Les Cahiers du GERAD ISSN: 0711–2440

DGR-ELM – Distributed generalized regularized ELM for classification

F.K. Inaba, E.Ottoni Teatini Salles,
S. Perron, G. Caporossi

G–2016–32

May 2016

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

DGR-ELM – Distributed generalized regularized ELM for classification

Fernando Kentaro Inaba [a]
Evandro Ottoni Teatini Salles [a]
Sylvain Perron [b]
Gilles Caporossi [b]

[a] *Federal University of Espírito Santo, Vitória, Brazil*

[b] *GERAD & HEC Montréal, Montréal (Québec) Canada*

entaro@ele.ufes.br
evandro@ele.ufes.br
sylvain.perron@gerad.ca
gilles.caporossi@gerad.ca

May 2016

Les Cahiers du GERAD
G–2016–32

Abstract: Extreme Learning Machine (ELM) has recently increased popularity and has been successfully applied to a wide range of applications. Variants using regularization are now a common practice in the state of the art in ELM field. The most commonly used regularization is the ℓ_2 norm which improves generalization but result in a dense network. Regularization based on the elastic net has also been proposed but mainly applied to regression and binary classification problems. In this paper, we propose a generalization of regularized ELM (R-ELM) for multiclass classification problems, termed GR-ELM. We achieve such generalization using the $\ell_{2,1}$ and Frobenius norm. Traditional R-ELM is a particular case of our method when binary classification tasks are considered. We also propose an alternative algorithm for GR-ELM when training data is distributed, namely GR-ELM. We use alternating direction method of multipliers (ADMM) for solving the resulting optimization problems. Message passing interface (MPI) in a single program, multiple data (SPMD) programming style is chosen for implementing DGR-ELM. Extensive experiments are conducted to evaluate the proposed method. Our experiments show that GR-ELM and DGR-ELM have similar training and testing accuracy when compared to R-ELM, although usually faster testing time is obtained with our method due to the compactness of the resulting network.

Key Words: $\ell_{2,1}$ norm, Regularization, Extreme Learning Machine, Muticlass Classfication, Alternating direction method of multipliers

Acknowledgments: The authors wish to acknowledge the support of the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES) through Grant 88881.062156/2014-01. Fernando Inaba is supported by a scholarship from the Brazilian Council for Development of Science and Technology (CNPq).

1 Introduction

Extreme Learning Machine (ELM) proposed in [1, 2] and its variants [3] has recently increased popularity and has been successfully used in a wide range of applications such as computer vision, image processing, time series analysis and biomedical applications [3]. In such research areas, ELM can achieve good generalization performance while maintaining low computational cost, justifying its popularity. The main idea of ELM is to generate, randomly, the input weights of a single hidden layer feedforward neural network (SLFN) and then deterministically find the output weights.

In ELM [2], one of the parameters that need to be well chosen is the number of neurons in the hidden layer to achieve a good underfitting/overfitting trade-off. To overcome this, Deng et al. [4] proposed a regularized version of ELM using ridge regression theory. Although this approach achieves good generalization, the resulting network is dense and usually requires more storage space and testing time for large-scale applications [5]. Also, for large-scale learning tasks, traditional and ridge regularized ELM might suffer from the limitation of memory and intensive computational cost of large matrices inversion [6].

Using elastic net theory, Martínez-Martínez et al. [7] proposed the regularized extreme learning machine (R-ELM) for automatic selection of the architecture of ELM networks in regression problems. For binary classification, an extension of R-ELM should be straightforward. For multiclass problems, approaches using one-versus-rest classifiers can be adopted [1, 8]. However, a larger number of classes usually require more computation time in training step. Although using elastic net penalty (R-ELM) usually result in a compacter network, it suffers from the same drawbacks as ridge regularized ELM for large-scale learning tasks.

Recently, Wang et al. [6] proposed the parallel regularized ELM (PR-ELM) to improve the computational efficiency of ELM in handling large scale tasks. In PR-ELM, the dataset is split into several small chunks that are treated on different computational nodes and avoid the computationally intense procedure of large-scale matrix multiplication and inverse operations. However, PR-ELM considers only ridge penalty ELM and the resulting network is dense and usually require more testing time.

In this paper, we generalize the R-ELM for multiclass classification problems. We use the combination of the Frobenius norm and $\ell_{2,1}$ norm of the output weights as a penalty for ELM. In the binary classification problems, our approach is the same as elastic net penalty ELM used in R-ELM. As a consequence, with appropriate choice of regularization parameters, we have ridge penalty ELM as a particular case of our method. To solve our optimization problem, we use alternating direction method of multipliers (ADMM) [9] for implementation. ADMM is a simple but powerful algorithm that solves a large global problem through solutions of small local subproblems [9]. Furthermore, since ADMM is well suited to distributed convex optimization, we extend our method for large-scale learning tasks using global variable consensus with regularization. Such extension is accomplished using message passing interface (MPI) in a single program, multiple data (SPMD) programming style.

The paper is organized as follows. In Section 2 we briefly review the background of ELM and related variants using regularization. We describe the GR-ELM and DGR-ELM in Section 3. Experimental results are presented in Section 4, and Section 5 concludes the paper.

2 Related work

In this section, we first brief review the extreme learning machine. Then we present some existing variations, in specific the regularized ones, of the ELM.

2.1 ELM

Huang et al. [2] proposed ELM for SFLNs (Single-Hidden Layer Feedforward Neural Networks) which only the output weights needs to be determined. This is done by randomly choosing the hidden nodes parameters so one can analytically obtain the output weights using Moore-Penrose's generalized inverse.

For a given N samples $(\boldsymbol{x}_k, \boldsymbol{t}_k)$, where $\boldsymbol{x}_k \in \mathbb{R}^n$ is the input and $\boldsymbol{t}_k \in \mathbb{R}^m$ the target, the SFLNs with $\widetilde{N}$ hidden nodes and activation function $h(\cdot)$ is described as

$$\boldsymbol{o}_k = \sum_{i=1}^{\widetilde{N}} \boldsymbol{\beta}_i h(\boldsymbol{x}_k; \boldsymbol{a}_i, \nu_i), \quad k = 1, \ldots, N \tag{1}$$

where $\boldsymbol{a}_i \in \mathbb{R}^n$ and $\nu_i \in \mathbb{R}$ are, in the context of ELM, the randomly assigned parameters of the i-th hidden node, $\boldsymbol{\beta}_i \in \mathbb{R}^m$ is the weight vector connecting the i-th hidden node to the output node.

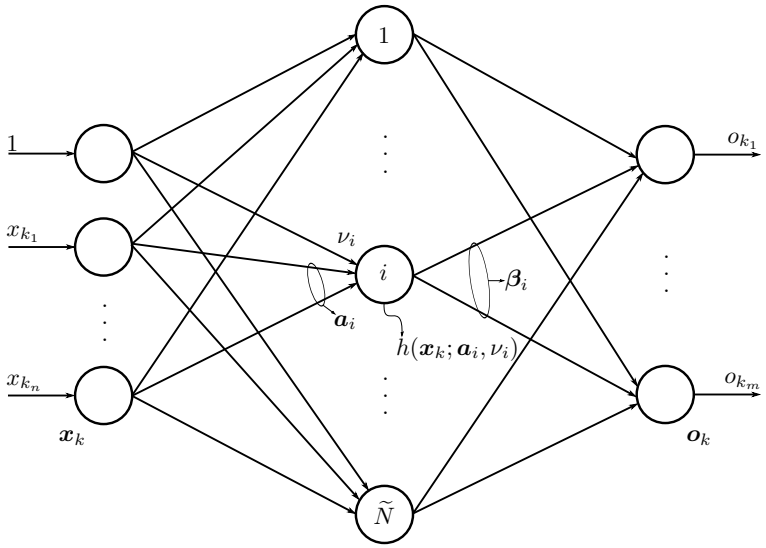

Figure 1: Illustration of an architecture of ELM with n inputs, $\widetilde{N}$ hidden nodes and m outputs.

Figure 1 illustrates a typical architecture for ELM for multiclass classification problems with m classes, n inputs, and $\widetilde{N}$ hidden nodes.

Let $\boldsymbol{B} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_{\widetilde{N}})^\mathsf{T}$, $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{\widetilde{N}})^\mathsf{T}$ and

$$\boldsymbol{H}(\boldsymbol{X}; \boldsymbol{A}, \boldsymbol{\nu}) = \begin{bmatrix} h(\boldsymbol{x}_1; \boldsymbol{a}_1, \nu_1) & \cdots & h(\boldsymbol{x}_1; \boldsymbol{a}_{\widetilde{N}}, \nu_{\widetilde{N}}) \\ \vdots & \ddots & \vdots \\ h(\boldsymbol{x}_N; \boldsymbol{a}_1, \nu_1) & \cdots & h(\boldsymbol{x}_N; \boldsymbol{a}_{\widetilde{N}}, \nu_{\widetilde{N}}) \end{bmatrix}, \tag{2}$$

$\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^\mathsf{T}$, $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{\widetilde{N}})^\mathsf{T}$ and $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_{\widetilde{N}})^\mathsf{T}$.

If an SLFN with $\widetilde{N}$ hidden nodes can approximate the N given samples $(\boldsymbol{x}_k, \boldsymbol{t}_k)$ with zero error, i.e. $\boldsymbol{o}_k = \boldsymbol{t}_k$, Equation 1 can be expressed compactly as

$$\boldsymbol{H}\boldsymbol{B} = \boldsymbol{T}. \tag{3}$$

The output weights can be obtained following the smallest norm least-squares solution $\boldsymbol{B} = \boldsymbol{H}^\dagger \boldsymbol{T}$, where $\boldsymbol{H}^\dagger$ is the Moore-Penrose's generalized inverse [10] of the matrix. The Moore-Penrose's generalized inverse can be calculated using orthogonal projection method [10] in two cases: when $\boldsymbol{H}^\mathsf{T}\boldsymbol{H}$ is nonsingular and $\boldsymbol{H}^\dagger = (\boldsymbol{H}^\mathsf{T}\boldsymbol{H})^{-1}\boldsymbol{H}^\mathsf{T}$, or when $\boldsymbol{H}\boldsymbol{H}^\mathsf{T}$ is nonsingular and $\boldsymbol{H}^\dagger = \boldsymbol{H}^\mathsf{T}(\boldsymbol{H}\boldsymbol{H}^\mathsf{T})^{-1}$.

2.2 R-ELM

Towards a more stable solution and better generalization performance, Deng et al. [4] proposed the weighted regularized extreme learning machine (WR-ELM) by adding a positive value to the diagonal of $\boldsymbol{H}^\mathsf{T}\boldsymbol{H}$ or $\boldsymbol{H}\boldsymbol{H}^\mathsf{T}$. The following optimization problem was proposed in [4]:

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \quad & \frac{1}{2} \parallel \boldsymbol{\beta} \parallel^2 + \frac{C}{2} \parallel \boldsymbol{D}\boldsymbol{\epsilon} \parallel^2 \\ \text{subject to} \quad & \boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{y} = \boldsymbol{\epsilon}, \end{aligned} \tag{4}$$

where $\boldsymbol{\epsilon}$ is the error vector, C is a regularization parameter, $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{\widetilde{N}})^\mathsf{T}$ is the vector connecting the $\widetilde{N}$ hidden nodes and output $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^\mathsf{T}$, and $\boldsymbol{D}$ is a diagonal weight matrix added to the model to improve outlier robustness. The solution of $\boldsymbol{\beta}$ is given by

$$\boldsymbol{\beta} = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^\mathsf{T}\boldsymbol{D}\boldsymbol{H}\right)^{-1} \boldsymbol{H}^\mathsf{T}\boldsymbol{D}^2\boldsymbol{y}, \tag{5}$$

with $\widetilde{N} \ll N$. When $\boldsymbol{D}$ is equal to the identity matrix, $\boldsymbol{\beta}$ is determined by

$$\boldsymbol{\beta} = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^\mathsf{T}\boldsymbol{H}\right)^{-1} \boldsymbol{H}^\mathsf{T}\boldsymbol{y}, \tag{6}$$

which is called *unweighted* regularized ELM. The performance evaluation presented in [4] uses the sigmoid additive type of SLFNs. Guang-Bin Huang et al. [1] extended the work done in [4] to generalized SLFNs with different types of hidden node functions as well as kernels. Furthermore, Guang-Bin Huang et al. [1] present an alternative solution to the case $\widetilde{N} > N$.

Zhang and Luo [11] proposed an outlier robust ELM (OR-ELM), which instead of using the ℓ_2 norm of the error vector, the ℓ_1 norm loss function is used. The resulting optimization problem is

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \quad & \parallel \boldsymbol{\epsilon} \parallel_1 + \lambda_2 \parallel \boldsymbol{\beta} \parallel^2 \\ \text{subject to} \quad & \boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{y} = \boldsymbol{\epsilon}. \end{aligned} \tag{7}$$

Although WR-ELM and OR-ELM show good results for outlier problems, the output weights are dense, and an appropriate number of neurons needs to be chosen.

Martínez-Martínez et al.[7] considered several penalties for least squares regression to determine the output weights. In particular, lasso [12], ridge regularization [13] and the elastic net [14] were studied. The three regularization methods were described in a generalized way as

$$\underset{(\beta_0, \boldsymbol{\beta}) \in \mathbb{R}^{\widetilde{N}+1}}{\text{minimize}} \; \frac{1}{2N} \sum_{k=1}^{N} (y_k - \beta_0 - \boldsymbol{h}_k^\mathsf{T}\boldsymbol{\beta})^2 + \lambda P_\alpha(\boldsymbol{\beta}), \tag{8}$$

where $\boldsymbol{h}_k^\mathsf{T}$ is the k-th row of matrix $\boldsymbol{H}$, λ is the regularization parameter and

$$P_\alpha(\boldsymbol{\beta}) = \sum_{i=1}^{\widetilde{N}} \left[\frac{1}{2}(1-\alpha)\beta_i^2 + \alpha|\beta_i|\right] \tag{9}$$

is the elastic net penalty. P_α is a trade-off between the ridge regression penalty ($\alpha = 0$) and the lasso penalty ($\alpha = 1$) [7, 15]. The idea of using the elastic net penalty is to identify the degree of relevance of the weights

between hidden nodes and the output. Thus, a compacter network can be achieved by removing the irrelevant or low relevance hidden nodes while preserving the generalization ability of the network [7].

The ridge regularization estimator ($\alpha = 0$) with $\beta_0 = 0$ is

$$\boldsymbol{\beta}_{\text{ridge}} = (\lambda \boldsymbol{I} + \boldsymbol{H}^\top \boldsymbol{H})^{-1} \boldsymbol{H}^\top \boldsymbol{y}, \tag{10}$$

which is the same solution of Equation 6, the unweighted regularized ELM. As pointed by Martínez-Martínez et al.[7], ridge regularization does not have the ability to prune the architecture of network efficiently, though our experiments and literature shows good generalization results for classification problems.

3 Proposed method

In this section, we present our proposed method. We first generalize the regularized extreme learning machine for multiclass classification problems. Then, we present the updates for the ADMM that we use to solve the resulting optimization problem of the proposed method. We also show an alternative for solving our proposed method for large learning tasks using consensus ADMM.

3.1 GR-ELM

Instead of applying R-ELM for each column of $\boldsymbol{T}$, we search for jointly sparse solutions. The main idea of our approach is to find a set of solutions that share a common nonzero support such that a compacter network is obtained.

We propose to use the following minimization problem

$$\underset{\boldsymbol{B}}{\text{minimize}} \; \frac{C}{2} \parallel \boldsymbol{H}\boldsymbol{B} - \boldsymbol{T} \parallel_F^2 + \lambda_1 \parallel \boldsymbol{B} \parallel_{2,1} + \frac{\lambda_2}{2} \parallel \boldsymbol{B} \parallel_F^2, \tag{11}$$

where $\| \cdot \|_F$ is the Frobenius norm, $\parallel \boldsymbol{B} \parallel_{2,1} = \sum_{i=1}^{\tilde{N}} \parallel \boldsymbol{b}_{i,\cdot} \parallel$ and $\boldsymbol{b}_{i,\cdot}$ is the i-th row of $\boldsymbol{B}$. We use alternating direction method of multipliers (ADMM) [9] to solve Equation 11.

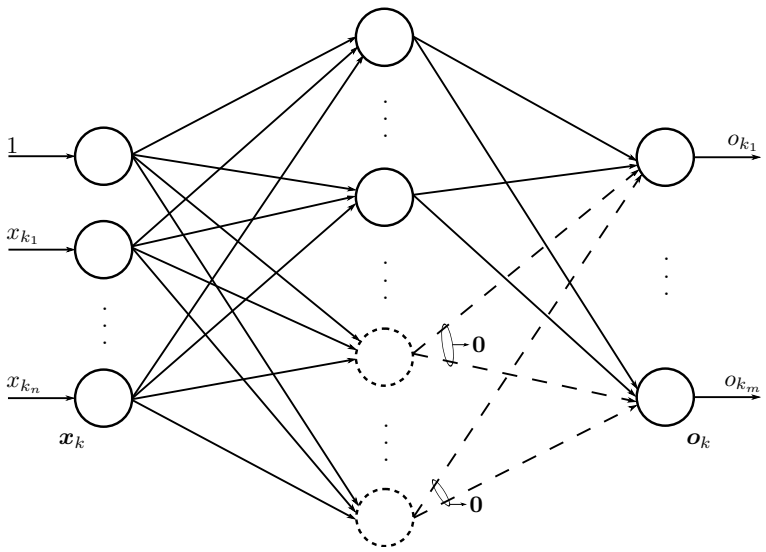

Figure 2: Illustration of an architecture of GR-ELM with n inputs and m outputs.

Since we consider multiclass classification problems, the Frobenius norm is a natural extension of the ℓ_2 norm for dealing with multiple outputs. We use the $\ell_{2,1}$ norm since we search for a compacter network. By compact, we mean that we would like to have a smaller number of neurons in the hidden layer without compromise the training and testing accuracy. If one impose the ℓ_1 norm in each column of $\boldsymbol{B}$, its columns will be sparse, but not necessarily all the elements in a row of $\boldsymbol{B}$ will be zero. When a row of $\boldsymbol{B}$ is zero, we can eliminate the neuron in the hidden node associated with this row. The $\ell_{2,1}$ norm regularization performs the role of eliminating neurons by making all the elements of some rows equals zero, and this is illustrated in Figure 2.

The minimization problem stated in Equation 11 is equivalent to the following constrained problem

$$\begin{aligned} \underset{\boldsymbol{B},\boldsymbol{Z}}{\text{minimize}} \quad & \frac{C}{2} \parallel \boldsymbol{HB} - \boldsymbol{T} \parallel_F^2 + \frac{\lambda_2}{2} \parallel \boldsymbol{B} \parallel_F^2 + \lambda_1 \parallel \boldsymbol{Z} \parallel_{2,1} \\ \text{subject to} \quad & \boldsymbol{B} - \boldsymbol{Z} = \boldsymbol{0}, \end{aligned} \tag{12}$$

which objective function is separable in $\boldsymbol{B}$ and $\boldsymbol{Z}$. In each iteration of ADMM, alternating minimization of augmented Lagrangian over $\boldsymbol{B}$ and $\boldsymbol{Z}$ is performed.

The augmented Lagrangian function of Equation 12 is

$$\begin{aligned} L(\boldsymbol{B}, \boldsymbol{Z}, \boldsymbol{Y}) = & \frac{C}{2} \parallel \boldsymbol{HB} - \boldsymbol{T} \parallel_F^2 + \frac{\lambda_2}{2} \parallel \boldsymbol{B} \parallel_F^2 \\ & + \lambda_1 \parallel \boldsymbol{Z} \parallel_{2,1} + \frac{\rho}{2} \parallel \boldsymbol{B} - \boldsymbol{Z} \parallel_F^2 + \langle \boldsymbol{Y}, \boldsymbol{B} - \boldsymbol{Z} \rangle, \end{aligned} \tag{13}$$

where $\boldsymbol{Y}$ is the Lagrangian multiplier, and $\rho > 0$ is the penalty parameter.

Using the scaled dual variable, we can write the augmented Lagrangian in a slightly different form as

$$\begin{aligned} L(\boldsymbol{B}, \boldsymbol{Z}, \boldsymbol{Y}) = & \frac{C}{2} \parallel \boldsymbol{HB} - \boldsymbol{T} \parallel_F^2 + \frac{\lambda_2}{2} \parallel \boldsymbol{B} \parallel_F^2 \\ & + \lambda_1 \parallel \boldsymbol{Z} \parallel_{2,1} + \frac{\rho}{2} \parallel \boldsymbol{B} - \boldsymbol{Z} + \boldsymbol{U} \parallel_F^2 - \frac{\rho}{2} \parallel \boldsymbol{U} \parallel_F^2, \end{aligned} \tag{14}$$

where $\boldsymbol{U} = (1/\rho)\boldsymbol{Y}$ is the scaled dual variable.

At iteration k, ADMM consists of the following update rules for

1) $\boldsymbol{B}^{k+1}$, we have the following subproblem

$$\boldsymbol{B}^{k+1} := \arg\min_{\boldsymbol{B}} \frac{C}{2} \parallel \boldsymbol{HB} - \boldsymbol{T} \parallel_F^2 + \frac{\lambda_2}{2} \parallel \boldsymbol{B} \parallel_F^2 + \frac{\rho}{2} \parallel \boldsymbol{B} - \boldsymbol{Z}^k + \boldsymbol{U}^k \parallel_F^2 . \tag{15}$$

Making the gradient of the objective function with respect to $\boldsymbol{B}$ equals $\boldsymbol{0}$, we have

$$\boldsymbol{H}^\mathsf{T}(\boldsymbol{HB} - \boldsymbol{T}) + \frac{\lambda_2}{C}\boldsymbol{B} + \frac{\rho}{C}(\boldsymbol{B} - \boldsymbol{Z}^k + \boldsymbol{U}^k) = \boldsymbol{0}, \tag{16}$$

and the minimizer for Equation 15 is

$$\boldsymbol{B} = (\boldsymbol{H}^\mathsf{T}\boldsymbol{H} + \eta\boldsymbol{I})^{-1} \left[\boldsymbol{H}^\mathsf{T}\boldsymbol{T} + \frac{\rho}{C}(\boldsymbol{Z}^k - \boldsymbol{U}^k) \right], \tag{17}$$

where $\eta = (\lambda_2 + \rho)/C$.

2) $\boldsymbol{Z}^{k+1}$, we have the following subproblem

$$\boldsymbol{Z}^{k+1} := \arg\min_{\boldsymbol{Z}} \frac{\lambda_1}{\rho} \parallel \boldsymbol{Z} \parallel_{2,1} + \frac{1}{2} \parallel \boldsymbol{B}^{k+1} - \boldsymbol{Z} + \boldsymbol{U}^k \parallel_F^2 . \tag{18}$$

This optimization problem is equivalent to the following row-wise optimization problems for all rows of $\boldsymbol{Z}$, $\boldsymbol{B}$ and $\boldsymbol{U}$

$$\boldsymbol{z}_{i,\cdot}^{k+1} := \arg\min_{\boldsymbol{z}_{i,\cdot}} \frac{\lambda_1}{\rho} \parallel \boldsymbol{z}_{i,\cdot} \parallel + \frac{1}{2} \parallel \boldsymbol{b}_{i,\cdot}^{k+1} - \boldsymbol{z}_{i,\cdot} + \boldsymbol{u}_{i,\cdot}^k \parallel^2 . \tag{19}$$

Equation 19 has closed form solution [9, 16] given by

$$\boldsymbol{z}_{i,\cdot}^{k+1} = S_{\frac{\lambda_1}{\rho}}(\boldsymbol{b}_{i,\cdot}^{k+1} + \boldsymbol{u}_{i,\cdot}^{k}), \tag{20}$$

where $S_\kappa : \mathbb{R}^m \to \mathbb{R}^m$, with $\kappa \in \mathbb{R}$, is the block soft-thresholding operator [9, 16] defined as

$$S_\kappa(\boldsymbol{a}) = \left(1 - \frac{\kappa}{\| \boldsymbol{a} \|}\right)_+ \boldsymbol{a}, \tag{21}$$

with $S_\kappa(\boldsymbol{0}) = \boldsymbol{0}$ and $(d)_+ \equiv \max(0, d)$. Let $\mathcal{S}_\kappa(\boldsymbol{A})$ be the operator that applies the block soft-thresholding Equation 21 in each row of $\boldsymbol{A}$. We can write the solution for Equation 18 as

$$\boldsymbol{Z}^{k+1} = \mathcal{S}_{\frac{\lambda_1}{\rho}}(\boldsymbol{B}^{k+1} + \boldsymbol{U}^k). \tag{22}$$

3) $\boldsymbol{U}^{k+1}$, the Lagrangian multiplier is updated by

$$\boldsymbol{U}^{k+1} := \boldsymbol{U}^k + \boldsymbol{B}^{k+1} - \boldsymbol{Z}^{k+1}. \tag{23}$$

Algorithm 1 resumes the GR-ELM.

Algorithm 1: Generalized regularized extreme learning machine

Require: Training samples $(\boldsymbol{x}_p, \boldsymbol{t}_p)$, $p = 1, \ldots, N$, regularization parameters λ_1, λ_2, C, ρ and activation function $h(\cdot)$.
Initialize: $\boldsymbol{B}^0$, $\boldsymbol{U}^0$, $\boldsymbol{Z}^0$ and $k = 0$.
Randomly assign parameters $\boldsymbol{A}$ and $\boldsymbol{\nu}$.
Calculate the hidden layer output matrix $\boldsymbol{H}$ using Equation 2.
repeat
  $\boldsymbol{B}^{k+1} := \arg\min_{\boldsymbol{B}} \frac{C}{2} \| \boldsymbol{H}\boldsymbol{B} - \boldsymbol{T} \|_F^2 + \frac{\lambda_2}{2} \| \boldsymbol{B} \|_F^2 + \frac{\rho}{2} \| \boldsymbol{B} - \boldsymbol{Z}^k + \boldsymbol{U}^k \|_F^2$
  $\boldsymbol{Z}^{k+1} := \arg\min_{\boldsymbol{Z}} \frac{\lambda_1}{\rho} \| \boldsymbol{Z} \|_{2,1} + \frac{1}{2} \| \boldsymbol{B}^{k+1} - \boldsymbol{Z} + \boldsymbol{U}^k \|_F^2$
  $\boldsymbol{U}^{k+1} := \boldsymbol{U}^k + \boldsymbol{B}^{k+1} - \boldsymbol{Z}^{k+1}$
  $k = k + 1$
until *meet stopping criterion*

3.2 Distributed GR-ELM

We shall now discuss one approach for solving Equation 11 when the number of training samples is large, with a modest number of neurons. We use global variable consensus ADMM [9] to solve the problem in a distributed way, so each processor handles a subset of training data. Applications where data is stored or collected in a distributed fashion, or there are so many training data that to process them on a single machine is impossible or inconvenient, may be benefited with this approach [9].

We partition the training set in M parts. Let

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 \\ \vdots \\ \boldsymbol{X}_M \end{bmatrix}, \quad \boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_1 \\ \vdots \\ \boldsymbol{T}_M \end{bmatrix} \tag{24}$$

with $\boldsymbol{X}_j \in \mathbb{R}^{N_j \times n}$, and $\boldsymbol{T}_j \in \mathbb{R}^{N_j \times m}$ where $\sum_{j=1}^M N_j = N$. We consider that each processor j has access to the parameters of the hidden nodes so it can generate the matrix $\boldsymbol{H}_j \in \mathbb{R}^{N_j \times \tilde{N}}$ following Equation 2. Thus, $\boldsymbol{H}_j$ and $\boldsymbol{T}_j$ will be handled by the j-th processor.

We can write Equation 12 in the consensus form

$$\begin{aligned} \underset{\boldsymbol{B},\boldsymbol{Z}}{\text{minimize}} \quad & \frac{C}{2} \sum_{j=1}^M \| \boldsymbol{H}_j\boldsymbol{B}_j - \boldsymbol{T}_j \|_F^2 + \frac{\lambda_2}{2} \sum_{j=1}^M \| \boldsymbol{B}_j \|_F^2 + \lambda_1 \| \boldsymbol{Z} \|_{2,1} \\ \text{subject to} \quad & \boldsymbol{B}_j - \boldsymbol{Z} = \boldsymbol{0}, \quad j = 1, \ldots, M, \end{aligned} \tag{25}$$

where the local variables $\boldsymbol{B}_j \in \mathbb{R}^{\widetilde{N}\times m}$ and the global variable $\boldsymbol{Z} \in \mathbb{R}^{\widetilde{N}\times m}$. The resulting ADMM algorithm, in scaled form, is

$$\boldsymbol{B}_j^{k+1} := \arg\min_{\boldsymbol{B}_j} \frac{C}{2} \parallel \boldsymbol{H}_j\boldsymbol{B}_j - \boldsymbol{T}_j \parallel_F^2 + \frac{\lambda}{2} \parallel \boldsymbol{B}_j \parallel_F^2 + \frac{\rho}{2} \parallel \boldsymbol{B}_j - \boldsymbol{Z}^k + \boldsymbol{U}_j^k \parallel_F^2 \tag{26}$$

$$\boldsymbol{Z}^{k+1} := \arg\min_{\boldsymbol{Z}} \frac{\lambda_1}{\rho} \parallel \boldsymbol{Z} \parallel_{2,1} + \frac{1}{2}\sum_{j=1}^{M} \parallel \boldsymbol{B}_j^{k+1} - \boldsymbol{Z} + \boldsymbol{U}_j^k \parallel_F^2 \tag{27}$$

$$\boldsymbol{U}_j^{k+1} := \boldsymbol{U}_j^k + \boldsymbol{B}_j^{k+1} - \boldsymbol{Z}^{k+1}. \tag{28}$$

We can express the $\boldsymbol{Z}$-update (Equation 27) as an averaging step and proximal step involving the sum of norms [9]

$$\boldsymbol{Z}^{k+1} := \arg\min_{\boldsymbol{Z}} \left(\frac{\lambda_1}{M\rho} \parallel \boldsymbol{Z} \parallel_{2,1} + \frac{1}{2} \parallel \overline{\boldsymbol{B}}^{k+1} - \boldsymbol{Z} + \overline{\boldsymbol{U}}^k \parallel_F^2, \right) \tag{29}$$

where $\overline{\boldsymbol{B}}^{k+1} = 1/M \sum_{j=1}^{M} \boldsymbol{B}_j^{k+1}$ and $\overline{\boldsymbol{U}}^{k+1} = 1/M \sum_{j=1}^{M} \boldsymbol{U}_j^{k+1}$. The solutions for Equation 26 and Equation 29 is

$$\boldsymbol{B}_j = \left(\boldsymbol{H}_j^\intercal \boldsymbol{H}_j + \frac{\lambda_2 + \rho}{C} \boldsymbol{I} \right)^{-1} \left[\boldsymbol{H}_j^\intercal \boldsymbol{T}_j + \frac{\rho}{C} (\boldsymbol{Z}^k - \boldsymbol{U}_j^k) \right] \tag{30}$$

and

$$\boldsymbol{Z}^{k+1} = \mathcal{S}_{\frac{\lambda_1}{M\rho}} (\overline{\boldsymbol{B}}^{k+1} + \overline{\boldsymbol{U}}^k), \tag{31}$$

respectively, where $\mathcal{S}_\kappa$ is the block soft-thresholding operator for matrices defined as before. We implement the Distributed GR-ELM (DGR-ELM) in Message Passing Interface (MPI) using a single program, multiple data (SPMD). Algorithm 2 resumes the DGR-ELM.

Algorithm 2: Distributed generalized regularized extreme learning machine

1 **Require**: M processors, with each processor j storing training samples $(\boldsymbol{X}_j, \boldsymbol{T}_j) = \{\boldsymbol{x}_{p_j}, \boldsymbol{t}_{p_j}\}_{p_j=1}^{N_j}$, regularization parameters λ_1, λ_2, C, ρ and activation function $h(\cdot)$, parameter $\boldsymbol{A}$ and $\boldsymbol{\nu}$.
2 **Initialize**: M processes, along with $\boldsymbol{B}_j$, $\boldsymbol{U}_j$ and $\boldsymbol{Z}$.
3 Calculate the j-th hidden layer output matrix $\boldsymbol{H}_j(\boldsymbol{X}_j, \boldsymbol{A}, \boldsymbol{\nu})$ using Equation 2.
4 **repeat**
5 $\quad \boldsymbol{U}_j := \boldsymbol{U}_j + \boldsymbol{B}_j - \boldsymbol{Z}$
6 $\quad \boldsymbol{B}_j := \arg\min_{\boldsymbol{B}} \frac{C}{2} \parallel \boldsymbol{H}_j\boldsymbol{B}_j - \boldsymbol{T}_j \parallel_F^2 + \frac{\lambda_2}{2} \parallel \boldsymbol{B}_j \parallel_F^2 + \frac{\rho}{2} \parallel \boldsymbol{B}_j - \boldsymbol{Z} + \boldsymbol{U}_j \parallel_F^2$
7 $\quad$ Let $\boldsymbol{W} := \boldsymbol{B}_j + \boldsymbol{U}_j$
8 $\quad$ *Allreduce* $\boldsymbol{W}$
9 $\quad \boldsymbol{Z} := \arg\min_{\boldsymbol{Z}} \frac{\lambda_1}{M\rho} \parallel \boldsymbol{Z} \parallel_{2,1} + \frac{1}{2} \parallel \boldsymbol{W} - \boldsymbol{Z} \parallel_F^2$
10 **until** *meet stopping criterion*

In step 1 we assume that each processor has access to the regularization parameters, activation function and parameters $\boldsymbol{A}$ and $\boldsymbol{\nu}$ but the parameters and activation function could be generated/stored in only one processor and broadcasted to other processors.

In step 8, *Allreduce* denotes the operation `MPI_Allreduce` to compute the global sum over all processors and store the result in $\boldsymbol{W} = \sum_{j=1}^{M}(\boldsymbol{B}_j + \boldsymbol{U}_j) = M(\overline{\boldsymbol{B}} + \overline{\boldsymbol{U}})$ on every processor. In step 9, all processors (redundantly) compute the $\boldsymbol{Z}$-update. Although a single processor could compute the Z-update and brodcast the result to other processors, this approach complicates the code and is generally no faster [9].

3.3 Computational notes

The update of $\boldsymbol{B}$ stated in Equation 17 involves the inverse of $\boldsymbol{G} = (\boldsymbol{H}^\mathsf{T}\boldsymbol{H} + \eta\boldsymbol{I})$ that we solve by Cholesky factorization followed by a forward and backward substitution. Since we maintain ρ fix, we can factor $\boldsymbol{G}$ once, and use the cached factorization in subsequent solve steps.

When $\widetilde{N} > N$ we can use the matrix inversion lemma to reduce computation

$$\begin{aligned}(\eta\boldsymbol{I} + \boldsymbol{H}^\mathsf{T}\boldsymbol{H})^{-1} &= (\eta\boldsymbol{I})^{-1} - (\eta\boldsymbol{I})^{-1}\boldsymbol{H}^\mathsf{T}(\boldsymbol{I}^{-1} + \boldsymbol{H}(\eta\boldsymbol{I})^{-1}\boldsymbol{H}^\mathsf{T})^{-1}\boldsymbol{H}(\eta\boldsymbol{I})^{-1} \\ &= \frac{1}{\eta}(\boldsymbol{I} - \boldsymbol{H}^\mathsf{T}(\eta\boldsymbol{I} + \boldsymbol{H}\boldsymbol{H}^\mathsf{T})^{-1}\boldsymbol{H}). \end{aligned} \tag{32}$$

Then, factorization of $(\eta\boldsymbol{I} + \boldsymbol{H}\boldsymbol{H}^\mathsf{T})$ can be cached and forward and backward substitution can be used in the updates. This procedure also applies to Equation 30.

3.4 Stopping criterion

We follow the suggestions given in [9] for stopping criterion. We terminate the algorithm when primal and dual residuals, given by

$$\boldsymbol{R}^k = \boldsymbol{B}^k - \boldsymbol{Z}^k \text{ and } \boldsymbol{S} = \rho(\boldsymbol{Z}^k - \boldsymbol{Z}^{k-1}), \tag{33}$$

respectively, satisfy

$$\| \boldsymbol{R}^k \|_F \leq \epsilon^{\text{pri}} \text{ and } \quad \| \boldsymbol{S}^k \|_F \leq \epsilon^{\text{dual}}. \tag{34}$$

The tolerances $\epsilon^{\text{pri}} > 0$ and $\epsilon^{\text{dual}} > 0$ are set using an absolute and relative criterion,

$$\epsilon^{\text{pri}} = \sqrt{m\widetilde{N}}\,\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max\{\| \boldsymbol{B}^k \|_F, \| \boldsymbol{Z}^k \|_F\} \tag{35}$$

$$\epsilon^{\text{dual}} = \sqrt{m\widetilde{N}}\,\epsilon^{\text{abs}} + \epsilon^{\text{rel}}\rho \| \boldsymbol{U}^k \|_F, \tag{36}$$

where $\epsilon^{\text{abs}} > 0$ and $\epsilon^{\text{rel}} > 0$ are the absolute and relative tolerance, respectively.

3.5 Discussion

Our approach is a generalization of the regularized extreme learning machine. If $\boldsymbol{T} = \boldsymbol{y} \in \mathbb{R}^m$, then Equation 11 is the elastic net regularization problem (Equation 8). Also, when $\lambda_1 = 0$ and $C = 1$ we have ridge regularization for multi outputs with common ridge parameter, λ_2, and the solution is an extension of Equation 10 given by

$$\boldsymbol{B} = (\boldsymbol{H}^\mathsf{T}\boldsymbol{H} - \lambda_2\boldsymbol{I})^{-1}\boldsymbol{H}^\mathsf{T}\boldsymbol{T}. \tag{37}$$

In Equation 11 we would only need to use the regularization parameters λ_1 and λ_2 and we would still have an equivalent minimization problem. However, when applying ADMM for solving Equation 11 the update of $\boldsymbol{B}^{k+1}$ (Equation 15) depends on $\eta = (\lambda_2 + \rho)/C$, which would be limited if $C = 1$, since we usually fix $\rho = 1$. If $C = 1$ then η would always be greater than one. The only way having $\eta < 1$ when fixing $C = 1$ is changing both λ_2 and ρ which we avoid since ρ is also used in the update of $\boldsymbol{Z}^{k+1}$ (Equation 22).

Although outlier robustness is out of the scope of this paper, with a small change in the objective function, GR-ELM could be robust to outliers. In Equation 11, if we change the Frobenius norm with the matrix 1-norm of the error, then for the particular case of single output and appropriate choice of regularization parameters we would have the same optimization problem as the OR-ELM.

4 Experiments

In this section, we evaluate the proposed GR-ELM. We selected 14 binary classification datasets (Table 1) and eight multiclass classification datasets (Table 2). These datasets are taken from UCI Repository and LIBSVM portal [17, 18].

For binary classification datasets, labels are either -1 or 1. For a multiclass dataset with m classes, the k-th class is represented by a m-dimensional vector with all elements equals to -1 except for the k-th element of the vector, which is equal 1. Attributes of training data are normalized to have values in $[-1, 1]$. The testing data are normalized accordingly to the factors used in the normalization of training data.

All the experiments were conducted in a 3.6 GHz core i7 with 8 GB of RAM. Simulations for GR-ELM are carried in MATLAB 8.5 and simulations for DGR-ELM are done in C using MPI and GNU Scientific Library (GSL) for linear algebra with ATLAS library.

4.1 Parameters specifications

The initial number of neurons ($\widetilde{N}$) in the hidden layer is set to 1000 for all datasets. For ADMM, we fixed $\rho = 1$, and regularization parameters λ_1 and C are selected by 5-fold cross-validation. We tested 14 values for C: $[0.01, 0.1, 0.2, 0.5, 1, 2, 5, 20, 50, 100, 200, 500, 1000]$. For λ_1, 100 values were tested decreasing from 100 to $\lambda_{\min}$ in a log scale. The value of $\lambda_{\min}$ is defined as 0.0001 if $N > n$ and 0.01 otherwise. The parameter λ_2 is fixed to 0.1. We use the same parameters for both GR-ELM and DGR-ELM. The values of $\boldsymbol{A}$ and $\boldsymbol{\nu}$ are randomly generated based on uniform distribution and the sigmoid function$h(\boldsymbol{x}_k; \boldsymbol{a}_i, \nu_i) =$

Table 1: Information about binary classifcation datasets

Datasets	# Training data	# Testing Data	# Attributes
Bupa	173	172	6
Australia	346	344	14
Breast Cancer	342	341	10
Diabetes	384	384	8
Heart	135	135	13
Ionosphere	176	175	34
Mushroom	4062	4062	22
SVMGuide1	3089	3089	4
Magic	9510	9510	11
COD RNA	29768	29767	8
Colon Cancer	31	31	2000
Leukemia	38	34	7129
Spambase	2301	2300	57
Adult	6414	26147	123

Table 2: Information about multiclass classifcation datasets

Datasets	# Training data	# Testing Data	# Features	# Classes
Iris	75	75	4	3
Wine	90	88	13	3
Vowel	528	462	10	11
Segment	1155	1155	19	7
Satimage	4435	2000	36	6
DNA	2000	1186	180	3
SVMGuide2	197	194	20	3
USPS	7291	2007	256	10

$1/\ (1+\ \exp(-\boldsymbol{a}_i \cdot \boldsymbol{x}\ -\ \nu_i))$ is chosen as activation function for GR-ELM and DGR-ELM. For GR-ELM, the absolute and relative tolerance is $\epsilon^{\text{abs}} = 10^{-3}$ and $\epsilon^{\text{rel}} = 10^{-2}$, respectively. For convenience and code simplification we fixed the number of iteration of DGR-ELM to 100.

4.2 Performance of GR-ELM

We compare the proposed GR-ELM with ELM with ridge regularization. Table 3 and Table 4 resumes the training & testing accuracy (and corresponding standard deviation) and time for binary and multiclass classification datasets, respectively, of 20 trials for each dataset. In each trial, random permutation within training data set and testing data set is performed. Table 5 summarize the regularization parameters obtained through 5-fold cross-validation. Table 6 show the mean number of neurons of 20 trials for binary and multiclass classification datasets.

Table 3: Mean accuracy with corresponding standard deviation and running time for training and testing in binary class datasets ($\lambda_2 = 0.1$ and $\widetilde{N} = 1000$)

Datasets	GR-ELM				ELM			
	Train. Acc.	Test. Acc.	Train. Time	Test Time	Train. Acc.	Test. Acc.	Train. Time	Test Time
Bupa	80.46 ± 0.81	**71.77 ± 0.89**	0.0107	0.0022	81.07 ± 0.67	71.42 ± 0.95	0.0008	0.0025
Australian	88.73 ± 0.48	**84.59 ± 0.46**	0.0137	0.0006	88.53 ± 0.35	84.52 ± 0.39	0.0025	0.0053
Breast Cancer	97.50 ± 0.20	96.60 ± 0.26	0.0170	0.0005	97.87 ± 0.19	**96.74 ± 0.09**	0.0023	0.0050
Diabetes	79.02 ± 0.43	76.85 ± 0.55	0.0490	0.0005	79.93 ± 0.35	**76.86 ± 0.37**	0.0033	0.0058
Heart	87.63 ± 0.35	**87.26 ± 0.70**	0.0078	0.0008	87.41 ± 0.00	87.15 ± 0.69	0.0007	0.0022
Ionosphere	96.14 ± 0.57	**87.29 ± 0.67**	0.0096	0.0008	97.05 ± 0.40	87.14 ± 0.57	0.0010	0.0028
Mushroom	100.00 ± 0.00	**100.00 ± 0.00**	0.1049	0.0835	100.00 ± 0.00	**100.00 ± 0.00**	0.0403	0.0822
SVMGuide	97.14 ± 0.05	**96.74 ± 0.04**	0.1556	0.0854	97.09 ± 0.05	96.67 ± 0.05	0.0392	0.0817
Magic	88.00 ± 0.11	**86.63 ± 0.14**	0.1632	0.1424	87.87 ± 0.09	**86.63 ± 0.10**	0.0993	0.1841
COD RNA	95.68 ± 0.04	**95.23 ± 0.05**	0.4347	0.6348	95.50 ± 0.05	95.20 ± 0.06	0.3435	0.6553
Colon Cancer	100.00 ± 0.00	**82.90 ± 4.07**	0.0052	0.0015	100.00 ± 0.00	82.42 ± 4.25	0.0001	0.0022
Leukemia	100.00 ± 0.00	**76.91 ± 4.80**	0.0050	0.0046	100.00 ± 0.00	76.76 ± 4.47	0.0001	0.0059
Spambase	95.49 ± 0.17	**93.09 ± 0.20**	0.1038	0.0452	95.08 ± 0.12	92.99 ± 0.21	0.0389	0.0529
Adult	85.20 ± 0.18	84.18 ± 0.08	0.1302	0.3905	84.98 ± 0.15	**84.21 ± 0.06**	0.0622	0.5340

Table 4: Mean accuracy with corresponding standard deviation and running time for training and testing in multiclass datset ($\lambda_2 = 0.1$ and $\widetilde{N} = 1000$)

Datasets	GR-ELM				ELM			
	Train. Acc.	Test. Acc.	Train. Time	Test. Time	Train. Acc.	Test. Acc.	Train. Time	Test. Time
Iris	100.0 ± 0.00	**94.53 ± 0.96**	0.0117	0.0004	98.67 ± 0.00	93.07 ± 1.02	0.0003	0.0014
Wine	100.0 ± 0.00	**95.57 ± 0.82**	0.0106	0.0015	100.0 ± 0.00	**95.57 ± 0.82**	0.0005	0.0015
Vowel	100.0 ± 0.00	50.02 ± 1.93	0.2739	0.0072	100.0 ± 0.00	**50.18 ± 1.97**	0.0061	0.0073
Segment	97.82 ± 0.14	**95.75 ± 0.26**	0.4296	0.0200	97.98 ± 0.14	95.74 ± 0.26	0.0252	0.0210
Satimage	93.52 ± 0.14	90.26 ± 0.32	0.3359	0.0459	93.86 ± 0.15	**90.35 ± 0.29**	0.0439	0.0426
DNA	97.99 ± 0.18	**93.38 ± 0.55**	0.2079	0.0267	98.03 ± 0.17	93.36 ± 0.56	0.0288	0.0267
SVMGuide2	92.28 ± 0.80	**81.96 ± 0.73**	0.0162	0.0008	93.68 ± 0.31	81.34 ± 0.88	0.0010	0.0031
USPS	97.87 ± 0.12	92.21 ± 0.25	0.2646	0.0535	97.95 ± 0.11	**92.26 ± 0.22**	0.0713	0.0522

Our simulations indicate that GR-ELM has similar testing accuracy when compared to ELM with ridge regularization, although usually faster testing and compacter network is achieved. Training time for GR-ELM is higher when compared to the training time of ELM, which is expected since solving Equation 11 is computationally more demanding than solving ridge regularized ELM. However, there is no need to know the optimal value of the number of neurons when working with GR-ELM. If the initial number of neurons is larger enough, ridge regularized ELM and GR-ELM should perform similarly in testing accuracy perspective. However, faster testing and compacter network is expected for GR-ELM.

Table 5: Parameters specifications of binary dataset and mean number of neurons ($\lambda_2 = 0.1$ and $\widetilde{N} = 1000$)

Datasets	GR-ELM			ELM	
	λ_1	C	Numb. Neur.	C	Numb. Neur.
Bupa	0.0201	5	803.80	50	1000
Australian	1.3219	10	66.95	5	1000
Breast Cancer	0.5722	5	48.50	10	1000
Diabetes	6.1359	50	23.95	10	1000
Heart	0.0534	0.5	256.90	1	1000
Ionosphere	0.1630	2	217.40	5	1000
Mushroom	0.1630	500	950.05	1000	1000
SVMGuide	0.0001	1000	1000.0	1000	1000
Magic	0.5722	50	778.55	200	1000
COD RNA	1.1498	1000	963.45	1000	1000
Colon Cancer	0.0145	2	642.45	10	1000
Leukemia	0.0100	1000	742.20	1000	1000
Spambase	0.0351	2	880.20	10	1000
Adult	0.0201	0.1	732.40	0.5	1000

Table 6: Parameters specifications of multiclass dataset and mean number of neurons ($\lambda_2 = 0.1$ and $\widetilde{N} = 1000$)

Datasets	GR-ELM			ELM	
	λ_1	C	Numb. Neur.	C	Numb. Neur.
Iris	2.3101	1000	186.90	1000	1000
Wine	0.0001	50	1000.0	500	1000
Vowel	0.0001	1000	1000.0	1000	1000
Segment	0.1630	100	999.65	1000	1000
Satimage	0.1630	10	999.55	100	1000
DNA	0.0038	0.2	999.90	2	1000
SVMGuide2	0.5722	5	193.00	10	1000
USPS	0.2154	1	1000.0	5	1000

4.3 Performance of DGR-ELM

We now evaluate the distributed version of GR-ELM. Although DGR-ELM has an appealing application on large datasets, we evaluated DGR-ELM using all datasets to see its behavior using small datasets as well. Table 7 resume the mean accuracy of 20 runs for binary classification problems using DGR-ELM for 1, 2 and 4 processes. The same experiment is done for multiclass classification datasets and the results are summarized in Table 8.

As expected, the training and testing accuracy for both, binary and multiclass classification problems, are similar to training and testing accuracy obtained for GR-ELM. In other words, training data can be distributed in different machines and DGR-ELM should achieve similar training and testing accuracy as if we would have all training set concentrated in one machine.

Table 9 summarize the mean training and testing time for 20 runs using DGR-ELM with 1, 2 and 4 processes for binary classification problems. The same results for multiclass classification problems are shown in Table 10. We point out that faster training time is not the main concern of this paper. Faster training time can be achieved by using LAPACK instead of GSL for Cholesky factorization, and by using a GPU-based linear algebra package as suggested by Boyd [9].

Table 7: Mean accuracy with corresponding standard deviation for training and testing in binary classification problems using DGR-ELM for 1, 2 and 4 processes ($\lambda_2 = 0.1$ and $\widetilde{N} = 1000$)

Datasets	1		2		4	
	Train. Acc.	Test. Acc.	Train. Acc.	Test Acc.	Train. Acc.	Test. Acc.
Bupa	79.95 ± 0.55	72.66 ± 0.81	78.92 ± 0.49	72.34 ± 0.29	77.46 ± 0.00	72.09 ± 0.00
Australian	89.25 ± 0.42	84.80 ± 0.54	88.73 ± 0.23	84.40 ± 0.50	88.17 ± 0.37	84.82 ± 0.12
Breast Cancer	97.61 ± 0.21	96.91 ± 0.23	97.46 ± 0.14	96.60 ± 0.17	97.47 ± 0.14	96.63 ± 0.15
Diabetes	78.98 ± 0.55	76.86 ± 0.44	79.06 ± 0.19	77.04 ± 0.33	78.48 ± 0.13	77.17 ± 0.70
Heart	88.01 ± 0.28	87.59 ± 0.55	87.39 ± 0.10	87.42 ± 0.10	87.41 ± 0.00	87.21 ± 0.32
Ionosphere	96.41 ± 0.42	87.45 ± 0.86	94.73 ± 0.26	87.02 ± 0.26	93.20 ± 0.80	86.49 ± 0.53
Mushroom	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00
SVMGuide	97.13 ± 0.04	96.73 ± 0.04	97.10 ± 0.05	96.76 ± 0.05	97.09 ± 0.05	96.77 ± 0.04
Magic	87.95 ± 0.11	86.63 ± 0.10	87.77 ± 0.09	86.62 ± 0.10	87.52 ± 0.09	86.51 ± 0.10
COD RNA	95.37 ± 0.06	95.13 ± 0.05	95.30 ± 0.05	95.10 ± 0.05	95.24 ± 0.05	95.07 ± 0.05
Colon Cancer	100.0 ± 0.00	80.84 ± 2.18	100.0 ± 0.00	79.42 ± 3.15	100.0 ± 0.00	80.77 ± 2.66
Leukemia	100.0 ± 0.00	77.88 ± 4.09	100.0 ± 0.00	78.53 ± 3.24	100.0 ± 0.00	78.18 ± 4.60
Spambase	95.45 ± 0.18	93.06 ± 0.21	94.90 ± 0.20	93.05 ± 0.19	94.18 ± 0.13	92.96 ± 0.21
Adult	85.21 ± 0.15	84.17 ± 0.09	84.73 ± 0.11	84.16 ± 0.07	84.27 ± 0.14	84.09 ± 0.07

Table 8: Mean accuracy and corresponding standard deviation in multiclass classification problems using DGR-ELM for 1, 2 and 4 processes ($\lambda_2 = 0.1$ and $\widetilde{N} = 1000$)

Datasets	1		2		4	
	Train. Acc.	Test. Acc.	Train. Acc.	Test Acc.	Train. Acc.	Test. Acc.
Iris	100.0 ± 0.00	94.64 ± 0.19	100.0 ± 0.00	93.33 ± 0.00	98.67 ± 0.00	93.33 ± 0.00
Wine	100.0 ± 0.00	95.45 ± 0.00	100.0 ± 0.00	96.57 ± 0.16	100.0 ± 0.00	96.59 ± 0.00
Vowel	100.0 ± 0.00	50.42 ± 1.78	100.0 ± 0.00	48.76 ± 1.96	100.0 ± 0.00	49.71 ± 1.60
Segment	97.89 ± 0.13	95.77 ± 0.24	97.31 ± 0.16	95.54 ± 0.22	96.74 ± 0.13	95.00 ± 0.23
Satimage	93.53 ± 0.16	90.29 ± 0.25	92.75 ± 0.17	89.91 ± 0.25	91.89 ± 0.15	89.46 ± 0.25
DNA	97.90 ± 0.23	93.21 ± 0.40	97.33 ± 0.22	93.25 ± 0.38	96.34 ± 0.30	92.67 ± 0.63
SVMGuide2	91.30 ± 0.74	81.44 ± 0.34	91.21 ± 0.24	81.94 ± 0.42	89.73 ± 1.10	82.11 ± 0.76
USPS	97.86 ± 0.13	92.20 ± 0.37	97.56 ± 0.10	92.11 ± 0.30	97.15 ± 0.11	91.86 ± 0.35

5 Conclusions

In this paper, we proposed a generalization of R-ELM, namely GR-ELM, and we derive an alternative algorithm when training data is distributed (DGR-ELM). By using $\ell_{2,1}$ norm and Frobenius norm, we were able to extend the R-ELM for multiclass classification problems. R-ELM is a particular case of our proposed method when considering single output in the network. Our experiments showed that by using GR-ELM instead of ELM with ridge regularization, usually a compacter network is achieved without compromising the testing accuracy. As a consequence, testing time was usually faster when using GR-ELM since ELM with ridge regularization results in dense networks. For DGR-ELM, our results indicated that similar training and testing accuracy can be obtained when training data is distributed.

For future investigations we will focus on the following observations: (a) although we only tested the GR-ELM and DGR-ELM for classification problems, there is no restriction for applying it to other problems such as regression; (b) in our simulations, sigmoid additive type of SLFNs were used and extensions to other types of activation functions as well as kernel can be done; (c) although we have not tested in this paper, simple modifications can make GR-ELM robust to outlier; and (d) improvements in training time for DGR-

ELM can be achieved by using GPU-based linear algebra package and by using LAPACK instead of GSL for Cholesky factorization.

Table 9: Mean time for binary classification problems using DGR-ELM for 1, 2 and 4 processes ($\lambda_2 = 0.1$ and $\widetilde{N} = 1000$)

Datasets	1		2		4	
	Train. Time	Test. Time	Train. Time	Test. Time	Train. Time	Test. Time
Bupa	0.0408	0.0055	0.0159	0.0062	0.0169	0.007
Australian	0.0875	0.0009	0.0414	0.0011	0.025	0.002
Breast Cancer	0.0855	0.0006	0.046	0.0011	0.0244	0.0017
Diabetes	0.1015	0.0004	0.0489	0.0005	0.0361	0.0006
Heart	0.0334	0.0013	0.0135	0.0021	0.015	0.0032
Ionosphere	0.0416	0.0016	0.0175	0.0025	0.0175	0.0036
Mushroom	0.486	0.1978	0.4132	0.2033	0.5163	0.2141
SVMGuide	0.4165	0.1926	0.3801	0.1961	0.6586	0.2004
Magic	0.8767	0.3986	0.6108	0.4148	0.657	0.4538
COD RNA	2.3304	1.1616	1.4491	1.3217	1.1825	1.4507
Colon Cancer	0.008	0.008	0.008	0.0104	0.01	0.0118
Leukemia	0.0087	0.0321	0.008	0.0386	0.0096	0.0426
Spambase	0.3597	0.1066	0.3446	0.1127	0.4251	0.1188
Adult	0.6548	1.3012	0.4992	1.4524	0.5684	1.6618

Table 10: Mean time for multiclass classification problems using DGR-ELM for 1, 2 and 4 processes ($\lambda_2 = 0.1$ and $\widetilde{N} = 1000$)

Datasets	1		2		4	
	Train. Time	Test. Time	Train. Time	Test. Time	Train. Time	Test. Time
Iris	0.0324	0.0006	0.024	0.0009	0.0216	0.0015
Wine	0.0401	0.0038	0.0303	0.0041	0.0234	0.0045
Vowel	0.4905	0.0202	0.2442	0.021	0.1877	0.0218
Segment	0.5817	0.0577	0.5156	0.0578	0.3324	0.06
Satimage	0.766	0.1029	0.8923	0.1051	1.4223	0.1091
DNA	0.4373	0.0794	0.5015	0.0818	0.4767	0.0862
SVMGuide2	0.0717	0.0016	0.0462	0.0022	0.0461	0.0035
USPS	1.1786	0.1585	1.387	0.1657	2.2268	0.1774

References

[1] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, Rui Zhang, Extreme Learning Machine for Regression and Multiclass Classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (2) (2012) 513–529.

[2] G.-B. Huang, Q.-y. Zhu, C.-k. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (1-3) (2006) 489–501.

[3] G. Huang, G.-B. Huang, S. Song, K. You, Trends in extreme learning machines: A review, Neural Networks 61 (2015) 32–48.

[4] W. Deng, Q. Zheng, L. Chen, Regularized Extreme Learning Machine, in: 2009 IEEE Symposium on Computational Intelligence and Data Mining, no. 60825202, IEEE, (2009) 389–395.

[5] Zuo Bai, Guang-Bin Huang, Danwei Wang, Han Wang, M. B. Westover, Sparse Extreme Learning Machine for Classification, IEEE Transactions on Cybernetics 44 (10) (2014) 1858–1870.

[6] Y. Wang, Y. Dou, X. Liu, Y. Lei, PR-ELM: Parallel regularized extreme learning machine based on cluster, Neurocomputing 173 (2016) 1073–1081.

[7] J. M. Martínez-Martínez, P. Escandell-Montero, E. Soria-Olivas, J. D. Martín-Guerrero, R. Magdalena-Benedito, J. Gómez-Sanchis, Regularized extreme learning machine for regression problems, Neurocomputing 74 (17) (2011) 3716–3721.

[8] J. Chorowski, J. Wang, J. M. Zurada, Review and performance comparison of SVM- and ELM-based classifiers, Neurocomputing 128 (2014) 507–516.

[9] S. Boyd, Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, Foundations and Trends® in Machine Learning 3 (1) (2010) 1–122.

[10] C. R. Rao, S. K. Mitra, Generalized Inverse of Matrices and Its Applications, Probability and Statistics Series, Wiley, 1971.

[11] K. Zhang, M. Luo, Outlier-robust extreme learning machine for regression problems, Neurocomputing 151 (2015) 1519–1527.

[12] R. Tibshirani, Regression Selection and Shrinkage via the Lasso, Journal of the Royal Statistical Society B (Methodological) 58 (1) (1996) 267–288.

[13] A. E. Hoerl, R. W. Kennard, Ridge Regression: Biased Estimation for Nonorthogonal Problems, Technometrics 12 (1) (1970) 55–67.

[14] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67 (2) (2005) 301–320.

[15] J. Friedman, T. Hastie, R. Tibshirani, Regularization Paths for Generalized Linear Models via Coordinate Descent, Journal of Statistical Software 33 (1) (2010) 1–22.

[16] D. L. Donoho, I. Johnstone, A. Montanari, Accurate Prediction of Phase Transitions in Compressed Sensing via a Connection to Minimax Denoising, IEEE Transactions on Information Theory 59 (6) (2013) 3396–3433.

[17] D. J. N. A. Asuncion, UCI Machine Learning Repository (2007)

[18] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (3) (2011) 27:1–27.

Fernando Kentaro Inaba is a doctoral student in the Department of Electrical Engineering, Federal University of Espírito Santo in Vitória, Brazil. His research interests include neural networks, pattern recognition, sparse representation and image processing. He has a Master's degree in electrical engineering from the Federal University of Espírito Santo (2012).

Evandro Ottoni Teatini Salles received the B.Sc. and Master's degrees in electrical engineering from the Federal University of Espírito Santo, Vitória, Brazil, in 1987 and 1994, respectively, and the Doctorate degree in electrical engineering from the State University of Campinas, Campinas, Brazil, in 2001. He is currently an Associate Professor in the Department of Electrical Engineering, Federal University of Espírito Santo. His current research interests include neural networks, digital signal and image processing, pattern recognition, and video processing.

Sylvain Perron received the B.A.A. in Quantitative Management Technique and Master's degree in Operations Research from HEC Montréal, Québec, Canada, in 1995 and 1998, respectively, and Ph.D. in Operations Research from the École Polytechnique de Montréal, Québec, Canada, in 2004. He is currently an Associate Professor in the Department of Decision Science, HEC Montréal. He is also member of Group for Research in Decision Analysis (GERAD). His current research interests include mathematical programming, global and combinatorial optimization, clustering, column generation and data mining.

Gilles Caporossi received the B.A.A. in Quantitative Management Techniques and his Master's degree in Operations Research from HEC Montréal, Québec, Canada, in 1993 and 1995, respectively, and Ph.D. in Applied Mathematics from Polytechnique Montréal, Québec, Canada, in 2001. He is currently a professor at HEC Montréal in the Department of Decisions Sciences and a member of GERAD, Group for Research in Decision Analysis. His current research interests include Data Mining, Complex Networks and Computer aided Graph Theory.