Knowl Inf Syst
DOI 10.1007/s10115-017-1131-4

REGULAR PAPER

A novel ensemble-based wrapper method for feature selection using extreme learning machine and genetic algorithm

Xiaowei Xue[1] · **Min Yao**[1] · **Zhaohui Wu**[1]

Received: 10 February 2016 / Revised: 16 March 2017 / Accepted: 8 November 2017

Abstract This paper presents a novel wrapper feature selection algorithm for classification problems, namely hybrid genetic algorithm (GA)- and extreme learning machine (ELM)-based feature selection algorithm (HGEFS). It utilizes GA to wrap ELM to search for the optimum subsets in the huge feature space, and then, a set of subsets are selected to make ensemble to improve the final prediction accuracy. To prevent GA from being trapped in the local optimum, we propose a novel and efficient mechanism specifically designed for feature selection problems to maintain GA's diversity. To measure each subset's quality fairly and efficiently, we adopt a modified ELM called error-minimized extreme learning machine (EM-ELM) which automatically determines an appropriate network architecture for each feature subsets. Moreover, EM-ELM has good generalization ability and extreme learning speed which allows us to perform wrapper feature selection processes in an affordable time. In other words, we simultaneously optimize feature subset and classifiers' parameters. After finishing the search process of GA, to further promote the prediction accuracy and get a stable result, we select a set of EM-ELMs from the obtained population to make the final ensemble according to a specific ranking and selecting strategy. To verify the performance of HGEFS, empirical comparisons are carried out on different feature selection methods and HGEFS with benchmark datasets. The results reveal that HGEFS is a useful method for feature selection problems and always outperforms other algorithms in comparison.

Keywords Feature selection · Genetic algorithm · Extreme learning machine · Ensemble learning

✉ Min Yao
myao@zju.edu.cn

Xiaowei Xue
xwxue@zju.edu.cn

Zhaohui Wu
wzh@zju.edu.cn

[1] School of Computer Science and Technology, Zhejiang University, Hangzhou 310007, Zhejiang, China

Published online: 17 November 2017

1 Introduction

Feature selection is an important issue in many fields such as machine learning, data mining and pattern recognition [8,24,26]. In real-world problems, the target concept is always represented by a set of features which are often unknown a priori in related problems. To better represent the domain, many candidate features are introduced. Unfortunately, many of these candidate features are always irrelevant or redundant. These unnecessary features make it more difficult to capture the latent patterns in data by increasing the search space size or bringing more bad perturbations. Meanwhile, with the rapid advance of computer technologies, the proliferation of large and high-dimensional data sets within many domains poses unprecedented challenges to the traditional methods [4,36]. In this case, more and more researchers and practitioners are realizing that using feature selection methods to remove the redundant and irrelevant features is an important or even indispensable component to their systems.

Till now, researchers have proposed a plenitude of different feature selection methods, which in general can be divided into three categories: filter [13], wrapper [26] and embedded methods [32,33]. In filter methods, feature selection is performed as a pre-processing step in which features are scored and ranked based on some predefined measures. In particular, such predefined measures are independent of the actual generalization performance of the learning algorithms. So there is no guarantee that the selected features can improve the performance of the learning algorithms. On the contrary, the wrapper methods employ a search strategy to explore the combinatorial space of feature subsets and wrap a learning algorithm to evaluate the subsets. Since each subset needs a learning algorithm to measure their qualities, wrapper methods are generally more computationally intensive than filter methods but more effective. Different from filter and wrapper methods, embedded methods incorporate feature selection as part of the process in building a specific model. Some examples of the embedded methods are decision tree learner, such as ID3 [32] and C4.5 [33].

In this paper, we focus on the wrapper methods for feature selection problems. For a wrapper method, its main components are search strategies and learning algorithms. The search strategies in wrapper model can be classified into three groups: complete, heuristic and random search. The complete search exhausts all possible subsets and finds the optimal one, which is impractical for problems due to the large amount of computational effort [31]. Unlike the time-consuming complete search, heuristic search strategies trade off the optimality for the search efficiency. Sequential backward selection (SBS) and sequential forward selection (SFS) are two most commonly used wrapper methods [24]. However, these two approaches have a monotonic assumption that an added feature can no longer be removed and a removed feature can no longer be added, which makes it prone to get stuck in local minima. Different from these two search strategies, the random search always uses evolutionary methods as their well-known global search ability. Till now several evolutionary algorithms, such as genetic algorithms (GAs) [9,20], ant colony optimization (ACO) [42], particle swarm optimization (PSO) [29] and simulated annealing (SA), have been applied as wrappers. In these methods, through iterations, numerous heuristically selected subsets are evaluated by the performance of classifiers. Compared with the deterministic algorithms, evolutionary search methods like GA can be more capable of avoiding getting stuck in local optima and can find small feature subsets as they can effectively capture feature redundancy and interaction without the monotonic assumption. However, on the downside, the wrapper methods based on random search are a very computationally demanding task.

On the other hand, a well-chosen classifier also plays a significant role in designing a wrapper method as the process of feature selection is tied to the performance of the specific classifier. Due to the characteristic of wrapper methods, several issues should be taken into consideration for choosing an appropriate classification method. Firstly, as the classifier is used to predict the selected subset, it should have good prediction accuracy and good generalization ability. Secondly, for each subset, we need to train a classifier to measure its quality. Therefore, the training speed is another important issue that we should consider. Thirdly, the different subsets' input features may have a very big difference. In this case, even carefully chosen-fixed parameters of classifiers are not able to be appropriate for all the feature subsets to get best prediction accuracy. To counter this problem, it is better to automatically determine the parameters of the classifier for each different feature subset to avoid to bring biases when judging the quality of these subsets. In the previous works, many classification methods have been applied in wrapper methods, such as SVM [14], KNN [27], logistic regression [35], decision tree [33], Naive Bayes [3], etc. However, they cannot simultaneously consider all the issues mentioned above.

Taking full consideration of the issues mentioned above, we propose a novel hybrid wrapper method (HGEFS) using a genetic algorithm to wrap extreme learning machine for ensemble. Note that our primary focus is on obtaining a better overall classification performance in an affordable time. As a wrapper method, HGEFS is also very time-consuming so that it is just suitable for dealing with the datasets with a small or medium number of features. In HGEFS, the objective of the GA is to combine the search for optimum chromosome choices with that of finding an optimum classifier for each choice. And a novel strategy called extinction and immigration strategy (EI strategy) is proposed. It is specially designed for feature selection problems to improve the diversity of GA to better address the premature convergence problem and help GA avoid being trapped in local optimum points. For the classifier, HGEFS adopts a modified extreme learning machine called EM-ELM, which has high prediction accuracies and a very fast training speed that would significantly reduce the wrapper method's computational time and allow us to perform wrapper feature selection process in an affordable time. Moreover, EM-ELM can automatically determine the parameters to provide fairer judgments for different subsets. In addition, we utilize the ensemble mechanisms to further promote the stability and performance of HGEFS. In our method, after finishing the evolution process of GA, we render the EM-ELMs in the final population for the final ensemble tailored to the generalization properties of ELM with a specific special ranking strategy. At last, we aggregate the outputs of several EM-ELMs to produce the final results.

The main contributions of this article are summarized as follows:

- Proposing a novel wrapper methods using GA and EM-ELM;
- Proposing a novel strategy specially designed for feature selection problems to better address the premature convergence problem and help GA to avoid being trapped in local optimum points in which we make full use of the information gained in the iterations;
- Using a ranking method based on the generalization theory of ELM to select a set of EM-ELM for ensemble to further improve the final accuracy and stability;
- Conducting experiments on various datasets to demonstrate the effectiveness of HGEFS.

The rest of this paper is organized as follows. Section 2 presents the background information, namely the related works and a brief overview of genetic algorithm and EM-ELM. The detail of our proposed approach is shown in Sect. 3. In Sect. 4, the experimental results are given, and finally, the paper is concluded in Sect. 5.

2 Background

This section reviews typical related work on wrapper feature selection methods and provides background about genetic algorithm and extreme learning machine.

2.1 Related work

In this subsection, we briefly review the related works. Over the years, many evolutionary method-based feature selection approaches have been proposed. Nevertheless, most of them just focused on improving the searching efficiency while ignoring the role of classifiers played in the methods. For example, in [39], Yang et al. divided the chromosome into several segments according to the number of feature groups to obtain strong searching ability at the beginning of the evolution. And [28] focused on the strategies of generating the initial population of a genetic algorithm. These works just simply tried to improve the performance of GA. On the other hand, some works tried to use the information gained from filter methods to improve the quality of each individual in GA. For example, in [23,43], they all combined GA with local search. In the local search, they used different filter methods to add or delete features in each subset. Note that the information they used has no correlations with the final classifiers. While in our method, the prior knowledge used in EI strategy is generated based on the search process of GA and the performance of EM-ELM, which is germane to EM-ELM.

There are also some works trying to optimize the classifiers' parameters in the feature selection process. Huang and Wang [20] encoded two parameters of SVM and the subsets into a chromosome of GA. And [19] used PSO to simultaneously optimize the input feature subset selection and the SVM's parameters too. Compared with ELM, there are always two parameters in SVM that need to be optimized for each subset. It is not as efficient as HGEFS since ELM just has one parameter needed to be optimized, and meanwhile, we do not need to encode such parameter into the chromosome of GA. It is worth noting that in many previous works as mentioned above, SVM is always selected to be the classifier as many pilot studies comparing the use of popular Naive Bayes algorithm, logistic regression and C4.5 decision trees confirmed SVM's good prediction accuracy and good generalization ability. However, for wrapper feature selection problem, one obstacle of SVM is its relatively slow learning speed, and such defect would be obvious when the datasets are pretty large. Although there are also some linear-version SVMs such LS-SVM [21], their speed is much less than the original SVM. But when compared with ELM, their training speeds are still higher than ELM and they have more parameters that need to be optimized. Besides, SVM is a binary classifier and more SVMs need to be built for multi-class problems. On the contrary, ELMs provide an elegant and unified model for binary classification, multi-class classification and regression. Till now, there are also some works using ELM as the classifier for wrapper feature selection problems. For example, in [1], Alexandre et al. used ELM as the classifier and constrained the size of each chromosome. However, compared with HGEFS, they do not automatically determine the number of hidden nodes and just randomly delete or add features to fix the size of subset without prior knowledge. And they do not take full consideration of ELM's characteristic for feature selection problems.

In summary, different from the methods mentioned above, we optimize HGEFS as a whole around ELM through improving the efficiency of search strategy, providing a fairer evaluation for each subset and using ensemble mechanism to improve the method's performance and stability. And all the optimization methods used in HGEFS are germane to the ELM to yield a better prediction accuracy.

2.2 Genetic algorithm

Genetic algorithm [18] mimics the process of natural evolution that is widely used to generate useful solutions to optimization and search problems. It belongs to a large class of evolutionary algorithms and uses mutation, crossover and selection operators. To solve feature problems in this paper, each feature subset is mapped to a chromosome, namely individual. And GA firstly initials a population of individuals where each individual is generated by randomly selecting a different subset of features. Then, new candidate individuals are produced by using crossover and mutation on the chromosome, where crossover varies the programming of the chromosomes while mutation only alters particular genes to maintain genetic diversity of chromosomes. After reproducing a certain number of new individuals, a subset of individuals is selected to survive for the next generation. This process of producing new individuals and selecting a subset continues a number of time, known as the number of generations. After a predefined number of generations, the evolution process stops and the fittest individuals make up the final population.

For feature selection problems, it is natural and efficient to use the binary coding to encode the feature subset. In GA, the original crossover and mutation operators are designed based on binary coding. When compared with other evolutionary algorithms such as PSO, GA shows powerful search capacity with binary coding. However, PSO always uses the real number for coding. Of course, there are some binary PSO. However, binary PSO has potential limitations, such as the position of a particle in binary PSO is updated solely based on the velocity, while the position in standard PSO is updated based on both the velocity and current position. On the other hand, the main advantage of GA is that more strategies can be adopted together to improve its performance. In our method, the EI strategy is designed to be combined with GA to improve the performance of HGEFS and the experiments proved its efficiency.

2.3 Error-minimized extreme learning machine

Extreme learning machine is a novel efficient single-hidden-layer feedforward (SLFN) neural network proposed by Huang et al. [22]. Compared with the traditional neural network, ELM is a tuning-free algorithm with an extreme learning speed by randomly generating the input weights and the hidden biases instead of iteratively adjusting learning parameters. By adopting the squared loss of prediction error, training output weights turns into a least squares problem which can be solved effectively. Compared to gradient based algorithms, ELMs are much efficient and usually lead to better performance, and recent papers show that the predicting accuracy achieved by ELMs is comparable with or even higher than that of SVMs.

In this subsection, we briefly introduce the original ELM. For given N arbitrary samples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N}$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbf{R}^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbf{R}^m$. The ELM using an active function g(x) with K hidden nodes can be modeled as:

$$\sum_{j=1}^{K} \boldsymbol{\beta}_j \boldsymbol{g}(\boldsymbol{\omega}_j, \mathbf{b}_j, \mathbf{x}_i) = \mathbf{t}_i, \quad i = 1, 2, \ldots, N \tag{1}$$

where $\boldsymbol{\omega}_j = [\omega_{j1}, \omega_{j2}, \ldots, \omega_{jn}]^T$ is the input weight connecting the jth hidden node; $\mathbf{b}_j = [b_{j1}, b_{j2}, \ldots, b_{jK}]^T$ is the bias of the jth hidden node; $\boldsymbol{\beta}_j = [\beta_{j1}, \beta_{j2}, \ldots, \beta_{jm}]^T$ is the output weight connecting the jth hidden node to the output nodes; and the $\boldsymbol{g}(\omega_j, \mathbf{b}_j, \mathbf{x})$ denotes the output of the jth hidden node with respect to the input $\mathbf{x}$. In this case, the ELM can approximate the N samples with zero error if there exit $\boldsymbol{\beta}_j, \boldsymbol{\omega}_j$ and $\boldsymbol{b}_j$.

And we can rewrite the N equations in Eq. (1) in a matrix form:

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

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{g}(\omega_1, \mathbf{b}_1, \mathbf{x}_1) & \dots & \mathbf{g}(\omega_K, \mathbf{b}_K, \mathbf{x}_1) \\ \vdots & \dots & \vdots \\ \mathbf{g}(\omega_1, \mathbf{b}_1, \mathbf{x}_N) & \dots & \mathbf{g}(\omega_K, \mathbf{b}_K, \mathbf{x}_N) \end{bmatrix}_{N\times K}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \mathbf{b}_1^T \\ \vdots \\ \mathbf{b}_K^T \end{bmatrix}_{K\times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N\times m}$$

Here, $\mathbf{H}$ is called the hidden layer output matrix of the network. The column of $\mathbf{H}$ is the ith hidden node's output vector with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and the jth row of $\mathbf{H}$ is the output vector of the hidden layer with respect to $\mathbf{x}_j$.

After generating the $\boldsymbol{\omega}$ and $\boldsymbol{b}$ randomly, the $\boldsymbol{\beta}$ can be calculated as follows:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{3}$$

where $\boldsymbol{H}^{\dagger}$ is the Moore–Penrose generalized inverse of matrix $\boldsymbol{H}$.

During the whole process to train an ELM, the only parameter that needs to be preset is the number of hidden nodes, which would effect the generalization performance. To automatically determine the number of hidden nodes, we use a modified ELM called the error-minimized extreme learning machine (EM-ELM) [10]. The EM-ELM not only inherits the good prosperities of the original ELM such as good generalization performance and high learning speed but also can automatically determine the architecture of network to adapt itself to different inputs.

In EM-ELM, the hidden nodes can be added to the network through iterative computations from an initial number of hidden nodes until the termination condition is reached. EM-ELM should preset the initial number of hidden nodes, the maximum number of hidden nodes and the expected learning accuracy ε. From the initial hidden nodes, if the expected accuracy is generated and the number of hidden nodes is more than the maximum number, new hidden nodes are stopped to add into ELM. EM-ELM reduces the computation complexity by only updating weights incrementally each time. Namely, the new output weights are updated on the basis of the previous output weights.

The output weights can be fast updated recursive way as:

$$\boldsymbol{D}_j = \left(\left(\boldsymbol{I} - \boldsymbol{H}_j \boldsymbol{H}_j^{\dagger}\right)\boldsymbol{\delta H}_j\right)^{\dagger} \tag{4}$$

$$\boldsymbol{U}_j = \boldsymbol{H}_j^{\dagger}\left(\boldsymbol{I} - \boldsymbol{\delta H}_j^{\boldsymbol{T}}\boldsymbol{D}_j\right) \tag{5}$$

$$\boldsymbol{\beta}_{j+1} = \boldsymbol{H}_{\boldsymbol{j+1}}^{\dagger}\boldsymbol{T} = \begin{bmatrix} \boldsymbol{U}_j \\ \boldsymbol{D}_j \end{bmatrix}\boldsymbol{T} \tag{6}$$

where $\boldsymbol{H}_j$ is the hidden-layer output matrix in the jth iteration; $\boldsymbol{\delta H}_j$ is the output matrix of new added hidden nodes and $\boldsymbol{H}_{j+1} = [\boldsymbol{H}_j + \boldsymbol{\delta H}_j]$. The detail of EM-ELM can be seen in [10].

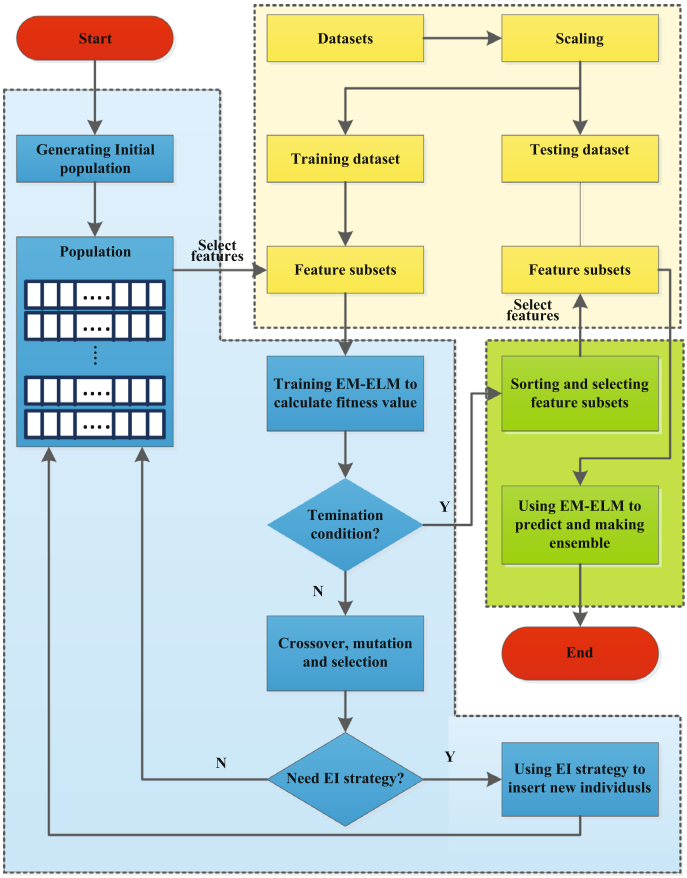


Fig. 1 The flowchart of our proposed method

3 Our proposed method

In this section, we introduce the proposed method for classification problems, which is depicted in Fig. 1. The blue part of the figure is the searching process of GA, the yellow part is the data processing, and the green part is the ensemble process. At first, the GA's population is initialized randomly in which each candidate feature subset is encoded as a chromosome. Subsequently, we trim the dataset based on the chromosome and train different EM-ELM neural networks to calculate fitness value for each candidate feature subset. Then, a new population is generated by using genetic operators. After a few generations, we utilize the extinction and immigration strategy to insert new individuals to improve the diversity of the whole population. This process repeats until the stopping conditions are satisfied. The genetic algorithm here is designed to maximize classification accuracy and minimize the size of feature subsets. After the searching process, a set of candidate subsets are selected based on their fitness value first. Then, a smaller set is chosen according to their corresponding networks' norm of output weights. Finally, the samples of the test dataset are predicted by all the selected EM-ELM and the final results are calculated by majority voting.

In the rest of this section, the detail of our proposed method is specified, including the coding for feature subsets, the fitness function for individuals, the crossover and mutation operators, the extinction and immigration strategy, and the ensemble mechanism.

3.1 Chromosome encoding

For the feature selection problem, it is natural for us to encode the possible feature subset solutions with a binary string of length equal to the feature set size where '1' indicates the presence of the feature and '0' otherwise:

$$\boldsymbol{\theta} = [\theta(1), \theta(2), \ldots, \theta(n)] \tag{7}$$

where $\boldsymbol{\theta}$ is the chromosome, $\theta(i) \in \{1, 0\}$ and n is the number of the whole features. For example, a chromosome 1010110 indicates that the first, third, fifth and sixth features remained to be the feature subset.

3.2 Fitness evaluation

In this paper, the genetic algorithm is designed to optimize two objectives: maximize classification accuracy of the feature subset and minimize the number of the selected features. In the evolution process of GA, the subsets with higher prediction accuracies are more likely to survive to the next generation. In other words, GA tries to maximize the subsets' corresponding accuracies. In order to minimize the feature subset to satisfy the given subset size requirement, we add a penalty term to define the fitness function as follows:

$$\begin{aligned} F(\boldsymbol{\theta}) &= accuracy(\boldsymbol{\theta}) - \lambda|n - m| \\ &= \sqrt{\frac{\sum_{j=1}^{N} \| \sum_{i=1}^{K} \beta_i g(\omega_i \cdot x_j + b_i) - t_j \|_2^2}{N}} - \lambda|n - m| \end{aligned} \tag{8}$$

where N is the number of validation samples and K is the number of hidden nodes; $\boldsymbol{\theta}$ is each corresponding feature subset, n is the number of the whole features, and m is the given number of features that we need. λ is a penalty coefficient to achieve a tradeoff between the accuracy and the size of feature subset obtained. When calculating the term $accuracy(\boldsymbol{\theta})$, the training dataset is partitioned into internal training dataset to train related EM-ELMs and validation set to evaluate the prediction accuracies.

3.3 Crossover, mutation and selection

During each successive generation, a proportion of the existing population is selected to breed a new generation. We require the fittest individuals have greater chances of survival than weaker ones. Accordingly, we adopt the rank-based roulette wheel selection scheme. And to guarantee the fast convergence ability, an elitism strategy is also used so that the best 10% of the individuals in the current population can remain directly for the next generation.

To breed new generations, many pairs of selected parent individuals reproduce child individuals through crossover and mutation operators. The purpose of crossover is to exchange information between the selected individuals, and the mutation operator introduces new genes into the population and retains genetic diversity. In our method, we use the two-point crossover operator and the simple mutation operator.

3.4 Extinction and immigration strategy

After evolving several generations, many chromosomes in the population might become similar. The whole population loses the diversity, which may lead HGEFS to be trapped in local optima. To prevent this problem, Yao and Sethares [40] proposed an extinction and immigration strategy. The common and simple way is just to replace some existing individuals with some new randomly generated individuals to provide a perturbation to help GA to escape from the local optimum. The new generated individuals can improve, to some extent, the diversity of the whole population. However, it takes no consideration of the characteristic of feature selection problems and is prone to be inefficient. Firstly, as the fitness function contains a penalty term to constrain the feature numbers, new generated individuals may lose competitiveness in the population without considering the requirement of feature numbers. Secondly, the randomly generated individuals always have lower fitness so that we need to improve their qualities to help some of them survive to the next generation.

Considering the problems above, we propose a new extinction and immigration strategy (EI strategy) specifically designed for feature selection problems, which can not only improve the diversity of the population but also ensure requirement of feature number. Using the GA as the search mechanism, a large number of subsets are evaluated by the learning algorithm after some generations. Through a statistical analysis, we find that some features are always involved in good subsets, while some features barely appear in the subsets with high qualities. We intuitively think that the features with higher probabilities of appearing in good subsets have significant correlation with the high prediction accuracy. Besides, another issue should be taken into consideration. Note that, for example, in a population, the best subset's quality, namely the prediction accuracy (without the penalty part), is 90%, while the worst one's fitness is just 70%. We cannot simply treat such two different combinations of features in the same way. Based on these considerations, both the frequency of occurrence of features and corresponding prediction accuracy are adopted to define a score to measure the qualities of each feature.

In each generation, we select some top chromosomes with higher prediction accuracy values and multiply them with their corresponding prediction accuracies. Note that these selected chromosomes come from 'child' individuals rather than 'parent' individuals in each generation. The goal here is to avoid calculating some outstanding individuals repeatedly as some outstanding individuals would always remain in the population for the next generation. To formalize this idea, we define the score for each feature as follows:

$$S(f_i) = \sum_{p=1}^{n1} \sum_{q=1}^{n2} \theta_q^p(i) * accuracy\left(\theta_q^p\right), \quad i \in \{1, \ldots, n\} \tag{9}$$

where f_i is the ith feature, $n1$ is the number of generations when we calculate the scores and $n2$ is the number of chromosomes (we select chromosomes whose corresponding accuracies rank top $n2$); θ_q^p is the qth chromosome in the pth generation, $\theta_q^p(i)$ is the ith entry of θ_q^p and the $accuracy(\theta_q^p)$ is prediction accuracy of θ_q^p.

The rest of work is just how to use these scores as a kind of prior knowledge to improve the subsets' qualities. We should consider not only the scores of each feature but also the size of subsets. Compared with inserting randomly generated individuals, we tailor the randomly generated chromosomes to a fixed size, which is ascertained according the average feature number of each subset in the whole population. By doing so, the inserted chromosomes would not lose competitiveness due to the penalty of feature numbers. To get the inserted individuals, firstly, we randomly generate a chromosome $\boldsymbol{\theta}$, we define X and Y as the sets of

selected features and excluded features encoded in $\boldsymbol{\theta}$, respectively. Here, $|X| + |Y| = n$. Our objective function is simply defined as:

$$J(X) = max_X \sum_{x \in X} S(x), \quad s.t. \ |X| = N_{ave} \tag{10}$$

where N_{avg} is determined by the average subsets' feature number in the corresponding generation and $|X|$ is the number of features in $\boldsymbol{\theta}$. We also define two EI strategy, namely "ADD" operator and "DEL" operator. If the $|X|$ is greater than N_{avg}, we would utilize the "DEL" operator while $|X|$ is less than N_{avg}, "ADD" operator is adopted. The "ADD" operator searches for the feature y in Y and add y to the X until $|X|$ is equal to N_{avg}, i.e., $y = arg\ max_{y \in Y} J(X \bigcup \{y\})$. In the same way, "DEL" operator searches for the feature x in X and move x to the Y until $|X|$ is equal to N_{avg}, i.e., $x = arg\ max_{x \in X} J(X - \{x\})$. In this case, all the inserted individuals have the same number of features. As the individuals are randomly generated, the tailored individuals would still be able to introduce new feature combinations to improve the whole diversity of the population.

3.5 Ensemble strategy

In this paper, we focus on improving the prediction accuracy and getting a robust result. In this case, instead of spending a more computational time to search for the optimal subset, we combine the outputs of several near-optimal feature subsets to reduce the risk of choosing an unstable subset and give a better approximation to the optimal subset, namely feature selection ensemble.

As we know, a good ensemble is one where the base classifiers in the ensemble are accurate and have diversities in their predictions. It is obvious that combining several identical predictors produces no gain. In our method, by varying the feature subsets which generate the base classifiers to view a problem from different perspectives, it is possible to promote diversity and produce base classifiers that tend to err in different parts of the instance space. Next, we need to guarantee that the base classifiers have accurate predictions. However, small fitness value just indicates a small training error and cannot guarantee the generalization performance. According to the generalization theory of ELM [2], the ELM network tends to show better generalization with not only small training error but also small norm of output weights. In HGEFS, after performing the GA's search, the individuals with high fitness value are remained to comprise the final population. Selecting the base classifiers from the final population can satisfy the requirement of small training error. To further satisfy ELM's generalization theory, we need select the ones with smaller norm of output weights further. Based on such consideration, we first sort the individuals according to their fitness value and select $2M$ (where M is the number of classifiers for ensemble) fittest individuals. Then, M subsets with a smaller norm of output weights are chosen from these $2M$ subsets to make up the final ensemble model.

Finally, we use a simple and effective technique called majority voting for the ensemble model. In the ensemble phrase, for each testing sample $\boldsymbol{x}^{test}$, we can get M prediction results obtained by the remaining independent EM-ELMs. Then, a corresponding vector $\boldsymbol{L}_{\boldsymbol{x}^{test}} \in \boldsymbol{R}^C$ (C is the number of class labels) with dimension equal to the number of class labels is utilized to store all these M results of $\boldsymbol{x}^{test}$. If the mth ($m \in \{1, ..., M\}$) ELM's prediction is the ith class label, the value of corresponding entry i in the vector $\boldsymbol{L}_{\boldsymbol{x}^{test}}$ would be increased by one, that is

$$\boldsymbol{L}_{\boldsymbol{x}^{test}}(i) = \boldsymbol{L}_{\boldsymbol{x}^{test}}(i) + 1 \tag{11}$$

After all the M results are assigned to $\boldsymbol{L}_{\boldsymbol{x}^{test}}$, the final decision of the EM-ELM ensemble $f_{ens}(\boldsymbol{x}^{test})$ for the given test sample $\boldsymbol{x}^{test}$ due to the majority voting is determined by

$$f_{ens}\left(\boldsymbol{x}^{test}\right) = arg\ max_{i\in\{1,\ldots C\}}\boldsymbol{L}_{\boldsymbol{x}^{test}}(i) \tag{12}$$

In [16], Hansen et al. shows that the ensemble of neural network by majority voting has better performance than a single classifier. And the experimental results in [6] also support such conclusion. As analysis in [6], for a C-label classification problem, we assume that the true label for the test sample $\boldsymbol{x}^{test}$ is c. Given a trained EM-ELM, the probability of correctly predicting the test sample $\boldsymbol{x}^{test}$ is $p(c|\boldsymbol{x}^{test})$. If the following inequality holds

$$p\left(c|\boldsymbol{x}^{test}\right) > max\left\{p\left(i|\boldsymbol{x}^{test}\right)\right\}_{i\in\{1,\ldots C\}\ and\ i\neq c} \tag{13}$$

where $p(i|\boldsymbol{x}^{test})$ is the probability that EM-ELM classifies $\boldsymbol{x}^{test}$ to category i that is different from the class c, and then, with a sufficiently large independent training number M, the ensemble of EM-ELM is able to correctly classify $\boldsymbol{x}^{test}$ with probability one. Since EM-ELM has a good generalization performance, Eq. (13) would always hold. However, it is not practical to train sufficiently large number of EM-ELM to make up for ensemble. As suggested in [6], for most practical applications, it suffices to ensemble 5 to 35 ELMs. In light of this, in our experiment, we set the M as 10.

Algorithm 1 illustrates the process conformed in our feature selection method.

4 Performance verification

In this section, a series of experiments have been carried out to study the proposed method and verify its effectiveness. After a brief introduction of datasets and experiment environment in Sect. 4.1, we investigate some important characteristics of HGEFS including the search ability of genetic algorithm, the efficiency of the novel EI strategy and the effect of automatic determination of neural networks in Sect. 4.2. At last, in Sect. 4.3, we provide a detailed comparison of HGEFS with other famous feature selection methods on different real-world datasets to verify HGEFS's applicability and effectiveness for feature selection problems.

4.1 The benchmark datasets and experimental setup

In our simulation experiments, 10 benchmark datasets were adopted to validate the proposed method's effectiveness. Eight of them are available from the UCI Machine Repository, and two microarray datasets, i.e., colon and DLBCL. For convenience, we cut the top two features of Musk dataset. In Sect. 4.2, we select two representative datasets from these 10 datasets, i.e., Sonar and Musk to study HGEFS's characteristics. Table 1 summarizes some general information of these datasets. For missing values, we replaced them with the most frequently used values and means for nominal and numeric features, respectively. Note that these datasets differ greatly in the instance size (range from 62 to 3196) and the number of features (range from 18 to 7129). All these datasets are widely used for evaluating learning algorithm and hence can provide a comprehensive testing for feature selection problems.

Moreover, our implementations were carried out on the Matlab R2014a development environment and Weka platform with open-source and default parameters (Waikato environment for knowledge analysis). All experiments were running on Intel 3.0 GHz CPU and 8G RAM. In EM-ELM, the activation function we use is the sigmoid function $g(x) = \frac{1}{1+e^{\omega x+b}}$. There are some parameters else in HGEFS which need to be specified. In our method, we employed

Algorithm 1 HGEFS.

Input:

a training data, a validation data, a testing data, the expected learning accuracy ε, the maximum number of hidden nodes l_{max}, the minimum number of hidden nodes l_0, number of iterations I, number of networks for ensemble M, number of selected individuals for calculating scores $n2$

Training phase:

(1) Randomly generate individuals for population
(2) Evaluate fitness for each individual by EM-ELM, all the parameters of EM-ELM for each individual are preserved.
(3) For $i = 1, 2, ..., I$
  (i) Breed new individuals through crossover and mutation operators as new population.
  (ii) Calculate each new individual's fitness value
  (iii) Select the top $n2$ new individuals to calculate scores
  (iv) if $i \gg \frac{I}{2}$ and $i\%10 == 0$
    (iva) Utilize the Extinction and Immigration Strategy
    else
    (ivb) goto step (v)
  (v) Select individuals for the next generation from the exiting populations
  (vi) $k = k + 1$
(4) End for
(5) Sort the individuals in the population based on fitness value and select the first $2M$ individuals. Then, sort the $2M$ individuals in the descent ordering of the norm of output weights $\|\beta\|$. The top M individuals remain for making up ensemble.

Ensemble phase:

(1) For $j = 1, 2, ..., M$
  (i) Given a testing sample $(\mathbf{x}, \mathbf{t})$. Use each individual's corresponding EM-ELM to predict the result for this sample.
(2) End for
(3) $f_{ens}(\boldsymbol{x}) = are\ max_{i \in [1,...C]} L_{\boldsymbol{x}}(i)$

a population size of 40 and a stopping criterion of 200 generations for UCI datasets. On the colon dataset, the population size is increased to 100 and the stopping criterion is increased to 500. And the crossover and mutation probabilities are 0.9 and 0.1, respectively. For EI strategy, we select the top 15 individuals from new generated generations to calculate scores. And the number of neural networks for ensemble is 10 for HGEFS. It is worth noting that the configurations of the parameter used here have been investigated empirically for the data sets considered.

4.2 Performance of the proposed method

4.2.1 Study on the search ability of genetic algorithm

In this subsection, we utilize two datasets to examine the effectiveness of GA's search capability. To graphically illustrate the progress of genetic algorithm as it searches for optimal solutions, we take generations as the horizontal coordinate and the fitness values as the vertical coordinate. The processes of GA searching for optimal solutions for Sonar and Musk dataset are given in Fig. 2a, b. The blue line in Fig. 2 is the best fitness value in each generation, while the red line is the average of the whole fitness values.

Table 1 The descriptions of 10 datasets in our experiment

Datasets	# Feature	# Instance	# Class
Vehicle	18	846	4
WDBC	30	569	2
Ionosphere	34	351	2
Chess	36	3196	2
Sonar	60	258	2
Splice	61	3190	3
Musk	166	476	2
Arrhythmia	279	452	16
Colon	2000	62	2
DLBCL	7129	77	2

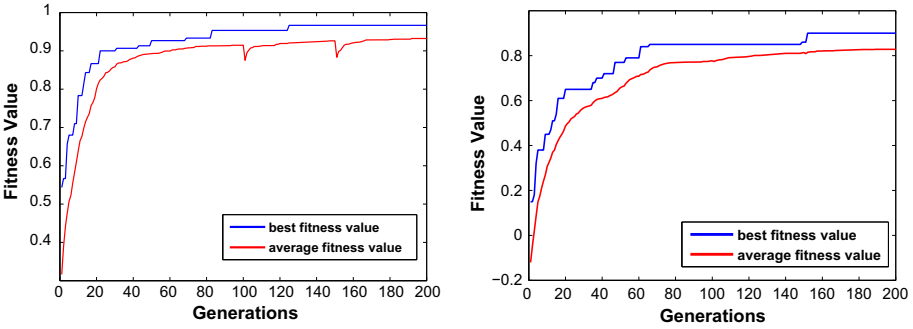


Fig. 2 The search process of GA. **a** Sonar dataset, **b** Musk dataset

The results in Fig. 2 illustrate the process of improvement of the global best individual and the average of fitness values as the number of generations increase. As you can see from the figure, GA has a powerful exploration ability, which is a gradual searching process that approaches the optimal solutions. In the early iteration, the convergence speed is pretty fast, and in about 60th generation the result is just a little worse than the final result in both experiments.

In the fitness function, we consider not only classification accuracies but also the number of features for each subset. As ELM has a very powerful generalization ability, it is, to some extent, not sensitive to a slight difference of the number of features. In this consideration, we add a penalty term to limit the number of subsets' features. To better understand the dynamic evolution of HGEFS, Fig. 3a, b shows the change of the average feature number of the whole subsets in each generation. It is worth noticing that we set m (the number of features we need) as 10 for Sonar dataset and 30 for Musk dataset. From the figures, we can see that HGEFS can effectively reduce the number of subsets' features.

4.2.2 Study on the extinction and immigration (EI) Strategy

For the EI strategy, the main purpose is to improve the populations' diversity by inserting new randomly generated individuals into the population. Compared with the common way, the new randomly generated individuals are tailored based on the scores for each feature

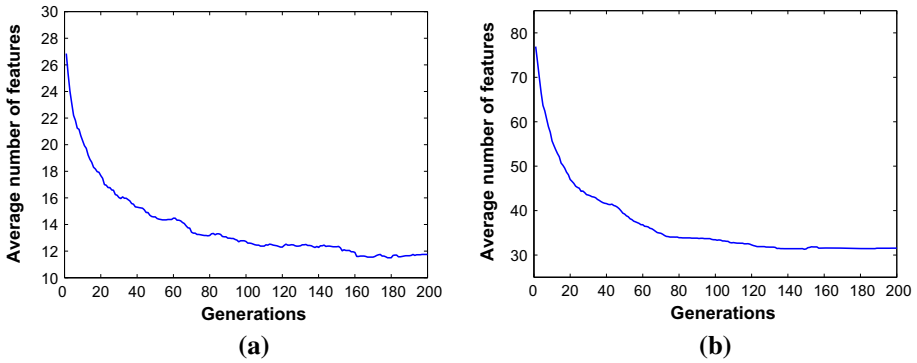


Fig. 3 The changes of average feature number in each generation. **a** Sonar dataset, **b** Musk dataset

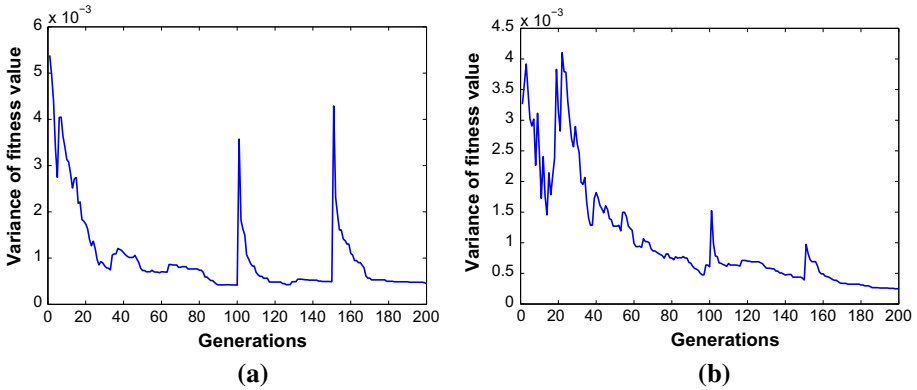


Fig. 4 The variance of subsets' fitness value. **a** Sonar dataset, **b** Musk dataset

to a fixed size according to the average feature number of subsets in the corresponding generation. We reduce the number of features for each subset, and meanwhile, we want to maintain these subsets' qualities or even improve them. To better understand the EI strategy, we systematically examine the changes of population's diversity, the qualities of new inserted individuals and the effectiveness of the 'prior' knowledge.

Firstly, we examine whether EI strategy can improve the diversity of the population. As we know, for GA the variance of the individual fitness can reflect the diversity of population in some degree. As shown in Fig. 4, the variance of subsets' accuracies decreases as the generations increase, meaning that the population tends to be homogeneous and lose the diversity. Nevertheless, we can see that in generation 100 and 150 the variance increases as we adopt the EI strategy. In particular, the improvement on Sonar dataset is pretty obvious. The improvement of variance is due to the insertion of new individuals, bringing some new genes to improve the diversity of the population.

Then, we utilize two datasets to compare the inserted individuals' qualities and the randomly generated individuals' qualities. The modification is twofold: the corresponding accuracies and number of features. Note that to be fair we compare their accuracies rather than their corresponding fitness values. Figure 5 shows the prediction accuracies of inserted subsets and randomly generated subsets. We can find that most of inserted individuals outperform randomly generated individuals in terms of prediction accuracies, meaning that our

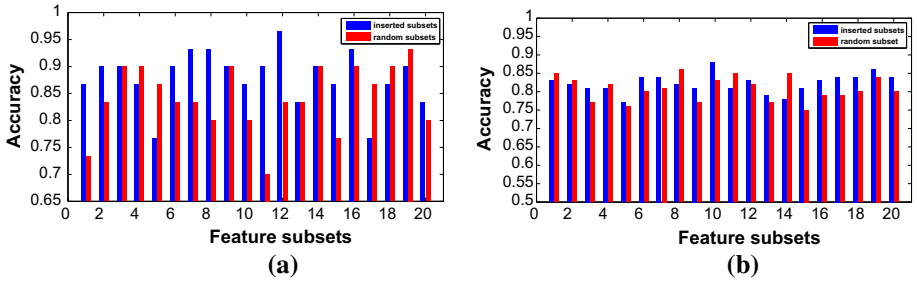


Fig. 5 The comparison of accuracies between inserted subsets and randomly generated subsets. **a** Sonar dataset, **b** Musk dataset

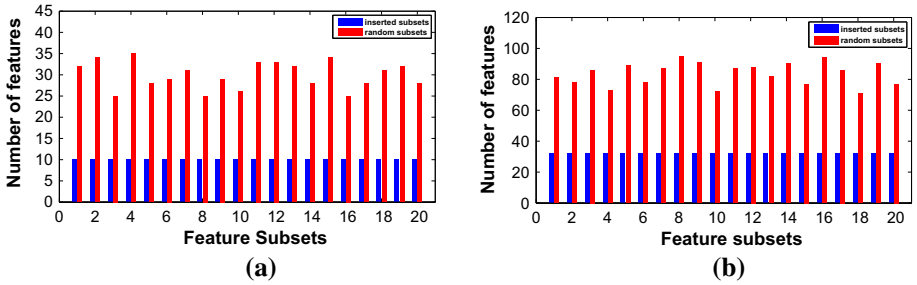


Fig. 6 The comparison of feature numbers between inserted subsets and randomly generated subsets. **a** Sonar dataset, **b** Musk dataset

EI strategy outperforms than the original EI strategy. In addition, the number of features in inserted individuals is significantly reduced through EI strategy's tailor as shown in Fig. 6, which would improve the inserted individuals' competitiveness. It is worth noticing that in HGEFS the fitness value of the randomly generated individuals would be much smaller than that of inserted individuals as the effect of the penalty term. In other words, the inserted individuals in our EI strategy have greater chance to survive in the next generation with better accuracies and smaller number of features. The results discussed above indicate that the EI strategy would help GA to get an appropriate balance between the genetic search and the population's diversity.

At last, the effectiveness of the 'prior' knowledge is also examined. In the process of evolution, the GA is guided toward a better accuracy and features that appear more frequently in the subsets which have higher accuracies may represent more significant ones for the class prediction. And it is natural for us to consider whether these scores can be regarded as a criterion to measure the qualities of features. To validate the scores' effectiveness, we select the best m features—the top m features in the descent ordering of scores—as a subset and compare such subset with other subsets selected by some famous filter methods. Here we choose five filter methods implemented on the Weka platform for comparison: ChiSquare [30], GainRatio [34], InfoGain [11], ReliefF [25] and SymmetricalUncert [41]. To be fair, we select the same amount of features for different methods: 10 features for Sonar dataset and 20 features for Musk dataset. All the results are conducted on the subsets by tenfold cross-validation, and the classifiers used here are Libsvm [7] for Sonar, Naive Bayes for Musk and EM-ELM for both datasets. All the experiments have the same preset parameters.

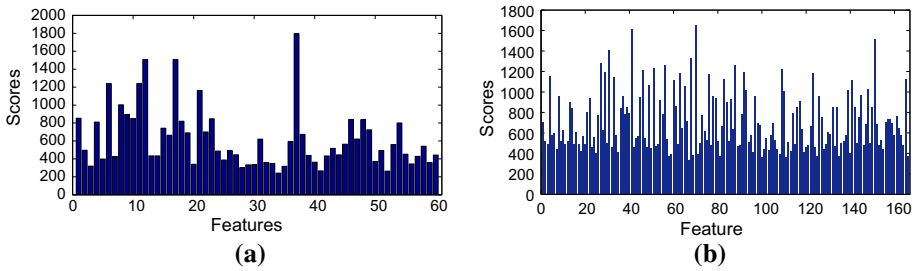


Fig. 7 The scores for each feature. **a** Sonar dataset, **b** Musk dataset

Figure 7 shows the scores for different features, and Table 2 presents the top 10 and 20 features selected by different methods and the corresponding accuracies. Since these criteria are very diverse and motivated by different theoretic arguments, they would produce different outcomes even conducted on the same dataset. Such phenomena are supported by the results in Table 2. Even so, there are still some features selected by almost all the methods in the Sonar dataset, namely #9, #10, #11, #12. But in the Musk dataset, the top 20 features of each method have a big difference. For the accuracy, the performance of EI score outperforms other methods in majority cases with three different classifiers. And we can also find that the performance of EI score with EM-ELM would always get a better result as the EI scores are germane to the EM-ELM. As is seen in experimental results, the EI score, in some degree, is an effective criterion to measure the qualities of features effectively as other four filter methods.

4.2.3 The effect of automatical determination of neural networks' architecture

In our proposed method, we take feature selection and classifier design into account simultaneously, using EM-ELM to automatically select an appropriate number of hidden nodes for each neural network during the feature selection process. It is, however, not clear the effects of architecture determination in the whole system of HGEFS. To observe such effect, here, we conduct a set of experiments. To simulate the feature selection process, we randomly select a set of subsets from the datasets and use ELM and EM-ELM to make predictions. For ELM, we test 25 and 35 hidden nodes on two datasets for all subsets while EM-ELM automatically selects the hidden node number. Table 3 presents the average results over 10 runs of tenfold cross-validation. As seen in Table 3, we can find that the results of ELM are almost inferior in all cases to that of EM-ELM. And for different subsets, their corresponding appropriate number of hidden nodes varies. In addition, by choosing an appropriate number of hidden nodes, EM-ELM can reduce the effect of inappropriate hidden node number so that we can measure feature subsets more fairly. For example, in Sonar dataset the subset with 30 features would have a much better quality when we use EM-ELM. And in Musk dataset, when comparing the qualities of subsets with 80 and 110 features, the subset with 110 features using ELM to measure would be chosen while through adjusting the hidden nodes number we can find that the subset with 80 features tends to be better. In light of this, EM-ELM can not only improve the prediction accuracy but also measure the feature subset more fairly. Therefore, EM-ELM is much more applicable than ELM for feature selection problems.

Table 2 The features selected by different methods and the corresponding accuracy

Dataset	Algorithm	Selected features	Accuracy (%)	Accuracy (EM-ELM) (%)
Sonar	EI Score	37, 12, 17, 11, 6, 21, 8, 9, 1, 10	75.17	77.25
	ChiSquare	11, 12, 9, 10, 13, 48, 49, 51, 47, 52	72.11	72.90
	GainRatio	11, 12, 9, 44, 13, 54, 10, 45, 47, 48	69.23	70.70
	InfoGain	11, 12, 9, 10, 13, 48, 49, 51, 47, 45	69.71	71.10
	ReliefF	12, 11, 10, 36, 9, 45, 48, 13, 49, 46	78.84	75.25
	SymmetricalUncert	11, 12, 9, 10, 13, 48, 49, 45, 44, 47	69.71	70.95
Musk	EI Score	70, 41, 151, 31, 68, 27, 88, 56, 51, 109 46, 29, 92, 63, 123, 76, 4, 33, 83, 165	72.47	77.26
	ChiSquare	90, 20, 164, 34, 160, 19, 113, 68, 128, 93 134, 56, 57, 100, 66, 1, 83, 146, 48, 40	72.26	72.32
	GainRatio	66, 110, 21, 127, 43, 156, 97, 41, 114, 112 111, 79, 32, 125, 48, 77, 10, 92, 126, 58	59.03	75.53
	InfoGain	20, 90, 34, 164, 113, 66, 68, 19, 134, 93 160, 48, 83, 1, 128, 146, 56, 57, 27, 108	71.84	73.32
	ReliefF	159, 94, 2, 122, 63, 160, 90, 127, 97, 95 32, 163, 67, 129, 37, 161, 82, 114, 41, 22	62.81	72.89
	SymmetricalUncert	66, 90, 68, 134, 48, 1, 83, 21, 113, 110 146, 160, 19, 34, 10, 8, 108, 49, 20, 127	67.85	70.43

Table 3 The performance comparisons with ELM, EM-ELM for feature selection problems

Datasets	# Feature	# Hidden nodes		Accuracy (%)	
		ELM	EM-ELM	ELM	EM-ELM
Sonar	10	25	32.5	67.5	70.5
	20	25	35.5	70.5	72
	30	25	49	71.5	75
	40	25	53	72	73.5
	50	25	46.5	74.5	75.5
	All	25	42	75	77.5
Musk	20	35	53	67.19	70.45
	50	35	65.5	70.17	72.02
	80	35	67	71.49	76.81
	110	35	76.5	75.96	76.38
	140	35	86	74.47	77.45
	All	35	79	74.13	77.74

4.3 Performance comparison on benchmark datasets

In this subsection, we compare the performance of HGEFS with different kinds of feature selection methods, namely four filter methods: correlation-based feature subset selection (CFS) [15], ReliefF [25], Gain Ratio [34], ChiSquare [30], two hybrid wrapper methods: PSO-SVM [12] and GA-ELM [1], two embedded methods: C4.5 [33] and SVM-RFE[14], and three ensemble feature selection methods: Attribute Bagging (AB) [5], Multi-View Adaboost (MVA) [37] and Random Subspacing Ensemble (RSE) [17]. Here, two embedded methods and PSO-SVM are all implemented in the Weka platform, while others were implemented in Matlab. And specifically, the four filter methods were firstly conducted in the Weka platform and then used EM-ELM to predict in Matlab. In the CFS, we use the sequential forward search (SFS). PSO-SVM has the same population size and generations as HGEFS. To be fair, AB, MVA and RSE all train 10 EM-ELMs and the mechanism of sample ensemble for determining the output value is the same as HGEFS. For the bagging method in AB and Adaboost method in MVA, 40% of the training data is used to resample to train EM-ELM and the number of iterations for AB is 10. Both AB and RSE use randomly chosen subspaces of the original input space and for the colon dataset and DLBCL dataset, to improve the performance of AB and RSE, we first use Gain Ratio to select the top 300 features and then randomly select from these 300 features to make RSE ensemble. In the MVA, we use the different subsets selected by two filter methods (Gain Ratio and ChiSquare) to make ensemble and the number of iterations for each subset is 5. The results are mainly compared in terms of prediction accuracies. For the high-dimensional datasets (colon and DLBCL), their searching spaces are very huge. Directly using HGEFS to search for the subsets with good qualities and a small number of features becomes very computationally demanding and its run time can be prohibitive. Just as many other wrapper methods, to deal with high-dimensional datasets, we can adopt a two-phase hybrid combination of filtering and wrapping to reduce the time required for training as many wrapper methods utilize. In our experiments, we firstly use Gain Ratio to rank the features and select the top 500 features for HGEFS.

In our experiments, datasets were firstly fed into different feature selectors, which may produce different feature subsets. For filter feature selection methods, the top t features with

Table 4 Performance comparisons with ReliefF, Gain Ratio, ChiSquare, CFS-SFS and C4.5

Datasets	Unselected	ReliefF	Gain Ratio	ChiSquare	CFS-SFS	C4.5	HGEFS
Vehicle	78.81	80.87	79.48	79.76	69.17	73.64	**82.02**
WDBC	94.46	95.54	94.46	95.11	95.80	93.14	**97.10**
Ionosphere	87.79	87.57	87.63	87.86	89.06	91.16	**91.33**
Chess	91.20	96.43	94.35	93.98	94.03	**99.43**	98.74
Sonar	77.50	78.70	79.45	77.65	78.75	71.15	**83.00**
Splice	85.50	86.73	88.92	86.37	87.64	**94.07**	93.24
Musk	77.74	77.81	78.72	78.63	79.55	84.87	**88.13**
Arrhythmia	60.44	62.78	62.22	63.78	63.00	63.27	**68.22**
Colon	83.33	86.67	88.83	85.17	90.00	82.25	**93.67**
DLBCL	87.14	90.86	90.97	91.71	92.44	72.73	**97.41**
Average	82.39	84.39	84.38	84.50	83.94	82.58	**89.28**

the highest rank are chosen to induce EM-ELM classifier. We increase t from 1 to m (m is just half of the features number and less than 60), and the optimal t is determined when we obtain the best classification accuracy. And the prediction accuracy of EM-ELM without performing feature selection serves as a baseline. All the results are the average results over 10 runs of tenfold cross-validation to lessen the impact of random factors. The cross-validation process of HGEFS is like a double cross-validation loop: in the outer loop the dataset is split in the training dataset and the test dataset, in the inner loop the training dataset is subsequently again split in a training subset to train EM-ELM and validation set to evaluate the quality of the feature subsets. In the outer loop, the dataset is first partitioned into 10 equal sized sets, and then, each set is in turn used as the test dataset while the other nine datasets are used as the training dataset. In the inner loop, one-third of the training dataset is randomly selected as a validation set for each EM-ELM to evaluate the quality of each subset and the other training dataset is used to train EM-ELM. Different from the outer loop, it uses the hold-out method in the inner loop as we need to maintain the trained EM-ELM classifier to make ensemble.

The experimental results about classification performance on 10 datasets using different algorithms are presented in Tables 4 and 5, where UnSelect depicts the accuracies of EM-ELM on datasets with original features. For each data set, the bold value emphasizes the best accuracy found among all methods.

Table 4 shows the comparative results between our method and ReliefF, Gain Ratio, ChiSquare, CFS-SFS and C4.5. It can be observed that HGEFS outperforms all four filter methods and C4.5 in most cases. As mentioned above, the major disadvantage of the filter approach is that it totally ignores the effects of the selected feature subset on the performance of the induction algorithm. The selected subsets by filter methods are totally independent of EM-ELM so that there is no guarantee that the selected features can improve the performance of EM-ELM. However, in HGEFS the selected subsets are germane to the performance of EM-ELM as their qualities are evaluated by EM-ELM. With the global search ability of GA, the subsets with good prediction accuracies are preserved to make ensemble. In this case, HGEFS can always yield better prediction accuracies. When compared with C4.5, an interesting fact observed in all the datasets considered is that C4.5 performs better on some datasets than HGEFS, namely chess, splice. It is worth noting that these datasets are discrete.

Table 5 Performance comparisons with SVM-RFE, PSO-SVM, GA-ELM, AB and RSE

Datasets	SVM-RFE	PSO-SVM	GA-ELM	AB	MVA	RSE	HGEFS
Vehicle	78.49	80.76	81.68	80.95	81.20	77.32	**82.02**
WDBC	92.62	94.68	96.53	95.14	95.73	94.84	**97.10**
Ionosphere	89.84	90.12	90.36	89.54	90.14	89.01	**91.33**
Chess	97.78	96.82	97.63	95.12	95.37	94.86	**98.74**
Sonar	80.73	81.28	82.16	80.23	80.17	79.50	**83.00**
Splice	89.86	90.89	91.94	87.54	89.76	86.96	**93.24**
Musk	85.42	84.96	86.26	85.27	85.63	84.73	**88.13**
Arrhythmia	64.82	65.73	66.73	63.75	64.58	63.11	**68.22**
Colon	77.42	85.47	91.74	85.36	90.14	84.17	**93.67**
DLBCL	91.32	94.10	95.76	94.29	95.71	92.86	**97.41**
Average	84.83	86.51	88.01	85.72	86.84	84.73	**89.28**

For example, in the chess dataset, all the features are just composed of 7 discrete labels. As a modified decision tree method, C4.5 can yield very good prediction accuracies in some discrete datasets. However, it is not stable as HGEFS and in some cases, Colon and DLBCL, the prediction accuracies of HGEFS are over 20% higher. Compared with the five feature selection methods shown in Table 4, HGEFS can always yield not only good prediction accuracies but also much more stable results.

In Table 5, we compare HGEFS with SVM-RFE, PSO-SVM, GA-ELM, Attribute Bagging (AB), Multi-View Adaboost (MVA) and Randomly Subspace Ensemble (RSE). And the comparison results also indicate that HGEFS is superior to the other methods in Table 5 in terms of prediction accuracy. For SVM-RFE, it returns a ranking of the features of a classification problem by training an SVM with a linear kernel and removing the feature with a smallest ranking criterion. Compared with SVM-RFE, HGEFS can always get a better prediction accuracy for several reasons. Firstly, we adopt a classifier with good generalization performance and high predicting accuracy which is comparable with or even higher than that of linear-SVMs. Secondly, as SVM maximizes the minimum margin between two groups, SVM-RFE is not robust against noisy data even with soft-margin SVM. However, in HGEFS we use ensemble mechanism to reduce the influence of noisy data. Thirdly, the parameter settings in SVM-RFE can not be adjusted according to different feature subsets. But in our method, we simultaneously optimize the feature subset and classifiers' parameters to optimize the prediction accuracies. For PSO-SVM, it uses PSO as the search strategy to wrap SVM. Compared with HGEFS, PSO-SVM does not adjust the parameters of SVM for each different subset to get a better result and meanwhile does not adopt the ensemble mechanism to further improve the final prediction accuracies. For GA-ELM, it is similar to HGEFS to use GA to wrap ELM to solve feature selection problems. In the GA-ELM, its individuals have the same size of features through randomly adding or deleting features to fix the feature size without using any prior knowledge. However, HGEFS generates a score based on the search process and the performance of EM-ELM for each feature to adjust the individual rather than just randomly adds or deletes features. Moreover, we adopt EM-ELM to automatically determine the architecture of ELM. However, in GA-ELM, they just preset the architecture for each different feature subset.

To illustrate the efficiency of ensemble mechanism used in HGEFS, we compare our method with three famous ensemble-based feature selection methods, namely AB, MVA and

RSE. Table 5 shows that HGEFS displays better performance than the other three feature selection ensemble methods. As mentioned above, for a good ensemble algorithm, we should consider two issues: having diversities in their predictions and the high accuracy of the base classifiers. For diversity, all four ensemble methods all ensemble different feature subsets. In addition, AB and MVA adopt Bagging and Adaboost mechanism to resample the training data to train classifiers, respectively. As we know, when the base learner is a weak learning method, the enhancement of prediction accuracy of Bagging and Adaboost is apparent. With EM-ELM, Bagging and Adaboost do not perform as well as they do in weak learning methods. And as is shown in [38], the performance using Bagging and Adaboost with ELM just slightly improves when compared with the simple ensemble. In this case, in terms of improving the diversity, all the four methods are similar. So the factor that influences the final performance mainly focuses on the prediction accuracy of the base classifiers. In AB and RSE, the subsets are randomly generated while MVA uses different filter methods to select different subsets. These subsets are all selected independently of the performance of EM-ELM, which can not guarantee the prediction accuracy. On the contrary, the subsets in HGEFS are germane to the performance of EM-ELM. In HGEFS, through the evolution of GA, the EM-ELMs remained in the final population are expected to have small training error. Based the theory proven in [2], the EM-ELMs with smaller training error and smaller norm of weights tend to have a better generalization performance. To further improve their generalization performance, we select the EM-ELMs with smaller norm of output weights based on the generalization performance theory of ELM. Compared with the other three ensemble methods, the base EM-ELMs in HGEFS tend to get higher prediction accuracies. In other words, the average of testing error of EM-ELMs of HGEFS tends to be smaller than that of the other three methods. As we know, the testing error of ensemble is smaller than the average testing error of each EM-ELMs, which means that HGEFS can yield the best prediction accuracy among the four ensemble methods with the same voting mechanism for ensemble. And the experimental results also support it.

At last, we give some analysis of different methods' computational complexities. In our paper, we compare our method with different filter, wrapper and embedding methods. Among these three kinds of methods, filter methods do not need to train any classifier so that their training time is much smaller than the other two kinds of methods. For embedded methods, they just train a classifier, while the wrapper methods always train many classifiers to evaluate different subsets. In this case, embedded methods always have less training time than wrapper methods. In our experiments, we utilize EM-ELM for the final classification for filer methods. Compared with the time for training EM-ELM, the training time of different filter methods using Weka platform can be ignored. As EM-ELM can be seen as a linear system, the computational complexity of EM-ELM is $O(mn)$, where m is the number of samples and n is the number of features. In this case, the computational complexities of ReliedF, Gain Ratio, ChiSquare, CFS-SFS are $O(mn)$. For the embedded methods, the computational complexity of C4.5 is $O(n \log m)$, while the computational complexity of SVM-RFE is $O(mn^2)$. For wrapper methods, the main processing time is training many classifiers. In our method, we train $pg + p/20$ different EM-ELMs, where p is the size of population and g is the number of search generations ($p/20$ is due to the EI strategy). In this case, the computational complexity of HGEFS is about $(pg + p/20)O(mn)$. For GA-ELM, its computational complexity is about $pgO(mn)$. For PSO-SVM, the computational complexity of kernel SVM we used is $O(mn^2)$ so that its computational complexity is about $pgO(mn^2)$. For ensemble methods, as we train N EM-ELMs for ensemble, their computational complexities are about $NO(mn)$. To better understand the processing time of each method, we take the Arrhythmia dataset as an example in which we use 90% of the dataset to train the related models and then make predictions. For

the filter methods, we firstly use the Weka platform to get the subset and then use EM-ELM to make predictions. The processing time of Weka platform is too small so we ignore it. In this case, the four filter methods have similar processing time, namely 0.56 s. The processing times of C4.5 and SVM-RFE in Weka platform are 0.6 and 226.4 s, respectively. The processing times of AB, MVA and RSE are 5.3, 6.4 and 4.7 s, respectively. For the wrapper methods, the processing times of PSO-SVM, GA-ELM and HGEFS are 50493.1, 4373.8 and 4936.7 s, respectively. Our method is time-consuming when compared with the filter, embedded and ensemble methods. However, when compared with the SVM-based wrapper methods, the processing time of HGEFS is much smaller. When compared with GA-ELM, our process time is a little longer. However, HGEFS can yield better prediction accuracy.

5 Conclusions

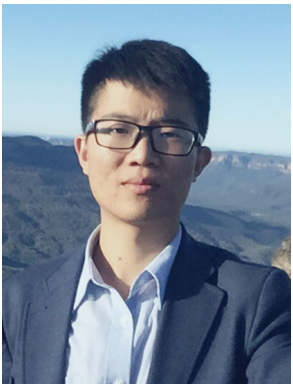
In this correspondence, we propose a novel wrapper feature selection based on genetic algorithm and extreme learning machine. As compared to the other conventional wrapper feature selection algorithms, we adopt EM-ELM for its extremely fast learning speed and high prediction accuracy to allow us to perform feature selection process in an affordable time. And to improve the population's diversity of GA, we introduced an efficient EI mechanism. To further settle the feature selection problem, both the ELMs' architecture optimization and feature subsets optimization were simultaneously executed by HGEFS, and then, an efficient ranking method designed according to the ELM's generalization theory is utilized to select several ELMs. Finally, the selected ELMs perform the classification tasks using the appropriate networks' architecture and subsets of features to make up the final ensemble to improve the prediction and stability. Both the performance and the applicability of the method have been well studied by experiments on various types of datasets. And the results are found strongly to demonstrate the effectiveness of the proposed HGEFS approach.

References

1. Alexandre E, Cuadra L, Salcedo-Sanz S, Pastor-Sánchez A, Casanova-Mateo C (2015) Hybridizing extreme learning machines and genetic algorithms to select acoustic features in vehicle classification applications. Neurocomputing 152:58–68
2. Bartlett PL (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(2):525–536
3. Bermejo P, Gámez JA, Puerta JM (2014) Speeding up incremental wrapper feature subset selection with Naive Bayes classifier. Knowl Based Syst 55:140–147
4. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2013) A review of feature selection methods on synthetic data. Knowl Inf Syst 34(3):483–519
5. Bryll R, Gutierrez-Osuna R, Quek F (2003) Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognit 36(6):1291–1302
6. Cao J, Lin Z, Huang GB, Liu N (2012) Voting based extreme learning machine. Inf Sci 185(1):66–77
7. Chang CC, Lin CJ (2011) Libsvm: a library for support vector machines. ACM Trans Intell Syst Technol TIST 2(3):27
8. Dash M, Liu H (1997) Feature selection for classification. Intell Data Anal 1(1):131–156
9. El Akadi A, Amine A, El Ouardighi A, Aboutajdine D (2011) A two-stage gene selection scheme utilizing MRMR filter and GA wrapper. Knowl Inf Syst 26(3):487–500
10. Feng G, Huang GB, Lin Q, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20(8):1352–1357
11. Fuchs CA, Peres A (1996) Quantum-state disturbance versus information gain: uncertainty relations for quantum information. Phys Rev A 53(4):2038

12. García-Nieto J, Alba E, Jourdan L, Talbi E (2009) Sensitivity and specificity based multiobjective approach for feature selection: application to cancer diagnosis. Inf Process Lett 109(16):887–896
13. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
14. Guyon I, Weston J, Barnhill S, Vapnik V (2002) Gene selection for cancer classification using support vector machines. Mach Learn 46(1–3):389–422
15. Hall MA (1999) Correlation-based feature selection for machine learning. Ph.D. thesis, The University of Waikato
16. Hansen LK, Salamon P (1990) Neural network ensembles. IEEE Trans Pattern Anal Mach Intell 10:993–1001
17. Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844
18. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, Cambridge
19. Huang CL, Dun JF (2008) A distributed pso-svm hybrid system with feature selection and parameter optimization. Appl Soft Comput 8(4):1381–1391
20. Huang CL, Wang CJ (2006) A GA-based feature selection and parameters optimization for support vector machines. Exp Syst Appl 31(2):231–240
21. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B Cybern 42(2):513–529
22. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: 2004 IEEE International Joint Conference on Neural Networks. Proceedings, vol 2. IEEE, pp 985–990
23. Huang J, Cai Y, Xu X (2007) A hybrid genetic algorithm for feature selection wrapper based on mutual information. Pattern Recognit Lett 28(13):1825–1844
24. Jain A, Zongker D (1997) Feature selection: evaluation, application, and small sample performance. IEEE Trans Pattern Anal Mach Intell 19(2):153–158
25. Kira K, Rendell LA (1992) A practical approach to feature selection. In: Proceedings of the ninth international workshop on machine learning, pp 249–256
26. Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97(1):273–324
27. Li S, Harner EJ, Adjeroh DA (2011) Random KNN feature selection—a fast and stable alternative to random forests. BMC Bioinform 12(1):450
28. Li X, Xiao N, Claramunt C, Lin H (2011) Initialization strategies to enhancing the performance of genetic algorithms for the p-median problem. Comput Ind Eng 61(4):1024–1034
29. Lin SW, Chen SC, Wu WJ, Chen CH (2009) Parameter determination and feature selection for back-propagation network by particle swarm optimization. Knowl Inf Syst 21(2):249–266
30. Liu H, Setiono R (1995) Chi2: feature selection and discretization of numeric attributes. In: TAI. IEEE, p 388
31. Pudil P, Novovičová J, Kittler J (1994) Floating search methods in feature selection. Pattern Recognit Lett 15(11):1119–1125
32. Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106
33. Quinlan JR (1996) Improved use of continuous attributes in c4.5. J Artif Intell Res 4:77–90
34. Quinlan JR (2014) C4.5: programs for machine learning. Elsevier, Amsterdam
35. Singh S, Kubica J, Larsen S, Sorokina D (2009) Parallel large scale feature selection for logistic regression. In: Proceedings of the 2009 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, vol. 2009. pp 1172–1183
36. Tan M, Tsang IW, Wang L (2014) Towards ultrahigh dimensional feature selection for big data. J Mach Learn Res 15(1):1371–1429
37. Xu Z, Sun S (2010) An algorithm on multi-view adaboost. In: Neural information processing. Theory and algorithms. Springer, Berlin, pp 355–362
38. Xue X, Yao M, Wu Z, Yang J (2014) Genetic ensemble of extreme learning machine. Neurocomputing 129:175–184
39. Yang W, Li D, Zhu L (2011) An improved genetic algorithm for optimal feature subset selection from multi-character feature set. Exp Syst Appl 38(3):2733–2740
40. Yao L, Sethares WA (1994) Nonlinear parameter estimation via the genetic algorithm. IEEE Trans Signal Process 42(4):927–935
41. Yu L, Liu H (2003) Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the 20th international conference on machine learning (ICML-03). pp 856–863
42. Zhang Y, Wang S, Phillips P, Ji G (2014) Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. Knowl Based Syst 64:22–31

43. Zhu Z, Ong YS, Dash M (2007) Wrapper–filter feature selection algorithm using a memetic framework. IEEE Trans Syst Man Cybern Part B Cybern 37(1):70–76



Xiaowei Xue received the B.Sc. degree from Northeastern University, Shenyang, China in 2011. He received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2017. His current research interests include machine learning, data analysis.

Min Yao received his Ph.D. degree in Biomedical Engineering and Instrument from Zhejiang University, China, in 1995. He is currently a professor at the college of Computer Science and Technology, Zhejiang University. His research interests include computational intelligence, pattern recognition, knowledge discovery and knowledge service.

Zhaohui Wu received the Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, and Kaiserslautern University, Germany, in 1993. He is currently a professor in the College of Computer Science and the vice principal at Zhejiang University. His research interests include distributed artificial intelligence, grid computing and systems, and embedded ubiquitous computing. He is a senior member of the IEEE and a member of the IEEE Computer Society.