11-2021

Expediting the accuracy-improving process of SVMs for class imbalance learning

Bin CAO
Zhejiang University of Technology

Yuqi LIU
Zhejiang University of Technology

Chenyu HOU
Zhejiang University of Technology

Jing FAN
Zhejiang University of Technology

Baihua ZHENG
Singapore Management University, bhzheng@smu.edu.sg

See next page for additional authors

Expediting the Accuracy-improving Process of SVMs for Class Imbalance Learning

Bin Cao, *Member, IEEE,* Yuqi Liu, Chenyu Hou, Jing Fan, Baihua Zheng, and Jianwei Yin

Abstract—To improve the classification performance of support vector machines (SVMs) on imbalanced datasets, cost-sensitive learning methods have been proposed, e.g., DEC (Different Error Costs) and FSVM-CIL (Fuzzy SVM for Class Imbalance Learning). They relocate the hyperplane by adjusting the costs associated with misclassifying samples. However, the error costs are determined either empirically or by performing an exhaustive search in the parameter space. Both strategies can not guarantee effectiveness and efficiency simultaneously. In this paper, we propose *ATEC*, a solution that can efficiently find a preferable hyperplane by automatically tuning the error cost for between-class samples. *ATEC* distinguishes itself from all existing parameter tuning strategies by two main features: (1) it can evaluate how effective an error cost is in terms of classification accuracy; and (2) it changes the error cost in the right direction if it is not effective. Extensive experiments show that compared with the state-of-art methods, SVMs that are equipped with *ATEC* can not only obtain comparable improvements in terms of F1 score of minority class, area under the precision-recall curve (AUC-PR) and area under the ROC curve (AUC-ROC) scores, but also outperform the grid-search parameter tuning strategy by two orders of magnitude in terms of the training time when a high F1 score is required.

Index Terms—Class imbalance learning, SVMs, Classification.

1 Introduction

Classification is a fundamental task in many machine learning based applications. Lots of classification approaches have been proposed over the last decades, among which, the *Support Vector Machines (SVMs)* have achieved great success in many classification tasks [1], [2]. However, standard SVMs assume the training samples are close to uniform distribution across different classes, and hence their performance could be significantly compromised by the imbalanced data distribution that exists in many applications, e.g., Twitter spam detection [3], video surveillance [2] and customer churn prediction [4]. This is because the SVM classifier model learned from an imbalanced dataset can be severely biased toward the majority class and easily misclassify the minority samples.

Therefore, it is very important to further improve the existing classification algorithms to make sure they are able to adapt to the imbalanced datasets and be able to achieve high accuracy even when the underlying datasets are imbalanced. In this paper, we focus on the imbalanced learning problem of binary classification [5], and treat the minority class and the majority class as the positive class and the negative class respectively.

To build a desirable SVM classifier that can perform well for both the majority and the minority classes, many techniques have been developed. In general, they can be categorized into two clusters, *sampling-based methods* and *cost-sensitive SVMs (CS-SVM)* [1], [6], [7]. The former attempts to create balanced data distributions by using different sampling strategies, e.g., oversampling and undersampling, while the latter adopts different error costs for different classes. Moreover, CS-SVMs have been empirically observed to outperform the sampling-based methods in many application domains [8]. For example, *Different Error Costs (DEC)* [6] and *Fuzzy SVM for Class Imbalance Learning (FSVM-CIL)* [1] are the two CS-SVM representatives that improve the SVMs effectively. They reduce the class imbalance by assigning higher error costs to the minority class samples. By incorporating these costs in the SVM learning process, it is possible to derive a hyperplane that can help mitigate the skewness toward the minority class, and hence the final classification accuracy for the minority class can be possibly increased. Note that, different from DEC where all the within-class samples share the same error costs, FSVM-CIL can reflect the importance of each training sample by assigning different weights to different samples, and hence FSVM-CIL is more robust to the outliers and noise.

The classification accuracy of the CS-SVM is significantly affected by the aforementioned error costs. Consequently, it is extremely important to assign proper values to error costs. In real implementations, we can determine the error costs either based on empirical values or by an exhaustive search in the parameter space. Here a dilemma arises. An assignment based on empirical values is simple but it cannot guarantee effective classification results. On the other hand, an assignment via exhaustive searches, e.g., grid search, can guarantee that the generated SVM classifier is able to achieve the best performance on validation set but it is extremely expensive. Therefore, it is very desirable to have an approach that is both efficient and effective. In this paper,

- *B.Cao, Y.Liu, C.Hou and J.Fan are with the College of Computer Science, Zhejiang University of Technology, Hangzhou, China*
  E-mail: bincao@zjut.edu.cn, yqliu@zjut.edu.cn, houcy@zjut.edu.cn, fanjing@zjut.edu.cn
- *B.Zheng is with the School of Information Systems, Singapore Management University.*
  E-mail:bhzheng@smu.edu.sg
- *J.Yin is with the College of Computer Science, Zhejiang University of Technology, Hangzhou, China*
  E-mail: zjuyjw@zju.edu.cn
- *The Corresponding Author is Jing Fan (fanjing@zjut.edu.cn).*

Manuscript received May 10, 2019

the efficiency refers to the time required to find the proper values of error costs, and the effectiveness is measured by the accuracy of the SVM classifier.

However, to the best of our knowledge, none of the existing approaches can achieve both efficiency and effectiveness simultaneously. Bayesian optimization technique [9] [10] is widely used to automatically tune hyperparameters of various machine learning algorithms, and the loss function on the data set is usually used as the optimization goal. However, in imbalanced learning, simply optimizing the loss function on the highly skewed data set may result in a biased result. In recent studies, LexiBoost [11] is the latest method that attempts to handle class imbalance without cost tuning. However, LexiBoost is actually an ensemble learning technique while we focus on addressing the efficiency and effectiveness of cost tuning for CS-SVM. Gu et al. [12] proposed an efficient approach to find the optimal hyperparameters (i.e. error costs for positive and negative samples) and obtain the global minimum cross validation error for CS-SVM. However, the algorithm cannot be extended to other assessment metrics currently, e.g., F1 value or receiver operating characteristic (ROC) curve.

Therefore, motivated by the above findings and the universal existence of imbalanced class distribution, we dedicate this paper to a novel solution, namely *Auto-Tuning of the Error Costs (ATEC)*. Different from all existing methods, ATEC is able to i) evaluate how effective an error cost is in terms of classification accuracy; and ii) change the value in the right direction if it is not effective. With the help of these two unique and desirable features, ATEC is expected to finish the auto-tuning process within a very short duration and meanwhile, the generated classifier is expected to be able to achieve a good classification accuracy. Note ATEC is built on top of SVMs and it is not to change the inner implementation of standard SVM or its improved versions (e.g., DEC and FSVM-CIL). Instead, ATEC can be viewed as an add-in component that can help existing SVMs to quickly find a proper value of the error cost such that the generated *preferable* hyperplane is able to achieve a good classification accuracy on class-imbalance datasets. In brief, the main contribution of this paper is three-fold.

1) We propose a guidance on the selection of error costs. To be more specific, the proposed guidance is able to evaluate how effective a given hyperplane $\mathcal{H}$ is in terms of classifying class-imbalance data and to direct the adjustment of the error cost and hence the hyperplane $\mathcal{H}$ if $\mathcal{H}$'s classification accuracy is not high.
2) Based on the proposed guidance, we propose *ATEC*, a method to efficiently and effectively locate a proper value of the error cost and hence help to improve existing SVM classifiers (e.g., DEC and FSVM-CIL) on the class-imbalance data.
3) We perform extensive experiments with tens of real imbalanced datasets from multiple domains. The result shows that the algorithms that are equipped with *ATEC* not only outperform their original forms for most, if not all of, the datasets in terms of F1 score of minority class, area under the precision-recall curve (AUC-PR) and area under the ROC curve (AUC-ROC) scores, but also require much less running time in order to achieve the same F1 scores.

TABLE 1
Frequently Used Notations

Notation	Description
N	number of samples
x, y	vector and label $\{-1, 1\}$ of a sample
p	total dimension of x
w, b	normal vector and bias of a hyperplane
α	the Lagrange multipliers
Φ	a mapping function that transforms the data to another feature space
C^+, C^-	error cost of positive class and negative class
f	the probability density function of samples distribution
Z	a new axis from dimensionality reduction
$\mathcal{H}$	a classification hyperplane
E^+, E^-	the entropy of positive interval and negative interval
d^+, d^-	the length of positive interval and negative interval
k	the error cost ratio $\frac{C^+}{C^-}$

The remainder of this paper is organized as follows. Section 2 presents preliminaries. Section 3 introduces the concept of *preferable hyperplanes* and presents a guidance to measure whether a given hyperplane is preferable. Section 4 presents the design and the implementation details of *ATEC*. Section 5 reports the results of experimental evaluation and Section 6 reviews the related work. Finally, Section 7 concludes this paper.

2 PRELIMINARIES

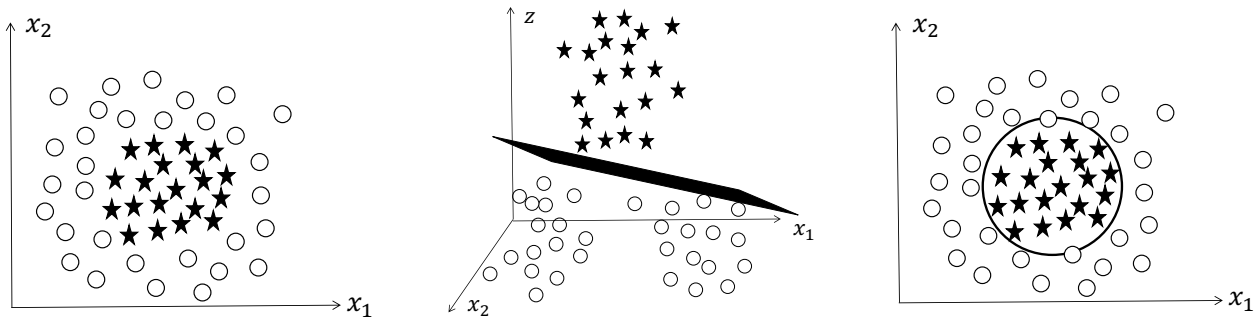
In this section, we first briefly review the SVM algorithm, together with two improved versions of SVM for imbalanced datasets, i.e., DEC and FSVM-CIL; we then present the concept of entropy, which will be used to evaluate the classification effectiveness of a given hyperplane. Table 1 summarizes the frequently used notations in the paper.

2.1 SVM

Let's consider a binary classification problem. Given a n-dimensional hyperspace, there are N samples represented by (x_i, y_i), $i = 1, 2, \cdots, N$, where $x_i \in \mathbf{R^p}$ is a p-dimensional feature vector, and $y_i \in \{-1, 1\}$ is the class of each sample. The goal of the SVM algorithm is to find a separating hyperplane that can split the samples properly, i.e., dividing them into two classes as correctly as possible. Moreover, considering that some real-word datasets are nonlinearly separable, a mapping function Φ is usually introduced in SVMs to transform the data into a higher dimensional feature space. Thereafter, a possible separating hyperplane that resides in the higher dimensional space can be produced and it can be denoted by $w \cdot \Phi(x) + b = 0$, where w is the normal vector to the hyperplane.

For example, in Figure 1(a), circles and stars represent positive and negative samples respectively. As observed, there is no good hyperplane that can separate them properly in the original feature space. However, after these samples are mapped to a higher dimensional space, a separating hyperplane as shown in Figure 1(b) could be found. Finally, this hyperplane can be transformed back to the original feature space and a classifying boundary denoted by a black circle in Figure 1(c) could be generated.

Hence, if these samples are completely separable, the SVM algorithm tries to find a hyperplane so that samples of the two classes can be distributed on different sides of



(a) Samples are non-linearly separable (b) Samples are classified in a higher dimensional space (c) The final hyperplane

Fig. 1. An example to illustrate the effectiveness of the mapping function Φ of SVMs

the hyperplane and the margin of the hyperplane (i.e., the distance from the hyperplane to the nearest data point on each side) is maximal. Such hyperplane can be found by solving the following optimization problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 \\ s.t. \quad & y_i(w \cdot \Phi(x_i) + b) \geq 1 \wedge i \in [1, N] \end{aligned} \tag{1}$$

However, many real-world datasets are not completely linearly separable even after they are mapped to a higher dimensional feature space. To address this, SVM introduces a slack variable ξ_i for each sample so that some of them can be misclassified. In this way, we can find the hyperplane by solving the following soft-margin optimization problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y_i(w \cdot \Phi(x_i) + b) \geq 1 - \xi_i \wedge \xi_i \geq 0 \wedge i \in [1, N] \end{aligned} \tag{2}$$

where C can be viewed as the error cost for a training sample.

Usually, Equation (2) can be solved by introducing Lagrange multipliers α_i and constructing a Lagrangian representation. Finally, the optimal values for α_i can be computed, and w can be recovered as follows:

$$w = \sum_{i=1} \alpha_i y_i \Phi(x_i) \tag{3}$$

Then based on Equation (3), the SVM decision function can be given by

$$\begin{aligned} f(x) &= sign(w \cdot \Phi(x) + b) \\ &= sign(\sum_{i=1} \alpha_i y_i K(x_i, x) + b) \end{aligned} \tag{4}$$

where $K(x_i, x) = \Phi(x_i) \cdot \Phi(x)$ is a kernel function, so that computing the mapping function $\Phi(x)$ can be replaced by a kernel function. There are various kinds of kernel functions corresponding to different mapping Φ, such as Gaussian kernel function, polynomial kernel function, etc.

As mentioned before, standard SVM performs well in classifying balanced datasets, but it can not produce satisfactory results when dealing with imbalanced datasets. Therefore, some methods are proposed to improve the performance of SVM, among which DEC [6] and FSVM-CIL [1] are two representatives. Without loss of generality, we assume the positive class is the minority class and the negative class is the majority class.

DEC. The main idea of DEC is that the error costs for positive and negative samples should be different. In order to amend the standard SVM to classify positive samples correctly, the error cost for the positive class should be bigger than that of the negative class. In other words, DEC assigns C^+ and C^- as the error costs to the positive samples and the negative samples respectively, with $C^+ > C^-$. Accordingly, the optimization problem in Equation (2) could be modified as follows:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C^+\sum_{\{i|y_i=1\}}\xi_i + C^-\sum_{\{i|y_i=-1\}}\xi_i \\ s.t. \quad & y_i(w \cdot \Phi(x_i) + b) \geq 1 - \xi_i \wedge \xi_i \geq 0 \wedge i \in [1, N] \end{aligned} \tag{5}$$

Through solving Equation (5), DEC can move the hyperplane towards the negative class by setting $\frac{C^+}{C^-} > 1$. Empirically, C^- is usually fixed to 1 while C^+ is tuned within the range $(1, +\infty)$.

FSVM-CIL. Similar to DEC, FSVM-CIL also assigns different error costs C^+ and C^- to positive and negative classes, with $C^+ > C^-$. In addition, FSVM-CIL uses a membership function to assign a value between 0 and 1 to each within-class sample, which can further reflect the importance of a sample in its own class. In this way, FSVM-CIL becomes less sensitive to noise and outliers. Accordingly, the optimization problem in Equation (5) is modified as follows:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C^+\sum_{\{i|y_i=1\}} f(x_i)\xi_i \\ & + C^-\sum_{\{i|y_i=-1\}} f(x_i)\xi_i \\ s.t. \quad & y_i(w \cdot \Phi(x_i) + b) \geq 1 - \xi_i \wedge \xi_i \geq 0 \wedge i \in [1, N] \end{aligned} \tag{6}$$

Function $f(x_i) \in [0, 1]$ in Equation (6) is the membership function used to measure the importance of each sample within a class. $f(x_i)$ can be defined in multiple ways, e.g., based on the distance from the own class center. However, there is no best definition since it is dataset-dependent.

2.2 Entropy

In information theory, an entropy [13] is defined as the average amount of information produced by a stochastic source of data. In other words, it can be used to measure the uncertainty of data. For example, given a variable V and a set of possible values $\{v_1, v_2, \cdots, v_n\}$, we assume the

probability of V being v_i is P_i. The entropy of the variable V can be calculated by Equation (7).

$$E(V) = \sum_{i=1}^{n} -P_i \log_b P_i \tag{7}$$

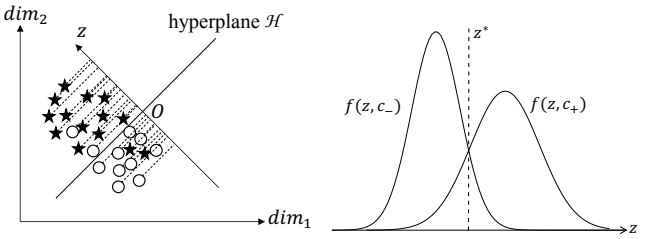
Here, b is the base of the logarithmic function which is usually set to 2, and $E(V)$ refers to the average information needed if we want to know the exact value of V. Based on the maximum entropy theory [14], we know that the value of $E(V)$ will be the maximum when the probability of each value is identical, i.e., $\forall i \in [1, n]$, $P_i = \frac{1}{n}$.

3 Error Costs Selection

SVM improvements such as DEC and FSVM-CIL contrive to reduce the class-imbalance problem by assigning different error costs C^+ and C^- to positive and negative classes with $C^+ > C^-$. However, as mentioned in Section 1, it is hard to determine a proper value for $\frac{C^+}{C^-}$ so that the corresponding hyperplane is able to differentiate positive samples from negative samples. The main reason behind this is that existing approaches have no idea of whether the hyperplane is in a good position or not. Although Akbani et.al [15] empirically affirmed that the setting of $\frac{C^+}{C^-} = I.R.$ (the ratio of the number of negative samples to that of positive samples) can produce good results, it heavily depends on the nature of the datasets and cannot guarantee the generated hyperplane can always achieve a good classification performance for all the datasets. Hence, knowing where a good hyperplane with high classification accuracy should be located in the search space of all possible $\frac{C^+}{C^-}$ values is critical for further improving existing SVMs. Instead of employing a general rule such as $\frac{C^+}{C^-} = I.R.$, we propose a more specific guidance that considers the nature of the underlying dataset on choosing a proper value for $\frac{C^+}{C^-}$. In the following, we first present a space reduction technique, then convert the problem of choosing a proper value for $\frac{C^+}{C^-}$ to a guidance on a preferable hyperplane in the reduced space.

3.1 Multi-Dimensional Space Reduction

Given the training samples which usually consist of multiple features, automatically finding a good hyperplane in such a multi-dimensional space is challenging. In our work, we strategically simplify this problem by projecting these multi-variable samples to a one-dimensional space, and we then search for a good hyperplane in this reduced space instead of the original high dimensional space. To implement this idea, we have to decide the dimension used for space reduction. Though there are many different dimension reduction methods, we choose the dimension based on the hyperplane learned from original samples. Specifically, we can invoke an existing SVM classifier (e.g., DEC) to output an initial hyperplane $\mathcal{H}$ based on given error costs and select the dimension that is perpendicular to $\mathcal{H}$ for reduction. The intuition behind is that the samples can be classified by the SVM hyperplane $\mathcal{H}$, i.e., the positive and negative classes are expected to be on different sides of the hyperplane. Consequently, if we simply project all the samples to the dimension that is orthogonal to $\mathcal{H}$, the separation of positive and negative samples remains the same.



(a) Space reduction example (b) Data distribution along Z-axis

Fig. 2. An illustration example for the proposed guidance

We plot an example in Fig. 2(a) for illustration. Circles and stars in Fig. 2(a) refer to two types of samples in a two-dimensional space; $\mathcal{H}$ is a hyperplane derived by an existing SVM technique. Note that, the hyperplane $\mathcal{H}$ might not be optimal and it could mis-classify certain samples. The space reduction technique proposed in this paper constructs a new axis (namely Z-axis) that is orthogonal to $\mathcal{H}$. Obviously, the direction of Z-axis is consistent with that of the normal of $\mathcal{H}$. If we project a sample x to the Z-axis, Equation (8) defines the position of x on Z-axis. Let point O denote the intersection between Z-axis and $\mathcal{H}$. The classifier based on $\mathcal{H}$ in the original 2-D space is equivalent to a classifier based on point O along Z-axis.

$$z = \frac{w \cdot \Phi(x) + b}{||w||} \tag{8}$$

Note that the numerator $(w \cdot \Phi(x) + b)$ in Equation (8) is the separating hyperplane. Similar to the computation of SVM, z can be computed with Lagrange multipliers α_i. Specifically, based on Equation (3) and Equation (4), Equation (8) can be rewritten as follows:

$$\begin{aligned} z &= \frac{\sum\limits_{i=1} \alpha_i y_i K(x_i, x) + b}{\sqrt{\sum\limits_{i=1} \alpha_i y_i \Phi(x_i) \cdot \sum\limits_{j=1} \alpha_j y_j \Phi(x_j)}} \\ &= \frac{\sum\limits_{i=1} \alpha_i y_i K(x_i, x) + b}{\sqrt{\sum\limits_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)}} \end{aligned} \tag{9}$$

3.2 The Guideline for Preferable Hyperplanes

Based on the multi-dimensional space reduction, we are ready to identify a good hyperplane through analyzing the data distribution along the Z-axis. As shown in Fig. 2(b), assume the projected samples on Z-axis are distributed as a Gaussian, and $f(z, c\_)$ and $f(z, c_+)$ are the probability density functions of negative ($c\_$) and positive (c_+) samples respectively. Based on the minimum misclassification rate decision rule [16], we know that *the best boundary on the Z-axis should lie in the position where the probability densities of positive and negative classes are the same*, i.e., $f(z, c\_) = f(z, c_+)$. For example, the dotted line ($z = z^*$) is the best separation between the classes in Fig. 2(b). It can be explained by the fact that if a boundary is at the position where $f(z, c\_) > f(z, c_+)$, we will misclassify the positive samples into the negative class. Therefore, only when the boundary is at the position that $f(z, c\_) = f(z, c_+)$ can

we assure that the probability of misclassifying reaches the minimum.

Although the above analysis is correct in theory, it is not practical in reality due to the difficulty of finding the probability density function for samples in real scenarios [17]. Actually, acquiring the exact probability density function for samples is one of the most challenging tasks. In this paper, we adopt entropy as an alternate. To be more specific, given a set of samples and a hyperplane $\mathcal{H}$, we calculate the entropy within a certain area on both sides of $\mathcal{H}$ based on Equation (7). The assumption behind this is that if both the entropy on one side of $\mathcal{H}$ and that on the other side of $\mathcal{H}$ reach their maximum, the hyperplane $\mathcal{H}$ is possibly a *preferable* hyperplane with good classification performance since the uncertainty of the samples on both sides reach the maximum.

Motivated by this, we propose a guideline that can heuristically judge whether the hyperplane is *preferable*, i.e., the hyperplane is able to generate good and desirable classification performance. Suppose the origin of the Z-axis for a given hyperplane is O, based on which we define two intervals along the Z-axis. They are the *positive interval* and the *negative interval*, marked as $[0, d^+]$ and $[-d^-, 0]$ respectively. In addition, we use E^+ and E^- to represent the entropy of samples mapped to the positive interval and that of samples mapped to the negative interval respectively. For example, take the positive interval I_1 in Fig. 3 as an example. There are two data points in I_1, with one being positive and the other being negative. Consequently, the corresponding entropy E^+ is $-(\frac{1}{2}log_2\frac{1}{2} + \frac{1}{2}log_2\frac{1}{2}) = 1$. Then, the guideline of a preferable hyperplane could be formalized as follows:

$$E^+ = E^- = MAX \quad \wedge \quad d^+ = d^- \tag{10}$$

Specifically, Equation (10) tries to achieve two goals, namely *maximum entropy* and *equal interval length*. The first goal of maximum entropy aims at locating the margin between two classes. $E^+ = E^- = MAX$ indicates that both the entropy on the positive side and that on the negative side reach their maximum values, i.e., numbers of the positive and negative samples are approximately equivalent in both positive and negative intervals. Note that, this goal only works for soft-margin classification tasks. We skip the discussion of hard-margin problems in this paper since they are overly sensitive to noise in data which makes them less applicable in real applications.

The second goal of equal interval length is to further restrict the area where a preferable hyperplane might be located at and to make it more resilient to noise. We use a counterexample plotted in Fig.3 to explain the intention behind imposing the constraint of $d^+ = d^-$. Assume the white circles and the black stars denote the negative samples and the positive samples respectively, and they are currently separated by a hyperplane H. Let I_1 and I_2 represent intervals corresponding to the positive and negative directions of the hyperplane H, then d^+ and d^- represent the lengths of I_1 and I_2 respectively. Note that I_1 and I_2 are very different, i.e., $d^+ < d^-$. In both intervals, the number of positive samples is the same as that of negative samples, hence the probability distributions of positive and negative samples are also same. Consequently, both E^+ and E^- reach 1, the maximum value an entropy can reach. However,

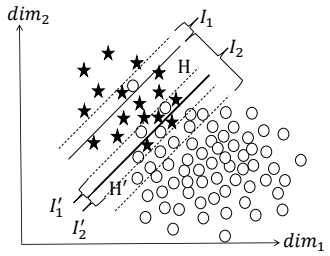


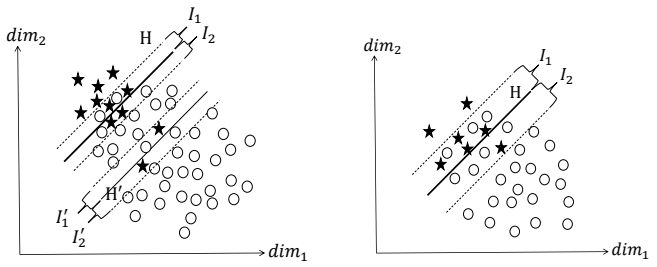
Fig. 3. An example for illustrating $d^+ = d^-$

hyperplane H is not preferable as it misclassifies many samples. An alternate hyperplane H' denoted by the thick line in Fig.3 performs much better since the misclassification of H' reaches the minimum within the range constructed by the intervals I_1' and I_2'. Hence, the requirement of equal interval length is to avoid the false-signal derived from similar cases.

Discussion. Our guideline presented in Equation (10) is able to move the hyperplane to the right location to achieve a desirable classification performance in most cases (as to be demonstrated by our simulation study presented later). However, we also understand that this guideline is based on heuristics and it cannot guarantee the output hyperplane is always able to achieve the best classification accuracy. In the following, we highlight two extreme cases where our guideline might not work. First, if there are multiple locations along the Z-axis, with their own d^+ and d^-, that define multiple subareas where both entropy E^+ and entropy E^- reach the maximum (i.e., multiple subareas containing roughly equal number of positive samples and negative samples), this guideline might not work. We use the example shown in Fig. 4(a) to illustrate this case. We plot two hyperplanes H and H', together with their corresponding positive intervals I_1, I_1' and negative intervals I_2 and I_2'. Areas defined by $I_1 \cup I_2$ and $I_1' \cup I_2'$ refer to two subareas where the number of positive samples is equivalent to the number of negative samples. Apparently, both H and H' satisfy Equation (10). However, H, but not H', is the global optimal. Under this case, there is a possibility that our guideline constrained by Equation (10) might return hyperplane H' as the desirable hyperplane although hyperplane H could achieve a much higher classification accuracy. Second, if none of the locations (including the optimal location) along Z-axis satisfies the conditions listed in Equation (10), our guideline might miss the optimal hyperplane. Refer to the example shown in Figure 4(b). The hyperplane H is able to achieve the best classification accuracy, while H does not satisfy Equation (10). It's possible that our guideline misjudges hyperplane H as non-preferable as conditions listed in Equation (10) are not valid under H. Nevertheless, this guideline is expected to provide correct guides in most of the cases. The experimental results to be reported in Section 5 will demonstrate its power.

4 ATEC IMPLEMENTATION

In this section, we propose a solution named *ATEC* to improve existing SVMs by Automatically Tuning the Error Cost. Specifically, ATEC consists of the following two main modules, i.e., *Hyperplane Generation and Evaluation (GE) Module* and *Hyperplane Adjustment Module*. The former adopts



(a) Case 1: multiple sub-areas satisfy Equation 10

(b) Case 2: none location in Z-axis satisfies Equation 10

Fig. 4. Illustration of extreme cases that our guideline does not work

an existing SVM technique to generate a hyperplane $\mathcal{H}$ based on a given error cost ratio $\frac{C^+}{C^-}$ (denoted as k) and makes full use of the aforementioned guidance to evaluate the performance of $\mathcal{H}$ and to find out the right direction along the reduced one-dimensional space that $\mathcal{H}$ should be moved to if $\mathcal{H}$ is not preferable. Guided by the output from the GE module, the latter performs the adjustment of the hyperplane. To be more specific, it changes the value of k and invokes the GE module again to generate a new hyperplane. Based on the evaluation result of the new hyperplane, it might perform a new adjustment to further tune the value of k. In other words, ATEC gradually adjusts the value of k and hence the position of the hyperplane via multiple iterations.

4.1 Hyperplane Generation and Evaluation

This module takes a set of training samples, a support vector machine, and an error cost ratio as input, and outputs a hyperplane generated based on the given error cost ratio and a variable $sign$ that indicates the adjustment required. The main idea is to invoke an existing SVM technique to generate a hyperplane $\mathcal{H}$ and to adopt guidance proposed in Section 3 to evaluate $\mathcal{H}$ based on the reduced one-dimensional space of Z-axis. The ultimate goal is to find out (1) whether $\mathcal{H}$ is already a preferable hyperplane, and (2) if not, which direction shall $\mathcal{H}$ be moved to for better classification power. Before we present the detailed algorithm, we first explain all the evaluation outcomes. According to Equation (10), there are in total five cases, detailed in the following.

Case 1: $E^+ = E^- \wedge d^+ = d^-$. The guideline defined in Equation (10) is fully satisfied, and the current hyperplane is preferable. Consequently, $sign$ is set to 0, indicating no further adjustment is required.

Case 2: $E^+ = E^- \wedge d^+ > d^-$. The maximum entropy goal is met, but not the goal of equal interval length. We set $sign$ to 1, indicating that the hyperplane shall be moved toward the positive direction of Z-axis.

Case 3: $E^+ = E^- \wedge d^+ < d^-$. Like Case 2, we set $sign$ to -1, indicating that the hyperplane shall be moved toward the negative direction of Z-axis.

Case 4: $E^+ < E^-$. According to Equation (7) and the fact that there are two possible values $\{1, -1\}$ of y, the entropy reaches its maximum value if $P(y = 1) = P(y = -1) = 0.5$. Given the fact that $E^+ < E^-$ and $E^- \leq 1$, we can conclude that the difference between $P(y = 1)$ and $P(y = -1)$ of the positive interval is larger than that of the negative interval. Accordingly, we shall move the hyperplane toward the negative direction (i.e., set $sign$ to -1) to have more negative samples located along the positive interval to increase E^+.

Case 5: $E^+ > E^-$. This case is the opposite of Case 4. Given the fact that $E^- < E^+$ and $E^+ \leq 1$, we can conclude that the difference between $P(y = 1)$ and $P(y = -1)$ of the negative interval is larger than that of the positive interval. Accordingly, we shall move the hyperplane toward the positive direction (i.e., set $sign$ to 1) to have more positive samples located along the negative interval to increase E^-.

Algorithm 1: Hyperplane Generation and Evaluation (in short: GE)

Input : Training samples $S = \{x_i, i = 1, \cdots, N\}$; A support vector machine SVM; $k = \frac{C^+}{C^-}$.
Output: A pair of $(c, sign)$ where $sign$ indicates which direction the hyperplane c should be moved to.

1 $c \leftarrow$ learning a SVM classifier based on k;
2 *Z-axis* $\leftarrow$ mapping S based on c and Equation (8);
3 $E^+, E^-, d^+, d^- \leftarrow 0$;
4 **for** *each point* z_i *on the positive side of the* Z-axis **do**
5 $\quad E_i \leftarrow$ calculate the entropy for the interval $[0, z_i]$;
6 $\quad$ **if** $E_i \geq E^+ \wedge z_i > d^+$ **then**
7 $\quad\quad E^+ \leftarrow E_i$, $d^+ \leftarrow z_i$;
8 **for** *each point* z_i *on the negative side of the* Z-axis **do**
9 $\quad E_i \leftarrow$ calculate the entropy for the interval $[z_i, 0]$;
10 $\quad$ **if** $E_i \geq E^- \wedge |z_i| > d_-$ **then**
11 $\quad\quad E^- \leftarrow E_i$, $d^- \leftarrow |z_i|$;
12 **if** $E^+ = E^- \wedge d^+ = d^-$ **then**
13 $\quad sign \leftarrow 0$;
14 **else if** $(E^+ = E^- \wedge d^+ > d^-) \vee E^+ > E^-$ **then**
15 $\quad sign \leftarrow 1$;
16 **else**
17 $\quad sign \leftarrow -1$;
18 **return** $(c, sign)$;

Algorithm 1 lists the pseudo code of the GE module. First, it employs a given support vector machine SVM to learn a hyperplane c based on the input error cost k (line 1). Second, it constructs a Z-axis orthogonal to the hyperplane c and maps all the samples to Z-axis according to Equation (8) (line 2). Next, it tries to find a proper positive interval with maximal entropy (lines 4-7). To be more specific, each sample z_i mapped to the positive side of the Z-axis defines a candidate positive interval $[0, z_i]$. It adopts a brute-force approach to evaluate all the candidate positive intervals, and chooses the one with the maximum entropy as the positive interval. It finds the negative interval following the same approach (lines 8-11). Thereafter, it sets $sign$ to a proper label (i.e., -1, 0, or 1) based on the values of E^+, E^-, d^+ and d^-, as discussed above (lines 12-17). Finally, it returns c and $sign$ to complete one iteration of hyperplane generation and evaluation.

4.2 Hyperplane Adjustment

Hyperplane adjustment module performs the real adjustment of a hyperplane, guided by the output of the previous module. It takes in four parameters as input, including i) the training set S, ii) a support vector machine SVM, iii) $count$ that defines the upper-bound of the total number of adjustments, and iv) $increment$ that decides how significant

Algorithm 2: Hyperplane Adjustment

Input : Training set S; A support vector machine SVM; $count$; $increment$.
Output: a preferable SVM classifier;

```
1  k ← 1, iter ← 0, sign ← ∞;
2  while iter < count ∧ sign ≠ 0 do
3      (c, sign) ← GE(S, SVM, k);
4      iter ← iter + 1;
5      if sign = −1 then
6          k ← k + increment;
7      else if sign = 1 then
8          increment ← increment/2;
9          k ← k − increment;
10 return classifier c;
```

each adjustment shall be. It outputs a hyperplane which is expected to be able to achieve a good classification performance.

Algorithm 2 lists the pseudo-code of hyperplane adjustment. Parameter k refers to the error cost ratio ($k = \frac{C^+}{C^-}$), parameter $iter$ counts the number of adjustments being performed, and label $sign$ guides the adjustment. Without any knowledge of the dataset, it sets the initial value of k to 1 (i.e., assuming $C^+ = C^-$). Meanwhile, it sets $iter$ to 0 and sets $sign$ to ∞ to finish the initialization. Thereafter, it starts a sequence of adjustments indicated by the *while* loop (lines 2-9). In each iteration of the loop, it invokes GE (i.e., Algorithm 1) to generate a hyperplane c based on the current k and evaluate the performance of c (line 3), increases the counter $iter$ by one (line 4) and adjusts the error cost ratio k accordingly (lines 5-9).

Note that the adjustment label $sign$ returned by GE has three possible values, i.e., -1, 0, and 1. If $sign$ is -1, we move the current hyperplane toward the negative direction of Z-axis. To this end, we increase k because a bigger k implies a higher cost of misclassifying the positive class so that the hyperplane learned based on this bigger k is expected to be less skewed toward the positive class (the minority class). Similarly, if $sign$ is 1, we decrease k which decreases the cost of misclassifying the positive class so that the hyperplane learned based on this smaller k is expected to be more skewed toward the positive class. The adjustment of k is based on the input parameter $increment$. Note that in order to make the auto-adjustment less sensitive to the value of $increment$, we reduce the value of $increment$ by half whenever k is reduced (line 8) to enable a finer tuning gradually. If $sign$ is 0, we can safely terminate the loop as the current hyperplane is expected to be able to achieve a good classification performance. The total number of times the loop is performed is bounded by parameter $count$.

Those who are familiar with existing SVM techniques may notice that *ATEC* requires two inputs (i.e., $count$ and $increment$) while existing SVM techniques only require one input (i.e., k). Does *ATEC* bring benefits to the tuning of k as the number of parameters *ATEC* relies on is larger than that of SVM? The answer is definitely yes. The classification accuracy of standard SVM and its variants is *extremely* sensitive to the value of k, and a small change in k could cause a significant change in the classification accuracy. However, *ATEC* is far less sensitive to the parameters $count$ and $increment$ which will be further demonstrated via experimental study to be presented later. In addition, $count$ and $increment$ are bounded to each other, as stated in Theorem 1.

Theorem 1. Let k^* denote a proper setting of k under which the generated hyperplane is expected to achieve a good classification accuracy. Given $count = m$, $R = [\frac{k^*-1}{m}, (k^* - 1) \cdot 2^{(m-1)}]$ defines a range such that as long as $increment$ is within the range R, the k tuned by ATEC is expected to be close to k^* or be equivalent to k^*.

Proof. The main idea of Algorithm 2 is to tune k to make it close to k^* as much as possible after m iterations. There are two extreme cases which help define the boundary of $increment$. In one extreme, the value of $increment$ is so small that *ATEC* needs to keep adding $increment$ to k for m times to move k nearer to k^*. Consequently, $\frac{k^*-1}{m}$ defines the lower bound of $increment$. In the other extreme, the value of $increment$ is very big such that $1 + increment$ already exceeds k^*. In this case, ATEC needs to keep deducting $\frac{increment}{2}$, $\frac{increment}{2^2}$, $\frac{increment}{2^3}$, ... from k in the remaining $m - 1$ iterations. In other words, after $m - 1$ iterations of reduction, $k = 1 + \frac{increment}{2^{(m-1)}}$. Therefore, $(k^* - 1) \cdot 2^{(m-1)}$ defines the maximum value of $increment$. ■

Given the fact that $count$ is not a small integer (e.g., ≥ 10), $[\frac{k^*-1}{count}, (k^* - 1) \cdot 2^{(count-1)}]$ defines a relatively wide range for $increment$. In other words, $count$ can be easily fixed at an integer (e.g., 10 or 20) and $increment$ has a wide range of value options. Consequently, it is not hard for *ATEC* to select proper $count$ and $increment$ to obtain a preferable hyperplane. Our empirical studies to be presented in Section 5 will further demonstrate the insensitivity of *ATEC* to parameters.

5 EXPERIMENT

In this section, we study the performance of ATEC from two perspectives, i.e., effectiveness and efficiency. Similar to existing work on class-imbalance learning [18], [19], [20], we use F1 score of the minority class, area under the precision-recall curve (AUC-PR) and area under the ROC curve (AUC-ROC) as the main performance metrics. We use RBF kernel [21] for all SVMs and tune the parameters C and γ to the optimum by performing grid-parameter-search. The grid search algorithm performs an exhaustive search through a manually specified subset of the hyperparameter space of a learning algorithm and outputs the settings that achieve the highest score in the validation procedure, typically measured by stratified cross validation on the data set. All experiments in our work are evaluated on a server with Intel(R) Core(TM) i5-6500 CPU 3.20 GHz processor and 8 GB RAM with Windows 7 and Python 3.6.4. Besides, we use the well-known libsvm software package [22] to implement all SVMs used in this paper.

5.1 Datasets

We employ two groups of datasets that are widely used for class-imbalance learning, namely *Reuters-21578* and *KEEL*.

TABLE 2
Description of five datasets from Reuters-21578

Dataset	Total	Pos.	Neg.	I.R.
coffee_vs_cocoa	211	68	143	2.10
trade_vs_jobs	582	68	514	7.56
grain_vs_cotton	636	62	574	9.26
acq_vs_coffee	2353	143	2210	15.45
acq_vs_cocoa	2278	68	2210	32.50

TABLE 3
Descriptions of the KEEL datasets

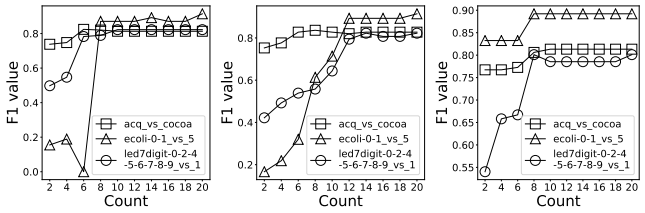
Dataset	Total	Pos.	Neg.	I.R.
wisconsin	683	239	444	1.86
pima	768	268	500	1.87
iris0	150	50	100	2.00
haberman	306	81	225	2.78
vehicle2	846	218	628	2.88
new-thyroid1	215	35	180	5.14
yeast3	1484	163	1321	8.10
vowel0	988	90	898	9.98
led7digit-0-2-4-5-6-7-8-9_vs_1	443	37	406	10.97
ecoli-0-1_vs_5	240	20	220	11.00
cleveland-0_vs_4	173	13	160	12.31
glass4	224	13	201	15.46
page-blocks-1-3_vs_4	472	28	444	15.86
dermatology-6	358	20	338	16.90
zoo-3	101	5	96	19.20
shuttle-6_vs_2-3	230	10	220	22.00
winequality-red-4	1599	53	1546	29.17
poker-9_vs_7	244	8	236	29.50

Reuters-21578, to be more specific *Reuters-21578 Text Categorization Test Collection* [23], is from news media and contains documents from different text categories. We only consider documents of those categories having sizes larger than 50 (a total of seven categories), and generate five synthetic datasets with different I.R. values. Each synthetic dataset is generated by simply merging two text categories that are in different sizes, with one serving as the majority (negative) class and the other serving as the minority (positive) class. Table 2 lists the detailed descriptions of these five datasets. For preprocessing these text documents, we use Stanford NLP tool [24] to perform the word segmentation and to remove stop-words and punctuation. Moreover, we use a vector to represent each word in a document based on the popular pre-trained word2vec model provided by Google [25], where 300-dimensional vectors for 3 million words and phrases are involved, and represent the document based on the average of all the word-vectors.

KEEL is from KEEL-dataset repository [26] that involves different domains, such as health, business, plant and etc. There are tens of imbalanced datasets in the repository and each domain contains multiple datasets. We select one dataset from each domain as the representative for evaluation, in total 18 datasets. Table 3 lists the detailed descriptions of these datasets.

5.2 Effectiveness Study

To evaluate the effectiveness of ATEC, we evaluate seven variants of SVMs and four non-SVM algorithms that are widely used for class-imbalance problem. Seven SVM variants include (1) **DEC**, a SVM variant which merely considers different error costs between classes; (2) **FSVM-CIL$^{\mathbf{cen}}_{\mathbf{lin}}$**, a variant of FSVM-CIL where the membership function is designed based on the distance from the own class center



(a) DEC-ATEC (b) FSVM-CIL$^{\text{cen}}_{\text{lin}}$-ATEC (c) FSVM-CIL$^{\text{hyp}}_{\text{lin}}$-ATEC

Fig. 5. ATEC v.s. $count$ ($increment = 100$)

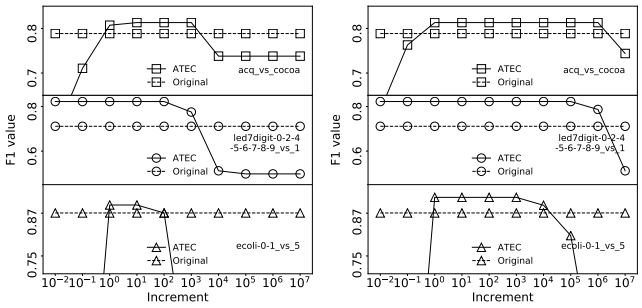
[1]; (3) **FSVM-CIL$^{\mathbf{hyp}}_{\mathbf{lin}}$**, a variant of FSVM-CIL where the membership function is designed based on the distance from the actual hyperplane [1]; (4) **DEC-ATEC**, where we use *ATEC* to improve DEC; (5) **FSVM-CIL$^{\mathbf{cen}}_{\mathbf{lin}}$-ATEC**, which is the improved FSVM-CIL$^{\text{cen}}_{\text{lin}}$ algorithm with *ATEC*; (6) **FSVM-CIL$^{\mathbf{hyp}}_{\mathbf{lin}}$-ATEC**, where we combine FSVM-CIL$^{\text{hyp}}_{\text{lin}}$ with *ATEC*; and (7) **GWSVM-RU** [27], the state-of-the-art SVM variant for class-imbalance problem. We purposely include GWSVM-RU into our comparison study to demonstrate that applying *ATEC* technique on other $\frac{C_+}{C_-}$ based SVMs is able to achieve the state-of-the-art improvement. The non-SVM algorithms we used are: (1) **EasyEnsemble** [28], an ensemble method combined with undersampling; (2) **CBO** (Cluster-Based Oversamping) [29] that employs the K-means technique to separately cluster the majority class and the minority class and performs random oversampling to balance between clusters in each class and balance between classes; (3) **LexiBoost** [11], a boosting method based on a lexicographic linear programming framework which does not need to set error costs; and (4) **Dual-LexiBoost** [11], a dual method of LexiBoost. In the following, we first study the sensitivity of *ATEC* to its parameters $count$ and $increment$, and we then compare the performance of above eleven algorithms.

5.2.1 *Inside* ATEC

In order to find a proper value of $\frac{C_+}{C_-}$, *ATEC* requires two parameters $count$ and $increment$ as inputs. Accordingly, we first study their impact on the effectiveness of *ATEC* and then propose a guideline to help those users who want to use *ATEC* to improve SVMs for class-imbalance problems to determine the parameters. We report the results of three different datasets from different domains (i.e., acq_vs_cocoa from *Reuters-21578*, led7digit-0-2-4-5-6-7-8-9_vs_1 and ecoli-0-1_vs_5 from *KEEL*). We observe similar findings from other datasets but skip those results for space-saving.

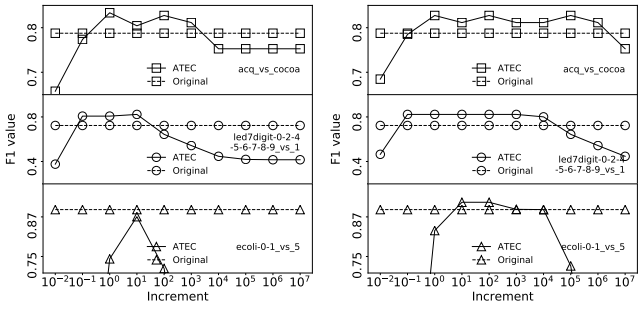
Firstly, we fix *increment* to 100 and vary $count$ from 2 to 20 to observe the performance changes of three SVM variants with *ATEC*. Figures 5(a), 5(b) and 5(c) plot the results of F1 value of DEC-ATEC, FSVM-CIL$^{\text{cen}}_{\text{lin}}$-ATEC and FSVM-CIL$^{\text{hyp}}_{\text{lin}}$-ATEC respectively. We can see that with the increase of $count$, there is a rough trend that F1 value firstly grows up and then stabilizes in all cases. It means that classifying performance can be improved by increasing $count$ and it will converge to a stable result.

Next, we fix $count$ to study the impact of $increment$. We perform two sets of experiments by fixing $count$ to 10 and 20 respectively. We evaluate the performance of SVMs with *ATEC* and that of their original SVMs without *ATEC*,



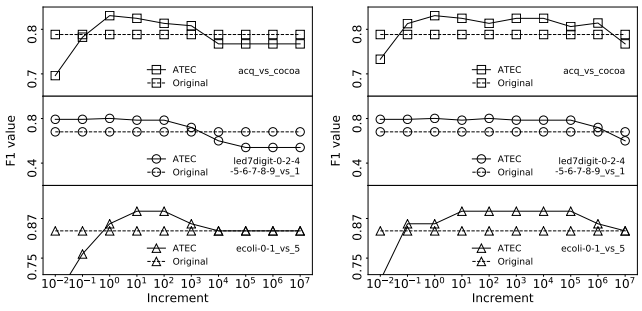
(a) $count = 10$ (b) $count = 20$

Fig. 6. DEC-ATEC v.s. $increment$



(a) $count = 10$ (b) $count = 20$

Fig. 7. FSVM-CIL$_{\text{lin}}^{\text{cen}}$-ATEC v.s. $increment$



(a) $count = 10$ (b) $count = 20$

Fig. 8. FSVM-CIL$_{\text{lin}}^{\text{hyp}}$-ATEC v.s. $increment$

with the results plotted in Figures 6, 7 and 8. The solid lines represent the performance of the algorithms with *ATEC* while the dotted lines represent the performance of the original algorithms where the error cost ratio is set to the imbalance ratio of the dataset.

In most cases, F1 value, with an increase of $increment$, is observed to firstly grow up to reach a relatively stable value. Thereafter, a further increase of $increment$ forces F1 value to decrease. Besides, F1 value corresponding to the algorithms with *ATEC* is higher than that of the respective original algorithms when the $increment$ is within a certain range. What's more important, this range is influenced by $count$. Take acq-cocoa dataset as an example. DEC-ATEC performs better than DEC when $count$ is 10 and $increment$ is within [1, 10^3] (see Figure 6(a)). When $count$ is set to 20, the range of $increment$ within which DEC-ATEC can perform better than DEC is extended to [0.5, 10^6] (see Figure 6(b)). It can be explained by Theorem 1. That is, the upper bound of $increment$ has an exponential relationship with $count$ while the lower bound of $increment$ has an inverse relationship with $count$. When $count$ increases, the lower bound of $increment$ will decrease slightly while its upper bound will grow up exponentially. This phenomenon is consistent with Theorem 1.

Based on the above observations, we can conclude that *ATEC* is not sensitive to its parameters and it is easy to select the parameter values for *ATEC* to get desirable results, e.g., using relatively small numbers of $count$ and $increment$ (e.g. both ≥ 20), *ATEC* can probably produce a better hyperplane for the given SVM.

5.2.2 Overall Performance

After we evaluate the sensitivity of *ATEC* on the parameters $count$ and $increment$, we are ready to report the F1 score, AUC-PR and AUC-ROC of the eleven algorithms. In total, we perform three-fold stratified cross validation on Reuters datasets and five-fold stratified cross validation on KEEL datasets. In the following, we first explain the experiment settings of different algorithms and then report the experimental results. For all seven SVM variants, we adopt different parameter settings (i.e., SVM's hyperparameters C and γ) for Reuters and KEEL datasets because the datasets have very different distributions. On Reuters datasets, $\log_2 C$ is selected from $\{0, 1, \ldots, 8, 9\}$ and $\log_2 \gamma$ is selected from $\{-9, -8, \ldots, -2, -1\}$. On KEEL datasets, we first conduct a coarse grid-parameter-search to find the optimal values of C and γ (say C^* and γ^*) over the following ranges: $\lg C = \{-2, -1, \ldots, 6, 7\}$, $\lg \gamma = \{-11, -10, \ldots, -2, -1\}$. We then conduct a fine-grained grid-parameter-search to find the final value of parameters over the ranges: $\lg C = \{C^* - 0.9, C^* - 0.8, \ldots, C^* + 0.8, C^* + 0.9\}$, $\lg \gamma = \{\gamma^* - 0.9, \gamma^* - 0.8, \ldots, \gamma^* + 0.8, \gamma^* + 0.9\}$. For all the original SVMs without *ATEC*, we perform the training where the error cost ratio k is set to I.R. of the dataset. For the SVMs with *ATEC*, we set the hyperparameters $count = 20$ and $increment = 100$. For EasyEnsemble, Classification And Regression Tree (CART) [30] is used to train weak classifier, and the number of AdaBoost [31] learners and the number of rounds in each AdaBoost are set to 20 and 100, respectively. For CBO, the K-means parameter of determining the number of clusters is selected from $\{2, 3, 4\}$ since the smallest number of training samples is 4, i.e., as shown in Table 3, four out of five positive samples in zoo-3 can only be used for five-fold training, and after sampling, CART is adopted to train classifier on the balance data. For LexiBoost and Dual-LexiBoost, we conduct corresponding experiments by using the same settings and source code as the original [11].

Reuters Datasets. F1, AUC-PR and AUC-ROC results on Reuters datasets are reported in Table 4, Table 5 and Table 6 respectively. In addition, we report the variance of cross validation in the brackets. There are three main observations. First, the variance is less than 0.1 for all cases, which demonstrates the evaluation scores measured by three-fold cross validation are stable in text data. Second, SVMs with *ATEC* always outperform the original versions without *ATEC* for both F1 score and AUC-PR, which demonstrates the effectiveness of *ATEC*. In terms of AUC-ROC, SVMs with *ATEC* outperform the original versions without *ATEC* in two datasets with substantial advantages; in the other three datasets, the difference between SVMs with *ATEC* and their original versions without *ATEC* is not significant. Third, among five Reuters datasets, the algorithm with

TABLE 4
F1 Values of the Reuters-21578

	acq_vs_ cocoa	trade_vs_ jobs	acq_vs_ coffee	coffee_vs_ cocoa	grain_vs_ cotton
DEC	0.789(0.007)	0.803(0.002)	0.825(0.001)	0.711(0.002)	0.605(0.001)
DEC-ATEC	0.813(0.002)	**0.808**(0.002)	**0.874**(0.000)	0.724(0.007)	0.615(0.001)
FSVM-CIL$_{lin}^{cen}$	0.788(0.003)	0.790(0.002)	0.823(0.000)	0.711(0.002)	0.579(0.003)
FSVM-CIL$_{lin}^{cen}$-ATEC	0.827(0.002)	0.792(0.001)	0.858(0.000)	0.713(0.005)	**0.635**(0.000)
FSVM-CIL$_{lin}^{hyp}$	0.788(0.000)	0.798(0.002)	0.828(0.000)	0.723(0.003)	0.561(0.001)
FSVM-CIL$_{lin}^{hyp}$-ATEC	0.813(0.007)	0.801(0.001)	0.862(0.000)	0.734(0.004)	0.629(0.000)
EasyEnsemble	0.364(0.000)	0.666(0.002)	0.522(0.004)	0.679(0.006)	0.407(0.000)
CBO	0.437(0.011)	0.651(0.001)	0.425(0.002)	0.644(0.006)	0.362(0.002)
LexiBoost	0.567(0.002)	0.656(0.002)	0.747(0.001)	0.659(0.001)	0.318(0.002)
Dual-LexiBoost	0.500(0.003)	0.704(0.000)	0.560(0.000)	0.729(0.000)	0.344(0.000)
GWSVM-RU	0.839(0.002)	0.746(0.006)	0.857(0.001)	**0.792**(0.004)	0.461(0.006)
DEC-BOA(20)	**0.844**(0.004)	0.725(0.005)	0.781(0.001)	0.653(0.006)	0.407(0.011)
DEC-BOA(1000)	0.743(0.012)	0.688(0.036)	0.764(0.002)	0.725(0.000)	0.411(0.013)
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.834(0.003)	0.737(0.005)	0.754(0.000)	0.642(0.005)	0.413(0.012)
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.785(0.008)	0.708(0.022)	0.818(0.005)	0.700(0.013)	0.401(0.042)
FSVM-CIL$_{lin}^{hyp}$-BOA(20)	0.832(0.001)	0.702(0.002)	0.783(0.000)	0.629(0.006)	0.418(0.011)
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.799(0.009)	0.658(0.031)	0.828(0.001)	0.713(0.005)	0.358(0.035)

TABLE 5
AUC-PR of the Reuters-21578

	acq_vs_ cocoa	trade_vs_ jobs	acq_vs_ coffee	coffee_vs_ cocoa	grain_vs_ cotton
DEC	0.838(0.002)	0.858(0.001)	0.901(0.001)	0.815(0.005)	0.624(0.004)
DEC-ATEC	0.852(0.004)	**0.891**(0.001)	0.907(0.001)	0.816(0.005)	0.671(0.003)
FSVM-CIL$_{lin}^{cen}$	0.860(0.005)	0.858(0.000)	0.898(0.000)	0.824(0.005)	0.639(0.004)
FSVM-CIL$_{lin}^{cen}$-ATEC	0.864(0.006)	0.881(0.000)	0.907(0.001)	0.827(0.004)	**0.681**(0.003)
FSVM-CIL$_{lin}^{hyp}$	0.846(0.002)	0.858(0.000)	0.899(0.001)	0.802(0.004)	0.624(0.005)
FSVM-CIL$_{lin}^{hyp}$-ATEC	0.856(0.005)	0.879(0.000)	**0.909**(0.002)	0.808(0.005)	0.665(0.003)
EasyEnsemble	0.719(0.005)	0.833(0.001)	0.829(0.002)	0.793(0.008)	0.584(0.004)
CBO	0.497(0.002)	0.652(0.004)	0.430(0.003)	0.708(0.006)	0.381(0.001)
LexiBoost	0.241(0.000)	0.526(0.007)	0.555(0.001)	0.483(0.004)	0.188(0.000)
Dual-LexiBoost	0.607(0.002)	0.748(0.000)	0.652(0.000)	0.783(0.000)	0.513(0.000)
GWSVM-RU	**0.891**(0.005)	0.847(0.014)	0.887(0.000)	0.867(0.004)	0.564(0.022)
DEC-BOA(20)	0.852(0.010)	0.769(0.019)	0.877(0.001)	0.727(0.008)	0.473(0.032)
DEC-BOA(1000)	0.858(0.009)	0.813(0.021)	0.889(0.000)	0.866(0.006)	0.526(0.044)
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.868(0.006)	0.753(0.023)	0.870(0.002)	0.719(0.003)	0.469(0.024)
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.884(0.006)	0.755(0.019)	0.900(0.000)	0.858(0.006)	0.545(0.048)
FSVM-CIL$_{lin}^{hyp}$-BOA(20)	0.883(0.005)	0.756(0.019)	0.871(0.002)	0.715(0.004)	0.476(0.033)
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.869(0.008)	0.810(0.015)	0.893(0.001)	**0.868**(0.004)	0.513(0.042)

ATEC achieves the best performance in three datasets while GWSVM-RU ranks the first in the other two datasets (i.e., acq_vs_cocoa and coffe_vs_cocoa) for both F1 score and AUC-PR. It indicates that SVMs with *ATEC* are comparable with the state-of-the-art SVM variant. According to the above observations, we can conclude that i) *ATEC* can improve the performance of existing SVM variants in term of F1 value of minority class, AUC-PR and AUC-ROC on text datasets, and ii) ATEC can help SVMs improve F1 value and AUC-PR score more effectively, as compared with AUC-ROC. Besides, our intuition that $\frac{C_+}{C_-} = I.R.$ is not the best choice for all cases is correct. The ratio of inter-class cost errors, i.e. $\frac{C_+}{C_-}$ found by *ATEC* is better than $I.R.$ in our experiments.

KEEL Datasets. To further verify our conclusion made based on above experiments, we run the experiments on KEEL dataset. Table 7, Table 8 and Table 9 respectively report the corresponding F1 value, AUC-PR score and AUC-ROC score. We also list the variance of cross validation in the brackets. From Table 7 we can see: (1) *ATEC* is able to improve the F1 score of the original algorithm in 15 out of 18 datasets. (2) Algorithms equipped with *ATEC* become the *only* best performer in 3 datasets and the best performer together with many other algorithms in 6 datasets. GWSVM-RU is the *only* best performer in 3 datasets. On the contrary, CBO and LexiBoost respectively become the only best performer in 1 dataset. The advantage of using *ATEC* is more obvious when we use AUC-PR metric. For the results shown in Table 8: (1) *ATEC* enhances the AUC-PR value of the original algorithm in 17 out of 18 datasets.(2) Algorithms equipped with *ATEC* become the *only* best performer in 3 datasets and the best performer together with many other algorithms in 7 datasets. GWSVM-RU and Dual-LexiBoost become the *only* best performer in 2 datasets respectively. On the contrary, EasyEnsemble becomes the only best performer in 1 dataset. The results of AUC-ROC are shown in Table 9: (1) *ATEC* could improve the AUC-ROC score of the original algorithm in 17 out of 18 datasets. (2) Algorithms equipped with *ATEC* become the *only* best performer in 2 datasets and the best performer together with many other algorithms in 9 datasets. GWSVM-RU becomes the *only* best performer in 2 datasets and EasyEnsemble becomes the only best performer in 1 dataset.

TABLE 6
AUC-ROC of the Reuters-21578

	acq_vs_ cocoa	trade_vs_ jobs	acq_vs_ coffee	coffee_vs_ cocoa	grain_vs_ cotton
DEC	0.963(0.001)	0.963(0.000)	0.982(0.000)	0.855(0.005)	0.884(0.002)
DEC-ATEC	0.962(0.001)	0.965(0.000)	0.980(0.000)	0.861(0.004)	**0.898**(0.001)
FSVM-CIL$_{lin}^{cen}$	0.964(0.002)	0.965(0.000)	0.983(0.000)	0.861(0.004)	0.886(0.001)
FSVM-CIL$_{lin}^{cen}$-ATEC	0.962(0.001)	0.966(0.000)	0.979(0.000)	0.874(0.002)	0.893(0.001)
FSVM-CIL$_{lin}^{hyp}$	0.962(0.001)	0.961(0.000)	**0.983**(0.000)	0.843(0.005)	0.875(0.001)
FSVM-CIL$_{lin}^{hyp}$-ATEC	0.961(0.002)	0.963(0.000)	0.980(0.000)	0.857(0.002)	0.891(0.000)
EasyEnsemble	**0.967**(0.000)	0.945(0.001)	0.975(0.000)	0.839(0.004)	0.799(0.011)
CBO	0.762(0.006)	0.782(0.004)	0.685(0.002)	0.719(0.001)	0.623(0.001)
LexiBoost	0.835(0.002)	0.859(0.000)	0.862(0.000)	0.700(0.000)	0.656(0.001)
Dual-LexiBoost	0.903(0.001)	0.911(0.000)	0.902(0.000)	0.803(0.000)	0.717(0.002)
GWSVM-RU	0.964(0.001)	**0.968**(0.001)	0.974(0.000)	0.888(0.007)	0.884(0.002)
DEC-BOA(20)	0.955(0.001)	0.953(0.001)	0.975(0.000)	0.828(0.002)	0.837(0.006)
DEC-BOA(1000)	0.957(0.001)	0.965(0.001)	0.979(0.000)	0.881(0.007)	0.894(0.003)
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.957(0.001)	0.951(0.001)	0.974(0.000)	0.838(0.004)	0.844(0.006)
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.961(0.001)	0.961(0.001)	0.975(0.000)	**0.890**(0.005)	0.878(0.007)
FSVM-CIL$_{lin}^{hyp}$-BOA(20)	0.960(0.001)	0.951(0.001)	0.976(0.000)	0.856(0.008)	0.834(0.007)
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.958(0.001)	0.963(0.001)	0.982(0.000)	0.882(0.004)	0.844(0.011)

Discussion. Furthermore, we compute the ranks of all eleven algorithms based on their performances in KEEL dataset. In our rankings, the best performer ranks 1, the second best gets the rank of 2, etc. Note that in the case of ties, the average of the ranks that would have been assigned without ties is assigned. For example, DEC-ATEC and FSVM-CIL$_{lin}^{cen}$-ATEC tie for first place on ecoli-0-1_vs_5, hence they both are assigned the ranks of 1.5 (i.e., $\frac{1+2}{2}$). After that, we conclude their final average ranks of all datasets in Table 10. We have the following three main observations. First, LexiBoost and Dual-LexiBoost have the low rankings,

reflecting that their performance on these datasets is less stable than that of SVM variants. Second, algorithms with *ATEC* achieve higher rankings than their original versions in most cases. Third, among seven SVM variants, GWSVM-RU and the algorithms with *ATEC* have the highest ranking in terms of F1 scores while algorithms with *ATEC* perform consistently better than GWSVM-RU in terms of AUC-PR. For AUC-ROC, *ATEC* does not demonstrate a big impact as the SVMs with *ATEC* and those without *ATEC* perform similarly. All these observations remain valid on Reuters dataset, as reported in Table 10. Based on the above results and discussion, we can safely conclude that *ATEC* not only improves existing SVM variants effectively but also remains competitive with the state-of-the-art imbalanced methods. In addition, we have observed that compared with the results of AUC-PR and F1 value, the advantage of using *ATEC* is not significant in AUC-ROC. This might be caused by *ATEC*'s unique characteristics, i.e., *ATEC* will not be biased toward either the minority class or the majority class, while ROC curves prefer the classifiers biased toward the majority class [32].

5.2.3 Friedman Test

In order to show the difference among 11 algorithms on the effectiveness, we further conduct a statistic significance test based on the average ranks reported in Table 10. We select the Friedman test [33] that can be used to investigate the difference among the various classifiers on multiple datasets. The null-hypothesis indicates that all the algorithms are equivalent so that they shall share equivalent average ranks. Take KEEL datasets as an example. The results of Friedman statistic F_F on F1 value, AUC-PR and AUC-ROC score are reported in Table 10. The p-values computed by the values of statistic F_F are smaller than 0.0001[1], which indicates that the null-hypothesis is rejected with high probability. That is to say the compared algorithms are significantly different. This conclusion remains also valid on Reuters dataset.

5.2.4 Comparison with Bayesian Optimization

In this subsection, we perform the comparative experiment of ATEC and the *Bayesian Optimization Algorithm (BOA)*. BOA is a universal method for tuning hyperparameters and is widely used in practice. In order to demonstrate the superior capability of ATEC in tuning the error cost for between-class samples, we implement another three variants of SVM that rely on BOA, instead of ATEC, for parameter tuning, including (1) **DEC-BOA**; (2) **FSVM-CIL$^{\text{cen}}_{\text{lin}}$-BOA**; and (3) **FSVM-CIL$^{\text{hyp}}_{\text{lin}}$-BOA**. We use the popular library *Hyperopt* to implement BOA that requires three inputs, including the minimum objective function, the number of iterations and the range of hyperparameter. We set the objective to $(1 - M_v)$, with metrics M_v taking the F1 value, or AUC-PR, or AUC-ROC score. Apparently, the objective function that minimizes $(1 - M_v)$ actually aims at maximizing the corresponding metric. The value of M_v is evaluated as the average of 4-fold cross-validation on the training dataset. We are only able to perform 4-fold as the smallest training dataset *zoo-3* has only 4 samples. We set the number of iterations to be either 20 or 1000. Note, $count$ value of ATEC that determines the number of iteration is set to 20. In addition to value of 20, we also test the performance of BOA under 1000 iterations to explore the effectiveness of BOA with the sufficient number of iterations. We obtain the optimal error cost after executing BOA and apply the optimal error costs to train SVM variants.

To save space, we report the results of SVMs with BOA in Tables 4 - 9, at the last six rows. For each SVM variant with BOA, we report two versions with different number of iterations, indicated by the number (either 20 or 1000) inside the bracket. In terms of F1 values, SVM variant with *ATEC* performs better than that with BOA in 16 out of 23 datasets and 5 ties; in terms of AUC-PR score, SVM variant with *ATEC* outperforms that with BOA in 14 out of 23 datasets and 7 ties; in terms of AUC-ROC score, SVM variant with *ATEC* outperforms that with BOA in 12 out of 23 datasets and 9 ties. These statistics demonstrate that *ATEC* is superior to BOA in most datasets under all three performance metrics. Furthermore, we report the average performance under three different metrics on 23 datasets in Table 11. Note a higher value indicates a better performance for the respective performance metric (i.e., F1 score, AUC-PR score or AUC-ROC score). First, SVM variant with *ATEC* always achieves a much higher value than the version with BOA, which indicates that *ATEC* tunes the error cost more effectively than BOA. Second, SVM variant with BOA performs better when the number of iterations is increased from 20 to 1000. It suggests that 20 iterations are not sufficient for BOA tuning the error cost while *ATEC* can achieve a reasonably good performance under 20 iterations.

5.3 Efficiency Study

In addition to the effectiveness, algorithm efficiency is also an objective when we design *ATEC*. In this set of experiments, we study the efficiency of *ATEC* when searching for a proper value of $\frac{C_+}{C_-}$. Because of space limitation, we only report the performance of DEC-ATEC. The reason that we select DEC-ATEC as the representative is because many error-cost based SVM variants, e.g., FSVM-CIL$^{\text{cen}}_{\text{lin}}$, are derived from DEC and DEC could serve as a base of error-cost based SVM variants. Note that *ATEC* is an add-on component to error-cost based SVMs for tuning $\frac{C_+}{C_-}$ to a proper value. We expect the findings from DEC-ATEC to remain valid on other error-cost based SVM variants. Indeed our expectation is consistent with what we find from another set of comparison experiments on other error-cost based SVMs and their enhanced versions with *ATEC*, whose results are skipped for space-saving. We report the time required by the popular grid-search strategy for tuning the value of $\frac{C_+}{C_-}$ to provide a benchmark.

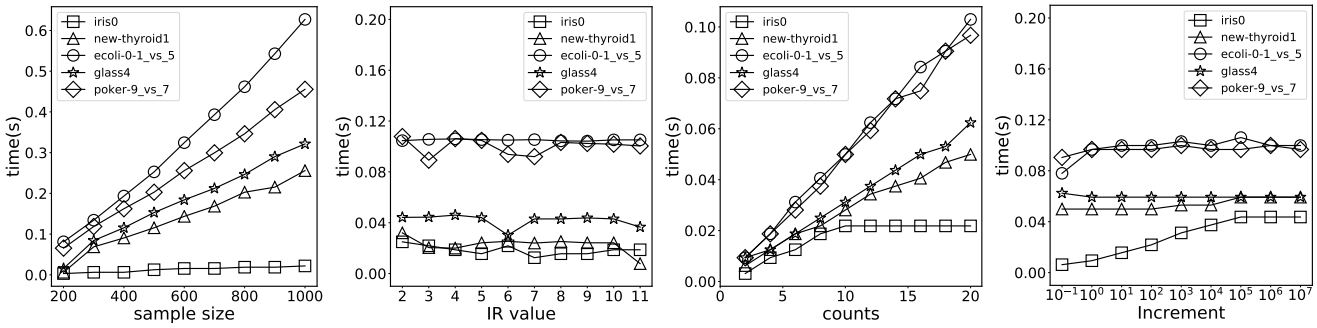
5.3.1 Evaluation of Different Factors

First of all, we study the efficiency of *ATEC* by evaluating the impact of four factors that might affect the efficiency, including sample sizes, imbalance ratios (I.R.), $count$ and $increment$. By default, we set the sample size and I.R. according to the original dataset and set $count$ and $increment$ to 20 and 100 respectively. We then change the value of one

1. The p-value is computed from the source - http://graphpad.com/quickcalcs/PValue1.cfm.

TABLE 7
F1 Values of the KEEL datasets

	cleveland-0_vs_4	dermatology-6	ecoli-0-1_vs_5	glass4	haberman	iris0	led7digit-0-2-4-5-6-7-8-9_vs_1	new-thyroid1	page-blocks-1-3_vs_4
DEC	0.771(0.019)	**1.000**(0.000)	0.871(0.006)	**0.933**(0.018)	0.521(0.005)	**1.000**(0.000)	0.711(0.032)	0.987(0.001)	0.869(0.009)
DEC-ATEC	0.656(0.034)	**1.000**(0.000)	**0.914**(0.005)	**0.933**(0.018)	0.495(0.003)	**1.000**(0.000)	**0.823**(0.013)	0.987(0.001)	0.873(0.007)
FSVM-CIL$_{lin}^{cen}$	0.730(0.062)	0.971(0.003)	0.892(0.003)	**0.933**(0.018)	0.518(0.006)	**1.000**(0.000)	0.723(0.031)	0.987(0.001)	0.869(0.009)
FSVM-CIL$_{lin}^{cen}$-ATEC	0.800(0.027)	0.971(0.003)	**0.914**(0.005)	**0.933**(0.018)	0.488(0.007)	**1.000**(0.000)	**0.823**(0.013)	0.987(0.001)	0.887(0.012)
FSVM-CIL$_{lin}^{hyp}$	0.793(0.013)	**1.000**(0.000)	0.833(0.013)	**0.933**(0.018)	**0.536**(0.004)	**1.000**(0.000)	0.681(0.011)	0.987(0.001)	0.869(0.009)
FSVM-CIL$_{lin}^{hyp}$-ATEC	0.740(0.062)	**1.000**(0.000)	0.892(0.003)	**0.933**(0.018)	0.504(0.011)	**1.000**(0.000)	0.801(0.013)	0.987(0.001)	0.874(0.011)
EasyEnsemble	0.450(0.021)	0.978(0.002)	0.641(0.024)	0.468(0.020)	0.427(0.005)	**1.000**(0.000)	0.486(0.024)	0.898(0.004)	0.816(0.004)
CBO	0.507(0.082)	0.978(0.004)	0.806(0.023)	0.660(0.085)	0.367(0.008)	**1.000**(0.000)	0.699(0.008)	0.911(0.010)	**0.982**(0.001)
LexiBoost	0.261(0.022)	0.949(0.001)	0.871(0.001)	0.826(0.007)	0.451(0.000)	**1.000**(0.000)	0.723(0.002)	0.958(0.002)	0.726(0.000)
Dual-LexiBoost	0.273(0.034)	0.801(0.002)	0.836(0.001)	0.720(0.016)	0.510(0.000)	**1.000**(0.000)	0.768(0.006)	0.917(0.000)	0.636(0.000)
GWSVM-RU	**0.807**(0.020)	**1.000**(0.000)	0.855(0.018)	0.867(0.027)	0.511(0.005)	**1.000**(0.000)	**0.823**(0.013)	**1.000**(0.000)	0.873(0.007)
DEC-BOA(20)	0.610(0.054)	0.978(0.002)	0.155(0.000)	**0.933**(0.018)	0.419(0.000)	0.604(0.001)	0.548(0.023)	0.948(0.002)	0.836(0.016)
DEC-BOA(1000)	0.530(0.114)	0.978(0.002)	0.114(0.052)	**0.933**(0.018)	0.370(0.019)	0.604(0.001)	0.805(0.013)	0.926(0.004)	0.862(0.010)
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.594(0.032)	0.950(0.010)	0.191(0.001)	0.893(0.019)	0.428(0.000)	0.696(0.001)	0.488(0.010)	0.960(0.001)	0.856(0.023)
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.594(0.032)	0.950(0.010)	0.564(0.027)	0.893(0.019)	0.477(0.003)	0.707(0.002)	0.620(0.025)	0.933(0.004)	0.817(0.024)
SVM-CIL$_{lin}^{hyp}$-BOA(20)	0.646(0.059)	0.978(0.002)	0.833(0.013)	0.893(0.019)	0.458(0.006)	0.990(0.000)	0.540(0.010)	0.917(0.007)	0.806(0.017)
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.605(0.120)	0.978(0.002)	0.833(0.013)	0.893(0.019)	0.493(0.006)	0.990(0.000)	0.793(0.016)	0.973(0.001)	0.821(0.016)
	pima	poker-9_vs_7	shuttle-6_vs_2-3	vehicle2	vowel0	winequality-red-4	wisconsin	yeast3	zoo-3
DEC	0.686(0.002)	**0.581**(0.106)	**1.000**(0.000)	0.981(0.000)	**1.000**(0.000)	0.180(0.001)	0.967(0.000)	0.747(0.001)	0.600(0.240)
DEC-ATEC	0.650(0.003)	0.467(0.160)	**1.000**(0.000)	**0.984**(0.000)	**1.000**(0.000)	0.206(0.008)	0.968(0.000)	0.795(0.001)	0.600(0.240)
FSVM-CIL$_{lin}^{cen}$	**0.689**(0.002)	0.481(0.062)	**1.000**(0.000)	0.977(0.000)	**1.000**(0.000)	0.177(0.001)	0.965(0.000)	0.751(0.002)	0.400(0.240)
FSVM-CIL$_{lin}^{cen}$-ATEC	0.660(0.001)	0.360(0.198)	**1.000**(0.000)	0.977(0.001)	**1.000**(0.000)	**0.208**(0.002)	0.965(0.000)	0.792(0.002)	0.533(0.204)
FSVM-CIL$_{lin}^{hyp}$	0.681(0.001)	0.433(0.151)	**1.000**(0.000)	0.981(0.000)	**1.000**(0.000)	0.187(0.013)	**0.969**(0.000)	0.748(0.002)	0.600(0.240)
FSVM-CIL$_{lin}^{hyp}$-ATEC	0.660(0.003)	0.400(0.151)	**1.000**(0.000)	**0.984**(0.000)	**1.000**(0.000)	0.204(0.003)	0.967(0.000)	0.798(0.001)	0.600(0.240)
EasyEnsemble	0.645(0.002)	0.183(0.006)	**1.000**(0.000)	0.945(0.001)	0.822(0.000)	0.114(0.001)	0.953(0.000)	0.665(0.001)	0.094(0.006)
CBO	0.589(0.001)	0.333(0.071)	**1.000**(0.000)	0.930(0.001)	0.889(0.002)	0.128(0.006)	0.951(0.000)	0.704(0.006)	0.467(0.178)
LexiBoost	0.562(0.000)	0.373(0.009)	**1.000**(0.000)	0.843(0.000)	0.995(0.000)	0.080(0.001)	0.959(0.000)	0.697(0.001)	**0.800**(0.010)
Dual-LexiBoost	0.653(0.000)	0.373(0.009)	**1.000**(0.000)	0.803(0.000)	0.995(0.000)	0.080(0.001)	0.965(0.000)	0.728(0.000)	0.667(0.006)
GWSVM-RU	0.687(0.000)	**0.581**(0.106)	**1.000**(0.000)	0.981(0.000)	**1.000**(0.000)	0.171(0.002)	0.962(0.000)	**0.802**(0.001)	0.600(0.240)
DEC-BOA(20)	0.571(0.000)	0.244(0.050)	0.083(0.000)	0.977(0.000)	0.995(0.000)	0.102(0.001)	0.790(0.024)	0.643(0.009)	0.600(0.240)
DEC-BOA(1000)	0.630(0.003)	0.278(0.072)	0.083(0.000)	0.977(0.000)	0.995(0.000)	0.118(0.009)	0.876(0.001)	0.789(0.002)	0.600(0.240)
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.526(0.000)	0.360(0.198)	0.083(0.000)	0.973(0.000)	0.984(0.000)	0.110(0.002)	0.964(0.000)	0.657(0.009)	0.400(0.240)
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.654(0.001)	0.360(0.198)	0.083(0.000)	0.977(0.000)	0.984(0.000)	0.143(0.002)	0.965(0.000)	0.782(0.003)	0.400(0.240)
FSVM-CIL$_{lin}^{hyp}$-BOA(20)	0.632(0.003)	0.215(0.062)	**1.000**(0.000)	0.979(0.000)	**1.000**(0.000)	0.108(0.005)	0.873(0.031)	0.689(0.004)	0.600(0.240)
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.669(0.001)	0.467(0.160)	**1.000**(0.000)	0.979(0.000)	**1.000**(0.000)	0.199(0.001)	0.957(0.000)	0.794(0.001)	0.600(0.240)



(a) v.s. sample size (b) v.s. I.R. value (c) v.s. $count$ (d) v.s. $increment$

Fig. 9. Efficiency Evaluation of different factors for ATEC (default: $count = 20$, $increment = 100$, sample size and IR are decided by the original datasets)

parameter only when we study the impact of that parameter, with results depicted in Figure 9.

The impact of the sample size on the DEC-ATEC training time is plotted in Figure 9(a). All five datasets have their original sample size in the range of [150, 250]. We can see that as the number of samples increases, the training time required by DEC-ATEC increases approximately linearly. This is expected since SVM requires a longer time for training more samples. However, the growth rate varies from dataset to dataset and particularly, the iris0 dataset requires an almost constant training time. Multiple factors could cause the difference among growth rates, e.g., the different feature dimension sizes and the dataset distribution. In summary, DEC-ATEC spends less than 1 second in training consistently across all the testing cases.

The DEC-ATEC training time under different I.R. values is plotted in Figure 9(b). We could observe that I.R. value has a limited impact on the training time of DEC-ATEC. Take galss4 dataset as an example. When the I.R. value increases from 2 to 11, the training time fluctuates around 0.04s. In fact, as stated in Section 4, ATEC itself is related to parameters $count$ and $increment$. In this set of tests, since $count$, $increment$ and the sample size are fixed, the training time is relatively stable. The fluctuations are probably caused by the SVM core which is slightly affected by the imbalance ratio and data distribution behind.

The relationship between the parameter $count$ and the time needed for searching the preferable hyperplane of DEC

TABLE 8
AUC-PR of the KEEL datasets

	cleveland-0_vs_4	dermatology-6	ecoli-0-1_vs_5	glass4	haberman	iris0	led7digit-0-2-4-5-6-7-8-9_vs_1	new-thyroid1	page-blocks-1-3_vs_4
DEC	0.862(0.007)	**1.000**(0.000)	0.931(0.002)	0.958(0.007)	0.521(0.013)	**1.000**(0.000)	0.811(0.005)	**1.000**(0.000)	0.873(0.013)
DEC-ATEC	0.794(0.102)	**1.000**(0.000)	0.931(0.002)	0.958(0.007)	0.525(0.012)	**1.000**(0.000)	0.817(0.013)	**1.000**(0.000)	0.870(0.032)
FSVM-CIL$_{lin}^{cen}$	0.801(0.066)	**1.000**(0.000)	**0.933**(0.003)	0.892(0.008)	0.513(0.012)	**1.000**(0.000)	0.805(0.009)	**1.000**(0.000)	0.871(0.014)
FSVM-CIL$_{lin}^{cen}$-ATEC	0.806(0.018)	**1.000**(0.000)	**0.933**(0.003)	0.958(0.007)	0.531(0.008)	**1.000**(0.000)	0.829(0.011)	0.996(0.000)	0.866(0.034)
FSVM-CIL$_{lin}^{hyp}$	0.864(0.012)	**1.000**(0.000)	0.921(0.002)	**1.000**(0.000)	0.524(0.019)	**1.000**(0.000)	0.806(0.009)	**1.000**(0.000)	0.873(0.013)
FSVM-CIL$_{lin}^{hyp}$-ATEC	**0.874**(0.016)	**1.000**(0.000)	0.921(0.002)	0.958(0.007)	0.555(0.019)	**1.000**(0.000)	0.824(0.010)	**1.000**(0.000)	0.870(0.032)
EasyEnsemble	0.657(0.083)	**1.000**(0.000)	0.822(0.031)	0.681(0.071)	0.383(0.007)	**1.000**(0.000)	0.711(0.059)	0.982(0.001)	**1.000**(0.000)
CBO	0.607(0.055)	0.980(0.008)	0.809(0.025)	0.749(0.015)	0.449(0.014)	**1.000**(0.000)	0.740(0.042)	0.927(0.004)	0.983(0.002)
LexiBoost	0.303(0.011)	0.956(0.001)	0.811(0.003)	0.834(0.006)	0.382(0.002)	**1.000**(0.000)	0.789(0.003)	0.965(0.001)	0.671(0.002)
Dual-LexiBoost	0.429(0.008)	0.832(0.001)	0.858(0.001)	0.795(0.008)	**0.592**(0.000)	**1.000**(0.000)	0.779(0.005)	0.929(0.000)	0.700(0.000)
GWSVM-RU	0.854(0.042)	**1.000**(0.000)	0.916(0.014)	0.958(0.007)	0.507(0.014)	**1.000**(0.000)	**0.841**(0.008)	**1.000**(0.000)	0.934(0.017)
DEC-BOA(20)	0.738(0.055)	**1.000**(0.000)	0.891(0.004)	0.958(0.007)	0.400(0.017)	**1.000**(0.000)	0.661(0.032)	**1.000**(0.000)	0.812(0.038)
DEC-BOA(1000)	0.738(0.055)	**1.000**(0.000)	0.895(0.004)	0.958(0.007)	0.448(0.014)	**1.000**(0.000)	0.754(0.033)	**1.000**(0.000)	0.836(0.031)
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.626(0.046)	**1.000**(0.000)	0.918(0.002)	0.958(0.007)	0.411(0.005)	**1.000**(0.000)	0.606(0.036)	**1.000**(0.000)	0.715(0.055)
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.869(0.017)	**1.000**(0.000)	0.918(0.002)	0.958(0.007)	0.441(0.008)	**1.000**(0.000)	0.733(0.038)	**1.000**(0.000)	0.827(0.047)
FSVM-CIL$_{lin}^{hyp}$-BOA(20)	0.779(0.041)	0.989(0.001)	0.921(0.002)	0.922(0.013)	0.327(0.020)	**1.000**(0.000)	0.783(0.006)	0.996(0.000)	0.823(0.042)
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.760(0.050)	**1.000**(0.000)	0.921(0.002)	0.939(0.007)	0.443(0.010)	**1.000**(0.000)	0.788(0.019)	0.996(0.000)	0.847(0.034)
	pima	poker-9_vs_7	shuttle-6_vs_2-3	vehicle2	vowel0	winequality-red-4	wisconsin	yeast3	zoo-3
DEC	0.726(0.002)	0.643(0.093)	**1.000**(0.000)	0.997(0.000)	**1.000**(0.000)	0.159(0.018)	**0.992**(0.000)	0.821(0.006)	**1.000**(0.000)
DEC-ATEC	0.734(0.002)	0.652(0.084)	**1.000**(0.000)	0.998(0.000)	**1.000**(0.000)	0.174(0.015)	**0.992**(0.000)	**0.825**(0.005)	**1.000**(0.000)
FSVM-CIL$_{lin}^{cen}$	0.725(0.002)	0.633(0.122)	**1.000**(0.000)	0.997(0.000)	**1.000**(0.000)	0.156(0.018)	**0.992**(0.000)	**0.825**(0.002)	**1.000**(0.000)
FSVM-CIL$_{lin}^{cen}$-ATEC	**0.738**(0.002)	0.633(0.122)	**1.000**(0.000)	0.997(0.000)	**1.000**(0.000)	0.171(0.004)	**0.992**(0.000)	0.823(0.005)	**1.000**(0.000)
FSVM-CIL$_{lin}^{hyp}$	0.726(0.002)	**0.718**(0.095)	**1.000**(0.000)	0.997(0.000)	**1.000**(0.000)	0.164(0.005)	**0.992**(0.000)	0.822(0.006)	**1.000**(0.000)
FSVM-CIL$_{lin}^{hyp}$-ATEC	0.733(0.003)	0.708(0.080)	**1.000**(0.000)	0.997(0.000)	**1.000**(0.000)	0.160(0.005)	**0.992**(0.000)	0.824(0.004)	**1.000**(0.000)
EasyEnsemble	0.667(0.002)	0.115(0.005)	**1.000**(0.000)	0.982(0.000)	0.986(0.000)	0.120(0.002)	0.990(0.000)	0.789(0.003)	0.441(0.210)
CBO	0.652(0.001)	0.612(0.089)	**1.000**(0.000)	0.932(0.000)	0.899(0.003)	0.155(0.005)	0.962(0.000)	0.745(0.003)	0.710(0.104)
LexiBoost	0.530(0.000)	0.560(0.007)	**1.000**(0.000)	0.668(0.005)	0.995(0.000)	0.117(0.001)	0.925(0.001)	0.612(0.006)	0.805(0.010)
Dual-LexiBoost	0.711(0.000)	0.537(0.020)	**1.000**(0.000)	0.826(0.000)	0.995(0.000)	**0.265**(0.001)	0.968(0.000)	0.764(0.000)	0.767(0.002)
GWSVM-RU	0.724(0.002)	0.651(0.075)	**1.000**(0.000)	**0.999**(0.000)	**1.000**(0.000)	0.163(0.013)	**0.992**(0.000)	0.820(0.007)	**1.000**(0.000)
DEC-BOA(20)	0.618(0.002)	0.643(0.093)	**1.000**(0.000)	0.996(0.000)	**1.000**(0.000)	0.082(0.001)	0.989(0.000)	0.790(0.010)	**1.000**(0.000)
DEC-BOA(1000)	0.729(0.001)	0.647(0.090)	**1.000**(0.000)	0.997(0.000)	**1.000**(0.000)	0.095(0.002)	0.991(0.000)	0.816(0.005)	**1.000**(0.000)
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.644(0.007)	0.633(0.122)	**1.000**(0.000)	0.995(0.000)	**1.000**(0.000)	0.145(0.007)	0.991(0.000)	0.801(0.006)	0.825(0.122)
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.722(0.002)	0.633(0.122)	**1.000**(0.000)	0.997(0.000)	**1.000**(0.000)	0.118(0.003)	0.991(0.000)	0.811(0.006)	0.658(0.175)
FSVM-CIL$_{lin}^{hyp}$-BOA(20)	0.671(0.001)	0.692(0.078)	**1.000**(0.000)	0.997(0.000)	**1.000**(0.000)	0.141(0.001)	0.991(0.000)	0.805(0.006)	**1.000**(0.000)
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.722(0.003)	0.692(0.078)	**1.000**(0.000)	0.997(0.000)	**1.000**(0.000)	0.119(0.002)	0.990(0.000)	0.817(0.005)	**1.000**(0.000)

by ATEC is plotted in Figure 9(c). We can observe that as $count$ increases, ATEC requires longer training times for most datasets. However, it's worth noting that when $count$ reaches 10, the training time corresponding to iris0 dataset becomes stable. This is because that *ATEC* can perfectly separate iris0 samples after 10 iterations so that *ATEC* algorithm will stop searching even when $count$ is larger than 10 and hence the time remains stable.

For the last factor of $increment$, we vary its value from 10^{-1} to 10^{7} on all datasets and report the performances in Figure 9(d). We find that as *increment* increases exponentially, there is almost no obvious difference in the training time for most datasets. This is because that we fix $count$ to 20, and ATEC repeats the while-loop of Algorithm 2 exactly $count = 20$ times in those datasets. Dataset iris0 is the only exception. Different from other datasets, the training time corresponding to iris0 dataset increases as $increment$ changes its value from 10^{-1} to 10^{4}, and it then becomes stable. Most likely it is because when $increment$ has not yet reached 10^{5}, ATEC is able to find a preferable hyperplane before the total number of adjustments reaches $count$ (i.e., the while-loop in Algorithm 2 is terminated because of condition $sign = 0$ but not $iter = count$). In other words, as $increment$ increases its value, the number of adjustments ATEC requires before the preferable hyperplane could be found increases (but still smaller than 20) which explains the increase of the training time. When $increment$ reaches 10^{5}, ATEC is not able to find the preferable hyperplane even after it performs 20 adjustments. That is to say when $increment \geq 10^{5}$, ATEC performs exactly $count = 20$ adjustments which explain the stable training time. Hence, based on the above analysis, we can conclude that comparing with the factors of sample size and $count$ that have a significant influence on the training time of *ATEC*, the $increment$ factor has a moderate impact while the effect from imbalance ratio factor is minor.

5.3.2 Comparison with Grid Search

We further study the efficiency of *ATEC* by comparing DEC-ATEC with the grid-search, a commonly used parameter tuning strategy. To this end, we fix desirable F1 values for different datasets and record the training time required by two techniques to achieve the given F1 score. To make grid-search comparable with *ATEC*, we adopt the following settings to perform the grid-search. We restrict the search space to the range [1,1000] and apply a variable $count$ in grid-search to control the search step size (that is similar to *ATEC*). Specifically, if $count$ is 10, it splits the range [1,1000] into 10 consecutive sub-ranges, e.g., [1,100], [100,200], $\cdots$ and we select the median of each sub-range as the error cost ratio k for training DEC, e.g., 50, 150, $\ldots$, 950. Then, we adopt the best error cost on cross validation as the result of the grid-search. In this way, we can respectively increase the value of $count$ one by one for DEC-ATEC and the grid-search, and record their corresponding training time when the given F1 value is achieved.

TABLE 9
AUC-ROC of the KEEL datasets

	cleveland-0_vs_4	dermatology-6	ecoli-0-1_vs_5	glass4	haberman	iris0	led7digit-0-2-4-5-6-7-8-9_vs_1	new-thyroid1	page-blocks-1-3_vs_4
DEC	0.987(0.000)	**1.000**(0.000)	0.991(0.000)	0.998(0.000)	0.725(0.004)	**1.000**(0.000)	**0.968**(0.000)	**1.000**(0.000)	0.979(0.000)
DEC-ATEC	0.980(0.000)	**1.000**(0.000)	0.991(0.000)	0.998(0.000)	0.730(0.002)	**1.000**(0.000)	0.965(0.001)	**1.000**(0.000)	0.991(0.000)
FSVM-CIL$_{lin}^{cen}$	0.985(0.000)	**1.000**(0.000)	0.991(0.000)	0.998(0.000)	0.723(0.001)	**1.000**(0.000)	0.967(0.001)	**1.000**(0.000)	0.984(0.000)
FSVM-CIL$_{lin}^{cen}$-ATEC	0.982(0.000)	**1.000**(0.000)	0.992(0.000)	0.998(0.000)	**0.741**(0.001)	**1.000**(0.000)	0.966(0.001)	**1.000**(0.000)	0.990(0.000)
FSVM-CIL$_{lin}^{hyp}$	**0.988**(0.000)	**1.000**(0.000)	**0.993**(0.000)	**1.000**(0.000)	0.730(0.002)	**1.000**(0.000)	0.966(0.001)	**1.000**(0.000)	0.979(0.000)
FSVM-CIL$_{lin}^{hyp}$-ATEC	**0.988**(0.000)	**1.000**(0.000)	0.992(0.000)	0.998(0.000)	0.731(0.002)	**1.000**(0.000)	0.965(0.001)	**1.000**(0.000)	0.991(0.000)
EasyEnsemble	0.946(0.003)	**1.000**(0.000)	0.953(0.003)	0.961(0.002)	0.622(0.003)	**1.000**(0.000)	0.943(0.002)	0.996(0.000)	**1.000**(0.000)
CBO	0.786(0.037)	0.999(0.002)	0.882(0.037)	0.818(0.031)	0.553(0.002)	**1.000**(0.000)	0.895(0.008)	0.960(0.003)	0.999(0.000)
LexiBoost	0.684(0.000)	0.974(0.000)	0.919(0.001)	0.940(0.002)	0.589(0.000)	**1.000**(0.000)	0.894(0.002)	0.981(0.001)	0.913(0.000)
Dual-LexiBoost	0.660(0.011)	0.962(0.000)	0.936(0.001)	0.925(0.002)	0.663(0.000)	**1.000**(0.000)	0.902(0.002)	0.981(0.000)	0.948(0.000)
GWSVM-RU	0.977(0.000)	**1.000**(0.000)	0.990(0.000)	0.998(0.000)	0.739(0.003)	**1.000**(0.000)	0.966(0.001)	**1.000**(0.000)	0.996(0.000)
DEC-BOA(20)	0.940(0.002)	**1.000**(0.000)	0.986(0.000)	0.998(0.000)	0.541(0.003)	**1.000**(0.000)	0.927(0.001)	**1.000**(0.000)	0.921(0.015)
DEC-BOA(1000)	0.940(0.002)	**1.000**(0.000)	0.986(0.000)	0.998(0.000)	0.716(0.001)	**1.000**(0.000)	0.942(0.001)	0.999(0.000)	0.897(0.011)
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.971(0.000)	**1.000**(0.000)	0.989(0.000)	0.998(0.000)	0.640(0.007)	**1.000**(0.000)	0.959(0.001)	0.999(0.000)	0.912(0.007)
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.958(0.002)	**1.000**(0.000)	0.990(0.000)	0.998(0.000)	0.706(0.002)	**1.000**(0.000)	0.964(0.001)	0.999(0.000)	0.987(0.000)
FSVM-CIL$_{lin}^{hyp}$-BOA(20)	0.982(0.000)	0.999(0.000)	0.992(0.000)	0.996(0.000)	0.619(0.005)	**1.000**(0.000)	0.940(0.001)	0.999(0.000)	0.940(0.005)
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.986(0.000)	**1.000**(0.000)	0.936(0.013)	0.996(0.000)	0.723(0.001)	**1.000**(0.000)	0.961(0.001)	0.999(0.000)	0.957(0.004)
	pima	poker-9_vs_7	shuttle-6_vs_2-3	vehicle2	vowel0	winequality-red-4	wisconsin	yeast3	zoo-3
DEC	0.834(0.000)	0.979(0.000)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.755(0.005)	**0.996**(0.000)	**0.977**(0.000)	**1.000**(0.000)
DEC-ATEC	0.831(0.000)	**0.981**(0.000)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.744(0.005)	**0.996**(0.000)	0.976(0.000)	**1.000**(0.000)
FSVM-CIL$_{lin}^{cen}$	0.831(0.000)	0.979(0.000)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.753(0.009)	**0.996**(0.000)	0.976(0.000)	**1.000**(0.000)
FSVM-CIL$_{lin}^{cen}$-ATEC	0.833(0.000)	0.979(0.000)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	**0.770**(0.009)	**0.996**(0.000)	0.974(0.000)	**1.000**(0.000)
FSVM-CIL$_{lin}^{hyp}$	0.834(0.000)	0.979(0.000)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.756(0.006)	**0.996**(0.000)	0.977(0.000)	**1.000**(0.000)
FSVM-CIL$_{lin}^{hyp}$-ATEC	0.832(0.001)	0.979(0.000)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.753(0.003)	**0.996**(0.000)	**0.977**(0.000)	**1.000**(0.000)
EasyEnsemble	0.799(0.001)	0.862(0.011)	**1.000**(0.000)	0.993(0.000)	0.998(0.000)	0.685(0.003)	0.994(0.000)	0.971(0.000)	0.853(0.060)
CBO	0.677(0.000)	0.648(0.038)	**1.000**(0.000)	0.951(0.001)	0.936(0.003)	0.555(0.003)	0.963(0.000)	0.838(0.001)	0.790(0.042)
LexiBoost	0.654(0.000)	0.780(0.012)	**1.000**(0.000)	0.864(0.000)	0.999(0.000)	0.544(0.002)	0.956(0.000)	0.749(0.002)	0.861(0.001)
Dual-LexiBoost	0.726(0.000)	0.770(0.013)	**1.000**(0.000)	0.899(0.000)	0.999(0.000)	0.540(0.002)	0.978(0.000)	0.920(0.000)	0.969(0.000)
GWSVM-RU	**0.836**(0.000)	0.979(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	0.710(0.009)	**0.996**(0.000)	0.975(0.000)	**1.000**(0.000)
DEC-BOA(20)	0.734(0.002)	0.977(0.001)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.673(0.011)	0.995(0.000)	0.973(0.000)	0.980(0.002)
DEC-BOA(1000)	0.828(0.000)	0.977(0.001)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.678(0.009)	0.995(0.000)	0.973(0.000)	**1.000**(0.000)
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.759(0.001)	0.979(0.000)	**1.000**(0.000)	0.998(0.000)	**1.000**(0.000)	0.736(0.009)	0.995(0.000)	0.972(0.000)	0.970(0.004)
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.831(0.001)	0.979(0.000)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.749(0.008)	**0.996**(0.000)	0.973(0.000)	0.949(0.004)
FSVM-CIL$_{lin}^{hyp}$-BOA(20)	0.759(0.002)	0.979(0.000)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.705(0.013)	0.995(0.000)	0.974(0.000)	**1.000**(0.000)
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.827(0.001)	0.979(0.000)	**1.000**(0.000)	0.999(0.000)	**1.000**(0.000)	0.725(0.008)	0.995(0.000)	0.975(0.000)	**1.000**(0.000)

TABLE 10
Average Ranks of Different Algorithms

Datasets	Reuters-21578			KEEL		
Performance Metrics	F1 scores	AUC-PR	AUC-ROC	F1 scores	AUC-PR	AUC-ROC
Friedman statistic F_F	12.57	20.36	9.21	10.26	13.37	17.71
DEC	4.8	5.4	5.0	4.6	4.7	4.5
DEC-ATEC	2.6	2.9	3.8	4.1	3.9	4.7
FSVM-CIL$_{lin}^{cen}$	6.6	4.0	3.4	5.4	5.2	4.7
FSVM-CIL$_{lin}^{cen}$-ATEC	3.4	1.9	3.5	4.8	4.5	4.3
FSVM-CIL$_{lin}^{hyp}$	5.2	5.8	5.5	4.8	4.1	3.8
FSVM-CIL$_{lin}^{hyp}$-ATEC	2.4	3.4	5.0	4.1	4.2	4.1
GWSVM-RU	4.0	4.8	3.4	4.1	4.8	4.5
EasyEnsemble	9.4	7.8	6.4	9.5	8.3	7.5
CBO	10.4	10.2	10.8	8.5	8.9	9.3
LexiBoost	9.4	10.8	10.2	8.2	9.5	9.6
Dual-LexiBoost	7.8	9.0	9.0	7.8	8.0	9.1

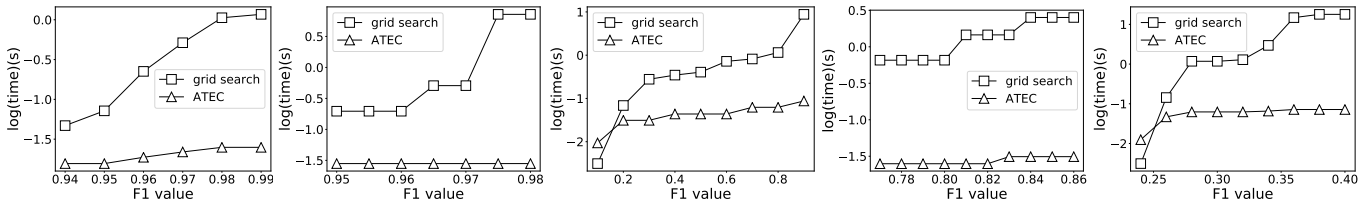
TABLE 11
Average Metrics of KEEL and Reuters Datasets

Performance Metrics	F1 scores	AUC-PR	AUC-ROC
DEC-ATEC	0.791	0.844	0.950
DEC-BOA(20)	0.628	0.795	0.921
DEC-BOA(1000)	0.643	0.820	0.940
FSVM-CIL$_{lin}^{cen}$-ATEC	0.788	0.845	0.952
FSVM-CIL$_{lin}^{cen}$-BOA(20)	0.630	0.780	0.932
FSVM-CIL$_{lin}^{cen}$-BOA(1000)	0.666	0.809	0.945
FSVM-CIL$_{lin}^{hyp}$-ATEC	0.791	0.849	0.950
FSVM-CIL$_{lin}^{hyp}$-BOA(20)	0.718	0.806	0.933
FSVM-CIL$_{lin}^{hyp}$-BOA(1000)	0.756	0.825	0.943

The comparison results between *ATEC* and grid-search on five different KEEL datasets are reported in Figure 10. The target F1 scores vary from dataset to dataset because of their different distributions and other reasons. For example, it is easy for DEC to obtain a F1 score of 0.95 by merely training once on new-thyroid1 dataset (Figure 10(b)); on the other hand, even after we set *count* to a large value, DEC can only achieve F1 score of 0.4 on poker-9_vs_7 dataset (Figure 10(e)). Two observations can be derived from Figure 10. First, for a given dataset, the required training time of both *ATEC* and grid-search techniques extends together with the increase of F1 scores. Second, grid search grows its training time at a much faster rate than *ATEC* and *ATEC* outperforms the grid-search by up to two orders of magnitude when the desired F1 score is set to a high value. For example, as shown in Figure 10(e), both grid-search and *ATEC* spend less than 0.01 second to reach F1 score of 0.2, but as F1 value increases to 0.4, the time goes up to 17.88 seconds for grid-search which is 255 times more than that of for *ATEC*, i.e., 0.07s.

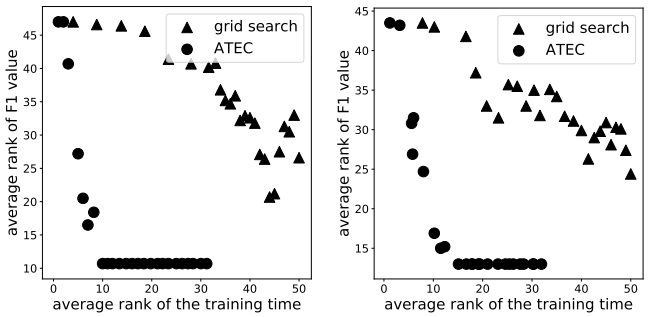
The fact that tuning a parameter requires 17.88 seconds is not unacceptable in real applications. However, there could be more parameters for tuning in SVMs and the size of the dataset could be much larger. If we employ grid-search to tune every single parameter, the final training time could be very expensive, e.g., a few hours. Hence, it is meaningful to perform *ATEC* to help reduce the time required to turn the error cost ratio without sacrificing the classification accuracy.

Last but not least, we study the trade-off between efficiency and effectiveness under 25 different *count* values. To be more specific, we perform in a total of 50 tests for



(a) iris0 (b) new-thyroid1 (c) ecoli-0-1_vs_5 (d) glass4 (e) poker-9_vs_7

Fig. 10. Efficiency study by comparing ATEC with grid search in the KEEL datasets



(a) Reuters datasets (b) KEEL datasets

Fig. 11. Comparing ATEC with grid search in effectiveness and efficiency

each dataset, 25 tests for *ATEC* under 25 different $count$ values and the other 25 tests for grid searches. Note that we have 5 datasets under Reuters and 18 datasets under KEEL, and the total number of tests we run is $50 \times (5 + 18)$. To facilitate the presentation of the results and to save space, we consider the average ranks across multiple datasets instead of the absolute F1 scores/training times in individual tests. Among the 50 tests corresponding to one dataset, the one with the highest F1 score gets rank 1, the one with the second highest F1 score gets rank 2 and so on in the dimension of F1 scores. Similarly, the one incurring the shortest training time gets rank 1, the one incurring the second shortest training time gets rank 2 and so on in the dimension of training time. Let's take *ATEC* with $count = 10$ as an example. In terms of F1 score, its ranks in five Reuters datasets are 5, 5, 7, 6, and 4 respectively. In terms of training time, its ranks in five Reuters datasets are 38, 42, 39, 40, and 45 respectively. It will be represented as a circle of $\langle 5.4, 40.8 \rangle$ in Figure 11(a), with x-axis value 5.4 indicating the average F1 score rank ($\frac{5+5+7+6+4}{5}$) and y-axis value 40.8 indicating the average training time rank ($\frac{38+42+39+40+45}{5}$) across five Reuters datasets. The results are reported in Figure 11. Ideally, a point shall be located at the left lower part of the plot where both the rank of training time and the rank of F1 value are small. In general, both *ATEC* and grid-search take more training time to acquire a higher ranking of F1 value. However, the scatters of *ATEC* are completely below that of grid-search on both Reuters and KEEL datasets. This demonstrates the superiority of *ATEC* in terms of optimization efficiency, i.e., using less time to derive a good classifier.

6 Related Work

The related work on classifying imbalanced datasets can be grouped into three categories, namely *sampling methods*, *algorithmic methods* and *hyperparameters optimization methods*.

Sampling Methods. The purpose of sampling methods is to provide a new balanced dataset by oversampling the minority class or undersampling the majority class with different sampling mechanisms. A classical mechanism is *synthetic sampling*. *SMOTE* [34] is a representative. It creates an artificial dataset based on existing minority examples. Based on SMOTE, some adaptive sampling methods are proposed, such as Borderline-SMOTE [35] and ADA-SYN [36]. Later, Liu et al. propose an *EasyEnsemble* mechanism [28] that trains multiple learners by undersampling the majority class and then combines the outputs of those learners. There are also other ensemble-based variant methods proposed recently [37], [38]. Besides, some works combine sampling methods and SVM. For example, GSVM-RU [20] samples the important examples of the majority class by utilizing the SVM's support vector sets, and Qi et al. [27] proposed GWSVM-RU which improves GSVM-RU by using the weighted SVM. Different from above sampling methods, *ATEC* makes no change to the dataset.

Algorithmic Methods. Algorithmic approaches relieve the imbalance by modifying typical classification algorithms. Wonji et.al [39] introduce a new weight adjustment factor for boosting algorithms by utilizing SVMs on imbalanced data. Khan et.al [40] propose a cost-sensitive deep neural network. Datta et.al [11] propose LexiBoost that takes the equal average hinge losses from the classes as the optimization goal to solve the imbalance problem without cost tuning. Similar to ATEC, the imbalance is addressed through the optimal trade-off between the classes. However, ATEC takes the equal maximizing entropy from the hyperplane sides as the goal to adjust the error cost. In addition, some imbalance algorithms based on SVM have also been proposed. Núñez et al. [41] introduce a new bias that adjusts the classification decision boundary learned by SVM. However, this work improves the minority classification by tuning one parameter of SVM model and may lack generalization performance in some cases. Datta et.al [42] propose to train SVMs as the multi-objective optimization and obtain Pareto optimal as the optimal trade-off between the imbalanced classes without parameter-tuning. However, the method requires a lot of training time for large-scale data sets which limits its usage in real applications. In the meantime, the cost-sensitive learning completely adjusts SVM model by modifying the optimization function, such as DEC [6], considering different margin between categories [43], FSVM [1], zSVM [44], entropy-based fuzzy SVM (EFSVM) [45], and cost-sensitive hinge loss (CSHL) [46]. Different from these SVMs where the weights or the error costs are manually set or searched by performing grid search, *ATEC* can obtain

a more preferable hyperplane by automatically tuning the error cost ratio in a very efficient manner.

Hyperparameters Optimization Methods. Though there are many algorithmic methods to improve the performance of SVM for imbalanced data, how to optimize the hyperparameters is an open problem. Grid search is the most common method which requires users to first manually select the search spaces of C and γ for the SVMs with RBF kernel and then measure each pair of them by some resampling procedures [47]. However, it is such an exhaustive search that many researchers propose other hyperparameters optimization methods. Bergstra and Bengio [49] propose a random search that improves grid search in high-dimensional spaces, but it doesn't work well for low-dimensional spaces (e.g., 1-d, 2-d). Keerthi et al. [48] propose a method to determine the optimal values of hyperparameters by computing the gradient of the error with respect to hyperparameters; X. Pan et al. [50] propose a method that safely accelerates the training process and applies to the parameter tuning process. There are also heuristic methods, such as evolutionary algorithms [51] and particle swarm optimization [52]. However, these methods only optimize the inherent hyperparameters of SVMs, i.e. C and γ. They are not capable of searching the best settings of error cost for between-class samples while our method *ATEC* can tune it automatically and efficiently.

7 Conclusion

To further improve existing cost-sensitive SVMs for imbalanced learning, this paper proposes *ATEC*, a solution that can efficiently find a more preferable position for the hyperplane learned by SVMs through automatically tuning the error cost of between-class samples. The core of *ATEC* is its modeling for the preferable hyperplane, where the maximum entropy within a certain area is used as the guideline for hyperplane search. Though *ATEC* requires two extra parameters to achieve its purpose, i.e., $count$ and $increment$, we have presented a theorem to prove that the accuracy is insensitive to these parameters when they exceed certain thresholds. Extensive experimental evaluation shows that *ATEC* can improve the performance of existing error-cost based SVMs such as DEC and FSVM-CIL in terms of the F1 value of the minority class, AUC-PR score and AUC-ROC score. Moreover, *ATEC* can achieve comparable performance with state-of-the-art imbalance methods, e.g., LexiBoost. More importantly, to train the SVM model that can achieve a relatively high F1 score, *ATEC* can be two orders of magnitude faster than the parameter tuning strategy based on grid searches. In our future work, we plan to extend the main idea of *ATEC* to support more SVMs as well as other sort of classifying algorithms such as logistic regression.

Acknowledgments

This research is supported in part by National Key R&D Program of China (2018YFB1402800) , and Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

References

[1] R. Batuwita and V. Palade, "Fsvm-cil: fuzzy support vector machines for class imbalance learning," *TFS*, vol. 18, no. 3, pp. 558–571, 2010.

[2] G. Wu and E. Y. Chang, "Class-boundary alignment for imbalanced dataset learning," in *ICML workshop on learning from imbalanced data sets II*, 2003, pp. 49–56.

[3] K. Thomas, C. Grier, D. Song, and V. Paxson, "Suspended accounts in retrospect: an analysis of twitter spam," in *IMC*, 2011, pp. 243–258.

[4] T. Verbraken, W. Verbeke, and B. Baesens, "A novel profit maximizing metric for measuring classification performance of customer churn prediction models," *TKDE*, vol. 25, no. 5, pp. 961–973, 2013.

[5] H. He and E. A. Garcia, "Learning from imbalanced data," *TKDE*, vol. 21, no. 9, pp. 1263–1284, 2009.

[6] K. Veropoulos, C. Campbell, N. Cristianini *et al.*, "Controlling the sensitivity of support vector machines," in *IJCAI*, 1999, pp. 55–60.

[7] A. Iranmehr, H. Masnadi-Shirazi, and N. Vasconcelos, "Cost-sensitive support vector machines," *Neurocomputing*, vol. 343, pp. 50 – 64, 2019.

[8] X.-Y. Liu and Z.-H. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," in *ICDM*, 2006, pp. 970–974.

[9] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 2951–2959.

[10] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., 2011, pp. 2546–2554.

[11] S. Datta, S. Nag, and S. Das, "Boosting with lexicographic programming: Addressing class imbalance without cost tuning," *TKDE*, pp. 1–1, 2019.

[12] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Cross validation through two-dimensional solution surface for cost-sensitive svm," *TPAMI*, vol. 39, no. 6, pp. 1103–1121, June 2017.

[13] C. E. Shannon, "A mathematical theory of communication," *MC2R*, vol. 5, no. 1, pp. 3–55, 2001.

[14] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra, "A maximum entropy approach to natural language processing," *Computational linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[15] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," in *ECML*, 2004, pp. 39–50.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[17] J. Shawe-Taylor and A. Dolia, "A framework for probability density estimation," in *Artificial Intelligence and Statistics*, 2007, pp. 468–475.

[18] M. Wasikowski and X.-w. Chen, "Combating the small sample class imbalance problem using feature selection," *TKDE*, vol. 22, no. 10, pp. 1388–1400, 2010.

[19] E. S. Xioufis, M. Spiliopoulou, G. Tsoumakas, and I. P. Vlahavas, "Dealing with concept drift and class imbalance in multi-label stream classification." in *IJCAI*, 2011, pp. 1583–1588.

[20] Y. Tang, Y. Zhang, N. V. Chawla, and S. Krasser, "Svms modeling for highly imbalanced classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 281–288, Feb 2009.

[21] B. Scholkopf, K.-K. Sung, C. J. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with gaussian kernels to radial basis function classifiers," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2758–2765, 1997.

[22] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *TIST*, vol. 2, pp. 27:1–27:27, 2011.

[23] D. D. Lewis, "Reuters-21578 text catego- rization test collection," 1987.

[24] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *ACL System Demonstrations*, 2014, pp. 55–60.

[25] Google, "Word2vec," 2013, accessed: April 7, 2018. [Online]. Available: https://code.google.com/archive/p/word2vec/

[26] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework," *MVLSC*, vol. 17, 2011.

[27] S. Z. L. M. F. W. Qi B., Jiang J., "A novel method for highly imbalanced classification with weighted support vector machine," *Knowledge Science, Engineering and Management*, 2019.
[28] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *TSMCB*, vol. 39, no. 2, pp. 539–550, 2009.
[29] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 40–49, 2004.
[30] L. Breiman, *Classification and regression trees*. Routledge, 2017.
[31] A. J. Wyner, M. Olson, J. Bleich, and D. Mease, "Explaining the success of adaboost and random forests as interpolating classifiers."
[32] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *ICML*, 2006, pp. 233–240.
[33] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *JMLR*, pp. 1–30, 2006.
[34] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of AI Research*, vol. 16, pp. 321–357, 2002.
[35] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning."
[36] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *IJCNN*, 2008, pp. 1322–1328.
[37] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *TKDE*, vol. 27, no. 5, pp. 1356–1368, 2015.
[38] J.-F. Díez-Pastor, J. J. Rodríguez, C. I. García-Osorio, and L. I. Kuncheva, "Diversity techniques improve the performance of the best imbalance learning ensembles," *Information Sciences*, vol. 325, pp. 98–117, 2015.
[39] W. Lee, C.-H. Jun, and J.-S. Lee, "Instance categorization by support vector machines to adjust weights in adaboost for imbalanced data classification," *Information Sciences*, vol. 381, pp. 92 – 103, 2017.
[40] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3573–3587, 2018.
[41] G.-A. L. . A. C. J. C. Nunez, H., "Improving svm classification on imbalanced datasets by introducing a new bias," *Journal of Classification*, vol. 34, pp. 427–443, October 2017.
[42] S. Datta and S. Das, "Multiobjective support vector machines: Handling class imbalance with pareto optimality," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1602–1608, May 2019.
[43] C.-Y. Yang, J.-S. Yang, and J.-J. Wang, "Margin calibration in svm class-imbalanced learning," *Neurocomputing*, vol. 73, no. 1, pp. 397 – 411, 2009.
[44] T. Imam, K. M. Ting, and J. Kamruzzaman, "z-svm: an svm for improved classification of imbalanced data," *Australian joint conference on artificial intelligence*, pp. 264–273, 2006.
[45] Q. Fan, Z. Wang, D. Li, D. Gao, and H. Zha, "Entropy-based fuzzy support vector machine for imbalanced datasets," *Knowledge-Based Systems*, vol. 115, pp. 87–99, 2017.
[46] A. Iranmehr, H. Masnadi-Shirazi, and N. Vasconcelos, "Cost-sensitive support vector machines," *Neurocomputing*, vol. 343, pp. 50 – 64, 2019.
[47] J. Wainer and G. C. Cawley, "Empirical evaluation of resampling procedures for optimising svm hyperparameters," *JMLR*, vol. 18, no. 15, pp. 1–35, 2017.
[48] S. S. Keerthi, V. Sindhwani, and O. Chapelle, "An efficient method for gradient-based adaptation of hyperparameters in svm models," in *Advances in neural information processing systems*, 2007, pp. 673–680.
[49] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *JMLR*, vol. 13, no. 1, pp. 281–305, 2012.
[50] X. Pan, Z. Yang, Y. Xu, and L. Wang, "Safe screening rules for accelerating twin support vector machine classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1876–1887, 2018.
[51] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple svm parameters," *Neurocomputing*, vol. 64, no. 1, pp. 107–117, 2005.
[52] S. Li and M. Tan, "Tuning svm parameters by using a hybrid clpso-bfgs algorithm," *Neurocomputing*, vol. 73, pp. 2089–2096, 2010.

Bin Cao received his Ph.D. degree in computer science from Zhejiang University, China in 2013. He then worked as a research associate in Hongkong University of Science and Technology and Noah's Ark Lab, Huawei. He joined Zhejiang University of Technology, Hangzhou, China in 2014, and is now an Associate Professor in the College of Computer Science. His research interests include spatio-temporal database and data mining.

Yuqi liu received his BS in computer science in Zhejiang University of Technology, Hangzhou, China, in 2016. He is now a PhD student of the Zhejiang University of Technology. His research interests are data mining and machine learning.

Chenyu Hou received his BS in software engineering in Zhejiang University of Technology, Hangzhou, China, in 2016. He is now a PhD student of the Zhejiang University of Technology. His research interests are database and data mining.

Jing Fan received her B.S., M.S. and Ph.D. degree in Computer Science from Zhejiang University, China in 1990, 1993 and 2003. She is now a Professor of School of Computer Science and Technology at Zhejiang University of Technology, China. She is a Director of China Computer Federation (CCF), and Chairman of Chapter Hangzhou of CCF. Her current research interest includes middleware, virtual reality and visualization.

Baihua Zheng received the PhD degree in computer science from Hong Kong University of Science & Technology, China, in 2003. She is currently an associate professor in the School of Information Systems, Singapore Management University, Singapore. Her research interests include mobile/pervasive computing, and spatial databases.

Jianwei Yin received his PhD degrees in computer science from Zhejiang University in 2001. He is currently a professor in the College of Computer Science at Zhejiang University. He is the visiting scholar of Georgia Institute of Technology, US, in 2008. His research interests include service computing and data management.