

Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem

Zhi-Hua Zhou, *Senior Member, IEEE*, and Xu-Ying Liu

Abstract—This paper studies empirically the effect of *sampling* and *threshold-moving* in training cost-sensitive neural networks. Both oversampling and undersampling are considered. These techniques modify the distribution of the training data such that the costs of the examples are conveyed explicitly by the appearances of the examples. Threshold-moving tries to move the output threshold toward inexpensive classes such that examples with higher costs become harder to be misclassified. Moreover, *hard-ensemble* and *soft-ensemble*, i.e., the combination of above techniques via hard or soft voting schemes, are also tested. Twenty-one UCI data sets with three types of cost matrices and a real-world cost-sensitive data set are used in the empirical study. The results suggest that cost-sensitive learning with multiclass tasks is more difficult than with two-class tasks, and a higher degree of class imbalance may increase the difficulty. It also reveals that almost all the techniques are effective on two-class tasks, while most are ineffective and even may cause negative effect on multiclass tasks. Overall, threshold-moving and soft-ensemble are relatively good choices in training cost-sensitive neural networks. The empirical study also suggests that some methods that have been believed to be effective in addressing the class imbalance problem may, in fact, only be effective on learning with imbalanced two-class data sets.

Index Terms—Machine learning, data mining, neural networks, cost-sensitive learning, class imbalance learning, sampling, threshold-moving, ensemble learning.



1 INTRODUCTION

IN classical machine learning or data mining settings, the classifiers usually try to minimize the number of errors they will make in dealing with new data. Such a setting is valid only when the costs of different errors are equal. Unfortunately, in many real-world applications, the costs of different errors are often unequal. For example, in medical diagnosis, the cost of erroneously diagnosing a patient to be healthy may be much bigger than that of mistakenly diagnosing a healthy person as being sick, because the former kind of error may result in the loss of a life.

In fact, cost-sensitive learning has already attracted much attention from the machine learning and data mining communities. As it has been stated in the Technological Roadmap of the MLnetII project (European Network of Excellence in Machine Learning, [29]), the inclusion of costs into learning has been regarded as one of the most relevant topics of future machine learning research. During the past years, many cost-sensitive learning methods have been developed [6], [11], [14], [23], [31]. However, although there are many research efforts devoted to making decision trees cost-sensitive [5], [17], [24], [33], [35], [37], only a few

studies discuss cost-sensitive neural networks [19], [21], while usually it is not feasible to apply cost-sensitive decision tree learning methods to neural networks directly. For example, the instance-weighting method [33] requires the learning algorithm to accept weighted-examples, which is not a problem for C4.5 decision trees but is difficult for common feedforward neural networks.

Recently, the *class imbalance* problem has been recognized as a crucial problem in machine learning and data mining because such a problem is encountered in a large number of domains and, in certain cases, it causes seriously negative effects on the performance of learning methods that assume a balanced distribution of classes [15], [25]. Much work has been done in addressing the class imbalance problem [38]. In particular, it has been indicated that learning from imbalanced data sets and learning when costs are unequal and unknown can be handled in a similar manner [22], and cost-sensitive learning is a good solution to the class imbalance problem [38].

This paper studies methods that have been shown to be effective in addressing the class imbalance problem applied to cost-sensitive neural networks. On one hand, such a study could help identify methods that are effective in training cost-sensitive neural networks; on the other hand, it may give an answer to the question: Considering that cost-sensitive learning methods are useful in learning with imbalanced data sets, are learning methods for the class imbalance problem also helpful in cost-sensitive learning?

In particular, this paper studies empirically the effect of *oversampling*, *undersampling*, and *threshold-moving* in training cost-sensitive neural networks. *Hard-ensemble* and *soft-ensemble*, i.e., the combination of oversampling, undersampling, and threshold-moving via hard or soft voting schemes

- Z.-H. Zhou is with the National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China, and the Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai 200433, China. E-mail: zhouzh@lamda.nju.edu.cn.
- X.-Y. Liu is with the National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China. E-mail: liuxy@lamda.nju.edu.cn.

Manuscript received 1 Apr. 2005; accepted 22 July 2005; published online 18 Nov. 2005.

For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0122-0305.

are also tested. It is noteworthy that none of these techniques need modify the architecture or training algorithms of the neural networks, therefore, they are very easy to use. Twenty-one UCI data sets with three types of cost matrices and a real-world cost-sensitive data set were used in the empirical study. The results suggest that the difficulties of different cost matrices are usually different; cost-sensitive learning with multiclass tasks is more difficult than with two-class tasks and a higher degree of class imbalance may increase the difficulty. The empirical study also reveals that almost all the techniques are effective on two-class tasks, while most are ineffective on multiclass tasks. Concretely, *sampling* methods are only helpful on two-class tasks, while often causing a negative effect on data sets with a big number of classes; threshold-moving is excellent on two-class tasks, which is capable of performing cost-sensitive learning even on seriously imbalanced two-class data sets and effective on some multiclass tasks; soft-ensemble is effective on both two-class and multiclass tasks given that the data set is not seriously imbalanced, which is much better than hard-ensemble. Overall, the findings of the empirical study suggest that threshold-moving and soft-ensemble are relatively good choices in training cost-sensitive neural networks. Moreover, the empirical study suggests that cost-sensitive learning and learning with imbalanced data sets might have different characteristics, or some methods such as sampling, which have been believed to be effective in addressing the class imbalance problem, may in fact only be effective on learning with imbalanced two-class data sets.

The rest of this paper is organized as follows: Section 2 presents the learning methods studied in this paper. Section 3 reports on the empirical study. Section 4 discusses some observations. Section 5 concludes.

2 LEARNING METHODS

Suppose there are C classes and the i th class has N_i number of training examples. Let $Cost[i, c]$ ($i, c \in \{1..C\}$) denote the cost of misclassifying an example of the i th class to the c th class ($Cost[i, i] = 0$) and $Cost[i]$ ($i \in \{1..C\}$) denote the cost of the i th class. Moreover, suppose the classes are ordered such that, for the i th class and the j th class, if $i < j$, then $(Cost[i] < Cost[j])$ or $(Cost[i] = Cost[j] \text{ and } N_i \geq N_j)$. $Cost[i]$ is usually derived from $Cost[i, c]$. There are many possible rules for the derivation, among which a popular one is $Cost[i] = \sum_{c=1}^C Cost[i, c]$ [7], [33].

2.1 Oversampling

Oversampling changes the training data distribution such that the costs of the examples are conveyed by the appearance of the examples. In other words, this method duplicates higher-cost training examples until the appearance of different training examples are proportional to their costs.

Concretely, the k th class will have N_k^* training examples after resampling, which is computed according to (1):

$$N_k^* = \left\lceil \frac{Cost[k]}{Cost[\lambda]} N_\lambda \right\rceil. \quad (1)$$

TABLE 1
The Oversampling Algorithm

Training phase:

1. Let S be the original training set, S_k be its subset comprising all the k -th class examples ($k \in \{1..C\}$).
2. Put all the original training examples into S^* .
3. For classes with $(N_k^* > N_k)$ ($k \in \{1..C\}$), resample $(N_k^* - N_k)$ number of examples from S_k and put them into S^* .
4. Train a neural network from S^* .

Test phase:

1. Generate real-value outputs with the trained neural network.
2. Return the class with the biggest output.

Here, the λ -class has the smallest number of training examples to be duplicated, which is identified according to (2):

$$\lambda = \arg \min_j \frac{\frac{Cost[j]}{\min_c Cost[c]} N_{\arg \min_c Cost[c]}}{N_j}. \quad (2)$$

If $N_k^* > N_k$, then $(N_k^* - N_k)$ number of training examples of the k th class should be resampled, which is accomplished here by random sampling with replacement. The presented oversampling algorithm is summarized in Table 1.

Note that oversampling is a popular method in addressing the class imbalance problem, which resamples the small class until it contains as many examples as the other class. Although some studies have shown that oversampling is effective in learning with imbalanced data sets [15], [16], [22], it should be noted that oversampling usually increases the training time and may lead to overfitting since it involves making exact copies of examples [8], [13]. Moreover, there are also some studies that have suggested that oversampling is ineffective on the class imbalance problem [13].

Besides the algorithm shown in Table 1, this paper also studies a recent variant of oversampling, i.e., SMOTE [8]. This algorithm resamples the small class through taking each small class example and introducing synthetic examples along the line segments joining its small class nearest neighbors. For example, assume the amount of oversampling needed is 200 percent, then, for each small class example, two nearest neighbors belonging to the same class are identified and one synthetic example is generated in the direction of each. The synthetic example is generated in the following way: Take the difference between the attribute vector (example) under consideration and its nearest neighbor; multiply this difference by a random number between 0 and 1 and add it to the attribute vector under consideration. Default parameter settings of SMOTE are used in the empirical study. The detailed description of the algorithm can be found in [8].

2.2 Undersampling

Like oversampling, undersampling also changes the training data distribution such that the costs of the examples are

explicitly conveyed by the appearances of examples. However, the working style of undersampling opposites that of oversampling in the way that the former tries to decrease the number of inexpensive examples while the latter tries to increase the number of expensive examples.

Concretely, the k th class will have N_k^* training examples after resampling, which is computed according to (1). Here, the λ -class has the smallest number of training examples to be eliminated, which is identified according to (3):

$$\lambda = \arg \max_j \frac{\frac{\text{Cost}[j]}{\max_c \text{Cost}[c]} N_{\arg \max_c \text{Cost}[c]}}{N_j}. \quad (3)$$

If $N_k^* < N_k$, then $(N_k - N_k^*)$ number of training examples of the k th class should be eliminated. Here, a routine similar to that used in [18] is employed, which removes redundant examples at first and then removes borderline examples and examples suffering from the class label noise.

Redundant examples are the training examples whose part can be taken over by other training examples. Here, they are identified by the 1-NN rule [9]. In detail, some training examples are put into S^* at first. Then, for a class to be shrunk, all its examples outside of S^* are classified according to 1-NN in S^* . If the classification is correct, then the example is regarded as being redundant.

Borderline examples are the examples close to the boundaries between different classes. They are unreliable because even a small amount of attribute noise can send the example to the wrong side of the boundary. The borderline examples and examples suffering from the class label noise can be detected using the concept of *Tomek links* [34]. The idea could be put as follows: Take two examples, i.e., \mathbf{x} and \mathbf{y} , such that each belongs to a different class. Let $\text{Dist}(\mathbf{x}, \mathbf{y})$ denote the distance between them. Then, the pair (\mathbf{x}, \mathbf{y}) is called a Tomek link if no example \mathbf{z} exists such that $\text{Dist}(\mathbf{x}, \mathbf{z}) < \text{Dist}(\mathbf{x}, \mathbf{y})$ or $\text{Dist}(\mathbf{y}, \mathbf{z}) < \text{Dist}(\mathbf{y}, \mathbf{x})$. Here, the distance between two examples are computed according to (4), where d is the number of attributes among which the first j attributes are binary or nominal:

$$\text{Dist}(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{l=1}^j \text{VDM}(\mathbf{x}_{1l}, \mathbf{x}_{2l}) + \sum_{l=j+1}^d |\mathbf{x}_{1l} - \mathbf{x}_{2l}|^2}. \quad (4)$$

Let $N_{a,u}$ denote the number of training examples holding value u on attribute a and $N_{a,u,c}$ denote the number of training examples belonging to class c and holding value u on a . Then, VDM [30] is defined according to (5), which is employed in (4) to deal with binary or nominal attributes:

$$\text{VDM}(u, v) = \sum_{c=1}^C \left| \frac{N_{a,u,c}}{N_{a,u}} - \frac{N_{a,v,c}}{N_{a,v}} \right|^2. \quad (5)$$

The presented undersampling algorithm is summarized in Table 2.

Note that undersampling is also a popular method in addressing the class imbalance problem, which eliminates training examples of the oversized class until it matches the size of the other class. Since it discards potentially useful training examples, the performance of the resulting classifier may be degraded. Nevertheless, some studies have

TABLE 2
The Undersampling Algorithm

Training phase:

1. Let S be the original training set, S_k be its subset comprising all the k -th class examples ($k \in \{1..C\}$).
2. Set S to S^* , for the k -th class ($k \in \{1..C\}$):
 - 2a. Set $(S^* - S_k)$ to S^* . If $N_k^* < N_k$, randomly remove $\lfloor N_k^*/2 \rfloor$ number of examples from S_k and put these removed examples into S^* ; otherwise remove all the examples from S_k and put them into S^* .
 - 2b. If $S_k \neq \emptyset$, randomly pick an example \mathbf{x} in S_k and classify it in S^* with the 1-NN rule. If the classification is correct, then remove \mathbf{x} from S_k . This process is repeated until all the examples in S_k have been examined or the number of removed examples reaches $(N_k - N_k^*)$. Merge S_k into S^* .
 - 2c. If there are more than N_k^* number of k -th class examples in S^* , randomly pick a k -th class example \mathbf{x} and identify its nearest neighbor, say \mathbf{y} , in S^* . If \mathbf{y} and \mathbf{x} belong to different classes and \mathbf{x} is the nearest neighbor of \mathbf{y} in S^* , then remove \mathbf{x} from S^* . This process is repeated until there are exactly N_k^* number of k -th class examples in S^* , or all the k -th class examples have been examined.
 - 2d. If there are more than N_k^* number of k -th class examples in S^* , randomly remove some examples until there are exactly N_k^* number of k -th class examples.
3. Train a neural network from S^* .

Test phase:

1. Generate real-value outputs with the trained neural network.
2. Return the class with the biggest output.

shown that undersampling is effective in learning with imbalanced data sets [15], [16], [22], sometimes even stronger than oversampling, especially on large data sets [13], [15]. Drummond and Holte [13] suggested undersampling to be a reasonable baseline for algorithmic comparison, but they also indicated that undersampling introduces nondeterminism into what is otherwise a deterministic learning process. With a deterministic learning process, any variance in the expected performance is largely due to testing on a limited sample, but, for undersampling, there is also variance due to the nondeterminism of the undersampling process. Since the choice between two classifiers might also depend on the variance, using undersampling might be less desirable. However, as Elkan indicated [14], sampling can be done either randomly or deterministically. While deterministic sampling risks introducing bias, it can reduce variance. Thus, undersampling via deterministic strategies, such as the one shown in Table 2, can be a baseline for comparison.

2.3 Threshold-Moving

Threshold-moving moves the output threshold toward inexpensive classes such that examples with higher costs become harder to be misclassified. This method uses the

TABLE 3
The Threshold-Moving Algorithm

Training phase:

1. Let S be the original training set.
2. Train a neural network from S .

Test phase:

1. Generate real-value outputs with the trained neural network.
2. For every output, multiply it with the sum of the costs of misclassifying the corresponding class to other classes.
3. Return the class with the biggest output.

original training set to train a neural network, and the cost-sensitivity is introduced in the test phase.

Concretely, let O_i ($i \in \{1..C\}$) denote the real-value output of different output units of the neural network, $\sum_{i=1}^C O_i = 1$ and $0 \leq O_i \leq 1$. In standard neural classifiers, the class returned is $\arg \max O_i$, while, in threshold-moving, the class returned is $\arg \max_i O_i^*$. O_i^* is computed according to (6), where η is a normalization term such that $\sum_{i=1}^C O_i^* = 1$ and $0 \leq O_i^* \leq 1$:

$$O_i^* = \eta \sum_{c=1}^C O_i \text{Cost}[i, c]. \quad (6)$$

The presented threshold-moving algorithm is summarized in Table 3, which is similar to the *cost-sensitive classification* method [19] and the method for modifying the internal classifiers of MetaCost [32].¹ It is obvious that threshold-moving is very different from sampling because the latter relies on the manipulation of the training data while the former relies on manipulating the outputs of the classifier.

Note that threshold-moving has been overlooked for a long time such that it is not as popular as sampling methods in addressing the class imbalance problem. Fortunately, it has recently been recognized that “the bottom line is that when studying problems with imbalanced data, using the classifiers produced by standard machine learning algorithms without adjusting the output threshold may well be a critical mistake” [25]. It has also been declared that trying other methods, such as sampling, without trying simply setting the threshold may be misleading [25]. A recent study has shown that threshold-moving is as effective as sampling methods in addressing the class imbalance problem [22].

2.4 Hard-Ensemble and Soft-Ensemble

Ensemble learning paradigms train multiple component learners and then combine their predictions. Ensemble techniques can significantly improve the generalization ability of single learners, therefore, ensemble learning has been a hot topic during the past years [10]. Since different

1. It is worth noting that the original MetaCost method [11] does not explicitly manipulate the outputs of the classifier. In fact, the original MetaCost can be regarded as a mixed method which computes the probability estimates on the training data and then manipulates the training data to construct a cost-sensitive classifier.

cost-sensitive learners can be trained with the oversampling, undersampling, and threshold-moving algorithms, it is feasible to combine these learners into an ensemble.

Two popular strategies are often used in combining component classifiers, that is, combining the crisp classification decisions or the normalized real-value outputs. Previous research on ensemble learning [2] shows that these two strategies can result in different performance, therefore, here, both of them are tried.

Concretely, in both hard-ensemble and soft-ensemble, every component learner votes for a class and then the class receiving the biggest number of votes is returned. If a tie appears, that is, there are multiple classes receiving the biggest number of votes, then the class with the biggest cost is returned. The only difference between hard-ensemble and soft-ensemble lies in the fact that the former uses binary votes while the latter uses real-value votes. In other words, the crisp classification decisions of the component learners are used in hard-ensemble while the normalized real-value outputs of the component learners are used in soft-ensemble.

Note that, here, the component learners are generated through applying the oversampling, undersampling and threshold-moving algorithms directly to the training set. But it is evident that other variations such as applying these algorithms to bootstrap samples of the training set can also be used, which may be helpful in building ensembles comprising more component learners. The hard-ensemble and soft-ensemble algorithms are summarized in Table 4.

3 EMPIRICAL STUDY

3.1 Configuration

Backpropagation (BP) neural network [28] is used in the empirical study, which is a popular cost blind neural network easy to couple with the methods presented in Section 2. Each network has one hidden layer containing ten units and is trained to 200 epoches. Note that, since the relative instead of absolute performance of the investigated methods are concerned, the architecture and training process of the neural networks have not been finely tuned.

Twenty-one data sets from the UCI Machine Learning Repository [4] are used in the empirical study, where missing values on continuous attributes are set to the average value while those on binary or nominal attributes are set to the majority value. Information on these data sets is tabulated in Table 5.

Three types of cost matrices are used along with these UCI data sets. They are defined as follows [33]:

- (a) $1.0 < \text{Cost}[i, j] \leq 10.0$ only for a single value of $j = c$ and $\text{Cost}[i, j \neq c] = 1.0$ for all $j \neq i$; $\text{Cost}[i] = \text{Cost}[i, c]$ for $j \neq c$ and $\text{Cost}[c] = 1.0$.
- (b) $1.0 \leq \text{Cost}[i, j] = H_i \leq 10.0$ for each $j \neq i$; $\text{Cost}[i] = H_i$. At least one $H_i = 1.0$.
- (c) $1.0 \leq \text{Cost}[i, j] \leq 10.0$ for all $j \neq i$;

$$\text{Cost}[i] = \sum_{c=1}^C \text{Cost}[i, c].$$

At least one $\text{Cost}[i, j] = 1.0$.

TABLE 4
The Hard-Ensemble and Soft-Ensemble Algorithms

Training phase:

1. Let S be the original training set, S_k be its subset comprising all the k -th class examples ($k \in \{1..C\}$).
2. Execute the following steps to train the neural network NN_1 :
 - 2a. Put all the original training examples into S_1^* .
 - 2b. For classes with $(N_k^* > N_k)$ ($k \in \{1..C\}$), resample $(N_k^* - N_k)$ number of examples from S_k and put them into S_1^* .
 - 2c. Train NN_1 from S_1^* .
3. Execute the following steps to train the neural network NN_2 :
 - 3a. Set S to S_2^* , for the k -th class ($k \in \{1..C\}$):
 - 3aa. Set $(S_2^* - S_k)$ to S_2^* . If $N_k^* < N_k$, randomly remove $\lfloor N_k^*/2 \rfloor$ number of examples from S_k and put these removed examples into S_2^* ; otherwise remove all the examples from S_k and put them into S_2^* .
 - 3ab. If $S_k \neq \emptyset$, randomly pick an example \mathbf{x} in S_k and classify it in S_2^* with the 1-NN rule. If the classification is correct, then remove \mathbf{x} from S_k . This process is repeated until all the examples in S_k have been examined or the number of removed examples reaches $(N_k - N_k^*)$. Merge S_k into S_2^* .
 - 3ac. If there are more than N_k^* number of k -th class examples in S_2^* , randomly pick a k -th class example \mathbf{x} and identify its nearest neighbor, say \mathbf{y} , in S_2^* . If \mathbf{y} and \mathbf{x} belong to different classes and \mathbf{x} is the nearest neighbor of \mathbf{y} in S_2^* , then remove \mathbf{x} from S_2^* . This process is repeated until there are exactly N_k^* number of k -th class examples in S_2^* , or all the k -th class examples have been examined.
 - 3ad. If there are more than N_k^* number of k -th class examples in S_2^* , randomly remove some examples until there are exactly N_k^* number of k -th class examples.
 - 3b. Train NN_2 from S_2^* .
4. Train the neural network NN_3 from S .

Test phase:

Hard-ensemble:

1. Generate real-value outputs with NN_1 and identify the class c_1 which is with the biggest output.
2. Generate real-value outputs with NN_2 and identify the class c_2 which is with the biggest output.
3. Generate real-value outputs with NN_3 , and then:
 - 3a. For every output, multiply it with the sum of the costs of misclassifying the corresponding class to other classes.
 - 3b. Identify the class c_3 which is with the biggest output.
4. Vote c_1 , c_2 and c_3 to determine the winner class; if a tie appears, take the one with the biggest cost as the winner class.

Soft-ensemble:

1. Generate real-value outputs with NN_1 and then normalize the outputs, which results in a C -dimensional vector V_1 .
2. Generate real-value outputs with NN_2 and then normalize the outputs, which results in a C -dimensional vector V_2 .
3. Generate real-value outputs with NN_3 , and then:
 - 3a. For every output, multiply it with the sum of the costs of misclassifying the corresponding class to other classes.
 - 3b. Normalize the resulting real-value outputs, which leads to a C -dimensional vector V_3 .
4. $V = \sum_i V_i$. Identify the biggest component of V and regard its corresponding class as the winner class; if V has multiple biggest components, take the one with the biggest cost and regard the corresponding class as the winner class.

Recall that, as explained in Section 2, there are C classes, $Cost[i, c]$ ($i, c \in \{1..C\}$) denotes the cost of misclassifying an example of the i th class to the c th class ($Cost[i, i] = 0$), and $Cost[i]$ ($i \in \{1..C\}$) denotes the cost of the i th class. Examples of these cost matrices are shown in Table 6. Note that the unit cost is the minimum misclassification cost and all the costs are integers. Moreover, on two-class data sets, these three types of cost matrices have no difference since all of them become type (c) cost matrices. Therefore, the experimental results on two-class tasks and multiclass tasks will be reported in separate sections.

Under each type of cost matrix, 10 times 10-fold cross validation are performed on each data set except on *waveform*, where a randomly generated training data size of 300 and test data size of 5,000 are used in 100 trials, which is the way this data set has been used in some other cost-sensitive learning studies [33]. In detail, except on *waveform*, each data set is partitioned into 10 subsets with similar sizes and distributions. Then, the union of nine subsets is used as the training set while the remaining subset is used as the test set. The experiment is repeated 10 times such that every subset is used once as a test set. The average test result is the result of the 10-fold cross validation. The whole process described above is then repeated 10 times with randomly generated cost matrices belonging to the same cost type, and the average results are recorded as the final results, where statistical significance are examined.

Besides these UCI data sets, a data set with real-world cost information, i.e., the KDD-99 data set [3], is also used in the empirical study. This is a really large data set, which is utilized in the same way as that of Abe et al. [1]. Concretely, the so-called 10 percent training set is used, which consists roughly of 500,000 examples, and is further sampled down by random sampling 40 percent of them to get the data set of size 197,605 which is used in this study. Information on this data set is shown in Table 7. In each experiment, two thirds of the examples in this data set are randomly selected for training while the remaining one third is selected for testing. The experiment is repeated 10 times with different training-test partition and the average result is recorded. Since this is a multiclass data set, the experimental results will be reported in the section devoted to multiclass tasks.

3.2 Two-Class Tasks

As shown in Table 5, there are twelve two-class data sets. The detailed 10 times 10-fold cross validation results on them are shown in Table 8.

To compare the robustness of these methods, that is, how well the particular method α performs in different situations, a criterion is defined similar to the one used in [36]. In detail, the relative performance of algorithm α on a particular data set is expressed by dividing its average cost $cost_\alpha$ by the biggest average cost among all the compared methods, as shown in (7):

$$r_\alpha = \frac{cost_\alpha}{\max_i cost_i}. \quad (7)$$

The worst algorithm α^* on that data set has $r_{\alpha^*} = 1$ and all the other methods have $r_\alpha \leq 1$. The smaller the value of r_α , the better the performance of the method. Thus, the sum of r_α over all data sets provides a good indication of the robustness of the method α . The smaller the value of the sum, the better the robustness of the method. The distribution of r_α of each compared method over the experimental data sets is shown in Fig. 1. For each method, the 12 values of r_α are stacked for the ease of comparison.

Table 8 reveals that, on two-class tasks, all the investigated methods are effective in cost-sensitive learning because the misclassification costs of all of them are apparently less than that of sole BP. This is also confirmed by Fig. 1 where the robustness of BP is the biggest, that is,

TABLE 5
UCI Data Sets Used in the Empirical Study

Data set	Size	Attribute		Class	Class distribution
<i>echocardiogram</i>	131	1B	6C	2	88/43
<i>hepatitis</i>	155	13B	6C	2	32/123
<i>heart_s</i>	270		13C	2	150/120
<i>heart</i>	303		13C	2	164/139
<i>horse</i>	368	4B 11N	7C	2	232/136
<i>credit</i>	690	4B 5N	6C	2	307/383
<i>breast-w</i>	698		9C	2	457/241
<i>diabetes</i>	768		8C	2	500/268
<i>german</i>	1000		24C	2	700/300
<i>euthyroid</i>	3163	18B	7C	2	293/2870
<i>hypothyroid</i>	3163	18B	7C	2	151/3012
<i>coding</i>	20000	15N		2	10000/10000
<i>lymphography</i>	148	9B 9N		4	2/4/61/81
<i>glass</i>	214		9C	6	9/13/17/29/70/76
<i>waveform</i>	300 + 5000		21C	3	100/100/100
<i>soybean</i>	683	16B 19N		19	8/14/15/16/20*9/44*2/88/91*2/92
<i>annealing</i>	898	22B 10N	6C	5	8/40/67/99/684
<i>vowel</i>	990		10C	11	90*11
<i>splice</i>	3190	60N		3	767/768/1655
<i>abalone</i>	4177	1N	7C	3	1307/1342/1528
<i>satellite</i>	6435		36C	6	626/703/707/1358/1508/1533

(B: Binary, N: Nominal, C: Continuous)

the worst. Table 8 and Fig. 1 also disclose that the performance of SMOTE is better than that of undersampling but worse than that of oversampling. Moreover, the performance of oversampling, undersampling, and SMOTE are worse than that of threshold-moving and ensemble methods, while the performance of threshold-moving is comparable to that of the ensemble methods. It is noteworthy that, on two seriously imbalanced data sets, i.e., *enthroid* and *hypothyroid*, only threshold-moving is effective, while all the other methods except soft-ensemble on *enthroid* cause a negative effect.

When dealing with two-class tasks, some powerful tools such as ROC curve [26] or cost curve [12] can be used to measure the learning performance. Note that ROC and cost curves are dual representations that can be easily converted into each other [12]. Here, cost curve is used since it

explicitly shows the misclassification costs. The x-axis of a cost curve is the probability-cost function for positive examples, defined as (8), where $p(+)$ is the probability of a given example belonging to the positive class, $Cost[+, -]$ is the cost incurred if a positive example is misclassified to a negative class, and $p(-)$ and $Cost[-, +]$ are defined similarly. The y-axis is the expected cost normalized with respect to the cost incurred when every example is incorrectly classified. Thus, the area under a cost curve is the expected cost, assuming a uniform distribution on the probability-cost. The difference in area under two curves gives the expected advantage of using one classifier over another. In other words, the lower the cost curve, the better the corresponding classifier.

$$PCF(+) = \frac{p(+)Cost[+, -]}{p(+)Cost[+, -] + p(-)Cost[-, +]}. \quad (8)$$

The cost curves on the two-class data sets are shown in Fig. 2. On each figure, the curves corresponding to BP, oversampling, undersampling, threshold-moving, hard-ensemble, soft-ensemble, and SMOTE are depicted. Moreover, the triangular region defined by the points (0, 0), (0.5, 0.5), and (1, 0), i.e., the *effective range*, is outlined, inside of which useful nontrivial classifiers can be identified [12]. Note that, in order to obtain these curves, experiments with different cost-ratios have been performed besides these reported in Table 8.

TABLE 6
Examples of Three Types of Cost Matrices, $Cost[i, j]$

	Type (a)			Type (b)			Type (c)		
	<i>j</i>			<i>j</i>			<i>j</i>		
	1	2	3	1	2	3	1	2	3
<i>i</i>	1	0	1	8	1	0	3	3	6
	2	1	0	9	2	1	0	1	2
	3	1	1	0	3	6	6	0	3
							4	5	0

TABLE 7
The KDD-99 Data Set Used in the Empirical Study

Basic information		Cost matrix (misclassify the row-class to the col-class)					
Size: 197,605	Class distribution		Normal	Probe	DOS	U2R	R2L
Attribute:	38,910 (19.69%)	Normal	0	1	2	2	2
4 Binary	1,642 (0.83%)	Probe	1	0	2	2	2
3 Nominal	156,583 (79.24%)	DOS	2	1	0	2	2
34 Continuous	20 (0.01%)	U2R	3	2	2	0	2
Class: 5	450 (0.23%)	R2L	4	2	2	2	0

Fig. 2 exhibits that, on *echocardiogram*, undersampling is slightly worse than the other methods in the effective range, while SMOTE is very poor when $PCF(+)$ is smaller than 0.3. On *hepatitis*, the ensemble methods are significantly better than the other methods in the effective range, undersampling is very bad when $PCF(+)$ is smaller than 0.4, and oversampling, threshold-moving, and ensemble methods are poor when $PCF(+)$ is bigger than 0.85. On *euthyroid*, threshold-moving is the best, undersampling is the worst in the effective range, and the ensemble methods become poor when $PCF(+)$ is bigger than 0.8. On the remaining nine data sets, all the methods work well. On *heart_s*, the ensemble methods are slightly better than others. On *heart*, the ensemble methods are apparently better than oversampling, undersampling, and threshold-moving. On *horse*, threshold-moving and the ensemble methods are better than the other methods. On *credit*, undersampling and SMOTE are apparently worse than others. On *breast-w*, undersampling is slightly worse than the other methods. On *diabetes*, threshold-moving is the best while undersampling is the worst. On *german*, the ensemble methods are better than others. On *hypothyroid*, threshold-moving and oversampling are better than the other methods while undersampling is the worst. On *coding*, the ensemble methods are slightly better, while SMOTE is slightly worse than others. Totally, Fig. 2 reveals that all the cost-sensitive learning methods are effective on two-class tasks because, on all the data sets, the cost curves have a large portion or even almost fully appear in the effective range. Moreover, it discloses that the ensemble methods and threshold-moving are often better, while undersampling is often worse than the other methods.

In summary, the observations reported in this section suggest that, on two-class tasks:

1. Cost-sensitive learning is relatively easy because all methods are effective.
2. A higher degree of class imbalance may increase the difficulty of cost-sensitive learning.
3. Although the sampling methods and SMOTE are effective, they are not so good as threshold-moving and ensemble methods.
4. Threshold-moving is a good choice which is effective on all the data sets and can perform cost-sensitive learning even with seriously imbalanced data sets.
5. Soft-ensemble is also a good choice which is effective on most data sets and rarely cause negative effect.

3.3 Multiclass Tasks

As shown in Table 5, there are nine multiclass UCI data sets. The detailed 10 times 10-fold cross validation results on them with types (a) to (c) cost matrices are shown in Tables 9, 10, and 11, respectively. The comparison on the robustness of different methods are shown in Figs. 3, 4, and 5, respectively.

Table 9 shows that on multiclass UCI data sets with a type (a) cost matrix, the performance of oversampling, threshold-moving, and ensemble methods are apparently better than that of sole BP, while the performance of undersampling and SMOTE are worse than that of sole BP. Fig. 3 shows that soft-ensemble plays the best, while the robustness of undersampling is apparently worse than that of sole BP. Table 9 and Fig. 3 also show that threshold-moving and soft-ensemble are effective on all data sets and hard-ensemble causes negative effect on *soybean*, which is with the biggest number of classes and suffering from serious class imbalance. It is noteworthy that the sampling methods and SMOTE cause a negative effect on several data sets suffering from class imbalance, that is, *glass*, *soybean*, and *annealing*.

Table 10 shows that, on multiclass UCI data sets with type (b) cost matrix, the performance of threshold-moving and ensemble methods are apparently better than that of sole BP, while the performance of sampling methods and SMOTE are worse than that of sole BP. Fig. 4 shows that soft-ensemble plays the best, while the robustness of undersampling is apparently worse than that of sole BP. Table 10 and Fig. 4 also show that threshold-moving is always effective and soft-ensemble only causes negative effect on the most seriously imbalanced data set *annealing*. SMOTE and hard-ensemble cause a negative effect on *soybean* and *annealing*. It is noteworthy that the sampling methods cause a negative effect on almost all data sets suffering from class imbalance, that is, *lymphography*, *glass*, *soybean*, and *annealing*. It can be found from comparing Tables 9 and 10 that all the methods degrade when a type (a) cost matrix is replaced with a type (b) cost matrix, which suggests that the type (b) cost matrix is more difficult to learn than the type (a) cost matrix.

Table 11 shows that, on multiclass UCI data sets with a type (c) cost matrix, the performance of threshold-moving and soft-ensemble are better than that of sole BP, while the performance of the remaining methods are worse than that of sole BP. In particular, the average misclassification costs of undersampling and SMOTE are even about 2.4 and 1.9 times of that of sole BP, respectively. Fig. 5 confirms that soft-ensemble plays the best, while the sampling methods

TABLE 8
Experimental Results on Two-Class Data Sets

Data set	BP	over-sampling	under-sampling	threshold-moving	hard-ensemble	soft-ensemble	SMOTE
Cost: Misclassification cost							
<i>echocardiogram</i>	14.70 ± 5.90	.767 ± .167	.728 ± .243	.716 ± .163	.676 ± .167	.707 ± .173	.861 ± .262
<i>hepatitis</i>	8.22 ± 3.75	.896 ± .182	1.113 ± .313	.870 ± .144	.800 ± .103	.785 ± .110	1.363 ± .486
<i>heart.s</i>	17.67 ± 7.57	.886 ± .190	.760 ± .200	.794 ± .182	.709 ± .164	.697 ± .149	.791 ± .188
<i>heart</i>	19.32 ± 9.64	.878 ± .130	.800 ± .206	.823 ± .135	.756 ± .172	.752 ± .155	.802 ± .205
<i>horse</i>	19.73 ± 7.15	.803 ± .104	.929 ± .152	.777 ± .102	.731 ± .153	.717 ± .158	.843 ± .182
<i>credit</i>	38.69 ± 14.68	.766 ± .164	.822 ± .211	.665 ± .181	.656 ± .159	.657 ± .167	.935 ± .198
<i>breast-w</i>	8.72 ± 3.09	.945 ± .173	1.003 ± .143	.854 ± .168	.793 ± .156	.796 ± .180	.851 ± .244
<i>diabetes</i>	61.65 ± 18.67	.625 ± .166	.656 ± .159	.555 ± .148	.581 ± .163	.585 ± .163	.645 ± .143
<i>german</i>	82.63 ± 39.60	.801 ± .214	.813 ± .255	.712 ± .180	.696 ± .232	.690 ± .219	.782 ± .207
<i>euthyroid</i>	37.44 ± 13.67	1.279 ± .230	1.719 ± .646	.829 ± .111	1.016 ± .075	.967 ± .086	1.280 ± .212
<i>hypothyroid</i>	11.86 ± 5.06	1.604 ± .515	1.717 ± .528	.834 ± .109	1.127 ± .210	1.021 ± .201	1.569 ± .454
<i>coding</i>	2782.20 ± 1027.49	.403 ± .171	.404 ± .171	.405 ± .170	.400 ± .169	.398 ± .167	.403 ± .171
<i>ave.</i>	258.57 ± 795.07	.888 ± .304	.955 ± .397	.736 ± .138	.745 ± .188	.731 ± .162	.927 ± .324
No. HC Errors: Number of high cost errors							
<i>echocardiogram</i>	2.20 ± .45	.565 ± .227	.434 ± .218	.526 ± .155	.438 ± .167	.461 ± .173	.815 ± .334
<i>hepatitis</i>	1.45 ± .22	.800 ± .252	.586 ± .366	.788 ± .175	.627 ± .144	.608 ± .141	1.286 ± .871
<i>heart.s</i>	2.58 ± .22	.824 ± .197	.493 ± .271	.661 ± .190	.565 ± .171	.554 ± .161	.692 ± .186
<i>heart</i>	3.12 ± .34	.789 ± .119	.555 ± .261	.698 ± .142	.614 ± .182	.611 ± .167	.692 ± .205
<i>horse</i>	3.48 ± .36	.665 ± .179	.593 ± .252	.586 ± .097	.531 ± .181	.506 ± .196	.673 ± .211
<i>credit</i>	5.15 ± .54	.599 ± .196	.490 ± .199	.449 ± .178	.442 ± .175	.445 ± .195	.801 ± .442
<i>breast-w</i>	1.50 ± .24	.872 ± .203	.897 ± .159	.746 ± .184	.704 ± .199	.708 ± .231	.789 ± .348
<i>diabetes</i>	8.91 ± 2.03	.317 ± .268	.302 ± .259	.205 ± .098	.242 ± .214	.237 ± .208	.373 ± .243
<i>german</i>	13.43 ± 2.65	.601 ± .353	.479 ± .383	.480 ± .170	.431 ± .271	.409 ± .262	.592 ± .345
<i>euthyroid</i>	4.88 ± .34	.661 ± .571	.640 ± .457	.584 ± .107	.598 ± .346	.589 ± .303	.766 ± .688
<i>hypothyroid</i>	1.95 ± .28	1.475 ± 1.079	1.146 ± .756	.600 ± .122	.905 ± .514	.764 ± .422	1.403 ± 1.043
<i>coding</i>	375.41 ± 48.41	.115 ± .091	.096 ± .073	.116 ± .087	.069 ± .064	.068 ± .066	.121 ± .092
<i>ave.</i>	35.34 ± 107.15	.690 ± .329	.559 ± .266	.537 ± .203	.514 ± .216	.497 ± .194	.750 ± .344
No. Errors: Total number of errors							
<i>echocardiogram</i>	4.36 ± .25	1.132 ± .128	1.319 ± .322	1.148 ± .127	1.160 ± .117	1.193 ± .140	1.013 ± .112
<i>hepatitis</i>	2.94 ± .25	1.068 ± .143	2.032 ± 1.200	1.007 ± .132	1.082 ± .179	1.078 ± .203	1.529 ± .301
<i>heart.s</i>	5.53 ± .45	1.024 ± .180	1.414 ± .312	1.093 ± .066	1.039 ± .131	1.030 ± .133	1.006 ± .166
<i>heart</i>	6.18 ± .43	1.056 ± .090	1.382 ± .333	1.099 ± .119	1.072 ± .110	1.073 ± .126	1.034 ± .091
<i>horse</i>	7.03 ± .25	1.085 ± .186	1.574 ± .407	1.108 ± .100	1.117 ± .209	1.123 ± .196	1.171 ± .161
<i>credit</i>	10.26 ± .52	1.250 ± .123	1.780 ± .572	1.253 ± .131	1.264 ± .163	1.277 ± .187	1.461 ± .459
<i>breast-w</i>	3.06 ± .36	1.068 ± .118	1.217 ± .202	1.068 ± .119	.959 ± .084	.957 ± .096	1.005 ± .124
<i>diabetes</i>	17.96 ± .59	1.424 ± .305	1.574 ± .391	1.350 ± .145	1.430 ± .265	1.451 ± .291	1.353 ± .282
<i>german</i>	25.93 ± 1.16	1.313 ± .277	1.647 ± .573	1.184 ± .187	1.313 ± .317	1.347 ± .364	1.273 ± .250
<i>euthyroid</i>	9.79 ± .33	3.068 ± 1.937	5.173 ± 4.549	1.501 ± .223	2.264 ± 1.291	2.089 ± .969	2.827 ± 1.627
<i>hypothyroid</i>	4.07 ± .28	1.809 ± .569	3.439 ± 5.120	1.300 ± .269	1.512 ± .648	1.481 ± .709	1.885 ± .787
<i>coding</i>	729.63 ± 5.13	1.253 ± .074	1.269 ± .064	1.257 ± .070	1.277 ± .076	1.272 ± .080	1.244 ± .075
<i>ave.</i>	68.90 ± 208.19	1.379 ± .576	1.985 ± 1.168	1.197 ± .140	1.291 ± .348	1.281 ± .303	1.400 ± .521

The table entries present the real results of BP or the ratio of other methods against that of BP. The values following “±” are standard deviations.

and SMOTE are worse than sole BP. Table 11 and Fig. 5 also show that soft-ensemble only causes a negative effect on *glass* and the most seriously imbalanced data set *annealing*, while hard-ensemble causes a negative effect on one more data set, i.e., *soybean*. Threshold-moving does not cause a negative effect on *glass*, but it causes a negative effect on *lymphography* and *vowel*. The sampling methods and SMOTE cause a negative effect on more than half of the data sets. It is noteworthy that neither method is effective on the most

seriously imbalanced data set *annealing*. Comparing Tables 9, 10, and 11, it can be found that the performance of all the methods degrade much more when a type (b) cost matrix is taken over by a type (c) matrix than when a type (a) cost matrix is taken over by a type (b) cost matrix, which suggests that the type (c) cost matrix may be more difficult to learn than the type (b) cost matrix, and the gap between the type (b) and (c) cost matrices may be bigger than that between the type (a) and (b) cost matrices.

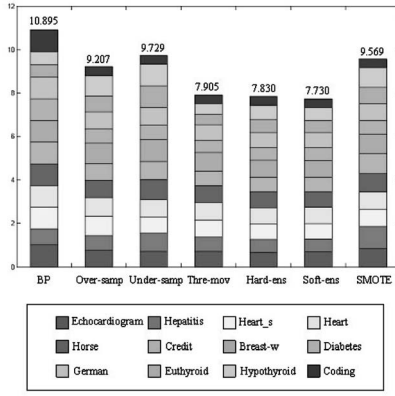


Fig. 1. Robustness of the compared methods on two-class data sets.

Table 12 presents the experimental results on the *KDD-99* data set. It can be found that the performance of threshold-moving is better than that of sole BP, while the performance of the ensemble methods and oversampling are worse than that of sole BP. However, pairwise two-tailed *t*-tests with .05 significance level indicate that these differences are without statistical significance. On the other hand, the performance of undersampling and SMOTE are apparently worse than that of sole BP. In other words, none of the studied cost-sensitive learning methods is effective on this data set, but oversampling, threshold-moving, and the ensemble methods do not cause a negative effect, while undersampling and SMOTE cause negative effect. The poor performance of undersampling is not difficult to be expected because on the *KDD-99* data set, the classes are seriously imbalanced,

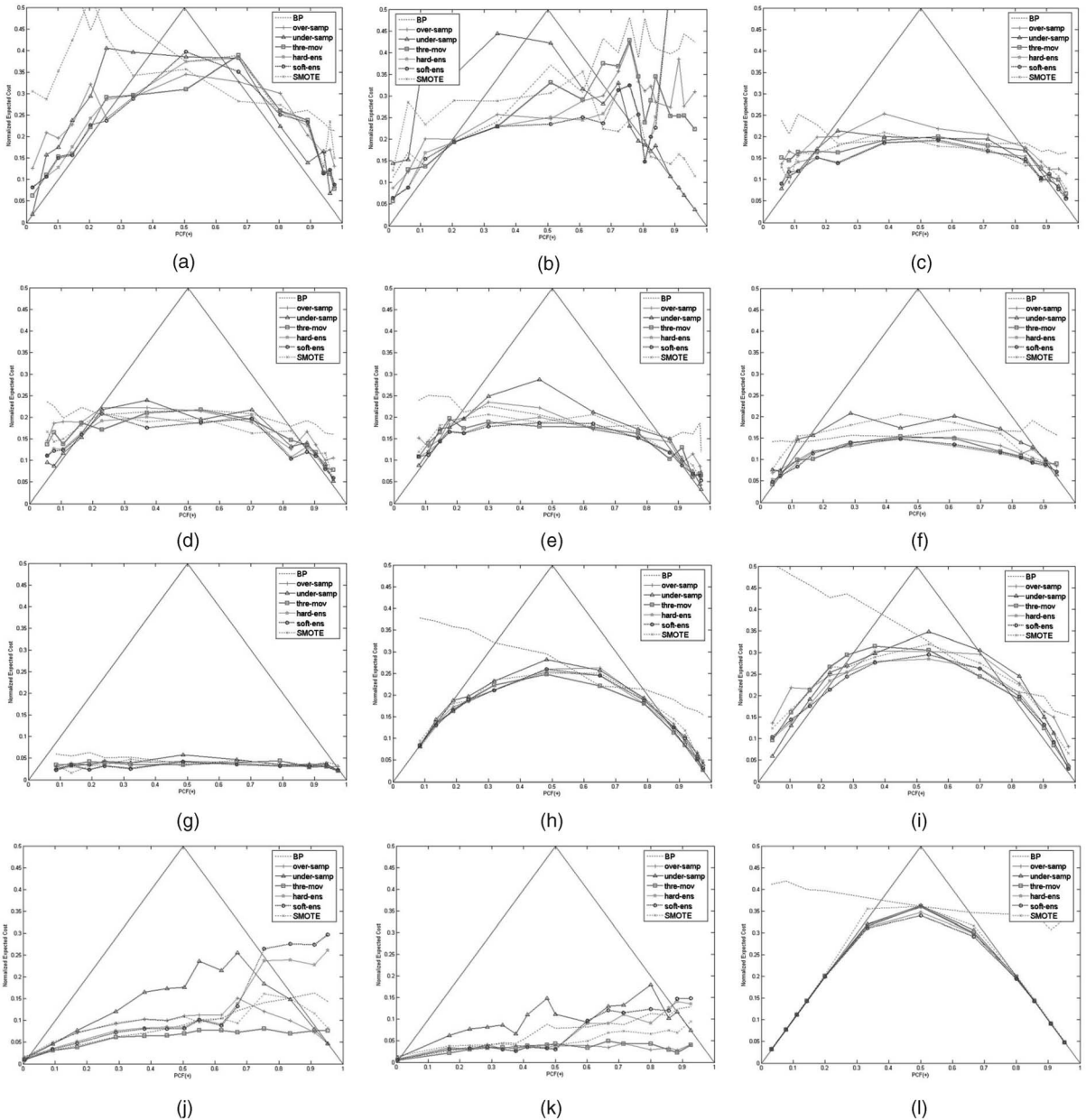


Fig. 2. Cost curves on two-class data sets. (a) echocardiogram. (b) hepatitis. (c) heart_s. (d) heart. (e) horse. (f) credit. (g) breast-w. (h) diabetes. (i) german. (j) euthyroid. (k) hypothyroid. (l) coding.

TABLE 9
Experimental Results on Multiclass UCI Data Sets with a Type (a) Cost Matrix

Data set	BP	over-sampling	under-sampling	threshold-moving	hard-ensemble	soft-ensemble	SMOTE
Cost: Misclassification cost							
<i>lymphography</i>	7.98 ± 4.42	.928 ± .336	1.934 ± 1.277	.894 ± .155	.660 ± .363	.675 ± .378	.732 ± .513
<i>glass</i>	12.18 ± 5.22	1.029 ± .313	1.244 ± .289	.913 ± .058	.875 ± .135	.935 ± .170	1.008 ± .295
<i>waveform</i>	2641.69 ± 539.59	.924 ± .055	.839 ± .094	.842 ± .044	.761 ± .056	.751 ± .057	.837 ± .069
<i>soybean</i>	8.02 ± 1.10	1.212 ± .256	8.042 ± 1.010	.993 ± .026	1.108 ± .125	.885 ± .072	7.886 ± 1.216
<i>annealing</i>	68.17 ± 42.22	1.327 ± .951	2.029 ± 1.598	.814 ± .148	.927 ± .496	.881 ± .442	1.241 ± .869
<i>vowel</i>	46.35 ± 7.46	.917 ± .127	.907 ± .111	.895 ± .080	.761 ± .110	.760 ± .080	.904 ± .171
<i>splice</i>	151.05 ± 55.83	.680 ± .237	.743 ± .241	.664 ± .178	.590 ± .179	.570 ± .170	.646 ± .233
<i>abalone</i>	502.03 ± 89.45	.433 ± .090	.435 ± .089	.431 ± .081	.432 ± .087	.432 ± .087	.435 ± .088
<i>satellite</i>	209.76 ± 79.07	.777 ± .214	.735 ± .160	.794 ± .145	.700 ± .153	.712 ± .162	.766 ± .212
<i>ave.</i>	405.25 ± 853.41	.914 ± .269	1.879 ± 2.375	.804 ± .167	.757 ± .198	.733 ± .161	1.606 ± 2.366
No. HC Errors: Number of high cost errors							
<i>lymphography</i>	.98 ± .67	.761 ± .371	5.654 ± 7.838	.686 ± .304	3.691 ± 5.049	3.169 ± 5.019	3.445 ± 4.525
<i>glass</i>	1.28 ± 1.28	1.200 ± 1.052	1.121 ± .827	.694 ± .179	.656 ± .355	.840 ± .391	1.239 ± 1.203
<i>waveform</i>	318.13 ± 27.95	.892 ± .063	.672 ± .170	.749 ± .046	.666 ± .076	.659 ± .078	.772 ± .091
<i>soybean</i>	.29 ± .43	1.980 ± .960	9.137 ± 3.674	.936 ± .088	.362 ± .248	1.051 ± .295	36.574 ± 29.080
<i>annealing</i>	1.45 ± 7.88	3.229 ± 5.924	6.895 ± 14.669	.501 ± .386	.542 ± .927	.877 ± 1.305	3.757 ± 7.901
<i>vowel</i>	3.41 ± .72	.625 ± .157	.602 ± .175	.628 ± .127	.394 ± .162	.527 ± .128	.650 ± .281
<i>splice</i>	19.40 ± 8.52	.497 ± .288	.480 ± .270	.384 ± .133	.357 ± .194	.339 ± .183	.458 ± .266
<i>abalone</i>	60.04 ± 22.44	.049 ± .102	.046 ± .092	.036 ± .071	.044 ± .091	.045 ± .093	.051 ± .105
<i>satellite</i>	21.03 ± 10.95	.484 ± .318	.457 ± .193	.569 ± .186	.403 ± .152	.452 ± .166	.475 ± .363
<i>ave.</i>	48.33 ± 102.91	1.080 ± .971	2.785 ± 3.459	.576 ± .255	.790 ± 1.104	.884 ± .910	5.269 ± 11.815
No. Errors: Total number of errors							
<i>lymphography</i>	2.88 ± .20	.973 ± .194	3.074 ± .263	1.030 ± .097	.937 ± .131	.931 ± .159	.956 ± .144
<i>glass</i>	7.12 ± .43	1.186 ± .180	1.502 ± .277	1.038 ± .069	1.115 ± .198	1.132 ± .176	1.173 ± .168
<i>waveform</i>	997.15 ± 14.03	1.011 ± .022	1.278 ± .128	1.035 ± .015	.978 ± .028	.971 ± .029	1.003 ± .023
<i>soybean</i>	7.31 ± .70	1.142 ± .178	7.847 ± .778	1.001 ± .021	1.172 ± .110	.881 ± .084	7.609 ± .778
<i>annealing</i>	19.12 ± 1.33	2.552 ± .739	3.457 ± .586	1.035 ± .055	2.293 ± .779	2.005 ± .670	2.437 ± .696
<i>vowel</i>	30.36 ± .81	1.083 ± .051	1.071 ± .039	1.052 ± .031	.956 ± .028	.895 ± .028	1.075 ± .063
<i>splice</i>	53.52 ± 9.10	1.080 ± .143	1.283 ± .163	1.145 ± .113	1.044 ± .123	1.028 ± .109	1.039 ± .152
<i>abalone</i>	186.86 ± 1.79	1.052 ± .044	1.062 ± .059	1.068 ± .063	1.056 ± .052	1.055 ± .050	1.051 ± .041
<i>satellite</i>	98.32 ± 1.83	1.123 ± .112	1.081 ± .157	1.040 ± .027	1.025 ± .079	1.010 ± .060	1.103 ± .124
<i>ave.</i>	155.85 ± 321.04	1.245 ± .495	2.406 ± 2.232	1.049 ± .040	1.175 ± .426	1.101 ± .348	1.938 ± 2.176

The table entries present the real results of BP or the ratio of other methods against that of BP. The values following “±” are standard deviations.

therefore, undersampling has removed so many big class examples that the learning process has been seriously weakened. Maybe SMOTE causes a negative effect because the serious imbalanced class distribution has hampered the generation of synthetic examples. In other words, some synthetic examples generated on the line segments connecting the small class examples may be misleading since the small class examples are surrounded by a large number of big class examples. The poor performance of undersampling may also causes the ineffectiveness of the ensemble methods. Nevertheless, it is noteworthy that threshold-moving and the ensemble methods have not caused a negative effect on this seriously imbalanced data set.

In summary, the observations reported in this section suggest that, on multiclass tasks:

1. Cost-sensitive learning is relatively more difficult than on two-class tasks.
2. A higher degree of class imbalance may increase the difficulty of cost-sensitive learning.
3. The sampling methods and SMOTE are usually ineffective and often cause negative effect, especially on data sets with a big number of classes.
4. Threshold-moving is a good choice which causes relatively fewer negative effects and may be effective on some data sets.
5. Soft-ensemble is also a good choice, which is almost always effective but may cause a negative effect on some seriously imbalanced data sets.

4 DISCUSSION

The empirical study presented in Section 3 reveals that cost-sensitive learning is relatively easy on two-class tasks while hard on multiclass tasks. This is not difficult to understand because an example can be misclassified in more ways in

TABLE 10
Experimental Results on Multiclass Data Sets with a Type (b) Cost Matrix

Data set	BP	over-sampling	under-sampling	threshold-moving	hard-ensemble	soft-ensemble	SMOTE
Cost: Misclassification cost							
<i>lymphography</i>	6.85 ± 2.45	1.008 ± .311	2.160 ± .883	.937 ± .093	.823 ± .373	.774 ± .352	.807 ± .406
<i>glass</i>	28.40 ± 8.35	1.005 ± .137	1.280 ± .238	.938 ± .078	.931 ± .124	.943 ± .121	.979 ± .167
<i>waveform</i>	4084.42 ± 1016.98	.927 ± .046	.933 ± .079	.868 ± .042	.798 ± .055	.787 ± .056	.858 ± .047
<i>soybean</i>	21.00 ± 5.49	1.104 ± .207	7.884 ± 1.745	.970 ± .066	1.108 ± .175	.858 ± .069	7.332 ± 1.448
<i>annealing</i>	93.66 ± 28.65	1.882 ± .793	2.941 ± 1.423	.934 ± .091	1.401 ± .383	1.156 ± .247	1.866 ± .735
<i>vowel</i>	158.91 ± 26.59	.887 ± .095	.919 ± .073	.960 ± .050	.839 ± .086	.791 ± .078	.871 ± .073
<i>splice</i>	217.42 ± 61.54	.763 ± .140	.884 ± .159	.847 ± .107	.696 ± .089	.678 ± .081	.739 ± .143
<i>abalone</i>	701.57 ± 265.53	.672 ± .148	.674 ± .149	.669 ± .145	.671 ± .148	.669 ± .148	.667 ± .146
<i>satellite</i>	486.10 ± 109.84	.851 ± .108	.849 ± .077	.865 ± .086	.790 ± .087	.778 ± .090	.845 ± .116
<i>ave.</i>	644.26 ± 1311.39	1.011 ± .352	2.058 ± 2.309	.888 ± .093	.895 ± .230	.826 ± .149	1.663 ± 2.156
No. HC Errors: Number of high cost errors							
<i>lymphography</i>	1.66 ± .70	1.037 ± .418	2.067 ± 1.906	.875 ± .125	.970 ± .836	.933 ± .847	.881 ± .755
<i>glass</i>	5.56 ± 1.52	1.038 ± .344	1.373 ± .448	.939 ± .186	.974 ± .297	1.011 ± .314	1.013 ± .365
<i>waveform</i>	662.64 ± 30.31	.908 ± .055	.820 ± .109	.844 ± .046	.756 ± .062	.742 ± .064	.829 ± .055
<i>soybean</i>	5.71 ± 1.56	1.145 ± .227	8.535 ± 2.157	.957 ± .108	1.181 ± .251	.861 ± .112	7.905 ± 1.693
<i>annealing</i>	14.85 ± 3.15	2.114 ± 1.238	3.229 ± 1.771	.981 ± .227	1.688 ± 1.057	1.383 ± .936	2.074 ± 1.171
<i>vowel</i>	27.32 ± 1.53	1.097 ± .128	1.137 ± .125	1.136 ± .125	1.066 ± .152	.994 ± .146	1.084 ± .140
<i>splice</i>	36.07 ± 6.02	.735 ± .203	.868 ± .329	.792 ± .160	.664 ± .201	.641 ± .192	.714 ± .191
<i>abalone</i>	115.50 ± 24.84	.654 ± .192	.654 ± .197	.645 ± .185	.651 ± .193	.649 ± .192	.647 ± .190
<i>satellite</i>	80.33 ± 15.80	.932 ± .249	.961 ± .392	.918 ± .150	.872 ± .241	.848 ± .212	.918 ± .229
<i>ave.</i>	105.52 ± 212.43	1.073 ± .423	2.183 ± 2.515	.899 ± .136	.980 ± .320	.896 ± .228	1.785 ± 2.333
No. Errors: Total number of errors							
<i>lymphography</i>	2.71 ± .28	1.063 ± .251	3.182 ± .625	1.065 ± .102	1.086 ± .266	1.003 ± .185	1.060 ± .212
<i>glass</i>	7.27 ± .41	1.331 ± .193	1.676 ± .254	1.131 ± .112	1.292 ± .199	1.317 ± .237	1.288 ± .168
<i>waveform</i>	987.82 ± 14.70	1.031 ± .031	1.446 ± .111	1.051 ± .028	1.020 ± .034	1.011 ± .033	1.013 ± .027
<i>soybean</i>	7.02 ± .94	1.271 ± .301	8.429 ± 1.186	1.050 ± .061	1.406 ± .229	.979 ± .181	7.900 ± 1.087
<i>annealing</i>	17.90 ± .70	2.426 ± .761	3.417 ± .768	1.209 ± .263	2.149 ± .998	1.900 ± 1.029	2.434 ± .705
<i>vowel</i>	30.97 ± 1.49	1.289 ± .128	1.325 ± .112	1.319 ± .100	1.279 ± .125	1.200 ± .108	1.273 ± .116
<i>splice</i>	50.98 ± 8.39	1.252 ± .296	1.511 ± .376	1.306 ± .139	1.228 ± .293	1.196 ± .295	1.224 ± .284
<i>abalone</i>	185.86 ± 1.55	1.098 ± .075	1.105 ± .094	1.097 ± .081	1.100 ± .082	1.098 ± .082	1.092 ± .071
<i>satellite</i>	99.01 ± 2.08	1.314 ± .290	1.379 ± .413	1.196 ± .169	1.234 ± .270	1.199 ± .245	1.254 ± .204
<i>ave.</i>	154.39 ± 318.12	1.342 ± .422	2.608 ± 2.337	1.158 ± .105	1.310 ± .336	1.212 ± .282	2.060 ± 2.232

The table entries present the real results of BP or the ratio of other methods against that of BP. The values following “±” are standard deviations.

multiclass tasks than it might be in two-class tasks, which means the multiclass cost function structure can be more complex to be incorporated in any learning algorithms. Unfortunately, previous research on cost-sensitive learning rarely pays attention to the differences between multiclass and two-class tasks.

Almost at the same time as this paper was written, Abe et al. [1] proposed an algorithm for solving multiclass cost-sensitive learning problems. This algorithm seems inspired by an earlier work of Zadrozny and Elkan [39], where every example is associated with an estimated cost. Since, in multiclass tasks, such a cost is not directly available, the *iterative weighting* and *data space expansion* mechanisms are employed to estimate, for each possible example, an optimal cost (or weight). These mechanisms are then unified in the GBSE (Gradient Boosting with Stochastic Ensembles) framework to use. Note that both the GBSE and our soft-ensemble

method exploit ensemble learning, but the purpose of the former is to make the iterative weighting process feasible while the latter is to combine the goodness of different learning methods. GBSE and soft-ensemble have achieved some success, nevertheless, investigating the nature of multiclass cost-sensitive learning and designing powerful learning methods remain important open problems.

Note that multiclass problems can be converted into a series of binary classification problems, and methods effective in two-class cost-sensitive learning can be used after the conversion. However, this approach might be troublesome when there are many classes and a user usually favors a more direct solution. This is just like that although multiclass classification can be addressed by traditional support vector machines via pairwise coupling, researchers still attempt to design multiclass support vector machines. Nevertheless, it will be an interesting future issue

TABLE 11
Experimental Results on Multiclass UCI Data Sets with a Type (c) Cost Matrix

Data set	BP	over-sampling	under-sampling	threshold-moving	hard-ensemble	soft-ensemble	SMOTE
Cost: Misclassification cost							
<i>lymphography</i>	7.65 ± 2.03	.961 ± .154	3.322 ± .894	1.017 ± .063	.965 ± .147	.910 ± .089	1.008 ± .207
<i>glass</i>	37.12 ± 8.58	1.134 ± .268	1.554 ± .356	.961 ± .106	1.057 ± .217	1.076 ± .191	1.135 ± .192
<i>waveform</i>	4889.81 ± 798.25	.988 ± .047	1.029 ± .044	.972 ± .032	.897 ± .046	.889 ± .044	.970 ± .057
<i>soybean</i>	21.11 ± 2.64	1.215 ± .255	8.334 ± 1.073	1.020 ± .048	1.267 ± .322	.897 ± .128	8.042 ± .815
<i>annealing</i>	105.62 ± 24.26	2.416 ± .707	3.590 ± .935	1.019 ± .090	1.937 ± .374	1.617 ± .269	2.258 ± .648
<i>vowel</i>	171.89 ± 16.85	1.008 ± .085	.991 ± .074	1.010 ± .031	.875 ± .067	.841 ± .046	1.011 ± .040
<i>splice</i>	241.85 ± 82.98	1.007 ± .195	1.068 ± .218	.993 ± .050	.875 ± .165	.860 ± .144	.953 ± .213
<i>abalone</i>	905.32 ± 211.00	.854 ± .299	.859 ± .304	.805 ± .252	.838 ± .288	.831 ± .283	.855 ± .296
<i>satellite</i>	555.35 ± 118.14	.965 ± .176	.945 ± .166	.936 ± .065	.887 ± .139	.880 ± .127	.971 ± .184
<i>ave.</i>	770.64 ± 1572.99	1.172 ± .478	2.410 ± 2.458	.970 ± .069	1.067 ± .353	.978 ± .250	1.911 ± 2.338
No. HC Errors: Number of high cost errors							
<i>lymphography</i>	2.05 ± .47	.968 ± .188	3.523 ± 1.176	1.025 ± .077	.974 ± .162	.904 ± .122	1.054 ± .211
<i>glass</i>	6.59 ± .88	1.200 ± .199	1.612 ± .196	1.040 ± .062	1.155 ± .144	1.166 ± .109	1.227 ± .113
<i>waveform</i>	781.98 ± 76.09	.991 ± .031	1.058 ± .037	.984 ± .017	.914 ± .015	.904 ± .016	.974 ± .036
<i>soybean</i>	6.15 ± .69	1.186 ± .257	8.192 ± .970	1.023 ± .049	1.269 ± .324	.896 ± .124	7.955 ± .695
<i>annealing</i>	17.08 ± 2.64	2.627 ± .681	3.818 ± 1.000	1.031 ± .077	2.151 ± .546	1.736 ± .405	2.325 ± .436
<i>vowel</i>	29.89 ± 1.67	1.029 ± .074	1.005 ± .077	1.020 ± .025	.889 ± .065	.847 ± .051	1.019 ± .048
<i>splice</i>	42.76 ± 6.07	1.085 ± .198	1.174 ± .261	1.022 ± .044	.936 ± .176	.917 ± .157	1.025 ± .215
<i>abalone</i>	155.85 ± 18.97	.820 ± .212	.825 ± .214	.812 ± .211	.810 ± .208	.809 ± .208	.825 ± .214
<i>satellite</i>	94.69 ± 5.93	.968 ± .170	.958 ± .162	.968 ± .060	.901 ± .132	.897 ± .123	.972 ± .181
<i>ave.</i>	126.34 ± 251.05	1.208 ± .545	2.463 ± 2.429	.992 ± .071	1.111 ± .415	1.009 ± .291	1.931 ± 2.303
No. Errors: Total number of errors							
<i>lymphography</i>	2.75 ± .26	.998 ± .105	3.166 ± .396	1.021 ± .037	.984 ± .079	.933 ± .066	1.032 ± .176
<i>glass</i>	6.87 ± .55	1.199 ± .173	1.603 ± .139	1.035 ± .053	1.155 ± .122	1.158 ± .094	1.231 ± .106
<i>waveform</i>	992.05 ± 28.40	.998 ± .028	1.105 ± .060	1.009 ± .008	.937 ± .018	.927 ± .017	.993 ± .034
<i>soybean</i>	6.98 ± .49	1.172 ± .212	8.334 ± .582	1.017 ± .042	1.260 ± .221	.873 ± .092	8.070 ± .564
<i>annealing</i>	18.28 ± 1.63	2.528 ± .477	3.667 ± .391	1.029 ± .079	2.109 ± .544	1.726 ± .404	2.325 ± .436
<i>vowel</i>	31.80 ± .74	1.036 ± .071	1.014 ± .080	1.025 ± .029	.901 ± .062	.856 ± .040	1.023 ± .038
<i>splice</i>	52.21 ± 7.38	1.040 ± .134	1.132 ± .207	1.053 ± .062	.924 ± .125	.903 ± .119	.995 ± .142
<i>abalone</i>	186.72 ± 1.54	1.045 ± .033	1.050 ± .034	1.040 ± .026	1.044 ± .027	1.043 ± .026	1.048 ± .025
<i>satellite</i>	99.48 ± 1.38	1.045 ± .074	1.017 ± .086	.992 ± .031	.957 ± .037	.949 ± .034	1.051 ± .078
<i>ave.</i>	155.24 ± 319.42	1.229 ± .493	2.454 ± 2.423	1.025 ± .018	1.141 ± .382	1.041 ± .273	1.974 ± 2.325

The table entries present the real results of BP or the ratio of other methods against that of BP. The values following “±” are standard deviations.

to compare the effect of doing multiclass cost-sensitive learning directly and the effect of decoupling multiclass problems and then doing two-class cost-sensitive learning.

We found that, although oversampling, undersampling, and SMOTE are known to be effective in addressing the class imbalance problem, they are helpless in cost-sensitive

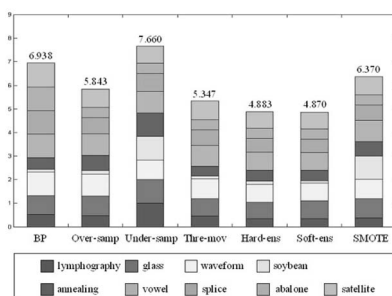


Fig. 3. Robustness of the compared methods on multiclass UCI data sets with a type (a) cost.

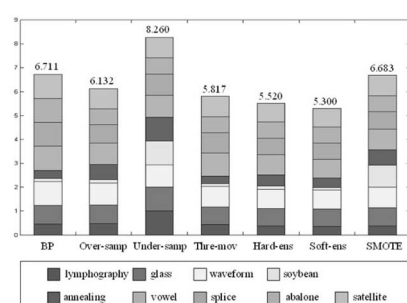


Fig. 4. Robustness of the compared methods on multiclass UCI data sets with a type (b) cost.

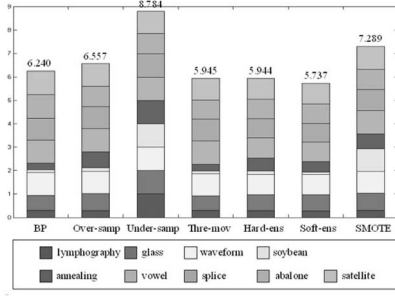


Fig. 5. Robustness of the compared methods on multiclass UCI data sets with a type (c) cost.

learning on multiclass tasks. This may suggest that cost-sensitive learning and learning with imbalanced data sets might have different characteristics. But it should be noted that, although many researchers believed that their conclusions drawn on imbalanced two-class data sets could be applied to multiclass problems [15], in fact, little work has been devoted to the study of imbalanced multiclass data sets. So, there is a big chance that some methods which have been believed to be effective in addressing the class imbalance problem may indeed be only effective on two-class tasks, if the claim “learning from imbalanced data sets and learning when costs are unequal and unknown can be handled in a similar manner” [22] is correct. Whatever the truth is, investigating the class imbalance problem on multiclass tasks is an urgently important issue for future work, which may set the groundwork for developing effective methods

in addressing the class imbalance problem and cost-sensitive learning simultaneously.

It is interesting that, although sampling methods are ineffective in multiclass cost-sensitive learning, ensemble methods utilizing sampling can be effective, sometimes even more effective than threshold-moving. It is well-known that the component learners constituting a good ensemble should be with high diversity as well as high accuracy. In order to explore whether the learners generated by oversampling, undersampling, and threshold-moving are diverse or not, the Q_{av} statistic recommended by Kuncheva and Whitaker [20] is exploited. The formal definition of Q_{av} is shown in (9), where L is the number of component learners, $Q_{i,k}$ is defined as (10), and N^{ab} is the number of examples that have been classified to class a by the i th component learner while classified to class b by the k th component learner. The smaller the value of Q_{av} , the bigger the diversity.

$$Q_{av} = \frac{2}{L(L-1)} \sum_{i=1}^{L-1} \sum_{k=i+1}^L Q_{i,k} \quad (9)$$

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}. \quad (10)$$

Table 13 shows the average Q_{av} values of the learners generated by oversampling, undersampling, and threshold-moving, while the performance of these learners have been presented in Tables 8, 9, 10, 11, and 12.

Table 13 shows that the Q_{av} values on multiclass tasks are apparently smaller than these on two-class tasks, which

TABLE 12
Experimental Results on *KDD-99*

Compared issue	BP	over-sampling	under-sampling	threshold-moving	hard-ensemble	soft-ensemble	SMOTE
Misclassification cost	420.1 ± 305.4	1,228.8 ± 1,546.9	70,761 ± 36,201	386.2 ± 322.1	624.0 ± 365.3	445.4 ± 264.6	617.3 ± 271.9
No. HC Errors	139.8 ± 125.3	560.2 ± 672.9	27,556 ± 21,314	132.2 ± 129.9	295.4 ± 184.0	200.5 ± 133.3	283.1 ± 136.7
No. Errors	184.8 ± 159.4	659.3 ± 874.8	43,200 ± 18,265	176.6 ± 164.5	320.5 ± 182.2	232.7 ± 134.0	327.3 ± 138.4

The table entries present the real results of BP and the studied cost-sensitive learning methods. The values following “±” are standard deviations.

TABLE 13
Average Q_{av} Values of the Learners Generated by Oversampling, Undersampling, and Threshold-Moving

Two-class data set	Q_{av}	Multi-class data set	Q_{av}		
			Cost (a)	Cost (b)	Cost (c)
<i>echocardiogram</i>	.779 ± .135	<i>lymphography</i>	.365 ± .092	.375 ± .170	.308 ± .142
<i>hepatitis</i>	.552 ± .201	<i>glass</i>	.615 ± .108	.615 ± .128	.638 ± .134
<i>heart_s</i>	.774 ± .083	<i>waveform</i>	.815 ± .037	.799 ± .025	.859 ± .023
<i>heart</i>	.790 ± .092	<i>soybean</i>	.518 ± .094	.476 ± .138	.474 ± .141
<i>horse</i>	.826 ± .064	<i>annealing</i>	.019 ± .219	.221 ± .171	-.030 ± .088
<i>credit</i>	.884 ± .038	<i>vowel</i>	.706 ± .040	.792 ± .044	.696 ± .039
<i>breast-w</i>	.805 ± .082	<i>splice</i>	.884 ± .029	.858 ± .056	.849 ± .072
<i>diabetes</i>	.948 ± .028	<i>abalone</i>	.974 ± .052	.993 ± .016	.965 ± .048
<i>german</i>	.774 ± .107	<i>satellite</i>	.936 ± .016	.945 ± .036	.939 ± .010
<i>euthyroid</i>	.902 ± .089	<i>ave.</i>	.648 ± .310	.675 ± .268	.633 ± .331
<i>hypothyroid</i>	.925 ± .074				
<i>coding</i>	.963 ± .036				
<i>ave.</i>	.829 ± .107	<i>KDD99</i>	.245 ± .315		

implies that the learners generated by oversampling, undersampling, and threshold-moving on multiclass tasks are more diverse than these generated on two-class tasks. Therefore, the merits of the component learners can be exerted better on multiclass tasks than on two-class tasks by the ensemble methods. Note that, on *KDD-99*, the Q_{av} value is quite small, but, as it has been reported in Table 12, the performance of the ensemble methods are not very good. This is because, although the learners generated by oversampling, undersampling, and threshold-moving are diverse, the individual performance, especially undersampling, is quite poor. It is obvious that, in order to obtain bigger profit from the ensemble methods, effective mechanisms for encouraging the diversity among the component cost-sensitive learners as well as preserving good individual performance should be designed, which is an interesting issue for future work.

5 CONCLUSION

In this paper, the effect of oversampling, undersampling, threshold-moving, hard-ensemble, soft-ensemble, and SMOTE in training cost-sensitive neural networks are studied empirically on 21 UCI data sets with three types of cost matrices and a real-world cost-sensitive data set. The results suggest that cost-sensitive learning is relatively easy on two-class tasks while it is difficult on multiclass tasks, a higher degree of class imbalance usually results in higher difficulty in cost-sensitive learning, and different types of cost matrices are usually with different difficulties. Both threshold-moving and soft-ensemble are found to be relatively good choices in training cost-sensitive neural networks. The former is a conservative method which rarely causes negative effect, while the latter is an aggressive method which might cause negative effect on seriously imbalanced data sets but its absolute performance is usually better than that of threshold-moving when it is effective. Note that threshold-moving is easier to use than soft-ensemble because the latter requires more computational cost and involves the employment of sampling methods.

The ensembles studied in this paper contain only three component learners. This setting is sufficient for exploring whether or not the combination of sampling and threshold-moving can work, but more benefits should be anticipated from ensemble learning. Specifically, although previous research has shown that using three learners to make an ensemble is already beneficial [27], it is expected that the performance can be improved if more learners are included. A possible extension of current work is to employ each of oversampling, undersampling, and threshold-moving to train multiple neural networks, such as applying these algorithms on different bootstrap samples of the training set, while another possible extension is to exploit more methods, each producing one cost-sensitive neural network. Both are interesting to try in future work.

Section 4 has raised several future issues. Additionally, in most studies on cost-sensitive learning, the cost matrices are usually fixed, while, in some real tasks, the costs might change due to many reasons. Designing effective methods for cost-sensitive learning with variable cost matrices is an interesting issue to be explored in the future.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous reviewers and the associate editor for their constructive comments and suggestions. This work was supported by the National Science Fund for Distinguished Young Scholars of China under Grant No. 60325207, the Jiangsu Science Foundation Key Project under Grant No. BK2004001, and the National 973 Fundamental Research Program of China under Grant No. 2002CB312002.

REFERENCES

- [1] N. Abe, B. Zadrozny, and J. Langford, "An Iterative Method for Multiclass Cost-Sensitive Learning," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 3-11, 2004.
- [2] E. Bauer and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning*, vol. 36, nos. 1-2, pp. 102-139, 1999.
- [3] S.D. Bay, "UCI KDD Archive," Dept. of Information and Computer Science, Univ. of California, Irvine, 2000, <http://kdd.ics.uci.edu/>.
- [4] C. Blake, E. Keogh, and C.J. Merz, "UCI Repository of Machine Learning Databases," Dept. of Information and Computer Science, Univ. of California, Irvine, 1998, <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- [5] J.P. Bradford, C. Kuntz, R. Kohavi, C. Brunk, and C.E. Brodley, "Pruning Decision Trees with Misclassification Costs," *Proc. 10th European Conf. Machine Learning*, pp. 131-136, 1998.
- [6] U. Brefeld, P. Geibel, and F. Wyszotzki, "Support Vector Machines with Example Dependent Costs," *Proc. 14th European Conf. Machine Learning*, pp. 23-34, 2003.
- [7] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*. Belmont, Calif.: Wadsworth, 1984.
- [8] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique," *J. Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002.
- [9] B.V. Dasarathy, *Nearest Neighbor Norms: NN Pattern Classification Techniques*. Los Alamitos, Calif.: IEEE CS Press, 1991.
- [10] T.G. Dietterich, "Ensemble Learning," *The Handbook of Brain Theory and Neural Networks*, second ed., M.A. Arbib, ed., Cambridge, Mass.: MIT Press, 2002.
- [11] P. Domingos, "MetaCost: A General Method for Making Classifiers Cost-Sensitive," *Proc. Fifth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 155-164, 1999.
- [12] C. Drummond and R.C. Holte, "Explicitly Representing Expected Cost: An Alternative to ROC Representation," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 198-207, 2000.
- [13] C. Drummond and R.C. Holte, "C4.5, Class Imbalance, and Cost Sensitivity: Why Under-Sampling Beats Over-Sampling," *Working Notes of the ICML'03 Workshop Learning from Imbalanced Data Sets*, 2003.
- [14] C. Elkan, "The Foundations of Cost-Sensitive Learning," *Proc. 17th Int'l Joint Conf. Artificial Intelligence*, pp. 973-978, 2001.
- [15] N. Japkowicz, "Learning from Imbalanced Data Sets: A Comparison of Various Strategies," *Working Notes of the AAAI'00 Workshop Learning from Imbalanced Data Sets*, pp. 10-15, 2000.
- [16] N. Japkowicz and S. Stephen, "The Class Imbalance Problem: A Systematic Study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429-450, 2002.
- [17] U. Knoll, G. Nakhaeizadeh, and B. Tausend, "Cost-Sensitive Pruning of Decision Trees," *Proc. Eighth European Conf. Machine Learning*, pp. 383-386, 1994.
- [18] M. Kubat and S. Matwin, "Addressing the Curse of Imbalanced Training Sets: One-Sided Selection," *Proc. 14th Int'l Conf. Machine Learning*, pp. 179-186, 1997.
- [19] M. Kukar and I. Kononenko, "Cost-Sensitive Learning with Neural Networks," *Proc. 13th European Conf. Artificial Intelligence*, pp. 445-449, 1998.
- [20] L.I. Kuncheva and C.J. Whitaker, "Measures of Diversity in Classifier Ensembles," *Machine Learning*, vol. 51, no. 2, pp. 181-207, 2003.
- [21] S. Lawrence, I. Burns, A. Back, A.C. Tsoi, and C.L. Giles, "Neural Network Classification and Prior Class Probabilities," *Lecture Notes in Computer Science 1524*, G.B. Orr and K.-R. Müller, eds., pp. 299-313, Berlin: Springer, 1998.

- [22] M.A. Maloof, "Learning When Data Sets are Imbalanced and When Costs are Unequal and Unknown," *Proc. Working Notes ICML'03 Workshop Learning from Imbalanced Data Sets*, 2003.
- [23] D.D. Margineantu and T.G. Dietterich, "Bootstrap Methods for the Cost-Sensitive Evaluation of Classifiers," *Proc. 17th Int'l Conf. Machine Learning*, pp. 583-590, 2000.
- [24] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk, "Reducing Misclassification Costs," *Proc. 11th Int'l Conf. Machine Learning*, pp. 217-225, 1994.
- [25] F. Provost, "Machine Learning from Imbalanced Data Sets 101," *Working Notes AAAI'00 Workshop Learning from Imbalanced Data Sets*, pp. 1-3, 2000.
- [26] F. Provost and T. Fawcett, "Analysis and Visualization of Classifier Performance: Comparison Under Imprecise Class and Cost Distributions," *Proc. Third ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 43-48, 1997.
- [27] J.R. Quinlan, "MiniBoosting Decision Trees," <http://www.cse.unsw.edu.au/~quinlan/miniboost.ps>, 1998.
- [28] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in The Microstructure of Cognition*, D.E. Rumelhart and J.L. McClelland, eds., vol. 1, pp. 318-362, Cambridge, Mass.: MIT Press, 1986.
- [29] *Machine Learning—A Technological Roadmap*, L. Saitta, ed. The Netherlands: Univ. of Amsterdam, 2000.
- [30] C. Stanfill and D. Waltz, "Toward Memory-Based Reasoning," *Comm. ACM*, vol. 29, no. 12, pp. 1213-1228, 1986.
- [31] K.M. Ting, "A Comparative Study of Cost-Sensitive Boosting Algorithms," *Proc. 17th Int'l Conf. Machine Learning*, pp. 983-990, 2000.
- [32] K.M. Ting, "An Empirical Study of MetaCost Using Boosting Algorithm," *Proc. 11th European Conf. Machine Learning*, pp. 413-425, 2000.
- [33] K.M. Ting, "An Instance-Weighting Method to Induce Cost-Sensitive Trees," *IEEE Trans. Knowledge and Data Eng.*, vol. 14, no. 3, pp. 659-665, Apr./May 2002.
- [34] I. Tomek, "Two Modifications of CNN," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 6, no. 6, pp. 769-772, 1976.
- [35] P.D. Turney, "Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm," *J. Artificial Intelligence Research*, vol. 2, pp. 369-409, 1995.
- [36] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, and N. Koudas, "Non-Linear Dimensionality Reduction Techniques for Classification and Visualization," *Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 645-651, 2002.
- [37] G.I. Webb, "Cost-Sensitive Specialization," *Proc. Fourth Pacific Rim Int'l Conf. Artificial Intelligence*, pp. 23-34, 1996.
- [38] G.M. Weiss, "Mining with Rarity—Problems and Solutions: A Unifying Framework," *SIGKDD Explorations*, vol. 6, no. 1, pp. 7-19, 2004.
- [39] B. Zadrozny and C. Elkan, "Learning and Making Decisions When Costs and Probabilities Are Both Unknown," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 204-213, 2001.



Zhi-Hua Zhou (S'00-M'01-SM'06) received the BSc, MSc, and PhD degrees in computer science from Nanjing University, China, in 1996, 1998, and 2000, respectively, all with the highest honor. He joined the Department of Computer Science and Technology of Nanjing University as a lecturer in 2001 and is a professor and head of the LAMDA group at present. His research interests are in machine learning, data mining, pattern recognition, information retrieval, neural computing, and evolutionary computing. In these areas, he has published more than 60 technical papers in refereed international journals or conference proceedings. He has won the Microsoft Fellowship Award (1999), the National Excellent Doctoral Dissertation Award of China (2003), and the Award of the National Science Fund for Distinguished Young Scholars of China (2003). He is an associate editor of *Knowledge and Information Systems* and is on the editorial boards of *Artificial Intelligence in Medicine*, the *International Journal of Data Warehousing and Mining*, the *Journal of Computer Science & Technology*, and the *Journal of Software*. He has served as a program committee member for various international conferences and has chaired a number of native conferences. He is a senior member of China Computer Federation (CCF) and the vice chair of CCF Artificial Intelligence & Pattern Recognition Society, an executive committee member of the Chinese Association of Artificial Intelligence (CAAI), the vice chair and chief secretary of the CAAI Machine Learning Society, and a senior member of the IEEE and IEEE Computer Society.



Xu-Ying Liu received the BSc degree in computer science from Nanjing University of Aeronautics and Astronautics, China, in 2003. Currently, she is a graduate student at the Department of Computer Science & Technology of Nanjing University and is a member of the LAMDA Group. Her research interests are in machine learning and data mining, especially in cost-sensitive and class imbalance learning.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.