LETTER

Kernel Based Asymmetric Learning for Software Defect Prediction

Ying MA†, Guangchun LUO†, Nonmembers, and Hao CHEN†, Member

SUMMARY A kernel based asymmetric learning method is developed for software defect prediction. This method improves the performance of the predictor on class imbalanced data, since it is based on kernel principal component analysis. An experiment validates its effectiveness.

key words: defect prediction, class imbalance, kernel principal component analysis, machine learning

1. Introduction

Software defect prediction is to predict the defect-prone modules for the next release of software or cross project software. Real world data mining applications, including software defect prediction domain, must address the issue of learning from imbalanced data sets. As pointed out by Khoshgoftaar et al. [1] and Menzies et al. [2], the majority of defects in a software system are located in a small percentage of the program modules, software defect data sets are highly class imbalanced.

Existing approaches to solving class imbalanced problem mainly include data sampling methods and adaptive algorithm methods. The results reported in [3] show that AdaBoost almost always outperforms even the best data sampling techniques in software defect prediction. AdaBoost is a typical adaptive algorithm, and has received a good deal of attention since being introduced by Freund and Schapire [4]. It attempts to reduce the bias generated by majority class data, by updating the weights of instances dynamically according to the errors in previous learning.

 Besides these methods, some studies improved dimension reduction methods for the class imbalanced problem [5]–[7]. Most recently, Qu et al. [7] proposed an asymmetric partial least squares classifier (APLSC) to tackle the class imbalance problem. They suggested that APLSC outperform other existing algorithms, because it can extract favorable features for unbalanced classification. However it is a bilinear classifier, in which the dimension is mapped to a bilinear subspace. In this paper, we develop a kernel based asymmetric learning method, called Asymmetric Kernel Principal Component Classification (AKPCC), which is more adaptive to general situations.

2. Kernel Based Asymmetric Learning

2.1 Background

Principal Component Analysis (PCA) [8] is an effective linear transformation, which maps high-dimensional data to a lower dimensional space. Kernel Principal Component Analysis (KPCA) [9] first performs nonlinear mapping \(\Phi(x)^\top\) to transform an input vector to a higher dimensional feature space. And then linear PCA is used in this feature space.

In both algorithms, the input data are centralized in the original space and the transformed high-dimensional space, i.e. \(\sum_{i=1}^{\ell} x_i = 0\) and \(\sum_{i=1}^{\ell} \Phi(x_i) = 0\), where \(\ell\) is the number of the labeled data, and \(x_i\) is the \(i\)th instance of the data set. PCA diagonalizes the correlation matrix, \(C = \frac{1}{\ell} \sum_{i=1}^{\ell} x_i x_i^\top\), while KPCA diagonalizes the correlation matrix, \(C^\Phi = \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(x_i)\Phi(x_i)^\top\). It is equal to solving the eigenvalue problem \(AV = \Lambda V\), where \(\Lambda\) is an eigenvalue, \(V\) is a matrix of eigenvectors in KPCA. It can also be written as \(n/\Lambda \Lambda = K\alpha\), where \(K = n C^\Phi\) is the kernel matrix.

The kernel principal component regression algorithm has been proposed by R. Rosipal et al. [9]. The standard regression model in the transformed feature space can be written as

\[ f(x) = \sum_{k=1}^{p} w_k \beta(x)_k + b \]  \hspace{1cm} (1)

where \(p\) is the number of components, \(w_k\) is the \(k\)th primal regression coefficient, and \(b\) is the regression bias. \(\beta(x)_k = V_k \Phi(x)\), \(V_k\) is the \(k\)th vector of \(V\). \(V\) and \(\Lambda\) are the eigenvectors and eigenvalues of the correlation matrix respectively.

2.2 Asymmetric Kernel Principal Component Classification (AKPCC)

The KPCA regression model does not consider the correlation between principal components and the class attribution. PCA dimension reduction is inevitably affected by asymmetric distribution [7]. We analyze the effect of class imbalance on KPCA. Considering the class imbalance problem, we propose an Asymmetric Kernel Principal Component

†Corresponding author.

‡The authors are with University of electronic science and technology of China, Chengdu, China.

a) E-mail: may@uestc.edu.cn


Copyright © 2012 The Institute of Electronics, Information and Communication Engineers
Classification (AKPCC), which retrieves the losses caused by this effect.

In defect predictor, \( L = \{(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)\} \subset X \times Y \) denotes the labeled example set with size \( t \) and \( U = \{x_{t+1}, x_{t+2}, \ldots, x_{t+u}\} \subset X \) denotes the unlabeled example set with size \( u \). For labeled examples, \( Y = [+1, -1] \), the defective modules are labeled ‘+1’, the non-defective modules are labeled ‘−1’.

Suppose \( S^\Phi_w = \sum_{i=1}^{n_i} n_i(\bar{u}_i - \bar{u})(\bar{u}_i - \bar{u})' \) denotes the between-class scatter matrix, \( S^\Phi = \sum_{i=1}^{N} \sum_{j=1}^{n_i} (\Phi(x_i^j) - \bar{u}_i)(\Phi(x_i^j) - \bar{u}_i)' \) the within-class scatter matrix, where \( \bar{u}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \Phi(x_i^j) \) is class-conditional mean vector, \( \bar{u} \) is mean vector of total instances, \( \Phi(x_i^j) \) is the \( j \)-th instance in the \( i \)-th class, and \( n_i \) is the number of instances of the \( i \)-th class. The total non-centralized scatter matrix in the form of kernel matrix is

\[
K = 2 \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} (\Phi(x_i^j) - \bar{u})(\Phi(x_j^j) - \bar{u})' \\
= 2 \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} (\Phi(x_i^j) - \bar{u}_i + \bar{u}_i - \bar{u})(\Phi(x_j^j) - \bar{u}_i + \bar{u}_i - \bar{u})' \\
= S^\Phi_w + S^\Phi_b + 2 \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} (\Phi(x_i^j) - \bar{u}_i)(\bar{u}_i - \bar{u})' + 2 \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} (\bar{u}_i - \bar{u})(\Phi(x_j^j) - \bar{u}_i)' \\
\tag{2}
\]

The third term of Eq. (2) can be rewritten as

\[
\sum_{i=1}^{\ell} \sum_{j=1}^{\ell} (\bar{u}_i - \bar{u})(\Phi(x_j^j) - \bar{u}_i)' \\
= \sum_{i=1}^{2} \left( \sum_{j=1}^{\ell} (\Phi(x_i^j) - \bar{u}_i)' \right)(\bar{u}_i - \bar{u})' \\
= 2 \sum_{i=1}^{n} (n_i \bar{u}_i - n_i \bar{u})(\bar{u}_i - \bar{u})' \\
\tag{3}
\]

Note that \( n_i \bar{u}_i = \sum_{j=1}^{n_i} \Phi(x_i^j) \). Then the third term and fourth term of Eq. (2) are equal to zero. Thus, we have the relation \( K = S^\Phi_w + S^\Phi_b = S^\Phi + P \Sigma^{\Phi}_P + \Sigma^{\Phi}_N \), where \( P \) is the number of positive instances, \( N \) is the number of negative instances, \( \Sigma^{\Phi}_P \) is the positive covariance matrices, and \( \Sigma^{\Phi}_N \) is the negative covariance matrices. Since class distribution has a great impact on \( S^\Phi_w \), the class imbalance also impacts the diagonalization problem of PCA.

In order to combat the class imbalance problem, we propose the AKPCC, based on kernel method. It considers the correlation between principal components and the class distribution. The imbalance ratio can be denoted as \( \frac{\sum_{i=1}^{\ell} I(y_i, +1)}{\sum_{i=1}^{\ell} I(y_i, -1)} = \frac{p}{N} \), which is the probabilities of the positive instances to the negative instances of training data, where \( I(\cdot) \) is an indicator function. We assume that future test examples are drawn from the same distribution, so the imbalance ratio of the training data is the same as that of the test data. Then, we have

\[
\sum_{i=1}^{\ell} (\bar{y}_i - \bar{b}) I(y_i, +1) = \frac{\sum_{i=1}^{\ell} I(y_i, +1)}{\sum_{i=1}^{\ell} I(y_i, -1)} = \frac{p}{N} \tag{4}
\]

where \( \bar{b} \) is the bias of the classifier, \( \bar{y}_i \) is the regression result of \( x_i \), \( y_i \) can be computed by regression model Eq. (1). Note that the regression is conducted on the \( p \) principal components. Solving this one variable equation, we get

\[
\bar{b} = \frac{N \left( \sum_{i=1}^{\ell} (\bar{y}_i I(y_i, +1)) \right) - P \left( \sum_{i=1}^{\ell} (\bar{y}_i I(y_i, -1)) \right)}{N^2 - \bar{p}^2} \tag{5}
\]

Based on principal components, Eq. (5) describes the detail deviation of the classifier. This deviation may be caused by class imbalance, noise et al. In order to retrieve the harmful effect, we compensate this deviation. By transforming the regression model Eq. (1), the classifier model can be written as

\[
H(x) = \text{sign} \left( \sum_{k=1}^{p} w_k b_k (x_k) + \bar{b} \right) \\
= \text{sign} \left( \sum_{k=1}^{p} w_k \sum_{i=1}^{\ell} \alpha_k^i k(x_i, x) + \bar{b} \right) \\
= \sum_{i=1}^{\ell} c_i k(x_i, x) + \bar{b} \tag{6}
\]

where \( \{c_i = \sum_{k=1}^{p} w_k \alpha_k^i, i = 1, 2, \ldots, \ell\}. \)

AKPCC is summarized as in Algorithm 1. AKPCC originates from the need to combat the class imbalance problem in the classification. It inherits the advantage of kernel method, which can conduct quite general dimensional feature space mappings. This algorithm removes the unreliable dimensions based on KPCA, and solves the imbalance problem based on the principal components.

**Algorithm 1 AKPCC**

**Require:**

- The set of labeled samples, \( L = \{(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)\} \);
- The set of unlabeled samples, \( U = \{x_{t+1}, x_{t+2}, \ldots, x_{t+u}\} \);

**Ensure:**

Kernel Principal Component Classifier, \( H \):

1. \( K_{ij} = k(x_i, x_j), i, j = 1, \ldots, \ell; \)
2. \( K = K - \frac{1}{\ell} J^T J K - \frac{1}{\ell} K J J^T + \frac{1}{\ell^2} J^T K J J^T \); % J is the all 1s vector
3. \( [V, A] = \text{eig}(K) \);
4. \( \alpha = \sum_{j=1}^{p} \frac{1}{\ell} \sum_{i=1}^{\ell} (V_i^T Y_i) W_i; \% Y = [y_1, y_2, \ldots, y_t] \) is the label vector
5. Calculate \( \bar{b}, H(x) \) according to Eqs. (5) and (6);
6. Return \( H \).

\(^1I(\cdot)\) is an indicator of the expression is true or not. Here, \( I(x, y) = 1 \) if \( x = y \), zero otherwise.
Table 1 Statistical F-measure(mean ± std) and GMeans(mean ± std) values of five classifiers on all data sets. The line w/t means that the algorithm at the corresponding AKPCC wins in w data sets, ties in t data sets, and loses in l data sets, compared with the algorithm at the corresponding column.

<table>
<thead>
<tr>
<th>project</th>
<th>APLSC</th>
<th>KCPC</th>
<th>AKPCC</th>
<th>AdaBoost</th>
<th>SMOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-measure metric</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar3</td>
<td>0.505 (0.070)</td>
<td>0.407 (0.115)</td>
<td>0.570 (0.087)</td>
<td>0.377 (0.037)</td>
<td>0.455 (0.002)</td>
</tr>
<tr>
<td>ar4</td>
<td>0.431 (0.026)</td>
<td>0.454 (0.058)</td>
<td>0.477 (0.037)</td>
<td>0.443 (0.002)</td>
<td>0.457 (0.000)</td>
</tr>
<tr>
<td>ar5</td>
<td>0.537 (0.061)</td>
<td>0.570 (0.085)</td>
<td>0.613 (0.037)</td>
<td>0.464 (0.014)</td>
<td>0.575 (0.001)</td>
</tr>
<tr>
<td>cm1</td>
<td>0.265 (0.043)</td>
<td>0.074 (0.033)</td>
<td>0.091 (0.022)</td>
<td>0.244 (0.003)</td>
<td>0.217 (0.000)</td>
</tr>
<tr>
<td>kc1</td>
<td>0.408 (0.005)</td>
<td>0.309 (0.006)</td>
<td>0.450 (0.006)</td>
<td>0.339 (0.005)</td>
<td>0.407 (0.000)</td>
</tr>
<tr>
<td>kc2</td>
<td>0.421 (0.011)</td>
<td>0.465 (0.016)</td>
<td>0.550 (0.026)</td>
<td>0.530 (0.001)</td>
<td>0.517 (0.000)</td>
</tr>
<tr>
<td>kc3</td>
<td>0.377 (0.037)</td>
<td>0.232 (0.032)</td>
<td>0.255 (0.072)</td>
<td>0.328 (0.001)</td>
<td>0.336 (0.000)</td>
</tr>
<tr>
<td>mw1</td>
<td>0.290 (0.010)</td>
<td>0.266 (0.083)</td>
<td>0.375 (0.049)</td>
<td>0.220 (0.002)</td>
<td>0.283 (0.000)</td>
</tr>
<tr>
<td>pc1</td>
<td>0.205 (0.021)</td>
<td>0.233 (0.028)</td>
<td>0.268 (0.026)</td>
<td>0.347 (0.001)</td>
<td>0.393 (0.000)</td>
</tr>
<tr>
<td></td>
<td>w/t</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMeans metric</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar3</td>
<td>0.705 (0.041)</td>
<td>0.455 (0.262)</td>
<td>0.744 (0.066)</td>
<td>0.679 (0.018)</td>
<td>0.635 (0.001)</td>
</tr>
<tr>
<td>ar4</td>
<td>0.674 (0.028)</td>
<td>0.592 (0.035)</td>
<td>0.612 (0.034)</td>
<td>0.640 (0.001)</td>
<td>0.612 (0.000)</td>
</tr>
<tr>
<td>ar5</td>
<td>0.734 (0.072)</td>
<td>0.698 (0.058)</td>
<td>0.813 (0.037)</td>
<td>0.538 (0.018)</td>
<td>0.712 (0.001)</td>
</tr>
<tr>
<td>cm1</td>
<td>0.438 (0.059)</td>
<td>0.208 (0.058)</td>
<td>0.211 (0.042)</td>
<td>0.525 (0.005)</td>
<td>0.410 (0.000)</td>
</tr>
<tr>
<td>kc1</td>
<td>0.595 (0.008)</td>
<td>0.416 (0.009)</td>
<td>0.462 (0.009)</td>
<td>0.612 (0.001)</td>
<td>0.573 (0.000)</td>
</tr>
<tr>
<td>kc2</td>
<td>0.639 (0.018)</td>
<td>0.594 (0.010)</td>
<td>0.682 (0.015)</td>
<td>0.706 (0.001)</td>
<td>0.657 (0.000)</td>
</tr>
<tr>
<td>kc3</td>
<td>0.692 (0.011)</td>
<td>0.113 (0.000)</td>
<td>0.110 (0.098)</td>
<td>0.571 (0.001)</td>
<td>0.523 (0.000)</td>
</tr>
<tr>
<td>mw1</td>
<td>0.664 (0.012)</td>
<td>0.392 (0.058)</td>
<td>0.412 (0.046)</td>
<td>0.491 (0.022)</td>
<td>0.481 (0.000)</td>
</tr>
<tr>
<td>pc1</td>
<td>0.479 (0.031)</td>
<td>0.378 (0.020)</td>
<td>0.386 (0.018)</td>
<td>0.639 (0.001)</td>
<td>0.570 (0.000)</td>
</tr>
<tr>
<td></td>
<td>w/t</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Data sets.

<table>
<thead>
<tr>
<th>project</th>
<th>modules</th>
<th>attributes</th>
<th>size(loc)</th>
<th>%defective</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar3</td>
<td>65</td>
<td>29</td>
<td>5,624</td>
<td>12.7</td>
</tr>
<tr>
<td>ar4</td>
<td>107</td>
<td>29</td>
<td>9,196</td>
<td>18.69</td>
</tr>
<tr>
<td>ar5</td>
<td>36</td>
<td>29</td>
<td>2,732</td>
<td>22.22</td>
</tr>
<tr>
<td>cm1</td>
<td>498</td>
<td>21</td>
<td>14,763</td>
<td>9.83</td>
</tr>
<tr>
<td>kc1</td>
<td>2,109</td>
<td>21</td>
<td>42,965</td>
<td>15.46</td>
</tr>
<tr>
<td>kc2</td>
<td>522</td>
<td>21</td>
<td>19,259</td>
<td>20.49</td>
</tr>
<tr>
<td>kc3</td>
<td>458</td>
<td>39</td>
<td>7,749</td>
<td>9.38</td>
</tr>
<tr>
<td>mw1</td>
<td>403</td>
<td>38</td>
<td>8,341</td>
<td>7.69</td>
</tr>
<tr>
<td>pc1</td>
<td>1,109</td>
<td>21</td>
<td>25,924</td>
<td>6.94</td>
</tr>
</tbody>
</table>

interval is conducted.

The results for the five methods are shown in Table 1. We can see that AKPCC outperforms other algorithms in the aspect of F-measure. In the aspect of GMeans, AKPCC is no better or even worse than other algorithms. But, AKPCC outperforms KPCC in the aspect of the two metrics. It means that, asymmetric learning method of kernel form is expected to set up a model adapted to more general situations.

4. Conclusion

In this paper, we introduce a kernel based asymmetric learning method for software defect prediction. It can combat the class imbalance problem in the software defect data. Experimental results on real data sets validate its effectiveness.

Acknowledgments

We thank R. Rosipal for providing us with the kernel PCA code, and LeVis Group for APLSC code. We also thank the anonymous reviewers for their great helpful comments.

References