Forecasting Time Series by SOFNN with Reinforcement Learning

Takashi Kuremoto, Masanao Obayashi, and Kunikazu Kobayashi

***Abstract*— A self-organized fuzzy neural network (SOFNN) with a reinforcement learning algorithm called Stochastic Gradient Ascent (SGA) is proposed to forecast a set of 11 time series. The proposed system is confirmed to predict chaotic time series before, and is applied to predict each/every time series in NN3 forecasting competition modifying parameters of threshold of fuzzy neurons only. The training results are obviously effective and results of long-term prediction give convincible trend values in the future of time series.**

I. Introduction

Though many artificial neural networks (ANN) are suitable to forecast time series, radial basis function network (RBFN) is still recommend to be applied on chaotic time series [1], [2], [3] and financial time series [4]. Meanwhile, how to design the structure of hidden-layer in RBFN is a puzzling and confusing problem in practice. Leung and Wang proposed a technique called the cross-validated subspace method to estimate the optimum number of hidden units, and applied the method to prediction of noisy chaotic time series [3] but the method may fall its efficiency when the training sample data is not enough. To overcome the problem of RBFN, we proposed a self-organization fuzzy network and its effectiveness on prediction of chaotic time series was investigated [5], [6].

Furthermore, the learning algorithm is so important to any artificial neural network. Reinforcement learning (RL), a kind of goal-directed learning, is well known for an agent adapting unknown environments [7], [8]. We have proposed to apply a kind of RL called stochastic gradient ascent (SGA) on nonlinear predations [5], [6], [9]. The accurate of forecasting in experiments using Lorenz chaotic time series shown its good efficiency whatever the type of ANN is either multilayer perception (MLP) or SOFNN.

In this paper, we intend to use SOFNN and SGA to forecast a set of 11 time series given by neural forecasting competition (NN3) [10]. The prediction system is introduced in Section 1 in detail, and all of train forecasting results and forecasting results are shown in Section 2.

II. Forecasting System

Flow of forecasting is shown in Fig. 1 and self-organized fuzzy neural network, ANN predictor, is a RBF-like neural network (Fig. 2). The input layer is given by data in history of time series (Subsection 2.1). The hidden layer consists Gaussian membership functions which number is decided by a threshold and the rule layer realizes fuzzy inference (Subsection 2.2 and 2.3). Forecasting is executed according to a probability policy which to determinate actions in the procedure of reinforcement learning and the error of forecasting is as reward value (Subsection 2.4).

Authors are with the Graduate School of Science and Systems Engineering, Yamaguchi University , Tokiwadai 2-16-1, Ube, Yamaguchi 755-8611, Japan, Tel: +81-836-859520, Fax: +81-836-859501, Email: {wu, m.obayas, koba}@yamaguchi-u.ac.jp

A. Embedding

According to the Takens embedding theorem, the inputs of prediction system on time t, can be reconstructed as a n dimensions vector space $X(t)$, which includes n observed points with same intervals on a time series $y(t)$.

$$X(t) = (x_1(t), x_2(t), \cdots, x_n(t)) \quad (1)$$

$$= (y(t), y(t-\tau), \cdots, y(t-(n-1)\tau) \quad (2)$$

where τ is time delay (interval of sampling) may be an arbitrary value in theory but it effects prediction accuracy in practice, n is the embedding dimension, $n > 1$.

B. Self-organized Fuzzy Neural Network (SOFNN)

Fig. 2 shows an architecture of self-organized fuzzy neural network (SOFNN) we proposed. The initial number of membership function and fuzzy rule is only 1, respectively.

1) Membership Function: To each element $x_i(t)$ of the input *X*(t), membership function $B_{ij}(x_i(t))$ is represented as

$$B_{ij}(x_i(t)) = exp\left\{-\frac{(x_i(t)-m_{ij})^2}{2\sigma_{ij}^2}\right\} \quad (3)$$

where m_{ij} and σ_{ij} are the parameters of mean and standard deviation of the Gaussian membership function in jth node, respectively. Initially, $j = 1$, and with increasing of input patterns, the new membership function will be added.

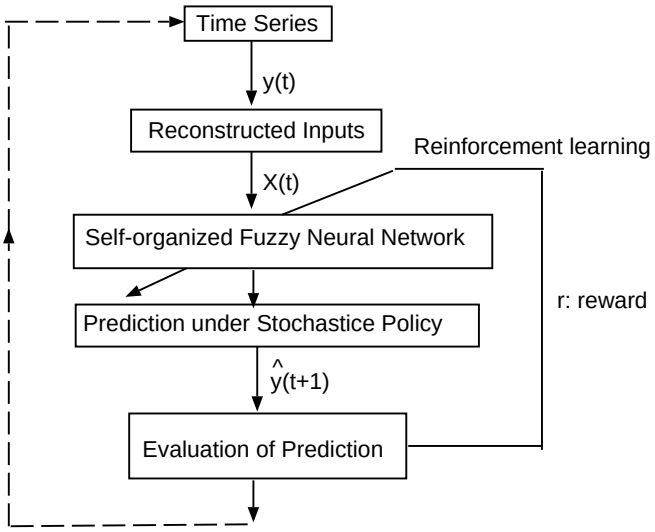

Fig. 1. Flow chart of training and forecasting.

2) Fuzzy Inference: The fitness $\lambda_k(X(t))$ is an algebraic product of membership functions which connects to rule k.

$$\lambda_k(X(t)) = \prod_{i=1}^{n} B_{io}(x_i) \tag{4}$$

where o is the number of membership function, connects with kth rule. $o \in \{1, 2, \cdots, l_i\}$.

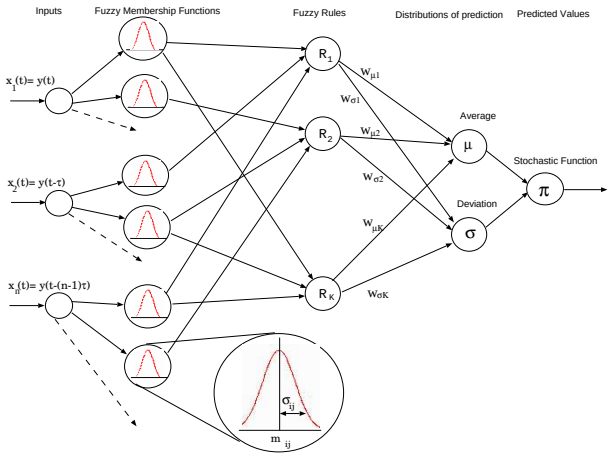

Fig. 2. A structure of self-organized fuzzy neural network (SOFNN)

3) Self-organization of Neural Network : A new membership function will be add if

$$B_{ij}(x_i(s)) < \sqrt[n]{F}(\bar{F} : threshold) \tag{5}$$

corresponding to the input,$l_i \leftarrow l_i + 1$.

A new fuzzy rule will be added also adapting to the new membership function.

4) Prediction Policy from Defuzzification: Integrate fuzzy rules with connection weight, fuzzy inference can be obtained. The output of signal can be considered as a new Gaussian distribution either, i.e.,

$$\mu(X(t), \omega_{\mu k}) = \frac{\sum_{k=1}^{K} \lambda_k \omega_{\mu k}}{\sum_{k=1}^{K} \lambda_k} \tag{6}$$

$$\sigma(X(t), \omega_{\sigma k}) = \frac{\sum_{k=1}^{K} \lambda_k \omega_{\sigma k}}{\sum_{k=1}^{K} \lambda_k} \tag{7}$$

tonaru. where μ is the mean of output,σ is its standard deviation. Weight $\omega_{\mu k}$ and $\omega_{\sigma k}$ are parameters concerning with inputs set$X(t)$, and will be renew during training.

$$\pi(\hat{y}(t+1), W, X(t)) = \frac{1}{\sqrt{2\pi}\sigma} exp\left\{ -\frac{(\hat{y}(t+1) - \mu)^2}{2\sigma^2} \right\} \tag{8}$$

where $\hat{y}(t+1)$ is the value of one-step ahead prediction, produce by regular random numbers. *W* means weights $\omega_{\mu k}$ and $\omega_{\sigma k}$.

C. Reinforcement Learning Algorithm (SGA)

Reinforcement learning has recently been well-known as a kind of intelligent machine learning [7], [8]. It needs not any model of operator but learns to approach to its goal by observing sensing rewards from environments. Kimura and Kobayashi suggested an algorithm called stochastic gradient ascent (SGA), which respect to continuous action and confirmed in control systems [8]. The error of forecasting is as reward to adjust probability policy in training process. SGA algorithm is shown under.

1) Accept an observation $X(t)$ from environment.
2) Predict a future data $\hat{y}(t+1)$ under a probability $\pi(\hat{y}(t+1), W, X(t))$.
3) Collate training samples of times series, take the error as reward r_i.
4) Calculate the degree of adaption $e_i(t)$, and its history for all elements ω_i of internal variable W. where γ is a discount ($0 \leq \gamma < 1$).

$$e_i(t) = \frac{\partial}{\partial \omega_i} ln\big(\pi(\hat{y}(t+1), W, X(t))\big) \tag{9}$$

$$D_i(t) = e_i(t) + \gamma D_i(t-1) \tag{10}$$

5) Calculate $\Delta\omega_i(t)$ by under equation.

$$\Delta\omega_i(t) = (r_i - b) D_i(t) \tag{11}$$

where b is a constant.

6) Improvement of policy: renew *W* by under equation.

$$\Delta W(t) = (\Delta\omega_1(t), \Delta\omega_2(t), \cdots, \Delta\omega_i(t), \cdots) \tag{12}$$

$$W \leftarrow W + \alpha(1-\gamma)\Delta W(t) \tag{13}$$

where α is a learning constant,non-negative.

7) Advance time step t to $t+1$,return to(1).

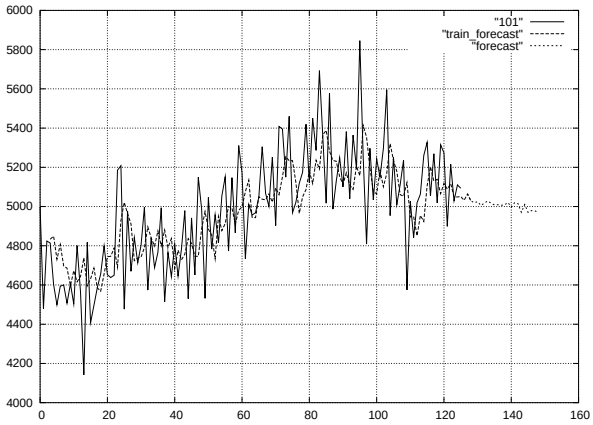

Fig. 3. Forecasting results of time series 101.

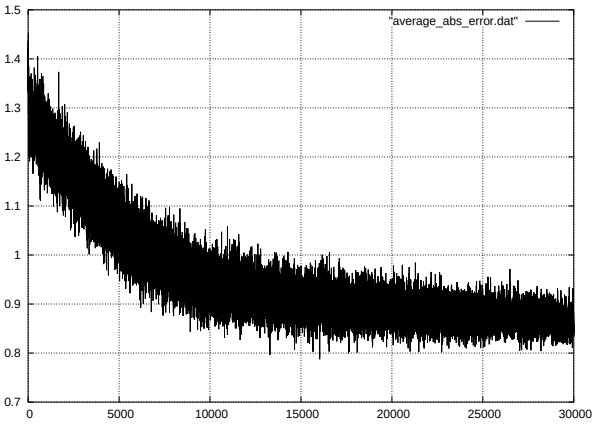

Fig. 4. Change of forecasting error in training.

III. Forecasting NN3 Time Series

We applied the proposed forecasting system on a set of 11 time series provided by Neural Forecasting Competition (NN3)[10]. The procedure is :

1) Obtain all data (from 1 to 126, or 123, or 115 steps) of one time series given by NN3 and normalize their value to around 0.0 using a linear transformation;

2) Let the input dimension be 3, and the delay of time be 1;

3) Tune parameters ,i.e. $\sqrt[n]{F}$ (threshold of membership function $B_{ij}(x_i(s))$), initial value of π (i.e, only σ_{ij} because m_{ij} was set to 0.0), value of reward r, to appropriate /satisfied values by observing change of training errors, then train forecasting is obtained when the training error (Mean Absolute Error) becomes to converge;

4)Forecast unknown data in the future 18 steps using trained system. The only pre-processing to input data was to modify values to around 0.0 using a fixed linear transformation, and the number of iterations in training was 30,000 for 11 time series.

TABLE I

Results and parameters in training and forecasting.

Data	$SMAPE_s$	$\sqrt[n]{F}$	σ	r
101	98.4	0.99	10.0	1.0
102	41.9	0.82	7.8	1.0
103	131.1	0.95	14.0	1.2
104	107.4	0.90	2.0	1.0
105	100.0	0.98	9.0	1.0
106	99.7	0.92	1.0	1.0
107	97.4	0.90	2.0	0.5
108	106.4	0.85	3.0	1.0
109	104.6	0.92	2.0	1.25
110	113.4	0.80	10.0	1.0
111	106.5	0.90	3.0	1.0

Fig. 3 shows training/forecasting result of time series 101. Fig. 4 shows its membership function number's change and its rule number's change is shown in Fig. 5.

The results of training shown the effectiveness of our method, and the results of long-term forecasting gave convincible trends of the time series in future.

The forecasting results for all 11 time series are shown in Table 1, where SMAPE means the mean Symmetric Mean Absolute Percent Error across a time series ($SMAPE_s = \frac{1}{n}\sum_{t=1}^{n}\frac{|x_t-\hat{y}_t|}{(x_t+\hat{y}_t)/2} * 100$)[10].

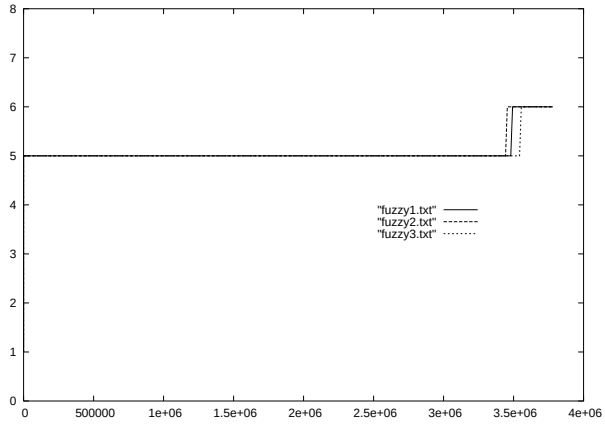

Fig. 5. Change of numbers of membership functions during training.

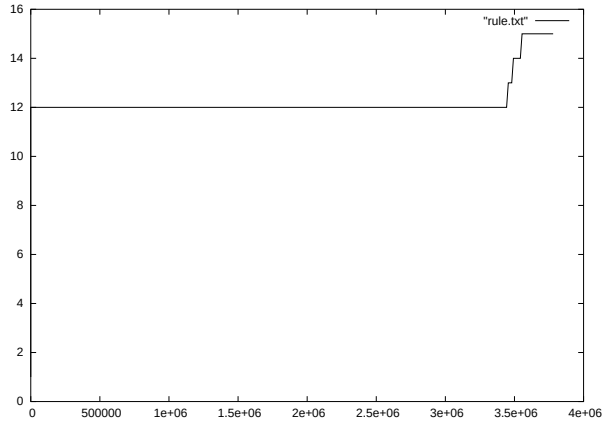

Fig. 6. Change of numbers of fuzzy rules during training.

Parameters of prediction system are reported under .

1) Reconstruction of input space by embedding(Equ.(1),(2)):
   Embedding dimension n = 3
   Time delay τ= 1
2) Self-organized fuzzy neural network:
   Initial value of weight $\omega_{\mu k}$ = 0.0
   Initial value of weight $\omega_{\sigma k}$ = 0.5
3) Reinforecement learning of SGA:
   Limitation of errors ε = 1.0
   Discount γ = 0.9
   Learning constant:
   For weight $\omega_{\mu k}$, $\alpha_{\omega_{\mu k}}$ = 0.003
   For weight $\omega_{\sigma k}$, $\alpha_{\omega_{\sigma k}}$ = 3.0E-6
   For mean and standard deviation m_{ij}, σij: $\alpha_{m_{ij}}$, $\alpha_{\sigma_{ij}}$ = 3.0E-6

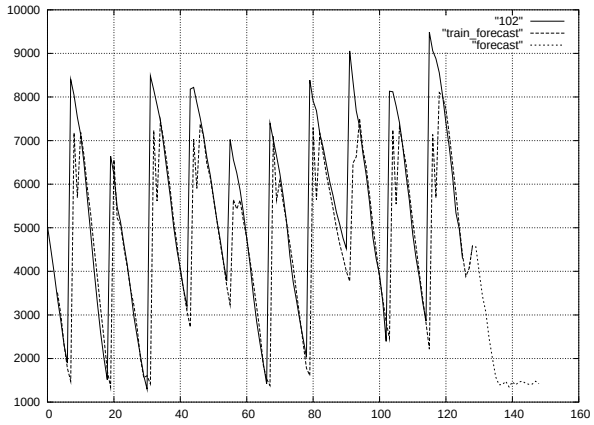

Fig. 7. Forecasting results of time series 102.

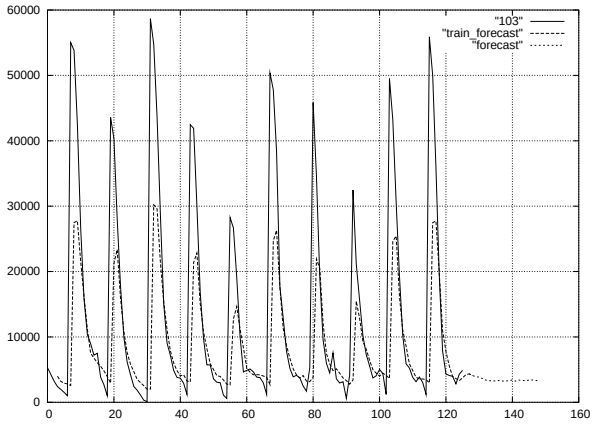

Fig. 8. Forecasting results of time series 103.

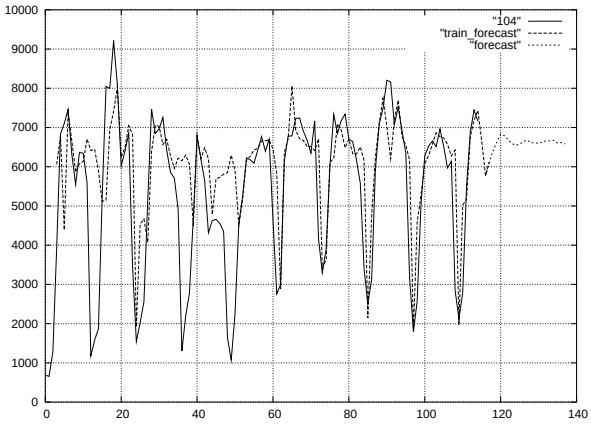

Fig. 9. Forecasting results of time series 104.

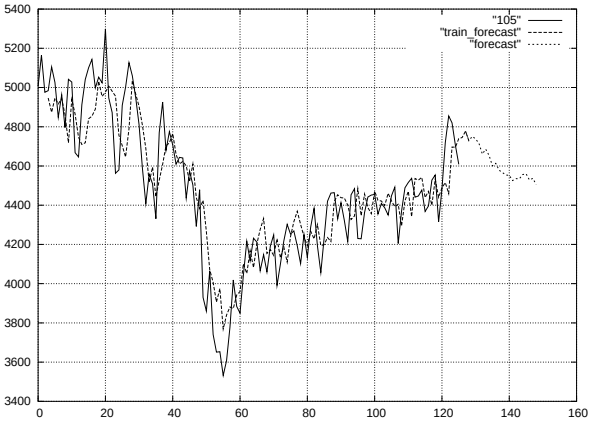

Fig. 10. Forecasting results of time series 105.

IV. Conclusion

The robustness of proposed self-organization fuzzy neural network with reinforcement learning algorithm is confirmed by a set of NN3 reduced data. The predictor is easy to use by only tuning its threshold and initial deviation of membership function and reward parameter. Except normalization to sample data, no any additional operation to original data and forecasting result. All train steps were limited to 30,000 and average training time was not over 2 minutes on a personal computer (3.0GHz CPU).

Acknowledgments

This work was partly supported by Grants-in-Aid for Scientific Research of JSPS (No.18500230).

References

[1] M. Casdagli, Nonlinear prediction of chaotic time series, *Physica D: Nonlinear Phenomena*, Vol. 35, Issue 3, pp.335-356, 1989

[2] K. A. de Oliveira, A. Vannucci, E. C. da Silva, Using artificial neural networks to forecast chaotic time series, *Physica A*, No.284, pp.393-404, 1996

[3] H. Leung, T. Lo, S. Wang, Prediction of Noisy Chaotic Time Series Using an Optimal Radial Basis Function, *IEEE Trans. on Neural Networks*, Vol. 12, No.5, pp.1163-1172, 2001

[4] V. Kodogiannis, A. Lolis, Forecasting Financial Time Series using Neural Network and Fuzzy System-based Techniques, *Neural Computing and Applications*, No.11, pp.90-102, 2002

[5] Kuremoto T., Obayashi M., Yamamoto A., and Kobayashi K., Neural Prediction of Chaotic Time Series Using Stochastic Gradient Ascent Algorithm. *Proceedings of the 35th ISCIE International Symposium on Stochastic Systems Theory and Its Applications (SSS'03)*, pp.17-22, 2003

[6] Kuremoto T., Obayashi M., Yamamoto A., and Kobayashi K., Predicting Chaotic Time Series by Reinforcement Learning. *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS2003)*, CD-ROM, 2003

[7] R.S.Sutton and A.G. Barto, *Reinforcement Learning: An introduction*, The MIT Press, 1998

[8] H. Kimura, S. Kobayashi, Reinforcement Learning for Continuous Action using Stochastic Gradient Ascent , *Intelligent Autonomous Systems*, Vol.5, pp.288-295, 1998

[9] Kuremoto T., Kobayashi K., Obayashi M., Nonlinear Prediction by Reinforcement Learning. *Lecture Notes in Computer Science (ICIC 2005)*, Vol.3644, pp.1085-1094, 2005

[10] http://www.neural-forecasting-competition.com/

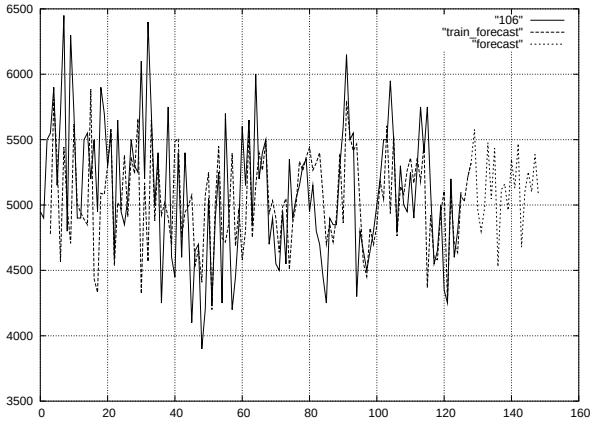

Fig. 11. Forecasting results of time series 106.

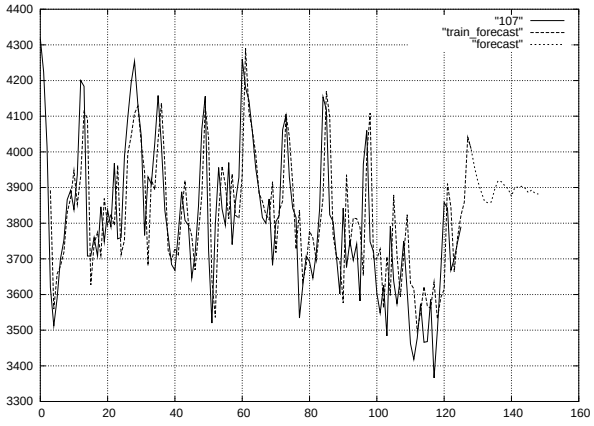

Fig. 12. Forecasting results of time series 107.

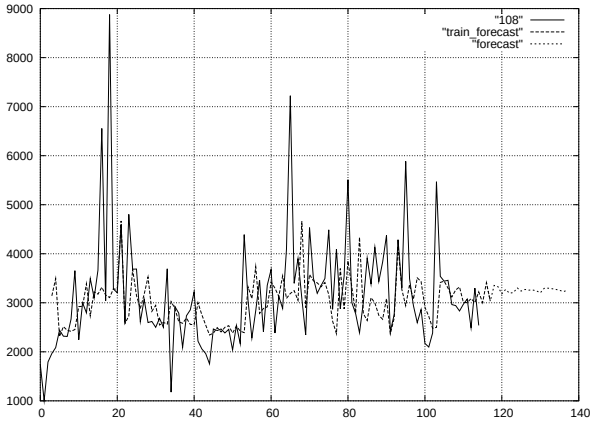

Fig. 13. Forecasting results of time series 108.

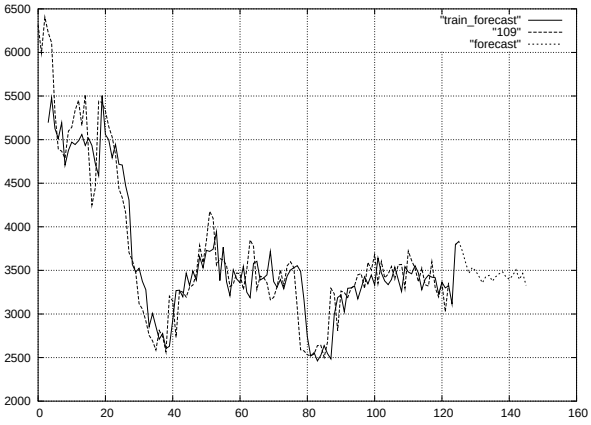

Fig. 14. Forecasting results of time series 109.

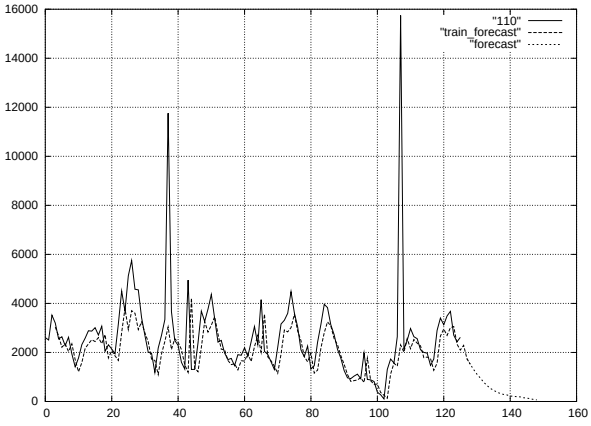

Fig. 15. Forecasting results of time series 110.

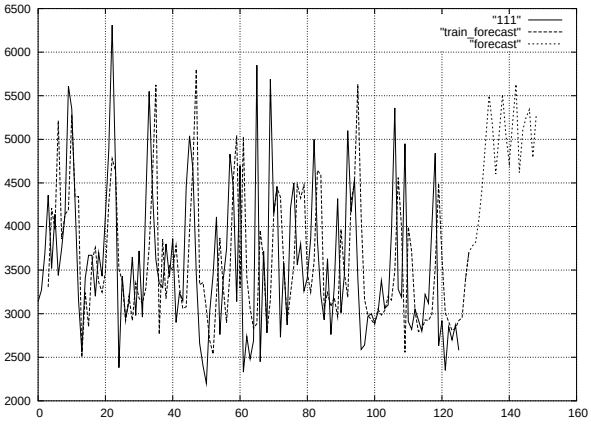

Fig. 16. Forecasting results of time series 111.