Distributed Reduced Kernel Extreme Learning Machine

Manrui Zhou
College of Science
Guilin University of Technology
Guilin, China
marryazhou@163.com

Wu Ai*
Center for Data Analysis and Algorithm Technology
Guilin University of Technology
Guilin, China
aiwu818@gmail.com

2021 China Automation Congress (CAC) | 978-1-6654-2647-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/CAC53003.2021.9728238

***Abstract*—Extreme learning machines (ELM) are popular in the field of pattern recognition and machine learning. The kernel extension of ELM (KELM) presents a better performance than traditional ELM. Although the KELM is able to solve complex nonlinear problems, it is time-consuming and memory-demanding when dealing with a large-size kernel matrix. Introducing reduced kernel technique can dramatically cut down the computational load and memory usage. However, as the amount of training data grows exponentially, it is not effective for a single worker to store the kernel matrix, making it infeasible for data mining in a centralized manner. In this paper, a distributed reduced kernel method for training ELM over decentralized data (DRKELM) is proposed. In the DRKELM, we randomly assign data to different nodes. The communication between nodes is fixed and does not depend on the size of the training data on each node, but on the network topology. Different from the existing reduced kernel ELM, the DRKELM is a fully distributed training algorithm based on the method of alternating direction method of multiplier (ADMM). Experiment with the large scale data set finds that the distributed method can achieve almost the same results as the centralized algorithm and even takes less time to a large extent. It greatly reduces the computation time consumption.**

***Index Terms*—Distributed learning, reduced kernel extreme learning machine (RKELM), alternating direction method of multipliers (ADMM).**

I. Introduction

In past research, extreme learning machine (ELM) is a new machine learning system under the base of the Feedforward Neural Network (FNN) [1], [2], which is suitable for supervised and unsupervised learning problems. The most obvious feature of this method is that the weights of hidden layer nodes are given randomly without updating, and only the output weights are calculated in the learning process. In the ELM, we can randomly select weights and the biases hidden layer nodes for regression and classification purposes rather than adjusting them through extensive testing. Therefore, it takes less time than other related algorithms [3], [4]. Besides, in the final output layer, pseudo-inverse and regularization methods are used to solve the output weights.

* Corresponding author. This work was supported by the National Natural Science Foundation of China (Nos. 62166013, 62166015 and 61703117), the Natural Science Foundation of Guangxi, China (No. 2017GXNSF-BA198113) and the Foundation of Guilin University of Technology (No. GLUTQD2007029).

Nevertheless, the performance of ELM is affected by many factors, among which the number of hidden layers and the choice of the activation functions are the main factors. Hence, the accuracy of each experiment of ELM varies with the different sizes of hidden layers selected. In this case, a kernel extreme learning machine (KELM) has been proposed in [5], [6], where the feature mapping in the hidden layer depends on the kernel matrix. In a recent paper [7], this research showed that KELM will randomly select a part of data from the datasets as the support vectors. This is different from traditional support vector machines, which can significantly save computational overhead and considerable accuracy by avoiding iterative learning. However, the efficiency of KELM is low when the data set is large because the kernel matrix is increasing quickly as the data sets get larger. Later, someone proposed an algorithm to simplify the kernel matrix [8]. Similarly, when the training set is very large, the kernel matrix also becomes very large. In this case, relying on a single workplace for storage is not an effective approach.

Under this case, we proposed a new distributed algorithm. In this algorithm, the dataset is randomly divided into several parts and executed on different sites respectively, and the communication between sites depends on the network topology rather than the data set. This method overcomes the problem that the kernel function will increase rapidly with the increase of data set.

In this paper, the objectives and achievements of this paper are mainly as follows: (1). We proposed a distributed algorithm for reduced kernel ELM. This method solves the problem of machine execution difficulty when the kernel matrix becomes larger as the data becomes larger. (2). This distributed algorithm that we proposed achieves faster computation speed than the centralized approach of the past.

II. Related work

A. ELM and KELM

Given the training data $\{(\boldsymbol{x}_i, y_i) \mid \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}^m\}_{i=1}^q$, where $\boldsymbol{x}_i$ is the i-th instance of input sample, y_i is the output vector of the i-th node. Accordingly, the output function is formulated as

$$f(\boldsymbol{x}) = \sum_{i=1}^{L} \beta_i u_i(\boldsymbol{x}) = \boldsymbol{u}(\boldsymbol{x})\boldsymbol{\beta}, \tag{1}$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_L]^\top$, $(i = 1, 2, \ldots, L)$ is the vector of the output weights. $\boldsymbol{u}(\boldsymbol{x}) = [u_1(\boldsymbol{x}), u_2(\boldsymbol{x}), \cdots, u_L(\boldsymbol{x})]$ represents the output vector of the i-th hidden node relative to the input $\boldsymbol{x}$.

$$\mathbf{U} = \begin{bmatrix} \boldsymbol{u}(\boldsymbol{x}_1) \\ \boldsymbol{u}(\boldsymbol{x}_2) \\ \vdots \\ \boldsymbol{u}(\boldsymbol{x}_q) \end{bmatrix} = \begin{bmatrix} u_1(\boldsymbol{x}_1) & \cdots & u_L(\boldsymbol{x}_1) \\ u_1(\boldsymbol{x}_2) & \cdots & u_L(\boldsymbol{x}_2) \\ \vdots & & \vdots \\ u_1(\boldsymbol{x}_q) & \cdots & u_L(\boldsymbol{x}_q) \end{bmatrix}, \tag{2}$$

In the initial implementation of ELM, the standard least square method was used [1], [9],

$$\boldsymbol{\beta} = \mathbf{U}^\dagger \mathbf{Y}, \tag{3}$$

here, $\mathbf{U}^\dagger$ is the Moore-Penrose generalized inverse of matrix $\mathbf{U}$.

$$\mathbf{U}^\dagger = \begin{cases} (\mathbf{U}^\top\mathbf{U})^{-\mathbf{1}}\mathbf{U}^\top & q \geq L \\ \mathbf{U}^\top(\mathbf{U}\mathbf{U}^\top)^{-\mathbf{1}} & q < L \end{cases} \tag{4}$$

For the feedforward neural network with smaller training error, the smaller the weight norm, the better generalization performance of the network. Thus, we need to minimize training errors and output weights, that is

$$\min \frac{1}{2}\|\mathbf{U}\boldsymbol{\beta} - \mathbf{Y}\|^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2, \tag{5}$$

where

$$\mathbf{Y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_q \end{bmatrix}, \tag{6}$$

$\mathbf{Y}$ denotes the output matrix of hidden layer, and $\boldsymbol{y}_i$ $(i = 1, \ldots, q)$ is i-th element of matrix $\mathbf{Y}$.

When $q \geq L$,

$$\boldsymbol{\beta} = (\mathbf{U}^\top\mathbf{U} + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{U}^\top\mathbf{Y}. \tag{7}$$

The classification problem of multiple output nodes for ELM can be described in the following way :

$$\min L_{(ELM)} = \frac{1}{2}\|\boldsymbol{\xi}\|^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2, \tag{8}$$

$$\text{s.t. } \mathbf{U}\boldsymbol{\beta} = \mathbf{Y} - \boldsymbol{\xi} \ . \tag{9}$$

According to previous literature research records, if $\boldsymbol{u}(\boldsymbol{x})$ is unknown, we can define a kernel matrix of the form as

$$\mathbf{K}_{ELM} = \mathbf{U}\mathbf{U}^\top : \mathbf{K}_{ELM\,i,j} = \boldsymbol{u}(\boldsymbol{x}_i) \cdot \boldsymbol{u}(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{10}$$

$$f(\boldsymbol{x}) = \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}_1) \\ k(\boldsymbol{x}, \boldsymbol{x}_2) \\ \vdots \\ k(\boldsymbol{x}, \boldsymbol{x}_q) \end{bmatrix}^\top (\lambda\mathbf{I} + \mathbf{K}_{ELM})^{-1}\mathbf{Y}. \tag{11}$$

Like SVM and LS-SVM, $\boldsymbol{u}(\boldsymbol{x})$ does not need to be known by us. On the contrary, the kernel matrix requires us to provide in advance. However, this work is only based on small data sets. When dealing with large datasets, the size of the kernel matrix will also become very large. This is an important question to be concerned for many research.

B. RKELM

Providing q different samples, $\{(\boldsymbol{x}_i, y_i) \mid \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}^m\}_{i=1}^q$, where q denotes the number of input instance and d is the dimension of the input space, m is the size of output nodes. In the classification problem, m represents the number of sample categories. Unlike the previous kernel-based algorithms, for example, the traditional algorithms [10] and BSGD-SVM [11]. The most striking feature of the BSGD-SVM is the reduction of kernel matrix dimensions from $q \times q$ to $q \times L$ and L is the size of random instances.

$$\sum_{s=1}^{L} \boldsymbol{\beta}_s k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{y}_i, i = 1, 2, \ldots, q, j = 1, 2, \ldots, L. \tag{12}$$

Where, $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_L]^\top$, is a matrix of output weight, and

$$\mathbf{K}_{q\times L} = \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}_1) \\ k(\boldsymbol{x}, \boldsymbol{x}_2) \\ \vdots \\ k(\boldsymbol{x}, \boldsymbol{x}_L) \end{bmatrix} \tag{13}$$

Furthermore, if $\mathbf{K}_{q\times L}$ is a strict positive definite kernel matrix, for different q, when $L = q$, that is

$$\|\mathbf{K}_{q\times q}\boldsymbol{\beta} - \mathbf{Y}\| = \mathbf{0}. \tag{14}$$

Therefore, when $L < q$, we will have $\|\mathbf{K}_{q\times L}\boldsymbol{\beta} - \mathbf{Y}\| < \boldsymbol{\epsilon}$, where, $\boldsymbol{\epsilon}$ is nonzero training error,and generally $L < q$. The above optimization constraint problem is denoted by

$$\min \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{\lambda}{2}\|\mathbf{K}_{q\times L}\boldsymbol{\beta} - \mathbf{Y}\|^{\mathbf{2}}. \tag{15}$$

The output weight vector can be shown as

$$\boldsymbol{\beta} = [\lambda\mathbf{I} + \mathbf{K}_{q\times L}^\top\mathbf{K}_{q\times L}]^{-1}\mathbf{K}_{q\times L}\mathbf{Y}. \tag{16}$$

Thus, we can get the output prediction function as follows:

$$f(\boldsymbol{x}) = \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}_1) \\ k(\boldsymbol{x}, \boldsymbol{x}_2) \\ \vdots \\ k(\boldsymbol{x}, \boldsymbol{x}_L) \end{bmatrix}^\top [\lambda\mathbf{I} + \mathbf{K}_{q\times L}^\top\mathbf{K}_{q\times L}]^{-1}\mathbf{K}_{q\times L}^\top\mathbf{Y}, \tag{17}$$

where, the kernel matrix dimension is reduced from $q \times q$ to $q \times L$.

III. Distributed process

When the size of the kernel matrix increases with the training data, a single machine is no longer sufficient. Therefore, we will introduce the distributed RKELM model based on the ADMM algorithm in this section.

Now, we have the following assumptions: the training data are randomly distributed in L sites, and each site denotes a node within a strongly connected network.

A. ADMM-based distributed RKELM

ADMM is an algorithm for solving a convex optimization problem by breaking it down into smaller parts, each part is easier to work with. It is a commonly distributed method in machine learning [12].

First, we restate problem Eq. (9) in a globally consistent form, introducing a local variable $\boldsymbol{\beta}_i$ to each node and urging them to be equal when converging. Here, the Eq. (15) can be defined as follows [13].

$$\min \frac{1}{2}\sum_{i=1}^{L}\|\mathbf{K}_i\boldsymbol{\beta}_i-\mathbf{Y}_i\|_2^2+\frac{\lambda}{2}\|\boldsymbol{z}_i\|_2^2 \tag{18}$$

$$\text{s.t. } \boldsymbol{\beta}_i=\boldsymbol{z}_i \;\; i=1,2,\cdots,L. \tag{19}$$

Then, we set up the Lagrange function as following:

$$L_\rho(\cdot)=\frac{1}{2}\sum_{i=1}^{L}\|\mathbf{K}_i\boldsymbol{\beta}_i-\mathbf{Y}_i\|_2^2+\frac{\lambda}{2}\|\boldsymbol{z}_i\|_2^2+ \sum_{i=1}^{L}\boldsymbol{\alpha}_i^\top(\boldsymbol{\beta}_i-\boldsymbol{z}_i)+\frac{\rho}{2}\sum_{i=1}^{L}\|\boldsymbol{\beta}_i-\boldsymbol{z}_i\|_2^2, \tag{20}$$

where $\boldsymbol{\alpha}_i$ represents the Lagrange multiplier, ρ is positive and denotes the regularization parameter. The iterative process of ADMM can be summarized as:

$$\boldsymbol{\beta}_i[k+1]:=\underset{\boldsymbol{\beta}_i}{\operatorname{argmin}}\, L_\rho(\boldsymbol{\beta}_i,\boldsymbol{z}_i[k],\boldsymbol{\alpha}_i[k]) \tag{21}$$

$$\boldsymbol{z}_i[k+1]:=\underset{\boldsymbol{z}_i}{\operatorname{argmin}}\, L_\rho(\boldsymbol{\beta}_i[k+1],\boldsymbol{z}_i,\boldsymbol{\alpha}_i[k]) \tag{22}$$

$$\boldsymbol{\alpha}_i[k+1]:=\boldsymbol{\alpha}_i[k]+\rho(\boldsymbol{\beta}_i[k+1]-\boldsymbol{z}_i[k+1]), \tag{23}$$

where, k is the iteration step length. As the number of iterations increases, $\boldsymbol{\beta}[k]$ gradually close to $\boldsymbol{z}[k]$.

Updating $\boldsymbol{\beta}_i[k+1]$, it is a convex quadratic programming problem.

To solve Eq. (21), we just need to work out the equation below.

$$0=\mathbf{K}_i^\top(\mathbf{K}_i\boldsymbol{\beta}_i[k]-\mathbf{Y}_i)+\rho(\boldsymbol{\beta}_i[k]-\boldsymbol{z}_i[k])+\boldsymbol{\alpha}_i[k]. \tag{24}$$

Finally, we get that

$$\boldsymbol{\beta}_i[k+1]=(\mathbf{K}_i^\top\mathbf{K}_i+\rho\mathbf{I})^{-1}(\mathbf{K}_i^\top\mathbf{Y}_i+\rho\boldsymbol{z}_i[k]-\boldsymbol{\alpha}_i[k]). \tag{25}$$

Updating $\boldsymbol{z}_i[k+1]$:

To solve Eq. (22), we just need to work out the equation below.

$$0=\lambda\boldsymbol{z}_i-\sum_{i=1}^{L}\boldsymbol{\alpha}_i[k]-(\boldsymbol{\beta}_i[k+1]-\boldsymbol{z}_i[k+1]). \tag{26}$$

$$\boldsymbol{z}_i[k+1]=\frac{1}{(\lambda+\rho)}(\boldsymbol{\alpha}_i[k]+\rho\boldsymbol{\beta}_i[k+1]). \tag{27}$$

Updating $\boldsymbol{\alpha}_i[p+1]$:

$$\boldsymbol{\alpha}_i[k+1]=\boldsymbol{\alpha}_i[k]+\rho(\boldsymbol{\beta}_i[k+1]-\boldsymbol{z}_i[k+1]). \tag{28}$$

IV. Experimental process

Next, we will analyze the performance of DRKELM. It is verified by comparing the traditional machine learning algorithm RKELM by using the database Landsat Satellite and Page. Landsat satellite database consists of multiple databases consisting of spectral value images of 3×3 neighborhood pixels in the satellite and the classification associated with the central pixels in each community. In this paper, all output labels are normalized within the range of $[-1,1]$. Besides, we select 4435 data for training and 2000 data for testing. The database Page contains 5473 instances. Similarly, we will select 4000 data for training and the other 1473 data for testing. For the classification of this paper, all results were averaged from 50 trials.

Fig. 1 depicts the testing accuracy distributed method DRKELM on the dataset Landsat Satellite. C is between 2^{-30} and 2^{10}, while different L of 200, 400, 600, 800, 1000, until to 3000. It can be seen that when L is fixed, the testing accuracy gradually increases as C gets larger. But when C is constant, the testing accuracy changes relatively little with the change of L. Especially when C and L is very small, there has no over-fitting happen, but when L and C is too large, the over-fitting phenomenon can be observed. This also makes clear that the selection of appropriate C can adjust the over-fitting phenomenon, thus the appropriate parameters of C can guarantee the performance of the model. We will also conduct 50 trials to measure the performance of the distributed method and the centralized approach RKELM. It can be seen from Fig. 3 that the distributed algorithm has a fast computation time. Fig. 2 depicts the primal residual and dual residual of DRKELM on the database Landsat Satellite. From Fig.4, we can get that the objective function Eq.(19) stabilizes at a certain value.

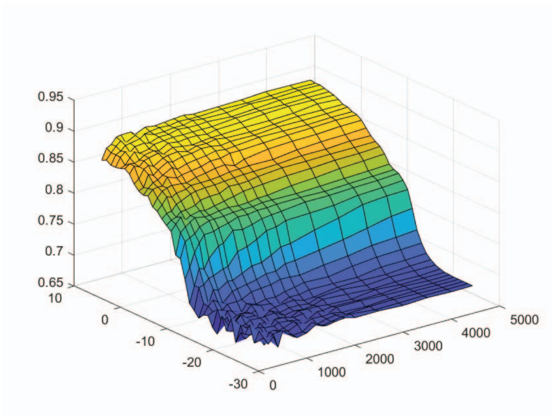

Fig. 1. The testing accuracy on different C and L.

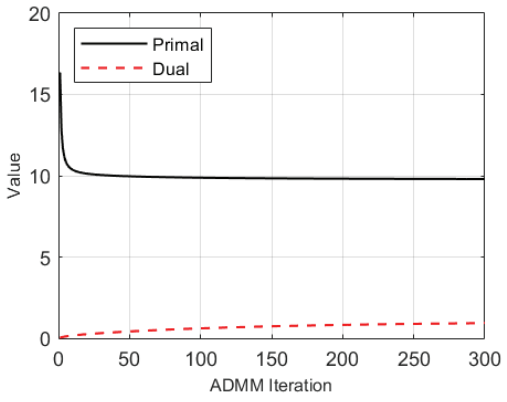

Fig. 2. Primal residual and dual residual of DRKELM.

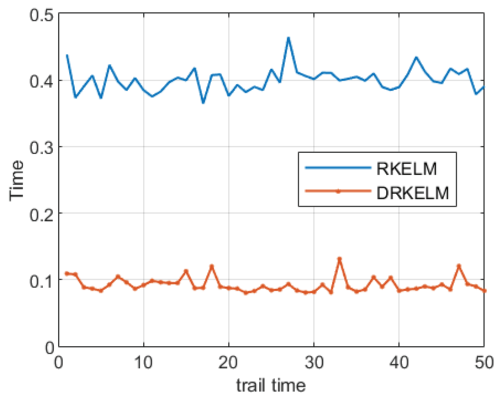

Fig. 3. The training time of two algorithm.

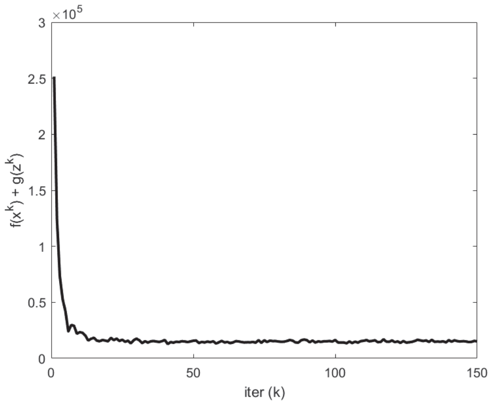

Fig. 4. The result graph of the convergence of the objective function.

V. CONCLUSION

In this paper, an algorithm of RKELM based on the ADMM method in a distributed environment is proposed, the optimal solution of output weight β is obtained through iterative updates. In recent years, many popular machine learning problems have used distributed algorithms because the data scale is getting more and more complex. Under this case, it is considered that the alternating direction method of the multiplier is very suitable for many fields like distributed optimization problems. The results show that the model performance of this distributed method is better and has a shorter computation time compared with the previous centralized algorithm, which greatly saves computation space and time consumption.

REFERENCES

[1] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1/3, pp. 489–501, 2006.

[2] G. B. Huang, "An insight into extreme learning machines: Random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.

[3] P. P. Das, R. Bisoi, and P. K. Dash, "Data decomposition based fast reduced kernel extreme learning machine for currency exchange rate forecasting and trend analysis," *Expert Systems with Applications*, p. S0957417417307352, 2018.

[4] C. X. You, J. Q. Huang, and F. Lu, "Recursive reduced kernel based extreme learning machine for aero-engine fault pattern recognition," *Neurocomputing*, pp. 1038–1045, 2016.

[5] A. Iosifidis and M. Gabbouj, "On the kernel extreme learning machine classifier," *Pattern Recognition Letters*, vol. 68, no. DEC.15, pp. 205–210, 2015.

[6] M. B. Li, G. B. Huang, P. Saratchandran, and N. Sundararajan, "Channel equalization using complex extreme learning machine with rbf kernels," *International Conference on Advances in Neural Networks*, 2006.

[7] W. Y. Deng, Q. H. Zheng, and Z. M. Wang, "Cross-person activity recognition using reduced kernel extreme learning machine," *Neural Networks*, vol. 53, pp. 1–7, 2014.

[8] Zheng, Qing-Hua, Deng, Wan-Yu, Ong, and Yew-Soon, "A fast reduced kernel extreme learning machine," *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 76, pp. 29–38, 2016.

[9] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *IEEE International Joint Conference on Neural Networks*, 2005.

[10] Fernndez-Delgado, Ribeiro, Neves, Cernadas, and Barro, "Direct kernel perceptron (dkp): Ultra-fast kernel elm-based classification with non-iterative closed-form weight calculation," *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 50, pp. 60–71, 2014.

[11] W. Zhuang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3103–3131, 2012.

[12] Scardapane, Simone, Uncini, Aurelio, Wang, Dianhui, Panella, and Massimo, "Distributed learning for random vector functional-link networks," *Information Sciences An International Journal*, 2015.

[13] Forero, A. Pedro, Cano, Alfonso, Giannakis, and B. Georgios, "Consensus-based distributed support vector machines." *Journal of Machine Learning Research*, 2010.