ISSN (e): 2250 – 3005 || Volume, 05 || Issue, 06 || June – 2015 ||
International Journal of Computational Engineering Research (IJCER)

PSO-based Training, Pruning, and Ensembling of Extreme Learning Machine RBF Networks

Praveen Malali[1] and Miltiadis Kotinis[2]
[1]Department of Mechanical and Aerospace Engineering
Old Dominion University, Norfolk, VA 23509

ABSTRACT:
The research presented in this article focuses on the development of a methodology for ensembling radial basis function (RBF) networks that have been trained using particle swarm optimization (PSO) and the extreme learning machine (ELM). PSO is used to find optimal values for the basis width and the coordinates of the kernel centers, while ELM provides the values of the network connection weights. The ensemble consists of RBF networks that correspond to the personal best positions found by the swarm particles during the search process. The swarm intelligence search mechanism is supplemented with a mutation operator, which incorporates substitution of the worst performing particles by the best performing particle, after the latter has been mutated. Pruning of the input layer of the RBF networks is also implemented in the algorithm. The generalization performance of the PSO-ELM algorithm is evaluated by applying it to a number of widely-utilized regression and time series prediction benchmark problems. The results reveal that the proposed methodology is very effective even when small RBF networks are utilized.

KEYWORDS: *Radial basis function, network extreme learning machine, particle swarm optimization, ensembling pruning.*

I. INTRODUCTION

The extreme learning machine (ELM) is a fast machine learning algorithm utilized for the training of single-hidden-layer feed forward neural networks (SLFNs) [1–3]. It was developed as an alternative to gradient-based learning algorithms, e.g., back-propagation, in order to accelerate the training of the network, provide good generalization performance by obtaining the smallest norm of the connection weights, and also obviate the need for timeconsuming algorithmic parameter tuning [1]. Various ELM-based algorithms have been proposed over the last few years [4,5] in an attempt to reduce the typically high number of hidden nodes required by the ELM due to the random determination of the connection weights between input and hidden layer. Furthermore, the ELM has been combined with evolutionary algorithms [6] in order to evolve the network parameters in tandem with the connection weights.

Radial basis function (RBF) networks [7, 8] are a particular type of SLFNs, which has been used extensively for function approximation and time series prediction. RBF networks are universal approximators [8], i.e., given a sufficiently large number of hidden layer nodes they can be trained to approximate any real multivariate continuous function on a finite data set. An RBF network utilizes a radial basis kernel in each hidden node in order to obtain accurate local, relative to the kernel center, approximations of the unknown function. The Gaussian and the inverse multiquadric kernels, which are radially symmetric and bounded, are frequently used as basis functions in RBF networks. The output of the network is obtained through a linear combination of the hidden nodes' output.

A comparison between the performance of an ELM-based RBF network and a support vector regression (SVR) algorithm in a very small number of regression problems is presented in [9]. The two methods have comparable performance in terms of approximation accuracy, but the ELM-based RBF network requires a significantly shorter time for training. Given that the kernel centers and basis widths are selected randomly in the aforementioned ELM-based methodology, the algorithmic performance would most likely improve via a less random selection scheme; however, such a scheme should not mitigate the major advantage of ELMs, i.e., the fast training of the network. Furthermore, as shown in [10], the performance of an RBF network in a number of time series prediction problems strongly depends on the choice of kernel function, number of hidden nodes, and basis width values.

The training of artificial neural networks (ANNs), including RBF networks, using evolutionary algorithms has been an active area of research during the last fifteen years. Evolutionary algorithms have been employed in order to evolve the network connection weights [11,12], the location of the kernel centers of an RBF network [13], and also to evolve basis width values, location of kernel centers, and connection weights simultaneously [14]. The determination of the values of the network connection weights in tandem with the network architecture has also been investigated in [15-17]. Finally, evolutionary multi-objective optimization algorithms have been employed in order to generate ensembles of neural networks and/or learning machines [18-21].

The main advantage of using stochastic evolutionary algorithms for the network training over traditional, gradient-based algorithms is the inherent capability of the former to minimize the risk of getting trapped in locally optimal values during the search/training process. Furthermore, most evolutionary algorithms are population-based, i.e., perform multiple parallel searches during a single run; this enables them to explore different regions of the decision variable space simultaneously and through the utilization of appropriate mechanisms to transmit search-related information across the population. In this work, PSO and the ELM are combined in order to develop an algorithm that generates ensembles of RBF networks. The generalization error of an ensemble of networks/learners is equal to the weighted average of the generalization error of the individual networks minus the ensemble ambiguity [22]; the later quantifies the diversity within the ensemble. Therefore, the objective when generating such an ensemble is that it comprises a diverse set of accurate learners. The global best (gbest) PSO search mechanism [23] attempts to direct each population member towards the global optimal solution vector that has been found up to the current iteration, but also towards the personal best position (solution vector) that has been found by the corresponding population member thus far. In this article, it is shown that these two features of the (gbest) search and network training mechanism provide the desirable diverse ensemble of accurate learners. Diversity is preserved via the attraction of each population member towards its current personal best solution and improved prediction accuracy is achieved via its attraction towards the solution with the current minimum validation error. When the stopping criterion of the training process has been met, the current set of personal best solution vectors comprises the ensemble of RBF networks that is utilized to compute the network output.

The rest of the article is organized as follows: The proposed methodology for training, pruning, and ensembling of RBF networks is presented in Sect. 2. The results of its application to regression and time series prediction benchmark problems and comparisons with other SLFN learners are presented in Sect. 3. Conclusions are provided in Sect. 4.

II. TRAINING, PRUNING, AND ENSEMBLING OF RBF NETWORKS USING PSO AND THE ELM

2.1 ELM-based RBF network

An RBF network is an SLFN with a radial basis function assigned to each hidden node. Therefore, the function to be approximated is represented as an expansion in basis functions, which are modeled using kernel functions. Even though, there are no connection weights between input and hidden layer, the coordinates of the kernel centers need to be determined and, thus, are considered parameters of the network. In this work, the inverse multiquadric kernel is utilized in the following form,

$$\phi(\mathbf{x}) = (\|\mathbf{x} - \bar{\mathbf{x}}\|^2 + \sigma^2)^{-1/2} \qquad (1)$$

where $\bar{\mathbf{x}}$ is the kernel center coordinate vector, $\mathbf{x}$ is the input vector, and σ is the basis width, or smoothing parameter, which also needs to be determined for each kernel. The RBF network output is computed as the weighted average of the output of the hidden nodes, including the contribution of a bias node. Assuming a network with N hidden layer nodes and a single output node, the value of the approximated function at $\mathbf{x}$ is computed as follows,

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n \phi_n(\mathbf{x}) + w_0 \qquad (2)$$

where w_n is the weight of the n^{th} radial basis function in the corresponding hidden node and w_0 is the bias node weight. These $N + 1$ weights are obtained through a supervised learning approach, i.e., the network is trained by adjusting its parameters so that the overall output error is minimized when it is evaluated on a training dataset.

The training objective is typically formulated as a minimization of the sum-of-squares problem.

$$\min_{(\sigma_n, \bar{x}_n, w_n)_1^N} \sum_{p=1}^{P} \left(f(\mathbf{x}_p) - y(\mathbf{x}_p) \right)^2 \qquad (3)$$

where*P* is the number of instances in the training dataset. The optimization problem defined in eq. (3) is nonconvex with multiple local minima [25]. Gradient descent can be utilized to obtain a solution for the network weights, the kernel centers, and the basis widths [8]. Given the local-approximator nature of bounded radial basis functions, a clustering algorithm, e.g., *K*-means, can also be employed at the initial phase of the training process to determine the positions of the kernel centers [26]. The ELM algorithm adapted for RBF networks [10] provides a much faster approach: The kernel centers and basis widths are initialized with random values from within a specific range and the problem of determining the weights is then formulated as follows,

$$\sum_{n=1}^{N} w_n \phi_n(\mathbf{x}_p) + w_0 = y(\mathbf{x}_p), \quad p \in \{1, \ldots, P\} \tag{4}$$

This corresponds to a linear system of P equations, which can be written in a compact matrix form as follows.

$$\mathbf{Hw} = \mathbf{Y} \tag{5}$$

The training of the network can then be accomplished by finding a least-squares solution $\hat{\mathbf{w}}$ of eq. (5):$\min_{\mathbf{w}} \|\mathbf{Hw} - \mathbf{Y}\|$. In most practical applications, the number of hidden nodes is much smaller than the size of the training dataset. In this case, eq. (5) corresponds to an over determined system of equations and the unique smallest-norm least squares solution is as follows.

$$\hat{\mathbf{w}} = \mathbf{H}^{\dagger}\mathbf{Y} \tag{6}$$

where$\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse matrix [27]. This can be computed using a number of methods; in this work this is done using the singular value decomposition (SVD) approach. As is pointed out in [10, 28], in general, the smaller the network weights, the better the generalization performance; using the $\mathbf{H}^{\dagger}$ matrix, the smallest hidden-to-output layer weights are obtained.

2.2Particle swarm optimization

The utilization of the ELM for the training of SLFNs results in a significant reduction in the training time compared to gradient-based tuning algorithms. However, as is reported in [6], when the ELM is employed for the training of ANNs, the random selection of the values of the input weights tends to favor networks with a larger number of hidden nodes compared to gradient-based network tuning. In order to address this issue, an evolutionary algorithm can be utilized to evolve the network parameters, as is done in [6] where a differential evolution algorithm is combined with the ELM to train ANNs. In addition to a shorter training time, a more compact network architecture could also result in better generalization performance. These observations are expected to be applicable to other types of SLFNs like RBF networks. In this work, PSO is utilized to evolve both the position of each kernel and the corresponding basis width.

The*gbest* PSO model [29] uses a population of swarm particles (solution vectors) that search for the optimal solution simultaneously and in a cooperative manner. The position vector of eachparticle$\mathbf{x} \in \mathbb{R}^J$is updated at each iteration $t + 1$ using the following scheme for every $j \in \{1, \ldots, J\}$:

$$x_j(t+1) = x_j(t) + v_j(t+1) \tag{7}$$

$$v_j(t+1) = \chi(v_j(t) + \phi_1 . U_j(0,1).\left(\hat{y}_j(t) - x_j(t)\right) + \phi_2 . U_j(0,1).\left(y_j(t) - x_j(t)\right) \tag{8}$$

where$x_j(t)$,$v_j(t)$, $x_j(t+1)$, and $v_j(t+1)$ are the particle's j^{th} position coordinate and velocity over a single time increment at iteration t and t + 1, respectively. $\phi_1$andϕ_2 are coefficients that adjust the attraction of the particle towards the global best solution that has been found by the swarm thus far, $\hat{\mathbf{y}}(t)$, and towards the best solution that has been found by the particle up to iteration t, $\mathbf{y}(t)$, respectively.$U_j(0,1)$is a uniformly distributed random number in (0, 1) sampled anew for each *j* and particle.

In order to prevent the velocity of each particle from increasing uncontrollably when using eq. (7), various methods have been proposed over the years; here the concept of the constriction coefficient [30] is adopted. The constriction coefficient, χ, is computed using the following scheme as shown.

$$\chi = \frac{2k}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|} \tag{9}$$

where $\phi = \phi_1 + \phi_2, \phi > 4, k \in [0, 1]$.

In this work, *k* is set equal to one in order to promote a high degree of exploration of the search space, ϕ is set equal to 4.1, as is suggested in [31], and $\phi_1$is set equal toϕ_2. The condition ϕ> 4 is a necessary condition for the convergence of the particle's trajectory to a position inside the search space. This is proven in [30], where the equations of motion are modeled as a discrete-time dynamic system and a stability analysis is performed in order to derive conditions for its convergence to an equilibrium point. Using the *gbest* model, the particle attractor (equilibrium point) corresponds to a weighted average between its personal best and global best positions. In the current application, when the network training has been completed, it is anticipated that the set of personal best positions contains solution vectors close to the global best solution, depending on the size of the attraction basin, which are also distinct enough to satisfy the diversity requirement for the ensemble members.

The positions of the particles are initialized randomly within the range of each coordinate (input variable): $x_j \in \left[x_j^{(L)}, x_j^{(U)}\right], j \in \{1, \dots \dots J\}$. The velocities are initialized with zero values. During the iterative search process, when a particle moves to a position outside of the allowable range in coordinate *j*, its position coordinate *j* is set equal to the closest boundary value and the corresponding velocity component is set equal to zero. At the end of each iteration, the performance of the swarm particles is assessed by computing the root mean squared error (RMSE) on a validation set, which contains data that are not included in the training dataset. This is done in order to update, if applicable, the global best and personal best solution vectors. The RBF network parameters that are optimized are the kernel center coordinates and basis widths.

In this research, the PSO algorithm is modified as follows: The particle with the worst (highest) RMSE value at the end of each iteration is replaced by a mutated (perturbed) copy of the global best solution vector. The mutation is performed using the following scheme $j \in \{1, \dots . J\}$,

$$x_j = \begin{cases} \hat{y}_j + msf.\ \left(x_j^{(U)} - x_j^{(L)}\right),\ if\ U_j(0,1) < mrt \\ \hat{y}_j,\ \ otherwise \end{cases} \qquad (10)$$

where*msf* is the mutation scaling factor and *mrt* is the mutation rate. In this way, the optimizer is able to perform a local search in the vicinity of the global best solution found thus far through small perturbations of the corresponding solution vector. During the initialization of the PSO parameters' values for each swarm particle, the input layer of each corresponding RBF network, *i*, is pruned by randomly selecting the input variables that will be included in the network as shown below,

$$x_{ij} = \begin{cases} deactivated,\ if\ U_{ij}(0,1) < prr \\ activated\ \ \ ,\ otherwise \end{cases} \qquad (11)$$

where*prr* is the pruning rate and $j \in \{1, \dots . , J\},\ i \in \{1, \dots \dots , I\}$.

The main reason for pruning the input layer is to remove variables that do not contribute towards a better understanding of the underlying process that produced the dataset and, thus, their inclusion does not cause a substantial increase in the accuracy of the approximation/prediction model. In the proposed approach, the importance of the input variables is not estimated explicitly; the determination of the optimal input layer architecture is done gradually through the aforementioned particle replacement operation as, at each iteration, the network with the worst performance is discarded and replaced by a network with the optimal input layer architecture that has been found thus far.

2.3 Implementation of the proposed algorithm for training and ensembling of RBF networks

The PSO algorithm described in the previous section is utilized for the training of the ELM-based RBF networks. The training of the ELM-based RBF networks is stopped if either the global optimal solution has not changed after I_{ch} iterations or the algorithm has reached the maximum allowable number of iterations I_{max}. Two distinct sets of data points are used during the training process; the first corresponds to the training data set, which is used to compute the network weights via Singular Value Decomposition (SVD). The particles (solution vectors) are then evaluated on a validation set in order to find global and personal best positions. In this way, the risk of overtraining the network is reduced. The global best position corresponds to the network with the smallest prediction error on the validation set. The prediction error is quantified by computing the root mean squared error (RMSE). The training and validation data sets, both input and output values, are normalized in the range [-1.0, 1.0].The ensembling process commences immediately after the training of the RBF networks has been finalized. The output of the ensemble is obtained by averaging the output of its members, i.e., the personal best solutions of the swarm particles. Prior to the evaluation of the generalization performance of the ensemble on a testing dataset, the existence of outliers among the ensemble members is investigated by applying Chauvenet's criterion [32]. This criterion specifies that all points that fall within a band around the mean that corresponds to a probability of $[1 - 1/(2E)]$should be retained.

E is the original size of the ensemble and, thus, is equal to the swarm population size. The criterion is applied only once for each point of the testing dataset. Using Gaussian probabilities, the ratio of maximum acceptable deviation to sample standard deviation is computed and utilized for the detection of outliers [33]. The algorithm has been developed in FORTRAN 95. The training and testing processes of the PSO-trained ELM-based RBF network ensemble are outlined in Fig. 1.

```
Specify the RBF network architecture
Initialize the swarm population particles (each particle corresponds to an RBF network)
iter = 0
do
   Compute the connection weights of each RBF network using the training data set and SVD
   Evaluate each RBF network on the validation data set by computing the RMSE
   Find the global best RBF network up to the currect iteration
   Update, if applicable, the personal best position of each particle
   Move each particle to a new position inside the search space using the gbest PSO algorithm
   iter = iter + 1
until stopping criterion is satisfied
Form RBF network ensemble by combining the personal best positions of the swarm particles
Apply Chauvenet's criterion while computing the ensemble prediction on the testing data set
```

Figure 1. Pseudo-code of the PSO-trained ELM-based RBFnetwork ensemble.

2.4 Experimental investigation

The generalization performance of the RBF networks trained using the proposed methodology is investigated and the results are presented in this section. In all the experiments, the swarm population size *I* is set equal to 20 and I_{ch} and I_{max} are set equal to 8 and 50, respectively. In the first part of this investigation, the number of hidden nodes is set equal to 10 in order to observe the algorithmic effectiveness and efficiency using a small-sized network. The coordinates of the kernel centers are allowed to vary within the range [-1.0, 1.0], while the basis width values within the range [1.0, 60.0]. The mutation parameters, *msf* and *mrt*, are set equal to 0.2 and 0.5, respectively, and the pruning rate *prr* is set equal to 0.2. The training and validation datasets are normalized in the range [-1.0, 1.0].

Ten widely-utilized benchmark problems are considered: Eight regression and two time series prediction problems. The datasets of the majority of these problems have been obtained from the UCI machine learning repository [34]. The problem features and additional references are provided in Table 1.

Table 1. Features of regression and time series prediction benchmark problems.

ID	Problem description	Number of data points	Number of inputs	Input types
BNK	Bank queues simulation	8192	8	integer, real
FF	Forest fires [36]	517	4	real
BH	Housing values in Boston	506	13	categorical, integer, real
CCS	Concrete compressive strength [37]	1030	8	real
SRV	Servomechanism	167	4	categorical, integer
CS	Concrete slump test [38]	103	7	real
CH	Computer hardware performance	209	7	integer
WBP	Breast Cancer Wisconsin (Prognostic)	198	32	real
BJ	Box-Jenkins time series [39]	290	10	real
MG	Mackey-Glass time series [40]	4898	11	real

The dataset of each problem is first randomized and then split into three groups: 40% of the data are used for training, 10% for validation, and 50% for testing. Fifty independent runs are performed for each problem. The RMSE and the mean absolute error (MAE) of the predictions on the testing set are computed using the network output, after it has been transformed back to its original scale, and recorded for the ensemble and for the RBF network with the lowest RMSE value on the validation set. The same 10 problems are used in all phases of this investigation. The computational cost of obtaining the ensemble predictions is negligible compared to the corresponding cost of the training process; on average, the time used to compute the ensemble predictions is equal to 0.7% of the time required for the training process on a machine with 16 GB of RAM and a quad-core 2.80 GHz processor running on a 64-bit Linux operating system.

III. EFFECTIVENESS OF THE PROPOSED ENSEMBLINGMETHODOLOGY

In the first part, the effectiveness of the proposed methodology is tested, and in particular the utilization of the mutation operator combined with the pruning of the input layer. The RMSE and the mean absolute error (MAE) of the predictions are computed on the testing dataset using the network output, after the latter has been transformed back to its original scale, and recorded for the ensemble (ENS) and for the global best RBF network (GB), i.e., the network with the lowest RMSE value on the validation set at the end of each run. The corresponding versions without mutation and pruning are denoted by ENS_NMP and GB_NMP, respectively. The lower the RMSE and the MAE values, the better the algorithmic performance.The results are shown in Table 2.

In all cases, a single hidden layer with 10 nodes is utilized and the maximum number of training iterations per run is set equal to 1000. A pairwise comparison between ENS and ENS_NMP to determine the statistical significance of the results is also performed using the two-tailed *p*-values, which have been computed using the *t*-test for unequal variances. In the problems where an algorithm has statistically better performance than the other at the 0.05 significance level, the mean value of its RMSE is highlighted in bold font.

The results reported in Table 2demonstrate the effectiveness of mutation and input-layer pruning on the algorithmic performance: ENS outperforms ENS_NMP in all 10 problems and in both metrics; the difference in the mean values is statistically significant at the 0.05 level in seven problems using either metric. Furthermore, the generalization performance of the ensemble (ENS) is clearly better than the performance of the global best network (GB) in both metrics when mutation and input-layer pruning are incorporated into the algorithm; the same conclusion cannot be drawn from a generalization performance comparison between GB_NMP and ENS_NMP, which further corroborates the claim that mutation and pruning enhance the PSO-training and ensembling effectiveness.

Table 2. RMSE and MAE results for ENS, GB, ENS_NMP and GB_NMP.

ID	RMSE & MAE	ENS	GB	ENS_NMP	GB_NMP
BNK	RMSE	$\mathbf{7.209}\cdot10^{-2}$	$7.254\cdot10^{-2}$	$8.721\cdot10^{-2}$	$8.725\cdot10^{-2}$
	MAE	$\mathbf{5.457}\cdot10^{-2}$	$5.497\cdot10^{-2}$	$6.797\cdot10^{-2}$	$6.793\cdot10^{-2}$
FF	RMSE	1.338	1.340	13.341	1.340
	MAE	**1.067**	1.069	1.096	1.092
BH	RMSE	**4.436**	4.601	4.986	4.893
	MAE	**3.411**	3.545	3.897	3.820
CCS	RMSE	$\mathbf{1.281}\cdot10^{1}$	$1.368\cdot10^{1}$	$1.531\cdot10^{1}$	$1.536\cdot10^{1}$
	MAE	$\mathbf{1.006}\cdot10^{1}$	$1.055\cdot10^{1}$	$1.258\cdot10^{1}$	$1.247\cdot10^{1}$
SRV	RMSE	$\mathbf{9.675}\cdot10^{-1}$	1.038	$9.973\cdot10^{-1}$	1.002
	MAE	$\mathbf{5.442}\cdot10^{-1}$	$5.941\cdot10^{-1}$	$6.200\cdot10^{-1}$	$6.156\cdot10^{-1}$
CS	RMSE	8.133	9.139	8.295	9.764
	MAE	6.612	7.253	6.728	7.781
CH	RMSE	$1.265\cdot10^{1}$	$1.474\cdot10^{1}$	$1.311\cdot10^{1}$	$1.617\cdot10^{1}$
	MAE	5.581	6.338	5.801	6.967
WBP	RMSE	$\mathbf{3.546}\cdot10^{1}$	$3.832\cdot10^{1}$	$4.160\cdot10^{1}$	$4.237\cdot10^{1}$
	MAE	$\mathbf{2.953}\cdot10^{1}$	$3.198\cdot10^{1}$	$3.497\cdot10^{1}$	$3.546\cdot10^{1}$
BJ	RMSE	$\mathbf{4.324}\cdot10^{-1}$	$4.437\cdot10^{-1}$	$4.414\cdot10^{-1}$	$4.559\cdot10^{-1}$
	MAE	$3.095\cdot10^{-1}$	$3.171\cdot10^{-1}$	$3.128\cdot10^{-1}$	$3.225\cdot10^{-1}$
MG	RMSE	$\mathbf{1.187}\cdot10^{-2}$	$1.276\cdot10^{-2}$	$2.165\cdot10^{-2}$	$2.144\cdot10^{-2}$
	MAE	$\mathbf{1.027}\cdot10^{-2}$	$1.034\cdot10^{-2}$	$1.783\cdot10^{-2}$	$1.759\cdot10^{-2}$

The performance of the ensemble (ENS) and of the global best (GB) of the PSO-ELM-trained RBF networks is compared with the performance of two other SLFN learners: An artificial neural network (ANN) that uses the back propagation algorithm for training and an RBF network that uses *K*-means clustering (RBF_K) to obtain the kernel parameters and linear regression to compute the net-work weights. Both algorithms are available in the open source data mining software WEKA [42]. The ANN uses a momentum term with value set equal to 0.2 and a learning rate with value set equal to 0.3. Both SLFN learners use a single hidden layer with 10 nodes; the number of training iterations is set equal to 1000.

Table 3. Mean and standard deviation values of RMSE for RBF_K, ANN, GB and ENS.

ID	Mean & Deviation	RBF_K	ANN	GB	ENS
BNK	Mean	$1.420 \cdot 10^{-1}$*	$8.130 \cdot 10^{-2}$*	$7.254 \cdot 10^{-2}$	**7.209**$\cdot 10^{-2}$
	Deviation	$1.484 \cdot 10^{-4}$	$1.276 \cdot 10^{-2}$	$2.640 \cdot 10^{-4}$	$3.545 \cdot 10^{-4}$
FF	Mean	1.359*	1.461*	1.340	1.338
	Deviation	$1.391 \cdot 10^{-2}$	$1.326 \cdot 10^{-1}$	$2.500 \cdot 10^{-2}$	$2.562 \cdot 10^{-2}$
BH	Mean	7.505*	5.455*	4.601	**4.436**
	Deviation	$2.919 \cdot 10^{-1}$	1.089	$4.694 \cdot 10^{-1}$	$2.954 \cdot 10^{-1}$
CCS	Mean	$1.844 \cdot 10^{1}$*	$1.727 \cdot 10^{1}$*	$1.368 \cdot 10^{1}$	**1.281**$\cdot 10^{1}$
	Deviation	1.870	3.071	2.052	1.969
SRV	Mean	1.524*	1.014	1.038	**9.675**$\cdot 10^{-1}$
	Deviation	$7.097 \cdot 10^{-2}$	$2.278 \cdot 10^{-1}$	$6.677 \cdot 10^{-2}$	$4.355 \cdot 10^{-2}$
CS	Mean	$1.169 \cdot 10^{1}$*	8.715*	9.139	**8.133**
	Deviation	9.547	1.381	1.085	$9.001 \cdot 10^{-1}$
CH	Mean	$1.261 \cdot 10^{2}$*	$1.380 \cdot 10^{1}$	$1.474 \cdot 10^{1}$	**1.265**$\cdot 10^{1}$
	Deviation	$3.996 \cdot 10^{1}$	$1.271 \cdot 10^{1}$	6.197	3.342
WBP	Mean	$3.835 \cdot 10^{1}$*	$4.172 \cdot 10^{1}$*	$3.832 \cdot 10^{1}$	**3.546**$\cdot 10^{1}$
	Deviation	1.222	9.467	2.114	$5.716 \cdot 10^{-1}$
BJ	Mean	1.364*	$5.893 \cdot 10^{-1}$*	$4.437 \cdot 10^{-1}$	**4.324**$\cdot 10^{-1}$
	Deviation	$1.102 \cdot 10^{-1}$	$1.913 \cdot 10^{-1}$	$2.878 \cdot 10^{-2}$	$2.236 \cdot 10^{-2}$
MG	Mean	$8.180 \cdot 10^{-2}$*	$1.068 \cdot 10^{-2}$	$1.276 \cdot 10^{-2}$	$1.187 \cdot 10^{-2}$
	Deviation	$5.506 \cdot 10^{-3}$	$3.257 \cdot 10^{-3}$	$3.624 \cdot 10^{-3}$	$2.809 \cdot 10^{-3}$

The computed mean (Mean) and standard deviation (Deviation) values of RMSE are listed in Table 3. Pairwise comparisons between ENS, RBF_K, and ANN are performed in order to determine the statistical significance of the results. If the performance of ENS in a problem is statistically better than the performance of another algorithm, then there is an asterisk (*) next to the other algorithm's corresponding mean RMSE value. If the difference in performance between ENS and GB is statistically significant at the 0.05 level, the mean value of the more accurate algorithm is highlighted in bold font.

The RMSE results displayed in Table 3 reveal that the PSO-ELM-trained RBF network ensemble has better generalization performance than the RBF_K learner in ten problems, a result that is statistically significant in all ten problems, and in nine problems compared to the ANN, a result that is statistically significant in seven problems. Furthermore, the variance in the ENS results is very small compared to the other two SLFN learners. In none among the ten problems the performance of either ANN or RBF_K is statistically better than the performance of ENS. A comparison between the results of GB and ENS shows that the latter performs better in all ten problems, a result that is statistically significant in eight problems. Overall, these results demonstrate that the PSO-trained ELM-based RBF network ensembling methodology has very good generalization performance even when applied to a small-sized network. The proposed PSO-ELM-based training methodology without the ensembling is also successful as GB has a lower mean RMSE value than the RBF_K and the ANN in ten and six problems, respectively.

3.1 RBF Networks with Optimal Number of Hidden Layer Nodes

In the final part, the number of hidden layer nodes is varied in an attempt to optimize the network size. Starting with two hidden nodes, the number is increased manually in steps of one node to a maximum number of twenty nodes. The network size of the ensemble (ENS_OPT) that produces the lowest mean RMSE value in each problem is (following the sequence used in Table 1): {20, 11, 5, 12, 20, 12, 20, 18, 20, 20}. The corresponding mean RMSE values are shown in Table 4.

The results obtained using the IB5 *k*-nearest neighbor algorithm [43], a Gaussian process (GP) learner, and M5P [44], a tree-based method with pruning, are also listed in Table 4. GP uses the Gaussian kernel function with a basis width that is varied manually from within the following set of discrete values: {0.25, 0.5, 1.0, 1.5, 2.0, 3.0, 5.0, 10.0}. The results that correspond to the basis width value that produces the lowest mean RMSE in each problem are shown in Table 4. The corresponding basis width values are: {1.0, 1.0, 1.5, 1.0, 0.5, 5.0, 1.5, 3.0, 2.0, 10.0}. The data mining software WEKA is utilized to generate the results for IB5, GP, and M5P. The lowest mean RMSE and MAE values in each problem are highlighted in bold font.

The generalization performance of the proposed methodology is significantly improved by using an optimal-sized hidden layer as is observed through a comparison between the results of ENS listed in Tables 2 and the results of ENS_OPT listed in Table 4 (next page). A comparison between the results of the GP and the IB5 learners and the results of ENS_OPT reveals that the latter outperforms both learners in all ten problems using either metric. It also outperforms M5P in nine problems using the RMSE metric and in eight problems using the MAE metric.

Table 4. RMSE and MAE results for GP, IB5, M5P and ENS_OPT.

ID	RMSE & MAE	GP	IB5	M5P	ENS_OPT
BNK	RMSE	$7.251 \cdot 10^{-2}$	$1.155 \cdot 10^{-1}$	$7.090 \cdot 10^{-2}$	$\mathbf{7.085} \cdot 10^{-2}$
	MAE	$5.491 \cdot 10^{-2}$	$8.940 \cdot 10^{-2}$	$\mathbf{5.322} \cdot 10^{-2}$	$5.331 \cdot 10^{-2}$
FF	RMSE	1.345	1.443	1.352	**1.336**
	MAE	1.060	1.095	1.124	**1.058**
BH	RMSE	4.715	6.615	**3.776**	4.139
	MAE	3.180	4.602	**2.797**	3.152
CCS	RMSE	$1.317 \cdot 10^{1}$	$1.743 \cdot 10^{1}$	$1.308 \cdot 10^{1}$	$\mathbf{1.264} \cdot 10^{1}$
	MAE	$1.094 \cdot 10^{1}$	$1.410 \cdot 10^{1}$	$1.036 \cdot 10^{1}$	**9.972**
SRV	RMSE	1.061	1.098	$9.261 \cdot 10^{-1}$	$\mathbf{9.030} \cdot 10^{-1}$
	MAE	$5.514 \cdot 10^{-1}$	$5.644 \cdot 10^{-1}$	$4.611 \cdot 10^{-1}$	$\mathbf{4.350} \cdot 10^{-1}$
CS	RMSE	8.218	8.519	8.194	**8.073**
	MAE	6.401	6.625	6.409	**6.395**
CH	RMSE	$4.328 \cdot 10^{1}$	$5.904 \cdot 10^{1}$	$3.189 \cdot 10^{1}$	**9.121**
	MAE	$1.678 \cdot 10^{1}$	$1.953 \cdot 10^{1}$	$1.476 \cdot 10^{1}$	**4.385**
WBP	RMSE	$3.664 \cdot 10^{1}$	$3.965 \cdot 10^{1}$	$3.549 \cdot 10^{1}$	$\mathbf{3.404} \cdot 10^{1}$
	MAE	$3.116 \cdot 10^{1}$	$3.291 \cdot 10^{1}$	$2.877 \cdot 10^{1}$	$\mathbf{2.770} \cdot 10^{1}$
BJ	RMSE	1.003	1.074	$4.521 \cdot 10^{-1}$	$\mathbf{3.915} \cdot 10^{-1}$
	MAE	$7.001 \cdot 10^{-1}$	$7.899 \cdot 10^{-1}$	$3.213 \cdot 10^{-1}$	$\mathbf{2.867} \cdot 10^{-1}$
MG	RMSE	$1.540 \cdot 10^{-2}$	$1.071 \cdot 10^{-2}$	$3.612 \cdot 10^{-2}$	$\mathbf{7.238} \cdot 10^{-3}$
	MAE	$1.180 \cdot 10^{-2}$	$8.700 \cdot 10^{-3}$	$2.860 \cdot 10^{-2}$	$\mathbf{5.758} \cdot 10^{-3}$

IV. CONCLUSIONS AND FUTURE RESEARCH

The development of a new methodology that combines PSO and ELM to train and generate ensembles of RBF networks is described in this article. PSO, supplemented with the proposed mutation operator and pruning of the input layer, is utilized in or-der to optimize the kernel parameters; this results in RBF networks with a compact architecture and very good generalization performance. Combining the networks that correspond to the personal best positions of the swarm particles to form an ensemble increases the robustness of the algorithm and further enhances its generalization performance. Optimizing the size of the hidden layer results in further improvement in the ensemble's generalization performance. These conclusions are drawn from comparisons between the ensemble's performance and the performance of other SLFNs on eight regression and two time series prediction benchmark problems.

The optimization of the RBF network's architecture without a substantial increase in the required training time is currently being investigated. Furthermore, the development of a PSO model tailored for ensembling purposes, e.g., having more control over the particles' trajectories, is also being considered.

REFERENCES

[1] G.B. Huang, Q.Y. Zhu, C.K. Siew. Extreme learning machine: theory and applications. Neurocomputing, 70(1-3):489-501, 2006.
[2] Y. Wang, F. Cao, Y. Yuan. A study on effectiveness of extreme learning machine. Neurocomputing, 74(16):2483-2490, 2011.
[3] M. van Heeswijk, Y. Miche, E. Oja, A. Lendasse. GPU-accelerated and parallelized ELM ensembles for large-scale regression. Neurocomputing, 74(16):2430-2437, 2011.
[4] G.B. Huang, L. Chen. Convex incremental extreme learning machine. Neurocomputing, 70(16-18):3056-3062, 2007.
[5] G. Feng, G.B. Huang,Q. Lin and R. Gay. Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Transactions Neural Networks, 20(8), pp. 1352-1357, 2009.
[6] Q.Y. Zhu, A.K. Qin, P.N. Suganthanand G.B. Huang. Evolutionary extreme learning machine, Pattern Recognition, 38(10), pp. 1759-1763, 2005.
[7] D.S. Broomheadand D. Lowe.Multivariate functional interpolation and adaptive networks, Complex Systems, 2, pp. 321-355, 1988.
[8] T. Poggio and F. Girosi. Networks for approximation and learning, Proc. IEEE, 78(9), pp. 1481-1497, 1990.
[9] C. Harphamand C.W. Dawson. The effect of different basis functions on a radial basis function network for time series prediction: A comparative study, Neurocomputing, 69(16-18), pp. 2161-2170, 2006.
[10] G.B. Huangand C.K. Siew. Extreme learning machine with randomly assigned RBF kernels, International Journal of Information Technology, 11(1), pp. 16-24, 2005.
[11] R. Mendes, P. Cortez, M. Rochaand J. Neves.Particle swarms for feedforward neural network training, In: Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, Hawaii, pp. 1895-1899, 2002.
[12] F. Han, H.F. Yao and Q.H. Ling. An improved evolutionary extreme learning machine based on particle swarm optimization, Neurocomputing, 116, pp. 87-93, 2013.
[13] M.W. Mak and K.W. Cho. Genetic evolution of radial basis function centers for pattern classification. In: Proceedings of the 1998 IEEE International Joint Conference on Neural Networks, Anchorage, Alaska, pp. 669-673, 1998.
[14] H.M. Feng.Self-generation RBFNs using evolutional PSO learning, Neurocomputing, 70(1-3), pp. 241-251, 2006.
[15] J. Gonzalez, I. Rojas, J. Ortega, H. Pomares, F.J. Fernandez and A.F. Diaz. Multi-objective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation, IEEE Transactions Neural Networks, 14(6), pp. 1478-1495, 2003.

[16] M. Rocha, P. Cortez and J. Neves. Evolution of neural networks for classification and regression, Neurocomputing, 70(16-18), pp. 2809-2816, 2007.
[17] H. Du and N. Zhang. Time series prediction using evolving radial basis function networks with new encoding scheme, Neurocomputing, 71(7-9), pp. 1388-1400, 2008.
[18] I. Kokshenevand P.A. Braga.A multi-objective approach to RBF network learning, Neurocomputing, 71(7-9), pp. 1203-1209, 2008.
[19] N. Kondo, T. Hatanaka and K. Uosaki. Pattern classification by evolutionary RBF networks ensemble based on multi-objective optimization, In: Proceedings of the 2006 International Joint Conference on Neural Networks, Vancouver, British Columbia, pp. 2919-2925, 2006.
[20] Y. Jin, T. Okabeand B. Sendhoff. Neural network regularization and ensembling using multi-objective evolutionary algorithms, In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, Portland, Oregon, pp. 1-8, 2006.
[21] A. Chandra and X. Yao. Evolving hybrid ensembles of learning machines for better generalization, Neurocomputing, 69(7-9), pp. 686-700, 2006.
[22] A. Kroghand J. Vedelsby.Neural network ensembles, cross validation, and active learning, Advances in Neural Information Processing Systems, 7, pp. 231-238, 1995.
[23] R.C. Eberhart, P.K. Simpson and R.W. Dobbins. Computational Intelligence PC Tools. Boston: Academic Press, 1995.
[24] R.L. Hardy. Multiquadric equation of topography and other irregular surfaces, Journal of Geophysical Research, 76(8), pp. 1905-1915, 1971.
[25] T. Hastie, R. Tibshirani and J. Friedman. The elements of statistical learning. New York: Springer Science plus Business Media, 2nd edition, pp. 212-214, 2009.
[26] J.E. Moodyand C. Darken. Learning with localized receptive fields, In: Touretzky D, Hinton G, Sejnowski T (Eds) Proceedings of the Connectionist Models Summer School, San Mateo, California, Morgan Kaufmann, pp. 133-143, 1989.
[27] C.R. Rao and S.K. Mitra. Generalized inverse of matrices and its applications. New York: Wiley publications, 1971.
[28] P.L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network, IEEE Transactions on Information Theory, 44(2), pp. 522-536, 1998.
[29] J. Kennedy and R.C. Eberhart. Particle swarm optimization, In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942-1948, 1995.
[30] M. Clerc and J. Kennedy, J. The particle swarm explosion, stability, and convergence in a multi-dimensional complex space, IEEE Transactions on Evolutionary Computing, 6(1), pp. 58-73, 2002.
[31] R. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization, In: Proceedings of the 2000 IEEE Congress on Evolutionary Computation, San Diego, California, pp. 84-88, 2000.
[32] W. Chauvenet. A manual of spherical and practical astronomy. Philadelphia: Lippincott, Vol. 2, 1st edition, 1863.
[33] H.W. Coleman and W.G. Steele. Experimentation and uncertainty analysis for engineers. New York: Wiley publications, 2nd edition, 2002.
[34] K. Bache and M. Lichman. UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, http://archive.ics.uci.edu/ml, 2014.
[35] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physiochemical properties, Decision Support Systems, 47(4), pp. 547-553, 2009.
[36] P. Cortez and A. Morais. Data mining approach to predict forest fires using meteorological data, In: Neves J, Santos MF, Machado J (Eds) New trends in artificial intelligence, Proceedings of the 13th EPIA, Portuguese Conference on Artificial Intelligence, Guimaraes, Portugal, pp. 512-523, 2007.
[37] I.C. Yeh. Modeling of strength of high performance concrete using artificial neural networks, Cement Concrete Research, 28(12), pp. 1797-1808, 1998.
[38] I.C. Yeh, I.C. Modeling slump flow of concrete using second-order regressions and artificial neural networks, Cement Concrete Composites, 29(6), pp. 474-480, 2007.
[39] G.E.P. Box and G.M. Jenkins. Time series analysis: forecasting and control. San Francisco: Holden-Day, Revised edition, 1976.
[40] M.C. Mackey and L. Glass. Oscillation and chaos in physiological control systems, Science, 197(4300), pp. 287-289, 1977.
[41] B.L. Welch. The generalization of 'Student's' problem when several different population variances are involved, Biometrika, 34(1-2), pp. 28-35, 1947.
[42] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemannand I.H. Witten. The WEKA data mining software: an update. ACM SIGKDD Exploration Newsletter, 11(1), pp. 10-18, 2009.
[43] D. Aha and D. Kibler. Instance-based learning algorithms, Machine Learning, 6, pp. 37-66, 1991.
[44] R.J. Quinlan. Learning with Continuous Classes, In: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, Singapore, pp. 343-348, 1992.