LETTER

Kernel CCA Based Transfer Learning for Software Defect Prediction

Ying MA[†a)], **Shunzhi ZHU**[†], **Yumin CHEN**[†], ***Nonmembers***, ***and*** **Jingjing LI**[††], ***Member***

SUMMARY An transfer learning method, called Kernel Canonical Correlation Analysis plus (KCCA+), is proposed for heterogeneous Cross-company defect prediction. Combining the kernel method and transfer learning techniques, this method improves the performance of the predictor with more adaptive ability in nonlinearly separable scenarios. Experiments validate its effectiveness.

key words: *machine learning, defect prediction, transfer learning, kernel canonical correlation analysis*

1. Introduction

Software defect prediction should identify defect-prone modules to improve software quality and testing efficiency. Many researchers apply software defect prediction on the basic assumptions, the source and target data must be in the same feature and must follow the identically distributed condition [1], [2]. In practice, the local software defect data sets are not available in many circumstances [3]–[5], such as a new project is started in a new domain, the defect prediction technique is applied for the first time, and defect data might not be collected in the software development process. At the same time, the auxiliary software defect data sets are available from open source software, which has become increasingly prevalent for software development.

To the best of our knowledge, few researchers have investigated the kernel based technology to support the defect prediction with auxiliary heterogeneous sources. Transfer learning [6] provides an approach to construct reliable classifiers through exploiting the useful knowledge from related auxiliary data sets. In order to improve the performance and enhance the adaptive ability of the predictor, we propose a method called Kernel Canonical Correlation Analysis based transfer learning algorithm (KCCA+), combining the kernel method with transfer learning technique in generalized way. The main feature of KCCA+ is transforming the source and target software data into high space to find the most appropriate weight matrix for learning, as shown in Fig. 1. The data in source domains can be represented with the mapping functions and transferred to the target domain.

Our experimental results on real data sets show that KCCA+ results in better performance than state of the art prediction methods. Since local labeled data sets are rare and auxiliary labeled data sets are rich, this method is helpful for practical application.

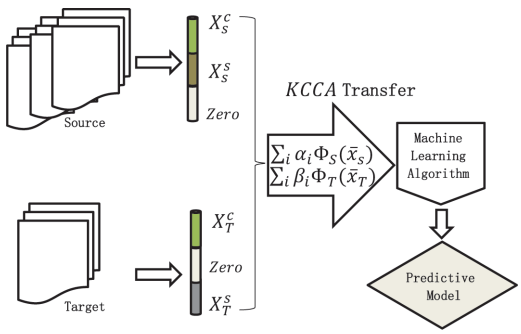

Fig. 1 Construct defect prediction model based on KCCA+ method.

2. Related Work

There are a few software defect prediction studies which exploit heterogeneous cross-company data sets in the predictors. These methods can be divided into two groups: instance-based approach and algorithm-based approach.

In the instance-based approach, Turhan et al. [7] built local predictor based on the similar cross-company samples which are select by nearest neighbor filtering (NN-filter). This method has better performance than original methods which are trained on cross-company data diretly. In comparison to Turhan's method, Zimmermann et al. [8] analyzed the effect of the various characteristics on prediction quality with decision trees. He et al. [9] proposed a three-step approach to automatically select training data for projects without historical data. After that, Turhan et al. [10] constructed models from a mix of within and cross project data, and found some improvements on within project defect predictions after adding data from other projects. Devine et al. [11] empirically investigated the affect of reuse across products and reuse across releases on accuracy of defect predictors. These articles are all applying traditional leaning methods on different data, which were preprocessed by different strategies.

One of the first reports for the algorithm-based approach is Transfer Naive Bayes (TNB) [12], which estimated the distribution of the test data, and transfered cross-company data information into the weights of the training data. Nam et al. [13] extended the Transfer Component

Manuscript received December 9, 2016.
Manuscript revised April 7, 2017.
Manuscript publicized April 28, 2017.
[†]The authors are with Xiamen University of Technology, Xiamen, China.
[††]The author is with University of Electronic Science and Technology of China, Chengdu, China.
a) E-mail: maying@xmut.edu.cn
DOI: 10.1587/transinf.2016EDL8238

Analysis (TCA) to improve cross-project prediction performance. Chen et al. [14] proposed a novel algorithm based on double boosting to improve the performance of cross-company defects prediction by reducing negative samples in cross-company data. Ryu et al. [15] proposed method called the value-cognitive boosting with support vector machine to combat the class imbalance problem in cross-project defect prediction. Jing et al. [16] introduced canonical correlation analysis (CCA), an effective transfer learning method, into cross-company defect prediction to make the data distributions of source and target companies similar.

The most approaches are based on the assumption that the data of source and target companies or projects should have the same software metrics. KCCA+ builds up a mapping between two related feature spaces to enable the knowledge transfer based on kernel technique, as shown in Fig. 1. The data in source companies can be represented with the mapping functions and transferred to the target companies. Therefore, it improves the performance of the predictor with more adaptive ability in nonlinearly separable scenarios, without the same software metrics assumption.

3. Kernel Canonical Correlation Analysis Based Transfer Learning for Software Defect Prediction

3.1 Back Ground

Canonical Correlation Analysis (CCA) [17] is well-known multivariate data analysis technique to seek matrixes W_S and W_T such that the random variables $U = W_S' X_S$ and $V = W_T' X_T$ maximize the correlation, where $X_S = x_S^1, x_S^2, \ldots, x_S^N$ and $X_T = x_T^1, x_T^2, \ldots, x_T^M$, $(.)'$ refers to the transpose of a vector or a matrix. Recently, CCA has been used in transfer learning for heterogeneous Cross-company defect prediction [16]. The projective transformation W_S and W_T based on the unified software metric are as follows.

$$\bar{X}_S = \begin{bmatrix} X_S^C \\ X_S^s \\ 0_{(d_t - d_c)} \times N \end{bmatrix} \text{ and } \bar{X}_T = \begin{bmatrix} X_T^C \\ 0_{(d_s - d_c)} \times M \\ X_T^s \end{bmatrix} \tag{1}$$

where X_S^C and X_T^C correspond to the same common metrics, X_S^s is the data in X_S containing source-company metrics except for the common metrics and X_T^s is the data in X_T containing target-company specific metrics.

Table 1 Symbols of the kernel canonical correlation analysis.

symbol	description
d_s, d_t	the number of metrics in source data, target data
N, M	the number of samples in source data, target data
$d_u = d_s + d_t - d_c$	the number of unified metrics
$d_u \uparrow$	the d_u in the transformed high feature space
$\bar{X}_S, \bar{X}_T \in \mathbb{R}^{d_u \times d_u}$	the unified metrics
$X_S \in \mathbb{R}^{d_s \times N}, X_T \in \mathbb{R}^{d_t \times M}$	the source and target matrix
$K_S, K_T \in \mathbb{R}^{d_u \uparrow \times d_u \uparrow}$	the source and target covariance matrix
$\Phi_S, \Phi_T \in \mathbb{R}^{d_u \uparrow \times d_u \uparrow}$	the transformation matrix
$\alpha, \beta \in \mathbb{R}^{d_u \uparrow \times d_u \uparrow}$	the weight matrix in the transformed space
$\bar{\mu}_S, \bar{\mu}_T$	mean vector of source and target unified data
$U, V \in \mathbb{R}^{d_u \times d_u}$	the transformed source and target matrix

3.2 Kernel Canonical Correlation Analysis Based Transfer Learning for Software Defect Prediction

The canonical correlation analysis method can find an appropriate representation of the source and target software data in a linear subspace. The CCA+ transfer model considers the defect data linearly separable. In order to achieve reliable results, we exploit the kernel CCA [18] to generalize the transfer methods for the software data sets which are non-linearly separable. By using the kernel method[†], the U and V can be rewritten as

$$U = C' \Phi_S(x_S) = \Sigma_i \alpha_i \Phi_S(\bar{x}_S) \tag{2}$$

$$V = D' \Phi_T(x_T) = \Sigma_i \beta_i \Phi_T(\bar{x}_T) \tag{3}$$

where the C, D are the weight matrix in the transformed high feature space, and α, β are the weight matrix, which are arranged by α_i, and β_i respectively in the original feature space. Then the correlation of U and V is

$$corr(U, V) = \frac{\alpha K_S K_T \beta}{\sqrt{\alpha' K_S^2 \alpha} \sqrt{\beta' K_T^2 \beta}} \tag{4}$$

where the source covariance matrices and the target covariance matrix in the kernel form are as follows.

$$K_S = \frac{1}{N} \sum_{j=1}^{N} (\Phi(\bar{x}_S^j) - \bar{u_S})(\Phi(\bar{x}_S^j) - \bar{u_S})' \tag{5}$$

$$K_T = \frac{1}{M} \sum_{j=1}^{M} (\Phi(\bar{x}_T^j) - \bar{u_T})(\Phi(\bar{x}_T^j) - \bar{u_T})' \tag{6}$$

where $\bar{u_i} = \frac{1}{n_i} \sum_{j=1}^{n_i} \Phi(x_i^j)$ is conditional mean vector, $\bar{u}$ is mean vector of total instances, $\Phi(x_i^j)$ is the jth instances in the source or target data set, $i \in \{S, T\}$, and n_i is the number of instances of S or T.

In order to maximize the $corr(U, V)$ value, we get the kernelized objective function:

$$\begin{aligned} &Max_{\alpha,\beta} \quad \alpha' K_S K_T \beta \\ &s.t. \quad \alpha' K_S \alpha = 1,\ \beta' K_T \beta = 1 \end{aligned} \tag{7}$$

The above problem can be modified to the following generalized eigenvalue problem.

$$\begin{bmatrix} 0 & K_S K_T \\ K_T K_S & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \lambda \begin{bmatrix} K_S^2 & 0 \\ 0 & K_T^2 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{8}$$

Note that CCA+ and KCCA+ is modified through the original represent of CCA and KCCA, which are prone to

[†]Kernel function ($\Phi(x)$) is introduced to reduce computation, for mapping the data nonlinearly into a feature space.

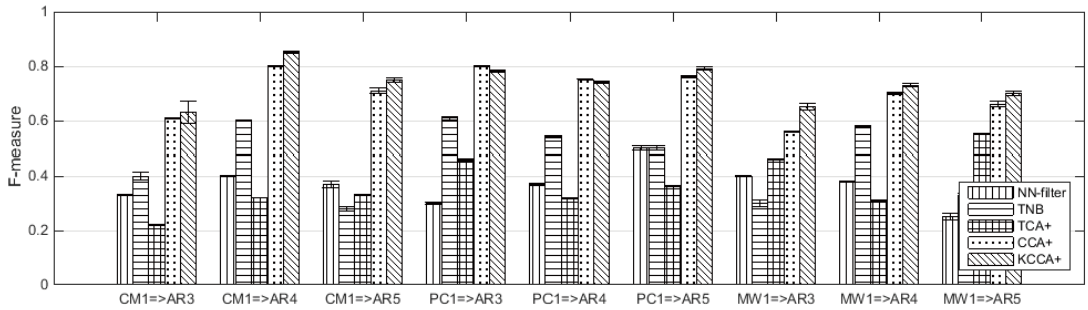

Fig. 2 The *F-measure* results of NN-filter, TNB, TCA+, CCA+, and KCCA+.

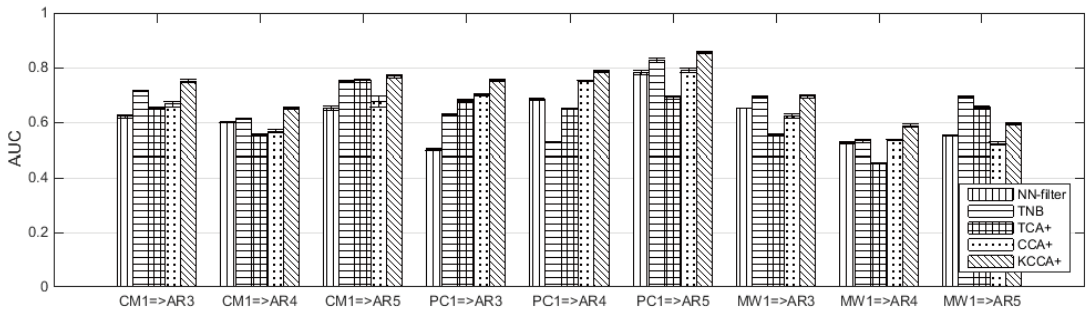

Fig. 3 The *AUC* results of NN-filter, TNB, TCA+, CCA+, and KCCA+.

occur overfitting problem. To limit overfitting, the regularized version should be considered as follows.

$$\begin{aligned} Max_{\alpha,\beta} \quad & \alpha' K_S K_T \beta \\ s.t. \quad & (1-\tau_\alpha)\alpha' K_S^2 \alpha + \tau_\alpha \alpha' K_S \alpha = 1 \\ & (1-\tau_\beta)\beta' K_T^2 \beta + \tau_\beta \beta' K_T \beta = 1 \end{aligned} \tag{9}$$

To solve the above optimal problem, lagrange multiplier method can be used to obtain generalized eigenvalue problem.

$$\begin{aligned} &\begin{bmatrix} 0 & K_S K_T \\ K_T K_S & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\ &= \lambda \begin{bmatrix} (1-\tau_\alpha)K_S^2 + \tau_\alpha K_S & 0 \\ 0 & (1-\tau_\beta)K_T^2 + \tau_\beta K_T \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \end{aligned} \tag{10}$$

The above problem can be modified to the following equivalent symmetry eigenvalue problem.

$$(R')^{-1} R_\beta R'_\alpha ((1-\tau_\alpha)R_\alpha R'_\alpha + \tau_\alpha I)^{-1} R_\alpha R'_\beta R^{-1} u = \lambda u \tag{11}$$

where $K_S = R'_\alpha R_\alpha$, $K_T = R'_\beta R_\beta$, and $(1-\tau_\beta)R_\beta R'_\beta + \tau_\beta I = R'R$ are the incomplete Cholesky factorization of the kernel matrices.

KCCA+ is summarized as in Algorithm 1. After obtaining the projected samples, we use the naive bayes classifier [1] to build predictor. KCCA+ originates from the need to transfer the target data information in the prediction task. This method inherits the advantage of kernel method, which can conduct quite general dimensional feature space mappings. This algorithm maximizes the correlations between transformed variables based on KCCA, and solves the different distributions between the source and target data.

Algorithm 1 KCCA+

Require:
Source and Target data sets S, T;
Ensure:
Kernel Canonical Correlation Analysis based predictor, M;
1: Run Unifier model algorithm on data set S and T to get K_S, K_T.
2: Decompose K_S, K_T to get R_α, R_β;
3: Decompose $(1-\tau_\beta)R_\beta R'_\beta + \tau_\beta I$ to get R;
4: Compute λ_i, u_β^j, according to Eq. (11);
5: Calculate $\beta^j = R^{-1}u_\beta$, $\beta^j = \|\beta^j\|$;
6: Calculate $\alpha^j = ((1-\tau_\alpha)R_\alpha R'_\alpha + \tau_\alpha I)^{-1} R_\alpha R'_\beta \beta^j$, $\alpha^j = \alpha^j / \|\alpha^j\|$;
7: Compute the source and target projected samples U, V, according to Eqs. (2) and (3);
8: Build predictor M on the projected samples;
9: Using M to Predict the result based on T corresponding to the V.
10: **return** M;

4. Experiments

4.1 Data Sets

The experimental data sets come from NASA and SOFTLAB, which can be obtained from PROMISE [19], as shown in Table 2. The SOFTLAB data sets (ar3, ar4, ar5), are drawn from three controller systems for a washing machine, a dishwasher, and a refrigerator in Turkish domestic appliances company respectively. They are all written in C code. Remaining data sets are developed at different sites by different teams from NASA aerospace software compa-

Table 2 Data sets.

Company	project	#metrics	#modules	size(loc)	#defective
NASA	CM1	37	327	14,763	42
	PC1	37	705	25,924	65
	MW1	37	253	8341	27
SOFTLAB	AR3	29	63	5,624	8
	AR4	29	107	9,196	20
	AR5	29	36	2,732	8

nies. They are all written in C++ code.

4.2 Experimental Result

In order to investigate the performance of our algorithm, we compare it with NN-filter [7], TNB [12], TCA+ [13], CCA+ [16] (Gaussian RBF kernel $\gamma = 10$, $c = 1000$). For each data set, we perform 5-fold cross validation. We use the area under the receiver operating characteristic curve (AUC) and the weighted harmonic mean of precision and recall (F-$measure$) performance metrics which are commonly used in the field of software defect prediction.

The results for all the five methods are shown in Figs. 2 and 3. We can see that KCCA+ is superior to other algorithms in the aspect of F-$measure$ except for (PC1⇒AR3, PC1⇒AR4). But even in these two cases, KCCA+ is also comparable to CCA+. KCCA+ is also superior to other algorithms in the aspect of AUC, except for (MW1⇒AR5). Note that the number of software modules in MW1 is the smallest, and AR5 is also the smallest data set in the target data. It may be because the quality of the prediction not only depends on promising method, but also depends on the absolute number of labeled instances. Both of the figures show that KCCA+ learning algorithm can improve the F-$measure$ and AUC performance by combining the kernel method and transfer learning technique.

5. Conclusion

In this paper, we introduce the kernel canonical correlation analysis based transfer learning method into software defect prediction. The proposed algorithm KCCA+ exploits the kernel technique and transfer learning technique to transfer the target data information for local labeling task. Experimental results on real data sets validate its effectiveness.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (Grant No. 61502404), Natural Science Foundation of Fujian Province of China (Grant No. 2015J05132), Foundation of Fujian Educational Committee (Grant No. JA14234), International S&T Cooperation Program (Grant Nos. E201402000). We thank the anonymous reviewers for their great helpful comments.

References

[1] 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, 2007.
[2] C. Catal, "Software fault prediction: A literature review and current trends," Expert Systems with Applications: An International Journal, vol.38, no.4, pp.4626–4636, 2011.
[3] T. Menzies, E. Kocaguneli, F. Peters, B. Turhan, and L.L. Minku, "Data science for software engineering," International Conference on Software Engineering, pp.1484–1486, 2013.
[4] Y. Kamei and E. Shihab, "Defect prediction: Accomplishments and future challenges," 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, pp.33–45, 2016.
[5] T. Menzies, L. Minku, and F. Peters, "The art and science of analyzing software data; Quantitative methods," 2015 IEEE/ACM 37th International Conference on Software Engineering, pp.959–960, 2015.
[6] S.J. Pan and Q. Yang, "A survey on transfer learning," Technical Report HKUST-CS 08-08, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, 2008.
[7] 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.
[8] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," Proc. 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), pp.91–100, 2009.
[9] 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.
[10] B. Turhan, A.T. Misirli, 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.
[11] T. Devine, K. Goseva-Popstojanova, S. Krishnan, and R.R. Lutz, "Assessment and cross-product prediction of software product line quality: Accounting for reuse across products, over multiple releases," Automated Software Engineering, vol.23, no.2, pp.253–302, 2016.
[12] 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.
[13] J. Nam, S.J. Pan, and S. Kim, "Transfer defect learning," Proc. 35th International Conference on Software Engineering (ICSE), pp.382–391, 2013.
[14] 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.
[15] D. Ryu, O. Choi, and J. Baik, "Improving prediction robustness of VAB-SVM for cross-project defect prediction," 2014 IEEE 17th International Conference on Computational Science and Engineering, pp.994–999, 2014.
[16] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning," Proc. 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), pp.496–507, 2015.
[17] D.R. Hardoon, S.R. Szedmak, and J.R. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," Neural Comput., vol.16, no.12, pp.2639–2664, 2004.
[18] K. Fukumizu, F.R. Bach, and A. Gretton, "Statistical consistency of kernel canonical correlation analysis," Journal of Machine Learning Research, vol.8, pp.361–383, 2007.
[19] G. Boetticher, T. Menzies, and T. Ostrand, The PROMISE Repository of Empirical Software Engineering Data, 2007. http://promisedata.org/repository