Heterogeneous Defect Prediction through Multiple Kernel Learning and Ensemble Learning

Zhiqiang Li*, Xiao-Yuan Jing*†, Xiaoke Zhu*§, Hongyu Zhang‡
*State Key Laboratory of Software Engineering, School of Computer, Wuhan University, Wuhan 430072, China
†School of Automation, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
‡School of Electrical Engineering and Computing, The University of Newcastle, Callaghan NSW 2308, Australia
§School of Computer and Information Engineering, Henan University, Kaifeng 475001, China
(lzq115@163.com, jingxy_2000@126.com, henuzxk@163.com, hongyu.zhang@newcastle.edu.au)
Corresponding author: Xiao-Yuan Jing

***Abstract*—Heterogeneous defect prediction (HDP) aims to predict defect-prone software modules in one project using heterogeneous data collected from other projects. Recently, several HDP methods have been proposed. However, these methods do not sufficiently incorporate the two characteristics of the defect prediction data: (1) data could be linearly inseparable, and (2) data could be highly imbalanced. These two data characteristics make it challenging to build an effective HDP model. In this paper, we propose a novel Ensemble Multiple Kernel Correlation Alignment (EMKCA) based approach to HDP, which takes into consideration the two characteristics of the defect prediction data. Specifically, we first map the source and target project data into high dimensional kernel space through multiple kernel leaning, where the defective and non-defective modules can be better separated. Then, we design a kernel correlation alignment method to make the data distribution of the source and target projects similar in the kernel space. Finally, we integrate multiple kernel classifiers with ensemble learning to relieve the influence caused by class imbalance problem, which can improve the accuracy of the defect prediction model. Consequently, EMKCA owns the advantages of both multiple kernel learning and ensemble learning. Extensive experiments on 30 public projects show that EMKCA outperforms the related competing methods.**

***Keywords*-heterogeneous defect prediction; kernel correlation alignment; multiple kernel learning; ensemble learning; linearly inseparable; class imbalance**

I. Introduction

To build high quality software systems, developers spend much effort on software testing and debugging. However, developers often have limited resources and need to prioritize these efforts. Many software defect prediction techniques have been presented to help with the prioritization of testing efforts by predicting defect-prone modules (instances) [1, 2, 3, 4, 5]. Most existing software defect prediction methods are designed for within-project defect prediction (WPDP), which detects the defect-prone instances using the prediction model built from historical data of the same project [6, 7, 8, 9, 10, 11]. Usually WPDP works when there is a sufficient amount of historical defect data available for training the model. However, in practice, historical defect data may be very limited for some projects [12, 13, 14, 15], which hinders the application of WPDP.

To address this problem, cross-project defect prediction (CPDP), which predicts defects in a project using prediction models trained from historical data of other projects, has been presented [16, 17, 18, 19, 20]. Existing CPDP methods require that the instances of source and target projects have the same metrics (features), i.e., the metric sets should be identical between projects. However, in many cases, source and target projects may share few common metrics [21, 22]. Finding other projects with multiple common metrics can be challenging. In this scenario, the existing CPDP methods using only common metrics cannot obtain desirable results, because the informative metrics necessary for building a good prediction model may not exist across the datasets [21, 22].

Recently, heterogeneous defect prediction (HDP) models [21, 22, 23, 24] are proposed to predict defects across projects with heterogeneous metrics sets, i.e., source and target projects have different metric sets. For example, Jing et al. [21] proposed a transfer CCA+ method by utilizing unified metric representation and CCA-based transfer learning technique for HDP. Nam and Kim [22] employed metric selection and metric matching techniques to predict defects across projects with heterogeneous metric sets.

In general, defect data consists of different types of metrics, e.g., Lines of Code (LOC), complexity, change metrics, and so on. Different metrics quantify the different aspects of the data. In practice, different metrics usually have different physical meanings and distributions, leading to the fact that the defect data instances usually lie on a nonlinear feature space. However, existing HDP methods [21, 22, 23, 24] assume that the feature space of instances is linear, and learn the linear correlation of features between source and target projects. For example, Jing et al. [21] aimed to learn the linear correlation of source and target metrics in the original feature space. Nam and Kim [22] proposed to match source and target metrics by measuring the linear similarity of each source and target metric pair. Hence, these methods may

978-1-5386-0992-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSME.2017.19

face a linearly inseparable problem [25, 26]. In other words, these methods are not able to accurately catch the nonlinear correlations among the defect data.

Furthermore, defect prediction data is often highly imbalanced [27, 28, 29, 30]. That is, the data contains much more non-defective instances (majority) than the defective ones (minority). It is challenging for most conventional classification algorithms to work with data that has an imbalanced class distribution. The imbalanced distribution could cause misclassification of the instances in the minority class, and this is an important factor accounting for the unsatisfactory prediction performance [1]. However, most HDP methods [21, 22, 23] do not take the class imbalance problem into consideration.

In this paper, we consider the above mentioned two characteristics of defect prediction data (i.e., linearly inseparable and class imbalance), and propose a novel HDP approach, named Ensemble Multiple Kernel Correlation Alignment (EMKCA). EMKCA simultaneously utilizes multiple kernel learning and ensemble learning to address the issues of linearly inseparable and imbalanced classification. More specifically:

(1) To handle the linearly inseparable problem, EMKCA maps source and target data into a kernel space, and reduces the differences of data distribution through kernel correlation alignment. In the kernel space, defective and non-defective instances can be better separated, thus the classification performance can be improved. By employing multiple kernel learning technique, EMKCA can make better use of the information contained in the source and target data.

(2) To alleviate the influence of the class imbalance problem, EMKCA combines multiple kernel classifiers together and fuses the prediction results based on the probabilistic outputs. The ensemble classifiers can make full use of the diversity of individual classifiers, thus the misclassification of the instances in the minority class can be decreased [31].

To evaluate the proposed EMKCA approach, we conduct large-scale experiments on 30 public projects from five groups (including NASA, SOFTLAB, ReLink, AEEEM, and PROMISE). The evaluation is centered around three research questions:

- RQ1: *Does EMKCA achieve better prediction performance than WPDP?*
- RQ2: *Does EMKCA achieve better prediction performance than the CPDP methods?*
- RQ3: *Does EMKCA achieve better prediction performance than the existing HDP methods?*

RQ1, RQ2 and RQ3 lead us to investigate whether our approach achieves better prediction performance than the existing WPDP, CPDP and HDP models, respectively. Our experimental results show that EMKCA outperforms related methods. In particular, comparing to WPDP, CPDP and HDP methods, our EMKCA approach improves the mean *AUC* on 30 projects by 13.76%, 12.90% and 9.42%, respectively. The non-parametric statistical significance test and the effect size test confirm the above results.

The remainder of the paper is organized as follows: Section II reviews the related work. Section III presents a detailed account of our approach. The experimental setup and results are given in Section IV. In Section V, we provide discussions about the proposed approach and some threats to validity. The conclusions are drawn in Section VI.

II. Background and Related Work

In this section, we briefly review the typical cross-project defect prediction and heterogeneous defect prediction methods, and point out their limitations.

A. Cross-project Defect Prediction Methods

Cross-project defect prediction (CPDP) refers to building defect prediction models for software modules in a target project using historical data from other existing projects. In recent years, we have witnessed a lot of interest in developing new CPDP methods [16, 32, 18, 33, 34, 35, 36]. Zimmermann et al. [12] performed 622 cross-project predictions and showed that careful selection of training data and the characteristics of data and process play important roles in successful CPDP. Turhan et al. [13] presented a nearest-neighbor filter (NN-filter) method to select training data close to within-project data. Ryu et al. [37] investigated the applicability of the class imbalance learning under CPDP setting and designed a boosting based model named value cognitive boosting (VCB). It sets similarity weights according to distributional characteristics, and combines the weights with the asymmetric misclassification cost designed by the boosting algorithm for appropriate resampling. Jing et al. [38] first employed the semi-supervised transfer component analysis (SSTCA) method to make the distributions of source and target data consistent, and proposed the SSTCA+ISDA prediction approach by combining ISDA (improved subclass discriminant analysis) for cross-project class imbalance problem.

However, existing CPDP methods learn the prediction models by using the common metrics contained in the source and target projects. In practice, the size of common metric sets across source and target data may be very small [21, 22], which will limit their prediction performance.

B. Heterogeneous Defect Prediction Methods

Recently, several HDP methods have been proposed [21, 22, 23, 24]. Since the metrics except the common metrics might have favorable discriminant ability, Jing et al. [21] proposed a HDP method, namely CCA+, which utilizes unified metric representation (UMR) and CCA-based transfer learning technique. Specifically, the UMR consists of three types of metrics, including the common metrics of the source and target projects, source-project-specific metrics, and target-project-specific metrics. By learning a pair

of projective transformations under which the correlation of the source and target data is maximized, CCA+ can make the data distribution of the target similar to that of the source. Experiments on public projects indicate that CCA+ can achieve good prediction results. Meanwhile, Nan and Kim [22] presented another solution for HDP. They firstly employed the metric selection technique to remove redundant and irrelevant metrics for source data. Then, they matched up the metrics of source and target data based on their similarity such as distribution or correlation. After these processes, they finally arrived at a matched source and target metric sets. With the obtained metric sets, they built HDP model to predict labels of target instances. They found that about 68% of HDP predictions outperform or are comparable to WPDP predictions with statistical significance. Additionally, He et al. [23] developed CPDP-IFS to address the problem of heterogeneous metric sets (Imbalanced Feature Sets) in CPDP. They used distribution characteristics vectors [14] of each instance as new metrics to enable defect prediction. Recently, Cheng et al. [24] presented a cost-sensitive correlation transfer support vector machine (CCT-SVM) method to deal with the class imbalance problem in HDP based on CCA+ [21]. They took different misclassification costs into consideration by incorporating the cost factors into the SVM model to alleviate the impact of imbalanced data.

C. Limitations of Existing HDP Methods

Although the HDP methods described in the previous section have made much progress, there are still ample room for further improvement. We found that these existing HDP methods do not sufficiently consider the two characteristics of the defect prediction data:

(1) Defect data could be not linearly separable [25, 26]. In practice, defect data is usually constituted by multiple different types of metrics, e.g., the NASA dataset [6, 39] contains Halstead, McCabe complexity, LOC and other miscellaneous metrics. These metrics reflect different aspects of defect data, and have different distributions from each other. For instance, cyclomatic complexity metric is subject to exponential distribution [6] and LOC may follow a lognormal distribution [40, 41]. These factors mean that defect data may not be linearly separable.

(2) Defect data could be highly imbalanced [27, 28, 29, 30], i.e., the number of non-defective instances is much more than that of defective ones. As shown in Table I, the percentage of defects is low in most projects, especially for the projects like PC1, AR1, LC and tomcat6.0, whose data is highly imbalanced.

These two data characteristics make it challenging to build an effective HDP model. With respect to the linearly inseparable problem, these HDP methods learn the defect predictors by regarding the feature space of instances to be linear, which will bring negative influence to the decision of classifiers [9], and further affect the subsequent prediction. With respect to the class imbalance problem, most HDP methods do not pay attention to this problem, therefore restricting their prediction performance [1]. Different from these methods, in this paper, we attempt to deal with the above mentioned two problems simultaneously.

III. Proposed Approach

In HDP, the main challenge is to overcome the data distribution differences between source and target projects when learning a prediction model [21, 22]. In fact, HDP can be viewed as a specific case of heterogeneous transfer learning, which aims to transfer the knowledge learned from a source project to a target project. Domain adaptation is an advanced transfer learning technique [42], which aims to handle distinct distributions between source and target domains, and uses the data in source domain to build a classifier that will perform well in target domain.

Correlation alignment [43], which is a recently presented unsupervised domain adaptation method, has achieved good results for object recognition and sentiment analysis tasks. However, it is only designed for homogeneous domain adaptation, i.e., the source and target domains have the same types of features. So it cannot be directly used for HDP. Motivated by [43], we design an Ensemble Multiple Kernel Correlation Alignment (EMKCA) based approach to HDP. Specifically, to address the linearly inseparable problem, EMKCA utilizes multiple kernel learning to map source and target data to high dimensional kernel space, and reduces the difference of data distribution through correlation alignment. To alleviate the influence caused by class imbalance problem, EMKCA employs ensemble learning to incorporate multiple kernel classifiers.

In this section, we first define some notations used in the paper, and then review the correlation alignment method. Finally, we describe the proposed EMKCA approach.

A. Notation

For the heterogeneous source and target data, assume that the source contains a data set $X_s = x_s^i|_{i=1}^{n_s}$ and a label set $Y_s = y_s^i|_{i=1}^{n_s}$, where x_s^i denotes the i^{th} instance in X_s, y_s^i is the corresponding label, and n_s is the number of instances in X_s. The target consists of an unlabeled data set $X_t = x_t^i|_{i=1}^{n_t}$, where x_t^i denotes the i^{th} instance and n_t is the number of instances in X_t. Instance in source X_s can be represented as $x_s^i = [a_s^{i1}, a_s^{i2}, \cdots, a_s^{id_s}]$ and instance in target X_t can be represented as $x_t^i = [a_t^{i1}, a_t^{i2}, \cdots, a_t^{id_t}]$. Here, a_s^{ij} (a_t^{ij}) represents the metric value corresponding to the j^{th} metric of x_s^i (x_t^i), d_s and d_t denote the size of metrics for the source and target data, respectively. It is worth mentioning that, the metric type or metric set size in X_s and X_t is different. We employ the z-score normalization (i.e., zero mean and unit standard deviation) to preprocess data, which is similar to the N2 normalization in [16].

B. Correlation Alignment

Correlation alignment [43] aims to transform the source domain to the target domain space. The idea is to minimize domain shift by aligning the second-order statistics (covariance) of source and target distributions in an unsupervised manner. It is a simple, effective and efficient domain adaption method.

To minimize the distance between the second-order statistics of the source and target data distributions, the objective of correlation alignment is to learn a linear transformation A for the original source metrics and is formulated as follows:

$$\min_{A} \left\| C_{\hat{S}} - C_T \right\|_F^2 = \min_{A} \left\| A^T C_S A - C_T \right\|_F^2 \qquad (1)$$

where $C_{\hat{S}}$, C_S and C_T are the covariance matrices of transformed source data $\hat{X}_s = X_s A$, source data X_s and target data X_t, respectively. $|| \cdot ||_F$ denotes the matrix Frobenius norm, which is defined as the square root of the sum of the absolute squares of its elements.

After some derivation, the solution of Eq. 1 can be solved as $A = C_S^{\frac{-1}{2}} C_T^{\frac{1}{2}}$. Then based on the transformed source data $\hat{X}_s$, the model can be built and the target data X_t can be predicted. Since correlation alignment changes the metrics only, it can be applied to any base classifiers.

C. EMKCA

Kernel-based learning methods [44, 45, 9] can map the historical defect data into a high dimensional feature space to improve its separability. Compared with single kernel learning, multiple kernel learning [44] can assemble different types of kernel functions, and leverage the advantage of each basic kernel function to improve the prediction accuracy. Ensemble learning uses a set of classifiers to make predictions and is widely exploited for the class imbalance problem [31, 28]. The generalization ability of an ensemble classifier is generally much stronger than that of each subclassifier. In this section, we mainly describe the designed EMKCA approach.

1) Multiple Kernel Correlation Alignment: Suppose that there are two nonlinear mappings $\phi : x_s \rightarrow \phi(x_s)$ and $\varphi : x_t \rightarrow \varphi(x_t)$, which can map X_s and X_t into high dimensional kernel feature space. And the mapped X_s and X_t can be formed as follows:

$$\begin{array}{l} \phi(X_s) = \left[\phi(x_s^1), \phi(x_s^2), \cdots, \phi(x_s^{n_s})\right] \\ \varphi(X_t) = \left[\varphi(x_t^1), \varphi(x_t^2), \cdots, \varphi(x_t^{n_t})\right] \end{array} , \qquad (2)$$

The kernel matrices of source and target data separately can be obtained by $K_s = k(X_s, X_s) = \phi(X_s)^T \phi(X_s) \in \mathcal{R}^{n_s \times n_s}$ and $K_t = k(X_t, X_t) = \varphi(X_t)^T \varphi(X_t) \in \mathcal{R}^{n_t \times n_t}$, where $k(\cdot, \cdot)$ is the kernel function.

To address the heterogeneous metrics problem, we use incomplete Cholesky decomposition (ICD) [46] to reduce kernel matrix. The goal of ICD is to find a low-rank approximation matrix Z of size $n \times r$, for small r, such that the difference $K - ZZ^T$ becomes arbitrarily small. It generally takes $O(n^3)$ operations to compute the entire kernel matrix $K \in \mathcal{R}^{n \times n}$, while ICD allows approximations to the kernel matrix in $O(r^2 n)$ operations (typically $r \ll n$) [46]. Besides computationally efficient, it has been shown that the reduced kernel can approximate the solution of the full kernel matrix well [47].

Algorithm 1 The EMKCA approach to HDP.

Input: Source project data X_s and the corresponding class labels Y_s, target project data X_t, M kernel functions $k(\cdot, \cdot)$ and the rank r

Output: The ensemble output H

Procedure:

1: Preprocess X_s and X_t using z-score normalization;
2: *for* $i = 1 : M$
(1). Map X_s and X_t into kernel spaces $\phi(X_s^i)$ and $\varphi(X_t^i)$ by using Eq. (2) to obtain K_s^i and K_t^i;
(2). Use ICD to obtain Z_s^i and Z_t^i of rank r;
(3). Obtain transformed source data $\hat{Z}_s^i = Z_s^i \hat{C}_S^{\frac{-1}{2}} \hat{C}_T^{\frac{1}{2}}$ according to the covariance matrices Z_s^i and Z_t^i;
(4). Compute Euclidean distance of $\hat{Z}_s^i$ and Z_t^i to obtain w_i ;
(5). Obtain an output for each pair of $\hat{Z}_s^i$ and Z_t^i;
endfor
3: Calculate the weight w by using Eq. (4);
4: Use Eq. (3) to obtain the ensemble output H by combining M base classifiers.

2) Ensemble Output: Considering the benefits of multiple kernel learning and ensemble learning, we use multiple kernel learning to exploit multiple different feature spaces and combine diversity information with ensemble to improve the prediction performance. Let $k_1, k_2, \cdots, k_M$ be a set of base kernel functions that would be used to compute kernel matrices. We can separately obtain M transformed source $\hat{Z}_s$ and target Z_t. For the i^{th} pair of $\hat{Z}_s^i$ and Z_t^i, we can obtain an output based on the base classifier h_i. Then, we can obtain an ensemble output by combining M base classifiers:

$$H(Z_t) = \sum\nolimits_{i=1}^{M} w_i h_i(Z_t^i), \qquad (3)$$

where w_i is the weight for h_i, and the weights w_i's are usually assumed to be constrained by $w_i \geq 0$, $\sum_{i=1}^{M} w_i = 1$.

A simple way is to get the weight for each base classifier by using averaging ensemble, which treats each base clas-

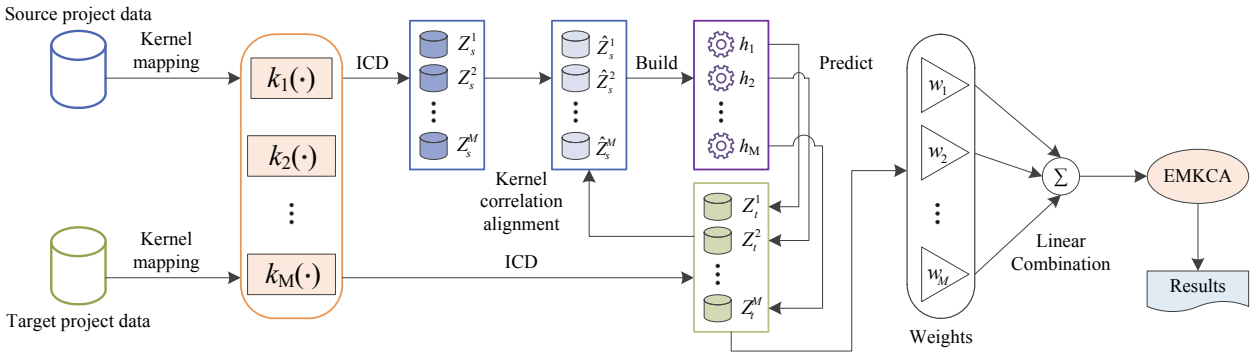

Figure 1. Overview of the proposed EMKCA approach for HDP.

sifier with equal importance. However, it may not hold true in practice, the outputs of individual classifiers should have different weights implying different importance. To this end, we use the popularly used Euclidean distance as similarity measure to calculate the weight of each base classifier. The weights are calculated according to the following rules. The more similar the data distribution of transformed source and target is, the smaller their distance is, and the larger the weight is. Specifically, for the i^{th} pair of $\hat{Z}_s^i$ and Z_t^i, we firstly calculate the mean distance d_i between $\hat{Z}_s^i$ and Z_t^i using Euclidean distance. Then, the weight w_i can be obtained by set it to $1/d_i$. Based on the obtained M weights, we normalize them as follows:

$$w_i = \frac{w_i}{\sum_{i=1}^{M} w_i} \in (0,1), i = 1, 2, \cdots, M. \quad (4)$$

Figure 1 illustrates the overall architecture of utilizing EMKCA for HDP and Algorithm 1 provides the detail execution process of EMKCA.

IV. Experiments

A. Benchmark Datasets

We employ 30 publicly available and commonly used projects from five different groups including NASA[1] [39], SOFTLAB[1] [13], ReLink[2] [48], AEEEM[3] [49] and PROMISE[1] [50] as the experiment data. Table I shows the details about these datasets.

NASA benchmark dataset is publicly available and widely used for defect prediction [6, 39]. Each dataset in NASA represents a NASA software system or sub-system, which contains the corresponding defect-marking data and various static code metrics. Static code metrics of NASA datasets include size, readability, complexity etc., which are closely related to software quality. We use only five projects including CM1, MW1, PC1, PC3 and PC4 from NASA dataset, since these five projects have 37 common metrics.

[1]http://openscience.us/repo/
[2]http://www.cse.ust.hk/~scc/ReLink.htm
[3]http://bug.inf.usi.ch/

Turkish software company (SOFTLAB) consists of AR1, AR3, AR4, AR5 and AR6 projects, which are embedded controller software for white-goods [13]. The projects in SOFTLAB are obtained from PROMISE repository. SOFTLAB has 29 metrics which includes Halstead and McCabe's cyclomatic metrics.

Datasets in ReLink were collected by Wu et al. [48] to improve the defect prediction performance by increasing the quality of the defect data. The defect information in ReLink has been manually verified and corrected. ReLink consists of three open source projects and each one has 26 complexity metrics.

AEEEM was used to benchmark different defect prediction models and collected by D'Ambros et al. [49]. Each AEEEM dataset consists of 61 metrics: 17 source code metrics, 5 previous-defect metrics, 5 entropy-of-change metrics, 17 entropy-of-source-code metrics, and 17 churn-of-source code metrics [16, 22].

The fifth group was originally collected by Jureczko and Madeyski [50] from the online PROMISE data repository, which consists of some open source projects. The datasets have 20 metrics in total, which contains McCabe's cyclomatic metrics, CK metrics and other OO metrics.

B. Evaluation Measure

In this paper, we employ the commonly used measure to evaluate the performance of defect prediction models: *AUC* [51, 52, 53, 22, 54].

AUC is the area under the receiver operating characteristic curve. This curve is plotted in a two-dimensional space with false positive rate as x-coordinate and the true positive rate (recall) as y-coordinate. The *AUC* is known as a useful measure for comparing different models and is widely used because it is unaffected by class imbalance and is independent of the prediction threshold. The default cut-off threshold is 0.5, which may not be the best cut-off value for other performance measures in practice [20]. Lessmann et al. [51] and Ghotra et al. [53] suggest to use the *AUC* for better comparability. Hence, we select *AUC* as our performance

Table I
DETAILS OF PROJECT USED IN EXPERIMENT

Group	Project	# of metrics	# of total instances	# of defective instances	% of defective instances
NASA	CM1	37	327	42	12.84%
	MW1	37	253	27	10.67%
	PC1	37	705	61	8.65%
	PC3	37	1077	134	12.44%
	PC4	37	1458	178	12.21%
SOFTLAB	AR1	29	121	9	7.44%
	AR3	29	63	8	12.70%
	AR4	29	107	20	18.69%
	AR5	29	36	8	22.22%
	AR6	29	101	15	14.85%
ReLink	Apache	26	194	98	50.52%
	Safe	26	56	22	39.29%
	ZXing	26	399	118	29.57%
AEEEM	EQ	61	324	129	39.81%
	JDT	61	997	206	20.66%
	LC	61	691	64	9.26%
	ML	61	1862	245	13.16%
	PDE	61	1497	209	13.96%
PROMISE	ant1.7	20	745	166	22.28%
	camel1.6	20	965	188	19.48%
	ivy2.0	20	352	40	11.36%
	jedit4.0	20	306	75	24.51%
	log4j1.0	20	135	34	25.19%
	lucene2.4	20	340	203	59.71%
	poi3.0	20	442	281	63.57%
	synapse1.2	20	256	86	33.59%
	tomcat6.0	20	858	77	8.97%
	velocity1.6	20	229	78	34.06%
	xalan2.4	20	723	110	15.21%
	xerces1.3	20	453	69	15.23%

measure. The higher *AUC* represents the better prediction performance and the *AUC* of 0.5 means the performance of a random predictor [52].

C. Baselines

We compare EMKCA with WPDP, NN-filter [13], VCB [37], SSTCA+ISDA [38], CPDP-IFS [23], CCA+ [21], HDP-KS (KSAnalyzer led to the best prediction performance in the paper) [22] and CCT-SVM [24].

Comparison with WPDP: Comparing EMKCA to WPDP will provide empirical evidence of whether EMKCA is applicable in practice.

Comparison with the CPDP methods: NN-filter, VCB and SSTCA+ISDA are three typical CPDP models. All of them demand that the instances of source and target projects have totally same metrics. For VCB and SSTCA+ISDA, these two methods are designed to address the cross-project class imbalance problem.

Comparison with the HDP methods: CPDP-IFS, CCA+, HDP-KS and CCT-SVM are four HDP models. These methods have achieved encouraging prediction results. Among them, only CCT-SVM considers the class imbalance problem. However, CCT-SVM utilizes the cost information in the classification stage, and it is only applied to the SVM classifier, thus it cannot be applied to other classifiers directly. Comparing to the experiments for CCT-SVM with 14 projects in four dataset groups, we conducted more extensive experiments with 30 projects in five dataset groups.

We implement the above baselines in MATLAB following the settings of the corresponding papers. The logistic regression (LR) classifier has been widely used in the defect prediction studies [12, 16, 22, 18, 55, 36]. To be fair, we choose LR for all compared methods except for CCT-SVM which uses the default SVM classifier [24]. Comparison of EMKCA and CCT-SVM using SVM classifier will be reported in Section IV-F4. We apply z-score normalization to the source and target data before running these methods.

D. Experimental Settings

We use 30 projects from NASA, SOFTLAB, ReLink, AEEEM and PROMISE groups as experimental data, and perform heterogeneous cross-project defect prediction. We select one project from 30 projects as the target, and each project from other groups is used as the source in turn. For example, when CM1 in NASA group of Table I acts as the target, there exists 25 (30-5) cross-project prediction combinations. Since we mainly focus on prediction across datasets with heterogeneous metric sets, we did not conduct defect prediction across projects in the same group where datasets have the same metric sets. In total, we have 672 possible prediction combinations from these 30 projects of five groups.

For WPDP, it is necessary to split datasets into training and test sets. We apply a two-fold cross validation that are used in the defect prediction models [16, 22, 55, 54]. Specifically, we use the first half for training and the second half for test. Then we use the second half for training and the first half for test in a reverse way. To tackle the randomness of sampling, we repeat the random splits for 50 times. In total, we have 100 tests for the classifier on each target. For the other methods, we use the same test data and test splits as used in the WPDP. We repeat the above cross-project study 100 times, each time using the available source data for training the models. Then, we report the mean results and the corresponding standard deviations for each target project.

E. Parameter Settings

For all experiments, we use a total of 10 base kernels $k(x_i, x_j)$ including 9 Gaussian kernels $e^{-||x_i-x_j||/2\sigma^2}$ with different σ in $(2^{-4}, 2^{-3}, \cdots, 2^4)$ and a linear kernel $x_i^T x_j$ on all features, where σ is the kernel parameter. For the parameter r used in ICD, we set $r = 60$ by default, and the different values of r will be discussed in Section V-A. For the other compared methods, we follow the default parameter settings used in their papers.

The LR is implemented in the popular LIBLINEAR [56] (an award-winning library for large linear classification) package. For LIBLINEAR execution, we use the options

"-S 0" (i.e., logistic regression) and "-B 1" (i.e., no bias term added) as used by Nam et al. [16].

F. Results

1) Comparison Result with Baselines: Table II reports the mean prediction results of AUC for each target project. The mean and median *AUC* across 30 target projects are also reported in the fourth and third last rows of the table (denotes "Mean", "Median") and the best values of each target project are in bold font. From the table, we can see that EMKCA achieves the best *AUC* performance in most cases when compared with the baselines. In terms of the overall performance on 30 target projects, EMKCA improves the mean *AUC* by 13.76%, 27.40%, 18.85%, 12.90%, 16.80%, 16.43% and 9.42% over WPDP, NN-filter, VCB, SSTCA+ISDA, CPDP-IFS, CCA+ and HDP-KS, respectively.

Possible reasons that EMKCA achieves better results are as follows. *Compared with WPDP:* Generally, good prediction models can be built when sufficient historical defect data is available for WPDP [12, 13, 14]. However, in practice, it is difficult to collect sufficient historical defect data for new projects. In this case, the performance of WPDP is usually limited. Different from WPDP, EMKCA uses source data from other projects which may contain some useful information for the target. *Compared with the CPDP methods including NN-filter, VCB and SSTCA+ISDA:* These CPDP methods only use the common metrics shared by the source and target projects, which will limit their performance, because some informative metrics necessary for building a good prediction model may not be in the common metrics across projects [21, 22]. *Compared with the HDP methods including CPDP-IFS, CCA+ and HDP-KS:* (1) These methods mainly focus on learning linear correlation or similarity between the source and target metrics. Different from them, EMKCA aims to learn the nonlinear correlation using multiple kernel technique and can accurately capture the nonlinear correlations among the data. The kernel techniques have been theoretically and empirically proven to be able to tackle linearly inseparable problems by transforming data to high dimensional kernel space [25, 26]. (2) These methods also do not consider the class imbalance problem. The class imbalanced distribution will make a lot of instances in the minority class misclassified, which degrades their prediction performance [27, 1]. On the contrary, EMKCA uses ensemble learning [31] to make predictions, which can relieve the influence caused by the class imbalance problem.

Besides, we observe that the performance of EMKCA in most SOFTLAB projects are not as good as that of the baselines. Possible reasons are that: (1) EMKCA may be sensitive to the projects with fewer instances such that it does not achieve good performance. As shown in Table I, we can see that most SOFTLAB projects have fewer instances than other datasets, especially for the projects like AR3 and AR5. (2) The weights based on Euclidean distance similarity for each base classifier is not optimized, such that their weight combination is not optimal for these targets. In this case, each base classifier does not play a positive role for the final ensemble output, they may be counterproductive and conflicting instead. Hence, the prediction performance will be degraded. In the future, we will use some heuristic or search-based optimization methods to learn the weight of each base classifier for the better ensemble.

2) Statistical Significance Test: To statistically analyze the detailed prediction results, we conduct a non-parametric Mann-Whitney U test at a confidence level of 95% on 20 random running. This statistic test has been used in the defect prediction studies [57, 13, 17, 54]. The advantage of using non-parametric statistical method is that it makes no assumptions about the distribution of the data. As recommended by Menzies et al. [57], Mann-Whitney U test does not demand that the two compared populations are of the same size and it can avoid the need of Bonferroni-Dunn test to counteract the results of multiple comparisons [58]. So we choose this test for evaluation. Then, we report the win/tie/loss (w/t/l) results of our approach against each baseline, like the work described in [59, 34, 14, 35, 18]. "Win" means that the results of our approach are significantly better than those of the baselines at a confidence level of 95%, "tie" means "equal" (no statistical significance), and otherwise "lose".

The second last row in Table II shows the w/t/l results among the baselines and our approach. By using the w/t/l evaluation, we can investigate the number of projects in which our approach can outperform the compared methods. From Table II, EMKCA can statistically significantly improve the performance of baselines in most cases. For example, compared with HDP-KS, EMKCA obtains statistically significant improvements for 23 projects out of 30 in terms of *AUC*. It demonstrates that the proposed EMKCA approach is effective with statistical significance.

3) Effect Size Test: To measure the degree of differences in *AUC* results between our approach and the compared methods, we compute Cliff's delta (δ), which is a non-parametric effect size test [60]. This test has been used in the related defect prediction studies [38, 18, 5, 54, 20, 61]. In this context, δ is a measure of how often the values in one method are larger than the values in a second method. All possible values of δ are in the closed interval $[-1, 1]$, where -1 or 1 indicates that all values in one method are smaller or larger than those of the other method, and 0 indicates that the measure in the two methods is completely overlapping. As Romano et al. suggested [60], the magnitude of the effect size is as follows: *negligible* (N, $|\delta| < 0.147$), *small* (S, $0.147 \leq |\delta| < 0.33$), *medium* (M, $0.33 \leq |\delta| < 0.474$) and *large* (L, $|\delta| \geq 0.474$).

The last row in Table II shows Cliff's δ for effect size among the baselines and our approach. As mentioned above,

Table II
MEAN *AUC* RESULTS (± STANDARD DEVIATION) FOR EACH TARGET PROJECT USING DIFFERENT METHODS

Target	WPDP	NN-filter	VCB	SSTCA+ISDA	CPDP-IFS	CCA+	HDP-KS	EMKCA
CM1	0.591±0.059	0.622±0.095	0.605±0.114	0.635±0.071	0.612±0.050	0.581±0.065	0.665±0.074	**0.842±0.113**
MW1	0.601±0.064	0.599±0.168	0.681±0.071	0.674±0.071	0.626±0.079	0.611±0.097	0.669±0.128	**0.855±0.084**
PC1	0.641±0.055	0.586±0.122	0.611±0.106	0.631±0.062	0.628±0.059	0.590±0.064	0.670±0.150	**0.885±0.144**
PC3	0.659±0.043	0.575±0.127	0.612±0.108	0.613±0.097	0.616±0.053	0.562±0.062	0.646±0.159	**0.841±0.202**
PC4	0.729±0.039	0.589±0.150	0.629±0.126	0.661±0.064	0.652±0.053	0.561±0.055	0.615±0.139	**0.798±0.149**
AR1	0.542±0.090	0.539±0.116	0.561±0.110	0.560±0.097	0.592±0.107	0.602±0.131	**0.709±0.110**	0.628±0.170
AR3	0.607±0.119	0.681±0.181	0.709±0.126	0.710±0.109	0.755±0.105	0.690±0.167	**0.831±0.106**	0.593±0.202
AR4	0.662±0.084	0.737±0.152	0.712±0.187	0.752±0.120	0.695±0.072	0.644±0.100	**0.782±0.104**	0.614±0.165
AR5	0.681±0.166	0.762±0.207	0.774±0.183	0.805±0.097	0.798±0.093	0.514±0.225	**0.877±0.106**	0.486±0.230
AR6	0.603±0.085	0.544±0.097	0.550±0.093	0.543±0.090	0.629±0.076	0.588±0.111	**0.642±0.100**	0.595±0.145
Apache	0.656±0.046	0.629±0.110	0.664±0.059	0.670±0.036	0.604±0.074	0.566±0.068	0.714±0.114	**0.761±0.086**
Safe	0.663±0.090	0.641±0.156	0.670±0.141	0.699±0.076	0.690±0.079	0.580±0.128	**0.770±0.140**	0.563±0.201
ZXing	0.568±0.030	0.563±0.054	0.569±0.055	0.581±0.029	0.572±0.038	0.597±0.048	0.622±0.077	**0.759±0.103**
EQ	0.690±0.022	0.572±0.138	0.577±0.131	0.642±0.045	0.632±0.074	0.645±0.084	0.675±0.150	**0.830±0.218**
JDT	0.741±0.019	0.532±0.191	0.640±0.140	0.699±0.040	0.671±0.079	0.716±0.062	0.665±0.156	**0.754±0.157**
LC	0.694±0.029	0.526±0.116	0.586±0.087	0.606±0.060	0.645±0.079	0.707±0.065	0.614±0.085	**0.871±0.224**
ML	0.656±0.023	0.531±0.134	0.580±0.099	0.642±0.027	0.599±0.048	0.650±0.033	0.637±0.075	**0.738±0.131**
PDE	0.644±0.021	0.603±0.127	0.613±0.104	0.658±0.030	0.616±0.049	0.654±0.052	0.659±0.113	**0.729±0.164**
ant-1.7	0.702±0.030	0.574±0.219	0.622±0.195	0.730±0.042	0.648±0.066	0.647±0.048	0.694±0.175	**0.745±0.196**
camel-1.6	0.594±0.028	0.526±0.076	0.533±0.074	0.580±0.022	0.552±0.031	0.595±0.043	0.579±0.057	**0.729±0.158**
ivy-2.0	0.662±0.061	0.563±0.250	0.680±0.190	0.756±0.057	0.621±0.083	0.684±0.072	0.683±0.197	**0.866±0.132**
jedit-4.0	0.676±0.041	0.487±0.179	0.650±0.116	0.676±0.058	0.609±0.063	0.626±0.068	0.648±0.132	**0.798±0.169**
log4j-1.0	0.708±0.066	0.645±0.162	0.664±0.145	0.708±0.059	0.671±0.089	0.645±0.082	0.702±0.142	**0.749±0.139**
lucene-2.4	0.627±0.046	0.514±0.105	0.579±0.085	0.602±0.037	0.611±0.046	0.592±0.059	0.610±0.089	**0.712±0.160**
poi-3.0	**0.710±0.040**	0.506±0.125	0.579±0.105	0.614±0.052	0.664±0.059	0.506±0.063	0.664±0.150	0.685±0.121
synapse-1.2	0.653±0.047	0.616±0.151	0.641±0.128	0.683±0.054	0.629±0.062	0.632±0.065	0.677±0.103	**0.790±0.178**
tomcat-6.0	0.680±0.049	0.566±0.238	0.654±0.197	0.744±0.053	0.659±0.060	0.719±0.047	0.690±0.170	**0.844±0.185**
velocity-1.6	0.647±0.045	0.497±0.090	0.580±0.061	0.586±0.040	0.540±0.073	0.552±0.066	0.602±0.133	**0.761±0.131**
xalan-2.4	0.653±0.039	0.554±0.200	0.629±0.159	0.702±0.044	0.607±0.063	0.667±0.048	0.650±0.156	**0.816±0.225**
xerces-1.3	0.692±0.050	0.501±0.142	0.606±0.099	0.638±0.052	0.617±0.051	0.685±0.070	0.649±0.100	**0.804±0.183**
Mean	0.654±0.078	0.584±0.164	0.626±0.136	0.659±0.091	0.637±0.088	0.639±0.106	0.680±0.142	**0.744±0.196**
Median	0.679	0.628	0.650	0.658	0.634	0.645	0.700	**0.799**
w/t/l	24/2/4	26/0/4	26/0/4	26/0/4	25/0/5	25/1/4	23/1/6	-
Cliff's δ	0.397 (M)	0.506 (L)	0.438 (M)	0.390 (M)	0.443 (M)	0.440 (M)	0.291 (S)	-

based on a Cliff's δ, we can estimate the magnitude of the effect size. For example, the Cliff's δ between CCA+ and EMKCA is 0.440 and its magnitude is *medium*. That is, the prediction performance of EMKCA is much better than CCA+ with practical significance in our experimental settings. The analysis between other baseline and EMKCA is similar to the above in terms of the Cliff's effect size test. In summary, EMKCA outperforms the baseline methods with significant differences.

4) Comparison of EMKCA and CCT-SVM Using SVM Classifier: CCT-SVM [24] is only applied to the SVM classifiers, and cannot be applied to other classifiers directly. To be fair, we compare EMKCA to CCT-SVM using the SVM classifier. To evaluate the prediction performance of EMKCA and CCT-SVM using the SVM classifier, we separately conduct experiment for these two methods. The mean *AUC* results of EMKCA and CCT-SVM with the SVM classifier across 30 target projects are 0.694 and 0.618, respectively. In terms of the overall mean performance, EMKCA improves the mean *AUC* by 12.30% over CCT-SVM. In short, EMKCA overall outperforms CCT-SVM in our experimental settings. The reason is that CCT-SVM does not address the linearly inseparable problem. Instead, EMKCA aims to learn nonlinear correlation using multiple kernel technique, which can tackle this problem by transforming data to high dimensional space. Besides, we find the performance of EMKCA with LR outperforms SVM. This is consistent with the work described in [53, 55], which show that SVM was one of the lowest ranked classifiers in their empirical study for defect prediction.

5) Summary: Answers to Research Questions: For RQ1, the proposed EMKCA approach can achieve better or comparable results as compared with WPDP. For RQ2, EMKCA performs better than the CPDP methods that use common metrics. For RQ3, EMKCA overall outperforms the competing HDP methods. The statistical significance test and the effect size test also validate our conclusions.

V. DISCUSSION

A. Effect of Different Setting of r

Recent work [62] has shown that the suboptimal parameter settings have a large impact on the performance of defect prediction models. In this section, to investigate the influence of parameter r used in ICD on EMKCA, we change the value of r in the range of $\{20, 30, 40, 50, 60, 70, 80, 90, 100\}$ to report the results across multiple runs. To this end, we

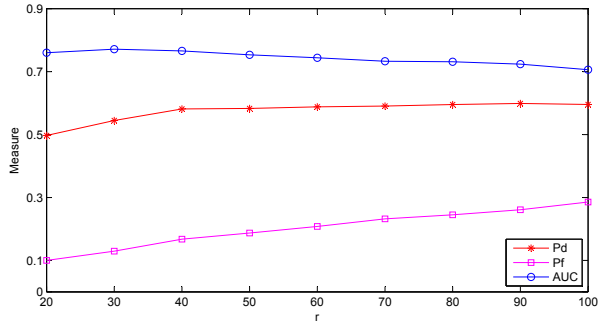

Figure 2. The *Pd*, *Pf* and *AUC* of EMKCA using different values of r.

report the overall mean *AUC* across 30 target projects as well as *Pd* (probability of detection or recall) and *Pf* (probability of false alarm) measures. These two measures are widely used in the defect prediction studies [6, 13, 8, 21, 9]. Figure 2 shows the mean *Pd*, *Pf* and *AUC* results achieved by EMKCA on various r values across 30 target projects. We can observe that the *AUC* results of EMKCA are gradually declined, while the *Pd* and *Pf* are gradually improved with the increasing of r from 20 to 100. Taking into account all aspects, it is feasible to set r to 60.

B. The Visualization Results

Kernel-based methods have been theoretically and empirically proven to be able to tackle the linearly inseparable problem by transforming data from its original feature space into a high dimensional kernel space [25, 26]. In this paper, EMKCA maps historical defect data to the kernel space. To graphically visualize the feature distribution of defect data, we provide the visualization results of the original feature space and the kernel feature space. Here, we take CM1 project of NASA group as an example. Figure 3 shows the feature distribution of the CM1 project, where the first two principle features of each projected features are used to illustrate these distributions. The principal component analysis (PCA) [63] is used to extract two principle features for showing the distribution in a two-dimensional space. As shown in Figure 3, defective and non-defective instances are entangled, and the distributions is relatively scattered in the original space. In the kernel space, the features of defective and non-defective instances are more likely to be learned. The separability in the kernel space is better than it in the original space.

C. Effect of Ensemble Learning

In this section, to examine the effect of ensemble learning, we evaluate our EMKCA approach with and without ensemble. We call the version of EMKCA without ensemble KCA. Because we use 10 different kernel functions (i.e., 9 Gaussian kernels and one linear kernel), there are 10 different results of KCA. Figure 4 shows the boxplot of *AUC*

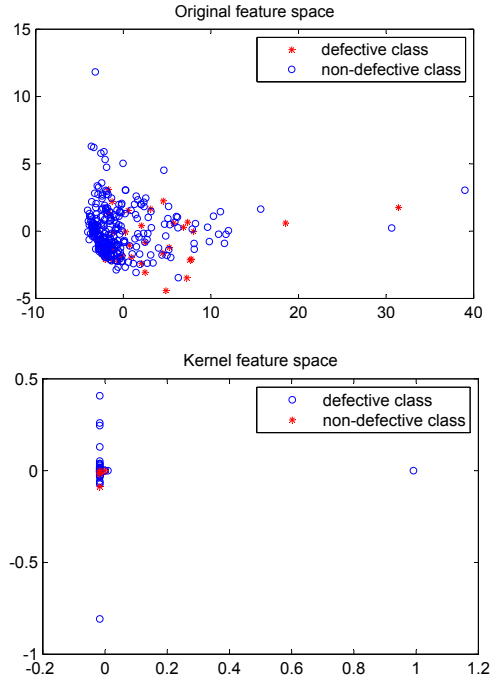

Figure 3. Feature distribution of CM1, where the first two principle features in original space and kernel space are used to illustrate these distributions.

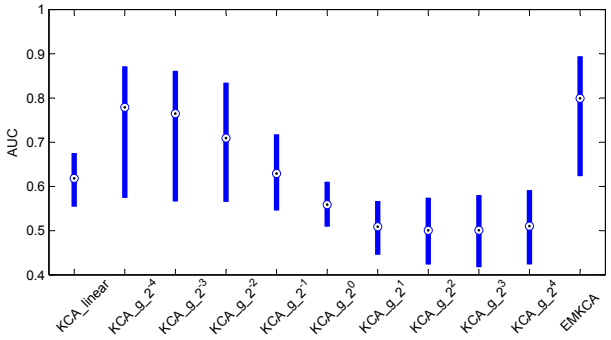

Figure 4. The boxplot of *AUC* across 30 projects for KCA/EMKCA with different kernels.

across 30 projects for KCA/EMKCA with different kernel functions. The bar indicates the first-third quartile range and the circle denotes the median. The minimal and maximal values are not displayed. Note that "KCA_g_2^3" denotes the KCA which uses Gaussian kernel with kernel parameter 2^3, "KCA_linear" denotes KCA which uses linear kernel, and EMKCA denotes assembling all kernels. From this figure, we can see that the *AUC* of EMKCA is higher than that of each KCA. The results demonstrate that the ensemble classifier generally outperforms each sub-classifier. We also observe that different kernel parameters have large impact on the prediction performance of KCA. Therefore, it is important to select appropriate kernel parameter.

D. Practical Guidelines for HDP

Prediction of software defects works well within the same project when sufficient historical defect data is available for training the prediction models [12, 13]. Therefore, for a target project, if there is sufficient historical data at hand, developers can employ WPDP to conduct defect prediction. However, there might not be enough historical data for a new software project or a new company. In practice, there usually exists plenty of data from other projects. If these external data has the same metric sets with the target project, developers can conduct CPDP by using the recently proposed CPDP methods [16, 35, 36, 37, 18, 38]. The above CPDP methods may not be feasible when the metric sets between the target and external projects are heterogeneous. In this scenario, the recently proposed HDP methods [21, 22] can be employed. In order to get more accurate predictions, developers can apply our proposed EMKCA approach.

In Section IV-F, we conduct extensive and large-scale experiments across 30 projects with heterogeneous metric sets from five groups. Experimental results show that EMKCA outperforms the typical CPDP methods (NN-filter, VCB and SSTCA+ISDA), the HDP methods (CPDP-IFS, CCA+, HDP-KS and CCT-SVM) and WPDP. The non-parametric Mann-Whitney U statistical test and Cliff's delta effect size test also validate this conclusion. Therefore, we suggest using EMKCA for projects lacking sufficient historical data to learn defect predictors.

E. Threats to Validity

Recent research points out that experimental design may impact the conclusions of the paper [64, 65]. In this paper, we have identified several potential threats to the validity about our empirical study.

Threats to Construct Validity. We used the Euclidean distance based similarity measure to calculate the weight for linear combination. Using other similarity techniques we might obtain different prediction performance. Besides, we only used LR and SVM classifiers to conduct experiments, the performance of EMKCA is unknown for other classifiers [51, 53]. In the future, we will employ other ensemble techniques and classifiers to validate our proposed EMKCA approach. Our results rely on two-fold cross validation. A recent study [66] points out that model validation techniques may produce different bias and variances of performance estimates. Thus, the conclusions may differ when applying other cross validation techniques.

Threats to Internal Validity. We used the *AUC* metric to evaluate the prediction performance, which has been widely used to evaluate the effectiveness of defect prediction [51, 53, 22, 54, 37]. Measuring prediction performance of other evaluation measures (e.g., G-mean, Matthews correlation coefficient) is left for future work. With respect to the related compared methods, we carefully reimplemented these methods by following the corresponding papers. However, our implementation may not be exactly the same as the original papers, leading to possible bias in the comparison between our approach and the baselines.

Threats to External Validity. Researchers should experiment with a broader selection of datasets and metrics in order to maximize external validity [67]. In this paper, we chose 30 projects from five groups that are widely used in papers published in top software engineering venues [49, 21, 6, 16, 22, 13, 48]. These projects come from both proprietary (NASA and SOFTLAB) and open-source (ReLink, AEEEM and PROMISE) projects. Therefore, our findings might not be generalizable to other closed software projects. In the future, we will reduce this threat by conducting further experiments on more defect data from open source and commercial software projects.

VI. Conclusion

Recently, heterogeneous defect prediction (HDP) has received much research interest and provides a new perspective to defect prediction. In this paper, we propose a novel ensemble multiple kernel correlation alignment (EMKCA) based approach to HDP. Extensive experiments are conducted on 30 publicly available projects from five groups. The non-parametric Mann-Whitney U statistical test and Cliff's delta effect size test are employed for the evaluation. Experimental results demonstrate the efficacy of EMKCA.

For the future work, we would like to use more software project data that contains both open source and commercial proprietary projects to validate the generalization ability of EMKCA. We will also examine the performance of EMKCA for the source and target projects with the same metric sets. In addition, we will try to address the HDP problem with other popular techniques, such as deep learning [33, 61]. We provide the MATLAB code that is used to conduct this study at *https://sites.google.com/site/enmkca/*.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions. This work was supported by the National Key Research and Development Program of China under Grant No. 2017YFB0202001, the National Nature Science Foundation of China under Project Nos. 61272273, 61672208, the Program of State Key Laboratory of Software Engineering under Grant No. SKLSE-1216-14, the Science and Technology Program in Henan province under Grant No. 1721102410064, and the Province-School-Region Project of Henan University under Grant No. 2016S11.

References

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2012.

[2] S. Kim, E. J. Whitehead, and Y. Zhang, "Classifying software changes: Clean or buggy?" *IEEE Transactions on Software Engineering*, vol. 34, no. 2, pp. 181–196, 2008.

[3] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann, "Local versus global lessons for defect prediction and effort estimation," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 822–834, 2013.

[4] S. Shivaji, E. J. Whitehead, R. Akella, and S. Kim, "Reducing features to improve code change-based bug prediction," *IEEE Transactions on Software Engineering*, vol. 39, no. 4, pp. 552–569, 2013.

[5] T. Lee, J. Nam, D. Han, S. Kim, and H. In, "Developer micro interaction metrics for software defect prediction," *IEEE Transactions on Software Engineering*, vol. 42, no. 11, pp. 1015–1035, 2016.

[6] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, 2007.

[7] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A large-scale empirical study of just-in-time quality assurance," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 757–773, 2013.

[8] X.-Y. Jing, S. Ying, Z.-W. Zhang, S.-S. Wu, and J. Liu, "Dictionary learning based software defect prediction," in *ICSE*, 2014, pp. 414–423.

[9] T. Wang, Z. Zhang, X.-Y. Jing, and L. Zhang, "Multiple kernel ensemble learning for software defect prediction," *Automated Software Engineering*, vol. 23, no. 4, pp. 569–590, 2016.

[10] Z. Zhang, X.-Y. Jing, and T. Wang, "Label propagation based semi-supervised learning for software defect prediction," *Automated Software Engineering*, vol. 24, no. 1, pp. 47–69, 2017.

[11] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in *ICSE*, 2011, pp. 481–490.

[12] 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," in *ESEC/FSE*, 2009, pp. 91–100.

[13] 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," *Empirical Software Engineering*, vol. 14, no. 5, pp. 540–578, 2009.

[14] Z. He, F. Shu, Y. Yang, M. 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.

[15] M. Li, H. Zhang, R. Wu, and Z. H. Zhou, "Sample-based software defect prediction with active and semi-supervised learning," *Automated Software Engineering*, vol. 19, no. 2, pp. 201–230, 2012.

[16] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *ICSE*, 2013, pp. 382–391.

[17] B. Turhan, A. T. Mısırlı, and A. Bener, "Empirical evaluation of the effects of mixed project data on learning defect predictors," *Information and Software Technology*, vol. 55, no. 6, pp. 1101–1118, 2013.

[18] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang, "Hydra: Massively compositional model for cross-project defect prediction," *IEEE Transactions on Software Engineering*, vol. 42, no. 10, pp. 977–998, 2016.

[19] Y. Kamei, T. Fukushima, S. McIntosh, K. Yamashita, N. Ubayashi, and A. E. Hassan, "Studying just-in-time defect prediction using cross-project models," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2072–2106, 2016.

[20] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model with rank transformed predictors," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2107–2145, 2016.

[21] X.-Y. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and cca-based transfer learning," in *FSE*, 2015, pp. 496–507.

[22] J. Nam and S. Kim, "Heterogeneous defect prediction," in *FSE*, 2015, pp. 508–519.

[23] P. He, B. Li, and Y. Ma, "Towards cross-project defect prediction with imbalanced feature sets," *CoRR*, vol. abs/1411.4228, 2014.

[24] M. Cheng, G. Wu, M. Jiang, H. Wan, G. You, and M. Yuan, "Heterogeneous defect prediction via exploiting correlation subspace," in *SEKE*, 2016, pp. 171–176.

[25] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, "Use of the zero norm with linear models and kernel methods," *The Journal of Machine Learning Research*, vol. 3, pp. 1439–1461, 2003.

[26] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *CVPR*, 2012, pp. 2074–2081.

[27] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[28] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 434–443, 2013.

[29] M. Tan, L. Tan, S. Dara, and C. Mayeux, "Online defect prediction for imbalanced data," in *ICSE*, 2015, pp. 99–108.

[30] H. Zhang and X. Zhang, "Comments on "data mining static code attributes to learn defect predictors"," *IEEE Transactions on Software Engineering*, vol. 33, no. 9, pp. 635–637, 2007.

[31] X.-Y. Liu and Z.-H. Zhou, *Ensemble Methods for Class Imbalance Learning*. John Wiley and Sons, Inc., 2013.

[32] F. Peters, T. Menzies, L. Gong, and H. Zhang, "Balancing privacy and utility in cross-company defect prediction," *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1054–1068, 2013.

[33] S. Wang, T. Liu, and L. Tan, "Automatically learning semantic features for defect prediction," in *ICSE*, 2016, pp. 297–308.

[34] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Information and Software Technology*, vol. 54, no. 3, pp. 248–256, 2012.

[35] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Negative samples reduction in cross-company software defects prediction," *Information and Software Technology*, vol. 62, pp. 67–77, 2015.

[36] G. Canfora, A. D. Lucia, M. D. Penta, R. Oliveto, A. Panichella, and S. Panichella, "Defect prediction as a multiobjective optimization problem," *Software Testing, Verification and Reliability*, vol. 25, no. 4, pp. 426–459, 2015.

[37] D. Ryu, O. Choi, and J. Baik, "Value-cognitive boosting with a support vector machine for cross-project defect prediction," *Empirical Software Engineering*, vol. 21, no. 1, pp. 43–71, 2016.

[38] 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 Transactions on Software Engineering*, vol. 43, no. 4, pp. 321–338, 2017.

[39] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the nasa software defect datasets," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1208–1215, 2013.

[40] H. Zhang, "An investigation of the relationships between lines of code and defects," in *ICSM*, 2009, pp. 274–283.

[41] H. Zhang and H. B. K. Tan, "An empirical study of class sizes for large java systems," in *ASPEC*, 2007, pp. 230–237.

[42] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[43] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *AAAI*, 2016, pp. 2058–2065.

[44] M. Gnen and E. Alpaydn, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.

[45] M. Ying, L. Guangchun, and C. Hao, "Kernel based asymmetric learning for software defect prediction," *IEICE Transactions on information and systems*, vol. 95, no. 1, pp. 267–270, 2012.

[46] S. v. Vaerenbergh, *Kernel methods for nonlinear identification, equal-*

ization and separation of signals. Universidad de Cantabria, 2010.

[47] Y.-R. Yeh, C.-H. Huang, and Y.-C. F. Wang, "Heterogeneous domain adaptation and classification by exploiting the correlation subspace," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2009–2018, 2014.

[48] R. Wu, H. Zhang, S. Kim, and S.-C. Cheung, "Relink: recovering links between bugs and changes," in *ESEC/FSE*, 2011, pp. 15–25.

[49] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," *Empirical Software Engineering*, vol. 17, no. 4-5, pp. 531–577, 2012.

[50] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *PROMISE*, 2010, pp. 1–10.

[51] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, 2008.

[52] F. Rahman, D. Posnett, and P. Devanbu, "Recalling the imprecision of cross-project defect prediction," in *ESEC/FSE*, 2012, pp. 1–11.

[53] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *ICSE*, 2015, pp. 789–800.

[54] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, "Cross-project defect prediction using a connectivity-based unsupervised classifier," in *ICSE*, 2016, pp. 309–320.

[55] J. Nam and S. Kim, "Clami: Defect prediction on unlabeled datasets," in *ASE*, 2015, pp. 1–12.

[56] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, no. 9, pp. 1871–1874, 2008.

[57] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, "Defect prediction from static code features: current results, limitations, new approaches," *Automated Software Engineering*, vol. 17, no. 4, pp. 375–407, 2010.

[58] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006.

[59] H. Zhang, A. Nelson, and T. Menzies, "On the value of learning from defect dense components for software defect prediction," in *PROMISE*, 2010, pp. 1–9.

[60] J. Romano and J. D. Kromrey, "Appropriate statistics for ordinal level data: Should we really be using t-test and cohen's d for evaluating group differences on the nsse and other surveys?" in *Annual meeting of the Florida Association of Institutional Research*, 2006.

[61] X. Yang, D. Lo, X. Xia, and Y. Zhang, "Deep learning for just-in-time defect prediction," in *QRS*, 2015, pp. 17–26.

[62] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," in *ICSE*, 2016, pp. 321–332.

[63] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[64] C. Tantithamthavorn, "Towards a better understanding of the impact of experimental components on defect prediction modelling," in *International Conference on Software Engineering Companion*, 2016, pp. 867–870.

[65] T. Menzies and M. Shepperd, "Special issue on repeatable results in software engineering prediction," *Empirical Software Engineering*, vol. 17, no. 1, pp. 1–17, 2012.

[66] C. Tantithamthavorn, S. Mcintosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, 2017.

[67] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "Comments on "researcher bias: The use of machine learning in software defect prediction," *IEEE Transactions on Software Engineering*, vol. 42, no. 11, pp. 1092–1094, 2016.