Journal of the Operational Research Society (2001) 52, 652–664 ©2001 Operational Research Society Ltd. All rights reserved. 0160-5682/01 $15.00

www.palgrave-journals.com/jors

Time series forecasting with neural network ensembles: an application for exchange rate prediction

GP Zhang[1]* and VL Berardi[2]

[1]*Georgia State University, Atlanta, GA, USA; and* [2]*Kent State University, Kent, OH, USA*

This paper investigates the use of neural network combining methods to improve time series forecasting performance of the traditional single keep-the-best (KTB) model. The ensemble methods are applied to the difficult problem of exchange rate forecasting. Two general approaches to combining neural networks are proposed and examined in predicting the exchange rate between the British pound and US dollar. Specifically, we propose to use systematic and serial partitioning methods to build neural network ensembles for time series forecasting. It is found that the basic ensemble approach created with non-varying network architectures trained using different initial random weights is not effective in improving the accuracy of prediction while ensemble models consisting of different neural network structures can consistently outperform predictions of the single 'best' network. Results also show that neural ensembles based on different partitions of the data are more effective than those developed with the full training data in out-of-sample forecasting. Moreover, reducing correlation among forecasts made by the ensemble members by utilizing data partitioning techniques is the key to success for the neural ensemble models. Although our ensemble methods show considerable advantages over the traditional KTB approach, they do not have significant improvement compared to the widely used random walk model in exchange rate forecasting.

Keywords: neural network ensemble; exchange rate; time series; forecasting

Introduction

Artificial neural networks have been widely used as a promising alternative approach to time series forecasting. A large number of successful applications have established the role of neural networks in time series modelling and forecasting. Neural networks are data-driven, self-adaptive nonlinear methods that do not require specific assumptions about the underlying model. Instead of fitting the data with a pre-specified model form, neural networks let the data itself serve as direct evidence to support the model's estimation of the underlying generation process. This nonparametric feature makes them quite flexible in modelling real-world phenomena where observations are generally available but the theoretical relationship is not known or testable. It also distinguishes neural network models from traditional linear models and other parametric nonlinear approaches, which are often limited in scope when handling nonlinear or nonstandard problems.

The most popular neural networks used in forecasting are the single multi-layer feedforward model or multi-layer perceptron (MLP). Although it has been shown theoretically that the MLP has a universal functional approximating capability and can approximate any nonlinear function with arbitrary accuracy, no universal guideline exists in choosing the appropriate model structure for practical applications. Thus, a trial-and-error approach or cross-validation experiment is often adopted to help find the 'best' model. Typically a large number of neural network models are considered. The one with the best performance in the validation set is chosen as the winner, and the others are discarded.

There are several limitations with this keep-the-best (KTB) approach in choosing a neural network model. First, the network ultimately selected may not be the true optimal model because of a large number of factors that could affect neural network training and model selection. These factors include network architecture, activation functions, training algorithm, and data normalization. Different choices of these factors can lead to alternative models being selected. Second, neural networks are data driven methods. Alternative data sampling from a stationary process can have a significant effect on individual model selection and prediction. This impact may be magnified if the process parameters evolve or shift over time. As a result, the final KTB model may overfit the specific sample

*Correspondence: GP Zhang, Department of Management, J Mack Robinson College of Business, Georgia State University, Atlanta, GA 30303, USA.
E-mail: gpzhang@gsu.edu

data, limiting its generalization ability to the unseen future data. Finally, by discarding many network models, useful information contained in these models as well as the efforts in training are wasted.

In this paper we investigate the potential of combining multiple neural network models for time series forecasting. Our purpose is to show that, by appropriately combining different neural forecasters, forecasting performance of the individual network can be improved. The combination of several models to form an ensemble of networks can be an effective way to overcome the problems associated with the keep-the-best single network model. In particular, it can alleviate the model selection problem typically encountered in neural network applications. With the ensemble of neural networks, it is no longer an issue to select one particular network model with the 'best' predictive ability. Rather, results from multiple models are combined to make forecasts. By solving the same problem using multiple networks, model selection bias and prediction error can be reduced.[1,2] In addition, the ensemble method is more likely to produce stable forecasts and less likely to make catastrophic predictions than any single network used in isolation. Another advantage of the neural network ensemble is that it often requires little additional effort due to the trial-and-error nature in the neural network model building process.

Ensembles represent a natural extension of common neural network practice that undoubtedly owes its roots to earlier works using conventional forecasting tools. Combining several models to improve the forecasting accuracy has been extensively studied in the traditional forecasting literature. Clemen[3] provided a comprehensive review and annotated bibliography in this area. The basic idea of model combination is to use each model's unique feature to capture different patterns in the data. The efficacy of forecast combination has been established both theoretically and empirically. In the well-known M-competition,[4] most conventional forecasting models are tested using more than 1000 real-time series and the combination of forecasts from more than one model often leads to improved forecasting performance. Because much of the research effort into neural ensemble performance has concentrated on classification problems to date, this study aims to provide some evidence on the effectiveness of forecasting combination using neural networks as well as to provide some guidelines to forming neural ensembles for time series applications.

One time series application that is notoriously difficult to predict is that of exchange rates. The field has long been dominated by the efficient market hypothesis, which postulates that current prices reflect all relevant knowledge and information about the markets and therefore future returns are essentially unpredictable. Under this assumption, the best theoretical model for exchange rate prediction is the random walk. Mixed empirical results, however, have been reported with regard to the ability of various linear and nonlinear models in outperforming the random walk model.[5–11]

Several researchers have applied neural networks in forecasting foreign exchange rates. For example, Kuan and Liu[12] examined the out-of-sample forecasting ability of neural networks on five exchange rates against the US dollar, including the British pound, the Canadian dollar, the German mark, the Japanese yen, and the Swiss franc. For the British pound and Japanese yen, they demonstrated that neural networks had significant market timing ability and/or achieve significantly lower out-of-sample MSE than the random walk model across three testing periods. For the other three exchange rate series, neural networks were not shown superior in forecasting performance. Their results also showed that different network models performed quite differently in out-of-sample forecasting. Borisov and Pavlov[13] applied neural networks to forecast the Russian ruble exchange rate. Both neural networks and exponential smoothing models were used to predict the exchange rates. Although a backpropagation-based neural network performed well in all cases, they found that, if the data set contained some extreme values, exponential smoothing was the preferred method. On the other hand, if the amount of noise in the series was limited, then a window-based neural network method should be used. In forecasting the US dollar/German mark exchange rate, Hann and Steurer[14] compared neural network models with linear monetary models. Out-of-sample results showed that, for weekly data, neural networks were much better than linear models and the naive prediction of a random walk model with regard to Theil's U measure, the hit rate, the annualized returns and the Sharpe ratio. However, if monthly data were used, neural networks did not show much improvement over linear models. Zhang and Hutchinson[15] reported the experience of forecasting the tick-by-tick Swiss franc *vs* US dollar exchange rate time series with neural networks. Using different sections of the data set, they found mixed results for neural networks in comparison with those from the random walk model.

It is important to note that all of these studies, and many others, used a single neural network model in modelling and predicting exchange rates. As the data-dependent neural networks tend to be more 'unstable' than the traditional parametric models, the performance of the KTB approach can vary dramatically with different models and data. Random variations resulting from data partitioning or subtle shifts in the parameters of the time series generating process can have a large impact on the learning and generalization capability of a single neural network model. These may be the reasons that neural networks perform differently for different exchange rate series and different time frequencies with different data partitions in the reported findings. Neural network ensembles seem promising for improving predictions over the KTB approach because they do not solely rely on the performance of a single neural network model.

In addition to providing insights in the application of ensemble methods to exchange rate forecasting, our major contribution of this paper is to develop novel methods to form neural network ensembles. We also evaluate many other principled combining methods and recommend the best ways to construct neural ensembles for time series forecasting. To date, there has been a high level of research activities in applying neural ensemble methods for classification problems with very few in time series forecasting. We believe that the methodology developed in this paper will be useful for general applications of time series prediction.

The remainder of the paper is organized as follows. In the next section, the theory underlying neural network ensembles is reviewed and general approaches to improving basic ensemble performance are discussed. In the section that follows, the research methodology is covered. This includes description of the data and a detailed discussion of the neural network ensemble designs employed. The final two sections contain the experimental results and conclusions.

The neural network ensemble

Because of their nonparametric nature, neural networks are generally able to fit the training or in-sample data very well. While this learning capability is desirable, it does not come without price. Namely, neural networks may become too dependent on the specific data upon which the model is built. This is the notorious overfitting problem. It occurs when a predictive model based upon in-sample data has poor generalization to unseen or out-of-sample data. Obviously this is undesirable for many forecasting problems.

This learning and generalization scheme is well studied under the concept of bias and variance tradeoff.[16] A model that is very complex or flexible tends to have small model bias but large model variance while a model that is relatively simple or inflexible may have a large bias but a small variance. Model bias reflects the learning ability of a neural network model to approximate the underlying data generating function. On the other hand, model variance measures the generalization capability of the trained model to the unseen future data with regard to the lack of stability of predictions from models developed from training data sets. Both bias and variance are important because they both contribute to the overall estimation and prediction error. However, with a given data set, it may not be possible to improve both bias and variance simultaneously because the effect to reduce the bias (or variance) is likely to increase the variance (or bias). Hence tradeoff between bias and variance is often necessary to build a useful model that is able to predict accurately and reliably.

Currently there is no simple, effective way to build and select a neural network model. The most commonly used approach employs the cross-validation. That is, a number of different network architectures are first trained with a portion of the data—a training sample—then the generalization or forecasting performance is monitored using another validation sample that is withheld before the training starts. The neural network with the best validation sample result is selected as the final prediction model, whose performance can be further tested through a third portion of the data called a test sample. Although this method can be effective in dealing with the overfitting problem, the model selected is still likely to under-perform in that it may not be the best model that will predict well for the future. Data randomness, limited sample size, data splitting, and instability in the parameters of the time series generating process can all attribute to poor generalization. In addition, because of the nonlinear optimization nature in neural network training, the parameter estimation cannot be guaranteed optimal in building a network model.

Combining multiple networks can be an effective way to overcome or reduce the overfitting problem as well as the drawbacks related to the traditional keep-the-best (KTB) single model selection approach. Instead of seeking one best solution to a problem, multiple alternative solutions are found and then combined in a systematic manner to make predictions. The main motivation for the neural ensemble method is to improve the generalization and prediction ability. Due to the inherent randomness in the training data, we are often willing to tradeoff some bias for low variance because a useful prediction model is the one that should perform well for out-of-sample data. This is especially important with neural networks given their direct reliance on the data for determining the model form. That combining different models can reduce variance of out-of-sample forecasting and therefore improve forecasting accuracy is also recognized in the traditional forecasting literature.[3,17]

Although neural network ensemble can be an effective method in reducing model variance and improving neural network forecasting performance, the effectiveness of the ensemble model may be limited if the errors generated by different component models are correlated.[15] Hence it is often advantageous to include different models in an ensemble that can generate independent forecasts for the future values.

Neural ensemble models can be formed in many different manners. One general approach to creating ensemble members is to hold the training data constant and to vary different factors/parameters of neural networks. For example, ensemble components can be networks trained with different initial random weights, networks trained with different algorithms, networks with different architectures that may include varying number of input and/or hidden nodes, or even networks with different types. Another broad approach is combining networks trained with different sample data. It is intuitive that networks built from different training samples are more likely to have less dependent prediction errors than those trained on the same data. Therefore, modifying training data to create the individual network components would be possibly a valuable approach to ensemble composition.

Although simple averaging is the most popular method for combining networks, weighted averaging methods and nonlinear combining methods can also be used. By means of weighted average, relative importance of the individual models can be accounted for. Different weighting schemes have been proposed and studied.[15,18,19] with no consensus on the best way to assign weights.

Research methodology

Data

Weekly exchange rate data between the British pound (BP) and the US dollar (USD) from 1976 to 1994 are used in this study for detailed examination. Daily data are obtained from DataStream International. The time series is compiled from daily rates by taking the closing rates on Wednesday as the representative rates of that week to avoid potential biases caused by the weekend effect. If a particular Wednesday happens to be a nontrading day, then either Tuesday or Thursday closing rates will be retained. In summary, a total of 978 data points is obtained in the data series. Figure 1 plots the exchange rate series.

To investigate the effectiveness of neural network ensembles, the total available data are divided into three parts for training, validation and test. Data from 1976 to 1990 with 782 observations are used for model building and the next 104 observations in 1991 and 1992 are used for the validation purpose. The out-of-sample period consists of the last 92 points.

Design of neural network ensemble

Multilayer feedforward neural network models are the most popular network paradigm for forecasting applications and are the focus of this paper. Compared to traditional statistical forecasting models, neural networks have more factors to be determined. Those factors related to neural network model architecture include the number of input variables, the number of hidden layers and hidden nodes, the number of output nodes, the activation functions for hidden and output nodes, and the training algorithm and process.[20] We are interested in one-step-ahead forecasts as in several previous studies.[21] Hence a single output node is employed. Because a single hidden-layer network has been shown both theoretically and empirically capable of modelling any type of functional relationship, it is exclusively used in our study. The activation functions used for all hidden nodes are the logistic function while the identity function is employed in the output layer. Node biases are used in both hidden and output layers.

The number of input nodes is perhaps the most critical parameter since it corresponds to the number of past lagged observations used to capture the underlying pattern of the time series. Since there is no theoretical result suggesting the best number of lags for a nonlinear forecasting problem, we will experimentally vary this parameter with 5 levels from 1–5. Another important factor is the number of hidden nodes, which empowers neural networks with the nonlinear modelling capability. However, it has been shown that the in-sample fit and the out-of-sample forecasting ability of neural networks are not very sensitive to the number of hidden nodes.[22] Therefore, we use three levels of hidden nodes of 2, 4; and 8 to experiment with in this study.

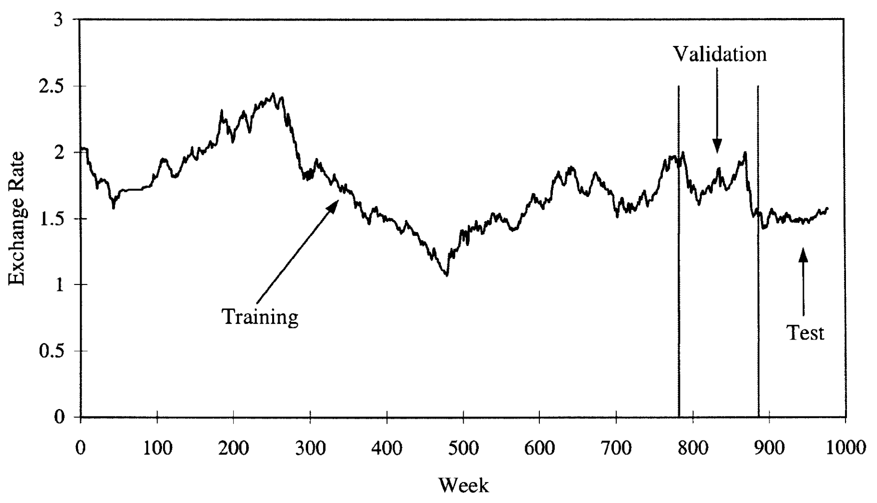

Figure 1 British pound *vs* US dollar exchange rate.

The most popular algorithm for training neural networks is the back-propagation method (Backprop).[23] Backprop is a first-order optimization method based on the steepest descent algorithm that requires a step size (or learning rate) to be specified. Because of the instability problem in the steepest descent algorithm, often a more robust training approach such as quasi-Newton, scaled conjugate gradient, Levenberg–Marquardt, and generalized reduced gradient is used. In this study, we use a training algorithm based on the scaled conjugate gradient.

As mentioned in the previous section, there are many different ways to build a neural network ensemble. While theoretical results indicate that, if properly constructed and employed, neural network ensembles can generalize better than any individual model used separately, they do not provide general guidelines about the selection of different models in the ensemble. This study evaluates the effectiveness of several methods of creating an ensemble in predicting the exchange rates. First, we combine neural networks trained with different initial random weights but with the same data. Since neural network training entails a nonlinear optimization problem, and the global optimal solution is not guaranteed by any of the currently available algorithms, a particular training run can be quite sensitive to the initial conditions of the weights in a network. That is, the neural network trained with different starting weights may be stuck with different local minima, each of which can have quite different forecasting performance.

Hence, using different starting weights to train a neural network and then keeping the best (KTB) network is a common practice to reduce the local minima effect. Combining results from different initial random weights seems to be a natural and efficient way to improve the performance of individual network because it makes use of all the information (local minima) and efforts used in the network training process. For each of the neural network structure, 50 different initial weights are randomly generated to start the training process. Then the prediction results generated from these 50 trained networks with the same architecture are combined in the ensemble.

The second approach considered combines different neural network architectures within an ensemble. This is also a potential method in improving neural network performance with little extra computational effort. Because of the lack of effective model selection approaches, an individual neural forecaster is often selected via a trial and error method. That is, several different network architectures are typically trained on the training sample with their prediction performances evaluated on a separate validation sample. The best network in terms of the validation result is selected to be the final model for forecasting. Due to the randomness in the total available data and the splitting of the data into several parts of training and validation, the selected model may not be the best one for future use. Hence combining different models potentially could be an effective way to relieve the single model selection problem inherent in neural network applications. In this study, network architectures with different input and hidden nodes are used to form the ensemble. A total of 15 neural network structures with five levels of input lags and three levels of hidden nodes are trained with the same training data and then the results are combined either across all hidden nodes or across all input nodes.

Note that while in this second approach the neural ensemble models are created with varying neural network factors, both approaches utilize the fixed training data set. Networks trained on different sample data can also be used to form neural ensembles. Here we propose two data-partitioning schemes (partition ensembles) for time series applications in order to reduce the harmful correlation among predictions from member networks in the ensemble.

The first scheme is called the *systematic* partition. Depending on the input lags used in the neural network, original time series data are partitioned into several subsamples. For a k-lag network structure, the whole training set is divided into k subsets of approximately same size. In this scheme, k data partitions with distinctive target values and input vectors are created from the original data set. Each data partition consists of k-lag input vectors systematically selected from all k-lag input vectors throughout the original training sample. An illustration of the data partitioning for 3-lag ensemble networks is presented in Table 1. The columns represent the data partition presented to each ensemble member while the composite ensemble is trained on the entire original training data set. Because one-step-ahead forecasting is considered, the next period value relative to the input vector is used as the current target. Note that with the systematic sampling scheme, the impact of the ensemble correlation effect is emphasized since different training samples with both non-overlapping input

Table 1 Illustration of the systematic data sampling with 3-lag input vectors

	Subsample 1		*Subsample 2*		*Subsample 3*	
Case	*Inputs*	*Target*	*Inputs*	*Target*	*Inputs*	*Target*
1	$\mathbf{x}_{1,1} = (x_1, x_2, x_3)$	$y_{1,1} = x_4$	$\mathbf{x}_{1,2} = (x_2, x_3, x_4)$	$y_{1,2} = x_5$	$\mathbf{x}_{1,3} = (x_3, x_4, x_5)$	$y_{1,3} = x_6$
2	$\mathbf{x}_{2,1} = (x_4, x_5, x_6)$	$y_{2,1} = x_7$	$\mathbf{x}_{2,2} = (x_5, x_6, x_7)$	$y_{2,2} = x_8$	$\mathbf{x}_{2,3} = (x_6, x_7, x_8)$	$y_{2,3} = x_9$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t/3$	$\mathbf{x}_{t/3,1} = (x_{t-5}, x_{t-4}, x_{t-3})$	$y_{t/3,1} = x_{t-2}$	$\mathbf{x}_{t/3,2} = (x_{t-4}, x_{t-3}, x_{t-2})$	$y_{t/3,2} = x_{t-1}$	$\mathbf{x}_{t/3,3} = (x_{t-3}, x_{t-2}, x_{t-1})$	$y_{t/3,3} = x_t$

vectors and target sets are used in building individual component networks in the ensemble. On the other hand, the ensemble as a whole sees fully overlapped patterns of the training data, which is important from a theoretical time series perspective. The systematic partitioning method in a sense, therefore balances ensemble methodology concerns with time series considerations.

The second data partitioning approach, called *serial* partitioning, divides the training time series into *k* mutually exclusive subsamples upon which *k* individual neural networks are built. These models are then combined to create a single ensemble forecaster. For example, if three partitions are used, then based on the chronological time sequence, the whole training set is partitioned into three disjoint subsamples of approximately the same size. The resulting ensemble in this case consists of three members. The first member is trained on the earliest one-third of the time series data, the second one is presented the middle-third of the data, and the last is trained on the most recent third of the data. Unlike in the systematic sampling scheme, serial ensemble members are presented fully overlapped and non-interrupted data. This is an important characteristic for time series modelling with regard to the development of the individual ensemble members. In addition, it is hoped that the effect of possible changes in the time series may be more effectively smoothed out as a result of the serial ensemble modelling approach.

Results

This section discusses the empirical results of the effectiveness of ensemble methods. The mean squared error (MSE) and the mean absolute error (MAE) are used to gauge the performance of the neural network models.

Table 2 reports the detailed results of the ensemble method for the training, validation and test sets using 50 different initial random weights. The results from the traditional keep-the-best (KTB) method are also listed for comparisons. With KTB, the neural network with the best training result among 50 training processes is selected and used for the prediction. It is generally observed that, as the neural network becomes more complex as the number of lags and/or the number of hidden nodes increase, training set MSE and MAE of the KTB method decrease while error measures in the validation set decrease first and then increase. This is the typical overfitting effect in neural network modelling, which often causes difficulty in model selection. On the other hand, no significant pattern in results is observed for the ensemble method because the averaging effect limits the impact of overfitting. Overall, from Table 2, we find that the basic ensembles formed by simply using different initial weights are not able to outperform the KTB method for all three samples across the 15 different neural network architectures.

Table 2 Results of ensemble with different random initial weights

		Training				*Validation*				*Test*			
		Ensemble		*KTB*		*Ensemble*		*KTB*		*Ensemble*		*KTB*	
Lag	*Hidden*	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*
1	2	0.0006537	0.0190663	0.0006493	0.0188754	0.0014607	0.0266757	0.0014442	0.0266700	0.0003812	0.0153665	0.0003761	0.0149920
1	4	0.0006502	0.0189457	0.0006490	0.0188773	0.0014533	0.0268277	0.0014471	0.0267961	0.0003785	0.0152271	0.0003775	0.0150958
1	8	0.0006507	0.0189166	0.0006489	0.0188876	0.0014525	0.0267409	0.0014462	0.0268241	0.0003789	0.0151690	0.0003770	0.0150890
2	2	0.0006538	0.0190675	0.0006500	0.0189212	0.0014562	0.0268244	0.0014470	0.0267879	0.0003831	0.0154245	0.0003781	0.0152016
2	4	0.0006493	0.0189060	0.0006486	0.0188913	0.0014466	0.0267606	0.0014442	0.0267670	0.0003786	0.0152141	0.0003772	0.0151629
2	8	0.0006505	0.0188939	0.0006480	0.0188756	0.0014496	0.0266658	0.0014417	0.0267774	0.0003803	0.0152155	0.0003769	0.0151435
3	2	0.0006528	0.0190501	0.0006484	0.0188744	0.0014512	0.0266994	0.0014429	0.0266883	0.0003820	0.0154124	0.0003783	0.0152229
3	4	0.0006491	0.0189093	0.0006462	0.0188558	0.0014469	0.0266429	0.0014439	0.0267205	0.0003795	0.0152672	0.0003791	0.0151255
3	8	0.0006503	0.0188994	0.0006462	0.0188273	0.0014450	0.0265006	0.0014480	0.0267281	0.0003797	0.0152152	0.0003793	0.0151492
4	2	0.0006501	0.0189770	0.0006439	0.0187793	0.0014792	0.0269165	0.0014684	0.0268775	0.0003875	0.0155358	0.0003830	0.0153133
4	4	0.0006471	0.0187889	0.0006423	0.0186519	0.0014748	0.0267916	0.0014548	0.0267533	0.0003837	0.0153257	0.0003772	0.0150808
4	8	0.0006485	0.0188004	0.0006429	0.0186503	0.0014761	0.0267744	0.0014673	0.0267126	0.0003862	0.0153494	0.0003791	0.0151473
5	2	0.0006490	0.0188992	0.0006427	0.0187544	0.0014824	0.0270535	0.0014694	0.0268935	0.0003862	0.0154545	0.0003867	0.0153783
5	4	0.0006464	0.0187661	0.0006433	0.0186933	0.0014738	0.0268554	0.0014545	0.0267208	0.0003837	0.0153185	0.0003821	0.0152951
5	8	0.0006505	0.0188402	0.0006412	0.0186355	0.0014855	0.0268398	0.0014598	0.0268665	0.0003847	0.0152041	0.0003822	0.0151145

The results of ensemble models created with different neural network architectures are given in Table 3. Two different combining schemes are considered. The first combines results from networks with the same number of input lags but different hidden nodes (ensemble across hidden) while the second uses the networks with different input nodes but the same number of hidden nodes (ensemble across lag). The average performance measures across different lags or hidden nodes for the KTB method are also reported for comparisons. Although the differences between the ensemble and KTB are relatively small, the ensemble method consistently gives better performance not only in the test sample, but also in both training and validation samples. It is important to note that, compared to results of ensembles across different hidden nodes, the ensembles across different lags have generally much better performance improvement than the KTB method. For example, from the test results in Table 3(c), we find that on average, the ensemble across hidden notes is 0.055% better in terms of MSE and 0.012% better in terms of MAE than KTB. However, these percentages are 0.20% on MSE and 0.053% on MAE, respectively, when comparing the basic ensemble across input nodes with the KTB method. Given the difficulty in exchange rate forecasting, these consistent albeit small performance improvements are encouraging.

As discussed earlier, the combining approach works better if individual predictions from members in an ensemble are less dependent. Given the apparent autocorrelations of time series data, the predictions from neural networks built from the same data are often highly correlated, which reduces the effectiveness of the ensemble method. Indeed, if the combination uses the same structured-network trained with different random initial weights, the correlation effect can be even stronger. Another problem with the random weight ensemble is that some of the initial weights may lead into 'bad' local minima. Therefore by averaging, the overall performance of the ensemble could be worse than the KTB model that uses the 'best' local minima solution to predict. Using different neural network architectures in the ensemble can reduce some of this correlation problem. The results from Table 3 clearly show that by combining

Table 3 (a) Training results of ensemble with different architectures. (b) Validation results of ensemble with different architectures. (c) Test results of ensemble with different architectures

	Ensemble		*KTB*	
Ensemble across hidden	*MSE*	*MAE*	*MSE*	*MAE*
(a)				
Lag 1	0.0006490	0.0188782	0.0006491	0.0188801
Lag 2	0.0006487	0.0188942	0.0006489	0.0188960
Lag 3	0.0006465	0.0188442	0.0006469	0.0188525
Lag 4	0.0006425	0.0186814	0.0006430	0.0186938
Lag 5	0.0006412	0.0186705	0.0006424	0.0186944
Across lag				
2 hidden	0.0006447	0.0188026	0.0006469	0.0188409
4 hidden	0.0006440	0.0187621	0.0006459	0.0187939
8 hidden	0.0006435	0.0187391	0.0006454	0.0187753
(b)				
Lag 1	0.0014458	0.0268223	0.0014458	0.0267634
Lag 2	0.0014442	0.0267774	0.0014443	0.0267774
Lag 3	0.0014446	0.0267103	0.0014449	0.0267123
Lag 4	0.0014626	0.0267801	0.0014635	0.0267811
Lag 5	0.0014601	0.0268219	0.0014612	0.0268269
Across lag				
2 hidden	0.0014516	0.0267636	0.0014544	0.0267834
4 hidden	0.0014466	0.0267111	0.0014489	0.0267515
8 hidden	0.0014493	0.0267378	0.0014526	0.0267817
(c)				
Lag 1	0.0003768	0.0150555	0.0003769	0.0150589
Lag 2	0.0003773	0.0151693	0.0003774	0.0151693
Lag 3	0.0003786	0.0151638	0.0003789	0.0151659
Lag 4	0.0003794	0.0151789	0.0003798	0.0151805
Lag 5	0.0003826	0.0152323	0.0003837	0.0152626
Across lag				
2 hidden	0.0003792	0.0152143	0.0003804	0.0152216
4 hidden	0.0003782	0.0151468	0.0003786	0.0151520
8 hidden	0.0003783	0.0151171	0.0003789	0.0151287

different models either with different hidden nodes or with different input nodes, the ensembles are able to perform better than the KTB approach on the average. Furthermore, since the number of input nodes determines, to a large extent, the autocorrelation structure of the time series, it seems obvious that the predictions from networks with different numbers of hidden nodes but the same number of lags should have higher correlation than those with different input lags but the same hidden nodes. This may explain why the combination of networks with different input nodes is more effective than the ensemble with different hidden nodes.

Seeking to reduce the harmful correlation effect among ensemble members further, we created *partition* ensembles with different training data, as discussed in the last section. With these disjoint training samples, individual member networks should exhibit much less correlation. Notice that since only a portion of the data is used for the individual ensemble member development, each individual member may not predict well as those trained on the entire data set. However, the ensemble as a whole should have the benefit of reduced correlation and increased modelling power. To see the effect of using only one portion of the data to develop the ensemble, we first investigate the basic ensemble approach of using 50 random weight initializations to generate the individual ensemble members trained on a single *systematic portion* of the training data. This method eliminates all training data overlap, which may be important in reducing ensemble correlation due to training on many similar data points. The ensemble validation and test results using only the first portion of data in column 1 of Table 1 are reported in Table 4. It shows that when only a portion of the total available sample data is used to develop both the individual ensemble members and the overall ensemble, they are, in general, not as effective as the KTB model. Similar results are obtained using other portions (columns) of the systematic sample.

Next we consider the performance of the individual *serial* ensemble members. The whole training data is partitioned into three sub-samples, each with approximately one-third of the original training series. The first ensemble member is trained on the earliest 260 observations in the time series, the second member on the next 261 data points, while the third on the last 261 data points. With each subsample, the 15 different neural architectures mentioned earlier are applied. Figure 2 illustrates the performance of the three individual *serial partition* ensemble members using the ratios of MSE of the serial ensemble members to the basic ensemble. It is clear that the individual ensemble members perform worse than the basic ensemble formed with the full training sample as the ratio is consistently above one. Note also that the variability among different network architectures is much higher in the first partition than in partitions 2 and 3. This suggests that ensembles with older data (such as partition 1) have worse forecasting performance than ensembles with more recent data (such as partitions 2 and 3). Examining Figure 1 reveals that almost all observations in the first subsample are above the level of the test sample, while the averages of the second and third sample are closer to the average level of the test sample. Figure 1 also suggests that some change in the time series generating process parameters may be present.

While the individual ensemble members show decreased performance relative to the basic ensemble approach and KTB method, attention is now turned to evaluating the performance of the overall partition ensembles. Results of the complete ensembles based on partitions of the training data are presented in Tables 5 and 6. The systematic

Table 4 The effect of single systematic partition of training data on ensemble

		Validation				*Test*			
		Ensemble		*KTB*		*Ensemble*		*KTB*	
Lag	*Hidden*	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*
1	2	0.0014607	0.0266757	0.0014442	0.0266700	0.0003812	0.0153665	0.0003761	0.0149920
1	4	0.0014533	0.0268277	0.0014471	0.0267961	0.0003785	0.0152271	0.0003775	0.0150958
1	8	0.0014525	0.0267409	0.0014462	0.0268241	0.0003789	0.0151690	0.0003770	0.0150890
2	2	0.0014583	0.0270137	0.0014470	0.0267879	0.0003949	0.0157477	0.0003781	0.0152016
2	4	0.0014484	0.0269478	0.0014442	0.0267670	0.0003871	0.0155470	0.0003772	0.0151629
2	8	0.0014465	0.0268448	0.0014417	0.0267774	0.0003857	0.0154914	0.0003769	0.0151435
3	2	0.0014697	0.0271265	0.0014429	0.0266883	0.0003758	0.0151277	0.0003783	0.0152229
3	4	0.0014671	0.0270691	0.0014439	0.0267205	0.0003745	0.0150053	0.0003791	0.0151255
3	8	0.0014712	0.0269950	0.0014480	0.0267281	0.0003757	0.0149880	0.0003793	0.0151492
4	2	0.0014848	0.0271231	0.0014684	0.0268775	0.0003828	0.0152889	0.0003830	0.0153133
4	4	0.0014735	0.0268471	0.0014548	0.0267533	0.0003796	0.0150929	0.0003772	0.0150808
4	8	0.0014944	0.0269939	0.0014673	0.0267126	0.0003851	0.0152165	0.0003791	0.0151473
5	2	0.0016189	0.0271401	0.0014694	0.0268935	0.0004089	0.0158738	0.0003867	0.0153783
5	4	0.0015783	0.0268482	0.0014545	0.0267208	0.0004019	0.0175125	0.0003821	0.0152951
5	8	0.0015841	0.0269127	0.0014598	0.0268665	0.0004040	0.0156809	0.0003822	0.0151145

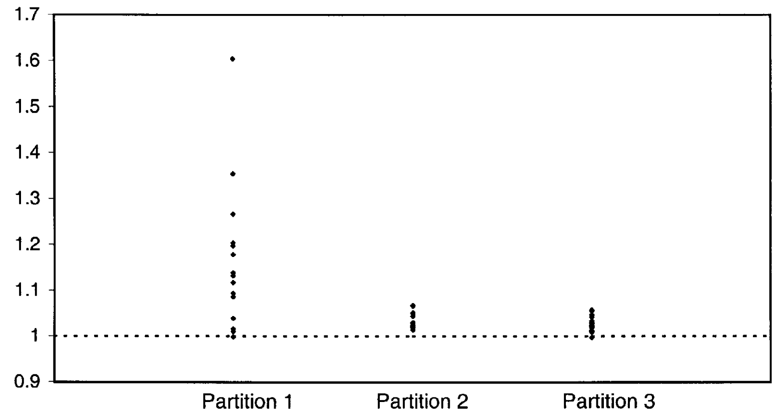

Figure 2 The effect of single serial partition data on ensemble: test set MSE comparisons.

ensemble method, which combines neural networks constructed on the systematically partitioned samples, is compared with the serial ensemble method, which is based on the networks with three serial partitioned samples. Table 5 reports the results with ensemble networks trained with different initial weights. Several observations can be made from Table 5. First, in nearly all cases, both systematic and serial ensembles have lower overall forecasting errors than the basic ensemble in Table 2. This holds true in both validation and test samples. Hence ensemble models created with individual members trained on different portions of the data can be more effective in forecasting than an ensemble trained on the same data. Secondly, the serial ensemble performed much better than the systematic ensemble. This may be attributed to the fact that the serial ensemble model uses non-interrupted, yet overlapping, training patterns in model building while disjoint, non-overlapping training patterns are used in developing the systematic ensemble model. For time series modelling, it is perhaps more important to have continuous training patterns in the training data set in order to better capture the autocorrelated data generating process. The serial partitioning of the training data may decorrelate ensemble predictions and thus lead to the observed performance improvement. Thirdly, using validation MSE as the model selection criterion, the best model with the serial ensemble has 3 lags and 8 hidden nodes while for the systematic ensemble, it has 2 lags and 4 hidden nodes. Compared to the best model with the basic ensemble (also 3 lags and 8 hidden nodes) in Table 2, the models selected utilizing the partition ensemble approaches can significantly outperform that with the basic ensemble on both MSE and MAE in out-of-sample forecasting.

Table 6 shows the validation and test results of both systematic and serial ensembles using different neural network architectures. We find that the serial ensemble

Table 5 Comparison between systematic ensemble and serial ensemble with different initial weights

		Validation				*Test*			
		Systematic ensemble		*Serial ensemble*		*Systematic ensemble*		*Serial ensemble*	
Lag	*Hidden*	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*
1	2	0.0014607	0.0266757	0.0014282	0.0266346	0.0003812	0.0153665	0.0003776	0.0149010
1	4	0.0011453	0.0268277	0.0014273	0.0268184	0.0003785	0.0152271	0.0003756	0.0148804
1	8	0.0011453	0.0267409	0.0014309	0.0268890	0.0003789	0.0151690	0.0003770	0.0151058
2	2	0.0011441	0.0267556	0.0014294	0.0268042	0.0003781	0.0152038	0.0003772	0.0148838
2	4	0.0011435	0.0267154	0.0014260	0.0267932	0.0003777	0.0151130	0.0003761	0.0149001
2	8	0.0014379	0.0267320	0.0014285	0.0268865	0.0003778	0.0151642	0.0003757	0.0149925
3	2	0.0014398	0.0266119	0.0014236	0.0267395	0.0003775	0.0152056	0.0003811	0.0149363
3	4	0.0014363	0.0265637	0.0014286	0.0268343	0.0003792	0.0151950	0.0003753	0.0148618
3	8	0.0014474	0.0266173	0.0014203	0.0267992	0.0003815	0.0153111	0.0003757	0.0149892
4	2	0.0014710	0.0268427	0.0014291	0.0268362	0.0003801	0.0152041	0.0003836	0.0150390
4	4	0.0014633	0.0267260	0.0014423	0.0270462	0.0003814	0.0152628	0.0003809	0.0149934
4	8	0.0014609	0.0267706	0.0014301	0.0269375	0.0003818	0.0152372	0.0003786	0.0148919
5	2	0.0014665	0.0269102	0.0014257	0.0267586	0.0003826	0.0151210	0.0003940	0.0150489
5	4	0.0014825	0.0270198	0.0014327	0.0269357	0.0003846	0.0152266	0.0003785	0.0149271
5	8	0.0014746	0.0269748	0.0014217	0.0267758	0.0003831	0.0151919	0.0003916	0.0154576

Table 6 Comparison between systematic ensemble and serial ensemble with different architectures

	Validation				*Test*			
	Systematic ensemble		*Serial ensemble*		*Systematic ensemble*		*Serial ensemble*	
Ensemble across hidden	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*	*MSE*	*MAE*
Lag 1	0.0014458	0.0268223	0.0014288	0.0267807	0.0003768	0.015 0555	0.0003767	0.0149624
Lag 2	0.0014382	0.0267335	0.0014280	0.0268280	0.0003771	0.0151523	0.0003763	0.0149029
Lag 3	0.0014410	0.026 5952	0.0014242	0.0267910	0.0003782	0.0152372	0.0003774	0.0149291
Lag 4	0.0014648	0.0267870	0.0014338	0.0269400	0.0003770	0.0152315	0.0003810	0.0149748
Lag 5	0.0014742	0.0269683	0.0014267	0.0268234	0.0003822	0.0151711	0.0003880	0.0151445
Across lag								
2 hidden	0.0014485	0.0267177	0.0014272	0.0267546	0.0003779	0.0151316	0.0003827	0.0149618
4 hidden	0.0014489	0.0267070	0.0014314	0.0268856	0.0003781	0.0151718	0.0003773	0.0149126
8 hidden	0.0014497	0.0267124	0.0014263	0.0268576	0.0003776	0.0151945	0.0003797	0.014 2074

model performs about the same as the systematic ensemble. In particular, both test MSE and MAE now indicate that, by combining networks with different input nodes or different hidden nodes, the forecasting ability of the systematic ensemble method can be significantly improved. Not only does this partition ensemble approach outperform the basic ensemble, but it also has better average forecasting performance than the KTB method (see Tables 3(*b*) and (*c*)).

To this point, only the forecasting performance of various neural network ensemble methods relative to the single best neural network predictor has been conducted. To assess the statistical significance between neural networks and the traditional random walk benchmark as well as the market timing ability of the ensembles, we employ the Diebold and Mariano[24] (DM) test and the Pesaran and Timmermann[25] (PT) test. In the DM test, only the squared loss function is presented as using the absolute error loss function yields similar results. The KTB method and the best ensemble approach, the serial partition ensemble, are each compared to the random walk model for both the validation and test sets.

Table 7 reports the results from the Diebold–Mariano test. Several observations are noted. First, when compared to the random walk predictions, the serial ensemble performs better than the KTB in nearly every case. In fact, the serial ensemble shows a consistent mild to moderate significance with a median *P*-value of 0.1607 in the validation sample while the median *P*-value for the KTB is 0.4052. This improvement of the serial ensemble compared to the KTB also carries over to the test sample. Second, although serial ensembles outperform the random walk model with some significance in the validation set, the significant difference is not seen between these two models in the test sample. The degradation in the significance of

Table 7 Diebold and Mariano test for predictive significance relative to random walk model

		Validation				*Test*			
		KTB		*Serial ensemble*		*KTB*		*Serial ensemble*	
Lag	*Hidden*	*Statistic*	P-*value*	*Statistic*	P-*value*	*Statistic*	P-*value*	*Statistic*	P-*value*
1	2	−0.9142	0.1803	−1.0030	0.1579	0.1811	0.5719	0.1766	0.5701
1	4	−1.0875	0.1384	−0.9916	0.1607	0.5356	0.7039	0.0411	0.5164
1	8	−1.0142	0.1552	−1.0473	0.1475	0.4235	0.6640	0.4305	0.6666
2	2	−0.2399	0.4052	−0.9860	0.1621	0.5585	0.7117	0.1455	0.5578
2	4	−0.4538	0.3250	−1.0375	0.1498	0.4566	0.6760	0.0928	0.5370
2	8	−0.6420	0.2604	−1.0105	0.1561	0.3931	0.6529	0.1012	0.5403
3	2	−0.5087	0.3055	−1.0783	0.1404	0.5466	0.7077	0.3461	0.6354
3	4	−0.4652	0.3209	−0.9697	0.1661	0.8411	0.7999	0.0128	0.5051
3	8	−0.1525	0.4394	−1.1431	0.1265	1.0151	0.8450	0.1134	0.5451
4	2	0.8900	0.8133	−0.7586	0.2240	1.2566	0.8956	0.5035	0.6927
4	4	0.2472	0.5976	−0.2891	0.3863	0.4098	0.6590	0.3797	0.6479
4	8	0.6056	0.7276	−0.6792	0.2485	0.8252	0.7954	0.3116	0.6223
5	2	0.6515	0.7426	−0.9263	0.1771	1.3059	0.9042	0.7779	0.7817
5	4	0.1677	0.5666	−0.7221	0.2351	1.5126	0.9348	0.2428	0.5959
5	8	0.3984	0.6548	−1.0234	0.1531	0.7824	0.7830	1.0700	0.8577

both the KTB and ensemble models relative to the random walk in the test set is clear judging by the *P*-values. Although it may not be surprising to obtain results that do not meet statistical significance thresholds in out-of-sample forecasting comparison given the difficulty suggested in the literature, a further look at the data may reveal why there is such an apparent shift in significance between the validation and test sets in this particular application. In Figure 1, we notice that the pattern of the time series appears changed in the test set timeframe relative to that in the training and validation time periods. The exchange rate series is quite stable during the test set time period, which is the most favorable environment for the no change forecasting of the random walk model. Nevertheless, it is encouraging to find that the serial ensemble consistently provides a better buffering prediction model than the KTB method does against possible changes in the time series generating process or its parameters. Finally, we find that the DM statistic and the associated *P*-value for the serial ensemble are much less variable than those for the KTB across all 15 neural network architectures for both validation and test sets. This conforms to the theoretical expectation of the ensemble development that improvement in generalization capability results primarily from a decrease in the variance component of overall prediction error.

Table 8 contains the results of the Pesaran and Timmermann test for significance in market timing ability. The PT test evaluates a model's ability to correctly predict increases or decreases in the exchange rates. The efficacy of both the KTB and the serial ensemble methods for market timing is less clear than that for the DM test. This may be because the neural networks were not specifically designed to identify changes in the direction of the exchange rates. Neural network market timing performance might be improved by training on the first-order differences in the series rather than the exchange rates. In this way, the targets in the training phase will include both positive and negative values, thereby emphasizing the importance of market timing ability.

Conclusions

Neural network ensembles can be an effective approach to improving neural network performance over a single 'best' network model. Although neural ensembles have been studied and applied for pattern classification problems, few applications have been reported in forecasting applications. This paper presents a detailed investigation of the effectiveness of neural network ensembles for exchange rate forecasting. Results show that by appropriately combining different neural networks, forecasting accuracy of individual networks can be largely improved. Although our ensemble methods show considerable advantages over the traditional KTB approach, they do not have significant improvement compared to the widely used random walk model in exchange rate forecasting.

In this study, several strategies to form the neural ensemble models are investigated. These include neural networks trained with different initial random weights, networks with different architectures, and networks trained with different data. Our results show that different approaches to forming ensembles for time series forecasting have quite different effects on the forecasting results. Neural network ensembles created by simply varying the starting random weights are not as competent as the traditional keep-the-best (KTB) model. Therefore this method of ensemble forecasting may not be effective for time series forecasting problems. On the other hand,

Table 8 Persaran and Timmerman test for significance of market timing ability

		Validation				*Test*			
		KTB		*Serial ensemble*		*KTB*		*Serial ensemble*	
Lag	*Hidden*	*Statistic*	P-*value*	*Statistic*	P-*value*	*Statistic*	P-*value*	*Statistic*	P-*value*
1	2	0.1911	0.5758	0.0423	0.5169	−0.8756	0.1906	−0.8756	0.1906
1	4	0.1118	0.5445	0.0423	0.5169	−0.8756	0.1906	−0.8756	0.1906
1	8	−0.0136	0.4946	0.1962	0.5778	−0.8756	0.1906	2.2425	0.9875
2	2	0.1024	0.5408	0.0423	0.5169	−0.2612	0.3970	−0.8756	0.1906
2	4	0.2823	0.6111	0.0423	0.5169	2.1711	0.9850	−0.8756	0.1906
2	8	−0.1255	0.4501	−0.7319	0.2321	1.4767	0.9301	0.1019	0.5406
3	2	1.1903	0.8830	0.5922	0.7231	1.3490	0.9113	−0.8756	0.1906
3	4	1.1903	0.8830	−0.5076	0.3059	0.7379	0.7697	−0.8756	0.1906
3	8	0.5588	0.7119	0.2847	0.6121	0.1829	0.5726	−0.8756	0.1906
4	2	−0.6650	0.2530	−0.0774	0.4692	0.1044	0.5416	−0.8756	0.1906
4	4	−0.4429	0.3289	−0.5808	0.2807	0.7818	0.7828	−0.8756	0.1906
4	8	−0.4429	0.3289	−0.6166	0.2687	1.7965	0.9638	−1.2453	0.1065
5	2	1.4595	0.9278	−0.0202	0.4919	−1.1683	0.1213	−0.8756	0.1906
5	4	0.8394	0.7994	0.4259	0.6649	−0.7183	0.2363	−0.8756	0.1906
5	8	0.4830	0.6855	−0.9311	0.1759	0.7818	0.7828	−0.8756	0.1906

ensemble models created with different neural network structures consistently perform well in a variety of situations and, hence, appear to be a preferred way to improve forecasting performance. Furthermore, the results indicate that it is more beneficial to include networks with different numbers of lags in an ensemble than to include models with different numbers of hidden nodes. This conclusion matches the expectation that it is the number of lags in the neural network model that largely determines the autocorrelation structure of a time series.

Theoretical work in neural network ensembles has concluded that they work better if different models in the ensemble disagree with each other strongly.[15,26] That is, if all models in the ensemble are independent of each other or the prediction errors are uncorrelated, then using the ensemble method can improve forecasting accuracy significantly. Moreover, if the negative covariance can be created among individual forecasts, the ensemble's improvement on performance can be even more significant.[27] To reduce the correlation effect, we have proposed and evaluated two ensemble models based on different data partitioning methods. The systematic ensemble is formed by networks built on non-overlapping systematic subsamples from the original training series while the serial ensemble is constructed by networks developed on chronologically separated subsamples. Empirical findings from this study indicate that partition ensembles based on different samples can significantly outperform the basic ensemble based on the same training sample. Moreover, the serial ensemble model appears to be a more promising approach in time series forecasting applications as demonstrated in this exchange rate example.

Accurate forecast of the exchange rate movement is of fundamental importance to multinational firms and portfolio managers in managing a major risk inherent in international transactions and thereby enabling better strategic and cross-border investment decisions. On the other hand, exchange rate forecasting is a difficult problem as numerous empirical studies suggest. Previous neural network research has yielded mixed findings on the out-of-sample predictability of the exchange rate. One of the possible reasons for the inconclusive results is the variability associated with the single KTB approach most commonly used in the literature. Although the purpose of this paper is primarily to demonstrate the effectiveness of the ensemble approach over the KTB practice, a comparison with the benchmark random walk model is included for completeness. The results indeed show that the partition ensemble models have a consistent and significant improvement over the single KTB modelling approach. The ensemble results also show moderately significant improvement over the random walk model in the validation sample that degrades in the test sample period. In the paper it is shown that this may result from a possible shift in the time series and that the stable pattern observed in this particular situation highly favours the no-change prediction of the random walk model over other prediction methods. Nevertheless, it is demonstrated that ensembles can be more robust against possible pattern changes in a time series and hence able to provide more reliable forecasts in a wider array of forecasting situations than the commonly used KTB approach.

Although this study evaluates several potential ensemble models for exchange rate forecasting, many other ensemble methods can be considered. For example, one potential method is based on bootstrapping samples randomly generated from the original whole training time series. While computationally expensive, ensemble models based on bootstrapping samples may provide further insights and evidence on the effectiveness of ensemble method for out-of-sample forecasting. Research efforts should also be devoted to the methods that can further reduce the correlation effect in combining neural networks and to quantifying the impact that shifts in the data generation parameters have on the various approaches. Simulation and experimental design methodology should prove useful and necessary in these endeavours.

Acknowledgements—We would like to thank Dr John Ranyard and two anonymous referees for their helpful comments which lead to important improvement of the presentation and substance. Financial support for the first author from the J Mack Robinson College of Business at Georgia State University is gratefully acknowledged.

References

1 Hansen LK and Salamon P (1990). Neural network ensembles. *IEEE Trans Pattern Anal Machine Intell* **12**: 993–1001.

2 Perrone MP and Cooper L (1993). When networks disagree: ensemble method for hybrid neural networks. In: RJ Mammone (ed). *Neural Networks for Speech and Image Processing*. Chapman and Hall: London, pp 126–142.

3 Clemen R (1989). Combining forecasts: a review and annotated bibliography with discussion. *Int J Forecast* **5**: 559–608.

4 Makridakis S *et al* (1982). The accuracy of extrapolation (time series) methods: results of a forecasting competition. *J Forecast* **1**: 111–153.

5 Meese RA and Rogoff K (1983a). Empirical exchange rate models of the seventies: do they fit out of sample? *J Int Econ* **14**: 3–24.

6 Meese RA and Rogoff K (1983b). The out of sample failure of empirical exchange rate models: sampling error or misspecification? In: Frankel J (ed). *Exchange Rates and International Economics*. University of Chicago Press: Chicago, pp 67–105.

7 Krugman P (1988). Target zones and exchange rate dynamics. NBER working papers 2481.

8 Mizrach B (1992). Multivariate nearest-neighbour forecasts of EMS exchange rates. *J Appl Econ* **7**(Suppl): S151–S164.

9 Meese RA and Rose A (1991). An empirical assessment of non-linearities in models of exchange rate determination. *Rev Econ Stud* **58**: 601–619.

10 Chiarella C, Peat M and Stevenson M (1994). Detecting and modelling nonlinearity in flexible exchange rate time series. *Asia Pac J Mngmnt* **11**: 159–186.

11 Brooks C (1996). Testing for non-linearity in daily sterling exchange rates. *Appl Financ Econ* **6**: 307–317.

12 Kuan CM and Liu T (1995). Forecasting exchange rates using feedforward and recurrent neural networks. *J Appl Econ* **10**: 347–364.

13 Borisov AN and Pavlov VA (1995). Prediction of a continuous function with the aid of neural networks. *Auto Control Comp Sci* **29**: 39–50.

14 Hann TH and Steurer E (1996). Much ado about nothing? Exchange rate forecasting: Neural networks vs. Linear models using monthly and weekly data. *Neurocomputing* **10**: 323–339.

15 Zhang X and Hutchinson J (1994). Simple architectures on fast machines: practical issues in nonlinear time series prediction. In: Weigend AS and Gershenfeld NA (eds). *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addition-Wesley: Reading, MA, pp 219–241.

16 Geman S, Bienenstock E and Doursat T (1992). Neural networks and the bias/variance dilemma, *Neural Comput* **5**: 1–58.

17 Makridakis S (1989). Why combining works? *Int J Forecast* **5**: 601–603.

18 Benediktsson J, Sveinsson J, Ersoy O and Swain P (1994). Parallel consensual neural networks with optimally weighted outputs. In: *Proceedings of the World Congress on Neural Networks III.* Lawrence Erlbaum Associates: Mahwah, NJ, USA, pp 129–137.

19 Hashem S (1997). Optimal linear combination of neural networks. *Neural Networks*, **10**: 599–614.

20 Zhang G, Patuwo E and Hu YM (1998). Forecasting with artificial neural networks: the state of the art. *Int J Forecast* **14**: 35–62.

21 Diebold FX and Nason JA (1990). Nonparametric exchange rate prediction? *J Int Econ* **28**: 315–332.

22 Zhang G (1998). Linear and nonlinear time series forecasting with artificial neural networks. PhD dissertation. Kent State University, Kent, OH.

23 Rumelhart DE, Hinton GE and Williams RJ (1986), Learning internal representations by error propagation. In: Rumelhart DE, McClelland JL and the PDP Group (eds). *Parallel Distributed Processing: Exploration in the Microstructure of Cognition. Vol. 1: Foundations*. MIT Press; Cambridge, MA, pp. 318–362.

24 Diebold FX and Mariano RS (1995). Comparing predictive accuracy, *J Bus Econ Statist* **13**: 253–263.

25 Pesaran MH and Timmermann A (1992). A simple nonparametric test of predictive performance. *J Bus Econ Statist* **10**: 461–465.

26 Krogh A and Vedelsby J (1995). Neural network ensembles, cross validation, and active learning. In: Tesauro G, Touretzky D and Leen T (eds). *Advances in Neural Information Processing Systems 7.* MIT Press: Cambridge, MA, pp 231–238.

27 Munro P and Parmanto B (1997). Competition among networks improves committee performance. In: Mozer M, Jordan MI and Petsche T (eds). *Advances in Neural Information Processing Systems* **9**, MIT Press: Cambridge, MA, pp 592–598.

Received September 1999;
accepted December 2000 after three revisions