Soft Computing (2019) 23:9009–9023
https://doi.org/10.1007/s00500-018-3509-y

METHODOLOGIES AND APPLICATION

A novel multi-objective particle swarm optimization for comprehensible credit scoring

Yan Guo[1] · Jing He[2,3] · Libo Xu[1] · Wei Liu[1]

Published online: 5 September 2018

Abstract
Credit scoring is an important tool for banks and financial institutions to measure credit risk. Linear discriminant analysis (LDA) which according to the score of each credit applicant categorizes these applicants by a cutoff is a comprehensible and robust method in the credit scoring domain. This work presents a novel multi-objective particle swarm optimization for credit scoring (MOPSO-CS), and MOPSO-CS focuses on enhancing credit scoring models based on LDA in three aspects: (i) to construct a higher accuracy credit scoring model which is easy to be interpreted; (ii) to find the most suitable cutoff for discriminating "good credit" customers and "bad credit" customers; and (iii) to improve the sensitivity of the classifier by using multi-objective particle swarm optimization. Finally, through the experiments with two real-world data sets and two benchmark data sets, our proposed MOPSO-CS is compared with 11 counterparts: NaiveBayes, LR, SVM, ANN, DT, CART, bagging-DT, bagging-ANN, RF, MC2 and XGBoost, the results of experiments demonstrate MOPSO-CS outperforms the above-mentioned counterparts in term of sensitivity while maintaining an acceptable accuracy rate.

Keywords Credit scoring · Linear discriminant analysis · Multiple objective optimization · Particle swarm optimization

1 Introduction

Credit risk evaluation decision is a crucial issue for banking industry because even a one percent improvement in early detection of bad credit account may avoid huge amount of losses (Lee and Chen 2005; He et al. 2010). Credit scoring is the most successful method that helps financial institutions to decide whether to grant or refuse a loan (Chi and Hsu 2012), and it can be formally defined as a statistical method for categorizing applicants as either "good credit" group that is likely to repay the financial obligation, or "bad credit" group, with high probability of defaulting their loan (Chen and Huang 2003). In this method, the information of loan applicants (such as age, education, marriage, occupation, income) are usually modeled by a set of features, and a credit score is a model-based estimate of the probability that a borrower will show some undesirable behavior in the future (Lessmann et al. 2015). The advantages of credit scoring are to reduce the cost and time of credit evaluation decision, improve cash flow and insure proper credit corrections (Lee and Chen 2005; Huang et al. 2007).

The popular methods used in building credit scoring models include:

- Linear discriminant analysis (LDA): Jo and Han (1997), He et al. (2010) and Kim et al. (2012);
- Logistic regression (LR): Hand and Henley (1997), Bahnsen and Aouada (2014), Fernandes and Artes (2016);
- Artificial neural networks (ANN): Nanni and Lumini (2009), Hajek (2011), West (2011), Blanco et al. (2013) and Zhao et al. (2015);
- Support vector machines (SVM): Martens et al. (2007), Bellotti and Crook (2009), Rezac (2011), Farquad and Bose (2012), Niklis et al. (2014) and Harris (2015);
- Decision tree (DT): Olson et al. (2012) and Xia et al. (2017);

Communicated by V. Loia.

✉ Yan Guo
guoyanbox@126.com

1 Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, China

2 Institute of Information Technology, Nanjing University of Finance and Economics, Nanjing, China

3 Department of Software and Electronical Engineering, Swinburne University of Technology, Melbourne, Australia

- Bagging: Breiman (1996), Wang et al. (2012) and Xiao et al. (2016);
- Random forest (RF): Yeh et al. (2012) and Lessmann et al. (2015);
- Evolutionary computation techniques: Huang et al. (2006) and Aliehyaei and Khan (2014).

LDA and LR are two statistics models commonly used in credit scoring. LDA which according to the score of each credit applicant categorizes these applicants by a cutoff is a comprehensible and robust method in the credit scoring domain (Lee and Chen 2005). LR is the industry standard, which is useful to examine how a new classifier compares to this approach (Lessmann et al. 2015). Because of assumptions: (i) linear relationship between dependence and independence variables, (ii) the independence variables included in the model are multivariate and normally distributed (Blanco et al. 2013). Both LDA and LR are designed for linear data sets and lack of enough accuracy, especially for nonlinear classification problems (Thomas 2000).

Compared with LDA and LR, ANN and SVM are reported to fit data well for nonlinear data sets, but they are also criticized for long consuming time for training data and building models (Lee and Chen 2002). Even though the accuracy of the credit scoring model is an important criterion, comprehensive and transportability of the model are also significant (Harris 2015). According to the regulation of banking supervision, financial institutions in some countries are obliged to present a comprehensible justification while the credit application is denied (Tomczak and Zięba 2015). Unfortunately ANN and SVM are lack of comprehensive and considered as black boxes (Martens et al. 2007). DT provides a new alternative to ANN and SVM in handling credit scoring tasks, particularly in situations where the scoring models ought to be interpretable. However, the rules of DT will be too complicated to be comprehensive if the number of attributes is immense.

In addition to the above individual models, ensemble methods such as bagging and RF, and evolutionary computation techniques, for example genetic programming (GP) and ant colony optimization (ACO), are recently used to deal with credit scoring problems. Based on the results of the experiments in Huang et al. (2006), two-stage genetic programming (2SGP) can provide better accuracy in credit scoring than LR, ANN and DT. Aliehyaei and Khan (2014) presented a hybrid credit scoring approach (GP-ACO) combining GP and ACO, and compared the performance of GP-ACO with GP and ACO using two real-world data sets. Recently, Lessmann et al. (2015) compared 41 different classification algorithms, and the results suggested that several classifiers outperformed the industry standard LR.

The major contributions and significance of this paper are summarized as follows:

(1) We present a higher accuracy credit scoring model based on LDA. To improve the accuracy of our model, two objectives are taken into consideration. The first objective is to minimize the distance from the sample misclassified to the cutoff, and the second is to maximize the distance from the sample classified correctly to the cutoff.
(2) According to the score of each credit applicant, our credit scoring model categories "bad credit" customers and "good credit" customers by a cutoff. Compared with ANN, SVM and bagging, our credit scoring approach is more comprehensive and easy to be implemented. By comparison with the other comprehensive credit scoring models, for example, DT and CART, the decision rule of out model is more simple, especially for data sets with a lot of features.
(3) In order to solve this problem efficiently, a novel multi-objective particle swarm optimization for credit scoring (MOPSO-CS) is proposed in this work. In MOPSO-CS, a procedure is designed to find the most suitable cutoff for discriminating credit applicants. Then, we provide two versions of MOPSO-CS. One is MOPSO-CS(ACC) which is good at handling balanced data sets, and the other is MOPSO-CS(SEN) which specializes in dealing with imbalanced data sets.
(4) Former literature usually employed imbalanced data sets to benchmark their credit scoring models. In this paper, the experiments employ both imbalanced and balanced data sets to test our proposed MOPSO-CS and the counterparts. The results of these experiments demonstrate MOPSO-CS outperforms the counterparts in term of sensitivity while maintaining an acceptable accuracy rate.

The remaining of this paper is organized as follows. In Sect. 2, the related background on linear discriminant analysis and particle swarm optimization is reviewed. Section 3 presents a comprehensive credit scoring model with a higher accuracy. Section 4 provides a novel multi-objective particle swarm optimization for credit scoring. The experimental results and comparisons are reported in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Background information

2.1 Linear discriminant analysis

Linear discriminant analysis (LDA) classifies credit applicants based on their discriminant scores, which are calculated by a discriminant function (Akkoc 2012).

$$\text{Score}_i = A_i X = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{iR}x_R$$

where Score_i is the credit score of sample i, R is the number of the attributes, $X = (x_1, \ldots, x_R)^T$ is the weight set, and $A_i = (a_{i1}, \ldots, a_{iR})$ is the value set of sample i.

For two-group classification, if $\text{Score}_i \geq b$, sample i belongs to class "Good" (negative); and if $\text{Score}_i < b$, sample i belongs to class "Bad" (positive), where b is a cutoff (boundary), $i = (1, \ldots, N)$, and N is the sample size. The aim of credit scoring method based on LDA is to determine the best coefficients of the variables, denoted by $X = (x_1, \ldots, x_R)^T$, and value b (a scalar) to separate two classes. For credit scoring practice, "Bad" means a group of bankrupt customers, "Good" means a group of normal customers.

There are several measures of classification performance commonly used in the credit industry (Bhattacharyya et al. 2011).

$$\begin{aligned}
&\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{TN} + \text{FN}),\\
&\text{Sensitivity} = \text{TP}/(\text{TP} + \text{FN}),\\
&\text{Specificity} = \text{TN}/(\text{FP} + \text{TN}),\\
&\text{Precision} = \text{TP}/(\text{TP} + \text{FP}),\\
&\text{Recall} = \text{Sensitivity} = \text{TP}/(\text{TP} + \text{FN}),\\
&F1\text{-measure} = (2 \cdot \text{Precision} \cdot \text{Recall})/(\text{Precision} + \text{Recall}),
\end{aligned}$$

where TP = true positives, FN = false negatives, TN = true negatives and FP = false positives.

2.2 Particle swarm optimization

Particle swarm optimization (PSO) is one of an evolutionary computation technology, which was proposed by Kennedy and Eberhart (1995). Similarly as the other evolutionary computation algorithms such as genetic algorithms (GA), GP and ACO, PSO is population based and considered as an efficient search and optimization technique (Omran et al. 2006; Das et al. 2008). A particle swarm consist several particles which fly in search space. The position of particle j denoted as vector $X_j = (x_{j1}, \ldots, x_{jR})$ represents a solution of the optimization problem.

At each iteration t, the velocity of particle j denoted as vector $V_j = (v_{j1}, \ldots, v_{jR})$ is influenced by the best of position visited by itself (Pbest_j) and the best position of the best particle in the swarm (Gbest) using the evolution equation as.

$$\begin{aligned}
v_{jr}(t+1) = {} & w v_{jr}(t) + c_1 r_1 [\text{Pbest}_{jr} - x_{jr}(t)]\\
& + c_2 r_2 [\text{Gbest}_r - x_{jr}(t)].
\end{aligned}$$

Then the position of particle j is updated as follow.

$$x_{jr}(t+1) = x_{jr}(t) + v_{jr}(t+1).$$

where $r = (1, \ldots, R)$ is the dimension index of particle velocity or position, w is an inertia weight, c_1, c_2 are the acceleration constants and r_1, r_2 are two random real numbers drawn from $U(0, 1)$.

In order to solve multi-objective optimization problem, many researchers changed a PSO to a multi-objective PSO (MOPSO). The popular MOPSO includes (i) Pareto-based (Dehuri and Cho 2009); (ii) Coevolutionary approach (Omkar et al. 2008; Goh et al. 2010). Compared with other multi-objective evolutionary algorithms, MOPSO is demonstrated higher convergence speed and efficiency (Goh et al. 2010).

3 A comprehensive credit scoring model with a higher accuracy

3.1 Improving the accuracy of the credit scoring models using LDA

For the credit scoring models using LDA, constructing a linear data set is a useful technique for improving the accuracy. As shown in Fig. 1, there are four samples (sample_1, sample_2, sample_3 and sample_4) in the data set. Let $J = (j_1, \ldots, j_R)^T$, $K = (k_1, \ldots, k_R)^T$ be two wight sets, and Score_{ij}, Score_{ik} is the score of sample i based on the weight sets J or K.

$$\begin{aligned}
&\text{Score}_{ij} = A_i J = a_{i1} j_1 + a_{i2} j_2 + \cdots + a_{iR} j_R,\\
&\text{Score}_{ik} = A_i K = a_{i1} k_1 + a_{i2} k_2 + \cdots + a_{iR} k_R.
\end{aligned}$$

As shown in Fig. 1a, b, according to weight set J, credit scoring model builds a linear data set; and in term of weight set K, credit scoring model constructs a nonlinear data set. If cutoff $= b$, the accuracy of weight set J is the same as the accuracy of weight set K. But as shown in Fig. 2, if we change the cutoff from b to d, the accuracy of the linear data set would be improved from 75 to 100%. However, in terms of the nonlinear data set shown in Fig. 1b, the accuracy is not able to be improved.

To accurately distinguish linear data sets and nonlinear data sets, we defined two variables α_i, β_i for the criteria and constraints as follows, and a graphical representation of these parameters is shown in Fig. 3.

- α_i: the distance from the score of sample i to the cutoff while sample i is misclassified;
- β_i: the distance from the score of sample i to the cutoff while sample i is classified correctly;

Furthermore, as shown in Fig. 3, we can see that the sum of α_i in a linear data set is less than that in a nonlinear data

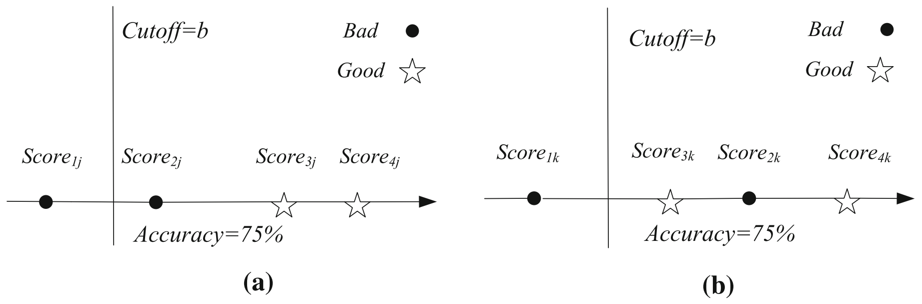

Fig. 1 Linear data set and nonlinear data set. **a** Linear data set, **b** nonlinear data set

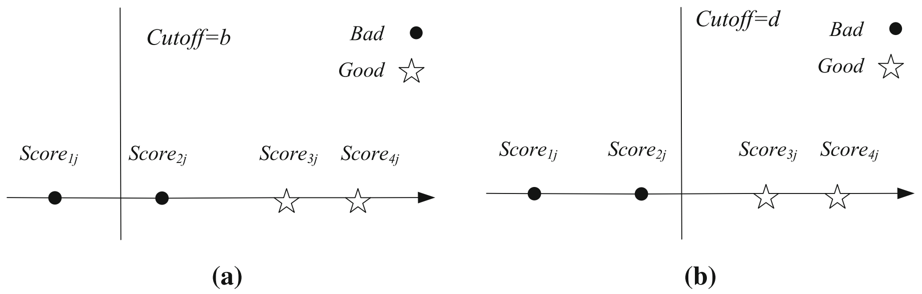

Fig. 2 Improving the accuracy of credit scoring model. **a** Accuracy = 75%. **b** accuracy = 100%

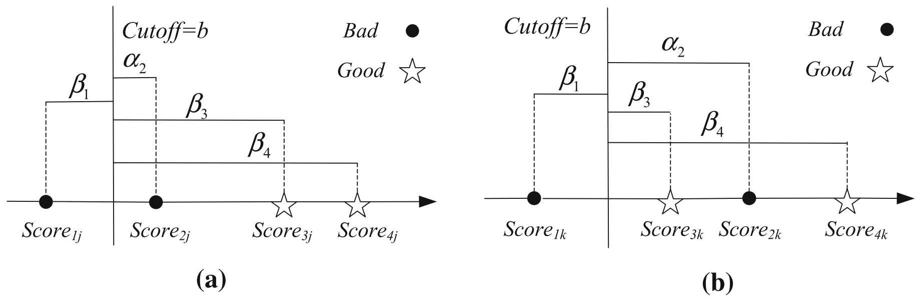

Fig. 3 Graphical representation of α_i, β_i. **a** Linear data set, **b** nonlinear data set

set; and the sum of β_i in a linear data set is more than that in a nonlinear data set.

3.2 A comprehensive multi-objective credit scoring model

In this paper, the multi-objective credit scoring problem based on LDA can be described as how to determine the best coefficients $X = (x_1, \ldots, x_R)^{\mathrm{T}}$ to achieve the following objectives:

(1) Minimize the sum of distances between the score of each sample misclassified and the cutoff

$$\min \mathrm{SMD} = \sum_{i=1}^{N} \alpha_i. \tag{1}$$

where SMD is the sum of α_i.
According to the definition of α_i, it is calculated as.

$$\alpha_i = d(i)\sqrt{(A_i X - b)^2}.$$

where A_i, b are given, X is unrestricted. $d(i)$ is a 0-1 variable, which $d(i) = 1$ if Sample$_i$ is misclassified and $d(i) = 0$ if Sample$_i$ is classified correctly.

- $d(i) = 0, if: A_i X < b, Sample_i \in Bad\ or\ A_i X \geq b, Sample_i \in Good.$
- $d(i) = 1, if: A_i X \geq b, Sample_i \in Bad\ or\ A_i X < b, Sample_i \in Good.$

(2) Maximizing the sum of distances between the score of each sample classified correctly and the cutoff

$$\max \mathrm{SCD} = \sum_{i=1}^{N} \beta_i. \tag{2}$$

where SCD is the sum of β_i.
According to the definition of β_i, it is calculated as.

$$\beta_i = c(i)\sqrt{(A_i X - b)^2}.$$

where $c(i)$ is a 0-1 variable, which $c(i) = 0$ if Sample$_i$ is misclassified and $c(i) = 1$ if Sample$_i$ is classified correctly.

- $c(i) = 0, if : A_i X \geq b, Sample_i \in Bad \ or \ A_i X < b, Sample_i \in Good.$
- $c(i) = 1, if : A_i X < b, Sample_i \in Bad \ or \ A_i X \geq b, Sample_i \in Good.$

These above-mentioned two objectives ensure that our model can obtain higher accuracy solutions than other credit scoring models based on LDA. Meanwhile, similarly as other LDA-based credit scoring models, our model is comprehensible and easy to be implemented. According to weight set X obtained by our model, it is easy for credit manager to explain to credit applicants how their credit scores are derived.

On the other hand, in traditional data classification problem, the accuracy is the main objective to be optimized. With regard to credit card customer classification, sensitivity is more important than other metrics. In this paper, we define a variable MCN to measure the quality of the classification and locate the trade-off between specificity and sensitivity through setting two coefficients w_G, w_B.

$$\mathrm{MCN} = \sum_{i=1}^{N} [w_G d_G(i) + w_B d_B(i)]. \tag{3}$$

where $d_G(i), d_B(i)$ are two 0-1 variables, which $d_G(i) = 0$ if Sample$_i$ is a bad credit customer or Sample$_i$ is a good credit customer which is classified correctly; and $d_G(i) = 1$ if Sample$_i$ is a good credit customer which is classified as bad. $d_B(i) = 0$ if Sample$_i$ is a good credit customer or Sample$_i$ is a bad credit customer which is classified correctly; and $d_B(i) = 1$ if Sample$_i$ is bad credit customer which is classified as good.

- $d_G(i) = 0, if : Sample_i \in Bad \ or \ A_i X \geq b, Sample_i \in Good.$
- $d_G(i) = 1, if : A_i X < b, Sample_i \in Good.$
- $d_B(i) = 0, if : Sample_i \in Good \ or \ A_i X < b, Sample_i \in Bad.$
- $d_B(i) = 1, if : A_i X \geq b, Sample_i \in Bad.$

w_G is the weight of the "Good" samples, and w_B is the weight of the "Bad" samples.

$$w_G + w_B = 1, \quad w_G, w_B \in (0, 1).$$

Furthermore bigger weight w_G would result in bigger specificity. Similarly, and bigger weight w_B would result in bigger sensitivity. Obviously, if $w_G = w_B = 0.5$, MCN is equal to half of the number of misclassified samples. $MCN = 0$ if all samples are classified correctly, and then we define the solution which classifies all samples correctly as the optimal solution.

Define 1 (Optimal solution) the solution $X_j = (x_{j1}, \ldots, x_{jR})$ which classifies all samples correctly.

$$F = \{X_j | \mathrm{MCN}(X_j) = 0, X_j \in \Omega\}. \tag{4}$$

where F is the set of optimal solutions, Ω is the set of all solutions, and $\mathrm{MCN}(X_j)$ is the MCN of solution X_j. According to Objective (1), the SMD of the optimal solution is equal to 0.

Usually, it is not able to obtain an optimal solution if the data set is linear inseparable. Then the objective of classification problem is to find the feasible solution whose MCN is the minimum.

Define 2 (Feasible solution) the solution with the minimum misclassification coefficient.

$$S = \{X_j | \mathrm{MCN}(X_j) = \mathrm{MCN}_{\min}, X_j \in \Gamma\}. \tag{5}$$

where S is the set of feasible solutions, $\mathrm{MCN}_{\min} = \min\{\mathrm{MCN}(X_l) | X_l \in \Gamma\}$, and Γ is the set of all solutions obtained by the algorithm. Obviously, all optimal solutions are feasible solutions.

4 Multi-objective particle swarm optimization for credit scoring

With the deep research on evolutionary computation technologies, scholars find that GA is usually used to deal with discrete combinatorial optimization, and PSO is suitable for solving the optimization problem with continuous variables. Because each solution X_j is continuous in the credit scoring problem, multi-objective particle swarm optimization is used to deal with this problem. Multiple subswarms and coevolutionary approach is able to improve the efficiency and effectiveness of MOPSO (Goh et al. 2010); in this paper, a coevolutionary MOPSO is transformed for credit scoring problem, and then we propose a new multi-objective particle swarm optimization: MOPSO-CS. MOPSO-CS employs two subswarms (Subswarm$_1$ and Subswarm$_2$) with the same population J to probe the search space and information is exchanged between them.

The subswarms are evaluated in an iterative manner: Subswarm$_1$ evaluates SMD, and Subswarm$_2$ evaluates SCD. But in terms of data classification, a particle rank in MOPSO-CS is different from Pareto rank which is usually used in traditional MOPSO. In this work, a particle rank $rank(j)$ is given by:

$$\mathrm{rank}(j) = 1 + n_j. \tag{6}$$

where n_j is the number of misclassified samples according to the solution of particle j.

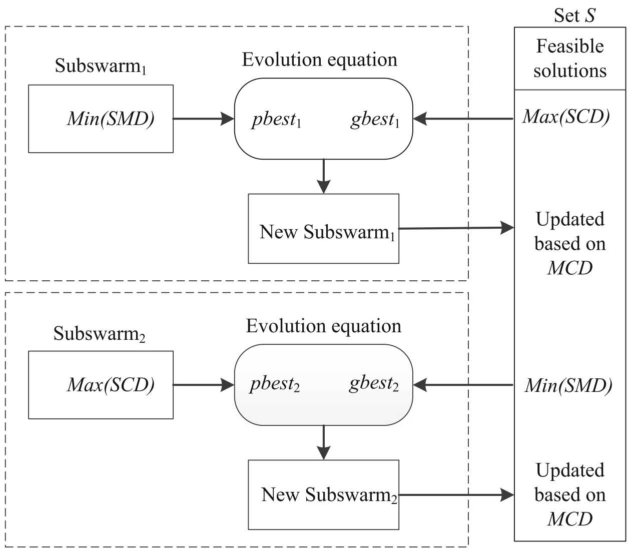

Fig. 4 Best position selection and updating process

4.1 Best position selection and updating process

Let the position of the particle in Subswarm$_1$ whose value of SMD is minimum be the personal best position (Pbest$_1$) of Subswarm$_1$, and let the position of the particle in Subswarm$_2$ whose value of SCD is maximum be the personal best position (Pbest$_2$) of Subswarm$_2$.

In order to search optimal solution as soon as possible, we select two feasible solutions from set S, and let the position of the feasible solution with maximum SCD be the global best position (Gbest$_1$) of Subswarm$_1$, and let the position of the feasible solution with minimum SMD be the global best position (Gbest$_2$) of Subswarm$_2$. Through this method (shown in Fig. 4), each swarm can share information from feasible solutions, so that MOPSO-CS is able to guide the particles to optimal solution as soon as possible.

MOPSO-CS is similar to coevolutionary MOPSO, and at each iteration t, particle j in the kth swarm updates its current position $x_{kjr}(t)$ and velocity $v_{kjr}(t)$ through each dimension r by the personal best position Pbest$_{kjr}$ and the global best position Gbest$_{kr}$ using evolution equation shown in Eqs. (7) and (8):

$$v_{kjr}(t+1) = wv_{kjr}(t) + c_1r_1[\mathrm{Pbest}_{kjr} - x_{kjr}(t)] + c_2r_2[\mathrm{Gbest}_{kr} - x_{kjr}(t)]. \quad (7)$$

$$x_{kjr}(t+1) = x_{kjr}(t) + v_{kjr}(t+1). \quad (8)$$

Because there are perhaps many particles with the same particle rank, SMD and SCD are also taken into consideration while each two particles are compared. The detailed criterion of rank for each particle is as follows:

Particle a is superior to particle b $\Longleftrightarrow$

rank(a) < rank(b)
Or rank(a) = rank(b) and SMD(a) < SMD(b)
Or rank(a) = rank(b) and SMD(a) = SMD(b) and SCD(a) > SCD(b).

where SMD(a), SCD(a) are the SMD, SCD of particle a.

4.2 The optimization process

For MOPSO-CS, the scope of the solution is R-dimensional search space where R is the number of attributes (variables). The position of particle j in Subswarm$_k$ is represented by $x_{kj} = (x_{kj1}, \ldots, x_{kjR})$, which corresponds to a solution for the problem. The rth dimension of the position x_{kjr} ($k = 1, 2$; $j = 1, \ldots, J$; $r = 1, \ldots, R$) denotes the rth coefficient used by each sample. The coding design of particle velocity is similar to the design of particle position, which is denoted by $v_{kj} = (v_{kj1}, \ldots, v_{kjR})$.

Firstly the algorithm randomly generates two subswarms. Then the positions and velocities of all particles in these subswarms are initialized. Each element x_{kjr} in particle position x_{kj} is randomly initialized within [− 100, 100]. Each element v_{kjr} of particle velocity v_{kj} is randomly initialized within [− 10, 10].

Secondly evaluate these subswarms. According to which has been initialized, calculate SMD and SCD of each particle using objective (1), (2). Then, evaluate the particle rank of each particle using Eq. (6). Moreover, the algorithm initializes the global best position of each subswarm and the personal best position of each particle.

In this paper, MOPSO-CS tries to find optimal solutions of this problem and terminates when the first optimal solution is found, or the maximum iteration number (T) is reached. Before the algorithm finds an optimal solution, all feasible solutions are used as the best found solutions. At each iteration of algorithm, MOPSO-CS updates the best found solutions in term of MCN and reserves the solutions with minimum MCN found by optimization process.

Thirdly update the position and velocity of each particle according to Eqs. (7) and (8). Finally, we select top $2J$ particles with minimum particle rank from old population and new population, and use them as the population for the next generation. The optimization process can be formally described as Fig. 5.

4.3 Finding the optimal cutoff for linear discriminant

In traditional credit scoring methods using LDA, the cutoff is fixed and assigned by experience. Because finding the optimal cutoff is significant for the accuracy of classification, many methods are proposed to find a better cutoff to improve the precision even if it is time-consuming. In this

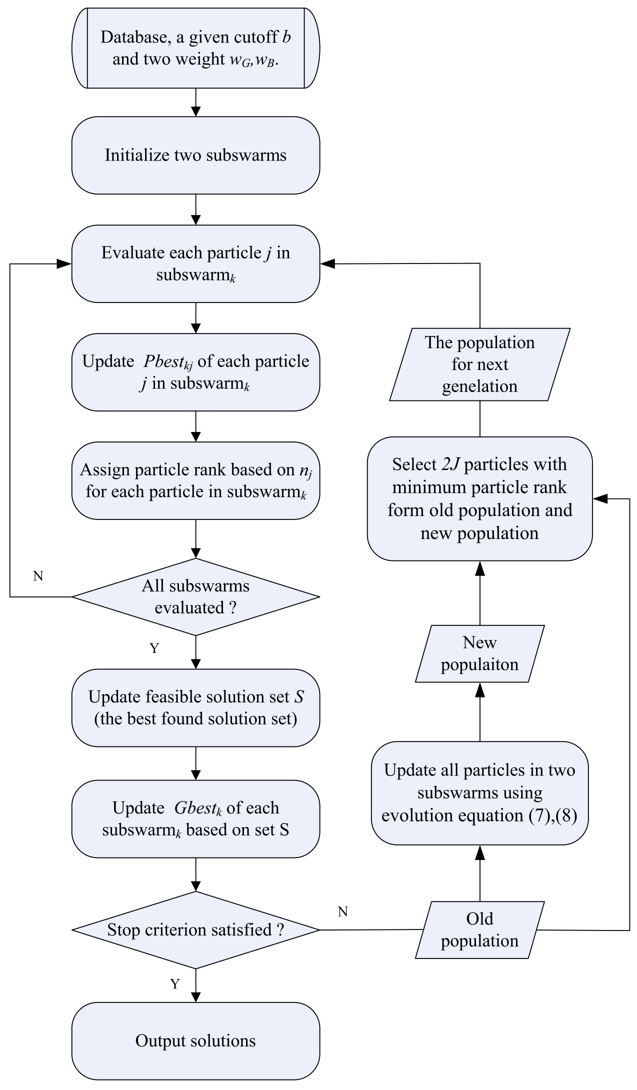
Database, a given cutoff b and two weight w_G, w_B.
Initialize two subswarms
Evaluate each particle j in subswarm$_k$
Update $Pbest_{kj}$ of each particle j in subswarm$_k$
Assign particle rank based on n_j for each particle in subswarm$_k$
All subswarms evaluated ?
N
Y
Update feasible solution set S (the best found solution set)
Update $Gbest_k$ of each subswarm$_k$ based on set S
Stop criterion satisfied ?
N
Y
Output solutions
The population for next genelation
Select 2J particles with minimum particle rank form old population and new population
New populaiton
Update all particles in two subswarms using evolution equation (7),(8)
Old population

Fig. 5 Algorithmic flow of MOPSO-CS

paper, a misclassification-number-based approach for finding the optimal cutoff is proposed.

In MOPSO-CS, each particle j represents a solution $X_j = (x_{j1}, \ldots, x_{jR})^{\mathrm{T}}$. Then we can calculate the credit score of each sample i in term of the solution X_j, and let $\text{Score}_{ij} = \sum_{r=1}^{R} a_{ir} x_{jr}$ be the credit score of sample i based on particle (solution) j. When the scores of all samples in training set have been calculated, MOPSO-CS can find the optimal cutoff with minimum number of misclassification as follows.

Let the number of records in training set be *NTRAIN*, then there are up to *NTRAIN* different scores in training set. Each score can be used as a candidate to the cutoff. By assigning each score to the cutoff one by one, MOPSO-CS is able to test the result of each assignment and select the optimal b with the minimum misclassification number as the cutoff. The detailed approach of finding the optimal cutoff is shown as Fig. 6.

If sample i is a bad credit customer, for example sample 3 in Fig. 6a, let $b = \text{Score}_{ij} + \xi$; and if sample i is a good credit customer, such as sample 4 in Fig. 6b, let $b = \text{Score}_{ij} - \xi$. ξ is a positive minimal real number, such as 0.0001. Then the number of misclassification (n_j) of solution X_j is able to be calculated in term of $b = \text{Score}_{ij} \pm \xi$. By the same method, each candidate cutoff $b = \text{Score}_{ij} \pm \xi$, $i \in (1, \ldots, NTRAIN)$ is able to be tested.

For the example in Fig. 6a,

- if $b = \text{Score}_{1j} + \xi, n_j = 1$
- if $b = \text{Score}_{2j} - \xi, n_j = 1$
- if $b = \text{Score}_{3j} + \xi, n_j = 1$
- if $b = \text{Score}_{4j} - \xi, n_j = 1$
- if $b = \text{Score}_{5j} - \xi, n_j = 2$

Finally, MOPSO-CS selects the optimal cutoff with minimum misclassification number from all candidate b. In the example shown in Fig. 6a, we can let $b = \text{Score}_{1j} + \xi$, and the precision of this classification is 80%. The pseudocode of finding cutoff process is shown in Algorithm 1.

Algorithm 1 Finding cutoff

```
1: for $i = 1; i \le NTRAIN; i++$ do
2:     Evaluate the credit score of sample $i$ in term of the solution $X_j$
3: end for
4: $Min = NTRAIN$
5: for $i = 1; i \le NTRAIN; i++$ do
6:     Calculate $n_j$ in term of $b = Score_{ij} \pm \xi$
7:     if $n_j < Min$ then
8:         $Min = n_j$
9:         $Cutoff = b$
10:    end if
11: end for
```

Obviously, the complexity of this process is $O(NTRAIN^2)$. Since the process of finding the optimal cutoff is complicated and time-consuming, we present two versions of MOPSO-CS. One is MOPSO-CS(ACC) which omits the optimization on the cutoff, and the other is MOPSO-CS(SEN) which includes the optimization on the cutoff. Through computational experiments, we discover that the time of computation of MOPSO-CS(SEN) is longer than MOPSO-CS(ACC), but the speed of convergence of MOPSO-CS(SEN) is faster than MOPSO-CS(ACC).

5 Computational experiments

5.1 The selection and preprocessing of credit data set

In computational experiments, we use various data sets that represent the credit behaviors of people from different coun-

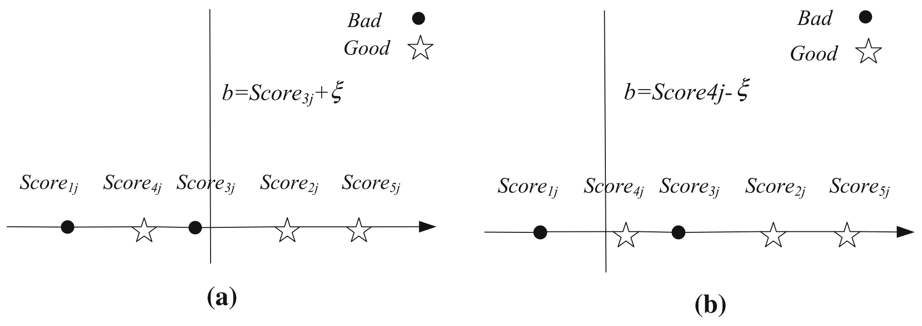

Fig. 6 Finding the optimal cutoff with minimum misclassification number. **a** Sample$_3$ is a bad credit customer. **b** Sample$_4$ is a good credit customer

Table 1 Selection and preprocessing of credit data set

Expt.	Standardized	Feature evaluator	Search method
1	No	All features	None
2	Yes	All features	None
3	No	CorrelationAttributeEval	Ranker
4	Yes	CorrelationAttributeEval	Ranker
5	No	cfsSubsetEval	BestFirst
6	Yes	cfsSubsetEval	BestFirst
7	No	InfoGainAttributeEval	Ranker
8	Yes	InfoGainAttributeEval	Ranker
9	No	WrapperSubsetEval	BestFirst
10	Yes	WrapperSubsetEval	BestFirst

Table 2 Employed classifiers

Classifier	Comprehensible	Software
MOPSO-CS(SEN)	Yes	Visual C++ 6.0
MOPSO-CS(ACC)	Yes	Visual C++ 6.0
NaiveBayes	No	WEKA3.8 Bayes(NaiveBayes)
LR	No	WEKA3.8 Functions(Logistics)
SVM	No	WEKA3.8 Functions(LibSVM)
ANN	No	WEKA3.8 Functions (MultilayerPerceptron)
DT	Yes	WEKA3.8 Trees(J48)
CART	Yes	WEKA3.8 Trees(SimpleCart)
Bagging-DT	No	WEKA3.8 meta(Bagging)
Bagging-ANN	No	WEKA3.8 meta(Bagging)
RF	No	WEKA3.8 Trees(Random Forest)

tries which include two real-world data sets from UK and two benchmark data sets, i.e., German and Taiwan. All these data sets classify people as Normal or Bankrupt credit risks based on a set of attributes (variables). In order to investigate the effectiveness of the proposed algorithm in imbalanced and balanced data sets, the real-world credit data from UK are separated to two date sets. The first is a UK bankruptcy data set (He et al. 2010), which collects 323 bankrupt and 902 normal customers with 14 variables. The second is a balanced data set derived from the first data set, which consists of 323 normal customers and 323 bankrupt customers. The other two data sets are German (700 normal and 300 bankrupt customers with 24 features) and Taiwan (3000 normal and 3000 bankrupt customers with 23 features), which are all from UCI Machine Learning databases (Xia et al. 2017).

Previous research (Koutanaei et al. 2015) has demonstrated that selecting suitable features can improve the accuracy in credit scoring problems, and then we have preprocessed the data before classification. Firstly, we create two versions of each data set, one is the raw data which is not standardized, the other is standardized. Secondly, we apply five different methods to select features. As shown in Table 1, for each data set, ten experiments are performed. The data in the experiments with odd index are not standardized, and the other experiments with even index are standardized.

5.2 Experimental evaluation

5.2.1 Implementation of the classifiers

In this section, our proposed method, MOPSO-CS, is compared with nine classic classification methods: NaiveBayes, logistics regression (LR), support vector machine (SVM), artificial neural network (ANN), decision tree (DT), CART, bagging-DT, bagging-ANN and random forest (RF). In these classifiers, MOPSO-CS, DT and CART are comprehensible. MOPSO-CS is conducted by Visual C++ 6.0, and the implementation software of these counterparts is WEKA 3.8. All experiments were performed on a PC Core i5 with 2.5GHz and 4GB RAM running under the Windows 10 operating system. Table 2 shows the implementation softwares that were explored to run the four data sets.

Because the real-world data sets are usually linear inseparable, it is hard for MOPSO-CS to obtain an optimal solution. We use the maximum iteration as the termination criteria of MOPSO-CS. The maximum iteration (T) is decided on

Table 3 Parameters of MOPSO-CS

Algorithm	Data set	w_G	w_B	T	Population	Cutoff
MOPSO-CS(SEN)	UK1	0.33	0.67	100	40	Variable
MOPSO-CS(ACC)	UK1	0.50	0.50	1000	40	− 1.1
MOPSO-CS(SEN)	UK2	0.45	0.55	100	40	Variable
MOPSO-CS(ACC)	UK2	0.50	0.50	1000	40	− 1.1
MOPSO-CS(SEN)	German	0.40	0.60	100	40	Variable
MOPSO-CS(ACC)	German	0.50	0.50	1000	40	− 1.1
MOPSO-CS(SEN)	Taiwan	0.33	0.67	100	40	Variable
MOPSO-CS(ACC)	Taiwan	0.50	0.50	1000	40	− 1.1

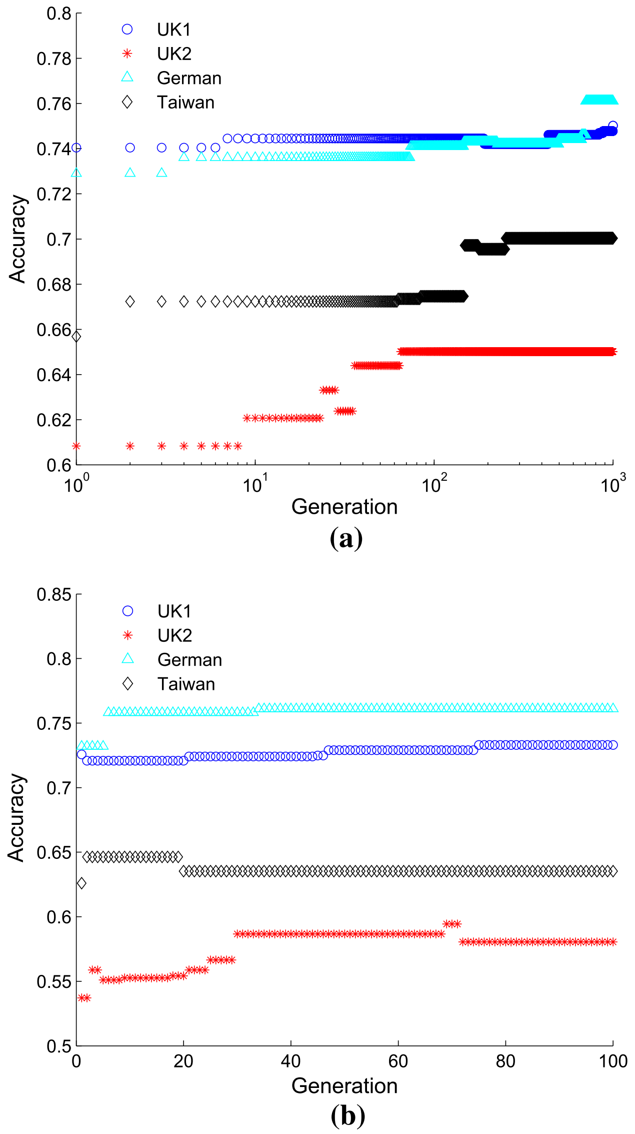

Fig. 7 Accuracy of MOPSO-CS of each data set. **a** MOPSO-CS(ACC). **b** MOPSO-CS(SEN)

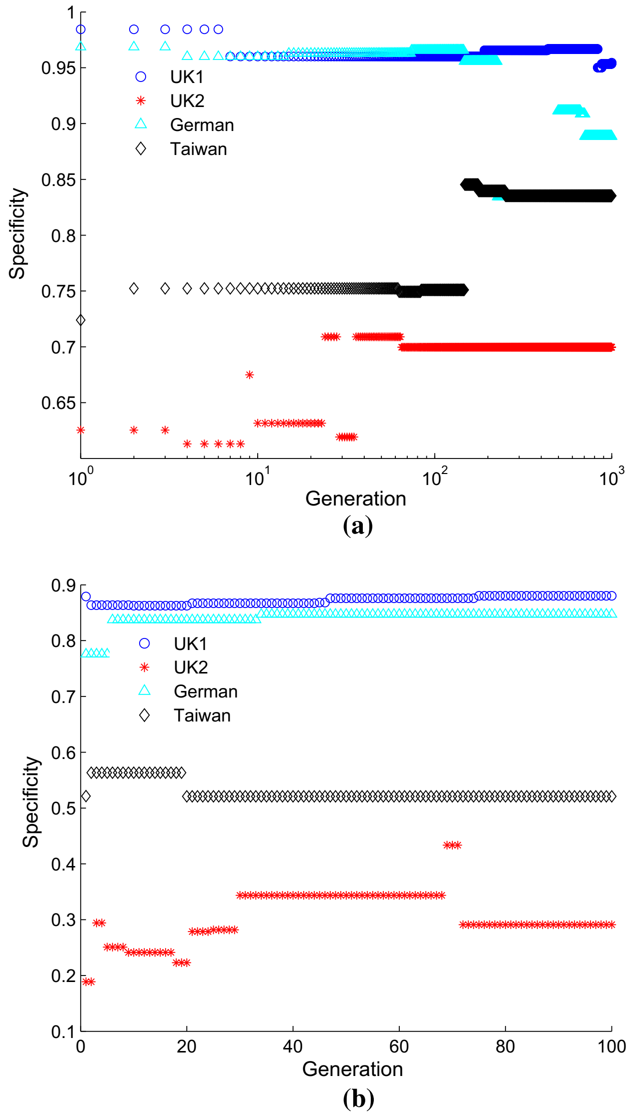

Fig. 8 Specificity of MOPSO-CS of each data set. **a** MOPSO-CS(ACC), **b** MOPSO-CS(SEN)

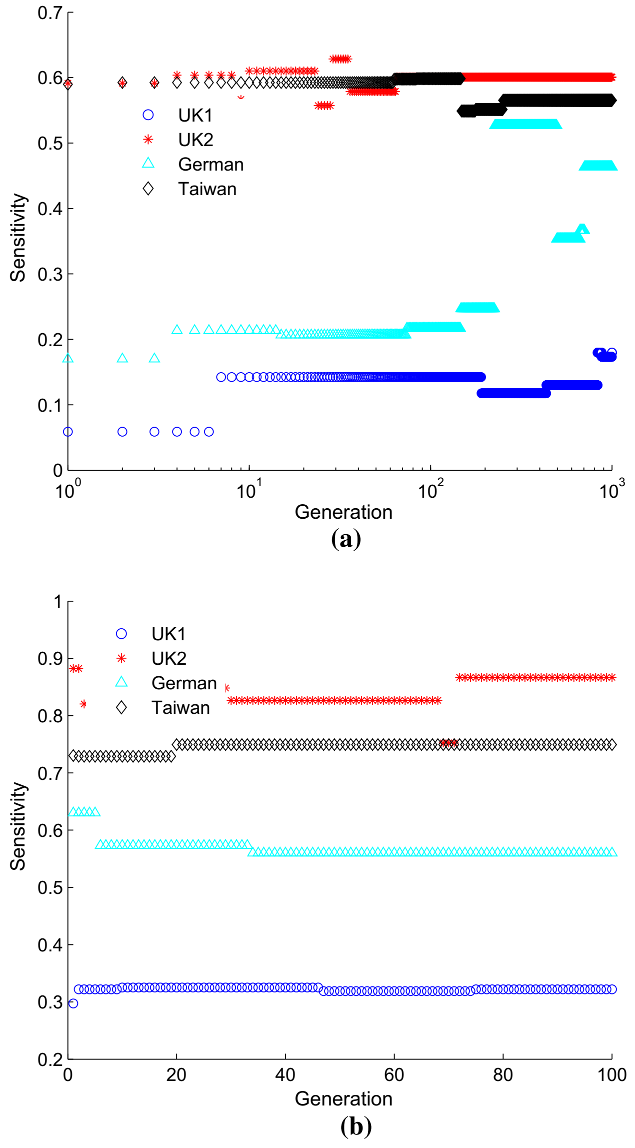

Fig. 9 Sensitivity of MOPSO-CS of each data set **a** MOPSO-CS(ACC), **b** MOPSO-CS(SEN)

stabilization principle which is testified by numerous tests. In our experiments, the cutoff is set to -1.1 in MOPSO-CS(ACC), and it is variable in the MOPSO-CS(SEN) as shown in Sect. 3.4. The parameters used in MOPSO-CS are presented in Table 3.

5.2.2 The convergence of MOPSO-CS

We conducted tenfold cross-validation to evaluate the performance of these classifiers, and the tenfold cross-validation process is repeated 10 times. In order to investigate the convergence of the proposed algorithm, we record the accuracy, specificity and sensitivity of the feasible solutions obtained by MOPSO-CS at each iteration. Figures 7, 8 and 9 show the accuracy , specificity and sensitivity of each iteration in the first tenfold cross-validation process of the experiment, which contains the highest accuracy.

From Fig. 7, the accuracy obtained by MOPSO-CS is rising with an increasing number of generation, and it shows that the multi-objective credit scoring method we proposed is effective for improving the accuracy of data classification. Compared with MOPSO-CS(SEN), MOPSO-CS(ACC) performs better in terms of accuracy.

The results of Fig. 8 indicate that specificity fluctuates with the iterations of MOPSO-CS. Moreover, because w_G in MOPSO-CS(ACC) is bigger than w_G in MOPSO-CS(SEN), the specificity of MOPSO-CS(ACC) is excel than that of MOPSO-CS(SEN).

Figure 9 indicates that sensitivity increases with the iterations of MOPSO-CS, and the sensitivity of MOPSO-CS(SEN) is better than that of MOPSO-CS(ACC). Moreover, the results shown in Figs. 7, 8 and 9 reveal that MOPSO-CS(SEN) has converged to steady solutions within 100 iterations, and MOPSO-CS(ACC) converges more slowly than MOPSO-CS(SEN).

5.2.3 Results and discussion

In this section, we compare the performance of MOPSO-CS with nine counterparts. For each date set, we perform ten experiments and define the experiment which contains the highest accuracy as standard experiment. In Tables 4, 5, 6, and 7, the last column represents the index of standard experiment of each classifier.

Table 4 shows the results of tenfold cross-validation of the UK1 data set (902 normal customers, 323 bankrupt customers). As shown in Table 4, the sensitivity and $F1$-measure of MOPSO-CS(SEN) excel all counterparts, and its accuracy is more than 73%. Although accuracy is not as important as sensitivity for credit scoring problem, we also try to optimize the precision of classification by MOPSO-CS. The accuracy of MOPSO-CS(ACC) is superior to NaiveBayes, LR, ANN, CART, bagging-DT, bagging-ANN and RF, only below SVM and DT.

Table 5 shows the results of tenfold cross-validation of the UK2 data set (323 normal customers, 323 bankrupt customers). Because the number of bankrupt customers is the same as that of normal customers in a balanced data set, the accuracy of this data set is harder to be improved than that of imbalanced data sets. MOPSO-CS(ACC) achieves the highest accuracy of all classifiers and exhibits the second best $F1$-measure after NaiveBayes. Moreover, MOPSO-CS(SEN) obtains the second best sensitivity, only slightly below NaiveBayes. DT's precision is the best, but its $F1$-measure is the poorest.

Table 6 shows the results of tenfold cross-validation of the German data set. The sensitivity and $F1$-measure of

Table 4 Results on the UK1 data set over performance measures

Classifier	Accuracy (%)	Specificity (%)	Sensitivity (%)	Precision (%)	$F1$-measure (%)	Expt.
MOPSO-CS(SEN)	73.19	88.07	**31.64**	48.71	**38.36**	1
MOPSO-CS(ACC)	74.79	96.41	14.43	59.66	22.97	1
NaiveBayes	73.88	95.60	13.30	51.80	21.20	10
LR	74.53	97.90	9.30	61.20	16.10	9
SVM	**75.18**	**98.80**	9.30	**73.20**	16.50	3
ANN	74.29	97.30	9.90	57.10	16.90	5
DT	75.10	98.60	9.60	70.50	16.90	5
CART	74.61	97.50	10.80	60.30	18.40	5
Bagging-DT	73.96	96.50	11.10	52.90	18.40	5
Bagging-ANN	74.69	98.40	8.40	65.90	14.80	5
RF	73.71	91.80	23.20	50.30	31.80	1

The bold characters indicate the largest value in a column

Table 5 Results on the UK2 data set over performance measures

Classifier	Accuracy (%)	Specificity (%)	Sensitivity (%)	Precision (%)	$F1$-measure (%)	Expt.
MOPSO-CS(SEN)	58.58	51.52	65.63	60.02	59.64	2
MOPSO-CS(ACC)	**64.32**	68.76	59.88	65.93	62.56	10
NaiveBayes	60.06	51.40	**68.70**	58.60	**63.20**	9
LR	61.76	60.40	63.20	61.40	62.30	10
SVM	63.31	70.00	56.70	65.40	60.70	10
NN	61.45	62.80	60.10	61.80	60.90	10
DT	60.68	**92.00**	29.40	**78.50**	42.80	9
CART	59.75	81.70	37.80	67.40	48.40	2
Bagging-DT	60.53	65.60	55.40	61.70	58.40	5
Bagging-ANN	62.23	66.30	58.20	63.30	60.60	10
RF	63.31	66.90	59.80	64.30	62.00	8

The bold characters indicate the largest value in a column

MOPSO-CS(SEN) excel those of all other counterparts. RF achieves the highest accuracy and best precision. The accuracy of MOPSO-CS(ACC) excel those of NaiveBayes and DT. Moreover, the accuracy of MOPSO-CS(SEN) is above 75%.

In the case of the Taiwan data set, MOPSO-CS(ACC) performs better than all classifies in terms of accuracy, specificity and precision (see Table 7). Meanwhile, MOPSO-CS(SEN) achieves the highest sensitivity, and its accuracy is defeated by most of the counterparts. In this experiment, the accuracy of MOPSO-CS(SEN) is still acceptable because it is better than the industry standard LR.

Another conclusion emerges from the results of our experiments. For different data sets, data standardization has different effects on improving the accuracy of data classification. For the UK1 data set, data standardization is not able to improve the accuracy, because most of the indexes are odd, except for NaiveBayes, as shown in the last column in Table 4. For the UK2 data set, data standardization is effective for improving the accuracy, because many of the indexes are even as shown in the last column in Table 5. For the German and Taiwan data sets, because the number of odd index is very close to the number of even index as shown in Tables 6 and 7, it is hard to say whether standardization is helpful in improving the accuracy.

5.2.4 Comparing with other algorithms focusing on credit scoring

In this subsection, MOPSO-CS is compared with two competitive approaches focusing on credit scoring proposed by former literatures. For the UK1 data set, we compare MOPSO-CS with multiple criteria and multiple constraint-level programming (MC2) presented by He et al. (2010). Similarly as MOPSO-CS, MC2 is an approach based on linear discriminate model. According to the results shown in Table 8, MOPSO-CS is able to improve the accuracy for the classifiers based on linear discriminate model. The best of accuracy obtained by MC2 (M11) is 65.14%, and MOPSO-CS(ACC) increases it by 9.65%. Although the best

Table 6 Results on the German data set over performance measures

Classifier	Accuracy (%)	Specificity (%)	Sensitivity (%)	Precision (%)	$F1$-measure (%)	Expt.
MOPSO-CS(SEN)	75.45	83.03	**57.76**	59.61	**58.37**	5
MOPSO-CS(ACC)	76.00	87.71	48.67	62.93	54.89	10
NaiveBayes	75.70	86.60	50.30	61.60	55.40	1
LR	76.90	88.60	49.70	65.10	56.30	1
SVM	77.10	**92.00**	42.30	69.40	52.60	4
ANN	76.00	89.10	45.30	64.20	53.10	10
DT	74.90	88.30	43.70	61.50	51.10	10
CART	76.40	88.30	48.70	64.00	55.30	10
Bagging-DT	76.30	88.10	48.70	63.80	55.20	9
Bagging-ANN	77.10	88.10	51.30	65.00	57.40	1
RF	**77.60**	91.70	44.70	**69.80**	54.50	2

The bold characters indicate the largest value in a column

Table 7 Results on the Taiwan data set over performance measures

Classifier	Accuracy (%)	Specificity (%)	Sensitivity (%)	Precision (%)	$F1$-measure (%)	Expt.
MOPSO-CS(SEN)	66.91	67.51	**66.26**	67.78	66.59	4
MOPSO-CS(ACC)	**70.42**	**85.71**	55.13	**79.63**	65.03	5
NaiveBayes	67.20	84.60	49.80	76.40	60.30	6
LR	66.67	76.30	57.00	70.60	63.10	9, 10
SVM	68.92	79.70	58.20	74.10	65.20	4
ANN	68.63	80.90	56.30	74.70	64.20	9, 10
DT	68.93	75.50	62.40	71.80	**66.80**	3
CART	68.55	74.00	63.10	70.80	66.70	3, 4
Bagging-DT	68.13	75.00	61.20	71.00	65.80	9, 10
Bagging-ANN	68.72	81.40	56.10	75.10	64.20	3, 4
RF	68.55	75.80	61.30	71.70	66.10	7

The bold characters indicate the largest value in a column

of sensitivity of MC2 (M2) is 85.37%, the accuracy of M2 is unacceptable. The sensitivity of MOPSO-CS(SEN) is 31.64%, meanwhile its accuracy is above 70%.

For the German and Taiwan data sets, we compare MOPSO-CS with XGBoost presented by Xia et al. (2017). From the results shown in Table 9, MOPSO-CS outperforms XGBoost on sensitivity. The best sensitivity of XGBoost (XGBoost-GS) is 50.21%, and MOPSO-CS(SEN) improves it by 7.55%. Meanwhile, the accuracy of MOPSO-CS(SEN) is only 1.38% lower than XGBoost-GS. Compared with XGBoost, MOPSO-CS is the best for predicting "Bad" customers with a satisfied accuracy rate for the German data set.

In the case of the Taiwan data set, as shown in Table 10, MOPSO-CS(ACC) performs better than XGBoost on accuracy and specificity. Meanwhile, the accuracy and sensitivity of MOPSO-CS(SEN) are defeated by XGBoost. It provides some evidences that MOPSO-CS(SEN) is not good at handling balanced data sets.

5.2.5 Statistical test and computation time

Nonparametric Friedman test is employed to examine whether the selected eleven classifiers are statistically different. The statistic of Friedman test is calculated as follows:

$$\chi_F^2 = \frac{12D}{K(K+1)}\left[\sum_{j=1}^{K}\left(\sum_{i=1}^{D} r_j^i\right)^2 - \frac{K(K+1)^2}{4}\right] \tag{9}$$

where D and K denote the numbers of the experiments and the classifiers and $D = 10, K = 11$. r_j^i is the rank of classifier j on experiment i. In this paper, the rank is based on the accuracy that each algorithm achieved, to the algorithm with the highest accuracy, rank 1; to the algorithm with the second highest, rank 2, etc.

If χ_F^2 is larger than a critical value, the null hypothesis is rejected. From the results shown in the last line of Table 11, we verify that the classifiers are statistically different.

Table 8 Results for MOPSO-CS versus MC2 on the UK1 data set

Classifier	Accuracy (%)	Specificity (%)	Sensitivity (%)
MOPSO-CS(SEN)	73.19	88.07	31.64
MOPSO-CS(ACC)	**74.79**	**96.41**	14.43
M1	63.43	77.82	23.21
M2	37.64	20.53	**85.37**
M1-1	64.98	72.95	42.70
M1-2	42.13	39.47	49.53
M11	65.14	73.28	42.39

The bold characters indicate the largest value in a column

Table 9 Results for MOPSO-CS versus XGBoost on the German data set

Classifier	Accuracy (%)	Specificity (%)	Sensitivity (%)
MOPSO-CS(SEN)	75.45	83.03	**57.76**
MOPSO-CS(ACC)	76.00	87.71	48.67
XGBoost-MS	76.85	89.23	47.98
XGBoost-GS	76.83	88.24	50.21
XGBoost-RS	77.18	90.43	46.27
XGBoost-TPE	**77.34**	**90.65**	46.29

The bold characters indicate the largest value in a column

Table 10 Results for MOPSO-CS versus XGBoost on the Taiwan data set

Classifier	Accuracy (%)	Specificity (%)	Sensitivity (%)
MOPSO-CS(SEN)	66.91	67.51	66.26
MOPSO-CS(ACC)	**70.42**	**85.71**	55.13
XGBoost-MS	69.15	58.66	**79.65**
XGBoost-GS	68.95	63.01	74.89
XGBoost-RS	69.35	62.11	72.66
XGBoost-TPE	69.36	62.19	76.54

The bold characters indicate the largest value in a column

With regard to average rank, MOPSO-CS(ACC) achieves the best average rank for the UK2 data set, and second best after bagging-ANN for the Taiwan data set. It proves that MOPSO-CS(ACC) performs well in balanced data sets. For the UK1 and German data sets, compared with the other two comprehensible classifiers, MOPSO-CS(ACC) is excel than DT and below CART. Because MOPSO-CS(SEN) is focusing on improving the sensitivity of data classification, the average rank of MOPSO-CS(SEN) is not ideal, and only above NaiveBayes.

Table 12 shows the computation time of a tenfold cross-validation process for the classifiers. Because MOPSO-CS is an algorithm based on iteration, and it consumes less computation time than ANN and bagging-ANN, but more than the other seven counterparts. Furthermore, similarly as SVM, MOPSO-CS is not sensitive for the number of attributes in data sets. The number of attributes in the German data set is approximately twice of that of the UK1 data set, but the computation time of MOPSO-CS dealing with the German data set is less than that of MOPSO-CS handling with the UK1 data set. Because of process for finding cutoff, the computation time of MOPSO-CS(SEN) is more than that of MOPSO-CS(ACC).

Through the computation experiments, we can observe that MOPSO-CS is more flexible than all eleven counterparts. By setting weight w_G, w_B, MOPSO-CS is able to trade-off between specificity and sensitivity for data classification. Compared with NaiveBayes, LR, SVM, ANN, DT, CART, bagging, RF, MC2 and XGBoost, MOPSO-CS(SEN) can obtain more superior sensitivity which is important for credit scoring problem. Moveover, MOPSO-CS(ACC) achieves the better accuracy, especially for balanced data sets.

6 Conclusion and future work

Traditional credit scoring models are usually based on distance; in these models, there are two kinds of objectives, one is to minimize the internal distance and the other is to maximize the external distance. But the solution with minimum internal distance or maximum external distance is not able to be proved to classify the samples correctly. In this

Table 11 Average rank of each classifier on each data set

Classifier	UK1	UK2	German	Taiwan
MOPSO-CS(SEN)	8.90	9.20	7.80	9.80
MOPSO-CS(ACC)	4.50	1.70	7.15	3.50
NaiveBayes	10.1	9.80	4.50	9.50
LR	3.70	6.55	2.95	8.40
SVM	2.05	5.65	6.90	5.90
ANN	7.10	5.80	8.30	4.40
DT	4.80	6.35	9.05	5.00
CART	4.30	6.00	5.15	4.40
Bagging-DT	8.10	5.05	5.70	6.10
Bagging-ANN	4.40	5.15	3.70	3.30
RF	8.05	4.75	4.80	5.70
χ_F^2 (Sig.)	59.429 (.000)	42.872 (.000)	35.606 (.000)	47.473 (.000)

Table 12 Computation time of the classifiers

Classifier	UK1 (s)	UK2 (s)	German (s)	Taiwan (s)
MOPSO-CS(SEN)	38.1	7.9	24.1	567.7
MOPSO-CS(ACC)	17.6	10.1	14.9	73.9
NaiveBayes	0.2	0.2	0.3	1.2
LR	0.6	0.4	0.8	2.5
SVM	4.9	1.5	1.8	256.4
ANN	10.9	6.0	20.0	140.6
DT	0.2	0.2	0.6	3.3
CART	1.2	0.8	1.3	10.5
Bagging-DT	2.9	1.3	1.9	25.7
Bagging-ANN	106.0	55.9	196.1	1315.2
RF	4.1	2.1	2.8	24.2

paper, we define a metric based on misclassification number to measure the quality of the classification, and then a novel multi-objective credit scoring model is proposed. By setting the weight in our model, we can locate the trade-off between specificity and sensitivity of credit scoring problem.

In the linear discriminant credit scoring model, finding the optimal cutoff is a critical issue. It is difficult to solve this problem using linear programming approach (He et al. 2010). But in our credit scoring approach, because the credit score of each sample is able to be calculated before the cutoff is optimized, it is easy to find the optimal cutoff with minimum misclassification number.

To credit scoring problem, it is important for credit scoring model to provide the credit score which is easy to be comprehended by credit applicants. Compared with black box technologies such as ANN and SVM, the credit score function in our approach is more comprehensible. Finally, our example and experimental studies based on benchmark data sets and real-world data sets confirm that our proposed method outperforms the counterparts in term of sensitivity while maintaining acceptable accuracy.

Future work can be carried out in several directions. Firstly, we would like to apply our proposed model to other application domains: such as life sciences and social sciences. Secondly, our model is designed for dealing with a two-group classification problem in this paper, and it could be extended to multi-group classification problems. Moreover, as well as feature selection, feature extraction may lead to better predictive performance that we would like to study and apply.

Funding This study was funded by Zhejiang provincial education department project (Y201636906), Ningbo innovative team project (2016C11024), Zhejiang provincial natural science foundation of China (Y16G010035) and national natural science foundation of China (71271191).

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Akkoc S (2012) An empirical comparison of conventional techniques, neural networks and the three stage hybrid Adaptive Neuro Fuzzy Inference System (ANFIS) model for credit scoring analysis: the case of Turkish credit card data. Eur J Oper Res 222:168–178

Aliehyaei R, Khan S (2014) Ant colony optimization, genetic programming and a hybrid approach for credit scoring: a comparative study. In: 8th international conference on software, knowledge, information management and applications (SKIMA), pp 1–5

Bahnsen AC, Aouada D (2014) Example-dependent cost-sensitive logistic regression for credit scoring. In: 13th international conference on machine learning and applications (ICMLA), pp 263–269

Bellotti T, Crook J (2009) Support vector machines for credit scoring and discovery of significant features. Expert Syst Appl 36:3302–3308

Bhattacharyya S, Jha S, Tharakunnel K, Westland JC (2011) Data mining for credit card fraud: a comparative study. Decis Support Syst 50:602–613

Blanco A, Mejias RP, Lara J, Rayo S (2013) Credit scoring models for the microfinance industry using neural networks: evidence from Peru. Expert Syst Appl 40:356–364

Breiman L (1996) Bagging predictors. Mach Learn 24:123–140

Chen MC, Huang SH (2003) Credit scoring and rejected instances reassigning through evolutionary computation techniques. Expert Syst Appl 24:433–441

Chi BW, Hsu CC (2012) A hybrid approach to integrate genetic algorithm into dual scoring model in enhancing the performance of credit scoring model. Expert Syst Appl 39:2650–2661

Das S, Abraham A, Konar A (2008) Automatic kernel clustering with a multi-elitist particle swarm optimization algorithm. Pattern Recognit Lett 29:688–699

Dehuri S, Cho SB (2009) Multi-criterion Pareto based particle swarm optimized polynomial neural network for classification: a review and state-of-the-art. Comput Sci Rev 3:19–40

Farquad M, Bose I (2012) Preprocessing unbalanced data using support vector machine. Decis Support Syst 53:226–233

Fernandes GB, Artes R (2016) Spatial dependence in credit risk and its improvement in credit scoring. Eur J Oper Res 249:517–524

Goh CK, Tan KC, Liu DS, Chiam SC (2010) A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. Eur J Oper Res 202:42–54

Hajek P (2011) Municipal credit rating modelling by neural networks. Decis Support Syst 51:108–118

Hand DJ, Henley WE (1997) Statistical classification methods in consumer credit scoring: a review. J R Stat Soc Ser A (Stat Soc) 160:523–54

Harris T (2015) Credit scoring using the clustered support vector machine. Expert Syst Appl 42:741–750

He J, Zhang YC, Shi Y (2010) Domain-driven classification based on multiple criteria and multiple constraint-level programming for intelligent credit scoring. IEEE Trans Knowl Data Eng 22:826–838

Huang CL, Chen MC, Wang CJ (2007) Credit scoring with a data mining approach based on support vector machines. Expert Syst Appl 33:847–856

Huang JJ, Tzeng GH, Ong CS (2006) Two-stage genetic programming (2SGP) for the credit scoring model. Appl Math Comput 174:1039–1053

Jo H, Han I (1997) Bankruptcy prediction using case-based reasoning, neural networks, and discriminant analysis. Expert Syst Appl 13:97–108

Kennedy J, Eberhart R (1995) Particle swarm optimization. Proc IEEE Int Conf Neural Netw 4:1942–1948

Kim JC, Kim DH, Kim J, Ye JS, Lee HS (2012) Segmenting the Korean housing market using multiple discriminant analysis. Constr Manag Econ 18:2650–2661

Koutanaei FN, Sajedi H, Khanbabaei M (2015) A hybrid data mining model of feature selection algorithms and ensemble learning classifiers for credit scoring. J Retail Consum Serv 27:11–23

Lee TS, Chen IF (2005) A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. Expert Syst Appl 28:743–752

Lee TS, Chen NJ (2002) Investigating the information content of non-cash-trading index futures using neural networks. Expert Syst Appl 22:225–234

Lessmann S, Baesens B, Seow HV, Thomas LC (2015) Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research. Eur J Oper Res 247:124–136

Martens D, Baesens B, Gestel TV, Vanthienen J (2007) Comprehensible credit scoring models using rule extraction from support vector machines. Eur J Oper Res 183:1466–1476

Nanni L, Lumini A (2009) An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. Expert Syst Appl 36:3028–3033

Niklis D, Doumpos M, Zopounidis C (2014) Combining market and accounting-based models for credit scoring using a classification scheme based on support vector machines. Appl Math Comput 234:69–81

Olson DL, Delen D, Meng Y (2012) Comparative analysis of data mining methods for bankruptcy prediction. Decis Support Syst 52:464–473

Omkar SN, Mudigere D, Naik GN, Gopalakrishnan S (2008) Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures. Comput Struct 86:1–14

Omran MGH, Salman A, Engelbrecht AP (2006) Dynamic clustering using particle swarm optimization with application in image segmentation. Pattern Anal Appl 8:332–344

Rezac M (2011) Advanced empirical estimate of information value for credit scoring models. Acta Univ Agric Silvic Mendel Brunenis 35:267–274

Thomas LC (2000) A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. Int J Forecast 16:149–172

Tomczak JM, Zięba M (2015) Classification restricted Boltzmann machine for credit comprehensible scoring model. Expert Syst Appl 42:1789–1796

Wang G, Ma J, Huang LH, Xu KQ (2012) Two credit scoring models based on dual strategy ensemble trees. Knowl Based Syst 26:61–68

West D (2011) Neural network credit scoring models. Comput Oper Res 27:1131–1152

Xia Y, Liu C, Li Y, Liu N (2017) A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. Expert Syst Appl 78:225–241

Xiao H, Xiao Z, Wang Y (2016) Ensemble classification based on supervised clustering for credit scoring. Appl Soft Comput 43:73–86

Yeh CC, Lin FY, Hsu CY (2012) A hybrid KMV model, random forests and rough set theory approach for credit rating. Knowl Based Syst 33:166–172

Zhao ZY, Xu SX, Kang BH, Kabir MMJ, Liu YL, Wasinger R (2015) Investigation and improvement of multi-layer perception neural networks for credit scoring. Expert Syst Appl 42:3508–3516