A model for software defect prediction using support vector machine based on CBA

Xiaotao Rong, Feixiang Li and Zhihua Cui*

Complex System and Computational Intelligence Laboratory, Taiyuan University of Science and Technology, 030024, China
Email: rongxtdk@163.com
Email: lifeixiang126@126.com
Email: cuizhihua@gmail.com
*Corresponding author

Abstract: Software defection prediction is not only crucial for improving software quality, but also helpful for software test effort estimation. As is well-known, 80% of the fault happens in 20% of the modules. Therefore, we need to find out the most error prone modules accurately and correct them in time to save time, money, and energy. Support vector machine (SVM) is an advanced classification method that fits the defection classification. However, studies show that, the value of parameters of SVM model has a remarkable influence on its classification accuracy and the selection process lacks theory guidance that makes the SVM model uncertainty and low efficiency. In this paper, a CBA-SVM software defect prediction model is proposed, which take advantage of the non-linear computing ability of SVM model and optimisation capacity of bat algorithm with centroid strategy (CBA). Through the experimental comparison with other models, CBA-SVM is proved to have a higher accuracy.

Keywords: software defect prediction; centroid strategy; bat algorithm; SVM; support vector machine.


Biographical notes: Xiaotao Rong is now pursuing her Master degree in Computer Science and Technology Department in the Taiyuan University. Her current research interests focus on the areas of bio-inspired algorithms and software testing.

Feixiang Li is now pursuing his Master degree in Computer Science and Technology Department in the Taiyuan University. His current research interests focus on the area of bio-inspired algorithms.

Zhihua Cui is a Professor at the School of Computer Science and Technology in the Taiyuan University of Science and Technology, China. He received his PhD in System Engineering from the Xi’an Jiaotong University in 2008. His current

Copyright © 2016 Inderscience Enterprises Ltd.
research interests are in swarm intelligence and bio-inspired algorithms. He has published nearly 60 international journal and conference papers.

1 Introduction

With the development of the technology of software engineering, software function is becoming increasingly complex. It means that more modules in one software make more mistakes. Once the software failure happens, it will cause huge loses including the shortage of crucial functions, making the software system breakup and further more lose the trust from customers (Gondra, 2008). There is a famous principle in software industries that 80% of the defections are caused by 20% of the modules (Weyuker et al., 2008). It is the 20% drawbacks that cause these errors. So, there’s no need to check all the modules individually as long as find out the error-prone modules that can reduce development costs and optimise the allocation of resources. At the same time, the effort of test can be estimated on the basis of the number of the predicted defections (Srivastava et al., 2012).

The key of the software development is how to accurately predict those defective modules which attracts a lot attention from many areas. Software defect prediction has been one of the active areas of software engineering since 1990. The general models conduct prediction based on the feature values of all the modules. A great number of studies have been proposed, BPNN (Neumann, 2002), Clustering (Zhong et al., 2004), Bayesian model (Dai et al., 2007), and support vector machine (SVM) are used to predict software defect (Gray et al., 2009).

Support vector machine (SVM) is proposed by Vapnik (Cortes and Vapnik, 1995) to solve the binary classification problems due to the high accuracy and the outstanding ability of non-liner computing ability. SVM module is widely used for classification that based on VC theory and structural risk minimisation principle on the basis of statistical theory. Hsu et al. (2003) showed that the performance of SVM is influenced remarkably by the value settings of the parameter $C$ and $\sigma$. The traditional SVM uses trail-and-error method to determine the values of parameters that has blindness and uncertainty to the decision. Gray et al. (2009) have shown that the SVM can be used successfully as a classification method for defect prediction. Elish and Elish (2008) found that the prediction performance of SVM is generally better than, or at least comparable to these compared models.

Bat algorithm (BA) is a novel swarm intelligence meta-heuristic optimisation algorithm of simulating micro-bats echolocation behaviour, which was proposed by Yang (2010). The algorithm model is simple, fast convergence rate and has the characteristics of potential parallelism and distributed, it has been widely applied to many areas (Yang, 2012). BA has powerful search performance, but its local search is relatively poor because of lacking of communication between individuals (Yang and Gandomi, 2012). Therefore, Cao et al. (2014) proposed a new variant called bat algorithm with centroid strategy (CBA), simulation results show CBA is validity.

In this paper, CBA is used to optimise the SVM parameters called CBA-SVM model, and then CBA-SVM model is applied to predict the software defect. Thus, it provides theory guidance to the SVM parameters selection. The results of experiment designed in MATLAB program show that the performance of the model can achieve high accuracy and quick prediction for software defect classification.
A model for software defect prediction using support vector machine

The rest of this paper is organised as follows: Section 2 provides a short introduction for software prediction and the SVM is illustrated in Section 3. Section 4 presents the bat algorithm with centroid strategy. Section 5 shows the CBA-SVM model, and Section 6 and Section 7 will describe the experimental results and the conclusions.

2 Software defect prediction

With the rapid development of computer software and computer technology, software has been used in almost every aspects of social life (Zhou and Leung, 2006). Because of the existence of the invisible software that makes our lives convenient and fast. We are so dependent on the software that we do not know when and where will have a failure that can do harm to our work and life. Meanwhile, the technology is facing a crucial problem that the cost of test and maintaining of software turn out to be larger and larger along with the multi-function and high complexity software system. Software defect means that software could not achieve its function for its own shortcomings. Once a failure happens, it will cause huge loss, including great economic losses, very bad social effect and low degree of customer satisfactory (Ohlsson and Alberg, 1996).

Nevertheless, the defect can be predicted by using software defect prediction models according to the history data of the company and the features of software (Cagatay and Banu, 2009). Software defect prediction model is very helpful for decision-makings such as the allocation of resource in module verification and validation. What we need to do is to find the defect and then use more detailed methods to test the software and correct it. Therefore, people can reach such situation where software works with high reliability, maintainability and performance, short development time, low development and maintenance cost (Catal and Diri, 2009).

For an established software company, there are often some data collected from past projects or released systems. Usually, we can quantify the quality of software products based on an amount of product maintainability, reusability metrics, and stability metrics. The metrics can be got by the depth of class inheritance, coupling degree, the lines of code (Catal, 2011). These history data can be abstracted as a vector which contains the feature parameters. The value of the vector is the feature value of the past module. The last dimension is the category label. If the history module is defective, the label equals 1, otherwise –1. We use these vectors to conduct experiment in order to find a classification model. They are useful for the prediction of software defection. Nevertheless, due to the complicated situations of software development process in the early stage, the applicability and accuracy of these models are still under research.

The application of software defect prediction methods are useful for software production process, such as obtaining a highly reliable software systems, focusing more error-prone modules, allocating resources based on the predicted results and further make full use of limited resources to improve the quality of software products (Gyimthy et al., 2005).

Nowadays, a lot of software defect prediction techniques and models are proposed. These techniques include statistical methods, machine learning methods, parametric models, and mixed algorithms (Challagulla et al., 2005). Software defect prediction models mainly include Markov model, Classification and regression tree model, artificial neural network model, linear discriminant analysis model, LSTSA, and classification tree model. But these method exist some problems that yet to solve that cannot reach a satisfactory level. For example, the Markov model needs to make kinds of assumptions; poor generalisation
capability of Classification and regression tree model and the artificial neural network model does not have a unified guiding theory for the selection of network structure.

Logistic regression (LR) further improved the linear regression for the incapacity to solve the illegal probably value beyond 0 and 1. Concerning this issue, LR used logarithmic transformation that made the value not be confined to 0 to 1, which can be an arbitrary value between positive and negative infinity. LR conducted quadratic discriminant that can increase the computing speed and recognition rate that lead to a widely use in software defect prediction model. Whereas, it is more often used in large sample for the result for small sample is not ideal (Gondra, 2008).

3 Support vector machine

The main idea for SVM, aimed at binary classification problem, is looking for a hyperplane in a high-dimensional space as a parting plane for two aspects in order to ensure minimum error rate. SVM transforms a problem of computing classification function into a quadratic programming problem with constrains.

If the sample sets are linear and separable, \((x_1, y_1), \ldots, (x_m, y_m), x \in \mathbb{R}^n, y \in \{-1, 1\}\) are assumed. \(m\) stands for the number of samples and \(y\) stands for category, \(n\) stands for entered dimension. In this case, there will be a hyperplane that separate these two types of samples completely. This plane can be described as equation (1):

\[
 w_i x + b = 0. \tag{1}
\]

The symbol \(\cdot\) stands for dot product of two vectors. The parameters, \(w\) and \(b\), are the normal vector and the bias of the hyperplane. The issue mentioned above can be expressed as a general form:

\[
 f(x) = w_i x + b. \tag{2}
\]

If \(f(x) \geq 0\), the category label equals to +1, otherwise, –1. At the time of classification of the vectors, we hope the training data can be parted accurately and the distance between the closest data and hyperplane can be enlarged as big as possible. This problem can be described as a quadratic programming problem like equation (3).

\[
 \min \left( \frac{1}{2} ||w||^2 \right)
\]

\[
 \text{s.t.} (wx_i + b) - 1 \geq 0 \quad i = 1, 2, \ldots, L. \tag{3}
\]

The optimisation function is a quadratic form and the constrain conditions is liner that the issue above is a typically quadratic programming problem. It can be solved by the introducing Lagrange operator \(a_i \geq 0 i = 1, 2, \ldots, l\), and then be transformed into its dual form (Dai et al., 2007; Williams, 2003; Fung and Mangasarian, 2004; Weerahandi and Hausman, 1994).
A model for software defect prediction using support vector machine

\[
\begin{align*}
\min & \quad \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} y_i y_j (x_i, x_j) a_i a_j - \sum_{i=1}^{L} a_i \\
\text{s.t.} & \quad \sum_{i=1}^{L} y_i a_i = 0, a_i \geq 0, i = 1, 2, \ldots, L.
\end{align*}
\]

(4)

To solve and analysis this kind of constrained optimisation problems, Karush-Kuhn-Tucker (KKT) condition will play an important role. The issue mentioned above must satisfied the condition below (equation (5)).

\[
a_i \{y_i (w \ast x + b) - 1\} = 0, i = 1, 2, \ldots, L.
\]

(5)

The sample, whose equals 0, does not work in the classifying problem. It is the very samples that \(a_i > 0\) that has affect and further decide the consequence of classification. The classification function is obtained as follow:

\[
f(x) = \text{sgn}\{ (w \ast x) + b \} = \text{sgn}\left\{ \sum_{i=1}^{L} y_i \ast a_i (x_i \ast x) + b \right\}.
\]

(6)

In most cases, the object of classification problem is linearly non-separable sample set. On this occasion, the SVM firstly transforms the input space into a high dimensional space and then solve the problem by using liner classifier.

We employ kernel function to do the transformation that implicitly mapping input data points into higher dimensional feature space and to take the inner-product in that feature space. There are lots of kernel functions that applied to different algorithm. The most common function is Polynomial function, radial basis function, and dynamic kernel function and so on. In this thesis, we use radial basis function as the kernel function because of the capability of handle non-linear problems and fewer parameters required.

\[
k(x, x_i) = \exp\left( -\frac{|x - x_i|^2}{\sigma^2} \right).
\]

(7)

The parameter \(\sigma\) controls the radius of RBF. The parameter \(\sigma\) should be selected carefully for that it will lead to different problems if it is too large or too small. The regulated parameter \(\sigma\) will lead to a very high flexibility of RBF that make this function widely used. However, there is still outlier that if it is neglected the hyperplane we found is perfect. Slack variable is introduced to allow the special data be away from the hyperplane. The quadratic programming problem, we mentioned before can be showed as:

\[
\min \left( \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \right).
\]

(8)

\(C\) is the error cost parameter, a variable that determines the trade-off between minimising the training error and maximising the margin. This parameter need to be set at the beginning. If the value of \(C\) is relatively high, it will be more serious about the result. All the change will be used in the process of computing the new classification plane. And the classification function is obtained as follows:

\[
f(x) = \text{sgn}\left\{ \sum_{i=1}^{n} y_i \ast a_i k(x_i \ast x) + b \right\}.
\]

(9)
4 Bat algorithm with centroid strategy

4.1 Standard bat algorithm

The bat algorithm is a novel metaheuristic swarm intelligence optimisation method developed for the global numerical optimisation, in which the search algorithm is inspired by social behaviour of bats and the phenomenon of echolocation to sense distance.

In order to propose the standard bat algorithm, three idealised rules are used:

- All bats use echolocation to sense distance, and they also ‘know’ the difference between food/prey and background barriers in some magical way.
- Bats fly randomly with velocity $\vec{v}_i$ and a fixed frequency $f_i$ at position $\vec{x}_i$, varying wavelength $\lambda_i$ and loudness $A_i$ to hunt for prey. They can spontaneously accommodate the wavelength of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target.
- Although the loudness can change in different ways, it is supposed that the loudness varies from a minimum constant $A_{\text{min}}$ to a large value.

In bat algorithm, each bat is defined by its position $\vec{x}_i(t)$, velocity $\vec{v}_i^{(t)}$, frequency $f_i$, loudness $A_i^{(t)}$, and the emission pulse rate $r_i(t)$ in a $D$-dimensional search space. The new solutions $\vec{x}_i^{(t+1)}$ at time $t + 1$ is given by

$$x_{ik}^{(t+1)} = x_{ik}^{(t)} + v_{ik}^{(t+1)}$$

(10)

and the velocity $v_{ik}^{(t+1)}$ is updated by

$$v_{ik}^{(t+1)} = v_{ik}^{(t)} + (x_{ik}^{(t)} - p_k^{(t)}) \cdot f_i$$

(11)

where $\vec{p}^{(t)}$ is the current global best location found by all bats in the past generations. Frequency $f_i$ is dominated by:

$$f_i = f_{\text{min}} + (f_{\text{max}} - f_{\text{min}}) \cdot \beta,$$

(12)

where $\beta \in [0, 1]$ is a random number drawn from a uniform distribution, two parameters $f_{\text{min}}$ and $f_{\text{max}}$ are assigned to 0.0 and 5.0 in practical implementation, respectively, and each bat is randomly given a frequency which is drawn uniformly from $[f_{\text{min}}, f_{\text{max}}]$. The detail position generation process is explained in Figure 1.

To improve the exploitation capability, there exists one local search strategy for each bat, once a solution (e.g., $\vec{x}_j^{(t+1)}$) is selected, it will be changed as with a random walk:

$$\vec{x}_j^{(t+1)} = \vec{p}^{(t)} + \varepsilon \overline{A}(t),$$

(13)

where $\varepsilon$ is a random number uniformly distributed within $[-1, 1]$, and $\overline{A}(t)$ is the average loudness of all the bats at time step $t$.

For bat $j$, the loudness $A_i^{(t+1)}$ and the rate $r_i^{(t+1)}$ of pulse emission are updated as follows:

$$A_i^{(t+1)} = \alpha A_i^{(t)}$$

(14)

$$r_i^{(t+1)} = r_i^{(0)}[1 - \exp(-\gamma t)].$$

(15)
In this paper, \( r_i(0) = 0.9, \gamma = 0.9, \alpha = 0.99 \), and for each bat \( i, A_i(0) = 0.9 \). All of them are coming from experiments.

Based on the above description, the pseudocode of the standard bat algorithm can be described in Algorithm 1.

**Algorithm 1 Standard Bat Algorithm**

Objective function \( f(\vec{x}) = (x_1, x_2, ..., x_D)^T \);

Initialize the parameters for each bat: position \( \vec{x}_i(0) \), velocity \( \vec{v}_i(0) \), loudness \( \vec{A}_i(0) \) and rate \( \vec{r}_i(0) \);

Define repulse frequency \( f_i \) at \( \vec{x}_i(0) \);

\( t = 0; \)

\[ \text{while} \ t < \text{Largest iterations} \text{ do} \]

\[ \text{Update the velocity and position for each bat with Eq.(5)-Eq.(7);} \]

\[ \text{if} \ \text{rand} > \vec{r}_i(t) \text{ then} \]

\[ \text{Re-update the position and velocity of bat } j \text{ around the selected best solution } \vec{p}(t) \text{ with Eq.(8);} \]

\[ \text{end if} \]

\[ \text{if} \ \text{rand} < \vec{A}_i(t) \& f(\vec{x}_i(t)) \text{ < } f(\vec{p}(t)) \text{ then} \]

\[ \text{Accept the new solution } \vec{x}_i(t+1) \text{ and velocity } \vec{v}_i(t+1) \text{ with Eq.(10)}; \]

\[ \text{Increase } \vec{A}_i(t) \text{ with Eq.(9)}; \]

\[ \text{end if} \]

Rank the bats and find the current best \( \vec{p}(t) \);

\( t = t+1; \)

\[ \text{end while} \]

Output the best solution \( \vec{p}(t) \);

**4.2 Bat algorithm with centroid strategy**

According to the updating formula, the individuals are away from the optimal position to search. Figures below illustrate the change of direction and velocity. The local search can be represented by a formula (13).

We can observe from the above picture that the local search is conducted in the innermost circle and the bat fly away from the outmost circle. We can conclude that the standard algorithm cannot coverage the domain of definition such as the shaded area (Yang, 2012).

In order to improve the performance of the global search of algorithm bat, the update formula (11) will be modified by deleting its speed memory, obtaining the following update speed mode:

\[ v_i^{t+1} = (x_i^t - x^*) \cdot f_i. \] (16)

For standard bat algorithm, the individual is renewed when there is a better position. Therefore, from this perspective, the individual position of the bat algorithm is supposed to be the optimal position of the individual. Thus, the best position is the current position of the bat. According to equation (16) the velocity updating formula, the bat speed in the moment \( t + 1 \) equals 0 when the bats will stand still. Under the circumstance, we call that bat 0-speed bats and the individual’s position is modified by using the following manner:

\[ x_j^{t+1} = x_{\text{min}} + \text{rand}(1) \cdot (x_{\text{max}} - x_{\text{min}}). \] (17)
In which, $x_j^{t+1}$ represents the position in $t+1$ iterative. $[x_{\text{min}}, x_{\text{max}}]^D$ expressed its position in the D-dimensional search space, $\text{rand}(1)$ is uniformly distributed random numbers.

**Figure 1** Illustration of bat algorithm

![Illustration of bat algorithm](image1)

Apparently, the position for search can be generated randomly in the domain of definition that can ensure the global convergence for the whole area. Usually, people use two types of information in the decision-making process. One of them is the individual’s own information and the other is the information from other individuals. In other words, use their own information and information of other individuals to make decisions in the decision-making process. Inspired by this, we introduce the arithmetic centroid policy. The centroid sample takes advantage of three adjacent vertices of the triangle information and to predict the next position that can effectively improve the utilisation ration of individual information.

Centroid-based bat algorithm is globally convergent. In order to improve search efficiency, bat algorithm local search procedure (only in search of an empty area in Figure 2) made some adjustments to improve the search efficiency in shaded area by introduce the centroid strategy.

**Figure 2** Illustration of centroid strategy

![Illustration of centroid strategy](image2)
Of course, in order to ensure local search efficiency in the late stages, the centroid strategies only be employed in the early period. The procedure will introduce the concept of centroid ratio. When \( t \leq \lambda \cdot \text{Largest\_generation} \) execute the centroid strategies, otherwise, execute local search (13).

Firstly, we select individuals in the group. If bat \( j \) is selected, then predict the better fitness individuals by doing the formula bellow:

\[
x'_t = \frac{1}{m} \sum_{i=1}^{m} x_i^t.
\]

Among them, the value of \( x_i^t \) is the predicted location that its performance is better than before.

Obviously, from a mathematical point of view, it is equivalent to the arithmetic average of these positions, so called arithmetic centroid strategies.

The centroid strategies can be found in Figure 3, the performance of P2, P3, P4, P5 four bats are better than the bat P1 and the computing result is P1'.

**Figure 3** Illustration of centroid generation process (see online version for colours)

The procedure of improved bat algorithm is described as:

- **Step 1**: Determine the objective function and initialise the basic parameters.
- **Step 2**: Initialise bat population position \( x_i^0 (i = 1, 2, \ldots, n) \) and speed \( v_i^0 \) according to the formula (12) generates each bat pulse frequency \( f_i^0 \) pulse emission frequency \( r_i^0 \) and loudness \( A_i^0 \) and then calculate the fitness and find out the optimal population individuals.
- **Step 3**: Ordinary bat individual in accordance with the formula (16) update bat speed, and then according to the formula (10) update bat individual position. 0-speed bats according to equation (17) update location.
- **Step 4**: For each individual bat, generates a random number \( \text{rand}(1) \) and if \( \text{rand}(1) \geq R_{ij} \) go to the next step, otherwise, the processing shifts Step 6.
- **Step 5**: Analyse that if current generation is less than the proportion of the centroid and then select individuals according to formula (18) centroid optimisation or according to the formula (13) to re-generate the disturbance in the vicinity of the current best individual.
- **Step 6**: Calculating the fitness value of all bats individual at new location.
Step 7: Check whether the new fitness value is better than the best value in its history. If so, accept this new solution, and adjust $r^t_i$ based on formula (15).

Step 8: Updating the global best position.

Step 9: Determining whether the termination condition is satisfied. If it is, the algorithm is terminated and output the result, otherwise transfer to Step 3.

5 Support vector machine prediction model

5.1 Based on bat algorithm with centroid strategy

The performance of SVM model, which uses RBF as its kernel function, is largely dependent on parameter $C$ and $\sigma$. But SVM model usually adopts the trial-and-error method to determine its parameters, which is lacked of guidance in theory (Weerahandi and Hausman, 1994). So we use CBA to optimise the parameters of SVM in order to improve the accuracy of the prediction model. The main idea of the optimisation algorithm is that a pair of SVM parameters is considered as a particle in CBA, particle updates itself, until the algorithm reaches its terminal condition. The index of measuring particle is the accuracy of the prediction model under this pair of parameters. The concrete steps of the SVM parameters optimisation algorithm based on CBA are as following:

Step 1: Setting the parameters value of CBA, such as particle number, cycle number, and so on. Randomly generate initial particles.

Step 2: Compute the fitness value of each particle.

Step 3: Update and with the fitness value of each particle using equations (10) and (11).

Step 4: Each particle moves to its next position.

Step 5: Terminal condition checking. If the maximum iteration is meet, the cycle stops, output the optimal parameters. Otherwise, turn to step 2. These steps are illustrated by Figure 4.

On the basis of SVM parameters optimisation algorithm above, a new model for software defect prediction named CBA-SVM is presented in this paper. The steps of the model are as following:

Step 1: Achieve the dataset.

Step 2: Establish the software defect prediction model based on SVM.

Step 3: Use SVM parameters optimisation algorithm based on CBA to obtain the optimal SVM parameters.

Step 4: Resort to optimised SVM prediction model to predict the dataset. If the result of prediction reaches the terminal condition, the cycle stops, output the result of software detects prediction. Otherwise, go to step 3. The overall flowchart based on area centroid strategy bat algorithm and SVM model is shown in Figure 5.

The overall flowchart CBA-SVM model is shown in Figure 2.
6 Simulation

6.1 Datasets

In this paper, we will use the datasets from National Aeronautics and Space Administration (NASA). Four kinds of datasets are selected for the evaluation of the proposed method. The details of the datasets are described in Table 1. The dataset are also available from the website of NASA (Chapman et al., 2004)
Table 1: Datasets of defect prediction model

<table>
<thead>
<tr>
<th>Data</th>
<th>Number of modules</th>
<th>Defect rate (%)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JM1</td>
<td>10,878</td>
<td>19.30</td>
<td>A real-time predictive ground system</td>
</tr>
<tr>
<td>CM1</td>
<td>505</td>
<td>9.70</td>
<td>Spacecraft instrumentation system</td>
</tr>
<tr>
<td>KC1</td>
<td>2107</td>
<td>15.40</td>
<td>A storage management system</td>
</tr>
<tr>
<td>PC1</td>
<td>1107</td>
<td>6.30</td>
<td>Flight software for an earth orbiting satellite</td>
</tr>
</tbody>
</table>

6.2 Evaluation methods

Software modules can be divided into defective and non-defective modules. We can calculate the accuracy from the classification model according to Table 2.

Table 2: Fuzzy matrix

<table>
<thead>
<tr>
<th>Classification</th>
<th>Defective modules</th>
<th>Non-defective modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real value (TP)</td>
<td>True positive</td>
<td>False negative</td>
</tr>
<tr>
<td>Defective modules</td>
<td>examples (TP)</td>
<td>examples (FN)</td>
</tr>
<tr>
<td>Non-defective modules</td>
<td>False positive</td>
<td>True negative</td>
</tr>
<tr>
<td></td>
<td>examples (FP)</td>
<td>examples (TN)</td>
</tr>
</tbody>
</table>

6.3 Evaluation methods

Software modules can be divided into defective and non-defective modules. We can calculate the accuracy from the classification model according to Table 2.

Accuracy is used to measure the performance of prediction models in our work. Accuracy is defined as the ratio of the number of modules correctly predicted to the total number of modules. It can be calculated by the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}.$$  \hfill (19)

Precision is defined as the ratio of the number of defect modules correctly predicted to the total number of defect modules. It can be calculated by the following formula:

$$\text{Precision} = \frac{TP}{TP + FP}.$$  \hfill (20)

Recall is defined as the ratio of the number of real defect modules predicted to the number of defect modules correctly. It can be calculated by the following formula:

$$\text{Recall} = \frac{TP}{TP + FN}.$$  \hfill (21)

F-measure is defined as harmonic mean of precision and recall. It can be calculated by the following formula:
As mentioned above, an SVM with an RBF kernel function requires the selection of optimal values for parameters $C$ and $\gamma$ for maximal performance. We use changing range bat algorithm to optimise it, then the pair of parameters values which yield the highest average accuracy are taken as the optimal parameters and used in CBA-SVM model for classification.

### 6.4 Simulation results

Experiments are used several methods to establish a classification model, they are back-propagation neural network (BPNN), logistic regression (LR), cluster analysis (CA) and linear discriminant analysis (LDA). These experiments results are compared with SVM and bat algorithm with centroid strategy (CBA-SVM). In our experiments, the largest generation is set as 50 for saving time and $r_i(0) = 0.9, \gamma = 0.9, \alpha = 0.99$. Because datasets has large numbers of modules, we select 200 modules randomly as the training sample in the dataset, and select 200 modules randomly as the prediction sample.

The experimental results of some methods is given in Tables 3–6 on datasets JM1, CM1, KC1 and PC1. We can draw the conclusion that this new method proposed by us can get better accuracy, precision, recall and F-measure than other prediction model in these four datasets. The traditional SVM model may also obtain relatively good classification results, however it is often limited from the model parameters. To some extent, CBA-SVM not only overcomes the influence of the dataset redundant and SVM model parameters, simulation results show its efficiency, but also strengthens the global convergence by computing centroid.

**Table 3** Comparison among prediction models results based on JM1

<table>
<thead>
<tr>
<th>Prediction model</th>
<th>Accuracy</th>
<th>Prediction</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBA-SVM</td>
<td>94.0</td>
<td>95.6</td>
<td>97.7</td>
<td>96.6</td>
</tr>
<tr>
<td>SVM</td>
<td>86.0</td>
<td>95.7</td>
<td>88.7</td>
<td>91.8</td>
</tr>
<tr>
<td>BPNN</td>
<td>89.0</td>
<td>91.3</td>
<td>78.5</td>
<td>84.4</td>
</tr>
<tr>
<td>CA</td>
<td>88.8</td>
<td>86.6</td>
<td>74.7</td>
<td>80.3</td>
</tr>
<tr>
<td>LR</td>
<td>83.2</td>
<td>84.8</td>
<td>73.1</td>
<td>78.4</td>
</tr>
<tr>
<td>LDA</td>
<td>80.4</td>
<td>82.5</td>
<td>71.6</td>
<td>75.8</td>
</tr>
</tbody>
</table>

**Table 4** Comparison among prediction models results based on CM1

<table>
<thead>
<tr>
<th>Prediction model</th>
<th>Accuracy</th>
<th>Prediction</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBA-SVM</td>
<td>94.0</td>
<td>98.0</td>
<td>94.0</td>
<td>96.0</td>
</tr>
<tr>
<td>SVM</td>
<td>90.5</td>
<td>97.0</td>
<td>92.1</td>
<td>94.5</td>
</tr>
<tr>
<td>BPNN</td>
<td>90.5</td>
<td>94.2</td>
<td>88.2</td>
<td>85.4</td>
</tr>
<tr>
<td>CA</td>
<td>88.2</td>
<td>89.2</td>
<td>82.4</td>
<td>78.7</td>
</tr>
<tr>
<td>LR</td>
<td>82.4</td>
<td>81.6</td>
<td>78.5</td>
<td>72.1</td>
</tr>
<tr>
<td>LDA</td>
<td>79.2</td>
<td>78.7</td>
<td>71.8</td>
<td>70.6</td>
</tr>
</tbody>
</table>
Table 5  Comparison among prediction models results based on KC1

<table>
<thead>
<tr>
<th>Prediction model</th>
<th>Accuracy</th>
<th>Prediction</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBA-SVM</td>
<td>95.5</td>
<td>98.8</td>
<td>96.1</td>
<td>97.4</td>
</tr>
<tr>
<td>SVM</td>
<td>91.0</td>
<td>94.3</td>
<td>95.4</td>
<td>94.9</td>
</tr>
<tr>
<td>BPNN</td>
<td>90.1</td>
<td>92.1</td>
<td>88.5</td>
<td>89.7</td>
</tr>
<tr>
<td>CA</td>
<td>86.8</td>
<td>85.3</td>
<td>76.4</td>
<td>81.2</td>
</tr>
<tr>
<td>LR</td>
<td>81.9</td>
<td>80.4</td>
<td>72.2</td>
<td>79.1</td>
</tr>
<tr>
<td>LDA</td>
<td>79.4</td>
<td>72.4</td>
<td>70.1</td>
<td>71.6</td>
</tr>
</tbody>
</table>

Table 6  Comparison among prediction models results based on PC1

<table>
<thead>
<tr>
<th>Prediction model</th>
<th>Accuracy</th>
<th>Prediction</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBA-SVM</td>
<td>95.5</td>
<td>99.4</td>
<td>95.6</td>
<td>97.4</td>
</tr>
<tr>
<td>SVM</td>
<td>88.5</td>
<td>97.6</td>
<td>89.3</td>
<td>93.3</td>
</tr>
<tr>
<td>BPNN</td>
<td>91.2</td>
<td>94.9</td>
<td>89.4</td>
<td>87.6</td>
</tr>
<tr>
<td>CA</td>
<td>87.4</td>
<td>89.5</td>
<td>82.4</td>
<td>80.3</td>
</tr>
<tr>
<td>LR</td>
<td>81.7</td>
<td>85.1</td>
<td>79.2</td>
<td>76.5</td>
</tr>
<tr>
<td>LDA</td>
<td>80.2</td>
<td>76.3</td>
<td>71.0</td>
<td>70.8</td>
</tr>
</tbody>
</table>

7  Conclusion

This paper proposes hybrid SVM model with changing range bat algorithm for software defect prediction, which is benefit for overcoming the influence of the redundant datasets and increasing the prediction accuracy. Experiments prove that this method can get better performance than other traditional methods.

Acknowledgement

This paper is supported by National Natural Science Foundation of China under Grant No. 61003053 and the Program for the Top Young Academic Leaders of Higher Learning Institutions of Shanxi.

References


A model for software defect prediction using support vector machine


