

Neural Network Prediction with Noisy Predictors.

A. Adam Ding

Department of Mathematics, Northeastern University, Boston, MA 02115.

abstract

Very often the input variables for neural network predictions contain measurement errors. In particular, this may happen because the original input variables are often not available at the time of prediction and have to be replaced by predicted values themselves. This issue is usually ignored and results in non-optimal predictions. This paper shows that under some general conditions, the optimal prediction using noisy input variables can be represented by a neural network with the same structure and the same weights as the optimal prediction using exact input variables. Only the activation functions have to be adjusted. Therefore we can achieve optimal prediction without costly retraining of the neural network. We explicitly provide an exact formula for adjusting the activation functions in a logistic network with Gaussian measurement errors in input variables. This approach is illustrated by an application to short term load forecasting.

KEY WORDS: Measurement Error Model; Short Term Load Forecasting (STLF).

The author thanks Dr. Milan Casey Brace of Puget Sound Power & Light Company in Washington for providing the load forecasting data which was analyzed and reported in Section 3.2. I am grateful for comments from an associate editor and three referees that led to a better presentation of this paper. Also, I want to thank Professor Samuel Gutmann and Dr. Pete Chvany for reading the manuscript.

1 INTRODUCTION

Neural networks have been used as a prediction tool in many areas: stock price prediction, weather forecasting, utility load forecasting, etc. [1-13] Generally neural network prediction proceeds as following:

(i) Relate the response Y with some predictors \underline{X} by a neural network

$$g_{\underline{\theta}}(\underline{X});$$

(ii) Use the data set $(\underline{X}_1, Y_1), \dots, (\underline{X}_n, Y_n)$ to train the neural network $g_{\underline{\hat{\theta}}}(\underline{X})$

(i.e., to estimate the weights $\underline{\hat{\theta}}$);

(iii) Predict a future response Y_{n+1} by neural network response $g_{\underline{\hat{\theta}}}(\underline{X}_{n+1})$.

However, often \underline{X}_{n+1} is not available at the time of forecasting and has to be replaced by surrogate predictors \underline{W}_{n+1} . Therefore, in practice, we are actually using $g_{\underline{\hat{\theta}}}(\underline{W}_{n+1})$ to predict Y_{n+1} . This prediction is not optimal if we do not adjust for the replacement of \underline{X}_{n+1} by \underline{W}_{n+1} . This paper discusses how to adjust for this replacement.

To illustrate applications where this adjustment is needed, we consider the following examples.

Example 1. Short Term Load Forecasting (STLF).

It has been recognized that the electric power system load pattern depends heavily on weather variables [14, 15]. Neural network technology has

been proposed to relate the system load Y with current and past load variables and temperature variables X , and to predict a future system load (Y_{n+1}) for hours or days ahead [5-13]. However, we can not observe the temperature variables \underline{X}_{n+1} for hours or days ahead. Therefore, they are usually replaced by the weather reporter's forecast of temperature variables (\underline{W}_{n+1}).

Example 2. Stock Price prediction.

Neural networks can be used to model the relationship between a stock's price Y and predictors \underline{X} such as the company's earnings, cash flow, growth rate and its recent stock price. However, to predict the stock price next year (Y_{n+1}), we need to know its earnings, cash flow and growth rate next year (\underline{X}_{n+1}). Obviously, these variables have to be replaced by expert predictions (\underline{W}_{n+1}).

In this paper, we shall study this problem using the statistical measurement error model setup [16]. That is, we consider the surrogate predictors \underline{W} as measurements of \underline{X} with random errors. However, the goal here is in a sense the "reverse" of that of the usual statistical measurement error model. Usually, for a measurement error model, we estimate the parameters $\underline{\theta}$ using a data set which consists of observations on the surrogate predictors \underline{W} and the response Y . The goal is to adjust the estimates for the true relationship between the predictors \underline{X} and the response Y . Here, we have already ob-

tained the best estimate of the true relationship between the predictors \underline{X} and the response Y by directly training on a data set of observations on \underline{X} and Y . The goal here is to find the best prediction of Y conditional on \underline{W} .

An expert on the measurement error model may be quick to point out a simple method of getting the optimal prediction of response Y conditional on the surrogate predictors \underline{W} : train the network on observations of Y and \underline{W} (instead of \underline{X}). However, while variables such as temperatures and earnings (\underline{X}) are well documented, data on the weather reporter’s forecast of temperatures and the Wall Street expert’s forecast of future earnings (\underline{W}) are generally not well kept. To train a neural network directly using these surrogate predictors (\underline{W}) means that we have to discard most data and use a much smaller data set. Hence we need the “reverse” measurement error model setup. The reverse model setup leads to some interesting implications which will be stated in Proposition 3 later.

In Section 2, we show that the optimal neural network prediction for Y based on the noisy predictors \underline{W} can be represented by a three-layer neural network with the same structure and weights as the optimal neural network prediction based on the original predictors \underline{X} . The only difference is in the activation functions of the hidden nodes. We also derive an explicit formula to adjust the activation functions of the logistic network assuming Gaussian

noise ($\underline{W} - \underline{X}$). This provides a simple solution when some original predictors are replaced by their surrogates \underline{W} : use the already trained neural network and adjust the activation functions according to the provided formula.

A numerical simulation in Section 3.1 shows that the proposed approach works well and is indeed optimal. In Section 3.2 we illustrate the proposed methodology using an STLF application on a real data set from Puget Sound Power & Light Company.

2 OPTIMAL NEURAL NETWORK PREDICTION USING SURROGATE PREDICTORS

We shall use a three-layer feedforward neural network whose structure is shown in Figure 1. The neural network consists of three layers of neurons: the input layer, the hidden layer and the output layer. The j th hidden neuron receives a signal that is a linear combination of the input variables x_1, x_2, \dots, x_d with respect to weights $\underline{\beta}_j = (\beta_{j1}, \dots, \beta_{jd})^t$. Here and throughout the paper, the superscript t denotes the transpose. The signal is then processed by an activation function $f_j(\cdot)$ to yield an output signal. The output signals of the hidden neurons are then linearly combined with respect to weights $(\alpha_0, \dots, \alpha_k)$ to give an output y for the output layer. Hence the output of

such a network is represented by

$$g_{\underline{\theta}}(\underline{x}) = \alpha_0 + \sum_{j=1}^k \alpha_j f_j(\beta_{j0} + \underline{\beta}_j^t \underline{x}). \quad (1)$$

where $\underline{\theta} = (\alpha_0, \dots, \alpha_k, \beta_{10}, \dots, \beta_{k0}, \underline{\beta}_1^t, \dots, \underline{\beta}_k^t)$ are the weights or the unknown parameters. The f_1, f_2, \dots, f_k are called the activation functions of the hidden nodes. This type of neural network is widely used and capable of approximating most functions well [17, 18]. For a review of statistical neural network theory, see Cheng and Titterington [19].

Place of Figure 1

Since not all of the input variables x_1, \dots, x_d are observable at the time of forecast, some of them will need to be replaced by surrogate variables. Following the notations in [16], we denote the unobservable predictors as $\underline{X} = (x_1, \dots, x_{d_1})^t$ and the observable predictors as $\underline{Z} = (z_1, \dots, z_{d_2})^t$, where $d_1 + d_2 = d$. And the neural network function with input variables \underline{X} and \underline{Z} is rewritten as

$$g_{\underline{\theta}}(\underline{X}, \underline{Z}) = \alpha_0 + \sum_{j=1}^k \alpha_j f_j(\beta_{j0} + \underline{\beta}_j^t \underline{X} + \underline{\gamma}_j^t \underline{Z}). \quad (2)$$

where $\underline{\theta} = (\alpha_0, \dots, \alpha_k, \beta_{10}, \dots, \beta_{k0}, \underline{\beta}_1^t, \dots, \underline{\beta}_k^t, \underline{\gamma}_1^t, \dots, \underline{\gamma}_k^t)$.

We assume that the response variable Y and the predictors \underline{X} and \underline{Z} are

related by a neural network model

$$Y = g_{\theta}(\underline{X}, \underline{Z}) + \varepsilon, \quad (3)$$

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

Further, we assume that the surrogate variables \underline{W} are related to \underline{X} by

$$\underline{W} = \underline{X} + \underline{U} \quad (4)$$

where \underline{U} denotes the random measurement error. We use $d\mu(\underline{U})$ to denote the density for the distribution of \underline{U} . Without loss of generality, the mean of \underline{U} is assumed to be a zero vector $\underline{0}$. (In practice, we adjust \underline{W} first to be an unbiased estimator for \underline{X} as proposed below.)

Before stating our main results, we first introduce some neural network terminologies. The activation functions for hidden nodes, f_j , are usually chosen to be nondecreasing symmetric sigmoidal functions. A function is called sigmoidal if

$$\begin{cases} f(x) \rightarrow 0, & \text{as } x \rightarrow -\infty \\ f(x) \rightarrow 1, & \text{as } x \rightarrow +\infty \end{cases} \quad (5)$$

and is called symmetric sigmoidal if it also satisfies

$$f(x) + f(-x) = 1. \quad (6)$$

The most popular choice of activation function is the logistic function:

$$f(x) = \frac{1}{1 + e^{-x}}.$$

We shall call a three-layer neural network a sigmoidal (or logistic or symmetric sigmoidal) network if the activation functions of all hidden nodes are sigmoidal (or logistic or symmetric sigmoidal) functions.

Now we are ready to state the main result of this paper.

Theorem 1 *Suppose that the response variable Y , predictors \underline{X} , \underline{Z} and surrogate predictors \underline{W} satisfy the model (2), (3) and (4).*

(a) *Under the square loss function, the best prediction for Y given observations \underline{W} and \underline{Z} is represented by a three-layer neural network $\bar{g}_\theta(\underline{W}, \underline{Z})$ that has the same structure and weights (parameters) as $g_\theta(\underline{X}, \underline{Z})$, and only differs in the activation functions. That is, a neural network*

$$\bar{g}_\theta(\underline{W}, \underline{Z}) = \alpha_0 + \sum_{j=1}^k \alpha_j \bar{f}_j(\beta_{j0} + \underline{\beta}_j^t \underline{W} + \underline{\gamma}_j^t \underline{Z}) \quad (7)$$

achieves the smallest mean squared error for predicting Y . Here

$$\bar{f}_j(x) = \int f_j(x - \underline{\beta}_j^t \underline{U}) d\mu(\underline{U}). \quad (8)$$

Furthermore, if $g_\theta(\underline{X}, \underline{Z})$ is a (nondecreasing) sigmoidal network, then $\bar{g}_\theta(\underline{W}, \underline{Z})$ is also a (nondecreasing) sigmoidal network.

(b) *If \underline{U} is distributed symmetrically about $\underline{0}$, then $g_\theta(\underline{X}, \underline{Z})$ being a symmetric sigmoidal network implies that $\bar{g}_\theta(\underline{W}, \underline{Z})$ is also a symmetric sigmoidal network.*

The proof is included in the Appendix.

Because of Theorem 1, we can adjust our prediction for surrogate predictors \underline{W} according to formulas (7) and (8) without retraining the neural network. In practice, usually $d\mu(\underline{U})$ is unknown. However, we generally have a validation data set containing observations $(\underline{x}_j, \underline{w}_j)$, $j = 1, 2, \dots, m$ that comes from model (4). Hence theoretically we can estimate $d\mu(\underline{U})$ using statistical density estimation methods [20] and then plug into (8).

Typically we use back-propagation learning with a logistic network $g_{\underline{\theta}}(\underline{X}, \underline{Z})$ for forecasting. Also the measurement errors \underline{U} are often assumed to be normally distributed (i.e., Gaussian noise). In this case, the formula for adjustment can be simplified in the next Proposition. (See the Appendix for proof.)

Proposition 2 *Assume that model (2), (3) and (4) hold with $g_{\underline{\theta}}(\underline{X}, \underline{Z})$ being a logistic network. Furthermore, suppose that the measurement error \underline{U} is distributed as $N(\underline{0}, \mathbf{B})$, where \mathbf{B} is the covariance matrix. Then the best prediction of Y given \underline{W} and \underline{Z} is*

$$\bar{g}_{\underline{\theta}}(\underline{W}, \underline{Z}) = \alpha_0 + \sum_{j=1}^k \alpha_j \bar{f}_j(\beta_{j0} + \underline{\beta}_j^t \underline{W} + \underline{\gamma}_j^t \underline{Z}), \quad (9)$$

where

$$\bar{f}_j(x) = \int_{t=-\infty}^{\infty} f_j(x - \sqrt{\underline{\beta}_j^t \mathbf{B} \underline{\beta}_j} t) \phi(t) dt. \quad (10)$$

Here $\phi(x) = \exp(-x^2/2)/\sqrt{2\pi}$ denotes the density of the standard normal distribution.

Next, we describe the proposed neural network prediction procedure. Generally, we have a sample of n observations $(y_i, \underline{x}_i, \underline{z}_i)$ generated from the model (3),

$$y_i = g_{\underline{\theta}}(\underline{x}_i, \underline{z}_i) + \varepsilon_i \quad i = 1, 2, \dots, n.$$

This data set is called the training set in neural network terminology. We should also have a validation data set containing observations $(\underline{x}_j, \underline{w}_j)$, $j = 1, 2, \dots, m$ that comes from model (4). The objective is to produce a prediction for a future response y_{n+1} based on \underline{w}_{n+1} and \underline{z}_{n+1} . The procedure is:

Step 1. Use the training set to train a network $g_{\underline{\hat{\theta}}}(\underline{x}_{n+1}, \underline{z}_{n+1})$.

Step 2. Use the validation set to estimate the bias of \underline{W} by

$$\bar{\underline{u}} = \frac{1}{m} \sum_{j=1}^m (\underline{w}_j - \underline{x}_j).$$

Replace w_{n+1} by the unbiased predictor $w_{n+1} - \bar{\underline{u}}$.

Step 3. Use the validation set to judge whether the $(\underline{w}_j - \underline{x}_j)$'s are normally distributed by statistical diagnostic methods such as a normal probability plot. If they can be reasonably assumed to be normally distributed and $g_{\underline{\theta}}$ is a logistic network then go to step 4. Otherwise, go to step 4*.

Step 4. Use the validation set to estimate covariance matrix \mathbf{B} by

$$\hat{\mathbf{B}} = \frac{1}{m} \sum_{j=1}^m (\underline{w}_j - \underline{x}_j - \bar{\underline{u}})(\underline{w}_j - \underline{x}_j - \bar{\underline{u}})^t.$$

Apply formulas (9) and (10) with $\hat{\mathbf{B}}$ to obtain $g_{\hat{\theta}}(\underline{w}_{n+1} - \bar{\underline{u}}, \underline{z}_{n+1})$.

Step 4*. Use the validation set to estimate the density $d\hat{\mu}(\underline{U})$ by nonparametric methods. Apply formulas (7) and (8) to obtain $g_{\hat{\theta}}(\underline{w}_{n+1} - \bar{\underline{u}}, \underline{z}_{n+1})$.

Remark 1: The above procedure is mainly recommended when the noise in the predictors is indeed Gaussian (hence Step 4 is followed). Since there is no need to retrain the neural network, we need only to follow the usual procedure in Step 1. Comparing with the training effort, the extra computations in Steps 2-4 are minimal. The noise-in-predictors problem is solved here with little extra computational effort.

Remark 2: Although theoretically we can follow Step 4* when the noise is non-Gaussian, its application involves practical difficulties. Generally the size of the validation subset is small in practice, making the nonparametric density estimation of $d\hat{\mu}(\underline{U})$ very inaccurate. The formulas (7) and (8) may involve integration of $d\hat{\mu}(\underline{U})$ over the tail region where no density estimation method can work well. Also, while Step 4 needs only a one-dimensional integration, Step 4* requires high-dimensional numerical integration which is

much more computationally demanding. Other methods should be explored in this case. One idea, to be presented in another paper in preparation, is to train the neural network with artificially introduced noises similar to [21].

Remark 3: As pointed out in the proof of Theorem 1, the optimal prediction using noisy predictors is the conditional mean of the response variable given the surrogate predictors. It is a particular property of the sigmoidal network that this conditional mean remains a sigmoidal network with the same weights and structure. For networks using other activation functions, we can still get the optimal prediction by computing this conditional mean. However, the result is generally no longer a network of the same type and weights. In some special cases, the optimal prediction may still remain a network of the same type. For example, if the response variable Y is related to the predictors \underline{X} by a radial basis function network (RBFN) $g_{\underline{\theta}}(\underline{X}) = \sum_{j=1}^k \alpha_j \exp[-(\underline{X} - \underline{\beta}_j)^t(\underline{X} - \underline{\beta}_j)/\sigma_j^2]$ [5], and the measurement errors in the surrogate predictors are Gaussian noise $N(\underline{0}, \sigma^2 I)$, then the optimal prediction remains a RBFN $g_{\underline{\theta}^*}(\underline{W}) = \sum_{j=1}^k \alpha_j^* \exp[-(\underline{W} - \underline{\beta}_j^*)^t(\underline{W} - \underline{\beta}_j^*)/(\sigma_j^*)^2]$, where

$$\alpha_j^* = \alpha_j \cdot \left(\frac{\sigma_j}{\sqrt{\sigma_j^2 + 2\sigma^2}} \right)^d, \quad \underline{\beta}_j^* = \underline{\beta}_j, \quad \sigma_j^* = \sqrt{\sigma_j^2 + 2\sigma^2}.$$

The proof of this fact is just straightforward calculation of the conditional mean and details are omitted here.

Remark 4: As mentioned in the introduction, models (3) and (4) are very similar to the measurement error models but the goals are reversed. In the usual measurement error models, the parameter estimations are done with data on Y and \underline{W} , and the object is to make inferences about the relationship between Y and \underline{X} . Here, we do the parameter estimation (neural network training) with data on Y and \underline{X} , and the object is to adjust for the relationship between Y and \underline{W} . This reversal of goals has some interesting implications. By Proposition 2, we can train a logistic network with data on Y and \underline{X} and then adjust for \underline{W} if the noise is Gaussian. However, if we assume the same Gaussian noise \underline{U} , then we should not train a logistic network with data on Y and \underline{W} .

Proposition 3 *Assume that models (2), (3) and (4) hold, and the measurement error \underline{U} is distributed as $N(\underline{0}, \mathbf{B})$. Also assume that the activation functions are nondecreasing sigmoidal functions as usual. Then there exists no neural network $g_{\theta}(\underline{X}, \underline{Z})$ such that the resulting best prediction of Y given \underline{W} is a logistic network $\bar{g}_{\theta}(\underline{W}, \underline{Z})$.*

The proof of Proposition 3 can be found in the Appendix.

3 Numerical Studies

3.1 A simulation study

To check the performance of the proposed procedure, a simulation was conducted. Training data sets of size $n = 500$ are generated according to (3)

$$y_i = g_{\underline{\theta}}(\underline{x}_i, \underline{z}_i) + \varepsilon_i \quad i = 1, 2, \dots, n.$$

Here a 3-node logistic net $g_{\underline{\theta}}$ is used with three input variables

$$g_{\underline{\theta}}(\underline{x}_i, \underline{z}_i) = f(0.5 + 2x_{1,i}) + f(x_{2,i} + z_{1,i}) + f(x_{2,i} - z_{1,i})$$

where $\underline{x}_i = (x_{1,i}, x_{2,i})^t$, $\underline{z}_i = z_{1,i}$, and f is the logistic function. The input variables $(x_{1,i}, x_{2,i}, z_{1,i})$ are generated according to a 3-dimensional standard normal distribution. The random errors ε_i are generated from a normal distribution with standard deviation 0.1. Thus the signal/noise ratio (ratio of the standard deviation of $g_{\underline{\theta}}(\underline{X}, \underline{Z})$ to the standard deviation of ε) is approximately 4.9.

Next, a 3-node logistic net $g_{\underline{\hat{\theta}}}$ was trained on the training data set and used to predict a future output $y_{n+1} = g_{\underline{\hat{\theta}}}(\underline{x}_{n+1}, \underline{z}_{n+1}) + \varepsilon_{n+1}$ generated from the same model as the training data. The prediction is based on the observation of some noisy predictors $\underline{w}_{n+1} = (w_{1,n+1}, w_{2,n+1})^t$ and \underline{z}_{n+1} . The noisy predictors $w_{1,n+1}$ and $w_{2,n+1}$ were generated using $x_{1,n+1}$ and $x_{2,n+1}$ respectively with measurement bias 0.1 and standard deviation 0.1. For the

proposed procedure, we estimated the measurement bias and variance from a similarly generated validation set of size $m = 30$,

$$\underline{w}_j = \underline{x}_j + \underline{u}_j, \quad j = 1, 2, \dots, m.$$

And each element in \underline{u}_j was generated according to a normal distribution with mean 0.1 and standard deviation 0.1.

The predictions \hat{y}_{n+1} were then compared with the real output y_{n+1} and the prediction errors are recorded. This process was repeated 1000 times, and the average squared prediction errors are reported in Table 1.

The simulation code was written in Splus language. The training of the neural network was done using the *nnet* function from Venables and Ripley [22] with a weight decay rate of 0.005 and a maximum of 1000 iterations.

Place of Table 1

The first line in Table 1 reports the error of the naive prediction by plugging in \underline{w}_{n+1} directly for \underline{x}_{n+1} . The second line reports the error achieved by the proposed procedure. We see a definite reduction of error from 0.0182 to 0.0137. To better understand the magnitude of this reduction, we also report the best prediction error achievable under various situations.

The prediction error comes from several sources. First, there is the mod-

eling error (variance of ε) that cannot be avoided in any procedure unless the model is changed. This equals the prediction error achieved assuming that we know the true underlying link between input and output and also observe the input variables. The modeling error is reported (0.01) in the last line of Table 1. The second part of the error is introduced by the inaccurate estimation of the weights $\underline{\theta}$, which decreases as the training data size increases. The sum of the modeling error and weight estimation error is reflected by the prediction of $g_{\underline{\theta}}(\underline{X}, \underline{Z})$ as reported (0.0115) in the second to last line of Table 1. The third part of error, which I call *noisy input error*, comes from the fact that we do not have accurate observed values of the input variables \underline{X} . The best prediction using the noisy predictors \underline{W} is $\bar{g}_{\underline{\theta}}(\underline{W}, \underline{Z})$. This prediction error reflects the sum of modeling error and noisy input error, as reported (0.0123) in the third line of Table 1. The three parts of the error discussed above are unavoidable for the prediction problem we considered here. The sum of the three parts of the error is approximately 0.0138 ($= 0.0115 + 0.0123 - 0.01$). We can see from the first row of Table 1 that the naive plug-in prediction does introduce much more error. On the other hand, with a validation data size of only $m = 30$, the proposed procedure seems to eliminate all errors not from these three sources.

3.2 An example of STLF application

In this section, we apply the proposed methods to an example of short-term power system load forecasting (STLF). It has been long recognized that accurate short term (up to 7 days) load forecasting can be used for great cost savings for electric utility corporations [23, 24]. It has been established that the variation of electricity demand is closely related to the change in temperature [14, 15]. The application of neural network technology to the STLF problem has also been explored extensively in the literature [5-13].

Here we shall use a data set provided by Puget Sound Power and Light Company to illustrate the prediction procedure proposed in previous sections. Puget Sound Power and Light Company had collected hourly temperature and system load data over a period of several years. According to practical considerations, the company wanted the following predictions. The forecasts are to be ready at 8 a.m. each working day, Monday through Friday, for each of the hours of the following day. That is, to forecast the system load from 16 hours to 40 hours ahead. Park *et. al.* [6] proposed a neural network approach for STLF and tested it on the data set. The forecasting achieved better prediction error than forecasting methods being used at that time by Puget Sound Power and Light Company. In 1990 and 1991, a competition was held to develop a model that generated hourly forecasts of system elec-

tricity load for the six month winter period. The Quantitative Economic Research Institute (QUERI) won the competition with a regression model [25]. The winning QUERI model was built with significant model building effort. Later, Connor *et. al.* [7] showed that using an input and output configuration similar to the QUERI model, a recurrent neural network model can reduce the prediction error further.

In both [7] and [25], a separate model was built for each hour. Also, system load on weekdays and weekends were modeled separately. As an example, we shall only concentrate on weekdays forecasting of the system load at 11 a.m. the following day. This means a prediction 27 hours ahead.

To keep the example simple, here we do not go through the extensive predictor selection process as in [7] and [25]. The predictors are chosen intuitively:

- $l8(0)$: the system load at 8 a.m.;
- $l11(-1)$: the system load at 11 a.m. the previous day;
- $l11(-6)$: the system load at 11 a.m. six days ago (i.e. one week before the following day);
- $t8(0)$: the temperature at 8 a.m.;
- $t11(1)$: the temperature at 11 a.m. the following day;

- $tm(1)$: the average morning (6 a.m. to 10 a.m.) temperature on the following day;
- $tw(0)$: the average 11 a.m. temperature for the past week (that is, the average of the temperatures at 11 a.m. for the past seven days).

A three-layer logistic network with five hidden nodes based on these predictors is used for forecasting.

Clearly the variables $t11(1)$ and $tm(1)$ are not available at 8 a.m., and need to be replaced by the weather reporter's predictions $\hat{t}11(1)$ and $\hat{t}m(1)$. In the notations of previous sections, $\underline{X} = (t11(1), tm(1))^t$, $\underline{W} = (\hat{t}11(1), \hat{t}m(1))^t$ and $\underline{Z} = (l8(0), l11(-1), l11(-6), t8(0), tw(0))^t$.

Puget Sound Power & Light Company provided hourly system load data and hourly temperature data from 1 a.m. 1/1/85 to midnight 10/12/92. Hourly weather reporter's forecasted temperatures from two winter periods, 10/1/90 to 4/1/91 and 10/1/91 to 3/31/92, were also recorded. We tested our procedure for the system load forecasting in the 91-92 winter period 10/1/91 to 3/31/92. To be realistic, the neural network was trained on the data available before 10/1/91, i.e., the weekdays data from 1/1/85 to 9/30/91.

To test the performance of the predictions, in Table 2 we report three measures that were used in [6, 7]. The first measure is the Mean Absolute

Percentage Error (MAPE). The percentage error is defined as

$$\text{percentage error} = \frac{|\text{actual load} - \text{forecasted load}|}{\text{actual load}} \times 100\%$$

Taking the average of this percentage error over all testing points yields the MAPE for the load forecasting. The other two measures are Mean-squared error (MSE) and Median of squared error (Median SE). At each testing point, the prediction error is defined as

$$r_i = \hat{y}_i - y_i.$$

The usual measure of prediction error performance is the average over the testing set, $MSE = (1/n) \sum_{i=1}^n r_i^2$. According to Theorem 1 and Proposition 2, our proposed method should minimize this measure. To discount the influence of outliers, we can also use an alternative measure, the sample median of $\{r_1^2, \dots, r_n^2\}$ (Median SE). To make the prediction performance more comparable to MAPE, these two measures are usually scaled by $Mean(y) = (1/n) \sum_{i=1}^n y_i$. Conventionally, $\sqrt{MSE}/Mean(y) \times 100\%$ is called the coefficient of variability (CV). Similarly, here we denote $\sqrt{Median SE}/Mean(y) \times 100\%$ by Median CV.

The first row of Table 2 contains MAPE, CV and Median CV of the neural network prediction $g_{\hat{\theta}}(\underline{X}, \underline{Z})$. This reflects the prediction error level achieved by this neural network model. The second row contains the result

for the prediction $g_{\hat{\theta}}(\underline{W}, \underline{Z})$, which is commonly used in practice. We can see that this replacement of \underline{X} by its noisy estimation \underline{W} increases the prediction error.

Place of Table 2

Next, we applied the proposed procedure. Step 1 was already completed with the estimation of $\hat{\theta}$. For Step 2 we estimated the bias of the weather reporter's forecasted temperatures. Since this estimation had to be done before 10/1/91, we used only the data prior to that time for this estimation. From data for the winter period from 10/1/90 to 4/1/91, the biases of the forecasted temperatures $\hat{t}_{11}(1)$ and $\hat{t}_m(1)$ were estimated to be

$$\underline{\bar{\mu}} = \begin{pmatrix} -0.4615 \\ -0.1165 \end{pmatrix}$$

Hence we adjusted $\underline{W} = (\hat{t}_{11}(1) + 0.4615, \hat{t}_m(1) + 0.1165)^t$. In Step 3, we checked the normality assumptions using the normal probability plots for $\hat{t}_{11}(1)$ and $\hat{t}_m(1)$, which are shown in Figure 2. Since the plots were very close to straight lines, it is reasonable to assume that $\hat{t}_{11}(1)$ and $\hat{t}_m(1)$ are normally distributed. (The plots may look like step functions because of the discrete measurement unit.)

Place of Figure 2

Therefore, we used Step 4. The covariance matrix of \underline{W} estimated from data of the 1990-91 winter period is

$$\widehat{\mathbf{B}} = \begin{pmatrix} 11.696 & 6.912 \\ 6.912 & 8.433 \end{pmatrix}$$

Then we applied formula (10) with the above estimated \mathbf{B} . The prediction error of this adjusted load forecasting $\bar{g}_{\hat{\theta}}(\underline{W}, \underline{Z})$ is reported in the third row of Table 2.

To serve as a benchmark, the prediction error levels for the winner QUERI model are reported in the fourth row of Table 2.

From the first row to the second row of Table 2, we see that the replacement of \underline{X} by its noisy estimators \underline{W} increases the prediction error. However, not all the increases in the prediction error are intrinsically due to the usage of noisy predictors. Much of the increased prediction error can be recovered (as reported in the third row) by using the simple adjustment formulas provided in this paper.

The neural network model here achieved prediction error levels close to the QUERI model. As mentioned earlier, the proposed procedure is only adjusting the prediction for the use of noisy surrogate predictors. Hence by itself it won't be able to improve the prediction beyond the neural network

prediction obtainable knowing the true predictors as reported the first row of Table 2. If we want to beat the QUERI model, then we need to put in extra modeling effort for the neural network model [7]. The principles of the adjustment proposed in this paper, however, should apply to more complicated neural network procedures.

4 Conclusions and Discussions

In this paper, we studied the prediction using neural networks when there is noise in predictors. It is shown that for a sigmoidal network the optimal prediction using noisy surrogate predictors is also a sigmoidal network with the same weights and structure. Based on this fact, we can adjust our prediction for the noise in predictors using the network previously trained on noiseless predictors. We proposed a procedure for this adjustment that is easy to implement and involves little extra computational effort when the noise is Gaussian. Simulation studies show that the proposed procedure does work.

The procedure is intended to correct only for the noise in the predictors, and should not be confused with an effort to build a more accurate neural network prediction model. Rather, this procedure should be automatically applied after a neural network is built and trained traditionally, improving the prediction with little extra computational effort.

Further research is needed for the case of non-Gaussian noise. Although the proposed procedure also applies to the non-Gaussian noise, a better method is needed for practical implementation.

5 Appendix

Proof of Theorem 1.

(a). For any prediction $\hat{Y}(\underline{W}, \underline{Z})$, the risk function is

$$\begin{aligned}
& E[(\hat{Y}(\underline{W}, \underline{Z}) - Y)^2 | \underline{W}, \underline{Z}] \\
&= [\hat{Y}(\underline{W}, \underline{Z}) - E(Y | \underline{W}, \underline{Z})]^2 + E[(E(Y | \underline{W}, \underline{Z}) - Y)^2 | \underline{W}, \underline{Z}] \\
&\quad + 2[\hat{Y}(\underline{W}, \underline{Z}) - E(Y | \underline{W}, \underline{Z})]E[(E(Y | \underline{W}, \underline{Z}) - Y) | \underline{W}, \underline{Z}] \\
&= [\hat{Y}(\underline{W}, \underline{Z}) - E(Y | \underline{W}, \underline{Z})]^2 + \text{Var}(Y | \underline{W}, \underline{Z}).
\end{aligned}$$

Hence $E(Y | \underline{W}, \underline{Z})$ is the best prediction, and achieves the minimum risk $\text{Var}(Y | \underline{W}, \underline{Z})$. We now only need to show that $E(Y | \underline{W}, \underline{Z})$ is a three-layer neural network $\bar{g}_\theta(\underline{W}, \underline{Z})$.

Direct calculation yields

$$\begin{aligned}
& E(Y | \underline{W}, \underline{Z}) \\
&= E(g_\theta(\underline{X}, \underline{Z}) | \underline{W}, \underline{Z}) \\
&= \alpha_0 + \sum_{j=1}^k \alpha_j E[f_j(\beta_{j0} + \underline{\beta}_j^t \underline{X} + \underline{\gamma}_j^t \underline{Z}) | \underline{W}, \underline{Z}] \\
&= \alpha_0 + \sum_{j=1}^k \alpha_j E[f_j(\beta_{j0} + \underline{\beta}_j^t \underline{W} + \underline{\gamma}_j^t \underline{Z} - \underline{\beta}_j^t \underline{U}) | \underline{W}, \underline{Z}] \\
&= \alpha_0 + \sum_{j=1}^k \alpha_j \bar{f}_j(\beta_{j0} + \underline{\beta}_j^t \underline{W} + \underline{\gamma}_j^t \underline{Z}),
\end{aligned}$$

with $\bar{f}_j(x) = \int f_j(x - \underline{\beta}_j^t \underline{U}) d\mu(\underline{U})$.

Let $p(v)$ denote the density for the induced distribution of $v = \underline{\beta}_j^t \underline{U}$. Then

$$\bar{f}_j(x) = \int_{v=-\infty}^{\infty} f_j(x - v) p(v) dv.$$

If f_j is sigmoidal, then the Lebesgue Convergence Theorem implies that as

$$\begin{aligned}
 x \rightarrow \infty \quad \bar{f}_j(x) &= \int_{v=-\infty}^{\infty} f_j(x-v)p(v)dv \\
 &\rightarrow \int_{v=-\infty}^{\infty} 1 \cdot p(v)dv \\
 &= 1,
 \end{aligned}$$

and similarly as $x \rightarrow -\infty$, $\bar{f}_j(x) \rightarrow 0$. Hence \bar{f}_j is sigmoidal too.

If f_j is nondecreasing, then for any $x > y$,

$$\begin{aligned}
 \bar{f}_j(x) - \bar{f}_j(y) &= \int_{v=-\infty}^{\infty} [f_j(x-v) - f_j(y-v)]p(v)dv \\
 &\geq \int_{v=-\infty}^{\infty} 0 \cdot p(v)dv \\
 &= 0,
 \end{aligned}$$

i.e. \bar{f}_j is also nondecreasing.

(b) Since \underline{U} is distributed symmetrically about $\underline{0}$, $v = \underline{\beta}_j^t \underline{U}$ is distributed symmetrically about 0 for any $\underline{\beta}_j^t$. Hence $p(v) = p(-v)$. Therefore,

$$\begin{aligned}
 \bar{f}_j(x) &= \int_{v=-\infty}^{\infty} f_j(x-v)p(v)dv \\
 &= \int_{v=-\infty}^{\infty} f_j(x-v)p(-v)dv \\
 &= \int_{t=-\infty}^{\infty} f_j(x+t)p(t)dt \quad (t = -v) \\
 &= \int_{v=-\infty}^{\infty} f_j(x+v)p(v)dv.
 \end{aligned}$$

Since $f_j(x) + f_j(-x) = 1$,

$$\begin{aligned}
 &\bar{f}_j(x) + \bar{f}_j(-x) \\
 &= \int_{v=-\infty}^{\infty} f_j(x+v)p(v)dv + \int_{v=-\infty}^{\infty} f_j(-x-v)p(v)dv \\
 &= \int_{v=-\infty}^{\infty} [f_j(x+v) + f_j(-x-v)]p(v)dv \\
 &= \int_{v=-\infty}^{\infty} 1 \cdot p(v)dv \\
 &= 1.
 \end{aligned}$$

This completes the proof.

Proof of Proposition 2.

Since \underline{U} is distributed as $N(\underline{0}, \mathbf{B})$, $v = \underline{\beta}_j^t \underline{U}$ is normally distributed with

mean 0 and variance $\underline{\beta}_j^t \mathbf{B} \underline{\beta}_j$. Hence from Theorem 1,

$$\begin{aligned} \bar{f}_j(x) &= \int_{v=-\infty}^{\infty} f_j(x-v) \exp\left(-\frac{v^2}{2\underline{\beta}_j^t \mathbf{B} \underline{\beta}_j}\right) / \sqrt{2\pi \underline{\beta}_j^t \mathbf{B} \underline{\beta}_j} dv \\ (t = v / \sqrt{\underline{\beta}_j^t \mathbf{B} \underline{\beta}_j}) &= \int_{t=-\infty}^{\infty} f_j(x - \sqrt{\underline{\beta}_j^t \mathbf{B} \underline{\beta}_j} t) \exp(-t^2/2) / \sqrt{2\pi} dt \\ &= \int_{t=-\infty}^{\infty} f_j(x - \sqrt{\underline{\beta}_j^t \mathbf{B} \underline{\beta}_j} t) \phi(t) dt. \end{aligned}$$

Proof of Proposition 3.

If there exists $g_{\underline{\theta}}(\underline{X}, \underline{Z})$ such that $\bar{g}_{\underline{\theta}}(\underline{W}, \underline{Z})$ is a logistic network, then by Theorem 1,

$$\bar{f}_j(x) = \int_{v=-\infty}^{\infty} f_j(x-v) dP(v) \quad (11)$$

must be the logistic function $f(x) = \frac{1}{1+e^{-x}}$. Here $P(v)$ denotes the cumulative distribution function (CDF) of the distribution for $v = \underline{\beta}_j^t \underline{U}$.

Notice that a nondecreasing sigmoidal function can always be considered as a CDF for a random variable. Therefore we can study the corresponding characteristic functions. The characteristic function of a function $f(x)$ is defined as $\tilde{f}(t) = \int_{x=-\infty}^{\infty} e^{itx} f(x) dx$. Here $f'(x)$ denotes the derivative of $f(x)$.

By the Convolution Theorem (p37, Lukacs 1970 [26]), (11) implies that

$$\tilde{f}(t) = \tilde{f}_j(t) \cdot \tilde{P}(t).$$

For logistic $f(x)$ we have

$$\tilde{f}(t) = \frac{2\pi t}{e^{\pi t} - e^{-\pi t}}.$$

Since $P(v)$ is the CDF of the distribution of $v = \underline{\beta}_j^t \underline{U}$, i.e. $N(0, \underline{\beta}_j^t \mathbf{B} \underline{\beta}_j)$,

$$\tilde{P}(t) = \exp\left(-\frac{\underline{\beta}_j^t \mathbf{B} \underline{\beta}_j}{2} t^2\right).$$

Together, the last three expressions imply that

$$\tilde{f}_j(t) = \frac{2\pi t}{e^{\pi t} - e^{-\pi t}} \exp\left(\frac{\underline{\beta}_j^t \mathbf{B} \underline{\beta}_j}{2} t^2\right).$$

This last expression, however, implies that $\tilde{f}_j(t)$ is unbounded as $t \rightarrow \infty$.

This contradicts the fact that $\tilde{f}_j(t)$ is a characteristic function (p15, Lukacs 1970 [26]). Therefore the Proposition holds.

References

- [1] A.N. Refenes, A. Zapranis, G. Francis, "Stock Performance Modeling Using Neural Networks: A Comparative Study With Regression Models," *Neural networks*, vol. 7, pp. 375, 1994.
- [2] E. Schoneburg, "Stock price prediction using neural networks: A project report," *Neurocomputing*, vol. 2, pp. 17, 1990.
- [3] M. Malliaris and L. Salchenberger, "A Neural Network Model for Estimating Option Prices," *Journal of Applied Intelligence*, vol. 3, pp. 193-206, 1993.
- [4] R.J. Kuligowski, A.P. Barros, "Experiments in Short-Term Precipitation Forecasting Using Artificial Neural Networks," *Monthly weather review*, vol. 126, pp. 470, 1998.
- [5] D.K. Ranaweera, N.F. Hubele, and A.D. Papalexopoulos, "Application of Radial Basis Function Neural Network Model for Short-Term Load Forecasting," *IEE Proc.-Gener. Transm. Distrib.*, vol. 142, pp. 45-50, 1995.
- [6] D.C. Park, M.A. El-Sharkawi, R.J. Marks II, L.E. Atlas and M.J. Damborg, "Electric Load Forecasting Using An Artificial Neural Network," *IEEE Transactions on Power Systems*, vol. 6, pp. 442-449, 1991.

- [7] J.T. Connor, R.D. Martin, and L.E. Atlas, "Recurrent Neural Networks and Robust Time Series Prediction," *IEEE Transactions on Neural Networks*, vol. 5, pp. 240-254, 1994.
- [8] T.W.S. Chow, C.T. Leung, "Neural Network Based Short-Term Load Forecasting Using Weather Compensation," *IEEE transactions on power systems*, vol. 11, pp. 1736, 1996.
- [9] Jen-Lun Yuan and T.L. Fine, "Forecasting demand for electric power," *Advances in Neural Information Processing Systems*, vol. 5, S.J. Hanson, J.D. Cowan, and C.L. Giles, Eds. San Matteo, CA: Morgan Kauffman, pp. 739-746, 1993.
- [10] A.D. Papalexopoulos, S. Hao, and T.M. Peng, "An Implementation of a Neural Network Based Load Forecasting Model for the EMS," *IEEE Transactions on Power Systems*, vol. 9, pp. 1956-1962, 1994.
- [11] T.M. Peng, N.F. Hubele, and G.G. Karady, "Advancement in the Application of Neural Networks for Short-Term Load Forecasting," *IEEE Transactions on Power Systems*, vol. 7, pp. 250-257, 1992.
- [12] H.L. Willis and J.E.D. Northcote-Green, "Comparison Tests of Fourteen Distribution Load Forecasting Methods," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-103, pp. 1190-1197, 1984.

- [13] J.T.G. Hwang and A.A. Ding, "Prediction Intervals for Artificial Neural Networks," *Journal of the American Statistical Association*, vol. 92, pp. 748-757, 1997
- [14] G. Gross and F. Galiana, "Short-Term Load Forecasting," *Proceedings of IEEE*, vol. 75, pp. 1558-1573, 1987.
- [15] M. Segal, H. Shafir, M. Mandel, P. Alpert, and Y. Balmor, "Climatic-related Evaluation of the Summer Peak-hours electric load in Israel," *Journal of Applied Meteorology*, Vol. 31, pp. 1492-1498, 1992.
- [16] R.J. Carroll, D. Ruppert, and L.A. Stefanski, *Measurement Error in Nonlinear Models*, London: Chapman & Hall, 1995.
- [17] G. Cybenko, "Approximations by Superpositions of A Sigmoidal Function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303-314, 1989
- [18] H. White, "Connectionist Nonparametric Regression: Multilayer Feed-forward Networks Can Learn Arbitrary Mappings," *Neural Networks*, vol. 3, pp. 535-549, 1990.
- [19] B. Cheng and D.M. Titterton, "Neural Networks: A Review from a Statistical Perspective," *Statistical Science*, vol. 9, pp. 2-54, 1994.

- [20] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, London: Chapman & Hall, 1986.
- [21] L. Holmström and P. Koistinen, “Using Additive Noise in Back-Propagation Training,” *IEEE Transactions on Neural Networks*, vol. 3, pp. 24-38, 1992.
- [22] W.N. Venables and B.D. Ripley, *Modern Applied Statistics with S-PLUS, 2nd Ed.*, London: Springer, 1997
- [23] K.K. Hara, M. Kimurs and N. Honda, “A Method for Plneural networking Economic Unit Commitment and Maintenance of Thermal Power Systems,” *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-85, pp. 427-436, 1966.
- [24] C.K. Pang, G.B. Sheble, and F. Albuyeh, “Evaluation of Dynamic Programming Based Methods and Multiple Area Representation for Thermal Unit Commitments,” *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-100, pp. 1212-1218, 1981.
- [25] R. Engle III, F. Clive, W.J. Granger, R. Ramanathan, F. Vahid, and M. Werner, “Construction of the Puget Sound Forecasting Model,” Quantitative Economic Research Institute, San Diego, Ca., EPRI Project # RP919, 1992.

[26] E. Lukacs, *Characteristic Functions, 2nd Ed.*, London: Griffin, 1970

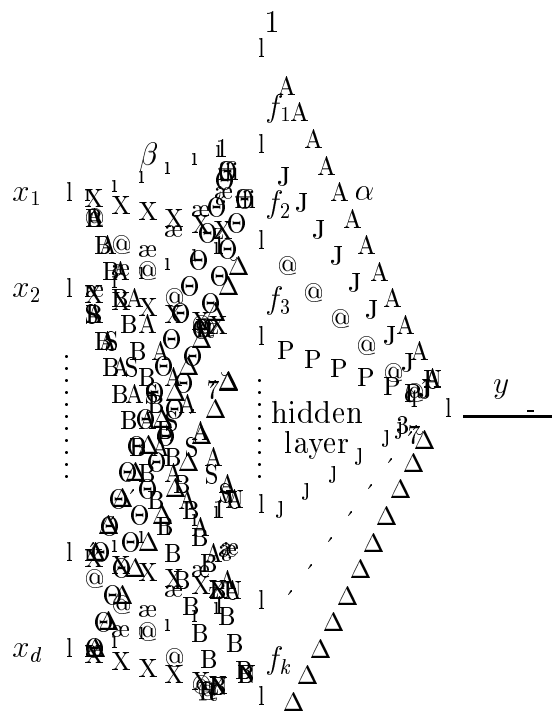


Figure 1: A feedforward neural network with one hidden layer

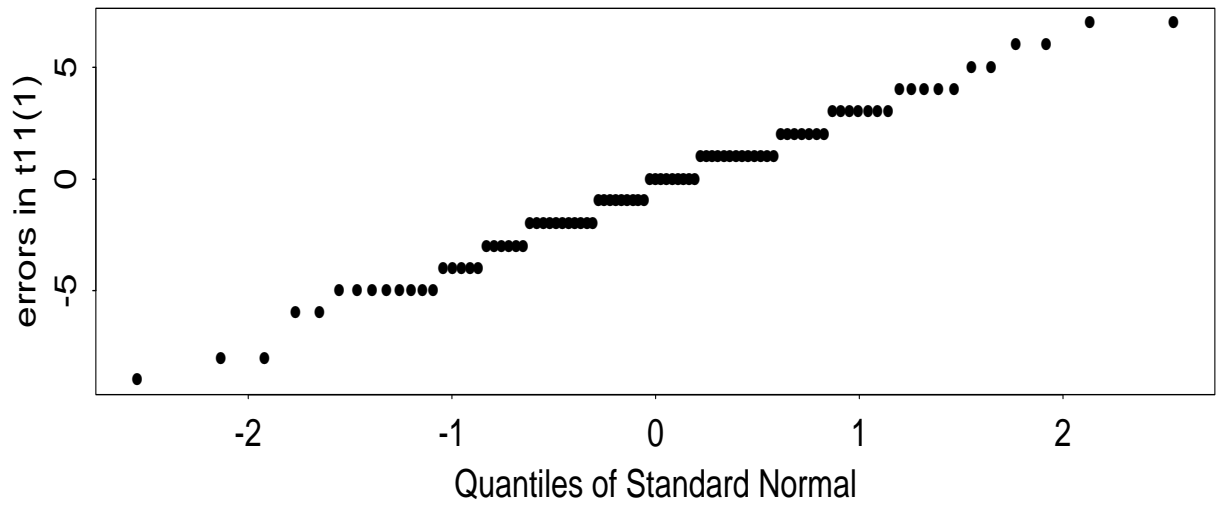
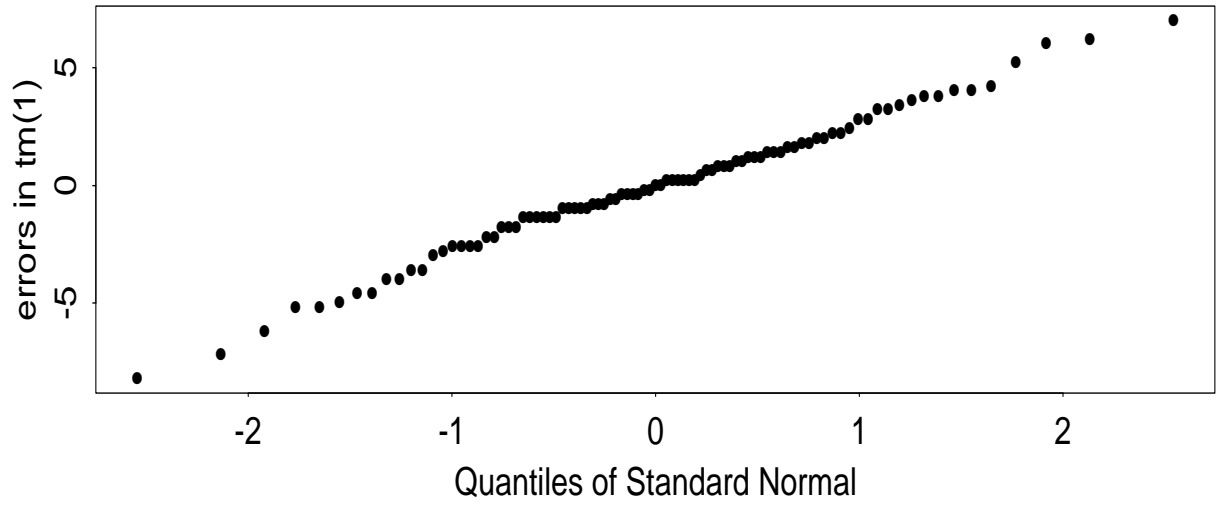


Figure 2: Normal probability plots for \underline{W} .

- Figure 1 A feedforward neural network with one hidden layer.
- Figure 2. Normal probability plots for \underline{W} .

- Table 1 Simulation Result.
- Table 2 Results for Utility Load Forecasting.

Table 1: Simulation Result.

Methods	Mean Squared Error
$\hat{g}_{\underline{\theta}}(\underline{W}, \underline{Z})$	0.0182
$\overline{\hat{g}}_{\underline{\theta}}(\underline{W}, \underline{Z})$	0.0137
$\overline{\hat{g}}_{\underline{\theta}}(\underline{W}, \underline{Z})$	0.0123
$\hat{g}_{\underline{\theta}}(\underline{X}, \underline{Z})$	0.0115
$\hat{g}_{\underline{\theta}}(\underline{X}, \underline{Z})$	0.0100

Table 2: Results for Utility Load Forecasting.

Methods	MAPE	CV	Median CV
$\hat{g}_{\underline{\theta}}(\underline{X}, \underline{Z})$	2.46%	3.32%	1.89%
$\hat{g}_{\underline{\theta}}(\underline{W}, \underline{Z})$	2.78%	3.56%	2.08%
$\overline{\hat{g}}_{\underline{\theta}}(\underline{W}, \underline{Z})$	2.66%	3.41%	2.07%
QUERI model	2.39%	3.09%	1.96%