k-NN Based Neuro-fuzzy System for Time Series Prediction

Conference Paper · July 2013

DOI: 10.1109/SNPD.2013.68

3 authors, including:

Shie-Jue Lee
National Sun Yat-sen University
168 PUBLICATIONS **2,860** CITATIONS

SEE PROFILE

2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing

A k-NN Based Neuro-Fuzzy System for Time Series Prediction

Chia-Ching Wei
Department of Electrical Engineering,
National Sun Yat-Sen University,
Kaohsiung 80424, Taiwan
Email: ccwei@water.ee.nsysu.edu.tw

Thao-Tsen Chen
Department of Electrical Engineering,
National Sun Yat-Sen University,
Kaohsiung 80424, Taiwan
Email: TTChen@water.ee.nsysu.edu.tw

Shie-Jue Lee
Department of Electrical Engineering,
National Sun Yat-Sen University,
Kaohsiung 80424, Taiwan
Email: leesj@mail.ee.nsysu.edu.tw

***Abstract*—Neuro-fuzzy systems have been proposed for different applications for many years. In this paper, a k-NN based neuro-fuzzy predictor is developed for time series prediction. We use a neuro-fuzzy system to generate prediction results. A set of fuzzy rules can be generated by a self-constructing clustering method. These rules can be refined by a hybrid learning algorithm. In stead of using all training data to training a model, we utilize the k-NN method to dynamically select k instances for each prediction. Experimental results show that our approach can provide more accurate predictions than other methods.**

key words: Neuro-fuzzy system, kNN based method, time series, learning algorithm

I. INTRODUCTION

Forecasting the future behavior of a real-world time series data is an important research and application area. It has often been used in many fields, such as stock price prediction, transportation prediction [1] and electrical load prediction [2]. A reliable time series prediction technique can help researchers model the underlying system and forecast its behavior.

The essence of time series prediction is to predict future values of a time series based on the patterns or knowledge learned from the past sequential values of the time series. The prediction techniques can be classified into two categories, statistical techniques and techniques based on advanced tools such as neural networks and fuzzy systems. Box and Jenkins proposed an ARMA model [3]. The model consists of autoregression and moving average terms. This model is not effective since it assumes that the time series process is stationary. Recently, many soft computing methods have been proposed for time series forecasting, such as artificial neural network [4]–[13], support vector regression machines [14]–[17], and fuzzy theory [18]–[20]. Neuro-fuzzy is another popular technology for time series modeling and prediction. It combines the advantages of fuzzy theory and neural networks, i.e., adaptability, quick convergence, and high accuracy.

In general, we use all historical data to create a global model. Huang et al. [21] proposed a k-NN based time series prediction algorithm. This method used Euclidean distance and the trend of a sequence to measure data similarity. The k instances of historical data that are most similar to the testing input are used for training a model which is then used for prediction. As a result, different models are adopted for different testing inputs.

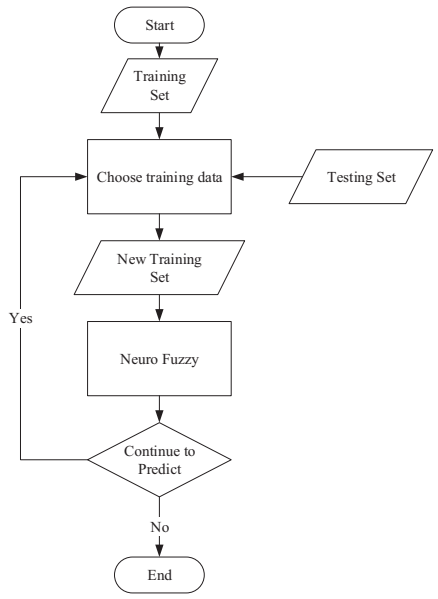


Fig. 1. System Architecture.

In this paper, we propose a k-NN based neuro-fuzzy prediction system. We choose the k most similar data to train a model for each test input. A set of fuzzy rules can be generated by a self-constructing clustering method. These rules can be refined by a hybrid learning algorithm. The remainder of this paper is organized as follows. We give an overview about our prediction system in Section II. Then we describe how to measure data similarity in Section III. In Section IV, we introduce the neuro-fuzzy system. Experiments are presented in Section V. Finally, a conclusion is given in Section VI.

II. SYSTEM ARCHITECTURE

In this section, we give an overview about the proposed prediction system. Fig. 1 shows the system architecture of our proposed k-NN based neuro-fuzzy predictor. The prediction

978-0-7695-5005-3/13 $26.00 © 2013 IEEE
DOI 10.1109/SNPD.2013.68

system integrates the k-NN method and the neuro-fuzzy modeling technique. The k-NN method is used to generate the training data set. Neuro-fuzzy modeling is used to perform prediction based on the training data. We use the k-NN method to select he k instances of historical data that are most similar to the testing input. These data are used to create a predicting model. The model will be retrained at each prediction. The algorithm involved can be described as below:

Step 1: Compute similarity between test input and historical data. Then find the k instances which is most similar to the input.

Step 2: Use the data chosen at step 1 to train a neuro-fuzzy model, and use the model to perform prediction for the input.

Step 3: If more predictions are to be done, go to step 1, else stop the process.

Details about each component in the prediction system are given below.

III. Data Similarity

We use the k-NN method to select k instances from training set for each testing vector. The k-NN method is employed to reduce the training set by selecting the k instances in the training data which are closest to the testing vector.

Euclidean distance is often used to measure the similarity between data. However, for a sequence of time series data, the trend of the sequence should also be considered [21]. Suppose we have a time series y. We choose D_y previous data to predict the value at the next point $t+1$. The testing vector can be expressed as

$$(y(t), y(t-1) \ldots, y(t-D_y+1).$$

The ith training vector can be expressed as

$$(y(i), y(i-1), \ldots, y(i-D_y+1).$$

The Euclidean distance between the testing vector and the ith training vector can be express as:

$$ED(t) = \sqrt{\sum_{j=0}^{D_y-1} (y(t-j) - y(i-j))^2}. \quad (1)$$

Now we take the first difference of the testing vector:

$$(y(t) - y(t-1), \ldots, y(t-D_y+2) - y(t-D_y+1)$$

and the first difference of the ith training vector:

$$(y(i) - y(i-1), \ldots, y(i-D_y+2) - y(i-D_y+1)$$

We calculate the Euclidean distance between the differential testing vector and the ith differential training vector:

$$Diff(t) = \sum_{j=0}^{D_y-2} ((((y(t-j) - y(t-j-1)) - \quad (2)$$

$$(y(i-j) - y(i-j-1)))^2))^{\frac{1}{2}}. \quad (3)$$

Then we normalize $ED(t)$ and $Diff(t)$. The total distance is obtained by the following equation:

$$TotalDistance = \frac{Diff(t) - MIN(Diff)}{MAX(Diff) - MIN(Diff)} + \frac{ED(t) - MIN(ED)}{MAX(ED) - MIN(ED)} \quad (4)$$

where $MAX(Diff), MIN(Diff), MAX(ED)$ and $MIN(ED)$ are the maximum and minimum value for $Diff$ and ED. We use this distance measure to find the k most similar data to form the new training set to predict the value at the next point $t+1$.

IV. Neuro Fuzzy Modeling

Neuro-fuzzy techniques possess the advantages of fuzzy theory and neural networks, i.e., adaptability, quick convergence, and high accuracy. In this paper, we use a neuro-fuzzy modeling technique with self-constructing rule generation and hybrid SVD-based learning [22] to perform prediction. The modeling process consists of two stages, structure identification and parameter identification. The former is used to generated initial fuzzy rules, and the latter is used to refine the parameters involved in a model.

A. Structure Identification

Structure identification focuses on initializing the structure of a neuro-fuzzy system. In this phase, a set of initial fuzzy rules are developed from the given input-output training data.

The idea is based on input-similarity and output-similarity. We use a self-constructing fuzzy clustering algorithm to partition the k training data obtained in Section III into fuzzy clusters. The membership function is defined with the mean and deviation of the data points included in a cluster. Then, the initial fuzzy rules are obtained from the obtained fuzzy clusters. Suppose $x_1, x_2, ..., x_n$ are input variables and y is the output variable. By input-output similarity tests, the given training data are partitioned into J fuzzy clusters. Each cluster C_j is described as $(G_j(\vec{x}), c_j)$ where $G_j(\vec{x})$ contains the mean vector $\vec{m}_j = [m_{1j}, m_{2j}, ..., m_{nj}]$ and the deviation vector $\vec{\sigma}_j = [\sigma_{1j}, \sigma_{2j}, ..., \sigma_{nj}]$, and c_j is the height associated with C_j, $1 \leq j \leq J$. The jth fuzzy rule, R_j, is defined as follows:

$$\begin{aligned} &\text{IF } x_1 \text{ IS } \mu_{1j}(x_1) \text{ AND ... AND } x_n \text{ IS } \mu_{nj}(x_n) \\ &\text{THEN } y \text{ IS } f(x_1, \ldots, x_n) \end{aligned} \quad (5)$$

where $\mu_{ij}(x_i)$ is the membership function of the fuzzy set associated with x_i and $f(x_1, \ldots, x_n) = b_{0j} + b_{1j}x_1 + \ldots + b_{nj}x_n$. Note that the membership function is assumed to be a Gaussian function, i.e.,

$$\mu_{ij} = g(x_i; m_{ij}, \sigma_{ij}) = exp\left[-\left(\frac{x_i - m_{ij}}{\sigma_{ij}}\right)^2\right] \quad (6)$$

where m_{ij} is its mean and σ_{ij} is its deviation. With these J fuzzy rules, we can construct a five-layer neural-fuzzy network as shown in Figure 2. The operation of each the neuro-fuzzy network is described as follows.

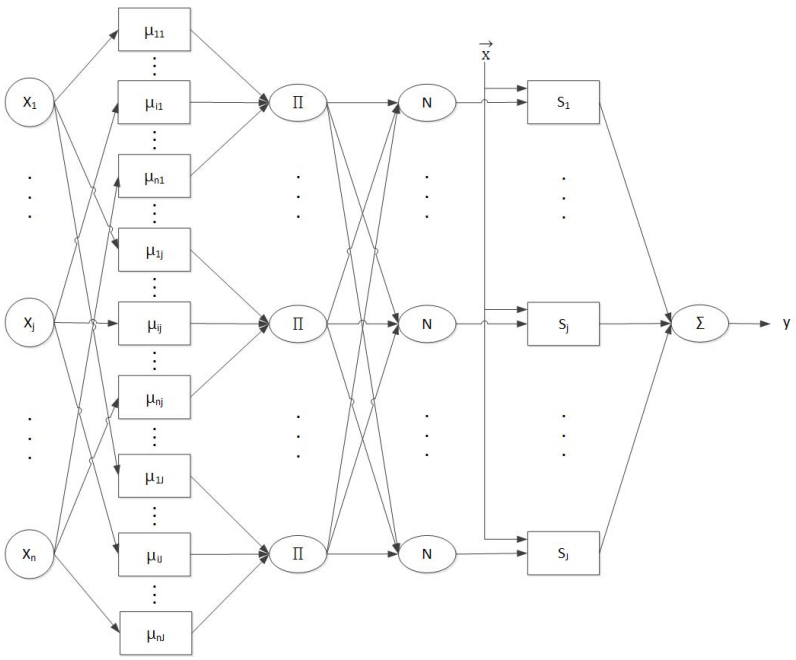


Fig. 2. Architecture of the neuro-fuzzy network.

1) Layer 1: Computing the matching degree to a fuzzy condition involving one variable,i.e.,

$$o_{ij}^{(1)} = exp\left[-\left(\frac{x_i - m_{ij}}{\sigma_{ij}}\right)^2\right] \tag{7}$$

for $1 \leq i \leq n$, $1 \leq j \leq J$

2) Layer 2: Computing the firing strength of each rule,i.e.,

$$o_j^{(2)} = \prod_{i=1}^{n} o_{ij}^{(1)} \tag{8}$$

for $1 \leq j \leq J$

3) Layer 3: Normalizing the matching degree for each rule,i.e.,

$$o_j^{(3)} = \frac{o_j^{(2)}}{\sum_{k=1}^{J} o_k^{(2)}}, 1 \leq j \leq J \tag{9}$$

4) Layer 4: Computing the conclusion inferred by each fuzzy rule,i.e.,

$$\begin{aligned} o_j^{(4)} =& o_j^{(3)} \times f_j(\vec{x}) \\ =& o_j^{(3)} \times (b_{0j} + b_{1j}x_1 + b_{2j}x_2 + ... + b_{nj}x_n) \\ 1 \leq j \leq J \end{aligned} \tag{10}$$

5) Layer 5: Combining the conclusions of all fuzzy rules and obtaining the network output:

$$y = o^{(5)} = \sum_{j=1}^{J} o_j^4 \tag{11}$$

B. Parameter Identification

The neuro-fuzzy system should be properly trained to generate an optimal input/output mapping. In the parameter identification phase, we use neural network techniques to optimize the parameters associated with the initial fuzzy rules. Ouyang et al. [22], [23] developed a hybrid learning algorithm, which combines a recursive SVD-based least squares estimator and the gradient descent method, to refine the parameters. The recursive SVD-based least squares estimator is used to modify b_{0j}, b_{1j}, ..., b_{nj}, and the gradient descent method is used to modified m_{ij} and σ_{ij}.

In Figure 2, the output y is computed by centroid defuzzification as:

$$y = \frac{\sum_{j=1}^{J} \alpha_j(\vec{x}) \times f_j(\vec{x})}{\sum_{j=1}^{J} \alpha_j(\vec{x})} \tag{12}$$

$$= \frac{\sum_{j=1}^{J} \alpha_j(\vec{x}) \times (b_{0j} + b_{1j}x_1 + b_{2j}x_2 + \cdots + b_{nj}x_n)}{\sum_{j=1}^{J} \alpha_j(\vec{x})} \tag{13}$$

where $\alpha_j(\vec{x})$ is the degree the input $\vec{x}$ matches rule R_j and is computed by the product operator:

$$\alpha_j(\vec{x}) = \mu_1(x_1) \times \mu_2(x_2) \times ... \times \mu_n(x_n). \tag{14}$$

The parameters involved are refined as follows. Suppose ($\vec{p}_v$,q_v) is the vth training pattern, where $\vec{p}_v = [p_{1v}, ..., p_{nv}]$

is the input vector and q_v is the desire output. According to Eq.11, we have

$$v.o^{(5)} = \sum_{j=1}^{J} v.o^{(3)} \times (b_{0j} + b_{1j}p_{1v} + ... + b_{nj}p_{nv}). \quad (15)$$

We want to minimize $\mid q_v - v.o^{(5)} \mid$ for the vth training pattern. For all the k training patterns, we have k equations in the form of Eq.15. Clearly, we would like

$$J(X) = \|B - AX\| \quad (16)$$

to be minimized. Note that $\|D\| = \sqrt{trace(D^T D)}$ and

$$B = \begin{bmatrix} q_1 & q_2 & ... & q_k \end{bmatrix}^T \quad (17)$$

$$A = \begin{bmatrix} a_{11} & \dots & a_{11}p_{n1} & \dots a_{1J} & \dots a_{1J}p_{n1} \\ a_{21} & \dots & a_{21}p_{n2} & \dots a_{2J} & \dots a_{2J}p_{n2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{k1} & \dots & a_{k1}p_{nk} & \dots a_{kJ} & \dots a_{kJ}p_{nk} \end{bmatrix} \quad (18)$$

$$X = \begin{bmatrix} b_{01} & b_{11} & \dots & b_{n1} & \dots b_{0J} & b_{1J} \dots b_{nJ} \end{bmatrix}^T \quad (19)$$

We treat m_{ij} and σ_{ij} as fixed, so we can find an optimal X to minimize Eq.16. We use recursive SVD-based least squares estimator to find X. Details about the recursive SVD-based least squares estimator can be found is in [23].

Parameters m_{ij} and σ_{ij} are refined by the gradient descent method. The error function is

$$E = \frac{1}{2k} \sum_{v=1}^{k} (q_v - y_v)^2 \quad (20)$$

The learning rules for m_{ij} and σ_{ij} are

$$m_{ij}^{new} = m_{ij}^{old} - \alpha_1 (\frac{\partial E}{\partial m_{ij}}) \quad (21)$$

$$\sigma_{ij}^{new} = \sigma_{ij}^{old} - \alpha_2 (\frac{\partial E}{\partial \sigma_{ij}}) \quad (22)$$

where α_1 and α_2 are predefined learning rates.

V. EXPERIMENTAL RESULT

In this section, we use two time series data to compare the performance of the proposed method with others. In Experiment I, we use TAIEX (Taiwan Stock Exchange Capitalization Weighted Stock Index) [24] to evaluate the prediction ability of the U_FTS (univariate fuzzy time series model) [25], U_R (univariate conventional regression model,) [25], U_NN(univariate neural network model) [25], U_NN_FTS (univariate neural network-based fuzzy time series model) [25], U_NN_FTS_S (univariate neural network-based fuzzy time series model with substitutes) [25], Neuro-Fuzzy (Global) [23], and our proposed method. For convenience, our proposed method is abbreviated as kNF. In Experiment II, we use the Poland Electricity Load data set to demonstrate the effectiveness of our proposed method. Poland Electricity Load [26] is a commonly used benchmark data set for time series prediction. This time series represent the daily load of Poland during around 1500 days in the 1990s.

TABLE I
TESTING ERROR OF kNF WITH DIFFERENT SETTINGS OF k FOR TAIEX

k	1999	2000	2001	2002	2003	2004	Average
50	109	156	114	68	63	66	96.04
60	100	152	**113**	67	59	66	92.86
70	100	147	115	66	58	63	91.42
80	103	146	115	66	56	63	91.40
90	102	137	116	66	54	61	89.40
100	101	137	116	67	55	60	89.16
110	101	**134**	116	68	55	57	88.59
120	100	**134**	117	68	54	57	88.28
130	100	**134**	118	68	**53**	**55**	**87.92**
140	**99**	**134**	119	67	**53**	57	88.11
150	**99**	137	120	67	**53**	56	88.61
160	**99**	137	118	67	54	56	88.46
170	101	137	116	67	53	55	88.18
180	102	139	116	**66**	53	57	88.78
190	102	138	116	67	53	56	88.74
200	103	138	117	67	53	57	89.03

TABLE II
COMPARISON ON TESTING ERROR IN RMSE FOR TAIEX

Methods	1999	2000	2001	2002	2003	2004	Average
U_FTS	120	176	148	101	74	84	117.4
U_R	164	420	1070	116	329	146	374.2
U_NN	107	309	259	78	57	60	145.0
U_NN_FTS	109	255	130	84	56	116	125.0
U_NN_FTS_S	109	152	130	84	56	116	107.8
NF_Global	104	137	**116**	67	53	56	88.8
kNF	**99**	**137**	119	**67**	**52**	**55**	**88.1**

In the following, MAE, MSE, and RMSE are used for evaluation of the prediction ability for each method. They are defined as:

$$MAE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \quad (23)$$

$$MSE = \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n} \quad (24)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \quad (25)$$

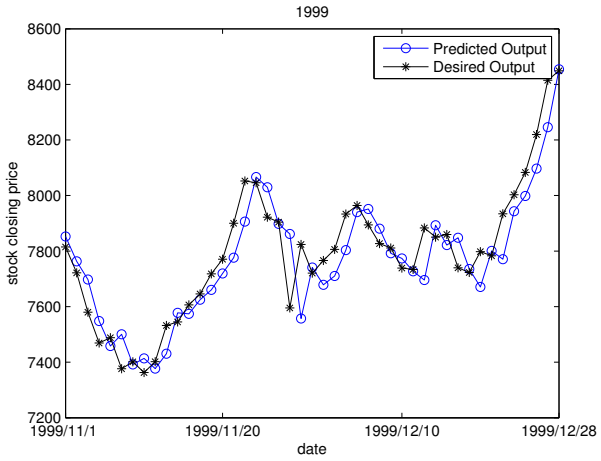
where $\hat{y}$ is the predicted value, y is the actual value, i is the ith testing points for $1 \leq i \leq n$, and n is the number of testing points.

A. Experiment I

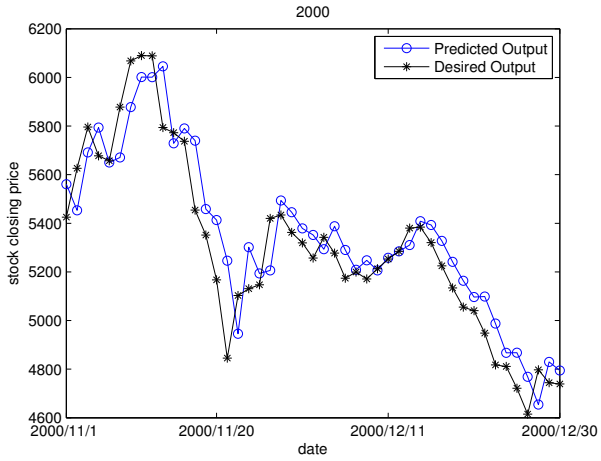
In this experiment, we use the TAIEX data set. We adopt mostly a 10-month/2-month split to generate the training and testing data. We use 3 or 5 previous historical data to do prediction for a given time. Table. I shows the RMSE values obtained by kNF with different k. As shown in the table, setting k to be 130≈150 can provide better prediction. A comparison for testing error is shown in Table II. The results show that kNF can have the least RMSE and average RMSE. The method can offer a higher accuracy than the others. Fig. 3 shows pictorially the prediction results using kNF from 1999 to 2004.

B. Experiment II

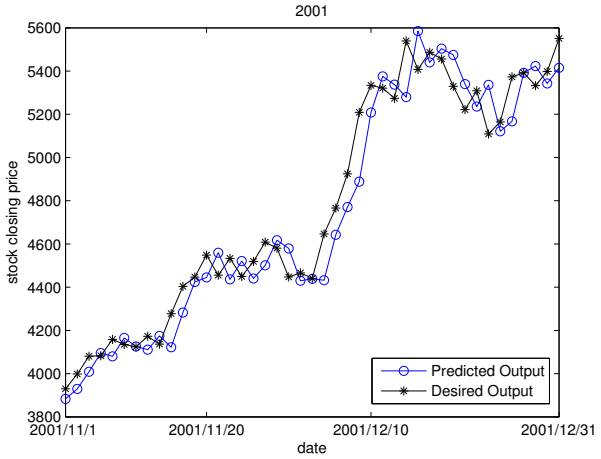
In this experiment, the Poland Electricity Load data set is used. The data set is shown in Fig. 4. We use 1000 first values



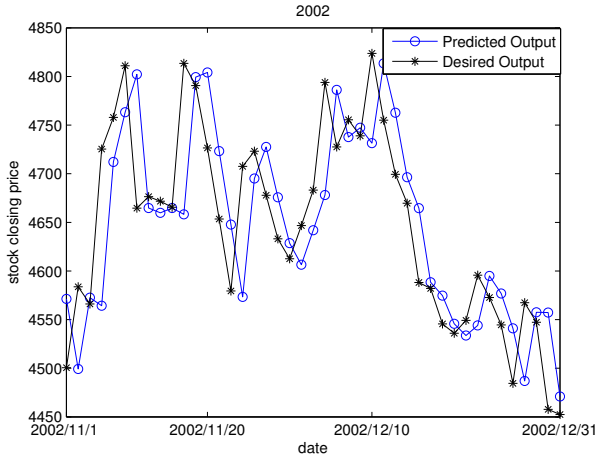
(a) 1999



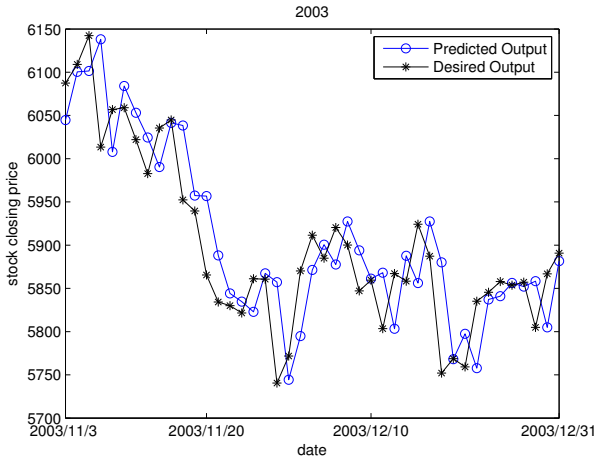
(b) 2000



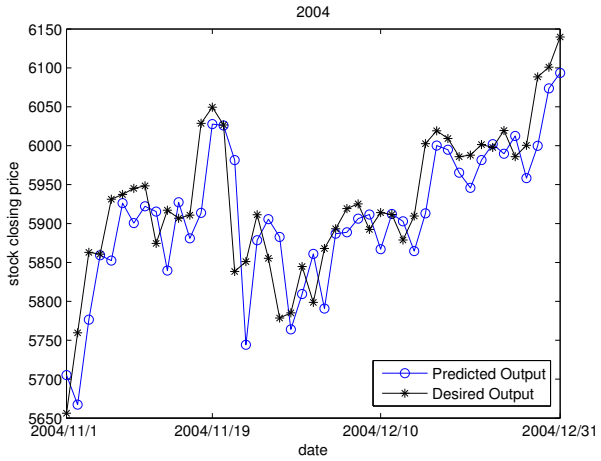
(c) 2001



(d) 2002



(e) 2003



(f) 2004

Fig. 3. Predicted results by kNF for Experiment I.

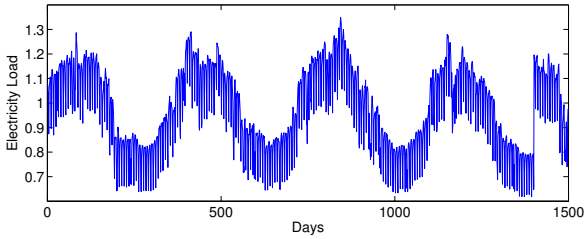


Fig. 4. The Poland Electricity Load data set.

TABLE III
TESTING ERROR OF kNF WITH DIFFERENT SETTINGS OF k FOR POLAND

k	MAE	MSE
100	0.0374	0.0042
150	**0.0364**	**0.0032**
250	0.0391	0.0037
350	0.0404	0.0037
450	0.0438	0.0043
550	0.0452	0.0044
650	0.0468	0.0045
750	0.0477	0.0046
850	0.0487	0.0047
950	0.0507	0.0049

for training, and the rest, over 500 values, for testing. In this experiment, k is set to be 150 and 3 previous data are used to do prediction for a given time.

In Table. III, the results obtained by kNF with different k are shown. Note that setting k to be around 150 can provide better prediction. Table. IV shows a comparison on training error among ARMA, Neural Network, Neuro-Fuzzy, and kNF using the Poland Electricity Load data set. From this table, it is obvious that kNF can have the least MAE and MSE. The method can perform much better than others.

VI. CONCLUSION

We have proposed a k-NN based neuro-fuzzy predictor for time series prediction. Instead of using all historical data to training a global model, we choose the k most similar data as the training set to create a local model at each prediction. In this way, the training set each time is more relevant to the prediction to be done than the global model approach. The fuzzy rules are generated automatically by a self-constructing clustering method and the parameters involved in a model are refined by a hybrid learning algorithm. Experimental results have shown that our approach can provide more accurate predictions than other methods.

TABLE IV
COMPARISON ON TESTING ERROR FOR POLAND

Method	MAE	MSE
ARMA	0.3744	0.4807
Neural Network	0.0639	0.0835
NF_Global	0.0521	0.0055
kNF	**0.0364**	**0.0032**

REFERENCES

[1] S. F. Crone. Artificial neural network & computational intelligence forecasting competition. www.neural-forecasting-competition.com/, 2010.

[2] M. Nosrati Maralloo, A. R. Koushki, C. Lucas, and A. Kalhor. Long term electrical load forecasting via a neurofuzzy model. In *In Proceedings of the 14th International CSI Computer Conference*, pages 35–40, 2009.

[3] G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting And Control*. WILEY, 2008.

[4] Y. Yoon, J. George Swales, and T.M. Margavio. A comparison of discriminant analysis versus artificial neural networks. *The Journal of the Operational Research Society*, 44:51–60, 1993.

[5] E.W. Saad, D.V. Prokhorov, I. Donald, and C. Wunsch. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks*, 9:1456–1470, 1998.

[6] J. C. Principe, N. R. Euliano, and W. C. Lefebvre. *Neural and Adaptive Systems: Fundamentals through Simulations*. John Wiley & Sons, New York, USA, December 1999.

[7] K.-J. Kim and I. Han. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Applications*, 19:125–132, August 2000.

[8] M. Ghiassi and H. Saidane. A dynamic architecture for artificial neural networks. *Neurocomputing*, 63:397–413, 2005.

[9] K. Huarng and T.H.-K. Yu. The application of neural networks to forecast fuzzy time series. *Physica A: Statistical Mechanics and its Applications*, 363:481–491, 2006.

[10] Y.-K. Kwon and B.-R. Moon. A hybrid neurogenetic approach for stock forecasting. *IEEE Transactions on Neural Networks*, 18:851–864, 2007.

[11] H.-J. Kim and K.-S. Shin. A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets. *Applied Soft Computing*, 7:569–576, 2007.

[12] T.-J. Hsieh, H.-F. Hsiao, and W.-C. Yeh. Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, 11:2510–2525, 2011.

[13] M. Khashei and M. Bijari. A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing*, 11:2664–2675, 2011.

[14] T.V. Gestel, J.A.K. Suykens, D.E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, and J. Vandewalle B.D. Moor. Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks*, 12:809–821, 2001.

[15] L. Cao and F.E.H. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14:1506–1518, 2003.

[16] G. Valeriy and B. Supriya. Support vector machine as an efficient framework for stock market volatility forecasting. *Computational Management Science*, 3:147–160, 2006.

[17] C.-Y. Yeh, C.-W. Huang, and S.-J. Lee. A multiple-kernel support vector regression approach for stock market price forecasting. *Expert Systems with Applications*, 38:2177–2186, 2011.

[18] Q. Song and B. S. Chissom. Fuzzy time series and its model. *Fuzzy Sets and Systems*, 54:269–277, 1993.

[19] Q. Song and B. S. Chissom. Forecasting enrollments with fuzzy time seriesxpart i. *Fuzzy Sets and Systems*, 54:1–9, 1993.

[20] Q. Song and B. S. Chissom. Forecasting enrollments with fuzzy time seriesxpart ii. *Fuzzy Sets and Systems*, 62:1–7, 1994.

[21] Z. Huang and M.-L. Shyu. k-nn based ls-svm framework for long-term time series prediction. In *Information Reuse and Integration (IRI), 2010 IEEE International Conference on*, 2010.

[22] S.-J. Lee and C.-S. Ouyang. A neuro-fuzzy system modeling with self-constructing rule generation and hybrid svd-based learning. *IEEE TRANSACTIONS ON FUZZY SYSTEMS*, 11:341–353, 2003.

[23] C.-S. Ouyang, W.-J. Lee, and S.-J. Lee. A tsk-type neurofuzzy network approach to system modeling problems. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSXPART B: CYBERNETICS,*, 35:751–767, 2005.

[24] Taiwan Futures Exchange Corporation. http://www.taifex.com.tw/.

[25] T. H.-K. Yu and K.-H. Huarng. A bivariate fuzzy time series model to forecast the taiex. *Expert Systems with Applications*, 34, 2008.

[26] Poland electricity load data website:. http://research.ics.aalto.fi/eiml/.