A New Software Fault Prediction Model in Imbalanced Data
Shihai Wang\textsuperscript{1,a,*}, He Ping\textsuperscript{2,b}

\textsuperscript{1}The School of Reliability and System Engineering, Beihang University, Beijing, China
\textsuperscript{2}The School of Reliability and System Engineering, Beihang University, Beijing, China
\textsuperscript{a}wangshihai@buaa.edu.cn, \textsuperscript{b}zy1314113@buaa.dse.edu.cn
*Corresponding author

**Keywords:** Formatting, SMOTE, Fault Prediction, Weights, Boosting.

**Abstract.** Early fault-prone prediction is a critical technique for achieving high reliability software. The prediction results help the software developers to pay more attentions on these high-risk modules. For software fault prediction modeling, machine learning techniques have been widely employed. Model selection problem is always a challenge for generating an efficient predictor. In this paper, we come up with a new model of software fault prediction. We distinguish the importance and credibility of the 3 kind of samples: original fault-prone samples, original not-fault-prone samples and artificial samples by SMOTE. In our study, we consider the 3 kind of samples have obvious differences in significance and credibility. We verified our method through the experiments on the MDP datasets.

**Introduction**

In the last few decades, the importance of software quality has gained a wide attention. It would threaten the safety of people or make a risk on the business for software with poor quality [1]. Many experts have pointed out that an early efficient fault prediction can make a significant contribution on software quality especially on the condition that the recourses are limited [2]. The fault prediction models can make a prediction on the number of faults and the fault-prone modules. So the developers can focus on high-risk components during the testing stages [3]. An efficient way to predict fault is to build the relationships between software metrics and fault proneness. In other areas, many similar problems have been solved by the method of machine learning. Introducing machine learning modeling methods into software fault prediction has become a popular research topic. There are many software fault prediction models based on machine learning method, such as Artificial Neural Network [4], Support Vector Machines [5, 6] and many others. But software defect prediction is an imbalanced data sets problem which is difficult to learn a satisfactory classifier. The imbalanced data distribution is characterized as one or some classes have more instances than others. When one classes have more than 90% samples in given data sets, a trivial classifier that labels everything with the majority class can achieve high accuracy. It is apparent that for domains with imbalanced distributions, classification accuracy for all samples is not a sufficient performance measure. ROC analysis [7] and metrics such as precision, recall and F-value [8,9] have been used to evaluate the performance of the learning algorithm on the minority class. The prevalence of class imbalance in various scenarios has caused a surge in research dealing with the minority classes. Several approaches for dealing with imbalanced data sets were recently introduced [10,11]. A confusion matrix in Table 1 is used to evaluate performance of a machine learning algorithm for imbalanced data sets problems. In classification problems, assuming class “C” as the minority...
class of the interest, and “NC” as a conjunction of all the other classes, there are four possible outcomes when detecting class “C”.

In machine learning area, it is well known that combination of classifiers is an effective technique for improving prediction accuracy. As one of the most popular combining techniques, boosting [12] uses adaptive sampling of instances to generate a highly accurate ensemble of classifiers whose individual global accuracy is only moderate. The fundamental issue with imbalanced data classification is the classification of imbalanced data has posed a significant drawback of the performance of most standard learning algorithms, which assume or expect balanced class distribution or equal misclassification costs. Boosting is a kind of method generally for effectively improving the accuracy of a given learning algorithm [13]. To solve imbalance data sets problem, experts have come up with some specific algorithms such as an alternative of AdaCost [14], AsymBoost [15] (a cost-sensitive extension of Real AdaBoost), AdaC2 [16], cost-sensitive IDBoosting algorithms proposed in [17], robust asymmetric Adaboost [18] et al.

On the other hand, many experts pay more attention to the oversampling techniques for minority classes. One famous oversampling method is SMOTE (Synthetic Minority Oversampling Technique) [19], a technique for counter the effect of having few instances of the minority class in a dataset. But most software fault prediction molds with this method does not distinguish the importance and credibility of the 3 kind of samples: original fault-prone samples (program modules), original not-fault-prone samples and artificial samples by SMOTE. In our study, we consider the 3 kind of samples have obvious differences in significance and credibility. To be specific, the original fault-prone samples are more important than others and the artificial samples are less credible than others. Meanwhile, in our study, we find that the original not-fault-prone which are very close to the fault-prone ones are doubtable.

In this paper, build on the exiting SMOTE and AdaBoost algorithms, we come up with a new software fault prediction model by giving all samples suitable weights respectively. The main goal of this study is to come up with a new model of software fault prediction. We distinguish the importance and credibility of the 3 kind of samples: original fault-prone samples, original not-fault-prone samples and artificial samples by SMOTE.

This paper is organized as follows: In the next section, we presented our methodology. Section 3 shows the experiments and results. Finally, in section 4, we summarized and presented our conclusions.

Algorithm

AdaBoost performs a forward gradient descent procedure to seek F for minimizing J(F) in the k-th stage of AdaBoost, AdaBoost attempts to minimize J(F_k-1 + a_k f_k) for seeking a_k and f_k where F_{k-1} = \sum_{t=1}^{k-1} a_t f_t(x_t). J(F) was named as cost function. The cost function of AdaBoost is (4). Obviously, the weights of all samples are the same (1/n).

\[ J(F_{k-1} + a_k f_k) = \sum_{i=1}^{n} \frac{1}{n} e^{-y_i \sum_{t=1}^{k} a_t f_t(x_i)}. \] (1)

In our study, we change the cost function as follows:
\[ J(F_{k-1} + a_k f_k) = \sum_{i=1}^{n} w_i e^{-y_i \sum_{t=1}^{k} a_t f_t(x_i)} . \]  \hspace{1cm} (2)

Where

\[ w_i = \begin{cases} 
1 & \text{sample } i \text{ is original fault-prone} \\
\frac{1}{c_0} \sum_{c=1}^{c_0} \frac{\min_{1 \leq c \leq c_0} d_{ic})y_c}{d_c} & \text{other conditions} 
\end{cases} \]  \hspace{1cm} (3)

\[ y_c = \begin{cases} 
1 & \text{sample } c \text{ is original fault-prone} \\
-1 & \text{sample } c \text{ is original not fault-prone} 
\end{cases} \]  \hspace{1cm} (4)

The variable \( d_{ic} \) is the distance between the sample \( i \) and its \( c \)-th nearest original sample and the variable \( y_c \) is the label of this sample.

The algorithm can be summarized as follows:

**Input:** \((x_1, y_1), \ldots, (x_n, y_n)\) ; where \( x_i \in X \), \( y_i \in \{0, 1\}, \) \( i = 1, 2, \ldots, n \) ;

**Initialization:** Set weights \( w_i^{(0)} = 1 \) on \( X \).

Find the original not-fault-prone samples and the artificial samples \((\hat{x}_1, \hat{y}_1), \ldots, (\hat{x}_m, \hat{y}_m)\) :

**Repeat** for \( j = 1, 2, \ldots, m \):

(a) Find the nearest \( c_0 \) original samples of \( \hat{x}_j \) in \( X \).

(b) Set \( w_j^{(0)} = \frac{1}{c_0} \sum_{c=1}^{c_0} \frac{\min_{1 \leq c \leq c_0} d_{ic})y_c}{d_c} \).

**Normalize** \( w_i^{(0)} = w_i^{(0)}/\sum_{i=1}^{n} w_i^{(0)} \).

**Repeat** for \( t = 1, 2, \ldots, T \):

(a) Train weak learner \( f_t \) using weights \( w_i^{(t-1)} \) on the training data, where

\[ f_t = \arg \max_{f \in H} \sum_{i=1}^{n} w_i^{(t-1)} y_i h(x_i) . \]  \hspace{1cm} (6)

(b) Compute the (nonnegative) weight \( \alpha_t \) of \( f_t \) :

\[ \alpha_t = \frac{1}{2} \ln \frac{1-err}{err} . \]  \hspace{1cm} (7)

where

\[ err = \frac{\sum_{y_i \neq h(x_i)} w_i^{(t-1)}}{\sum_{i=1}^{n} w_i^{(t-1)}} . \]  \hspace{1cm} (8)

(c) Reweight: update weights of training data:

\[ w_i^{(t)} = w_i^{(t-1)} e^{-y_i \alpha t f_t(x_i)} . \]  \hspace{1cm} (9)

**Output:** output the final classifier:
\[ F_T = \text{sign} \left( \sum_{t=1}^{T} \alpha_t f_t(x) \right). \] 

(10)

**Experiments**

Our experiments are carried out on 12 data sets from the data metrics program NASA’s MDP (Metrics Data Program) data repository.

To compare the power of different prediction models fairly, every dataset is separated to 2 parts: training set and testing set. For each dataset, every learning model need to learn a classifier in the training set and verify the accuracy in the testing set. There are 3 learning models being tested for the capacity of software fault prediction.

The first one is AdaBoost. AdaBoost is a modular ensemble framework that can embody any weak learning algorithms. It has sound theoretical performance guarantees and strong experimental results. If the AdaBoost framework can be improved to adapt the imbalanced data classification, then any conventional weak learning algorithm can be integrated without many modifications. Compared with other ensemble learning methods such as bagging, AdaBoost is more successful in variance reduction. With stumps (decision tree with single splitting node) as weak classifiers, AdaBoost performs well and can realize feature selection at the same time, which guarantees that AdaBoost can be used in high dimensional data classification such as images.

The second one is SMOTE & AdaBoost. SMOTE creates synthetic instances of the minority class by operating in the “feature space”. In the beginning, SMOTE synthetically generates more instances of the minority class. Then, the AdaBoost trains the classifier in the new ‘balanced datasets’. In the end, use this classifier to verify the accuracy in the testing set.

The third one is the model proposed in this paper. We call it SMOTE & Weight-Boosting. It is an improvement of the second one. At the beginning, SMOTE creates synthetic instances. Then, use our method to compute the weight of every sample in the training set. Finally, use boosting algorithm to train a classifier and verify the accuracy in the testing set.

The results can be seen in the Table 1.

<table>
<thead>
<tr>
<th>Data set</th>
<th>The F-VALUE</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AdaBoost</td>
<td>SMOTE &amp; AdaBoost</td>
<td>SMOTE &amp; Weight-Boosting</td>
</tr>
<tr>
<td>MC2</td>
<td>0.13442</td>
<td>0.55104</td>
<td>0.60605</td>
</tr>
<tr>
<td>KC3</td>
<td>0.07832</td>
<td>0.30773</td>
<td>0.43453</td>
</tr>
<tr>
<td>MW1</td>
<td>0.09526</td>
<td>0.31734</td>
<td>0.32214</td>
</tr>
<tr>
<td>CM1</td>
<td>0.06783</td>
<td>0.38652</td>
<td>0.38861</td>
</tr>
<tr>
<td>PC1</td>
<td>0.11374</td>
<td>0.37671</td>
<td>0.39834</td>
</tr>
<tr>
<td>PC3</td>
<td>0.05734</td>
<td>0.40786</td>
<td>0.3966</td>
</tr>
<tr>
<td>PC4</td>
<td>0.08765</td>
<td>0.59939</td>
<td>0.59309</td>
</tr>
<tr>
<td>PC2</td>
<td>0.09874</td>
<td>0.18004</td>
<td>0.18323</td>
</tr>
<tr>
<td>KC1</td>
<td>0.05372</td>
<td>0.47018</td>
<td>0.47572</td>
</tr>
<tr>
<td>MC1</td>
<td>0.06873</td>
<td>0.51539</td>
<td>0.5211</td>
</tr>
<tr>
<td>JM1</td>
<td>0.02455</td>
<td>0.13866</td>
<td>0.1129</td>
</tr>
<tr>
<td>PC5</td>
<td>0.06424</td>
<td>0.42789</td>
<td>0.43248</td>
</tr>
</tbody>
</table>
In the table, we can find that the first model get a high score in the overall accuracy but a low score in the minority accuracy; The second model can get well-balanced scores in both overall accuracy and minority accuracy; The third model can get well-balanced scores too and do better than the second one usually.

Summary
In this paper, we come up with a new model of software fault prediction. We distinguish the importance and credibility of the 3 kind of samples: original fault-prone samples, original not-fault-prone samples and artificial samples by SMOTE. In our study, we consider the 3 kind of samples have obvious differences in significance and credibility. To be specific, the original fault-prone samples are more important than others and the artificial samples are less credible than others. Meanwhile, in our study, we find that the original not-fault-prone which are very close to the fault-prone ones are doubtable.

We verified our method through the experiments on the MDP datasets. We can summarize that our model can enhance the consequence in most cases compare with the traditional method.

Acknowledgement
Thanks to my teacher. He help me to find the heart of the matter. Thanks to my schoolmates. They help us to solve problems. Thanks to everyone who support and encourage us.

References

Reference to a book:


