Agent-Based Population Learning Algorithm for RBF Network Tuning

Ireneusz Czarnowski and Piotr Jędrzejowicz

Department of Information Systems, Gdynia Maritime University
Morska 83, 81-225 Gdynia, Poland
{irek,pj}@am.gdynia.pl

Abstract. Radial Basis Function Neural Networks (RBFNs) are quite popular due to their ability to discover and approximate complex non-linear dependencies within the data under analysis. The performance of the RBF network depends on numerous factors. One of them is a value of the RBF shape parameter. This parameter has a direct impact on performance of the transfer function of each hidden unit. Values of the transfer function parameters, including the value of its shape, are set during the RBFN tuning phase. Setting values of the transfer function parameters, including its shape can be viewed as the optimization problem in which the performance of the considered RBFN is maximized. In the paper the agent-based population learning algorithm finding the optimal or near optimal value of the RBF shape parameter is proposed and evaluated.

1 Introduction

Artificial Neural Networks are used to solve many different kind of problems such as classification, signal processing, pattern recognition, prediction, time series analysis, image preprocessing, speaker identification, etc. The RBF networks are considered as an universal approximation tool similarly to the multilayer perceptrons (MLPs). However, radial basis function networks usually achieve faster convergence since only one layer of weights is required [10].

A RBF network is constructed from a three-layer architecture with a feedback. The input layer consisting of a set of source units connects the network to the environment. The hidden layer consists of hidden neurons with radial basis functions [10]. RBFNs use different functions at each hidden unit. Nevertheless, RBFN design is not straightforward. One of the main problems with neural networks is the lack of consensus on how to best implement them [18].

RBFNs are generally non-linear and belong to a special class of tools which performance depends on the distance between an input vector and a center vector, called centroid, prototype or kernel of the basis function. RBFNs ability is to approximate complex non-linear mapping directly from the input-output data [13]. The performance of the RBF network depends on numerous factors. The basic problem with the RBFNs is to set an appropriate number of radial basis function, i.e. a number of hidden units. Deciding on this number results in fixing the number of clusters and their centroids. Another factor, called a shape

L. Rutkowski et al. (Eds.): ICAISC 2013, Part I, LNAI 7894, pp. 41–51, 2013.

parameter of radial basis function, plays also an important role from the point of view of accuracy and stability of the RBF-based approximations. In numerous reported applications RBFs contain free shape parameters, which can be tuned by users. In [9] it is viewed as a disadvantage and somewhat ironic since the user is forced to make a decision on the choice of the shape parameter. Such an approach can also decrease chances to finding the optimal network structure [9].

In [11] it was suggested that the shape of radial basis functions should be changed depending on the data distribution. Such a flexibility should result in assuring better approximation effect in comparison with other approaches, where, for example, radial basis function parameters are set by some *ad hoc* criterion. A discussion of several approaches to setting shape parameter values can be found in [11].

In RBFNs the transfer function is represented by the radial basis function in each hidden unit. The transfer function is a composition of the activation function and the output function. A large number of different transfer functions have been proposed in the literature. Universal transfer functions have been proposed by Hoffmann [12]. Their feature is an ability to change shape smoothly from one function form to another. A taxonomy of different transfer functions used for neural network design can be found in [8]. Several possibilities of using transfer functions (i.e. activation and output functions) of different types in neural network models, including regularization of networks with heterogeneous nodes, are discussed in [8].

The paper deals with the problem of deciding on the RBF shape parameter values with a view to optimize transfer function design. It is shown how the agent-based population learning algorithm can be used for the RBF shape parameter setting through selection of transfer functions and their parameters. In [5] the agent-based population learning algorithm was used to locate only prototypes within the produced clusters. In the proposed extended version of the algorithm, firstly clusters are produced. Next, the prototypes are determined. In the second step the parameters of the output function for each hidden unit are also determined including the type of the transfer function with its shape.

The goal of the paper is to show through computational experiment that the agent-based population learning algorithm used to locate prototypes and to set values of parameters of the radial basis functions can be competitive in comparison with its earlier version presented in [5], as well as with other RBFN training algorithms. To validate the approach, an extensive computational experiment has been carried-out. Performance of the proposed algorithm has been evaluated using several benchmark datasets from the UCI repository [1].

The paper is organized as follows. Section 2 gives a basic account of the RBF networks. Idea of the agent-based population learning algorithm is presented in Section 3. Section 4 explains main features of the proposed implementation of the agent-based population learning algorithm. Section 5 provides details on the computational experiment setup and discusses its results. Finally, the last section contains conclusions and suggestions for future research.

2 RBF Neural Network Background

The output of the RBF network is a linear combination of the outputs of the hidden units, i.e. a linear combination of the nonlinear radial basis function generating approximation of the unknown function. In case of the classification problems the output value is produced using the sigmoid function with a linear combination of the outputs of the hidden units as an argument. In general, the RBFN output function has the following form:

$$f(x, w, p) = \sum_{i=1}^{M} w_i G_i(r_i, p_i), \tag{1}$$

where M defines the number of hidden neurons, G_i is a radial basis function associated with $i-$th hidden neuron, p_i is a vector of parameters, which can include the location of centroids, dispersion or other parameters describing the radial function.

One of the most popular output functions of the RBF hidden units is the Gaussian function [3], which has been chosen to best fit data from each cluster. In such a case the output function takes the following form:

$$G(r, b) = e^{-(\frac{r}{b})^2}, \tag{2}$$

where r is a norm function denoted as $r = \|x - c\|$, where x is an input instance, c represents a centroid and b is a value of dispersion (or "width") of the radial function. The output function of the RBF hidden unit most frequently is calculated using the Euclidean distance although other measures of distance can be also used. Thus, in general case, r refers to the Euclidean norm [9].

The Gaussian function is an example of function where the input instances are analyzed with respect to the one particular point (centroid) in the data space. The Gaussian function is also an example of a simple local function. In [8] it has been concluded that such local functions are useful to produce circular neurons and a solution especially for classification problems. In [8] it is also shown, that RBFN constructed using local functions can be very sensitive when the data are incomplete

Alternatively to the Gaussian function bicentral functions are considered to be a more promising option. The bicentral functions are formed from N pair of sigmoids and are defined with respect to two centers $c_i - e^{b_i}$ and $c_i + e^{b_i}$. The bicentral functions are window type localized functions and are separable. A feature of the bicentral transfer functions is their flexibility in representing various probability densities. They can also produce decision region with convex shapes. These properties result from possibilities of moving the location of two centers, changing the function dispersion and setting a slope of the function [8]. The bicentral output function has the following form:

$$G(x, c, b, s) = \prod_{i}^{N} \sigma(e^{s_i}(x_i - c_i + e^{b_i}))(1 - \sigma(e^{s_i'}(x_i - c_i - e^{b_i}))), \tag{3}$$

where c represents a centroid, b is a corresponding width of the radial function, N is the dimension of the instance (i.e. the number of attributes), s and s' represent a slope of the function for the left and the right side in any dimension of the centroid, σ is a sigmoid function with argument z defined as follows: $\sigma(z) = \frac{1}{1+e^{-\beta z}}$, where β determines the slope of the function and is equal to s or s' respectively.

The RBF network initialization is a process, where the set of parameters of the radial basis functions, including the number of the radial basis functions, their shapes and the number of centroids with their locations, needs to be calculated or drawn. It is performed during the RBFN tuning. On the other hand, RBFNs involve finding a set of weights of links between neurons such that the network generates a desired output signals. The weights are determined in the RBF network training process. Both processes can be also viewed as solving the optimization task, where the optimization objective is to minimize the value of the target function by finding the optimal values of vector weights and vector of RBF parameters.

Since the RBF neural network initialization and training belong to the class of computationally difficult combinatorial optimization problems [10], it is reasonable to apply to solve this task one of the known metaheuristics. In this paper the agent-based population learning, proposed originally in [2], is applied as a collaborative approach to neural network tuning (see, for example [15]). In the paper the agent-based population learning algorithm is proposed for the purpose of the RBFN initialization including prototype selection and choice of shape parameters. In next sections details of the proposed approach are included.

3 Agent-Based Population Learning Algorithm

In [2]it has been shown that agent-based population learning search can be used as a robust and powerful optimizing technique. In the agent-based population learning implementation both - optimization and improvement procedures are executed by a set of agents cooperating and exchanging information within an asynchronous team of agents (A-Team). The A-Team concept was originally introduced in [15].

The concept of the A-Team was motivated by several approaches like blackboard systems and evolutionary algorithms, which have proven to be able to successfully solve some difficult combinatorial optimization problems. Within an A-Team agents achieve an implicit cooperation by sharing a population of solutions, to the problem to be solved.

An A-Team can be also defined as a set of agents and a set of memories, forming a network in which every agent remains in a closed loop. Each agent possesses some problem-solving skills and each memory contains a population of temporary solutions to the problem at hand. It also means that such an architecture can deal with several searches conducted in parallel. In each iteration of the process of searching for the best solution agents cooperate to construct,

find and improve solutions which are read from the shared, common memory. All agents can work asynchronously and in parallel.

Main functionality of the agent-based population learning approach includes organizing and conducting the process of search for the best solution. It involves a sequence of the following steps:

- Generation of the initial population of solutions to be stored in the common memory.
- Activation of optimizing agents which execute some solution improvement algorithms applied to solutions drawn from the common memory and, subsequently, store them back after the attempted improvement in accordance with a user defined replacement strategy.
- Continuation of the reading-improving-replacing cycle until a stopping criterion is met. Such a criterion can be defined either or both as a predefined number of iterations or a limiting time period during which optimizing agents do not manage to improve the current best solution. After computation has been stopped the best solution achieved so far is accepted as the final one.

More information on the population learning algorithm with optimization procedures implemented as agents within an asynchronous team of agents (A-Team) can be found in [2]. In [2] also several A-Team implementations are described.

4 An Approach to the RBF Network Tuning

The paper deals with the problem of RBFN initialization through applying the agent-based population learning algorithm. The main goal is to find the optimal set of RBF network parameters with respect to:

- Producing clusters and determining their centroids.
- Determining the kind of transfer function for each hidden units and other parameters of the transfer function.

4.1 Producing Clusters and Determining Their Centroids

Under the proposed approach clusters are produced at the first stage of the initialization process. They are generated using the procedure based on the similarity coefficient calculated as proposed in [6]. Clusters contain instances with identical similarity coefficient and the number of clusters is determined by the value of the similarity coefficient. Thus the clusters are initialized automatically, which also means that the number of radial basis function is initialized automatically (for details see, for example, [6]).

Next from thus obtained clusters of instances centroids are selected. An agent-based algorithm with a dedicated set of agents is used to locate centroids within clusters. In the proposed approach, it is assumed that maximum two centroids can be selected from each cluster. Obviously, from clusters containing exactly one instance, only one centroid can be selected.

4.2 Determining the Kind of Transfer Function

Under the proposed approach the number of cluster centroids determines the kind of transfer function associated with a given hidden unit. When only one centroid is selected, the output of the RBF hidden unit is calculated using the Gaussian function (see equation no. 2). By introducing this condition it is assumed that such hidden have a circular shape of the transfer function.

When the number of selected centroids is greater than one the output of the RBF hidden unit is calculated using the bicentral function (see equation no. 3). In this case the algorithm uses a dedicated set of agents responsible for finding optimal values for the left and the right slope of the transfer function.

The proposed approach may result in producing a heterogenous function network.

4.3 Agent-Based Population Learning Algorithm Implementation

The main feature of the proposed agent-based population learning algorithm is its ability to select centroids and transfer function parameters in cooperation between agents. Most important assumptions behind the approach, can be summarized as follows:

- Shared memory of the A-Team is used to store a population of solutions to the RBFN initialization problem.
- A solution is represented by a string consisting of two parts. The first contains integers representing numbers of instances selected as centroids. The length of the first part of the string is equal to, at least, the number of clusters (i.e. the number of hidden units) and can be greater than the number of hidden units when one of the clusters is associated with two centroids. The detailed conditions on the centroid number have been introduced in subsection 4.2. The second part consists of real numbers for representing left and right slope of the transfer functions. This part contains $2N$ parameters per one hidden unit.
- The initial population is generated randomly.
- Initially, potential solutions are generated through randomly selecting one or two centroids from each of the considered clusters.
- Initially, the real numbers representing slopes are generated randomly.
- Each solution from the population is evaluated and the value of its fitness is calculated. The evaluation is carried out by estimating classification accuracy or error approximation of the RBFN, which is initialized using centroids, set of transfer function parameter indicated by the solution and trained using backpropagation algorithm.

The RBFN initialization problem is solved using two groups of optimizing agents. The first group includes agents executing procedures for centroid selection. These procedures are a local search with the tabu for prototype selection and a simple local search. The both procedures modify a solution by replacing a randomly

selected instance with some other randomly chosen instance thus far not included within the improved solution, or by modification through adding or removing an instance from the improved solution. The only difference between them is that in the first procedure the replacing takes place only for instances which are not on the tabu list. After the replacement, the move is placed on the tabu list and remains there for a given number of iterations. Replacement and modification are performed randomly with the same probability equal to 0.5.

The second group of optimizing agents includes procedures for estimation of the slope parameters. One of them is the standard mutation which modifies the second part of a solution by generating new values of element in the string. The modification is carried out with a mutation rate of p_m. If the fitness function value has improved then the change is accepted. The second procedure is an application of the non-uniform mutation. The non-uniform mutation acts through modifying a solution by repeatedly adjusting value of the randomly selected element in the string until the fitness function value has improved or until k consecutive improvements have been attempted unsuccessfully. The value of the adjustment is calculated as:

$$\triangle(t', y) = y(1 - q^{(1-\frac{t'}{2N'})q}), \tag{4}$$

where q is the uniformly distributed real number from $(0, 1]$, N' is equal to the length of the current string with values representing the slop of the transfer function and t' is a current number of adjustment. The mutation is performed with probability p_{mu}. Both mutation procedures have been successfully applied in [4].

5 Computational Experiment

This section contains the results of several computational experiments carried out with a view to evaluate the performance of the proposed approach. In particular, the reported experiments aimed at evaluating quality of the RBF-based classifiers constructed using the proposed approach. Experiments aimed at answering the question whether the proposed agent-based approach to RBF network tuning ($ABRBF\_Tuning$) performs better than classical methods of RBFN initialization? The proposed approach has been also compared with the earlier version of the approach called $ABRBFN$ 1 introduced in [5], where the agent-based population learning algorithm has been used only to perform search for a location of centroids within each of the Gaussian kernel-based clusters.

In the reported experiments the following RBFN initialization approaches have been also compared:

- The k-*means* clustering with the agent-based population learning algorithm used to locate prototypes (in this case at the first stage the k-*means* clustering has been implemented and next, from thus obtained clusters, the prototypes have been selected using the agent-based population learning algorithm) - denoted as k-$meansABRBFN$.

- The k-*means* algorithm used to locate centroids for each of the Gaussian kernels (in this case at the first stage the k-*means* clustering has been implemented and the cluster centers have been used as prototypes) - denoted as k-*means*$RBFN$.
- The random search for kernel selection - denoted as *random*$RBFN$.

Evaluation of the proposed approaches and performance comparisons are based on the classification and the regression problems. For both cases the proposed algorithms have been applied to solve respective problems using several benchmark datasets obtained from the UCI Machine Learning Repository [1]. Basic characteristics of these datasets are shown in Table 1.

Each benchmark problem has been solved 50 times, and the experiment plan involved 10 repetitions of the 10-cross-validation scheme. The reported values of the quality measure have been averaged over all runs. The quality measure in case of the classification problems was the correct classification ratio - accuracy (Acc). The overall performance for regression problems has been computed by the mean squared error (MSE) calculated as the approximation error over the test set.

Parameter settings for computations involving $ABRBF\_Tuning$ are shown in Table 2. Values of the some parameters have been set arbitrarily in the trials and errors procedure.

Table 1. Datasets used in the reported experiment

Dataset	Type of problem	Number of instances	Number of attributes	Number of classes	Best reported results
Forest Fires	Regression	517	12	-	-
Housing	Regression	506	14	-	-
WBC	Classification	699	9	2	97.5% [1] (Acc.)
Credit	Classification	690	15	2	86.9% [1] (Acc.)
Sonar	Classification	208	60	2	97.1% [1] (Acc.)
Satellite	Classification	6435	36	6	-

Table 2. Parameter settings for $ABRBF\_Tuning$ in the reported experiment

Parameter	
Max number of iteration during the search	500
Max number of epoch reached in RBF network training	1000
Population size	60
Probability of mutation for the standard and non-uniform mutation (p_m, p_{mu})	20%
Range values for left and right slope of the transfer function	[-1,1]

The dispersion of the Radial function has been determined as suggested in [15]: $b_i = \min_{k,i=1,\ldots,m;k\neq i}\{\|c_k - c_i\|\}$, where c_k and c_i are centers of clusters.

Table 3 depicts the performance comparison involving *ABRBF_Tuning*, its earlier version (*ABRBFN 1*) and some other approaches to RBF initialization including the k-*means* clustering with the agent-based population learning algorithm. From the results it can be observed that the proposed algorithm assures competitive results in comparison to other approaches. The proposed approach to the RBF network tuning proves to be quite competitive in case of the regression problems. In case of the classification problem the *ABRBF_Tuning* has improved accuracy only for one dataset. In three cases the *ABRBF_Tuning* algorithm is not better than *ABRBFN 1*. The experiment results show that *ABRBF_Tuning* applied to the RBF initialization performs better than k-*meansABRBFN*, k-*meansRBFN* and *randomRBFN*. Only for one dataset the k-*meansABRBFN* produced better results.

The results in Table 3 further demonstrate that the *ABRBF_Tuning* can be superior to the other methods including MLP, Multiple linear regression, SVM and C4.5. This statement is supported by the fact that in seven cases the proposed algorithm has been capable to improve the generalization ability.

Table 3. Results obtained for different variants of the proposed algorithm applied to the task of the RBNF's training and their comparison with performance of several different competitive approaches

Problem:	Forest fires	Housing	WBC	Credit	Sonar	Satellite
Algorithm:	MSE		Acc. (%)			
ABRBF_Tuning	**2.07**	**34.92**	94.24	84.05	**83.34**	83.32
ABRBFN 1 [5]	2.15	35.24	94.56	**84.56**	82.09	**85.05***
k-*meansABRBFN* [5]	2.29	35.87	**95.83**	84.16	81.15	83.57*
k-*meansRBFN* [5]	2.21	36.4	93.9	82.03	78.62	81.4*
randomRBFN [5]	3.41	47.84	84.92	77.5	72.79	74.84*
Neural network - MLP	2.11 [19]	40.62 [19]	96.7 [7]	84.6 [7]	**84.5** [7]	83.75 [14]
Multiple linear regression	2.38 [19]	36.26 [19]	-	-	-	-
SVR/SVM	**1.97** [19]	44.91 [19]	**96.9** [7]	84.8 [7]	76.9 [7]	85.0 [17]
C 4.5	-	-	94.7 [7]	**85.5** [7]	76.9 [7]	-

* Not present in [5].

6 Conclusions

In this paper the agent-based population learning algorithm for RBF neural network tuning is proposed. The task of the algorithm is to find optimal parameters of the transfer function including the type of the function and its shape, and the appropriate centroids within initialized clusters for each hidden units of the RBF network.

Important feature of the approach is that the number of clusters, location of centroids and the transfer function parameters are determined in parallel using a set of dedicated agents. In the reported computational experiment the proposed algorithm has proved to be not worse from the earlier its version and in some cases outperforms other techniques for RBF initialization.

Future research will focus on finding more effective configurations of the RBF networks by extending the approach adding ability to estimate output weights of the RBFN. A new set of optimizing agents is planned to be implemented. It is also planned to carry-out more refined statistical analysis of the results to obtain a better insight into properties of the proposed approach.

In the future it also is planned to implement the proposed agent-based population learning algorithm for construction of the cascade correlation neural network. It is believed that that selection of the most promising transfer function for each candidate unit from a pool of candidates by the agent-based population learning algorithm can bring benefits in term of the classification accuracy or the approximation error.

Acknowledgement. This research has been supported by the Polish Ministry of Science and Higher Education with grant no. N N519 576438 for years 2010-2013.

References

1. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
2. Barbucha, D., Czarnowski, I., Jędrzejowicz, P., Ratajczak-Ropel, E., Wierzbowska, I.: e-JABAT - An Implementation of the Web-Based A-Team. In: Nguyen, N.T., Jain, I.C. (eds.) Intel. Agents in the Evol. of Web and Appl. SCI, vol. 167, pp. 57–86. Springer, Heidelberg (2009)
3. Broomhead, D.S., Lowe, D.: Multivariable Functional Interpolation and Adaptive Networks. Complex Systems 2, 321–355 (1988)
4. Czarnowski, I., Jędrzejowicz, P.: An agent-based approach to ANN training. Knowledge-Based Systems 19, 304–308 (2006)
5. Czarnowski, I., Jędrzejowicz, P.: An Approach to Cluster Initialization for RBF Networks. In: Graña, M., Toro., C., Posada, J., Howlett, R., Jain, L.C. (eds.) Advances in Knowledge-Based and Intelligent Information and Engineering Systems. Frontiers in Artificial Intelligence and Applications, vol. 243, pp. 1151–1160. IOS Press (2012)
6. Czarnowski, I.: Cluster-based Instance Selection for Machine Classification. Knowledge and Information Systems 30(1), 113–133 (2012)
7. Datasets used for classification: comparison of results. In. directory of data sets, `http://www.is.umk.pl/projects/datasets.html` (accessed September 1, 2009)
8. Duch, W., Jankowski, N.: Transfer Functions: Hidden Possibilities for Better Neural Networks. In: Proceedings of the 9th European Symposium on Artificial Neural Networks (ESANN), Brugge, pp. 81–94 (2001)
9. Fasshauer, G.E., Zhang, J.G.: On Choosing "Optimal" Shape Parameters for RBF Approximation. Numerical Algorithms 45(1-4), 345–368 (2007)

10. Gao, H., Feng, B., Hou, Y., Zhu, L.: Training RBF Neural Network with Hybrid Particle Swarm Optimization. In: Wang, J., Yi, Z., Żurada, J.M., Lu, B.-L., Yin, H. (eds.) ISNN 2006. LNCS, vol. 3971, pp. 577–583. Springer, Heidelberg (2006)
11. Hanrahan, G.: Artificial Neural Networks in Biological and Envoronmental Analysis. Analytical Chemistry Series. CRC Press, Taylor & Francis Group (2011)
12. Hoffmann, G.A.: Adaptive Transfer Functions in Radial Basis Function (RBF) Networks. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3037, pp. 682–686. Springer, Heidelberg (2004)
13. Huang, G.-B., Saratchandra, P., Sundararajan, N.: A Generalized Growing and Pruning RBF(GGAP-RBF) Neural Network for Function Approximation. IEEE Transactions on Neural Networks 16(1), 57–67 (2005)
14. Liang, N.-Y., Huang, G.-B., Saratchandran, P., Sundararajan, N.: A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. IEEE Transactions on Neural Networks 17(6), 1411–1423 (2006)
15. Sánchez, A.V.D.: Searching for a solution to the automatic RBF network design problem. Neurocomputing 42(1-4), 147–170
16. Talukdar, S., Baerentzen, L., Gove, A., de Souza, P.: Asynchronous Teams: Cooperation Schemes for Autonomous, Computer-Based Agents. Technical Report EDRC 18-59-96, Carnegie Mellon University, Pittsburgh (1996)
17. Wang, L., Yang, B., Chen, Y., Abraham, A., Sun, H., Chen, Z., Wang, H.: Improvement of Neural Network Classifier Using Floating Centroids. Knowledge Information Systems 31, 433–454 (2012)
18. Yonaba, H., Anctil, F., Fortin, V.: Comparing Sigmoid Transfer Functions for Neural Network Multistep Ahead Streamflow Forecasting. Journal of Hydrologic Engineering 15(4), 275–283 (2010)
19. Zhang, D., Tian, Y., Zhang, P.: Kernel-based Nonparametric Regression Method. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp. 410–413 (2008)