Neural Comput & Applic
DOI 10.1007/s00521-013-1385-z

ORIGINAL ARTICLE

A novel self-adaptive extreme learning machine based on affinity propagation for radial basis function neural network

Shifei Ding · Gang Ma · Zhongzhi Shi

Received: 27 October 2012 / Accepted: 1 March 2013

Abstract In this paper, a novel self-adaptive extreme learning machine (ELM) based on affinity propagation (AP) is proposed to optimize the radial basis function neural network (RBFNN). As is well known, the parameters of original ELM which developed by G.-B. Huang are randomly determined. However, that cannot objectively obtain a set of optimal parameters of RBFNN trained by ELM algorithm for different realistic datasets. The AP algorithm can automatically produce a set of clustering centers for the different datasets. According to the results of AP, we can, respectively, get the cluster number and the radius value of each cluster. In that case, the above cluster number and radius value can be used to initialize the number and widths of hidden layer neurons in RBFNN and that is also the parameters of coefficient matrix $\boldsymbol{H}$ of ELM. This may successfully avoid the subjectivity prior knowledge and randomness of training RBFNN. Experimental results show that the method proposed in this thesis has a more powerful generalization capability than conventional ELM for an RBFNN.

Keywords Extreme learning machine (ELM) · Affinity propagation (AP) · Radial basis function neural network (RBFNN)

S. Ding (✉) · G. Ma
School of Computer Science and Technology,
China University of Mining and Technology,
Xuzhou 221116, 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 Sciences, Beijing 100190, China

1 Introduction

The radial basis function neural network (RBFNN) owned a high approximation, generalization capability, good local specialization, and global generalization ability, and it has become more and more popular in time series prediction, function approximation, data classification, pattern recognition, control, and nonlinear system identification in recent years [1–3]. As one of feedforward neural network type, RBFNN has been demonstrated to have the ability to approximate any reasonable continuous function mapping with a satisfactory level of accuracy in theory at least [4, 5].

However, the common learning methods of neural networks are based on the thinking of adjusting parameters iteratively, and its most serious weakness throughout the training process is high time-consuming which decreases the time efficiency of training neural networks [6]. Some people have provided that the learning ability depends on the numbers of hidden layer neurons and it has no relation with the weights of input layer neurons for single-hidden layer feedforward neural networks (SLFN) [7]. Although this view inspires us 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 iteratively adjusting parameters. G.-B. Huang proposed an ELM algorithm to train SLFN by analyzing and summarizing the thinking of iteratively adjusting parameters [8], the whole process completes in one operation without iteration, which not only solves the high time-consuming problem of training SLFN by the thinking of adjusting parameters iteratively, but also tremendously increases the training and testing accuracy of neural networks [9].

However, the ELM algorithm proposed by G.-B. Huang also has deficiencies. For example, how to determine

Published online: 17 March 2013

self-adaptively the parameters of coefficient matrix $\boldsymbol{H}$ of ELM, the numbers and widths of neurons in RBFNN hidden layer, and so on. The general method determined the above parameters by randomly giving or artificially setting. In that case, there are some researchers used the conventional strategies including hierarchical clustering, *k*-means clustering, and fuzzy *k*-means clustering to determine the parameters of RBFNN hidden layer [10, 11]. Those strategies also show that select randomly from input data and adjust constantly by the clustering algorithms until they no longer change and cannot determine the number of hidden neurons when no prior knowledge is provided. In addition, in recent years, evolution algorithms, such as the genetic algorithm (GA) and differential evolution (DE), have been adopted widely to optimize RBF centers [12–14]. Although the more reduced structure can be achieved through evolution algorithms, a better generalization performance cannot usually be guaranteed.

AP, a novel cluster algorithm called "affinity propagation," was proposed by Brendan J. Frey* and Delbert Dueck in 2007 [15]. The AP clustering algorithm has been shown to be very useful for many applications in face images, gene expression, and text summarization. The AP simultaneously considers all data points as potential exemplars and recursively transmits real-valued messages along the edges of the network until a good set of centers is generated [16].

Therefore, this paper mainly aims to solve the problem about how to self-adaptively and objectivity determine the numbers and widths of hidden layer neurons in RBFNN, the parameters of coefficient matrix $\boldsymbol{H}$ of ELM according to the different realistic datasets based on the AP algorithm. In the whole process that avoids the subjectivity prior knowledge and randomness of training RBFNN.

This paper is organized as follows: Sect. 1 introduces why the ELM is proposed to train RBFNN based on AP. Section 2 represents the general structure of RBFNN. Section 3 describes the general framework of ELM training SLFN and the theory of AP algorithm. Section 4 reveals our proposed algorithm, ELM based on AP (AP–ELM). Section 5 shows experimental studies for verifying the proposed algorithm in this paper. Section 6 is the conclusion and prospect of the thesis.

2 The structure of radial basis function neural network

Radial basis function neural network (RBFNN) is composed of three different layers with feedforward architecture: input layer, hidden layer, and output layer. The common structure of RBFNN is shown in Fig. 1.

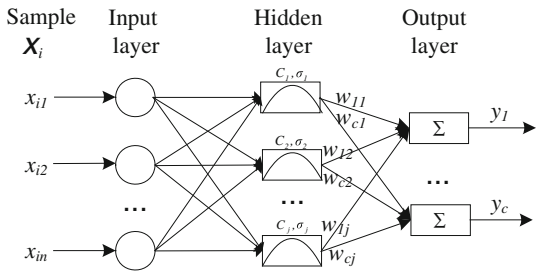

Fig. 1 The structure of RBFNN

Here, the input layer is a set of n neurons, which accepts the elements of an *n*-dimension input vector $\mathbf{X}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. The neurons in input layer are fully connected to a hidden layer that is composed of *j* hidden neurons, which are called radial basis function (RBF) units. The RBF units are connected directly to all elements in the output layer, which activates the response of the neural network to the input pattern, where the vector $(y_1, y_2, \ldots, y_c)$ is the output class of whole neural network in output layer, and *c* is the number of neurons in output layer.

Especially, the each RBF unit in hidden layer contains an RBF, a decision-making model. Usually, the Gaussian function is used as the RBF basis function:

$$R_k(\mathbf{C}_k, \sigma_k, \mathbf{X}_i) = \exp\left(-\|\mathbf{X}_i - \mathbf{C}_k\|^2 \Big/ 2\sigma_k^2\right)$$
$$k = 1, 2, \ldots, j; \quad i = 1, 2, \ldots, N; \quad \mathbf{X}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$$

where *j* is the number of neurons in hidden layer, we call $\boldsymbol{X}_i$ the input value of the neural network in input layer, which also is the *n*-dimension input vector of each unit in hidden layer, $\mathbf{C}_k$ and σ_k represent the center and width of the *k*th unit, respectively, $\|\mathbf{X}_i - \mathbf{C}_k\|$ is the distance between input vector $\boldsymbol{X}_i$ and the center of hidden layer neurons of RBF $\mathbf{C}_k$.

The output of RBFNN is a linear combination of the responses of RBF units in the hidden layer. The values of the output neurons can be calculated as:

$$y_m = \sum_{k=1}^{j} w_{mk} R_k(\mathbf{C}_k, \sigma_k, \mathbf{X}_i)$$
$$k = 1, 2, \ldots, j; \quad m = 1, 2, \ldots, c; \quad i = 1, 2, \ldots, N.$$

where w_{mk} is the weight between the *k*th neuron of hidden layer and the *m*th neuron of output layer, R_k is the output value of the *k*th neuron in hidden layer, and y_m is the output value of the *m*th neuron in output layer.

3 Preliminaries

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 a linear equation be:

$$Ax = y$$

If A was an n-order matrix and $\det A \neq 0$, the solution of linear equation is existence and uniqueness and can be described as follows:

$$x = A^{-1}y$$

However, the matrix A generally is a singular matrix or any $m \times n$ dimension matrix ($m \neq n$) in practical problems; obviously, the common inverse A^{-1} is not existing. In order 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 equation can be represented as a compact style, that is:

$$x = Gy$$

Definition 1: Let $A \in R^{m\times n}$, if there was a matrix G ($n \times m$), and it satisfies the following conditions:

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

Then, the matrix G is called plus sign (+) inverse, pseudo-inverse, or Moore–Penrose inverse of matrix A, and it is written as A^+ [17].

Definition 2: for any $y \in R^m$, if $x_0 \in R^n$ could satisfy the following conditions:

$$\|x_0\| \leq \|x\|, \forall x \in \{x : \|Ax - y\| \leq \|Az - y\|, \forall z \in R^n\}$$

Then, x_0 is regarded to be a minimum norm least-squares solution of a general linear system $Ax = y$, where $\|\cdot\|$ is a norm in Euclidean space [18].

The above formula 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 Extreme learning machine

In order to understand how the combination of the AP and the ELM is applied to the learning of a RBFNN further, it is important for us to know the ELM learning algorithm proposed for the single-hidden layer feedforward networks previously [19].

For N arbitrary distinct samples $(\boldsymbol{x}_i, \boldsymbol{t}_i)$, where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ and $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in R^m$, the standard SLFN with k hidden neurons function $g(x)$ is mathematically modeled as:

$$\sum_{i=1}^{k} \beta_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) = \boldsymbol{o}_j, \quad j = 1, 2, \ldots, N.$$

$\boldsymbol{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector full connecting the ith hidden neuron and the input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector full connecting the ith hidden neuron and the output neurons, and $\mathrm{b_i}$ is the threshold value of the ith hidden neuron, $w_\mathrm{i} \cdot x_\mathrm{j}$ denotes the inner product of w_i and x_j.

The standard SLFN which have k hidden layer neurons and each with an activation function $g(x)$ can approximate these N samples with zero error, that means $\sum_{j=1}^{k} \|o_j - t_j\| = 0$, that is, there exist β_i, w_i, and $\mathrm{b_i}$ such that:

$$\sum_{i=1}^{k} \beta_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) = \boldsymbol{t}_j, \quad j = 1, 2, \ldots, N.$$

The above N equations can be written compactly as:

$$\mathbf{H}\beta = \mathbf{T}$$

Where:

$$\begin{aligned} \boldsymbol{H}(\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{x}) &= \boldsymbol{H}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k, \ldots, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)\\ &= \begin{bmatrix} g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_1 + b_1) & \cdots & g(\boldsymbol{w}_k \cdot \boldsymbol{x}_1 + b_k)\\ \vdots & \cdots & \vdots\\ g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_N + b_1) & \cdots & g(\boldsymbol{w}_k \cdot \boldsymbol{x}_N + b_k) \end{bmatrix}_{N\times k}\\ \beta &= \begin{bmatrix} \beta_1^T\\ \vdots\\ \beta_k^T \end{bmatrix}_{k\times m} \quad ; \quad \boldsymbol{T} = \begin{bmatrix} t_1^T\\ \vdots\\ t_N^T \end{bmatrix}_{N\times m} \end{aligned}$$

as named in Huang and Babri [20] and Huang [21], $\boldsymbol{H}$ is called the hidden layer output matrix of the neural network, the ith column of $\boldsymbol{H}$ is the ith hidden layer neuron output with respect to inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$.

Now, we can solve easily the linear system equation $H\beta = T$ adopting the knowledge from the Sect. 3.1 as follows:

$$\beta = \mathbf{H}^{+}\mathbf{T}$$

3.3 Affinity propagation

It is important to cluster data for processing sensory signals and detecting patterns by identifying a subset of representative examples. For instance, there are some other

popular cluster algorithms, like k-means clustering technique, it can use the method of randomly choosing an initial subset of data points to iteratively refine such "exemplars;" yet, this works well only if that initial choice is very close to a good solution. However, the AP clustering algorithm can avoid those deficiencies. How the AP works is described as follows [14].

Affinity propagation clusters N data points according to the similarities between each other, a negative squared error (Euclidean distance) is used for each similarity: For points $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$, $s(i,k) = -||x_i - x_k||^2$, these similarities are symmetrical. Then, the similarity matrix $\boldsymbol{S}$ is composed of the similarity $s\ (i,\ k)$ between arbitrary points $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$. AP algorithm does not need to give the number of clustering centers beforehand, and the all data points can be used to the potential clustering centers, they are called exemplars.What we can see, between all data points there are two kinds of message exchanged, and the different message takes into account a different kind of competition. The "responsibility" $r(i,\ k)$, that is, a kind of message sent from data point i to candidate exemplar point k, reflects the accumulated evidence that the point k, as a exemplar, is how well suited to serve for the other point i, explaining other potential exemplars for point i (Fig. 2a). The "availability" $a(i,k)$, that is, a kind of message sent from candidate exemplar point k to point i, reflects the accumulated evidence for it would be how appropriate for the point i to choose the point k as its exemplar, explaining the support from other points that point k should be an exemplar (Fig. 2b). $r(i,k)$ and $a(i,k)$ can be deemed as log-probability ratios.

First of all, the algorithm uses zero to initialize the availabilities: $a\ (i,\ k) = 0$. Then, the responsibilities are calculated using the following rule:

$$r(i,k) = s(i,k) - \max\{a(i,j) + s(i,j)\}$$
$$j \in \{1,2,\cdots,N; \quad \text{and} \quad j \neq k\}$$

The responsibility update will lead to all candidate exemplars compete for the ownership of a data point. Afterward, the following availability update collects evidence from data points that determine whether each candidate exemplar would make a good exemplar:

$$a(i,k) = \min\{0, r(k,k) + \sum_j \{\max(0, r(j,k))\}\}$$
$$j \in \{1,\ 2,\ldots,N; \quad \text{and} \quad j \neq i, \quad j \neq k\}$$

The availability $a(i,k)$ is updated by the self-responsibility $r(k,\ k)$ plus the sum of the positive responsibilities that candidate exemplar k receives from other points. It is only necessary for a good exemplar to take into account some data points well; therefore, just only the positive portions of incoming responsibilities are added.

In order to restrict the impact of strong incoming positive responsibilities, the whole sum is threshold so that it cannot be greater than zero. The "self-availability" $a(i,k)$ is updated differently:

$$a(k,k) = \sum_j \max\{0, r(j,k)\} \quad j \in \{1,\ 2,\ldots,N; \quad \text{and} \quad j \neq k\}$$

The message, in above formula, suggests accumulated evidence that point k become an exemplar, based on the positive responsibilities that sent to candidate exemplar k from the other points.

The previous-mentioned update rules need only some local, simple computations which are so easily implemented, and exchanging messages occurs between pairs of points with known similarities. The algorithm can combine availabilities and responsibilities to identify exemplars for any point during affinity propagation.

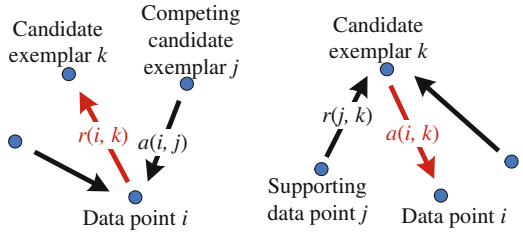

Fig. 2 Two kinds of message exchanged between data points

4 The proposed optimizing method

4.1 ELM for RBFNN

Of all the parameters, the dimension of coefficient matrix $\boldsymbol{H}$ of ELM is vital, and it reflects the complexity of matrix $\boldsymbol{H}$ inversion, the execution efficiency is extremely high if we can confirm a best dimension number, so that an optimal ELM can be gotten. When ELM is applied to train RBFNN, the matrix $\boldsymbol{H}$ will be described as follows form:

$$\boldsymbol{H}(\boldsymbol{C},\sigma,\boldsymbol{x}) = \boldsymbol{H}(\boldsymbol{C}_1,\ldots,\boldsymbol{C}_k,\sigma_1,\ldots,\sigma_k,\ldots,\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N)$$
$$= \begin{bmatrix} R(\boldsymbol{C}_1,\sigma_1,\boldsymbol{x}_1) & \cdots & R(\boldsymbol{C}_k,\sigma_k,\boldsymbol{x}_1) \\ \vdots & \cdots & \vdots \\ R(\boldsymbol{C}_1,\sigma_1,\boldsymbol{x}_N) & \cdots & R(\boldsymbol{C}_k,\sigma_k,\boldsymbol{x}_N) \end{bmatrix}_{N\times k}$$

Of course, from the above formula, we can see that the computational efficiency of $\boldsymbol{H}^+$ is limited by the parameter k in matrix $\boldsymbol{H}$; therefore, it is so important to how to obtain a optimal dimension to improve the computational efficiency of $\boldsymbol{H}^+$. While the parameter k comes from the

Table 1 The description of AP–ELM algorithm

Algorithm 1 (AP–ELM): AP clustering for confirming the parameters of the coefficient matrix $\boldsymbol{H}$ of ELM

Input: The sample datasets $\boldsymbol{X}$ including sample attributes and classes, and the sample size is N

Output: the coefficient matrix $\boldsymbol{H}$

Step 1: Initialize the parameters of AP. Define the maximum iterations *maxIter* and damping factor *lam* and initialize similarity matrix $S_{N\times N}$ ($S_{ij} = S(i, j)$) according to the similarity measured with Euclidean distance between the data points in datasets.

Step 2: Calculate the responsibility r (i, k) and availability a (i, k).

Step 3: Assess. Judge whether point k is the center of clustering with the following rule:

$$r(k,k) + a(k,k) > 0$$

Step 4: Weighted update the $r(i, k)$ and $a(i, k)$ as follows:

$$r(i,k) = (1 - \text{lam}) * r(i,k) + \text{lam} * r(i-1,k)$$

$$a(i,k) = (1 - \text{lam}) * a(i,k) + \text{lam} * a(i-1,k)$$

Step 5: If the convergence conditions of AP clustering are met, go to step 6; otherwise, return to step 2.

Step 6: Record the centers and the number of clusters according to the results of AP clustering, and calculate the widths of the clustering centers as follows:

$$\sigma_i = \sqrt{\frac{1}{NC_i}\sum_{j=1}^{NC_i}\left\|\mathbf{X}_j - \mathbf{C}_i\right\|^2}$$

where $\boldsymbol{C}_i$ denotes the center vector of the ith cluster, NC_i represents the number of data points in the ith cluster, $\boldsymbol{X}_j$ is an arbitrary data point in the ith cluster, and $\left\|\mathbf{X}_\mathrm{j} - \mathbf{C}_\mathrm{i}\right\|$ is the Euclidean distance between data point $\boldsymbol{X}_j$ and the clustering center $\mathbf{C}_\mathrm{i}$.

Step 7: Construct matrix $\boldsymbol{H}(\mathbf{C}, \sigma, \boldsymbol{x})$, definite expression in Sect. 4.1, according to the parameters C, σ, k and datasets $\boldsymbol{X}$.

End this algorithm.

number of clusters, so a set of better centers $\boldsymbol{C}$ and widths σ can facilitate the linear system results higher classification accuracy.

4.2 Description of the proposed method

So, how to search some better centers and widths also is most important. Consider to adopt a clustering idea to automatically confirm the parameters, $\boldsymbol{C}$, σ and k, of matrix $\boldsymbol{H}$ in advance. Here, $\boldsymbol{C}$ and σ are the center and radius of each cluster, respectively, and k is the number of all clusters.

In view of some common clustering algorithms, such as K-means algorithm, that must require the user to give a parameter k, the final clustering number, in advance before the algorithm works, which could not achieve an automatic status. Whereas, AP clustering algorithm mentioned in Sect. 3.3 can solve the problem of confirming the clustering number automatically. We adopt it to give the parameters, $\boldsymbol{C}$ σ and k, of coefficient matrix $\boldsymbol{H}$ of ELM automatically in advance, which can be called as AP–ELM. The working process of AP–ELM is shown in Table 1.

4.3 AP–ELM for RBFNN

A novel learning algorithm about RBFNN, AP–ELM, has been formulated in the Sect. 4.2. Now, how to apply AP–ELM to the RBFNN is listed in Table 2.

Table 2 The algorithm procedure of AP–ELM training RBFNN

Algorithm 2: AP–ELM trains RBFNN

Input: The sample datasets $\boldsymbol{X}$ including sample attributes and classes, and the sample size is N

Output: the information about training and generalizing of RBFNN

Step 1: Calculate the coefficient matrix $\boldsymbol{H}(\boldsymbol{C}, \sigma, \boldsymbol{x})$ using the algorithm 1, AP–ELM, in the Sect. 4.2.

Step 2: Calculate $\boldsymbol{H}^+$, Moore–Penrose generalized inverse of $\boldsymbol{H}(\boldsymbol{C}, \sigma, \boldsymbol{x})$.

Step 3: Get β by solving the linear system $H\beta = T$ according to $\boldsymbol{H}^+$ from the above step.

Step 4: Compute the training and generalizing accuracy. End this algorithm.

5 Experimental results

In order to demonstrate the performance of our proposed algorithm, AP–ELM, for RBFNN, we present two groups of experiments in this section based on the artificial datasets and the UCI datasets. Where the artificial datasets are from the bivariate Gaussian distribution, which contain 200 samples, the UCI datasets contain 6 datasets who are hepatitis, heart, sonar, wine, zoo, and iris datasets. They are all used to illustrate the characters and classification ability of the proposed algorithm in this paper compared with the original ELM proposed by G.-B. Huang. For each datasets in the experiments, half used for training and half used for

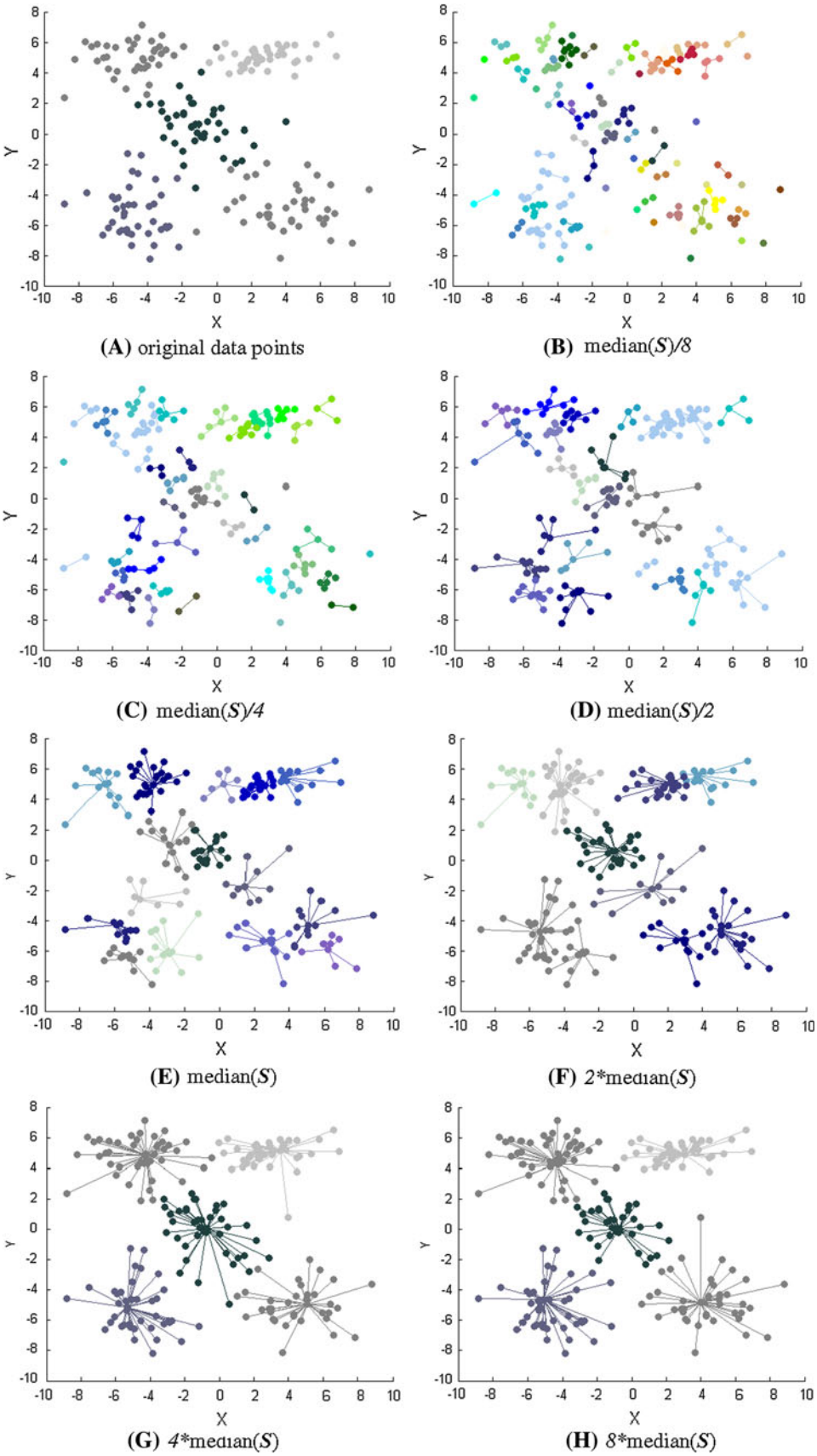

Fig. 3 AP clustering results for different preferences

Table 3 The efficiency of RBFNN for different preference of AP–ELM based on artificial datasets

Preference	Time(s)		Accuracy	
	Train	Test	Train	Test
Median(**S**)*/8*	3.2131	0.0469	0.9700	0.8700
Median(**S**)*/4*	3.1286	0.0313	0.9700	0.9600
Median(**S**)*/2*	3.2051	0.0156	0.9500	0.9100
Median(**S**)	3.1009	0.0000	0.9400	0.9700
*2**median(**S**)	3.0756	0.0156	0.9600	0.9500
*4**median(**S**)	2.5271	0.0156	0.9400	0.9500
*8**median(**S**)	2.9653	0.0156	0.9300	0.9600

Table 4 The experiment data from UCI datasets

	Number of samples	Number of attributes	Number of classes
Hepatitis	155	20	2
Heart	270	14	2
Sonar	208	61	2
Wine	178	14	3
Zoo	101	17	7
Iris	150	5	3

testing. The following two groups of experiments were done in the same experiment circumstance, that is, hardware system of computer is Intel(R) Core(TM) 2 CPU T5300, computer main frequency is 1.73 GHz, memory is 1.00 GB. Operation system is Microsoft Windows XP Professional. Programing environment is MATLAB 7.1.

5.1 Experiment on artificial datasets

In this experiment on artificial datasets, we can illustrate intuitively the effect of AP clustering algorithm with the different s (i, i) which is called preference, it is a reference value that the points i become a clustering center. The different preference will lead to the different cluster centers and the number of clusters and then influences the performance of AP–ELM. Generally, we use the median of the similarity matrix $\boldsymbol{S}$.

The artificial datasets containing 200 data points which come from the bivariate Gaussian distribution belong to 5 classes, respectively, and the each class owns 40 data points. In the experiment, we adopt 7 kinds of preferences to cluster the 200 data points, and the results mapping of each experiment were displayed in Fig. 3.

The images from (A) to (H) in Fig. 3 are clustering results according to different preference, original data points, median(***S***)*/8*, median(***S***)*/4*, median(***S***)*/2*, median(***S***), *2**median(***S***), *4**median(***S***), *8**median(***S***), based on a same set of datasets. It is not difficult to find that the number of cluster centers declines from 200 to 5 upon the decreasing preference. Where the number of cluster centers determines the dimension of coefficient matrix $\boldsymbol{H}$ of AP–ELM, so how

Table 5 The results of RBFNN with AP–ELM

Preference	Time(s)		Accuracy	
	Train	Test	Train	Test
Hepatitis				
Median(***S***)*/8*	1.1652	0.0781	1.0000	0.7922
Median(***S***)*/4*	1.0109	0.0469	0.9740	0.7403
Median(***S***)*/2*	1.2330	0.0469	0.9870	0.6234
Median(***S***)	1.2232	0.0156	0.9351	0.8442
*2**median(***S***)	1.2138	0.0156	0.9221	0.8701
*4**median(***S***)	0.9744	0.0000	0.8571	0.7922
*8**median(***S***)	0.9552	0.0000	0.8312	0.7792
Heart				
Median(***S***)*/8*	8.8860	0.1875	0.9852	0.6815
Median(***S***)*/4*	8.2460	0.1406	0.9778	0.5704
Median(***S***)*/2*	9.0572	0.0625	0.8741	0.7407
Median(***S***)	7.6441	0.0156	0.8444	0.8000
*2**median(***S***)	7.7889	0.0156	0.8370	0.8444
*4**median(***S***)	8.1604	0.0156	0.8444	0.8074
*8**median(***S***)	6.0934	0.0000	0.7852	0.7407
Sonar				
Median(***S***)*/8*	3.0893	0.2031	1.0000	0.5288
Median(***S***)*/4*	3.8365	0.1250	1.0000	0.5288
Median(***S***)*/2*	3.1488	0.0781	0.9904	0.8365
Median(***S***)	3.1920	0.0000	0.5865	0.6346
*2**median(***S***)	2.8508	0.0156	0.8173	0.8173
*4**median(***S***)	2.6794	0.0000	0.7500	0.7019
*8**median(***S***)	2.5826	0.0000	0.7115	0.6538
Wine				
Median(***S***)*/8*	1.6273	0.0938	1.0000	0.3258
Median(***S***)*/4*	1.4130	0.0938	1.0000	0.6292
Median(***S***)*/2*	2.0624	0.0313	1.0000	0.9888
Median(***S***)	1.7557	0.0000	1.0000	0.9663
*2**median(***S***)	1.5554	0.0000	0.9775	0.9663
*4**median(***S***)	1.5921	0.0000	0.9775	0.9213
*8**median(***S***)	1.4671	0.0000	0.9663	0.9551
Zoo				
Median(***S***)*/8*	0.2781	0.0156	1.0000	0.5800
Median(***S***)*/4*	0.2197	0.0156	1.0000	0.7000
Median(***S***)*/2*	0.1500	0.0000	0.8200	0.6800
Median(***S***)	0.2577	0.0000	0.9200	0.9400
*2**median(***S***)	0.3125	0.0000	0.9200	0.9200
*4**median(***S***)	0.2316	0.0000	0.8000	0.7800
*8**median(***S***)	0.2458	0.0000	0.7600	0.7800
Iris				
Median(***S***)*/8*	1.0178	0.0469	1.0000	0.8400
Median(***S***)*/4*	1.0769	0.0156	0.9733	0.8400
Median(***S***)*/2*	0.9887	0.0000	0.9733	0.8933
Median(***S***)	1.1588	0.0156	0.9733	0.9867
*2**median(***S***)	0.9836	0.0156	0.8933	0.9467
*4**median(***S***)	0.9041	0.0000	0.8800	0.9067
*8**median(***S***)	0.8491	0.0000	0.3333	0.3333

to obtain an appropriate preference is extremely vital. In Table 3, we get multiple training and testing accuracies about RBFNN adopting AP–ELM with different preferences based on the artificial datasets.

From the Table 3, a conclusion can be drawn as the best preference is median($\boldsymbol{S}$) for the current artificial datasets, the number of cluster centers is 15 in the image (E) from the Fig. 3, and then the dimension of coefficient matrix $\boldsymbol{H}$ of AP–ELM is 200-by-15. That avoids a blindness of determining the hidden layer neuron number of RBFNN randomly.

5.2 Experiment on UCI datasets

In this experiment on UCI datasets, we still choose 7 kinds of preferences to cluster the datasets, respectively, also take the class number of cluster centers as the dimension of coefficient matrix $\boldsymbol{H}$ of AP–ELM and the hidden layer neuron number of RBFNN. The main information of all datasets is shown in Table 4.

We get the number and radiuses of all clustering centers from original datasets through the algorithm 1 described in the Sect. 4.2, and then take them as the number and widths of hidden layer neurons in RBFNN, which also are the parameters of coefficient matrix $\boldsymbol{H}$ of ELM in the Sect. 4.3. Compute accuracy mainly depends on the mean-square error between the output of RBFNN and the expected output of the samples.

We select 7 different values of preference with each datasets, including median($\boldsymbol{S}$)/8, median($\boldsymbol{S}$)/4, median($\boldsymbol{S}$)/2, median($\boldsymbol{S}$), 2*median($\boldsymbol{S}$), 4*median($\boldsymbol{S}$), and 8*median($\boldsymbol{S}$), where the median($\boldsymbol{S}$) is the median value of similarity matrix $\boldsymbol{S}$ in AP algorithm. The classification results of RBFNN with AP–ELM based on each datasets from Table 2 are listed in Table 3, where the information is obtained from the average value of 10 times experiments.

In Table 5, what we can see that the values with brunet background are the preference of obtaining optimal learning efficiency of RBFNN learned by AP–ELM and mainly distribute in the interval 2*median($\boldsymbol{S}$) and median($\boldsymbol{S}$)/2 (median($\boldsymbol{S}$) < 0), and also the preference influences the training time and testing time, where the preference value is smaller and the above two kinds of time become less, that is, a good suggestion for the future experiments about RBFNN learned by AP–ELM based on the other datasets.

5.3 The comparison about AP–ELM and ELM

In order to demonstrate the classification ability of our method proposed in this paper further, we have chosen the original ELM algorithm to compare with AP–ELM based on the same datasets from Table 2 and the same experimental platform as follows. In the comparison experiment, the number of hidden layer neurons of RBFNN learned by ELM is 50, and the width of each in neuron hidden layer and the linear weights connecting the hidden layer and input layer generate randomly, while the above parameters in AP–ELM generate by the results of AP clustering and choose median($\boldsymbol{S}$) as the preference. The results of comparison experiment are listed in the Table 6.

Table 6 The results of comparison experiment about AP–ELM and ELM

	Accuracy of AP–ELM		Accuracy of ELM	
	Train	Test	Train	Test
Hepatitis	0.9221	0.8701	0.987013	0.831169
Heart	0.8370	0.8444	0.866667	0.792593
Sonar	0.9904	0.8365	0.788462	0.701923
Wine	1.0000	0.9888	0.988764	0.955056
Zoo	0.9200	0.9400	1.000000	0.740000
Iris	0.9733	0.9867	1.000000	0.946667

Obviously, from the above Table 6, we can get that the classification efficiency of RBFNN learned by AP–ELM is higher than ELM. In general, the generalization performance of our method in this paper is reasonably good compared with original ELM. Which make known that the AP cluster algorithm can search a set of more appropriate parameters compared with the random method for ELM.

6 Conclusions

This paper mainly introduced a novel learning method, AP–ELM, about RBFNN with ELM based on AP algorithm. Through AP algorithm preprocess, the training datasets and obtain the number of clustering centers and the radius of each cluster. Then, those number and radiuses of clustering centers are applying to determine the parameters of ELM. Therefore, the parameters of coefficient matrix $\boldsymbol{H}$ of ELM reach optimal composite state, and the classification accuracy of RBFNN is improved compared with the normal ELM. In the future, we will turn to the higher efficiency of RBFNN trained by AP–ELM about large-scale datasets.

Acknowledgments This work is supported by the National Basic Research Program of China (No. 2013CB329502), the National Natural Science Foundation of China (No. 41074003), and the Opening Foundation of the Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (IIP2010-1).

References

1. Muzhou Hou, Xuli Han (2011) The multidimensional function approximation based on constructive wavelet RBF neural network. Appl Soft Comput 11(2):2173–2177

2. Beyhan S, Alci M (2010) Stable modeling based control methods using a new RBF network. ISA Trans 49(4):510–518
3. Han H-G, Chen Q, Qiao J-F (2011) An efficient self-organizing RBF neural network for water quality prediction. Neural Netw 24(7):717–725
4. Zhang A, Zhang L (2004) RBF neural networks for the prediction of building interference effects. Comput Struct 82(27):2333–2339
5. Zeng W, Wang C (2012) Human gait recognition via deterministic learning. Neural Netw 35:92–102
6. Shifei D, Ma G, Xu X (2011) A rough RBF neural networks optimized by the genetic algorithm. Adv Inf Sci Serv Sci 3(7): 332–339
7. Mao Z, Jia M (2012) ELM based LF temperature prediction model and its online sequential learning. Control and Decision Conference (CCDC), 2012 24th Chinese, vol 5. pp 2362–2365
8. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1-3): 489–501
9. Liu X, Gao C, Li P (2012) A comparative analysis of support vector machines and extreme learning machines. Neural Netw 33:58–66
10. Park H-S, Chung Y-D, Oh S-K, Pedrycz W, Kim H-K (2011) Design of information granule-oriented RBF neural networks and its application to power supply for high-field magnet. Eng Appl Artif Intell 24(3):543–554
11. Shen W, Guo X, Wu C, Wu D (2011) Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. Knowl Based Syst 24(3):378–385
12. Wu X, Zhu X, Cao G, Tu H (2008) Predictive control of SOFC based on a GA-RBF neural network model. J Power Sources 179(1):232–239
13. Qu N, Mi H, Wang B, Ren Y (2009) Application of GA-RBF networks to the nondestructive determination of active component in pharmaceutical powder by NIR spectroscopy. J Taiwan Inst Chem Eng 40(2):163–167
14. dos Coelho LS, Santos AAP (2011) A RBF neural network model with GARCH errors: application to electricity price forecasting. Electr Power Syst Res 81(1):74–83
15. Frey BJ, Dueck D (2007) Clustering by passing messages between data points. Science 315(5814):972–976
16. Zhong Y, Zheng M, Wu J, Shen W, Zhou Y, Zhou C (2012) Search the optimal preference of affinity propagation algorithm. In: Intelligent computation technology and automation (ICICTA), 2012 fifth international conference, vol 1. pp 304–307
17. Cheng L, Hu J (2009) Matrix theory. China University of Mining and Technology Press, Xuzhou
18. Huang G, Siew C-K (2004) Extreme learning machine: RBF network case. Control Autom Robot Vis Conf 2:1029–1036
19. Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feed forward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004), vol 6. pp 25–29
20. Huang G-B, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Networks 9(1):224–229
21. Huang G-B (2003) Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans Neural Networks 14(2):274–281