Defect Prediction Model for Object Oriented Software Based on Particle Swarm Optimized SVM

Defect Prediction Model for Object Oriented Software Based on Particle Swarm Optimized SVM

Yanan Wang[1], Ran Zhang[2, 1, *], Xiangzhou Chen[1], Shanjie Jia[3], Huixia Ding[1], Qiao Xue[1] and Ke Wang[2]

[1]China Electric Power Research Institute, Beijing 100192, China

[2]North China Electric Power University, Beijing 102206, China

[3]State Grid Shandong Electric Power Company Economic and Technological Research Institute, Shandong 250000, China

* Corresponding Author: Ran Zhang; *email: zrzhangran111@163.com*

Abstract. In terms of the security problem of power information system，this paper analysed the importance of the software defect prediction method in object-oriented software development，and proposed a software prediction model based on particle swarm optimized Support Vector Machine (SVM) corresponding to the features of object-oriented software. The model mainly consists of three parts: the first is the pre-processing module which normalizes the original data and selects feature，then the second is adaptive inertia weight particle swarm module which optimizes the parameters of SVM with the prediction accuracy as the fitness. Finally，the last SVM classification module predicts categories of reduced-dimension data using the optimal parameters from the second module. Experimental results show that the accuracy of the proposed model is 8.2%－12.2% higher than the comparative model，and 9.9%, 5.6% and 7.7% higher on the precision，recall and F value，which proves the validity of the proposed model.

1. Introduction

The power information system software was developed using object-oriented design techniques in the long-term development process. Object-oriented system design is the process of searching the power software architecture and the solutions of power software functional model in essence. The object-oriented power system software gradually evolves from a single function to a comprehensive development for the reason that the comprehensiveness and reliability of the functions. While the more complex of the structure and model of the software itself and the larger scale of the object, which lead to more serious security problems faced by the software of the power information system. The corresponding solutions are proposed including the allocation of resources such as human funds and the control and arrangement of development progress by determining whether the module to be tested and the object have errors which are predicted by the software defect prediction in the early stage of software development. Therefore, software defects and loopholes should be discovered and resolved as soon as possible in the process of software development to provide a necessary protection for national production and normal market operation and offer an important approach for reducing the cost and cycle of testing and improving the quality of the software in the grid [1].

The software defect prediction technology can be divided into static and dynamic defect prediction techniques. The dynamic defect prediction technology focuses on the defect distribution and the variation of quantity over time in the whole life cycle or test phase of the software to predict the future defect distribution of the software. While the static prediction technology which is more commonly used pays more attention to the metrics of the software defect and takes defect prediction based on metrics of different defect-related attributes. It can provide corresponding prediction functions in the software analysis and design stage and in the early stage of development. The existing static defect prediction techniques are basically based on different machine learning algorithms, such as decision tree, random forest, naive Bayes, BP neural network and artificial immune system. The module attributes are obtained according to the software metrics that describes the software characteristics and then classified (defective or not) [2]. These methods have a certain degree of defect prediction ability, while imply some problems more or less. Taking some methods for example. The decision tree ignores the correlation between feature attributes due to overfitting; The naive Bayes has higher requirements for attribute independence which needs to know the prior probability. The neural network is prone to the problem of local optimization or insufficient fitting. It is necessary to obtain the factors related to the defects according to the expert experience like the Bayesian model, while with the low calculation efficiency; The Support Vector Machine has capabilities of good learning and expansion without a unified and efficient method for setting the optimal parameters. Moreover, various algorithms need to use a large number of class and object feature attributes to measure the software in dealing with the object-oriented software which lead to the "dimensionality disaster". And the predictive model is less practical with the long detection time.

The software metrics method, as a standardized method for obtaining the software architecture and module attributes, analyzes the attributes of the software entities and describes the characteristics of the software quantitatively according to the corresponding metrics essentially, which can provide the necessary data sources for the work of the software quality assessment and the metric-based software defect prediction. Traditional structured software metrics include the McCabe structural complexity metrics, LOC statement line metrics and Halstead software science metrics, etc. The complexity software metrics are aimed at the attributes within the software module, while the attributes between modules should also be taken into account because of the significance of the interaction between modules with the development of the object-oriented software technology. The attributes specific to the object-oriented software such as data abstraction, encapsulation, inheritance, polymorphism, information hiding, cohesion and coupling are unable to be extracted by the previous structured metrics in the process of the object-oriented analysis and object-oriented design [3]. The specific metrics models need to be proposed for the object-oriented software because the feature attributes of the software obtained based on the traditional metrics are not sufficient to fully represent the intrinsic characteristics of the object-oriented software. The most representative object-oriented software metrics models are the MOOD model and the CK metric proposed by Chidamber and Kemerer [4]. The CK metric contains six metric attributes that are based on a rigorous metric theory to characterize the scale and complexity of the object-oriented software design. These metric indicators include as follows: Weighted method of class WMC, depth of inheritance tree DIT, number of children in the class NOC, coupling between objects CBO, response for class RFC and lack cohesion of method LCOM. They not only contain most features of the object-oriented software, but also expand their attribute sets correspondingly in each data set to cope with the complexity of the software. The CK metric which is flexible is used for the software defect detection to test the effectiveness of our proposed model in this paper [5].

Defect prediction model for object-oriented software based on particle swarm optimized SVM (Support Vector Machine) is proposed in this paper in order to optimize the performance of the software defect prediction methods. That is, the feature selection is carried out by the relief algorithm firstly, and then the defect prediction of the model is taken to obtain the optimal prediction results, in which the model is combined the strong generalization ability of SVM with the high efficiency of optimization of the PSO (Particle Swarm Optimization) algorithm which updates the inertia weight dynamically with the degree of particle aggregation.

2. Problem description and related algorithms

2.1. Problem description
The defect detection problem of the software in the power information system is researched in this paper. The generation of the software defect is inevitable in the implementation of programming and has a significant impact on the software quality. Therefore, we need to utilize the statistical learning technology to predict the number and type of defects in software systems based on the historical data, existing fault data sets or the software metric data such as defects that have been discovered, and count the number of defects that are not found but may still exist to determine whether the system can be delivered for use [6]. In the process of using the statistical learning technology, it is necessary to reduce the dimensionality of the feature attributes of the classification set and normalize the attribute values to avoid the "dimensionality disaster" and prevent the excessive processing cost firstly. Secondly, the appropriate parameters should be set for the selected classification algorithm. While the parameters set by the expert experience are unable to show the characteristics of the new prediction problem accurately. Therefore, a suitable optimization algorithm is needed to find the corresponding parameters of these classification algorithms to obtain the optimal defect prediction results finally.

2.2. Support Vector Machine
SVM is a general-purpose feedforward neural network that can be used for pattern classification and non-linear regression [7]. The SVM learning algorithm comes from the statistical theory with the core of the empirical risk minimization principle and the basic idea of solving the kernel function and quadratic programming problem. The reason why SVM is chosen in this paper is that it is more suitable to obtain the classification optimal solution from the finite sample. And the introduction of the kernel function enables the SVM model to deal with the non-linear problem of software defect detection effectively meanwhile. The kernel function maps the data to the high-dimensional feature space which makes the sample set linearly separable in it. And the most commonly used kernel function is the Radial Basis Function which preserves the ability to map to the infinite dimensional space while ensuring the positive semi definite of the kernel matrix with the wider scope of the application and the stronger learning ability.

The training sample set is $\boldsymbol{T} = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \dots, (\boldsymbol{x_n}, y_n)\}, y_i \in \{-1, +1\}$ with the number of samples n in it. And the dimension of each sample feature attribute is d, that is $\boldsymbol{x_i} = (x_i^1, x_i^2, \dots, x_i^d)$. The purpose of SVM classification is to find a partition hyperplane $\boldsymbol{w^T x} + b = 0$ that maximizes the interval between the two heterogeneous support vectors in the feature space. Separating samples by class produces the most stable classification results and the strongest processing capacity for the future samples. It is necessary to use the method of Lagrange multipliers to obtain the dual problem for the basic model of the SVM convex quadratic programming to find the partition hyperplane with the largest interval. The basic Lagrangian function is as shown in equation (1):

$$\mathrm{L}(\boldsymbol{w}, b, \boldsymbol{k}) = \sum_{i=1}^{d} k_i \left(1 - y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)\right) + \frac{1}{2}||\boldsymbol{w}||^2 \quad (1)$$

The dual problem as shown in equation (2) can be obtained according to the equation above.

$$\max_{k} P = \sum_{i=1}^{d} k_i - \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} k_i k_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$
$$\text{s.t.} \quad \sum_{i=1}^{d} k_i y_i = 0 \quad (2)$$

$\boldsymbol{x}$ is mapped to the high-dimensional feature space to solve the problem of non-linear partitioning. The original model is converted to $f(\boldsymbol{x}) = \boldsymbol{w}^T \varphi(\boldsymbol{x}) + b$, in which $\varphi(\boldsymbol{x})$ represents the eigenvector after mapping $\boldsymbol{x}$ to the high-dimensional feature space. The dual problem is also converted to the following equation (3).

$$\max_{k} P=\sum_{i=1}^{d} k_i-\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} k_i k_j y_i y_j \varphi(\boldsymbol{x}_i)^T\varphi(\boldsymbol{x}_j)$$

$$\text{s.t.}\quad \sum_{i=1}^{d} k_i y_i=0;\ \ k_i\le 0 \tag{3}$$

The kernel function $\varphi(\boldsymbol{x}_i)^T\varphi(\boldsymbol{x}_j)$ is the inner product of the high-dimensional feature space, which is defined as the result of the function $\varphi(\boldsymbol{x}_i, y_i)$ in the original sample feature space to define the feature space of the map implicitly [8].

When using the above model for the sample classification, there may be a "hard margin" problem which means that each sample is required to meet the constraint requirements but with the high cost and it is difficult to ensure that the classification is not derived from overfitting. So we relax the constraints of the sample and try to get as few unqualified samples as possible while maximizing the interval to alleviate this problem. And then the alternative loss function hinge is added to the optimization goal and a slack variable is introduced to obtain a new optimization goal.

$$\min_{w,b,\zeta}\frac{1}{2}||\boldsymbol{w}||^2+C\sum_{i=1}^{d}\zeta_i$$

$$\text{s.t.}\quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)+\zeta_i\ge 1;\quad \zeta_i\ge 0 \tag{4}$$

In equation (4), the former term indicates the interval of the support vector called “structural risk” which describes the property of the model after regularization; The latter term represents the error called “experience risk” which shows the matching degree of the training data set and the current model; The parameter penalty factor C is used to balance the weights of the two terms to obtain the model that meets people's requirements with the principle of minimizing experience. The Lagrangian function which is an optimized objective function is also calculated by the method of Lagrange multipliers in equation (5).

$$L(\boldsymbol{w},\boldsymbol{\kappa},\boldsymbol{\lambda},\boldsymbol{\zeta},b)=\frac{1}{2}||\boldsymbol{w}||^2+C\sum_{i=1}^{d}\zeta_i+\sum_{i=1}^{d}k_i\left(1-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-\zeta_i\right)-\sum_{i=1}^{d}\lambda_i\zeta_i \tag{5}$$

The dual problem in equation (6) can be obtained according to the equation above.

$$\max_{k} P=\sum_{i=1}^{d} k_i-\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} k_i k_j y_i y_j \varphi(\boldsymbol{x}_i,\boldsymbol{x}_j)$$

$$\text{s.t.}\quad \sum_{i=1}^{d} k_i y_i=0;0\le k_i\le C \tag{6}$$

The support vector expansion $f(x)=\sum_{i=1}^{d} k_i y_i\varphi(\boldsymbol{x}_i,\boldsymbol{x}_j)+b$ can be obtained after calculating the equations above. Thus the classification function is as shown in equation (7).

$$classify(\boldsymbol{x})=\operatorname{sgn}\left(\sum_{i=1}^{d} k_i y_i\varphi(\boldsymbol{x},\boldsymbol{x}_i)+b\right) \tag{7}$$

2.3. Particle Swarm Optimization

An evolutionary technique called the Particle Swarm Optimization (PSO) was put forward based on the behavioural research on the predation of birds proposed by Eberhart and Kennedy [9]. The PSO algorithm initializes a set of stochastic populations firstly which is also called particle swarms. Each particle passes through the solution space at the rate of initialization, in which the velocity is a function of the historical behaviour (speed, position, fitness) of the particle and others in the swarms and changes in each iteration. The position attribute of each particle corresponds to a potential solution. And the pros and cons of the current solution can be evaluated by calculating the fitness of the current particle. If it fails to meet the requirements of the solution, the particle velocity and position will be updated and the next iteration process will be entered until we find the optimal solution or the maximum number of iterations that meets the requirement.

3. Software defect prediction based on particle swarm optimized SVM

3.1. Software defect prediction model

The software defect prediction model proposed in this paper is shown in figure 1. The training samples are normalized and preprocessed numerically and the relief algorithm is used for feature selection at first to solve the problem of low performance and fitness of the large sample data dealt by the PSO-SVM model. Then the parameter penalty factor C of the SVM model and the bandwidth σ of the Gaussian kernel are optimized by the PSO algorithm in order to find out the parameters with the highest accuracy of the classification model. The trained model is used to predict the defects of the test data and future data at last [10].

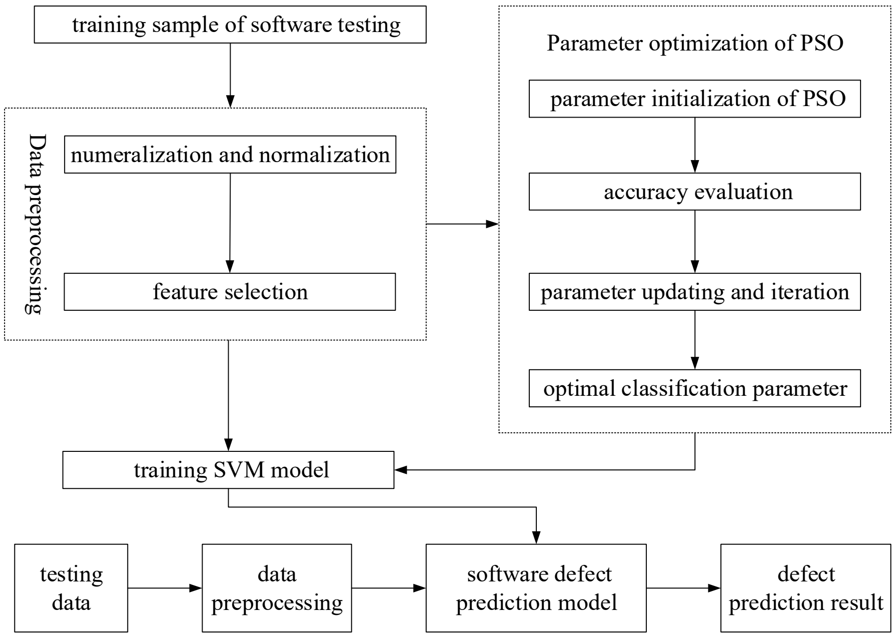

training sample of software testing
Data preprocessing
numeralization and normalization
feature selection
Parameter optimization of PSO
parameter initialization of PSO
accuracy evaluation
parameter updating and iteration
optimal classification parameter
training SVM model
testing data
data preprocessing
software defect prediction model
defect prediction result

Figure 1. Software defect prediction model based on particle swarm optimized SVM.

3.2.Feature attribute pre-processing

Each data set to be tested contains a large number of sample feature attributes in the actual project, which brings an extremely heavy computational burden to the defect prediction model. Moreover, it is impossible to delete or add some attributes arbitrarily due to the different contribution of each attribute in the classification. Therefore we need the feature selection algorithm called relief algorithm to solve this problem. The relief algorithm is designed for the classification problem of the two kinds of data, which can calculate the weight and ranking of each feature attribute for the software defect prediction problem, and ensure the accuracy of the classification model with the high operation efficiency after completing the feature selection. The algorithm is as follows [11].

The sample instance space is $\boldsymbol{T} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \dots, \boldsymbol{T}_n\}$, in which $\boldsymbol{T}_i = (t_1, t_2, \dots, t_m)$.

Step 1: Assigning the value 0 to the weight of each feature attribute of the sample, quantifying the attribute represented by the non-numeric value, and then normalizing all the values according to the method of maximum and minimum normalization.

Step 2: Selecting a sample $\boldsymbol{T}_i$ randomly and choosing the nearest neighbour samples $\boldsymbol{T}_{\text{hit}}$ and $\boldsymbol{T}_{\text{miss}}$ from its samples of the same and different types.

Step 3: Updating the weight of each attribute t_k as follows in equation (8):

$$weight(t_k) = weight(t_k) + \frac{1}{n} \times \frac{D(\boldsymbol{T}_i, \boldsymbol{T}_{miss}, t_k)}{\max(D(t_k))} - \frac{1}{n} \times \frac{D(\boldsymbol{T}_i, \boldsymbol{T}_{hit})}{\max(D(t_k))} \tag{8}$$

$D(\boldsymbol{T}_i, \boldsymbol{T}_j, t_k)$ represents the Euclid distance of $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ over the attribute t_k. $\max(D(t_k))$ means the maximum Euclid distance of all samples on the attribute t_k.

Step 4: Iterating the process for n times from the begin of step 2, and calculating the average weight of each feature attribute at last, according to which the feature attributes are sorted, and then compared with the feature attribute threshold or the final retained feature attribute set is obtained according to the preset quantity for the training of the SVM model.

3.3. Optimization of software defect prediction model parameters

The model of SVM has been discussed in Chapter 2. The kernel function named the Radius Basis Function (RBF) is selected to improve the generalization ability of the software defect prediction model in equation (9), that is:

$$\varphi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-||\boldsymbol{x}_i^2 - \boldsymbol{x}_j^2||/2\sigma^2) \tag{9}$$

Therefore, the correlation parameters that need to be optimized by the particle swarm optimization are the penalty factor C and the bandwidth σ of the Gaussian kernel in the software defect prediction model proposed in this paper. The algorithm is as follows [12].

Step 1: Initializing the speed interval, learning factors c_1 and c_2, inertia weight w, the number of iterations and the particle swarm$\boldsymbol{S} = \{(s_{1_c}, s_{1_\sigma}), (s_{2_c}, s_{2_\sigma}), \dots, (s_{num_c}, s_{num_\sigma})\}$, in which num is the swarm quantity, including the position of each particle (s_{i_c}, s_{i_σ}) and the velocity (v_{i_c}, v_{i_σ}).

Step 2: Calculating the fitness $fitness(\boldsymbol{S}_i^k)$ of this iteration of the particle for the position of each particle. The prediction accuracy of the SVM model obtained by the current penalty factor s_{i_c} and the bandwidth s_{i_σ} of the Gaussian kernel is used as the return value of the fitness function in this paper, as is shown in equation (10), that is:

$$fitness(\boldsymbol{S}_i^k) = accuracy_{SVM}(s_{i\_c}, s_{i\_\sigma}) \tag{10}$$

Step 3: The individual extremum of the particle will be updated by $fitness(\boldsymbol{S}_i^k)$ if $fitness(\boldsymbol{S}_i^k)$ is better than the individual extremum $\boldsymbol{PBest}_i$. And the swarm extremum $\boldsymbol{GBest}^k$ will be updated by $fitness(\boldsymbol{S}_i^k)$ in this iteration when $fitness(\boldsymbol{S}_i^k)$ is superior to the individual extremum of all other particles and the swarm extremum $\boldsymbol{GBest}^{k-1}$ in the previous iteration meanwhile.

Step 4: The iteration will be quit and the swarm extremum $\boldsymbol{GBest}^k$ will be output as the optimal parameter for training the SVM model if the maximum number of iteration is reached or the current swarm extremum $\boldsymbol{GBest}^k$ satisfies the accuracy requirement. Otherwise, according to the following equation (11):

$$\begin{cases} \boldsymbol{V}_i^{k+1} = w \times \boldsymbol{V}_i^k + c_1 \times rand1() \times (\boldsymbol{PBest}_i^k - \boldsymbol{S}_i^k) + c_2 \times rand2() \times (\boldsymbol{GBest}^k - \boldsymbol{S}_i^k) \\ \boldsymbol{S}_i^{k+1} = \boldsymbol{S}_i^k + \boldsymbol{V}_i^{k+1} \end{cases} \tag{11}$$

Updating the speed and position of each particle and starting to the next iteration from step 2. c_1 and c_2 mainly affect the balance between the individual and swarm memory of particles. And we set $c_1 = 1.6$, $c_2 = 1.5$ according to the experience.

The inertia weight w mainly influences the balance between the historical memory of the particle and the current state: When the particle is close to the optimal solution with an excessive value, it ignores the effect of the local search in order to pay attention to the result of the global search without falling into the local optimization which results in the overriding of the optimal solution. Otherwise, the speed of the movement is too slow to approach the optimal solution as quickly as possible [13]. Therefore, a dynamic inertia weight particle swarm optimization algorithm is proposed in this paper, which makes the swarm gradually reduce the speed of movement in the process of concentrating near the optimal solution to make each particle search the fitness of the surrounding space more accurately and enhance the overall performance of the standard PSO algorithm. A variable *close* is defined to indicate the degree of swarm aggregation in this paper based on the analysis above. The swarm aggregation of each iteration *k* is as follows.

$$close_k = \frac{1}{num \times |\boldsymbol{S}_{max} - \boldsymbol{S}_{min}|} \times \sum_{i=1}^{num} D(\boldsymbol{S}_i^k, \overline{\boldsymbol{S}_i^k}) \tag{12}$$

In equation (12), $close \in (0,1)$. $D(\boldsymbol{S}_i^k, \overline{\boldsymbol{S}_i^k})$ represents the Euclid distance of the average position (center of gravity) between each particle and the swarm. $|\boldsymbol{S}_{max} - \boldsymbol{S}_{min}|$ means the maximum diameter length of the solution space. *close* describes the case of approaching the optimal solution space of the particle swarm after each iteration. And the larger the value, the more dispersed the swarm, on the contrary, the more concentrated it is.

It is necessary to reduce the value of the inertia weight w gradually after the particles are aggregated, otherwise, the inertia weight should be increased. They can be mapped to the solution space of the inertia weights by quantifying the degree of aggregation of particles to obtain the values of inertia weights at different concentration levels. Using the definition (13) to calculate the value of w to achieve the above purpose.

$$w_k = \frac{1}{1+\exp(-10(close_k-0.38))} \times (w_{max} - w_{min}) + w_{min} \quad (13)$$

It optimizes the local optimization after quickly closing to the optimal space, and w_{min} and w_{max} are the lower and upper bounds of w which are set to 0.8 and 1.2 by experience.

4. Software prediction model experiment based on PSO-SVM

4.1. Experimental data set

The model proposed in the paper is implemented based on Matlab and compared with LE-SVM and LE-KNN in order to verify the performance of the defect prediction model for object-oriented software. Four experimental data sets that conform to the CK metric in this paper are used to verify the validity of the defect prediction model. The first one is the Class-level data for KC1 provided by National Aeronautics and Space Administration (NASA), which contains 145 samples for a total of 89 feature attributes, 60 defect-free samples and 85 defective samples [14]. The second one is the eclipse2.0 data set based on the real data of open source eclipse [15]. There are 6728 different samples, including 975 defective samples and 5753 defect-free samples. The next one is eclipse3.0 data set containing 9470 samples with 1522 defective samples, and the last one named the ant-1.7 data set which has 745 samples with a total of 166 defect-free samples [16]. Since whether there is a defect is represented by the number of bugs for the eclipse and ant data sets, we need to update them to logical variables 1 and 0 that indicate whether there is a defect firstly. Meanwhile, 700 samples are randomly selected from the last three data sets and divided into two groups with equal numbers as the training sets and test sets respectively because the manifold learning algorithm will have the problem of data point loss in the process of high dimensional dimensionality reduction.

The swarm size of the PSO algorithm is set to 50 and the number of evolutions is set to 100, at the same time, the SVM model is trained by the ten-fold cross-test method to obtain the defect detection accuracy of the SVM on the test set in this paper. The SVM model is implemented on Matlab R2015a based on libsvm-3.21 [17]. The output feature dimension in the LE algorithm is 10. The penalty factor of the SVM model of the LE-SVM algorithm on the KC1 data set is $C = 1$, and its Gaussian kernel bandwidth $\sigma = 0.01176$. While the penalty factor $C = 1$ with the Gaussian kernel bandwidth $\sigma = 0.04176$ on other data sets. The feature dimension retained by relief is 15, the search space of the penalty factor C of the SVM model is (0.1, 100.0) and that of the Gaussian kernel parameter is (0.01, 100.0) in the algorithm proposed in this paper.

Table 1. Cross matrix of actual defect situations and prediction results.

Actual defect situation	Defect prediction result	
	Defective module	Defect-free module
Defective module	correct prediction, defective-$N1$	error prediction, defect-free-$N2$
Defect-free module	error prediction, defective-$N3$	correct prediction, defect-free-$N4$

In table 1,The total number of test samples is $N = N1 + N2 + N3 + N4$, in which the correct number of samples predicted is $N1 + N4$ and the total number of samples predicted incorrectly is $N2 + N3$. The performance of the model defect prediction is analyzed according to the general evaluation index in this paper [18].

The accuracy represents the ratio of the number of samples with the correct prediction to the total number of samples being predicted, in which the correct prediction means that the defective modules were detected as defective accurately, and the non-defective modules were not misjudged. The calculation equation is as follows in equation (14).

$$accuracy = (N1 + N4)/N \tag{14}$$

The precision indicates the ratio of the number of samples that are actually defective and predicted to be defective to the number of all defective samples being predicted, which can be expressed as follows in equation (15).

$$precision = N1/(N1 + N4) \tag{15}$$

The recall shows the ratio of the number of samples that are actually defective and predicted to be defective to all actual defective samples. The calculation equation is as follows in equation (16).

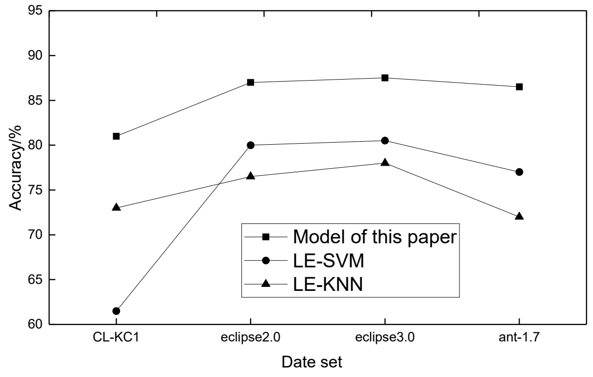
$$recall = N1/(N1 + N2) \tag{16}$$

The value of F is the harmonic mean of the accuracy and recall, and it can be expressed as follows in equation (17).

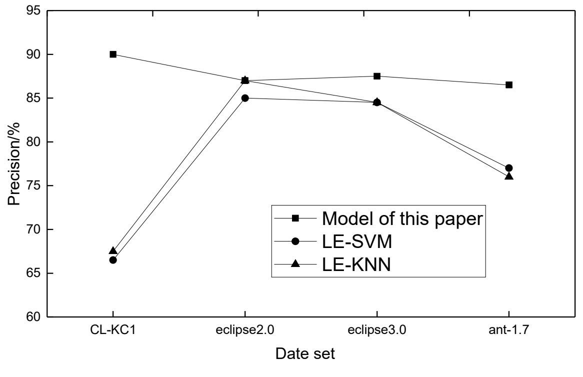
$$F = \frac{2}{1/recall + 1/precision} \tag{17}$$

4.2. Result analysis

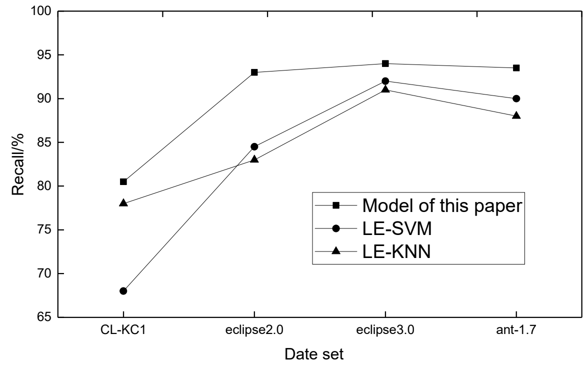
The accuracy can reflect the performance difference between the software defect prediction model proposed in this paper and the comparative model more objectively because the training accuracy of the SVM model is taken as the fitness of the PSO algorithm. Figure 2 shows the accuracy of the proposed model and the comparative model on the four data sets. It can be seen from that the proposed model leads the comparative algorithm on the four data sets which is mainly reflected in the following aspects [19]. The attributes that are unfavourable to the classification such as the attribute with too small numerical differences can be removed by the proposed model through the relief algorithm to make the prediction result more accurate. At the same time, the penalty factor and Gaussian kernel bandwidth that optimize the performance of the SVM training model are obtained through PSO to further improve the accuracy of the prediction results. While the parameters of the comparative algorithm can only be obtained through the experience with the lack of the process of optimization, which leads that the result has a certain distance from the optimal solution and the prediction result has a certain gap with the proposed model. The accuracy of the proposed model on the four data sets is higher than the comparative model 8.2% ~ 12.2%.



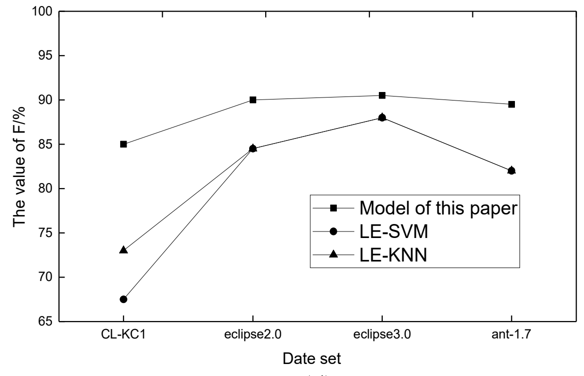
(a)



(b)



(c)



(d)

Figure 2. Performance on each data set.

On the basis of ensuring the overall performance of the model which means making sure that the prediction model has a sufficient good defect prediction accuracy, it is necessary to improve the ability of the defect prediction model to identify the defective modules in the test set, that is, to obtain the higher prediction accuracy and recall. These two indicators respectively illustrate the proportion of the defective results in the predicted defective modules and the proportion of predicted defective results in the defective modules. The former reflects the need to detect overhead on the defect-free modules, and the latter reflects the additional overhead that needs to be re-predicted on the defect-free prediction modules. Therefore, the accuracy and recall should be improved to ensure the lower additional overhead.

The value of F is the harmonic mean of the accuracy and recall which is a balance and comprehensive consideration of the two indicators. The reason for evaluating this indicator is to avoid the problem that the extra cost cannot be reduced led by the case that the single indicator is very high while the other one is poor.

The accuracy, recall and the value of F on the four data sets of the proposed and comparative models are shown in figure 2 (a) ～ (d) respectively. It can be seen that the three indicators of the LE-SVM model and the LE-KNN model are similar in the last three data sets, and the gap of those in CL-KC1 is mainly caused by the recall. This phenomenon mainly occurs because the main information in the original data set is saved in the newly generated low-dimensional data set instead of retaining some of the original feature attributes after the manifold learning for feature dimension reduction. The impact on the prediction results by using the prediction model will be reduced for such data. Similarly, the difference in the accuracy between the two models in figure 2 is relatively small, which can also illustrate the problem [20]. The relief algorithm which is efficient and extremely suitable for the two-class problem is used in the dimension reduction processing for the prediction model proposed in this paper. The problem of the fixed convergence rate and the local optimization are avoided to the utmost by the improved dynamic inertia weight PSO algorithm because of the optimization process of the penalty factor and the Gaussian kernel bandwidth solution space. So that the algorithm has the corresponding optimal model parameters in the face of different test sets to obtain the optimal results of the defect prediction. It can be obtained that the proposed model has a 9.9%, 5.6% and 7.7% lead in accuracy, recall and the value of F respectively over the LE-SVM model with better performance by calculating the average of the indicators of the three models on the four data sets.

5. Conclusion

A corresponding defect prediction model is proposed in this paper for the problem of the defect prediction for object-oriented software in the power information system. The basic idea is that the relief algorithm performs the feature selection to avoid the problem with the excessive feature dimension. And then the optimal SVM training model is obtained by the PSO algorithm which updates the inertia weight dynamically according to the degree of the particle aggregation, so that it is used to predict the defect for the software module of CK metric. The experimental result shows that the model can get a good prediction result with a good performance on the four general indicators. While the model cannot cope with the prediction problem in the software running process as a static analysis model. Moreover, the time performance of the model needs further optimization due to the iterative loop process of the PSO algorithm. The next step is to optimize the model prediction results and the algorithm runtime to cope with future defect prediction.

Acknowledgments
This paper was financially supported by the Science and Technology Project from State Grid Corporation of China: "Research on Distributed Simulation and optimization technology of optical transmission network for Electric power communication (5442XX180003-XX71-18-006)".

References

[1] Gang LI, Yinsheng SU and Chen CHEN 2001 *J. Journal of Computer Applications.* **21**(9): 78-80.
[2] ABAEI G and SELAMAT A 2014 *J. Vietnam Journal of Computer Science.* **1**(2): 79-95.
[3] Yao ZHANG, Zhihai YUAN and Haiyan JIANG 2010 *J. Computer Technology and Development.* **20**(8): 37-40.
[4] Tong YI 2011 *J. Application Research of Computers.* **28**(2): 427-34.
[5] Lukui SHI, Chunjuan MA and Jingxin WANG 2014 *J. Computer Engineering and Design.* **35**(11): 3859-63.
[6] Ramandeep Kaur and Harpreet Kaur 2017 *J.* Volume-3, Issue-1.
[7] CORTES C and VAPNIK V 1995 *J. Machine Learning.* **20**(3): 273-97.
[8] CRISTIANINI N and SCHOLKOPF B 2002 *J. AI Magazine.* **23**(3): 31-41.
[9] KENNEDY J 2011 *Encyclopedia of Machine Learning* (Berlin: Springer) p 760-66.
[10] Zhang H G, Zhang S and Yin Y X. 2017 *Chin J Eng.* **39** (1): 39.
[11] KIRA K and RENDELL L A 1992 *Proceedings of the Ninth International Workshop on Machine Learning* (San Francisco: Morgan Kaufmann Publishers) p 249-56.
[12] Wenli SHANG, Shengshan ZHANG and Ming WAN 2014 *J. Electronic Journal.* **42**(11): 2314-20.
[13] HUANG C L and DUN J F 2008 *J. Applied Soft Computing.* **8**(4): 1381-91.
[14] BOETTICHER G, MENZIES T and OSTRAND T (2007-01-01) [2013-03-17] *DB/OL. http: // promise data. org / repository.*
[15] ZIMMERMANN T, PREMRAJ R and ZELLER A 2007 *Proceedings of the 3rd International Workshop on Predictor Models in Software Engineering* (Washington, DC: IEEE Computer Society) No. 9.
[16] JURECZKO M and MADEYSKI L 2010 *Proceedings of the 6th International Conference on Predictive Models in Software Engineering* (New York: ACM) Article No. 9.
[17] CHANG C C and LIN C J. LIBSVM 2010 *J. ACM Transactions on Intelligent Systems and Technology.* **2**(3): Article No. 27.
[18] Huiyan JIANG, Mao ZONG and Xiangying LIU 2011 *J. Chinese Journal of Computers.* **34**(6): 1148-54.
[19] Yan B Y, Sheng Z F and Li G 2016 *J. Solid Rocket Technol.* **39**(1): 106.
[20] Nanshuai WANG, Jingfeng XUE and Changzhen HU 2015 *J. Chinese scientific papers.* **10**(2):159-163.