Tuning Learners before Predicitting: New Reflection in Defect Prediction

A, B, C

ABSTRACT

Data mining techniques have been widely applied to software defect prediction with various empirical data sets. Those proposed leaners are always evaluated against the state of the art predictors by performing statistical analysis. Undoubtedly, given the bias from data sets and accuracy indicators, the newer predictors outperform counterparts in selected experimental settings. However, most of data mining algorithm based predictors (e.g. CART, random forest, neural networks, SVM) have built-in *magic* parameters, like the number of trees in random forest algorithm. the impact of internal parameters in those methods have been neglected during evaluation. In this paper, we investigate this impact by tuning parameters in defect predictors with search-based software engineering algorithm. Specifically, we used differential evolution to tune the CART and a new predictor based on WHERE algorithm with local data, and then predictors with optimal parameters obtained from tuning preocess will be applied to predict defects. By comparing the performance of predictors with and without tuning process, we observe that tuning improves the predictors' performance and predictors woking with different data sets need different parameters. Our results also suggest that we should not use the predictors of the shelf with their default parameters and tuning should be a processor combined with any predictor with built-in parameters.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Cost estimation*

General Terms

Experimentation, Algorithm

Keywords

Defect prediction, Data Mining, Tuning Parameters, CART, WHERE required for Proceedings

1. INTRODUCTION

Software has becoming a large and complex system and delivering reliable and quality software is imperative for development teams. Empirical study shows that the longer the defects exist in software systems, the more the cost of time and money it will take to fix it [**need a ref**]. Therefore, project managers and software programers strive to find defects in their system as early as possible. Defect prediction has been investigated extensively in industrial and academia during the past two decades. As an important research field, building data miners [1, 2, 3, 4, 5, 6, 7] over static code features of software system has been demonstrated to be a way to predict which models are more likely to contain defects.

Classification is an important approach to predict whether some modules in the projects are defective or non-defective. The general idea is to train the learners by using parts of data sets(e.g. ant 1.3, 1.4 in PROMISE[1]) and predict with remaining ones(ant 1.5, 1.6 and 1.7). Many types of defect predictors have been proposed based an different data mining classifier, including

What's the problem in those result?

RQ:

briefly describe our study and our result; observation

structure of this paper.

2. LEARNER AND TUNER

Description of WHERE-based predictor

brief description of CART, Randforest (????)

Description of DE

3. EXPERIMENT

Description of Data set

Experiment 1: naive where vs tuned where

Experiment 2: naive cart vs tuned cart vs random forest

4. DISCUSSION

5. RELATED WORK

[1]http://openscience.us/repo/

Tuning in efforts estimation, software engineering.

Defect Prediction

6. THREATS TO VALIDITY

Internal and external threats

7. CONCLUSION

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] Stefan Lessmann, Bart Baesens, Christophe Mues, and Swantje Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *Software Engineering, IEEE Transactions on*, 34(4):485–496, 2008.

[2] Thomas J McCabe. A complexity measure. *Software Engineering, IEEE Transactions on*, (4):308–320, 1976.

[3] Tim Menzies, Jeremy Greenwald, and Art Frank. Data mining static code attributes to learn defect predictors. *Software Engineering, IEEE Transactions on*, 33(1):2–13, 2007.

[4] Tim Menzies, Zach Milton, Burak Turhan, Bojan Cukic, Yue Jiang, and Ayşe Bener. Defect prediction from static code features: current results, limitations, new approaches. *Automated Software Engineering*, 17(4):375–407, 2010.

[5] Yue Jiang, Bojan Cukic, and Tim Menzies. Can data transformation help in the detection of fault-prone modules? In *Proceedings of the 2008 workshop on Defects in large software systems*, pages 16–20. ACM, 2008.

[6] Tim Menzies, Andrew Butcher, Andrian Marcus, Thomas Zimmermann, and David Cok. Local vs. global models for effort estimation and defect prediction. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pages 343–351. IEEE Computer Society, 2011.

[7] Qinbao Song, Zihan Jia, Martin Shepperd, Shi Ying, and Jin Liu. A general software defect-proneness prediction framework. *Software Engineering, IEEE Transactions on*, 37(3):356–370, 2011.