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 over-sampling and under-sampling 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 multi-class 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 multi-class 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**

I. 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 much 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 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 effect 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 *over-sampling*, *under-sampling* and *threshold-moving* in training cost-sensitive neural networks. *Hard-ensemble* and *soft-ensemble*, i.e. the combination of over-sampling, under-sampling and threshold-moving via hard or soft voting schemes, 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 multi-class tasks is more difficult than with two-class tasks, and a higher degree of class imbalance may increase the

Manuscript received July 12, 2004; revised April 1, 2005. This work was supported by the the National Science Fund for Distinguished Young Scholars of China under the Grant No. 60325207, the Jiangsu Science Foundation Key Project under the Grant No. BK2004001, and the National 973 Fundamental Research Program of China under the Grant No. 2002CB312002.

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@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.

difficulty. The empirical study also reveals that almost all the techniques are effective on two-class tasks, while most are ineffective on multi-class tasks. Concretely, *sampling* methods are only helpful on two-class tasks, while often cause negative effect on data sets with 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 multi-class tasks; soft-ensemble is effective on both two-class and multi-class 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.

II. 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 $(Cost[i] = Cost[j] \ and \ N_i \geq N_j)$. $Cost[i]$ 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].

A. Over-Sampling

Over-sampling changes the training data distribution such that the costs of the examples are conveyed by the appearances of the examples. In other words, this method duplicates higher-cost training examples until the appearances 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 Eq. 1.

$$N_k^* = \left\lfloor \frac{Cost[k]}{Cost[\lambda]} N_\lambda \right\rfloor \tag{1}$$

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

$$\lambda = \arg\min_j \frac{\frac{Cost[j]}{\min\limits_c Cost[c]} N_{\arg\min\limits_c Cost[c]}}{N_j} \tag{2}$$

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 over-sampling algorithm is summarized in Table I.

TABLE I
The over-sampling 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.

Note that over-sampling 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 over-sampling is effective in learning with imbalanced data sets [15] [16] [22], it should be noted that over-sampling 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 over-sampling is ineffective on the class imbalance problem [13].

Besides the algorithm shown in Table I, this paper also studies a recent variant of over-sampling, 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 over-sampling needed is 200%, 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].

B. Under-Sampling

Like over-sampling, under-sampling 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 under-sampling opposites that of over-sampling 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 Eq. 1. Here the λ-class has the smallest number of training examples to be eliminated, which is identified according to Eq. 3.

$$\lambda = \arg\max_j \frac{\frac{Cost[j]}{\max\limits_c Cost[c]} N_{\arg\max\limits_c Cost[c]}}{N_j} \tag{3}$$

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 shrank, 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 $Dist(\mathbf{x}, \mathbf{y})$ 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 $Dist(\mathbf{x}, \mathbf{z}) < Dist(\mathbf{x}, \mathbf{y})$ or $Dist(\mathbf{y}, \mathbf{z}) < Dist(\mathbf{y}, \mathbf{x})$. Here the distance between two examples are computed according to Eq. 4, where d is the number of attributes among which the first j attributes are binary or nominal.

$$\text{Dist}\left(\mathbf{x_1}, \mathbf{x_2}\right) = \sqrt{\sum_{l=1}^{j} VDM\left(\mathbf{x_{1}}_l, \mathbf{x_{2}}_l\right) + \sum_{l=j+1}^{d} \left|\mathbf{x_{1}}_l - \mathbf{x_{2}}_l\right|^2} \tag{4}$$

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

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

The presented under-sampling algorithm is summarized in Table II.

TABLE II
THE UNDER-SAMPLING 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.
```

Note that under-sampling is also a popular method in addressing the class imbalance problem, which eliminates training examples of the over-sized 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 shown that under-sampling is effective in learning with imbalanced data sets [15] [16] [22], sometimes even stronger than over-sampling, especially on large data sets [13] [15]. Drummond and Holte [13] suggested under-sampling to be a reasonable baseline for algorithmic comparison, but they also indicated that under-sampling introduces non-determinism 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 under-sampling there is also variance due to the non-determinism of the under-sampling process. Since the choice between two classifiers might also depend on the variance, using under-sampling 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, under-sampling via deterministic strategies, such as the one shown in Table II, can be a baseline for comparison.

C. 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 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 $\arg\max_i O_i$, while in threshold-moving the class returned is $\arg\max_i O_i^*$. O_i^* is computed according to Eq. 6, where η is a normalization term such that $\sum_{i=1}^{C} O_i^* = 1$ and $0 \leq O_i^* \leq 1$.

$$O_i^* = \eta \sum_{c=1}^{C} O_i Cost[i, c] \tag{6}$$

The presented threshold-moving algorithm is summarized in Table III, which is similar to the *cost-sensitive classification*

TABLE III

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.
```

TABLE IV

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.
```

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 so popular as sampling methods in addressing the class imbalance problem. Fortunately, recently it has 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].

D. 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 cost-sensitive learners can be trained with the over-sampling, under-sampling 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 over-sampling, under-sampling and threshold-moving algorithms directly to the training set. But it is evident

[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.

TABLE V
UCI DATA SETS USED IN THE EMPIRICAL STUDY (B: BINARY, N: NOMINAL, C: CONTINUOUS)

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

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 IV.

III. EMPIRICAL STUDY

A. 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 II. 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 that on binary or nominal attributes are set to the majority value. Information on these data sets is tabulated in Table V.

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

(a) $1.0 < Cost[i,j] \leq 10.0$ only for a single value of $j = c$ and $Cost[i, j \neq c] = 1.0$ for all $j \neq i$; $Cost[i] = Cost[i,c]$ for $j \neq c$ and $Cost[c] = 1.0$.

(b) $1.0 \leq Cost[i,j] = H_i \leq 10.0$ for each $j \neq i$; $Cost[i] = H_i$. At least one $H_i = 1.0$.

(c) $1.0 \leq Cost[i,j] \leq 10.0$ for all $j \neq i$; $Cost[i] = \sum_{c=1}^{C} Cost[i,c]$. At least one $Cost[i,j] = 1.0$.

Recall that as explained in Section II, 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 VI. 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 multi-class tasks will be reported in separate subsections.

TABLE VI
EXAMPLES OF THREE TYPES OF COST MATRIX, $Cost[i,j]$

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

Under each type of cost matrix, 10 times 10-fold cross validation are performed on each data set except on *waveform* where randomly generated training data size of 300 and test data size of 5000 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 ten 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 ten 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 ten 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

TABLE VII

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

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% training set is used, which consists roughly of 500,000 examples, and further sampled down by random sampling 40% 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 VII. In each experiment, two thirds of the examples in this data set is randomly selected for training while the remaining one third for testing. The experiment is repeated ten times with different training-test partition and the average result is recorded. Since this is a multi-class data set, the experimental results will be reported in the subsection devoting to multi-class tasks.

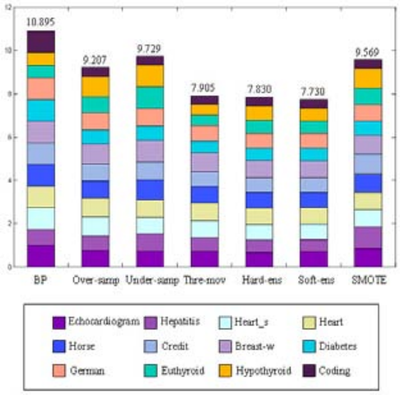


Fig. 1. Robustness of the compared methods on two-class data sets

B. Two-Class Tasks

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

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 Eq. 7.

$$r_\alpha = \frac{cost_\alpha}{\max\limits_i cost_i} \tag{7}$$

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 twelve values of r_α are stacked for the ease of comparison.

Table VIII 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, the worst. Table VIII and Fig. 1 also disclose that the performance of SMOTE is better than that of under-sampling but worse than that of over-sampling. Moreover, the performance of over-sampling, under-sampling, and SMOTE are worse than that of threshold-moving and ensemble methods, 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. *enthyroid* and *hypothyroid*, only threshold-moving is effective, while all the other methods except soft-ensemble on *enthyroid*, cause 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 Eq. 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 negative class, and $p(-)$ and $Cost[-,+]$ are defined similarly. The **y**-axis is 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.

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

The cost curves on the two-class data sets are shown in Fig. 2. On each figure, the curves corresponding to BP, over-sampling, under-sampling, threshold-moving, hard-ensemble,

TABLE VIII

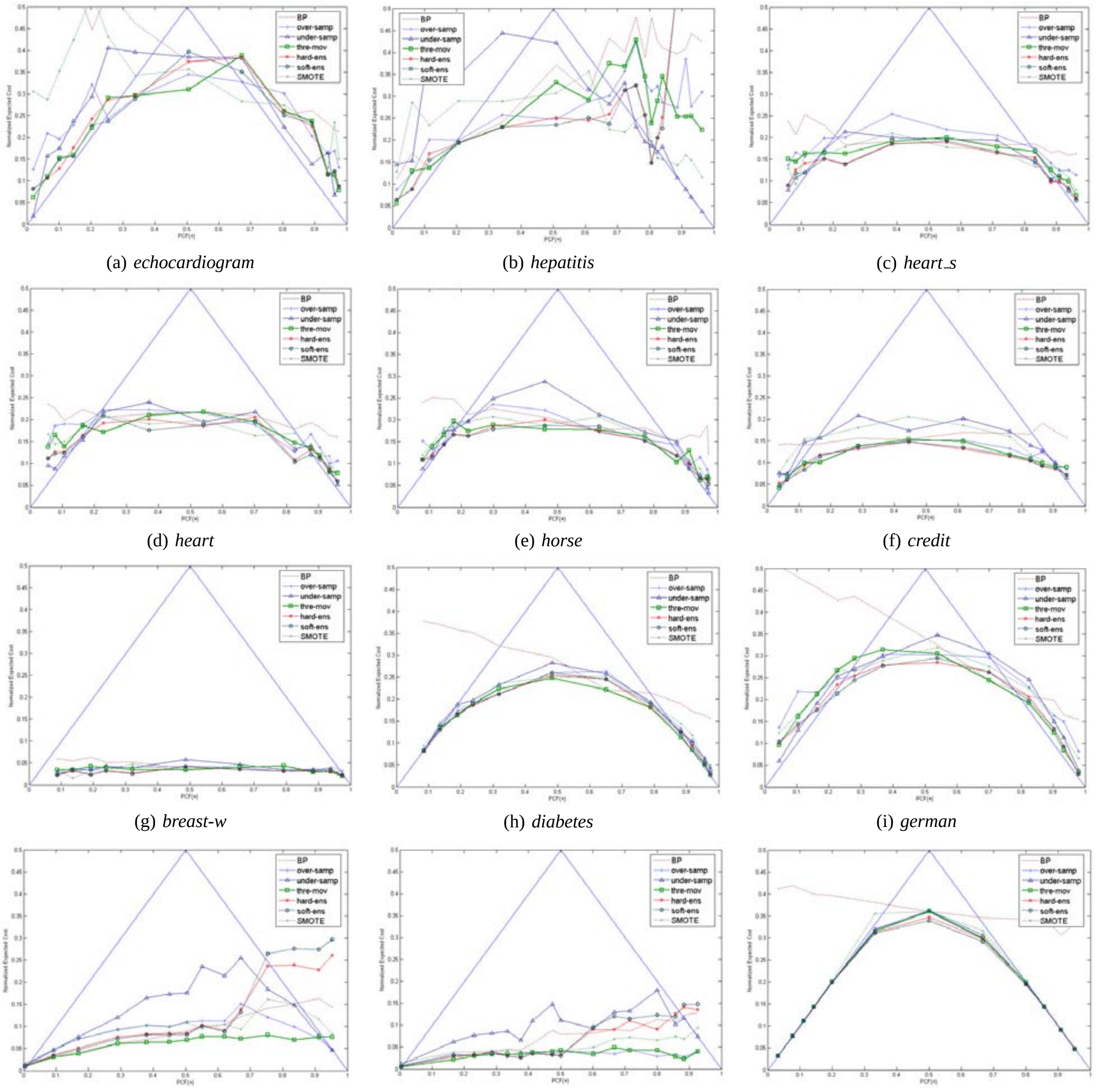
EXPERIMENTAL RESULTS ON TWO-CLASS DATA SETS. THE TABLE ENTRIES PRESENT THE REAL RESULTS OF BP OR THE RATIO OF OTHER METHODS AGAINST THAT OF BP. THE VALUES FOLLOWING '±' ARE STANDARD DEVIATIONS.

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

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 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 VIII.

Fig. 2 exhibits that on *echocardiogram*, under-sampling 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, under-sampling is very bad when $PCF(+)$ is smaller than 0.4, and over-sampling, threshold-moving and ensemble methods are poor when $PCF(+)$ is bigger than 0.85. On *euthyroid*, threshold-moving is the best, under-sampling is the worst in the effective range, while 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 over-sampling, under-sampling, and threshold-moving. On *horse* threshold-moving and the ensemble methods are better than the other methods. On *credit* under-sampling and SMOTE are apparently worse than others. On *breast-w* under-sampling is slightly worse than the other methods. On *diabetes* threshold-moving is the best while under-sampling is the worst. On *german* the ensemble methods are better than others. On *hypothyroid* threshold-moving and over-sampling are better than the other methods while under-sampling 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 under-sampling are often worse than the other methods.

In summary, the observations reported in this subsection suggest that on two-class tasks: 1) Cost-sensitive learning is relatively easy because all methods are effective; 2) Higher



(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*

Fig. 2. Cost curves on two-class data sets

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.

C. Multi-Class Tasks

As shown in Table V, there are nine multi-class 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 IX, X, and XI, respectively. The comparison on the robustness of different methods are shown in Figs. 3 to 5, respectively.

Table IX shows that on multi-class UCI data sets with type (a) cost matrix, the performance of over-sampling, threshold-moving and ensemble methods are apparently better than that of sole BP, while the performance of under-sampling and SMOTE are worse than that of sole BP. Fig. 3 shows that soft-ensemble plays the best, while the robustness of under-sampling is apparently worse than that of sole BP. Table IX and Fig. 3 also show that threshold-moving and soft-ensemble are effective on all data sets, 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 negative effect

TABLE IX

EXPERIMENTAL RESULTS ON MULTI-CLASS UCI DATA SETS WITH TYPE (A) COST MATRIX. 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.

Data set	BP	over-sampling	under-sampling	threshold-moving	hard-ensemble	soft-ensemble	SMOTE
			Cost: Misclassification cost				
lymphography	7.98 ± 4.42	.928 ± .336	1.934 ± 1.277	.894 ± .155	.660 ± .363	.675 ± .378	.732 ± .513
glass	12.18 ± 5.22	1.029 ± .313	1.244 ± .289	.913 ± .058	.875 ± .135	.935 ± .170	1.008 ± .295
waveform	2641.69 ± 539.59	.924 ± .055	.839 ± .094	.842 ± .044	.761 ± .056	.751 ± .057	.837 ± .069
soybean	8.02 ± 1.10	1.212 ± .256	8.042 ± 1.010	.993 ± .026	1.108 ± .125	.885 ± .072	7.886 ± 1.216
annealing	68.17 ± 42.22	1.327 ± .951	2.029 ± 1.598	.814 ± .148	.927 ± .496	.881 ± .442	1.241 ± .869
vowel	46.35 ± 7.46	.917 ± .127	.907 ± .111	.895 ± .080	.761 ± .110	.760 ± .080	.904 ± .171
splice	151.05 ± 55.83	.680 ± .237	.743 ± .241	.664 ± .178	.590 ± .179	.570 ± .170	.646 ± .233
abalone	502.03 ± 89.45	.433 ± .090	.435 ± .089	.431 ± .081	.432 ± .087	.432 ± .087	.435 ± .088
satellite	209.76 ± 79.07	.777 ± .214	.735 ± .160	.794 ± .145	.700 ± .153	.712 ± .162	.766 ± .212
ave.	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				
lymphography	.98 ± .67	.761 ± .371	5.654 ± 7.838	.686 ± .304	3.691 ± 5.049	3.169 ± 5.019	3.445 ± 4.525
glass	1.28 ± 1.28	1.200 ± 1.052	1.121 ± .827	.694 ± .179	.656 ± .355	.840 ± .391	1.239 ± 1.203
waveform	318.13 ± 27.95	.892 ± .063	.672 ± .170	.749 ± .046	.666 ± .076	.659 ± .078	.772 ± .091
soybean	.29 ± .43	1.980 ± .960	9.137 ± 3.674	.936 ± .088	.362 ± .248	1.051 ± .295	36.574 ± 29.080
annealing	1.45 ± 7.88	3.229 ± 5.924	6.895 ± 14.669	.501 ± .386	.542 ± .927	.877 ± 1.305	3.757 ± 7.901
vowel	3.41 ± .72	.625 ± .157	.602 ± .175	.628 ± .127	.394 ± .162	.527 ± .128	.650 ± .281
splice	19.40 ± 8.52	.497 ± .288	.480 ± .270	.384 ± .133	.357 ± .194	.339 ± .183	.458 ± .266
abalone	60.04 ± 22.44	.049 ± .102	.046 ± .092	.036 ± .071	.044 ± .091	.045 ± .093	.051 ± .105
satellite	21.03 ± 10.95	.484 ± .318	.457 ± .193	.569 ± .186	.403 ± .152	.452 ± .166	.475 ± .363
ave.	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				
lymphography	2.88 ± .20	.973 ± .194	3.074 ± .263	1.030 ± .097	.937 ± .131	.931 ± .159	.956 ± .144
glass	7.12 ± .43	1.186 ± .180	1.502 ± .277	1.038 ± .069	1.115 ± .198	1.132 ± .176	1.173 ± .168
waveform	997.15 ± 14.03	1.011 ± .022	1.278 ± .128	1.035 ± .015	.978 ± .028	.971 ± .029	1.003 ± .023
soybean	7.31 ± .70	1.142 ± .178	7.847 ± .778	1.001 ± .021	1.172 ± .110	.881 ± .084	7.609 ± .778
annealing	19.12 ± 1.33	2.552 ± .739	3.457 ± .586	1.035 ± .055	2.293 ± .779	2.005 ± .670	2.437 ± .696
vowel	30.36 ± .81	1.083 ± .051	1.071 ± .039	1.052 ± .031	.956 ± .028	.895 ± .028	1.075 ± .063
splice	53.52 ± 9.10	1.080 ± .143	1.283 ± .163	1.145 ± .113	1.044 ± .123	1.028 ± .109	1.039 ± .152
abalone	186.86 ± 1.79	1.052 ± .044	1.062 ± .059	1.068 ± .063	1.056 ± .052	1.055 ± .050	1.051 ± .041
satellite	98.32 ± 1.83	1.123 ± .112	1.081 ± .157	1.040 ± .027	1.025 ± .079	1.010 ± .060	1.103 ± .124
ave.	155.85 ± 321.04	1.245 ± .495	2.406 ± 2.232	1.049 ± .040	1.175 ± .426	1.101 ± .348	1.938 ± 2.176

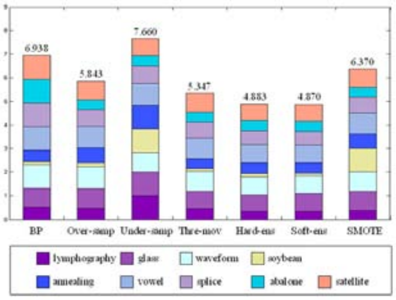


Fig. 3. Robustness of the compared methods on multi-class UCI data sets with type (a) cost

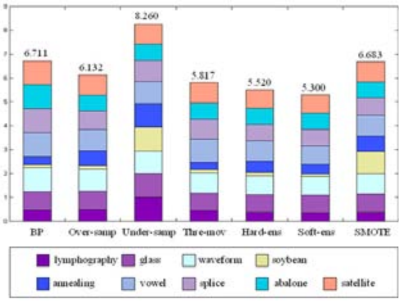


Fig. 4. Robustness of the compared methods on multi-class UCI data sets with type (b) cost

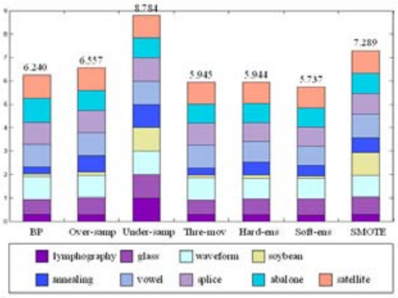


Fig. 5. Robustness of the compared methods on multi-class UCI data sets with type (c) cost

on several data sets suffering from class imbalance, that is, *glass*, *soybean* and *annealing*.

Table X shows that on multi-class 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 under-sampling is apparently worse than that of sole BP. Table X 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 negative effect on *soybean* and *annealing*. It is noteworthy that the sampling methods cause 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 IX and X that all the methods degrade when type (a) cost matrix is replaced with type (b) cost matrix, which suggests that the type (b) cost matrix is more difficult to learn than the type (a) cost matrix.

Table XI shows that on multi-class UCI data sets with 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 under-sampling and SMOTE are even about 2.4 and 1.9

TABLE X

EXPERIMENTAL RESULTS ON MULTI-CLASS DATA SETS WITH TYPE (B) COST MATRIX. 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.

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

times of that of sole BP, respectively. Fig. 5 confirms that soft-ensemble plays the best, while the sampling methods and SMOTE are worse than sole BP. Table XI and Fig. 5 also show that soft-ensemble only causes negative effect on *glass* and the most seriously imbalanced data set *annealing*, hard-ensemble causes negative effect on one more data set, i.e. *soybean*. Threshold-moving does not cause negative effect on *glass*, but it causes negative effect on *lymphography* and *vowel*. The sampling methods and SMOTE cause 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 IX to XI, it can be found that the performance of all the methods degrade much more when type (b) cost matrix is taken over by type (c) matrix than when type (a) cost matrix is taken over by 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 types (b) and (c) cost matrices may be bigger than that between the types (a) and (b) cost matrices.

Table XII 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 over-sampling 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 under-sampling 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 over-sampling, threshold-moving and the ensemble methods do not cause negative effect while under-sampling and SMOTE cause negative effect. The poor performance of under-sampling is not difficult to be expected because on the *KDD-99* data set, the classes are seriously imbalanced therefore under-sampling has removed so many big class examples that the learning process has been seriously weakened. SMOTE causes negative effect may 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 under-sampling may also causes the ineffectiveness of the ensemble methods. Nevertheless, it is noteworthy that threshold-moving and the ensemble methods have not cause negative effect on this seriously imbalanced data set.

In summary, the observations reported in this subsection suggest that on multi-class tasks: 1) Cost-sensitive learning is relatively more difficult than that on two-class tasks; 2) 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

TABLE XI

Experimental results on multi-class UCI data sets with type (c) cost matrix. 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.

Data set	BP	over-sampling	under-sampling	threshold-moving	hard-ensemble	soft-ensemble	Smote
Cost: Misclassification cost							
lymphography	7.65 ± 2.03	.961 ± .154	3.322 ± .894	1.017 ± .063	.965 ± .147	.910 ± .089	1.008 ± .207
glass	37.12 ± 8.58	1.134 ± .268	1.554 ± .356	.961 ± .106	1.057 ± .217	1.076 ± .191	1.135 ± .192
waveform	4889.81 ± 798.25	.988 ± .047	1.029 ± .044	.972 ± .032	.897 ± .046	.889 ± .044	.970 ± .057
soybean	21.11 ± 2.64	1.215 ± .255	8.334 ± 1.073	1.020 ± .048	1.267 ± .322	.897 ± .128	8.042 ± .815
annealing	105.62 ± 24.26	2.416 ± .707	3.590 ± .935	1.019 ± .090	1.937 ± .374	1.617 ± .269	2.258 ± .648
vowel	171.89 ± 16.85	1.008 ± .085	.991 ± .074	1.010 ± .031	.875 ± .067	.841 ± .046	1.011 ± .040
splice	241.85 ± 82.98	1.007 ± .195	1.068 ± .218	.993 ± .050	.875 ± .165	.860 ± .144	.953 ± .213
abalone	905.32 ± 211.00	.854 ± .299	.859 ± .304	.805 ± .252	.838 ± .288	.831 ± .283	.855 ± .296
satellite	555.35 ± 118.14	.965 ± .176	.945 ± .166	.936 ± .065	.887 ± .139	.880 ± .127	.971 ± .184
ave.	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							
lymphography	2.05 ± .47	.968 ± .188	3.523 ± 1.176	1.025 ± .077	.974 ± .162	.904 ± .122	1.054 ± .211
glass	6.59 ± .88	1.200 ± .199	1.612 ± .196	1.040 ± .062	1.155 ± .144	1.166 ± .109	1.227 ± .113
waveform	781.98 ± 76.09	.991 ± .031	1.058 ± .037	.984 ± .017	.914 ± .015	.904 ± .016	.974 ± .036
soybean	6.15 ± .69	1.186 ± .257	8.192 ± .970	1.023 ± .049	1.269 ± .324	.896 ± .124	7.955 ± .695
annealing	17.08 ± 2.64	2.627 ± .681	3.818 ± 1.000	1.031 ± .077	2.151 ± .546	1.736 ± .405	2.325 ± .436
vowel	29.89 ± 1.67	1.029 ± .074	1.005 ± .077	1.020 ± .025	.889 ± .065	.847 ± .051	1.019 ± .048
splice	42.76 ± 6.07	1.085 ± .198	1.174 ± .261	1.022 ± .044	.936 ± .176	.917 ± .157	1.025 ± .215
abalone	155.85 ± 18.97	.820 ± .212	.825 ± .214	.812 ± .211	.810 ± .208	.809 ± .208	.825 ± .214
satellite	94.69 ± 5.93	.968 ± .170	.958 ± .162	.968 ± .060	.901 ± .132	.897 ± .123	.972 ± .181
ave.	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							
lymphography	2.75 ± .26	.998 ± .105	3.166 ± .396	1.021 ± .037	.984 ± .079	.933 ± .066	1.032 ± .176
glass	6.87 ± .55	1.199 ± .173	1.603 ± .139	1.035 ± .053	1.155 ± .122	1.158 ± .094	1.231 ± .106
waveform	992.05 ± 28.40	.998 ± .028	1.105 ± .060	1.009 ± .008	.937 ± .018	.927 ± .017	.993 ± .034
soybean	6.98 ± .49	1.172 ± .212	8.334 ± .582	1.017 ± .042	1.260 ± .221	.873 ± .092	8.070 ± .564
annealing	18.28 ± 1.63	2.528 ± .477	3.667 ± .391	1.029 ± .079	2.109 ± .544	1.726 ± .404	2.325 ± .436
vowel	31.80 ± .74	1.036 ± .071	1.014 ± .080	1.025 ± .029	.901 ± .062	.856 ± .040	1.023 ± .038
splice	52.21 ± 7.38	1.040 ± .134	1.132 ± .207	1.053 ± .062	.924 ± .125	.903 ± .119	.995 ± .142
abalone	186.72 ± 1.54	1.045 ± .033	1.050 ± .034	1.040 ± .026	1.044 ± .027	1.043 ± .026	1.048 ± .025
satellite	99.48 ± 1.38	1.045 ± .074	1.017 ± .086	.992 ± .031	.957 ± .037	.949 ± .034	1.051 ± .078
ave.	155.24 ± 319.42	1.229 ± .493	2.454 ± 2.423	1.025 ± .018	1.141 ± .382	1.041 ± .273	1.974 ± 2.325

TABLE XII

Experimental results on *KDD-99*. The table entries present the real results of BP and the studied cost-sensitive learning methods. The values following '±' are standard deviations.

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

on data sets with a big number of classes; 4) Threshold-moving is a good choice which causes relatively fewer negative effect and may be effective on some data sets; 5) Soft-ensemble is also a good choice, which is almost always effective but may cause negative effect on some seriously imbalanced data sets.

IV. Discussion

The empirical study presented in Section III reveals that cost-sensitive learning is relatively easy on two-class tasks while hard on multi-class tasks. This is not difficult to understand because an example can be misclassified in more ways in multi-class tasks than it might be in two-class tasks, which means the multi-class 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 multi-class and two-class tasks.

Almost as the same time as this paper was written, Abe et al. [1] proposed an algorithm for solving multi-class 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 multi-class 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 multi-class cost-sensitive learning and designing powerful learning methods remain important open problems.

TABLE XIII

AVERAGE Q_{av} VALUES OF THE LEARNERS GENERATED BY OVER-SAMPLING, UNDER-SAMPLING, AND THRESHOLD-MOVING.

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

Note that multi-class 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 user usually favors a more direct solution. This is just like that although multi-class classification can be addressed by traditional support vector machines via pairwise coupling, researchers still attempt to design multi-class support vector machines. Nevertheless, it will be an interesting future issue to compare the effect of doing multi-class cost-sensitive learning directly and the effect of decoupling multi-class problems and then doing two-class cost-sensitive learning.

We found that although over-sampling, under-sampling, and SMOTE are known to be effective in addressing the class imbalance problem, they are helpless in cost-sensitive learning on multi-class 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 multi-class problems [15], in fact few work has been devoted to the study of imbalanced multi-class data sets. So, there are big chances that some methods which have been believed to be effective in addressing the class imbalance problem may be indeed 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 multi-class tasks is an urgently important issue for future work, which may set ground 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 multi-class 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 over-sampling, under-sampling, 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 Eq. 9, where L is the number of component learners, $Q_{i,k}$ is defined as Eq. 10, 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.

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

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

Table XIII shows the average Q_{av} values of the learners generated by over-sampling, under-sampling, and threshold-moving, while the performance of these learners have been presented in Tables VIII, IX, X, XI, and XII.

Table XIII shows that the Q_{av} values on multi-class tasks are apparently smaller than these on two-class tasks, which implies that the learners generated by over-sampling, under-sampling, and threshold-moving on multi-class tasks are more diverse than these generated on two-class tasks. Therefore, the merits of the component learners can be exerted better on multi-class 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 XII, the performance of the ensemble methods are not very good. This is because although the learners generated by over-sampling, under-sampling, and threshold-moving are diverse, the individual performance, especially under-sampling, 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.

V. CONCLUSION

In this paper, the effect of over-sampling, under-sampling, threshold-moving, hard-ensemble, soft-ensemble, and SMOTE in training cost-sensitive neural networks are studied empirically on twenty-one 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 difficult on multi-class tasks, a higher degree of class imbalance usually results in bigger 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 over-sampling, under-sampling 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 IV has raised several future issues. Besides, 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.

ACKNOWLEDGEMENT

The authors wish to thank the anonymous reviewers and the associate editor for their constructive comments and suggestions.

REFERENCES

[1] N. Abe, B. Zadrozny, and J. Langford, “An iterative method for multi-class cost-sensitive learning,” in Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, 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, no.1–2, pp.102–139, 1999.

[3] S.D. Bay, “UCI KDD archive” [http://kdd.ics.uci.edu/], Department of Information and Computer Science, University of California, Irvine, CA, 2000.

[4] C. Blake, E. Keogh, and C.J. Merz, “UCI repository of machine learning databases” [http://www.ics.uci.edu/∼mlearn/MLRepository.html], Department of Information and Computer Science, University of California, Irvine, CA, 1998.

[5] J.P. Bradford, C. Kuntz, R. Kohavi, C. Brunk, and C.E. Brodley, “Pruning decision trees with misclassification costs,” in Proceedings of the 10th European Conference on Machine Learning, Chemnitz, Germany, pp.131–136, 1998.

[6] U. Brefeld, P. Geibel, and F. Wysotzki, “Support vector machines with example dependent costs,” in Proceedings of the 14th European Conference on Machine Learning, Cavtat-Dubrovnik, Croatia, pp.23–34, 2003.

[7] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, Classification and Regression Trees, Belmont, CA: Wadsworth, 1984.

[8] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” Journal of Artificial Intelligence Research, vol.16, pp.321–357, 2002.

[9] B.V. Dasarathy, Nearest Neighbor Norms: NN Pattern Classification Techniques, Los Alamitos, CA: IEEE Computer Society Press, 1991.

[10] T.G. Dietterich, “Ensemble learning,” in The Handbook of Brain Theory and Neural Networks, 2nd edition, M.A. Arbib, Ed. Cambridge, MA: MIT Press, 2002.

[11] P. Domingos, “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, CA, pp.155–164, 1999.

[12] C. Drummond and R.C. Holte, “Explicitly representing expected cost: an alternative to ROC representation,” in Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, pp.198–207, 2000.

[13] C. Drummond and R.C. Holte, “C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling,” in Working Notes of the ICML’03 Workshop on Learning from Imbalanced Data Sets, Washington, DC, 2003.

[14] C. Elkan, “The foundations of cost-senstive learning,” in Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, WA, pp.973–978, 2001.

[15] N. Japkowicz, “Learning from imbalanced data sets: a comparison of various strategies,” in Working Notes of the AAAI’00 Workshop on Learning from Imbalanced Data Sets, Austin, TX, 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,” in Proceedings of the 8th European Conference on Machine Learning, Catania, Italy, pp.383–386, 1994.

[18] M. Kubat and S. Matwin, “Addressing the curse of imbalanced training sets: one-sided selection,” in Proceedings of the 14th International Conference on Machine Learning, Nashville, TN, pp.179–186, 1997.

[19] M. Kukar and I. Kononenko, “Cost-sensitive learning with neural networks,” in Proceedings of the 13th European Conference on Artificial Intelligence, Brighton, UK, 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,” in Lecture Notes in Computer Science 1524, G.B. Orr and K.-R. Müller, Eds. Berlin: Springer, pp.299–313, 1998.

[22] M.A. Maloof, “Learning when data sets are imbalanced and when costs are unequal and unknown,” in Working Notes of the ICML’03 Workshop on Learning from Imbalanced Data Sets, Washington, DC, 2003.

[23] D.D. Margineantu and T.G. Dietterich, “Bootstrap methods for the cost-sensitive evaluation of classifiers,” in Proceedings of the 17th International Conference on Machine Learning, San Francisco, CA, pp.583–590, 2000.

[24] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk, “Reducing misclassification costs,” in Proceedings of the 11th International Conference on Machine Learning, New Brunswick, NJ, pp.217–225, 1994.

[25] F. Provost, “Machine learning from imbalanced data sets 101,” in Working Notes of the AAAI’00 Workshop on Learning from Imbalanced Data Sets, Austin, TX, pp.1–3, 2000.

[26] F. Provost and T. Fawcett, “Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions,” in Proceedings of the 3rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, 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,” in Parallel Distributed Processing: Explorations in The Microstructure of Cognition, D.E. Rumelhart and J.L. McClelland, Eds. Cambridge, MA: MIT Press, vol.1, pp.318–362, 1986.

[29] L. Saitta, Ed. Machine Learning - A Technological Roadmap, The Netherland: University of Amsterdam, 2000.

[30] C. Stanfill and D. Waltz, “Toward memory-based reasoning,” Communications of the ACM, vol.29, no.12, pp.1213–1228, 1986.

[31] K.M. Ting, “A comparative study of cost-sensitive boosting algorithms,” in Proceedings of the 17th International Conference on Machine Learning, San Francisco, CA, pp.983–990, 2000.

[32] K.M. Ting, “An empirical study of MetaCost using boosting algorithm,” in Proceedings of the 11th European Conference on Machine Learning, Barcelona, Spain, pp.413–425, 2000.

[33] K.M. Ting, “An instance-weighting method to induce cost-sensitive trees,” IEEE Transactions on Knowledge and Data Engineering, vol.14, no.3, pp.659–665, 2002.

[34] I. Tomek, “Two modifications of CNN,” IEEE Transactions on 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, ” Journal of 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,” in Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, pp.645–651, 2002.

[37] G.I. Webb, “Cost-sensitive specialization,” in Proceedings of the 4th Pacific Rim International Conference on Artificial Intelligence, Cairns, Australia, 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,” in Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, 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 & 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 over 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 National Science Fund for Distinguished Young Scholars of China (2003). He is an associate editor of *Knowledge and Information Systems*, and on the editorial boards of *Artificial Intelligence in Medicine*, *International Journal of Data Warehousing and Mining*, *Journal of Computer Science & Technology*, and *Journal of Software*. He served as program committee member for various international conferences and 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 Chinese Association of Artificial Intelligence (CAAI), the vice chair and chief secretary of CAAI Machine Learning Society, and a senior member of IEEE and IEEE Computer Society.

Xu-Ying Liu received her 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 .