Using Genetic Algorithms to Optimize Artificial Neural Networks.

Article *in* Journal of Convergence Information Technology · October 2010

DOI: 10.4156/jcit.vol5.issue8.6 · Source: DBLP

4 authors, including:

Shifei Ding
China University of Mining and Technology
282 PUBLICATIONS **6,796** CITATIONS

SEE PROFILE

Li Xu
ZaoZhuang University
13 PUBLICATIONS **194** CITATIONS

SEE PROFILE

Using Genetic Algorithms to Optimize Artificial Neural Networks

[1,2]Shifei Ding, [1]Li Xu , [1]Chunyang Su, [1]Hong Zhu
[*1, Corresponding] *School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, dingsf@cumt.edu.cn*
[2,] *Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing China 100080, dingsf@cumt.edu.cn*
doi: 10.4156/jcit.vol5.issue8.6

Abstract

Artificial Neural Networks (ANNs), as a nonlinear and adaptive information processing systems, play an important role in machine learning, artificial intelligence, and data mining. But the performance of ANNs is sensitive to the number of neurons, and chieving a better network performance and simplifying the network topology are two competing objectives. While Genetic Algorithms (GAs) is a kind of random search algorithm which simulates the nature selection and evolution, which has the advantages of good global search abilities and learning the approximate optimal solution without the gradient information of the error functions. This paper makes a brief survey on ANNs optimization with GAs. Firstly, the basic principles of ANNs and GAs are introduced, by analyzing the advantages and disadvantages of GAs and ANNs, the superiority of using GAs to optimize ANNs is expressed. Secondly, we make a brief survey on the basic theories and algorithms of optimizing the network weights, optimizing the network architecture and optimizing the learning rules, and make a discussion on the latest research progresses. At last, we make a prospect on the development trend of the theory.

Keywords*: Artificial neural networks (ANNs), Genetic algorithms (GAs), Network weights, Network architecture*

1. Introduction

Artificial Neural Networks (ANNs)[1] are the nonlinear and adaptive information processing systems which are combined by numerous processing units, with the characteristics of self-adapting, self-organizing and real-time learning. From 1980s, the research of ANNs has obtained notable developments, and ANNs have been widely applied. However, with the development of the research, we've encountered many problems, such as the selection of the structure and the parameters of the networks, the selection of the learning samples, the selection of the initial values, the convergence of the learning algorithms and so on. It is known that the performance of neural networks is sensitive to the number of neurons. Too few neurons can result in poor approximation, while too many neurons may contribute to overfitting problems. Obviously, achieving a better network performance and simplifying the network topology are two competing objectives.

Genetic Algorithms (GAs) is a kind of random search algorithm which simulates the nature selection and evolution. It has the advantages of good global search abilities and learning the approximate optimal solution without the gradient information of the error functions. By its special nature evolution rules and superiority of population optimizing search, GAs can provide new ideas and methods for solving problems[2]. In recent years, the research of combination GAs with ANNs is paid attention by more and more researchers, and a field called Evolutionary Neural Networks (ENNs) is formed [3-6]. The typical characteristic of ENNs is its adaptivity to dynamic environment, the adaptivity has two forms: evolution and learning. ENNs can be seen as a self-adapting system that can automatically adjust its network architecture and learning rules. Currently the most research is the integration of GAs and ANNs[7]. They can reinforce them by the mutual complementation, so as to gain more powerful abilities of representing and solving the practical problems. GAs and ANNs both are the theoretical results of applying biological principles to the science research. More and more researchers try to combine GAs with ANNs in recent years [8-13]. This paper makes a brief survey on ANNs optimization with GAs, arranges as follows. The section 2 introduces the basic principles of ANNs and GAs. Make a survey on GAs-based Optimization of ANNs, include optimizing the network weights, optimizing the network architecture and optimizing the network learning rules in section 3. In

the section 4, we make a prospect on the development trend of the theory.

2. Basic Principles of ANNs and GAs

2. 1. Basic Principle of ANNs

ANNs are composed of a set of processing units which are also called neurons and connected with each other. It can be described as an oriented graph; each neuron is a transfer function. The neuron is usually a multi-input and single-output nonlinear element. Thee architecture of a neural network is determined by all of the connections of the network and the transfer functions of the neurons[14]. Fig. 1 is an architecture of BP neural network. In the network, there is an input layer, an output layer, with one or more hidden layers in between them.

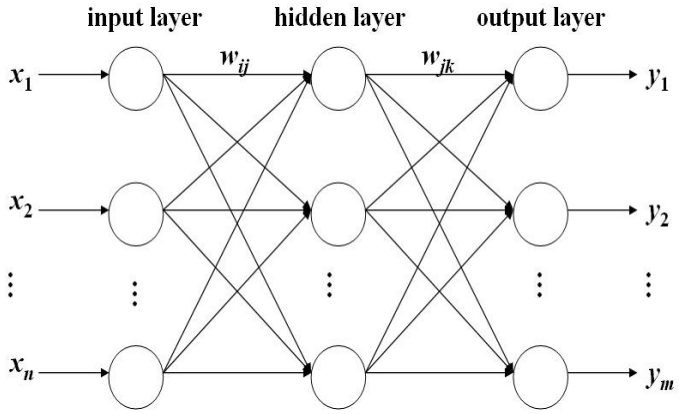

Figure 1. The architecture of BP neural network

Suppose there are n inputs and m outputs in the network, s neurons in the hidden layer, the output of the hidden layer is b_j, the threshold value of the hidden layer is θ_j, the threshold value of the output layer is θ_k, the transfer function of the hidden layer is f_1, the transfer function of the output layer is f_2, the weight from input layer to hidden layer is w_{ij}, the weight form hidden layer to output layer is w_{jk}, then we can get the output of the network y_k, the desired output is t_k, the output of j th neuron of the hidden layer is:

$$b_j = f_1\left(\sum_{i=1}^{n} w_{ij} x_i - \theta_j\right) \quad (i = 1,2,\cdots,n;\ j = 1,2,\cdots,s) \tag{1}$$

Calculate the output y_k of the output layer, that is:

$$y_k = f_2\left(\sum_{j=1}^{s} w_{jk} b_j - \theta_k\right) \quad (j = 1,2,\cdots,s;\ k = 1,2,\cdots,m) \tag{2}$$

Define the error function by the network actual output, that is:

$$e = \sum_{k=1}^{m} (t_k - y_k)^2 \tag{3}$$

The network training is a continual readjustment process to the weight and the threshold value, to make the network error reduce to a pre-set minimum or stop at a pre-set training step. Then input the forecasting samples to the trained network, and obtain the forecasting results.

The learning in ANNs is usually implemented by examples; the leaning process is also called 'training', because the learning is achieved by iteratively adjusting the connection weights. The learning of ANNs can be divided into supervised, unsupervised and reinforcement learning. Supervised learning is based on the direct compare between the actual output and the excepted output. The optimization algorithm based on gradient descent such as back propagation algorithm can be used to iteratively adjusting the connection weights to minimize the error. Reinforcement learning is a special case of supervised learning; it is only based on whether the actual output is correct. Unsupervised learning is only based on the correlation between the input data. The essence of a learning algorithm is the learning rule, which determine the weight update rule. Several popular learning rules are delta rule, Hebbian rule, and competitive learning rule.

2. 2. Basic Principle of GAs

GAs are is an important branch of evolution algorithms. It draws lessons from the theory of biological evolution which is "selecting the superior and eliminating the inefficient, survival of the fittest". After select, crossover and mutate from one generation to another, the population tends to a balance state where some features have relative potency. GAs starts from a population that represents the hidden solution set of the problem; the solution set is composed by a designated number individual got by gene coding[15,16]. In fact, each individual is the substance of some chromosome with features. A chromosome is the set of several genes, so at first it is need to implement the mapping from phenotype to genotype by coding. In each generation of evolvement, GAs select the individuals according to their fitness, then crossovers and mutates to generate a new population of the solution set. At last GAs can get the approximate optimum solution from the decoding result of the optimum individual of the final population. In practical problems, the coding methods of the feasible solution and the design of genetic operators are the two main problems when constructing GAs; different problems may use different coding methods and different genetic operators [17,18]. Fig. 2 shows the flow of chart of GAs.

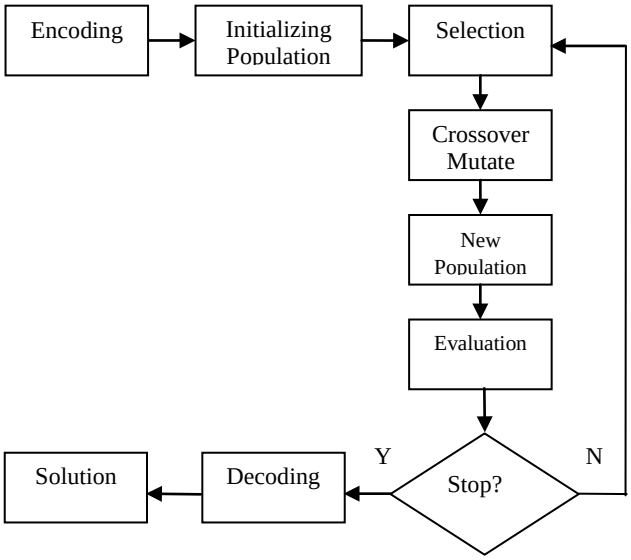

Figure 2. The flow chart of GAs

Basic GAs only used three basic genetic operators, its evolution process is simple and it is easy to understand; it is the foundation of other GAs. Basic GAs includes: the coding methods for the chromosomes, the evaluation to the individual fitness, the selection of the genetic operators and the

determination of the running parameters, etc. Basic GAs uses fixed length binary digit strings to represent the individuals; the individuals of the initial population are generated by random numbers. Basic GAs determines the probability of each individual of the current population inherited by the next generation according to the individual fitness. The genetic operators used by basic GAs are scale selection operator, single point crossover operator, basic bit mutation operator or well-proportioned mutation operator.

3. GAs-based Optimization of ANNs

ANNs have its own limitations; they are easy to fall into local minimum and sometimes hard to adjust the architecture. GAs may not be satisfactory when used in simple problems, and its performance may not be equal with the special algorithms for certain problem. But GAs are robust, nonlinear, parallel, it can be commonly used for it is based on the knowledge of any special problem. GAs can be used to process various problems, such as the training of connection weight, the design of the architecture, learning rules, selecting the input features, the initialization of the connection weights and extracting rules from ANNs. Currently the most research is the integration of GAs and ANNs.

The main parameters used to describe ANNs architecture are: the number of layers, the number of neurons of each layer, the connecting way between neurons, etc. Designing network architecture in fact is to determine the combination of the parameters that are suited to solve certain problem according to some performance evaluation rule. Kolmogorov theorem explains that three-layer feedforward network can approach any continuous function with reasonable architecture and proper weights, but the theorem has not given the method for determining the reasonable architecture, researchers can only design the architecture by the former experiences. Sometimes it is difficult to construct the network artificially, ANNs need efficient and automatic design methods, and GAs supplies a good way.

Now we will discuss the research developments of the ANNs based on GAs from the three aspects: optimizing network weights, optimizing network architecture and optimizing network learning rules.

3. 1. Optimizing Network Weights

The optimizing the connection weights draws into a self-adapting and global method to train; it is especially used in the reinforcement learning and recurrent network learning where the training algorithms based on gradient encounter great problems.

One aspect of using GAs in ANNs is to use GAs to learn the weights of ANNs, that is to use GAs to replace some traditional learning algorithms to overcome their defects. The traditional network weight training generally uses gradient descent method; these algorithms are easy to fall into local optimum and can not reach the global optimum. Training the weights by certain GAs can find the weight set that approaches global optimum while do not need to compute gradient information; the individual fitness can be defined by the error of between the expected output and actual output and the network complexity.

The steps to use GAs to optimize network weights of ANNs are as follows.

Step 1 Randomly generate a group of distribution, using some coding scheme to code the weights of the group.

Step 2 Compute the error function of the generated neural network, so as to determine its fitness function value. The error is larger, the fitness is smaller.

Step 3 Select several individuals with the largest fitness and reserve to the next generation.

Step 4 Deal with the current population using crossover, mutation and other operators and generate a new generation.

Step 5 Repeat Step 2 to Step4, make the initial weights evolve continuously, until the training goal is achieved.

The involved main problems are:

(1) Encoding scheme (determine the expression of the connection weights of the ANNs)

There are two ways to encode the connection weights and the threshold values in the ANNs. One is binary encoding, the other is real encoding. Binary encoding is that each weight is expressed by fixed length 0-1 string. When the encoding length is limited, the expression precision of binary encoding is not enough; if certain precision limit is satisfied, the search space will increase correspondingly and

hurt the speed of the evolutionary process. Real encoding is to express each weight with a real number; it overcomes the abuse of binary encoding, but it needs to re-design the operators. Montana et al. successfully applied real encoding GAs in the evolution of large-scale neural network weights.

(2) Determine the fitness function

Using GAs to evolve the weights of ANNs, if the network architecture is fixed, generally, the network with big error, the fitness is small. For example, the fitness function may be: F=C-E, Here F is the fitness of the individual, C is a big number, E is the training error of the network.

(3) Evolution process

Determine the global search operator of the algorithm, such as selection, crossover and mutation operator, also can design special operators.

(4) Train the network

Because GAs are good at searching large-scale, complex, non-differentiable and multimodal spaces; it doesn't need the gradient information of the error function. On the other hand, it doesn't need to consider whether the error function is differentiable, so some punishment may be added to the error function, so as to improve the network commonality, reduce the complexity of the network.

The results of GAs and BP algorithm are both sensitive to the parameters, the result of BP algorithm also depends on the original state of the network, but BP algorithm appears to be more effective when used in local search, GAs are good at global search. On the fact of this, connection weight evolution algorithm can be implemented in the following way. First, use GAs to optimize the initial weight distribution and locate some better search spaces in the solution space. Then use BP algorithm to search the optimal solution in these small solution spaces. Generally, the efficiency of the hybrid training is superior to the training methods that only uses BP evolution or only use BP training [19].

The problem that should be considered when using GAs to train ANNs is the permutation of the algorithm[20], that is the algorithm converges to locally optimal solution. This is mainly because the phenomenon of several to one mapping from gene space to actual solution space. Generally speaking, it is often because of the appearance of the networks which have the same functions but different gene sequences. The permutation phenomenon caused by this reason makes the inefficiency of crossover operation; good filial generation individuals can not be generated. So EP and ES which are mainly mutation operations will better restrain the detrimental effects caused by permutation than GAs which is mainly crossover operation.

3. 2. Optimizing Network Architecture

The architecture of the ANNs has important influence on the information process ability of the ANNs. At present, the two commonly used methods for designing the network architecture are adding method and reducing method [21,22]; the adding method adds some neurons and connections form the possible least scale and the reducing method is opposite. These methods are liable to trap in local minimum value and the search space is a small sub-space of the entire space [23]. The evolution of the architecture of the ANNs make different tasks can fit the network topology.

For a given problem, the process ability of an ANNs with few connections and hidden neurons is limited, but if the connections and hidden neurons are so many, the noise may be trained together and the generalization ability of the network will be poor. Formerly people often used try error try method to design the architecture, the effect overly depends on the subjective experience.

The architecture of the ANNs includes the network connections and the transfer functions. A good architecture can solve the problems in a satisfactory way, and doesn't allow the existence of redundant nodes and connections. With the development of GAs, people consider regarding the design of the network as a search problem, using learning accuracy, generalization ability, and noise immunity as the evaluation standard, find the best architecture with best performance in the architecture space. The development of architecture evolution is mainly reflected in the architecture coding and operator design.

The steps of using GAs to evolve the ANNs architecture are as follows.

Step 1 Randomly generate N architectures, code each architecture.

Step 2 Train the architectures in the individual set using many different initial weights.

Step 3 Determine the fitness in each individual according to the trained result and other strategies.
Step 4 Select several individuals whose fitness are the largest, directly genetic to the next generation.
Step 5 Do crossover and mutation to the current population, so as to generate the next generation.
Step 6 Repeat Step 2 to Step 5, until some individual in the current population meets the demand.

The network architecture includes much information, such as the number of hidden layers, the number of neurons, the connection way, the transfer functions, etc. A good coding way should include useful information as much as possible and exclude the useless information which can interfere the evolution, and the coding length should not be very long. The current coding methods can be divided into two main classes: direct coding and indirect coding.

Direct coding is to code each connection into a binary string, and it uses a square C to express. $C=(Cij)N\cdot N$, N is the number of the network nodes, C_{ij} explains whether there're connections between network nodes, $C_{ij}=n$ (n>0,$n \in R$) represents existing and the weight is n, $C_{ij}=0$ represents not existing. So, each matrix expresses an ANNs, connecting all of the rows we can get a binary string which corresponds to the ANNs architecture. The particularity of the ANNs can be expressed by special matrix form. This coding pattern is simple, direct and is applicable to the evolution of the ANNs with simple architecture, when the architecture of the ANNs is complex, the coding length will be very long and the search space will increase observably. The modified method is to use domain knowledge to initialize the matrix. Fullmer proposed a direct coding method based on label which only codes the existent connections. This method also has no limitation on the number of neurons and the connections, so it can ensure the network architecture perform well.

Indirect coding includes parameterization coding and growing rule coding. Parameterization coding only codes the most important characteristics of the related architecture, such as the number of hidden layers, the number of nodes of each layer, the number of connections, etc. It gives the rough connection pattern but doesn't give the detailed information. This method can observably reduce the length of the string, but have some limitations to the network architecture; it will reduce the search space, and may lead to not finding the optimum network architecture.

Developmental rule representation organizes the developmental rule into the chromosome, evolves the rules continuously and generates suitable neural network. Developmental rule representation can be use to design large-scale networks, it has good regularity and generalization ability, This approach is capable of preserving promising building blocks found so far, and can thus lessen the damage of crossover. The connectivity pattern of the architecture in the form of a matrix is constructed from a single-element matrix, by repeatedly applying suitable developmental rules to nonterminal elements in the current matrix until the matrix only contains terminal elements that indicate the presence or absence of a connection. The developmental rule representation method normally separates the evolution of architectures from that of connection weights[24]. As a consequence, the direct encoding scheme of network architectures is very good at generating a compact architecture, while the indirect encoding scheme is suitable for finding a particular type of network architecture quickly.

In the genetic backpropagation (G-Prop) method [25], the GAs select the initial weights and changes the number of neurons in the hidden layer through the application of five specific genetic operators, that is, mutation, multipoint crossover, addition, elimination and substitution of hidden units. Addition operator and elimination operator are used to adaptively adjust the number of the neurons in hidden layers, the addition operator is used for adding new neurons when necessary, while the elimination operator is used for preventing the network growing too much. The fitness is determined by the hidden layer size and the classification ability. The BP is used to train from these weights. This makes a clean division between global and local search.

3. 3. Optimizing Network Learning Rules

The optimization to the learning rules can be seen as a process of "learning to learn", the self-adapting of the learning rules is implemented by evolution, it can be seen as a self-adapting process of automatically finding new learning rules. Learning rules determine the functions of the system. In the traditional ANNs training, learning rules are previously assigned, such as the generalized δ rule of BP

network; the design of the algorithms mainly depends on special network architecture, and it is difficult to design a learning rule when there is not enough prior knowledge of the network architecture. So we can use GAs to design the learning rules of ANNs [26,27].

The steps of optimizing network learning rules are as follows.

Step 1 Randomly generate N individuals; each individual represents a learning rule.
Step 2 Construct a training set, each element in it represents a neural network whose architecture and connection weights are randomly assigned or previously determined, then train the elements in the training set using each learning rule respectively.
Step 3 Compute the fitness of each learning rule, select according to the fitness.
Step 4 Generate next generation by doing genetic operations to each individual.
Step 5 Repeat Step 2 to Step 4, until the goal is reached.

There are many parameters in learning rules. These parameters are used to adjust the network behavior, such as learning rate can speed up the network training. At the stage of coding, code the learning parameters and the architecture together, then evolve the architecture, at the same time the learning parameters are evolved. Belew use GAs to find the optimization of the learning parameters of BP algorithm under the premise that the network architecture is fixed. Harp et al. evolved the network architecture and learning parameters at the same time. The learning parameters got by the former algorithm are near-optimality to special network architecture, but its generalization is relatively poor; the latter can get the optimalizing combination of architecture and parameters, but it pays the price of big search space and slow convergence velocity.

The objects of evolving the learning rules are the learning rule itself and the weight adjustment rule; evolving the learning rules makes the evolved network suit to the dynamic environment much better. Different from optimizing the connection weights and the network architecture, optimizing the learning rules is dynamic behavior. Learning rules are the heart of ANNs training algorithms. Optimizing the learning rules implies two basic assumptions: firstly, the update of weights only depends on the local information of the neurons; secondly, all of the connections use the same learning rule. Following the above assumptions, Chalmers defined the weight adjustment rule as the linear function about 4 independent variables and the their products, evolving the original population after about 1000 generations, Chalmers got the famous δ rule and some of its variants, this fully shows the optimization is effective.

GAs can be used to train all kinds of neural networks, irrespective of the network topology. To date, GAs are mainly used for training feed-forward neural networks, such as multilayer perceptron, radial basis function network, etc. Some researchers used it in recurrent neural network [28]. With the development of GAs, many new GAs are used for evolving the ANNs, such as multi-objective evolutionary algorithm[29], parallel genetic algorithm[30], etc.

4. Prospects

ANNs and GAs are the artificial intelligence methods that borrow ideas from some behavior characteristics and structure attribute of biological individuals or biological world. ANNs place extra emphasis on the description of biological individual learning intelligence. And GAs simulates the evolution characteristics of biological world. So, the combination of GAs and ANNs will show more complete intelligence. This combination not only makes ANNs have the capabilities of learning and evolving, show stronger intelligence, but also can solve some problems that exist in the design and realization of ANNs, make ANNs have more excellent performance. Currently, the design and realization of ANNs based on GAs have become an important research and developing direction of ANNs. After the appearance of artificial life, the integration of GAs and ANNs are considered as a promising way for reproduce intelligent behavior.

However, in general, the researches of this aspect still have problems, the theories and methods need to be improved and standardized, and the application researches need to be strengthened. The current researches are mostly based on special cases, the general methodology has not been formed, it needs to compare with other optimization algorithms, such as particle swarm optimization and so on; the evolved networks are usually feed-forward networks, the evolved algorithms for other class of

networks need to be studied; the calculation is complex, the evolution processes need to be improved, especially the parallel GAs methods should be valued; the GAs theory itself, such as the representation of coding, the genetic operators, the population convergence and multiformity, etc. needs to be further improved and developed; currently, the reaearches of ANNs, GAs, fuzzy logics, chaos, wavelet, fractal are intercrossed and penetrated, some new valuable evolved ANNs and their applications emerge continuously. The researches of evolving ANNs make the design of ANNs no longer formidable, and make the superiority of ANNs better reflect in the more large-scale and complex networks. Computation and algorithms are the research areas that require a great deal of emphasis from ancient times.

5. Acknowledgements

This work is supported by the Basic Research Program (Natural Science Foundation) of Jiangsu Province of China (No.BK2009093), and the National Natural Science Foundation of China (No.60975039).

6. References

[1] S. Mccllochw, W. Pitts, “A Logical Calculus of the Ideas Immanent in Nervous Activity,” Bulletin of Mathematical Biophysics, VoL.10, No.5, 1943, pp. 115-133.
[2] L.D. Davis, “Handbook of Genetic Algorithms,” Newyork: van Nostrand Reinhold, 1991.
[3] W.S. Yao, Q. Wan, Z.Q. Chen, et al., “The researching overview of evolutionary neural networks”, Computer Science, Vol.3, No.1, pp. 125-129, 2004.
[4] X. Yao, “A review of evolutionary artificial neural networks,” International Journal of Intelligent Systems, Vol. 8, No. 4, pp.539-567, 2007.
[5] X. Yao, “Evolving Artificial Neural Networks,” Proceedings of the IEEE, Vol. 87, No. 9, 1423-1447, 1999.
[6] J.X. Xie, “A brief review on evolutionary computation”, Control and Decision, Vol.12, No. 1, pp.1-7, 1997.
[7] L. Zhong, W.B. Rao, C.M. Zou, “Artificial Neural Networks and the Integrated Application Techniques,” Beijing: Science Press, 2007.
[8] D. Venkatesan, K. Kannan, R. Saravanan, “A genetic algorithm-based artificial neural network model for the optimization of machining processes,” Neural Computing and Applications, Vol. 18, No. 2, pp. 135-140, 2009.
[9] D. Whitley, T. Starkweather, C. Bogart, “Genetic algorithms and neural networks: optimizing connection and connectivity,” Parallel Computing, Vol.14, No. 3, 347-361, 1990.
[10] Z.W. Ren, Z. San, “Improvement of Real-valued Genetic Algorithm and Performance Study”, Acta Electronica Sinica, Vol.35, No. 2, 269-274, 2007.
[11] D.J. Montana, L. Davis, “Training Feedforward Neural Networks Using Genetic Algorithm”, Proceedings of IJCAI, pp.762-767, 1989.
[12] X.P. Chen, S.L. Yu, “Improvement on crossover strategy of real-valued genetic algorithm”, Acta Electronica Sinica, Vol.31, No. 1, pp.71-74, 2003.
[13] Z.J. Zheng, S.Q. Zheng, “Study on a Mutation Operator in Evolving Neural Networks”, Journal of Software, Vol.13, No. 2, pp.726-731, 2002.
[14] S.F. Ding, W.K. Jia, C.Y. Su, et al., "An Improved BP Neural Network Algorithm Based on Factor Analysis", JCIT: Journal of Convergence Information Technology, Vol.5, No.4, pp.103-108, 2010.
[15] J. Hertz, “Introduction to the Theory of Neural Computation,” MA: Addison-Wesley Press, 1991.
[16] X. Yao, Y. Xu, “Recent Advances in Evolutionary Computation,” Journal of Computer Sciences and Technology. Vol. 21, No. 1, 1-18, 2006.
[17] G.Q. Cai, L.M. Jia, J.W. Yang, et al., "Improved Wavelet Neural Network Based on Hybrid Genetic Algorithm Applicationin on Fault Diagnosis of Railway Rolling Bearing", JDCTA: International Journal of Digital Content Technology and its Applications, Vol. 4, No. 2, pp. 13-41, 2010.
[18] M.S. Soliman, G.Z. Tan, "Conditional Sensor Deployment Using Evolutionary Algorithms", JCIT: Journal of Convergence Information Technology, Vol. 5, No. 2, pp. 146~154, 2010.

[19] X.P. Meng, H.G. Zhang, W.Y. Tan, "A Hybrid Method of GAs and BP for Short-Term Economic Dispatch of Hydrothermal Power Systems", Mathematics and Computers in Simulation, Vol.51, No. 3-4, pp.341-348, 2000.
[20] X. Yao, Y. Liu, P. Darwen, "A New Evolutionary System for Evolving Artificial Neural Networks", IEEE Transactions on Neural Networks, Vol. 8, No. 3, pp.694-713, 1997.
[21] M. Fream, "The upstart algorithm: A method for constructing and training feedforward neural networks," Neural Computation, Vol. 2, No. 2, pp. 198-209, 1990.
[22] A. Roy, L.S. Kim, S. Mukhopaduyay, "A polynomial time algorithm for the construction and training of a class of multilayer perceptrons," Neural Networks, Vol. 6, No. 4, pp.535-545, 1993.
[23] P.J. Angeline, "An evolutionary algorithm that constructs recurrent neural networks," IEEE Transactions on Neural Networks, Vol. 5, No. 2, pp. 54-65, 1994.
[24] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," Complex systems, Vol. 4, No. 4, pp. 461-476, 1990.
[25] Castillo, P. A., Merelo, J. J., Rivas, V. et al., "G-Prop: Global optimization of multilayer perceptrons using GAs", Neurocomput, Vol.35, pp.149-163, 2000.
[26] A. Alvarez, "A neural network with evolutionary neurons," Neural Process Letter, Vol. 16, No. 1, pp. 43-52, 2002.
[27] H.B. Kim, S.H. Jung, T.G. Kim et al., "Fast learning method for back-propagation neural network by evolutionary adaptation of learning rates," Neurocomput, Vol. 11, No. 1, pp. 101-106, 1996.
[28] J.A. Peter, M.S. Gregory, B.P. Jordan, "An Evolutionary Algorithm that Constructs Recurrent Neural Networks", IEEE Transactions on Neural Networks, Vol.5, No. 1, pp.54-65, 1994.
[29] G.G. Yen, "Multi-Objective Evolutionary Algorithm for Radial Basis Function Neural Network Design", Studies in Computational Intelligence, Vol.16, pp.221-239, 2006.
[30] S. Mendivil, O. Castillo, P. Melin, "Optimization of Artificial Neural Network Architectures for Time Series Prediction Using Parallel Genetic Algorithms," Soft Computing for Hybrid Intelligent Systems, Vol.154, pp.387-399, 2008.