Neural-Network-based Metamodeling for Financial Time Series Forecasting

Kin Keung Lai [1 2] Lean Yu [2 3] Shouyang Wang [3] Chengxiong Zhou [3]

[1] College of Business Administration, Hunan University, Changsha, China
[2] Department of Management Sciences, City University of Hong Kong, Hong Kong
[3] Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

Abstract

In the financial time series forecasting field, the problem that we often encountered is how to increase the predict accuracy as possible using the noisy financial data. In this study, we discuss the use of supervised neural networks as the metamodeling technique to design a financial time series forecasting system to solve this problem. First of all, a cross-validation technique is used to generate different training subsets. Based on the different training subsets, the different neural predictors with different initial conditions or training algorithms is then trained to formulate different forecasting models, i.e., base models. Finally, a neural-network-based metamodel can be produced by learning from all base models so as to improve the model accuracy. For verification, two real-world financial time series is used for testing.

Keywords: Metamodeling, neural network, financial time series forecasting, cross-validation.

1. Introduction

Financial market is a complex evolved dynamic market with high volatility. Due to its irregularity, financial time series forecasting is regarded as a rather challenging task. For traditional statistical methods, it is extremely difficult to capture the irregularity. In the last decades, many emerging artificial intelligent techniques, such as neural networks, were widely used in the financial time series forecasting and obtained good prediction performance [1].

However, neural networks are a kind of unstable learning methods, i.e., small changes in the training set and/or parameter selection can produce large changes in the predicted output. The results of many experiments have shown that the generalization of single neural network is not unique. In other words, the neural network's results are not stable. Even for some simple problems, different structures of neural networks (e.g., different number of hidden layers, different number of hidden nodes and different initial conditions) result in different patterns of network generalization. In addition, even the most powerful neural network model still cannot cope well when dealing with complex data sets containing some random errors or noisy data. Thus, the performance for these data sets may not be as good as expected [2-3].

To overcome the drawback of single neural predictor, metalearning-based metamodeling is proposed to solve the problem. Metalearning [4], which is defined as learning from learned knowledge, provides a promising solution to these challenges. The basic idea is to use intelligent learning algorithms to extract knowledge from some data subsets and then use the knowledge from these individual learning algorithms to create a unified body of knowledge that well represents the entire knowledge about overall data set. Therefore metalearning seeks to compute a metamodel that integrates in some principled fashion the separately learned models to boost overall predictive accuracy. Actually, a metamodel is a model of a model. It can be viewed as a transformation model that converts the output in one world view to another world view [5]. In this study, a triple-phase neural-network-based meta-modeling technique is proposed. First of all, a cross-validation technique is used to generate different training sets, validation set and testing set. Based on the different training sets, the different neural network models with different initial conditions or different training algorithms is then trained to formulate different neural network prediction models, i.e., base forecasting models. These base models' training processes do not affect the overall execution efficiency of time series forecasting system due to its parallel mechanism of metalearning. That is, the base model can be formulated in a parallel way. In the third phase, a neural-network-based metamodel can be produced by learning from all base forecasting models. The main reason that neural network is selected as a metamodeling tool is their ability to approximate any arbitrary function arbitrarily well and provide flexible mapping between inputs and outputs [6].

The main motivation of this study is to take full advantage of the flexible mapping capability of neural network and inherent parallelism of metalearning to design a powerful financial time series forecasting system. The rest of this study is organized as follows. Section 2 presents a neural-network-based meta-modeling process in detail. For verification, two real-world time series is used for testing in Section 3. And Section 4 concludes the paper.

2. Neural Network Metamodeling

As Section 1 mentioned, metalearning [4], which is defined as learning from learned knowledge, is an emerging technique recently developed to construct a metamodel that deals with the problem of computing a metamodel from large databases. Broadly speaking, learning is concerned with finding a model $f = f_a[i]$ from a single training set $\{TR_i\}$, while metalearning is concerned with finding a metamodel $f = f_a$ from several training sets $\{TR_1, TR_2, \ldots, TR_n\}$, each of which has an associated model (i.e., base model) $f = f_a[i]$ (i =1, 2, …, n). The n base models derived from the n training sets may be of the same or different types. Similarly, the metamodel may be of a different type than some or all of the component models. Also, the metamodel may use data from a meta-training set (*MT*), which are distinct from the data in the single training set TR_i.

Generally, the main process of metamodeling is first to generate a number of independent models by applying different learning algorithms to a collection of data sets in parallel. The models computed by learning algorithms are then collected and combined to obtain a global model or metamodel. Fig. 1 shows a generic metamodeling process, in which a global model or a metamodel is obtained on *Site Z*, starting from the original data set *DS* stored on *Site A*.

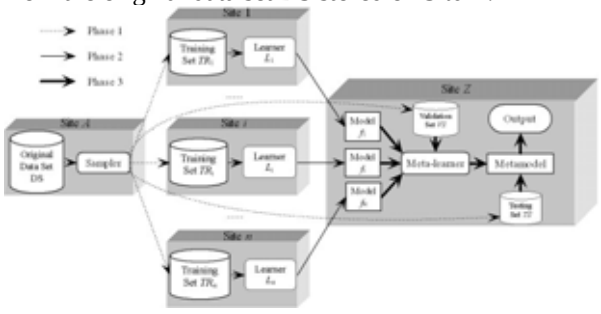

Fig. 1: The generic metalearning process

As can be seen from Fig. 1, the generic meta-modeling process consists of three phases, which can be described as follows.

Phase I: on *Site A*, training sets TR_1, TR_2, …, TR_n, validation set *VS* and testing set *TS* are extracted from *DS*. Then TR_1, TR_2, …, TR_n, *VS* and *TS* are moved from *Site A* to *Site* 1, *Site* 2, …, *Site n* and to *Site Z*.

Phase II: on each *Site i* (i = 1, 2, …, n) the different models f_i is trained from TR_i by the different learners L_i. Then each f_i is moved from *Site i* to *Site Z*. It is worth noting that the training process of n different model can be implemented in parallel.

Phase III: on *Site Z*, the f_1, f_2, …, f_n models are combined and validated on *VS* and tested on *TS* by the meta-learner *ML* to produce a metamodel.

From the generic metamodeling process, we can find that there are still three main problems to be further addressed, i.e., (a) how to create n training data sets TR_i, validation set *VS* and testing set *TS* from the original data set *DS*; (b) how to create different base models f_i with different learners L_i for the neural network; and (c) how to formulate a metamodel with the different results produced by different models f_i.

2.1. Data Set Partition

In order to create different neural predictors, different training subsets should be generated. Common methods for different training subsets include bagging [7], noise injection [8] and cross-validation [9]. In view of the particularity of neural network predictors, cross-validation technique is selected.

Cross-validation [9] is a tool borrowed from statistics. For example, the available data set is randomly divided into m disjoint subsets. By selecting one of these subsets as a testing set, the remainders are rejoined as its corresponding training set. For this case, m numbers of overlapping training set and m independent testing sets are obtained. As each training set is different somehow, the errors they generate after training are expected to fall in different local error minima and thus lead to different results. Model performance is measured on the corresponding testing set. This approach ideally requires that the number of partitions is the same as the number of data exemplars. In practice, ten-fold and twenty-fold cross-validations are adopted, i.e., m is equal to 10 and 20, respectively.

2.2. Single Model Creation

With the work about bias-variance trade-off [10], a metamodel consisting of diverse models (i.e., base models) with much disagreement is more likely to have a good performance. Therefore, how to create the diverse model is the key to the creation of an effective metamodel. For neural network, there are four methods for generating diverse models.

(1) Initializing different starting weights for each neural network models.

(2) Training neural network models with different training subsets.

(3) Varying the architecture of neural network, e.g., changing the different numbers of layers or different numbers of nodes in each layer.

(4) Using different training algorithms, such as the back-propagation and Bayesian regression algorithms.

In this study, the individual neural network forecasting models with different training subsets generated by previous phase are therefore used as base learner L_1, L_2, …, L_n, as illustrated in Fig. 1. Through training, base models f_1, f_2, …, f_n can be formulated in a parallel way.

2.3. Metamodel Generation

As Fig. 1 illustrated, the initial data set is first divided into subsets, and then these subsets are input to the different individual models which could be executed concurrently. These individual models are called "base models'. In this phase, the main task is to generate a metamodel to assimilate knowledge from different base models. Basically, the metamodel generation process is described as follows.

Based upon different training sets, the base models can be generated in the previous phase. Using the validation set *VS* and testing set *TS*, the performance of the base models can be assessed. Afterwards, the whole validation set *VS* is applied to these base models and corresponding results produced by these base models are used as input of another individual neural network model. By validation and verification, a metamodel can be generated using the results generated by the base model as input, combined with their targets or expected values. In this sense, neural network learning algorithm is used as a meta-learner (*ML*) shown in Fig. 1 for metamodel generation.

In this process of metamodel generation, another neural network model used as metal-learner (ML) can be viewed as a nonlinear information processing system that can be represented as:

$$\hat{f} = f(\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_n) \quad (1)$$

where $(\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_n)$ is the output of individual neural network predictors, $\hat{f}$ is the output of the neural-network-based metamodel by integrating the outputs of all individual neural network predictors, $f(\cdot)$ is nonlinear function determined by another neural network learning algorithm.

3. Experimental Analysis

In the experiments, one stock index, S&P 500, and one foreign exchange rate, euros against US dollars (EUR/USD), are used for testing purpose. The historical data are daily and are obtained from Wharton Research Data Service (WRDS), provided by Wharton School of the University of Pennsylvania. The entire data set covers the period from January 1 2000 to December 31 2004 with a total of 1256 observations. The data sets are divided into two periods: the first period covers January 1 2000 to December 31 2003 with 1004 observations, while the second period is from January 1 2004 to December 31 2004 with 252 observations. The first period, which is assigned to in-sample estimation, is used for network learning, i.e., training set. The second period, which is reserved for out-of-sample evaluation, is used for validation, i.e., testing set. For space limitation, the original data are not listed in this paper, and detailed data can be obtained from the WRDS.

In order to increase model accuracy for financial time series forecasting, ten different training subsets are generated by cross-validation technique, i.e., m = 10. Using these different training subsets, different neural network base models with different initial weights are presented. For neural network base models, a three-layer back-propagation neural network with 10 TANSIG neurons in the hidden layer and one PURELIN neuron in the output layer is used. The network training function is the TRAINLM. For the neural-network-based metamodel, a similar three-layer back-propagation neural network with 10 input neurons, 8 TANSIG neurons in the hidden layer and one PURELIN neuron in the final layer is adopted for metamodel generation. Besides, the learning rate and momentum rate is set to 0.1 and 0.15. The accepted average mean squared error is 0.05 and the training epochs are 1600. The above parameters are obtained by trial and error.

To evaluate the performance of the proposed neural-network-based metamodel, several typical time series forecasting models, random walk (RW) model, auto-regression integrated moving average (ARIMA), exponential smoothing (ES) model and individual back-propagation neural network (BPNN) model, are selected as benchmarks. For comparison, the root mean squared error (*RMSE*) and D_{stat} [1] are used the evaluation criteria and corresponding results are reported in Table 1.

Table 1 The comparison of prediction results

Forecasting Model	S&P 500		EUR/USD	
	RMSE	D_{stat}(%)	*RMSE*	D_{stat}(%)
RW	14.37	54.36	0.0066	58.69
ARIMA	9.63	63.43	0.0041	68.47
ES	10.51	65.71	0.0048	66.86
BPNN	7.56	72.58	0.0052	74.35
Metamodel	3.14	86.34	0.0038	87.28

For the S&P 500, the proposed neural network metamodel outperforms the other four typical time series forecasting models in terms of both *RMSE* and D_{stat}. Focusing on the *RMSE* indicator, the proposed neural network metamodel performs the best, followed by BPNN model, ARIMA model, exponential smoothing model and random walk model. Comparing with individual BPNN model, the RMSE of the proposed neural network metamodel is much smaller. From the viewpoint of D_{stat}, the performance of the proposed neural network metamodel is the best of the all. Relative to the individual BPNN model, the performance improvement arrives at 13.76% (86.34%-72.58%), implying that the proposed neural network metamodel has powerful forecasting capability.

For the exchange rate of EUR/USD, the performance of the proposed neural network metamodel is the best, similar to the results of the S&P 500. Likewise, the proposed neural-network-based metamodel has gained much improvement relative to the individual BPNN model and other three typical linear forecasting models. Interestedly, the *RMSE* of the individual BPNN model is slightly worse than that of the ARIMA and ES model, but the directional performance (i.e., D_{stat}) of the individual BPNN model is somewhat better than that of the ARIMA and ES model. The possible reasons are needed to be further addressed later.

In summary, we can conclude that (1) the proposed neural network metamodel performs consistently better than other comparable forecasting models for both the stock index and foreign exchange rate; (2) the evaluation value of the two criteria of the proposed metamodel is much better than that of the individual BPNN model, indicating that the proposed neural network metamodel can effectively capture multiple information and significantly improve prediction performance. One possible reason for this is that the neural network metamodel has nonlinear mapping and multiple-source information integration capability.

4. Conclusions

In this study, a neural-network-based metamodeling technique is proposed to predict financial time series. Through the practical data experiments, we have obtained good prediction results and meantime demonstrated that the neural-network-based meta-model outperforms all the benchmark models listed in this study. These advantages imply that the proposed neural-network-based metamodel can provide a promising solution to financial time series forecasting.

Acknowledgements

This work is partially supported by National Natural Science Foundation of China (NSFC No. 70221001); Key Laboratory of Management, Decision and Information Systems of Chinese Academy of Sciences and Strategic Research Grant of City University of Hong Kong (SRG No. 7001677).

References

[1] L. Yu, S.Y. Wang, K.K. Lai, "A Novel Nonlinear Ensemble Forecasting Model Incorporating GLAR and ANN for Foreign Exchange Rates," *Computers and Operations Research*, vol. 32, pp. 2523-2541, 2005.

[2] U. Naftaly, N. Intrator, D. Horn, "Optimal Ensemble Averaging of Neural Networks," *Network Computation in Neural Systems*, vol. 8, pp. 283-296, 1997.

[3] J. Carney, P. Cunningham, "Tuning Diversity in Bagged Ensembles," *International Journal of Neural Systems*, vol. 10, pp. 267-280, 2000.

[4] P. Chan, S. Stolfo, "Meta-Learning for Multistrategy and Parallel Learning," *Proceedings of the Second International Workshop on Multistrategy Learning*, pp. 150–165, 1993.

[5] A.B.Badiru, D.B. Sieger, "Neural Network as a Simulation Metamodel in Economic Analysis of Risky Projects," *European Journal of Operational Research*, vol. 105, pp. 130-142, 1998.

[6] H. White, "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings," Neural Networks, vol. 3, pp. 535-549, 1990.

[7] L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 26, pp. 123-140, 1996.

[8] Y. Raviv, N. Intrator, "Bootstrapping with Noise: An Effective Regularization Technique," *Connection Science*, vol. 8, pp. 355-372, 1996.

[9] A. Krogh, J. Vedelsby, "Neural Network Ensembles, Cross Validation and Active Learning," *Advances in Neural Information Processing Systems*, vol. 7, pp. 231-238, 1995.

[10] L. Yu, K.K. Lai, S.Y. Wang, W. Huang, "A Bias-Variance-Complexity Trade-off Framework for Complex System Modeling," *Lecture Notes in Computer Science*, vol. 3980, pp. 518-527, 2006.