Neural Process Lett
DOI 10.1007/s11063-013-9326-5

A Rough RBF Neural Network Based on Weighted Regularized Extreme Learning Machine

Shifei Ding · Gang Ma · Zhongzhi Shi

Abstract Adjusting parameters iteratively is a traditional way of training neural networks, and the Rough RBF Neural Networks (R-RBF-NN) follows the same idea. However, this idea has many disadvantages, for instance, the training accuracy and generalization accuracy etc. So how to change this condition is a hot topic in Academics. On the basis of Extreme Learning Machine (ELM), this paper proposes a Weighted Regularized Extreme Learning Machine (WRELM), taking into account both minimizing structured risk and weighted least-squares principle, to train R-RBF-NN. The traditional iterative training method is replaced by the minimal norm least-squares solution of general linear system. The method proposed in this paper, increasing controllability of the entire learning process and considering the structured risk and empirical risk, can improve the performance of learning and generalization. Experiments show that it can reach a very superior performance in both time and accuracy when WRELM trains the Rough RBF Neural Networks in pattern classification and function regression, especially in pattern classification, which can improve the generalization accuracy more than 3.36 % compared with ELM. Obviously, the performance of the method proposed in this paper is better than the traditional methods.

Keywords R-RBF-NN · ELM · WRELM · Minimal norm least-squares solution

1 Introduction

Rough RBF Neural Networks(R-RBF-NN) was proposed on processing the verbose messages. So R-RBF-NN has both the characters of traditional RBF Neural Networks and the

S. Ding (✉) · G. Ma
School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China
e-mail: dingsf@cumt.edu.cn

S. Ding · G. Ma · Z. Shi
Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Science, Beijing, China
e-mail: magangcumt@gmail.com

Published online: 12 November 2013

powerful capacity to process verbose messages [1]. However, the training speed and the generalization ability are still the problem in both R-RBF-NN and traditional RBF Neural Networks. The common learning methods of Neural Networks are based on the thinking of adjusting parameters iteratively, its biggest weakness is time-consuming, which decreases the time efficiency of training Neural Networks throughout the training process. Some people have provided that the learning ability depends on the number of hidden layer neurons, and it has no relation with the weights of input layer neurons of Single-hidden Layer Feedforward Neural Networks (SLFN). Although this view is illuminating to propose a novel learning algorithm of Neural Networks, it did not cause people's attention adequately. So far, a lot of improved algorithms of Neural Networks are based on the thinking of adjusting parameters iteratively without fundamental penetration. G.-B. Huang proposed a ELM to train SLFN by analysizing and summarizing the thinking of adjusting parameters iteratively [2], the whole processes complete in one operation without iteration, which not only solves the problem of time-consuming of training traditional SLFN which adopted the thinking of adjusting parameters iteratively but also increases the time efficiency of training Neural Networks tremendously.

However, the ELM proposed by G.-B. Huang has its deficiencies as well, for instance, it was based on empirical risk minimization principle, which probably leads to over training of the whole network as well as the thinking of adjusting parameters iteratively, and it can decrease the controllability of the whole network because of completing the training process in one operation. Consider all arguments presented above, this paper proposes an improved algorithm which is called Weighted Regularized Extreme Learning Machine (WRELM) on the basis of SLFN by ELM.

The experimental analysis and test show that using WRELM to train R-RBF-NN can solve the problem of time efficiency from the traditional thinking of adjusting parameters iteratively and can get better accuracy of training and testing compared with ELM. So WRELM proposed in this paper is effective and feasible. This paper is organized as follows. Section 1 introduces why propose WRELM to train R-RBF-NN. Section 2 introduces the mechanism of attribution reduction in rough layer of R-RBF-NN. Section 3 gives a mathematical model for WRELM to train R-RBF-NN. Section 4 presents an algorithm description for WRELM to train R-RBF-NN. Section 5 shows experimental studies for verifying the proposed algorithm in this paper. Section 6 is the conclusion and prospect of the thesis.

2 Determine the Importance Degree of Sample Attributes in Rough Set

2.1 Introduce Relevant Concepts about Rough Set

Definition 1 Information system is the system $S = (U,A)$, and $U = \{u_1, u_2, L, u_n\}$ is a finite nonempty set, which is called universe of discourse or object space, the elements in U are called object or case; $A = \{a_1, a_2, \ldots, a_{|A|}\}$ is a finite nonempty set too, the elements in A are called attributes, for each $a \in A$, there is a mapping $f : U \to a(U)$, and $a(U) = \{a(u)|u \in U\}$, a(U) is called range V of attribute a. Universe of discourse U, attribute set A, range V and mapping f constitute the four factors of information system, therefore information system S can be described as $S = (U,A,V, f)$ [3].

If $A = C \cup D, C \cap D = \Phi$, then the information system (U,A) is called a decision table, and the attributes in C are called conditional attribute, the attributes in D are called decision attribute.

Definition 2 The undistinguishable relation is a binary equivalence IND(P) in information system S = (U,C ∪ D), and P is an arbitrary subset of C ∪ D, P ≠ Φ, IND(P) = {∀a ∈ P|(x,y) ∈ U × U : a(x) = a(y)} [4].

All equivalence class sets from IND(P) are noted as U/IND(P), they form a partition of U, which is called a knowledge of U. U/IND(P) is abbreviated to U/P.

Definition 3 Set R be an equivalence relation in universe of discourse U, and (U,R) is called approximation space.

Definition 4 Set (U,R) be approximation space, and $U/R = \{X_1, X_2, \ldots, X_n\}$. If $P(X_i) = \frac{|X_i|}{|U|}$, the entropy of knowledge U/R is defined as follows [5]:

$$H(U/R) = -\sum_{i=1}^{n} P(X_i)\log P(X_i) \tag{1}$$

Theorem 1 *Set S = (U,A) is a information system, P,Q ∈ A, if IND(Q) ⊆ IND(P), then* $H(U/P) \leq H(U/Q)$ *[6].*

Theorem 1 shows that the knowledge entropy monotonically increases along with the diminishing of information granularity (by the smaller partition).

2.2 Measure the Knowledge Entropy of the Sample Attributes

The definition and calculation formula given by formula (1) are educed from the information theory. Because there is no definition of probability measure in approximation space, the partition formed by an arbitrary equivalence relation in the universe of discourse can be regarded as a random variable defined in δ algebra which is composed of subsets of the universe of discourse. Now, the probability of each set can be defined as the proportion of the cardinal number of this set to the universe of discourse [7], that is $P(X_i) = \frac{|X_i|}{|U|}$.

The numeric range of the above calculated knowledge entropy is $[1, +\infty)$, obviously, it doesn't satisfy the weight. So the other definition about the knowledge entropy can be given by the literature [3].

Definition 5 Set $\Omega = \{X_1, X_2, \ldots, X_n\}$ be a partition in U, and $P(X_i) = \frac{|X_i|}{|U|}$, so the knowledge entropy of Ω is defined as follows:

$$H(\Omega) = \sum_{i=1}^{n} P(X_i) \times P(X_i^c) \tag{2}$$

$H(\Omega)$ is called quantity of information in the literature [3], here it is defined as the other measure method of knowledge entropy, and has the below properties:

(1) $0 \leq H(\Omega) \leq 1 - \frac{1}{|\Omega|}$;
(2) $H(\Omega) = 0$, if and only if $\Omega = \{U\}$;
(3) $H(\Omega) = 1 - \frac{1}{|\Omega|}$, if and only if $|X_i| = |X_j|$, (i,j = 1, 2, . . . , n);
(4) $P(X_i^c) = 1 - P(X_i)$.

The above measure method of knowledge entropy is easy to be calculated, from property (1) we can know: if n = |U|, then $0 \leq H(\Omega) \leq 1 - 1/n$, which better satisfy weight; the property (2) indicates: the weight (the knowledge entropy) is 0 if there is no uncertainty; the property (3) suggests: the weight can approach the maximum when the circumstances of classification are uniform, and at that moment the knowledge used to eliminate uncertainty is more important.

2.3 Calculate the Importance Degree of the Samples

The method to determine the weight of each feature by the ability of sample attributes partition the sample library without the other prior information is advanced at the objectivity. So the computational procedure of the importance degree of the sample attributes based on knowledge entropy is shown as follows:

(1) The pretreatment of the sample data library. In order to satisfy the process requirement of Rough Set, the continuous attribute value need to be discretized, that is to say the continuous range of values are partitioned into some discrete intervals by choosing some right partition points, and the different integer values are used to represent the attribute values of every subinterval.

(2) Calculate the knowledge entropy of sample attributes. Determine the partition of the current attributes to sample library on the discretized sample attributes, and the partition of a_j is:

$$U/IND(a_j) = \left\{ \bigcup_{a_j(x)=v_k} x \mid x \in U, v_k \in v \right\} = (X_1, X_2, \ldots, X_n) \tag{3}$$

According to the Definition 5, the knowledge entropy is:

$$H(a_j) = \sum_{i=1}^{n} P(X_i) \times P(X_i^c) = \sum_{i=1}^{n} P(X_i) \times [1 - P(X_i)] = 1 - \sum_{i=1}^{n} P^2(X_i) \tag{4}$$

(3) Assign the importance degree. Determine how to assign the importance degree according to the knowledge entropy of each sample attribute, namely the knowledge entropy is normalized. The formula of calculating importance degree is as follows:

$$w_j = H(a_j) / \sum H(a_j) \tag{5}$$

3 A General Linear System Model of R-RBF-NN Trained by WRELM

3.1 Moore–Penrose generalized Inverse and Minimum Norm Least-Squares Solution of a General Linear System

Generalized inverse matrix is the generalization of the common inverse matrix. At first, let the style of linear equations be:

$$Ax = y \tag{6}$$

When A is an n order matrix and det $A \neq 0$, the solution of linear equations (6) is existence and uniqueness, which can be described as follows:

$$x = A^{-1}y \tag{7}$$

However, the matrix A generally is a singular matrix or any $m \times n$ matrix ($m \neq n$) in practical problems, obviously the common inverse A^{-1} is not existing. To solve this problem, a matrix G which has a similar feature with common inverse must be brought in, which makes the solution of linear equations can be described as a compact style which is similar to formula (7), that is:

$$x = Gy \tag{8}$$

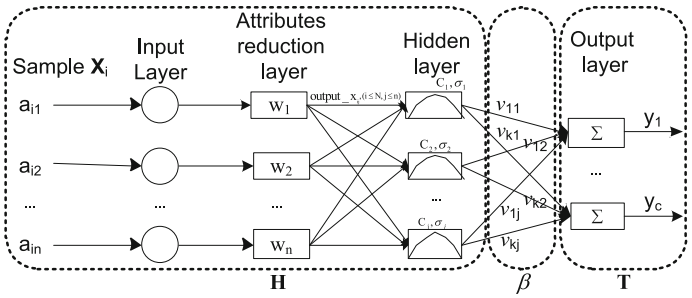

Fig. 1 Linear structure of R-RBF-NN

Definition 6 Let $A \in R^{m\times n}$, if there is a matrix $G(n \times m)$, and it satisfies the following conditions:

$$\begin{aligned} AGA &= A; \\ GAG &= G; \\ (AG)^T &= AG; \\ (GA) &= GA; \end{aligned} \tag{9}$$

So the matrix G is called + inverse, pseudo inverse or Moore–Penrose inverse of matrix A, and it is written as A^+ [8].

Definition 7 $x_0 \in R^n$ is regarded to be a minimum norm least-squares solution of a general linear system $\mathbf{Ax} = \mathbf{y}$ if for any $y \in R^m$ there is:

$$\|x_0\| \le \|x\|, \quad \forall x \in \left\{x: \|Ax - y\| \le \|Az - y\|, \forall z \in R^n\right\} \tag{10}$$

where $\|\bullet\|$ is a norm in Euclidean space [9].

Formula (10) means that a solution x_0 is regarded to be a minimum norm least-squares solution of a general linear system $Ax = y$ if it has the smallest norm among all the least-squares solutions.

3.2 The Model and Solution of General Linear System Model of R-RBF-NN

Let sample set $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N)$ whose size is N and the number of whose attributes is n, where $\mathbf{X}_i = (a_{i1}, a_{i2}, \ldots, a_{in})$, the importance degree array of attributes $\mathbf{W} = [w_1, w_2, \ldots, w_n]$ can be calculate by formula (5) by computing, then using $\mathbf{W}$ to make a attributes reduction for every sample, the reduced sample set is $\mathbf{OutPutX} = (\mathbf{OutPutX}_1, \mathbf{OutPutX}_2, \ldots, \mathbf{OutPutX}_N)$, where $\mathbf{OutPutX}_i = (\text{output\_}x_{i1}, \text{output\_}x_{i2}, \ldots, \text{output\_}x_{in})$ and $\text{output\_}x_{ij} = a_{ij} \times w_j$, $w_j \in \mathbf{W}$, $a_{ij} \in \mathbf{X}_i$, number i means the ith sample, number j means the j-th attribute. The linear structure of R-RBF-NN is shown in Fig. 1.

In Fig. 1, $a_{i1}, a_{i2}, \ldots, a_{in}$ are the condition attributes of input sample $\mathbf{X}_i$, $y_1, y_2, \ldots, y_c$ are the numbers of output class, each neuron of hidden layer have a decision-making model, and their excitation functions are:

$$\begin{aligned} &R_k(\mathbf{OutPutX}_i) = \exp\left(-\|\mathbf{OutPutX}_i - \mathbf{C}_k\|^2/2\sigma_k^2\right) \\ &k = 1, 2, \ldots, j;\ i = 1, 2, \ldots, N;\ \mathbf{OutPutX}_i = \left(\text{output\_}x_{i1}, \text{output\_}x_{i2}, \ldots, \text{output\_}x_{in}\right) \end{aligned} \tag{11}$$

where j is the number of neurons of hidden layer, c is the number of neurons of output layer, $\mathbf{OutPutX}_i$ is the output value of attributes reduction layer of the ith sample $\mathbf{X}_i = (a_{i1}, a_{i2}, \ldots, a_{in})$, which is the input vector of n dimensions of hidden layer, $\mathbf{C}_k$ and σ_k are the center and width of RBF of the kth neuron respectively, $\|\mathbf{OutPutX}_i - \mathbf{C}_k\|$ is the distance between input vector hidden layer $\mathbf{OutPutX}_i$ and the center of RBF $\mathbf{C}_k$.

Finally, the output value of R-RBF-NN is:

$$y_m = \sum_{k=1}^{j} v_{mk} R_k(\mathbf{OutPutX}_i); \quad k = 1, 2, \ldots, j; \; m = 1, 2, \ldots, c; \; i = 1, 2, \ldots, N. \tag{12}$$

where v_{mk} is the weight between the kth neuron of hidden layer and the mth neuron of output layer, R_k is the output value of the kth neuron of hidden layer, y_m is the output value of the mth neuron of output layer.

The R-RBF-NN proposed in this paper is one of SLFN substantially. So, if the centers and width of R-RBF-NN were given, the formula (12) can be written as a compact style similar to $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$($\mathbf{H}$ is the output matrix of hidden layer, $\boldsymbol{\beta}$ is the weight between hidden layer and output layer, $\mathbf{T}$ is the output matrix of neural network). Obviously, the output weight $\boldsymbol{\beta}$ matching with the other parameters of RBF neural network can be worked out easily by solving the equation $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$ if the matrixes $\mathbf{H}$ and $\mathbf{T}$ are given.

However, in the applications, we cannot use $\boldsymbol{\beta} = \mathbf{H}^{-1}\mathbf{T}$ to solve the equation $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$ considering the matrix $\mathbf{H}$ may be a singular matrix or any m × n matrix (m ≠ n). Moore-Penrose inverse of $\mathbf{H}$ is adopted to solve the equation, that is $\boldsymbol{\beta} = \mathbf{H}^{+}\mathbf{T}$. We can obtain $\hat{\boldsymbol{\beta}}$ (the solution of $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$) by that method which has a similar process to get a minimum norm least-squares solution of a general linear system $\mathbf{Ax} = \mathbf{y}$.

Modify it to be a compact style according to the formula (12):

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{13}$$

where:

$$\begin{aligned} \mathbf{H}(\mathbf{OutPutX}, \mathbf{C}, \sigma) &= \mathbf{H}\left(\mathbf{OutPutX}_1, \ldots, \mathbf{OutPutX}_N, \mathbf{C}_1, \ldots, \mathbf{C}_j, \ldots, \sigma_1, \ldots, \sigma_j\right) \\ &= \begin{bmatrix} R\left(\mathbf{OutPutX}_1, \mathbf{C}_1, \sigma_1\right) & \ldots & R\left(\mathbf{OutPutX}_1, \mathbf{C}_j, \sigma_j\right) \\ \vdots & \ldots & \vdots \\ R\left(\mathbf{OutPutX}_N, \mathbf{C}_1, \sigma_1\right) & \ldots & R\left(\mathbf{OutPutX}_N, \mathbf{C}_j, \sigma_j\right) \end{bmatrix}_{N \times j} \end{aligned} \tag{14}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_j^T \end{bmatrix}_{j \times c}; \quad \mathbf{T} = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times c} \tag{15}$$

In the formula (14), the matrix $\mathbf{H}(\mathbf{OutPutX}, \mathbf{C}, \boldsymbol{\sigma})$ means the output matrix of hidden layer of R-RBF-NN, the kth (k = 1, 2, ..., j) column of $\mathbf{H}(\mathbf{OutPutX}, \mathbf{C}, \boldsymbol{\sigma})$ is the output value of N attributes reduction samples ($\mathbf{OutPutX}_1, \mathbf{OutPutX}_2, \ldots, \mathbf{OutPutX}_N$) corresponding to the kth center of hidden layer.

The center parameters of hidden layer $\mathbf{C}$ and σ can be given randomly and have no need to be trained. So, for the any center parameters of hidden layer $\mathbf{C}_k$ and σ_k(k = 1, 2, ..., j) given randomly, training R-RBF-NN can be convert to solve $\hat{\boldsymbol{\beta}}$ the least-squares solution of a general linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$, as follows:

$$\left\|\mathbf{H}\left(\mathbf{C}_1, \ldots, \mathbf{C}_j, \sigma_1, \ldots, \sigma_j\right)\hat{\boldsymbol{\beta}} - \mathbf{T}\right\| = \min_{\beta} \left\|\mathbf{H}\left(\mathbf{C}_1, \ldots, \mathbf{C}_j, \sigma_1, \ldots, \sigma_j\right)\boldsymbol{\beta} - \mathbf{T}\right\| \tag{16}$$

However, in the applications, j $\neq$ N(the number of neurons in hidden layer is not equal to the number of training samples, the number of neurons in hidden layer maybe is smaller than the number of training samples, that is j << N). The accuracy solution $\boldsymbol{\beta}$ of $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$ can't be obtained if $\mathbf{H}$ is a singular matrix or an any m × n matrix (m $\neq$ n), therefore we must use the Moore–Penrose generalized inverse and minimum norm least-squares solution of a general linear system to solve $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$ in order to get $\hat{\boldsymbol{\beta}}$, which is described as follows:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{+}\mathbf{T} \tag{17}$$

In the formula (17), $\mathbf{H}^{+}$ is the Moore–Penrose generalized inverse matrix of $\mathbf{H}$, $\hat{\boldsymbol{\beta}}$ is the weight between the hidden layer and the output layer of trained R-RBF-NN.

3.3 The Mathematical Model of WRELM about R-RBF-NN

The method mentioned in Sect. 3.2, ELM, can change the multiple iterative learning into the simple linear system, which improves time efficiency compared with traditional RBF neural network. However, a lot of experimental results of UCI data set show that ELM has some weaknesses yet.

- ELM adopts least-squares principle to calculate the weight between hidden layer and output layer, but the center parameters and the number of neurons of hidden layer cannot be adjusted, which results in a lower controllability of the entire learning process.
- ELM just use the known empirical data, training sample, to get the weight $\boldsymbol{\beta}$ between hidden layer and output layer by the algorithm of minimizing error, that is to say, ELM just considers the empirical risk, and pays no attention to the structured risk, which maybe result in overfitting.

To solve these above problems, we propose a novel training method based on ELM, Weighted Regularized Extreme Learning Machine (WRELM), which takes into account both minimizing structured risk and weighted least-squares principle, also the error weights of training sample are joined in the model.

The structured risk is defined as the sum of the empirical risk and the confidence interval in statistical learning theory. In machine learning theory, the principle of minimum empirical risk is consistent on the condition that the number of samples approaches infinity, therefore using the empirical risk to assess forecasted risk is cursory when the number of samples is limited, the structured risk is the upper bound of forecasted risk [10].

Therefore, the model with good generalization capability should balance the above two risks and give an optimal balanced point. If WRELM don't have weight values, just only thinking to adjust the ratio above two risks by the parameter γ [11]. No-weighted WRELM mathematic model can be written as follows:

$$\begin{aligned} &\underset{\boldsymbol{\beta}}{argmin}\mathrm{E}(\mathbf{W}) = \underset{\boldsymbol{\beta}}{argmin}\left(\tfrac{1}{2}||\boldsymbol{\beta}||^2 + \tfrac{1}{2}\gamma||\boldsymbol{\varepsilon}||^2\right) \\ &s.t.\sum_{\mathrm{k}=1}^{\mathrm{j}}\boldsymbol{\beta}_{\mathrm{k}}\mathrm{R}_{\mathrm{k}}(\mathbf{OutPutX}_{\mathrm{i}}) - \mathrm{t}_{\mathrm{i}} = \boldsymbol{\varepsilon}_{\mathrm{i}}\,; \quad \mathrm{i} = 1,\ldots,\mathrm{N} \end{aligned} \tag{18}$$

where the sum of squares of errors $||\boldsymbol{\varepsilon}||^2$ represents the empirical risk, $||\boldsymbol{\beta}||^2$ is the structured risk which comes from the principle of maximum contour distance of statistical learning theory [12,13], γ is the ratio coefficients above two risk, and the optimal balanced point can be gotten by using the cross validation to computing γ.

This paper makes a weighted process for the errors of different sample in order to get a steady and anti-jamming model, that is to say, $||\boldsymbol{\varepsilon}||^2$ is expanded into $||\boldsymbol{D\varepsilon}||^2$. Where,

$D = diag(v_1, v_2, \ldots, v_N)$ is the weighted value diagonal matrix of error, so, the mathematics model of WRELM of R-RBF-NN is:

$$\begin{aligned} &\underset{\boldsymbol{\beta}}{argmin}\left(\tfrac{1}{2}||\boldsymbol{\beta}||^2 + \tfrac{1}{2}\gamma||\boldsymbol{D\varepsilon}||^2\right) \\ &s.t. \sum_{\mathrm{k}=1}^{\mathrm{j}} \boldsymbol{\beta}_{\mathrm{k}} \mathrm{R_k}(\mathbf{OutPutX_i}) - \mathrm{t_i} = \boldsymbol{\varepsilon}_{\mathrm{i}}\,; \quad \mathrm{i} = 1, \ldots, \mathrm{N} \end{aligned} \tag{19}$$

The formula (19) is a conditional extremum problem, we can convert the conditional extremum problem into the unconditional extremum problem by Lagrange function [14]:

$$\begin{aligned} \ell(\boldsymbol{\beta}, \boldsymbol{\varepsilon}, \boldsymbol{\alpha}) &= \frac{\gamma}{2}||\boldsymbol{D\varepsilon}||^2 + \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{\mathrm{k}=1}^{\mathrm{j}} \boldsymbol{\alpha}_{\mathrm{k}}(\mathrm{R_k}(\mathbf{OutPutX_i}) - \mathrm{t_i} - \boldsymbol{\varepsilon}_{\mathrm{i}}) \\ &= \frac{\gamma}{2}||\boldsymbol{D\varepsilon}||^2 + \frac{1}{2}||\boldsymbol{\beta}||^2 - \boldsymbol{\alpha}(\mathbf{H\beta} - \mathbf{T} - \boldsymbol{\varepsilon}) \end{aligned} \tag{20}$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_{\mathrm{j}}], \boldsymbol{\alpha}_{\mathrm{k}} \in \mathrm{R}^m(\mathrm{k} = 1, 2, \ldots, \mathrm{j})$ is Lagrange multiplier, i ($\mathrm{i} = 1, 2, \ldots, \mathrm{N}$) is the order of input samples.

Calculate the gradient of above Lagrange function, and let the gradient be 0, as follows:

$$\begin{cases} \frac{\partial \ell}{\partial \boldsymbol{\beta}} \rightarrow \boldsymbol{\beta}^T = \boldsymbol{\alpha}\mathbf{H} & ① \\ \frac{\partial \ell}{\partial \boldsymbol{\varepsilon}} \rightarrow \gamma \boldsymbol{\varepsilon}^T \boldsymbol{D}^2 + \boldsymbol{\alpha} = 0 & ② \\ \frac{\partial \ell}{\partial \boldsymbol{\alpha}} \rightarrow \mathbf{H\beta} - \mathbf{T} - \boldsymbol{\varepsilon} = 0 & ③ \end{cases} \tag{21}$$

Combine formula ③ and ② is:

$$\boldsymbol{\alpha} = -\gamma(\mathbf{H\beta} - \mathbf{T})^T \boldsymbol{D}^2 \tag{22}$$

Bring formula (22) into ① is:

$$\boldsymbol{\beta} = \left(\mathbf{I}/\gamma + \mathbf{H}^T \boldsymbol{D}^2 \mathbf{H}\right)^+ \mathbf{H}^T \boldsymbol{D}^2 \mathbf{T} \tag{23}$$

Formula (23) just has an operation to solve the Moore_Penrose inverse of a $\mathrm{j} \times \mathrm{j}(\mathrm{j} << N)$ matrix, so, the calculation of $\boldsymbol{\beta}$ is very fast.

In the applications, this model is very steady and the training and testing results are rather satisfactory if there are few outliers in the data set. The error weights can be set to be same in order to increase the training speed, at the moment, the matrix $\boldsymbol{D} = diag(v_1, v_2, \ldots, v_{\mathrm{N}})$ is an identical matrix which does not need to be calculated, so the calculating expression of Lagrange multiplier $\boldsymbol{\alpha}$ is:

$$\boldsymbol{\alpha} = -\gamma(\mathbf{H\beta} - \mathbf{T})^T \tag{24}$$

Obviously, the solution $\boldsymbol{\beta}$ of linear function will degrade to be:

$$\boldsymbol{\beta} = \left(\mathbf{I}/\gamma + \mathbf{H}^T \mathbf{H}\right)^+ \mathbf{H}^T \mathbf{T} \tag{25}$$

In the formula (25), it is easy to find that if $\gamma \rightarrow \infty$, then $\mathbf{I}/\gamma \rightarrow 0$, this formula will degrade to be $\boldsymbol{\beta} = \left(\mathbf{H}^T \mathbf{H}\right)^+ \mathbf{H}^T \mathbf{T}$ again, and simplify it further:

$$\boldsymbol{\beta} = \mathbf{H}^+ \mathbf{T} \tag{26}$$

The formula (25) and (23) describe the same problem, and the formula (23) is a more common mathematics training model after R-RBF-NN introducing structured risk, empirical risk and weighted least-squares principle based on ELM.

The formula (25) is still an ideal model as well. In the applications, the steadiness and anti-interfering of this model will decrease if there were some outliers in the data set, that is to say, the robustness will be weakened. So we must add weight to this model in order to increase the robustness, then the error weight of each sample will be different, the matrix $\boldsymbol{D} = diag(v_1, v_2, \cdots, v_N)$ is no longer an identical matrix, and the training model is called WRELM. There are many methods to calculate the weight, and we adopt the method mentioned in documentation [15] in this paper.

$$v_i = \begin{cases} 1 & |\boldsymbol{\varepsilon}_i/\hat{s}| \le c_1 \\ \frac{c_2 - |\boldsymbol{\varepsilon}_i/\hat{s}|}{c_2 - c_1} & c_1 \le |\boldsymbol{\varepsilon}_i/\hat{s}| \le c_2 \\ 10^{-4} & \text{otherwise} \end{cases} \tag{27}$$

In the formula (26), $\boldsymbol{\varepsilon}_i = -\boldsymbol{\alpha}_i/\gamma$ is the sample error of unweighted RELM, $\hat{s}$ is the estimate of standard deviation which can be gotten by $\hat{s} = 1.483 * MAD(\mathbf{OutPutX}_i)$, where MAD is median absolute deviation and 1.483 is an empirical value. According to Gaussian distribution principle, we can recognize that there is no error which is larger than $2.5\hat{s}$, so we set the constant c_1 and c_2 be $c_1 = 2.5$, $c_2 = 3$ normally. Consider all the above arguments, the formula (23) should be used to calculate $\boldsymbol{\beta}$ if there are some outliers in the data set, conversely, the formula (25) should be used to compute $\boldsymbol{\beta}$.

4 The Algorithm Description of WRELM of R-RBF-NN

To sum up, the general linear system model of WRELM of R-RBF-NN has been built by the discussion of Sect. 3, and the detail algorithm procedure of this mathematical model is:

Step 1 Preprocess the input sample. The sample attributes are changed to adapt the data style of Rough set.

Step 2 Calculate the importance degree of samples attributes. Use discretized samples attributes to calculate the knowledge entropy of each attribute by formula (4), and calculate the importance degree by formula (5).

Step 3 Make an attributes reduction on samples by importance degree of samples attributes. Calculate the output value **OutPutX** of attributes reduction layer by the attributes importance degree of sample **X.**

Step 4 Initialize the network parameter. Randomly set the weight, center, width between the attributes reduction layer and the hidden layer.

Step 5 Calculate the output matrix **H**. Use the reduced sample **OutPutX** in step 3 and network parameter in step 4 to calculate the output matrix **H** by the formula (14).

Step 6 Calculate the weight between the hidden layer and the output layer. Use the output matrix **H** in step 5 to calculate the weight matrix $\boldsymbol{\beta}$ between the hidden layer and the output layer of R-RBF-NN trained by unweighted RELM.

Step 7 Using output matrix **H** in step 5 and weight matrix $\boldsymbol{\beta}$ in step 6 to compute Lagrange multipler $\boldsymbol{\alpha}$.

Step 8 Use confirmed γ and $\boldsymbol{\alpha}$ in step 7 to calculate samples error $\boldsymbol{\varepsilon}_i$ of unweighted RELM by $\boldsymbol{\varepsilon}_i = -\boldsymbol{\alpha}_i/\gamma$.

Step 9 Use $\hat{s} = 1.483MAD(\mathbf{OutPutX}_i)$ to calculate the estimate of standard deviation of error $\boldsymbol{\varepsilon}_i$.

Step 10 Calculate $\boldsymbol{D} = diag(v_1, v_2, \ldots, v_N)$. Use $(\boldsymbol{\varepsilon}_i, \hat{s})$ from step 8 and step 9 to calculate the $\boldsymbol{D} = diag(v_1, v_2, \ldots, v_N)$ by the formula (27).

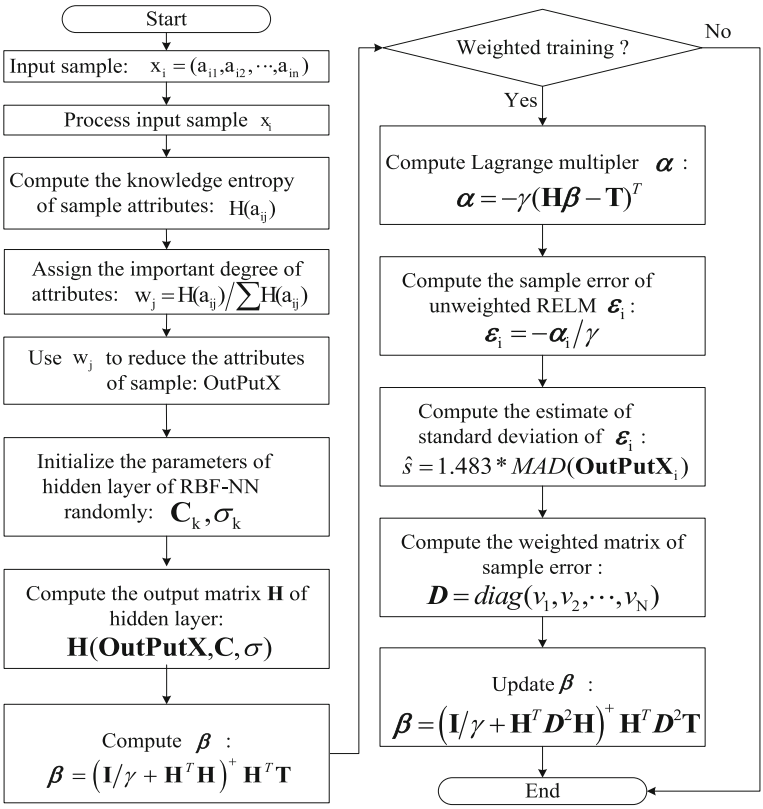

Fig. 2 The procedure of R-RBF-NN trained by WRELM

Step 11 Use formula (23) to calculate the weight $\boldsymbol{\beta}$ between the hidden layer and the output layer of R-RBF-NN trained by WRELM.

The above 11 procedures show a detail training process of R-RBF-NN trained by WRELM. Now, the flow chart of the entire algorithm is as follows:

Obviously, we can recognize that the training process of R-RBF-NN trained by WRELM doesn't have any iteration process, the algorithm changes the complex iteration process into a simple mathematics computing, which increases the processing efficiency of the entire algorithm immensely (Fig. 2).

5 Experimental Results

In order to prove the performance of R-RBF-NN trained by WRELM in this paper, we do two classical experiments of machine learning: pattern classification and function regression. In the two experiments, we respectively analysis ELM training method, WRELM training method and the hidden layer center clustering (HLCC) method that adapts the traditional thinking of adjusting parameters iteratively in both the accuracy and time efficiency. In pattern classification experiment, the number of neurons of hidden layer is 20, 30, 40 respectively when the ELM and WRELM train the R-RBF-NN, while the number of neurons of HLCC increases from 0, and the number of neurons of hidden layer is more than 150. Moreover, the

Table 1 The simulation results of Wdbc data set

	Training accuracy	Testing accuracy	Training time	Testing time
ELM	0.9542	0.9583	0.0781	0.0625
WRELM	0.9754	0.9672	0.5625	0.0781
HLCC	0.9831	0.4648	13.8594	0.1094

Table 2 The simulation results of Wine data set

	Training accuracy	Testing accuracy	Training time	Testing time
ELM	0.9438	0.9326	0.0313	0.0156
WRELM	0.9663	0.9551	0.2031	0.0156
HLCC	0.9836	0.5618	0.6250	0.0625

Table 3 The simulation results of Iris data set

	Training accuracy	Testing accuracy	Training time	Testing time
ELM	0.9867	0.9600	0.0313	0.0156
WRELM	1	0.9733	0.2031	0.0156
HLCC	0.9889	0.3867	0.5000	0.0469

Table 4 The simulation results of Hepatitis data set

	Training accuracy	Testing accuracy	Training time	Testing time
ELM	0.9481	0.7922	0.0937	0.2187
WRELM	0.8961	0.8701	0.6719	0.1250
HLCC	0.9986	0.4805	0.3594	0.0312

value of the ratio coefficients γ presented in Sect. 3.3 distributes in the interval $[2^{-50}, 2^{50}]$, WRELM will choose automatically an optimal γ from the interval, and the simulation results from Tables 1, 2, 3 and 4 derive respectively from an optimal γ of WRELM for different data set. In function regression experiment, the number of neurons of hidden layer is 50, 60, 70 respectively when the ELM and WRELM train the R-RBF-NN, and the number of neurons of HLCC increases from 0, finally the number of neurons of hidden layer is more than 200. The simulation results above two experiments were the average values of 10-times experiments, and done in the same experiment circumstance, that is: hardware system of computer is Intel(R) Core(TM)2 CPU T5300, computer main frequency is 1.73GHz, memery is 1.00GB. Operation system is Microsoft Windows XP Professional. Programming environments is Matlab 7.1.

5.1 Pattern classification

In pattern classification experiment, the data set include Wdbc, Wine, Iris and Hepatitis from UCI data set. Wdbc data set consists of 569 Breast cancer cases which contains 30 attributes and 1 binary classified variable. Wine data set consists of 178 wine samples which contains 13 attributes and 1 ternary classified variable. Iris data set consists of 150 iris samples which

Table 5 The average performance of the three algorithm for all data sets

	Average training accuracy	Average testing accuracy	Average training time	Average testing time
ELM	0.9582	0.9108	0.0586	0.0781
WRELM	0.9595	0.9414	0.4101	0.0586
HLCC	0.9886	0.4735	3.8359	0.0625

contains 4 attributes and 1 ternary classified variable. Hepatitis data set has 155 hepatitis samples which contains 19 attributes and 1 ternary classified variable. The attributes number of above four data sets is different from each other, which involves high, medium and low three different input dimensions.

The four different data sets are done preliminary processing before experiment, that is to say, each data set is grouped according to the classified variable. The data samples have a same class are divided into the same group. In the experiment, the odd-row data samples are used to train and the even-row data samples are used to test which grantee half of the data used to train and half of the data used to test. It can promise the full coverage of all class of training and testing set to reach a good training and testing effect. The experiment results of the different algorithm on the above four data sets are shown in Tables 1, 2 and 3.

From the Table 1 we can see that, the training accuracy of Wdbc data set is very high, but the testing accuracy is undesirability which due to that the HLCC doesn't iteratively train R-RBF-NN according to the default training times and training accuracy until one of them is satisfied and the number of neurons of hidden layer is increasing with iterations. However, the ELM and WRELM is an unsupervised pseudo learning, they both calculate matrix $\mathbf{H}$ of $\mathrm{H}\beta = \mathrm{T}$ by the formula (14) at once, the entire process is unsupervised, which is just only a pure mathematical computation called pseudo learning. That is the reason why the time efficient of training and testing of ELM and WRELM is higher than HLCC method. Although the time efficient of ELM is higher than WRELM, the loss of WRELM in time gains the promotion of the accuracy in training and testing. Therefore this promotion is quite valuable in our applications and can be acceptable.

As shown in Table 2, the number of attributes of Wine data set is less than Wdbc data set by 57 %, the number of samples of Wine data set is less than Wdbc data set by 69 %. So the time efficiency increases significantly, and the training and testing accuracy of WRELM is till highest. Although training accuracy of HLCC is very high, the testing accuracy is till undesirability. The best excited thing is that the time gaps between WRELM and ELM decreases to 0.1718 s in Table 2 from 0.5547 s in Table 1, the testing time decreases to 0s in Table 2 from 0.0156 s in Table 1, and the increasing of training time efficiency of HLCC is obvious. The simulation result in Table 2 shows that the training and testing time efficiency of three algorithms increases observably and WRELM is faster than others.

In Table 3, the simulation results of Iris data set are better than other data sets, especially, the training accuracy of WRELM reaches 1, and the testing accuracy reaches 0.9733. Its training accuracy and testing accuracy are both highest in the three algorithms.

From the simulation results of Hepatitis data set in Table 4, the performance of accuracy and time is low a little compared with the other three data sets. However, the testing accuracy of WRELM is also highest, and testing time is medium level in three algorithms.

From Table 5, it is quite easy to see that the combination property of WRELM is best in all data sets. Only the average efficiency of training time of WRELM are lower than ELM a little, while average testing time of WRELM are more than ELM by 33.28 %, and

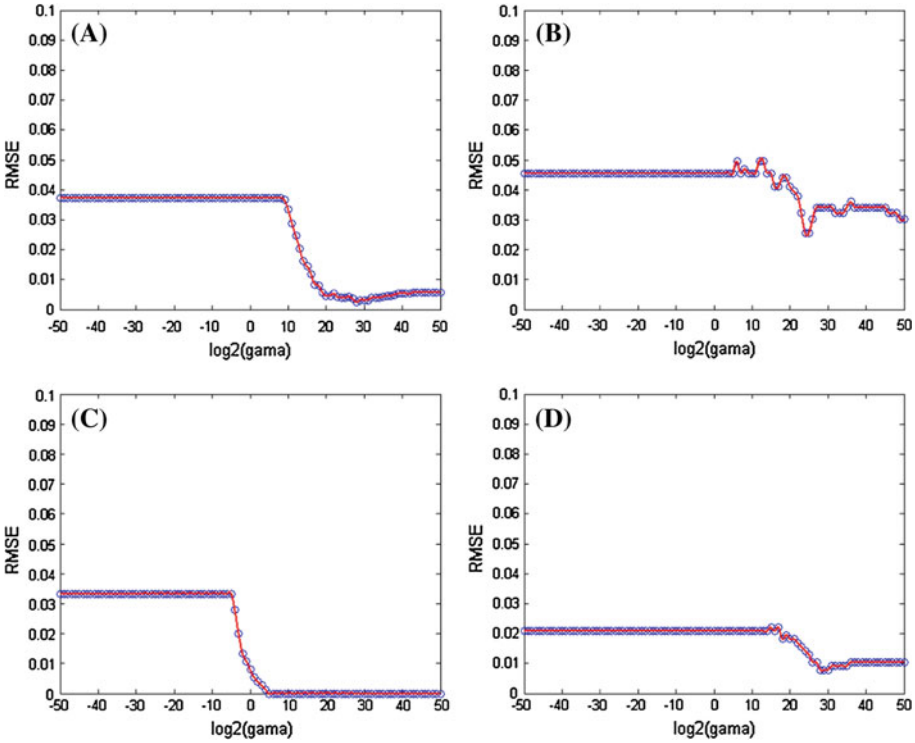

Fig. 3 RMSE curve of WRELM with different γ for different data set. **a** RMSE curve of WRELM with different γ **b** RMSE curve of WRELM with different γ based on Wdbc data set based on Wine data set. **c** RMSE curve of WRELM with different γ. **d** RMSE curve of WRELM with different γ based on Iris data set based on Hepatitis data set

more than HLCC by 6.66 %. In addition, the average testing accuracy are more than ELM by 3.36 %, and more than HLCC by 98.82 %. Why the algorithm we proposed has better generalization ability than ELM and HLCC? That because the algorithm we proposed takes into consideration simultaneously both empirical risk and structured risk, while ELM and HLCC just only consider empirical risk.

Figure 3 could prove that WRELM's performance is increasing with the value of γ growing, and each data set trained by WRELM has an optimal γ, 2^28, 2^24, 2^5, 2^28, respectively based on Wdbc, Wine, Iris, Hepatitis data set, from RMSE curves. However, careful people maybe notice WRELM's performance is declining after the optimal γ, which prove nicely WRELM will degenerate into ELM as $\gamma \to \infty$.

5.2 Function Regression

In function regression experiment, the adopted regression function is nonlinear function $\mathbf{F(x) = 20 + x^2 - 10^*cos(2^*\pi^*x) - 10^*sin(2^*\pi^*x)}$, the number of independent variables of training samples **x** of R-RBF-NN is 200 groups which come from the 200 random numbers set $\{x_1, x_2, \ldots x_{200}\}$ in intervals $[-1.5, 1.5]$, and the function values $\{F_1, F_2, \ldots, F_{200}\}$ suited to **x** are calculated by the formula **F(x)**, the training results of R-RBF-NN are from training data set [**x F**] when training the network. The number of testing samples of R-RBF-NN is 30 groups, the independent variable **x1** of function **F(x1)** is $\{x1_1, x1_2, \ldots x1_{30}\}$ which come

Table 6 The average performance of the three algorithm for function **F(x)**

	Average training accuracy	Average testing accuracy	Average training time	Average testing time
ELM	0.999497	0.999898	0.062500	0.015625
WRELM	0.999569	0.999995	0.200000	0.015625
HLCC	0.998542	0.998386	0.265625	0.031250

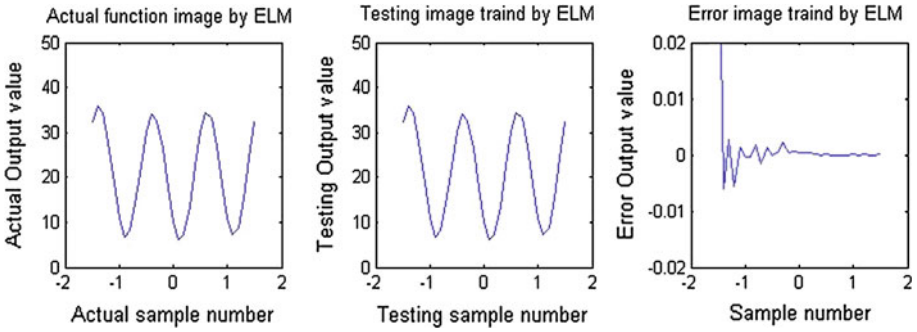

Fig. 4 The performance analyses graphs of ELM

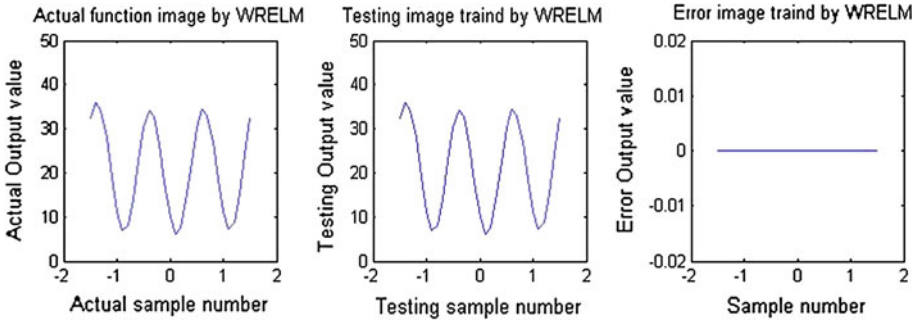

Fig. 5 The performance analyses graphs of WRELM

from intervals $[-1.5, 1.5]$ with the step 0.1, and the function values $\{F_1, F_2, \ldots, F_{200}\}$ suited to **x1** are calculated by the formula **F(x1)**, the testing results of R-RBF-NN are from training data set [**x1 F1**] when training the network. We make a performance analysis for the three different algorithms ELM, WRELM and HLCC in accuracy and time by using the training data set [**x F**] and testing data set [**x1 F1**] which come from the function **F(x)**, the detailed result of analysis are shown in the Table 6.

From Table 6 we can see that the training accuracy and testing accuracy of WRELM are the highest, only the training time increases a little, while the testing time is the shortest which flats with ELM in the three algorithms. Comprehensive study, the average performance of WRELM in the three algorithms is highest, the analytical picture of three algorithms for function regression are shown in Figs. 4, 5 and 6.

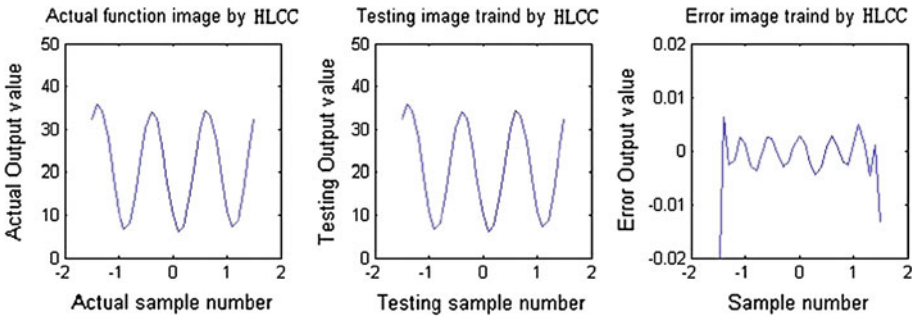

Fig. 6 The performance analyses graphs of HLCC

6 Conclusions

Adopting WRELM to training the R-RBF-NN not only breaks through the thinking of adjusting parameters iteratively but also avoids the disadvantage of bad controllability, bad stability and overtraining, besides, in this paper, we also considered two risks—the structured risk and the empirical risk. Therefore the R-RBF-NN could obtain a high performance learning method. From the above experiments we can see that WRELM has a great high learning and generalization performance in both pattern classification and function regression, it not only has high learning speed but also has high generalization accuracy. The training speed of WRELM is at millisecond level for the general data set, especially the testing accuracy increases more than 50% compared with HLCC, which show the high performance of WRELM. In the rest research, it is planning to bring WRELM into process mass data to improve the time efficiency, for instance, the classification of texts and pictures, and how to improve the controllability of WRELM in neural networks.

Acknowledgments This work is supported by the National Natural Science Foundation of China (No. 61379101), the National Key Basic Research Program of China (No. 2013CB329502), and the Opening Foundation of the Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (No.IIP2010-1).

References

1. Chiang J-H, Ho S-H (2008) A combination of rough-based feature selection and RBF neural network for classification using gene expression data. IEEE Trans Nanobioscience 7(1):91–99
2. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applica-tions. Neurocomputing 70(1–3):489–501
3. Zhang W, Qiu G (2005) Uncertain decision-making based on rough set. Tsinghua University Press, Beijing
4. Pawlak Z, Wong SKM, Ziarko W (1988) Rough sets: probabilistic versus deterministic approach. Int J of Man Mach Stud 29(1):81–95
5. Duntsch I, Gediga G (1998) Uncertainty measures of rough set prediction. Artif Intell 106(1):106–109
6. Chen H, Mao H (2010) A new method of weights allocation to case feature attributes based on rough set theory. J Xi'an Technolog Univ 30(4):402–404
7. Huang D (2003) Means of weights allocationwith multi-factors based on impersonal message entropy. Syst Eng Theory Methodol Appl 12(4):321–324
8. Cheng L, Hu J (2009) Matrix theory. China University of Mining and Technology Press, Xuzhou
9. Huang G-B, Siew C-K (2004) Extreme learning machine: RBF network case. Control Autom Robotics Vis Conf 2:1029–1036
10. Ding S, Qi B, Tan H (2011) An overview on theory and algorithm of support vector machines. J Univ Electron Sci Technol China 40(1):2–10

11. Deng W-Y, Zheng Q-H, Chen L, Xu X-B (2010) Research on extreme learning of neural networks. Chin J Comput 33(2):279–287
12. Deng N-Y, Tian Y-J (2004) A new method of data mining-support vector machine. Science Press, Beijing
13. Zhang X-G (2000) Introduction to statistical learning theory and support vector machines. Acta Autom Sin 26(1):32–42
14. Ding S, Jin F, Zhao X (2013) Modern data analysis and information pattern recognition. Science Press, Beijing
15. Suykens JAK, De Brabanter J, Lukas L, Vandewaile J (2002) Weighted least squares support vector machines: robustness and sparse approximation. Neurocomputing 48(1):85–105