21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Cost-Sensitive Feature Selection with Application in Software Defect Prediction

Linsong Miao, Mingxia Liu and Daoqiang Zhang*
Department of Computer Science and Engineering, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China
**E-mail Address: {linsongmiao, mingxialiu, dqzhang }@nuaa.edu.cn*

Abstract

In many real-world applications, different mis-classification errors will cause different costs. However, cost-sensitive learning only applied in classification phase and not in the feature selection phase to address this problem. In this paper, we study cost-sensitive feature selection and propose a framework which incorporates a cost matrix into traditional feature selection methods. And we developed three corresponding methods, namely, Cost-Sensitive Variance Score (CSVS), Cost-Sensitive Laplacian Score (CSLS), Cost-Sensitive Constraint Score (CSCS). Experiments on real software defect prediction benchmark data sets demonstrate that cost-sensitive feature selection methods are more efficacy than traditional ones in reducing the total cost.

1. Introduction

Machine learning techniques have been applied to construct prediction models for software defect prediction for many years [1]. However, with the growth in size and complexity of modern large software systems, learning a software defect predictor will have to face amounts of high-dimensional data. Thus, feature selection is first required to find the appropriate feature subset before learning the software defect prediction model.

Traditional feature selection [2][3][4] attempts to attain low classification errors instead of low costs. However, in software defect prediction, misclassifying a defect-prone module will cause higher cost than misclassifying a not-defect-prone module. Consequently, traditional feature selection is not suitable to reduce the total cost in software defect prediction. Traditional feature selection also does not consider the imbalance problem since they assign the same weighting to samples from different classes when the samples are used to evaluate each feature. However, data sets in software defect prediction are imbalanced. Over the past decade, cost-sensitive learning has been applied to reduce the total cost and address the imbalanced problem [5] [6]. However, these methods only apply the cost-sensitive learning methods in classification phase and not in feature selection phase.

To deal with the scenarios where different mis-classification cause different costs and the data sets are imbalanced in feature selection phase, we propose a framework which incorporates a cost matrix into traditional feature selection methods and developed three corresponding methods, namely, Cost-Sensitive Variance Score (CSVS), Cost-Sensitive Laplacian Score (CSLS), Cost-Sensitive Constraint Score (CSCS). Three cost-sensitive feature selection methods combine the cost information (cost matrix) in three traditional feature selection methods in the way of emphasizing the samples with higher costs and deemphasizing the samples with lower costs in the feature selection phase. Different from traditional feature selection, we argue that cost-sensitive feature selection can address the imbalance problem in the phase of feature selection in the way similar to cost-sensitive learning, which address imbalance problem in the phase of learning. And also, cost-sensitive feature selection aims to attain low cost instead of low errors. Experimental results validate the effectiveness and deficiency of the proposed methods.

2.Cost-Sensitive Feature Selection

2.1. Analysis

Assume that there are all c classes. Without loss of generality, we assume that misclassifying a sample of i-th class ($i \in \{1,\ldots,c-1\}$) to the c-th class will cause higher cost than misclassifying a sample of the c-th class to other classes. Here, we call the class from 1-th class to (c-1)-th class 'in-group' class, while the c-th class is called as 'out-group' class. As analyzed above, we divide the costs into three types:

1) Cost of false acceptance C_{OI}: cost of misclassifying 'out-group' class as 'in-group' class

2) Cost of false rejection C_{IO}: cost of misclassifying 'in-group' class as 'out-group' class
3) Cost of false identification C_{II}: cost of misclassifying one 'in-group' class as another 'in-group' class.

Following [7], we assign $C_{OI}=C_{OI}/C_{II}$, $C_{IO}=C_{IO}/C_{II}$, and $C_{II}=1$ with result unchanged. According to common sense, we assume that accepting any 'out-group' class will cause the same cost, and misclassifying a 'in-group' class as another 'in-group' class or 'out-group' class will cause different cost. Let cost(i, j) ($i, j \in \{1,\ldots,c\}$) denote the cost of misclassifying an example of the i-th class to the j-th class. The class label of sample X_i is assumed to be $l_i \in \{I_1,\ldots,I_{C-1},O\}$, where I_j (j=1, …, c-1) is the 'in-group' class and O is the 'out-group' class. Then, we can construct the cost matrix C, which is shown in Table I. In this paper, we propose the cost matrix should be set to be given by users.

Table 1. Cost Matrix

	I_1	…	I_{C-1}	O
I_1	0	…	C_{II}	C_{IO}
…	…	…	…	…
I_{C-1}	C_{II}	…	0	C_{IO}
O	C_{OI}	…	C_{OI}	0

2.2.CSVS

As in [7] [8], we define the function f(k) to describe the importance of the k-th class, where $1 \le k \le c$. The function f(k) is computed as follows:

$$f(k)\begin{cases}(c-2)C_{II}+C_{IO} & if\ k=1,2,\ldots c-1\\ (c-1)C_{OI} & otherwise\end{cases} \quad (1)$$

Where c is the number of classes, C_{II}, C_{IO}, C_{OI} are defined as above. Eq (1) denotes that the larger is the cost of the k-th class, the larger is the value of the f(k).

According to Variance feature selection method, we should select the features with maximum variance. In this paper, we also assume that, a "good" feature should the one on which the variance of 'out-group' class is as larger as possible than the variance of 'in-group' classes. Thus, the cost-sensitive variance of r-th feature is computed as follows:

$$V_r = \frac{\frac{1}{n_c}\sum_{i=1}^{n_c} f(c)(f_{ri}-\mu_r)^2}{\sum_{i=1}^{c-1}\frac{1}{n_i}\sum_{j=1}^{n_i} f(i)(f_{rj}-\mu_r)^2} \quad (2)$$

Where n_i is the samples number of i-th class ($1 \le i \le c$). f_{ri} denote the r-th feature of the i-th sample X_i, i=1,…,m; r=1,…,n. Define $\mu_r = \frac{1}{m}\sum_i f_{ri}$. f(i) is the important function. In Eq. (2), we use f(i) to weight the classes, in such way the variance of different class is weighted at the same time.

2.3.CSLS

Let y_i denote the class label of sample x_i. Similar to Laplacian Score, the cost-sensitive laplacian score L_r of the r-th feature, which should be minimized, is defined as follows:

$$L_r = \sum_{i,j}[-\cos t(y_i,y_j)](f_{ri}-f_{rj})^2 S_{ij} - \lambda\sum_i f(i)(f_{ri}-\mu_r)^2 D_{ii} \quad (3)$$

Where cost(y_i, y_j) can be easily obtained from the cost matrix, which is shown in Table I. D is a diagonal matrix with $D_{ii} = \sum_j S_{ij}$, and S_{ij}, which reflect the neighborhood relationship between sample X_i (i=1,…,m), is defined as follows [3]:

$$S_{ij}=\begin{cases}e^{-\frac{\|X_i-X_j\|^2}{t}}, & if\ X_i\ and\ X_j\ are\ neighbors\\ 0, & otherwise\end{cases} \quad (4)$$

Where t is a constant to be set, and 'X_i and X_j are neighbors' means that either X_i is among k nearest neighbors of X_i, or X_j is among k nearest neighbors of X_i. There is a regularization coefficient λ , which is used to balance the contribution of the two terms in Eq.(3). Since f(i) is usually larger than cost(y_i, y_j), we set $\lambda < 1$ in this paper.

2.4.CSCS

Given a set of samples X=[x_1, x_2, … , x_m], we can utilize its pairwise must-link constraints M={(x_i, x_j)| x_i and x_j belong to the same class} and pairwise cannot-link constraints C={(x_i, x_j)| x_i and x_j belong to the different class} as the supervision information. The cost-sensitive constraint score of C_r, which should be minimized, is defined as follows:

$$C_r = \sum_{(x_i-x_j)\in M}\cos t(y_i,y_j)(f_{ri}-f_{rj})^2 - \lambda\sum_{(x_i-x_j)\in C} f(y_i)(f_{ri}-f_{rj})^2 \quad (5)$$

Where C is sets of pairwise cannot-link constraints and M is sets of pairwise must-link constraints. λ is a regularization coefficient, which is used to balance the two terms in Eq.(5).

3. Experiments

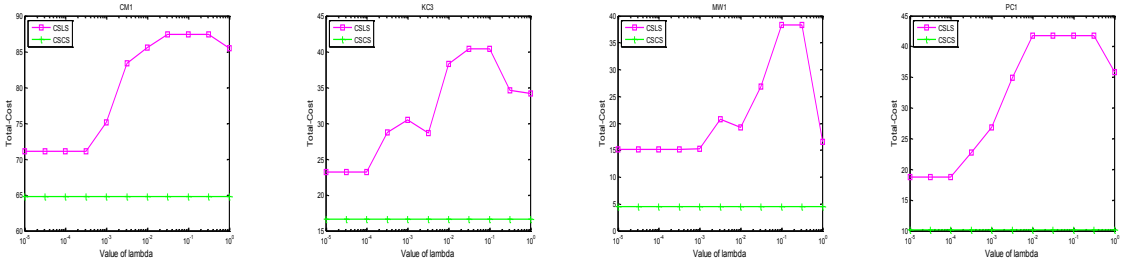
In this section, several experiments were carried out on NASA MDP data sets to demonstrate the efficiency and effectiveness of our algorithms.

3.1. Experimental Design

Seven software defect prediction sets, CM1, KC3, MW1, PC1, PC2, PC3 and PC4 [9], are studied in this paper. We take pre-processing for each data set as in [10]. LIbSVM, an open source library for SVM experimentation, are used to classify the data after feature selection. A ten-fold cross-validation is used in the experiment, the final performance estimation is obtained from averaging of the results of the tenfold.

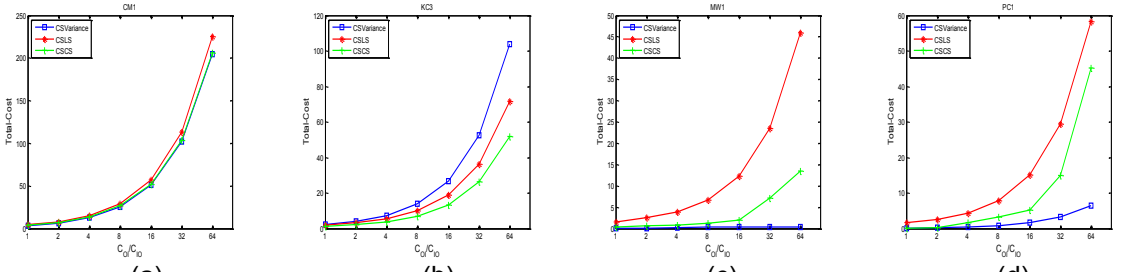
Table 2 Comparison on Total-Cost, Sensitivity, Accuracy, Specificity on the MDP Datasets (the numbers in the bracket represent the optimal features)

Mehtod		Feature Selecion+SVM						Baseline
		VS	CSVS	LS	CSLS	CS	CSCS	SVM
CM1	Cost	83.5(38)	63.3(5)	87.4(38)	71.1(31)	85.7(16)	64.8(13)	90.65
	Accuracy (%)	87.81	90.52	87.59	90.09	87.26	91.33	85.59
	Sensitivity(%)	95.50	96.53	96.22	97.03	95.41	97.84	94.00
	Specificity(%)	48.75	62.24	45.80	57.23	48.84	59.91	45.29
KC3	Cost	28.7(31)	22.8(9)	36.4(39)	23.2(15)	32.4(36)	16.6(12)	36.58
	Accuracy (%)	94.1	94.8	93.57	93.24	94.54	96.14	93.98
	Sensitivity(%)	97.65	97.31	98.69	95.93	98.69	98.00	98.68
	Specificity(%)	58.64	73.64	38.64	65.45	52.73	80.45	48.77
MW1	Cost	22.4(30)	0.4(2)	36.1(34)	15.1(12)	20.5(9)	4.5(10)	29.56
	Accuracy (%)	96.21	98.92	95.33	95.27	96.21	98.11	95.59
	Sensitivity(%)	98.86	98.86	99.71	96.86	98.57	98.57	98.97
	Specificity(%)	51.11	100	37.22	72.78	65.56	90.00	56.55
PC1	Cost	34(27)	2.0(1)	39.6(38)	18.7(12)	39.9(32)	10.2(7)	45.53
	Accuracy (%)	96.27	99.89	96.40	98.35	96.39	99.30	96.54
	Sensitivity(%)	97.75	100	98.20	99.21	97.87	99.78	98.73
	Specificity(%)	79.05	98.33	75.95	88.81	80.00	95.49	70.23
PC2	Cost	14.3(29)	0.1(2)	12.2(29)	6.1(11)	10(16)	0.1(1)	22.02
	Accuracy (%)	99.30	99.93	99.43	99.72	99.65	99.72	98.93
	Sensitivity(%)	99.78	99.93	99.86	99.93	100	99.93	99.69
	Specificity(%)	72.50	100	75.00	87.50	80.00	87.50	50.75
PC3	Cost	105.7(26)	0(1)	119.9(30)	19.6(11)	92.2(15)	17.7(11)	119.21
	Accuracy (%)	90.13	100	89.30	98.32	92.74	98.27	88.59
	Sensitivity(%)	92.71	100	92.30	98.76	95.19	98.68	91.43
	Specificity(%)	68.57	100	62.18	94.7	72.46	95.09	64.83
PC4	Cost	79.5(29)	0(2)	83.7(34)	44.1(11)	93.7(27)	30(15)	91.01
	Accuracy (%)	96.07	100	94.31	96.89	93.96	97.46	94.64
	Sensitivity(%)	98.74	100	96.83	98.20	96.85	98.28	97.58
	Specificity(%)	78.95	100	78.01	88.33	75.07	92.11	75.51



(a) (b) (c) (d)

Fig. 1.Total cost vs. different value of lambda in Cost-Sensitive Laplacian Score and Cost-Sensitive Constraint Score on 4 MDP data sets (a) CM1 (b)KC3 (c)MW1 (d)PC1



(a) (b) (c) (d)

Fig. 2.Total cost vs. different cost ratios in Cost-Sensitive Laplacian Score and Cost-Sensitive Constraint Score on 4 MDP data sets (a) CM1 (b)KC3 (c)MW1 (d)PC1

The parameter in Eq. (3) and Eq. (4) is always set to 10^{-4}, 0.8 separately, if without extra explanations. For each data set, a total of 100 pairwise constraints including 50 must-link and 50 cannot-link constraints are used in Constraint Score and its cost-sensitive variants [11]. C_{IO} is fixed as 1, and C_{II}, C_{OI} are set to 2, 20 separately in this study.

3.2. Experimental Results

In Table 2, we record the lowest cost of three algorithms after optimal dimensions are selected. Accuracy, sensitivity, specificity and the numbers of optimal dimensions are also recorded in Table 2 at the same times. From Table 2, we can find that the cost-sensitive methods achieve smaller total cost than their traditional counterparts in the best performance. And we also find that cost-sensitive methods need less features to achieve the best performance. It is evident that the cost-sensitive methods have larger sensitivity than their traditional counterparts. The larger sensitivity of cost-sensitive feature selection methods demonstrates the efficacy of cost-sensitive feature selection methods on imbalance datasets. The larger sensitivity indicates that cost-sensitive methods attain small total cost by preventing high-cost errors (error of false acceptance). All this demonstrates that cost-sensitive feature selection methods performance better than their traditional counterparts.

3.3. Discussion

First, we study the performance of cost-sensitive Laplacian Score and cost-sensitive Constraint Score with different value of the parameter λ. Fig.1 plots the total cost vs. λ values on four MDP data sets. From Fig.1, we can find that the different value of λ will lead to different cost for CSLS. However, we can find that the performance of CSCS has not been improved with the λ changed and the reason is that the first term in Eq. (5) is more important than first term for real software defect prediction benchmark data sets.

Then, we study the influence of the cost ratios on the performance of three cost-sensitive feature selection algorithms. Here, we study the adaptation of the cost-sensitive feature selection algorithm to different cost ratios. First, we set C_{IO} as 1. Then we select C_{OI}/C_{IO} from {1, 2, 4, 8, 16, 32, 64}. The results of three cost-sensitive feature selection algorithms are shown in Fig.2. In Fig.2, the cost ratio axes in log-scale for a better plot. Note that in the experiments, although C_{OI} increases exponentially, the total cost of three cost-sensitive methods do not increase exponentially. The reason is that the cost-sensitive feature selection methods control the total cost via selecting the features, which tends to reduce the high-cost errors of false acceptance. Thus, we can use cross-validation to balance the cost and accuracy in practical applications.

4. Conclusion

In this paper, we propose a cost-sensitive feature selection framework and develop three cost-sensitive feature selection algorithms, i.e., CSVS, CSLS, CSCS, which consider unequal misclassification costs and imbalance problem simultaneously in feature selection phase. Compared with traditional feature selection, cost-sensitive feature selection aim to minimize the total cost rather than the total error rate. Experimental results on public datasets show that cost-sensitive feature selection has better performance in reducing costs and addressing imbalance problem than traditional feature selection. In the experiments above, the cost matrix is given by users. Learning cost matrix from specific dataset is an interesting future issue.

References

[1] J. Zheng,"Cost-Sensitive Boosting Neural Network for Software Defect Prediction," Expert Systems with Applications, vol. 37, pp. 4537-4543, 2010.

[2] C. M. Bishop, "Neural Networks for Pattern Recognition," Oxford University Press. 1995

[3] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," In: Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 2005, pp. 507-514.

[4] D. Zhang, S. Chen, and Z. Zhou, "Constraint Score: A new filter method for feature selection with pairwise constraints," Pattern Recognition, vol. 41(5), pp. 1440-1451, 2008.

[5] Y. Zhang, and Z. H. Zhou, "Cost-sensitive face recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32(10), pp. 1758-1769, 2010.

[6] Z. H. Zhou, and X. Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," IEEE Transaction on Knowledge and Data Engineering, 2005.

[7] Z. H. Zhou, and X. Y. Liu, "On multi-class cost-sensitive learning," Proc. 21st National Conf. Artificial Intelligence (AAAI'06), Boston, MA, pp. 567-572, 2006.

[8] J. Lu, and Y. P. Tan, "Cost-Sensitive Subspace Learning for Face Recognition," IEEE Conf. Computer Vision and Pattern Recognition, (IEEE CVPR'2010), 2010.

[9] M. Chapman, P. Callis, and W. Jackson, "Metrics Data Program. NASA IV and V Facility," http:,, mdp.ivv.nasa.gov/, 2004.

[10] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Using the Support Vector Machine as a Classification Method for Software Defect Prediction with Static Code Metrics," Engineering Applications of Neural Networks,vol. 43, pp. 223-23,2009.

[11] D. Sun, and D.Q. Zhang, "Bagging Constraint Score for feature selection with pairwise constraints," Pattern Recognition, vol. 43(6), pp. 2106-2118, 2010.