Towards Cost-Sensitive Learning for Real-World Applications*

Xu-Ying Liu[1,2] and Zhi-Hua Zhou[2]

[1] School of Computer Science and Engineering, Southeast University, China
[2] National Key Laboratory for Novel Software Technology, Nanjing University, China
liuxy@seu.edu.cn, zhouzh@nju.edu.cn

Abstract. Many research work in cost-sensitive learning focused on binary class problems and assumed that the costs are precise. But real-world applications often have multiple classes and the costs cannot be obtained precisely. It is important to address these issues for cost-sensitive learning to be more useful for real-world applications. This paper gives a short introduction to cost-sensitive learning and then summaries some of our previous work related to the above two issues: (1) The analysis of why traditional Rescaling method fails to solve multi-class problems and our method Rescale$_{new}$. (2) The problem of learning with cost intervals and our CISVM method. (3) The problem of learning with cost distributions and our CODIS method.

1 Introduction to Cost-Sensitive Learning

1.1 Unequal Costs

Machine learning and data mining methods often aim at minimizing error rate. This implies that the costs of different misclassification errors are all equal. But in many real-world applications misclassification costs are often different. For a guard system, it is very dangerous to let in a stranger by mistake while a false alarm can be endured. In medical diagnosis, the cost of misdiagnose a patient having a life-threatening disease being healthy is much larger than the cost of misdiagnose a healthy person as a patient.

Here are two real applications of unequal costs. The first one is the network intrusion detection problem of KDD Cup 1999 [11]. The goal is to detect four types of network intrusions from normal connections: DOS (denial-of-service), R2L (unauthorized access from a remote machine), U2R (unauthorized access to local superuser (root) privileges) and probing (surveillance and other probing). The costs of different types of classification errors are different. For example, the consequence of giving access to a R2L connection is much more serious than to a probe one.

* The content of this paper is mainly from the Ph.D dissertation of the first author. This research was supported by Startup Foundation of Southeast University (4009001126) and Open Foundation of National Key Laboratory for Novel Software Technology of China (KFKT2011B01).

L. Cao et al. (Eds.): PAKDD 2011 Workshops, LNAI 7104, pp. 494–505, 2012.

In this application, the costs are only dependent on classes. That is to say, the costs of misclassifying the examples of one class to another class are all the same. This kind of cost is called class-dependent cost, and it can be represented by a matrix which is called cost matrix.

The second one is the donation problem of KDD Cup 1998 [11]. The task is to send promotions by mail to potential donors to achieve maximum benefit. The donation amounts are different for different person. Some people will donate more than \$100, some people will donate about \$5, while some others will not donate at all resulting a loss of \$0.68 of sending a mail. Therefore, the cost of missing a major donor is quite larger than missing a common one, and sending to those who will not donate will lose money.

In this application, there are two classes, i.e., donor and non-donor. The cost of sending a mail to a non-donor is *mail_cost*, and the benefit of sending a mail to a donor is $donate\_amount - mail\_cost$. So the costs dependent on examples. This type of cost is called example-dependent cost. It is different from class-dependent cost and cannot be represented by a cost matrix.

1.2 Formulation

Cost-sensitive learning tries to minimize total cost instead of error rate to handle unequal costs.

Suppose X is the d-dimensional input space and Y is the output space with $y \in \{1, 2, ..., c\}$. When costs are class-dependent, a training set $S = \{(x_i, y_i)\}_{i=1}^{n}$ are i.i.d. drawn from distribution D over $X \times Y$. Assume the cost of misclassifying an i-th class example to j-th class is $cost_{ij}$. The learning goal is to learn a hypothesis $h : X \to Y$ to minimize the expected cost:

$$\arg\min_{h} E_{(x,y)\sim D}[cost_{yh(x)}]. \tag{1}$$

When costs are example-dependent, let $cost(x, y, y_1)$ denote the cost of predicting example x belong to class y to class y_1, and $\bar{C}_{x,y}$ denote the cost vector of $[cost(x, y, 1), \ldots, cost(x, y, c)]$. Then, a training set $T = \{(x_i, y_i, \bar{C}_{x_i,y_i})\}_{i=1}^{n}$ are i.i.d. drawn from distribution $\mathcal{D}$ over $X \times Y \times R^{+c}$. The learning goal is to learn a hypothesis $h : X \to Y$ to minimize the expected cost:

$$\arg\min_{h} E_{(x,y,\bar{C}_{x,y})\sim \mathcal{D}}[cost(x, y, h(x))]. \tag{2}$$

1.3 Evaluation

It is reasonable to use total cost for evaluation when costs are fixed for a specific application. To evaluate a cost-sensitive learning method, it should be tested with different costs since learning methods are designed for general usage. Cost curve [6] is a popular evaluation method for cost-sensitive learning. The x-axis is the probability-cost function for positive examples, which is defined as:

$$PC(+) = \frac{p(+)cost_{+,-}}{p(+)cost_{+,-} + p(-)cost_{-,+}}, \tag{3}$$

where, $p(+)/p(-)$ is the prior distribution of positive/negative class. The y-axis is the total cost normalized by the cost of every example being misclassified:

$$Norm(E[Cost]) = \frac{fnr * p(+) * cost_{+,-} + fpr * p(-) * cost_{-,+}}{p(+)cost_{+,-} + p(-)cost_{-,+}}, \quad (4)$$

where, fpr is the false positive rate and fnr is the false negative rate. The lower a cost curve, the better the performance.

ROC curve [9] can also be used to evaluate cost-sensitive learning methods. It plots (fpr, tpr) (tpr is true positive rate) pairs. ROC curve is a dual representation of cost curve. But it cannot directly show how a method performs with a specific cost. While Cost curve is designed particular for cost-sensitive learning, and it has many good properties that ROC curve does not have.

1.4 Learning Methods

Cost-sensitive learning has attracted much attention from machine learning and data mining communities and has become an important research field [3,31,37,32,2]. The inclusion of costs into learning has been regarded not only as one of the most relevant topics of future machine learning research [22], but also one of the most challenging problems in data mining research [34].

Many cost-sensitive learning methods are developed, which can be roughly categorized into two types: (1) general methods which can adapt any standard classification methods minimizing error rate to handle unequal costs, including threshold-moving, sampling, and instance-weighting; (2) embedded methods which embed cost sensitivity specifically for a particular learning method.

Threshold-moving is a very popular cost-sensitive learning method and it is guaranteed by Bayes risk theory [25]. It lowers the decision threshold of posterior probability $p(y|x)$ for the class with higher cost so that expensive examples are easier to be predicted right. Suppose there is a positive class and a negative class, an example x should be predicted as positive when $p(+|x) \geq 0.5$ according to Bayes theory. When positive class has higher cost, the decision for x to be positive is made when $p(+|x) \geq p_0$, where p_0 is smaller than 0.5. This is because Bayes risk theory predicts an example to the class with the minimum expected loss. MetaCost [4], one of the pioneering work, is a threshold-moving method. It uses bagging on decision trees to predict posterior probability $p(y|x)$, then relabels training examples to the class with the minimum expected risk. After that, the relabeled data is used to train a classifier minimizing error rate. Many work devoted to improve the quality of probability estimation since threshold-moving relies greatly on it, such as [35,20].

Sampling methods gain cost-sensitivity by altering the class distributions $p(y)$. This type of methods increases or decreases examples for the class with higher cost or lower cost, respectively. Then a classifier trained to minimize error rate of the new data is sensitive to unequal costs of the original problem. Sampling methods are guaranteed by Elkan theorem [7]. Over-sampling can increase examples and under-sampling can decrease examples. Random over-sampling has the risk of over-fitting since it replicates examples and the new data is not i.i.d.

sampled. Costing is proposed [36] to overcome this problem by using rejection sampling which samples an example from the data and then keep it with a probability proportional to its misclassification cost. The size of the sampled data by rejection sampling is usually small. Costing uses bagging to ensemble classifiers trained by multiple sampled data to ease this problem. Roulette sampling [23] is proposed to solve this problem further. It can generate sampled data sets with different sizes.

Instance-weighting methods assign weights to examples proportional to their misclassification costs. The examples with higher costs have larger weights so that they are easier to be predicted correctly. C4.5CS [28] is very popular in cost-sensitive learning and it is one of instance-weighting methods. It exploits C4.5's ability of handling missing values [21] to utilize weighted examples. That is, the weights are used to calculate the probability of node t belong to the j-th class $p(j|t)$. There are also other instance-weighting methods, such as cost-sensitive Naive Bayes [36] and cost-sensitive support vector machines [36,1].

Though threshold-moving, sampling and instance-weighting make classifiers sensitive to unequal costs in different ways, they are closely connected to each other via Bayes risk theory. These general cost-sensitive methods are called Rescaling methods [41], since they all rescale the influence of different examples in the learning process in proportion to their misclassification costs. This will be introduced with more details in Section 3.1.

There are many embedded methods which design cost sensitivity in a particular way. Many efforts were devoted to make AdaBoost [10] cost-sensitive, such as CSB0, CSB1, CSB2 [27], AdaC1, AdaC2, AdaC3 [24], AdaCost [8], Asymmetric-AdaBoost [30] and Asymmetric Boosting [18]. And there are many cost-sensitive decision trees [5,29], cost-sensitive neural networks [14,40], and cost-sensitive Naive Bayes [12,13].

2 Towards Real-World Applications

Though cost-sensitive learning has gained some achievements, there are often strong assumptions for these methods to be applied successfully in real-world applications.

Before the year of 2004, most of the cost-sensitive learning methods were designed for binary-class problems. While many real-world applications have multiple classes, such as the network intrusion detection problem of KDD Cup 1999 (see Section 1.1). Simple extension of the methods designed for binary-class cases failed to reduce total cost for these multi-class problems.

In 2004, Lee et al. [15] proposed a multi-class cost-sensitive SVM and Abe et al. proposed an iterative method GBSE for multi-class cost-sensitive learning. In 2006, our previous work [39] analyzed why Recaling method fails to solve multi-class problems and proposed Rescale$_{new}$[1], which will be introduced in Section 3. These are some early work on multi-class cost-sensitive learning. Recent advances on this problem include a logistic regression method mcKLR proposed by Zhang

[1] A longer version is [41].

and Zhou [38], a Boosting method L_p-CSB proposed by Lozano and Abe [17], a threshold-moving method proposed by O'Brien et al. [19], and a reduction method proposed by Xia et al. [33]. Though our method Rescale$_{new}$ is one of the early methods, it still achieves good results compared to many others [33].

On the other hand, the cost information is provided by domain knowledge and is assumed to be precise. The classifiers will then be well tuned to reduce the total cost w.r.t. this particular cost value. However, in many real-world situations, although the user knows that one type of mistake is more severe than another type, it may be difficult to specify a precise cost value. The aspects that can lead to imprecise costs include but not limited to:

- Inherent impreciseness. Some information is naturally stochastic, e.g., one may donate \$1 or \$5 randomly.

  Unknown information. Sometimes we can't know everything about a system. For example, customers' SSN and salary information can't be obtained because of privacy.
- Variations in modeling. In the process of modeling risk, the approach may sample data, transform input space, or using different parameters. Due to these variations, the model may not provide precise assessment for costs.
- Expert opinions. Experts may have different opinions. And sometimes, experts can just give fair estimates of real risks.
- Dynamic environments. Environments always change. The risk assessed today may change tomorrow, e.g., due to the appearance of a new competitor.

Our previous work [16] quantifies imprecise costs with cost intervals and cost distributions and then proposed methods for both cases, which will be introduced in Section 4. Current cost-sensitive methods can be applied only when precise costs are given. To the best of our knowledge, there is no methods learning with cost intervals or cost distributions. A related work is [26], which considers that costs change over time. But it assumes true cost is known at time of classification. In our assumption, true cost is always unknown. ROC curve can evaluate classifiers under imprecise class distributions or misclassification costs. The classifiers with larger AUC (area under ROC curve) are regarded as better ones. This essentially assumes that nothing whatsoever is known about the relative severity of costs, which is a very rare situation in real-world problems. In our problem settings, cost interval or cost distribution is known.

3 Extending Rescaling to Multi-class Problems

Our previous work [39] analyzed why traditional Rescaling methods are not effective for multi-class problems and then proposed Rescale$_{new}$ method.

3.1 Analysis

Suppose the costs are class-dependent. Recall the notations defined in Section 1.2. We further assume that correct predictions cost zero, i.e., $cost_{ii} = 0$

for $i = 1, \ldots, c$. Let n_i denote the size of the i-th class. To simplify the discussion, all classes have the same size, that is, $n_i = n/c, (i = 1, \ldots, c)$.

Bayes risk theory predicts x to the class with the minimum expected cost. When there are 2 classes, the optimal decision of x to be the 1st class when the following holds:

$$(1 - p) \times cost_{21} \leq p \times cost_{12}, \tag{5}$$

where $p = p(class = 1|x)$. The left and right term is the expected cost of predicting the 1st class and 2nd class, respectively (recall that correct predictions cost zero). When the inequality of Eq. 5 becomes equality, predicting either class is optimal. And the p value making the equality holds is called decision threshold, which is denoted by p^*:

$$p^* = \frac{cost_{21}}{cost_{21} + cost_{12}}. \tag{6}$$

When $p \geq p^*$, the optimal decision for x is the 1st class. Threshold-moving directly applies Bayes risk theory. It firstly estimates posterior probability $p(y|x)$, then makes decisions according to Eq. 5. In threshold-moving, the reverse of the decision threshold for a class reflects how important this class is, which satisfies:

$$\frac{1/p^*}{1/(1 - p^*)} = \frac{cost_{12}}{cost_{21}}. \tag{7}$$

Sampling methods are guaranteed by Elkan theorem, which can be easily derived from Bayes risk theory. Due to page limit, please refer to [7] for details.

Theorem 1. *Elkan Theorem [7]: To make a target probability threshold p^* correspond to a given probability threshold p_0, the number of the 2nd class examples in the training set should be multiplied by $\frac{p^*}{1-p^*}\frac{1-p_0}{p_0}$.*

When the classifier has no bias to any class (i.e., minimizing error rate), the threshold p_0 is 0.5. Then according to Elkan Theorem and Bayes risk theory, when the number of the 2nd class examples is multiplied by $cost_{21}/cost_{12}$, p^* is the optimal solution of Bayes risk theory. This means when the 2nd class has higher cost, its size should be increased. Thus, a cost-sensitive problem can be reduced to a standard classification problem by altering class distributions accordingly. Suppose n'_i is the altered class size, then we have:

$$\frac{n'_1/n_1}{n'_2/n_2} = \frac{1}{\frac{cost_{21}}{cost_{12}}} = \frac{cost_{12}}{cost_{21}}. \tag{8}$$

That implies the change of the size of a class reflects its importance.

Instance-weighting uses examples' weights to reflect their importance. Assuming w_i is the weight for the examples in the i-th class, then,

$$\frac{w_1}{w_2} = \frac{cost_{12}}{cost_{21}}. \tag{9}$$

Therefore, threshold-moving, sampling and instance-weighting can be represented in a unified framework, which is Rescaling method. It rescales the 1st

class against the 2nd class according to:

$$\tau_{opt}(1,2) = \frac{cost_{12}}{cost_{21}}. \tag{10}$$

Where, $\tau_{opt}(1,2)$ is called the rescaling ratio.

When there are c classes with $c > 2$, the rescaling ratio should satisfy:

$$\tau_{opt}(i,j) = \frac{cost_{ij}}{cost_{ji}}. \tag{11}$$

The traditional Rescaling method uses $cost_i$ to reflect the importance of a class:

$$cost_i = \sum_{j=1}^{c} cost_{ij}, \tag{12}$$

$$\tau_{old}(i,j) = \frac{cost_i}{cost_j}. \tag{13}$$

When $c = 2$, $\tau_{old}(i,j) = \tau_{opt}(i,j)$. When $c > 2$, $\tau_{old}(i,j)$ is usually not equal to $\tau_{opt}(i,j)$. This explains why tradition Rescaling method fails to solve multi-class problems.

3.2 Rescale$_{new}$

Suppose each class is assigned a weight w_i by instance-weighting. In order to appropriately rescale all the classes simultaneously, the weights are expected to satisfy:

$$\frac{w_i}{w_j} = \tau_{opt}(i,j) = \frac{cost_{ij}}{cost_{ji}}. \tag{14}$$

This implies the following $\binom{2}{c}$ constraints should hold in the meanwhile:

$$\begin{array}{cccc} \frac{w_1}{w_2} = \frac{cost_{12}}{cost_{21}}, & \frac{w_1}{w_3} = \frac{cost_{13}}{cost_{31}}, & \ldots, & \frac{w_1}{w_c} = \frac{cost_{1c}}{cost_{c1}} \\ & \frac{w_2}{w_3} = \frac{cost_{23}}{cost_{32}}, & \ldots, & \frac{w_2}{w_c} = \frac{cost_{2c}}{cost_{c2}} \\ & \cdots & \cdots & \cdots \\ & & & \frac{w_{c-1}}{w_c} = \frac{cost_{c-1,c}}{cost_{c,c-1}} \end{array} \tag{15}$$

When these constraints hold simultaneously, Rescaling$_{new}$ directly applies the rescaling method, with the weights being the values that satisfy these constraints. When they can not hold simultaneously, Rescaling$_{new}$ splits the multiclass problem into a series of binary-class problems via pairwise coupling, then uses Rescaling to solve them one by one. The pseudo code can be found in [41].

4 Handling Imprecise Costs

Our previous work [16] studies two forms of imprecise costs: cost intervals and cost distributions.

Cost intervals is the cost information represented by intervals. There are several ways to obtain cost intervals, including but not limited to:

1. Natural cost intervals. In some applications, costs naturally have upper and lower bounds, e.g., stock investigating.
2. Expert opinions. It is much easier for domain expert to provide a cost interval than "precise" costs.
3. Transforming from confidence intervals. When there's no clear upper and lower bound of cost, we can use a confidence interval of cost instead. For example, the 95% confidence interval indicates cost will appear in the interval with a probability of 0.95.

In additional to cost intervals, sometimes we can known more information about costs, such as cost distributions. In some applications, experts can provide the costs distributions according to their experience. For example, the normal and uniform distributions are very popular and can be easily recognized from experience. For complex distributions, we can build models to assess costs. Then the values provided by different models can be regarded as samples from the underlying cost distribution.

4.1 Learning with Cost Intervals

We consider class-dependent costs and binary classification problems with $y \in \{=1, -1\}$. Assume correct prediction cost 0, and let c_+ and c_- denote the cost of misclassifying a positive and negative example, respectively. Assume positive class has higher cost, i.e., $c_+ \geq c_-$. Since the optimal decisions are unchanged when a cost matrix is multiplied by a positive constant [7], we can simplify the costs by fixing the cost of negative class so that we only need to consider the cost of positive class, i.e., $c_- = 1$, $c_+ = c$ ($c \geq 1$). Let $C_\mu = 0.5(C_{min} + C_{max})$ denote the mean cost. The empirical risk w.r.t. a cost value c of a classifier h is:

$$\tilde{R}(h, c) = \sum_{i=1}^{n} l(c, h(x), y), \tag{16}$$

$$l(c, h(x), y) = cI(h(x) \neq y \wedge y = +) + I(h(x) \neq y \wedge y = -) \tag{17}$$

where, $l(c, h(x), y)$ is the real loss of x, $I(a) = 1$ if $a = true$ and 0 otherwise.

When the true cost is unknown and a cost interval is available, we assume that the unique true cost C^* is a random value in the interval $[C_{min}, C_{max}]$. The goal is to learn a classifier H^* minimizing the *true risk* $\tilde{R}^* = \tilde{R}(h, C^*)$. Unfortunately, since the true cost is unknown, $\tilde{R}^*$ can't be obtained to guide the learning process. To overcome this difficulty, some risk $\tilde{R}_s$ can be used instead to learn a classifier, which is in fact determined by some cost C_s, i.e., $\tilde{R}_s = \tilde{R}(h, C_s)$. Since in general $\tilde{R}_s$ and C_s are different from the true risk $\tilde{R}^*$ and the true cost C^*, respectively, we call them "surrogate risk" and "surrogate cost". By minimizing surrogate risk $\tilde{R}_s$, the optimal classifier h_s^* is expected to minimize the true risk $\tilde{R}^*$. But this is infeasible since $\tilde{R}^*$ is unknown. However, since the true cost can be any value in the cost interval, it is expected that any possible risk of h_s^* should be small enough. Obviously, not all surrogate risk will be good enough for this purpose, so an appropriate surrogate cost C_s must be carefully chosen. Thus, in order to learn a classifier making any possible risks small enough, we can

formulate the problem of learning with cost intervals as Eq. 18, by considering C_s as a variable for learning.

$$\begin{aligned} &\min_{h,C_s} \tilde{R}(h, C_s) \\ &\text{s.t. } p(\tilde{R}(h, c) < \epsilon) > 1 - \delta, \ \forall c \in [C_{min}, C_{max}] \\ &\quad C_{min} \leq C_s \leq C_{max}. \end{aligned} \tag{18}$$

Since there are infinite constrains in Eq. 18, it is intractable to get optimal solutions. To overcome this difficulty, CISVM tries to solve a relaxation with a small number of informative constraints. The first one is the worst case risk which is the upper bound of the risks w.r.t. any c in $[C_{min}, C_{max}]$. its optimal solution can make all the constraints in Eq. 18 hold. So, the worst case risk is appropriate to be used as surrogate risk $\tilde{R}(h, C_s)$ to guide the learning process. However, the worst case risk could be far away from the true risk. So its optimal solution could not make the true risk small enough sometimes. CISVM overcomes this difficulty by minimizing a second risk, the "mean" risk (the risk w.r.t. the mean cost C_μ) in the meanwhile to avoid overfitting to surrogate risk. This is because when cost distribution is unknown, the mean risk has the smallest maximal distortion of the true risk, so it is the best choice to reflect how good a classifier performs on the entire interval.

Assuming that the prediction function is $f = w^{\mathrm{T}}x + b$, CISVM utilizes a surrogate loss in the following form:

$$L(C_p, f(x), y) = I_{y=+}[C_p - yf(x)]_+ + I_{y=-}[1 - yf(x)]_+, \tag{19}$$

where $[a]_+ = max(a, 0)$ and $I_a = I(a)$. It means that, the loss for a negative and positive example is $L^- = [1 - yf(x)]_+$ and $L^+ = [C_p - yf(x)]_+$, respectively. $L(C_{max}, f(x), y)$ is the worse case risk and $L(C_\mu, f(x), y)$ is the mean risk. This form of loss has theoretical guarantee to have smaller risk distortion than SVM's hinge loss.

CISVM involves two parts: (1) minimizing the regularized worst case risk by learning a variation of SVM:

$$\begin{aligned} &\min_{w,b,\xi \geq 0} \ \|w\|^2/2 + \lambda \sum\nolimits_{i=1}^{n} \xi_i \\ &\text{s.t. } y_i(w^{\mathrm{T}}\phi(x_i) + b) \geq C_{max} - \xi_i, \ \forall i : y_i = +1 \\ &\quad y_i(w^{\mathrm{T}}\phi(x_i) + b) \geq 1 - \xi_i, \ \forall i : y_i = -1 \end{aligned} \tag{20}$$

where $\phi(x)$ is a feature map induced by a kernel function. (2) minimizing the mean risk by parameter selection on a validation set. The pseudo code can be found in [16].

An intuitive way to handle cost intervals is that, taking some value in a cost interval as the true cost, such as the minimal value, mean value, or maximal value, and then applying standard cost-sensitive learning methods. We showed theoretically that, they are not the best solutions. Experiments showed that CISVM is significantly superior to all of them.

4.2 Learning with Cost Distributions

Assume that cost c is independently drawn from distribution v with domain C, which is independent of X. Then the goal is to find a classifier h minimizing the expected risk over v:

$$\begin{aligned} R_{CD}(h, v) &= E_{c\sim v}[R(h, c)] \\ &= E_{c\sim v}E_{D(X,Y)}[l(c, h(x), y)]. \end{aligned} \tag{21}$$

Note that, this is different from learning with example-dependent costs (see Section 1.2) because costs are independent of examples in our settings.

An intuitive way to handle cost distributions is taking the expected cost $E[c]$ as true cost then exploiting standard cost-sensitive learning methods to minimize $R(h, E[c]) = E_{D(X,Y)}[l(E[c], h(x), y)]$. However, minimizing the risk is not equivalent to minimizing $R_{CD}(h, v)$ generally. So the intuitive way is not the optimal solution.

CODIS handles cost distributions by reducing the problem to a special case of example-dependent cost-sensitive learning problem, which has a theoretical guarantee (see Theorem 3 in [16]). Firstly, a cost sample c_i is drawn from v (or provided by a risk model) for each example (x_i, y_i) in training set S to form a new example set $\hat{S} = \{(x_i, y_i, c_i)\}_{i=1}^{n}$. Secondly, a standard example-dependent cost-sensitive method is called to learn a classifier minimizing the risk of $\hat{S}$. Furthermore, to reduce the variance caused by sampling from v, cost are sampled multiple times from v for a single example (x_i, y_i) since v and D are independent. Thus, the first two steps are repeated several times and all the classifiers form an ensemble. The pseudo code can be found in [16].

5 Conclusion

Many research work in cost-sensitive learning focused on binary class problems and assumed that the costs are precise. But these assumptions cannot hold in many real-world applications. This paper summaries some of our previous work towards relaxing these assumptions: (1) The analysis of the failure of traditional Rescaling method in multi-class problems and our method Rescale$_{new}$. (2) propose two methods to learn from cost intervals and cost distributions, respectively. Due to page limit, we only introduce the analysis and the proposed methods in this paper. Please refer to [39,41,16] for the detailed experimental results.

References

1. Brefeld, U., Geibel, P., Wysotzki, F.: Support Vector Machines with Example Dependent Costs. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 23–34. Springer, Heidelberg (2003)
2. Chawla, N., Japkowicz, N., Zhou, Z.-H. (eds.): Proceedings on PAKDD 2009 Workshop on Data Mining When Classes are Imbalanced and Errors Have Costs (2009)

3. Dietterich, T., Margineantu, D., Provost, F., Turney, P. (eds.): Proceedings of the ICML 2000 Workshop on Cost-Sensitive Learning (2000)
4. Domingos, P.: MetaCost: A general method for making classifiers cost-sensitive. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, San Diego, California, pp. 155–164 (1999)
5. Drummond, C., Holte, R.C.: Exploiting the cost of (in)sensitivity of decision tree splitting criteria. In: Proceedings of the 17th International Conference on Machine Learning, pp. 239–246. Morgan Kaufmann, San Francisco (2000)
6. Drummond, C., Holte, R.C.: Cost curves: An improved method for visualizing classifier performance. Machine Learning 65, 95–130 (2006)
7. Elkan, C.: The foundations of cost-sensitive learning. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, Washington, pp. 973–978 (2001)
8. Fan, W., Stolfo, S.J., Zhang, J., Chan, P.K.: AdaCost: Misclassification cost-sensitive boosting. In: Proceedings of the 16th International Conference on Machine Learning, Bled, Slovenia, pp. 97–105 (1999)
9. Fawcett, T.: ROC graphs: Notes and practical considerations for researchers. Tech. rep., HP Laboratories, Palo Alto, CA (2004)
10. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
11. Hettich, S., Bay, S.D.: The UCI KDD archive. University of California, Department of Information and Computer Science, Irvine, CA (1999), `http://kdd.ics.uci.edu`
12. Kolcz, A.: Local sparsity control for Naive Bayes with extreme misclassification costs. In: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, pp. 128–137 (2005)
13. Kołcz, A., Chowdhury, A.: Improved Naive Bayes for Extremely Skewed Misclassification. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 561–568. Springer, Heidelberg (2005)
14. Kukar, M., Kononenko, I.: Cost-sensitive learning with neural networks. In: Proceedings of the 13th European Conference on Artificial Intelligence, pp. 445–449 (1998)
15. Lee, Y., Lin, Y., Wahba, G.: Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. Journal of American Statistical Association 99(465), 67–81 (2004)
16. Liu, X.-Y., Zhou, Z.-H.: Learning with cost intervals. In: Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, pp. 403–412 (2010)
17. Lozano, A.C., Abe, N.: Multi-class cost-sensitive boosting with p-norm loss functions. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, pp. 506–514 (2008)
18. Masnadi-Shirazi, H., Vasconcelos, N.: Asymmetric boosting. In: Proceedings of the 24th International Conference, Corvalis, Oregon, pp. 609–61 (2007)
19. O'Brien, D.B., Gupta, M.R., Gray, R.M.: Cost-sensitive multi-class classification from probability estimates. In: Proceedings of the 25th International Conference on Machine learning, pp. 712–719 (2008)
20. Provost, F., Domingos, P.M.: Tree induction for probability-based ranking. Machine Learning 52(3), 199–215 (2003)
21. Quinlan, J.R.: C4. 5: Programs for machine learning. Morgan Kaufmann (2003)

22. Saitta, L., Lavrac, N.: Machine learning - a technological roadmap. Tech. rep. University of Amsterdam, The Netherland (2000)
23. Sheng, V.S., Ling, C.X.: Roulette Sampling for Cost-Sensitive Learning. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 724–731. Springer, Heidelberg (2007)
24. Sun, Y., Wong, A.K.C., Wang, Y.: Parameter Inference of Cost-Sensitive Boosting Algorithms. In: Perner, P., Imiya, A. (eds.) MLDM 2005. LNCS (LNAI), vol. 3587, pp. 21–30. Springer, Heidelberg (2005)
25. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 3rd edn. Elsevier (2006)
26. Ting, K.M., Zheng, Z.: Boosting Trees for Cost-Sensitive Classifications. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 190–195. Springer, Heidelberg (1998)
27. Ting, K.M.: A comparative study of cost-sensitive boosting algorithms. In: Proceedings of the 17th International Conference on Machine Learning, Standord, CA, pp. 983–990 (2000)
28. Ting, K.M.: An instance-weighting method to induce cost-sensitive trees. IEEE Transactions on Knowledge and Data Engineering 14(3), 659–665 (2002)
29. Turney, P.D.: Cost -sensitive classification: empirical evaluation of a hybrid genetic sensitive classification. Journal of Artificial Intelligence Research 2, 369–409 (1995)
30. Viola, P., Jones, M.: Fast and robust classification using asymmetric AdaBoost and a detector cascade. In: Advances in Neural Information Processing Systems, vol. 14, pp. 1311–1318 (2002)
31. Weiss, G.M., Saar-Tsechansky, M., Zadrozny, B. (eds.): Proceedings of the 1st International Workshop on Utility-Based Data Mining. ACM Press, Chicago (2005)
32. Weiss, G.M., Saar-Tsechansky, M., Zadrozny, B.: Special issue on utility-based data mining. Data Mining and Knowledge Discovery 17(2) (2008)
33. Xia, F., Yang, Y., Zhou, L., Li, F., Cai, M., Zeng, D.D.: A closed-form reduction of multi-class cost-sensitive learning to weighted multi-class learning. Pattern Recognition 42(7), 1572–1581 (2009)
34. Yang, Q., Wu, X.: 10 challenging problems in data mining research. International Journal of Information Technology and Decision Making 5(4), 597–604 (2006)
35. Zadrozny, B., Elkan, C.: Learning and making decisions when costs and probabilities are both unknown. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, pp. 204–213 (2001)
36. Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In: Proceedings of the 3rd IEEE International Conference on Data Mining, Melbourne, Florida, pp. 435–442 (2003)
37. Zadrozny, B., Weiss, G.M., Saar-Tsechansky, M. (eds.): Proceedings of the Second International Workshop on Utility-Based Data Mining. ACM Press, Philadelphia (2006)
38. Zhang, Y., Zhou, Z.-H.: Cost-sensitive face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(10), 1758–1769 (2010)
39. Zhou, Z.-H., Liu, X.-Y.: On multi-class cost-sensitive learning. In: Proceedings of the 21st National Conference on Artificial Intelligence, pp. 567–572 (2006)
40. Zhou, Z.H., Liu, X.Y.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on Knowledge and Data Engineering 18(1), 63–77 (2006)
41. Zhou, Z.-H., Liu, X.-Y.: On multi-class cost-sensitive learning. Computational Intelligence 26(3), 232–257 (2010)