12-2003

Prediction Interval Estimation Techniques for Empirical Modeling Strategies and their Applications to Signal Validation Tasks

Brandon Peter Rasmussen
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a dissertation written by Brandon Peter Rasmussen entitled "Prediction Interval Estimation Techniques for Empirical Modeling Strategies and their Applications to Signal Validation Tasks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

J. Wesley Hines, Major Professor

We have read this dissertation and recommend its acceptance:

Robert E. Uhrig, Belle R. Upadhyaya, Hamparsum Bozdogan, Andrei Gribok

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Brandon Peter Rasmussen entitled "Prediction Interval Estimation Techniques for Empirical Modeling Strategies and their Applications to Signal Validation Tasks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

J. Wesley Hines
Major Professor

We have read this dissertation
and recommend its acceptance:

Robert E. Uhrig

Belle R. Upadhyaya

Hamparsum Bozdogan

Andrei Gribok

Accepted for the Council:

Anne Mayhew
Vice Provost and Dean of
Graduate Studies

(Original signatures are on file with official student records.)

Prediction Interval Estimation Techniques for Empirical Modeling Strategies and their Applications to Signal Validation Tasks

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Brandon P. Rasmussen

December 2003

DEDICATION

for Shasten

ACKNOWLEDGEMENTS

I would like to thank the members of my dissertation committee. Dr. J. Wesley Hines, my major professor, who guided me through my years of graduate study. Dr. Hines is an excellent and dedicated professional and it has been a privilege to work with him these past 6 years. Dr. Robert E. Uhrig, who has taken the time to review my work and provide guidance based on his considerable experience in both artificial intelligence and the nuclear power industry. Dr. Belle R. Upadhyaya, who has taken the time to review this work and has always been ready and willing to help out along the way. Dr. Hamparsum Bozdogan, who has brought his expertise in statistics to the table in reviewing and guiding this work. Dr. Andrei Gribok, who has provided the basis for portions of this dissertation through his work on regularization and nonparametric regression analysis.

While three members of the Nuclear Engineering department's faculty were on my dissertation committee, Dr. J. Wesley Hines, Dr. Belle R. Upadhyaya, and Dr. Robert E. Uhrig, I would also like to thank the rest of the faculty members. Dr. Lawrence W. Townsend, Dr. Ronald E. Pevey, Dr. Peter G. Groer, Dr. Arthur E. Ruggles, and Dr. Laurence F. Miller, all of whom are among the finest professors I have had the pleasure of studying under.

I would like to recognize the efforts of our department head Dr. H. Lee Dodds. His assistance and guidance over the course of my graduate studies has been a valuable asset, and his dedication to his work is commendable.

I would especially like to thank the organizations and companies that provided the funding for my studies and the completion of this work, the Department of Energy under a Nuclear Engineering Fellowship, Smart Signal, Lisle, Illinois, and the Institutt for Energiteknikk, Halden, Norway.

Finally, I would like to thank the staff of the Nuclear Engineering department who always provided any assistance that I requested. Our computer gurus Gary Graves and

Dick Bailey always knew what to do and how to fix it. Kristin England, Ellen Fisher, Lydia Salmon, and Juvy Melton never let me get by without the necessary paperwork, and were always ready to provide their assistance.

ABSTRACT

The basis of this work was to evaluate both parametric and non-parametric empirical modeling strategies applied to signal validation or on-line monitoring tasks. On-line monitoring methods assess signal channel performance to aid in making instrument calibration decisions, enabling the use of condition-based calibration schedules. The three non-linear empirical modeling strategies studied were: artificial neural networks (ANN), neural network partial least squares (NNPLS), and local polynomial regression (LPR). These three types are the most common nonlinear models for applications to signal validation tasks. Of the class of local polynomials (for LPR), two were studied in this work: zero-order (kernel regression), and first-order (local linear regression).

The evaluation of the empirical modeling strategies includes the presentation and derivation of prediction intervals for each of three different model types studied so that estimations could be made with an associated prediction interval. An estimate and its corresponding prediction interval contain the measurements with a specified certainty, usually 95%. The prediction interval estimates were compared to results obtained from bootstrapping via Monte Carlo resampling, to validate their expected accuracy.

The estimation of prediction intervals applied to on-line monitoring systems is essential if widespread use of these empirical based systems is to be attained. In response to the topical report "On-Line Monitoring of Instrument Channel Performance," published by the Electric Power Research Institute [Davis 1998], the NRC issued a safety evaluation report that identified the need to evaluate the associated uncertainty of empirical model estimations from all contributing sources. This need forms the basis for the research completed and reported in this dissertation.

The focus of this work, and basis of its original contributions, were to provide an accurate prediction interval estimation method for each of the mentioned empirical modeling techniques, and to verify the results via bootstrap simulation studies. Properly determined prediction interval estimates were obtained that consistently captured the

uncertainty of the given model such that the level of certainty of the intervals closely matched the observed level of coverage of the prediction intervals over the measured values. In most cases the expected level of coverage of the measured values within the prediction intervals was 95%, such that the probability that an estimate and its associated prediction interval contain the corresponding measured observation was 95%. The results also indicate that instrument channel drifts are identifiable through the use of the developed prediction intervals by observing the drop in the level of coverage of the prediction intervals to relatively low values, e.g. 30%.

While all empirical models exhibit optimal performance for a given set of specifications, the identification of this optimal set may be difficult to attain. The developed methods of prediction interval estimation were shown to perform as expected over a wide range of model specifications, including misspecification. Model misspecification occurs through different mechanisms dependent on the type of empirical model. The main mechanisms under which model misspecification occur for each empirical model studied are: ANN – through architecture selection, NNPLS – through latent variable selection, LPR – through bandwidth selection. In addition, all of the above empirical models are susceptible to misspecification due to inadequate data and the presence of erroneous predictor variables in the set of predictors. A study was completed to verify that the presence of erroneous variables, i.e. unrelated to the desired response or random noise components, resulted in increases in the prediction interval magnitudes while maintaining the appropriate level of coverage for the response measurements.

In addition to considering the resultant prediction intervals and coverage values, a comparative evaluation of the different empirical models was performed. The evaluation considers the average estimation errors and the stability of the models under repeated Monte Carlo resampling. The results indicate the large uncertainty of ANN models applied to collinear data, and the utility of the NNPLS model for the same purpose. While the results from the LPR models remained consistent for data with or without collinearity, assuming proper regularization was applied.

The quantification of the uncertainty of an empirical model's estimations is a necessary task for promoting the use of on-line monitoring systems in the nuclear power industry. All of the methods studied herein were applied to a simulated data set for an initial evaluation of the methods, and data from two different U.S. nuclear power plants for the purposes of signal validation for on-line monitoring tasks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AANN -	Autoassociative Artificial Neural Network
AC -	Analytic Coverage
AMISE -	Asymptotic Mean Integrated Squared Error
ANN -	Artificial Neural Network
API -	Analytic Prediction Interval
BC -	Bootstrap Coverage
BPI -	Bootstrap Prediction Interval
EPRI -	Electric Power Research Institute
GRNN -	Generalized Regression Neural Network
IANN -	Inferential Artificial Neural Network
ISE -	Integrated Squared Error
KR -	Kernel Regression
LL -	Local Linear (Regression)
LM -	Levenberg - Marquardt
LPR -	Local Polynomial Regression
MAE -	Mean Absolute Error
MGRNN -	Modified Generalized Regression Neural Network
MISE -	Mean Integrated Squared Error
MLP -	Multi-Layer Perceptron
MSE -	Mean Squared Error
MSET -	Multivariate State Estimation Technique
NaN -	Not a Number
NIPALS -	Nonlinear Iterative PArtial Least Squares
NN -	Neural Network
NNPLS -	Neural Network Partial Least Squares
NRBF -	Normalized Radial Basis Function
NRC -	Nuclear Regulatory Commission
NTNN -	N-Tuple Neural Network
OLS -	Ordinary Least Squares
PC -	Principal Component
PCA -	Principal Component Analysis
PCR -	Principal Component Regression
PI -	Prediction Interval
PIM -	Prediction Interval Magnitude
PLS -	Partial Least Squares
PNN -	Probabilistic Neural Network
RBF -	Radial Basis Function
RBFN -	Radial Basis Function Network
RR -	Ridge Regression
SISO -	Single Input Single Output
SSE -	Sum Squared Error
TSVD -	Truncated Singular Value Decomposition

1.0 INTRODUCTION

The motivation behind empirical model based signal validation is to provide a means of indicating the calibration status of the corresponding measurement instruments. This can be carried out either continuously, or on a time-delayed batch mode schedule. Continuous validation on-line will provide the most expedient status identification. As stated in EPRI's *On-line Monitoring of Instrument Channel Performance* [Davis 1998]: On-line monitoring is the assessment of channel performance and calibration while the channel is operating. The term channel refers to a sensor, the isolator, and intervening components. Isolators are a design requirement for safety related instrument loops of a nuclear power plant, thus considering these parameters, on-line monitoring starts at the output of an isolator. The sensor typically exhibits the greatest variability, and thus it is a common practice to associate observed drift with the sensor, though the true source can only be confirmed through additional investigation.

On-line monitoring differs from channel calibration in that the channel is not adjusted by the process of on-line monitoring. Instead, on-line monitoring compares channel measurements to empirical model estimations to determine if a channel calibration is necessary. Thus, on-line monitoring presents an alternate approach to time-directed periodic calibration, required by a nuclear power plant's technical specifications, which will provide calibration maintenance capabilities that are equivalent to the traditional periodic calibration schedules. On-line monitoring allows for the use of condition-based calibration schedules, since the calibration status of the monitored channels is continuously verified to be within tolerable limits. Condition based calibration schedules present several advantages, including: reduced labor costs, reduced personnel radiation exposure, and increased instrument availability. Condition based calibration activities may also increase the usable life of an instrument through reducing the risks of instrument damage incurred during adjustments.

Empirical signal validation models are often constructed using sets of highly correlated predictor variables. Variable matrices characterized by strong linear dependencies

between columns are said to be collinear and present an ill-posed problem to empirical modeling tasks [Næs 2001, Wold 1984, Hadamard 1923]. This causes an increase in estimation noise levels and causes unstable and unrepeatable results. The problem arises due to the numerical complexities of inverting collinear matrices, which can also be described as ill-conditioned matrices. The data sets are often highly correlated due to the number of redundant and related sensors used to monitor large-scale processes, such as a nuclear power plant or other types of electrical generation facilities. These sensors are located throughout the system to monitor parameters that are physically related and thus, their measurements are also related. The use of redundant sensors to monitor safety critical parameters exacerbates the problem by increasing the average correlations of the data set.

The modeling strategies applied to signal validation tasks in this work were: artificial neural networks (ANN), neural network partial least squares (NNPLS), and local polynomial regression (LPR). These 3 modeling paradigms have been the most commonly reported for applications to signal validation tasks in on-line monitoring applications. To provide a basis for parametric modeling, discussions and a theoretical basis are provided for ordinary least squares (OLS), principal component analysis (PCA), and partial least squares (PLS). Prediction interval estimation expressions are presented for these methods and later extended to the more complex non-linear models: ANN, NNPLS, and LPR.

The prediction interval estimation methods explored in this work were based on standard nonlinear regression theory, for the ANN and NNPLS models, and based on nonparametric regression theory for the LPR models. The focus of this work was to provide point-wise prediction intervals which contain the measured responses to a specified significance level, namely 95%. Signal validation models of the 3 types under study have been reported to be appropriate for use in on-line monitoring systems and commercially available software utilize these models or derivatives thereof. One of the functions of an on-line monitoring system is to report when an empirical model's

estimations are significantly deviating from the measured values of the monitored process. While the ability to detect these significant deviations has been proven, the quantification of the uncertainty associated with the empirical model estimates is rarely addressed. To verify that the observed deviations are significant, in that they exceed all observed effects of modeling uncertainty, prediction interval techniques need to be developed, proven, and incorporated into existing and future software for on-line monitoring applications.

1.1 ORIGINAL CONTRIBUTIONS

Signal validation using empirical models which are fundamentally different are consistently reported in engineering literature. In this work, quantitative comparisons were drawn based on a measure of prediction error, stability of the resultant solutions, the magnitude of the prediction intervals associated with each empirical model, and the consistency with which the computed prediction intervals contain the observed signal values, i.e. coverage.

The following original contributions are made in this dissertation:

- A comparative evaluation of the non-linear empirical modeling approaches: ANNs, NNPLS, and LPR, with respect to their estimation errors, applied to nuclear power plant data.
- A comparative evaluation of the uncertainty, through prediction interval computations, of the empirical models: ANNs, NNPLS, and LPR.
- The validity of the established prediction interval estimation methodologies was confirmed by comparing the results to bootstrap prediction intervals for each empirical model studied.
- The application of prediction intervals to the empirical model estimates within the framework of on-line monitoring systems.
- The development and evaluation of prediction interval estimation methods over a wide range of model specifications, including model misspecification.

1.2 ORGANIZATION OF DOCUMENT

A condensed summary of the most relevant literature reviewed in preparing this dissertation is presented in chapter 2. Chapter 3 provides the theoretical foundations of the linear and non-linear empirical modeling strategies. Issues related to the implementation of the different models are also discussed in chapter 3. The methods of prediction interval estimation are presented in chapter 4. Also included in chapter 4 are a section defining prediction intervals and a section discussing the sources of uncertainty that influence the prediction interval computations. Chapter 4 ends with a summary of the equations required for prediction interval estimation of each empirical model under study, and a summary of the assumptions inherent to the empirical modeling techniques as well as the assumptions made in the prediction interval computations. In chapter 5 the specifics related to the methods in which results were obtained are presented and concise definitions for the performance metrics used for comparisons are given. The structure and contents of the results are also described. The results from each of the three data sets are then presented and discussed in individual sections, followed by a summary section of all results. The conclusions of this dissertation are drawn and presented in chapter 6 along with recommendations for future work. The list of references, appendix, and vita follow the concluding chapter. The appendix contains the MATLAB [Mathworks 2003] functions written to construct the empirical models and compute the described prediction intervals.

2.0 LITERATURE REVIEW

This chapter presents, in condensed form, a summary of the most relevant prior research in the areas of prediction interval estimation for linear and non-linear empirical models. Other related research describing empirical modeling issues is also discussed. In the appropriate sections, the reviewed works are expanded and discussed in more detail.

OLS is one of the simplest and most common empirical modeling techniques. In addition, the derivation of the OLS solution is straight forward, and easily highlights the resultant complications in solving for the model parameters under conditions of collinear data. A collinear data matrix exhibits near or exact linear dependence between its columns. When constructing empirical models with data (**X)** that exhibit collinear behavior, the resulting variance-covariance matrix ($\mathbf{X}^T\mathbf{X}$) will be ill-conditioned, leading to unreliable model parameters [Baffi 2002, Næs 2001, Næs 1998, Sundberg 2000, Wold 1984, Woodward 1996]. A similar situation arises with non-linear regression methodologies, including artificial neural networks [Qin 1997]. In these cases, the optimization algorithms generally tend to become unstable or converge towards non-optimal solutions in the presence of variable correlations [Qin 1993]. The arguments regarding the complications related to empirical modeling with collinear data have also been extended to Kramer's [1992] autoassociative artificial neural network (AANN) architecture, which has been used extensively at the University of Tennessee [Hines 1997a, Hines 1997b, Upadhyaya 1992, Xu 1999, Xu 2000], and by other researchers [Fantoni 1996, 1998, 2002], for signal validation tasks.

Multivariate calibration is a general technique which models the relation between a dependent variable and a set of measured predictor variables. Common approaches to multivariate calibration include OLS, PCA, and PLS. Each of these techniques has had extensive coverage in the literature: OLS [Draper 1966], PCA [Jolliffe 2002], PLS [Martens 1989]. The model parameters are obtained in the form of a regression vector, which are used to estimate the dependent variable. It is desirable to not only provide a

point estimate for the dependent variable, but also an interval estimate for future predictions.

Faber and Kowalski [1997] presented expressions for prediction intervals, regression vector variances, and regression vector bias measures for OLS, principal component regression (PCR), and PLS. Previous reports have established these expressions, though the presentation by Faber and Kowalski [1997] was found to be the most thorough and concise. Though their work carries the evaluation through to include measurement errors in the independent variables, only consideration of the measurement errors in the dependent variables is given herein. Due to the large scope of this work, covering 3 fundamentally different empirical modeling paradigms, the decision was made to maintain the assumption of noise-free predictor variables in the basic derivations and to incorporate modifications that would account for the presence of noise in the predictors.

The statistical derivation of confidence intervals for parameters estimated by nonlinear least squares models is based upon the assumption that the model residuals are normal, independent, and identically distributed (i.i.d). Under these assumptions, the approach to calculating confidence intervals can be extended to neural networks since they can be regarded as a non-linear least squares modeling problem [Shao 1997]. Chryssolouris et. al. [1996] derived a technique to quantify the confidence intervals for the prediction of neural network models by adopting a variant of the linearization methodology. In addition to the assumption that the residuals are i.i.d., prediction interval estimation methods also assume that the neural networks are trained to convergence [de Veaux 1998], and that their architecture is predetermined. The confidence interval estimations methods are asymptotically valid when the number of training vectors becomes infinite [Hwang 1997]. The linearization method presented by Chryssolouris et. al. [1996] is based solely on the Jacobian matrix, which they deemed more favorable than other variants based on the use of the full Hessian matrix because the use of the Jacobian requires less computation, is more computationally stable, and at least as accurate.

A later analysis incorporated the influence of the distribution of the training data [Shao 1997], i.e. the distribution of the residuals. This approach incorporated the influence of the training data density via a wavelet-based density estimator. Using the wavelet-based density estimator provides a coefficient which is then applied to the standard confidence intervals as derived through the non-linear least squares modeling problem. The coefficient decreases for new observations which occur in regions where the training data is sparse, and the coefficient increases for new observations occurring in regions where the training data is more dense. This is directly related to the distribution of the model residuals, assuming the neural network has been trained to convergence, because the residual distribution will accordingly follow the relative density of the training data.

The work of Hwang and Ding [1997] provides a method analogous to that proposed by Chryssolouris et. al. [1996] regarding the construction of prediction intervals. In addition, they proved that the estimator is asymptotically valid when the number of training points goes to infinity, i.e. $P(y_0 \in \hat{y}_0 \pm c) \rightarrow (1-a)$ where c is the prediction interval magnitude and $1-a$ is the significance level of the interval. Their derivation provides a more rigorous body of proofs and lemmas to confirm the asymptotic properties of the prediction intervals. They also reported that the computation of prediction intervals via delete-one jack-knifing could be used to estimate the number of hidden neurons in a single hidden layer feedforward architecture. The delete-one-jack-knifing approach omits an observation from the training set and finds a neural network solution based on this reduced set of training observations. The prediction intervals are then computed over all training samples, including the omitted sample. This is repeated n times for each of the n observations in the training set. The coverage probability is then obtained by counting the number of times the computed interval contains the omitted response. The optimal number of neurons is such that the prediction interval estimation of the delete-one jack-knifing provides the maximum coverage (predicted points within the confidence interval).

DeVeaux et. al. [1998] showed that the proposed methods for interval estimation work well when the training set is large; however, when the training set is small and the network is trained to convergence the estimates become numerically unstable due to the near singularity of $\mathbf{F}^T\mathbf{F}$ matrix, where $\mathbf{F}$ is the Jacobian matrix. It was also noted that stopping the neural network training prior to convergence reduces the effective number of parameters and can lead to confidence intervals that are too wide. They proposed training via the weight decay method to prevent overfitting. The weight decay objective function contains an additional term other than the usual squared error. The additional term sums the squared weights of the neural network, thus the weight decay method simultaneously minimizes the residual error and the magnitude of the neural network weights. The weight decay approach to neural network training provides an available number of free parameters to achieve appropriate interval estimates and overfitting is avoided because unnecessary weights are forced to zero via the objective function of the training algorithm.

Yang et. al. have evaluated the confidence bound methodologies on a 10 dimensional function, as well as some simpler functions [2000]. In their analysis they focused on the percentage of targets which fell within the confidence bounds. They referred to this quantity as coverage. Though they found that the estimated confidence intervals were not always exact, they did report that with a large number of training points the confidence intervals normally reflect the distribution of the training data.

Wansink and Yang [2001] followed the work of Yang et. al. [2000] to provide similar results for a porosity estimation problem [De Groot 1996] which was evaluated in all of the above referenced works regarding confidence bound estimation for ANNs.

Leonard et. al. [1992] determined the prediction confidence intervals for radial basis function networks. Their validity index network (VI-net) also computes the density of the training data, and an indicator to show when the network is extrapolating from the training data. These computations are implemented as extra neural network units

associated with the original network. Note that the density estimation implemented by Leonard et. al. [1992] is performing a similar task to the wavelet-based density estimator of Shao et. al. [1997].

Additional applications of prediction interval estimation for neural networks have also been reported [Carney 1999, DaSilva 2000, Edwards 2002, Ho 2001, Papadopoulos 2001, Ye 2000, Tibshirani 1996, Dybowski 1997, and Derks 1998].

PLS has many attractive features making it an important model to consider for signal validation purposes. The creation of orthogonal bases from which univariate regression models can be derived, eliminates the numerical problems associated with collinearity. An additional feature is the inherent regularization of the method, which provides reproducible, stable solutions. PLS is a class of techniques for modeling the associations between blocks of observed variables by means of latent variables. These latent variables are created through a supervised transformation, whereby an orthogonal basis of latent variables results. The inner relationships, between latent variable pairs, of the standard PLS algorithm are constructed using univariate regressions on the latent vectors. In attempts to enhance the technique, quadratic functions have been introduced into the inner relationship [Baffi 1999a, Ni 1995, Wold 1989, Wold 1992]. However, quadratics are still linear in their parameters and do not guarantee a proper solution [Bro 1995]. The use of single hidden layer feedforward neural networks has been suggested [Gemperline 1991], and applied in the field of chemical engineering [Baffi 1999b, Holcomb 1992, Malthouse 1997, Qin 1992, Qin 1996]. Radial basis function networks have also been applied [Wilson 1997a, Wilson 1997b]. Additional work has provided a non-linear dynamic version incorporating polynomials, radial basis functions networks, or feedforward neural networks [Baffi 2000].

The method incorporating feedforward ANNs into PLS has been under study at the University of Tennessee, for the purposes of signal validation in large-scale processes, beginning in 1999 [Hines 2000c, Hines 2001, Rasmussen 2002, 2000a, 2000b]. The

method used herein uses fully connected feedforward neural networks in the inner relationships and maintains the linear orthogonalization in the outer relationships. This method will be referred to as neural network partial least squares (NNPLS), to avoid confusion with the various nonlinear methods utilizing quadratics, radial basis functions, and splines.

In a recent work, Baffi et. al. [2002] extended the work of Chryssolouris et. al. [1996] regarding the construction of prediction intervals for feedforward neural networks to the NIPALS-based (non-linear iterative partial least squares) linear and non-linear PLS. Prediction intervals are computed using a first order Taylor series expansion and the Jacobian matrix of the functional mapping provided by the PLS algorithm. The derivation of the Jacobian provided by Baffi et. al. [2002] was based on a single response variable (PLS-1); however, it was noted that it can be extended to the case of more than one response variable (PLS-2) by applying the same procedure to each output variable. This method of prediction interval estimation for NNPLS will be analyzed in detail, though other methods based on empirical estimations, analytical methods, bootstrapping, and jack-knife estimation have been reported [Berger 1997, De Vries 1995, Denham 1997, Faber 1996, 2000a, 2000b, Høy 1998, Martens 2000a, Martens 2000b, Morsing 1998, Phatak 1993, Wehrens 1997].

The final technique under evaluation is that of nonparametric regression, more specifically Local Polynomial Regression (LPR). A good review of the general class of lazy learning methods is presented by Atkeson et. al. [1996]. The thorough treatment of LPR by Fan and Gijbels [1996] is an indispensable resource. A lazy learning toolbox is available on-line with a corresponding manual. This toolbox has been developed and made available by Mauro Birattari and Gianluca Bontempi [1999], and is available at **http://www.iridia.ulb.ac.be/~lazy/**. The toolbox will provide local polynomial fits up to a polynomial degree of 2, i.e. local quadratic.

Nonparametric regression comprises a set of methods in which data processing is deferred until a prediction at a query point needs to be made. These methods are also referred to as memory based methods due to the approach of storing the training data, and recalling relevant training data when a query is made. Local, nonparametric models attempt to fit the data in a region surrounding the query point.

Nonparametric models are susceptible to estimation problems near the boundaries of the observation intervals. Consider a predictor variable x and a response variable y. At the boundary of the observation interval, the local averaging process becomes asymmetric because half of the weights are undefined and outside the boundary creating a bias related to the tangential behavior at the boundary. Boundary effects in nonparametric regression have been addressed by Härdle [1990], Wand and Jones [1995], and others. These boundary effects were reported to be of reduced influence for local linear models [Fan 1993]. An interesting perspective presented by Ruppert and Wand [1994], further considers this boundary effect, and explains that it is not simply due to the asymmetric data distribution near the boundary.

A general set of local regression estimators is the class of LPR estimators. LPR estimators obtain an estimate for a given query point by fitting a d^{th} degree polynomial using weighted least squares. The training points are weighted based on their distance from the query point using a kernel function centered at the query point. The kernel's influence can be adjusted via the kernel bandwidth. For various values of the degree of the polynomial (d) different terminology has evolved. The process of fitting constants ($d = 0$) using a locally weighted training criterion is known as Kernel Regression (KR) [Aït-Sahalia 2001, Cai 2001, Kauermann 1998, Liero 1992, Pawlak 1997, Wu 1994, Ziegler 2002]. An excellent text reference for KR is *Applied Nonparametric Regression* by W. Härdle [1990]. Wand and Jones [1995] provide another text resource with a thorough evaluation of the asymptotic properties of KR estimators. Local Linear (LL) regression is equivalent to LPR using $d = 1$ [Fan 1992a, Fan 1992b, Fan 1993], and local quadratic regression is equivalent to LPR using $d = 2$. Additional works covering LPR

techniques are available [Atkeson 1996, Cleveland 1979, Cleveland 1988, Fan 1995a, Fan 1995b, Ruppert 1994].

The memory based nature of nonparametric regression has strong similarities to the associative memories of neural networks. Kernel regression can be analogously presented as a generalized regression neural network (GRNN) [Specht 1991, Schiøler 1992]. The GRNN is a special extension of the radial basis function network (RBFN) [Park 1991, 1993], and is closely related to Specht's probabilistic neural network (PNN) [1990]. RBF networks were originally proposed by Broomhead and Lowe [1990]. The main difference between GRNNs and RBFs are that the GRNN output layer performs a weighted average while the RBF performs a weighted sum [Heimes 1998]. The basic GRNN has similarities with the methods of Moody and Darken [1989], RBFs [Park 1991, 1993], the cerebellar model articulation controller [Kolcz 1999], and nonparametric kernel based techniques stemming from the work of Nadaraya [1964] and Watson [1964]. Procedures for a neural network-type architecture with a memory of training vectors which computed estimates based on a neighborhood of points local to the query point were first presented by Steinbuch [1963] and Taylor [1959, 1960]. The original schemes were based on a winner takes all approach where the winning node was determined based on a distance metric between the query point and the stored training vectors.

Since kernel regression is based on established statistical principles and converges with an increasing number of samples asymptotically to the optimal regression surface, the same claim can be made for a GRNN [Tomandl 2001]. A further modification to the standard GRNN, modified GRNN (MGRNN), adds the abilities to deal with data vectors of unequal length thus allowing for the use of data structures such as data trees, and the ability to compute the gradient of the regression surface directly without the need for numerical approximations [Tomandl 2001]. The MGRNN utilizes a different kernel bandwidth for each training vector, similar to the adaptive GRNN presented by Specht and Romsdahl [1994] and later by Masters [1995]. Earlier GRNN versions utilized a single kernel bandwidth for the whole network [Specht 1991]. All of the GRNN models

are consistent in their use of spherical kernel functions. Training of GRNNs usually consists of optimizing the bandwidth parameters based on the MSE of the training vectors. Additional work using GRNNs has been reported by Kolcz and Allinson [1996] under the alternate architecture of an N-tuple regression network, or N-tuple neural network (NTNN). They site an increased capacity for training vector storage and faster computation.

Prediction interval estimation for nonparametric regression has been studied by Härdle et. al. [1988, 1990, 1991]. The basis of Härdle's work was the derivation of confidence intervals for nonparametric regression estimators via bootstrapping [Efron 1982]. Early work on confidence bound estimation with respect to kernel estimators was completed by Bickel and Rosenblatt [1973], and later extended by Johnston [1982]. The confidence intervals derived from this early work require a bandwidth selection that results in minimal bias. Due to the bias-variance trade off this leads to a relatively high variance estimator. As will be discussed in chapter 3, the bandwidth of a kernel regression model controls the complexity of the model. Lower bandwidth models exhibit higher complexity while higher bandwidth models exhibit lower complexity. The bias/variance trade-off is that for low-bandwidth high-complexity models the variance is at its maximum while the bias is at its minimum. As the bandwidth increases, the complexity of the model decreases. The variance decreases with complexity, while the bias increases due to the over-regularization (over-smoothing) of the model. This is the bias-variance trade-off. A later work by Eubank and Speckman [1993] resulted in a bias-corrected confidence band that allows the bandwidth selection to remain data driven to avoid the constraint of minimum bias causing high variance estimators. Other methods have been derived for confidence interval estimation [Knafl 1984, Knafl 1985, Nychka 1988, Nychka 1990, Wahba 1983], though all seem to not be applicable to a wide array of problems. The variance and bias properties of LPR models is well known, though direct use of these properties for prediction interval estimation is prohibited due to their dependence on unknown quantities such as the derivatives of the function being modeled. Fan and Gijbels [1996] present an approach based on asymptotic approximations to the

variance and bias of the LPR model, and other work has been based on a similar approach [Gribok 2003]. The method used in this dissertation considers both the variance and the bias in computing prediction interval estimates. The variance estimator utilized is well known and has been presented in various sources [Wand 1995, Fan 1996, Ruppert 1994]. This variance estimator has also been applied [Gribok 2003] to the Multivariate State Estimation Technique (MSET) [Gross 1998, Singer 1996]. MSET is an empirical model in the class of nonparametric estimation techniques. A method of bias estimation has been incorporated into the methods used in this dissertation so that the effects of the bias/variance trade-off appropriately influence the resultant prediction intervals.

The bootstrap is a method of Monte Carlo simulation whereby no assumptions are made about the population from which a random sample was obtained. The random sample is taken to be an estimate of the population, and each value within the random sample has an identical probability of occurrence. For each random sample, an estimate of a specific population parameter can be obtained. Repetitive sampling and consequent estimations provide a distribution for the parameter of interest. In the case of regression estimation, the parameter of interest is the estimate of the response, and its distribution can be used to construct prediction intervals around the estimate. Full coverage of the bootstrap technique is provided by several sources [Efron 1993, Hall 1992, Efron 1982]. The use of bootstrap techniques for constructing prediction intervals for nonlinear regression models has been documented by Derks and Buydens [1998] who assessed bootstrap resampling methods and the delta method for estimating prediction intervals for multi-layer feedforward neural networks. Similar efforts have been reported by Tibshirani [1996] and Dybowski [1997]. The utility of the bootstrap is best prescribed in the following statement by Dybowski [1997]: "The bootstrap is a computer-based method for assigning measures of accuracy to statistical estimates, and it will provide a nonparametric confidence interval for any population parameter whatsoever."

The approach to obtaining prediction intervals for neural networks via bootstrapping has been studied by various researchers, with the general consensus being that the bootstrap prediction intervals are more robust than their analytically derived counterparts. Bootstrapping methodologies have also been applied to neural network ensembles [Carney 1999]. A neural network ensemble is a set of neural networks each trained using a different bootstrap sample for the training set. Estimates from all of the trained networks are combined to provide generalization performance superior to that of a single neural network as well as reduced prediction intervals for the estimates.

The presented summary contains the most relevant information that was reviewed in preparing this dissertation. While some of the works were presented to provide references to related studies, many of the brief discussions presented in this chapter will be expanded in the appropriate sections of chapters 3 and 4.

3.0 DESCRIPTION OF MODELING STRATEGIES

In this chapter, prior to diving in to the advanced empirical models under study, a theoretical basis is presented for the more basic empirical regression models of OLS, PCR, and PLS. The discussion of these models is relevant to understanding not only the basis of the models under study, but also the treatment of prediction intervals for the basic models provides significant insight into the development of the prediction interval estimation methodologies for the advanced regression models that are the subject of this work.

The simplest type of regression model that is an indispensable member of the set of empirical models is OLS. A brief description of the mathematical mechanisms of OLS regression is initially provided. The extremely important consideration of collinearity is discussed thereafter. PCA and PCR are also covered briefly, followed by a slightly more rigorous development of PLS. In the same section, the NNPLS model is developed as an extension of the base PLS design. A general basis for ANNs is given, leaving the more detailed mathematics to the section in the next chapter regarding ANN prediction intervals to avoid repetitious presentations. Both NNPLS and ANNs can be represented within the framework of nonlinear regression, from which the analytical prediction interval estimation methodologies are derived. Thus, the overall nonlinear regression methods are described. The training algorithm used to train the neural networks for this work was the Levenberg-Marquardt algorithm, which is briefly discussed in its appropriate section. The last of the 3 empirical modeling approaches falls into the more general category of nonparametric regression; thus an overview of nonparametric regression is provided. This is followed by in depth discussions of LPR, and its derivative models of KR and LL. To close, two brief discussions are presented to highlight the similarities and differences between LPR and ANNs, and to present some of the discussion surrounding the differences between parametric and nonparametric approaches.

3.1 ORDINARY LEAST SQUARES

To begin a study of advanced empirical modeling techniques it is relevant to at first consider the more basic models, because the issues related to the simpler models are also issues for the advanced models. Here, keeping discussions to a minimum are presented the mathematical bases for OLS. In matrix notation the OLS equations can be presented as:

$$\mathbf{y} = \mathbf{Xb} + \mathbf{e}$$

$$\underset{(nx1)}{\mathbf{y}} = \underset{(nxp)}{\mathbf{X}} \bullet \underset{(px1)}{\mathbf{b}} + \underset{(nx1)}{\mathbf{e}}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ . \\ . \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & . & . & . & x_{1p} \\ 1 & x_{21} & x_{22} & . & . & . & x_{2p} \\ . & . & . & & & & . \\ . & . & . & & & & . \\ . & . & . & & & & . \\ . & . & . & & & & . \\ . & . & . & & & & . \\ . & . & . & & & & . \\ 1 & x_{n1} & x_{n2} & . & . & . & x_{np} \end{bmatrix} \bullet \begin{bmatrix} b_0 \\ b_1 \\ . \\ . \\ . \\ . \\ b_p \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ . \\ . \\ . \\ . \\ . \\ . \\ e_n \end{bmatrix}$$

y is the response variable vector

X is the matrix of predictor variables

b is the vector of regression coefficients

e is the vector of error terms

The assumptions of the linear regression model are:

- the model is linear
- the predictor variables (**X**) are noise free
- all required predictor variables are available

- the predictor variables are independent
- $e_1 = e_2 = \cdots = e_i = \cdots e_n$, i.e. constant variance errors (homoscedasticity)
- the error terms are normally distributed and independent

The least squares solution is found by minimizing the sum of squares of the residuals, with respect to the regression coefficient vector $\mathbf{b}$:

$$\min|e| = \min \sum_{i=1}^{n} e_i^2$$

The e_i's are assumed independent, normally distributed random variables with a mean of zero, and a constant variance.

$$|e| = (\mathbf{y} - \mathbf{Xb})^{\mathbf{T}} (\mathbf{y} - \mathbf{Xb})$$

$$\frac{\partial |e|}{\partial \mathbf{b}} = -2\mathbf{X}^{\mathbf{T}}\mathbf{y} + 2\mathbf{X}^{\mathbf{T}}\mathbf{Xb}$$

Setting equal to zero and solving for $\mathbf{b}$ results in the normal equations:

$$\mathbf{X}^{\mathbf{T}}\mathbf{Xb} = \mathbf{X}^{\mathbf{T}}\mathbf{Y}$$

The solution requiring the solution of p normal equations with p unknowns is thus:

$$\mathbf{b} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{Y}$$

Confirmation that the solution is a minimum is obtained by computing the second derivative, and recognizing that the second derivative is a positive semi-definite matrix.

$$\frac{\partial^2 |e|}{\partial \mathbf{b}^2} = 2\mathbf{X}^{\mathbf{T}}\mathbf{X}$$

Prediction for future independent observations is then obtained through:

$$\hat{y}_i = \mathbf{x}_i (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{Y}$$

$\mathbf{x}_i$ is a $1 \times p$ observation vector, and $\hat{y}_i$ is the corresponding scalar response estimate.

3.2 COLLINEARITY AND ITS EFFECTS ON EMPIRICAL MODELS

A matrix $\mathbf{X}$ is said to be collinear if the columns in $\mathbf{X}$ are approximately or exactly linearly dependent. Collinearity means that the matrix $\mathbf{X}$ will have some dominating

types of variability that carry most of the available information [Martens 1989]. A square matrix is said to be singular if there is at least one linear dependency among the rows (or columns) of the matrix. For the case of a singular matrix the determinant is zero: $|\mathbf{X}| = 0$. The rank of a matrix is defined as the maximum number of linearly independent columns (or rows). Based on this, matrices containing linearly dependent columns are said to be rank-deficient, or ill-conditioned. In relation to empirical modeling, the matrix of concern is $\mathbf{X}^T\mathbf{X}$. Any linear dependence that exists in a matrix $\mathbf{X}$ will be preserved in the new matrix $\mathbf{X}^T\mathbf{X}$. When the determinant of a matrix is zero its inverse does not exist, and when it is near zero, its inverse contains values of extremely large magnitude. Thus, situations where the determinant of $\mathbf{X}^T\mathbf{X}$ is near zero lead to unstable estimates of the regression coefficients, OLS, which may be unreasonably large or have the wrong sign [Massart 1988]. This can be easily seen by viewing the matrix equation for the variance of the regression coefficients: $\text{var}(\mathbf{b}) = s^2(\mathbf{X}^T\mathbf{X})^{-1}$, where $s^2 = \text{var}(e)$.

Collinearity in a data set leads to an ill-posed problem that causes inconsistent results when data based models such as OLS or neural networks are used [Gil 1998]. Hadamard [1923] defined a well-posed problem as a problem which satisfies the following 3 conditions:

1. The solution for the problem exists.
2. The solution is unique.
3. The solution is stable or smooth under small perturbations in the data, i.e. small perturbations in the data produce small perturbations in the solution.

If any of these conditions are not met the problem is referred to as an ill-posed problem and special considerations must be taken to ensure a reliable solution.

Data sets from large-scale processes, such as an electrical generating facility, are often plagued with collinearity. Thus, OLS is not a suitable method of empirical modeling for these situations, due to the required matrix inversion to obtain the solution. The use of autoassociative artificial neural networks (AANN), used extensively for signal validation [Fantoni 1998, Hines 1997b, Upadhyaya 1992, Xu 1999], is also inhibited by collinearity

for similar reasons [Qin 1997]. Some type of regularization method should be utilized when dealing with collinear data and ill-posed problems. A simple description of regularization is the introduction of prior information into a problem for the purposes reducing the set of possible solutions, stabilizing the result. Regularization can be integrated into neural network development through the use of cross validation training and robust training techniques however; these methods are time consuming and require significant oversight and knowledge [Xu 2000]. Other methods of regularization include truncated singular value decomposition (TSVD), and ridge regression (RR) [Hoerl 1970, Hines 2000b]. The utility of the PLS algorithm in eliminating the problems associated with collinearity is an indispensable feature of the method. Though this feature is also available in PCR, the orthogonal projections to latent structures in PCR is unsupervised with regard to the response variables, focusing only on the variability contained in the predictor variable set. PLS on the other hand is supervised in that its orthogonal projections are performed to capture the maximum covariance between the predictor variables and the desired response, in its latent variables. The first latent variable contains the maximum covariance in a given direction and subsequent latent variables contain the maximum covariance remaining, in a direction orthogonal to all of the previous latent variables.

3.3 PRINCIPAL COMPONENT ANALYSIS AND PRINCIPAL COMPONENT REGRESSION

Karl Pearson introduced component analysis in 1901 [Pearson 1901] by defining the closest plane to a system of points in space such that the sum of squares of the perpendicular residuals onto that plane was the least. Hotelling [1933] solved the problem of finding the linear combination of variables that had maximum variance, which is the same as finding the direction cosines of the major axis of the concentration ellipsoids of a multinormal distribution. This problem can be extended to finding the hyper-plane containing greatest variance, which is clearly the same as the hyper-plane with the least perpendicular sum of squares. The scores (dimensions the same as a row or observation) give the coordinates of the projection of the i^{th} observation onto the fitted

plane. The linear coefficients that project the data onto the fitted plane are the direction cosines defining the principal axes of the concentration ellipsoids and equivalently define the space contained by Pearson's plane of best fit. These coefficients are commonly called loadings.

A comprehensive treatment of principal component analysis (PCA) was compiled by I.T. Jolliffe [2002]. The following description has been extracted from Hodouin, et. al. [1993]. PCA is a method for explaining the variance of the input variable matrix in terms of a number of new latent variables called principal components (PCs). In the p-dimensional input variable space, the first loading vector $\mathbf{a}_1$ defines the direction of greatest variability, and the score vector $\mathbf{z}_1$ represents the projection of each observation vector (input variable pattern or sample) onto $\mathbf{a}_1$. Alternatively, $\mathbf{a}_1$ and $\mathbf{z}_1$ are the first eigenvectors of $\mathbf{X}^\mathbf{T}\mathbf{X}$ and $\mathbf{X}\mathbf{X}^\mathbf{T}$, respectively. The second PC is that linear combination of the input variables explaining the next greatest amount of variability subject to the condition that it is orthogonal to the first PC, that is $\mathbf{z}_2 = \mathbf{E}_1\mathbf{a}_2$, where $\mathbf{E}_1 = \mathbf{X} - \mathbf{z}_1\mathbf{a}_1^\mathbf{T}$ is the residual matrix left after removing the predictions of the first PC.

This process can be repeated until p PCs are obtained. Considering all of the PCs, the dimensionality of the input variables has not been reduced, but the axes of the input variable matrix have been rotated to a new orthogonal basis. With large data sets of correlated variables, after defining A PCs, where $A << p$, most of the variation in the input variable matrix ($\mathbf{X}$) can be explained by the first A PCs. This leads to a dimensionality reduction from p to A.

The input variable matrix can be reconstructed via:

$$\mathbf{X} = \mathbf{z}_1\mathbf{a}_1^\mathbf{T} + \mathbf{z}_2\mathbf{a}_2^\mathbf{T} + ... + \mathbf{z}_A\mathbf{a}_A^\mathbf{T} + \mathbf{E}_A$$

where: $$\mathbf{E}_A = \mathbf{X} - \sum_{i=1}^{A} \mathbf{z}_i\mathbf{a}_i^T$$

Ideally, $\mathbf{E}_A$ should represent random error or noise.

The scores can be considered linear combinations of the original variables and the loadings are the weights of the original variables used to form these new linear combinations.

3.3.1 Solution by LaGrange Multipliers

Computation of PCs reduces to the solution of an eigenvalue-eigenvector problem for a positive-semidefinite symmetric matrix. Below is an iterative procedure, which is not the most efficient method of computing PCs; however, it provides insight into the details of the method [Jolliffe 2002]. The method begins by looking for a linear function $\mathbf{a}_1^{\mathrm{T}}\mathbf{X}$ which has maximum variance, where:

$$\mathbf{a}_1^{\mathrm{T}}\mathbf{X} = a_{11}\mathbf{x}_1 + a_{21}\mathbf{x}_2 + \cdots + a_{p1}\mathbf{x}_p = \sum_{j=1}^{p} a_{j1}\mathbf{x}_j$$

$\mathbf{X}$ is an nxp matrix and $\mathbf{a}_1$ is a $px1$ vector.

The next step in PCA is to look for a linear function $\mathbf{a}_2^{\mathrm{T}}\mathbf{X}$ with maximum variance and subject to the constraint that it is uncorrelated to $\mathbf{a}_1^{\mathrm{T}}\mathbf{X}$. This process continues until a maximum of p linear functions are obtained which are uncorrelated with one another. The column vector $\mathbf{a}_k$ is the k^{th} largest eigenvector of the covariance matrix of $\mathbf{X}$, and the corresponding k^{th} largest eigenvalue is l_k. Additionally, the PC scores $\mathbf{z}_k$ can be defined as: $\mathbf{z}_k^{\mathrm{T}} = \mathbf{a}_k^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}$, where $\mathbf{z}_k$ is an $nx1$ vector of PC scores. If the eigenvectors, $\mathbf{a}_k$, are scaled to unit length, then $\mathrm{var}(\mathbf{z}_k) = l_k$.

The first PC maximizes the variance of the first PC score, i.e. $\mathrm{var}(\mathbf{z}_1) = \mathbf{a}_1^{\mathrm{T}}\mathrm{S}\mathbf{a}_1$, where S is the covariance matrix of $\mathbf{X}$. This maximum will not be achieved for finite $\mathbf{a}_1$, so a normalization constraint is applied, $\mathbf{a}_1^{\mathrm{T}}\mathbf{a}_1 = 1$. Using the technique of LaGrange multipliers, maximize $\mathbf{a}_1^{\mathrm{T}}\mathrm{S}\mathbf{a}_1 - l(\mathbf{a}_1^{\mathrm{T}}\mathbf{a}_1 - 1)$ by taking the derivative with respect to $\mathbf{a}_1$,

and set equal to zero. This results in the following: $\mathrm{S}\mathbf{a}_1 - l\,\mathbf{a}_1 = (\mathrm{S} - l\mathbf{I}_p)\mathbf{a}_1 = 0$, where $\mathbf{I}_p$ is the identity matrix of size pxp. Thus, l is an eigenvalue of S, and $\mathbf{a}_1$ the corresponding eigenvector. To determine which of the eigenvectors results in the desired maximization, consider the quantity to be maximized: $\mathbf{a}_1^{\mathbf{T}}\mathrm{S}\mathbf{a}_1 = \mathbf{a}_1^{\mathbf{T}} l\,\mathbf{a}_1 = l\,\mathbf{a}_1^{\mathbf{T}}\mathbf{a}_1 = l$, so l must be as large as possible; thus, $\mathbf{a}_1$ is the eigenvector corresponding to the largest eigenvalue l_1.

The second PC maximizes $\mathbf{a}_2^{\mathbf{T}}\mathrm{S}\mathbf{a}_2$, under the additional constraint that $\mathbf{a}_1$ is uncorrelated with $\mathbf{a_2}$, i.e. $\mathbf{a}_2^{\mathbf{T}}\mathbf{a}_1 = 0$. Again, using the method of LaGrange multipliers, where l and f are LaGrange multipliers, and assigning the normalization constraint $\mathbf{a}_2^{\mathbf{T}}\mathbf{a}_2 = 1$. The function to maximize is then:

$\mathbf{a}_2^{\mathbf{T}}\mathrm{S}\mathbf{a}_2 - l\,(\mathbf{a}_2^{\mathbf{T}}\mathbf{a}_2 - 1) - f\,(\mathbf{a}_2^{\mathbf{T}}\mathbf{a}_1)$

Differentiation with respect to $\mathbf{a_2}$ yields:

$\mathrm{S}\mathbf{a}_2 - l\,\mathbf{a}_2 - f\,\mathbf{a}_1 = 0$

Multiplication by $\mathbf{a}_1^{\mathbf{T}}$ yields:

$\mathbf{a}_1^{\mathbf{T}}\mathrm{S}\mathbf{a}_2 - l\,\mathbf{a}_1^{\mathbf{T}}\mathbf{a}_2 - f\,\mathbf{a}_1^{\mathbf{T}}\mathbf{a}_1 = 0$

This then reduces due to the constraints of normalization and uncorrelation:

$f = 0$

Substituting this result into the differentiated equation yields the eigenvalue-eigenvector problem:

$\mathrm{S}\mathbf{a}_2 - l\,\mathbf{a}_2 = (\mathrm{S} - l\mathbf{I}_p)\mathbf{a}_2 = 0$

Similar to before, $\mathbf{a_2}$ is an eigenvector, and l an eigenvalue of S. Additionally, $\mathbf{a}_2^{\mathbf{T}}\mathrm{S}\mathbf{a}_2 = l$; therefore, l is the largest eigenvalue l_2, and $\mathbf{a_2}$ the corresponding eigenvector. This process continues until all principal components are obtained or the maximum number of principal components desired is obtained. One method to determine the importance of consecutive principal components is based on the relative magnitude of the corresponding eigenvalues. Consider the eigenvalue matrix:

$$\Lambda = \begin{bmatrix} l_1 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & l_2 & 0 & & & \cdot \\ \cdot & 0 & \cdot & 0 & & \cdot \\ \cdot & & 0 & \cdot & 0 & \cdot \\ \cdot & & & 0 & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & 0 & l_p \end{bmatrix}$$

The standard eigenvalue problem in matrix form can then be represented by:

$$S\mathbf{A} = L\mathbf{A}$$

The total variance of the data is given by:

$trace(\Lambda) = \sum_{i=1}^{p} l_i$ = total variance of the data.

A measure of the importance of each PC expressed as percent variance explained is given by:

$$\frac{l_i}{trace(\Lambda)} \times 100\%$$

By looking at consecutive importance values one can determine the number of relevant principal components using a simple heuristic. A simple heuristic is to choose the number of principal components that provides a given percentage of the total variance, e.g. 95%.

3.3.2 Principal Component Regression

The application of PCA techniques to regression is referred to as PCR. PCR uses the PC scores ($\mathbf{Z}$) in place of the predictor variables in the standard OLS model.

$\mathbf{Z} = \mathbf{XA}$ = the $n \times a$ reduced matrix of the PC scores, where the full matrix $\mathbf{X}$ is $n \times p$, $p > a$.

Due to the orthogonality of $\mathbf{A}$, this can be also be expressed as:

$$\mathbf{X} = \mathbf{ZA}^{\mathbf{T}}$$

Beginning with the standard OLS equation:

$$\mathbf{y} = \mathbf{X}\hat{\mathbf{b}} + \mathbf{e}$$

Substituting in the PCA equivalent for $\mathbf{X}$:

$$\mathbf{y} = \mathbf{Z}\mathbf{A}^{\mathbf{T}}\hat{\mathbf{b}} + \mathbf{e}$$

Define $\hat{\mathbf{c}} = \mathbf{A}^{\mathbf{T}}\hat{\mathbf{b}}$, such that:

$$\mathbf{y} = \mathbf{Z}\hat{\mathbf{c}} + \mathbf{e}$$

$\hat{\mathbf{c}}$ = the regression coefficients estimated via least squares

$\mathbf{e}$ = the residual produced via the estimation

Calculation of the regression coefficients $\hat{\mathbf{c}}$ for PCR is not susceptible to collinearity issues, as is OLS, due to the orthogonality of $\mathbf{Z}$ ($\mathbf{Z}^{\mathbf{T}}\mathbf{Z} = \mathbf{S}^2$), where $\mathbf{S}^2 = Var(\mathbf{X})$, assuming $\mathbf{X}$ is mean-centered. The elimination of the effects of collinearity is enabled by the reduction of the variance in the matrix of PC scores $\mathbf{Z}$, with respect to the original data $\mathbf{X}$. Thus it is assumed that the matrix $\mathbf{Z}$ is well-conditioned, which is a reasonable assumption if in fact $a << p$, i.e. only the most important (importance being defined as variance) principal components are retained. The regression coefficients $\hat{\mathbf{c}}$ are computed via:

$$\hat{\mathbf{c}} = (\mathbf{Z}^{\mathbf{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathbf{T}}\mathbf{y} = \mathbf{S}^{-2}\mathbf{Z}^{\mathbf{T}}\mathbf{y}$$

Thus leading to the estimation of the response:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{A}\hat{\mathbf{c}}$$

3.4 PARTIAL LEAST SQUARES (LINEAR AND NONLINEAR METHODS)

First introduced by H. Wold in the field of econometrics [Wold 1966], PLS has become an important technique in many areas, including psychology, economics, chemical engineering, medicine, pharmaceutical science, and process modeling. PLS is a class of techniques for modeling the associations between blocks of observed variables by means of latent variables [Wegelin 2000]. These latent variables are created through a supervised transformation, whereby an orthogonal basis results.

The PLS algorithm used in this work is a special case of the standard technique, where the **Y** block consists of a single column or response variable. This case of PLS modeling is referred to as PLS-1 [Gil 1998] and is employed here due to its regularization properties, putting it in the same class as the methods of RR and PCR [Wegelin 2000]. Extensive discussions of PLS methods for the general case of more than one response variable are available in the literature [Geladi 1986, Höskuldsson 1988, Höskuldsson 1995, Martens 1989]. All discussions presented here refer to the PLS-1 technique. The PLS-1 technique is an inferential modeling method. The inferential structure provides the additional benefit of eliminating the ability for perturbations in a specific variable to directly propagate to the prediction of that variable. This is due to the architecture of inferential models, in which a variable's response is inferred through the use of other variables correlated with it.

PLS-1 is an inferential modeling technique of the class of regularization techniques. It has been directly compared to RR and TSVD in a signal validation study [Hines 1999], as well as to RR and PCR in a chemical application [Wold 1984]. In both cases, favorable results were reported for the PLS algorithm comparisons, and stable solutions were obtained. The chemical application resulted in a PLS model exhibiting a solution which was stabilized in comparison to the OLS solution, and comparable in prediction error to RR. The signal validation study was based on a predictor variable block plagued with collinearity, and reported a highly stable PLS solution in comparison to un-regularized neural networks, and OLS solutions. This stability results from the orthogonalization of the predictor variable block, and the reduced variance of the estimates due to the elimination of higher ordered latent variable pairs. These higher ordered latent variable pairs contain the least amount of variation related to the response variable, and are generally considered to be noise.

The inner relationships of the standard PLS algorithm are constructed using univariate regressions on the latent vectors. In attempts to enhance the technique, quadratic functions have been introduced into the inner relationship [Baffi 1999a, Ni 1995, Wold

1989, Wold 1992]. However, quadratics are still linear in their parameters and do not guarantee a proper solution [Bro 1995]. The use of single hidden layer feedforward neural networks (NN) has been suggested [Gemperline 1991], and applied in the field of chemical engineering [Baffi 1999b, Holcomb 1992, Malthouse 1997, Qin 1992, Qin 1996]. These methods have been under study at the University of Tennessee, for the purposes of signal validation in large-scale processes, beginning in late 1999 [Rasmussen 2000b]. The method used herein, uses fully connected feedforward neural networks in the inner relationships and maintains the linear orthogonalization in the outer relationships. This method will be referred to as neural network partial least squares (NNPLS), to avoid confusion with the various nonlinear methods utilizing quadratics or radial basis functions [Wilson 1997a, Wilson 1997b]. A NNPLS signal validation system has been implemented, on a trial basis, at the 9th unit of Tennessee Valley Authority's Kingston fossil plant, in Harriman, Tennessee, USA [Hines 2000c, Hines 2001, Rasmussen 2000a, Rasmussen 2000b].

Two features of the PLS algorithm provide extensive benefits for designing empirical models for signal validation tasks. The first is its elimination of numerical problems associated with collinearity. The second beneficial feature of PLS is its elimination of the higher ordered latent vectors from passing through to the inner relationships of the model. These higher order latent variables contain decreasing amounts of variance related to the response variable block $\mathbf{Y}$. This inherent regularization, via supervised dimensionality reduction, provides reproducible and stable solutions. The benefits associated with regularization, to stabilize the solutions of ill-posed problems, are making its incorporation into the design of signal validation systems essential [Dayal 1997].

There are two sets of relationships involved in the PLS technique, the outer relationships and the inner relationships. Consider an $n \times p$ block of predictor variables, $\mathbf{X}$, and an $p \times 1$ response variable, $\mathbf{Y}$. Note that the restriction of the response variable matrix to a single variable is the case for PLS-1. The outer relationships are linear relationships between the predictor variables, $\mathbf{X}$, and the latent variables for the same block. The

creation of latent variable pairs is an iterative process. The diagram of the mapping performed by a single iteration of the PLS-1 algorithm provides further illustration (Figure 3.4.1).

Following the creation of a latent variable pair, the predictor variables are successively deflated by the information extracted by the current latent variable pair. For this reason, the observed variable block is often referred to as a residual matrix, and herein will be referred to as the input residual matrix. Similarly, there is an output residual matrix created by the deflation of the response variable matrix at the end of each iteration. The deflation operations are not an essential step in the PLS algorithm [Höskuldsson 1996]; however, the residual matrices provide information regarding the amount of variance contained in each successive set of latent variable pairs. The outer relationships for the predictor variables and the subsequent deflation of the input residual matrix are given below.

Define:

$\mathbf{E}$ = input residual matrix

$\mathbf{f}$ = output residual matrix

$\mathbf{w}$ = predictor variable transformation weights

$\mathbf{t}$ = input latent variable

$\mathbf{b}$ = regression coefficient

$\mathbf{p}$ = input loading vector

$a = 1,..., R$ (iteration index)

$j = 1,..., n$ (column index)

The outer relationship can be explicitly written as:

$$\mathbf{t}^a = \mathbf{E}^{a-1}_{(:,1)}\mathbf{w}_{(1,a)} + \mathbf{E}^{a-1}_{(:,2)}\mathbf{w}_{(2,a)} + ... + \mathbf{E}^{a-1}_{(:,n)}\mathbf{w}_{(n,a)}$$

The deflation of the input residual matrix is given by:

$$\mathbf{E^a} = \mathbf{E}^{a-1} - \mathbf{t}^a(\mathbf{p}^a)^T$$

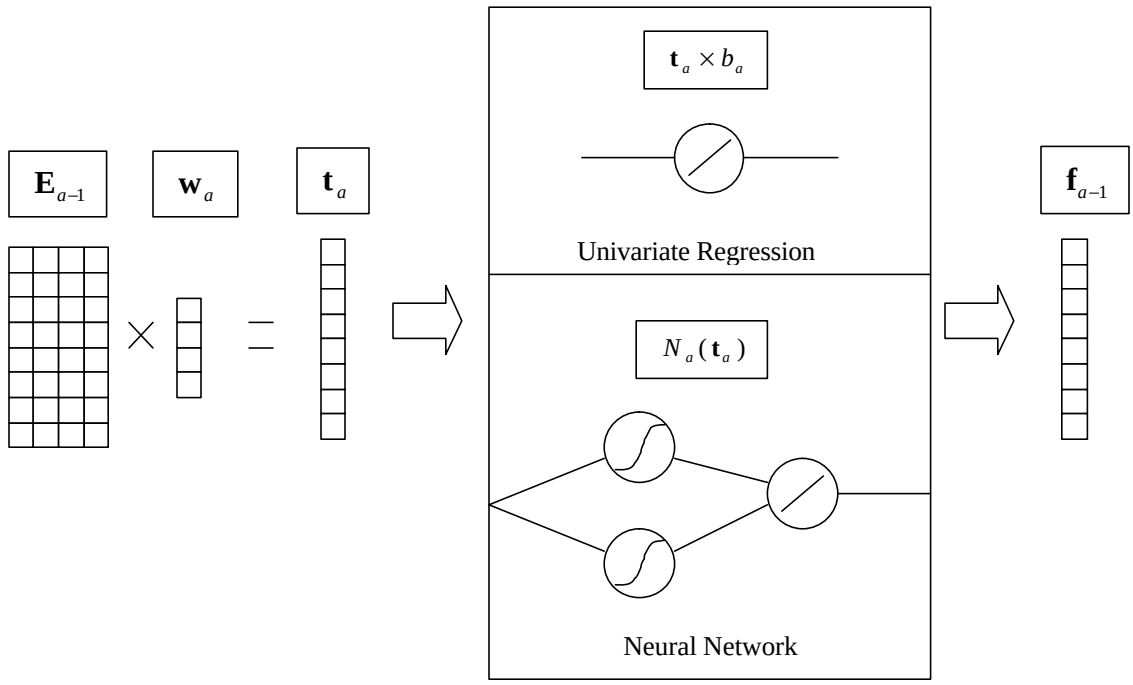


Figure 3.4.1: Schematic of PLS-1 / NNPLS-1 latent variable pair mapping

Note that $\mathbf{E}^0 = \mathbf{X}$, subscripts indicate the matrix indices, and superscripts indicate the iteration index. The number of iterations of the outer relationship, a, dictates the number of pairs of latent variables, and constitutes selection of the model. The number of latent variable pairs computed is the rank of the PLS model, R, and is often determined via a cross validation technique [Hines 1999]. The PLS-1 algorithm does not require the transformation of the response variable, since it is a rank one matrix. Hence, there is no corresponding outer relationship for the response variable. The optional task of deflation of the output residual vector may still be carried out if desired, to provide a quantification of the variability explained by each consecutive input latent variable. The deflation of the output residual is given by:

$$\mathbf{f}^a = \mathbf{f}^{a-1} - \mathbf{t}^a \mathbf{b}^a$$

Note that $\mathbf{f}^0 = \mathbf{y}$, and $\mathbf{b}^a$ is the coefficient from the univariate regression of $\mathbf{t}^a$ onto $\mathbf{f}^a$. All of the discussions regarding the input outer relationships of the PLS algorithm also apply to the NNPLS algorithm. Differences between the methods occur in the inner relationships, and slightly modify the equation for the deflation of the output residual.

The inner relationships of the PLS-1 algorithm are a set of univariate regressions between each latent variable pair up to the rank of the model. Substituting these univariate regressions with feedforward neural networks provides the algorithm with nonlinear mapping capabilities, and will often produce a more parsimonious model [Malthouse 1997]. Following the method of S.J. Qin and T.J. McAvoy, each latent variable pair is mapped with a single input single output (SISO) neural network [Qin 1992]. The use of very simple neural networks circumvents the over-parameterized problem of the direct neural network approach. The combination of the PLS-1 linear outer relationships and the neural network inner relationships will be referred to as NNPLS-1.

3.4.1 Presentation of NNPLS-1 Algorithm

The neural networks used in the inner relation contain a single hidden layer of neurons containing hyperbolic tangent activation functions, and a single output neuron with a

linear activation function. The neural networks were trained with the Levenberg-Marquardt (LM) training algorithm [Levenberg 1944, Marquardt 1963]. Cross validation training was used for early stopping of the LM algorithm. Initialization of the neural network weight and bias values can be based on the univariate regression solution. This method of initialization will decrease the time required to train the neural networks, and due to the training algorithm finding the nearest local minimum from the initial point, the solution will be better than the initial linear model [Qin 1992].

Each latent variable pair requires a SISO neural network. The neural network weights and biases for each set of score vectors are determined during the current iteration and the explained variation is subtracted from the output residual via network simulation. The full NNPLS-1 algorithm is given below, where the 1 indicates that it is an inferential design (a single response variable):

1.	$a = 1$	Initialize iteration index
2.	$\mathbf{E}_0 = \mathbf{X}$	Input residual matrix
3.	$\mathbf{f}_0 = \mathbf{y}$	Output residual vector
4.	$\mathbf{w}_a^* = \mathbf{E}_{a-1}^{\mathbf{T}} \mathbf{f}_{a-1}$	Calculation of transformation weights
5.	$\mathbf{w}_a = \mathbf{w}_a^* / \| \mathbf{w}_a^* \|$	Normalization
6.	$\mathbf{t}_a = \mathbf{E}_{a-1} \mathbf{w}_a$	Calculation of input scores
7.	$\mathbf{p}_a^{\mathbf{T}} = \mathbf{t}_a^{\mathbf{T}} \mathbf{E}_{a-1} / \mathbf{t}_a^{\mathbf{T}} \mathbf{t}_a$	Calculation of input loadings
8.	$TRAIN\{N_a(\mathbf{t}_a, \mathbf{f}_{a-1})\}$	Neural network training
9.	$\mathbf{f}_a = \mathbf{f}_{a-1} - N(\mathbf{t}_a)$	Calculation of output residual variable
10.	$\mathbf{E}_a = \mathbf{E}_{a-1} - \mathbf{t}_a \mathbf{p}_a^{\mathbf{T}}$	Calculation of input residual matrix
11.	*if* $a < R$	Update iteration index if less then rank
	$a = a + 1$	
	goto 4	
else		
end		

The rank of the model, R, is the number of latent variable pairs computed. It is difficult to determine the optimal rank or dimension of the model without some measure of the model's performance at each dimension. A common performance measure is the sum squared error (SSE), and is computed with respect to an independent set of data that was not used during model development. The optimal dimension is then defined as the dimension at which the SSE was at its minimum. Data should be scaled to zero mean and unit variance prior to using the NNPLS-1 algorithm.

3.5 ARTIFICIAL NEURAL NETWORKS

Artificial neural network (ANN) models, inspired by biological neurons, contain layers of simple computing nodes that operate as nonlinear summing devices. These nodes are highly interconnected with weighted connection lines, and these weights are adjusted when training data are presented to the ANN during the training process. In a broad sense, this is an emulation of the learning process of the highly interconnected set of neurons of the human brain. Though, the magnitude and complexity of the human system is unattainable through computational methods, it is a model by which complex problems can be solved in limited form. Additionally, the training process often identifies underlying relationships between environmental variables that were previously unknown. Successfully trained ANNs can perform a variety of tasks, the most common of which are: prediction of an output value, classification, function approximation, and pattern recognition.

Neural networks are currently being used in many fields including military target recognition, financial decision support, detection of manufacturing defects, robotics, control systems, medical diagnostics, image processing, instrument monitoring, and medical decision support [Dayhoff 2001]. Additional applications have been presented for signal validation in the power industry [Fantoni 1996, Hines 1997a, Xu 2000].

The first computational, trainable neural networks were developed in 1962 [Rosenblatt 1962, Widrow 1985]. This network, labeled the perceptron, contained two layers of computational nodes and a single layer of interconnections. To clarify the terminology, only layers of a neural network that have an associated set of connection weights will be recognized as legitimate processing layers. The input layer of a neural network is not a true processing layer because it does not have an associated set of weights. The output layer on the other hand does have a set of associated weights. Thus, the most efficient terminology for describing the number of layers in a neural network is through the use of the term, hidden layer. A hidden layer is a *legitimate* layer exclusive of the output layer. Thus, the perceptron can be described as having zero hidden layers, only an output layer. The perceptron was limited to the solution of linearly-separable problems. The expansion to solve nonlinear problems, discrimination and function approximation, was made available in 1974 [Werbos 1974], and referred to as the multilayered perceptron (MLP). MLPs were trained with a gradient-descent method referred to as back-error propagation. The basic MLP consists of a single hidden layer and an output layer, though additional hidden layers can be included (Figure 3.5.1). Note that MLPs have at least one additional processing layer than the perceptron. Most MLPs are fully connected in that each processing unit of each layer is connected to all processing units of the next layer.

The computational capabilities of MLPs were proven by Hecht-Nielson [1989], Hornik [1989] and Cybenko [1989] in the general function approximation theorem which states that a neural network could approximate an arbitrary nonlinear function. Additional references for the universal approximation theorem of neural networks are Hornik [1990, 1993], and White [1989]. The result that artificial neural networks with a single hidden layer can approximate any nonlinear function combined with the neural network's abilities to create the functional form and the fitting function simultaneously, provide a decided advantage over traditional statistical multivariate regression techniques where a fit is forced to a prechosen function [Dayhoff 2001]. Artificial neural networks are multifactorial mathematic models that have been applied successfully in the prediction,

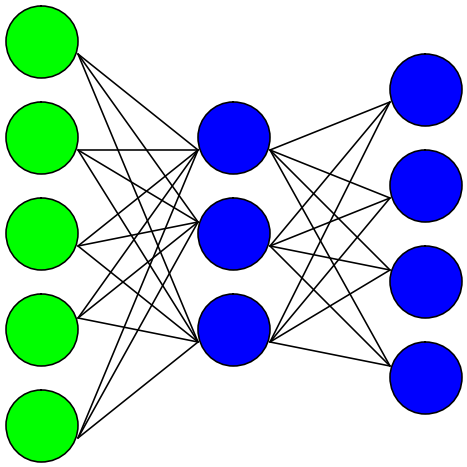


Figure 3.5.1: Single hidden layer multilayer perceptron

classification, function estimation, pattern recognition, and pattern completion problems in many disciplines. Because all of these problems can be formulated as function estimation problems, and because there is theoretic evidence that neural networks can approximate any nonlinear function with any degree of accuracy, neural networks have the potential to be highly effective in practically any discipline [Dayhoff 2001].

The neural network training process begins with the initialization of its weights to small random numbers. The network is then presented with the training data which consists of a set of input vectors and corresponding desired outputs, often referred to as targets. The neural network training process is an iterative adjustment of the internal weights to bring the network's outputs closer to the desired values, given a specified set of input vector / target pairs. Weights are adjusted to increase the likelihood that the network will compute the desired output. The training process attempts to minimize the mean squared error (MSE) between the network's output values and the desired output values. While minimization of the MSE function is by far the most common approach, other error functions are available. The error surface often contains a variety of local minima along with a global minimum. Local minima can be problematic during network training because of the use of the gradient descent methods for weight adjustment. The presence of many local minima, along with the random initialization methods of neural networks leads to the variety of solutions obtained when training the same neural network repeatedly, using the same training set. It should also be recognized that the resultant trained neural network is dependent on the data used to train the network. Changing the training set will most likely lead to an alternate solution, even if the same initial random values are selected for the connection weights. Because the neural network has undergone generalized learning rather than memorization, it is, in a literal mathematical sense, interpolating or extrapolating for new incoming cases. An extremely important consideration in training neural networks is that the training data must provide examples of all conditions for which accurate predictions will be queried. That is not to say that all possible conditions must exist in the training data, but that the training data should provide adequate coverage of these conditions. Neural networks will provide

interpolative predictions, but the training data must provide adequate coverage above and below the interpolation site for this prediction to be sufficiently accurate. Accurate extrapolation, i.e. providing estimations for data that resides outside of the training of the training data, is not possible for neural networks due to the shape of the nonlinear functions utilized in the processing units of the hidden layers of the neural network.

A trained neural network does not provide a unique solution because the final set of neural network weights depends on several factors, including: the initial random weight values, the division of the data into the training, validation, and verification sets, and the neural network training algorithm. Methods of evaluating the effects of these different factors are usually based on repeated trials. For example, to evaluate the effects of the random weight initialization repeated results are obtained using the same data sets with different initial weights. The effect of data set division can be evaluated through bootstrap sampling (random sampling with replacement).

Important considerations when applying neural networks are: methods of data collection, data types and data-scaling, data division and validation schemes, and quantification of confidence intervals for the predictions. Neural network development requires three non-overlapping sets of data: training set, validation set, and test or verification set. Often the data are divided at random into these three sets; however, some oversight should be incorporated to ensure that the data in the training set adequately represents the conditions present in the validation and verification sets. The training set is used to guide the adjustment of the connection weights during the training process. The validation set is used to prevent overtraining. Overtraining occurs when a network begins to learn the unique characteristics of the training set that are not characteristic of the validation and verification sets. This often is stated as: "the network learns the noise in the training set." To prevent overtraining, the connection weights are adjusted to minimize the MSE between the training set predictions and the desired outputs, then using these weights, output values are computed for the validation set along with the corresponding validation MSE. This process is alternately repeated and results in the training and validation MSEs

decreasing during the early stages of the training process. Eventually, the MSE of the validation set will reach a minimum and then begin to increase. At this point, the weights of the neural network are being fine-tuned to memorize the conditions present in the training set which are unique; the network is learning the noise in the training set. Because these unique details are not consistent in the validation set, its MSE begins to increase. The optimal solution for the network occurs when the validation MSE is at its minimum, and network training is stopped at this point. Once a network has been trained, it is presented with the verification set for which it produces output values. The performance on the verification set is reported and used in determining the validity of the neural network solution. The verification set is independent of the training and validation sets, and the neural network results on the verification set can be considered a true (unbiased) estimation of the neural network performance on new data. The performance on the verification set provides a proper benchmark evaluation metric for the performance of the neural network as a predictor or classifier when deployed for actual use.

While neural networks provide powerful nonparametric function approximation tools, it should be noted that in many situations, the neural network architecture and training can be shown to be analogous to standard statistical modeling paradigms which often have a stronger theoretical foundation regarding the properties of the estimators. Many applications employ neural networks as a black box modeling approach with no considerations of the statistical properties of the resultant neural network solution to a given situation. In many cases, the learning algorithms applied to neural network training are slow to converge and are inefficient with respect to standard optimization algorithms such as those used for nonlinear regression. Neural network algorithms are inefficient because they are intended to be implemented on parallel computers, but are most often implemented on standard PCs performing serial computations. Most MLPs are equivalent to nonlinear regression or nonlinear discriminant models and nonlinear regression optimization algorithms can perform an equivalent fit to the data orders of magnitude faster than the standard neural network algorithms [Sarle 1994].

The standard perceptron (one layer of trainable weights) has been shown to be equivalent to linear regression, logistic regression, and a linear discriminant function utilizing the linear [Weisberg 1985, Myers 1986], logistic [Hosmer 1989], and threshold activation functions respectively [Hand 1981, McLachlan 1992, Weiss 1991]. MLPs have also been shown to be analogous to standard statistical procedures depending on the complexity (number of hidden neurons and hidden layers). An MLP with one hidden layer is essentially the same as the projection pursuit regression model except that the MLP uses a predetermined functional form for the hidden layer activation functions, whereas projection pursuit uses a flexible nonlinear smoother [Sarle 1994]. If the number of hidden neurons is allowed to increase with sample size, an MLP becomes a nonparametric sieve [White 1992]. Sarle [1994] indicates that nonparametric sieves are a useful alternative to kernel regression and smoothing splines.

MLPs have some attractive features such as the ability to vary complexity by adding or removing neurons or layers. They are easy to extend to multiple inputs and multiple outputs without an exponential increase in the number of parameters. The discussion regarding the relationships between neural networks and more standard statistical methods intends only to illuminate the fact that there is no black box approach to statistical modeling and estimation. Even a simple linear regression requires considerable knowledge of the method itself as well as the positive and negative influences on regression models due to the data and its distributions. The knowledge required increases for nonlinear regression. This is not to say that a neural network model, of sufficient accuracy, cannot be easily constructed and deployed using one of the many available software packages; however, without oversight and considerable knowledge the apparent accuracy cannot be guaranteed and the limitations of the model are relatively unknown. The full mathematical details of the ANNs used during this work are provided in the appropriate section

3.6 NONLINEAR REGRESSION

Both the ANN model and the NNPLS model can be represented within the framework of nonlinear regression. It is the linearization of the nonlinear regression model, also referred to as the delta method that provides the basis for the analytical prediction intervals used in this work; thus, it is a relevant endeavor to first study the general nonlinear regression model as presented by Draper and Smith [1966]:

A general linear model is given by :

$$Y = b_0 + b_1 Z_1 + b_2 Z_2 + \cdots b_{p-1} Z_{p-1} + e$$

Z_i can represent any functions of the basic predictor variables $X_1, X_2, ..., X_k$.

Any model that is not of the form as above is considered a nonlinear model, i.e. nonlinear in the parameters. In the following discussions when referring to a nonlinear model, it is assumed that the model is *intrinsically nonlinear*, such that it is impossible to convert the model into a form linear in the parameters. The well established notations for nonlinear least squares provide alternate symbols for the predictor variables and the model parameters. In this work, the only change that is adopted is the use of the parameter notation of $\vec{q} = [q_1 \quad q_2 \quad \cdots \quad q_p]$, rather than the linear notation $\boldsymbol{\beta} = [b_0 \quad b_1 \quad \cdots \quad b_p]$.

Thus a general nonlinear model can be written as:

$$\mathbf{Y}_i = f(\mathbf{x}_i, \vec{q}) + \mathbf{e}_i$$

where: $\mathbf{x}_i = [X_{i,1} \quad X_{i,2} \quad \cdots \quad X_{i,p}]$, $i = 1, ..., n$

p is the number of predictor variables

$\mathbf{e} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \cdots \quad \mathbf{e}_n]$

The assumptions of normality and independence are $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}s^2)$.

The sum of squares error is then:

$$S(\vec{q}) = \sum_{i=1}^{n} \left\{ \mathbf{Y}_i - f(\mathbf{x}_i, \vec{q}) \right\}^2$$

A value of $\vec{q}$ which minimizes $S(\vec{q})$ is denoted $\hat{\vec{q}}$

If the assumptions of normality and independence are valid, the least squares estimator of $\vec{q}$, is also the maximum likelihood estimate of $\vec{q}$. Consider the likelihood function for this problem: $\ell(\vec{q}, s^2) = (2ps^2)^{-\frac{n}{2}} e^{-\frac{S(\vec{q})}{2s^2}}$

If s^2 is known, maximizing the likelihood function with respect to $\vec{q}$ is equivalent to minimizing $S(\vec{q})$ with respect to $\vec{q}$. To obtain the least squares estimator, the derivative of the sum of squares error with respect to $\vec{q}$, is set to zero, resulting in the p normal equations:

$\sum_{i=1}^{n} \left\{ \mathbf{Y}_i - f(\mathbf{x}_i, \hat{\vec{q}}) \right\} \cdot \left[\frac{\partial f(\mathbf{x}_i, \vec{q})}{\partial q_k} \right]_{\vec{q}=\hat{\vec{q}}}$ Where $k = 1, ..., p$, the total number of parameters in the model.

Solving the p normal equations for $\hat{\vec{q}}$ provides the least squares estimate.

The solution of the normal equations can be very difficult to obtain and iterative methods must be used in almost all cases and multiple solutions may exist.

Three common methods of obtaining the parameter estimates are: linearization, steepest descent, and Marquardt's compromise. Each of these is discussed in the following 3 sections.

3.6.1 The Linearization Approach to the Least Squares Solution for Nonlinear Regression

$\vec{q}_0 = [q_{1,0} \quad q_{2,0} \quad \cdots \quad q_{p,0}]$ are the initial values of the parameters.

A Taylor series expansion (to the first order) of $f(\mathbf{x}_i, \vec{q})$ about $\vec{q}_0$ is given by:

$$f(\mathbf{x}_i, \vec{q}) = f(\mathbf{x}_i, \vec{q}_0) + \sum_{k=1}^{p} \left\{ \left[\frac{\partial f(\mathbf{x}_i, \vec{q})}{\partial q_k} \right]_{\vec{q}=\vec{q}_0} (q_k - q_{k,0}) \right\}$$

Define:

$$f_i^0 = f(\mathbf{x}_i, \vec{q}_0),\ b_k^0 = q_k - q_k^0, \text{ and } Z_{i,k}^0 = \left[\frac{\partial f(\mathbf{x}_i, \vec{q})}{\partial q_k}\right]_{\vec{q}=\vec{q}_0}$$

The nonlinear model near $\vec{q}_0$, can now be approximated by the linear form:

$$\mathbf{Y}_i = f(\mathbf{x}_i, \vec{q}) = f_i^0 + \sum_{k=1}^{p} b_k^0 Z_{i,k}^0 + e_i$$

Define:

$$\mathbf{Z}_0 = \begin{bmatrix} Z_{1,1}^0 & Z_{1,2}^0 & \cdot & \cdot & \cdot & Z_{1,p}^0 \\ Z_{2,1}^0 & Z_{2,2}^0 & \cdot & \cdot & \cdot & Z_{2,p}^0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ Z_{i,1}^0 & Z_{i,2}^0 & \cdot & \cdot & \cdot & Z_{i,p}^0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ Z_{n,1}^0 & Z_{n,2}^0 & \cdot & \cdot & \cdot & Z_{n,p}^0 \end{bmatrix}, \quad \mathbf{b}_0 = \begin{bmatrix} b_1^0 \\ b_2^0 \\ \cdot \\ \cdot \\ \cdot \\ b_p^0 \end{bmatrix}, \text{ and } \quad \mathbf{y}_0 = \mathbf{Y} - \mathbf{f}^0 = \begin{bmatrix} \mathbf{Y}_1 - f_1^0 \\ \mathbf{Y}_2 - f_2^0 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{Y}_i - f_i^0 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{Y}_n - f_n^0 \end{bmatrix}$$

Then the least squares estimate of the model parameters is given by:

$\mathbf{b}_0 = \left(\mathbf{Z}_0^{\mathbf{T}} \mathbf{Z}_0\right)^{-1} \mathbf{Z}_0^{\mathbf{T}} \mathbf{y}_0$, where $\mathbf{b}_0$ is the estimate of $\mathbf{ß}_0 = \begin{bmatrix} b_1^0 & b_2^0 & \cdot & \cdot & \cdot & b_p^0 \end{bmatrix}$.

Which minimizes the sum of squared errors of the linearized model:

$$S(\vec{q}) = \sum_{i=1}^{n} \left\{ \mathbf{Y}_i - f(\mathbf{x}_i, \vec{q}_0) - \sum_{k=1}^{p} b_i^0 Z_{i,k}^0 \right\}^2$$

The scalar values of the estimated parameter vector, $\mathbf{b}_0$, represent $b_k^0 = q_{k,1} - q_{k,0}$, and the $q_{k,1}$ $(k = 1,..., p)$ can be thought of as the revised estimates of $\vec{q}$.

This process is repeated by placing the current estimate in the role of the initial estimate and following the same procedure. Convergence is defined as:

$$\left| \{ q_{k,j+1} - q_{k,j} \} / q_{k,j} \right| < d$$

where $\vec{q}_j = \left[q_{1,j} \quad q_{2,j} \quad \cdot \quad \cdot \quad \cdot \quad q_{p,j} \right]$, and $\vec{q}_{j+1} = \vec{q}_j + \mathbf{b}_j$

For models which are linear in their parameters, the surface contours of $S(\vec{q})$, in the p dimensional space, are ellipsoid having a single minimum height. Where the height is relative to the magnitude of $S(\vec{q})$. Irregular contours result from nonlinear models, often having several local and global minima. A global minimum is the minimum height of the $S(\vec{q})$ contour in the p dimensional space. There may be more than one $\vec{q}$ where the minimum height is attained. The local minima are points where low heights in the error contours occur, but do not necessarily correspond to a global minimum. When the shape of the error surface contours near the least squares estimator, $\hat{\vec{q}}$, are elongated such that many different values of $\hat{\vec{q}}$ produce sufficiently similar results, the problem is said to be ill-conditioned. Ill-conditioning may indicate overparameterization, or inadequate data for the specified model. A simple measure of Ill-conditioning can be obtained from the ratio of the largest to the smallest eigenvalues of $\left(\mathbf{Z}_j^{\mathrm{T}} \mathbf{Z}_j \right)$. This quantity is often referred to as the condition number. Reasonable values are less than 100, and for data sets with a high degree of collinearity, as in many signal validation applications, the condition numbers may be $>10^3$.

In general, when a linearized form of a nonlinear model is used, all of the usual formulae and analyses of linear regression theory can be applied. Any results obtained are valid only to the extent that the linearized approximation provides a good approximation to the true value. There are some exceptions to this generalization when the model is nonlinear. Assuming $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}s^2)$, $\hat{\vec{q}}$ is no longer normally distributed, $s^2 = S(\hat{\vec{q}})/(n-p)$ is no

longer an unbiased estimate of s^2, and there is no variance-covariance matrix of the form $(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}s^2$.

3.6.2 The Gradient Descent Approach to the Least Squares Solution for Nonlinear Regression

An alternate approach to finding $\hat{\vec{q}}$ which minimizes $S(\vec{q})$ is the gradient descent approach. The basic process of gradient descent is to start from an initial point $\vec{q}_0$, and move along the vector with components: $\left[-\frac{\partial S(\vec{q})}{\partial q_1} \quad -\frac{\partial S(\vec{q})}{\partial q_2} \quad \cdot \quad \cdot \quad \cdot \quad -\frac{\partial S(\vec{q})}{\partial q_p}\right]$.

To avoid evaluating the derivatives, the vector slope components at various places on the surface $S(\vec{q})$ can be fit with planar approximations. A number of locations on the surface are used and defined by the selected levels of $q_1, q_2, ..., q_p$. A linear model can then be formed using the evaluated $S(\vec{q})$ values as the dependent variable, and the combinations of levels as the predictor variable observations:

Observed $S(\vec{q}) = b_0 + \sum_{k=1}^{p} b_k (q_k - \bar{q}_k)/l_k + \mathbf{e}$

$\bar{q}_k = \sum_{r=1}^{m} q_{k,r}/m$, and l_k is a scaling factor chosen so that $\sum_{r=1}^{m}\left[(q_{k,r} - \bar{q}_k)/l_k\right]^2$ = constant.

r is the run index, or the number of selected locations at which the surface function is evaluated, $r = 1, ..., m$.

The negative of the coefficients obtained from the least squares minimization of the observed $S(\vec{q})$ indicate the direction of steepest descent. As long as the linear approximation is reasonable, the maximum decrease in $S(\vec{q})$ will be obtained by moving along the line which contains points such that: $(q_k - \bar{q}_k)/l_k \propto -b_k$. Defining a proportionality factor this can be written as $(q_k - \bar{q}_k)/l_k = -r \cdot b_k$, for $r > 0$. A number

of values for the proportionality factor are selected, and the path of steepest descent is followed as long as $S(\vec{q})$ decreases. Repeated applications using different proportionality factors can be used to obtain the minimum $S(\vec{q})$. This brief overview of gradient descent follows the presentation in Draper and Smith [1966], and full details are provided in Box and Draper [1987].

In general, the steepest descent method progresses rapidly during the initial steps but slows considerably when the surface $S(\vec{q})$ is from a nonlinear model, especially under circumstances of ill-conditioning. The steepest descent method rapidly approaches the region of the surface where the maximum likelihood estimator, $\hat{\vec{q}}$, resides; however; near the location of a minimum in the surface linearization tends to work better.

3.6.3 Marquardt's Compromise for Finding the Least Squares Solution for Nonlinear Regression

Marquardt [1963] developed a method that combines the rapid initial convergence of gradient descent with the superior performance of the linearization technique when approaching a minimum. This combined method has proven to work well for a variety of nonlinear problems and while no method is optimal for all problems, in the absence of *apriori* knowledge, it is the best choice. The compromise provides a method to interpolate between the two vector directions provided by gradient descent and linearization, as well as a method for obtaining a suitable step size. Interested readers are referred to Seber and Wild [1989].

3.7 LEVENBERG-MARQUARDT TRAINING ALGORITHM

The application of Marquardt's compromise to ANNs is referred to as the Levenberg-Marquardt (LM) algorithm [Marquardt 1963, Levenberg 1944]. The LM training algorithm was used exclusively in this work. Its benefits regarding rapid convergence to

the region of the error minimum and optimal performance near the minimum, due to the continuous adjustment of a scaling factor, were shown to provide sufficient training results for a variety of test cases studied. The basics of the algorithm are provided below presented, in modified form, from Fine [1999].

Consider the sum of squares error function $S(\vec{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, where $i = 1,...,n$ and n is the number of response values in the training set. Define $e_i = (y_i - \hat{y}_i)$, and a set of p neural network parameters $\vec{w} = [w_1 \quad w_2 \quad \cdots \quad w_j \quad \cdots \quad w_p]$. The $(p \times n)$ Jacobian with respect to the network parameters is given by $\mathbf{J}$, each element of which can be specified from: $J_{j,i} = \frac{\partial e_i}{\partial w_j}$. The p dimensional gradient for the quadratic error function is given by: $\mathbf{g}(\vec{w}) = \mathbf{J} \cdot \mathbf{e}$, where $\mathbf{e}$ is the $(n \times 1)$ vector of error terms.

The Hessian can similarly be defined:

$$H = \frac{\partial[S(\vec{w})]}{\partial w_j \partial w_k} = \frac{1}{2}\sum_{i=1}^{n}\frac{\partial^2 e_i^2}{\partial w_j \partial w_k} = \sum_{i=1}^{n}\left[e_i \frac{\partial^2 e_i^2}{\partial w_j \partial w_k} + \frac{\partial e_i}{\partial w_j}\frac{\partial e_i}{\partial w_k}\right] = \sum_{i=1}^{n}\left[e_i \frac{\partial^2 e_i^2}{\partial w_j \partial w_k} + J_{j,i} \cdot J_{k,i}\right]$$

Defining $\mathbf{D} = \sum_{i=1}^{n} e_i^2 \nabla^2 e_i^2$, allows the above to be rewritten as:

$$\mathbf{H}(\vec{w}) = \mathbf{J}\mathbf{J}^T + \mathbf{D}$$

Levenberg-Marquardt algorithms replace the matrix of second derivatives ($\mathbf{D}$) with a positively scaled unit matrix $e\mathbf{I}$, resulting in the following approximation:

$$\hat{\mathbf{H}}(\vec{w}) = \mathbf{J}\mathbf{J}^T + e\mathbf{I}$$

The neural network weights are updated via: $\vec{w}_{t+1} = \vec{w}_t - a_t[\hat{\mathbf{H}}(\vec{w}_t)]^{-1}\mathbf{g}(\vec{w}_t)$. The learning rate, a_t is determined through a line search, and the scaling term e acts as a regularization parameter. Proper choice of the scaling term ensures that the matrix

inversion operates on a well conditioned matrix. Thus, the update mechanism of the Levenberg-Marquardt training algorithm uses the following:

$$\vec{w}_{t+1} = \vec{w}_t - a_t[\mathbf{J}\mathbf{J}^T + e\mathbf{I}]^{-1}\mathbf{J}\mathbf{e}_t$$

When the scaling factor is zero, this weight update is the same as Newton's method, also referred to as the linearization approach, with an approximate Hessian. When the scaling factor is large, this weight update operates as a gradient descent method. Newton's method performs better near an error minimum, and gradient descent performs better when far from a minimum. Thus, the compromise of the Levenberg-Marquardt weight updates is to begin with a larger scaling factor to exploit the rapid convergence properties of gradient descent when far from an error minimum. With each improvement in the error term, the scaling factor is gradually reduced to slow the rapid convergence and to begin moving towards Newton's method as the training nears an error minimum.

3.7.1 Neural Network Initialization Methods

One of the considerations for the implementation of ANNs is the choice of initialization method for the weight and bias values. Figure 3.7.1 shows a 3 hidden layer inferential ANN. Referring to this architecture, there are 13 parameters which much be initialized. In this dissertation, when a set of predictor variables had largely differing magnitudes from one variable to the next, the predictor variable matrix was scaled. Scaling was performed such that the scaled variables had a mean of zero, and a standard deviation of 1. If scaling was similarly performed for the response variable, than the estimate from the ANN would be a scaled version of the response rather than the actual estimated value. Thus, post scaling would need to be performed. In addition, the computation of prediction intervals for the output would require a similar scaling. To avoid the required post-scaling operations at the output of an ANN, the response variable was not altered. This is a suitable scenario for an ANN with a linear output neuron, since the linear output neuron has no bounded output.

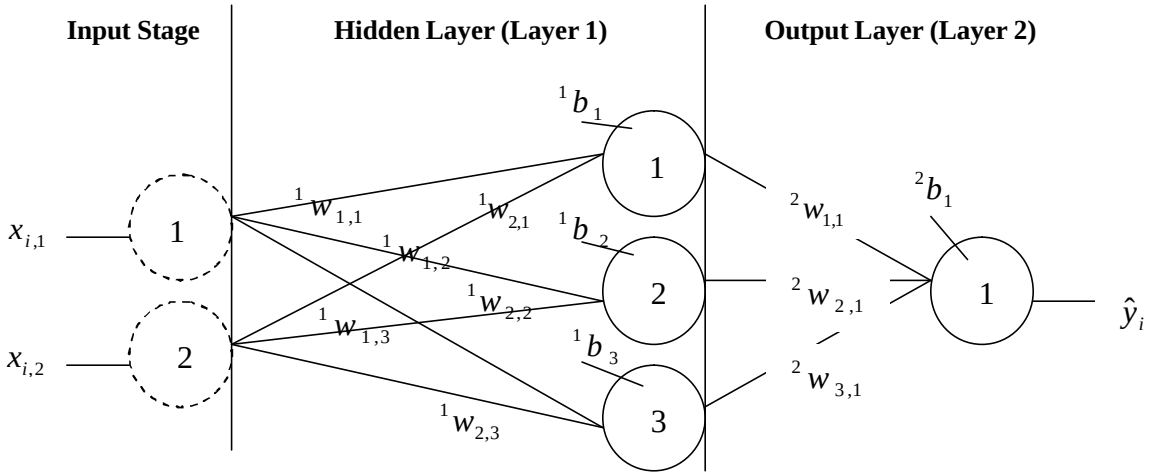


Figure 3.7.1: Schematic of 3 hidden neuron inferential ANN.

Two different methods of initialization were used in producing ANNs for this dissertation:

1. Random Initialization
2. Mean-Standard Initialization

The first method is the standard approach of initializing all weight and bias values to small random numbers. While this approach works fine in most cases, if the response variable magnitude is significantly different than the magnitudes of the predictor variables training will be slow. In addition, the slow training often produces sub-optimal solutions at local minima due to the great distance through the error surface the training algorithm must travel when starting from small random numbers. This argument is based on response variables of large magnitude $\geq$1000, with respect to the zero mean predictors.

Due to the noted difficulties in using an un-scaled response, a second method of ANN initialization was adopted. An assumption made is that random initialization to small random numbers will produce an output with a near zero mean, and a standard deviation near one. While this is never exactly true, it is a gross approximation that can be used to initialize the weight and bias values to at least begin training in a region of appropriate magnitude with respect to the response. Since an approximate mean value and standard deviation of the response is known from the training data the output layer weights and bias value can be initialized to simulate a post-scaling operation. This process is described below, referring to figure 3.7.1.

The Mean-Standard initialization method requires (refer to figure 3.7.1):

1. Random Initialization of all weight and bias values in the hidden layer
2. $^{2}w_{1,1} = {}^{2}w_{2,1} = {}^{2}w_{3,1} = \dfrac{\hat{s}_y}{3}$
3. $^{2}b_1 = \hat{\bar{y}}$

Where $\hat{s}_y$ is an estimate of the standard deviation of the response based on the training data and $\hat{\bar{y}}$ is an estimate of the mean of the response based on the training data.

For a more general inferential single hidden layer ANN architecture, the same procedure applies, except that the 3 in step 2 is replaced by the appropriate number of hidden neurons. This approach to initialization resulted in much faster training procedures, and produced more consistent solutions. This of course is due to the mild bias inserted through the procedure. The Mean-Standard initialization was applied to the training of ANN for the ANN models as well as the NNPLS models, though for the latter, the variables x and y were replaced with their corresponding latent vectors. For both of the nuclear power plant data sets, this initialization method was applied. For the simulated data set, random initialization performed adequately.

3.8 GENERAL NONPARAMETRIC MODEL

While ANN models and NNPLS models can be represented within the framework of nonlinear regression, the third empirical strategy, local polynomial regression, falls under the general framework of nonparametric regression. A nonparametric regression model can be of either fixed design, or random design. In the univariate case, the fixed design consists of $x_i,...,x_n$ which are ordered non-random numbers. For the fixed design case the response variables are assumed to satisfy:

$$Y_i = m(x_i) + v^{1/2}(x_i)e_i, \qquad i = 1,...,n$$

where $e_1,...,e_n$ are independent random variables for which:

$E(e_i) = 0$ and $Var(e_i) = 1$

m is the regression function, where $E(Y_i) = m(x_i)$, and v is the variance function, where $Var(Y_i) = v(x_i)$. If $v(x_i) = s^2$ for all i, the model is homoscedastic otherwise it is heteroscedastic.

The random design regression model arises when bivariate samples of random pairs are observed. i.e. $(X_1,Y_1),...,(X_n,Y_n)$, such that the model can be written as:

$$Y_i = m(X_i) + v^{1/2}(X_i)e_i, \qquad i = 1,...,n$$

Here, the e_i are independent random variables with zero mean and unit variance conditional on $X_1,...,X_n$. The regression and variance functions are thus:

$m(x) = E(Y \mid X = x)$ and $v(x) = Var(Y \mid X = x)$

Most regression estimators are linear in there response, and are thus referred to as linear smoothers [Fan 1992b]. The general form of these estimators is given by:

$$\sum_{i=1}^{n} w_i(x; X_1,...,X_n)Y_i$$

The use of a linear smoother that applies constant approximations to the mean regression function is generally referred to as kernel regression, whereas the use of a linear smoother that applies linear approximations to the mean regression function is generally referred to as local linear regression. There is some ambiguity in the terminology regarding non-parametric techniques since both of these methods are referred to as kernel-type methods. In this work the term kernel regression will be reserved for non-parametric techniques which apply constant approximations to the regression function.

3.8.1 Local Polynomial Regression Methods

Local polynomial regression (LPR) models are often referred to as lazy learning methods. Lazy learning comprises a set of methods in which data processing is deferred until a prediction at a query point needs to be made. These methods are also referred to as memory based methods due to the approach of storing the training data, and recalling relevant training data when a query is made. A good review of lazy learning methods, focusing on locally weighted regression, is presented by Atkeson et. al. [1996]. A training data set is comprised of a set of input vectors, and a corresponding set of output values. A query point is an input vector for which an output is to be determined. The query point input vector may or may not be in the training set. Relevant data is identified by the use of a distance function where maximum relevance occurs when a query point matches a point in the training set, relevance diminishes from this maximum as the

distance between a query point and the training points increases. Nonparametric regression using data in a neighborhood of the present query point is generally referred to as a local model. Local models attempt to fit the data in a region surrounding the query point with a polynomial.

Early work in local fitting, via polynomials, was applied to equally spaced points in time by Macaulay [1931]. Local fitting in the context of regression analysis was introduced by Watson [1964], Stone [1977], and Cleveland [1979]. Cleveland and Devlin [1988] expanded Cleveland's method and evaluated its mathematical properties. Local likelihood procedures were evaluated by Hastie [1986]. The general local modeling approach is LPR. LPR is also called KR when the degree of the fitted polynomials is 0, and is called LL regression when the degree of the fitted polynomials is 1.

The following sections cover the methods of KR, LL, and LPR. The derivation of the specific computations for KR and LL are developed from the general LPR model. Finally, two relevant discussions are presented: the relationship between ANNs and LPR, and a comparison of parametric and nonparametric regression.

3.8.2 Kernel Regression

The process of fitting constants using a locally weighted training criterion is known as kernel regression [Atkeson 1996]. The basis of kernel regression is to estimate a response using a weighted average of points, in a training set, which are local to the query point. The most comprehensive presentation of kernel regression is that of Wand and Jones [1995]. The kernel regression estimator is a linear smoother and thus, has the form: $\sum_{i=1}^{n} w_i(x; X_1,..., X_n)Y_i$

To formally define the general terms of a linear smoother for the kernel regression application, let (X, Y) be a bivariate random variable, and let $(X_1, Y_1),...,(X_n, Y_n)$ be a

random sample, of size n, from the population (X, Y). The kernel regression model is then:

$$Y_i = m(X_i) + v^{1/2}(X_i)e_i, \qquad i = 1,...,n$$

Here, the e_i are independent random variables with zero mean and unit variance conditional on $X_1,..., X_n$. The regression and variance functions are thus:

$$m(x) = E(Y \mid X = x) \qquad \text{and} \qquad v(x) = Var(Y \mid X = x)$$

The weights for the kernel regression application can be defined by:

$$w_i = \frac{K_h(X_i - x)}{\sum_{i=1}^{n} K_h(X_i - x)}$$

Where K_h is the, bandwidth dependent, scaled kernel function: $K_h(u) = \frac{1}{h}K\left(\frac{u}{h}\right)$. A typical kernel is the normal or Gaussian kernel: $K(u) = \frac{1}{\sqrt{2p}}e^{-\frac{u^2}{2}}$. The bandwidth parameter is given by h, and controls the range of influence which may affect the estimate. Considering the Gaussian kernel function it can be seen that the bandwidth parameter is analogous to the more commonly used variance parameter.

The requirements of a weighting or kernel function $K(u)$ are that the maximum value should occur when $u = 0$, and should decrease smoothly as the distance u increases. The kernel function should be continuous, since discontinuities in the kernel function will lead to discontinuities in the predictions. The kernel function should always be positive. In addition, a kernel function should be a bounded and symmetric real function K that integrates to 1: $\int K(u)du = 1$.

The kernel regression estimator can now be formed based on the above definitions and the general form of the linear smoother as:

$$\hat{m}_h(x) = \frac{\sum_{i=1}^{n} [K_h(X_i - x)Y_i]}{\sum_{i=1}^{n} K_h(X_i - x)}$$

This form is referred to as the Nadaraya [1964]-Watson [1964] estimator. Note that the subscript h on the estimate indicates the dependence of the estimator on the kernel bandwidth. The larger the bandwidth, the more data used to determine the estimate at a given point. Figure 3.8.1 shows the Gaussian kernel function for a variety of bandwidths.

In applying the kernel regression estimator, the term training data is often used to describe the set of data $(X_1, Y_1),...,(X_n, Y_n)$. This term is ambiguous in the context of kernel regression, considering that there is no training process involved; however, the term training data is maintained to identify the set of data which is used to represent the model under consideration. Finally, when applying the estimator to a point not included in the training data, the new point is referred to as the query point, often denoted by x_q, though in this work the query point will be identified as x_0. Thus when applying the estimator in practice, it takes the form: $\hat{m}_h(x_0) = \frac{\sum_{i=1}^{n} [K_h(X_i - x_0)Y_i]}{\sum_{i=1}^{n} K_h(X_i - x_0)}$

3.8.3 Multivariate Kernel Regression

The multivariate KR estimator is a direct extension of the univariate case. Consider a *d*-variate random sample $\mathbf{X}_1,..., \mathbf{X}_n$ having density f. The components of $\mathbf{X}_i$ are denoted by $\mathbf{X}_i = (X_{i1},..., X_{id})^T$, and a generic vector $x \in \Re$ will be denoted as $\mathbf{x} = (x_1,..., x_d)^T$.

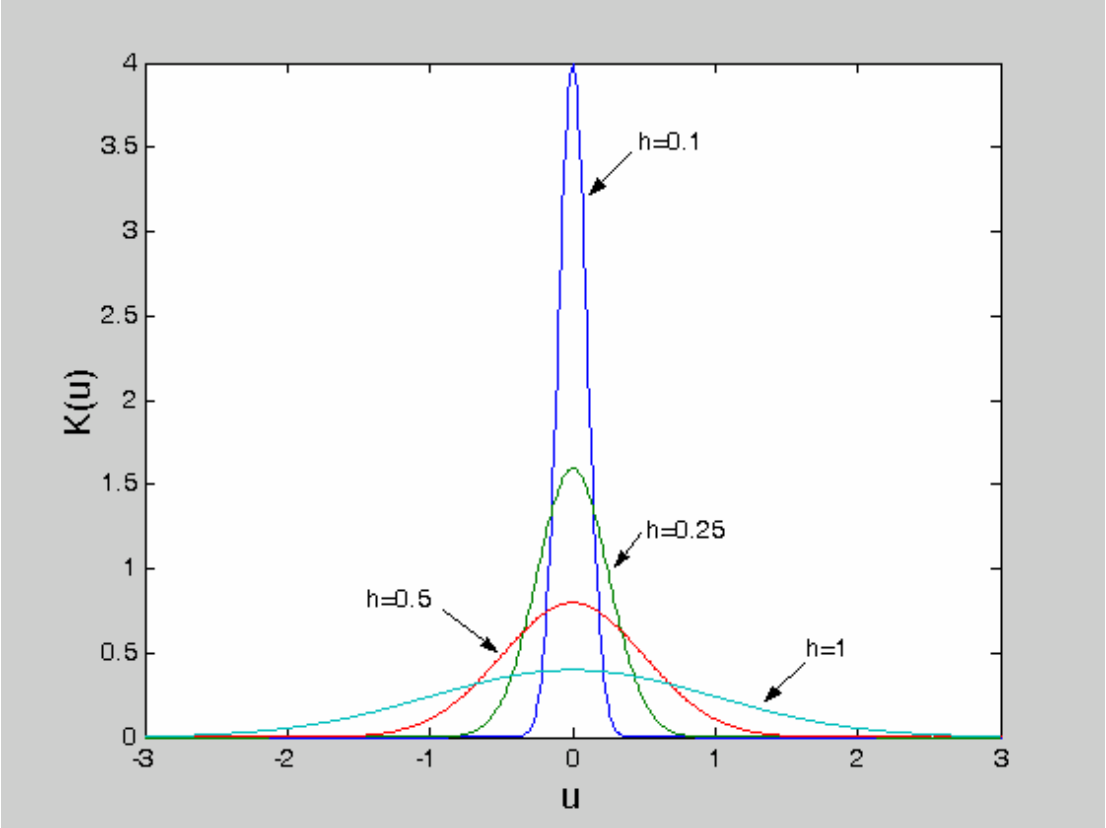


Figure 3.8.1: Gaussian kernel function for various bandwidth parameters.

Defining the vector $\mathbf{u}$ to replace the scalar u for the univariate case results in $\mathbf{u} = [u_1 \quad u_2 \quad \cdots \quad u_d]$. The multivariate Gaussian kernel can be formed from a product of d univariate kernels:

$$K(\mathbf{u}) = \prod_{j=1}^{d} K(u_j) = \left[\frac{1}{\sqrt{2p}} e^{-\frac{u_1^2}{2}}\right] \cdot \left[\frac{1}{\sqrt{2p}} e^{-\frac{u_2^2}{2}}\right] \cdots \left[\frac{1}{\sqrt{2p}} e^{-\frac{u_d^2}{2}}\right]$$

$$K(\mathbf{u}) = \left(\frac{1}{\sqrt{2p}}\right)^d \exp\left[-\frac{1}{2}\mathbf{u}^T\mathbf{u}\right]$$

The bandwidth dependent multivariate kernel is then:

$$K_{\mathbf{H}}(\mathbf{u}) = |\mathbf{H}|^{-1/2} K(\mathbf{H}^{-1/2}\mathbf{u})$$

where $\mathbf{H}$ is a symmetric positive definite $d \times d$ bandwidth matrix.

The multivariate KR estimator is then:

$$\hat{m}_h(\mathbf{x}_0) = \frac{\sum_{i=1}^{n} [K_{\mathbf{H}}(\mathbf{X}_i - \mathbf{x}_0) Y_i]}{\sum_{i=1}^{n} K_{\mathbf{H}}(\mathbf{X}_i - \mathbf{x}_0)}$$

Restricting $\mathbf{H}$ to a diagonal matrix simplifies the computations, and results in the following kernel estimator:

$$K_{\mathbf{h}}(\mathbf{X}_i - \mathbf{x}_0) = \left(\prod_{l=1}^{d} h_l\right)^{-1} \sum_{i=1}^{n} K\left(\frac{X_{i,1} - x_{0,1}}{h_1}, \ldots, \frac{X_{i,d} - x_{0,d}}{h_d}\right)$$

Further restricting to a single value for the bandwidth results in:

$$K_h(\mathbf{X}_i - \mathbf{x}_0) = \frac{1}{h^d} \sum_{i=1}^{n} K\{(\mathbf{X}_i - \mathbf{x}_0)/h\}$$

3.8.4 Local Linear Regression

The local linear regression assumes the same model as described in the kernel regression discussion of the previous section, namely:

$$Y_i = m(X_i) + v^{1/2}(X_i)e_i, \qquad i = 1,...,n$$

The local linear regression curve at a given point x is obtained by applying the standard linear regression technique to a set of data local to the point x. The local regression model, with the local region defined by h, is given by:

$$Y_i = a(x) + b(x)X_i + e \qquad \text{for } X_i \in x \pm h$$

$a(x)$ and $b(x)$ are the parameters of a liner fit in the neighborhood of x, specified by h. The local model parameters are thus dependent on the points, X_i, in the neighborhood of x. The set of observations $(X_1, Y_1),...,(X_n, Y_n)$, are independent and identically distributed.

The standard least squares problem of the linear model is:

$$\sum_{i=1}^{n} [Y_i - a(x) + b(x)X_i]^2$$

where n is the number of points satisfying $X_i \in x \pm h$.

Thus, n is a function of x and h.

A weighting scheme can also be incorporated to allow local points to influence the model in a manner relative to their distance from the given point x. Using the kernel function as described in the previous section: $K\{(X_i - x)/h\}$. The weighted least squares problem is then:

$$\sum_{i=1}^{n} K_h(X_i - x)(Y_i - b_0 - b_1(X_i - x))^2$$

Note that if the weighting method is applied, the neighborhood parameters are inherently defined by the bandwidth.

If $\hat{b}_0(x)$ and $\hat{b}_1(x)$ are found through the minimization of the weighted least squares problem, then the local linear regression estimator is: $\hat{m}(x) = \hat{b}_0$. The explanation for this result is presented in the following section on locally weighted regression. The local linear regression estimator at a point, x, can also be written as:

$$\hat{m}(x) = \sum_{i=1}^{n} \frac{\{\hat{s}_2(x) - \hat{s}_1(x)(X_i - x)\} K_h(X_i - x) Y_i}{\hat{s}_2(x)\hat{s}_0(x) - [\hat{s}_1(x)]^2}$$

$$\hat{s}_r(x) = \sum_{i=1}^{n} (X_i - x)^r K_h(X_i - x)$$

When the bandwidth is small, the local linear estimator interpolates the data points and over-parameterizes the unknown function resulting in noisy estimates [Fan 1996]. When the bandwidth is large the local linear estimator reduces to the standard parametric linear regression estimate, which, with the exception of the cases where the model is correctly specified, under-parameterizes the regression function resulting in a large modeling bias.

This approach can easily be generalized to local models of higher order, as will be done in the next section. Multivariate extensions of the local linear model will also be developed in the next section.

3.8.5 Local Polynomial Regression

A general set of kernel estimators is the class of local polynomial kernel estimators. Local polynomial kernel estimators obtain an estimate for a given query point by fitting a d^{th} degree polynomial using weighted least squares. The training points are weighted based on their distance from the query point using a kernel function centered at the query point. The kernel's influence can be adjusted via the kernel bandwidth.

For a general local polynomial model of order p, if one chooses $h = \infty$, the resultant nonparametric model reduces to the equivalent parametric model of order p. Smaller bandwidths result in lower bias for the local estimator, but since there are only a few

points in a neighborhood defined by a small bandwidth the variance of the parameters of the local model ($\hat{a}(x)$ and $\hat{b}(x)$) will be large. Conversely, large bandwidths result in lower variance but higher modeling bias. This alludes to the bias-variance tradeoff that is paramount in the selection of the optimal bandwidth for local modeling applications. While a variety of theoretical approaches for optimal bandwidth selection are available, they are not covered in this work. The reason for this is that these theoretical approaches depend on unknown quantities. In this work, an empirical approach to bandwidth optimization is used which attempts to minimize the overall contributions of bias and variance based on data-driven estimates of these quantities.

Note that local averaging smoothers, i.e. kernel smoothers and local polynomial smoothers, can be highly influenced by *outliers* in the response variable. Cleveland [1979] proposed a method of robust locally weighted regression to reduce these influences. The basic approach performs a LPR, residuals are then computed and each residual is assigned a weight (large residuals receive lesser weighting and vice versa). LPR is then repeated using weights defined by the product of the weights from the initial LPR fit and the weights assigned to the residuals. Thus, large residuals are *downweighted*. This procedure can be iteratively repeated to enhance its effects. An example of this procedure was provided by Fan and Gijbels [1996].

Thorough studies of local polynomial regression have been presented by Stone [1977 1980 1982], Cleveland [1979], Fan [1992a 1993], Fan and Gijbels [1992b], and Ruppert and Wand [1994]. Their bases all stem from a Taylor series expansion of the local regression function. The procedure that follows is attributed to Fan and Gijbels [1996].

For the case of a univariate nonparametric regression model, consider the bivariate data: $(X_1, Y_1),...,(X_n, Y_n)$, which form an i.i.d. sample from the entire population (X, Y). The local polynomial regression procedure attempts to estimate the regression function $m(x_0) = E(Y \mid X = x_0)$, and its derivatives $m'(x_0), m''(x_0),..., m^{(p)}(x_0)$. The point x_0 is

often referred to as the query point, and the set of bivariate data is often referred to as the training data. The data-generating model is:

$Y = m(X) + n^{1/2}(X)e$, where $E(e) = 0$, $Var(e) = 1$, X and e are independent, $v(x) = Var(Y \mid X = x)$, and $m(x) = E(Y \mid X = x)$.

If p is the order of the local polynomial, then suppose that the $(p+1)^{\text{th}}$ derivative of $m(x)$ at x_0 exists. A Taylor expansion for x in a neighborhood of x_0 is given by:

$$m(x) \approx m(x_0) + m'(x_0)(x - x_0) + \frac{m''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{m^{(p)}(x_0)}{p!}(x - x_0)^p$$

The weighted least squares problem which should be minimized is:

$$\sum_{i=1}^{n} \{Y_i - \sum_{j=0}^{p} b_j (X_i - x_0)^j\}^2 K_h(X_i - x_0)$$

$\hat{b}_j$ ($j = 0,..., p$) is the solution to the above least squares problem. If $\hat{b}_j = \left[\hat{b}_0,..., \hat{b}_p\right]$ is the minimizer of the weighted least squares problem, than an estimator of the regression function is: $\hat{m}(x) = \hat{b}_0$, and an estimator of its first derivative is: $\hat{m}'(x) = \hat{b}_1$. In general the set of estimators of the function and its p derivatives are given by: $\hat{m}^k(x) = k!\hat{b}_k$, where $k = 0,..., p$. It is clear from this discussion, that to obtain an estimate for the regression function at a point x, one must first compute the minimizer of the weighted least squares problem, $\hat{b}_j = \left[\hat{b}_0,..., \hat{b}_p\right]$. From these p parameters, the only one required for the estimation of the regression function at x is $\hat{b}_0$, regardless of the degree of the local polynomial. The remaining parameter estimates, $\hat{b}_1,..., \hat{b}_p$ provide estimates of the derivatives of the regression function via: $\hat{m}^k(x) = k!\hat{b}_k$. Thus, local polynomial regression provides an efficient way to estimate the regression function and its derivatives.

Alternatively, in matrix notation the weighted least squares problem can be written as:

$$(\mathbf{y} - \mathbf{X}_x b)^T \mathbf{W}(\mathbf{y} - \mathbf{X}_x b)$$

Where $b = (b_0,..., b_p)^T$, and $\hat{b} = (\hat{b}_0,..., \hat{b}_p)^T$ is the solution to the above least squares problem obtained from:

$$\hat{b} = (\mathbf{X}_x^T \mathbf{W}\mathbf{X})^{-1} \mathbf{X}_x^T \mathbf{W}\mathbf{y}$$

Defining the terms for the matrix version of the weighted least squares problem:

$$\mathbf{X}_x = \begin{bmatrix} 1 & (X_1 - x_0) & \cdot & \cdot & \cdot & (X_1 - x_0)^p \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ 1 & (X_n - x_0) & \cdot & \cdot & \cdot & (X_n - x_0)^p \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} Y_1 \\ \cdot \\ \cdot \\ \cdot \\ Y_n \end{bmatrix}$$

$$\mathbf{W} = diag\{K_h(X_i - x_0)\}$$

The local polynomial regression estimator at point $\mathbf{x}_0$ is thus:

$$\hat{m}(\mathbf{x}_0) = \hat{b}_0(\mathbf{x}_0)$$

The determination of the order of the polynomial to use for a given model is guided by the trend of a general increase in variance as the order of the local polynomial is increased; however, when moving from an even order polynomial to an odd order polynomial there is no associated variance increase. Conversely, when moving from an odd order polynomial to the consecutive even ordered polynomial there is an associated variance increase. For these reasons, even-order fits are not recommended and it is suggested that the lowest odd-order should be used for most applications, i.e. $p = k + 1$ and occasionally $p = k + 3$. For a detailed explanation of this effect, the reader is referred to Fan and Gijbels [1996]. The main points can be summarized by noting that an odd ordered fit $2q + 1$, introduces an extra parameter in comparison with its closest even

ordered fit, $2q$. This extra parameter allows for a reduction in bias, while not increasing the variance. Further elaboration on this is given in section 4.8.

It is apparent that the general model for local polynomial regression can be reduced to a local linear model by specifying the order of the local polynomial as $p=1$. Similarly, kernel regression results if the order is specified as $p=0$. The next 2 sections provide the results that LPR with $p=0$ reduces to the Nadaraya-Watson kernel estimator, and with $p=1$ reduces to the LL regression estimator.

3.8.6 From Local Polynomial Regression to Kernel Regression

This procedure is detailed for the univariate case; however, direct extensions to the multivariate case are straightforward. From the matrix definitions of the local polynomial regression model, using $p=0$, the following are obtained:

$$\mathbf{X}_x = \begin{bmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 1_n \end{bmatrix} \qquad \mathbf{W}_x = \begin{bmatrix} w_1 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & w_2 & 0 & \cdot & \cdot & \cdot \\ \cdot & 0 & \cdot & 0 & \cdot & \cdot \\ \cdot & \cdot & 0 & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & 0 & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & 0 & w_n \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} Y_1 \\ \cdot \\ \cdot \\ \cdot \\ Y_n \end{bmatrix} \qquad \mathbf{e}_1 = 1$$

Note that the above definition of $\mathbf{e}_1$ applies only for kernel regression.

The solution to the regression equation occurs for:

$$\hat{m} = \mathbf{e}_1^{\mathbf{T}} (\mathbf{X}_x^{\mathbf{T}} \mathbf{W}_x \mathbf{X})^{-1} \mathbf{X}_x^{\mathbf{T}} \mathbf{W}_x \mathbf{y}$$

$$\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x = \begin{bmatrix} 1 & 1 & \cdot & \cdot & \cdot & 1_n \end{bmatrix} \cdot \begin{bmatrix} w_1 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & w_2 & 0 & \cdot & \cdot & \cdot \\ \cdot & 0 & \cdot & 0 & \cdot & \cdot \\ \cdot & \cdot & 0 & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & 0 & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & 0 & w_n \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & \cdot & \cdot & \cdot & w_n \end{bmatrix}$$

$$\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X} = \begin{bmatrix} w_1 & w_2 & \cdot & \cdot & \cdot & w_n \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 1_n \end{bmatrix} = \sum_{i=1}^{n} w_i$$

$$\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{y} = \begin{bmatrix} w_1 & w_2 & \cdot & \cdot & \cdot & w_n \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ \cdot \\ \cdot \\ \cdot \\ Y_n \end{bmatrix} = \sum_{i=1}^{n} w_i Y_i$$

$$\hat{m} = \mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X})^{-1}\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{y} = \frac{\sum_{i=1}^{n} w_i Y_i}{\sum_{i=1}^{n} w_i}$$

Using the weight function $w_i = K_h(X_i - x_0)$, it can be seen that the Nadaraya-Watson estimator results:

$$\hat{m}_h(x_0) = \frac{\sum_{i=1}^{n} [K_h(X_i - x_0) Y_i]}{\sum_{i=1}^{n} K_h(X_i - x_0)}$$

3.8.7 From Local Polynomial Regression to Local Linear Regression

This procedure is detailed for the univariate case; however, direct extensions to the multivariate case are straightforward. From the matrix definitions of the local polynomial regression model, using $p=1$, the following are obtained:

The vector of observations surrounding the query point x_0 is given by:

$$\mathbf{X}_x=\begin{bmatrix}1 & (X_1-x_0)\\ 1 & (X_2-x_0)\\ \cdot & \cdot\\ \cdot & \cdot\\ \cdot & \cdot\\ 1 & (X_n-x_0)\end{bmatrix} \quad \mathbf{W}_x=\begin{bmatrix}w_1 & 0 & \cdot & \cdot & \cdot & 0\\ 0 & w_2 & 0 & \cdot & \cdot & \cdot\\ \cdot & 0 & \cdot & 0 & \cdot & \cdot\\ \cdot & \cdot & 0 & \cdot & 0 & \cdot\\ \cdot & \cdot & \cdot & 0 & \cdot & 0\\ 0 & \cdot & \cdot & \cdot & 0 & w_n\end{bmatrix} \quad \mathbf{y}=\begin{bmatrix}Y_1\\ \cdot\\ \cdot\\ \cdot\\ Y_n\end{bmatrix} \quad \mathbf{e}_1=\begin{bmatrix}1\\ 0\end{bmatrix}$$

Note that the above definition of $\mathbf{e}_1$ applies only for univariate local linear regression, for multivariate local linear regression it is a $(d+1)\times 1$ vector of zeros having one in the first entry.

The solution to the regression equation occurs for:

$$\hat{m}=\mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X})^{-1}\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{y}$$

$$\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x=\begin{bmatrix}1 & (X_1-x_0)\\ 1 & (X_2-x_0)\\ \cdot & \cdot\\ \cdot & \cdot\\ \cdot & \cdot\\ 1 & (X_n-x_0)\end{bmatrix}^{\mathbf{T}}\cdot\begin{bmatrix}w_1 & 0 & \cdot & \cdot & \cdot & 0\\ 0 & w_2 & 0 & \cdot & \cdot & \cdot\\ \cdot & 0 & \cdot & 0 & \cdot & \cdot\\ \cdot & \cdot & 0 & \cdot & 0 & \cdot\\ \cdot & \cdot & \cdot & 0 & \cdot & 0\\ 0 & \cdot & \cdot & \cdot & 0 & w_n\end{bmatrix}$$

$$\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x=\begin{bmatrix}w_1 & w_2 & \cdot & \cdot & \cdot & w_n\\ (X_1-x_0)w_1 & (X_2-x_0)w_2 & \cdot & \cdot & \cdot & (X_n-x_0)w_n\end{bmatrix}$$

$$\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x = \begin{bmatrix} w_1 & w_2 & \cdot & \cdot & \cdot & w_n \\ (X_1 - x_0)w_1 & (X_2 - x_0)w_2 & \cdot & \cdot & \cdot & (X_n - x_0)w_n \end{bmatrix} \cdot \begin{bmatrix} 1 & (X_1 - x_0) \\ 1 & (X_2 - x_0) \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & (X_n - x_0) \end{bmatrix}$$

$$\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x = \begin{bmatrix} \sum_{i=1}^{n} w_i & \sum_{i=1}^{n}\left[w_i(X_i - x_0)\right] \\ \sum_{i=1}^{n}\left[w_i(X_i - x_0)\right] & \sum_{i=1}^{n}\left[w_i(X_i - x_0)^2\right] \end{bmatrix}$$

Recall the notation from the local linear section:

$$\hat{s}_r(x) = \sum_{i=1}^{n}(X_i - x)^r K_h(X_i - x) = \sum_{i=1}^{n}\left[w_i(X_i - x)^r\right]$$

The last matrix product can now be rewritten as:

$$\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x = \begin{bmatrix} \hat{s}_0 & \hat{s}_1 \\ \hat{s}_1 & \hat{s}_2 \end{bmatrix}, \text{ and thus: } \left(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x\right)^{-1} = \begin{bmatrix} \dfrac{\hat{s}_2}{\hat{s}_0\hat{s}_2 - \hat{s}_1^2} & \dfrac{\hat{s}_1}{\hat{s}_1^2 - \hat{s}_0\hat{s}_2} \\ \dfrac{\hat{s}_1}{\hat{s}_1^2 - \hat{s}_0\hat{s}_2} & \dfrac{\hat{s}_0}{\hat{s}_0\hat{s}_2 - \hat{s}_1^2} \end{bmatrix}$$

$$\mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X})^{-1} = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \dfrac{\hat{s}_2}{\hat{s}_0\hat{s}_2 - \hat{s}_1^2} & \dfrac{\hat{s}_1}{\hat{s}_1^2 - \hat{s}_0\hat{s}_2} \\ \dfrac{\hat{s}_1}{\hat{s}_1^2 - \hat{s}_0\hat{s}_2} & \dfrac{\hat{s}_0}{\hat{s}_0\hat{s}_2 - \hat{s}_1^2} \end{bmatrix} = \begin{bmatrix} \dfrac{\hat{s}_2}{\hat{s}_0\hat{s}_2 - \hat{s}_1^2} & \dfrac{\hat{s}_1}{\hat{s}_1^2 - \hat{s}_0\hat{s}_2} \end{bmatrix}$$

$$\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{y} = \begin{bmatrix} w_1 & w_2 & \cdot & \cdot & \cdot & w_n \\ (X_1 - x_0)w_1 & (X_2 - x_0)w_2 & \cdot & \cdot & \cdot & (X_n - x_0)w_n \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ \cdot \\ \cdot \\ \cdot \\ Y_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} w_i Y_i \\ \sum_{i=1}^{n}\left[w_i(X_i - x_0)Y_i\right] \end{bmatrix}$$

$$\hat{m} = \mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X})^{-1}\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{y} = \begin{bmatrix} \dfrac{\hat{s}_2}{\hat{s}_0\hat{s}_2 - \hat{s}_1^2} & \dfrac{\hat{s}_1}{\hat{s}_1^2 - \hat{s}_0\hat{s}_2} \end{bmatrix} \cdot \begin{bmatrix} \sum_{i=1}^{n} w_i Y_i \\ \sum_{i=1}^{n}\left[w_i(X_i - x_0)Y_i\right] \end{bmatrix}$$

$$\hat{m}(x_0) = \frac{\hat{s}_2 \sum_{i=1}^{n} w_i Y_i}{\hat{s}_0 \hat{s}_2 - \hat{s}_1^2} + \frac{\hat{s}_1 \sum_{i=1}^{n} [w_i (X_i - x_0) Y_i]}{\hat{s}_1^2 - \hat{s}_0 \hat{s}_2} = \frac{\hat{s}_2 \sum_{i=1}^{n} w_i Y_i - \hat{s}_1 \sum_{i=1}^{n} [w_i (X_i - x_0) Y_i]}{\hat{s}_0 \hat{s}_2 - \hat{s}_1^2}$$

$$\hat{m}(x_0) = \sum_{i=1}^{n} \frac{\hat{s}_2 w_i Y_i - \hat{s}_1 [w_i (X_i - x_0) Y_i]}{\hat{s}_0 \hat{s}_2 - \hat{s}_1^2}$$

Substituting in the weight function $w_i = K_h(X_i - x_0)$ results in the form provided in the previous section on local linear regression.

$$\hat{m}(x_0) = \sum_{i=1}^{n} \frac{\{\hat{s}_2(x) - \hat{s}_1(x)(X_i - x)\} K_h(X_i - x) Y_i}{\hat{s}_2(x)\hat{s}_0(x) - [\hat{s}_1(x)]^2}$$

3.8.8 The relationship between Neural Networks and Local Polynomial Regression

The relationship between the associative memory of neural networks and the memory based estimation of lazy learning techniques is easily identified. The discussion below explains some of these relationships, and is provided so that the similarities between the different empirical methodologies can be illustrated.

Kernel regression can be analogously represented as a generalized regression neural network (GRNN) [Specht 1991, Schiøler 1992]. The GRNN is a special extension of the radial basis function network (RBFN) [Park 1991, 1993], and is closely related to Specht's probabilistic neural network (PNN) [1990]. RBF networks were originally proposed by Broomhead and Lowe [1990]. The main difference between GRNNs and RBFs is that the GRNN output layer performs a weighted average while the RBF performs a weighted sum [Heimes 1998]. In addition, GRNN are non-parametric models, whereas RBF are parametric or semi-parametric models. The basic GRNN has similarities with the methods of Moody and Darken [1989], RBFs [Park 1991, 1993], the cerebellar model articulation controller [Kolcz 1999], and nonparametric kernel based techniques stemming from the work of Nadaraya [1964] and Watson [1964]. Procedures for a neural network-type architecture with a memory of training vectors which computed

estimates based on a neighborhood of points local to the query point were first presented by Steinbuch [1963] and Taylor [1959, 1960]. The original schemes were based on a winner takes all approach where the winning node was determi ned based on a distance metric between the query point and the stored training vectors. This work was extended by Atkeson [1995] to incorporate the use of local polynomials.

Almost all activation functions can be classified as radial basis functions. A radial basis function is of the form [Scarselli 1998]:

$$g(x,a,b) = k\left(\frac{x-a}{b}\right)$$

a is the center of the radial basis function

b is the smoothing factor of the radial basis function

k is assumed to be integrable on R^d and $\int_{R^d} k(x)dx \neq 0$

A common radial basis function is the Gaussian: $g(x,a,b) = \exp[-\|x-a\|^2 / b]$.

From this it can be seen that the radial basis activation functions are similar to the kernel functions of kernel regression [Sarle 1994]. If the hidden layer outputs of a RBFN are normalized to sum to 1, each observation vector is taken as an RBF center, and the weights are taken to be the target values, the outputs are simply weighted averages of the target values and the network is identical to the Nadaraya-Watson kernel regression estimator [Sarle 1994]. This approach was carried out by Heimes and van Heuveln [1998] in their presentation of the *new* Normalized RBF (NRBF). Their derivations are equivalent to the Nadaraya-Watson kernel estimator under two assumptions: assuming the GRNN output weights are taken to be the target values, and a hidden unit is centered for each available target vector. Since kernel regression is based on established statistical principles and converges with an increasing number of samples asymptotically to the optimal regression surface, the same claim can be made for a GRNN [Tomandl 2001]. All of the GRNN models are consistent in their use of spherical kernel functions. Training of GRNNs usually consists of optimizing the bandwidth parameters based on the MSE of the training vectors. Additional work using GRNNs has been reported by

Kolcz and Allinson [1996] under the alternate architecture of an N-tuple regression network, or N-tuple neural network (NTNN). They site an increased capacity for training vector storage and faster computation.

3.9 PARAMETRIC VS NONPARAMETRIC

A parametric model can be specified by a finite number of parameters. While it is easy to see that an OLS, PCR, or PLS model is a parametric model, it is not as clear for models such as NNPLS. Obviously, a small ANN can be considered to be parametric; however, they are often referred to as being nonparametric, or semi-parametric. During this work, the only *true* nonparametric model considered were the local polynomial regression models; though, even these models can approach a parametric model. Cases where a local polynomial regression in fact becomes a parametric model are for extremely large bandwidth parameters, or for an extremely small set of data for model representation. Nonparametric modeling theories present some advantages for the present applications; however, one must be sure to provide the necessary freedom to the nonparametric model so that it in fact is nonparametric. Keeping these thoughts in mind, the key distinguishing points in the parametric vs. nonparametric discussion are presented below.

In some cases parametric models may be too rigid due to the imposed limit on the number of parameters. For example a linear model requiring 2 parameters, intercept and slope, would be too rigid for modeling a quadratic function. By eliminating the restriction that the model belongs to a parametric family, this rigidity condition can be overcome. A tacit assumption of the parametric approach is that the curve can be represented in terms of the parametric model or that, at least, it is believed that the approximation bias of the best parametric fit is negligible [Härdle 1990]. Nonparametric models do not impose the restrictions of a parametric model, and have a greater tendency to let the data define the model. The term nonparametric thus refers to the flexible functional form of the regression curve. When the data are in fact linear, a nonparametric

model will suggest a simple parametric model, such that parametric and nonparamteric models are not mutually exclusive.

While parametric models are susceptible to large model bias under conditions of model misspecification, nonparametric models may result in more variable estimates, especially for the case of a small sample size. For parametric models, the error in prediction typically decreases in proportion to $n^{-1/2}$ [Sarle 1994], where n is the sample size. For kernel regression estimators, the error in prediction typically decreases in proportion to $n^{-p/(2p+d)}$, where p is the number of derivatives of the regression function, and d is the number of inputs [Härdle 1990]. Thus, when applying nonparametric models one must be cautious to ensure appropriate numbers of samples since nonparametric methods tend to require larger sample sizes than parametric models.

The boundary between parametric and nonparametric models is defined by the bandwidth of the nonparametric model. If the bandwidth is maximized, $h = \infty$, the two models are the same. In parametric modeling different families of parametric models are often used to determine the best result. The parametric modeling approach inherently assumes that all of the data are useful for the estimation of a point at any location in the variable space. In addition parametric modeling assumes that a single parametric model describes the data over their entire range. Nonparametric models relax some of the restrictions of standard parametric modeling. The common approach of choosing different bandwidths for a local model and computing estimates of the variance and bias of the model for each bandwidth value sheds light on two interesting points. The first is that most parametric models applied to signal validation tasks are of low order. Along these lines, for a low order local polynomial estimator with a large bandwidth the bias estimate of the model will tend towards unreasonably large values. This indicates that the equivalent parametric model (of the same order) is strongly biased. Thus, the parametric model for this case would be the incorrect choice. If however, an increase in bias was not exhibited as the bandwidth was increased then one could propose that the parametric model was a suitable choice for the given application. The nonparametric approach combines the

flexibility of localization with the ability to act globally, all with the adjustment of the bandwidth. The other point is that in complex processes there are often different relationships between system variables at different points of operation throughout the variable's ranges. The benefits are clear for this purpose since the appropriate bandwidth choice would ensure that the estimator was dependent on the appropriate data. An additional flexibility of the local modeling approach is the ease with which different ordered polynomial models can be combined when necessary.

4.0 ANALYSIS OF PREDICTION INTERVALS

Before stepping into the details of the construction of prediction intervals, a discussion is presented regarding the definition of a prediction interval, and the relationship between confidence intervals and prediction intervals. The sources of uncertainty that contribute to the prediction intervals are presented as well. The focus of this chapter is to cover the development of prediction interval estimation methods for all of the empirical models discussed in the previous chapter. Beginning with a treatment of prediction intervals for the OLS, PCR, and PLS techniques. There are two components of a prediction interval: variance and bias. These two components are often discussed and quantified separately. The prediction interval estimation methods for ANNs and NNPLS result directly from the delta method (linearization method) of nonlinear regression. Thus, before proceeding to the prediction intervals for ANNs and NNPLS a general procedure is provided for nonlinear regression models. The details of the required mathematical equations are presented for ANNs and NNPLS in their appropriate sections. The treatment of LPR concludes the development of analytical approaches to prediction interval estimation. As will be discussed, there are some sources of error that elude the analytical approach to prediction interval estimation; thus, a bootstrap approach based on Monte Carlo resampling is presented. The basic idea behind the bootstrap technique is that repeated applications of the same empirical model to different samples from the population of the data will result in a distribution of predicted values from which a prediction interval can be computed directly.

4.1 PREDICTION INTERVALS DEFINED

To provide a measure of the confidence in a model's estimations, there are two components to consider. Assume a set of input values $\mathbf{x}$, and a corresponding set of desired responses or targets $t(\mathbf{x})$. The relationship is: $t(\mathbf{x}) = f(\mathbf{x}) + e(\mathbf{x})$. The first component is the accuracy of the estimate of the true relationship of the data, i.e. the distribution of the quantity $f(\mathbf{x}) - \hat{f}(\mathbf{x})$. This component is quantified as a confidence

interval. The second component is concerned with the estimate of confidence in the prediction of the targets (desired response values) themselves, i.e. the distribution of the quantity $t(\mathbf{x}) - \hat{f}(\mathbf{x})$. This component is quantified as a prediction interval. Viewing the direct result: $t(\mathbf{x}) - \hat{f}(\mathbf{x}) = [f(\mathbf{x}) - \hat{f}(\mathbf{x})] + e(\mathbf{x})$, it can be seen that confidence intervals are enclosed within the prediction intervals [Carney 1999]. Prediction intervals are of more practical use because they provide the accuracy with which we can predict the desired response, not just the accuracy of the model itself.

The interval for a new measurement will be wider than the confidence interval for the value of the regression function. These intervals are called prediction intervals rather than confidence intervals because the latter are for parameters, and a new measurement is a random variable, not a parameter. Confidence intervals are more narrow than the corresponding prediction intervals, since the prediction intervals must also include variation due to noise in the response, (i.e. $e(\mathbf{x})$) while the confidence interval does not.

4.2 SOURCES OF UNCERTAINTY

There are several sources of uncertainty associated with empirical models and there estimations. In this section we will examine the selection of training data, model misspecification, and noise in both the predictor variables and the response variable.

The selection of a training set is prone to sampling variation because there is variability in the random sampling from the entire population of data. This variability will be consistent in signal validation applications because the data sets used to produce predictive models are required to contain a representative set of data from the entire population, where the entire population would contain observations from all possible plant operational conditions. In addition, for a given observation there is a random fluctuation in the value of the response. Thus, a given set of training data is only one of a possibly infinite set of choices. Since each possible training set will produce a different model, there is a distribution of predictions for a given observation. The bootstrap

approach inherently incorporates this distribution into its standard error of prediction estimates, while the analytical delta method does not. The delta method approach is thus improved when the standard error calculation is based on the predictions of more than one model. The issues relative to fluctuations in the response for a given observation are reduced as the training data set size increases, i.e. more training data lowers the uncertainty of the estimates, assuming the data are representative of the process or function being modeled.

Model misspecification occurs when a given model is incorrect, and a bias is introduced due to the improper model, e.g. fitting non-linear data to a linear model will result in a biased model. Model misspecification may occur for the ANN models and the NNPLS models if their complexity does not meet that of the data to be modeled, though given the proper number of free parameters both techniques are proven to perform adequately, i.e. with minimal bias. While misspecification cannot technically occur for a local polynomial model, due to the absence of the specification of a model, there may be a bias due to the selection of the bandwidth parameter, as this controls the complexity of the local regression model. Small bandwidth models may be under-regularized and overfit the data causing an increased variance. The analog to this for a neural network model is too many free parameters. Both cases present situations of overly complex models for the given task. On the other hand, uncertainty is also influenced by a level of complexity that is too low for the given task. For the case of neural networks, this occurs for network architectures with too few free parameters to adequately perform the desired mapping, while for nonparametric models this occurs for large bandwidth values resulting in over-smoothed solutions. Proper selection of the model architecture optimally matches the complexity of the model to that of the data and correspondingly minimizes the model uncertainty. Models of complexity different from the optimal level for the given task will have an increased uncertainty for both cases of lower than ideal complexity (increased bias) and higher than ideal complexity (increased variance). Model misspecification may also occur if the underlying assumptions of a model are not met.

Aside from the misspecification due to architecture or bandwidth selection, 2 other types of misspecification can occur. The first is due to an incorrect set of predictor variables. Either the predictor variable set may not contain the necessary information to accurately model the desired response (resulting in a model bias) or the predictor variable set may contain variables that are unrelated to the desired response (resulting in an increased solution variance). Both of these conditions result in model misspecification. The second type of misspecification is due to an inadequate distribution of training data. If the training data do not provide an adequate representation of the underlying relationships between the predictor variables and the response variable the model is also considered to be misspecified, though it is not necessarily the model in this case but the data.

Noise in both the input and output data is a consistent source of uncertainty for all empirical models applied to *real*, non-simulated data. All of the analytical approaches to prediction interval estimation presented herein consider only the noise in the dependent, or response, variable. Alternate theories based on the error-in-variables model are available for including the noise in the predictor variables in developing prediction intervals; however, they require knowledge of the noise level present, which is generally unknown. Due to the large scope of this work, covering 3 fundamentally different empirical modeling paradigms, the decision was made to maintain the assumption of noise-free predictor variables in the basic derivations and to incorporate modifications that would account for the presence of noise in the predictors. These modifications are summarized in section 4.12, and discussed in the appropriate sections of this chapter. In addition, the use of the bootstrap approach to prediction interval estimation includes the influence of the predictor variable noise to a limited extent. Thus, comparisons to the computed intervals with respect to the bootstrap intervals can be drawn to verify that noise in the predictor variables is sufficiently accounted for via the implemented modifications. If the modifications properly account for the predictor variable noise than the prediction intervals will provide the expected coverage. Noise in the predictors will be discussed in the recommendations for future work. In addition, the predictor variable noise is known to be helpful in neural network training. The presence of noise in the

predictor variables provides improved generalization for neural network models, by reducing the possibility of overfitting, assuming cross-validation training is used.

Additional sources of uncertainty with respect to the neural network parameters are the presence of local minima in the error surface, and early termination of training at a sub-optimal solution. For the NNPLS models there is also uncertainty regarding the number of selected latent variables.

4.3 EVALUATION OF THE VARIANCE OF THE REGRESSION COEFFICIENTS AND THE PREDICTION INTERVALS FOR ORDINARY LEAST SQUARES (OLS)

This section draws on the basis for OLS that was provided in section 3.1. The stability of the OLS solution can be related to the variances of the regression coefficients, $\mathbf{b}$, which are given by: $\text{var}(\mathbf{b}) = s_e^2 (\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}$

The standard error of the estimates of the regression coefficients is given by:

$$std \quad error_\mathbf{b} = s_e \sqrt{(\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}}$$

where: $s_e^2 = \dfrac{\sum_{i=1}^{n} e_i^2}{n-p}$ can be used to approximate the variance of the error.

The $100(1-a)\%$ confidence intervals for the regression coefficients are thus:

$$\mathbf{b} \pm t_{a/2,n-p} s_e \sqrt{(\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}}$$

In viewing the required computations of the OLS solution, and the variance of the regression coefficients, it is obvious that collinear data will result in unstable and highly variable estimates for the regression coefficients. The variance of the coefficients depends on the estimate of error variance s_e, the number of data points, n, and the collinearity of the predictor variable matrix. Although there are other sources of

uncertainty which may be present, when the OLS model assumptions are met these other sources do not exist. Large magnitude regression coefficients may lead to models which exhibit a greater variance at the output than that which was present at the input. This is an obvious result, since excessively large magnitude regression coefficients will force large changes at the output when only slight changes are presented at the input.

In addition to constructing confidence intervals on the regression coefficients, the estimates of the response variable can also be bounded by confidence intervals. There are two different approaches to constructing these intervals. The first approach provides a confidence interval on the mean of the response at a given observation vector: $\mathbf{x} = \mathbf{x}_0 = \begin{bmatrix} 1 & x_{01} & x_{02} & \cdot & \cdot & \cdot & x_{0p} \end{bmatrix}$. The second approach gives probabilistic bounds on a new observation at fixed conditions of the predictor variables. The first approach provides confidence bounds on $E(y \mid \mathbf{x} = \mathbf{x}_0)$, whereas the second approach provides the prediction interval on a new observation. Prediction and the corresponding standard error and confidence limits apply to observations where interpolation or extrapolation were necessary.

The $100(1-a)\%$ confidence bounds on $E(y \mid \mathbf{x} = \mathbf{x}_0)$, assuming normal errors are given by [Draper 1966]: $\hat{y}(\mathbf{x}_0) \pm t_{a/2,n-p} s \sqrt{\mathbf{x}_0^{\mathbf{T}} (\mathbf{X}^{\mathbf{T}} \mathbf{X})^{-1} \mathbf{x}_0}$

The $100(1-a)\%$ prediction interval on a new observation is given by [Draper 1966]: $\hat{y}(\mathbf{x}_0) \pm t_{a/2,n-p} s \sqrt{1 + \mathbf{x}_0^{\mathbf{T}} (\mathbf{X}^{\mathbf{T}} \mathbf{X})^{-1} \mathbf{x}_0}$

4.4 EXTENDING REGRESSION VECTOR VARIANCE, BIAS CALCULATIONS AND PREDICTION INTERVAL ESTIMATION TO PCR AND PLS

This section details the computations for prediction interval estimation of the base models of OLS, PCR and PLS, to complement the later treatment of the advanced empirical

modeling strategies which are the focus of this work. This brief presentation summarizes the work detailed by Faber and Kowalski [1997].

Estimating the regression by Ordinary Least Squares (OLS):

$\hat{\boldsymbol{ß}}_{OLS} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$ $\mathbf{X}$ is $N \times K$, and $\mathbf{y}$ is $N \times 1$

The covariance of the OLS regression coefficients is given by:

$\mathbf{V}(\hat{\boldsymbol{ß}}_{OLS}) = s^{2}_{\Delta \mathbf{y}}(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}$ $s^{2}_{\Delta y}$ is the variance of the measurement errors

Estimating the regression by Principal Component Regression (PCR):

$$\hat{\boldsymbol{ß}}_{PCR} = \left(\sum_{a=1}^{A} l_a^{-1} \mathbf{v}_a \mathbf{v}_a^{\mathbf{T}} \right) \mathbf{X}^{\mathbf{T}}\mathbf{y}$$

l_a is the a^{th} eigenvalue of $\mathbf{S} = \mathbf{X}^{\mathbf{T}}\mathbf{X}$

$\mathbf{v}_a$ is the a^{th} column of the eigenvector matrix $\mathbf{V}$

$\mathbf{V}$ is the $K \times K$ eigenvector matrix of $\mathbf{S} = \mathbf{X}^{\mathbf{T}}\mathbf{X}$

A is the dimension of the PCA model

The eigenvectors, $\mathbf{v}_a$, are obtained from:

$$\mathbf{X}^{\mathbf{T}}\mathbf{X} = \mathbf{V}\Lambda\mathbf{V}^{\mathbf{T}} = \sum_{a=1}^{P} l_a \mathbf{v}_a \mathbf{v}_a^{\mathbf{T}}$$

Λ is the $K \times K$ eigenvalue matrix, and

P is the rank of $\mathbf{X}$

The covariance of the PCR regression coefficients is given by:

$$\mathbf{V}(\hat{\boldsymbol{ß}}_{PCR}) = s^{2}_{\Delta \mathbf{y}} \left(\sum_{a=1}^{A} l_a^{-1} \mathbf{v}_a \mathbf{v}_a^{\mathbf{T}} \right)$$

Estimating the regression vector by PLS:

$\hat{\boldsymbol{ß}}_{PLS} = \hat{\mathbf{K}}_A(\hat{\mathbf{K}}_A^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{X}\hat{\mathbf{K}}_A)^{-1}\hat{\mathbf{K}}_A^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{y}$

Where the $(K \times A)$ Krylov matrix is given by:

$\hat{\mathbf{K}}_A = [\tilde{\mathbf{s}}, \mathbf{S}\tilde{\mathbf{s}}, ..., \mathbf{S}^{A-1}\tilde{\mathbf{s}}]$ $\tilde{\mathbf{s}} = \mathbf{X}^{\mathbf{T}}\mathbf{y}$.

Within the framework of PLS, this can be written as:

$$\hat{\boldsymbol{ß}}_{PLS} = \left(\sum_{a=1}^{A} \mathbf{r}_a \mathbf{r}_a^{\mathbf{T}} \right) \mathbf{X}^{\mathbf{T}} \mathbf{y}$$

$\mathbf{r}_a$ is the a^{th} column of the $K \times A$ matrix of PLS weights $\mathbf{R}_A$

The vectors $\mathbf{r}_a$ can be obtained from the orthogonal score matrix: $\hat{\mathbf{T}}_A = \mathbf{X}\hat{\mathbf{R}}_A$.

The use of the Krylov matrix is generally avoided due to numerical problems.

Contrary to the expressions derived for the covariance of the regression coefficients for OLS and PCR, there is no is exact equation for PLS. The PLS estimator of the regression vector is not linear, and thus approximate expressions must be employed. Phatak et. al. [1993] linearized the PLS estimator via a Taylor series expansion (truncating after a single term), such that an approximation of the covariance of the regression vector can be obtained. The resultant expression was verified through a Monte Carlo study, and is shown below:

$$\mathbf{V}_1(\hat{\boldsymbol{ß}}_{PLS}) = s^{\,2}_{\Delta \mathbf{y}} \mathbf{J}_{\mathbf{y}} \mathbf{J}_{\mathbf{y}}^{\mathbf{T}}$$

$\mathbf{J}_{\mathbf{y}}$ is the $(k \times n)$ Jacobian matrix of first order derivatives of each component of $\boldsymbol{ß}_{PLS}$ with respect to $\mathbf{y}$. The subscript 1 indicates that it is a first order approximation.

$$\mathbf{J}_{\mathbf{y}} = \left(\mathbf{S}_A^{-} + \sum \mathbf{B}_a \mathbf{S}^{a-1} \right) \mathbf{X}^{\mathbf{T}}$$

$$\mathbf{B}_a = [\mathbf{K}_A \mathbf{q}_a \mathbf{s}^{\mathbf{T}} + \mathbf{s}^{\mathbf{T}} \mathbf{K}_A \mathbf{q}_a \mathbf{I}] \mathbf{G}_{\mathbf{RP}}$$

$\mathbf{q}_a$ is the a^{th} column of $\mathbf{Q}_A^{-1} = (\hat{\mathbf{K}}_A^{\mathbf{T}} \mathbf{S} \hat{\mathbf{K}}_A)^{-1}$, and $\mathbf{G}_{\mathbf{RP}} = \mathbf{I} - \mathbf{R}_A \mathbf{P}_A^{\mathbf{T}}$. Due to the numerical instability of the Krylov matrix, Phatak et al. [1993] have introduced the following substitutions: replace $\mathbf{K}_A$ by the weight matrix $\mathbf{R}_A$, and $\mathbf{q}_a$ by $\mathbf{m}_a$, the a^{th} column of $(\mathbf{M}^{\mathbf{T}})^{-1}$, where $\mathbf{M} = \mathbf{P}_A^{\mathbf{T}} \mathbf{K}_A$.

An alternate approximation proposed by Höskuldsson [1988] is based on the observation that $\mathbf{S}_A^{-}$ is an approximate covariance matrix. Thus, neglecting the second term in the Jacobian equation leads to the approximation:

$$\mathbf{V}_0(\hat{\boldsymbol{ß}}_{PLS}) = s^{\,2}_{\Delta \mathbf{y}} \mathbf{S}_A^{-} = s^{\,2}_{\Delta \mathbf{y}} \sum_{a=1}^{A} \mathbf{r}_a \mathbf{r}_a^{\mathbf{T}}$$

The zero subscript indicates a zero[th] order approximation. The relative size of the term omitted to obtain this approximation depends greatly on the dimensionality of the model. The sum has terms that decrease with increasing dimensionality (e.g. $\mathbf{S}^{a-1}$), as well as terms that increase (e.g. $\mathbf{G}_{\mathbf{RP}}$). Both approaches should be evaluated, to ensure that the simpler approximation is adequate for the given data.

The variance of the prediction, $\hat{y}_u$, can be obtained from:

$$\breve{V}(\hat{y}_u) = \frac{s^2_{\Delta y}}{N} + \mathbf{x}_u^{\mathrm{T}} \breve{V}(\hat{\boldsymbol{ß}}) \mathbf{x}_u$$

N is the number of observations in $\mathbf{X}$

Note that, $\breve{\mathbf{V}}(\hat{\boldsymbol{ß}})$ stands for $\breve{\mathbf{V}}(\hat{\boldsymbol{ß}}_{OLS})$, $\breve{\mathbf{V}}(\hat{\boldsymbol{ß}}_{PCR})$, or $\breve{\mathbf{V}}(\hat{\boldsymbol{ß}}_{PLS})$.

This equation holds for OLS and PCR due to the linear property of the estimators; however, for PLS the following replacement must be made:

$$\frac{s^2_{\Delta \mathbf{y}}}{N} = \left(\frac{1}{N} + 1 \right) s^2_e$$

A prediction interval for the estimate y_u can be constructed as [Faber 1997]:

$$\mathbf{Pr}\left[\left| y_u - \hat{y}_u \right| \le \hat{s}_{\Delta y} t_{v_y, a/2} \sqrt{(V(\hat{y}_u) s^{-2}_{\Delta y}} \right] = 1 - a$$

4.4.1 Summary of prediction interval computations for OLS, PCR, and PLS

The general equation for the variance of an estimate $\hat{y}_u$, given an observation $\mathbf{x}_u$ is:

$$V(\hat{y}_u) = \frac{s^2_{\Delta y}}{N} + \mathbf{x}_u^{\mathrm{T}} \mathbf{V}(\hat{\boldsymbol{ß}}) \mathbf{x}_u$$

This equation holds for OLS and PCR due to the linear property of the estimators; however, for PLS the following replacement must be made:

$$\frac{s^2_{\Delta \mathbf{y}}}{N} = \left(\frac{1}{N} + 1 \right) s^2_e$$

$\breve{\mathbf{V}}(\hat{\boldsymbol{ß}})$ stands for $\mathbf{V}(\hat{\boldsymbol{ß}}_{OLS})$, $\mathbf{V}(\hat{\boldsymbol{ß}}_{PCR})$, or $\mathbf{V}(\hat{\boldsymbol{ß}}_{PLS})$

$$\mathbf{V}(\hat{\boldsymbol{ß}}_{OLS}) = s^{2}_{\Delta \mathbf{y}} (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}$$

$$\mathbf{V}(\hat{\boldsymbol{ß}}_{PCR}) = s^{2}_{\Delta \mathbf{y}} \left(\sum_{a=1}^{A} l_a^{-1} \mathbf{v}_a \mathbf{v}_a^{\mathbf{T}} \right)$$

$$\mathbf{V}_0(\hat{\boldsymbol{ß}}_{PLS}) = s^{2}_{\Delta \mathbf{y}} \mathbf{S}^{-}_{A} = s^{2}_{\Delta \mathbf{y}} \sum_{a=1}^{A} \mathbf{r}_a \mathbf{r}_a^{\mathbf{T}}$$

$$\mathbf{V}_1(\hat{\boldsymbol{ß}}_{PLS}) = s^{2}_{\Delta \mathbf{y}} \mathbf{J}_{\mathbf{y}} \mathbf{J}_{\mathbf{y}}^{\mathbf{T}}$$

The subscripts for the PLS equations refer to the order of approximation for the derivatives of the PLS weights with respect to the regression coefficients.

The general equation for the prediction intervals is:

$$\mathbf{Pr}\left[\left| y_u - \hat{y}_u \right| \le \hat{s}_{\Delta y} t_{v_y, a/2} \sqrt{(V(\hat{y}_u) s^{-2}_{\Delta y}} \right] = 1 - a$$

This can be interpreted as, the probability that the following holds true is $1-a$:

$$\hat{y}_u - \hat{s}_{\Delta y} t_{v_y, a/2} \sqrt{(V(\hat{y}_u) s^{-2}_{\Delta y}} \le y_u \le \hat{y}_u + \hat{s}_{\Delta y} t_{v_y, a/2} \sqrt{(V(\hat{y}_u) s^{-2}_{\Delta y}}$$

A more compact form for an estimate and its corresponding prediction interval is given by:

$$\hat{y}_u \pm \hat{s}_{\Delta y} t_{v_y, a/2} \sqrt{(V(\hat{y}_u) s^{-2}_{\Delta y}}$$

Substituting in the appropriate variance equations, the following prediction intervals can be obtained:

OLS: $\hat{y}_u \pm \hat{s}_{\Delta \mathbf{y}} t_{n_y, a/2} \sqrt{\frac{1}{N} + \mathbf{x}_u^{\mathbf{T}} (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1} \mathbf{x}_u}$

PCR: $\hat{y}_u \pm \hat{s}_{\Delta \mathbf{y}} t_{n_y, a/2} \sqrt{\frac{1}{N} + \mathbf{x}_u^{\mathbf{T}} \left[\sum_{a=1}^{A} l_a^{-1} \mathbf{v}_a \mathbf{v}_a^{\mathbf{T}} \right] \mathbf{x}_u}$

PLS*: $\hat{y}_u \pm \hat{s}_{\Delta \mathbf{y}} t_{n_y, a/2} \sqrt{\frac{1}{N} + 1 + \mathbf{x}_u^{\mathbf{T}} \left[\sum_{a=1}^{A} l_a^{-1} \mathbf{r}_a \mathbf{r}_a^{\mathbf{T}} \right] \mathbf{x}_u}$

* - Based on zero order approximations

4.5 PREDICTION INTERVALS FOR NONLINEAR REGRESSION MODELS

From the discussion on nonlinear regression, we've established the following:

$\mathbf{y}_i = f(\mathbf{x}_i, \vec{q}) + \mathbf{e}_i$

where: $\mathbf{x}_i = [X_{i,1} \quad X_{i,2} \quad \cdots \quad X_{i,p}]$, $i = 1,..., n$

p is the number of predictor variables

$\mathbf{e} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \cdots \quad \mathbf{e}_n]$

The assumptions of normality and independence are $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I} s_e^2)$.

The sum of squares error is then:

$S(\vec{q}) = \sum_{i=1}^{n} \left\{ \mathbf{y}_i - f(\mathbf{x}_i, \vec{q}) \right\}^2$

A value of $\vec{q}$ which minimizes $S(\vec{q})$ is denoted $\hat{\vec{q}}$

A Taylor series expansion (to the first order) of $f(\mathbf{x}_i, \vec{q})$ about $\hat{\vec{q}}$ is given by:

$$f(\mathbf{x}_i; \hat{\vec{q}}) \approx f(\mathbf{x}_i, \vec{q}_0) + \sum_{k=1}^{p} \left\{ \left[\frac{\partial f(\mathbf{x}_i, \vec{q})}{\partial q_k} \right]_{\vec{q} = \vec{q}_0} \left(\hat{q}_k - q_k \right) \right\}$$

Which can be rewritten as:

$f(\mathbf{x}_i; \hat{\vec{q}}) \approx f(\mathbf{x}_i, \vec{q}) + \mathbf{f}_0^T \cdot [\hat{\vec{q}} - \vec{q}]$

where: $$\mathbf{f}_0^{\mathbf{T}} = \left(\frac{\partial f(\mathbf{x}_0; q^*)}{\partial q_1^*}, \frac{\partial f(\mathbf{x}_0; q^*)}{\partial q_2^*}, ..., \frac{\partial f(\mathbf{x}_0; q^*)}{\partial q_p^*} \right)$$

The subscript 0 denotes an observation other than those used for least squares estimation of $\vec{q}$, i.e. an independent observation for which an estimate is desired. Thus it follows that the variance of $f(\mathbf{x}_i; \hat{\vec{q}})$ is given by [Dybowski 1997]:

$$Var\left[f(\mathbf{x}_i;\hat{\vec{q}}) \right] = \mathbf{f}_0^T \mathbf{S} \mathbf{f}_0$$

The variance covariance matrix can be estimated in a variety of ways:

$\mathbf{S} = s^2 \mathbf{H}^{-1}$ [Dybowski 1997]

$\mathbf{S} = s^2 \mathbf{H}^{-1} \left[\mathbf{F}^T \mathbf{F} \right] \mathbf{H}^{-1}$ [Tibshirani 1996]

$\mathbf{S} = s^2 \left[\mathbf{F}^T \mathbf{F} \right]^{-1}$ [Chryssolouris 1996]

s^2 is an estimate of the noise variance s_e^2.

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} \left[y_i - f(\mathbf{x}_i;\hat{\vec{q}}) \right]^2$$ [Tibshirani 1996]

$$s^2 = \frac{1}{n-p} \sum_{i=1}^{n} \left[y_i - f(\mathbf{x}_i;\hat{\vec{q}}) \right]^2$$ [Chryssolouris 1996]

$\mathbf{F}$ is the $n \times p$ Jacobian matrix of first order partial derivatives with respect to the parameters determined from the least squares minimization of $S(\vec{q})$, i.e. $\mathbf{F}_{i,j} = \frac{\partial f(\mathbf{x}_i;\hat{\vec{q}})}{\partial q_j}$.

The number of observations used to develop the model (*training data*) is given by $i = 1,...,n$ and the number of parameters in the model is $j = 1,...,p$.

$\mathbf{H}$ is the $p \times p$ Hessian matrix of second order partial derivatives of $S(\vec{q})$, evaluated at $\vec{q} = \hat{\vec{q}}$, i.e. $\mathbf{H}_{j,k} = \frac{\partial^2 S(\hat{\vec{q}})}{\partial \hat{q}_j \partial \hat{q}_k}$, where $j = 1,...,p$ and $k = 1,...,p$.

Results from a Monte Carlo study of the above methods of covariance matrix estimation [Hogg 1987] indicated that the method based solely on the Jacobian matrix gave the best results with the least amount of effort.

Consider the nonlinear regression model for an observation $\mathbf{x}_0$:

$$y_0 = f(\mathbf{x}_0;\vec{q}) + e_0$$

The estimate of the response for this observation is given by:

$$\hat{y}_0 = f(\mathbf{x}_0;\hat{\vec{q}})$$

$$y_0 - \hat{y}_0 = f(\mathbf{x}_0;\vec{q}) + e_0 - f(\mathbf{x}_0;\hat{\vec{q}})$$

Using the Taylor series expansion:

$$f(\mathbf{x}_i;\hat{\vec{q}}) \approx f(\mathbf{x}_i,\vec{q}) + \mathbf{f}_0^T \cdot [\hat{\vec{q}} - \vec{q}]$$

$$y_0 - \hat{y}_0 \approx f(\mathbf{x}_0;\vec{q}) + e_0 - \left[f(\mathbf{x}_i,\vec{q}) + \mathbf{f}_0^T \cdot [\hat{\vec{q}} - \vec{q}] \right] = e_0 - \mathbf{f}_0^T \cdot [\hat{\vec{q}} - \vec{q}]$$

Since e_0 and $\hat{\vec{q}}$ are independent:

$$Var[y_0 - \hat{y}_0] \approx Var[e_0] + Var\left[\mathbf{f}_0^T \cdot [\hat{\vec{q}} - \vec{q}] \right]$$

$e_0 \sim N(0, s^2 \mathbf{I}_n)$, and $[\hat{\vec{q}} - \vec{q}] \sim N(0, \mathbf{S})$

$$Var[y_0 - \hat{y}_0] \approx s^2 + \mathbf{f}_0^T \mathbf{S} \mathbf{f}_0$$

The Student's t-distribution is then:

$$t_{n-p} \sim \frac{y_0 - \hat{y}_0}{\sqrt{Var[y_0 - \hat{y}_0]}} \approx \frac{y_0 - \hat{y}_0}{\sqrt{s^2 + \mathbf{f}_0^T \mathbf{S} \mathbf{f}_0}}$$

Where the variance is replaced by its estimate: $s^2 \approx s^2$

The estimation with its associated prediction interval is then:

$$\hat{y}_0 \pm t_{a/2}^{n-p} \sqrt{\mathbf{f}_0^T \mathbf{S} \mathbf{f}_0 + s^2}$$

Using the variance estimate: $\mathbf{S} = s^2 \left[\mathbf{F}^T \mathbf{F} \right]^{-1}$, this becomes:

$$\hat{y}_0 \pm t_{a/2}^{n-p} \cdot s \sqrt{1 + \mathbf{f}_0^T (\mathbf{F}^T \mathbf{F})^{-1} \mathbf{f}_0}$$

4.6 PREDICTION INTERVALS FOR ARTIFICIAL NEURAL NETWORKS

The inferential, single hidden layer, ANN is used as the base architecture for evaluation in this work. Extensions to multivariate outputs can be drawn; however, the mathematical treatment presented here is limited to the inferential design. It is worth mentioning that some alternate approaches to prediction interval estimation other than the one described in the previous section are available. One of which relies on the

construction of a second ANN to model the variance of the predictions. The approach is as follows [Dybowski 1997]:

If $f(\mathbf{x};\hat{\vec{q}})$ approximates $E[y \mid \mathbf{x}]$, when $\hat{\vec{q}}$ is obtained from training an ANN with the standard error function $S(\vec{q}) = \sum_{i=1}^{n} \left\{ \mathbf{y}_i - f(\mathbf{x}_i;\vec{q}) \right\}^2$, then $f(\mathbf{x};\hat{\mathbf{u}})$ approximates $Var[y \mid \mathbf{x}]$ when $\hat{\mathbf{u}}$ is obtained from training an ANN with the error function: .

$$S(\mathbf{u}) = \sum_{i=1}^{n} \left\{ \left[f(\mathbf{x}_i;\hat{\vec{q}}) - y_i \right]^2 - \left[f(\mathbf{x}_i;\mathbf{u}) \right] \right\}^2$$

Consider the expression $\left[f(\mathbf{x}_i;\hat{\vec{q}}) - y_i \right]^2$, since $f(\mathbf{x};\hat{\vec{q}})$ approximates $E[y \mid \mathbf{x}]$, this can be rewritten as $\left[E[y \mid \mathbf{x}] - y_i\right]^2$, which is the squared deviation of the i^{th} response observation from its mean. Thus, if $S(\vec{q})$ is considered the sum of squared errors between the observations and predictions, $S(\mathbf{u})$ is considered the sum of the squared errors between the observed variance of y and its prediction $\hat{s}_y^2 = f(\mathbf{x}_i;\hat{\mathbf{u}}) \approx Var[y \mid \mathbf{x}]$. Prediction intervals can then be prepared from the estimates of both ANNs at point y_0:

$$\hat{y}_0 \pm PI_{0.95} = f(\mathbf{x}_0;\hat{\vec{q}}) \pm z_{0.025}\sqrt{f(\mathbf{x}_0;\hat{\mathbf{u}})}$$

$PI_{0.95}$ is the 95% prediction interval and $z_{0.025}$ is the critical point of the standard normal distribution.

The approach based on linearization as described in the previous section on nonlinear regression is the one that is followed here. This approach assumes that $\hat{\vec{q}}$ is close to the true value of the set of parameters, $\vec{q}$, verified through observing sufficient prediction of the system behavior. This approach requires the computation of the following derivatives:

$$\mathbf{f}_0^{\mathbf{T}} = \left(\frac{\partial f(\mathbf{x}_0;q^*)}{\partial q_1^*}, \frac{\partial f(\mathbf{x}_0;q^*)}{\partial q_2^*}, \ldots, \frac{\partial f(\mathbf{x}_0;q^*)}{\partial q_p^*} \right)$$

The subscript 0 denotes an observation other than those used for least squares estimation of $\hat{\hat{q}}$, i.e. an independent observation for which an estimate is desired. The Jacobian matrix $\mathbf{F}\cdot(\hat{q})$ has the form:

$$\mathbf{F}\cdot(\hat{q}) = \frac{\partial f(\mathbf{x},\hat{q})}{\partial \hat{q}} = \begin{bmatrix} \left(\frac{\partial f_1(x_1,\hat{q})}{\partial \hat{q}_1}\right) & \left(\frac{\partial f_1(x_1,\hat{q})}{\partial \hat{q}_2}\right) & \cdot & \cdot & \cdot & \left(\frac{\partial f_1(x_1,\hat{q})}{\partial \hat{q}_p}\right) \\ \left(\frac{\partial f_2(x_2,\hat{q})}{\partial \hat{q}_1}\right) & \cdot & & & & \left(\frac{\partial f_2(x_2,\hat{q})}{\partial \hat{q}_p}\right) \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ \left(\frac{\partial f_n(x_n,\hat{q})}{\partial \hat{q}_1}\right) & \left(\frac{\partial f_n(x_n,\hat{q})}{\partial \hat{q}_2}\right) & \cdot & \cdot & \cdot & \left(\frac{\partial f_n(x_n,\hat{q})}{\partial \hat{q}_p}\right) \end{bmatrix}$$

The dimensions of the Jacobian are $(n\times p)$, where n is the number of samples used to obtain $\hat{\hat{q}}$ and p is the number of parameters $\hat{q}_i$ which compose $\hat{q}$. To provide an overall mathematical perspective on the required computations of a standard inferential ANN, refer to figure 4.6.1. Each section (input stage, hidden layer, and output layer) of the Inferential ANN (IANN) is described in detail:

Input Stage:

The input section of the IANN is referred to as the *input stage* to reserve the term *layer* for a true computational layer, which consists of a set of weights, biases, and activation functions. To emphasize this, each input location is denoted by a dashed circle rather than a completed circle. A dashed circle does not contain an activation function whereas a completed circle does contain an activation function. Input values to the ANN are represented by an $(n\times n_0)$ matrix $\mathbf{X}$. The number of samples or observations in $\mathbf{X}$ is n, and the number of predictor variables in $\mathbf{X}$ is n_0. To simplify the illustration, the inputs are presented to the IANN as scalar quantities $x_{i,j}$ which represent the i^{th} observation for the j^{th} predictor variable in $\mathbf{X}$, where $i = 1,2,...n$ and $j = 1,2,...,n_0$. The number of input locations is n_0.

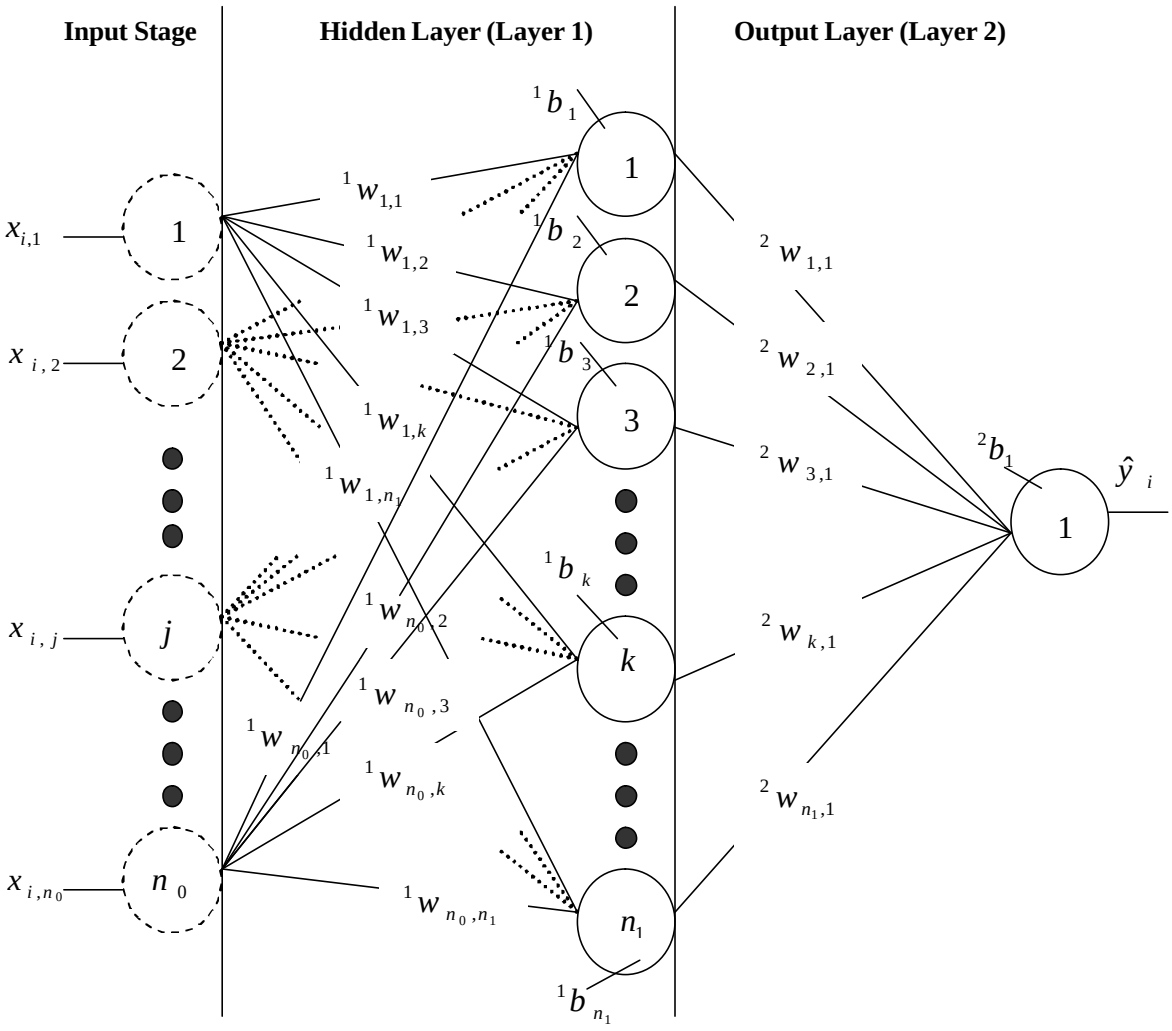


Figure 4.6.1: Schematic of an inferential ANN

Hidden Layer (Layer 1):

The hidden layer is the main computing layer of the IANN architecture. Each neuron in the hidden layer has an associated weight, bias, and activation function. Each weight in the hidden layer has 3 unique identifying descriptors, two subscripts and a superscript. The general form of a hidden layer weight is: ${}^{1}w_{j,k}$. The superscript 1 indicates that the weight belongs to layer 1, the hidden layer. The first subscript, j, refers to an input location, $j = 1,2,...,n_0$, and n_0 is the number of locations in the input stage. The second subscript, k, refers to a neuron in the hidden layer, $k = 1,2,...,n_1$, and n_1 is the number of neurons in the hidden layer. The first subscript indicates the origin of the connection (in the previous layer), and the second subscript indicates the destination of the connection (in the current layer specified by the superscript). For example: ${}^{1}w_{1,3}$ - is the hidden layer weight which acts upon values originating from the first input location and destined for the 3rd neuron in layer one. Each bias value in the hidden layer has 2 unique identifiers. The general form of a hidden layer bias is: ${}^{1}b_k$. Again the superscript 1 indicates that it is a hidden layer bias. The subscript, k, indicates the specific neuron with which the bias is associated. For example ${}^{1}b_k$ - is the bias associated with the k^{th} neuron in layer 1.

Output Layer (Layer 2):

The output layer performs a weighted sum of the output values from the hidden layer, and adds a scalar bias. The weights in the output layer are identified by 3 unique indices, similar to the hidden layer weights. The general form of the output layer weight is: ${}^{2}w_{k,1}$. The superscript 2 indicates that the weight belongs to layer 2, the output layer. The first subscript, k, refers to the origin of the connection in the hidden layer. The second subscript refers to the neuron in the output layer, which can only be 1 in the IANN architecture. For example: ${}^{2}w_{3,1}$ - is the output layer weight which acts upon values being output from the 3rd hidden neuron and destined for the output neuron. Because

there is only 1 output neuron in the IANN structure there is only one bias value in the output layer, denoted as: ${}^{2}b_1$. The superscript identifies that the bias is located in the output layer, and the subscript of 1, though not necessary here because of the IANN architecture, is maintained to follow the conventions described herein. The output neuron activation function is a linear function.

The IANN computations and the construction of the Jacobian matrix are now described in detail. To construct the Jacobian matrix of first order partial derivatives of the IANN function approximation with respect to the network parameters, first the set of parameters, $\hat{q}$, must be defined:

$\hat{q} = \left[\hat{q}_1 \quad \hat{q}_2 \quad \cdot \quad \cdot \quad \cdot \quad \hat{q}_p\right]$, where $p = n_0 n_1 + 2n_1 + 1$

$$\hat{q} = \begin{cases} \left[{}^{1}w_{1,1} \quad {}^{1}w_{1,2} \quad \cdots \quad {}^{1}w_{1,k} \quad \cdots \quad {}^{1}w_{1,n_1} \quad \cdots \right. \\ {}^{1}w_{2,1} \quad {}^{1}w_{2,2} \quad \cdots \quad {}^{1}w_{2,k} \quad \cdots \quad {}^{1}w_{2,n_1} \quad \cdots \\ {}^{1}w_{n_0,1} \quad {}^{1}w_{n_0,2} \quad \cdots \quad {}^{1}w_{n_0,k} \quad \cdots \quad {}^{1}w_{n_0,n_1} \quad \cdots \\ {}^{1}b_1 \quad {}^{1}b_2 \quad \cdots \quad {}^{1}b_k \quad \cdots \quad {}^{1}b_{n_1} \quad \cdots \\ \left. {}^{2}w_{1,1} \quad {}^{2}w_{2,1} \quad \cdots \quad {}^{2}w_{k,1} \quad \cdots \quad {}^{1}w_{n_1,1_1} \quad \cdots \quad {}^{2}b_1 \right] \end{cases}$$

To construct the Jacobian matrix, the partial derivatives of $\hat{y}_i$ with respect to each of the weight and bias parameters of the ANN must be obtained. Accordingly, there will be p partial derivatives for each observation i. Before determining the partial derivatives, the activation functions must be chosen. The output neuron activation function is linear, and the hidden neuron activation functions will be either the hyperbolic tangent function, or the sigmoidal activation function. It is easiest to first determine the partial derivatives with respect to the parameters in the output layer. Figure 4.6.2 provides a schematic of the output layer neuron of the IANN.

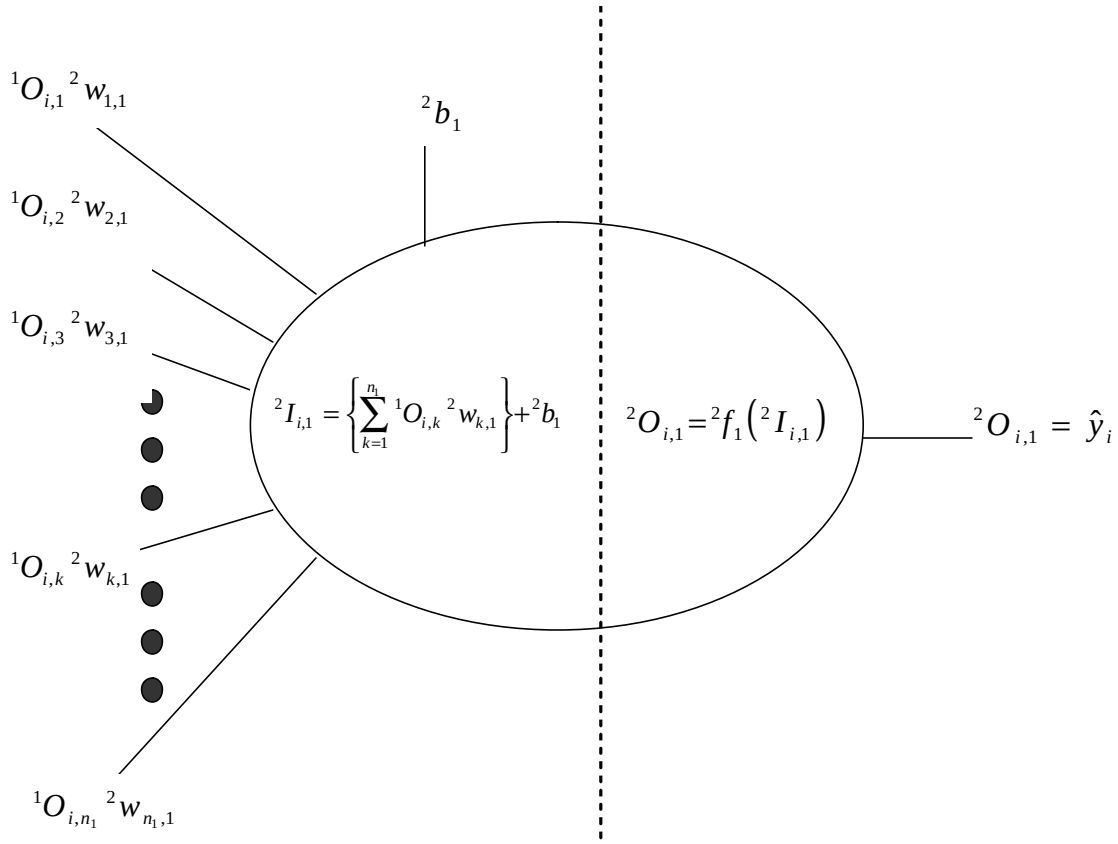


Figure 4.6.2: IANN output layer neuron

The quantities coming into the neuron are the products of the outputs of the previous layer (hidden layer) and the output layer weights, e.g. ${}^{1}O_{i,k}$ is the output of the k^{th} neuron in layer 1 (indicated by the superscript). Since there are n_1 hidden neurons, there will be n_1 products of output layer weights and hidden neuron outputs. In addition, the output neuron bias enters at this point and the overall input to the output linear neuron is identified as ${}^{2}I_{i,1}$, the sum of the products of the previous layers outputs and the current layer's weights and the output neuron bias. The activation function of the output layer neuron operates on this input with its linear function: $f(u)=u$. In keeping with the defined variable conventions, this is written as ${}^{2}f_1\left({}^{2}I_{i,1}\right)={}^{2}I_{i,1}$. The resultant quantity shown on the right side of figure 4.6.2 is the final IANN output for observation i, ${}^{2}O_{i,1}=\hat{y}_i$. The required partial derivatives for the output layer parameters are:

$$\frac{\partial\hat{y}_i}{\partial\left({}^{2}b_1\right)}=\frac{\partial\hat{y}_i}{\partial\left({}^{2}O_{i,1}\right)}\cdot\frac{\partial\left({}^{2}O_{i,1}\right)}{\partial\left({}^{2}b_1\right)}=\frac{\partial\hat{y}_i}{\partial\left({}^{2}O_{i,1}\right)}\cdot\frac{\partial\left({}^{2}O_{i,1}\right)}{\partial\left({}^{2}I_{i,1}\right)}\cdot\frac{\partial\left({}^{2}I_{i,1}\right)}{\partial\left({}^{2}b_1\right)}=1$$

$$\frac{\partial\hat{y}_i}{\partial\left({}^{2}w_{k,1}\right)}=\frac{\partial\hat{y}_i}{\partial\left({}^{2}O_{i,1}\right)}\cdot\frac{\partial\left({}^{2}O_{i,1}\right)}{\partial\left({}^{2}w_{k,1}\right)}=\frac{\partial\hat{y}_i}{\partial\left({}^{2}O_{i,1}\right)}\cdot\frac{\partial\left({}^{2}O_{i,1}\right)}{\partial\left({}^{2}I_{i,1}\right)}\cdot\frac{\partial\left({}^{2}I_{i,1}\right)}{\partial\left({}^{2}w_{k,1}\right)}={}^{1}O_{i,k}$$

These results can be verified by evaluating the derivatives separately:

$${}^{2}O_{i,1}={}^{2}f_1\left({}^{2}I_{i,1}\right) \qquad \hat{y}_i={}^{2}O_{i,1}, \text{ and } \frac{\partial\hat{y}_i}{\partial\left({}^{2}O_{i,1}\right)}=1$$

$${}^{2}f_1\left({}^{2}I_{i,1}\right)={}^{2}I_{i,1} \qquad {}^{2}I_{i,1}=\left\{\sum_{k=1}^{n_1}{}^{1}O_{i,k}\,{}^{2}w_{k,1}\right\}+{}^{2}b_1$$

$$\frac{\partial\left({}^{2}O_{i,1}\right)}{\partial\left({}^{2}I_{i,1}\right)}=1 \qquad \frac{\partial\left({}^{2}I_{i,1}\right)}{\partial\left({}^{2}b_1\right)}=1, \text{ and } \frac{\partial\left({}^{2}I_{i,1}\right)}{\partial\left({}^{2}w_{k,1}\right)}={}^{1}O_{i,k}$$

Moving to the hidden layer parameters, consider figure 4.6.3.

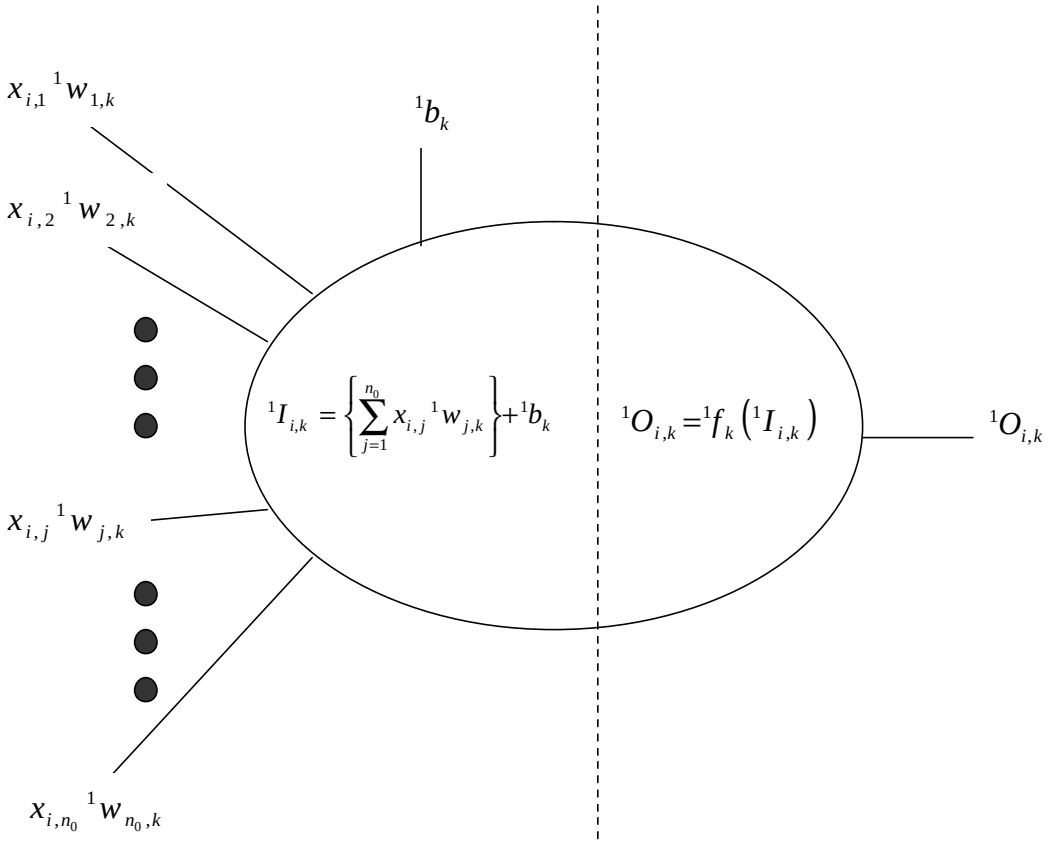


Figure 4.6.3: IANN hidden layer neuron

The quantities coming into the neuron are the products of the outputs of the previous layer (input stage), $x_{i,j}$, and the hidden layer weights, $w_{j,k}$. Since there are n_0 locations in the input stage, there will be n_0 products of hidden layer weights and input values. In addition, a bias enters at this point and the overall input to the k^{th} hidden layer neuron is identified as ${}^1I_{i,k}$, the sum of the products of the input stage values and the hidden layer's weights and the k^{th} hidden neuron bias. The activation functions of the hidden layer neurons operate on this input with a nonlinear function. Two common choices are the hyperbolic tangent function $f(u) = \tanh(u) \approx \left[\frac{2}{1+e^{-2u}} - 1\right]$ and the logarithmic sigmoid transfer function $f(u) = \frac{1}{1+e^{-u}}$. Depending on the choice of activation function, and expressing in the defined terms, the output of the hidden neurons is given by:

- for hyperbolic tangent:

If ${}^1f_k\left({}^1I_{i,k}\right) = \tanh\left({}^1I_{i,k}\right) \approx \left[\frac{2}{1+\exp\left(-2\left({}^1I_{i,k}\right)\right)} - 1\right]$,

then ${}^1O_{i,k} = {}^1f_k\left({}^1I_{i,k}\right) \approx \left[\frac{2}{1+\exp\left(-2\left({}^1I_{i,k}\right)\right)} - 1\right]$,

and $\frac{\partial\left({}^1O_{i,k}\right)}{\partial\left({}^1I_{i,k}\right)} = 1 - \left[{}^1I_{i,k}\right]^2$.

- for logarithmic sigmoid:

If ${}^1f_k\left({}^1I_{i,k}\right) = \frac{1}{1+\exp\left(-\left({}^1I_{i,k}\right)\right)}$,

then ${}^1O_{i,k} = {}^1f_k\left({}^1I_{i,k}\right) = \frac{1}{1+\exp\left(-\left({}^1I_{i,k}\right)\right)}$,

and $\frac{\partial\left({}^1O_{i,k}\right)}{\partial\left({}^1I_{i,k}\right)} = \left({}^1I_{i,k}\right)\left[1 - \left({}^1I_{i,k}\right)\right]$.

Hyperbolic tangent activation functions will be used in deriving the partial derivatives for the hidden layer parameters:

$$\frac{\partial \hat{y}_i}{\partial\left({}^1b_k\right)} = \frac{\partial \hat{y}_i}{\partial\left({}^2O_{i,1}\right)} \cdot \frac{\partial\left({}^2O_{i,1}\right)}{\partial\left({}^2I_{i,1}\right)} \cdot \frac{\partial\left({}^2I_{i,1}\right)}{\partial\left({}^1O_{i,k}\right)} \cdot \frac{\partial\left({}^1O_{i,k}\right)}{\partial\left({}^1I_{i,k}\right)} \cdot \frac{\partial\left({}^1I_{i,k}\right)}{\partial\left({}^1b_k\right)} = (1)\cdot(1)\cdot\left({}^2w_{k,1}\right)\cdot\left(1-\left[{}^1I_{i,k}\right]^2\right)\cdot(1)$$

$$\frac{\partial \hat{y}_i}{\partial\left({}^1b_k\right)} = \left({}^2w_{k,1}\right)\left(1-\left[{}^1I_{i,k}\right]^2\right)$$

$$\frac{\partial \hat{y}_i}{\partial\left({}^1w_{j,k}\right)} = \frac{\partial \hat{y}_i}{\partial\left({}^2O_{i,1}\right)} \cdot \frac{\partial\left({}^2O_{i,1}\right)}{\partial\left({}^2I_{i,1}\right)} \cdot \frac{\partial\left({}^2I_{i,1}\right)}{\partial\left({}^1O_{i,k}\right)} \cdot \frac{\partial\left({}^1O_{i,k}\right)}{\partial\left({}^1I_{i,k}\right)} \cdot \frac{\partial\left({}^1I_{i,k}\right)}{\partial\left({}^1w_{j,k}\right)} = (1)\cdot(1)\cdot\left({}^2w_{k,1}\right)\cdot\left(1-\left[{}^1I_{i,k}\right]^2\right)\cdot\left(x_{i,j}\right)$$

$$\frac{\partial \hat{y}_i}{\partial\left({}^1w_{j,k}\right)} = \left({}^2w_{k,1}\right)\left(1-\left[{}^1I_{i,k}\right]^2\right)\left(x_{i,j}\right)$$

These results can be verified by evaluating the derivatives separately. From the previous results of the output layer neuron:

$$\frac{\partial \hat{y}_i}{\partial\left({}^2O_{i,1}\right)} = 1 \quad \text{and} \quad \frac{\partial\left({}^2O_{i,1}\right)}{\partial\left({}^2I_{i,1}\right)} = 1.$$

From the input section of the output neuron: ${}^2I_{i,1} = \left\{\sum_{k=1}^{n_1} {}^1O_{i,k}\,{}^2w_{k,1}\right\} + {}^2b_1$,

thus: $\frac{\partial\left({}^2I_{i,1}\right)}{\partial\left({}^1O_{i,k}\right)} = 1$.

From the previous results for the hyperbolic tangent activation function:

$$\frac{\partial\left({}^1O_{i,k}\right)}{\partial\left({}^1I_{i,k}\right)} = 1-\left[{}^1I_{i,k}\right]^2$$

From the input section of the output neuron: ${}^1I_{i,k} = \left\{\sum_{j=1}^{n_0} x_{i,j}\,{}^1w_{j,k}\right\} + {}^1b_k$

thus: $\frac{\partial\left({}^{1}I_{i,k}\right)}{\partial\left({}^{1}b_{k}\right)}=1$ and $\frac{\partial\left({}^{1}I_{i,k}\right)}{\partial\left({}^{1}w_{j,k}\right)}=x_{i,j}$.

This completes the derivations of the full complement of partial derivatives required for construction of the Jacobian matrix. The $(1\times p)$ vector of partial derivatives for the i^{th} observation is defined as:

$$\frac{\partial\hat{y}_i}{\partial\hat{q}}=\begin{Bmatrix}\left[\frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{1,1}\right)} & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{1,2}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{1,k}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{1,n_1}\right)} & \cdots\right.\\ \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{2,1}\right)} & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{2,2}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{2,k}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{2,n_1}\right)} & \cdots\\ \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{n_0,1}\right)} & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{n_0,2}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{n_0,k}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{n_0,n_1}\right)} & \cdots\\ \frac{\partial\hat{y}_i}{\partial\left({}^{1}b_{1}\right)} & \frac{\partial\hat{y}_i}{\partial\left({}^{1}b_{2}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{1}b_{k}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{1}b_{n_1}\right)} & \cdots\\ \frac{\partial\hat{y}_i}{\partial\left({}^{2}w_{1,1}\right)} & \frac{\partial\hat{y}_i}{\partial\left({}^{2}w_{2,1}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{2}w_{k,1}\right)} & \cdots & \frac{\partial\hat{y}_i}{\partial\left({}^{1}w_{n_1,1}\right)} & \cdots & \left.\frac{\partial\hat{y}_i}{\partial\left({}^{2}b_{1}\right)}\right]\end{Bmatrix}$$

Substituting the derived forms of the partial derivatives results in:

$$\frac{\partial\hat{y}_i}{\partial\hat{q}}=\begin{Bmatrix}\left[\left({}^{2}w_{1,1}\right)\left(1-\left[{}^{1}I_{i,1}\right]^2\right)\left(x_{i,1}\right) & \left({}^{2}w_{2,1}\right)\left(1-\left[{}^{1}I_{i,2}\right]^2\right)\left(x_{i,1}\right) & \cdots & \left({}^{2}w_{k,1}\right)\left(1-\left[{}^{1}I_{i,k}\right]^2\right)\left(x_{i,1}\right) & \cdots & \left({}^{2}w_{n_1,1}\right)\left(1-\left[{}^{1}I_{i,n_1}\right]^2\right)\left(x_{i,1}\right) & \cdots\\ \left({}^{2}w_{1,1}\right)\left(1-\left[{}^{1}I_{i,1}\right]^2\right)\left(x_{i,2}\right) & \left({}^{2}w_{2,1}\right)\left(1-\left[{}^{1}I_{i,2}\right]^2\right)\left(x_{i,2}\right) & \cdots & \left({}^{2}w_{k,1}\right)\left(1-\left[{}^{1}I_{i,k}\right]^2\right)\left(x_{i,2}\right) & \cdots & \left({}^{2}w_{n_1,1}\right)\left(1-\left[{}^{1}I_{i,n_1}\right]^2\right)\left(x_{i,2}\right) & \cdots\\ \left({}^{2}w_{1,1}\right)\left(1-\left[{}^{1}I_{i,1}\right]^2\right)\left(x_{i,n_0}\right) & \left({}^{2}w_{2,1}\right)\left(1-\left[{}^{1}I_{i,2}\right]^2\right)\left(x_{i,n_0}\right) & \cdots & \left({}^{2}w_{k,1}\right)\left(1-\left[{}^{1}I_{i,k}\right]^2\right)\left(x_{i,n_0}\right) & \cdots & \left({}^{2}w_{n_1,1}\right)\left(1-\left[{}^{1}I_{i,n_1}\right]^2\right)\left(x_{i,n_0}\right) & \cdots\\ \left({}^{2}w_{1,1}\right)\left(1-\left[{}^{1}I_{i,1}\right]^2\right) & \left({}^{2}w_{2,1}\right)\left(1-\left[{}^{1}I_{i,2}\right]^2\right) & \cdots & \left({}^{2}w_{k,1}\right)\left(1-\left[{}^{1}I_{i,k}\right]^2\right) & \cdots & \left({}^{2}w_{n_1,1}\right)\left(1-\left[{}^{1}I_{i,n_1}\right]^2\right) & \cdots\\ {}^{1}O_{i,1} & {}^{1}O_{i,1} & \cdots & {}^{1}O_{i,1} & \cdots & {}^{1}O_{i,1} & 1\end{Bmatrix}$$

The full $(n\times p)$ Jacobian matrix can then be constructed from the n, $\frac{\partial\hat{y}_i}{\partial\hat{q}}$ vectors as follows:

$$\mathbf{F}\cdot\left(\hat{q}\right)=\frac{\partial\hat{\mathbf{y}}}{\partial\hat{q}}=\begin{bmatrix}\frac{\partial\hat{y}_1}{\partial\hat{q}}\\ \frac{\partial\hat{y}_2}{\partial\hat{q}}\\ \cdots\\ \frac{\partial\hat{y}_i}{\partial\hat{q}}\end{bmatrix}$$

To construct prediction intervals for a new observation $\mathbf{x}_0$, the vector of partial derivatives of $f(\mathbf{x}_0;q^*)$ must also be constructed for each new observation.

$$\mathbf{f}_0^{\mathbf{T}}=\left(\frac{\partial f(\mathbf{x}_0;q^*)}{\partial q_1^*},\frac{\partial f(\mathbf{x}_0;q^*)}{\partial q_2^*},...,\frac{\partial f(\mathbf{x}_0;q^*)}{\partial q_p^*}\right)$$

This vector can be constructed using the form provided by $\frac{\partial\hat{y}_i}{\partial\hat{q}}$, with the simple substitution of 0 for i, and noting that while $i=1,...,n$, the 0 subscript refers to a single observation. The relationship is given by:

$$\mathbf{f}_0^{\mathbf{T}}=\frac{\partial\hat{y}_0}{\partial\hat{q}}$$

The overall approach is to calculate the full Jacobian matrix based on the training data and the resultant IANN weight and bias parameters. An estimate of the variance of the error term due to model limitations is then obtained based on the training data, and the following equation:

$$s^2=\frac{\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2}{n}$$ [Tibshirani 1996]

For each new observation, $\mathbf{x}_0$, compute the IANN estimate $\hat{y}_0=f(\mathbf{x}_0;\hat{q})$, the vector of partial derivatives, $\mathbf{f}_0$, and the $(100-a)\times100\%$ prediction interval from:

$$\hat{y}_0\pm t_{a/2}^{n-p}\cdot s\sqrt{1+\mathbf{f}_0^T(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{f}_0}$$

4.7 PREDICTION INTERVALS FOR NEURAL NETWORK PARTIAL LEAST SQUARES

To begin, a brief description of the approach analogous to the model used by Chryssolouris et. al. [1996] is presented. After providing the basics, a detailed description and derivation of the partial derivative equations ensues. Consider the nonlinear functional mapping of a set of input variables, as presented by Baffi et. al. [2002] $\mathbf{X}$ to an output variable $\mathbf{y}$, where the predicted output variable $\hat{\mathbf{y}}$ can be written as:

$$\hat{\mathbf{y}} = f(\mathbf{X}, \hat{\vec{q}})$$

where $\hat{\vec{q}}$ denotes the vector of model parameters $\hat{\vec{q}} = \begin{bmatrix} \hat{q}_1 & \hat{q}_2 & \cdots & \hat{q}_c & \cdots & \hat{q}_N \end{bmatrix}$, where $c = 1,..., N$, and N is the total number of parameters of the NNPLS model.

The Jacobian matrix for this case is thus:

$$\mathbf{F} = \mathbf{F}(\Theta) = \left[\frac{\partial f(\mathbf{X}, \vec{q})}{\partial q_c} \right]$$

The functional mapping for PLS-1 can be written as:

$$\hat{\mathbf{y}} = f_{PLS}(\mathbf{X}, \mathbf{R}, \hat{\vec{q}}, \mathbf{q})$$

Where $\mathbf{R}$ denotes the weight matrix that maps the input variables $\mathbf{X}$ onto the input latent variables $\mathbf{T}$, and $\mathbf{q}$ is the vector of coefficients for the outer mapping between $\mathbf{U}$ and $\hat{\mathbf{y}}$. The parameter vector $\hat{\vec{q}}$ is obtained from the least squares minimization of the error between the input and output latent variables ($\mathbf{T}$ and $\mathbf{U}$, respectively),

$S(\vec{q}) = \sum_{j=1}^{m} \left\{ \mathbf{u}_j - f(\mathbf{t}_j; \vec{q}) \right\}^2$, where $j = 1,..., m$ and m is the number of latent variables in the NNPLS model. The Jacobian matrix for this case can be written in expanded form as:

$$\mathbf{F} = [\mathbf{F}_R \mathbf{F}_{\vec{q}} \mathbf{F}_Q] = \left[\frac{\partial f_{PLS}}{\partial \mathbf{R}} \frac{\partial f_{PLS}}{\partial \vec{q}} \frac{\partial f_{PLS}}{\partial \mathbf{q}} \right]$$

Simplifying further for each dimension of the PLS model into the product of the appropriate partial derivatives, i.e. for each pair of input/output latent variables $\mathbf{t}_j / \mathbf{u}_j$:

$$\mathbf{F}_R = \left[\frac{\partial f_{PLS}}{\partial \mathbf{R}_j}\right] = \left[\frac{\partial f_{PLS}}{\partial \mathbf{u}_j}\frac{\partial \mathbf{u}_j}{\partial \mathbf{t}_j}\frac{\partial \mathbf{t}_j}{\partial R_{a,j}}\right]$$

a and j denote the indices for the input variables and latent variables respectively. Thus, $R_{a,j}$ is the weighting coefficient between the a^{th} input variable ($a = 1,..., p$, p is the number of input variables) and the j^{th} latent variable, $\mathbf{t}_j$.

$$\mathbf{F}_{\vec{q}} = \left[\frac{\partial f_{PLS}}{\partial \vec{q}_j}\right] = \left[\frac{\partial f_{PLS}}{\partial \mathbf{u}_j}\frac{\partial \mathbf{u}_j}{\partial \vec{q}_j}\right]$$

$\vec{q}_j$ is the set of inner model coefficients for the j^{th} pair of input/output latent variables.

$$\mathbf{F}_Q = \left[\frac{\partial f_{PLS}}{\partial q_j}\right]$$

q_j is the output loading for the j^{th} latent variable.

Rewriting the PLS model as:

$$\hat{\mathbf{y}} = \sum_{j=1}^{m} \mathbf{u}_j \cdot q_j$$

$$\mathbf{u}_j = f(\mathbf{t}_j, \vec{q}_j)$$

$$\mathbf{t}_j = \sum_{a=1}^{p} \mathbf{x}_i \cdot R_{a,j}$$

Note that $\mathbf{F}$ is the Jacobian matrix computed using the training data, and $\mathbf{f}_0$ is the Jacobian computed for a new observation $\mathbf{x}_0$ used for computing prediction intervals for the corresponding prediction $\hat{y}_0$.

Assuming the data are normally distributed, the $(1-a)*100\%$ confidence interval can be estimated by:

$$\hat{y}_0 \pm t_{n-p,1-a/2} \cdot s \cdot \sqrt{(1 + \mathbf{f}_0^{\mathbf{T}} \cdot (\mathbf{F}^{\mathbf{T}} \cdot \mathbf{F})^{-1} \cdot \mathbf{f}_0)}$$

Now that the overall procedure has been outlined the details of the model and the computation of the required partial derivatives are described. The derivations which follow are based on the nonlinear iterative partial least squares algorithm (NIPALS), which is presented below, with a modification noted for the inner relationship models:

1. Mean center and scale $\mathbf{X}$ and $\mathbf{Y}$
2. Set the output scores $\mathbf{u}$ equal to a column of $\mathbf{Y}$.
3. Compute input weight $\mathbf{w}$ by regressing $\mathbf{X}$ on $\mathbf{u}$ $\quad \mathbf{w}^T = \dfrac{\mathbf{u}^T\mathbf{X}}{\mathbf{u}^T\mathbf{u}}$
4. Normalize Bw to unit length $\quad \mathbf{w} = \dfrac{\mathbf{w}}{\|\mathbf{w}\|}$
5. Calculate the input scores $\mathbf{t}$ $\quad \mathbf{t} = \dfrac{\mathbf{X}\mathbf{w}}{\mathbf{w}^T\mathbf{w}}$
6. Compute output loadings $\mathbf{q}$ by regressing $\mathbf{Y}$ on $\mathbf{t}$ $\quad \mathbf{q}^T = \dfrac{\mathbf{t}^T\mathbf{Y}}{\mathbf{t}^T\mathbf{t}}$
7. Normalize $\mathbf{q}$ to unit length $\quad \mathbf{q} = \dfrac{\mathbf{q}}{\|\mathbf{q}\|}$
8. Calculate new output scores $\mathbf{u}$ $\quad \mathbf{u} = \dfrac{\mathbf{Y}\mathbf{q}}{\mathbf{q}^T\mathbf{q}}$
9. Check convergence on $\mathbf{u}$: if yes go to 10, else go to 3
10. Compute input loadings $\mathbf{p}$ by regressing $\mathbf{X}$ on $\mathbf{t}$ $\quad \mathbf{p}^T = \dfrac{\mathbf{t}^T\mathbf{X}}{\mathbf{t}^T\mathbf{t}}$
11. Compute inner model * (neural network) $\quad \mathbf{u} = f(\mathbf{t})$
12. Calculate input residual matrix $\quad \mathbf{E} = \mathbf{X} - \mathbf{t}\mathbf{p}^T$
13. Calculate output residual matrix $\quad \mathbf{F} = \mathbf{Y} - f(\mathbf{t})\mathbf{q}^T$
14. If additional dimensions necessary replace $\mathbf{X}$ and $\mathbf{Y}$ by $\mathbf{E}$ and $\mathbf{F}$ respectively, and repeat steps 2-13.

A projection matrix $\mathbf{R}$ is also defined as $\mathbf{R} = \mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1}$, which can be used to compute the scores directly from the input data $\mathbf{X}$ via: $\mathbf{T} = \mathbf{X}\mathbf{R}$

Partial least squares approaches are often described in two parts, the inner relationships, and the outer relationships. The outer relationships consist of a set of transformations, from the input space to the orthogonal latent variable space and from the output space to a set of output score vectors. For the inferential design, the output space is a vector, and thus the output transformation weights are simply ones. To follow convention, the output weights are maintained as variables in the following discussions. Figure 4.7.1 provides a schematic of the outer relationships of the inferential NNPLS design. Note that in the diagram, a single observation vector is mapped to a scalar response. Extensions to matrix notation are straightforward.

Regarding figure NNPLS1, the observation vector can be represented as:
$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_a \quad \cdots \quad x_p]$, and the corresponding vector of latent values is given by: $\mathbf{t} = [t_1 \quad t_2 \quad \cdots \quad t_j \quad \cdots \quad t_m]$. Note that the parameters displayed in figure 4.7.1 are determined by the NNPLS1 algorithm, and there is no direct iterative training required to determine these parameters. However, these parameters are affected by the results of the training of the inner relationship models. The number of input variables is p, and the number of latent variables is m, where the usual condition is $m < p$. The solid connection lines between the observation vector and the latent values have their corresponding transformation weights shown (r's), whereas the dotted lines do not have their weights shown. The dotted lines are maintained to show the overall structure. The first transformation weight subscript identifies the feature of the observation that is acted upon, and the second subscript indicates the destination latent variable for the product. The orthogonal transformation can also be written as:
$t_j = \mathbf{x} \cdot \mathbf{r}_j$, where $\mathbf{r}_j = [r_{1,j} \quad r_{2,j} \quad \cdots \quad r_{a,j} \quad \cdots \quad r_{p,j}]^{\mathbf{T}}$.
The subscripts on the output score values and the output transformation weights indicate the corresponding latent variable.

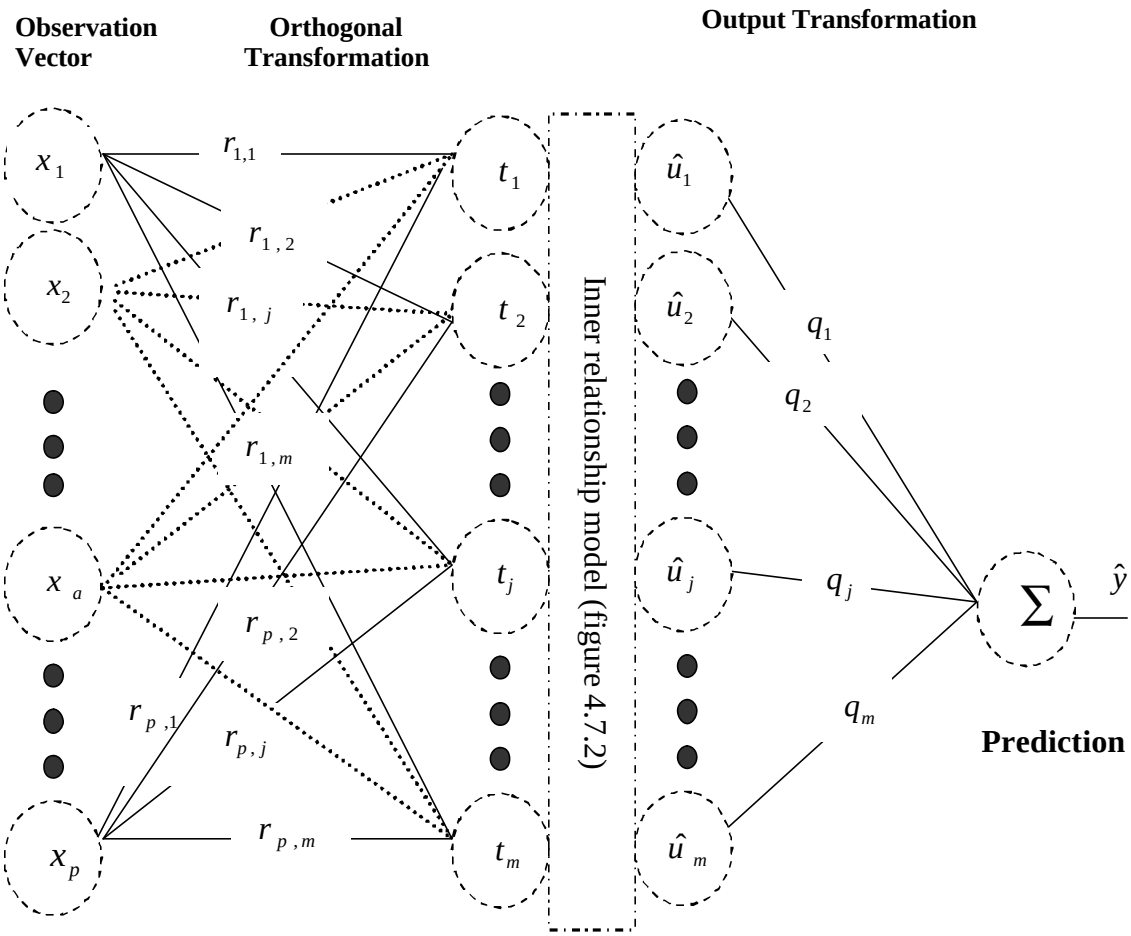


Figure 4.7.1: Schematic of the outer relationships of an inferential NNPLS model.

The output transformation can be written as: $\hat{y} = \sum_{j=1}^{m} \hat{u}_j \cdot q_j$, or in vector notation:

$\hat{y} = \hat{\mathbf{u}} \cdot \mathbf{q}$, where $\hat{\mathbf{u}} = \begin{bmatrix} \hat{u}_1 & \hat{u}_2 & \cdots & \hat{u}_j & \cdots & \hat{u}_m \end{bmatrix}$ and $\mathbf{q} = \begin{bmatrix} q_1 & q_2 & \cdots & q_j & \cdots & q_m \end{bmatrix}^{\mathbf{T}}$.

The vertically oriented rectangle (figure 4.7.1) represents the inner relationship mapping between the input and output latent variable pairs. The details of this relationship are presented in figure 4.7.2.

The notation of figure 4.7.2 is based on a single input latent variable vector $\mathbf{t} = [t_1 \quad t_2 \quad \cdots \quad t_j \quad \cdots \quad t_m]$, and its corresponding output latent variable vector estimate $\hat{\mathbf{u}} = \begin{bmatrix} \hat{u}_1 & \hat{u}_2 & \cdots & \hat{u}_j & \cdots & \hat{u}_m \end{bmatrix}$. These two vectors are obtained from a single observation vector $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_a \quad \cdots \quad x_p]$ and its corresponding estimated response $\hat{y}$ via: $t_j = \mathbf{x} \cdot \mathbf{r}_j$, and $\hat{y} = \hat{\mathbf{u}} \cdot \mathbf{q}$. The overall inner relationship model consists of a set of j single input single output neural networks. Each of these neural networks contains a single hidden layer of hyperbolic tangent activation functions, and a single linear output neuron. The input stage presents the latent observation vector to the hidden layer. The hidden layer independently maps each element of the input latent variable vector to a set of neurons. The corresponding weights and biases are indicated in the figure 4.7.2. Consider element j of the latent vector t_j. The associated hidden layer weights are defined as: $\vec{\ell}_{(:,1,j)} = [\ell_{1,1,j} \quad \ell_{2,1,j} \quad \cdots \quad \ell_{k_j,1,j} \quad \cdots \quad \ell_{n_j,1,j}]$, and the associated biases are $\mathbf{b}_{(:,1,j)} = [b_{1,1,j} \quad b_{2,1,j} \quad \cdots \quad b_{k_j,1,j} \quad \cdots \quad b_{n_j,1,j}]$. The number of neurons in the j^{th} neural network is given by n_j, and the index k_j is defined as $k_j = 1,...,n_j$. The number of hidden neurons each of the j neural networks is not required to be the same. Moving to the output layer, still following the path of t_j, the corresponding output layer weights are $\vec{\ell}_{(:,2,j)} = [\ell_{1,2,j} \quad \ell_{2,2,j} \quad \cdots \quad \ell_{k_j,2,j} \quad \cdots \quad \ell_{n_j,2,j}]$, and the output bias is $b_{1,2,j}$. Each weight and bias value has 3 subscripts identifying its location in the overall inner relationship model. The first subscript identifies the appropriate neuron, the second

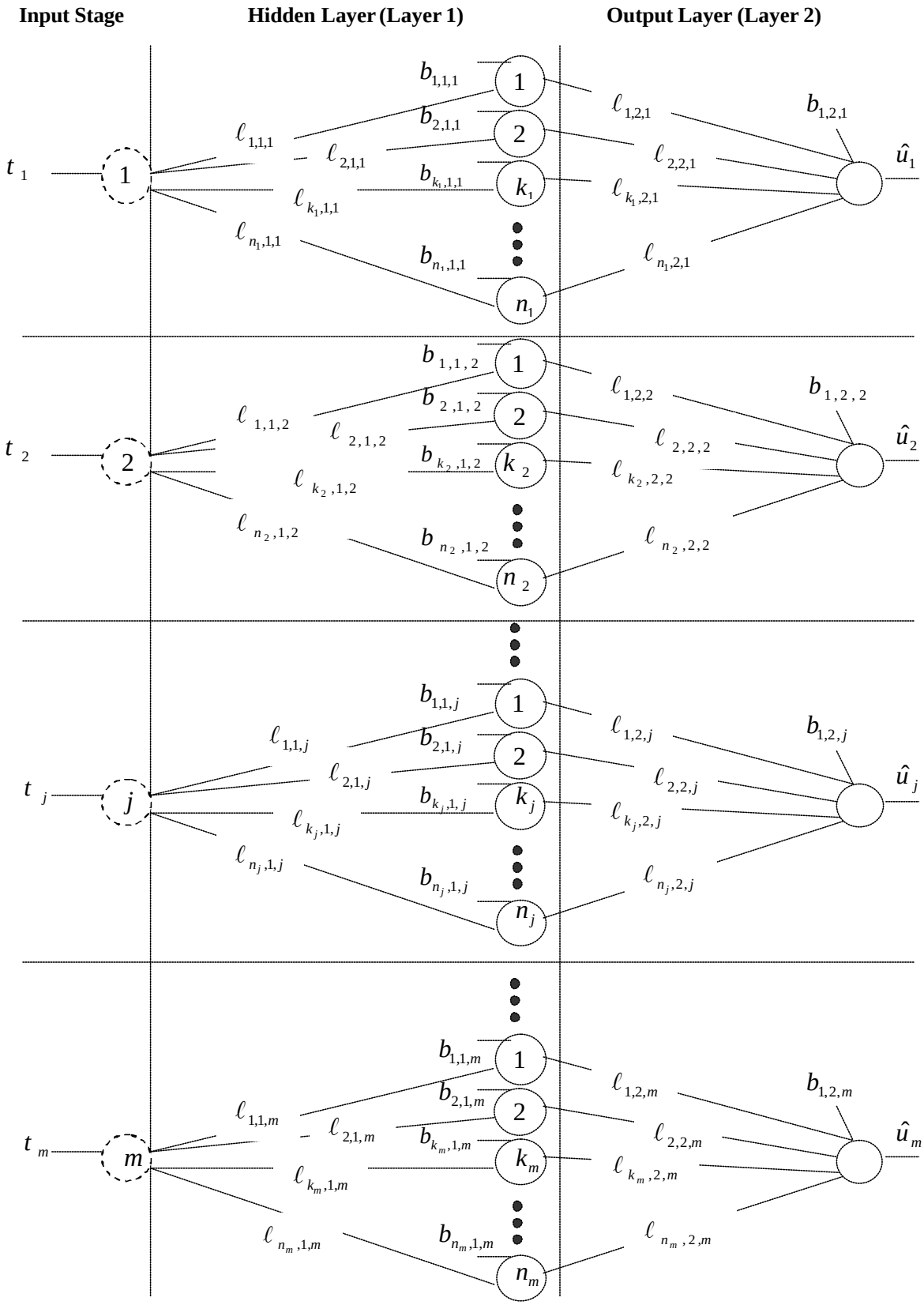


Figure 4.7.2: Schematic of the inner relationships of the NNPLS-1 model

subscript identifies the appropriate layer, and the third subscript identifies the appropriate latent variable.

Based on the above descriptions of the NNPLS inner and outer relationships, the unknown parameter vector can be explicitly defined, where the number of parameters in each set is indicated on the right:

$$\hat{\tilde{q}} = \left\{ \begin{array}{ll} & \text{\# of parameters} \\ \left[r_{1,1} \quad r_{2,1} \quad \cdots \quad r_{a,1} \quad \cdots \quad r_{p,1} \right. & \\ r_{1,2} \quad r_{2,2} \quad \cdots \quad r_{a,2} \quad \cdots \quad r_{p,2} \cdots & m \times p \\ r_{1,j} \quad r_{2,j} \quad \cdots \quad r_{a,j} \quad \cdots \quad r_{a,m} \cdots & \\ r_{p,1} \quad r_{p,2} \quad \cdots \quad r_{p,j} \quad \cdots \quad r_{p.m} & \\ \ell_{1,1,1} \quad \ell_{2,1,1} \quad \cdots \quad \ell_{k_1,1,1} \quad \cdots \quad \ell_{n_1,1,1} & \\ \ell_{1,1,2} \quad \ell_{2,1,2} \quad \cdots \quad \ell_{k_2,1,2} \quad \cdots \quad \ell_{n_2,1,2} \cdots & \sum_{j=1}^{m} n_j \\ \ell_{1,1,j} \quad \ell_{2,1,j} \quad \cdots \quad \ell_{k_j,1,j} \quad \cdots \quad \ell_{n_j,1,j} \cdots & \\ \ell_{1,1,m} \quad \ell_{2,1,m} \quad \cdots \quad \ell_{k_m,1,m} \quad \cdots \quad \ell_{n_m,1,m} & \\ b_{1,1,1} \quad b_{2,1,1} \quad \cdots \quad b_{k_1,1,1} \quad \cdots \quad b_{n_1,1,1} & \\ b_{1,1,2} \quad b_{2,1,2} \quad \cdots \quad b_{k_2,1,2} \quad \cdots \quad b_{n_2,1,2} \cdots & \sum_{j=1}^{m} n_j \\ b_{1,1,j} \quad b_{2,1,j} \quad \cdots \quad b_{k_j,1,j} \quad \cdots \quad b_{n_j,1,j} \cdots & \\ b_{1,1,m} \quad b_{2,1,m} \quad \cdots \quad b_{k_m,1,m} \quad \cdots \quad b_{n_m,1,m} & \\ \ell_{1,2,1} \quad \ell_{2,2,1} \quad \cdots \quad \ell_{k_1,2,1} \quad \cdots \quad \ell_{n_1,2,1} & \\ \ell_{1,2,2} \quad \ell_{2,2,2} \quad \cdots \quad \ell_{k_2,2,2} \quad \cdots \quad \ell_{n_2,2,2} \cdots & \sum_{j=1}^{m} n_j \\ \ell_{1,2,j} \quad \ell_{2,2,j} \quad \cdots \quad \ell_{k_j,2,j} \quad \cdots \quad \ell_{n_j,2,j} \cdots & \\ \ell_{1,2,m} \quad \ell_{2,2,m} \quad \cdots \quad \ell_{k_m,2,m} \quad \cdots \quad \ell_{n_m,2,m} & m \\ b_{1,2,1} \quad b_{1,2,2} \quad \cdots \quad b_{1,2,j} \quad \cdots \quad b_{1,2,m} & m \\ q_1 \quad q_2 \quad \cdots \quad q_j \quad \cdots \quad q_m \left. \right] & \end{array} \right.$$

The total number of parameters for the NNPLS model is given as the sum of the values specified for each set of parameters in the column above (n_j is the number of hidden neurons used in the mapping between t_j and $\hat{u}_j$):

$N = m(2+p)+3\sum_{j=1}^{m} n_j$. For each observation in the training data, $i = 1,...,n$, there will be a vector of partial derivatives $\frac{\partial \hat{y}_i}{\partial \hat{\vec{q}}}$ with dimensions $1 \times N$. The full Jacobian is then constructed from all of these vectors and its overall dimensions are $n \times N$.

To compute the prediction intervals, the Jacobian matrix must be obtained; thus, the derivatives with respect to all of the parameters in the set $\hat{\vec{q}}$, must be determined. Beginning with the output side of the model:

$\hat{y} = \sum_{j=1}^{m} \hat{u}_j q_j$, thus we have: $\frac{\partial \hat{y}_i}{\partial q_j} = \hat{u}_j$, and $\frac{\partial \hat{y}_i}{\partial \hat{u}_j} = q_j$.

Note that for the inferential version of the NNPLS model, $q_j = 1$, for $j = 1,...,m$.

To determine the derivatives of the output layer parameters, consider figure 4.7.3, which is a magnified diagram of the j^{th} output neuron of the model.

The input quantities to the j^{th} output neuron are the hidden layer outputs, from the j^{th} neural network, multiplied by the corresponding output layer weights, and the j^{th} neural network hidden layer bias. These quantities are summed at the input of the output neuron to calculate $I_{1,2,j}$ as shown in figure 4.7.3 and the output of the neuron is determined by ${}^{2}f_j$, which is the output neuron activation function. The output layer activation functions, ${}^{2}f_j$, for $j = 1,...,m$ are given by ${}^{2}f_j(x) = x$. The final output of the j^{th} output neuron is the corresponding output score, $\hat{u}_j$, which is the estimate of the input score t_j .

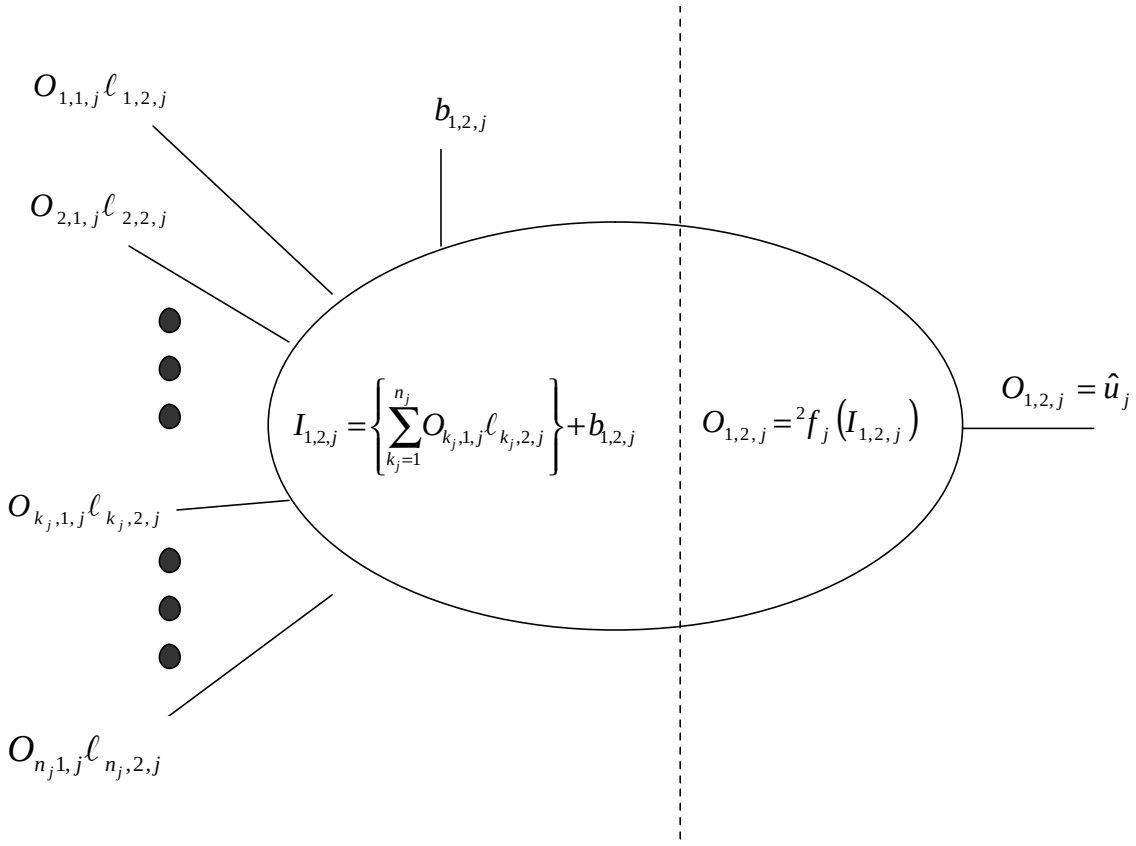


Figure 4.7.3: NNPLS output layer neuron

The partial derivatives with respect to the output neuron parameters are given by:

$$\frac{\partial \hat{y}_i}{\partial b_{1,2,j}} = \frac{\partial \hat{y}_i}{\partial \hat{u}_j} \cdot \frac{\partial \hat{u}_j}{\partial O_{1,2,j}} \cdot \frac{\partial O_{1,2,j}}{\partial I_{1,2,j}} \cdot \frac{\partial I_{1,2,j}}{\partial b_{1,2,j}}$$

$$\frac{\partial \hat{y}_i}{\partial \ell_{k_j,2,j}} = \frac{\partial \hat{y}_i}{\partial \hat{u}_j} \cdot \frac{\partial \hat{u}_j}{\partial O_{1,2,j}} \cdot \frac{\partial O_{1,2,j}}{\partial I_{1,2,j}} \cdot \frac{\partial I_{1,2,j}}{\partial \ell_{k_j,2,j}}$$

The intermediate derivatives can be determined from the equations shown in figure 4.7.3.

Since $\hat{u}_j = O_{1,2,j}$, then $\frac{\partial \hat{u}_j}{\partial O_{1,2,j}} = 1$.

Since $O_{1,2,j} = {}^2f_j\left(I_{1,2,j}\right) = I_{1,2,j}$, then $\frac{\partial O_{1,2,j}}{\partial I_{1,2,j}} = 1$.

Since $I_{1,2,j} = \left\{ \sum_{k_j=1}^{n_j} O_{k_j,1,j} \ell_{k_j,2,j} \right\} + b_{1,2,j}$, then $\frac{\partial I_{1,2,j}}{\partial O_{k_j,1,j}} = \ell_{k_j,2,j}$, $\frac{\partial I_{1,2,j}}{\partial b_{1,2,j}} = 1$, and

$\frac{\partial I_{1,2,j}}{\partial \ell_{k_j,2,j}} = O_{k_j,1,j}$.

Therefore:

$$\frac{\partial \hat{y}_i}{\partial b_{1,2,j}} = q_j \cdot 1 \cdot 1 \cdot 1 = 1$$

$$\frac{\partial \hat{y}_i}{\partial \ell_{k_j,2,j}} = q_j \cdot 1 \cdot 1 \cdot O_{k_j,1,j}$$

Recall that q_j is one for the inferential NNPLS model, $j = 1,...,m$ (m is the number of latent variables), $k_j = 1,...,n_j$ (n_j is the number of hidden neurons used in the mapping between t_j and $\hat{u}_j$). To determine the derivatives of the hidden layer parameters, consider figure 4.7.4, which is a magnified diagram of hidden neuron k_j. To determine the location of this neuron in the overall scheme of the model, refer to figure 4.7.2.

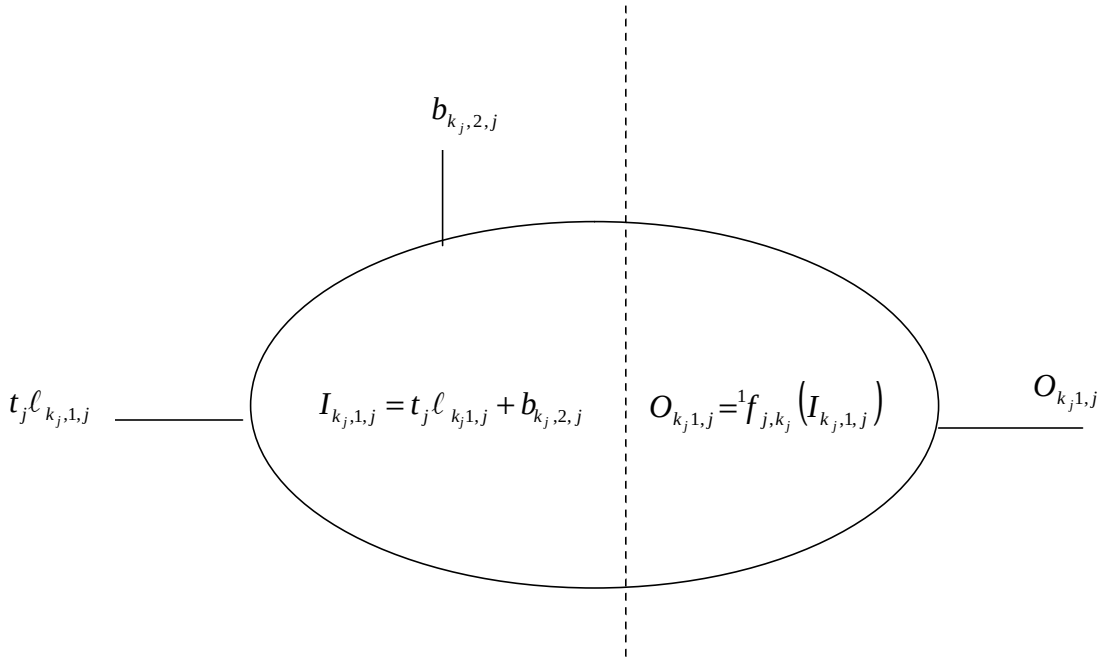


Figure 4.7.4: NNPLS hidden layer neuron k_j.

The input quantity is the sum of the product of the input score and the corresponding hidden layer weight, and the associated hidden layer bias. The output from the hidden layer neuron depends on the choice of activation function. The hidden layer activation functions are denoted by a prior superscript of 1, denoting layer 1 (the hidden layer). The first subscript identifies which input score vector model the neuron belongs to, and the second subscript identifies the specific hidden neuron within the specified score vector model. In this work all of the hidden layer activation functions are hyperbolic tangent functions; thus, the function notation can be simplified by utilizing a single prior superscript of 1, indicating the it is a hidden layer function, i.e.:

$${}^{1}f(z) = \tanh(z), \text{ and } \frac{\partial {}^{1}f}{\partial z} = 1 + z^2$$

For the other common choice of a logarithmic sigmoid activation function, see section 4.6 for the function and its derivative.

The partial derivatives with respect to the hidden neuron parameters are given by:

$$\frac{\partial \hat{y}_i}{\partial b_{k_j,1,j}} = \frac{\partial \hat{y}_i}{\partial \hat{u}_j} \cdot \frac{\partial \hat{u}_j}{\partial O_{1,2,j}} \cdot \frac{\partial O_{1,2,j}}{\partial I_{1,2,j}} \cdot \frac{\partial I_{1,2,j}}{\partial O_{k_j,1,j}} \cdot \frac{\partial O_{k_j,1,j}}{\partial I_{k_j,1,j}} \cdot \frac{\partial I_{k_j,1,j}}{\partial b_{k_j,1,j}}$$

$$\frac{\partial \hat{y}_i}{\partial b_{k_j,1,j}} = \frac{\partial \hat{y}_i}{\partial \hat{u}_j} \cdot \frac{\partial \hat{u}_j}{\partial O_{1,2,j}} \cdot \frac{\partial O_{1,2,j}}{\partial I_{1,2,j}} \cdot \frac{\partial I_{1,2,j}}{\partial O_{k_j,1,j}} \cdot \frac{\partial O_{k_j,1,j}}{\partial I_{k_j,1,j}} \cdot \frac{\partial I_{k_j,1,j}}{\partial \ell_{k_j,1,j}}$$

The first four arguments of the partial derivatives for the hidden layer parameters were defined previously during the determination of the partial derivatives with respect to the output layer neurons. The remaining arguments are defined below:

Since $O_{k_j 1,j} = {}^{1}f_{j,k_j}\left(I_{k_j,1,j}\right) = \tanh\left(I_{k_j,1,j}\right)$, then $\frac{\partial O_{k_j,1,j}}{\partial I_{k_j,1,j}} = \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right)$.

Since $I_{k_j,1,j} = t_j \ell_{k_j 1,j} + b_{k_j,2,j}$, then $\frac{\partial I_{k_j,1,j}}{\partial t_j} = \ell_{k_j 1,j}$, $\frac{\partial I_{k_j,1,j}}{\partial b_{k_j,1,j}} = 1$ and $\frac{\partial I_{k_j,1,j}}{\partial \ell_{k_j,1,j}} = t_j$.

Therefore:

$$\frac{\partial \hat{y}_i}{\partial b_{k_j,1,j}} = q_j \cdot 1 \cdot 1 \cdot \ell_{k_j,2,j} \cdot \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right) \cdot 1 = \ell_{k_j,2,j} \cdot \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right)$$

$$\frac{\partial \hat{y}_i}{\partial \ell_{k_j,1,j}} = q_j \cdot 1 \cdot 1 \cdot \ell_{k_j,2,j} \cdot \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right) \cdot t_j = \ell_{k_j,2,j} \cdot t_j \cdot \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right)$$

The final remaining partial derivatives to be evaluated are with respect to the transformation weights, $r_{a,j}$.

$$\frac{\partial \hat{y}_i}{\partial r_{a,j}} = \frac{\partial \hat{y}_i}{\partial \hat{u}_j} \cdot \frac{\partial \hat{u}_j}{\partial O_{1,2,j}} \cdot \frac{\partial O_{1,2,j}}{\partial I_{1,2,j}} \cdot \frac{\partial I_{1,2,j}}{\partial O_{k_j,1,j}} \cdot \frac{\partial O_{k_j,1,j}}{\partial I_{k_j,1,j}} \cdot \frac{\partial I_{k_j,1,j}}{\partial t_j} \cdot \frac{\partial t_j}{\partial r_{a,j}}$$

From the previous discussions following figure 4.7.1, the following can be obtained:

$$t_j = x_1 r_{1,j} + x_2 r_{2,j} + \cdots + x_a r_{a,j} + \cdots + x_p r_{p,j}$$

$$\frac{\partial t_j}{\partial r_{a,j}} = x_a$$

Therefore:

$$\frac{\partial \hat{y}_i}{\partial r_{a,j}} = q_j \cdot 1 \cdot 1 \cdot \ell_{k_j,2,j} \cdot \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right) \cdot \ell_{k_j,1,j} \cdot x_a = x_a \cdot \ell_{k_j,1,j} \cdot \ell_{k_j,2,j} \cdot \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right)$$

To summarize, all of the partial derivatives are given in table 4.7.1.

Define the $(1 \times N)$ vector of partial derivatives, where $N = m(2+p) + 3\sum_{j=1}^{m} n_j$, for the i^{th} observation/response as:

$$\frac{\partial \hat{y}_i}{\partial \hat{\vec{q}}} = \left[\frac{\partial \hat{y}_i}{\partial r_{a,j}} \quad \frac{\partial \hat{y}_i}{\partial \ell_{k_j,1,j}} \quad \frac{\partial \hat{y}_i}{\partial b_{k_j,1,j}} \quad \frac{\partial \hat{y}_i}{\partial \ell_{k_j,2,j}} \quad \frac{\partial \hat{y}_i}{\partial b_{1,2,j}} \quad \frac{\partial \hat{y}_i}{\partial q_j}\right]$$

Table 4.7.1: Definition of First Order Partial Derivatives for the NNPLS model based on an observation vector $\mathbf{x}_i = [x_1 \quad x_2 \quad \cdots \quad x_a \quad \cdots \quad x_p]$ and its scalar response y_i.

Partial Derivative	Subscript Definition	Total number of values per derivative
$\frac{\partial \hat{y}_i}{\partial r_{a,j}} = x_a \cdot \ell_{k_j,1,j} \cdot \ell_{k_j,2,j} \cdot \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right)$	$a = 1,\ldots,p$ $j = 1,\ldots,m$	$m \times p$
$\frac{\partial \hat{y}_i}{\partial \ell_{k_j,1,j}} = \ell_{k_j,2,j} \cdot t_j \cdot \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right)$	$j = 1,\ldots,m$ $k_j = 1,\ldots,n_j$	$\sum_{j=1}^{m} n_j$
$\frac{\partial \hat{y}_i}{\partial b_{k_j,1,j}} = \ell_{k_j,2,j} \cdot \left(1 + \left[\tanh\left(I_{k_j,1,j}\right)\right]^2\right)$	$j = 1,\ldots,m$ $k_j = 1,\ldots,n_j$	$\sum_{j=1}^{m} n_j$
$\frac{\partial \hat{y}_i}{\partial \ell_{k_j,2,j}} = O_{k_j,1,j}$	$j = 1,\ldots,m$ $k_j = 1,\ldots,n_j$	$\sum_{j=1}^{m} n_j$
$\frac{\partial \hat{y}_i}{\partial b_{1,2,j}} = 1$	$j = 1,\ldots,m$	m
$\frac{\partial \hat{y}_i}{\partial q_j} = \hat{u}_j$	$j = 1,\ldots,m$	m

The two derivative quantities required for the prediction interval computation are $\mathbf{f}_0$ (0 denotes derivatives based on a future observation) and $\mathbf{F}$ (based on the set of training observations).

$$\mathbf{F} = \frac{\partial \hat{\mathbf{y}}}{\partial \hat{\vec{q}}} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial \hat{\vec{q}}} \\ \frac{\partial \hat{y}_2}{\partial \hat{\vec{q}}} \\ \cdots \\ \frac{\partial \hat{y}_i}{\partial \hat{\vec{q}}} \end{bmatrix} \text{ and } \mathbf{f}_0^{\mathrm{T}} = \frac{\partial \hat{y}_0}{\partial \hat{\vec{q}}}$$

$\hat{\mathbf{y}} = [\hat{y}_1 \quad \hat{y}_2 \quad \cdots \quad \hat{y}_i \quad \cdots \quad \hat{y}_n]^T$ represents the full set of training responses. The derivative vector overall dimensions are: $\mathbf{f}_0 \rightarrow (N \times 1)$, and $\mathbf{F} \rightarrow (n \times N)$.

The overall approach is to calculate the full Jacobian matrix based on the training data and the resultant NNPLS model weight and bias parameters. An estimate of the variance of the error term due to model limitations is then obtained based on the training data, and the following equation:

$$s^2 = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$ [Tibshirani 1996]

For each new observation, $\mathbf{x}_0$, compute the NNPLS estimate $\hat{y}_0 = f(\mathbf{x}_0; \hat{q})$, the vector of partial derivatives, $\mathbf{f}_0$, and the $(100 - a) \times 100\%$ prediction interval from:

$$\hat{y}_0 \pm t_{a/2}^{n-p} \cdot s \sqrt{1 + \mathbf{f}_0^T (\mathbf{F}^T \mathbf{F})^{-1} \mathbf{f}_0}$$

4.8 VARIANCE AND BIAS OF LOCAL POLYNOMIAL REGRESSION

The derivation of the bias and variance properties of the local polynomial regression estimators provides the general results from which the properties can be obtained for the cases of kernel regression, and local linear regression, as well as higher order polynomials. Direct computation of the bias and variance is not possible because of the dependence on unknown quantities; however, asymptotic approximations are available that can be exploited to construct prediction intervals. Here, a brief presentation is provided for the general case. In the following section a detailed account of the bias and variance of the kernel regression estimator is provided, followed by an extension to local linear and local polynomial estimators.

(X, Y) is a generic member of the set of observations $(X_1, Y_1), \ldots, (X_n, Y_n)$ whose conditional mean and conditional variance are:

$m(x) = E(Y \mid X = x)$ and $n(x) = Var(Y \mid X = x)$

The general assumed model is:

$Y_i = m(X_i) + v^{1/2}(X_i)e_i, \qquad i = 1,...,n$.

Where $v(x)$ is finite and the e_i are mutually independent and identically distributed random variables with zero mean and unit variance. The e_i's are also independent of the X_i's.

The performance of an estimator is conventionally assessed via its mean squared error (MSE):

$MSE(x) = E\left[\{\hat{m}(x) - m(x)\}^2 \mid \mathbf{X}\right]$, where $\mathbf{X} = (X_1,..., X_n)$.

The bias-variance decomposition of the MSE can also be written:

$MSE(x) = \left[E\{\hat{m}(x) \mid \mathbf{X}\} - m(x)\right]^2 + Var\{\hat{m}(x) \mid \mathbf{X}\}$

The first term is the squared bias while the second term is the variance. The derivation of the computations for the squared bias and variance are presented below:

Recall from section 3.8.5:

$\hat{b} = (\mathbf{X}_x^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}_x^T \mathbf{W} \mathbf{y}$ is the solution of the weighted least squares problem:

$(\mathbf{y} - \mathbf{X}_x \boldsymbol{ß})^{\mathrm{T}} \mathbf{W} (\mathbf{y} - \mathbf{X}_x \boldsymbol{ß})$ which was derived from the Taylor series expansion of the conditional mean of the LPR estimator.

$$m(x) \approx m(x_0) + m'(x_0)(x - x_0) + \frac{m''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{m^{(p)}(x_0)}{p!}(x - x_0)^p$$

Using the relation $m^k(x) = k!\, b_k$, where $m^k(x)$ is the k^{th} derivative of $m(x)$, this can be written as: $m(x) \approx b_0 + b_1(x - x_0) + \frac{b_2}{2!}(x - x_0)^2 + \cdots + \frac{b_p}{p!}(x - x_0)^p$

Thus the solution to the weighted least squares problem results in estimates of $m(x)$ and its p derivatives at the point x_0: $\hat{b}_j = \left[\hat{b}_0,...,\hat{b}_p\right]$. For the purpose of obtaining the expected value at x_0, $m(x_0)$, the following approximation is used:

$\hat{m}(x) = \hat{b_0}$. Defining $\mathbf{e}_1$ as a vector of zeros containing a 1 in its first element, the following is obtained:

$$\hat{m}(x) = \hat{b_0} = \mathbf{e}_1(\mathbf{X}_x^T\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{X}_x^T\mathbf{W}_x\mathbf{y}$$

For the univariate case the terms are defined as:

$$\mathbf{X}_x = \begin{bmatrix} 1 & (X_1 - x_0) & \cdot & \cdot & \cdot & (X_1 - x_0)^p \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ 1 & (X_n - x_0) & \cdot & \cdot & \cdot & (X_n - x_0)^p \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} Y_1 \\ \cdot \\ \cdot \\ \cdot \\ Y_n \end{bmatrix}$$

$$\mathbf{W} = diag\{K_h(X_i - x_0)\}$$

The length of the vector $\mathbf{e}_1$ depends on the order of the local polynomial used in the weighted least squares regression, and the number of predictor variables for the multivariate case.

To derive the bias of the estimator, first consider its expected value:

$$E(\hat{m}(x) \mid \mathbf{X}) = \mathbf{e}_1^T(\mathbf{X}_x^T\mathbf{W}\mathbf{X}_x)^{-1}\mathbf{X}_x^T\mathbf{W}\mathbf{m}$$

where $\mathbf{m} = \{m(X_1),...,m(X_n)\}^T$.

A Taylor series expansion provides:

$$\mathbf{m} = \mathbf{X}_x\begin{bmatrix} m(x) \\ D_m(x) \end{bmatrix} + \frac{1}{2}Q_m(x) + R_m(x)$$

D_m is the vector of first-order partial derivatives.. Q_m contains the second order terms based on the Hessian matrix H_m, and R_m is the vector of Taylor series remainder terms. To show the expansion in detail, consider the case for local linear regression, i.e. $p=1$.

$$\mathbf{m}_{(p=1)} = \begin{bmatrix} 1 & (X_1 - x) \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & (X_n - x) \end{bmatrix} \cdot \begin{bmatrix} m(x) \\ D_m(x) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} (X_1 - x)^T H_m(x)(X_1 - x) \\ \cdot \\ \cdot \\ \cdot \\ (X_n - x)^T H_m(x)(X_n - x) \end{bmatrix} + R_m(x)$$

For $p=1$, D_m is $d \times 1$, H_m is $d \times d$, and Q_m, R_m are $n \times 1$.

The computation of $\mathbf{m}$ thus depends on the order of the local polynomial. Returning to the expected value computation:

$$E(\hat{m}(x) \mid \mathbf{X}) = \mathbf{e}_1^T (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x^T \mathbf{W}_x \left[\mathbf{X}_x \begin{bmatrix} m(x) \\ D_m(x) \end{bmatrix} + \frac{1}{2} Q_m(x) + R_m(x) \right]$$

Using: $\mathbf{e}_1^T (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x \begin{bmatrix} m(x) \\ D_m(x) \end{bmatrix} = \mathbf{e}_1^T \begin{bmatrix} m(x) \\ D_m(x) \end{bmatrix} = m(x)$

$$E(\hat{m}(x) \mid \mathbf{X}) = m(x) + \mathbf{e}_1^T (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x^T \mathbf{W}_x \left[\frac{1}{2} Q_m(x) + R_m(x) \right]$$

The resulting scalar $\mathbf{e}_1^T (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x^T \mathbf{W}_x R_m(x)$, note that $R_m(x)$ is a $n \times 1$ vector, is negligible compared to the terms arising from $Q_m(x)$.

Thus, the bias estimate can be written as:

$$E\left\{\hat{m}(x \mid \mathbf{X}) - m(x)\right\} = E(\hat{m}(x) \mid \mathbf{X}) - m(x) = \frac{1}{2} \mathbf{e}_1^T (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x^T \mathbf{W}_x Q_m(x)$$

where $Q_m = \left[(X_1 - x)^T H_m(x)(X_1 - x) \quad \cdots \quad (X_n - x)^T H_m(x)(X_n - x) \right]^T$ is an $n \times 1$ vector.

This equation is not directly computable due to the unknown quantity $Q_m(x)$ which requires the second derivatives of the function itself.

The variance term of the MSE can be written as

$$Var(\hat{m} \mid \mathbf{X}) = (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1} (\mathbf{X}_x^T \Sigma \mathbf{X}_x) (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1}$$

where $\Sigma = diag\{K_h^2(X_i - x_0) n(X_i)\}$.

From the above bias and variance equations, first-order asymptotic approximations can be derived. The derivations for the case of LPR are available in Fan & Gijbels [1996].

Assume that $f(x_0) > 0$, and that $f(\cdot)$, $m^{(p+1)}(\cdot)$, and $n(\cdot)$ are continuous in a neighborhood of x_0. Assume that $h \to 0$ and $nh \to \infty$.

$$Var\{\hat{m}_k(x_0) \mid \mathbf{X}\} = \mathbf{e}_{k+1}^T \mathbf{S}^{-1} \mathbf{S}^* \mathbf{S}^{-1} \mathbf{e}_{k+1} \frac{k!^2 n(x_0)}{f(x_0) n h^{1+2k}} + o_P\left(\frac{1}{nh^{1+2k}}\right)$$

For $p - k$ odd:

$$Bias\{\hat{m}_k(x_0) \mid \mathbf{X}\} = \mathbf{e}_{k+1}^T \mathbf{S}^{-1} c_p \frac{k!}{(p+1)!} m^{(p+1)}(x_0) h^{p+1-k} + o_P(h^{p+1-k})$$

For $p - k$ even, provided that $f'(\cdot)$ and $m^{(p+2)}(\cdot)$ are continuous in a neighborhood of x_0 and $nh^3 \to \infty$:

$$Bias\{\hat{m}_k(x_0) \mid \mathbf{X}\} = \mathbf{e}_{k+1}^T \mathbf{S}^{-1} \tilde{c}_p \frac{k!}{(p+2)!} \left\{ m^{(p+2)}(x_0) + (p+2) m^{(p+1)}(x_0) \frac{f'(x_0)}{f(x_0)} \right\} h^{p+2-k} + o_P(h^{p+2-k})$$

$f(x_0)$ is the common density of the predictor variables X_i, k identifies the estimate (i.e., $k = 0$ to estimate $m(x_0)$, $k = 1$ to estimate $m'(x_0)$, p is the order of the local polynomial. Also:

$\mathbf{e}_{k+1}^T = (0,...,0,1,0,...,0)^T$, where the 1 is in the $(k+1)^{\text{th}}$ position.

$c_p = (m_{p+1},..., m_{2p+1})^T \quad \tilde{c}_p = (m_{p+2},..., m_{2p+2})^T$

$\mathbf{S} = (m_{j+l})_{0 \le j,l \le p} \qquad \mathbf{S}^* = (v_{j+l})_{0 \le j,l \le p}$

$v_j = \int u^j K^2(u) du \qquad m_j = \int u^j K(u) du$

The additional term in the bias for odd order polynomial fits illustrates the bias reduction when moving from an even ordered polynomial to the subsequent odd ordered polynomial. This phenomenon has been presented by various researchers as evidence of the superiority of odd-ordered polynomials; however, it is important to consider the specific function at hand, and depending on the values in the additional term it is possible for an even ordered polynomial to outperform an odd ordered polynomial, though in most cases the opposite will be true.

The asymptotic bias and variance expressions for the estimator $\hat{m}_0(x_0) = \hat{b}_0$ are provided in table 4.8.1 for the cases of $p = 0$ (Nadaraya-Watson) and $p = 1$ (local linear).

Constant-approximation type nonparametric models are susceptible to estimation problems near the boundaries of the observation intervals. Consider a predictor variable x and a response variable y. At the boundary of the observation interval, the local averaging process becomes asymmetric because half of the weights are undefined and outside the boundary creating a bias related to the tangential behavior at the boundary. In addition, the minimum and maximum points of a nonparametric regression curve will be reduced to some extent due to the averaging process of the local neighborhood of points [Härdle 1990]. This argument regarding the boundary effects of kernel regression was also stated by Fan and Gijbels [1992b]. Ruppert and Wand [1994] provide a word of caution for this argument, stating that, near the boundary, the parameters of the local model are no longer asymptotically orthogonal as they are in the interior. They continue to mention that for finite samples if the curvature, $f'(0)$, is small relative to the variance, $n(0)$, then for x near the boundary the Nadaraya-Watson estimate could be considerably more accurate than the local linear estimate.

Note that the Nadaraya-Watson estimator is not the only approach for local constant approximation, though it is the most widely used. Other versions of local constant estimators such as the Gasser-Müller [Gasser and Müller 1979] are available which have

Table 4.8.1: Pointwise asymptotic bias and variance of kernel regression estimators at interior points of the support of the design density [Fan 1992a]

Method	Bias	Variance
Nadaraya-Watson	$\left(\frac{1}{2}m''(x)+\frac{m'(x)f'(x)}{f(x)}\right)h^2\int_{-\infty}^{\infty}u^2K(u)du$	$V_n=\frac{n(x)}{f(x)nh}\int_{-\infty}^{\infty}K^2(u)du$
Local Linear	$\left(\frac{1}{2}m''(x)\right)h^2\int_{-\infty}^{\infty}u^2K(u)du$	$V_n=\frac{n(x)}{f(x)nh}\int_{-\infty}^{\infty}K^2(u)du$

improved asymptotic bias and variance properties, but not as improved as that obtained via local linear estimators. The slope parameter of the local linear estimator allows for the bias reduction with respect to the Nadaraya-Watson estimator.

The Nadaraya-Watson estimator has zero minimax efficiency. The Nadaraya-Watson estimator may exhibit high bias at the boundaries of the support and while methods are available for corrections near the boundary, they are not as efficient as the automatic boundary correction of the local linear model. In general, local polynomial regression does not share the boundary effects of the Nadaraya-Watson estimator and the bias at the boundary stays automatically of the same order as in the interior. Cheng, Fan, and Marron [1993] provide that the local linear estimator is efficient in correcting boundary bias in an asymptotic minimax sense. Additional work on the minimax efficiency of local linear estimators was completed by Fan [1992a, 1993] and extended to the general case of local polynomial estimators Fan et. al. [1995b]. The result of the studies on the asymptotic minimax efficiencies of the various smoothers provide that local polynomial estimators are nearly best linear estimators, and the efficiency of the local linear fit may be as high as 100% relative to that of 0 for the Nadaraya-Watson estimator.

4.9 VARIANCE AND BIAS ESTIMATION IN KERNEL REGRESSION

In this section a detailed derivation is provided for the asymptotic approximations of the bias and variance terms of the MSE for KR. The kernel estimator is given by:

$$\hat{f}(x;h) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

A common substitution can be made:

$$\hat{f}(x;h) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - X_i), \text{ where} \qquad K_h(u) = \frac{1}{h}K\left(\frac{u}{h}\right)$$

The mean squared error between the estimator $\hat{q}$ and the target parameter q is given by:

$$MSE(\hat{q}) = E\left[(\hat{q} - q)\right]^2$$

Decomposing into variance and squared bias:

$$MSE(\hat{q}) = Var(\hat{q}) + (E\left[\hat{q}\right] - q)^2$$

Since the estimator is $\hat{q} = \hat{f}(x;h)$, we have:

$$MSE\left\{\hat{f}(x;h)\right\} = Var\left[\hat{f}(x;h)\right] + (E\left[\hat{f}(x;h)\right] - f(x))^2$$

Analyzing the terms separately beginning with the bias term provides:

$$E\left[\hat{f}(x;h)\right] = E\left[K_h(x - X)\right] = \int K_h(x - y)\, f(y) dy$$

Using the convolution notation $(f * g)(x) = \int f(x - y) g(y) dy$ allows:

$$E\left[\hat{f}(x;h)\right] = (K_h * f)(x)$$

Such that the squared bias term becomes:

$$(E\left[\hat{f}(x;h)\right] - f(x))^2 = \left[(K_h * f)(x) - f(x)\right]^2$$

Consider the variance term:

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{n}\left[E\left[\hat{f}^2(x;h)\right] - \left(E\left[\hat{f}(x;h)\right]\right)^2\right]$$

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{n}\left[E\left[K_h^2(x - X_i)\right] - \left(E\left[\hat{f}(x;h)\right]\right)^2\right]$$

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{n}\left[\int K_h^2(x - y)\, f(y) dy - \left(E\left[\hat{f}(x;h)\right]\right)^2\right]$$

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{n}\left[\left(K_h^2 * f\right)(x) - \left(K_h * f\right)^2(x)\right]$$

Thus the MSE can be rewritten:

$$MSE\left\{\hat{f}(x;h)\right\} = \frac{1}{n}\left[\left(K_h^2 * f\right)(x) - \left(K_h * f\right)^2(x)\right] + \left[\left(K_h * f\right)(x) - f(x)\right]^2$$

The mean squared error is derived at a point x, to evaluate the error between $\hat{f}(x;h)$ and $f(x)$. To evaluate the error globally consider the integrated squared error (ISE) criterion to quantify the error between $\hat{f}(\cdot;h)$ and f.

$$ISE\left\{\hat{f}(\cdot;h)\right\} = \int\left\{\hat{f}(x;h) - f(x)\right\}^2 dx$$

The ISE criterion considers the data set at hand. To account for other possible data sets from the density f it is more appropriate to evaluate the expected value of the ISE, the mean integrated squared error $(MISE)$:

$$MISE\left\{\hat{f}(\cdot;h)\right\} = E\left[ISE\left\{\hat{f}(\cdot;h)\right\}\right] = E\int\left[\hat{f}(x;h) - f(x)\right]^2 dx = \int E\left[\hat{f}(x;h) - f(x)\right]^2 dx = \int MSE\left\{\hat{f}(x;h)\right\}dx$$

Substituting the expression for MSE yields:

$$MISE\left\{\hat{f}(\cdot,h)\right\} = \frac{1}{n}\int\left[\left(K_h^2 * f\right)(x) - \left(K_h * f\right)^2(x)\right]dx + \int\left[\left(K_h * f\right)(x) - f(x)\right]^2 dx$$

Manipulation of the above expression yields:

$$MISE\left\{\hat{f}(\cdot,h)\right\} = \frac{1}{nh}\int K^2(x)dx + \left(1 - \frac{1}{n}\right)\int\left(K_h * f\right)^2(x)dx - 2\int\left(K_h * f\right)(x)f(x)dx + \int f(x)^2 dx$$

This expression depends on the bandwidth in a complicated way. To simplify, consider asymptotic approximations (large sample approximations) for the MSE.

Asymptotic approximations are based on the following assumptions:

1. f'' is continuous, square integrable, and ultimately monotone (i.e. one that is monotone over both $(-\infty,-M)\ \&\ (M,\infty)$ for $M > 0$).
2. The bandwidth h depends on n and is a non-random sequence of positive numbers. This dependence is suppressed in the asymptotic approximation derivations. It is assumed that $h \to 0$ as $n \to \infty$, though the rate of decrease for h lags the rate of increase for n.

$\lim_{n\to\infty}(h) = 0$ and $\lim_{n\to\infty}(nh) = \infty$

3. K is a bounded probability density function having a finite 4th moment and is symmetric about the origin.

The asymptotic approximations are derived for the MSE expression beginning with the bias term:

$$E\left[\hat{f}(x;h)\right] = \int K\left(\frac{x-y}{h}\right) f(y)dy$$

Using $z = (x-y)/h$, we can rewrite as:

$$E\left[\hat{f}(x;h)\right] = \int K(z)\, f(x-hz)dz$$

Using the Taylor series about x of $f(x-hz) = f(x) - hzf'(x) + \frac{1}{2}h^2z^2f''(x) + o(h^2)$, we can rewrite as:

$$E\left[\hat{f}(x;h)\right] = \int K(z)\left[\, f(x) - hzf'(x) + \frac{1}{2}h^2z^2f''(x) + o(h^2)\right]dz$$

$$E\left[\hat{f}(x;h)\right] = \int K(z)f(x)dz - \int K(z)hzf'(x)dz + \int K(z)\frac{1}{2}h^2z^2f''(x)dz + o(h^2)$$

Assumption 3 provides that:

$\int K(z)dz = 1$, $\int zK(z)dz = 0$, and $\int z^2K(z)dz < \infty$

Thus, we can rewrite as:

$$E\left[\hat{f}(x;h)\right] - f(x) = \frac{1}{2}h^2f''(x)\int z^2K(z)dz + o(h^2)$$

Using $m_2(K) = \int z^2K(z)dz$, we can rewrite as:

$$E\left[\hat{f}(x;h)\right] - f(x) = \frac{1}{2}h^2m_2(K)f''(x) + o(h^2)$$

This is the expression that will be used in the asymptotic approximation. Due to the dependence on the second derivative, the bias is large when $|f''(x)|$ is large, which occurs in regions of high curvature. Thus, the estimator $\hat{f}(x;h)$ has a tendency to smooth out the "peaks" and "valleys" resulting in an increased bias in these regions.

Consider the variance term:

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{n}\left[\int K_h^2(x-y)\,f(y)dy - \left(E\left[\hat{f}(x;h)\right]\right)^2\right]$$

Using the change in variables as above:

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{nh}\int\left(K\left(\frac{x-y}{h}\right)\right)^2 f(y)dy - \frac{1}{n}\left(E\left[\hat{f}(x;h)\right]\right)^2$$

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{nh}\int\left(K(z)\right)^2 f(x-hz)dz - \frac{1}{n}\left(E\left[\hat{f}(x;h)\right]\right)^2$$

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{nh}\int\left(K(z)\right)^2 f(x-hz)dz - \frac{1}{n}\left(f(x) + \frac{1}{2}h^2 f''(x)\int z^2 K(z)dz\right)^2 + o(h^2)$$

Again using the third assumption, the variance expression can be reduced to:

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{nh}f(x)\int\left(K(z)\right)^2 dz + o\left(\frac{1}{nh}\right)$$

Using $R(K) = \int K^2(z)dz$, we can rewrite the variance expression as:

$$Var\left\{\hat{f}(x;h)\right\} = \frac{1}{nh}R(K)\,f(x) + o\left(\frac{1}{nh}\right)$$

Since the variance is of order $1/nh$, assumption 2 provides that the variance converges to zero. The asymptotic approximation for MSE can now be written as:

$$MSE\left\{\hat{f}(x;h)\right\} = \frac{1}{nh}R(K)\,f(x) + \frac{1}{4}h^4 m_2(K)^2 f''(x)^2 + o\left\{\left(\frac{1}{nh}\right) + h^4\right\}$$

The $MISE$ is then:

$$MISE\left\{\hat{f}(\cdot;h)\right\}=\frac{1}{nh}R(K)+\frac{1}{4}h^4 m_2(K)^2 R(f'')+o\left\{\left(\frac{1}{nh}\right)+h^4\right\}$$

From this we can define the Asymptotic $MISE$ ($AMISE$) which provides a useful large sample approximation to the $MISE$.

$$AMISE\left\{\hat{f}(\cdot;h)\right\}=\frac{1}{nh}R(K)+\frac{1}{4}h^4 m_2(K)^2 R(f'')$$

Viewing the $AMISE$ expression, the bias variance trade-off becomes evident. The leading term depends on $1/nh$, whereas the bias term depends on h^4. For very small bandwidth values, the bias of the estimator is near zero; however, the variance is relatively high. This condition would result in an estimator with a very "spiky" profile, and repeated sampling would alter the location of these spikes. On the other hand, large bandwidth values result in an estimator with a smooth profile, exhibiting very little to zero variance. This condition is oversmoothed due to the excessive introduction of bias, and the data are essentially ignored.

Differentiating the $AMISE$ expression, and setting equal to zero, results in a closed form expression for the optimal bandwidth:

$$h_{AMISE}=\left[\frac{R(K)}{m_2(K)^2 R(f'')n}\right]^{1/5}$$

The function $R(f'')$ measures the total curvature of f, thus for densities of little curvature, a large optimal bandwidth is called for, and for high curvature, a small optimal bandwidth is appropriate. $R(f'')$ cannot be computed directly in practical situations, though several rules for selecting the bandwidth based on estimates of this function are available.

Substituting the optimal bandwidth into the *AMISE* equation yields an expression for the minimum *AMISE* for estimation of f with kernel K:

$$\inf_{h>0} AMISE\left\{\hat{f}(\cdot;h)\right\} = \frac{5}{4n^{-4/5}}\left\{m_2(K)^2 R(K)^4 R(f'')\right\}^{1/5}$$

Thus, the optimal rate of convergence for the kernel estimator is $n^{-4/5}$.

Keep in mind that the *AMISE* is only a large sample approximation, and if one wants to analyze the exact finite sample performance of the estimator for a given f and K, then one would compute the *MISE*. Exact *MISE* calculations can be achieved for the normal kernel as well as linear combinations of normal densities, i.e. normal mixture densities.

This section provided a detailed examination of the derivation of the asymptotic approximations for computing the MSE of the KR estimator. These ideas are extended to LL and LPR in the following section.

4.9.1 Extensions to Local Linear and Local Polynomial Regression

Consider the random design:

$$Y_i = m(X_i) + v^{1/2}(X_i)e_i, \qquad i = 1,...,n$$

Assumptions:

1. m'' and n are both continuous on [0,1]
2. K is symmetric about zero and is supported on [-1,1
3. The bandwidth $h = h_n$ is a sequence satisfying that $h \to 0$ and $nh \to \infty$ as $n \to \infty$.
4. The point x at which the estimation is taking place satisfies $h < x < 1-h$ for all $n \geq n_0$ where n_0 is fixed.
5. f' is continuous

The results for the large sample approximations for the linear case are shown below. The derivations are covered in Wand and Jones [1995]:

The conditional bias is given by:

$$E\{\hat{m}(x; p=1, h) - m(x) \mid X_1, ..., X_n\} = \frac{1}{2} h^2 m''(x) m_2(K) + o_P(h^2)$$

The variance is given by:

$$Var\{\hat{m}(x; p=1; h) \mid X_1, ..., X_n\} = \{(nh)^{-1} R(K) / f(x)\} v(x) + o_P\{(nh)^{-1}\}$$

Extending these results to the general polynomial case, we rely on assumptions 1-4 above, and additionally require that $m^{(p+2)}(x)$ is continuous. Define the following notation: $\mathbf{N}_p$ is the $(p+1)\times(p+1)$ matrix whose (i, j) entry is $m_{i+j-2}(K)$, where $m_l(K) = \int_{-1}^{1} z^l K(z) dz$. $\mathbf{M}_p(u)$ is the same as $\mathbf{N}_p$, but with the first column replaced by $(1, u, ..., u^p)^T$. The kernel for the general case is: $K_p(u) = \left\{ \frac{|\mathbf{M}_p(u)|}{|\mathbf{N}_p|} \right\} K(u)$, is a $(p+1)$ order kernel when p is odd, and a $(p+2)$ order kernel when p is even, since it can be shown that $K_p = K_{p+1}$ for even p.

The conditional bias for the general case is given by:

For odd p:

$$E\{\hat{m}(x; p; h) - m(x) \mid X_1, ..., X_n\} = \frac{1}{(p+1)!} h^{p+1} m^{p+1}(x) m_{p+1}(K_p) + o_P(h^{p+1})$$

For even p:

$$E\{\hat{m}(x; p; h) - m(x) \mid X_1, ..., X_n\} = h^{p+2} \left\{ \frac{1}{(p+1)!} m^{p+1}(x) \frac{f'(x)}{f(x)} + \frac{1}{(p+2)!} m^{p+2}(x) \right\} m_{p+2}(K_p) + o_P(h^{p+2})$$

The variance is given by:

$$Var\{\hat{m}(x;p;h) \mid X_1,...,X_n\} = \{(nh)^{-1} R(K_{(p)})/\ f(x)\}v(x) + o_P\{(nh)^{-1}\}$$

For the general case, the degree of the polynomial determines the order of the bias of $\hat{m}(x;p;h)$. Note that the bias expression for odd p is a simpler expression, especially in its dependence on $\hat{m}(x;p;h)$. Wand and Jones show that for sufficiently smooth regression functions, the asymptotic performance of $\hat{m}(\cdot;p;h)$ improves for higher values of p; however, the variance of the estimator becomes large for these higher degree polynomials. Including the considerations that odd degree polynomials have attractive bias and boundary properties, Wand and Jones suggest the use of either $p=1$ or $p=3$.

Extending to the multivariate case leads to the notational changes for the random design case of X_i to $\mathbf{X}_i$, and the density, f, now being d-variate. To simplify, consider only the case for $p=1$. The fitted polynomial is of the form: $b_0 + \mathbf{ß}_1^T \mathbf{x}_i$ where b_0 is the scalar intercept and $\mathbf{ß}_1 = (b_{11},...,b_{1d})^T$. $\mathbf{H}$ is an appropriate bandwidth matrix, and K is a multivariate kernel function satisfying:

K is a bounded, compactly supported d-variate kernel satisfying: $\int K(\mathbf{z})d\mathbf{z} = 1$, $\int \mathbf{z}K(\mathbf{z})d\mathbf{z} = 0$, and , where $m_2(K) = \int z_i^2 K(\mathbf{z})d\mathbf{z} = m_2(K)\mathbf{I}$.

The multivariate local linear kernel estimator is given by:

$$\hat{m}(\mathbf{x};p=1;\mathbf{H}) = \mathbf{e}_1^T (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x^T \mathbf{W}_x \mathbf{Y}$$

where: $\mathbf{X}_x = \begin{bmatrix} 1 & (\mathbf{X}_1 - \mathbf{x})^T \\ . & . \\ . & . \\ . & . \\ 1 & (\mathbf{X}_n - \mathbf{x})^T \end{bmatrix}$, and $\mathbf{W}_x = diag\{K_{\mathbf{H}}(\mathbf{X}_1 - \mathbf{x}),...,K_{\mathbf{H}}(\mathbf{X}_n - \mathbf{x})\}$

Assume that $\mathbf{H}$ satisfies:

$\mathbf{H} = \mathbf{H}_n$ is a sequence of bandwidth matrices such that $n^{-1}|\mathbf{H}|^{-1/2}$ and all entries of $\mathbf{H}$ approach zero as $n \to \infty$. Additionally, the ratio of the largest and smallest eigenvalues of $\mathbf{H}$ is bounded for all n.

Assume that $f(x)$, $v(x)$, and all entries of the Hessian matrix $H_m(\mathbf{x})$ are continuous. The conditional bias, and variance are then:

$$E\{\hat{m}(\mathbf{x}; p=1; h) - m(\mathbf{x}) \mid \mathbf{X}_1,...,\mathbf{X}_n\} = \frac{1}{2} m_2(K) tr\{\mathbf{H} H_m(\mathbf{x})\} + o_P\{tr(\mathbf{H})\}$$

$$Var\{\hat{m}(\mathbf{x}; p=1; \mathbf{H}) \mid \mathbf{X}_1,...,\mathbf{X}_n\} = \frac{1}{n}|\mathbf{H}|^{-1/2} \frac{R(K)v(\mathbf{x})}{f(\mathbf{x})} + o\left\{\frac{1}{n|H|^{-1/2}}\right\}$$

4.9.2 Applied Prediction Intervals for Local Polynomial Regression Estimators

The previous 2 sections provided detailed information regarding the derivation of asymptotic approximations of the bias and variance of LPR estimators. The variance of the local polynomial regression estimator is given by:

$$Var\{\hat{m}_0(\mathbf{x}_0) \mid \mathbf{X}\} = \mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}(\mathbf{X}_x^{\mathbf{T}}\Sigma\mathbf{X}_x)(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{e}_1$$

Where: $\Sigma = diag\{K_h^2(X_i - x_0)n(X_i)\}$, and the vector $\mathbf{e}_1^{\mathbf{T}}$ is defined based on the order of the local approximation: for $p=0$, $\mathbf{e}_1^{\mathbf{T}} = 1$, for $p=1$, $\mathbf{e}_1^{\mathbf{T}} = [1 \quad 0 \quad \cdots \quad 0_d]$ the $[1 \times (d+1)]$ vector. d is the number of variables in $\mathbf{X}$

Using the following relation:

$\Sigma = \mathbf{W}_x\mathbf{V}\mathbf{W}_x$, where $\mathbf{V} = diag\{n(X_1),...,n(X_n)\}$, and $v(X_i) = \text{var}(Y \mid X = X_1)$.

The variance can be rewritten as:

$$Var\{\hat{m}_0(\mathbf{x}_0) \mid \mathbf{X}\} = \mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{V}\mathbf{W}_x\mathbf{X}_x(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{e}_1$$

This equation can be used directly based on estimations of the conditional variances of the response, Y.

The bias of the estimator is given by (from section 4.8):

$$E\{\hat{m}(x_0) - m(x_0) \mid X_1,\ldots, X_n\} = \frac{1}{2}\mathbf{e}_1^T(\mathbf{X}_x^T\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{X}_x^T\mathbf{W}_x\mathbf{Q}_m(x_0)$$

For the case of a local linear model:

$\mathbf{Q}_m(x) = [(X_1 - x)^T \mathrm{H}_m(x)(X_1 - x),\ldots,(X_n - x)^T \mathrm{H}_m(x)(X_n - x)]^T$ dimensions $(n\times 1)$

$\mathrm{H}_m(x)$ is the $d\times d$ Hessian matrix of $m(x)$, where d is the dimension of the input space. The difficulty in the bias approximation is that it requires the computation of the Hessian matrix of the unknown function $m(x)$. Thus, approximations for the bias term are required. An approximation for the bias term can be computed based on a set of data sampled independent of the data in the training matrix, but from the same distribution. An additional assumption is that the data provide a true representation of the function being modeled, $m(x)$.

In this work, the contribution of the bias to the uncertainty was computed based on the training data as well as an independent set of validation data:

$Bias^2 = \frac{1}{r}\sum_{i=1}^{r}\{m(\mathbf{x}_i) - \hat{m}(\mathbf{x}_i)\}^2$, where $\mathbf{x}_i$ represents the i^{th} observation of the combined set of training and validation data, $i = 1,\ldots,r$ (r is the total number of training and validation observations). The bias estimate is based on both the training and validation data so that an adequate estimate of bias can still be obtained for bandwidth values that are too small, resulting in an overfit model. An overfit model will report a minimal bias estimate for the training data, while the same estimate on an independent set of validation error will be respectively much larger. Because the intent is to apply the prediction interval methodologies to independent sets of data, it is imperative that an adequate bias estimate is obtained that represents the expected results for all representative data, not just the training data. The use of this bias approximation is based on the assumption that the measured responses are the *true* responses.

A 95% prediction interval estimate can be constructed from the variance and bias as follows: $\hat{m}(\mathbf{x}_0) \pm 2 \times \sqrt{Var\{\hat{m}(\mathbf{x}_0)\} + Bias^2}$

4.9.3 Summary of Previous Research into Prediction Interval Estimation for Nonparametric Regression

Previous research at the University of Tennessee provided a basis for the prediction interval estimation techniques applied for the nonparametric techniques [Gribok 2002]. The methods that were studied and developed are presented here, and the relationships between the previously derived uncertainty estimates and the prediction interval estimation methods derived herein are explained.

The most general approach to the analysis of uncertainty in statistically based techniques is bias-variance decomposition of the mean squared error (MSE), which can be written as:

$$E_D[\hat{m}_h(x) - m(x)]^2$$

where E_D is the expectation over the ensemble of possible data sets D of fixed sample size N and $\hat{m}_h(x)$ is an estimate of the true function $m(x)$. The MSE is a natural measure of the effectiveness of the predictor $\hat{m}_h(x)$ because it measures how close the estimate is to the true function, on the average. The MSE can be decomposed in the following manner:

$$E_D[\hat{m}_h(x) - m(x)]^2 = E_D[\hat{m}_h(x) - E_D(\hat{m}_h(x)) + E_D(\hat{m}_h(x)) - m(x)]^2 =$$
$$E_D\left[(\hat{m}_h(x) - E_D(\hat{m}_h(x)))^2 + (E_D(\hat{m}_h(x)) - m(x))^2 + 2*(\hat{m}_h(x) - E_D(\hat{m}_h(x)))*(E_D(\hat{m}_h(x)) - m(x))\right] =$$
$$E_D\left[(\hat{m}_h(x) - E_D(\hat{m}_h(x)))^2\right] + E_D\left[(E_D(\hat{m}_h(x)) - m(x))^2\right] + 2*E_D\left[(\hat{m}_h(x) - E_D(\hat{m}_h(x)))*(E_D(\hat{m}_h(x)) - m(x))\right] =$$
$$E_D\left[(\hat{m}_h(x) - E_D(\hat{m}_h(x)))^2\right] + [(E_D(\hat{m}_h(x)) - m(x))]^2 = Var \;+\; Bias^2$$

To arrive at the final formula, the following considerations were used:

$$2 * E_D[(\hat{m}_h(x) - E_D(\hat{m}_h(x))) * (E_D(\hat{m}_h(x)) - m(x))] = 0$$

$$E_D[(\hat{m}_h(x) - E_D(\hat{m}_h(x))] = E_D(\hat{m}_h(x)) - E_D(E_D(\hat{m}_h(x))) = E_D(\hat{m}_h(x)) - E_D(\hat{m}_h(x)) = 0$$

.

The last passage is made because $E_D(E_D(\hat{m}_h(x))) = (E_D(\hat{m}_h(x))$, as the mathematical expectation of a mathematical expectation, is just a mathematical expectation. For the same reason:

$$E_D\left[(E_D(\hat{m}_h(x)) - m(x))^2\right] = [E_D(\hat{m}_h(x)) - m(x)]^2 .$$

This last expression is the squared bias as it reflects the deviation of the mathematical expectation of the estimate from the true function. Bias represents a systematic error due to the error of modeling. Bias also reflects the limited explanatory power of an estimate. One of the major and nontrivial results of Statistical Learning Theory [Vapnik 1998] is that bias is necessary in order to perform consistent learning from data. In other words, unbiased systems (systems with an infinite or very large Vapnik-Chervonenkis (VC) dimension) fit all the peculiarities of a particular data set, thus losing generalization capabilities. Generalization is the ability of a model to properly predict samples not in the training set.

The problem of "bias design" is of utmost importance in statistical learning from data. The bias/variance dilemma shows that one can eliminate a substantial part of the variance by purposefully introducing bias; however, one must ensure that the bias is in fact harmless for the problem at hand. One of the most well-known and widely used biases in statistical learning from data is the smoothness bias [Tikhonov 1997], which favors functions with bounded derivatives over other functions. A measure of function smoothness can also be an integral operator of the following type:

$$\int_X \left[\frac{d^2 f(x)}{dx^2}\right]^2 dx$$

Physically the smoothness bias assumes that the system reacts smoothly to small perturbations, i.e. small changes in the input cause small changes in the output. The smoothness bias is very appropriate for most technologically generated data, for example, nuclear power plants are designed to be "smooth". Other biases which are frequently used in industrial applications are: the linearity of the underlying function, a limited domain of the function, and invariance of the sought mapping under certain groups of transformations. The most well known bias in science is a bias towards simplicity or simple models, which is more commonly known as Occam's razor. Notice, the bias term in involves the true function, which is generally unknown. However, the theoretical results on the comparison of biases of different competing estimators are very important as they show how quickly an estimator approaches its target as the sample size grows. Obviously, the faster the rate of convergence the better the estimator. Unfortunately, many theoretical results show that the convergence rates are prohibitively slow for all kinds of non-parametric estimators, especially in high dimensions. This brought many scholars to the conclusion that generalization based on pure data is practically impossible and all data driven methods have to be assisted by first principles models and strong prior information.

Variance, on the other hand, represents the uncertainty of an estimate due to a particular noise realization. The variance term measures the deviation of a given estimate obtained on a particular data set from the average estimate, which could be obtained by averaging over all the data sets of size N. The variance term does not depend on the true dependency (model) and is easier to estimate, although it requires an estimation of the noise variance in the response variable.

The result of the original research provided that a 95% prediction interval for a kernel regression estimator could be produced from:

$$\pm 2\sqrt{Var\{\hat{m}(x) \mid \mathbf{X}\}}$$

The general assumed model is:

$$Y_i = m(X_i) + v^{1/2}(X_i)e_i, \qquad i = 1,...,n.$$

Where $v(x)$ is finite and the e_i are mutually independent and identically distributed random variables with zero mean and unit variance. The e_i's are also independent of the X_i's.

$$Var(\hat{m} \mid \mathbf{X}) = (\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1} (\mathbf{X}_x^T \Sigma \mathbf{X}_x)(\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x)^{-1}$$

$$\Sigma = diag\{K_h^2(X_i - x_0)n(X_i)\}$$

The assumption is made that the bias term is negligible in the prediction interval computation if the bandwidth of the kernel function is properly chosen. The basis for this is provided below. The conclusions drawn were based on the use of the Multivariate State Estimation Technique (MSET). Consider the typical bias / variance relationship of the MSE decomposition (figure 4.9.1).

As the bandwidth increases, the bias increases and the variance falls accordingly. The sum of squared bias and variance represents the mean integrated squared error (MISE); the minimum of which delivers the minimum uncertainty to the overall estimate. An important observation is that at the point of minimum uncertainty, the variance contribution to MISE dominates the bias contribution making it possible to ignore bias in practical uncertainty estimations.

One of the goals of this dissertation was to develop a prediction interval methodology that could be applied to nonparametric regression techniques regardless of the bandwidth chosen. If the training data fully represents the system being modeled, there will be bandwidth values for which bias is negligible. However, as the bandwidth increases the bias will no longer be negligible. Thus, omitting considerations of model bias may be appropriate for a select set of bandwidths but will not be appropriate to compute prediction intervals over a wide range of bandwidths. For these purposes a method of bias estimation was adopted and implemented in this dissertation. Another reason for including an estimate of bias is that in practical situations with data from measurement

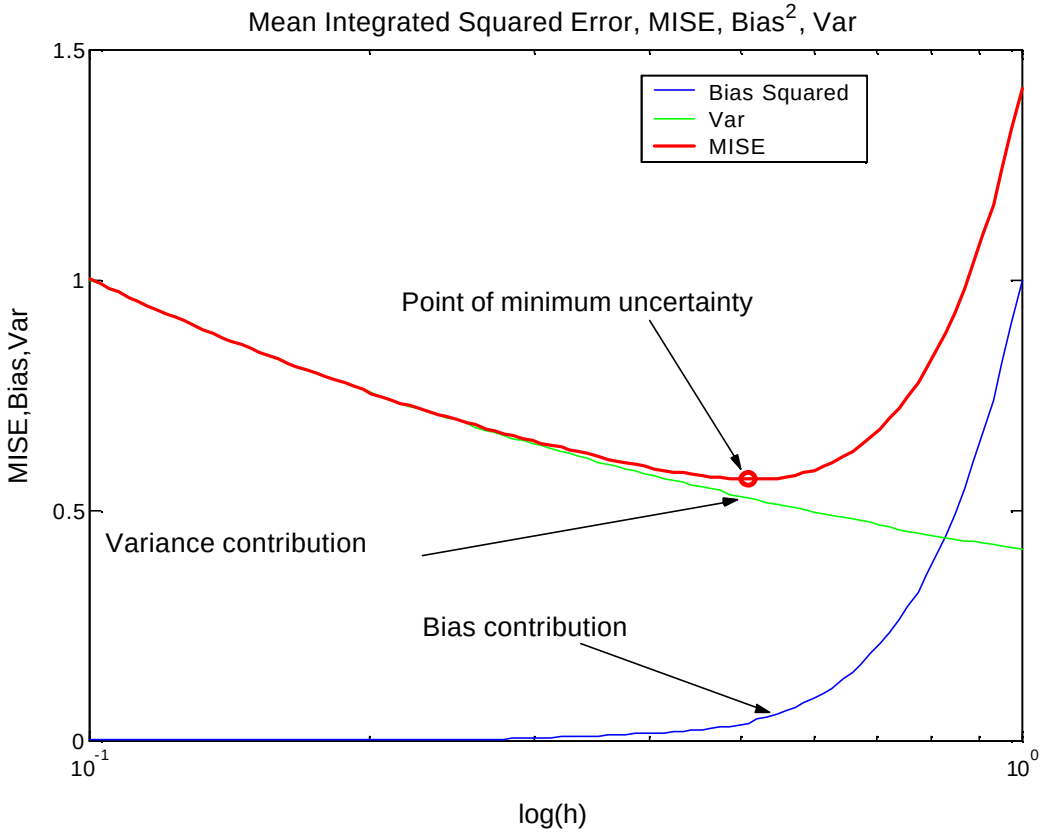


Figure 4.9.1: Nonparametric regression bias / variance trade-off.

instruments of a process, collecting a training data set the adequately represents the entire range of a variable is a difficult task. The problem is worsened by the presence of noise in the data. If the selected training data do not provide adequate representation, the variance estimate will be lower than the true variance.

It is important to keep in mind that the bias and variance computations derived do not treat the presence of noise in the predictor variables. This is a common assumption of empirical models in general. The bias estimation developed in this dissertation also considers the noise in the predictor variables through the utilization of validation data observations in the bias estimate. The bias of the training data set in the large majority of cases will be lower, sometimes significantly lower depending on the noise level in the data, than that for the validation data. Including the estimations on the independent validation data set into the bias estimate attempts to compensate for the noise in the predictor variables.

Thus, the difference between the prior research at the University of Tennessee and what has been developed for this dissertation is that the form of the prediction interval estimate is:

$$\pm 2\sqrt{Var\{\hat{m}(x) \mid \mathbf{X}\} + Bias\{\hat{m}(x) \mid \mathbf{X}\}^2}$$

The presence of a negligible bias at certain bandwidths was evidenced during this work. For these cases, the chosen method of bias estimation provided validation of the original assumptions drawn. Including the bias estimate for cases of negligible bias will have no influence on the resultant prediction intervals and the inclusion of its estimate is maintained in all cases studied. The benefit of the inclusion of the bias estimate in the prediction interval computation is that over-regularized models, due to large bandwidth selection, will result in prediction intervals that reflect this. In other words, poor bandwidth choices result in large prediction intervals.

4.10 PREDICTION INTERVAL ESTIMATION VIA THE BOOTSTRAP

The bootstrap is a method of Monte Carlo simulation whereby no assumptions are made about the population from which a random sample was obtained. The random sample is taken to be an estimate of the population, and each value within the random sample has an identical probability of occurrence. For each random sample, an estimate of a specific population parameter can be obtained. Repetitive sampling and consequent estimations provide a distribution for the parameter of interest. In the case of regression estimation, the parameter of interest is the estimate of the response, and its distribution can be used to construct confidence intervals around the estimate. Full coverage of the bootstrap technique is provided by several sources [Efron and Tibshirani 1993, Hall 1992, Efron 1982]. The use of bootstrap techniques, for constructing prediction intervals in nonlinear regression models, has been documented by various researchers. Derks and Buydens [1998] assess bootstrap resampling methods and the delta method for estimating prediction intervals for multi-layer feedforward neural networks. The utility of the bootstrap is prescribed in the following statement by Dybowski [1997]: "The bootstrap is a computer-based method for assigning measures of accuracy to statistical estimates, and it will provide a nonparametric confidence interval for any population parameter whatsoever."

Tibshirani [1996] presented a comparison of methods for estimating the standard error of prediction for MLPs. The three methods discussed were the delta method (incorporating the Hessian matrix), the *sandwich* estimator, and bootstrap estimators. The sandwich method augments the Hessian based delta method to produce estimates that may be improved under model misspecification. Their findings were that the bootstrap approach was the optimal technique, partly because it captures the variability due to the random initialization of the neural network weights. Neither the delta method, or the sandwich method capture the variability due to the random initialization of the neural network weights. A final observation is that the bootstrap approach, while computationally intensive, does not require the existence of derivatives or suffer from matrix inversion

problems. The delta method for estimating the standard error of prediction was also employed for NNs by Derks et. al. [1998], and is of the same form as that reported by Chryssolouris [1996], though the Hessian matrix was replaced by the Jacobian matrix in the latter.

The delta method is so named because it is based on the Taylor series approximation. The delta method, also known as the Taylor series method, has been reported to be less reliable than the bootstrap, with an occasional tendency to badly underestimate the true standard error [Efron 1992]. Bootstrap resampling estimation of prediction intervals should contain all sources of variation, including variation due to re-initialization and re-training [Derks 1998].

There are two general approaches to bootstrapping in regression settings: bootstrap pairs, and bootstrap residuals. A presentation of the two approaches was provided by Tibshirani [1996], and is repeated here:

The bootstrap pairs algorithm:

1. Generate B samples (typically in the range $20 \le B \le 200$), each one of size n drawn with replacement from the n training observations $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ...(\mathbf{x}_n, y_n)\}$. Denote the b^{th} sample by $\{(\mathbf{x}_1^{*b}, y_1^{*b}), (\mathbf{x}_2^{*b}, y_2^{*b}), ...(\mathbf{x}_n^{*b}, y_n^{*b})\}$.
2. For each bootstrap sample $b = 1, ..., B$, minimize $\sum_{i=1}^{n} [y_i^{*b} - y(\mathbf{x}_i^{*}; q)]^2$ to obtain $\hat{q}^{*b}$. y_i^{*b} are the targets or outputs of the b^{th} bootstrap data set, $y(\mathbf{x}_i^{*b}; q)$ are the estimates of the targets, and $\hat{q}^{*b}$ is the set of parameters for the b^{th} model.
3. Estimate the standard error of the i^{th} predicted value using: $\left\{ \frac{1}{B-1} \sum_{i=1}^{B} [y(\mathbf{x}_i; \hat{q}^{*b}) - y(\mathbf{x}_i; \cdot)]^2 \right\}^{1/2}$, where $y(\mathbf{x}_i; \cdot) = \sum_{b=1}^{B} y(\mathbf{x}_i; \hat{q}^{*b}) / B$.

The bootstrap residuals algorithm:

1. Estimate $\hat{q}$ from the training data and let $r_i = y_i - y(\mathbf{x}_i;\hat{q})$, for $i = 1,...,n$.
2. Generate B bootstrap samples, each one of size n drawn with replacement from $r_1, r_2, ..., r_n$. Denote the b^{th} sample by $r_1^{*b}, r_2^{*b}, ..., r_n^{*b}$ and let $y_i^{*b} = y(\mathbf{x}_i;\hat{q}) + r_i^{*b}$.
3. For each bootstrap sample $b = 1,...,B$, minimize $\sum_{i=1}^{n}[y_i^{*b} - y(\mathbf{x}_i;q)]^2$ to obtain $\hat{q}^{*b}$.
4. Estimate the standard error of the i^{th} predicted value using:

$$\left\{\frac{1}{B-1}\sum_{i=1}^{B}[y(\mathbf{x}_i;\hat{q}^{*b}) - y(\mathbf{x}_i;\cdot)]^2\right\}^{1/2}, \text{ where } y(\mathbf{x}_i;\cdot) = \sum_{b=1}^{B} y(\mathbf{x}_i;\hat{q}^{*b})/B.$$

Standard errors and confidence intervals produced via the resampling pairs approach will be asymptotically valid in the presence of heteroscedasticity or other forms of nonhomogeneity [Carroll 1995]. The drawbacks associated with this approach involve the fact that in this way estimations are obtained for unconditional sampling distributions, rather than conditional sampling distributions. Efron and Tibshirani [1993] indicate that conditional and unconditional standard errors are often nearly equal. Also, unconditional variances are generally larger, and thus more conservative, than conditional variances.

Consider the two components of uncertainty, the squared bias and the variance. While the use of the bootstrap standard error to construct prediction intervals has been reported by various researchers [Derks 1998, Tibshirani 1996], a better estimator is proposed which adds to the variance an estimate of the bias. Thus, this adjusted standard error calculation includes both a contribution from the variance, and the squared bias.

The bootstrap prediction intervals are computed via:

$$\hat{y}_j \pm 2 \times \sqrt{SE_j}$$

The bootstrap variance computation is shown below:

$$Var(\hat{y}_j) = \frac{1}{B-1} \sum_{b=1}^{B} \left[\hat{y}_j^{*b} - \overline{\hat{y}_j^*} \right]^2$$

$\hat{y}_j^{*b} = f\left(\mathbf{x}_j; \hat{q}^{*b}\right)$ is the estimate of the response $\hat{y}_j^{*b}$ for the observation $\mathbf{x}_j$ based on the model parameters $\hat{q}^{*b}$, and $\overline{\hat{y}_j^*} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_j^{*b}$.

The traditional standard error computation is given as:

$$SE_j = \sqrt{Var(\hat{y}_j)}$$

It was found to be desirable to add an estimate of bias into this computation due to the ease with which nonlinear regression models can be misspecified due to overparameterization or underparameterization. In either case there will be significant bias contributions that need to be accounted for. For the purpose of developing robust prediction intervals which are not susceptible to underestimating the uncertainty for conditions of misspecified models, an empirical method of bias estimation was adopted.

The bootstrap bias estimate will be based on the pool of data used to develop the models. It will therefore be a single value that attempts to quantify the model misspecification. The pool of development data represents all available data, excluding the defined set of test data. The reason for computing bias estimates based on the data pool, rather than the training data is that the training data bias estimate for overfit models is lower than a bias estimate obtained on an independent set of data. For the ANN and NNPLS methods, overfit models result from overtraining, and overparameterization. For LPR models (e.g. KR, LL), overfit models result when the bandwidth parameter is too small. In attempting to develop a prediction interval methodology that can provide the expected coverage regardless of the model fit, one must ensure that the prediction intervals reflect that a

model is overfit. To simplify, overfit models should have larger prediction intervals than a properly fit model. This increase in prediction interval magnitude can be realized by ensuring that the estimate of bias includes not only the training data, too which the model has overfit and will report a relatively small bias, but also an independent set of data, for which the reported bias will be higher. By computing the bias on all of the data in the development pool, the resultant value provides a sufficient estimate of the overall bias of the model.

Define the pool of data from which all bootstrap samples are drawn as: y^{POOL}. The number of observation / response pairs in y^{POOL} is K, and y_k^{POOL} will be defined as the k^{th} response in the pool, where $k = 1,\ldots,K$. In addition, to differentiate between the estimated responses from the B different models defined by their corresponding parameter sets $\hat{q}^{*b}$, the notation $\hat{y}_k^{POOL,*b}$ will be used. $\hat{y}_k^{POOL,*b}$ specifies the estimation of the response y_k from the b^{th} model.

$$Bias_1 = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{B}\sum_{b=1}^{B}\left[\left|\hat{y}_k^{POOL,*b} - \overline{\hat{y}_k^{POOL*}}\right|\right]$$

$\hat{y}_k^{POOL,*b} = f\left(\mathbf{x}_k^{POOL};\hat{q}^{*b}\right)$ is the estimate of the response $\hat{y}_k^{POOL,*b}$ for $\mathbf{x}_k^{POOL}$ based on the model parameters $\hat{q}^{*b}$, and $\overline{\hat{y}_k^{POOL*}} = \frac{1}{B}\sum_{b=1}^{B}\hat{y}_k^{POOL,*b}$

During the course of this work it was noted that the above bootstrap bias estimator did not adequately quantify the observed bias. For this reason, a second method of bias estimation was developed:

$$Bias_2 = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\frac{1}{B}\sum_{b=1}^{B}\left[\hat{y}_k^{POOL,*b} - y_k^{POOL,*b}\right]^2}$$

The $Bias_1$ estimate is referred to as the average deviation bias estimate, and the $Bias_2$ estimate is referred to as the average squared error bias estimate.

To provide prediction intervals for the estimations based on the pool of development data, as well as independent sets of test data, the following computational method was adopted:

$\hat{y}_j \pm 2 \times \sqrt{Var(\hat{y}_j) + Bias_g^2}$ $g = 1$ or $g = 2$ corresponds to the appropriate bias estimate above.

This proposed estimator of the prediction interval is more robust to model misspecification due to the inclusion of the bias estimation into the calculation. If the model is correctly specified then the bias term will be correspondingly negligible. If on the other hand there is a high degree of misspecification, this will be reflected in the computation.

While it is apparent that the method of bootstrapping prediction intervals is a robust and exhaustive approach, one must consider the practical applications of this technique. It was noted that if the average of 3 neural networks are used as the prediction, the standard error of prediction decreases [Tibshirani 1996]. For relatively small model dimensions, the use of the bootstrap approach to direct prediction interval estimation should be realizable; though, there will be an upper limit on the size of model for which this technique can be applied in *real* time.

4.11 SUMMARY OF PREDICTION INTERVAL COMPUTATIONS

The information presented in this chapter relative to the model specific methods of prediction interval computation is summarized below. The equations below exactly describe the computational methods applied to obtain the results presented in the next chapter.

For ANN and NNPLS, the general nonlinear regression model is:

$\mathbf{y}_i = f(\mathbf{x}_i, \vec{q}) + \mathbf{e}_i$

where: $\mathbf{x}_i = [X_{i,1} \quad X_{i,2} \quad \cdots \quad X_{i,p}]$, $i = 1,...,n$

p is the number of predictor variables

$\mathbf{e} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \cdots \quad \mathbf{e}_n]$

The assumptions of normality and independence are $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}s_e^2)$.

The general prediction interval computation is given by:

$$\hat{y}_0 \pm t_{a/2}^{n-p} \cdot s\sqrt{1+\mathbf{f}_0^T(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{f}_0}$$

To simplify, the significance-level dependent interval was modified to the following form which represents a 95% prediction interval.

$$\hat{y}_0 \pm 2 \cdot s\sqrt{1+\mathbf{f}_0^T(\mathbf{F}^T\mathbf{F})^{-1}\mathbf{f}_0}$$

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}\left[y_i - f(\mathbf{x}_i;\hat{\hat{q}}) \right]^2 \quad \text{[Tibshirani 1996]}$$

$\mathbf{F} = \frac{\partial \hat{\mathbf{y}}^T}{\partial \hat{q}}$ $\hat{\mathbf{y}}$ is the vector of training responses

$\mathbf{f}_0^{\mathbf{T}} = \frac{\partial \hat{y}_0}{\partial \hat{q}}$ $\hat{y}_0 = f(\mathbf{x}_0;\hat{q})$ is the estimate of the response for the new observation $\mathbf{x}_0$. The assumption that $t_{a/2}^{n-p} \approx 2$ ($a = 0.05$) is valid for $n - p \geq 60$, for greater degrees of freedom the assumption results in a slight over-estimation, the limit of this value as $n \to \infty$ is 1.96 (for $a = 0.05$). All training data sets used in this dissertation provide a number of degrees of freedom where $n - p \geq 60$.

The specific parameter vectors $\hat{q}$ for the ANN model and NNPLS model are given in sections 4.6 and 4.7, respectively. In the early stages of this work, it was noted that the fractional coverage values of the prediction intervals based on an s^2 estimate computed for the training data were lower than the expected 0.95, especially for cases where the neural networks were allowed to overtrain or the models were overparameterized for the given problem. Investigation led to the observation that in many cases, the s^2 estimate based on an independent set of data drawn from the same population of data as the training set was higher than that computed based solely on the training data. Because the

focus of this work was to provide a prediction interval estimation method that would provide the expected coverage on sets of data that were independent, this observation led to a modification of the s^2 estimate of Tibshirani [1996]. It should be noted that the same estimator based on training data was also reported in Chryssolouris [1996], with the exception that the denominator was reduced by the degrees of freedom of the model. The modification is to compute the s^2 estimate based on a combined set of data consisting of the training data and an independent set of validation data. Both the training and validation sets are drawn from the same population or pool of data. This estimation method proved to be much more stable for all model architectures rather than only performing as expected when the ideal model was used. The stability referred to here is with respect to consistent coverage of the measured values by the estimate and its point-wise prediction interval, to the expected level of 95%.

The overall approach for ANNs and NNPLS is:

1. Calculate the full $n \times p$ Jacobian matrix ($\mathbf{F}$) based on the full matrix of training data.
2. Estimate s^2, based on the combined set of training and validation data.
3. For each new observation (test data), $\mathbf{x}_0$, compute the estimate $\hat{y}_0 = f(\mathbf{x}_0; \hat{q})$, the vector of partial derivatives, $\mathbf{f}_0^{\mathrm{T}}$, and the corresponding 95% prediction interval.

For the LPR models (KR and LL), the random design nonparametric regression model can be written as:

$$Y_i = m(X_i) + v^{1/2}(X_i)e_i, \qquad i = 1,...,n$$

The e_i are independent random variables with zero mean and unit variance conditional on $X_1,...,X_n$, i.e. $E(e_i) = 0$ and $Var(e_i) = 1$. The regression and variance functions are thus:

$$m(x) = E(Y \mid X = x) \qquad \text{and} \qquad v(x) = Var(Y \mid X = x)$$

The variance of the estimator based on a new observation $\mathbf{x}_0$ is given by:

$$Var\{\hat{m}_0(\mathbf{x}_0) \mid \mathbf{X}\} = \mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{V}\mathbf{W}_x\mathbf{X}_x(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{e}_1$$

To obtain an estimate for the bias of a given KR or LL model, the following computation was employed:

$$Bias^2 = \frac{1}{r}\sum_{i=1}^{r}\{m(\mathbf{x}_i) - \hat{m}(\mathbf{x}_i)\}^2$$

$\mathbf{x}_i$ represents the i^{th} observation of the combined set of training and validation data, $i = 1,...,r$ (r is the total number of training and validation observations). The bias estimate is based on both the training and validation data so that an adequate estimate of bias can still be obtained for bandwidth values that are too small, resulting in an overfit model. The argument for this is similar to that of the previous paragraph regarding neural network overfitting. An overfit model will report a minimal bias estimate for the training data, while the same estimate on an independent set of validation error will be respectively much larger.

The 95% prediction interval estimates for KR and LL were constructed from the variance and bias as follows:

$$\hat{m}(\mathbf{x}_0) \pm 2 \times \sqrt{Var\{\hat{m}(\mathbf{x}_0)\} + Bias^2}$$

The overall approach for KR and LL regression models is:

1. Obtain an estimate for the variance of the response $v(x_i) = Var(Y \mid X = x_i)$, assume that $v(x_i) = s^2$ for all i, i.e. the model is homoscedastic.
2. Obtain a bias estimate for the current model based on a combined set consisting of the training data and an independent validation data set.
3. For each new observation $\mathbf{x}_0$ (test data) compute the variance estimate based on the new observation, and the corresponding 95% prediction interval.

For all models, the bootstrap prediction interval estimates were obtained in the same way, based on 100 different resampled data sets. The bootstrap prediction intervals were computed based on the following form:

$$\hat{y}_0 \pm 2 \times \sqrt{Var(\hat{y}_0) + Bias_g^2}$$

$g = 1$ or $g = 2$ corresponds to the appropriate bias estimate (section 4.10).

$$Bias_1 = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{B}\sum_{b=1}^{B}\left[\left|\hat{y}_k^{POOL,*b} - \overline{\hat{y}_k^{POOL*}}\right|\right]$$

$$Bias_2 = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\frac{1}{B}\sum_{b=1}^{B}\left[\hat{y}_k^{POOL,*b} - y_k^{POOL,*b}\right]^2}$$

$$Var(\hat{y}_0) = \frac{1}{B-1}\sum_{b=1}^{B}\left[\hat{y}_0^{*b} - \overline{\hat{y}_0^{*}}\right]^2$$

The bootstrap approach for all models was to obtain the estimations for B different models based on B different sets of training and validation data. All training and validation data samples were drawn from the same data pool. Descriptions of the variables in the bootstrap prediction interval equations were provided in section 4.10.

4.12 SUMMARY OF ASSUMPTIONS OF MODELS AND PREDICTION INTERVAL COMPUTATIONS

In this section, the assumptions inherent to the empirical models are summarized. In addition, various assumptions were required for the derivation of the prediction interval computations. These assumptions are also described. In cases where the assumptions are not valid a description is provided as to how the proposed prediction interval computations account for this. All of the studied empirical models fall within two general categories: nonlinear regression models (NNPLS and ANN) and nonparametric regression models (KR and LL). Each category is discussed separately. All of the equations in this section were presented in greater detail in the appropriate sections of this dissertation.

Beginning with nonlinear regression, the general model is:

$\mathbf{y}_i = f(\mathbf{x}_i, \vec{q}) + \mathbf{e}_i$

where: $\mathbf{x}_i = [X_{i,1} \quad X_{i,2} \quad \cdots \quad X_{i,p}]$, $i = 1,...,n$

p is the number of predictor variables

$\mathbf{e} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \cdots \quad \mathbf{e}_n]$

The assumptions of normality and independence are $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}s_e^2)$.

Other assumptions are:

- the predictor variables are noise free
- all required predictor variables are available
- the predictor variables are independent
- $\mathbf{e}_1 = \mathbf{e}_2 = \cdots = \mathbf{e}_i = \cdots \mathbf{e}_n$, i.e. constant variance errors (homoscedasticity)
- the error terms are normally distributed and independent

The general prediction interval computation is given by:

$$\hat{y}_0 \pm t_{a/2}^{n-p} \cdot s\sqrt{1 + \mathbf{f}_0^T (\mathbf{F}^T \mathbf{F})^{-1} \mathbf{f}_0}$$

To simplify, the significance-level dependent interval was modified to the following form which represents a 95% prediction interval. The effect of this modification is to produce more conservative prediction interval values since the limiting value of the Student's t distribution for large n is ~1.96, at the significance level 0f 95%.

$$\hat{y}_0 \pm 2 \cdot s\sqrt{1 + \mathbf{f}_0^T (\mathbf{F}^T \mathbf{F})^{-1} \mathbf{f}_0}$$

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} \left[y_i - f(\mathbf{x}_i; \hat{\vec{q}}) \right]^2 \quad \text{[Tibshirani 1996]}$$

$$\mathbf{F} = \frac{\partial \hat{\mathbf{y}}^T}{\partial \hat{q}} \qquad \hat{\mathbf{y}} \text{ is the vector of training responses}$$

$\mathbf{f}_0^{\mathbf{T}} = \dfrac{\partial \hat{y}_0}{\partial \hat{q}}$ $\hat{y}_0 = f(\mathbf{x}_0; \hat{q})$ is the estimate of the response for the new observation $\mathbf{x}_0$.

The prediction interval computations assume that the first order Taylor approximation shown below is valid:

$$f(\mathbf{x}_i; \hat{\vec{q}}) \approx f(\mathbf{x}_i, \vec{q}_0) + \sum_{k=1}^{p} \left\{ \left[\frac{\partial f(\mathbf{x}_i, \vec{q})}{\partial q_k} \right]_{\vec{q}=\vec{q}_0} \left(\hat{q}_k - q_k \right) \right\}$$

The validity of the use of the first order Taylor series expansion depends on $\hat{\vec{q}}$ (the estimated set of parameters) being close to the true set of parameters $\vec{q}$.

The prediction intervals computations also rely on the assumption that the variance covariance matrix can be estimated via:

$\mathbf{S} = s^2 \left[\mathbf{F}^T \mathbf{F} \right]^{-1}$ [Chryssolouris 1996]

s^2 is an estimate of the noise variance s_e^2.

Results from a Monte Carlo study of the above methods of covariance matrix estimation [Hogg 1987] indicated that this method based solely on the Jacobian matrix gave the best results with the least amount of effort.

It was also assumed that the variance of the noise can be estimated via:

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} \left[y_i - f(\mathbf{x}_i; \hat{\vec{q}}) \right]^2$$ [Tibshirani 1996]

This assumption carries with it the inherent assumption that the model is correct, such that:

$$y_i - f(\mathbf{x}_i; \hat{\vec{q}}) = e_i$$

Model correctness relies on the proper selection of predictor variables, training observations, and model architecture. The variance of the noise estimate above was defined with respect to the response observations and corresponding estimates of the model training data. Because in this work the proposed methodology strays from the

assumption that the model is correct, this computation was modified to include a larger sampling of data than provided by the training data. The proposed larger set of data over which the above computation was extended consists in part of training data, which defines the adjustments to the model parameters during training. The remainder of the data over which this computation has been extended consists of a set of data that is independent of the training data, though still sampled from the same data population. the entire set of data over which the noise variance computation has been extended is herein referred to as the data pool. By doing this, the constraints of the assumption that the model is *correct* can be relaxed. For models that are in fact correct, increases in the resultant calculation will be negligible, and in cases where the model is not correct the increase in the computed value will suitably reflect the degree of model misspecification.

The above prediction interval computation has also been reported to provide prediction intervals that are too wide when training via early stopping is applied [deVeaux 1998]. While this is a consideration, the fact that the error is on the side of being more conservative implies that the measured observations are still contained in the prediction intervals. Because the purpose of the proposed intervals in this dissertation was to produce intervals that contain the measured observations to the expected level, the possible increase in prediction intervals incurred through the implementation of cross-validation training (early-stopping) is an acceptable result. The task at hand should still be achieved, to contain the measured observations in the proposed prediction intervals.

The proposed interval estimate has also been reported to produce unreliable results when the number of observations in the training data set is small relative to the size of the network [de Veaux 1998]. With a large number of training observations however, the prediction intervals have been shown to reflect the distribution of the training data [Wansink 2001]. The asymptotic properties of the prediction interval estimator have been investigated by Hwang and Ding [1997], who confirm that the intervals are valid for large training data sets.

A final assumption is that of noise-free predictor variables. Accommodations for relaxing this assumption are also inherently included in extending the estimated noise variance calculation to cover the entire data pool. This assumes that the noise in the predictors will be reflected in the estimated response and it will produce a higher noise variance estimate for the data pool. Additional comments on this assumption are provide in the recommendations for future work (section 6.1).

Considering the general univariate nonparametric regression model, the bivariate data: $(X_1, Y_1),...,(X_n, Y_n)$, form an i.i.d. sample from the entire population (X, Y). Multivariate extensions of this model are straightforward. The local polynomial regression procedure attempts to estimate the regression function $m(x_0) = E(Y \mid X = x_0)$, and its derivatives $m'(x_0), m''(x_0),..., m^{(p)}(x_0)$. The point x_0 is often referred to as the query point, and the set of bivariate data is often referred to as the training data. The data-generating model is:

$Y = m(X) + n^{1/2}(X)e$,

where: $E(e) = 0$, $Var(e) = 1$, X and e are independent, e is i.i.d.,

and $v(x) = Var(Y \mid X = x)$, and $m(x) = E(Y \mid X = x)$.

If p is the order of the local polynomial, then suppose that the $(p+1)^{\text{th}}$ derivative of $m(x)$ at x_0 exists. A Taylor expansion for x in a neighborhood of x_0 is given by:

$$m(x) \approx m(x_0) + m'(x_0)(x - x_0) + \frac{m''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{m^{(p)}(x_0)}{p!}(x - x_0)^p$$

The validity of the resultant nonparametric model depends on the validity of the Taylor series approximation at the query point x_0. Note that the validity of this assumption is also dependent on the bandwidth value which factors into the kernel function in the weighted least squares problem below. The weighted least squares problem which should be minimized is:

$$\sum_{i=1}^{n}\{Y_i - \sum_{j=0}^{p} b_j (X_i - x_0)^j\}^2 K_h(X_i - x_0)$$

$\hat{b}_j$ ($j = 0,..., p$) is the solution to the above least squares problem. If $\hat{b}_j = \left[\hat{b}_0,..., \hat{b}_p\right]$ is the minimizer of the weighted least squares problem, than an estimator of the regression function is: $\hat{m}(x) = \hat{b}_0$. The remaining regression parameters represent estimates of the derivatives of the regression function ($\hat{m}^k(x) = k!\hat{b}_k$, where $k = 0,..., p$).

The exact expressions for the conditional mean and variance of $\hat{m}(x)$ are shown below [Ruppert 1994]:

$$Var\{\hat{m}(x) \mid \mathbf{X}\} = \mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{V}\mathbf{W}_x\mathbf{X}_x(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{e}_1$$

$$E\left\{\hat{m}(x \mid \mathbf{X}) - m(x)\right\} = \frac{1}{2}\mathbf{e}_1^T(\mathbf{X}_x^T\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{X}_x^T\mathbf{W}_x\left\{Q_m(x) + R_m(x)\right\}$$

Both of these expressions rely on unknown quantities. In the case of the variance equation the unknown quantity is $\mathbf{V} = diag\{v(Y \mid X = X_1),..., v(Y \mid X = X_n)\}$. This variance can be estimated however, and the additional assumption of homoscedasticity can be applied resulting in:

$s_Y^2 = v(Y \mid X = X_1) = v(Y \mid X = X_2) = v(Y \mid X = X_n)$

Obtaining an estimate for this quantity is fairly straightforward given a sufficient amount of training data. Regarding the bias expression, the unknown quantity is the Hessian matrix which is embedded in

$Q_m = \left[(X_1 - x)^T H_m(x)(X_1 - x) \quad \cdots \quad (X_n - x)^T H_m(x)(X_n - x)\right]^T$. Note that the quantity $R_m(x)$ is a vector of Taylor series remainder terms and is generally neglected.

The 95% prediction interval estimate for the general nonparametric regression is of the form:

$$\hat{y} \pm 2 \cdot \sqrt{Var\{\hat{m}(x \mid \mathbf{X})\} + Bias\{\hat{m}(x \mid \mathbf{X})\}^2} \qquad \text{where } Bias\{\hat{m}(x \mid \mathbf{X})\} = E\{\hat{m}(x \mid \mathbf{X}) - m(x)\}$$

Asymptotic expressions for both the bias and variance are available; however, both maintain their reliance on unknown parameters. While these expressions are valid, their numerical evaluation remains to be a difficult task and the resultant estimates are strongly dependent on the methods of approximation used. Rather than utilize the asymptotic results, the variance component was estimated using the exact variance expression above. Of course the variance computation is not exact due to the required estimate of the conditional variance of the response. The bias component cannot be estimated from the exact expression without significantly greater efforts due to its reliance on the second derivatives of the function $m(x)$, which are required for computation of the Hessian matrix $H_m(x)$. Attempts to estimate the bias term based on the derivative estimates from the weighted least squares problem have been presented [Fan 1996]; however, these methods rely on the validity of the Taylor series approximation at the query point and the optimization of a bandwidth parameter for proper estimation of the derivatives. Rather than rely on the model directly for the bias estimate, in this dissertation a bias was estimated from the empirical model estimates, of the form:

$Bias(\mathbf{x}_i) = \sqrt{\frac{1}{r}\sum_{i=1}^{r}\{m(\mathbf{x}_i) - \hat{m}(\mathbf{x}_i)\}^2}$, where $\mathbf{x}_i$ represents the i^{th} observation of the combined set of training and validation data, $i = 1,...,r$ (r is the total number of training and validation observations).

This estimate relies on the assumption that the measured responses are the *true* values of the function $m(x)$, and any deviations from this are due to model bias. An additional assumption is that the estimates are not overly influenced by unrelated variation, i.e. that the estimates appropriately reflect the model and not fluctuations due to noise in the predictor variables. For the ideal case where all assumptions are met, the bias computation simply quantifies the error between the true measured values and the

smooth, and proper, model estimates. These deviations in the absence of variable noise would be due only to model bias. In applying this estimate of bias, as was done for the nonlinear regression case, the data over which the estimate was evaluated was the expanded to include an independent set of validation data as well as the usual set of training data. This avoids improper bias estimates based solely on the training data, e.g. for a very small bandwidth value, the bias estimate for the training data alone would be relatively negligible indicating little to no bias. If the model is correctly specified and all of the required information was included in the training data than this would be appropriate; however, for data from process measurements, the idea of a training data set that fully represents the entire set of relationships being modeled is not achievable, especially due to the presence of measurement noise in both the predictor variables and the response variable. Computing the bias estimate on a data set which contains the training observations as well as a relatively adequately sized set of independent observations from the same data population is a more robust estimator of bias. The assumption that the measured values are correct is maintained; however, the assumption of a fully representative set of training data (model correctness) is relaxed. This results in bias estimates at low bandwidth that are not blind to poor training data set construction, i.e. if the training data set presents an inadequate representation, then the bias estimate would reflect this due to the contributions in the bias computation from the observations in the data which were independent from those in the training data.

The assumption of noise-free predictors also pertains to the standard nonparametric model and its asymptotic variance estimator. In this work the asymptotic variance estimator was not used, rather the exact computation was employed based on an estimate of the conditional variance of the response. The use of the stated method for bias estimation for the nonparametric model prediction interval computation inherently includes variation due to the predictor variables noise. Thus, as was the case for the modified noise variance estimate for the nonlinear regression model prediction intervals, the predictor variable noise is inherently accounted for through the empirically observed estimates from the nonparametric model, quantified in the computation of the bias

component for the nonparametric case. A final comment on the presence of noise in the predictor variables is that if properly accounted for in the prediction interval computations, it can be helpful in providing stable and robust empirical models. The presence of predictor variable noise reduces the opportunity for overfitting in neural network training; thus, enhancing the network generalization properties. In fact, the purposeful introduction of noise into predictor variables while maintaining the same response variable is a commonly employed practice for regularizing neural networks, *training with jitter* [Reed 1998]. For the case of nonparamteric regression, the same properties of regularization also apply through an averaging effect whereby the added random noise provides a smoother regression function, assuming an appropriately large set of training data. Though, for the case of nonparametric regression it should be noted that a very small bandwidth model will still overfit the predictor variable data.

In summary, for the both the nonlinear regression models and the nonparametric models, the assumption of normally distributed residuals is made. In addition it is assumed that the variance of these residuals is homoscedastic. While this assumption may be violated in some cases it is an inherent modeling assumption that can not be avoided without significant efforts through the use of empirically derived densities and suitably modifying the model equations. Considering the task of applying these techniques for the purposes of signal validation in nuclear power plants, due to their steady state operation, the assumption of normally distributed residuals is usually appropriate.

While early stopping of neural network training may lead to increased prediction interval magnitudes, the error is on the conservative side and is an accepted consequence of the proposed methods. To ensure the asymptotic validity of the intervals the number of training samples used was high relative to the size of the neural networks being trained.

The validity of the resultant nonparametric model depends on the validity of the Taylor series approximation at the query point x_0, and the bandwidth dependent kernel function.

The assumptions of model correctness and noise free predictors for the nonlinear regression prediction intervals is compensated for in the modified noise variance computation, whereby the computation is carried out with respect to a large data pool which includes the training data observations as well as a suitably sized set of observations independent of the training data. Thus, model misspecification will be reflected in this computation. A similar argument can be presented for the nonparametric prediction interval computations, though for this case the effects of the predictor variable noise are accounted for in the bias computation, for which there is no explicit representation in the case of nonlinear regression model prediction interval techniques.

The idea of model correctness implies that the model is unbiased. For the case of nonparametric regression, the bias component is explicitly included in the prediction interval computation whereas for the case of nonlinear regression it is implicitly included in the modified noise variance estimate. Considering this further, it can be seen that the residual variance estimate for the nonlinear regression prediction interval computation is equivalent to the bias estimate of the nonparametric prediction interval computation. The inherent assumption for the nonlinear regression case is that the model is correct, there are two inherent assumptions for the implemented nonparametric regression case: that the model is correct, and that the estimates do not reflect noise, i.e. produce a smooth estimator for the response.

The prediction interval techniques described in this section were applied to one simulated data set and two *real* data sets from operational U.S. nuclear power plants. In the next chapter, the results from the application of these techniques are provided.

5.0 RESULTS

In this chapter the methods used to compile the results from the different empirical models is explained. The computation of the performance measures are explained in detail and the method by which the results were presented is described. Each of the three data sets is described in detail in the appropriate sections. The first data set was a simulated data set. The benefits of using a simulated data set are complete and accurate knowledge of the predictor variables and the response variable. In addition, noise and erroneous variables can be inserted into a simulated data set and since the exact value of the response is known, the direct effects of the inserted noise and erroneous variables can be observed and quantified. The last two data sets were from operating U.S. nuclear power plants. In both of these the response variable contains a known drift. The drifting channels were selected as the desired responses so that the application of the developed prediction interval methodologies could be studied with respect to their abilities to identify out-of-calibration instrument channels.

5.1 DESCRIPTION OF APPROACH TO REPEATED SAMPLING, MODELING CONSTRUCTION, AND PREDICTION INTERVAL COMPUTATION

In this section the steps carried out to construct and evaluate 4 different model types are provided in a brief list. The 4 different models include: NNPLS, ANN, KR, and LL. The last two types both fall under the more general heading of LPR models. Because one of the goals of the proposed prediction interval methodologies was to produce intervals of sufficient coverage (coverage is defined later in this section) over a wide range of model specifications, for each model a variety of different architectures were evaluated. For example an ANN model with 1 hidden neuron was evaluated then hidden neurons were increased up to some maximum. Each architecture, defined by the number of neurons, was evaluated. The purpose was to produce architectures that were under-parameterized as well as over-parameterized. The prediction intervals were expected to provide sufficient coverage for all model architectures. A similar range of architectures were evaluated for the other model types as well. For the LPR models, low bandwidths

represent under-regularized models, while high bandwidth models represent over-regularized models. In addition, for the bootstrap prediction interval computations, repeated resampling was performed to produce 100 different training / validation data sets. For each of the 100 resampled sets all model types and architectures were evaluated. Due to the evaluation of a range of architectures for each of 4 models on 100 different data sets, the volume of the results produced was extremely large. Thus, to provide a clear and concise description of the methods in which these results were produced and compiled is important. To this end, an initial set of 6 steps is provided which detail the steps required to produce the results, followed by a list and description of the results produced after the 6 steps have been completed for a given data set. The performance measures are then described in detail in the following section.

For each of the empirical modeling techniques the approach used to compile the results is presented below. For each set of data discussed the same procedure applies:

1. The available data were first divided into development data, and test data. The development data consists of a large pool of data from which training and validation samples can be drawn. The test data set is fixed.
2. The confidence levels for the prediction intervals were set at 95% in all cases. Specific to the type of empirical model are the following settings:
   Artificial Neural Network (ANN):
   - The initialization method for the weight and bias values.
   - The minimum and maximum number of hidden neurons to consider.

   Neural Network Partial Least Squares (NNPLS):
   - The initialization method for the weight and bias values.
   - While the number of hidden neurons to use for a given latent variable model can be defined, a method of cross-validation was adopted here.
   - The maximum number of possible latent variables for a given model was specified.

   Local Polynomial Regression (LPR):

- A minimum and maximum bandwidth value was specified as well as a set of values in between. Models were developed for all specified bandwidths. The conditional variance of the response variable was specified for the computation of prediction intervals.

3. 100 iterations were completed, for which a training set was constructed by random selection of observations from the data pool. The validation set was also constructed via random sampling from the data pool. Sampling was done with replacement. The number of samples in the training set was specified as n_{tr}, and for the validation set n_v. The number of samples in the test data set was specified as n_{ts}.
4. For each of the 100 iterations, for all possible model definitions given by the settings discussed in step 2, an ANN was trained and an NNPLS model was trained. The entire data pool and the test data were evaluated with each of the trained ANN and NNPLS model, as well as a KR model and a LL model.
5. The appropriate analytical prediction intervals were calculated for all models on both the data pool and the test data for each iteration.
6. After all 100 iterations are complete, the bootstrap prediction intervals were obtained for all models under evaluation for the data pool and the test data.

After the above procedure is completed, the following data sets, prediction intervals and coverage values were available for all possible specified architectures of each model. The data sets resulted from the iterative resampling. The analytic and bootstrap prediction intervals result from the prediction interval computations as discussed in section 4.11.

Data Sets and Estimations

- 100 training data sets: $(\mathbf{X}^b_{tr}, \mathbf{Y}^b_{tr})$, of dimensions $(n_{tr} \times p)$, and $(n_{tr} \times 1)$ respectively.
- 100 corresponding estimation sets for the training data $(\hat{\mathbf{X}}^b_{tr}, \hat{\mathbf{Y}}^b_{tr})$ of dimensions $(n_{tr} \times p)$, and $(n_{tr} \times 1)$ respectively.

- 100 validation data sets $(\mathbf{X}^b{}_v, \mathbf{Y}^b{}_v)$ of dimensions $(n_v \times p)$ and $(n_v \times 1)$ respectively.
- 100 corresponding estimation sets for the validation data $(\hat{\mathbf{X}}^b_{tr}, \hat{\mathbf{Y}}^b_{tr})$ of dimensions $(n_v \times p)$ and $(n_v \times 1)$ respectively.
- 1 test data set $(\mathbf{X}_{ts}, \mathbf{Y}_{ts})$ of dimensions $(n_{ts} \times p)$ and $(n_{ts} \times 1)$ respectively.
- 100 estimation sets for the test data $(\hat{\mathbf{X}}^b_{ts}, \hat{\mathbf{Y}}^b_{ts})$ of dimensions $(n_{ts} \times p)$ and $(n_{ts} \times 1)$ respectively.
- 1 data pool $(\mathbf{X}_{pool}, \mathbf{Y}_{pool})$ of dimensions $(n_{pool} \times p)$ and $(n_{pool} \times 1)$ respectively.
- 100 estimation sets for the data pool $(\hat{\mathbf{X}}^b_{pool}, \hat{\mathbf{Y}}^b_{pool})$ of dimensions $(n_{pool} \times p)$ and $(n_{pool} \times 1)$ respectively.

Analytic Prediction Intervals

- 100 sets of point-wise prediction interval values for the data pool API^b_{pool} $(b = 1,...,100)$ each having dimensions $(n_{pool} \times 1)$.
- 100 sets of point-wise prediction interval values for the test data API^b_{ts} $(b = 1,...,100)$ each having dimensions $(n_{ts} \times 1)$.

Bootstrap Prediction Intervals

- A single bootstrap bias value computed as the overall MSE for all 100 iterations based on the data pool.
- A set of bootstrap prediction intervals for the data pool BPI_{pool} of dimensions $(n_{pool} \times 1)$.
- A set of bootstrap prediction intervals for the test data BPI_{ts} of dimensions $(n_{ts} \times 1)$.

Analytic Prediction Interval Coverage

- 100 coverage values for the analytic prediction intervals C^b_{pool}, corresponding to the 100 estimation sets for the data pool obtained from the fraction of the n_{pool} response observations satisfying $\hat{y}^b_{(i),pool} + API^b_{(i),pool} \le y_{(i),pool} \le \hat{y}^b_{(i),pool} + API^b_{(i),pool}$, for $i = 1,...,n_{pool}$.
- 100 coverage values for the analytic prediction intervals C^b_{ts}, corresponding to the 100 estimations sets for the test data obtained from the fraction of the n_{ts} response observations satisfying $\hat{y}^b_{(i),ts} + API^b_{(i),ts} \le y_{(i),ts} \le \hat{y}^b_{(i),ts} + API^b_{(i),ts}$, for $i = 1,...,n_{ts}$.

Bootstrap Prediction Interval Coverage

- 100 coverage values for the bootstrap prediction intervals BC^b_{pool}, corresponding to the 100 estimations sets for the data pool obtained from the fraction of the n_{pool} response observations satisfying: $\hat{y}^b_{(i),pool} + BPI_{(i),pool} \le y_{(i),pool} \le \hat{y}^b_{(i),pool} + BPI_{(i),pool}$, for $i = 1,...,n_{pool}$.
- 100 coverage values for the bootstrap prediction intervals BC^b_{ts}, corresponding to the 100 estimations sets for the test data obtained from the fraction of the n_{ts} response observations satisfying: $\hat{y}^b_{(i),ts} + BPI_{(i),ts} \le y_{(i),ts} \le \hat{y}^b_{(i),ts} + BPI_{(i),ts}$, for $i = 1,...,n_{ts}$.

Note that in the case of the coverage values for the bootstrap prediction intervals, the same set of prediction intervals was applied to all estimations, whereas in the case of the coverage values for the analytic prediction intervals the b^{th} set of prediction intervals was applied to the b^{th} set of estimations. This is because the analytic prediction intervals were based on the specific set of training and validation data of the current iteration,

while the bootstrap prediction intervals were based on the overall results from the 100 iterations.

5.2 DEFINITIONS AND COMPUTATIONS OF TERMS USED TO QUANTIFY RESULTS

This section defines the terms used to quantify and compare the performances of the different models and prediction interval estimation techniques. While this is the general format that was used throughout; in some cases modifications were made. These cases are noted as they appear in the results section.

Table 5.2.1 is an excerpt from a table of results which is included here so that the methods adopted to compare the results can be better illustrated. The performance parameters utilized to quantify the results are: coverage, PI magnitude, MSE, and MAE, which are defined below. The discussions below provide examples based on a data pool $(\mathbf{X}_{pool}, \mathbf{Y}_{pool})$ of dimensions $(n_{pool} \times p)$ and $(n_{pool} \times 1)$ respectively, and the corresponding 100 estimation sets for the data pool $(\hat{\mathbf{X}}^{b}_{pool}, \hat{\mathbf{Y}}^{b}_{pool})$ of dimensions $(n_{pool} \times p)$ and $(n_{pool} \times 1)$ respectively.

PI Magnitude (prediction interval magnitude): All estimates are of the form: $\hat{y} \pm PI$, where $\hat{y}$ is the scalar estimate and PI is the corresponding 95% prediction interval. The term PI magnitude refers to the scalar value PI. To distill all of the computations into a single number for plotting and quantification, the approach used was as follows: For bootstrap prediction intervals, a single PI Magnitude was presented for each model and architecture. It is computed as the average PI for the n observations of the respective prediction interval set, e.g. BPI_{pool} represents the $(n_{pool} \times 1)$ vector of point-wise prediction interval values: $\left[bpi(1)_{pool} \quad bpi(2)_{pool} \quad \cdots \quad bpi(n_{pool})_{pool} \right]$, for the data pool.

A single PI magnitude value for this example is computed as $\overline{BPI}_{pool} = \frac{1}{n_{pool}} \sum_{i=1}^{n_{pool}} bpi(i)_{pool}$.

Table 5.2.1: Sample Results Table*

Model	Type	API		BPI		AC		BC		MAE	MSE
		m	s	m	s	m	s	m	s		
NNPLS	1	14.88	1.03	16.85	5.38	0.97	0.004	0.98	0.003	5.40	64.7
NNPLS	2	14.44	1.60	16.09	5.46	0.97	0.005	0.98	0.003	4.91	57.3
ANN	2	30.06	62.14	78.99	44.82	0.96	0.008	0.99	0.028	7.12	1069.1
ANN	3	39.74	83.68	82.89	47.15	0.97	0.011	0.99	0.026	7.48	1174.0
KR	0.005	4.54	0.13	6.83	1.74	0.88	0.007	0.97	0.005	1.20	6.0
KR	0.025	4.62	0.12	6.95	5.53	0.86	0.008	0.96	0.006	2.08	8.0
LL	0.005	6.34	0.34	13.82	6.55	0.81	0.012	0.98	0.006	2.82	25.6
LL	0.025	5.65	0.68	21.27	89.94	0.90	0.009	0.99	0.004	1.77	12.9

* Model type has a different meaning depending on the model, for NNPLS model type refers to the number of latent variables used, for ANN models, type refers to the number of hidden neurons, for KR and LL models type refers to the value of the global bandwidth. The following definitions are explained below: API – analytic prediction interval, BPI – bootstrap prediction interval, AC – analytic coverage, BC – bootstrap coverage, MAE – mean absolute error, MSE – mean squared error.

To complement this mean value, the standard deviation over the n_{pool} values was also provided. Referring to table 5.2.1, these results are presented in the columns labeled **BPI** m and s .

The computation for the analytical prediction intervals is slightly different because rather than a single vector of prediction interval values there were 100 vectors, API^{b}_{pool} ($b = 1,...,100$). First, 100 average values were computed via:

$\overline{API}^{b}_{pool} = \frac{1}{n_{pool}} \sum_{i=1}^{n_{pool}} api(i)^{b}_{pool}$, then a single value was obtained via:

$\overline{API}_{pool} = \frac{1}{100} \sum_{b=1}^{100} \overline{API}^{b}_{pool}$. The corresponding standard deviation of the 100 $\overline{API}^{b}_{pool}$ values was also provided in each case. Referring to table 5.2.1, these results are presented in the columns labeled **API** m and s .

Coverage: This quantity is defined as the fraction of observations that are bounded by the estimate and its corresponding prediction interval. For each model and architecture specified there will be 100 coverage values; thus, the results presented regarding a coverage value for a given architecture are average values over the 100 iterations. Result reporting based on mean values is always complemented with a set of error bars showing the corresponding standard deviation. For the bootstrap coverage computations, the 100 coverage values are computed based on a single set of prediction intervals, BPI_{pool}, for the given model and architecture, and the 100 estimation sets $\hat{\mathbf{Y}}^{b}_{pool}$. The averages and standard deviations of these 100 coverage values were reported in the results tables in the columns labeled (see table 5.2.1): **BC** m and s . For the analytic coverage values, each of the 100 iterations resulted in a different set of prediction intervals, API^{b}_{pool} , corresponding to the current set of estimations $\hat{\mathbf{Y}}^{b}_{pool}$. Consequently, 100 individual coverage values were again obtained. The averages and standard deviations of these 100

coverage values were reported in the results tables in the columns labeled (see table 5.2.1): **AC** m and s .

MAE: This is an error quantity, the mean absolute error. The usefulness of this error term is due to the ease of interpretation since the units of error directly correspond to the units of measurement. The b^{th} MAE value is computed as:

$MAE_b = \frac{1}{n_{pool}} \sum_{i=1}^{n_{pool}} \left| y(i)_{pool} - y(i)_{pool}^{b} \right|$, where the b^{th} set of estimations for

$\mathbf{Y}_{pool} = \left[y(1)_{pool} \quad y(2)_{pool} \quad \cdots \quad y(n_{pool})_{pool} \right]$ are given by

$\hat{\mathbf{Y}}_{pool}^{b} = \left[\hat{y}(1)_{pool}^{b} \quad \hat{y}(2)_{pool}^{b} \quad \cdots \quad \hat{y}(n_{pool})_{pool}^{b} \right]$.

For each model and architecture, 100 MAE values were computed corresponding to the 100 iterations. Result reporting for this quantity uses the mean value of the 100 individual MAE values. Thus, the values reported in the results tables were computed via:

$$\overline{MAE} = \frac{1}{100} \sum_{b=1}^{100} MAE_b$$

These overall mean values were reported in the results tables in the column labeled **MAE** (see table 5.2.1).

MSE: This is the standard error quantity of empirical models. The b^{th} MSE value is computed as:

$$MSE_b = \frac{1}{n_{pool}} \sum_{i=1}^{n_{pool}} \left[y(i)_{pool} - y(i)_{pool}^{b} \right]^2$$

The 100 values were then combined for result reporting via: $\overline{MSE} = \frac{1}{100} \sum_{b=1}^{100} MSE_b$. The overall MSE values were reported in the results tables in the column labeled **MSE** (see table 5.2.1).

5.3 CASCADE DATA SET

The cascade data set was chosen as the simulated data set to evaluate due to the simple non-linear relationships between the predictors and the response. In addition, the use of a simulated data set allows for complete knowledge of the *true* response values. While noise was inserted into the response variable, the *true* value is still known. In addition, the use of a simulated data set allows for the study of the effects of erroneous predictors in the predictor variable set on the prediction interval computations.

In this section, the data set is initially described followed by the parameter settings that were used (settings described in 5.1). A standard evaluation of the results is then presented including the evaluation of 2 different bootstrap prediction interval estimators (see section 4.10). The results of this initial evaluation indicate that one of the proposed bootstrap estimators consistently failed to produce the desired coverage; thus its evaluation is not carried out for the remaining 2 data sets in sections 5.4 and 5.5. After the standard evaluation is completed, a study of the effects of erroneous predictors (noise variables) was completed. The existence of erroneous predictors represents a case of model misspecification, where in this case the misspecification is due to the set of predictor variables. At the end of this section, a summary of the results for the cascade data set is presented.

The cascade data set contains 4 predictor variables A, B, C, D and a single response y.

$A = \sin(10x_1)$
$B = x_2$
$C = \sin(5x_3)$
$D = \exp[\sin(15x_4)]$

x_1, x_2, x_3, x_4, uniformly spaced on the interval $[-0.5, 0.5]$

$y = \exp[2A\sin(pD)] + \sin(50BC)$

The data pool $(\mathbf{X}_{pool}, \mathbf{Y}_{pool})$ of dimensions $(n_{pool} \times p)$ and $(n_{pool} \times 1)$ was specified by the following: $n_{pool} = 3000$, and $p = 4$. The observation vectors of the data pool were obtained by obtaining a generic $n_{pool} \times 1$ vector $\mathbf{x}$ of uniformly spaced points on the interval $[-0.5, 0.5]$. This vector was then used to obtain the 4 simulated inputs, each of dimensions $n_{pool} \times 1$, via: $\mathbf{A} = \mathbf{sin}(10 \times \mathbf{x})$, $\mathbf{B} = \mathbf{x}$, $\mathbf{C} = \mathbf{sin}(5 \times \mathbf{x})$, and $\mathbf{D} = \mathbf{exp}[\mathbf{sin}(15 \times \mathbf{x})]$. Finally, the full data pool was constructed via: $\mathbf{X}_{pool} = [\mathbf{A} \quad \mathbf{B} \quad \mathbf{C} \quad \mathbf{D}]$, and $\mathbf{Y}_{pool} = \mathbf{y}$.

The vector $\mathbf{y}$ has as its i^{th} element $y_i = \exp[2A_i \sin(p D_i)] + \sin(50 B_i C_i) + R_i$, where R_i is drawn from a $n_{pool} \times 1$ vector of random noise drawn from a normal distribution $R \sim N(0, 0.4)$. The addition of noise intends to make the simulation more applicable to the data sets to follow which come from real operating systems. Additive noise was not inserted into the predictor variables to minimize the complexity of this first analysis. This was purposefully implemented, and while the results are expected to provide larger prediction intervals when noise exists in the predictors, the resultant computations on noise-free predictors are still valid. The later evaluations of *real* data do contain predictor variable noise, thus if those results significantly deviate from what was observed here than a flaw in the described methods could be that predictor variable noise is not properly accounted for. Fortunately, this was not the case, and the evaluations for the *real* data sets were consistent with that described here. In addition, the study of the effects of erroneous predictors indirectly studies the effect of noise in the predictors since the erroneous predictors were normal random variables. The cascade data set pool response variable is shown in figure 5.3.1, and the predictor variables are shown in figure 5.3.2.

The data pool $(\mathbf{X}_{pool}, \mathbf{Y}_{pool})$ of dimensions $(n_{pool} \times p)$ and $(n_{pool} \times 1)$ respectively were specified by: $n_{pool} = 3000$, $p = 4$. The 100 training data sets: $(\mathbf{X}^b{}_{tr}, \mathbf{Y}^b{}_{tr})$, of dimensions $(n_{tr} \times p)$, and $(n_{tr} \times 1)$ respectively were specified by: $n_{tr} = 500$, $p = 4$. The observations

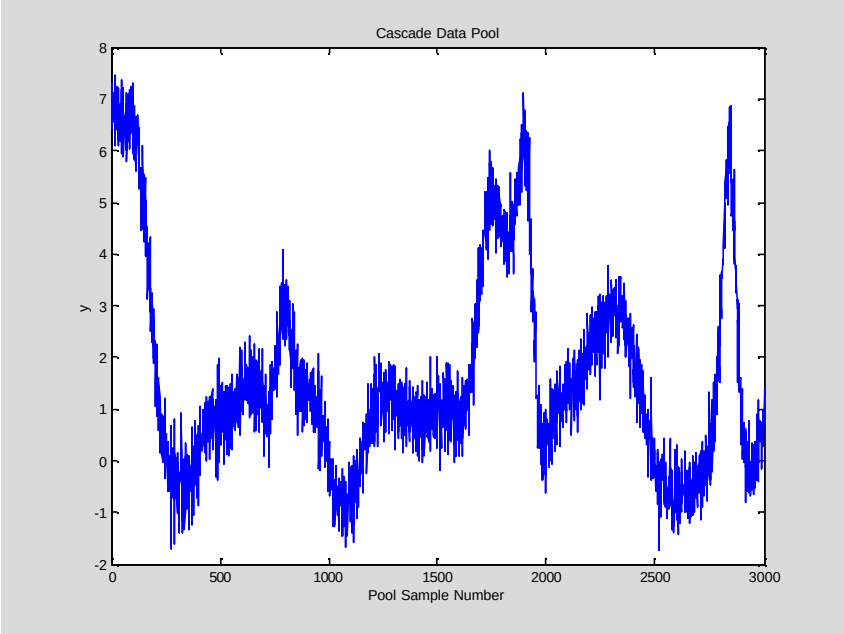


Figure 5.3.1: Cascade Data Pool Response Variable

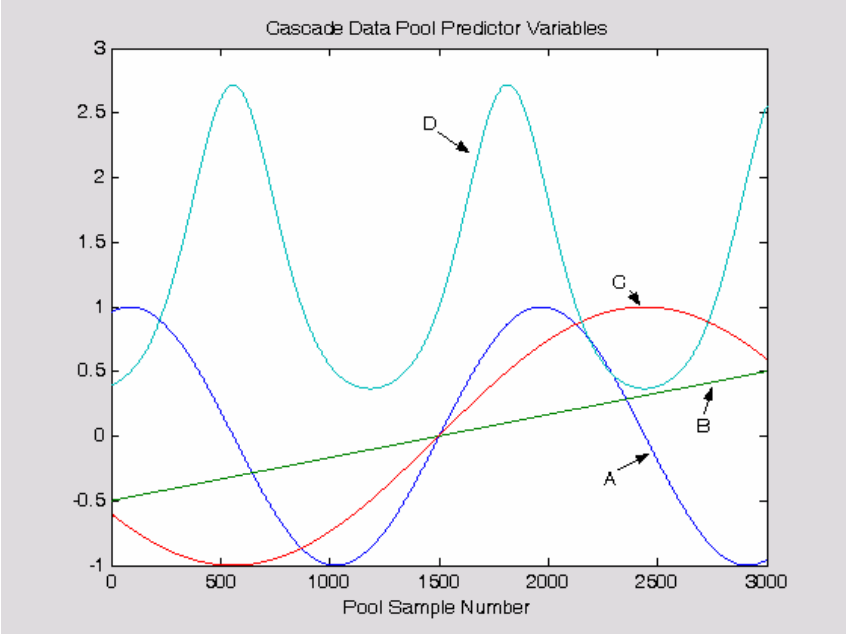


Figure 5.3.2: Cascade Data Pool Predictor Variables

were drawn at random, with replacement, from the data pool. In all cases, $b = 1,...,B$ and $B = 100$ (note that this is not the same B as used in the predictor variable definitions. The 100 validation data sets $(\mathbf{X}^{b}{}_{v}, \mathbf{Y}^{b}{}_{v})$ of dimensions $(n_v \times p)$ and $(n_v \times 1)$ respectively were specified by: $n_v = 250$, $p = 4$. The observations were drawn at random, with replacement, from the data pool. The test data set $(\mathbf{X}_{ts}, \mathbf{Y}_{ts})$ of dimensions $(n_{ts} \times p)$ and $(n_{ts} \times 1)$ respectively was specified by: $n_{ts} = 1500$, $p = 4$. The test data were created in the same way that the data pool was created.

The required model specifications used for the cascade data are provided below:

NNPLS:

Normal Random Initialization was applied. The maximum number of latent variables to evaluate was set at the dimension of the predictor variable matrix, 4.

ANN:

Normal Random Initialization was applied. The minimum and maximum number of hidden neurons evaluated were 1 and 10, respectively.

LPR:

The vector of bandwidths to be evaluated was:

[0.0025 0.005 0.01 0.025 0.05 0.1 0.15 0.2 0.25 0.3 0.4 0.5 0.75 1 2]

The conditional variance of the response was known for this case, thus $s_{y}^{2} = 0.16$.

A full analysis was carried out for the cascade data as described. Following which, a study of the effect of erroneous predictor variables on the magnitude and variance of the point-wise prediction interval estimates was carried out. For this task, two normal random variables with zero means and $s^{2} = 0.1$ were added to the predictor variable set. Full results for the normal set of 4 predictor variables are provided, followed by a set of comparisons between the results from the normal set of predictors and the augmented set of predictors which includes the normal random erroneous variables.

The overall results for the cascade data pool are presented in table 5.3.1, and for the test data in table 5.3.2. All of the following discussions for the cascade data set results refer to the values in these two tables. Note that the bootstrap prediction intervals being reported in these tables were based on the average squared error bias estimate ($Bias_2$). A full set of results is not reported for the bootstrap prediction intervals based on the $Bias_1$ computation due to the insufficient coverage that results from its use. This will become clear throughout the discussions of this section.

The PI magnitude results for the data pool are plotted in figure 5.3.3, and for the test data in figure 5.3.4. The method of computation for the analytic, bootstrap 1, and bootstrap 2 intervals was as described in section 4.10. For all architectures of all of the models, the bootstrap 1 PI magnitudes (PIMs) were consistently lower than the analytic or bootstrap 2 PIMs. The bootstrap 1 PIMs for more complex ANN and NNPLS models for the most part followed the trend of the bootstrap 2 PIMs; however, for the LPR models of increasing bandwidth this was not the case and the bootstrap 1 PIMs were significantly lower than the bootstrap 1 PIMs. This indicates that the bootstrap 1 approach for prediction interval estimation of LPR models is not sufficient under conditions of reasonable to large model bias. This is due to the reliance of the bootstrap 1 estimate on a bias dependent on the deviation of the bias values over the 100 iterations, rather than the squared error. The deviation of the bias values is not representative of the *true* model bias because the influence of increasing bandwidth is to reduce the variability in the estimates. This variance reduction has an associated bias increase that is not accounted for in the bootstrap 1 prediction interval bias estimate. The variation in the PIMs for the NNPLS models (figure 5.3.3.a and 5.3.4.a) is relatively consistent over all architectures for both the data pool and the test data. An exception being that for the 1 latent variable models, the PIMs variance was slightly lower in all cases. This is because the 1 latent variable model includes the highest level of regularization and the reduced variation in the PIMs is a result of this regularization. The variation in the PIMs for the ANN models tends to decrease as the number of hidden neurons increases. This is due to a better fit of the models as the complexity increases. It should be noted that this trend is not

Table 5.3.1: Tabulated results for cascade data pool*.
(For full explanation of terms see section 5.2)

Model	Type	API		BPI		AC		BC		MAE	MSE
		m	s	m	s	m	s	m	s		
NNPLS	1	2.962	0.123	3.080	0.122	0.938	0.011	0.951	0.012	1.140	2.195
NNPLS	2	2.278	0.224	2.625	0.181	0.954	0.009	0.982	0.011	0.878	1.276
NNPLS	3	2.122	0.255	2.465	0.175	0.956	0.009	0.982	0.012	0.799	1.086
NNPLS	4	2.060	0.220	2.374	0.176	0.956	0.009	0.981	0.012	0.765	1.008
ANN	1	2.961	0.580	3.382	0.542	0.947	0.018	0.979	0.007	1.098	2.258
ANN	2	2.491	0.558	2.964	0.393	0.945	0.015	0.976	0.019	0.912	1.611
ANN	3	1.988	0.458	2.446	0.323	0.943	0.012	0.977	0.023	0.723	1.053
ANN	4	1.653	0.454	2.050	0.322	0.943	0.010	0.979	0.028	0.602	0.740
ANN	5	1.378	0.391	1.702	0.282	0.947	0.010	0.980	0.034	0.512	0.514
ANN	6	1.133	0.229	1.347	0.214	0.948	0.007	0.979	0.018	0.436	0.341
ANN	7	1.011	0.184	1.194	0.140	0.948	0.007	0.979	0.020	0.403	0.276
ANN	8	0.961	0.133	1.105	0.094	0.949	0.007	0.977	0.015	0.386	0.244
ANN	9	0.896	0.069	1.009	0.061	0.948	0.007	0.973	0.011	0.366	0.213
ANN	10	0.908	0.185	1.061	0.109	0.948	0.007	0.978	0.023	0.369	0.227
KR	0.0025	0.978	0.023	1.347	0.134	0.936	0.025	0.990	0.003	0.393	0.283
KR	0.005	0.968	0.022	1.281	0.106	0.930	0.010	0.987	0.003	0.392	0.273
KR	0.010	0.929	0.021	1.207	0.103	0.932	0.007	0.984	0.003	0.384	0.253
KR	0.025	0.858	0.019	1.059	0.082	0.937	0.007	0.979	0.004	0.363	0.216
KR	0.050	0.831	0.019	0.970	0.059	0.941	0.007	0.974	0.004	0.352	0.197
KR	0.100	0.877	0.022	0.976	0.055	0.944	0.007	0.968	0.005	0.366	0.211
KR	0.150	0.999	0.031	1.087	0.059	0.945	0.006	0.965	0.006	0.408	0.270
KR	0.200	1.167	0.042	1.249	0.057	0.945	0.007	0.962	0.007	0.468	0.364
KR	0.250	1.355	0.049	1.433	0.052	0.944	0.007	0.959	0.008	0.535	0.486
KR	0.300	1.547	0.054	1.622	0.047	0.942	0.007	0.956	0.008	0.604	0.629
KR	0.400	1.913	0.058	1.983	0.039	0.942	0.008	0.953	0.008	0.740	0.952
KR	0.500	2.241	0.061	2.306	0.032	0.943	0.008	0.953	0.007	0.872	1.299
KR	0.750	2.900	0.076	2.951	0.017	0.938	0.009	0.943	0.009	1.130	2.154
KR	1.000	3.319	0.087	3.357	0.007	0.932	0.007	0.934	0.005	1.278	2.802
KR	2.000	3.747	0.109	3.770	0.001	0.929	0.008	0.930	0.004	1.436	3.544
LL	0.0025	1.575	0.254	2.632	0.925	0.972	0.016	0.985	0.005	0.507	0.846
LL	0.005	2.512	0.885	4.432	1.587	0.987	0.010	0.985	0.006	0.669	2.846
LL	0.010	3.027	1.191	5.747	2.255	0.991	0.008	0.987	0.006	0.744	4.829
LL	0.025	2.303	1.482	4.697	2.210	0.985	0.011	0.990	0.006	0.591	3.442
LL	0.050	1.056	0.235	1.544	0.554	0.957	0.014	0.990	0.005	0.398	0.411
LL	0.100	0.848	0.033	0.966	0.085	0.947	0.007	0.974	0.004	0.350	0.196
LL	0.150	0.827	0.018	0.909	0.041	0.949	0.006	0.969	0.004	0.342	0.183
LL	0.200	0.838	0.018	0.905	0.036	0.950	0.007	0.966	0.004	0.346	0.187
LL	0.250	0.901	0.021	0.962	0.026	0.949	0.006	0.964	0.004	0.367	0.216
LL	0.300	1.000	0.025	1.060	0.028	0.951	0.006	0.964	0.004	0.400	0.266
LL	0.400	1.289	0.035	1.350	0.034	0.949	0.006	0.960	0.004	0.501	0.438
LL	0.500	1.547	0.043	1.609	0.037	0.950	0.006	0.959	0.003	0.588	0.627
LL	0.750	1.976	0.057	2.035	0.028	0.953	0.006	0.959	0.004	0.750	1.013
LL	1.000	2.204	0.061	2.260	0.024	0.956	0.006	0.961	0.004	0.851	1.253
LL	2.000	2.670	0.073	2.716	0.019	0.961	0.005	0.963	0.003	1.056	1.818

* API – Analytic prediction intervals, BPI – Bootstrap prediction intervals, m - mean result, s - standard deviation of results, MAE – Mean Absolute Error, MSE – Mean Squared Error, Type for NNPLS refers to the number of latent variables, for ANNs refers to the number of hidden neurons, and for KR and LL refers to the bandwidth parameter. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

Table 5.3.2: Tabulated results for cascade test data*.
(for full explanation of terms see section 5.2)

Model	Type	API		BPI		AC		BC		MAE	MSE
		m	s	m	s	m	s	m	s		
NNPLS	1	2.962	0.123	3.080	0.122	0.939	0.010	0.951	0.011	1.134	2.164
NNPLS	2	2.278	0.224	2.625	0.181	0.955	0.010	0.982	0.011	0.869	1.262
NNPLS	3	2.122	0.255	2.465	0.175	0.956	0.009	0.982	0.013	0.790	1.073
NNPLS	4	2.060	0.220	2.374	0.176	0.956	0.009	0.981	0.013	0.756	0.996
ANN	1	2.962	0.580	3.382	0.543	0.947	0.018	0.979	0.009	1.103	2.267
ANN	2	2.491	0.558	2.965	0.393	0.946	0.017	0.976	0.017	0.916	1.629
ANN	3	1.988	0.458	2.446	0.323	0.943	0.014	0.975	0.023	0.729	1.073
ANN	4	1.653	0.454	2.051	0.322	0.942	0.011	0.977	0.029	0.607	0.757
ANN	5	1.378	0.391	1.703	0.282	0.946	0.010	0.979	0.035	0.514	0.525
ANN	6	1.133	0.229	1.347	0.214	0.946	0.007	0.978	0.019	0.437	0.349
ANN	7	1.011	0.184	1.194	0.140	0.946	0.008	0.978	0.020	0.402	0.280
ANN	8	0.961	0.133	1.105	0.094	0.946	0.008	0.975	0.016	0.385	0.247
ANN	9	0.896	0.069	1.009	0.062	0.947	0.007	0.970	0.012	0.365	0.216
ANN	10	0.908	0.185	1.061	0.109	0.947	0.008	0.976	0.024	0.368	0.230
KR	0.0025	0.985	0.026	1.347	0.131	0.928	0.029	0.982	0.004	0.454	0.328
KR	0.005	0.969	0.022	1.282	0.107	0.920	0.012	0.978	0.004	0.442	0.310
KR	0.010	0.929	0.021	1.207	0.103	0.923	0.008	0.977	0.004	0.419	0.282
KR	0.025	0.858	0.019	1.059	0.082	0.929	0.008	0.973	0.005	0.378	0.230
KR	0.050	0.831	0.019	0.970	0.059	0.938	0.007	0.969	0.005	0.355	0.202
KR	0.100	0.877	0.022	0.976	0.055	0.944	0.007	0.966	0.005	0.362	0.213
KR	0.150	0.999	0.031	1.087	0.059	0.943	0.008	0.964	0.007	0.405	0.270
KR	0.200	1.167	0.042	1.249	0.057	0.942	0.007	0.960	0.009	0.465	0.364
KR	0.250	1.355	0.049	1.433	0.052	0.941	0.007	0.957	0.009	0.533	0.487
KR	0.300	1.547	0.054	1.622	0.047	0.940	0.007	0.954	0.008	0.604	0.631
KR	0.400	1.913	0.058	1.983	0.039	0.941	0.008	0.953	0.008	0.743	0.955
KR	0.500	2.241	0.061	2.306	0.032	0.943	0.008	0.954	0.007	0.876	1.303
KR	0.750	2.900	0.076	2.951	0.017	0.942	0.009	0.947	0.009	1.138	2.161
KR	1.000	3.319	0.087	3.357	0.007	0.931	0.007	0.933	0.006	1.284	2.806
KR	2.000	3.747	0.109	3.770	0.001	0.928	0.008	0.929	0.004	1.440	3.541
LL	0.0025	1.546	0.221	2.628	0.843	0.968	0.019	0.984	0.005	0.576	0.911
LL	0.005	2.514	0.886	4.421	1.572	0.985	0.011	0.985	0.006	0.733	2.853
LL	0.010	3.029	1.192	5.756	2.250	0.990	0.009	0.987	0.006	0.802	4.893
LL	0.025	2.303	1.479	4.695	2.214	0.982	0.014	0.991	0.006	0.626	3.460
LL	0.050	1.057	0.237	1.546	0.580	0.952	0.015	0.989	0.005	0.414	0.436
LL	0.100	0.848	0.034	0.967	0.091	0.947	0.007	0.970	0.004	0.352	0.201
LL	0.150	0.827	0.018	0.909	0.044	0.948	0.006	0.966	0.004	0.340	0.186
LL	0.200	0.839	0.018	0.905	0.038	0.950	0.006	0.964	0.004	0.343	0.189
LL	0.250	0.901	0.021	0.962	0.026	0.947	0.006	0.962	0.005	0.364	0.219
LL	0.300	1.000	0.025	1.060	0.029	0.946	0.007	0.960	0.006	0.398	0.271
LL	0.400	1.289	0.035	1.350	0.034	0.944	0.006	0.955	0.004	0.499	0.444
LL	0.500	1.547	0.043	1.609	0.037	0.946	0.007	0.955	0.004	0.585	0.632
LL	0.750	1.976	0.057	2.035	0.028	0.949	0.006	0.955	0.003	0.748	1.020
LL	1.000	2.204	0.061	2.260	0.024	0.953	0.006	0.958	0.003	0.849	1.263
LL	2.000	2.670	0.073	2.716	0.019	0.964	0.005	0.967	0.003	1.055	1.824

* API – Analytic prediction intervals, BPI – Bootstrap prediction intervals, m - mean result, s - standard deviation of results, MAE – Mean Absolute Error, MSE – Mean Squared Error, Type for NNPLS refers to the number of latent variables, for ANNs refers to the number of hidden neurons, and for KR and LL refers to the bandwidth parameter. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

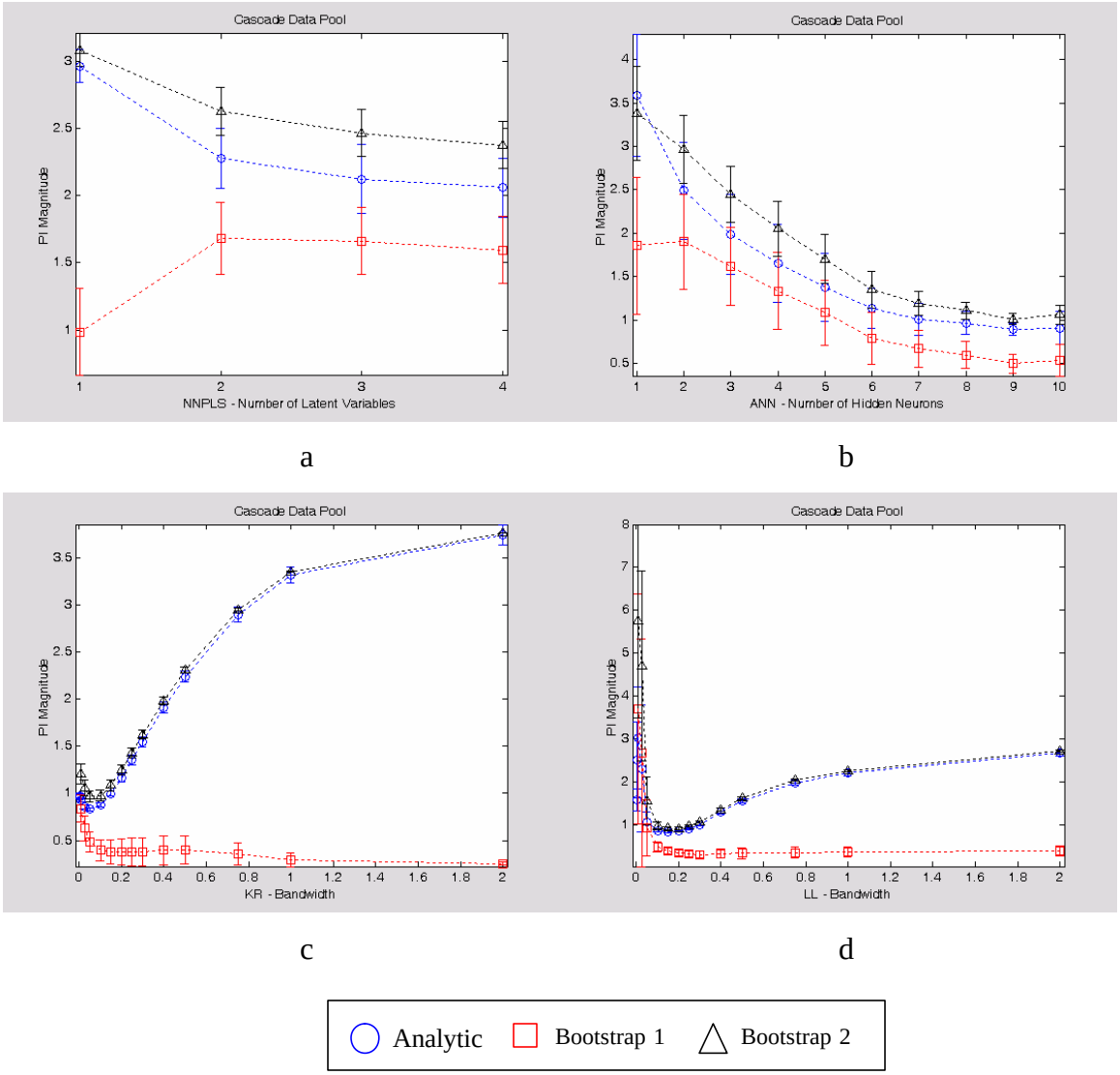


Figure 5.3.3: PI Magnitude Results for Cascade Data Pool.

The models used are specified along the x-axis. The bands represent the 1*s* variation in the results over the 100 iterations. Bootstrap 1 refers to the bootstrap intervals based on the average deviation bias estimate; whereas Bootstrap 2 refers to the bootstrap intervals based on the average squared error bias estimate (For more information, see section 4.10). NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

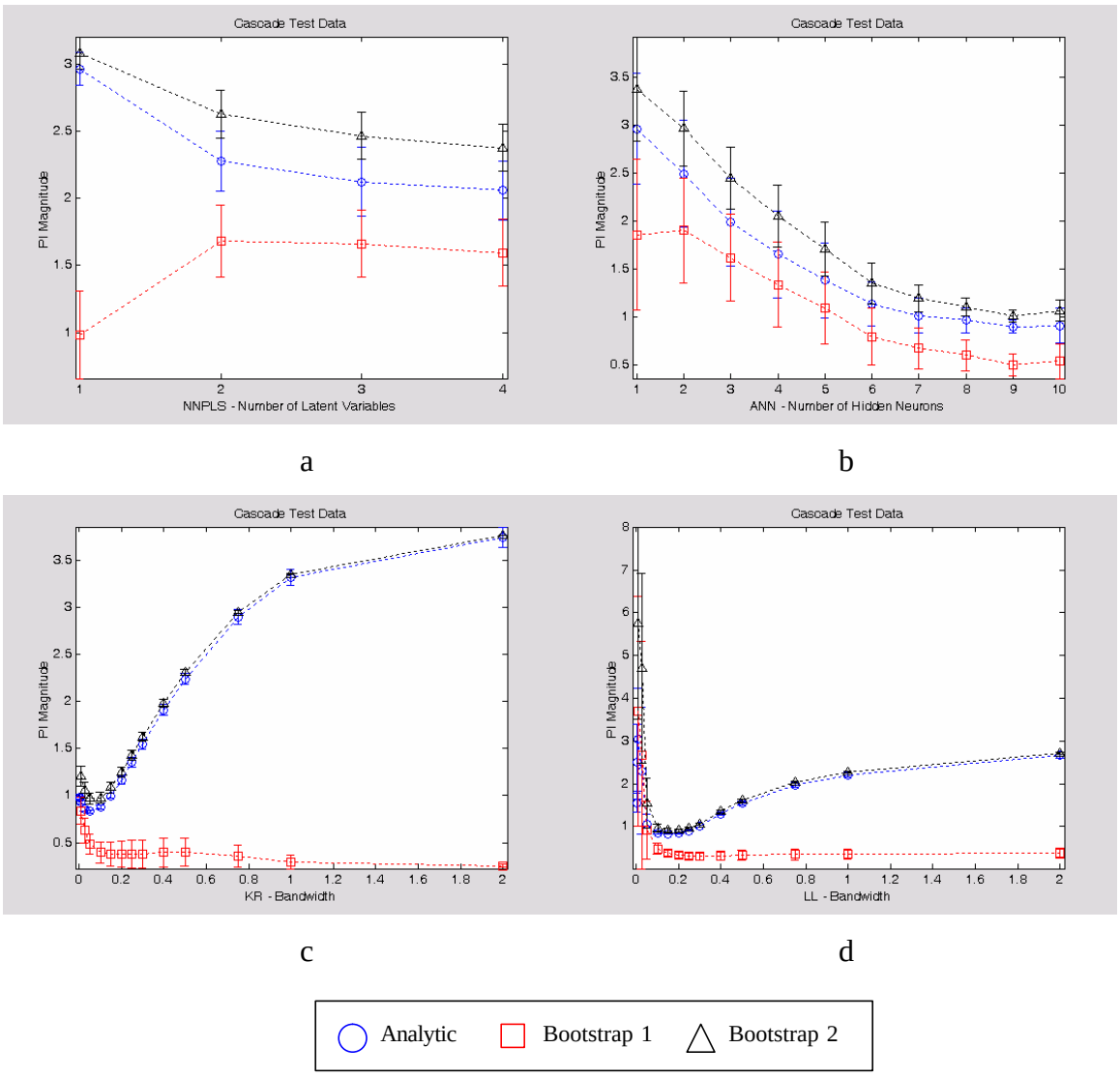


Figure 5.3.4: PI Magnitude Results for Cascade Test Data.

The models used are specified along the x-axis. The bands represent the 1*s* variation in the results over the 100 iterations. Bootstrap 1 refers to the bootstrap intervals based on the average deviation bias estimate; whereas Bootstrap 2 refers to the bootstrap intervals based on the average squared error bias estimate (For more information, see section 4.10). NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

continuous, there is a limit beyond which the variance of the PIMs will be constant or increase with complexity as the ANNs become overparametrized. Evidence of this is seen when going from a 9 hidden neuron model to a 10 hidden neuron model (figure 5.3.3.b and 5.3.4.b). For the KR models, the variance of the PIMs at first decreases with bandwidth, reaches a minimum, and then begins to slowly increase (figure 5.3.3.c and 5.3.4.c). This is due to the influence of the variance component for small bandwidths, and the influence of the bias for larger bandwidths in the prediction interval computation. One important point is that for very small bandwidth values there are some observations in both the data pool as well as in the test data for which estimates are unavailable. This is because the bandwidth values are too small and some observations result in all kernel weights being zero. The result of this is a division by zero in the kernel estimator and the production of a NaN (not a number) estimation for the given observation. These estimates are not included in the PIMs averages due to their obvious effects. For KR, as the bandwidth increases the number of observations for which NaN estimates result goes to zero. The maximum number of observations for which NaN estimates were obtained with respect to bandwidth for both KR and LL regression were: h=0.0025, 6 in the data pool, 4 in the test data | h=0.005, 3 in the data pool, 1 in the test data. Estimations were obtained for all observations for bandwidth values $\geq$ 0.01. The trend of variation in the PIMs for the LL models requires some additional interpretation (figure 5.3.3.d and 5.3.4.d). The discussion regarding NaN applies to LL regression as well. Before explaining the observed trend for the LL PIMs, consider table 5.3.3.

Table 5.3.3 provides the average maximum and minimum values of the KR and LL estimations over the 100 iterations for the cascade data pool. The minimum and maximum estimates from KR are bounded by the minimum and maximum response values in the training data set. This is not the case for LL regression. Recall that for h=0.0025 there were 6 observations in the data pool for which NaN estimates were obtained and for h=0.05 there were 3 estimates in the data pool for which NaN estimates were obtained. These estimations are not included in the PIMs calculations, resulting in a lower than expected variation in the PIMs results. Considering table 5.3.3 and the results

Table 5.3.3: Average minimum and maximum cascade data pool estimates for LPR models over 100 iterations

Bandwidth	KR Estimates		LL Estimates	
	Average Minimum	Average Maximum	Average Minimum	Average Maximum
0.0025	-1.535	7.235	-9.320	13.762
0.005	-1.491	7.171	-19.236	23.776
0.010	-1.379	7.047	-28.700	26.238
0.025	-1.164	6.865	-18.428	15.491
0.050	-0.911	6.751	-3.730	8.423
0.100	-0.671	6.654	-1.075	7.112
0.150	-0.581	6.511	-0.900	7.076
0.200	-0.516	6.293	-0.814	6.850
0.250	-0.443	6.039	-0.717	6.868
0.300	-0.347	5.777	-0.702	6.908
0.400	-0.074	5.267	-0.689	7.042
0.500	0.259	4.752	-0.644	7.571
0.750	0.704	3.489	-0.653	8.722
1.000	0.946	2.689	-0.065	9.024
2.000	1.401	1.854	-0.516	7.604

in table 5.3.1, it is observed that the trend of the variation in the PIMs is analogous to the trend of the ranges of the average minimum and maximum values with respect to bandwidth. That is, initially (h=0.0025 and h=0.005) the range of estimations is moderate noting that the observations which would dominate this range are not being included in the calculations at this point because of the NaN estimation results. For h=0.01 and h=0.025, the range of the estimations increases because the observations corresponding to the previous NaN estimates for smaller bandwidths are now being complemented by *real* estimates. These *real* estimates provide the maximum and minimum values for the range of estimations. As the bandwidth increases further, more and more training observations fall within the neighborhood of influence for the corresponding responses and these effects are eventually reduced. For bandwidth values of $h \geq 0.1$, these effects are sufficiently muted. This phenomenon occurs for the cascade test data set as well (table 5.3.2, figure 5.3.4.d).

Consider the value of the PIMs for the different models and architectures illustrated in figures 5.3.3 and 5.3.4. The NNPLS PIMs (figure 5.3.3a and 5.3.4.a) were minimized for the 4 latent variable models. The cascade data set is not an ideal candidate data set for the NNPLS architecture due to the minimal correlations between the predictor variables. It has been reported that optimal performance of the NNPLS model for signal validation tasks relies on strong correlations between the predictor variables [Rasmussen 2000a, Rasmussen 2000b]. The ANN PIMs (figure 5.3.3.b and 5.3.4.b) continually decrease to a minimum value for 9 hidden neurons, the more complex 10 hidden neuron models exhibit an increase in the PIMs due to the addition of unnecessary parameters. The KR PIM values (figure 5.3.3.c and 5.3.4.c) indicate a trend that is expected based on the known tradeoff between bias and variance, this effect is mimicked for the LL models (figure 5.3.3.d and 5.3.4.d) with the initial increase and subsequent decrease being accounted for by the effects of the previous discussions regarding the variability of the LL estimates with respect to the bandwidth parameter.

Comparing the results of the bootstrap 2 PIMs to those from the analytic PIMs, the bootstrap 2 PIMS are consistently larger though in many cases the difference is marginal. In addition the difference reaches its minimum in the vicinity of the proper architecture for a given model, e.g. the difference between the bootstrap 2 PIM and the analytic PIM for the 9 hidden neuron ANN model for the cascade data pool (figure 5.3.3.b) is at its minimum. It is noted that the differences between the bootstrap2 and analytic PIMs are greater for the ANN and NNPLS models than for the KR and LL models. Because both the bootstrap 2 and analytic prediction interval computations utilize a very similar bias estimate for KR and LL, than it is observed that the bootstrap 2 variance estimate very closely parallels that from the analytic variance estimate. For the ANN and NNPLS models, the bias is not explicitly accounted for in the analytic prediction interval computations, though indirectly some of the bias will be accounted for by the incorporation of the validation data set into the variance estimate s^2 (see sections 4.6 and 4.7). Thus, the bootstrap intervals reflect this, and their magnitudes are consistently higher.

The coverage of the prediction intervals for the cascade data pool and test data are presented numerically in tables 5.3.1 and 5.3.2, and graphically in figure 5.3.5 and 5.3.6. Quickly viewing the graphs it is obvious that the bootstrap 1 prediction intervals result in insufficient coverage. The bootstrap 1 results will not be discussed further in this section, and will not be implemented for the remaining data sets. Viewing all of the coverage results over all architectures and models for both the data pool and the test data, it is evident that the expected coverage is achieved. This statement is disregarding the results from the bootstrap 1 prediction intervals and their corresponding coverage values. The tabulated results of tables 5.3.1 and 5.3.2 provide the numerical values plotted in figure 5.3.5 and 5.3.6. While there are a few cases where the coverage is slightly below 0.95, the deviation is minimal. It is also noted that the variation in the coverage values from 1 iteration to the next is small, indicating that the prediction interval coverage is consistent. Comparing the bootstrap 2 prediction interval coverage with the analytic prediction interval coverage, for all models and architectures (figures 5.3.5 and 5.3.6), the bootstrap

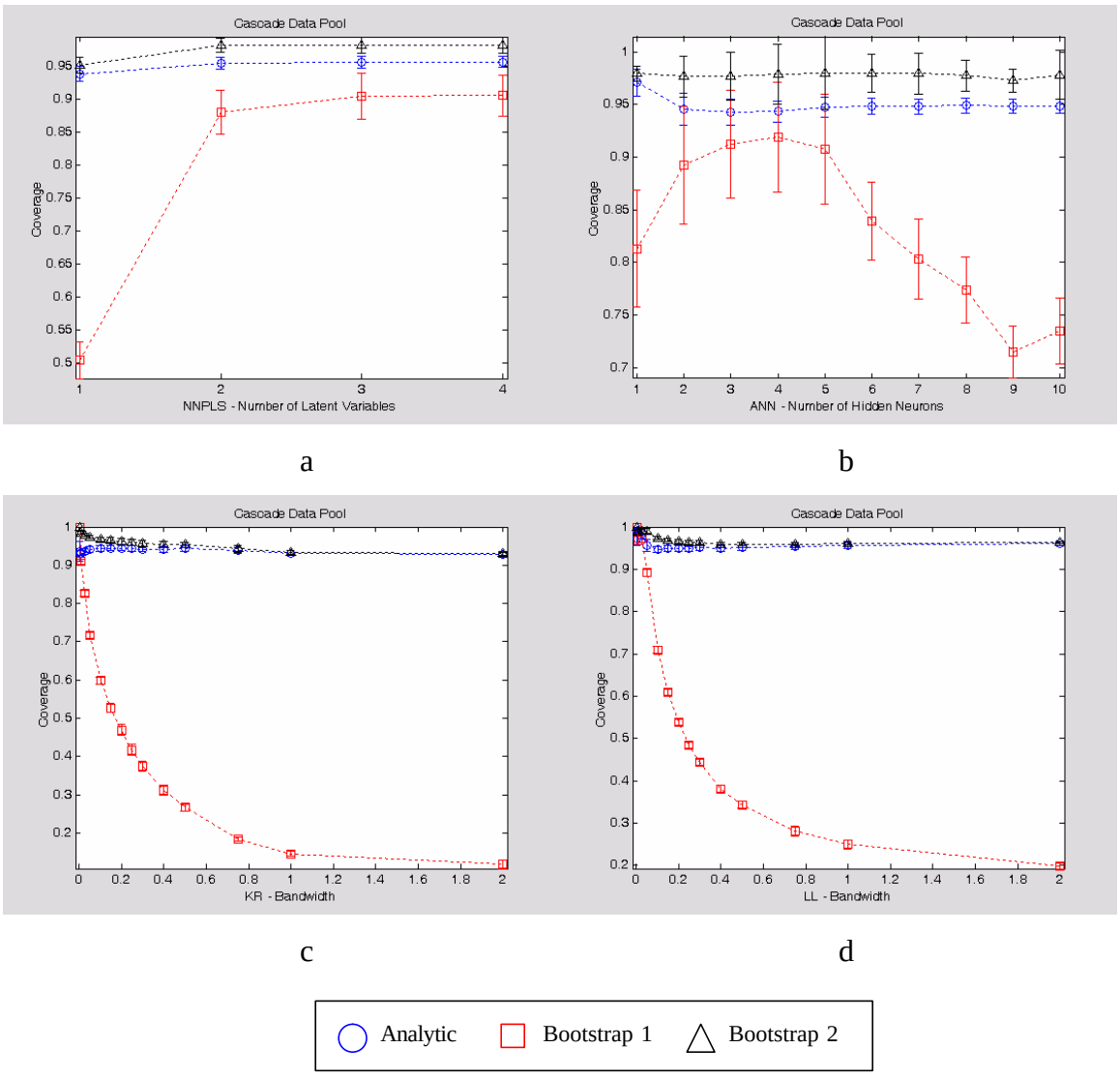


Figure 5.3.5: Coverage Results for Cascade Data Pool.

The models used are specified along the x-axis. The bands represent the 1*s* variation in the results over the 100 iterations. Bootstrap 1 refers to the coverage of the bootstrap intervals based on the average deviation bias estimate; whereas Bootstrap 2 refers to the coverage of the bootstrap intervals based on the average squared error bias estimate (For more information, see section 4.10). NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

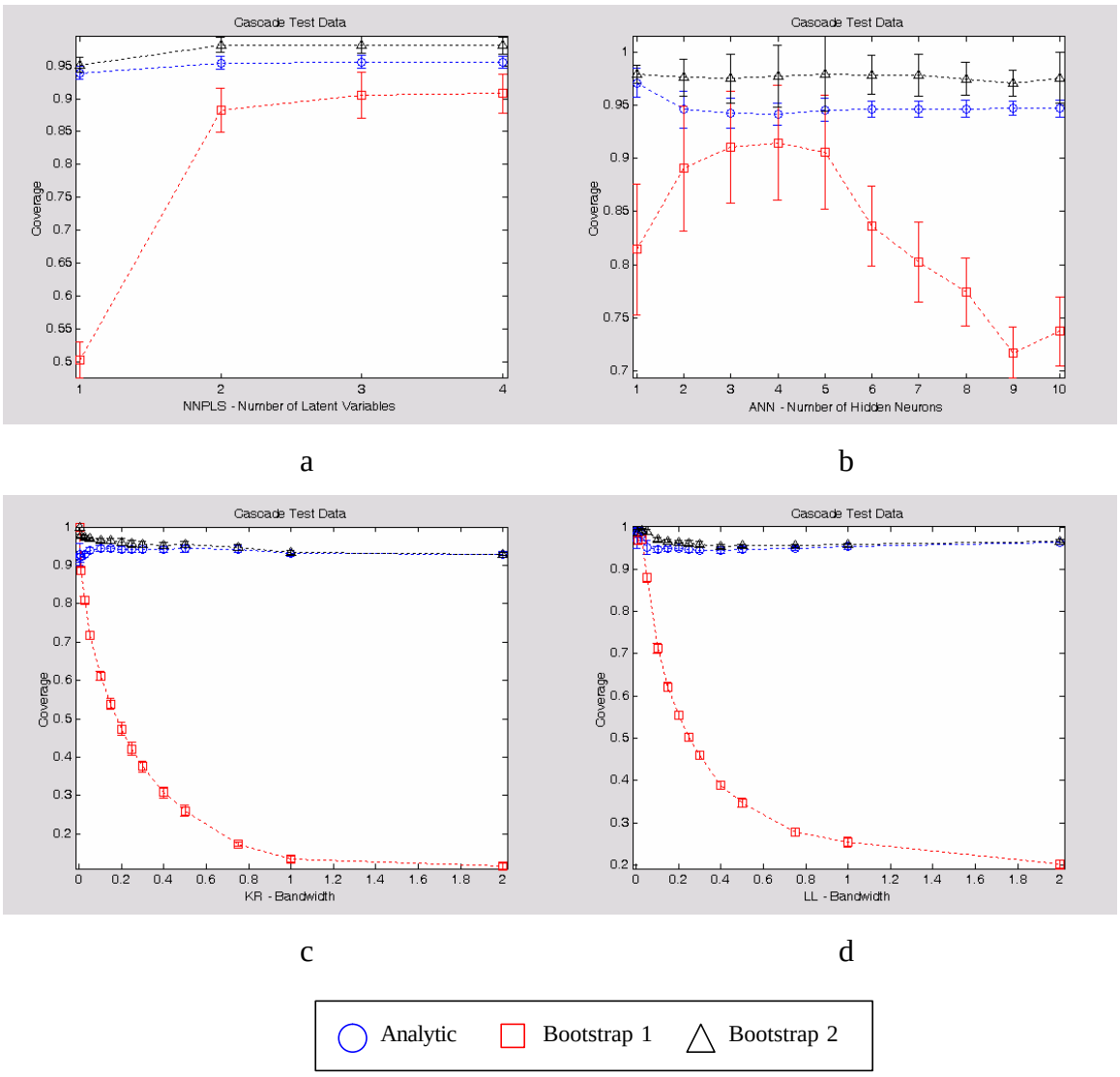
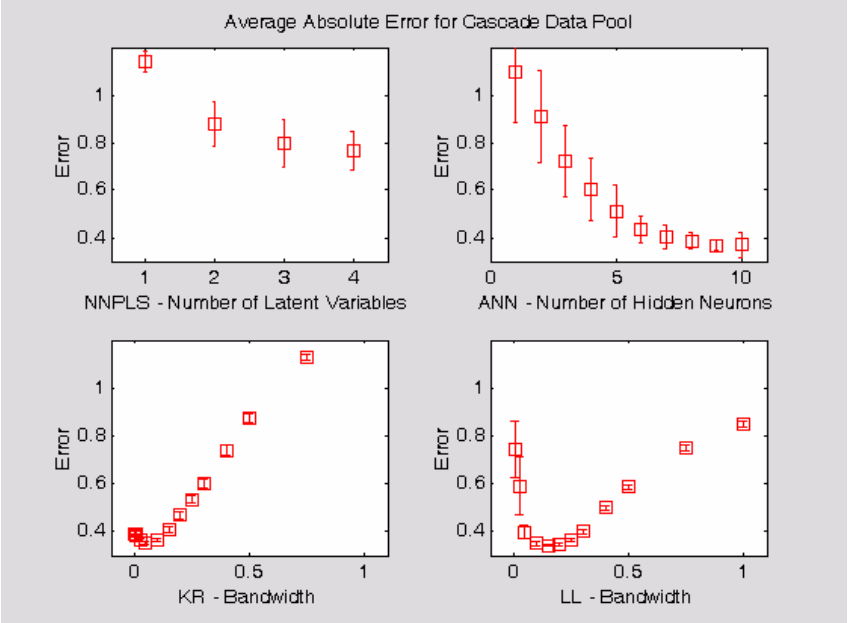


Figure 5.3.6: Coverage Results for Cascade Test Data.

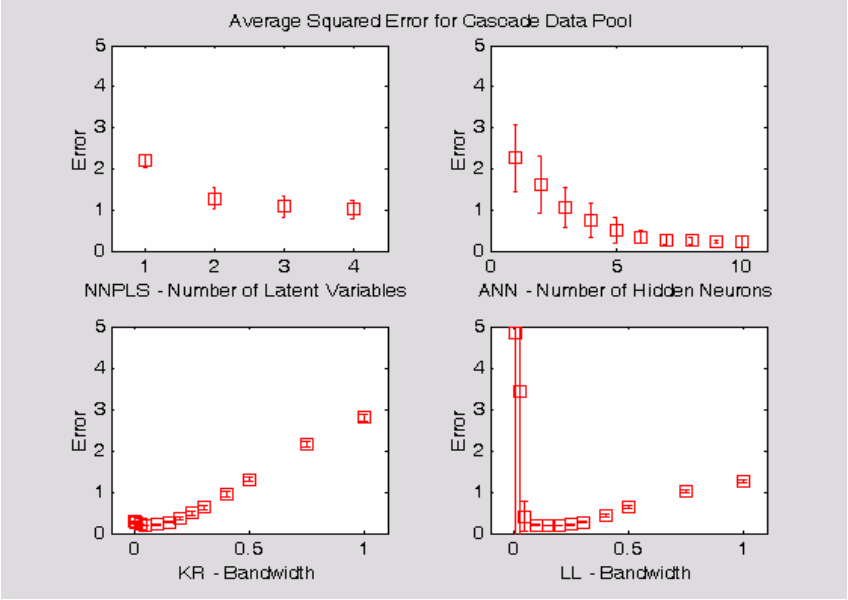
The models used are specified along the x-axis. The bands represent the 1*s* variation in the results over the 100 iterations. Bootstrap 1 refers to the coverage of the bootstrap intervals based on the average deviation bias estimate; whereas Bootstrap 2 refers to the coverage of the bootstrap intervals based on the average squared error bias estimate (For more information, see section 4.10). NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

coverage values are slightly higher, though both methods result in sufficient coverage. The lower values for the analytic coverage values for the KR and LL models at low bandwidths indicate that the bootstrap bias estimate is a better estimate of the *true* bias of the LPR models for these low bandwidths because it is less influenced by the overfit of the model to the training data. While the analytic bias estimate does consider the validation data in its bias estimate, the bootstrap bias estimate considers the entire data pool.

The MAE and MSE results for the cascade data pool and test data are provided numerically in tables 5.3.1 and 5.3.2, and graphically in figures 5.3.7 and 5.3.8. The results for both the data pool and test data exhibit similar trends, whereby the average results are slightly higher for the test data with respect to the data pool, which is expected. Also, the variation in the results is greatest for the ANN and LL models. Note that the KR models would exhibit similar trends if the NaN estimates were included in the computations. The ANN models exhibit more variation than the NNPLS models due to the inherent regularization of the NNPLS for models where the number of latent variables is less than the number of predictor variables. Eventually when the number of hidden neurons in the ANN models was sufficient for an adequate fit to the data, the variability in the ANN estimates was lower than that for the NNPLS models. Thus, the bias inserted into the NNPLS models provides an initial stabilization, though its benefits are eventually overcome when the ANN models are sufficiently complex for the task. The large variation for the LL models was due to the under-regularization of small bandwidth models resulting in a large range of possible estimates for a given observation. For reasonable bandwidth values this effect is eliminated. The large variation does not occur in the KR error results because of the elimination of the NaN estimates from the calculations as previously discussed. The MSE values for all models are presented due to the common use of this quantity to report model fit. The trends for the MSE values follow those for the MAE values, though the effects noted for the MAE values are exaggerated in the MSE plots.



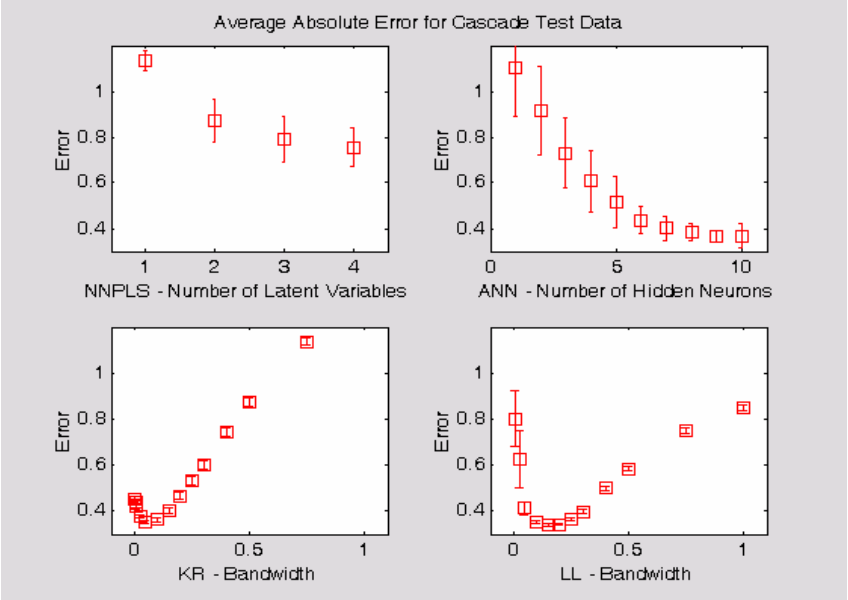
a



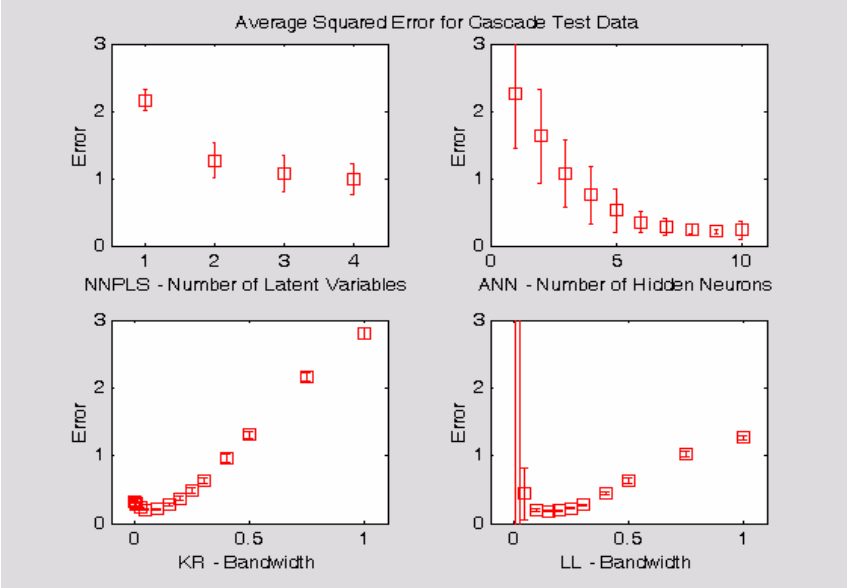
b

Figure 5.3.7: Error Results for Cascade Data Pool.

Plot (a) shows the overall Mean Absolute Error, and plot (b) shows the overall Mean Squared Error. The models used are specified along the x-axis. The bands represent the 1*s* variation in the results over the 100 iterations. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.



a



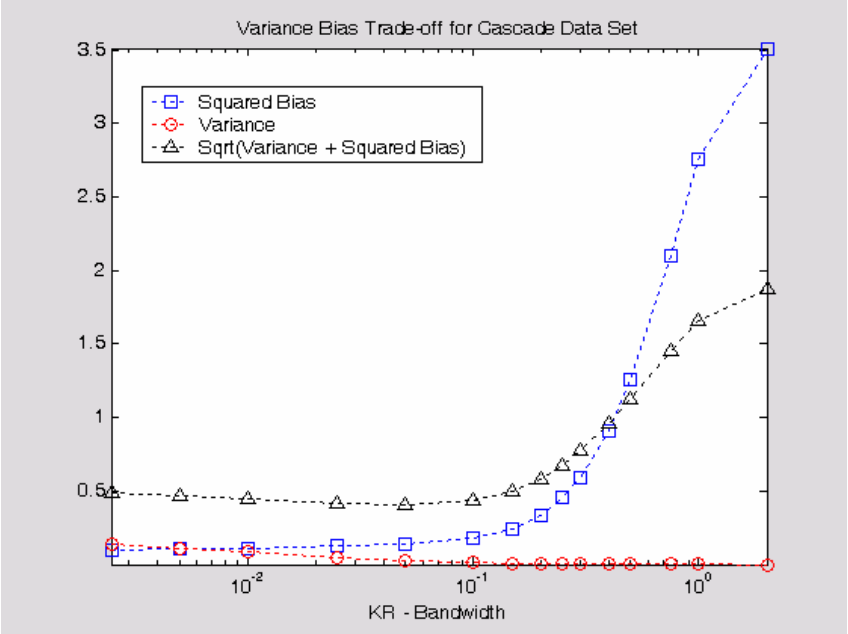
b

Figure 5.3.8: Error Results for Cascade Test Data.

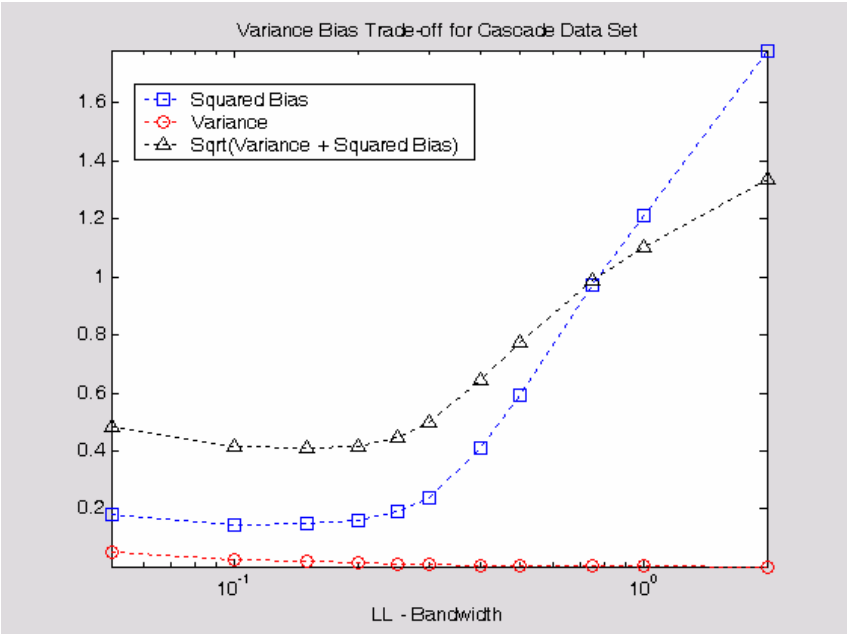
Plot a shows the overall Mean Absolute Error, and plot b shows the overall Mean Squared Error. The models used are specified along the x-axis. The bands represent the 1*s* variation in the results over the 100 iterations. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

Considering all of the results for the cascade data presented thus far, the optimal NNPLS architecture was determined to be the 4 latent variable architecture and the optimal ANN architecture was that containing 9 hidden neurons. Selection of the optimal bandwidth for the KR and LL models is based on the bias-variance trade-off of nonparametric models. Figure 5.3.9a shows the average variance of the KR estimator based on the training data along with the squared bias estimate based on the combined set of training and validation data. Due to the incorporation of the validation data into the squared bias estimate, the squared bias does not reach zero. The optimal bandwidth is located by minimizing the sum of the squares of the variance and squared bias components. The minimum for the KR models occurred for h=0.05. Analogous results are provided for the LL models (figure 5.3.9.b). The optimal bandwidth for the LL models was h=0.15. An important observation here is that the bias based on both the training and validation data is higher for the LL models at lower bandwidth values than for the KR models, and lower for the LL models at higher bandwidth values than for the KR models. This contradicts the notion that theoretically a LL model is less biased than a KR model. This is due to the use of the validation data in the bias estimate. The smaller bias for KR with respect to LL is observable at small bandwidth values if only the training data are considered. It is assumed that the larger observed bias for the LL models at low bandwidths is due to the inadequacy of the training data density for these low bandwidth models. The result of this is a large variability in the LL estimates at low bandwidths leading to the larger bias estimates. For all bandwidth values $h \geq 0.1$, the expected observation of smaller bias for LL models with respect to KR models was exhibited.

Figure 5.3.10 provides the distribution of the coverage values of the analytic prediction intervals for the optimally determined architectures and bandwidths for the cascade test data set. Figure 5.3.11 shows analogous results for the bootstrap prediction intervals. Viewing the results, it seems that the bootstrap prediction intervals result in coverage values slightly higher than the expected value of 0.95, while the distribution of the analytic prediction interval coverage values are more centered at the expected value of 0.95. The noted exception was that for the KR models, where the coverage distribution



a



b

Figure 5.3.9: Bias and Variance plots for cascade data.

KR (a), and LL (b) as a function of bandwidth. KR – Kernel Regression, LL – Local Linear Regression.

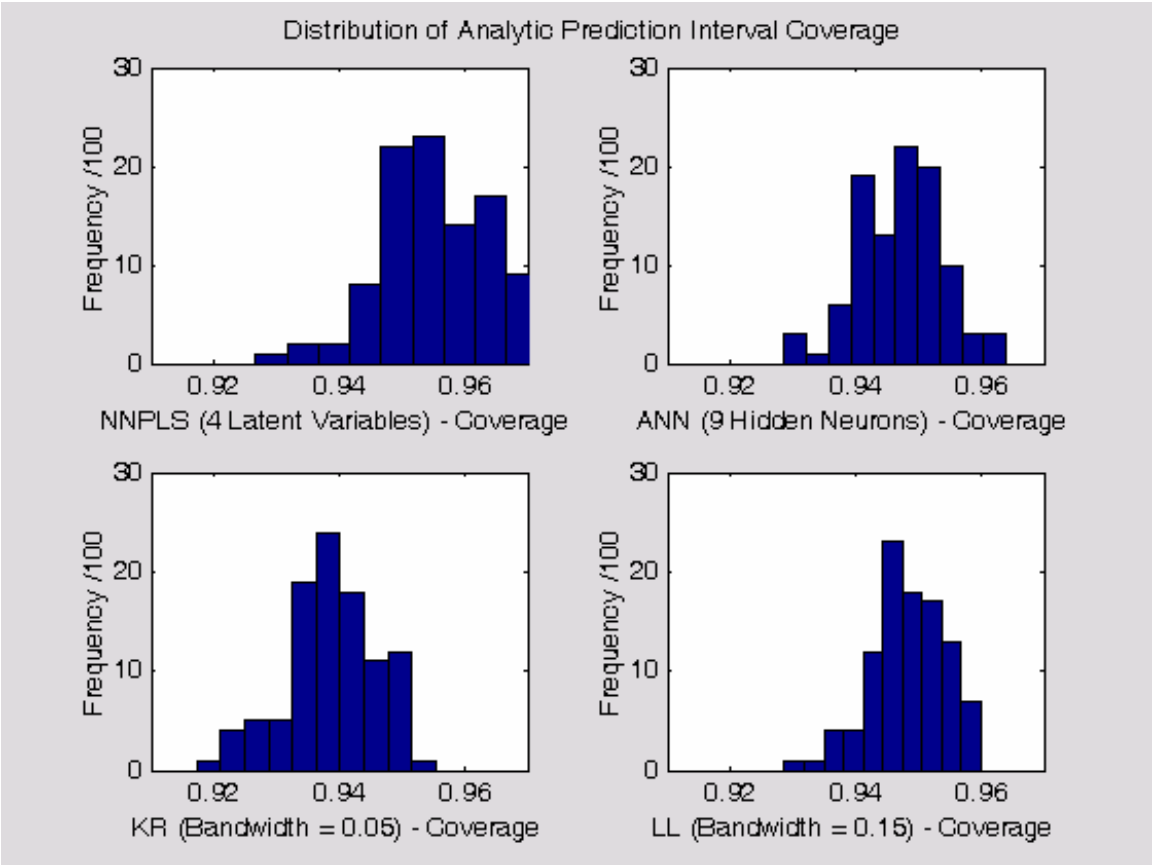


Figure 5.3.10: Analytic prediction interval coverage values for cascade test data set. The plot shows the distributions of the coverage values for the analytic prediction interval methods over 100 iterations. These coverage values are with respect to the Cascade Test Data. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

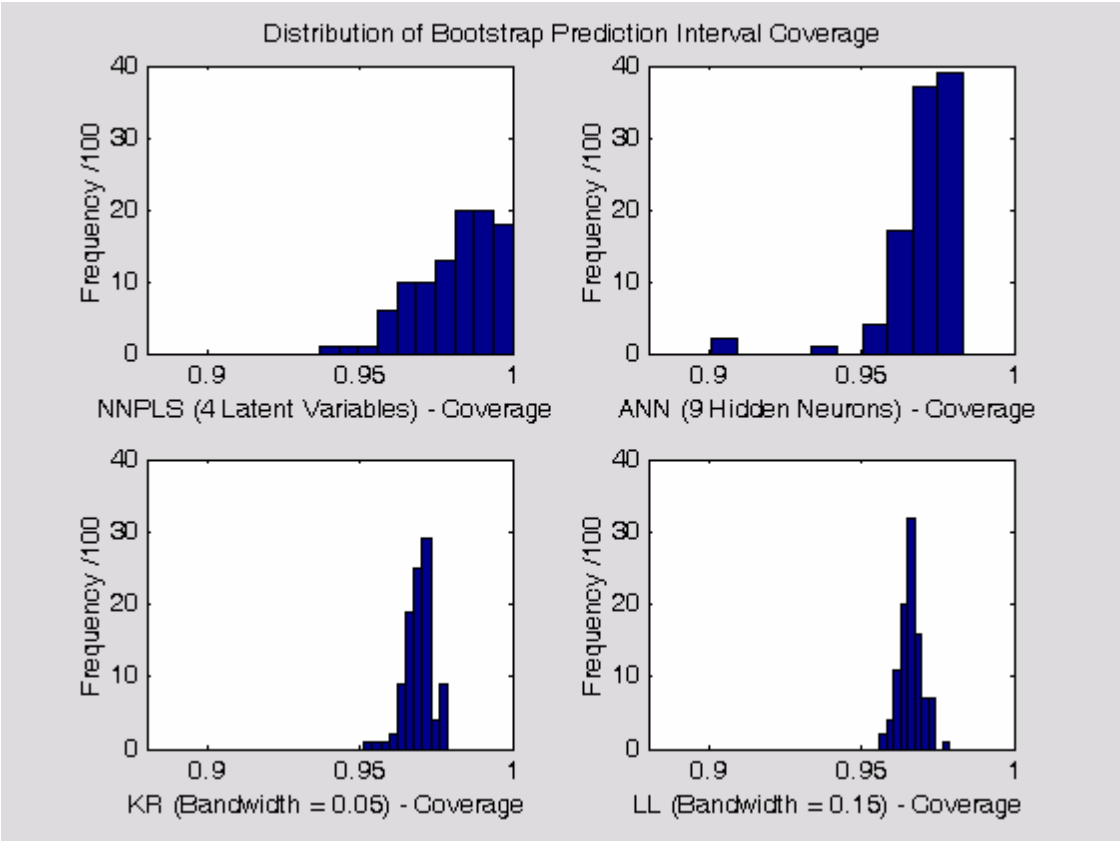


Figure 5.3.11: Bootstrap prediction interval coverage values for cascade test data set. The plot shows the distributions of the coverage values for the bootstrap prediction intervals over 100 iterations. These coverage values are with respect to the Cascade Test Data, and were based on the coverage of the bootstrap prediction intervals using the squared error bias estimate. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

was centered slightly below the expected value of 0.95. Considering the results of tables 5.3.1 and 5.3.2 it is observed that the average coverage for the KR models of the cascade data set analytic prediction intervals were all slightly below 0.95. Considering that even in regions where the PIMs are much larger for the KR models than they are for the LL models the resultant coverage values are still lower for the KR models, the only result that can be drawn is that the LL models provide a better fit to the data. The slightly lower coverage values for the KR models in this case may be due to an inadequate bias estimate in regions of proper bandwidth where the fit of the KR models is optimized, and for higher bandwidths when the bias estimate is relatively large, the slightly reduced coverage values are due to the over-regularized fit. While this minor underestimation for the coverage of the KR analytic prediction intervals is noted here, further investigations are warranted to obtain a better understanding of the root cause of the discrepancy.

To provide illustrative results for the cascade test data, 4 of the 100 models of each optimal architecture were selected and their estimations plotted along with the analytic prediction intervals. Figure 5.3.12 provides the random results for the NNPLS models, figure 5.3.13 for the ANN models, figure 5.3.14 for the KR models, and figure 5.3.15 for the LL models.

The random results are fairly similar for the most part, with the exceptions being the results for the NNPLS models which in the case of the cascade data set were not able to provide sufficiently certain fits to the data. To better illustrate, a representative result for each model at its optimal architecture, is provided in figures 5.3.16-5.3.19. The representative result for each case was selected as the model of optimal architecture among the 100 models for that architecture which exhibited the median MAE result for the cascade test data.

To study the effects of erroneous predictor variables on the magnitude of the prediction intervals, 2 additional predictor variables were inserted, while the response variable remained unchanged. The purpose of this study was to evaluate the effects of a type of

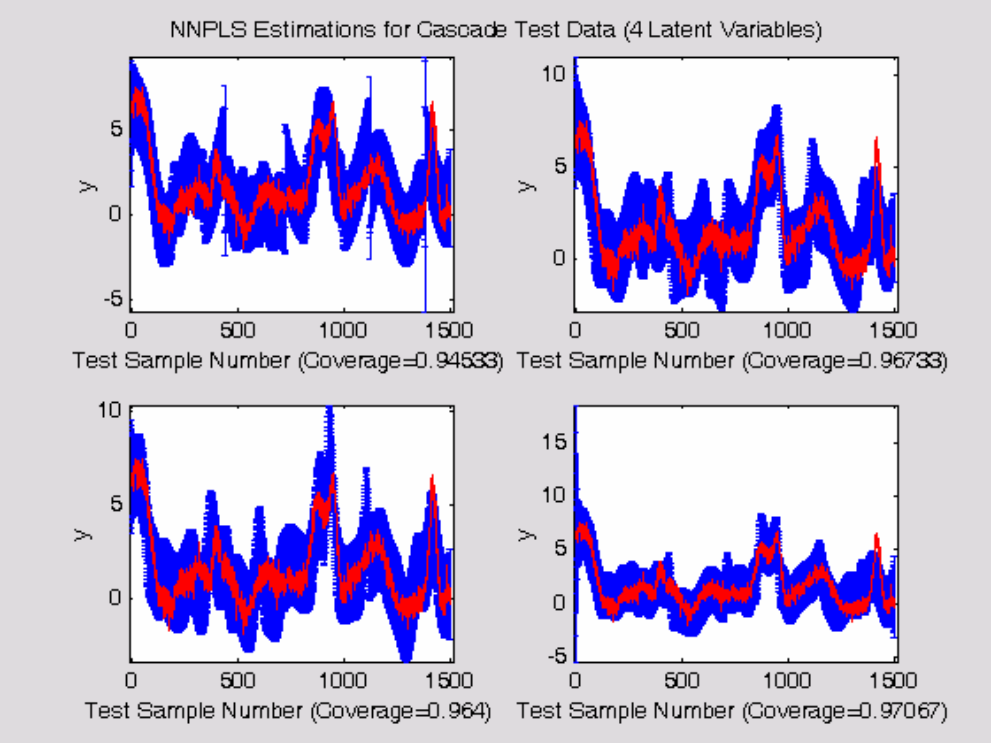


Figure 5.3.12: NNPLS cascade test data estimations and analytic prediction intervals. Test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 models. The coverage values for each are shown along the x-axis. NNPLS – Neural Network Partial Least Squares.

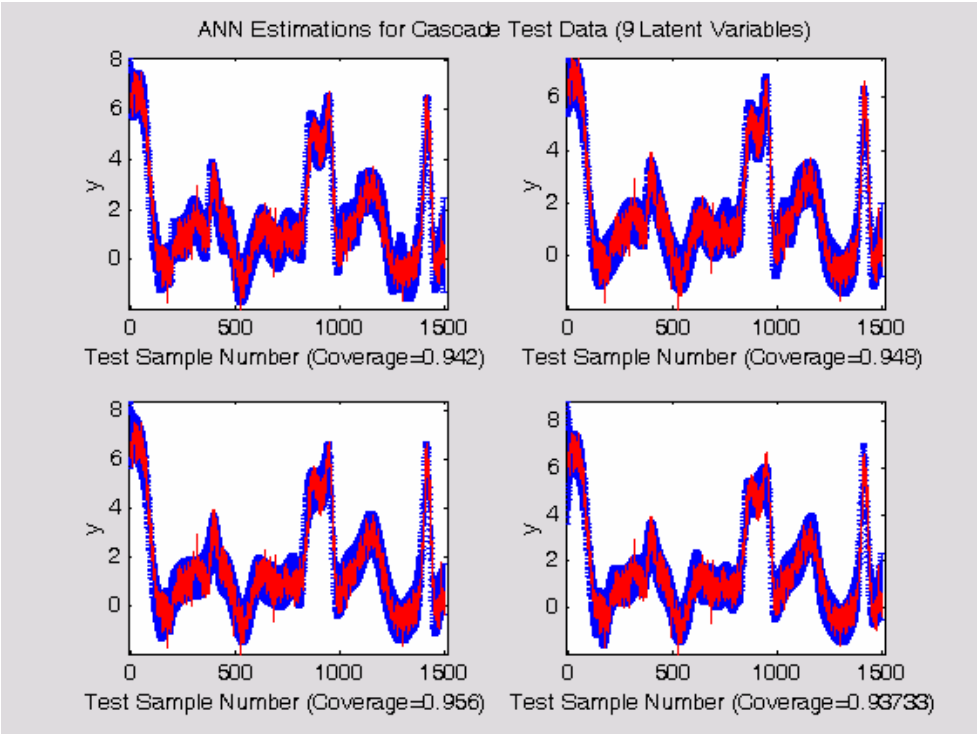


Figure 5.3.13: ANN cascade test data estimations and analytic prediction intervals. Test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 models. The coverage values for each are shown along the x-axis. ANN – Artificial Neural Network.

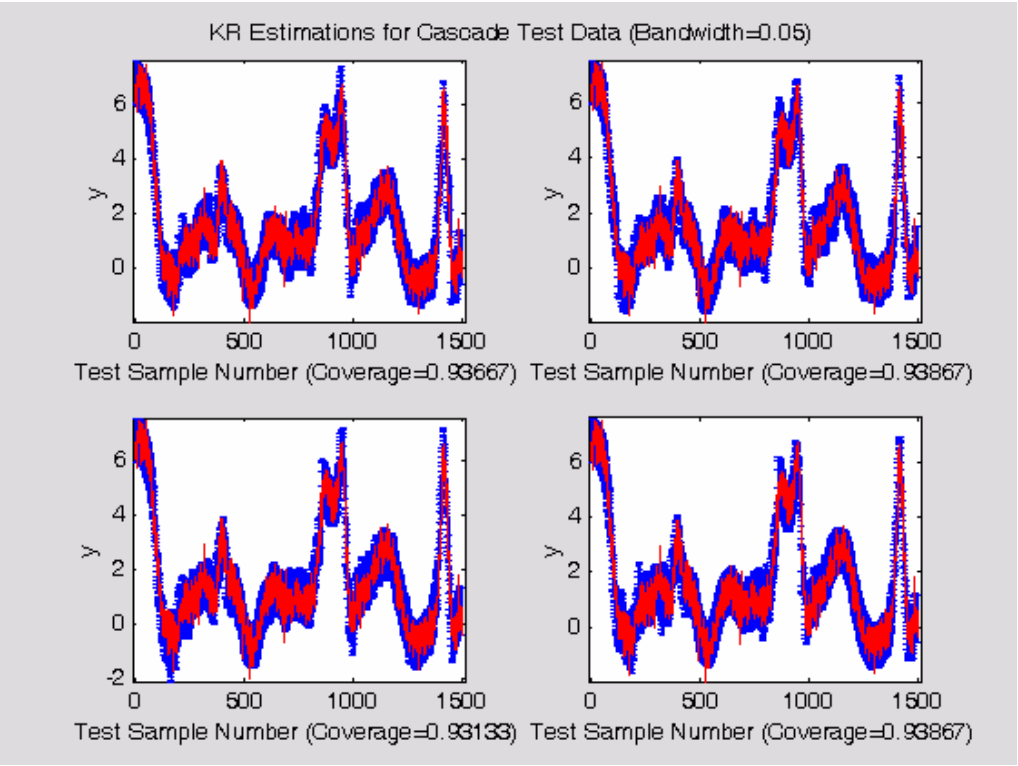


Figure 5.3.14: KR cascade test data estimations and analytic prediction intervals. Test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 models. The coverage values for each are shown along the x-axis. KR – Kernel Regression.

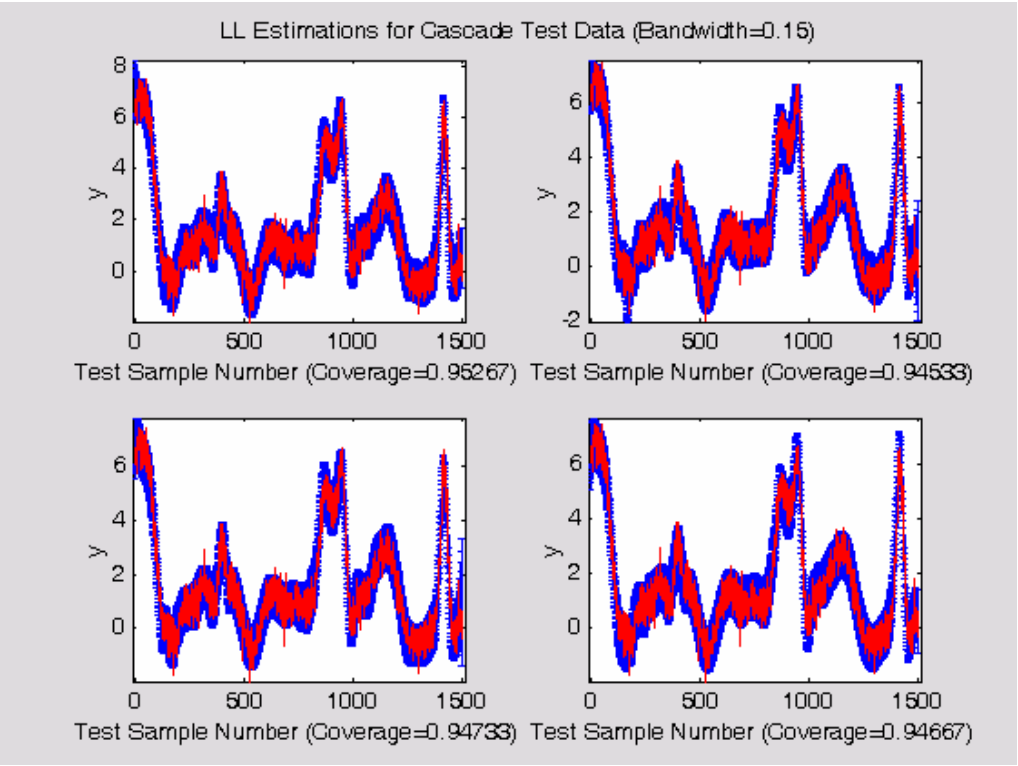


Figure 5.3.15: LL cascade test estimations and analytic prediction intervals.

Test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 models. The coverage values for each are shown along the x-axis. LL – Local Linear Regression.

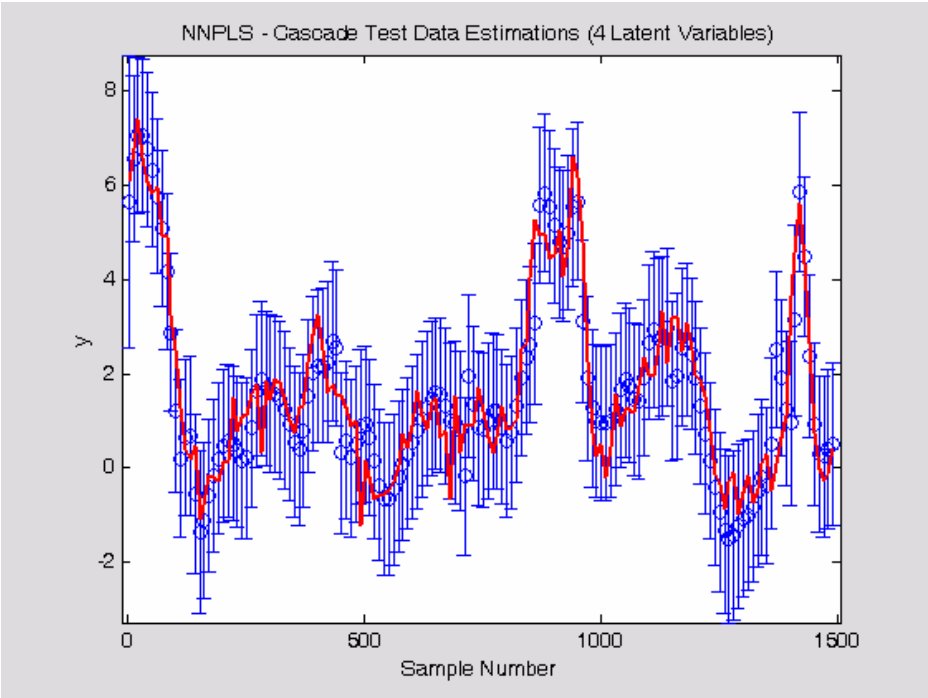


Figure 5.3.16: NNPLS median result estimation and analytic prediction intervals. Representative result for NNPLS 4 latent variable model showing every 10th estimate and the corresponding analytic prediction intervals. This model exhibited the median value of the 100 Mean Absolute Error results for the 4 latent variable models. NNPLS – Neural Network Partial Least Squares.

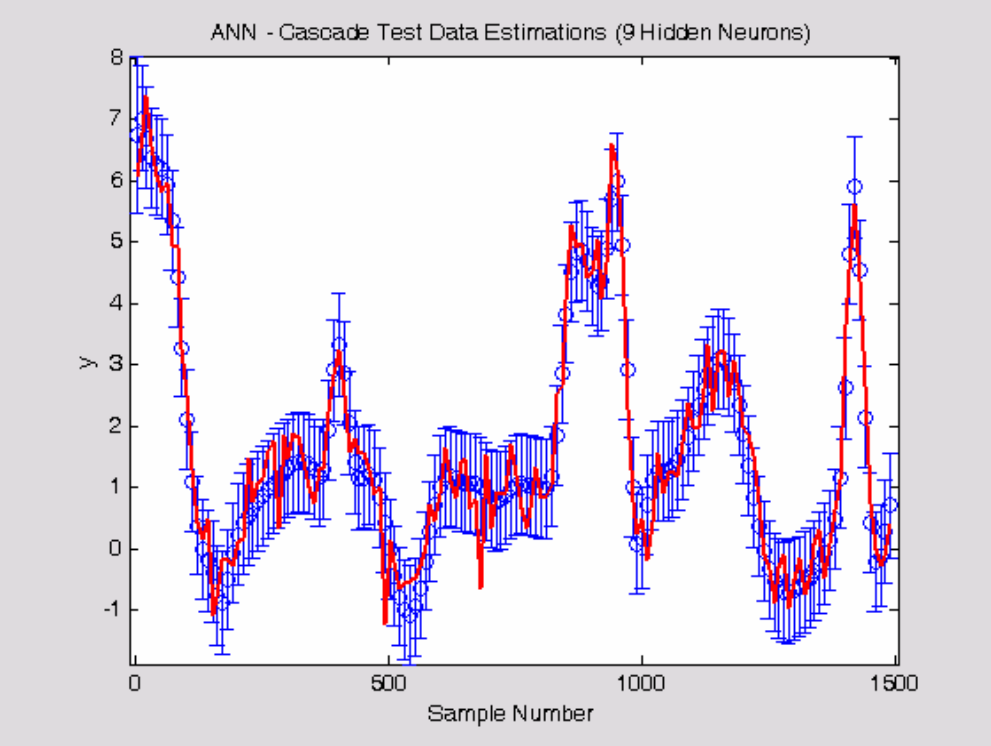


Figure 5.3.17: ANN median result estimation and analytic prediction intervals Representative result for ANN 9 hidden neuron model showing every 10th estimate and the corresponding analytic prediction intervals. This model exhibited the median value of the 100 Mean Absolute Error results for the 9 hidden neuron models. ANN – Artificial Neural Network.

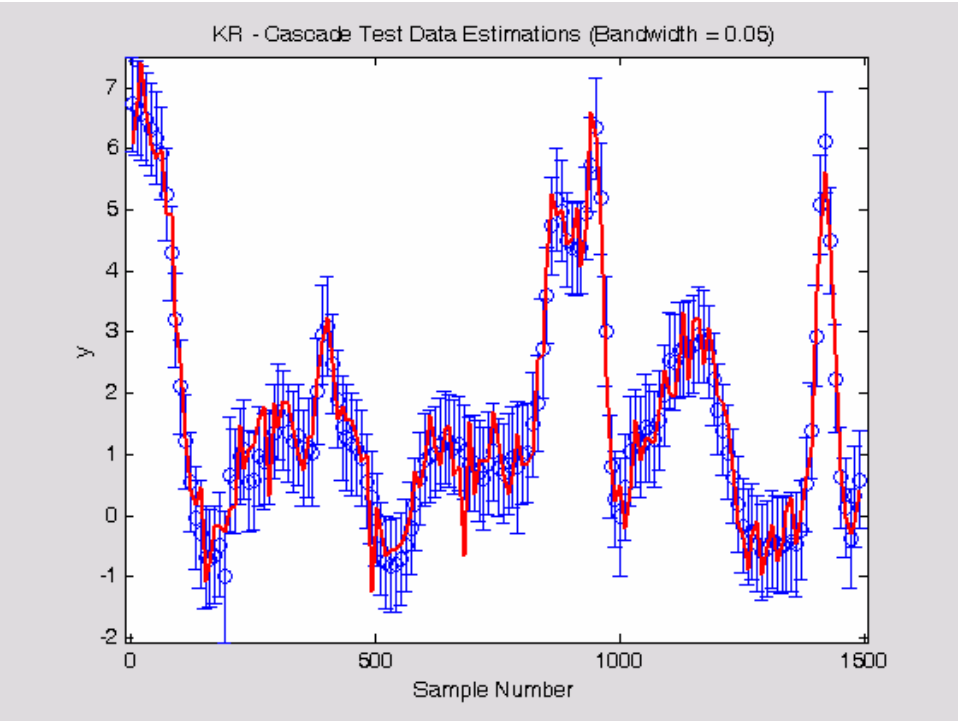


Figure 5.3.18: KR median result estimation and analytic prediction intervals Representative result for KR model (Bandwidth = 0.05) showing every 10^{th} estimate and the corresponding analytic prediction intervals. This model exhibited the median value of the 100 Mean Absolute Error results for the Bandwidth=0.05 models. KR – Kernel Regression.

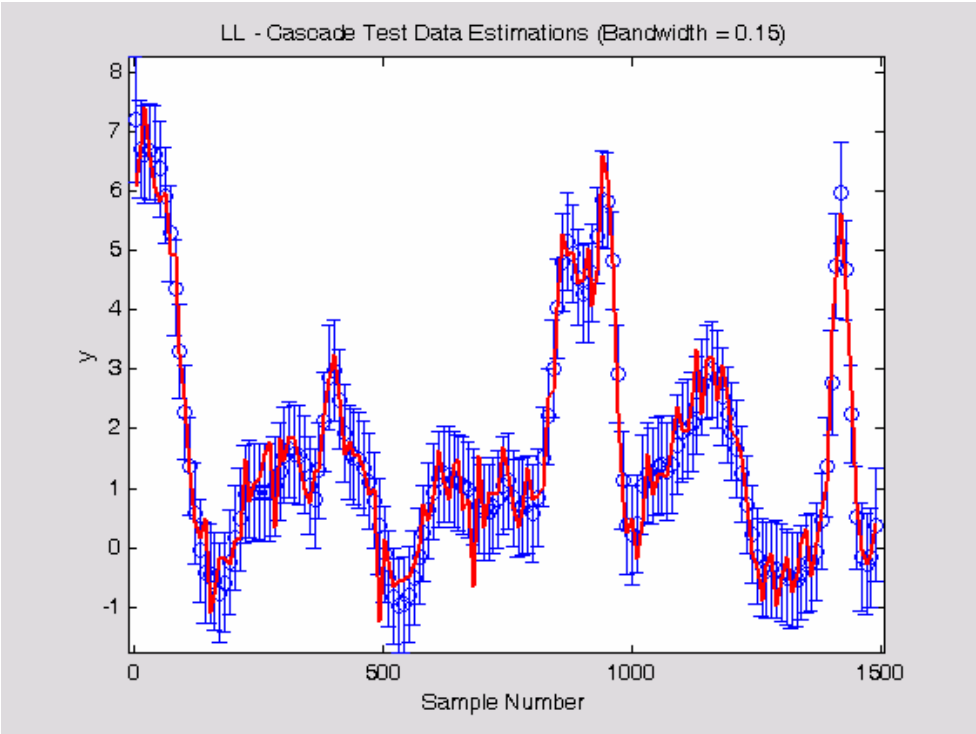


Figure 5.3.19: LL median result estimation and analytic prediction intervals

Representative result for LL model (Bandwidth = 0.15) showing every 10th estimate and the corresponding analytic prediction intervals. This model exhibited the median value of the 100 Mean Absolute Error results for the Bandwidth=0.15 models. LL – Local Linear Regression.

model misspecification on the prediction interval computations and their resultant coverage values. In this case the type of misspecification is due to the selection of predictor variables. Recall the response variable was shown in figure 5.3.1, and the original predictors were shown in figure 5.3.2. The new set of predictors contained two noise signals drawn from $N(0,\sqrt{0.1})$, i.e. the 2 additional erroneous inputs are normal random variables with zero means, and $s^2 = 0.1$. The augmented cascade predictor variables are shown in figure 5.3.20. To be clear in the discussions, the normal set of data with 4 predictors will be referred to as cascade1, whereas the set of data including the 2 erroneous predictors will be referred to as cascade2.

Comparing the results for the cascade2 data with respect to the results for the cascade1 data, the NNPLS PIMs for the analytic prediction intervals as well as for the bootstrap intervals was consistently higher for all model architectures (figure 5.3.21.a). This was the expected result indicating that the prediction intervals increase when nuisance parameters are inserted into the predictor variable set. The larger deviation in the analytic results as the number of latent variables increases is expected of the NNPLS architecture, since for 1 latent variable the majority of influence of the erroneous predictors is transferred to the higher latent variables. As the number of latent variables increases, the influences of the erroneous predictors increases as more of their content is included in the overall model fit; and thus the deviation in the PIMs for the two data sets increases. The ANN PIMs for cascade1 and cascade2 are shown in figure 5.3.21.b. While the bootstrap intervals show a significant increase, especially with increasing model complexity, the analytic intervals for both data sets are consistently similar. The KR and LL PIM results for the cascade1 and cascade2 data sets are shown in figures 5.3.21.c and 5.3.21.d. The bootstrap prediction interval magnitudes for both LPR models exhibited significant increases, especially at lower bandwidth values. While, the analytic prediction interval magnitudes also exhibited increases, they were not as drastic, and the results for both the cascade1 and cascade2 data sets quickly converged as the bandwidth increased.

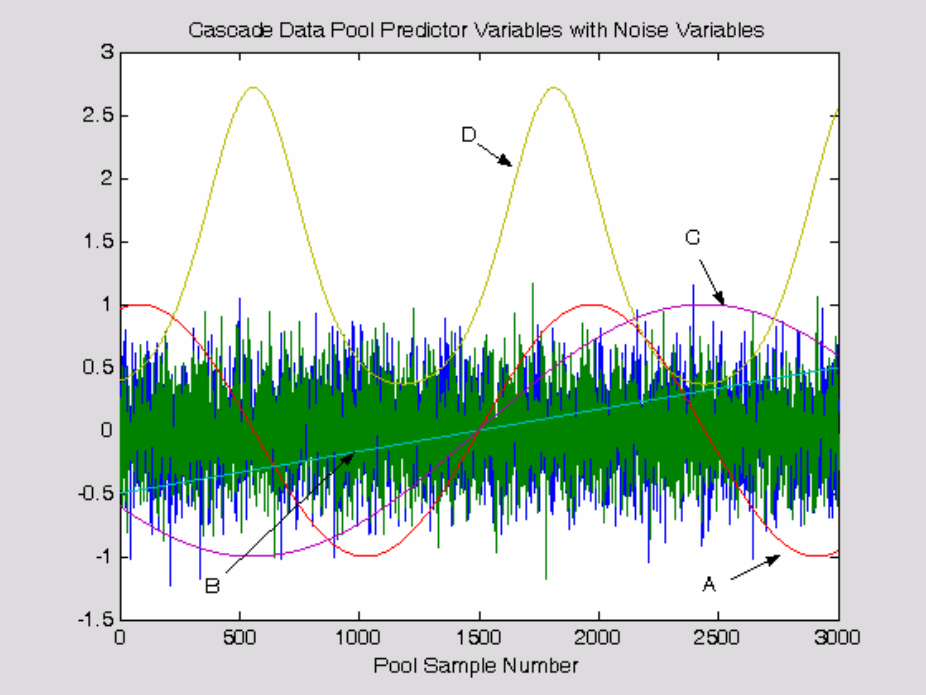


Figure 5.3.20: Cascade Data Pool predictor variables with 2 additional noise variables.

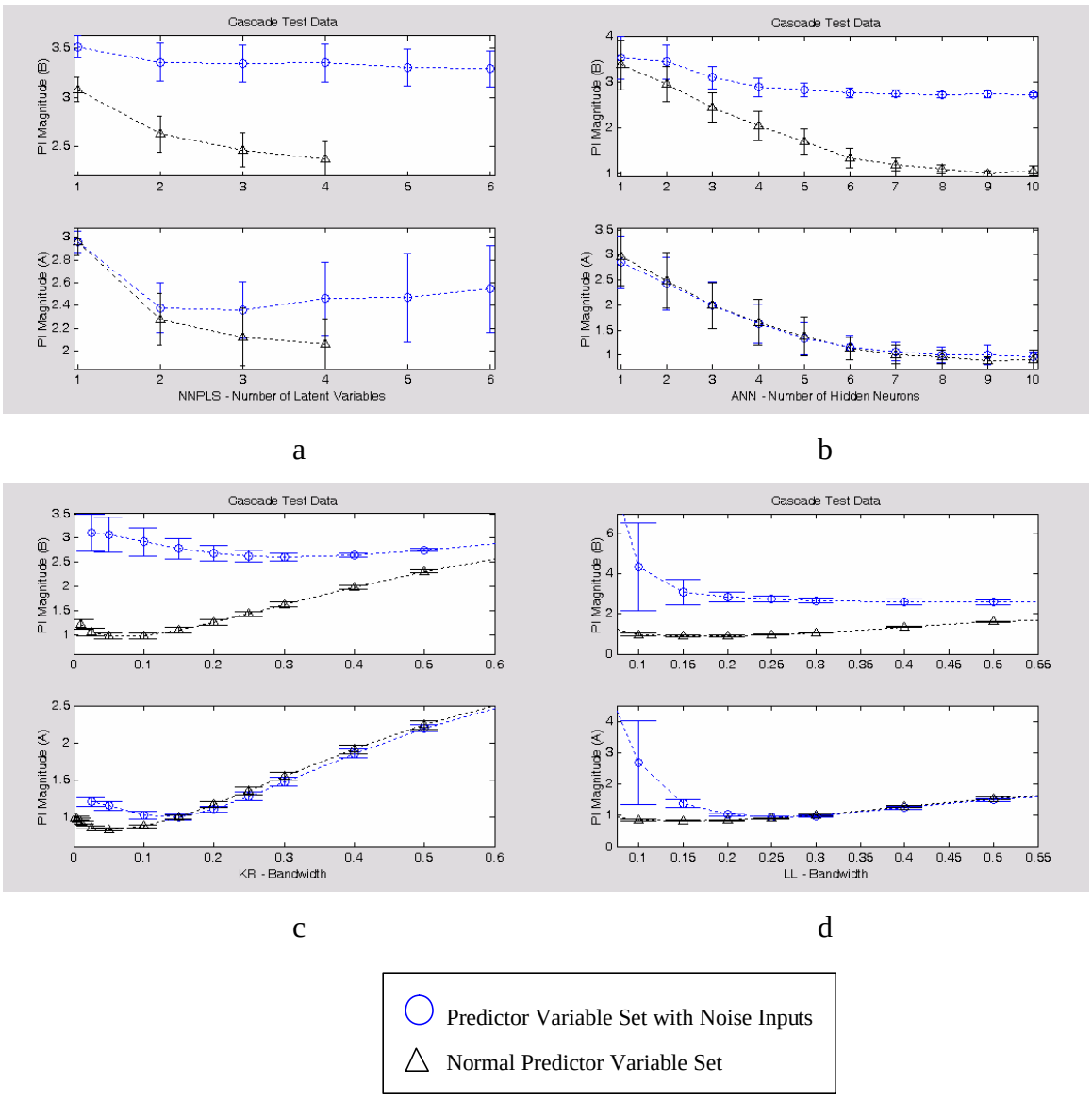


Figure 5.3.21: Comparison of PI magnitude results for Cascade Test Data Pool for normal set of predictors and the augmented set that included 2 normal random variables. The models used are specified along the x-axis. The bands represent the 1*s* variation in the results over the 100 iterations. The upper subplot of each plot contains the results based on the analytic prediction interval methods, indicated by an (A) on the y-axis. The lower subplot of each plot contains the results based on the bootstrap prediction intervals utilizing the squared error bias estimate (For more information, see section 4.10), and is indicated by a (B) on the y-axis. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

Considering the NNPLS coverage values for the cascade2 test data (figure 5.3.22a) it is seen that all results for both prediction interval methods provide adequate coverage for all architectures with at least 2 latent variables. The larger bootstrap PIM for the NNPLS models result in higher than expected coverage values for both data sets. As the bootstrap method intends to capture all possible sources of variation this is not unexpected, and is characteristic of the bootstrap intervals computed throughout this work. The coverage values for the ANN models for the cascade1 and cascade2 data sets are provided in figure 5.3.22.b. The average results are at the expected value of 0.95 for both data sets, though it is noted that the variability increases slightly for the cascade2 data. Overall the ANN model's estimations and analytic prediction intervals did not appear to be overly influenced by the erroneous inputs, though the bootstrap intervals exhibited moderate increases, especially for increasingly complex ANN models. The KR and LL model coverage values for the cascade1 and cascade2 data sets are shown in figures 5.3.22.c and 5.3.22.d. In both cases the coverage values for the cascade2 test data for the bootstrap prediction intervals exceeded the expected value of 0.95. Considering the significant increases in the PIMs for the bootstrap prediction intervals this was an expected result. The LL analytic prediction interval coverage for cascade2 at the previously determined optimal bandwidth of h=0.15 is adequate, though for increasing bandwidths, the fractional coverage falls short. For bandwidth values below the optimal, the coverage of the LL analytical prediction intervals is adequate due to the influence of the bias estimate. As previously discussed, for small bandwidths, LL models will exhibit large prediction intervals due to the inadequacy of the training data density for the use of such small bandwidth values. As the bandwidth increases beyond the h=0.15 optimum for the LL case, a drop in coverage is observed as the estimations stabilize in a region of moderate bias. Here the bias is underestimated. With increasing bandwidth, the bias estimate also increases bringing the coverage closer to the expected value. A similar discussion can be presented for the KR coverage results of the cascade2 data, though the effect of over-estimating bias for small bandwidths as occurs for LL models, was not exhibited for the KR models.

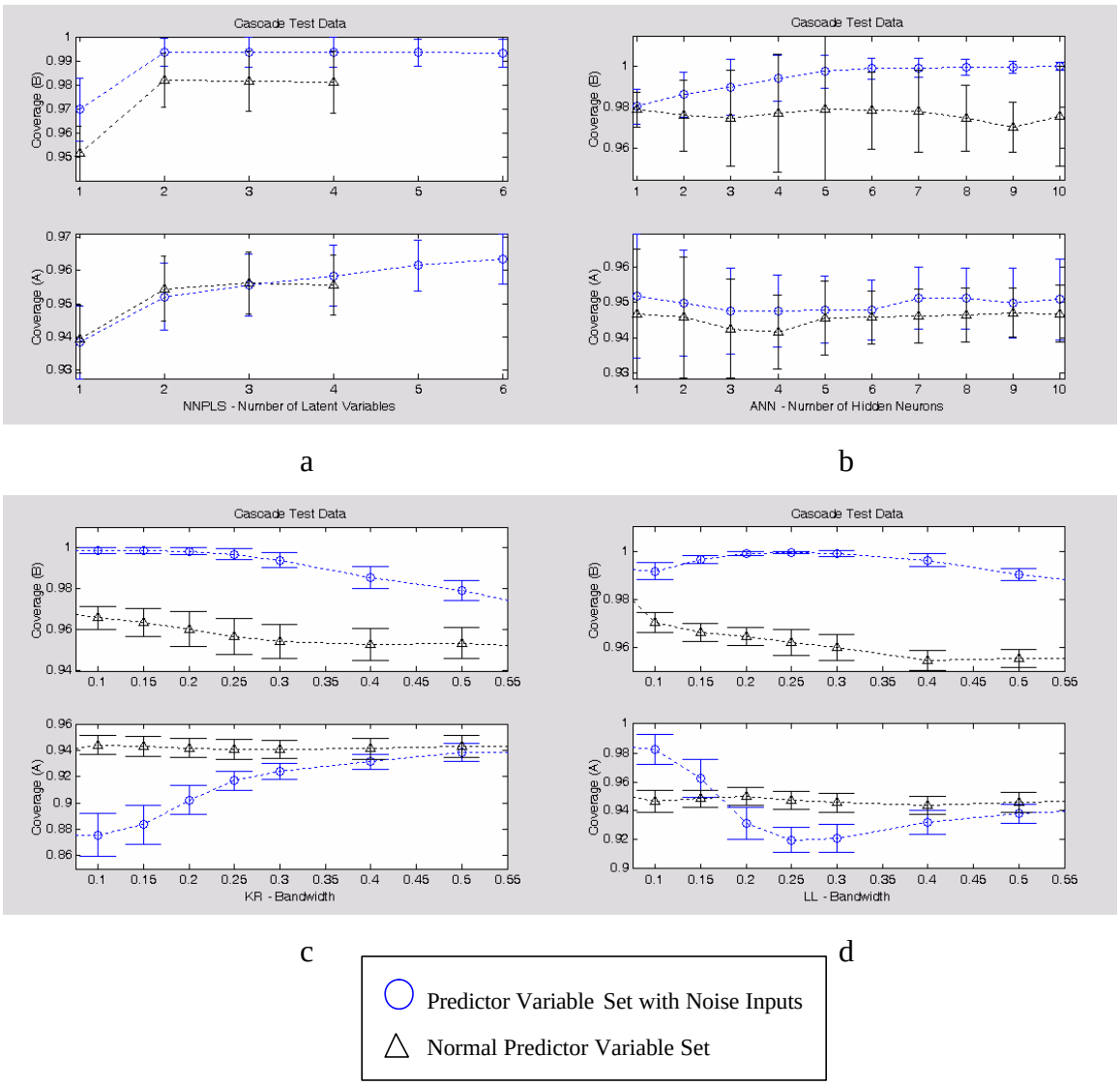


Figure 5.3.22: Comparison of coverage Results for Cascade Test Data Pool for normal set of predictors and the augmented set that included 2 normal random variables. The models used are specified along the x-axis. The bands represent the 1*s* variation in the results over the 100 iterations. The upper subplot of each plot contains the coverage results based on the analytic prediction interval methods, indicated by an (A) on the y-axis. The lower subplot of each plot contains the coverage results based on the bootstrap prediction intervals utilizing the squared error bias estimate (For more information, see section 4.10), and is indicated by a (B) on the y-axis. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

Overall, the study of the effects of erroneous predictors on the prediction interval estimates and their corresponding coverage values revealed that for NNPLS and ANN models, the analytic intervals consistently provide the expected coverage, on average, though their variation over repeated iterations did increase slightly for the ANN models. With respect to the LPR models it was observed that the analytic prediction interval methods did not maintain the expected level of coverage consistently over the entire bandwidth range when erroneous predictors were included.

5.3.1 Cascade Data Set Summary of Results

Regarding the cascade data set, optimal performance, with respect to MAE and MSE, was obtained with the LL models of bandwidth 0.15. The KR models of bandwidth 0.05 had very similar error values exhibiting only a marginal increase, followed by another slight increase for the 9 hidden neuron ANN models. The NNPLS model errors were significantly larger for all evaluated architectures; thus its use for this data set was not appropriate. The reason for this is due to the strong non-linear relationships in the data. Previous work using the NNPLS model architecture states that sufficient model performance requires a high level of linearity in the predictor variables [Rasmussen 2002]. This fact is evidenced for the cascade data set. The performance of all empirical models for the cascade data set was extremely consistent for the data pool as well as the test data. While this is often the case when evaluating simulated data, in practical applications the test errors are generally larger for the test data.

The average magnitudes of the analytic prediction intervals were minimized for the LL models (h=0.15), followed closely by KR (h=0.05), and ANNs (9 hidden neurons). The NNPLS average analytic intervals were greater than twice the average for the ANN models. This is a positive result indicating that the computed intervals were properly influenced by the inaccuracy of the NNPLS model, for the given data set. As was described for the error values, the results for both the data pool and the test data were extremely similar. The average magnitudes of the corresponding bootstrap prediction intervals were consistently larger, to a relatively small extent, than the analytic intervals.

This result is due to the use of various assumptions in deriving the computed prediction intervals, whereas the bootstrap approach to prediction interval estimation has no associated assumptions. In the regions where the empirical models performed at their respective optimal levels it is noted that the differences between the bootstrap and analytic intervals is minimized. Thus, it can be said that the analytic intervals do reflect a reduction in performance, with respect to the bootstrap intervals, as the model architecture changes from its optimal structure. While this effect was exhibited in the results, it was also noted that the performance of the models, as they strayed from their optimal architecture, with respect to their coverage values was not significantly degraded. This will be further discussed below.

Regarding the analytic coverage values for the cascade data set, all LL models resulted in average coverage values greater than 95% for the data pool, while 7 out of 15 of the different models for the test data had coverage values that were slightly below, though the lowest value was 94%. The bootstrap coverage values were all above the expected value for both the data pool and the test data. In general, the bootstrap coverage values for the cascade data were all above the 95% level with the exception of 3 over-regularized KR models, though the lowest value of the 3 was 93%. The analytic coverage values for the KR models were all $\geq$93%, for the NNPLS models $\geq$94%, and for the ANN models $\geq$94%. Note that rounding to 2 digits is being performed, while the tabulated results in section 5 provide 3 digits. For the optimal architectures of the different models the lowest average analytic coverage value observed was for the KR models, at 94%. It was also noted that the KR coverage values were consistently just under the expected value of 95%. As discussed in section 5, the KR models have upper and lower limits on their range of estimations. None of the other empirical models have this limit. Thus, the bias estimates for the KR models are also limited. It is interesting to note that the bootstrap intervals for the KR models exhibit the lowest average values compared to the other empirical models. These minor deviations do not pose extremely significant issues in applying the derived methodologies. The results indicate that the computed analytic intervals indicate the optimal model structure, which in all cases for

the cascade data occurred where the average error values were minimized, and for the optimal model structure the lowest average coverage was 94%. Assuming that one would perform some type of model optimization, the resultant prediction intervals for models near the optimum resulted in sufficiently accurate prediction intervals.

The study of the effects of erroneous predictor variables on the computed prediction intervals revealed that the intervals suitably increase under these conditions. While the more significant increases were incurred for the bootstrap intervals, the analytic intervals also increased. For the KR models, the increases in the prediction intervals were not significant enough to maintain the expected level of coverage at the optimally selected bandwidth, though for larger bandwidth values the results improved. For the local linear models a slight reduction in the coverage values was also noted. For both the ANN and NNPLS models, the prediction interval magnitudes increased appropriately, and the coverage of the intervals was not degraded at all by the erroneous predictors. The slight reduction in coverage of the intervals for the cases of KR and LL models is not extremely severe, and the coverage values were maintained at or above 87% for the KR models and 92% for the LL models.

5.4 FEEDWATER FLOW RATE DATA SET

In this section, a similar analysis was completed as was performed for the cascade data set. An initial description of the data and model parameters is given followed by tabulated and plotted results. For this data set, the discussion of the results is done on a model type basis, each model being discussed in a separate section. A summary at the end (section 5.4.5) draws comparisons between the results from the different model types. An additional evaluation was performed for the feedwater flow data which investigated the effect of the conditional variance estimate of the LPR models on the resultant prediction interval computations. The study intended to determine that the coverage values of the computed prediction intervals were not overly influenced by the conditional variance estimate, such that drift detection could be performed based on the computed prediction intervals as long as the conditional variance estimate was

reasonable. Quantification of the drift estimates of the different models was also completed, and the average results along with the variation of these results are reported.

The feedwater flow rate data set was provided by a nuclear power plant, was recorded with a 30 minute sampling rate, and has dimensions 17-input / 1-output. The number of observations in the data pool was a 1440, all of which were selected from the first month of data available (assumed to be fault free). The response data for the data pool are shown in figure 5.4.1.

The feedwater flow venturi meter is susceptible to a fouling phenomenon that has been the subject of considerable work at the University of Tennessee [Gribok 2001]. It is expected that the measurements from the venturi meter will drift from the *true* value of the flow rate, and thus it is expected that the estimations for the feedwater flow rate will deviate from the measured values as the fouling occurs. The test data for this case covers a period of 900 samples spaced 300 minutes apart. The reason for the large sampling increment was to reduce computational requirements, while still obtaining estimations over a long period of operation, and it is assumed that the results obtained would be representative for a faster sampling rate, not to exceed that available in the training data. The 900 test samples were further reduced to eliminate observations that would lead to significant contributions to the uncertainty or errors of the resultant models. The final set of test data consisted of 847 observations.

The data pool $(\mathbf{X}_{pool}, \mathbf{Y}_{pool})$ of dimensions $(n_{pool} \times p)$ and $(n_{pool} \times 1)$ respectively were specified by: $n_{pool} = 1440$, $p = 17$. The observations in the data pool cover the first 30 days following the onset of steady-state operations after a start-up. The 100 training data sets: $(\mathbf{X}^{b}{}_{tr}, \mathbf{Y}^{b}{}_{tr})$, of dimensions $(n_{tr} \times p)$, and $(n_{tr} \times 1)$ respectively were specified by: $n_{tr} = 900$, $p = 17$. The training observations were drawn at random, with replacement,

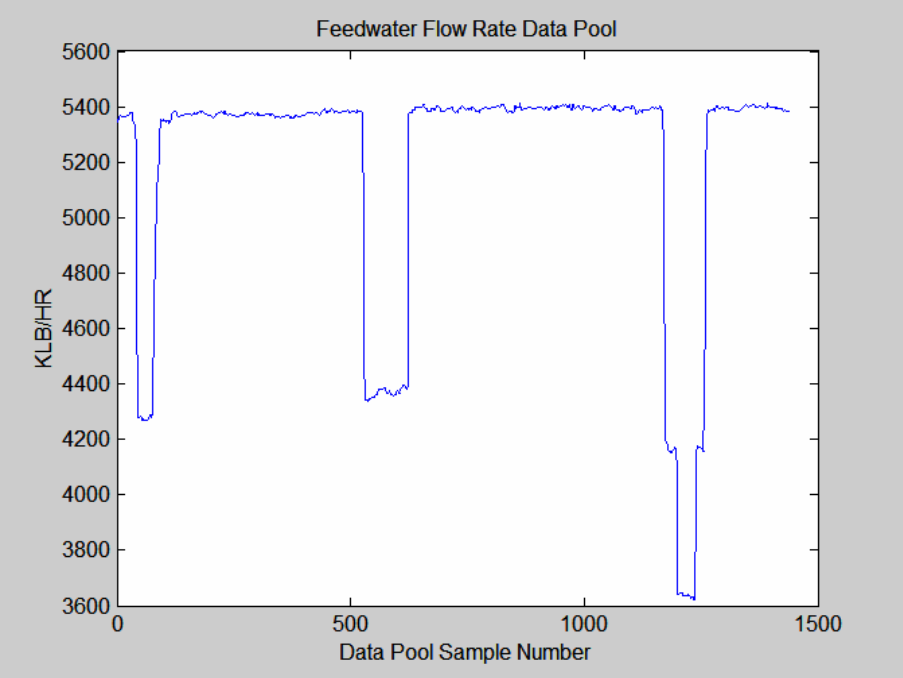


Figure 5.4.1: Feedwater flow rate data pool.

from the data pool. In all cases, $b = 1,...,B$ and $B = 100$. The 100 validation data sets $(\mathbf{X}^b{}_v, \mathbf{Y}^b{}_v)$ of dimensions $(n_v \times p)$ and $(n_v \times 1)$ respectively were specified by: $n_v = 600$, $p = 17$. The validation observations were drawn at random, with replacement, from the data pool. The test data set $(\mathbf{X}_{ts}, \mathbf{Y}_{ts})$ of dimensions $(n_{ts} \times p)$ and $(n_{ts} \times 1)$ respectively was specified by: $n_{ts} = 847$, $p = 17$. The test data cover approximately 6 months of plant operation (300 minutes between samples).

The set of predictor variables for the feedwater flow rate data set is provided in table 5.4.1. The required model specifications used for the feedwater flow rate data are provided below:

NNPLS:
Mean-standard initialization was applied. The maximum number of latent variables to evaluate was set at 4.
ANN:
Mean-standard initialization was applied. The minimum and maximum numbers of hidden neurons evaluated were 1 and 8, respectively.
LPR:
The vector of bandwidths to be evaluated was:
[0.005 0.025 0.05 0.06 0.07 0.08 0.09 0.1 0.125 0.15 0.2 0.25 0.5 0.75];
The conditional variance was estimated as $s_y^2 = 4.47$.

The overall results for the feedwater flow rate data pool are presented in tables 5.4.2, and for the test data in table 5.4.3. All of the following discussions for the feedwater flow rate data set results refer to the values in these two tables. Note that the bootstrap prediction intervals being reported in these tables were based on the average squared error bias estimate ($Bias_2$). No results are reported for the bootstrap prediction intervals

Table 5.4.1: Feedwater flow rate data set predictor variable descriptions

Variable #	Description	Range	Units
1	Feedwater Pump Speed	0-7500	RPM
2	A OTSG Efic. High Level	0-100	%
3	Feedwater Pump A Speed	0-7500	RPM
4	Linear Power Channel NI-6	0-125	%
5	Heater 3A Inlet Cond. Temp	40-300	DEGF
6	Heater 3B Outlet Cond. Temp	40-300	DEGF
7	Steam Gen. A Level (OP)	0-100	%
8	Steam Gen. A Level (Full)	40-640	Inches
9	Steam Gen. A Level (Start-up)	0-250	Inches
10	Steam Gen. B Inlet FW Temp	0-500	DEGF
11	Steam Gen. B Level (Start-up)	0-250	Inches
12	Steam Gen. A Inlet FW Temp.	40-600	DEGF
13	Steam Gen. B Inlet FW Temp.	40-600	DEGF
14	Reheater A Cold Reheat Pressure	0-200	PSIG
15	Reheater D Cold Reheat Pressure	0-200	PSIG
16	Reheater C Cold Reheat Pressure	0-200	PSIG
17	No. 2A Extr. LP Turbine Pressure	0-20	PSIA

Table 5.4.2: Tabulated results for feedwater flow rate data pool*.
(For full explanation of terms see section 5.2)

Model	Type	API		BPI		AC		BC		MAE	MSE
		m	s	m	s	m	s	m	s		
NNPLS	1	14.882	1.032	16.851	5.379	0.968	0.004	0.981	0.003	5.401	64.670
NNPLS	2	14.439	1.598	16.093	5.456	0.969	0.005	0.981	0.003	4.905	57.273
NNPLS	3	14.389	3.897	15.637	5.866	0.969	0.006	0.985	0.004	4.648	51.737
NNPLS	4	14.574	7.195	14.926	5.953	0.970	0.006	0.987	0.005	4.410	46.348
ANN	1	33.590	80.953	104.257	57.039	0.959	0.005	0.997	0.015	8.182	1830.357
ANN	2	30.055	62.139	78.994	44.821	0.963	0.008	0.991	0.028	7.115	1069.085
ANN	3	39.735	83.680	82.889	47.145	0.967	0.011	0.992	0.026	7.484	1173.977
ANN	4	46.502	77.855	86.260	48.449	0.970	0.012	0.992	0.024	8.333	1269.562
ANN	5	44.451	66.198	71.051	42.375	0.976	0.012	0.993	0.018	7.240	877.431
ANN	6	48.974	78.335	64.152	36.988	0.979	0.012	0.994	0.020	6.961	703.174
ANN	7	64.445	91.830	67.694	39.829	0.981	0.013	0.993	0.019	6.944	791.471
ANN	8	76.425	119.33	70.026	39.383	0.983	0.012	0.992	0.021	7.662	830.449
KR	0.005	4.542	0.128	6.833	1.743	0.878	0.007	0.971	0.005	1.200	6.007
KR	0.025	4.622	0.117	6.945	5.528	0.864	0.008	0.961	0.006	2.078	7.980
KR	0.050	8.093	0.141	10.247	10.974	0.924	0.006	0.954	0.005	3.474	18.311
KR	0.060	9.006	0.160	11.054	10.862	0.927	0.006	0.954	0.005	3.856	22.209
KR	0.070	10.089	0.179	12.061	10.717	0.935	0.006	0.956	0.005	4.308	27.495
KR	0.080	11.352	0.191	13.427	12.362	0.938	0.006	0.958	0.005	4.845	34.440
KR	0.090	12.686	0.196	14.707	12.184	0.944	0.007	0.964	0.005	5.422	42.687
KR	0.100	13.971	0.202	15.935	12.016	0.949	0.007	0.968	0.005	5.975	51.440
KR	0.125	16.679	0.224	18.854	12.942	0.952	0.005	0.967	0.003	7.150	72.779
KR	0.150	18.669	0.262	20.755	12.648	0.955	0.006	0.968	0.004	8.002	90.686
KR	0.200	21.116	0.338	23.088	12.221	0.958	0.004	0.967	0.003	9.053	115.465
KR	0.250	22.551	0.394	24.370	11.769	0.958	0.004	0.966	0.003	9.652	130.746
KR	0.500	26.665	0.593	28.198	10.162	0.945	0.005	0.955	0.006	11.240	181.491
KR	0.750	30.609	1.223	31.929	7.847	0.928	0.009	0.939	0.011	12.597	237.867
LL	0.005	6.335	0.340	13.822	6.550	0.813	0.012	0.977	0.006	2.815	25.558
LL	0.025	5.653	0.681	21.270	89.935	0.895	0.009	0.986	0.004	1.769	12.915
LL	0.050	6.187	1.263	35.605	178.90	0.902	0.009	0.985	0.005	2.288	12.956
LL	0.060	6.494	0.907	34.358	177.39	0.911	0.008	0.983	0.005	2.403	12.229
LL	0.070	6.684	0.839	32.659	171.66	0.916	0.009	0.981	0.005	2.484	12.171
LL	0.080	6.772	0.809	31.050	167.75	0.919	0.008	0.979	0.005	2.531	11.857
LL	0.090	6.932	0.818	33.441	194.50	0.922	0.007	0.977	0.005	2.575	11.857
LL	0.100	7.065	0.862	31.582	186.31	0.924	0.008	0.976	0.005	2.611	11.994
LL	0.125	7.218	0.932	33.656	202.18	0.927	0.007	0.973	0.005	2.669	12.251
LL	0.150	7.214	0.868	32.557	197.12	0.929	0.007	0.970	0.005	2.691	12.045
LL	0.200	7.223	0.797	26.507	171.69	0.931	0.007	0.965	0.004	2.706	11.722
LL	0.250	7.232	0.756	22.413	151.32	0.931	0.007	0.962	0.004	2.709	11.497
LL	0.500	6.944	0.566	10.505	47.963	0.935	0.007	0.957	0.004	2.725	11.242
LL	0.750	6.919	0.611	7.994	17.555	0.937	0.006	0.956	0.004	2.749	11.315

* API – Analytic prediction intervals, BPI – Bootstrap prediction intervals, m - mean result, s - standard deviation of results, MAE – Mean Absolute Error, MSE – Mean Squared Error, Type for NNPLS refers to the number of latent variables, for ANNs refers to the number of hidden neurons, and for KR and LL refers to the bandwidth parameter. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

Table 5.4.3: Tabulated results for feedwater flow rate test data*.
(For full explanation of terms see section 5.2)

Model	Type	API		BPI		AC		BC		MAE	MSE
		m	s	m	s	m	s	m	s		
NNPLS	1	16.038	1.177	17.208	3.194	0.258	0.013	0.283	0.013	27.281	973.261
NNPLS	2	18.331	2.980	16.684	3.273	0.279	0.039	0.283	0.017	26.152	903.384
NNPLS	3	24.697	12.835	17.963	3.739	0.349	0.138	0.279	0.023	27.457	1017.975
NNPLS	4	29.283	14.525	17.424	3.814	0.424	0.156	0.286	0.028	27.974	1069.974
ANN	1	20.614	20.941	98.338	35.798	0.452	0.240	0.996	0.016	21.277	1417.183
ANN	2	31.728	61.605	133.64	90.015	0.447	0.225	0.987	0.045	22.772	5945.110
ANN	3	41.806	81.998	236.59	142.708	0.476	0.228	0.976	0.090	38.605	19658.063
ANN	4	47.217	66.301	256.74	157.283	0.533	0.251	0.967	0.112	47.530	24031.558
ANN	5	51.994	72.202	257.48	152.518	0.576	0.247	0.962	0.114	51.427	24400.742
ANN	6	58.143	73.612	187.57	107.359	0.646	0.245	0.960	0.104	44.912	12710.981
ANN	7	80.863	105.06	205.16	111.031	0.662	0.254	0.955	0.120	50.466	14181.608
ANN	8	102.02	123.49	229.60	126.444	0.742	0.240	0.963	0.108	54.625	18939.579
KR	0.005	4.529	0.147	7.907	3.632	0.743	0.008	0.814	0.006	8.236	129.741
KR	0.025	5.023	0.154	11.093	11.210	0.219	0.011	0.334	0.026	21.364	730.367
KR	0.050	8.188	0.141	14.061	18.323	0.248	0.011	0.309	0.016	20.005	600.388
KR	0.060	9.056	0.159	14.456	18.074	0.259	0.012	0.316	0.015	18.743	512.123
KR	0.070	10.116	0.178	15.135	17.776	0.290	0.016	0.347	0.017	17.257	421.476
KR	0.080	11.368	0.191	16.064	17.425	0.354	0.027	0.416	0.027	15.754	341.702
KR	0.090	12.697	0.196	17.134	17.042	0.459	0.039	0.528	0.037	14.372	278.868
KR	0.100	13.980	0.201	18.210	16.648	0.593	0.042	0.657	0.039	13.171	232.787
KR	0.125	16.686	0.224	20.581	15.677	0.853	0.023	0.899	0.020	10.991	169.970
KR	0.150	18.674	0.262	22.333	14.863	0.919	0.010	0.954	0.007	9.811	147.614
KR	0.200	21.120	0.338	24.464	13.718	0.938	0.007	0.967	0.006	9.041	143.629
KR	0.250	22.553	0.394	25.598	12.837	0.942	0.007	0.968	0.006	9.060	155.571
KR	0.500	26.664	0.593	28.684	8.989	0.937	0.003	0.951	0.004	10.600	312.042
KR	0.750	30.608	1.223	32.147	7.657	0.935	0.005	0.945	0.004	11.676	456.164
LL	0.005	5.765	0.398	16.704	6.811	0.698	0.006	0.761	0.003	65.008	8680.320
LL	0.025	57.284	13.627	107.10	132.064	0.679	0.063	0.801	0.052	98.556	39298.701
LL	0.050	20.913	2.295	74.847	218.438	0.411	0.095	0.561	0.098	40.530	8644.626
LL	0.060	15.643	1.488	64.660	210.334	0.329	0.066	0.450	0.078	36.891	6499.984
LL	0.070	13.132	1.225	59.736	208.593	0.302	0.044	0.399	0.056	33.742	4986.995
LL	0.080	11.793	1.175	54.598	202.675	0.296	0.029	0.376	0.041	31.211	4062.140
LL	0.090	10.899	1.163	50.776	196.243	0.296	0.022	0.369	0.032	28.764	3202.631
LL	0.100	10.268	1.125	46.576	185.336	0.297	0.019	0.365	0.027	26.789	2634.232
LL	0.125	9.419	1.108	43.270	191.025	0.300	0.018	0.363	0.023	22.938	1608.474
LL	0.150	9.050	1.072	37.239	168.109	0.303	0.018	0.360	0.022	20.398	1027.417
LL	0.200	8.466	0.731	27.104	129.771	0.306	0.020	0.355	0.022	17.843	556.750
LL	0.250	7.965	0.396	19.979	104.182	0.303	0.021	0.345	0.023	16.939	433.685
LL	0.500	7.407	0.275	9.923	11.651	0.291	0.020	0.327	0.023	16.136	374.035
LL	0.750	7.196	0.164	8.870	5.259	0.288	0.018	0.316	0.020	15.900	361.063

* API – Analytic prediction intervals, BPI – Bootstrap prediction intervals, m - mean result, s - standard deviation of results, MAE – Mean Absolute Error, MSE – Mean Squared Error, Type for NNPLS refers to the number of latent variables, for ANNs refers to the number of hidden neurons, and for KR and LL refers to the bandwidth parameter. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

based on the $Bias_1$ computation, due to the insufficiency of the method to adequately cover the response measurements, evidenced with respect to the cascade data set (section 5.3). A full analysis was carried out for the feedwater flow rate data for each model type in the following 4 sections.

5.4.1 Neural Network Partial Least Squares Models and Analyses

The PI magnitude results for the data pool are plotted in figure 5.4.2.a and for the test data in figure 5.4.2.b. The variation in the PIMs for the NNPLS models is consistent for the bootstrap intervals over all architectures for both the data pool and the test data. The only trend in the bootstrap PIMs variation is a slight increase with model dimension. This is expected since the number of free parameters increases with model dimension. In addition, the mean values of the bootstrap PIMs exhibit only a minor decrease with model dimension. It is noted that for the case of the test data, the intervals are larger, though the difference is slight. Viewing the corresponding analytic PIMs shows that for the data pool the mean values and variations are lower than for the bootstrap PIMs. It is noted that the increase in variation of the analytic PIMs with model dimension is much more obvious than in the case of the bootstrap PIMs. The test data analytic PIMs show that for 1 latent variable, the mean PIM value is lower than the bootstrap PIM value, while for higher dimensions the opposite is true. The coverage values for the data pool (figure 5.4.2.c) indicate that the expected level of 0.95 is achieved for both the analytic and bootstrap prediction intervals. The slightly higher value for the bootstrap results is due to the larger corresponding prediction intervals. The coverage values for the test data (figure 5.4.2.d) indicate that both interval estimation methods provide less than 30% coverage for the test measurements (considering 1 and 2 latent variable models). This is also expected since it is known that the test response signal contains a drift.

Based on the results for the feedwater data pool and the test data, the optimal number of latent variables was determined to be 2. While there are some improvements in the residual errors (figure 5.4.3) and PI magnitudes for the data pool if more latent variables are used, the increased variance that comes along with the additional latent variables is

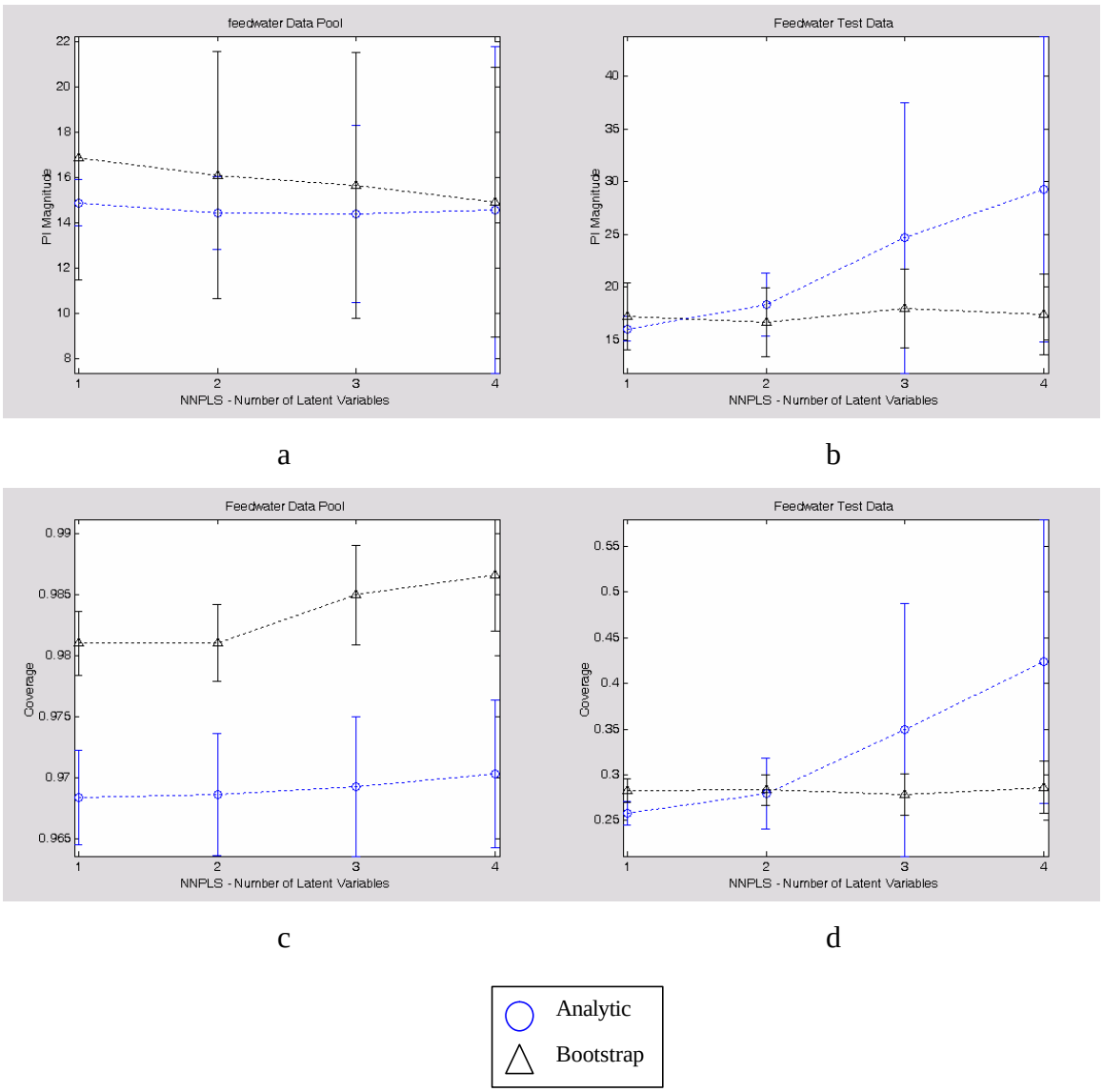
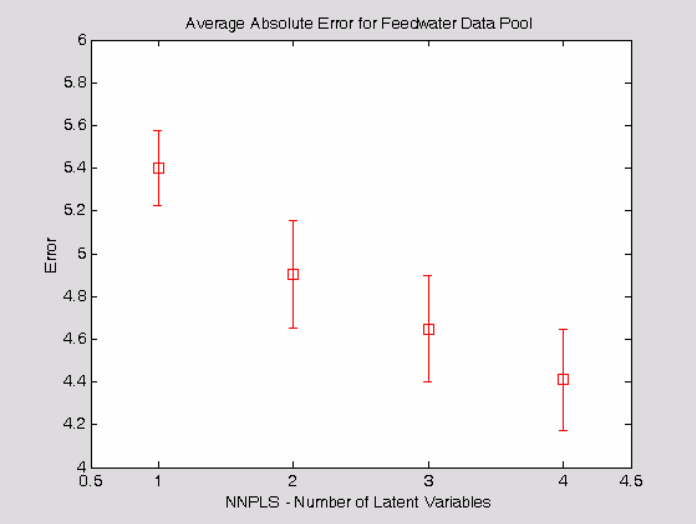
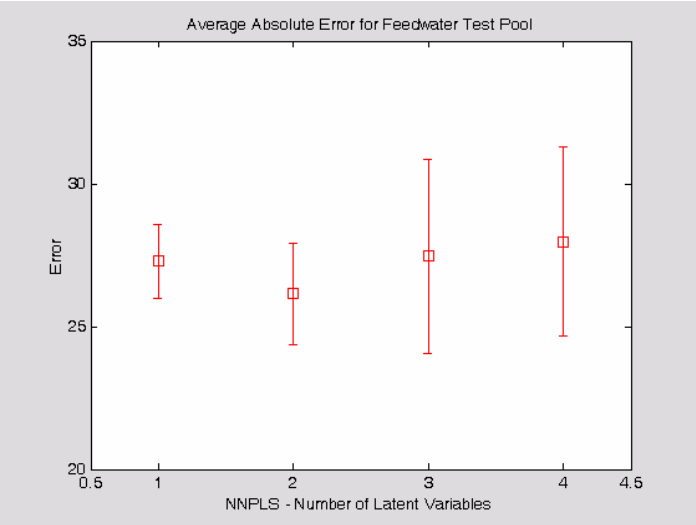


Figure 5.4.2: PI magnitude and coverage results for NNPLS models of feedwater data pool. The bands represent the 1*s* variation in the results over the 100 iterations.



a



b

Figure 5.4.3: Mean absolute error results for NNPLS models of feedwater flow rate data pool (a) and test data (b).

The bands represent the 1*s* variation in the results over the 100 iterations.

detrimental. This effect can be seen in the results for the test data leading to higher PIMs. The PIM reduction seen for the data pool as more latent variables are added is due to the decreasing bias; however, the bias with respect to the test data will not decrease at the same rate and the variance increase will eventually be the dominant contributor in the PIM values. Viewing the test data coverage values and PIM values, it is apparent that a single latent variable model is the optimal choice; however, in developing signal validation models the test data is not readily available; thus, model architecture decisions should be based on the data pool and not the test data. The average MAE values for the feedwater test data are provided (figure 5.4.3.b) to show that the variability in the error for the test data remains reasonable upon repeated trials. The increased magnitude is expected since the test data contain a known drift. The decrease in the MAE values for the data pool (figure 5.4.3.a) is due to the better fit to the training data as the model dimension is increased. Figure 5.4.4 presents the estimation results for 4 randomly selected 2 latent variable NNPLS models from the available 100. The drift is identifiable for all cases presented. The random results are shown to provide a representative illustration of the expected results from a 2 latent variable NNPLS model for the feedwater flow rate data.

To provide a more clear illustration of the NNPLS estimates and analytic prediction intervals, the median result (based on the model exhibiting the median MAE value with respect to the data pool) is plotted in figure 5.4.5. The drifting measurement is clearly identifiable and is not bounded by the prediction intervals. Therefore within the context of a signal validation model, this drift can be reported with a level of confidence of 95%.

Of course, each model will result in a slightly different value for the drift estimate. Figure 5.4.6 shows the distribution of the drift estimate results for all 100 models in its upper subplot, and only for models which provided 5% or less coverage of the test measurements over the range from sample 700 to 847. The results of the two subplots for the case of the NNPLS models are not significantly different; however, for other model

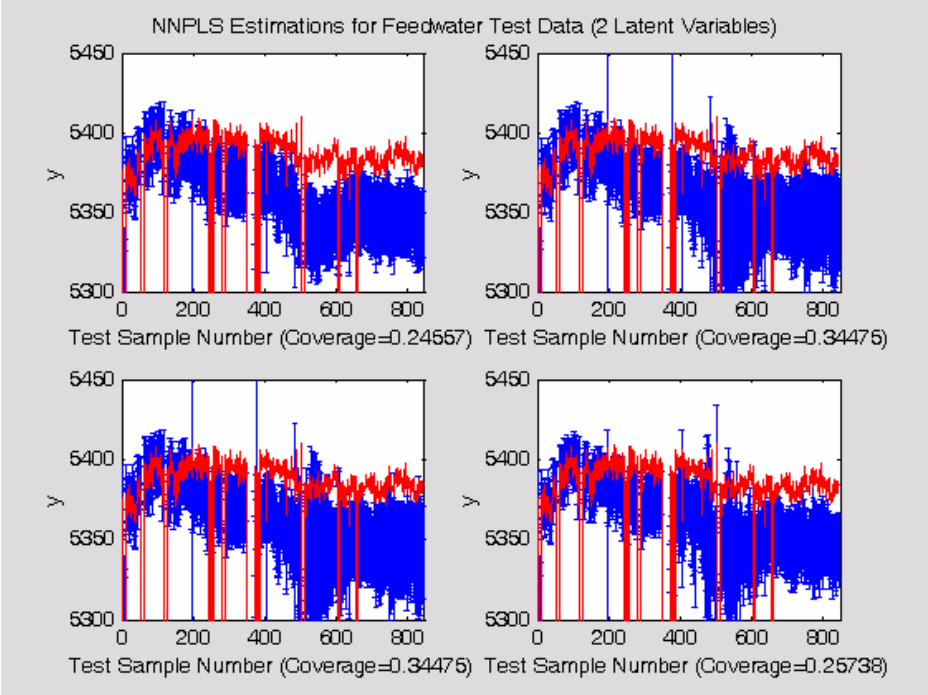


Figure 5.4.4: Test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 available 2 latent variable NNPLS models. The coverage value for each case is indicated on the x-axis.

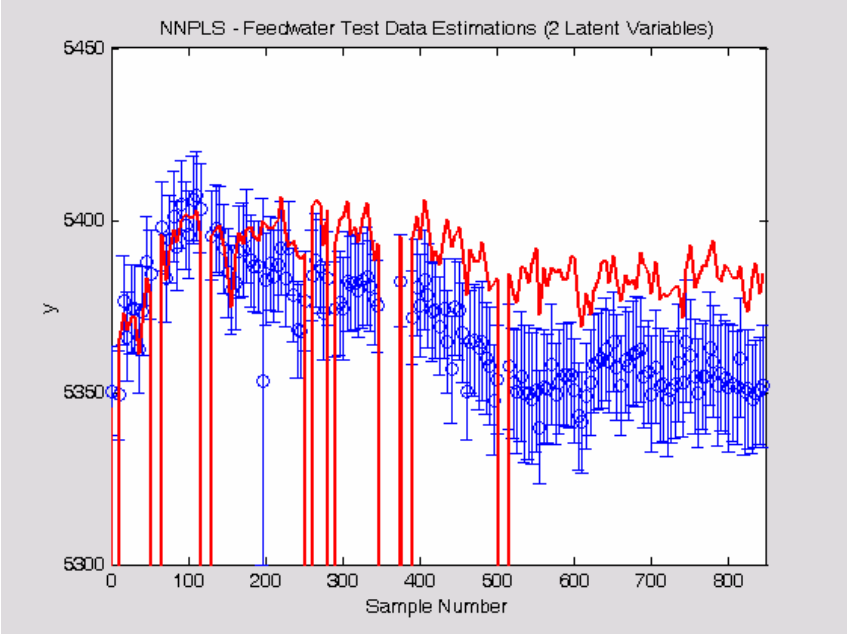


Figure 5.4.5: NNPLS median result estimation and analytic prediction intervals. Representative result for NNPLS 2 latent variable model showing every 5th estimate and the corresponding analytic prediction intervals. This model exhibited the median value of the 100 Mean Absolute Error results for the 2 latent variable models.

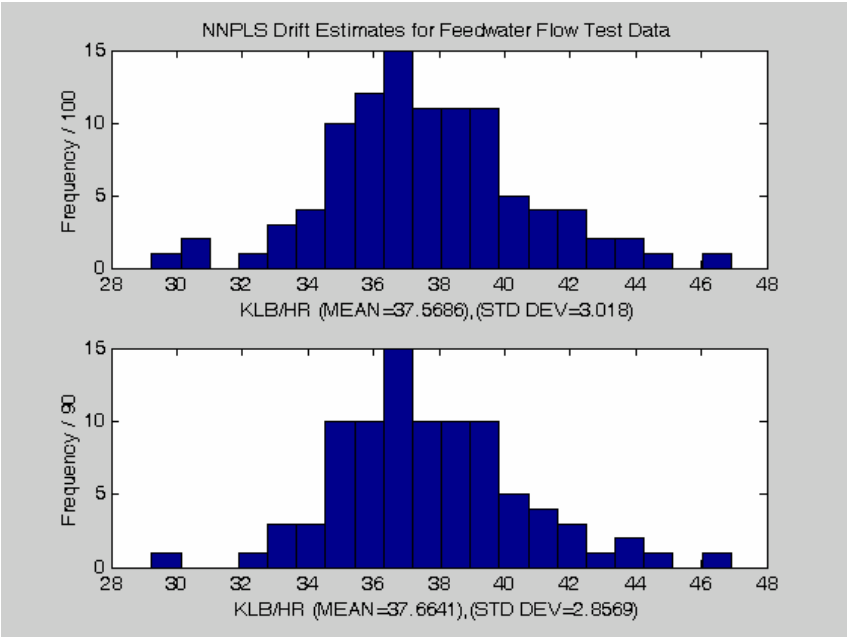


Figure 5.4.6: Distribution of drift estimate for 2 latent variable NNPLS models. Drift estimates obtained from the 100 NNPLS models via computing the MAE over test samples 700 to 847. The upper subplot shows all results, while the lower subplot excludes any results for which coverage was greater than 5%.

types there will be significant differences and this plot will be used as a point of comparison.

The NNPLS models and corresponding prediction intervals performed well for the case of the feedwater flow data set. While the bootstrap prediction intervals indicate slightly higher PIM values than the analytic PIMs, both approaches provide adequate coverage of the measurements in the data pool. In addition, the PIM for the test data did not significantly increase over those obtained for the data pool, indicating that the data pool provided an adequate representation of the system being modeled. Through the inherent regularization of the NNPLS design, reducing the 17 predictor variables to 2 latent variables, a stable model solution can be obtained and its repeatability has been proven in this case, within a small margin of variation. In addition, the NNPLS models successfully report the feedwater flow rate channel drift in all cases.

5.4.2 Artificial Neural Network Models and Analyses

The ANN models proved to be extremely variable for the feedwater flow data set. Figure 5.4.7a and 5.4.7.c indicate that the PI magnitudes averaged over the 100 iterations are excessively high for both the data pool and the test data. This is due to the collinearity of the feedwater data set, resulting in highly unstable ANN parameter solutions. A reduced set of models was obtained (see below), and the results computed again. The results from the reduced set of ANN models are provided in figures 5.4.7.b and 5.4.7.d. To aid in understanding the source of variation here, consider the empirical density of the analytic PIMs from the 100 observations for the case of one hidden neuron (figure 5.4.8).

For the 1 hidden neuron case, 72 of the 100 iterations provided average PI magnitudes less than 20 KLB/HR, while the range for the remaining 28 results was from >20 KLB/HR to ~600 KLB/HR. The overall average for this case is 33.6 KLB/HR, whereas the median of the 100 trials is 9.2 KLB/HR.

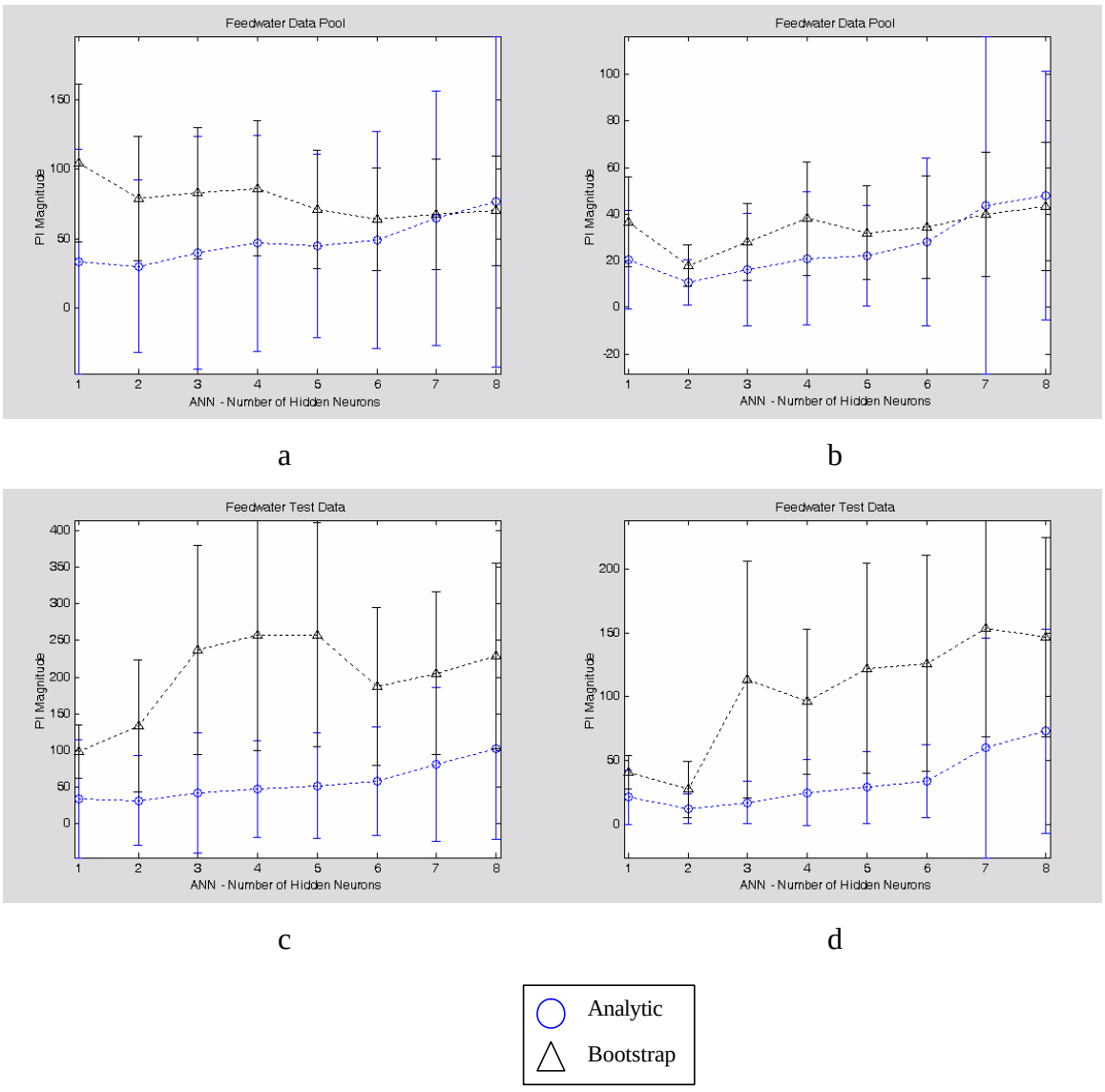


Figure 5.4.7: PI magnitude results for ANN models of feedwater data pool and test data. The bands represent the 1*s* variation in the results over the 100 iterations. (a) and (c) provide results for all 100 iterations and (b) and (d) provide results for a reduced set of models as explained below.

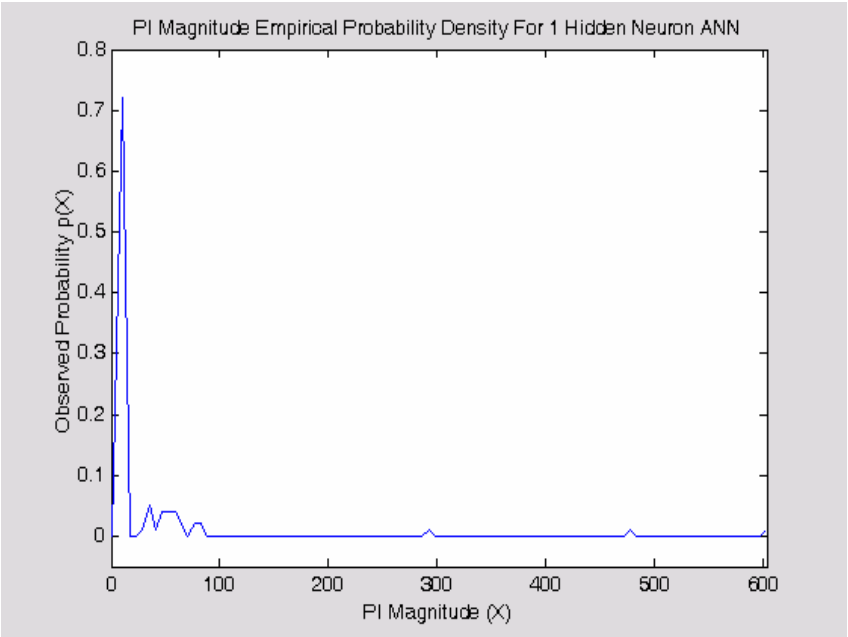


Figure 5.4.8: Empirical probability density of PI magnitudes for 1 neuron ANN feedwater flow models.

After eliminating the results from the ANNs that exhibited MAE values greater than a specified threshold, a set of modified PI Magnitudes were obtained (figures 5.4.7.b and 5.4.7.d). The threshold was specified as the mean plus 1 standard deviation of the MAE values from the 100 iterations. These mean PIM results from the reduced set of models are much lower than the original set indicating the extreme influence of a few poorly trained ANNs on the overall results. Note the extremely large change between the bootstrap values with respect to the data pool in figure 5.4.7.a and those in figure 5.4.7.b. For example the 2 hidden neuron result in figure 5.4.7.b for mean bootstrap PIM was ~20KLB/HR, whereas in figure 5.4.7.a the mean bootstrap PIM magnitude was ~80KLB/HR. The number of ANNs eliminated to obtain the reduced sets were: for the 1 hidden neuron case, 3 ANNs were eliminated and for the 8 neuron case 14 ANNs were eliminated. For all other cases the number of eliminated ANNs was between 3 and 14.

Analogous results were obtained for the test data (figure 5.4.7.c and 5.4.7.d). The reduction in the PIMs for the test data follows what was observed for the case of the data pool; however, the values are still excessively high. Note that the reduction for the analytic PIMs is not as drastic for the test data as it was for the data pool when the poor performing ANNs were eliminated. The interpretation of these results can be best summarized by stating that the ANN models for the feedwater flow data set were under-regularized. While the use of early stopping via cross-validation provides some regularization, it is not enough. This is evidenced by the high average PIM values for the data pool and their significant increases incurred when computed for the test data. The bootstrap PI values are strongly dependent on the distribution of the estimations from the 100 iterations, or fewer if poorly performing ANNs are eliminated. Thus, the results show a greater variance for the test estimations with respect to those for the data pool.

The corresponding coverage values for the ANN models are shown for all models in figures 5.4.9.a and 5.4.9.c, and for the reduced set of models in figures 5.4.9.b and 5.4.9.d.

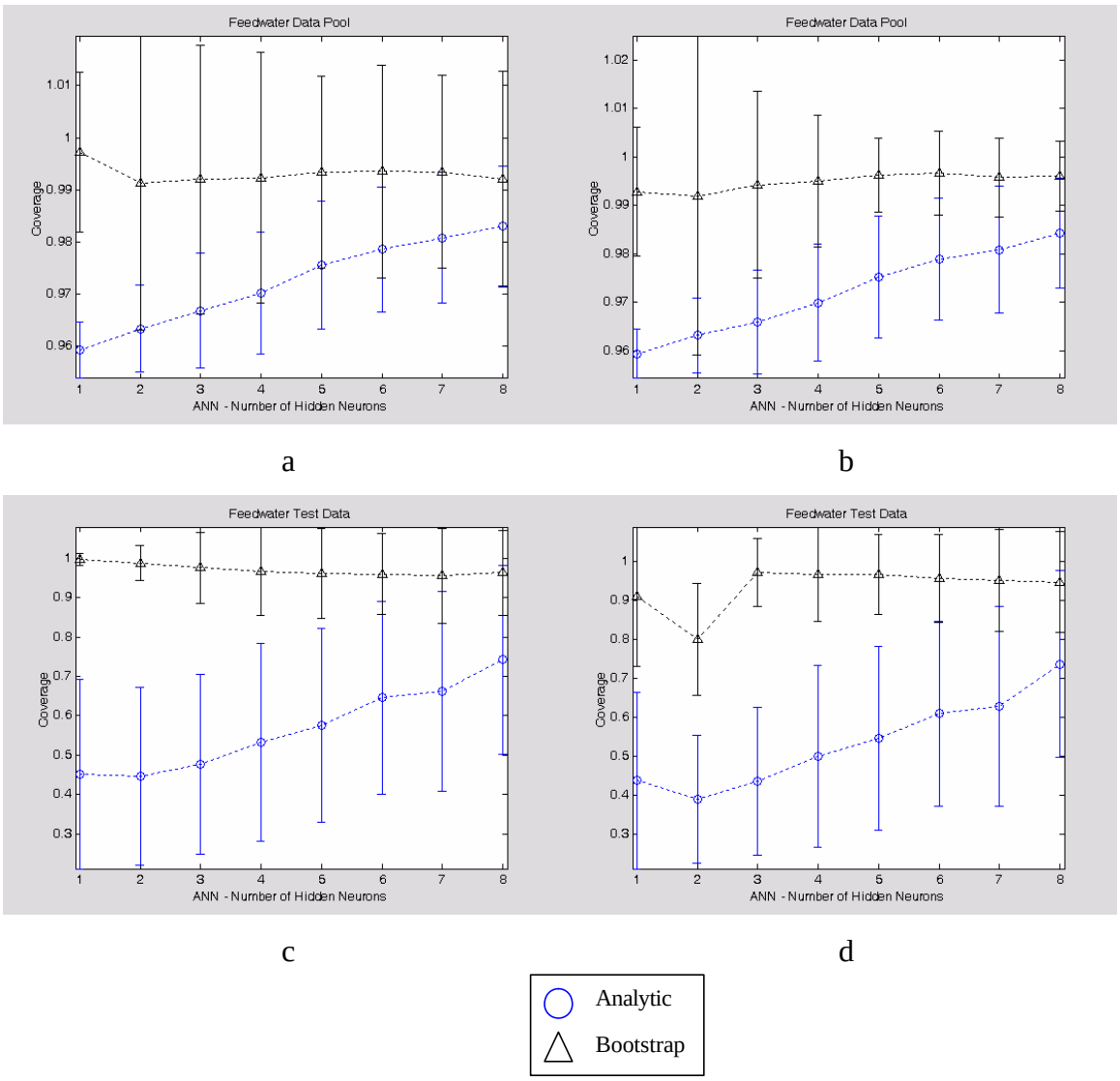


Figure 5.4.9: Coverage values for ANN models of feedwater data pool and test data. The bands represent the 1*s* variation in the results over the 100 iterations. (a) and (c) provide results for all 100 iterations and (b) and (d) provide results for a reduced set of models as explained below.

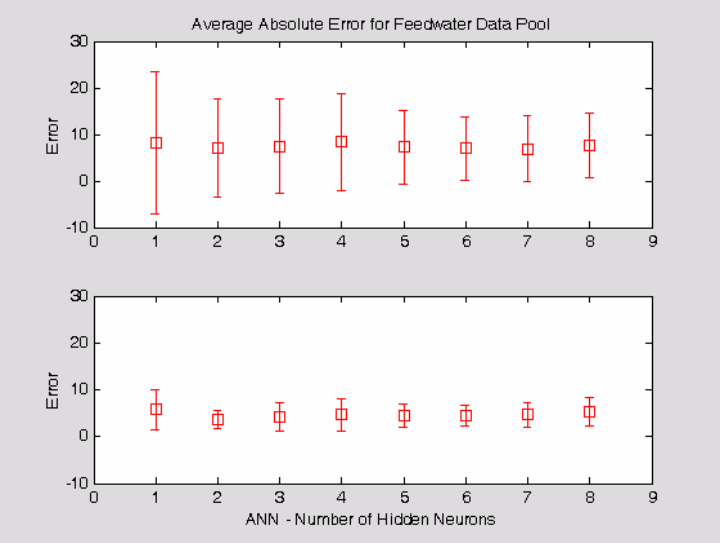
Upon reevaluation of the coverage values for the modified prediction intervals, after removing the poor performing ANNs, it was found that there were no significant changes (figure 5.4.9). This is because the eliminated ANNs had extremely high prediction intervals, and thus their corresponding coverage values were very near or equal to 1.

The instability of the ANN solutions for the feedwater flow data is reflected in the large prediction intervals. Consequently, on average the point-wise prediction intervals and corresponding estimations for the test data bound the majority of the test measurements. In this case it is known that the test measurements are not accurate; thus, the coverage of these measurements is not the desired result. The high coverage values point to the large uncertainty involved in the ANN models for this highly collinear data set and indicate that further regularization is required.

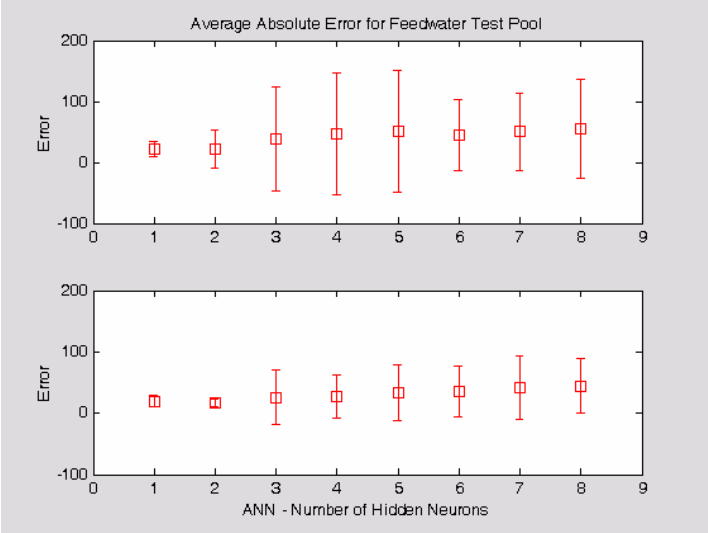
The MAE results for the feedwater flow rate data pool are provided in figure 5.4.10.a, and for the test data in figure 5.4.10.b. Extremely large variation is seen in the results for the full set of models with respect to both the data pool and the test data. The reduced set of models reduces this variation to a moderate level; however, it is still relatively high. Using 1 or 2 hidden neurons provided the most reasonable results for the ANN models, though the resultant prediction intervals were significantly larger and more variable than for the NNPLS models. As can be seen in tables 5.4.2 and 5.4.3, the ANN models resulted in the largest prediction intervals overall.

Based on the evaluations, the 2 hidden neuron model was selected as the most appropriate ANN architecture. To illustrate possible results for a 2 hidden neuron model, 4 models were selected at random from the 100 and their results plotted in figure 5.4.11.

While the lower 2 subplots (figure 5.4.11) provide the appropriate results for the test data, the upper left plot shows a case where the ANN model was poorly trained. The extremely large prediction intervals reflect this, and therefore drift detection is not



a



b

Figure 5.4.10: Mean absolute error results for ANN models of feedwater flow rate data. The bands represent the 1*s* variation in the results. The upper subplots provide the results for all 100 models and the lower subplots provide the result for the reduced set of models. (a) – data pool, (b) - test data

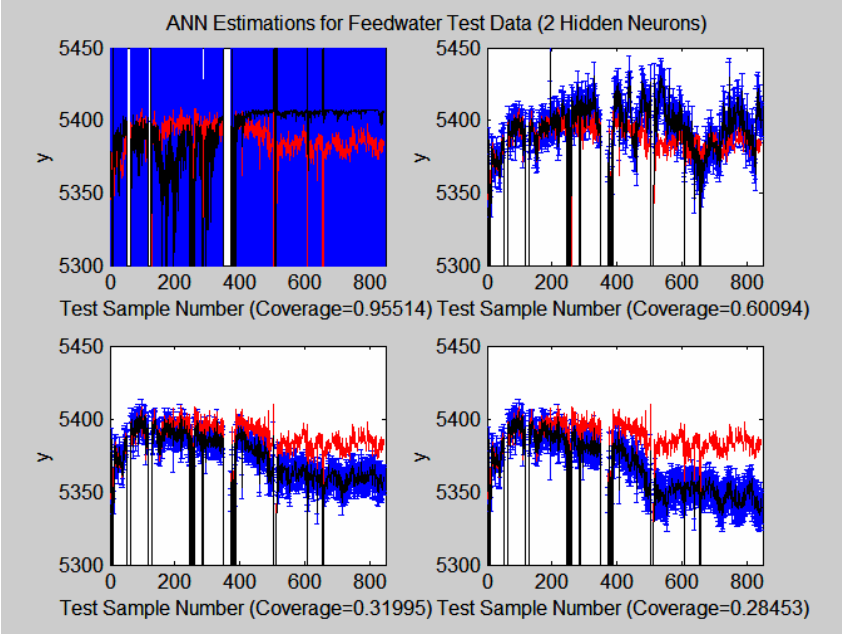


Figure 5.4.11: Test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 available 2 hidden neuron ANN models. The coverage value for each case is indicated on the x-axis.

available from this model's solution. The right upper subplot shows another scenario where the ANN training again produced a poor model. Consider the results produced by the 2 hidden neuron model which exhibited the median value of MAE over all 100 models (figure 5.4.12). This median result shows that a decent ANN model is achievable for the feedwater data set; however, the fluctuations in the solutions from one model to the next produce highly variable results.

Figure 5.4.13 shows the large variation in the estimated drift from the 100 2 hidden neuron ANN models. The lower subplot only includes drift estimates from models whose coverage of the test measurements was 5% or less from test sample 700 to 847, which for the 2 hidden neuron ANN models results in 78 models out of 100. Even after removing 22 of the models with high variation, the distribution of the drift estimates was still relatively wide.

Overall, the ANN models for the feedwater flow rate data resulted in large prediction intervals that were highly variable. Both the bootstrap and analytic intervals were extremely large in magnitude. The coverage of the computed intervals was consistently at or above the expected level of 95% for the data pool. Thus, the prediction intervals reflect the high uncertainty associated with ANNs when dealing with collinear data. While, the ANN models are not the best choice for this data, the prediction interval methods performed as expected.

5.4.3 Kernel Regression Models and Analyses

The analyses for the KR models will provide interpretations of the results, a description of the optimal bandwidth determination, and an analysis of the effect of the conditional variance of the response estimate on the magnitude of the prediction intervals.

The PIMs and coverage values for the KR models are provided in figure 5.4.14. The coverage values corresponding to the smallest bandwidths require some interpretation (figures 5.4.14.c and 5.4.14.d). Because the bandwidths are very small, there will be a

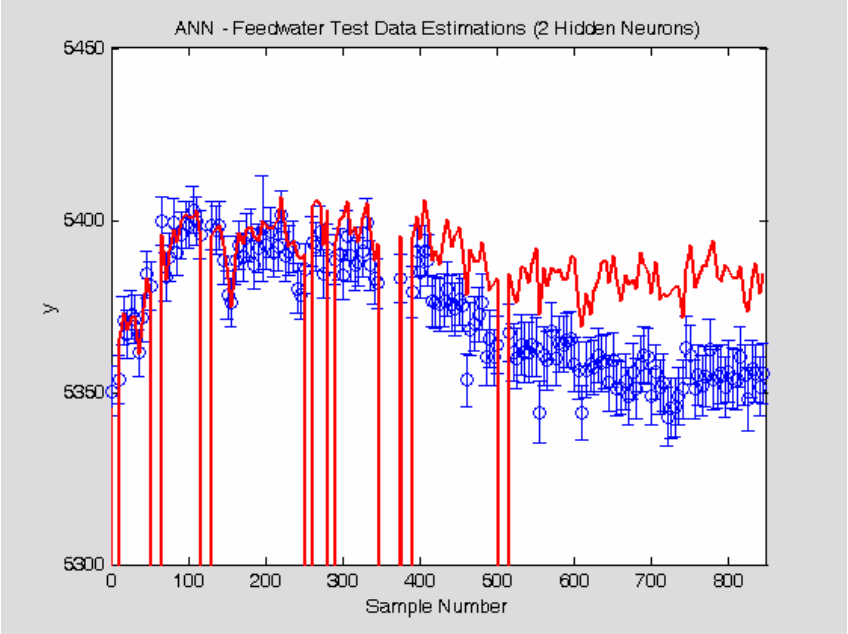


Figure 5.4.12: ANN median result estimation and analytic prediction intervals. Representative results for 2 hidden neuron ANN model showing every 5th estimate and the corresponding analytic prediction intervals. This model exhibited the median value of the 100 Mean Absolute Error results for the 2 hidden neuron models.

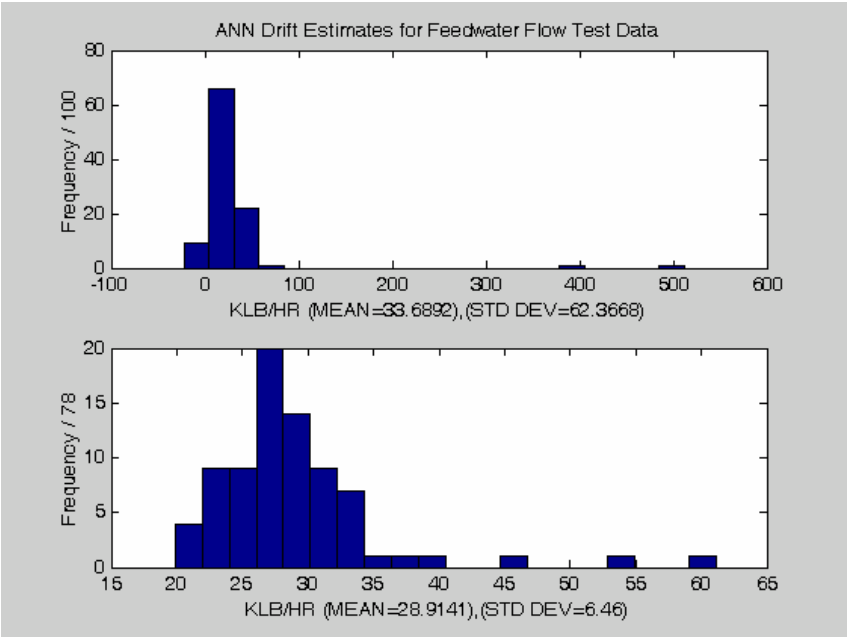


Figure 5.4.13: Distribution of drift estimate for 2 hidden neuron ANN models. Drift estimates obtained from the 100 ANN models via computing the MAE over test samples 700 to 847. The upper subplot shows all results, while the lower subplot excludes any results for which coverage was greater than 5%.

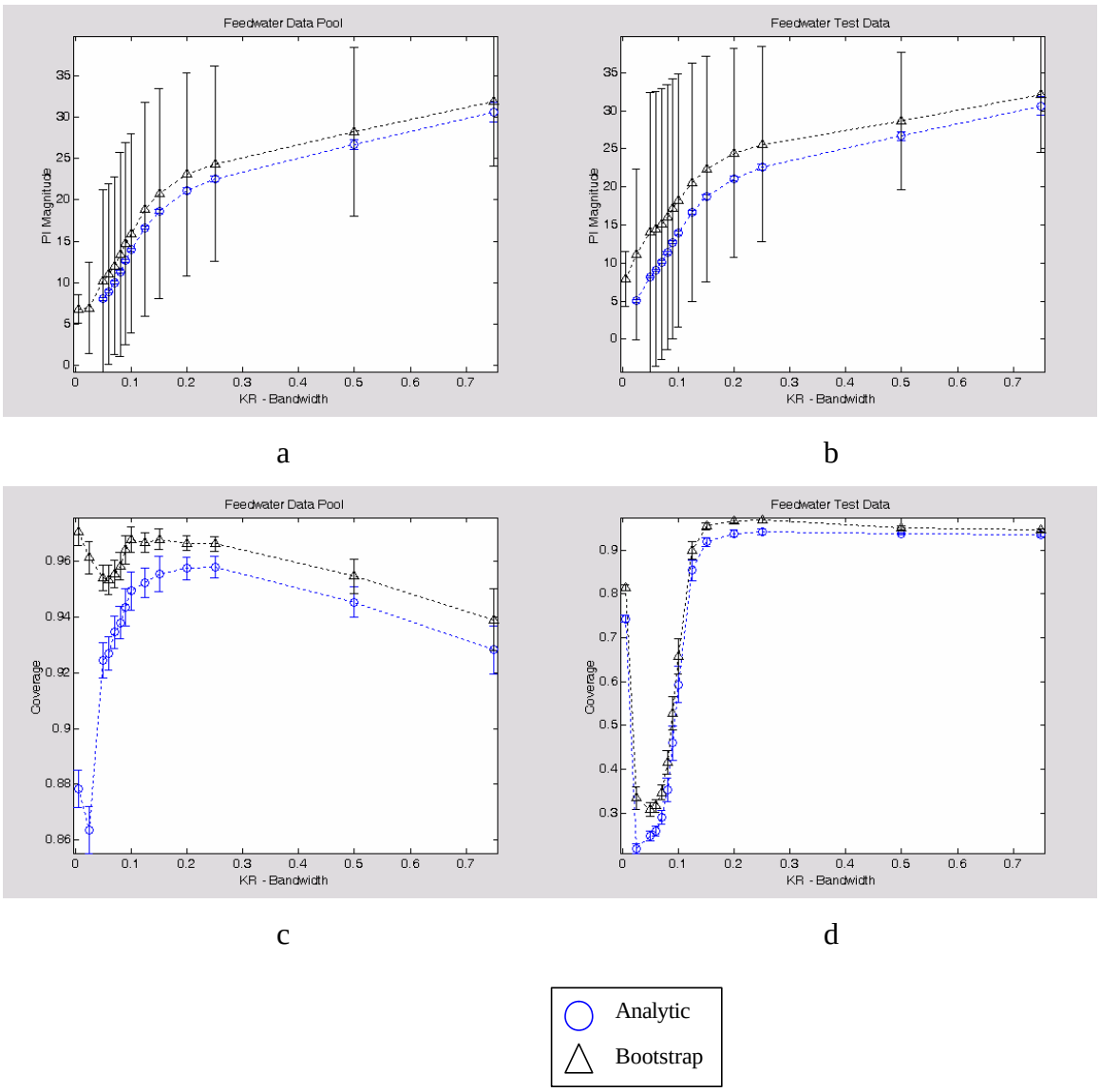


Figure 5.4.14: PI magnitude and coverage results for KR models of feedwater data pool. The bands represent the 1*s* variation in the results over the 100 iterations.

number of observations in both the data pool as well as the test data that will result in all kernel weights being zero. In computing the estimates for points as described, the KR computations will result in a division by zero; hence, the estimate will be returned as NaN. Similarly the PI Magnitude will be NaN. The result of this is that observations with associated PI magnitudes of NaN will be counted as *covered* in the determination of the fractional coverage. These points can be removed of course; however, they were retained because logically it makes sense that since the model cannot provide an estimate for the current observation, the associated uncertainty for that observation is unbounded. It is necessary that these values be removed to calculate the PIMs, otherwise the average would also be returned as NaN. It is important to note that these numerical issues are only relevant for the very small bandwidths that should not be used in practice. For reasonable bandwidths values, estimates are available for all of the observations.

As the bandwidth values increase beyond approximately 0.1 for both the data pool and the test data, the coverage values indicate that >90% of the measurements are bounded by the prediction intervals. While in the case of the data pool, this seems reasonable, in the case of the test data it is not. In both cases for large bandwidths, the solution is over-regularized. An over-regularized model produces a nearly constant estimation for the response, and the larger values of the PIMs (figure 5.4.14) for the over-regularized solution result in prediction intervals which bound all of the measurements. This eliminates the model's ability to detect the drifting feedwater flow rate channel.

Considering the PIMs for the KR data pool and test data, it is obvious that the bootstrap PIMs have a much greater variation than the analytic PIMs. This indicates that there are sources of uncertainty that are not being properly accounted for in the analytic prediction interval computations.

Investigating the source of variability in the bootstrap PIMs leads to an illustration of the deviation of the 100 individual KR model estimations from the mean value of the 100 estimations (figure 5.4.15). The plot illustrates that the majority of the variability is

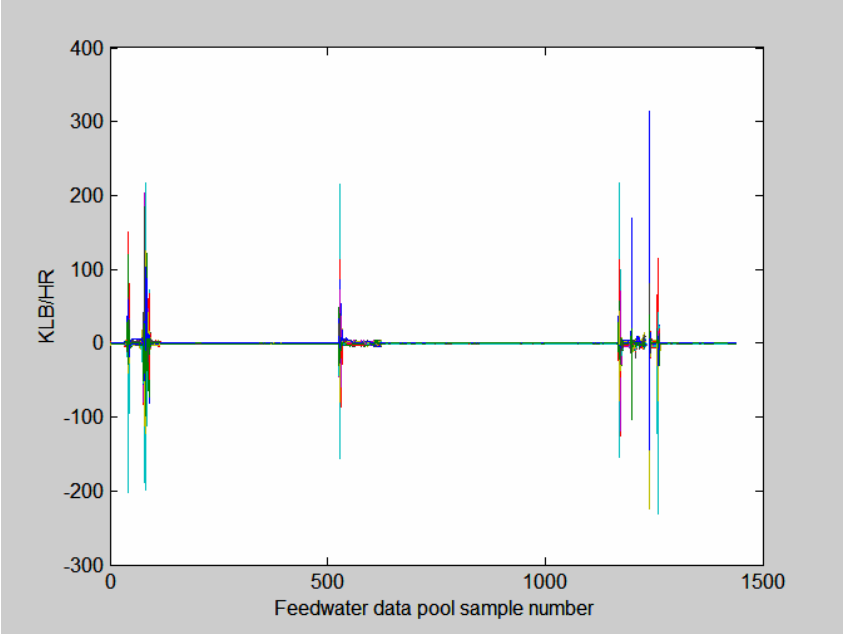


Figure 5.4.15: Deviation of estimates from the 100 KR models of the feedwater flow data pool with respect to the mean estimate for each observation.

centers around the regions of the data pool where transients exist. Because the values in figure 5.4.14 are based on overall averages, the results are strongly influenced by the observations exhibiting the large variability in the response estimates. Due to the nature of the process changes during transients the relationships of a static model may not be consistent. Comparing figure 5.4.15 to figure 5.4.1 clearly indicates that the problematic observations occur during transient operations. Incorporating time based information into the empirical models, through the addition of time-delayed variables in the predictor variable set, would provide better performance during the transient periods.

Recall that the bootstrap computations result in a single set of prediction intervals for the data pool, derived from all 100 estimations. The bootstrap prediction intervals for the data pool are shown below in figure 5.4.16. The obvious correlation between figures 5.4.16 and 5.4.15 indicate that the bootstrap prediction intervals are extremely large for a relatively small number of observations. Moreover, these observations occur in transient regions with respect to the feedwater flow rate response.

A similar effect occurs for the analytic prediction interval computations (figure 5.4.17); however, due to the limits on the variance component of the computation the variation is much more reduced. For the observations where the bootstrap prediction intervals exhibit excessively large values, the analytic intervals correspondingly are at their maximum values. The maximum value for the analytic prediction intervals will be discussed later in this section.

Figure 5.4.18 shows the original bootstrap average PIMs along with the PIMs from the modified set of observations (modified set described below). The reduction in variability is inherently obvious, while changes in the overall mean values were only slight.

The overall averages for the original intervals are shown in the upper plot, and the averages over the modified set of observations are shown in the lower plot. The modified set was created by eliminating observations where the PI magnitude was greater than 20

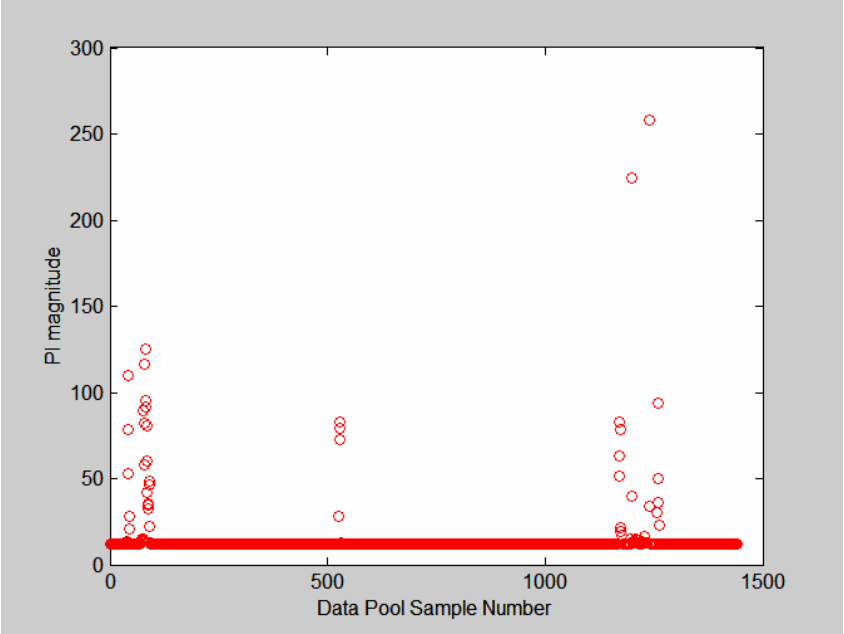


Figure 5.4.16: Bootstrap prediction intervals for KR models of the feedwater flow rate data pool, bandwidth = 0.025.

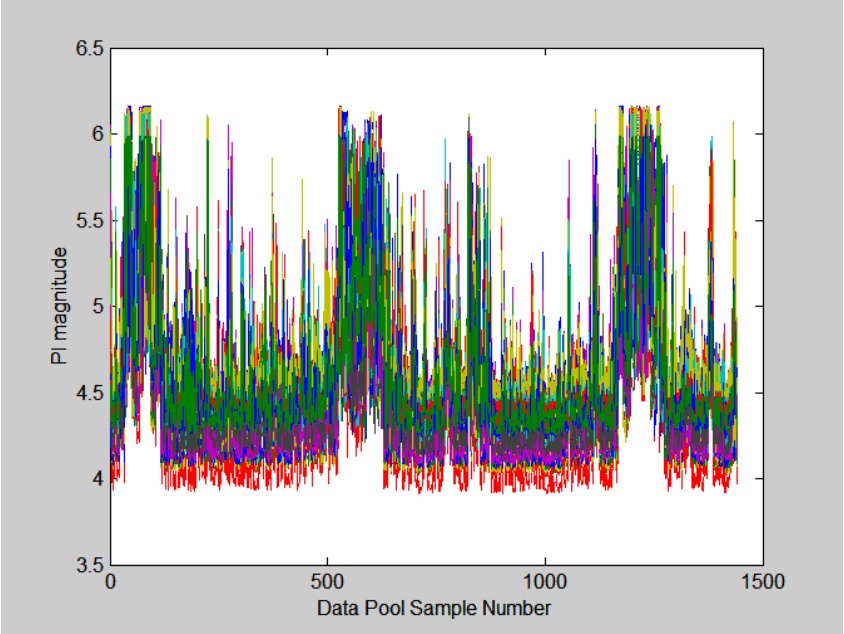


Figure 5.4.17: Analytic prediction intervals for all 100 KR models of feedwater flow rate data pool, bandwidth = 0.025.

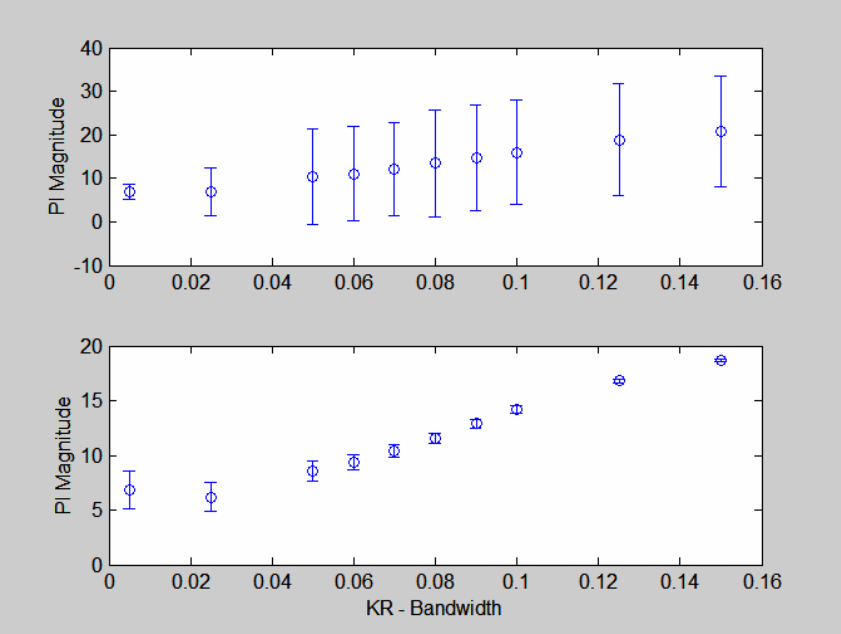


Figure 5.4.18: Original and modified bootstrap prediction intervals for KR models of the feedwater flow rate data pool.

KLB/HR (see table 5.4.4). The bands represent the 1s variation in the results over the 100 iterations.

The number of excluded observations as a function of bandwidth is shown in table 5.4.4. Note that results for bandwidths >0.15 were not computed due to the large bias dominating the prediction interval computations, requiring a higher threshold for removal of observations.

The increase in the number of observations resulting in largely varying estimates is because as the bandwidth increases, the range of influence of the training vectors increases, e.g. a vector causing large response variations can only affect 10 points for a small bandwidth value, but can affect 30 points with a correspondingly larger value.

Figure 5.4.19 shows the original bootstrap coverage values along with the values resulting from the modified intervals. In both cases the expected coverage level is achieved or exceed for all bandwidths. This indicates that the reduction in the bootstrap intervals did not affect the ability of the computed intervals to adequately contain the measurements of the data pool.

The question of how will this estimation variance affect the performance of these intervals in practical applications can be answered by considering that the coverage of the analytic intervals remains consistent. The bootstrap intervals tend to overestimate the variability in the estimates, and the observed effects exaggerate the prediction interval magnitudes. In addition, in practical applications models are often specialized over a specific range of the response. Limiting the range of data will reduce or eliminate the observed effects. Because the analytic intervals performed properly, and the bootstrap intervals excessive fluctuation were related to a relatively small number of observations, it is stated that the analytic prediction intervals adequately cover the measurements and compare sufficiently well to the bootstrap prediction intervals after accounting for the few observations of large variance.

Table 5.4.4: The number of observations in the data pool for which the KR estimate's bootstrap PIM was >20KLB/HR

Bandwidth	0.005	0.025	0.05	0.06	0.07	0.080	0.090	0.100	0.125	0.15
# Observations Removed	0	22	36	37	38	39	40	39	48	53

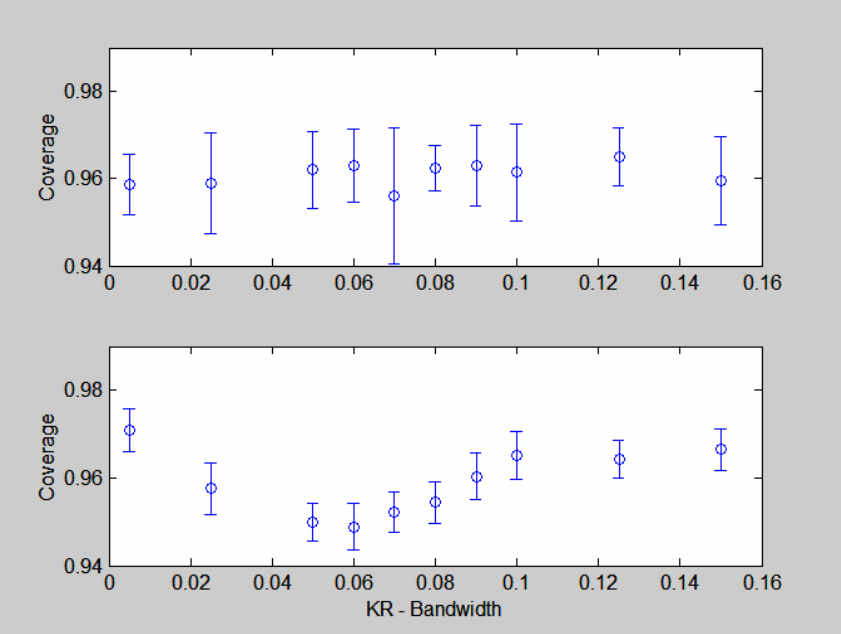


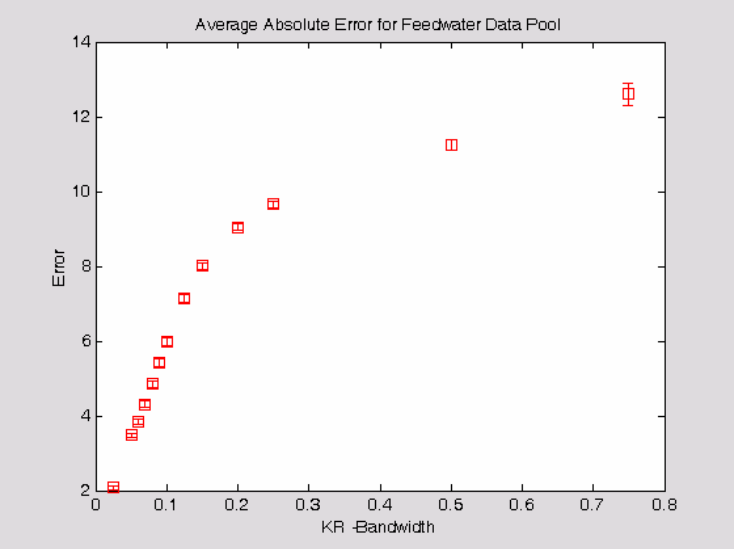
Figure 5.4.19: Original and modified bootstrap prediction interval coverage values for KR models of feedwater flow rate data pool.

The overall averages for the original values are shown in the upper plot, and the averages for the modified set of observations are shown in the lower plot. The modified set was created by eliminating observations where the PI magnitude was greater than 20 KLB/HR (see table 5.4.4). The bands represent the $1s$ variation in the results over the 100 iterations.

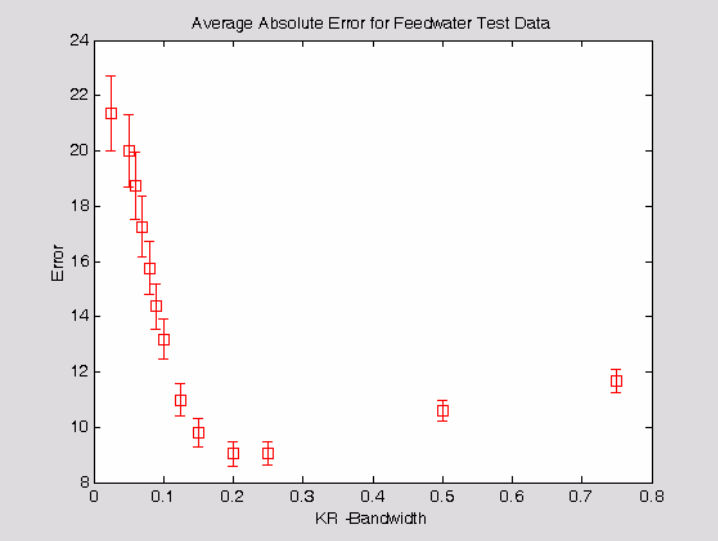
Consider the MAE values for the KR models with respect to the data pool (figure 5.4.20.a) and the test data (figure 5.4.20.b). The lower errors at lower bandwidths for the data pool are expected due to the smaller level of model bias. As the bandwidth increases, the error increases. Regarding the test data, the opposite trend is noted. In this case, the larger errors at lower bandwidths are correct due to the drift in the channel being estimated. The test error decreases to a minimum as the bias influences the estimation. Eventually the bias becomes excessively large and the error increases as the overall response tends toward a constant value.

To determine the bandwidth, the usual method of optimizing the trade-off between variance and bias can be followed. Figure 5.4.21 shows this relationship based on the training data. The optimal bandwidth based solely on the training data would be ~0.02. When considering only the training data, a bias value near zero is achievable for very small bandwidths. As the bandwidth increases the bias increases. On the other hand, for small bandwidths the variance is at its maximum because the KR model is mapping every point in the training set exactly. The maximum variance is thus equal to the conditional variance estimate of the response.

Note that the squared bias in figure 5.4.21 is simply the training MSE. While theory provides that the MSE of the kernel regression estimator is equal to the sum of the variance and squared bias of the estimator, in practice at small bandwidths this is rarely observed. This is due to the use of a limited set of data that is assumed to adequately describe the predictor variable space. Of course attempts are made to provide adequate representation in the training data, though this assumption can never be fulfilled entirely using data recorded from process sensors. Other contributing factors to the cumulative sum of squared bias and variance not being equivalent to the MSE are the existence of noise in the predictor variables, and the estimation of the response variable noise. Because in practice subtracting the variance from the MSE, at small bandwidths, would result in negative values this approach is not used for bias estimation. Further, if this approach were carried out, the decrease applied to the bias is negligible in most cases. To



a



b

Figure 5.4.20: Mean absolute error results for KR models of feedwater flow rate data. The bands represent the 1*s* variation in the results over the 100 iterations. (a) – data pool, (b) – test data.

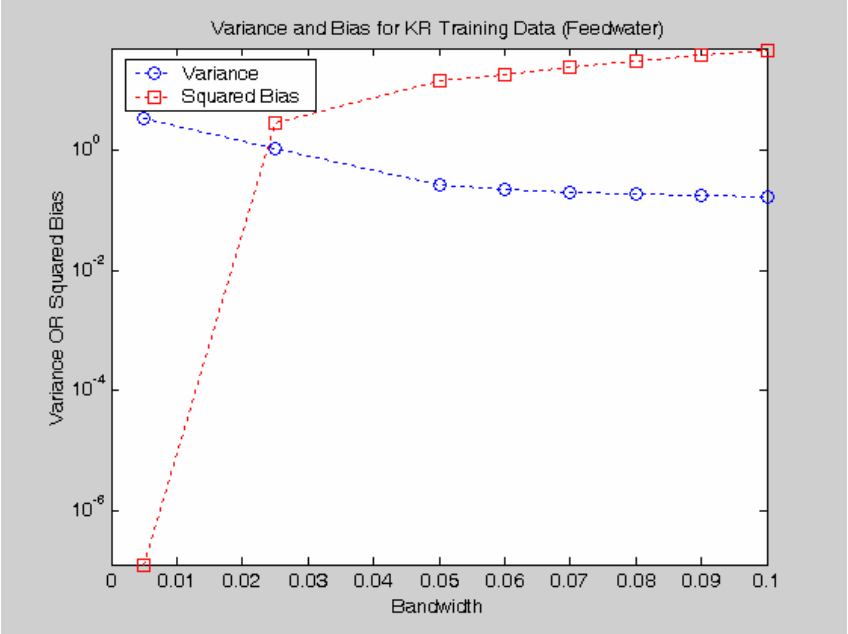


Figure 5.4.21: Variance / squared bias plot for KR models of feedwater flow rate training data.

illustrate this, consider Figure 5.4.22 which shows the difference between the MSE and Variance for the training data and compares this to the MSE itself. From this figure it is apparent that using the MSE as the bias estimate will not overly influence the overall prediction interval computation.

Another consideration is that the bias estimate for the training data will be significantly different than the bias estimate for another independent set of data randomly sampled from the same data pool. Figure 5.4.23 shows this result by comparing the bias and variance estimates obtained for the training data with those obtained for a validation data set.

Because the models developed will be used to provide estimations for data that is independent from the training data, it is important that the bias estimate be applicable to all data of the given data pool. To combine all of this information to choose the optimal bandwidth the bias estimate utilized is the overall mean of the training and validation data MSE, and the variance estimate is that obtained for all of the data in the data pool. Figure 5.4.24 provides this information and points to an optimal bandwidth choice of 0.025. Note that the computations of variance and bias for the bandwidth value of 0.005 eliminate a sufficient amount of NaN estimates, as discussed above; therefore, the *true* variance for this bandwidth is much higher. For the next larger bandwidth of 0.025, estimates are available for all observations in the data pool.

Using the optimal global bandwidth value (0.025), the estimates for the test data from 4 different KR models are presented (figure 5.4.25). To provide a representative result in larger detail, the estimations and prediction intervals from the model which exhibited the median value of MAE, out of the 100 possible KR models with a bandwidth of 0.025, are shown in figure 5.4.26. In addition, an over-regularized solution is presented to illustrate the effect of a large bandwidth and its associated biasing effect (figure 5.4.27).

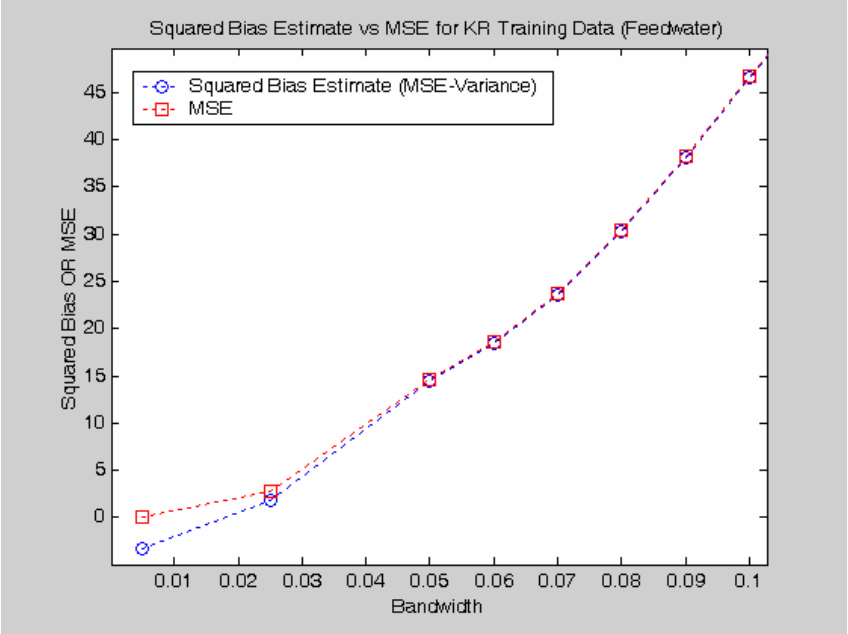


Figure 5.4.22: Comparison of feedwater flow rate data MSE and MSE-Variance.

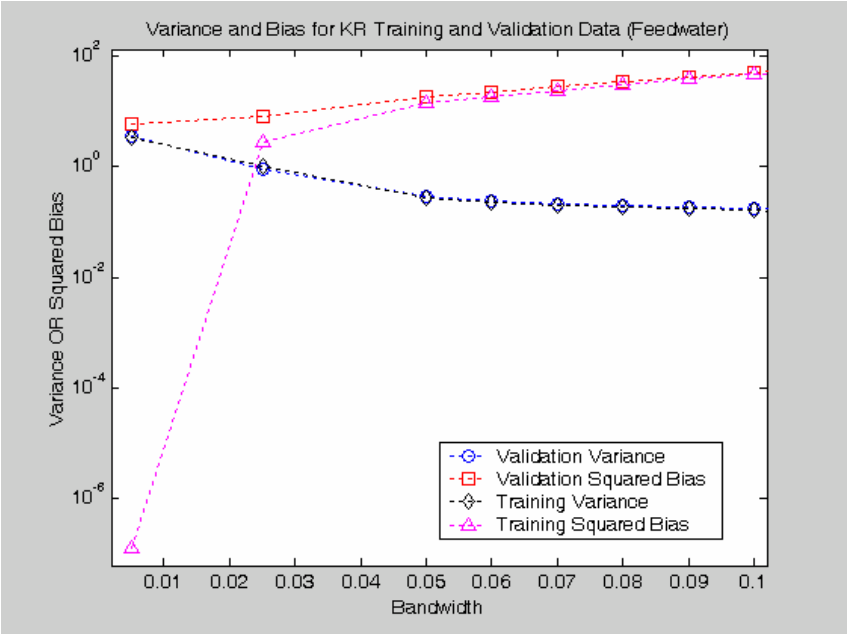


Figure 5.4.23: Variance / squared bias plots for KR models of feedwater flow rate training data and validation data.

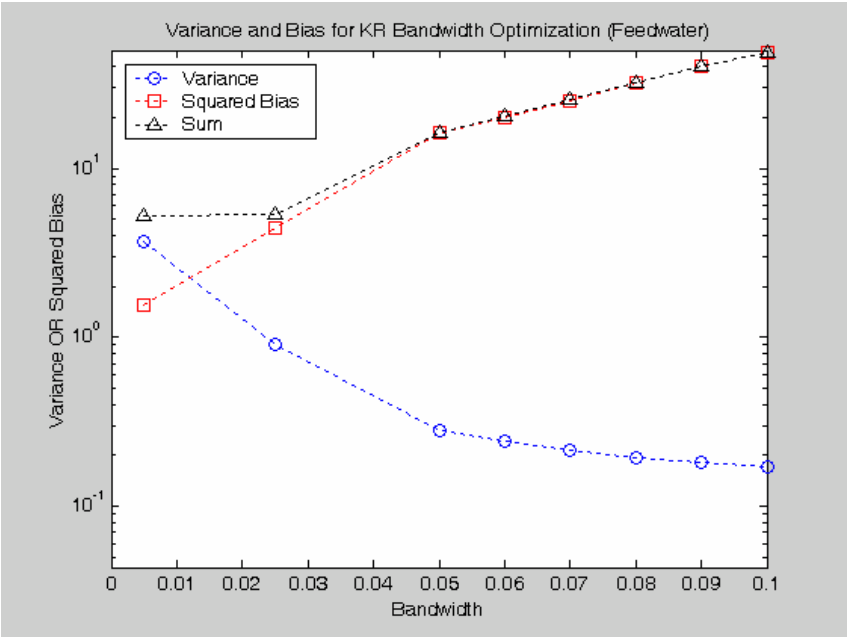


Figure 5.4.24: Combined variance of training and validation data vs. squared bias of data pool for KR models of feedwater flow rate data.

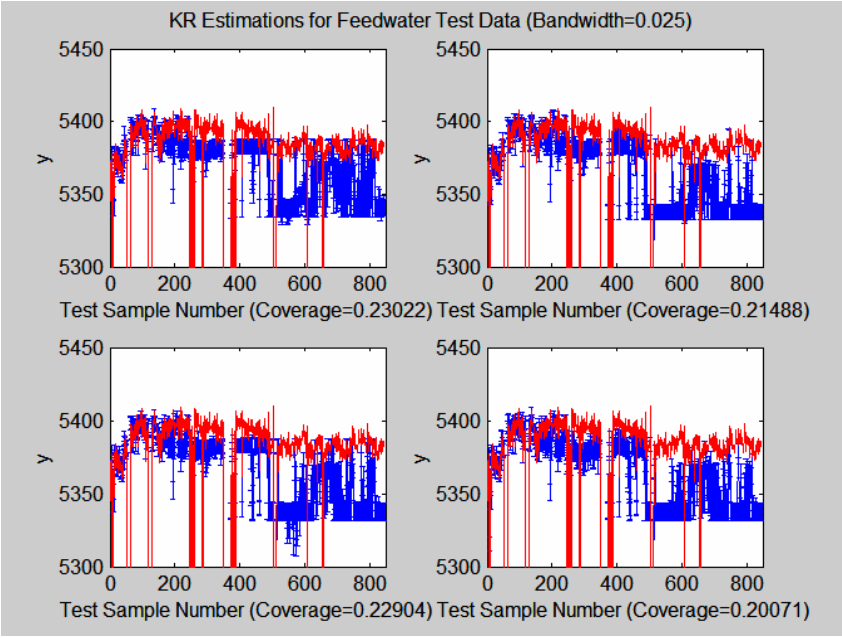


Figure 5.4.25: Test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 available KR models with bandwidth 0.025. The coverage value for each case is indicated on the x-axis.

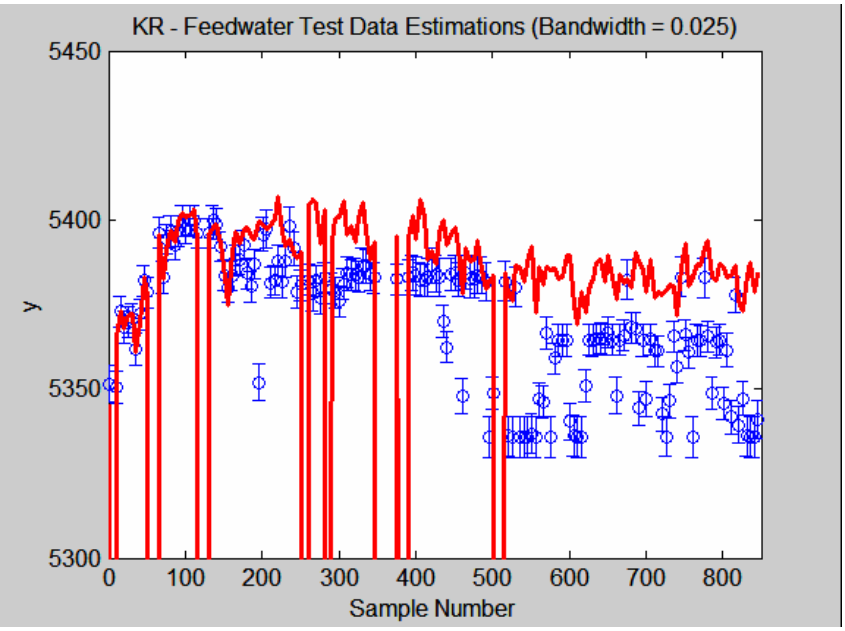


Figure 5.4.26: KR median result estimation and analytic prediction intervals. Representative results for KR model of bandwidth=0.025 showing every 5th estimate and the corresponding analytic prediction intervals. This model exhibited the median value of the 100 Mean Absolute Error results for the KR models with a bandwidth of 0.025.

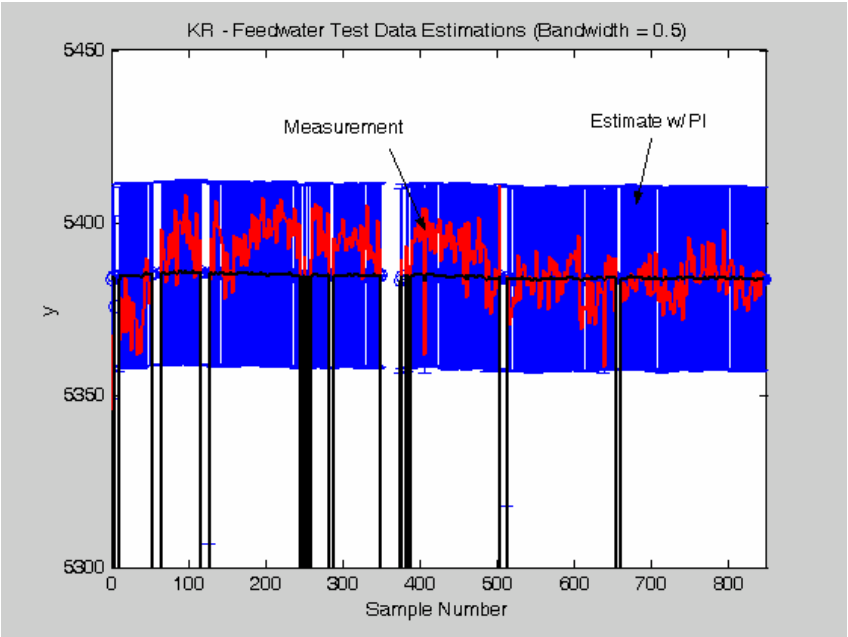


Figure 5.4.27: Example of an over-regularized fit of a KR model to the feedwater flow rate test data (bandwidth=0.5).

As a final study, the effect of the estimate of the conditional variance of the response with respect to the prediction interval magnitudes was analyzed. A simplified representation of the analytic prediction interval computation is: $\pm 2\sqrt{\text{var} + bias^2}$. Setting the bias term to zero, and expanding the variance term by separating out the estimate of conditional variance (assume homoscedasticity) of the response yields: $\pm 2\sqrt{s^2 \mathbf{w}_n{}^{\mathbf{T}} \mathbf{w}_n}$, where $\mathbf{w}_n$ is a vector of normalized kernel weights, and $\text{var} = s^2 \mathbf{w}_n{}^{\mathbf{T}} \mathbf{w}_n$. Due to the normalization of the weights $0 \le \mathbf{w}_n^{\mathbf{T}} \mathbf{w}_n \le 1$; therefore, $0 \le \text{var} \le s^2$ and $0 \le 2\sqrt{s^2 \mathbf{w}_n^{\mathbf{T}} \mathbf{w}_n} \le 2s$.

For the feedwater flow data set $s^2 = 4.47$, and $2s = 4.23$. Consider figure 5.4.28. The upper subplot shows the PIMs, for a KR model of bandwidth h=0.025, where $s = 1$. The plot shows that the PIMs when $bias = 0$, range from 0 to 2, confirming that $0 \le \text{var} \le 2s$ (when $bias = 0$). Adding in the bias component increases the magnitude noting that for this case, the average squared bias is ~4.5[KLB/HR]^2. The middle subplot also shows the PIMs with and without the bias component, but in this case the actual variance estimate is inserted ($s^2 = 4.47$). Finally the lower subplot provides the same information for the case of $2s^2 = 8.947$. The overall trend is obvious, that the PIMs increase as the conditional variance increases. The trend also exists when the bias component is included in the PIM computations; however, the differences are less significant as the variance estimate increases. For the problem at hand, both the middle and lower subplots provide prediction intervals that will indicate that the feedwater test response has drifted. Thus, in this case the sensitivity of the PIMs to the conditional variance estimate is not very significant. This is not always the case and in situations where the conditional variance is significantly lower than that used here the sensitivity to the estimate is much more pronounced.

The discussions regarding the estimation of the variance of the response intend to ensure that the results presented are stable and repeatable, and not dependent on the specific response variance estimate. Proper implementation of prediction intervals requires

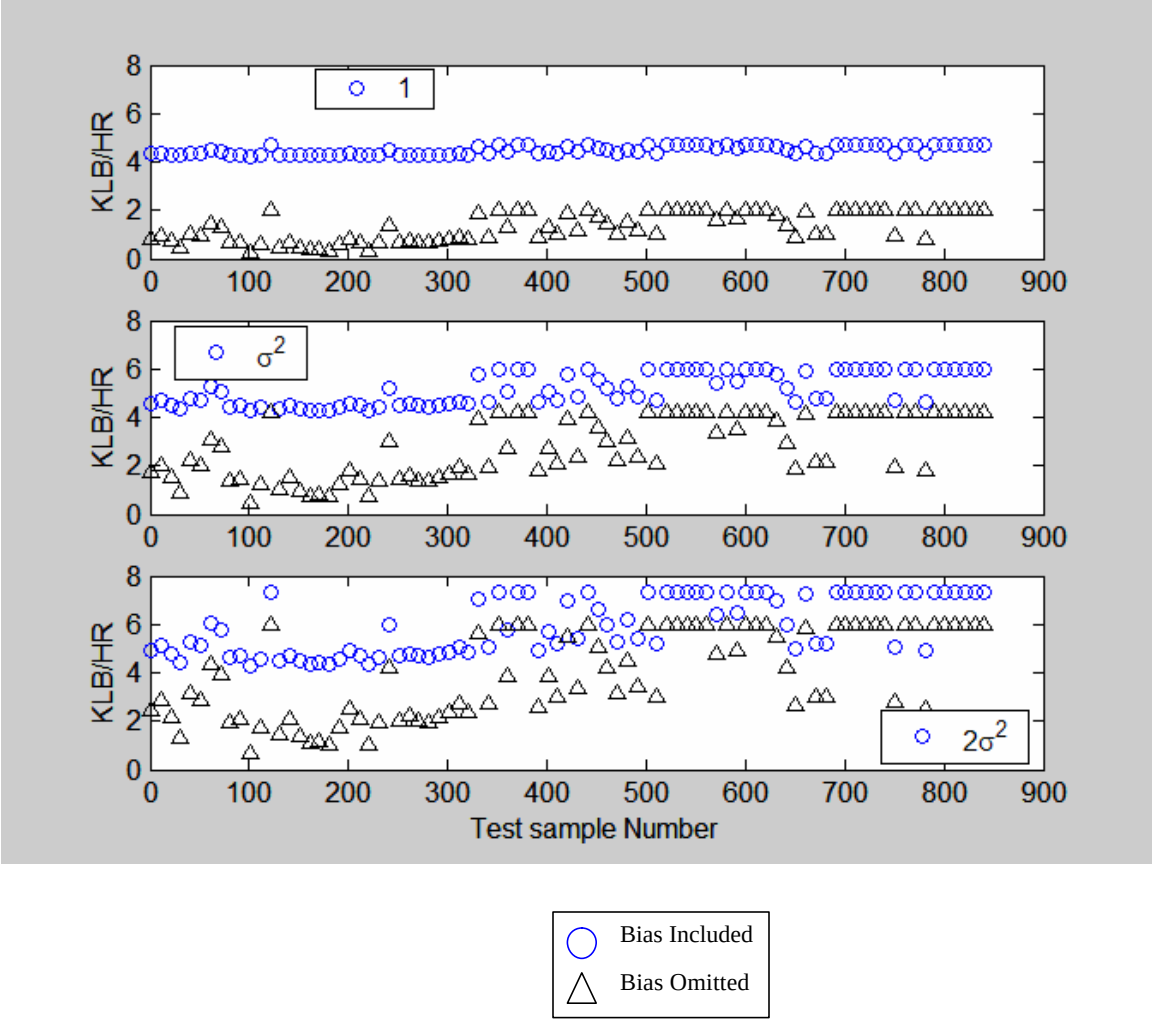


Figure 5.4.28: Analytic prediction interval magnitudes, of feedwater test data KR models, for various conditional variance estimates.

Each subplot contains the prediction intervals with and without the bias component for the KR models. The model bandwidth for the case presented is 0.025. Each subplot provides the results for a different value of the conditional variance estimate: upper – 1, middle - $s^2 = 4.47$, and lower - $2s^2 = 8.947$. Average squared bias value, for KR models, of bandwidth 0.025 is ~4.5[KLB/HR]^2

sufficient consideration of the variance estimate, though as evidenced, the resultant PIMs are not overly sensitive to the estimate.

The distribution of the drift estimates from the 100 KR models of optimal bandwidth (0.025) are shown in the upper subplot (figure 5.4.29) and the distribution of those models exhibiting coverage values of less than 5% in the rang of test sample 700 to 847 is shown in the lower subplot.

The analytic prediction intervals for bandwidth values at or near the optimal value were the smallest on average of all the models applied to the feedwater flow estimation problem. It was also observed that the analytic coverage values were slightly less than the expected value for bandwidth values ≤ 0.1. This infers that the variance estimate is under-valued since this is the region of the bandwidth range where the variance component exerts the majority of its influence. The coverage values for the bootstrap intervals were all at or above the expected level (0.95).
While the initially computed bootstrap prediction intervals exhibited extremely large variations over the set of observations of the response variable, it was shown that the source of this variation was due to a limited number of observations. When the response estimates due to these observations were eliminated from the computations, the modified bootstrap prediction intervals were very similar to the analytic prediction intervals in mean and variance. In addition, the coverage values for the modified bootstrap prediction intervals were consistently above the expected level of 0.95.

The study of the effect of the conditional response estimate on the computed prediction intervals indicated slight changes for an estimate 2x greater than the original estimate, though considering the resultant increase in prediction interval magnitude of ~1-2KLB/HR with respect to the magnitude of the response signal estimates >5000KLB/HR, this is not considered to be a significant sensitivity.

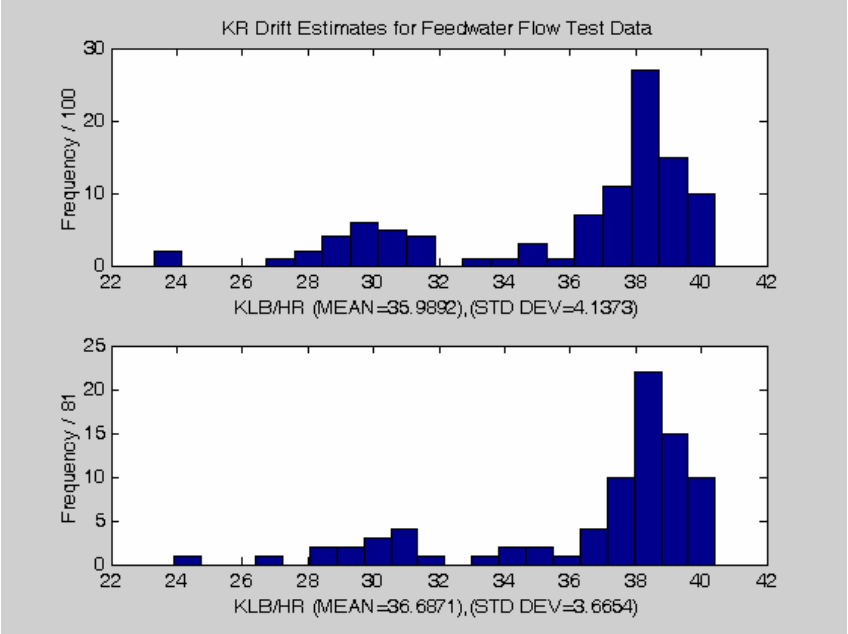


Figure 5.4.29: Distribution of drift estimate for KR models of bandwidth 0.025. Drift estimates obtained from the 100 KR models via computing the MAE over test samples 700 to 847. The upper subplot shows all results, while the lower subplot excludes any results for which coverage was greater than 5%.

The average drift estimate was similar to what was observed for the 2 latent variable NNPLS models, though for the case of KR (bandwidth = 0.025), the variation in the estimates was slightly greater.

5.4.4 Local Linear Regression Models and Analyses

The analyses for the LL models provide interpretations of the results, and a description of the optimal bandwidth determination. An evaluation of the effect of the conditional variance estimate on the prediction interval computations is also completed, as was done for the KR models in the previous section.

The analytic and bootstrap prediction interval magnitudes for the data pool and test data are provided in figures 5.4.30.a and 5.4.30.b. The corresponding coverage values are provided in figures 5.4.30.c and 5.4.30.d. Similar to what was seen for the KR models, the bootstrap intervals have excessively large variability with respect to the PIMs. Also seen here is a large difference between the bootstrap PIM average values with respect to the analytic PIM average values. The large differences between the average values are due to the limited range of the estimations from the KR models vs. the essentially unlimited range of the estimations for the LL models. The bootstrap computation involves the squared deviation of the individual estimates from the mean value (over 100 estimates) and thus is more influenced by the differences in the estimates from one model to the next. The analytic computation involves the average of the squared deviations from the measured values over the entire sampling space for a data set, thud specific influential points are averaged in and their overall effect is reduced\ with respect to the influence in the bootstrap computation. The effect is great enough to notice an increase in the variance of the analytic PIMs results with respect to the results obtained for the KR models. Further explanation for the large differences between the bootstrap and analytic average PIMs will be provided below.

The coverage results indicate typical results. For small bandwidths, the analytic coverage values with respect to the data pool are slightly lower than the expected value of 0.95

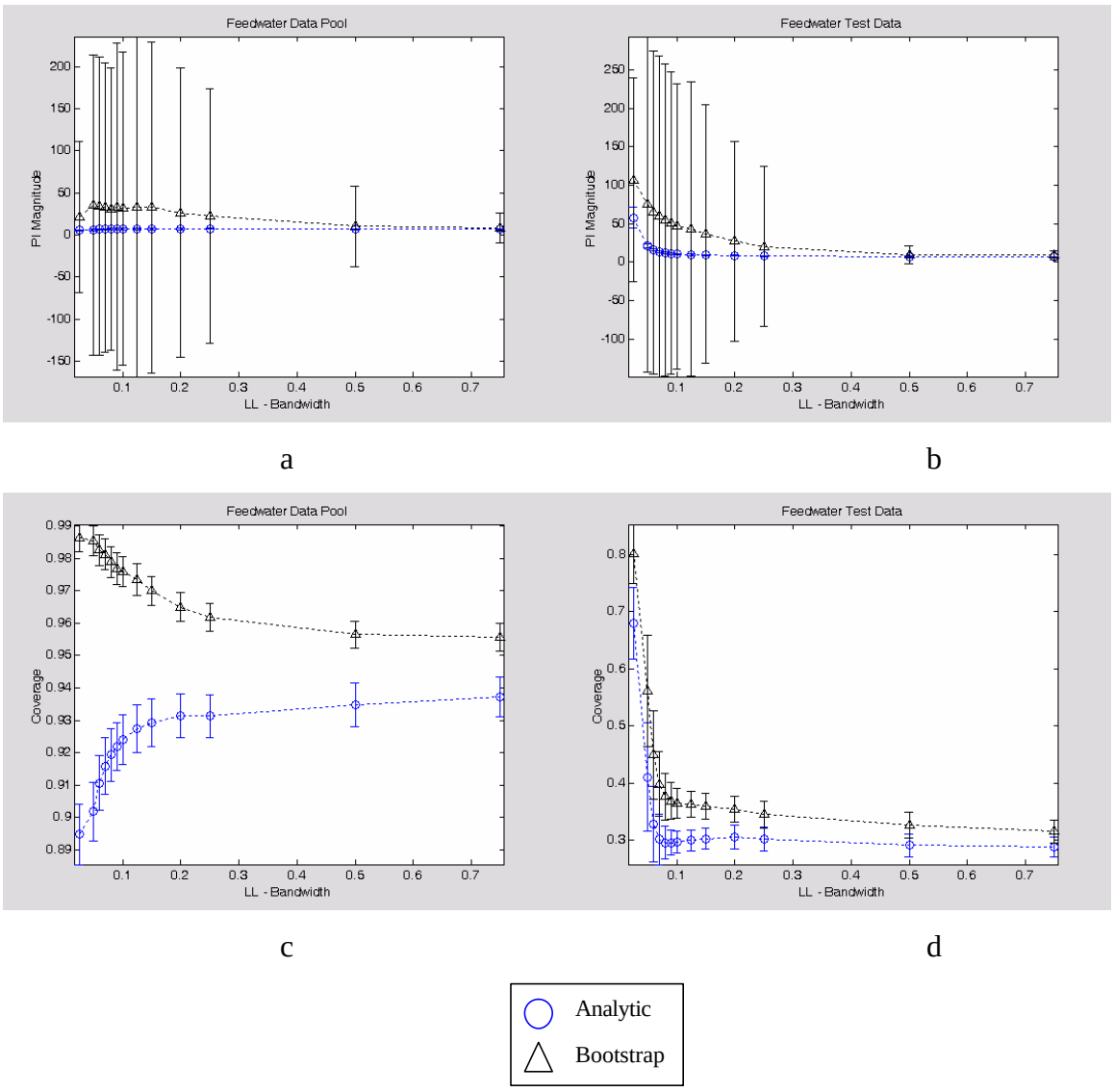


Figure 5.4.30: Prediction interval magnitude and coverage results for LL models of feedwater data pool. The bands represent the $1s$ variation in the results over the 100 iterations.

because the model is under-regularized. As the bandwidths increase the coverage values approach the expected value of 0.95. The bootstrap coverage values correspond to their large average values and variation. The coverage values for the test data are too high for the smallest bandwidths due to the large PIMs. When reasonable bandwidth values are used, the coverage of the intervals for the test data falls to the appropriate levels for this case of a drifted signal.

Figure 5.4.31 shows the set of bootstrap prediction intervals (bandwidth = 0.08) and figure 5.4.32 shows the analytic prediction intervals for all 100 LL models (bandwidth=0.08). The bootstrap intervals show behavior similar to what was seen for the case of the KR models (figure 5.4.16); however, for the LL case the effect is much more pronounced. The excessive prediction interval values can be understood if one considers the range of estimates from a KR model with respect to the unlimited range of estimates from a LL model.

Table 5.4.5 provides the empirically observed minimum and maximum estimates for the data pool from both the KR models and the LLmodels. For the previous discussion on KR recall that the estimates were bound by a minimum and maximum value based on the set of observations which comprise the training set. For LL models this is not the case. The bootstrap prediction intervals for LL exhibit a much larger variation than those for KR, and the analytic prediction intervals now exhibit behavior similar to the bootstrap intervals. This is due to the additional flexibility of the LL approach. Consider the minimum and maximum bounds for the estimations from KR vs. those from LL. The minimum and maximum values of the KR estimator are the minimum and maximum values of the response vector in the training set. For the LL regression estimator there is no minimum or maximum.

In addition the resultant numerical values in tables 5.4.2 and 5.4.3 require some interpretation. The bootstrap intervals standard deviations are computed as the standard deviation of the bootstrap PIMs for the number of samples in the data pool. Conversely,

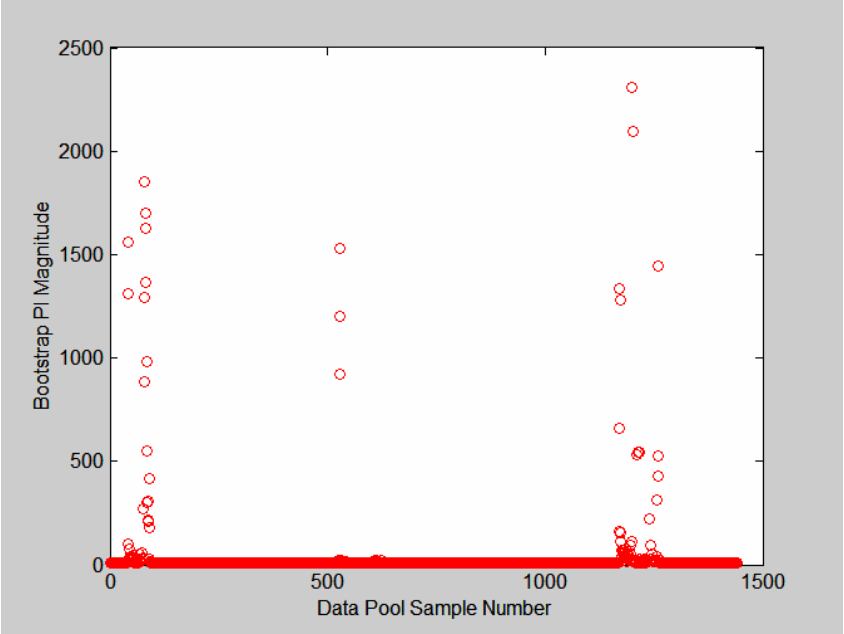


Figure 5.4.31: Bootstrap prediction intervals for LL models of feedwater flow rate data pool, bandwidth = 0.08.

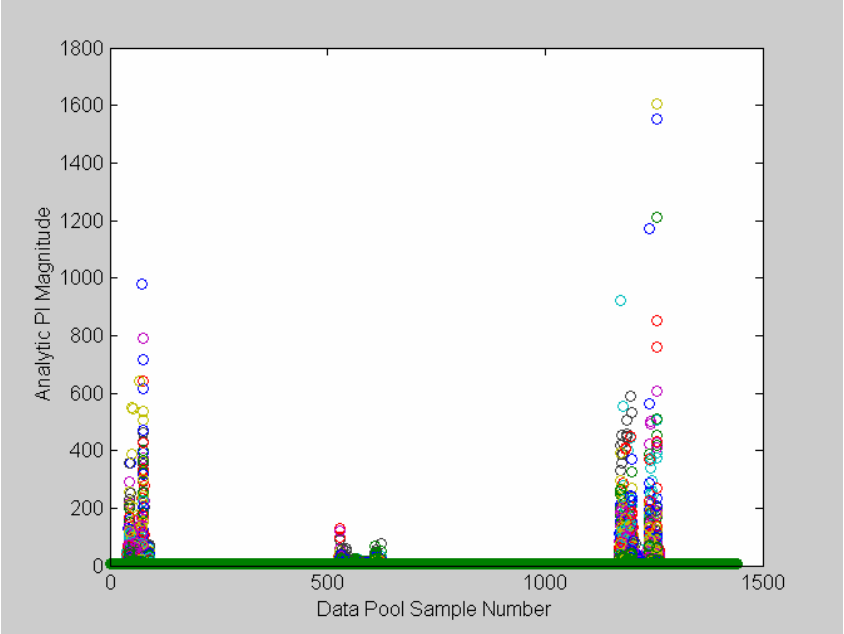


Figure 5.4.32: Analytic prediction intervals for all 100 LL models of feedwater flow rate data pool, bandwidth = 0.08.

Table 5.4.5: Minimum and maximum response estimates for KR and LL models with respect to the feedwater flow rate data pool.

	Minimum (KLB/HR)	Maximum (KLB/HR)
Training Response	3620.4	5411.3
KR estimator	3620.4	5411.3
LL estimator	253.3	5582.6

the analytic prediction interval standard deviation is computed as the standard deviation of the 100 separate averages over the number of samples in the data pool. Thus, the bootstrap interval standard deviations show much greater variability than the corresponding analytic prediction intervals.

Again, the limited number of observations which result in these excessively large PIMs can be removed and the computations repeated. The number of excluded observations as a function of bandwidth is shown in table 5.4.6.

The initial increase in the number of observations resulting in largely varying estimates is due to the bandwidth increase allowing influential training observations to affect a greater number of estimates. This trend levels off beyond a certain bandwidth value where the number of observations being included in the current estimation becomes large enough to mute the effects of the influential training observation. The original and recomputed bootstrap prediction intervals are provided in figure 5.4.33, and the corresponding coverage values are provided in figure 5.4.34. The results indicate that the bootstrap PIMs were significantly reduced, down to the level of the corresponding analytic PIMs. In addition the excessive variability has been removed. The corresponding coverage values indicate that the modified prediction intervals still contain the measurements to the expected degree.

The MAE results for the LL models are shown in figure 5.4.35. The increase in the MAE values for the data pool was also seen for the KR models and is due to the addition of model bias resulting in a larger error value. For the test data, as the bandwidth increases, the estimate stabilizes and the error becomes relatively constant with increasing bandwidths. The large errors which occur at the smallest bandwidths, for the test data, are due to under-regularization.

To determine the optimal bandwidth for the LL models, consider the variance and squared bias of both the training data and the validation data (figure 5.4.36). As was seen

Table 5.4.6: The number of observations in the data pool for which the LL estimate's bootstrap PIM was >40KLB/HR

Bandwidth	0.005	0.025	0.05	0.06	0.07	0.08	0.09	0.1	0.125	0.15	0.2	0.25	0.5	0.75
# Observations Removed	17	52	77	79	72	63	63	61	54	44	36	30	10	4

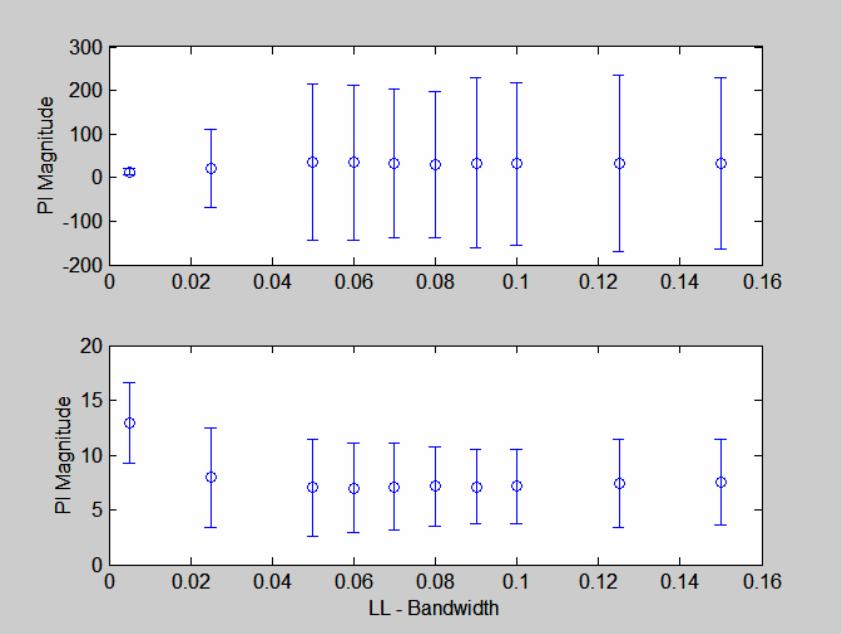


Figure 5.4.33: Original and modified bootstrap prediction intervals of the LL models the for feedwater flow rate data pool.

The overall averages for the original intervals are shown in the upper plot, and the averages over the modified set of observations are shown in the lower plot. The modified set was created by eliminating observations where the PI magnitude was greater than 40 KLB/HR (see table 5.4.6). The bands represent the 1*s* variation in the results over the 100 iterations.

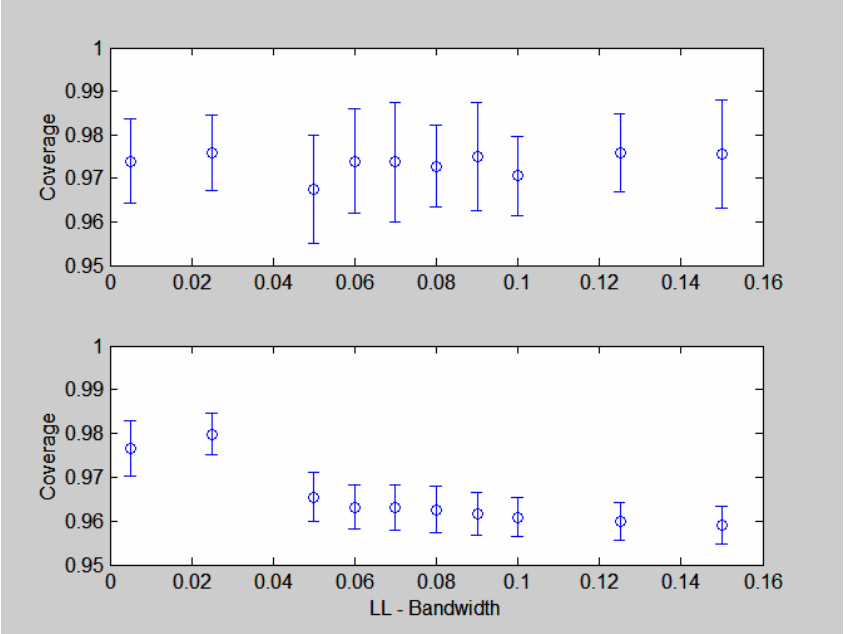
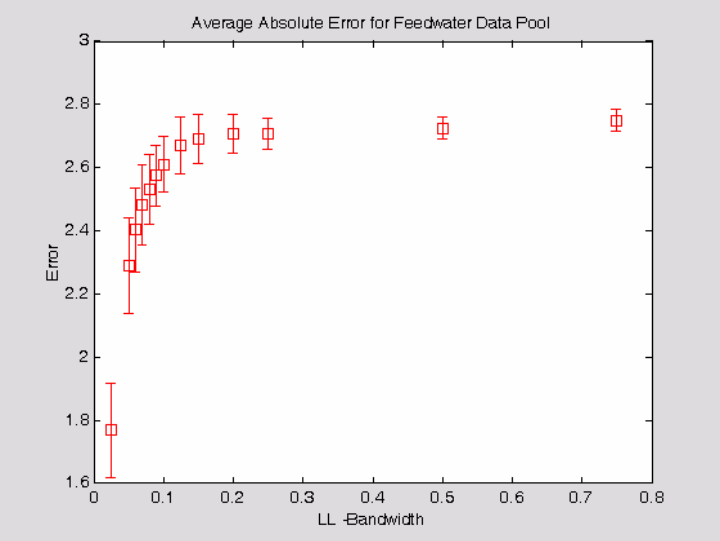
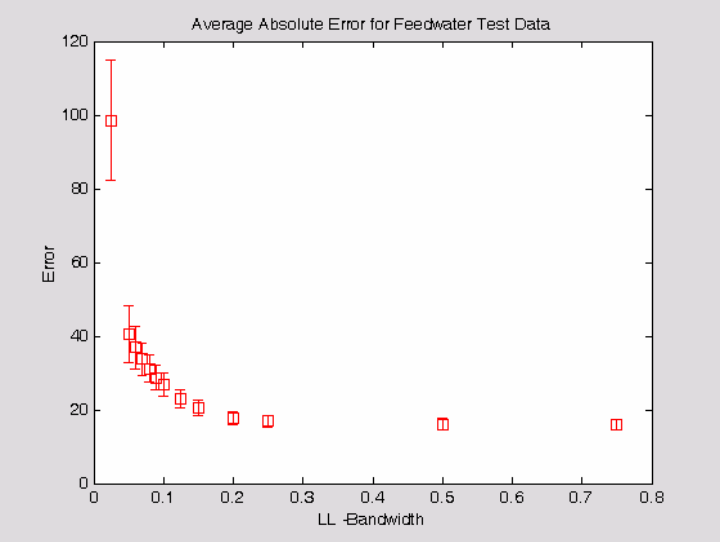


Figure 5.4.34: Original and modified bootstrap prediction interval coverage values for LL models of feedwater flow rate data pool.

The overall averages for the original values are shown in the upper plot, and the averages for the modified set of observations are shown in the lower plot. The modified set was created by eliminating observations where the PI magnitude was greater than 40 KLB/HR (see table 5.4.6). The bands represent the $1s$ variation in the results over the 100 iterations.



a



b

Figure 5.4.35: Mean absolute error results for LL models of feedwater flow rate data. The bands represent the $1s$ variation in the results over the 100 iterations. (a) data pool, (b) test data.

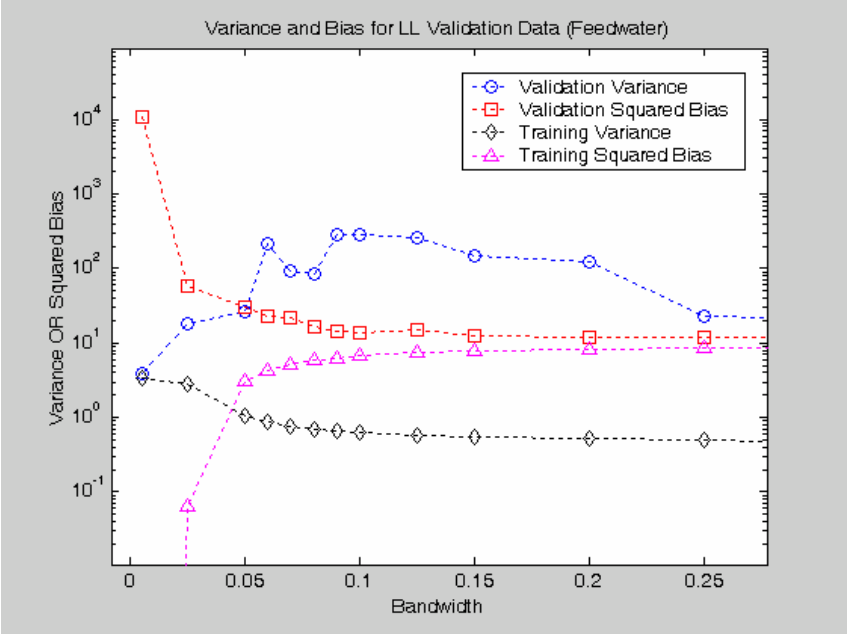


Figure 5.4.36: Variance / squared bias plots for LL models of feedwater flow rate training data and validation data.

for the KR case, the validation data bias is extremely large at low bandwidth values; whereas the training bias approaches zero. This is due to the under-regularized solution which results from a bandwidth value that is too small. The LL models for these small bandwidths are fitting the training vectors almost exactly; however, their abilities to generalize on new data, drawn from the same data pool, are poor. Combining all of the information from both the training and validation data by looking at the combined variance of the training and validation data and the bias of the data pool (figure 5.4.37) it is seen that the optimal bandwidth for the LL models is 0.08.

Based on this optimal bandwidth (0.08) 4 models of the 100 available were chosen at random and their estimations for the test data and corresponding analytic prediction intervals are shown in figure 5.4.38. In cases depicted, the drift is clearly discernable. To provide a detailed illustration of the prediction intervals, the median result based on the median MAE value for all 100 LL models of bandwidth = 0.08 (figure 5.4.39).

A similar analysis to what was performed for the KR models regarding the effect of the conditional variance estimate on the prediction interval computations and corresponding coverage values was performed (figure 5.4.40). As the magnitude of the conditional variance estimate increases, the most noticeable change occurs in the region beyond sample number 500. The interval magnitudes begin to increase in this region due to the variance dominating the analytic prediction interval computation at the bandwidth value of 0.08 (evidenced in figure 5.4.37). As was the case for kernel regression, the sensitivity of the analytic prediction intervals to the estimate of the conditional variance of the response is relatively negligible for the determination of drift in the feedwater flow rate test data. The use of a valid estimate is recommended to ensure optimal performance; however, detailed noise analyses of the response channel are unnecessary for this application.

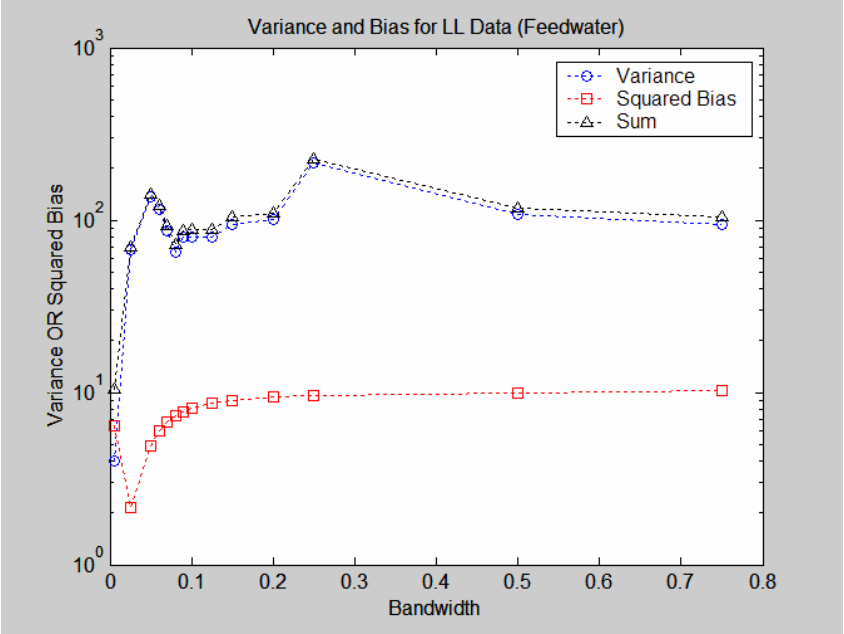


Figure 5.4.37: Combined variance of training and validation data vs. squared bias of data pool for LL models of feedwater flow rate data.

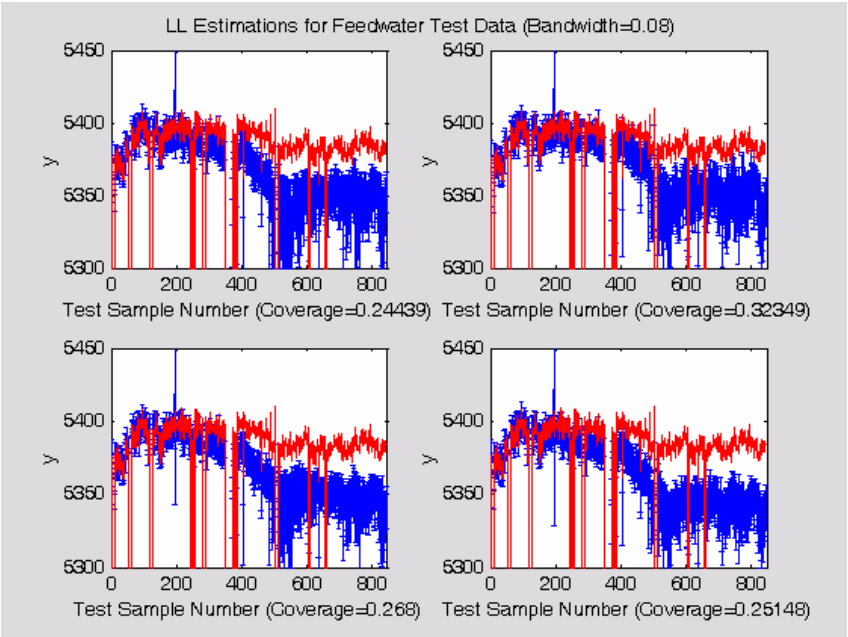


Figure 5.4.38: Test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 available LL models with bandwidth 0.08. The coverage value for each case is indicated on the x-axis.

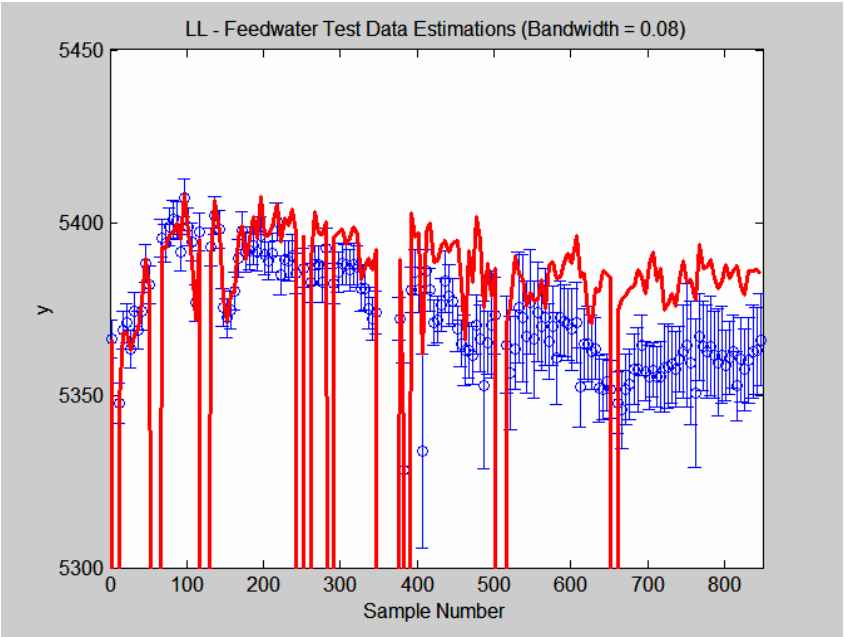


Figure 5.4.39: LL median result estimation and analytic prediction intervals. Representative results for LL model of bandwidth=0.08 showing every 5th estimate and the corresponding analytic prediction intervals. This model exhibited the median value of the 100 Mean Absolute Error results for the LL models with a bandwidth of 0.08

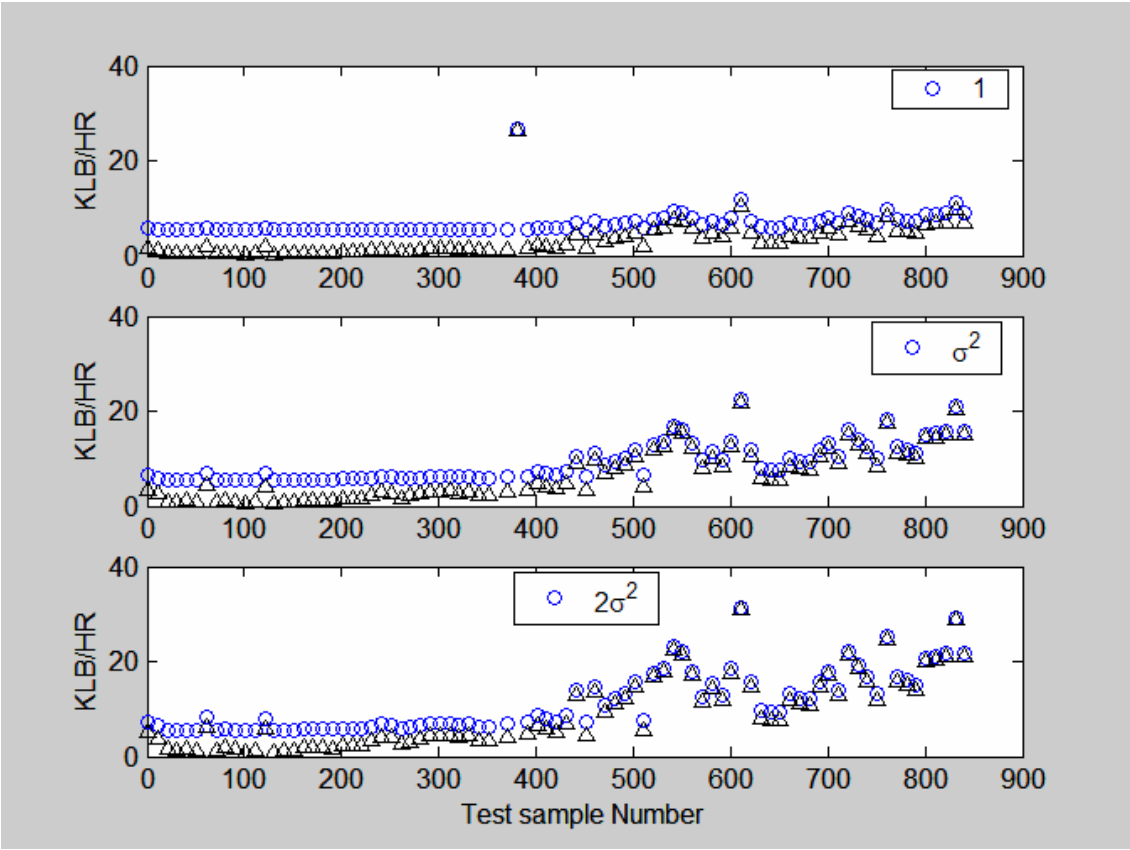


Figure 5.4.40: Analytic prediction interval magnitudes, of feedwater test data LL models, for various conditional variance estimates.

Each subplot contains the prediction intervals with and without the bias component for the KR models. The model bandwidth for the case presented is 0.08. Each subplot provides the results for a different value of the conditional variance estimate: upper – 1, middle - $s^2 = 4.47$, and lower - $2s^2 = 8.947$. Average squared bias value for LL model bandwidth 0.08 is ~7.3[KLB/HR]^2

Considering the distribution of the drift estimates from the 100 LL models of bandwidth 0.08 (figure 5.4.41), 97 out of the 100 models indicated coverage of less than 5% over test samples 700 – 847. This was the highest level of confirmed drift for all empirical models studied.

Overall, the LL models for the feedwater flow data provided analytic prediction intervals slightly higher than those observed for the KR models, though less than those observed for the ANN and NNPLS models. The bootstrap prediction intervals exhibited extreme variation due to the results from a limited number of response estimates. Again, if these estimates are eliminated from the bootstrap computations, the resultant prediction intervals would be reduced and similar to the corresponding analytic prediction intervals. The analytic prediction intervals resulted in coverage values which were slightly lower for all bandwidths evaluated. This is in part due to a less than adequate variance estimate. It is expected that an improved variance estimator, in the prediction interval computation, would increase the resultant coverage values. The error values for the data pool were relatively low with respect to the other models. The variation in the analytic prediction intervals was larger than those observed for the KR models. This is mainly due to the increased range of possible estimates for the LL models with respect to the KR models.

The study of the effect of the conditional variance estimate on the prediction interval magnitudes revealed that their sensitivity to this estimate is not extremely significant, and a reasonable estimate for the conditional variance estimate will suffice.

The drift estimates from the LL models provided the largest average drift result along with a relatively large variation in the estimates, exceeded only by the ANN model results.

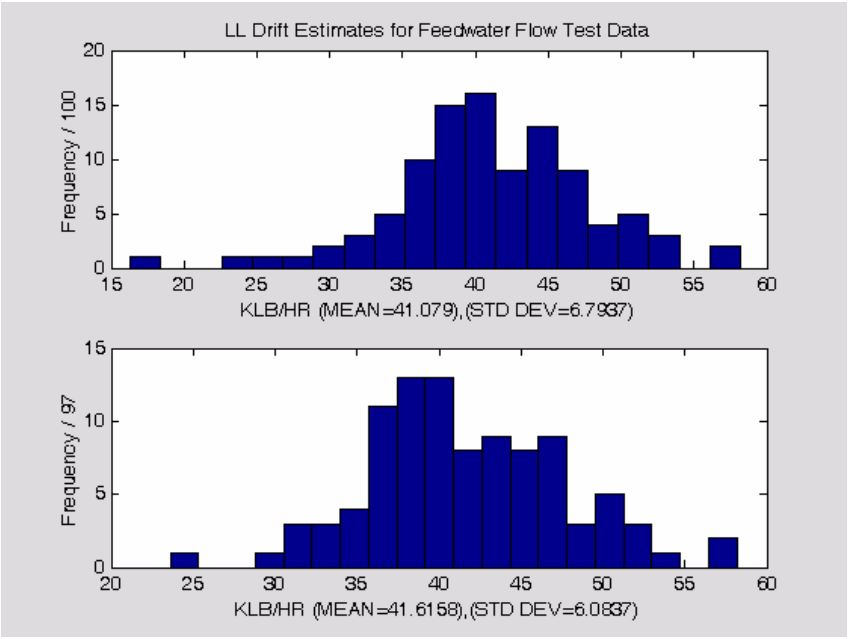


Figure 5.4.41: Distribution of drift estimate for LL models of bandwidth 0.08. Drift estimates obtained from the 100 LL models via computing the MAE over test samples 700 to 847. The upper subplot shows all results, while the lower subplot excludes any results for which coverage was greater than 5%.

5.4.5 Feedwater Flow Rate Data Set Summary of Results

Regarding the average errors with respect to the data pool from the optimal architecture from each model: NNPLS – 2 latent variables, ANN – 2 hidden neurons, KR – h=0.025, LL – h=0.08, the minimum was observed for the KR models, then in increasing order, LL, NNPLS, and ANN. Regarding the analytic prediction intervals from the optimal architectures of each model, the order of minimum to maximum average value is the same as above for the error values. The variation of the prediction interval magnitudes follows an identical trend. Considering the bootstrap prediction intervals from the optimal architecture models, the minimum average occurred for the KR models, while the minimum variation occurred for the NNPLS models. The KR models average analytic prediction interval magnitude was significantly lower (~7KLB/HR) with respect to the NNPLS value (~16KLB/HR). The variation in the average results was for KR, ~5.53KLB/HR, and for NNPLS, ~5.46KLB/HR. The greater average value for the NNPLS models is due to the greater bias resulting from the compression of the information from 17 predictor variables into 2 resultant latent variables. The benefit of this bias introduction is a reduced variability in the estimates, while the drawback is an increased bias reflected in the larger error value for the NNPLS models (MAE=4.9KLB/HR) with respect to the KR models (MAE=2.1KLB/HR). The average bootstrap prediction intervals for the LL models were significantly greater (~31KLB/HR) with a correspondingly large variation (~168KLB/HR). This excessive fluctuation results from the unlimited estimate range of the LL models with respect to the bounded range of the KR models, as discussed in sections 5.4.3 and 5.4.4. The source of this excessive variation was traced back to a limited number of response estimates overall. Removal of these response estimates from the calculations reduced the bootstrap intervals for the LL models with respect to the data pool to a similar level as observed for the analytic intervals, while maintaining the expected level of coverage (0.95). Of course, the ANN bootstrap intervals were extremely high, due to the instability of the ANN models in lieu of the highly collinear data set.

Focusing on the test data results, the analytic prediction interval magnitudes for the NNPLS and KR models exhibited mild increases in average value and variation, while the increases for the LL and ANN models were more substantial. On the other hand, the increases in average value and variation for the bootstrap intervals with respect to the test data exhibited significant increases for all optimal model architectures, with the exception of the NNPLS models for which the change was negligible. Again, this infers the utility of the NNPLS model for collinear data sets, illustrated by the stability of the range of estimates produced, quantified by the bootstrap prediction intervals.

The average coverage values for the data pool were all above the expected level of 95% for the NNPLS models. This again was observed for the ANN models. While the intervals for the ANN models were extremely large in magnitude and exhibited tremendous variation, this is appropriate for the resultant estimates obtained from the ANN models. In other words, the excessive variation in the ANN estimators was properly reflected in the prediction interval computations. For the KR and LL models it was noted that the coverage values for smaller bandwidths were lower than they were at higher bandwidth values. This points to a low variance estimate as the culprit, since lower bandwidth regions are where variance provides its influence. At the optimal bandwidths, the KR models (h=0.025) provided analytic intervals with an average coverage of 86% and the LL models (h=0.08) provided analytic intervals with an average coverage of 92%.

The coverage values of the analytic prediction intervals for the test data were consistently small for the NNPLS models. For the ANN models, these values were moderately increased, and as the complexity of the ANN models increased the ability to discern the drift was reduced. Increased complexity results in increased variance in the estimates, thus this was an expected result. For the KR and LL models, the analytic prediction interval coverage values at the smallest of bandwidths are influenced by numerical effects as discussed in section 5.3. Beyond these small bandwidths, the results for the analytic coverage values with respect to the test data were near expected levels (~30%) for the KR

models near the optimal bandwidth. As the bandwidth increased, the bias introduction led to significant increases in the prediction interval magnitudes, and consequently the coverage values increased. For the KR models where h>0.1, the drift was no longer identifiable. For the LL models, the test data coverage values were consistently small for bandwidths in the range of the optimal bandwidth (0.08). The rate of bias increase for the LL models was evidenced to be much slower than observed for the KR models. Moreover, for the LL models the analytic prediction interval computations were dominated by the variance component, even at the larger bandwidth values of the range of bandwidths evaluated.

Overall, the analytic prediction intervals for the NNPLS models were sufficiently similar to the bootstrap prediction intervals. Noted exceptions were for the 3 and 4 latent variable models with respect to the test data, where the analytic intervals exceed the bootstrap intervals by a large margin. The coverage values for both the bootstrap and analytic intervals were consistently above the expected value of 0.95, and for the test data consistently in the range of 0.3. Thus, the uncertainty of the NNPLS were sufficiently accounted for in the prediction interval computations for the feedwater flow data set.

The ANN bootstrap prediction intervals were significantly greater than the analytic intervals with respect to the test data. Consequently the bootstrap intervals consistently contained the majority of the response measurements for the test data. This was an improper result considering the known drift. The coverage values for the analytic intervals were lower, though did not consistently provide a situation where the drift in the response was clearly discernable. It was noted that the average values for bootstrap prediction intervals became more similar to the average values for the analytic intervals as the model complexity increased. Thus, reflecting the increased contribution to the analytic computations from the additional, unnecessary free parameters. Overall, the ANN models produced poor estimations for the feedwater flow rate data. The instability of the standard ANN design for collinear data sets was clearly demonstrated.

Regarding the data pool, the KR analytic intervals were general lower in average magnitude than the corresponding bootstrap prediction intervals across the range of bandwidths. The coverage values for the analytic intervals remained slightly below the expected value of 0.95, though the deviation decreased at larger bandwidths. Because the variance component decreases as the bandwidth increases, the effect of an insufficient estimate of this variance would be reduced as the bandwidth increases. Thus, the slightly lower than expected values for the coverage of the analytic prediction intervals is attributed to an insufficient variance estimate. Introducing a greater variance influence into the analytic prediction interval computation should effectively compensate for the observed differences between the interval estimates and produce coverage values at the expected level.

The above discussions regarding the KR model results with respect to the data pool also applies to the test data results. An additional observation was noted for the test data that large bandwidth models are unable to identify the drift in the response. Overall, the KR models for the feedwater flow rate data performed adequately, resulting in the minimum average error with respect to the data pool when the optimal architecture was used (h=0.025). The coverage of the analytic prediction intervals of the data pool measurements at the optimal bandwidth was 86%. While, the discussed lower than average coverage value results require that modifications be made to the variance computation, the resultant models performed consistently. An implementation that may improve the resultant coverage results is that of a density estimator. The KR models were seen to be relatively unstable for estimations based on query observations with little representation in the training data set. The implementation of an estimate of the density of the training data at the current query point should reduce these effects. Query observations in regions of correspondingly high density regions of the training data set should have corresponding estimates of higher certainty than estimates from observations occurring in regions of the training data set where the density of observations is low. In addition, there were many observed estimates at or near the boundaries of the training set response. The boundary effects of KR models have been well documented, and the

inclusion of a measure of distance from the boundary may also produce more adequate interval estimates.

Considering the results for the LL models with respect to the data pool, similar observations as to those reported for the KR models are again seen. The bootstrap intervals were consistently higher than the analytic prediction intervals. As the bandwidth increased and the variance contribution decreased, the deviation between the results from the analytic and bootstrap prediction intervals was reduced. Again, the proposed assumption is that an improved estimate of the variance of the estimator will lessen the noted deviations. The overall MAE values were consistently the lowest of all models with respect to the data pool, with a couple of exceptions at small bandwidths. The coverage values for the analytic intervals were slightly lower than the expected value, and at the optimal bandwidth the analytic prediction interval coverage value was 92%. As the bandwidth increased, the coverage values moved closer to the expected value.

One of the major observed differences between the KR and LL model results is that the region of optimal bandwidth for KR is much more narrow than that for the LL models. This is a result of the much more rapid bias increase exhibited by the KR models with respect to the LL models. This is a logical result considering that over-regularized KR models fit a constant; whereas, over-regularized LL models fit a line. Assuming the functional form is more complex than a line, of the two, the overregularized LL model would be less biased.

Considering the LL results for the test data, the excessively large bootstrap prediction intervals was traced back to a limited set of points within the response test data set. Reducing the influence of these observations results in much more reasonable prediction intervals. Thus, the bootstrap intervals are reflecting the huge variation that exists at a select few points in the response data.

The study of the effect of the conditional variance estimate on the computed prediction interval magnitudes revealed that the interval computations were not overly sensitive to this estimate. While it is important to produce a fairly accurate estimate, the described methods will not fail as long as a reasonable estimate is made.

The drift estimates for the feedwater flow test data are summarized below in table 5.4.7. The results indicate that the most stable drift estimates were from the 2 latent variable NNPLS models. This illustrates the utility of the NNPLS algorithm for highly collinear data sets. Along the same lines, due to the collinear data, the ANN model drift estimates were extremely variable. Due to the consistent coverage of the analytic prediction intervals from the NNPLS models with respect to the data pool to the expected level, the drift estimate produced is assumed to be the most *certain* estimate. While the KR and LL models produced higher estimates, the coverage of their corresponding prediction intervals with respect to the data pool were slightly below the expected level, thus the estimates from these models is assumed to be less certain than those from the NNPLS models. Table 5.4.7 also includes average drift estimates from a reduced set of models, where the reduction was performed by removing results corresponding to models where the coverage value was $\geq$5%. This was done to remove the influence of poor models for the task at hand of estimating the drift. For the most part the changes in the average drift were small. The only major change was incurred for the ANN models which in some cases produced extremely high drift estimates.

5.5 TURBINE PRESSURE DATA SET

In this section, an initial description of the data and model parameters is given followed by tabulated and plotted results. For this data set, the discussion of the results is done on a model type basis, each model being discussed in a separate section. A summary at the end (section 5.5.5) draws comparisons between the results from the different model types. Similar to the feedwater flow data set, the turbine pressure data set response variable contained a known drift. Quantification of the drift estimates of the different

Table 5.4.7: Summary of Feedwater Flow Rate Drift Estimation

Model Type	All Estimates		# of Results where Coverage $\leq$ 5%	Reduced # of Estimates	
	m (KLB/HR)	s (KLB/HR)		m (KLB/HR)	s (KLB/HR)
NNPLS (2)	37.6	3.0	90	37.7	2.9
ANN (2)	33.7	62.4	78	28.9	6.5
KR (0.025)	36.0	4.1	81	36.7	3.7
LL (0.08)	41.1	6.8	97	41.6	6.1

* Numbers in parentheses under model type indicate the number of latent variables (NNPLS), the number of hidden neurons (ANN), or the bandwidth (KR and LL).

Coverage $\leq$5% based on test samples 700-847.

models was also completed, and the average results along with the variation of these results are reported.

The turbine pressure data set contains 5 predictor variables: 3 steam generator steam pressures, a turbine first stage pressure, and the unit gross generation. The response variable is also a turbine first stage pressure channel, though not the same channel as the one included in the predictor variable set. The data was provided by the Electric Power Research Institute (EPRI) and is from an operating U.S nuclear power plant. One-minute sampled data were available from 3/2001 to 1/2002. Similar to the feedwater flow data set, the response variable contains a known drift. The data pool was constructed using data drawn from the period of operation covering 3/2001 – 6/2001. Test data cover the entire period from 3/2001-1/2002. The turbine pressure data pool with respect to the response variable is shown in figure 5.5.1. When the turbine pressure channel is mentioned throughout these discussions, it is the *response* turbine first stage pressure to which is being referred and not the turbine first stage pressure channel of the predictor variable set. A distinction will be made if referring to the turbine first stage pressure channel of the predictor variable set.

The data pool $(\mathbf{X}_{pool}, \mathbf{Y}_{pool})$ of dimensions $(n_{pool} \times p)$ and $(n_{pool} \times 1)$ respectively were specified by: $n_{pool} = 1675$, $p = 5$. The observations in the data pool cover the first 4 months of the period from 3/2001 – 1/2002. The 100 training data sets: $(\mathbf{X}^b{}_{tr}, \mathbf{Y}^b{}_{tr})$, of dimensions $(n_{tr} \times p)$, and $(n_{tr} \times 1)$ respectively were specified by: $n_{tr} = 800$, $p = 5$. The training observations were drawn at random, with replacement, from the data pool. In all cases, $b = 1,..., B$ and $B = 100$. The 100 validation data sets $(\mathbf{X}^b{}_v, \mathbf{Y}^b{}_v)$ of dimensions $(n_v \times p)$ and $(n_v \times 1)$ respectively were specified by: $n_v = 300$, $p = 5$. The validation observations were drawn at random, with replacement, from the data pool. The test data set $(\mathbf{X}_{ts}, \mathbf{Y}_{ts})$ of dimensions $(n_{ts} \times p)$ and $(n_{ts} \times 1)$ respectively was specified by: $n_{ts} = 1432$, $p = 5$. The test data cover approximately 11 months of plant operation (300 minutes between samples).

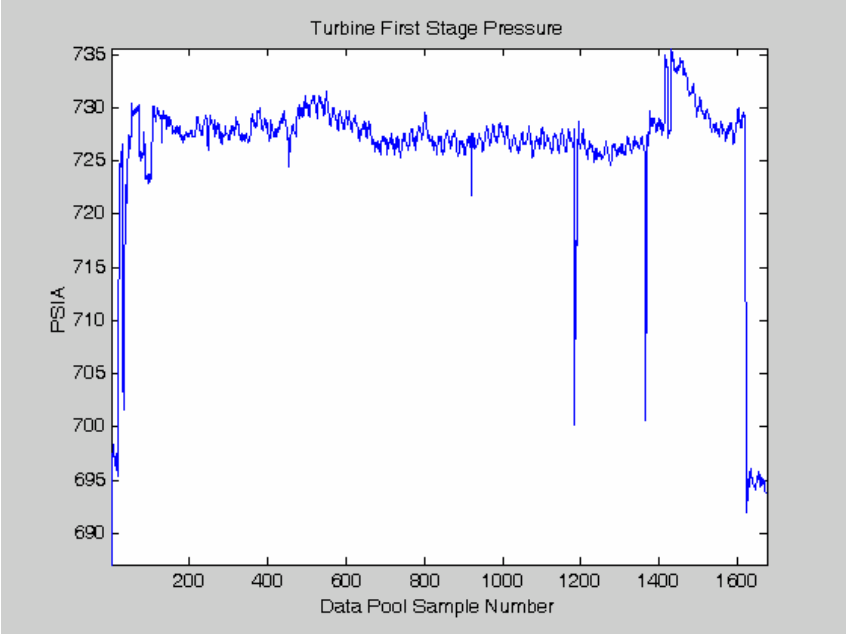


Figure 5.5.1: Turbine first stage pressure response data pool.

The required model specifications used for the feedwater flow rate data are provided below:

NNPLS:

Mean-standard initialization was applied. The maximum number of latent variables to evaluate was set at 5.

ANN:

Mean-standard initialization was applied. The minimum and maximum numbers of hidden neurons evaluated were 1 and 4, respectively.

LPR:

The vector of bandwidths to be evaluated was:

[0.005 0.025 0.05 0.06 0.07 0.08 0.09 0.1 0.125 0.15 0.2 0.25 0.5 0.75];

The conditional variance was estimated as $s_y^2 = 4.32$.

The overall results for the turbine pressure data pool are presented in table 5.5.1, and for the test data in table 5.5.2. All of the following discussions for the turbine pressure data set results refer to the values in these two tables. The 4 sections that follow present the results for each model type and a summary at the end (section 5.5.5) draws comparisons between the results for the different models.

5.5.1 Neural Network Partial Least Squares Model Results and Analyses

The prediction interval magnitudes and coverage values for the NNPLS models of the turbine pressure data are provided in figure 5.5.2. The PIMs for the data pool follow the expected trends for both the bootstrap and the analytic approaches. The significant increases in magnitude and variation for the NNPLS model's test data PIMs, for the 4 and 5 latent variable models (figure 5.5.2.b), are due to the increased complexity of the model as the number of latent variables increases. The data pool coverage values also indicate the expected behavior. The analytic coverage values, with respect to the test data, for the 1 latent variable model show extreme variation due to the PIMs being very close to the drift level in this case. The average PIM for the 1 latent variable models for the test data is ~9PSIA, and the drift level in the test data is ~10PSIA. Thus, slight variability in the

Table 5.5.1: Tabulated results for turbine pressure data pool*.

(For full explanation of terms see section 5.2)

Model	Type	API		BPI		AC		BC		MAE	MSE
		m	s	m	s	m	s	m	s		
NNPLS	1	7.982	4.261	10.936	4.249	0.953	0.016	0.984	0.017	1.981	22.044
NNPLS	2	4.136	1.417	6.212	2.837	0.961	0.014	0.991	0.014	1.293	6.531
NNPLS	3	3.698	1.284	5.791	2.821	0.966	0.014	0.991	0.013	1.150	5.610
NNPLS	4	3.663	1.289	5.977	3.003	0.966	0.014	0.992	0.012	1.132	5.993
NNPLS	5	3.648	1.283	5.953	3.000	0.967	0.013	0.992	0.013	1.120	5.949
ANN	1	1.274	0.024	1.290	0.011	0.953	0.005	0.956	0.003	0.516	0.411
ANN	2	1.172	0.049	1.248	0.138	0.960	0.006	0.970	0.006	0.474	0.354
ANN	3	1.103	0.070	1.206	0.180	0.957	0.007	0.972	0.006	0.441	0.320
ANN	4	1.053	0.075	1.185	0.218	0.953	0.007	0.972	0.007	0.417	0.301
ANN	5	1.005	0.092	1.141	0.225	0.952	0.008	0.974	0.009	0.393	0.276
KR	0.005	3.871	0.021	0.905	0.233	1.000	0.000	0.989	0.003	0.186	0.104
KR	0.025	3.704	0.022	0.964	0.328	0.999	0.001	0.979	0.004	0.208	0.124
KR	0.050	3.153	0.023	0.905	0.432	0.996	0.002	0.975	0.005	0.212	0.114
KR	0.060	2.856	0.022	0.889	0.551	0.994	0.002	0.973	0.005	0.219	0.112
KR	0.070	2.561	0.019	0.887	0.687	0.993	0.002	0.969	0.005	0.227	0.112
KR	0.080	2.294	0.017	0.890	0.805	0.993	0.002	0.966	0.005	0.236	0.113
KR	0.090	2.066	0.014	0.891	0.859	0.991	0.002	0.963	0.006	0.246	0.117
KR	0.100	1.879	0.013	0.902	0.896	0.989	0.002	0.959	0.005	0.257	0.122
KR	0.125	1.565	0.013	0.951	0.992	0.983	0.003	0.955	0.006	0.287	0.142
KR	0.150	1.405	0.015	1.008	0.977	0.974	0.004	0.952	0.006	0.321	0.172
KR	0.200	1.327	0.022	1.169	0.957	0.954	0.005	0.948	0.006	0.395	0.255
KR	0.250	1.387	0.026	1.335	0.922	0.937	0.007	0.939	0.007	0.465	0.356
KR	0.500	1.764	0.046	1.819	0.856	0.919	0.008	0.926	0.004	0.669	0.732
KR	0.750	1.973	0.052	2.036	0.764	0.930	0.008	0.935	0.004	0.779	0.948
LL	0.005	8.052	0.405	20.486	1.364	0.678	0.015	0.998	0.001	5.893	78.654
LL	0.025	9.338	1.419	25.194	24.127	0.843	0.009	0.978	0.007	3.243	74.839
LL	0.050	6.974	0.543	13.979	32.561	0.954	0.005	0.988	0.004	0.865	12.681
LL	0.060	6.073	0.428	11.657	34.991	0.964	0.004	0.989	0.004	0.619	6.538
LL	0.070	5.394	0.402	9.983	41.860	0.971	0.004	0.989	0.004	0.431	2.245
LL	0.080	4.830	0.309	9.174	47.449	0.977	0.003	0.988	0.004	0.338	0.795
LL	0.090	4.382	0.264	8.331	48.449	0.981	0.003	0.985	0.004	0.292	0.338
LL	0.100	4.035	0.423	8.491	51.691	0.983	0.002	0.978	0.005	0.272	0.210
LL	0.125	3.360	0.440	8.028	53.157	0.986	0.002	0.967	0.006	0.262	0.148
LL	0.150	2.844	0.263	7.094	50.080	0.986	0.002	0.960	0.006	0.266	0.139
LL	0.200	2.260	0.159	5.872	45.625	0.983	0.003	0.951	0.005	0.282	0.142
LL	0.250	1.943	0.112	4.640	39.263	0.979	0.003	0.948	0.004	0.296	0.149
LL	0.500	1.513	0.803	2.093	22.750	0.966	0.004	0.942	0.004	0.339	0.180
LL	0.750	1.264	0.065	1.321	15.138	0.960	0.005	0.941	0.004	0.372	0.211

* API – Analytic prediction intervals, BPI – Bootstrap prediction intervals, m - mean result, s - standard deviation of results, MAE – Mean Absolute Error, MSE – Mean Squared Error, Type for NNPLS refers to the number of latent variables, for ANNs refers to the number of hidden neurons, and for KR and LL refers to the bandwidth parameter. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

Table 5.5.2: Tabulated results for turbine pressure test data.*
(For full explanation of terms see section 5.2)

Model	Type	API		BPI		AC		BC		MAE	MSE
		m	s	m	s	m	s	m	s		
NNPLS	1	8.784	4.338	10.964	4.201	0.577	0.270	0.629	0.073	7.402	84.076
NNPLS	2	6.048	3.696	6.749	3.724	0.360	0.041	0.431	0.033	7.213	77.208
NNPLS	3	9.307	13.966	6.752	3.877	0.397	0.070	0.453	0.044	7.177	77.003
NNPLS	4	16.265	30.919	16.820	12.901	0.495	0.130	0.818	0.058	7.556	156.079
NNPLS	5	35.681	72.374	17.490	13.693	0.620	0.168	0.830	0.060	7.642	163.973
ANN	1	1.284	0.024	1.298	0.013	0.322	0.004	0.323	0.003	5.489	46.979
ANN	2	1.201	0.050	1.443	0.457	0.318	0.011	0.331	0.011	5.613	49.514
ANN	3	1.193	0.106	1.421	0.424	0.321	0.010	0.333	0.009	5.630	50.281
ANN	4	1.193	0.097	1.595	0.521	0.321	0.012	0.340	0.011	5.754	52.675
ANN	5	1.200	0.117	1.549	0.496	0.323	0.012	0.344	0.012	5.720	52.305
KR	0.005	3.866	0.033	0.927	0.219	0.994	0.002	0.991	0.002	0.334	0.471
KR	0.025	3.760	0.062	1.036	0.469	0.392	0.006	0.392	0.003	6.347	61.160
KR	0.050	3.412	0.060	1.046	0.717	0.336	0.007	0.356	0.004	6.440	61.893
KR	0.060	3.250	0.061	1.357	1.871	0.329	0.005	0.347	0.008	6.460	62.216
KR	0.070	3.088	0.061	1.418	2.101	0.326	0.004	0.343	0.008	6.463	62.275
KR	0.080	2.928	0.063	1.426	2.143	0.324	0.004	0.341	0.008	6.463	62.282
KR	0.090	2.772	0.064	1.414	2.136	0.322	0.004	0.341	0.008	6.464	62.288
KR	0.100	2.621	0.066	1.415	2.145	0.321	0.004	0.340	0.008	6.466	62.295
KR	0.125	2.270	0.064	1.444	2.174	0.316	0.004	0.338	0.008	6.473	62.341
KR	0.150	1.978	0.056	1.480	2.168	0.312	0.003	0.336	0.008	6.483	62.402
KR	0.200	1.633	0.038	1.586	2.095	0.307	0.004	0.335	0.007	6.511	62.596
KR	0.250	1.546	0.032	1.719	2.015	0.304	0.005	0.333	0.007	6.546	62.912
KR	0.500	1.810	0.047	2.098	1.571	0.297	0.007	0.323	0.009	6.719	64.783
KR	0.750	1.985	0.053	2.180	1.030	0.295	0.004	0.305	0.007	6.864	66.524
LL	0.005	7.463	0.441	19.147	0.912	0.858	0.010	0.998	0.001	8.887	105.183
LL	0.025	8.777	3.050	41.330	22.913	0.354	0.010	0.490	0.011	89.611	15527.01
LL	0.050	34.575	10.959	62.279	68.884	0.716	0.035	0.793	0.021	51.663	12154.75
LL	0.060	36.916	9.652	44.874	65.273	0.829	0.024	0.713	0.016	35.733	9445.950
LL	0.070	33.486	6.588	29.133	57.744	0.873	0.016	0.598	0.014	27.743	7676.229
LL	0.080	28.927	4.420	22.151	55.031	0.879	0.014	0.524	0.014	24.844	6933.865
LL	0.090	24.846	3.246	19.212	55.222	0.870	0.015	0.479	0.016	23.412	6435.847
LL	0.100	21.449	2.531	19.038	58.911	0.851	0.017	0.445	0.018	22.238	5875.719
LL	0.125	15.646	1.985	20.390	68.006	0.784	0.022	0.396	0.019	19.113	4376.451
LL	0.150	11.952	1.820	20.064	69.297	0.706	0.023	0.368	0.016	16.074	3067.412
LL	0.200	7.979	1.335	15.459	58.257	0.592	0.018	0.355	0.010	9.429	715.011
LL	0.250	6.143	0.918	8.291	38.624	0.519	0.016	0.352	0.007	6.106	68.741
LL	0.500	2.802	0.684	2.139	17.460	0.360	0.004	0.314	0.005	5.776	52.131
LL	0.750	1.894	0.146	0.997	0.311	0.346	0.003	0.304	0.002	5.694	50.373

* API – Analytic prediction intervals, BPI – Bootstrap prediction intervals, m - mean result, s - standard deviation of results, MAE – Mean Absolute Error, MSE – Mean Squared Error, Type for NNPLS refers to the number of latent variables, for ANNs refers to the number of hidden neurons, and for KR and LL refers to the bandwidth parameter. NNPLS – Neural Network Partial Least Squares, ANN – Artificial Neural Network, KR – Kernel Regression, LL – Local Linear Regression.

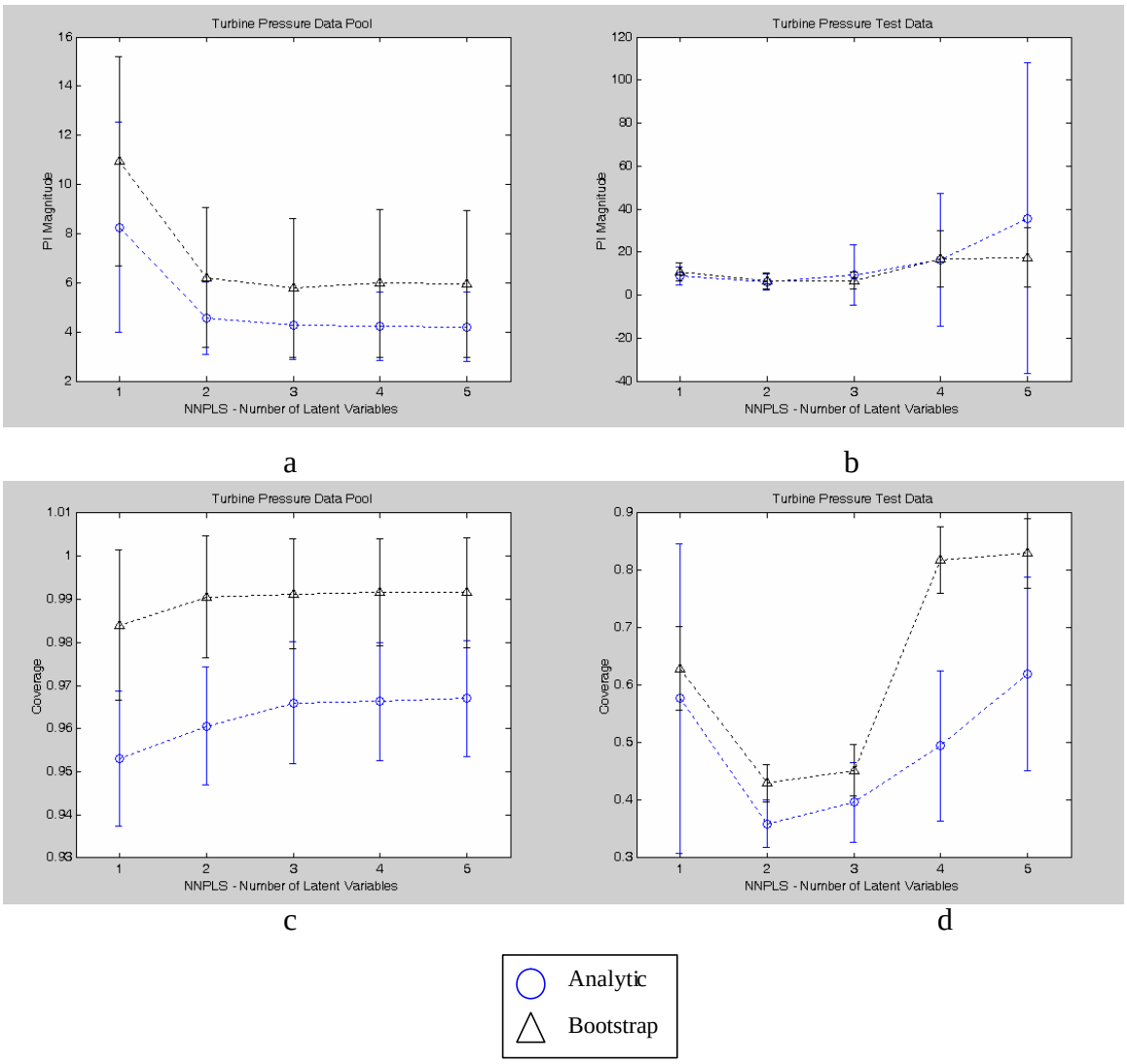


Figure 5.5.2: PI magnitude and coverage results for NNPLS models of turbine pressure data pool. The bands represent the 1*s* variation in the results over the 100 iterations.

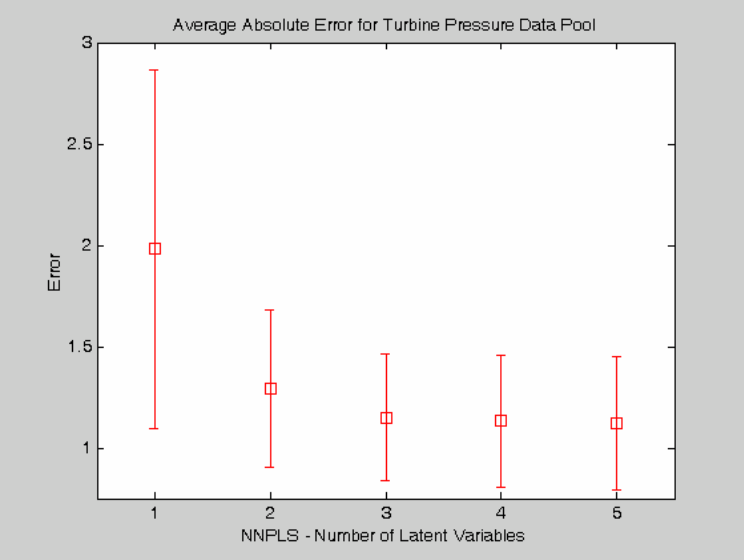
estimates will result in large changes in the coverage values. The large variation in the analytic PIM for the 5 latent variable models was similar to what was observed for the feedwater flow data. Increases in model complexity that provide no benefits with respect to model performance can only reduce the stability of the solution. The remainder of the PIMs and coverage value results were consistent with what was observed for the previous data sets.

The error values (figure 5.5.3) correspond well with the PIM results, and indicate that the 2 latent variable models provide the best fit over all architectures evaluated. A sampling of estimations from randomly selected 2 latent variable models is provided in figure 5.5.4. The highly varying prediction intervals indicate that the NNPLS models of the turbine pressure channel are highly unstable. The turbine pressure data is not well suited for the NNPLS architecture because it does not exhibit high linear correlations between the predictor variables and the response.

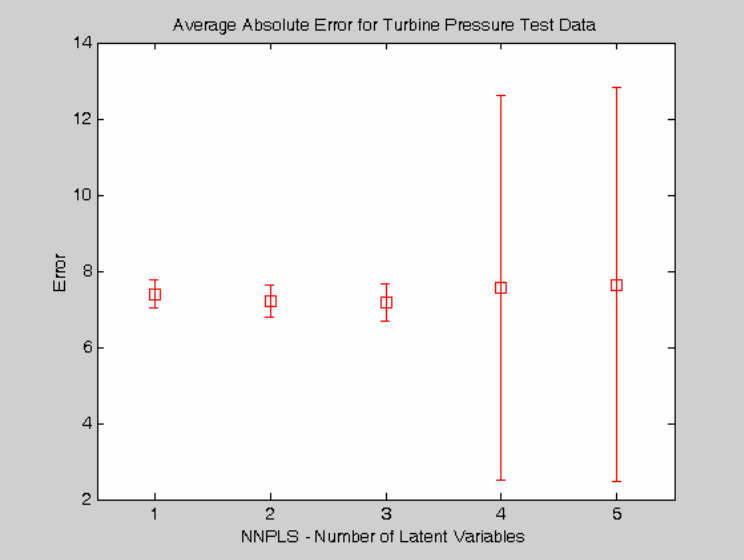
There are a significant number of cases for which the analytic PIMs become excessively large in localized areas. Consider figure 5.5.5 which shows a 2 latent variable model with excessively large prediction intervals over a localized region. On the other hand, figure 5.5.6 shows a better result where the prediction intervals are consistent over the entire response variable.

Consider the distribution of the magnitude of the drift estimate over all 100 models (figure 5.5.7), and for only those models which provided prediction intervals which exhibited a coverage value of $\leq$5% over test samples 600 to 1430. The drift estimate was fairly stable due to the use of the MAE as its estimator; however, the prediction intervals indicate that the NNPLS architecture is not well suited for the turbine pressure data set.

Overall, the NNPLS prediction intervals reflect the high uncertainty associated with the NNPLS models for the given data set. The coverage values for the computed prediction intervals were consistently greater than or equal to the expected level for all model



a



b

Figure 5.5.3: Mean absolute error results for NNPLS models of turbine pressure data. The bands represent the 1*s* variation in the results over the 100 iterations. (a) – data pool, (b) – test data

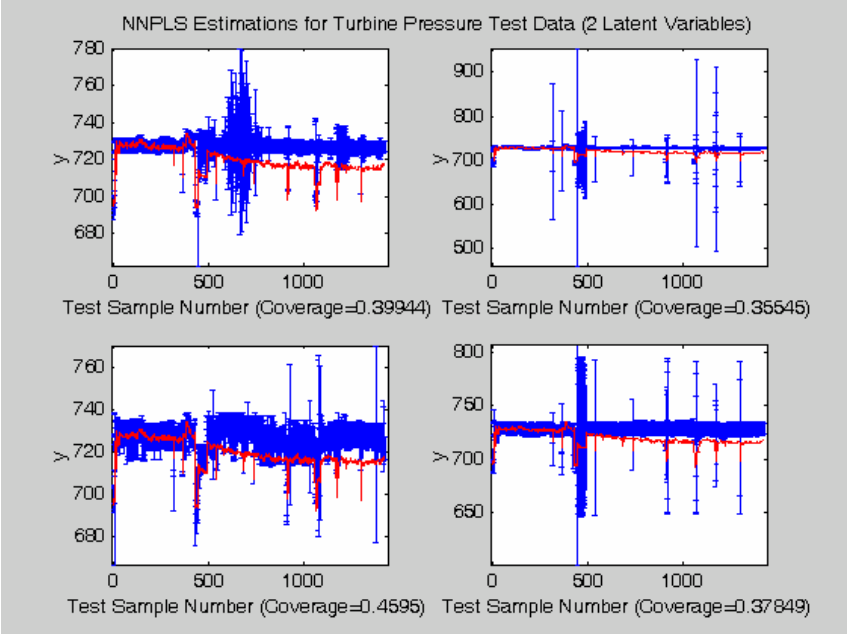


Figure 5.5.4: NNPLS test estimations and their corresponding analytic prediction intervals.

These 4 results were selected at random from the 100 available 2 latent variable NNPLS models. The coverage value for each case is indicated on the x-axis.

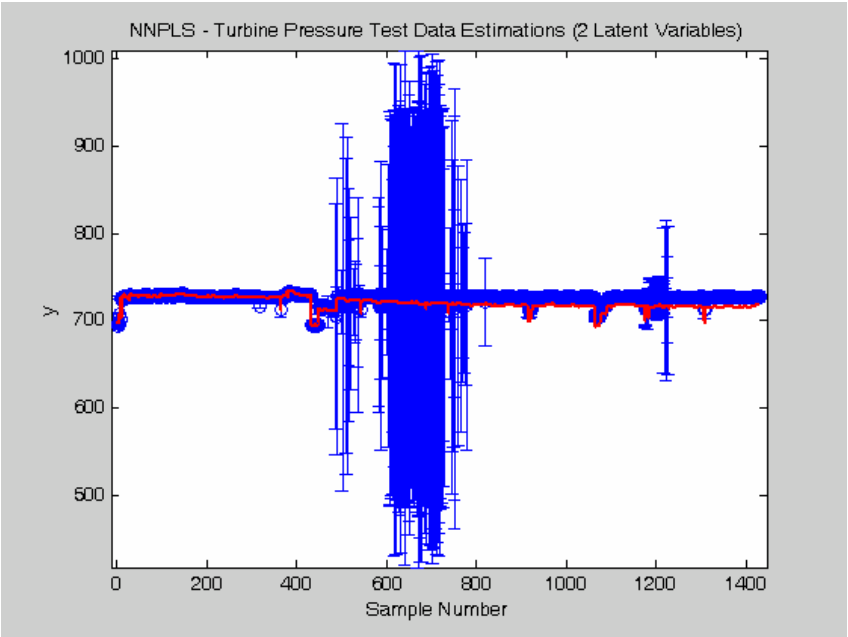


Figure 5.5.5: NNPLS median result estimation and analytic prediction intervals. Representative result of NNPLS 2 latent variable models. This model exhibited the median value of the 100 Mean Absolute Error results for the 2 latent variable models.

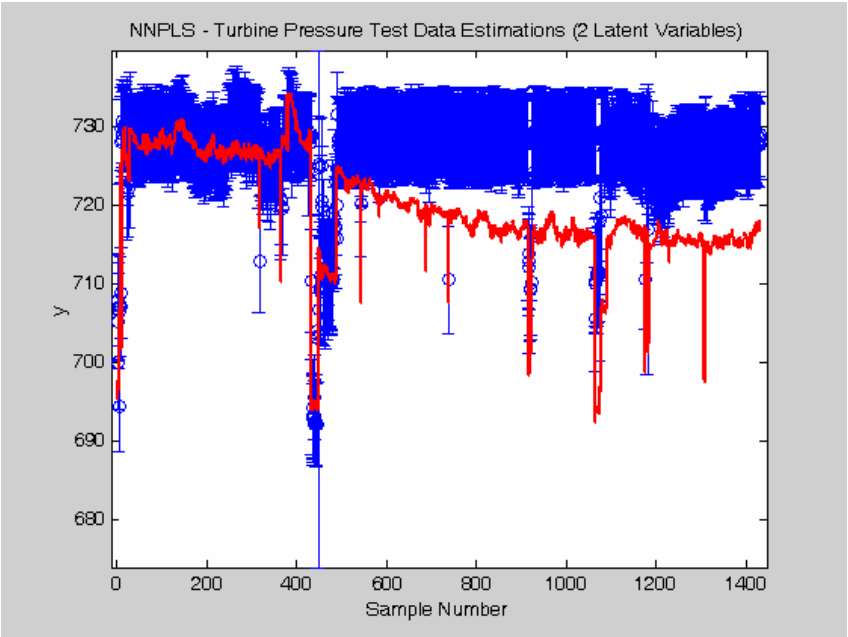


Figure 5.5.6: 2 latent variable NNPLS model with good behavior for turbine pressure test data.

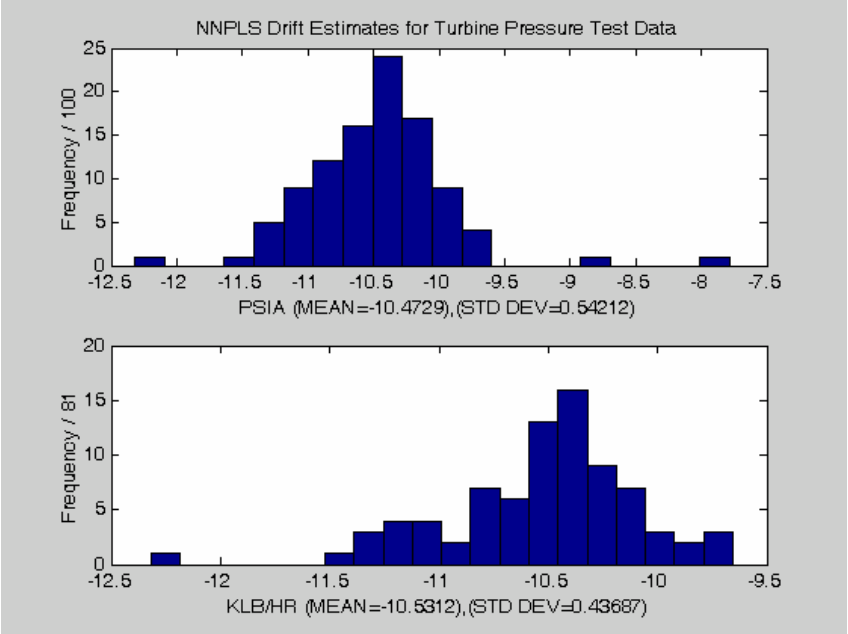


Figure 5.5.7: Distribution of drift estimate for 2 latent variable NNPLS models. Drift estimates obtained from the 100 NNPLS models via computing the MAE over test samples 600 to 1430. The upper subplot shows all results, while the lower subplot excludes any results for which coverage was greater than 5%.

architectures with respect to the data pool. Coverage values for the test data were higher than expected considering the known drift, though accurately reflecting the uncertainty of the NNPLS models for this data set.

5.5.2 Artificial Neural Network Model Results and Analyses

The PIMs and coverage values for the ANN models of the turbine pressure data set are shown in figure 5.5.8. The first observation is that the magnitudes of the PIMs are much lower than that from the NNPLS models. The slightly higher magnitudes for the bootstrap PIMs is similar to what has been seen for both of the previous data sets with respect to ANN models. The average coverage values for the data pool are all above the expected value of 0.95. The test data coverage values indicate the presence of drift in the response data via their low values.

The MAE values (figure 5.5.9) show that the errors for the data pool were consistently low, and the larger values for the test data are due to the deviation of the estimates from the measurements which include the drift.

The 2 hidden neuron architecture was selected as optimal for this data set. This decision was based on the results with respect to the data pool only. Test data results should not influence optimal model determination. Of the 100 2 hidden neuron models for the turbine pressure data, 4 were selected at random and their estimations and analytic prediction intervals were plotted (figure 5.5.10). In each of the 4 plots the drift is clearly identifiable and the prediction intervals are consistent over the entire test response. A representative result is shown in greater detail in figure 5.5.11.

Finally, the distribution of the resultant drift estimates is shown in figure 5.5.12. In this case, all models exhibited analytic prediction intervals that covered the response variable measurements, between test samples 600 to 1430, to a level of $\leq$5%. The drift estimation distribution shows that the ANN models were able to consistently identify the drift in the turbine pressure response, and the variability in the estimates was sufficiently small.

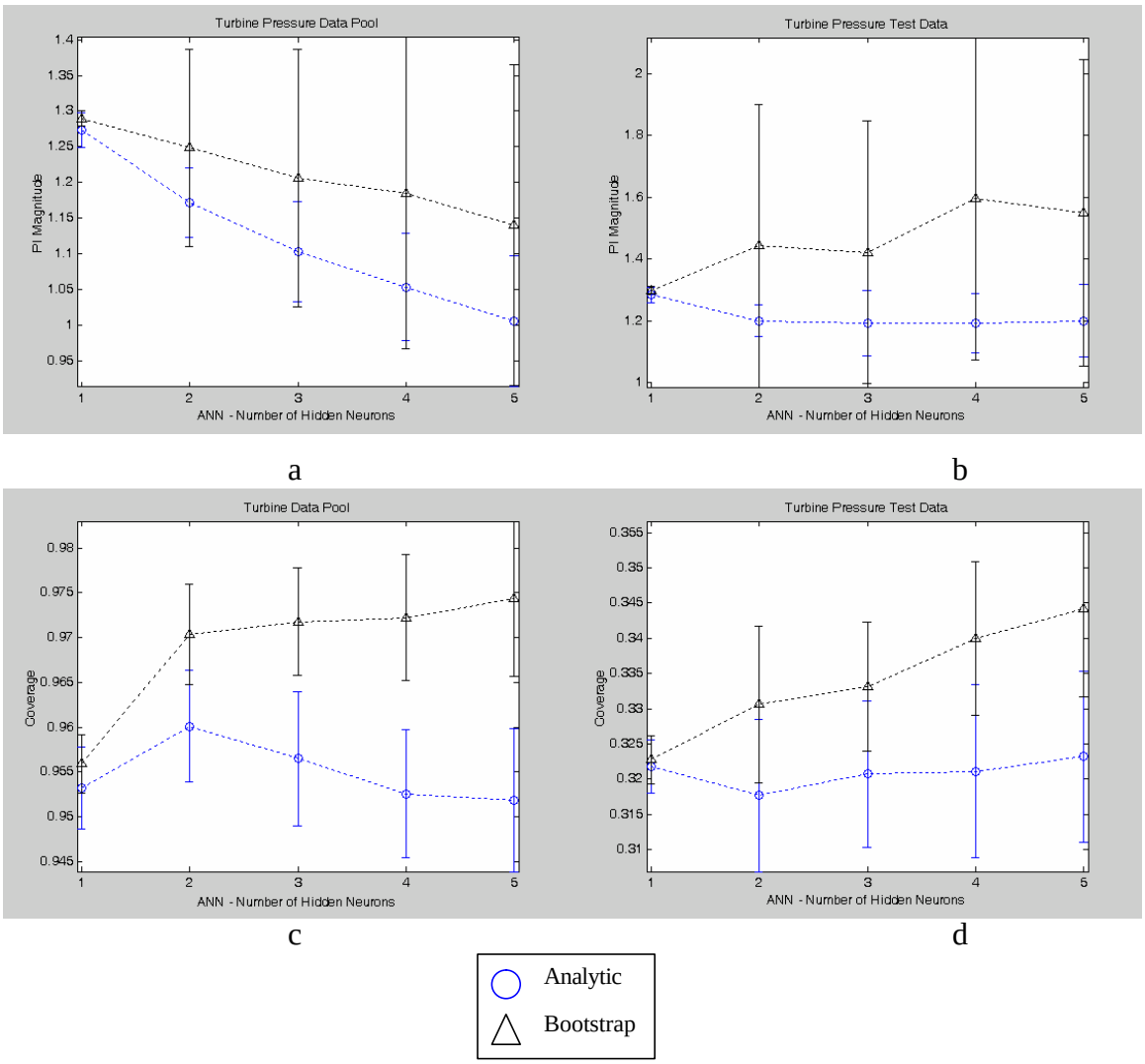
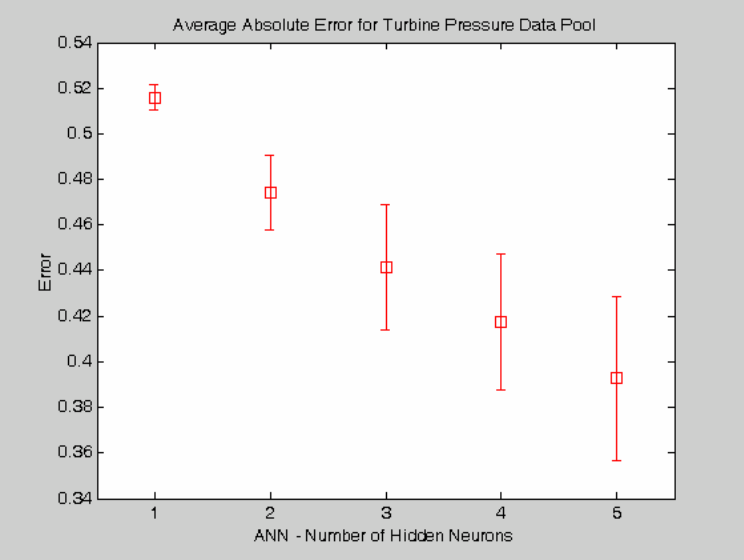
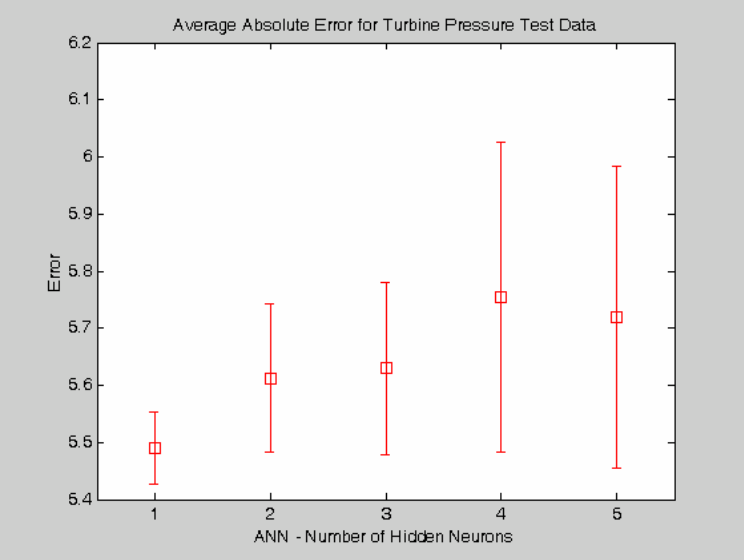


Figure 5.5.8: PI magnitude and coverage results for ANN models of turbine pressure data pool. The bands represent the 1*s* variation in the results over the 100 iterations.



a



b

Figure 5.5.9: Mean absolute error results for ANN models of turbine pressure data. The bands represent the $1s$ variation in the results over the 100 iterations. (a) – data pool, (b) – test data

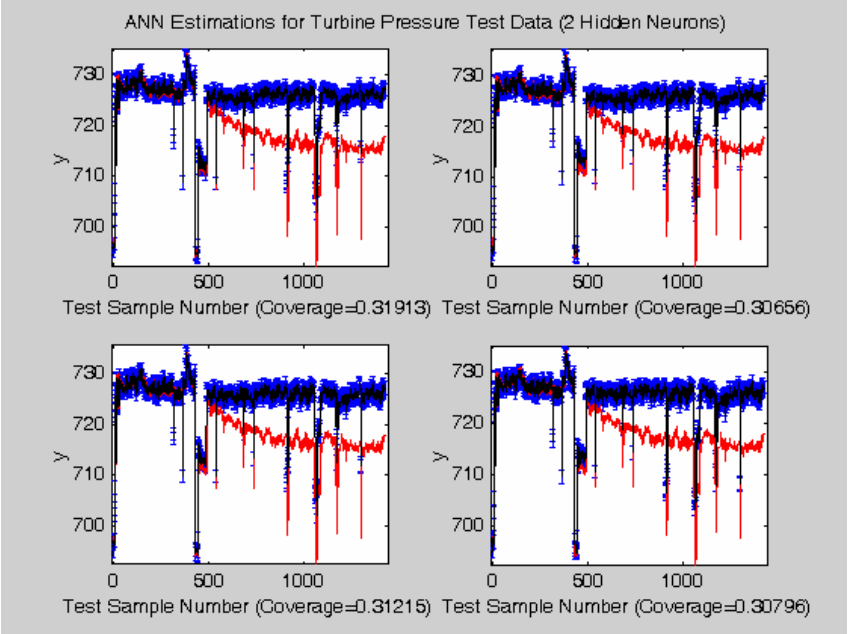


Figure 5.5.10: ANN test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 available 2 hidden neuron ANN models. The coverage value for each case is indicated on the x-axis.

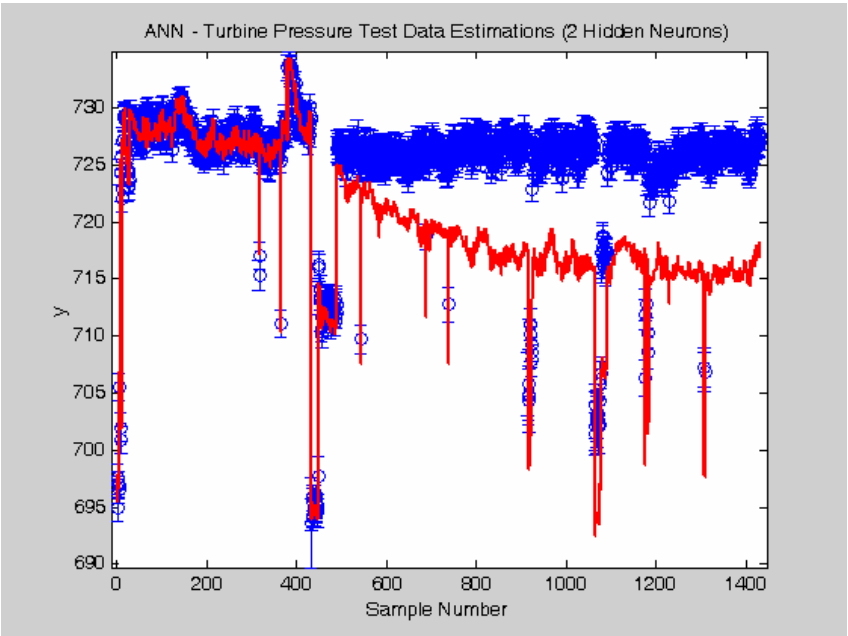


Figure 5.5.11: ANN median result estimation and analytic prediction intervals. Representative result of 2 hidden neuron ANN models. This model exhibited the median value of the 100 Mean Absolute Error results for the 2 hidden neuron models.

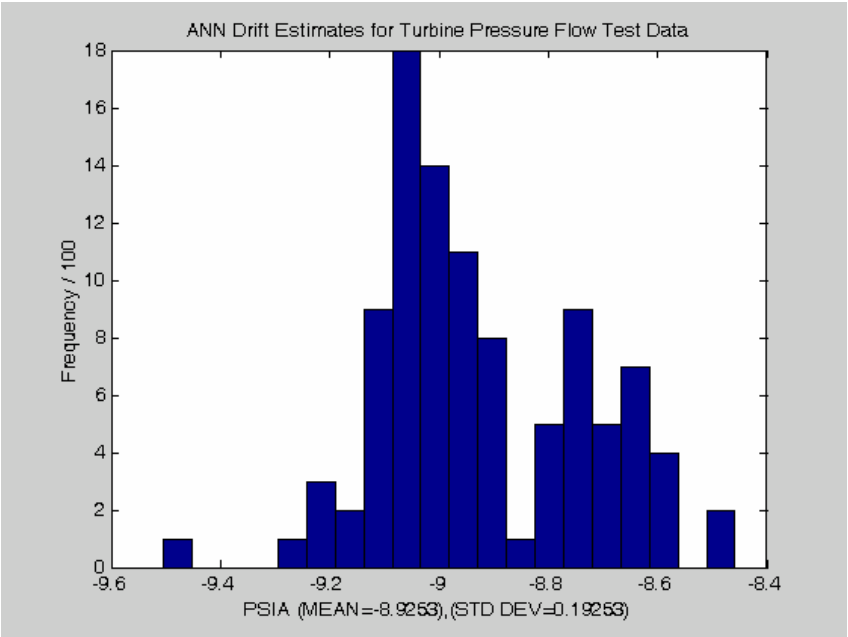


Figure 5.5.12: Distribution of drift estimate from 2 hidden neuron ANN models. Drift estimates obtained from the 100 ANN models via computing the MAE over test samples 600 to 1430. All 100 models provided $\leq$5% coverage in the region of drift.

Overall the ANN models for the turbine pressure data set were stable and consistent. Issues regarding collinearity that resulted in highly unstable models in the case of the feedwater flow data set are not present here. In the absence of collinearity, the ANN architecture produced consistent estimations and analytic prediction intervals that covered the response observations to the expected level of 95%.

5.5.3 Kernel Regression Model Results and Analyses

The PIMs and coverage values for the KR models of the turbine pressure data are provided in figure 5.5.13. As was seen for the KR models of the feedwater flow data set, the bootstrap PIMs exhibit extremely large variation. This was due to a limited number of points for which the estimations were highly variable. Elimination of those points would reduce the variation to more usual levels. The higher values for the analytic PIMs at small bandwidths were due to the large similarly indicate higher than expected coverage. The coverage values for the data pool were consistently above the expected level, and those of the test data consistently inferred a drift in the response.

The MAE results (figure 5.5.14) indicate the expected trend of increasing error with bandwidth due to the influence of bias. The effect was incurred for both the data pool as well as the test data. To determine the optimal bandwidth, the usual analysis of bias and variance was completed (figure 5.5.15). The combined results from the training and validation are shown in figure 5.5.16. The optimal bandwidth was determined to be 0.2.

The plot of the estimations from 4 different KR models with bandwidth of 0.2 (figure 5.5.17) indicates that the results from the KR models are stable and consistent. In each case the drift is clearly identifiable. A more detailed view of a representative result is also provided (figure 5.5.18).

The drift estimate distribution from the KR models of bandwidth 0.2 are extremely consistent and exhibit a minimal variation with respect to the results from the NNPLS and ANN models (figure 5.5.19).

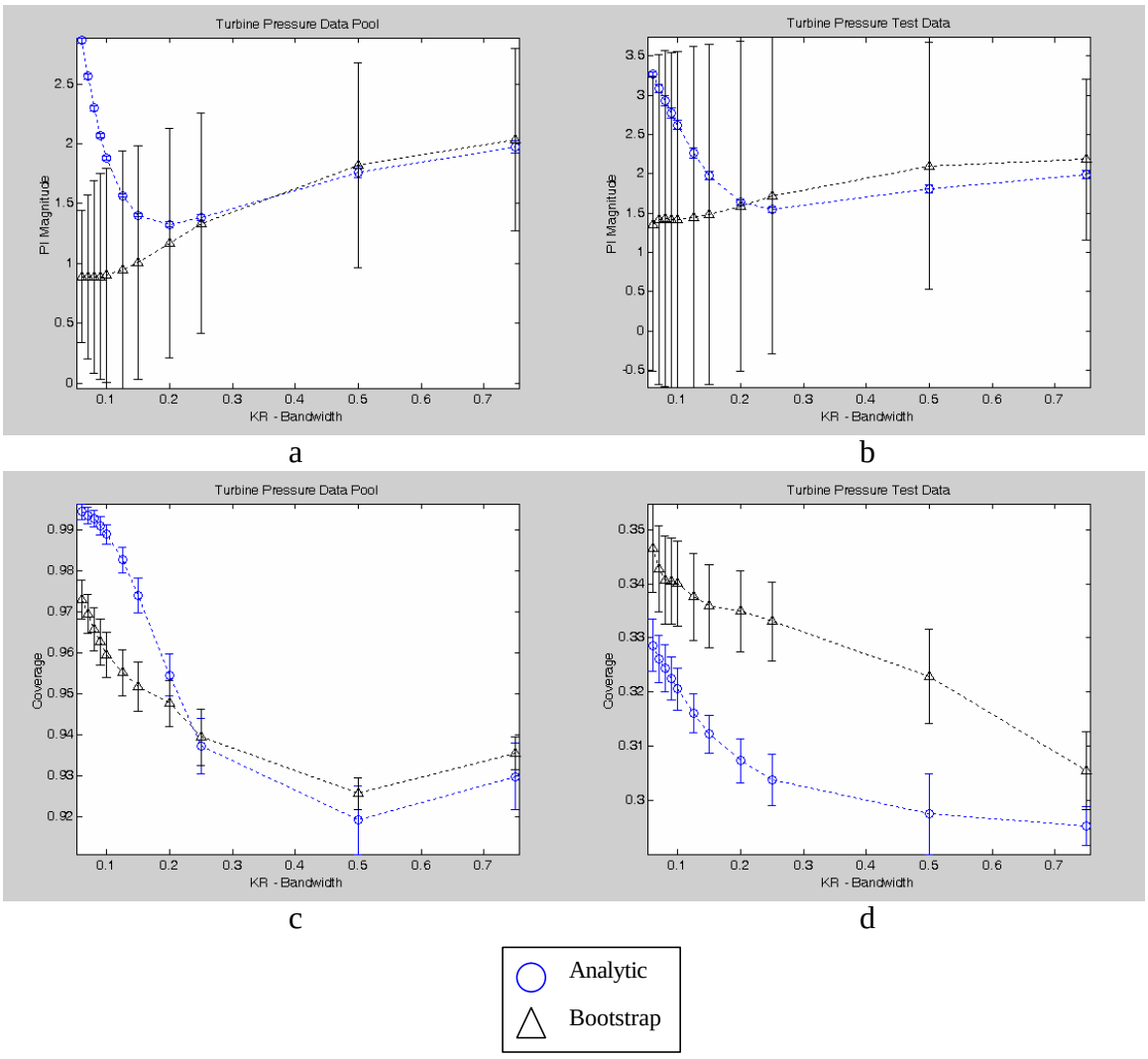
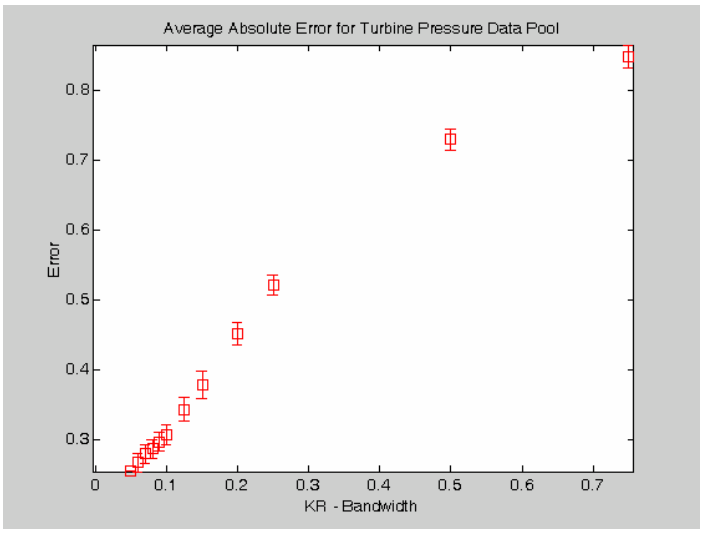
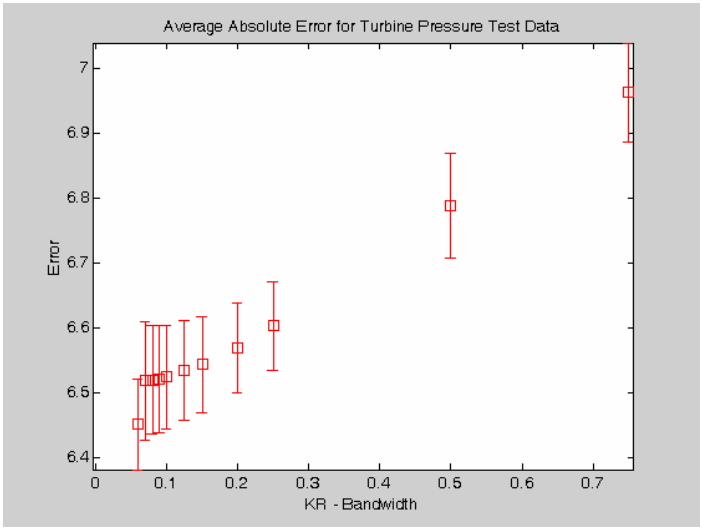


Figure 5.5.13: PI magnitude and coverage results for KR models of turbine pressure data pool. The bands represent the $1s$ variation in the results over the 100 iterations.



a



b

Figure 5.5.14: Mean absolute error results for KR models of turbine pressure data. The bands represent the $1s$ variation in the results over the 100 iterations.

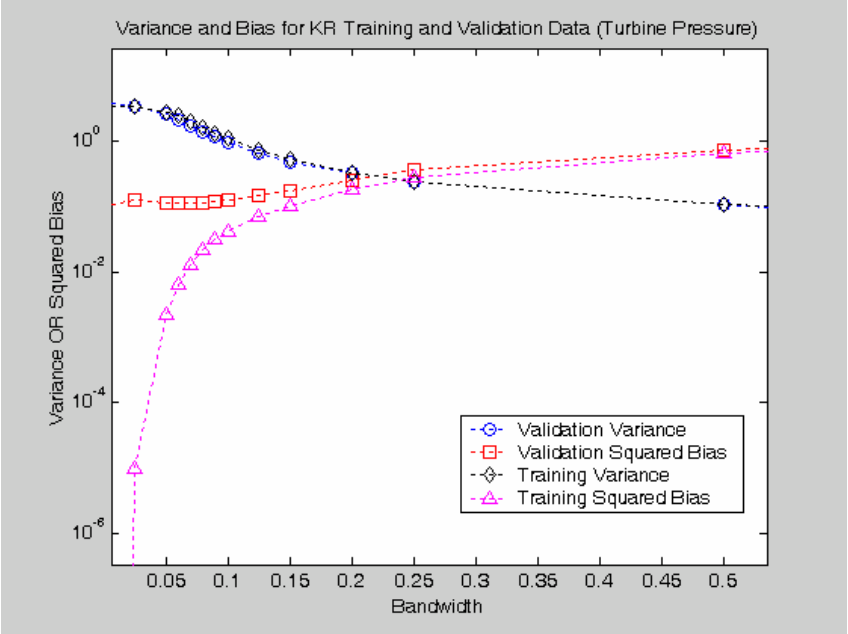


Figure 5.5.15: Variance / squared bias plots for KR models of turbine pressure training and validation data.

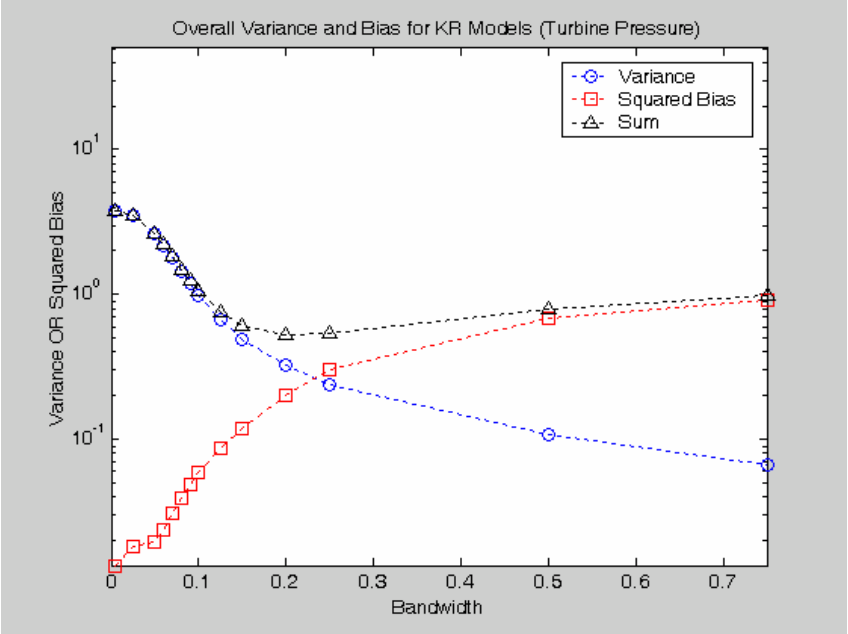


Figure 5.5.16: Combined variance of training and validation data vs. squared bias of data pool for KR models of turbine pressure data.

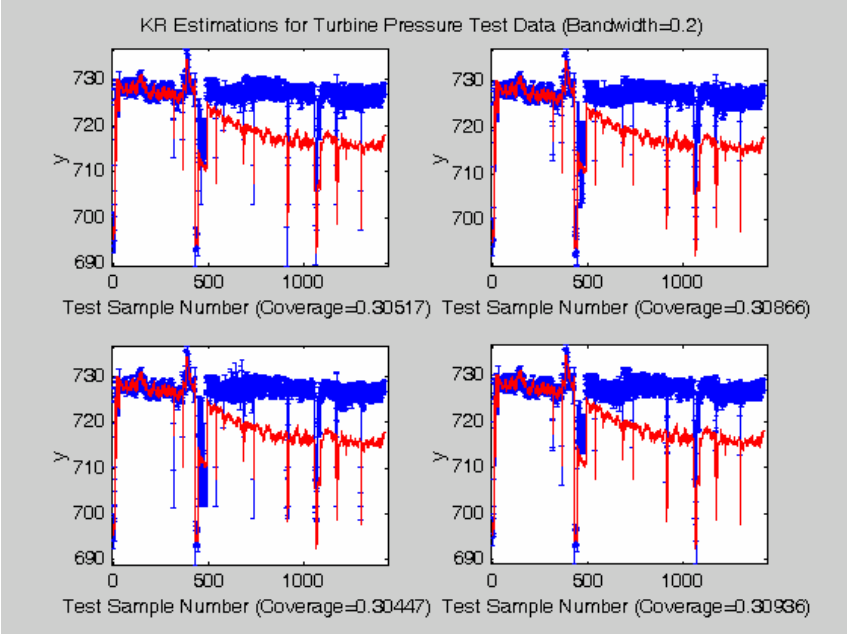


Figure 5.5.17: KR test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 available 0.2 bandwidth KR models. The coverage value for each case is indicated on the x-axis.

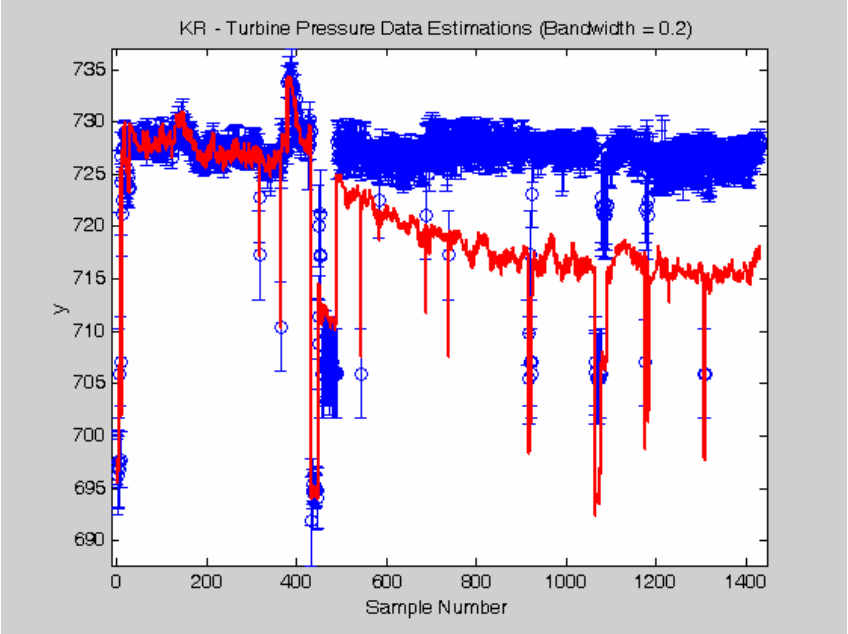


Figure 5.5.18: KR median result estimation and analytic prediction intervals. Representative result from KR model with optimal bandwidth of 0.2. This model exhibited the median value of the 100 Mean Absolute Error results for the bandwidth=0.2 KR models.

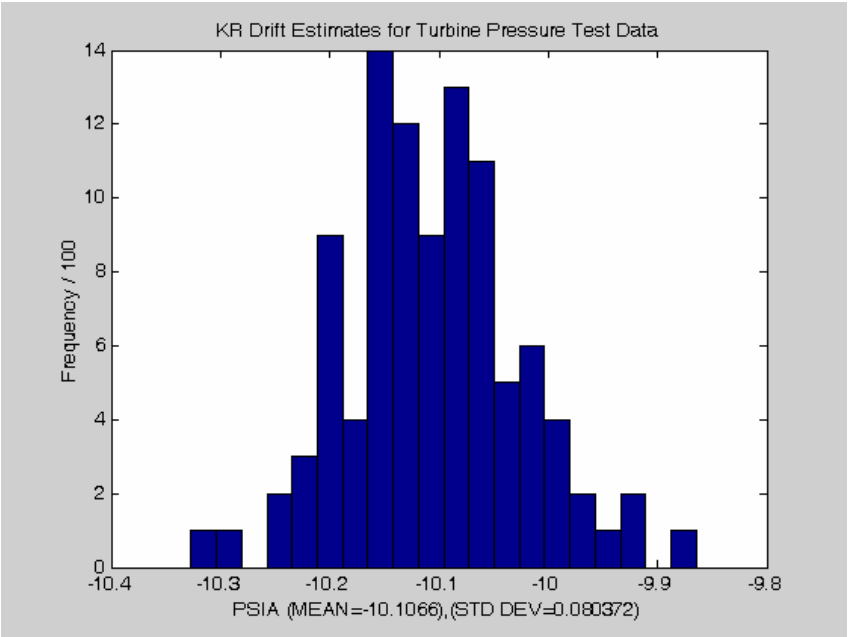


Figure 5.5.19: Distribution of drift estimate from KR models (bandwidth = 0.2). Drift estimates obtained from the 100 KR models via computing the MAE over test samples 600 to 1430. All 100 models provided $\leq$5% coverage in the region of drift.

Overall, the KR models for the turbine pressure data produced consistent models and corresponding prediction intervals. The coverage of the observations in the data pool was at or above the expected value for all cases at or below the optimal bandwidth of 0.2. Above the optimal bandwidth, the coverage values were slightly lower than the expected value, though marginally. The drift estimation was consistent and exhibited little variation from one model to the next.

5.5.4 Local Linear Regression Results and Analyses

The PIMs and Coverage values for the LL models of the turbine pressure data are provided in figure 5.5.20. Again the large deviation of the bootstrap prediction intervals is seen. The odd behavior of the intervals and coverage values for small bandwidths is due to the presence of numerical effects associated with small bandwidths previously discussed and are not relevant to the overall analysis. For all reasonable bandwidth values, the coverage values are at or above the expected level for the data pool, and consistently indicate a drift in the response for the test data.

The MAE results (figure 5.5.21) show the same trend as observed for the KR models, though for the majority of bandwidths the overall averages are higher. This occurs because the optimal bandwidth in this case is at the upper extreme of the bandwidth range for the LL models, whereas it was well within the range for the KR models.

Viewing the individual bias and variance contributions for the training and validation data (figure 5.5.22) as well as the combined contributions (figure 5.5.23), the optimal bandwidth was determined to be 0.75.

Of the 100 LL models with a bandwidth of 0.75, the estimations and prediction intervals from 4 random selected models are shown in figure 5.5.24. The results indicate a clearly discernable drift in all cases, and fairly consistent prediction intervals over the entire response. Some estimates resulted in very wide prediction intervals, but these events were minimal and can be attributed to the additional flexibility of the LL models over the

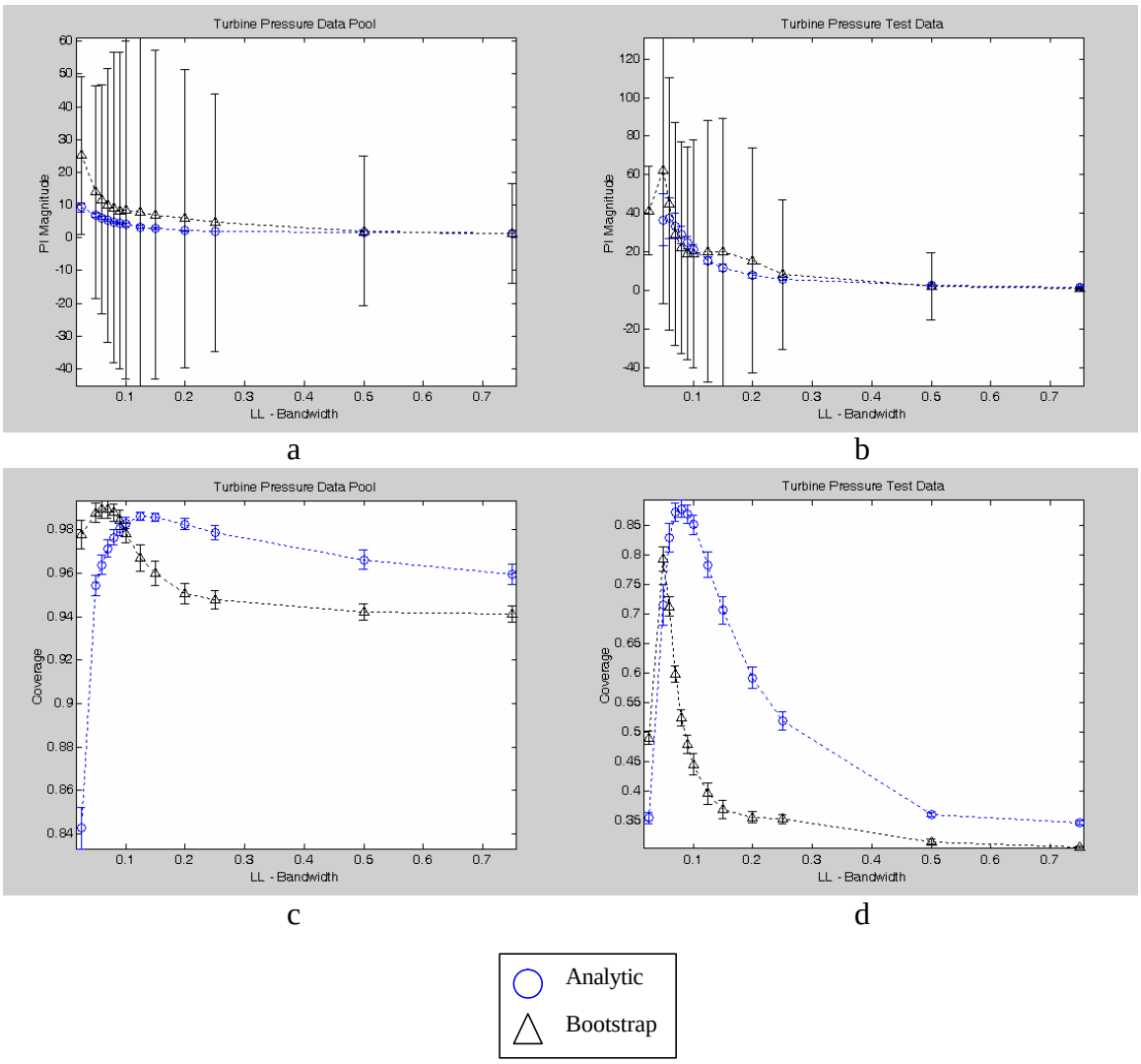
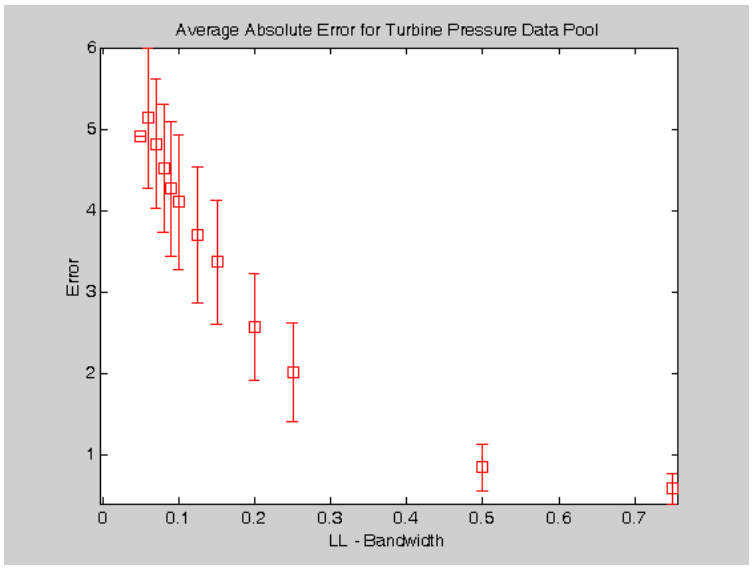
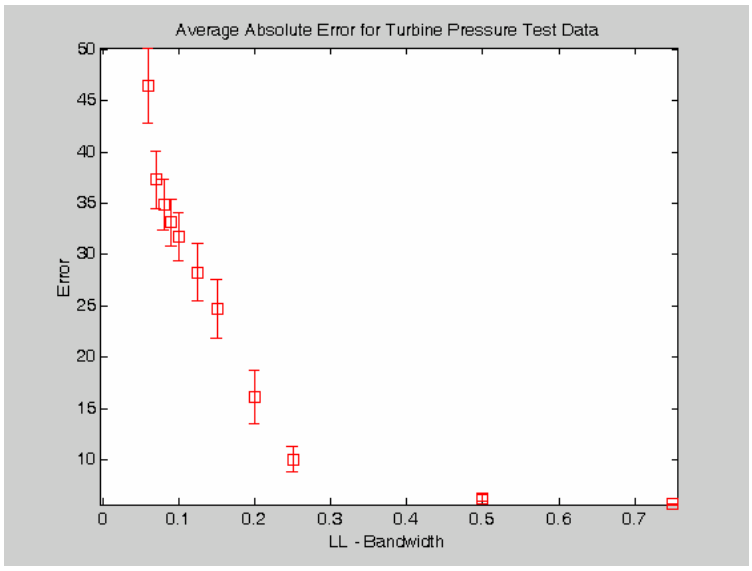


Figure 5.5.20: PI magnitude and coverage results for LL models of turbine pressure data pool. The bands represent the $1s$ variation in the results over the 100 iterations.



a



b

Figure 5.5.21: Mean absolute error results for LL models of turbine pressure data. The bands represent the $1s$ variation in the results over the 100 iterations.

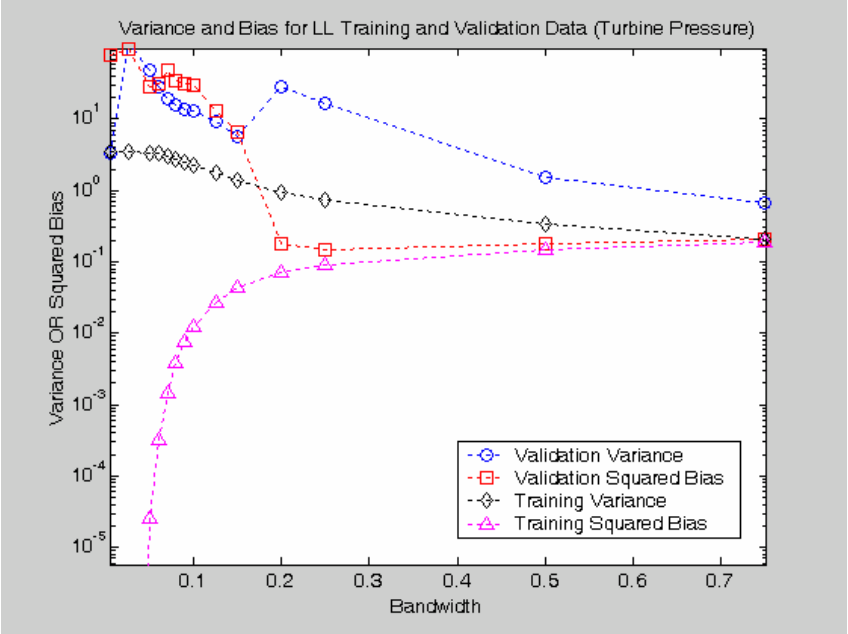


Figure 5.5.22: Variance / squared bias plots for LL models of turbine pressure training and validation data.

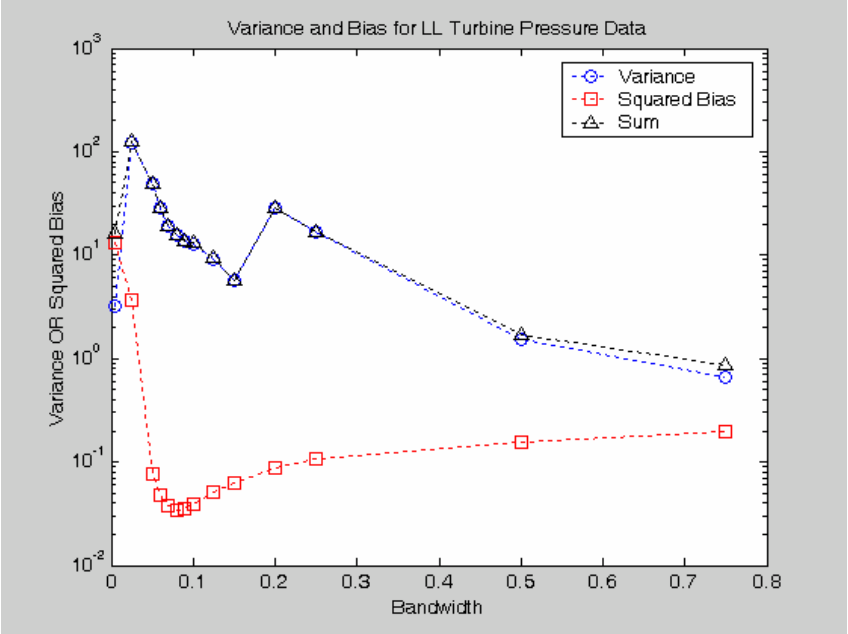


Figure 5.5.23: Combined variance of training and validation data vs. squared bias of data pool for LL models of turbine pressure data.

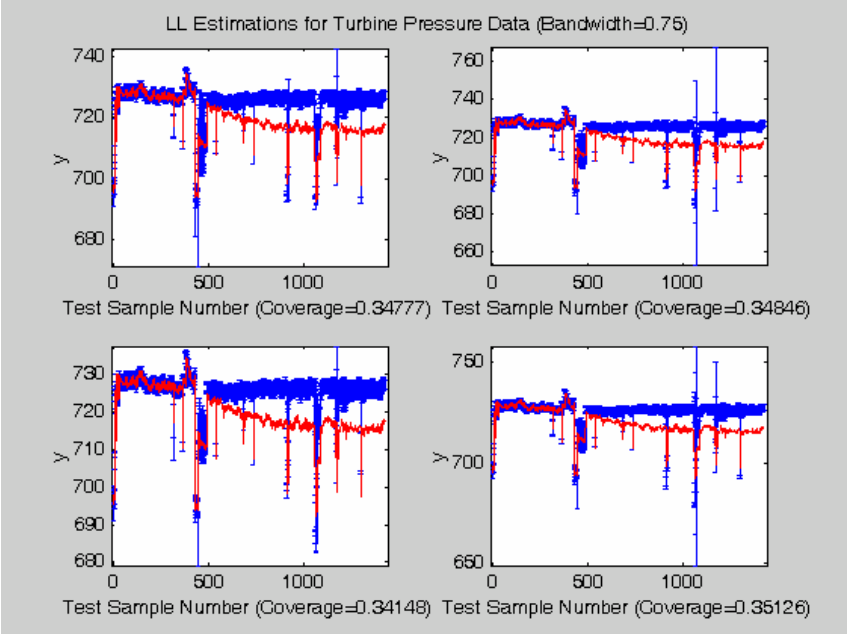


Figure 5.5.24: LL test estimations and their corresponding analytic prediction intervals. These 4 results were selected at random from the 100 available 0.75 bandwidth LL models. The coverage value for each case is indicated on the x-axis.

KR models as far as the range of their estimations. A more detailed view of a representative result is provided in figure 5.5.25.

The distribution of the drift estimates is shown in figure 5.5.26. The LL models consistently estimate the drift magnitude with little variation.

Overall the LL models produced stable estimates and consistent prediction intervals with the appropriate coverage values. The drift estimates are consistent and non-variable from one model to the next.

5.5.5 Turbine Pressure Data Set Summary of Results

Referring to the results obtained based on the data pool, both the KR and LL models provided MAE values of ~0.4PSIA at their optimal architectures which were h=0.2 and h=0.75 respectively. The corresponding results for the 2 hidden neuron ANN models was ~0.5PSIA, and for the 2 latent variable NNPLS model was ~1.3PSIA. Regarding the analytic prediction interval magnitude averages, both KR and LL models provided ~1.3PSIA, the ANN models provided ~1.2PSIA, and the NNPLS models provided ~4.1PSIA. The variation in the interval estimates did not vary much and for the most part matched what would be expected considering the average values. The smallest deviation was noted for the optimal KR models (~0.02PSIA), followed by the ANN models (~0.5PSIA), the LL models (~0.07), and the NNPLS models (~1.4PSIA).

For the NNPLS models, the bootstrap prediction interval magnitudes were significantly higher than their analytic counterparts, with respect to the data pool. While this was noted, it was also seen that the coverage values for all of the NNPLS prediction intervals were at or greater than the expected value. The bootstrap interval coverage values were consistently near 1. Thus, it is proposed that the bootstrap estimates are slightly overestimating the true uncertainty for the NNPLS models in this case, and that the analytic prediction intervals perform as expected and provide sufficient coverage in all cases.

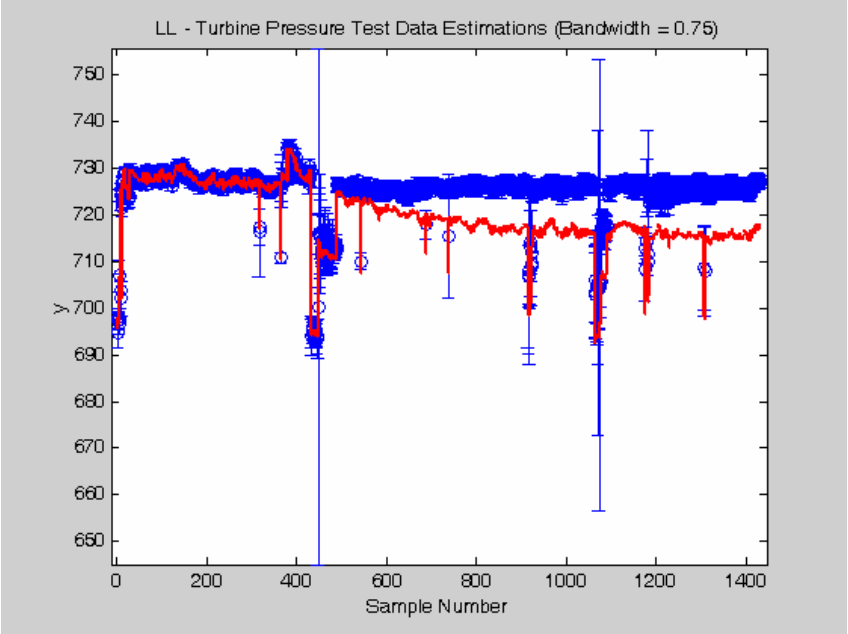


Figure 5.5.25: LL median result estimation and analytic prediction intervals. Representative result from LL model with optimal bandwidth of 0.75. This model exhibited the median value of the 100 Mean Absolute Error results for the bandwidth= 0.75 LL models.

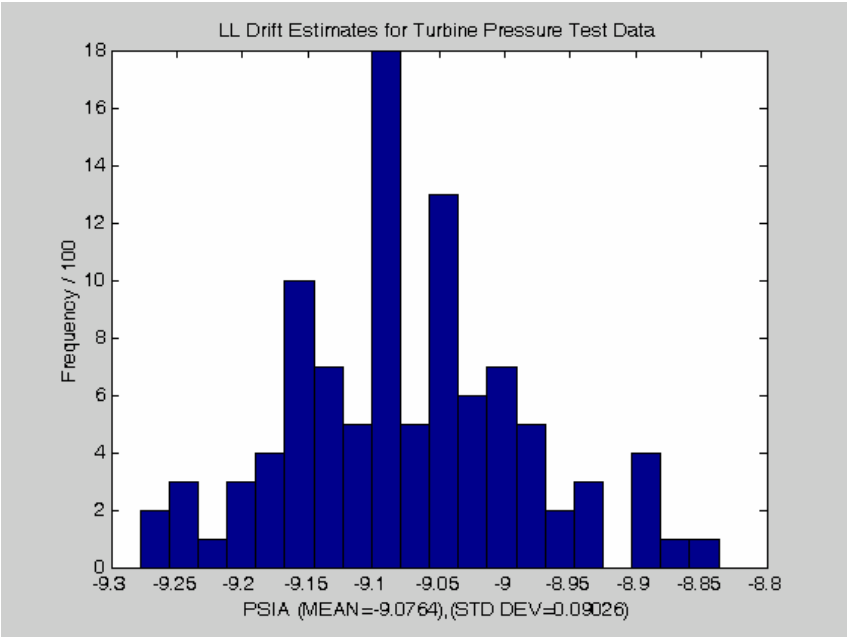


Figure 5.5.26: Distribution of drift estimate from LL models (bandwidth = 0.75). Drift estimates obtained from the 100 KR models via computing the MAE over test samples 600 to 1430. All 100 models provided $\leq$5% coverage in the region of drift.

For the ANN models, the analytic and bootstrap intervals for the data pool were very similar for all evaluated architectures. The coverage values for both approaches to interval estimation remained at or above the expected level for all evaluated architectures.

The results with respect to the data pool for the KR models illustrated the usual trends of high prediction intervals for low bandwidths, decreasing to a minimum, and then increasing as bandwidth increases further due to the influence of the increasing bias. The average bootstrap prediction interval magnitudes at bandwidths below h=0.2 were consistently lower than the average analytic prediction interval magnitudes. This was found to be due to the large variance component of the analytic intervals for these lower bandwidth value models. As this variance component subsides, the analytic and bootstrap prediction interval magnitudes agree within a small margin. The variation in the bootstrap intervals was larger than that observed for the analytic intervals, this effect can be traced to a limited number of observations and is not a substantial issue for the implementation of the proposed interval estimation methodologies. The majority of the coverage values were at or above the expected level. Noted exceptions were for large bandwidths and occurred for both the analytic and bootstrap prediction interval coverage values. Because this effect appears for both interval estimation methodologies it is proposed that the cause in this case is due to the increasing bias stabilizing the model estimates near a constant value, without a correspondingly significant increase in the prediction intervals. Recall that for extremely large bandwidths, the kernel estimator will provide a constant response for all query observations. If in fact a constant estimator results and the prediction intervals do not correspondingly increase, many of the response observations will occur outside of the prediction intervals. Though the most extreme case was described, it is the trend towards this condition that is being proposed as the cause of the slightly lower than expected coverage values for these large bandwidth models. The observed deviation from the expected levels of coverage is relatively minor, the smallest result being 92%.

The LL models for the data pool were not evaluated for a wide enough range of bandwidths to observe the full expected trend of larger prediction intervals reducing to a minimum and then increasing again as the bandwidth of the models goes from smallest to largest. Evidence that this trend would occur is present in the results and if larger bandwidths were evaluated it is expected that a bias increase, exceeding the variance contribution, would be observed (figure 5.5.23). The much larger bootstrap prediction intervals with respect to the corresponding analytic prediction intervals result from the influence of a limited number of observations of the response variable. This effect was previously discussed and illustrated in section 5.4, and was also responsible for the observed variation of the bootstrap intervals. The coverage values of the prediction intervals with respect to the data pool were at or above the expected value, with the following exceptions noted: for the analytic intervals at small bandwidths and for the bootstrap intervals at large bandwidths. The low coverage values for the analytic intervals at small bandwidths indicate that the variance estimate may be slightly lower than what was observed. While the deviation from the expected levels for the bootstrap intervals may be do to the effects of over-regularization as discussed in the previous paragraph.

Focusing on the results for the test data, the NNPLS models prediction intervals were relatively large with respect to the other models. This reflects the uncertainty of the NNPLS architecture for this data set. Correspondingly the coverage values of the prediction intervals for the test data were consistently higher than expected considering the known drift in the test response. At the optimal architecture of 2 latent variables the coverage value was ~36%.

The ANN models were stable for the test data as they were for the data pool. No significant increases in the prediction intervals were observed for the test data over that observed for the data pool. The average coverage value for the optimal architecture (2 hidden neurons) was 32%.

The KR models for the test data also produced prediction interval magnitudes similar to what was observed for the data pool. At the optimal architecture (h=0.2) the coverage of the analytic intervals with respect to the test data response measurements was ~31%.

The odd behavior at low bandwidths for the coverage values of the analytic intervals, for the LL models, was due to the elimination of the NaN estimations from the computations. When the bandwidth increases enough to produce estimations, these estimates strong influence the overall results. As the bandwidth then increases further, the usual trends can be seen. At the optimal bandwidth of those evaluated, the average coverage value for the test data was ~30%. Thus, the drift was clearly discernable.

The drift estimations from the optimal architecture of the evaluated models are summarized in table 5.5.3. Contrary to the results for the feedwater flow rate drift estimates, all optimal model architectures, with the exception of the NNPLS models, clearly identified the drift 100% of the time to the level of certainty of 95%. Excluding the NNPLS models, all optimal architecture models provided consistent drift estimates with minor variation. Because two of the average results are near -9PSIA, it is assumed that this is closer to the true value of the drift. The computed averages for the reduced number of estimates is only relevant for the NNPLS models which in some cases exhibited coverage values $\geq$5%, though the average based on the reduced number of estimates varied negligibly from the original average estimate.

Overall, the NNPLS models performed poorly for this data set. Unlike the highly collinear feedwater flow data set, the turbine pressure data set contained only mild correlations. Previous work documents the need for high correlations in the predictor variables for successful implementation of the NNPLS architecture [Rasmussen 2002]. While, the NNPLS models were not appropriate for the given data, the resultant prediction intervals reflected this fact and the coverage values were consistently at or above the expected level. The coverage of the computed prediction intervals for the test data was higher than incurred for any of the other model architectures (at their optimal

Table 5.5.3: Summary of Turbine Pressure Drift Estimation

Model Type	All Estimates		# of Results where Coverage $\leq$ 5%	Reduced # of Estimates	
	m (PSIA)	s (PSIA)		m (PSIA)	s (PSIA)
NNPLS (2)	-10.5	0.54	81	-10.5	0.44
ANN (2)	-8.9	0.19	100	N/A	N/A
KR (0.2)	-10.1	0.08	100	N/A	N/A
LL (0.75)	-9.1	0.09	100	N/A	N/A

* Numbers in parentheses under model type indicate the number of latent variables (NNPLS), the number of hidden neurons (ANN), or the bandwidth (KR and LL).

Coverage $\leq$5% based on test samples 600-1430.

architecture). This is a logical result considering the uncertainty of the NNPLS models for this data set and the correspondingly large prediction intervals.

The ANN models for this data set on the performed sufficiently well. The prediction intervals provided the appropriate coverage for the data pool, the average errors were minimal, and the coverage of the intervals for the test data consistently inferred the drift in the response variable.

Both the KR and LL models also performed well for this case. The prediction intervals provided the appropriate coverage for the majority of different bandwidth models evaluated. The errors with respect to the data pool were slightly lower than those observed for the ANN models, and the drift in the test data response was clearly identifiable in all cases.

5.6 SUMMARY OF RESULTS

This section provides a summary of the results based on the three data sets independently, followed by an overall summary of the results and observations of this dissertation, and conclusions in section 6.0.

Regarding the cascade data set, optimal performance, with respect to MAE and MSE, was obtained with the optimized LL models. The optimized KR models had very similar error values exhibiting only a marginal increase, followed by another slight increase for the optimized ANN models. This is to be expected for non-linear data since the KR models are known to be more biased than the LL models. The NNPLS model errors were significantly larger for all evaluated architectures; thus its use for this data set was not appropriate. The reason for this is due to the strong non-linear relationships in the data. Previous work using the NNPLS model architecture states that sufficient model performance requires a high level of linearity in the predictor variables [Rasmussen 2002]. The average prediction interval magnitude values for the cascade data followed the same trend as described for the error values, assuming optimal model architecture.

The average magnitudes of the corresponding bootstrap prediction intervals were consistently larger, to a relatively small extent, than the analytic intervals. The minimum average coverage value of the computed analytic prediction intervals for the optimal architecture models was 94% for the optimized KR models. This value is just below the expected value of 95%. All of the analytic prediction interval coverage values for the KR models were between 93% and 95%. Although not exactly equal to 95%, there was expected variability due to the sources of uncertainty, and the values were consistently just below the expected level only to a marginal extent. The only other average coverage value resulted for the over-regularized 1 latent variable NNPLS model, 94%. The corresponding average coverage values for the bootstrap intervals of all model architectures were above the expected level with the exception of large bandwidth KR models, ~93-94%. The cascade data set results for the data pool and the test data were very similar.

The study of the effect of erroneous predictor variables with respect to the cascade test data provided that for both the ANN and NNPLS models, the prediction interval magnitudes increased appropriately, and the coverage of the intervals was not degraded at all by the erroneous predictors. A slight reduction in coverage of the intervals for the cases of the KR and LL models was observed. The minimum average coverage values based on the predictor variable set with erroneous predictor variables were 87% for the KR models and 92% for the LL models.

Regarding the feedwater flow rate data pool, based on the optimal architectures from each model, the minimum average errors were observed for the KR models, than in increasing order, LL, NNPLS, and ANN. In the case of the ANN models the error was significantly greater. Again, these results were expected due to the highly linear relationships between the predictor variables and the response. Regarding the analytic prediction intervals from the optimal architectures of each model, the order of minimum to maximum average value is the same as above for the error values. The variation of the prediction interval magnitudes follows an identical trend. Both the average value and

variation in the prediction intervals of the optimal ANN models were significantly greater than those observed for the optimal architectures of the other models. This uncertainty observed for the ANN models based on the highly collinear set of predictors was an expected result. The large uncertainty for ANN models under these conditions is well known. On the other hand, this data set was ideal for the NNPLS architecture. The results for the NNPLS models were stable and consistent, though did not produce the minimum error or prediction interval magnitudes. The bias inserted through the reduction of the set of 17 predictors to 2 latent variables remains a significant contribution to the computed prediction intervals.

Considering the feedwater flow rate test data, the only major unexpected changes in the analytic prediction intervals with respect to the observed results for the data pool occurred for the LL models. For these models, the bootstrap prediction intervals exhibited significant increases. This event was traced to a limited number of observations in the data set severely influencing the resultant computations.

The coverage values of the computed prediction intervals for the feedwater flow data pool were at or above the expected level for all architectures of the ANN and NNPLS models. On the other hand deviations were observed for the KR and LL models. The range of the average coverage values for the KR model analytic prediction intervals was 86-96%, where 8 of the 15 values were below 95%. For the LL models, all of the values were below 95%, ranging from 81-94%. Part of the observed deviations for the LL models was due to the limited number of observations where an extremely wide distribution of response estimates were observed. Another factor was involved in producing the observed effects, since they were observed for both types of LPR models for this data set as well as the cascade data set.

The drift estimates from the ANN models were unreliable and highly variable, whereas for the remainder of the models the results were fairly consistent. For the optimal model architectures, the average coverage values of the analytic prediction intervals were

between 28-38% overall, excluding the results from the ANN models. In the region of the test data where the drift occurred, the coverage levels were generally $\leq$5%. This result is interpreted in the following statement: "The coverage of the intervals is $\leq$5%, thus the probability that the observed drift represents a *true* instrument channel drift is $\geq$95%."

The results of a study of the effect of the conditional variance estimate for the LPR models on the analytic prediction interval magnitudes revealed that the computations are were not overly sensitive to this estimate. This ensures that the observed results are extendable to other data sets as long as a reasonable estimate of the conditional variance of the response is made.

The best performance for the feedwater flow data set was obtained for the KR and LL models with respect to average errors, though with respect to consistent drift estimations, the best performance was obtained for the NNPLS models.

Considering the turbine pressure data set, the NNPLS models performed poorly. Unlike the highly collinear feedwater flow data set, the turbine pressure data set contained only mild correlations. While, the NNPLS models were not appropriate for the given data, the resultant prediction intervals reflected this fact and the coverage values were consistently at or above the expected level. The ANN models for this data set performed sufficiently well. The prediction intervals provided the appropriate coverage for the data pool, the average errors were minimal, and the coverage of the intervals for the test data consistently inferred the drift in the response variable. Both the KR and LL models also performed well for this case. The prediction intervals provided the appropriate coverage for the majority of different bandwidth models evaluated. The errors with respect to the data pool were slightly lower than those observed for the ANN models, and the drift in the test data response was clearly identifiable in all cases.

The best performance for the turbine data set was observed for the optimal bandwidth LL models, followed by the KR models, and ANN models. The NNPLS architecture is not appropriate for this data set. The most stable drift estimation was observed for the LL models, followed by the KR models, and ANN models. In all cases of optimal model architecture the variation in the drift estimates was minimal (excluding NNPLS model results).

Overall trends in the results are identified and conclusions are drawn in chapter 6. Section 6.1 contains recommendations for future work.

6.0 CONCLUSIONS

One of the main trends observed in the computation of prediction interval estimates was that, considering the nonlinear regression models (ANN and NNPLS), for models below that which were sufficiently complex for the given data set the interval magnitudes were moderate in size. As the complexity of the models was increased, the prediction interval magnitudes decreased to a minimum value. The architecture that exhibited the minimum prediction interval magnitude was of optimal complexity. Further increases in model size resulted in increases in the average prediction intervals. A similar situation was consistently observed for the KR and LL models though rather than model size, the complexity is defined by the model bandwidth.

The analytic prediction interval estimation techniques presented in this dissertation were shown to consistently provide the expected level of coverage for the nonlinear regression models (ANN and NNPLS). For the nonparametric regression models (KR and LL) the computed analytic prediction intervals were slightly lower than the expected value for a fairly sufficient number of models, though the most reduced levels of coverage were $\geq$80%, and for the most part were ~90-94%.

The presence of the slightly lower than expected coverage values for the cascade data set KR models eliminates the notion that these lower values are due to the presence of noise in the predictor variables, since the predictors for the cascade data set were noise-free. In addition, the influence of the conditional variance of the response was known exactly for the cascade data set. Thus, the notion that this estimate is the source of the observed discrepancies is also refuted by the results. In the recommendations (section 6.1) several suggestions are provided for the improvement of the proposed prediction interval estimation techniques.

Regarding the observed excessive variations in the bootstrap prediction intervals for the LPR models, keeping in mind that these values are averages, the majority of the average point-wise bootstrap intervals were similar to those from the analytic computations. The

huge increases for the bootstrap intervals with respect to the analytic intervals on average, appears excessive but all things considered is not extremely significant. Assuming calibration decisions will be made based on estimates over a period of time, and not instantaneously, the extremely variable behavior of the estimates and intervals at a select few observations does not preclude the successful use of these techniques and corresponding prediction interval computations in practical applications.

Overall, the proposed prediction interval techniques were consistently proven to be adequate for the ANN and NNPLS models. For the LPR models, the resultant coverage values were slightly below the expected value though the deviations were generally minimized in the range of optimal bandwidth. The trends in the average magnitudes of the prediction intervals suitable reflect the complexity of the models, and were also shown to be relatively consistent under conditions of model misspecification due to erroneous predictor variables. The minor deficiencies in the coverage of the intervals for the KR and LL models observed were slightly worsened for the misspecified model. While the coverage values for the ANN and NNPLS models remained consistent under the model misspecification. For both data sets where the response variable contained a known drift, the resultant coverage values reflected the presence of the drift, indicating that the proposed methods can provide a supporting 95% significance to predictions of instrument channel drift.

6.1 RECOMMENDATIONS FOR FUTURE WORK

The instability of the ANN models when applied to collinear data sets can be sufficiently reduced through the implementation of a weight decay term in the objective function. The effect of the weight decay term is the effective elimination of unnecessary free parameters. Investigations into prediction interval estimation for ANN models trained with a weight decay term in the objective function have been reported [De Veaux 1998]. In addition this technique can be applied to the NNPLS architecture. Proposed benefits of the weight decay objective function and the associated modified prediction intervals

are to reduce the widening of prediction intervals which occurs when early-stopping is used in neural network training.

Improvements in the nonparametric regression estimators have been reported when a variable bandwidth is implemented [Fan 1992a]. Consider the explanation provided by Ruppert and Wand [1994]: if the true function $m(x)$ has a high amount of curvature near x, then it will be important to have more information from nearby observations so one would want the bandwidth to been to these nearby points and reduce the bias of the estimator. However, if $m(x)$ is closer to being linear at x, then variance considerations dictate that one would want more data included in the fitting process and it would be better to have a larger bandwidth.

In most cases, empirical models will be used to estimate values from within the regions where the data used to develop the models were collected. For predictions in which the model is applied outside the trained region (extrapolation) the confidence intervals will not suitably reflect the relatively large confidence intervals which should be imposed at this point. Proper implementation of confidence interval estimation should include accommodations for allowing the density of the training data in the region of the current prediction to influence the level of confidence at the given point. To enable users without expertise in empirical modeling to employ these models in various situations, indications that confidence is low due to extrapolated predictions is necessary; even knowledgeable persons may at times neglect to realize when extrapolation is occurring. Reports of a wavelet based density estimator in the context of feedforward neural network prediction intervals has been presented by Shao et. al. [1997], and Leonard et. al. [1992] have investigated density estimation for radial basis function networks. It is recommended that the use of a density estimator be investigated to further this research.

The presence of measurement noise in the predictor variables is unavoidable in all practical applications of empirical models for signal validation in industrial processes. The effect of this predictor variable noise on the resultant prediction intervals should be quantified. In this dissertation it was shown that the computed prediction intervals

provided the expected coverage for the cascade data set, where the predictors were noise-free. It was also shown that proper coverage resulted for the *real* data sets where the predictor variables contained noise. Thus, the prediction intervals perform adequately in both situations. As discussed in section 4.12, the presence of noise is not necessarily harmful for constructing empirical models. In fact, noise is often added to predictor variables to provide a level of regularization to neural network models. The added random noise reduces the possibility of overfitting and provides improved generalization capabilities. A study of the change in the magnitudes of the computed prediction intervals for noise containing vs. noise-free predictors would better quantify the effect, and further studies may reveal a predictive relationship between the noise level in the predictors and the magnitude increases in the prediction intervals.

Another study that would lead to further understanding the prediction interval techniques is to begin with a small training data set and to increase the number of observations in the training data set, evaluating the prediction intervals for each training sample. The expected effect is that the uncertainty, reflected in the magnitudes of the computed prediction intervals, should decrease as the size of the training set increases. This would be illustrative for practical purposes in forming a guideline by which training data size can be evaluated for a given architecture and model type. It should be mentioned that for small numbers of training observations the prediction interval methods for ANNs have been reported to be unreliable [De Veaux 1998]. In addition, the prediction interval computations for small training sample sizes will need to be modified to divide by the appropriate degrees of freedom (n-p). Where n is the number of observations and p is the number of parameters. In addition the critical values from the t distribution should be used rather than the simplification of assuming $t_{n-p}^{0.05/2} \approx 2$ as $n \to \infty$. For small sample sizes, the estimated variance components will be strongly affected by this modification.

A final recommendation proposes to improve the coverage values that were observed to be marginally deficient with respect to the nonparametric models. The definitions of variables in the equations presented below were provided in the appropriate sections of

chapter 4. The standard error of the estimator for the case of the nonlinear regression models is estimated by [Tibshirani 1996]:

$$se^*[\hat{y}_0] = \left[\mathbf{f}_0^{\mathbf{T}}(\mathbf{F}^{\mathbf{T}}\mathbf{F})^{-1}\mathbf{f}_0\right]^{-1/2}$$

The variance of the nonlinear regression estimator is given by [Chryssolouris 1996:

$$\text{var}[\hat{y}_0] = s^2\mathbf{f}_0^{\mathbf{T}}(\mathbf{F}^{\mathbf{T}}\mathbf{F})^{-1}\mathbf{f}_0$$

The approximate 95% prediction interval for the nonlinear regression estimator is then:

$$2\sqrt{s^2 + s^2\mathbf{f}_0^{\mathbf{T}}(\mathbf{F}^{\mathbf{T}}\mathbf{F})^{-1}\mathbf{f}_0}$$

In the above equations, there are no explicit accommodations for model bias. It was described in section 4.12 how to modify the noise variance estimate to account for bias via extending the computation over a set of data which includes observations that were omitted from the training data set.

Consider the simplified form of the 95% prediction interval estimate for nonparametric regression:

$$2\sqrt{\text{var}(\hat{m}) + bias(\hat{m})^2}$$

Here there is an explicit representation for the model bias. The bias estimate for this case is equivalent to the root mean square error, extended over a set of data which includes observations independent from the data set. This allows for a better overall measure of model bias. Note that this is the same as the quantity used in the nonlinear regression case to estimate the variance. The assumption is that by broadening the scope of the root mean square error calculation, both the noise variance and model bias are accounted for. The variance in the above equation can be written as [Ruppert 1994]:

$$Var(\hat{m}) = \mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{V}\mathbf{W}_x\mathbf{X}_x(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{e}_1$$

Assuming the noise variance is homoscedastic, this can be rewritten as:

$$Var(\hat{m}) = s^2\mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{W}_x\mathbf{X}_x(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{e}_1$$

Noting the similarity between this form and that of the variance for the nonlinear estimator, suggests the following estimate for the 95% prediction interval:

$$2\sqrt{s^2 + s^2\mathbf{e}_1^{\mathbf{T}}(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{W}_x\mathbf{X}_x(\mathbf{X}_x^{\mathbf{T}}\mathbf{W}_x\mathbf{X}_x)^{-1}\mathbf{e}_1}$$

where $s \approx s = \sqrt{\frac{[y - \hat{y}]^2}{n}}$ is computed over an extended set of data including the training data as well as an adequate sampling of independent data.

The motivation for the recommendation of modifying the prediction interval computation for the nonparametric regression models is to reduce the observed occurrences of lower than expected coverage values. These effects were noted throughout the results sections of this dissertation. The suggested modified form of the prediction interval computation is an analogous form to that used for the nonlinear regression models. Because the coverage values for the prediction intervals computed for the nonlinear regression models were consistently at or above the expected value of 0.95, it is suggested that an analogous representation for the nonparametric regression models could produce the same results.

A final recommendation is to consider an alternate form for the estimate of bias. This approach to bias estimation is based on the Fisher Information matrix, i.e. the expected value of the Hessian matrix. The Fisher information matrix measures the average accuracy that would be obtained from repeated samples of the same underlying experiment. It also provides a measure of the information available in a sample to estimate the parameters of a model. These ideas are strongly based on information theory to which interested readers are referred [Cover 1991], and a simple search on the internet will also provide a wealth of information regarding Fisher information and its applications. The basic ideas are presented here. Given a density function $f(y_i \mid x_i)$, the likelihood function can be defined as:

$$L \propto f(y_1, y_2, ..., y_n \mid x_i) = \prod_{i=1}^{n} f(y_i \mid x_i)$$

The log-likelihood function is then:

$$\ell = \ln(L) \propto \ln\left[f(y_1, y_2, ..., y_n \mid x_i) \right] = \sum_{i=1}^{n} \ln\left[f(y_i \mid x_i) \right]$$

Two forms of the Fisher information matrix can be defined, the inner product form (F) and the outer product form (R):

$$F = -E\left[\frac{\partial^2 \ell}{\partial q^{\mathrm{T}} \partial q}\right] \qquad R = E\left[\frac{\partial \ell}{\partial q} \times \frac{\partial \ell}{\partial q}\right]$$

The two forms of the Fisher information matrix will be equivalent for correctly specified models, and different for misspecified models; thus, an estimate of bias can be defined as:

$Bias = \frac{1}{n} trace\left(\hat{F}^{-1}\hat{R}\right)$, where the $\hat{}$ indicates that the matrices are estimated from the n training observations, and *trace* performs a summation over the diagonal elements of a matrix. This approach to bias estimation is based in information theory, and provides a viable alternative to the more empirical based bias estimate used in this work.

REFERENCES

Ait-Sahalia, Yacine, Peter J. Bickel, Thomas M. Stoker, (2001), "Goodness-of-fit tests for kernel regression with an application to option implied volatiles," *Journal of Econometrics*, **105**, 363-412.

Atkeson, Christopher G., Andrew W. Moore, and Stefan Schaal, (1996), "Locally weighted learning."

Atkeson, Christopher G., and Stefan Schaal, (1995), "Memory-based neural networks for robot learning," *Neurocomputing*, **9**, 243-269.

Baffi, Giuseppe, Elaine Martin, and Julian Morris, (2002), "Prediction intervals for non-linear projection to latent structures regression models," *Chemometrics and Intelligent Laboratory Systems*, **61**, 151-165.

Baffi, G., E.B. Martin, and A.J. Morris, (2000), "Non-linear dynamic projection to latent structures modeling," *Chemometrics and Intelligent Laboratory Systems*, **52**, 5-22.

Baffi, G., E.B. Martin, and A.J. Morris, (1999a), "Non-linear projection to latent structures revisited: the quadratic PLS algorithm," *Comp. Chem. Engng.*, **23**, 395-411.

Baffi, G., E.B. Martin, and A.J. Morris, (1999b), "Non-linear projection to latent structures revisited (the neural network PLS algorithm)," *Comp. Chem. Engng.*, **23**, 1293-1307.

Berger, Andrew J., and Michael S. Feld, (1997), "Analytical method of estimating chemometric prediction error," *Applied Spectroscopy*, **51**, no. 5, 725-732.

Berglund, Anders, and Svante Wold, (1997), "INLR, Implicit non-linear latent variable regression," *Journal of Chemometrics*, **11**, 141-156.

Bickel, P.J., M. Rosenblatt, (1973), "On Some Global Measures of the Deviations of Density Function Estimates," *Annals of Statistics*, **1**, 1071-1095.

Birattari, Mauro, and Gianluca Bontempi, (1999), The Lazy Learning Toolbox: user's manual, Technical Report TR/IRIDIA/99-7, Iridia, Université Libre de Bruxelles, internet source: http://iridia.ulb.ac.be/~lazy/.

Box, G.E.P. and N.R. Draper, (1987), "Empirical model-building and response surfaces, John Wiley & Sons, New York.

Bro, Rasmus, (1995), "Short Communication: algorithm for finding an interpretable simple neural network solution using PLS," *Journal of Chemometrics*, **9**, 423-430.

Broomhead, D.S., and D. Lowe, (1990), "Multi-varial functional interpolation and adaptive networks," *Complex Systems*, **2** 321-355.

Cai, Zongwu, (2001), “Weighted Nadaraya-Watson regression estimation,” *Statistics & Probability Letters*, **51**, 307-318.

Carney, John G., Pádraig Cunningham, and Umesh Bhagwan, (1999), “Confidence and prediction intervals for neural network ensembles,” *IEEE International Joint Conference on Neural Networks*, 1215-1218.

Carroll, R.J., D. Ruppert, and L.A. Stefanski, (1995), Measurement Error in Nonlinear Models, *Monographs on Statistics and Applied Probability*, Chapman and Hall, London.

Cheng, M.Y., J. Fan, and J.S. Marron, (1993), "Minimax efficiency of local polynomial fit estimators at boundaries," *Institute of Statistics Mimeo Series #2098*, University of North Carolina at Chapel Hill.

Chryssolouris, George, Moshin Lee, and Alvin Ramsey, (1996), "Confidence interval prediction for neural network models," *IEEE Transactions on Neural Networks*, **7**, no. 1, 229-232.

Cleveland, William S., and Susan J. Devlin, (1988), “Locally weighted regression: an approach to regression analysis by local fitting,” *Journal of the American Statistical Association*, **83**, 596-610.

Cleveland, William S., (1979), “Robust locally weighted regression and smoothing scatterplots,” *Journal of the American Statistical Association*, **74**, 829-836.

Cover, T.M., and J.A. Thomas, (1991), "Elements of Information Theory," John Wiley & Sons, NY.

Cybenko, G., (1989), "Approximation by superposition of a signoidal function," *Math Control Signals Syst.*, **2**, 303-314.

Da Silva, Alexandre P., and Luciano S. Moulin, (2000), “Confidence intervals for neural network based short-term load forecasting,” *IEEE Transactions on Power Systems*, **15**, no. 4, 1191-1196.

Davis, Eddie, Dan Funk, Dave Hooten, and Richard Rusaw, (1998), “On-Line Monitoring of Instrument Channel Performance,” EPRI TR-104965.

Dayal, Bhupinder S., and John F. MacGregor, (1997), “Improved PLS algorithms,” *Journal of Chemometrics*, **11**, 73-85.

Dayhoff, Judith E., and James M. DeLeo, (2001), “Artificial Neural Networks: opening the black box,” *Cancer*, supplement, **91**, no. 8, 1615-1635.

De Groot, P.F.M., A.H. Bril, F.J.T. Floris, and A.E. Campbell, (1996), "Monte Carlo simulation of wells," *Geophysics*, **61**, 631-638.

De Veaux, R.D., J. Schumi, J. Schweinsberg, and L.H. Unger, (1998), "Prediction intervals for neural networks via nonlinear regression," *Technometrics*, **40**, 273-282.

De Vries, S., and Cajo J.F. Ter Braak, (1995), "Prediction error in partial least squares regression: a critique on the deviation used in the Unscrambler," *Chemometrics and Intelligent Laboratory Systems*, **30**, 239-245.

Denham, Michael C., (1997), "Prediction intervals in partial least squares," *Journal of Chemometrics*, **11**, 39-52.

Derks, E.P.P.A., and L.M.C. Buydens, (1998), "Aspects of network training and validation on noisy dta Part 2. validation aspects," *Chemometrics and Intelligent Laboratory Systems*, **41**m 185-193.

Draper, N.R., and H. Smith, (1966), Applied Regression Analysis, John Wiley & Sons, New York.

Dybowski, Richard, (1997), "Assigning confidence intervals to neural network predictions," Technical Report 2, March, Summary of presentation at the Neural Computing Applications Forum (NCAF), London, January, 1997.

Edwards, Peter J., Andrew M. Peacock, David Renshaw, John M. Hannah, and Alan F. Murray, (2002), "Minimizing risk using prediction uncertainty in neural network estimation fusion and its application to papermaking," *IEEE Transactions on Neural Networks*, **13**, no. 3, 726-731.

Efron B., and R.J. Tibshirani, (1993), "An introduction to the bootstrap," Chapman and Hall, New York.

Efron, B., (1992), "Six questions raised by the bootstrap," In: Exploring the Limits of Bootstrap, Ed: Raoul Lepage, and Lynne Billard, John Wiley & Sons, New York.

Efron, B., (1982), "The jacknife, the bootstrap, and other resampling plans" Society for Industrial and Applied Mathematics, Philadelphia.

Eubank, R.L., and P.L. Speckman, (1993), "Confidence bands in nonparametric regression," *Journal of the American Statistical Association*, **88**, 1287-1301.

Faber, Nicolaas (Klaas) M., (2000a), "Comparison of two recently proposed expressions for partial least squares regression prediction error," *Chemometrics and Intelligent Laboratory Systems*, **52**, 123-134.

Faber, Nicolaas (Klaas) M., (2000b), “Response to Comments on construction of confidence intervals in connection with partial least squares,” *Journal of Chemometrics*, **14**, 363-369.

Faber, Nicolaas (Klaas) M., (1999), “A closer look at the bias-variance trade-off in multivariate calibration,” *Journal of Chemometrics*, **13**, 185-192.

Faber, Klaas, and Bruce R. Kowalski, (1997), “Propagation of measurement errors for the validation of predictions obtained by principal component regression and partial least squares,” *Journal of Chemometrics*, **11**, 181-238.

Faber, Klaas, and Bruce R. Kowalski, (1996), "Prediction error in least squares regression: Further critique on the deviation used in the Unscrambler," *Chemometrics and Intelligent Laboratory Systems*, **34**, 283-292.

Fan, Jianqing, and Irene Gijbels, (1996), Local Polynomial Modelling and its Applications, Chapman and Hall, London.

Fan, Jianqing, Nancy E. Heckman, and M.P. Wand, (1995a), "Local polynomial kernel regression for generalized linear models and quasi-likelihood functions," *Journal of the American Statistical Association*, **90**, 141-150.

Fan, J., T. Gasser, I. Gijbels, M. Brockman, and J. Engel, (1995b), "On nonparametric estimation via local polynomial regression," Discussion Paper #9511, Institute of Statistics, Catholic University of Louvain, Louvain-la-Neuve, Belgium.

Fan, Jianqing, (1993a), “Local linear regression smoothers and their minimax efficiencies,” *Annals of Statistics*, **21**, no. 1, 196-216.

Fan, Jianqing, (1992a), “Design-adaptive nonparametric regression,” *Journal of the American Statistical Association*, **87**, 998-1004.

Fan, Jianqing, and Iréne Gijbels, (1992b), “Variable bandwidth and local linear regression smoothers,” *Annals of Statistics*, **20**, 2008-2036.

Fantoni, Paolo F., Mario Hoffmann, Ramesh Shankar, and Eddie Davis, (2002), "On-line monitoring of instrument channel performance in nuclear power plant using PEANO," *Proceedings of the 10th International Conference on Nuclear Engineering*, April 14-18, Arlington, VA.

Fantoni, P.F., and A. Mazzola, (1996), "A pattern recognition-artificial neural networks based model for signal validation in nuclear power plants," *Annals of Nuclear Energy*, **23**, no. 13, 1069-1076.

Fine, Terrence L., (1999), Feedforward Neural Network Methodology, Springer-Verlag, NY.

Gasser, T., and H.G. Müller, (1979), "Kernel estimation of regression functions," In: Smoothing techniques for curve estimation, *Lecture notes in mathematics*, **757**, 23-68, Springer-Verlag, NY.

Geladi, Paul, and Bruce R. Kowalski, (1986), "Partial least squares regression: a tutorial," *Analytica Chimica Acta*, **185**, 1-17.

Gemperline, Paul J., (1997), "Rugged spectroscopic calibration for process control," *Chemometrics and Intelligent Laboratory Systems*, **39**, 29-40.

Gil, Juan A., and Rosario Romera, (1998), "On robust partial least squares (PLS) methods," *Journal of Chemometrics*, **12**, 365-378.

Gribok, Andrei V., Aleksey M. Urmanov, and J. Wesley Hines, (2003), "Uncertainty Analysis of Memory Based Sensor Validation Techniques," *Real Time Systems* Special Issue on "Applications of Intelligent Real-Time Systems for Nuclear Engineering", accepted for publication.

Gribok, Andrei V., J. Wesley Hines, and Aleksey M. Urmanov, (2002), "Uncertainty analysis of MSET sensor estimates," final report under research contract with Smart Signal Inc, Lisle, IL.

Gross, K.C., S.W. Wegerich, R.M. Singer, and J.E. Mott, (1998), "Industrial Process Surveillance System," U.S. Patent # 5,764,509.

Hadamard, J., (1923), "Lectures on Cauchy's problem in linear partial differential equations," Yale University Press, New Haven, CT.

Hall, Peter, (1992), "On bootstrap confidence intervals in nonparametric regression," *Annals of Statistics*, **20**, no. 2, 695-711.

Hand, D.J., (1981), "Discrimination and Classification," Wiley, New York.

Härdle, W., and J.S. Marron, (1991), "Bootstrap simultaneous error bars for nonparametric regression," *Annals of Statistics*, **19**, 778-796.

Härdle, Wofgang, (1990), Applied Nonparametric Regression, *Econometric Society Monographs No. 19*, Cambridge University Press.

Härdle, W., and A. Bowman, (1988), "Bootstrapping in nonparametric regression: local adaptive smoothing and confidence bands," *Journal of the American Statistical Association*, **83**, 102-110.

Hastie, T.J., and R. Tibshirani, (1986), "Generalized additive models," *Statist. Sci.*, **1**, 297-318.

Hecht-Nielson, R., (1989), "NeuroComputing," Addison-Wesley, Reading, MA.

Heimes, Felix, and Bram van Heuveln, (1998), "The normalized radial basis function neural network," *IEEE.*

Hines, J. Wesley,

Hines, J. Wesley, and Brandon Rasmussen, (2001), "On-line sensor calibration verification: a survey," *14th International Congress and Exhibition on Condition Monitoring & Diagnostic Engineering Management*, Manchester, England, September.

Hines, J. Wesley, Andrei V. Gribok, Robert Uhrig, and Ibrahim Attieh, (2000a), "Neural network regularization techniques for a seneor validation system," American Nuclear Society Annual Meeting, San Diego, California, June 4-8.

Hines, J. Wesley, Andrei V. Gribok, Ibrahim Attieh, and Robert E. Uhrig, (2000b), "Improved methods for on-line sensor calibration verification," 8th International Conference on Nuclear Engineering, Baltimore, MD, April 2-6.

Hines, J. Wesley, Brandon Rasmussen, and Robert E. Uhrig, (2000c), "An on-line sensor calibration monitoring system," 13th International Congress and Exhibition on Condition Monitoring and Diagnostic Engineering Management, Houston, Texas, December 3-8.

Hines, J. W., Andrei V. Gribok, Ibrahim Attieh, and Robert E Uhrig,, (1999), "Regularization Methods for Inferential Sensing in Nuclear Power Plants," In: Fuzzy Systems and Soft Computing in Nuclear Engineering,, Ed: Da Ruan, Springer, 1999.

Hines, J. Wesley, Robert Uhrig, Chris Black, and Xiao Xu, (1997a), "An evaluation of instrument calibration monitoring using artificial neural networks," *Proceedings of the American Nuclear Society*, Albuquerque, NM, November 16-20.

Hines, J.W., and R. E. Uhrig, (1997b), "Use of Autoassociative Neural Networks for Signal Validation,", *Journal of Intelligent and Robotic Systems*, Kluwer Academic Press, 1997b.

Ho, S.L., M. Xie, L.C. Tang, K. Xu, and T.N. Goh, (2001), "Neural network modeling with confidence bounds: a case study on the solder paste deposition process," *IEEE Transactions on Electronics Packaging Manufacturing*, **24**, no. 4, 323-332.

Hodouin, D., J.F. MacGregor, M. Hou, and M. Franklin, ( ), "Multivariate statistical analysis of mineral processing plant data," *Mineral Processing*.

Hoerl, A.E., and R.W. Kennard, (1970), "Ridge regression. Biased estimation to nonorthogonal problems," *Technometrics*, **12**, 55-67.

Hogg, R.V., and J. Ledolter, (1987), *Engineering Statistics*, Macmillan, New York

Holcomb, T.R., and M. Morari, (1992), "PLS / neural networks," *Comp. Chem. Engng.*, **16**, no. 4, 393-411.

Hornik, K.M., (1993), "Some new results on neural network approximation," *Neural Net*, **6**, 1069-1072.

Hornik, K.M., (1990), "Approximation capabilities of multilayer feedforward neural networks," *Neural Networks*, **4**, 251-257.

Hornik, K., M. Stinchcombe, and H. White, (1989), "Multilayer feedforward networks are universal approximators," *Neural Networks*, **2**, 359-366.

Höskuldsson, Agnar, (1996), "Experimental design and priority PLS regression," *Journal of Chemometrics*, **10**, 637-668.

Höskuldsson, Agnar, (1995), "A combined theory for PCA and PLS," *Journal of Chemometrics*, **9**, 91-123.

Höskuldsson, Agnar, (1992), "The H-principle in modelling with applications to chemometrics," *Chemometrics and Intelligent Laboratory Systems*, **14**, 139-153.

Höskuldsson, Agnar, (1988), "PLS regression methods," *Journal of Chemometrics*, **2**, 211-228.

Hosmer, D.W., and S. Lemeshow, (1989), "Applied Logistic Regression," John Wiley & Sons, New York.

Hotelling, Harold, (1933), "Analysis of a complex of statistical variables into principal components," *The Journal of Educational Psychology*, 498-520.

Høy, Martin, Frank Westad, and Harald Martens, (2002), “Combining bilinear modeling and ridge regression,” *Journal of Chemometrics*, **16**, 313.318.

Høy, Martin, Kay Steen, and Harald Martens, (1998), "Review of partial least squares regression prediction error in Unscrambler," *Chemometrics and Intelligent Laboratory Systems*, **44**, 123-133.

Hwang, J.T. Gene, and A. Adam Ding, (1997), "Prediction intervals for artificial neural networks," *Journal of the American Statistical Association*, **92**, no. 438, 748-757.

Johnston, G.J., (1982), "Probabilities of maximal deviations for nonparametric regression function estimates," *Journal of Multivariate Analysis*, **12**, 402-414.

Jolliffe, I.T., (2002), "Principal Component Analysis," Springer-Verlag, New York.

Kauermann, Göran, Marlene Müller, and Raymond J. Carroll, (1998), "The efficiency of bias-corrected estimators for nonparametric kernel estimation based on local estimating equations," *Statistics and Probability Letters*, **37**, 41-47.

Knafl, G., J. Sacks, and D. Ylvisaker, (1985), "Confidence bands for regression functions," *Journal of the Royal Statistical Society Series B*, **27**, 251-263.

Knafl, G., J. Sacks, J. Spiegelman, and D. Ylvisaker, (1984), "Nonparametric calibration," *Technometrics*, **26**, 233-241.

Kolcz, Aleksander, and Nigel M. Allinson, (1999), "The general memory neural network and its relationship with basis function architectures, *Neurocomputing*, **29**, 57-84.

Kolcz, Aleksander, and Nigel M. Allinson, (1996), "N-tuple regression network," *Neural Networks*, **9**, no. 5, 855-869.

Kramer, M.A., (1992), "Autoassociative neural networks," *Comp. Chem. Engng.*, **16**, no. 4, 313-328.

Leonard, J.A., M.A. Kramer, and L.H. Ungar, (1992), "A neural network architecture that computes its own reliability," *Computers in Chemical Engineering*, **16**, no. 9, 819-835.

Levenberg, K., 1944, "A method for the solution of certain problems in least squares," *SIAM J. Numer. Anal.*, **16**, 588-604.

Liero, Hannelore, (1992), "Asymptotic normality of a weighted integrated squared error of kernel regression estimates with data-dependent bandwidth," *Journal of Statistical Planning and Inference*, **30**, 307-325.

Macauley, R.R., (1931), "The smoothing of Time Series," National Bureau of Economic Research, New York.

Malthouse, E.C., A.C. Tamhane, and R.S.H. Mah, (1997), "Nonlinear partial least squares," *Comp. Chem. Engng.*, **21**, no. 8, 875-890.

Marquardt, D.W., (1963), "An algorithm for least squares estimation of nonlinear parameters," *Journal of the Society for Industrial & Applied Mathematics*, **2**, 431-441.

Martens, Harald, and Magni Martens, (2000a), "Modified jack-knife estimation of parameter uncertainty in bilinear modelling by partial least squares regression (PLSR)," *Food Quality and Preference*, **11**, 5-16.

Martens, Harald, Garmt B. Dijksterhuis, and Derek V. Byrne, (2000b), "Power of experimental designs, estimated by Monte Carlo simulation," *Journal of Chemometrics*, **14**, 441-462.

Martens, H., and T. Næs, (1989), "Multivariate Calibration," John Wiley and Sons, Chichester.

Massart, D.L., B.G.M. Vandeginste, S.N. Deming, Y. Michotte, and L. Kaufman, (1988), "Chemometrics: A textbook," El Sevier Science Publishers, Amsterdam.

Masters, T., (1995), "Advanced algorithms for neural networks," John Wiley, New York.

Mathworks, (2003), MATLAB – Matrix Laboratory, www.mathworks.com.

McLachlan, G.J., (1992), "Discriminant Analysis and Statistical Pattern Recognition," Wiley, New York.

Moody, J., and C. Darken, (1989), "Fast learning in networks of locally-tuned processing units," *Neural Computation*, **1**, 281-294.

Morsing, Tomas, and Claes Ekman, (1998), "Short communication: Comments on construction of confidence intervals in connection with partial least squares," *Journal of Chemometrics*, **12**, 295-299.

Myers, R.H., (1986), "Classical and Modern Regression with Applications," Duxbury Press, Boston.

Nadaraya, E.A., (1964), "On estimating regression," *Theory Prob. Appl.*, **9**, 141-142.

Næs, Tormod, and Bjørn-Helge Mevik, (2001), "Understaning the collinearity problem in regression and discriminant analysis," *Journal of Chemometrics*, **15**, 413-426.

Næs, Tormod, and U. Indahl, (1998), "A unified description of classical classification methods for multicollinear data," *Journal of Chemometrics*, **12**, 205-220.

Ni, Yongnian, and Zhaohong Peng, (1995), "Determination of mixed ions by complexometric titration and nonlinear partial least squares calibration," *Analytica Chimica Acta*, **304**, 217-222.

Nychka, D., (1990), "The average posterior variance of a smoothing spline and a consistent estimate of average squared error," *Annals of Statistics*, **18**, 415-428.

Nychka, D., (1988), "Bayesian confidence intervals for smoothing splines," *Journal of the American Statistical Association*, **83**, 1134-1143.

Papadopoulos, Georgios, Peter J. Edwards, and Alan F. Murray, (2001), "Confidence estimation methods for neural networks: a practical comparison," *IEEE Transactions on Neural Networks*, **12**, no. 6, 1278-1287.

Park, J., and I.W. Sandberg, (1993), "Approximation and radial basis function networks," *Neural Computation*, 305-316.

Park, J., and I.W. Sandberg, (1991), "Universal approximation using radial basis function networks," *Neural Computation*, **3**, 246-257.

Pawlak, M., and U. Stadtmüller, (1997), "Kernel regression estimators for signal recovery," *Statistics and Probability Letters*, **31**, 185-198.

Pearson, Karl, (1901), "On lines and planes of closest fit to systems of points in space," *Philos. Mag.*, 559-573.

Phatak, A., P.M. Reilly, and A. Penlidis, (1993), "An approach to interval estimation in partial least squares regression," *Analytica Chimica Acta*, **277**, 495-501.

Qin, S. Joe, (1997), "Neural networks for intelligent sensors and control – practical issues and some solutins," 213-234.

Qin, S. Joe, and T.J. McAvoy, (1996), "Nonlinear FIR modeling via a neural net PLS approach," *Comp. Chem. Engng.*, **20**, no. 2, 147-159.

Qin, S. Joe, and Balu Rajagopal, (1993), "Combining statistics and expert systems with neural networks for empirical process modeling," *ISA*, 1711-1720.

Qin, S. Joe, and T.J. McAvoy, (1992), "Nonlinear PLS modeling using neural networks," *Comp. Chem. Engng.*, **16**, no. 4, 379-391.

Rasmussen, B., (2002), "Neural Network Partial Least Squares for Instrument Surveillance and Calibration Verification," MS Thesis, The University of Tennessee, Knoxville.

Rasmussen, Brandon, J. Wesley Hines, and Robert E. Uhrig,, (2000a), "A Novel Approach to Process Modeling for Instrument Surveillance and Calibration Verification," The Third American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation and Control and Human-Machine Interface Technologies, Washington DC, November 13-17.

Rasmussen, Brandon, J. Wesley Hines, and Robert E. Uhrig, (2000b), "Nonlinear Partial Least Squares Modeling for Instrument Surveillance and Calibration Verification," *Proceedings of the Maintenance and Reliability Conference*,, Knoxville, TN, May 7-10.

Reed, Russell D., and Robert J. Marks II, (1999), "Neural Smithing Supervised Learning in Feedforward Artificial Neural Networks," MIT Press, Cambridge, MA.

Rosenblatt, F., (1962), Principles in Neurodynamics, Spartan Books, Washington D.C.

Ruppert, D., and M.P. Wand, (1994), "Multivariate locally weighted least squares regression," *Annals of Statistics*, **22**, 1346-1370.

Ruscio, David Di, (2000), "A weighted view on the partial least-squares algorithm," *Automatica*, **36**, 831-850.

Sarle, Warren S., (1994), "Neural networks and statistical models," *Proceedings of the 19th Annual SAS Users Group International Conference*.

Scarselli, Franco, and Ah Chung Tsoi, (1998), "Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results," *Neural Networks*, **11**, no. 1, 15-37.

Schiøler, H., and U. Hartmann, (1992), "Mapping neural networks derived from the Parzen window estimator," *Neural Networks*, **5**, 903-909.

Seber, G.A.F., and C.J. Wild, (1989), "Nonlinear Regression," Wiley, New York.

Shao, R., E.B. Martin, J, Zhang, and A.J. Morris, (1997), "Confidence bounds for neural network representations," *Comp. Chem. Engng.*, **21**, Supplement, S1173-S1178.

Singer, R.M., K.C. Gross, J.P. Herzog, R.W. King, and S.W. Wegerich, (1996), "Model-based nuclear power plant monitoring and fault detection: theoretical foundations," *Proceedings of the 9th International Conference on Intelligent Systems Applications to Power Systems*, Seoul Korea.

Specht, D.F., and H. Romsdahl, (1994), "Experience with adaptive probabilistic neural networks and adaptive general regression neural networks," *Proceedings of the IEEE World Congress on Computational Intelligence*, **2**, 1203-1208.

Specht, D.F., (1991), "A general regression neural network," *IEEE Transactions on Neural Networks*, **2**, 568-576.

Specht, D.F., (1990), "Probabilistic neural networks," *Neural Networks*, **3**, 109-118.

Steinbuch, K., and U.A.W. Piske, (1963), "Learning matrices and their applications," *IEEE Transactions Electronic Computing*, EC-12, 846-862.

Stone, Charles J., (1982), "Optimal rates of convergence for nonparametric regression," *Ann. Statistic.*, **10**, 1040-1053.

Stone, Charles J., (1980), "Optimal rates of convergence for nonparametric estimators," *Ann. Statist.*, **8**, 1348-1360.

Stone, Charles J., (1977), "Consistent nonparametric regression," *Annals of Statistics*, **5**, no. 4, 595-620.

Sundberg, Rolf, (2000), "Aspects of statistical regression in sensometrics," *Food Qualtiy and Preference*, **11**, 17-26.

Taylor, W.K., (1959), "Pattern recognition by means of automatic analogue apparatus," *Proc. Instit. Electrical Engineers*, 106B, 198-209.

Taylor, W. K., (1960), "A parallel analogue reading machine," *Control*, **3**, 95-99.

Tibshirani, Robert, (1996), "A comparison of some error estimates for neural network models," *Neural Computation*, **8**, 152-163.

Tikhonov, A.N., and V.Y. Arsenin, (1997), "Solution of ill-posed problems," Winston, Washington D.C.

Tomandl, Dirk, and Andreas Schober, (2001), "A modified general regression neural network (MGRNN) with new, efficient training algorithms as a robust black-box tool for data analysis," *Neural Networks*, **14**, 1023-1034.

Upadhyaya, Belle R., and Evren Eryurek, (1992), "Application of neural networks for sensor validation and plant monitoring," *Nuclear Technology*, **97**, 170-176.

Vapnik, V.N., (1998), "Statistical Learning Theory," John Wiley & Sons, NY.

Wahba, G., (1983), "Bayesian confidence intervals for the cross-validated smoothing spline," *J. Roy. Statist. Soc.* B, **45**, 133-150.

Wand, and Jones, (1995), Kernel Smoothing, Monographs on Statistics and Applied Probability, Chapman & Hall, London.

Wansink, Geerteke, and Luren Yang, (2001), "A new confidence bound estimation method for neural networks, an application example," *EAGE 63rd Conference and Technical Exhibition*, Amsterdam, Netherlands, June 11-15, 1-4.

Watson, G.S., (1964), "Smooth Regression Analysis," *Sankhya* Ser. A, **26**, 359-372.

Wegelin, Jacob A., (2000), "A survey of partial least squares (PLS) methods, with emphasis on the two-block case," University of Washington, Seattle, March 9, 2000, Technical Report No. 371.

Wehrens, R., and W.E. Van Der Linden, (1997), "Bootstrapping principal component regression models," *Journal of Chemometrics*, **11**, 157-171.

Weisberg, S., (1985), "Applied Linear Regression," John Wiley & Sons, New York.

Weiss, S.M., and C.A. Kulikowski, (1991), "Computer Systems that Learn," Morgan Kaufmann, San Mateo, CA.

Werbos P.J., (1974), Beyond Regression: new tools for prediction and analysis in the behavioral sciences, PhD thesis, Harvard University, Cambridge, MA.

White, H., (1992), "*Artificial Neural Networks: Approximation and Learning Theory*, Blackwell, Oxford.

White, Halbert, (1989), "Some asymptotic results for learning in single hidden-layer feedforward network models," *Journal of the American Statistical Association*, **84**, no. 408,1003-1013.

Wilson, D.J.H., G. W. Irwin, and G. Lightbody, (1997a), "Nonlinear PLS modelling using radial basis functions," *Proceedings of the American Control Conference*, Albuquerque, NM, 3275-3276.

Wilson, D.J.H., G.W. Irwin, and G. Lightbody, (1997b), "Neural networks and multivariate SPC," *Institution of Electrical Engineers*, 5/1-5/5

Wold, Svante, (1992), "Nonlinear partial least squares modelling II. spline inner relationship," *Chemometrics and Intelligent Laboratory Systems*, **14**, 71-84.

Wold, Svante, Nouna Kettaneh-Wold, and Bert Skagerberg, (1989), "Nonlinear PLS modeling," *Chemometrics and Intelligent Laboratory Systems*, **7**, 53-65.

Wold, S., A. Ruhe, H. Wold, and W.J. Dunn III, (1984), "The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses," *SIAM J. Sci. Stat. Comput.*, **5**, no. 3, 735-743.

Wold, H., (1966), "Nonlinear Estimation by Iterative Least Squares Procedures," F. N. David, Ed., John Wiley, New York, 1966.

Woodward, A.M., et. al. (1996), *Bioelectrochemistry and Bioenergetics*, **40**, 99-132.

Wu, J.S., and C.K. Chu, (1994), "Nonparametric estimation of a regression function with dependent observations," *Stochastic Processes and their Applications*, **50**, 149-160.

Xu, Xiao, (2000), Automated neural network-based instrument validation system, PhD Dissertation, The University of Tennessee, Knoxville, TN.

Xu, X., J.W. Hines, and R.E. Uhrig, (1999), "Sensor Validation and fault detection using neural networks," *Proceedings of the Maintenance and Reliability Conference*, Gatlinburg, TN May.

Yang, Luren, Tom Kavli, Mats Carlin, Sigmund Clausen, and Paul F.M. de Groot, (2000), "An evaluation of confidence bound estimation methods for neural networks," *ESIT*, September 14-15, Aachen, Germany, 322-329.

Ye, Nong, Qiu Zhong, and Gregory E. Rahn, (2000), "Confidence assessment of quality prediction from process measurement in sequential manufacturing processes," *IEEE Transactions on Electronics Packaging Manufacturing*, **23**, no. 3, 177-184.

Ziegler, Klaus, (2002), "On the asymptotic normality of kernel regression estimators of the mode in nonparametric random design model," *Journal of Statistical Planning and Inference*, In Press.

APPENDIX

The majority of the functions implemented for this work utilize the functions of the standard MATLAB software package. A few functions from the Neural Network Toolbox, and the Statistics toolbox were also used. All of the functions used in this work are attached here for the interested readers and programmers. Efforts were made to document the code and eliminate unnecessary computations; however, these functions underwent constant modification and the existence of remnants from previous versions may appear. The main purpose of including all of these functions is so that the exact computations employed can be studied by those wishing to apply these techniques.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% UIANNboot2.m IS THE MAIN FUNCTION FOR CREATING INFERENTIAL ANNS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS FUNCTION RETURNS ESTIMATIONS FOR THE THREE SETS OF DATA SUPPLIED AT THE INPUT
% THE POINT-WISE PREDICTION INTERVALS ARE ALSO RETURNED

function [TRAINU,VALU,TESTU,YHATTRAIN,YHATVAL,YHATTEST]=UIANNboot2(n1min,n1max,XT,XV,YT,YV,X,Y,im,pp,num_it);

% n1min is the minimum number of hidden neurons to evaluate
% n1max is tyhe maximum number of hidden neurons to evaluate
% XT training predictors (nxd)
% YT training responses (nx1)
% XV validation predictors (pxd)
% YV validation responses (px1)
% X test predictors (mxd)
% Y test response (mxd)
% im is the neural network initialization method (see iann4boot5.m)
%pp=probability, or significance level for prediction intervals (usually 95%)
%num_it=number of iterations to complete per number of hidden neurons

% define the number of possible architectures as nn=n1max-n1min+1;

% YHATTRAIN is a (num_it x n x nn) matrix of estimates for the training responses
% YHATVAL is a (num_it x p x nn) matrix of estimates for the validation responses
% YHATTEST is a (num_it x m x nn) matrix of estimates for the test responses
% TRAINU is a (num_it x n x nn) matrix of point-wise prediction intervals for the training response estimations
% VALU is a (num_it x p x nn) matrix of point-wise prediction intervals for the validation response estimations
% TESTU is a (num_it x m x nn) matrix of point-wise prediction intervals for the test response estimations

ntrain=size(XT,1);
n0=size(XT,2);

for n1=n1min:n1max; % # OF HIDDEN NEURONS INDEX

   for j=1:num_it; % ITERATION INDEX
      % CALL MAIN SUBFUNCTION TO COMPUTE ESTIMATES AND PREDICTION INTERVALS
      [yhatTRAIN(j,:),yhatTEST(j,:),yhatVAL(j,:),trainunc(j,:),valunc(j,:),testunc(j,:)]=iann4boot5(XT,XV,YT,YV,X,Y,n1,im,pp);
   end;

   % COMPILE DATA FOR OUTPUT
   TRAINU(:,:,n1)=trainunc;
   VALU(:,:,n1)=valunc;
   TESTU(:,:,n1)=testunc;
   YHATTRAIN(:,:,n1)=yhatTRAIN;
   YHATVAL(:,:,n1)=yhatVAL;
   YHATTEST(:,:,n1)=yhatTEST;

   % CLEAR UNNEEDED VARIABLES
  clear W W2 b1 b2 trainunc valunc testunc TRAINCOVERAGE VALCOVERAGE TESTCOVERAGE yhatTRAIN yhatVAL yhatTEST;

end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SUBFUNCTION OF UIANNboot2.m FOR OBTAINING ANN ESTIMATES AND POINT-WISE PREDICTION INTERVALS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS FUNCTION ACCEPTS 3 DATA SETS AND PROVIDES ESTIMATIONS AND POINT-WISE PREDICTION INTERVALS\\
% FOR THE UNIVARIATE RESPONSE OF ALL 3 DATA SETS

function [yhatTRAIN,yhatTEST,yhatVAL,trainunc,valunc,testunc]=iann4boot5(NXT,NXV,YT,YV,NX,Y,n1,im,pp);

% NXT training predictors (nxd)
% YT training responses (nx1)
% NXV validation predictors (pxd)
% YV validation responses (px1)
% NX test predictors (mxd)
```

```
% Y test response (mxd)
% n1 specifies the number of hidden neurons
% im is the ANN initialization method.
%   the hidden layer weights and biases are always randomly initialized.
%   the im options refer to the output layer weights and biases
%   im=0, random initialization for output layer parameters
%   im=1, output layer weights set to (standard deviation(YT) / n1), output bias set to mean(YT)
%   im=2, output layer weights and biases initialized to the OLS solution of the output of the hidden layer to the target output
% pp=probability, or significance level for prediction intervals (usually 95%)

% yhatTRAIN is a (1 x n) matrix of estimates for the training responses
% yhatVAL is a (1 x p) matrix of estimates for the validation responses
% yhatTEST is a (1 x m) matrix of estimates for the test responses
% trainunc is a (1 x n) matrix of point-wise prediction intervals for the training response estimations
% valunc is a (1 x p) matrix of point -wise prediction intervals for the validation response estimations
% testunc is a (1 x m) matrix of point-wise prediction intervals for the test response estimations

 [m,n0]=size(NXT); % SETS m TO THE NUMBER OF TRAINING SAMPLES AND n0 TO THE NUMBER OF INPUTS
 n2=1; % SETS NUMBER OF OUTPUT NEURONS TO 1
 % SET ANN PARAMETERS
 nx=[(min(NXT)') (max(NXT)')];
 net=newff(nx,[n1 n2],{'tansig' 'purelin'},'trainlm');
 net.trainParam.epochs=5000;
 net.trainParam.show=500;

 % INITIALIZE ANN WEIGHTS AND BIASES
 net=init(net);
 [a1,a2]=(size(net.IW{1,1}));
 net.IW{1,1}=rands(a1,a2);
 [a1,a2]=(size(net.LW{2,1}));
 net.LW{2,1}=rands(a1,a2);
 [a1,a2]=(size(net.b{1}));
 net.b{1}=rands(a1,a2);
 [a1,a2]=(size(net.b{2}));
 net.b{2}=rands(a1,a2);

  % Initialization method
  if im==0;
     %fprintf('Random Initialization\n\n')
  elseif im==1;
     net.LW{2,1}=ones(1,n1)*(std(YT)/n1);
     net.b{2}=mean(YT);
     %fprintf('Mean / Std Initialization\n\n');
  elseif im==2;
     %fprintf('OLS Initialization\n\n');
     W=net.IW{1,1}';
     W2=net.LW{2,1}';
     b1=net.b{1}';
     b2=net.b{2};
     for i=1:size(NXT,1);
       x=NXT(i,:);
       for r=1:n1;
          outnn(r,i)=tansig(x*W(:,r)+b1(r));
       end;
     end;
     OUT=[ones(m,1) outnn'];
     OLS_b=inv(OUT'*OUT)*(OUT'*YT);
     net.LW{2,1}=OLS_b(2:length(OLS_b))';
     net.b{2}=OLS_b(1);
  end;

  % SPECIFY VALIDATION DATA AND TRAIN NETWORK
  VV.P=NXV'; VV.T=YV';
  [net,tr]=train(net,NXT',YT',[],[],VV);

  % PRODUCE ESTIMATIONS AND ERRORS
  yhatTRAIN=sim(net,NXT');
  trainerr=YT-yhatTRAIN';
  yhatTEST=sim(net,NX');
  testerr=Y-yhatTEST';
  yhatVAL=sim(net,NXV');
  valerr=YV-yhatVAL';

   % EXTRACT WEIGHTS AND BIASES FROM NET STRUCTURE
   W=net.IW{1,1}';
   W2=net.LW{2,1}';
   b1=net.b{1}';
   b2=net.b{2};
```

```
% CALCULATE JACOBIAN MATRIX WITH TRAINING DATA

X=NXT;
for i=1:size(X,1);
  II1(i,:)=X(i,:)*W+b1;
  EE1(i,:)=( 2 ./ (1+exp(-2*II1(i,:)))   )-1;
  B1(i,:)=(1-(II1(i,:).^2))*diag(W2);
  dydthet=[];

  for j=1:(n0+1);
      if j<=n0;
        dydthet=[dydthet (X(i,j)*B1(i,:))];
      else
        dydtheta(i,:)=[dydthet B1(i,:) EE1(i,:) 1];
      end;
 end;
end;


% CALCULATION OF JACOBIAN VECTORS FOR TEST DATA
  PPINV=pinv(dydtheta'*dydtheta);
  for i=1:size(NX,1);
    x=NX(i,:);

    for r=1:n1;
       out(r)=tansig(x*W(:,r)+b1(r));
    end;

    fout(i)= (out*W2)+b2;
   yI1=(x*W)+b1;
   yE1=( 2 ./ (1+exp(-2*yI1))   )-1;
   yB1=(1-(yI1.^2))*diag(W2);
   ydydthet=[];
   for j=1:(n0+1);
        if j<=n0;
          ydydthet=[ydydthet (x(j)*yB1)];
        else
          f(:,i)=[ydydthet yB1 yE1 1]';
        end;
   end;
     Ff(i)= f(:,i)'*PPINV*f(:,i);
 end;


 % OBTAIN ESTIMATE FOR STD DEVIATION OF ESTIMATES
N=(n0*n1)+(2*n1)+1; % TOTAL NUMBER OF ANN PARAMETERS
 TV=tinv(pp,(m-N)); % CRITICAL VALUE FROM T DISTRIBUTION
 s2=sqrt(mean([(trainerr.^2);(valerr.^2)]));
 % COMPUTE PREDICTION INTERVALS FOR TEST DATA
 testunc='TV*  sqrt((s2^2) + ((s2^2)*Ff));


% CALCULATION OF JACOBIAN VECTORS FOR TRAINING DATA
  for i=1:size(NXT,1);
    xx=NXT(i,:);

    for r=1:n1;
       tout(r)=tansig(xx*W(:,r)+b1(r));
    end;
   tyI1=(xx*W)+b1;
   tyE1=( 2 ./ (1+exp(-2*tyI1))   )-1;
   tyB1=(1-(tyI1.^2))*diag(W2);
   tydydthet=[];
   for j=1:(n0+1);
        if j<=n0;
          tydydthet=[tydydthet (xx(j)*tyB1)];
        else
          tf(:,i)=[tydydthet tyB1 tyE1 1]';
        end;
   end;
    tFf(i)= tf(:,i)'*PPINV*tf(:,i);
 end;

 % COMPUTE PREDICTION INTERVALS FOR TRAINING DATA
 trainunc='TV*sqrt((s2^2) + ((s2^2)*tFf));
```

```
    % CALCULATION OF JACOBIAN VECTORS FOR VALIDATION DATA
     for i=1:size(NXV,1);
       xx=NXV(i,:);
       for r=1:n1;
          tout(r)=tansig(xx*W(:,r)+b1(r));
       end;
       vyI1=(xx*W)+b1;
       vyE1=( 2 ./ (1+exp(-2*vyI1))   )-1;
       vyB1=(1-(vyI1.^2))*diag(W2);
       vydydthet=[];
       for j=1:(n0+1);
           if j<=n0;
              vydydthet=[vydydthet (xx(j)*vyB1)];
           else
              vf(:,i)=[vydydthet vyB1 vyE1 1]';
           end;
       end;
        vFf(i)= vf(:,i)'*PPINV*vf(:,i);
     end;
     % COMPUTE PREDICTION INTERVALS FOR VALIDATION DATA
     valunc=TV*sqrt((s2^2) + ((s2^2)*vFf));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OPTIMIZED KERNEL REGRESSION PROGRAM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function is an optimized version of function LWR9999v2 to expedite computations
% for the case of p=0.  The increased speed is due to the elimination of the matrix inversions
% of the standard matrix locally weighted regression estimator and variance equations.

function [TRAINVAR,VALVAR,TESTVAR,YHATTRAIN,YHATVAL,YHATTEST,bias]=KR(XT,YT,XV,YV,X,Y,h_KR,V,sig);

% p is the degree of the local polynomial (CAN ONLY BE SET TO 0, or 1)
% XT training predictors (nxd)
% YT training responses (nx1)
% XV validation predictors (pxd)
% YV validation responses (px1)
% X test predictors (mxd)
% Y test response (mxd)
% h_KR is a 1xz vector of bandwidths. Only global bandwidths are applied
%   in this program, so for each bandwidth an estimation will be obtained for each query point.
% V is a global conditional variance, [Var(Y|x=x(i))]
% sig is used for display purposes only to indicate progress for the case of more than one signal

% TRAINVAR is the (nxz) matrix of variance estimates corresponding to the training responses YT,
%   the z columns correspond to the z different bandwidths
% VALVAR is the (pxz) matrix of variance estimates corresponding to the validation responses YV,
%   the z columns correspond to the z different bandwidths
% TESTVAR is the (mxz) matrix of variance estimates corresponding to the test responses Y,
%   the z columns correspond to the z different bandwidths
% YHATTRAIN is the (nxz) matrix of estimates of the training responses YT.
% YHATVAL is t he (pxz) matrix of estimates of the validation responses YV.
% YHATTEST is the (mxz) matrix of estimates of the test responses Y.
% bias is a (1xz) vector of bias estimates

[N,ni]=size(XT);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OBTAIN RESPONSE ESTIMATES AND VARIANCE ESTIMATES FOR VALIDATION DATA

t_new=XV; % DEFINES THE VALIDATION DATA AS THE QUERY DATA

% INITIALIZE MATRICES
YHATVAL=zeros(length(t_new),length(h_KR));
VALVAR=zeros(length(t_new),length(h_KR));

for k=1:length(h_KR); % BANDWIDTH INDEX
   fprintf('validation');bandwidth=h_KR(k),sig=sig %DISPLAY PROGRESS
   for n=1:length(t_new); %SAMPLE INDEX
      x=t_new(n,:);
     % SET UP TRAINING DATA MATRIX
      XX=ones(N,1);
      TT=XT-(ones(N,1)*x);
      % COMPUTE GAUSSIAN KERNEL WEIGHTS
      KH=prod(((1/h_KR(k))*exp((-.5*(TT./h_KR(k)).^2))),2);
     % OBTAIN VALIDATION DATA RESPONSE ESTIMATES
      YHATC=KH'/sum(KH);
```

```
        YHATVAL(n,k)=YHATC*YT;
        % ESTIMATE VARIANCE FOR VALIDATION DATA
        VALVAR(n,k)=YHATC*YHATC'*V(1,1);
  end;
end;

% END VALIDATION DATA ESTIMATION STEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OBTAIN RESPONSE ESTIMATES AND VARIANCE ESTIMATES FOR TRAINING DATA

t_new=XT; % DEFINES THE TRAINING DATA AS THE QUERY DATA
% INITIALIZE MATRICES
YHATTRAIN=zeros(length(t_new),length(h_KR));
TRAINVAR=zeros(length(t_new),length(h_KR));

for k=1:length(h_KR); % BANDWIDTH INDEX
    fprintf('training');bandwidth=h_KR(k),sig=sig %DISPLAY PROGRESS
    for n=1:length(t_new); % SAMPLE INDEX
        x=t_new(n,:);
        % SET UP TRAINING DATA MATRIX
        XX=ones(N,1);
        TT=XT-(ones(N,1)*x);
        % COMPUTE GAUSSIAN KERNEL WEIGHTS
        KH=prod(((1/h_KR(k))*exp((-.5*(TT./h_KR(k)).^2))),2);
        % OBTAIN TRAINING DATA RESPONSE ESTIMATES
        YHATC=KH'/sum(KH);
        YHATTRAIN(n,k)=YHATC*YT;
        % ESTIMATE VARIANCE FOR TRAINING DATA
        TRAINVAR(n,k)=YHATC*YHATC'*V(1,1);
  end;
end;

% END TRAINING DATA ESTIMATION STEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OBTAIN RESPONSE ESTIMATES AND VARIANCE ESTIMATES FOR TEST DATA

t_new=X; % DEFINES THE TEST DATA AS THE QUERY DATA
% INITIALIZE MATRICES
YHATTEST=zeros(length(t_new),length(h_KR));
TESTVAR=zeros(length(t_new),length(h_KR));

for k=1:length(h_KR); % BANDWIDTH INDEX
   fprintf('test');bandwidth=h_KR(k),sig=sig % DISPLAY PROGRESS
    for n=1:length(t_new); % SAMPLE INDEX
        x=t_new(n,:);
        % SET UP TRAINING DATA MATRIX
        XX=ones(N,1);
        TT=XT-(ones(N,1)*x);
        % COMPUTE GAUSSIAN KERNEL WEIGHTS
        KH=prod(((1/h_KR(k))*exp((-.5*(TT./h_KR(k)).^2))),2);
        % OBTAIN TEST DATA RESPONSE ESTIMATES
        YHATC=KH'/sum(KH);
        YHATTEST(n,k)=YHATC*YT;
        % ESTIMATE VARIANCE FOR TEST DATA
        TESTVAR(n,k)=YHATC*YHATC'*V(1,1);
    end;
end;

% END TEST DATA ESTIMATION STEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ESTIMATE BIAS BASED ON TRAINING AND VALIDATION RESPONSE ESTIMATES
for k=1:length(h_KR);
    bias(k)=mean([((YV-YHATVAL(:,k)).^2);((YT-YHATTRAIN(:,k)).^2)]);
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Locally weighted Regression Program for p=0, and p=1.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [TRAINVAR,VALVAR,TESTVAR,YHATTRAIN,YHATVAL,YHATTEST,bias]=LWR9999v2(p,XT,YT,XV,YV,X,Y,h_KR,V);
```

```
% p is the degree of the local polynomial (CAN ONLY BE SET TO 0, or 1)
% XT training predictors (nxd)
% YT training responses (nx1)
% XV validation predictors (pxd)
% YV validation responses (px1)
% X test predictors (mxd)
% Y test response (mxd)
% h_KR is a 1xz vector of bandwidths. Only global bandwidths are applied
%   in this program, so for each bandwidth an estimation will be obtained for each query point.
% V is a global conditional variance, [Var(Y|x=x(i))]

% TRAINVAR is the (nxz) matrix of variance estimates corresponding to the training responses YT,
%   the z columns correspond to the z different bandwidths
% VALVAR is the (pxz) matrix of variance estimates corresponding to the validation responses YV,
%   the z columns correspond to the z different bandwidths
% TESTVAR is the (mxz) matrix of variance estimates corresponding to the test responses Y,
%   the z columns correspond to the z different bandwidths
% YHATTRAIN is the (nxz) matrix of estimates of the training responses YT.
% YHATVAL is the (pxz) matrix of estimates of the validation responses YV.
% YHATTEST is the (mxz) matrix of estimates of the test responses Y.
% bias is a (1xz) vector of bias estimates

if p>2;
   error('Only set up for p=0, or p=1')
end;

[N,ni]=size(XT);

% Define E1 vector based on degree of local polynomial
    if p==0;
        E1=1;
    elseif p==1;
        E1=[zeros(ni+1,1)];
        E1(1)=1;
    end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OBTAIN RESPONSE ESTIMATES AND VARIANCE ESTIMATES FOR VALIDATION DATA

t_new=XV; % DEFINES THE VALIDATION DATA AS THE QUERY DATA

% INITIALIZE MATRICES
YHATVAL=zeros(length(t_new),length(h_KR));
VALVAR=zeros(length(t_new),length(h_KR));

for k=1:length(h_KR); % BANDWIDTH INDEX
    fprintf('validation');bandwidth=h_KR(k),p=p % DISPLAY PROGRESS
    for n=1:length(t_new); % SAMPLE INDEX
        x=t_new(n,:);
        % SET UP TRAINING DATA MATRIX
        XX=ones(N,1);% DEFINES DEFAULT FOR p=0
        if p==1;
            for jj=1:p;
                XX(:,(ni*(jj-1)+2):(ni*jj+1))=[(XT-(ones(N,1)*x)).^jj];
            end;
        end;
       % COMPUTE GAUSSIAN KERNEL WEIGHTS AND FORM WEIGHT MATRIX
       TT=XT-(ones(N,1)*x);
       KH=prod(((1/h_KR(k))*exp((-.5*(TT./h_KR(k)).^2))),2);
       WX=diag(KH);
      % OBTAIN VALIDATION ESTIMATES
      YHATC=E1'*pinv(XX'*WX*XX)*XX'*WX;
      YHATVAL(n,k)=YHATC*YT;
      % OBTAIN VARIANCE ESTIMATE
      VALVAR(n,k)=YHATC*YHATC'*V(1,1);
  end;
end;

% END VALIDATION DATA ESTIMATION STEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OBTAIN RESPONSE ESTIMATES AND VARIANCE ESTIMATES FOR TRAINING DATA

t_new=XT; % DEFINES THE TRAINING DATA AS THE QUERY DATA

% INITIALIZE MATRICES
YHATTRAIN=zeros(length(t_new),length(h_KR));
TRAINVAR=zeros(length(t_new),length(h_KR));
```

```
for k=1:length(h_KR); % BANDWIDTH INDEX
   fprintf('training');bandwidt h=h_KR(k), p=p % DISPLAY PROGRESS
   for n=1:length(t_new); % SAMPLE INDEX
      x=t_new(n,:);
      % SET UP TRAINING DATA MATRIX
      XX=ones(N,1);% DEFINES DEFAULT FOR p=0
      if p==1;
         for jj=1:p;
            XX(:,(ni*(jj-1)+2):(ni*jj+1))=[(XT-(ones(N,1)*x)).^jj];
         end;
      end;
     % COMPUTE GAUSSIAN KERNEL WEIGHTS AND FORM WEIGHT MATRIX
     TT=XT-(ones(N,1)*x);
     KH=prod(((1/h_KR(k))*exp((-.5*(TT./h_KR(k)).^2))),2);
     WX=diag(KH);
     % OBTAIN TRAINING ESTIMATES
     YHATC=E1'*pinv(XX'*WX*XX)*XX'*WX;
     YHATTRAIN(n,k)=YHATC*YT;
     % OBTAIN VARIANCE ESTIMATES
     TRAINVAR(n,k)=YHATC*YHATC'*V(1,1);
  end;
end;

% END TRAINING DATA ESTIMATION STEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OBTAIN RESPONSE ESTIMATES AND VARIANCE ESTIMATES FOR TEST DATA

t_new=X; % DEFINES THE TRAINING DATA AS THE QUERY DATA

% INITIALIZE MATRICES
YHATTEST=zeros(length(t_new),length(h_KR));
TESTVAR=zeros(length(t_new),length(h_KR));

for k=1:length(h_KR); % BANDWIDTH INDEX
  fprintf('test');bandwidth=h_KR(k),p=p % DISPLAY PROGRESS
   for n=1:length(t_new); % SAMPLE INDEX
      x=t_new(n,:);
      XX=ones(N,1);% DEFINES DEFAULT FOR p=0
      % SET UP TRAINING DATA MATRIX
      if p==1;
         for jj=1:p;
            XX(:,(ni*(jj-1)+2):(ni*jj+1))=[(XT-(ones(N,1)*x)).^jj];
         end;
      end;
     % COMPUTE GAUSSIAN KERNEL WEIGHTS AND FORM WEIGHT MATRIX
     TT=XT-(ones(N,1)*x);
     KH=p rod(((1/h_KR(k))*exp((-.5*(TT./h_KR(k)).^2))),2);
     WX=diag(KH);
     % OBTAIN TEST ESTIMATES
     YHATC=E1'*pinv(XX'*WX*XX)*XX'*WX;
     YHATTEST(n,k)=YHATC*YT;
     % OBTAIN VARIANCE ESTIMATES
     TESTVAR(n,k)=YHATC*YHATC'*V(1,1);
  end;
end;
% END TEST DATA ESTIMATION STEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ESTIMATE BIAS BASED ON TRAINING AND VALIDATION RESPONSE ESTIMATES
for k=1:length(h_KR);
   bias(k)=mean([((YV-YHATVAL(:,k)).^2);((YT-YHATTRAIN(:,k)).^2)]);
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NNPLS MAIN FUNCTION FOR RESPONSE VARIABLE ESTIMATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [YPT,YPV,YPTS,TRAINU,VALU,TESTU,factor,num_hidden]=nnplsi_main77f(TRAIN,VALIDATE,TEST,num_it,im,maxfactor);

% TRAIN is an [nx(d+1)] matrix of training data, where the first n columns are the predictor variables,
%   and the last column contains the corresponding responses.
% VALIDATE is an [px(d+1)] matrix of validation data, where the first n columns are the predictor variables,
%   and the last column contains the corresponding responses.
% TEST is an [mx(d+1)] matrix of validation data, where the first n columns are the predictor variables,
%   and the last column contains the corresponding responses.
% num_it is the number of iterations to carry out
% im is the neural network initialization method see plsnet8if.m function below
```

```
% maxfactor is the maximum number of latent variables to evaluate

% YPT is a [n x maxfactor] matrix of the training response estimates
% YPV is a [p x maxfactor] matrix of the validation response estimates
% YPTS is a [m x maxfactor] matrix of the test response estimates
% TRAINU is a [n x maxfactor] matrix of the point-wise prediction interval estimates for YPT
% VALU is a [p x maxfactor] matrix of the point-wise prediction interval estimates for YPV
% TESTU is a [m x maxfactor] matrix of the point-wise prediction interval estimates for YPTS
% factor is the maximum possible dimension of the NNPLS model
% num_hidden is a [1 x maxfactor] vector returning the number of hidden neurons used for each of the latent
%   variables form 1 to maxfactor.

for it=1:num_it; % ITERATION INDEX

for h=size(TRAIN,2):size(TRAIN,2);  % SETS THE SIGNAL INDEX H TO [d+1], SO THAT ESTIMATES ARE OBTAINED FOR
                          % THE APPROPRIATE RESPONSE VARIABLE LOCATED IN THIS COLUMN

   % The nnpls_create5f function is the primary function which creates the model,
   % and returns the corresponding weights, etc.

   [factor,p,w,wbs,T,num_hidden]=nnplsi_create5f(TRAIN,VALIDATE,maxfactor,h,im);

   % The nnplsi_predictall function is the primary function for obtaining estimates and
   % point-wise prediction intervals

   [ypt,ypv,ypts,trainunc,valunc,testunc] = nnplsi_predictall(TRAIN,VALIDATE,TEST,w,wbs,p,maxfactor);


end;

% PREPARE DATA FOR OUTPUT
YPT(:,:,it)=ypt;
YPV(:,:,it)=ypv;
YPTS(:,:,it)=ypts;
VALU(:,:,it)=valunc;
TRAINU(:,:,it)=trainunc;
TESTU(:,:,it)=testunc;

end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SUBFUNCTION OF nnplsi_main77f.m FOR PRODUCING MODEL PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [factor,p,w,WBS,T,num_hidden]=nnplsi_create5f(train,validate,maxfactor,h,im);

% This function creates an NNPLS model for the hth signal in the data
% sets train.  It returns the input loadings, transformation weights,
% neural network weights and biases,it calls the function nnpls_outer
% to perform the transformations of the outer relationships

% train, and validate are the TRAIN and VALIDATE matrices input to nnplsi_main77f.m
% maxfactor is the maximum number of factors (latent vectors), passed from nnplsi_main77f.m
% h is the signal index specifying which of the d signals to develop a model for.
% im is the neural network initilalization method, see plsnet8if.m


% factor is the maximum possible dimension of the NNPLS model
% p = input score vectors
% w = input transformation weights
% WBS = the neural network weights and biases
% T are the input latent variables for the training data
% num_hidden is a [1 x maxfactor] vector returning the number of hidden neurons used for each of the latent
%   variables form 1 to maxfactor.


% Separate data into training inputs (tx), training output (ty),
%               validation inputs(vx), and validation output (vy).
[tx,ty]=divide(train,h);
[vx,vy]=divide(validate,h);

% This function creates the outer and inner relationship models
[p,w,WBS,T] = nnplsi_outer5f(tx,ty,vx,vy,maxfactor,im);

% Extract the number of hidden neurons for each latent variable from the neural network parameters matrix
for n=1:maxfactor;
```

```
  num_hidden(n)=WBS(1,1,n); % number of nodes per component
end;

% Set maximum possible NNPLS model dimension
factor=size(tx,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SUBFUNCTION OF nnplsi_main77f.m FOR CONSTRUCTING NNPLS OUTER RELATIONSHIPS AND MAPPING INNER RELATIONHSIPS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [p,w,WBS,t] = nnplsi_outer5f(x,y,vx,vy,maxrank,im)

% This function carries out the NNPLS-1 algorithm.
% It returns the input loadings, transformation weights,
% and neural network weights and biases.
% It calls the function plsnet8if.m to initialize and train the neural networks
%   for the inner relationships.

%  x       is the matrix of predictor variables (inputs)
%  y       is the response  variable (output)
%  vx                is the matrix of predictor variable validation data
%  vy      is the response variable validation data
%  maxrank  is the limit to the number of latent variables calculated
%
%  * - vx and vy are used in cross-validation neural network training - *

%           p       is the matrix of input score vectors
%           w       is the matrix of weights
%           wbs     is the vector containing the neural network parameters
%  t       is the matrix of input latent variables corresponding to (x,y)

% SET MAXIMUM NUMBER OF HIDDEN NEURONS PER LATENT VARIABLE
maxnhn=10;
% INITIALIZE NEURAL NETWORK PARAMETER MATRIX
WBS(:,:,1)=zeros(maxnhn+1,4);

for h = 1:maxrank; % LATENT VARIABLE INDEX
  u(:,h)=y; % INITIALIZE TRAINING OUTPUT VECTOR
  w(:,h)=(x'*y)/(sqrt((y'*x)*(x'*y))); % CALCULATE TRANSFORMATION WEIGHTS
  t(:,h)=x*w(:,h); % CALCULATE INPUT LATENT VARIABLES FOR TRAINING DATA
  p(h,:)=(t(:,h)'*x)/(t(:,h)'*t(:,h)); % CALCULATE INPUT SCORE VECTORS FOR TRAINING DATA
  vt(:,h) = vx*w(:,h); % CALCULATE INPUT LATENT VARIABLES FOR VALIDATION DATA
  vu(:,h) = vy; % INITIALIZE VALIDATION OUTPUT VECTOR
  vp(h,:) = vt(:,h)' * vx / (vt(:,h)' * vt(:,h)); % CALCULATE INPUT SCORES FOR VALIDATION DATA

  % This calls the function plsnet8if.m to train a single hidden layer network to map the
  %         hth inner relationship.  The vt and vu vectors are used for cross-validation training.
  % Adjustments to the training routine can be made within the plsnet8if.m function,
  %                 such as the training algorithm, number of hidden nodes, and stopping criterion.
  [net]=plsnet8if(t(:,h),u(:,h),vt(:,h),vu(:,h),im,maxnhn);

   % Extract the weight and bias values from the returned network structure "net".
   % size(net.IW{1,1},1) returns the number of neurons in the single hidden layer network
   % The code is written to allow up to four hidden neurons, however I usually use 2
   % The loop extracts the weights into the "wbs" matrix.
   % From the wbs matrix, the weight vectors are assigned as follows:
   %                 w1 - input layer weights
   %     b1 - input layer bias
   %                 w2 - output layer weights
   %                 b2 - output layer bias

   w1=net.IW{1,1};
   w2=net.LW{2,1};
   b1=net.b{1};
   b2=net.b{2};
   nhn=length(w1);

   % COMPILE NEURAL NETWORK PARAMETERS INTO WBS MATRIX
   WBS(1,:,h)=ones(1,4)*nhn;
   WBS(2:nhn+1,:,h)=[w1 w2' b1 (ones(nhn,1)*b2)];

   % The response variable (y) is deflated by the information contained in the hth neural network
   % The validation data for the response variable is also deflated in a similar way.
   y = y - ((w2*(tansig(w1*t(:,h)'+(b1*ones(1,size(t,1)))))+b2)');
   vy = vy -((w2*(tansig(w1*vt(:,h)'+(b1*ones(1,size(vt,1)))))+b2)');
```

```
    % The predictor variables are also deflated by the information in the hth component
    x = x - (t(:,h) * p(h,:));
    vx= vx - (vt(:,h) * vp(h,:));

  end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SUBFUNCTION OF nnplsi_create5f.m FOR ANN TRAINING OF THE INNER RELATIONSHIPS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS FUNCTION RETURNS A TRAINED ANN AS NET BASED ON THE SPECIFIED INPUT TRAINING AND
% VALIDATION DATA

function [net] = plsnet8if(t,u,vt,vu,im,nhn);

%
% This function takes the inner relationship
%    in a PLS model and fits an ANN to it.
%
% t     is the input training latent vector
% u     is the output training vector
% vt    is the input validation latent vectors
% vu    is the output validation vector
% im is the ANN initialization method.
%   the hidden layer weights and biases are always randomly initialized.
%   the im options refer to the output layer weights and biases
%   im=0, random initialization for output layer parameters
%   im=1, output layer weights set to (standard deviation(u) / n1), output bias set to mean(u)

% net   is the resulting network model

for nh=1:nhn; % HIDDEN NEURON INDEX
   %INITIALIZE ANN PARAMETERS
   net=newff(minmax(t'),[nh 1] ,{'tansig','purelin'},'trainlm');
   net.trainParam.show=100;
   net.trainParam.epochs=800;
   net.trainParam.mu_max=1e20;
   net.trainParam.max_fail=20;
   [a1,a2]=(size(net.IW{1,1}));
   net.IW{1,1}=rands(a1,a2);
      [a1,a2]=(size(net.LW{2,1}));
   net.LW{2,1}=rands(a1,a2);
      [a1,a2]=(size(net.b{1}));
   net.b{1}=rands(a1,a2);
         [a1,a2]=(size(net.b{2}));
   net.b{2}=rands(a1,a2);

   if im==0;
        %fprintf('Random Initialization\n\n')
     elseif im==1;
        net.LW{2,1}=ones(1,nh)*(std(u)/nh);
        net.b{2}=mean(u);
        %fprintf('Mean / Std Initialization\n\n');

     end;
   % SPECIFY VALIDATION DATA AND TRAIN ANN
   VV.P=vt';VV.T=vu';
   [net,tr]=train(net,t',u',[],[],VV);
   % EVALUATE ANN ON VALIDATION DATA
   vp=sim(net,vt')';
   er(nh)=mean((vu-vp).^2); % MSE OF VLIDATION DATA
   % SAVE ANN IN TEMPORARY DIRECTORY
   NNET(nh)=struct('N',net);
   fname=['c:\temp\NNET' eval('num2str(nh)')];
   save(eval('fname'),'net');
end;

% DETERMINE BEST ANN BASED ON MINIMUM VALIDATION ERROR
bestnh=find(er==min(er));
% RELOAD BEST ANN TO PASS BACK AS OUTPUT
bname=['c:\temp\NNET' eval('num2str(bestnh)')];clear net;
load(eval('bname'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SUBFUNCTION OF nnplsi_main77f.m TO COMPUTE ESTIMATIONS AND POINT-WISE PREDICTION INTERVALS
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS FUNCTION COMPUTES THE ESTIMATES AND PREDICTION INTERVALS FOR THE SPECIFIED DATA SETS
% AND NNPLS MODEL

function [ypt,ypv,ypts,trainunc,valunc,testunc] = nnplsi_predictall(x,xx,xxx,w,wbs,p,fac);

% x is an [nx(d+1)] matrix of training data, where the first n columns are the predictor variables,
%   and the last column contains the corresponding responses.
% xx is an [px(d+1)] matrix of validation data, where the first n columns are the predictor variables,
%   and the last column contains the corresponding responses.
% xxx is an [mx(d+1)] matrix of validation data, where the first n columns are the predictor variables,
%   and the last column contains the corresponding responses.
% w is the transofrmation weight matrix of the current NNPLS model
% wbs is the matrix of ANN parameters for the NNPLS model inner relationships
% p is the input score vector matrix
% fac is the maximum number of latent variables to evaluate

% ypt is a [n x fac] matrix of the training response estimates
% ypv is a [p x fac] matrix of the validation response estimates
% ypts is a [m x fac] matrix of the test response estimates
% trainunc is a [n x fac] matrix of the point-wise prediction interval estimates for YPT
% valunc is a [p x fac] matrix of the point-wise prediction interval estimates for YPV
% testunc is a [m x fac] matrix of the point-wise prediction interval estimates for YPTS

% INITIALIZE DIMENSIONS AND SEPARATE PREDICTORS FROM RESPONSES
N=size(x,1);N2=size(xx,1);N3=size(xxx,1);
NI=size(x,2);y=x(:,NI);x=x(:,1:(NI-1));
NI2=size(xx,2);yy=xx(:,NI2);xx=xx(:,1:(NI2-1));
NI3=size(xxx,2);yyy=xxx(:,NI3);xxx=xxx(:,1:(NI3-1));
NI=size(x,2);
NI2=size(xx,2);
NI3=size(xxx,2);
ox=x;ox2=xx;ox3=xxx;

% INITIALIZE RESPONSE ESTIMATION MATRICES AND DERIVATIVE MATRICES
yp=zeros(size(x,1),fac);yp2=zeros(size(xx,1),fac);yp3=zeros(size(xxx,1),fac);
ALLFR=[];ALLFT=[];ALLFR2=[];ALLFT2=[];ALLFR3=[];ALLFT3=[];

for h=1:fac; % LATENT VECTOR INDEX

   % COMPUTE LATENT VECTORS
   T(:,h)=x*w(:,h);    T2(:,h)=xx*w(:,h);    T3(:,h)=xxx*w(:,h);

  % Determine the number of hidden nodes used in the ANN for each component
  %             then extract the corresponding weight and bias values from the wbs matrix
  nn1=wbs(1,1,h);
  w1=wbs(2:nn1+1,1,h);
  w2=wbs(2:nn1+1,2,h)';
  b1=wbs(2:nn1+1,3,h);
  b2=wbs(2,4,h);

  % COMPUTE DERIVATIVES

  for k=1:wbs(1,1,h);
     O(:,k,h)=tanh((w1(k)*T(:,h))+b1(k));
     U1(:,k,h)=O(:,k,h)*w2(k);
     FT1(:,k,h)=((w2(k)*(1+O(:,k,h).^2)).*T(:,h));
     FT2(:,k,h)=(w2(k)*(1+O(:,k,h).^2));
     FR1(:,k,h)=(w1(k)*(1+O(:,k,h).^2)*w2(k));
     FT3(:,k,h)=O(:,k,h);
  end;

  for k=1:wbs(1,1,h);
     Ov(:,k,h)=tanh((w1(k)*T2(:,h))+b1(k));
     U1v(:,k,h)=Ov(:,k,h)*w2(k);
     FT1v(:,k,h)=((w2(k)*(1+Ov(:,k,h).^2)).*T2(:,h));
     FT2v(:,k,h)=(w2(k)*(1+Ov(:,k,h).^2));
     FR1v(:,k,h)=(w1(k)*(1+Ov(:,k,h).^2)*w2(k));
     FT3v(:,k,h)=Ov(:,k,h);
  end;

  for k=1:wbs(1,1,h);
     Oz(:,k,h)=tanh((w1(k)*T3(:,h))+b1(k));
     U1z(:,k,h)=Oz(:,k,h)*w2(k);
     FT1z(:,k,h)=((w2(k)*(1+Oz(:,k,h).^2)).*T3(:,h));
     FT2z(:,k,h)=(w2(k)*(1+Oz(:,k,h).^2));
     FR1z(:,k,h)=(w1(k)*(1+Oz(:,k,h).^2)*w2(k));
     FT3z(:,k,h)=Oz(:,k,h);
```

```
  end;

  U(:,h)=(sum(U1(:,:,h)')+b2)';
  U2(:,h)=(sum(U1v(:,:,h)')+b2)';
  U3(:,h)=(sum(U1z(:,:,h)')+b2)';
  FQ(:,h)=U(:,h);   FQ2(:,h)=U2(:,h);   FQ3(:,h)=U3(:,h);
  FT4=ones(N,1);   FT4v=ones(N2,1);   FT4z=ones(N3,1);
  FR2(:,h)=sum(FR1(:,:,h)')';
  FR2v(:,h)=sum(FR1v(:,:,h)')';
  FR2z(:,h)=sum(FR1z(:,:,h)')';

  % Update y by adding the Contribution from ANN h.

  if h==1;
     yp(:,h) = ((w2*(tansig(w1*T(:,h)'+(b1*ones(1,size(T,1)))))+b2)');
     yp2(:,h) = ((w2*(tansig(w1*T2(:,h)'+(b1*ones(1,size(T2,1)))))+b2)');
     yp3(:,h) = ((w2*(tansig(w1*T3(:,h)'+(b1*ones(1,size(T3,1)))))+b2)');
  else
    yp(:,h) = yp(:,h-1)+((w2*(tansig(w1*T(:,h)'+(b1*ones(1,size(T,1)))))+b2)');
    yp2(:,h) = yp2(:,h-1)+((w2*(tansig(w1*T2(:,h)'+(b1*ones(1,size(T2,1)))))+b2)');
    yp3(:,h) = yp3(:,h-1)+((w2*(tansig(w1*T3(:,h)'+(b1*ones(1,size(T3,1)))))+b2)');
 end;

  ypt(:,h)=yp(:,h);    ypv(:,h)=yp2(:,h);    ypts(:,h)=yp3(:,h);
  % Input Residual Matrix Deflation
    x=x-(T(:,h) * p(h,:));    xx=xx-(T2(:,h) * p(h,:));    xxx=xxx-(T3(:,h) * p(h,:));

  % COMPILE DERIVATIVES INTO MATRICES
  for hh=1:NI;
     ALLFR=[ALLFR (FR2(:,h).*ox(:,hh))];
  end;
  for hh=1:NI2;
     ALLFR2=[ALLFR2 (FR2v(:,h).*ox2(:,hh))];
  end;
  for hh=1:NI3;
     ALLFR3=[ALLFR3 (FR2z(:,h).*ox3(:,hh))];
  end;

  ALLFT=[ALLFT FT1(:,(1:wbs(1,1,h)),h) FT2(:,(1:wbs(1,1,h)),h) FT3(:,(1:wbs(1,1,h)),h) FT4];
  ALLFT2=[ALLFT2 FT1v(:,(1:wbs(1,1,h)),h) FT2v(:,(1:wbs(1,1,h)),h) FT3v(:,(1:wbs(1,1,h)),h) FT4v];
  ALLFT3=[ALLFT3 FT1z(:,(1:wbs(1,1,h)),h) FT2z(:,(1:wbs(1,1,h)),h) FT3z(:,(1:wbs(1,1,h)),h) FT4z];
  F=[ALLFR ALLFT FQ];    F2=[ALLFR2 ALLFT2 FQ2];    F3=[ALLFR3 ALLFT3 FQ3];

  % ESTIMATE ESTIMATE STANDARD DEVIATION BASED ON TRAINING AND VALIDATION DATA
   s(h)=sqrt(mean([((y-ypt(:,h)).^2);((yy-ypv(:,h)).^2)]));

  nump=size(F,2); % NUMBER OF PARAMETERS
  dof=size(x,1)-nump; % DEGREES OF FREEDOM

  % OBTAIN POINT-WISE PREDICTION INTERVAL VALUES FOR ALL DATA SETS
  for ii=1:N;
     trainunc(ii,h)=  tinv(0.95,dof)*s(h)*sqrt(1+(F(ii,:)*pinv(F'*F)*F(ii,:)'));
  end;

  for ii=1:N2;
     valunc(ii,h)=  tinv(0.95,dof)*s(h)*sqrt(1+(F2(ii,:)*pinv(F'*F)*F2(ii,:)'));
  end;

  for ii=1:N3;
     testunc(ii,h)=  tinv(0.95,dof)*s(h)*sqrt(1+(F3(ii,:)*pinv(F'*F)*F3(ii,:)'));
  end;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SUBFUNCTION OF nnplsi_create5f.m FOR SEPARATING A SPECIFIED COLUMN FROM A FULL MATRIX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,y]=divide(data,npred);
%
% THIS FUNCTION SEPARATES THE COLUMN SPECIFIED BY NPRED FROM THE MATRIX DATA.
%
% data is an nxp matrix which contains n sample vectors for p variables.
% npred is a scalar corresponding to the column to be separated out from data
%
% x is the [nx(p-1)] remainder of the original matrix
% y is the (nx1) extracted column
```

```
y=data(:,npred);

if npred==1;
        x=data(:,2:size(data,2));
elseif npred==size(data,2);
        x=data(:,1:(npred-1));
else;
        x=[data(:,1:(npred-1)) data(:,(npred+1:size(data,2)))];
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

VITA

Brandon Rasmussen was born on February 13, 1973 in West Islip, NY. He grew up in Copaigue, NY, moved to Evans City, PA at the age of 10, and graduated from Vincentian High School in Pittsburgh, PA in 1990. He began college at Kent State University, Kent, OH, transferred the following year to Gannon University, Erie, PA, and received a Bachelor of Science in Health Physics from Francis Marion University, Florence, SC, in August of 1995. In May of 2002, he received a MS degree in Nuclear Engineering from the University of Tennessee – Knoxville.

As a Health Physics intern, he spent the summer of 1993 at Carolina Power and Light's Shearon Harris Nuclear Power Plant, in New Hill, NC. He was a member of the Francis Marion University Health Physics club from 1993 through 1995, and was the active president from 1994 to 1995. He received a Francis Marion University Alumni Scholarship in 1993 and 1994, was awarded a National Academy for Nuclear Training Scholarship in 1994, and he received the Carolina Power and Light Environmental Health Physics award in 1995.

He entered the Nuclear Engineering graduate program at the University of Tennessee in 1997. In 1999, he and John P. Getty, under the direction of Dr. Arthur E. Ruggles, received first place honors in the American Nuclear Society's Graduate Design Competition for their design entitled "Pressure Vessel Diameter Reduction of a Modular Pebble Bed Reactor." The American Nuclear Society has awarded him Graduate Scholarships in 1999, 2001, and 2002.

In February of 2002, he spent 3 weeks in Halden, Norway to collaborate with the personnel of the Halden Reactor Project. The purpose of the collaboration was to evaluate the utility of the NNPLS algorithm in their existing Plant Evaluation and Analysis by Neural Operators (PEANO) software.

He has completed and presented 5 conference publications as the primary author, and has co-authored 7 other conference publications. He has also co-authored 2 journal publications and was the primary author of a recent publication in *Nuclear Technology*.

He received a Department of Energy Nuclear Engineering Fellowship in 2002, under which the balance of this dissertation was completed.

The author is a member of the American Nuclear Society and Phi Kappa Phi National Honor Society. He has also received a Maintenance and Reliability Certification from the Maintenance and Reliability Center at the University of Tennessee at the MS level.