Parameter estimation of general regression neural network using Bayesian approach

Conference Paper · August 2015

DOI: 10.1063/1.4940858

6 authors, including:

Achmad Syahrul Choir
Politeknik Statistika STIS
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Rindang Bangun Prasetyo
Statistics Indonesia
6 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Brodjol Sutijo Suprih Ulama
Institut Teknologi Sepuluh Nopember
12 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Nur Iriawan
Institut Teknologi Sepuluh Nopember
75 PUBLICATIONS **119** CITATIONS
SEE PROFILE

Parameter estimation of general regression neural network using Bayesian approach

Achmad Syahrul Choir, Rindang Bangun Prasetyo, Brodjol Sutijo Suprih Ulama, Nur Iriawan, Kartika Fitriasari, and Mohammad Dokhi

Citation: AIP Conference Proceedings **1707**, 080001 (2016); doi: 10.1063/1.4940858
View online: http://dx.doi.org/10.1063/1.4940858
View Table of Contents: http://aip.scitation.org/toc/apc/1707/1
Published by the American Institute of Physics

Parameter Estimation of General Regression Neural Network Using Bayesian Approach

Achmad Syahrul Choir[1, 2], Rindang Bangun Prasetyo[3, 4], Brodjol Sutijo Suprih Ulama[5], Nur Iriawan[6], Kartika Fitriasari[7], Mohammad Dokhi[8]

[1] *Statistics Department, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, East Java, Indonesia*
Email: madsyair@bps.go.id
[2]*Statistics Indonesia, Jakarta, Indonesia.*
[3]*Statistics Department, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, East Java, Indonesia*
Email: rindang@bps.go.id
[4]*Statistics Indonesia, Jakarta, Indonesia.*
[5]*Statistics Department, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, East Java, Indonesia.*
Email: brodjol_su@statistika.its.ac.id
[6]*Statistics Department, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, East Java, Indonesia.*
Email: nur_i@statistika.its.ac.id
[7]*Statistics Department, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, East Java, Indonesia.*
kartika_f@statistika.its.ac.id
[8]*Statistics Department, Institute of Statistics (STIS,) Jakarta, Indonesia.*
dokhi@stis.ac.id

Abstract. General Regression Neural Network (GRNN) has been applied in a large number of forecasting/prediction problem. Generally, there are two types of GRNN: GRNN which is based on kernel density; and Mixture Based GRNN (MBGRNN) which is based on adaptive mixture model. The main problem on GRNN modeling lays on how its parameters were estimated. In this paper, we propose Bayesian approach and its computation using Markov Chain Monte Carlo (MCMC) algorithms for estimating the MBGRNN parameters. This method is applied in simulation study. In this study, its performances are measured by using MAPE, MAE and RMSE. The application of Bayesian method to estimate MBGRNN parameters using MCMC is straightforward but it needs much iteration to achieve convergence.

Keywords: Bayesian, MCMC, MBGRNN, Mixture, BUGS
PACS:02.50.Tt

INTRODUCTION

Bayesian approach has been applied to estimate parameter of Neural Network (NN) model for more than two decades. Bayesian method initially was used on Feed-Forward Neural Network (FFNN) [1-3] and furthermore was applied to Radial Basis Function (RBF) [4,5]. The utilization of Bayesian approach for NN has succeeded to solve a large number of classification and forecasting problems [6,7].

The Bayesian method are also successfully applied to Nadaraya-Watson Regression [8,9] which is similar to General Regression Neural Network (GRNN) [10]. The GRNN was extended based on mixture distribution [11].

Proceedings of The 7th SEAMS UGM International Conference on Mathematics and Its Applications 2015
AIP Conf. Proc. 1707, 080001-1–080001-7; doi: 10.1063/1.4940858

We call it as Mixture-Based GRNN (MBGRNN).The benefit of this method is that naturally less node than GRNN in first hidden layer.

Considering the advantage of MBGRNN, the aims of this paper are to describe Bayesian approach for estimating MBGRNN parameters. Bayesian theorem is applied to construct posterior parameter distribution from likelihood and prior. We use simulation studies to meassure our proposed method and employing BUGS (Bayesian Using Gibbs Sampling) for its computation.

GENERAL REGRESSION NEURAL NETWORK

The GRNN is another term for Nadaraya-Watson kernel regression on neural network context which proposed by Specht [10]. It has relationship with RBF and has 4 layers as shown in Fig. 1 (b): input layer, 2 hidden layer (pattern layer, summation layer), and output layer. Number of node in input layer is depend on number of predictor variable. Input layer is fully linked to pattern layer where number of node in pattern layer equal to number of elements in dataset. The linkage weight between input layer and pattern layer is smoothing parameter (σ). Commonly, node in pattern layer use Gaussian basis function. Furthermore, y_k connect kth node in pattern layer and numerator node in summation layer. However, the weight that connect pattern layer and denominator node set to 1. The node in output layer receives result from summation layer and calculates the output. The GRNN given by following formula

$$m(\mathbf{x}) = \frac{\sum_{k=1}^{n} y_k \exp\left(-0.5\left(\mathbf{x}-\mathbf{x}_k\right)\boldsymbol{\Sigma}^{-1}\left(\mathbf{x}-\mathbf{x}_k\right)'\right)}{\sum_{k=1}^{n} \exp\left(-0.5\left(\mathbf{x}-\mathbf{x}_k\right)\boldsymbol{\Sigma}^{-1}\left(\mathbf{x}-\mathbf{x}_k\right)'\right)} \tag{1}$$

where y_k is kth element of response data , $\mathbf{x}_k$ is kth element of predictor data,

$$\boldsymbol{\Sigma}_{p\times p} = \begin{bmatrix} \sigma^2 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \sigma^2 \end{bmatrix}$$

$\boldsymbol{\Sigma}$ is smoothing factor matrix.

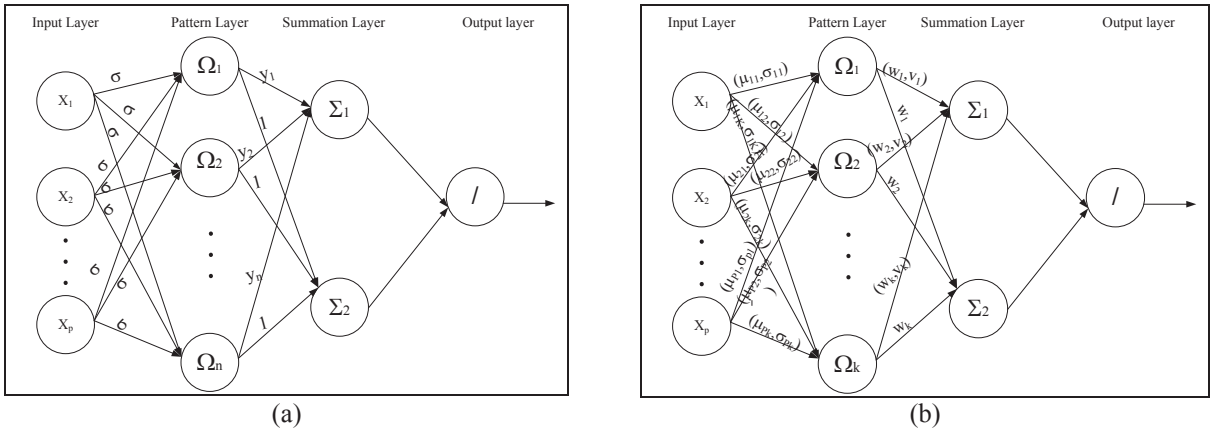

FIGURE 1. Architecture of GRNN: (a) GRNN, (b) MBGRNN

Eq. (1) can be extended by replacing kernel estimator with adaptive mixture model [11]. The MBGRNN is denoted by

$$m(\mathbf{x}) = \frac{\sum_{k=1}^{K} \mathrm{v}_k \mathrm{w}_k \exp\left(-0.5(\mathbf{x}-\boldsymbol{\mu}_k)\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)'\right)}{\sum_{k=1}^{K} \mathrm{w}_k \exp\left(-0.5(\mathbf{x}-\boldsymbol{\mu}_k)\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)'\right)} \tag{2}$$

where $\boldsymbol{\mu}_k = \begin{bmatrix} \mu_{1k} & \mu_{2k} & \ldots & \mu_{pk} \end{bmatrix}$, $w_k = \frac{u_k}{\sum_k^K u_k}$, u_k has positive value,

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} \sigma^2 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \sigma^2 \end{bmatrix} \tag{3}$$

and K is number of node in pattern layer. Considering that adaptive σ is better than constant one [12], we use adaptive parameter σ_k, denoted by:

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} {\sigma_{1k}}^2 & 0 & 0 & 0 \\ 0 & {\sigma_{2k}}^2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & {\sigma_{pk}}^2 \end{bmatrix} \tag{4}$$

The MBGRNN model is showed in Fig. 1. (b).

BAYESIAN METHOD FOR GENERAL REGRESSION NEURAL NETWORK

Bayesian approach considered all uncertainty phenomena should be taken into model using probability, and statistical inference should be logical conclusions in the light of the law of probability [13]. It considers model parameters as random variables. The knowledge of parameter preceding data was referred to prior. The prior was then being updated with data by using Bayes' theorem to obtain posterior which is explained by following formula:

$$p(\theta \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \theta)\, p(\theta)}{p(\mathbf{y})} \tag{5}$$

$$p(\theta \mid \mathbf{y}) \propto p(\mathbf{y} \mid \theta)\, p(\theta) \tag{6}$$

where $p(\mathbf{y} \mid \theta)$ is likelihood, $p(\theta)$ is prior distribution and $p(\theta \mid \mathbf{y})$ is posterior distribution.

Let

$$y = m(\mathbf{x}) + e \tag{7}$$

where $e \sim N(0, \tau^{-1})$ and $m(\mathbf{x})$ are Eq. (2). In this paper, we considered using following prior for each parameter:

$$\tau \sim \text{gamma}(\alpha_0 / 2, \beta_0 / 2) \tag{8}$$

$$\mu_{jk} \sim N(\xi_0, \psi_0) \tag{9}$$

$${\sigma^2}_{jk} \sim U(a, b) \tag{10}$$

$$v_k \sim N(\gamma_0, \eta_0) \tag{11}$$

$$u_k \sim \text{gamma}(\delta_0, 1) \tag{12}$$

The likelihood for this model is

$$l(\boldsymbol{\theta} \mid y_1, y_2, \ldots, y_{n,}\mathbf{x}) = \left(2\pi\tau^{-1}\right)^{-\frac{n}{2}} \exp\left(-\frac{\tau}{2}\sum_{i=1}^{n}\left(y_i - m(\mathbf{x}_i)\right)^2\right) \tag{13}$$

where $\boldsymbol{\theta} = (\tau, \boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{v}, \mathbf{u})$.

The posterior distribution was obtain by Bayes' Theorem

$$p(\boldsymbol{\theta} \mid y_1, y_2, ..., y_{n,} \mathbf{x}) \propto \tau^{\frac{n}{2}} \exp\left(-\frac{\tau}{2} \sum_{i=1}^{n} \left(y_i - m(\mathbf{x}_i) \right)^2 \right) \tau^{\alpha_0/2+1} \exp\left(-\frac{\tau \beta_0}{2} \right) \prod_{k=1}^{k} \prod_{j=1}^{p} \exp\left(-\frac{\psi_0}{2} \left(\mu_{jk} - \xi_0 \right)^2 \right) \times$$
$$\prod_{k=1}^{K} \exp\left(-\frac{\eta}{2} \left(v_k - \gamma_0 \right)^2 \right) \prod_{k=1}^{K} u_k^{\delta_0 + 1} \exp\left(-u_k \right) \tag{14}$$

Full conditional posterior distribution derived from posterior distribution were:

$$p\left(\tau \mid . \right) \propto \tau^{(\alpha_0 + n)/2+1} \exp\left(-\frac{\tau}{2} \left(\beta_0 + \sum_{i=1}^{n} \left(y_i - m(\mathbf{x}_i) \right)^2 \right) \right) \tag{15}$$

$$p(\mu_{jk} \mid .) \propto \exp\left(-\frac{\tau}{2} \sum_{i=1}^{n} \left(y_i - m(\mathbf{x}_i) \right)^2 - \frac{\psi_0}{2} \left(\mu_{jk} - \xi_0 \right)^2 \right) \tag{16}$$

$$p(\sigma^2{}_{jk} \mid .) \propto \exp\left(-\frac{\tau}{2} \sum_{i=1}^{n} \left(y_i - m(\mathbf{x}_i) \right)^2 \right) \tag{17}$$

$$p(v_k \mid .) \propto \exp\left(-\frac{\tau}{2} \sum_{i=1}^{n} \left(y_i - m(\mathbf{x}_i) \right)^2 - \frac{\eta_0}{2} \left(v_k - \gamma_0 \right)^2 \right) \tag{18}$$

$$p(u_k \mid .) \propto u_k^{\delta_0 + 1} \exp\left(-u_k - \frac{\tau}{2} \sum_{i=1}^{n} \left(y_i - m(\mathbf{x}_i) \right)^2 \right) \tag{19}$$

IMPLEMENTATION IN BUGS

The BUGS Project was developed based on Gibbs Sampler algorithm. Its engine applied an "expert system" to determine which available MCMC algorithm were appropriate for a given problem. The expert systems classify full conditional distribution of each node, or block of node within the specified graphical model. When the distribution of each node is a closed form, parameter drawn from certain density type (normal, gamma, etc.). Alternatively, if a closed form does not exist, the expert system can choose appropriate sampling algorithm (adaptive rejection sampling, metropolis, slice sampler, etc) [14].

Gibbs Sampler [15] is special case of Blockwise Metropolis Hasting Sampler with an acceptance probability equal to one [16]. Therefore the proposed move is accepted in all iterations. One advantage of Gibbs Sampler algorithm is the random value can be generated from a one-dimensional distribution for which a various existing computational tools [17].

The algorithm is summarized by following steps [17]:

1. Set initial value $\boldsymbol{\theta}^{(0)}$
2. Repeat the following steps for t=1,2,…,T
   i. Set $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t-1)}$
   ii. For j=1,2,…,d , generate θ_j from $\theta_j \sim f\left(\theta_j \mid \boldsymbol{\theta}_{-j}, \mathbf{y} \right)$, where

$$f\left(\theta_j \mid \boldsymbol{\theta}_{-j}, \mathbf{y} \right) = f\left(\theta_j \mid \theta_1^{(t)}, ..., \theta_{j-1}{}^{(t)}, \theta_{j+1}{}^{(t-1)}, ..., \theta_d{}^{(t-1)}, \mathbf{y} \right) \tag{20}$$

   iii. Set $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}$ and save it as the generated set of value at t+1iteration of algorithm.

There are several softwares that implement BUGS: WinBUGS, OpenBUGS, and Just Another Gibbs Sampling (JAGS). WinBUGS is first BUGS software that runs in Microsoft Windows. It has graphical user interface for BUGS. Unfortunately, WinBUGS was no longer developed. Development of BUGS software is continuously implemented on OpenBUGS which based on open source principle. It has portability and better communication to other software. If WinBUGS and OpenBUGS were written in Component Pascal Programming language, JAGS was written in C++. JAGS could run either in Windows, Linux or Mac OS X. It can run in parallel computation when be combined with R-packages, therefore it can run faster than the others[14].

The implementation of Bayesian GRNN computation in BUGS syntax shows in following steps:

1. Model GRNN is created as Eq. (2),

2. Likelihood is specified from distribution of y. In this paper, $y \sim N(m(\mathbf{x}), \tau^{-1})$,
3. Prior is specified as Eq. (8) – (12).
4. BUGS syntax created based on model, likelihood and prior in step 1−3.
5. BUGS syntax is running using BUGS software: WinBUGS, OpenBUGS or JAGS.
6. Convergence of parameter outputs are checked.
7. We make inference parameter based on the parameter output.

SIMULATION STUDY

We applied Bayesian method on MBGRNN by using simulation studies. Our results is determined by the following step:

1. Bayesian method can be implemented to small data. Therefore, in this study,we generated 29 data point from uniform distribution for explanatory variable **x**. While the response variable y was calculated by
$$y = \exp(x_1 + x_2 + x_3 + x_4 + x_5) + \varepsilon \tag{21}$$
where ε is generated from N(0,0.5).
2. Next, we split them into 2 parts: 25 data point were used for training and the other 4 data point is used for validation. These data was applied to several MBGRNN architectures whereas each has 3 to 9 node in pattern layer.
3. Further, we made pre-processing data by transformation using standardized.
4. Later, standardized data training were clustered by using K-mean where numbers of clustered were based on the numbers of node in pattern layer for each architectures. The results were used as prior for MBGRNN
5. We run BUGS for MBGRNN using software Just Another Gibbs Sampler (JAGS) and R.
6. Post-processing data were used for data prediction.
7. Performance of each MBGRNN architectures were compared using 3 criteria: Root Mean Square Error (RMSE):
$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_{i(pred)}\right)^2}{n}}, \tag{22}$$
Mean Absolute Error (MAE):
$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_{i(pred)}\right|, \tag{23}$$
and Mean Absolute Percentage Error (MAPE):
$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_{i(pred)}}{y_i}\right| \tag{24}$$
where
$$\hat{y}_{pred} = E\left[y_{pred} \mid \mathbf{x}, \boldsymbol{\theta}\right] = \int m(\mathbf{x}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid D) \tag{25}$$
where D is training data $\{\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n\}$ .This is approximated using sample value, $\boldsymbol{\theta}^{(t)}$, which are generated from posterior distribution of parameters, denoted by
$$\hat{y}_{pred} \approx \frac{1}{n}\sum_{t=1}^{N} m\left(\mathbf{x}, \boldsymbol{\theta}^{(t)}\right) \tag{26}$$

DISCUSSION

We ran BUGS by using JAGS and R software on all architecture. We ran JAGS with 5000 iteration for burn-in and then ran it by using “thin” 100 and 1.000.000 iteration to achieve convergence. Performance criterion was then computed for data model and validation model. Table 1. show the result of our experiment. MAE of training data tends to decrease not only for MBGRNN architectures with 3 to 9 node in pattern layer but also occurred in MAPE

and RMSE training data. Those MBGRNN performance pattern of validation data was decreased until GRNN architecture increased from 6 to 9 nodes. For these criteria, 5 nodes in first hidden layer was shown as best MBGRNN architecture on this study. Therefore, the numbers of nodes in pattern layer were less than the number of data. Parameter estimation of MBGRNN with 5 nodes in pattern layer showed in Table 2.

TABLE 1. Performance of Bayesian GRNN in Various Architecture after Convergence

Performance	*Number of Node in Patter Layer*						
Criteria	3	4	5	6	7	8	9
MAE Training	4.63	4.41	4.26	4.01	3.86	3.78	3.74
MAE Validation	12.10	12.08	11.81	12.17	12.18	12.37	12.3
MAPE Training	0.65	0.62	0.59	0.55	0.53	0.52	0.52
MAPE Validation	0.5889	0.5766	0.5720	0.5819	0.5833	0.5879	0.5866
RMSE Training	5.740	5.466	5.277	4.973	4.78	4.676	4.633
RMSE Validation	15.06	15.09	14.61	15.02	15.03	15.21	15.12

TABLE 2. BUGS output for MBGRNN parameters

Parameter	Estimation	Credible Interval		Parameter	Estimation	Credible Interval	
	Value	2.5%	97.5%		Value	2.5%	97.5%
τ	9.263	2.809	20.82	σ_{11}	0.2143	0.1014	0.7235
μ_{11}	1.03	-0.466	2.53	σ_{12}	0.2329	0.1016	0.8201
μ_{12}	-0.6017	-2.087	0.8124	σ_{13}	0.1879	0.1011	0.5825
μ_{13}	0.4099	-1.012	1.838	σ_{14}	0.1796	0.1011	0.521
μ_{14}	-0.1274	-1.481	1.167	σ_{15}	0.1692	0.1009	0.4734
μ_{15}	-0.3861	-1.913	1.28	σ_{21}	0.2146	0.1016	0.7144
μ_{21}	0.3497	-1.028	1.789	σ_{22}	0.2244	0.1016	0.7708
μ_{22}	0.9783	-0.4072	2.418	σ_{23}	0.2173	0.1016	0.7137
μ_{23}	-1.096	-2.61	0.3143	σ_{24}	0.2012	0.1014	0.6436
μ_{24}	-0.2341	-1.597	1.243	σ_{25}	0.1943	0.1011	0.6202
μ_{25}	-0.1724	-1.598	1.151	σ_{31}	0.1937	0.1013	0.5932
μ_{31}	-0.9211	-2.171	0.4023	σ_{32}	0.2132	0.1016	0.6957
μ_{32}	-0.2778	-1.577	1.124	σ_{33}	0.2129	0.1016	0.6852
μ_{33}	-0.7517	-2.062	0.5843	σ_{34}	0.2016	0.1014	0.6594
μ_{34}	0.133	-1.189	1.688	σ_{35}	0.278	0.1017	1.077
μ_{35}	1.105	-0.3585	2.56	σ_{41}	0.1799	0.1011	0.5266
μ_{41}	0.3214	-1.163	1.65	σ_{42}	0.2165	0.1016	0.7269
μ_{42}	0.7411	-0.57	2.18	σ_{43}	0.1971	0.1013	0.6276
μ_{43}	0.2659	-1.208	1.739	σ_{44}	0.2875	0.1022	1.076
μ_{44}	-1.15	-2.701	0.4003	σ_{45}	0.168	0.1009	0.4897
μ_{45}	0.515	-1.154	1.989	σ_{51}	0.1854	0.1009	0.5934
μ_{51}	-0.3099	-2.153	1.162	σ_{52}	0.1982	0.1012	0.6312
μ_{52}	-0.07276	-1.454	1.17	σ_{53}	0.1702	0.101	0.4683
μ_{53}	1.247	-0.1493	2.549	σ_{54}	0.1922	0.1014	0.5852
μ_{54}	0.2461	-1.167	1.468	σ_{55}	0.1607	0.101	0.4193
μ_{55}	0.5859	-0.9446	1.987				

TABLE 2. Continued

Parameter	Estimation Value	Credible Interval 2.5%	Credible Interval 97.5%	Parameter	Estimation Value	Credible Interval 2.5%	Credible Interval 97.5%
v_1	0.0724	-0.8589	2.379	w_1	0.199	0.0065	0.596
v_2	-0.2739	-0.9785	0.441	w_2	0.2021	0.006451	0.6039
v_3	-0.509	-1.014	0.137	w_3	0.2013	0.006459	0.6036
v_4	0.676	-0.892	2.531	w_4	0.2002	0.006448	0.6037
v_5	1.265	-0.7384	2.552	w_5	0.1973	0.006238	0.5985

CONCLUSION

We show that Bayesian approach can be implemented for estimating MBGRNN parameter in small data. BUGS language makes its computation in MBGRNN model easier to be implemented. However, we note that BUGS needs many iteration for achieve convergent. MBGRNN have good performance even have a little node in pattern layer.

ACKNOWLEDGMENTS

The research presented in this paper was partially supported by a grant of Competence Grant 2015, Directorate General of Higher Education (DIKTI), Ministry of Research, Technology and Higher Education, Republic of Indonesia. The authors thank Yuna Puteri Kadarisman and Erie Sadewo for some comments and corrections. Two of authors thank Statistics Indonesia for funding PhD study.

REFERENCES

1. W.L. Buntine. and A.S Weigand., Complex Systems, **5**, 603-643 (1991).
2. D.J.C Mackay, Neural Computation, **4**, 448-472 (1992).
3. R.M.Neal, Bayesian Learning for Neural Network, (Springer, New York, 1996).
4. C.C.Holmes and B.K. Mallick, Neural Computing, **10**, 1217-1233 (1998).
5. C. Andrieu., N. de Freitas, and A. Doucet, Neural Computing, **13**, 2359-2407 (2001).
6. J. Lampinen and A. Vehtari, Neural Networks, **14**, 7-24 (2001).
7. D.M. Titterington, Statistical Science, **19**, 128–139 (2004).
8. X. Zhang, X., R.D. Brooks, and M.L. King, Journal of Econometrics, **153**, 21-32 (2009).
9. H.L. Shang, Computational Statistics, **29**, 829-84 (2014).
10. D.F. Specht, IEEE Trans. on Neural Network, **2**, 568-576 (1991).
11. C.M. Bishop, Neural Networks for Pattern Recognition, (Oxford University Press, New York,1995).
12. D. Tomandl, D. and A. Schober., Neural Networks, **14**, 1023-1034 (2001).
13. R. Christensen, W. Johnson, A. Branscum., and T.E. Hanson, Bayesian Ideas and Data Analysis: An Introduction for Scientists and Statisticians, (CRC Press, Boca Raton, 2011).
14. D. Lunn,.,C. Jackson., N. Best, A. Thomas, and D. Spiegelhalter, The BUGS Book: A Practical Introduction to Bayesian Analysis, (CRC Press, Boca Raton, 2013).
15. S. Geman, and D. Geman, IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-**6**, 721-740 (1984).
16. W.M. Bolstad, Understanding Computational Bayesian Statistics, (John Wiley and Son, New Jersey, 2010).
17. I. Ntzoufras, I., Bayesian Modeling Using Winbugs. (John Wiley and Son, New Jersey, 2009).