RandGA: injecting randomness into parallel genetic algorithm for variable selection

ARTICLE *in* JOURNAL OF APPLIED STATISTICS · MARCH 2015

Impact Factor: 0.42 · DOI: 10.1080/02664763.2014.980788

Chun-Xia Zhang

Xi'an Jiaotong University

25 PUBLICATIONS **220** CITATIONS

SEE PROFILE

Journal of Applied Statistics, 2015
Vol. 42, No. 3, 630–647, http://dx.doi.org/10.1080/02664763.2014.980788

RandGA: injecting randomness into parallel genetic algorithm for variable selection

Chun-Xia Zhang[a*] Guan-Wei Wang[b] and Jun-Min Liu[a]

[a]*School Mathematics and Statistics, Xi'an Jiaotong University, Xi'an Shaanxi 710049, People's Republic of China;* [b]*School of Mechatronic Engineering, Xi'an Technological University, Xi'an Shaanxi 710021, People's Republic of China*

(*Received 14 February 2014; accepted 22 October 2014*)

Recently, the ensemble learning approaches have been proven to be quite effective for variable selection in linear regression models. In general, a good variable selection ensemble should consist of a diverse collection of strong members. Based on the parallel genetic algorithm (PGA) proposed in [41], in this paper, we propose a novel method RandGA through injecting randomness into PGA with the aim to increase the diversity among ensemble members. Using a number of simulated data sets, we show that the newly proposed method RandGA compares favorably with other variable selection techniques. As a real example, the new method is applied to the diabetes data.

Keywords: variable selection; ensemble learning; genetic algorithm; randomness; strength-diversity tradeoff

AMS Subject Classification: 62J05; 68T05; 62J07

1. Introduction

Given n observations $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^{\mathrm{T}}$ is a p-dimensional vector of predictors and y_i is the response variable. We consider the following linear regression model in this article:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \ i = 1, 2, \ldots, n. \tag{1}$$

Here, we assume that the response variable and the predictors are mean-corrected, so there is no intercept term in model (1).

Variable selection is an important topic in the linear regression analysis, especially when the true underlying model has a sparse representation. In practice, a large number of predictors are usually introduced at the initial stage of modeling to attenuate possible model biases. On the

*Corresponding author. Email: cxzhang@mail.xjtu.edu.cn other hand, to enhance prediction accuracy or to select important variables for explanatory purpose, statisticians generally use some variable selection method. Subset selection [1,25] and coefficient shrinkage [8,9,12,15,33,44,45] may be the two most commonly used procedures to select significant variables. Although subset selection is practically useful, it has two fundamental limitations. First, when the number of predictors is large, it is computationally infeasible to execute subset selection. Second, subset selection is instable because of its inherent discreteness as analyzed in [2].

As an alternative, the methods that implement both variable selection and shrinkage in a single procedure gain lots of attention. Lasso (least absolute shrinkage and selection operator) [33] is the most prominent member of these procedures. Lasso works by putting an L_1 penalty on the coefficients, which has the effect of automatically performing variable selection by setting certain coefficients to 0 and shrinking the remainder. This method was made particularly appealing by the advent of the least angle regression (LARS) algorithm [8] which provides a highly efficient means to simultaneously produce the set of Lasso fits for all values of the penalty parameter. Although Lasso has shown success in many situations, it has two limitations in practice. When the model includes several highly correlated variables, all of which are related to the response variable, Lasso tends to pick only one or a few of them while shrinking the rest to 0. Meanwhile, Lasso can identify at most only n variables before it saturates when $p > n$. As a result, numerous improved techniques have been suggested for Lasso including the elastic net [44], adaptive Lasso [43], SCAD [9,45], VISA [26,28], relaxed Lasso [21], etc.

Among the existing statistical modeling methods for solving prediction problems, ensemble learning approaches [14,24,30] have received considerable attention due to their potential to significantly improve the generalization ability of a single model. Boosting [6,10,11], bagging [3], random forest [4] and rotation forest [29] are the most prevalent representatives in ensemble learning. More recently, ensemble methods have become popularized in the variable selection context, for example, parallel genetic algorithm (PGA) [41], stability selection [22,31], random Lasso [34] and stochastic stepwise ensembles (ST2E) [35]. Here, we have to differentiate the terms 'prediction ensemble (PE)' from 'variable-selection ensemble (VSE)' as stated in [35,42] since they are used for different purposes. Specifically, PE aims to maximize *prediction accuracy* so that the future data can be well predicted. With respect to VSE, the purpose is to maximize *selection accuracy* in order that the underlying important variables can be identified as accurate as possible. In addition, Shmueli [32] also pointed out that there exist various distinctions depending on whether the target of statistical modeling is interpretation or prediction.

In order to facilitate later discussions, we first present a brief introduction of VSEs. Suppose that there are p variables, a VSE (of size B) can be represented by a $B \times p$ matrix, say, $\mathbf{E}$, whose the jth column contains B repeated but slightly different measures of how important the variable j is. Based on the matrix $\mathbf{E}$, one can obtain the average importance of each variable using a majority-vote combination rule, that is,

$$R(j) = \frac{1}{B}\sum_{b=1}^{B}\mathbf{E}(b,j), \quad j = 1,2,\ldots,p, \tag{2}$$

where $\mathbf{E}(b,j)$ stands for the (b,j)th entry of the matrix $\mathbf{E}$. Subsequently, the p variables can be ordered and those variables ranked 'considerably higher' than the rest are then selected. This can be done as follows: sort the values $R(1), R(2), \ldots, R(p)$ in a descending order; search for the biggest gap between any consecutive entries; and select the variables that are located above the gap.

The key for generating a VSE lies in producing multiple measures of importance for each candidate variable. In contrast, traditional variable selection methods such as subset selection and Lasso typically produce just one such measure, that is, $B = 1$. Generally speaking, averaging over a number of independent measures is often beneficial. The great success of PEs in various domains such as disease diagnosis [27], concept drift learning [23] and so on is a lively example of this fact. This is also the main reason why VSEs are attractive and more powerful than many traditional procedures.

Similar to the techniques for creating PEs, there can be many ways to construct VSEs. But currently, the commonly used method is to employ a stochastic mechanism so that we can repeatedly perform traditional variable selection and obtain slightly different answers each time. One way is to use a stochastic rather than a deterministic search algorithm to perform variable selection, for example, PGA [41] and ST2E [35]. PGA repeatedly calls genetic algorithm (GA) to perform stochastic optimization of the Akaike information criterion (AIC), and deliberately stops each optimization path prematurely. Early termination forces each optimization path to produce suboptimal rather than optimal solutions, while stochastic optimization allows each of these suboptimal solutions to be different from each other. The main idea of ST2E is to execute stochastic stepwise selection for multiple times. Unlike the traditional stepwise selection which adds or deletes one variable at a time, its stochastic version used in ST2E adds or deletes a *group* of variables at a time, and the group size is *randomly* decided. In the meantime, only a randomly selected few rather than all possible groups are assessed and the best one is chosen. In this manner, a VSE constructed by ST2E contains a series of different variable selectors. Another way to build a VSE is to perform the selection on bootstrap samples. Stability selection [22,31] and random Lasso [34] belong to this type of methods. These two methods both consist of running a randomized Lasso repeatedly on many bootstrap samples and aggregating the results. However, they differ on how Lasso is randomized and the details can refer to [22,31,34].

For PEs, many researchers [5,18,39] have pointed out that diversity and accuracy (or strength) of the ensemble members are two critical ingredients for an ensemble machine to achieve better generalization ability than its each constituent member. In order words, an ensemble machine with good performance should be composed of highly accurate members which at the same time disagree as much as possible. However, diversity and accuracy are mutually contradictory. Take an ensemble classifier as an example, if its base classifiers become more diverse, they must become less accurate. Conversely, as they become more accurate, they must be less diverse. As a result, a good tradeoff between these two terms should be achieved if we want to build an ensemble with good performance. Additionally, Krogh and Vedelsy [16] noticed that increasing the diversity of the ensemble members while without affecting their individual accuracies can improve the performance of an ensemble classifier. The significant superiority of random forest [4] over bagging [3] provides strong evidence of this point.

Bagging [3] and random forest [4] are two methods developed by the statistician Leo Breiman to create PEs. Bagging only utilizes bootstrap sampling to encourage the diversity among its constituent members. Random forest takes a decision tree as its base learner. Besides using bootstrap sampling to obtain different training sets like bagging, random forest tries to produce additional diversity between base classifiers by adding a randomization principle in the tree induction process, which randomly selects a feature subset at each nonterminal node and chooses the best split among it. The combination rule employed by bagging and random forest is the same, that is, simple majority voting for classification or simple averaging for regression. Although the difference is subtle, random forest is seen to almost always perform much better than bagging [14,29,36,38].

Borrowing the similar idea to VSEs, to insure a VSE to be effective, individual members of the ensemble must be as good variable selectors as possible, and they must also be uncorrelated to each other as possible. Motivated by the success of random forest, we present an innovative technique (called RandGA) for generating a VSE by injecting additional randomness into PGA. Notice that in PGA [41], the number of generations for GA to evolve is the same for all the ensemble members. In our newly proposed method RandGA, we make it be randomly decided so that the individuals of the ensemble become more different and uncorrelated from each other. In order to investigate the effectiveness of RandGA, we carried out experiments with some simulations as well as a real-world data set. At the same time, it is compared with several other variables selection techniques. Furthermore, the strength-diversity tradeoff of the VSEs constructed by various methods is also studied. The obtained results indicate the favorable performance of the newly proposed procedure RandGA.

The remainder of this paper is organized as follows. Section 2 provides a brief introduction of single-path genetic algorithm as well as PGA. This is followed by presenting the novel method RandGA to construct a VSE in Section 3. In Section 4, some simulations are used to examine the performance of the proposed method. Meanwhile, the newly proposed approach RandGA as well as PGA is applied to a real example. Finally, conclusions and some future work are outlined in Section 5.

2. A brief introduction of SGA and PGA

Since the newly proposed method is based on genetic algorithm (GA) [7] and PGA, we first give a brief description of them in this section. With the aim to differ GA from the VSEs constructed on the basis of it, using the notations defined in [41], we denote GA as single-path genetic algorithm (SGA) in later discussions. The idea of applying SGA to variable selection is straightforward. Let $\mathcal{C} = \{x_1, x_2, \dots, x_p\}$ be the set containing p predictors and Ω be the collection of all subsets of $\mathcal{C}$. For example, $\omega = \{x_1, x_3, x_4, x_5\} \in \Omega$. In order to make it feasible for applying SGA, ω can be coded as a binary string of length p, namely, $\omega = \{x_1, x_3, x_4, x_5\}$ can be written as $\omega = (1, 0, 1, 1, 1, 0, \dots, 0)$ where the jth location taking 1 indicates the j predictor is selected whereas 0 not. Under this type of encoding mechanism, Ω can be considered as the space of all possible such binary strings. The goal of variable selection is to find, in terms of some criterion, the best $\omega \in \Omega$. This is a typical combinatorial optimization problem and SGA is well suited to find its good solutions.

The optimization strategy used by SGA is very simple. Start with a randomly initialized population of size m (namely, m candidate solutions), SGA uses the Darwinian principle of 'survival of the fittest', the weaker (less optimal) candidates are gradually eliminated and the stronger ones are allowed to survive and generate offsprings. This goes on for a number of generations until, in the end, good solutions are produced. Based on the initial population, a new generation is produced with three genetic operators, namely, selection, crossover and mutation. The main steps of implementing SGA for variable selection are summarized in Algorithm 1.

As for the two important tuning parameters (i.e. the population size of each generation and mutation rate) in SGA, we follow the scheme used in [41]. The population size m is made to be an even number so that the survival pool always contains the best $m/2$ individuals. For simplicity, The mutation rate is assumed to be identical for all generations and it is taken to be the reciprocal of the population size. As far as the fitness score of each individual $\omega \in \mathcal{P}(t)$ is concerned, the residual sum of squares calculated by leave-one-out cross-validation or generalized cross-validation, as well as the AIC can be adopted. Considering that our aim is to select important variables, we employ AIC to achieve this goal in later experiments. Note that the higher the fitness score is, the better the individual is. Thus, the fitness score s for ω is computed as the negative AIC, namely,

$$F(\omega) = -(2|\omega| + \ln \mathrm{RSS}(\omega)), \tag{3}$$

where $|\omega|$ stands for the number of parameters of the model that is determined by ω. $\mathrm{RSS}(\omega)$ indicates the corresponding residual sum of squares, that is, $\mathrm{RSS}(\omega) = (1/n)\sum_{i=1}^{n}(y_i - \hat{y}_i(\omega))^2$.

The main idea of PGA is to execute SGA for multiple times so that some strong and diverse variable selectors can be obtained. In Algorithm 2, we list the main steps of PGA.

Algorithm 1 The SGA for variable selection.

Inputs

y: $n \times 1$ vector.
X: $n \times p$ design matrix.
π: Prior probability to initialize SGA; default = 0.3.
m: population size; default $= p$ if p is even and $p + 1$ otherwise.
N: number of generations to evolve.
v_t: mutation rate for the tth generation; default $= 1/p$ for each generation.

Output

The population at the last generation, $\mathcal{P}(N)$.

Evolution process of SGA

```
1. Use the prior probability $\pi$ to initialize a random population, say, $\mathcal{P}(0)$, of size $m$. In doing
   so, every individual $\omega_i \in \mathcal{P}(0)$ contains $\pi p$ 1's and $(1-\pi)p$ 0's on average.
2. While $t < N$
   a. For each $\omega_i \in \mathcal{P}(t)$, compute its fitness score $s_i = F(\omega_i)$.
   b. Determine the survival pool $\mathcal{S}(t)$ so that the best $m/2$ individuals are retained, that is,
      $\mathcal{S}(t) = \{\omega_i \in \mathcal{P}(t) : s_i > s_{m/2}\}$ where $s_{m/2}$ indicates the median of $s_i$ $(i = 1, 2, \dots, m)$.
   c. Let $\mathcal{P}(t+1) = \mathcal{S}(t)$ and notice that $|\mathcal{P}(t+1)| = m/2 < m$ at this time.
   d. Use the genetic operators (i.e. selection, crossover and mutation) to generate a new
      individual through repeating the following steps.
      • Randomly select a father and a mother from $\mathcal{S}(t)$.
      • Randomly select a crossover position and produce a child from the father and mother.
      • Randomly choose a mutation position and flip the genetic code (i.e. flipping a 0 to a 1
        and vice versa) of the child at this position with probability $v_t$.
      • Insert this child into $\mathcal{P}(t+1)$.
   e. Until $|\mathcal{P}(t+1)| = m$.
   f. Let $t = t + 1$.
3. End while
```

Algorithm 2 The PGA for variable selection.

Inputs

All inputs of SGA (i.e., $\mathbf{y}, \mathbf{X}, \pi, m, N, v_t$);
B: number of paths

Output

The matrix **E** with its (j, b)th entry as $r(j, b)$ $(j = 1, 2, \dots, p, b = 1, 2, \dots, B)$

Algorithm PGA

```
1. For $b = 1, 2, \dots, B$
   a. Let $\mathcal{P}(b, N) =$ result from $SGA(\mathbf{y}, \mathbf{X}, \pi, m, N, v_t)$.
   b. Calculate $r(j, b)$ according to the definition in (4).
2. End for
```

Evidently, if SGA reaches convergence each time, the variable selectors will be similar and even identical to each other. It will be of no use to aggregate these results. Therefore, PGA makes SGA terminate early before its convergence is reached to enhance the diversity among its constituent members. When applying PGA to a specific problem in practice, it is very important to specify the number of generations of SGA appropriately. Zhu and Chipman [41] put forward a criterion based on entropy of a collection of binary sequences. For each problem, they first run the SGA for a couple of times (e.g. 5 or 10 times) using different initial populations and record the average number of generations needed to achieve convergence. Then, half of that number of generations is used for each parallel path.

If we want to use PGA to generate B importance measures for each variable, we should first specify N to be relatively small. Then, start a different evolutionary path with randomly initialized population for $b = 1, 2, \dots, B$ and make it evolve N generations. For ease of simplicity, the population size for each path can be fixed at m in every generation. Let

$$\mathcal{P}(b,t) = \text{the } t\text{th generation in the } b\text{th path.}$$

The core of PGA is to define a measure of the importance, $0 \le r(j,b) \le 1$, for variable j based on N generations of evolution in path b. Zhu and Chipman [41] defined

$$r(j,b) = \frac{1}{m}\sum_{i=1}^{m} \omega_i(j) \quad \text{for } \omega_1, \omega_2, \dots, \omega_m \in \mathcal{P}(b,N). \tag{4}$$

In other words, for each path b, $r(j,b)$ is the percentage of the candidates in the last generation including variable j. The important variables are then selected by taking a majority vote across all pathes. That is, take

$$\bar{r}_j = \frac{1}{B}\sum_{b=1}^{B} r(j,b) \tag{5}$$

as the importance measure of the jth variable and use $\bar{r}_j$ $(j = 1, 2, \dots, p)$ to rank all the variables. Then, the top $q < p$ variables are chosen. In order to determine q, that is, the number of variables to select, one can make an order plot as suggested by Zhu and Chipman [41]. This can be done by plotting $\bar{r}_j$ from the largest to the smallest, looking for the largest gap and then choosing the variables lying above the gap.

Finally, there is one point deserved to be mentioned. Sometimes, a scientist may not expect to identify the correct variables. In many situations, one would simply prefer a properly ranked list of candidate predictors and hope that the important ones are ranked at or near the top of the list. Under this situation, the purpose is to obtain a ranked list whose ordering improves upon that provided by the existing methods. As also remarked in [17,35], the task of ranking is the most fundamental. Once the ordering of the variables is available, from a decision-theoretic framework, the choice of thresholding has more to do with one's belief on the tradeoff between false positive (i.e. incorrectly select unimportant variables) and false negative (i.e. missing some important variables).

3. RandGA: A novel method for constructing VSEs

As we analyzed in introduction, diversity and strength are two critical factors for the effectiveness of a VSE. For PEs, many scholars exploited the tradeoff between accuracy and diversity [4,29,38] to analyze the reasons for the good performance of an ensemble classifier. In the study of VSEs, there are also some researchers employing the similar tool to explain why a VSE works better than traditional variable selection methods. For example, Zhu and Fan [42] offered a method to qualify the strength and diversity of a VSE and related these measures to its variable-selection performance. Their results reveal that PGA produces reasonably strong individual members of VSEs while preserving relatively higher diversity than bagged stepwise selection, which leads PGA to generate more effective VSEs. Xin and Zhu [35] also noticed the importance of strength-diversity tradeoff which is applied to specify an appropriate tuning parameter for their ST2E algorithm.

In light of the findings of Krogh and Vedelsy [16], increasing the diversity among the members of a PE while without affecting their individual accuracies too much can improve its performance. The significant advantage of random forest over bagging provides us a convincing example of this fact. Many improvements [29,37,38] of the popular PE algorithms such as boosting also confirm this statement. Motivated from this, we develop an innovative technique for creating a VSE by injecting randomness into PGA. For ease of presentation, the new method is abbreviated as RandGA in later descriptions. It should be noticed that in PGA, once the number of evolving generations for SGA not converging is determined, the same value is applied to all paths to run SGA.

The key idea of RandGA is to randomly determine how many generations, say, N', a SGA need to evolve for each path so that more diversity is promoted whereas the strength of each member is kept simultaneously. We summarize the main steps of RandGA in Algorithm 3.

Algorithm 3 The main steps of RandGA for variable selection.

```
Inputs
All inputs of SGA (i.e. $\mathbf{y}, \mathbf{X}, \pi, m, N, v_t$);
$f$: an integer taking value $1 \sim 5$;
$B$: number of paths
Output
The matrix $\mathbf{E}$ with its $(j, b)$th entry as $r(j, b)$ $(j = 1, 2, \ldots, p, b = 1, 2, \ldots, B)$
Algorithm RandGA

1. For $b = 1, 2, \ldots, B$
    a. Let $N'$ be an integer drawn from the discrete uniform distribution on the interval $[1, fN]$.
    b. $\mathcal{P}(b, N')$ = result from $SGA(\mathbf{y}, \mathbf{X}, \pi, m, N', v_t)$.
    c. Calculate $r(j, b)$ according to the definition in (4).
2. End for
```

We now explain how the tuning parameter N' of RandGA is determined. For each path in RandGA, we make N' be an integer whose value is drawn in random from a discrete uniform distribution on the pre-specified interval $[1, imax]$. To decide the value of *imax*, we introduce another factor f here aside from N, namely, the number of generations to evolve for SGA that is determined similarly as in PGA. The integer f takes value from 1 to 5. Once the value of f is fixed, *imax* is set to be fN. Then, N' is taken as a random integer that is uniformly sampled from $[1, imax]$.

Note that if $f = 1$, N' will be almost always smaller than N. This implies that each SGA evolves less generations than that in PGA. Because each SGA in PGA has been made to be not convergent, its strength may be reduced if the evolution steps are taken to be smaller. Thus, f should be larger than 1 to ensure the strength of individuals in RandGA. However, too large value of f is also not very helpful. Under this circumstance, some paths of SGA will converge and their variable-selection results thus become more correlated. In other words, the diversity of the individuals in RandGA will be decreased. In practice, the value of f should be cautiously chosen and it can be determined by some data-driven methods such as a validation set according to the specific task at hand. Given a data set, we can also first do some preliminary experiments to find a proper value for f and use it in later experiments.

With regard to the other steps of RandGA, they are similar to those in PGA. As long as one SGA completes its evolution, we calculate the importance measure $r(j, b)$ for variable j according to the formula (4). Then, the measures output from all B paths are fused via majority vote as done in Equation (5). Finally, we can employ the order plot and the largest gap on the plot to identify the most informative variables or just provide a ranked list of the variables depending on the final goal of the given problem.

As for the computational complexity of RandGA, PGA and SGA, it will be analyzed briefly as follows. After checking the main steps of SGA, it can be found that the complexity of SGA is $O(Nnp^2)$ where N denotes the number of evolving generations, n stands for the training sample size and p denotes the number of exploratory variables. Notice that in PGA, each path of SGA is made to evolve about $\lfloor \frac{1}{2}N \rfloor$ generations and there are totally B paths. Therefore, the complexity of PGA should be $O(NBnp^2)$. As far as RandGA is concerned, its complexity will be $O(NBfnp^2)$ since the average number of evolving generations for SGA involved in it is about $\lfloor (f/2)N \rfloor$. Because the value of B and f is generally taken to be small, the complexity of RandGA is only slightly higher than that of SGA and PGA. In later experiments, the parameter f in RandGA is taken to be 3 and RandGA even costs less time than PGA in some situations.

4. Experimental studies

In this section, we conduct some simulations to examine the performance of the proposed method RandGA and compare it with some other variable selection techniques.

4.1 *Simulation* 1

Similar to the simulation used in [41], in this experiment we created a simulated data set consisting of $n = 40$ observations and $p = 20$ variables. The variables are generated from normal distributions and the model is

$$\mathbf{y} = \mathbf{x}_5 + 2\mathbf{x}_{10} + 3\mathbf{x}_{15} + \boldsymbol{\epsilon}, \tag{6}$$

where $\boldsymbol{\epsilon}$ is an error term and $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$. In this model, only variable 5,10 and 15 are actually used in generating the response y. Regarding the mean and covariance for the variables, we considered the following five different situations:

Variation 0: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{20} \sim N(\mathbf{0}, \mathbf{I})$;
Variation 1: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{19} \sim N(\mathbf{0}, \mathbf{I})$, $\mathbf{x}_{20} = \mathbf{x}_5 + 0.25\mathbf{z}$, $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$;
Variation 2: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{19} \sim N(\mathbf{0}, \mathbf{I})$, $\mathbf{x}_{20} = \mathbf{x}_{10} + 0.25\mathbf{z}$, $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$;
Variation 3: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{19} \sim N(\mathbf{0}, \mathbf{I})$, $\mathbf{x}_{20} = \mathbf{x}_{15} + 0.25\mathbf{z}$, $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$;
Variation 4: $\mathbf{x}_j = \mathbf{z} + \boldsymbol{\epsilon}_j, j = 1, 2, \ldots, 20$, $\boldsymbol{\epsilon}_j \sim N(\mathbf{0}, \mathbf{I})$, $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$.

Table 1 lists the value of σ and the correlation structure used for the above five variations. For each variation, we repeat the simulation for 100 times with different random number generating seeds to create data sets in line with the same mechanism. For a method $\mathcal{M}$, we adopted the proportion of times that the correct model is identified, say, $T(\mathcal{M})/100$, to assess its performance.

Table 1. The σ value and correlation structure for the five variations.

Scenario	σ	Correlation structure
Variation 0	1	$\rho(\mathbf{x}_j, \mathbf{x}_k) = 0, \forall j \neq k$
Variation 1	1	$\rho(\mathbf{x}_j, \mathbf{x}_k) \approx 0.97$ for $j = 5$ and $k = 20$
Variation 2	1	$\rho(\mathbf{x}_j, \mathbf{x}_k) \approx 0.97$ for $j = 10$ and $k = 20$
Variation 3	1	$\rho(\mathbf{x}_j, \mathbf{x}_k) \approx 0.97$ for $j = 15$ and $k = 20$
Variation 4	2	$\rho(\mathbf{x}_j, \mathbf{x}_k) \approx 0.80$, for all $j \neq k$

Table 2. Performance of various methods for five variations.

	Scenario				
Method	Variation 0	Variation 1	Variation 2	Variation 3	Variation 4
SGA	0.18	0.17	0.26	0.30	0.19
PGA(hard)	0.91	0.10	0.60	0.91	0.45
PGA(soft)	1.00	0.60	0.86	0.99	0.78
RandPGA(hard)	0.94	0.23	0.69	0.89	0.54
RandPGA(soft)	1.00	0.58	0.83	0.99	0.79

Here, we calculated the 'hard' and 'soft' metrics introduced in [41] for both PGA and RandGA. To compute the 'hard' metric, we simply found the largest gap in the order plot that is obtained on the basis of B paths and selected the variables above the gap. If these variables coincide exactly with the true ones (i.e. variables 5, 10 and 15) used to generate the model, then the method is considered to have found the right model. To calculate the 'soft' metric, we put our emphasis mainly on how variable j is ranked by $\bar{r}_j$. A method is deemed as correctly identifying the right model if all of the true important variables are ranked ahead of the rest using $\bar{r}_j$. According to the detailed explanation by Zhu and Chipman [41], the 'hard' and 'soft' metrics of a method can be, respectively, considered the lower and upper bounds of its performance. As for SGA, we chose the individual with the highest fitness score from its last generation. And the variables which correspond to the locations taking value 1 were selected. SGA is considered to have found the right model if the variables are the same as the true ones.

In this experiment, the number of paths B for PGA and RandGA was taken to be $B = 25$. According to the strategy proposed in [41], we first run SGA for 10 times using different initial populations and found that SGA can reach convergence by evolving about 20 generations on average. Hence, the parameter N in PGA and RandGA was set to be 10. For RandGA, the value for the factor f was set to be 3 since the preliminary experiments demonstrate that it works very well with this setting. With respect to SGA, its number of generations was taken to be 250 so that the total amount of computation of SGA, PGA and RandGA is almost the same. Table 2 summarizes the results for each method.

Generally speaking, the results in Table 2 lead to the following conclusions. SGA is not successful in finding the correct model and its recognition rate is too low in comparison with PGA and RandGA. Variation 0 and variation 3 are relatively easy problems, whereas variation 4 is slightly more difficult because all the variables are correlated with each other. Comparing RandGA with PGA under these situations, the former is seen to perform better than the latter in most cases in terms of either hard or soft metric. For the most difficult cases (i.e. variations 1 and 2), RandGA is observed to behave significantly better than PGA when evaluating them with

Table 3. Widely used benchmark.

Method	Average number of zero coefficients: $\mathbf{x}_j \in$ important variables ($j = 1, 2, 5$)	Average number of zero coefficients: $\mathbf{x}_j \in$ unimportant variables ($j = 3, 4, 6, 7, 8$)	Average model size
$n = 40, \sigma = 3$			
SGA	1.97	3.68	2.18
PGA	0.86	4.96	2.41
RandGA	0.69	4.90	2.35
$n = 60, \sigma = 1$			
SGA	2.02	3.58	2.58
PGA	0.47	4.95	2.92
RandGA	0.10	4.98	2.40
Oracle	0	5	3.00

hard metric. Meanwhile, PGA has a little advantage over RandGA in terms of soft metric. This simulation reveals that RandGA can act as a highly competitive variable selection tool.

4.2 *Simulation* 2

Here, we considered a widely used benchmark simulation [9,21,33–35]. There are $p = 8$ variables and each one is generated from a normal distribution. Furthermore, the pairwise correlation between two variables is $\rho(\mathbf{x}_i, \mathbf{x}_j) = 0.5^{|i-j|}$ for all $i \neq j$. The response $\mathbf{y}$ is generated by

$$\mathbf{y} = 3\mathbf{x}_1 + 1.5\mathbf{x}_2 + 2\mathbf{x}_5 + \sigma\boldsymbol{\epsilon}, \tag{7}$$

where $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$. That is, only three variables (i.e. variables 1, 2 and 5) are truly important and the remaining five ones are unimportant. This benchmark was first used by Tibshirani [33], but it has been used by many researchers to test a variable-selection technique ever since.

First, we chose $n = 40$ and $\sigma = 3$. Then, we reduced σ to 1 and increased the sample size to 60. For each case, the experiment was repeated for 100 times for each method. As for the parametric setup, it was similar to that used in Simulation 1. The number of evolving generations for SGA was taken to be 250. For PGA and RandGA, the parameter N is set to be 10 and $B = 25$. The factor f in RandGA was taken as 3 based on some pre-experiments. In order to evaluate the performance of each method, we recorded the average numbers of zero coefficients, respectively, for the important (i.e. $j = 1, 2, 5$) and unimportant variable group (i.e. $j = 3, 4, 6, 7, 8$) and they are listed in Table 3. In the meantime, we also reported the average model size, namely, the mean size of the selected model over 100 runs of experiments. The method 'Oracle' refers to fitting the model while pretending that we knew in advance the true model contains only the variables $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_5$.

It can be observed in Table 3 that the performance of SGA is the worst regardless of which measure is used. When the noise level is high, PGA is slightly better than RandGA to exclude unimportant variables. However, PGA is much more likely than RandGA to miss truly important variables (i.e. estimate the coefficients for important variables to be zero with wrong), which will become more clearer in Table 4. As far as the average model size is concerned, the result of PGA is slightly nearer to that of Oracle than RandGA. This result should be understood cautiously. In each replication of the experiment, the model size indicates the number of variables

Table 4. Variable selection frequencies of different methods.

	$\mathbf{x}_j \in$ important variables ($j = 1, 2, 5$)			$\mathbf{x}_j \in$ unimportant variables ($j = 3, 4, 6, 7, 8$)		
Method	Minimal	Median	Maximum	Minimal	Median	Maximum
$n = 50, \sigma = 1$						
SGA	27	29	37	24	27	31
PGA	36	96	100	0	0	1
RandGA	87	97	100	0	2	3
$n = 50, \sigma = 3$						
SGA	29	31	34	23	32	36
PGA	51	88	99	1	2	4
RandGA	74	93	100	5	8	12
$n = 50, \sigma = 6$						
SGA	31	32	35	26	35	36
PGA	40	58	89	5	8	11
RandGA	48	60	88	6	7	13

having nonzero coefficients. However, these variables do not necessarily coincide with the truly important ones.

As the second part of this experiment, we recorded the variable selection frequencies of different methods for several cases. The detailed results are reported in Table 4. Here, we list the minimal, median and maximum number of times out of 100 simulations among all important or unimportant variables are selected, respectively. To facilitate the understanding of these statistics, we will briefly describe how these frequencies are computed. It should be noticed that we repeated the experiment for 100 times for each combination of n and σ. In each replication, the frequency associated with a variable is increased by 1 if this variable is considered to be important by an algorithm. Evidently, the frequency for a variable which is more often selected will approximate 100. Because the experimental data are artificially generated, we know in advance that the indices of truly important and unimportant variables. For important variable group (i.e. $j = 1, 2, 5$), we can thus computed the minimal, median and maximum number of times that these variables are identified to be important. For another variable group (unimportant variables, i.e. $j = 3, 4, 6, 7, 8$), the similar calculation can be executed. Ideally, the frequencies for important variables should have high values close to 100 whereas the frequencies for unimportant ones should have low values close to 0.

From Table 4, we can see that PGA controls the number of false discoveries a little more effectively than RandGA, but PGA behave poorly in terms of catching truly important variables. For example, the minimal number of times for PGA finding the important variables is only 36 out of 100 simulations when $n = 50$ and $\sigma = 1$. However, the corresponding value of RandGA reaches as high as 87. This phenomenon further convinces us that RandGA can efficiently identify important variables while excluding noise ones.

4.3 *Simulation* 3

In this simulation, the performance of SGA, PGA and RandGA to deal with high-dimensional data will be examined. Here, we considered a linear regression model in which there exist $p = 100$ exploratory variables which are generated from a multivariate normal distribution.

Table 5. High-dimensional data set.

Method	Average number of zero coefficients.		Average model size
	$\mathbf{x}_j \in$ important variables ($j = 1, 2, 3, 4, 5$)	$\mathbf{x}_j \in$ unimportant variables ($j = 6, \ldots, 100$)	
$n = 100, \sigma = 1$			
SGA	3.03	66.08	30.89
PGA	0.56	94.84	4.60
RandGA	0.36	94.39	5.25
$n = 200, \sigma = 3$			
SGA	2.90	67.89	29.21
PGA	1.23	94.29	4.48
RandGA	0.90	93.42	5.68
Oracle	0	~~5~~	5.00

Meanwhile, the pairwise correlation between two variables is $\rho(\mathbf{x}_i, \mathbf{x}_j) = 0.5^{|i-j|}$ for all $i \neq j$. The response $\mathbf{y}$ is created by

$$\mathbf{y} = 1.175 \sum_{i=1}^{5} \mathbf{x}_i + \sigma \boldsymbol{\epsilon}, \tag{8}$$

where $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$. That is, only the first five variables have actual influence on the response and the remaining ones are unimportant.

Under this circumstance, we carried out two experiments by varying the value of the sample size n and σ. In order to determine the evolving generations of SGA used in PGA and RandGA, we first run SGA for 10 times by using different initial populations. The results indicate that SGA converges after evolving about 35 generations on average. Thus, the parameter N in PGA and RandGA was set to be 15. For the parameter f in RandGA, we set its value to be 3 similar to Simulations 1 and 2. At the same time, the number of paths for PGA and RandGA was taken to be $B = 25$. Aiming to make the amount of computation of SGA, PGA and RandGA be the same, SGA was set to evolve 400 generations. To assess the performance of each algorithm to identify important variables, we employed the statistics utilized in Simulation 2. The experimental results for the considered high-dimensional data set are reported in Tables 5 and 6.

From Tables 5 and 6, we can find that PGA does a little better work than RandGA to exclude unimportant variables. Nevertheless, the false negative rate of PGA is worse than that of RandGA. This means that PGA is more likely to miss some important variables. In this situation, SGA is significantly outperformed by PGA and RandGA.

4.4 *Strength-Diversity tradeoff*

As we analyzed in introduction and Section 3, a good VSE should be composed of a diverse collection of strong individuals. In particular, Zhu and Fan [42] provided a method to quantify the strength and diversity of VSEs. In this subsection, we will utilize their method to analyze the strength-diversity tradeoff of VSEs that are constructed by PGA and RandGA.

Given p potential covariates, let $\mathbf{E}$ be a VSE of size B. Furthermore, let $\mathcal{C}_1$ denote the set of important variables and $\mathcal{C}_0$, the set of unimportant or noise variables. According to the definition

Table 6. Variable selection frequencies of different methods for the high-dimensional data set.

	$\mathbf{x}_j \in$ important variables $(j = 1, 2, 3, 4, 5)$			$\mathbf{x}_j \in$ unimportant variables $(j = 6, \ldots, 100)$		
Method	Minimal	Median	Maximum	Minimal	Median	Maximum
$n = 100, \sigma = 1$						
SGA	31	42	47	18	30	40
PGA	75	90	97	0	0	2
RandGA	86	94	97	0	0	5
$n = 200, \sigma = 3$						
SGA	37	42	47	18	29	38
PGA	64	75	87	0	1	3
RandGA	71	82	90	0	1	7

proposed in [42], the mean strength of **E** is measured by

$$\mathcal{S}(\mathbf{E}) = \frac{1}{B}\sum_{b=1}^{B}\left[\frac{1}{|\mathcal{C}_1|}\sum_{j\in\mathcal{C}_1}\mathbf{E}(b,j) - \frac{1}{|\mathcal{C}_0|}\sum_{j\in\mathcal{C}_0}\mathbf{E}(b,j)\right] \tag{9}$$

and the within-ensemble variation by

$$\mathcal{D}(\mathbf{E}) = 1 - \frac{1}{B(B-1)}\sum_{b=1}^{B}\sum_{b'\neq b}\rho(\mathbf{E}(b,\cdot),\mathbf{E}(b',\cdot)), \tag{10}$$

where

$$\rho(\mathbf{u},\mathbf{v}) = \frac{\sum(u_j-\bar{u})(v_j-\bar{v})}{\sqrt{\sum(u_j-\bar{u})^2\sum(v_j-\bar{v})^2}}$$

is the correlation between vectors **u** and **v**. Based on the simulated data sets in Simulations 1–3, the above measures were calculated and averaged over the 100 runs for different VSEs, and displayed in Table 7.

Together with the results reported in Tables 2–6, we can see from Table 7 that, like PEs, the delicate balance between strength and diversity is critical for the ultimate performance of VSEs. Particularly, PGA tends to produce stronger members for VSEs than RandGA, which is more clear in Simulation 1. However, the diversity measured with the within-ensemble variation of RandGA is much higher than that of PGA. Based on this fact, one of the reasons for the satisfactory performance of RandGA may be that it achieves a better tradeoff between strength and diversity of its ensemble members.

Similar to the kappa-error diagram [19] used to investigate the accuracy-diversity pattern of some ensemble classifiers, here we also plotted the results of PGA and RandGA for two cases in Figure 1. From these two plots, it can be seen more clearly that the strength of PGA is a little better than that of RandGA. On the other hand, however, RandGA is more diverse than PGA. These conclusions are similar to those yielded from Table 7.

4.5 *Analysis of diabetes data*

Finally, we analyzed a real-world example, that is, ‘diabetes’ data, which were used as the main example in the ‘LARS’ article [8]. This data set has $n = 442$ diabetes patients and $p = 10$

Table 7. Strength-diversity analysis of PGA and RandGA.

	Strength		Diversity	
Cases	PGA	RandGA	PGA	RandGA
Simulation 1				
Variation 0	0.4553	0.4083	0.5495	0.5725
Variation 1	0.4039	0.3649	0.5727	0.5903
Variation 2	0.4359	0.3943	0.5637	0.5789
Variation 3	0.4417	0.4037	0.5599	0.5745
Variation 4	0.4236	0.3825	0.5480	0.5593
Simulation 2				
$n = 50, \sigma = 1$	0.3016	0.3085	0.5740	0.5891
$n = 50, \sigma = 3$	0.3026	0.3020	0.5302	0.5420
$n = 50, \sigma = 6$	0.2410	0.2302	0.4536	0.4990
Simulation 3				
$n = 100, \sigma = 1$	0.5254	0.4703	0.7562	0.7685
$n = 200, \sigma = 3$	0.5256	0.4657	0.6862	0.6879

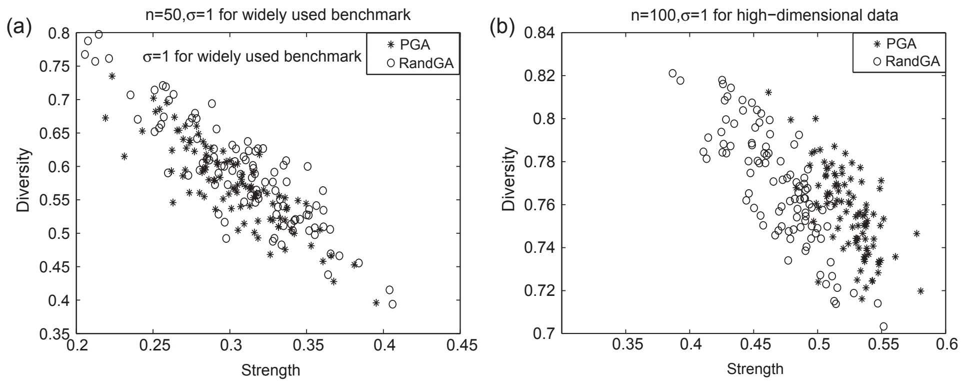

Figure 1. The strength-diversity pattern of PGA and RandGA. (a) A widely used benchmark. (b) A high-dimensional data set.

variables, such as age, sex, body mass index and so on. The response is a measure of disease progression.

For this example, the result of LARS was obtained by using the ‘lars’ package in R software. The corresponding results for PGA and RandGA were calculated using Matlab. Similar to the previous simulations, their parameters were determined in light of some preliminary experiments. Here, $B = 25$, $N = 10$ and the factor f in RandGA was set to be 3.

Figure 2 shows the results from LARS, PGA and RandGA. For LARS, the entire solution paths are displayed for all the variables. As the penalty size increases, the variables enter the model sequentially. The order in which they enter the model is listed in Table 8. For PGA and RandGA, the variable importance measure is plotted for each variable. The order in which these variables are ranked by PGA and RandGA is also depicted in Figure 2.

According to Table 8, the three methods all agree that the variables ‘bmi’, ‘ltg’ and ‘map’ are the most important ones, whereas ‘age’ is the least important variable. Moreover, the ranked list provided by RandGA is almost identical with that of LARS except for the order of ‘tc’ and

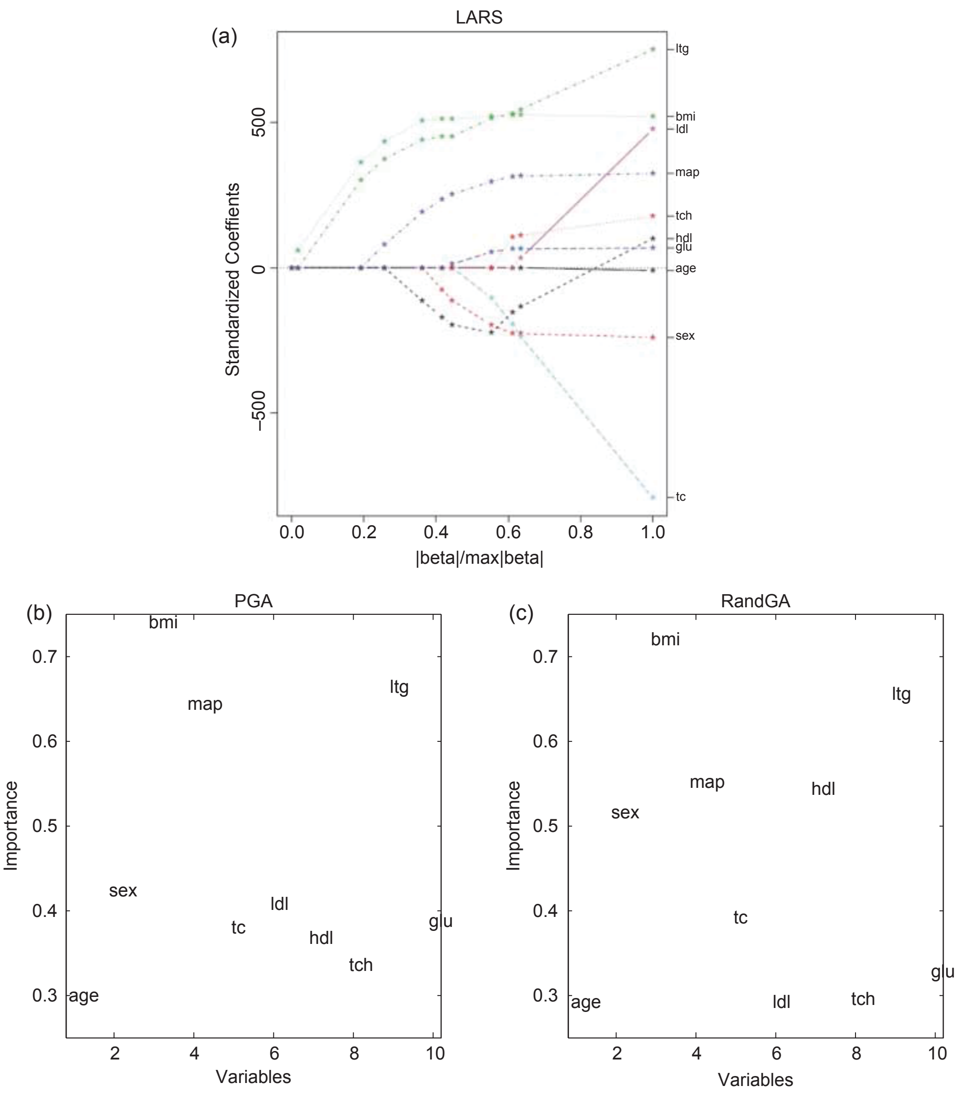

Figure 2. For Diabetes data, the solution path for LARS as well as the ranked list of variables provided by PGA and RandGA. (a) Results from the LARS algorithm. (b) Results from PGA. (c) Results from RandGA.

Table 8. Analysis of diabetes data.

Method	Ordering and ranking of variables top → bottom									
LARS	bmi	ltg	map	hdl	sex	glu	tc	tch	ldl	age
PGA	bmi	ltg	map	sex	ldl	glu	tc	hdl	tch	age
RandGA	bmi	ltg	map	hdl	sex	tc	glu	tch	ldl	age

‘glu’. Comparing PGA with LARS, they seem to disagree on the relative performance for some more intermediate variables. Especially, the variables ‘ldl’ is the last one to be entered into the model by LARS before the variable ‘age’, which indicates that ‘ldl’ is perhaps an unimportant variable. However, PGA ranks ‘ldl’ in the middle of the ranked list. Hence, the variable selection results produced by RandGA are more consistent than PGA with those reported in the previous research [8].

5. Conclusions and future work

Recently, ensemble learning approaches have gained more and more attention of statisticians to do variable selection because of their high efficiency. Generally speaking, a good VSE should be composed of some variable selectors which are strong while not correlated to each other as much as possible. Based on the PGA proposed in [41], this paper presents a novel approach RandGA for construct an ensemble to select significant variables for a linear regression model. The main idea of RandGA is to inject additional randomness into PGA to encourage the diversity among the ensemble members whereas keeping their strength at a certain level so that a better strength-diversity tradeoff can be obtained. The experimental results obtained with a number of simulations as well as a real-world example demonstrate that RandGA performs much better than SGA. In comparison with PGA, their ability to exclude noise variables is comparable. Nevertheless, RandGA does better than PGA to identify the truly important variables.

Even though some techniques [22,31,34,35,41,42] have been proposed for constructing VSEs so far, we believe that the study of ensemble learning for variable selection is still in its initial developing stage and there are many interesting problems deserved to be investigated further. For instance, based on the great success of boosting algorithms [6,10,11,15] in solving classification and regression tasks, whether the idea of boosting can be introduced in a suitable way to implement variable selection? On the other hand, selective ensemble learning methods [13,20,40] have been proven to be quite effective to improve the generalization ability of a PE, reduce its storage need and speed up the prediction process. With respect to VSEs, we surmise that some ensemble members may also be of no use and even deleterious to the performance of a VSE. Thus, some techniques to effectively detect these individuals with the aim to prune a VSE and further enhance its variable-selection ability are also deserved to be studied.

Acknowledgements

We thank the anonymous reviewers and the editor for their valuable comments and suggestions, which have significantly improved this paper. This research was supported by the National Basic Research Program of China (973 Program, No. 2013CB329406), the National Natural Science Foundation of China (No. 11201367), the Major Research Project of the National Natural Science Foundation of China (No. 91230101), the Fundamental Research Funds for the Central Universities of China (No. xjj2011048).

References

[1] L. Breiman, *Better subset regression using the nonnegative garrote*, Technometrics 37(4) (1995), pp. 373–384.
[2] L. Breiman, *Heuristics of instability and stabilization in model selection*, Ann. Stat. 24(6) (1996), pp. 2350–2383.
[3] L. Breiman, *Bagging predictors*, Mach. Learn. 24(2) (1996), pp. 123–140.
[4] L. Breiman, *Random forests*, Mach. Learn. 45(1) (2001), pp. 5–32.
[5] G. Brown, J. Wyatt, R. Harris, and X. Yao, *Diversity creation methods: A survey and categorization*, Inf. Fusion 6(1) (2005), pp. 5–20.
[6] P. Bühlmann and T. Hothorn, *Boosting algorithms: Regularization, prediction and model fitting*, Statist. Sci. 22(4) (2007), pp. 477–505.
[7] S. Chatterjee, M. Lauadto, and L.A. Lynch, *Genetic algorithms and their statistical applications: An introduction*, Comput. Stat. Data Anal. 22 (1996), pp. 633–651.

[8] B. Efron, T. Hastie, I. Hohnstone, and R. Tibshirani, *Least angle regression*, Ann. Stat. 32(2) (2004), pp. 407–499.

[9] J.Q. Fan and R.Z. Li, *Variable selection via nonconcave penalized likelihood and its oracle properties*, J. Am. Statist. Assoc. 96(456) (2001), pp. 1348–1360.

[10] Y. Freund and R. Schapire, *Experiments with a new boosting algorithm*, Proceedings of the 13th International Conference on Machine Learning, Bari, Italy, 1996, pp. 148–156.

[11] J.H. Friedman, *Greedy function approximation: A gradient boosting machine*, Ann. Stat. 29(5) (2001), pp. 1189–1232.

[12] W.J. Fu, *Penalized regression: The bridge versus the lasso*, J. Comput. Graph. Stat. 7(3) (1998), pp. 397–346.

[13] L. Guo and S. Boukir, *Margin-based ordered aggregation for ensemble learning*, Pattern Recognit. Lett. 34(6) (2013), pp. 603–609.

[14] M. Hamza and D. Larocque, *An empirical comparison of ensemble methods based on classification trees*, J. Statist. Comput. Simul. 75(8) (2005), pp. 629–643.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., Springer, 2009.

[16] A. Krogh and J. Vedelsby, *Neural network ensembles, cross validation, and active learning*, in *Advances in Neural Information Processing Systems*, Vol. 7, G. Tesauro, D.S. Touretzky, T.K. Leen, eds., MIT Press, Cambridge, pp. 231–238.

[17] C.Y. Lin, *Some recent developments in parametric and nonparametric regression models*, Ph.D. diss., Raleigh, North Carolina State University, 2012.

[18] S.S. Mao, L.C. Jiao, L. Xiong, and S.P. Gou, *Greedy optimization classifiers ensemble based on diversity*, Pattern Recognit. 44(6) (2011), pp. 1245–1261.

[19] D. Margineantu and T. Dietterich, *Pruning adaptive boosting*, in *Proceedings of the 14th International Conference on Machine Learning*, Morgan, Kafmann, San Francisco, CA, 1997, pp. 211–218.

[20] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez, *An analysis of ensemble pruning techniques based on ordered aggregation*, IEEE Trans. Pattern Anal. Mach. Intell. 31(2) (2009), pp. 245–259.

[21] N. Meinshausen, *Relaxed lasso*, Comput. Stat. Data Anal. 52(1) (2007), pp. 374–393.

[22] N. Meinshausen and P. Bühlmann, *Stability selection (with discussion)*, J. R. Statist. Soc. (Ser. B) 72(4) (2010), pp. 417–473.

[23] D. Mejri, R. Khanchel, and M. Limam, *An ensemble method for concept drift in nonstationary environment*, J. Stat. Comput. Simul. 83(6) (2013), pp. 1115–1128.

[24] J. Mendes-Moreira, C. Soares, A.M. Jorge, and J.F. de Sousa, *Ensemble approaches for regression: A survey*, ACM Comput. Surv. 45(1) (2012), p. 10. Article 10.

[25] A. Miller, *Subset Selection in Regression*, 2nd ed., Chapman & Hall/CRC Press, New York, 2002.

[26] A. Mkhadri and M. Ouhourane, *An extended variable inclusion and shrinkage algorithm for correlated variables*, Comput. Stat. Data Anal. 57(1) (2013), pp. 631–644.

[27] C.O. Plumpton, L.I. Kuncheva, N.N. Oosterhof, and S.J. Jognston, *Naive random subspace ensemble with linear classifiers for real-time classification of fMRI data*, Pattern Recognit. 45(5) (2012), pp. 2101–2108.

[28] P. Radchenko and G.M. James, *variable inclusion and shrinkage algorithms*, J. Amer. Statist. Assoc. 103(483) (2008), pp. 1304–1315.

[29] J.J. Rodríguez, L.I. Kuncheva, and C.J. Alonso, *Rotation forest: A new classifier ensemble method*, IEEE Trans. Pattern Anal. Mach. Intell. 28(10) (2006), pp. 1619–1631.

[30] L. Rokach, *Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography*, Comput. Stat. Data Anal. 53(12) (2009), pp. 4046–4072.

[31] R.D. Shah and R.J. Samworth, *Variable selection with error control: another look at stability selection*, J. R. Statist. Soc. (Ser. B) 75(1) (2013), pp. 55–80.

[32] G. Shmueli, *To explain or to predict?* Statist. Sci. 25(3) (2010), pp. 289–310.

[33] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. R. Statist. Soc. (Ser. B) 58(1) (1996), pp. 267–288.

[34] S.J. Wang, B. Nan, S. Rosset, and J. Zhu, *Random lasso*, Ann. Stat. 5(1) (2011), pp. 468–485.

[35] L. Xin and M. Zhu, *Stochastic stepwise ensembles for variable selection*, J. Comput. Graph. Stat. 21(2) (2012), pp. 275–294.

[36] C.X. Zhang, G.W. Wang, and J.S. Zhang, *An empirical bias-variance analysis of DECORATE ensemble method at different training sample sizes*, J. Appl. Stat. 39(4) (2012), pp. 829–850.

[37] C.X. Zhang and J.S. Zhang, *A local boosting algorithm for solving classification problems*, Comput. Stat. Data Anal. 52(4) (2008), pp. 1928–1941.

[38] C.X. Zhang and J.S. Zhang, *A novel method for constructing ensemble classifiers*, Stat. Comput. 19(3) (2009), pp. 317–327.

[39] M.L. Zhang and Z.H. Zhou, *Exploiting unlabeled data to enhance diversity*, Data Min. Knowl. Discov. 26(1) (2013), pp. 98–126.

[40] Z.H. Zhou, J. Wu, and W. Tang, *Ensembling neural networks: Many could be better than all*, Artif. Intell. 137(1-2) (2002), pp. 239–263.

[41] M. Zhu and H.A. Chipman, *Darwinian evolution in parallel universes: A parallel genetic algorithm for variable selection*, Technometrics 48(4) (2006), pp. 491–502.

[42] M. Zhu and G.Z. Fan, *Variable selection by ensembles for the Cox model*, J. Statist. Comput. Simul. 81(12) (2011), pp. 1983–1992.

[43] H. Zou, *The adaptive lasso and its oracle properties*, J. Amer. Statist. Assoc. 101(476) (2006), pp. 1418–429.

[44] H. Zou and T. Hastie, *Regulzarization variable selection via the elastic net*, J. R. Statist. Soc. (Ser. B) 67(2) (2005), pp. 301–320.

[45] H. Zou and R.Z. Li, *One-step sparse estimates in nonconcave penalized likelihood models*, Ann. Stat. 36(4) (2008), pp. 1509–1533.