# Robust Combining Methods in Committee Neural Networks

S.A.Jafari, S.Mashohor

Department of Computer and Communication Systems Engineering, Faculty of Engineering, University Putra Malaysia

***Abstract*- Combining a set of suitable experts can improve the generalization performance of the group when compared to single experts alone. The classical problem in this area is to answer the question about how to combine the ensemble members or the individuals. Different methods for combining the outputs of the experts in a committee machine (ensemble) are reported in the literature. The popular method to determine the error in every prediction is Mean Square Error (MSE), which is heavily influenced by outliers that can be found in many real data such as geosciences data. In this paper we introduce Robust Committee Neural Networks (RCNNs). Our proposed approach is the Huber and Bisquare function to determine the error between measured and predicted value which is less influenced by outliers. Therefore, we have used a Genetic Algorithm (GA) method to combine the individuals with the Huber and Bisquare as the fitness functions. The results show that the Root Mean Square Error (RMSE) and R-square values for these two functions are improved compared to the MSE as the fitness function and the proposed combiner outperformed other five existing training algorithms.**

***Keywords*—Committee Machine; Neural Network; genetic Algorithms; Bisquare and Huber Function**

## I. INTRODUCTION

An ensemble is a set of learning machines that their outputs are combined to improve the performance of the whole system. The ensemble has a successful performance when its members are in disagreement. In such an intelligent system, each member estimates a target variable. These estimated outputs then combined with different methods which will be discussed later in the next section. Both theoretical [1] and empirical studies [2, 3] have shown that the ensemble output can be enhanced when individuals are accurate enough and the error of each member is occurred on a different part of the input space. In the past two decades a number of researchers have shown that a simply combining the output of many individuals can improve their prediction than single network alone [4, 5]. In particular, combining separately trained neural networks has been demonstrated to be successful [6, 7]. Therefore the main reason to use ensembles is due to improvement in the generalization ability. Different committee members can be created by different types of input training data, initialization, topology or different training algorithms and so on. There are many methods to create individuals for an ensemble in literature such as, Bootstrap aggregation or Bagging [4], Cross-validation [1], Stacking [5], Boosting by filtering [8], AdaBoost [9], and etc. After selecting individuals and training them, their generated results will be combined by some methods. Many investigations have been done to find accurate and useful combining methods to combine the network outputs and produce the final outputs such as Simple averaging [10], Majority voting [11], Naive Bayesian fusion [12], Ranking [13], Weighted averaging [14], Fuzzy integral [15], Decision templates [16], Dumpster Shafer combination [17], weighted majority voting [18], and etc. Some of them are suitable for classification and some of them are more applicable for regression. In this paper, we provide a review on robustness issues in ensemble neural network. A general ensemble structure can be viewed in Figure 1.

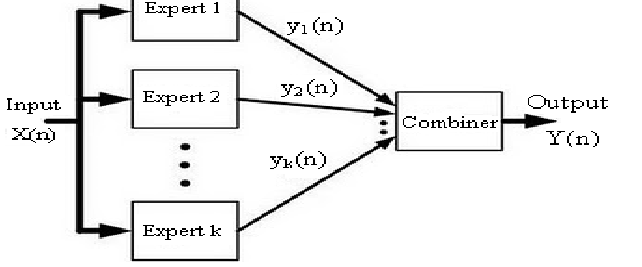


Figure 1: A graphical illustration of ensemble k experts.

## II. COMBINATION METHODS

As mentioned in introduction part, an ensemble approach can implemented using two methods: generating individuals or committee members and combining the individual predictions. In this section we explain a brief description on second parts and introduce some of the traditional methods.

### A. Simple Averaging

This combining method is most commonly used in ensemble. The final output in this method is obtained by averaging the whole individual outputs. It is easy to illustrate by Cauchy's inequality which the MSE for Committee Neural Network (CNN) with the simple averaging method is less than or equal to the average of MSE for whole networks. The disadvantage of this method is due to giving equal weights to all the individuals so the important individuals cannot be emphasized.

### B. Weighted Averaging

In this method every committee member has a suitable weight related to their ability to generalization. In [14] the researcher introduced a gating method to determine the weight of every experts. The authors in [19] have used GA to determine the weight of each member. To obtain the optimal weights for combining with GA algorithm, the fitness function is defined as below:

$$MSE_{GA}=\sum_{i=1}^{n}\frac{1}{n}(w_1y_{1i}+w_2y_{2i}+\ldots+w_ky_{ki}-T_i)^2\ ;\qquad \sum_{i=1}^{k}w_i=1 \tag{1}$$

where, $y_{1i}$ is the output of first network on the *ith* input or *ith* training pattern, $T_i$ is the target value of *ith* input, and n is the number of training data. The major disadvantage of this method is that it depends on the distribution of environmental variables; highly roughly distributions can scramble the order of weighted averaging of different species[20].

### *C. Ranking*

This method uses experimental results obtained from applying the experts on a set of datasets to generate rankings for those experts. All experts are ranked by an input dataset in each experiment. This act is repeated for all datasets of a set. The rankings obtained are then used in suitable methods such as average rank, success rate ration, significant wins, etc to evaluate the experts and to generate their final rankings [21]. The number of classes that has the same ranks depends on the number of classifiers used. Therefore, this method is useful only if the number of classifiers is small relative to the number of classes. Therefore the drawback in this method is that the combined ranking may have many ties. Otherwise, most of the classes are involved in ties and the final ranking is not interesting[22].

### *D. Majority Voting*

This combination method is most popular for classification problems. If more than half of the individuals vote a prediction, majority voting selects this prediction to be the final output. Majority voting also has some disadvantages, for example, a winning expert that obtains only a minority of correct results is often neglected by the majority voting decision rules and this downgrades the diversity of the ensembles which is the basic reason for using ensembles.

However, there is no unique criterion for selecting a combination method and it is mainly dependent on the specific application under the focus. The nature of application in addition to the size and quality of datasets and the generated errors on the region of the input space are the most significant parameters involved in such selection [23].

## III. A REVIEW ON ROBUST STATISTICS

The focus of classical statistics is mainly on optimal procedures where the assumptions about the models are true. For example, the mean value is the most efficient estimator for the expected value only when a sample comes from a normal distribution. Also the accuracy of the Least Squares (LS) method for linear regression is totally dependent on such assumptions in a same fashion. As long as the model assumption (in Equation 2), where the $e_i$s are independent and identically distributed (i.i.d), [24].

$$y_i = b_0 + b_1 x_{i1} + ... + b_m x_{im} + e_i = X_i^{'} b + e_i \qquad (2)$$

However, it is obvious that even single outliers are capable of changing the mean value or the predicted coefficient of the regression function. Also the classical statistics presumes that the data is made up of independent samples from an optimal distribution, where in contrast the robust statistics considers the data to be mixed with an unknown noise distribution [24]. The extended discussions about these subjects and the related methods are provided in [25].

## IV. M-ESTIMATORS FOR GA METHODS

Let us consider the case of GA method to obtain the weight of individuals in neural network ensemble. The most common, fitness function in GA method is based on the MSE which is optimal for the normal error distribution. In the MSE, the square of the residuals is minimized to achieve the best fitting between ensemble output and the targets. However, when the errors are generated even from different Gaussian distributions with different means and variances, the MSE method may lose its efficiency. The robust method is decreasing the influence of this data by using another error function which is less influence by them. Given training data $D = \{(X, T)\}$ where $X = (X_1, ..., X_n)$ is input data and $T = (T_1, ..., T_n)$ is the target data. The fitness function is minimized as follows:

$$\sum_{i=1}^{n} \rho(e_i) = \sum_{i=1}^{n} \rho(Y_i - T_i); \quad Y_i = \sum_{j=1}^{k} w_j y_{ij} \quad and \quad \sum_{j=1}^{k} w_j = 1 \qquad (3)$$

Where $\rho$ is a suitable error measure, $w_j$ is the weight of each member, and $y_{ij}$ is the output of the jth individual from ith input and $e_i$ is the ith error. A suitable error function must satisfy the conditions in (4).

$$\rho(e_i) \geq 0, \qquad \rho(e) = \rho(-e), \qquad \rho(e_i) \geq \rho(e_j) \ if \ \ i \geq j \qquad (4)$$

A parameter estimation based on an objective (fitness) function of equation (3) and (4) is called an M-estimator. Figure 2 shows a graph of the error function and the weighting function for the Least Square (LS) method. LS means that the overall solution minimizes the sum of the squares of the errors made in solving every single equation but MSE means the average of this errors. Figure 3 and 4 provides the error function ρ and the weight function w for the Huber and Bisquare methods respectively.

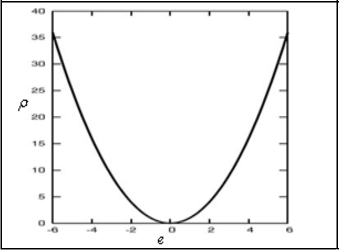
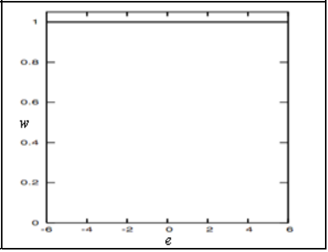
| $\rho(e)$ | $w(e)$ |
|---|---|
| $e^2$ | 1 |
|  |  |

Figure 2: The error function $\rho$ and the weight function $w$ for the LS method.

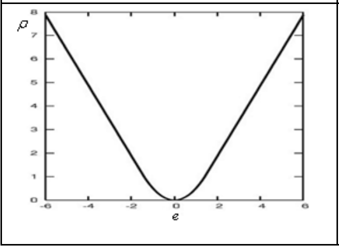
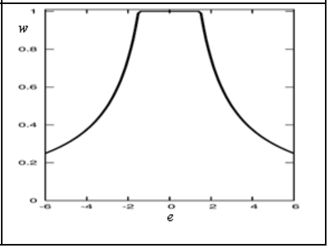
| $\rho(e)$ | $w(e)$ |
|---|---|
| $\begin{cases} \dfrac{e^2}{2} & if \lvert e \rvert \leq k \\ k\lvert e \rvert - \dfrac{k^2}{2} & if \lvert e \rvert > k \end{cases}$ | $\begin{cases} 1 & if \lvert e \rvert \leq k \\ k / \lvert e \rvert & if \lvert e \rvert > k \end{cases}$ |
|  |  |

Figure 3: The error function $\rho$ and the weight function $w$ for Huber.

In LS method, the function ρ increases in the quadratic manner with increasing distance, but the weights are always constant. In the Huber and Bisquare methods, the error function ρ is divided in two parts. In first part (small errors; means the absolute of errors are less than the constant k), the ρ is quadratic, and for second part (large errors; means the absolute of errors are greater than the constant k), the ρ is linearly increasing. Furthermore as shown in figure 2, 3 and 4, the Bisquare and Huber methods are stronger than MSE function, because for larger errors the weight w not only is not constant, but also decreases for extreme outliers.

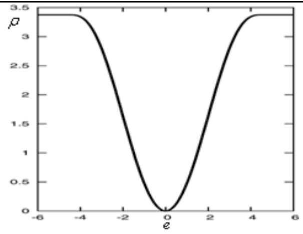
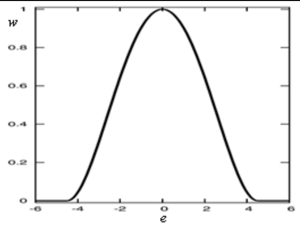
| *ρ(e)* | *w(e)* |
|---|---|
| $\begin{cases} \frac{k^2}{6}\left(1-\left(1-\left(\frac{e}{k}\right)^2\right)^3\right); & if \lvert e \rvert \le k \\ \frac{k^2}{6} ; & if \lvert e \rvert > k \end{cases}$ | $\begin{cases} \left(1-\left(\frac{e}{k}\right)^2\right)^2 ; & if \lvert e \rvert \le k \\ 0 ; & if \lvert e \rvert > k \end{cases}$ |
|  |  |

Figure 4: The error function *ρ* and the weight function *w* for Bisquare.

## V. SIMULATION RESULTS

The simulation is performed on Matlab on a personal computer with Intel core 2 duo 2.2 GHZ processor and 2 GB of memory. In this section, the performances of five neural networks with different training algorithms are evaluated. These five training algorithms were:

1) Levenberg–Marquardt (LM), [26] and [27],

2) Scaled Conjugate Gradient (SCG), [28],

3) Bayesian regularization (BR), [29],

4) Resilient Back-Propagation (RP), [30],

5) Gradient Descent with momentum and adaptive learning rate backpropagation (GDX), [31].

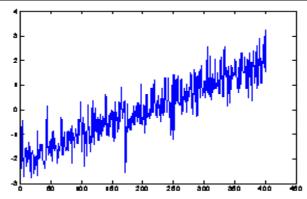
| GA Parameters | Configuration |
|---|---|
| Population | Population type: double vector, Population size: 50, Initial range: [0; 1] |
| Scaling function | Rank |
| Selection function | Stochastic uniform |
| Reproduction | Elite count: 2, crossover fraction: 0.8 |
| Mutation function | Use constraint dependent default |
| Crossover function | Scattered |
| Migration | Direction: forward, Fraction: 0.2, Interval: 20 |
| Stopping criteria | Generations: 1000, Time limit: Inf, Fitness limit: Inf, Stall generations: 1000, stall time: Inf |

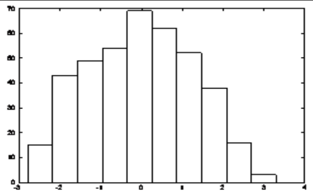
Table 1: The configurations of GA Parameters for calculate the weight of individuals.

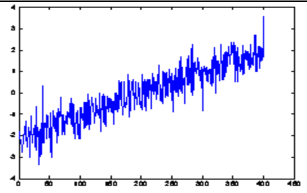
After that three methods for combining the outputs of individuals are implemented. All of these three methods are based on GA, with different fitness functions; MSE, Huber, and Bisquare. GA Parameters of the Matlab which used in this work are listed in Table 1. The data points used for this experiment consists of two independent variables and one dependent variable such as data sets that have been described in [32]. The input data are generated in the range of [-2, 2], and the equation (5) is used to calculate the dependent variable. After that the data points are corrupted in both x and y-coordinates with Gaussian noise with dB=20 using Matlab software. Pictures of the corrupted X1, X2 and Y are illustrated in figure 5 with the corresponding histograms.

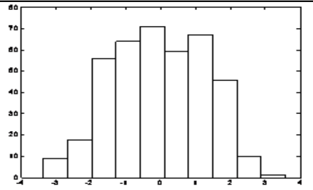
$$y = x_1 e^{-m}, \quad m = x_1^{\,2} + x_2^{\,2}; \quad x_1, x_2 \in [-2,2] \tag{5}$$

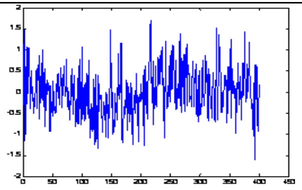
To assess the performance for each estimator and compare them, the RMSE and R-square has been used.


a) $X_1$-values with 20dB Gaussian noise vs time.


b) A histogram of $X_1$.


c) $X_2$-values with 20dB Gaussian noise vs time.


d) A Histogram of $X_2$.


e) Y-values with 20dB Gaussian noise vs time.

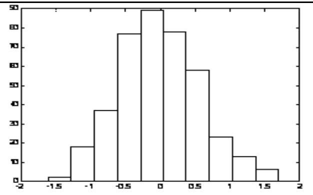

f) A Histogram of Y.

Figure 5: The plot of $X_1$, $X_2$ and Y values after noise addition and their corresponding histograms.

## VI. RESULTS AND DISCUSSIONS

In this paper we presented two robust fitness (objective) functions for neural network ensemble with GA as combining method. It is based on the robust estimator and introduces two robust approaches to the error criterion to reduce the influence of the outliers. The performance of the weighted averaging with GA constructed from combining five different training algorithms are shown in figure 6. According to result, the GA with MSE as fitness function using LM, SCG, BR, RP, and GDX methods has produced the minimum error whereas, combination of these algorithms with Huber and Bisquare are associated with the minimum error. The GA-MSE derived values for $w_1$, $w_2$, $w_3$, $w_4$ and $w_5$ corresponding to LM, SCG, BR , RP, and GDX estimations are 0.091, 0.161, 0.01, 0.717 and 0.022 respectively, also are 0.025, 0.746, 0.098, 0.083, 0.047 for Huber and 0.083, 0.13, 0.144, 0.643, 0.001 for Bisquare fitness functions. Over all estimation of outputs by RCNN was calculated as below:

$$Output_{RCNN} = w_1 y_1 + w_2 y_2 + w_3 y_3 + w_4 y_4 + w_5 y_5 \tag{6}$$

Figure 6 shows the comparison of RMSE and R-square testing data points from different algorithms including LM, SCG, BR, RP, GDX and figure 7 shows the comparison of RMSE and R-square data points from GA as combination method with different fitness function including GA-MSE, GA-Huber, and GA-Bisquare (all learning algorithms).

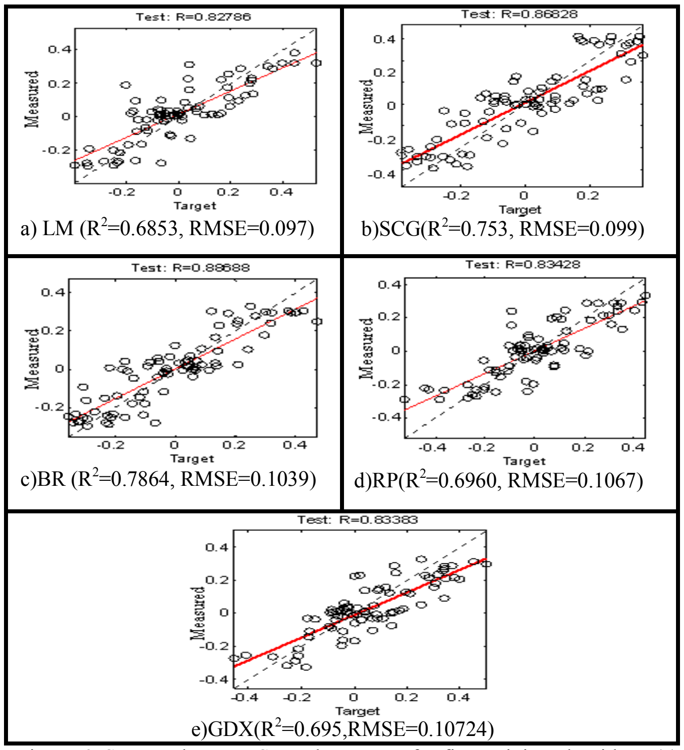
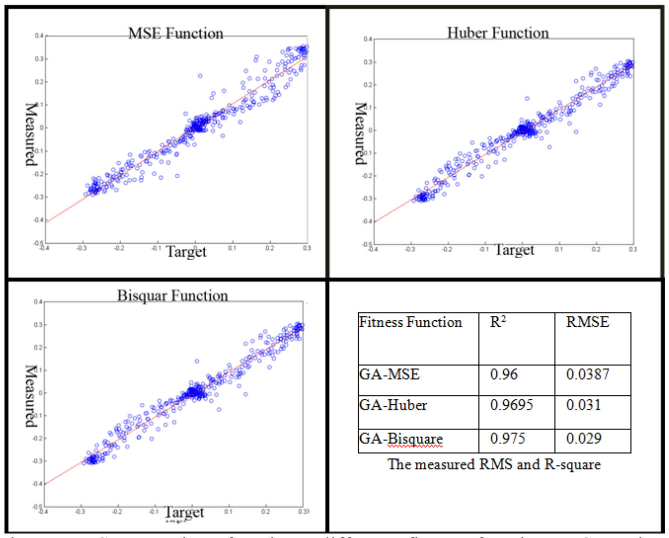


Figure 6: Scatter plots, RMSE and R-square for five training algorithms (a) LM, (b) SCG, (c) BR, (d) RP, (e) GDX.

Considering cross plots of Fig 6 (a–e), among the five neural networks with different training algorithms used, the smallest error and highest R-square are 0.097 and 0.7864 respectively. Based on Figure 6 and 7, the weighted averaging with applying GA for construction committee neural network using LM, SCG, BR, RP and GDX methods has produced the minimum error and maximum R-square algorithms than each training algorithm individually.



| Fitness Function | $R^2$ | RMSE |
|---|---|---|
| GA-MSE | 0.96 | 0.0387 |
| GA-Huber | 0.9695 | 0.031 |
| GA-Bisquare | 0.975 | 0.029 |

The measured RMS and R-square

Figure 7: Scatter plots for three different fitness functions LS, Huber, Bisquare, and the measured RMS and R-square using genetic algorithm.

Also according to Figure 7, the genetic algorithm with Bisquare function has provided the smallest error (RMSE=0.0295) and R-square value of 0.97 then genetic algorithm with Huber and MSE function. This indicates that Bisquare function had a significant improvement as objective function for GA. Namely; Bisquare function performs better than any of the individual training algorithms acting alone for prediction. Also it has provided better results than constructed CNN with GA as combining method by Huber and MSE function. It might be noticed that in our case study the GA with Bisquare as objective function performed better than Huber or MSE function; in some cases it may not be so.

## VII. CONCLUSION

The results for GA based combiner have shown an improved R-square and RMSE values in Huber and Bisquare function compared with MSE function. Robustness issues have been ignored in ensemble methods and have not been investigated in this type of machine learning approach, especially application with real data which generally are not clear-cut and most of times are associated with uncertainties such as geosciences data. These robust methods can be easily adapted to many types of ensemble neural networks and committee machine with different type of members and applications. This indicates that robust fitness functions have a significant improvement than any of the individual training algorithms acting alone and also GA with conventional fitness function (MSE function).

## REFERENCES:

[1] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning " *Advances in Neural Information Processing Systems,* vol. 7, pp. 231-238, 1995.

[2] S. Hashem, *et al.*, "Optimal linear combinations of neural networks: an overview," in *IEEE World Congress on Computational Intelligence., 1507-1512 vol.3*, 1994.

[3] D. Opitz and R. Maclin, "Popular Ensemble Methods: An Empirical Study," *Journal of Artificial Intelligence Research,* vol. 11, pp. 169 - 198, 1999.

[4] L. Breiman, "Bagging predictors," *Machine Learning,* vol. 24, pp. 123-140, 1996.

[5] D. H. Wolpert, "Stacked Generalization," *Neural Networks,* vol. 5, pp. 241-259, 1992.

[6] H. Drucker*, et al.*, "Boosting and Other Ensemble Methods," *Neural Computation,* vol. 6, pp. 1289-1301, 1994.

[7] Maclin and Opitz, "An empirical Evaluation of bagging and Boosting," in *Proceedings of the Fourteenth National Conference on Articial Intelligence, 456-551*, 1997, pp. 456-551.

[8] R. E. Schapire, "The Strength of Weak Learnability," *Mach. Learn.,* vol. 5, pp. 197-227, 1990.

[9] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *European Conference on Computational Learning Theory*, 1995, pp. 23-37.

[10] W. Lincoln and J. Skrzypek, "Synergy of clustering multiple back propagation networks," *Advances in Neural Information Processing systems-2,* pp. 650-657, 1990.

[11] L. K. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 12, pp. 993-1001, 1990.

[12] L. Xu, *et al.*, "Methods of combining multiple classifiers and their applications to handwriting recognition," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 22, pp. 418-435, 1992.

[13] H. Tin Kam, *et al.*, "Decision combination in multiple classifier systems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 16, pp. 66-75, 1994.

[14] R. A. Jacobs, "Methods For Combining Experts' Probability Assessments," *Neural Computation,* vol. 7, pp. 867-888, 1995.

[15] S.-B. Cho and J. H. Kim, "An HMM/MLP architecture for sequence recognition," *Neural Comput.,* vol. 7, pp. 358-369, 1995.

[16] L. I. Kuncheva, *et al.*, "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern Recognition,* vol. 34, pp. 299-314, 2001.

[17] M. R. Ahmadzadeh and M. Petrou, "Use of Dempster-Shafer theory to combine classifiers which use different class boundaries," *Pattern Analysis & Applications,* vol. 6, pp. 41-46, 2003.

[18] L. I. Kuncheva, "Classifier Ensembles for Changing Environments," ed, 2004, pp. 1-15.

[19] D. W. Opitz and J. W. Shavlik, "Actively Searching for an Effective Neural Network Ensemble," *Connection Science,* vol. 8, pp. 337 - 354, 1996.

[20] J. C. J. R. H. G. Jongman, C. W. N. Looman, C. J. F. ter Braak, O. F. R. van Tongeren, P. A. Burrough, j. a. f. oudhof, a. barendregt, t. j. van de Nes, "Data Analysis in Community and Landscape Ecology," p. 324, 1997.

[21] P. Brazdil and C. Soares, "A Comparison of Ranking Methods for Classification Algorithm Selection," ed, 2000, pp. 63-75.

[22] T. K. Ho, *et al.*, "Decision Combination in Multiple Classifier Systems," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 16, pp. 66-75, 1994.

[23] S. Yang and A. Browne, "Neural network ensembles: combining multiple models for enhanced performance using a multistage approach," *Expert Systems,* vol. 21, pp. 279-288, 2004.

[24] F. Klawonn and F. Höppner, "Fuzzy Cluster Analysis from the Viewpoint of Robust Statistics," in *Views on Fuzzy Sets and Systems from Different Perspectives*, ed, 2009, pp. 439-455.

[25] P. J. Huber, *Robust Statistics* Wiley Series in Probability and Statistics, 2004.

[26] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quart. Appl. Math., 2 (1944), pp. 164-168.,* 1994.

[27] D. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *SIAM Journal on Applied Mathematics* vol. 11, pp. 431–441, 1963.

[28] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks,* vol. 6, pp. 525-533, 1993.

[29] D. J. C. MacKay, "Bayesian interpolation," *Neural Comput.,* vol. 4, pp. 415-447, 1992.

[30] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *IEEE International Conference on Neural Networks*, 1993, p. 586-591, pp. 586-591 vol.1.

[31] L. Baird and A. Moore, "Gradient descent for general reinforcement learning," presented at the Proceedings of the 1998 conference on Advances in neural information processing systems II, 1999.

[32] M. El-Melegy, *et al.*, "Robust Training of Artificial Feedforward Neural Networks," in *Foundations of Computational Intelligence (1)*, ed, 2009, pp. 217-242.