Neural Process Lett
DOI 10.1007/s11063-015-9480-z

# An Improved Neural Network with Random Weights Using Backtracking Search Algorithm

**Bingqing Wang[1] · Lijin Wang[1,2] · Yilong Yin[1,3] · Yunlong Xu[1] · Wenting Zhao[1] · Yuchun Tang[4]**



**Abstract** This paper proposes a hybrid algorithm by combining backtracking search algorithm (BSA) and a neural network with random weights (NNRWs), called BSA-NNRWs-N. BSA is utilized to optimize the hidden layer parameters of the single layer feed-forward network (SLFN) and NNRWs is used to derive the output layer weights. In addition, to avoid over-fitting on the validation set, a new cost function is proposed to replace the root mean square error (RMSE). In the new cost function, a constraint is added by considering RMSE on both training and validation sets. Experiments on classification and regression data sets show promising performance of the proposed BSA-NNRWs-N.

✉ Yilong Yin
ylyin@sdu.edu.cn

Bingqing Wang
wangbqing@qq.com

Lijin Wang
lijinwang@fafu.edu.cn

Yunlong Xu
xuyl0104@gmail.com

Wenting Zhao
inweriok@qq.com

Yuchun Tang
tyc@sdu.edu.cn

1 School of Computer Science and Technology, Shandong University, Jinan 250101, China

2 College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou 350002, China

3 School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan 250014, China

4 Research Center for Sectional and Imaging Anatomy, Shandong University School of Medicine, Jinan 250012, China

Published online: 15 December 2015

**Keywords** Neural network · Random weights · Backtracking search optimization algorithm · Cost function

## 1 Introduction

Artificial neural networks (ANNs), as a key component of computational intelligence techniques, play an important role in machine learning and cognitive science by approximating complex nonlinear mappings directly from input samples and producing learner models [1,2]. Feed-forward neural networks with gradient-descent based learning algorithms e.g., back-propagation are usually used to solve these problems because of their universal approx-

The comprehensive experiments in comparison with other algorithms on classification and regression data sets demonstrate that the proposed cost function brings a significant advantage over the traditional one, and BSA can enhance the performance of classification and regression. Additionally, BSA-NNRWs-N is applied to retinal segmentation, and shows its advantage.

The remainder of this paper is organized as follows. Section 2 is the preliminaries including neural networks with random weights and backtracking search algorithm. Section 3 introduces the proposed hybrid algorithm using backtracking search optimization algorithm optimizing neural networks with random weights. Section 4 reports the experimental results. Section 5 discusses the new introduced coefficient $\alpha$ and hidden layer parameters. Section 6 concludes this paper.

## 2 Preliminaries

### 2.1 Neural Networks with Random Weights

NNRWs was proposed in [7], where the parameters $w_j$ and $b_j$ of the hidden layer are considered as random variables and selected randomly and independently in advance, and $\beta_j$ the weight vector between the $j$th hidden node and the output layer nodes is estimated by calculating the Moore–Penrose (MP) generalized inverse. Pao et al. introduced a similar idea, and termed the learner model with random weights as random vector function-link (RVFL) nets [11]. Moreover, more details about theoretical investigations of RVFL can be found in [12].

Suppose we use $N$ distinct samples $\{(x_i, y_i)\}^N$ to train a SLFN model whose number of hidden nodes is $L$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T \in R^d$ and $y_i = (y_{i1}, y_{i2}, \ldots, y_{im})^T \in R^m$. $d$ is the dimension of the samples and $m$ is the number of classes of the training data. In fact, SLFN can be mathematically expressed as:

$$o_i = \sum_{j=1}^{L} \beta_j f(w_j, b_j, x_i) \qquad i = 1, 2, \ldots, N \tag{1}$$

where, $o_i$ is the output of the $i$th sample, $w_j \in R^d$ is the weight vector of the $j$th hidden node with the input layer nodes and $b_j \in R$ is the $j$th hidden node bias ($j = 1, 2, \ldots, L$). $\beta_j \in R^m$ is the weight vector between $j$th hidden node and the output layer nodes. $f(w_j, b_j, x_i)$ is the output of the $j$th hidden node with respect to the $i$th sample. The method to find good weights of SLFN is to minimize the cost function described as follow:

$$err = \sqrt{\sum_{i=1}^{N} \|o_i - t_i\|^2} \tag{2}$$

To NNRWs, the hidden layer parameters can be generated randomly. Hence, the model can be expressed as a linear system as follow:

$$H\beta = T \tag{3}$$

where $H = \{h_{ij}\}(i = 1, 2, \ldots, N \ \ and \ \ j = 1, 2, \ldots, L)$ is the output matrix of the hidden layer and $h_{ij} = f(w_j, b_j, x_i)$ is the output of the $j$th hidden layer node as formerly described. $\beta = \{\beta_1, \beta_2, \ldots, \beta_L\}^T \in R^L$ is the output weights vector and

$T = \{t_1, t_2, \ldots, t_m\}^T \in R^m$ is the desired output. Through the linear system, $\beta$ can be computed as follow:

$$\hat{\beta} = H^{\dagger} T \tag{4}$$

where $H^{\dagger}$ is the Moore–Penrose (MP) generalized inverse of the matrix $H$.

### 2.2 BSA

Backtracking search optimization algorithm is proposed by Pinar Civicioglu in [13] which is a new EA to solve optimizing problems. One obvious property is its few manual setting parameters, just needing one, compared with the traditional EAs which eases the burden of the researchers by avoiding assembling different parameter's value and improves the algorithm's generalization ability. Another advantage of BSA is its preserve of the historical population which provides a good evolution direction. So, BSA shows a promising result with the compared algorithms naturally. The detailed procedures of BSA are described as below.

Step 1. Initialization

The population is initialized in a region with a uniform distribution U.

$$pop_{i,j} \sim U(low_j, up_j) \tag{5}$$

where $i = 1, 2, \ldots, N$ is the number of samples and $j = 1, 2, \ldots, D$ is the dimension of the data set. $pop_{i,j}$ is the value of the $i$th individual's $j$th dimension and $low_j$ and $up_j$ are the lower and upper boundary of $j$th dimension.

Step 2. Selection-I

This stage of BSA aims to preserve the historical information denoted as $his\_pop$ of the population. At the beginning of the evolution, $his\_pop$ is also uniformly sampled from the searching space.

$$his\_pop_{i,j} \sim U(low_j, up_j) \tag{6}$$

After then, at the $g$th generation, $his\_pop$ is updated by the current population randomly.

$$his\_pop = \begin{cases} pop_g & \text{if } a \leq b \\ his\_pop & \text{if } a > b \end{cases} \tag{7}$$

$a$ and $b$ are random numbers belonging to a uniform distribution $U(0, 1)$. This strategy makes $his\_pop$ not be far away from the current population $pop$. It will be ranked randomly as described in Eq. (8).

$$his\_pop = permuting(his\_pop) \tag{8}$$

The $permuting$ function is used to randomly change the order of the individuals in $his\_pop$.

Step 3. Mutation

BSA's mutation strategy can be defined as follow:

$$M = pop + F \cdot (his\_pop - pop) \tag{9}$$

$F$ is a coefficient of the searching direction matrix $(his\_pop - pop)$. In our experiments, $F$ is calculated by the equation $F = 3 * N(0, 1)$ where $N(0, 1)$ is a normal distribution with the mean being 0 and deviation being 1.

Step 4. Crossover

This stage of BSA is divided into two parts. One part is to create a $N * D$ matrix denoted as *map* which only contains two integer value 0 and 1. This *map* matrix controls which dimension of the individuals will be replaced by the mutated one. The second part is to force at least one dimension of each individual to participate into the crossover procedure.

Step 5. Selection II

The successfully evolved individuals are updated by the new ones with the greedy selection strategy and the failed ones will be maintained unchanged.

The program repeats Step 2 to Step 5 until it reaches the terminal condition.

## 3 BSA-NNRWs-N

### 3.1 New Cost Function

Generally speaking, the testing data set and the validation data set share the same distribution because they are derived from the same data set randomly. Thus, in previous investigations, RMSE based on the validation set is a reasonable cost function.

$$C_v = \sqrt{\frac{\sum_{i=1}^{N} \parallel \sum_{j=1}^{L} \beta_j f(w_j \cdot x_i + b_j) - v_j \parallel_2^2}{m \cdot N_v}} \tag{10}$$

$N_v$ is the size of the validation data set, and $v_j$ is the validation set's $j$th sample. However, using optimization algorithms to select the hidden layer parameters makes the model fit validation set step by step because of the greedy strategy it uses. Also, SLFN model is a discriminative model and a high accuracy on validation set may accompany with a low accuracy on testing data set. In the above two cases, it is easy to result in over-fitting on validation data set. Therefore, to alleviate this problem, a constraint term is added to the loss function.

$$C = C_v + \alpha \cdot abs(C_t - C_v) \tag{11}$$

*abs* is an absolute value function and $\alpha$ is a coefficient which will be discussed in Section 5.2. $C_v$ is the RMSE on validation data set as described in Eq. (10), $C_t$ is the RMSE on training data set and $C$ is the objective function.

The first item in Eq. (11) shows that the correct classification or predication of the validation data set is contributed to the performance on the testing set. In the second item, the constraint between the training set and validation set is reflected on the coefficient $\alpha$. In this case, the constraint makes the model hard to fawn on validation data set. On one hand, over-fitting on the training set or on the validation set will amplify the cost function, resulting in abdicable model. On the other hand, if the model is fitted well both on the training and validation data sets, then the constraint item will not be amplified, resulting in a acceptable model.

### 3.2 Framework of BSA-NNRWs-N

BSA-NNRWs-N algorithm uses backtracking search algorithm to select the input layer parameters and uses NNRWs deriving the output layer parameters. The detail steps are described in follow.

Firstly, this paper uses sigmoid hidden functions. The hidden layer weights $w_{i,j}$ and bias $b_i$ are extracted randomly from an uniform distribution over the interval $[-1, +1]$. More discussions about the different initial interval can be found in Section 5.2. In this case, we initialize the population by randomly generated individuals, $X_i$ =

$\{w_{11}, w_{12}, \ldots, w_{1L}, \ldots, w_{m1}, \ldots, w_{mk}, b_1, b_2, \ldots, b_L\}$. After initialization, the mutation, crossover strategy are processed as done in BSA.

Secondly, the output layer parameters are calculated using Eq. (4). The fitness $C$ (the output error of the cost function is denoted as fitness here to be consist with the name used in optimization algorithms) is used to evaluate the model and selects out the best one.

Thirdly, the selection-II part is separated into two stages. The first stage is to update all individuals and the second stage is to update the global best individual. In the first stage, we need to compare the fitness of each individual with the mutant ones. If the fitness becomes smaller, the individual will be replaced by the mutant one. In the second stage, if the mutant individual's fitness is better than the global best one, the global best one will be replaced. Even though the mutant one has a slightly bigger fitness value, the global best one will still be replaced if its norm of the output layer's weights is bigger than the mutant one because the smaller the norm of the output layer is, the higher generation ability the algorithm gets [15]. In other conditions, the global best will remain unchanged. This can be seen as follows:

$$X_{best,g+1} = \begin{cases} X_{i,g+1} & \text{if } C(X_{best,g}) - C(X_{i,g+1}) > \varepsilon \cdot C(X_{best,g}) \\ X_{i,g+1} & \text{if } C(X_{best,g}) - C(X_{i,g+1}) < \varepsilon \cdot C(X_{best,g}) \,\&\&\, Con \\ X_{best,g} & \text{else} \end{cases} \tag{12}$$

$$Con = \parallel \beta_{X_{i,g+1}} \parallel < \parallel \beta_{X_{i,g}} \parallel \tag{13}$$

where $X_{best,g+1}$ and $X_{best,g}$ are the best individuals in the $(g+1)$th and $g$th generation, respectively. $C(X_{i,g+1})$ stands for the cost of $X_{i,g+1}$ which is the $i$th individual in the $(g+1)$th generation. $\parallel \beta_{X_{i,g+1}} \parallel$ is the norm of the output weights of individual $X_i$ in the $(g+1)$th generation. $\varepsilon$ is set to be 0.02.

## 4 Experimental Verifications

In this section, to verify the performance of BSA-NNRWs-N, we make a comparison with NNRWs-1 [16] and NNRWs-2 [17]. Moreover, we divide the experiments into three parts. The first part describes the experiments on classification problems compared with NNRWs, NNRWs-1, NNRWs-1-N, BSA-NNRWs and NNRWs-2. NNRWs-1 and BSA-NNRWs use RMSE on validation set as the cost function, while NNRWs-1-N and BSA-NNRWs-N use the new cost function. The second part shows experiments on regression problems. The third part applies our proposed algorithm on retinal segmentation application. Except for NNRWs-2, each algorithm is performed 50 times run for each problem, and the mean result is recorded in different tables. The population size of NNRWs-1, NNRWs-1-N, BSA-NNRWs and BSA-NNRWs-N for classification problems and retinal segmentation application is 20, while it is set to be 80 for regression problems. For classification and regression problems, the number of hidden layer node is 4 and 5 respectively, while it is 50 for retinal segmentation application. Furthermore, the iteration number of the four algorithms is 200. Additionally, to show the generalization ability of the proposed new cost function and BSA-NNRWs-N, we introduce a new criterion denoted as $Rat$ which is the ratio of the accuracy on testing set and the accuracy on validation set in solving the classification problems. But in solving regression problems, $Rat$ is the ratio of RMSE on validation set and the RMSE on testing set. This new criterion is based on the assumption that the validation set and the testing set have the same distribution. This is true in most cases because they are often generated from the same data set. If the accuracy on validation set and testing set is different widely, this is a bad model and $Rat$ can represent this.

### 4.1 Classification Data Sets

In this section, BSA-NNRWs-N is used to solve classification problems, and the data sets information is listed in Table 1. The results are given in Table 2, where $Train\_acc$ and $Test\_acc$ stand for the training accuracy and the testing accuracy, respectively, and $Dev$ denotes by the variance of testing set.

From Table 2, we can see that the improved neural networks with random weights perform much better than NNRWs on four classification data sets. In particular, the improved NNRWs achieves a great improvement in testing accuracy. For example, for Diabetes data set, BSA-NNRWs and BSA-NNRWs-N obtain 76.16 and 77.54 %, respectively. NNRWs-1-N outperforms NNRWs-1 in terms of the testing accuracy and $Rat$ in all classification data sets.

**Table 1** The classification data sets information

| Names | Attributes | Classes | Number of the observation | | |
|---|---|---|---|---|---|
| | | | Training | Testing | Validation |
| Diabetes | 8 | 2 | 500 | 134 | 134 |
| Iris | 4 | 3 | 100 | 25 | 25 |
| Disease | 13 | 2 | 100 | 85 | 85 |
| Wine | 13 | 3 | 100 | 39 | 39 |

**Table 2** The classification data sets information

| Datasets | Algorithm | Train_acc (%) | Test_acc (%) | $Dev$ | $Rat$ (%) |
|---|---|---|---|---|---|
| Diabetes | NNRWs | 70.33 | 70.40 | 0.0540 | – |
| | BSA-NNRWs-N | 78.26 | 77.54 | 0.0380 | 0.9551 |
| | BSA-NNRWs | 75.46 | 76.10 | 0.0425 | 0.9209 |
| | NNRWs-1-N | 78.22 | 77.31 | 0.0540 | 0.9531 |
| | NNRWs-1 | 75.10 | 76.16 | 0.0451 | 0.9186 |
| Iris | NNRWs | 83.80 | 83.20 | 0.0800 | – |
| | BSA-NNRWs-N | 98.16 | 95.12 | 0.0437 | 0.9520 |
| | BSA-NNRWs | 91.76 | 92.08 | 0.0557 | 0.9215 |
| | NNRWs-1-N | 97.84 | 94.56 | 0.0463 | 0.9471 |
| | NNRWs-1 | 91.08 | 91.28 | 0.0570 | 0.9128 |
| Disease | NNRWs | 73.82 | 72.09 | 0.0647 | – |
| | BSA-NNRWs-N | 87.48 | 81.69 | 0.0418 | 0.9108 |
| | BSA-NNRWs | 81.62 | 79.51 | 0.0381 | 0.8725 |
| | NNRWs-1-N | 88.20 | 82.78 | 0.0376 | 0.9164 |
| | NNRWs-1 | 81.88 | 81.74 | 0.0401 | 0.8862 |
| Wine | NNRWs | 79.98 | 76.77 | 0.1092 | – |
| | BSA-NNRWs-N | 99.50 | 95.59 | 0.0328 | 0.9559 |
| | BSA-NNRWs | 92.44 | 92.26 | 0.0441 | 0.9226 |
| | NNRWs-1-N | 99.94 | 95.74 | 0.0318 | 0.9574 |
| | NNRWs-1 | 92.18 | 92.10 | 0.0445 | 0.9210 |

**Table 3** The classification data sets information used to compare with NNRWs-2

| Names | Attributes | Classes | Number of the observation | | |
|---|---|---|---|---|---|
| | | | Training | Testing | Validation |
| Diabetes | 8 | 2 | 252 | 258 | 258 |
| Satellite image | 39 | 7 | 4435 | 1000 | 1000 |
| Image segmentation | 19 | 7 | 1500 | 405 | 405 |

**Table 4** The experiment on classification problem with NNRWs-2

| Datasets | Algoritm | Train_acc (%) | Test_acc (%) | *Dev* | Norm | Condition |
|---|---|---|---|---|---|---|
| Diabetes | BSA-NNRWs-N | 79.81 | 77.02 | 0.0182 | 39.4616 | 438.4069 |
| | NNRWs-2 | 77.70 | 76.40 | 0.0241 | 72.4941 | 1.0556e+3 |
| Satellite Image | BSA-NNRWs-N | 88.53 | 87.30 | 0.0116 | 9.1162 | 438.3238 |
| | NNRWs-2 | 88.03 | 87.30 | 0.0115 | 14.7202 | 630.1026 |
| Image segmentation | BSA-NNRWs-N | 95.82 | 94.09 | 0.0138 | 84.8649 | 8.726e+3 |
| | NNRWs-2 | 95.33 | 94.62 | 0.0123 | 94.0883 | 5.113e+3 |

The same conclusion can also be drawn on BSA-NNRWs-N. Additionally, BSA-NNRWs-N is overall equal to NNRWs-1-N. BSA-NNRWs-N archives the best performance on Diabetes and Iris sets, while NNRWs-1-N obtains the highest testing accuracy and generalization ability on Disease and Wine.

Additionally, BSA-NNRWs-N is compared with NNRWs-2 [17]. In our experiments, we use data sets adopted in [17], which are listed in Table 3. Moreover, to compare fairly, the two criteria, $Norm$ and $Condition$ which are used in [17], are employed in this section to show the performance of BSA-NNRWs-N. The compared results are shown in Table 4, where the results of NNRWs-2 are taken directly from [17].

On Diabetes data set, our algorithm has higher accuracy, more stability and better generation ability than NNRWs-2. On Satellite Image data set, BSA-NNRWs-N achieves the same accuracy with NNRWs-2 and gets a similar $Dev$ showing the same ability of stability. But BSA-NNRWs-N has the smaller $Norm$ and $Condition$ number which means better generalization ability than NNRWs-2. So, BSA-NNRWs-N still wins NNRWs-2 on Satellite Image data set. On Image segment data set, BSA-NNRWs-N loses to NNRWs-2 on accuracy, stability. It has a smaller Norm number even through its $Condition$ value is larger. So we can't say one has better generalization than the other.

### 4.2 Regression Data Sets

In this section, we verify BSA-NNRWs-N on regression sets, which are listed in Table 5. In addition, the experimental results are listed in Table 6.

Form Table 6 we can see that, the algorithms improved the neural networks with random weights show a great improvement on six data sets compared with NNRWs.

The algorithms based on the new proposed cost function get lower error, more stability, better generalization ability than the algorithms based on traditional cost function on Autompg, AutoPrice, CPU, Servo data sets.

**Table 5** The regression data sets information

| Names | Attributes | | Number of the observation | | |
|---|---|---|---|---|---|
| | Continue | Nominal | Training | Testing | Validation |
| Autompg | 8 | 0 | 220 | 86 | 86 |
| Autoprice | 14 | 1 | 80 | 40 | 39 |
| Boston House | 13 | 0 | 250 | 128 | 128 |
| Breast cancer | 32 | 0 | 100 | 47 | 47 |
| CPU | 6 | 0 | 100 | 54 | 54 |
| Servo | 0 | 4 | 80 | 43 | 43 |

**Table 6** The experiments on regression problem

| Datasets | Algorithm | RMSE-train | RMSE-test | *Dev* | *Rat* |
|---|---|---|---|---|---|
| Autompg | NNRWs | 0.1106 | 0.1108 | 0.0173 | – |
| | BSA-NNRWs-N | 0.0768 | 0.0769 | 0.0082 | 0.9471 |
| | BSA-NNRWs | 0.0800 | 0.0778 | 0.0083 | 0.9070 |
| | NNRWs-1-N | 0.0779 | 0.0776 | 0.0086 | 0.9595 |
| | NNRWs-1 | 0.0805 | 0.0788 | 0.0096 | 0.9185 |
| Autoprice | NNRWs | 0.1144 | 0.1250 | 0.0272 | – |
| | BSA-NNRWs-N | 0.0721 | 0.0888 | 0.0149 | 0.7381 |
| | BSA-NNRWs | 0.0904 | 0.0994 | 0.0219 | 0.5721 |
| | NNRWs-1-N | 0.0714 | 0.0897 | 0.0157 | 0.7109 |
| | NNRWs-1 | 0.0900 | 0.0975 | 0.0204 | 0.5766 |
| Boston House | NNRWs | 0.1599 | 0.1689 | 0.0218 | – |
| | BSA-NNRWs-N | 0.1054 | 0.1132 | 0.0109 | 0.8929 |
| | BSA-NNRWs | 0.1070 | 0.1126 | 0.0117 | 0.8694 |
| | NNRWs-1-N | 0.1062 | 0.1138 | 0.0109 | 0.9011 |
| | NNRWs-1 | 0.1060 | 0.1123 | 0.0107 | 128 |
| Breast Cancer | NNRWs | 0.2559 | 0.2787 | 0.0225 | – |
| | BSA-NNRWs-N | 0.2429 | 0.2714 | 0.0234 | 0.8447 |
| | BSA-NNRWs | 0.2532 | 0.2737 | 0.0219 | 0.8043 |
| | NNRWs-1-N | 0.2395 | 0.2710 | 0.0224 | 0.8294 |
| | NNRWs-1 | 0.2517 | 0.2741 | 0.0209 | 0.7904 |
| CPU | NNRWs | 0.0472 | 0.0601 | 0.0283 | – |
| | BSA-NNRWs-N | 0.0080 | 0.0144 | 0.0099 | 0.4957 |
| | BSA-NNRWs | 0.0100 | 0.0175 | 0.0158 | 0.3646 |
| | NNRWs-1-N | 0.0082 | 0.0144 | 0.0110 | 0.5186 |
| | NNRWs-1 | 0.0106 | 0.0176 | 0.0126 | 0.3859 |
| Servo | NNRWs | 0.1577 | 0.1664 | 0.0217 | – |
| | BSA-NNRWs-N | 0.1182 | 0.1268 | 0.0234 | 0.8774 |
| | BSA-NNRWs | 0.1218 | 0.1294 | 0.0208 | 0.8338 |
| | NNRWs-1-N | 0.1183 | 0.1278 | 0.0225 | 0.8838 |
| | NNRWs-1 | 0.1231 | 0.1314 | 0.0227 | 0.8303 |

On Boston House data set, the algorithms based on the traditional cost function have a lower error than the algorithms based on the new proposed one. But NNRWs-1 has similar $Dev$ with NNRWs-1-N and BSA-NNRWs loses to BSA-NNRWs-N in this criterion. The algorithms based on the new cost function have better generalization ability than the algorithms based on the traditional cost function. The difference between the new proposed cost function and the traditional one on testing error is so small but on $Rat$ is big enough to support the better generalization conclusion of the algorithms based on the new proposed cost function.

On Breast Cancer data set, the algorithms based on the new cost function loses to the algorithm based on the traditional cost function by $Dev$ criterion, but they have a lower error and better generalization ability. So, in solving the regression problems, the new proposed cost function also shows a better performance than the traditional one.

On these six data sets, BSA-NNRWs-N wins four times and loses one time than NNRWs-1-N by testing error and wins three times loses two times on $Dev$ criterion. It wins two times and loses four times by generalization ability compared with NNRWs-1-N. In the whole, BSA-NNRWs-N has a better advantage than NNRWs-1-N on regression problems.

### 4.3 Application on Retinal Segmentation

The retinal vessel segmentation experiment is conducted on two data sets, DRIVE which has two data sets (training data set and test data set) and each of them has 20 raw images and STARE which has 20 raw retinal vessel segment images. On DRIVE data set, 40,000 samples are randomly selected from each image in the training data set. So, the whole training data set contains 800,000 samples. On the STARE data set, we adopt leave-one-out strategy because of the lack of the training data set and each image also contributes 40,000 randomly selected samples. So the training data set on STARE contains 760,000 samples. On the DRIVE data set, every sample has 19 dimensions and on the STARE data set, every sample has 18 dimensions. The feature vectors in both data sets are extracted from green channel because of the high contrast. We use accuracy ($ACC$), sensitivity ($SEN$), specificity ($SPE$) to evaluate our method. $TP$ and $TN$ show the correctly detected pixels of the blood vessel and background respectively. $FP$ and $FN$ show the false pixels of the blood vessel and background respectively. So the above criteria can be defined as:

$$SEN = \frac{TP}{TP + FN} \tag{14}$$

$$SPE = \frac{TN}{TN + FP} \tag{15}$$

$$SEN = \frac{TP + TN}{TP + FP + TN + FN} \tag{16}$$

The result can be seen in Table 7. $SEN$ means the ratio of the vessel pixels being correctly detected and $SPE$ means the ratio of the background pixels being correctly detected. On the DRIVE data set and STARE data set, BSA-NNRWs-N achieves higher accuracy than NNRWs. On the DRIVE data set, the difference estimated by the both criteria $SEN$ and $SPE$ of NNRWs and BSA-NNRWs-N is small which means that it is ease to segment the vessel from the background. On the STARE data set, the difference between $SEN$ and $SPE$ is big. The low value of SEN shows the difficult of segmentation task. But BSA-NNRWs-N achieves higher accuracy than NNRWs on $SEN$. BSA-NNRWs-N has more ability to segment the retinal vessel.

**Table 7** retinal vessel segment result

| Data set | Algorithm | ACC (%) | SEN (%) | SPE (%) |
|---|---|---|---|---|
| DRIVE | NNRWs | 92.99 | 93.17 | 92.98 |
| | BSA-NNRWs-N | 93.89 | 90.88 | 94.15 |
| STARE | NNRWs | 95.06 | 77.53 | 97.09 |
| | BSA-NNRWs-N | 95.70 | 83.18 | 96.95 |

## 5 Discussion

### 5.1 Sensitivity Study on the Coefficient $\alpha$

One contribution to this paper is introducing a new cost function with a coefficient $\alpha$. To show the performance influenced by the coefficient $\alpha$, we conduct a sensitivity study on the above classification and regression sets. For each data set, we sample the value of $\alpha$ from 0.1 to 1 with the step 0.1, and the proposed algorithm is performed 50 times. Table 8 lists the average classification accuracy in the case of each $\alpha$, while Table 9 shows the average prediction error under the condition of each $\alpha$.

From Tables 8 and 9, we can see that the performance obtained by the proposed algorithm for each data set is overall similar in the case of different $\alpha$. This shows the robustness of the new cost function. Nevertheless, we rank the average performance on each data set to select the trade-off one, seen as Tables 10 and 11. It can be observed that the larger $\alpha$, such as 0.7, 0.8 and 0.9, is overall in favor of improving the classification accuracy, while the smaller

**Table 8** The average accuracy on classification sets with different $\alpha$

| Data sets | $\alpha$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1 (%) | 0.2 (%) | 0.3 (%) | 0.4 (%) | 0.5 (%) | 0.6 (%) | 0.7 (%) | 0.8 (%) | 0.9 (%) | 1.0 (%) |
| Diabetes | 76.18 | 76.99 | 76.81 | 77.02 | 77.73 | 77.54 | 77.66 | 78.43 | 77.70 | 77.34 |
| Disease | 82.63 | 82.56 | 82.14 | 82.45 | 83.60 | 81.69 | 82.56 | 82.19 | 82.63 | 82.07 |
| Iris | 95.44 | 96.16 | 94.72 | 95.28 | 97.52 | 95.12 | 96.16 | 96.40 | 96.56 | 95.84 |
| Wine | 95.65 | 95.34 | 95.08 | 95.70 | 95.55 | 95.60 | 95.85 | 95.24 | 95.39 | 93.90 |

**Table 9** The average prediction error on regression sets with different $\alpha$

| Data sets | $\alpha$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Autompg | 0.0781 | 0.0768 | 0.0748 | 0.0750 | 0.0759 | 0.0769 | 0.0757 | 0.0762 | 0.0762 | 0.0755 |
| Autoprice | 0.0976 | 0.0897 | 0.0930 | 0.0853 | 0.0898 | 0.0888 | 0.0905 | 0.0913 | 0.0861 | 0.0884 |
| Boston House | 0.1139 | 0.1140 | 0.1132 | 0.1132 | 0.1108 | 0.1132 | 0.1147 | 0.1121 | 0.1122 | 0.1159 |
| Breast Cancer | 0.2797 | 0.2802 | 0.2694 | 0.2764 | 0.2673 | 0.2714 | 0.2710 | 0.2731 | 0.2761 | 0.2694 |
| CPU | 0.0171 | 0.0148 | 0.0152 | 0.0166 | 0.0146 | 0.0144 | 0.0160 | 0.0147 | 0.0166 | 0.0153 |
| Servo | 0.1301 | 0.1256 | 0.1259 | 0.1243 | 0.1211 | 0.1268 | 0.1276 | 0.1304 | 0.1298 | 0.1372 |

**Table 10** The ranking according to the average accuracy on classification sets

| Data sets | $\alpha$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Diabetes | 10 | 8 | 9 | 7 | 2 | 5 | 4 | 1 | 3 | 6 |
| Disease | 2 | 4 | 8 | 6 | 1 | 10 | 4 | 7 | 2 | 9 |
| Iris | 7 | 4 | 10 | 8 | 1 | 9 | 4 | 3 | 2 | 6 |
| Wine | 3 | 7 | 9 | 2 | 5 | 4 | 1 | 8 | 6 | 10 |
| Average rank | 5.50 | 5.75 | 9.00 | 5.75 | 2.25 | 7.00 | 3.25 | 4.75 | 3.25 | 7.75 |
| Final rank | 5 | 6 | 10 | 6 | 1 | 8 | 2 | 4 | 2 | 9 |

**Table 11** The ranking according to the average prediction error on regression sets

| Data sets | $\alpha$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Autompg | 10 | 8 | 1 | 2 | 5 | 9 | 4 | 7 | 6 | 3 |
| Autoprice | 10 | 5 | 9 | 1 | 6 | 4 | 7 | 8 | 2 | 3 |
| Boston house | 7 | 8 | 4 | 5 | 1 | 6 | 9 | 2 | 3 | 10 |
| Breast cancer | 9 | 10 | 2 | 8 | 1 | 5 | 4 | 6 | 7 | 3 |
| CPU | 10 | 4 | 5 | 8 | 2 | 1 | 7 | 3 | 9 | 6 |
| Servo | 8 | 3 | 4 | 2 | 1 | 5 | 6 | 9 | 7 | 10 |
| Average rank | 9.00 | 6.33 | 4.17 | 4.33 | 2.67 | 5.00 | 6.17 | 5.83 | 5.67 | 5.83 |
| Final rank | 10 | 9 | 2 | 3 | 1 | 4 | 8 | 6 | 5 | 6 |

**Table 12** The accuracy in different intervals of classification problems

| Data sets | Interval | BSA-NNRWs-N (%) | BSA-NNRWs (%) | NNRWs-1-N (%) | NNRWs-1 (%) |
|---|---|---|---|---|---|
| Diabetes | [−0.5,0.5] | 77.84 | 76.71 | 78.14 | 76.45 |
| | [−1,1] | 77.54 | 76.10 | 77.31 | 76.16 |
| | [−2,2] | 78.16 | 75.55 | 77.96 | 76.01 |
| | [−3,3] | 78.13 | 76.34 | 78.25 | 77.06 |
| Disease | [−0.5,0.5] | 82.23 | 79.33 | 83.37 | 82.31 |
| | [−1,1] | 81.69 | 80.02 | 82.78 | 81.74 |
| | [−2,2] | 81.83 | 79.35 | 83.89 | 81.48 |
| Iris | [−3,3] | 81.59 | 80.55 | 83.21 | 81.13 |
| | [−0.5,0.5] | 96.24 | 91.04 | 94.88 | 91.04 |
| | [−1,1] | 95.12 | 92.08 | 94.56 | 91.28 |
| | [−2,2] | 96.64 | 92.24 | 95.04 | 90.80 |
| | [−3,3] | 96.24 | 90.88 | 94.56 | 90.48 |
| Wine | [−0.5,0.5] | 94.93 | 92.88 | 95.89 | 91.95 |
| | [−1,1] | 95.59 | 92.26 | 95.74 | 92.10 |
| | [−2,2] | 94.82 | 92.88 | 95.84 | 91.13 |
| | [−3,3] | 95.18 | 91.90 | 95.28 | 90.87 |

**Table 13** The rank of accuracy to classification problems

| Data sets | Interval | BSA-NNRWs-N | BSA-NNRWs | NNRWs-1-N | NNRWs-1 | Average |
|---|---|---|---|---|---|---|
| Diabetes | [−0.5,0.5] | 3 | 1 | 2 | 2 | 2 |
| | [−1,1] | 4 | 3 | 4 | 3 | 3.5 |
| | [−2,2] | 1 | 4 | 3 | 4 | 3 |
| | [−3,3] | 2 | 2 | 1 | 1 | 1.5 |
| Disease | [−0.5,0.5] | 1 | 4 | 2 | 1 | 2 |
| | [−1,1] | 3 | 2 | 4 | 82 | 2.75 |
| | [−2,2] | 2 | 3 | 1 | 3 | 2.25 |
| | [−3,3] | 4 | 1 | 3 | 4 | 3 |
| Iris | [−0.5,0.5] | 2 | 3 | 2 | 2 | 2.25 |
| | [−1,1] | 4 | 2 | 3 | 1 | 2.25 |
| | [−2,2] | 1 | 1 | 1 | 3 | 1.5 |
| | [−3,3] | 2 | 4 | 3 | 4 | 3.25 |
| Wine | [−0.5,0.5] | 3 | 1 | 1 | 2 | 1.75 |
| | [−1,1] | 1 | 3 | 3 | 1 | 2 |
| | [−2,2] | 4 | 1 | 2 | 3 | 2.5 |
| | [−3,3] | 2 | 4 | 4 | 4 | 3.5 |

**Table 14** The sum of *Average* column in Table 13

| | [−0.5,0.5] | [−1,1] | [−2,2] | [−3,3] |
|---|---|---|---|---|
| Final rank | 8 | 10.5 | 9.25 | 11.25 |

$\alpha$, like 0.2, 0.3 and 0.4 is the benefit of prediction performance. Eclectically, $\alpha = 0.5$ is reasonable due to the first ranking on both classification data sets and regression data sets.

## 5.2 Analysis of the Hidden Layer Parameters

The selection of the hidden layer parameters is important to the neural networks, but the interval which should be selected to the parameters is not clear. Hence, we test four intervals to explore this problem. Table 12 lists the accuracy on classification data sets of different intervals and different algorithms and the Table 15 shows the prediction error on regression problems. To show the best interval to the parameters obviously, we rank the accuracy of classification problems in Table 13 and the prediction error of regression problems in Table 16. In Table 13, to every algorithm on each data set, we rank the accuracy from 1 to 4. For example, on Diabetes data set, BSA-NNRWs-N has the accuracy 77.84, 77.54, 78.16, 78.13 % in interval [−0.5,0,5], [−1,1], [−2,2], [−3,3]; so the rank is 3, 4, 1, 2. The final column shows the average to one interval on all algorithms. This is the same to regression problem which is shown in Table 16. Although the data shown in Tables 13 and 15 in the final column reflects which data set the interval is suit for, it can't explain which interval is good for all data sets. So, we sum the *Average* column value in Tables 13 and 16 and show it in Tables 14 and 17. In Table 14, we can see that on classification problems the interval [−0.5,0.5] is the best one, but the other interval is just a little worse. In Table 17, which shows the results on regression

**Table 15** The prediction error in different intervals of regression problems

| Data sets | Interval | BSA-NNRWs-N | BSA-NNRWs | NNRWs-1-N | NNRWs-1 |
|---|---|---|---|---|---|
| Autompg | [−0.5,0.5] | 0.0763 | 0.0781 | 0.0773 | 0.0793 |
| | [−1,1] | 0.0769 | 0.0778 | 0.776 | 0.0788 |
| | [−2,2] | 0.0777 | 0.0778 | 0.0772 | 0.0792 |
| | [−3,3] | 0.0781 | 0.0788 | 0.0783 | 0.0792 |
| Autoprice | [−0.5,0.5] | 0.0875 | 0.1017 | 0.0907 | 0.0984 |
| | [−1,1] | 0.0888 | 0.0994 | 0.0897 | 0.0975 |
| | [−2,2] | 0.0827 | 0.0948 | 0.0843 | 0.0935 |
| | [−3,3] | 0.0821 | 0.0949 | 0.0856 | 0.0902 |
| Boston house | [−0.5,0.5] | 0.1121 | 0.1116 | 0.1139 | 0.1095 |
| | [−1,1] | 0.1132 | 0.1126 | 0.1138 | 0.1123 |
| | [−2,2] | 0.1156 | 0.1159 | 0.1165 | 0.1148 |
| | [−3,3] | 0.1113 | 0.1107 | 0.1126 | 0.1107 |
| Breast cancer | [−0.5,0.5] | 0.2716 | 0.2798 | 0.2720 | 0.2736 |
| | [−1,1] | 0.2714 | 0.2737 | 0.2710 | 0.2741 |
| | [−2,2] | 0.2723 | 0.2724 | 0.2710 | 0.2714 |
| | [−3,3] | 0.2744 | 0.2724 | 0.2744 | 0.2748 |
| CPU | [−0.5,0.5] | 0.0172 | 0.0198 | 0.0171 | 0.0184 |
| | [−1,1] | 0.0144 | 0.175 | 0.0144 | 0.0176 |
| | [−2,2] | 0.0150 | 0.0187 | 0.0169 | 0.0202 |
| | [−3,3] | 0.0152 | 0.0176 | 0.0151 | 0,0181 |
| Servo | [−0.5,0.5] | 0.1264 | 0.1284 | 0.1220 | 0.1323 |
| | [−1,1] | 0.1268 | 0.1294 | 0.1278 | 0.1314 |
| | [−2,2] | 0.1371 | 0.1381 | 0.1345 | 0.1417 |
| | [−3,3] | 0.1338 | 0.1395 | 0.1278 | 0.1424 |

problems, the interval [−1,1] is the best one. In general, the interval is the best one for all data sets.

## 6 Conclusion and Future Work

In this paper, the new evolving NNRWs algorithm, BSA-NNRWs-N was proposed. BSA-NNRWs-N utilized BSA to optimize the new cost function where RMSE on training and validating data set were both considered. Comprehensive experiments showed that BSA-NNRWs-N and the new cost function are effective.

In this paper, we made the following contributions: (i) We proposed a new hybrid algorithm solving redundant hidden nodes existing in NNRWs. (ii) We proposed a new cost function avoiding the algorithm being over-fitting on the validation data set. (iii) We proposed a new criterion to evaluate the generation ability.

**Table 16** The rank of prediction error to regression problems

| Data sets | Interval | BSA-NNRWs-N | BSA-NNRWs | NNRWs-1-N | NNRWs-1 | Average |
|---|---|---|---|---|---|---|
| Autompg | [−0.5,0.5] | 1 | 3 | 2 | 4 | 2.5 |
| | [−1,1] | 2 | 1 | 3 | 1 | 1.75 |
| | [−2,2] | 3 | 1 | 1 | 2 | 1.75 |
| | [−3,3] | 4 | 4 | 4 | 2 | 3.5 |
| Autoprice | [−0.5,0.5] | 3 | 4 | 4 | 3 | 3.5 |
| | [−1,1] | 4 | 3 | 3 | 4 | 3.5 |
| | [−2,2] | 2 | 1 | 1 | 2 | 1.5 |
| | [−3,3] | 1 | 2 | 2 | 1 | 1.5 |
| Boston House | [−0.5,0.5] | 2 | 2 | 3 | 1 | 2 |
| | [−1,1] | 3 | 3 | 2 | 3 | 1.75 |
| | [−2,2] | 4 | 4 | 4 | 4 | 4 |
| | [−3,3] | 1 | 1 | 1 | 2 | 1.25 |
| Breast Cancer | [−0.5,0.5] | 2 | 1 | 3 | 2 | 2 |
| | [−1,1] | 1 | 4 | 1 | 3 | 2.25 |
| | [−2,2] | 3 | 2 | 1 | 1 | 1.75 |
| | [−3,3] | 4 | 2 | 4 | 4 | 3.5 |
| CPU | [−0.5,0.5] | 4 | 4 | 4 | 3 | 3.75 |
| | [−1,1] | 1 | 1 | 1 | 1 | 1 |
| | [−2,2] | 2 | 3 | 3 | 4 | 3 |
| | [−3,3] | 3 | 2 | 2 | 2 | 2.25 |
| Servo | [−0.5,0.5] | 1 | 1 | 1 | 2 | 1.25 |
| | [−1,1] | 2 | 2 | 2 | 1 | 1.75 |
| | [−2,2] | 4 | 3 | 4 | 3 | 3.5 |
| | [−3,3] | 3 | 4 | 2 | 4 | 3.25 |

**Table 17** The sum of *Average* column in Table 16

| | [−0.5,0.5] | [−1,1] | [−2,2] | [−3,3] |
|---|---|---|---|---|
| Sum average | 15 | 11 | 15.5 | 15.25 |

There are several interesting works in the future. Firstly, we plan to apply the BSA-NNRWs-N to other classification and regression for further examinations. Secondly, we will try to make an adaptive strategy to the number of hidden nodes.

**Acknowledgments** The authors thank the anonymous reviewers for their very helpful and constructive comments and suggestions. This work was supported by the NSFC Joint Fund with Guandong of China under Key Project U120158, the Shandong Natural Science Funds for Distinguished Young Scholar under Grant No. JQ201316, and the Fundamental Research Funds of Shandong University No. 2014JC028.

## References

1. Kecman V (2001) Learning and soft computing: support vector machines, neural networks, and fuzzy logic models. MIT press, Cambridge

2. Wang L, Xiuju F (2006) Data mining with computational intelligence. Springer, Heidelberg
3. Alhamdoosh M, Wang DH (2014) Fast decorrelated neural network ensembles with random weights. Inf Sci 264:104–117
4. Han M, Fan JC, Wang J (2011) A dynamic feedforward neural network based on gaussian particle swarm optimization and its application for predictive control. IEEE Trans Neural Netw 22:1457–1468
5. Slowik A (2011) Application of an adaptive differential evolution algorithm with multiple trial vectors to artificial neural network training. IEEE Trans Ind Electron 58:3160–3167
6. Song Y, Chen ZQ, Yuan ZZ (2007) New chaotic PSO-based neural network predictive conrol for nonlinear process. IEEE Trans Neural Netw 18:595–601
7. Schmidt W, Kraaijveld M, Duin R (1992) Feedforward neural networks with random weights. In: Proceedings of 11th IAPR international conference on pattern recognition methodology and systems, pp 1–4
8. Cao FL, Tan YP, Cai MM (2014) Sparse algorithms of random weight networks and appllications. Expert Syst Appl 41:2457–2462
9. Cao FL, Ye HL, Wang DH (2015) A probabilistic learning algorithm for robust modeling using neural networks with random weights. Inf Sci 313:62–78
10. Zhao JW, Wang ZH, Cao FL, Wang DH (2015) A local learning algorithm for random weights networks. Knowl-Based Syst 74:159–166
11. Pao YH, Takefji Y (1992) Functional-link net computing. IEEE Comput J 25:76–79
12. Igelnik B, Pao YH (1995) Stochastic choice of basis functions in adaptive function approximation and the functional-link net. IEEE Trans Neural Netw 6:1320–1329
13. Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput 219:8121–8144
14. Chen WN, Zhang J, Lin Y et al (2013) Particle swarm optimization with an aging leader and challengers. IEEE Trans Evol Comput 7:241–258
15. Bartlett PL (1998) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inf Theory 44:525–536
16. Cao J, Lin Z Z, Huang GB (2012) Self-adaptive evolutionary extreme learning machine. Neural Process Lett 36:285–305
17. Han F, Yao HF, Ling QH (2012) An improved extreme learning machine based on particle swarm optimization, Bio-inspired computing and applications. Springer, Heidelberg