Joint Domain Adaption and Pseudo-Labeling for Cross-Project Defect Prediction

Fei WU†, Member, Xinhao ZHENG†, Ying SUN†, Yang GAO†, and Xiao-Yuan JING††, Nonmembers

SUMMARY Cross-project defect prediction (CPDP) is a hot research topic in recent years. The inconsistent data distribution between source and target projects and lack of labels for most of target instances bring a challenge for defect prediction. Researchers have developed several CPDP methods. However, the prediction performance still needs to be improved. In this paper, we propose a novel approach called Joint Domain Adaption and Pseudo-Labeling (JDAPL). The network architecture consists of a feature mapping sub-network to map source and target instances into a common subspace, followed by a classification sub-network and an auxiliary classification sub-network. The classification sub-network makes use of the label information of labeled instances to generate pseudo-labels. The auxiliary classification sub-network learns to reduce the distribution difference and improve the accuracy of pseudo-labels for unlabeled instances through loss maximization. Network training is guided by the adversarial scheme. Extensive experiments are conducted on 10 projects of the AEEEM and NASA datasets, and the results indicate that our approach achieves better performance compared with the baselines.

key words: cross-project defect prediction, domain adaption, pseudo-labeling, adversarial learning

1. Introduction

Software defect prediction (SDP)[1] is a popular topic in the field of software engineering. Software defect prediction expects to identify defective instances in the early stages of project development. SDP usually trains a model with the historical data of a software project, and then uses the trained model to predict whether other instances within the same project are defective or not, which is called within-project defect prediction (WPDP)[2]. However, a new project often does not have much historical data available for model training, which can cause difficulties for WPDP. One solution is to train a SDP model on historical data from other projects (source projects), then the model can be used to predict whether a new project (target project) contains defective instances or not, which is called cross-project defect prediction (CPDP)[3].

The difficulty of CPDP is that the distributions of source and target projects are usually inconsistent. In recent years, many CPDP methods have been proposed. Cruz et al.[4] studied an application of data transformation by building three univariate logistic regression models to predict fault-prone instances. Ma et al.[5] presented a method called transfer naive Bayes (TNB). By using all proper feature information from the training data, the cross-company data information is transferred into the weights of the training data. Based on these weighted data, a defect prediction model is built. Liu et al.[6] developed a two-phase transfer learning model (TPTL). Two source projects with highly similar distribution to the target project are selected from the candidate projects in the first phase, and then TCA+ based on transfer component analysis (TCA) is used to build a prediction model with the selected projects in the second phase. Zhu et al.[7] proposed a method named CooBA based on adversarial transfer learning, which obtains information from cross-project bug reports and focuses only on common features across projects, and CooBA incorporates adversarial learning to ensure that the common information can be extracted effectively. Yuan et al.[8] presented an active learning and tradaboost (ALTRA) method. It firstly selects some instances from the source project, which have the most similar distributions to that of the target project, and uses a iterative way in the target instances until the specific number of instances are selected. Sun et al.[9] presented a prediction method named discriminative adversarial feature learning (DAFL), which tries to minimize the distribution discrepancy by playing a min-max game between generative model and discriminative model.

The target project usually has no or limited labeled data. To make use of both labeled and unlabeled defect data to build the defect prediction model, some semi-supervised learning methods have been presented. C. Catal[10] introduced the semi-supervised low-density separation (LDS) algorithm, which computes graph-based distances and then applies the gradient TSVM algorithm to defect prediction. Wu et al.[11] presented a cost-sensitive kernelized semi-supervised dictionary learning method, and thus provided a dictionary learning based semi-supervised defect prediction solution.

Furthermore, an feasible solution to explore useful information from both labeled and unlabeled defect data is to generate a pseudo-label for each unlabeled instance. However, a challenge is to ensure the high accuracy of pseudo-labels. Researchers in the field of domain adaptation have addressed several methods to generate labels for target instances. Wang et al.[12] improved the accuracy of pseudo-labels by incorporating supervised subspace learning into an iterative learning framework. Zhang et al.[13] analyzed the
marginal distribution discrepancy to ensure high accuracy of pseudo-labels based on generalized bounds. To ensure desirable accuracy of pseudo-labels while minimizing difference in data distributions, inspired by [13], we propose a novel approach named Joint Domain Adaption and Pseudo-Labeling (JDAPL). The contributions of our work are summarized as follows:

1) To effectively realize domain adaptation and pseudo-labeling for unlabeled target instances jointly, we design a classification sub-network and an auxiliary classification sub-network. The former makes use of the label information of labeled instances to generate pseudo-labels. The latter designs the domain adaptation loss and modified cross-entropy loss on unlabeled instances, and intends to maximize these losses, which acts as the adversary. Network training is guided by the adversarial scheme.

2) To further reduce distribution difference between domains and fully explore discriminant information in defect data, a feature mapping sub-network is designed to transform data of source and target domains into a common space, and supervised contrastive loss is introduced to learn discriminant feature representations.

3) The experiments are conducted on two defect prediction datasets, i.e., AEEEM and NASA, and the experimental results demonstrate the effectiveness of JDAPL.

2. Our Approach

The general network framework of our JDAPL approach is shown in Fig. 1. JDAPL consists of three parts: a feature mapping sub-network with parameter $\theta_m$, a classification sub-network with parameter $\theta_s$, and an auxiliary classification sub-network with parameter $\theta_{sc}$. The feature mapping sub-network is composed of a two-layer fully-connected neural network (FNN) $L1$ and $L2$, the classification sub-network is composed of a one-layer FNN $L3$, and the auxiliary classification sub-network is composed of a one-layer FNN $L4$. Furthermore, we adopt a Gradient Reversal Layer (GRL) [14] before the auxiliary classification sub-network.

2.1 Problem Formulation

Given the source project $X_s = \{x_1^s, x_2^s, \ldots, x_{ns}^s\} \in \mathbb{R}^{d \times ns}$, $Y_s = \{y_1^s, y_2^s, \ldots, y_{ns}^s\} \in \mathbb{R}^{c \times ns}$ is the corresponding label set. $x_i^s$ is the $i^{th}$ instance in $X_s$, $n_s$ is the number of instances in $X_s$, $d$ is the feature dimension, and $c$ is the number of classes.

$X_t = \{x_1^t, x_2^t, \ldots, x_{nt}^t\} = (X_t \cup X_{tu}) \in \mathbb{R}^{d \times nt}$ is the set of instances of the target project, which is divided into the set of labeled instances $X_tl = \{x_1^tl, x_2^tl, \ldots, x_{nl}^tl\} \in \mathbb{R}^{d \times nl}$ and the set of unlabeled instances $X_{tl} = \{x_1^tl, x_2^tl, \ldots, x_{nt}^tl\} \in \mathbb{R}^{d \times nt}$, where $Y_t = \{y_1^t, y_2^t, \ldots, y_{nt}^t\} \in \mathbb{R}^{c \times nt}$ is the corresponding label set for $X_t$, $x_i^t$ represents the $i^{th}$ instance in $X_t$, and $nt_l$ represents the number of instances in $X_{tl}$, $x_i^tl$ represents the $i^{th}$ instance in $X_{tl}$, and $nt_u$ represents the number of instances in $X_{tu}$. The labeled instance set is defined as $X_t = (X_t \cup X_{tl}) \in \mathbb{R}^{d \times nt} = [x_1^t, \ldots, x_{nt}^t]$ with the corresponding label set $Y_t = [y_1^t, \ldots, y_{nt}^t]$, and $nt_l$ is calculated as $nt_l = nt - nt_u$, which represents the number of labeled instances.

2.2 Discriminant Feature Mapping

To align the feature distributions between source and target projects, we construct a feature mapping sub-network, which aims to map the data of source and target projects to a common subspace. We denote the mapping function as $\tilde{X} = f_m(X; \theta_m)$, where $X = \{X_t, X_s\}$, and $\tilde{X} = [\tilde{x}_1, \ldots, \tilde{x}_d]$ denotes the feature matrix after mapping. Our objective is to learn an optimal parameter $\theta_m$, such that the distribution discrepancy between source and target projects can be reduced and the discriminant information can be effectively explored.

In order to make full use of the label information of labeled instances for exploring the discriminant information, inspired by [15], we introduce the supervised contrastive loss $L_{sc}$, which is defined as follows:

$$L_{sc} = \frac{1}{n_t} \sum_{i=1}^{n_t} -\log \frac{\exp(\tilde{x}_i \cdot \tilde{x}_p^l / \tau)}{\sum_{a \in A(i)} \exp(\tilde{x}_i \cdot \tilde{x}_a^l / \tau)}$$

(1)

where $A(i) \equiv \{1, \ldots, n_t\} \setminus \{i\}$ is the set of indexes of all labeled instances except the instance with the index $i$. $P(i) \equiv \{p \in A(i) : y_i^t = y_p^t\}$, and all indexes $p$ in $P(i)$ belong to $A(i)$, and the label of the instance with the index $p$ in $P(i)$ is the same as the label of the instance with index $i$. $|P(i)|$ is its cardinality representing the number of $p$ with the same label as $i$, the symbol $\cdot^\top$ denotes the inner (dot) product, and $\tau \in \mathbb{R}$ is scalar temperature parameter.

The supervised contrastive loss $L_{sc}$ makes the correlation of instances with the same class be as large as possible, while the correlation of instances with different classes be as small as possible, which will facilitate the model to learn discriminative features.

2.3 Joint Domain Adaption and Pseudo-Labeling

In this paper, we adopt a classification sub-network to predict labels for labeled instances. To guide the learning of
classification sub-network, we define the following classification loss:

$$L_c = -\frac{1}{n_t} \sum_{i=1}^{n_t} y_i^t \log(p(x_i^t)) - \frac{1}{n_d} \sum_{i=1}^{n_d} y_i^d \log(p(x_i^d)) \quad (2)$$

where $p(\cdot)$ denotes a probability prediction function. For each input $x_i^t$ or $x_i^d$, the corresponding pseudo-label can be obtained by using this prediction function. In Eq. (2), the label information of labeled instances is fully used to train the classifier.

The designed auxiliary classification sub-network serves two purposes, i.e., to minimize the discrepancy in data distributions between source and target projects, and to improve the accuracy of the pseudo-labels generated by the classification sub-network for unlabeled instances. To minimize the discrepancy in data distributions of two projects, we measure the difference between the source and target projects based on the output of the classification sub-network and the auxiliary classification sub-network. Inspired by [13], we design the domain adaption loss $L_{da}$ as follows:

$$L_{da} = \frac{\gamma}{n_t} \sum_{i=1}^{n_t} p(\hat{x}_i^t) \log(\hat{p}(\hat{x}_i^t)) - \frac{1}{n_t} \sum_{i=1}^{n_t} p(\hat{x}_i^t) \log(p(\hat{x}_i^t))$$

where $\hat{p}(\cdot)$ is the probability prediction function of the auxiliary classification sub-network and $\gamma$ is an adjustable hyper-parameter. This loss should be maximized.

To improve the accuracy of the pseudo-labels generated by the classification sub-network for unlabeled instances, we introduce the modified cross-entropy loss $L_{mc}$ as follows:

$$L_{mc} = -\frac{1}{n_{tu}} \sum_{i=1}^{n_{tu}} p(\hat{x}_i^{tu}) \log(1-p(\hat{x}_i^{tu}))$$

We also need to maximize this loss, such that for each unlabeled instance, the corresponding output of the auxiliary classification sub-network is consistent with the corresponding generated pseudo-label of the classification sub-network.

The overall adversarial learning loss can be defined as follows:

$$\arg\min_{\theta_c, \theta_c} L_c(\theta_m, \theta_c) + \alpha L_{sc} (\theta_m) + \left( L_{mc}(\theta_m, \hat{\theta}_{ac}) - \beta L_{da}(\theta_m, \theta_c, \hat{\theta}_{ac}) \right)$$

where $\alpha$, $\beta$ are balance factors to balance the importance between $L_c$ and $L_{sc}$, and between $L_{mc}$ and $L_{da}$.

### 3. Experiments

#### 3.1 Datasets

In experiment, we use two widely used datasets, i.e., AEEEM and NASA. Each dataset contains 5 projects. Detailed information about these two datasets is shown in Table 1.

#### 3.2 Experimental Settings and Baselines

In experiment, source project contains labeled data. Following the prior studies [9], we select 20% of the instances from the target project as labeled data and the sampling without replacement is performed. We perform 5 random runs to report the average results. We perform one-to-one CPDP prediction, e.g., when EQ in AEEEM is used as the source project, other projects will be selected as the target project for prediction in turn. We report the overall average of prediction results across different one-to-one prediction for each target project on the measures of AUC and $G$-measure. For our approach, we set the maximum number of iterations to 1000.

We perform the comparison between JDAPL and four methods including a semi-supervised method, i.e., LDS [10], and three CPDP methods, i.e., CamcrgoCruz [4], TPTL [6], and DAFL [9]. The hyper-parameters of JDAPL, i.e., the balance factors $\alpha$ and $\beta$ in Eq. (5), $\tau$ in Eq. (1), and $\gamma$ in Equation (3), are tuned using the grid search strategy. Specifically, these hyperparameters are set as $\alpha = 1.0, \beta = 0.95, \tau = 0.07$, and $\gamma = 2.0$, respectively, according to the best prediction results.

#### 3.3 Evaluation Measures

We use two currently widely used measures to evaluate the performance of the model, i.e., AUC and $G$-measure. $G$-measure is the geometric average of $pd$ and $(1-pf)$, and its calculation is related to the threshold, which is defined as

$$G\text{-measure} = \frac{2 \times Pd \times (1 - Pf)}{Pd + (1 - Pf)}$$

where $Pd$ (probability of detection) is defined as $TP/(TP + FN)$ and $Pf$ (probability of false alarm) is defined as $FP/(FP + TN)$. If a defective instance is regarded as a positive and a defect-free instance is regarded as a negative, the prediction can be categorized into four cases: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). The confusion matrix of prediction results is shown in Table 2.
Table 2: Confusion matrix.

|                | Actual defective | Actual defect-free |
|----------------|------------------|-------------------|
| Predicted as   | TP               | FN                |
| defective      |                  |                   |
| Predicted as   | FP               | TN                |
| defect-free    |                  |                   |

Table 3: Comparison results on G-measure for each target project.

| Dataset  | Target project | LDS    | TPTL   | CamargoCruz | DAFL   | JDAPL  |
|----------|----------------|--------|--------|-------------|--------|--------|
| AEEEM    | EQ             | 0.6334 | 0.6247 | 0.6353      | 0.6482 | 0.7246 |
|          | JDT            | 0.6539 | 0.6910 | 0.6911      | 0.7213 | 0.7587 |
| NASA     | LC             | 0.5851 | 0.6874 | 0.6883      | 0.7012 | 0.7064 |
|          | ML             | 0.4897 | 0.5748 | 0.7215      | 0.7317 | 0.7356 |
|          | PDE            | 0.5108 | 0.6555 | 0.5312      | 0.6824 | 0.6946 |
| Overall average | 0.6013 | 0.6252 | 0.6951 | 0.7313 | 0.7410 |

Table 4: Comparison results on AUC for each target project.

| Dataset  | Target project | LDS    | TPTL   | CamargoCruz | DAFL   | JDAPL  |
|----------|----------------|--------|--------|-------------|--------|--------|
| AEEEM    | EQ             | 0.6890 | 0.7230 | 0.6471      | 0.6531 | 0.7299 |
|          | JDT            | 0.6943 | 0.7376 | 0.7123      | 0.7441 | 0.7591 |
| NASA     | LC             | 0.6372 | 0.7553 | 0.6734      | 0.7110 | 0.7199 |
|          | ML             | 0.5369 | 0.6029 | 0.7252      | 0.7363 | 0.7369 |
|          | PDE            | 0.5629 | 0.7196 | 0.5444      | 0.6912 | 0.7020 |
| Overall average | 0.6470 | 0.7178 | 0.6830 | 0.7185 | 0.7392 |

AUC indicates the area under the ROC curve, and it is calculated independent of the threshold. The value range for both AUC and G-measure is between 0 and 1, with higher values indicating better performance.

3.4 Experimental Results

In this section, we compare our approach with the baselines. Tables 3 and 4 show the comparison results on G-measure and AUC of compared methods on AEEEM and NASA datasets. In these tables, the best results are shown in bold. From these tables, compared with competing methods, our approach improves G-measure and AUC at least by 0.0224 = (0.7294 – 0.7070) and 0.0207 = (0.7392 – 0.7185) according to the average results, respectively. These two tables also show that our approach can achieve better results for most of projects.

4. Conclusions

In this paper, we propose a novel approach JDAPL for CPDP. JDAPL maps the features of instances from different projects into a common subspace, uses adversarial training to minimize the discrepancy of data distributions among source and target projects, and makes full use of label and pseudo-label information to improve the discriminative ability of the prediction model. Extensive experiments on the AEEEM and NASA datasets demonstrate the effectiveness of our approach.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 62076139, 61702280), the National Postdoctoral Program for Innovative Talents (No. BX20180146), China Postdoctoral Science Foundation (No. 2019M661901), Jiangsu Planned Projects for Postdoctoral Research Funds (No. 2019K024), Zhejiang Lab (No. 2021KF0AB05), and 131 Talent Program of Nanjing University of Posts and Telecommunications.

References

[1] Y.M. Zhou, Y.B. Yang, H.M. Lu, L. Chen, Y.H. Li, Y.Y. Zhao, J.Y. Qian, and B.W. Xu, “How far we have progressed in the journey? an examination of cross-project defect prediction,” ACM Transactions on Software Engineering and Methodology, 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,” Automated Software Engineering, vol.24, no.1, pp.47–69, 2017.
[3] Z.M. He, F.D. Shu, Y. Yang, M.S. Li, and Q. Wang, “An investigation on the feasibility of cross-project defect prediction,” Automated Software Engineering, vol.19, no.2, pp.167–199, 2012.
[4] A.E.C. Cruz and K. Ochimizu, “Towards logistic regression models for predicting fault-prone code across software projects,” International Symposium on Empirical Software Engineering and Measurement, pp.460–463, 2009.
[5] Y. Ma, G.C. Luo, X. Zeng, and A.G. Chen, “Transfer learning for cross-company software defect prediction,” Information and Software Technology, vol.54, no.3, pp.248–256, 2012.
[6] C. Liu, D. Yang, X. Xia, M. Yan, and X.H. Zhang, “A two-phase transfer learning model for cross-project defect prediction,” Information and Software Technology, vol.107, pp.125–136, 2019.
[7] Z.Y. Zhu, Y. Li, H.H. Tong, and Y. Wang, “Cooba: Cross-project bug localization via adversarial transfer learning,” International Joint Conference on Artificial Intelligence, pp.3565–3571, 2020.
[8] Z.D. Yuan, X. Chen, Z.Q. Cui, and Y.Z. Mu, “Altra: Cross-project software defect prediction via active learning and tradaboost,” IEEE Access, vol.8, pp.30037–30049, 2020.
[9] Y. Sun, X.-Y. Jing, F. Wu, J.J. Li, D.L. Xing, H.W. Chen, and Y.F. Sun, “Adversarial learning for cross-project semi-supervised defect prediction,” IEEE Access, vol.8, pp.32674–32687, 2020.
[10] C. Catal, “A comparison of semi-supervised classification approaches for software defect prediction,” Journal of Intelligent Systems, vol.23, no.1, pp.75–82, 2014.
[11] F. Wu, X.-Y. Jing, Y. Sun, J. Sun, L. Huang, F.Y. Cui, and Y.F. Sun, “Cross-project and within-project semisupervised software defect prediction: A unified approach,” IEEE Trans. Rel., vol.67, no.2, pp.581–597, 2018.
[12] Q. Wang and T. Breckon, “Unsupervised domain adaptation via structured prediction based selective pseudo-labeling,” AAAI Conference on Artificial Intelligence, pp.6243–6250, 2020.
[13] Y.C. Zhang, T.L. Liu, M.S. Long, and M. Jordan, “Bridging theory and algorithm for domain adaptation,” International Conference on Machine Learning, pp.7404–7413, 2019.
[14] K. Osumi, T. Yamashita, and H. Fujiyoshi, “Domain adaptation using a gradient reversal layer with instance weighting,” International Conference on Machine Vision Applications, pp.1–5, 2019.
[15] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y.L. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, “Supervised contrastive learning,” arXiv preprint arXiv:2004.11362, 2020.