ADJUSTING THE PARAMETERS OF WRBF NETWORK FOR FUNCTION APPROXIMATION USING CMA-ES

*Javad Mohebbi Najm Abad*¹ and Ali Soleimani² and Mohammad Reza Mohebi³

¹Department of Computer Engineering, Quchan Branch, Islamic Azad University, Quchan, Iran
²Department of Computer Engineering, Shahrood University of Technology, Shahrood, Iran
³Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran

*Author for Correspondence

ABSTRACT

In this paper, we investigate the implementation of CMA-ES and some evolutionary algorithms such as PSO, ICA and GA in the design of Weighted Radial Basis Function (WRBF) neural networks to solve regression problems. WRBF have been employed in pattern classification. Since RBF network usually is used for function approximation, we studied WRBF in this direction. In WRBF some weights are set between input layer and hidden layer to increase networks performance. The back propagation learning algorithm for neural network training might converge to a set of sub-optimal Network parameters from which it cannot escape. The problem in hand is optimizing of radial-basis functions centers, spreads and the weights of the network. Experimental results show that using WRBF network is more efficient than RBF network in regression problems. Moreover, training WRBF network with CMA-ES outperform than training with conventional methods or some evolutionary algorithms in terms of time and results on a set of test functions.

Keywords: RBF, PSO, ICA, GA, WRBF, Optimization, Neural Networks Training

INTRODUCTION

Radial Basis Function (RBF) networks are a popular class of feed forward fully connected Artificial Neural Networks (ANN) which were introduced by Broomhead and Lowe (Broomhead et al., 1988). Due to their good approximation capabilities, simple network structure and computational efficiency they are considered as a good alternative to Multi-Layer Perceptron (MLP) networks and have been used in many scientific and engineering fields (Xiaopeng et al., 2012; Lijun et al., 2012). Basic form of an RBF network consists of three layers with entirely different roles (George et al., 2012). First layer is a passive layer which is called input layer. Second layer is the hidden layer, which is the only nonlinearity in the network which employs radial-basis functions as hidden unit’s activation function. Output layer is the third layer which performs a linear mapping (Figure 1). The activation function of i-th hidden unit is defined as

\[ \phi_i(x) = \phi(||x - \mu_i||, \sigma_i) \]  

Figure 1: Architecture of an RBF neural network
Where $\phi$ is a radial-basis function which is radically symmetric and has a unique maximum at $\mu_i$ called center. This function’s value decreases rapidly for far distances from the center. Decrease rate of the function is controlled by the spread parameter $\sigma_i$. The RBF function takes considerable values only when the distance between center and input $x_i = \| x - \mu_i \|$, is smaller than $\sigma_i$.

Gaussian function is one the most commonly used basis functions in RBF networks which is defined as

$$\phi_i(x) = \exp\left(-\frac{\| x - \mu_i \|^2}{2\sigma_i^2}\right)$$

(2)

where $\|\| \|$ denotes the Euclidian norm (George et al., 2012).

Training of an RBF network consists of finding centers and spreads of the hidden layer and the weights that connect hidden layer to output layer.

Suitable selection of centers and spread of the radial-basis functions has a great influence on the generalization error of an RBF network. There are four different strategies for training this type of neural network depending on how the centers of the radial-basis functions of the network specified (George et al., 2012).

A. Fixed centers selected at random

In this method, locations of the centers of the radial-basis functions are randomly chosen from the samples of the training set. It is noteworthy that in this method the samples of training set should be good representatives of the input space to this method have a good performance. After choosing the centers, spread of the radial-basis functions are set to fixed values. For example in this method Gaussian functions are defined as

$$\phi_i(x) = \exp\left(-\frac{m}{d\max^2} \| x - \mu_i \|\right)$$

(3)

where $m$ is the number of chosen centers and $d\max$ is the maximum distance between selected centers.

As can be seen in Equation (1) spread of the Gaussian function is set to the $\sigma = d\max / \sqrt{2m}$ fixed value. The only parameter that needs to be learned in this method is the weights of the output layer. As we know

$$YU = D$$

(4)

where $U$ and $D$ are matrix form of the output layer’s weights and desired outputs of the training set respectively and

$$Y = \{ y_{ji} \}$$

(5)

$$y_{ji} = \exp\left(-\frac{m}{d\max^2} \| x_j - \mu_i \|\right), j = 1 ... n, i = 1 ... m$$

(6)

where $x_j$ is j-th training sample.

$U$ can be found as

$$U = Y^+ D$$

(7)

Where $Y^+$ is the pseudo inverse of matrix $Y$ which can be calculated using Singular Value Decomposition (SVD). As mentioned before, this method’s performance extremely depends on well positioning of the centers of the radial-basis functions and most of the times it has poor generalization.

B. Self-organized Selection of Centers

This method employs a two stage hybrid learning process. First stage uses an unsupervised learning process such as K-means (Simon et al., 1999; Moody et al., 1999). Or self-organizing map (Hossein et al., 2012) neural network to find suitable locations for the centers of the radial-basis functions. In second stage a supervised learning method is employed to learn the weights of the output layer.

C. Strict Interpolation with Regularization

It is a method for designing RBF networks that combines elements of regularization theory and kernel regression estimation theory. More information about this method can be found in (George et al., 2012).

D. Supervised Selection of Centers

In this method, all parameters of the network including centers and spreads of the networks and weights of the output layer are determined through a supervised learning process such as gradient descent. Usually this method has a good generalization performance.
MATERIALS AND METHODS

Weighted RBF Network

In the basic form of RBF network, only centers and spreads of radial-basis functions and the weights of output layer can be learned and input layer has a passive role. Weighted RBF (WRBF) introduces a weight vector, \( W \), between the input and hidden layer, so they can do some feature mapping and help network to perform better (Hossein et al., 2012). \( W \) is applied on input vector and performs a linear mapping (Figure 2). For example in this method Gaussian activation function is changed as follows:

\[
\phi_i(x) = \exp \left( - \frac{\|W_i x - \mu_i\|^2}{2\sigma_i^2} \right)
\]

Where \( W_i = [w_{i1}, w_{i2}, \ldots, w_{in}] \) is the weight vector between input vector and \( i \)-th hidden unit. This vector performs some feature mapping and helps the network’s performance. This network is trained using back-propagation algorithm and it has superior performance compared to traditional RBF neural network (Hossein et al., 2012).

In this paper WRBF neural network is used for function approximation. Estimating high dimension function with back-propagation algorithm is not practical. For these problems, parameters are determined by some evolutionary algorithms such as PSO (Esmaeili et al., 2009). To authors’ best knowledge, no prior attempts have been made to consider adjusting parameters of WRBF neural network for function approximation by using evolutionary algorithms and strategies. In the other direction, we want to compare some usual evolutionary algorithms with CMA-ES for determining WRBF parameters.

The rest of this paper is organized as follows. PSO, ICA and GA optimization algorithms briefly describe in section 3. CMA-ES is introduced in section 4. Section 5 describes our proposed methods. Section 7 reports analysis results of the simulation on three UCI data sets and finally paper concludes is the section 8.

Evolutionary Algorithms

In this section some popular and valuable optimization algorithms, which are investigated in this work, briefly introduced.

A. Particle Swarm Optimization

Particle Swarm Optimization (PSO) which introduced by Kennedy et al., (1995), is a form of swarm intelligence which mimics biological behavior of folks of birds or a school of fish. When a swarm looks for food, its individuals spread in the environment and look for food independently from each other. When an individual finds a food source, announces the location of the food to other individuals so they can move toward it. PSO tries to simulate this behavior to solve optimization problems.

In PSO every solution for the optimization problem is an individual in search space which is called “particle”. PSO starts with a population of randomly initialized particles and tries to produce better
solutions in each generation. Every particle has a velocity vector which states the next location of the particle. In each generation, velocity vector of a particle is updated according to three vectors. First one is its current value, second one is the best position that the particle has found so far (pbest) and third one is the best position of the entire population that has been reached till current generation (gbest). After finding the pbest and gbest, position of every particle is updated as

\[
v_{ij}^{t+1} = w(t)v_{ij}^t + c_1r_1(p_{i,pbest}^t - x_{ij}^t) + c_2r_2(p_{i,gbest}^t - x_{ij}^t) \quad (9)
\]

\[
x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1} \quad (10)
\]

Where \( t \) is generation number, \( w(t) \) is a coefficient called inertia factor, \( r_1 \) and \( r_2 \) are random numbers uniformly driven from interval \([0,1]\), \( c_1 \) and \( c_2 \) are positive constants called self-recognition constant and social constant respectively. \( x_{ij}^t \) is \( j \)-th dimension of \( i \)-th particle, \( x_{i,pbest}^t \) is \( j \)-th dimension of pbest and \( x_{i,gbest}^t \) is \( j \)-th dimension of gbest.

For increasing exploration, in each iteration, mutation operator for some individual could be used. Mutation change small part of particle randomly.

**B. Imperilist Competitive Algorithm**

Imperialist Competitive Algorithm (ICA) is a rather new evolutionary optimization algorithm based of socio-political behavior of humans (Atashpaz-Gargari et al., 2007). Like the other evolutionary algorithms, this algorithm starts with a randomly initialized population of solutions called countries. In ICA, some of the best countries are selected as the imperialists and the others are set to be their colonies. Number of colonies of each imperialist is proportional to its power. Consequently the most powerful imperialist will have the most number of colonies and vice versa. In an N dimensional search space a country is defined as:

\[
country = [p_1, p_2, ..., p_N] \quad (11)
\]

After initialization some steps are done in each generation. First step is the absorption process. In this step, colonies are absorbed toward their imperialist. Absorption policy is the core step of the ICA and makes countries to move toward optimum value. To improve search ability of the algorithm, an amount of deviation, \( \theta \), is added to the direction of colony movement towards the imperialist. \( \theta \) is randomly derived from uniform distribution i.e.

\[
\theta \sim U(-\gamma, \gamma) \quad (12)
\]

where \( U \) shows the uniform distribution and \( \gamma \) is constant value usually equal to \( \frac{\pi}{4} \).
Absorption policy makes the colonies to move $x$ units toward their imperialist which is randomly derived from interval $[0 ... \beta \times d]$ i.e.
\[
x \sim U(0, \beta \times d)
\] (13)
where $d$ is the distance between colony and its imperialist and $\beta$ is constant usually equal to 2. After absorption step, revolution operator is applied. Revolution which is very similar to mutation in Genetic Algorithm increases diversity between countries. This operator randomly selects some colonies and changes their position randomly. After revolution and absorption, a colony may reach a better position.

In the next and last step of each generation imperialist compete for the weakest colony of weakest imperialist. In this direction, firstly total cost of each imperialist is calculated as
\[
T.C_n = \text{cost}(\text{imperialist}_n) + \xi \text{mean}(\text{cost}(\text{colonies of imperialist}_n))
\] (14)
After calculating the total cost of imperialists, possession of the weakest colony of weakest imperialist randomly is changed to another imperialist with a probability proportional to its total cost.

C. Genetic Algorithm

Genetic algorithms (GAs) are well suited for searching global optimal values in complex search space (multi-modal, non-linear, discontinuous, and highly constrained space). Genetic algorithm belong to the class of Evolutionary Algorithm (EA), which generate solution to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection and crossover. Initially many individual solutions are randomly generated to form an initial population. For evolution some generation happen until a termination condition has been reached. During each generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness based process. Then a new population of solutions generate from those selected through crossover and/or mutation. For producing each new solution, a pair of parent solution is selected for reproduction from the pool selected previously. Then with using the above methods of mutation and crossover, a new solution is created which typically shares many of characteristics of its parents.

D. Simulated Annealing

**SA for single objective**

1. Initialize the temperature.
2. Start with a randomly generated initial solution vector, $X$, and generate the objective function.
3. Give a random perturbation and generate a new solution vector, $Y$, in the neighbourhood of current solution vector, $X$, reevaluate the objective function and apply penalty function approach to the objective function, if necessary.
4. If the generated solution vector is archived, make it the current solution vector by putting $X = Y$. Update the existing optimal solution and go to Step 6.
5. Else accept $Y$ with the probability:
\[
P = \exp(-\Delta s / T)
\]
where $\Delta s = Z(Y) - Z(X)$.
If the solution is accepted, replace $X$ with $Y$.
6. Decrease the temperature periodically.
7. Repeat Steps 2–6 until stopping criterion is met.

---

**Figure 4: Pseudo-code for SA**
Simulated annealing (SA) is a compact and robust technique with a substantial reduction in computation time, which provides excellent solutions to optimization problems (Suman et al., 2006). It is based on an analogy of thermodynamics with the way metals cool and anneal. If a liquid metal is cooled slowly, its atoms form a pure crystal corresponding to the state of minimum energy for the metal. The metal reaches a state with higher energy if it is cooled quickly. SA is one of the many evolutionary algorithms which designed to give a good, not necessarily optimal solution. It is simple to formulate and can be used in continuous and discrete problems. The pseudo code of SA is illustrated in figure 4.

**Covariance Matrix Adaptation-evolution Strategy (cma-es)**

The CMA-ES is a stochastic method for real parameter optimization of non-linear, non-convex functions. At each iteration, a population of candidate solution is sampled. The initial population is generated by sampling a multivariate normal distribution. One of the most important properties of CMA-ES is its invariance (Hansen et al., 2000; Igel et al., 2007). CMA-ES has two method of derandomized and cumulation. Distribution change dramatically in derandomized mutation based on successful search steps. By using information from successful search steps, covariance matrix mutation distribution modifies and the probability of successful step increase. Cumulation information from previous information is used in self-adaptation by considering the search-path of population. The algorithm derived from intuitive concept and from requirements of non-linear, non-convex search in continuous domain is as follow.

**Step 1: Set Parameters and Initialization**

Parameters, which are including population size $\lambda$, parent number $\mu$, recombination weights $\omega_i=1\cdots\mu$, learning rate for the cumulation for the step size control $c_\sigma$, damping parameters $d_\sigma$, learning rate for the cumulation for the rank-one update $c_c$, $c_1$ and $c_{\mu}$, set to their default values from Hansen 2011 (Hansen et al., 2011). Evolution paths $p_\sigma$, $p_c$, covariance matrix $C=1$, and iteration number $g$ set to their default values too. In the other direction, distribution mean $m$ and step size $\sigma$ is chosen based on population and parameters domain.

**Step 2: Evolution Loop**

Evolution loop is stop while one of the termination criterions is met. In each iteration, new populations of search points are generated by following formula:

For $k = 1,\cdots,\lambda$

1. $z_k \sim \mathcal{N}(0,1)$
2. $y_k = BDz_k \sim \mathcal{N}(0,C)$
3. $x_k = m + \sigma y_k \sim \mathcal{N}(m,\sigma^2 C)$

Selection and recombination is as follow:

$$\langle y \rangle_w = \sum_{i=1}^{\mu} \omega_i y_{i,\lambda} \quad \text{where} \quad \sum_{i=1}^{\mu} \omega_i = 1, \quad \omega_i > 0$$

$$m \leftarrow m + \sigma \langle y \rangle_w = \sum_{i=1}^{\mu} \omega_i x_{i,\lambda}$$

Step size is controlled by:

$$p_\sigma^{(g+1)} \leftarrow (1 - c_\sigma) p_\sigma^{(g)} + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}^c} \langle y \rangle_w$$

$$\sigma^{(g+1)} \leftarrow \sigma^{(g)} \exp \left( \frac{C_\sigma}{d_\sigma} \left( \frac{\|p_\sigma^{(g+1)}\|}{E(\|\mathcal{N}(0,1)\|)} - 1 \right) \right)$$

And covariance matrix is adapted by:

$$p_c^{(g+1)} \leftarrow (1 - c_c) p_c^{(g)} + H_\sigma^{(g+1)} \sqrt{c_c (2 - c_c) \mu_{\text{eff}}} \langle y \rangle_w$$

© Copyright 2014 | Centre for Info Bio Technology (CIBTech)
Proposed Training Algorithms for WRBF

RBF network is a special case of neural network, which is originally used for regression problems, such as function approximation. In the other direction, in recent decades, RBF network is used for classification problems (Wang et al., 2002; Chen et al., 2008; Meng et al., 2010). In this direction WRBF network is proposed that provide valuable result in comparing with RBF network (Hossein et al., 2012).

Gradient decent is used for training weights, centers and spreads of WRBF network. In this paper, we use WRBF network for regression problem. WRBF network has more sophisticated architecture rather than RBF network because of the weights between input and hidden layers. These weights increase dimensions in optimization problem. As a result, training WRBF network has been done by some evolutionary algorithms and strategies which are proper for high dimension functions.

To train a WRBF network with evolutionary optimization, initially we should specify the networks encoding. As mentioned before, the weights between input and hidden layer, centers and spreads of radial-basis functions and output weights are tunable parameters of WRBF. So we used the following direct encoding

| $w_i$, $\sigma_i$, $\mu_i$, $u_i$ | ..., ..., $w_n$, $\sigma_n$, $\mu_n$, $u_n$ |

where $w_i$, $\sigma_i$, $\mu_i$, and $u_i$ are the weights between input $i$-th hidden unit, center of $i$-th hidden unit, spread of $i$-th hidden unit and output weight of $i$-th hidden unit respectively.

RMSE is defined as fitness function to evaluate the individuals in Evolutionary algorithms and CMA-ES:

$$RMSE = \left( \frac{1}{N} \sum_{i=1}^{N} ||t_i - o_i|| \right)^{\frac{1}{2}}$$

(24)

Hence the searching space for high dimension is too large; it is more likely that PSO, ICA and other evolutionary algorithm converge to local optima. There are some improvements on the basis algorithms to escape from local optima. For example in the PSO, we used mutation operator as a solution to this problem. Mutation improves exploring capability of the algorithms and increases diversity between individuals. In each generation this operator randomly with probability $p$ Mutation selects some of the particles and changes their position with following equation:

$$x^j = x^j + \sigma N(0, \alpha)$$

where $x^j$ is the $j$-th dimension of selected particle and $\sigma$, $\alpha$ are constant values.

In ICA, every time that algorithm cannot improve the cost for a predefined number of iterations, we add a new imperialist. This new imperialist selects some of the colonies of other imperialists as his colonies randomly proportional to their costs.

Number of population in CMA-ES is impressive in the probability of finding global optima especially in multimodal problems. When algorithm converges to local optima in early iteration, algorithm restarts with new population that their number is more than previous population.

RESULTS AND DISCUSSION

Table 1: Characteristics of used data sets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Number of instances</th>
<th>Number of features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concrete Comp. Strength(CCT)</td>
<td>1030</td>
<td>9</td>
</tr>
<tr>
<td>Concrete Slump Test(CST)</td>
<td>103</td>
<td>10</td>
</tr>
<tr>
<td>Wine Quality(WQ)</td>
<td>4898</td>
<td>12</td>
</tr>
<tr>
<td>Housing(H)</td>
<td>506</td>
<td>13</td>
</tr>
</tbody>
</table>
We used four UCI (Blake et al.,) regression data sets to experimentally evaluate our proposed methods. The characteristics of these data sets are listed in table 1. We have compared our proposed method with some evolutionary algorithms and CMA-ES method. These methods have been introduced in previous parts.

We implemented all of algorithms in Matlab environment. For each data set and each method 10 simulation were performed. In each simulation we split the data set in two training (70%) and test (30%) parts.

Figure 4, 5 and 6 compared using ICA, PSO and GA for training RBF and WRBF on CCT dataset in one simulation.

Table 2 and 3 illustrate comparing results of all evaluated evolutionary algorithms and CMA-ES in adjusting WRBF parameters. Table values are the mean value of ten times algorithm results. In all evaluated data sets CMA-ES has the best results in function approximation. In the other direction, run time of CMA-ES is lower than many evolutionary algorithms such as GA, PSO and ICA. PSO, population base SA and ABC have appropriate accuracy but their run time is more than CMA-ES. Simple PSO and SA have short run time but their results are not suitable for selected problems. As a result, Improved PSO and SA with longer runtime have used in compression.

Training RBF and WRBF networks are compared for function approximation on UCI data sets. As can see in Figure 4, 5 and 6, using WRBF for CCT dataset has more accuracy rather than RBF network. Increased accuracy is happened in all evaluated UCI datasets. Vertical axis is the minimum RMSE in population and horizontal axis shows iteration.
Table 2: Using Evolutionary algorithms and CMA-ES on adjusting WRBF parameters on wine quality dataset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RMSE</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td>7.906</td>
<td>116.4193</td>
</tr>
<tr>
<td>BA</td>
<td>7.7049</td>
<td>806.0164</td>
</tr>
<tr>
<td>SA</td>
<td>7.6504</td>
<td>31.1492</td>
</tr>
<tr>
<td>PSO</td>
<td>7.4692</td>
<td>700.6954</td>
</tr>
<tr>
<td>Population base SA</td>
<td>7.3981</td>
<td>1543.7698</td>
</tr>
<tr>
<td>ABC</td>
<td>7.1413</td>
<td>180.5373</td>
</tr>
<tr>
<td>CMA-ES</td>
<td>5.2975</td>
<td>66.8028</td>
</tr>
</tbody>
</table>

Table 3: Using Evolutionary algorithms and CMA-ES on adjusting WRBF parameters on Housing dataset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RMSE</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td>0.80076</td>
<td>191.3438</td>
</tr>
<tr>
<td>BA</td>
<td>0.79909</td>
<td>1116.201</td>
</tr>
<tr>
<td>SA</td>
<td>0.80814</td>
<td>86.8133</td>
</tr>
<tr>
<td>PSO</td>
<td>0.77735</td>
<td>182.4084</td>
</tr>
<tr>
<td>Population base SA</td>
<td>0.80237</td>
<td>2651.4186</td>
</tr>
<tr>
<td>ABC</td>
<td>0.77868</td>
<td>349.4308</td>
</tr>
<tr>
<td>CMA-ES</td>
<td>0.70793</td>
<td>174.4078</td>
</tr>
</tbody>
</table>

Figure 8: Adjusting WRBF parameters on Wine Quality dataset

Figure 9: Adjusting WRBF parameters on Housing dataset

Figure 5 and 6 show comparing results of all evaluated evolutionary algorithms and CMA-ES in adjusting WRBF parameters on Wine Quality and Housing data sets respectively. Vertical axis is the minimum RMSE in population and horizontal axis shows iteration.

Conclusion

This paper proposed WRBF network for regression problems. In addition, some evolutionary algorithms have compared with CMA-ES in adjusting WRBF parameters which is an optimization problem. In order to increase exploration, mutation operator is used in the PSO. In this direction in the ICA, when algorithm cannot improve the cost for a predefined number of iterations, a new imperialist is created. For increasing
accuracy and quality of SA algorithm, population base SA has been used. The experimental results on different benchmark regression problems from UCI repository revealed the effectiveness and efficiency of the CMA-ES in determining the parameters of WRBF network. In addition, experimental results show using WRBF instead of RBF is more efficient for regression problems. The results show that on four datasets WRBF outperforms RBF network.

ACKNOWLEDGEMENT
We are grateful to Islamic Azad University, Quchan branch authorities, for their useful collaboration.

REFERENCES