Subtractive Clustering Based RBF Neural Network Model for Outlier Detection

Peng Yang
School of Computer Science, Chongqing University, Chongqing, China
Email: llylab@21cn.com

Qingsheng Zhu and Xun Zhong
School of Computer Science, Chongqing University, Chongqing, China
Email: qszhu@cqu.edu.cn

***Abstract*—Outlier detection has many important applications in the field of fraud detection, network robustness analysis and intrusion detection. Some researches have utilized the neural network to solve the problem because it has the advantage of powerful modeling ability. In this paper, we propose a RBF neural network model using subtractive clustering algorithm for selecting the hidden node centers, which can achieve faster training speed. In the meantime, the RBF network was trained with a regularization term so as to minimize the variances of the nodes in the hidden layer and perform more accurate prediction. By defining the degree of outlier, we can effectively find the abnormal data whose actual output is serious deviation from its expectation as long as the output is certainty. Experimental results on different datasets show that the proposed RBF model has higher detection rate as well as lower false positive rate comparing with the other methods, and it can be an effective solution for detecting outliers.**

***Index Terms*—outlier detection, radial basis function, neural network, subtractive clustering**

I. INTRODUCTION

An outlier is defined as the data point which is very different from the rest of data [1][18][20]. It may be caused by measurement error or the result of inherent data variability. Many data mining algorithms try to minimize the influence of outliers or eliminate them all together. However, this could result in the loss of important hidden information [2][19][21]. Some data mining applications are focused on outlier detection, and it is the essential result of a data analysis. For example, while detecting fraudulent credit card transactions, the outliers are typical examples that may indicate fraudulent activity, and the entire data mining process is concentrated on their detection [3].

Several different techniques have been proposed for outlier detection. Distribution-based methods are mostly used in early studies [4]. However, a large number of tests are often required to decide which distribution model fits the arbitrary dataset best. Thus, many other categories of methods are proposed. In [5], distance-based method is firstly introduced. Then, S. Ramaswamy and S. Kyuseok propose a novel formulation based on the distance of a point from its k nearest neighborhood and declare the top n points in this ranking to be outliers. Deviation-based identifies outliers by examining the main characteristics of objects in a group [6][23]. Objects that deviate from this description are considered outliers. Additionally, some density-based outlier detection algorithms are proposed. They rely on the local outlier factor of each point or depend on the local density of its core object neighborhood. Furthermore, many data mining algorithms in the literatures find outliers as a side-product of clustering algorithm. From the viewpoint of a clustering algorithm, outliers are objects not located in clusters of dataset [7][22].

If the information regarding the values of input and output attributes is available, supervised learning method such as SVM can be used to train a classification model and identify the outliers. SVM distinguishes outlier from the rest of the feature space in given dataset. It is characterized by efficient handling of high dimensional spaces and systemic nonlinear classification using advanced kernel function [8]. However, the time cost for training is high in SVM [9]. Since neural network can perform calculations from time to time with it units, and does not involve the complex process, it can be used for finding outliers. Tzyy ect. proposed a neural network with quantum evolutionary algorithm for the establishment of a nonlinear map when data are subject to outliers [13]. Shirish dealt with a wrapper method that build an initial model using neural network and treated values at the output of neurons in the output layer as the typicality scores [3]. Instances with lowest output values were treated as potential outliers. These researchers, however, aimed to reject outliers and obtain more robust network, rather than identifying the abnormal data. In [9], a RBF neural network based on recursive least square algorithm was proposed for anti-money laundering. But its generalization need be further studied. In this paper,

we used RBF network to construct a classification model for outlier detection and the experimental results validated its effectiveness.

The remainder of this paper is organized as follows. In section 2, the subtractive clustering method is introduced. We can use the algorithm to select the hidden node centers for RBF neural network. In section 3, we modified the error function in leaning procedure of RBF and defined degree of outlier for each input instances. Finally, an outlier detection algorithm based on RBF network with subtractive clustering was proposed. In section 4, some experiments are carried out to demonstrate advantages of the proposed algorithm. We use the detection rate and false positive rate to evaluate performance of different outlier detection algorithms. The conclusions are addressed in section 5.

II. SUBTRACTIVE CLUSTERING

The Subtractive Clustering (SC) method is adopted in this paper to determine the hidden node centers in RBF network since it allows a scatter input-output space partitioning [10][11]. SC is an improved version of the Mountain method. The Mountain method depends heavily on the grid resolution and the dimension of data, thereby it will be computationally inefficient when applied to high dimensional dataset with increasing grid resolution. With SC method, however, the mountain function is calculated on data point rather than grid point.

Let (x_i, y_i) be the training dataset, $1 \le i \le N$. In SC algorithm, x_i is considered as a potential hidden node center, whose potential is defined as Eq.(1) [12].

$$P_i = \sum_{l=1}^{N} \exp(-\alpha \| x_i - x_l \|^2) \quad (1)$$

where $\alpha = 4 / r_a^2$ and $r_a>0$ denotes the neighborhood radius for each cluster center. The potential associated with each data depends on its distance to all the neighborhoods. Obviously, the potential of a data point is high when its neighborhood is dense. After calculating potential for each point, the one with the highest potential value will be selected as the first hidden node center. Then the potential of each point is reduced to avoid closely spaced clusters. Selecting centers and revising potential is carried out iteratively until a stopping criteria satisfied. The SC algorithm can be described as follows.

Step 1: Calculate the potential P_i for each point, $1 \le i \le N$;

Step 2: Set the number of hidden node L=1 and select the data point with the highest potential value as the first hidden node center. Let x_1^* be the location of the point and P_1^*, its corresponding potential value;

Step 3: Revise the potential of each data point according to Eq.(2).

$$P_i = P_i - P_1^* \exp(-\beta \| x_i - x_1^* \|^2) \quad (2)$$

Step 4: If $\max_i P_i \le \varepsilon P_1^*$, terminate the algorithm; otherwise, set $L=L+1$ and find the data point with the highest potential value. Let x_L^* be the location of the point and P_L^*, its corresponding potential value;

Step 5: Select x_L^* as a new hidden node center. Revise the potential of eaxh data point according to Eq.(3). Goto step 4.

$$P_i = P_i - P_L^* \exp(\| x_i - x_L^* \|^2) \quad (3)$$

Note that in Eq.(2), $\beta = 4 / r_b^2$ and $r_b>0$ represents the radius of the neighborhood for which significant potential revising will occur. In order to avoid the selection of closely located hidden node centers, β is chosen to be less than α. Typically, $r_b = 1.25 r_a$. In step 4, the parameter ε should be selected within (0, 1). If ε is selected to be close to 0, a large number of hidden node centers will be generated. On the contrary, a value of ε close to 1 will lead to a small network structure. The algorithm never selects the same data point more than once.

We used several datasets to test the training speed of RBF network based on SC. The variables were scaled in [0,1] so that all the input and output values were of the same order of magnitude. The parameters ε, α and β were set to 0.4, 0.7 and 0.4 respectively. We compared this training model with the conventional RBF model which randomly selects an instance in training set as the hidden node center. The results are summarized in table 1. SC can select more desirable hidden node centers, thereby leading to a proper network structure. For each dataset, the mean squared error of SC based RBF network is lower. Especially, its training time is remarkably reduced. The experimental environment and datasets will be described in section 4.

TABLE I. COMPARISON OF DIFFERENT TRAINING ALGORITHMS

Dataset	Mean squared error of conventional RBF	Mean squared error of SC based RBF	Training time of conventional RBF (s)	Training time of SC based RBF (s)
Iris	0.187	0.159	1.92	1.03
Wine	0.210	0.195	6.75	2.09
Glass	0.192	0.172	7.97	2.47
Survival	0.197	0.188	36.70	13.43
Voting	0.227	0.201	43.28	21.56
Bands	0.263	0.225	80.14	32.01

III. OUTLIER DETECTION WITH SC BASED RBF NETWORK

An RBF network is a three-layer feed forward neural network which consists of an input layer, a hidden layer and an output layer.

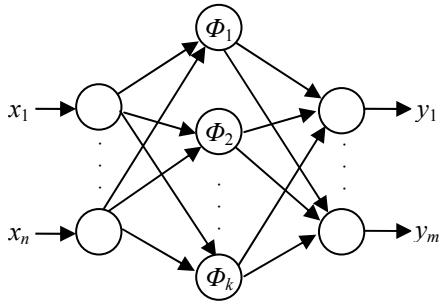


Figure 1. The RBF network structure

The structure of a traditional RBF network is shown in figure 1. The number of neurons in input layer depends on number of input attributes of training samples. The number of neurons in the output layer depends on number of target class labels. Assume that x is an n-dimensional input vector, there are m neurons in output layer and k neurons in hidden layer. It can be formulated as

$$\phi_j(x) = e^{-\|x-c_j\|/2\sigma_j^2} \tag{4}$$

$$y_i = \sum_{j=1}^{k} W_{ij}\phi_j(x) \tag{5}$$

where $i=\{1,2,\ldots,m\}$, $j=\{1,2,\ldots,k\}$, $\Phi_j(x)$ is the output of the jth hidden node, C_j is the center vector of Gaussian kernel function which has the same number of dimension with x, and σ_j is the width of Gaussian kernel function at the jth hidden node. Besides, y_i is the output of the ith output neurons and W_{ij} is the connect weight from the jth hidden node to the ith output node.

A. Learning Procedure

Most of the existing learning algorithms for RBF neural network employ different schemes for updating the output weights, i.e., the weights that connect the RBF with the output units, and the centers of the RBF, i.e., the vectors in the input space that represent the prototypes of the input vectors included in the training set [14]. In this section, the incremental learning method was used to construct a classification model because of its simple and easy implementation. We attempted to train the network to implement a desired input-output mapping by producing incremental changes of the weights of the network. If the responses of the radial basis functions are not substantially affected by incremental changes of the model, then the learning process reduces to incremental changes of the weights of the associative network which alone are unlikely to implement non-trivial input-output mappings. The ability of the network to implement a desired input-output mapping depends very strongly on the sensitivity of the responses of the radial basis functions to incremental changes of their corresponding model.

Assume that the training sample sequence is $\{x(l), d(l)\}_{l=1}^{N}$, where $x(l)$ is the lth input instance and $d(l)$ is its corresponding desired output. The difference between the actual output and expectation of sample $\{x(t),d(t)\}$ at t will be calculated to decide whether to update the network parameters and evaluate the output at t+1. The conventional error function is defined as $E = \frac{1}{2}\sum_{i=1}^{m}(d_i - y_i)^2$, where d_i and y_i are the expectation and actual output of the ith neurons in output layer. To eliminate the impact of outliers included in the samples, we used a regularization term instead of the outlier for training. It is a term added to the error function which aims to limit σ_j during training. The modified error function is defined as

$$E = \frac{1}{2}\sum_{i=1}^{m}(d_i - y_i)^2 + \frac{\lambda}{2k}\sum_{j=1}^{k}\sigma_j^2 \tag{6}$$

During the training, network parameters can be updated using the gradient descent technique.

$$W_{ij}^{t+1} = W_{ij}^{t} - \eta_w \frac{\partial E_t}{\partial W_{ij}^{t}} \tag{7}$$

$$C_j^{t+1} = C_j^{t} - \eta_c \frac{\partial E_t}{\partial C_j^{t}} \tag{8}$$

$$\sigma_j^{t+1} = \sigma_j^{t} - \eta_\sigma \frac{\partial E_t}{\partial \sigma_j^{t}} \tag{9}$$

where η. is the learning rate. Let $W_{ij}^{t+1} - W_{ij}^{t} = \Delta W_{ij}$, $C_j^{t+1} - C_j^{t} = \Delta C_j$, $\sigma_j^{t+1} - \sigma_j^{t} = \Delta\sigma_j$ and calculate the partial derivatives, the following equations can be obtained.

$$\Delta W_{ij} = -\eta_w \frac{\partial E_t}{\partial W_{ij}^{t}} = -\eta_w \frac{\partial E_t}{\partial y_i}\bullet\frac{\partial y_i}{\partial W_{ij}^{t}} = \eta_w \phi_j(x)(d_i - y_i) \tag{10}$$

$$\begin{aligned}\Delta C_j &= -\eta_c \frac{\partial E_t}{\partial C_j^{t}} = -\eta_c \frac{\partial E_t}{\partial y_i}\bullet\frac{\partial y_i}{\partial \phi_j(x)}\bullet\frac{\partial \phi_j(x)}{\partial C_j^{t}} \\ &= \eta_c \phi_j(x)\frac{(x - C_j)}{\sigma_j^2}\sum_{i=1}^{m}(W_{ij}(d_i - y_i))\end{aligned} \tag{11}$$

$$\begin{aligned}\Delta\sigma_j &= -\eta_\sigma \frac{\partial E_t}{\partial \sigma_j^{t}} = -\eta_\sigma \frac{\partial E_t}{\partial y_i}\bullet\frac{\partial y_i}{\partial \phi_j(x)}\bullet\frac{\partial \phi_j(x)}{\partial \sigma_j^{t}} \\ &= \eta_\sigma \phi_j(x)\frac{\| x - C_j \|^2}{\sigma_j^3}\sum_{i=1}^{m}(W_{ij}(d_i - y_i)) + \frac{\lambda}{k}\sigma_j\end{aligned} \tag{12}$$

Whenever a new instance is provided to the network, the parameters will be modified according to the

equations above if necessary. After finite training epochs, the network output of the instance can be controlled at the scope of allowable error.

B. Outlier detection

In RBF network, most of the methods employ unsupervised learning utilizing only the normal class during training since abnormal patterns are not usually available. Although outliers are not sufficient to construct a binary classifier, they can help refine the boundary around the normal patterns determined by the unsupervised method [15]. Generalization is to include the normal patterns within the boundary while specialization is to exclude patterns from all other classes. A balance between the two concepts is critical to classification accuracy. Even though outlier detection model are able to generalize from the normal data, most of them cannot specialize from data but only from a particular internal bias since they do not take outliers into consideration. In this sense, utilizing abnormal data helps a classification model specialize from data. It has been experimentally shown that one can achieve a higher accuracy by utilizing abnormal data. As mentioned above, while training the RBF neural network, we added a regularization term to the network error function so as to prevent reconstruction errors of abnormal patterns from being lower than a threshold. Then, the unseen data will be input into the trained network and those with higher degree of outlier, which is defined below, will be candidate outliers

Generally, a neural network classifier uses one neuron in the output layer for problems with two classes. If number of target classes is greater than two, then '*n*' neurons are used in the output layer one for each of the target classes. For the two class problems having classes, say class 0 and class 1, assuming a decision boundary at 0.5, the neuron in the output layer generates the outputs between 0.0 and 0.5 for all the instances that are classified into, say, class 0. For the instances that are classified into class 1, the neuron output is in the range of 0.5 to 1.0. The actual output of the neuron is transformed back to real-world values to indicate the actual class labels. For multi-class problems with *n* target classes, the classifier normally designates *one* neuron in the output layer for each of the *n* target classes [3]. For example, a neural network classifier may use three output neurons to represent three target classes (Class 1, 2 and 3) in Iris dataset. Classification decision is provided in the form of 1-of-*N* code. Thus, instances that belong to class 1, the output neuron corresponding to class 1 will have values between 1.0 and 0.5 and the other two neurons will have values between 0.0 and 0.5. Instances that belong to class 2, the designated output neuron for class 2 will have values between 1.0 and 0.5 and the other two neurons will have values between 0.0 and 0.5. The rest may be deduced by analogy.

The output y of a neural network can be interpreted as the probability that an instance belongs to a class. When y approaches 1, we feel more certain that the instance is in this class. On the contrary, the instance will not be considered within this class. The certainty $C(y)$ of a neural network output can be defined as follow.

$$C(y)=\begin{cases} y & \text{if } y \geq 0.5 \\ 1\text{-}y & \text{otherwise} \end{cases} \tag{13}$$

Note that the certainty behaves symmetrically with respect to positive and negative decisions. Considering the case of positive decisions, i.e., y≥0.5, if $C(y_1)< C(y_2)$ we say that network output y_1 is less certain than y_2. We can define the degree of outlier for instance x as follow.

$$DO(x)=\frac{1}{m}\sum_{i=1}^{m} C(y_i)\bullet| d_i - y_i | \tag{14}$$

where m is the number of neuron in output layer. y_i is the output of neural network and $C(y_i)$ is the corresponding certainty, which are defined by Eq.(5) and Eq.(13) respectively. The intuitive meaning of this definition is that if the actual output of instance x is serious deviation from its expectation and the output is certainty, x can be determined as outlier with great probability. Thus, the top n instances with the maximal $DO(x)$ value will be considered as outliers. Now, the outlier detection algorithm based on RBF (referred to as SC-RBF) can be described as follows.

Step 1: Select the network hidden node center C_j (j=1,2,…m) using SC algorithm;

Step 2: Calculate the parameter σ for Gaussian function, select the connect weights W_{ij} randomly and initiate the error threshold θ, t=1;

Step 3: Calculate the output error E for each training data according to Eq.(6). If $E>\theta$ goto step 4;

Step 4: Calculate $\Delta\sigma_j$, ΔW_{ij} and ΔC_j according to Eq.(10)-(12), adjust corresponding Gaussian width, linear weight and hidden node center;

Step 5: t=t+1. If $t\leq MaxS$ (the maximal number of training sample), goto step 3; otherwise, terminate training;

Step 6: Classify all instances using the trained model and record the output values of the neurons in the output layer for each of the testing instances;

Step 7: Calculate DO for all input instances according to Eq.(14). Sort DO values by ascending order and the corresponding instances of last n records will be considered as outliers.

We firstly use SC algorithm to train a RBF network as a classification model. As mentioned in section 2, it helps to reduce the training epochs. Then, the incremental learning method was used to train RBF network. Since the training set included outliers, we modified the conventional error function and added a regularization term to eliminate the influence of these abnormal data. The training process did not terminate until the input instance exhausted. Finally, using the trained model to

classify the input testing instances, calculate the degree of outlier for all the instances and sort them by ascending order. Obviously, the least typical instances of each class will appear last and they are candidate outliers.

IV. EXPERIMENTAL RESULTS

In this section, we conducted a few experiments to validate the proposed algorithm and compared it with the other two typical methods: support vector machine (SVM) based outlier detection algorithm and K-means based outlier detection algorithm. We use the SVM algorithm in the package provided by LIBSVM [16]. The SVM algorithm is based on the HVDM distance [17]. The incorrectly classification punishment factor G is set to 100 and the control parameter η is set to 1. The outlier could be discriminated by the average and standard deviation [9]. The K-means based algorithm and our proposed algorithm were implemented in MATLAB. In K-means based algorithm, we set the parameter k according to the actual clustering number of all datasets and the instance which could not be included in a cluster will be considered as outlier. All algorithms were running on Pentium 1.6G PC with 1G main memory. We tested each of the algorithms on datasets acquired from the UCI Machine Learning Repository. For each dataset, classification models were constructed using the standard technique of 10-fold Cross Validation. Some abnormal data were added deliberately into the original datasets. Table 2 describes the characteristics of these datasets.

TABLE II. CHARACTERISTICS OF DATASETS

Dataset	No. of training data	No. of testing data	No. of outliers	No. of attributes	No. of classes
Iris	80	80	10	4	3
Wine	120	70	12	13	3
Glass	150	80	16	9	2
Survival	220	100	14	3	2
Voting	350	100	15	16	2
Bands	400	130	18	39	2

We use evaluation metric detection rate (DR) and false positive rate (FPR) to test how well the outlier detection algorithms work, which are defined as follow.

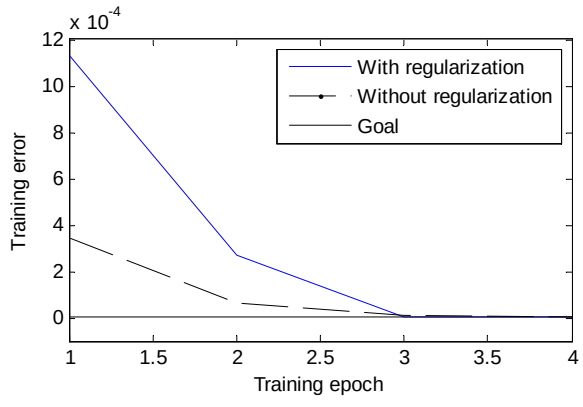
$$DR = \frac{NO}{TNO} \tag{15}$$

where NO is the number of outliers detected by the algorithm and TNO is the total number of outliers presented in the testing set.

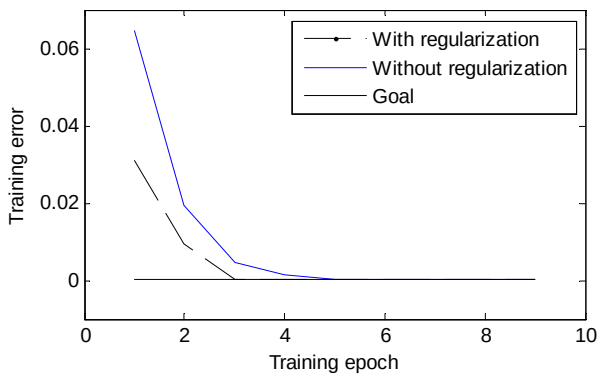
$$FPR = \frac{INO}{TNN} \tag{16}$$

where INO is the number of normal instances which are incorrectly identified as outliers and TNN is the total number of normal instances in the testing set.

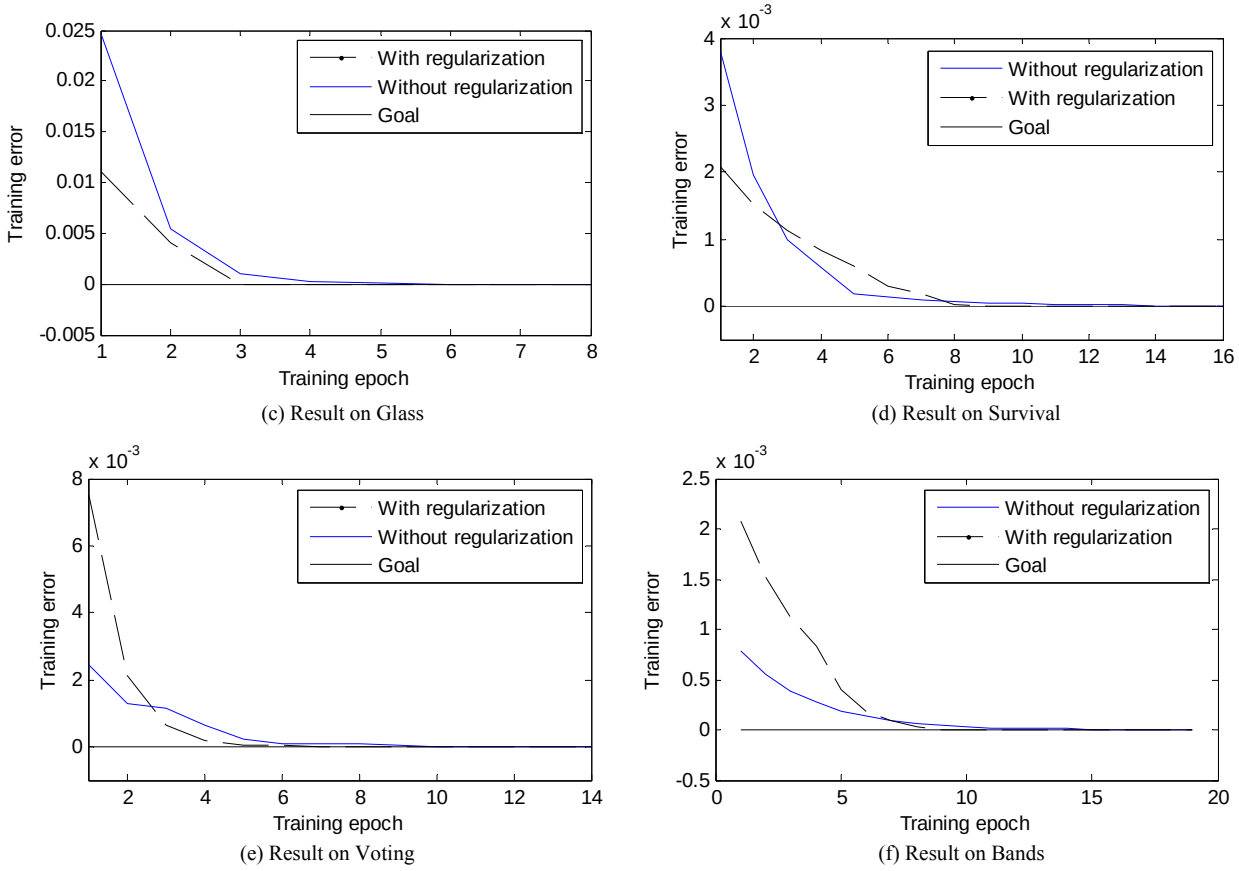
SC algorithm was used in RBF network to initiate the hidden node center and the leaning rate was set to 0.02 for all experiments. The regularization term is used to limit the expansion of Gaussian width during training to achieve outlier rejection. The regularization coefficient λ was empirically set to 0.02 and the termination criterion θ is set to 1E-6. The error curve of SC-RBF network trained with/without the regularization term is depicted as figure 2. X-axis expresses the training epochs and Y-axis expresses the training error. While training on Iris, for example, the mean epochs of SC-RBF network with the regularization term were 3, which was less than that of without regularization, to meet the training goal (see fig.2a). It shows that by using the regularization term, the CPU time for training can be greatly reduced. For other datasets, the similar results can be achieved.



(a) Result on Iris



(b) Result on Wine



(c) Result on Glass (d) Result on Survival

(e) Result on Voting (f) Result on Bands

Figure 2. Error curve of SC-RBF network

The training for RBF network is completed when the error function converges. Then we select the top n points in testing set with the highest DO values, which will be considered as outliers. The experimental results by applying different outlier detection algorithms are shown in table 3. Bold entries indicate which algorithm had the best performance.

TABLE III. DR AND FPR COMPARISON OF DIFFERENT ALGORITHMS

Dataset	SC-RBF with regularization term		SC-RBF without regularization term		K-means		SVM	
	DR	FPR	DR	FPR	DR	FPR	DR	FPR
Iris	0.70	0.020	0.60	0.027	**0.80**	**0.013**	**0.80**	**0.013**
Wine	**0.92**	**0.006**	0.83	0.011	0.83	0.011	0.75	0.017
Glass	**0.93**	**0.005**	0.88	0.009	0.81	0.014	0.75	0.019
Survival	**0.93**	**0.003**	0.86	0.007	0.79	0.010	0.71	0.013
Voting	**0.87**	**0.005**	0.73	0.009	0.67	0.011	0.60	0.014
Bands	**0.94**	**0.002**	0.89	0.004	0.83	0.006	0.78	0.008

Iris consists of 160 instances with 4 attributes and includes 3 classes. In RBF training model, there are 4 neurons in the input layer, 3 neurons in the hidden layer and 3 neurons in the output layer. The DR and FPR of SC-RBF with regularization term is 0.7 and 0.02 respectively. Comparing with the other two algorithms, the performance of SC-RBF is lower. This is because only 80 instances were used to train the model in such case. However, SVM is more adapted to small samples and it has better performance.

Wine consists of 190 instances with 13 attributes and includes 3 classes. In RBF training model, there are 13 neurons in the input layer, 4 neurons in the hidden layer and 3 neurons in the output layer. The DR and FPR of SC-RBF with regularization term is 0.92 and 0.006 respectively. This time, SC-RBF has the best performance among all the algorithms.

Glass consists of 230 instances with 9 attributes and includes 2 classes. In RBF training model, there are 9 neurons in the input layer, 3 neurons in the hidden layer and 2 neurons in the output layer. The DR and FPR of SC-RBF with regularization term is 0.93 and 0.005 respectively. SC-RBF has the best performance among all the algorithms.

Survival consists of 320 instances with 3 attributes and includes 2 classes. In RBF training model, there are 3

neurons in the input layer, 2 neurons in the hidden layer and 2 neurons in the output layer. The DR and FPR of SC-RBF with regularization term is 0.932 and 0.003 respectively. Again, SC-RBF has the best performance among all the algorithms.

Voting consists of 450 instances with 16 attributes and includes 2 classes. In RBF training model, there are 16 neurons in the input layer, 3 neurons in the hidden layer and 2 neurons in the output layer. The DR and FPR of SC-RBF with regularization term is 0.87 and 0.005 respectively. SC-RBF has the best performance among all the algorithms.

Bands consists of 530 instances with 39 attributes and includes 2 classes. In RBF training model, there are 20 neurons in the input layer, 5 neurons in the hidden layer and 2 neurons in the output layer. The DR and FPR of SC-RBF with regularization term is 0.92 and 0.006 respectively. Again, SC-RBF has the best performance among all the algorithms.

In summary, SC-RBF outperformed against the other two algorithms for outlier detection as long as with enough training instances. Note that in all cases, the performance of SC-RBF with regularization was better than that without regularization, which means that the modified error function with regularization term do eliminated the impact of outliers included in the samples and helped to perform more accurate prediction.

V. CONCLUSIONS

In this paper, we proposed a RBF neural network model for outlier detection. The SC algorithm is used to initiate the hidden node centers in RBF network, which can lead to a proper network structure and achieve computation efficiency. In training procedure, a regularization term was added to error function to eliminate the effect of outliers. Finally, according to the degree of outlier for each input instance we can obtain the candidate outliers. Experimental result show that the SC-RBF based model is efficient and effective for outlier detection with lower false positive rate and higher detection rate.

ACKNOWLEDGMENT

This work is supported by the National Science & Technology Pillar Program (No. 2007BAH08B04) and the Chongqing University Postgraduates' Science and Innovation Fund (No. 200811A1B0080297).

REFERENCES

[1] C. C. Aggarwal and S. P. Yu, "An effective and efficient algorithm for high-dimensional outlier detection," *The VLDB Journal*, 2005, vol. 14, pp. 211–221.

[2] Z. A. Bakar, R. Mohemad, A. Ahmad and M. M. Deris, "A Comparative Study for Outlier Detection Techniques in Data Mining," *Proceedings of 2006 IEEE Conference on Cybernetics and Intelligent Systems*, 2006, pp. 1–6.

[3] S. S. Shirish and A. G. Ashok, "Use of Instance Typicality for Efficient Detection of Outliers with Neural Network Classifiers," *Proceedings of the 9th International Conference on Information Technology*, 2006.

[4] Y. Barnett and T. Lewis, "Outliers in Statistical Data," *John Wiley and Sons*, USA, 1994.

[5] J. Han and M. Kamber, "Data Mining Concepts and Techniques", *Morgan Kaufmann*, USA, 2006.

[6] E. Knorr and R. Ng, "Algorithms for mining distance-based outliers in large datasets," *Proceedings of the 24th Conference on VLDB*, New York, 1998, pp. 392–403.

[7] B. S. Everitt, "Cluster Analysis," *John Wiley and Sons*, USA, 2001.

[8] Y. Hwanjo, "SVMC: Single-class classification with support vector machines," *Proceedings of International Joint Conference on Artificial Intelligence*, 2003, pp. 567–572.

[9] T. L. Lin and L. Z. Jiu, "A RBF Neural Network Model for Anti-money Laundering," *2008 International Conference on Wavelet Analysis and Pattern Recognition*, 2008, pp. 209–215.

[10] S. L. Chiu, "Extracting fuzzy rules for pattern classification by cluster estimation," *Proceedings of the 6th International Fuzzy Systems Association World Congress*, 1995, pp. 1–4.

[11] M. Eftekhari and S. D. Katebi, "Extracting compact fuzzy rules for nonlinear system modeling using subtractive clustering, GA and unscented filter," *Applied Mathematical modeling*, 2008, pp. 2634–2651.

[12] S. Haralambos, A. Alex and B. George, "A fast training algorithm for RBF networks based on subtractive clustering," *Neurocomputing*, 2003, pp. 501–505.

[13] T. C. Lu, J. C. Juang and G. R. Yu, "On-line Outliers Detection by Neural Network with Quantum Evolutionary Algorithm," *Proceedings of International Conference on Innovative Computing, Information and Control, Kumamoto*, 2007.

[14] B. Karayiannis, "Reformulated Radial Basis Neural Networks Trained by Gradient Descent," *IEEE Transaction on Neural Networks*, VOL. 10, NO. 3, 1999, pp. 657–671.

[15] H. J. Lee and S. Cho, "Application of LVQ to novelty detection using outlier training data," *Pattern Recognition Letters 27*, 2006, pp. 1572–1579.

[16] http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[17] D. R. Wilson and T. R. Martinez, "Improved Heterogeneous Distance Functions," *Journal of Artificial Intelligence Research 6*, 1997.

[18] M. M. Breunig, H. P. Kriegel and R. T. Ng, "LOF: Identifying densitybased local outliers," *ACM Conference Proceedings*, 2000, pp. 93–104.

[19] E. M. Knorr, R. T. Ng and V. Tucakov, "Distance-based outliers: algorithms and applications," *The VLDB Journal*, 2000, vol. 8, pp. 237–253.

[20] J. Han, and M. Kamber, "Data Mining Concepts and Techniques," *Morgan Kaufmann*, USA, 2001.

[21] S. Ramaswamy, R. Rastogi and S. Kyuseok, "Efficient algorithms formining outliers from large data sets," *Proc. of the ACM SIGMOD International Conference on Management of Data, 2000*, pp. 93–104.

[22] M. F. Jiang, S. S. Tseng and C. M. Su, "Two-phase clustering process foroutlier detection," *Pattern recognition letters*, 2001, vol. 22(6-7), pp.691–700.

[23] F. Angiulli and C. Pizzuti, "Outlier Mining in Large High-Dimensional Data Sets," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 2, 2005, pp. 203–215.

Peng Yang Born in 1976. Received his M. A's degree from the School of Computer Science, Wuhan University of Technology in 2006. Ph. D. candidate in computing science from Chongqing University, China. Member of China Computer Federation. His main research interests include data mining and mobile network.

Qingsheng Zhu Born in 1956. Received his M. A's degree and Ph. D. degree from Chongqing University, China. Professor, doctoral supervisor and senior member of China Computer Federation. His main research interests include business intelligence and image processing.