Kishor Kumar Katha* and Suresh Pabboju

AGCS Technique to Improve the Performance of Neural Networks

https://doi.org/10.1515/jisys-2017-0423
Received August 18, 2017; previously published online March 4, 2019.

Abstract: In this paper, a fresh method is offered regarding training of particular neural networks. This technique is a combination of the adaptive genetic (AG) and cuckoo search (CS) algorithms, called the AGCS method. The intention of training a particular artificial neural network (ANN) is to obtain the finest weight load. With this protocol, a particular weight is taken into account as feedback, which is optimized by means of the hybrid AGCS protocol. In the beginning, a collection of weights is initialized and the similar miscalculation is discovered. Finally, during training of an ANN, we can easily overcome the training complications involving ANNs and also gain the finest overall performance with training of the ANN. We have implemented the proposed system in MATLAB, and the overall accuracy is about 93%, which is much better than that of the genetic algorithm (86%) and CS (88%) algorithm.

Keywords: Adaptive genetic (AG) algorithm, cuckoo search (CS) algorithm, genetic algorithm (GA), back propagation algorithm (BPA), artificial neural network (ANN), Levy flight.

1 Introduction

Artificial neural network (ANN) is a new field connected with computational science, which integrates various strategies for difficulty resolution that cannot be consequently effortlessly achieved without the help of an algorithmic conventional concentration. The particular ANNs stand for big as well as diverse classes connected with computational types [8]. Together with biological template modules, these networks are created basically by more or less comprehensive examples. The particular general approximates as well as computational types having specific traits, such as the ability to learn or even adapt, to prepare or to generalize data, are recognized as the particular ANNs. Contemporary development of a (near) optimal network architecture is carried out by people skilled with the use of a monotonous learning from trial-and-error processes. Neural networks are generally algorithms for optimization as well as searching, and are freely dependent on principles prompted merely by research on the characteristics of the brain. For optimization as well as searching using neural networks in tandem with genetic algorithms (GAs), there are generally two techniques, each with its own advantages coupled with weaknesses. By means of different paths, the two processes are commonly evolved [13, 14].

Optimization algorithms, in addition to being called learning algorithms, are, according to a number of features of scientific advancement, usually known as GAs. An easy method involving encoding answers to the issue in chromosomes; an assessment purpose, which, on return, starts to attain chromosome directed at the idea; involving initializing a population of chromosomes; and workers that could be put on parents whenever they reproduce to correct the genetic composition are the five expected criteria. Integration could be by mutation or crossover, in addition to site-specific workers. Parameter configurations for the particular criterion are the workers or can be anything else [12]. The criteria can develop populations involving much better ones in addition to the individuals, converging eventually in effect to an international ideal, every time

***Corresponding author: Kishor Kumar Katha,** Department of Computer Science and Engineering, Osmania University, Hyderabad 500 007, India, e-mail: kishorkumar0376@gmail.com
Suresh Pabboju: Department of Information Technology, Chaitanya Bharathi Institute of Technology, Hyderabad 500 075, India

a GA actually run has a portrayal, which encodes a solution to a problem in addition to workers that can generate much better young ones through good parents [7]. In numerous situations, with regard to performing the particular optimization on the standard workers, mutation in addition to crossover is usually adequate [2]. GAs can assist similar to a black box function optimizer, without requiring any kind of know-how about computers of the particular domain in these cases. However, understanding of the particular domain is frequently exploited to raise the performance of particular GAs with the incorporation involving new workers highlighted in this particular paper [17].

In neural networks, functionality enhancement partition space can be a space that is used to classify data samples right after the test is actually mapped by a neural network [19]. Centroid can be a space with partition and also denotes the particular middle of the class. In traditional neural network functionality, the location of centroids and the partnership involving centroids and also classes are generally fixed personally [21]. Furthermore, with reference to the quantity of classes, a variety of centroids are actually set. To locate the optimum neural network, this particular set centroid restriction minimizes the risk [21]. GAs have emerged to be practical software as heuristic alternatives for sophisticated discrete optimization issues. Particularly, there has been large fascination with their use in the most effective arrangement and timetabling issues [18]. However, nowadays, there have been several attempts to become listed on both systems. Neural networks could be rewritten, as a type of GA is termed as some sort of classifier program and also vice versa [11]. GA is meant for training feed-forward networks. It does not merely work with its own task but also performs rear distribution, the normal training criteria [10]. This accomplishment comes from tailoring the particular GA to the domain of training neural networks.

Cuckoo search (CS) is a recent meta-heuristic optimization algorithm launched by Yang and Deb in 2009 [20]. This is a certain kind of swarm intelligence that is based on brooding actions of several kinds of cuckoo birds. CS is utilized for dealing with intricate marketing difficulties, and according to the maximum remedies received by simply C, it can be more effective than the finest remedies achieved with simply different swarm intelligent algorithms.

Although the ANN feature has several advantages, it also has several complications similar to convergence in addition to receive neighborhood minima during training by means of back propagation [9]. That is why analysts are engaged to produce a new and the finest protocol to train the ANN. On this routine optimization protocol, algorithms similar to particle swarm optimization (PSO), CS, and others have been utilized to enhance the overall performance; however, they certainly do not fulfill the overall performance prerequisites. On this impression, our system is also thought to be able to overcome a real issue during training involving ANN. Thus, we proudly propose the novel AGCS technique for training the ANN.

2 Related Work

Karegowda et al. [5] have recommended a method to enhance the overall performance associated with neural networks. To involve various elements like providing the optimal quantity of concealed covering, offering actual relevant feedback factors, and also offering maximum connection loads, different model varieties of neural networks have been proposed. To initialize and also optimize the connection loads associated with the back propagation network (BPN), the authors finally recommended the application of a hybrid type of built-in GA and also BPN wherever GAs were employed. Moreover, this process also had necessary capabilities identified by a couple approach. Thus, to help spot diabetes mellitus, decision tree and also GA-correlation based feature selection processes had been used with feedback toward the hybrid type. The effects showed that a GA-optimized BPN strategy had outperformed the actual BPN tactic without having GA optimization. Moreover, the hybrid GA-BPN having relevant inputs additionally enhanced the classification detail in comparison with the effects made by GA-BPN by itself with a number of repetitive inputs.

Sagar et al. [16] systematically suggested a way to the neural network, which has been measured for adaptive technique that increasingly self-organizes to approximate the solution, creating the situation to

solve of the need as well as decidedly specify the actual methods headed for the solution. Moreover, with unnatural neural network, evolutionary computation may very well be included in order to increase the actual performance by various amounts, and consequently this kind of neural network is named as evolutionary ANN. Moreover, they have suggested one particular difficulty associated with neural network, particularly realignment associated with connection loads for understanding display by means of GA around a feed-forward architecture. The particular performance associated with developed solution evaluation is presented with respect to well-established means of gradient procedure. A standard problem associated with category, XOR, has been taken in order to rationalize the actual research. The offered procedure has not only been used for acquiring extreme possibility to obtain the actual international minima but also for very quick convergence.

Iqbal [4] remarkably offered a strategy to help confirm diverse attributes and importance with a specific learning problem. Even though essential, a number of attributes are usually much less appropriate. The protocol might be with all this information about characteristic value which depends on expert opinion or even earlier learning which is an alternative associated with choosing the most appropriate attributes to employ characteristic selection. In the event of pupils taking characteristic values into account, learning is usually quicker and more correct. Correlation-aided neural networks (CANNs) seem to present this kind of protocol. CANN presents characteristic values because of its effects coefficient relating to the concentration on credit and also the attributes. A CANN custom-made regular feed-forward neural network (FFNN) adjusts each correlation value and training data. Empirical evaluation has shown that CANN seems to be quicker and more specific compared to making use of the two-step strategy associated with characteristic selection after applying the regular learning algorithms.

Arotaritei [1] has deftly recommended a technique for fuzzy feed-forward (FFNR) and fuzzy recurrent networks (FRNN), which have turned out to be solutions for real-world problems. When it comes to fuzzy numbers, the learning algorithms have been based on gradient strategies designed for fuzzy logic together with heuristic principles. A new learning process based on GAs together with local crossover that is put on various topologies of fuzzy neural networks with fuzzy numbers is recommended in this paper. Having L-R fuzzy numbers as inputs, outputs, and loads along with fuzzy arithmetic for the future transmission propagation, their process had ended up being put on FFNR as well as FRNN. Their α-cuts and also fuzzy biases had been furthermore investigated. The effectiveness of their recommended process was proven using a couple of programs, as evidenced by the mapping of a new vector of triangular fuzzy numbers straight into another vector of triangular fuzzy numbers for FFNR plus the vibrant record of fuzzy sinusoidal oscillations for FRNN.

Credit goes to Okkan [15] for the unique and outstanding features offered for monthly runoff prediction by using wavelet change and FFNNs. For the modeling in the hybrid model, discrete wavelet transform (DWT) and Levenberg-Marquardt optimization criteria-based FFNN had been regarded as appropriate and therefore applied. In the Turkish Aegean coast, the study area covering the actual basins associated with the Medar River was found. DWT meteorological facts, which usually characterize the study area, were decomposed into wavelet sub-time sequences. By utilizing Mallow's coefficient based on almost all achievable regression methods to prevent co-linearity, unsuccessful sub-time sequences were eradicated. Consequently, powerful sub-time sequence elements constituted the new inputs associated with FFNN. Several preferred evaluation measures such as determination coefficient, adjusted determination coefficient, Nash-Sutcliffe efficiency coefficient, root mean squared error, and weighted mean absolute percentage error were used in order to determine modeling performances.

Kawam and Mansour [6] have amazingly recommended an innovative training algorithm for ANN by employing the CS approach. They put in place the CS criteria regarding training a feed-forward multilayer perceptron network. They have compared and contrasted the performance together with some other approaches such as PSO and guaranteed convergence PSO on four benchmark problems. By those evaluations, CS has shown its superiority. The particular CS criteria are applied below to the training involving ANN. To boost the performance, each of our recommended approach uses the adaptive genetic (AG) with the CS algorithm for the ANN training.

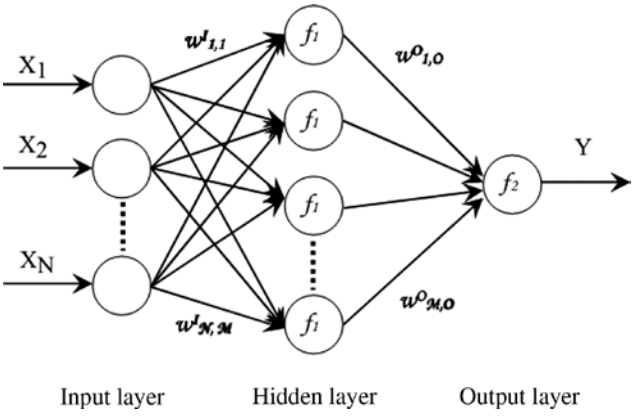

Figure 1: Example of ANN.

3 Proposed Technique for Training ANN

An ANN is a programmed computational model that aims to duplicate the neural structure and the functioning of the human brain. It is made up of an interconnected structure of artificially produced neurons that function as pathways for data transfer. ANNs are flexible and adaptive, learning and adjusting with each different internal or external stimulus. ANNs are used in sequence and pattern recognition systems, data processing, robotics, and modeling. The ANN consists of a single input layer and a single output layer in addition to one or more hidden layers. All nodes are composed of neurons except the input layer. The number of nodes in each layer varies depending on the problem. The complexity of the architecture of the network is dependent on the number of hidden layers and nodes. Training an ANN means to find a set of weights that would give desired values at the output when presented with different patterns at its input. Figure 1 shows an example of a simple ANN.

Training is a very important process in ANN. A traditional method for training the ANN is the back propagation algorithm (BPA). However, while using this method for training the ANN, we have to face some problems, and in order to overcome these problems, we have developed a new technique for training the ANN method called the AGCS technique, which is explained clearly in following sections.

3.1 AGCS Technique

In order to overcome the difficulties and also to improve the performance of neural networks, we have proposed the innovative AGCS technique for training neural networks. The main goal of the training algorithm is to obtain optimized weights. Hence, to obtain the best suitable weights, the hybrid of the AG and CS algorithms is used in our system. The flowchart for our AGCS technique is shown in Figure 2.

In the AGCS training method, the weights are considered as the input, i.e. weights are initialized. Then, the fitness value is evaluated for the initialized weights. After that, parent weights are selected as per the fitness vale. Then, the selected parent weights are processed by two optimization algorithms such as the AG and CS algorithms. The separate solutions obtained by both methods are merged to get the best solution or weight for the ANN.

3.1.1 Step-by-Step Procedure of the AGCS Technique

The aim of the training algorithm of ANN is to get the best suitable weights. In this sense, our technique is suggested for training the ANN. In order to get the best optimized weight, we have used the hybrid of two

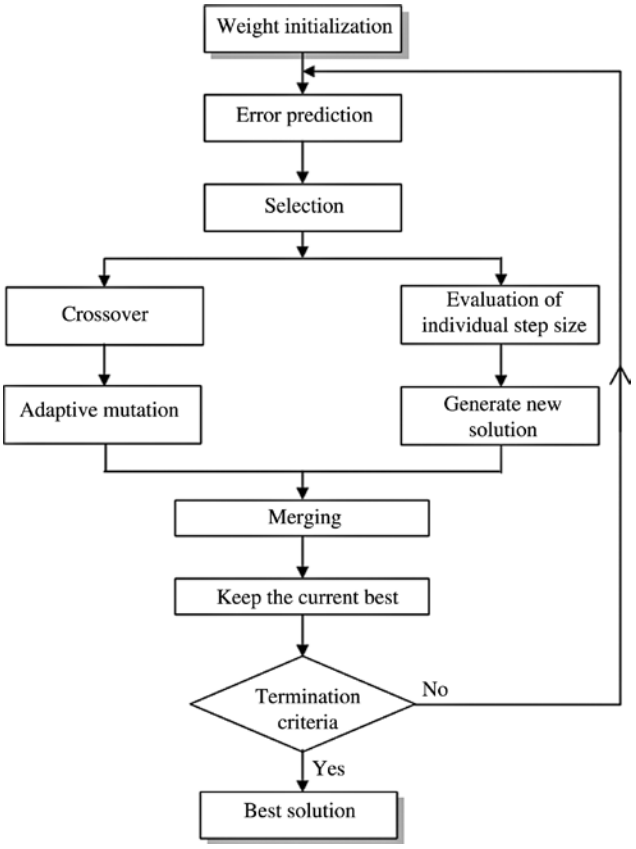

Figure 2: Block diagram of the AGCS technique.

optimization algorithms such as adaptive GA (AGA) and CS algorithm. The step-by-step procedure of our proposed AGCS technique for training the ANN is shown as follows.

Step 1: Initialization of Weights

At first, weights are initialized in a random manner and trained to get the optimal weights. In this initialization step, the hidden biased values are also assigned. From the second step, these initialized values are optimized to obtain the best weights.

Step 2: Error Prediction

In this step, the error is identified for the initialized weights. In our method, the error is evaluated by using the equation given below:

$$E = AV - H, \tag{1}$$

where E is the error and H is the activation function and AV is the actual value. The activation function is found by using the following equation:

$$H = \sum_{j=1}^{M} \frac{w_j^O}{1 + \exp(-\sum_{i=1}^{N} x_i w_{ij}^I)}, \tag{2}$$

where M is the total number of hidden neurons, N is the total number of inputs, w_j^O is the j^{th} weight assigned between the hidden and output layers, w_{ij}^I is the weight assigned between the input and hidden layers, and x_i is the i^{th} input value.

Step 3: Parent Weights Selection

The main aim of an optimization algorithm is to reduce the error. Hence, in this step, the parent weights are selected as per the error predicted in the previous step. The weights having high error are selected as parent for the optimization purpose. In the upcoming steps, the selected parents are optimized by two different algorithms, such as the AG and CS, to reduce the error.

Step 4: Crossover

The crossover is performed to generate new offspring based on the AG algorithm. The new offspring is generated by combining two parent chromosomes.

Step 5: Adaptive Mutation

Adaptive mutation is performed to obtain complete new offspring (weights). In general, by using a mutation rate, the mutation process is performed. In the traditional GA scheme, the mutation rate is selected at random. However, in our adaptive method, we have modified it and used an expression to find the mutation rate. The equation to find the mutation rate is given below:

$$m_r = \sqrt{\frac{c}{2\pi}} * \frac{\exp(-c/(2(R-\mu)))}{(R-\mu)^{3/2}}, \tag{3}$$

where m_r is the mutation rate, R is a random value [0, 1] that changes in every iteration, and c and μ are mutation parameters considered as 2 (i.e. $c=2$ and $\mu=2$).

After performing the mutation process, we will obtain the complete new offspring. The new offspring is nothing but the new optimized weight by the AGA; that is, a set of new optimized weights will be obtained after performing steps 4 and 5.

Step 6: Step Size Evaluation

In this step, the step sizes for individual parent weights obtained in step 3 are evaluated. The following expression is used to determine the step size:

$$S_z = \alpha S(w^t - w^t_{\text{best}}) \cdot r, \tag{4}$$

where S_z is the step size, α is a step size parameter ($\alpha=0.01$), w^t is the current parent weight, w^t_{best} is the best solution so far, r is a random number from a standard normal distribution [0, 1], and S is step. Step S is determined by using Mantegna's algorithm, shown in the equation below:

$$S = \frac{u}{|v|^{1/\beta}}, \tag{5}$$

where β is a parameter arising in the interval [8, 13], which we choose in our system as 1.5 (i.e. $\beta=3/2$), and u and v are normal distributions, which are estimated as follows:

$$u \sim N(0, \sigma_u^2),\ v \sim N(0, \sigma_v^2), \tag{6}$$

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\text{Sin}(\pi\beta/2)}{\Gamma[(1+\beta/2)]\beta \cdot 2^{(\beta-1)/2}} \right\}^{1/\beta},\ \sigma_v = 1. \tag{7}$$

Step 7: Generation of New Solution

In this step, the new optimized solutions or weights are generated for the corresponding parent weights based on the CS algorithm (Levy flight). The new weights are generated by using the step size values obtained in step 6. The new weights are generated by using the expression given below:

$$w^{(t+1)} = w^t + S_z, \tag{8}$$

where $w^{(t+1)}$ is the new weight, S_z is step size, and w^t is the current parent weight. From this step, another set of optimized weights are obtained.

Step 8: Merging
In this step, the solutions obtained in steps 5 and 7 are merged to get the best solutions or weights. The merged weights are considered as the optimized weights and also the resultant weights of the AGCS training algorithm.

Step 9: Termination Criteria
The maximum number of weights to train the ANN is considered as the termination criterion. If the process meets this criterion, it is terminated; else, step 2 is started for the next iteration.

4 Results and Discussion

The proposed AGCS technique for training the ANN is implemented in the working platform of MATLAB (version 7.1, MathWorks, Natick, MA, USA) with the following system configuration:

Processor: Intel core i5
CPU Speed: 3.20 GHz
OS: Windows 7
RAM: 4 GB

In order to analyze and prove the performance of our proposed AGCS technique, we have to train the ANN by some similar training methods like GA, AGA, and BP, which are at last compared to our proposed method. To train the ANN, selected features have been normalized. This normalization is necessary to prevent non-uniform learning. Hence, the weights associated with some features increase the speed of convergence. The ALL/AML (Acute Lymphoblastic Leukemia/Acute Myeloid Leukemia) data set for this experimental analysis is collected from Ref. [3]. After the normalization, the randomly chosen sample is divided into three categories such as training, cross-validation, and testing data sets. The training data set is used for learning the network. Cross-validation is used to measure the training performance during the training as well as to stop the training if necessary. Leukemia contains four types. In our system, we consider only two major types, namely ALL and AML. For this training purpose, we consider 38 patients who are divided into two clusters with 25 and 13 patients for ALL and AML, respectively, with 7129 genes overall.

Tables 1–4 show the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) of BPA, GA, AGA, and AGCS for different data sets.

The graphical representation of sensitivity, specificity, accuracy, positive prediction value (PPV), negative prediction value (NPV), false positive rate (FPR), Matthews's correlation coefficient (MCC), and false discovery rate (FDR) of the BPA, GA, AGA, and AGCS techniques for different numbers of data sets are shown and discussed below.

Table 1: TP, TN, FP, and FN for data set 1.

Training method	TP	TN	FP	FN
BPA	3	2	1	3
GA	2	3	2	3
AGA	2	5	2	0
AGCS	2	6	1	0

Table 2: TP, TN, FP, and FN for data set 2.

Training method	TP	TN	FP	FN
BPA	2	3	1	3
GA	3	0	1	5
AGA	1	6	1	1
AGCS	3	6	0	0

Table 3: TP, TN, FP, and FN for data set 3.

Training method	TP	TN	FP	FN
BPA	0	5	3	1
GA	2	3	3	1
AGA	2	3	1	2
AGCS	3	5	0	1

Table 4: TP, TN, FP, and FN for data set 4.

Training method	TP	TN	FP	FN
BPA	2	3	1	3
GA	3	3	1	2
AGA	3	3	2	1
AGCS	2	6	1	0

Figure 3 shows the sensitivity comparison chart of different training algorithms. From this algorithm, we can prove that the sensitivity of our proposed method is good and comparatively higher than those of the other training algorithms. The sensitivity of our proposed AGCS method is 100% except for the third data set, which also has a greater sensitivity of 75%. It proves that the AGCS technique has an overall sensitivity that is 25% greater than those of the other methods.

Figure 4 shows the specificity comparison; from this chart itself, we can clearly prove that the AGCS technique has higher specificity. Numerically, the overall specificity of AGCS is 93%, which is nearly 20% greater than that of the other training methods.

Figure 5 shows the accuracy comparison. The accuracy of our AGCS technique is also good like the other methods. The overall accuracy is about 93%, which is 20% higher than that of the other methods.

Figure 6 shows the PPV comparison chart. The PPV for our AGCS technique is 83% on average, which is comparatively 7% higher than those of the other methods.

The NPV comparison is shown in Figure 7. The overall NPV of AGCS is 96%, which is 16% higher than that of the other training algorithms, viz. ANN.

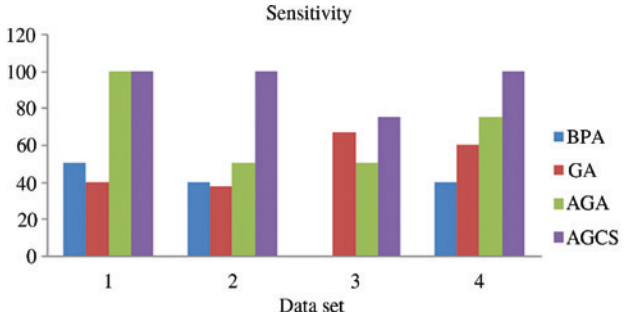

Figure 3: Sensitivity comparison of different training algorithms.

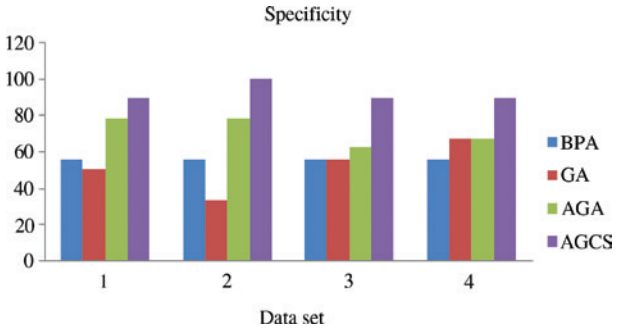

Figure 4: Specificity comparison of different training algorithms.

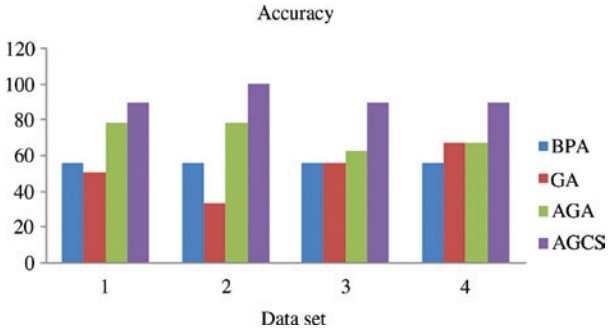

Figure 5: Accuracy comparison of different training algorithms.

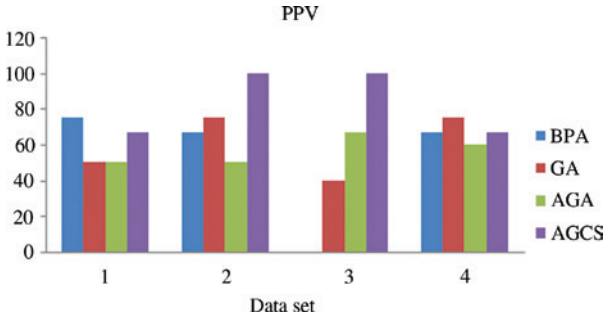

Figure 6: PPV comparison of different training algorithms.

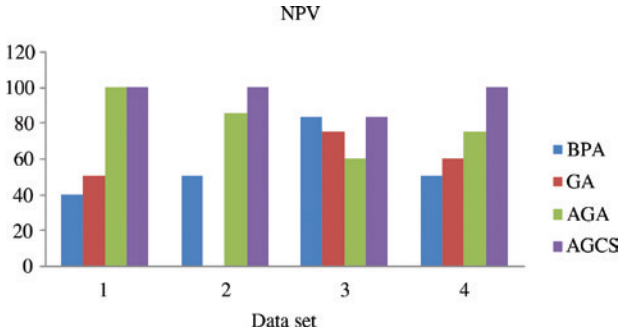

Figure 7: NPV comparison of different training algorithms.

The comparison of FPR and FDR is shown in Figures 8 and 9, respectively. Both the values have low rates, i.e. 7.1% for FPR and 17% for FDR, which are comparatively 20% and 26% lower than the other methods. It shows that our AGCS technique has higher performance.

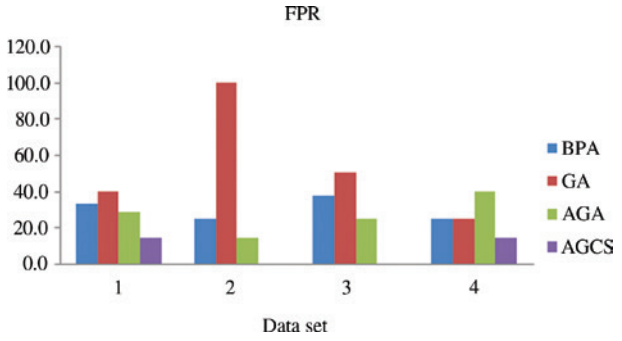

Figure 8: FPR comparison of different training algorithms.

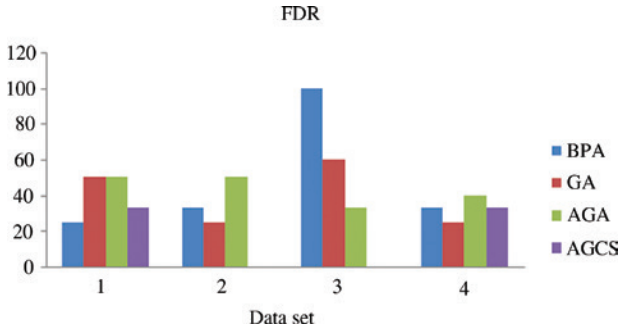

Figure 9: FDR comparison of different training algorithms.

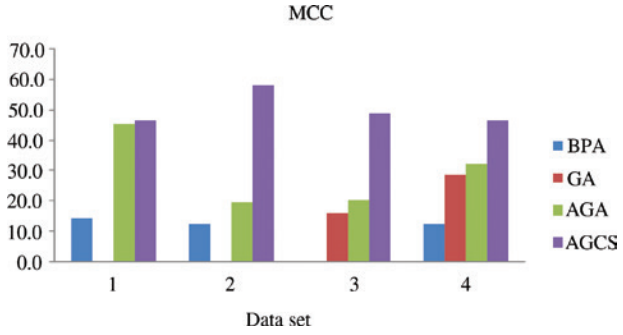

Figure 10: MCC comparison of different training algorithms.

The MCC comparison is shown in Figure 10. From the chart, we can show that the MCC value of our AGCS technique is higher than that of the other training algorithms. The overall MCC of AGCS is nearly 50%, which is 21% higher than that of the other training methods.

From these performance analyses, we can prove that our AGCS technique for training the ANN has the best performance when compared to the other training methods like BPA, GA, and AGA.

5 Conclusion

In this work, we have proposed the AGCS technique for training ANNs. It utilizes two major algorithms for training ANNs, such as the AGA and CS algorithm. These two algorithms are run separately, and the outputs are merged to make the best solution. Hence, AGCS is nothing but the hybrid of the AG and CS algorithms. By using this training technique, we can get better performance. The performance of the AGCS technique is

verified by comparing with those of the other traditional training methods. In this paper, we have used BPA, GA, and AGA for training ANN to compare their performance with our AGCS technique. In the performance review, we have proved the effectiveness of the AGCS technique. From these results, we suggest that our proposed AGCS technique is the best training algorithm for ANN vis-a-vis the other traditional training methods.

Bibliography

[1] D. Arotaritei, Genetic algorithm for fuzzy neural networks using locally crossover, *Int. J. Comput. Commun. Control* **6** (2011), 8–20.

[2] S. Das, A. Abraham, U. K. Chakraborty and A. Konar, Differential evolution using a neighborhood-based mutation operator, *IEEE Trans. Evol. Comput.* **13** (2009), 526–553.

[3] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield and E. S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* **286** (1999), 531–537.

[4] R. A. Iqbal, Using feature weights to improve performance of neural networks, *Comput. Vis. Pattern Recogn.* (2011).

[5] A. G. Karegowda, A. S. Manjunath and M. A. Jayaram, Application of genetic algorithm optimized neural network connection weights for medical diagnosis of Pima Indians diabetes, *Int. J. Soft Comput.* **2** (2011), 15–23.

[6] A. A. L. Kawam and N. Mansour, Meta heuristic optimization algorithms for training artificial neural networks, *Int. J. Comput. Inform. Technol.* **1** (2012), 156–161.

[7] W. B. Langdon, R. I. McKay and L. Spector, Genetic programming, *Int. Ser. Oper. Res. Manage. Sci.* **146** (2010), 185–225.

[8] M. Lukosevicius and H. Jaeger, Reservoir computing approaches to recurrent neural network training, *Comput. Sci. Rev.* **3** (2009), 127–149.

[9] L. Mingguang and L. Gaoyang, Artificial neural network co-optimization algorithm based on differential evolution, in: *Proceedings of Second International Symposium on Computational Intelligence and Design*, 2009.

[10] P. Mirowski, D. Madhavan, Y. LeCun and R. Kuzniecky, Classification of patterns of EEG synchronization for seizure prediction, *Clin. Neurophysiol.* **120** (2009), 1927–1940.

[11] P. V. Missiuro, *Predicting Genetic Interactions in Caenorhabditis elegans Using Machine Learning*, Thesis, Massachusetts Institute of Technology, pp. 191–204, 2010.

[12] D. J. Montana and L. Davis, Training feed-forward neural networks using genetic algorithms, in: *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 20–25 August, pp. 762–767, San Mateo, CA, USA, 1989.

[13] D. E. Moriarty, A. C. Schultz and J. J. Grefenstette, Evolutionary algorithms for reinforcement learning, *J. Artif. Intell. Res.* **11** (1999), 241–276.

[14] L. Negri, A. Nied, H. Kalinowski and A. Paterno, Benchmark for peak detection algorithms in fibre Bragg grating interrogation and a new neural network for its performance improvement, *Sensors* **11** (2011), 3466–3482.

[15] U. Okkan, Using wavelet transform to improve generalization capability of feed forward neural networks in monthly runoff prediction, *Sci. Res. Essays* **7** (2012), 1690–1703.

[16] G. V. R. Sagar, V. Chalam and M. K. Singh, Evolutionary algorithm for optimal connection weights in artificial neural networks, *Int. J. Eng.* **5** (2011), 333–340.

[17] A. M. Smith and M. Mateas, Answer set programming for procedural content generation: a design space approach, *IEEE Trans. Comput. Intell. AI Games* **3** (2011), 187–200.

[18] J. Tanomaru, Staff scheduling by a genetic algorithm with heuristic operators, in: *Proceedings of the IEEE Conference on Evolutionary Computation*, pp. 456–461, 1995.

[19] L. Wang, B. Yang, Y. Chen, A. Abraham, H. Sun, Z. Chen and H. Wang, Improvement of neural network classifier using floating centroids, *Knowl. Inf. Syst.* **31** (2012), 433–454.

[20] X. S. Yang and S. Deb, Cuckoo search via Lévy flights, in: *Proceedings of World Congress on Nature & Biologically Inspired Computing*, pp. 210–214, 2009.

[21] L. Zhang, L. Wang, X. Wang, K. Liu and A. Abraham, Research of neural network classifier based on FCM and PSO for breast cancer classification, *Hybrid Artif. Intell. Syst.* **7208** (2012), 647–654.