# Self-adaptive Differential Evolution Extreme Learning Machine for the Classification of Hyperspectral Images

Junhua Ku[12]
[1]Department of Information Engineering, Hainan Institute of Science & Technology, Haikou ,China
[2]School of Computer Science China University of Geosciences Wuhan, China
e-mail: kujunhua@163.com

Zhihua Cai
School of Computer Science China University of Geosciences Wuhan, China

Xiuying Yang
Department of Information Engineering, Hainan Institute of Science & Technology, Haikou ,China

**Abstract—In this paper, we propose an efficient classification method for hyperspectral images based on the extreme learning machine (ELM) and self-adaptive differential evolution (jDE).The approach of ELM is characterized by a unified formulation for regression, binary, and multiclass classification problems, and the related solution is given in an analytical compact form. In order to address the selection issue that is associated with the ELM, we have developed an automatic method to solve the model selection issue that is associated with this classifier based on the jDE optimization. The self-adaptive control mechanism is used to change control parameters, i.e. select weighting factor F and crossover constant CR, during the run. This simple yet powerful evolutionary optimization algorithm uses cross-validation accuracy as a performance indicator for determining the optimal ELM parameters. Experimental results obtained from hyperspectral data set confirm the attractive properties of the proposed jDE-ELM method in terms of classification accuracy and computation time.**

***Keywords- Differential Evolution ; Self-adaptive ; Extreme Learning Machine ; Hyperspectral Images ; Machine Learning***

## I. INTRODUCTION

The issue of the supervised classification of hyperspectral images has attracted much researchers, the current objective for researchers is to further boost the classification accuracy [1]. Therefore, one can discern that an important direction of the research is related to the choice of classification approach. We know that the support vector machines (SVMs) learning method is popular. Recently, the extreme learning machine (ELM) has been introduced in the literature [2,3]. The ELM aims at minimizing the training error and the norm of the output weights. The ELM is characterized by a unified formulation for binary, multiclass, and regression problems.

In this paper, we extend ELMs to handle for the classification of hyperspectral images based on the self-adaptive Differential Evolution. This simple yet powerful evolutionary optimization algorithm uses cross-validation accuracy as a performance indicator for determining the optimal ELM parameters. Due to the hyperspectral images are characterized by high dimensional spectral features, we first apply feature reduction [(such as the principal component analysis (PCA)] to reduce the dimensionality of the data. Then, in the second step, we apply morphological operations (opening and closing operations with reconstruction) to these features to generate an extra set of MP features. To address the model selection issue that is associated with the ELM, simple grid selection procedures could be used.

## II. PROPOSED CLASSIFICATION METHOD

### *A. Classification With ELM*

Let $D = \{(x_i, y_i)\}_{i=1}^{N}$ be the training set that is composed of N training feature vectors $x_i$ of dimension $d$, and $y_i \in \{1, \cdots, P\}$ are class labels. $P$ represents the number of classes. Each vector $x_i$ is composed of two parts: the $N_{PCA}$ features obtained by applying the PCA to the original spectral features and the MP features obtained by applying the opening and closing operations with reconstruction to these $N_{PCA}$ features with a structural element of different sizes.

The $j$th output of a multiclass ELM classifier with $P$ output nodes is given by

$$f_j(\mathrm{x}) = \mathrm{h}(\mathrm{x})\mathrm{w}_j,\ j = 1,\ \cdots,\ P \tag{1}$$

where $\mathrm{w}_j \in R^L$ is the vector of the output weights between the hidden layer of L nodes and the $j$th output node. In the case of binary classification, the ELM is characterized by one output node. $\mathrm{h}(\mathrm{x}) \in \Re^L$ is the (row) output vector of the hidden layer with respect to input x. It maps the input data from the d-dimensional space to the L-dimensional ELM feature space. This feature mapping could be done in a finite space, such as in standard neural network

classifiers, or in an infinite space by applying the kernel trick, as shown later.

The $l_2$ norm optimization problem that is associated with the ELM is given as follows:

$$\text{Minimize } L_{Primal} = \frac{1}{2}\|\mathrm{w}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2$$

$$\text{Subject to } \mathrm{h}(\mathrm{x}_i)\mathrm{w} = \eta_i^T - \xi_i^T, i == 1, \cdots, N \quad (2)$$

where $C$ is a regularization parameter. $\mathrm{w} = [\mathrm{w}_1, \ldots, \mathrm{w}_P]$ is a matrix of size $P \times L$ formed by staking the vectors of output weights $\mathrm{w}_j, j = 1, \ldots, P$ . $\eta_i = [\eta_{i1}, \ldots, \eta_{iP}]^{\mathrm{T}}$ and $\xi_{\mathrm{i}} = [\xi_{\mathrm{i1}}, \ldots, \xi_{\mathrm{iP}}]^{\mathrm{T}}$ are the target and training error vectors of the P output nodes, respectively, with respect to the training sample $\mathrm{x}_i$. Target vector $\eta$ has all its values set to 0, except the entry that matches the class label $y_i$, which is set to 1. Based on the Karush–Kuhn–Tucker (KKT) theorem, the determination of the weights training an ELM is equivalent to solving the following dual optimization problem:

$$L_{Dual} = \frac{1}{2}\|\mathrm{w}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 - \sum_{i=1}^{N}\sum_{j=1}^{P}\alpha_{ij}(\mathrm{h}(\mathrm{x}_i)\mathrm{w}_j - \eta_{ij} - \xi_{ij}) \quad (3)$$

It can be shown that by taking the KKT optimal conditions, the optimal vector of weights $\mathrm{w}^*$ can be given by the following compact matrix form [4]:

$$\mathrm{w}^* = \mathrm{H}^T\left(\frac{1}{C} + \mathrm{H}\mathrm{H}^T\right)^{-1}\eta \quad (4)$$

where H is the hidden-layer output matrix, and it is defined as follows:

$$\mathrm{H} = \begin{bmatrix} \mathrm{h}(\mathrm{x}_1) \\ \vdots \\ \mathrm{h}(\mathrm{x}_N) \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} h_1(\mathrm{x}_1) & \cdots & h_L(\mathrm{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathrm{x}_N) & \cdots & h_L(\mathrm{x}_N) \end{pmatrix} \end{bmatrix} \quad (5)$$

η is a matrix of size N × P built from target output vector $\eta_i^T$ as in the following:

$$\eta = \begin{bmatrix} \eta_1^T \\ \vdots \\ \eta_N^T \end{bmatrix} = \begin{pmatrix} \eta_{11} & \cdots & \eta_{1P} \\ \vdots & \ddots & \vdots \\ \eta_{N1} & \cdots & \eta_{NP} \end{pmatrix} \quad (6)$$

and E is an identity matrix of size N × N. Hence, the output function of the ELM is

$$f(\mathrm{x}) = \mathrm{h}(\mathrm{x})\mathrm{w}^* = \mathrm{h}(\mathrm{x})\mathrm{H}^T\left(\frac{1}{C} + \mathrm{H}\mathrm{H}^T\right)^{-1}\eta \quad (7)$$

During the prediction phase, test sample $x_\ell$ will be assigned to the index of the output node that has the highest value. In other words, if we let $f(x_\ell) = [f_1(x_\ell), \cdots, f_P(x_\ell)]^T$ , then the predicted class label for test sample $\mathrm{x}_\ell$ is

$$y_\ell^* = \text{argmax}_{j\in\{1,\ldots,P\}} f_j(\mathrm{x}_\ell) \quad (8)$$

In the kernel space, the prediction that is associated with test sample $\mathrm{x}_\ell$ is given in the following compact form:

$$f(\mathrm{x}_\ell) = \begin{bmatrix} k(\mathrm{x}_\ell, \mathrm{x}_1) \\ \vdots \\ k(\mathrm{x}_\ell, \mathrm{x}_N) \end{bmatrix}^T \left(\frac{1}{C} + K\right)^{-1}\eta \quad (9)$$

The first term of (9) is a vector of length $N$, and it represents the kernel distances between test point $\mathrm{x}_\ell$ and the training samples. In this case, the number of hidden neurons L need not be given as kernel matrix $\mathrm{K} = \mathrm{H}\mathrm{H}^{\mathrm{T}}$ , which is only related to the training samples. It is not relevant to the number of output nodes and to the training target values. From (9), one can clearly see that training and predicting with the ELM is done in a simple way. See [5] for a detailed description of the ELM and its comparison with other classifiers.

*B. Model Selection With jDE (jDE-ELM)*

DE is a powerful evolutionary algorithm for global numeric optimization [6,7,8],which is much simpler and straightforward to implement. The population of the original DE algorithm (contains *NP* individuals. An individual is defined as a *D*-dimensional vector. If *G* denotes the generation, the population at generation *G* consists of:

$$\mathrm{v}_{i,G} = \{v_{i,1,G}, v_{i,2,G}, v_{i,\mathrm{D},G}\}, i = 1, 2, \ldots, NP \quad (10)$$

During one generation for each vector $\mathrm{x}_{i,G}$ , DE employs mutation and crossover operations to produce a trial vector:

$$\mathrm{q}_{i,G} = \{q_{i,1,G}, q_{i,2,G}, q_{i,\mathrm{D},G}\}, i = 1, 2, \ldots, NP \quad (11)$$

Then a selection operation is used to choose vectors for the next generation (*G*+1).The initial population is usually selected uniformly randomly between the lower $x_{j,low}$ and upper $x_{j,upp}$ bounds defined for each variable $xj$ . These bounds are specified according to the nature of the problem.

*a) Mutation operation*

Mutation for each population vector $\mathrm{v}_{i,G}$ creates a mutant vector $\mathrm{u}_{i,G}$ :

$$\mathrm{u}_{i,G} = \{u_{i,1,G}, u_{i,2,G}, u_{i,\mathrm{D},G}\}, i = 1, 2, \ldots, NP \ . \quad (12)$$

A new mutant vector can be created using the mutation strategy.

$$\mathrm{u}_{i,G} = \mathrm{v}_{r_1,G} + F(\mathrm{v}_{r_2,G} - \mathrm{v}_{r_3,G}) \quad (13)$$

*b)Crossover operation*

After mutation, a 'binary' crossover operation forms the trial vector $\mathrm{q}_{i,G}$ according to the target vector $\mathrm{v}_{i,G}$ and its corresponding mutant vector $\mathrm{u}_{i,G}$

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if } rand(0,1) \le CR \text{ or } j = j_{rand} \\ x_{i,j,G} & \text{otherwise} \end{cases} \quad (14)$$

$$i = 1,2, ...,NP \text{ and } j = 1,2, ...,D.$$

where $CR$ is a crossover control parameter within the range [0,1) and presents the probability of creating parameters for a trial vector from the mutant vector. Index $j_{rand}$ is a randomly chosen integer within the range [1,$NP$]. It ensures that the trial vector contains at least one parameter from the mutant vector.

*c)Selection operation*

The selection operator selects between the target and corresponding trial vectors. we will use the following selection rule:

$$v_{i,G+1} = \begin{cases} q_{i,G} & \text{if } f(q_{i,G}) \le f(v_{i,G}) \\ v_{i,G} & \text{otherwise} \end{cases} \quad (15)$$

*C. Self-adaptive differential evolution*

The self-adaptive differential evolution (jDE) algorithm, based on the self-adapting control parameter mechanism, was proposed by [7]. The self-adaptive control mechanism is used to change control parameters, i.e. select weighting factor $F$ and crossover constant $CR$, during the run. The self-adaptive control parameters $F_{i,G+1}$ and $CR_{i,G+1}$ are calculated as follows:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \cdot F_u & \text{if } rand_2 \le \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases},$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 \le \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases} \quad (16)$$

They produce control parameters $F$ and $CR$ in a new parent vector. The quantities $rand_j$, $j \in \{1,2,3,4\}$ represent uniform random values within the range [0,1].Constants $\tau_1, \tau_2, F_l, F_u$ are assigned fixed values to 0.1,0.1,0.1,0.9,respectively. The control parameter values $F_{i,G+1}$ and $CR_{i,G+1}$ are obtained before the mutation operation is performed. This means, they influence the mutation, crossover, and selection operations of the new vector $v_{i,G+1}$.

In this paper, we propose to apply the jDE to solve the model selection problem of the ELM. To this end, the target vector represents the regularization and kernel parameters. In the case of the radial basis function kernel, this vector is given as follows:

$$v_i = [C\ \gamma]\ . \quad (17)$$

As the objective function, we propose to minimize the widely used cross-validation error measure, as in the following:

$$g(v_i) = CV_{err}. \quad (18)$$

Such error is computed by splitting the training set during the training phase into k-folds and then by training the ELM on k-1 folds and computing the error on the remaining fold. This operation is done for all possible fold combinations, and then, the average error is taken [9]. We provide the main steps of the proposed jDE-ELM algorithm in the following.

**Algorithm: jDE-ELM**

Input: - Training set D

- DE parameters: NP, and the maximum number of function evaluations $FES_{max}$

Output: - Classification result

Step 1) Initialization:

Step 1.1) Set the index of generations G = 0

Step 1.2) Generate random target vectors $v_{i,G}$, i=1, . . . , NP from solution space to form an initial population of size NP.

Step 1.3) For each vector $v_{i,G}$, run an ELM classifier on the training set and compute the corresponding objective function $g(v_{i,G})$. Set the number of function evaluations FES = NP.

Step 2) jDE

for i = 1 to NP, do

uniformly choose random $r_1 \neq r_2 \neq r_3 \neq i$ //self-adaptive parameters setting

If $rndreal_j[0,1] < \tau_1$ then

$CR_i = rndreal_j[0,1]$ (Generate new $CR_i$)

Endif

If $rndreal_j[0,1] < \tau_2$ then

$F_i = rndreal_j[0,1]$ (Generate new $F_i$)

Endif

$j_{rand} = rndint(1,D)$

for j=1 to D do //crossover step

if $rndreal_j[0,1] < CR_i$ or $j = j_{rand}$ then

$q_i(j) = x_{r_1,G} + F_i \times (x_{r_2,G} - x_{r_3,G})$ //mutation step

else

$q_i(j) = x_i(j)$

end if

end for

end for

for i = 1 to NP do //selection step

if $g(q_i) \le g(v_i)$, then $v_{i,G+1} = q_i$

else $v_{i,G+1} = v_{i,G}$

end if

end for

Step 3) Set FES = FES + NP

Step 4) if FES<FESmax then Go to Step 6 Else G = G + 1 and return to step 2

Step 5) Select the optimal vector $v_i^*$ corresponding to the minimum objective function $g(v_i^*)$.

Step 6) Train the jDE-ELM using the optimal parameter vector $v_i^*$, and compute the decision function for test sample $x_\ell$ according to (9).

## III. Experimental results

To assess the effectiveness of the jDE-ELM classification method, low spatial resolution and high spatial resolution Washington, DC data set is used in the experiments, as shown in Table 1. See [10] for a detailed description of these data set.

TABLE I. Washington, DC hyperspectral data set

| Image size | Spatial Res. | #bands | (Train,Test) | #Classes |
|---|---|---|---|---|
| 1280*307 | 2m | 191 | (350,56919) | 7 |

In the experiment, for the data set, we consider 50 training samples per class, and the remaining samples are left for the test. For the data set, we repeat the experiments ten times with different training and test samples, and then, we present the averaged results in terms of the overall (OA) and average (AA) standard deviation, i.e., σOA and σAA, respectively. In addition to these measures, we also use McNemar's statistical test [9] to compare the ELM performances with the SVM from a statistical point of view. For the SVM. Table 2 lists the classification results obtained by both classifiers using the spectral, $MP^{PCA(5)}$, and $MP^{PCA(10)}$ features.

TABLE II Classification results obtained for the high spatial resolution data set of Washingto,DC

| Features | jDE-ELM | | DE-ELM | | DE-SVM | |
|---|---|---|---|---|---|---|
| | $OA \pm \sigma_{OA}$<br>$AA \pm \sigma_{AA}$ | Time [s] | $OA \pm \sigma_{OA}$<br>$AA \pm \sigma_{AA}$ | Time [s] | $OA \pm \sigma_{OA}$<br>$AA \pm \sigma_{AA}$ | Time [s] |
| Spectral | 85.27±1.33<br>81.22±0.68 | 37 | 85.77±1.63<br>81.12±0.73 | 8 | 84.68±1.46<br>80.62±1.72 | 170 |
| $MP^{PCA(5)}$ | 95.47±0.67<br>93.15±0.29 | 19 | 95.49±0.63<br>93.91±0.33 | 5 | 95.17±0.66<br>93.16±0.26 | 36 |
| $MP^{PCA(10)}$ | 96.30±0.54<br>94.78±0.26 | 26 | 96.27±0.76<br>94.62±0.38 | 7 | 95.77±0.65<br>94.14±0.45 | 64 |

.

As for the ELM, we implement analytical solution (9) using a few lines of codes also in MATLAB 2013b. The term $(I/C + K)^{-1}\eta$ is computed using the matrix division operator $(I/C + K) \setminus \eta$, which produces the solution by using Gaussian elimination. The division operator "\" produces a solution that is two to three times faster than the command "inv". In the experiments, for both classifiers, we adopt the common Gaussian kernel $k(xi, x) = exp(-\gamma \|xi - x\|^2)$ ,where γ represents a parameter that is inversely proportional to the width of the Gaussian kernel. For a fair comparison, we also tune the parameters of the SVM C and γ using the DE. We set the parameters of the DE as follows: population size=10, number of function evaluation FESmax = 100, and CR and F to the standard value of 0.9.We set the search boundary of C and γ in the ranges [10−3, 1000] and [10−3, 10], respectively. To compute objective function $g(v_i)$ (i.e., the cross-validation accuracy), we set the number of folds k = 3.

From these results, one can see that the jDE-ELM and DE-ELM are statistically better with respect to the DE-SVM in several cases. In terms of computation time, the results show that the jDE-ELM and DE-ELM are faster than the DE-SVM.

## IV. Conclusion

In this paper, we propose an efficient classification method for hyperspectral images based on the ELM and MP features. In particular, we have developed an automatic method to solve the model selection issue that is associated with this classifier based on the jDE optimization. The experimental results obtained by the proposed jDE-ELM on hyperspectral data set shows that the jDE-ELM provides better classification accuracy than SVM, and that it is faster as its solution is simple, only requiring the inversion of a kernel matrix that is computed from the training samples.


## Acknowledgment

This work was supported by the Natural Science Foundation of Hainan province under Grant No. 614248, the Young and middle-aged scientific research projects of Hainan Institute of Science and Technology under grant No.HKYZQJ2014-05.

## REFERENCES


[1] N. Alajlan, Y. Bazi, F. Melgani, and R. R. Yager, "Fusion of supervised and unsupervised learning paradigms for improved classification of hyperspectral images," Inf. Sci., vol. 217, pp. 39–55, Dec. 2012.

[2] G. B. Huang, L. Chen, and S. K. Siew, "Universal approximation using incremental constructive feedforward neural networks with random hidden nodes," IEEE Trans. Neural Netw., vol. 17, no. 4, pp. 879–892, Jul. 2006.

[3] J. Cao, Z. Lin, and G. B. Huang, "Self-adaptive evolutionary extreme machine learning," Neural Process. Lett., vol. 36, no. 3, pp. 285–305, Dec. 2012.

[4] N. Alajlan, Y. Bazi, H. AlHichri, F. Melgani, and R. R. Yager, "Using OWA operators for the classification of hyperspectral images," IEEE J.Sel. Topics Appl. Earth Obser. Remote Sens., vol. 6, no. 2, pp. 602–614,Apr. 2013.

[5] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 42, no. 2, pp. 513–529, Apr. 2012.

[6] ]S. Das and P. N. Suganthan, "Differential evolution: A survey of state-of-the-art," IEEE Trans. Evol. Comput., vol. 15, no. 1, pp. 4–31, Feb. 2011.

[7] Brest J, Greiner S, Bo˘skovi´c B, Mernik M, ˇ Zumer V. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Transactions on Evolutionary Computation 10(6):646–657.

[8] Brest J, ˇ Zumer V, Mau˘cec MS.Self-adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization.In: The 2006 IEEE Congress on Evolutionary Computation CEC2006, IEEE Press, pp 919–926.

[9] Y. Bazi and F. Melgani, "Toward an optimal SVM Classification system for hyperspectral remote sensing images," IEEE Trans. Geosci. Remote Sens., vol. 44, no. 11, pp. 3374–3385, Nov. 2006.

[10] J. Ham, Y. Chen, M. M Crawford, and J. Ghosh, "Investigation of the random forest framework for classification of hyperspectral data," IEEE Trans. Geosci. Remote Sens., vol. 43, no. 3, pp. 492-501,Mar.2005.