ISBN: 978-979-792-552-9

A Comparison of Radial Basis Probabilistic Neural Network and Radial Basis Function Neural Network Performance Based on Sensitivity Analysis

Hasanuddin[1,2]

[1] Mathematics Education Department
Universitas Islam Negeri Sultan Syarif Kasim Riau, Pekanbaru 28293

[2] Institute for Sciences and Learning Development, Pekanbaru, 28293

hasanuddin@uin-suska.ac.id

Abstrak

This paper presents a comparative study of the performance learning algorithm for Radial Basis Probabilistic Neural Network (RBPNN), and the Radial Basis Function Neural Network (RBFNN), are evaluated and compared for their ability to classify data based on sensitivity analysis. RBPNN generally performs similarly to RBFNN. Both of them are trained using gradient descent. In this research, sensitivity analysis is used to prune the feature data. The results show that the network still works well after pruning. The issues of network optimization and computational efficiency in use are discussed. Finally, to evaluate the performance, our experiments are demonstrated by two examples of real life data set.

Kata kunci: RBPNN, RBFNN, pruning criteria, sensitivity analysis, classification

1 Pendahuluan

The comparison of the Radial Basis Probabilistic Neural Networks (RBPNN) and Radial Basis Function Neural Network (RBFNN) are discussed in this paper. The construction of a RBPNN involves four different layers: one input layer, two hidden layers and one output layer. The first hidden layer is nonlinear processing layer. The second hidden layer selectively sums the output of the first hidden layer, and it's generally has the same size as the output layer for a labeled pattern classification problem. The weight between the first hidden layer and the second hidden layer of the network are ones or constant (including zero)[5 Huang D.S. and Zhao W.B.]. RBPNN have been applied to a wide variety of different areas including human face recognition. In the other side, RBFNN consists of three layers, namely input layer, hidden layer and

ISBN: 978-979-792-552-9

output layer [3 Du, K.-L. and Swamy, M.N.S].

The performance of the RBPNN and RBPNN depends on the number and position of centers, spreads, and the method used for learning the input-output mapping. The existing learning strategy can be classified as follows: 1) strategies selecting the centers randomly from the training data 2) Strategy employing unsupervised procedure for selecting centers 3) strategy supervised procedure for selecting the centers [6 Karayiannis, N.B]. There are many alternative learning method for neural networks. In the case of multilayer networks the first succesful algorithm was the classical backpropogation [1 Castillo, E., Guijarro-Berdinas, B., Fontenla- Romero, O., and Alonso-Betanzos, A].

The simplication of NN have been proposed, such as improving the training algortihm, sensitivity based feature selection and sensitivity analysis of the centers of the hidden neurons [9 **Wang, X.-Z., Li, C.-G., Yeung, D.S., Song, S.J. and Feng, H.M**]. The sensitivity based network simplification is one of the most effective and most promising techniques. Sensitivity analysis of features is a fundamental issue in neural network design. It can be used as the means of feature selection, simplifying the neural network and improving the generalization performance [9 **Wang, X.-Z., Li, C.-G., Yeung, D.S., Song, S.J. and Feng, H.M**]. The Sensitivity analysis of the input is an efficient way to simplify the structure of the RBPNN [4 Hasanuddin, Irawan, MI].

Selection of the center for the hidden neurons of RBFNN and RBPNN follow the model in determining the center on PNN, using all the training samples as a center. These methods inefficient because it causes the network too fit, network generalization to be low. Furthermore, the development of methods of determining the center that involves training the model using Recursive Orthogonal Least Square (ROLSA). Method of determining center for RBFNN can also be done on RBPNN. One method used to deterine the center of RBFNN using the techniques clutering [Huang, DS and Zhao, W.B.]. Clustering is a data analysis tool to characterize the distribution of the data set. This technique will be applied to determine the center of RBFNN. A set of training grouped into appropriate clusters, in which the prototype was then used as a center RBFNN.

This paper is organized as follows. Section 2 present a supervised learning algorithm for RBPNN based on gradient descent. Section 3 present sensitivity measures to input over training dataset and rule for selecting the redunandt feature. Section 4 evaluates the performance of the RBFNN and RBPNN and trained using the proposed learning algortihm and Section 5 contains concluding remarks.

2 Determining Centers, Spread and Weight

One method is widely accepted because of its ability to provide good results is the K-means clustering. The basic idea of this algorithm is to collect the data into several groups and choose a center based on the size of the centers by using Euclid distance. Furthermore, each associated with a cluster point Gauss on RBFNN hidden neurons. The above considerations, this study uses k-means clustering to determine the center.

ISBN: 978-979-792-552-9

The basic idea of the algorithm is to collect the data into several groups and choose a center based on the size of the centers by using Euclid distance. Furthermore, every point cluster (center) associated with Gauss hidden neurons in the network. RBFNN and RBPNN that uses Gaussian kernel also use the spread. Spread can be selected by using the average of all the Euclidean distance between the centers with its immediate environment. in this study, a simple method used in determining the spread as follows

$$\sigma = \frac{d_{\max}}{\sqrt{K}} \tag{1}$$

where $d_{\max}$ is maximum euclidean distance of learning dataset and K is total point of dataset.

RBFNN weights determined by Eq. (2). This method is based on the gradient decent. Suppose RBFNN has s output, then the output vector $(y_1, y_2, ..., y_s)$ and $(w_{k1}, w_{k2}, w_{k3}, ..., w_{kn})^T$ is expressed as a weight vector for the k-th output. so each output vector can be written as follows

$$y_k = \sum_{j=1}^{m} w_{kj} \cdot \exp\left(\sum_{i=1}^{n} \frac{(x_i - u_{ji})^2}{-2\sigma_j^2} \right), \; k = 1, 2, ..., s \tag{2}$$

Where n is number of input features, m is number of RBFNN center, x_i is input vektor.

RBPNN learning can be viewed as a mapping vektor $\boldsymbol{x}_l \in R^N$ to vektor $\boldsymbol{y}_l \in R^M$, thus forming the input-output pairs $(\boldsymbol{x}_l, \boldsymbol{y}_l)$, $l = 1,2,...,L$. So, for every input vektor $\boldsymbol{x}$ can be written

$$\hat{y}_i = \sum_{k=1}^{M} w_{ik} h_k(\boldsymbol{x}), \; i = 1, 2, ..., M, \tag{3}$$

Where $h_k(\boldsymbol{x})$ is output neuron in the second hidden layer , that is

$$h_k(\boldsymbol{x}) = \sum_{i=1}^{n_k} \phi_i(\boldsymbol{x}, \boldsymbol{c}_{ki}) = \sum_{i=1}^{n_k} \phi_i\left(\left\| \boldsymbol{x} - \boldsymbol{c}_{ki} \right\|_2 \right), \; k = 1, ..., M,$$

3 Sensitivity Analysis

Response sensitivity of function f according to variabel $x = [x_1, x_2, ..., x_N]$ given by its gradient. So, input sensitivity can be defined as vektor of

$$\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_N} \right) \tag{4}$$

Sensitivity in eq. (4) can be considered as as a sensitivity coeffisient. Especially, output sensitivities y_m wich refers to the input x_n

ISBN: 978-979-792-552-9

$$s_{x_n}^{y_m} = \frac{\partial y_m}{\partial x_n},$$

If the output of the network ismultiple, then the sensitivities can be written as jacobian matrix as follow

$$J\left(\mathbf{x}^{(l)}\right) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_N} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial y_M}{\partial x_1} & \frac{\partial y_M}{\partial x_2} & \cdots & \frac{\partial y_M}{\partial x_N} \end{bmatrix}, \quad (5)$$

Where $x^{(l)}$ are input points.

a) Sensitivity measures of RBFNN

Considering RBFNN with multiple outputs. Assuming that the RBFNN has *M* output, then the output vektor can be expressed by $(y_1, y_2, ..., y_M)$. Thus, each output expressed by

$$y_k = \sum_{j=1}^{m} w_{kj} \cdot \exp\left(\sum_{i=1}^{n} \frac{\left(x_i - u_{ji}\right)^2}{-2\sigma_j^2} \right), \text{ k = 1, 2, ..., } s \quad (6)$$

The partial derivatives of Eq. 6 can be written as follow

$$s_{x_i}^{y_k} = \frac{\partial y_k}{\partial x_i} = \sum_{j=1}^{M} w_{kj} \cdot \left(\exp\left(\frac{\sum_{i=1}^{N} \left(x_i - u_{ji}\right)^2}{-2\sigma_j^2} \right) \right) \frac{\left(x_i - C_{kj}\right)}{-\sigma_j^2}, \quad (7)$$

Eq.(7) used to calculated sensitivities of *m*-th output wich refers to *n*-th input.

b) Sensitivity measures of RBPNN

Sensitivity measure of RBPNN almost similiar to the sensitivity measure of RBPNN. Assuming that the RBPNN has *M* output, then output vektor can be expresssed as $(y_1, y_2, ..., y_M)$. Thus, each output can be formulated as follow

$$y_i = \sum_{k=1}^{M} w_{ik} \sum_{i_k=1}^{n_k} \phi_k \left(\left\| \mathbf{x} - \mathbf{c}_{ki_k} \right\|_2 \right), \quad i = 1,2,...,M, \quad (8)$$

The sensitivity of RBPNN can be obtained by partial derivatives as follow

ISBN: 978-979-792-552-9

$$s_{x_n}^{y_i} = \sum_{k=1}^{M} w_{ik} \sum_{i_k=1}^{n_k} \exp\left(\frac{\sum_{t=1}^{N} \left(x_t - c_{i_k t}\right)^2}{-2\sigma_{i_k}^2} \right) \left(\frac{x_n - c_{i_k n}}{-\sigma_{i_k}^2} \right), \quad (9)$$

Where $n = 1, 2, ..., N$.

Eq. (9) used to calculate the sensitivity of *m*-th output wich refer to *n*-th input.

c) Criteria for selecting the redunandt features

Sensitivity matrix, ***S,*** used to determine which inputs have the smallest effect on output. When one or more input sensitivity is relatively small when compared with the others, then the dimension of the input to the neural network can be pruned by eliminating the input features. Furthermore, significance measures for every *n*-th input calculated by

$$S_n = \max_{m=1,...,M} \left\{ \hat{S}_{mn} \right\}, n = 1,..., N, \quad (10)$$

Where $\hat{S}_{mn}$ is sensitivity *n*-th input over *m*-th output.

After measuring the sensitivity using eq. 9, then each input sensitivity ordered by

$$S_{n_{u+1}} \le S_{n_u}, u = 1,..., N-1,$$

Where n_u is sequences of sorted input. Sensitivity gap outlined below based on the ratio of two neighboring term of sequence g_u,

$$g_u = \frac{S_{n_u}}{S_{n_{u+1}}}, u = 1,..., N-1,$$

Then the largest gap is given by the following equation:

$$g_{\max 1} = \max_u \left\{ g_u \right\}, \quad u = 1,2,..., N-1,$$

$g_{\max}$ position can be determined as effective feature that have a significant effect for the networks.

$$u_{\text{cut}} = u,$$

The second largest sensitivity gap determined by

$$g_{\max 2} = \max_{u \neq u_{cut}} \left\{ g_u \right\}, \quad u = 1,2,..., N-1,$$

If $g_{\max 1}$ largest than $g_{\max 2}$, $C g_{\max 1} > g_{\max 2}$, chose arbitrarily within reasonable range, e.g $C = 0.5$, then the input feature are smaller than $S_{n_{u_{cut}}}$ can be pruned.

4 Experiments and Results

A series of numerical simulations have been performed to evaluate the performance of RBPNN and RBFNN. The study focused on the classification accuracy and sensitivity analysis RBPNN input features. There are two datasets used in this experiment, Iris

ISBN: 978-979-792-552-9

dataset and E-Coli. Both were obtained from the UCI Machine Learning website.

a. Dataset Iris

Iris dataset has 150 points consisting of four attributes, sepal length, sepal width, petal length and petal width, the four attributes has units of cm. Iris dataset is divided into three subset: 50 points for the type Setosa, 50 points for Versicolour and 50 points for virginica. In this experiment, daata divided into two parts, 70% are used as training data while the remaining 30% is used as the test data.

Table 1 and Table 2 shown the classification performance comparison for Iris dataset identification between RBFNN and RBPNN.

Tabel 1. Classification Results using RBFNN Algorithm for Iris Dataset

Data	Amount	Classification Error	Classification Percentage
Training	105	3	97.14%
Testing	45	1	97.78%

Table 2. Classification Results using RBPNN Algorithm for Iris Dataset

Data	Amount	Classification Error	Classification Percentage
Training	105	3	97.14%
Testing	45	1	97.78%

Table 1 shown classfication results using RBFNN. For training dataset, the classification percentage achieve 97.14% and testing dataset achieve 97.78 %. Table 2 shown classfication results using RBPNN. For training dataset, the classification percentage achieve 97.14% and testing dataset achieve 97.78 %. Table 1 and Table 2 shown the classification performance of RBFNNN and RBPNN is similiar. In other word, the effectivity of RBFNN and RBPNN is equal.

Table 3. Time Effeciency and Epoch during RBFNN and RBPNN training for iris dataset

Neural Network	Time (second)	MSE	Epoch (Iteration)
RBFNN	1.7344	0.3029	820
RBPNN	0.5469	0.3473	143

Table 3 shows the comparison of time effeciency and epoch between RBFNN and RBPNN. CPU time for RBFNN is 1.7344 second, whereas 0.5469 second for RBPNN. RBFNN epoch need 820 iteration, whereas RBPNN need 143 iteration. This means RBPNN more efficient than RBFNN. MSE for RBFNN is 0.3029 more small than RBPNN MSE that is 0.3473. Figure 1 and 2 shows error convergence for RBFNN and RBPNN learning.

ISBN: 978-979-792-552-9

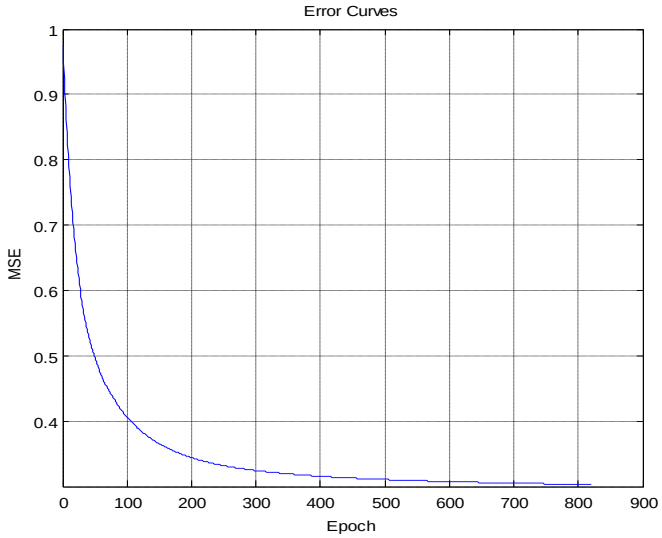


Figure 1: MSE Graphic during RBFNN Training for Iris dataset

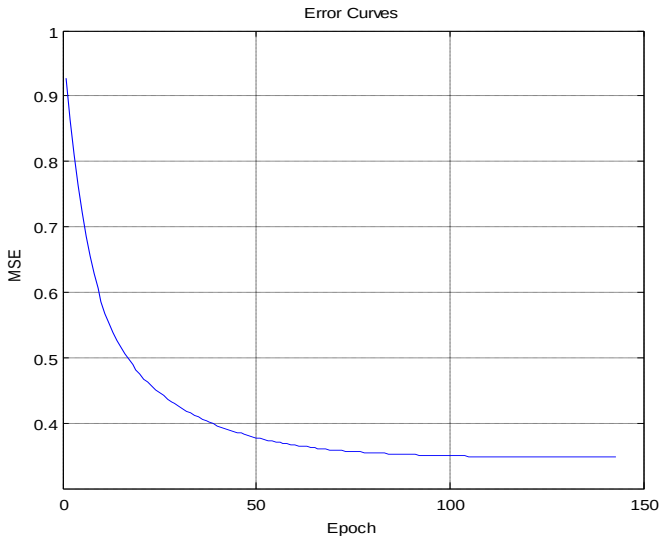
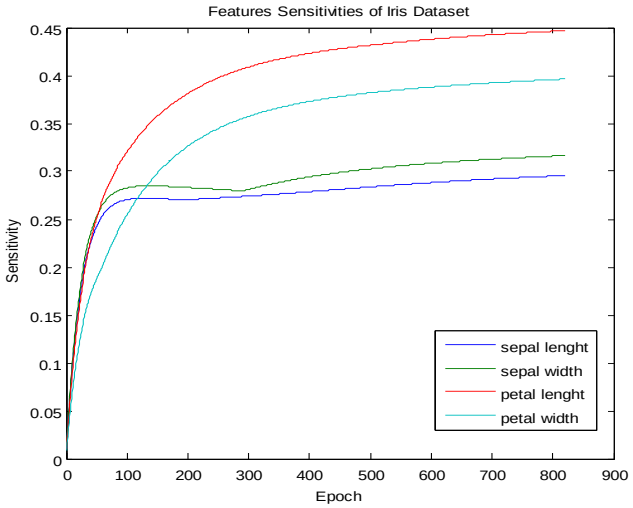
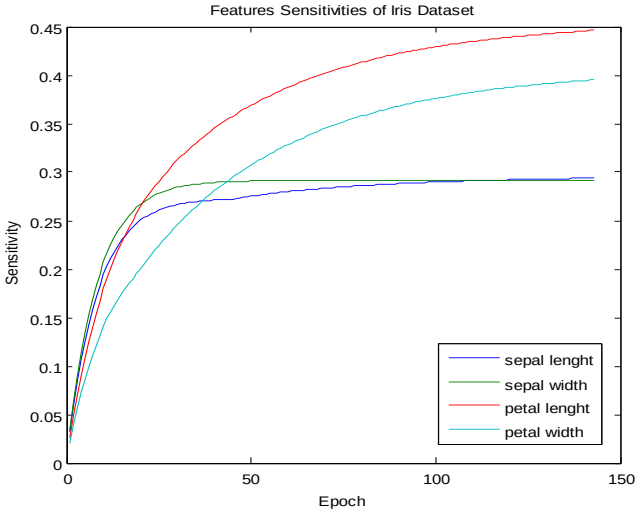


Figure 2: MSE Graphic during RBPNN Training for Iris Dataset

ISBN: 978-979-792-552-9



(a) Features Sensitivities during RBFNN Training



(b) Features Sensitivities during RBPNN Training

Figure 3: Input Significance changes during training for iris Dataset

Generally, input features sensitivities to iris dataset output fetaures can be see in the Figure 3. It can be seeen in that initial sensitivitas are low, during the training all of sensitivity increase.

Table 4 shows input feature sensitivities, input sensitivities for RBFNN, sensitivity of x_3 (petal length) is 0.4468, input feature x_4 (petal width) sensitivity is 0.3963, x_1 (sepal length) sensitivity is 0.3164 and x_2 (sepal length) is 0.2955. Input sensitivities for RBPNN, x_3 sensitivity is 0.44636, x_4 sensitivity is 0.39434, x_1 sensitivity is 0.29341 and x_2 sensitivity is 0,28656.

ISBN: 978-979-792-552-9

Tabel 4: Feature Sensitivity and Gap for RBFNN and RBPNN for Iris Dataset.

Input Feature	**RBFNN**		**RBPNN**	
	Sensitivity	**Gap**	**Sensitivity**	**Gap**
3	0.4468	1.0707	0.44636	1.131917
4	0.3963	1.1274	0.39434	1.34399
1	0.3164	1.2527	0.29341	1.023904
2	0.2955	0	0.28656	0
g_{max1}	1.1274		1.34399	
g_{max2}	1.2527		1.131917	
g_{max2}/g_{max1}	0.90000	>C	0.842206	>C

After calculation of RBFNN sensitivity gaps, the biggest gap is g_{max1} = 1.1274 and the second largest gap is g_{max2} = 1.2527. Feature reduction criteria selected is C = 0.5, then there is no feature that recomended to pruned. In other words, if the input feature reduction is done, it will cause RBFNN classification performance will decrease. as well as calculations for RBPNN, the biggest gap is g_{max1} = 1.34399 and the second largest gap is g_{max2} = 1.131917. Feature reduction criteria selected is C = 0.5, then there is no feature recomended to be pruned. In other words, if the input feature reduction is done, it will cause RBPNN classification performance will decrease.

b. Dataset E-Coli

E-coli dataset created by Kenta Nakai of the Institute of molecular and Cellular Biology, Osaka. E-coli dataset consisting of 336 data points and 7 and 8 types of input features subdata. The data are used as the training dataset is 244 points, whereas for the test dataset is 94 points (there are 2 repetition points).

E-coli dataset trained using RBFNN and RBPNN algorithm. Furthermore, each input features sensitivity analyzed, the sensitivity analysis using methods that have been formulated in the previous discussion.

Table 5: Classification Results using RBFNN Algorithm for E-Coli Dataset.

Data	Amount	Classfication Error	Classification Percentage
Training	244	38	84.43%
Testing	94	20	79,17%

ISBN: 978-979-792-552-9

Tabel 6: Classification Results using RBPNN Algorithm for E-Coli Dataset.

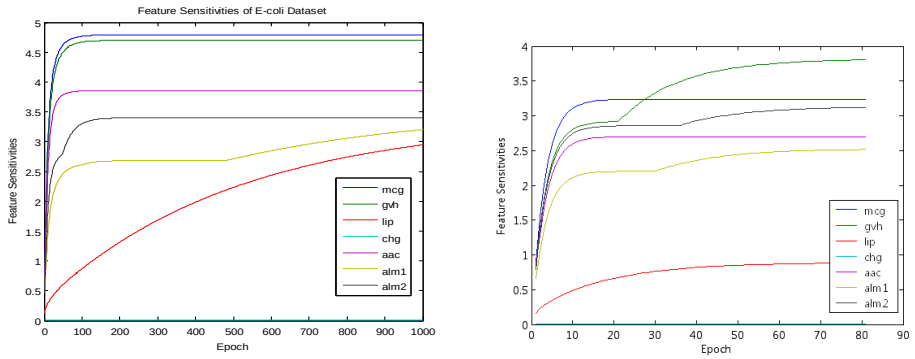
Data	Amount	Classfication Error	Classification Percentage
Training	244	41	83,20%
Testing	94	19	80,21%

Table 5 shows clasisfication results using RBFNN. For training dataset, the classification percentage achieve 84.43% and testing dataset achieve 79.17%. Table 6 shows classification results using RBPNN. For training dataset, the classification percentage achieve 83.20% and testing dataset achieve 80.21%. Table 5 and Table 6 shows the classification performance for training dataset of RBFNNN better than RBPNN performance, conversely, for testing dataset RBPNN better than RBFNN performance.

Table 7: Time Effeciency and Epoch during RBFNN and RBPNN training for iris dataset

Neural Network	Time (second)	MSE	Epoch (iteraion)
RBFNN	40.9219	0.6760	100
RBPNN	5.1094	0.7910	81

Table 7 shows the comparison between RBFNN and RBPNN, time for training RBFNN is 40.9219 seconds and to RBPNN is 5.1094 seconds. respectively, each epoch for RBFNN and RBPNN are 1000 iterations and 81 iterations This means that the RBPNN more efficient when compared with RBFNN. The RBFNN MSE is 0.6760 is smaller than the RBPNN MSE is 0.7910.



(a) **(b)**

Figure 4. (a) Input significance during RBFNN training, (b). Input significance during RBPNN training

The sensitivity of the input feature dataset with E-coli RBFNN training can be seen in Figure 4 (a). Features that have the highest sensitivity is gvh, followed by feature

ISBN: 978-979-792-552-9

mcg, alm2, aac, alm1 and lip. While chg is a feature that has the lowest sensitivity is 3.1837 x 10^{-011}. Meanwhile, the sensitivity of the input feature dataset RBPNN Training E-coli can be seen in Figure 4 (b). Features that have the highest sensitivity is gvh, followed by feature mcg, alm2, aac, alm1 and lip. While chg is a feature that has the lowest sensitivity is 3.1597 x 10^{-011}.

The sensitivity of each input features can be seen in Table 9. Gap each feature is also calculated, the biggest gap is g_{max1} = 9.2503x10^{010}, while the second largest gap is g_{max2} = 1.0872. Criteria for reduction of input feature is calculated based on the criteria presented, value obtained 1.3162x10^{-10}. There are no specific criteria of determination of the value of C, the experiment have been the value of C = 0.5. 1.3162x10^{-10} < C. u_{cut} position is determined by the position of the biggest gap, ie u_{cut} = 6. Ucut using these criteria, all input features that have sensitivity under u_{cut}, can be reduced without reducing the ability of the network performance. Table 9 shows that features 4 has a lower sensitivity when compared to other input features position.

Tabel 9: Feature Sensitivity and Gap fusing RBFNN for E-coli Dataset

Feature	**Sensitivity**	**Gap**
1	4.7890	1.0196
2	4.6968	1.2176
5	3.8575	1.1348
7	3.3993	1.0617
6	3.2019	**1.0872**
3	2.945	**9.2503e+010**
4	3.1837e-011	0
g_{max1}	**9.2503e+010**	
g_{max2}	**1.0872**	
g_{max1}/g_{max2}	1.3162e-011	< *C* = 0.5

Table 10 shows the sensitivity of each input features for RBPNN training. Gap each feature also calculated, the biggest gap is g_{max2} = 279339620.85, while the second largest gap is g_{max2} = 2.846040724. Criteria for reduction of input feature is calculated based on the criteria presented. Value 1.017276x10^{-10} obtained. The experiment shows 1.017276 x 10^{-10} <C. u_{cut} position is determined by the position of the biggest gap, ie u_{cut} = 6. Ucut using these criteria, all input features that have sensitivity under u_{cut}, can be reduced without reducing the ability of the network performance. In Table 10 can be seen that features 4 has a lower sensitivity when compared to the features of the input position.

ISBN: 978-979-792-552-9

Tabel 10: Feature Sensitivity and Gap fusing RBPNN for E-coli Dataset.

Fitur	**Sensitivitas**	**Gap**
2	3.8052	1.175460274
1	3.2372	1.037431099
7	3.1204	1.155062003
5	2.7015	1.073770818
6	2.5159	**2.846040724**
3	0.884	**27977339620.85**
4	3.16E-11	0
g_{max1}	27977339620.85	
g_{max2}	2.846040724	
g_{max1}/g_{max2}	1.01727E-10	$< C = 0.5$

After the input feature 4 (*chg*) pruned, network retrained using RBFNN and RBPNN algorithm. Percentage of ability can be seen in Table 11 for RBFNN and Table 12 for RBPNN.

Tabel 11: Classification Error Percentage and Classification Percentagusing RBFNN Algorithm after Input Feature Pruned

Data	**Error Percentage**	**Classification Persentage**
Training	15,57%	84,43%
Testing	20,83%	79,17%

Table 11 shows the percentage of classification error for the training data after input feature chg reduced still is 15.57% with the percentage of classification capabilities as much as 84.43%. While the percentage of classification error for the test data is 20.83%, and the performance of classification is 79.17%.

Tabel 12: Classification Error Percentage and Classification Percentagusing RBPNN Algorithm after Input Feature Pruned

Data	**Error Percentage**	**Classification Persentage**
Training	15,57%	84,43%
Testing	19,79%	80,21%

Table 12 shows the percentage of classification error for the training data after input feature chg reduced still is 15.57% with the percentage of classification capabilities as much as 84.43%. While the percentage of classification error for the test data is 19.79%, and the performance of classification is 80.21%.

ISBN: 978-979-792-552-9

Conclusion

Efficiency and effectiveness an artificial neural networks can be improved by applying the sensitivity of each input feature. The ability of RBFNN network classification better than RBPNN, but RBFNN learning speed is slower than the RBPNN. The results also showed that sensitivity analysis can be used to improve the efficiency of neural network structure and classification capabilities remain effective. RBFNN model's accuracy is better than the RBPNN. Based on simulation results using a computer program for data classification capabilities RBFNN Iris can reach 97.14%, while RBPNN reach 97.14% .

Sensitivity analysis of RBFNN and RBPNN applied to see the sensitivity of each input to output feature RBFNN and RBPNN. If the criteria for the reduction fullfilled, the input feature can be pruned. For Iris dataset are not allowed to perform the reduction of the input feature. Whereas, for E-coli dataset results showed that after the reduction of the input feature classification ability of RBFNN and RBPNN is not change.

Artificial neural networks are basically can be applied into various things. For further development, the model RBFNN and RBPNN still allowed to be studied and developed, especially regarding the method of determining the effective center and the determination of the initial values and other important parameters. In addition, the study of theory, RBFNN and RBPNN can be applied to pattern recognition, kalasifikasi for practical purposes.

Acknowledgements. Thanks to the Directorate of Islamic Higher Education Ministry of Religious Affairs has provided funding for this research.

References

[1] Castillo, E., Guijarro-Berdinas, B., Fontenla-Romero, O., and Alonso-Betanzos, A., (2006), A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis, *Journal of Machine Learning Research*, **7** (2006), 1159–1182.

[2] Du., J.-X., Huang, D.S., Zhang, G.-J., Wang, Z.-F., A Novel Full Structure Optimization for Radial Basis Probabilistic Neural Networks", *Neurocomputing*, 70 (2006), Issue 1-3, 592-596. http://dx.doi.org/10.1016/ j.neucom.2006.05.003

[3] Du, K.-L. and Swamy, M.N.S., *Neural Networks in a Softcomputing Framework*, Springer-Verlag, London, Inggris. 2006.

[4] Guo L. And Huang D.S., Human Recognition Based on Radia; Basis Probabilistic Neural Network, *Proceedings of The International Joint Conference on Neural Networks*, 3, (2003), 2208-2211, http://dx.doi.org/ 10.1109/IJCNN.2003.1223751

[5] Hasanuddin and Irawan, M.I., (2009), The study of Sensitivity of Radial Basis Probabilistic Neural Network, *The Fifth International Conference on Mathematics, Statistics and their Application*, 344-349

[6] Huang, D.S. and Zhao, W.B., Determining the Center of Radial Basis Probabilistic

Neural Networks by Recursive Orthogonal Least Square Algorithm, *Applied Mathematics and Computing*, 162 (2005), 461-473. http://dx.doi.org/10.1016/ j.amc.2003.12.105

[7] Karayiannis, N.B., Reformulated Radial Basis Neural Networks by Gradient Descent, *IEEE Transaction on Neural Network*, 10 (1999), No. 3, 657-671. http://dx.doi.org/ 10.1109/72.761725

[8] Shi, D., Yeung, D.S. and Gao, J., Sensitivity Analysis Applied to the Construction of Radial Basis Function Networks, *Journal of Neural Networks*, 18(2005), 951–957. http://dx.doi.org/10.1016/j.neunet.2005.02.006

[9] Wang, X.-Z., Li, C.-G., Yeung, D.S., Song, S.J. and Feng, H.M., A Definition of Partial Derivative of Random Functions and Its Application to RBFNN Sensitivity Analysis, *Journal of Neurocomputing*, 71 (2008), 1515–1526. http://dx.doi.org/10.1016/ j.neucom.2007.05.005

[10] Zhao, W.B., Huang, D.S., and Guo L., (2003), Comparative Study between Radial Basis Probabilistic Neural Networks and Radial Basis Function Neural Networks, *LCNCS* 2690, Springer-Verlag, Berlin, pp 389-296.

[11] Zhao, W.B., and Huang, D.S., (2004), Radial Basis Probabilistic Neural Networks of Genetic Optimization of Full Structure, *Journal Infrared Millim Waves*, Vol. 23, No. 2, pp 113-118.

[12] Zurada, J..M., Malinowski, A. and Usui, S., Pertubation for Deleting Redunandt Inputs of Perceptron Networks, *Neurocomputing*, 14 (1997), 177-193. http://dx.doi.org/10.1016/ S0925-2312(96)00031-8