CrossMark

METHODOLOGIES AND APPLICATION

Classification with boosting of extreme learning machine over arbitrarily partitioned data

Ferhat Özgür Çatak[1]

Abstract Machine learning-based computational intelligence methods are widely used to analyze large-scale data sets in this age of big data. Extracting useful predictive modeling from these types of data sets is a challenging problem due to their high complexity. Analyzing large amount of streaming data that can be leveraged to derive business value is another complex problem to solve. With high levels of data availability (i.e., Big Data), automatic classification of them has become an important and complex task. Hence, we explore the power of applying MapReduce-based distributed AdaBoosting of extreme learning machine (ELM) to build a predictive bag of classification models. Accordingly, (1) data set ensembles are created; (2) ELM algorithm is used to build weak learners (classifier functions); and (3) builds a strong learner from a set of weak learners. We applied this training model to the benchmark knowledge discovery and data mining data sets.

Keywords Extreme learning machine · AdaBoost · Ensemble methods · MapReduce

1 Introduction

It is clear that there has been an unexpected increase in the quantity and variety of data generated worldwide by computers, mobile phones, and sensors. Just as computer technology evolved, the quantity and variety of data have also increased, becoming more focused on storing every type of data, the so-called Big Data. As the volume of data to build a predictive model increases, the complexity of that training increases too. As a result, building actionable predictive modeling of a large-scale unstructured data set is a definitive Big Data problem. Predictive learning models try to discover patterns of training data and label new data instances to the correct output value. To efficiently handle unstructured large-scale big data sets, it is critical to develop new machine learning methods that combine several boosting and classification algorithms.

Extreme learning machine (ELM) was proposed by Huang et al. (2006b) based on generalized single-hidden layer feedforward networks (SLFNs). The main characteristics of ELM are small training time compared to traditional gradient-based learning methods, high generalization property of predicting unseen examples with multi-class labels and parameter free with randomly generated hidden nodes. ELM algorithm is used in many different areas including document classification (Zhao et al. 2011), bioinformatics (Wang and Wang 2006) multimedia recognition (Zong and Huang 2011; Lan et al. 2013).

In recent years, much computational intelligence research has been devoted to building predictive modeling of distributed and parallel frameworks. In this research, the proposed learning model creates data chunks with varying size and bag of classifier functions using ELM algorithm trained with these arbitrary chosen sub-data sets with AdaBoosting method for large-scale predictions. By creating data chunks from the training data set using the MapReduce paradigm, each subset of the training data set is used to find out the set of ELM ensembles as a single global classifier function.

The main objective of this work is to train large-scale data sets using ELM and AdaBoost. Another objective is to achieve the model's classification performance with same or

✉ Ferhat Özgür Çatak
ozgur.catak@tubitak.gov.tr

[1] TÜBİTAK BİLGEM, Cyber Security Institute, Kocaeli, Turkey

close to the conventional ELM method. Conventional ELM training cannot be applied to large-scale data sets on a single computer because of their complexity. Then experiments section is split into two subsections: "commonly used data sets" in Sect. 5.1.1 and "large-scale data sets" in Sect. 5.1.2. Commonly used data sets are suitable for training on a single computer with the conventional ELM algorithm. We trained these data sets both conventional and proposed methods to show the classification performance changes of the proposed method. Classification performance results are shown in Sect. 5.3.

The contributions of this paper are as follows:

– A generative MapReduce technique-based AdaBoosted ELM classification model is proposed for learning and, thus, faster classification model training is achieved.
– This research proposes a new learning method for AdaBoosted ELM that achieves parallelization both in the large-scale data sets and reduced computational time of learning algorithm.
– Training computations of working nodes are independent of each other, thus minimizing the data communication. The other approaches, including Support Vector Machine training, need data communication for the support vector exchange (Lu et al. 2008; Sun and Fox 2012; Catak et al. 2013).

The rest of the paper is organized as follows: Sect. 2 briefly introduces some of the earlier works related to our problem. Section 3 describes algorithm ELM, AdaBoost and MapReduce technique. Sections 4 and 5 evaluate the proposed learning model. Section 6 concludes this paper.

2 Related work

In this section, we describe the general overview of the literature review. Section 2.1 describes the general distributed ELM methods. Section 2.2 shows the MapReduce-based ELM training methods.

2.1 Literature review overview

MapReduce-based learning algorithms from distributed data chunks have been studied by many researchers. Many different MapReduce-based learning solutions over arbitrary partitioned data have been proposed recently. Some popular MapReduce-based solutions to train machine learning algorithms in the literature include the following. Panda et al. proposed a learning tree model which is based on the series of distributed computations, and implements each one using the MapReduce model of distributed computation (Panda et al. 2009). Zhang et al. (2012) develop some algorithms

using MapReduce to perform parallel data joins on large-scale data sets . Sun et al. (2009) use batch updating-based hierarchical clustering to reduce computational time and data communication. Their approach uses co-occurrence-based feature selection to remove noisy features and decrease the dimension of the feature vectors. He et al. proposed parallel density-based clustering algorithm (DBSCAN). They developed a partitioning strategy for large-scale non-indexed data with a 4-stages MapReduce paradigm (He et al. 2011). Zhao et al. (2009) proposed parallel k-means clustering based on MapReduce. Their approaches focus on implementing k-means with the read-only convergence heuristic in the MapReduce pattern.

2.2 MapReduce-based ELM training methods

Sections 2.2.1, 2.2.2, 2.2.3, 2.2.4 and 2.2.5 describe five different MapReduce training methods of ELM algorithm.

2.2.1 ELM⋆

Xin et al. proposed MapReduce-based ELM training method called as ELM* (Xin et al. 2014). The main idea behind this method is to calculate matrix multiplication of ELM to find weight vector. They show that Moore–Penrose generalized inverse operator is the most expensive computation part of the algorithm. As we know, matrix multiplication can be divided into smaller parts. Using this property, they proposed an efficient implementation of training phase to manage massive data sets. The final output of this method is a single classifier function. In this paper, they proposed two different versions of ELM*, naive and improved. In naive-ELM*, the algorithm has two classes, Class Mapper and Class Reducer. Both classes contain only one method. In improved ELM*, they decompose the calculation of matrix multiplication using MapReduce framework. Moreover, the proposed algorithm decreases the computation and communication cost. In the experimental platform, they used their synthetic data sets to evaluate the performance of the proposed algorithms with MapReduce framework.

2.2.2 OS-ELM-based classification in hierarchical P2P network

Sun et al. proposed OS-ELM (Liang et al. 2006)-based distributed ensemble classification in P2P networks (Sun et al. 2011). They apply the incremental learning principle of OS-ELM to hierarchical P2P network. They proposed two different versions of the ensemble classifier in hierarchical P2P, one-by-one ensemble classification and parallel ensemble classification. In one-by-one learning method, each peer, one by one, calculates the classifier with all the data. Therefore, this approach has a large network delay. In the parallel

ensemble learning, all the classifiers are learnt from all the data in a parallel manner. Conversely to ELM*, their experimental results are based on three different real data sets downloaded from the UCI repository.

2.2.3 Parallel online sequential ELM: POS-ELM

Wang et al. (2013) have been proposed parallel online sequential extreme learning machine (POS-ELM) method. The main idea behind in this approach is to analyze the dependency relationships and the matrix calculations of OS-ELM (Liang et al. 2006). Their experimental results are based on nine different real data sets downloaded from the UCI repository.

2.2.4 Distributed and kernelized ELM: DK-ELM

Bi et al. (2013) have been proposed both distributed and kernelized ELM (DK-ELM) based on MapReduce. The difference between ELM and kernelized ELM is that K-ELM applies kernels opposite to create random feature mappings. They provide a distributed implementation RBF kernel matrix calculation in massive data learning applications. Their experimental results are based on four different real data sets downloaded from the UCI repository and four synthetic data sets.

2.2.5 ELM-MapReduce

Chen et al. (2013) have been proposed MapReduce-based ELM ensemble classifier called ELM-MapReduce, for large-scale land cover classification of remote sensing data. Their approach contains two sequential phases: parallel training of multiple ELM classifiers and voting mechanism. In parallel training phase of proposed method, each Map function computes an ELM classifier with a given training data set. In second phase called voting mechanism, a new MapReduce job is executed with a new partitioned test set into each Map function with notation $data_j$. In $Reduce$ function of this phase, each $data_j$ is predicted with each ELM classifier trained in parallel training phase. Final classification predictions are the output of final $Reduce$ function. Therefore, this

approach has a high communication cost. Their experimental results are based on synthetic remote sensing image of the training data.

2.3 The differences between proposed model and literature review

The main differences are:

– In ELM★, they use matrix multiplication decomposition. Each Map function is responsible to calculate the Moore–Penrose generalized inverse operation. And their method produces one single classifier. In the proposed model in our paper, each $Reduce$ function produces ensemble classifier based on AdaBoost method. The final output ensemble classifier is a voting-based combination of ensemble classifier trained in each $Reduce$ phase.
– In OS-ELM-based classification in hierarchical P2P Network, POS-ELM and DK-ELM, they propose ensemble classifier that combines multiple classifier trained with data chunks. Each peer classifier is learned from the local data. Therefore, each peer produces a single ELM classifier. In our method, each node (or peer) produces ensemble classifier to increase the classification accuracy.
– In ELM-MapReduce, they propose ensemble classifier with two different MapReduce jobs. In first MpaReduce job, their approach produces a single ELM classifier in each Map function. In second MapReduce job, the test set is partitioned into each Map function and produces final predicted labels based on the voting mechanism of ELM classifiers that are trained in the first MapReduce job. In our method, prediction is not included; our aim is to create a final ensemble classifier in only one MapReduce job.

Table 1 shows the main differences of all proposed methods. There are five different columns that are ensemble methods, single pass MapReduce, matrix multiplication, entire data set and network communication. Ensemble column shows that the method builds a set of classifier function

Table 1 The differences between proposed model and literature review

Method	Ensemble	Single pass MapReduce	Matrix multiplication	Entire data set	Network communication
ELM★	No	Yes	No	Yes	No
OS-ELM	Yes	Yes	No	No	Yes
POS-ELM	Yes	Yes	No	Yes	No
DK-ELM	Yes	Yes	No	Yes	No
ELM-MapReduce	Yes	No	No	Yes	Yes
Proposed method	Yes	Yes	No	No	No

(i.e., ensemble model) to improve the accuracy performance of the final classification model. If an ensemble method is applied, then the performance of final model will have better accuracy result (Kuncheva and Whitaker 2003). Single pass MapReduce column shows that an iterative approach is not applied to the model. Entire learning phase is performed in a single pass of data through the job. Matrix multiplication column shows that the hidden layer matrix is calculated in each *Map* function. The hidden layer matrix computation is a compute intensive operation. Entire data set column shows that each *Map* operation needs entire data set to build a final classifier model. Network communication column shows that each *MapReduce* job needs to communicate with another job. Network communication will affect negatively on training time of the algorithm.

3 Preliminaries

In this section, we introduce preliminary knowledge of ELM, AdaBoost and MapReduce briefly.

3.1 Extreme learning machine

ELM was originally proposed for the single-hidden layer feedforward neural networks (Huang et al. 2006a, b). Then, ELM was extended to the generalized single-hidden layer feedforward networks where the hidden layer may not be neuron like (Huang and Chen 2005, 2006). The main advantages of the ELM classification algorithm are that ELM can be trained hundred times faster than traditional neural network or support vector machine algorithm since its input weights and hidden node biases are randomly created and output layer weights can be analytically calculated using a least-squares method (Tang et al. 2015; Huang et al. 2008). The most noticeable feature of ELM is that its hidden layer parameters are selected randomly.

Given a set of training data $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, n\}, \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{1, 2, \ldots, K\}\}$ sampled independently and identically distributed (i.i.d.) from some unknown distribution. The goal of a neural network is to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{X}$ is instance and $\mathcal{Y}$ is the set of all possible labels. The output label of an single hidden-layer feedforward neural networks (SLFNs) with N hidden nodes can be described as:

$$f_N(\mathbf{x}) = \sum_{i=1}^{N} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}), \ \mathbf{x} \in \mathbb{R}^n, \ \mathbf{a}_i \in \mathbb{R}^n \quad (1)$$

where $\mathbf{a}_i$ and b_i are the learning parameters of hidden nodes and β_i is the weight connecting the ith hidden node to the output node.

The output function of ELM for generalized SLFNs can be identified by

$$f_N(\mathbf{x}) = \sum_{i=1}^{N} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) = \beta \times h(\mathbf{x}) \quad (2)$$

For the binary classification applications, the decision function of ELM becomes

$$f_N(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{N} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})\right) = \mathrm{sign}\left(\beta \times h(\mathbf{x})\right) \quad (3)$$

Equation 2 can be written in another form as:

$$H\beta = T \quad (4)$$

where H and T are, respectively, hidden layer matrix and output matrix. Hidden layer matrix can be described as:

$$H(\tilde{a}, \tilde{b}, \tilde{x}) = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix}_{N \times L} \quad (5)$$

where $\tilde{a} = a_1, \ldots, a_L, \tilde{b} = b_1, \ldots, b_L, \tilde{x} = x_1, \ldots, x_N$. Output matrix can be described as:

$$T = \begin{bmatrix} t_1 \ldots t_N \end{bmatrix}^T \quad (6)$$

The hidden nodes of SLFNs can be randomly generated. They can be independent of the training data.

3.2 AdaBoost

The AdaBoost (Freund and Schapire 1995) is a supervised learning algorithm designed to solve classification problems (Freund et al. 1999). The algorithm takes as input a training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where the input sample $\mathbf{x}_i \in \mathbb{R}^p$, and the output value, y_i, in a finite space $y \in 1, \ldots, K$. AdaBoost algorithm assumes, like ELM, a set of training data sampled independently and identically distributed (i.i.d.) from some unknown distribution $\mathcal{X}$.

Given a space of feature vectors X and two possible class labels, $y \in \{-1, +1\}$, AdaBoost goal is to learn a strong classifier $H(\mathbf{x})$ as a weighted ensemble of weak classifiers $h_t(\mathbf{x})$ predicting the label of any instance $\mathbf{x} \in X$ (Landesa-Vzquez and Alba-Castro 2013).

$$H(\mathbf{x}) = \mathrm{sign}(f(\mathbf{x})) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) \quad (7)$$

Pseudocode for AdaBoost is given in Alg. 1

Algorithm 1 The AdaBoost algorithm.

1: **Inputs:**
   $\mathcal{D} = \{\{(\mathbf{x}_i, y_i) \mid i = 1, ..., n\}, \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{-1, +1\}\}_{i=1}^m$
2: Initialize $\mathcal{D}(i) = \frac{1}{m}$ for all i
3: **while** $t < T$ **do**
4:      Train **WeakLearner** using distribution $\mathcal{D}_t$
5:      get back a weak hypothesis $h_t : X \rightarrow \{1, 2, ..., K\}$
6:      calculate the error of $h_t : \epsilon_t = Pr_{i \sim \mathcal{D}_t}[h_t(\mathbf{x}_i) \neq y_i]$
7:      Sets $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
8:      update distribution $\mathcal{D}_{t+1} = \frac{\mathcal{D}_t}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } h_t = y_i \\ e^{\alpha_t}, & \text{if } h_t \neq y_i \end{cases}$
9:      equivalently $\mathcal{D}_{t+1} = \frac{\mathcal{D}_t \times exp(-\alpha_t y_i h_t(x_i))}{Z_t}$ where Z_t is a normalization constant.
10: **end while**
11: **Outputs:**
   final hypothesis $h^* = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

3.3 MapReduce

MapReduce is a new programming model to run parallel applications for large-scale data sets processing to support data-intensive applications. It is derived from the map and reduce function combination from functional programming. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. The MapReduce was originally developed by Google and built on principles in a parallel manner (Dean and Ghemawat 2008). The MapReduce framework first takes the input, divides it into smaller data chunks, and distributes them to worker nodes. MapReduce is divided into three major phases called map, reduce and a separated internal shuffle phase. The MapReduce framework automatically executes all those functions in a parallel manner over any number of processors/servers (Schatz 2009).

Pseudo-code of MapReduce framework is shown in Eq. 8.

$$\text{map}(key_1, value_1) \rightarrow \text{list}(key_2, value_2)$$
$$\text{reduce}(key_2, \text{list}(value_2)) \rightarrow \text{list}(key_3, value_3) \quad (8)$$

Mapreduce programming technique is widely used on different scientific fields, i.e., cyber-security (Choi et al. 2014; Ogiela et al. 2014), high energy physics (Bhimji et al. 2014), and biology (Xu et al. 2014).

4 Proposed approach

In this section, we provide the details of the MapReduce-based distributed AdaBoosted ELM algorithm. The basic idea of AdaBoost-ELM based on MapReduce technique is introduced in Sect. 4.1. The MapReduce implementation of AdaBoosted ELM is described in Sect. 4.3.

Table 2 Commonly used variables and notations

Variables/notation	Description
M	Data chunk split size
h	A single classifier function
X_m	Data chunk m of input values of $\mathcal{D}$
Y_m	Data chunk m of output values of $\mathcal{D}$
ϵ	Error rate
# Chunk	Number of data chunk
T	AdaBoost T size
# H. Nodes	Number of hidden nodes used in ELM
Acc	Accuracy of classifier hypothesis
k	Number of classes

4.1 Basic idea

Our main task is to parallel and distributed execute the computation of AdaBoosted ELM classification method. AdaBoosted ELM's basic idea is to calculate ensemble of classifier functions over partitioned data (X_m, Y_m) in a parallel manner. In Table 2, a summary of commonly used variables and notations to assess the classifier model performance of the AdaBoosted ELM method is given for convenience.

4.2 Analysis of the proposed algorithm

Barlett showed that the size of the weights is more important than the size of the neural network (Bartlett 1998). Kragh et al. also showed that ensemble methods of neural networks get better accuracy performance over unseen examples (Krogh and Vedelsby 1995). The main motivation of the this work is the idea that small size ELM ensembles can obtain more accurate classifier model that are comparable to individual classifiers.

In the proposed model, at every data chunk, there is a set of classifier functions that acts as a single classification model. The single model at every data chunk m is defined as follows:

$$f^{(m)}(\mathbf{x}) = \arg\max_k \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \quad (9)$$

The selected ensemble ELM classifier models from the reduce phase of MapReduce algorithm are combined into one single classification model.

$$\hat{h}(\mathbf{x}) = \arg\max_k \sum_{i=1}^{m} f^{(m)}(\mathbf{x}) \quad (10)$$

4.3 Implementation of the model

The pseudo-codes of MapReduce-based AdaBoost ELM are shown in Algorithm 2 and Algorithm 3. The *Map* proce-

dure of our training model is implemented based on random assignment of each row of the training data set with split size of data, M, in line 2 of Algorithm 2. The input, $\mathbf{x}$, is a row of training data set $\mathcal{D}$. Map procedure partition the input matrix by row, producing $< randomSplitId, \mathbf{x} >$ key-value pairs. $randomSplitId$ is the identifier of the data chunk and is transferred as the input key to $Reduce$ phase. The pseudo-code of $Reduce$ phase is shown in Algorithm 3. $Reduce$ procedure is implemented based on the for-loop of lines 3 - 8 of Algorithm 3. The output ELM classifier of sub-data set $(\mathbf{X}_k, \mathbf{y}_k)$ is calculated using AdaBoost constantly block by block, so every reduce task completes training phase and outputs an AdaBoosted set of classifier functions. The $mapper$'s input k is the $randomSplitId$ to create the data chunk and created in the Map phase of our training model.

Algorithm 2 AdaBoostedELM::Map

1: **Inputs:**
   Training record $(\mathbf{x}, y) \in \mathcal{D}$, Data set split size M
2: $k \leftarrow rand(0, M)$
3: $Output(k, (\mathbf{x}, y))$

Algorithm 3 AdaBoostedELM::Reduce

1: **Inputs:**
   Key k, Value Set V, AdaBoost Iteration Size T
2: Split V into input space $\mathbf{X}_n$ and out labels $\mathbf{y_n}$ with $(\mathbf{X}_n, \mathbf{y}_n) \leftarrow V$
3: **for** $t = 1..T$ **do**
4:    Train sub data set with ELM: $h_t \leftarrow ELM(\mathbf{X}, y)$
5:    $\mathbf{y}_{pred}, \epsilon_t \leftarrow h_t(X)$
6:    $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$
7:    $\mathcal{D}_{t+1} = \frac{\mathcal{D}_t \times exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
8: **end for**
9: **Outputs:**
   Final hypothesis for the reduce function m :
   $h_m \leftarrow \arg\max_k \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$

---

5 Experiments

In this section, we perform experiments on real-world data sets from the public available data set repositories. Public data sets are used to evaluate the proposed learning method. Then, classification models of each data set are compared for accuracy results with the single instance of learning algorithm performance.

In Sect. 5.1, we explain the data sets and parameters that are used in experiments. The conventional ELM is applied all data sets and we find the accuracy performance over number of hidden nodes in Sect. 5.3. In Sect. 5.2, we show the empirical results of proposed distributed Adaboost ELM training algorithm.

5.1 Experimental setup

In this section, we apply our approach to five different data sets to verify its effectivity and efficiency. To demonstrate the effectiveness and performance of the proposed model, we apply it on various classification data sets from public

Table 3 Description of the testing data sets used in the experiments

Data set	#Train	#Test	#Classes	#Attributes
Pendigit	7494	3498	10	16
Skin	220,543	24,507	2	3
Statlog/Shuttle	43,500	14,500	7	9
Page-blocks	4500	973	5	10
Waveform	4400	600	3	21

data set repositories. To obtain an optimal value of Mapper size, m, we range it in the range from 20 to 100.

5.1.1 Commonly used classification data sets

We experiment on five public data sets which are summarized in Table 3, including Pendigit, Letter, Statlog, Page-blocks and Waveform. They are all multiclass data sets. All experiments are repeated 5 times and the results are averaged. All data sets are publicly available in svmlight format on the LIBSVM website (LIBSVM 2015).

Pendigit data set is a collection of pen-based recognition of handwritten digits (Alimoglu and Alpaydin 1996). The data set contains 250 samples from 44 people. The first 7494 instances written by 30 people are used for the training data set, and the digits written by other 14 people are used for the independent testing purpose.

Skin data set is a collection of skin segmentation constructed over R, G, B color space (Bhatt et al. 2009). The data set contains face images of different age groups (young, middle, old), genders and racial groups (White, Black, Asian). The data set contains 245,057 instances; out of which 50,859 is the skin labeled instances and 194,198 is non-skin instances.

Statlog/shuttle data set is a collection of space shuttle created by NASA (Hsu and Lin 2002). The data set contains 43,500 training instances and 14.500 testing instances. 80 % of the data belong to class 1.

Page blocks data set is a collection of page layout of a document that has been detected by a segmentation process (Malerba et al. 1996). The data set contains 4500 training instances and 973 testing instances.

Waveform data set is a collection of Breiman's waveform domains of CART book's (Breiman et al. 1984). The data set contains 4400 training instances and 600 testing instances.

5.1.2 Large-scale classification data sets

We experiment on three public large-scale data sets which are summarized in Table 4, including "Record Linkage Comparison Patterns (Donation)", "SUSY" and "HIGGS". All

Table 4 Description of the testing large-scale data sets used in the experiments

Data set	#Train	#Test	#Classes	#Attributes
Donation	5,749,132	1,000,000	2	12
SUSY	5,000,000	50,000	2	18
HIGSS	11,000,000	1,000,000	2	28

experiments are repeated 5 times and the results are averaged.

Donation represent individual data, including first and family name, sex, date of birth and postal code, which were collected through iterative insertions in the course of several years. The comparison patterns in this data set are based on a sample of 100.000 records dating from 2005 to 2008 (Schmidtmann et al. 2009). The data set contains 5,749,132 training instances and 1,000,000 testing instances. The data set is available on UCI website (UCI 2011).

SUSY is a classification data set that distinguishes between a signal process which produces supersymmetric particles and a background process which does not (Baldi et al. 2014). The first 8 features are kinematic properties measured by the particle detectors in the accelerator. The last ten features are functions of the first 8 features. The data set contains 5,000,000 training instances and 50,000 testing instances. The data set is available on UCI website (UCI 0000).

HIGSS is a classification problem to distinguish between a signal process which produces Higgs bosons and a background process which does not (Baldi et al. 2014). The first 21 features (columns 2–22) are kinematic properties measured by the particle detectors in the accelerator. The last seven features are functions of the first 21 features. The data set contains 11,000,000 training instances and 500,000 testing instances. The data set is available on UCI website (UCI 2014).

5.2 Evaluation

Since the data sets that are used in our experiments are highly imbalanced, traditional accuracy-based performance evaluation is not enough to find out an optimal classifier. We used four different metrics, the overall prediction accuracy, average recall, average precision (Turpin and Scholer 2006) and F score, to evaluate the classification accuracy, which are common measurement metrics in information retrieval (Manning et al. 2008; Makhoul et al. 1999).

Precision is defined as the fraction of retrieved samples that are relevant. Precision is shown in Eq. 11.

$$\text{Precision} = \frac{\text{Correct}}{\text{Correct} + \text{False}} \tag{11}$$

Recall is defined as the fraction of relevant samples that is retrieved. Recall is shown in Eq. 12.

$$\text{Precision} = \frac{\text{Correct}}{\text{Correct} + \text{Missed}} \tag{12}$$

The proposed evaluation model calculates the precision and recall for each class from prediction scores and then finds their mean. Average precision and recall are shown in Eqs. 13 and 14.

$$\text{Precision}_{\text{avg}} = \frac{1}{n_{\text{classes}}} \sum_{i=0}^{n_{\text{classes}}-1} \text{Prec}_i \tag{13}$$

$$\text{Recall}_{\text{avg}} = \frac{1}{n_{\text{classes}}} \sum_{i=0}^{n_{\text{classes}}-1} \text{Recall}_i \tag{14}$$

F-measure is defined as the harmonic mean of precision and recall. The

$$F_1 = 2 \times \frac{\text{Prec}_{\text{avg}} \times \text{Recall}_{\text{avg}}}{\text{Prec}_{\text{avg}} + \text{Recall}_{\text{avg}}} \tag{15}$$

5.3 Data set results with conventional ELM

Figure 1 shows that the accuracy performance of ELM for experimental data sets becomes steady state after a threshold value of N. The testing classification performance is measured through accuracy, precision, recall and F_1 measure. N varies from 150 to 500.

Table 5 shows the best performance of the conventional ELM method of each data set.

The conventional ELM training algorithm can be applied only in Sect. 5.1.1. The large-scale data sets in Sect. 5.1.2 are not feasible to train on a single computer.

5.4 Testing accuracy analysis

Because two different data set types ("commonly used", "large scale") are used, the results are divided into two different sections. In Sect. 5.4.1, the figures and the plots show the implementation results of commonly used classification data sets. Section 5.4.2 shows the large-scale data sets results.

5.4.1 Commonly used classification data sets

The results of accuracy and performance tests with real data are shown in Table 6 and Figs. 2, 3, 4, 5 and 6. According to the these results, AdaBoost T size and Mapper size have more impact on the accuracy of ensemble ELM classifier than number of hidden nodes in ELM network.

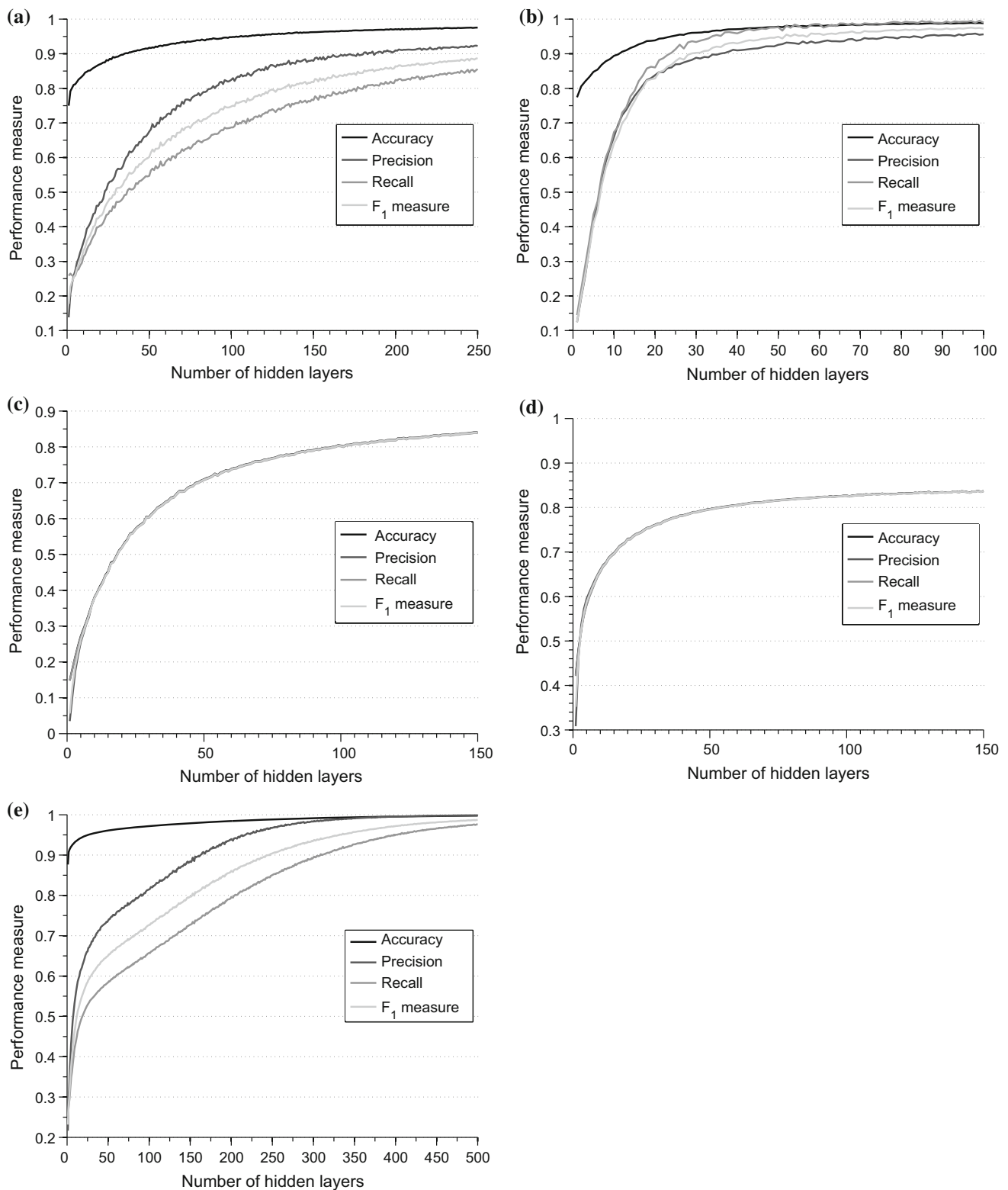The accuracy of classification models is visualized by heatmap color coding according to

Fig. 1 Number of hidden nodes in ELM versus classifier precision. **a** Statlog data set, **b** Skin data set, **c** Pen digit data set, **d** Waveform data set, **e** Page blocks data set

– Map size (M)–AdaBoost size (T)
– Map size (M)–Number of hidden nodes (nh)
– AdaBoost size (T)–Number of hidden nodes (nh)

Figure 2, 3, 4, 5 and 6 are used to plot the quantitative differences in accuracy score of each data set. Heatmaps are two-dimensional graphical representations of data with

Table 5 Data set results with conventional ELM

Data set	F_1	Recall	Precision	Accuracy
Pendigit	0.8404	0.8393	0.8416	0.8407
Skin	0.9754	0.9956	0.9583	0.9894
Statlog	0.8871	0.8556	0.9237	0.9757
Page-blocks	0.9873	0.9764	0.9988	0.9977
Waveform	0.8372	0.8368	0.8375	0.8376

Table 6 Best performance results of data sets

Data set	# C.	T	# H.N.	Acc	Prec.	Recall	F_1
Pendigit	20	10	21	0.8256	0.8369	0.8234	0.8301
Skin	21	5	21	0.9892	0.9773	0.9913	0.9842
Statlog	11	2	21	0.9103	0.7486	0.5069	0.6045
Page Blocks	1	1	340	0.9404	0.9027	0.5756	0.7030
Waveform	19	6	40	0.862	0.8680	0.8605	0.8642

a pre-defined colormap to display the values of a matrix (Khomtchouk et al. 2014). Heatmaps can be used to understand what parameters affect the accuracy of the classification model. The figures are used to comparatively illustrate accuracy levels across a number of different parameters including

Map size, AdaBoost size and the number of hidden nodes in ELM algorithm obtained from the proposed learning method.

According to Table 7, classification performance results of the proposed method have almost the same values with the conventional ELM method.

5.4.2 Large-scale classification data sets

Figure 7 shows the speed up on mapper size over proposed method on large-scale data sets. To asses the effectiveness of the learning algorithm, the time is measured with varying mapper size. Because of high dimensionality, the data sets cannot be trained on a single computer. Then, the standard speed up percentage is modified, such that

$$S_p = \frac{t_{\arg\min m \in M}}{t_p} \qquad (16)$$

where $t_{\arg\min m \in M}$ is the total time on minimum mapper that can be achieved to build a classifier model.

As can be seen from the figure, the data sets achieve performance improvement in learning time of the algorithm. By examining the trends observed as the number of mappers increases, one can see that non-linear speed up is achieved.
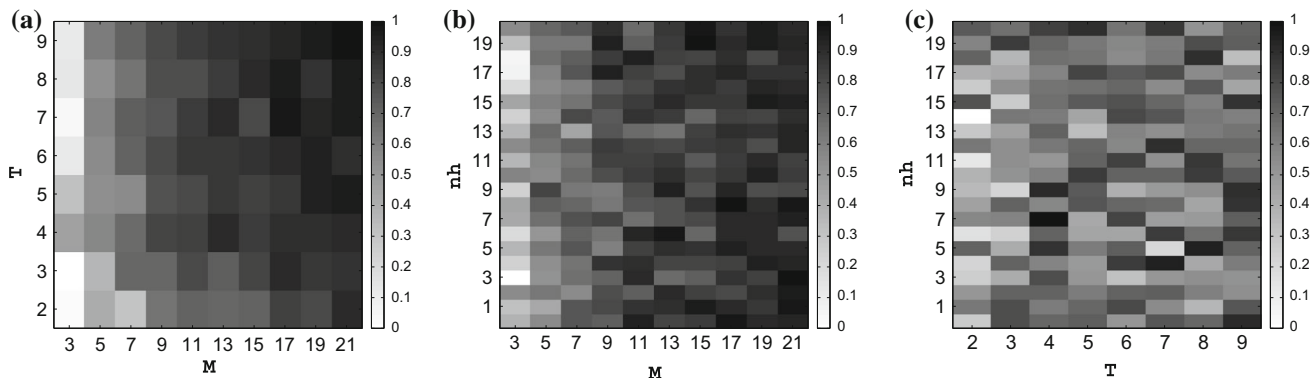

Fig. 2 Statlog data set heatmap. **a** Split size and adaboost T size, **b** Split size and number of nh, **c** Adaboost T size and number of nh
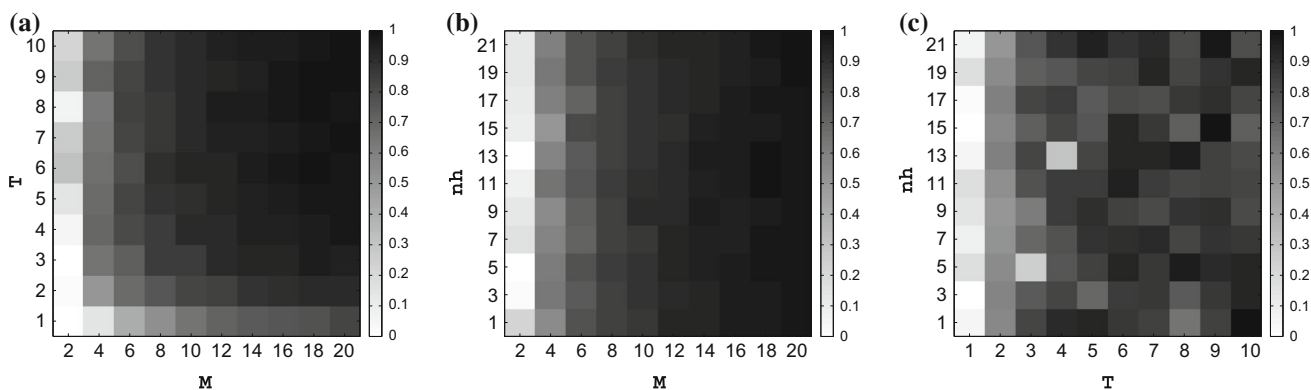

Fig. 3 Pendigit data set heatmap. **a** Split size and adaboost T size, **b** Split size and number of nh, **c** Adaboost T size and number of nh
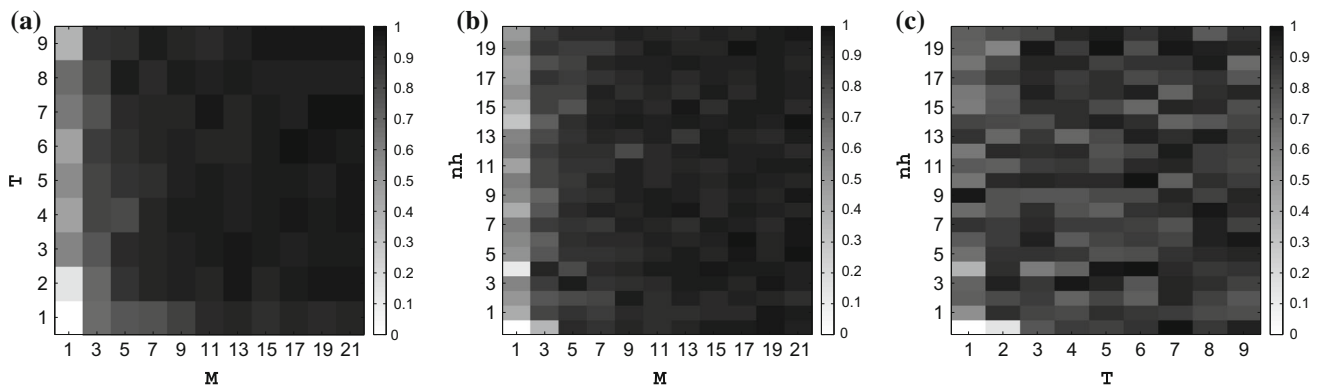
Fig. 4 Skin data set heatmap. **a** Split size and adaboost *T* size, **b** Split size and number of *nh*, **c** Adaboost *T* size and number of *nh*
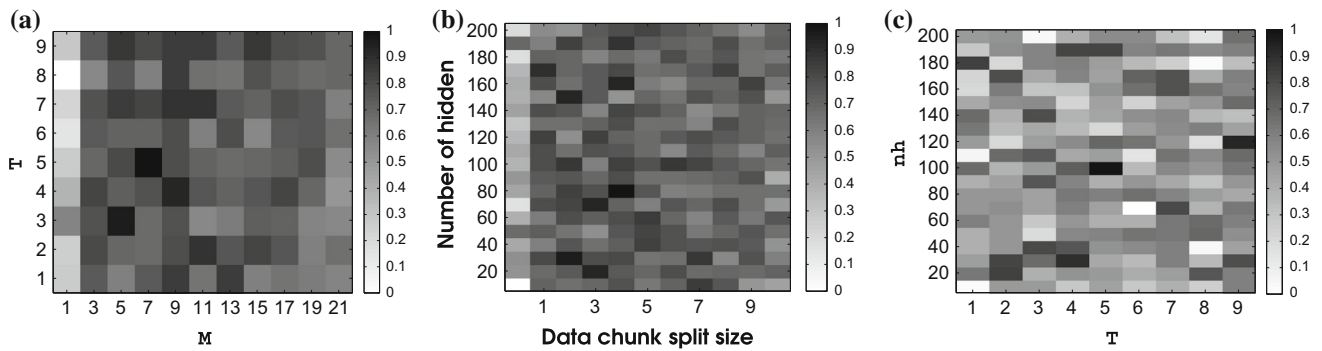

Fig. 5 Page blocks data set heatmap. **a** Split size and adaboost *T* size, **b** Split size and number of *nh*, **c** Adaboost *T* size and number of *nh*
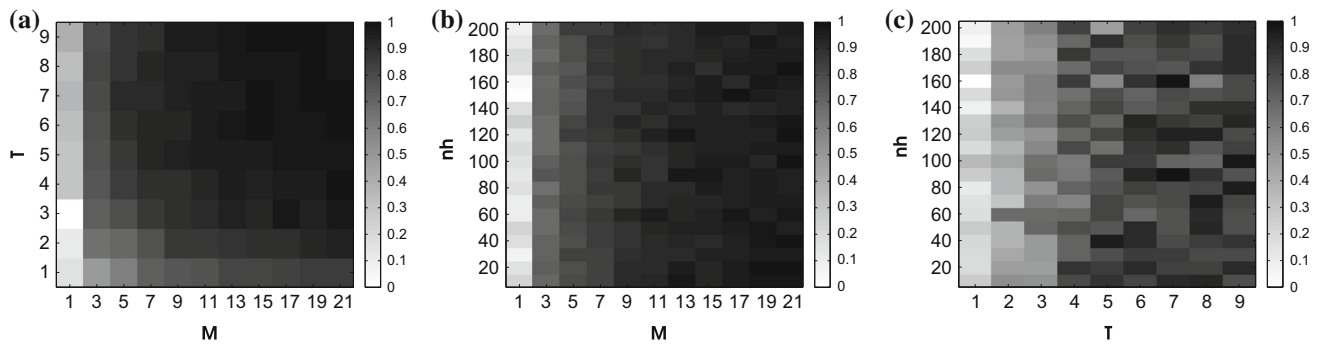

Fig. 6 Waveform data set heatmap. **a** Split size and adaboost *T* size, **b** Split size and number of *nh*, **c** Adaboost *T* size and number of *nh*

Table 7 Performance comparison of ELM and proposed model

| Data set	Method	F_1	Recall	Precision	Accuracy
Pendigit	Conventional	0.8404	0.8393	0.8416	0.8407
	Proposed	0.8301	0.8234	0.8369	0.8256
Skin	Conventional	0.9754	0.9956	0.9583	0.9894
	Proposed	0.9842	0.9913	0.9773	0.9892
Statlog	Conventional	0.8871	0.8556	0.9237	0.9757
	Proposed	0.6045	0.5069	0.7486	0.9103
Page-blocks	Conventional	0.9873	0.9764	0.9988	0.9977
	Proposed	0.7030	0.5756	0.9027	0.9404
Waveform	Conventional	0.8372	0.8368	0.8375	0.8376
	Proposed	0.8642	0.8605	0.8680	0.8620
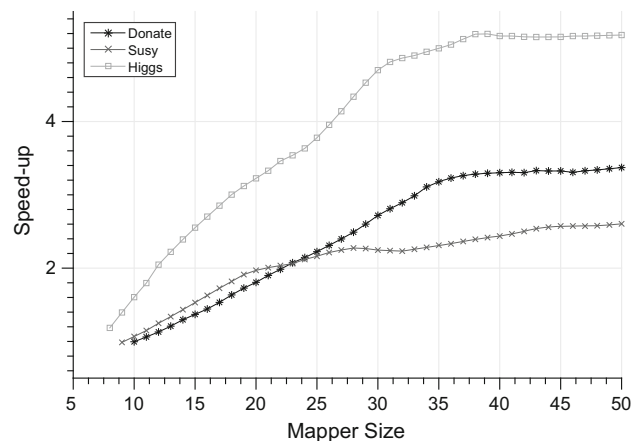
Fig. 7 Stability analysis of ensemble ELM classifiers with Mapper size
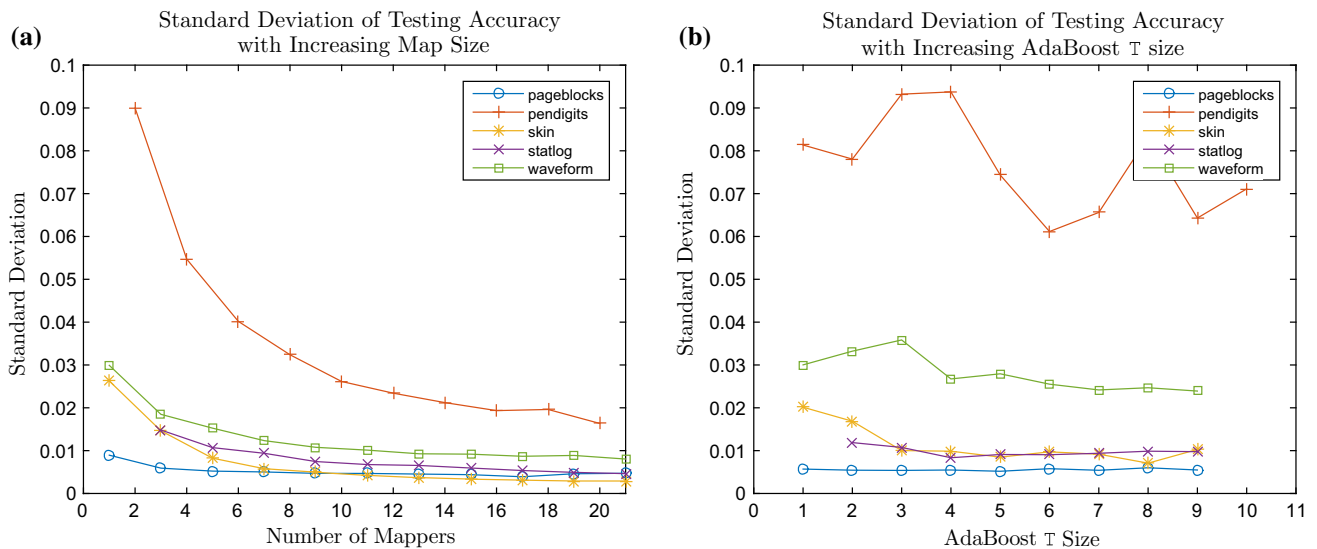
Fig. 8 Stability analysis. **a** Stability analysis of ensemble ELM classifiers with Mapper size, **b** Stability analysis of ensemble ELM classifiers with AdaBoost T size

5.5 Stability analysis

Standard deviation of testing accuracy of the method is shown in Fig. 8a, b. We analyzed the stability of ensemble ELM classifier with two aspects, Mapper size and AdaBoost T size. Mapper size is the most important variable for the model stability according to the Fig. 8a. From Fig. 8a, b, we can find that standard deviation of testing accuracy decreases enormously with the increasing of Mapper function size. Through this analysis, one can argue that a model with high Mapper function size has higher stability than low Mapper function size.

6 Conclusion and future works

In this paper, a parallel AdaBoost extreme learning machine algorithm implementation has been proposed for massive data learning. By creating the overall data set into data chunks, MapReduce-based learning algorithm reduces the training time of ELM classification. To overcome the accuracy performance decreasing, distributed ELM is enhanced with AdaBoost method. The experimental results show that AdaBoosted ELM reduces not only the training time of large-scale data sets, but also evaluation metrics of the accuracy performance as compared with the conventional ELM.

The proposed AdaBoost-based ELM has three different trade-off parameters which are (1) data chunk split size, M, (2) maximum number of iterations, T, in AdaBoost Algorithm and lastly (3) number of hidden layer nodes nh in ELM algorithm. The empirical results in heatmap figures show that parameters M and T are more dominant than parameter nh for the classification accuracy of the hypothesis.

The algorithm is designed to deal with large-scale data set ELM training problems. Another objective is to achieve the model's classification performance with same or close to the conventional ELM method. Classification performance results are shown in Sect. 5.3. The empirical results show us that classification performance results of the proposed method have almost the same values with the conventional ELM method.

References

Alimoglu F, Alpaydin E (1996) Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition. In: Proceedings of the fifth Turkish artificial intelligence and artificial neural networks symposium (TAINN 96)

Baldi P, Sadowski P, Whiteson D (2014) Searching for exotic particles in high-energy physics with deep learning. Nature Commun 5

Bartlett P (1998) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inf Theory 44:525–536

Bhatt R, Sharma G, Dhall A, Chaudhury S (2009) Efficient skin region segmentation using low complexity fuzzy decision tree model. In: 2009 Annual IEEE India Conference (INDICON), pp 1–4

Bhimji W, Bristow T, Washbrook A (2014) Hepdoop: high-energy physics analysis using hadoop. J Phys Conf Ser 513:022004 (IOP Publishing)

Bi X, Zhao X, Wang G, Zhang P, Wang C (2015) Distributed extreme learning machine with kernels based on mapreduce. Neurocomputing 149:456–463. Advances in neural networks selected papers from the tenth international symposium on neural networks (ISNN 2013) Advances in extreme learning machines selected articles from the international symposium on extreme learning machines (ELM 2013)

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC Press, Boca Raton

Catak F, Balaban M (2013) Cloudsvm: training an svm classifier in cloud computing systems. In: Zu Q, Hu B, Eli A (eds) Pervasive computing and the networked world, vol 7719 of Lecture Notes in Computer Science. Springer, Berlin Heidelberg, pp 57–68

Chen J, Zheng G, Chen H (2013) Elm-mapreduce: Mapreduce accelerated extreme learning machine for big spatial data analysis. In: 2013 10th IEEE International Conference on Control and Automation (ICCA), pp 400–405

Choi J, Choi C, Ko B, Kim P (2014) A method of ddos attack detection using http packet pattern and rule engine in cloud computing environment. Soft Comput 18(9):1697–1703

Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. Commun ACM 51:107–113

Freund Y, Schapire R, Abe N (1999) A short introduction to boosting. J Jpn Soc Artif Intell 14(771–780):1612

Freund Y, Schapire RE (1995) A desicion-theoretic generalization of on-line learning and an application to boosting. In: Computational learning theory. Springer, New York, pp 23–37

He Y, Tan H, Luo W, Mao H, Ma D, Feng S, Fan J (2011) Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce. In: 2011 IEEE 17th International conference on parallel and distributed systems (ICPADS), pp 473–480

Hsu C-W, Lin C-J (2002) A comparison of methods for multiclass support vector machines. Trans Neural Netw 13:415–425

Huang GB, Chen L (2006) Enhanced random search based incremental extreme learning machine. Neurocomputing 71(1618):3460–3468. Advances in neural information processing (ICONIP 2006)/brazilian symposium on neural networks (SBRN 2006)

Huang GB, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70(1618):3056–3062. Neural network applications in electrical engineering selected papers from the 3rd international work-conference on artificial neural networks (IWANN 2005)

Huang GB, Li MB, Chen L, Siew CK (2008) Incremental extreme learning machine with fully complex hidden nodes. Neurocomputing 71(46):576–583. Neural networks: algorithms and applications 4th international symposium on neural networks 50 years of artificial intelligence: a neuronal approach campus multidisciplinary in perception and intelligence

Huang GB, Zhu QY, Siew CK (2006a) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the international joint conference on neural networks, pp 985–990

Huang GB, Zhu QY, Siew CK (2006b) Extreme learning machine: theory and applications. Neurocomputing 70(13):489–501. Neural networks selected papers from the 7th Brazilian symposium on neural networks (SBRN '04)

Huang G-B, Chen L, Siew C-K (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17:879–892

Khomtchouk B, Van Booven D, Wahlestedt C (2014) Heatmapgenerator: high performance rnaseq and microarray visualization software suite to examine differential gene expression levels using an r and c++ hybrid computational pipeline. Source Code Biol Med 9(1)

Krogh A, Vedelsby J (1995) Neural network ensembles, cross validation, and active learning. Adv Neural Inf Process Syst 231–238 (MIT Press)

Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach Learn 51(2):181–207

Lan Y, Hu Z, Soh YC, Huang G-B (2013) An extreme learning machine approach for speaker recognition. Neural Comput Appl 22(3–4):417–425

Landesa-Vzquez I, Alba-Castro JL (2013) Double-base asymmetric adaboost. Neurocomputing 118:101–114

Liang N-Y, Huang G-B, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17:1411–1423

LIBSVM (2015) Libsvm data: classification, regression, and multi-label. http://ntucsu.csie.ntu.edu.tw/

Lu Y, Roychowdhury V, Vandenberghe L (2008) Distributed parallel support vector machines in strongly connected networks. IEEE Trans Neural Netw 19:1167–1178

Makhoul J, Kubala F, Schwartz R, Weischedel R (1999) Performance measures for information extraction. In: Proceedings of DARPA broadcast news workshop, pp 249–252

Malerba D, Esposito F, Semeraro G (1996) A further comparison of simplification methods for decision-tree induction. In Fisher D, Lenz H (eds) Learning. Springer, New York, pp 365–374

Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, New York

Ogiela M, Castiglione A, You I (2014) Soft computing for security services in smart and ubiquitous environments. Soft Comput 18(9):1655–1658

Panda B, Herbach JS, Basu S, Bayardo RJ (2009) Planet: massively parallel learning of tree ensembles with mapreduce. Proc VLDB Endow 2:1426–1437

Schatz MC (2009) CloudBurst: highly sensitive read mapping with MapReduce. Bioinformatics (Oxford, England) 25:1363–1369

Schmidtmann I, Hammer G, Sariyar M, Gerhold-Ay A, des öffentlichen Rechts K (2009) Evaluation des krebsregisters nrw–schwerpunkt record linkage. Abschlußbericht vom 11

Sun Z, Fox G (2012) Study on parallel svm based on mapreduce. In: International conference on parallel and distributed processing techniques and applications. Citeseer, pp 16–19

Sun T, Shu C, Li F, Yu H, Ma L, Fang Y (2009) An efficient hierarchical clustering method for large datasets with map-reduce. In: 2009 International conference on parallel and distributed computing, applications and technologies, pp 494–499

Sun Y, Yuan Y, Wang G (2011) An os-elm based distributed ensemble classification framework in P2P networks. Neurocomputing 74(16):2438–2443. Advances in extreme learning machine: theory and applications biological inspired systems. Computational and ambient intelligence selected papers of the 10th international work-conference on artificial neural networks (IWANN2009)

Tang J, Deng C, Huang G-B, Zhao B (2015) Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine. IEEE Trans Geosci Remote Sens 53:1174–1185

Turpin A, Scholer F (2006) User performance versus precision measures for simple search tasks. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06, (New York, NY, USA). ACM, pp 11–18

UCI (2011) Record linkage comparison patterns data set. https://archive.ics.uci.edu/ml/datasets/Record+Linkage+Comparison+Patterns

UCI (2014) Higgs data set. https://archive.ics.uci.edu/ml/datasets/HIGGS

UCI (2014) Susy data set. https://archive.ics.uci.edu/ml/datasets/SUSY

Wang B, Huang S, Qiu J, Liu Y, Wang G (2015) Parallel online sequential extreme learning machine based on mapreduce. Neurocomputing 149:224–232. Advances in neural networks selected papers from the 10th international symposium on neural networks (ISNN 2013) Advances in extreme learning machines selected articles from the international symposium on extreme Learning machines (ELM 2013)

Wang G, Zhao Y, Wang D (2008) A protein secondary structure prediction framework based on the extreme learning machine.

Neurocomputing 72(13):262–268. Machine learning for signal processing (MLSP 2006)/life system modelling, simulation, and bio-inspired computing (LSMS 2007)

Xin J, Wang Z, Chen C, Ding L, Wang G, Zhao Y (2014) Elm: distributed extreme learning machine with mapreduce. World Wide Web 17(5):1189–1204

Xu L, Kim H, Wang X, Shi W, Suh T (2014) Privacy preserving large scale dna read-mapping in mapreduce framework using fpgas. In: 2014 24th International conference on field programmable logic and applications (FPL). IEEE, pp 1–4

Zhang C, Li F, Jestes J (2012) Efficient parallel knn joins for large data in mapreduce. In: Proceedings of the 15th international conference on extending database technology, EDBT '12, (New York, NY, USA). ACM, pp 38–49

Zhao X-G, Wang G, Bi X, Gong P, Zhao Y (2011) Xml document classification based on elm. Neurocomputing 74(16):2444–2451

Zhao W, Ma H, He Q (2009) Parallel k-means clustering based on mapreduce. In: Jaatun M, Zhao G, Rong C (eds) Cloud computing, vol 5931 of Lecture Notes in Computer Science. Springer, Berlin Heidelberg, pp 674–679

Zong W, Huang GB (2011) Face recognition based on extreme learning machine. Neurocomputing 74(16):2541–2551. Advances in extreme learning machine: theory and applications biological inspired systems. Computational and ambient intelligence selected papers of the 10th international work-conference on artificial neural networks (IWANN2009)