International Journal of Computer and Communication Technology

Volume 7 | Issue 4 Article 2

October 2016

MEASURING CODE QUALITY USING SUPPORT VECTOR MACHINE CLASSIFIER FOR IMPROVED CODING TECHNIQUES

A.K. IAVARASI
Department of Computer Science and Engineering, Sona College of Technology, Salem, TN, India,
Arthy.feb26@gmail.com

S. AARTHY
Department of Computer Science and Engineering, Sona College of Technology, Salem, TN, India,
Ilavarasi.shakthi@gmail.com

MEASURING CODE QUALITY USING SUPPORT VECTOR MACHINE CLASSIFIER FOR IMPROVED CODING TECHNIQUES

A.K.IAVARASI[1] & S.AARTHY[2]

[1,2]Department of Computer Science and Engineering, Sona College of Technology, Salem, TN, India
Email: Arthy.feb26@gmail.com, Ilavarasi.shakthi@gmailcom

Abstract- classification is the problem of identifying a set of categories to a new comments.To improve the efficiency of the code,quality metrics are applied for evaluation.The binary classifier,predicts the false positive rates with lesser accuracy,and limited number of classes only to predict the accuracy for classifier.To address this problem,support vector machine classifier is used,which helps in detecting the false positive rates,improving code quality and the accuracy will also increased.

Indexterms:*Binary classifier,support vector machine, kernal machine,mchine learning.*

I: INTRODUCTION

Formal program specification are difficult for humans to construct the correct specification mining.The false positive rate is a candidate specifications that does not describe a correct behaviour specification.In large software specifications are difficult to debug errors in the source code.These specification are typically use two state temporal properties and they are limited to expressive powers.Temporal property produce a large set of candidate specifications[1].In learning model, a given set of features to be associated with a potential pair(a,b).Recall and Precision are the two models for binary classifier in learning model.Recall measuers the probability of the given specification.precision are the probability to a returned candidate specification.The linear classifier applications is a traning of extremely sufficient[1].

In this paper develops a learning model called support vector machine.support vector machine is a set of learning machines used for classification and regression.The classification and regression tool that uses for machine learning to maximize predictive accuracy and automatically avoiding the data.Support vector machine defined as a systems of a linear function in a high dimensional space.

Support vector machine is used for many applications such as hand writing analysis,especially used for pattern classification and regression based applications.In statistical learning problem for support vector machine,the given traning set of data $\{(\mathbf{x}_1,y_1)... (\mathbf{x}_l,y_l)\}$ in $R^n \times R$ s according to unknown probability $P(\mathbf{x},y)$, and a loss function $V(y,f(\mathbf{x}))$ that measures the error, for a given $\mathbf{x}$, $f(\mathbf{x})$ is "predicted" instead of the actual value y.

II:RELATED WORK

1.Privately Finding Specifications:

In this paper, the notion of privacy to the domain of specification mining The definition of privacy that is motivated by the context of finding specifications. Intuitively, the definition considers the prior knowledge that an attacker has and the following knowledge that an attacker gains upon seeing the published information. Privacy is preserved if the subsequent knowledge is about the same as the prior knowledge.To introduce blurry traces, which convert information in a trace into essentially random noise. To show that, if a participant publishes a blurry trace, then privacy is preserved[12].

An algorithm for learning program specifications that preserves the privacy of the participant's traces. This allows equally distrusting participants to collect all benefits of sharing basic trace data without allowing observers to the unfavorable conclusions about the quality of their code.

2.CROSS VALIDATION AND BOOSTRAP FOR ACCURACY ETIMATION AND MODEL SELECTION:

In this paper,compare the two estimation methods are,cross validation and bootstrap. Estimating the accuracy of a classifier induced by supervised learning algorithm is important not only to predict its features prediction accuracy ,but also for choosing a classifier[3].For estimating the final accuracy of a classifier,we would like an estimation method with low bias and low variance.Some of the assumptions made by the different estimation methods and present concrete examples for each method fails.
The bias of a method to estimate a parameter θ is defined as the expected value minus the estimated value.The unbiased method is a method that has zero bias.When a given method may have bias may be poor due to low variance[4].The results is a better scheme for both bias and variance,when compared to cross validation.In bootstrap has low variance,but extremely large on some problems.

3.MINING TEMPORAL SPECIFICATIONS FOR ERROR DETECTION:

Specifications are necessary in order to find software bugs using program verification tools.In this paper, presents a novel automatic specification mining algorithm that uses information about error handling to learn temporal safety rules[2].

It is based on the observation that programs often make mistakes along exceptional control-flow paths, even when they behave correctly on normal execution paths. This focus improves the effectiveness of the miner for discovering specifications beneficial for bug finding[1]. Finally, we give a quantitative comparison of our technique's bug-finding powers to generic "library" policies. For our domain of interest, mining finds 250 more bugs. We also show the relative unimportance of ranking candidate policies. In all, we find 69 specifications that lead to the discovery over 430 bugs in 1 million lines of code.

```
Session sess = sfac.openSession();
Transaction tx;
try
     {
       tx = sess.beginTransaction();
    // do some work
      tx.commit();
    }
  catch (Exception e)
 {
    if (tx != null)
   tx.rollback();
   throw e;
}
finally
sess.close();
```

Fig 1.hibernate2 Session **class documentation pseudocode** and temporal **safety policy**, given as an FSM over a six-event alphabet. Edge labels (events) are either successful method invocations or method errors. Other transitions involving these six events violate the policy, but other events are not constrained.

Given a set of traces, we consider all event pairs (a, b) from those traces such that all of the following occur:[2]

Exceptional Control Flow-Our novel filtering criterion is that event b must occur at least once in some cleanup code .we require Eab > 0. We assume that if the policy is important to the programmer,language-level error handling will be used at least once to enforce it.

One Error-There must at least one error trace with a but without b: we require Ea > 0.

Same Package- Events a and b must be declared in the same package.

Dataflow- Every value and receiver object expression in b must also be in a. When dealing with static traces we require that every non-primitive type in b also occur in a.

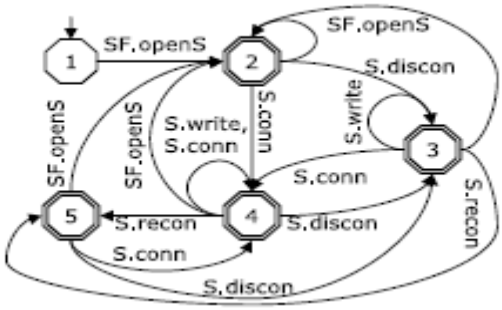

Fig. 2. A slice of the Session policy learned by Strauss: the full learned specification has 10 states and 45 transitions.

Strauss's use of frequency information means that common sequences of events like find and delete are included as part of the policy[7]. Paths through states 2–6 are all particular instantiations of the "do some work" state.

4. SPECIFICATION MINING WITH FEW FALSE POSITIVES:

In this paper, a novel technique that automatically infers limited correctness specification with a very low false positive rates.[13] The existing specification miners false positives because they assign equal weights to all program behavior.

To evaluate our technique in two ways: as a preprocessing step for an existing specification miner and as part of a novel specification inference algorithm. Our technique identifies which traces are most indicative of program behavior, which allows off-the-shelf mining techniques to learn the same number of specifications using 60% of their original input. This results in many fewer false positives as compared to state of the art techniques, while still finding useful specifications on over 800,000 lines of code. When minimizing false alarms, we obtain a 5% false positive rate, an order of magnitude improvement over previous work[8]. When combined with bug finding software, our mined specifications locate over 250 policy violations.

```
void bad ( Socket s, Conn c) {
string message = s. read ();
string query = " select * " +
" from emp where name = " +
message ;
c. submit ( query );
s. write (" result = " +
c. result ());
}

void good ( Socket s, Conn c) {
string message = s. read ();
c. prepare (" select * from "
+ " emp where name = ?",
message );
c. exec ();
s. write (" result = " +
c. result ());
}
```

Figure 2.1: Pseudocode for an example internet service. The bad method passes untrusted data to the database; good works correctly. Important events are italicized.

III: PROPOSED WORK:

Support vector machines (SVM) are a set of related supervised learning methods used for classification and regression.In another terms, Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. Support Vector machines can be defined as systems which use hypothesis space of a linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory.

In support vector machine(SVM) is mainly used to detect the attacks.To reduce the execution time speed.Use more number of classes for measuring the code.The SVM classifier that is combined with the k nearest neighbor of the source code.A decision hyperplane can be defined by an intercept term and a decision hyperplane normal vector which is perpendicular to the hyperplane. This vector is commonly referred to in the machine learning literature as the weight vector . To choose among all the hyperplanes that are perpendicular to the normal vector, we specify the intercept term . Because the hyperplane is perpendicular to the normal vector, all points on the hyperplane satisfy .

CONCLUSION:

In binary classifier only two classes are involved.So the space complexity in specification will be increased and also execution time is high .So the specification results will not be efficient. The support vector machine is one of the factor of a standard machine tool their clear efficiency in dealing with overfitting.The time complexity of the source code is reduced and the space complexity of the code will be used in large scale software systems.

REFERENCE:

[1]. Claire Le Goues and Westley Weimer, ”Measuring Code Quality to Improve Specification Mining,” IEEE Trans. Software Engg…, vol. 38, no. 1,Jan/Feb. 2012.

[2]. W. Weimer and G.C. Necula, “Mining Temporal Specifications for Error Detection,” Proc. Int’l Conf. Tools and Algorithms for the Construction and Analysis of Systems, pp. 461-476, 2005.

[3]. R. Kohavi, “A Study of Cross-Validation and Bootstrap forAccuracy Estimation and Model Selection,” Proc. Int’l Joint Conf Artificial Intelligence, pp. 1137-1145, 1995.

[4]. G. Ammons, R. Bodik, and J.R. Larus, “Mining Specifications,”Proc. ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages, pp. 4-16, 2002.

[5]. V.R. Basili, L.C. Briand, and W.L. Melo, “A Validation of Object- Oriented Design Metrics as Quality Indicators,” IEEE Trans Software Eng., vol. 22, no. 10, pp. 751-761, Oct. 1996.

[6]. R.P.L. Buse and W. Weimer, “A Metric for Software Readability,” Proc. Int’l Symp. Software Testing and Analysis, pp. 121-130, 2008.

[7]. M. Gabel and Z. Su, “Symbolic Mining of Temporal Specifications,”Proc. Int’l Conf. Software Eng., pp. 51-60, 2008.

[8]. R.P.L. Buse and W. Weimer, “A Metric for Software Readability,”Proc. Int’l Symp. Software Testing and Analysis, pp. 121-130, 2008.

[9]. R. Subramanyam and M.S. Krishnan, “Empirical Analysis of CKMetrics for Object-Oriented Design Complexity: Implications for Software Defects,” IEEE Trans. Software Eng., vol. 29, no. 4,pp. 297-310, Apr. 2003.

[10]. M. Vokac, “Defect Frequency and Design Patterns: An Empirical Study of Industrial Code,” IEEE Trans. Software Eng., vol. 30, no. 12, pp. 904-917, Dec. 2004.

[11]. J. Whaley, M.C. Martin, and M.S. Lam, “Automatic Extraction of Object-Oriented Component Interfaces,” Proc. ACM SIGSOFT Int’l Symp. Software Testing and Analysis, 2002.

[12]. W. Weimer and N. Mishra, “Privately Finding Specifications,”IEEE Trans. Software Eng., vol. 34, no. 1, pp. 21-32, Jan./Feb. 2008.

[13]. M. Gabel and Z. Su, “Symbolic Mining of Temporal Specifications,”Proc. Int’l Conf. Software Eng., pp. 51-60, 2008.

[14]. M. Shepperd, “A Critique of Cyclomatic Complexity as a Software Metric,” Software Eng. J., vol. 3, no. 2, pp. 30-36, 1988.

[15]. J. Yang, D. Evans, D. Bhardwaj, T. Bhat, and M. Das, “Perracotta:Mining Temporal API Rules from Imperfect Traces,” Proc. Int’lConf. Software Eng., pp. 282-291, 2006.

◈◈◈