Soft Methodology for Cost-and-error Sensitive Classification

Te-Kang Jan, Da-Wei Wang, Chi-Hung Lin and Hsuan-Tien Lin

Abstract—Many real-world data mining applications need varying cost for different types of classification errors and thus call for cost-sensitive classification algorithms. Existing algorithms for cost-sensitive classification are successful in terms of minimizing the cost, but can result in a high error rate as the trade-off. The high error rate holds back the practical use of those algorithms. In this paper, we propose a novel cost-sensitive classification methodology that takes both the cost and the error rate into account. The methodology, called soft cost-sensitive classification, is established from a multicriteria optimization problem of the cost and the error rate, and can be viewed as regularizing cost-sensitive classification with the error rate. The simple methodology allows immediate improvements of existing cost-sensitive classification algorithms. Experiments on the benchmark and the real-world data sets show that our proposed methodology indeed achieves lower test error rates and similar (sometimes lower) test costs than existing cost-sensitive classification algorithms. We also demonstrate that the methodology can be extended for considering the weighted error rate instead of the original error rate. This extension is useful for tackling unbalanced classification problems.

Index Terms—Classification, Cost-sensitive learning, Multicriteria optimization, Regularization

1 Introduction

Classification is important for machine learning and data mining [1], [2]. Traditionally, the regular classification problem aims at minimizing the rate of misclassification errors. In many real-world applications, however, different types of errors are often charged with different costs. For instance, in bacteria classification, mis-classifying a Gram-positive species as a Gram-negative one leads to totally ineffective treatments and is hence more serious than mis-classifying a Gram-positive species as another Gram-positive one [3], [4]. Similar application needs are shared by targeted marketing, information retrieval, medical decision making, object recognition and intrusion detection [5]–[10], and can be formalized as the cost-sensitive classification problem. In fact, cost-sensitive classification can be used to express any finite-choice and bounded-loss supervised learning problems [11]. Thus, it has been attracting much research attention in recent years, in terms of both new algorithms and new applications [3], [7], [12]–[16].

Studies in cost-sensitive classification often reveal a trade-off between cost and error rate [13], [15], [16]. Mature regular classification algorithms can achieve significantly lower error rate than their cost-sensitive counterparts, but result in higher expected cost; state-of-the-art cost-sensitive classification algorithms can reach significantly lower expected cost than their regular classification counterparts, but are often at the expense of higher error rate. In addition, cost-sensitive classification algorithms are "sensitive" to large cost components and can thus be conservative or even "paranoid" in order to avoid making any big mistakes. The sensitivity makes cost-sensitive classification algorithms prone to overfitting the data or the cost. In fact, it has been observed that for some simpler classification tasks, cost-sensitive classification algorithms are inferior to regular classification ones in terms of even the expected test cost because of the overfitting [13], [15].

The expense of high error rate and the potential risk of overfitting holds back the practical use of cost-sensitive classification algorithms. Arguably, applications call for classifiers that can reach low cost *and* low error rate. The problem of obtaining such a classifier has been studied for binary cost-sensitive classification [17], but the more general problem for multiclass cost-sensitive classification is yet to be tackled.

In this paper, we propose a methodology to tackle the problem. The methodology takes both the cost and the error rate into account and matches the realistic needs better. We name the methodology *soft cost-sensitive classification* to distinguish it from existing *hard cost-sensitive classification* algorithms that focus on only the cost. The methodology is designed by formulating the associated problem as a multicriteria optimization task [18]: one criterion being the cost and the other being the error rate. Then, the methodology solves the task by the weighted sum approach for multicriteria optimization [19]. The simplicity of the weighted sum approach allows immediate reuse of modern cost-sensitive classification algorithms as the core tool. In other words, with our proposed methodology, promising (hard) cost-sensitive classification algorithms can be immediately improved via soft cost-sensitive classification, with performance guarantees on cost and error rate supported by the theory behind multicriteria optimization.

Error rate, however, is sometimes not the basic

criterion of interest. For instance, many cost-sensitive classification data sets in the real world are also unbalanced, such as the intrusion detection data set in KDD Cup 1999 [20]. For such an unbalanced data set, the error rate favors only the majority classes and is thus less meaningful in assessing the quality of classification results. Then, the weighted error rate that balances the influence of each class can be more meaningful. We extend the proposed methodology to consider the weighted error rate instead of the error rate. The extended methodology can then be used to improve the performance of cost-sensitive classification algorithms for unbalanced classification problems.

We conduct a complete comparison to validate the performance of the proposed methodology. The comparison involves not only twenty-two benchmark and two real-world data sets, but also uses four state-of-the-art (hard) cost-sensitive classification algorithms as well as their soft siblings. To the best of our knowledge, the comparison is the most extensive empirical study on multiclass cost-sensitive classification in terms of the numbers of data sets and algorithms. Experimental results suggest that soft cost-sensitive classification can indeed achieve both low cost and low error rate. In particular, soft cost-sensitive classification algorithms out-perform regular ones in terms of the test cost on most of the data sets. In addition, soft cost-sensitive classification algorithms reach significantly lower test error rate than their hard siblings, while achieving similar (sometimes better) test cost. The observations are consistent across three different sets of tasks: the traditional benchmark tasks in cost-sensitive classification [22], new benchmark tasks designed for examining the effect of using large cost components, and the real-world medical task for classifying bacteria [3].

We also conduct experiments on unbalanced classification tasks for validating the extended methodology. The unbalanced data sets include not only the benchmark data sets but also a real-world task, the KDD 1999 data set on intrusion detection [20]. The results justify that soft cost-sensitive classification can consider cost and weighted error rate jointly to reach better performance.

The paper is organized as follows. We formally introduce the regular and the cost-sensitive classification problems in Section 2, and discuss related works on cost-sensitive classification. Then, we present the proposed methodology of soft cost-sensitive classification in Section 3. We discuss the empirical performance of the proposed methodology on the benchmark and the real-world data sets in Section 4. Finally, we conclude in Section 5.

A short version of the paper appeared in 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining [23]. The paper is then enriched by

1) introducing another state-of-the-art cost-sensitive classification algorithm [21] in Section 2, and including it in the experimental comparison in Section 4 with two types of different costs that are added for making a fair comparison with this algorithm;
2) extending the proposed methodology to take both weighted error rate and cost into account in Section 3, and validate its performance in Section 4;
3) studying the issue of parameter selection for soft cost-sensitive classification substantially in Section 4.

2 COST-SENSITIVE CLASSIFICATION

We shall start by defining the regular classification problem and then extend it to the cost-sensitive one. Then, we briefly review existing works on cost-sensitive classification.

In the regular classification problem, we are given a training set $\mathcal{S} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, where the input vector $\mathbf{x}_n$ belongs to some domain $\mathcal{X} \subseteq \mathbb{R}^D$, the label y_n comes from the set $\mathcal{Y} = \{1, \ldots, K\}$ and each example $(\mathbf{x}_n, y_n)$ is drawn independently from an unknown distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$. The task of regular classification is to use the training set $\mathcal{S}$ to find a classifier $g\colon \mathcal{X} \to \mathcal{Y}$ such that the expected error rate $E(g) = \underset{(\mathbf{x},y)\sim\mathcal{D}}{\mathcal{E}} [\![y \neq g(\mathbf{x})]\!]$ is small,[1] where the expected error rate $E(g)$ penalizes every type of mis-classification error equally.

Cost-sensitive classification extends regular classification by charging different cost for different types of classification errors. We adopt the example-dependent setting of cost-sensitive classification, which is rather general and can be used to express other popular settings [12], [13], [15], [16], [24]. The example-dependent setting couples each example $(\mathbf{x}, y)$ with a cost vector $\mathbf{c} \in [0, \infty)^K$, where the k-th component of $\mathbf{c}$ quantifies the cost for predicting the example $\mathbf{x}$ as class k. The cost $\mathbf{c}[y]$ of the intended class y is naturally assumed to be 0, the minimum cost. Consider a cost-sensitive training set $\mathcal{S}_c = \{(\mathbf{x}_n, y_n, \mathbf{c}_n)\}_{n=1}^N$, where each cost-sensitive training example $(\mathbf{x}_n, y_n, \mathbf{c}_n)$ is drawn independently from an unknown cost-sensitive distribution $\mathcal{D}_c$ on $\mathcal{X} \times \mathcal{Y} \times [0, \infty)^K$, the task of cost-sensitive classification is to use $\mathcal{S}_c$ to find a classifier $g\colon \mathcal{X} \to \mathcal{Y}$ such that the expected cost $E_c(g) = \underset{(\mathbf{x},y,\mathbf{c})\sim\mathcal{D}_c}{\mathcal{E}} \mathbf{c}[g(\mathbf{x})]$ is small.

One special case of the example-dependent setting is the class-dependent setting, in which the cost vectors $\mathbf{c}$ are taken from the y-th row of a cost matrix $\mathbf{C}\colon \mathcal{Y} \times \mathcal{Y} \to [0, \infty)^K$. Each entry $\mathbf{C}(y, k)$ of the cost matrix represents the cost for predicting a

1. The Boolean operation $[\![\cdot]\!]$ is 1 when the argument is true and 0 otherwise.

class-y example as class k. The special case is commonly used in some applications and some benchmark experiments [3], [13], [16].

Regular classification can be viewed as a special case of the class-dependent setting, which is in term a special case of the example-dependent setting. In particular, take a cost matrix that contains 0 in the diagonals and 1 elsewhere, which equivalently corresponds to the regular cost vectors $\bar{\mathbf{c}}_y$ with entries $\bar{\mathbf{c}}_y[k] = [\![y \neq k]\!]$. Then, the expected cost $E_c(g)$ with respect to $\{\bar{\mathbf{c}}_y\}$ is the same as the expected error rate $E(g)$. In other words, regular classification algorithms can be viewed as "wiping out" the given cost information and replacing it with a naïve cost matrix. Intuitively, such algorithms may not work well for cost-sensitive classification because of the wiping out.

Another special case of the class-dependent setting considers a cost matrix where row y equals $w_y \cdot \bar{\mathbf{c}}_y$, with some weight $w_y \geq 0$ for each y. The weights can be used to adjust the influence of each class, and are widely used when solving unbalanced classification problems. This special case is commonly named weighted classification.

Existing cost-sensitive classification algorithms can be grouped to two categories: the binary ($K = 2$) cases and the multiclass ($K > 2$) cases. Binary cost-sensitive classification is well-understood in theory and in practice. In particular, every binary cost-sensitive classification problem can be reduced to a binary regular classification one by re-weighting the examples based on the cost [25], [26]. Multiclass cost-sensitive classification, however, is more difficult than the binary one, and is an ongoing research topic.

MetaCost [22] is one of the earliest multiclass cost-sensitive classification algorithms and it can only be applied to the class-dependent setting. MetaCost makes any regular classification algorithm cost-sensitive by re-labeling the training examples. Somehow the re-labeling procedure depends on an overly-ideal assumption, which makes it hard to rigorously analyze the performance of MetaCost in theory. Many other early approaches suffer from similar shortcomings [27].

In order to design multiclass cost-sensitive classification algorithms with stronger theoretical guarantees, modern cost-sensitive classification algorithms are mostly reduction-based, which allows not only reusing mature existing algorithms for cost-sensitive classification, but also extending existing theoretical results to the area of cost-sensitive classification. For instance, [10] reduces the multiclass cost-sensitive classification problem into several multiclass weighted classification problems using a boosting-style method and some intermediate traditional classifiers. The reduction is somehow too sophisticated for practical use.

Zhou and Liu proposed another reduction approach (CSZL; [21]) from multiclass cost-sensitive classification to multiclass weighted classification based on re-weighting with the solution to a linear system. The CSZL approach can only work in the class-dependent setting. When the cost matrix is consistent (i.e. coefficient matrix of the linear system is not of full rank), CSZL comes with sound theoretical guarantees for choosing the the weights, and then plugs these weights into some weighted classification algorithm as an internal learner; otherwise, CSZL decomposes the multiclass cost-sensitive classification problem into several binary cost-sensitive classification problems based on pairwise comparisons of the classes to get an approximate solution [21].

There are quite a few other studies on reducing multiclass cost-sensitive classification to binary cost-sensitive classification by decomposing the multiclass problem with a suitable structure and embedding the cost vectors into the weights in those binary classification problems. For instance, cost-sensitive one-versus-one (CSOVO; [13]) and weighted all-pair (WAP; [11]) are also based on pairwise comparisons of the classes. Another leading approach within the family is cost-sensitive filter tree (CSFT; [12]), which is based on a single-elimination tournament of competing classes.

Yet another family of approaches reduce the multiclass cost-sensitive classification problem into regression ones by embedding the cost vectors in the real-valued labels instead of the weights [28]. A promising representative of the family is to reduce to one-sided regression (OSR; [15]).

Based on some earlier comparisons on general benchmark data sets [15], [16], OSR, CSOVO and CSFT are some of the leading algorithms that can reach state-of-the-art performance. Each algorithm corresponds to a popular sibling for regular classification. In particular, the common one-versus-all decomposition (OVA) [29] is the special case of OSR, the one-versus-one decomposition (OVO) [29] is the special case of CSOVO, and the modern filter tree decomposition (FT) [12] is the special case of CSFT. The regular classification algorithms, OVA, OVO and FT, do not consider any cost during their training. On the other hand, the cost-sensitive ones, OSR, CSOVO and CSFT, respect the cost faithfully during their training.

Note that the regular classification sibling for CSZL is not as explicit as the other cost-sensitive classification algorithms. When the cost matrix consists of $\{\bar{\mathbf{c}}_y\}$, the cost is consistent for CSZL and its corresponding linear system can be solved by setting all classes to be of equal weights. Thus, the regular classification sibling of CSZL is the regular classification sibling of its internal learner. Because CSZL takes one-versus-one decomposition for the inconsistent cost, we consider (weighted) OVO as the internal learner for CSZL for the consistent cost in this work. Hence the regular classification sibling of CSZL is simply OVO.

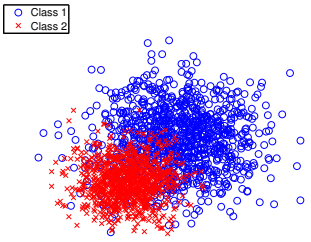


Fig. 1. a two-dimensional artificial data set

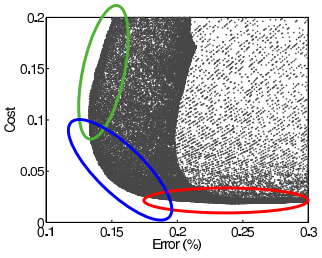


Fig. 2. the different goals of regular (green), cost-sensitive (red) and soft cost-sensitive (blue) classification algorithms

3 Soft Cost-sensitive Classification

The difference between regular and cost-sensitive classification is illustrated with a binary and two-dimensional artificial data set shown in Figure 1. Class 1 is generated from a Gaussian distribution of standard deviation $\frac{4}{5}$; class 2 is generated from a Gaussian distribution of standard deviation $\frac{1}{2}$; the centers of the two classes are of $\sqrt{2}$ apart. We consider a cost matrix of $\begin{bmatrix} 0 & 1 \\ 30 & 0 \end{bmatrix}$. Then, we enumerate many linear classifiers in $\mathbb{R}^2$ and evaluate their average error and average cost. The results are plotted in Figure 2. Each black point represents the achieved (error, cost) of one linear classifier.[2] We can see that there is a region of low-cost linear classifiers, as circled in red. There is also a region of low-error linear classifiers, as circled in green. Modern cost-sensitive classification algorithms are designed to seek for *something* in the red region, which contains classifiers with a wide range of different errors. Traditional regular classification algorithms, on the other hand, are designed to locate something in the green region (without using the cost information), which is far from the lowest achievable cost. In other words, there is a trade-off between the cost and the error, while cost-sensitive and regular classification each takes the trade-off to the extreme.

Many real-world applications, however, do not need the extreme classifiers in the red and green regions, but call for classifiers with both low cost *and* low error rate as depicted in the blue region in Figure 2. In particular, the applications take the cost to be the subjective measure of performance and the error to be the objective safety-check as the basic criterion. The blue region improves the green one (regular) by taking the cost into account; the blue region also improves the red one (cost-sensitive) by keeping the error under control. The three regions, as depicted, are not meant to be disjoint. The blue region may contain the better cost-sensitive classifiers in its intersection with the green region, and the better regular classifiers in its intersection with the red region.

Figure 2 results from a simple artificial data set for the illustrative purpose. When applying more sophisticated classifiers on real-world data sets, the set of achievable (error, cost) may be of a more complicated shape—possibly non-convex, for instance. Somehow the essence of the problem remains the same: cost-sensitive classification only knocks down the cost and results in a red region at the bottom; regular classification only considers the error and lands on a green region at the left; our proposed methodology focuses on a blue region at the left-bottom, hopefully achieving the better for both criteria.

Formally speaking, regular classification algorithm is a process from $\mathcal{S}$ to g such that $E(g)$ is small. Cost-sensitive classification algorithm, on the other hand, is a process from $\mathcal{S}_c$ to g such that $E_c(g)$ is small. We now want a process from $\mathcal{S}_c$ to g such that both $E(g)$ and $E_c(g)$ are small, which can be written as

$$\min_g \mathbf{E}(g) = [E_c(g), E(g)] \text{ subject to all feasible } g. \quad (1)$$

The vector $\mathbf{E}$ represents the two criteria of interest.

Such a problem belongs to multicriteria optimization [18], which deals with multiple objective functions. The general form of multicriteria optimization is

$$\min_g \mathbf{F}(g) = [F_1(g), F_2(g), \ldots, F_M(g)] \quad \text{subject to all feasible } g, \quad (2)$$

where M is the number of criteria. For a multicriteria optimization problem (2), often there is no global optimal solution g^* that is the best in terms of every dimension (criterion) within $\mathbf{F}$. Instead, the goal of (2) is to seek for the set of "better" solutions, usually referred to as the Pareto-optimal front [30]. Formally speaking, consider two feasible candidates g_1 and g_2. The candidate g_1 is said to *dominate* g_2 if $F_m(g_1) \leq F_m(g_2)$ for all m while $F_i(g_1) < F_i(g_2)$ for some i. The Pareto-optimal front is the set of all non-dominated solutions [18].

Solving the multicriteria optimization problem is not an easy task, and there are many sophisticated techniques, including evolutionary algorithms like Non-dominated Sorting Genetic Algorithms [31] and Strength Pareto Evolutionary Algorithms [32]. One

2. Ideally, the points should be dense. The uncrowded part comes from simulating with a finite enumeration process.

important family of techniques is to transform the problem to a single-criterion optimization one that we are more familiar with. A simple yet popular approach of the family considers a non-negative linear combination of all the criteria F_m, which is called the weighted sum approach [19]. In particular, the weighted sum approach solves the following optimization problem:

$$\min_g \sum_{m=1}^{M} \alpha_m F_m(g) \text{ subject to all feasible } g, \tag{3}$$

where $\alpha_m \geq 0$ is the weight (importance) of the m-th criterion. By varying the values of α_m, the weighted sum approach identifies *some* of the solutions that are on the tangential of the Pareto-optimal front [18]. The drawback of the approach [33] is that *not all* the solutions within the Pareto-optimal front can be found when the achievable set of $\mathbf{F}(g)$ is non-convex.

We can reach the goal of getting a low-cost and low-error classifier by formulating a multicriteria optimization problem with $M = 2$, $F_1(g) = E_c(g)$ and $F_2(g) = E(g)$. Without loss of generality, let $\alpha_1 = 1 - \alpha$ and $\alpha_2 = \alpha$ for $\alpha \in [0, 1]$, the weighted sum approach solves

$$\min_g (1 - \alpha) E_c(g) + \alpha E(g), \tag{4}$$

which is the same as

$$\min_g \mathop{\mathcal{E}}_{(\mathbf{x},y,\mathbf{c})\sim\mathcal{D}_c} (1 - \alpha)\Big(\mathbf{c}[g(\mathbf{x})]\Big) + \alpha\Big(\bar{\mathbf{c}}_y[g(\mathbf{x})]\Big) \tag{5}$$

with the regular cost vectors $\bar{\mathbf{c}}_y$ defined in Section 2. For any given α, such an optimization problem is exactly a cost-sensitive classification one with modified cost vectors $\tilde{\mathbf{c}} = (1 - \alpha)\mathbf{c} + \alpha\bar{\mathbf{c}}_y$. Then, modern cost-sensitive classification algorithms can be applied to locate a decent g, which would belong to the Pareto-optimal front with respect to $E_c(g)$ and $E(g)$.

The weighted sum approach has also been implicitly taken by other algorithms in machine learning. For instance, [34] combines the pairwise ranking criterion and squared regression criterion and shows that the resulting algorithm achieves the best performance on both criteria. Our proposed methodology similarly utilizes the simplicity of the weighted sum approach to allow seamless reuse of modern cost-sensitive classification algorithms. If other techniques for multicriteria optimization (such as evolutionary computation) are taken instead, new algorithms need to be designed to accompany the techniques. Given the prevalence of promising cost-sensitive classification algorithms (see Section 2), we thus choose to study only the weighted sum approach.

The parameter α in (4) can be intuitively explained as a soft control of the trade-off between cost and error, with $\alpha = 0$ and $\alpha = 1$ being the two extremes. The traditional (hard) cost-sensitive classification problem is a special case of soft cost-sensitive classification with $\alpha = 0$. On the other hand, the regular classification problem is a special case of soft cost-sensitive classification with $\alpha = 1$.

Another explanation behind (4) is regularization. From Figure 2, there are many low-cost classifiers in the red region. When picking one classifier using only the limited information in the training set $\mathcal{S}_c$, the classifier can be over-fitting. The added term $\alpha E(g)$ can be viewed as restricting the number of low-cost classifiers by only favoring those with lower error rate. This similar explanation can be found from [17], which considers cost-sensitive classification in the binary case. Furthermore, the restriction is similar to common regularization schemes, where a penalty term on complexity is used to limit the number of candidate classifiers [35].

We illustrate the regularization property of soft cost-sensitive classification with the data set vowel as an example. The details of the experimental procedures will be introduced in Section 4. The test cost of soft cost-sensitive classification with various α when coupled with the one-sided regression (OSR) algorithm is shown in Figure 3. For this data set, the lowest test cost does not happen at $\alpha = 0$ (hard cost-sensitive) nor $\alpha = 1$ (non cost-sensitive). By choosing the regularization parameter α appropriately, some intermediate, non-zero values of α (soft cost-sensitive) could lead to better test performance. The figure reveals the potential of soft cost-sensitive classification not only to improve the test error with the added $\alpha E(g)$ term during optimization, but also to possibly improve the test cost with the effect of regularization.

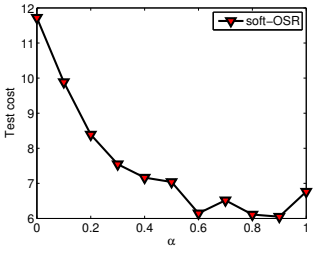


Fig. 3. the effect of the regularization parameter α on soft cost-sensitive classification

The simplicity of (4) allows soft cost-sensitive classification to modify the basic criterion easily. For instance, in an unbalanced classification problem, the weighted error rate $E_w(g) = \mathop{\mathcal{E}}_{(\mathbf{x},y)\sim\mathcal{D}} w_y \cdot \bar{\mathbf{c}}_y[g(\mathbf{x})]$ instead of $E(g)$ is often used to respect the influence of each class properly. If we replace $E(g)$ with $E_w(g)$ in (4), we get

$$\min_g \mathop{\mathcal{E}}_{(\mathbf{x},y,\mathbf{c})\sim\mathcal{D}_c} (1 - \alpha)\Big(\mathbf{c}[g(\mathbf{x})]\Big) + \alpha\Big(w_y \cdot \bar{\mathbf{c}}_y[g(\mathbf{x})]\Big) \tag{6}$$

The modified methodology (6) can also be solved by

modern cost-sensitive classification algorithms to get a decent g for both E_c and E_w.

4 EXPERIMENTS

In this section, we set up experiments to validate the usefulness of the proposed methodology of soft cost-sensitive classification in various procedures. We take four state-of-the-art multiclass cost-sensitive classification algorithms (see Section 2). Then we examine if the proposed methodology can improve them. The four algorithms are one-sided regression (OSR), cost-sensitive one-versus-one (CSOVO), cost-sensitive filter tree (CSFT) and cost-sensitive classification by Zhou and Liu (CSZL). We also include their regular classification siblings, one-versus-all (OVA), one-versus-one (OVO), and filter tree (FT) for comparisons. Note that OVO is also the regular classification sibling of CSZL and hence is denoted as OVO/ZL.

We couple all the algorithms with the support vector machine (SVM) [36] with the perceptron kernel [37] as the internal learner for the reduced problem, and take LIBSVM [38] as the SVM solver.[3] The regularization parameter λ of SVM is chosen within $\{2^{10}, 2^7, \ldots, 2^{-2}\}$. For the hard cost-sensitive classification algorithms, the best parameter setting is chosen by minimizing the 5-fold cross-validation cost. For the regular classification algorithms, which are not supposed to access any cost information in training or in validation, the best parameter λ is chosen by minimizing the 5-fold cross-validation error. We will study more about selecting the parameter α for soft cost-sensitive classification in Section 4.1.

We consider four sets of tasks: the traditional benchmark tasks for balancing the influence of each class, a real-world biomedical task for classifying bacteria (see Section 1), new benchmark tasks for emphasizing some of the classes, and the KDD Cup 1999 task for intrusion detection. These four tasks will demonstrate that soft cost-sensitive classification is useful both as a general algorithmic methodology and as a specific application tool.

4.1 Parameter Selection for Soft Cost-Sensitive Classification

An important issue for soft cost-sensitive classification is to choose the regularization parameter α properly. In particular, given two criteria of interest in soft cost-sensitive classification, it is non-trivial to decide the cross-validation criterion for picking the best parameter combination. We study two possible scenarios: For the first one, we simply take the cost to be the cross-validation criterion, with ties broken by choosing the largest α (most regularization); for the second one, we intend to choose a parameter that leads to both low error and low cost, and hence use $\max$(error, normalized cost) as the cross-validation criterion to be minimized. We report the results by running OSR on eight data sets: iris, wine, glass, vehicle, vowel, segment, dna, satimage, while similar observations have been found on other datasets and algorithms. For the cost, we take the benchmark one which will be introduced in Section 4.2.1. We normalize the sum of the cost matrix to be equal to sum of the naïve cost matrix that contains $\{\bar{\mathbf{c}}_y\}$.

The results are shown in Table 1 and Table 2 using a pairwise one-tailed t-test of significance level 0.1. The results confirm the trade-off between error and cost. In particular, CV by cost reaches lower cost than CV by $\max$(error, normalized cost) in 3 out of 8 data sets, but CV by $\max$(error, normalized cost) achieves lower error rate in 6 out of 8 data sets. Based on the study, we decide to use CV by cost for its simplicity and its better performance on the major criterion (cost).

TABLE 1
average test cost results for two validation criteria, with t-test for cost

	CV by cost	CV by $\max$(error, normalized cost)	t-test
iris	18.78 ± 3.71	21.58 ± 4.37	$\approx$
wine	12.28 ± 2.96	11.39 ± 2.70	$\approx$
glass	129.42 ± 9.50	139.72 ± 9.32	◯
vehicle	95.43 ± 10.41	109.65 ± 9.07	◯
vowel	6.42 ± 1.10	7.04 ± 1.03	$\approx$
segment	13.02 ± 1.08	13.33 ± 1.02	$\approx$
dna	22.76 ± 1.46	23.23 ± 1.26	$\approx$
satimage	34.86 ± 2.13	37.56 ± 1.93	◯

◯: CV by cost significantly better than the other procedure
×: CV by cost significantly worse than the other procedure
≈: otherwise

TABLE 2
average test error rate results for two validation criteria, with t-test for error rate

	CV by cost	CV by $\max$(error, normalized cost)	t-test
iris	4.73 ± 0.73	5.00 ± 0.71	$\approx$
wine	2.44 ± 0.38	1.88 ± 0.42	×
glass	31.94 ± 1.21	31.11 ± 0.98	$\approx$
vehicle	22.78 ± 0.72	21.56 ± 0.77	×
vowel	2.01 ± 0.34	1.59 ± 0.22	×
segment	2.96 ± 0.17	2.71 ± 0.13	×
dna	4.87 ± 0.27	4.16 ± 0.14	×
satimage	9.01 ± 0.33	7.30 ± 0.13	×

◯: CV by cost significantly better than the other procedure
×: CV by cost significantly worse than the other procedure
≈: otherwise

3. We use the cost-sensitive SVM implementation at http://www.csie.ntu.edu.tw/~htlin/program/cssvm/

4.2 Comparison on Benchmark Tasks

Twenty-two real-world data sets (iris, wine, glass, vehicle, vowel, segment, dna, satimage, usps, zoo ,yeast, pageblock, anneal, solar, splice, ecoli, nursery, soybean, arrhythmia, optdigits, mfeat, pendigit) are used in our next experiments. All data sets come from the UCI Machine Learning Repository [39] except usps [40]. In each run of the experiment, we randomly separate each data set with 75% of the examples for training and the rest 25% for testing. All the input vectors in the training set are linearly scaled to $[0, 1]$ and then the input vectors in the test set are scaled accordingly. These data sets do not contain any cost information and we generate two types of costs for each benchmark data set, one is inconsistent cost, and another is consistent cost (see Section 2).

4.2.1 Inconsistant Cost Matrix

We first generate costs similar to the procedure used by [11], [13], [15]. In particular, the benchmark is class-dependent and is based on a cost matrix $\mathbf{C}(y, k)$, where the diagonal entries $\mathbf{C}(y, y)$ are 0, and the other entries $\mathbf{C}(y, k)$ are uniformly sampled from $\left[0, \frac{|\{n:y_n=k\}|}{|\{n:y_n=y\}|}\right]$. This means that misclassifying a rare class as a frequent one is of a high cost in expectation. We further scale every $\mathbf{C}(y, k)$ to $[0, 1]$ by dividing it with the largest component in $\mathbf{C}$. We then record the average test cost and their standard errors for all algorithms over 20 random runs in Table 3. We also report the average test errors in Table 4.

From Table 3, soft-OSR and soft-CSOVO usually result in the lowest test cost. Most importantly, soft-OSR is among the best algorithms (bold) on 17 of the 22 data sets, and achieves the lowest cost on 8 of them. The follow-ups, OSR and CSOVO, were the state-of-the-art algorithms in cost-sensitive classification and reach promising performance often. Filter-tree and CSZL algorithms (CSFT, soft-CSFT, CSZL, soft-CSZL) are generally falling behind, and so are the regular classification algorithms (OVA, OVO, FT). The results justify that soft cost-sensitive classification can lead to similar and sometimes even better performance when compared with state-of-art cost-sensitive classification algorithms.

The experiments from Table 3 also indicate cost-sensitive classification algorithms are sometimes overfitting in cost. For instance, in data set vowel, all state-of-the-art cost-sensitive algorithms are inferior to their regular sibling algorithms in cost. In data set dna, although OSR achieves the similar cost to OVA, the two hard cost-sensitive classification algorithms CSOVO and CSFT are worse to OVO and FT, respectively. For these two data sets, soft cost-sensitive algorithms generally perform better than their hard siblings, and can often achieve lower costs than regular algorithms. The results justify the usefulness of soft cost-sensitive classification.

When we move to Table 4, regular classification algorithms like OVA and OVO generally achieve the lowest test errors. The hard cost-sensitive classification ones result in the highest test errors; soft ones lie in between.

Soft cost-sensitive classification does not improve CSZL significantly in terms of either the cost or the error rate. In particular, soft-CSZL ties with CSZL in cost on all 22 data sets, and results in lower error rate in only two of the data sets. One possible reason is that CSZL is implicitly "soft" in using the cost information when the cost matrix is inconsistent (i.e. CSZL needs to resort to an approximate solution), and readily leads to low error rate. In particular, CSZL (based on weighted OVO) reaches better error rate than CSOVO on 16 of the 22 data set; Thus, there is less room to improve CSZL with the proposed methodology. We see that there is no harm in using the soft methodology, though, because the hard CSZL is simply a special case of soft-CSZL with $\alpha = 0$.

4.2.2 Consistent Cost Matrix

Next, we consider consistent cost. We use the the same data sets and the normalize procedures. The consistent cost matrices are generated as follows:

Assume the class number is K. We first randomly generate a K-dimensional vector that contains increasing components within $[0, 1]$. We then use those values as solutions of the linear system that CSZL solves. Then, those components become weights of classes. We associate higher weights to the less frequent classes. The upper triangular of cost matrix $\mathbf{C}(k, y), \forall y > k$, can then be uniquely determined from the linear system; we generate the lower triangular of cost matrix $\mathbf{C}(y, k), \forall y > k$ from the uniformly sampled $\left[0, \frac{|\{n:y_n=y\}|}{|\{n:y_n=k\}|}\right]$ and set $\mathbf{C}(k, k)$ to zero.

Table 5 and Table 6 are the results when the cost is consistent for CSZL. The results are similar to the results for inconsistent cost. soft-CSOVO is among the best algorithms (bold) on 18 of the 22 data sets in terms of the cost, followed by soft-OSR, OSR and CSOVO. Filter-tree, CSZL and regular classification algorithms are falling behind. The results again justify that soft cost-sensitive classification could head to better performance when compared with state-of-art cost-sensitive classification algorithms.

From Table 5 and Table 6, we observe that soft cost-sensitive classification still could not improve CSZL much in error rate. Note that even when the cost is consistent, the modified cost in (5) is almost always inconsistent for CSZL when $\alpha > 0$. Such a phase change could be why soft-CSZL does not lead to much improvement, but it is usually no worse than hard CSZL, either.

TABLE 3
average test cost ($\cdot 10^{-3}$) on benchmark data sets for inconsistant cost
(note that the regular sibling of CSZL is also OVO)

data set	OVA	OSR	soft-OSR	FT	CSFT	soft-CSFT	OVO	CSOVO	soft-CSOVO	CSZL	soft-CSZL
iris	**18.34±4.48**	**17.21±3.84**	**18.79±3.72**	23.80±5.21	19.54±4.67	**15.91±3.55***	21.93±4.99	20.74±4.32	**19.34±4.26**	20.56±3.88	21.20±3.89
wine	**12.98±3.37**	**13.42±2.55**	**12.97±2.93**	15.21±3.49	**11.87±3.09**	15.62±4.44	15.04±4.05	**11.45±3.53***	**13.91±4.33**	**13.71±3.71**	16.26±4.11
glass	159.19±10.37	**126.84±9.71***	**129.42±9.51**	151.06±10.20	143.78±8.66	143.22±9.85	145.90±10.36	**128.56±9.77**	**132.69±9.62**	**136.44±9.56**	141.11±10.71
vehicle	114.14±9.08	**95.33±10.29***	**97.81±10.85**	112.48±7.71	**105.58±10.90**	106.74±11.27	112.31±8.82	**103.63±11.17**	**97.34±11.16**	**100.69±10.72**	**98.23±11.05**
vowel	**6.76±0.93**	11.72±1.44	**6.43±1.11**	9.53±1.31	13.71±1.58	11.87±1.47	**6.29±0.94**	9.58±1.08	**6.82±0.90**	**6.21±0.96***	**6.38±0.95**
segment	**14.02±1.17**	**13.84±0.94**	**13.03±1.08***	15.01±1.33	14.17±1.15	15.36±1.26	14.15±1.18	**14.00±1.11**	**14.10±1.31**	**13.95±1.21**	14.39±1.27
dna	24.43±1.26	24.40±1.55	**22.76±1.47***	27.94±2.34	31.49±2.09	29.23±2.28	24.51±1.37	28.26±2.04	24.51±1.52	25.04±1.41	**23.46±1.32**
satimage	40.20±2.08	**35.04±2.16**	**34.86±2.11***	41.98±2.08	40.16±2.10	39.63±2.23	40.43±1.92	**36.49±2.27**	**36.46±2.31**	38.70±2.04	38.97±1.89
usps	6.87±0.28	7.32±0.23	**6.58±0.27***	9.05±0.29	8.97±0.40	8.59±0.27	7.08±0.27	7.20±0.26	6.98±0.25	7.12±0.24	7.08±0.25
zoo	6.26±1.81	**2.49±0.50**	6.02±1.84	5.32±1.54	3.55±0.91	3.87±1.30	6.62±1.85	2.77±0.64	**2.26±0.47***	5.75±1.81	6.29±1.78
yeast	36.66±3.37	**0.58±0.07**	**0.58±0.07**	38.97±3.88	**0.62±0.09**	0.64±0.09	39.71±3.62	**0.55±0.08***	**0.55±0.08**	0.66±0.11	0.64±0.09
pageblock	2.80±0.48	0.18±0.04	0.19±0.04	2.78±0.48	**0.16±0.03**	**0.16±0.03***	2.59±0.45	**0.16±0.03**	**0.16±0.03**	**0.16±0.03**	**0.16±0.03**
anneal	0.85±0.23	**0.35±0.12***	**0.38±0.13**	0.85±0.23	0.58±0.16	0.64±0.16	0.83±0.23	0.61±0.16	0.67±0.17	0.61±0.15	0.67±0.16
solar	46.08±6.53	25.35±4.06	25.32±4.05	47.18±7.14	20.54±2.64	20.43±2.06	44.51±6.31	**18.04±1.94**	**17.89±1.95***	22.08±2.46	21.16±2.49
splice	14.01±0.84	**12.59±1.11**	**12.85±0.71**	16.64±0.79	18.19±1.62	16.06±1.17	13.97±0.76	17.06±1.26	13.28±0.88	**12.39±0.82**	**12.27±0.77***
ecoli	17.11±2.85	1.27±0.31	0.92±0.18	20.43±4.49	**0.85±0.14**	1.96±1.13	19.93±2.61	1.35±0.49	1.11±0.41	**0.76±0.12***	0.94±0.15
nursery	0.62±0.20	**0.00±0.00**	**0.00±0.00***	1.42±0.45	0.00±0.00	0.39±0.34	0.07±0.06	0.00±0.00	0.00±0.00	0.06±0.06	0.07±0.06
soybean	9.84±1.60	2.78±0.36	2.99±0.43	9.61±1.57	3.07±0.52	3.97±0.55	11.41±1.85	**2.13±0.29**	**2.08±0.30***	5.80±0.70	6.66±1.00
arrhythmia	6.46±1.23	0.55±0.08	0.63±0.08	8.69±1.78	0.57±0.19	0.55±0.17	7.32±1.48	**0.36±0.05***	**0.37±0.05**	**0.40±0.06**	**0.40±0.06**
optdigits	5.33±0.34	5.64±0.26	**4.90±0.35***	6.23±0.34	7.67±0.43	6.57±0.35	**4.98±0.26**	6.12±0.32	**5.23±0.31**	**4.92±0.27**	**4.92±0.27**
mfeat	**7.99±0.55**	9.27±0.74	**7.56±0.55***	11.74±0.76	11.23±0.89	10.87±0.83	8.74±0.59	8.36±0.61	8.70±0.64	8.59±0.60	8.54±0.60
pendigit	1.99±0.11	2.46±0.12	**1.88±0.09**	2.12±0.11	2.36±0.11	2.43±0.19	**1.88±0.10**	1.95±0.08	1.95±0.08	**1.85±0.12**	**1.80±0.11***

(those with the lowest mean are marked with *; those within one standard error of the lowest one are in bold)

TABLE 4
average test error (%) on benchmark data sets for inconsistant cost

data set	OVA	OSR	soft-OSR	FT	CSFT	soft-CSFT	OVO	CSOVO	soft-CSOVO	CSZL	soft-CSZL
iris	**4.21±0.78***	6.71±0.98	**4.74±0.73**	**4.61±0.79**	7.11±1.24	**4.47±0.81**	**4.74±0.80**	10.66±2.32	5.26±0.72	8.03±1.30	5.39±0.60
wine	**1.78±0.43**	4.00±0.62	**2.00±0.41**	2.22±0.47	**1.67±0.44***	2.22±0.57	2.11±0.51	**1.78±0.51**	**1.78±0.54**	**2.09±0.48**	2.56±0.50
glass	**28.52±0.82***	32.22±1.11	31.94±1.21	29.81±0.96	39.17±2.35	36.02±2.52	**28.89±0.84**	44.26±2.73	45.28±2.52	33.80±1.73	32.50±1.39
vehicle	**20.66±0.62**	24.15±0.83	22.78±0.73	**20.75±0.64**	29.88±2.92	30.40±3.04	**20.31±0.67***	28.73±2.19	25.14±1.57	24.39±1.68	23.00±1.24
vowel	**1.27±0.17***	5.38±0.47	1.88±0.27	1.94±0.24	6.25±1.43	2.74±0.39	**1.29±0.18**	5.93±0.63	**1.43±0.17**	**1.31±0.18**	**1.31±0.18**
segment	**2.60±0.16***	3.69±0.27	**2.76±0.15**	2.78±0.15	4.30±0.62	3.43±0.35	**2.60±0.15**	5.57±0.95	4.11±0.59	**2.67±0.18**	2.78±0.20
dna	**4.20±0.14**	6.96±0.65	4.87±0.27	4.81±0.24	9.14±1.52	5.32±0.30	**4.19±0.13***	7.90±0.80	5.81±0.85	4.74±0.25	4.39±0.15
satimage	**7.19±0.10***	9.52±0.30	9.01±0.34	7.55±0.11	10.58±0.63	9.85±0.75	**7.24±0.09**	12.55±0.66	12.51±0.68	7.87±0.16	7.99±0.30
usps	**2.19±0.07***	3.82±0.13	2.66±0.11	2.79±0.06	6.26±0.86	3.50±0.10	2.28±0.06	5.27±0.70	3.53±0.17	2.33±0.06	2.27±0.06
zoo	**5.19±0.83**	15.38±1.61	12.50±1.51	**4.81±0.81***	12.69±2.54	8.27±2.26	6.15±1.03	10.77±1.71	8.08±1.74	10.77±2.83	14.04±3.01
yeast	40.38±0.64	73.76±0.55	73.68±0.55	40.20±0.52	77.02±0.92	76.70±0.81	**39.27±0.56***	76.58±0.68	76.70±0.67	75.96±0.70	76.31±0.65
pageblock	3.22±0.09	39.25±4.36	38.54±4.74	**3.10±0.10**	78.25±6.10	81.82±5.81	**3.06±0.08***	76.75±6.18	76.75±6.18	80.51±5.88	83.14±5.56
anneal	**1.40±0.15***	8.78±0.94	6.98±1.13	**1.47±0.17**	11.31±1.94	9.47±4.40	**1.51±0.15**	19.02±4.24	10.60±4.53	9.44±4.50	12.07±6.19
solar	27.27±0.42	34.83±1.16	35.22±1.75	27.27±0.46	46.15±3.12	43.48±2.85	**26.61±0.43***	47.49±3.30	47.83±3.12	41.21±3.30	41.54±3.36
splice	**3.86±0.15***	7.68±1.16	5.21±0.56	4.62±0.18	9.59±1.46	6.52±0.74	**3.92±0.12**	13.34±2.69	8.13±2.60	5.49±0.96	5.34±0.98
ecoli	15.12±0.99	32.68±1.67	33.63±1.61	16.85±1.14	36.73±2.72	40.89±3.85	**14.05±0.75***	37.80±3.30	38.45±3.19	38.57±3.00	38.99±2.99
nursery	0.11±0.02	33.33±0.17	31.02±1.54	0.32±0.08	33.89±0.44	20.04±3.61	**0.02±0.01***	37.62±2.17	3.31±2.21	1.69±1.63	**0.02±0.01**
soybean	**6.55±0.32***	24.53±0.82	21.67±1.42	7.13±0.38	35.41±2.48	28.48±3.40	7.46±0.34	39.06±3.51	40.12±3.76	24.47±3.25	20.50±3.56
arrhythmia	**28.41±0.93**	66.37±2.25	66.42±2.11	30.40±0.62	88.81±2.47	86.15±3.12	**27.92±0.74***	85.18±2.49	83.05±3.37	86.68±2.66	84.87±3.34
optdigits	1.09±0.06	1.85±0.06	1.15±0.07	1.35±0.05	2.14±0.24	1.55±0.05	**1.04±0.05***	2.25±0.09	1.36±0.12	1.09±0.05	**1.04±0.05**
mfeat	**1.69±0.09***	3.10±0.18	1.84±0.11	2.45±0.10	3.89±0.37	2.99±0.38	1.86±0.08	4.32±0.53	2.50±0.22	1.85±0.08	1.90±0.09
pendigit	**0.40±0.02**	0.85±0.04	**0.39±0.02**	0.45±0.02	0.62±0.04	0.52±0.03	**0.38±0.02***	0.65±0.03	0.42±0.02	**0.40±0.02**	**0.39±0.02**

(those with the lowest mean are marked with *; those within one standard error of the lowest one are in bold)

Mostly (especially for CSOVO and OSR), soft cost-sensitive classification is better than the regular sibling in terms of the cost, the major criterion. It is similar to (sometimes better than) the hard sibling in terms of the cost, and usually better in terms of the error. We further justify the claims above by comparing the average test cost between soft cost-sensitive classification algorithms with their corresponding siblings using a pairwise one-tailed t-test of significance level 0.1, as shown in Table 7 for inconsistent cost and Table 9 for consistent cost. The results of these two cost are very similar: for each family of algorithms (OVA, OVO/ZL or FT), soft cost-sensitive classification algorithms are generally among the best of the three, and are significantly better than their regular siblings (except CSZL).

Table 8 and Table 10 shows the same t-test for comparing the test errors between soft cost-sensitive classification algorithms and their hard siblings in inconsistent and consistent costs, respectively. For inconsistent cost, we see that soft-OSR improves OSR on 16 of the 22 data sets in terms of the test error; soft-CSOVO improves CSOVO on 13 of the 22; soft-CSFT improves CSFT on 14 of the 22; soft-CSZL improves CSZL on 2 of the 22. For consistent cost, we see that soft-OSR improves OSR on 14 of the 22 data sets in terms of the test error; soft-CSOVO improves CSOVO on 13 of the 22; soft-CSFT improves CSFT on 17 of the 22; soft-CSZL improves CSZL on 3 of the 22. Given the similar test cost between soft and hard cost-sensitive classification algorithms in Table 7, the significant improvements on the test error justify that soft cost-sensitive classification algorithms are better choices for practical applications.

4.3 Comparison on a Real-world Biomedical Task

To test the validity of our proposed soft cost-sensitive classification methodology on true applications, we use two real-world data sets for our experiments. The first one is a biomedical task [3], and the other one to be introduced later is from KDDCup 1999 [20]. Both data sets go through similar splitting and scaling procedures, as we did for the benchmark data sets.

The biomedical task is on classifying the bacterial meningitis, which is a serious and often life-threatening form of the meningitis infection. The inputs are the spectra of bacterial pathogens extracted by the Surface Enhanced Raman Scattering (SERS)

TABLE 5
average test cost ($\cdot 10^{-3}$) on benchmark data sets for consistant cost
(note that the regular sibling of CSZL is also OVO)

data set	OVA	OSR	soft-OSR	FT	CSFT	soft-CSFT	OVO/ZL	CSOVO	soft-CSOVO	CSZL	soft-CSZL
iris	46.03±5.11	**38.05±6.12**	**37.73±5.36***	44.92±4.25	**40.03±4.50**	46.13±4.90	45.17±3.80	**40.03±4.50**	**42.21±5.07**	**42.79±4.04**	**41.00±4.48**
wine	**11.18±2.36**	16.59±2.12	**12.24±2.91**	**12.04±2.42**	13.99±1.98	**11.45±2.62**	**10.88±2.53**	**11.40±1.94**	**10.61±2.00**	**9.63±2.66***	13.43±2.59
glass	100.14±9.30	**86.57±8.73***	**91.88±9.49**	100.27±8.93	**93.40±9.59**	**94.38±9.01**	106.84±10.59	**91.37±10.52**	**91.29±9.74**	99.10±9.92	**93.19±9.71**
vehicle	105.20±3.98	103.85±4.61	101.81±4.51	112.48±7.71	104.18±6.14	98.77±5.32	99.28±3.82	102.76±6.75	**93.30±5.06***	99.37±3.85	99.26±3.92
vowel	6.99±1.29	10.98±1.12	6.73±1.26	9.81±1.34	12.61±1.75	10.32±1.40	**5.49±1.01**	10.16±1.28	**5.74±1.04**	**5.21±1.01**	**5.04±1.00***
segment	**12.18±1.17**	12.72±1.21	**12.31±1.18**	**12.48±1.21**	14.29±1.33	**12.55±1.28**	**11.58±1.04***	**12.42±1.22**	**11.60±1.07**	**11.63±1.04**	**11.63±1.04**
dna	**19.84±1.31**	23.05±1.43	**19.80±1.37***	25.27±1.61	25.78±2.05	23.91±1.61	**20.69±1.42**	25.86±2.02	**20.97±1.46**	**20.45±1.37**	**20.93±1.49**
satimage	30.32±2.10	**29.23±2.11**	**29.08±2.10**	31.40±2.07	30.02±2.07	30.53±2.26	30.92±2.07	**28.02±2.02**	**27.97±2.05***	31.03±2.04	30.99±2.04
usps	6.15±0.25	6.05±0.21	**5.71±0.22***	9.05±0.29	7.41±0.31	7.07±0.35	6.38±0.28	6.01±0.31	**5.91±0.29**	6.35±0.29	6.32±0.28
zoo	12.95±2.81	11.57±1.71	10.73±2.41	**7.29±1.80***	9.91±2.26	11.25±2.74	11.25±2.45	**7.97±1.72**	9.23±2.38	12.46±2.60	11.79±2.68
yeast	8.42±0.69	7.34±0.82	7.40±0.80	27.29±2.62	7.79±0.85	7.53±0.75	8.06±0.67	**6.51±0.76***	**6.66±0.75**	8.20±0.73	8.20±0.74
pageblock	7.76±0.81	5.85±0.82	6.02±0.79	**4.70±0.49***	7.32±0.95	6.59±0.83	7.52±0.81	5.83±0.80	5.83±0.83	7.81±0.80	7.57±0.82
anneal	4.73±0.91	4.13±0.57	**3.67±0.53**	5.77±1.04	**3.48±0.57**	**3.57±0.52**	3.92±0.65	**3.30±0.49**	**3.29±0.51***	3.92±0.65	3.93±0.65
solar	54.51±2.51	43.53±2.69	43.66±2.84	49.67±4.40	46.85±3.65	44.07±3.32	58.93±2.90	**35.82±2.69**	**35.68±2.70***	56.12±3.00	55.28±2.64
splice	**19.67±1.06**	25.08±1.35	**19.62±0.98**	**18.86±1.73***	28.83±1.82	22.64±1.37	**19.87±1.04**	29.36±2.01	**19.73±0.96**	**20.05±1.03**	**19.91±1.07**
ecoli	4.84±0.54	**4.67±0.60**	**4.20±0.53***	**4.28±0.80**	5.89±0.60	5.76±0.86	**4.46±0.52**	5.11±0.49	**4.51±0.43**	**4.44±0.51**	**4.45±0.51**
nursery	0.01±0.00	0.04±0.01	**0.01±0.00**	1.42±0.45	0.01±0.00	0.02±0.01	**0.01±0.00***	0.01±0.00	**0.01±0.00**	**0.01±0.00**	**0.01±0.00**
soybean	4.17±0.48	2.83±0.29	3.39±0.40	**1.63±0.07***	3.95±0.50	4.59±0.48	5.02±0.66	2.23±0.24	2.23±0.26	4.53±0.61	4.48±0.58
arrhythmia	24.35±2.63	**8.72±1.71**	**8.68±1.68**	**8.69±1.78**	14.58±2.11	14.39±2.58	26.21±2.90	**8.13±1.77**	**7.96±1.62***	25.27±3.06	25.18±3.10
optdigits	4.24±0.24	4.24±0.24	**3.94±0.26***	6.23±0.34	5.85±0.38	5.59±0.36	**4.08±0.27**	4.32±0.21	**4.14±0.24**	**4.18±0.34**	**4.10±0.28**
mfeat	**7.32±0.50**	7.53±0.58	**6.90±0.56***	11.74±0.76	9.49±0.61	9.15±0.57	8.03±0.56	7.55±0.54	7.78±0.58	8.00±0.55	8.21±0.59
pendigit	1.98±0.13	2.22±0.09	**1.85±0.12**	2.24±0.13	2.38±0.15	2.27±0.11	**1.85±0.10**	2.09±0.10	**1.87±0.10**	**1.83±0.12**	**1.78±0.11***

(those with the lowest mean are marked with *; those within one standard error of the lowest one are in bold)

TABLE 6
average test error (%) on benchmark data sets for consistent cost

data set	OVA	OSR	soft-OSR	FT	CSFT	soft-CSFT	OVO/ZL	CSOVO	soft-CSOVO	CSZL	soft-CSZL
iris	5.66±0.60	**4.74±0.69***	**4.74±0.63**	5.53±0.52	**5.00±0.56**	5.66±0.57	5.53±0.49	**5.00±0.56**	**5.26±0.62**	**5.39±0.57**	**5.13±0.57**
wine	**1.78±0.37**	3.33±0.43	2.11±0.46	2.11±0.46	2.33±0.33	2.00±0.41	**1.78±0.40**	2.11±0.33	**1.89±0.32**	**1.56±0.39***	2.00±0.38
glass	**30.65±1.31**	34.26±1.84	32.78±1.53	**30.74±1.22**	42.87±3.56	36.94±2.83	34.54±3.52	40.56±3.45	39.91±3.55	**30.83±1.47**	**30.28±1.48***
vehicle	19.62±0.54	22.97±1.42	22.48±1.36	20.75±0.64	24.79±2.17	23.09±2.29	**18.56±0.60**	32.10±2.85	24.98±2.72	**18.51±0.61***	19.62±0.68
vowel	1.75±0.27	5.56±0.67	1.83±0.28	2.38±0.21	7.46±1.38	3.10±0.31	**1.45±0.19**	9.27±1.52	1.75±0.30	**1.39±0.18**	**1.35±0.18***
segment	**2.34±0.11**	3.04±0.13	2.51±0.12	2.44±0.13	3.49±0.28	3.04±0.37	**2.26±0.10**	3.83±0.47	2.98±0.35	**2.25±0.10***	**2.28±0.09**
dna	**4.10±0.13***	9.22±1.86	4.47±0.25	5.12±0.29	12.50±3.41	6.34±1.06	4.34±0.14	10.35±2.33	8.34±2.59	4.31±0.16	4.46±0.20
satimage	**7.34±0.09***	8.89±0.23	8.33±0.30	7.64±0.09	10.78±0.70	9.52±0.61	7.51±0.08	12.60±0.84	12.38±0.90	7.52±0.08	7.55±0.09
usps	**2.25±0.06***	4.18±0.29	2.53±0.13	2.79±0.06	6.00±0.91	3.33±0.10	2.35±0.06	6.51±1.28	5.35±1.33	2.33±0.06	2.32±0.06
zoo	**5.19±0.83**	7.69±1.12	**5.19±1.13**	6.15±1.03	5.58±1.35	**4.62±0.89***	**5.00±0.72**	6.54±0.98	**5.19±0.78**	5.58±0.92	**5.00±0.77**
yeast	40.43±0.56	46.59±1.12	45.82±1.16	**39.91±0.58**	56.33±2.09	51.64±2.31	**39.47±0.51***	53.38±1.39	53.45±1.40	41.00±0.46	41.00±0.46
pageblock	3.19±0.09	4.49±0.25	4.67±0.30	3.22±0.10	6.47±0.60	6.45±0.88	**3.06±0.10***	7.29±0.60	7.43±0.67	**3.14±0.09**	**3.13±0.09**
anneal	3.17±1.75	5.56±0.77	3.27±0.71	**1.49±0.15**	4.11±1.49	3.44±1.13	**1.49±0.15**	4.31±1.04	2.47±0.74	**1.47±0.16***	**1.51±0.17**
solar	27.28±0.46	31.25±0.92	30.23±1.03	**26.44±0.38***	39.32±1.74	41.41±2.37	26.90±0.45	41.85±1.84	42.34±1.91	27.44±0.53	28.23±0.46
splice	**3.87±0.15***	6.60±0.59	4.08±0.19	4.47±0.24	6.99±0.63	4.81±0.32	**3.87±0.12**	6.74±0.53	**3.92±0.13**	**3.88±0.13**	**3.91±0.13**
ecoli	15.12±0.99	21.61±2.79	18.81±2.80	16.85±0.72	28.57±3.90	23.10±2.90	**13.99±0.76**	32.86±4.95	25.00±4.52	**13.93±0.73***	**13.99±0.76**
nursery	0.09±0.02	3.44±0.34	0.04±0.01	0.32±0.08	6.23±2.01	0.35±0.07	**0.01±0.00***	6.55±2.30	**0.02±0.00**	0.06±0.05	**0.01±0.00**
soybean	**6.43±0.30***	11.61±1.52	10.18±1.62	7.02±0.28	22.89±2.50	14.42±2.76	7.31±0.36	25.99±3.19	23.95±3.22	8.74±1.12	8.65±1.14
arrhythmia	**28.67±0.89**	69.11±2.59	68.41±3.11	30.40±0.62	73.14±5.96	66.28±6.31	**28.05±0.74***	86.59±3.62	84.38±4.41	32.70±3.50	32.88±3.48
optdigits	**1.09±0.06**	2.48±0.14	1.22±0.06	1.35±0.05	3.35±1.08	1.74±0.08	**1.05±0.05***	3.72±0.59	1.58±0.15	**1.05±0.06**	**1.05±0.05**
mfeat	**1.69±0.09***	3.45±0.22	1.93±0.10	2.45±0.10	4.85±1.00	2.82±0.13	1.81±0.08	4.62±0.83	2.49±0.18	1.85±0.08	1.86±0.08
pendigit	0.40±0.02	0.89±0.04	0.41±0.02	0.47±0.03	0.73±0.05	0.54±0.03	**0.38±0.02**	0.74±0.05	0.44±0.02	0.43±0.02	**0.38±0.02***

(those with the lowest mean are marked with *; those within one standard error of the lowest one are in bold)

platform [41]. In this paper, we call the task SERS, which contains 79 clinical samples of ten meningitis-causing bacteria species collected in the National Taiwan University Hospital and 17 standard bacteria samples from American Type Culture Collection. The cost matrix of SERS is shown in Table 11, which is specified by two human physicians who are specialized in infectious diseases.

The results are shown in Table 12. Among the eleven algorithms, soft-CSOVO gets the lowest cost. If we compare the other eight algorithms with soft-CSOVO using a pairwise one-tailed t-test of significance level 0.1, we see that soft-CSOVO is significantly better than all other algorithms. The results confirm the usefulness of soft cost-sensitive classification for this real-world task.

SERS is an interesting data set in which regular classification algorithms like OVO/ZL or FT can perform better than their hard cost-sensitive classification siblings like CSOVO or CSFT or CSZL. Given the small number of examples in SERS, the phenomenon can be attributed to overfitting with respect to the cost—i.e. over-using the cost information. Soft cost-sensitive classification provides a balanced alternative between over-using (hard) or not using (regular) the cost. The balancing can lead to significantly lower test cost, as demonstrated by the promising performance of soft-CSOVO on this biomedical task.

4.4 Comparison on New Benchmark Tasks: Emphasizing Cost

Next, we explore the usefulness of the algorithms with a new benchmark. There are two situations when emphasizing different classes: The first situation is that one wants to indicate each class in the data set to be of different influence, which corresponds to scaling the *rows* of the cost matrix as discussed in Section 2. The second situation is to avoid that the examples of some classes to be wrongly predicted as some emphasized classes, which corresponds by scaling up some *columns* of the cost matrix. As mentioned in Section 2, cost-sensitive classification is more sophisticated than re-weighting. In particular, it allows us to mark important classes by scaling up some *columns* or some *rows* of the cost matrix. In this benchmark task, we emphasize the *columns* of the cost matrix by an emphasis parameter u.

We design the emphasizing cost to examine the

TABLE 7
comparisons on the test cost between the algorithms and their soft cost-sensitive classification sibling using a pairwise one-tailed t-test of significance level 0.1 in inconsistent cost

data set	OVA	OSR	OVO/ZL	CSOVO	FT	CSFT	OVO/ZL	CSZL
iris	≈	≈	≈	≈	◯	≈	≈	≈
wine	≈	≈	≈	≈	≈	≈	≈	≈
glass	◯	≈	≈	≈	≈	≈	≈	≈
vehicle	◯	≈	◯	◯	≈	≈	◯	≈
vowel	≈	◯	≈	◯	≈	◯	≈	≈
segment	◯	◯	≈	≈	≈	≈	≈	≈
dna	◯	◯	≈	◯	≈	◯	≈	≈
satimage	◯	≈	◯	≈	◯	≈	≈	≈
usps	≈	◯	≈	≈	◯	≈	≈	≈
zoo	≈	◯	◯	≈	≈	≈	≈	≈
yeast	◯	≈	◯	≈	◯	≈	◯	≈
pagblock	◯	≈	◯	≈	◯	≈	◯	≈
anneal	◯	≈	≈	≈	≈	≈	◯	≈
solar	◯	≈	◯	≈	◯	≈	≈	≈
splice	◯	≈	≈	◯	≈	◯	◯	≈
ecoli	◯	≈	◯	◯	◯	≈	◯	≈
nursery	◯	≈	≈	≈	≈	≈	≈	≈
soybean	◯	≈	◯	≈	◯	≈	◯	≈
arrhythmia	◯	≈	◯	≈	◯	≈	◯	≈
optdigits	◯	◯	≈	◯	≈	◯	≈	≈
mfeat	≈	◯	≈	≈	≈	≈	≈	≈
pendigit	≈	◯	≈	≈	×	≈	≈	≈

◯: soft cost-sensitive algorithms significantly better
×: soft cost-sensitive algorithms significantly worse
≈: otherwise

TABLE 9
comparisons on the test cost between the algorithms and their soft cost-sensitive classification sibling using a pairwise one-tailed t-test of significance level 0.1 in consistent cost

data set	OVA	OSR	OVO/ZL	CSOVO	FT	CSFT	OVO/ZL	CSZL
iris	◯	≈	≈	≈	≈	≈	≈	≈
wine	≈	≈	≈	≈	≈	≈	≈	≈
glass	◯	◯	≈	≈	◯	≈	≈	◯
vehicle	◯	≈	◯	≈	≈	◯	≈	≈
vowel	≈	◯	≈	≈	≈	◯	≈	≈
segment	≈	≈	≈	◯	≈	◯	≈	≈
dna	≈	◯	≈	≈	≈	◯	≈	≈
satimage	◯	≈	≈	≈	◯	≈	≈	≈
usps	◯	◯	◯	≈	◯	≈	≈	≈
zoo	≈	≈	≈	≈	◯	≈	≈	≈
yeast	◯	≈	◯	≈	◯	≈	◯	≈
pagblock	◯	≈	◯	≈	◯	≈	◯	◯
anneal	≈	≈	◯	≈	≈	≈	≈	≈
solar	◯	≈	≈	≈	◯	≈	◯	≈
splice	≈	◯	≈	◯	≈	◯	≈	≈
ecoli	◯	◯	≈	≈	≈	◯	≈	≈
nursery	◯	◯	◯	◯	≈	◯	≈	≈
soybean	◯	◯	◯	◯	◯	≈	≈	≈
arrhythmia	◯	≈	≈	≈	◯	≈	≈	≈
optdigits	≈	≈	≈	≈	≈	≈	≈	≈
mfeat	◯	≈	◯	≈	≈	≈	≈	≈
pendigit	◯	◯	≈	≈	≈	◯	≈	≈

◯: soft cost-sensitive algorithms significantly better
×: soft cost-sensitive algorithms significantly worse
≈: otherwise

TABLE 8
comparison on the test errors between the hard cost-sensitive classification algorithms and their soft sibling using a pairwise one-tailed t-test of significance level 0.1 in inconsistent cost

data set	OSR	CSOVO	CSFT	CSZL
iris	◯	◯	◯	◯
wine	◯	≈	≈	≈
glass	◯	≈	≈	≈
vehicle	◯	◯	≈	≈
vowel	◯	◯	◯	≈
segment	◯	◯	◯	≈
dna	◯	◯	◯	≈
satimage	◯	◯	◯	≈
usps	◯	◯	◯	≈
zoo	◯	◯	◯	≈
yeast	≈	≈	≈	≈
pagblock	≈	≈	≈	≈
anneal	≈	◯	◯	≈
solar	≈	≈	≈	≈
splice	◯	≈	◯	≈
ecoli	◯	≈	≈	≈
nursery	≈	◯	◯	≈
soybean	◯	≈	◯	◯
arrhythmia	≈	≈	≈	≈
optdigits	◯	◯	◯	≈
mfeat	◯	◯	◯	≈
pendigit	◯	◯	◯	≈

◯: soft cost-sensitive algorithms significantly better
×: soft cost-sensitive algorithms significantly worse
≈: otherwise

TABLE 10
comparison on the test errors between the hard cost-sensitive classification algorithms and their soft sibling using a pairwise one-tailed t-test of significance level 0.1 in consistent cost

data set	OSR	CSOVO	CSFT	CSZL
iris	≈	≈	≈	≈
wine	◯	≈	≈	≈
glass	≈	≈	◯	≈
vehicle	≈	◯	≈	◯
vowel	◯	◯	◯	≈
segment	◯	◯	≈	≈
dna	◯	≈	◯	◯
satimage	◯	◯	≈	≈
usps	◯	◯	◯	≈
zoo	◯	◯	≈	≈
yeast	≈	≈	◯	≈
pagblock	≈	≈	≈	≈
anneal	◯	◯	≈	≈
solar	◯	≈	≈	≈
splice	◯	◯	◯	≈
ecoli	◯	◯	◯	≈
nursery	◯	◯	◯	≈
soybean	◯	≈	◯	≈
arrhythmia	≈	◯	≈	≈
optdigits	◯	◯	≈	≈
mfeat	◯	◯	◯	≈
pendigit	◯	◯	◯	◯

◯: soft cost-sensitive algorithms significantly better
×: soft cost-sensitive algorithms significantly worse
≈: otherwise

stability of the algorithms when using large u. In this experiment, we vary the the emphasis parameter u between $\{10^2, 10^3, \ldots, 10^6\}$. The results are shown in Figure 4. Due to the page limits, we only report the results of OSR and soft-OSR on iris, vehicle, and segment. The figures plot the scaled test cost E_c/u on different values of $\log_{10} u$. From the three figures, we see that soft-OSR is better than OSR across all u. When the emphasis is very high (like 10^6), OSR can be conservative and "paranoid." It avoids classifying any of the test examples as the emphasized class, which results in the worse performance.

TABLE 11
cost matrix on SERS

real class \ classify to	Ab	Ecoli	HI	KP	LM	Nm	Psa	Spn	Sa	GBS
Ab	0	1	10	7	9	9	5	8	9	1
Ecoli	3	0	10	8	10	10	5	10	10	2
HI	10	10	0	3	2	2	10	1	2	10
KP	7	7	3	0	4	4	6	3	3	8
LM	8	8	2	4	0	5	8	2	1	8
Nm	3	10	9	8	6	0	8	3	6	7
Psa	7	8	10	9	9	7	0	8	9	5
Spn	6	10	7	7	4	4	9	0	4	7
Sa	7	10	6	5	1	3	9	2	0	7
Gbs	2	5	10	9	8	6	5	6	8	0

TABLE 12
experiment results on SERS, with t-test for cost

	error (%)	cost ($\cdot 10^0$)	t-test
OVA	23.0 ± 2.51	1.056 ± 0.097	◯
OSR	27.6 ± 2.27	0.986 ± 0.092	◯
soft-OSR	25.8 ± 2.80	1.024 ± 0.095	◯
OVO/ZL	23.2 ± 2.55	0.970 ± 0.106	◯
CSOVO	27.4 ± 1.53	1.150 ± 0.109	◯
soft-CSOVO	26.6 ± 2.55	0.906 ± 0.069	$*$
FT	23.0 ± 2.51	0.986 ± 0.092	◯
CSFT	27.6 ± 1.40	1.118 ± 0.090	◯
soft-CSFT	31.4 ± 4.09	1.054 ± 0.040	◯
CSZL	26.0 ± 3.42	1.030 ± 0.110	◯
soft-CSFT	24.0 ± 2.87	0.990 ± 0.105	◯

$*$: best entry of cost
◯: best entry significantly better in cost
$\approx$: otherwise

On the other hand, the curves of soft-OSR remain mostly flat, which demonstrate that soft cost-sensitive classification is less sensitive (paranoid) to large cost components. The results again justify the superiority of soft-OSR, a promising representative of soft cost-sensitive classification, over its hard sibling.

4.5 Comparison on New Benchmark Tasks: Unbalanced Classification

The goal of this experiment is to examine the benchmark tasks with cost-sensitive and unbalanced data set. As discussed in Section 3, weighted error rate is a more suitable basic criterion compared to error rate, and the corresponding methodology can be solved by using (6).

In this benchmark data set experiment, we set $w_y = \frac{1}{|\{n:y_n\}|}$. We further scale every weight w_y to $[0, 1]$ by dividing it with the largest component in weight. For the cost, we adapt the inconsistent benchmark cost mentioned in Section 4.2.1. We choose ten unbalanced benchmark data sets, as shown in Table 13. Then we compare three algorithms: OSR fed with the benchmark cost in Section 4.2.1, weighted OVA with w_y as weights, and soft OSR with the benchmark cost and weighted error. Table 14 and Table 15 show the cost and weighted error for those data sets. From Table 14, OSR achieve the lowest cost on most data set (except glass); soft OSR is close to OSR in cost; weighted OVA falls behind. The results are similar to the findings in Section 4.2. On the other hand, from Table 15, weighted OVA reaches the lowest weighted error; OSR reaches the highest; soft-OSR is in between the two. The results justify that soft cost-sensitive classification can be used to achieve both low cost and low weighted error.

We further compare OSR with soft-OSR using another criterion: G-mean. G-mean is the geometric mean accuracy of each class [42]. Higher G-mean reflects better performance for unbalanced classification tasks. The results are shown in Table 16. We see that soft-OSR out perform OSR in 8 out of 10 data sets. The results justify the usefulness of extending soft cost-sensitive classification with weighted error.

TABLE 13
unbalanced benchmark data sets

data set	size	features	class	class distribution
pageblock	5473	10	5	4913/329/28/88/115
wine	178	13	3	59/71/48
glass	214	9	6	70/76/17/13/9/29
dna	3186	180	3	767/765/1654
satimage	6435	36	6	1533/703/1358/626/707/1508
zoo	101	16	7	41/20/5/13/4/8/10
yeast	1484	8	10	5/463/244/163/51/44/35/30/20/429
anneal	898	84	5	684/40/8/67/99
solar	1389	44	6	287/327/212/51/396/116
splice	3190	287	3	767/768/1655

TABLE 14
average test cost ($\cdot 10^{-3}$) on unbalanced benchmark data sets

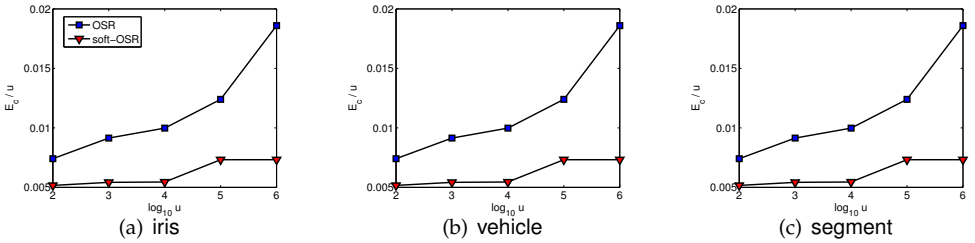
data set	weighted OVA	OSR	soft OSR
pageblock	2.23±0.11	**0.19±0.04***	3.05±0.49
wine	16.29±3.68	**13.00±2.62**	**11.24±3.10***
glass	**93.01±4.48***	126.20±9.84	149.74±10.39
dna	30.84±1.01	**24.33±1.54***	**24.50±1.27**
satimage	53.46±0.99	**35.18±2.14***	40.07±2.06
zoo	42.31±5.29	**2.49±0.50***	6.26±1.81
yeast	13.60±0.44	**0.58±0.07***	36.66±3.37
anneal	3.88±0.66	**0.35±0.12***	0.85±0.23
solar	84.66±2.17	**25.35±4.06***	46.08±6.53
splice	29.12±1.20	**12.59±1.11***	14.01±0.84

(those with the lowest mean are marked with *; those within one standard error of the lowest one are in bold)

TABLE 15
average weighted error on unbalanced benchmark data sets

data set	weighted OVA	OSR	soft OSR
pageblock	**3.06±0.15***	21.40±0.76	20.49±0.93
wine	**0.73±0.17***	1.48±0.26	**0.79±0.16**
glass	**5.02±0.24***	5.81±0.31	5.73±0.32
dna	**24.58±0.81***	42.29±3.52	30.18±2.16
satimage	**86.02±1.59***	101.48±3.35	92.28±2.46
zoo	**1.10±0.14***	2.28±0.26	**1.20±0.20**
yeast	**5.05±0.16***	8.68±0.18	8.97±0.29
anneal	**0.87±0.15***	2.71±0.37	1.63±0.25
solar	**29.46±0.75***	35.51±1.02	34.71±0.98
splice	**23.23±0.96***	49.84±6.99	28.07±1.75

(those with the lowest mean are marked with *; those within one standard error of the lowest one are in bold)



(a) iris (b) vehicle (c) segment

Fig. 4. test E_c/u of OSR and soft-OSR with the emphasizing cost for different emphasis parameter u

TABLE 16
average gmean on unbalanced benchmark data sets

data set	soft OSR	OSR	t-test
pageblock	**0.72±0.71***	0.00±0.00	◯
wine	**94.08±1.23***	87.25±2.89	◯
glass	**1.53±0.54**	**1.36±0.46***	≈
dna	**85.04±1.00***	79.32±1.59	◯
satimage	**52.59±1.07***	48.98±1.46	◯
zoo	**9.17±4.93***	0.00±0.00	◯
yeast	**0.00±0.00***	**0.00±0.00**	≈
anneal	**39.88±7.22***	19.59±5.85	◯
solar	**0.69±0.25***	0.45±0.17	◯
splice	**85.96±0.92***	72.14±4.81	◯

◯: soft cost-sensitive algorithms significantly better
×: soft cost-sensitive algorithms significantly worse
≈: otherwise
(those with the higest mean are marked with *; those within one standard error of the highest one are in bold)

TABLE 17
average test results on kdd99, with t-test for cost

	error (%)	cost ($\cdot 10^{-3}$)	t-test
OSR	0.11 ± 0.003	1.80 ± 0.171	$*$
soft-OSR	0.11 ± 0.003	1.89 ± 0.163	≈
CSOVO	0.11 ± 0.003	1.81 ± 0.169	≈
soft-CSOVO	0.11 ± 0.003	1.82 ± 0.161	≈
CSFT	0.11 ± 0.003	1.83 ± 0.171	≈
soft-CSFT	0.11 ± 0.003	1.81 ± 0.165	≈
CSZL	0.11 ± 0.003	1.83 ± 0.170	≈
soft-CSZL	0.11 ± 0.003	1.81 ± 0.162	≈

$*$: best entry of cost
◯: best entry significantly better in cost
≈: otherwise

4.6 Comparison on the KDD Cup 1999 Task: Cost-sensitive and unbalanced Classification

The KDDCup 1999 data set (kdd99) is another real-world cost-sensitive classification task [20]. The task contains an intrusion detection problem for distinguishing the "good" and "bad" connections. Following the usual procedure in literature [10], we extract a random 40% of the 10%-training set for our experiments. The test set accompanied is not used because of the known mismatch between training and test distributions [10]. We take the given cost matrix in the competition for our experiments.[4] This data set is also highly unbalanced. In particular, the size of the majority class is over 8000 times more than the size of the minority class. Therefore, we use the soft cost-sensitive classification and adapt weighted error rate as the basic criterion in this comparison.

The results are listed in Table 17. While the cost-sensitive classification algorithm OSR achieves the lowest test cost, other algorithms (soft, hard, or regular) all result in similar performance. The reason of the similar performance is because all the algorithms are of error rate less than 1% and are thus of low weighted error and low cost. That is, the data set is easy to classify, and there is almost no room for improvements.

To further compare the performance of the algorithms, we consider a more challenging version of the real-world task. The version is called kdd99-balanced, which adopted in our previous work [23]. The cost on kdd99-balanced is scaled by the number of examples, which is generated by scaling down the y-th row of the cost matrix by the size of the y-th class.

The results on kdd99-balanced are shown in Table 18, and the t-test are listed in Table 19. All algorithms share the similar cost except CSFT. However, soft cost-sensitive classification (with weighted error as the basic criterion) could reach the lower weighted error and the better G-mean significantly. The results again demonstrate the usefulness of soft cost-sensitive classification in reaching low cost *and* low weighted error on this real-world task.

TABLE 18
average test results on kdd99-balanced

	weighted error	G-mean (%)	cost ($\cdot 10^{-6}$)
OSR	9.10 ± 0.56	38.98 ± 2.78	1.68 ± 0.11
soft-OSR	7.97 ± 0.69	42.77 ± 4.14	1.74 ± 0.15
CSOVO	8.78 ± 0.60	40.19 ± 3.12	1.63 ± 0.10
soft-CSOVO	8.28 ± 0.55	41.63 ± 3.35	1.68 ± 0.13
CSFT	8.86 ± 0.41	36.98 ± 1.51	2.18 ± 0.09
soft-CSFT	8.87 ± 0.65	43.02 ± 3.63	1.66 ± 0.12
CSZL	9.84 ± 0.59	36.47 ± 2.51	1.85 ± 0.06
soft-CSZL	9.02 ± 0.73	41.39 ± 4.01	1.79 ± 0.14

5 CONCLUSIONS

We have explored the trade-off between the cost and the error rate in cost-sensitive classification tasks, and have identified the practical needs to reach both low cost and low error rate. Based on the trade-off,

4. http://www.kdd.org/kddcup/site/1999/files/awkscript.htm

TABLE 19
t-test on kdd99-balanced for weighted error, G-mean and cost

	weighted error	G-mean	cost
OSR	◯	◯	≈
soft-OSR	∗	≈	≈
CSOVO	◯	≈	∗
soft-CSOVO	≈	≈	≈
CSFT	◯	◯	◯
soft-CSFT	◯	∗	≈
CSZL	◯	◯	◯
soft-CSZL	◯	≈	◯

∗ : best entry of the column
◯ : best entry being significantly better
≈ : otherwise

we have proposed a simple and novel methodology between traditional regular classification and modern cost-sensitive classification. The proposed methodology, soft cost-sensitive classification, takes both the cost and the error (or the weighted error) into account by a multicriteria optimization problem. By using the weighted sum approach to solving the optimization problem, the proposed methodology allows immediate improvements of existing cost-sensitive classification algorithms in terms of similar or sometimes lower costs, and of lower errors. The significant improvements have been observed on a broad range of benchmark and real-world tasks in our extensive experimental study.

Our work reveals a new insight for cost-sensitive classification in machine learning and data mining: Feeding in the exact cost information for the machines to learn may not be the best approach, much like how fitting the provided data faithfully without regularization may lead to overfitting. Our work takes the error rates to "regularize" the cost information and leads to better performance. Another interesting direction for future research is to consider other types of regularization on the cost information.

6 Acknowledgments

The authors thank Yao-Nan Chen, Ku-Chun Chou, Chih-Han Yu and the anonymous reviewers for valuable comments. This work was supported by the National Science Council (NSC 100-2628-E-002-010 and NSC 100-2120-M-001-003-CC1) of Taiwan.

References

[1] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan Kaufmann, 2011.
[2] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.
[3] T.-K. Jan, H.-T. Lin, H.-P. Chen, T.-C. Chern, C.-Y. Huang, C.-Y. Huang, C.-W. Chung, Y.-J. Li, Y.-C. Chuang, L.-L. Li, Y.-J. Chan, J.-K. Wang, Y.-L. Wang, C.-H. Lin, and D.-W. Wang, "Cost-sensitive classification on pathogen species of bacterial meningitis by surface enhanced Raman scattering," in *Proc. IEEE BIBM*, 2011, pp. 406–409.
[4] K. Schleifer, "Classification of bacteria and archaea: past, present and future," *Syst. Appl. Microbiol.*, vol. 32, no. 8, pp. 533–542, 2009.
[5] A. Freitas, "Building cost-sensitive decision trees for medical applications," *AI Comm.*, vol. 24, no. 3, pp. 285–287, 2011.
[6] W. Lee, W. Fan, M. Miller, S. Stolfo, and E. Zadok, "Toward cost-sensitive modeling for intrusion detection and response," *JCS*, vol. 10, no. 1/2, pp. 5–22, 2002.
[7] M. Tan, "Cost-sensitive learning of classification knowledge and its applications in robotics," *ML*, vol. 13, no. 1, pp. 7–33, 1993.
[8] W. Fan, W. Lee, S. Stolfo, and M. Miller, "A multiple model cost-sensitive approach for intrusion detection," in *Proc. ECML*, 2000, pp. 142–154.
[9] Y. Sun, M. Kamel, A. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *PR*, vol. 40, no. 12, pp. 3358–3378, 2007.
[10] N. Abe, B. Zadrozny, and J. Langford, "An iterative method for multi-class cost-sensitive learning," in *Proc. SIGKDD*, 2004, pp. 3–11.
[11] A. Beygelzimer, V. Daniand, T. Hayes, J. Langford, and B. Zadrozny, "Error limiting reductions between classification tasks," in *Proc. ICML*, 2005, pp. 49–56.
[12] A. Beygelzimer, J. Langford, and P. Ravikumar, "Multiclass classification with filter trees," 2007, downloaded from http://hunch.net/~jl.
[13] H.-T. Lin, "A simple cost-sensitive multiclass classification algorithm using one-versus-one comparisons," 2010, downloaded from http://www.csie.ntu.edu.tw/~htlin/paper/doc/csovo.pdf.
[14] A. Bernstein, F. Provost, and S. Hill, "Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification," *IEEE TKDE*, vol. 17, no. 4, pp. 503–518, 2005.
[15] H.-H. Tu and H.-T. Lin, "One-sided support vector regression for multiclass cost-sensitive classification," in *Proc. ICML*, 2010, pp. 1095–1102.
[16] T.-K. Jan, "A comparison of methods for cost-sensitive support vector machines," Master's thesis, National Taiwan University, 2010.
[17] S. Rosset, "Value weighted analysis: Building prediction models for data with observation," 2002, downloaded from http://www.tau.ac.il/~saharon/.
[18] C. Hillermeier, *Nonlinear multiobjective optimization*. Birkhauser, 2001.
[19] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE TAC*, vol. 8, no. 1, pp. 59–60, 1963.
[20] S. D. Bay, "UCI KDD archive. Department of Information and Computer Sciences, University of California, Irvine," 2000, downloaded from http://kdd.ics.uci.edu/.
[21] Z.-H. Zhou and X.-Y. Liu, "On multi-class cost-sensitive learning," in *Proc. AAAI*, 2006, pp. 567–572.
[22] P. Domingos, "MetaCost: A general method for making classifiers cost-sensitive," in *Proc. SIGKDD*, 1999, pp. 155–164.
[23] T.-K. Jan, D.-W. Wang, C.-H. Lin, and H.-T. Lin, "A simple methodology for soft cost-sensitive classification," in *Proc. SIGKDD*, 2012, pp. 141–149.
[24] J. Langford and A. Beygelzimer, "Sensitive error correcting output codes," in *Proc. COLT*, 2005, pp. 158–172.
[25] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. IJCAI*, 2001, pp. 973–978.
[26] B. Zadrozny, J. Langford, and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting," in *Proc. ICDM*, 2003, pp. 435–442.
[27] D. D. Margineantu, "Methods for cost-sensitive learning," Ph.D. dissertation, Oregon State University, 2001.
[28] H.-H. Tu, "Regression approaches for multi-class cost-sensitive classification," Master's thesis, National Taiwan University, 2009.
[29] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE TNN*, vol. 13, no. 2, pp. 415–425, 2002.
[30] J. Horn, N. Nafpliotis, and D. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. IEEE WCCI*, 1994, pp. 82–87.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE TEC*, vol. 6, no. 2, pp. 182–197, 2002.
[32] D. Corne, N. Jerram, J. Knowles, and M. Oates, "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proc. GECCO*, 2001.
[33] I. Das and J. E. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems," *Struct. Multidiscip. Opti.*, vol. 14, no. 1, pp. 63–69, 1996.
[34] D. Sculley, "Combined regression and ranking," in *Proc. SIGKDD*, 2010, pp. 979–988.
[35] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning from Data: A Short Course*. AMLBook, 2012.
[36] V. N. Vapnik, *Statistical Learning Theory*. Wiley, 1998.
[37] H.-T. Lin and L. Li, "Support vector machinery for infinite ensemble learning," *JMLR*, vol. 9, no. 2, pp. 285–312, 2008.
[38] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM TIST*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[39] S. Hettich, C. L. Blake, and C. J. Merz, "UCI repository of machine learning databases," 1998.
[40] J. J. Hull, "A database for handwritten text recognition research," *IEEE TPAMI*, vol. 16, no. 5, pp. 550–554, 1994.
[41] A. Campion and P. Kambhampati, "Surface enhanced Raman scattering," *Chem. Soc. Rev.*, vol. 27, no. 4, pp. 241–250, 1998.
[42] R. Alejo, V. Garca, J. M. Sotoca, R. A. Mollineda, and J. S. Snchez, "Improving the performance of the rbf neural networks trained with imbalanced samples," pp. 162–169, 2007.