Efficient Training of RBF Networks Via the BYY Automated Model Selection Learning Algorithms

Kai Huang, Le Wang, and Jinwen Ma*

Department of Information Science, School of Mathematical Sciences
And LMAM, Peking University, Beijing, 100871, China
jwma@math.pku.edu.cn

Abstract. Radial basis function (RBF) networks of Gaussian activation functions have been widely used in many applications due to its simplicity, robustness, good approximation and generalization ability, etc.. However, the training of such a RBF network is still a rather difficult task in the general case and the main crucial problem is how to select the number and locations of the hidden units appropriately. In this paper, we utilize a new kind of Bayesian Ying-Yang (BYY) automated model selection (AMS) learning algorithm to select the appropriate number and initial locations of the hidden units or Gaussians automatically for an input data set. It is demonstrated well by the experiments that this BYY-AMS training method is quite efficient and considerably outperforms the typical existing training methods on the training of RBF networks for both clustering analysis and nonlinear time series prediction.

1 Introduction

The radial basis function (RBF) networks [1]-[3] are a typical class of forward neural networks widely used in the fields of pattern recognition and signal processing. Clearly, it was developed from the approximation theory of radial basis functions for multivariate interpolation. That is, the RBFs are embedded in a two-layer neural network such that each hidden unit implements a radial basis function and the output units implement a weighted sum of hidden unit outputs. With such a structure, a RBF network can approximate any continuous function at a certain degree as long as the number of hidden units are large enough. Moreover, the structure of the input data can be appropriately matched via the selection of receptive fields of the radial basis functions with the hidden units, which leads to a good generalization of the RBF network. Therefore, the RBF network has been widely applied to various fields of pattern recognition and signal processing such as speech recognition, clustering analysis, time series prediction, industrial control etc., and the most commonly used radial basis functions in the RBF networks are Gaussian activation functions.

However, the training of a RBF network of Gaussian activation functions is still a rather difficult task. In fact, the main crucial problem is how to select the number and locations of the hidden units appropriately for a practical problem. In the previous

* The corresponding author.

D. Liu et al. (Eds.): ISNN 2007, Part I, LNCS 4491, pp. 1183–1192, 2007.

approaches, the number of hidden units was just the number of sample data and the locations of the hidden units were those sample data. However, it was proved that such a training or selection is expensive in terms of memory requirement. Moreover, the exact fit to the training set might cause a bad generalization. In order to overcome these problems, many training methods were proposed for training RBF networks and most of them utilized a test-and-growing or evolutionary approach to selecting the number of hidden units on the practical applications (e.g., [4]-[6]). Actually, a good selection of hidden units should appropriately match the structure of the input data associated with the practical problem. If the input data set consists of k clusters, the RBF network should select k hidden units with their locations being the centers of the k clusters, respectively. However, the selection of number of clusters for an input data set is also a difficult problem [7].

With development of competitive learning, the rival penalized competitive learning (RPCL) algorithm was proposed to determine the number of clusters or Gaussians in a sample data automatically [8]-[9]. Therefore, it has provided a new tool for the training of the RBF network (e.g., [10]-[11]). On the other hand, the scatter-based clustering (SBC) method [12] and the least biased fuzzy clustering method [13] were also proposed to determine the best number of hidden units in a RBF network.

Recently, based on the Bayesian Ying-Yang (BYY) harmony learning theory [14]-[16], a new kind of automated model selection (AMS) learning mechanism has been established for the Gaussian mixture modeling [17]-[19]. Actually, this kind of BYY-AMS learning rules can determine the number of Gaussians automatically during the parameter learning, which can be utilized to select the number of Gaussians as the hidden units in the RBF network.

In this paper, we utilize the BYY-AMS adaptive gradient learning algorithm [19] to select the appropriate number and initial locations of the Gaussians automatically on an input data set for the train of the RBF network. It is demonstrated by the experiments that this new training method is quite efficient and considerably outperforms some typical existing methods on the training of a RBF network for both clustering analysis and nonlinear time series prediction.

2 BYY-AMS Adaptive Gradient Learning Algorithm

We begin to introduce the adaptive gradient learning algorithm of automated model selection on the Gaussian mixture model proposed in [19] in the light of the BYY harmony learning theory. A BYY system describes each observation $x \in X \subset R^d$ and its corresponding inner representation $y \in Y \subset R^m$ via the two types of Bayesian decomposition of the joint density $p(x, y) = p(x)p(y \mid x)$ and $q(x, y) = q(x \mid y)q(y)$, being called Yang machine and Ying machine, respectively. For the Gaussian mixture modeling, y is only limited to be an integer variable, i.e., $y \in Y = \{1, 2, \cdots, K\} \subset R$ with m = 1. Given a data set $D_x = \{x_t\}_{t=1}^N$, the task of learning on a BYY system consists of specifying all the aspects of

$p(y \mid x)$, $p(x)$, $q(x \mid y)$, $q(y)$ via a harmony learning principle implemented by maximizing the functional:

$$H(p \| q) = \int p(y \mid x) p(x) \ln[q(x \mid y) q(y)] dx dy - \ln z_q, \tag{1}$$

where z_q is a regularization term.

If both $p(y \mid x)$ and $q(x \mid y)$ are parametric, i.e., from a family of probability densities with some parameter θ, the BYY system is called to have a Bi-directional Architecture (or BI-Architecture for short). For the Gaussian mixture modeling, we use the following specific BI-architecture of the BYY system. $q(y = j) = \alpha_j$ with $\alpha_j \geq 0$ and $\sum_{j=1}^{K} \alpha_j = 1$. Also, we ignore the regularization term z_q (i.e., set z_q =1) and let $p(x)$ be the empirical density $p(x) = \frac{1}{N} \sum_{t=1}^{N} \delta(x - x_t)$. Moreover, the BI-architecture is constructed with the following parametric form:

$$p(y = j \mid x) = p(j \mid x) = \frac{\alpha_j q(x \mid \theta_j)}{q(x, \Theta_K)} \quad , \quad q(x, \Theta_K) = \sum_{j=1}^{K} \alpha_j q(x \mid \theta_j) \ , \tag{2}$$

where $q(x \mid \theta_j) = q(x \mid y = j)$ with θ_j consisting of all its parameters and $\Theta_K = \{\alpha_j, \theta_j\}_{j=1}^{K}$ is the set of parameters for the finite mixture model.

Substituting these component densities into Eq. (1), we have

$$H(p \| q) = J(\Theta_K) = \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{K} \frac{\alpha_j q(x_t \mid \theta_j)}{\sum_{i=1}^{K} \alpha_j q(x_t \mid \theta_j)} \ln[\alpha_j q(x_t \mid \theta_j)]. \tag{3}$$

That is, $H(p \| q)$ becomes a harmony function $J(\Theta_K)$ on the parameters Θ_K . Furthermore, we let $q(x \mid \theta_j)$ be a Gaussian density given by

$$q(x \mid \theta_j) = q(x \mid m_j, \Sigma_j) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_j|^{\frac{1}{2}}} e^{\frac{1}{2}(x - m_j)^T \Sigma_j^{-1} (x - m_j)}, \tag{4}$$

where m_j is the mean vector and Σ_j is the covariance matrix which is assumed positive definite.

According to the harmony function given in Eq.(3), we can construct an adaptive gradient algorithm or rule to search for a maximum of $J(\Theta_K)$ as an estimate of the parameters Θ_K with the sample data set D_x . For convenience of derivation, we let

$$\alpha_j = \frac{e^{\beta_j}}{\sum_{i=1}^{K} e^{\beta_j}}, \qquad \Sigma_j = B_j B_j^T, \quad j = 1, 2, \cdots, K,$$

where $-\infty < \beta_1, \cdots, \beta_K < +\infty$ and B_j is a nonsingular square matrix. Via these transformations, the parameters in $J(\Theta_K)$ turn into $\{\beta_j, m_j, B_j\}_{j=1}^{K}$.

Denoting $U_j(x) = \alpha_j q(x \mid m_j, \Sigma_j)$ for $j = 1, \cdots, K$, $J(\Theta_K)$ has the following simple expression:

$$J(\Theta_K) = \frac{1}{N}\sum_{t=1}^{N} J_t(\Theta_K),\ J_t(\Theta_K) = \sum_{j=1}^{K} \frac{U_j(x_t)}{\sum_{i=1}^{K} U_i(x_t)} \ln U_j(x_t). \tag{5}$$

Furthermore, we have the derivatives of $J(\Theta_K)$ with respect to β_j, m_j and B_j, respectively, as follows.

$$\frac{\partial J_t(\Theta_K)}{\partial \beta_j} = \frac{1}{q(x_t \mid \Theta_k)} \sum_{i=1}^{K} \lambda_i(t)(\delta_{ij} - \alpha_j) U_i(x_t), \tag{6}$$

$$\frac{\partial J_t(\Theta_K)}{\partial m_j} = p(j \mid x_t)\lambda_j(t)\Sigma_j^{-1}(x_t - m_j), \tag{7}$$

$$vec[\frac{\partial J_t(\Theta_K)}{\partial B_j}] = \frac{\partial (B_j B_j^T)}{\partial B_j} vec[\frac{\partial J_t(\Theta_k)}{\partial \Sigma_j}], \tag{8}$$

where δ_{ij} is the Kronecker function, $vec[A]$ denotes the vector obtained by stacking the column vectors of the matrix A, and

$$\lambda_i(t) = 1 - \sum_{l=1}^{K} (p(l \mid x_t) - \delta_{il}) \ln[\alpha_l q(x_t \mid m_l, \Sigma_l)], \tag{9}$$

$$\frac{\partial J_t(\Theta_K)}{\partial \Sigma_j} = \frac{1}{2} p(j \mid x_t)\lambda_j(t)[\Sigma_j^{-1}(x_t - m_j)(x_t - m_j)^T \Sigma_j^{-1} - \Sigma_j^{-1}], \tag{10}$$

$$\frac{\partial (BB^T)}{\partial B} = I_{d\times d} \otimes B_{d\times d}^T + E_{d^2\times d^2} \bullet B_{d\times d}^T \otimes I_{d\times d},$$

where $\otimes$ denotes the Kronecker product (or tensor product), and

$$E_{d^2\times d^2} = \frac{\partial B^T}{\partial B} = (\Gamma_{ij})_{d^2\times d^2} = \begin{pmatrix} \Gamma_{11} & \cdots & \Gamma_{1d} \\ \vdots & \ddots & \vdots \\ \Gamma_{d1} & \cdots & \Gamma_{dd} \end{pmatrix}_{d^2\times d^2},$$

where Γ_{ij} is an d × d matrix whose $(j,i)^{th}$ element is just 1, with all the other elements being zero. With the above expression of $\frac{\partial (BB^T)}{\partial B}$, we have

$$vec[\frac{\partial J(\Theta_K)}{\partial B_j}] = \frac{1}{2} p(j \mid x_t)\lambda_j(t)(I_{d\times d} \otimes B_{d\times d}^T + E_{d^2\times d^2} \bullet B_{d\times d}^T \otimes I_{d\times d})$$

$$\times vec[\Sigma_j^{-1}(x_t - m_j)(x_t - m_j)^T \Sigma_j^{-1} - \Sigma_j^{-1}]. \tag{11}$$

Based on the above preparations, we have the adaptive gradient learning algorithm as follows.

$$\Delta\beta_j = \frac{\eta}{q(x_t \mid \Theta_k)} \sum_{i=1}^{K} \lambda_i(t)(\delta_{ij} - \alpha_j)U_i(x_t), \tag{12}$$

$$\Delta m_j = \eta\, p(j \mid x_t)\lambda_j(t)\Sigma_j^{-1}(x_t - m_j), \tag{13}$$

$$\Delta vecB_j = \frac{\eta}{2} p(j \mid x_t)\lambda_j(t)(I_{d\times d} \otimes B_{d\times d}^T + E_{d^2\times d^2} \bullet B_{d\times d}^T \otimes I_{d\times d})$$

$$\times vec[\Sigma_j^{-1}(x_t - m_j)(x_t - m_j)^T \Sigma_j^{-1} - \Sigma_j^{-1}], \tag{14}$$

where η denotes the learning rate that starts from a reasonable initial value and then reduces to zero with the iteration number n in such a way that $0 \le \eta(n) \le 1$, and

$$\lim_{n\to\infty} \eta(n) = 0 \ , \quad \sum_{n=1}^{\infty} \eta(n) = \infty, \tag{15}$$

i.e. , in the way used in the conventional stochastic approximation procedure [20]. The typical example of the learning rate satisfying Eq.(15) is $\eta(n) = \eta_0 / n$, where η_0 is a positive constant, which will be used in the following experiments.

This kind of BYY harmony learning can make model selection automatically on the Gaussian mixture model by forcing the mixing proportions of the extra Gaussians to be reduced to zero during the parameters learning as long as K is set to be larger than the number of actual Gaussians in the sample data. Actually, it had shown by the experiments in [19] that this BYY-AMS adaptive gradient learning algorithm can make model selection efficiently with a good estimate for the true parameters of the Gaussian mixture generating the sample data D_x .

3 Training of the RBF Network

We now consider the training of the RBF network via the above BYY-AMS adaptive gradient learning algorithm. In fact, the RBF network is just a two-layer forward neural network and its outputs are given by

$$y_j(x) = \sum_{j=1}^{n} w_{ij}\phi_i(x), \tag{16}$$

where n is the number of hidden units or RBF's, w_{ij} is the connection weight from the i^{th} hidden unit to the j^{th} output unit. $\phi_i(x)$ is just a Gaussian radial basis function (RBF) as the activation function corresponding to the output of the i^{th} hidden unit given by

$$\phi_j(x) = \phi_j(\| x - m_j \|) = \exp(-\frac{\| x - m_j \|^2}{2\sigma_j^2}), \tag{17}$$

where m_j, σ_j are the center and scale of the Gaussian RBF $\phi_j(x)$, respectively.

Without loss of generality, we just consider the case of the RBF network with one single output unit. In this special case, the output function of the RBF network takes a simple form as follows.

$$y(x) = \sum_{j=1}^{n} \lambda_j \phi_j(x) = \sum_{j=1}^{n} \lambda_j \exp(-\frac{\| x - m_j \|^2}{2\sigma_j^2}), \tag{18}$$

where λ_j is the connection weight from the j^{th} hidden unit or RBF to the output unit. Thus, the parameters of the RBF network are just $\lambda_j, m_j, \sigma_j (>0)$ for $j = 1, 2, \cdots, n$. Moreover, the mean square error of the RBF network on a sample set $D_{(x,y)} = \{(x_i, y_i) : i = 1, 2, \cdots, N\}$ can be given as follows.

$$\begin{aligned} E &= \frac{1}{2}\sum_{i=1}^{N}[y_i - f(x_i)]^2 = \frac{1}{2}\sum_{i=1}^{N}[y_i - \sum_{j=1}^{n}\lambda_j\phi_j(x_i)]^2 \\ &= \frac{1}{2}\sum_{i=1}^{N}[y_i - \sum_{j=1}^{n}\lambda_j \exp(-\frac{\left\| x_i - m_j \right\|^2}{2\sigma_j^2})]^2 \ . \end{aligned} \tag{19}$$

According to the least mean square error principle, we have the following learning rules on the parameters of the single-output-unit RBF network as follows:

$$\begin{cases} \Delta\lambda_j = \eta_\lambda \sum_{i=1}^{N}[y_i - \sum_{l=1}^{n}\lambda_l\phi_l(x_i)]\phi_j(x_i); \\ \Delta m_j = \eta_m \sum_{i=1}^{N}[y_i - \sum_{l=1}^{n}\lambda_l\phi_l(x_i)]\phi_j(x_i)(x_i - m_j)\lambda_j / \sigma_j^2; \\ \Delta\sigma_j = \eta_\sigma \sum_{i=1}^{N}[y_i - \sum_{l=1}^{n}\lambda_l\phi_l(x_l)]\phi_j(x_i)(x_i - m_j)^T(x_i - m_j)\lambda_j / \sigma_j^3, \end{cases} \tag{20}$$

where η_λ, η_m, η_σ are the learning rates for the updates of the parameters λ_j, m_j, σ_j, respectively, which are assumed to be invariant with the index j.

Usually, these learning rates are selected as some small positive constants by experience.

However, the LMS learning algorithm given by Eq.(20) has a major disadvantage that it is very sensitive to the selection of n and the initial values of the other parameters. Fortunately, the BYY-AMS adaptive gradient learning algorithm can be utilized to solve this sensitiveness problem. Actually, based on the input data set, the BYY-AMS adaptive gradient learning algorithm can determine an appropriate number of Gaussians, i.e., Gaussian RBF's, for the network. That is, we let $n = K^*$, where K^* is the number of actual Gaussians in the input sample data obtained by the the BYY-AMS adaptive gradient learning algorithm. Moreover, the final values of the mixing proportions and mean vectors can serve the initial values of the weights and centers of the n RBF's, respectively. And the initial value of σ_j can be set by

$$\sigma_j = \frac{1}{N_j} \sum_{x_t \in C_j} (x_t - m_j)^T (x_t - m_j), \tag{21}$$

where C_j is the set of the input sample set x_t with the maximum posteriori probability $p(j \mid x_t)$, N_j is the number of elements in C_j and m_j is the final value of the mean vector obtained by the BYY-AMS adaptive gradient learning algorithm. Augmented with the BYY-AMS adaptive gradient learning algorithm in this way, the LMS learning algorithm becomes very efficient on the training of the RBF network, which will be demonstrated by the experiments in the next section. For clarity, we refer to this compound training method just as the BYY-AMS training method for the RBF network.

4 Experiment Results

In this section, two kinds of experiments are carried out to demonstrate the efficiency of the BYY-AMS adaptive gradient learning algorithm on the training of a RBF network. Moreover, we compare the BYY-AMS training method with some other typical existing training methods.

4.1 On the Noisy XOR Problem

The noisy XOR problem [11] is a typical non-linear classification problem and we use a RBF network to learn it. The sample data are shown in Fig. 1 such that the sample points around the centers (1,0) and (-1,0) are in the first class and their outputs should be 1, while the sample points around the centers (0,1) and (0,-1) are in the second class and their outputs should be 0. For this problem, we generated 800 sample points totally, and 200 sample points per each center. We took 100 points per each center, and total 400 points to form the training set, and let the other 400 points be the test set. The actual outputs of the RBF network obtained by the BYY-AMS training method on the 400 test samples are given in Fig. 2.

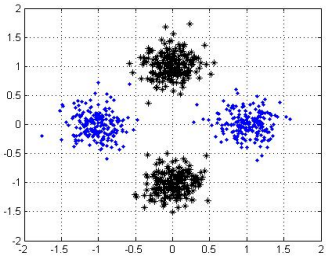

Fig. 1. The sample points of the noisy XOR problem

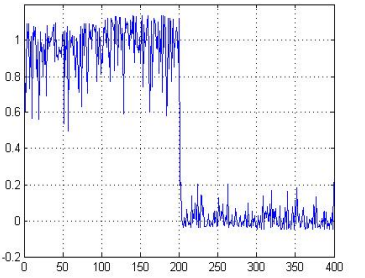

Fig. 2. The outputs of the trained RBF network on the test sample points

When we let the output of RBF network on an input point be processed via a threshold function such that if the output value of the network is over 0.5, the classification output is considered to be 1, otherwise, the classification output is considered to be 0, the correct classification rate of the trained RBF network on the test sample points can reach to 99.75%. However, in the same situation, the correct classification rates of the trained RBF networks with the RPCL and SBC methods can only reach to 97.5% and 97.75%, respectively.

4.2 On the Mackey-Glass Time Series Prediction

We further trained the RBF network with the help of the BYY-AMS adaptive gradient learning algorithm for time series prediction. As shown in Fig. 3, a piece of the Mackey-Glass time series was generated via the following iterative equation:

$$x(t+1) = (1-b)x(t) + \frac{ax(t-\tau)}{1+x(t-\tau)^{10}}, \tag{22}$$

where $a = 0.2, b = 0.1, \tau = 17$. Particularly, 1000 sample data were generated to form pieces of time series as $\{x(t-18), x(t-12), x(t-6), x(t), x(t+6)\}, 118 \le t \le 1117$, where the first four data were considered as an input data of the RBF network, while the last one was considered as the prediction result of the RBF network. Mathematically, the mapping relation behind the Mackey-Glass time series can be given as $y_i = f(x_i)$, where $x_i = [x(t-18), x(t-12), x(t-6), x(t)]^T$, $y_i = x(t+6)$, and $i = 1, \cdots, N$. In our experiment, we divided these 1000 sample data into two sets: the training and test sets with the preceding and remaining 500 sample data, respectively. The mean square error (MSE) was used to measure the prediction accuracy.

We implemented the BYY-AMS training method to train the RBF network for the prediction of this time series and the prediction result on the test data is given in Fig. 4, with the prediction mean square error 0.0033, which may be the lowest prediction error on the the Mackey-Glass time series. For comparison, we also implemented the least biased fuzzy clustering (LBFC) method to train the RBF network on the same data set and obtained the prediction result with the prediction mean square error as 0.2328, which is much greater than that of the RBF network via the BYY-AMS training method.

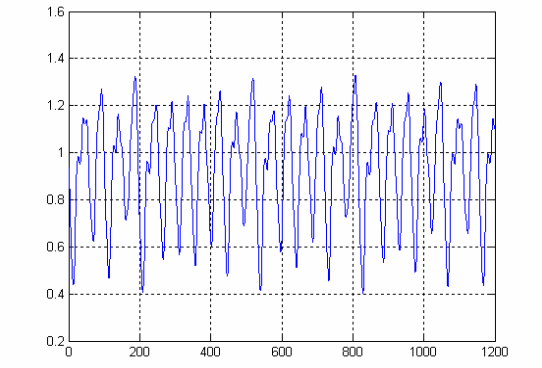

Fig. 3. The sketch of the piece of the Mackey-Glass time series

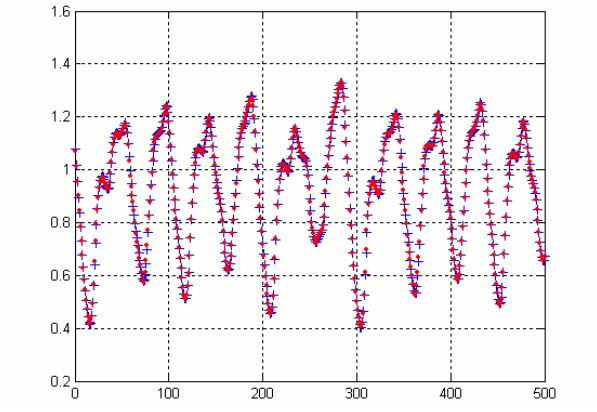

Fig. 4. The prediction result with the BYY-AMS adaptive gradient learning algorithm, where + represents the sample datum, while·represents the prediction datum

5 Conclusions

We have investigated the training of a RBF network with the help of a new kind of automated model selection learning algorithm based on the BYY harmony learning theory. Since this BYY-AMS learning algorithm can detect the structure of the input sample data, it can make the RBF network be more appropriate to a practical problem and improve the approximation and generalization or prediction abilities. The experimental results show that this BYY-AMS training method is very efficient and considerably outperforms the typical existing training methods on the training of a RBF network for clustering analysis and nonlinear time series prediction.

Acknowledgements

This work was supported by the Natural Science Foundation of China for Project 60471054.

References

1. Broomhead, D.S., Lowe, D.: Multivariable Functional Interpolation and Adaptive Networks. Complex System **2** (1988) 521-355
2. Moody, J., Darken, C.: Fast Learning in Networks of Locally Tuned Processing Units. Neural Computation **1** (1989) 281-294
3. Poggio, T., Girosi, F.: Regularization Algorithms for Learning that are Equivalent to Multiplayer Networks. Science **247** (1990) 978-982
4. Platt, J.: A Resource-allocating Network for Function Interpolation. Neural Computation **3** (1991) 213-225
5. Esposito, A., Marinaro, M., Oricchio, D., Scarpetta, S.: Approximation of Continuous and Discontinuous Mapping by a Growing Neural RBF-based Algorithm. Neural Networks **13** (2000) 651-665
6. Sanchez, A.V.D.: Searching for a Solution to the Automatic RBF Network Design Problem. Neurocomputing **42** (2002) 147-170
7. Hartigan, J. A.: Distribution Problems in Clustering. In J. Van Ryzin Editor, Classification and clustering, Academic Press, New York (1977) 45-72
8. Xu, L., Krzyzak, A., Oja, E.: Rival Penalized Competitive Learning for Clustering Analysis, Rbf Net and Curve Detection. IEEE Trans. on Neural Networks **4** (1993) 636–649
9. Ma, J., Wang, T.: A Cost-functionApproach to Rival Penalized Competitive Learning (RPCL). IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, **36** (4) (2006) 722-737
10. Krzyzak, A., Linder, T., Lugosi, G.: Nonparametric Estimation and Classification Using Radial Basis Function Nets and Empirical Risk Minimization. IEEE Trans. on Neural Networks, 7(1996): 475-487
11. Bors A. G. and Pitas I.: Median radial basis function neural network. IEEE Transactions on Neural Networks **7** (1996) 1351-1364
12. Sohn, I., Ansari, N.: Configure Rbf Neural Networks. Electronics Letters **34** (1998) 684- 685
13. Beni, G., Liu, X.: A Least Biased Fuzzy Clustering Method. IEEE Transactions On Pattern Analysis and Machine Intelligence **16** (1994) 954-960
14. Xu, L.: Ying-Yang Machine: A Bayesian Kullback Scheme for Unified Learning and New Results on Vector Quantization. Proceedings of International Conference on Neural Information Processing (ICONIP95) **2** 977-988
15. Xu, L.: A Unified Learning Scheme: Bayesian-Kullback Ying-Yang Machine. Advances in Neural Information Processing Systems 8 (1996) 444-450
16. Xu, L.: Best Harmony, Unified RPCL and Automated ModelSelection for Unsupervised and SupervisedLearning on Gaussian mixtures,Three-layer Nets and Me-rbf-svm Model. International Journal of Neural System **11** (2001) 43-69
17. Ma, J., Wang, T., Xu, L.: A Gradient BYY Harmony Learning Rule on Gaussian Mixture with Automated Model Selection. Neurocomputing **56** (2004) 481-487
18. Ma, J., Gao, B., Wang, Y., Cheng, Q.: Conjugate and Natural Gradient Rules for BYY Harmony Learning on Gaussian Mixture with Automated Model Selection. International Journal of Pattern Recognition and Artificial Intelligence **19** (2005) 701-713
19. Ma, J., Wang, L.: BYY Harmony Learning on Finite Mixture: Adaptive Gradient Implementation and a Floating RPCL Mechanism. Neural Processing Letters **24** (2006) 19-40
20. Robbins, H., Monro, S.: A Stochastic Approximation Method. Annals of Mathematical Statistics **22** (1951) 400-407