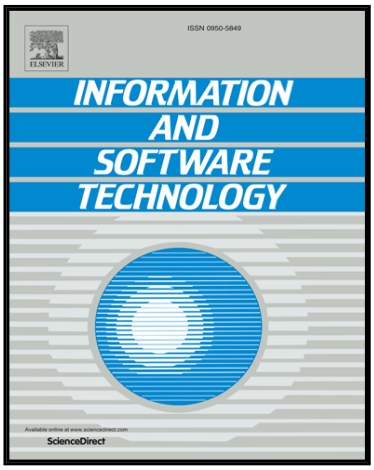

Software Defect Prediction Using Stacked Denoising Autoencoders and Two-stage Ensemble Learning

Haonan Tong , Bin Liu , Shihai Wang

PII: S0950-5849(17)30011-3
DOI: 10.1016/j.infsof.2017.11.008
Reference: INFSOF 5914

To appear in: *Information and Software Technology*

Received date: 7 January 2017
Revised date: 13 October 2017
Accepted date: 13 November 2017

Please cite this article as: Haonan Tong , Bin Liu , Shihai Wang , Software Defect Prediction Using Stacked Denoising Autoencoders and Two-stage Ensemble Learning, *Information and Software Technology* (2017), doi: 10.1016/j.infsof.2017.11.008

Software Defect Prediction Using Stacked Denoising Autoencoders and Two-stage Ensemble Learning

Haonan Tong[12a], Bin Liu[12b], Shihai Wang[12*]
[1]*School of Reliability and Systems Engineering, Beihang University, Beijing, 100083, China*
[2]*Science & Technology on Reliability & Environmental Engineering Laboratory*
[a]Email: tonghaonan@buaa.edu.cn
[b]Email: liubin@buaa.edu.cn
[*]Corresponding author, email: wangshihai@buaa.edu.cn

Abstract: **Context**: Software defect prediction (SDP) plays an important role in allocating testing resources reasonably, reducing testing costs, and ensuring software quality. However, software metrics used for SDP are almost entirely traditional features compared with deep representations (DPs) from deep learning. Although stacked denoising autoencoders (SDAEs) are powerful for feature learning and have been successfully applied in other fields, to the best of our knowledge, it has not been investigated in the field of SDP. Meanwhile, class-imbalance is still a pressing problem needing to be addressed.

Objective: In this paper, we propose a novel SDP approach, *SDAEsTSE*, which takes advantages of SDAEs and ensemble learning, namely the proposed two-stage ensemble (TSE).

Method: Our method mainly includes two phases: the deep learning phase and two-stage ensemble (TSE) phase. We first use SDAEs to extract the DPs from the traditional software metrics, and then a novel ensemble learning approach, TSE, is proposed to address the class-imbalance problem.

Results: Experiments are performed on 12 NASA datasets to demonstrate the effectiveness of DPs, the proposed TSE, and *SDAEsTSE*, respectively. The performance is evaluated in terms of F-measure, the area under the curve (AUC), and Matthews correlation coefficient (MCC). Generally, DPs, TSE, and *SDAEsTSE* contribute to significantly higher performance compared with corresponding traditional metrics, classic ensemble methods, and benchmark SDP models.

Conclusions: It can be concluded that 1) deep representations are promising for SDP compared with traditional software metrics, 2) TSE is more effective for addressing the class-imbalance problem in SDP compared with classic ensemble learning methods, and 3) the proposed *SDAEsTSE* is significantly effective for SDP.

Keywords: software defect prediction; stacked denoising autoencoders; ensemble learning; software metrics; deep learning

1. INTRODUCTION

Software testing, aiming to detect as many defects as possible before the software is released, plays an important role in ensuring software quality. However, with the growth of software scale and complexity, testing cost and duration of traditional software testing are increasing dramatically. How to improve testing efficiency with limited testing resources to assure software quality is a great challenge to practitioners and researchers. Software defect prediction (SDP) technique was proposed to help to allocate testing resources reasonably, determine the testing priority of different software modules, and improve software quality. By using the results of SDP, software practitioners can efficiently judge that which software modules are more likely to be defective, the possible number of defects in a module, or other information related to software defects before software testing. Great achievements have been made in the field of SDP. Existing SDP studies can be divided into four categories: (1) Classification, (2) Regression, (3) Mining association rules, (4) Ranking.

The first category study aim to classify software entities (classes, functions, modules, files, etc.) into defect proneness and non-defect proneness or various levels of defect severity by using statistical techniques (such as discriminant analysis [1], and logistic regression [2]) and machine learning methods (such as support vector machines [3], and artificial neural networks [4]). The second category study aims to estimate the number of defects in the software entities by utilizing various methods, such as genetic programming [5], and support vector regression [6]. The third category study uses association rule mining approaches, such as relational association rule [7], and CBA2 algorithm [8], to mine the relationship between the