IMPROVING NONLINEAR PROCESS MODELING USING MULTIPLE NEURAL NETWORK COMBINATION THROUGH BAYESIAN MODEL AVERAGING (BMA)

Z. AHMAD, TANG PICK HA, RABIATUL 'ADAWIAH MAT NOOR

School of Chemical Engineering, Universiti Sains Malaysia, Engineering Campus, Seri Ampangan, 14300 Nibong Tebal, Seberang Perai Selatan, Pulau Pinang, Malaysia

e-mail: chzahmad@eng.usm.my

ABSTRACT: Improving model generalization of aggregated multiple neural networks for nonlinear dynamic process modeling using Bayesian Model Averaging (BMA) is proposed in this paper. Using BMA method, the *posterior probability* of a particular network being the true model is used as the combination weight for aggregating the network despite of using fixed combination weight as the model. The posterior probabilities are calculated using the sum square error (SSE) from the training data on each of the sample time, and tested to the testing data. The selections for the final weight are based on the least SSE calculated when each of the *posterior probability* is applied to the testing data. The likelihood method is employed for calculating the network error for each input data. Then, it is used to calculate the combination weight for the networks. Two non-linear dynamic system-modeling case studies are selected for this proposed method, which are water tank level prediction and pH neutralization process. Application result demonstrates that the combination using BMA technique can significantly improve model generalization compared to other linear combination approaches.

KEY WORDS: *Multiple Neural Networks, Process Control, Mathematical Modeling, Bayesian Model Averaging, Nonlinear Process, Nonlinear Dynamics*

1. INTRODUCTION

As we know, many processes are complex and require a lot of time and effort to develop and validate the detailed theoretical dynamic models which often end up with a multivariable complicated mathematical model. As an alternative, a simpler empirical nonlinear model is often developed using experimental data. Neural networks or artificial neural networks (ANN) are attractive whenever it is necessary to model a complex or complicated processes with large input-output data sets as well as to replace models that are too complicated to solve in real time. Hence, neural networks have been used extensively in the recent years to model a wide range of physical and chemical phenomena and to model other non-engineering situations [1]. As mentioned by Willis et al. [2] more accurate representations of the processes are required to ensure good process control performance especially in Advanced Process Control.

Therefore, neural network models must be robust, stable and reliable when they are applied to a new (unseen) data. Zhang [3] stated that even though neural networks have a

significant capability in representing nonlinear functions, inconsistency of accuracy still seems to be a problem where neural network models may not perform well when applied to unseen data. Even though neural network models are very powerful non-linear modeling tools, noises in the input data sometimes cause the model over-fitting [4]. It can lead to a situation where poor generalization of a neural network model. Such circumstance is undesirable since it obviously strays from the goal of using neural network model. Another example is phenomena over fitting and under fitting in neural network training where a lot of researchers tried to solve it.

Over-fitting and under-fitting are two of the problems that should be alleviated in utilizing neural networks. In over-fitting, the error on the training data set is driven to a very small value, but when applied to unseen data, the network error is large and the generalization capability of the neural network is poor. While under fitting is due to that the neural network itself cannot cope with or fails to capture the relationship within the complex data [5].

In order to improve the robustness of neural networks and overcome the over and under fitting phenomena, a number of techniques have been developed lately like regularization (e.g [6]) and the early stopping method (e.g. [7]). Ohbayashi [8] implemented the universal learning rule and second order derivatives to increase the robustness in neural network models. Robustness is enhanced by minimizing the change in the values of criterion function caused by the small changes around nominal values of system parameters. Lack of robustness in individual neural networks is basically due to the over fitting of the models (e.g. [9]). Therefore, many researchers concentrate on how over fitting can be eliminated by improving the learning algorithm or by combining multiple imperfect neural networks.

The method combining multiple neural networks seems to show some encouraging results in terms of improving the robustness of neural networks [10]. There are several methods of combining multiple neural networks such as stacked neural network and bootstrap aggregated networks where multiple networks are created on bootstrap resample of the original training data. Using this method, several networks are combined (aggregated) and the aggregated predictor is used as the final representation as shown in Fig. 1. They can be both the same or different structure of neural network from different data sets and the training algorithm.

The overall output of the stacked neural network is a weighted combination of the individual network outputs. The main objective of this approach is to improve the generalization capability of the neural network models in such away that it will guard against the failure of individual component networks. Combining a set of imperfect models (networks) can be thought of as a way of managing the recognized limitation of the individual models, each is known to have errors but they are combined in so the effect of these errors are minimized. Proper determination of the stacking weights is essential for good modeling performance [10 - 12]. This is because each network output is weighted prior to summation of all the neural networks in the model, which will give the prediction value.

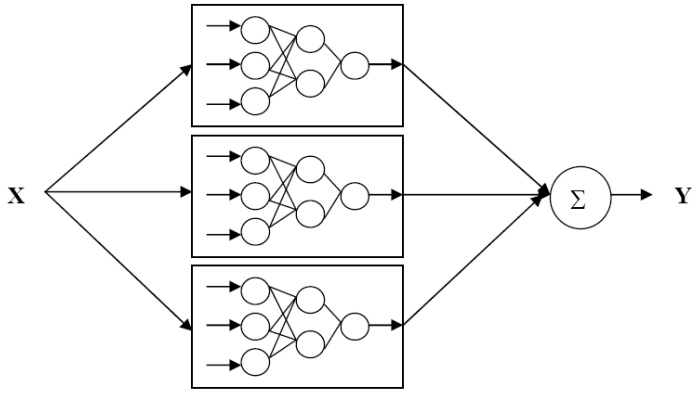

Fig. 1: Combining multiple neural networks. Re-sampling of X to get final output of Y

In this paper, it is proposed that BMA technique is used to combine multiple neural networks and to compare this method with linear combination techniques which are simple averaging and multiple linear regressions (MLR) techniques. Other methods of combining multiple neural networks are nonlinear combination of neural networks such as Demspter-Shafers which uses rank-based information, voting, order statistic and also Turner and Ghosh methods. In BMA approach, a proper model is selected at each sample of the model input in training data using the probabilities analysis. The probability of an individual network being the true model is calculated using the sum of squared errors (SSE) on the training data. Each set of weight from each sample time resulting from the training data is then applied to the testing data. For each prediction in the testing data, SSE is calculated and the set of weight that produced least SSE is selected for the final prediction output and employed to the unseen data as the final model.

The paper is organized as follows. Section 2 presents the combination using optimum selection of Bayesian Model Averaging (BMA). The applications of the proposed technique to two case studies are explained in Section 3. Section 4 covers the results and discussion and finally, the last section concludes this paper.

2. BAYESIAN MODEL AVERAGING (BMA) METHOD FOR COMBINING MULTIPLE NEURAL NETWORKS

Bootstrap re-sampling technique is proposed in this paper to randomly re-sample twenty different set of training, testing and validation data. Instead of choosing one 'best' neural network model, a set of imperfect estimators (net), which has different prediction performance are combined using BMA method to improve the generalization capability of the model.

The Bayesian approach allows the predictive performance and better accounting of uncertainty. This can be seen as this approach tries to minimize the effect of the errors of the model through its method of using the least SSE. It also takes into account every sample and set in the calculation. In some cases, for instance the heart attack data in Raftery [13], more than one model may have high posterior probability and these models will give different predictions. By using predictions from only a single model will grossly underestimate the variability of the estimates since it ignores the fact that another model

with significant posterior probability gives a different prediction. Despite of that, one should calculate predictions by using a weighted average over all models in the search space, where the weights are the posterior probabilities of the models [14, 15]. The BMA based combination of multiple neural networks can be represented as followed:

$$y^* = p_1 y_1 + p_2 y_2 + p_3 y_3 + \ldots + p_k y_k \quad (1)$$

where y^* is the aggregated network model prediction and p_k is the weight for combining the k-th network with y_k prediction.

For a dynamic process modeling, each of the individual networks model can be represented as Eq. (1). Suppose that neural network models are to be developed from the data set {X, Y}, where $X \in R^{N \times p}$ is the input data, $Y \in R^{N \times q}$ is the output data, N is the number of samples, p is the number of input variables and q is the number of output variables. To develop an aggregated neural network model containing n individual networks, the original data set can be re-sampled using bootstrap re-sampling with replacement to form replications of the original data set [16]. The n replications can be denoted as $\{X_{(1)}, Y_{(1)}\}$, $\{X_{(2)}, Y_{(2)}\}, \ldots, \{X_{(n)}, Y_{(n)}\}$, where $X_{(i)} \in R^{N \times p}$, $Y_{(i)} \in R^{N \times q}$, $i=1, 2, \ldots, n$. A neural network model can be developed on each of these replications, which can be partitioned into a training data set and a testing data set if cross-validation is used in network training and network structure selection and the predictions of these n network on the original data set are denoted as $\hat{Y}_1$, $\hat{Y}_2$, ….

For the start, one should determine the aggregating weight for k-th networks as the posterior probability that k-th network is a true model with the posterior probabilities of the entire n network sum to 1. This aggregating weight is calculated from the training data,

$$p_t^k = \frac{\left(1/\sqrt{2\pi\sigma}\right) p_{t-1}^k e^{-\left[\left(y_t - y_t^k\right)/\sigma\right]^2}}{\sum_{k=1}^{20} \left(1/\sqrt{2\pi\sigma}\right) p_{t-1}^k e^{-\left[\left(y_t - y_t^k\right)/\sigma\right]^2}} \quad (2)$$

where σ is the variance of the errors of the actual process output, y_t for t-th training data, k is the number of variables and t is the sampling data [12, 17].

However, there is a problem when applying Eq. (2) to practical applications. The error of each network at time t is required which is unknown as the true value of the process output at time t is unknown and is to be predicted. Thus, Eq. (2) needs to be modified in order to suit the practical applications. *Likelihood* method is then used to obtain the estimates of the likely errors at time t. This method works in such way that a given model input data point is compared to the model input data in the training and testing data sets. Data point in the training and testing data set that are likely is then obtained.

BMA method is used to select the final value of the prediction output as that of the network with give least SSE value of the posterior probability at any given time when apply to the testing data. Since earlier, model input data that are likely are employed. Therefore, the network prediction error for this data point seems to be closed to its "likely" error in the training and testing data. As the result, the "likely" error in the training and testing data sets are used as an estimate of the error of the network for the given input data point. From the obtained error, aggregated weight is then calculated based on Eq. (2).

The process of developing a model proceeds to the implementing step for the aggregated weight. They are employed to the testing data. The same equation as Eq. (1) is used to calculate the value of SSE for the testing data with y^* is the aggregated network model prediction and p_k is the weight for combining k-th network with y_k prediction taken from the least SSE on the testing data. The value of SSE of each sample time is then plotted. The global optimum of SSE is selected from all the plotted SSE of the testing data. Then, the selected weight based on the least or optimum SSE value is applied to the validation data to obtain the final prediction output. The steps of obtaining optimum weight can be summarized as follows:

Step 1: Generate n neural networks using bootstrap re-samples of the original training and testing data.
Step 2: Calculate the SSE for all networks on the original training data set for each sample time.
Step 3: Set the prior distribution of the 20 neural networks as equal. The neural networks are assumed to have the same prior probability with the sum of the probability of all the networks is 1. Hence, each network has a prior of 0.05.
Step 4: Calculate likelihood or conditional probability that k-th network is the true model in the reduced universe of the 20 neural networks prediction for the training data using the SSE in Step 2. The *likelihood* is calculated as follows:

$$\ell^k = \left(1/\sqrt{2\pi\sigma}\right) e^{-\left[\left(y_t - y_t^k\right)/\sigma\right]^2} \tag{3}$$

In this case, the probability distribution is assumed as a normal distribution where k is the number of variables and t is the sampling data.
Step 5: Find the posterior probability for each network by dividing the joint probability with the marginal probability of the first data set.
Ste p 6: In training data set, use the posterior probability of the first data set as the prior for the second data set. In the same way, the second data set posterior is set as the prior for the third data set. The iteration continues with new posterior probability calculated using the posterior after the previous data set as the prior for the next data set.
Step 7: Apply each posterior probability in every samples time calculated in Step 5 to the testing data. Compute SSE for each iteration. The selection of the final weight (or *posterior probability*) will be based on the least SSE produced in the testing data. Then, the weight is chosen as the final weight to obtain the aggregated neural network prediction.

3. APPLICATIONS TO TWO NONLINEAR PROCESS MODELING CASE STUDIES

Two case studies are used in this paper for evaluation and analysis of the proposed combination method for multiple neural networks. In each of the case study, 20 networks with fixed identical structure and 20 networks with various structures were developed from bootstrap re-samples of the original training and testing data. The meaning of various structures in this study is the number of neuron in the hidden layer is different in each individual network which randomly select in a certain range. The rational to choose 20 networks for all combination is based on the reference by Zhang [16, 18] where 30 neural network models through bootstrap re-sampling method and applied the simple averaging combination method. After combining the aggregated networks, a constant SSE was

observed after about 15 combinations. Therefore, combining 20 neural networks is reasonably fine. If the number of networks is too small, one might not get the optimum reduction of the SSE in the combination of the network. As a result, the "best" model to represent the system cannot be achieved.

In re-sampling the training and testing data using bootstrap re-sampling techniques, the training and testing data were already in discrete time function; therefore by re-sampling the discrete time function, it does not affect the sequence of input-output mapping of the prediction. Then, the individual networks were trained by the Levenberg-Marquardt optimization algorithm with regularization and "early stopping". All weights and biases were randomly initialized in the range from –0.1 to 0.1. The individual networks are single hidden layer feed forward neural networks. Hidden neurons use the logarithmic sigmoid activation function whereas output layer neurons use the linear activation function. To cope with the different magnitudes in the input and output data, all the data were scaled to zero mean and unit standard deviation.

The data for neural network model building need to be divided into: (1) Training data (for network training); (2) Testing data (for cross-validation based network structure selection and early stopping); and (3) Unseen validation data (for evaluation of the final selected model). The neural network models for both cases are developed using MATLAB (The Mathworks Inc.). In networks with fixed structure, the network structures, i.e. the number of hidden neurons, were determined through cross validation. Single hidden layer neural networks with different numbers of hidden neurons were trained on the training data and tested on the testing data. The network with the lowest sum of squared errors (SSE) on the testing data was considered as having the best network topology. In assessing the developed models, SSE on the unseen validation data is used as the performance criterion. For dynamic system modeling, multi-step-ahead or long range predictions should be taken into consideration. In many advanced process control applications, such as optimal control of batch processes, accurate long range predictions are required.

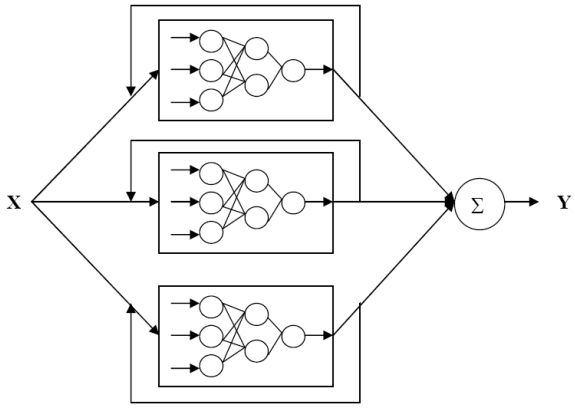

Fig. 2: Long-Range predictions with feedback before the combination of individual networks

For one-step-ahead predictions, the process output at time *(t-1), y (t-1),* is used as a model input to predict the process output at time *t,* $\hat{y}(t)$*,* as follows:

$$\hat{y}(t) = f[y(t-1), y(t-2), ..., y(t-n), u(t-1), u(t-2), ..., u(t-m)] \quad (4)$$

where *u(t-1)* is the process input at time *(t-1)*, *y(t)* is the predicted process output at time *t, m* and *n* are the time lags in the process input and output respectively. In long range predictions or multi-step-ahead predictions, the current and past model predictions are used to predict the future values of the model outputs:

$$\hat{y}(t) = f[\hat{y}(t-1), \hat{y}(t-2), ..., \hat{y}(t-n), u(t-1), u(t-2), ..., u(t-m)] \quad (5)$$

where the model prediction, $\hat{y}(t-1)$ to $\hat{y}(t-n)$, are used in place of the process outputs, *y(t-1)* to *y(t-n)*, to predict $\hat{y}(t)$. Here, one should bear in mind that for one-step-ahead predictions, the predicted process output, $\hat{y}(t)$ is calculated based on previous process output whereas for multi-step-ahead predictions, the predicted process output is determined based on previous and current process predictions. In other words, for one-step-ahead predictions, they request for previous real process output in order to predict future process output while multi-step-ahead predictions, they use previous and current predicted process output to predict future process output.

Accurate long range predictions are much more difficult to achieve than accurate one-step-ahead predictions due to the accumulation of the errors in the recursive predictions [19]. To obtain long range predictions from an aggregated network, two types of network output feedback schemes can be used as indicated in Fig. 2 and 3. In Fig. 2, the output of each individual network is fed back to its own input and the outputs of the individual networks are combined. This is equivalent to the step of combining the long range predictions of individual networks. In the scheme shown in Fig. 3, outputs of the individual networks are combined and the combined output is fed back to the inputs of all individual networks. In this paper, the first feedback scheme shown in Fig. 2 was adopted.

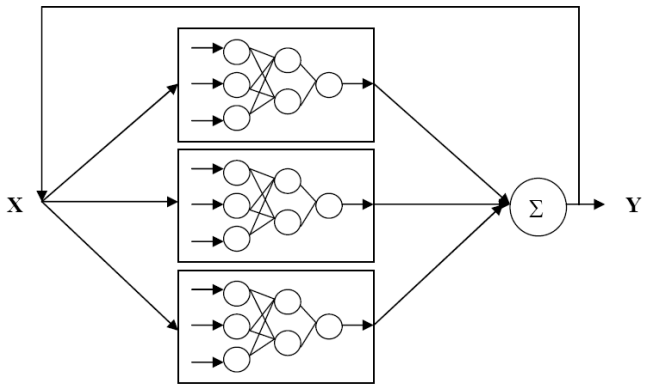

Fig. 3: Long-Range predictions with feedback after the combination of individual networks.

To test the performance of the proposed technique, the following networks/combination schemes are investigated: (A) Mean SSE with fixed structure; (B) Mean SSE with various structure; (C) Simple average of networks with fixed structures; (D) Simple average of networks with various structures; (E) MLR combination of networks with fixed structures,

(F) MLR combination of networks with various structures; (G) BMA combination for fixed structures; (H) BMA combination for various structures

3.1 Case Study 1: Water Tank Level Prediction

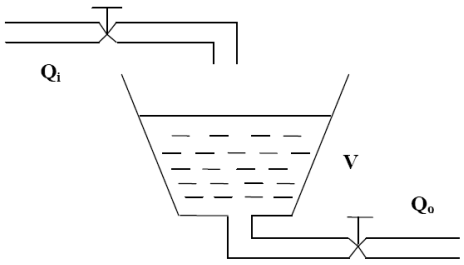

Fig. 4: A conic water tank

Figure 4 shows the schematic diagram of a conic water tank. There is an inlet stream to the tank and an outlet stream from the tank. Manipulating the inlet water flow rate will regulate the water tank level [3].

Let V, Q_i and Q_o be the volume of water in the tank and the inlet and outlet water flows rates respectively, then the material balance in the water tank can be written as:

$$\frac{dV}{dt} = Q_i - Q_o \tag{6}$$

The outlet water flow rate, Q_o, is related to the tank level, h, by the following equation:

$$Q_o = k\sqrt{h} \tag{7}$$

where k is constant for a fixed valve opening. The volume of water in the tank is related to the tank level by the following equation:

$$V = \pi h\left[r^2 + \frac{hr}{\tan\theta} + \frac{h^2}{3(\tan\theta)^2}\right] \tag{8}$$

where r is the tank bottom radius and is the angle between the tank boundary and the horizontal plane. Combining Eq. (6) to Eq. (8), the following dynamics model for the tank level can be obtained:

$$\frac{dh}{dt} = \frac{Q_i - k\sqrt{h}}{\pi\left[r^2 + \frac{2rh}{\tan\theta} + \frac{h^2}{(\tan\theta)^2}\right]} \tag{9}$$

Based on the above model, a simulation program is developed to simulate the process. The parameters used in the simulation are r = 10 cm, k = 34.77 $cm^{2.5}/s$ and θ = 60°. The sampling time is 10 seconds. The above equation indicates that the relationship between the inlet water flow rate and the water level in the tank is quite non linear. The outlet valve characteristic determines that the static gain increases with tank level. Because the tank is of a conical shape, the time constant of the processes increases with the tank level. Thus,

both the static and dynamic characteristics of the process vary with the operating condition. All the network building data were generated from the simulation programme and normally distributed noise with zero mean and a standard deviation of 0.7 cm were added to the simulated tank level.

For one-step-ahead predictions, the dynamic model for tank level prediction is of the form:

$$\hat{y}(t) = f\left[y(t-1), u(t-1)\right] \tag{10}$$

This is based on the one-step-ahead predictions equation as stated earlier in Equation 4, where $\hat{y}(t)$ represents the tank level prediction, y represents tank level, u represents the in let flow rate, f is a nonlinear function represented by the neural networks and t is the discrete time. Meanwhile, for long range predictions, it can be calculated as follows:

$$\hat{y}(t) = f\left[\hat{y}(t-1), u(t-1)\right] \tag{11}$$

where $\hat{y}(t)$ represents the tank level prediction which is included in the input of the next prediction, f is a nonlinear function represented by the neural networks and u represents the inlet flow rate. In the networks with fixed structure, the number of hidden neurons was determined based on cross validation technique. It was found that using 4 hidden neurons gives the least SSE on the testing data.

3.2 Case Study 2: pH Neutralization Process

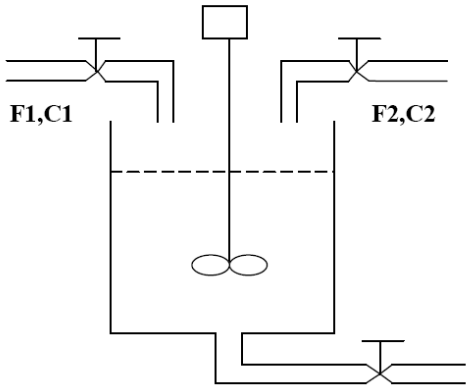

Fig. 5: CSTR for pH neutralization process

The neutralization process takes place in a CSTR and the re are two input streams to the CSTR. One is acetic acid of concentration C_1 at flow rate F_1 and the other is sodium hydroxide of concentration C_2 at flow rate F_2 [20]. The apparatus of the CSTR are shown in Fig. 5 and the mathematical equations of the CSTR can be described as follows by assuming that the tank level is perfectly controlled:

$$V\frac{d\zeta}{dt} = F_1C_1 - (F_1 + F_2)\zeta \tag{12}$$

$$V\frac{d\zeta}{dt} = F_2C_2 - (F_1 + F_2)\zeta \tag{13}$$

$$\left[H^+\right]+(K_a+\varsigma)\left[H^+\right]^2+\left[K_a(\varsigma-\zeta)K_w\right]\left[H^+\right]-K_wK_a \quad (14)$$

$$pH=\log_{10}\left[H^+\right] \quad (15)$$

where

$$\zeta=\left[HAC\right]+\left[AC^-\right] \quad (16)$$

$$\varsigma=\left[Na^+\right] \quad (17)$$

The meanings and nominal values of the variables in the above equations are given in Table 1. These equations show that the dynamic relation ship between the titration flow and pH in the CSTR is very nonlinear. To generate training, testing and validation data, multi-level random perturbations were added to the flow rate of acetic acid while other inputs to the reactor were kept constant. The pH measurements were corrupted with normally distributed random noise with zero mean and a standard deviation of 0.2. For one-step-ahead predictions, the dynamic model representing the neutralization process is of the form:

$$\hat{y}(t)=f\left[y(t-1),y(t-2),u(t-1),u(t-2)\right] \quad (18)$$

The above equation is reduced from Eq. (4) in the previous section, where $\hat{y}(t)$ is the pH in the reactor at time t, f is a nonlinear function represented by neural networks and $u(t)$ is the acid flow rates at time t. Long range predictions are calculated as follows:

$$\hat{y}(t)=f\left[\hat{y}(t-1),\hat{y}(t-2),u(t-1),u(t-2)\right] \quad (19)$$

where the model predictions, $\hat{y}(t-1)$ to $\hat{y}(t-2)$,are used in place of the process outputs, *y(t-1)* to *y(t-2),* to predict *y(t)*. It was found that using 5 hidden neurons gives the least SSE on the testing data in cross validation techniques.

4. RESULTS AND DISCUSSION

4.1 Case Study 1: Water Tank Level Prediction

In this case study, 20 single networks with fixed number of hidden neurons (4) and 20 networks with varying number of hidden neurons (between 1 to 10) were developed. Figure 6 shows the SSE on the unseen validation data of the individual networks for one-step-ahead predictions. For this case study, 79, 99 and 199 data were used for the purposes of training, testing and validation respectively. It can be seen that their performance varies quite significantly. This demonstrates the different generalization capabilities of the individual networks. These phenomena were also observed in long-range prediction. Once these individual networks were developed, they were combined using the different combination schemes proposed in section 2 and 3. It is shown in Fig. 6 that the lowest SSE on the unseen validation data for the single neural networks with fixed number of hidden neurons is 3.3032 (the 17th network) and the majority of these networks have SSE over 3.5. The mean SSE for fixed and various structures are 3.8122 and 4.0325 respectively.

Figure 7 shows the pattern of the SSE on the testing data when the BMA combination is applied. It is clearly shown that there is a couple of local minima in this combination and

the final weight for the combination are based on the iteration number 49 and 33 for fixed and various structures respectively which show the least SSE. Global minima were obtained after certain period of iteration where sometimes it can be assumed as the local minima and this will affect the final prediction.

In long-range prediction, the final weight were obtained after 7 and 33 iteration for fixed and various structures respectively as shown in Fig. 8. For long-range predictions, SSE in the testing data is quite consistent where the local minima are less compare to one-step-ahead predictions. It might be due to the individual neural networks predictions where the SSE is high due to the recursive error occurred in the predictions. The overall result for this case study is shown in Table 2 for both one-step and long-range predictions.

For one-step-ahead predictions, BMA shows quite a superior result compare to MLR and gives SSE of 3.45 and 3.83 for fixed and various structures, respectively. However, simple averaging performance is slightly better than BMA with SSE of 3.43 and 3.39 for fixed and various structures. For long-range prediction, BMA combination method is superior compare to mean, simple averaging and MLR. The SSE for BMA is 11.68 and 10.32 for fixed and various structures respectively compare to MLR with an SSE of 142.04 and 99.31. MLR ~~is~~ gave the worst performance in this case study and it might be due to the co-linearity of the individual neural networks that deteriorated the performance of MLR.

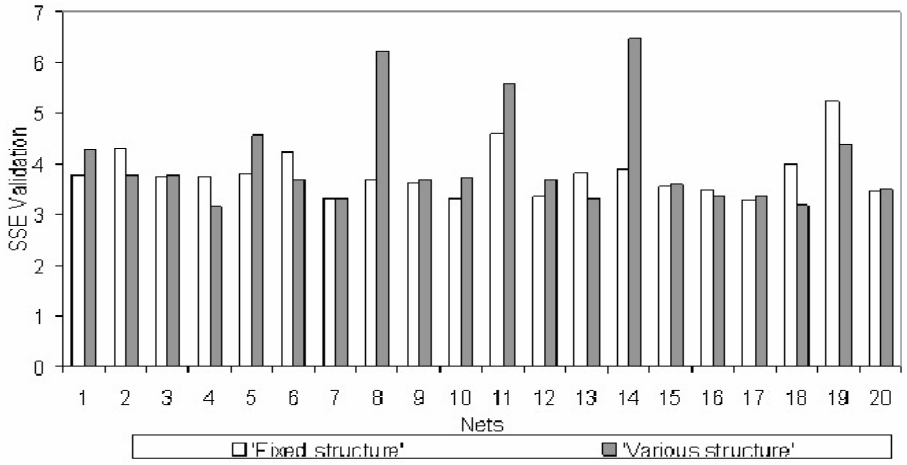

Fig. 6: Validation SSE in single networks with fixed and varying structures for Case Study 1

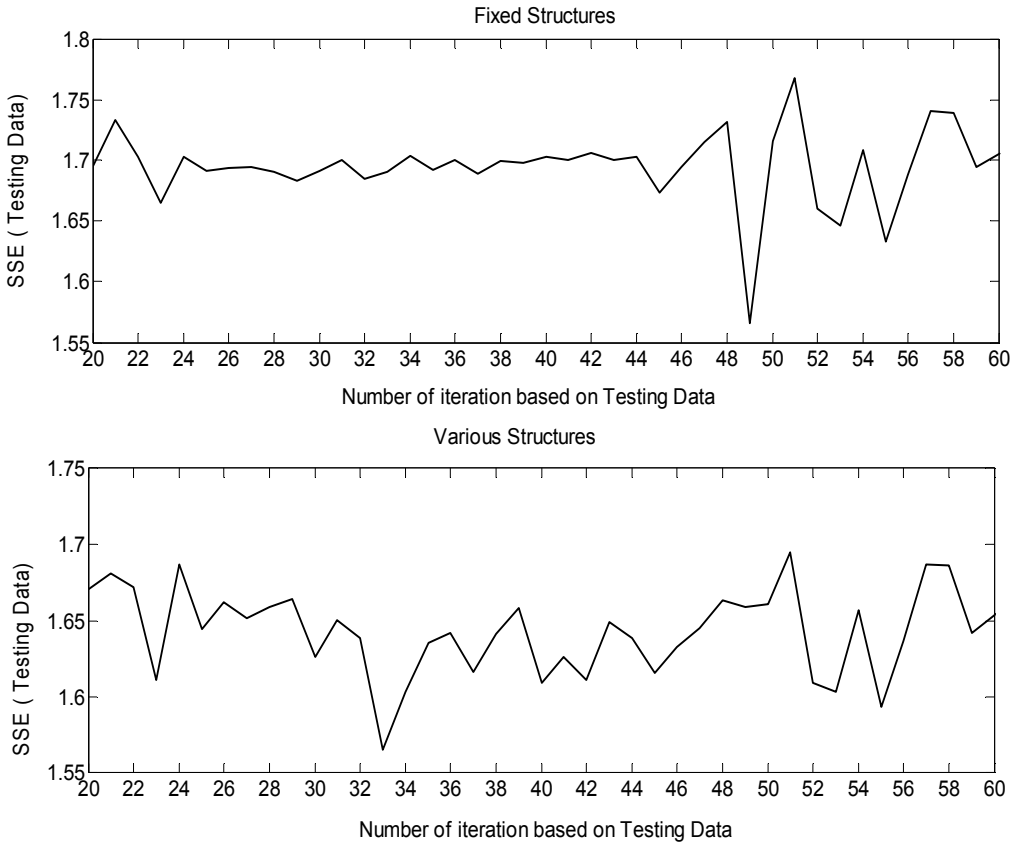

Fig. 7: SSE in testing data for fixed and varying structures using weight from the training data for Case Study 1: one-step-ahead prediction

4.2 Case Study 2: pH Neutralization Process

It is well known that the dynamics of pH is highly nonlinear. In this case study, 20 networks with fixed number of hidden neurons (5) and 20 networks with various numbers of hidden neurons (between 1 and 10) were developed. For this case study, 300 data were used for each training, testing and validation steps. Figure 9 shows the SSE of long-range predictions on the unseen validation data from the individual networks. It can be seen from Fig.10 that single networks give various performance. This demonstrates the variation in individual network performance and the non-robust nature of single networks.

Figure 9 also indicates that the lowest SSE achieved by single networks with fixed and various structures are 49.3 and 39.77, respectively. The worst performance for single networks with fixed structure is from network 9 with an SSE of 2966 on the unseen validation data. The worst performance for single networks with various structures is from network 20 with an SSE of 293 on the unseen validation data. The mean SSE for fixed and various structures are 328.4 and 106.6. The proposed combination scheme was then applied with the results given in Fig. 10, Fig. 11 and the overall result are shown in Table 3. Figure 10 shows the variation of SSE when applying the weight from the training data to the testing data for fixed and various structures. It is clearly shown that, it is not easy to

obtain the global minima in this case study. The least SSE obtained on iteration number 260 for fixed structures and iteration number 136 for various structures.

Table 1: Nominal values for Case Study 2

Variables	Meanings	Nominal Values
V	Volume of the tank	1 L
F_1	Flow rate of acid	0.081 L/min
F_2	Flow rate of base	0.512 L/min
C_1	Concentration of acids in F1	0.32 mol/L
C_2	Concentration of acids in F2	0.05 mol/L
K_a	Acid equilibrium constant	1.8×10^{-5}
K_w	Water equilibrium constant	1.0×10^{-14}

In long-range predictions, the global minima are easy to obtain where the variation of SSE is quite consistent as shown in Fig. 11. The global minima was achieved after 135 and 84 iteration for fixed and various structures respectively. Then, the final weight corresponding to the least SSE in the testing data is chosen as the final model for prediction output. The combined networks with fixed and various structures based on the BMA give SSE of 8.06 and 6.59 respectively on the validation data for one-step-ahead predictions. It is shown in Table 4 that the BMA combination scheme gives smaller SSE on the unseen validation data than mean and simple averaging method for all networks except in MLR with SSE of 7.25 and 7.84 respectively. However, this does not leave much effect compare to the BMA combination approach. The 'best' combination is coming from scheme H which is a BMA combination with various structures with SSE of 6.59.

Table 2 Overall results for Case Study 1: One-step-ahead predictions and long-range predictions

Combination Schemes	SSE (Validation)	
	One-step-ahead predictions	Long-range predictions
A	3.80	92.34
B	4.03	52.79
C	3.44	20.62
D	3.99	12.12
E	5.94	142.04
F	7.27	99.31
G	3.45	11.68
H	3.83	10.32

Table 3: Overall results for Case Study 2: One-step-ahead predictions and long-range predictions

Combination Schemes	SSE (Validation)	
	One-step-ahead predictions	Long-range predictions
A	10.44	328.00
B	10.12	106.55
C	8.78	57.31
D	7.84	43.84
E	7.25	124.21
F	6.39	270.63
G	8.06	53.04
H	6.59	37.21

In long-range predictions, combination using BMA approach outperforms the other combinations with SSE for validation data as 53.04 and 37.21 for fixed and various structures respectively. MLR shows quite poor performance in this case with SSE of 124.21 and 270.63 for fixed and various structures. The 'best' combination in this case is coming from combination scheme H which is a BMA combination with various structures with SSE of 37.21.

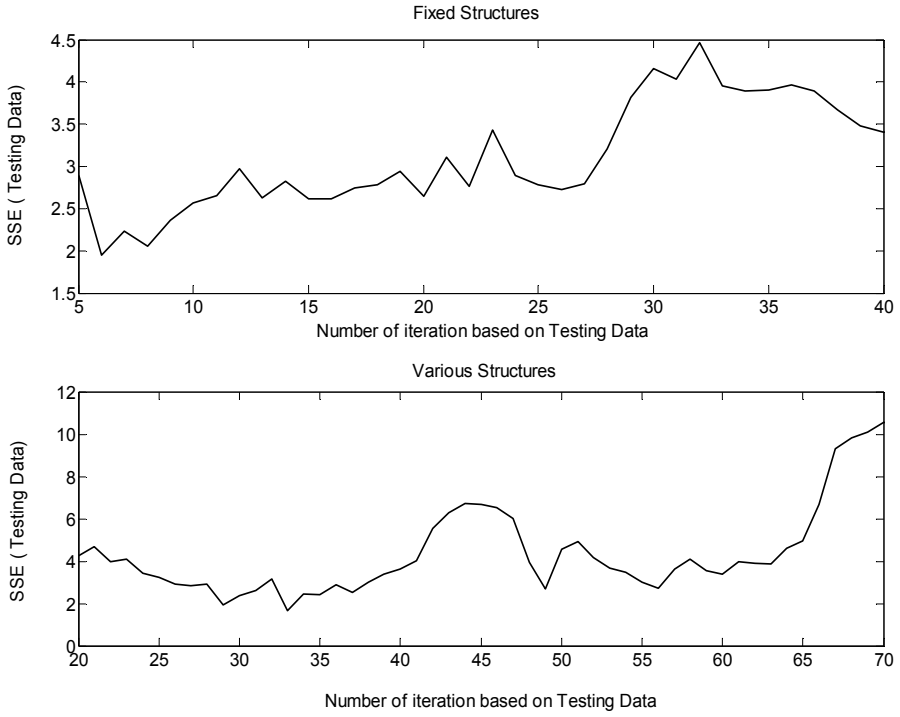

Fig. 8: SSE in testing data for fixed and varying structures using weight from the training data for Case Study 1: long-range predictions

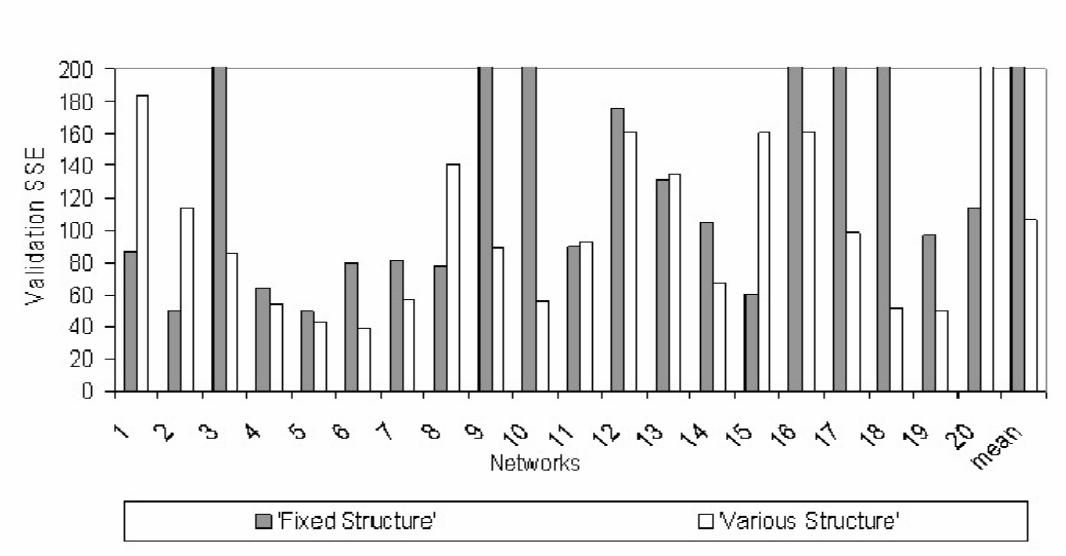

Fig. 9: SSE of long-range predictions on the validation data from single networks with fixed and various structures in Case Study 2

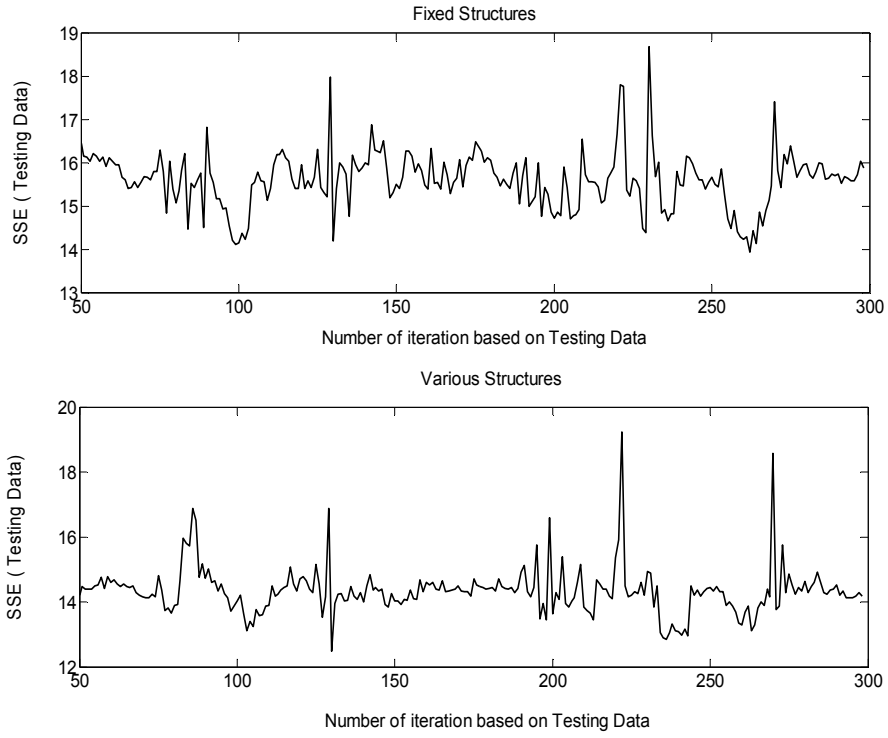

Fig. 10: SSE in testing data for fixed and varying structures using weight from the training data for Case Study 2: one-step-ahead predictions

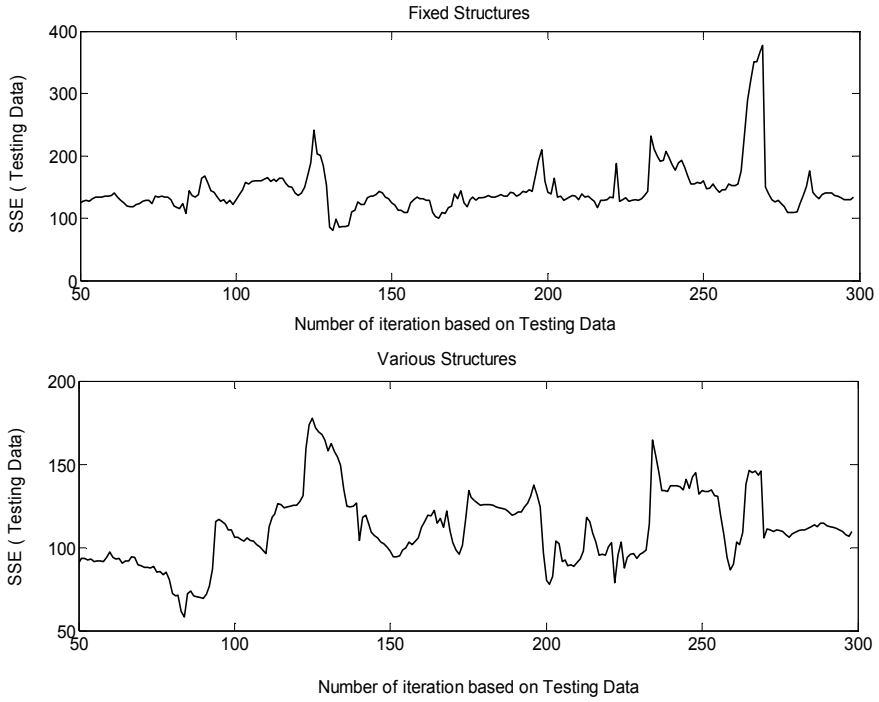

Fig. 11: SSE in testing data for fixed and varying structures using weight from the training data for Case Study 2: long-range predictions

5. CONCLUSION

In multiple neural networks, the generalization capability of each network is somehow different. Thus, it is not possible to find the 'best' network to model the process by looking at the SSE of the training and testing data. Combining these networks have proved to be promising in improving the robustness of the model by sharing and averaging out the errors. Therefore, in this paper, combination using BMA approach is introduced by selectively choose the optimum weight by looking at the weight that have the least SSE in training data. As a comparison, linear combination of simple averaging and MLR were implemented.

The simple averaging and MLR combinations have not been able to perform well compare to BMA proved by the larger prediction errors and unsatisfactory resulting model by using these approaches. Even though they are relatively easy to understand, the simple averaging and MLR combinations do not perform adequately well for this application. BMA shows a superior result in this study where in both nonlinear case studies, clearly, BMA have the advantages. The prediction error is much smaller or decreased compare to other linear combinations and at the same time, BMA improves the model selection (weight selection).

ACKNOWLEDGEMENTS

This work was supported by University Science Malaysia (USM) through short term grant 6035182 and 6035253.

REFERENCES

[1] D. E. Seborg, T. F. Edgar and D. A. Mellichamp, "Process Dynamics and Control", John Wiley & Sons, US, 2004.

[2] M. J. Willis, C. Di Massimo, G. A. Montaque, M. T. Tham and A. J. Morris, "Artificial Neural Networks In Process Estimation And Control", Automatica, Vol. 28, No. 6, 1181-1187, 1992.

[3] J. Zhang, "Developing Robust Neural Network Models By Using Both Dynamic And Static Process Operating Data", Ind. Eng. Chem. Res., Vol. 40, 234-241, 2001.

[4] S. Mc Loone and G. Irwin, "Improving neural networks training solution using regularization", Neurocomputing,Vol. 37, 71-90, 2001.

[5] [X. Guyon and J. Yao, "On the Under-Fitting and Over-Fitting Sets Of Model Chosen by Order Selection Criteria", Journal of Multivariate Analysis, Vol. 70, 221-249, 1999.

[6] F. Girosi, M. Jones and T. Poggio, "Regularization Theory and Neural Networks Architecture", Neural Computation, Vol. 7, 219-269, 1995.

[7] N. Morgan and H. Bourlard, "Generalization And Parameter Estimation In Feedforward Nets: Some Experiments", in: Touretzkey, D.S. (Eds.), Advances in Neural Information Processing System 2, San Mateo, CA, 630-637, 1990.

[8] M. Ohbayashi, K. Hirasawa, K. Toshimitsu, J. Murata and J. Hu, "Robust Control For Non-Linear System By Universal Learning Networks Considering Fuzzy Criterion And Second Order Derivatives", Proceedings of IEEE World Congress on Computational Intelligence, 968-973, 1998.

[9] R. Caruana, S. Lawrence and C. Lee Giles, "Over-Fitting In Neural Networks: Back Propagation, Conjugate Gradient And Early Stopping", Neural Information Processing System, 13, 402-408, 2000.

[10] Z. Ahmad and J. Zhang, "Bayesian Selective Combination of Multiple Neural Networks for Improving Long-Range Predictions In Nonlinear Process Modelling", Neural Computing and Applications, 14, 78-87, 2005.

[11] J. Zhang and A. J. Morris, "Recurrent Neuro-Fuzzy Networks For Nonlinear Process Modeling", IEEE Transaction of Neural Networks, Vol. 10, No. 2, 313-326, 1999.

[12] Z. Ahmad and J. Zhang, "Combination of Multiple Neural Networks Using Data Fusion Techniques for Enhanced Nonlinear Process Modeling", Computers and Chemical Engineering, Vol. 30, 295–308, 2005.

[13] A. Raftery, "Approximate Bayes Factors and Accounting for Model Uncertainty in Generalized Linear Models", Biometrika, Vol. 83, 251-266, 1996.

[14] E. E. Leamer, "Specification Searches: Ad Hoc Inference with Non-experimental Data", Wiley, New York, 1978.

[15] R. E. Kass and A. E. Raftery, "Bayes Factors", Journal of the American Statistical Association, Vol. 90, No. 430, 773-795, 1995.

[16] J. Zhang, "Developing Robust Non-Linear Models Through Bootstrap Aggregated Neural Networks", Neurocomputing, Vol. 25, 93-113, 1999.

[17] S. Kiartzis, A. Kehagias, A. Bakirtzis and A. Petridis, "Short Term Load Forecasting Using A Bayesian Combination Method", Electrical Power and Energy System, Vol. 19, No. 3, 171-177, 1997.

[18] J. Zhang, "Inferential Estimation Of Polymer Quality Using Bootstrap Aggregated Neural Networks", Neural Networks, Vol. 12, 927-938, 1999.

[19] J. Zhang, A. J. Morris and E. B. Martin, "Long-Term Prediction Models Based On Mixed Order Locally Recurrent Neural Networks", Computers and Chemical Engineering, Vol. 22, No. 7-8, 1051-1063, 1998.

[20] T. J. McAvoy, E. Hsu and S. Lowenthal, "Dynamics of pH in Controlled Stirred Tank Reactor", Ind. Chem. Process Des. Dev., Vol.11, 68-70, 1972.

NOMENCLATURE

C1	Acetic acid inlet concentration, mol/L
C2	Sodium hydroxide inlet concentration, mol/L
F1	Acetic acid input flow rate, L/min
F2	Sodium hydroxide input flow rate, L/min
K_a	Acid equilibrium constant
K_w	Water equilibrium constant
N	Number of samples
P	Loading matrix
Q_i	Inlet water flow rate, cm^3/s
Q_o	Outlet water flow rate, cm^3/s
V	Volume of water in the tank, cm^3
X	Input data vector
Y	Output data vector
h	Water level in the tank, cm
k	Constant for a fixed valve opening, $cm^{2.5}/s$
r	Radius of the tank, cm
t	Sampling time for case study 1, s
	Sampling time for case study 2, min