1996

Process modeling using stacked neural networks

Dasaratha Sridhar
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Process modeling using stacked neural networks

by

Dasaratha Sridhar

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Chemical Engineering

Major Professors: Richard C. Seagrave and Eric B. Bartlett

Iowa State University

Ames, Iowa

1996

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Dasaratha Sridhar

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Co-major Professor

Signature was redacted for privacy.

Co-major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College

In fond memory of
my father
Sri. Dasaratha Venkataraman

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I wish to thank the 3M corporation and NASA for partial funding of this research. I would like to thank my dissertation advisors Dr. Richard Seagrave and Dr. Eric Bartlett for guidance and support over the past five years. I would also like to thank Dr. Rollins, Dr. Ulrichson and Dr. Udpa for being on my program committee. Special thanks to Mike Rohricht from 3M for his input into this dissertation. I would also like to thank the faculty, staff and students of the chemical engineering department who have made my stay at Iowa State University very enjoyable. I am also grateful to the members of the Adaptive Computing Laboratory for their support and input into this research. I would like to thank my mother, sister and brother-in-law for their encouragement and support throughout this dissertation. Finally, I wish to thank my wife Ramya for her understanding and support through some difficult times.

GENERAL INTRODUCTION

Prediction, control and optimization of chemical processes is an important problem in the chemical industry. Effective prediction and optimization of chemical processes requires accurate modeling of process behavior. A process model essentially represents the relationship between the process inputs and outputs. For example, the process inputs could be the process variables and the process output could be the product quality. An accurate process model would then effectively capture the relationships between the process variables and the product quality, and could be used to optimize product quality. Process models can be based on an understanding of the physical and chemical events taking place in the process. An alternative approach is to develop an empirical process model from process data. An empirical modeling approach is particularly useful when the physical and chemical phenomena underlying the process are complex and not well understood.

In recent years, artificial neural networks (ANNs) have been proposed as a promising tool for empirical modeling of chemical processes. Neural networks can learn complex multivariate input-output relationships from data. ANNs have been successfully used in applications where it is difficult to specify the form of the relationship between the output and input variables. An artificial neural network consists of a number of simple linear or nonlinear computing elements interconnected in complex ways and typically organized into layers. The nature of these interconnections is defined by the network architecture. Development of a neural network model requires specification of the network architecture and a learning technique to estimate the parameters of the network. In general it is

difficult to decide what network architecture and learning technique will work best for a given problem. Neural network modelers typically train multiple neural networks in the hope that a single optimal network will be found. The model that is optimal is determined by evaluating all of the networks on data available for testing the model and selecting the model with the best performance. The optimal model is then used for the desired application and the rest of the models are discarded. However there is no assurance that any individual network has extracted all of the useful information in the data set. It is possible that some of the discarded networks may contain information that has not been captured by the network that is considered to be optimal. Selecting and using a single model would then waste the information in the discarded networks and could result in a process model that may not accurately capture the relationship between the process inputs and outputs.

The work in this dissertation presents a novel approach for empirical modeling of chemical processes. In contrast to the common approach of selecting and using a single neural network model, a scheme that allows multiple neural networks to be selected and combined is presented and evaluated. The integrated neural network system is defined to be a stacked neural network. The central idea behind the stacked neural network approach is that improved process modeling would be possible by integrating the knowledge acquired by the candidate networks. Any scheme that uses multiple ANNs requires a combining technique that can effectively integrate the candidate networks. The feasibility of the stacked neural network technique is first explored using linear combinations. However the linear combination is limited in that it can use only those networks whose outputs have a high linear correlation to the output. The main focus of this research is to develop and evaluate a general technique that can identify and combine informative neural network models regardless of how their outputs relate to the process output. When a large number of models are used, estimating an arbitrary nonlinear combination can become very complex. An algorithm that can select an informative

subset of the candidate models is then essential. Such an approach would be able to effectively select and integrate the most informative neural network models to provide improved process models. Improved modeling would in turn allow better optimization and control of plant processes.

Dissertation organization

The main body of this dissertation comprises three self-contained papers that represent three distinct phases of this research. They are presented in a chronological order. Dasaratha Sridhar is the principal investigator. Dr. Eric B. Bartlett and Dr. Richard C. Seagrave appear as the second and third authors.

The first paper has been published in the AIChE journal. This paper describes a stacked neural network (SNN) that uses a linear combination of neural networks for modeling of chemical processes. The SNN approach allows multiple neural networks to be combined and used to model a given process. The figures and tables for this paper have been collected at the end of the paper.

The second paper has been submitted to the Computers and Chemical Engineering journal. This paper lays the foundation for an algorithm that can identify the most informative models. The paper describes a scheme to identify important inputs for a model based on information theoretic analysis. The proposed approach helps identify simpler models with better generalization and eliminates the use of redundant inputs. The figures and tables for this paper have been collected at the end of the paper.

The third paper has been submitted to the Neural Networks journal. In this paper the concepts developed in the second paper are used to develop a novel approach for combining neural network models. The information theoretic stacking (ITS) algorithm proposed in this paper can combine multiple neural networks without assuming the form of the combination. An information theoretic approach is used to identify the

most important models to be used for the combination. The combination is determined from the data and informative models can be identified and combined regardless of how they actually relate to the output.

The three papers are followed by a general summary that summarizes the entire dissertation. It addresses issues arising from this work and lists avenues for investigation. Finally, additional references that have been consulted but not referenced by the three papers are listed following the section on general conclusions.

PROCESS MODELING USING STACKED NEURAL NETWORKS

Dasaratha Sridhar, Eric B. Bartlett and Richard C. Seagrave

Abstract

In this work, a new technique for neural network based modeling of chemical processes is proposed. Stacked Neural Networks (SNNs) allow multiple neural networks to be selected and used to model a given process. The idea is that improved predictions can be obtained using multiple networks, instead of simply selecting a single hopefully optimal network as is usually done. A methodology for stacking neural networks for plant process modeling has been developed. This method is inspired by the technique of stacked generalization proposed by Wolpert (1992). The proposed method has been applied and evaluated for three example problems including the dynamic modeling of a nonlinear chemical process. Results obtained demonstrate the promise of this approach for improved neural network based plant process modeling.

Introduction

In the chemical industry, nonlinear models are typically required for process control, process optimization and prediction of process behavior. Development of such models is

often a difficult task for processes that are complex or poorly understood. When theoretical modeling is difficult, empirical, data driven modeling provides a useful alternative. In recent years, artificial neural networks (ANNs) have been proposed as a promising tool for identifying empirical process models from process data (Bhat and McAvoy, 1990; Pollard *et al.*, 1992). Neural networks are very useful because of their ability to model complex nonlinear processes, even when process understanding is limited (Mah and Chakravarty, 1992). These neural network models can be used for prediction, provided the process correlation structure does not change (MacGregor, 1991). Typically, the main objective in ANN modeling is to accurately predict steady state or dynamic process behavior in order to monitor and improve process performance. Some of the important applications of ANNs in chemical engineering include the following:

- Fault Diagnosis in chemical plants: Hoskins and Himmelblau (1988); Venkatasubramaniam and Chan (1989)
- Dynamic modeling of chemical processes: Bhat and McAvoy (1990); Ydstie (1990); Pollard *et al.* (1992)
- System identification and control: Psichogios and Ungar (1991)
- Sensor data analysis: Kramer (1992)
- Prediction of product quality: Joseph *et al.* (1992)
- Chemical composition analysis and property prediction: McAvoy *et al.* (1989); Piovoso and Owens (1991)
- Inferential control: DiMassimo *et al.* (1992)

The typical approach to ANN based process modeling has been to consider a number of candidate models, and to select one model which is expected to best predict the

process outputs, given the process inputs. The selected model is the one that is expected to have least prediction error in the future. The expected prediction error of the candidate models is usually computed by evaluating them on data available for testing the model. When a separate data set is not available for selecting the model, cross-validation (Weiss and Kukilowski, 1991) can be used. Cross-validation allows model selection by using the same sample for model development as well as to provide a reasonable estimate of the expected prediction error in the future. Using a single optimal model implicitly assumes that one ANN model can extract all the information available in the data set and that the other candidate models are redundant. In general, there is no assurance that any individual model has extracted all relevant information from the data set. It is well recognized in the forecasting literature that identifying and using a single model is suboptimal for prediction (Bates and Granger. 1969; Granger and Ramanathan, 1984). Clemen (1989) provides a comprehensive review on the use of multiple models for forecasting. The primary conclusion of Clemen's work is that combining multiple models usually leads to increased forecast accuracy. Combining ANN models is based on the premise that different neural networks capture different aspects of process behavior: Aggregating this information should reduce uncertainty and provide more accurate predictions. Hence stacking or combining different models can be beneficial for predictive modeling. Stacked Neural Networks (SNNs), introduced here, allows multiple neural networks to be selected and combined in an attempt to obtain a better predictive model. The methodology is inspired by the general method for combining models known as stacked generalization (Wolpert, 1992). In this work, any empirical modeling approach based on stacked generalization will be referred to as stacked modeling or stacking.

The remainder of this paper is organized in the following manner. First the rationale for combining ANN models is presented followed by a discussion of stacked generalization. Second, the methodology for implementation of stacked neural networks

is provided. The idea of stacked modeling is then illustrated using a simple example. Following this, the feasibility of SNNs is explored through application to two examples including a dynamic process modeling problem. Results obtained demonstrate that stacked neural networks (SNNs) can achieve highly improved performance as compared to selecting a single optimal network using cross-validation.

Combining ANN models

Combining models to improve prediction accuracy is an idea which appears to have originated with the work of Bates and Granger (1969). Bates and Granger combined two different models for forecasting a time series and reported improved predictions using the combined model. Since Bates and Granger's pioneering efforts, a large amount of work has been performed on combining models for forecasting. It is now widely accepted that combining of forecasts is a pragmatic and useful approach for producing better forecasts (Granger, 1989).

In the classical approach to modeling, one typically assumes that a certain ideal model is "true" and, based on available evidence, an actual model is rejected or accepted. If it is felt that the model is misspecified, a better model is sought. This approach is appropriate when there is theory which guides the specification and evaluation of the models (Diebold, 1989). However, a more pragmatic approach can be taken: Models can be combined to improve predictions. Combining models is equivalent to admitting that a "true" single model is not easily attainable and that multiple models should be viewed as different pieces of information which can be integrated. Pooling models has a short-term perspective with an emphasis on real world applications (Winkler, 1989).

The above arguments hold for neural network based empirical modeling of plant processes. As in time-series forecasting, the emphasis is on maximizing accuracy on future predictions. In ANN modeling, it is difficult to specify the optimum ANN architecture

a priori. Therefore, neural networks can be viewed as inherently misspecified models since they are an approximation to the underlying process (White, 1989). Hence, there is no reason to believe a given network architecture represents the actual underlying structure of the data generating process. Different neural networks are capable of approximating different class of functions: A sufficiently large network can approximate any arbitrary function (Cybenko, 1989; Hornik, Stichombe and White, 1989). However, because of finite sample sizes, one tends to use architectures which are smaller than required. Smaller networks have lesser variance but show larger bias, as compared to the larger networks. This problem is well-known as the bias-variance dilemma (Geman et al., 1992). For a given problem, a number of network architectures are evaluated and a hopefully optimal architecture is selected, based on its performance on some test set. In our discussion here we define an optimal network architecture as one that performs better than any other model over the entire input space. Optimality in this sense cannot be guaranteed by simply selecting the model that produces the least average prediction error over some test set. It is quite possible that different networks perform better in different regions of the input space. This situation can arise because:

- The optimal architecture was not considered by the modeler or could not be found by the automatic network construction methods used. Methods like Cascade Correlation (Fahlman, 1990) and dynamic node architecture (Basu and Bartlett, 1994) can automatically develop good neural network models. However, optimality as defined above cannot be guaranteed as all these methods typically focus on minimizing the average error over some training set, and not over the entire input space.

- As explained by Hansen and Solomon (1990), multiple networks are desirable because optimization of network parameters is a problem with many local minima. Even for a given architecture, final parameters can differ between one run of the

algorithm to the next. For example, varying the initial starting conditions for network training can lead to a different solution for the network parameters. This tends to make the errors produced by the different networks uncorrelated. Therefore, the different networks might make independent errors on different subsets of the input space.

- Different activation functions and learning algorithms can lead to different generalization characteristics and no one activation function or learning algorithm may be uniformly best (Hashem, 1993).

- The convergence criteria used for network training can lead to very different solutions for a given network architecture. Networks can be trained using convergent training which means that the neural network is trained to convergence on the entire training data set. Alternately stopped training can be used. Stopped training involves monitoring the performance of the network as it trains on a training set and training is stopped when the error on the cross-validation set reaches a minimum. Networks trained by stopped training could be very different from those trained using convergent training (Finnoff et al., 1993).

Therefore, it is possible that a combination of ANN models could outperform a single model. If better performance is achieved using multiple models, it implies a better single model can be found. However, finding such a model requires further time and effort, with no assurance that the better single model will be found. Instead of searching for a better model, combining existing neural networks can provide a practical approach to develop a better overall model for prediction by using the models already at hand.

Stacked Generalization

In order to maximize the benefits of combining models, effective schemes for combining them must be used. From the forecasting literature, the usual approach is to model the true output as a linear combination of the outputs of the individual models. This can be shown to have lesser mean square error than any of the individual models (Granger and Ramanathan, 1984). In the context of neural networks it has been seen that a linear combination of neural networks often improved predictions (Hashem, 1993; Perrone and Cooper, 1993). The primary limitation of these methods is that model selection is not considered in conjunction with model combination. When using non-parametric methods like neural networks, it is possible that overparametrized models may be included in the combination. It is often difficult to judge which ANN model is a good generalizer. Hence models with poor generalization could be included along with good models causing performance of the combined model to be poor. For instance, Hashem (1993) uses a weighted average of the outputs of the candidate networks, estimated from the training data. To check the robustness of the combined model he proposes testing the individual models and the combined model on a cross validation set. While determining the combination, the reliability of the individual networks is not taken into account. Hashem later proposes algorithms which check if performance of the combined model is better than the individual models. In Hashem's work, the issue of whether the combined model is a good predictor is addressed only after a set of models has been selected and combined.

Stacked generalization (Wolpert, 1992) has been been proposed as a technique for combining models. Preliminary efforts by Sridhar *et al.* (1995) showed promise for improved predictive modeling using stacked generalization. The novel feature of stacked generalization is that it attempts to simultaneously solve the problem of model selection and estimation of model combinations to improve model predictions. This feature is

inherent in the stacked generalization procedure because it has been developed as an extension of nonparametric statistical methods used for model selection. The idea is that by analyzing the performance of the original models, using a statistical resampling scheme, one can estimate a combination of the original models which will maximize prediction accuracy in the future. Stacked generalization is based on the assumption that the resampling methods provide more information than simply specifying which model is the best: It can allow estimation of combinations of models which can provide more accurate predictions. The data used for fitting a stacked generalization model are generated using the same method as for cross-validation. Stacked generalization can be viewed as a more general solution to estimating an optimum predictive model, while accounting for the bias-variance tradeoff. Cross-validation simply selects one model, that minimizes the expected prediction error. Thus cross-validation provides a solution to minimizing future prediction error subject to the assumption that the minimization needs to be achieved by a single model. In fact, cross-validation can be viewed as a special case of stacked generalization. Stacking relaxes the assumption that one model needs to be selected and used for prediction. Multiple models can be selected and optimally combined to maximize prediction accuracy. The main goal is to maximize expected generalization in the future. If a combination of models reduces the expected prediction error then it is preferable. Hence, stacking provides a method to estimate combinations of models that can in theory, maximize generalization accuracy.

The details of stacked generalization can be found in Wolpert's paper: In this paper, we provide a brief overview of this technique. The objective of stacked generalization is to fit a model for the true output. The inputs to the stacked generalization (SG) model are the outputs of the different models. The concept of stacking different models is shown in Figure 1. The models that are developed from the original training set are referred to as the level 0 models. Consider the simplest case when we have two level 0 models M_1 and M_2. Let the training set be partitioned into L_2, containing one pattern (x,y) and

L_1, containing the remaining patterns. Both M_1 and M_2 are trained on L_1 and then tested on L_2. Let the outputs of the two models be y_1 and y_2, while the true output is y. This information can be considered to be input-output information in a different space, with two inputs (the predictions of the two models) and one output (the actual output). Repeating the above procedure with each training pattern in L_2, we obtain a data set L' containing the predictions of the two models and the true output. This data constitutes a new data set that can be used to train a new model M_3 to predict y given y_1 and y_2. M_3 is known as the level 1 model. For any novel input vector $\mathbf{x}_{new}$, the predictions y_1 and y_2 are obtained and these predictions are combined by M_3 to produce the final prediction y_3. Extension of this approach, when more than two models are considered, is based on the same principles discussed above and is fairly straightforward.

Stacked neural networks

When the candidate level 0 models are artificial neural networks and they are combined using stacked generalization, we define the resulting model as a stacked neural network (SNN). In this work, attention is restricted to level 0 models which are single layer backpropagation networks. Of course other level 0 models such as radial basis function networks, general regression networks and backpropagation networks with more than one hidden layer can be used. The methodology developed here to stack level 0 neural networks can easily be extended to include any level 0 model. The level 1 models considered in this paper are linear models. The proposed architecture for SNNs is shown in Figure 2.

Assume that a data set $D((y_n, \mathbf{x}_n) : 1 \leq n \leq N)$ has been collected on some process of interest. Here, y denotes the output variable, x denotes the input vector, N is the number of training patterns and n is the pattern index. It is assumed that M neural networks (N_1, N_2, ...,N_M) are the candidate level 0 models. The level 0 data set will

be denoted by D_{L0}. The data used to develop the level 1 model will be denoted by D_{L1}. The following algorithm is used for both stacking the ANN models, and for selecting the best ANN model using k-fold cross-validation. The approach can be easily extended for problems with multiple outputs.

1. Set the level 0 data set D_{L0} equal to the data set D. Train the M level 0 networks using D_{L0}. Denote the j^{th} network trained on D_{L0} as $N_j(D_{L0})$, and denote the set of these level 0 networks as $N(D_{L0}) = \{N_j(D_{L0}) : 1 \leq j \leq M\}$. The $N(D_{L0})$ should be saved for use in the SNN model.

2. Divide the data set D into k disjoint subsets with equal number of patterns, D_1, D_2 ....D_k, as in k-fold cross-validation. Define CV_i as $D - D_i$. CV_i contains all the patterns in the data set D except the patterns in D_i.

3. Repeat the following for $1 \leq i \leq k$: Train the M candidate ANN models using the data set CV_i. Denote the j^{th} network trained on CV_i as $N_j(CV_i)$, and denote the set of these trained networks as $N(CV_i) = \{N_j(CV_i) : 1 \leq j \leq M\}$. Note that the $N(CV_i)$ have the same architecture as the M level 0 networks, however they are trained on data set CV_i. The sole reason for developing the networks $N(CV_i)$ is to construct the level 1 data set D_{L1}. Recall $N(CV_i)$ on the data set D_i. Denote the prediction of the j^{th} network for pattern n in data set D_i as yp_{nj}. For pattern n, the output of the candidate models is collected in a M dimensional vector $\mathbf{yp}_n = \{yp_{nj} : 1 \leq j \leq M\}$. The actual output y_n and the network outputs $\mathbf{yp}_n$ constitute the output and input respectively for the n^{th} pattern in data set D_{L1}. Discard the networks $\{N_j(CV_i) : 1 \leq j \leq \text{M}\}$.

4. Step 3 produces the level 1 data set $D_{L1}(y_n, \mathbf{yp}_n)$. Data set D_{L1} contains the true output and the predictions of the M models, for all the N training patterns.

5. Compute the prediction sums of squares errors (PRESS) for each of the M networks in $N(D_{L0})$. PRESS for network j is calculated as:

$$PRESS_j = \sum_{n=1}^{N} (y_n - yp_{nj})^2 \tag{1}$$

The network with the minimum PRESS is considered to be the optimal single network.

6. For developing the stacked model, the following level 1 model will be used to combine the M level 0 neural networks in $N(D_{L0})$.

$$y = \sum_{j=1}^{M} \alpha_j yp_j \tag{2}$$

where all the model coefficients α_j are required to be positive. Similar combining rules have been used for combining models for forecasting (Bates and Granger, 1969; Granger and Ramanathan, 1984) and more recently for combining regression models (Brieman, 1992). D_{L1} is used to estimate the parameters of the level 1 model. The objective function to be minimized for the level 1 model is the output sums of squares errors over D_{L1}. More complex level 1 models can also be used if desired.

7. For prediction on a novel input pattern, the input is fed through all of the candidate models. The outputs of the level 0 neural networks are combined using the level 1 model to produce the final prediction.

The schematic for the proposed method is shown in Figure 3. It should be noted that the above method can be used not only with k-fold cross-validation, but with any data based model-selection technique. One commonly used approach is to divide the data set into two data sets: D_1 used for training the candidate models and D_2 for testing the developed models. Development of SNNs in this case, is a straightforward variation of

the method discussed above. D_1 constitutes the level 0 data set D_{L0} and testing on D_2 produces the level 1 data set D_{L1}.

The methodology for developing stacked models will first be illustrated using a simple example. Application of the stacked neural network (SNN) approach to two example problems, with a discussion of results will then be presented.

Example 1

The first example involves identification of a simple function, using linear regression and nearest neighbor models as the candidate models. Neural networks are not consider as candidate models in this example, since it is intended to illustrate the idea of stacked modeling in a simple manner. Assume that the true parent function generating the data is

$$w = x + y + z \tag{3}$$

x, y, z are independent variables assumed to be drawn from a uniform distribution on the interval [0,1]. A data set of eight patterns corresponding to the eight corners of the unit cube, defined by the independent variables, has been collected for identifying the function (Table 1). The following models w_1, w_2, w_3 (level 0 models) are considered as candidates for identifying the true function w:

1. w_1 = x
2. w_2 = ay + bz
3. Nearest Neighbor model using variables y and z. For the nearest neighbor model

   w_3 = output for the nearest pattern in the data set

   For example, for the pattern (0.05, 0.1, 0.1) the closest training pattern is (0, 0, 0). Therefore the output of the nearest neighbor model is 0.

The above functions are deliberately chosen to illustrate the central ideas behind stacked modeling. It is obvious that each of the models alone are incapable of representing the true function. Models 1 and 2 together contain all the information required to identify the true function. Model 3 which is the nearest neighbor model provides a perfect fit to the training data, unlike models 1 and 2. However model 3 is not a good approximation to the true function and is in fact redundant, given models 1 and 2. Ideally it is desired that stacking the three candidate models should result in models 1 and 2 being combined while model 3 should be rejected. Table 1 also shows the outputs of the models obtained using leave-one out cross-validation. $w_{ij}{}^{cv}$ is the prediction of model i on the j^{th} pattern when that pattern is left out and the model is developed using rest of the data. Table 2 shows the expected prediction error, E(MSE), for all the three models, computed using cross-validation. The performance of the three models, evaluated on 10000 test patterns is also shown. The performance measures used are the mean square error (MSE) and the linear correlation coefficient (r), between the desired output and the model's actual output. Model 2 is identified by cross-validation as the best single predictor, as it has the least expected prediction error. Model 2 is indeed the best individual predictor, as it has the least error on the test data set. However cross-validation has ignored the useful information in Model 1. Clearly using a single model is suboptimal in this case. Stacked modeling, on the other hand, attempts to identify a combination of the three models. Fitting the combining rule defined by Equation 2 to the level 1 data shown in Table 1, results in the following level 1 model:

$$w_s = 1.22w_1 + 0.728w_2 \tag{4}$$

Substituting for the functions w_1 and w_2 in terms of the independent variables x, y and z gives the final model:

$$w = 1.22x + 0.92y + 0.92z \tag{5}$$

The expected prediction error and the error on the test data set for the stacked model are both less than that of any of the other single models. Combining the three models results in a stacked model that is quite close to the actual function. The stacked model is better than any of the individual models investigated. It can be seen from Equation 4 that model 3 is completely rejected, and models 1 and 2 are combined to effectively approximate the true function. This example illustrates the usefulness of combining, when the candidate models are misspecified. If we had specified one of the candidate models more accurately, we could have exactly identified the true function. In this example, if we had attempted to fit

$$w = ax + by + cz \tag{6}$$

the true function would have been identified exactly. If we can accurately specify models, combining would not be required. In the present example, it is possible to determine the true model. However in complex real world problems model specification, as well as determining the optimal model, can be quite difficult. In such situations we would expect that stacked modeling will provide us with better predictions by integrating all the useful information that the various candidate models may have acquired. It should also be noted that in the above example none of the models used all the variables. It is therefore obvious that the models have information that is independent of each other. The problem was designed to illustrate the benefits of stacked modeling in a simple manner. The use of Equation 2 for stacking is very effective for this example. In general, it is not known what level 1 model is most effective for stacking the candidate models. However, for the sake of simplicity, we have restricted our attention in this paper, to the linear level 1 model.

Example 2

It is often possible that all the candidate models considered for approximating a given process use all of the independent variables. The independence of information contained in the outputs of these models arises from the inherent differences in the approximation abilities of the candidate models. Of course some correlation between the candidate models is inevitable, since they all make use of information from the same data. However, the information that is captured could be different, and this is what we seek to take advantage of and integrate through stacked modeling. For this example, the objective is to estimate the 6-dimensional additive function,

$$y = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + 0x_6 \tag{7}$$

The x_i were generated from a uniform distribution in the six-dimensional hypercube. This function has been used as a benchmark for MARS (Friedman, 1991) and more recently for comparing neural network and statistical methods (Cherkassky *et al.*, 1995). We follow an approach similar to Cherkassky *et al.* for this study. The training data set S_1 consisted of 100 samples, and constitutes the level 0 data. The added noise was Gaussian with a signal to noise ratio (SNR) of 4.0. The signal to noise ratio is defined as

$$SNR = \sigma_y / \sigma_N \tag{8}$$

where, σ_y is the standard deviation of y and σ_N is the standard deviation of the noise. In addition we also generated a level 1 data set S_2 consisting of 50 patterns. Data set S_2 was used for model selection and model integration. A test data set S_3 of 10000 patterns without any noise was generated to measure the performance of the different models. The performance metric used was the normalized root mean square error (NRMS), which is the average root mean square error on the test data set normalized by the standard deviation of the data set. NRMS represents the fraction of unexplained standard devi-

ation. Four different single hidden layer backpropagation networks N_1, N_2, N_3 and N_4 with 5, 10, 20 and 40 hidden nodes respectively were developed. Networks were trained using the scaled conjugate gradient algorithm (Moller, 1993). N_1 was trained on all 100 patterns. For the remaining networks stopped training was used. These networks were trained on 70 randomly selected patterns and their performance was monitored on the remaining 30 patterns. The weights which gave the least error on the 30 patterns were considered to be the optimal weights for the networks. Performance of the candidate networks on the data set S_3 was evaluated by computing the NRMS. The different models were then stacked to obtain the SNN model. The NRMS for the SNN was also computed. The performance comparisons are shown in Table 3. Using cross-validation, N_2 is selected as the best model. N_2 has an NRMS of 0.34 on the test data. N_1 also performs similar to N_2. The larger networks N_3 and N_4 perform poorly and appear to be overparameterized. Testing on S_3 shows that these models are indeed poor generalizers. The competing networks were then combined using SNN and resulted in the following level 1 model:

$$y_s = 0.469 * y_1 + 0.531 * y_2 \tag{9}$$

where y_s is the output of the SNN and the y_i are the outputs of the i_{th} network. The RMSE of the SNN on the test data was found to be 0.25. Thus, a reduction of 26% in the prediction error was obtained using SNN over the cross-validation selected network. It is interesting to note that Cherkassky *et al.* (1995) report an NRMS of 0.319 for this problem. Their results are slightly better than those obtained in this work with N_1 (NRMS = 0.32) and N_2 (NRMS = 0.34). However SNN helped us significantly improve the model performance. SNN had NRMS of about 22% less than that reported in Cherkassky's work. N_3 and N_4 were rejected by the SNN model, while N_1 and N_2 were combined to provide an improved model. SNN was able to successfully integrate the knowledge acquired by the candidate networks. In terms of overall performance, both

N_1 and N_2 are equally good (NRMS within 6% of each other). Improved performance by combining the two models means that the models make errors of different types. Indeed the linear correlation coefficient between the errors made by the two models is very small (r=0.11). As pointed out earlier, a number of reasons can cause networks to perform very differently. In this case, the network architecture, the use of both convergent and stopped training and the different weight initializations lead to the significant differences between models N_1 and N_2. The error independence of the two models allowed SNN to integrate the two models and develop a better model. It can be argued here that a single network could have been found which has the same performance as the SNN. Indeed this is theoretically possible. Obtaining this optimal model would however require searching for such a model. The SNN method avoided a further search and provided a direct approach towards obtaining a better model.

Dynamic modeling example

This example is intended to demonstrate the performance of SNNs for a more practical modeling problem. In this example, the performance of the stacked neural networks was examined for training sets of different sizes and various noise levels.

The process considered is an ideal, adiabatic continuous first order exothermic reaction in a continuous stirred tank reactor (CSTR), shown in Figure 4. This process has been used as a benchmark for internal model control (Economou, 1986), radial basis function modeling (Leonard et al., 1992) and comparison of backpropagation networks and multivariate adaptive regression splines (DeVeaux et al., 1993). The feed to the CSTR is pure A and the desired product is R. The system is simulated by the following coupled ordinary differential equations.

$$\begin{aligned} \frac{dA_o}{dt} &= \frac{A_i - A_o}{\tau} - k_1 A_o + k_{-1} R_o \\ \frac{dR_o}{dt} &= \frac{R_i - R_o}{\tau} + k_1 A_o - k_{-1} R_o \end{aligned} \tag{10}$$

$$\frac{dT_o}{dt} = \frac{-\Delta H_R}{\rho C_p}(k_1 A_o - k_{-1} R_o) + \frac{T - T_o}{\tau}$$

where,

$$k_1 = c_1 exp\left(\frac{-Q_1}{RT_o}\right)$$
$$k_{-1} = c_{-1} exp\left(\frac{-Q_{-1}}{RT_o}\right)$$

The variables A_o, R_o and T_o represent the state variables of the system and are the feed species concentration, the output product species concentration and the reactor's temperature, respectively. The goal of the modeling is to learn the open loop relationship between the controlled variable which is the concentration of species R and the manipulated variable which is the temperature of the feed stream, T_i. It is desired to predict the concentration of species R at the next time step, given the state variables A_o, R_o, T_o and the manipulated input T_i. The input and output dimensionality of the system are 4 and 1 respectively. The steady state operating point and the process parameters are as given in the paper by Economou *et al.*, and are shown in Table 4. A method similar to that in DeVeaux *et al.*, was used to generate the training data for the neural network modeling. The above set of differential equations were integrated with T varying randomly with a uniform distribution within 15% of the steady state operating point. Sampling time for the process was 30 seconds. During the simulation, the systems state variable as well as manipulated input were recorded. At the end of the sampling instant, the system's response was also recorded. Measurement noise was added to the noise-free simulation. The noise was assumed to be Gaussian. A time-series plot of one realization of the data is shown in Figure 5. Training data sets S_1 of sizes 50, 70, 100 and 120 were generated at five different noise levels with standard deviations of 2.5%, 5.0%, 7.5%, 10%, 12.5% of the range of the output, respectively. Thus a total of twenty data sets were considered in this study. A noise-free test data set S_2 of 10000 patterns was also generated in order to test and compare all the models developed.

In the present analysis, the following terminology will be used:

1. Cross-validation selected Network (CVSN): This is the network that is selected using the cross-validation scheme

2. Stacked Neural Networks (SNN): SNN represents the stacked model, where the competing ANN models are stacked using the technique described earlier in this work

3. Best *a* posteriori Network (BAPN): After the CVSN and SNN have been developed and tested, all of the candidate models are tested on the test data S_2. The network with the least error on the test data set S_2 will be referred to as the Best *a posteriori* network.

The CVSN and SNNs will be compared to the BAPN in the following analysis. The rationale for using the BAPN as a baseline for comparison is that BAPN represents the best performance that could have been obtained given the candidate networks and assuming we wanted to select and use only a single network as our model. The idea behind using cross-validation is to identify the BAPN beforehand. However the BAPN may or may not be successfully identified by cross-validation. In either case, the CVSN can never be better than the BAPN. SNNs, on the other hand, have the potential to identify stacked models that may be better than the BAPN.

Eight neural network models $[N_i : 1 \leq i \leq 8]$ with 1, 2, 3, 4, 8, 12, 16 and 20 hidden nodes respectively were considered as the level 0 models for modeling the process. For the training algorithm, the scaled conjugate gradient algorithm (SCGA) was used (Moller, 1993). The methodology described earlier in this work was used to stack the eight networks and k-fold cross-validation was used to identify the CVSN. The neural nets considered in this example had one hidden layer and used the hyperbolic tangent function as the activation function. Weight initialization for networks N_1, N_2, N_3, N_4

was performed using the method based on principal component regression, proposed by Piovoso and Owens (1991). The remaining networks were initialized with random weights distributed uniformly on [-0.1,0.1].

After training, the performance of the SNN and the CVSN was tested on the 10000 patterns. Finally, all eight networks were evaluated on the test data set to identify the BAPN. The mean absolute prediction error (MAPE) (Makradakis *et al.*, 1983) was used for making the comparisons. Table 5 shows the CVSN, the MAPE for the CVSN, the MAPE for the SNN and the percent reduction in MAPE using SNN as compared to using the CVSN, for all the 20 cases. For convenience, the numbers have been multiplied by 10000. As can be seen from Table 5, SNNs outperformed the CVSN in all cases. Reduction in prediction errors achieved by using the SNN, as compared to using the CVSN, ranged from 7% to 41%.

Graphical comparisons of the SNN with the CVSN and the BAPN for the different cases are shown in Figure 6. For clarity, in all the figures, the predicted errors have been normalized with respect to the errors for the BAPN. When normalized in this manner, BAPN always has an average error of 1.0. The results shown in Figure 6 indicate that the behavior of k-fold cross-validation tends to be erratic. Difference between the MAPE of CVSN and BAPN ranged from 0% to as high as 130%. There was also no apparent correlation between the performance of CVSN and the SNN with respect to sample size and noise levels. The reason for this behavior is that there are a number of other factors that can influence the level 1 data generated. The random partitioning of the data set into k data sets for cross-validation and the random weight initialization used in neural network training strongly influence the level 1 data. Moreover, it is known cross-validation estimate of prediction error can show high variance (Weiss and Kukilowski, 1991). These factors make it difficult to discern any systematic behavior in the performance of the CVSN and the SNN with respect to sample size and noise. However, these factors equally affect both SNN and CVSN, since they both make use

of the level 1 data. Therefore, it is easier to compare the relative performance of SNNs with respect to CVSNs. In the following discussion we focus attention on the relative performance of SNN with respect to the CVSN and how it is affected by noise and sample size.

Figures 6(a) shows the performance comparisons of CVSN and SNN to BAPN for a sample size of 50, at different noise levels. The performance of CVSN for this sample size is good for low noise levels, however it performed very poorly at high noise. For a noise level of 2.5%, a reduction of 32% in error was obtained using SNN, as compared to using the BAPN. At higher noise levels SNN produced errors greater than BAPN. However, cross-validation performed very poorly for these cases. For 10% and 12.5% noise, the CVSN produced 130% and 67% greater error than BAPN. Poor performance of k-fold cross-validation implies that the level 1 data does not provide a reliable indication of model accuracy. Inaccurate level 1 data also affects the SNN adversely. SNN produced 36% and 18% greater error than BAPN. Nonetheless, SNN was significantly better than the CVSN: This shows that even when level 1 data is inaccurate, SNN performed better than the CVSN. When level 1 data is unreliable, cross-validation is inaccurate and it is difficult to decide which model is appropriate. Under these conditions it seems unreasonable to simply select one model and reject the others. SNN weights the other models, instead of simply using the wrong model. Averaging, as accomplished by stacking, seems to be beneficial under these circumstances. The results here demonstrate another important point: Performance of SNN and CVSN will be correlated. Correlation between the two methods is to be expected since SNN is using the same level 1 data as CVSN although in a different manner. The performance of both techniques depend on the accuracy of the level 1 data.

Figure 6(b) shows the results for a sample size of 70. For this sample size, SNN outperformed the BAPN, for all noise levels. Clearly no single model is optimal, and model integration accomplished by SNN finds a better model. Again, the correlation

between CVSN and SNN is noticeable. CVSN performs very well in all cases shown. At the most, it produces 13% greater error than BAPN, indicating that the level 1 data is accurate. SNNs makes better use the level 1 data to construct a stacked model that produced average errors which were 10-26% lesser than BAPN. Accurate level 1 data allowed the SNNs to successfully integrate the independent information provided by the various candidate models. Hence it appears that SNNs can outperform the BAPN when a number of competing models are good predictors, there is low correlation between the errors made by the models and the level 1 data is accurate. Figure 6(c) shows the results for sample size of 100. Cross-validation performs poorly at the low noise levels. It produces errors of up to 40% greater than BAPN. Maximum error for the stacked model is 15% greater than BAPN. Figure 6(d) shows the results for sample size of 120. Cross-validation produces models with up to 34% greater error than BAPN. SNNs, on the other hand performs as well as or better than BAPN, for all the cases. Figures 6(c) and 6(d) also show SNN and CVSN performance similar to Figures 6(a) and 6(b). Correlation between SNN and CVSN performance is clearly evident. As discussed earlier, such a correlation is to be expected. The interesting point is that the combination determined by the level 1 model consistently outperforms the single model selected by k-fold cross-validation.

To examine the robustness of SNNs with respect to varying noise levels, we plot the percent reduction in prediction error using stacking for different noise levels at several sample sizes (Figure 7). Stacking performance does not appear to be sensitive to noise since substantial improvements over CVSN could be obtained at all the noise levels used in this study. It can also be seen that the maximum improvement using SNN was produced at the smallest sample size of considered in this example (sample size of 50). This can be expected because variation between the models tends to be larger and errors made by different models showed lesser correlation for small data sets. The linear correlation between the errors made by the models ranged from 0.2 to 0.6 for a sample

size of 50. For the larger sample sizes, the corresponding correlation coefficient ranged from 0.4 to 0.8. Hence combining was most beneficial at the small sample size. Smaller data sets also cause greater uncertainty regarding model selection, hence stacking can be a safer strategy as opposed to selecting a single model.

The above results suggest that the SNN can consistently outperform the CVSN irrespective of the accuracy of the level 1 data. It appears that the SNNs make better use of the level 1 data, as compared to k-fold cross-validation. Of course, situations can arise when cross-validation can be better than the stacked modeling approach. For example, a poor level 1 generalizer can undermine the benefits of stacking. This has not been encountered by us, indicating the efficacy of the level 1 model used in this work. Since the level 1 model used in this example is linear, models with higher linear correlation to the actual output tend to be more heavily weighted. It is however possible that models whose outputs have low linear correlation could be important in a nonlinear sense. Such models cannot be effectively integrated by the linear level 1 model. It is important to identify and develop level 1 models which can develop nonlinear combinations of the candidate models. Identification of such general nonlinear level 1 models should further enhance the advantages of stacked modeling, and deserves further investigation.

Conclusions

In this work, a novel approach to empirical neural network based modeling of chemical processes has been proposed. Conventional approaches identify and use a single model for prediction. We have presented a novel architecture, stacked neural networks (SNNs), that effectively integrates the knowledge acquired by different networks to obtain a better predictive model. A technique for stacking neural networks has been developed and implemented. A linear model was used to stack the candidate networks. However any nonlinear model can also be used. Stacked modeling was illustrated using

a simple example and the performance of SNN was studied for two examples including the dynamic modeling of a chemical process. The examples demonstrate the feasibility of using stacked neural networks for improved modeling of chemical processes.

The main results of this work are summarized below:

1. Stacking consistently outperformed cross-validation for all the cases studied. The SNN approach described in this work appears to be a promising technique for empirical ANN based plant process modeling. The only disadvantage of this method appears to be the increased computation time associated with developing the combination.

2. The linear level 1 model used in this work was seen to be an effective method for combining the level 0 neural networks

3. For the dynamic modeling example, SNN performance did not degrade with increasing noise. SNN appears to be beneficial over a wide range of noise levels. Furthermore, the largest improvement using the SNN was obtained at the smallest sample size considered in the example.

The concept of stacked modeling can be easily extended to models other than linear combinations of backpropagation neural networks. Other empirical modeling methods like radial basis function modeling, polynomial regression, partial least squares can also be combined using the approach described in this work. A diversity of models used as the level 0 models should further enhance the performance of the present technique, since the errors produced by a diversity of models are less likely to be correlated. It is important to determine what other models can be combined with neural networks. Achieving optimal performance would also require effective combining rules. Identification of such rules is an important issue for further research, and should be examined.

Acknowledgments: This work was made possible by the generous support of the Minnesota Mining and Manufacturing Company. Their support is greatly appreciated, however it does not constitute an endorsement of the views expressed in this article.

References

Basu, A., and E. B. Bartlett, "Detecting Faults in a Nuclear Power Plant by Using Dynamic Node Architecture Artificial Neural Networks," *Nucl. Sci. Eng*, 116, 313 (1994)

Bates, J. M., and C. W. J. Granger, "The Combination of Forecasts," *Oper. Res. Q.*, 20(4), 451 (1969)

Bhat, N., and T. J. McAvoy, "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process System," *Comput. Chem. Eng.*, 14(4/5), 573 (1990)

Brieman, L., "Stacked Regressions," Tech. Rep. No. 367, Statistics Dept., Univ. of California at Berkeley (1992)

Cherkassky, V., Gehring, D., and Mulier, F., "Pragmatic Comparison of Statistical and Neural Network Methods for Function Estimation," *Proc. World Cong. on Neural Networks Conf.*, Vol. 2, Washington, DC, p. 917 (1995)

Clemen, R. T., "Combining forecasts: A Review and Annotated Bibliography," *Int. J. Forecasting*, 5(4), 559 (1989)

Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Math. Contr. Signals Syst.*, 2, 303 (1989)

De Veaux, R. D., D. C. Psichogios and L. H. Ungar, " A Comparison of Two Nonparametric Estimation Schemes: MARS and Neural Networks," *Comput. Chem. Eng.*, 17, 819 (1993)

Diebold, F. X., "Forecast Combining and Encompassing: Reconciling Two Divergent Literatures," *Int. J. Forecasting*, 5, 589 (1989)

DiMassimo, C., G.Montague, M. J. Willis, M. T. Tham, and A. J. Morris, "Towards Improved Pencillin Fermentation via Artificial Neural Networks," *Comput. Chem. Eng.*, 16, 382 (1992)

Economou, C. G., M. Morari and B. O. Palsson, "Internal Model Control. 5. Extension to Nonlinear Systems," *Ind. Eng. Chem. Process. Des. Dev.*, 25. 403 (1986)

Fahlman, S. E., and C. Lebiere, "The Cascade-Correlation Learning Architecture," Advances in Neural Information Processing Systems 2, D. S. Touretzky, ed., Morgan Kauffman, San Mateo, CA, p524 (1990)

Finnoff, W., H. Hergert, and H. G. Zimmerman, "Improving Model Selection by Nonconvergent Methods," *Neural Networks*, 6, 771 (1993)

Friedman, J. H., "Multivariate Adaptive Regression Splines," *Ann. Statist.*, 19, 1 (1991)

Geman, S., E. Bienenstock, and R. Doursat, "The Bias-Variance Dilemma," *Neural Comput.*, 4, 158 (1992)

Granger, C. W. J., and R. Ramanathan, "Improved Methods of Forecasting," *J. Forecasting*, 3, 197 (1984)

Granger, C. W. J., "Combining Forecasts: Twenty Years Later," *J. of Forecasting*, 8, 167 (1989)

Hansen, L. K., and P. Salamon, "Neural Network Ensembles," *IEEE Trans. Pattern Anal. and Mach. Intell.*, 12(10), 993 (1990)

Hashem, S., " Optimal Linear Combinations of Neural Networks," PhD. dissertation, Purdue Univ., West Lafayette, IN (1993)

Hornik, K., M. Stichcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, 2, 359 (1989)

Hoskins, J. C., and D. M. Himmelblau, "Artificial Neural Network Models of Knowledge Representation in Chemical Engineering," *Comput. Chem. Eng.*, 12, 881 (1988)

Joseph, B., F. H. Wang, and D. S. Shieh "Exploratory Data Analysis: A Comparison of Statistical Methods with Artificial Neural Networks," *Comput. Chem. Eng.*, 16, 413 (1992)

Kramer, M. A., "Autoassociative Neural Networks," *Comput. Chem. Eng.*, 16, 313 (1992)

Leonard, J. A., M. A. Kramer and L. H. Ungar, "A Neural Network Architecture that Computes its Own Reliability," *Comput. Chem. Eng.*, 16, 819 (1992)

MacGregor, J. F., T. E. Marlin, and J. V. Kresta, "Some Comments on Neural Networks and Other Empirical Modelling Methods," *Proc. CPC-IV Int. Conf. on Chemical Process Control*, Padre Island, TX, p.665 (1991)

Mah, R. S. H., and V. Chakravarty, "Pattern Recognition Using Neural Networks," *Comput. Chem. Eng.*, 16(4), 371 (1992)

Makridakis, S., C. Chatfield, M. Hibon, M. Lawrence, T. Mills, K. Ord, and L. F. Simmons, "The M2 Competition: A Real Time Judgementally Based Forecasting Study," *Int. J. Forecasting*, 9, 5 (1983)

McAvoy, T. J., N. S. Wang, and N. Bhat, "Interpreting Biosensor Data via Backpropagation," 1989 American Control Conference, Pittsburgh (1989)

Moller, M. F., "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, 6(4), 525 (1993)

Perrone, M. P., and L. N. Cooper, "When Networks Disagree: Ensemble Methods for Hybrid Neural Networks," *Neural Networks for Speech and Image Processing*, R. J. Mammone, ed., Chapman & Hall, London (1993)

Piovoso, M. J., and A. J. Owens, "Sensor Data Analysis Using Neural Networks," *Proc. CPC-IV Int. Conf. on Chemical Process Control*, Padre Island, TX, p.101 (1991)

Pollard, J. F., M. R. Broussard and D. B. Garrison, "Process Identification Using Neural Networks," *Comput. Chem. Eng.*, 16, 253 (1992)

Psichogios, D. C., and L. H. Ungar, "Direct and Indirect Model Based Control Using Artificial Neural Networks," *Ind. Eng. Chem. Res.*, 30, 2564 (1991)

Rogova, G., "Combining the results of several neural network classifiers," Neural Networks, 7(5), 777-781 (1994)

Sridhar, D. V., R. C. Seagrave, and E. B. Bartlett, "Improving Neural Network Based Predictive Modeling Using Stacked Generalization," *Intelligent Engineering Systems through Artificial Neural Networks*, Vol. 5, C. H. Dagli, M. Akay, C. L. P. Chen, and B. Fernandez, eds., American Society of Mechanical Engineers, St. Louis, MO (1995)

Venkatasubramanian, V., and K. Chan, "A Neural Network Methodology for Process Fault Diagnosis," *AIChE J.*, 35, 1993 (1989)

Weiss, S. M., and C. A. Kukilowski, "Computer Systems that Learn," Morgan Kaufman, San Mateo, CA (1991)

White, H., "Some Asymptotic Results for Learning in Single Hidden Layer Feedforward Network Models," *J. Amer. Stat. Assoc.*, 84, 1008 (1989)

Winkler, R. L., "Combining Forecasts: A Philosophical Basis and Some Current Issues," *Int. J. of Forecasting*, 5, 605 (1989)

Wolpert, D. H., "Stacked Generalization," *Neural Networks*, 5(2), 241 (1992)

Ydstie, B. E., "Forecasting and Control Using Adaptive Connectionist Networks," *Comput. Chem. Eng.*, 14, 583 (1990)

Table 1 Training data and cross-validation results for Example 1

Pattern	Training Data				Cross-validation		
	x	y	z	w	w_{1j}^{cv}	w_{2j}^{cv}	w_{3j}^{cv}
1	0	0	0	0	0.0	0.0	1.0
2	0	0	1	1	0.0	1.5	2.0
3	0	1	0	1	0.0	1.5	2.0
4	0	1	1	2	0.0	3.0	3.0
5	1	0	0	1	1.0	0.0	0.0
6	1	0	1	2	1.0	1.0	1.0
7	1	1	0	2	1.0	1.0	1.0
8	1	1	1	3	1.0	2.5	2.0

Table 2 Comparisons of models for Example 1

Model	E(MSE)	MSE (Test Data)	r (Test Data)
w=x	1.50	1.33	0.57
w=1.33y+1.33z	0.59	0.36	0.81
Nearest Neighbor	1.00	0.50	0.70
Stacked model	0.02	0.08	0.99

Table 3 Comparisons of models for Example 2

Network	RMSE (Data Set S_2)	RMSE (Data Set S_3)	NRMS on S_3
6x5x1	6.24	5.13	0.32
6x10x1	6.06	5.33	0.34
6x20x1	14.15	12.60	0.80
6x40x1	12.54	11.50	0.73
SNN	5.40	3.94	0.25

Table 4 Constants and Steady-State Operating Conditions for the CSTR

τ	60s
c_1	5x10^3 s^{-1}
c_{-1}	1x10^6s^{-1}
Q_1	10000 cal mol^{-1}
Q_{-1}	15000 cal mol^{-1}
R	1.987 cal mol^{-1} K^{-1}
ΔH_R	5000 cal mol^{-1}
ρ	1 Kg/L
C_p	1000 cal kg^{-1} K^{-1}
A_i	1.0 mol/L
R_i	0.0 mol/L
A_o	0.492 mol/L
R_o	0.508 mol/L
T_i	427K
T_o	430K

Table 5 Comparison of the models for the continuous stirred tank reactor data

Sample size	% Noise	CVSN	MAPE (CVSN)	MAPE (SNN)	%reduction in MAPE using SNN
50	2.5	N_2	77.7	48.7	37
50	5.0	N_3	92.9	67.5	27
50	7.5	N_2	125.2	89.4	29
50	10.0	N_2	130.1	77.2	41
50	12.5	N_3	90.1	63.4	29
70	2.5	N_4	74.7	59.6	20
70	5.0	N_3	67.0	54.4	19
70	7.5	N_3	67.8	53.8	20
70	10.0	N_3	73.5	48.8	34
70	12.5	N_4	67.9	60.0	12
100	2.5	N_3	92.9	78.6	15
100	5.0	N_4	103.5	81.0	22
100	7.5	N_2	83.3	77.6	7
100	10.0	N_2	80.9	69.4	14
100	12.5	N_2	76.3	65.8	14
120	2.5	N_5	94.6	72.1	24
120	5.0	N_4	95.0	68.7	28
120	7.5	N_2	95.3	73.6	23
120	10.0	N_3	73.4	58.0	21
120	12.5	N_2	63.8	58.7	8

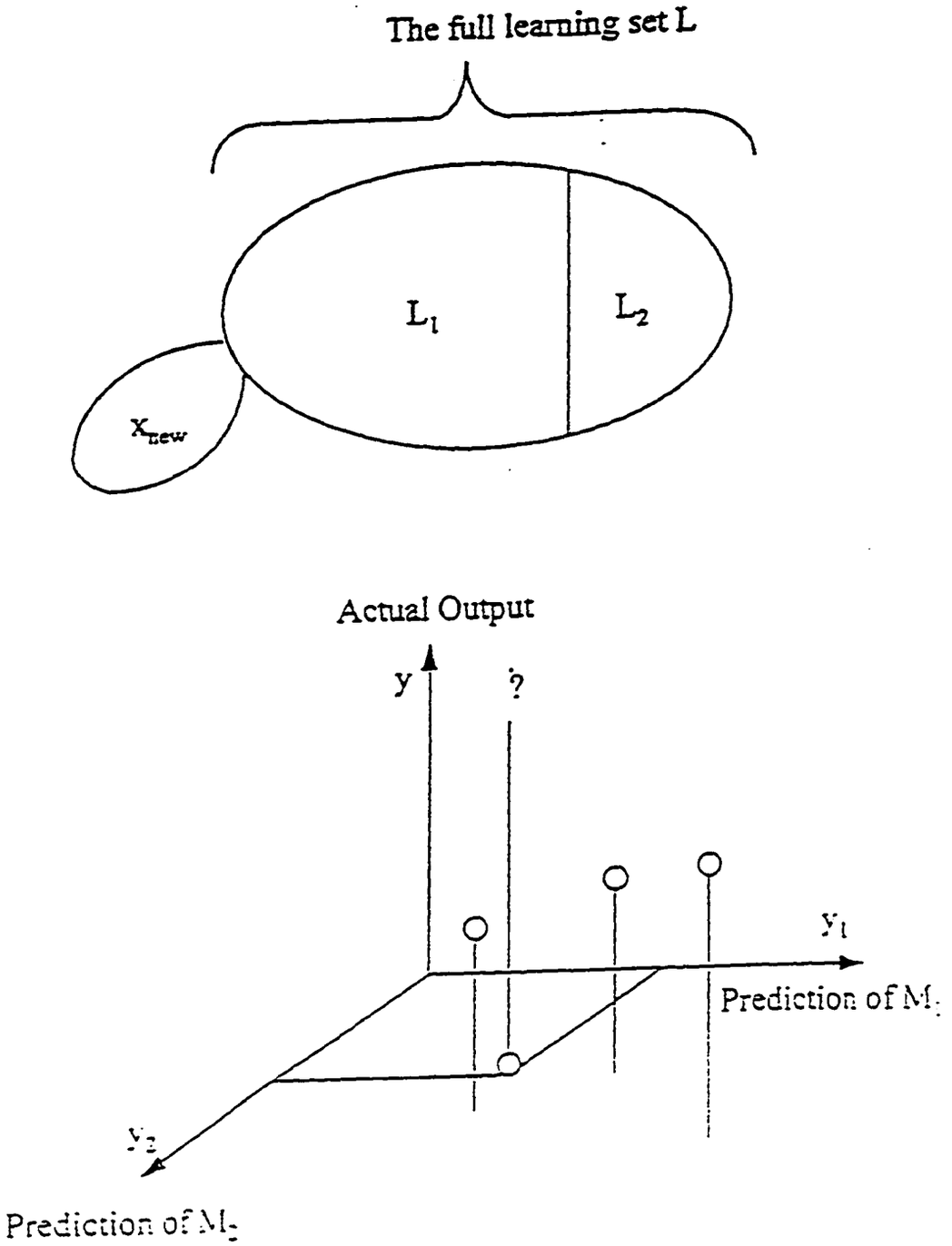


Figure 1 An illustration of stacked generalization

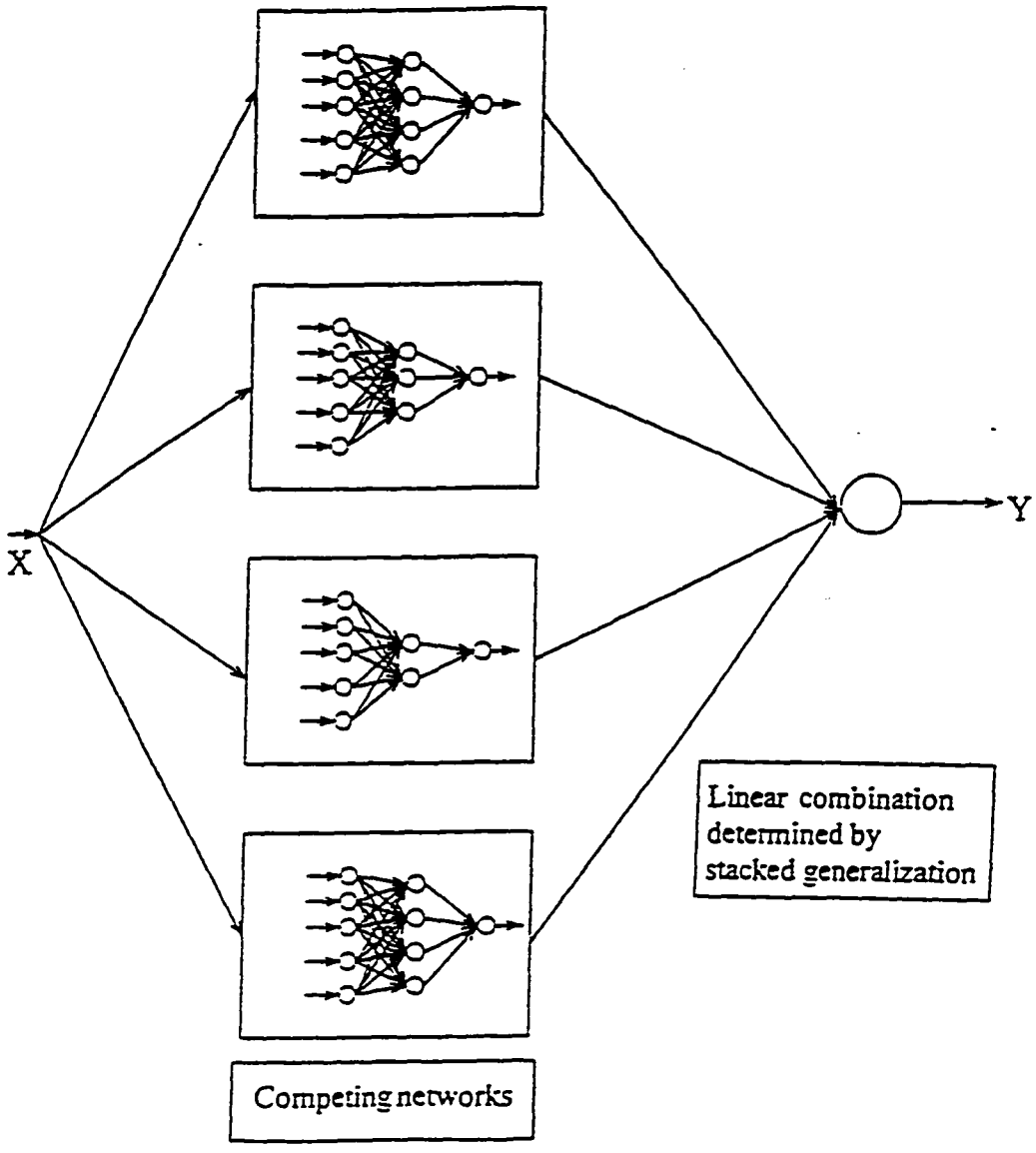


Figure 2 Architecture for stacked neural networks

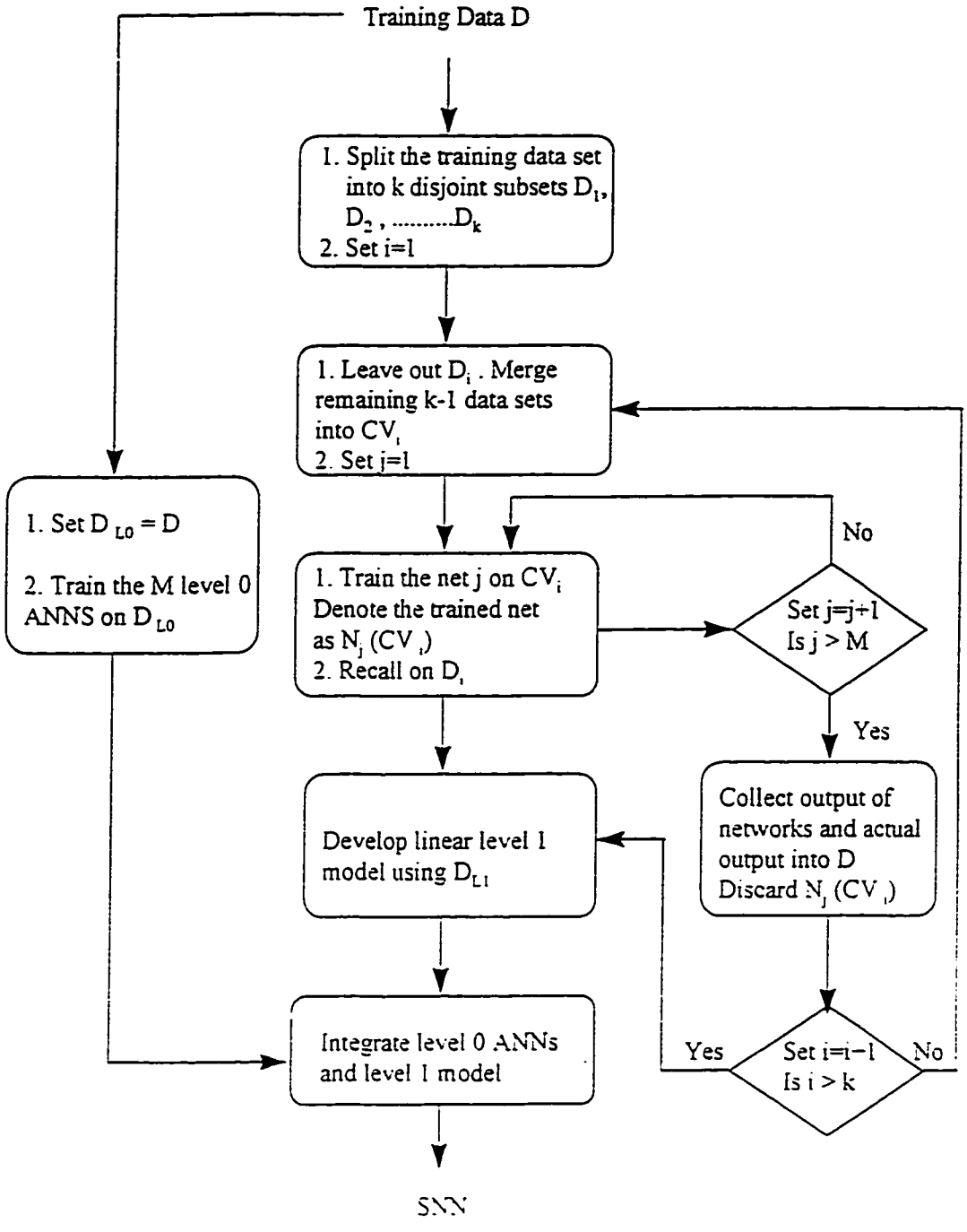


Figure 3 Schematic for developing the stacked neural network

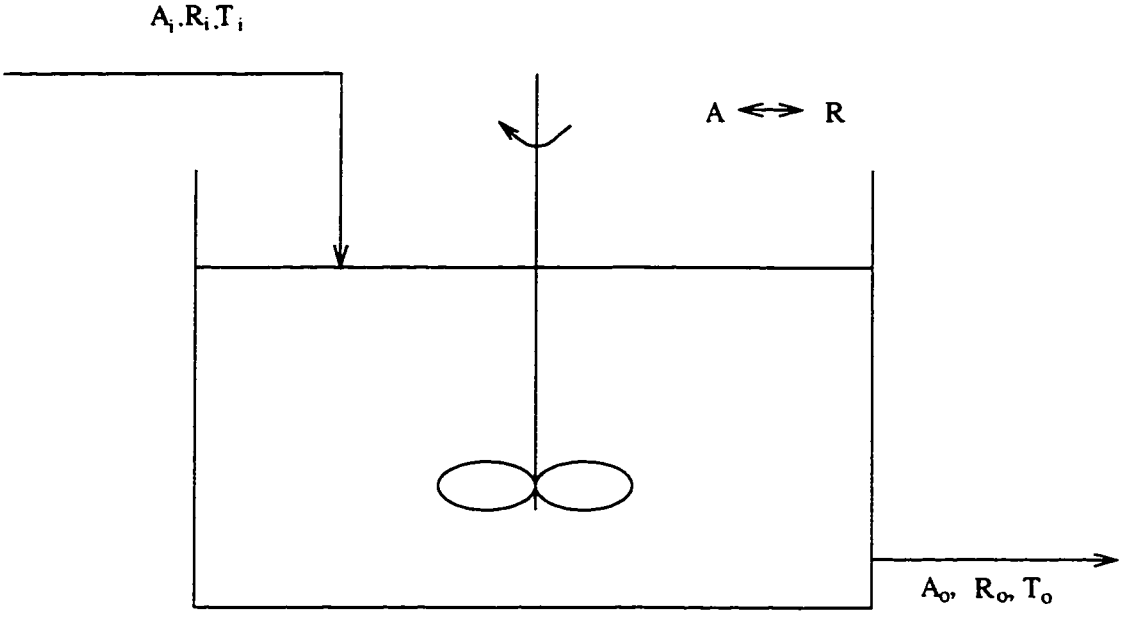


Figure 4 Schematic diagram of CSTR

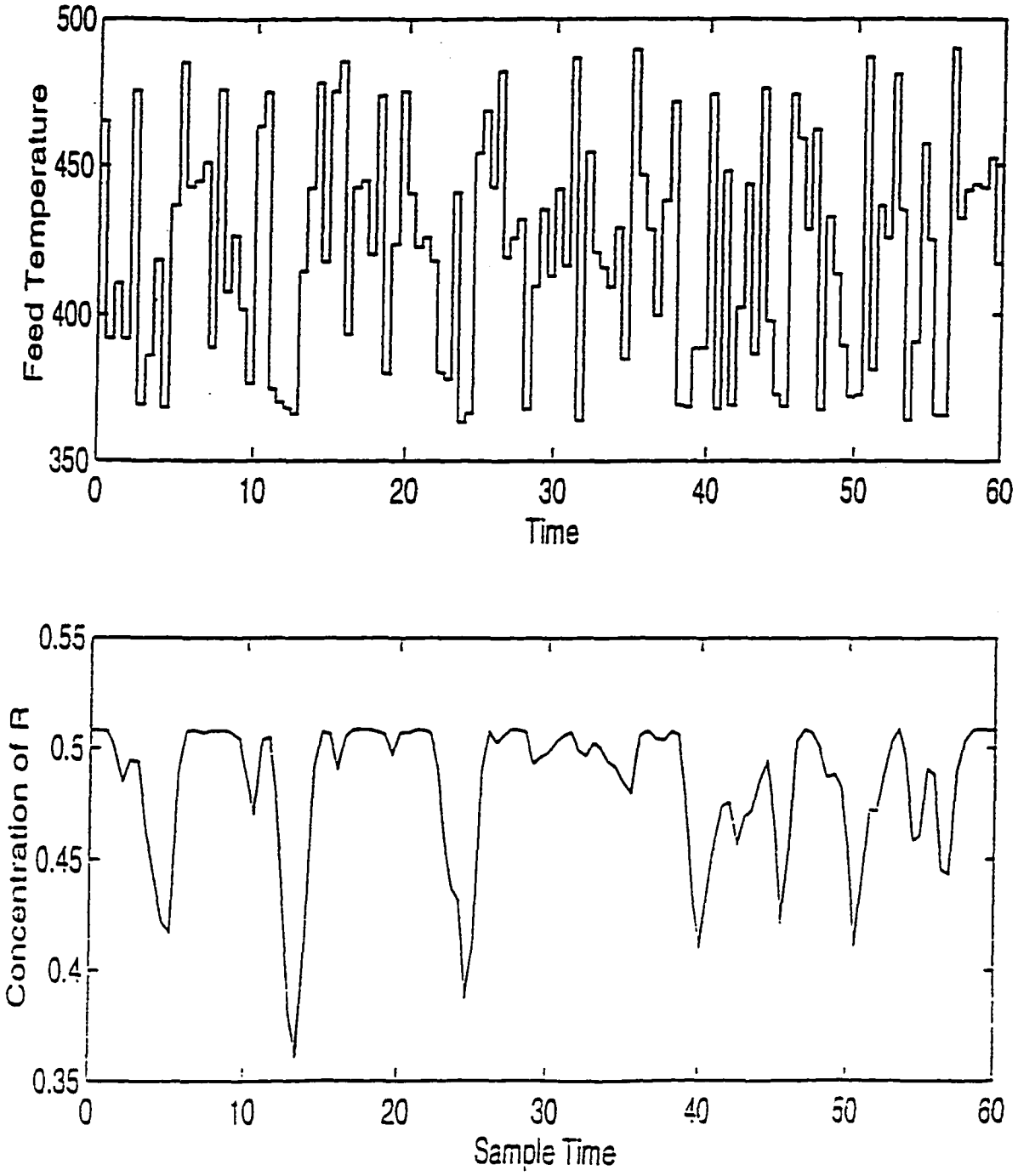
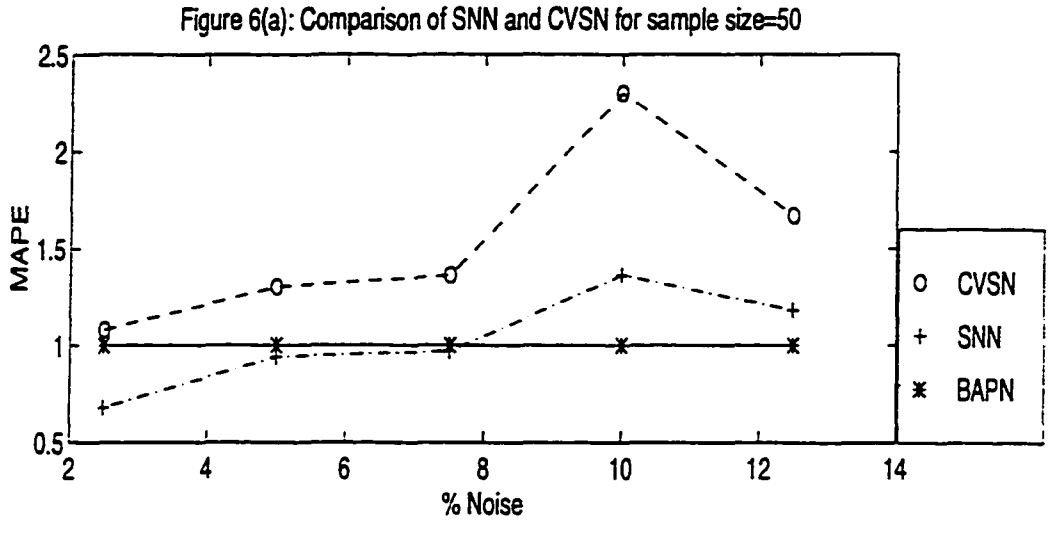


Figure 5 Feed temperature sequence and concentration response of CSTR



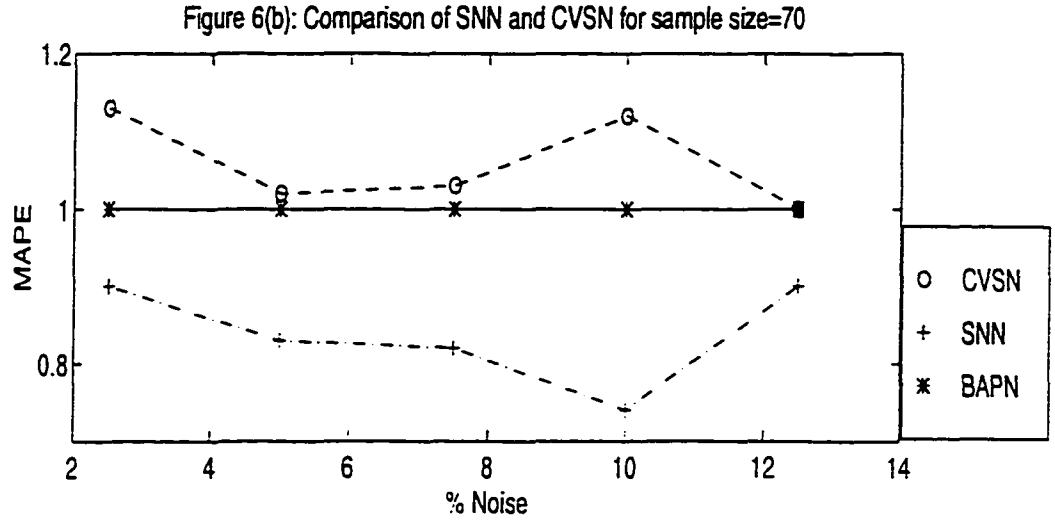


Figure 6 Comparison of "SNN" with "BAPN" and "CVSN"

Figure 6(c): Comparison of SNN and CVSN for sample size=100

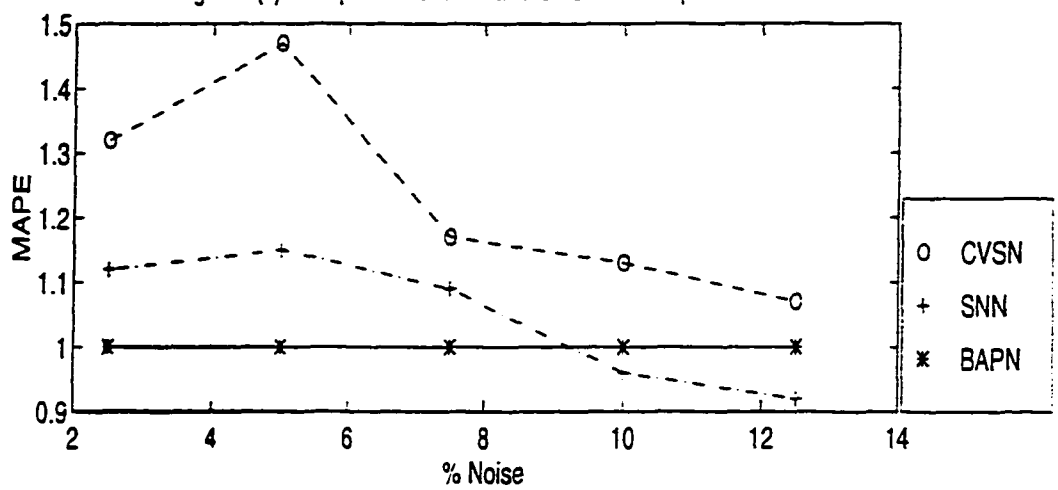


Figure 6(d): Comparison of SNN and CVSN for sample size=120

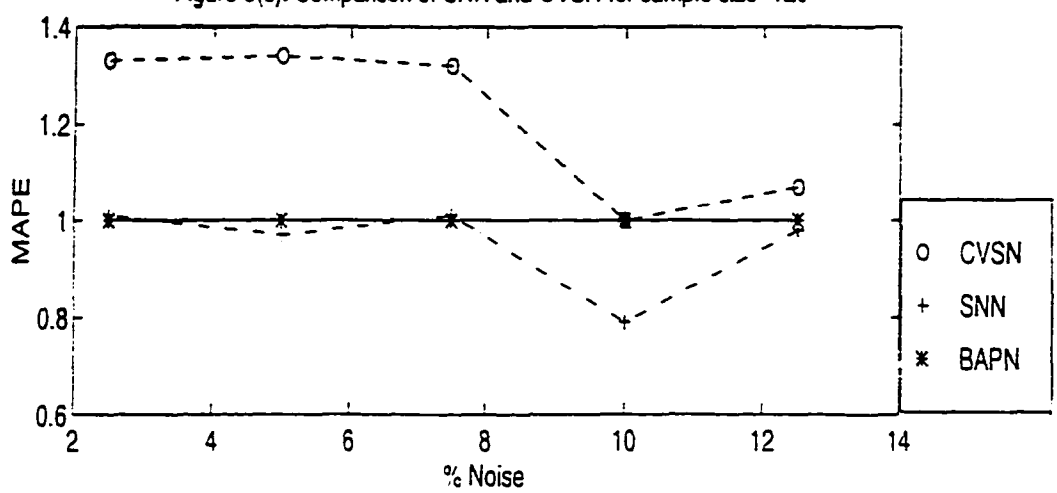


Figure 6 (continued)

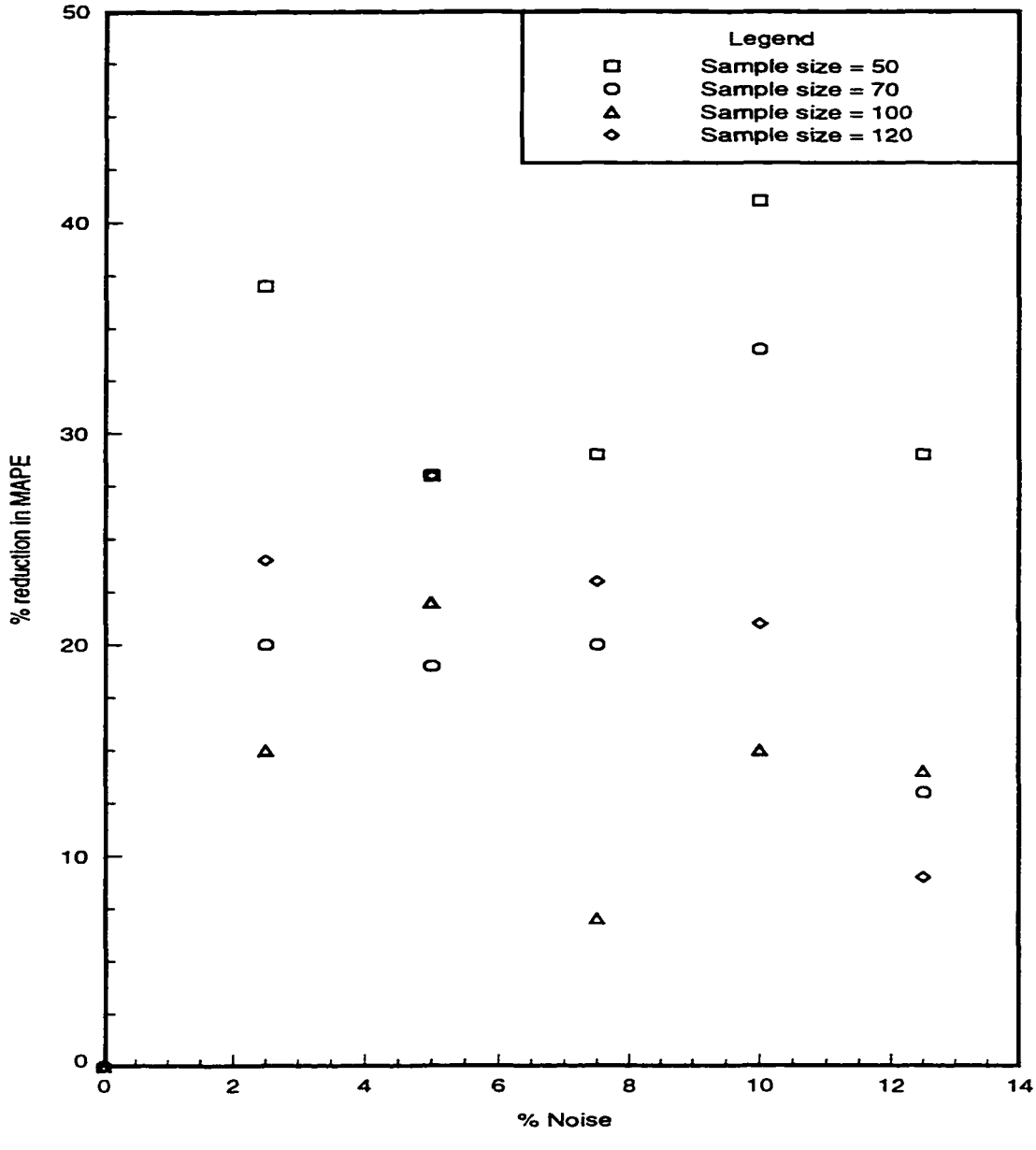


Figure 7 Reduction in MAPE using "SNN" vs. using "CVSN" for different noise levels

INFORMATION THEORETIC SUBSET SELECTION FOR NEURAL NETWORK MODELS

A paper submitted to the Computers and Chemical Engineering journal

Dasaratha Sridhar, Eric B. Bartlett and Richard C. Seagrave

Abstract

In this work, an information theoretic subset selection (ITSS) scheme for neural network based modeling of chemical processes is proposed. Neural network models have proven to be useful empirical models for identifying complex nonlinear chemical processes. ITSS selects an informative subset to be used as input data for constructing a neural network model. ITSS can select useful subsets regardless of the dependencies between the process outputs and inputs and are thus appropriate for neural network based process modeling. The feasibility of the ITSS method is explored through its application to three example problems. Results obtained show that ITSS is capable of identifying subsets for developing viable ANN models. ITSS can often identify simpler neural models with better generalization and ease of interpretation, while reducing computational costs of network training.

Introduction

Industrial processes usually exhibit complex nonlinear behavior which may not be well understood. Engineers commonly use empirical models to approximate the behavior of such processes. Recently artificial neural networks (ANNs) have been applied for empirical modeling of chemical plant processes (Bhat and McAvoy, 1990; Pollard, 1992; Piovoso and Owens, 1991). These applications exploit the ability of the ANN to approximate arbitrarily complex functions (Hornik *et al.*, 1989; Cybenko, 1989). In addition, ANNs are able to generalize and are considered to be noise and fault-tolerant (Lippman, 1987).

In practice, several factors may limit the successful application and development of a neural network model. One of the critical factors is the dimensionality of the input space. The problems associated with high input dimensionality in neural network modeling are well recognized. A large number of inputs dramatically increases the computational cost of the learning phase (Judd, 1990). Judd shows that the neural network learning problem is non-polynomial time complete: As the number of input variables n increases, the cost of obtaining a solution increases faster than a polynomial of order n. The presence of a large number of inputs would also require a large number of network parameters to be estimated and can cause poor network generalization (Geman *et al.*, 1991). For example, it is known for radial basis function networks that the number of hidden units required increases exponentially with increasing input size (Haykin, 1994). As a result, there is a large increase in the number of weights to be estimated for the network with increasing input dimensionality. This could result in poor generalization. To avoid the problems associated with large input dimensionality, it is important to extract and train the ANN with only those input features relevant to the mapping to be learned. Identifying a subset of the input space that is sufficient for developing the neural network model of interest is the focus of this work.

There are two approaches that are used for extracting the information relevant to a mapping to be learned by the neural network. The first approach is to project all the information in the original set of input variables to a lower dimensional space retaining most of the information. Principal component analysis (PCA) (Jolliffe, 1986) is one such technique for reducing the dimensionality of the input space. PCA transforms the n-dimensional input data vector into an n-dimensional vector of uncorrelated and mutually orthogonal principal components. Each principal component is a linear combination of the original input variables. Dimensionality is reduced by using only those principal components that account for a certain amount of the total variance in the original input space. Leen *et al.* (1990) used PCA to reduce the dimension of speech signals, and used the compressed signals to train a neural network for vowel recognition. They report significant reduction in training time, while maintaining the classification accuracy. PCA has also been used for extracting features from acoustic emission signals, which are then classified using a neural network (Yang and Dumont, 1991). Yang and Dumont also report significant reduction in network training time and size, without affecting classification accuracy. However. there are three main disadvantages to using principal component analysis. First PCA is a linear technique and nonlinear relationships between the inputs are ignored while estimating each principal component. Second, the variables in the principal component space do not have any physical meaning and are difficult to interpret. Third the principal components are simply transformations of the input variables, determined without taking the output variable into account. Hence the first few principal components may not neccesarily be the most important for predicting the output (Holcomb and Morari, 1992). Nonlinear principal component analysis (NLPCA) (Kramer, 1991) is a generalization of linear PCA as it can uncover and remove nonlinear relationships between the input variables. NLPCA operates by training a feedforward neural network to perform autoassociation: The network outputs are same as the network inputs. However the nonlinear principal components are even more complex and

difficult to interpret than the linear principal components. NLPCA is also a transformation of the input space and does not account for the output, while determining each nonlinear principal component. There is also the additional difficulty associated with training a neural network to accomplish NLPCA. The main advantage of both PCA and NLPCA is that the entire input space is used to determine the reduced set of variables. This is because both the linear and nonlinear principal components are combinations of the original set of input variables.

In contrast to PCA/NLPCA, the second approach for input variable dimensionality reduction attempts to identify a subset of the original input variables that are relevant to the ANN mapping. Such methods are appealing because the input variables that are used in the ANN model can be easily interpreted by engineers and process operators, as they are the original physical input variables. The idea of using a reduced subset of the original input variables for a neural network model has been investigated by Bhat and McAvoy (1992) and Karnin (1990). These methods require the network be first trained using all of the input variables, and finally the inputs irrelevant to the ANN mapping are eliminated. While these methods lead to smaller networks with improved generalization, the entire input set needs to be used to train the network at the beginning. When the input dimensionality is large, it can be difficult to train a network using the entire input vector, and the computational cost of these network pruning techniques will be high. Moreover, these methods will be affected by the learning algorithm used for ANN development. An alternative approach is to grow a network starting from just one input, considered to be the most important (Basu, 1995). However if the network is not provided with an appropriate set of inputs initially, valuable training time could be wasted in learning a mapping with an incomplete set of inputs.

The information theoretic subset selection approach (ITSS) described in this work, attempts to identify a subset of the input variables that contains most of the information in the input space relevant to the desired ANN mapping, prior to training the

ANN. Unlike methods based on linear correlation, no assumptions are made regarding the nature of the relationship between the outputs and the inputs. The central idea behind ITSS is to use information theory (Shannon and Weaver, 1949; Watanabe, 1969) to analyze the interdependency between process output and inputs. ITSS does not depend on the learning technique used, as the subset is identified prior to neural network model development. Information theoretic methods have been used elsewhere for subset selection in the context of neural network based classification (Battiti, 1994). Battiti's approach is based on assessing the importance of each individual feature, one at a time, for classification. While this approach is simple, variables that are jointly important for classification may not be identified. The ITSS method is applicable to continuous outputs and considers information jointly held by the input variables. Another limitation of Battiti's work is that no indication is provided as to when a selected subset is sufficient to model the output. Battiti's method requires the size of the subset desired to be specified *a priori*. In contrast, ITSS does not need the size of the subset to be specified. ITSS provides an estimate of the percentage of the total information contained in a subset with respect to the entire input vector. Any subset that contains a large percentage of the information in the entire input vector is a candidate subset to be used for ANN development. The results of using the ITSS are plotted on a cumulative information theoretic curve (CITC). The CITC shows the percentage information the candidate subsets contain relative to using the entire input vector to predict the output. The CITC provides an indication of which subsets contain most of the information in the complete input vector that is important for predicting the output. Process engineers can also use the CITC to obtain insight about the input variables that are most important for predicting the output, prior to model development.

Identifying a subset using ITSS can help alleviate the problem of poor generalization and poor estimation, while reducing training costs. It is also easier to examine a smaller network and extract information from it. For example. it is simpler to obtain insights

into the dynamics of a process using a smaller network (Bhat and McAvoy, 1992). There is also no need to limit the analysis to the subset that is generated using ITSS. It is possible that some variable important to the ANN model is different from that determined using the ITSS. The ITSS based approach can then be used as a starting point for network pruning and growing techniques. Network pruning need not be accomplished after training with the entire input vector and networks need not be grown using the time consuming procedure of starting from just one input and then adding one input at a time. Pruning techniques can be applied to a ANN model trained with the ITSS based subset or networks can be grown starting with the ITSS based subset. The approach used would depend on whether the subset used initially underestimates or overestimates the actual size of the input space required. In either case, initiating network training with an informative subset should provided significantly higher training speeds while developing a model with good generalization. Although subsets selected using ITSS have many potential benefits, optimality cannot be guaranteed. For instance, the ANN might extract information in a different manner when presented with the entire input vector rather than an ITSS-based subset. The ITSS based ANN model can always be compared to other competing models, using some model selection criteria. For instance, when it is feasible to develop the ANN using the entire input vector, we can compute the error of the ANN based on the full input vector and the ITSS-based ANN, over a test data set. In this manner, we can check if a better neural network model has indeed been identified using the ITSS approach. We will demonstrate this idea in one of the example problems provided in this paper.

The remainder of this paper is organized in the following manner. First information theoretic concepts are explained. The idea of subset selection using information theoretic methods is then discussed. Next the methodology for developing and using information theoretic subset selection (ITSS) and the cumulative information theoretic curve (CITC) is provided. The method is first illustrated using a simple example, fol-

lowed by application to two process identification problems. Results obtained show that ITSS can successfully identify the input variables important for predicting the output, regardless of the nature of their relationship to the output and prior to development of the neural network model. By identifying a smaller set of important input variables, ITSS can aid in the rapid development of neural networks with better generalization and ease of interpretation.

Information Theoretic Analysis

Overview

Information theory (Shannon and Weaver, 1949; Watanabe, 1969; Ash, 1990) is concerned with the engineering and analysis of communication systems. Our interest in information theory is in its ability to accomplish interdependency analysis (Press *et al.*, 1986). Information theoretic methods are useful for developing nonlinear measures of association between variables and have found application in neural network modeling. Examples include the development of training algorithms with relative entropy as the objective function (Bichsel and Seitz, 1989), for self-determination of input variable importance using neural networks (Bartlett, 1994a), and a dynamic node architecture learning to automatically determine the optimal hidden layer size for neural networks (Bartlett, 1994b).

Interdependency analysis using information theory

Shannon's information theory provides a formalism for quantifying the information content of any vector $\mathbf{x}$. It also allows measures of association to be developed between any two vectors $\mathbf{x}$ and $\mathbf{y}$. We first explain the basic concepts considering a discrete vector $\mathbf{x}$ that can take on M discrete values $\mathbf{x}_1, \mathbf{x}_2 ..... \mathbf{x}_M$. Define p_i as the probability

that $\mathbf{x}$ can take on the value $\mathbf{x}_i$.

$$p_i = p(\mathbf{x} = \mathbf{x}_i) \tag{1}$$

The probability p_i can be estimated using the frequency of occurrence of the vector $\mathbf{x}_i$

$$p_i = \frac{N_i}{N} \tag{2}$$

where N_i is the number of occurrences of the vector $\mathbf{x}_i$ in the data set and N is the total number of patterns in the data set. The entropy or information presented by the variable $\mathbf{x}$ can then be defined as

$$H(\mathbf{x}) = -\sum_{i=1}^{M} p_i ln(p_i) \tag{3}$$

If $\mathbf{x}$ can take only one value the information contained in $\mathbf{x}$ is 0. The information in $\mathbf{x}$ is maximized when there is equal probability of occurrence of each of the M possible vectors. The joint information content H($\mathbf{x}$,$\mathbf{y}$) of two vectors $\mathbf{x}$ and $\mathbf{y}$ can be defined analogous to Equation 3.

$$H(\mathbf{x},\mathbf{y}) = -\sum_{i,j} p_{ij} ln(p_{ij}) \tag{4}$$

where p_{ij} is the probability that $\mathbf{x}$ will take on the value $\mathbf{x}_i$ and $\mathbf{y}$ will take on the value $\mathbf{y}_j$. p_{ij} is given by the equation

$$p_{ij} = p(\mathbf{x} = x_i, y = y_j) = \frac{N_{ij}}{N} \tag{5}$$

where N_{ij} is the number of joint occurrences of the vectors x_i and y_j in the data set. The entropy of $\mathbf{y}$ given $\mathbf{x}$, H($\mathbf{y}$|$\mathbf{x}$), is a measure of the information in the vector $\mathbf{y}$ when $\mathbf{x}$ is known. H($\mathbf{y}$|$\mathbf{x}$) is given by:

$$H(\mathbf{y}|\mathbf{x}) = -\sum_{i,j} p_{ij} ln \frac{p_{ij}}{p_i} \tag{6}$$

It can be shown that

$$H(\mathbf{y}|\mathbf{x}) = H(\mathbf{x},\mathbf{y}) - H(\mathbf{x}) \tag{7}$$

An asymmetric dependency coefficient (ADC) that measures the dependency of y on x, U(y|x) can then be defined as

$$U(\mathbf{y}|\mathbf{x}) = \frac{H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x})}{H(\mathbf{y})} \tag{8}$$

The above equation can also be written as:

$$U(\mathbf{y}|\mathbf{x}) = \frac{H(\mathbf{y}) + H(\mathbf{x}) - H(\mathbf{x}, y)}{H(\mathbf{y})} \tag{9}$$

Equation 9 gives a very useful measure of association between the vectors y and x. U(y|x) does not depend on the nature of the functional relationship between y and x. U(y|x) measures the extent to which knowledge of x provides information about y. If U(y|x) is 0 it means that x does not contain any useful information about y, and it is not possible to predict y to any extent using x. A value of 1 implies that knowledge of x completely determines y, and it is possible to predict y exactly using x.

For real world problems, variables are often continuous rather than discrete. For a vector x of continuous variables, the entropy can be defined as

$$H(\mathbf{x}) = -\int p(\mathbf{x}) log(p(\mathbf{x}))dx \tag{10}$$

where p(x) is the probability density function of x. Equation 10 is difficult to evaluate since p(x) often needs to be estimated from the provided samples. However we can divide the input domain into a finite number of regions within which p(x) is assumed to be constant. Equation 10 can now be approximated by Equation 3, and it is then possible to evaluate the integral. Similarly we can compute H(y) and H(x,y) using Equation 3 and Equation 4, after dividing the corresponding vector domain into a finite number of regions. The $U(\mathbf{y}|\mathbf{x})$ for continuous variables can then be estimated from Equation 9.

Information theoretic subset selection

Before the information theoretic approach to subset selection is discussed, a few words about the idea of subset selection are in order. A comprehensive discussion on

subset selection for linear models can be found in the book by Miller (1990). Much of the ideas behind subset selection in the context of linear models can be extended to neural network models. While developing a neural network model one might be inclined to use all of the available input variables in the model. However as more input variables are used, a larger number of weights are needed in the neural network. Larger numbers of weights means that there is greater uncertainty in estimating the weights, and this uncertainty can lead to increases in the variance of model predictions. This can cause poor network generalization. Hence it is important to include only those variables which contain significant information regarding the output.

The central idea behind information theoretic subset selection (ITSS) is to find a vector $\mathbf{x}_s$ which is a subset of the original data vector $\mathbf{x}$ such that $\mathbf{x}_s$ contains almost all of the information in $\mathbf{x}$ that is important for predicting the output vector $\mathbf{y}$. Let the dimensionality of the input and output vectors be m and n, respectively. Denote a candidate subset of the original vector $\mathbf{x}$ as $\mathbf{x}_{sp}$, where p indicates the dimensionality of the subset input vector. Define the asymmetric dependency between the subset input vector and the output as $U(\mathbf{y}|\mathbf{x}_{sp})$. Define the asymmetric dependency between the entire input vector and the output as $U(\mathbf{y}|\mathbf{x})$. Then the objective of ITSS is to determine $\mathbf{x}_{sp}$ such that:

$$U(\mathbf{y}|\mathbf{x}_{sp}) - U(\mathbf{y}|\mathbf{x}) < \epsilon \tag{11}$$

where ϵ is deemed to be an acceptably small loss of information in the input space with respect to predicting the output. Equation 11 can be used to evaluate the loss of information by using any subset instead of the entire input vector. A general approach to information theoretic subset selection (ITSS) for ANN models is given below:

1. Calculate $U(\mathbf{y}|\mathbf{x})$.

2. Generate a candidate subset $\mathbf{x}_{sp}$ and determine $U(\mathbf{y}|\mathbf{x}_{sp})$

3. Use Equation 11 to check if the candidate subset is satisfactory.

4. If the candidate subset is satisfactory stop, else go to step 2.

5. Train an ANN on the selected subset.

6. Use some model selection criteria to determine if the ITSS-based ANN model is more acceptable that other competing models. One possible way to select a model is to compute the error of all the models over a test set, and select the model with the least estimated prediction error. For instance, when it is feasible to develop the ANN using the entire input vector, we can compute the error of an ANN based on the full input vector and the ITSS-based ANN, over a test data set. The ITSS-based ANN can then be selected if it has lower estimated prediction error.

In practice, two main issues need to be addressed to implement the ITSS algorithm. First of all, a method to estimate $U(\mathbf{y}|\mathbf{x}_{sp})$ from the sample data is required. Secondly, an algorithm to generate candidate subsets for evaluation is needed.

Computation of $U(\mathbf{y}|\mathbf{x}_{sp})$

Estimation of the asymmetric dependency requires an estimation of the probability density functions for $\mathbf{x}$, $\mathbf{y}$ and the joint probability density function for $\mathbf{x}$ and $\mathbf{y}$. A common approach is to approximate the probabilities using a large number of discrete "bins" and then counting the number of patterns in each bin (Bartlett, 1994). It is crucial not to use too many "bins" since random noise may be considered as important functional variations. On the other hand, not using enough bins can result in a loss of accuracy. Although the approach of using bins to estimate probabilities is simple, it has limitations. For example, consider a system with two variables x_1 and x_2 which are in the interval [0,1]. Suppose we set the number of information bins to 20 for both the variables. Then any variation above 0.05 in either x_1 or x_2 would be considered

important information. The problem with this approach is that the binning criteria is applied to the variables one at a time. Consider a input pattern P_1 (x_1, x_2). Consider another input pattern P_2, $(x_1 + 0.04, x_2 + 0.04)$. Applying the binning criteria would assume that P_2 is no different from P_1 and provides no useful information. However this may not be very reasonable, since we have looked at each variable one at a time, and not at the complete pattern. Based on the complete pattern, one might be inclined to decide that P_2 is a different pattern. Therefore, it is essential to include another criteria to judge if two patterns are different that accounts for joint variation in the input variables. Our approach is to form clusters based on the Euclidean distance as well as the bins to decide if a new pattern falls in a different cluster from a previous one. A pattern P_2 is deemed to fall in the same cluster as P_1 if each variable is in the same bin and the Euclidean distance is also less than a certain value. We have found this approach better estimates the probability density functions, and hence the information theoretic dependency measures.

Generation of candidate subsets

There are several well known search algorithms for generating candidate subsets. An obvious method is to use an exhaustive search over all the input variables; this however would be computationally infeasible except for small input vectors. A review of various search algorithms for input feature selection can be found in the book by Fukunaga (1990). The most commonly used techniques include forward selection and backward elimination. We briefly discuss how these methods can be applied in the context of information theoretic subset selection. The forward selection procedure generates a new subset by adding one input variable at a time to the current subset. At each stage, the variable to be added is selected so that the new enlarged input vector maximizes $U(\mathbf{y}|\mathbf{x}_{sp})$. The variable addition is terminated when the condition in Equation 11 is satisfied. In the backward elimination procedure we start with the entire input vector,

and delete one input at a time until further deletion of any variable results in the Equation 11 condition being violated. More sophisticated search procedures like the branch and bound algorithm also exist (Fukunaga, 1990). For the sake of simplicity we restrict attention to the forward selection algorithm in this paper.

We show the results of using the ITSS by plotting $U(\mathbf{y}|\mathbf{x}_{sp})$ against the number of variables in the subset. This plot will be referred to as the cumulative information theoretic curve (CITC). CITC not only demonstrates the progress of the variable selection approach but also shows the important input variables. Such a plot can be of great value to process operators and engineers since an idea of important process parameters can be obtained prior model development. An example of such a plot is shown in Figure 1. The CITC in Figure 1 shows the result of using ITSS on a process with four input variables and one output variable. The CITC shows that variable 4 is the single most important variable. Variable 4 along with variable 2 are the two variables that are jointly the most important. The curve also shows that variables 4 and 2 contain almost all of the information contained in the input space, that is important to predict the output. It is therefore clear to the modeler that variables 4 and 2 are important process input variables that ought to be included in any predictive model.

Example 1

The first example is intended to illustrate ITSS in a simple manner. The data set for this problem is shown in Table 1. The data set consists of 12 patterns with four inputs x_1, x_2, x_3, x_4 and one output y. All variables in this problem are discrete. x_1 can take on any one of three discrete values from the set $\{0,1,2\}$. x_2 and x_4 can take on values from the set $\{1,2\}$. x_3 is a function of x_2, given by

$$x_3 = x_2^2 \tag{12}$$

The dependent variable y is given by:

$$y = x_1 x_2 \tag{13}$$

Although the superficial dimensionality of the input space in this problem is 4, the output y can be predicted using only two of the four input variables. x_4 is a redundant variable. x_2 and x_3 are correlated and either one of these variables along with x_1, is sufficient to predict y.

ITSS was applied to this problem to identify subsets sufficient for modeling the system. Table 2 shows the results of using ITSS. In Stage I, ITSS considers the asymmetric dependency between the output and each individual variable. x_1 is selected as the most important variable with an ADC of 0.65. The reason that x_1 is the most important can be understood by examining the data set shown in Table 1. Knowing x_1, y can only take on two values. For example when x_1 is 1, y is either 1 or 2, and when x_1 is 2, y is either 2 or 4. Knowing x_2 however leaves us with three possible values for y. When x_2 is 1, y can take on the values 0, 1 and 2 and when x_2 is 2, y can take on the values 0, 2 and 4. Therefore knowledge of x_1 is more valuable in reducing the uncertainty in y, as compared to using x_2. Information theoretic methods provide us with a methodology to quantitatively express the above reasoning. x_2 and x_3 are shown to be equally important and both have an ADC of 0.17. The output variable y is related differently to x_2 as compared to x_3. However, it is obvious that both of them are equally good predictors of y. This has been correctly identified using the ITSS algorithm. x_4 has no information regarding the output as it has an ADC of 0.0. In Stage II, subsets with two variables are considered. One of the variables in any subset evaluated in Stage II is x_1 as it was the most important variable in Stage I. Therefore the subsets evaluated in Stage II are $\{(x_1, x_j), j = 2 \leq j \leq 4\}$. The subsets $\{x_1, x_2\}$ and $\{x_1, x_3\}$ both result in a $U(\mathbf{y}|\mathbf{x})$ of 1.0. The subset $\{x_1, x_4\}$ has the same information as $\{x_1\}$, as expected. ITSS has determined two subsets that contain all of the relevant information in the

four-dimensional input space to learn the required mapping. No assumption was made whatsoever regarding the nature of the mapping. The results of the ITSS procedure are shown on the CITC in Figure 2. CITC shows that x_1 is the single most important input variable for determining the output of the system. The curve also provides insight into the important groups of variables. As can be seen from Equation 13, there is a strong interaction between x_1 and x_2. The presence of such an interaction is clearly indicated in the CITC. An important point to note here is that ITSS cannot determine whether the causal variables are x_1 and x_2 or x_1 and x_3. In any case, this causality is impossible to determine from the data alone.

Example 2

The second example illustrates the application of ITSS to modeling of a nonlinear plant. The plant to be modeled is given by the following equation:

$$y_p(k+1) = f(y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)) \tag{14}$$

where $y_p(k+1)$ is the next time sample of the output of the plant, $y_p(k)$ is the current output, $y_p(k-1)$ and $y_p(k-2)$ are the output of the plant at the previous two time samples of the plant. The current input is u(k), and u(k-1) is the previous input. Denoting $y_p(k)$, $y_p(k-1)$, $y_p(k-2)$, u(k), u(k-1) as x_1, x_2, x_3, x_4 and x_5, respectively, the function f has the form

$$f[x_1, x_2, x_3, x_4, x_5] = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_2^3} \tag{15}$$

Narendra and Parthasarthy (1990) have used backpropagation neural networks (Werbos, 1994) to model the plant defined by Equations 14 and 15. Specht (1991) showed that the same plant could also be modeled using a General Regression Neural Network (GRNN). We demonstrate here that ITSS can be used to identify the important variables that affect the plant dynamics before model development is undertaken. Furthermore,

by using ITSS with the GRNN models, we will show that simpler models with better generalization can be obtained. The training data are generated in the same manner as described in Specht (1991). To generate the training data, a random input signal uniformly distributed in [-1,1] was used. The data collection was carried out for 1000 time steps. A large test data set of 10000 patterns was also generated in a similar manner. ITSS was applied to this problem to analyze the training data. Figure 3 shows the CITC obtained for this data. Input 4, or u(k), which is the input at the previous time step is the single most important variable. u(k) contains 62% of the information in the input space useful for predicting $y_p(k+1)$. u(k) and $y_p(k-1)$ jointly contain 74% of the useful information and u(k), $y_p(k-1)$ and $y_p(k-2)$ contain 92% of the information useful for predicting y. Therefore, three of the five variables account for a significant part of the dynamics. These results can be understood by examining the structure of Equation 15. We can rewrite Equation15 as below

$$f[x_1, x_2, x_3, x_4, x_5] = \frac{x_1 x_2 x_3 x_5 (x_3 - 1)}{1 + x_2^2 + x_2^3} + \frac{x_4}{1 + x_2^2 + x_2^3} = f_1[x_1, x_2, x_3, x_5] + f_2[x_2, x_3, x_4] \tag{16}$$

Note that the function can be represented as the sum of two functions f_1 and f_2. The first term f_1 does not depend on x_4, while the second term f_2 does not depend on x_1 or on x_5. Based on the results of the information theoretic analysis, we suspected that the contribution of f_1 to f would be small relative to the contribution of f_2. To verify whether this is true, we computed the average absolute value of f_1 for the entire training data set. This value was found to be 0.02. The plots of f and f_2 are shown in Figure 4. It is evident from the plot that there is not much difference between the two functions, f and f_2. This means that the plant dynamics can indeed be accurately represented to a large extent using f_2, which is a function of x_2, x_3, and x_4. ITSS has extracted this knowledge from the training data, without assuming the form of functional relationship and without using any prior knowledge about the plant. The analysis demonstrates the

capability of ITSS to identify the important variables that influence the plant dynamics.

We now discuss GRNN model development based on ITSS. Examination of the CITC shows that the subsets $\{x_4, x_3, x_2\}$, has $U(y|\mathbf{x})$ greater than 0.9 and was considered to be a candidate model. The ITSS based model as well as a GRNN model which uses all five inputs were developed using the approach discussed in Specht (1991). For comparison, we also developed a model based on $\{x_2, x_3, x_4, x_5\}$, which is the subset of four input variables identified by ITSS. For each model we minimized the prediction sums of square errors (PRESS) computed using leave one-out cross-validation (Weiss and Kukilowski, 1991). The performance of the models was evaluated by computing the mean square error (MSE) over the test data set. The results obtained are shown in Table 3. The model based on the subset $\{x_4, x_3, x_2\}$ is seen to have the least PRESS, and therefore is preferable to using the entire input vector. The MSE obtained on the test data confirm the results: Model 1 is indeed the best with an MSE of 0.044, which is 24% less than using Model 3, that is the full model. A graphical comparison of ITSS-based GRNN and the conventional GRNN model, for the first two hundred test patterns is shown in Figure 5. The ITSS-based GRNN is seen to be at least as effective as the full GRNN model in tracking the plant dynamics.

To examine the performance of ITSS at smaller training data sample sizes, the entire procedure was then repeated with sample sizes of 100 and 50. As before, the CITC was generated for both cases, and the results are shown in Figure 6 and Figure 7. For a sample size of 100, the subset $\{x_4, x_2\}$ had an ADC greater than 0.9 and were selected as an alternative to using the entire input set. The subsets $\{x_4, x_2, x_3\}$, and $\{x_4, x_2, x_3, x_5\}$ were larger candidate subsets identified by ITSS and were also considered for purposes of comparison. Similarly, for a sample size of 50, the subsets $\{x_4, x_2\}$, $\{x_4, x_2, x_1\}$, and $\{x_4, x_2, x_1, x_5\}$ were considered as candidate subsets. The entire model development process that was used previously with a sample size of 1000, was then repeated. The results for the sample sizes of 100 and 50 are shown in Table 4 and Table 5 respectively.

It is seen that the models based on ITSS lead to better generalization of the GRNN models. Model 2 which uses the subset x_4, x_2, x_3 has the least PRESS For a sample size of 100. On the test data, Model 2 has an MSE of 35% less than using Model 4 which used all of the input variables. Model 1 with only two input variables x_4, x_2 had the least PRESS for the sample size of 50. MSE on the test data was 0.09 for Model 1, a 26% reduction over using Model 4 with all of the input variables. The graphical comparisons between the ITSS-based GRNN and the conventional GRNN for sample sizes of 100 and 50 are shown in Figure 8 and Figure 9 respectively. Again it is seen that the ITSS-based GRNN is at least as effective as the full GRNN model.

These results show that the ITSS based GRNN models lead to simpler models that consistently outperform the conventional GRNN models based on using the entire input vector for this problem. The example provided a good illustration of the advantages of modeling using ITSS-based subsets of the original input vector. It is very evident from Equation 15 that the plant output is a function of all the five input variables. However, better generalizers were developed using only two or three of the input variables as predictors. Although using all the input variables clearly has more information in this problem, this was offset by the increased input dimensionality and increased model complexity. This is because a larger number of weights are used in a GRNN model with higher dimensionality. Generating of subsets based on ITSS, allowed us to reduce dimensionality of the input space by using only the most important input variables, which in turn helped identify better predictive models than the models based on using the entire input set.

Modeling the dynamics of a pH CSTR

This example is intended to demonstrate the performance of information theoretic subset selection (ITSS) on a more practical chemical process modeling problem. The

problem, presented by Bhat and McAvoy (1990), is to model the dynamic response of pH in a continuous stirred tank reactor (CSTR). The pH tank is known to be a first-order system with a highly nonlinear gain. The pH CSTR, shown in Figure 10, has two input streams, one containing sodium hydroxide and the other containing acetic acid (HAC). Stream 1 has a flowrate of F_1 and an HAC concentration of C_1. Stream 2 has a flowrate of F_2 and an NaOH concentration of C_2. The concentration of Na^+ and total acetate (HAC + AC^-) are ζ and ξ, respectively. The system is simulated by the following set of equations.

$$\begin{aligned} V\frac{d\xi}{dt} &= F_1C_1 - (F_1+F_2)\xi \\ V\frac{d\zeta}{dt} &= F_2C_2 - (F_1+F_2)\zeta \\ K_a &= \frac{[AC^-][H^-]}{[HAC]} \\ K_w &= [H^+]+[OH^-] \\ \zeta+[H^+] &= [OH^{-1}]+[AC^-] \end{aligned} \tag{17}$$

The steady state operating point and the process parameters are as given in the paper by Bhat and McAvoy (1990), and are shown in Table 6. The objective is to identify a process model that can predict the future pH in the CSTR in response to changes in the flowrate of the input stream F_2. Sampling time for the process was 0.2 minutes. A database was developed by randomly varying F_2 within 10% of the steady state operating point. The simulation was run for four hours. The data for the first hour was used as the training data set. The data collected over the next three hours were used as test data.

It is assumed that the model order and the process dead-time are unknown. As in Bhat and McAvoy (1992), the past four values of the pH and flowrate are considered as inputs for the model. Therefore the model to be fit is of the form

$$pH(t+1) = f(q(t-3), q(t-2), q(t-1), q(t), pH(t-3), pH(t-2), pH(t-1), pH(t)) \tag{18}$$

Before model development, ITSS was used to analyze the data set. The CITC for the pH tank system is shown in Figure 11. Variable 8, that is pH(t), is seen to be the most important input variable as it contains 73% of the information important for predicting the pH at the next time step. The second most important variable is variable 4, which is q(t). pH(t) and q(t) together contain 97% of the information for predicting the output. Therefore, ITSS has correctly identified the system to be of first order prior to model development, without making any assumptions about the pH dynamics. A neural network model can be constructed with just pH(t) and q(t) as the inputs to the system. We used a backpropagation network with 10 hidden nodes. No attempt was made to determine the optimal number of hidden nodes, since our interest is in determining whether the appropriate number of nodes have been used in the input layer. A 2x10x1 (2 input nodes, 10 hidden nodes and 1 output node) backpropagation network was trained using the scaled conjugate gradient algorithm (Moller, 1993). The result of testing the network are shown in Figure 12. The ANN model is seen to closely follow the dynamics of the pH process. The test mean square error was 0.03. For comparison purposes, we also trained a network using all of the input variables. The number of hidden nodes for the full models was also chosen to be 10. The architecture of this network was 8x10x1. The mean square error for the full model over the test data set was 0.08. The performance of the full BPN model is shown in Figure 13. The generalization of the full model is marginally worse when compared to the ITSS-based ANN. The ITSS-based model has the additional advantage of having a simpler architecture, and required only 41 weights to be estimated. Using the full BPN required 101 weights to be estimated.

In their work, Bhat and McAvoy proposed the StripNet algorithm to identify the appropriate size of the input layer. They train a network with all eight inputs and then strip the network weights to an appropriate size by eliminating unnecessary inputs and hidden nodes. The StripNet algorithm determines that the same two inputs are important. By using ITSS, the appropriate inputs were determined before initiating

training. There is no need to use all the eight inputs in the beginning. The network needs to be trained only with the two inputs that are important for predicting the output. Network pruning algorithms like StripNet can always be used to strip the network of any unneccesary hidden nodes. However processing the data with ITSS will significantly reduce the computational burden on the network pruning methodology since the appropriate inputs are known *a priori*. In this example, the hidden node pruning would have to be applied to a 2x10x1 neural network, as compared to a 8x10x1 network.

The ITSS methodology can also be used with network growing approaches as in Basu (1995), or algorithms like cascade correlation (Fahlman, 1990) that simply grow the hidden layer. Basu's method initiates training with the single most important input. As training progresses, the other inputs are added. However, with the ITSS approach it has already been identified that two variables together contain most of the information. Hence training could be initiated with 2 input nodes rather than just one, which we know is insufficient. There is no need to spend time in attempting to train a model with just one input. In problems where a relatively large number of inputs are important, ITSS can be very valuable when used with network growing approaches. Since it determines an appropriate input vector prior to training, the ITSS can considerably reduce the computational burden on these network growing techniques.

We also experimented with ITSS to determine if the system could correctly identify the dead time in the pH dynamic system. As in Bhat and McAvoy (1992), six units of dead time was added to the simulation of the pH system. Assuming that the dead time is unknown, 16 inputs, which include the past eight values of the flow rate and the past eight values of the pH were considered. Although the superficial dimensionality for this problem has doubled as compared to the previous study with no dead time, the underlying dimensionality of the process dynamics has not changed. Application of ITSS results in the CITC shown in Figure 14. Variable 16, which is pH(t), is identified to be the most important variable as it contains 73% of the information. Addition of variable

2, which is q(t-6) results in a subset that contains 95% of the information. pH(t) and q(t-6) appear to be sufficient for modeling the dynamics of the system. Therefore ITSS has correctly identified the dead time and model order of the pH tank system and we need to use only two of the sixteen inputs provided to develop a neural network model. The StripNet algorithm, by comparison, would require a network with 16 input nodes to be trained, and then stripped to the appropriate size. As in the previous study, we trained a network using only the two important inputs and another network that used all of the 16 input variables. The mean square error (MSE) on the test data set was 0.031 for the ITSS-based network. In comparison, the MSE for the full model was 0.11. Hence, we again see that ITSS helped identify a simpler network with better generalization, while reducing computational costs.

Conclusions

In this work, a information theoretic subset selection (ITSS) scheme for neural network based modeling of chemical processes is proposed. The ITSS method uses information theory to analyze the interdependency between process outputs and inputs. The technique is general and can be used with any nonlinear empirical modeling approach. ITSS attempts to identify a subset of the input space that contains most of the information in the input space relevant to the desired mapping, prior to developing the neural network model. ITSS can select appropriate subsets regardless of the dependencies between the process outputs and inputs. The methodology considered in this work is based on the forward selection approach similar to the method used for development of linear regression models. However, the ITSS method is general and can be used with any subset selection scheme such as backward elimination or branch and bound methods. Identification of effective subset selection schemes that can be used with ITSS, is an important subject for future work. The results of using the ITSS are presented on

a cumulative information theoretic curve (CITC), which provides insight on the input variables that are collectively important, to predict the output. The CITC can provide the process engineers and operators with a useful estimate of the importance of different variables groups, prior to model development. The CITC also provides an indication of which subsets contain most of the information in the complete input vector.

The application of the ITSS approach was first illustrated using a simple example. Then the feasibility of the ITSS method was explored through its application to two process identification problems, including the dynamic modeling of a nonlinear chemical reactor. Results obtained show that ITSS is capable of identifying subsets for developing viable ANN models. In both the process identification examples, we found that processing the data using the ITSS algorithm provided us with insight into the process dynamics prior to model development. Application of ITSS is seen to lead to simpler neural network models, that often show improved generalization, while greatly reducing the computational burden associated with network training. Simpler models are easier to analyze and interpret, which is of immense value to process engineers and operators. The only limitation of the ITSS approach is that the success of the method depends on the availability of sufficient training data which is, in any case, needed to develop an effective empirical model. We have also discussed how it would be beneficial to integrate the ITSS approach with network growing or pruning techniques. Investigation of such an approach is an interesting avenue for further research.

Acknowledgments: This work was made possible by the generous support of the Minnesota Mining and Manufacturing Company. Their support is greatly appreciated, however it does not constitute an endorsement of the views expressed in this article.

References

Ash, R. B., "Information Theory," Dover Publications, New York (1990)

Bartlett, E. B., "Self Determination of Input Variable Importance Using Neural Networks," *Neural, Parallel and Scientific Computation*, 2, 103 (1994a)

Bartlett, E. B., "Dynamic Node Architecture Learning: An Information Theoretic Approach," *Neural Networks*, 7, 129 (1994b)

Basu, A., "Nuclear Plant Diagnostics Using Neural Networks with Dynamic Input Selection," PhD. dissertation, Iowa State University, Ames, IA (1995)

Battiti, R., "Using Mutual Information for Selecting Features in Supervised Neural Net Learning," *IEEE Trans. Neural Networks*, 5, 537 (1994)

Bhat, N., and T. J. McAvoy, "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process System," *Comput. Chem. Eng.*, 14(4/5), 573 (1990)

Bhat, N. V., and T. J. McAvoy, "Determining Model Structure for Neural Models by Networks Stripping," *Comput. Chem. Eng.*, 16, 271 (1992)

Bichsel, M., and M. Seitz, "Minimum Class Entropy: A Maximum Information Theoretic Approach to Layered Networks," *Neural Networks*, 2, 133 (1989)

Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Math. Contr. Signals Syst.*, 2, 303 (1989)

Fahlman, S. E., and C. Lebiere, "The Cascade-Correlation Learning Architecture," Advances in Neural Information Processing Systems 2, D. S. Touretzky, ed., Morgan Kauffman, San Mateo, CA, p524 (1990)

Fukunaga, K., "Introduction to Statistical Pattern Recognition," Academic Press, Boston (1990).

Geman, S., E. Bienenstock, and R. Doursat, "The Bias-Variance Dilemma," *Neural Computation*, 4, 158 (1992)

Haykin, S., "Neural Networks: A Comprehensive Foundation," Macmillan College Publishing Company, New York (1994)

Holcomb, T. R., and M. Morari, "PLS/Neural Networks," *Comput. Chem. Eng.*, 16(4), 393 (1992)

Hornik, K., M. Stichcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, 2, 359 (1989)

Jolliffe, I. T., "Principal Component Analysis," Springer-Verlag, New York (1986)

Judd, J. S., "Neural Network Design and the Complexity of Learning," The MIT Press, Cambridge, MA (1990)

Karnin, E. D., "A Simple Procedure for Pruning Back-propagation Trained Neural Networks," *IEEE Trans. Neural Networks*, 1, 239 (1990)

Kramer, M. A., "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks," *AIChE J.*, 37, 233 (1991)

Leen, T. K., M. Rudnick, and D. Hammerstorm, "Hebbian Feature Discovery Improves Classifier Efficiency," *Proc. Int. Joint Conf. on Neural Networks*, Vol. 1, San Diego, CA, p.51 (1990)

Lippmann, R. P., "An Introduction to Computing with Neural Nets," *IEEE, Acoustics Speech and Signal Processing Magazine*, 4, 4 (1987)

Miller, A. J, "Subset Selection in Regression," Chapman and Hall, New York (1990)

Moller, M. F., "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, 6(4), 525 (1993)

Narendra, K. S., and K. Parthasarthy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. Neural Networks*, 1(1), 4 (1990)

Piovoso, M. J., and A. J. Owens, "Sensor Data Analysis Using Neural Networks," *Proc. CPC-IV Int. Conf. on Chemical Process Control*, Padre Island, TX, p.101 (1991)

Pollard, J. F., M. R. Broussard, and D. B. Garrison, "Process Identification Using Neural Networks," *Comput. Chem. Eng.*, 16, 253 (1992)

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "Numerical Recipes: The Art of Scientific Computing," Cambridge University Press, New York (1986)

Shannon, C. E., and W. Weaver, "The Mathematical Theory of Communication," University of Illinois Press, Urbana (1949)

Specht, D. F, "A General Regression Neural Network," *IEEE Transactions on Neural Networks*, 2, 568 (1990)

Watanabe, S., "*Knowing and Guessing: A Quantitative Study of Inference and Information*," John Wiley and Sons, New York (1969)

Weiss, S. M. and Kukilowski C. A., "Computer systems that learn," Morgan Kaufman, San Mateo, CA (1991)

Werbos, P. J., "The Roots of Backpropagation," John Wiley and Sons, New York (1994)

Yang, J., and G. A. Dumont, "Classification of Acoustic Emission Signals via Hebbian Feature Extraction," *Proc. Int. Joint Conf. on Neural Networks*, Vol. 1, Seattle, WA, p.113 (1991)

Table 1 Training data for Example 1

x_1	x_2	x_3	x_4	y
0	1	1	2	0
1	2	4	2	2
2	1	1	1	2
0	2	4	2	0
1	1	1	1	1
2	2	4	2	4
0	1	1	1	0
1	2	4	1	2
2	1	1	2	2
0	2	4	1	0
1	1	1	2	1
2	2	4	1	4

Table 2 Application of ITSS to the data set in Example 1

Stage	Candidate Subset	$U(y\|x_{sp})$	Subset selected
I	x_1	0.65	x_1
	x_2	0.17	
	x_3	0.17	
	x_4	0.00	
II	x_1,x_2	1.00	x_1,x_2
	x_1,x_3	1.00	x_1,x_3
	x_1,x_4	0.65	

Table 3 Application of ITSS in Example 2 for a sample size of 1000

Model#	Subset	U	PRESS	MSE (Test Data)
1	x_4,x_3,x_2	0.92	0.043	0.044
2	x_4,x_3,x_2,x_5	0.99	0.054	0.055
3	x_4,x_3,x_2,x_5,x_1	1.00	0.056	0.058

Table 4 Application of ITSS in Example 2 for a sample size of 100

Model#	Subset	U	PRESS	MSE (Test Data)
1	x_4,x_2	0.93	0.10	0.083
2	x_4,x_2,x_3	1.0	0.08	0.068
3	x_4,x_2,x_3,x_5	1.0	0.11	0.099
4	x_4,x_2,x_3,x_5,x_1	1.0	0.12	0.105

Table 5 Application of ITSS in Example 2 for a sample size of 50

Model#	Subset	U	PRESS	MSE (Test Data)
1	x_4,x_2	0.98	0.109	0.090
2	x_4,x_2,x_1	1.00	0.13	0.100
3	x_4,x_2,x_1,x_5	1.00	0.14	0.110
4	x_4,x_2,x_1,x_5,x_3	1.00	0.146	0.122

Table 6 Steady state operating condition of the pH CSTR

Parameters used in the simulation	Value
Volume of the tank	1000 l.
Flow rate of acetic acid	81 l min^{-1}
Steady state flow rate of NaOH	515 l min^{-1}
Steady state pH	7
Concentration of acetic acid	0.32 mol l^{-1}
Concentration of NaOH	0.05 mol l^{-1}
Initial concentration of sodium in the CSTR	0.0432 mol l^{-1}
Initial concentration of sodium in the CSTR	0.0435 mol l^{-1}

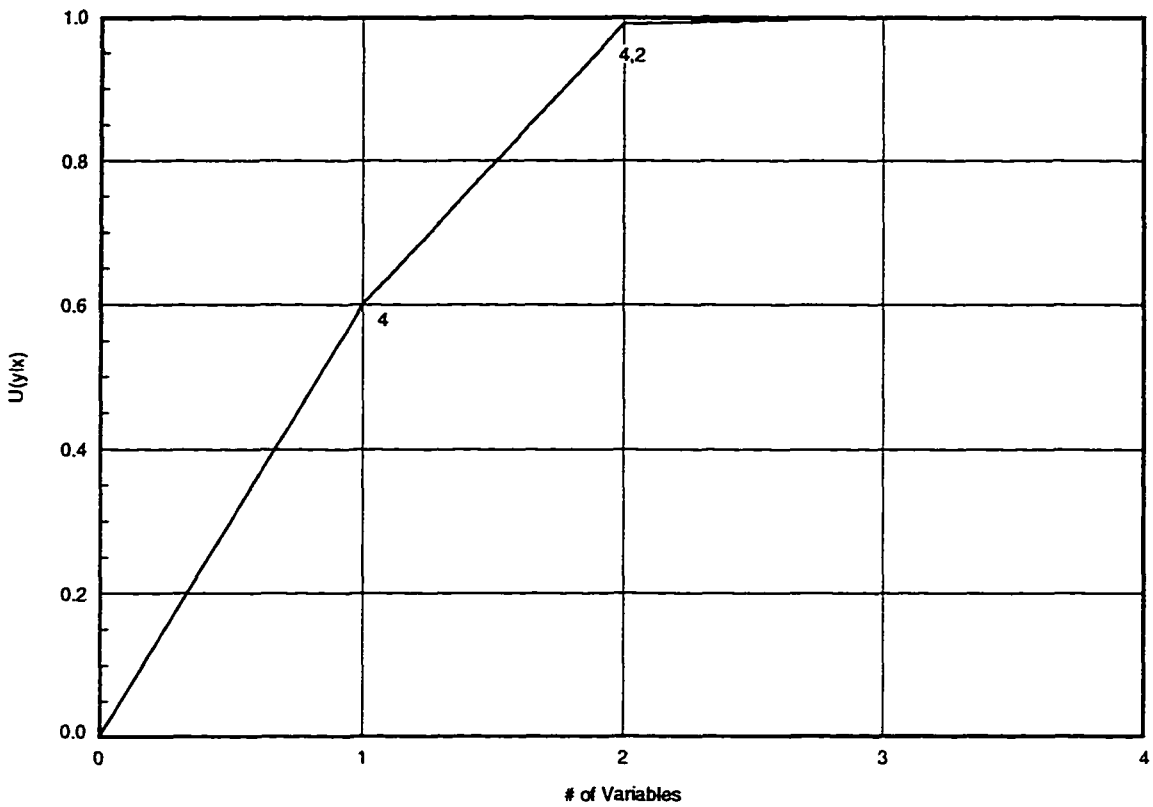


Figure 1 A sample Cumulative Information Theoretic Curve (CITC)

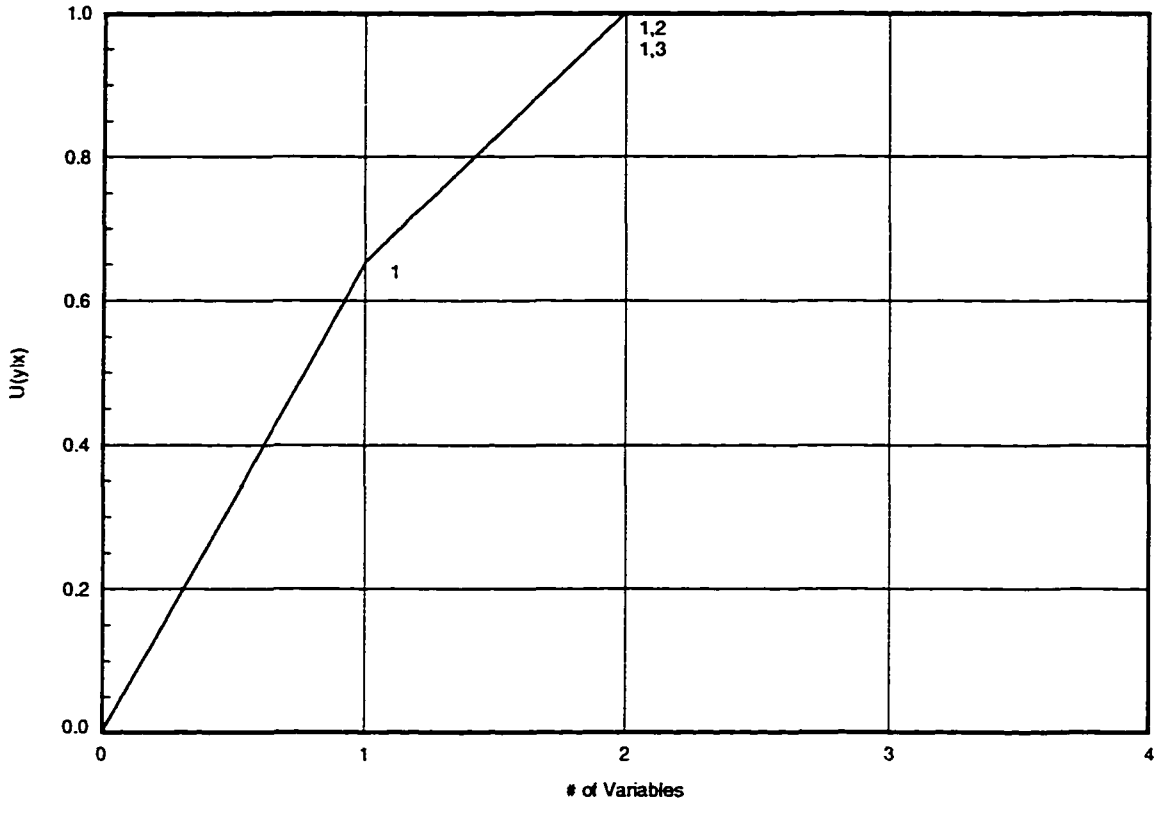


Figure 2 The Cumulative Information Theoretic Curve (CITC) for Example 1. Note that both subsets $\{x_1, x_2\}$ and $\{x_1, x_3\}$ have $U(y|\mathbf{x})$ of 1.0 and are sufficient to predict y

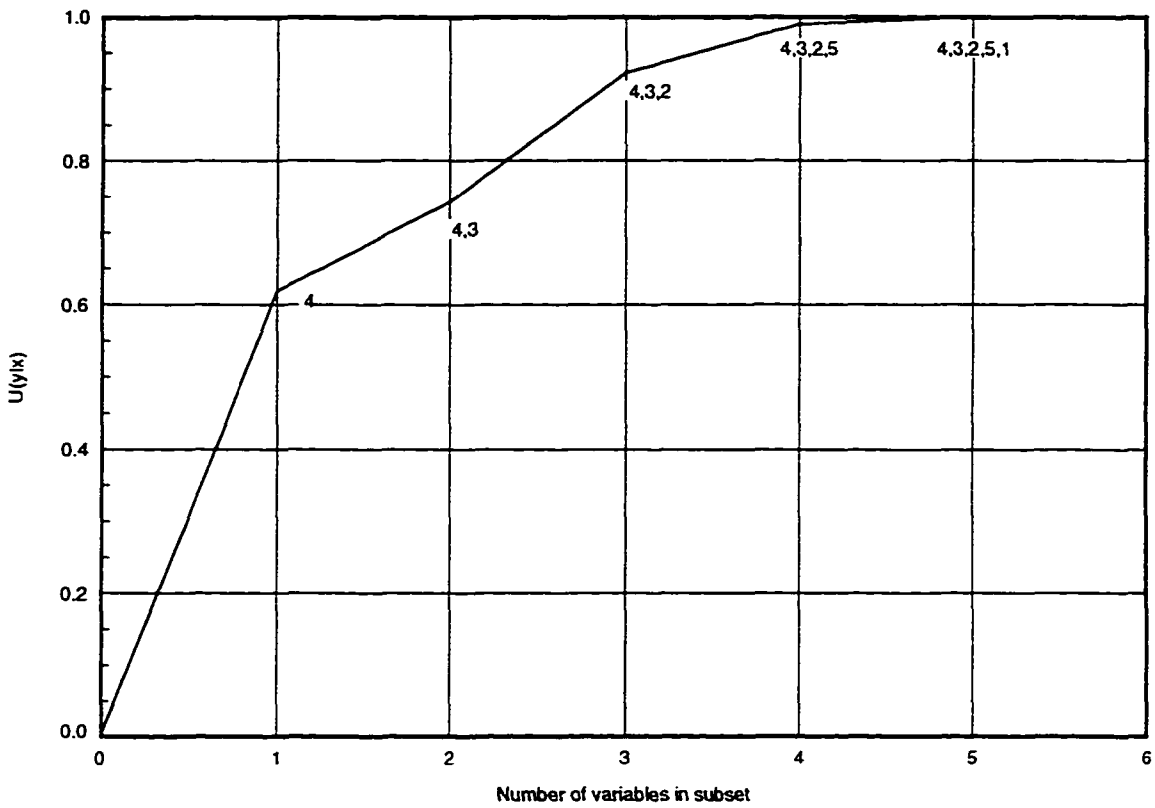


Figure 3 Example 2 CITC. for a sample size of 1000. Subset $\{x_4, x_3, x_2\}$ has $U(y|\mathbf{x})$ of 0.92 and is a candidate subset for the GRNN model

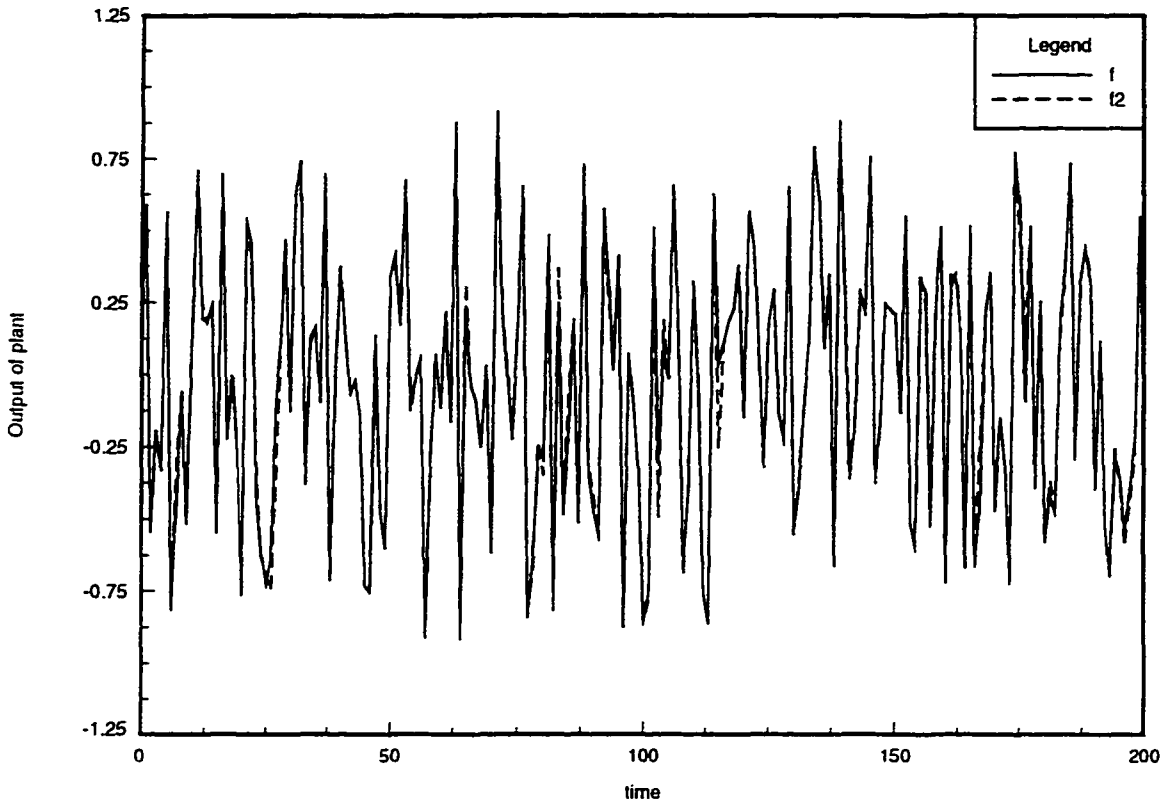


Figure 4 Plots of f and f_2 for Example 2, showing that the two functions are not very different

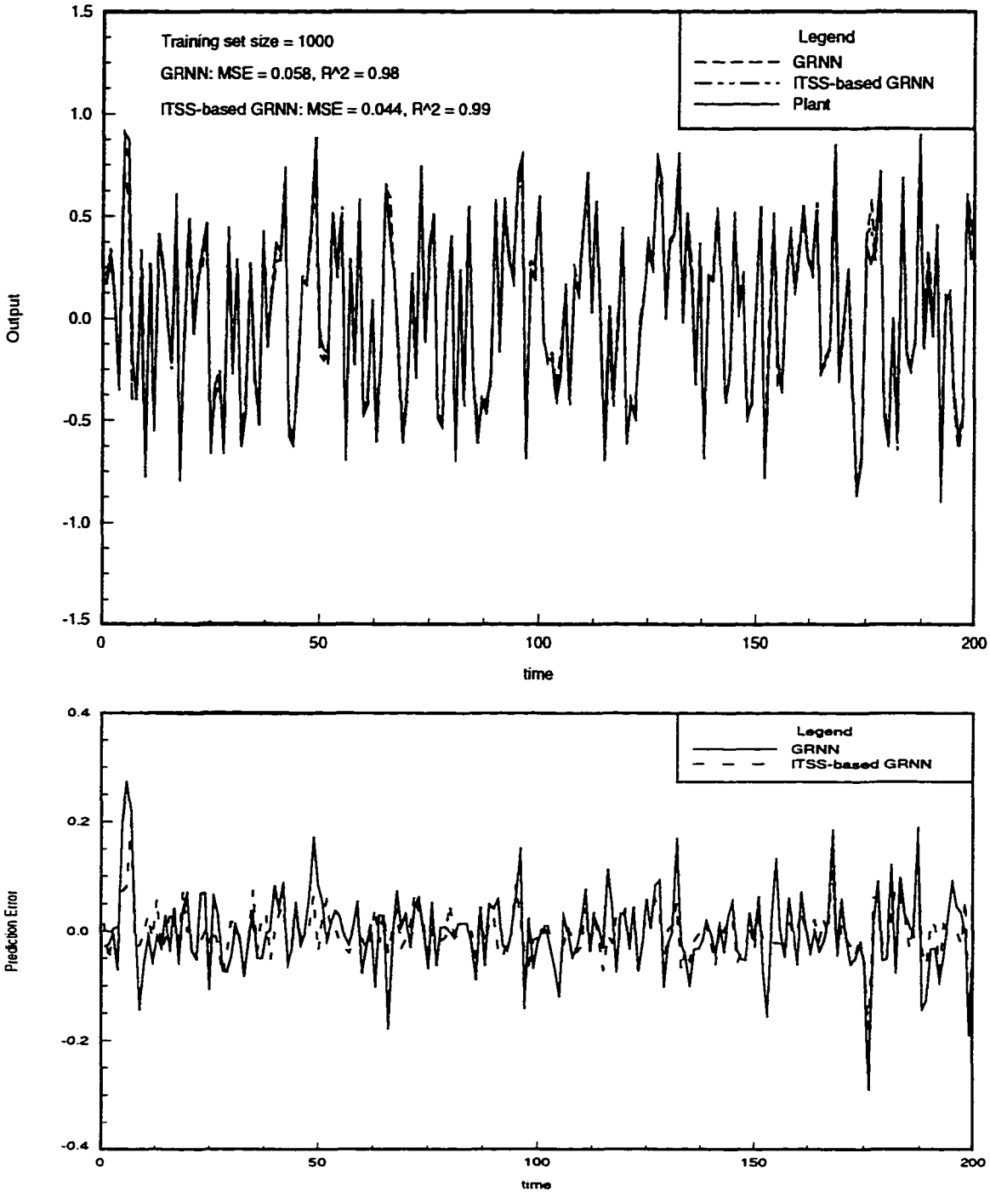


Figure 5 Example 2. Comparison of GRNN and ITSS-based GRNN for a sample size of 1000

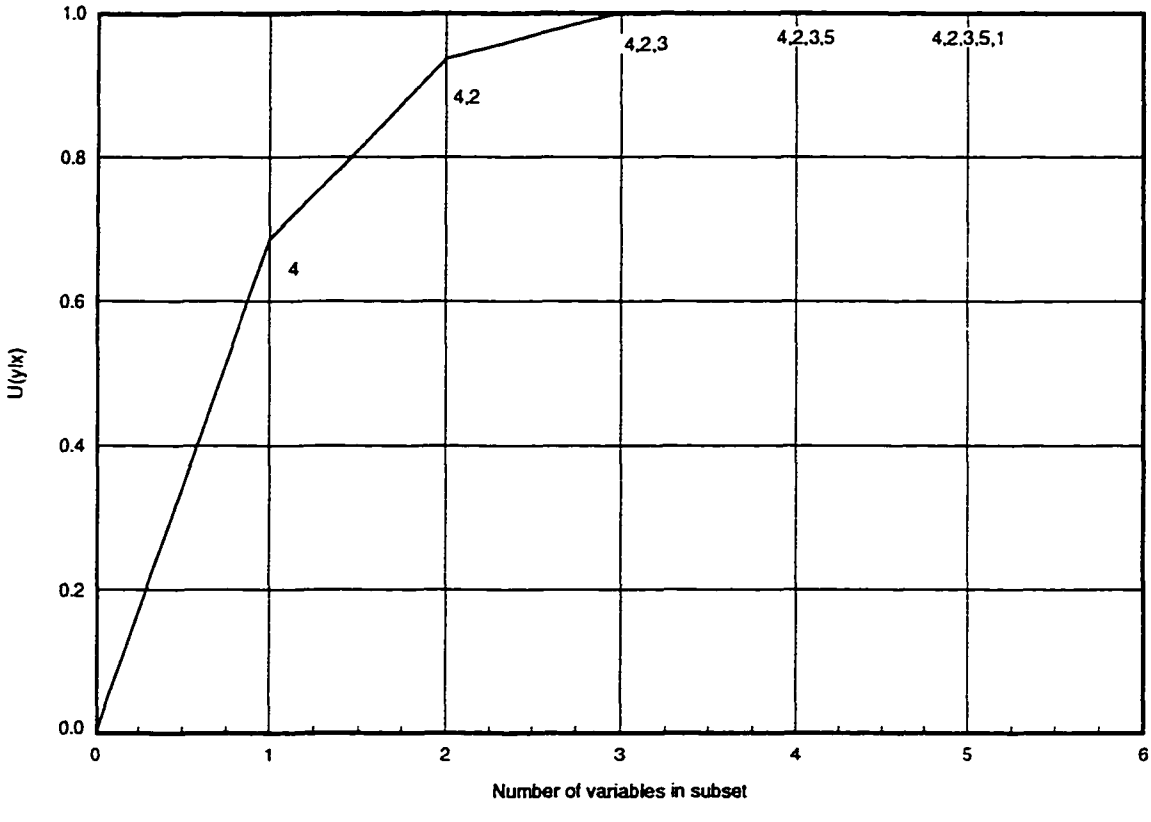


Figure 6 Example 2 CITC for a sample size of 100. The subset $\{x_4, x_2\}$ has a $U(y|\mathbf{x})$ of 0.93 and is a candidate subset for the GRNN model

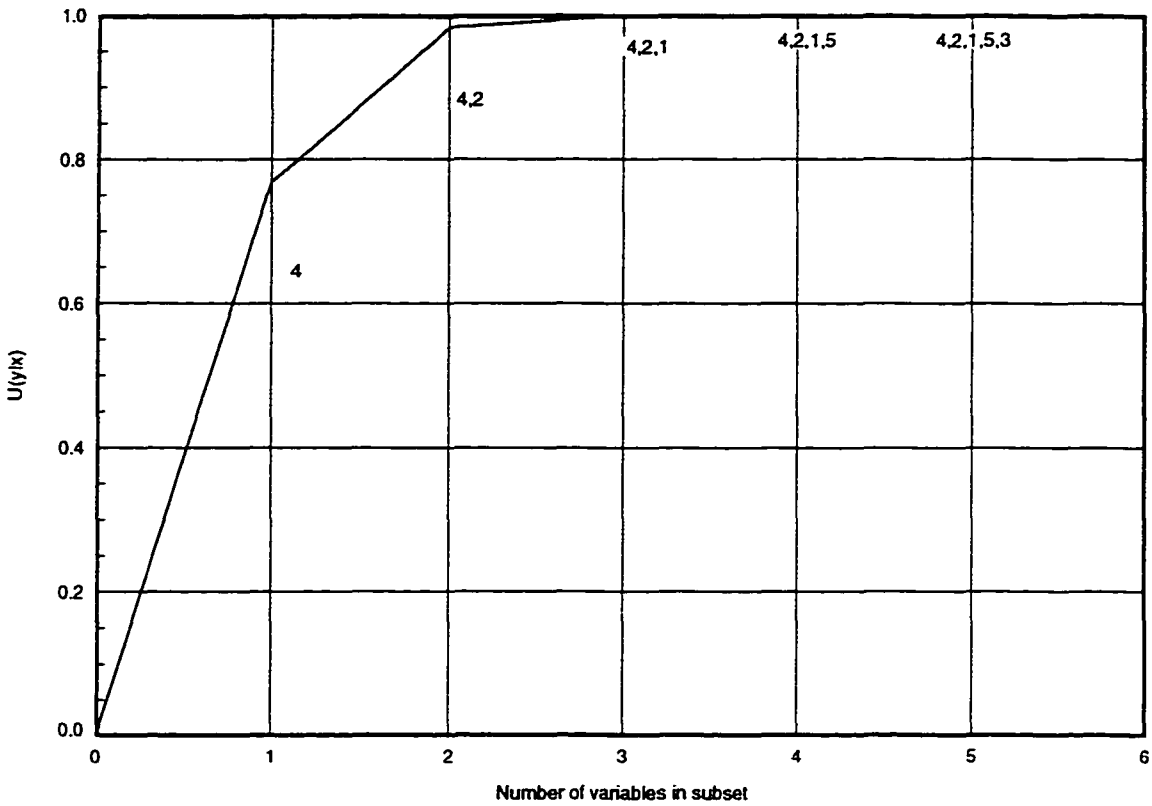


Figure 7 Example 2 CITC for a sample size of 50. The subset $\{x_4, x_2\}$ has a $U(y|\mathbf{x})$ of 0.98 and is a candidate subset for the GRNN model

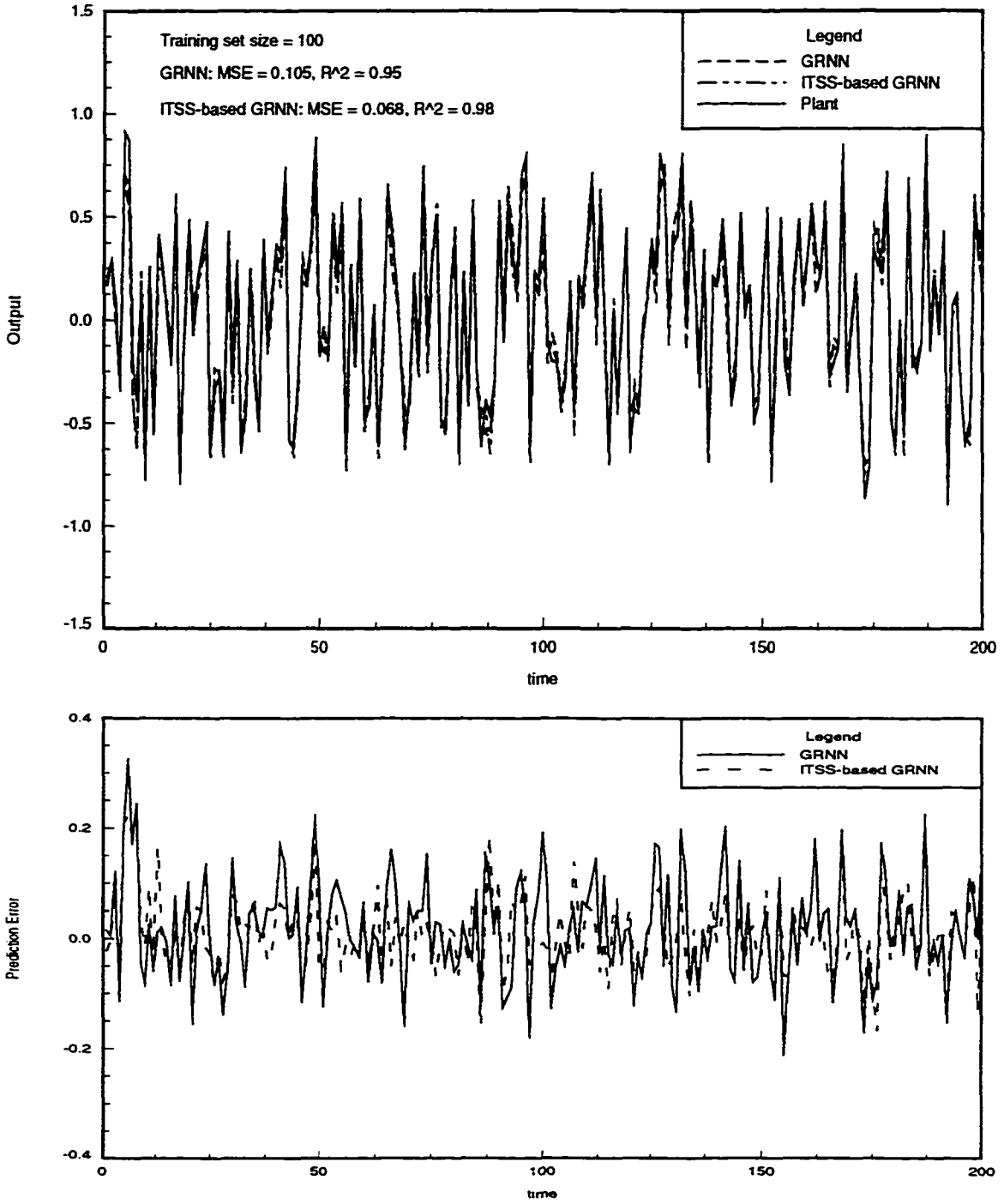


Figure 8 Example 2. Comparison of GRNN and ITSS-based GRNN for a sample size of 100

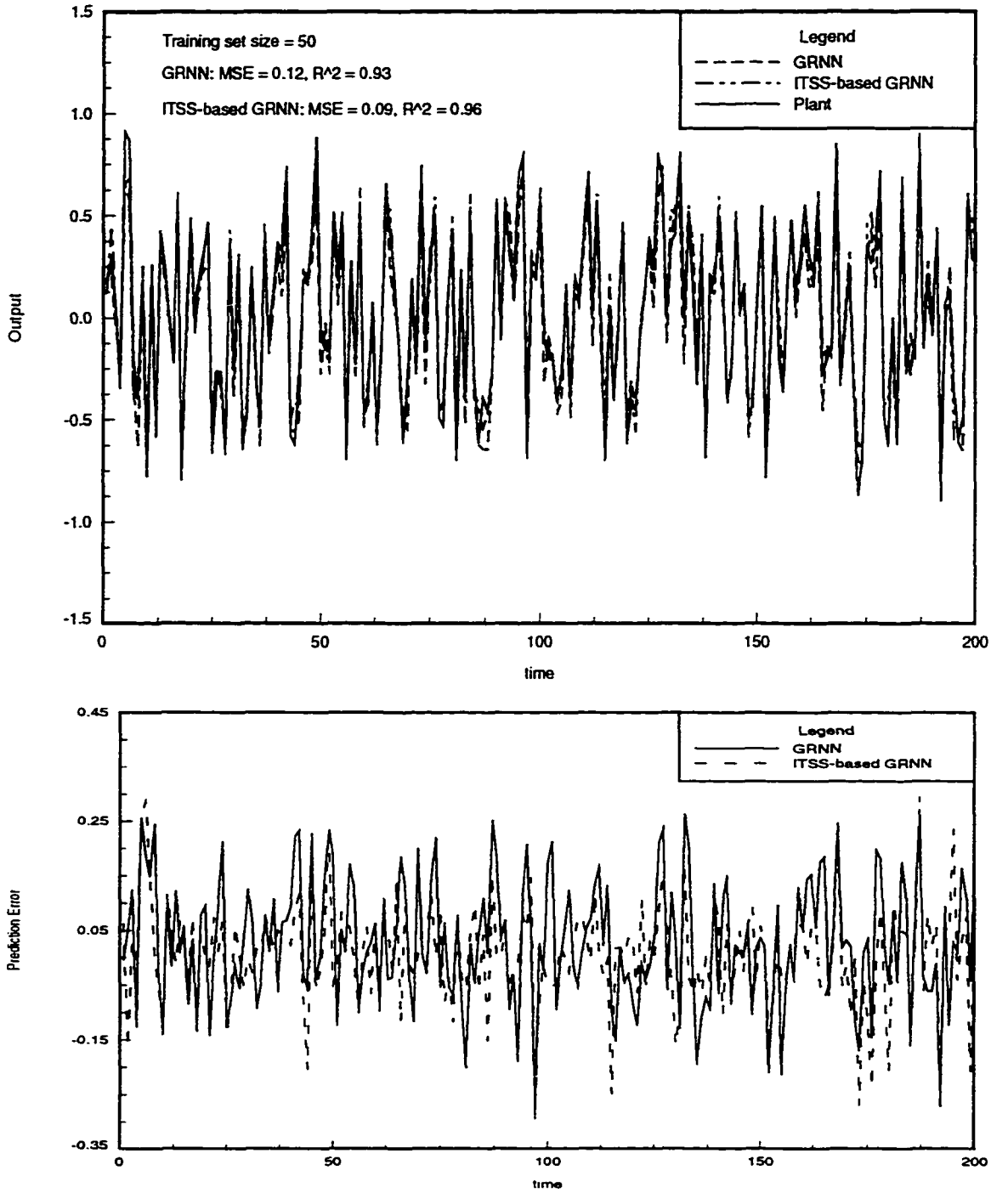


Figure 9 Example 2. Comparison of GRNN and ITSS-based GRNN for a sample size of 50

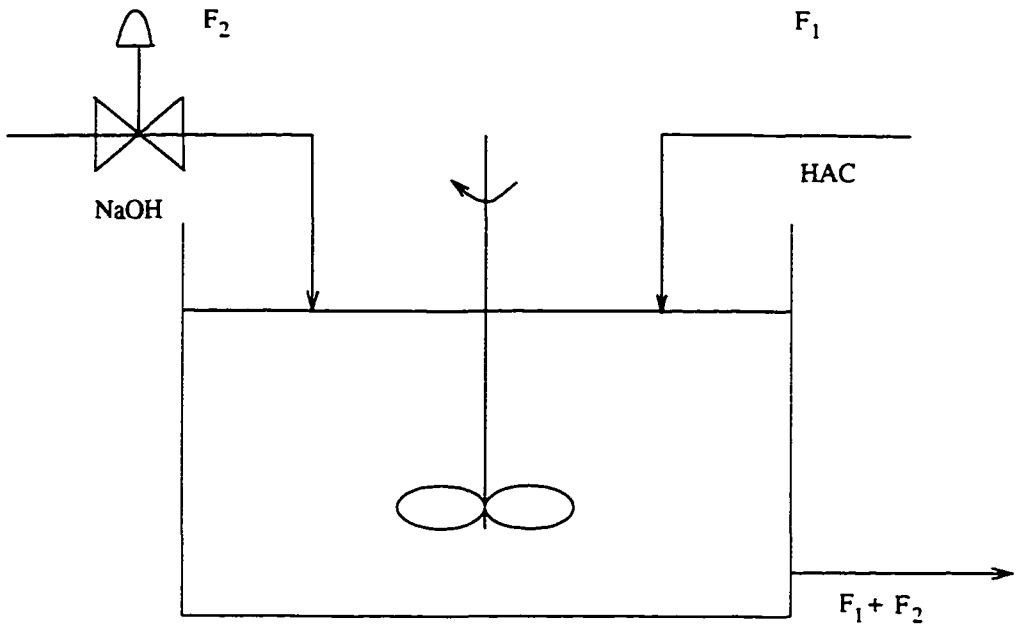


Figure 10 Schematic of the pH CSTR

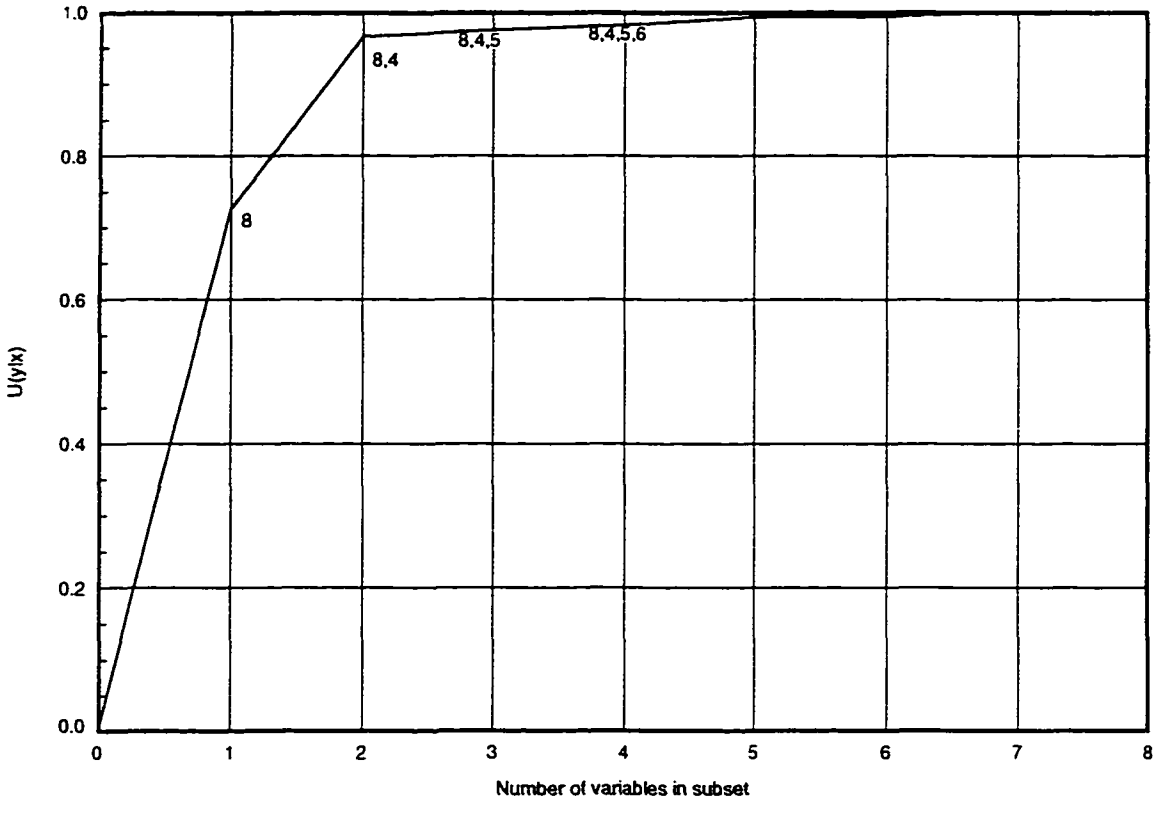


Figure 11 The Cumulative Information Theoretic Curve (CITC) for the pH CSTR showing that pH(t) and q(t) are sufficient to predict pH(t+1)

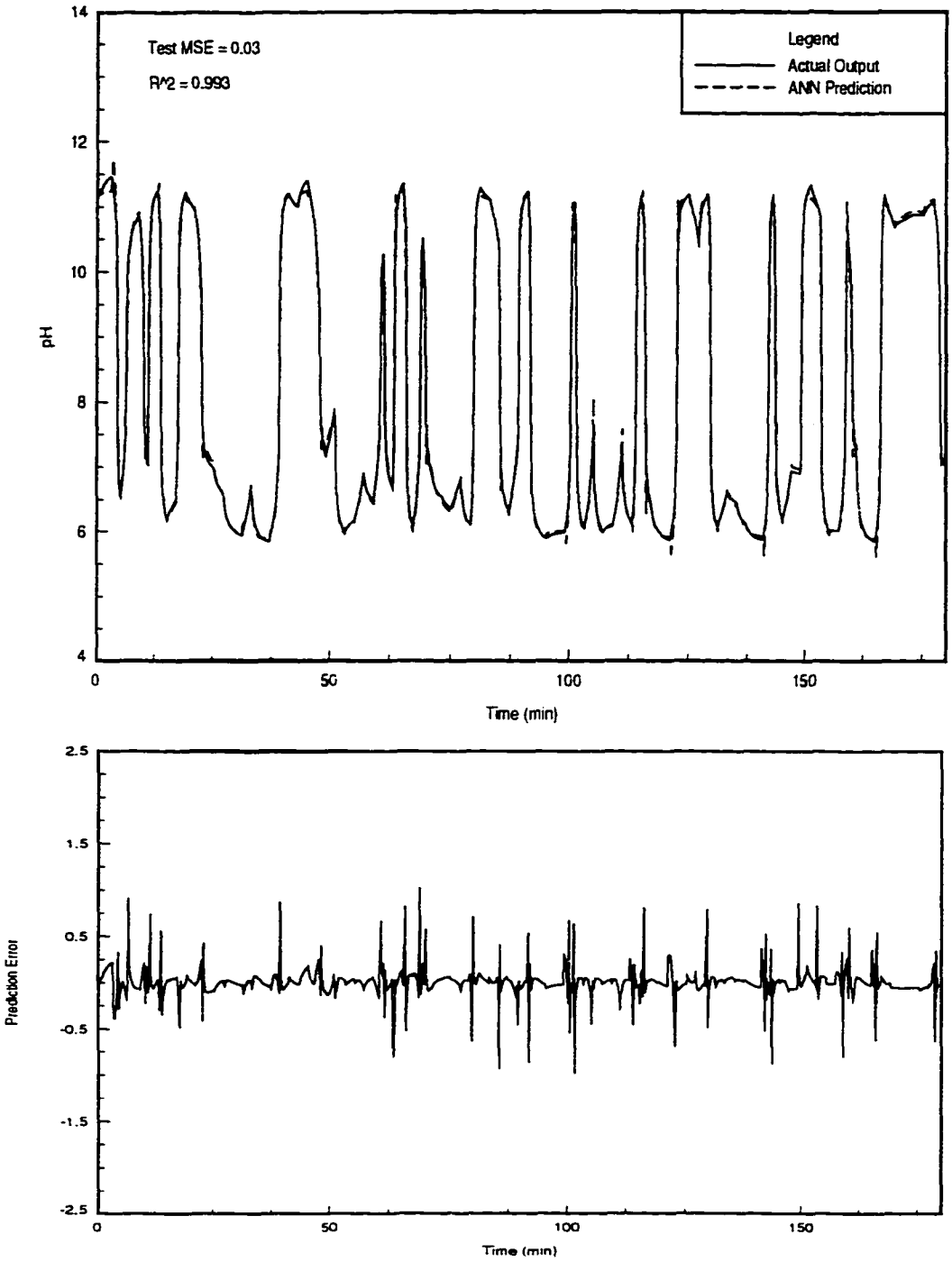


Figure 12 Prediction of the 2x10x1 neural network model for the pH tank system

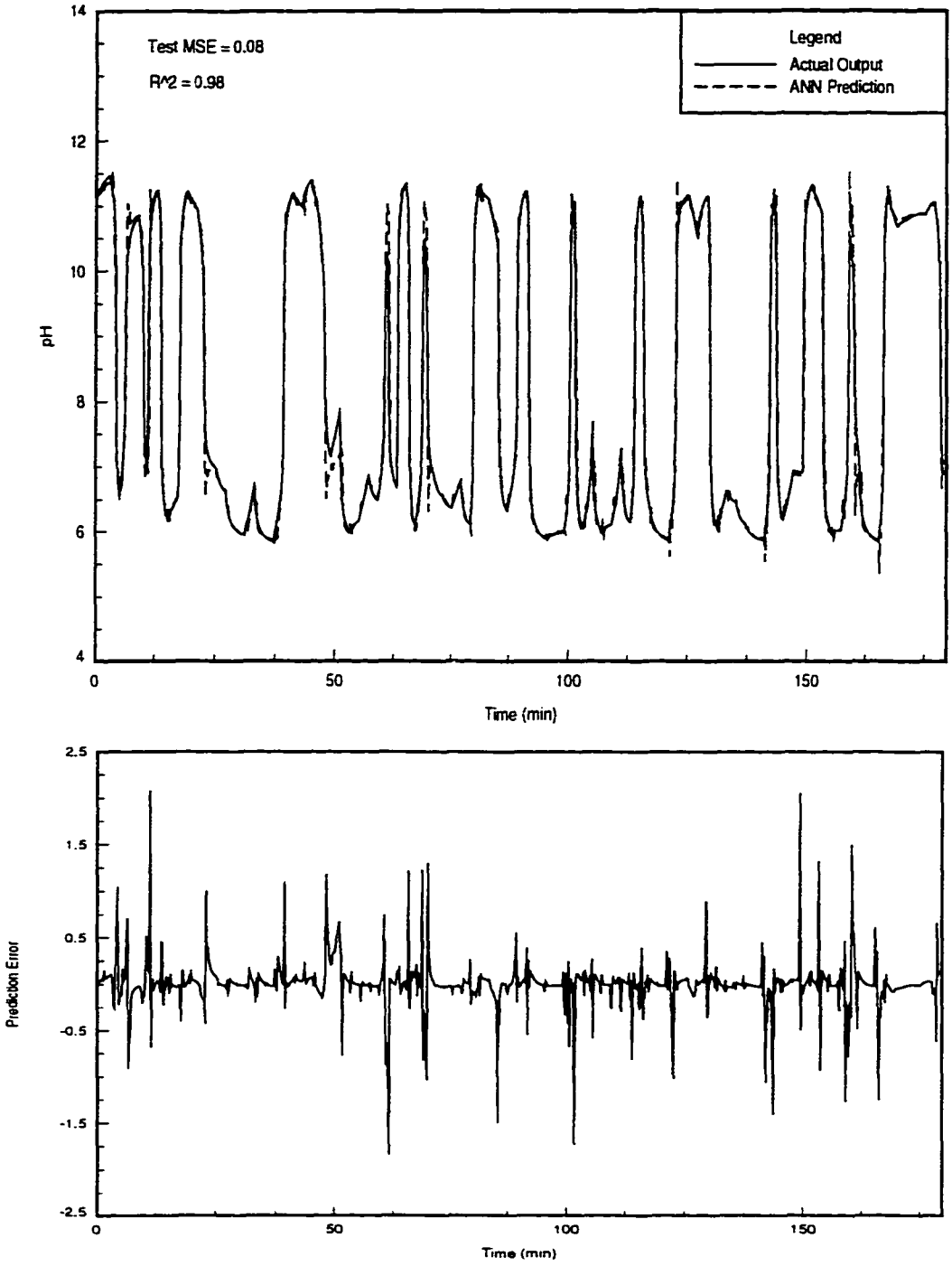


Figure 13 Prediction of the 8x10x1 neural network model for the pH tank system

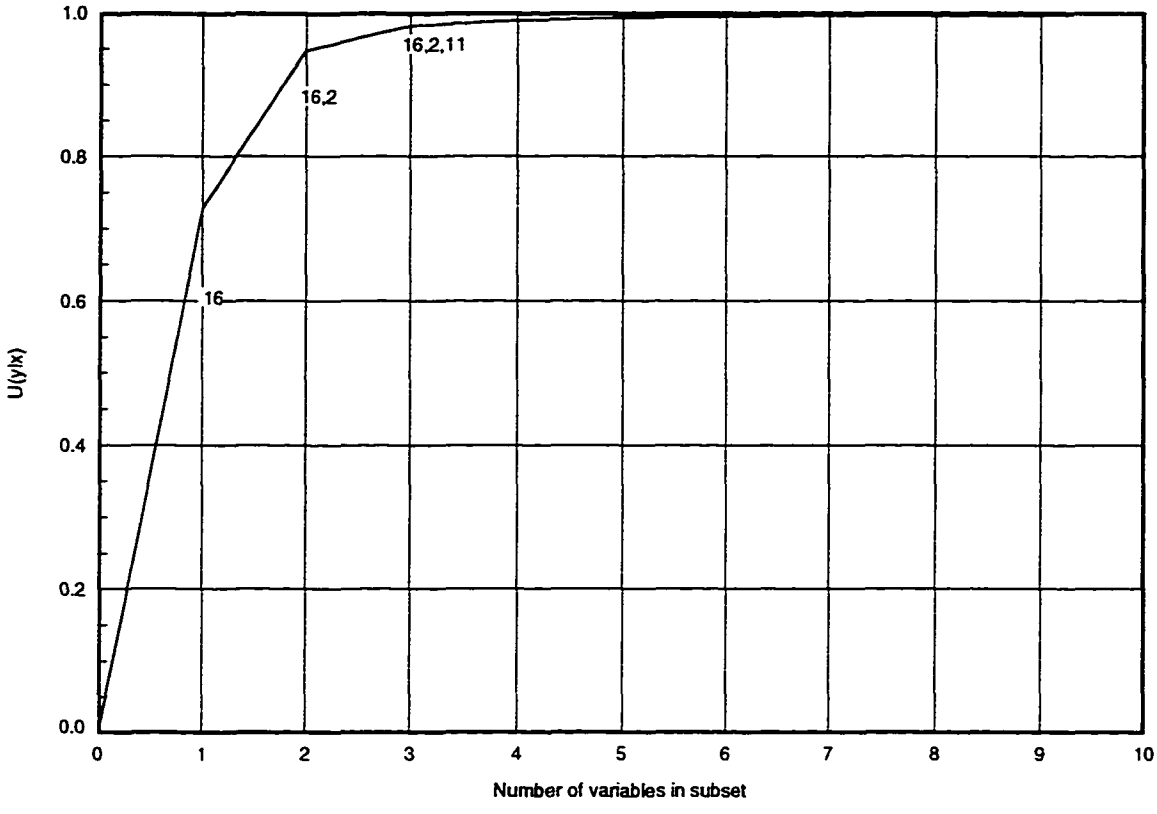


Figure 14 The Cumulative Information Theoretic Curve (CITC) for the pH CSTR showing that pH(t) and q(t-6) are sufficient to predict pH(t+1)

AN INFORMATION THEORETIC APPROACH FOR COMBINING NEURAL NETWORK PROCESS MODELS

A paper submitted to the Neural Networks journal

Dasaratha Sridhar, Eric B. Bartlett and Richard C. Seagrave

Abstract

Typically neural network modelers in chemical engineering focus on identifying and using a single hopefully optimal neural network model. Using a single optimal model implicitly assumes that one neural network model can extract all the information available in a given data set and that the other candidate models are redundant. In general, there is no assurance that any individual model has extracted all relevant information from the data set. Recently, Wolpert (1992) proposed the idea of stacked generalization to combine multiple models. Sridhar *et al.* (1996) implemented stacked generalization for neural networks models by integrating multiple neural networks into an architecture known as stacked neural networks (SNNs). Stacked neural networks consist of a combination of the candidate neural networks and were shown to provide improved modeling of chemical processes. However, in Sridhar's work stacked neural networks were limited to using a linear combination of ANNs. While a linear combination is simple and easy to use, it can utilize only those model outputs that have a high linear correlation to the output. Models that are useful in a nonlinear sense are wasted if a linear combination

is used. In this work we propose an information theoretic stacking (ITS) algorithm for combining neural network models. The ITS algorithm identifies and combines useful models regardless of the nature of their relationship to the actual output The power of the ITS algorithm is demonstrated through three examples including application to a dynamic process modeling problem. Results obtained demonstrate that the SNNs developed using the ITS algorithm can achieve highly improved performance as compared to selecting and using a single hopefully optimal network or using SNNs based on a linear combination of neural networks.

Introduction

Artificial Neural Networks (ANNs) have been used in chemical engineering for dynamic modeling (Bhat and McAvoy, 1990; Pollard *et al.*, 1992), fault diagnosis (Hoskins and Himmelblau, 1988; Venkatasubramaniam and Chan, 1989) and steady state process modeling (Joseph *et al.*, 1992). The approach to ANN based process modeling has been to consider a number of candidate models, and to select one model which is expected to best predict the process outputs, given the process inputs. The selected model is the one that is expected to have least prediction error in the future. The expected prediction error of the candidate models is usually computed on data reserved for testing the model. Using a single optimal model implicitly assumes that one ANN model can extract all the information available in the data set and that the other candidate models are redundant. In general, there is no assurance that any individual model has extracted all relevant information from the data set. Recently Sridhar *et al.* (1996) showed that better predictive models can be obtained by stacked modeling. Stacked modeling refers to the idea of stacking or integrating the candidate networks into an architecture known as Stacked Neural Networks (SNNs). Stacked neural networks were inspired by the stacked generalization approach proposed by Wolpert (1992). However, in our previous work,

SNN development was limited to a linear combination of the candidate networks. While a linear combination is simple and easy to use, a linear combiner can make use of only those models which have high linear correlation to the output variable. It is however possible that models whose outputs have low linear correlation could be important in a nonlinear sense. Such models cannot be effectively integrated by a linear combiner. It is therefore important to develop algorithms which can combine useful ANN models regardless of the nature of their relationship to the actual output. The focus of this paper is to describe and demonstrate the application of a new technique called the information theoretic stacking (ITS) algorithm that determines a combination of neural networks to obtain improved process models. The ITS algorithm is a general technique for combining candidate ANNs that does not require the form of the combination to be specified. Instead the ITS uses the data itself to come up with the appropriate combining rule.

ITS uses a two stage approach to combine the candidate ANNs. In the first stage, ITS selects an informative subset of models to be used in the combination. The selection of an informative subset of models is based on applying information theoretic analysis (Watanabe, 1969; Shannon and Weaver, 1949). Information theory allows interdependency analysis (Press *et al.*, 1986) without making any assumptions about the nature of the relationship between variables. As a result ITS can identify useful subsets of the candidate models without making any assumption about how they relate to the actual output. Selection of a subset of the candidate models has several advantages. The presence of a large number of models increases input dimensionality for the combining algorithm and makes accurate estimation difficult with limited data. As a result the full benefits of combining the candidate networks may not be realized. By using an informative subset ITS avoids the problems associated with high input dimensionality. Using only a subset of the candidate models has the additional advantage of keeping the structure of the SNN as simple as possible and it also decreases the time required to develop the nonparametric combining rule.

In the second stage a nonparametric modeling technique is used to develop a combination of the models that were selected in the first stage. The advantage of using a nonparametric combiner is that we do not restrict the combination to be of a specific form. Instead we allow models to be combined in a manner that is determined from the data. The nonparametric combiner used in this work is the general regression neural network (GRNN) (Specht, 1990). The GRNN model is a nonparametric technique that can model any process regardless of the nature of the relationship between the outputs and inputs. Moreover, the GRNN is a memory based system and is therefore easy to develop.

The remainder of this paper is organized in the following manner. We first discuss the theory behind the ITS technique. Second, the methodology for implementation of the ITS algorithm is provided. The ITS algorithm is first illustrated on a classification problem. The second example demonstrates application of ITS to a function mapping problem. Following this, the feasibility of the ITS algorithm is explored through application to a dynamic process modeling problem. Results obtained demonstrate that SNNs using the ITS algorithm can achieve highly improved performance as compared to selecting and using a single hopefully optimal network or using a linear combination of the candidate networks.

Theory

ANNs have been used for process modeling due to their ability to approximate arbitrarily complex functions (Hornik, 1989; Cybenko, 1989). Neural networks will not be reviewed in this paper. For details of artificial neural networks see Hecht-Nielsen (1990), Lippman (1987) and for an introduction to neural nets in chemical engineering, see Hoskins and Himmelblau (1989) or Venkatasubramanian and Chan (1989). A neural network model trained to learn a given mapping can be considered as a system that

reduces uncertainty about the output by using information in the input vector. Ideally we would like this uncertainty to be zero. In the real world this may not be possible. There are two reasons:

- The process inputs contain insufficient information about the output
- The neural network model is suboptimal because it does not completely utilize the information in the inputs. The network can waste some of the information in the inputs because of insufficient training or approximation failures.

In the traditional approach to modeling, several neural network models are built and the one that is considered to be the best is used for the application. This approach is depicted in Figure 1(a). The network that is best is determined by splitting the data set into two parts: the training and the test data set. The training data set is used for developing the various neural models. All of the models are then evaluated on the test data set. The model with the least error over the test set is then selected and used. This approach however may not be satisfactory. As shown by Sridhar *et al.* (1996), stacked modeling can provide improved modeling. Stacked modeling refers to the idea of developing and using a combination of the candidate networks rather than simply selecting the best network. The schematic for the stacked modeling approach is shown in Figure 1(b). In this case, the test data set is used to develop a combination of the candidate networks. The rationale is that one model may not have extracted all the information that is relevant for predicting the output. By using the information that other models have to offer, stacked modeling attempts to provide better prediction of the output.

Shannon's information theory provides a suitable formalism for quantifying the above concepts. Consider a process output or a quality attribute $\mathbf{y}$ that we need to predict using the vector of process inputs $\mathbf{x}$. Information theory allows us to quantify the information content of any vector. It also allows measures of association to be developed between

the two vectors x and y. For simplicity assume that y is a discrete vector that can take on R discrete values $y_1, y_2, ...y_R$. Similarly assume that x is a discrete vector that can take on S discrete values $x_1, x_2, ...x_S$. For real world problems, x and y are often continuous rather than discrete. However we can divide the variable domain into a finite number of regions within which the variable is assumed to be constant. The following discussion is then valid for continuous variables also. Define p_i as the probability that y can take on the value y_i.

$$p_i = p(y = y_i) \tag{1}$$

The probability p_i can be estimated using the frequency of occurrence of the vector y_i

$$p_i = \frac{N_i}{N} \tag{2}$$

where N_i is the number of occurrences of the vector y_i in the data set and N is the total number of patterns in the data set. The entropy or information presented by the variable y can then be defined as

$$H(y) = -\sum_{i=1}^{R} p_i ln(p_i) \tag{3}$$

If y can take only one value the information contained in y is 0. This would imply that the process output is always constant and predicting the value of the output is a trivial task. The joint information content H(x,y) of two vectors x and y can be defined analogous to Equation 3.

$$H(x, y) = -\sum_{i,j} p_{ij} ln(p_{ij}) \tag{4}$$

where p_{ij} is the probability that x will take on the value x_i and y will take on the value y_j. p_{ij} is given by the equation

$$p_{ij} = p(x = x_i, y = y_j) = \frac{N_{ij}}{N} \tag{5}$$

where N_{ij} is the number of joint occurrences of the vectors x_i and y_j in the data set. The mutual information (MI) or the entropy of y given x, H(y|x), is a measure of the

information in the vector y when x is known. H(y|x) is given by:

$$H(\mathbf{y}|\mathbf{x}) = -\sum_{i,j} p_{ij} ln \frac{p_{ij}}{p_i} \tag{6}$$

It can be shown that

$$H(\mathbf{y}|\mathbf{x}) = H(\mathbf{x},\mathbf{y}) - H(\mathbf{x}) \tag{7}$$

An asymmetric dependency coefficient (ADC) that measures the dependency of y on x, U(y|x) can then be defined as

$$U(\mathbf{y}|\mathbf{x}) = \frac{H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x})}{H(\mathbf{y})} \tag{8}$$

The above equation can also be written as:

$$U(\mathbf{y}|\mathbf{x}) = \frac{H(\mathbf{y}) + H(\mathbf{x}) - H(\mathbf{x},y)}{H(\mathbf{y})} \tag{9}$$

Equation 9 gives a very useful measure of association between the vectors y and x. U(y|x) does not depend on the nature of the functional relationship between y and x. U(y|x) measures the extent to which knowledge of x provides information about y. If U(y|x) is 0 it means that x does not contain any useful information about y, and it is not possible to predict y to any extent using x. A value of 1 implies that knowledge of x completely determines y, and it is possible to predict y exactly using x. The above concepts will now be used to explain the logic behind the ITS technique.

Assume that a data set $D((y_t, \mathbf{x}_t) : 1 \leq t \leq T)$ has been collected on some process of interest. Here, y denotes the output variable, x denotes the input vector, T is the number of training patterns and t is the pattern index. For the present discussion assume that the data set D is split into two parts: D_1 used for training the candidate models and D_2 for testing the trained networks. However the analysis is general and holds for other methods of partitioning the data set as in k-fold cross-validation (Weiss and Kukilowski, 1991) or bootstrapping (Efron, 1993). Assume that D_1 contains N_{TR} patterns and D_2 contains N_{TS} patterns. It is assumed that M neural networks (N_1, N_2, ...N_M) are the candidate

models. The candidate ANNs will be referred to as the level 0 models following Wolpert (1992). The data used to train the candidate networks is known as the level 0 data set and will be denoted by D_{L0}. In the present discussion D_1 is the level 0 data set. In general, a neural network model can be thought of as a nonlinear transformation of the input vector. We hope that the transformation retains all of the relevant information about the output and filters out the redundant information in the input vector. Using different neural network models is equivalent to taking several nonlinear transformations of the input vector. In other words, we try several transformations, $y_1 = f_1(x)$, $y_2 = f_2(x)$,... $y_M = f_M(x)$. The parameters of each of the networks is estimated using D_1 and then the output of all the ANNs is computed on all of the patterns in D_2 in order to evaluate each network. Denote the prediction of the j^{th} network for pattern n in data set D_2 as yp_{nj}. For pattern n, the output of the candidate ANNs is collected in a M dimensional vector $\mathbf{yp}_n = \{yp_{nj} : 1 \leq j \leq M\}$. The actual output y_n and the network outputs $\mathbf{yp}_n$ constitute the output and input respectively in a new space which is defined as the level 1 space. Repeating this procedure for all N_{TS} patterns in D_2 produces the level 1 data set $D_{L1}(y_n, \mathbf{yp}_n)$. Data set D_{L1} contains the true output and the predictions of the M models, for all the N_{TS} patterns. Assuming that the modeler is not interested in trying any further models we now have a data set D_{L1} which needs to be used to develop a predictive model. At this point we know that

$$U(y|\mathbf{yp}) \leq U(y|\mathbf{x}) \tag{10}$$

Equation 10 implies that at best we can hope that all of the networks collectively retain what is important about the process inputs. In general we cannot assure that all information in $\mathbf{x}$ about the output has been retained in $\mathbf{yp}$. It is also difficult to determine how the information in $\mathbf{x}$ is distributed in $\mathbf{yp}$. In any case there is nothing that can be done about the information that has been lost. A logical approach then is to utilize all of $\mathbf{yp}$ to predict the process output. This is the central idea behind stacked modeling.

Using information in all of the candidate ANNs requires a combining rule that can integrate the information in the ANNs to predict the output. Therefore the objective of stacked modeling is to determine

$$y_s = g(y_1, y_2, ...y_M) \tag{11}$$

such that

$$U(\mathbf{y}|\mathbf{y}_s) - U(\mathbf{y}|y_1, y_2, ...y_M) < \epsilon \tag{12}$$

where y_s is the prediction of the stacked neural network and g is known as the level 1 model. In general the form of the level 1 model g is unknown. The traditional approach to neural network modeling simply sets

$$g(y_1, y_2, ...y_M) = y_k \tag{13}$$

such that y_k is closest to y in a euclidean sense. This is clearly not appropriate since

$$U(y|y_i) \leq U(y|\mathbf{yp}) \tag{14}$$

The traditional approach is equivalent to assuming that there exists a transformation $y_i = f_i(x)$ such that

$$U(y|f_i(x)) = U(y|f_1(x), f_2(x), ...f_M(x)) \tag{15}$$

Practically it is difficult to ensure that the assumption in Equation 15 will hold. As discussed in our previous work there are several reasons why the above assumption may be violated. These reasons are listed below.

- Use of suboptimal network architectures or failure of any automatic network construction method used to determine the optimal network.
- Sensitivity of the neural network to the initial starting conditions for network training.

- Use of inappropriate activation functions and learning algorithms.
- Different convergence criteria used for network training can lead to very different solutions for a given network architecture.

The linear combination proposed in our earlier work is equivalent to assuming that

$$g(y_1, y_2, ...y_M) = a * y_1 + b * y_2 + ...m * y_M \tag{16}$$

The above combining rule can integrate only those models which have high linear correlation to the output. Models that are important in a nonlinear sense are ignored. It is therefore not desirable to make any assumptions about the combining rule and it is preferable to let the data determine the appropriate combination.

Although it is desirable that the function g should be an arbitrary nonlinear function that needs to be developed from the level 1 data, problems can arise in practice. We usually deal with a finite data set and the test set is a fraction of the entire data set. Hence the level 1 data set that is used for developing g could be relatively small. Moreover if many models are used then the dimensionality of the level 1 input space can become high. High input dimensionality of the level 1 input space increases the complexity of the learning phase for the combining algorithm, and leads to longer training times. Presence of too many inputs would also require a large number of parameters to be estimated and can cause poor network generalization by the level 1 model. It is also desirable to keep the SNN structure as simple as possible by avoiding the use of models that are redundant.

This problem can be resolved if we select a subset of the candidate ANNs using information theoretic analysis. The idea is to find a vector $\mathbf{yp}_s$ which is a subset of the level 1 input space vector $\mathbf{yp}$. The subset $\mathbf{yp}_s$ is determined such that it contains almost all of the information in $\mathbf{yp}$ that is important for predicting the output vector $\mathbf{y}$. Denote a candidate subset of the the vector $\mathbf{yp}$ as $\mathbf{yp}_{sk}$, where k indicates the dimensionality of

the subset input vector. Define the asymmetric dependency between the subset input vector and the output as $U(\mathbf{y}|\mathbf{yp}_{sk})$. Then the objective is to determine $\mathbf{yp}_{sk}$ such that:

$$U(\mathbf{y}|\mathbf{yp}_{sk}) - U(\mathbf{y}|\mathbf{yp}) < \epsilon \tag{17}$$

where ϵ is deemed to be an acceptably small loss of information in the input space with respect to predicting the output. Equation 17 can be used to evaluate the loss of information by using any subset instead of the entire input vector. Only the selected subset of models are then used to develop the level 1 model. This reasoning forms the basis of the information theoretic stacking (ITS) algorithm for stacking neural networks

Methods

In this section we develop stacked neural networks using the ITS algorithm. The architecture of an SNN using a combiner determined by the ITS algorithm is shown in Figure 2. The SNN model takes a given input x and passes it through all of the neural network models. The outputs of these models are then combined using the ITS-based combiner to produce the final prediction of the output. A general approach for developing the Stacked Neural Networks (SNNs) using the ITS algorithm is given below:

1. Train the M level 0 networks using D_1. Denote the j^{th} network trained on D_1 as $N_j(D_1)$, and denote the set of these level 0 networks as $N(D_1) = \{N_j(D_1) : 1 \leq j \leq M\}$. The $N(D_1)$ should be saved for use in the SNN model.

2. Recall the M networks on the data set D_2. Denote the prediction of the j^{th} network for pattern n in data set D_2 as yp_{nj}. For pattern n, collect the output of the candidate models in a M dimensional vector $\mathbf{yp}_n = \{yp_{nj} : 1 \leq j \leq M\}$. The actual output y_n and the network outputs $\mathbf{yp}_n$ constitute the output and input respectively for the n^{th} pattern in data set D_{L1}. Repeat the above procedure for all N_{TS} patterns to produce the level 1 data set $D_{L1}(y_n, \mathbf{yp}_n)$.

3. Identify a subset of the candidate models which contain most of the information relevant to predicting the output.

   (a) Calculate U(y|**yp**)

   (b) Generate a candidate subset $\mathbf{yp}_{sk}$ with $1 \leq k \leq M$ where k is the dimensionality of the generated subset

   (c) Compute U(y |$\mathbf{yp}_{sk}$)

   (d) Check if the condition in Equation 17 is satisfied to decide whether selected subset is satisfactory.

   (e) If subset is satisfactory stop else go to step b

4. Use a nonparametric model to learn the relationship between the actual output y and the selected subset $\mathbf{yp}_{sk}$.

5. For prediction on a novel input pattern, the input is fed through all of the candidate models. The outputs of the level 0 neural networks are combined using the nonlinear combiner to produce the final prediction.

The proposed method is shown in Figure 3. To implement the methodology there are two issues that need to be addressed. First we need a method to generate the candidate subsets. Second we need a technique to combine the selected subset of models. These issues are discussed below:

1. Subset selection: There are several well known search algorithms for generating candidate subsets. An obvious methods is to use an exhaustive search over all the input variables; this however would be computationally infeasible except for small input vectors. In the present work we use the simple approach of forward selection. The forward selection procedure generates a new subset by adding one input variable at a time to the current subset. At each stage. the variable to be added is

selected so that the new enlarged input vector maximizes $U(\mathbf{y}|\mathbf{yp}_{sk})$. The variable addition is terminated when the condition in Equation 17 is satisfied. Other techniques like backward elimination, branch and bound algorithms (Fukunaga, 1990) can also be used but will not be considered in this article.

2. Selection of the nonlinear combining method: An important requirement of the nonlinear combiner is that it should be able to determine arbitrary combinations that is determined from the data. In this article we will use the general regression neural network (GRNN) model (Specht, 1990). The GRNN is a memory-based system and is very quick to develop. Furthermore GRNN is an universal approximator and therefore arbitrary nonlinear combinations can be developed. For details of the GRNN model development the reader is referred to Specht (1990).

The generalized XOR problem

The purpose of this simple example is to demonstrate the effectiveness of the ITS-based SNN in a concise manner. The problem considered is the Generalized XOR (Hansen and Solomon, 1990) shown in Figure 4. The problem requires a classifier to output a 1 for any point from the first and third quadrant and a 0 for any point from the second and the fourth quadrant. The training data set D_1 consisted of 1000 data points with equal number of points from each quadrant. Another data set D_2 with 1000 patterns was generated to evaluate the candidate networks and to develop the SNN. Two backpropagation networks with 1 and 2 hidden nodes respectively were considered as the candidate networks. The networks used the hyperbolic tangent activation function and were trained using the scaled conjugate algorithm (Moller, 1993). We used a high value of gain so that the activation function of the networks is very similar to a step function. A large data set D_3 with 10,000 patterns was also generated to estimate the classification accuracy of the individual networks and the SNN.

The results of training the networks on D_1 and testing on D_2 are shown in Table 1. The data in this table shows the output of the two networks and the actual output and is the level 1 data for this problem. Since the output of each network can take on two possible values and the actual output can take on two values, there can be 2^3 possible patterns in the level 1 data set. The frequency of occurrence of each of these 8 patterns is listed in Table 1. The results indicate that performance of both networks is quite poor. When N_1 outputs a 0 there is almost an equal probability of the true output being 0 or 1. When N_1 outputs a 1 there is almost an equal probability of the true output being a 0 or a 1. Therefore the output of N_1 does not seem to have any information about the true output. Similar reasoning indicates that performance of the N_2 will also be poor. N_1 however is marginally better than N_2. Performance of N_1 and N_2 was poor due to insufficient nodes in the hidden layer, and inappropriate choice of initial weights and network parameters. These nets were deliberately trained poorly so that we could illustrate the improvement due to ITS-based stacking. If we wanted to use a single model, N_1 would be selected as it appears to be the better model. Evaluating the performance of N_1 on D_3 shows that the classification accuracy is only 50.6%. The SNN based on the ITS algorithm on the other hand integrates the two networks and achieves a classification accuracy of 97.9%. The ITS-based SNN has integrated two networks which show poor performance to develop a classifier that is highly accurate.

Analysis of the results in Table 1 explains the large performance improvement obtained using the SNN. The SNN looks at both networks jointly instead of examining each network separately. Notice that when both networks output a 0 the probability of the correct answer being 1 is 0.94. When both networks output a 1, the probability of the correct answer being 1 is 0.996. If any of the networks outputs a 0 the probability of the correct answer being 0 is 0.99. It appears that the two networks jointly contain significantly more information about the actual output as compared to each network used by itself. A logical approach would then be to look at the outputs of both networks

while classifying a pattern . If both networks output a 0 or both output a 1 then we would predict the output to be 1. If one of the networks guesses a 0 then we would predict the output to be 0. It is apparent that a nonlinear combiner is required to implement this combining rule. ITS essentially follows the above reasoning to arrive at a accurate SNN classifier. ITS first computes the ADC for each individual network. For both N_1 and N_2, the ADC is close to 0. ITS then determines that U(y | y_1,y_2) is 0.86. At this point the algorithm decides that both networks must be jointly used to achieve good classification. The GRNN model then learns the required combining rule using the level 1 data. The above approach results in a SNN that has very high classification accuracy. In contrast, a linear combining rule would fail to effectively make use of the two networks since it cannot encode the desired combining rule.

A graphical illustration of these results is provided in Figure 5. N_1 essentially draws a line that separates Q1 and Q4 from Q2 and Q3. N2 draws a line that separates Q4 and Q3 from Q1 and Q2. Obviously neither of the two networks has drawn the correct classification boundary. However if we used both the networks it is equivalent to drawing the two lines that represents the correct classification boundary. The two networks jointly contain most of the information required for the classification. A nonlinear combiner is however required to effectively utilize this information. This is achieved by the ITS algorithm leading to the large improvement. It should be pointed out that a single network with larger number of nodes can also be developed to solve this problem. In such a situation the ITS algorithm would simply set the stacked model to be the same as the single network and ignore other models. However in complex problems, it could be difficult to identify a single optimal network, and SNNs built using the ITS algorithm can be a safer strategy as compared to using a single model or even a linear combination of the candidate networks.

A function mapping example

In this example we will demonstrate the effectiveness of the ITS algorithm for a function mapping problem. For this example, the objective is to estimate the 6-dimensional additive function,

$$y = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + 0x_6 \tag{18}$$

The x_i were generated from a uniform distribution in the six-dimensional hypercube. This function has been used as a benchmark for MARS (Friedman, 1991). The training data set S_1 consisted of 100 samples, and constitutes the level 0 data. The added noise was Gaussian with a signal to noise ratio (SNR) of 4.0. The signal to noise ratio is defined as

$$SNR = \sigma_y / \sigma_N \tag{19}$$

where, σ_y is the standard deviation of y and σ_N is the standard deviation of the noise. In addition we also generated a level 1 data set S_2 consisting of 50 patterns. Data set S_2 was used for model selection and model integration. A test data set S_3 of 10000 patterns without any noise was generated to measure the performance of the different models. The performance metric used was the normalized root mean square error (NRMS), which is the average root mean square error on the test data set normalized by the standard deviation of the data set. NRMS represents the fraction of unexplained standard deviation. Six different layered feedforward networks N_1, N_2, N_3, N_4, N_5 and N_6 were developed. N_1, N_2, N_3, N_4 had one hidden layer with 1, 2, 4 , and 8 hidden nodes respectively. N_5 had two hidden layers with 4 nodes in each layer and N_6 had two hidden layers with 8 nodes in each layer. All of the networks were trained using the scaled conjugate gradient algorithm (Moller, 1993). N_1, N_2 and N_5 were trained on all 100 patterns. For the remaining networks stopped training was used. These networks were trained on 70 randomly selected patterns and their performance was monitored on

the remaining 30 patterns. The weights which gave the least error on the 30 patterns were considered to be the optimal weights for these networks. Performance of the candidate networks on the data set S_3 was evaluated by computing the NRMS. The different ANNs were then stacked to obtain the SNN model. The NRMS for the ITS-based SNN was also computed. Using cross-validation, N_3 is selected as the best model. N_3 had an NRMS of 0.32 on the S_3. The competing networks were then combined to form a SNN using the ITS algorithm. The RMSE of the SNN on the test data was found to be 0.19. Thus, a reduction of 41% in the prediction error was obtained using SNN over the cross-validation selected network. The ITS algorithm determined that N_3, N_4, and N_6 contained 95% of the information important for predicting the output and used only these three networks to form the SNN. For comparison purposes we combined all of the six models and tested on S_3. The NRMS was 0.27, which is about 16% less than the NRMS of the single model. Notice that the improvement is significantly higher using the subset determined by the ITS algorithm as compared to using all of the candidate models. The example clearly demonstrates the ability of ITS to select and combine only those models with relevant information about the output resulting in a SNN which provides highly improved predictions.

Dynamic modeling of a CSTR

This example is intended to demonstrate the application of an ITS-based SNN for improved modeling of a chemical process. The performance of the ITS-based SNN will be compared to the SNN based on a linear combining technique and also to the traditional approach of selecting and using a single model. In addition to the standard layered feedforward networks, cascade correlation networks will also be considered as a candidate models in this example.

The process considered is an ideal, adiabatic continuous first order exothermic re-

action in a continuous stirred tank reactor (CSTR), shown in Figure 6. This system has been used for comparison of backpropagation networks and multivariate adaptive regression splines (DeVeaux et al., 1993). Recently, Sridhar *et al.* (1996) demonstrated the effectiveness of SNNs on this problem using linear combiners as the level 1 model. Here we will show that the nonlinear combination determined by the ITS algorithm is significantly better. The feed to the CSTR is pure A and the desired product is R. The system is simulated by the following coupled ordinary differential equations.

$$\begin{aligned} \frac{dA_o}{dt} &= \frac{A_i - A_o}{\tau} - k_1 A_o + k_{-1} R_o \\ \frac{dR_o}{dt} &= \frac{R_i - R_o}{\tau} + k_1 A_o - k_{-1} R_o \\ \frac{dT_o}{dt} &= \frac{-\Delta H_R}{\rho C_p}(k_1 A_o - k_{-1} R_o) + \frac{T - T_o}{\tau} \end{aligned} \tag{20}$$

where,

$$\begin{aligned} k_1 &= c_1 exp\left(\frac{-Q_1}{RT_o}\right) \\ k_{-1} &= c_{-1} exp\left(\frac{-Q_{-1}}{RT_o}\right) \end{aligned}$$

The variables A_o, R_o and T_o represent the state variables of the system and are the feed species concentration, the output product species concentration and the reactor's temperature, respectively. The goal of the modeling is to learn the open loop relationship between the controlled variable which is the concentration of species R and the manipulated variable which is the temperature of the feed stream, T_i. It is desired to predict the concentration of species R at the next time step, given the state variables A_o, R_o, T_o and the manipulated input T_i. The input and output dimensionality of the system are 4 and 1 respectively. The steady state operating point and the process parameters are as given in the paper by Economou *et al.*, and are shown in Table 2. A method similar to that in DeVeaux *et al.*, was used to generate the training data for the neural network modeling. The above set of differential equations were integrated with T varying randomly with a uniform distribution within 15% of the steady state

operating point. Sampling time for the process was 30 seconds. During the simulation, the systems state variable as well as manipulated input were recorded. At the end of the sampling instant, the system's response was also recorded. Measurement noise was added to the noise-free simulation. The noise was assumed to be Gaussian. A data set of 350 patterns was generated. The first 250 patterns were used to develop the model and the next 100 patterns were used to identify the best model and to develop the linear and nonlinear model combinations. A noise-free test data set for was also generated by simulating the process for 500 minutes. This data set was used to test and compare all the models that were developed.

Eight neural network models $[N_i : 1 \leq i \leq 8]$ were considered as the level 0 models for modeling the process. The architecture of the eight networks were:

1. BPN with 2 hidden layers. The number of nodes in each layer was three
2. BPN with 2 hidden layers. The number of nodes in each layer was eight
3. BPN with single hidden layer: Number of hidden nodes was three.
4. BPN with single hidden layer: Number of hidden nodes was fifteen.
5. CCN with sigmoidal activation function
6. CCN with sigmoidal activation function (Different weight initialization)
7. CCN with gaussian activation function
8. CCN with gaussian activation function (Different weight initialization)

Networks N_1, N_3, N_5, N_7 used convergent training. The remaining networks were trained using stopped training. For these networks the original data was split into two data sets with 70% and 30% of the data set respectively. The larger data set was used to train the network , while the error over the remaining data was monitored, and training was

stopped when the error reached a minimum. For training the backpropagation networks, the scaled conjugate gradient algorithm (SCGA) was used (Moller, 1993). The CCN was trained using the cascade correlation learning algorithm (Fahlman, 1990). Cross validation was used to determine the best network. In addition the networks were stacked with the linear technique and the ITS algorithm. The performance of the three approaches were then compared by testing them on the 10000 data patterns.

Model 2 (4x8x8x1) is determined to be the single best model using cross-validation as it has the least error over the data set D_2. A linear combination of the different networks was obtained. Finally the ITS algorithm was used to determine a nonlinear combination of the candidate networks. ITS selects N_2 as the first model in the combination with U(y | y_2) of 0.53. ITS then searches for a second model that adds the most information to the first model. As a result, N_6 is added to the combination with U(y|y_1, y_2) of 0.89. Further addition of any other model does not lead to much of an increase in the information content. Using all of the models leads to U(y |y_1,...y_8) of 0.91. A nonlinear combination was then developed by training a GRNN model on the level 1 data set.

All three approaches were then tested on D_3. The single model (4x8x8x1) BPN had an NRMS of 0.4 with R^2 of 0.84. The linear combination was slightly better with an NRMS of 0.37 and R^2 of 0.85. The ITS algorithm in comparison had an NRMS of 0.28, a reduction of 30% in NRMS as compared to using the single model and a reduction of 25% as compared to using the linear combiner. Graphical comparisons of the three different approaches are shown in Figure 7.

The superior performance of the ITS can be attributed to its ability to identify the most informative models regardless of the nature of its relationship to the actual output. Further the GRNN allows combinations to be estimated from the data without making any assumptions about the nature of the combination. The traditional approach uses only the information in Model 2 and ignores what the other models have to offer. The linear combiner is able to use only those models with a high linear correlation to the

output. The ITS is able to identify the models which have the most information and builds an effective nonlinear combiner resulting in highly improved predictive modeling of the process dynamics.

Conclusions

In this article we presented a new approach to combine neural network models for improved modeling of chemical processes. Conventional approaches identify and use a single model for prediction. More recently stacked neural networks (SNNs) have attempted to integrate the knowledge acquired by different networks to obtain a better predictive model. However, SNNs were limited to a linear combination of the candidate networks. We have pointed out the limitations of such an approach. The ITS algorithm proposed in this work effectively combines candidate networks without making any assumptions about the combining rule. Instead ITS determines the appropriate combination from the data. ITS first generates a number of subsets and selects an informative subset of models which are then combined using a nonparametric modeling technique. In this work we used forward selection to generate the different subsets and the general regression neural network (GRNN) model was used to build the combination.

The ITS algorithm was applied to three examples including the dynamic modeling of a chemical process. The examples demonstrate that the ITS-based stacked neural networks consistently outperformed the approach of using a single model or a linear combination of the candidate models. The ITS-based stacked neural network appears to be a promising technique for improved modeling of chemical processes. Future research needs to focus on alternate strategies for generating subsets and combining methods based on models other than the GRNN should also be investigated.

Acknowledgments: This work was made possible by the generous support of the Min-

nesota Mining and Manufacturing Company. Their support is greatly appreciated, however it does not constitute an endorsement of the views expressed in this article.

References

Bhat, N., and T. J. McAvoy, "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process System," *Comput. Chem. Eng.*, 14(4/5), 573 (1990)

Cherkassky, V., Gehring, D., and Mulier, F., "Pragmatic Comparison of Statistical and Neural Network Methods for Function Estimation," *Proc. World Cong. on Neural Networks Conf.*, Vol. 2, Washington, DC, p. 917 (1995)

Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Math. Contr. Signals Syst.*, 2, 303 (1989)

De Veaux, R. D., D. C. Psichogios and L. H. Ungar, " A Comparison of Two Nonparametric Estimation Schemes: MARS and Neural Networks," *Comput. Chem. Eng.*, 17, 819 (1993)

Efron, B., "An introduction to the boostrap," Chapman & Hill, New York (1993)

Friedman, J. H., "Multivariate Adaptive Regression Splines," *Ann. Statist.*, 19, 1 (1991)

Fukunaga, K., "Introduction to Statistical Pattern Recognition," Academic Press, Boston (1990).

Hansen, L. K., and P. Salamon, "Neural Network Ensembles," *IEEE Trans. Pattern Anal. and Mach. Intell.*, 12(10), 993 (1990)

Hecht-Nielsen, R., "Neurocomputing," Addison Wesley, Reading, MA(1990)

Hornik, K., M. Stichcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, 2, 359 (1989)

Hoskins, J. C., and D. M. Himmelblau, "Artificial Neural Network Models of Knowledge Representation in Chemical Engineering," *Comput. Chem. Eng.*, 12, 881 (1988)

Joseph, B., F. H. Wang, and D. S. Shieh "Exploratory Data Analysis: A Comparison of Statistical Methods with Artificial Neural Networks," *Comput. Chem. Eng.*, 16, 413 (1992)

Lippmann, R. P., "An Introduction to Computing with Neural Nets," *IEEE, Acoustics Speech and Signal Processing Magazine*, 4, 4 (1987)

Fahlman, S. E., and C. Lebiere, "The Cascade-Correlation Learning Architecture," Advances in Neural Information Processing Systems 2, D. S. Touretzky, ed., Morgan Kauffman, San Mateo, CA, p524 (1990)

Moller, M. F., "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, 6(4), 525 (1993)

Pollard, J. F., M. R. Broussard, and D. B. Garrison, "Process Identification Using Neural Networks," *Comput. Chem. Eng.*, 16, 253 (1992)

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "Numerical Recipes: The Art of Scientific Computing," Cambridge University Press, New York (1986)

Shannon, C. E., and W. Weaver, "The Mathematical Theory of Communication," University of Illinois Press, Urbana (1949)

Specht, D. F, "A General Regression Neural Network," *IEEE Transactions on Neural Networks*, 2, 568 (1990)

Sridhar D. V., R. C. Seagrave, and E. B. Bartlett, "Process Modeling Using Stacked Neural Networks," *AIChE J.*, 42, 2529 (1996)

Venkatasubramanian, V., and K. Chan, "A Neural Network Methodology for Process Fault Diagnosis," *AIChE J.*, 35, 1993 (1989)

Watanabe, S., "*Knowing and Guessing: A Quantitative Study of Inference and Information*," John Wiley and Sons, New York (1969)

Weiss, S. M. and Kukilowski C. A., "Computer systems that learn," Morgan Kaufman, San Mateo, CA (1991)

Werbos, P. J., "The Roots of Backpropagation," John Wiley and Sons (1994)

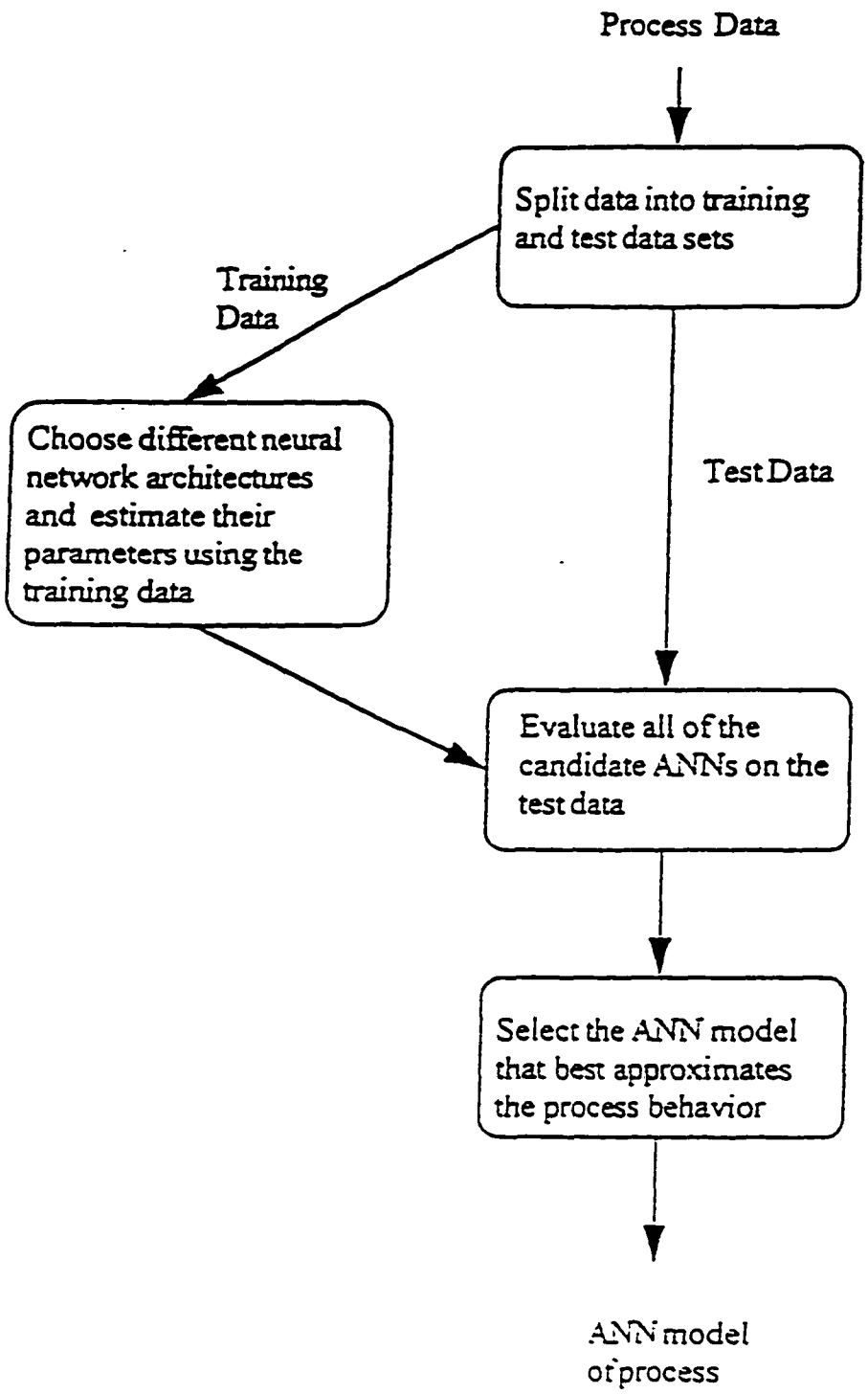
Wolpert, D. H., "Stacked Generalization," *Neural Networks*, 5(2), 241 (1992)

Table 1 Level 1 data for Example 1 showing the outputs of N_1 and N_2 and the actual output

Index	Output of N_1	Output of N_2	Actual Output	Pattern Frequency
1	0	0	0	0
2	0	1	0	250
3	1	0	0	244
4	1	1	0	6
5	0	0	1	236
6	0	1	1	8
7	1	0	1	7
8	1	1	1	249

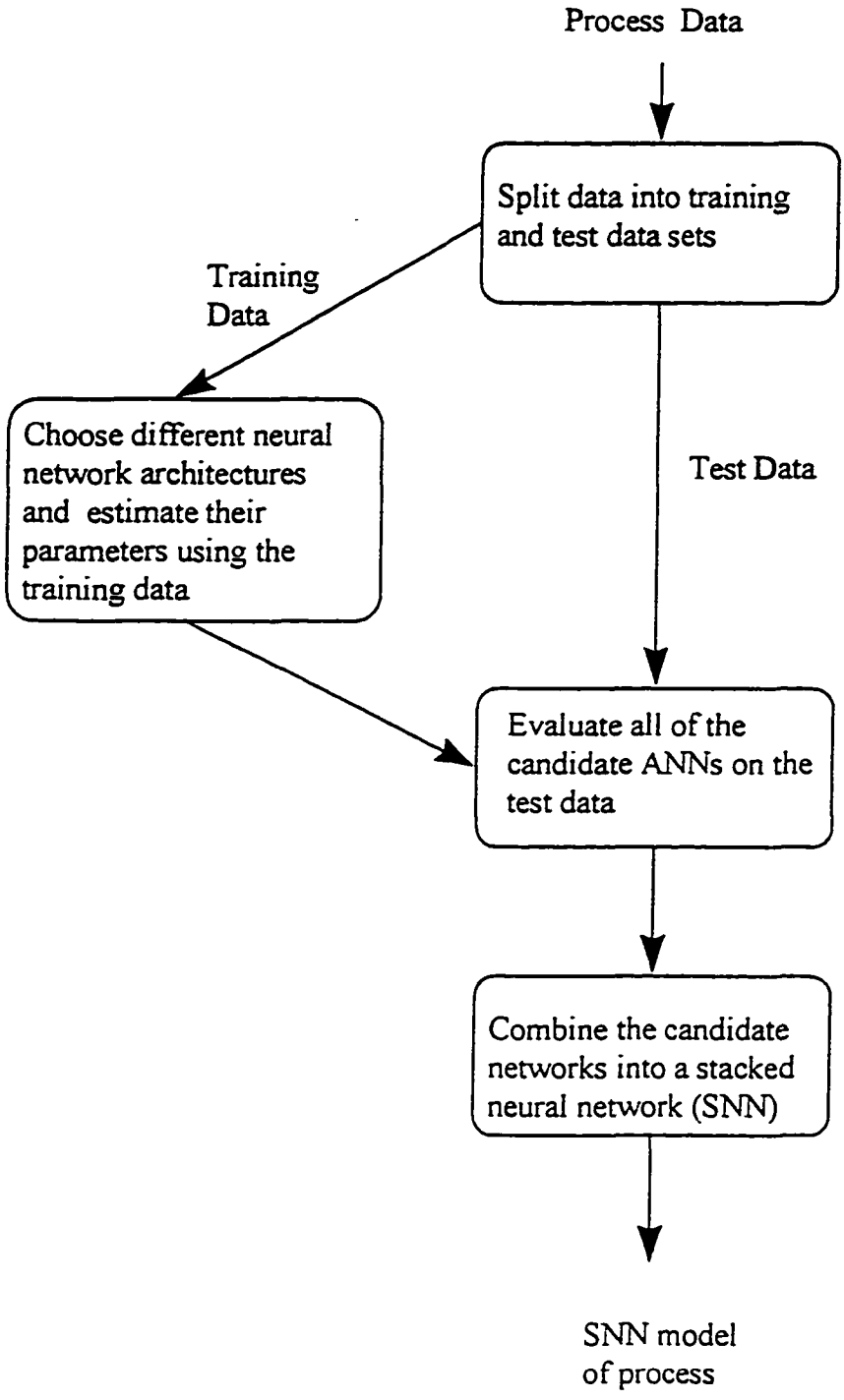
Table 2 Constants and Steady-State Operating Conditions for the CSTR

τ	60s
c_1	5x10^3 s^{-1}
c_{-1}	1x10^6s^{-1}
Q_1	10000 cal mol^{-1}
Q_{-1}	15000 cal mol^{-1}
R	1.987 cal mol^{-1} K^{-1}
ΔH_R	5000 cal mol^{-1}
ρ	1 Kg/L
C_p	1000 cal kg^{-1} K^{-1}
A_i	1.0 mol/L
R_i	0.0 mol/L
A_o	0.492 mol/L
R_o	0.508 mol/L
T_i	427K
T_o	430K



1(a): Traditional neural network process modeling

Figure 1 Comparison of traditional and stacked neural network modeling



1(b): Stacked neural network process modeling

Figure 1 continued

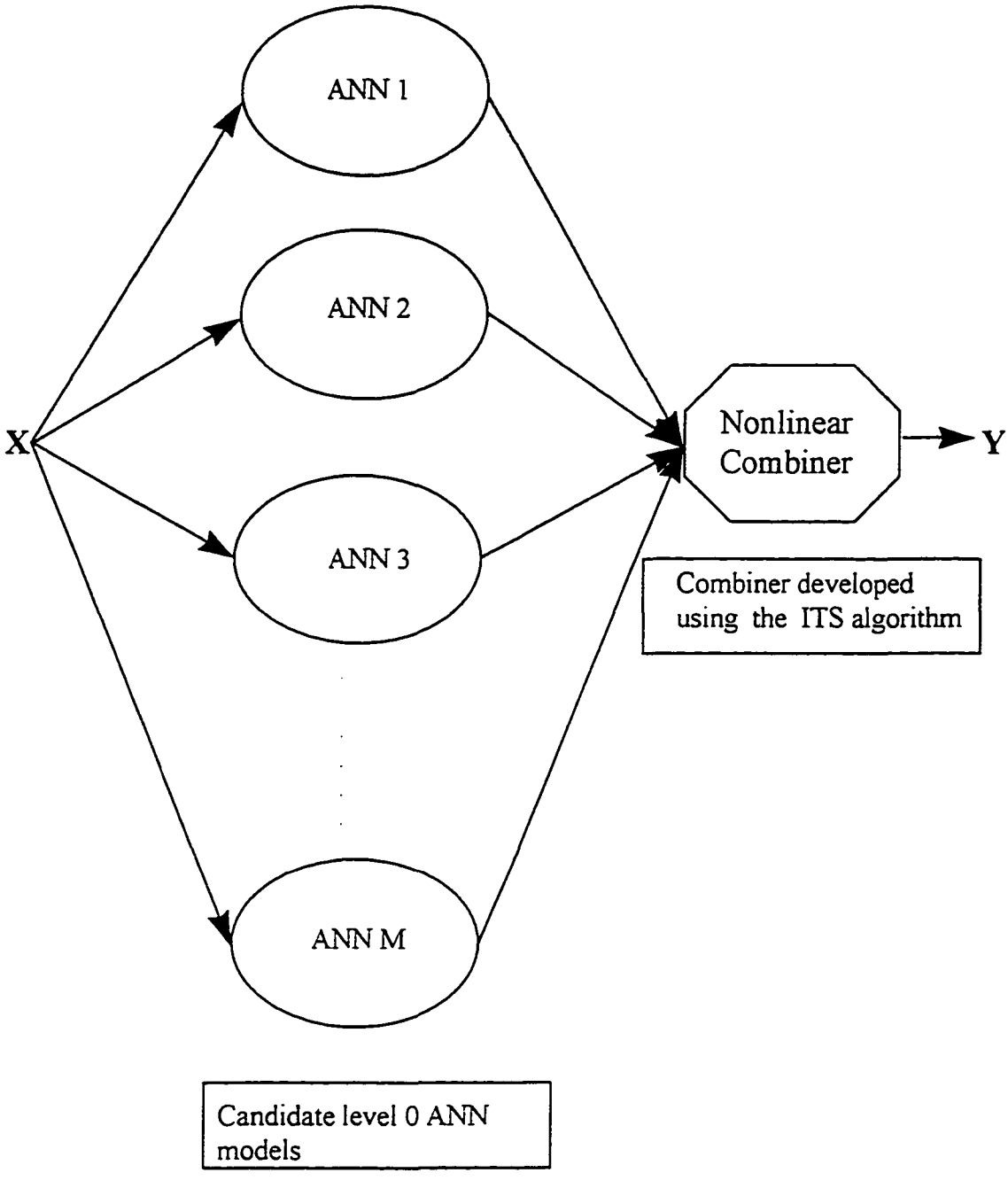


Figure 2 Architecture of stacked neural networks using the ITS algorithm

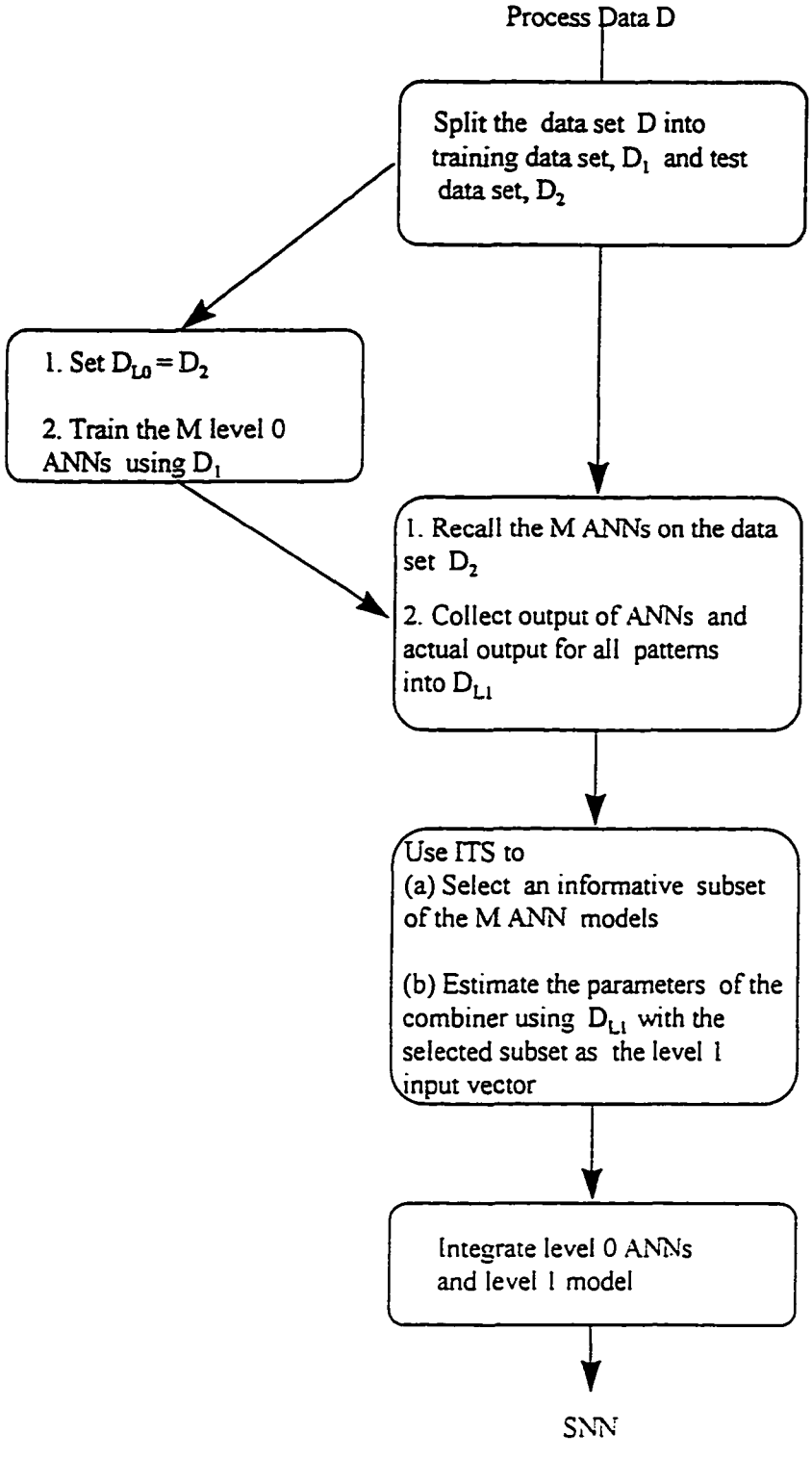


Figure 3 Schematic for developing the SNN using the ITS algorithm

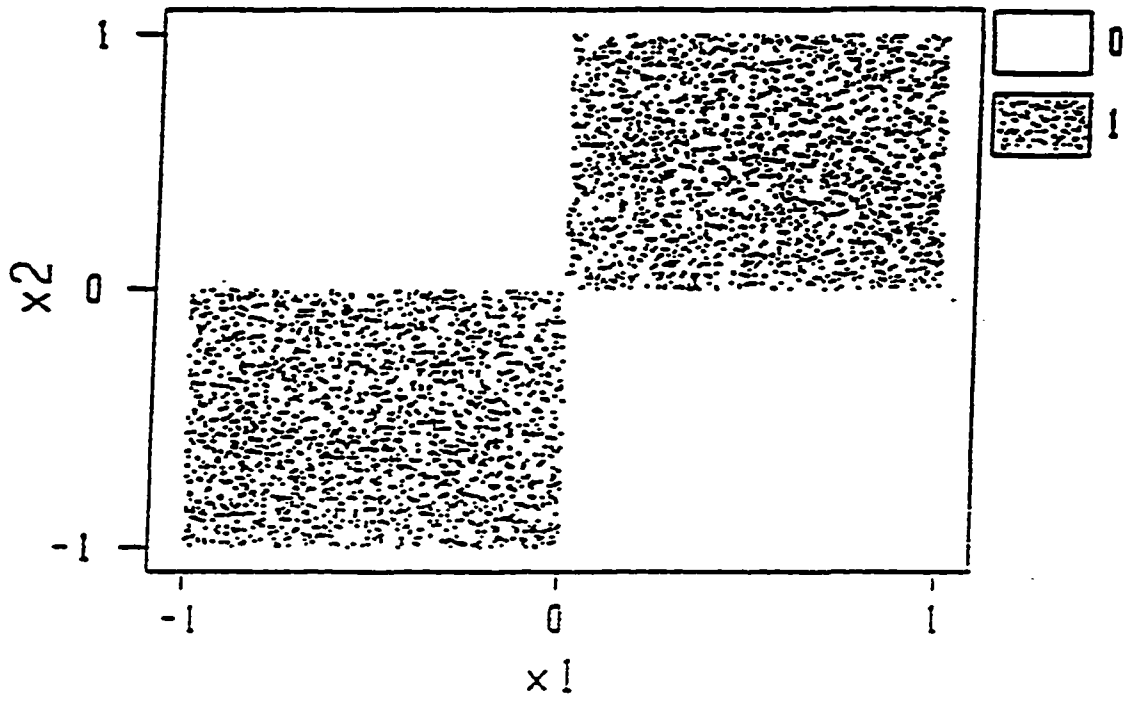


Figure 4 The Generalized XOR

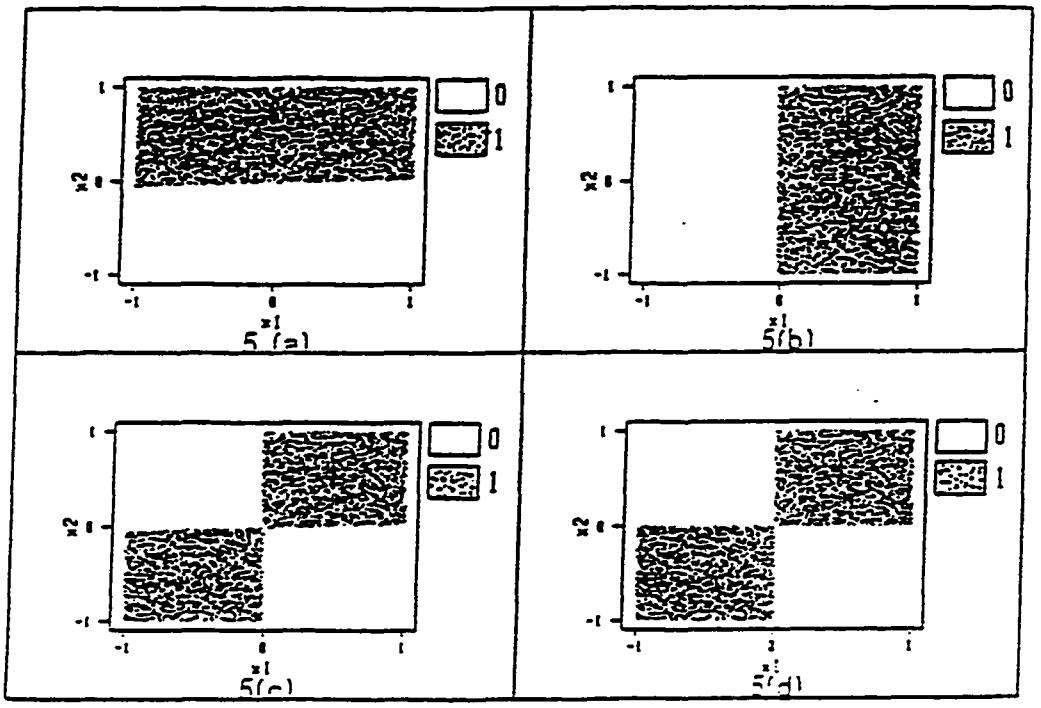


Figure 5 Performance of the ITS-based SNN for the Generalized XOR problem

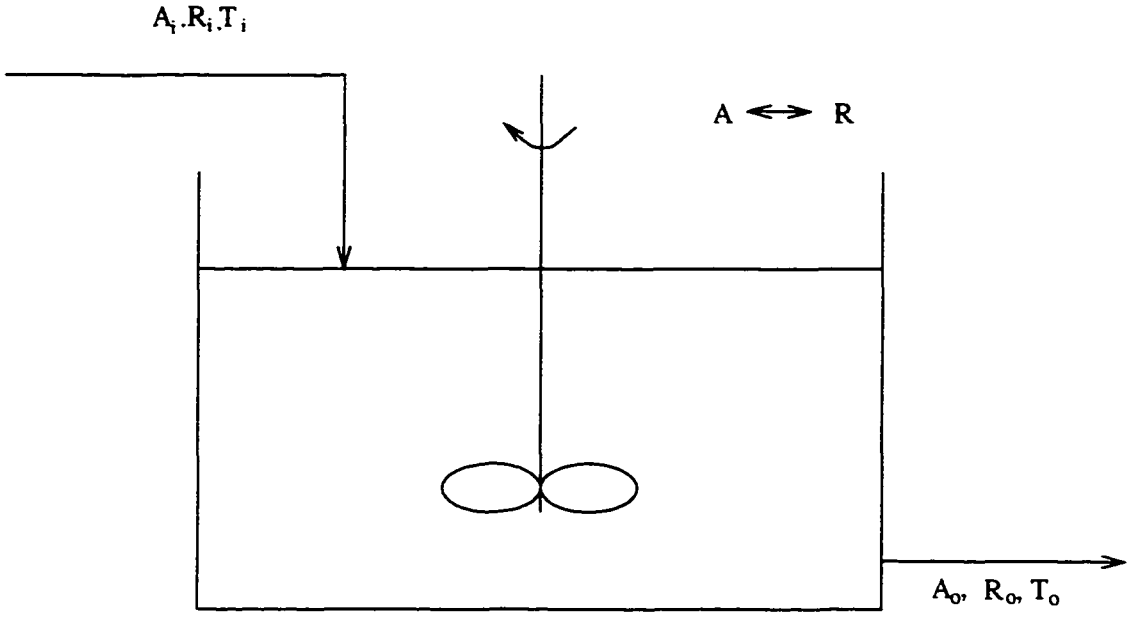
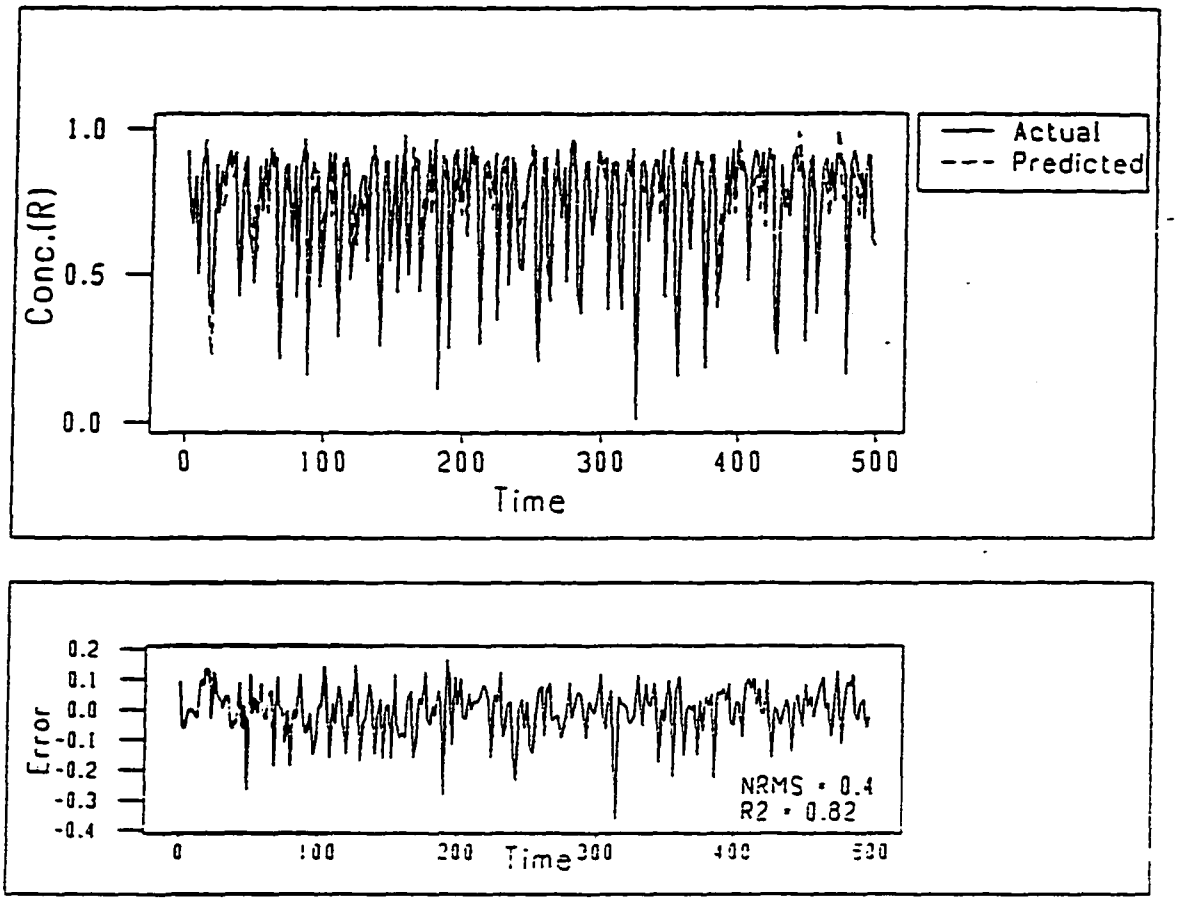
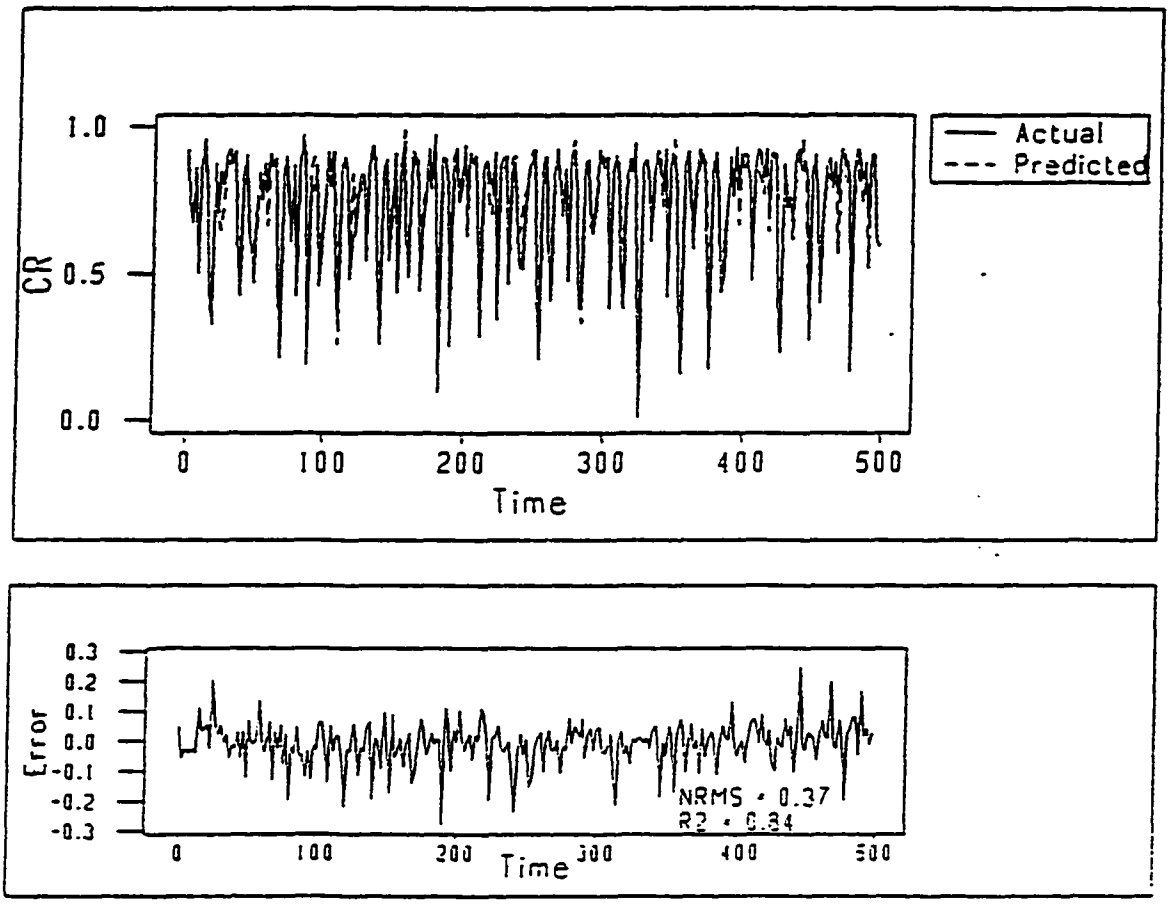


Figure 6 Schematic diagram of CSTR



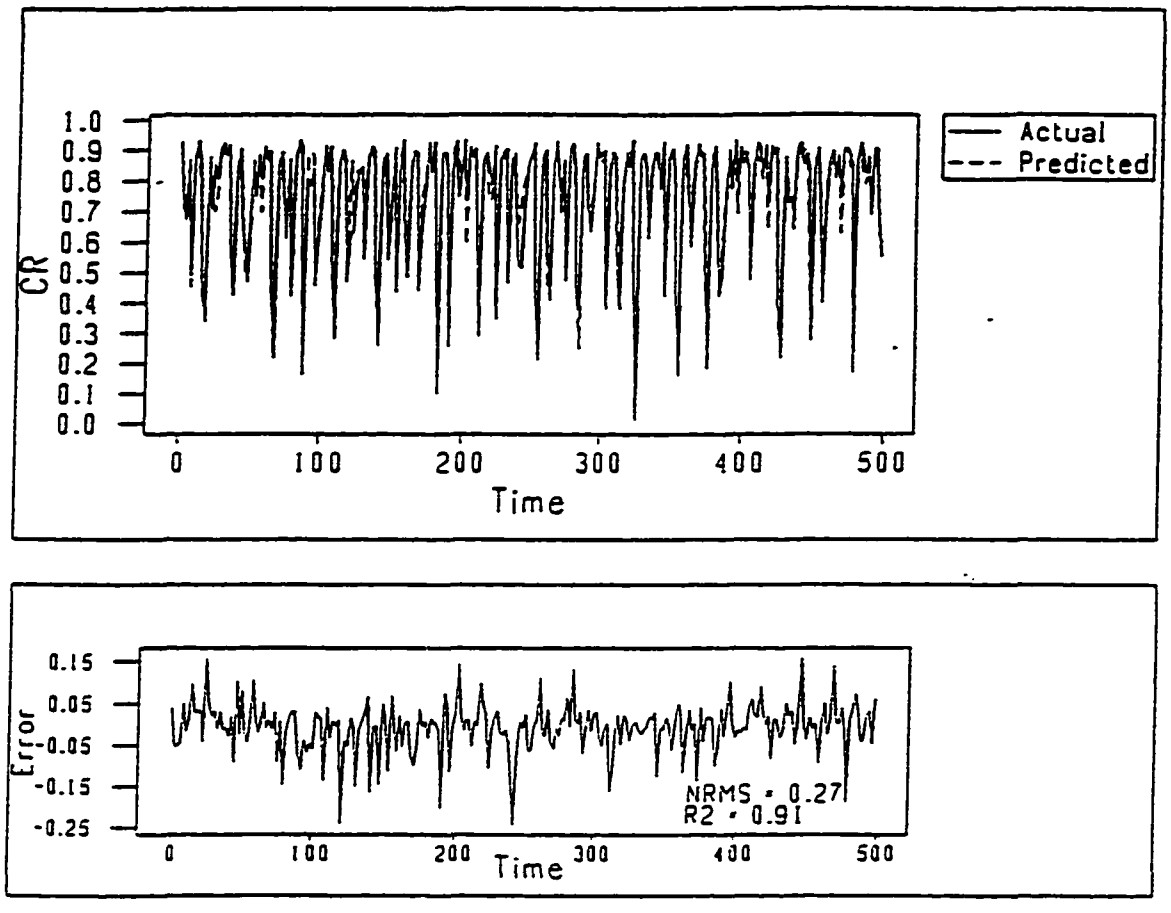
7(a)

Figure 7 Comparison of the three approaches on the CSTR problem (a) single model (b) SNN using a linear combiner (c) SNN using ITS



7(b)

Figure 7 continued



7(c)

Figure 7 continued

GENERAL CONCLUSIONS

In this dissertation, a new approach for neural-network-based modeling of chemical processes was proposed. Neural network modeling typically require multiple networks to be trained and tested. Conventional approaches identify and use a single hopefully optimal neural network model and discard the remaining networks. Such an approach wastes information in the discarded networks and may lead to a suboptimal process model. The stacked neural network approach presented in this dissertation allows multiple neural networks to be combined and used to model a process. The approach effectively integrates candidate networks to obtain better process models. The feasibility of the approach was first demonstrated using a stacked neural network that consisted of a linear combination of the candidate networks. The stacked neural network was shown to outperform the conventional approach of using a single hopefully optimal model. However, the linear combining rule can integrate only those networks which have a high linear correlation to the output. The information theoretic stacking (ITS) algorithm was then developed to overcome the limitations of a linear combination. The ITS approach does not assume the form of the combining rule. Instead the ITS allows the form of the combination to be determined by the data. The ITS-based stacked neural network was evaluated for several example problems including modeling of a chemical process. Results obtained show that the ITS-based SNN consistently outperformed the approach of using a single model or an SNN based on a linear combining rule. The ITS algorithm used the general regression neural network model to develop the combination. The general regression neural network model was used because it is a memory based

system and therefore easy to develop. However, the ITS approach is general in that any nonparameteric model can be used to develop the combination. Future research needs to focus on combining methods based on models other than the general regression neural network model.

ADDITIONAL REFERENCES

Barron, A. R., "Approximation and Estimation Bounds for Artificial Neural Networks," *Machine Learning*, 14, 115 (1994)

Borowiak, D. S., "Model Discrimination for Nonlinear Regression Models," Marcel Dekker, NewYork (1989)

Cheng, B., and Titterington, D. M., "Neural Networks: A Review from a Statistical Perspective," *Statistical Science*, 9(1), 2 (1994)

Foster, W. R., F. Collopy, and L. H. Ungar, "Neural Network Forecasting of Short, Noisy Time Series," *Computers and Chemical Engineering*, 16, 293 (1992)

Kresta, J. V., J. F. MacGregor, and T. E. Marlin, "Multivariate Statistical Monitoring of Process Operating Performance," *The Canadian Journal of Chemical Engineering*, 69, 35 (1991)

Rummelhart, D. E., G. E.Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol 1., p318, MIT Press (1986)

Schaffer, C., "Selecting a Classification Method by Cross-validation," *Machine Learning*, 13, 135 (1993)

Wasserman, Philip D., "Advanced Methods in Neural Computing," Van Nostrand Reinhold, New York (1993)

Wetherill, G. B., "Regression Analysis and Applications," Chapman and Hall, New York (1990)

Zhang, X., J. P. Mesirov and D. L. Waltz, "Hybrid System for Protein Secondary Structure Prediction," *Journal of Molecular Biology*, 22, 1049 (1992)