Information Sciences 381 (2017) 92–103

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification

Wonji Lee[a], Chi-Hyuck Jun[a,*], Jong-Seok Lee[b]

[a] Department of Industrial & Management Engineering, POSTECH, Pohang 37673, South Korea
[b] Department of Industrial Engineering, Sungkyunkwan University, Suwon 16419, South Korea

ARTICLE INFO

Article history:
Received 9 March 2016
Revised 18 October 2016
Accepted 25 November 2016
Available online 25 November 2016

Keywords:
Class imbalance
SVM
Instance categorization
AdaBoost
Weight adjustment

ABSTRACT

To address class imbalance in data, we propose a new weight adjustment factor that is applied to a weighted support vector machine (SVM) as a weak learner of the AdaBoost algorithm. Different factor scores are computed by categorizing instances based on the SVM margin and are assigned to related instances. The SVM margin is used to define borderline and noisy instances, and the factor scores are assigned to only borderline instances and positive noise. The adjustment factor is then employed as a multiplier to the instance weight in the AdaBoost algorithm when learning a weighted SVM. Using 10 real class-imbalanced datasets, we compare the proposed method to a standard SVM and other SVMs combined with various sampling and boosting methods. Numerical experiments show that the proposed method outperforms existing approaches in terms of F-measure and area under the receiver operating characteristic curve, which means that the proposed method is useful for relaxing the class-imbalance problem by addressing well-known degradation issues such as overlap, small disjunct, and data shift problems.

1. Introduction

Class imbalance is one of the most challenging problems in data mining and machine learning [38]. It occurs when the number of instances in one class (the minority class) is significantly small compared to that in the other class (the majority class). The minority class is typically of interest, and hence, is called a positive (+) class, while the majority class is called a negative (−) class. We use the pairs of terms {positive, negative} and {minority, majority} interchangeably throughout this paper. Although we focus on a method for solving a binary classification problem, the proposed method can be extended and applied to multiclass classification problems by constructing multiple classifiers based on the one-versus-all scheme, as demonstrated in previous studies [15,17,19,36].

Most standard learning algorithms use global evaluation measures, such as overall accuracy; thus, standard classifiers tend to be biased toward negative classes and perform poorly when classifying positive instances [12]. In addition, classifiers might identify positive instances as noise and ignore them in the learning process [20]. The class-imbalance problem occurs frequently in real applications, such as facial age estimation [3], anomaly detection [14], software defect prediction [25], and image annotation [39].

The class-imbalance problem degrades the classification accuracy of traditional classification methods [32]. Six significant factors are known to cause degradation [20]: a) overlap between classes, b) borderline instances, c) areas with small

* Corresponding author. Fax: +82 54 279 2870.
E-mail addresses: wonji0129@postech.ac.kr (W. Lee), chjun@postech.ac.kr (C.-H. Jun), jongseok@skku.edu (J.-S. Lee).

http://dx.doi.org/10.1016/j.ins.2016.11.014
0020-0255/

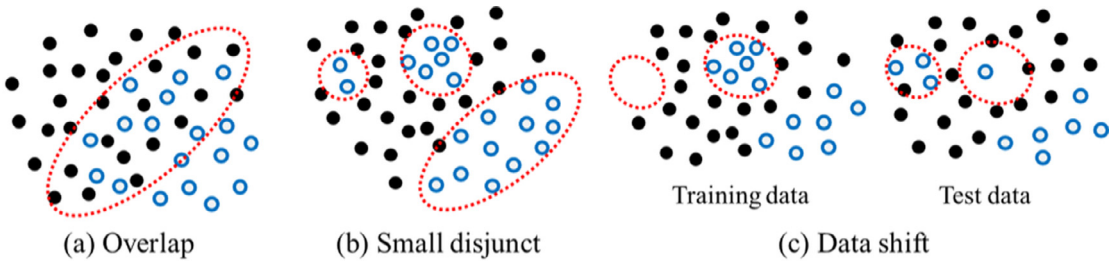

Fig. 1. Principal degradation issues in class-imbalanced data.

disjuncts, d) noisy data, e) low density and lack of information in the training data, and f) possible differences in the distribution of training and test data (or data shift). These factors can be grouped into three principal issues: class overlap (a and b), small disjuncts (c and d), and data shift (e and f). The dotted ellipses in Fig. 1 illustrate these degradation issues. Overlap involves data classes being not completely separable, as shown in Fig. 1(a). This problem occurs frequently in class-imbalanced data [6] and is a principal cause of degrading classification accuracy [26]. As can be seen in Fig. 1(b), the small disjunct problem occurs when each of the several small clusters contains few, but not negligible, positive instances. If the clusters are scattered, constructing proper decision boundaries is a daunting task. This becomes more challenging with class-imbalanced data because misclassifying the disjuncts of a positive class greatly affects the performance (true positive rate or sensitivity) of a classifier. The data shift problem occurs when the distributions of training and test data differ (Fig. 1(c)). This problem becomes severe in class-imbalanced data because the low density of positive classes can result in a more distinct difference between the distributions of the training and test data [11,16,21].

Approaches to the class-imbalance problem are typically categorized as data or algorithm levels. The former typically involves data pre-processing, which aims at balancing highly skewed class distributions using various sampling methods [27,35]. Although this approach is widely used as a simple solution to the class-imbalance problem, most data pre-processing methods have a common limitation due to the random sampling. A pre-processed dataset can be completely different from the original, thereby resulting in a severely biased data distribution. The latter, i.e., algorithm-level approaches, involves modifying algorithms such as cost-sensitive learning and ensemble schemes that include boosting and bagging [9,16]. Cost-sensitive learning is based on the assumption that misclassification costs are already known [5,7]. However, we usually have no prior information regarding these costs; thus, we encounter difficulty when determining costs. If positive instances are sparse, cost-sensitive learning may not construct appropriate decision boundaries [22]. Among ensemble approaches, boosting is frequently used to relax the class-imbalance problem because this approach can detect small disjuncts effectively by highlighting the misclassified instances at each iteration and can reduce the bias from data by combining classification results from several weak learners.

Although various classification techniques can be embedded into a boosting algorithm, studies using support vector machines (SVMs) have reported good classification results. The simplest form of a boosted SVM is to apply SVMs to a standard AdaBoost scheme. However, this form was found to be ineffective for significantly improving classification performance for two-class problems; thus, this form is used for multiclass problems [15] or is applied to large-scale data [30] due to its relatively simple structure. In several studies, a boosting algorithm with an SVM has been advanced by increasing the diversity of weak learners by adjusting the SVM kernel parameters [17,19,37]. Another group of studies used cost-sensitive SVMs as weak AdaBoost learners [23,36]. A previous study [36] used a linear kernel in an SVM with no parameters to weaken the classifier, rather than controlling the kernel parameters. They also set a stopping criterion to terminate the iterations when the G-mean [20] of the test data began to decrease. Rather than using a cost-sensitive weak learner, the boosting framework itself can be made cost-sensitive, typically by defining an upper bound of the total misclassification cost of a strong classifier and then finding instance weights and the significance of a weak learner. Cost-sensitive AdaBoost algorithms are generally used [8,31,33], with AdaC2 demonstrating good and relatively stable performance [31].

In this paper, we propose an algorithm-level approach that combines a weighted SVM with a boosting algorithm to address the class-imbalance problem. Our method is distinct from the existing boosting approaches in that we introduce a new weight adjustment factor in the proposed algorithm. The adjustment factor is inspired by a simple notion that a trained SVM provides information regarding bounded and unbounded support vectors. Since these support vectors are located near a decision boundary, we expect to address the class overlap issue by carefully considering these instances. In addition, we can identify which instances are misclassified outside the SVM margin. This information enables us to investigate the presence of small disjuncts. The weight adjustment factor is designed to assign more or less weight to instances corresponding to these two degradation issues. In addition, the proposed method is expected to address the data shift issue by adopting the AdaBoost framework.

The remainder of this paper is organized as follows. In Section 2, we describe the proposed method and its main components, i.e., instance categorization based on an SVM, the weight adjustment factor, and the boosting algorithm. Numerical experiments are presented in Section 3. Section 4 concludes the paper.

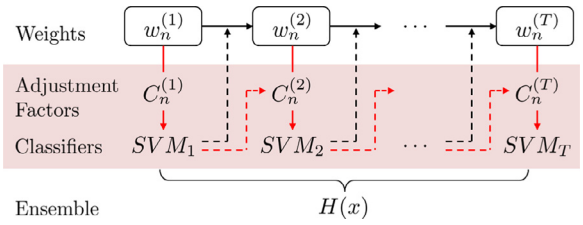

Fig. 2. Scheme of the proposed weight adjustment in SVM-AdaBoost based on instance categorization.

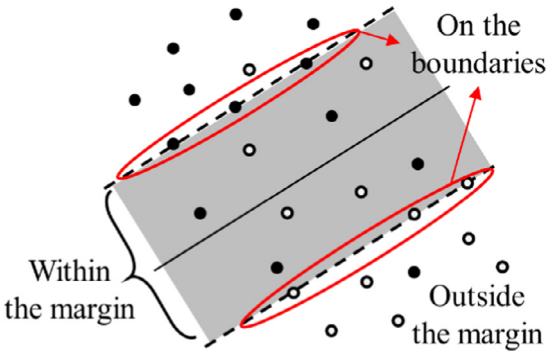

Fig. 3. Instance categorization based on the margin and boundaries of an SVM.

2. Proposed method: weight adjustment in SVM AdaBoost based on instance categorization

The proposed method works within the AdaBoost framework, with the weak learner being a weighted SVM (Fig. 2). The key idea is to categorize instances based on the margin boundaries of the SVM and to compute weight adjustment factor scores for individual instances considering their categories. We then use the scores to train a weighted SVM in the next iteration, together with the instance weights in the AdaBoost algorithm. In Fig. 2, T is the total number of iterations and $\{w_n^{(t)}\}_{n=1}^N$ and $\{C_n^{(t)}\}_{n=1}^N$ are the sets of instance weights and factor scores at the t-th iteration, respectively, where N is the number of instances.

2.1. Instance categorization and weight adjustment

Once an SVM is trained, the location of each instance is determined, i.e., whether it is within the margin of two boundaries, on the boundaries, or outside the margin (Fig. 3). An instance within the margin is referred to as a "bounded support vector" (BSV), while that on the margin boundaries is referred to as a "support vector" (SV). Since BSVs and SVs are near the separating hyperplane, we refer to them as "borderline instances." The number of SVs is not related to the total number of instances, whereas the number of BSVs scales at least linearly with the total number of instances [1]. Thus, we consider them separately because the size difference between the classes is important in imbalanced data classification. An instance outside the margin is either a "safe instance" or "noise" according to its classification result, where a safe instance is classified correctly and noise is not.

We divide all instances into four categories (BSV, SV, positive noise, and others) based on their locations, classes, and classification results. Borderline instances corresponding to the BSV and SV categories are of interest because the class-imbalance ratio in the overlap region has the most significant influence on classification [10,28]. We expect that paying more attention to such instances will help address the overlap problem, possibly resulting in better classification. Positive noise, which is a false negative located outside the margin, is also of interest because we expect to detect the small disjuncts of a positive class. Therefore, we design the weight adjustment factor for instances that only belong to the first three categories. Table 1 shows our instance categorization. The BSV and SV categories must be considered as either positive or negative. Instances outside the margin other than positive noises are categorized as others because we take no action with such instances since they are useless when addressing the small disjunct issue. Note that the weight adjustment factor is only applied to the boldfaced instances in Table 1.

If the n-th instance is categorized as a BSV at the t-th iteration, then the following adjustment factor is applied:

$$C_n^{(t)} = \begin{cases} \dfrac{\#(BSV)}{2\#(negative\ BSV)}, & \text{if } y_n = - \\ \dfrac{\#(BSV)}{2\#(positive\ BSV)}, & \text{if } y_n = + \end{cases} \tag{1}$$

Table 1
Instance categorization based on location, class, and classification result.

Location	Class and classification result				Degradation issue
	Positive		Negative		
	Correctly classified	Misclassified	Correctly classified	Misclassified	
Within the margin	**Positive BSV**		**Negative BSV**		Overlap
On the boundaries	**Positive SV**		**Negative SV**		
Outside the margin	Safe instance	**Positive noise**	Safe instance	Negative noise	Small disjunct

where y_n is the class variable of the n-th instance. Similarly, the following is applied if the n-th instance is categorized as an SV at the t-th iteration.

$$C_n^{(t)} = \begin{cases} \dfrac{\#(SV)}{2\#(negative\ SV)}, & \text{if } y_n = - \\ \dfrac{\#(SV)}{2\#(positive\ SV)}, & \text{if } y_n = + \end{cases} \tag{2}$$

In fact, the factor structure in Eqs. (1) and (2) is similar to the cost structure of many other cost-sensitive learning methods. For both BSV and SV, the factor score is inversely proportional to the number of instances belonging to each class for both positive and negative classes, respectively. Note that $\min\{C_n^{(t)}\ for\ y_n = -,\ \ C_n^{(t)}\ for\ y_n = +\} < 1$ and $\max\{C_n^{(t)}\ for\ y_n = -,\ \ C_n^{(t)}\ for\ y_n = +\} > 1$. For instances falling into the BSV and SV categories, the positive class instances take a value of $C_n^{(t)}$ greater than one, whereas a $C_n^{(t)}$ value less than one is assigned to negative class instances. The score difference between two classes increases as the class imbalance among BSV or SV instances becomes severe. For both the BSV and SV categories, we assign more weight to positive class instances. This implies that the BSV and SV instances in the positive class are more important for better classification than those in the negative class. Regardless of the classification results (correct classification or misclassification), we apply the same factor scores to the BSV instances (the SV instances are always classified correctly) because the classification of borderline BSV and SV instances can be changed easily during iterations as they are located near the decision boundary. Nonetheless, we can still pay more attention to the misclassified BSV instances by assigning a greater weight ($w_n^{(t)}$) in the AdaBoost algorithm.

To detect the small disjuncts of a positive class, we also assign a factor score (greater than one) to positive noises. Accordingly, the factor score is chosen as the exponential of the proportion of positive noisy instances among the total positive instances, as shown in Eq. (3).

$$C_n^{(t)} = \exp\left(\frac{\#(positive\ noisy\ instances)}{\#(total\ positive\ instances)}\right) \tag{3}$$

As the proportion of positive noise instances increases, we assign a larger factor score to those instances with the intention of detecting positive small disjuncts, because we believe that small disjuncts appear when the proportion of positive noise instances is relatively large. If a misclassified positive instance is actual noise, its weight ($w_n^{(t)}$) shows nearly no change because its factor score is very close to one, as we adjust the instance weight by multiplying the factor score. Note that the exponential function in Eq. (3) results in the factor score becoming greater than one and less than e (approximately 2.7). The lower bound involves no positive noise (case of one), whereas all positive instances are misclassified (case of e) for the upper bound. Since the weak learner in the proposed method is an SVM, reaching the upper bound is nearly impossible in practice and the adjustment factor in Eq. (3) is nearly linear, despite it being an exponential function. The proposed learning algorithm is described in the following subsection.

2.2. Learning algorithm

When a weighted SVM as a weak learner is trained at each iteration, we update the categorization of instances and evaluate the corresponding factor scores. We then use the scores to adjust the instance weights when learning the next classifier in the subsequent iteration. This adjustment is performed by multiplying the instance weights by the factor scores.

With the exception of instance categorization and weight adjustment, we follow the original AdaBoost framework, as described in the following algorithm. Here, $h_t(x_n)$ is the class of the n-th instance predicted by a weak learner (i.e., the SVM) at the t-th iteration (i.e., the t-th weak learner), α_t is the significance of the t-th weak learner, and $w_n^{(t)}$ is the instance weight of the n-th instance at the t-th iteration. Then, α_t is computed using Eq. (4).

$$\alpha_t = \frac{1}{2}\ln\frac{\sum_{h_t(x_n)=y_n} w_n^{(t)}}{\sum_{h_t(x_n)\neq y_n} w_n^{(t)}} \tag{4}$$

The weight of the n-th instance at the $(t+1)$-th iteration is updated as follows:

$$w_n^{(t+1)} = \frac{w_n^{(t)} \exp\{-\alpha_t h_t(x_n) y_n\}}{Z_t} \tag{5}$$

Table 2
Description of data sets.

	Name	#instances	#variables	Positive ratio (%)
1	Seismic-bumps	2738	20	6.21
2	Hypertension	10,000	24	7.11
3	Cardiotocography	1831	21	9.61
4	Churn	3333	17	14.49
5	Complications	10,814	22	16.08
6	SPECTF heart	267	44	20.60
7	Indian liver patient	583	10	28.64
8	German credit	1000	24	30.00
9	Vertebral column	310	6	32.26
10	Spambase	4601	57	39.40

where Z_t is a normalization constant given by Eq. (6).

$$Z_t = \sum_{n=1}^{N} w_n^{(t)} \exp\{-\alpha_t h_t(x_n) y_n\} \tag{6}$$

Algorithm. SVM AdaBoost with proposed weight adjustment factor.

Training data: $(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)$ *where* $t_n \in \{-1, +1\}$

1. Initialize $w_n^{(1)} = \frac{1}{N}$, $C_n^{(1)} = 1$ $(n = 1, \ldots, N)$
2. For $t = 1, \ldots, T$
   (1) Fit a weighted SVM $h_t(x)$ for training data with weights $\{w_n^{(t)} \cdot C_n^{(t)}\}_{n=1}^N$.
   (2) Update $\{C_n^{(t+1)}\}_{n=1}^N$ using the categorization based on $h_t(x)$.
   (3) Evaluate the significance for the t^{th} iteration using Eq. (4).
   (4) Update the weights of instances $\{w_n^{(t+1)}\}_{n=1}^N$ using Eq. (5).
3. Create the final model by combining components.

$$H(x) = sign\left\{\sum_{t=1}^{T} \alpha_t h_t(x)\right\}$$

Note that the currently evaluated adjustment factor is used temporarily for learning the next weighted SVM. It is not used to update the next factor and the instance weights. The evaluation is performed using only the current learning and its corresponding categorization, while the instance weights at each iteration are updated using the previous weights. This is intended to preserve the theoretical foundations of AdaBoost. Thus, the theoretical upper bound of the error rate of the strong classifier $H(x)$ decreases as the iterations proceed.

3. Performance evaluation

3.1. Datasets

We used 10 class-imbalanced datasets with various positive ratios (6–40%), as shown in Table 2. The datasets were obtained from the University of California, Irvine, Machine Learning Repository [18], and included Seismic-bumps [29], Cardiotocography, SPECTF heart, Indian liver patient, German credit, Vertebral column, and Spambase. The original Cardiotocography dataset has three classes (normal, suspect, and pathologic); however, we used only the normal and pathologic classes to create a two-class classification problem. We established the Hypertension and Complications datasets to predict incidences of hypertension and hypertension complications, respectively. Their raw data originated from the National Health Insurance Corporation in South Korea. The Churn dataset can be obtained from Professor Pardoe's website [24].

3.2. Experimental design

We compared the proposed method to the 10 SVM-based methods listed in Table 3, including a standard SVM (SVM), a cost-sensitive SVM (CS-SVM) [34], SVMs with four data pre-processing methods, and four ensemble methods based on an SVM. For CS-SVM, we set the misclassification cost of each class according to the ratio in Eq. (7) [23,36].

$$\frac{misclassification\ cost\ (positive\ class)}{misclassification\ cost\ (negative\ class)} = \frac{\#(total\ negative\ instances)}{\#(total\ positive\ instances)} \tag{7}$$

The proposed method is designed for imbalanced data classification; therefore, the benchmarking methods must include methods developed for the same purpose. Consequently, we selected four data pre-processing methods and four ensemble methods. The former includes random under-sampling, the neighborhood-cleaning rule [16], the synthetic positive oversampling technique (SMOTE) [4], and borderline SMOTE (BSMOTE) [11], which were combined with an SVM. The ensemble

Table 3
Classification methods in our comparative study.

Form	Name	Description	Parameters
Standard	SVM	A single SVM	γ_{best} by grid search in $2^{[-20:20]}$
Cost-sensitive	CS-SVM	A cost-sensitive SVM	
Data pre-processing	NCR-SVM	SVM applied to NCR processed data	
	RUS-SVM	SVM applied to RUS processed data	
	SMOTE-SVM	SVM applied to SMOTE processed data	
	BSMOTE-SVM	SVM applied to BSMOTE processed data	
Ensemble (Boosting)	Ada-SVM	AdaBoost with SVM	$\gamma_0 = 10^{-5}$ (γ is adjusted when no changes in weights)
	γAda-SVM	γ-adaptive AdaBoost with SVM	
	Ada-CSSVM	AdaBoost with cost-sensitive SVM	
	AdaC2-SVM	AdaC2 with SVM	
	Proposed	Proposed method	

methods were assigned as follows: (1) AdaBoost with SVM (Ada-SVM) [30]; (2) γ-adaptive AdaBoost with SVM (γAda-SVM), automatically adjusting γ in boosting [17]; (3) AdaBoost with CS-SVM (Ada-CSSVM) [36]; and (4) AdaC2 with SVM (AdaC2-SVM) [31]. Note that for the cost-sensitive boosting methods, i.e., Ada-CSSVM and AdaC2-SVM, the cost setting is the same as that for CS-SVM.

For all SVMs in our experiments, we employed the following radial basis function kernel:

$$k(x_i,\ x_j) = \exp\left(\left\|\gamma x_i - x_j{}^2\right\|\right) \tag{8}$$

where γ is the bandwidth parameter that must be predetermined. With the exception of the ensemble methods, we attempted to select the best γ value from a grid search of the γ range between 2^{-20} and 2^{20}, which is wider than that in a previous study [13]. The initial γ value for the ensemble methods, including the proposed method, was set to a small value (i.e., 10^{-5}) to make the SVM a weak learner in the boosting framework. If the accuracy of the SVM component in boosting is 100%, then boosting no longer improves classification accuracy even though the iteration continues. In this case, we decrease the γ value to make the SVM a weak learner. If the accuracy of the subsequent component is still 100%, we stop the iterations. The number of iterations was set to 10 (T= 10) for both the proposed method and the other ensemble methods [31].

Rather than global evaluation measures, we used sensitivity (Sens), specificity (Spec), F-measure with $\beta = 1$ (F-1 measure) [22], and area under the receiver operating characteristic curve (AUC) [2] to evaluate the performance of the classification methods with class-imbalanced data. We repeated five-fold cross-validations 10 times and reported the mean and standard deviation of each performance measure over 50 runs.

3.3. Results

Tables 4–13 show the experimental results for each dataset. The mean of each performance measure over 50 runs is given, and the standard deviation is shown in parentheses.

The classification results obtained using the proposed method appear quite promising. For nearly all datasets, the proposed method performs the best in terms of both F-measure and AUC, while it ranks second only in the experiment with the "Complications" data in terms of F-measure (the proposed method still ranks first in terms of AUC). These results indicate that the proposed method improves classification performance more than the trade-off between sensitivity and specificity.

A single SVM shows the worst performance with relatively lower sensitivity than specificity. CS-SVM, another stand-alone classifier, and the data pre-processing methods show obvious improvement. However, the improvement is not as significant as that of the proposed method. CS-SVM is reversely biased to a positive class for several datasets, probably because its misclassification costs are based on only the number of class instances. The data pre-processing methods sometimes show worse performance than a single SVM because they may distort the original distribution by sampling and generating instances. Furthermore, they appear incapable of solving the data shift problem for the positive class because large differences in sensitivity between the training and test data appear in the experiments with these methods. In contrast, the ensemble methods, e.g., Ada-CSSVM, AdaC2-SVM, and the proposed method, appear to address the data shift issue because even higher performance measure values in the test data than those in the training data are obtained by these methods. However, as can be seen from the results for Ada-SVM and γAda-SVM, the boosting scheme itself, which does not consider class imbalance in data, appears inappropriate for solving the class-imbalance problem because they do not outperform Ada-CSSVM, AdaC2-SVM, and the proposed method. Thus, we conclude that an ensemble approach that considers class imbalance in data can be an effective tool for addressing the data shift issue and solving the class-imbalance problem.

Table 4
The classification results for Seismic-bumps data (positive rate = 6.21%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.5834	1.0000	0.7363	0.7917	0.0000	0.9999	0.0000	0.5000
	(0.0037)	(0.0000)	(0.0020)	(0.0009)	(0.0000)	(0.0000)	(0.0000)	(0.0000)
CS-SVM	0.9237	0.9477	0.6812	0.9357	0.0553	0.9346	0.0540	0.4950
	(0.0009)	(0.0000)	(0.0013)	(0.0003)	(0.0013)	(0.0002)	(0.0012)	(0.0004)
NCR-SVM	0.7178	0.9981	0.8217	0.8580	0.0045	0.9968	0.0083	0.5007
	(0.0058)	(0.0000)	(0.0020)	(0.0014)	(0.0003)	(0.0000)	(0.0010)	(0.0001)
RUS-SVM	0.9386	0.5070	0.2009	0.7228	0.8516	0.4717	0.1736	0.6616
	(0.0010)	(0.0043)	(0.0002)	(0.0004)	(0.0066)	(0.0047)	(0.0011)	(0.0018)
SMOTE-SVM	0.9181	0.9135	0.5712	0.9158	0.6528	0.6846	0.2029	0.6687
	(0.0002)	(0.0003)	(0.0031)	(0.0001)	(0.0079)	(0.0006)	(0.0019)	(0.0020)
BSMOTE-SVM	0.7152	0.9667	0.6458	0.8410	0.6365	0.7246	0.2196	0.6806
	(0.0005)	(0.0001)	(0.0014)	(0.0001)	(0.0221)	(0.0007)	(0.0039)	(0.0053)
Ada-SVM	1.0000	1.0000	1.0000	1.0000	0.0192	0.9846	0.0303	0.5019
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0010)	(0.0000)	(0.0021)	(0.0003)
γAda-SVM	1.0000	1.0000	1.0000	1.0000	0.0192	0.9846	0.0303	0.5019
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0010)	(0.0000)	(0.0021)	(0.0003)
Ada-CSSVM	1.0000	1.0000	1.0000	1.0000	0.6716	0.3493	0.0895	0.5104
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.1571)	(0.1336)	(0.0026)	(0.0007)
AdaC2-SVM	1.0000	1.0000	1.0000	1.0000	0.6902	0.5461	0.1426	0.6182
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.1670)	(0.0610)	(0.0059)	(0.0071)
Proposed	1.0000	1.0000	1.0000	1.0000	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)

Table 5
The classification results for Hypertension data (positive rate = 7.11%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.9234	1.0000	0.9501	0.9617	0.0611	**0.9999**	0.1148	0.5305
	(0.1201)	(0.0000)	(0.1275)	(0.0300)	(0.0005)	(0.0000)	(0.0019)	(0.0001)
CS-SVM	0.9953	0.9921	0.9841	0.9937	0.0800	0.9916	0.1258	0.5358
	(0.0074)	(0.0211)	(0.0825)	(0.0133)	(0.0622)	(0.0228)	(0.0014)	(0.0024)
NCR-SVM	0.7991	0.9994	0.8298	0.8992	0.0577	0.9993	0.1074	0.5285
	(0.1806)	(0.0000)	(0.1913)	(0.0451)	(0.0010)	(0.0000)	(0.0032)	(0.0002)
RUS-SVM	0.7665	0.6016	0.2199	0.6841	**0.7606**	0.6014	0.2182	0.6810
	(0.0012)	(0.0003)	(0.0000)	(0.0001)	(0.0020)	(0.0004)	(0.0005)	(0.0004)
SMOTE-SVM	0.9677	0.9436	0.8884	0.9556	0.1714	0.9384	0.1401	0.5549
	(0.0135)	(0.0405)	(0.1539)	(0.0252)	(0.1264)	(0.0389)	(0.0034)	(0.0064)
BSMOTE-SVM	0.9999	0.9962	0.9756	0.9980	0.1209	0.9585	0.1450	0.5397
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0007)	(0.0000)	(0.0008)	(0.0002)
Ada-SVM	0.6771	0.8020	0.3176	0.7395	0.3729	0.7773	0.1736	0.5751
	(0.0003)	(0.0000)	(0.0001)	(0.0001)	(0.0015)	(0.0002)	(0.0003)	(0.0004)
γAda-SVM	0.6771	0.8020	0.3176	0.7395	0.3729	0.7773	0.1736	0.5751
	(0.0003)	(0.0000)	(0.0001)	(0.0001)	(0.0015)	(0.0002)	(0.0003)	(0.0004)
Ada-CSSVM	0.2665	0.8803	0.1694	0.5734	0.2596	0.8671	0.1518	0.5633
	(0.0602)	(0.0111)	(0.0029)	(0.0049)	(0.0601)	(0.0111)	(0.0038)	(0.0052)
AdaC2-SVM	0.9381	0.9915	0.9195	0.9648	0.1931	0.9328	0.1865	0.5629
	(0.0761)	(0.0010)	(0.0823)	(0.0233)	(0.0008)	(0.0001)	(0.0006)	(0.0002)
Proposed	0.6596	0.7042	0.2388	0.6819	0.6599	0.7042	**0.2386**	**0.6820**
	(0.0001)	(0.0000)	(0.0000)	(0.0000)	(0.0018)	(0.0001)	(0.0006)	(0.0004)

Among the benchmarking methods, Ada-CSSVM and AdaC2-SVM can be considered the most similar to the proposed method because they are ensemble methods that apply different costs to instances in the learning components and boosting, respectively. The proposed method demonstrates higher F-measure and AUC values than these benchmarking methods; thus, the key idea of the proposed method (instance categorization and weight adjustment) is considered effective for solving the class-imbalance problem. Furthermore, different misclassification costs based on only the number of instances can be a limitation, as seen in the results for CS-SVM.

Note that the standard deviation values of the proposed method are (sometimes significantly) less than those of the other methods, which implies that the proposed method performs stably. In contrast, the data pre-processing methods show relatively high standard deviations due to the randomness in their algorithms; data are sampled randomly or artificial data are generated using randomly chosen nearest neighbors, which can easily result in destroying the original data distribu-

Table 6
The classification results for Cardiotocography data (positive rate = 9.61%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.8763	0.9958	0.9137	0.9360	0.8085	**0.9931**	0.8607	0.9008
	(0.0040)	(0.0000)	(0.0021)	(0.0010)	(0.0074)	(0.0000)	(0.0039)	(0.0019)
CS-SVM	0.9872	0.9743	0.8884	0.9807	**0.9273**	0.9717	0.8449	0.9495
	(0.0002)	(0.0002)	(0.0035)	(0.0002)	(0.0077)	(0.0003)	(0.0032)	(0.0016)
NCR-SVM	0.9286	0.9903	0.9190	0.9594	0.8603	0.9894	0.8755	0.9248
	(0.0045)	(0.0000)	(0.0020)	(0.0012)	(0.0064)	(0.0000)	(0.0026)	(0.0016)
RUS-SVM	0.9530	0.9433	0.7671	0.9481	0.9263	0.9393	0.7414	0.9328
	(0.0016)	(0.0002)	(0.0022)	(0.0004)	(0.0037)	(0.0005)	(0.0044)	(0.0008)
SMOTE-SVM	0.9974	0.9921	0.9646	0.9948	0.8279	0.9880	0.8534	0.9079
	(0.0004)	(0.0005)	(0.0084)	(0.0004)	(0.0107)	(0.0005)	(0.0036)	(0.0025)
BSMOTE-SVM	0.9915	0.9868	0.9431	0.9891	0.8459	0.9808	0.8354	0.9134
	(0.0001)	(0.0019)	(0.0150)	(0.0006)	(0.0077)	(0.0030)	(0.0168)	(0.0017)
Ada-SVM	0.9958	0.9894	0.9511	0.9926	0.8952	0.9713	0.8459	0.9332
	(0.0015)	(0.0002)	(0.0024)	(0.0004)	(0.0026)	(0.0124)	(0.0361)	(0.0034)
γAda-SVM	0.9958	0.9894	0.9511	0.9926	0.8952	0.9713	0.8459	0.9332
	(0.0015)	(0.0002)	(0.0024)	(0.0004)	(0.0026)	(0.0124)	(0.0361)	(0.0034)
Ada-CSSVM	0.8998	0.9792	0.8599	0.9395	0.8472	0.9765	0.8180	0.9119
	(0.0022)	(0.0001)	(0.0022)	(0.0005)	(0.0062)	(0.0002)	(0.0060)	(0.0017)
AdaC2-SVM	0.9615	0.9942	0.9540	0.9779	0.9018	0.9875	0.8927	0.9447
	(0.0006)	(0.0000)	(0.0009)	(0.0002)	(0.0023)	(0.0000)	(0.0022)	(0.0007)
Proposed	0.9283	0.9919	0.9261	0.9601	0.9244	0.9915	**0.9223**	**0.9580**
	(0.0001)	(0.0000)	(0.0001)	(0.0000)	(0.0014)	(0.0000)	(0.0014)	(0.0004)

Table 7
The classification results for Churn data (positive rate = 14.49%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.3664	0.9978	0.5293	0.6821	0.2722	0.9937	0.4141	0.6329
	(0.0038)	(0.0000)	(0.0052)	(0.0010)	(0.0028)	(0.0000)	(0.0044)	(0.0007)
CS-SVM	0.7940	0.9097	0.6789	0.8519	0.4852	0.8674	0.4284	0.6763
	(0.0202)	(0.0001)	(0.0064)	(0.0047)	(0.0025)	(0.0009)	(0.0026)	(0.0007)
NCR-SVM	0.4382	0.9797	0.5616	0.7089	0.3749	0.9771	0.4961	0.6760
	(0.0010)	(0.0000)	(0.0007)	(0.0002)	(0.0020)	(0.0001)	(0.0023)	(0.0005)
RUS-SVM	0.5317	0.8624	0.4560	0.6971	0.4978	0.8563	0.4295	0.6770
	(0.0178)	(0.0067)	(0.0042)	(0.0029)	(0.0057)	(0.0072)	(0.0049)	(0.0014)
SMOTE-SVM	0.9887	0.9908	0.9689	0.9897	0.2013	0.9525	0.2686	0.5769
	(0.0366)	(0.0018)	(0.0397)	(0.0137)	(0.0080)	(0.0008)	(0.0038)	(0.0010)
BSMOTE-SVM	0.9683	0.9595	0.8785	0.9639	0.2213	0.8753	0.2170	0.5483
	(0.0003)	(0.0005)	(0.0043)	(0.0004)	(0.0128)	(0.0031)	(0.0112)	(0.0012)
Ada-SVM	0.2182	0.9974	0.3532	0.6078	0.1972	**0.9971**	0.3235	0.5971
	(0.0009)	(0.0000)	(0.0017)	(0.0002)	(0.0016)	(0.0000)	(0.0033)	(0.0004)
γAda-SVM	0.2182	0.9974	0.3532	0.6078	0.1972	**0.9971**	0.3235	0.5971
	(0.0009)	(0.0000)	(0.0017)	(0.0002)	(0.0016)	(0.0000)	(0.0033)	(0.0004)
Ada-CSSVM	0.5923	0.9157	0.5696	0.7540	0.5401	0.9121	0.5250	0.7261
	(0.0048)	(0.0072)	(0.0026)	(0.0002)	(0.0032)	(0.0007)	(0.0030)	(0.0009)
AdaC2-SVM	0.9751	0.9924	0.9655	0.9837	0.4617	0.9049	0.4563	0.6833
	(0.0002)	(0.0000)	(0.0002)	(0.0001)	(0.0023)	(0.0001)	(0.0024)	(0.0007)
Proposed	0.5812	0.9234	0.5717	0.7523	**0.6079**	0.9121	**0.5720**	**0.7600**
	(0.0002)	(0.0000)	(0.0002)	(0.0000)	(0.0044)	(0.0006)	(0.0029)	(0.0010)

tion. In conclusion, the proposed method appears successful in addressing the three degradation issues in imbalanced data classification, i.e., overlap, small disjunct, and data shift.

4. Discussion and conclusion

The class-imbalance problem occurs when one class is associated with rare events; thus, the number of instances in the class is smaller (or significantly smaller) than that in the other class. Overlap, small disjunct, and data shift are the principal issues that degrade the accuracy of classifiers in class-imbalanced data. Such degradation issues in class-imbalanced data are known; however, they remain challenging.

To address the principal degradation issues in imbalanced data classification, we proposed a new weight adjustment factor applied to a weighted SVM as a weak learner of the AdaBoost algorithm. Different factor scores were computed by categorizing instances based on the SVM margin and applied to related instances in the learning process. The SVM margin

Table 8
The classification results for Complications data (positive rate = 16.08%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.0431	1.0000	0.0634	0.5216	0.0001	**0.9999**	0.0001	0.5000
	(0.0322)	(0.0000)	(0.0698)	(0.0081)	(0.0000)	(0.0000)	(0.0000)	(0.0000)
CS-SVM	0.8005	0.3622	0.3505	0.5814	**0.6873**	0.3516	0.2606	0.5195
	(0.0273)	(0.1038)	(0.0842)	(0.0473)	(0.0901)	(0.0813)	(0.0033)	(0.0002)
NCR-SVM	0.4551	0.9831	0.4886	0.7191	0.0280	0.9782	0.0456	0.5031
	(0.0581)	(0.0001)	(0.0712)	(0.0134)	(0.0005)	(0.0002)	(0.0011)	(0.0001)
RUS-SVM	0.6627	0.4580	0.2954	0.5604	0.6354	0.4531	**0.2822**	0.5443
	(0.0345)	(0.0513)	(0.0008)	(0.0010)	(0.0382)	(0.0505)	(0.0010)	(0.0006)
SMOTE-SVM	0.9523	0.9424	0.9150	0.9474	0.0818	0.9345	0.0526	0.5082
	(0.0405)	(0.0582)	(0.1238)	(0.0478)	(0.0883)	(0.0548)	(0.0188)	(0.0012)
BSMOTE-SVM	0.9880	0.9828	0.9667	0.9854	0.0922	0.9117	0.1117	0.5019
	(0.0234)	(0.0309)	(0.0668)	(0.0270)	(0.0266)	(0.0235)	(0.0032)	(0.0001)
Ada-SVM	0.5400	0.4600	0.1493	0.5000	0.4400	0.5600	0.1226	0.5000
	(0.0403)	(0.0403)	(0.0031)	(0.0000)	(0.0602)	(0.0602)	(0.0049)	(0.0000)
γAda-SVM	0.5400	0.4600	0.1493	0.5000	0.4400	0.5600	0.1226	0.5000
	(0.0403)	(0.0403)	(0.0031)	(0.0000)	(0.0602)	(0.0602)	(0.0049)	(0.0000)
Ada-CSSVM	0.4277	0.6585	0.2626	0.5431	0.3981	0.6599	0.2457	0.5290
	(0.0134)	(0.0104)	(0.0025)	(0.0038)	(0.0148)	(0.0108)	(0.0021)	(0.0025)
AdaC2-SVM	0.4363	0.8330	0.3780	0.6347	0.2342	0.7998	0.2010	0.5170
	(0.0554)	(0.0058)	(0.0515)	(0.0214)	(0.0082)	(0.0043)	(0.0013)	(0.0003)
Proposed	0.3669	0.7584	0.2792	0.5626	0.3671	0.7584	0.2791	**0.5627**
	(0.0001)	(0.0001)	(0.0000)	(0.0000)	(0.0007)	(0.0001)	(0.0003)	(0.0002)

Table 9
The classification results for SPECTF heart data (positive rate = 20.60%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.3660	0.9942	0.3962	0.6801	0.1230	**0.9653**	0.1438	0.5442
	(0.0929)	(0.0001)	(0.0834)	(0.0224)	(0.0464)	(0.0033)	(0.0423)	(0.0085)
CS-SVM	1.0000	0.5579	0.6207	0.7789	0.8305	0.5121	0.4694	0.6713
	(0.0000)	(0.0710)	(0.0225)	(0.0178)	(0.0611)	(0.0632)	(0.0163)	(0.0093)
NCR-SVM	0.8990	0.7973	0.6718	0.8481	0.7606	0.7758	0.5716	0.7682
	(0.0077)	(0.0022)	(0.0051)	(0.0027)	(0.0317)	(0.0063)	(0.0216)	(0.0112)
RUS-SVM	0.9941	0.6766	0.6218	0.8354	**0.8641**	0.6480	0.5374	0.7560
	(0.0004)	(0.0141)	(0.0094)	(0.0035)	(0.0311)	(0.0240)	(0.0148)	(0.0059)
SMOTE-SVM	0.9983	0.9313	0.9325	0.9648	0.1780	0.9163	0.1209	0.5471
	(0.0003)	(0.1095)	(0.0719)	(0.0273)	(0.1983)	(0.0998)	(0.0731)	(0.0203)
BSMOTE-SVM	0.9900	0.8553	0.8575	0.9226	0.3776	0.8278	0.2796	0.6027
	(0.0248)	(0.1269)	(0.0701)	(0.0306)	(0.1196)	(0.1103)	(0.0452)	(0.0145)
Ada-SVM	0.8702	0.6108	0.5298	0.7405	0.7322	0.6162	0.4576	0.6742
	(0.0320)	(0.0304)	(0.0352)	(0.0176)	(0.0561)	(0.0247)	(0.0248)	(0.0143)
γAda-SVM	0.8702	0.6108	0.5298	0.7405	0.7322	0.6162	0.4576	0.6742
	(0.0320)	(0.0304)	(0.0352)	(0.0176)	(0.0561)	(0.0247)	(0.0248)	(0.0143)
Ada-CSSVM	0.8158	0.9319	0.7845	0.8739	0.4669	0.8422	0.4463	0.6545
	(0.0081)	(0.0011)	(0.0055)	(0.0021)	(0.0316)	(0.0030)	(0.0282)	(0.0103)
AdaC2-SVM	0.8062	0.9253	0.7705	0.8657	0.4748	0.8552	0.4630	0.6650
	(0.0045)	(0.0012)	(0.0050)	(0.0019)	(0.0334)	(0.0021)	(0.0304)	(0.0113)
Proposed	0.8519	0.8231	0.6744	0.8375	0.7850	0.8765	**0.6990**	**0.8307**
	(0.0118)	(0.0038)	(0.0011)	(0.0008)	(0.0156)	(0.0074)	(0.0116)	(0.0032)

was used to define borderline and noisy instances, and we assigned the factor scores to only borderline instances and positive noise. The computed factor scores were then used to adjust the instance weights in the AdaBoost framework by multiplying the factor scores and instance weights. Assigning the factor scores to BSVs and SVs was intended to address the overlap degradation issue, and we attempted to solve the small disjunct issue using the adjustment factor for positive noise. The ensemble AdaBoost framework was adopted to circumvent the data shift issue.

Using 10 real datasets, we conducted an intensive empirical study to compare the proposed method with two stand-alone methods, four data pre-processing methods, and four ensemble methods with resampling and cost-sensitive approaches. The experimental results showed that our weight adjustment approach in the SVM-AdaBoost framework generally outperforms the benchmarking methods in terms of F-measure and AUC. In addition, its low standard deviations for the performance measures in repeated cross validation demonstrated the stability of the proposed method. The reasons for such good performance come from the simultaneous consideration of three degradation issues in contrast to other existing methods that

Table 10
The classification results for Indian patient data (positive rate = 28.64%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.1683	0.9856	0.2148	0.5770	0.0580	**0.9686**	0.0855	0.5133
	(0.1307)	(0.0008)	(0.1308)	(0.0338)	(0.0106)	(0.0040)	(0.0135)	(0.0009)
CS-SVM	0.9269	0.5273	0.6127	0.7271	0.8019	0.5074	0.5175	0.6546
	(0.0014)	(0.0462)	(0.0295)	(0.0157)	(0.0967)	(0.0453)	(0.0227)	(0.0037)
NCR-SVM	0.9066	0.5795	0.6176	0.7430	0.7844	0.5591	0.5412	0.6718
	(0.0035)	(0.0087)	(0.0053)	(0.0044)	(0.0167)	(0.0140)	(0.0057)	(0.0053)
RUS-SVM	0.8715	0.5222	0.5726	0.6968	0.8169	0.5032	0.5311	0.6600
	(0.0033)	(0.0146)	(0.0072)	(0.0067)	(0.0138)	(0.0191)	(0.0049)	(0.0022)
SMOTE-SVM	0.9061	0.6075	0.6541	0.7568	0.7089	0.5742	0.4889	0.6415
	(0.0037)	(0.0445)	(0.0333)	(0.0169)	(0.0883)	(0.0381)	(0.0228)	(0.0040)
BSMOTE-SVM	0.9244	0.5230	0.6068	0.7237	0.8088	0.4816	0.5143	0.6452
	(0.0012)	(0.0375)	(0.0209)	(0.0124)	(0.0481)	(0.0225)	(0.0059)	(0.0024)
Ada-SVM	0.9908	0.3088	0.5336	0.6498	**0.9601**	0.2958	0.5145	0.6279
	(0.0001)	(0.0008)	(0.0002)	(0.0002)	(0.0013)	(0.0026)	(0.0028)	(0.0008)
γAda-SVM	0.9908	0.3088	0.5336	0.6498	**0.9601**	0.2958	0.5145	0.6279
	(0.0001)	(0.0008)	(0.0002)	(0.0002)	(0.0013)	(0.0026)	(0.0028)	(0.0008)
Ada-CSSVM	0.6814	0.6880	0.5524	0.6847	0.6172	0.6662	0.4979	0.6417
	(0.0109)	(0.0033)	(0.0024)	(0.0012)	(0.0175)	(0.0044)	(0.0079)	(0.0037)
AdaC2-SVM	0.7316	0.6983	0.5904	0.7150	0.6171	0.6547	0.4899	0.6359
	(0.0105)	(0.0129)	(0.0012)	(0.0011)	(0.0302)	(0.0146)	(0.0100)	(0.0049)
Proposed	0.8276	0.5832	0.5773	0.7054	0.8290	0.5831	**0.5752**	**0.7060**
	(0.0008)	(0.0006)	(0.0003)	(0.0002)	(0.0047)	(0.0037)	(0.0040)	(0.0027)

Table 11
The classification results for German credit data (positive rate = 30.00%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.1480	0.9894	0.2265	0.5687	0.0921	0.9747	0.1505	0.5334
	(0.0446)	(0.0001)	(0.0600)	(0.0112)	(0.0056)	(0.0012)	(0.0101)	(0.0007)
CS-SVM	0.6213	0.7450	0.5363	0.6832	0.5814	0.7264	0.5001	0.6539
	(0.0918)	(0.0703)	(0.0593)	(0.0122)	(0.0790)	(0.0705)	(0.0424)	(0.0060)
NCR-SVM	0.6559	0.7346	0.5397	0.6952	0.5563	0.6903	0.4614	0.6233
	(0.0725)	(0.0127)	(0.0224)	(0.0067)	(0.0473)	(0.0199)	(0.0130)	(0.0030)
RUS-SVM	0.4569	0.7973	0.4329	0.6271	0.4319	0.7809	0.4073	0.6064
	(0.0403)	(0.0081)	(0.0199)	(0.0057)	(0.0355)	(0.0134)	(0.0180)	(0.0042)
SMOTE-SVM	0.6951	0.9428	0.7155	0.8190	0.2706	0.8867	0.3374	0.5787
	(0.0907)	(0.0083)	(0.0655)	(0.0209)	(0.0344)	(0.0077)	(0.0144)	(0.0031)
BSMOTE-SVM	0.7025	0.7719	0.6145	0.7372	0.5884	0.6982	0.4949	0.6433
	(0.0953)	(0.1120)	(0.0618)	(0.0184)	(0.0704)	(0.0922)	(0.0372)	(0.0053)
Ada-SVM	0.0500	0.9922	0.0930	0.5211	0.0467	**0.9909**	0.0868	0.5188
	(0.0004)	(0.0000)	(0.0013)	(0.0001)	(0.0006)	(0.0001)	(0.0018)	(0.0002)
γAda-SVM	0.0500	0.9922	0.0930	0.5211	0.0467	**0.9909**	0.0868	0.5188
	(0.0004)	(0.0000)	(0.0013)	(0.0001)	(0.0006)	(0.0001)	(0.0018)	(0.0002)
Ada-CSSVM	0.6232	0.7976	0.5946	0.7104	0.5753	0.7757	0.5478	0.6755
	(0.0059)	(0.0043)	(0.0032)	(0.0019)	(0.0070)	(0.0047)	(0.0033)	(0.0012)
AdaC2-SVM	0.7609	0.8705	0.7377	0.8157	0.5738	0.7962	0.5590	0.6850
	(0.0072)	(0.0020)	(0.0084)	(0.0042)	(0.0055)	(0.0008)	(0.0044)	(0.0019)
Proposed	0.6000	0.7800	0.5677	0.6900	**0.6024**	0.7783	**0.5670**	**0.6904**
	(0.0003)	(0.0001)	(0.0002)	(0.0001)	(0.0032)	(0.0009)	(0.0026)	(0.0009)

only focus on some of these issues. The SVM enables us to pay more attention to instances in the overlapped region, i.e., near the decision boundary. Knowing that BSVs and SVs are near the decision boundary, we can assign greater weight to the misclassified instances, thereby enabling the next weak classifier to pay more attention to the correct classification. This property also enables us to construct an additional adjustment factor for the misclassified objects that are neither BSVs nor SVs. Another advantage of the SVM is that its kernel can build nonlinear decision boundaries so that we can detect arbitrary borderline instances and small disjuncts of a positive class. From the experimental results, we conclude that the proposed method is an effective contribution to the field of imbalanced data classification.

Table 12
The classification results for Vertebral data (positive rate = 33.26%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.7983	0.8932	0.7828	0.8458	0.7576	**0.8701**	0.7323	0.8139
	(0.0778)	(0.0041)	(0.0587)	(0.0143)	(0.0765)	(0.0077)	(0.0600)	(0.0127)
CS-SVM	0.9612	0.7655	0.7861	0.8633	0.9346	0.7519	0.7588	0.8432
	(0.0006)	(0.0042)	(0.0015)	(0.0008)	(0.0118)	(0.0093)	(0.0052)	(0.0021)
NCR-SVM	0.9707	0.7730	0.7950	0.8719	0.9138	0.7578	0.7514	0.8358
	(0.0015)	(0.0026)	(0.0020)	(0.0015)	(0.0123)	(0.0081)	(0.0047)	(0.0023)
RUS-SVM	0.9658	0.7541	0.7811	0.8599	**0.9424**	0.7336	0.7527	0.8380
	(0.0009)	(0.0100)	(0.0036)	(0.0019)	(0.0045)	(0.0145)	(0.0052)	(0.0025)
SMOTE-SVM	0.9564	0.8095	0.8250	0.8829	0.8027	0.7816	0.6962	0.7922
	(0.0011)	(0.0337)	(0.0114)	(0.0071)	(0.0961)	(0.0512)	(0.0496)	(0.0125)
BSMOTE-SVM	0.9676	0.7634	0.7920	0.8655	0.9091	0.7427	0.7411	0.8259
	(0.0006)	(0.0246)	(0.0078)	(0.0053)	(0.0422)	(0.0294)	(0.0110)	(0.0063)
Ada-SVM	0.8419	0.8453	0.7777	0.8436	0.8169	0.8342	0.7488	0.8255
	(0.0044)	(0.0018)	(0.0004)	(0.0003)	(0.0128)	(0.0061)	(0.0038)	(0.0025)
γAda-SVM	0.8419	0.8453	0.7777	0.8436	0.8169	0.8342	0.7488	0.8255
	(0.0044)	(0.0018)	(0.0004)	(0.0003)	(0.0128)	(0.0061)	(0.0038)	(0.0025)
Ada-CSSVM	0.8842	0.8393	0.7971	0.8618	0.8350	0.8378	0.7682	0.8364
	(0.0028)	(0.0037)	(0.0012)	(0.0007)	(0.0092)	(0.0097)	(0.0057)	(0.0032)
AdaC2-SVM	0.8113	0.8639	0.7741	0.8376	0.7985	0.8489	0.7535	0.8237
	(0.0040)	(0.0053)	(0.0009)	(0.0004)	(0.0083)	(0.0060)	(0.0054)	(0.0029)
Proposed	0.8301	0.8571	0.7793	0.8436	0.8565	0.8417	**0.7804**	**0.8491**
	(0.0004)	(0.0002)	(0.0002)	(0.0001)	(0.0058)	(0.0040)	(0.0034)	(0.0019)

Table 13
The classification results for Spambase data (positive rate = 39.40%).

	Training				Test			
	Sens	Spec	F1	AUC	Sens	Spec	F1	AUC
SVM	0.7578	0.9275	0.8005	0.8426	0.7088	0.8632	0.7298	0.7860
	(0.0340)	(0.0011)	(0.0217)	(0.0111)	(0.0256)	(0.0010)	(0.0120)	(0.0054)
CS-SVM	0.8562	0.8498	0.8210	0.8530	0.8170	0.7831	0.7585	0.8000
	(0.0166)	(0.0071)	(0.0125)	(0.0089)	(0.0121)	(0.0020)	(0.0058)	(0.0038)
NCR-SVM	0.9275	0.7645	0.8095	0.8460	**0.8966**	0.7201	0.7698	0.8083
	(0.0140)	(0.0010)	(0.0059)	(0.0046)	(0.0134)	(0.0010)	(0.0048)	(0.0034)
RUS-SVM	0.9033	0.8449	0.8437	0.8741	0.8620	0.7740	0.7802	0.8180
	(0.0046)	(0.0029)	(0.0045)	(0.0037)	(0.0042)	(0.0013)	(0.0034)	(0.0024)
SMOTE-SVM	0.9261	0.8985	0.8897	0.9123	0.8662	0.7832	0.7863	0.8247
	(0.0397)	(0.0177)	(0.0219)	(0.0142)	(0.0288)	(0.0056)	(0.0094)	(0.0047)
BSMOTE-SVM	0.9254	0.8764	0.8758	0.9009	0.8915	0.7822	0.8010	0.8369
	(0.0033)	(0.0073)	(0.0060)	(0.0051)	(0.0013)	(0.0016)	(0.0016)	(0.0014)
Ada-SVM	0.8121	0.8972	0.8270	0.8547	0.8152	0.8370	0.7991	0.8261
	(0.0049)	(0.0652)	(0.0059)	(0.0086)	(0.0078)	(0.1101)	(0.0108)	(0.0148)
γAda-SVM	0.8121	0.8972	0.8270	0.8547	0.8152	0.8370	0.7991	0.8261
	(0.0049)	(0.0652)	(0.0059)	(0.0086)	(0.0078)	(0.1101)	(0.0108)	(0.0148)
Ada-CSSVM	0.8657	0.8946	0.8538	0.8801	0.8520	0.8905	0.8434	0.8712
	(0.0002)	(0.0001)	(0.0000)	(0.0000)	(0.0002)	(0.0003)	(0.0002)	(0.0001)
AdaC2-SVM	0.8836	0.8782	0.8538	0.8809	0.8706	0.8674	0.8402	0.8690
	(0.0006)	(0.0048)	(0.0008)	(0.0006)	(0.0008)	(0.0042)	(0.0006)	(0.0006)
Proposed	0.8581	0.9066	0.8573	0.8823	0.8559	**0.9098**	**0.8582**	**0.8829**
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0002)	(0.0001)	(0.0002)	(0.0001)

Acknowledgments

This study was supported by the Basic Science Research Program through the National Research Foundation of Korea from the Ministry of Education, Science and Technology (NRF-2013R1A2A2A03068323). The work by Jong-Seok Lee was supported by the MSIP, Korea, under the G-ITRC support program (IITP-2016-R6812-16-0001) supervised by the IITP.

References

[1] L. Bottou, C.-J. Lin, Support vector machine solvers, in: L. Bottou, O. Chappelle, D. DeCoste, J. Weston (Eds.), Large Scale Kernel Machines, MIT Press, Cambridge, MA, 2007.
[2] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognit. 30 (1997) 1145–1159.
[3] W.-L. Chao, J.-Z. Liu, J.-J. Ding, Facial age estimation based on label-sensitive learning and age-oriented regression, Pattern Recognit. 46 (2013) 628–641.
[4] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.

[5] F. Cheng, J. Zhang, C. Wen, Cost-sensitive large margin distribution machine for classification of imbalanced data, Pattern Recognit. Lett. 80 (2016) 107–112.
[6] M. Denil, T. Trappenberg, Overlap versus imbalance, in: A. Farzindar, V. Keselj (Eds.), Advances in Artificial Intelligence, Springer, Berlin, 2010, pp. 220–231.
[7] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the 17th International Joint Conference on Artificial Intelligence, 2001, pp. 973–978.
[8] W. Fan, S.J. Stolfo, J.X. Zhang, P.K. Chan, AdaCost: misclassification cost-sensitive boosting, in: Proceedings of the 16th International Conference on Machine Learning, 1999, pp. 97–105.
[9] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, Ordering-based pruning for improving the performance of ensembles of classifiers in the framework of imbalanced datasets, Inf. Sci. 354 (2016) 178–196.
[10] V. García, R.A. Mollineda, J.S. Sánchez, R. Alejo, J.M. Sotoca, et al., When overlapping unexpectedly alters the class imbalance effects, in: J. Marti, et al. (Eds.), Pattern Recognition and Image Analysis, Springer, Berlin, 2007, pp. 499–506.
[11] H. Han, W.-Y. Wang, B.-H. Mao, et al., Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: D.S. Huang, et al. (Eds.), Advances in Intelligent Computing, Springer, Berlin, 2005, pp. 878–887.
[12] M.N. Haque, N. Noman, R. Berretta, P. Moscato, Heterogeneous ensemble combination search using genetic algorithm for class imbalanced data classification, PLoS One 11 (2016) 1–28.
[13] C.-W. Hsu, C.-C. Chang, C.-J. Lin, A Practical Guide to Support Vector Classification, Technical Report, Department of Computer Science, National Taiwan University, Taiwan, 2003 http://www.csie.ntu.edu.tw/~cjlin.
[14] W. Khreich, E. Granger, A. Miri, R. Sabourin, Adaptive ROC-based ensembles of HMMs applied to anomaly detection, Pattern Recognit. 45 (2012) 208–230.
[15] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, S.Y. Bang, Constructing support vector machine ensemble, Pattern Recognit. 36 (2003) 2757–2767.
[16] J. Laurikkala, Improving identification of difficult small classes by balancing class distribution, in: Proceedings of the 8th Conference on AI in Medicine in Europe, 2001, pp. 63–66.
[17] X. Li, L. Wang, E. Sung, AdaBoost with SVM-based component classifiers, Eng. Appl. Artif. Intell. 21 (2008) 785–795.
[18] M. Lichman, UCI Machine Learning Repository, 2013, http://archive.ics.uci.edu/ml.
[19] N.H.C. Lima, A.D.D Neto, J.D. DeMelo, Creating an ensemble of diverse support vector machines using adaboost, in: Proceedings of International Joint Conference on Neural Networks, 2009, pp. 1802–1806.
[20] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inf. Sci. 250 (2013) 113–141.
[21] V. López, A. Fernández, F. Herrera, On the importance of the validation technique for classification with imbalanced datasets: addressing covariate shift when data is skewed, Inf. Sci. 257 (2014) 1–13.
[22] V. López, A. Fernández, J.G. Moreno-Torres, F. Herrera, Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification – open problems on intrinsic data characteristics, Expert Syst. Appl. 39 (2012) 6585–6608.
[23] K. Morik, P. Brockhausen, T. Joachims, Combining statistical learning with a knowledge-based approach: a case study in intensive care monitoring, Universität Dortmund, 1999 Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen.
[24] I. Pardoe, Churn data, 2006, http://www.iainpardoe.com.
[25] L. Pelayo, S. Dick, Applying novel resampling strategies to software defect prediction, in: Proceedings of Annual Meeting of the North American Fuzzy Information Processing Society, 2007, pp. 69–72.
[26] R.C. Prati, G.E. Batista, M.C. Monard, et al., Class imbalances versus class overlapping: an analysis of a learning system behavior, in: R. Monroy, et al. (Eds.), Advances in Artificial Intelligence, Springer, Berlin, 2004, pp. 312–321.
[27] W.A. Rivera, P. Xanthopoulos, A priori synthetic over-sampling methods for increasing classification sensitivity in imbalanced data sets, Expert Syst. Appl. 66 (2016) 124–135.
[28] J.A. Sáez, J. Luengo, J. Stefanowski, F. Herrera, SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, Inf. Sci. 291 (2015) 184–203.
[29] M. Sikora, L. Wróbel, Application of rule induction algorithms for analysis of data collected by seismic hazard monitoring systems in coal mines, Arch. Min. Sci. 55 (2010) 91–114.
[30] J. Stork, R. Ramos, Case Study Report: Building and analyzing SVM ensembles with Bagging and AdaBoost on big data sets, Cologne University of Applied Sciences, 2013 CIOP Technical Report.
[31] Y. Sun, M.S. Kamel, A.K. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recognit. 40 (2007) 3358–3378.
[32] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, Y. Zhou, A novel ensemble method for classifying imbalanced data, Pattern Recognit. 48 (2015) 1623–1637.
[33] K.M. Ting, A comparative study of cost-sensitive boosting algorithms, in: Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 983–990.
[34] K. Veropoulos, C. Campbell, N. Cristianini, Controlling the sensitivity of support vector machines, in: Proceedings of the International Joint Conference on Artificial Intelligence, 1999, pp. 55–60.
[35] S. Vluymans, D.S. Tarragó, Y. Saeys, C. Cornelis, F. Herrera, Fuzzy rough classifiers for class imbalanced multi-instance data, Pattern Recognit. 53 (2016) 36–45.
[36] B.X. Wang, N. Japkowicz, Boosting support vector machines for imbalanced data sets, Knowl. Inf. Syst. 25 (2010) 1–20.
[37] X. Wang, C. Wu, C. Zheng, W. Wang, Improved algorithm for Adaboost with SVM base classifiers, in: Proceedings of 5th IEEE International Conference on Cognitive Informatics, 2006, pp. 948–952.
[38] Q. Yang, X. Wu, 10 challenging problems in data mining research, Int. J. Inf. Technol. Decis. Mak. 5 (2006) 597–604.
[39] D. Zhang, M.M. Islam, G. Lu, A review on automatic image annotation techniques, Pattern Recognit. 45 (2012) 346–362.