Neural Comput & Applic
DOI 10.1007/s00521-014-1559-3

MR-ELM: a MapReduce-based framework for large-scale ELM training in big data era

Jiaoyan Chen · Huajun Chen · Xiangyi Wan · Guozhou Zheng

Received: 19 November 2013 / Accepted: 10 February 2014

Abstract In the big data era, extreme learning machine (ELM) can be a good solution for the learning of large sample data as it has high generalization performance and fast training speed. However, the emerging big and distributed data blocks may still challenge the method as they may cause large-scale training which is hard to be finished by a common commodity machine in a limited time. In this paper, we propose a MapReduce-based distributed framework named MR-ELM to enable large-scale ELM training. Under the framework, ELM submodels are trained parallelly with the distributed data blocks on the cluster and then combined as a complete single-hidden layer feedforward neural network. Both classification and regression capabilities of MR-ELM have been theoretically proven, and its generalization performance is shown to be as high as that of the original ELM and some common ELM ensemble methods through many typical benchmarks. Compared with the original ELM and the other parallel ELM algorithms, MR-ELM is a general and scalable ELM training framework for both classification and regression and is suitable for big data learning under the cloud environment where the data are usually distributed instead of being located in one machine.

Keywords Extreme learning machine · Big data · MapReduce · Distributed

J. Chen · H. Chen (✉) · X. Wan · G. Zheng
College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
e-mail: huajunsir@zju.edu.cn

1 Introduction

Today, we have entered the big data era, in which data are expanding with an extremely fast speed in different ways [1]. Knowledge discovery from the large-volume, unstructured, distributed and complex data has been widely focused in various fields, such as remote sensing, sensor network, internet, business and biology. Among all the challenges of knowledge discovery, big data learning has become the focus of attention in both industry and academy.

Extreme learning machine (ELM) is a novel machine learning algorithm for single-hidden layer feedforward neural networks (SLFNs) with the capabilities of regression and multiclass classification [2–4]. In ELM, both input weights and hidden layer biases are randomly chosen, and the output weights can be analytically determined through calculating the Moore–Penrose generalized inverse of hidden layer output matrix H. Compared with many traditional training methods, ELM can achieve much faster training speed as well as higher generalization performance. Moreover, ELM is widely applied in real-world data analysis problems including face recognition [5], protein sequence classification [6], signal processing [7] and so on. Accordingly, ELM can be a suitable solution for big data learning.

Although ELM has the advantage of fast training speed, the efficiency and scalability will still be the great challenges in face of big sample data as the big sample number together with the massive hidden nodes usually lead to a huge matrix H and complex matrix calculation. Moreover, the parameter of hidden nodes number usually needs to be optimized by incremental or pruning training algorithms like I-ELM [2, 8] and P-ELM [9], which further improves training scale. In most situations, it is impossible for a

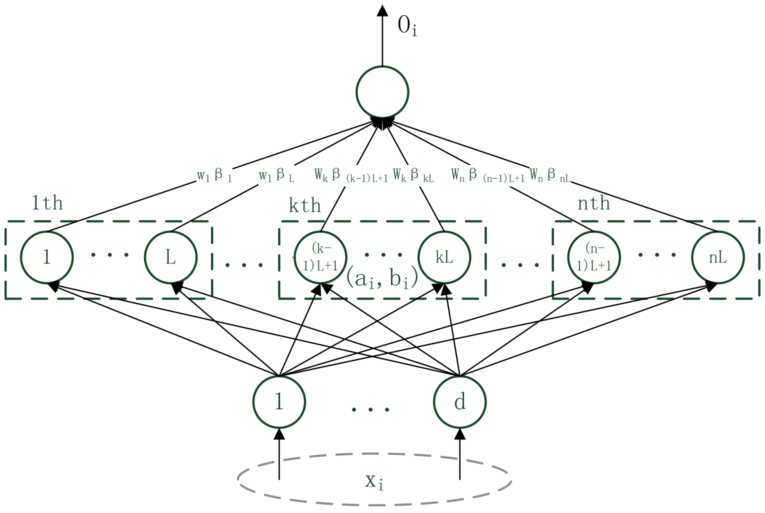


Fig. 1 Distributed network construction framework in MR-ELM

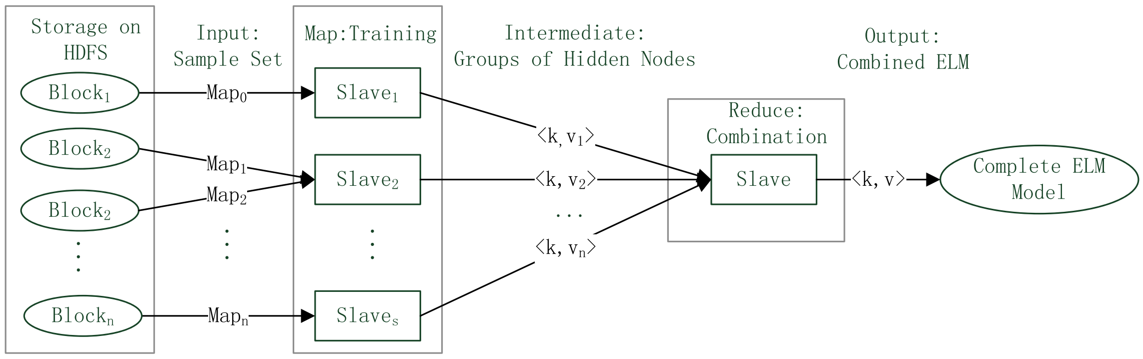


Fig. 2 Workflow of MR-ELM on hadoop cluster

hardware resource limited commodity machine to finish so large-scale ELM training in a limited time because of low efficiency and out-of-memory problem, not alone to the distributed data blocks. Therefore, ELM alone may still encounter some problems to overcome the challenges of big data learning.

Luckily, current cloud computing technology, e.g. MapReduce [10], provides a way to solve the efficiency and scalability problems of big data learning. Usually, these works such as COMET [11] and PLANET [12] apply the distributed computing framework to train multiple models with huge and distributed data blocks and then combine them with the ensemble algorithms. For example, COMET is a single-pass MapReduce-based large-scale learning algorithm which builds multiple random forest ensembles on distributed blocks of data and merges them into a mega-ensemble [11]. Inspired by these works, distributed computing can also be applied to train multiple ELM models and then combine them as a complete ELM model, which is more feasible as SLFN has only one layer of hidden nodes.

According to the above idea, we propose a MapReduce-based distributed ELM training framework named MR-ELM to deal with big data learning. MR-ELM is designed for real-world cloud environment where large-volume sample blocks are located in different nodes of hadoop cluster and can be accessed by hadoop file system (HDFS). Through the framework of MapReduce, training is moved to hadoop nodes, which contributes to high parallelism and costs few I/O resources. Namely, each hadoop node trains an ELM submodel—a group of hidden nodes of SLFN, with the local sample block, and the groups of trained hidden nodes are collected and combined with some algorithms to form the final ELM model. The whole framework of MR-ELM is displayed in Fig. 1, while its workflow on hadoop cluster is shown in Fig. 2.

Under the MR-ELM framework, we further propose the method to combine the ELM submodels. We adopt the least square method to calculate the weights of each groups of hidden nodes for regression problems and simply merge the groups of hidden nodes for classification problems. In this paper, we have theoretically proven that the distributed framework for ELM training has the capabilities of classification and regression, and the generalization performance is shown to be as high as that of the original ELM and some common ELM ensemble algorithms with many typical classification and regression benchmarks.

Briefly, MR-ELM framework is proposed to enable large-scale and distributed ELM training on hadoop cluster, and it contributes to big data learning in the following three aspects.

1. MR-ELM has the capabilities of both classification and regression, and its generalization performance is as high as that of the original ELM and some common ELM ensemble algorithms.
2. MR-ELM is suitable for the learning of distributed big data blocks in cloud environment as it moves training to sample blocks through MapReduce.
3. MR-ELM has high efficiency and scalability on hadoop cluster for big data learning, which is displayed in our experiment with large sample sets.

The remaining of the paper is organized as follows. In Sect. 2, we describe the related work of parallel ELM training. Section 3 gives the details of MR-ELM including the architecture, workflow on the hadoop cluster, classification and regression capabilities and hidden node combination approaches. Section 4 shows our experiments and the result evaluation. Section 5 is the conclusion and some further discussion.

2 Related work

As data volume is so large today, some parallelization methods have been proposed for efficient classification or regression with ELM. Both single ELM and ELM ensemble have been parallelized by P2P networks, GPU and MapReduce.

Heeswijk et al. [13, 14] proposed a GPU-accelerated and parallelized ELM ensemble method for large-scale regression. In the method, individual ELM training as well as its model structure selection was accelerated by GPU. Moreover, multiple GPU and CPU cores were also used for the acceleration of model training and model structure selection. P2P networks have been applied to OS-ELM-based ensemble learning. The framework proposed by Sun et al. [15] applied the incremental learning principle of OS-ELM to the hierarchical P2P network to generate an parallel ensemble classifier. These works usually focused on the training parallelization with parallel computing framework and ignored the features of huge and distributed sample set.

He et al. [16] proposed a parallel ELM method for the regression problems with MapReduce. The algorithm parallelized the calculation of matrix $H^{\mathrm{T}} \times H$ and matrix $H^{\mathrm{T}} \times T$ as the matrix H would be a too large when training data were large. However, the work only focused on the parallelization of the matrix calculation for regression problem through MapReduce and was not to propose a general framework for current distributed big data blocks. The work of Xin et al. [17] is to propose a parallel ELM training framework named ELM*, but its main contribution is also to calculate the matrix multiplication effectively with MapReduce in parallel.

Compared with the above related works, our MR-ELM is to provide a general framework to enable large-scale and distributed ELM training, thus overcoming the problems of big data learning. Although the inspiration of MR-ELM comes from ensemble learning, we further enhance the idea of MapReduce-based ensemble learning [11, 12] when it is applied to ELM with the characteristics of ELM and SLFN. Namely, SLFN has only one layer of hidden nodes, which enables linear combination of multiple SLFNs. Moreover, as the input weights and biases of the hidden nodes are randomly determined, it is reasonable to combine the ELM submodels to form one ELM model through the recalculation of the output weights.

3 MR-ELM: framework for distributed ELM training

3.1 Distributed network construction

In MR-ELM, an SLFN is constructed by combining groups of trained hidden nodes, while each group of hidden nodes is trained parallelly with the ELM algorithm and the local sample block. Training is moved to the nodes that store sample blocks and the trained groups of hidden nodes are collected for combination. To merge the hidden nodes, different methods can be flexibly adopted for high generalization performance. Namely, combination weights of each node w can be calculated according to the real-world problems with specific approaches. The framework of the distributed network construction in MR-ELM is shown in Fig. 1.

In Fig. 1, n groups of hidden nodes are trained by n sample blocks and each group is trained as a whole SLFN with the basic ELM algorithm. Namely, the input weight a, output weight β and biases b of each hidden node are calculated according to local sample block. When combined, both input weights and output weights of each hidden node are remained, while the output weights are recalculated with the combination weights w. In our experiments, w is directly set as 1 for classification or calculated by least square method with resampled data set for regression.

The distributed network construction framework mainly has two advantages. First, it is suitable for the cloud environment in which the big sample data are usually distributed in various nodes. Moving the training tasks to the sample blocks and collecting the trained hidden nodes cost much less I/O resources than collecting all the data blocks together for centralized training. Second, training large network parts by parts greatly releases the memory consumption and the computation complexity, which enables the learning of a big sample set on a single commodity machine. Briefly, the distributed framework provides a suitable approach for large-scale ELM training with big sample blocks.

3.2 Workflow of MR-ELM on hadoop cluster

To implement the framework of MR-ELM, MapReduce [10] parallel programming framework and hadoop [18] middleware are utilized. We assume that the HDFS on the cluster provides the cloud environment to store the distributed sample blocks and the programming interfaces are provided to ensure that the Map processes on each hadoop node can access the local sample blocks. The workflow of the MR-ELM implementation on hadoop cluster is shown in Fig. 2. In the figure, the whole workflow mainly contains two steps: training ELM submodels in the Map processes and calculating the combination weights in the Reduce process.

In Fig. 2, we assume that our hadoop cluster totally contains s slave nodes and the HDFS contains n sample blocks. As each sample block will train one ELM submode, n Map processes are allocated for the training on s slave nodes. The hidden nodes trained by Map processes are transformed to string with carefully designed format. Each Map process outputs one intermediate pair of $<k, v_i>$. The strings of v_i that contain each hidden node's input weights a, biases of hidden nodes b and output weights β are all transferred to the slave node that will execute the Reduce process. In the Reduce process, all the ELM submodes are collected and adopted to calculate the combination weights. After combing all the ELM submodels, the Reduce process outputs information of the whole neural network $<k, v>$ to HDFS.

3.3 Capabilities of classification and regression

After the description of MR-ELM, we are to show that it can work for both classification and regression problems. We define the whole sample set as $D = \{(\mathbf{x_l}, \mathbf{t_l}) | \mathbf{x_l} \in R^d, \mathbf{t_l} \in R^m, l = 1, 2, \ldots, n_0\}$. Assuming D contains n sample distributed sample blocks: $D_i \bigcap D_j = \emptyset, i \neq j$ and $D_1 + D_2 + \cdots + D_n = D$. According to MR-ELM framework, each sample block trains one simple ELM submodel with L hidden nodes. Namely, sample block D_k trains kth group of hidden nodes: $\{(a_j, b_j, \beta_j) | j = (k-1)L + 1, \ldots, kL\}$. After all the hidden nodes are combined, the final neural network contains $L \times n$ hidden nodes.

For sample input $\mathbf{x_i}$, the prediction result of the kth trained ELM submode is represented as $p_k(\mathbf{x_i})$, where

$$p_k(\mathbf{x_i}) = \sum_{j=(k-1)L+1}^{kL} \beta_j G(a_j, b_j, \mathbf{x_i}) \tag{1}$$

The prediction of the whole ELM model can be represented as:

$$\mathbf{o_i} = w_1 p_1(\mathbf{x_i}) + \cdots + w_k p_k(\mathbf{x_i}) + \cdots + w_n p_n(\mathbf{x_i}), \tag{2}$$

where $\mathbf{o_i}$ is the predicted result of $\mathbf{x_i}$ by the final combined ELM model. In the following work of this subsection, we are to show that the predicted result $\mathbf{o_i}$ can be accurate for both situations of classification and regression with carefully calculated combination weight w.

For classification, we can assume that w in Fig. 1 and Formula (2) is directly set to 1, namely $w_i = 1, i = 1, 2, \ldots, n$. In this situation, the prediction result of $\mathbf{x_i}$, namely $\mathbf{o_i}$, can be calculated according to the following formula:

$$\mathbf{o_i} = \mathbf{o_{i1}} + \mathbf{o_{i2}} + \cdots + \mathbf{o_{in}}, \tag{3}$$

where $\mathbf{o_{ik}} = p_k(\mathbf{x_i}), k = 1, 2, \ldots, n$ and is the predicted result of $\mathbf{x_i}$ by the kth ELM submodel. From Formula (3), we can find that each ELM submodel actually outputs a label vector $\mathbf{o_{ik}}$. The value of each attribute of the label vector represents the votes to each possible label. When the label vectors are simply added, the votes to each label from different parts of the neural network are summed. In fact, this procedure of calculation equals to voting-based ensemble algorithm which has already been proven to work for classification problems. Therefore, we can conclude that our MR-ELM framework has the capability of classification.

For regression, we assume that an additional resampled data subset is generated through extracting specific number of samples from each sample block. It is defined as $D_{n+1} = \{(\mathbf{x_i}, \mathbf{t_i}) | \mathbf{x_i} \in R^d, \mathbf{t_i} \in R^m, i = 1, 2, \ldots, N\}$. With the resampled data subset of D_{n+1}, we can show that the framework of MR-ELM can achieve low residual error for the regression problems.

Based on the assumption, we can adopt the least square method in our MR-ELM framework to minimize the residual error of the final ELM model to the resampled data subset. In the least square method, the weights $\{w_k | k = 1, 2, \ldots, n\}$ are optimized to minimize cost function:

$$E = \sum_{i=1}^{i=N} ||\mathbf{o_i} - \mathbf{t_i}||_2 \tag{4}$$

That is equal to minimize $\|PW - T\|$, where

$$P = \begin{bmatrix} p_1(x_1) & \cdots & p_n(x_1) \\ \vdots & \ddots & \vdots \\ p_1(x_N) & \cdots & p_n(x_n) \end{bmatrix}_{N\times n}, W = \begin{bmatrix} w_1^{\mathrm{T}} \\ \vdots \\ w_n^{\mathrm{T}} \end{bmatrix}_{n\times m},$$

$$T = \begin{bmatrix} \mathbf{t_1}^{\mathrm{T}} \\ \vdots \\ \mathbf{t_N}^{\mathrm{T}} \end{bmatrix}_{N\times m}$$

One solution of $||P\hat{W} - T|| = \min||PW - T||$ is:

$$\hat{W} = (P^{\mathrm{T}}P)^{-1}P^{\mathrm{T}}T = P^{\dagger}T, \tag{5}$$

where $P^{\dagger}$ is the Moore–Penrose generalized inverse of matrix P. In combination with the groups of hidden nodes, output weights of the hidden nodes of kth group is updated with the corresponding weight w_k as shown below.

$$\hat{\beta}_j = w_{\lfloor (j-1)/L \rfloor + 1}\beta_j, \quad j = 1, 2, \ldots, n \times L \tag{6}$$

Although the above optimization of the weight w is based on the minimization of the residual error of the resampled data subset, the procedure of resampling and the size of resampled data subset are controllable. Therefore, the MR-ELM framework can always avoid training an ELM model with too large residual error for the regression problems.

3.4 Hidden node combination algorithm

According to the above description, the weights of w should be calculated to combine the ELM submodels. For classification, w is simply set to 1 and the procedure for combination is simple, while for regression, the least square method and a resampled data subset are adopted to optimize the weights w. When the least square method is adopted in MR-ELM framework, the details of the whole procedure based on MapReduce is shown in Algorithm 1.

```
Algorithm 1 Weighted Combination Algorithm under MR-ELM Framework based on MapReduce
Require: Set Reduce number to one and Map number to $n$, define the sample
    blocks as $\{D_1, D_2, \ldots, D_n\}$ and the resample data subset as $D_{n+1}$, hidden
    node number and activation function of each ELM submodel are set to $L$
    and $g$.
    Map 1) Train an ELM submodel with local sample subset $D_k$.
    Map 2) Transform the $L$ hidden nodes of the local ELM submodel to string
    $str\_nodes$.
    Map 3) Output $< map\_id, str\_nodes >$.
    Reduce 1) Collect all the values of $str\_nodes$.
    Reduce 2) Predict each $x$ in $D_{n+1}$ with each ELM: $str\_nodes$.
    Reduce 3) Construct matrix $P$, calculate $P^{\dagger}$ and $W$.
    Reduce 4) Update output weights $\beta$ with $W$.
    Reduce 5) Transform the $n \times T$ hidden nodes of the combined ELM model
    to string $str\_elm$.
    Reduce 6) Output $< reduce\_id, str\_elm >$.
```

In Algorithm 1, there are various mathematical methods, e.g. SVD decomposition, to calculate the Moore–Penrose generalized inverse of a matrix, namely to calculate $P^{\dagger}$, which is the step that causes the main CPU time and hardware resources. However, as resampled subset D_{n+1} is usually a small part of the whole large sample set D, the matrixes of both P and T will not be too large. Therefore, the CPU time and hardware cost for the combination algorithm are limited.

4 Experiments and evaluation

4.1 Experiments

We mainly design two experiments to evaluate the contributions of this paper. The first experiment is designed to display the generalization performance of MR-ELM for the typical classification and regression benchmarks, while the second one is to show the performance of MR-ELM on hadoop cluster for large-scale ELM training with distributed big data blocks.

To implement MR-ELM, hadoop clusters with the size of 6 nodes and 8 nodes are constructed using commodity machines, each of which is configured with 1G RAM, 512M java heap and one processor of Intel(R) Xeon(R) 2.4 GHZ CPU E5620. Two hadoop clusters with different hidden nodes are adopted, because we want to evaluate the influence of the cluster scale to the performance of MR-ELM. The nodes in the cluster are connected by the network with the bandwith of 100 m/s. The framework of MR-ELM as well as the hidden node combination approaches are implemented by the java programming language with JDK 1.6.0_20 and hadoop-1.0.3.

The first experiment mainly tests the testing accuracies of MR-ELM for 9 classification benchmarks and 12 regression benchmarks, both of which come from two famous machine learning repositories, UCI[1] and FCUP[2]. They may be not very large, but that is enough to test the generalization performance of MR-ELM. The details of those benchmarks including sample size and attribute number are shown in Tables 1 and 2. Additionally, both attributes of the classification benchmarks and expected values of the regression benchmarks are normalized for accurate prediction and intuitive comparison.

For each classification or regression benchmark, classification accuracy or residual sum of squares is calculated. Moreover, testing of MR-ELM with each benchmark is repeated multiple times, and we calculate the average value as well as the standard deviation, both of which are token into consideration for the analysis of generalization performance. To accurately evaluate the generalization

[1] http://archive.ics.uci.edu/ml/.

[2] http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html.

Table 1 Benchmarks for classification

Name	Classes	Attributes	Training data	Testing data
Statlog	2	14	570	120
Diabetes	2	8	576	192
Segment	7	19	1,500	810
Digit	10	62	3,823	1,797
Spambase	2	57	4,000	601
Waveform	3	21	4,400	600
Satimage	6	36	4,435	2,000
Page_blocks	5	10	4,500	973
Magic4	2	10	15,000	4,020

Table 2 Benchmarks for regression

Name	Attributes	Training data	Testing data
Puma32H	32	3,000	1,499
Puma8NH	8	3,000	1,499
Bank8FM	8	3,400	1,099
Ailerons	40	4,000	3,154
Delta_ailerons	5	4,000	3,129
Kin8nm	8	5,000	3,192
Sinc	1	5,000	5,000
Cpu_act	8	6,000	2,192
Delta_elevators	6	6,000	4,517
Cal_housing	8	10,000	10,640
Cart_delve	10	20,000	20,768
Fried_delve	10	20,000	20,768

performance, MR-ELM is compared with the original ELM and some common ELM ensemble algorithms for all the benchmarks. For classification, we adopt voting-based ELM ensemble algorithm (V-ELM) which has been proven to have high generalization performance [19]. For regression, we apply a simple Bagging [20] mechanism to ELM ensemble through calculating the average value of all the predicted results, which is called Avg-ELM in this paper.

In order to ensure fairness, the hidden node number of all the three algorithms are optimized and the one that leads to the highest generalization performance is selected. Therefore, the highest testing accuracies or the lowest residual sums of squares that they can achieve are recorded for each benchmark after optimization.

The second experiment is designed to display the efficiency and scalability of MR-ELM for large-scale ELM training on hadoop cluster. The efficiency of MR-ELM can be measured by the value of speedup which is defined in Formula (7). Speedup reflects the acceleration of hadoop cluster for MR-ELM.

$$\text{Speedup} = \frac{\text{computing time on 1 computer}}{\text{computing time on cluster}} \tag{7}$$

The scalability of MR-ELM can be measured by the value of sizeup which is defined in Formula (8). Higher sizeup means the system or the algorithm can deal with increasing data in a limited CPU time.

$$\text{Sizeup} = \frac{\text{computing time for processing m} \times \text{data}}{\text{computing time for processing data}}. \tag{8}$$

To measure speedup and sizeup, the MR-ELM framework is tested with both big classification and regression data sets on two hadoop clusters and on one standalone commodity machine. The big classification sample set called digit contains 640,000 samples, which have 62 attributes and 10 possible classes, while the big regression sample set called cart_delve has 100,000 samples and each sample consists of 10 attributes and one target value. Although both two data sets are not very big absolutely, they are a few orders of magnitude larger than the common sample sets, which makes sense for the evaluation of big data learning. To simulate the real-world cloud environment, both large sample sets are partitioned into blocks and stored in different nodes of the hadoop cluster. In addition, classification sample sets and regression sample sets with different sizes ranging from 100,000 to 1,000,000 are manually generated to simulate increasing processing data for testing of sizeup.

Table 3 Testing accuracy of MR-ELM, ELM and V-ELM for classification

Benchmark	Size	MR-ELM		ELM		V-ELM	
		Mean	SD	Mean	SD	Mean	SD
Statlog	570	0.6600	0.0125	0.6460	0.0182	**0.6607**	0.0270
Diabetes	576	0.7524	0.0251	**0.7635**	0.0151	0.7442	0.7332
Segment	1,500	**0.9412**	0.00395	0.9367	0.0031	0.9326	0.0069
Digit	3,823	**0.9389**	0.01277	0.8570	0.0121	0.9090	0.0067
Spambase	4,000	0.8955	0.00890	**0.8969**	0.0185	0.8926	0.0041
Satimage	4,435	**0.8772**	0.0013	0.8565	0.0058	0.8697	0.0034
Waveform	4,400	0.8883	0.01198	0.8740	0.0084	**0.8987**	0.0071
Page_blocks	4,500	0.9621	0.0021	0.9454	0.0025	**0.9623**	0.0032
Magic4	15,000	**0.8739**	0.0011	0.8630	0.0007	0.8698	0.0023

The highest mean among MR-ELM, ELM and V-ELM are in bold

The lowest SD among MR-ELM, ELM and V-ELM are underlined

Table 4 Residual sum of squares of MR-ELM, ELM and Avg-ELM for regression

Benchmark	Size	MR-ELM		ELM		Avg-ELM	
		Mean	SD	Mean	SD	Mean	SD
Puma32H	3,000	0.1243	<u>0.0003</u>	**0.1239**	0.0005	0.1254	0.0007
Puma8NH	3,000	0.4070	0.0050	0.4315	<u>0.0048</u>	**0.4070**	0.0092
Bank8FM	3,400	**0.0692**	<u>0.0080</u>	0.0898	0.0257	0.0813	0.0111
Ailerons	4,000	0.2264	<u>0.0003</u>	**0.2136**	0.0006	0.2274	0.0005
Delta_ailerons	4,000	**0.0002**	4.4E-6	0.0005	8.3E-6	**0.0002**	<u>3.0E-6</u>
Kin8nm	5,000	**0.2385**	0.0149	0.3217	0.0421	0.2518	<u>0.0037</u>
Sinc	5,000	**0.0087**	<u>0.0016</u>	0.0144	0.0092	0.0307	0.0481
Cpu_act	6,000	0.8776	0.0012	**0.8846**	0.0029	0.8797	<u>0.0005</u>
Delta_elevators	6,000	**0.0019**	0.0001	0.0022	0.0002	0.0021	<u>0.0001</u>
Cal_Housing	10,000	0.4869	<u>0.0002</u>	0.4865	0.0003	**0.4862**	0.0006
Cart_delve	20,000	0.1421	<u>0.0038</u>	**0.1122**	0.0084	0.1547	0.0057
Fried_delve	20,000	**0.1192**	<u>0.0016</u>	0.1292	0.0069	0.1202	0.0024

The highest mean among MR-ELM, ELM and V-ELM are in bold

The lowest SD among MR-ELM, ELM and V-ELM are underlined

Table 5 Training time of MR-ELM with different data blocks on one commodity machine, 6 nodes cluster and 8 nodes cluster

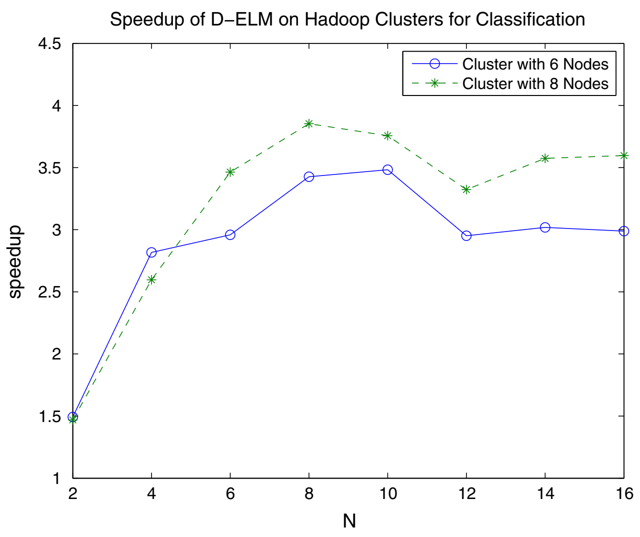
Blocks	Classification			Regression		
	Single (s)	6 nodes (s)	8 nodes (s)	Single (s)	6 nodes (s)	8 nodes (s)
1	OutofMemory	NaN	NaN	OutofMemory	NaN	NaN
2	2,006.0	1,137.4	1,052.1	1,256.9	842.4	855.7
4	1,991.9	607.6	545.1	1,206.7	428.3	464.75
6	1,988.8	596.5	428.7	1,000.8	338.3	289.0
8	1,990.4	495.0	429.0	841.3	245.6	218.3
10	1,969.6	470.6	396.7	667.1	191.6	177.6
12	1,963.0	496.9	394.4	537.6	182.2	161.8
14	1,906.1	505.3	397.2	483.7	160.2	135.3
16	1,847.5	503.7	393.4	461.1	154.2	128.2

4.2 Generalization performance of MR-ELM

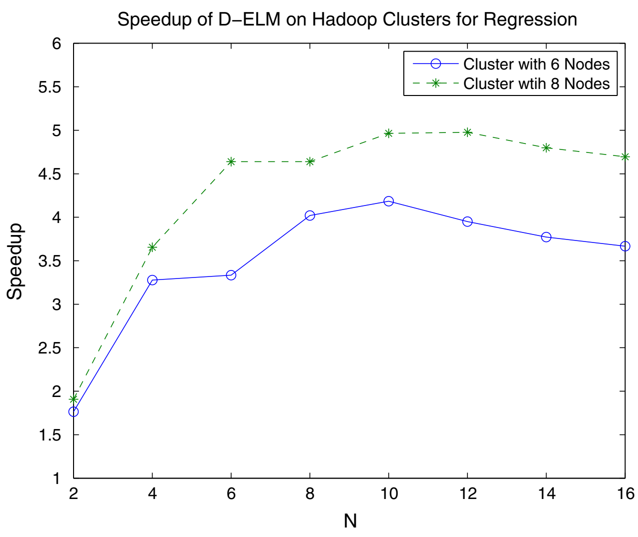
Generalization performance of MR-ELM is evaluated by the first experiment, in which various classification and regression benchmarks are tested and the results are compared with the original ELM and V-ELM. For each algorithm, we optimize the hidden node number and record the highest average testing accuracy or the lowest average residual sum of squares, which are shown in Tables 3 and 4 together with their standard deviations.

From Table 3, we can find that MR-ELM can achieve higher average testing accuracies than ELM and V-ELM for four benchmarks, but the goodness is not very prominent. For the other five benchmarks, V-ELM or the original ELM performs better than MR-ELM. Actually, the average testing accuracies of all the three algorithms are very close for most of the classification benchmarks. To consider prediction stability, MR-ELM has lowest standard deviation for four benchmarks, while V-ELM has the lowest for three benchmarks and the original ELM has the lowest for two benchmarks. In fact, for most of the benchmarks, standard deviations of MR-ELM are in the same order of magnitude with the original ELM and V-ELM. Briefly, we can conclude that MR-ELM can achieve the similar testing accuracy and stability as the original ELM and V-ELM.

As shown in Table 4, MR-ELM outperforms the original ELM and Avg-ELM for six benchmarks in testing accuracy, and it has lower standard deviation for seven benchmarks. Especially for benchmarks of Sinc, Cart_delve and Bank8FM, MR-ELM has much higher performance in both prediction accuracy and stability than V-ELM. For the other seven benchmarks, MR-ELM has higher residual sums of error than the other algorithms, but the gap is only a little. Moreover, the standard deviations of MR-ELM, the original ELM and Avg-ELM are basically in the same magnitude. Therefore, we can conclude that MR-ELM can achieve the similar testing accuracy and stability as the original ELM and the average-based ELM ensemble for regression problems.



(a)



(b)

Fig. 3 Speedup of MR-ELM on hadoop clusters. **a** Classification. **b** Regression

Table 6 Training time of MR-ELM for different data sizes on 6 nodes cluster and 8 nodes cluster

Size	Blocks	Classification		Regression	
		6 nodes (s)	8 nodes (s)	6 nodes (s)	8 nodes (s)
100,000	10	112.2	110.2	71.4	69.7
200,000	20	199.6	170.5	118.5	89.2
300,000	30	252.9	234.0	197.0	114.6
400,000	40	319.8	299.7	264.5	190.2
500,000	50	401.3	353.3	325.2	260.0
600,000	60	448.9	413.9	407.2	445.1
700,000	70	490.9	460.8	553.9	539.5
800,000	80	567.0	533.6	692.5	676.8
900,000	90	524.1	683.9	802.5	775.6
1,000,000	100	677.6	639.2	892.1	793.6

4.3 Performance of MR-ELM on hadoop cluster

Performance of MR-ELM framework on hadoop cluster is evaluated by the results of the second experiment. Through analysis of efficiency and scalability, we prove that MR-ELM is suitable for distributed big data learning in cloud environment.

4.3.1 Efficiency analysis

Using the big classification sample set and regression sample set, MR-ELM is tested on two hadoop clusters and on one commodity machine with different data distributions. Namely, the number of data blocks n ranges from 1 to 16 with a step of 2, which is to ensure that different data distributions are evaluated. The average CPU time of each training is recorded in Table 5.

From Table 5, we can firstly find that training time on hadoop clusters is much less than that on single commodity machine, which means that hadoop clusters accelerate large-scale ELM training greatly. Secondly, training time on single commodity machine and both clusters decreases with the increase in data block number *n*. This phenomenon indicates that distributed training with distributed data can alleviate computing complexity as the dimensions of matrix *H* are shrunk. Thirdly, hadoop cluster with 8 nodes costs less time than the hadoop cluster with 6 nodes, which reflects positive correlation between efficiency and cluster scale. Finally, when the data block is 1, MR-ELM actually becomes the basic ELM algorithm which load the whole sample set into memory, which encounters the out-of-memory problem in our experiment. Therefore, except for efficiency improvement, MR-ELM also enables large-scale ELM training on hardware limited commodity computers.

To describe the efficiency of MR-ELM on hadoop cluster quantitatively, speedups under different data distribution situations are calculated according to Formula (7) and are shown in Fig. 3. From Fig. 3a, we can find that 6 nodes hadoop cluster accelerates MR-ELM more than three times, while 8 nodes hadoop cluster accelerates MR-ELM more than 3.5 times. Speedup of 8 nodes hadoop cluster is 0.5 more than 6 nodes hadoop cluster. When the number of Map processes is less than the number of slave nodes, speedup is smaller as salve nodes are not totally utilized. In Fig. 3b, hadoop clusters accelerates MR-ELM even more times. Six nodes hadoop cluster accelerates MR-ELM 3.5 times more and 8 nodes hadoop cluster accelerates MR-

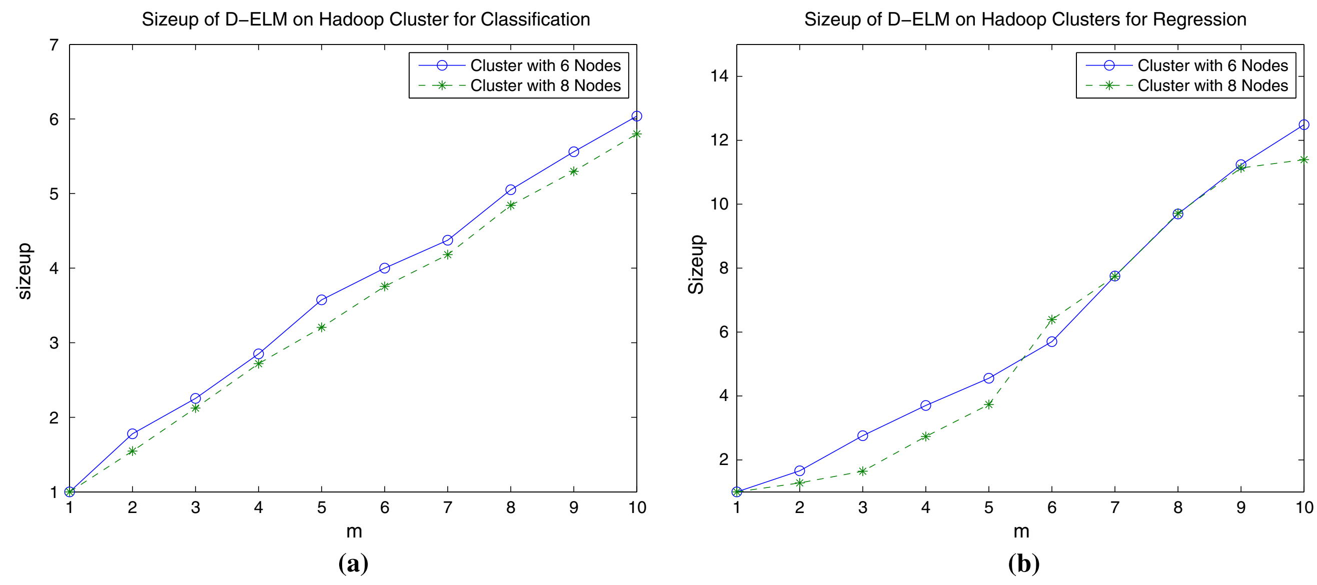


Fig. 4 Sizeup of MR-ELM on hadoop clusters. **a** Classification. **b** Regression

ELM 4.5 times more. Both figures show that MR-ELM on hadoop cluster do have high efficiency.

4.3.2 Scalability analysis

To evaluate the scalability of MR-ELM to big sample data, MR-ELM is tested on hadoop clusters with increasing classification and regression sample sets. When the sizes of both sample sets increase from 100,000 to 1,000,000 with the step of 100,000, ELM training scale increases from 10×50 hidden nodes to 100×50 hidden nodes. Each training is executed multiple times, and the average training time is recorded in Table 6.

In Table 6, training time of MR-ELM on 6 nodes cluster increases from 112.2 to 677.6 s for classification. When size of sample set and scale of SLFN increase to 10 times, training increases to <7 times. The results on 8 nodes hadoop cluster are similar. That means sizeup is <1 and the training system is scalable to increasing learning data.

For regression, training time increases to 12 times more on 6 nodes clusters and 11 times more on 8 nodes cluster when training data size and ELM training scale increase to 10 times. This may be caused by least square algorithm adopted in our experiment for hidden nodes combination as weights calculation is finished by single Reduce process and is not parallelized. Therefore, the results still support the performance of scalability of MR-ELM.

For more details, sizeups of all the data increment are calculated and shown in Fig. 4. In Fig. 4a, sizeup curves of both clusters are almost linear and their slope is much <1. In Fig. 4b, we can find that sizeup is <1 when n is <8, and it exceeds 1 when n is large. This phenomenon strongly supports our argument that least square algorithm adopted instead of MR-ELM framework affects scalability, because weights calculation will be much more complex and takes larger percentage of the whole training time when number of partitions is large. Therefore, the framework MR-ELM itself is scalable enough to increasing scale of ELM training for both classification and regression.

5 Conclusion and discussion

This paper proposes a MapReduce-based distributed ELM training framework named MR-ELM for big data learning. Instead of parallelizing the matrix calculation, MR-ELM is to support distributed training of multiple ELM submodels with distributed data blocks and combine the submodels as a complete ELM model. We firstly prove that MR-ELM has the capabilities of classification and regression when some combination methods are adopted. Our experiment with typical classification and regression benchmarks shows that MR-ELM can achieve as high generalization performance as the original ELM and common ELM ensemble algorithms. Secondly, we display the efficiency and scalability of MR-ELM on hadoop cluster with big sample sets. The high speedups and low sizeups show that MR-ELM is suitable for large-scale ELM training with distributed big data.

Currently, MR-ELM is a distributed and large-scale ELM training framework for big and distributed data. Because of its high scalability and efficiency, it can deal with big static sample data in a limited data with a hadoop cluster. For example, MR-ELM can learn from multiple large remote sensing images which usually have several hundreds of megabytes per image and now are being tried to be stored in HDFS for various advantages [21]. What is more, because of its quick response to big data set, MR-ELM can even be applied to the knowledge discovery of massive data streams which usually have high rates and each window of data needs to be immediately processed.

Meanwhile, MR-ELM should be further improved in some aspects, such as submodel combination algorithm, to meet the properties of big data. For different distributions of the big data, different optimization methods should be proposed for hidden node combination to achieve the highest generalization performance. Moreover, some real-world big and distributed sample blocks together with a large hadoop cluster should be applied in the experiments to further evaluate the generalization performance, efficiency and scalability of MR-ELM.

Acknowledgments This work is funded by LY13F020005 of NSF of Zhejiang, NSFC61070156, YB2013120143 of Huawei and Fundamental Research Funds for the Central Universities.

References

1. Lynch C (2008) Big data: how do your data grow. Nature 455(7209):28–29
2. Huang G-B, Chen L, Siew C-K (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. Neural Netw IEEE Trans 17:879–892
3. Bin Huang G, Yu Zhu Q, Kheong Siew C (2006) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the international joint conference neural network, pp 985–990
4. Huang G-B, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. Syst Man Cybern Part B Cybern IEEE Trans 42:513–529
5. Mohammed A, Minhas R, Wu QJ, Sid-Ahmed M (2011) Human face recognition based on multidimensional pca and extreme learning machine. Pattern Recognit 44(1011):2588–2597
6. Wang D, Bin Huang G (2005) Protein sequence classification using extreme learning machine. In: Neural networks, 2005. IJCNN '05. Proceedings. 2005 IEEE international joint conference on, vol 3, pp 1406–1411
7. Liang N-Y, Huang G-B, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. Neural Netw IEEE Trans 17(6):1411–1423
8. Huang G-B, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71:3460–3468
9. Rong H-J, Ong Y-S, Tan A-H, Zhu Z (2008) A fast pruned-extreme learning machine for classification problem. Neurocomputing 72(13):359–366
10. Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. Commun ACM 51:107–113
11. Basilico J, Munson M, Kolda T, Dixon K, Kegelmeyer W (2011) Comet: a recipe for learning and using large ensembles on massive data. In: Data mining (ICDM), 2011 IEEE 11th international conference on, pp 41–50
12. Panda B, Herbach JS, Basu S, Bayardo RJ (2009) Planet: massively parallel learning of tree ensembles with mapreduce. In: Proceedings of the 35th international conference on very large data bases (VLDB-2009)
13. van Heeswijk M, Miche Y, Oja E, Lendasse A, Verleysen M (2010) Solving large regression problems using an ensemble of gpu-accelerated elms. In: European symposium on artificial neural networks (ESANN) 2010
14. van Heeswijk M, Miche Y, Oja E, Lendasse A (2011) Gpu-accelerated and parallelized elm ensembles for large-scale regression. Neurocomputing 74(16):2430–2437
15. Sun Y, Yuan Y, Wang G (2011) An os-elm based distributed ensemble classification framework in p2p networks. Neurocomputing 74(16):2438–2443
16. He Q, Shang T, Zhuang F, Shi Z (2013) Parallel extreme learning machine for regression based on mapreduce. Neurocomputing 102:52–58
17. Xin J, Wang Z, Chen C, Ding L, Wang G, Zhao Y (2013) Elm*: distributed extreme learning machine with mapreduce. World Wide Web. doi:10.1007/s11280-013-0236-2
18. Apache hadoop. http://hadoop.apache.org/
19. Cao J, Lin Z, Huang G-B, Liu N (2012) Voting based extreme learning machine. Inf Sci 185:66–77
20. Breiman L (1996) Bagging predictors. Mach Learn 24:123–140
21. Xiao Z, Liu Y (2011) Remote sensing image database based on NOSQL database. In: Geoinformatics, 2011 19th international conference on, pp 1–5