Cross-Project Defect Prediction via Semi-Supervised Discriminative Feature Learning

Danlei XING†, Nonmember, Fei WU††a, Member, Ying SUN†, and Xiao-Yuan JING††b, Nonmembers

SUMMARY Cross-project defect prediction (CPDP) is a feasible solution to build an accurate prediction model without enough historical data. Although existing methods for CPDP that use only labeled data to build the prediction model achieve great results, there are much room left to further improve on prediction performance. In this paper we propose a Semi-Supervised Discriminative Feature Learning (SSDFL) approach for CPDP. SSDFL first transfers knowledge of source and target data into the common space by using a fully-connected neural network to mine potential similarities of source and target data. Next, we reduce the differences of both marginal distributions and conditional distributions between mapped source and target data. We also introduce the discriminative feature learning to make full use of label information, which is that the instances from the same class are close to each other and the instances from different classes are distant from each other. Extensive experiments are conducted on 10 projects from AEEEM and NASA datasets, and the experimental results indicate that our approach obtains better prediction performance than baselines.

key words: cross-project defection prediction, semi-supervised learning, discriminative feature learning

1. Introduction

Software defect prediction (SDP) [1] is a popular research topic in software engineering. By mining and analyzing software historical databases, the researchers build the defect prediction datasets, and then use the prediction model trained by these datasets to predict defects on new software projects. However, a new project usually does not have enough historical data to train the prediction model well, so a new solution called cross-project defect prediction (CPDP) [2] was proposed, which uses other projects (source projects) to construct the prediction model for new projects (target projects).

One of the challenges of CPDP is how to reduce the distribution difference between source and target projects and it attracts the attention of researchers at home and abroad. To address the challenge, Herbold [3] selected the most appropriate training data for the target project based on expectation maximization clustering and the nearest neighbor algorithm. Nam et al. [4] transferred knowledge from the source project to the target project by using the transfer component analysis technique [5].

The studies for the semi-supervised learning have made some progress in recent years, many methods based on semi-supervised learning have been proposed for SDP, such as low-density separation (LDS) [6] and random undersampling tri-training [7]. The experimental results show that using both labeled and unlabeled data can improve the prediction performance of the model.

To alleviate the issue of the distribution difference and make full use of the available information from source and target projects, we propose a new approach called Semi-Supervised Discriminative Feature Learning (SSDFL) for CPDP. SSDFL consists of two major parts: knowledge transfer and label prediction. We use a three-layer fully-connected neural network (FNN) to transfer knowledge of both labeled and unlabeled data into a common space. We adopt a two-layer neural network (NN) to predict the labels of all source and target data. To reduce the distribution difference, we minimize the distance between both marginal distributions and conditional distributions of mapped source and target data.

The contributions of this paper are as follows:

1) We propose a semi-supervised method SSDFL for CPDP. SSDFL can use both labeled and unlabeled data to build an accurate prediction model.
2) We introduce the discriminative feature learning to make full use of label information, which means that the instances from the same class can be as closer as possible, and the instances from different classes can be distant from each other.
3) Experiments on 10 projects from NASA and AEEEM datasets show that the SSDFL can obtain competitive prediction performance compared with baselines.

2. Related Work

Up to now, the researchers have proposed lots of methods for CPDP. Turhan et al. [8] proposed a NN-Filter method to choose the best training data from source data based on the k-nearest neighbors. Their experimental results show that filtered source data yields much better results than raw source data. Liu et al. [9] proposed a two-phase transfer learning (TPTL) model that consists of two phases. In the first phase, TPTL selects two projects from source projects by a source project estimator, which have the highest distribution similarity to a target project. Next, transfer com-
ponent analysis plus [4] is used to establish two prediction models based on two selected projects, and the final prediction result is the combined result of two models.

In view of the lack of labeled instances, using both labeled and unlabeled instances is proved to be feasible by researchers and they proposed some semi-supervised methods. Catal [6] compared and analyzed four semi-supervised methods for SDP. In this study, the approach based on low-density separation (LDS) algorithm performs well with limited fault data. Thung et al. [10] proposed an active semi-supervised defect prediction (ASDP) method. ASDP actively selects a small subset of diverse and informative defect examples to label, and makes full use of both labeled and unlabeled instances in the prediction model learning.

3. Our Approach

The details of our approach SSDFL are presented in this section. Suppose that the source project contains a data set $x_s = x_s^{[n_s]} \in R^{d \times n_s}$ and a label set $y_s = y_s^{[n_s]} \in R^{c \times n_s}$, where $x_s^{[i]}$ denotes the $i$th instance in $x_s$. $y_s^{[i]}$ is the corresponding label, $n_s$ is the number of instances in $x_s$, $d$ is the dimension of metrics and $c$ is the number of classes. The target project contains a label set $y_t = y_t^{[n_t]} \in R^{c \times n_t}$ and a data set $x_t = x_t^{[n_t]} \in R^{d \times n_t}$ where $x_t^{[i]}$ denotes the $i$th instance in $x_t$, $y_t^{[i]}$ is the corresponding label, $n_t$ is the number of instances in $x_t$, $n_t$ is defined as $n_t = n_t + n_u$, $d$ and $c$ are same as the definitions described above.

The general framework of SSDFL is shown in Fig. 1. The blue part represents the knowledge transfer process and the green part represents the label prediction process. To mine potential similarities of source and target data, we adopt a three-layer FNN consisting of L1, L2 and L3 to transfer knowledge of source and target data into a common space and a two-layer NN consisting of L4 and L5 to predict labels. The FNN essentially implements a mapping from the input to output. The corresponding rule between input and output can be represented by the mapping function $\phi(\cdot)$.

Due to different projects having different data distributions, if we train a prediction model using the mapped source and target data directly, the prediction performance of the model will be degraded. To alleviate this issue, similar to the prior study [11], we minimize the divergence of marginal distributions between the mapped source and target data, which denotes the distance between the centroids of both the source and target data. We also minimize the divergence of conditional distributions, which denotes the distance between centroids of each class in the mapped source and target data. The divergence loss is defined as

$$L_{divc} = D_m[\phi(x_s), \phi(x_t)] + D_c[\phi(x_s), \phi(x_t)] \tag{1}$$

where $D_m[\phi(x_s), \phi(x_t)]$ is the divergence of marginal distributions and $D_c[\phi(x_s), \phi(x_t)]$ is the divergence of conditional distributions. To be specific, $D_m$ is defined as

$$D_m[\phi(x_s), \phi(x_t)] = \frac{1}{n_t} \sum_{i=1}^{n_t} \bar{x}^i_s - \frac{1}{n_s} \sum_{i=1}^{n_s} \bar{x}^i_t^2 \tag{2}$$

and $D_c$ is defined as

$$D_c = \sum_{i=1}^{c} \frac{1}{n_t} \sum_{j=1}^{n_s} x_{ij}^k - \frac{1}{n_s} \sum_{i=1}^{n_s} x_{ij}^k + \frac{1}{n_t} \sum_{i=1}^{n_t} \sum_{j=1}^{n_s} \lambda_{ij} x_{ij}^k \tag{3}$$

where $x_{ij}^k = \frac{1}{n_t}$ is the $i$th mapped labeled source (target) instance of class $k$, $n_t^k$ ($n_t^k$) denotes the number of labeled source (target) instances of class $k$, and $\lambda_{ij} x_{ij}^k$ is the $i$th mapped unlabeled target instance of class $k$. Additionally, $\lambda_{ij} x_{ij}^k$ is defined as $\frac{n_t^k}{T}$, where $T$ is the total number of iterations, $t$ is the $i$th iteration, and $\lambda_{ij}$ is the probability of $x_{ij}^k$ belonging to class $k$ that is calculated by the softmax function. We observe that the classifier’s performance gradually improves with the increase of the number of the iterations, which means that the labels of unlabeled target data will be more and more reliable. So, we design the adaptive coefficient $\lambda_{ij}$ as the weight of unlabeled target data to calculate the centroids of the classes. The labels of unlabeled target data will be more and more accurate with the gradual increase of the weight of unlabeled target data during the iterative process.

In order to make full use of label information, we introduce the discriminative feature learning, which denotes that the instances from the same class should be as close as possible, and the instances from different classes should be distant from each other. The discriminative loss is formulated as

$$L_{disc} = \sum_{i,j=1}^{n_s} D(\bar{x}_{st}^i, \bar{x}_{st}^j) \tag{4}$$

$$D(\bar{x}_{st}^i, \bar{x}_{st}^j) = \begin{cases} 0, & ||\bar{x}_{st}^i - \bar{x}_{st}^j||_2 < m_1 \quad C_{ij} = 1 \\ \max(0, \frac{||\bar{x}_{st}^i - \bar{x}_{st}^j||_2^2 - m_2}{C_{ij}}), & C_{ij} = 0 \end{cases} \tag{5}$$

where $\bar{x}_{st}^i$ is the $i$th mapped instance in all source and target data, $\bar{x}_{st}^j$ is the $j$th mapped instance in all source and target data, $n_{st}$ is the total number of instances in all source and target data.
target data, \( C_{ij} = 1 \) denotes that \( \tilde{x}_i \) and \( \tilde{x}_j \) are from the same class, and \( C_{ij} = 0 \) means that \( \tilde{x}_i \) and \( \tilde{x}_j \) are from different classes. \( m_1, m_2 \) are the thresholds of the distance between intra-class instances and the distance between the paired inter-class instances \((m_1 > m_2)\).

To obtain accurate classification results, we minimize the error between the actual label and the predicted label of the mapped instances. The classification loss is formulated as

\[
L_{\text{clas}} = \frac{L(y_{sl}, p(\tilde{x}_l))}{n_s + n_t}
\]

where \( L(\cdot, \cdot) \) is the cross-entropy function \([12]\), \( p(\cdot) \) is the softmax function and \( \tilde{x}_l \) is mapped instances in labeled source and target data, and \( y_{sl} \) is the corresponding label.

In general, the objective function of SSDFL is defined as

\[
\min_{\alpha, \beta} L_{\text{clas}} + \alpha L_{\text{disc}} + \beta L_{\text{diver}}
\]

where \( L_{\text{clas}} \) is the classification loss, \( L_{\text{disc}} \) is the discrimina
tive loss, \( L_{\text{diver}} \) is the divergency loss, \( \alpha \) and \( \beta \) are trade-off parameters to balance the importance among \( L_{\text{clas}}, L_{\text{disc}} \) and \( L_{\text{diver}} \).

4. Experiments

4.1 Benchmark Datasets

In this paper, we employ 10 projects as experimental data including 5 projects with the same number of metrics from NASA dataset [13] and 5 projects from AEEEM dataset [14]. The details of these projects are shown in Table 1.

4.2 Evaluation Measures

We employ two widely used evaluation measures to evaluate the performance of methods: G-measure [15] and AUC [16]. The G-measure is defined as

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

where \( Pd \) is defined as \( TP/(TP + FN) \) and \( Pf \) is defined as \( FP/(FP + TN) \). \( TP, TN, FP \) and \( FN \) are defined in Table 2.

AUC is a widely-used evaluation measure to assess the model. Moreover, AUC is independent from the prediction threshold as well as being unaffected by the class imbalance problem. Both AUC and G-measure range from 0 to 1, the higher the value, the better the prediction model performance.

4.3 Experiment Settings and Baselines

In the experiment, we use each project in NASA and AEEEM datasets as the labeled source project, and use the other one from the same dataset as the target project in turn.

We divide the instances from the target project into five parts, and select one of them as labeled data in turn and the rest as unlabeled data. The model parameters \( m_1, m_2, \alpha \) and \( \beta \) of SSDFL are set as 0, 1, 0.05 and 0.5. We set the number of nodes of L1 according to the number of metrics. Furthermore, we set the number of nodes of L2, L3 and L4 according to half, a quarter, and one-eighth of the number of nodes of L1 and we set the number of nodes of L5 according to the number of classes. And then, we make a fine adjustment of the number of nodes and conduct the experiments on each dataset. Finally, we choose one set of values as the number of nodes of L1, L2, L3, L4 and L5 that yields the best G-measure and AUC values on each dataset. The number of nodes in each layer of networks is shown in Table 3.

We compare SSDFL with four methods including two CPDP methods (NN-Filter [8], TPTL [9]) and two SSDP methods (LDS [6], ASDP [10]).

4.4 Experimental Results

The G-measure and AUC results of each target project for all methods are in shown in Table 4 and 5. According to Table 4 and 5, we can observe that SSDFL obtains higher G-measure and AUC values than baselines in most cases. In addition, SSDFL obtains higher overall average values of G-measure and AUC across 10 projects than all baselines. To statistically analyze the detailed results corresponding to Tables 4 and 5, similar to the prior studies [9], we report the Win/Tie/Loss results of SSDFL against each baseline. “Win/Tie/Loss” means that our method obtains a better, equal and worse performance than baselines. According to Table 6, we can observe that SSDFL statistically significantly improves the performance of baselines in most cases as compared with baselines. In general, our approach SSDFL obtains better prediction performance than baselines.

5. Conclusions and Future Works

In this paper, we propose a semi-supervised method SSDFL to solve the cross-project defect prediction problem. Differ-
Table 3 The number of nodes in each layer.

| Layer | AEEEM | NASA |
|-------|-------|------|
| L1    | 61    | 37   |
| L2    | 35    | 23   |
| L3    | 10    | 10   |
| L4    | 5     | 5    |
| L5    | 2     | 2    |

Table 4 Results of G-measure on each project.

| Dataset | Target project | NN-Filter LDS ASDP TPTL SSDFL |
|---------|----------------|-----------------|-----|-----|-----|
| EQ      | 0.4139 0.6334 0.6537 0.6247 0.7073 |
| JDT     | 0.6485 0.6539 0.6625 0.6910 0.7524 |
| AEEEM                            |
| LC      | 0.4813 0.5851 0.6209 0.6874 0.7027 |
| ML      | 0.5415 0.4897 0.5757 0.5748 0.6976 |
| PDE     | 0.5214 0.5108 0.6439 0.6555 0.6941 |
| NASA                            |
| CMI     | 0.5432 0.2652 0.6825 0.6672 0.7103 |
| MW1     | 0.6089 0.6299 0.6533 0.6669 0.7370 |
| AEEEM                            |
| LC      | 0.5763 0.6285 0.6438 0.6995 0.6864 |
| ML      | 0.6133 0.6274 0.7075 0.6884 0.7541 |
| PDE     | 0.7144 0.6292 0.7063 0.6692 0.7934 |
| NASA                            |
| CMI     | 0.5663 0.6013 0.6550 0.6625 0.7235 |

Table 5 Results of AUC on each project.

| Dataset | Target project | NN-Filter LDS ASDP TPTL SSDFL |
|---------|----------------|-----------------|-----|-----|-----|
| EQ      | 0.5790 0.6890 0.7056 0.7230 0.7281 |
| JDT     | 0.6776 0.6943 0.7122 0.7376 0.7543 |
| AEEEM                            |
| LC      | 0.5836 0.6372 0.6080 0.7553 0.7098 |
| ML      | 0.6011 0.5369 0.6589 0.6029 0.7007 |
| PDE     | 0.5820 0.5629 0.6296 0.7196 0.6974 |
| NASA                            |
| CMI     | 0.6044 0.6499 0.7106 0.6999 0.7232 |
| MW1     | 0.6487 0.7675 0.6801 0.7265 0.7452 |
| AEEEM                            |
| LC      | 0.6498 0.6736 0.6948 0.7436 0.6956 |
| ML      | 0.6651 0.6732 0.7203 0.7452 0.7558 |
| PDE     | 0.7212 0.6760 0.7258 0.7260 0.7956 |
| NASA                            |
| CMI     | 0.6312 0.6470 0.6915 0.7178 0.7306 |

Table 6 The Win/Tie/Loss results of SSDFL against each baseline.

| Measure | SSDFL |
|---------|-------|
|         | NN-Filter LDS ASDP TPTL |
| G-measure | 10/0/0 | 10/0/0 | 10/0/0 | 9/0/1 |
| AUC     | 10/0/0 | 10/0/0 | 9/0/1 | 7/0/3 |

ent from most existing methods for CPDP, SSDFL can build an accurate prediction model by using both labeled and unlabeled data. SSDFL also introduces the discriminative feature learning, which can guarantee mapped source and target data with great intra-class compactness and inter-class separability. We perform extensive experiments on AEEEM and NASA datasets, and the experimental results show that our SSDFL is effective.

In the future, we will evaluate our approach with more datasets and adjust it to further improve the prediction performance. We will also study the impact of the ratio of unlabeled data on the prediction performance.

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. 2018CF024), the National Natural Science Foundation of China (No. 61702280), National Postdoctoral Program for Innovative Talents (No. BX20180416), Natural Science Foundation of Jiangsu Province (No. BK20170900), China Postdoctoral Science Foundation (No. 2019M661901), Jiangsu Planned Projects for Postdoctoral Research Funds (No. 2019K024), and CCF-Tencent Open Fund WeBank Special Funding (No. CCF-WebankRAGR20190104).

References

[1] V.U.B. Challagulla, F.B. Bastani, I.-L. Yen, and R.A. Paul, “Empirical assessment of machine learning based software defect prediction techniques,” Int. J. Artif. Intell. Tools., vol.17, no.2, pp.389–400, 2008.
[2] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, “An investigation on the feasibility of cross-project defect prediction,” Autom. Softw. Eng., vol.19, no.2, pp.167–199, 2012.
[3] S. Herbold, “Training data selection for cross-project defect prediction,” Proc. 9th Int. Conf. on Predictive Models in Softw. Eng., no.6, pp.1–10, 2013.
[4] J. Nam, S.J. Pan, and S. Kim, “Transfer defect learning,” Proc. 35th Int. Conf. on Softw. Eng., pp.382–391, 2013.
[5] S.I. Pan, I.W. Tsang, I.T. Kwok, and Q. Yang, “Domain adaptation via transfer component analysis,” IEEE Trans. Neural Netw., vol.22, no.2, pp.199–210, 2011.
[6] C. Catal, “A comparison of semi-supervised classification approaches for software defect prediction,” J. Intell. Syst., vol.23, no.1 pp.75–82, 2014.
[7] Y. Ma, W. Pan, S. Zhu, H. Yin, and J. Luo, “An improved semi-supervised learning method for software defect prediction,” J. Intell. Fuzzy Syst., vol.27, no.5, pp.2473–2480, 2014.
[8] B. Turhan, T. Menzies, A.B. Bener, and J.D. 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] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, “A two-phase transfer learning model for cross-project defect prediction,” Int. Softw. Technol., vol.107, pp.75–82, 2014.
[10] F. Thung, X.-B.D. Le, and D. Lo, “Active semi-supervised defect categorization,” Proc. IEEE 23rd Int. Conf. Program Comprehension, pp.60–70, 2015.
[11] Y. Yao, Y. Zhang, X. Li, and Y. Ye, “Heterogeneous domain adaptation via soft transfer network,” Proc. 27th ACM Int. Conf. on Multimedia, pp.1578–1586, 2019.
[12] G.E. Nasr, E.A. Badr, and C. Joun, “Cross entropy error function in neural networks: Forecasting gasoline demand,” FLAIRS Conf., pp.381–384, 2002.
[13] M. D’Ambros, M. Lanza, and R. Robbes, “An extensive comparison of bug prediction approaches,” Proc. IEEE 7th Working Conf. on Mining Softw. Repositories, pp.31–41, 2010.
[14] T. Menzies, J. Greenwald, and A. Frank, “Data mining static code attributes to learn defect predictors,” IEEE Trans. Softw. Eng., vol.33, no.1, pp.2–13, 2006.
[15] Z.Q. Li, X.-Y. Jing, X.K. Zhu, H.Y. Zhang, B.W. Xu, and S. Ying, “On the multiple sources and privacy preservation issues for heterogeneous defect prediction,” IEEE Trans. Softw. Eng., vol.45, no.4, pp.391–411, 2019.
[16] A.P. Bradley, “The use of the area under the roc curve in the evaluation of machine learning algorithms,” Pattern Recogn., vol.30, no.7, pp.1145–1159, 1997.