Selective Pseudo-Labeling Based Subspace Learning for Cross-Project Defect Prediction

Ying SUN¹, Xiao-Yuan JING††, Nonmembers, Fei WU†††(*), Member, and Yanfei SUN††††, Nonmember

SUMMARY Cross-project defect prediction (CPDP) is a research hot recently, which utilizes the data form existing source project to construct prediction model and predicts the defect-prone of software instances from target project. However, it is challenging in bridging the distribution difference between different projects. To minimize the data distribution differences between different projects and predict unlabeled target instances, we present a novel approach called selective pseudo-labeling based subspace learning (SPSL). SPSL learns a common subspace by using both labeled source instances and pseudo-labeled target instances. The accuracy of pseudo-labeling is promoted by iterative selective pseudo-labeling strategy. The pseudo-labeled instances from target project are iteratively updated by selecting the instances with high confidence from two pseudo-labeling technologies. Experiments are conducted on AEEEM dataset and the results show that SPSL is effective for CPDP.

key words: cross-project defect prediction, pseudo-labeling, subspace learning

1. Introduction

Software defect prediction (SDP) [1] is a vital step in software development process and attracts researchers’ attention both in academia and industry. Most SDP methods can conduct the prediction model by using historical data of a project, and then predicting the new instance that is defective or defect-free from the same project, which are called within-project defect prediction (WPDP) [2] methods. However, lacking sufficient historical defect data makes WPDP perform worse. Fortunately, there are some public labeled datasets that available. Some studies utilize the data of external project (source project) to conduct defect prediction on a new project (target project with no or very limited labeled data), which is called cross-project defect prediction (CPDP) [3]. CPDP is known as a promising solution for above issue. However, the significant gap of data distribution between two projects affect the prediction performance. Prediction model that is trained by utilizing the data from source projects suffers from performance reduction when the model is directly applied to target project. Different projects may contain instances collected from different programming languages and developers, i.e., their distributions are inconsistent, and thus, the model trained from one project performs poorly for instances from other projects. Zimmermann et al. [3] evaluated CPDP performance and only 21 pairs of 622 cross-project pairs performed well. Subspace learning [4] algorithm maps the original feature space to a low dimensional subspace that specific statistical properties can be well preserved. We introduce subspace learning to obtain well-aligned feature representations in the common subspace by conducting subspace transformation using the instances from different projects. In this paper, we can map the source and target project in a common space, where the distribution difference among projects is minimized.

The target project usually does not have labeled data or has limited labeled data. One effective method of exploring target data is assigning pseudo-labels to the target project, and then the instances can be ready for supervised subspace learning. The accuracy of pseudo-labeling process is vital for the feature learning. Most existing pseudolabeling methods overlook the structural information of target project in the process of pseudo-labeling. Selective pseudo-labeling [5], [6] is a way to improve the accuracy of pseudo-labeling, which takes the confidence in target instance labeling into consideration. Inspired by [6], we use selective pseudo-labeling for CPDP. In this paper, we explore the structure information of data and propose a novel approach using both labeled source data and pseudo-labeled target data. The contributions of our study are summarized as follows:

(1) Considering the distribution differences of different projects in CPDP, we learn a common subspace in which the distribution across projects can be similar.

(2) We propose an iterative Selective Pseudo-labeling based Subspace Learning (SPSL) approach for CPDP. SPSL uses selective pseudo-labeling on large amount of unlabeled instances from target together with the labeled instances from source project in common subspace. SPSL can explore the structure information of target project after mapping and promote the accuracy of pseudo-label.

(3) The experiments are conducted on AEEEM dataset, and the experimental results verify the effectiveness of SPSL.
2. Related Work

Recently, CPDP has has drawn researches’ attention and many new CPDP methods are proposed. Zimmermann et al. [3] first performed CPDP model and the experimental results showed that CPDP is challenging. He et al. [7] selected suitable training data. They showed that the selection of training data is important for the prediction results. Turhan et al. [8] proposed a filter method for CPDP by k-nearest neighbor algorithm. Herbold et al. [9] investigated existing 24 CPDP methods and the comparison results show that CamargoCruz09 [10] performs the best. Jing et al. [11] proposed a unified method to solve the class imbalance problem for WPDP and CPDP named SSTCA+ISDA. Zhou et al. [1] proposed simple module size models that utilize ranking strategy for CPDP. Wu et al. [12] proposed the cost-sensitive kernelized semi-supervised dictionary learning (CKSDL) method for CPDP that first utilizes semi-supervised dictionary learning technology in SDP.

3. Proposed Approach

Selective pseudo-labeling based subspace learning aims to find a common subspace where the transformed features of source and target project in the subspace are close to each other. We employ SPSL to learn a transformation matrix which maps instances from both source and target projects into the latent subspace. In this subspace, SPSL combines labeled source instances and pseudo-labeled target instances to iteratively update the transform matrix. The procedure of SPSL is stated in the following subsections.

3.1 Problem Formulation

Given the source project \( X^s = \{x_1^s, x_2^s, \ldots, x_n^s\} \in R^{d_s} \), \( Y^s = \{y_1^s, y_2^s, \ldots, y_m^s\} \in R^{d_t} \). \( x_i^s \) is the \( i \)-th instance and \( y_i^s \in \{1, \ldots, c\} \) is the corresponding label source project, \( m \) is the numbers of the instances in \( X^s \). \( X^t = \{x_1^t, x_2^t, \ldots, x_n^t\} \) is target project dataset, where \( x_i^t \) denotes \( i \)-th instance, \( n \) denotes the numbers of the target instances. \( d \) is the feature dimension.

3.2 Feature Mapping

We aim to learn a common feature subspace \( Z \) for the source and target projects. We learn the optimal transformation matrix \( P \) by minimizing the objective function as follows:

\[
\min_P \sum_{i,j} \left\| P^T \tilde{x}_i - P^T \tilde{x}_j \right\|_2^2 W_{ij} \tag{1}
\]

where \( \tilde{X} = \{\tilde{x}_i\} \in R^{(m+n') \times (m+n')} \) is the collection of labeled source data and selected pseudo-labeled target data, and \( n' \) is the number of selected pseudo-labeled instances. The similarity matrix \( W \) is defined as follows:

\[
W_{ij} = \begin{cases} 
1, & \text{if } y_i = y_j \\
0, & \text{otherwise}
\end{cases}
\tag{2}
\]

where \( W \in R^{(m+n') \times (m+n')} \) is a similarity matrix that captures the pairwise similarities among all the instances regardless of which project they are from. When \( \tilde{x}_i \) and \( \tilde{x}_j \) have the same label, i.e., \( y_i = y_j \), \( W_{ij} \) is set to 1. The similarity matrix also considers the local structure of the instances based on the distance between different instances. We aim at the instances from the same class stay close in the subspace by minimizing the Formula (1). Following the treatment in [6], the Formula (1) is rewritten as:

\[
\max_P \frac{P^T XD\tilde{X}^T P}{P^T (D\tilde{X}\tilde{X}^T + I) P} \tag{3}
\]

where \( L = D - W \) is the Laplacian matrix, and \( D_{ii} = \sum_{j=1}^{m+n'} W_{ij} \) is a diagonal matrix. \( I \) is added for the regularization. The problem in Formula (3) is equivalent to the following generalized eigenvector problem:

\[
\tilde{X}D\tilde{X}^T P = \lambda (D\tilde{X}\tilde{X}^T + I) P \tag{4}
\]

The source and target instances in the learned subspace are \( z^t_i = P^T \tilde{x}^t_i \) and \( z^t = P^T \tilde{x}^t \), and \( z = \{z^t, z^u\} \) is all the source instances and target instances. To get the pseudo-labels of target project for the feature mapping, we describe the steps of selective pseudo-learning in the following sections.

3.3 Selective-Pseudo-Learning

We learn a mapping matrix \( P \) by using labeled data from source project, and then once the target data are labeled, we combine the pseudo-labeled instances from target project with the labeled instances from source project and update the mapping matrix \( P \). The learning process of mapping matrix \( P \) requires labeled instances form source and target projects. The pseudo-labels of target projects are obtained via nearest class prediction and structured prediction. As a result, the feature distributions between different projects in learned subspace becomes more similar and the intrinsic data structures can be preserved.

To improve the accuracy of pseudo-labeling from target project, and avoid propagating the labeling errors to the next iteration during the subspace learning process, we utilize a confidence-aware pseudo label selection strategy, with which we select the instances with high confidence from target project to combine with source instances for the next iteration.

3.3.1 Pseudo-Labeling via Nearest Neighbor Prediction

To make the instances from different classes can be well-separated, we follow [6] and use centralization (\( z \leftarrow z - \bar{z} \)), where \( \bar{z} \) is the mean of \( z \) and L2 normalization (\( z = z/\|z\|_2 \)) to \( z \) in the subspace. The mean vector of instances from
source project whose labels are \( y \) in subspace can be computed as follows:

\[
\bar{L}z_i = \frac{\sum_{j=1}^{m} z_j \delta (y, y_j^z)}{\sum_{j=1}^{m} \delta (y, y_j^z)}
\]

where \( \delta (y, y_j^z) = 1 \) if \( y = y_j^z \), and \( \delta (y, y_j^z) = 0 \) otherwise.

Given target instances \( z_i \), the conditional probability of \( z_i \) belonging to class \( y \) is computed as follows:

\[
p_{\text{nnp}}(y|z_i) = \frac{\exp(-\|z_i - \bar{z}_y\|_2)}{\sum_{y=1}^{k} \exp(-\|z_i - \bar{z}_y\|_2)}
\]

where \( |c| \) denotes the number of classes.

### 3.3.2 Pseudo-Labeling via Structured Prediction

Nearest neighbor prediction does not consider the intrinsic structure of instances from the target projects. Structured prediction can explore the structure information of the target instances in the process of pseudo-learning. In target project, we use K-means to generate clusters of the transformation instances \( z \). The cluster centers are initialized with the mean vector of the source instances in Sect. 3.2.1. Then we match the cluster from target project with the class from the source project, thus the sum of distances of matched pairs is minimized. To realize the match, we define the objective function as follows:

\[
\min \sum_{i=1}^{|s|} \sum_{j=1}^{|t|} B_{ij} L(z_i^s, \bar{z}_j^t)
\]

\[
s.t. \forall i, \sum_j B_{ij} = 1; \forall j, \sum_i B_{ij} = 1
\]

where \( \bar{z}_i^s \) denotes the \( i \)-th cluster of target project and \( \bar{z}_j^t \) denotes the \( j \)-th class of source project. \( B \in \{0, 1\}^{p|s|\times|t|} \) denotes the one-to-one assignment matrix. \( B_{ij} = 1 \) indicates that the \( i \)-th cluster of target project matches the \( j \)-th class of source project. \( L(\cdot) \) denotes the average similarity between source project and target project. Let \( \bar{x}_y \) denote the cluster center corresponding to the class \( y \), the conditional probability of \( z \) belonging to class \( y \) is calculated as follows:

\[
p_{\text{sp}}(y|z) = \frac{\exp(-\|z - \bar{x}_y\|_2)}{\sum_{y=1}^{m} \exp(-\|z - \bar{x}_y\|_2)}
\]

### 3.3.3 Iterative Learning Selection

The above two pseudo-labeling technologies both can provide useful pseudo-labels for target instances. Given an instance from target, nearest neighbor prediction outputs high probability to the instances that close to source instances while structured prediction outputs high probability to the instances that close to the target cluster center. We make use of the complementarity of these two technologies as follows:

\[
j_i^t = \max_{y \in \hat{Y}} \left\{ p_{\text{nnp}}(y|z_i^t), p_{\text{sp}}(y|z_i^t) \right\}
\]

To avoid the labeling errors in the early iterations propagates to the next iteration, we progressively select \( kn/T \) target instances with high confidence in the \( k \)-th iteration, where \( T \) is the number of iteration. We iteratively update the mapping matrix with the combination of labeled instances and pseudo-labeled instances from source and target projects. In updated subspace of the last iteration, we obtain all the pseudo-labels as prediction labels for target project.

### 4. Experiments

#### 4.1 Datasets

In experiment, we employ a defect dataset named AEEEM, which contains five projects. AEEEM is collected by D’Ambros et al. [13]. In this dataset, there exist 61 metrics including seventeen source code metrics (i.e., object-oriented metrics and CK metrics), five previous defect metrics, five entropy-of-change metrics, seventeen entropy-of-source-code metrics, and seventeen churn-of-source code metrics for each project. The details about this dataset is showed in Table 1.

| Project | Number of instances | Percentage of defective instances | Number of metrics |
|---------|---------------------|-----------------------------------|------------------|
| EQ      | 324                 | 39.81                             | 61               |
| JDT     | 997                 | 20.66                             | 61               |
| LC      | 691                 | 9.26                              | 61               |
| ML      | 1862                | 15.16                             | 61               |
| PDE     | 1497                | 13.96                             | 61               |

### 4.2 Baselines and Experiment Settings

We compare our approach with the CPDP methods NNfilter [7], SSTCA+ISDA [11], CamerogCruz [9], [10], CKSDL [12], and ManualDown [1]. In experiments, the parameter \( \lambda \) in formula 1 is set as 1.

We preform one-to-one CPDP in this paper. For example, when EQ in AEEEM acts as the target, there exist 4 cross-project pairs by treating other four projects as the source project separately. We report the average performance of methods for each target project.

### 4.3 Evaluation Measures

We employ two widely used measures for the evaluation: F-measure (FM) and G-measure (GM) [1]. F-measure is harmonic mean of Pd and Pre: $F$-measure $= \frac{2 \times Pd \times Pre}{(Pd + Pre)}$. Pd is defined as $TP/(TP + FN)$. Pre is defined as $TP/(TP + FP)$. 

- **Table 1** Details of project used in experiments

| Project | Number of instances | Percentage of defective instances | Number of metrics |
|---------|---------------------|-----------------------------------|------------------|
| EQ      | 324                 | 39.81                             | 61               |
| JDT     | 997                 | 20.66                             | 61               |
| LC      | 691                 | 9.26                              | 61               |
| ML      | 1862                | 15.16                             | 61               |
| PDE     | 1497                | 13.96                             | 61               |
G-Measure is the geometric mean of Pd and 1-Pf. Pf is defined as $FP/(FP+TN)$. G-measure is defined as $\frac{2Pd(1-Pf)}{Pd+(1-Pf)}$. Here, TP, TN, FP and FN are the number of actually defective instances that are right predicted, the number of defect-free instances that are correctly predicted, the number of defective instances that are wrong predicted, and the number of defective instances that are predicted as defect-free, respectively.

4.4 Experimental Results

In this section, we compare SPSL with baselines. Table 2 reports the mean F-measure and G-measure results on AEEEM dataset. SPSL achieves best prediction results on most projects. These tables show that the proposed approach achieves the best average prediction performance as compared with the baselines on AEEEM dataset. Specifically, on the average F-measure and G-measure values, SPSL improves at least 19.61% and 0.38% on AEEEM dataset. In summary, our approach can outperform state-of-art CPDP methods on AEEEM dataset.

5. Conclusions

In this paper, we propose a new approach SPSL for CPDP. It learns a common subspace on labeled data and pseudo-labeled data from both source and target projects. A selective pseudo-labeling technology is effective for choosing pseudo-labeled instances from target project in iterative subspace learning. Comprehensive experiments are constructed on a widely used AEEEM dataset and the result analyses prove the effectiveness of SPSL. In future, we will explore the potential of SPSL on more complex datasets.

Acknowledgments

This work was supported by the NSFC-Key Project of General Technology Fundamental Research United Fund (No. U1736211), the Natural Science Foundation Key Project for Innovation Group of Hubei Province under Grant (No. 2018CFA024), the National Natural Science Foundation of China (No. 61702280, No. 61772286, No. 61802208), Jiangsu Key Research and Development Program (No. BE2019742), National Postdoctoral Program for Innovative Talents (No. BX20180146), Natural Science Foundation of Jiangsu Province (No. BK20170900), China Postdoctoral Science Foundation (No. 2019M661901), Jiangsu Planned Projects for Postdoctoral Research Funds (No. 2019K024), CCF-Tencent Open Fund WeBank Special Funding (No. CCF-WebankRAGR20191014), and Scientific Research Starting Foundation for Introduced Talents in Nanjing University of Posts and Telecommunications (NUPTSF, No. NY217009).

References

[1] Y. Zhou, Y. Yang, H. Lu, L. Chen, Y. Li, Y. Zhao, J. Qian, and B. Xu, “How far have we progressed in the journey? an examination of cross-project defect prediction,” ACM Trans. Softw. Eng. Methodol., vol.27, no.1, pp.1–51, 2018.
[2] Z.W. Zhang, X.Y. Jing, and T.J. Wang, “Label propagation based semi-supervised learning for software defect prediction,” Automat. Softw. Eng., vol.24, no.1, pp.47–69, 2017.
[3] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, “Cross-project defect prediction: a large scale experiment on data vs. domain vs. process,” Proc. 17th ESEC/FSE, pp.91–100, ACM, 2009.
[4] B. Fernando, T. Tommasi, and T. Tuytelaars, “Joint cross-domain classification and subspace learning for unsupervised adaptation,” Pattern Recognit. Lett., vol.65, pp.60–66, 2015.
[5] W. Zhang, W. Ouyang, W. Li, and D. Xu, “Collaborative and adversarial network for unsupervised domain adaptation,” Prod. 2018 CVPR, 2018.
[6] Q. Wang and T.P. Breckon, “Unsupervised domain adaptation via structured prediction based selective pseudo-labeling.” Proc. 34th AAAI, 2020.
[7] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, “An investigation on the feasibility of cross-project defect prediction,” Automat. Softw. Eng., vol.19, no.2, pp.167–199, 2012.
[8] B. Turhan, T. Menzies, A.B. Bener, and J. Di Stefano, “On the relative value of cross-company and within-company data for defect prediction,” Empir. Softw. Eng., vol.14, no.5, pp.540–578, 2009.
[9] S. Herbold, A. Trautsch, and J. Grabowski, “A comparative study to benchmark cross-project defect prediction approaches,” IEEE Trans. Softw. Eng., vol.44, no.9, pp.811–833, 2017.
[10] A.E. Camargo Cruz and K. Ochimizu, “Towards logistic regression models for predicting fault-prone code across software projects,” Proc. 3rd Int. Symposium Empir. Softw. Eng. Measurement, pp.460–463, IEEE Computer Society, 2009.
[11] X.Y. Jing, F. Wu, X. Dong, and B. Xu, “An improved sda based defect prediction framework for both within-project and cross-project class-imbalance problems.” IEEE Trans. Softw. Eng., vol.43, no.4, pp.321–339, April 2017.
[12] F. Wu, X.Y. Jing, Y. Sun, J. Sun, L. Huang, F. Cui, and Y. Sun, “Cross-project and within-project semisupervised software defect prediction: A unified approach,” IEEE Trans. Reliab., vol.67, no.2, pp.581–597, June 2018.
[13] M. D’Ambros, M. Lanza, and R. Robbes, “Evaluating defect prediction approaches: a benchmark and an extensive comparison,” Empir. Softw. Eng., vol.17, no.4-5, pp.531–577, 2012.