International Journal of Uncertainty,
Fuzziness and Knowledge-Based Systems
Vol. 21, Suppl. 2 (December 2013) 143–154
© World Scientific Publishing Company
DOI: 10.1142/S0218488513400229

World Scientific
www.worldscientific.com

EVOLVING EXTREME LEARNING MACHINE PARADIGM WITH ADAPTIVE OPERATOR SELECTION AND PARAMETER CONTROL

KE LI[*], RAN WANG[†], SAM KWONG[‡] and JINGJING CAO[§]

Department of Computer Science, City University of Hong Kong,
83 Tat Chee Avenue, Kowloon, Hong Kong
[*] *keli.genius@gmail.com*
[†] *ranwang3-c@my.cityu.edu.hk*
[‡] *cssamk@cityu.edu.hk*
[§] *bettymore0501@gmail.com*

Received 13 September 2012
Revised 1 September 2013

Extreme Learning Machine (ELM) is an emergent technique for training *Single-hidden Layer Feedforward Networks* (SLFNs). It attracts significant interest during the recent years, but the randomly assigned network parameters might cause high learning risks. This fact motivates our idea in this paper to propose an evolving ELM paradigm for classification problems. In this paradigm, a *Differential Evolution* (DE) variant, which can online select the appropriate operator for offspring generation and adaptively adjust the corresponding control parameters, is proposed for optimizing the network. In addition, a 5-fold cross validation is adopted in the fitness assignment procedure, for improving the generalization capability. Empirical studies on several real-world classification data sets have demonstrated that the evolving ELM paradigm can generally outperform the original ELM as well as several recent classification algorithms.

Keywords: Extreme learning machine; adaptive operator selection; parameter control; differential evolution.

1. Introduction

Extreme Learning Machine (ELM)[6] is an emergent technique for training *Single-hidden Layer Feed-forward Neural networks* (SLFNs). Instead of adjusting the network parameters iteratively, in ELM, the input weights and hidden biases are chosen randomly while the output weights are calculated analytically by using *Moore-Penrose* (MP) generalized inverse. ELM has an error bound similar to RBFNN[2] which is experimentally proved to have better accuracy than the rule-based methodologies.[24] The architecture selection of networks in ELM has been discussed in Ref. 23. An ensemble strategy for ELM and an upper integral network classifier system have been proposed in Ref. 26, respectively. Moreover, in the past few years, much effort has been devoted to applying ELM for various real-world problems, such as image classification[7] and visual tracking.[19] A recent comprehensive survey on ELM can be found in Ref. 4. Compared with gradient based methods, ELM not only have faster training speed and better generalization performance, but also avoids the problem of local optima. However, it is argued that the randomly assigned input parameters may affect the final performance a lot and the selection of appropriate parameters can reduce the learning risk of ELM.[18]

In artificial intelligence, *Evolutionary Algorithm* (EA) is a subset of evolutionary computation, a generic population-based metaheuristic optimization algorithm. It is relatively easy to implement and can be used for problems even without good mathematical properties. In order to improve the robustness of the original ELM and reduce its learning risk, this paper presents an evolving ELM paradigm, which aims at finding a near optimal set of network parameters globally within a fixed network architecture. To be specific, it hybridizes a *Differential Evolution* (DE) variant,[8] with ELM to optimize the intrinsic network parameters. The DE variant under use is with adaptive operator selection and parameter control. In particular, four DE mutation operators with diverse search characteristics are combined to form an operator pool, and the proposed DE algorithm can online select the appropriate operator for the current offspring generation. Meanwhile, the hyper-parameters related to those operators are tuned adaptively according to the current fitness landscapes. Moreover, the generalization performance of the classifier and the norm of the output weights are combined to form a fitness function, which is used to evaluate the quality of an individual.

The remainder of this paper is organized as follows: Section 2 presents the background of this work. Section 3 gives the technical details of our proposed evolving ELM paradigm. Section 4 presents and analyzes the empirical results. Finally, Sec. 5 concludes this paper and highlights some possible future directions.

2. Background and Related Works

In this section, some background of ELM is introduced at first, and several related works on hybridizing EAs with ELM for training NNs are then reviewed.

2.1. *Extreme learning machine*

ELM theories, proposed by Huang *et al.*,[5] claim that the input weights and hidden biases of SLFNs can be randomly assigned if the activation function is infinitely differentiable and the output weights can be analytically determined by solving a linear system using the least square method, without any learning iteration.

Suppose that we have N arbitrary distinct samples (x_i, y_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ and $y_i = [y_{i1}, y_{i2}, \ldots, y_{im}]^T \in R^m$. Let H be the number of hidden neurons, $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ be the input weight vector connecting the ith hidden neuron and the input neurons, b be the $H \times 1$ bias values for each hidden neuron and β be the $H \times m$ output weight matrix, where $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$

is the weight vector connecting the ith hidden neuron and the output neurons. In general, the ELM network structure can be mathematically modeled as:

$$y_j = \sum_{i=1}^{H} \beta_i g(w_i \cdot x_j + b_i) \tag{1}$$

where $j \in \{1, 2, \ldots, N\}$, $w_i \cdot x_j$ denotes the inner product of w_i and x_j, and $g(x)$ is the activation function. Without loss of generality, the sigmoid additive activation function is chosen as $g(x)$ here, which is formulated as:

$$g(x) = \frac{1}{1 + \exp[-(w \cdot x + b)]}\,. \tag{2}$$

In ELM, the input weights and hidden biases are randomly chosen, while the output weights are calculated analytically as:

$$\beta = Y_H^{\dagger} Y \tag{3}$$

where $Y_H^{\dagger}$ is the Moore-Penrose (MP) pseudo-inverse of the hidden layer output matrix Y_H, and $Y = [y_1, y_2, \ldots, y_N]^T$.

2.2. *Methods of tuning ELM parameters*

As discussed in Sec. 1, ELM can achieve a good generalization performance with an extremely fast learning speed, comparing to the traditional gradient based learning algorithms. However, the network parameters, i.e. input weights and hidden biases, can largely influence the performance of ELM. This can be explained by the fact that ELM tries to find the global optimum of a localized linear system while this might not be the global optimum in the problem space. There are several works in this literature to find the near optimal network parameters of ELM. Liu *et al.*[12] proposed an ensemble based ELM (EN-ELM), in which ensemble learning and cross validation are embedded into the training phase to minimize the jeopardy caused by the overfitting problem and thus enhance the predictive stability. Miche *et al.*[13] proposed an optimally pruned ELM (OP-ELM), which improves the learning ability by pruning the useless neurons. Cao *et al.*[1] presented a voting based ELM (V-ELM) by employing the majority voting scheme. Zhu *et al.*[28] presented an evolutionary ELM (E-ELM) in which the basic DE algorithm is employed to evolve the network parameters. Based on the same framework of E-ELM, Silva *et al.*[20] and Xu *et al.*[25] proposed other two evolutionary ELMs. The only differences of the latter two are their evolutionary parts, which replace the original DE algorithm with *Group Search Optimization* (GSO)[3] and PSO, respectively. In Ref. 18, Saraswathi *et al.* proposed a hybrid EA with ELM, called ICGA-PSO-ELM, for gene selection and cancer classification.

3. Evolving ELM Learning Paradigm

In this section, the proposed DE algorithm and the system architecture of the evolving ELM paradigm are described in detail.

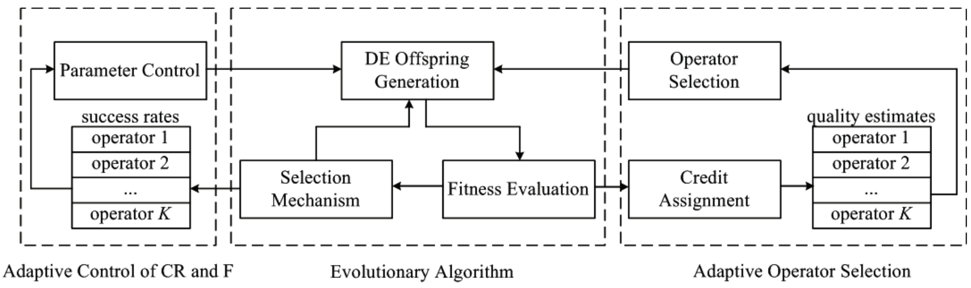

Fig. 1. The framework of the proposed DE algorithm with adaptive operator selection and parameter control.

3.1. *DE with adaptive operator selection and parameter control*

The general framework of the proposed DE algorithm with adaptive operator selection and parameter control (denoted as AdapDE) is given in Fig. 1. The right hand side is the adaptive operator selection module, which can select operators online based on their previous performances; while the parameter control module, which adjusts the parameters adaptively, is laid on the left hand side. The middle part is the main search engine, DE algorithm. Due to the page limitation, the pseudo-code of AdapDE is presented in supplementary file of this paper.[a]

In this paper, each solution in the evolutionary population is encoded with input weights and hidden biases as:

$$C = [w_{11}, \dots, w_{1H}, w_{22}, \dots, w_{2H}, \dots, w_{n1}, \dots, w_{nH}, b_1, \dots, b_H]\,. \tag{4}$$

Each input weight $w_{i,j}$ is initialized within the range of $[-1, 1]$ while the hidden bias b_i is initialized within the range of $[0, 1]$. The solution quality is evaluated as the following aggregation term:

$$\text{fitness} = E_{\text{accuracy}} - \alpha_w \times \|\beta\| \tag{5}$$

where E_{accuracy} is the K-fold cross validation accuracy, $\|\beta\|$ is the average norm of the output weights and α_w is a user defined parameter to adjust the penalty term to the cross validation accuracy. From our preliminary experiments, we set $\alpha_w = 0.01$ without loss of generality. It is obvious that the larger the *fitness* is, the better the individual would be.

3.2. *Adaptive operator selection*

Adaptive Operator Selection (AOS) paradigm aims at autonomously controlling which operator should be applied at each time point of the search, when solving the problem, based on their recent performances. In this paper, the proposed AOS paradigm consists of two components: the credit assignment module defines

[a]The supplementary document can be downloaded from `http://www.cs.cityu.edu.hk/~51888309`.

how each operator should be rewarded based on the impacts of its recent applications; and the operator selection mechanism decides which of the available operators should be applied next, according to their empirical quality estimates, which are built and constantly updated by the corresponding rewards.

3.2.1. *Credit assignment module*

In order to assign the credit value for an applied operator, in this work, we adopt the impact assessment scheme used in Ref. 15 as follows:

$$\xi = \frac{cf_i}{\varphi} \times |pf_i - cf_i| \tag{6}$$

where $i \in \{1, 2, \ldots, NP\}$. φ is the best-so-far fitness value in the current population. pf_i and cf_i are the fitness values of the parent solution and its corresponding offspring, respectively. It is worth noting that ξ is set as zero if $pf_i > cf_i$.

All impacts achieved by the application of mutation operator $a \in \{1, \ldots, K\}$ at generation g are stored in a specific set R_a. At the end of each generation, a unique credit value is assigned to each operator, based on the values stored in R_a:

$$r_a(g) = \sum_{i=1}^{|R_a|} \frac{R_a(i)}{|R_a|}\,. \tag{7}$$

3.2.2. *Operator selection: probability matching*

The operator selection mechanism usually selects the appropriate operator, according to the empirical quality estimates. These estimates are built and constantly updated by the received credit values. In this work, we apply the state-of-the-art operator selection mechanism *Probability Matching* (PM)[21] as follows. Let the set of operators be $S = \{s_1, \ldots, s_K\}$ where $K > 1$. The probability vector $P(g) = \{p_1(g), \ldots, p_K(g)\}(\forall t : p_{\min} \leq p_i(g) \leq 1; \sum_{i=1}^{K} p_i(g) = 1)$ represents the selection probability of each operator at generation g. The current empirical quality estimate of operator i is denoted as $\hat{q}_i(g)$. At each generation g, the ith operator is selected, by roulette wheel selection, according to the probability $p_i(g)$, and obtains a reward $r_i(g)$ from the credit assignment module. The empirical quality estimate of the ith operator $\hat{q}_i(g)$ is updated by using $\hat{q}_i(g+1) = (1-\alpha) \times \hat{q}_i(g) + \alpha \times r_i(g)$, where rate $\alpha \in [0, 1]$.

After the empirical quality estimation, the PM method updates the selection probability as follows:

$$p_i(g+1) = p_{\min} + (1 - K \times p_{\min}) \times \frac{\hat{q}_i(g+1)}{\sum_{i=1}^{K} \hat{q}_i(g+1)} \tag{8}$$

where $p_{\min} \in [0, 1]$ is the minimal selection probability value for each operator. This parameter is used to ensure that all operators will always have a minimal chance of being selected, in order to avoid "losing" a currently bad operator that might become useful at a further moment of the search. Then, any inefficient operator (getting only null credits, i.e., no impact) will have its selection probability converging towards $p_{\min}$, while the best operator (getting very good credits after some time) will be selected with probability $p_{\max} = 1 - (K - 1) \times p_{\min}$.

3.3. *Adaptive parameter control of CR and F*

The adaptive parameter control method used here is inspired by the mechanism used in Ref. 27. Let CR_i^a denotes the crossover rate for the individual i using operator $a \in \{1, \ldots, K\}$. At each generation, CR_i^a is independently generated according to the normal distribution with mean μ_{CR}^a and standard deviation 0.1, CR_i^a is regenerated whenever it exceeds 1. All successful crossover rates at generation g for operator a are stored in a specific set denoted as S_{CR}^a. In particular, the crossover rates, which can successfully generate offspring that survive for the next generation, are considered as available ones. The mean μ_{CR}^a is initialized to be 0.5 and updated after each generation as $\mu_{CR}^a = (1 - w_{cr}) \times \mu_{CR}^a + w_{cr} \times mean(S_{CR}^a)$, where $mean(S_{CR}^a)$ is the arithmetic mean of values in S_{CR}^a and w_{cr} is a random number generated as $w_{cr} = 0.2 \times rndreal()$. The adaptation of the scaling factor F_i^a is similar to that of CR_i^a. At each generation, F_i^a is independently generated according to a Cauchy distribution with location parameter μ_F^a and scale factor 0.1. The value of F_i^a is regenerated whenever F_i^a is out of the range $[0, 1]$. All successful scaling factors at the current generation for operator a are stored in a specific set denoted as S_F^a. The mean μ_F^a is initialized to be 0.5 and independently updated after each generation as $\mu_F^a = (1 - w_f) \times \mu_F^a + w_f \times mean_r(S_F^a)$, where $mean_r(S_{CR}^a)$ is the root-mean-square of values in S_F^a and w_f is a random number generated as $w_f = 0.1 \times rndreal()$.

3.4. *Operator pool*

Many different DE mutation operators have been proposed in this literature,[16] with each one owning its distinctive search behavior. As discussed in Ref. 17, the theoretical studies on the choice of the optimal number of operators in the pool and the organization of the pool with appropriate operators are still open. In this work, we generally maintain an operator pool including four well-known DE mutation operators, i.e., "DE/rand/1" "DE/best/1" "DE/rand/1", "DE/current-to-rand/1", with effective yet diverse characteristics. Due to the page limitation, more detailed description of these four mutation operators can be found in Refs. 9 and 16.

3.5. *System architecture of the evolving ELM paradigm*

The system architecture of the evolving ELM paradigm (denoted as Evo-ELM) based on the proposed AdapDE is shown in Fig. 2. In order to avoid the problem that attributes in greater numeric ranges dominate those in smaller numeric ranges, we adopt the linear scaling to preprocess the input data. Each feature value of the

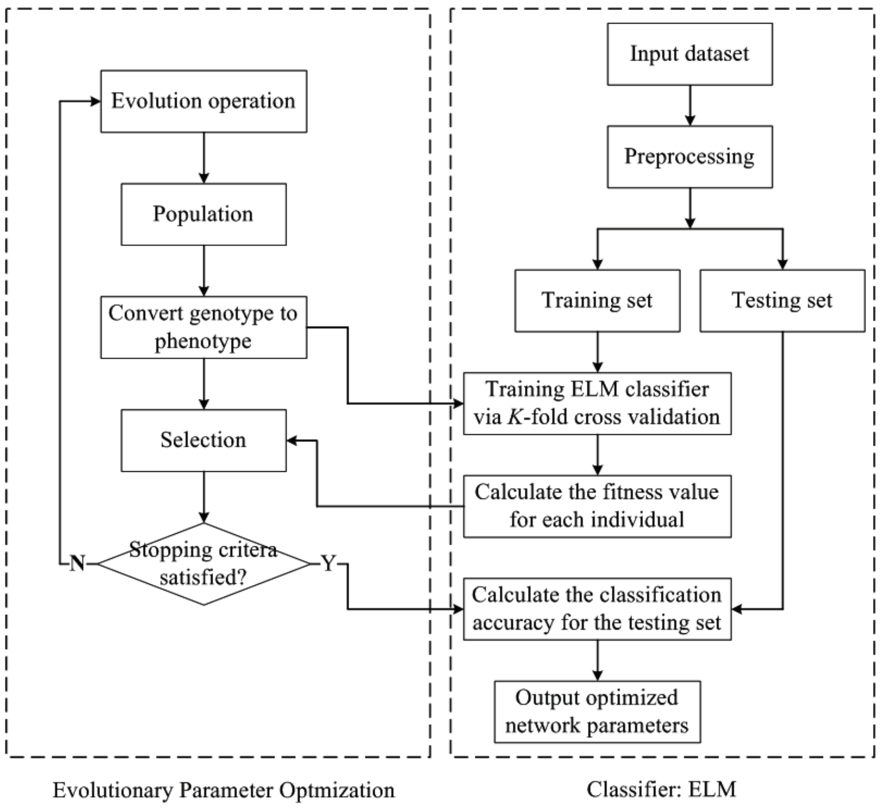
Evolution operation
Population
Convert genotype to phenotype
Selection
Stopping critera satisfied?
N
Y
Input dataset
Preprocessing
Training set
Testing set
Training ELM classifier via K-fold cross validation
Calculate the fitness value for each individual
Calculate the classification accuracy for the testing set
Output optimized network parameters
Evolutionary Parameter Optmization
Classifier: ELM

Fig. 2. System architecture of the proposed evolving ELM paradigm.

data set can be linearly scaled to the range $[0, 1]$ by $v^* = \frac{v - \min}{\max - \min}$, where v is the original value, v^* is the scaled value, max and min is the upper and lower bounds of the feature value, respectively.

4. Performance Verification

In this section, the performance of our proposed evolving ELM paradigm is compared with the original ELM, as well as several other recent classification methods, including evolutionary ELM (E-ELM),[28] voting based ELM (V-ELM),[1] ensemble based ELM (EN-ELM),[12] back-propagation algorithm (BP),[14] radial basis function NN (RBFNN)[2] and support vector machine (SVM).[22] Simulations are conducted on 14 real-world classification data sets[b] as listed in Table 1.

The related parameters of the comparative methods are generally set as follows. As for the proposed evolving ELM paradigm, the population size and the maximum number of generations are all set to be 100; the minimum operator selection

[b]UCI machine learning repository: http://archive.ics.uci.edu/ml/

Table 1. Specification of the classification problems

Data sets	# Classes	# Training samples	# Testing samples	# Features
SPECTF	2	178	45	44
Diabetes	2	576	192	8
Breast cancer	2	379	190	30
Iris	3	100	50	4
Wine	3	118	60	13
Yeast	10	989	495	8
Vowel	11	528	462	10
Soybean	19	307	376	35
Segment	7	1500	810	19
Car	4	1152	576	6
Optdigits	10	3823	1797	64
Pen	10	7494	3498	16
Satellite	6	4435	2000	36
Letter	26	15000	5000	16

probability $p_{\min} = 0.05$ and the adaptation rate $\alpha = 0.3$, as recommended in Ref. 21. As E-ELM is also based on DE algorithm, for the sake of peer comparison, the population size and maximum number of generations are set as the same as our proposed algorithm; while the control parameter of DE in E-ELM, i.e. CR and F, are set constantly to be 0.8 and 1.0, respectively, according to the suggestions in Ref. 28. Furthermore, the number of independent ELMs for voting and constructing ensemble in V-ELM and EN-ELM are all set to be 7, according to the recommendation in Ref. 1. For all the NNs, the number of hidden neurons is set to be 50 and the activation functions are set as sigmoid. As for SVM, the radial basis function kernel is chosen, the cost parameter C and the kernel parameter γ are searched in a grid formed by $C = [2^{12}, 2^{11}, \ldots, 2^{-2}]$ and $\gamma = [2^4, 2^3, \ldots, 2^{-10}]$, and the best combination is then obtained in terms of the generalization performance. Moreover, 30 independent runs are conducted with fixed size of SLFN on each data set, while the mean results and standard deviations are given in Table 2.

4.1. *Empirical results*

From the empirical results shown in Table 2, we can clearly find that our proposed Evo-ELM outperforms the original ELM on all data sets with statistical significance, which fully concludes that the performance of the original ELM has been improved by our network optimization process. As for E-ELM, which also optimizes network parameters by an EA, our proposed Evo-ELM shows significantly better performance than it on 12 data sets out of 14. Besides, V-ELM and EN-ELM are recently proposed ELM variants, both of which perform multiple independent ELM training. It is observed that the performance of EN-ELM is slightly inferior to that of V-ELM, nevertheless, both of them are outperformed by Evo-ELM on most of the cases, while V-ELM only wins on the data set Breast cancer. BP and RBFNN

Table 2. Performance comparisons with ELMs, BP and RBFNN.

Data sets	ELM	E-ELM	V-ELM	EN-ELM	BP	RBFNN	SVM	Evo-ELM
SPECTF	0.6978(9.43E-4)†	0.7150(1.52E-3)†	0.7217(3.02E-4)†	0.7217(1.12E-3)†	0.7502(5.54E-4)†	0.7440(4.91E-4)†	0.7520(3.41E-4)	**0.7585(2.61E-3)**
Diabetes	0.7325(1.67E-3)†	0.7203(3.31E-4)†	0.7342(7.01E-5)†	0.7328(2.25E-4)†	0.7240(4.04E-4)†	0.7366(6.76E-5)†	0.7557(4.01E-4)†	**0.7660(1.28E-3)**
Breast cancer	0.9484(1.19E-4)†	0.9470(1.70E-4)†	0.9607(5.11E-5)	0.9561(7.90E-5)	0.9570(2.50E-5)	**0.9654(2.84E-5)**‡	0.9555(8.64E-5)	0.9599(2.69E-4)
Iris	0.9273(1.31E-3)†	0.9227(1.70E-3)†	0.9240(1.00E-3)†	0.8933(1.31E-3)†	0.9307(4.49E-3)†	0.9616(4.71E-4)	0.9436(2.76E-4)†	**0.9633(1.03E-3)**
Wine	0.9750(6.04E-4)†	0.9500(6.90E-4)†	0.9700(2.38E-4)†	0.9517(5.24E-4)†	0.9778(1.60E-4)†	0.9577(1.16E-4)†	0.9878(2.67E-4)†	**0.9917(2.18E-4)**
Yeast	0.5888(4.59E-4)†	0.5838(3.15E-4)†	0.6018(5.79E-5)	0.5945(5.52E-5)	0.5012(1.34E-2)†	0.6002(9.57E-5)	**0.6128(5.14E-4)**	0.6043(4.54E-4)
Vowel	0.4457(9.11E-4)†	0.3961(9.29E-4)†	0.4973(3.48E-4)†	0.4612(1.20E-3)†	0.4837(1.17E-2)†	0.5089(9.05E-4)†	0.7002(7.99E-4)	**0.7733(8.71E-4)**
Soybean	0.8908(6.23E-4)†	0.8895(5.14E-4)†	0.9206(5.40E-5)†	0.8949(4.40E-4)†	0.6519(7.64E-2)†	0.8570(7.58E-4)†	**0.9867(1.32E-4)**†	0.9421(2.39E-4)
Segment	0.9165(9.29E-5)†	**0.9383(6.89E-5)**‡	0.9291(1.79E-5)	0.9288(4.90E-5)	0.8865(4.29E-2)†	0.9141(7.46E-5)†	0.9212(5.66E-5)†	0.9329(1.98E-4)
Car	0.8684(3.53E-4)†	0.9054(2.46E-4)	0.8646(7.23E-5)†	0.8559(9.48E-5)†	**0.9477(4.52E-3)**‡	0.8064(2.65E-4)†	0.9201(3.15E-4)‡	0.9051(1.60E-4)
Optdigits	0.8261(4.99E-4)†	0.8664(1.40E-4)†	0.9031(6.73E-5)†	0.8300(8.12E-4)†	0.8137(6.76E-2)†	**0.9452(3.14E-5)**‡	0.9210(4.20E-4)	0.9295(7.46E-5)
Pen	0.8240(5.53E-4)†	0.8639(3.03E-4)†	0.8451(8.00E-5)†	0.8065(4.83E-4)†	0.7284(7.93E-2)†	0.8882(1.01E-4)†	0.8911(9.82E-5)†	**0.9151(1.67E-4)**
Letter	0.5039(6.58E-4)†	0.6013(2.10E-4)†	0.6095(1.31E-4)†	0.5359(1.02E-3)†	0.6525(4.89E-2)†	0.3510(9.44E-5)†	0.6710(8.31E-5)†	**0.7146(9.02E-5)**
Satellite	0.5862(4.45E-3)†	0.7389(1.47E-4)†	0.6425(1.17E-5)†	0.6445(1.64E-5)†	0.8242(6.82E-3)‡	0.6365(7.32E-5)†	**0.8301(4.51E-4)**‡	0.7464(1.92E-4)
Statistics	0/14	1/14	0/14	0/14	2/14	2/14	4/14	7/14

The best mean value is highlighted in bold face with dark background. † and ‡ denote the corresponding algorithm is significantly worse than and better than that of the proposed Evo-ELM, respectively, according to the Wilcoxon's rank sum test at a 0.05 significance level.

are two classical NNs whose underlying mechanisms are different from that of ELM. The comparative results of them are given in the fifth and sixth columns of Table 2, respectively. Generally speaking, our proposed Evo-ELM still outperforms them on most of the data sets except for some special cases, i.e., BP outperforms Evo-ELM on Car and Satellite, RBFNN outperforms Evo-ELM on Breast cancer and Optdigits with statistical meaning. Finally, as shown in the seventh column of Table 2, SVM shows better performances on 4 data sets out of 14, with 2 significant results. Due to the page limitation, further investigations of the underlying mechanism of AdapDE are presented in the supplementary document of this paper.

5. Conclusion

In this paper, an evolving ELM paradigm that integrates ELM and DE is presented for classification problems. Specifically, a DE variant (AdapDE) with adaptive operator selection and parameter control is proposed to optimize the network parameters of the baseline ELM. AdapDE is able to online select the appropriate operator for offspring generation, while the control parameters can be adjusted in an adaptive manner. From the empirical studies on several real-world classification data sets, it is clear that not only the performance of the baseline ELM is improved, but also the proposed algorithm can outperform several recent classification methods.

In future, more advanced adaptive operator selection mechanism, such as multi-armed bandit proposed in Ref. 9, can be considered in AdapDE. Furthermore, other evolution operators, such as GA, jumping genes[11] and guided mutation operators[10] can also be considered in building the operator pool.

References

1. J.-W. Cao, Z.-P. Lin, G.-B. Huang and N. Liu, Voting based extreme learning machine, *Information Sciences* **185**(1) (2012) 66–77.
2. P. E. Hart and D. G. Stork, *Pattern Classification* (2001).
3. S. He, Q. H. Wu and J. R. Saunders, Group search optimizer: An optimization algorithm inspired by animal searching behavior, *IEEE Trans. Evolutionary Computation* **13**(5) (2009) 973–990.
4. G.-B. Huang, D.-H. Wang and Y. Lan, Extreme learning machines: a survey, *Int. J. Machine Learning and Cybernetics* **2** (2011) 107–122.
5. G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, Extreme learning machine: a new learning scheme of feedforwad neural networks, in *IJCNN'04: Proc. 2004 Int. Joint Conf. on Neural Network*, July 2004, pp. 25–29.
6. G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, Extreme learning machine: theory and applications, *Neurocomputing* **70** (2006) 489–501.
7. W. Jun, S.-T. Wang and F.-L. Chung, Positive and negative fuzzy rule system, extreme learning machine and image classification, *Int. J. Machine Learning and Cybernetics* **2** (2011) 261–271.
8. K. Li, Á. Fialho and S. Kwong, Multi-objective differential evolution with adaptive control of parameters and operators, in C. A. Coello Coello (ed.), *LION'11: Proc. 5th*

Int. Conf. on Learning and Intelligent Optimization, Springer LNCS, January 2011, pp. 473–487.

9. K. Li, Á. Fialho, S. Kwong and Q.-F. Zhang, Adaptive operator selection with bandits for multiobjective evolutionary algorithm based decomposition, *IEEE Trans. Evolutionary Computation*, in press.
10. K. Li, S. Kwong, R. Wang, J.-J. Cao and I. J. Rudas, Multi-objective differential evolution with self-navigation, in *SMC'12: Proc. 2012 IEEE Int. Conf. Systems, Man, and Cybernetics*, Seoul, South Korea, October 2012, IEEE, pp. 508–513.
11. K. Li, S. Kwong, R. Wang, K.-S. Tang and K.-F. Man, Learning paradigm based on jumping genes: A general framework for enhancing exploration in evolutionary multiobjective optimization, *Information Sciences* **226** (2013) 1–22.
12. N. Liu and H. Wang, Ensemble based extreme learning machine, *IEEE Signal Processing Letters* **17**(8) (2010) 754–757.
13. Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten and A. Lendasse, OP-ELM: Optimally Pruned Extreme Learning Machine, *IEEE Trans. Neural Networks* **21**(1) (2010) 158–162.
14. M. F. Møller, A scaled conjugate gradient algorithm for fast supervised learning, *Neural Networks* **6**(4) (1993) 525–533.
15. Y.-S. Ong, M.-H. Lim, N. Zhu and K. W. Wong, Classification of adaptive memetic algorithms: a comparative study, *IEEE Trans. Systems, Man, and Cybernetics, Part B* **36**(1) (2006) 141–152.
16. K. Price, R. Storn and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization* (Springer-Verlag, Berlin, Germany, 2005).
17. A. K. Qin, V. L. Huang and P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Trans. Evolutionary Computation* **13**(2) (2009) 398–417.
18. S. Saraswathi, S. Sundaram, N. Sundararajan, M. Zimmermann and M. Nilsen-Hamilton, Icga-pso-elm approach for accurate multiclass cancer classification resulting in reduced gene sets in which genes encoding secreted proteins are highly represented, *IEEE/ACM Transactions Computational Biology and Bioinformatics* **8**(2) (2011) 452–463.
19. P. Sarlin, Visual tracking of the millennium development goals with a fuzzified self-organizing neural network, *Int. J. Machine Learning and Cybernetics* **3** (2012) 233–245.
20. D. N. G. Silva, L. D. S. Pacifico and T. B. Ludermir, An evolutionary extreme learning machine based on group search optimization, in *CEC'11: Proc. 2011 IEEE Congress on Evolutionary Computation*, 2011, pp. 574–580.
21. D. Thierens, An adaptive pursuit strategy for allocating operator probabilities, in *GECCO'05: Proc. 2005 Conf. Genetic and Evolutionary Computation*, ACM, New York, USA, 2005, pp. 1539–1546.
22. V. N. Vapnik, *Statistical Learning Theory* (Wiley, New York, 1998).
23. X.-Z. Wang, Q. Y. Shao, Q. Miao and J.-H. Zhai, Architecture selection for networks trained with extreme learning machine using localized generalization error model, *Neurocomputing* **102** (2013) 3–9.
24. X.-Z. Wang, D. S. Yeung and E. C.C. Tsang, A comparative study on heuristic algorithms for generating fuzzy decision trees, *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics* **31**(2) (2001) 215–226.
25. Y. Xu and Y. Shu, Evolutionary extreme learning machine-based on particle swarm optimization, in *ISNN(1)*, 2006, pp. 644–652.

26. J.-H. Zhai, H.-Y. Xu and X.-Z. Wang, Dynamic ensemble extreme learning machine based on sample entropy, *Soft Computing* **16**(9) (2012) 1493–1502.
27. J.-Q. Zhang and A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, *IEEE Trans. Evolutionary Computation* **13**(5) (2009) 945–958.
28. Q.-Y. Zhu, A. K. Qin, P. N. Suganthan and G.-B. Huang, Evolutionary extreme learning machine, *Pattern Recognition* **38**(10) (2005) 1759–1763.