Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Three-way decisions based software defect prediction

Weiwei Li[a,b,*], Zhiqiu Huang[a], Qing Li[c]

[a] College of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[b] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[c] College of Command Information System, PLA University of Science and Technology, Nanjing 210007, China

ARTICLE INFO

Article history:
Received 28 December 2014
Revised 24 September 2015
Accepted 29 September 2015
Available online 19 October 2015

Keywords:
Software defect classification
Software defect ranking
Three-way decisions

ABSTRACT

Based on a two-stage classification method and a two-stage ranking method on three-way decisions, this paper introduces a three-way decisions framework for cost-sensitive software defect prediction. For the classification problem in software defect prediction, traditional two-way decisions methods usually generate a higher classification error and more decision cost. Here, a two-stage classification method that integrates three-way decisions and ensemble learning to predict software defect is proposed. Experimental results on NASA data sets show that our method can obtain a higher accuracy and a lower decision cost. For the ranking problem in software defect prediction, a two-stage ranking method is introduced. In the first stage, all software modules are classified into three different regions based on three-way decisions. A dominance relation rough set based ranking algorithm is next applied to rank the modules in each region. Comparison experiments with 6 other ranking methods present that our proposed method can obtain a better result on FPA measure.

1. Introduction

Software defect prediction attempts to predict the defect proneness of new software modules with the historical defect data. It plays an important role in improving the quality of software systems [2,16,22]. The prediction technology is supposed to estimate the number of defects and to locate the position of each defect. Unfortunately, it is still hard to obtain the exact number or position of all defects based on current methods. Instead, since more information can be extracted from the historical defect data, two types of machine learning tasks are usually applied in software defect prediction: classification and ranking [65]. In a classification problem, new incoming software modules will be classified into defect-prone and not-defect-prone categories, with no differentiation between how bad the defect ones will be. In a ranking problem, new incoming modules will be sorted in descending order according to their numbers of predicted defects. The purpose of ranking is to identify the modules most likely to contain the largest numbers of defects and to allocate more testing resources for them.

In existing work for software defect classification, most researchers adopted many popular classifications methods directly, such as Naive Bayes [53], SVM [18], decision tree [39], and neural networks [31]. Furthermore, by considering the misclassification cost, some studies regarded the problem as a cost-sensitive learning task [50], and employed cost-sensitive learning methods including boosted neural network [80] and cost-boosting [34]. With or without cost-sensitive learning, all these studies assumed the underlying classification is binary and employed two-way decisions methods. A two-way decisions method is a kind of "immediately" decision method, which means it can provide the classification result directly and quickly. However, a quick result is usually accompanied by a high classification error. For example, assume the conditional probability of one module being a defect is 0.51, which is computed by the classification model, then its conditional probability of being a not-defect is 0.49. According to the majority principle in simple two-way decisions methods, this module will be classified into the defect-prone category directly. For these "vague" modules with conditional probabilities around 0.5, it has a high probability to classify them into wrong categories followed by more misclassification cost.

In existing work for software defect ranking, most researchers combined the defect-prone modules with the not-defect-prone modules together and ranked them based on the same attributes. A common method is applying regression models [8,9,13,24,33]. Regression methods implicitly assume that there exists a quantitative relation between the class label and the part of (or the whole of) attributes. However, in most applications, there exists a qualitative relation between the class label and the attributes only, and the quantitative hypothesis is too restrict to be satisfied.

* Corresponding author at: College of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China. Tel.: +8613584061768.
E-mail addresses: liweiwei@nuaa.edu.cn, jiaxiuyi@gmail.com (W. Li), zqhuang@nuaa.edu.cn (Z. Huang), lqhyt1994@126.com (Q. Li).

http://dx.doi.org/10.1016/j.knosys.2015.09.035
0950-7051/© 2015 Elsevier B.V. All rights reserved.

In this paper we aim to solve the above problems. The three-way decisions theory is applied into the software defect prediction problem. The theory of three-way decisions is an extension of the common two-way decisions [69]. The essential ideal of three-way decisions is described in terms of a ternary classification according to evaluations of a set of criteria [71]. In widely used two-way decisions methods, an object will be accepted to classify into a particular category when it satisfies the criteria, and it will be rejected to classify into the particular category when it does not satisfy the criteria. By considering the uncertainty in many situations, three-way decisions methods classify objects based on the set of criteria with some degree, and use thresholds on the degrees of satisfiability to make one of three decisions: (a) accept an object as satisfying the set of criteria if its degree of satisfiability is above a certain level; (b) reject the object by treating it as not satisfying the criteria if its degree of satisfiability is below another level; and (c) neither accept nor reject the object but opt for a deferment. The primary advantage of applying three-way decisions over two-way decisions is that three-way decisions can choose some potentially misclassified objects for further-exam, which may lead to a lower classification error and less misclassification cost.

For software defect classification task, a two-stage classification method based on three-way decisions is proposed in this paper. In the first stage, a base classifier will be trained to compute the conditional probability of each module. After comparing to the thresholds, the modules with acceptance decision are classified into the defect-prone category, the modules with rejection decision are classified into the not-defect-prone category, and the left modules with deferment decisions wait for further-exam. In the second stage, for deferment modules, several classifiers will be combined as ensembles to provide a two-way decisions result. The experimental results on NASA data sets show that our proposed method can produce a higher classification accuracy and less misclassification cost.

For software defect ranking task, a two-stage ranking method is also proposed in this paper. In the first stage, all modules will be classified into three regions based on three-way decisions. In the following stage, a dominance relation rough set based algorithm will be introduced to rank the modules in each region, respectively. Several comparing experimental results validate the efficiency of our proposed method.

The rest of this paper is organized as follows. Some related work of software defect classification, software defect ranking, and three-way decisions are reviewed in Section 2. Section 3 presents the two-stage software defect classification method based on three-way decisions. Section 4 gives the two-stage software defect ranking method. All experiments are discussed in Section 5. Finally, we draw the conclusion in Section 6.

2. Related work

2.1. Software defect classification

An import work in software defect classification task is to construct defect-related feature space. For the defect-related features, many approaches have been proposed based on diverse information [16], such as code metrics [56], process metrics [23], previous defects [38] and so on [1,60].

Several individual studies reported that LOC data performs well in software defect prediction [77,83]. D'Ambros et al. [14] reported that change coupling correlates with defects. Nagappan and Ball [55] applied code churn together with dependency metrics to predict defect-prone modules. Khoshgoftaar and Seliya [35] considered 14 process metrics in their paper. Weyuker et al. [64] constructed a defect-count prediction model using a number of process measures in addition to structural measures. Besides existing studies which considered different features, some researches also focused on the feature selection on sets of metrics seems to improve the performance [7,32,50,62].

Another important work in software defect classification task is the construction or selection of the classification models or methods. Many statistical models and machine learning methods were applied in existing studies. Olague et al. [58] applied logistic regression to predict defects. Mizuno and Kikuno [54] reported that Orthogonal Sparse Bigrams Markov models were best suited to defect prediction based on their study. Bibi et al. [6] applied Regression via Classification method in software defect prediction. Khoshgoftaar et al. [37] built a classification tree using TREEDISC algorithm to predict whether a module was likely to have defects discovered by customers, or not, based on software product, process, and execution metrics. Classical machine learning algorithms were also widely used for software defect prediction such as Naive Bayes [53], Random Forests [52], and C4.5 [39].

Some comparison works among different methods were also studied by many researchers. Briand et al. [10] compared traditional regression techniques with multivariate adaptive regression splines. Khoshgoftaar and Seliya [36] compared 7 models in their work, and these models were evaluated against each other by comparing a measure of expected misclassification cost. Vandecruys et al. [63] compared ant colony optimization against C4.5, SVM, logistic regression, K-nearest neighbour, RIPPER and majority vote. Kanmani et al. [31] compared two variants of artificial neural networks against logistic regression and discriminant analysis. Guo et al. [20] compared 27 modeling techniques through the WEKA tool. Arisholm et al. [2] reported that their comprehensive performance comparison revealed no predictive differences between the 8 modeling techniques they investigated. Catal [12] reported that Random Forests provided the best prediction performance for large data sets and Naive Bayes was the best prediction algorithm for small data sets in terms of AUC evaluation parameter.

More literature reviews of existing studies on software defect classification techniques can be found in [2,11,12,22,65].

2.2. Software defect ranking

A scenario that is more useful in practice is to rank the classes by the predicted number of defects they will exhibit [16]. Given a predicted rank-order, one can focus on the top of the list due to the limitation of testing resources.

Many researches on ranking methods have been done. Based on their study at Ericsson, Ohlsson and Alberg [57] suggested that it was possible to predict the most defect-prone modules before coding has started. Khoshgoftaar and Allen [33] introduced a module-order model for defect prediction, and found that module-order models gave management more flexible reliability enhancement strategies than classification models. Ostrand et al. [5,59] predicted the expected number of defects by a negative binomial regression model, and then produced a predicted ranking of the units from most faulty to least faulty. Based on their ranking result, they could focus on the top *N*% of the files only [60]. Yang et al. [67] proposed a learning-to-rank algorithm for constructing defect prediction models.

Some researchers also considered the problem of how to measure the ranking performance. D'Ambros et al. [15] applied Spearman's correlation coefficient between the list of classes ranked by number of predicted defects and number of actual defects to measure the performance. In their further study [16], cumulative lift charts was employed as the evaluation measure. Weyuker et al. [65] defined a fault-percentile-average metric. Several comparison studies between different ranking methods on different performance measures were also reviewed in [65,66].