

Extreme learning machines' ensemble selection with GRASP

Ting Zhang · Qun Dai · Zhongchen Ma

Published online: 17 March 2015
© Springer Science+Business Media New York 2015

Abstract Credit scoring, which is also called credit risk assessment has attracted the attention of many financial institutions and much research has been carried out. In this work, a new Extreme Learning Machines' (ELMs) Ensemble Selection algorithm based on the Greedy Randomized Adaptive Search Procedure (GRASP), referred to as ELMsGraspEnS, is proposed for credit risk assessment of enterprises. On the one hand, the ELM is used as the base learner for ELMsGraspEnS owing to its significant advantages including an extremely fast learning speed, good generalization performance, and avoidance of issues like local minima and overfitting. On the other hand, to ameliorate the local optima problem faced by classical greedy ensemble selection methods, we incorporated GRASP, a meta-heuristic multi-start algorithm for combinatorial optimization problems, into the solution of ensemble selection, and proposed an ensemble selection algorithm based on GRASP (GraspEnS) in our previous work. The GraspEnS algorithm has the following three advantages. (1) By incorporating a random factor, a solution is often able to escape local optima. (2) GraspEnS realizes a multi-start search to some degree. (3) A better performing subensemble can usually be found with GraspEnS. Moreover, not much research on applying ensemble selection approaches to credit scoring has been reported in the literature. In this paper, we integrate the ELM with GraspEnS, and propose a novel ensemble selection algorithm based on GRASP (ELMsGraspEnS). ELMsGraspEnS naturally inherits the inherent advantages of both the ELM and GraspEnS, effectively

combining their advantages. The experimental results of applying ELMsGraspEnS to three benchmark real world credit datasets show that in most cases ELMsGraspEnS significantly improves the performance of credit risk assessment compared with several state-of-the-art algorithms. Thus, it can be concluded that ELMsGraspEnS simultaneously exhibits relatively high efficiency and effectiveness.

Keywords Credit risk assessment · Ensemble selection · Extreme learning machine (ELM) · Greedy randomized adaptive search procedure (GRASP) · Ensemble selection algorithm based on GRASP (GraspEnS) · ELMs' ensemble selection algorithm with GRASP (ELMsGraspEnS)

1 Introduction

Credit risk assessment has become an increasingly important research topic for financial institutions in recent years. With the increasing number of customers who default on loans, many financial institutions have suffered serious losses over several years. However, it is not reasonable for financial institutions to turn down all applications to avoid any credit risk. For this reason, proper classification of applicants is crucial [1–3]; that is, applicants with good credit scores and those with bad ones. Financial institutions can grant credit to applicants with good credit scores and reject applicants with bad ones. Thus, the accuracy of credit scoring is very important. Even a small improvement in the accuracy of credit risk assessment can greatly reduce financial institution losses.

Recently, many studies on credit risk assessment have been carried out classified mainly into two categories: statistical and intelligent techniques. Techniques such as linear

T. Zhang · Q. Dai (✉) · Z. Ma
College of Computer Science and Technology, Nanjing University
of Aeronautics and Astronautics, Nanjing 210016, China
e-mail: daiqun@nuaa.edu.cn

discriminant analysis [4, 5], logistic regression analysis [6, 7], and multivariate adaptive regression splines [8], amongst others, are classified as statistical techniques. However, when applying statistical techniques to credit scoring, a problem arises in that some assumptions, for example, the multivariate normality assumptions for independent variables, are frequently violated in the practice of credit scoring making these techniques theoretically ineffective for finite samples [9]. Intelligent techniques, like artificial neural networks (ANNs) [6, 10] decision trees (DTs) [11, 12], case-based reasoning [13, 14], and support vector machines (SVMs) [15–17], can be used as alternative approaches for credit scoring. In contrast to statistical techniques, intelligent techniques do not need to assume certain data distributions. These techniques voluntarily extract knowledge from training instances. According to previous research, intelligent techniques perform better than statistical techniques in credit scoring tasks, especially for nonlinear pattern classifications [9].

Each of the two categories has its own characteristics and the same approach can have significantly different performance on different credit datasets. Therefore, we cannot state conclusively which technique is the best. Recent research has focused on combinations of multiple classifiers, i.e., ensemble learning. Moreover, many experiments have shown that ensemble learning is an effective strategy for achieving high classification performance, especially with a number of varying structures in the base models, which make the prediction error independent [18–22]. For comparison with multiple classifiers and varying diversity, the performance of single classifiers was investigated by Tsai and Wu [23], with a neural network used as the base learner. Experimental results indicate that the ensemble method is better than single classifiers. Yu, Wang and Lai [24] proposed a method using a combination of multiple neural network models to predict credit risk. From a comparison with single classifiers, they conclude that the ensemble learning model achieves a more promising solution. In [25], Nanni and Lumini carried out an experiment to investigate the performance of several systems based on ensemble learning. The results show that ensemble learning methods can improve the performance of single classifiers for credit risk analysis. Hung and Chen [26] proposed a selective ensemble based on the expected probabilities method for credit risk assessment. This method uses three classifiers: a DT, ANN and SVM. The benefits of the proposed ensemble method are that the advantages and disadvantages of different classification methods are inherited and avoided, respectively. However, the operating principle of ensemble learning is based on multiple classifiers, which requires more computational time and storage space. Moreover, classifiers with both high and low predictive performances are included in the ensemble, and therefore

the prediction results of the entire ensemble are negatively affected by classifiers with low performance.

To overcome the existing limitations of ensemble learning, the paradigm of ensemble selection has been proposed, which is also referred to as ensemble pruning, ensemble thinning, or selective ensemble [27]. The purpose of ensemble selection is to select a proper subset from the entire set of original base classifiers. High efficiency and better classification performance are the two most important advantages of ensemble selection. On the one hand, owing to a reduction in ensemble size, execution time is shortened, while at the same time saving storage space. On the other hand, the selected ensemble performs better than the original ensemble in terms of classification, as verified by many previous studies [27–34].

Thus far, very little research has been conducted on applying ensemble selection approaches to credit scoring. Nevertheless, it is expected that the performance of credit risk assessment using the ensemble selection approach will be better than that of its competitors. Since credit risk assessment could be regarded as a special case of the classification problem, according to the description of Zhou et al. in [35], it can be expected that, a properly selected subensemble will perform better than the original ensemble in classification tasks, and more specifically, in credit scoring tasks.

Broadly speaking, ensemble selection can be considered as a combinatorial optimization problem. In recent years, various ensemble selection approaches based on a greedy searching strategy have been proposed [27, 30, 32, 36, 37]. There are three main elements in greedy ensemble selection approaches: the selection direction, the evaluation dataset, and the evaluation measure. In previous research on the greedy ensemble selection approach, researchers have focused on the design of better evaluation measures, while the serious problem of local optima caused by improper starting points or searching methods is often ignored.

Motivated by the above ideas, in this work, we propose a novel ensemble selection algorithm for credit risk assessment, called ELMsGraspEnS, which is based on the extreme learning machine (ELM) and greedy randomized adaptive search procedure (GRASP). Generally, in ELMsGraspEnS, the ELM is used as the base model in the original ensemble, and ensemble selection is implemented using the GRASP algorithm.

The ELM is an efficient learning approach for single hidden layer feedforward neural networks (SLFNs) [38]. Unlike the extensively applied conventional feedforward neural network learning algorithm, i.e., the gradient-based learning algorithm, the ELM can randomly initialize the SLFN weights and bias, while its input weights and biases do not need to be adjusted. With the ELM, it is possible to determine analytically the hidden layer output matrix and

the output weights of the SLFNs. The network is obtained with very few computational steps and very low computational cost. Owing to this feature, multiple ELMs can very easily be produced to build an ensemble. In addition, the learning speed of the ELM can be thousands of times faster than the traditional gradient-based learning algorithm, while its generalization performance is better than that of the latter algorithm in quite a few classification applications. Moreover, there are several issues with the traditional gradient-based learning algorithm, for example, local minima, improper learning rate, and overfitting. The ELM can avoid these problems efficiently [38].

Many applications of the ELM algorithm have been proposed in the literature. In [39], the ELM is applied to solve multi-category classification problems using a single ELM classifier and a series of ELM binary classifiers, this work provides a promising solution. Duan, Huang and Wang [40] proposed an ELM method for classification of bank clients, and compared the performance of the ELM with other existing learning methods, i.e., DT, NN and SVM. Their results show that the ELM method is superior to the other three methods. In [41], Huang et al. proved that for typical regression and multi-class classification problems, the ELM outperforms the traditional SVM method. The ELM is used as a classifier for a two-category data classification problem in [42], with the experimental results showing that the ELM method achieves better prediction performance than traditional methods in reduced time. In [43], using a hybrid grouping harmony search and the ELM approach to evaluate the internationalization success of a company, the authors clearly show that the performance of the ELM algorithm is better than other typical classifiers.

In [31], we proposed a novel ensemble selection algorithm based on GRASP (GraspEnS). GRASP is a meta-heuristic multi-start algorithm for combinatorial optimization problems introduced by Feo and Resende [44]. It is an iterative procedure [45, 46], where each iterative process includes a construction phase and a local search phase. A feasible solution is generated in the construction phase, and is refined in the local search phase. The two phases are iterated a number of times until the termination criterion is reached, that is, either the maximum number of iterations is reached or the target value of the objective function is satisfied. At the end of the procedure, the final solution output is the best solution obtained from the entire iterative process [44, 45]. GRASP has been applied widely in scheduling, routing, logic, partitioning, location and layout, amongst others. Our proposed GraspEnS is an application of GRASP to the problem of ensemble selection.

The GraspEnS algorithm has some similar characteristics to the classical greedy ensemble selection method. However, it differs significantly from the latter method. GraspEnS has four main advantages [31]. (1) The algorithm

can escape from local optima by incorporating random factors in the construction phase. (2) In the local search phase, each feasible solution obtained in the previous construction phase is regarded as a starting point, thereby allowing the GraspEnS algorithm to realize a multi-start search. (3) By searching the neighbors of a feasible solution constructed in the first phase, the solution is further improved, because it is iteratively replaced by a better solution; that is the neighbor of the current solution, with the process continuing until no better solution is found. (4) The generalization performance of GraspEnS is superior to that of the classical greedy ensemble selection approach, i.e., directed hill climbing ensemble pruning (DHCEP) algorithm.

This paper combines the ELM and GraspEnS, and proposes a novel ELMsGraspEnS algorithm, which fully uses not only the convenience of calculation of the ELM, but also the superior generalization performance of GraspEnS. Moreover, ELMsGraspEnS has been successfully applied to credit risk assessment and achieves very good performance. Generally speaking, it can be said that ELMsGraspEnS simultaneously exhibits relatively high efficiency and effectiveness. The contributions of this paper are as follows:

First, the ELMsGraspEnS algorithm intuitively inherits the inherent characteristics of the ELM, such as its fast learning speed, better generalization performance, avoidance of overfitting and local minima, and easy implementation.

Second, ensemble systems can usually be constructed using homogeneous or heterogeneous models. ELMsGraspEnS is constructed with multiple homogeneous ELMs. The ELM randomly sets the initial weights and bias of an SLFN, incorporating random factors into ELMsGraspEnS. Moreover, the SLFN is easy to construct.

Third, using ELMsGraspEnS, the ensemble selection technique is applied to credit scoring. At present, there is very little research on the application of ensemble selection techniques to credit risk assessment. By using the ensemble selection technique in our proposed ELMsGraspEnS, we can improve the performance of credit risk assessment.

Fourth, with ELMsGraspEnS our previously proposed GraspEnS is used as the ensemble selection approach. ELMsGraspEnS makes full use of the advantages of the GraspEnS method. Therefore, the proposed ELMsGraspEnS algorithm not only can escape local optima, but also realizes a multi-start search, making it superior to the classical greedy ensemble selection approach in generalization performance.

Last, to verify the efficacy of the ELMsGraspEnS algorithm, we carried out experiments to compare the credit risk assessment performance of the ELMsGraspEnS algorithm with several state-of-the-art ensemble selection approaches on three benchmark credit datasets. We also compared ELMsGraspEnS with an SVM and multi-layer perceptron

(MLP) to illustrate its efficiency. The experimental results indicate that our proposed ELMsGraspEnS algorithm significantly outperforms its competitors on all three datasets.

The remainder of the paper is organized as follows. Section 2 introduces some background knowledge of the ELM. Section 3 briefly reviews the ensemble methods and the classical greedy ensemble selection methods. Section 4 first introduces the GraspEnS algorithm based on GRASP, and then presents the proposed ELMsGraspEnS algorithm. Section 5 presents and discusses the experimental results. Finally, Section 6 summarizes the paper and suggests directions for future work.

2 Background knowledge of the ELM

Guang-Bin Huang et al. [38] originally proposed the ELM, which is an effective and efficient learning algorithm for SLFNs. The outstanding feature of the ELM is that the algorithm randomly initializes the input weights and hidden layer bias of an SLFN. Furthermore, in the ELM, the input weights and hidden layer bias do not need to be tuned. It is easy to calculate the hidden layer output matrix and the output weights explicitly once the random values have been assigned to these two groups of parameters.

Suppose there are N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbf{R}^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in \mathbf{R}^m$. Then, a standard SLFN with \tilde{N} hidden nodes and an activation function $g(\mathbf{x})$ can be modeled as:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, j = 1, \dots, N, \tag{1}$$

These N equations can be written compactly as:

$$\mathbf{H}\beta = \mathbf{T}, \tag{2}$$

where

$$\begin{aligned} & \mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}, \mathbf{x}_1, \dots, \mathbf{x}_N) \\ &= \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \dots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \dots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \\ & \beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m}, \text{ and } \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m}. \end{aligned}$$

Here, \mathbf{w}_i is the weight vector connecting the i -th hidden node and the input nodes, β_i is the weight vector connecting the i -th hidden node and the output nodes, b_i is the bias of the i -th hidden node, and $\mathbf{w}_i \cdot \mathbf{x}_j$ represents the inner

product of \mathbf{w}_i and \mathbf{x}_j . As named in Huang et al. [47, 48], \mathbf{H} is called the hidden layer output matrix of the neural network; the i -th column of \mathbf{H} is the i -th hidden node output with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$.

In traditional feedforward neural network learning methods, one may wish to find specific $\hat{\mathbf{w}}_i, b_i, \hat{\beta}_i (i = 1, \dots, \tilde{N})$ to minimize $\|\mathbf{H}\beta - \mathbf{T}\|$. Specifically,

$$\begin{aligned} & \|\mathbf{H}(\hat{\mathbf{w}}_1, \dots, \hat{\mathbf{w}}_{\tilde{N}}, \hat{b}_1, \dots, \hat{b}_{\tilde{N}})\hat{\beta} - \mathbf{T}\| \\ &= \min_{\mathbf{w}_i, b_i, \beta} \|\mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}})\beta - \mathbf{T}\|, \end{aligned} \tag{3}$$

which is equivalent to minimizing the cost function

$$Q = \sum_{j=1}^N \left(\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) - t_j \right)^2. \tag{4}$$

Vector \mathbf{W} , consisting of a set of weights (\mathbf{w}_i, β_i) and bias (b_i) parameters, is updated using gradient-based algorithms in the above minimization procedure as:

$$\mathbf{W}_k = \mathbf{W}_{k-1} - \eta \frac{\partial Q(\mathbf{W})}{\partial \mathbf{W}}, \tag{5}$$

where η is the learning rate. The backpropagation (BP) algorithm is a powerful supervised learning algorithm for training feedforward neural networks (FNNs). However, there are many shortcomings with the BP algorithm [38]. First, the performance of BP is easily affected by local optima. Second, there exists the problem of overfitting in BP, which can very easily result in very poor prediction performance. Third, BP is a time-consuming algorithm in many cases, and finally, it is difficult to determine the value of η .

These problems of the BP learning algorithm can be resolved by the ELM. Since the input weights and hidden layer bias do not need to be tuned, the hidden layer output matrix \mathbf{H} can be calculated once random values have been assigned to them. Then, (3) is equivalent to finding a least-squares solution $\hat{\beta}$ of the linear system $\mathbf{H}\beta = \mathbf{T}$:

$$\begin{aligned} & \|\mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}})\hat{\beta} - \mathbf{T}\| \\ &= \min_{\beta} \|\mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}})\beta - \mathbf{T}\|. \end{aligned} \tag{6}$$

If the number \tilde{N} of hidden nodes is equal to the number N of distinct training samples, $\tilde{N} = N$, matrix \mathbf{H} is square and invertible if the input weight vectors \mathbf{w}_i and the hidden biases b_i are randomly chosen, and these training samples can be approximated with zero error. However, in most cases the number of hidden nodes is much smaller than the number of distinct training samples, i.e., $\tilde{N} \ll N$. Then, \mathbf{H} is a nonsquare matrix and there may not exist $\mathbf{w}_i, b_i, \beta (i = 1, \dots, \tilde{N})$ such that $\mathbf{H}\beta = \mathbf{T}$. The ELM algorithm learns the output weights β using a Moore–Penrose generalized inverse of matrix \mathbf{H} , denoted as \mathbf{H}^+ [38]. Then, the smallest norm least squares solution of the

above linear system is:

$$\hat{\beta} = \mathbf{H}^+\mathbf{T}. \tag{7}$$

The solution $\hat{\beta}$ defined in (7) has the minimum norm over all the least squares solutions of the linear system defined in (2). Therefore, $\hat{\beta}$ has the best generalization performance over all the other least squares solutions [36].

Thus, the ELM algorithm can be described as follows:

Algorithm 1 Extreme Learning Machine

Input:

Tr – the training set,

$$Tr = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}] \in \mathbf{R}^n,$$

$$\mathbf{t}_i = [t_{i1}, t_{i2}, \dots, t_{im}] \in \mathbf{R}^m, i = 1, \dots, N\}$$

$g(x)$ – activation function

\tilde{N} – number of hidden nodes

Output: \mathbf{H} – hidden layer output matrix
 β – output weight

1. Randomly assign input weight \mathbf{w}_i and bias b_i ,
 $i = 1, \dots, \tilde{N}$.
 2. Calculate the hidden layer output matrix \mathbf{H} .
 3. Calculate the output weight β , $\beta = \mathbf{H}^+\mathbf{T}$,
 where $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_N]^T$.
 4. Return \mathbf{H} , β .
-

Compared with the traditional gradient-based learning algorithms for SLFNs, the ELM has several favorable and significant features: the learning speed of the ELM is extremely fast, the generalization performance of the ELM is superior to that of the gradient-based learning algorithms, and the ELM learning algorithm tends to yield solutions straightforwardly without facing several issues like local minima or overfitting, as is the case with the traditional gradient-based learning algorithms [38].

From Algorithm 1, we can see that the size of the hidden node layer \tilde{N} must first be set. From the discussion above, we know that the required number of hidden nodes \tilde{N} cannot exceed the number of distinct training samples. However, it is unreasonable to set too small a number of hidden nodes. Selection of the appropriate number of hidden nodes is discussed further in Section 5.

3 Ensemble learning methods and classical greedy ensemble selection algorithms

3.1 Ensemble learning methods

There are two important phases in ensemble learning methods: generation of multiple predictive models and the combination of these.

3.1.1 Producing the models

In the production of multiple predictive models for an ensemble, the ensemble can be composed of either homogeneous or heterogeneous models. Homogeneous models can be obtained by running the same learning algorithm in different ways. To ensure diversity of the constituent models, many approaches have been followed, including using different parameter values for the learning algorithm, incorporating random factors into the learning algorithm, or manipulating the input attributes and outputs of the models [49]. Bagging [50] and boosting [51] are the two best known classical approaches for generating ensembles with homogeneous constituent models.

Heterogeneous models are generated by applying different learning algorithms to the same dataset. Because the prediction performance of different learning algorithms can vary, diversity among heterogeneous constituent models is relatively easy to ensure.

3.1.2 Combining the models

Various approaches have been employed to combine classification models. The simplest and most frequently used approach is unweighted majority voting. Both homogeneous and heterogeneous models can be combined using this approach. For unweighted majority voting, each model outputs a class value (or ranking or probability distribution), and the final result is the class with the most votes (or the highest average ranking or largest average probability) [27].

Let \mathbf{x} represent an instance; then, the following mathematical equation expresses the output of unweighted majority voting method $F_E(\mathbf{x})$ for instance \mathbf{x} from the original ensemble $E = \{h_i(\mathbf{x})\}_i^k$, where k is the size of the original ensemble:

$$F_E(\mathbf{x}) = \arg \max_y \sum_{i=1}^k I(h_i(\mathbf{x}) = y), y \in Y, \tag{8}$$

where $I(\cdot)$ is an indicator function ($I(true) = 1, I(false) = 0$), $h_i(\mathbf{x})$ denotes the classification decision of the i -th model, whose value is a class label, and $Y = \{1, 2, \dots, C\}$ is the set of all possible class labels.

3.2 Classical greedy ensemble selection methods

Greedy ensemble selection methods greedily select the next state, which is the neighborhood of the current state, to visit. States refer to the different subsets of the initial ensemble $E = \{h_i(\mathbf{x})\}_i^k$, where k is the size of the original ensemble. The neighborhood of a subset of models $S \subseteq E$ contains those subsets that can be constructed by expanding one element into S or deleting one element from S [27]. As for

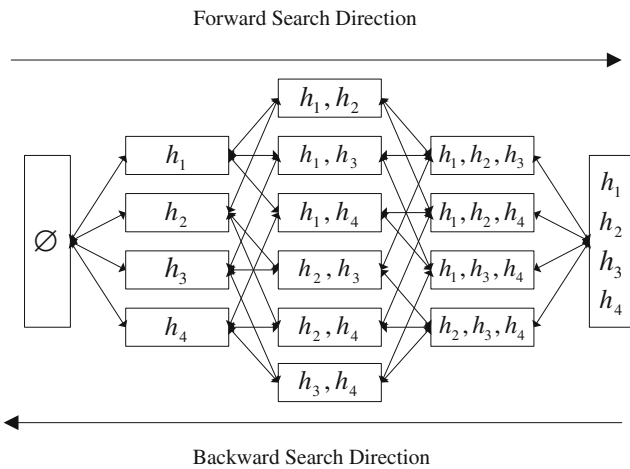


Fig. 1 Search space map for the greedy ensemble selection algorithm with an ensemble of four models

classical greedy ensemble selection methods, the selection direction, evaluation dataset, and evaluation measure are the three main factors.

3.2.1 Search direction

As depicted in Fig. 1, there are two main search directions for classical greedy ensemble selection strategies: forward and backward searching [27].

In forward searching, subset S is initialized as an empty set. The algorithm proceeds by iteratively adding into S model $h_i \in E \setminus S$ (where “ \setminus ” denotes subtraction of the two sets) based on a specific evaluation measure [27]. However, in a backward search, subset S is initialized as the entire original set, and the algorithm proceeds by repeatedly deleting from S model $h_i \in S$ according to a specific evaluation measure [27].

For both search directions, $\frac{k(k+1)}{2}$ subsets must be evaluated by the greedy ensemble selection method, with time complexity of $O(k^2 f(k, N_{pr}))$, where $f(k, N_{pr})$ represents the computational complexity of the evaluation function [25], and N_{pr} denotes the size of the pruning dataset.

3.2.2 Evaluation dataset

The evaluation dataset can refer to the pruning dataset, which can be generated by the training set [32, 37], a separate validation set [36], or even an artificial production. The advantage of using the training set for evaluation is that there is sufficient data for training and evaluation. However, this approach can easily suffer from overfitting. In contrast, using a separate validation set as the pruning dataset is less prone to overfitting; however, using a separate pruning set significantly reduces the amount of training data. A better approach based on k -fold cross-validation was proposed

in [33]. In each iteration, one fold is used as the pruning dataset, while the remaining folds are used as the training dataset. Finally, the evaluations are averaged across all folds. This approach is less susceptible to overfitting, and at the same time, makes full use of the training data.

3.2.3 Evaluation measures

Evaluation measures, which play an important role in the classical greedy ensemble selection method, can be roughly classified into two major categories: performance-based and diversity-based measures. The design and implementation of evaluation measures are based on the prediction accuracy of classifiers on the pruning dataset, defined as $Pr = \{(\mathbf{x}_i, y_i), i = 1, 2, \dots, N_{pr}\}$, where \mathbf{x}_i represents a feature vector, y_i denotes an unknown target value, and N_{pr} is the size of the pruning dataset.

Performance-based evaluation measures Many effective performance-based evaluation measures have been proposed, for example, accuracy (ACC), receiver operating characteristic (ROC) area, and average precision [27]. All these measures have the same intention, that is, to find a classifier that optimizes the performance of the ensemble generated by expanding (deleting) a candidate element into (from) the current ensemble. Their calculation is affected by the strategy for ensemble combination. The prediction result of the ensemble on the entire pruning dataset is required to calculate performance-based measures.

Among the performance-based evaluation measures, ACC is introduced in detail below. The ACC of model h with respect to ensemble S and pruning set Pr is defined as:

$$ACC_{Pr}(h, S) = \sum_{i=1}^{N_{pr}} I(F_{S \cup h}(\mathbf{x}_i) = y_i), y_i \in Y \tag{9}$$

where N_{pr} is the size of the pruning dataset. $F_{S \cup h}(\mathbf{x}_i)$ represents the result of unweighted majority voting for instance x_i from a new subensemble, generated by adding candidate classifier h into the current subensemble S , while the definition of function $F_E(\mathbf{x})$ has been given out in (8). The ACC measure prefers to select classifiers with the best accuracy at each step [31].

Diversity-based evaluation measures To achieve high generalization performance, great importance must be attached to the diversity among classifiers in an ensemble. Many well-known diversity-based measures have been proposed, including concurrency (CON), complementariness (COM), margin distance minimization (MAR), and uncertainty weighted accuracy (UWA) [27]. The similarities and differences between COM, CON, and UWA are analyzed below.

Before discussing these three measures in detail, it is necessary to present some additional notations. First, we must be able to distinguish the four events in the prediction results of candidate model h and subensemble S with respect to an instance (\mathbf{x}_i, y_i) [27].

$$\begin{aligned}
 e_{00}(h, S, \mathbf{x}_i, y_i) &: h(\mathbf{x}_i) \neq y_i \wedge S(\mathbf{x}_i) \neq y_i \\
 e_{01}(h, S, \mathbf{x}_i, y_i) &: h(\mathbf{x}_i) \neq y_i \wedge S(\mathbf{x}_i) = y_i \\
 e_{10}(h, S, \mathbf{x}_i, y_i) &: h(\mathbf{x}_i) = y_i \wedge S(\mathbf{x}_i) \neq y_i \\
 e_{11}(h, S, \mathbf{x}_i, y_i) &: h(\mathbf{x}_i) = y_i \wedge S(\mathbf{x}_i) = y_i
 \end{aligned} \tag{10}$$

The measures of COM, CON, and UWA are defined, respectively, as follows [27]:

$$COM(h, S) = \sum_{(\mathbf{x}_i, y_i) \in Pr} I(e_{10}(h, S, \mathbf{x}_i, y_i)), \tag{11}$$

$$\begin{aligned}
 CON(h, S) &= \sum_{(\mathbf{x}_i, y_i) \in Pr} (2I(e_{10}(h, S, \mathbf{x}_i, y_i)) \\
 &\quad + I(e_{11}(h, S, \mathbf{x}_i, y_i)) \\
 &\quad - 2I(e_{00}(h, S, \mathbf{x}_i, y_i))),
 \end{aligned} \tag{12}$$

$$\begin{aligned}
 UWA(h, S) &= \sum_{(\mathbf{x}_i, y_i) \in Pr} (I(e_{10}(h, S, \mathbf{x}_i, y_i))NT_i \\
 &\quad - I(e_{01}(h, S, \mathbf{x}_i, y_i))NF_i)
 \end{aligned} \tag{13}$$

and

$$+I(e_{11}(h, S, \mathbf{x}_i, y_i))NF_i - I(e_{00}(h, S, \mathbf{x}_i, y_i))NT_i)$$

where NT_i represents the proportion of member nets in the current subensemble S that correctly classify instance (\mathbf{x}_i, y_i) , $NF_i = 1 - NT_i$ represents the proportion of nets in S that misclassify the instance, and $I(\cdot)$ is an indicator function ($I(true) = 1$ and $I(false) = 0$).

As shown in their definitions, the evaluations of the three measures all depend on the decision of the current subensemble S and the candidate net h . COM focuses on an individual model that can help the current subset that obtains an incorrect result to evaluate the instance correctly. CON is similar to COM, but differs in that two extra events with their corresponding weights are added to CON. The UWA measure ameliorates the drawback of CON and COM by taking into account the uncertainty of the ensemble's decision. A series of experiments were conducted by Partalas et al. [27, 34], the results of which indicate that UWA achieves better performance under the same condition.

3.2.4 Amount of pruning

The final, but by no means the least significant factor of the classical greedy ensemble selection algorithm is the amount of pruning. Once the other factors have been determined, the

prediction performance of the selected ensemble is affected by the amount of pruning. In most cases, the error rate of the selected ensemble decreases with a decrease in its size at the beginning. However, after the minimum error rate has been obtained, the smaller the final amount of pruning is, the larger the error rate becomes.

The DHCEP algorithm, a popular greedy ensemble selection method proposed in previous research [27], automatically determines the pruning rate. In our work, DHCEP represents a general greedy ensemble selection approach, and is compared with the proposed ELMsGraspEnS algorithm, because of its automatic determination of pruning rate.

4 ELMs' ensemble selection with GRASP for credit scoring

Credit scoring has become a hot topic in financial institutions. Both intelligent and statistical techniques have been developed to solve this problem. However, no conclusion has been reached on which type of technique is the best. Recent studies have tended to combine multiple classifiers, known as ensemble learning, to predict the credit score of an applicant. Many experimental results indicate that ensemble learning has better generalization performance. However, to obtain better generalization performance, quite a few constituent classifiers are needed for the ensemble, resulting in fairly large execution time and storage space requirements. In addition, the performance of the entire ensemble is affected by classifiers with low generalization performance. Therefore, it is necessary to select an appropriate subset of the original ensemble. In this work, we propose ELMs-GraspEnS, which uses the ELM as the base learner for the ensemble. Overall, the novelty of the ELMsGraspEnS algorithm can be summarized as the application of the GraspEnS algorithm to search for a good subset of ELM ensemble classifiers.

In this section, we first discuss the GraspEnS algorithm in detail, and thereafter, the proposed ELMsGraspEnS algorithm for credit risk assessment is presented.

4.1 Ensemble selection algorithm based on GRASP

In one of our previously published works, we proposed the GraspEnS algorithm [31]. GRASP is an iterative two-phased metaheuristic procedure consisting of a greedy randomized construction phase and a local search phase. These two phases are repeated until the termination criterion is reached, for example, where the maximum number of iterations or the target value of an objective function is satisfied. The final solution is the best solution obtained from all the iterations [52].

Generally, an appropriate randomized greedy solution is obtained from the greedy randomized construction phase. Specifically, in each iteration of the construction phase, a restricted candidate list (RCL) is generated. Its size and elements are specified, respectively, by the greedy function and an invariable parameter α , where $\alpha \in [0, 1]$. Parameter α is important in GRASP, because it determines the greediness versus the randomness of GRASP. If $\alpha = 0$, the construction phase evolves into a greedy algorithm, whereas if $\alpha = 1$, the construction phase reduces to a random construction. An element is randomly selected from the RCL, and is then expanded into the current partial solution. Thereafter, the RCL is updated. The process terminates when a feasible solution of the required size is obtained.

In the local search phase, the solution obtained from the construction phase is refined. To improve the current solution, the neighbors thereof are searched. If a better neighbor is found, the current solution is replaced by this neighbor. The process is repeated a given number of times until no better solution is found.

Our previously proposed GraspEnS algorithm inherits the peculiarities of the GRASP framework, which improves the classical greedy ensemble selection approaches by incorporating random factors [31]. The classical greedy ensemble selection approaches are sensitive to the problem of local optima. However, GraspEnS allows a feasible solution to escape the local optima. The results of different iterations differ from one another owing to the incorporation of random factors. Because each feasible solution can be considered a new starting point, the GraspEnS algorithm effectively realizes a multi-start search. Moreover, in the local search phase, the feasible solution constructed in the construction phase is refined by searching its neighbors[31].

Algorithm 2 Ensemble Selection algorithm by GRASP

Input: α — an invariable parameter
 Pr— the pruning dataset, $\text{Pr} = \{(x_i, y_i), i = 1, 2, \dots, N_{pr}\}$
 E — the original ensemble, $E = \{h_i(x)\}_i^k$
 M — the required number of feasible solutions
 $Max_iterations$ — maximum number of iterations

Output: *BestSolution*

1. **for** $t = 1$ to $Max_iterations$ **do**
 2. $Solution \leftarrow Greedy_Randomized_Construction(M, \alpha);$
 3. $Solution \leftarrow Local_Search(Solution);$
 4. $Update_Solution(Solution, BestSolution);$
 5. **end for**
 - 6 **return** *BestSolution*;
-

The pseudo code for the GraspEnS algorithm [31], which is similar to GRASP, is given below.

4.1.1 Greedy randomized construction phase of GraspEnS

During the construction phase of the GraspEnS algorithm, a feasible solution is constructed. First, like the traditional greedy ensemble selection approaches, according to the predictive accuracy of the base classifiers on the pruning dataset, the classifier with the best accuracy is selected and added to the initial feasible solution. Then, the algorithm enters a loop, and the remaining classifiers are ranked according to the values of the greedy function $G(h_i, S)$. Next, the best α candidate classifiers are placed in the RCL. Finally, a classifier is randomly selected from the RCL and expanded into the feasible solution. The process is repeated until the size of the feasible solution is greater than or equal to the required size. The pseudo code for the greedy randomized construction phase of GraspEnS is given below [31].

Algorithm 3 Greedy_randomized_construction phase

Input: α — an invariable parameter
 Pr — the pruning dataset: $\text{Pr} = \{(x_i, y_i), i = 1, 2, \dots, N_{pr}\}$
 E — the original ensemble $E = \{h_i(x)\}_i^k$
 M — the required number of feasible solutions

Output: *Solution* — the feasible solution constructed in this phase

1. Select classifier h with the best performance on the pruning dataset Pr;
 2. $S = \{h\};$
 3. **while** $\|S\| < M$ // $\|S\|$ denotes the current number of the feasible solution
 4. Sort the remaining candidate classifiers in $E \setminus S$ using greedy function $G(h_i, S);$
 5. Select best α classifiers to build the RCL from the remaining ones in $E \setminus S;$
 6. Randomly select a classifier h_s from RCL;
 7. $S \leftarrow S \cup \{h_s\};$
 8. **end while**
 9. $Solution = S;$
 10. **return** *Solution*;
-

It is clear that this phase is similar to the forward search procedure of the classical greedy ensemble selection approaches, but differs in that random factors are incorporated into this phase. The feasible solutions obtained from each iteration are not the same as one another because of the

incorporated random factors. Moreover, each different feasible solution could be regarded as a new starting point for the local search phase, thereby allowing GraspEnS to realize a multi-start search.

The greedy function $G(h_i, S)$, the required size of the feasible solution M , and the constant parameter α play important roles in the GraspEnS algorithm, and need to be appropriately set and tuned. Since the greedy function $G(h_i, S)$ is directly related to the construction of the RCL, proper adoption thereof is particularly important. Any of the evaluation measures introduced in Section 3 can be used as the greedy function $G(h_i, S)$ for GraspEnS, including ACC, COM, CON, and UWA. The effect of the different values of parameters M and α and the different choices of greedy function $G(h_i, S)$ on our proposed ELMsGraspEnS algorithm is analyzed in Section 5.

4.1.2 Local search phase of GraspEnS

The solution obtained from the construction phase may not be the optimal solution. Generally, the solution is refined during the local search phase. Several factors could influence the successful realization of the local search phase, such as the structure of the neighborhood, the starting solution itself, and the neighborhood search method. For ELMsGraspEnS, the choice of neighbors is an important factor. In constructing the neighbors of the current feasible solution, if the classifier is included in the solution, it should be removed from it; otherwise the classifier should be extended into it [31]. For instance, consider the original ensemble $S = \{h_1, h_2, h_3, h_4\}$ and feasible solution $S_1 = \{h_2, h_3, h_4\}$. The neighborhood of S_1 is $\{\{h_1, h_2, h_3, h_4\}, \{h_3, h_4\}, \{h_2, h_4\}, \{h_2, h_3\}\}$. Algorithm 4 gives the pseudo code of the local search phase of GraspEnS [31].

As shown in Algorithm 4, the current feasible solution is updated once a better performing solution is found. The size of the final solution is indirectly affected by parameter M in the construction phase. The loop iterates k times, and a part of the entire neighborhood of the current subensemble is investigated.

4.2 ELMs' ensemble selection algorithm based on GRASP for credit scoring

For the ELMsGraspEnS algorithm, first the whole dataset is randomly divided into a training subset, a pruning subset, and a test subset and the ELM is used as the base learner for ELMsGraspEnS. Multiple ELM classifiers can be produced by implementing the ELM multiple times. The generated ELMs are inherently different from one another, thereby guaranteeing the diversity of the original ensemble. Next, an appropriate subset is selected from the original ensemble by

the GraspEnS algorithm according to the prediction performance of the base ELM classifiers on the pruning dataset. Finally, the unweighted majority voting method is used to combine the selected subset of ELM classifiers, so as to make predictions for the test data. The entire framework of ELMsGraspEnS is illustrated in Fig. 2, while Algorithm 5 gives the pseudo code for the proposed ELMsGraspEnS algorithm.

Algorithm 4 Local_Search phase

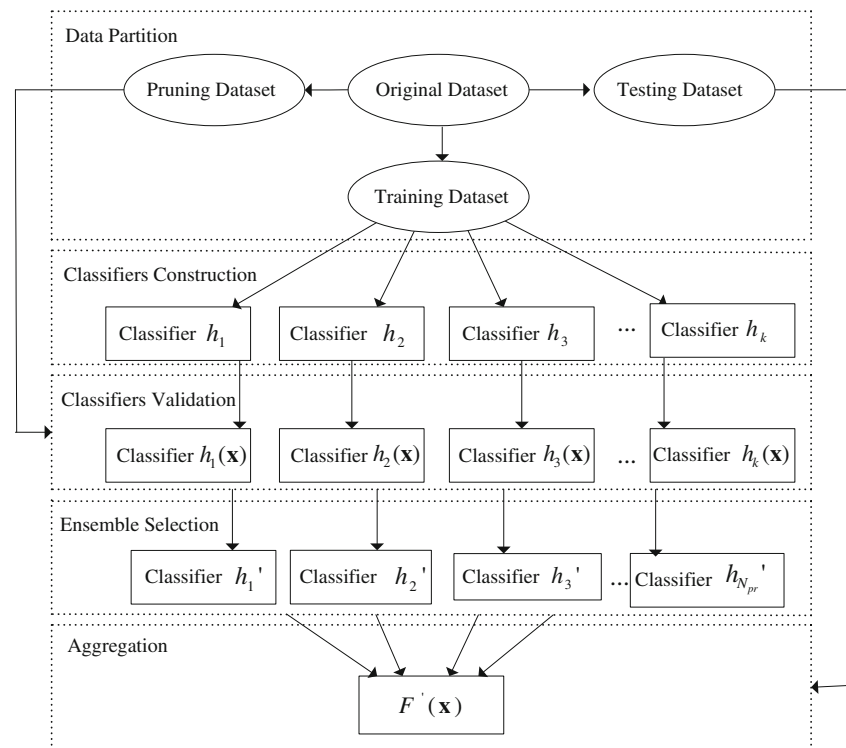
Input: α – an invariable parameter
 Pr – the pruning dataset, $Pr = \{(x_i, y_i), i = 1, 2, \dots, N_{pr}\}$
 E – the original ensemble, $E = \{h_i(x)\}_i^k$
 M – the required number of feasible solutions
Solution – the feasible solution obtained in the construction phase

Output: *Solution* – the refined feasible solution obtained in the local search phase

1. *currentSolution* = *Solution*;
2. **for** $i = 1 : k$
3. **if** $h_i \notin \text{currentSolution}$
4. $\text{tempSolution} = \{\text{currentSolution}, h_i\}$;
5. **else**
6. $\text{tempSolution} = \text{currentSolution} \setminus \{h_i\}$;
7. **end if**
8. **if** $ACC(\text{tempSolution}) > ACC(\text{currentSolution})$
9. $\text{currentSolution} = \text{tempSolution}$;
10. **end if**
11. **end for**
12. *Solution* = *currentSolution*;
13. **return** *Solution*;

The main advantages of the proposed ELMsGraspEnS algorithm are the following. First, the learning speed of the ELM is extremely high, and it is seldom that the ELM suffers from overfitting or the local minima problem. Moreover, the ELM algorithm is easy to implement. The input weights and bias of the ELM are initially randomly assigned, and these do not need to be adjusted. Furthermore, its hidden layer output matrix and hence, the output weights can be calculated explicitly. The network can be obtained with very little computational cost in only a few steps. Besides, the generalization performance of the ELM is better than the traditional feedforward neural network learning algorithm in several classification applications. Our proposed ELMsGraspEnS algorithm intuitively inherits the inherent advantages of the ELM, because the ELM is used as the base learner in the original ensemble.

Fig. 2 Framework of the ELMsGraspEnS algorithm for enterprise credit risk assessment



Second, homogeneous ELM models constitute the original ensemble investigated in this work. The ELM randomly sets the initial weights and bias of an SLFN, incorporating random factors into the ELMsGraspEnS, while the construction thereof is easy to implement.

Algorithm 5 ELMsGraspEnS algorithm

Input: Original dataset of N samples;
 α – an invariable parameter;
 M – the required number of feasible solutions;
 \tilde{N} – the number of hidden nodes;
 k – the size of original ensemble; $g(\mathbf{x})$
– activation function

Output: Prediction results of the test dataset

1. Randomly divide the original set of samples into three parts, i.e., training subset, pruning subset, and test subset;
 2. Generate k ELM classifiers;
 3. Prune the original ensemble produced in step 2 by GraspEnS, and obtain the best subensemble of the original ensemble;
 4. Aggregate the best subensemble using unweighted majority voting to make predictions on the test dataset.
-

Third, application research on credit risk assessment with ensemble selection approaches is limited. The popular ensemble learning method needs a large storage space and a vast amount of execution time, while its generaliza-

tion performance is often impacted by constituent models with low prediction performance. To address these problems, the ensemble selection approach is applied in our ELMsGraspEnS algorithm.

Fourth, the ELMsGraspEnS algorithm uses GraspEnS as its ensemble selection method, as discussed in Section 4.1. Therefore, it is obvious that the proposed ELMsGraspEnS algorithm naturally inherits the inherent characteristics of the GraspEnS method. ELMsGraspEnS is capable of escaping from local optima and realizing a multi-start search, and has superior generalization performance compared with the classical greedy ensemble selection approaches.

Last, ELMsGraspEnS achieves better classification performance than the classical greedy ensemble methods, like the DHCEP algorithm and three other methods, namely, the BSM, ALL and Adaboost.M1 methods. Moreover, the performance of ELMsGraspEnS is obviously superior to that of SVM and MLP. A detailed discussion of the performance comparison follows in the next section.

5 Experiments

5.1 Experimental datasets

Three real-world credit datasets from the UCI Machine Learning repository [53] were used to evaluate the performance of the proposed algorithm: the Australian credit

Table 1 Datasets used for credit risk assessment tasks

Data set	Total	Training (60%)	Pruning (20%)	Test (20%)	Attributes	Classes
Australian credit dataset	690	414	138	138	15	2
German credit dataset	1000	600	200	200	25	2
Japanese credit dataset	653	393	130	130	14	2

dataset, German credit dataset, and Japanese credit dataset. There were, however, some missing attribute values in the Japanese credit dataset. After removing these missing values from the original dataset, we were left with 653 data. In addition, because an m -class attribute must be replaced by $m-1$ binary attributes, attributes A6 and A7 of the Japanese credit dataset were found to have too many categories, greatly increasing the input space. Therefore, these two attributes were deleted from the dataset [24].

Each dataset was divided into three parts: a training set, pruning set, and test set, consisting of 60%, 20%, and 20% of the full dataset, respectively. Table 1 gives detailed information about the three datasets used in our experiments, including the number of attributes, classes, and the sizes of the whole dataset, training set, pruning set, and test set. In this work, we used a separate pruning set to reduce the risk of overfitting. Besides, the three adopted datasets were sufficiently large, so adequate data were available for simultaneously creating training, pruning, and test datasets from each.

5.2 Experimental setup

As mentioned in the previous sections, we executed the ELM algorithm many times to generate the original ensemble with multiple ELMs. Specifically, an original ensemble with 100 ELM classifiers was constructed in this work. To demonstrate the effectiveness and efficiency of the proposed ELMsGraspEnS algorithm, we compared it with the classical DHCEP algorithm. In addition, two baseline methods corresponding to two extreme pruning scenarios were implemented, i.e., the best single model (BSM) and ALL. The former selects the best single model from the original ensemble according to the performance of the classifiers on the pruning set, while the latter retains all the models in the original ensemble. In addition, Adaboost.M1 was also used as a comparative ensemble selection method to investigate whether the proposed ELMsGraspEnS algorithm has better classification and generalization performance. Finally, to illustrate the efficiency of our proposed method, an SVM and multi-layer perceptron (MLP) were also compared with the ELMsGraspEnS algorithm. For the SVM method, we

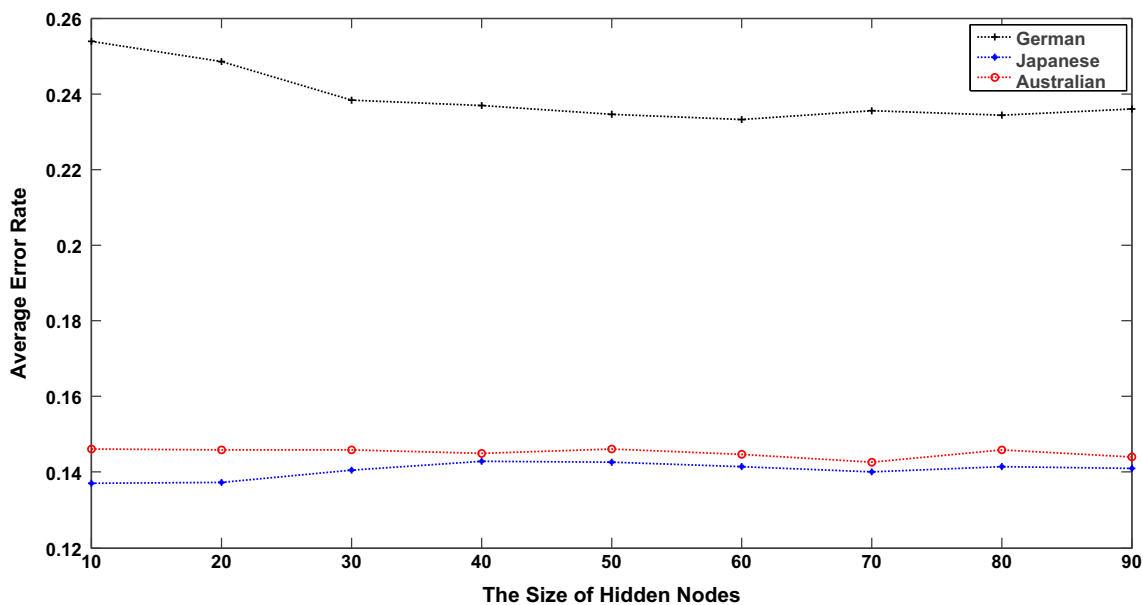


Fig. 3 Average error rates obtained with varying numbers of hidden nodes \tilde{N} using ELMsGraspEnS with CON as the greedy function $G(h_i, S)$ on three benchmark credit risk assessment datasets

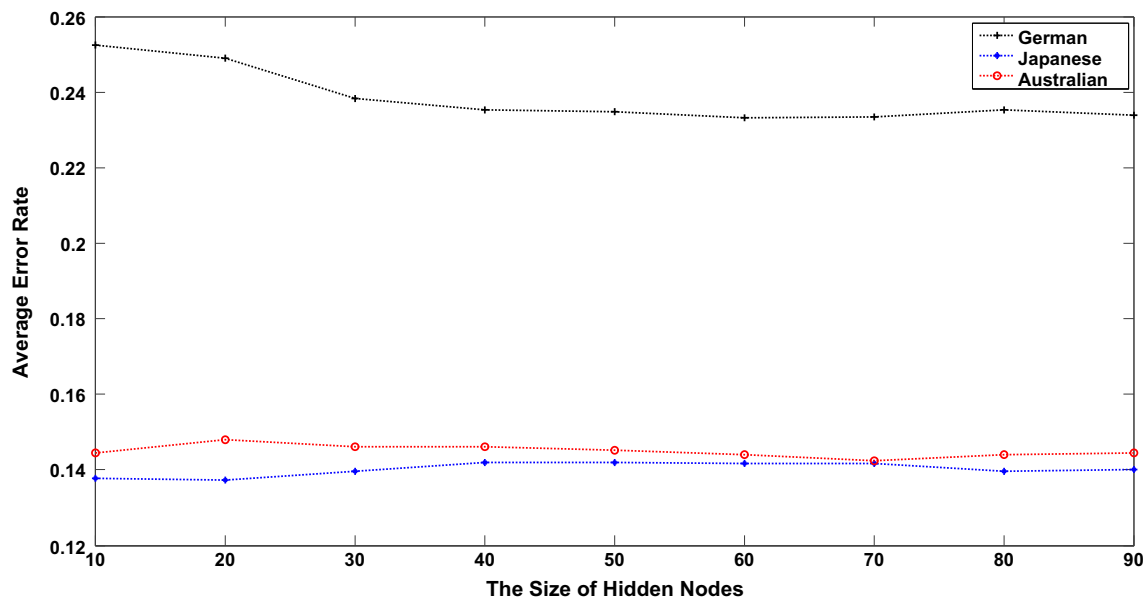


Fig. 4 Average error rates obtained with varying numbers of hidden nodes \tilde{N} using ELMsGraspEnS with UWA as greedy function $G(h_i, S)$ on three benchmark credit risk assessment datasets

used the radial kernel with gamma set to one; the cost parameter of the SVM was set to 0.5. The nodes in the hidden layer and the learning rate of MLP were set to 16 and 0.6, respectively.

To minimize the effect of the changes in the training, pruning, and test data, the experiments reported in this section were performed 50 times for each classification task using a different randomized ordering of its examples. All reported results are averages over these 50 repetitions.

5.3 Experimental results and discussion

From the discussion in Sections 2 and 4, it is clear that five important parameters must be properly set and tuned; that is, the number of hidden nodes \tilde{N} , greedy function $G(h_i, S)$, the required number of feasible solutions M , constant parameter α , and the number of Max.iterations. First

we discuss the number of Max.iterations, which determines the running times of ELMsGraspEnS. Too small a number of Max.iterations causes the classification performance to be far from the expected result, while too large a number may require much longer running time, and hence, reduce the practicability of the algorithm. In this work, the number of Max.iterations was set to 500, a generally appropriate number.

Next, we analyzed the number of hidden nodes \tilde{N} . We chose the CON and UWA evaluation measures as the greedy function, and set M to 30 and α to 0.4. It is clear that \tilde{N} directly affects the performance of the ELM. If \tilde{N} is too small or too large, the final result will be far from the expected one. Consequently, \tilde{N} was varied in the set $\{10, 20, 30, 40, 50, 60, 70, 80, 90\}$. Figures 3 and 4 show the average error rates of the base ELMs with different numbers of hidden nodes using the ELMsGraspEnS algorithm with

Table 2 Average error rates obtained with different evaluation measures using DHCEP and ELMsGraspEnS algorithms on the three benchmark credit risk assessment datasets over 50 runs

		Average Error Rate			
		ACC	COM	CON	UWA
German	DHCEP	0.2403	0.2459	0.2415	0.2391
	ELMsGraspEnS	0.2359	0.2372	0.2360	0.2352
Japanese	DHCEP	0.1497	0.1468	0.1462	0.1454
	ELMsGraspEnS	0.1435	0.1426	0.1460	0.1445
Australian	DHCEP	0.1430	0.1449	0.1458	0.1439
	ELMsGraspEnS	0.1401	0.1409	0.1406	0.1401

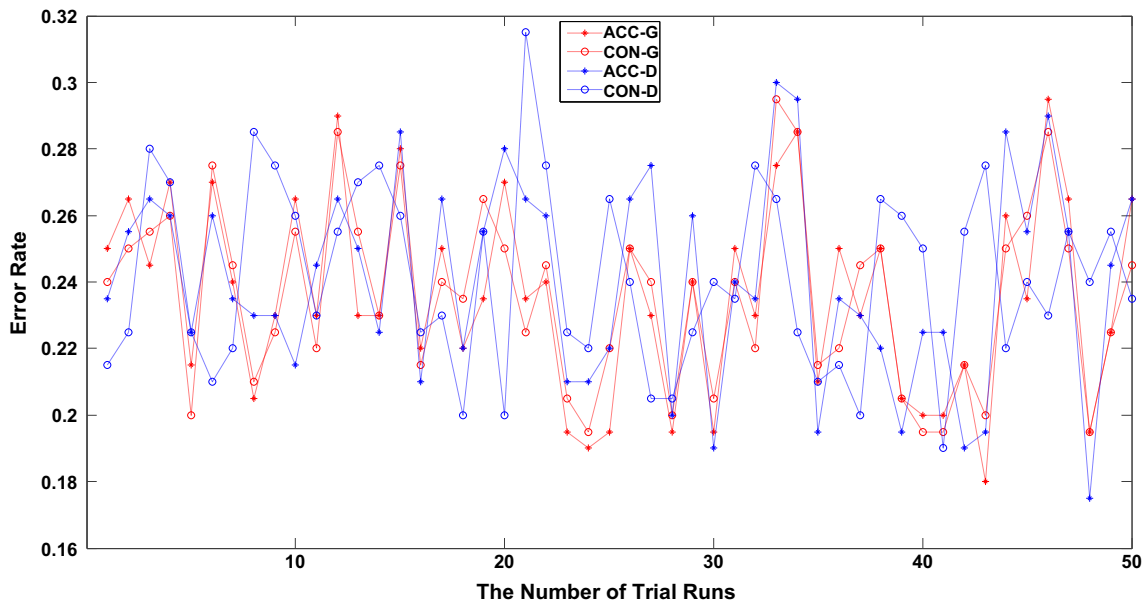


Fig. 5 Error rates obtained with the ACC and CON evaluation measures using DHCEP and ELMsGraspEnS on the German credit dataset

CON and UWA as the greedy function $G(h_i, S)$ on the three datasets.

From these two figures, we can see that by increasing the number of hidden nodes \tilde{N} , the average error rate changes only slightly, especially for the Australian and Japanese credit datasets. In other words, \tilde{N} has no clear effect on the prediction error rates in a certain range. Thus, in the remaining experiments, \tilde{N} was set to 70.

Next, we discuss parameters M and α . In [31], 15 datasets were used to verify the effect of these two parameters on the classification performance of GraspEnS. The

conclusion reached was that there was no significant effect for most of the datasets. According to the experimental results in [31], parameters M and α were set to 30 and 0.4, respectively, in this work.

The final and most significant parameter is the greedy function $G(h_i, S)$. Table 2 gives the performance of DHCEP and ELMsGraspEnS using ACC, COM, CON, and UWA as the evaluation measures on the three benchmark credit risk assessment datasets.

It is evident from Table 2 that ELMsGraspEnS achieves better performance. On all three datasets, ELMsGraspEnS

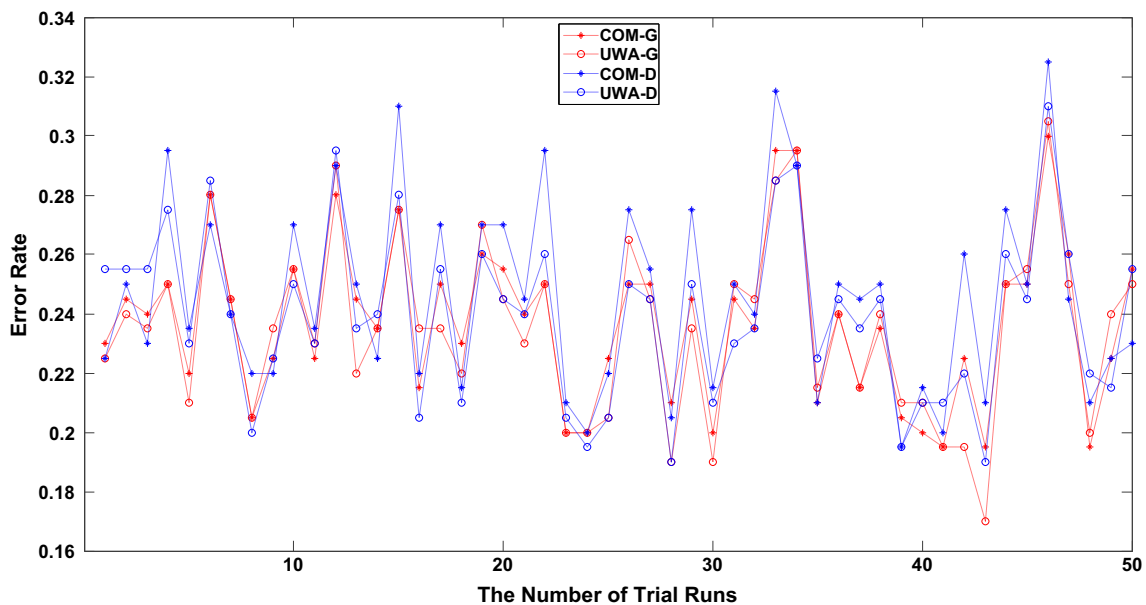


Fig. 6 Error rates obtained with the COM and UWA evaluation measures using DHCEP and ELMsGraspEnS on the German credit dataset

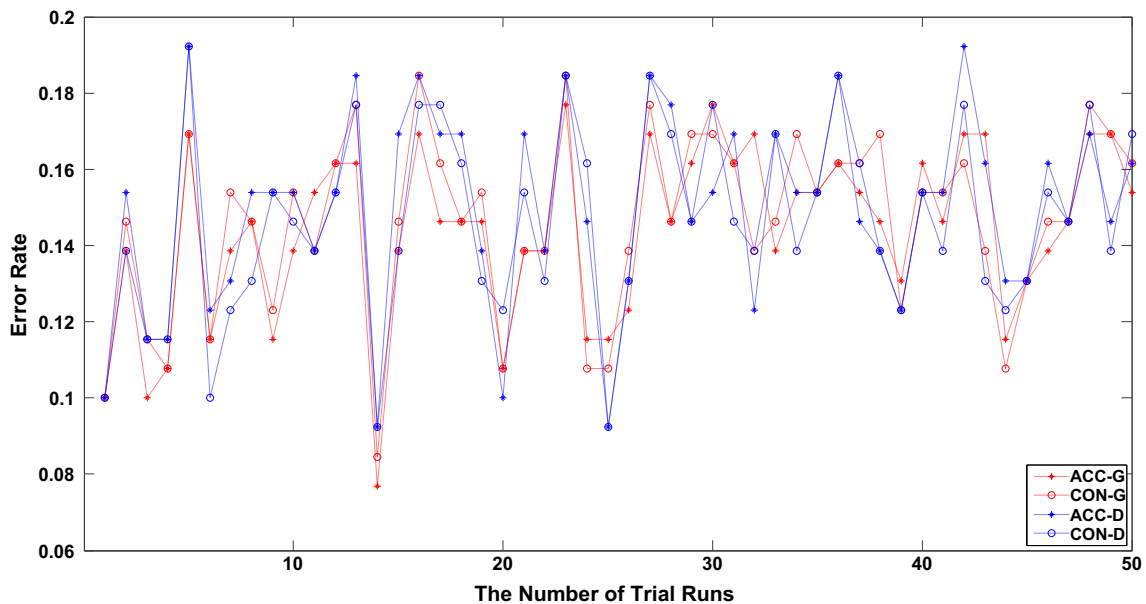


Fig. 7 Error rates obtained with the ACC and CON evaluation measures using DHCEP and ELMsGraspEnS on the Japanese credit dataset

has lower average error rates compared with DHCEP when using all four evaluation measures, i.e., ACC, COM, CON, and UWA. Moreover, the lowest average error rates for each dataset were obtained using ELMsGraspEnS.

Figures 5, 6, 7, 8, 9, and 10 show the detailed error rates obtained by different evaluation measures using the DHCEP and ELMsGraspEnS algorithms over 50 repeated experimental runs on the German, Japanese, and Australian credit datasets. It should be noted that the mnemonic ACC-G appearing in these figures refers to the corresponding

experimental results that were obtained using the proposed ELMsGraspEnS algorithm with the ACC measure, while ACC-D means that the corresponding results were obtained using the DHCEP method with the ACC measure. Using the same analogy, the meaning of the remaining mnemonics used in these figures should be clear.

It is obvious from the results shown in the six figures above and Table 2 that the error rates obtained by DHCEP are clearly higher than those by ELMsGraspEnS. Furthermore, in most cases, the highest error rate is obtained by

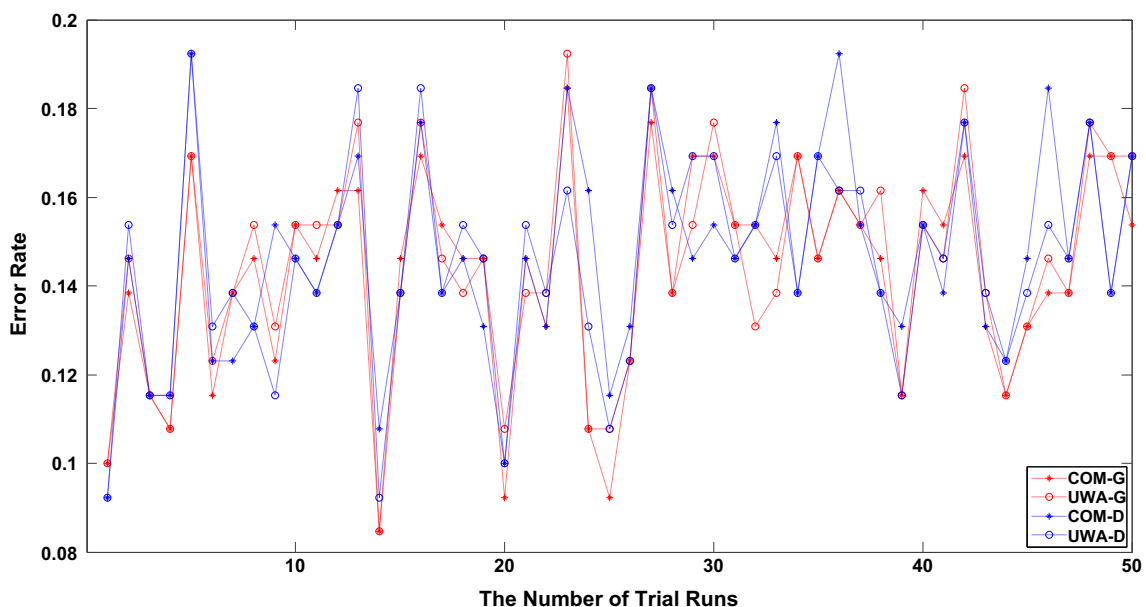


Fig. 8 Error rates obtained with the COM and UWA evaluation measures using DHCEP and ELMsGraspEnS on the Japanese credit dataset

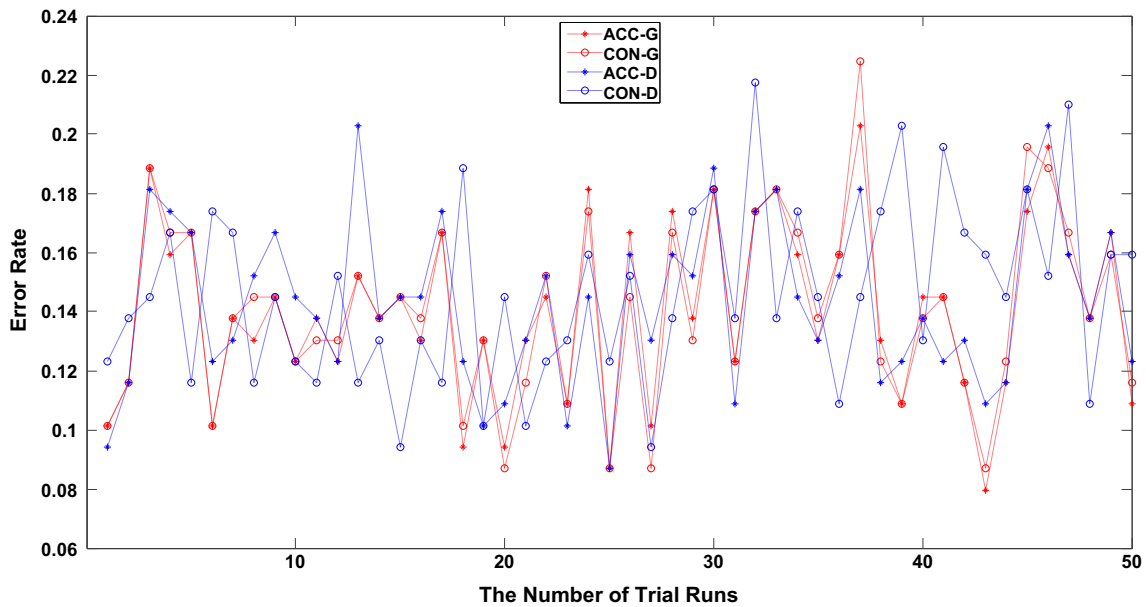


Fig. 9 Error rates obtained with the ACC and CON evaluation measures using DHCEP and ELMsGraspEnS on the Australian credit dataset

DHCEP. Therefore, from Figs. 5–10 and Table 2, it can be concluded that the ELMsGraspEnS algorithm achieves better performance than DHCEP on the three credit datasets.

Next, to ascertain whether ELMsGraspEnS is significantly superior to DHCEP, *t*-tests were applied to the results of the DHCEP and ELMsGraspEnS algorithms using the four measures on the three datasets. The null hypothesis H_0 indicates that there is no significant difference between Model A and Model B. The detailed *p*-values are shown in Table 3. Items marked with an asterisk (*) denote that

Model A is significantly better than Model B at the 5% significance level. The “Improvement” columns indicate the relative improvement of Model A over Model B in terms of average error rate.

From Table 3, it is clearly seen that ELMsGraspEnS yields better prediction results in all cases. For the German and Australian credit datasets, the classification performance of ELMsGraspEnS is significantly better than that of the classical DHCEP method with COM and UWA as the evaluation measures; these measures belong to the

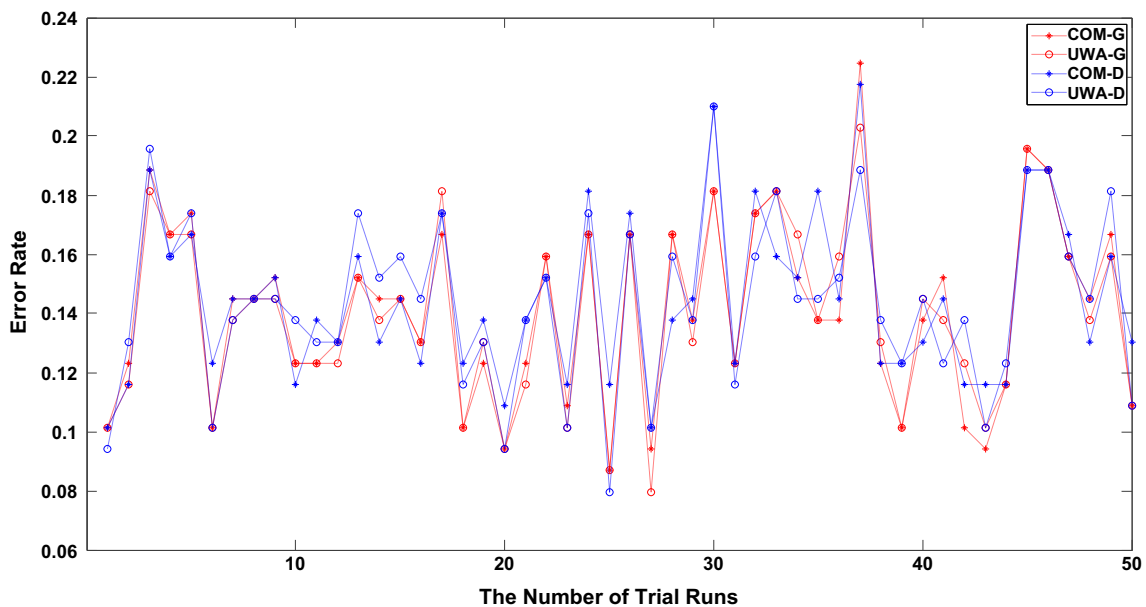


Fig. 10 Error rates obtained with the COM and UWA evaluation measures using DHCEP and ELMsGraspEnS on the Australian credit dataset

Table 3 Results of *t*-tests applied to the average error rates of the ELMsGraspEnS and DHCEP algorithms on the three datasets

Model A	Model B	Average Error Rate					
		German		Japanese		Australian	
		Improvement(%)	<i>p</i>	Improvement(%)	<i>p</i>	Improvement(%)	<i>p</i>
ACC-G	ACC-D	0.57	0.0503	0.73	0.0080*	0.33	0.1034
COM-G	COM-D	1.15	1.04e-4*	0.46	0.0516	0.46	0.0207*
CON-G	CON-D	0.72	0.1490	0.02	0.4740	0.59	0.1793
UWA-G	UWA-D	0.51	0.0226*	0.11	0.3199	0.44	0.0112*

diversity-based evaluation measures. In other words, for diversity-based evaluation measures, there are only 1/3 cases where the ELMsGraspEnS method does not significantly improve the classification performance obtained by the DHCEP algorithm. That is a normal result, because it is well known that the DHCEP method achieves excellent classification performance, and it is a state-of-the-art ensemble pruning algorithm. Regarding the performance-based evaluation measure ACC, the differences between the two algorithms are not significant. The reason for this may be that ACC focuses on accuracy and ignores diversity, whereas the property of diversity is important in ensemble learning. The results obtained for the Japanese credit dataset differ from those of the other two datasets partly because some values in the dataset were missing and two attributes were deleted, as mentioned in Section 5.1.

Finally, ELMsGraspEnS was compared with the other five algorithms mentioned previously, i.e., ALL, BSM, Adaboost.M1, SVM, and MLP. Table 4 shows the results obtained by ELMsGraspEnS with each of the four evaluation measures as the greedy function $G(h_i, S)$ and the other five algorithms on the three benchmark credit risk assessment datasets.

Table 4 clearly shows that the average error rates obtained by ELMsGraspEnS with the four corresponding measures are distinctly lower than those by the other five algorithms, except for ELMsGraspEnS with the CON measure on the Japanese credit dataset. On the German credit dataset,

UWA-G performs the best; COM-G achieves the best generalization performance on the Japanese credit dataset; and both ACC-G and UWA-G have the lowest average classification error rates on the Australian credit dataset.

Figures 11, 12, and 13 show the detailed error rates averaged over 50 experimental runs obtained by ELMsGraspEnS with the four corresponding evaluation measures and the five other algorithms, i.e., Adaboost.M1, ALL, BSM, SVM, and MLP, on the three benchmark credit datasets. These figures show that on all three datasets, the classification performance of the proposed ELMsGraspEnS algorithm is better than that of Adaboost.M1, ALL, BSM, SVM, and MLP. Figure 11 shows that in most cases on the German credit dataset, fewer test samples are misclassified by ELMsGraspEnS with the four corresponding evaluation measures than by BSM, SVM, and MLP. Although the classification error rates of Adaboost.M1, ALL, and ELMsGraspEnS are close to each other, the classification performance of ELMsGraspEnS is still better than that of the other two algorithms. From Fig. 12 and Table 4, it can be seen that COM-G achieves the best generalization performance on the Japanese credit dataset. The classification error rates of UWA-G and ACC-G are similar and are ranked second. Adaboost.M1 and ALL achieve similar average classification performances and are ranked third. CON-G performs a little worse than Adaboost.M1 and ALL, and is ranked fourth. BSM and SVM are ranked fifth and sixth, respectively. The highest average error rate is obtained by MLP. It can be seen from Fig. 13 and Table 4 that the

Table 4 Average error rates using the ELMsGraspEnS algorithm with the four different evaluation measures and five other methods on the three benchmark datasets

	Average Error Rate								
	ACC-G	COM-G	CON-G	UWA-G	Adaboost.M1	ALL	BSM	SVM	MLP
German	0.2359	0.2372	0.2360	0.2352	0.2384	0.2388	0.2569	0.2527	0.2727
Japanese	0.1435	0.1426	0.1460	0.1445	0.1446	0.1449	0.1468	0.1597	0.1765
Australian	0.1401	0.1409	0.1406	0.1401	0.1412	0.1420	0.1457	0.1561	0.1742

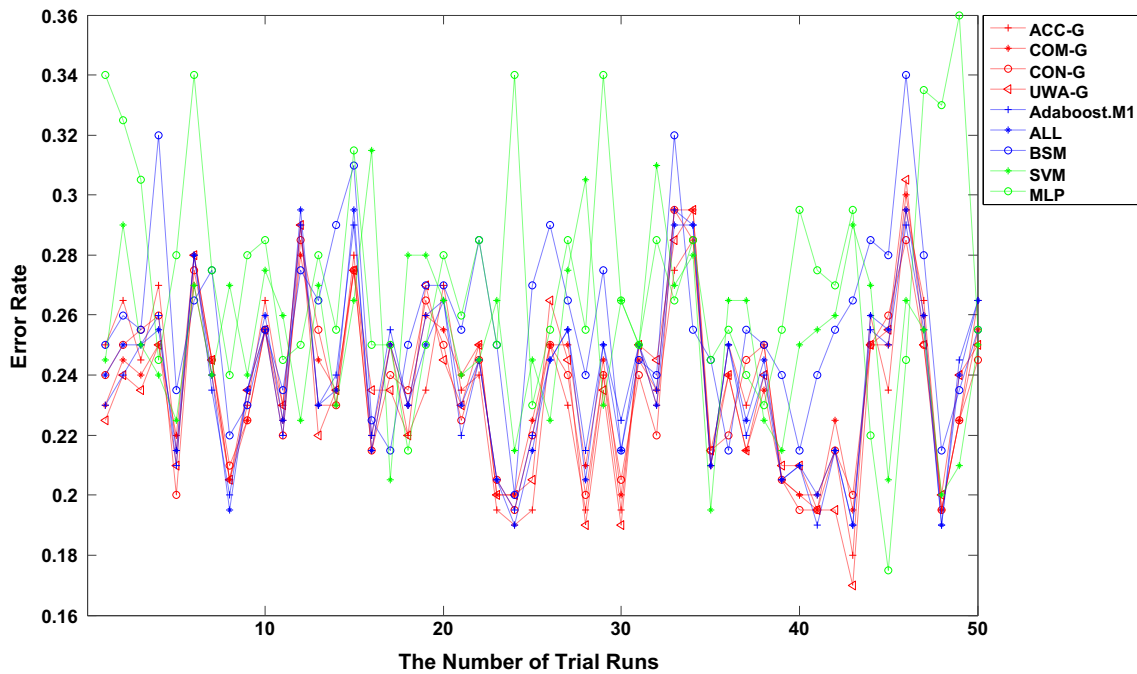


Fig. 11 Error rates obtained by ELMsGraspEnS with the four corresponding evaluation measures and the other five algorithms, i.e., Adaboost.M1, ALL, BSM, SVM, and MLP on the German credit dataset

generalization performances of ACC-G and UWA-G are the best on the Australian credit dataset. The performances of COM-G and CON-G are ranked second. The performances

of Adaboost.M1, ALL and BSM are close to each other, and are inferior to the other four algorithms, i.e., ACC-G, COM-G, CON-G and UWA-G. And the performance of

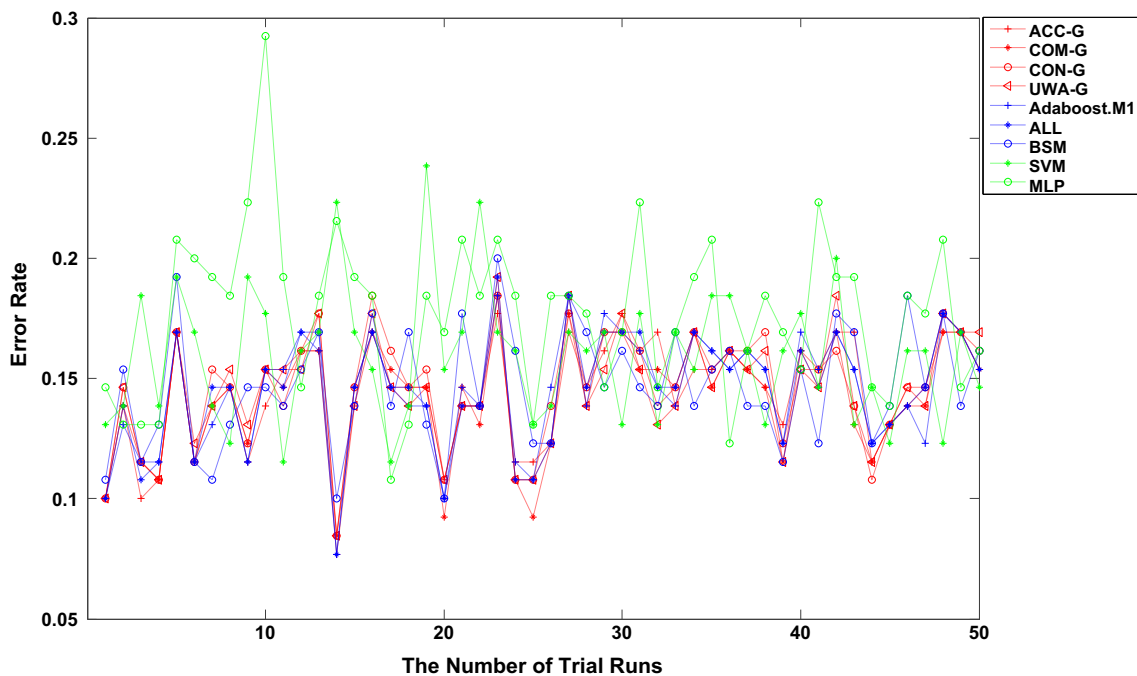


Fig. 12 Error rates obtained by ELMsGraspEnS with the four corresponding evaluation measures and the other five algorithms, i.e., Adaboost.M1, ALL, BSM, SVM, and MLP on the Japanese credit dataset

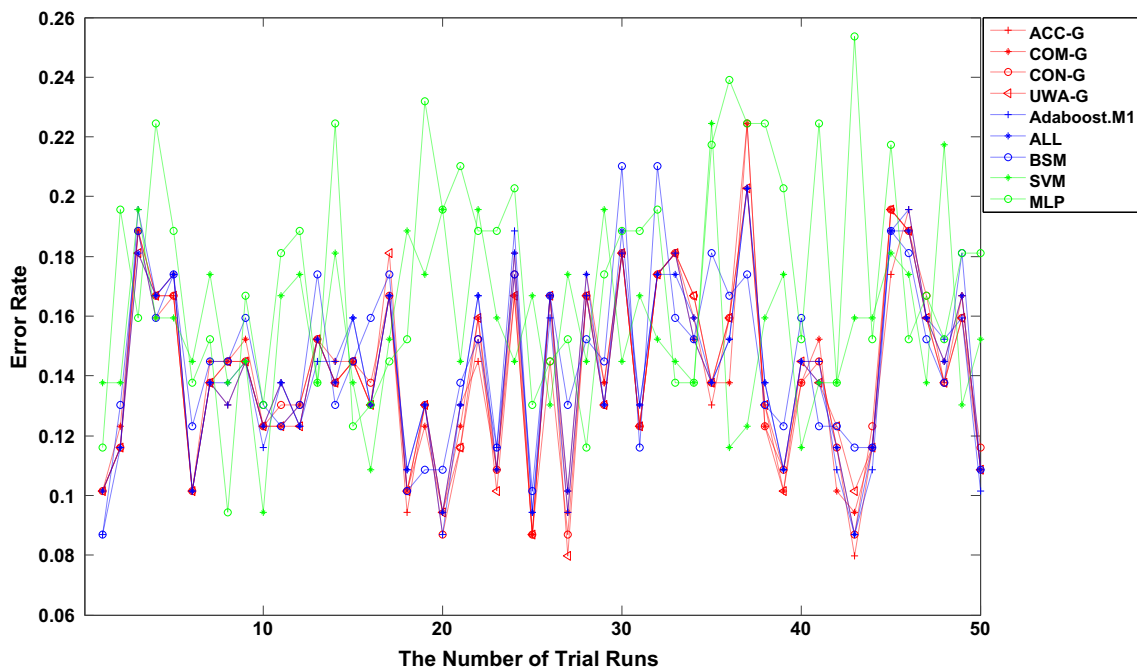


Fig. 13 Error rates obtained by ELMsGraspEnS with the four corresponding evaluation measures and the other five algorithms, i.e., Adaboost.M1, ALL, BSM, SVM, and MLP on the Australian credit dataset

Table 5 Results of *t*-tests comparing ELMsGraspEnS with the other five algorithms on the three benchmark credit datasets

Model A	Model B	Average Error Rate					
		German		Japanese		Australian	
		Improvement(%)	<i>p</i>	Improvement(%)	<i>p</i>	Improvement(%)	<i>p</i>
ACC-G	BSM	2.82	3.17e-7*	0.39	0.1152	0.66	0.0111*
	ALL	0.38	0.0143*	0.16	0.1050	0.22	0.0156*
	Adaboost.M1	0.32	0.0729	0.13	0.2096	0.13	0.1699
	SVM	2.24	0.0019*	1.93	0.0018*	1.9	0.0062*
	MLP	5.06	6.1e-7*	4.01	1.3e-7*	4.13	2.5e-6*
COM-G	BSM	2.65	5.0e-8*	0.49	0.0686	0.56	0.0300*
	ALL	0.21	0.0862	0.27	0.0117*	0.13	0.1538
	Adaboost.M1	0.18	0.1628	0.23	0.0397*	0.03	0.4075
	SVM	2.07	0.0021*	2.03	7.9e-4*	1.8	0.0079*
	MLP	4.89	1.1e-6*	4.12	4.8e-8*	4.03	3.6e-6*
CON-G	BSM	2.81	3.12e-9*	0.09	0.3950	0.59	0.0265*
	ALL	0.37	0.0339*	-0.001	0.8380	0.16	0.1041
	Adaboost.M1	0.31	0.0691	-0.002	0.8452	0.07	0.3259
	SVM	2.23	0.0012*	1.63	0.0056*	1.8	0.0081*
	MLP	5.05	8.6e-7*	3.7	3.9e-7*	4.07	3.2e-6*
UWA-G	BSM	2.92	1.91e-7*	0.27	0.1958	0.66	0.0113*
	ALL	0.47	0.0129*	0.05	0.3523	0.22	0.0341*
	Adaboost.M1	0.42	0.0275*	0.01	0.4580	0.13	0.2178
	SVM	2.34	0.0015*	1.8	0.0024*	1.9	0.0062*
	MLP	5.16	8.0e-7*	3.89	8.9e-8*	4.13	2.5e-6*

SVM and MLP is obviously lower than that of our proposed ELMsGraspEnS algorithm.

The results of *t*-tests comparing the proposed ELMsGraspEnS algorithm with the other five algorithms, i.e., BSM, ALL, Adaboost.M1, SVM, and MLP, are shown in Table 5. These results clearly show that for all credit datasets, our proposed method is significantly better than the state-of-the-art algorithms, i.e., SVM and MLP. With the exception of the Japanese credit dataset, ELMsGraspEnS shows performance improvements compared with the other three algorithms, i.e., Adaboost.M1, ALL and BSM, on the other two datasets. On the German and Australian credit dataset, for 10 out of all the 24 comparative cases, the proposed method ELMsGraspEnS did not significantly improve the prediction performance obtained by Adaboost.M1 or ALL. There were no cases where the difference between ELMsGraspEnS and the BSM method was not significant. In other words, our proposed method is significantly better than the BSM algorithm in all cases. Compared with the ALL method, there are only three cases where the new ELMsGraspEnS method is not significantly better. This result is not surprising, since no method exists that performs the best in all cases. There are seven cases where the difference between the typical Adaboost.M1 algorithm and the ELMsGraspEnS method is not significant; on the German credit dataset, however, ELMsGraspEnS with the UWA measure is significantly better than Adaboost.M1. This could lead us to conclude that the size and attribute information of the dataset, and the selection of the evaluation measure influence classification performance of the ELMsGraspEnS method. However, there are no significant differences between ELMsGraspEnS and the other three methods, i.e., BSM, ALL, Adaboost.M1, in most cases on the Japanese credit dataset. The reason for this may be the same as given above, i.e., it is partly because some values in the dataset were missing and two attributes were deleted.

In summary, the proposed ELMsGraspEnS algorithm achieves better generalization performance than the typical DHCEP ensemble selection method and the other five competitors mentioned in this paper.

6 Conclusion and future works

Credit risk assessment has become a main concern in the financial and banking sector, since these institutions need to decide whether to grant loans to applicants. Even a slight improvement in the assessment accuracy could greatly reduce the losses of financial institutions. In this study, we combined the ELM with our previously proposed GraspEnS algorithm to create the novel ELMsGraspEnS algorithm for credit risk assessment of enterprises. ELMsGraspEnS

not only inherits the inherent advantages of the ELM and GraspEnS, but also effectively fuses their advantages: the learning speed of the new method is extremely fast, the local optima problem is ameliorated, and multi-start is realized by our proposed algorithm, etc. At the same time, we employed the ensemble selection paradigm in this new method; the disadvantages of the original ensemble method are partly avoided by our proposed ELMsGraspEnS.

The ELMsGraspEnS algorithm was applied to three real world credit datasets, i.e., the Australian, Japanese, and German credit datasets, drawn from the UCI machine learning repository. The experimental results show that in most cases, application of ELMsGraspEnS substantially improves the performance of credit risk assessment compared with several other state-of-the-art algorithms; this is especially true for the ELMsGraspEnS algorithm using the UWA measure as its greedy function. It can be concluded that ELMsGraspEnS simultaneously exhibits high efficiency and effectiveness.

In our future work, we intend to investigate a feasible approach for assigning the number of hidden nodes of the constituent ELMs in the ensemble. Another possible research direction concerns the exploration of a better neighborhood structure or a better neighborhood search method to further improve the performance of our algorithm.

Acknowledgments This work is supported by the National Natural Science Foundation of China under Grants no. 61473150, 61100108, and 61375021. It is also supported by the Natural Science Foundation of Jiangsu Province of China under Grant no. BK20131365, and the Qing Lan Project, no. YPB13001. We would like to express our appreciation for the valuable comments from reviewers and editors.

References

1. Laha A (2007) Building contextual classifiers by integrating fuzzy rule based classification technique and k-nn method for credit scoring. *Adv Eng Inform* 21:281–291
2. Li ST, Shiue W, Huang MH (2006) The evaluation of consumer loans using support vector machines. *Expert Syst Appl* 30(4):772–782
3. Vellido A, Lisboa PJG, Vaughan J (1999) Neural networks in business: A survey of applications (1992–1998). *Expert Syst Appl* 17(1):51–70
4. Karels G, Prakash A (1987) Multivariate normality and forecasting of business bankruptcy. *J Business Finance Acc* 14(4):573–593
5. Reichert AK, Cho CC, Wagner GM (1983) An examination of the conceptual issues involved in developing credit-scoring models. *J Bus Econ Stat* 1(2):101–114
6. West D (2000) Neural network credit scoring models. *Comput Oper Res* 27(11–12):1131–1152
7. Thomas LC (2000) A survey of credit and behavioral scoring: Forecasting financial risks of lending to customers. *Int J Forecast* 16(2):149–172

8. Friedman JH (1991) Multivariate adaptive regression splines. *Ann Stat* pp. 1–141
9. Huang Z, Chen H, Hsu CJ, Chen WH, Wu SS (2004) Credit rating analysis with support vector machines and neural network: A market comparative study. *Decis Support Syst* pp. 543–558
10. Desai V, Crook J, Overstreet G (1996) A comparison of neural networks and linear scoring models in the credit union environment. *EJOR* 95(1):24–37
11. Hung C, Chen JH (2009) A selective ensemble based on expected probabilities for bankruptcy prediction. *Expert Syst Appl* 36(3):5297–5303
12. Polikar R (2006) Ensemble based systems in decision making. *IEEE Circ Syst Mag* 6(3):21–45
13. Buta P (1994) Mining for financial knowledge with CBR. *AI Expert* 9(2):34–41
14. Shin KS, Han I (2001) A case-based approach using inductive indexing for corporate bond rating. *Decis Support Syst* 32(1):41–52
15. Baesens B, van Gestel T, Viaene S, Stepanova M, Suykens J, Vanthienen J (2003) Benchmarking state-of-the-art classification algorithms for credit scoring. *J Oper Res Soc* 54(6):1082–1088
16. Huang CL, Chen MC, Wang CJ (2007) Credit scoring with a data mining approach based on support vector machines. *Expert Syst Appl* 33(4):847–856
17. Schebesch KB, Stecking R (2005) Support vector machines for classifying and describing credit applicants: Detecting typical and critical regions. *J Oper Res Soc* 56(9):1082–1088
18. Breiman L (1996) Bagging predictors. *Mach Learn* 24(2):123–140
19. Dietterich TG (2000) An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach Learn* 40(2):139–157
20. Opitz D, Maclin R (1999) Popular ensemble methods: An empirical study. *J Artif Intell Res* 11:169–198
21. Schapire RE (1990) The strength of weak learnability. *Mach Learn* 5(2):197–227
22. Wolpert DH (1992) Stacked generalization. *Neural Netw* 5(2):241–259
23. Tsai CF, Wu JW (2008) Using neural network ensembles for bankruptcy prediction and credit scoring. *Expert Syst Appl* 34(4):2639–2649
24. Yu LA, Wang SY, Lai KK (2008) Credit risk assessment with a multistage neural network ensemble learning approach. *Expert Syst Appl* 34(2):1434–1444
25. Nanni L, Lumini A (2009) An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. *Expert Syst Appl* 36(2):3028–3033
26. Hung C, Chen JH (2009) A selective ensemble based on expected probabilities for bankruptcy prediction. *Expert Syst Appl* 36(3):5297–5303
27. Partalas I, Tsoumakas G, Vlahavas I (2010) An ensemble uncertainty aware measure for directed hill climbing ensemble pruning. *Mach Learn* 81(2):257–282
28. Dai Q (2013) A competitive ensemble pruning approach based on cross-validation technique. *Knowl-Based Syst* 37:394–414
29. Dai Q, Liu Z (2013) ModEnPBT: A modified backtracking ensemble pruning algorithm. *Appl Soft Comput* 13:4292–4302
30. Dai Q (2013) A novel ensemble pruning algorithm based on randomized greedy selective strategy and ballot. *Neurocomputing* 122:258–265
31. Liu Z, Dai Q, Liu NZ (2013) Ensemble selection by GRASP. *Appl Intell*
32. Caruana R, Niculescu-Mizil A, Crew G, Ksikes A (2004) Ensemble selection from libraries of models, presented at Proceedings of the twenty-first international conference on machine learning
33. Caruana R, Munson A, Niculescu-Mizil A (2006) Getting the most out of ensemble selection. In: Proceedings of the international conference on data mining (ICDM), pp. 828–833
34. Partalas I, Tsoumakas G, Vlahavas I (2012) A study on greedy algorithm for ensemble pruning. In: Technical report TR-LPIS-360-1, Department of Informatics, Aristotle University of Thessaloniki, Greece
35. Zhou Z-H, Wu J, Tang W (2002) Ensembling neural networks: many could be better than all. *Artif Intell* 137:239–263
36. Martinez-Munoz G, Suarez A (2004) Aggregation ordering in bagging, Presented at international conference on artificial intelligence and applications
37. Banfield RE, Hall LO, Bowyer KW, Kegelmeyer WP (2005) Ensemble diversity measures and their application to thinning. *Information Fusion* 6:49–62
38. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. *Neurocomputing* 70(2):489–59
39. Rong HJ, Huang GB, Ong YS (2008) Extreme learning machine for multi-categories classification applications. *Int JT Conf Neural Netw*:1709–1713
40. Duan GL, Huang ZW, Wang JR (2009) Extreme learning machine for bank clients classification. *Int Conf Inf Manag, Innov Manag Ind Eng* 2:496–499
41. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multi-class classification. *IEEE Trans, Systems, Man, Cyber* vol. 42
42. Subbulakshmi CV, Deepa SN, Malathi N (2012) Extreme learning machine for two category data classification. *IEEE Int Conf Adv Commun Control Comput Technol* pp. 458–461
43. Landa-Torres I, Ortiz-García EG, Salcedo-Sanz SM, Segovia-Vargas J, Gil-López S, Miranda M, Leiva-Murillo JM, Del Ser J (2012) Evaluating the internationalization success of companies through a hybrid grouping Harmony Search - Extreme Learning Machine approach. *IEEE J Sel Top Sig Process* 6(4):1–11
44. Feo TA, Resende MG (1995) Greedy randomized adaptive search procedures. *J Glob Optim* 6:109–133
45. Gevezes T, Pitsoulis L (2013) A greedy randomized adaptive search procedure with path relinking for the shortest superstring problem. *J Glob Optim* pp. 1–25
46. Resende MGC, Ribeiro C (2003) Greedy randomized adaptive search procedures. In: *Handbook of Metaheuristics*. Springer, pp 219–249
47. Huang GB (2003) Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Trans Neural Netw* 14:274–281
48. Huang GB, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE Trans Neural Netw* 9(1):224–229
49. Dietterich TG (2000) Ensemble methods in machine learning. In: Proceedings of the 1st international workshop in multiple classifier systems, pp. 1–15
50. Breiman L (1996) Bagging predictors. *Mach Learn* 24:123–140
51. Schapire RE (1990) The strength of weak learnability. *Mach Learn* 5(2):197–227
52. Layeb A, Selmane M, Bencheikh Elhoucine M (2013) A new greedy randomized adaptive search procedure for multiple sequence alignment. *Int J Bioinforma Res Appl* 9:1–14
53. UCI repository of machine learning databases (2007). <http://www.ics.uci.edu/~mllearn/MLRepository.html> or <ftp://ics.uci.edu/pub/machine-learning-databases>



Ting Zhang was born in 1991. She received a bachelors Degree in Computer Science and Technology from Chaohu College, Anhui, China in 2013. Then she entered Nanjing University of Aeronautics and Astronautics, Nanjing, China, in the same year as a graduate student. Her research interests include pattern recognition, intelligent systems and machine learning.



Zhongchen Ma received a Bachelor of Science Degree in Information and Computing Science from Qingdao Agricultural University of China, Qingdao, China in 2012. Then he entered Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2012 as a graduate student. His research interests include time series prediction, pattern recognition, intelligent systems and machine learning.



Qun Dai received a B.Sc. from Nanjing University, PR China. In March, 2003, she completed her M.S. degree in computer science at Nanjing University of Aeronautics and Astronautics (NUAA), and then worked at the College of Information Science and Technology of NUAA as an assistant lecturer. There she received a Ph.D. degree in computer science in 2009. Since 2010, she has become an Associate Professor for the College of Computer Science and Technology of NUAA.

Her research interests focus on neural computing, pattern recognition and machine learning.