-forward neural networks", *Neurocomputing*, vol. 3, pp. 125-133, 1991.

14. A.C. Tsoi, "Application of Neural Network Methodology to the Modelling of the Yield Strength in a Steel Rolling Plate Mill", in *Advances in Neural Information Processing Systems 4*, ed. R.P. Lippmann, J.E. Moody, and S.J. Hanson, pp. 698-705, Morgan Kaufmann Publishers Inc, 1992. ISBN 1-55860-222-4

15. A.S. Weigend and N.A. Gershenfeld, *Time Series Prediction,* Addison-Wesley Publishing Company, 1994. ISBN 0-201-62602-0

16. H. White, "Option pricing in modern finance theory and the relevance of artificial neural networks", *NIPS tutorial program*, Denver, 1995.

17. R.G. Hoptroff, "The Principles and Practice of Time Series Forecasting and Business Modelling Using Neural Networks", *Neural Computing and Applications*, vol. 1, no. 1, pp. 59-66, 1993.

18. A.N. Refenes, M. Azema-Barac, L. Chen, and S.A. Karoussos, "Currency Exchange Rate Prediction and Neural Network Design Strategies", *Neural Computing and Applications*, vol. 1, no. 1, pp. 46-58, 1993.

19. C.N. Lu, H.T. Wu, and S. Vemuri, "Neural Network Based Short-Term Load Forecasting", *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 336-342, 1993.

20. A.D. Papalexopoulos, S. Hao, and T.M. Peng, "An Implementation of a Neural Network Based Load Forecasting Model for the EMS", *IEEE Transactions on Power Systems*, vol. 9, no. 4, pp. 1956-1962, 1994.

21. J.T. Connor, R.D. Martin, and L.E. Atlas, "Recurrent Neural Networks and Robust Time Series Prediction", *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 240-254, 1994.

22. B. Cheng and D.M. Titterington, "Neural Networks: A View from a Statistical Perspective (with Discussion)", *Statistical Science*, vol. 9, no. 1, pp. 2-54, 1994.

23. H. White, "Some Asymptotic Results for Learning in Single Hidden-Layer Feedforward Network Models", *Journal of the American Statistical Association*, vol. 84, no. 408, pp. 1003-1013, 1989.

24. W.S. Sarle, "Neural Networks and Statistical Models", *Proceedings of the Nineteenth Annual SAS Users Group International Conference, Cary, NC*, pp. 1538-1550, SAS Institute, 1994.

25. B.D. Ripley, "Neural Networks and Related Methods for Classification", *Journal of The Royal Statistical Society*, vol. 56, no. 3, pp. 409-456, 1994.

26. D. Faraggi and R. Simon, "Maximum Likelihood Neural Network Prediction Models", *Biometrical Journal*, vol. 37, pp. 713-725, 1995.

27. H.S. Stern, "Neural Networks in Applied Statistics", *Technometrics*, vol. 38, no. 3, pp. 205-220, 1996.

28. B. Warner and M. Misra, "Understanding Neural Networks as Statistical Tools", *The American Statistcian*, vol. 50, no. 4, pp. 284-293, 1996.

29. J. Friedman, "Introduction to Computational Learning and Statistical Prediction", *NIPS tutorial program*, Denver, 1995.

30. W. Hardle, *Applied nonparametric regression,* Cambridge University Press, 1990. ISBN 0-521-38248-3

31. W.S. Sarle, "Stopped Training and Other Remedies for Overfitting", in *Statistics and Computing: 27th Symposium on the Interface*, 1995. To appear

32. G.A.F. Seber and C.J. Wild, *Nonlinear Regression,* John Wiley & Sons, Inc., 1989. ISBN 0-471-61760-1

33. M.J. Korenberg and L.D. Paarmann, "Orthogonal Approaches to Time-Series Analysis and System Identification", *IEEE Signal Processing Magazine*, vol. 8, no. 3, pp. 29-43, 1991.

34. M.G. Kendall and W.R. Buckland, *A Dictionary of Statistical Terms,* Oliver & Boyd, 1971. ISBN 0 05 002280 6

35. N.R. Draper and H. Smith, *Applied Regression Analysis (2nd Edition),* John Wiley and Sons, Inc., 1981. ISBN 0-471-02995-5

36. J.O. Rawlings, *Applied Regression Analysis,* Wadsworth & Brooks, 1988. ISBN 0-534-09246-2

37. A.R. Gallant, *Nonlinear Statistical Models,* John Wiley & Sons, Inc., 1987. ISBN 0-471-80260-3

38. R.L. Eubank, *Spline Smoothing and Nonparametric Regression,* Marcel Dekker, Inc., 1988. ISBN 0-8247-7869-3

39. R.D. Cook, "On the Interpretation of Regression Plots", *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 177-189, 1994.

40. N.S. Altman, "Introduction to Kernel and Nearest-Neighbour Nonparametric Regression", *The American Statistician*, vol. 46, no. 3, pp. 175-185, 1992.

41. B.D. Ripley, *Pattern Recognition and Neural Networks,* Cambridge University Press, 1996. ISBN 0 521 46086 7

42. M.E. Tarter and M.D. Lock, *Model-Free Curve Estimation,* Chapman and Hall, 1993. ISBN 0-412-04251-7-5

43. T.J. Hastie and R.J. Tibshirani, *Generalised Additive Models,* Chapman and Hall, 1990. ISBN 0-412-34390-8

44. W.S. Cleveland and B. Kleiner, "A Graphical Technique for Enhancing Scatterplots with Moving Statistics", *Technometrics*, vol. 17, no. 4, pp. 447-454, 1975.

45. F.T. Wang and D.W. Scott, "The L_1 Method for Robust Nonparametric Regression", *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 65-76, 1994.

46. C.H. Reinsch, "Smoothing by Spline Functions", *Numerische Mathematik*, vol. 10, pp. 177-183, 1967.

47. R.L. Eubank, "A simple Smoothing Spline", *The American Statistician*, vol. 48, no. 2, pp. 103-106, 1994.

48. J.H. Friedman and B.W. Silverman, "Flexible Parsimonious Smoothing and Additive Modelling", *Technometrics*, vol. 31, no. 1, pp. 3-21, 1989.

49. S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilemma", *Neural Computation*, no. 4, pp. 1-58, 1992.

50. R. Bellman, *Adaptive Control Processes: A guided Tour,* Princeton University Press, 1961. L.C. Card 60-5740

51. P. Thrift, "Neural Networks and Nonlinear Modeling", *TI Technical Journal*, vol. 7, no. 6, pp. 16-21, Nov-Dec 1990.

52. D.R. Hush and B.G. Horne, "Progress in Supervised Neural Networks. What's New Since Lippmann?", *IEEE Signal Processing Magazine*, pp. 8-39, Jan. 1993.

53. K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Neural Networks are Universal Approximators", *Neural Networks*, vol. 2, pp. 359-366, 1989.

54. E.M. Beale, *Introduction to Optimisation,* John Wiley & Sons, 1988. ISBN 0 471 91760 5

55. B.S. Everitt, *Introduction to optimization methods and their application in statistics,* Chapman and Hall, 1987. ISBN 0-412-27210-S

56. R.A. Thisted, *Elements of Statistical Computing,* Chapman and Hall, 1988. ISBN 0-412-01371-1

57. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning representations by back-propagating errors", *Nature*, vol. 323, pp. 533-536, 1986.

58. R. Battiti, "First- and Second-Order methods for Learning: between Steepest Descent and Newton's Method", *Neural Computation*, vol. 4, pp. 141-166, 1992.

59. E.M. Johansson, F.U. Dowla, and D.M. Goodman, "Backpropagation Learning for Multilayer Feed-Forward Neural Networks using the Conjugate Gradient Method", *International Journal of Neural Systems*, vol. 2, no. 4, pp. 291-301, 1992.

60. M.F. Moller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", *Neural networks*, vol. 6, pp. 525-533, 1993.

61. H.S.M. Beigi and C.J. Li, "Learning Algorithms for Neural Networks Based on Quasi-Newton Methods with Self-Scaling", *Transactions of the ASME*, vol. 115, pp. 38-43, 1993.

62. M.J. Box, D. Davies, and W.H. Swann, *Non-linear Optimisation techniques,* Oliver & Boyd, 1969. ISBN 05 002128 1

63. J.E. Vitela and J. Reifman, "Premature Saturation in Backpropagation Networks: Mechanism and Necessary Conditions", *Neural Networks*, vol. 10, no. 4, pp. 721-735, 1997.

64. B. Efron, *An introduction to the bootstrap,* Chapman & Hall, Inc., 1993. ISBN 0-412-04231-2

65. R.D. Snee, "Validation of Regression Models: Methods and Examples", *Technometrics*, vol. 19, no. 4, pp. 415-428, 1977.

66. R.R. Picard and R.D. Cook, "Cross-Validation of Regression Models", *Journal of the American Statistical Association*, vol. 387, no. 79, pp. 575-583, 1984.

67. C.M. Bishop, *Neural Networks for Pattern Recognition,* Clarendon Press, Oxford, 1995. ISBN 0 19 853864 2

68. D. Barber, D. Saad, and P. Sollich, "Test Error Fluctuations in Finite Linear Perceptrons", *Neural Computation*, vol. 7, no. 4, pp. 809-821, 1995.

69. S. Amari, N. Murata, K.R. Muller, M. Finke, and H. Yang, "Asymptotic Statistical Theory of Overtraining and Cross-Validation", METR 95-06, University of Tokyo Physics Department, 1995.

70. W. Schoner, "Reaching the Generalisation Maximum of Backpropagation Networks", in *Artificial Neural Networks 2*, ed. I. Aleksander and J. Taylor, vol. 2, pp. 91-94, North-Holland, 1992. ISBN 0 444 89488 8

71. N. Morgan and H. Bourlard, "Generalisation and Parameter Estimation in Feedforward Nets: Some Experiments", in *Advances in Neural Information Processing Systems 2*, ed. D.S. Touretzky, pp. 630-637, Morgan Kaufmann Publishers Inc, 1992. ISBN 1-55860-100-7

72. A.N. Tikhonov and V.Y. Arsenin, *Solutions of Ill-Posed Problems,* John Wiley & Sons, Washington DC, 1977. ISBN 0-470-99124-0

73. F. O'Sullivan, "A Statistical Perspective on Ill-Posed Inverse Problems", *Statistical Science*, vol. 1, no. 4, pp. 502-527, 1986.

74. J. Moody, "The Effective Number of Parameters: An Analysis of Generalisation and Regularisation in Nonlinear Learning Systems", in *Advances in Neural Information Processing Systems, 4*, ed. J.E. Moody, S.J. hanson, and R.P. Lippman, pp. 847-854, Morgan Kaufmann, Inc., 1992. ISBN 1-55860-222-4

75. P.J. Green, "Penalised Likelihood for General Semi-parametric Regression Models", *International Statistical Review*, vol. 55, no. 3, pp. 245-259, 1987.

76. D.M. Titterington, "Common Structure of Smoothing Techniques in Statistics", *International Statistical Review*, vol. 53, no. 2, pp. 141-170, 1985.

77. R. Reed, R.J. Marks, and S. Oh, "Similarities of Error Regularization, Sigmoid Gain Scaling, Target Smoothing, and Training with Jitter", *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 529-538, 1995.

78. Y. Chauvin, "Generalization Performance and Overtrained Back-Propagation Networks", in *Neural Networks: EURASIP workshop*, ed. L.B. Almeida and C.J. Wellekens, pp. 46-55, 1990. ISBN 3-54-052255-7

79. C.M. Bishop, "Curvature-Driven Smoothing in Feedforward Networks", *preprint.*

80. C.M. Bishop, "Curvature-Driven Smoothing: A Learning Algorithm for Feedforward Networks", *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 882-884, 1993.

81. A.A. Minai, *The Robustness of Feed-Forward neural Networks: A Preliminary Investigation,* University of Virginia, 1992. Doctoral Thesis

82. T.K. Leen, "From Data Distributions to Regularization in Invariant Learning", *Neural Computation*, vol. 7, pp. 974-981, 1995.

83. A. Krogh and J.A. Hertz, "A Simple Weight Decay Can Improve Generalization", in *Advances in Neural Information Processing Systems 4*, ed. R.P. Lippmann, J.E. Moody, and S.J. Hanson, pp. 950-957, Morgan Kaufmann Publishers Inc, 1992.

84. A.S. Weigend, D.E. Rumelhart, and B.A. Huberman, "Generalization by Weight Elimination with Application to Forecasting", in *Advances in Neural Information Processing Systems 3*, ed. R.P. Lippmann, J.E. Moody, and D.S. Touretzky, pp. 875-882, Morgan Kaufmann Publishers Inc, 1991.

85. I.J. Good and R.A. Gaskins, "Nonparametric Roughness Penalties for Probability Densities", *Biometrika*, vol. 58, no. 2, pp. 255-277, 1971.

86. P.J. Green and B.W. Silverman, *Nonparametric Regression and Generalised Linear Models - A Roughness Penalty Approach,* Chapman and hall, 1994. ISBN 0 412 30040 0

87. B.W. Silverman, "Some Aspects of the Spline Smoothing Approach to Non-parametric Regression Curve Fitting", *Journal of The Royal Statistical Society*, vol. 47, no. 1, pp. 1-52, 1985.

88. G. Wahba and J. Wendelberger, "Some New Mathematical Methods for Variational Objective Analysis Using Splines and Cross Validation", *Monthly Weather Review*, vol. 108, pp. 1122-1143, 1980.

89. R.V. Lenth, "Robust Splines", *Communications in Statistics: Theory and Methods*, vol. A6, no. 9, pp. 847-854, 1977.

90. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: the art of scientific computing,* Cambridge University Press, 1992. ISBN 0-521-43108-5

91. P.J. Huber, "Robust Smoothing", in *Robustness in Statistics*, ed. R.L. Launer and G.N. Wilkinson, pp. 33-47, Academic Press, Inc, 1979. ISBN 0-12-438150-2

92. J.E. Moody and T.S. Rognvaldsson, "Smoothing Regularizers for Projective Basis Function Networks", OGI CSE Technical Report 96-006. Oregon Graduate Institute Dept. of Computer Science and Engineering

93. L.C. Ludeman, *Fundamentals of Digital Signal Processing,* John Wiley & Sons, Inc., 1987. ISBN 0-471-61306-1

94. J. Meinguet, "Multivariate Interpolation at Arbitrary Points Made Simple", *Journal of Applied Mathematics and Physics (ZAMP)*, vol. 30, pp. 292-304, 1979.

95. T. Hastie and R. Tibshirani, "Generalised Additive Models: Some Applications", *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 371-386, 1987.

96. T. Hastie and R. Tibshirani, "Generalised Additive Models", *Statistical Science*, vol. 1, no. 3, pp. 297-318, 1986.

97. G. Wahba, "Partial and Interaction Spline Models for the Semiparametric Estimation of Functions of Several Variables", *Computer Science and Statistics: Proceedings of the 18th Symposium on the Interface*, pp. 75-80, American Statistical Association, Washington DC, 1986.

98. L. Breiman, "The Π Method for Estimating Multivariate Functions From Noisy Data", *Technometrics*, vol. 33, no. 2, pp. 125-160, 1991.

99. K. Matsuoka, "Noise Injection into Inputs in Back-Propagation Learning", *IEEE transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 436-440, 1992.

100. G. An, "The Effects of Adding Noise During Backpropagation Training on a Generalization Performance", *Neural Computation*, vol. 8, no. 3, pp. 643-674, 1996.

101. C.M. Bishop, "Training with Noise is Equivalent to Tikhonov Regularisation", *Neural Computation*, vol. 7, pp. 108-116, 1995.

102. C. Goodall, "M-estimators of Location: An Outline of the Theory", in *Understanding Robust and Exploratory Data Analysis*, ed. D.C. Hoaglin, F. Mosteller, and J.W. Tukey, pp. 339-403, John Wiley and Sons, Inc, 1983. ISBN 0-471-09777-2

103. R. Reed, "Pruning Algorithms - A Survey", *IEEE Trans. on Neural Networks*, vol. 4, no. 5, pp. 740-747, 1993.

104. J. Hertz, A. Krough, and R.G. Palmer, in *Introduction to the Theory of Neural Computation*, pp. 176-188, Addison-Wesley Publishing Company, 1991. ISBN 0-201-51560-1

105. S. Weisberg, *Applied Linear Regression,* John Wiley and Sons, Inc., 1980. ISBN 0-471-04419-9

106. H.D. Vinod and A. Ullah, *Recent Advances in Regression Methods,* Marcel Dekker, Inc, 1981. ISBN 0-8247-1574-8

107. D.W. Marquardt, "Generalised Inverses, Ridge Regression, Biased Linear Estimation, and Nonlinear Estimation", *Technometrics*, vol. 12, no. 3, pp. 591-612, 1970.

108. J.E. Jackson, *A user's guide to principal components,* John Wiley & Sons, Inc., 1991. ISBN 0-471-62267-2

109. D. DeMers and G. Cottrel, "Non-Linear Dimensionality Reduction", in *Advances in Neural Information Processing Systems 5*, ed. S.J. Hanson, J.D. Cowan, and C.L. Giles, pp. 580-587, Morgan Kaufmann Publishers Inc, 1993. ISBN 1-55860-274-4

110. P.J. Brown, "Centering and scaling in ridge regression", *Technometrics*, vol. 19, no. 1, pp. 35-36, 1997.

111. G. Smith and F. Campbell, "A Critique of Some Ridge Regression Methods", *Journal of the American Statistical Association*, vol. 75, no. 369, pp. 75-103, 1980.

112. I.E. Frank and J. Friedman, "A Statistical View of Some Chemometrics Regression Tools", *Technometrics*, vol. 35, no. 2, pp. 109-148, 1993.

113. V. Kurkova and P.C. Kainen, "Functionally Equivalent Feedforward Neural Networks", *Neural Computation*, vol. 3, no. 4, pp. 543-558, 1994.

114. H.J. Sussman, "Uniqueness of the Weights for Minimal Feedforward Nets with a Given Input-Output Map", *Neural Networks*, vol. 5, pp. 589-593, 1992.

115. E.T. Langevin and W. Giguere, "On Line Curl Measurement and Control", *Preprint (presented at TAPPI Polymers, Lamination & Coatings)*, 1994.

116. L-E. Eriksson, S. Cavlin, C. Fellers, and L. Carlsson, "Curl and twist of paperboard - theory and measurement", *Nordic Pulp and Paper Research Journal*, vol. 2, no. 2, pp. 66-70, 1987.

117. M.B. Lyne, "Paper Requirements for Non-Impact", *International Printing and Graphic Arts Conference Proceedings*, pp. 89-97, TAPPI Press, 1988.

118. R. LeBel and M. Stradal, "Control of fine paper curl in papermaking", *Pulp & Paper Canada*, vol. 83, no. 6, pp. 112-117, 1982.

119. R.J.A. Little, "Regression With Missing X's: A Review", *Journal of the American Statistical Association*, vol. 87, no. 420, pp. 1227-1237, 1992.

120. R.J.A. Little and D.B. Rubin, *Statistical Analysis with Missing Data,* John Wiley and Sons, Inc, 1987. ISBN 0-471-80254-9

121. D.B. Rubin, *Multiple Imputation for Nonresponse in Surveys,* John Wiley and Sons, Inc, 1987. ISBN 0-471-80705-X

122. D.C. Hoaglin, F. Mosteller, and J.W. Tukey, *Understanding Robust and Exploratory Data Analysis,* John Wiley and Sons, Inc, 1983. ISBN 0-471-09777-2

123. V. Tresp, S. Ahmad, and R. Neuneier, "Training Neural Networks with Deficient Data", in *Advances in Neural Information Processing Systems, 6*, ed. J.D. Cowan, G. Tesauro, and J. Alspector, pp. 128-135, Morgan Kaufmann Publishers, Inc, 1994. ISBN 1-55860-322-0

124. R. Gonin and A.H. Money, *Nonlinear L_p-Norm Estimation,* Marcel Dekker, Inc, 1989. ISBN 0-8247-8125-2

125. D.A. Ratkowsky, *Nonlinear Regression Modelling,* Marcel Dekker, Inc., 1983. ISBN 0-8247-1907-7

126. R.A. Gallant, "Nonlinear Regression", *The American Statistician*, vol. 29, no. 1, pp. 73-81, 1975.

127. J.R. Donaldson and R.B. Schnabel, "Computational Experience With Confidence Regions and Confidence Intervals for Nonlinear Least Squares", *Technometrics*, vol. 29, no. 1, pp. 67-82, 1987.

128. J. Fox, *Linear Statistical Models and Related Methods,* John Wiley & Sons, Inc., 1984. ISBN 0-471-09913-9

129. F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel, *Robust Statistics - the Approach based on Influence Functions,* John Wiley and Sons, Inc, 1986. ISBN 0-471-82921

130. P.J. Rousseeuw, "Least Median of Squares Regression", *Journal of the American Statistical Association*, vol. 79, no. 388, pp. 871-880, Dec. 1984.

131. F.J. Anscombe, "Rejection of Outliers", *Technometrics*, vol. 2, no. 2, pp. 123-147, May 1960.

132. D.M. Hawkins, *Identification of Outliers,* Chapman and Hall Ltd, 1980. ISBN 0 412 21900 X

133. T.P. Hettmansperger and S.J. Sheather, "Resistant and Robust Procedures", in *Perspectives on Contemporary Statistics*, ed. D.C. Hoaglin and D.S. Moore, pp. 145-170, Mathematical Association of America, 1992. ISBN 0-88385-075-3

134. J.W. Tukey, "Robust Techniques for the User", in *Robustness in Statistics*, ed. R.L. Launer and G.N. Wilkinson, pp. 103-106, Academic Press, Inc, 1979. ISBN 0-12-438150-2

135. F. Mosteller and J.W. Tukey, *Data Analysis and Regression, a second course in statistics,* Addison-Wesley Publishing Company, 1977. ISBN 0-201-04854-X

136. R.V. Hogg, "Statistical Robustness: One View of Its Use in Applications Today", *The American Statistician*, vol. 33, no. 3, pp. 108-115, 1979.

137. P.J. Huber, *Robust Statistical Procedures,* J.W. Arrowsmith Ltd., 1977. SIAM Monographs 27

138. G.E.P. Box, "Robustness in the Strategy of Scientific Model Building", in *Robustness in Statistics*, ed. R.L. Launer and G.N. Wilkinson, pp. 201-236, Academic Press, Inc, 1979. ISBN 0-12-438150-2

139. P. Martin and L. Roberts, "Efficiency in Minitab", *Teaching Statistics*, vol. 18, no. 1, pp. 26-27, 1996.

140. J.E. Freund and R.E. Walpole, *Mathematical statistics,* Prentice Hall, Inc., 1987. ISBN 0-130562075-9 01

141. D.S. Moore and G.P. McCabe, *Introduction to the Practice of Statistics,* W.H. Freeman and Company, 1993. ISBN 0-7167-2250-X

142. W. Mendenhall, R.L. Scheaffer, and D.D. Wackerly, *Mathematical Statistics with Applications,* Wadsworth, Inc., 1981. ISBN 0-534-98019-8

143. W. Gilchrist, *Statistical Modelling,* John Wiley & Sons, 1984. ISBN 0 471 90391 4

144. P.J. Huber, *Robust Statistics,* John Wiley and Sons, Inc, 1981. ISBN 0-471-41805-6

145. J.R. Rice and J.S. White, "Norms for Smoothing and Estimation", *SIAM Review*, vol. 6, no. 3, pp. 243-256, 1964.

146. H. Ekblom, "L_p-Methods for Robust Regression", *BIT*, no. 14, pp. 22-32, 1974.

147. A.B. Forsythe, "Robust Estimation of Straight Line Regression Coefficients by Minimising p^{th} Power Deviations", *Technometrics*, vol. 14, no. 1, pp. 159-166, 1972.

148. R.V. Hogg, "Adaptive Robust Procedures: A Partial Review and Some Suggestions for Future Applications and Theory", *Journal of the American Statistical Association*, vol. 69, no. 348, pp. 909-927, 1974.

149. J. Hogel, W. Schmid, and W. Gaus, "Robustness of the Standard Deviation and Other Measures of Dispersion", *Biometrical Journal*, vol. 36, no. 4, pp. 411-427, 1994.

150. B. Iglewicz, "Robust Scale Estimators and Confidence Intervals for Location", in *Understanding Robust and Exploratory Data Analysis*, ed. D.C. Hoaglin, F. Mosteller, and J.W. Tukey, pp. 404-431, John Wiley and Sons, Inc, 1983. ISBN 0-471-09777-2

151. Y. Liu, "Robust Parameter Estimation And Model Selection For Neural Network Regression", in *Advances in Neural Information Processing Systems 4*, ed. R.P. Lippmann, J.E. Moody, and S.J. Hanson, pp. 192-199, Morgan Kaufmann Publishers Inc, 1992. ISBN 1-55860-222-4

152. D.F. Andrews, "A Robust Method for Multiple Linear Regression", *Technometrics*, vol. 16, no. 4, pp. 523-531, 1974.

153. G. Li, "Robust Regression", in *Exploring Data Trends, Tables and Shapes.*, ed. D.C. Hoaglin, F. Mosteller, and J.W. Tukey, pp. 281-343, John Wiley and Sons, Inc., 1985. ISBN 0-471-09776-4

154. D.S. Chen and R.C. Jain, "A Robust Back Propagation Learning Algorithm for Function Approximation", *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 467-479, 1994.

155. D. Ruppert and R.J. Carroll, "Trimmed Least Squares Estimation in the Linear Model", *Journal of the American Statistical Association*, vol. 75, no. 372, pp. 828-838, 1980.

156. P. Slade and T.D. Gedeon, "Bimodal Distribution Removal", *Proceedings IWANN International Conference of Neural Networks*, pp. 249-254, 1993.

157. S.J. Hanson and D.J. Burr, "Minkowski-r Back-Propagation: learning in Connectionist Models with Non-Euclidian Error Signals", in *Neural Information Processing Systems*, ed. D.Z. Anderson, American Institute of Physics, 1988. ISBN 0-88318-569-5

158. J.A. Joines and M.W. White, "Improved Generalisation Using Robust Cost Functions", *International Joint Conference on Neural Networks (IJCNN)*, vol. 3, pp. 911-918, 1991.

159. D.J. Smith, T.C. bailey, and A.G. Munford, "Robust classification of high-dimensional data using artificial neural networks", *Statistics and Computing*, vol. 3, pp. 71-81, 1993.

160. C. Croux, P.J. Rousseeuw, and O. Hossjer, "Generalised S-Estimators", *Journal of the American Statistical Association*, vol. 89, no. 428, pp. 1271-1281, 1994.

161. P. Huber, "Minimax Aspects of Bounded-Influence Regression", *Journal of the American Statistical Association*, vol. 78, no. 381, pp. 66-80, 1983.

162. J.W. McKean, S.J. Sheather, and T.P. Hettmansperger, "Robust and High-Breakdown Fits of Polynomial Models", *Technometrics*, vol. 36, no. 4, pp. 409-415, 1994.

163. R.D. Cook, D.M. Hawkins, and S. Weisberg, "Comparison of Model Misspecification Diagnostics Using Residuals from Least Mean of Squares and Least Median of Squares Fits", *Journal of the American Statistical Association*, vol. 87, no. 418, pp. 419-424, 1992.

164. J.W. McKean, S.J. Sheather, and T.P. Hettmansperger, "The Use and Interpretation of Residuals Based on Robust Estimation", *Journal of the American Statistical Association*, vol. 88, no. 424, pp. 1254-1263, 1993.

165. A.S. Weigend and B. LeBaron, "Evaluating Neural Network Predictors by Bootstrapping", *Proceedings of International Conference on Neural Information Processing*, vol. 2, pp. 1207-1212, 1994.

166. R.T. St. Laurent and R.D. Cook, "Leverage and Superleverage in Nonlinear Regression", *Journal of the American Statistical Association*, vol. 87, no. 420, pp. 985-990, 1992.

167. W.H. Ross, "The geometry of case deletion and the assessment of influence in nonlinear regression", *The Canadian Journal of Statistics*, vol. 15, no. 2, pp. 91-103, 1987.

168. D.A. Belsley, E. Kuh, and R.E. Welsch, *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity,* John Wiley & Sons., 1980. ISBN 0-471-05856-4

169. D. Pregibon, "Logistic Regression Diagnostics", *The Annals of Statistics*, vol. 9, no. 4, pp. 705-724, 1981.

170. J.D. Emerson, D.C. Hoaglin, and P.J. Kempthorne, "Leverage in Least Squares Additive-Plus-Multiplicative Fits for Two-Way Tables", *Journal of the American Statistical Association*, vol. 79, no. 386, pp. 329-335, 1984.

171. R. Schall and T. Dunne, "A note on the relationship between parameter collinearity and local influence", *Biometrika*, vol. 79, no. 2, pp. 399-404, 1992.

172. R.T. St. Laurent and R.D. Cook, "Leverage, local influence and curvature in nonlinear regression", *Biometrika*, vol. 80, no. 1, pp. 99-106, 1993.

173. R. Schall and R. Gonin, "Diagnostics for nonlinear L_p-norm estimation", *Computational Statistics and Data Analysis*, vol. 11, no. 2, pp. 189-198, 1991.

174. D.C. Hamilton, D.G. Watts, and D.M. Bates, "Accounting for Intrinsic Nonlinearity in Nonlinear Regression Parameter Inference Regions", *The Annals of Statistics*, vol. 10, no. 2, pp. 386-393, 1982.

175. R.P. Lippmann and L. Kukolich, "Predicting the Risk of Complications in Coronary Artery Bypass Operations usign Neural Networks", in *Advances in Neural Information Processing Systems 7*, ed. G. Tesauro, D. Touretzky, and T. Leen, pp. 1055-1062, The MIT Press, 1995. ISBN 0-262-20104-6

176. M. Stone, "Asymptotics for and against cross-validation", *Biometrika*, vol. 64, no. 1, pp. 29-35, 1977.

177. B. Efron and G. Gong, "A Leisurely look at the Bootstrap, the Jackknife, and Cross-Validation", *The American Statistician*, vol. 37, no. 1, pp. 36-48, 1983.

178. J.S. Urban Hjorth, *Computer Intensive Statistical Methods - validation model selection and bootstrap,* Chapman & Hall, 1994. ISBN 0 412 49160 5

179. M. Plutowski, S. Sakata, and H. White, "Cross-Validation Estimates IMSE", in *Advances in Neural Information Processing Systems, 6*, ed. J.D. Cowan, G. Tesauro, and J. Alspector, pp. 391-398, Morgan Kaufmann, Inc., 1994. ISBN 1-55860-322-0

180. Y.S. Chow, S. Geman, and L.D. Wu, "Consistent Cross-validated Density Estimation", *The Annals of Statistics*, vol. 11, no. 1, pp. 25-38, 1983.

181. B. Efron, "How Biased is the Apparent Error Rate of a Prediction Rule", *Journal of the American Statistical Association*, vol. 81, no. 394, pp. 461-470, 1986.

182. P. Burman, "A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods", *Biometrika*, vol. 76, no. 3, pp. 503-514, 1989.

183. P. Craven and G. Wahba, "Smoothing Noisy Data Using Spline Functions", *Numerische Mathematik*, vol. 31, pp. 377-403, 1979.

184. F. O'Sullivan and G. Wahba, "A Cross Validated Bayesian Retrieval Algorithm for Nonlinear Remote Sensing Experiments", *Journal of Computational Physics*, vol. 59, pp. 441-455, 1985.

185. Y. Liu, "Unbiased Estimate of Generalisation Error and Model Selection in Neural Network", *Neural Networks*, vol. 8, no. 2, pp. 215-219, 1995.

186. C.L. Mallows, "Some Comments on C_p", *Technometrics*, vol. 15, no. 4, pp. 661-675, 1973.

187. D.A. Girard, "The Fast Monte-Carlo Cross-Validation and C_L Procedures: Comments, New Results and Application to Image Recovery Problems", *Computational Statistics*, vol. 10, pp. 205-231, 1995.

188. R.J. Carroll and D. Ruppert, "Diagnostics and Robust Estimation When Transforming the Regression Model and the Response", *Technometrics*, vol. 29, no. 3, pp. 287-299, 1987.

189. R.D. Cook and S. Weisberg, *Residuals and Influence in Regression,* Chapman and Hall, 1982. ISBN 0-412-24280-X

190. Y. Liu, "Neural Network Model Selection Using Asymptotic jackknife Estimator and Cross-Validation", in *Advances in Neural Information Processing Systems, 5*, ed. S.J. Hanson, J.D. Cowan, and C.L. Giles, pp. 599-606, Morgan Kaufmann, Inc., 1993. ISBN 1-55860-274-7

191. T. Fox, D. Hinkley, and K. Larntz, "Jackknifing in Nonlinear Regression", *Technometrics*, vol. 22, no. 1, pp. 29-33, 1980.

192. J.S. Simonoff and C.L Tsai, "Jackknife-Based Estimators and Confidence Regions in Nonlinear Regression", *Technometrics*, vol. 28, no. 2, pp. 103-112, 1986.

193. G.T. Duncan, "An Empirical Study of Jackknife-Constructed Confidence Regions in Nonlinear Regression", *Technometrics*, vol. 20, no. 2, pp. 123-129, 1978.

194. E. Walker and J.B. Birch, "Influence Measures in Ridge Regression", *Technometrics*, vol. 30, no. 2, pp. 221-227, 1988.

195. D.A. Belsley, *Conditioning diagnostics: Collinearity and Weak Data in Regression,* John Wiley & Sons, Inc., 1991. ISBN 0-471-52889-7

196. G.H. Golub and C.F. Van Loan, *Matrix Computations,* North Oxford Academic, 1983. ISBN 0-046536-00-7

197. G.W. Stewart, "Collinearity and Least Squares Regression", *Statistical Science*, vol. 2, no. 1, pp. 68-100, 1987.

198. S. Saarinen, R. Bramley, and G. Cybenko, "Ill-conditioning in Neural Network Training Problems", *SIAM Journal of Scientific Computing*, vol. 14, no. 3, pp. 693-714, 1993.

199. R. Gonin, "Numerical Algorithms for Solving Nonlinear L_p-Norm Estimation Problems: Part I - A First Order Gradient Algorithm For Well-Conditioned Small Residual Problems", *Communications in Statistics (Simulation)*, vol. 15, no. 3, pp. 801-813, 1986.

200. R. Gonin and S.H.C. du Toit, "Numerical Algorithms for Solving Nonlinear L_p-Norm Estimation Problems: Part II - A Mixture Method for Large Residual and Ill-Conditioned Problems", *Communications in Statistics (Theory & Methods)*, vol. 16, no. 4, pp. 969-986, 1987.

201. P. Sprent, in *Applied Nonparametric Statistical Methods (2nd Edition)*, pp. 277-293, Chapman and Hall, 1993. ISBN 0 412 44980 3

202. R.J. Freund and P.D. Minton, *Regression Methods,* Marcel Dekker Inc, 1979. ISBN 0-8247-6647-4

203. J.W. Cotts, "Checking Model Validity in Linear Regression", *Teaching Statistics*, vol. 9, no. 3, pp. 82-89, 1987.

204. K.N. Berk and D.E. Booth, "Seeing a Curve in Multiple Regression", *Technometrics*, vol. 37, no. 4, pp. 385-398, 1995.

205. S.R. Searle, "Parallel Lines in Residual Plots", *The American Statistician*, vol. 42, no. 3, p. 211, 1988.

206. J.A. Nelder, "Nearly Parallel Lines in Residual Plots", *The American Statistician*, vol. 44, no. 3, pp. 221-222, 1990.

207. R.J. Brooke and G.C. Arnold, *Applied Regression Analysis and Experimental Design,* Marcel Dekker, Inc., 1985. ISBN 0-8247-7252-0

208. A.S. Weigend, D.E. Rumelhart, and B.A. Huberman, "Bačk-Propagation, Weight Elimination and Time Series prediction", in *Proceedings of the 1990 Connectionist Summer School*, ed. D.S. Touretsky, J.L. Elman, T.J. Senjowski, and G.E. Hinton, pp. 105-116.

209. Y. LeCun, J.S. Denker, and S.A. Solla, "Optimal Brain Damage", in *Advances in Neural Information Processing Systems 2*, ed. D.S. Touretzky, pp. 598-605, Morgan Kaufmann Publishers Inc, 1992. ISBN 1-55860-100-7

210. M. Cottrell, B. Girard, Y. Girard, M. Mangeas, and C. Muller, "Neural Modelling for Time Series: A Statistical Stepwise Method for Weight Elimination", *IEEE transactions on Neural Networks*, vol. 6, no. 6, pp. 1355-1364, 1995.

211. B.W. Lindgren, *Statistical Theory,* Chapman & Hall, 1993. ISBN 0-412-04181-2

212. S.D. Silvey, *Statistical Inference,* Chapman & Hall Ltd., 1975. ISBN 0 412 13820 4

213. D.R. Cox and D.V. Hinkley, *Theoretical Statistics,* Chapman and Hall, 1974. ISBN 0 412 12420 3

214. E.L. Lehmann, *Theory of Point Estimation,* John Wiley & Sons, Inc., 1983. ISBN 0-471-05849-1

215. M.G. Kendall and A. Stuart, *The Advanced Theory of Statistics,* Volume 2, Charles Griffin & Company Limited, 1961.

216. H. White, "Consequences and Detection of Misspecified Nonlinear Regression Models", *Journal of the American Statistical Association*, vol. 76, no. 374, pp. 419-433, 1981.

217. A.J. Stromberg and D. Ruppert, "Breakdown in Nonlinear Regression", *Journal of the American Statistical Association*, vol. 87, no. 420, pp. 991-997, 1992.

Appendix F

List of Publications

References

1. A.J. Myles, A.F. Murray, and A.R. Wallace, "Robust Training of Multilayer Perceptrons: Some Experimental Results", in *1st International Conference on Applications and Science of Artificial Neural Networks*, ed. S.K. Rogers and D.W. Ruck, pp. 974-984, SPIE Optical Society of America, Orlando, USA, 1995. ISBN 0-81-941845-5

2. A.J. Myles, A.F. Murray, A.R. Wallace, J. Barnard, and G. Smith, "Estimating MLP Generalisation Ability without a Test Set using Fast, Approximate, Leave-One-Out Cross-Validation", *Neural Computing and Applications*, vol. 5, no. 3, pp. 134-151, 1997.