ON THE USE OF VARIOUS INPUT SUBSETS FOR STACKED GENERALIZATION

ERIC B. BARTLETT AND ADAM P. WHITNEY

Department of Electrical and Computer Engineering,

Iowa State University, Ames, IA, 50011

ABSTRACT

Artificial Neural Networks (ANNs) can be useful for modeling real - world processes such as time series weather, financial, or chaotic data. The generalization and robustness of these models can be improved and estimates of the modeling error distributions can be made using a technique called Stacked Generalization (SG). SG, introduced by Wolpert and used by Kim, Sharkness, Sridhar, and Bartlett, has shown the ability to improve ANN model viability. SG uses a number of diverse ANN models, each of which is trained and queried on independent cross validation subsets of the process data. These models are then compared and the resulting information is used in the stacking process to train additional ANNs to combine the models and provide error estimates. However, since the SG improvements and error estimates depend on the individual model response diversity, we add the additional feature to SG of allowing these networks to have different input variable subspace vectors. Thus, each model's response expresses different features of the process to be modeled and a more complete representation of the overall process is obtained. This ability to have different input vectors improves overall accuracy. Examples are provided which demonstrate the advantages of our modifications to SG.

INTRODUCTION

Artificial neural network (ANN) modeling techniques have many fascinating characteristics. For example, ANNs are capable of learning by example and learning without explicit knowledge of the underlying system which they are mimicking [Rumelhart, McClelland & the PDP Research Group, 1986; Lippmann, 1987; Blum and Li, 1991; Hecht-Nielsen, 1990; Kurkova, 1992; Haykin, 1999]. Fortunately, ANNs are capable of generalizing this knowledge. Numerous applications of ANNs exist in engineering [Lapedes & Farber, 1987; Narendra & Parthasarathy, 1990; Zhang, Mesirov & Waltz, 1992], process control and planning [Bhat & McAvoy, 1990; Miller, Sutton & Werbos, 1990], plant monitoring [Uhrig, 1989; Upadhyaya & Eryurek, 1992], and fault diagnosis [Venkatasubramanian & Chan, 1989; Bartlett & Uhrig, 1992]. There are countless others.

Unfortunately, the lack of validation and verification options for ANN methods restricts the areas where they can be applied successfully. Only the courageous apply ANNs to important real – world applications. Typically, ANN users assume that the output of their ANNs are, in fact, reliable. They certainly hope so. However, the nonlinearity of neural networks provides no guarantees on the behavior of the *a posteriori* model prediction errors. Without a model error estimate – who would know about a model's shortfalls? This paper addresses an improved solution to this error prediction deficiency in the application and use of ANN techniques. Our method, called Modified Series Association (MSA) provides a reliable error estimate on each and every model prediction and does so in a concise and efficient manner.

Stacked Generalization (SG) was proposed as a method of using multiple models to provide improved accuracy or confidence intervals [Wolpert, 1992]. Wolpert suggests using a number of models, in our case ANNs, on different subsets of the data, in order to obtain models that are slightly different. These models are then recalled over the remainder of the data and this information is used in the stacking process. We propose to go an additional step further to obtain model diversity. We add the additional complexity of allowing the models to have different input sub-vectors of the input space. Obviously, this works well only on problems that are over - specified in the input vector space.

The information generated by querying these various networks on their respective cross validation sets constitutes new information, since they are not expected to produce exactly identical results. This information is then used to develop the error predictor and consolidation models. SG is a method of developing a stacked model that utilizes the results of two levels of multiple networks or generalizers. In our case, the first level of models, called the Level 0 networks, are trained not only on partitions of the given data set, but also subspaces input vector. The Level 0 models are then queried, or recalled, on the unused, or cross validation, data. The results of the recall of the Level 0 models on these “novel” cross validation partitions are then stored. The second level of models, called the Level 1 models, are then developed using the results of this Level 0 cross validation recall. These Level 1 models provide the consolidated, or stacked, model output and the corresponding errors for each prediction. These Level 1 results are based on the diverse behavior of the Level 0 models. Once the Level 1 models are created with the Level 0 cross validation data sets, then these Level 0 models are discarded. New Level 0 models are then developed on the complete training data set without partition using their appropriate input subspace. These new networks are then used in the stacked recall process to generate consolidated predictions and their associated uncertainties. We have shown this approach to be effective in conducting error analyses on ANN models for pattern recognition [Kim and Bartlett, 1996] and to provide improved functional models of the desired outputs [Sridhar, 1996]. When properly combined, these various models yield reliable error estimates for ANN function approximation, as will be shown in this paper.

MODEL CONSOLIDATION AND ERROR ESTIMATION

ANNs can be regarded as generalizers because they infer parent functions from sets of data [Cybenko, 1989; Wolpert, 1990; Kurkova, 1992]. Most other modeling methods can also be considered to be generalizers as well. For example, statistical and even first principle methods can be considered as generalizers. Therefore, the following discussion can be applied to a large class of modeling methods. It is, however, difficult to find methods for error estimation and consolidation of general data driven nonlinear modeling techniques such as ANNs, and this is where our MSA technique can be used to full effect [Narendra and Parthasarathy, 1990; Blum & Li, 1991; Bartlett, 1992 & 1994; Bartlett & Kim, 1993].

As a preliminary to our discussion of MSA, we need to define some useful concepts for the continuous modeling problem of interest. Given a set of N data exemplars of the input vector $\mathbf{x}$ and the output vector $\mathbf{y}$ to be modeled $\{\mathbf{x}, \mathbf{y}\}_N$ which may possibly contain noise, for n = 1,2, …, N, where $\mathbf{y}$ is a function of $\mathbf{x}$ and time and $\mathbf{x}$ is also a function of time. We have;

$$\mathbf{y}(t) = F\{\mathbf{x}(t), \mathbf{x}(t\text{-}1), \dots, \mathbf{x}(t\text{-}L); \mathbf{y}(t\text{-}1), \mathbf{y}(t\text{-}2), \dots, \mathbf{y}(t\text{-}K)\} \tag{1}$$

Where N is the total number of data exemplars patterns at our disposal at the time of model development. And $\mathbf{x}_n(t)$ and $\mathbf{y}_n(t)$ are vectors such that,

$$\mathbf{x}_n(t) = \{x_{n,1}(t), x_{n,2}(t), \dots, x_{n,I}(t)\} \tag{2}$$

and

$$\mathbf{y}_n(t) = \{y_{n,1}(t), y_{n,2}(t), \dots, y_{n,J}(t)\} \tag{3}$$

And, I and J are respectively the dimensions of the vectors **x**(t), and **y**(t). Therefore, **y** is a dependent variable on **x**. Then $\mathbf{x}_n(\mathrm{t}) \in \mathrm{R}^{\mathrm{I}}$ and $\mathbf{y}_n(\mathrm{t}) \in \mathrm{R}^{\mathrm{J}}$. The dimension of the input space of the unknown, yet desired, function $F(\cdot)$ is then $I * L + J * K$. The objective is to find a useful approximation to $F(\cdot)$, such that a good estimate of $\mathbf{y}(t) = F(\cdot)$ is $\hat{\mathbf{y}}(t) = \hat{F}(\cdot) \cong F(\cdot)$, and a reasonable estimate of the uncertainty associated with this approximation of **y**(t) is $\hat{\sigma}(t)$, so that the model predictions can be reported as $\hat{\mathbf{y}}(t) \pm \hat{\sigma}(t)$ and the true output **y**(t) is bounded by

$$\hat{\mathbf{y}}(t) \; - \; 2\hat{\sigma}(t) \;\; < \;\; \mathbf{y}(t) \;\; < \;\; \hat{\mathbf{y}}(t) \; + \; 2\hat{\sigma}(t) \tag{4}$$

to some statistical certainty, say, optimistically 95%. In the past, ANNs could not be used for modeling with this objective because their nature seemed to preclude error estimates. By combining the responses of a number of ANNs, using a method called Stacked Generalization (SG), of which our MSA is an extension, we can satisfy this objective as embodied in Equation 4.

A very simple approach for model consolidation and error estimation we call here "poor - man's stacking". Haykin describes this method as an ensemble averaging method for Committee Machines [1999]. By any name, poor - man's stacking of M individual models is accomplished by applying simple statistics to the outputs of the various models developed. The reported consolidated value, $\overline{\mathbf{y}}(t)$, for the output of interest, is the average of M models;

$$\overline{\mathbf{y}}(t) \; = \; \{\hat{\mathbf{y}}_1(t) \; + \; \hat{\mathbf{y}}_2(t) \; + \; \ldots \; + \; \hat{\mathbf{y}}_M(t)\} \, / \, M. \tag{5}$$

The consolidated model error $\hat{\sigma}(t)$ is then reported as the standard deviation amongst them;

$$\hat{\sigma}(t) = \sigma\{\hat{\mathbf{y}}_1(t), \hat{\mathbf{y}}_2(t), \ldots, \hat{\mathbf{y}}_M(t)\}. \tag{6}$$

This method for model consolidating is not only unsophisticated but also unappealing. Experience shows that this approach does not work very well in general. Therefore, a more sophisticated approach must be used.

Stacked generalization, unlike cross validation, can, not only be used to improve the generalization accuracy for one or more generalizers, but it can also be used to estimate the expected prediction errors of a generalizer that is presented with novel input. Let $\hat{F}_{L_0}$ be an ANN trained on a learning set L_0. Our goal is to estimate error bounds on outputs obtained by $\hat{F}_{L_0}$ for novel inputs that are not members of L_0. With a chosen partition criterion, let the learning set L_0 be partitioned into subsets. These subsets embody both the individual data exemplar partitions as well as our input subset vectors. Then let an ANN, called the Level 0 generalizer $\hat{F}_{L_0,i}$, is trained on the union of all except one of the j^{th} partitioned subsets and its associated input subspace of L_0. The performance of this model is measured by testing it on the remaining partitioned subset. Since this subset is not used in training of $\hat{F}_{L_0,i}$, the output of $\hat{F}_{L_0,i}$ will, in general, deviate from the desired output. Note that this desired output information is known from the unused data set partition. This deviation represents the error between the desired output and the ANN's output on novel data. Another partition is chosen and the process of training and testing is repeated until all of the partitions are queried, and the entire data set is modeled as novel or untrained. This data, which includes the untrained input, a distance measure from the untrained input to the nearest trained input in each respective subspace, and the output estimates, constitute a new learning space L_1 called the Level 1 learning set. A new ANN called the Level 1 generalizer $\hat{F}_{L_1}$ is trained on this Level 1 learning set L_1. The Level 1 generalizer $\hat{F}_{L_1}$ then learns the relationships between the desired outputs, and the known inputs, including the outputs of the various Level 0 models and the estimates made by the Level 0 networks.

MODIFIED SERIES ASSOCIATION TRAINING

Assume, for the moment, that we will use five fold cross validation for the partition criteria on the Level 0 data set. Thus, we partition the data into five approximately equal size subsets, where the union of these subsets is the entire original training set. The partition is performed without replacement so that no data exemplar is a member of more than one subset. If we then perform a ranking using forward GRNN ranking, we can obtain our various input subset spaces. Only the basics of our ranking methods are discussed here. The interested reader is pointer to Wichmann and Bartlett [1997] and Wichmann [1997].

GRNNs fit a regression surface to an underlining function and can therefore be used for modeling continuous variables [Specht, 1990]. To create a GRNN the user must estimate the underlying joint probability density function (pdf) of the inputs and the outputs, **x** and **y**, using existing data. To estimate the joint pdf, an estimator, first proposed by Parzen [1962], is used. When a forward GRNN ranking is made, GRNN models are made using just one of the input variables, each variable in turn, until a model has been made for each input variable. Then, the single variable that yields the lowest predicted error sums of squares (PRESS) is retained and models with two inputs are attempted. Once the best pair of variables is found then models with 3 inputs are developed. And so on, as a forward selection process. The order and corresponding PRESS and R^2 are recorded in a file for later examination. If we perform this ranking on our partitioned subsets of the data, we obtain various rankings and this is what we use to determine the input subspaces used for the various Level 0 networks.

Thus, we train five ANNs, each on the union of four of the five subsets, leaving one subset out each time. Then we will be able to recall each of the five ANNs on a partition of the data for which it was not trained. We can then use these five ANNs to recall the entire data set as if it were novel – since each untrained subset is indeed novel to each of the five networks. We store this information since we will use it for the Level 1 training set later on. In mathematical notation we will have, for the five networks trained of the four subsets;

$$\hat{\mathbf{y}}_{i,j,80\%} = \hat{F}_{L_0}(\mathbf{x}_{80\%}) . \tag{7}$$

We proceed for each of the five fold partitions *j* = 1, 2, 3, 4, 5, each time leaving out 20% of the data. The next step is to train the entire data set one more time using a sixth neural network, thus,

$$\hat{\mathbf{y}}_{i,100\%} = \hat{F}_{L_0}(\mathbf{x}_{100\%}) \tag{8}$$

for *i* = 1, 2, 3, ..., I, where *i* is the index of the numerous Level 0 networks, each of which may utilize a different subspace of the input data. We repeat this procedure for of each of the *i* Level 0 networks. We would like to have a sufficient number of Level 0 models – on the order of 3 to 5 typically. These Level 0 models should also be as diverse as possible in accordance with the necessity of good training and model building. For example, including a radial basis function network (RBFN) or general regression neural network (GRNN) as well as multilayer perceptrons (MLPs) with various architectures is a good idea. This model diversity is required because we want the Level 0 models to have small modeling errors that have different characteristics.

Once the Level 0 models are trained, the Level 1 training can be initiated. We train the Level 1 models using the Level 0 models that were developed previously:

$$\overline{\mathbf{y}}_{5\text{x}20\%} = \overline{F}_{L_1}\{\mathbf{x}_{100\%}, \hat{\mathbf{y}}_{1,5\text{x}20\%}, \hat{\mathbf{y}}_{2,5\text{x}20\%}, ..., \hat{\mathbf{y}}_{I,5\text{x}20\%}, \text{d}(\mathbf{x}_{20\%}^{80\%})_1, \text{d}(\mathbf{x}_{20\%}^{80\%})_2, ..., \text{d}(\mathbf{x}_{20\%}^{80\%})_I\} . \tag{9}$$

Where $\hat{\mathbf{y}}_{i,5\text{x}20\%}$ is the i^{th} Level 0 model output, and $\text{d}(\mathbf{x}_{20\%}^{80\%})_i$ is the i^{th} distance metric measuring the distance between the novel inputs and the training data in its respective subspace. Thus, the output, $\overline{\mathbf{y}}_{5\text{x}20\%}$ is the stacked estimate of the desired output and it depends on not only the inputs, $\mathbf{x}_{100\%}$, but also the responses of the various novel Level 0 estimates of the outputs of interest. The error model is trained as follows,

$$|\overline{\mathbf{y}}_{5\text{x}20\%} - \mathbf{y}\ \ | =$$

$$|\overline{\mathbf{u}}_{5\text{x}20\%}| = \overline{F}_{L_1}\{\mathbf{x}_{100\%}, \hat{\mathbf{y}}_{1,5\text{x}20\%}, \hat{\mathbf{y}}_{2,5\text{x}20\%}, ..., \hat{\mathbf{y}}_{I,5\text{x}20\%}, \overline{\mathbf{y}}_{5\text{x}20\%}, \text{d}(\mathbf{x}_{20\%}^{80\%})_1, \text{d}(\mathbf{x}_{20\%}^{80\%})_2, ..., \text{d}(\mathbf{x}_{20\%}^{80\%})_I\}$$

(10)

The overscore indicates Level 1 models, $\mathbf{x}_{100\%}$ is an input subspace without partition, $\hat{\mathbf{y}}_{i,5\text{x}20\%}$ is the output from the i^{th} networks at Level 0, each recalled on their novel, $\mathbf{x}_{20\%}$, partition. Note the additional input of $\overline{\mathbf{y}}_{5\text{x}20\%}$ in the uncertainty model. The $\overline{\mathbf{y}}_{5\text{x}20\%}$, from Level 1 dictates that the Level 1 output model must be developed before the error model is trained. The distance metrics, $\text{d}(\mathbf{x}_{20}^{80})_i$, are included to provide measures of the amount of extrapolation or interpolation required of the models in Level 0. This metric measures the Euclidean distance between the nearest exemplar in the training data, $\mathbf{x}_{80\%}$, and the novel data $\mathbf{x}_{20\%}$ exemplar. Many types of distance measures are possible, and in fact, Wolpert suggests using a distance vector, however this increases the dimensionality of the error model's input space dramatically and, as we show in this paper, is unnecessary. Thus, $|\overline{\mathbf{u}}_{5\text{x}20\%}|$ is the error estimate as a relationship of the known inputs, the responses of the Level 0 and Level 1 models recalled on their respective novel cross validation data sets, and a distance metrics $\text{d}(\mathbf{x}_{20}^{80})_i$ which measure the distance between the novel patterns and the training set.

THE RECALL METHOD

The following is the method for finding the MSA prediction and error estimate for an unknown, novel, data exemplar. First, we recall each of the Level 0 networks on the new, novel, exemplars. Note that these Level 0 networks were the ones that were trained on the entire data set from above. Also, note that we typically have no desired outputs for these exemplars. In the notation of the previous section, we have;

$$\hat{\mathbf{y}}_i = \hat{F}_{L_0,100}(\mathbf{x}_{\text{novel}}) . \tag{11}$$

for each of the *i* Level 0 networks. Once the Level 0 networks are queried, the Level 1 networks are called upon to provide $\overline{\mathbf{y}}$ and $|\overline{\mathbf{u}}|$. Thus, $\overline{\mathbf{y}}$ is obtained using the Level 0 predictions as input to the Level 1 network,

$$\overline{\mathbf{y}}_{\text{novel}} = \overline{F}_{L_1}\{\mathbf{x}_{\text{novel}}, \hat{\mathbf{y}}_1(\mathbf{x}_{\text{novel}}), \hat{\mathbf{y}}_2(\mathbf{x}_{\text{novel}}),..., \hat{\mathbf{y}}_I(\mathbf{x}_{\text{novel}}), \text{d}(\mathbf{x}_{\text{novel}}^{\text{train}})_1, \text{d}(\mathbf{x}_{\text{novel}}^{\text{train}})_2, \ldots, \text{d}(\mathbf{x}_{\text{novel}}^{\text{train}})_I\} \tag{12}$$

Or, alternately, the original inputs can be ignored at Level 1 for both training and testing, then

$$\overline{\mathbf{y}}_{\text{novel}} = \overline{F}_{L_1}\{\hat{\mathbf{y}}_1(\mathbf{x}_{\text{novel}}), \hat{\mathbf{y}}_2(\mathbf{x}_{\text{novel}}),..., \hat{\mathbf{y}}_I(\mathbf{x}_{\text{novel}}), \text{d}(\mathbf{x}_{\text{novel}}^{\text{train}})_1, \text{d}(\mathbf{x}_{\text{novel}}^{\text{train}})_2, \ldots, \text{d}(\mathbf{x}_{\text{novel}}^{\text{train}})_I\} \tag{13}$$

Once $\overline{\mathbf{y}}$ has been determined, the error estimate can be found. Notice that the distance measure is now between the entire training set and the novel input for each of the novel exemplars.

$$|\bar{\mathbf{u}}_{\mathrm{novel}}| = \overline{F}_{L_1}\{\mathbf{x}_{\mathrm{novel}}, \hat{\mathbf{y}}_1(\mathbf{x}_{\mathrm{novel}}), \hat{\mathbf{y}}_2(\mathbf{x}_{\mathrm{novel}}), \ldots, \hat{\mathbf{y}}_I(\mathbf{x}_{\mathrm{novel}}), \bar{\mathbf{y}}_{\mathrm{novel}}, \mathrm{d}(\mathbf{x}_{\mathrm{novel}}^{\mathrm{train}})_1, \mathrm{d}(\mathbf{x}_{\mathrm{novel}}^{\mathrm{train}})_2, \ldots, \mathrm{d}(\mathbf{x}_{\mathrm{novel}}^{\mathrm{train}})_I\}$$

(14)

If we wish to report the error as a standard deviation the we multiply $|\bar{\mathbf{u}}_{\mathrm{novel}}|$ by a constant, k, such that

$$k = \frac{\sigma\{\mathbf{u}(t)\}}{E[|\mathbf{u}(t)|]}$$

(15)

over the training set. Where $\sigma\{\mathbf{u}(t)\}$ is the standard deviation of the actual error and $E[\mathbf{u}(t)]$ is the expected value of the magnitude of the actual error, both of which are calculated over the training set. Thus, for each pattern in the testing set we report $\mathbf{y}(t) = \bar{\mathbf{y}}(t) \pm 2\sigma_{\bar{\mathbf{y}}}$ such that;

$$\bar{\mathbf{y}}(t) - 2k|\bar{\mathbf{u}}(t)| < \mathbf{y}(t) < \bar{\mathbf{y}}(t) + 2k|\bar{\mathbf{u}}(t)|$$

(16)

to 95% confidence assuming, optimistically, that the errors are normally distributed.

DEMONSTRATION EXAMPLE: THE MACKEY GLASS EQUATION

A problem that has received a lot of attention lately is the Mackey – Glass chaotic time series [Mackey and Glass, 1977]. The equation is as follows;

$$x(t+1) = (1-b)x(t) + \frac{ax(t-\tau)}{1+\{x(t-\tau)\}^{10}}$$

(17)

where a, b , and τ are set to 0.2, 0.2, and 17 respectively. The initial condition is taken to be *x*(0)=0.5. The objective is to predict *x*(*t*+85) given *x*(0), *x*(*t*-6), *x*(*t*-12, and *x*(*t*-18) [Schmitz and

Aldrich, 1999]. Iterations *t* = 4001 to 4500 are used for the training set and patterns 4501 to 5000 are used for the test set. In this paper, however, we propose to model x(t+85) given x(t), x(t-1), x(t-2), …, x(t-29) where only four inputs are used from those available.

Three Separate Models

Three separate models were created for the first phase of this first example demonstration. Once the data was normalized, two different ANNs and a GRNN were developed for each of our three separate modeling trials. Each trail used one of the three different input subspaces supplied by our ranking methods. First, the GRNN technique was developed with input subspace {*t*, *t*-6, *t*-13, *t*-19} as inputs. Then a multilayer feedforward perceptron (MLP) neural network was used with one hidden layer for the first ANN trial {*t*, *t*-12, *t*-16, *t*-24} and then a second ANN was developed with two hidden layers {*t-3*, *t*-11, *t*-15, *t*-19}. The test set performance results of each of the models is shown in Table 1 below.

Three Model Consolidation "Poor - Man's Stacking"

The three models developed in the section above were then combined using the poor – man's stacking approach as described above. The test set performance results of this combined model are shown in Table 2 below.

Modified Series Association

As a final demonstration on this chaotic time series system, an MSA model was developed. Three Level 0 models were used – a GRNN and two MLP ANNs using the same input subspaces as above. Five fold cross validation was also used at Level 0. The graph in Figure 1 shows the test set true, predicted, predicted plus 2σ, and predicted minus 2σ results. Credible error bounds can be clearly seen in the figure. Not only does MSA provide consolidated predictions, but it also provides credible error bounds on its results. The test set performance results of the MSA model are shown in Table 3 below.

CONCLUSIONS

In the past, general, non-parametric, data driven methods, such as ANNs, could not be used for modeling with a model error prediction objective. Modified Series Association provides the required reliably estimated error bounds as well as improved results though diverse model combination and consolidation. MSA can provide both an accurate model of the desired output and an estimation of the error bounds on the predicted output in an automated and user-friendly manner. The resulting models are both accurate and precise. MSA generalizes well. MSA also takes advantage of the results of our ranking methods to preprocess and partition the available data.

REFERENCES

Bartlett, E. B. (1994). “Dynamic node architecture learning: an information theoretic approach”, *Neural Networks 7,* 129-140.

Bartlett, E. B. and K. Kim, (1993). “Error Bounds on the Output of Artificial Neural Networks,” *ANS Trans.*, 69,197-199.

Bartlett, E. B. and R. E. Uhrig (1992). “Nuclear power plant status diagnostics using an artificial neural network,” *Nuclear Technology* 97, 272-281.

Bhat, N., and T. McAvoy, (1990). “Use of Neural Nets for Dynamic Modeling And Control of Chemical Process Systems,” *Computers Chem. Engng.* 14, 573-583.

Blum, E. K. and L. K. Li, (1991). “Approximation Theory and Feedforward Networks,” *Neural Networks*, 4, 511.

Cybenko, G. (1989). “Approximation by superposition of a sigmoidal function,” *Mathematics of Control, Signals, and Systems 2,* 303-314.

Haykin, S., (1999). *Neural Networks: A Comprehensive Foundation.* Prentice – Hall, Upper Saddle River, New Jersey.

Hecht-Nielsen, R. (1990). *Neurocomputing.* Addison-Wesley, Reading, Mass.

Kim, K. and E. B. Bartlett, (1996). “Nuclear Power Plant Fault Diagnosis Using Neural Networks with Error Estimation by Series Association,” IEEE Tranactions on Nuclear Engineering Vol 43, No 4, pp 2373 - 2388

Kurkova, V. (1992). “Kolmogorov's theorem and multilayer neural networks,” *Neural Networks*. 5,501-506

Lapades, A., and R. Farber, (1987). *Nonlinear signal processing using neural networks: prediction and system modeling.* Los Alamos National Laboratory Technical Report LA-UR-87-2662.

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

Mackey, M. C. and L. Glass (1977). “Oscillations and Chaos in Physiological Control Systems,” *Science*, 197, 287-289.

Miller, W. T., R. S. Sutton, and P. Werbos (Eds.), (1990). *Neural Networks for Control.* Press. Cambridge, Mass.

Narendra, K. S. and K. Parthasarathy, (1990). “Identification and Control of Dynamic Systems Using Neural Networks,” *IEEE Trans. Neural Networks*, 1, no. 1, 4.

Parzen , E., (1962). “On estimation of a probability density function and mode,” *Ann. Math.* fist. 33, 065-1076.

Rumelhart, D. E., J. L. McClelland, and the PDP Research Group, (1986). *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, Vol. 1 & 2, MIT Press, Cambridge, Massachusetts.

Schmitz, G. P. J. and C. Aldrich (1999). “Combinatorial Evolution of Regression nodes in Feedforward Neural Networks,” *Neural Networks*, 12, 175-189.

Specht, D. F., (1990). “A General Regression Neural Network,” *I&E Transactions on Neural Networks*, 2, 568.

Sridhar, D. V., (1996). “Process Modeling Using Stacked Neural Networks,” *Ph.D. Dissertation, Iowa State University, Ames, Iowa.*

Upadhyaya, B. R., and E. Eryurek, (1992). “Application of neural networks for sensor validation and plant monitoring,” *Nuclear Technology* 97, 170-176.

Uhrig, R. E. (1989). “Use of neural networks in nuclear power plant diagnostics,” *Proc. Int. Conf on Availability Improvements in Nuclear Power Plant*, 310-315 Madrid, Spain.

Venkatasubramanian, V., and K. Chan, (1989). “A neural network methodology for process methods,” *Journal of Applied Meteorology* 31, 405-420.

Wichmann, N. L. and E. B. Bartlett (1997), “Ranking Input Variables using General Regression Neural Networks”, *Intelligent Engineering Systems Through Artificial Neural Networks: Volume 7*, pp. 959 – 964.

Wichmann, N. L., (1997). “Variable Importance Ordering for Evapotranspiration using General Regression Neural Networks”, *M.S. Thesis, Iowa State University, Ames Iowa.*

Wolpert, D. H., (1990). “A Mathematical Theory of Generalization: Part I and Part II,” *Complex Systems*, 4, 151.

Wolpert, D. H., (1992). “Stacked Generalization”, *Neural Networks*, 5, 241.

Zhang, X., J. P. Mesirov, and D. L. Waltz, (1992). “Hybrid system for prediction for protein secondary structure prediction,” *Journal of Molecular Biology*, 225, 1049-1063.

Table 1

Test set performance results of three separate models

	Training Set R^2 : σ_{error}	Test Set R^2 : σ_{error}
GRNN $\hat{\mathbf{y}}(t+85)$	0.9738 : 0.0420	0.900 : 0.072
ANN 1 $\hat{\mathbf{y}}(t+85)$	0.9782 : 0.0347	0.947 : 0.054
ANN 2 $\hat{\mathbf{y}}(t+85)$	0.9909 : 0.0225	0.976 : 0.037

Table 2

Test set performance results of the poor - man’s stacking model

	Training Set R^2 : σ_{error}	Test Set R^2 : σ_{error}
Poor – Man’s Stacking $\hat{\mathbf{y}}(t+85)$	0.989 : 0.025	0.972 : 0.038

Table 3

Test set performance results of the Modified Series Association model

	Training Set R^2 : σ_{error}	Test Set R^2 : σ_{error}
Modified Series Association $\bar{\mathbf{y}}(t+85)$	0.991 : 0.022	0.981 : 0.031

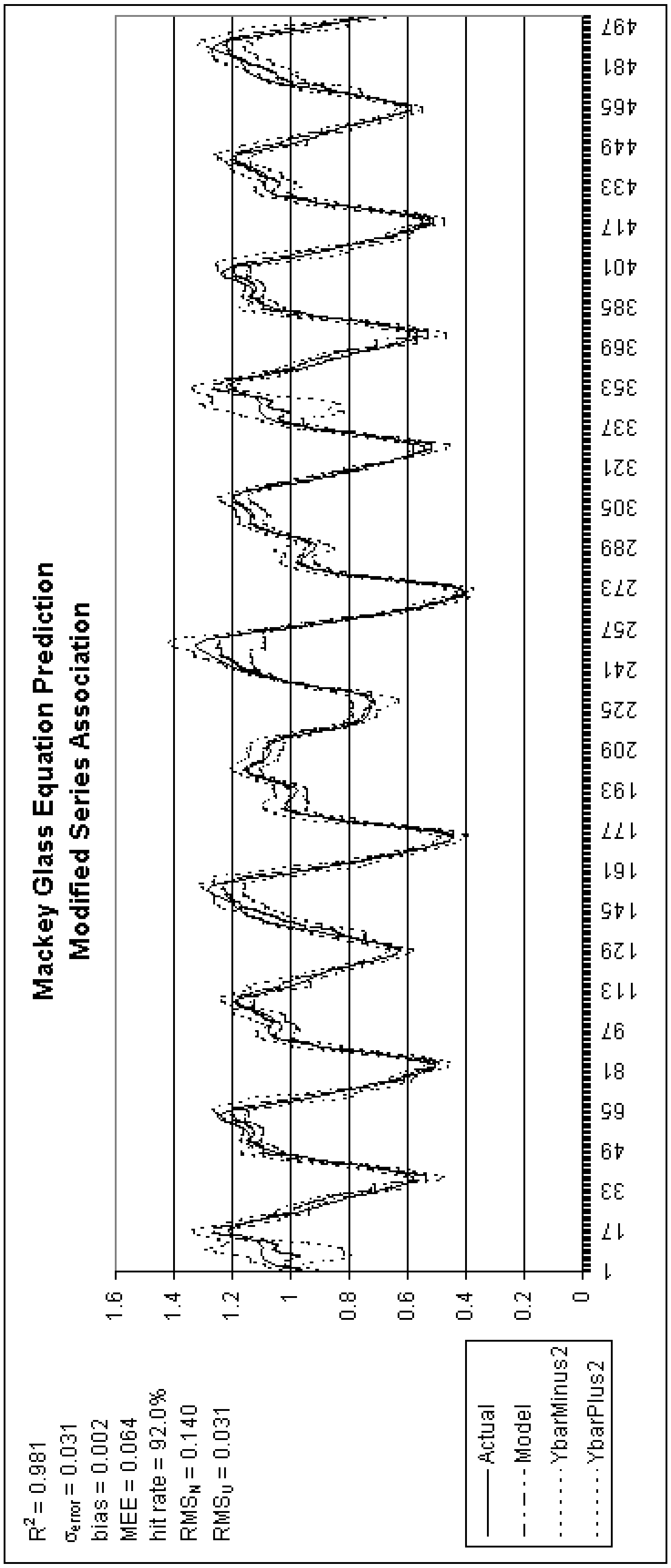
Mackey Glass Equation Prediction
Modified Series Association
$R^2 = 0.981$
$\sigma_{error} = 0.031$
bias = 0.002
MEE = 0.064
hit rate = 92.0%
$RMS_N = 0.140$
$RMS_U = 0.031$
Actual
Model
YbarMinus2
YbarPlus2
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
17
33
49
65
81
97
113
129
145
161
177
193
209
225
241
257
273
289
305
321
337
353
369
385
401
417
433
449
465
481
497

Figure 1