Author: D. Shi, J. Gao, D. So Yeung and F. Chen
Title: Radial Basis Function Network Pruning by Sensitivity Analysis
Editor: A. Y. Tawfik, Goodwin, Scott D.
Conference Name: Advances in artificial intelligence, Canadian AI 2004, 17th conference of the Canadian Society for Computational Studies of Intelligence
Conference Location: Berlin/London
Publisher: Springer
Year: 2004
Volume: 3060/2004
Pages: 380-390
Date: 17-19 May 2004

Abstract: Radial basis function (RBF) neural networks have been extensively used for classification and regression due to the fact that they can provide fast linear algorithms to approximate any regular function. The most critical issue in the construction of an RBF network for a given task is to determine the total number of radial basis functions, their centers and widths. Conventional methods of training an RBF network are to specify the radial basis function centers by searching for the optimal cluster centers of the training examples. This paper proposes a novel learning algorithm for construction of radial basis function by sensitive vectors (SenV), to which the output is the most sensitive. Our experiments are conducted on four benchmark datasets, and the results show that our proposed SenV-RBF classifier outperforms conventional RBFs and achieves the same level of accuracy as support vector machine.

Author Address:
jbgao@csu.edu.au

URL:
http://books.google.com.au/books?id=7Dih_iPJNUwC&dq=17th+conference+of+the+Canadian+Society+for+Computational+Studies+of+Intelligence&source=gbs_navlinks_s

http://dx.doi.org/10.1007/b97823

http://springerlink.metapress.com/content/1kx7ddx6hctdpeev/fulltext.pdf

http://springerlink.metapress.com/content/4lwth9x80m3n/

http://www.informatik.uni-trier.de/~ley/db/conf/ai/ai2004.html

http://researchoutput.csu.edu.au/R/-?func=dbin-jump-full&amp;object_id=12481&amp;local_base=GEN01-CSU01

CRO identification number: 12481

Radial Basis Function Network Pruning by Sensitivity Analysis

Daming Shi[1], Junbin Gao[2], Daniel So Yeung[3], Fei Chen[1],

[1] School of Computer Engineering, Nanyang Technological University, Singapore 639798
asdmshi@ntu.edu.sg
[2] School of Mathematics, Statistics and Computer Science, University of New England, Armidale, NSW 2351, Australia
jbgao@mcs.une.edu.au
[3] Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong
csdaniel@comp.polyu.edu.hk

Abstract. Radial basis function (RBF) neural networks have been extensively used for classification and regression due to the fact that they can provide fast linear algorithms to approximate any regular function. The most critical issue in the construction of an RBF network for a given task is to determine the total number of radial basis functions, their centers and widths. Conventional methods of training an RBF network are to specify the radial basis function centers by searching for the optimal cluster centers of the training examples. This paper proposes a novel learning algorithm for construction of radial basis function by sensitive vectors (SenV), to which the output is the most sensitive. Our experiments are conducted on four benchmark datasets, and the results show that our proposed SenV-RBF classifier outperforms conventional RBFs and achieves the same level of accuracy as support vector machine.

1 Introduction

A radial basis function (RBF) network provides a fast, linear algorithm capable of representing complex non-linear mappings [2], and can approximate any regular function [16]. An RBF classifier is a three-layer neural network model, in which an N-dimensional input vector $\mathbf{x}=(x_1\ x_2\ \dots\ x_N)$ is broadcast to each of K neurons in the hidden layer. Each hidden neuron produces an activation function, typically a Gaussian kernel:

$$h_i = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{2\sigma_i^2}\right), \qquad i=1,2,\dots,K, \tag{1}$$

where $\mathbf{c}_i$ and σ_i^2 are the center and width of the Gaussian basis function of the *i*th hidden unit, respectively. The units in the output layer have interconnections with all the hidden units. The *j*th output neuron has the form:

$$f_j(\mathbf{x}) = \mathbf{w}_j \mathbf{h} = \sum_{i=1}^{K} w_{ij} \exp\left(-\frac{\left\| \mathbf{x} - \mathbf{c}_i \right\|^2}{2\sigma_i^2} \right), \tag{2}$$

where $\mathbf{h}=(h_1\ h_2\ \ldots\ h_K)$ is the input vector from the hidden layer, and the w_{ij} is the interconnection weight between the *j*th output neuron and the *i*th hidden neuron.

The most critical issue in the construction of an RBF network for a given task is to determine the total number of radial basis functions along with their centers and widths. There are two different ways to specify these hidden units. One way, explicated by Bishop [2], is to cluster the training examples and then assign a neuron to each cluster. The other way, after Poggio [19], is to learn mapping from a set of examples, that is, to create a neuron for each training example and then to prune the hidden neurons by example selection.

Moody and Darken [14] located optimal set of centers using both the *k-means* clustering algorithm and learning vector quantization. This method is based on locating the dense regions of the training inputs. The centers are the means of the vectors in such regions. The drawback of this method is that it takes into consideration only the distribution of the training inputs, yet the output values influence the positioning of the centers, especially when the variation of the output in a cluster is high.

Bishop [2] introduced the Expectation-Maximization (EM) algorithm to optimize the cluster centers. Initial centers are found by clustering before applying the EM algorithm. The EM approach includes an expectation part which calculates the likelihood of the model, and a maximization part which maximizes the likelihood of the data with respect to the centers and radii.

Chen *et al.* [5] proposed orthogonal least square (OLS) learning to determine the optimal centers. The OLS combines the orthogonal transform with the forward regression procedure to select model terms from a large candidate term set. The advantage of employing orthogonal transform is that the responses of the hidden layer neurons are decorrelated so that the contribution of individual candidate neurons to the approximation error reduction can be evaluated independently. To improve the generalization and global optimization abilities of the OLS, advanced versions can be seen in [6] and [7] respectively.

Orr [15] examined regularized forward selection of centers of RBF networks. He considered the variation in the output in an optimal sense with a set of training samples selected by forward subset selection, regularization, and cross validation. Zero-order regularization along with either delete-1 or generalized cross-validation was incorporated to perform the regularized forward selection. Starting with an empty subset, one gradually selects those centers, whose contribution to reducing the error is appreciably large. This method can produce the same result as OLS.

Hwang [10] clustered training example class by class to construct an RBF network. A statistical approach called linear discrimination function is used to determine the weights. Two specific methods can be applied to obtain the linear discrimination function. One uses least maximum square error minimization procedure and the other derives the linear discriminant function under the assumption that the training patterns form a normal distribution and all the covariance matrices are the same.

Panchapakesan *et al*. [17] investigated ways to train an RBF network with a set of centers selected by unsupervised methods and then to fine tune the locations of centers. This can be done by first evaluating whether moving the centers would decrease the error and then, depending on the required level of accuracy, changing the center locations. They gave out bounds on the gradient and Hessian of the error function. If a change in centers is desired, bounds on the Hessian may be used to fix a step size, which depends on the current centers to get a guaranteed amount of decrease in the error.

Mao [12] selects RBF centers based on Fisher ratio class separability measure with the objective of achieving maximum discriminative power. The OLS is employed to decouple the correlations among the responses of the hidden units so that the class separability provided by individual RBF neurons can be evaluated independently. This method can select a parsimonious network architecture as well as centers providing large class separation.

The common feature of all the above methods is that the radial basis function centers are a set of the optimal cluster centers of the training examples. Schokopf *et al.* [21] calculated *support vectors* using a support vector machine (SVM), and then used these support vectors as radial basis function centers. Their experimental results showed that the support-vector-based RBF outperforms conventional RBFs. Although the motivation of these researchers was to demonstrate the superior performance of a full support vector machine over either conventional or support-vector-based RBFs, their idea that the *critical* vectors (training examples), rather than cluster centers, should construct the RBF for a given classification task, is worth borrowing.

This paper proposes a novel approach to determining the centers of RBF networks based on sensitivity analysis. The remainder of this paper is organized as follows: In section 2, we describe the concepts of sensitivity analysis. In section 3, the most critical vectors are obtained by OLS in terms of sensitivity analysis. Section 4 contains our experiments and Section 5 offers our conclusions.

2 Sensitivity Analysis on Neural Networks

Sensitivity is initially investigated for the construction of a network prior to its design, since problems (such as weight perturbation, which is caused by machine imprecision and noisy input) significantly affect network training and generalization. Sensitivity analysis applied to network pruning seems particularly useful and valuable when network training involves a large amount of redundant data [28].

In 1990, Stevenson established the sensitivity of Madaline to weight error and derived an analytical expression for the probability of error in Madaline [22]. However, his results are only fit for neurons with threshold activation functions and binary inputs, and they are not applicable to other continuous activation functions. In 1992, Choi and Choi presented a thorough study on sensitivity analysis of the multilayer perceptron (MLP) with differentiable activation functions [8]. They defined the sensitivity as the variant of error over the variance of the weight perturbation. Because the weights are unknown prior to network training, this method cannot be used for network design and construction purpose.

In 1995, Piche systematically discussed the selection of weight accuracies for Madaline using a statistical approach to sensitivity analysis [18]. According to his stochastic model, all neurons have the same activation function. All network inputs, weights, input perturbations, and weight perturbations are random variables. In this model, the sensitivity of Madaline is defined as the noise-to-signal (NSR) of the output layer, the output NSR of a layer of threshold Adaline is:

$$NSR = \frac{\sigma_{\Delta y}^2}{\sigma_y^2} = \frac{4}{\pi}\sqrt{\frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2}} \tag{3}$$

where σ_y^2, σ_x^2, σ_w^2, $\sigma_{\Delta y}^2$, $\sigma_{\Delta x}^2$ and $\sigma_{\Delta w}^2$ refer to the variances of output *y*, inputs *x*, weights *w*, output error Δy, input perturbation Δx and weight perturbation Δw, respectively. Piche's stochastic model is not generally valid because: (1) All neurons in the same layer are assumed to have the same activation function, but this is not the case in some network models. (2) To satisfy the central limit theorem, the number of neurons in hidden layers is assumed to be large. (3) Weight perturbations are assumed to be very small, but this would be too restrictive for network training.

Yeung and Sun [25] generalized Piche (1995)'s work in two significant ways: (1) No restriction on input and output perturbation, which widens the application areas of sensitivity analysis; (2) The commonly--used activation functions are approximated by a general function expression whose coefficient will be involved in the sensitivity analysis. This treatment provides a way to sensitivity analysis on activation functions. In contrast to equation (3), the sensitivity of a threshold neuron in [25] forms:

$$S = \frac{D(\hat{y} - y)}{D(y)} = \frac{4}{\pi}\sqrt{\frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} + \frac{\sigma_{\Delta x}^2}{\sigma_x^2}\cdot\frac{\sigma_{\Delta w}^2}{\sigma_w^2}} \tag{4}$$

where σ_x^2, σ_w^2, $\sigma_{\Delta x}^2$ and $\sigma_{\Delta w}^2$ follow the definitions of equation (3). $D(\bullet)$ denotes variance, $\hat{y}$ and *y* denote output with and without perturbations respectively. It can be seen that, when $\sigma_{\Delta x}^2/\sigma_x^2$ and $\sigma_{\Delta w}^2/\sigma_w^2$ are very small, equation (3) and (4) are nearly the same. But when $\sigma_{\Delta x}^2/\sigma_x^2$ and $\sigma_{\Delta w}^2/\sigma_w^2$ are large, the results are different. For the threshold activation function, the output error variance cannot exceed 2. Equation (4) satisfies this, but equation (3) fails. So Yeung and Sun (2002)'s model is valid when the perturbation is either small or large.

Zeng and Yeung [26, 27] proposed a quantified measure and its computation for the sensitivity of the MLP to its input perturbation. A bottom-up approach was adopted. A single neuron is considered first, and algorithms with approximately derived analytical expressions that are functions of expected input deviation are given for the computation of its sensitivity. Then another algorithm is given to compute the sensitivity of the entire MLP network. Some applications of the MLP, such as improving error tolerance, measuring generalization ability, and pruning the network architecture, would benefit from their theoretical study. However, this method would be even more useful if it took into considerations correlation among training examples.

Some researchers studied how the real world data to produce uncertainty to neural networks. Although a different terminology is used, such an uncertainty analysis is actually the sensitivity analysis described above. Error bar estimates are usually used to allow confidence intervals to be placed around a model prediction. Wright [24] has discussed most of approaches used in dealing with the input uncertainty and proposed a Bayesian approach for this problem using the Markov Chain Monte Carlo method.

3 Sensitive Vector Learning to Construct RBF Networks

In this section, RBF classifier's sensitivity is defined as the mathematical expectation of the square of output deviations caused by the perturbation of RBF centers. An algorithm will be given that can be used to select critical vectors.

3.1 RBF Classifiers' Sensitivity to the Kernel Function Centers

We use symbols $\hat{\mathbf{c}}_i$ and $\hat{\sigma}_i$ to denote the values of center and width of the *i*th hidden neuron under a perturbation. Then the deviation resulted from this perturbation is:

$$\Delta y_j = \hat{\mathbf{w}}_j \hat{\mathbf{h}} - \mathbf{w}_j \mathbf{h} = \sum_{i=1}^{K} \hat{w}_{ij} \exp\left(-\frac{\left\| \mathbf{x} - \hat{\mathbf{c}}_i \right\|^2}{2\hat{\sigma}_i^2} \right) - \sum_{i=1}^{K} w_{ij} \exp\left(-\frac{\left\| \mathbf{x} - \mathbf{c}_i \right\|^2}{2\sigma_i^2} \right) \quad (5)$$

Here, $\hat{\mathbf{c}}_i = \mathbf{c}_i + \Delta \mathbf{c}_i$ are the centers deviated from the centers under the perturbations, and the interconnection weights under the perturbations are $\hat{\mathbf{w}}_j = \mathbf{w} + \Delta \mathbf{w}$, where **w** can be calculated using a pseudo matrix inversion, or data training. The most common method for selecting RBF widths is to make all of them equal, i.e., $\forall i$, $\sigma_i = \sigma$, and then set σ to a constant value depending on the prior knowledge of the given application. Since both the RBF widths affect classification errors but not the sensitivity of the hidden neurons, here we focus on the perturbations on the centers and their interconnection weights. The perturbation on the *i*th RBF center and the weights connected to the *j*th output, $\Delta\mathbf{c}_i$ and $\Delta\mathbf{w}_j$, can be generated following a Gaussian distribution with 0 means and variances $\sigma_{\mathbf{c}_i}$, $\sigma_{\mathbf{w}_j}$, respectively.

The RBF centers will be selected recursively in the next subsection. To make the sensitivity analysis cater for the construction of RBF networks, a recursive definition of sensitivity is given below. At the *K*th time, suppose there are a number $(K-1)$ of RBF centers fixed already, the newcomer $\mathbf{c}_i$ is observed. Hence, the *j*th output neuron's sensitivity to the current number *K* of RBF centers is defined as the mathematical expectation of $(\Delta y_j)^2$ (square of output deviations caused by the perturbations of RBF centers) with respect to all $\Delta\mathbf{c}_i$ and the training example set $D = \{\mathbf{x}_l\}_{l=1}^{L}$, which is expressed as

$$S_j^{(K)} = E[(\Delta y_j)^2] \tag{6}$$

The difference between sensitivity-based and conventional RBF networks can be illustrated in Fig. 1.

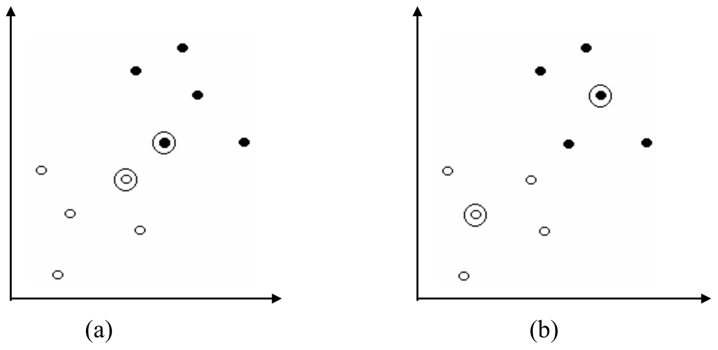

Fig. 1. Illustration of the difference between sensitivity-based and conventional RBF classifiers. The circles and balls represent data points of two classes respectively, and RBF centers are indicated by extra circles. (a) Sensitivity-based RBF, centers being the vectors sensitive to the classification. (b) Conventional RBF, centers being the cluster centroids.

3.2 Orthogonal Least Square Transform

The most critical vectors can be found by equation (7). However, the RBF centers cannot be determined by sensitivity measure only, because some of critical vectors may be correlated. The OLS [5] can alleviate this problem of redundant or correlated centers.

Let **Y**=($\boldsymbol{y}_1$, $\boldsymbol{y}_2$… $\boldsymbol{y}_L$)^T be the output matrix corresponding to all the number L of training examples, $\boldsymbol{y}_i$ (i=1,2,…,L) an M-dimensional vector, denoting number M of output units. We have

$$\mathbf{Y}=\mathbf{HW}=(\mathbf{QA})\mathbf{W} \tag{7}$$

Where **Y**, **H**, **W** are $L \times M, L \times L, L \times M$ matrices, respectively. The selection of RBF centers is equivalent to the selection of the most critical columns of **H**. The matrix **H** can be decomposed into **QA**, where **Q** is an $L \times L$ matrix with orthogonal columns [$\mathbf{q}_1$, $\mathbf{q}_2$, …, $\mathbf{q}_L$], and **A** is an $L \times L$ upper triangular matrix as follows:

$$\mathbf{H}=\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1L} \\ h_{12} & h_{12} & & \vdots \\ \vdots & & & \vdots \\ \vdots & & & \\ h_{1L} & \cdots & \cdots & h_{LL} \end{bmatrix}, \quad \mathbf{A}=\begin{bmatrix} 1 & a_{12} & \cdots & \cdots & a_{1L} \\ 0 & 1 & a_{23} & & \vdots \\ \vdots & 0 & \ddots & & \vdots \\ \vdots & & 0 & 1 & a_{(L-1)L} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}. \tag{8}$$

Only one column of **H** is orthogonalized. At the Kth time, the Kth column is made orthogonal to each of the K-1 previously orthogonalized columns and the operation for K=2, … , L is repeated. The computational procedure can be represented as follows [5]:

$$\begin{cases} \mathbf{q}_1 = \mathbf{h}_1, \\ a_{ik} = \dfrac{\mathbf{q}_i^T \mathbf{h}_K}{\|\mathbf{q}_i\|}, \qquad 1 \le i < K, \\ \mathbf{q}_K = \mathbf{h}_K - \sum_{i=1}^{K-1} a_{iK}\mathbf{h}_i \end{cases} \tag{9}$$

then the RBF centers are selected by sorting these columns.

3.3 Sensitive Vector Selection

Let $\mathbf{S}^{(K)}(\mathbf{c}_i)$ denote the sensitivity of the previous (K-1) RBF centers and a candidate RBF center $\mathbf{c}_i$ which is corresponding to $\mathbf{q}_i$ at the Kth time, where $1 \le i \le L$.

Substitute any interconnection weight in equations (5) and (7) by:

$$w_{ij}^{(K)} = \sum_{l=1}^{L} a_{li} \cdot w_{ij} \,, \tag{10}$$

and calculate the sensitivity for all the possible *K*-center RBF networks. Let $\mathbf{Q}^{(K)}$ denote the orthogonal matrix at the *K*th time, then the columns in $\mathbf{Q}^{(K)}$ are sorted in the order:

$$\left\|\mathbf{S}^{(K)}(\mathbf{c}_1)\right\| \geq \left\|\mathbf{S}^{(K)}(\mathbf{c}_2)\right\| \geq \cdots \geq \left\|\mathbf{S}^{(K)}(\mathbf{c}_L)\right\|. \tag{11}$$

A formal statement of the algorithm for the selection of critical vectors is given as follows:

STEP 1. Initialization. Form the matrix **H** in equation (8) by the RBF function responses of all the training examples.

STEP 2. First critical vector neuron selection. Calculate sensitivity of each column of **H** with equation (7). The column that provides the maximum sensitivity is selected as the first column of matrix $\mathbf{Q}^{(1)}$. Calculate the classification error $\mathbf{Err}^{(1)}$ with the selected RBF center.

STEP 3. Orthogonalization and critical vector selection. Let *K*=2.

STEP 4. Orthogonalize all remaining columns of **H** with all the columns of $\mathbf{Q}^{(K-1)}$ using equation (10).

STEP 5. Each training example, $\mathbf{c}_i$, is a candidate of the *K*th RBF center, which is corresponding to the orthogonalized column $\mathbf{q}_i$, $(K \leq i \leq L)$. Calculate interconnection weights using a pseudo matrix inversion and compute the sensitivity of the previous (*K*-1) RBF centers with each candidate center $\mathbf{S}^{(K)}(\mathbf{c}_i)$ with the weights updated by equation (11). Sort the columns in $\mathbf{Q}^{(K)}$ with equation (12), and the one yielding the maximum sensitivity is selected as the *K*th column of $\mathbf{Q}^{(K)}$. Calculate the classification error $\mathbf{Err}^{(K)}$ with the selected RBF centers.

STEP 6. If ($\mathbf{Err}^{(K)}$- $\mathbf{Err}^{(K-1)}$) is smaller than a predefined threshold, go to STEP 8.

STEP 7. *K*++, go to STEP 4.

STEP 8. End.

The critical vectors corresponding to he first *K* columns in $\mathbf{Q}^{(K)}$ will be selected as hidden neurons.

4 Experiments and Results

Our experiments were conducted on 2 datasets from UCI Repository [4], and 2 datasets from the Statlog collection [13]. The experiments are described as follows:

UCI Iris Plants (*iris*). This is the best--known database in the pattern recognition literature. The dataset contains 3 classes referring to 3 types of iris plant: Setosa, Versicolour and Virginica, respectively. One class is linearly separable from the others, but the other two classes are not linearly separable from each other. There are 150 instances (50 in each class), which are described by 4 attributes, sepal length, sepal width, petal length and petal width. 5-fold cross validation is conducted on the entire data set.

UCI Glass Identification (*glass*). This study is motivated by criminological investigation using the glass as evidence at the scene of the crime. There are 214 of instances from 6 classes. The 9 features include glass material information, such as sodium, silicon, or iron. 5-fold cross validation is conducted on the entire dataset and the best score is reported.

Statlog Satellite Images (*satimage*). There are 4435 training examples and 2000 testing examples in this dataset. Each instance consists of the pixel values in the 4 spectral bands of each of the 9 pixels in the 3×3 neighborhood, that are contained in a frame of a Landsat satellite image. The objective is to classify the central pixels into 6 attributes, such as red soil, or cotton crop.

Statlog Letters (*letter*). This dataset comprises a collection of 20,000 stimuli, consisting of 26 categories of capital English letters on different fonts. Each stimulus was converted into 16 primitive numerical attributes, such as statistical moments and edge counts. The first 15000 items are chosen to be training examples and remaining 5000 items to be testing examples.

We have compared the classification accuracies achieved with the proposed sensitivity vector RBF, support vector machine [11], conventional RBF [2], and decision tree C4.5 [20]. The training examples and testing examples for all the methods are the same as those chosen for our proposed SenV-RBF. The experimental results show that, data classification based on the proposed sensitivity-based RBF classifier performs better than the conventional RBF and the C4.5. The exception to this, *glass*, indicates that from case to case the Gaussian kernel function may have some blind spots. It can also be seen that both the sensitivity-based RBF and the SVM achieve basically the same level of accuracy, but the former enjoys the advantage of being easy to control and suitable for multiple-class problems, as well as being robust against noisy data. The advantages are accrued from the fact that the sensitivity-based RBF makes an effective trade-off between structural risk minimization and empirical risk minimization. The sensitivity-based RBF is time-consuming in training, but this is not a problem, as it provides a very fast and accurate classification in run-time, our area of concern.

5 Conclusion and Future Work

The conventional approach to constructing an RBF network is to search for the optimal cluster centers among the training examples. This paper proposes a novel approach to RBF construction that uses critical vectors selected by sensitivity analysis. Sensitivity is defined as the expectation of the square of output deviation caused by the perturbation of RBF centers. In training, orthogonal least square incorporated with a sensitivity measure is employed to search for the optimal critical vectors. In classification, the selected critical vectors will take on the role of the RBF centers. Our experimental results show that our proposed RBF classifier performs better than conventional RBFs and C4.5. The sensitivity-based RBF can achieve the same level of accuracy as SVM, but strikes a balance between structural risk minimization and empirical risk minimization. The advantage of the sensitivity-based RBF is that it is suitable for large scale multi-class applications, since it is easy to control as well as robust against noisy data.

In some cases, support vectors are exactly the same as sensitivity-oriented critical vectors, but the relationship between these two kinds of vectors remains unknown. Our future work will include investigating the relationship between sensitivity-based RBF and SVM in classification.

References

1. M. Bianchini, P. Frasconi and M. Gori, Learning without local minima in radial basis function networks, *IEEE Transactions on Neural Networks*, **6(3):**749-756, 1995.
2. C. M. Bishop, Improving the generalization properties of radial basis function neural networks, *Neural Computation*, **3(4):**579-581, 1991.
3. C. M. Bishop, Training with noise is equivalent to Tikhonov regularization, *Neural Computation*, **7(1):**108-116, 1995.
4. C. L. Blake, and C. J. Merz, UCI Repository of machine learning databases. *School of Information and Computer Science, University of California, Irvine, CA.* [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998
5. S. Chen, C. F. Cowan and P. M. Grant, Orthogonal least squares learning algorithms for radial basis function networks, *IEEE Transactions on Neural Networks*, **2(2):**302-309, 1991.
6. S. Chen, E. S. Chng and K. Alkadhimi, Regularized orthogonal least squares algorithm for constructing radial basis function networks. *International Journal of Control*, **64(5):**829-837, 1996.
7. S. Chen, Y. Wu and B. L. Luk, Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks, *IEEE Transactions on Neural Networks*, **10:**1239-1243, 1999.
8. J. Y. Choi and C. H. Choi, Sensitivity analysis of multilayer perceptron with differentiable activation functions, *IEEE Transactions on Neural Networks*, **3(1):**101-107, 1992, 1992.
9. V. Kadirkamanathan and M. Niranjan, A function estimation approach to sequential learning with neural networks, *Neural Computation*, **5(6):**954-975, 1993.
10. Y. S. Hwang and S. Y. Bang, An efficient method to construct a radial basis function neural network classifier, *Neural Networks*, **10(8):**1495-1503, 1997.
11. C. W. Hsu and C. J. Lin, A comparison of methods for multi-class support vector machines, *IEEE Transactions on Neural Networks*, **13(2):**415-425, 2002.
12. K. Z. Mao, RBF neural network center selection based on fisher ratio class separability measure, *IEEE Transactions on Neural Networks*, **13(5):**1211-1217, 2002.
13. D. Michie, D. J. Spiegelhalter and C. C. Taylor, Machine learning, neural and statistical classification. [Online]. Available: http://www.liacc.up.pt/ML/statlog/datasets.html, 1994.
14. J. Moody and C. J. Darken, Fast learning in networks of locally-tuned processing units, *Neural Computation*, **1:**281-294, 1989.

15.M. J. L. Orr, Regularization in the selection of radial basis function centers, *Neural Computation*, **7:**606-623, 1995.

16.J. Park, and I. W. Sandberg, Approximation and radial basis function networks, *Neural Computation*, **5:**305-316, 1993.

17.C. Panchapakesan, M. Palaniswami, D. Ralph and C. Manzie, Effects of moving the centers in an RBF network, *IEEE Transactions on Neural Networks*, **13(6):**1299-1307, 2002.

18.S. W. Piche, The selction of Weight Accuracies for Madalines, *IEEE Transactions on Neural Networks,* **6(2):**432-445, 1995.

19.T. Poggio and F. Girosi, Networks for approximation and learning. *Proceedings of the IEEE*, **78:**1481-1497, 1990.

20. J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

21.B. Scholkopf, K. K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik, Comparing support vector machines with Gaussian kernels to radial basis function classifiers, *IEEE Transactions on Signal Processing*, **45(11):**2758-2765, 1997.

22.M. Stevenson, R. Winter and B. Widrow, Sensitivity of feedforward neural networks to weight errors, *IEEE Transactions on Neural Networks,* 1(1):71-80, 1990.

23.Z. Wang, and T. Zhu, An efficient learning algorithm for improving generalization performance of radial basis function neural networks, *Neural Networks*, **13(4):**545-553, 2000.

24.W. Wright, Bayesian approach to neural network modeling with input uncertainty. *IEEE Transactions on Neural Networks*, **10(6):**1261-1270, 1999.

25.D. S. Yeung and X. Sun, Using function approximation to analyze the sensitivity of MLP with antisymmetric squashing activation function, *IEEE Transactions on Neural Networks*, **13(1):**34-44, 2002.

26.X. Zeng and D. S. Yeung, Sensitivity analysis of multilayer perceptron to input and weight perturbation, *IEEE Transactions on Neural Network*, **12(6):**1358-1366, 2001.

27.X. Zeng and D. S. Yeung, A quantified sensitivity measure for multilayer perceptron to input perturbation, *Neural Computation*, **15:**183-212, 2003.

28.J. M. Zurada, A. Malinowski and S. Usui, Perturbation method for deleting redundant inputs of perceptron networks. *Neurocomputing*, **14(2):**177-193, 1997.