Evolutionary artificial neural networks: A review

Article *in* Artificial Intelligence Review · March 2013

DOI: 10.1007/s10462-011-9270-6

5 authors, including:

Shifei Ding
China University of Mining Technology
204 PUBLICATIONS **1,684** CITATIONS

SEE PROFILE

Artif Intell Rev (2013) 39:251–260
DOI 10.1007/s10462-011-9270-6

Evolutionary artificial neural networks: a review

Shifei Ding · Hui Li · Chunyang Su · Junzhao Yu · Fengxiang Jin

Published online: 17 June 2011

Abstract This paper reviews the use of evolutionary algorithms (EAs) to optimize artificial neural networks (ANNs). First, we briefly introduce the basic principles of artificial neural networks and evolutionary algorithms and, by analyzing the advantages and disadvantages of EAs and ANNs, explain the advantages of using EAs to optimize ANNs. We then provide a brief survey on the basic theories and algorithms for optimizing the weights, optimizing the network architecture and optimizing the learning rules, and discuss recent research from these three aspects. Finally, we speculate on new trends in the development of this area.

Keywords Artificial Neural Networks (ANNs) · Evolution Algorithms (EAs) · Weights · Network architecture

1 Introduction

Artificial Neural Networks (ANNs) (Mccllochw and Pitts 1943) are adaptive nonlinear information processing systems which combine numerous processing units with a series of characteristics such as self-adapting, self-organizing and real-time learning. Since the 1980s, research on the ANNs has made remarkable developments, and ANNs have been widely applied. Despite the development of the research, we have encountered many problems as well. For instance, 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, etc.

S. Ding (✉) · H. Li · C. Su · J. Yu
School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, 221008 Jiangsu, China
e-mail: dingsf@cumt.edu.cn; dingshifei@sina.com

F. Jin
Geomatics College, Shandong University of Science and Technology, Qingdao, 266510 Shandong, China

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, it is contradictory between achieving a better network performance and simplifying the network topology. Since the 1990s, Evolution Algorithms(EAs) have been successfully used for optimizing the design and the parameters of ANNs (Whitley 2001).

EA is a random search algorithm which simulates the nature selection and evolution process (Xie 1997). It has the advantage of good global searching capability and learning the approximate optimal solution without the gradient information of the error functions. EA provides new ideas and methods for problem solving approach in terms of its special natural evolution rules and the superiority of population optimizing search.

Currently, intelligent computation is at the stage of rapid growth, its main techniques including fuzzy technology, neural network, evolutionary algorithm, etc. With the development of computer technology, these methods have already achieved huge progress and exhibited a merging tendency. They can reinforce each other by their mutual complementation; therefore, gaining more powerful abilities of representing and solving the practical problems. Both Evolutionary algorithms and ANNs are the theoretical results of applying biological principles to the science research. In recent years, more and more researchers try to combine EAs with ANNs (Yao et al. 2004), hope that by combining the advantages of them, they can create a more efficient method. This is mainly manifested in using EAs to optimize the network design, pre-process the network input data, the assemble of ANNs (Yao 1993), etc.

2 The basic principles of ANNs and EAs

2.1 The basic principles of ANNs

The advantages of the ANNs are mainly represented by the network architecture and the algorithms. Currently, the research of neural network is mainly focus on these two aspects.

An ANN consists of a set of processing units which also called neurons, and they are connected to each other. It can be described as an oriented graph, and each neuron is a transfer function. A neuron is usually a multi-input and single-output nonlinear element. The architecture of a neural network is determined by all the connections of the network and the transfer functions of the neurons.

2.1.1 The learning process in ANNs

The learning process in ANNs is usually implemented by examples; it is also called 'training', because the learning process is achieved by iteratively adjusting the connection weights. This learning process of ANNs can be divided into three types: supervised, unsupervised and reinforcement learning. Supervised learning is based on the direct compare between the actual output and the expected output. The optimization algorithms are based on gradient descent such as back propagation algorithm, they can be used to iteratively adjust the connection weights thereby minimizing 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 of 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 Evolutionary algorithms

EAs are a class of stochastic search and optimization techniques obtained by natural selection and genetics. They are population-based algorithms by simulating the natural evolution of biological systems. Individuals in a population compete and exchange information with one another. There are three basic genetic operations: selection, crossover, and mutation. The procedure of a typical EA is displayed as follows:

Step 1 Set t = 0.
Step 2 Randomize the initial population P(t).
Step 3 Evaluate the fitness of each individual of P(t).
Step 4 Select individuals as parents from P(t+1) based on the fitness.
Step 5 Apply search operators (crossover and mutation) to parents, and generate P(t+1).
Step 6 Set t = t + 1.
Step 7 Repeat step 3 to step 6 until the termination criterion is satisfied.

EAs are stochastic processes performing searches over a complex and multimode space. They are population-based algorithms by simulating the natural evolution of biological systems (other population-based algorithms such as particle swarm optimization, immune algorithm, ant colony algorithm). They all have the following advantages:

1. EAs can solve difficult problems reliable and fast. Thus, they are suitable for evaluation functions that are large, complex, noncontinuous, nondifferentiable, and multimodal.
2. The EA approach is a general-purpose approach that can be directly interfaced to existing simulations and models. EAs are extendable and easy to hybridize.
3. EAs are directed stochastic global search. They can reach the nearoptimum or the global maximum invariably.
4. EAs possess inherent parallelism by evaluating multipoint simultaneously.

EAs mainly include Evolutionary Strategy (ES), Evolutionary Programming (EP) and Genetic Algorithm (GA). The three evolutionary algorithms are consistent on the goal of using biological evolution mechanism to improve the ability of using computers to solve problems. However, they are different on the concrete measure: ES emphasizes the behavior change on the individual level, EP emphasizes the behavior change on the population level, and GA emphasizes the operation of the chromosome. In addition, there are also Genetic Programming, Memetic Algorithm, etc.

3 The optimization of ANNs based on EAs

Neural Network has its own limitations. It is easy to fall into local minimum and sometimes hard to adjust the architecture. EAs may not be satisfactory when used in simple problems, and its performance may not be equal with the special algorithms for certain problem. But GA is robust, nonlinear and parallel, it can be commonly used for it is based on the knowledge of any special problem. EC can be used to process various problems, such as the training of the connection weights, designing the architecture, learning rules, selecting the input features, initializing the connection weights and extracting rules from ANNs.

To describe neural network architecture, the main parameters are: the number of layers, the number of neurons of each layer, the connecting way between neurons, etc. Designing network architecture is to determine the combination of the parameters that is suited to solve certain problem according to some performance evaluation rules. Kolmogorov theorem speaks that three-layer feedforward network can approach any continuous function with reasonable architecture and proper weights, but the theorem has not give the way to determine the reasonable architecture, researchers can design the architecture only by the experiences. Sometimes it is difficult to construct the network artificially, what ANNs need is efficient and automatic methods to design, luckily, GA supplies a good way.

Now we will discuss the recent research developments on the ANN based on EC from the three aspects: optimizing weights, optimizing network architecture and optimizing learning rules.

3.1 The evolution to the weights of ANNs

The evolution of the connection weights draws into a self-adapting and global method for training;especially used in the reinforcement learning and recurrent network learning which based on gradient when encounter great problems.

One aspect of using EAs in ANNs is to learn the weights of ANNs, that is to say to replace some traditional learning algorithms to overcome their defects. The traditional network weight training generally uses gradient descent method (Hertz and Krogh 1991); these algorithms are easy to fall into local optimum but can not reach the global optimum (Sutton 1986). Training the weights by certain EA can find the weight set which approaches global optimum while do not need to compute the gradient information; the individual fitness can be defined by the error between the expected output and actual output and the network complexity.

The steps to use GA to evaluate the ANNs' connection weights are as follows:

Step 1 Randomly generate a group of distribution, using a 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 to 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 Step 4, make the initial weights evolve continuously, until the training goal is achieved.

The main problems involved:

1. Encoding scheme (determine the expression of the connection weights of the ANN)

There are two ways to encode the connection weights and the threshold values for the ANNs. One is binary encoding, the other is real encoding. Binary encoding (Whitley et al. 1990) 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 effect the speed of the evolutionary process. Real encoding is to express each weight with a real number (Ren and San 2007); it overcomes the demerit of binary encoding, but needs to re-design the operators. Montana et al. first successfully applied real encoding GA to the evolution of large-scale neural network weights (Montana and Davis 1989).

2. Determine the fitness function

When using GA to evolve the weights of ANNs, if the network architecture is fixed, generally, if the network with big error, the fitness is small. For example, the fitness function may be: F = C-E, C represent a big constant, E is the error.

3. Evolution process

Determine the global search operators of the algorithm, such as selection, crossover (Chen and Yu 2003) and mutation (Zheng and Zheng 2002) operator, which also can design special operators.

4. Train the network

Because GA is 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 punishments may be added into the error function, so as to improve the network commonality, reduce the complexity of the network.

The results of GA 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, while GA is good at global search. On the fact of this, connection weight evolution algorithm can be implemented in the following way. First, use GA 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 use BP evolution or only use BP training (Meng et al. 2000).

The problem that should be considered when using EAs to train ANNs is the permutation of the algorithm (Yao et al. 1997), 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 result in the inefficiency of crossover operation; and good filial generation individuals can not be generated. Consequently EP and ES which are mainly rely on mutation operations will better restrain the detrimental effects caused by permutation than GA which is mainly depend on crossover operation.

3.2 The evolution to the architecture of ANNs

The network architecture is usually predefined and fixed. The design of the optimal architecture can be treated as a search problem in architecture space, where each point represents an architecture. The architecture of the ANNs has important influence on the information process ability of the ANNs, which includes the network connections and the transfer functions. A good architecture can solve problems in a satisfactory way, and doesn't allow the existence of redundant nodes and connections. For a given problem, the process ability of an ANN with few connections and hidden neurons is limited, but if the connections and hidden neurons are too many, the noise may be trained together and the generalization ability of the network will be poor. Formerly people often used try and error method to design the architecture, but the effect overly depends on the subjective experience. At present, the two commonly used methods for designing the network architecture are adding method and reducing method (Fream 1990; Sietsma and Dow 1991; Roy et al. 1993); 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. This has promoted the research on how to identify an optimal and efficient neural network structure. AIC (Akaike Information Criterion) (Murata et al. 1994) and PMDL (Predictive Minimum Description Length) (Gao et al. 1996) are two well-adopted approaches. However, AIC can be inconsistent and has a tendency to overfit a model, while PMDL only succeeds in relatively simple neural network structures and seems very difficult to extend to a complex NN structure optimization problem. With the development of EAs, people regard the design of the network as a search problem, using learning accuracy, generalization ability, and noise immunity as evaluation criterion, find the best architecture with best performance in the architecture space. The evolution of the architecture makes the ANNs can fit the network topology in different tasks.The development of architecture evolution is mainly reflected in the architecture coding and operator design.

The steps of using GA to evolve the ANN architecture are as follows:

Step 1 Randomly generate N architecture,and code each architecture.
Step 2 Train the architecture in the individual set using many different initial weights.
Step 3 Determine the fitness of each individual according to the trained result and other strategies.
Step 4 Select several individuals whose fitness is the largest, directly passing on to the next generation.
Step 5 Do crossover and mutation to the current population, to generate the next generation.
Step 6 Repeat Step 2 to Step 5, until some individuals in the current population meet 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 encoding method should include useful information as much as possible and exclude the useless information which can interfere with the evolution, and the coding length should not be very long. The current encoding methods can be divided into two main classes: direct encoding and indirect encoding.

Direct encoding is to code each connection into a binary string, and it uses a square C to express. $C = (C_{ij})_{\mathrm{N}\cdot\mathrm{N}}$, N is the number of the network nodes, C_{ij} explains whether there're connections among the network nodes, $C_{ij} = n(n > 0, n \cdot \mathbf{R})$ represents existence and the weight is n, and $C_{ij} = 0$ represents inexistence. So, each matrix expresses an ANN, connecting all of the rows we can get a binary string which corresponds to an architecture of the ANNs. The particularity of the ANNs can be expressed by special matrix form. This encoding method is simple, direct and applicable to the evolution of the ANNs with simple architecture, when the architecture of the ANNs is complex, the encoding length will be very long and the search space will increase observably. Prior knowledge can be used to reduce the size of the matrix. For example, for the MLP, two adjacent layers are in complete connection, and therefore its architecture can be encoded by the number of hidden layers and the number of hidden units in each layer. This leads to indirect encoding.

Indirect encoding includes parameterization representation and developmental rule representation. Parameterization representation 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 a rough connection pattern but doesn't give detailed information. This method can observably reduce the length of the string. In this case, EAs can only search a limited subset of the whole feasible architecture space. This parametric representation method is the most suitable when the type of architecture is known.

Developmental rule representation organizes the developmental rule into the chromosome, evolves the rules continuously and generates suitable neural network. Developmental rule representation can be used to design large-scale networks, which 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 architecture from that of connection weights (Kitano 1990).

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 (Castillo et al. 2000), the GA selects 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, and 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 these weights. This makes a clean division between global search and local search.

In the process of the evolution of the network architecture, the computation of the fitness often uses the randomly generated weights; this will draw into noises inevitably. In order to reduce the noises as much as possible, many researchers proposed to evolve the architecture and the weights at the same time. Yao and Liu used EP algorithm to develop a neural network automatic design system—EPNet (Yao et al. 1997). EPNet used five mutation operators: hybrid training, node deletion, connection deletion, connection addition, and node addition. Behaviors between parents and their offspring are linked by various mutations, such as partial training and node splitting. The evolved neural network prefers the node/connection deletion operations to the node/connection addition operations so as to keep the network simple. The hybrid training operator that consists of a modified BP with adaptive learning rates and the simulated annealing is used for modifying the connection weights. After the evolution, the best evolved neural network will be further trained using the modified BP.

Peter. et al. thought the prospect of using GA to evolve the architecture of ANNs was limited (Peter et al. 1994). Because GA may exclude some hidden and useful networks, and the crossover operation will bring a series of problems. Compared with GA, EP uses mutation as the only gene recombination operation, it uses more natural expression to the tasks and it is more suitable for evolving the neural network architecture.

3.3 The evolution to the learning rules of ANNs

Different activation functions have different properties and different learning rules have different performance. For example, the activation functions can be evolved by selecting among some popular nonlinear functions such as the Heaviside, sigmoidal, and Gaussian functions (Alvarez 2002). The learning rate and the momentum factor of the BP algorithm can be evolved (Kim et al. 1996), and learning rules can also be evolved to generate new learning

rules. EAs are also used to select proper input variables for neural networks from a raw data space of a large dimension, that is, to evolve input features (Guo 1992).

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 GA to design the learning rules of ANNs, the steps 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, then 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 the learning rate can speed up the network training. At the stage of encoding, encode the learning parameters and the architecture together, then evolve the architecture, at the same time the learning parameters are evolved. Belew used GA to find the optimization of the learning parameters of BP algorithm under the premise where 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 algorithms 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 suitable to the dynamic environment much better. Learning rules are the heart of ANNs training algorithms. Evolving the learning rules implies two basic assumptions: first, the update of weights only depends on the local information of the neurons; second, all of the connections use the same learning rule.

EAs can be used to train all kinds of neural networks, irrespective of the network topology. To date, EAs are mainly used for training feed-forward neural networks, such as multilayer perceptron (Yao 1999), radial basis function network (Harpham et al. 2004), etc. Some researchers used it in recurrent neural network. With the development of EAs, many new EAs are used for evolving the ANNs, such as multi-objective evolutionary algorithm (Yen 2006), parallel genetic algorithm (Castillo et al. 2008), etc. Multi-objective evolutionary algorithm is used to evolve a set of near-optimal neural networks from the perspective of Pareto optimality to the designers, so that they can have more flexibility for the final decision-making based on certain preferences.

4 Prospects

ANNs and EAs 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 EAs simulate the evolution characteristics of biological world. So, the combination of EAs and ANNs will show more complete intelligence. This combination not only makes ANNs

have the capabilities of learning and evolving,and show stronger intelligence, but also can solve some problems that exist in the design and realization of ANNs,and make ANNs have more excellent performance. Currently, the design and realization of ANNs based on EAs have become an important research and developing direction of ANNs. After the appearance of artificial life, the integration of EAs and ANNs is considered as a promising way for reproducing 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, and the evolved algorithms for other class of networks need to be studied; the calculation is complex,and the evolution processes need to be improved, especially the parallel EA methods should be valued; the EA 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, EAs, fuzzy logics, chaos, wavelet and 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 ANN 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. In 1930s, Church, Kleene, Godel, Post, Turing and other mathematicians gave the exact mathematical definition of computability algorithms; This has a great impact on the later computation and algorithms. After 1980s, ANNs obtained noticeable results on the computation theories, the concepts of neural computation and EAs were formed, and attracted strong interests of many theorizers. Large-scale parallel computation is a fundamental shock to the discrete symbol theory based on Turing machine. In 1990s, people accepted it critically, and combined the two together. In recent years, neural computation and EAs are very active, having new developing directions.

Acknowledgments This work is supported by the Basic Research Program (Natural Science Foundation) of Jiangsu Province of China (No.BK2009093), and the National Nature Science Foundation of China (No.41074003).

References

Alvarez A (2002) A neural network with evolutionary neurons. Neural Process Lett 16(1):43–52

Castillo P, Merelo J, Rivas V et al (2000) G-prop: global optimization of multilayer perceptrons using gas. Neurocomput 35(4):149–163

Castillo O, Melin P, Kacprzyk J et al (2008) Optimization of artificial neural network architectures for time series prediction using parallel genetic algorithms. Soft Comput Hybrid Intell Syst 154:387–399

Chen X, Yu S (2003) Improvement on crossover strategy of real-valued genetic algorithm. Acta Electronica Sinica 31(1):71–74

Fream M (1990) The upstart algorithm: a method for constructing and training feedforward neural networks. Neural Comput 2(2):198–209

Gao X, Ovaska S, Hartimo Z (1996) Speech signal restoration using an optimal neural network structure. In: Proceedings of the IEEE intenational conference on neural networks, pp 1841–1846

Guo Z (1992) Using genetic algorithms to select inputs for neural networks. In: Proceedings of the IEEE international workshop on combinations of genetic algorithms and neural networks, pp 223–234

Harpham C, Dawson C, Brown M (2004) A review of genetic algorithms applied to training radial basis function networks. Neural Comput Applic 13(3):193–201

Hertz J, Krogh A (1991) Introduction to the theory of neural computation. Addison-Wesley Press, Boston

Kim H, Jung S, Kim T et al (1996) Fast learning method for back-propagation neural network by evolutionary adaptation of learning rates. Neurocomput 11(1):101–106
Kitano H (1990) Designing neural networks using genetic algorithms with graph generation system. Complex Syst 4(4):461–476
Mccllochw S, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 10(5):115–133
Meng X, Zhang H, Tan W (2000) A hybrid method of GA and BP for short-term economic dispatch of hydrothermal power systems. Math Comput Simul 51(3):341–348
Montana D, Davis L (1989) Training feedforward neural networks using genetic algorithm. In: Proceedings of the 11th international joint conference on artificial intelligence, San Francisco, CA
Murata N, Yoshizawa S, Amari S (1994) Network information criterion—determining the number of hidden units for an artificial neural network model. IEEE Trans Neural Netw 5(6):865–872
Peter J, Gregory M, Jordan B (1994) An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans Neural Netw 5(2):54–65
Ren Z, San Z (2007) Improvement of real-valued genetic algorithm and performance study. Acta Electronica Sinica 35(2):269–274
Roy A, Kim L, Mukhopaduyay S (1993) A polynomial time algorithm for the construction and training of a class of multilayer perceptrons. Neural Netw 6(4):535–545
Sietsma J, Dow R (1991) Creating artificial neural networks that generalize. Neural Netw 4(1):67–79
Sutton R (1986) Two problems with backpropagation and other steepest-descent learning procedures for networks. In: Proceedings of 8th annual conference of the cognitive science society, Hillsdale, NJ
Whitley D (2001) An overview of evolutionary algorithm: practical Issues and common pitfalls. Inf Softw Technol 43(14):817–831
Whitley D, Starkweather T, Bogart C (1990) Genetic algorithms and neural networks: optimizing connection and connectivity. Parallel Comput 14(3):347–361
Xie J (1997) A brief review on evolutionary computation. Control Decis 12(1):1–7
Yao X (1993) A review of evolutionary artificial neural networks. Int J Intell Syst 4(3):203–222
Yao X (1999) Evolving artificial neural networks. Proc IEEE 87:1423–1447
Yao X, Liu Y, Darwen P (1997) A new evolutionary system for evolving artificial neural networks. IEEE Trans Neural Netw 8(3):694–713
Yao W, Wan Q, Chen Z, Wang J (2004) The researching overview of evolutionary neural networks. Comput Sci 31(3):125–129
Yen G (2006) Multi-objective evolutionary algorithm for radial basis function neural network design. Stud Comput Intell 16:221–239
Zheng Z, Zheng S (2002) Study on a mutation operator in evolving neural networks. J Softw 13(4):726–731