KBA: Kernel Boundary Alignment Considering Imbalanced Data Distribution

Gang Wu and Edward Y. Chang, *Senior Member*, *IEEE*

Abstract—An imbalanced training data set can pose serious problems for many real-world data mining tasks that employ SVMs to conduct supervised learning. In this paper, we propose a kernel-boundary-alignment algorithm, which considers THE training data imbalance as prior information to augment SVMs to improve class-prediction accuracy. Using a simple example, we first show that SVMs can suffer from high incidences of false negatives when the training instances of the target class are heavily outnumbered by the training instances of a nontarget class. The remedy we propose is to adjust the class boundary by modifying the kernel matrix, according to the imbalanced data distribution. Through theoretical analysis backed by empirical study, we show that our kernel-boundary-alignment algorithm works effectively on several data sets.

Index Terms—Imbalanced-data training, support vector machines, supervised classification.

1 Introduction

In many data mining tasks, finding rare objects or events is of primary interest [25]. Some examples include identifying fraudulent credit card transactions [11], diagnosing medical diseases, and recognizing suspicious activities in surveillance videos [28]. The task of finding rare objects or events is usually formulated as a supervised learning problem. Training instances are collected for both target and nontarget events and, then, a classifier is trained on the collected data to predict future instances. Researchers in the data mining community have been using Support Vector Machines (SVMs) as the learning algorithm since SVMs have strong theoretical foundations and excellent empirical successes in many pattern-recognition applications such as handwriting recognition [23], image retrieval [22], and text classification [14]. However, for rare-object detection and event mining, when the training instances of the target class are significantly outnumbered by the other training instances, the class-boundary learned by SVMs can be severely skewed toward the target class. As a result, the false-negative rate can be excessively high in identifying important target objects (e.g., a surveillance event or a disease-causing agent) and can result in catastrophic consequences.

Skewed class boundary is a subtle but serious problem that arises from using an SVM classifier—in fact, from using *any* classifier—for real-world problems with imbalanced training data. To understand the nature of the problem, let us consider it in a binary classification setting (positive versus negative). We know that the Bayesian framework estimates the posterior probability using the class conditional and the prior [12]. When the training data are highly imbalanced, the results naturally tend to favor the majority class. Hence, when ambiguity arises in classifying a particular sample because of similar class-conditional densities for the two classes, the Bayesian framework will rely on the large class prior favoring the majority class to break the tie. Consequently, the decision boundary will skew toward the minority class.

To illustrate this skew problem graphically, Fig. 1 shows a 2D checkerboard example. The checkerboard divides a 200×200 square into four quadrants. The top-left and bottom-right quadrants are occupied by negative (majority) instances, but the top-right and bottom-left quadrants contain only positive (minority) instances. The lines between the classes represent the "ideal" boundary that separates the two classes. In the rest of this paper, we will use *positive* when referring to minority instances and *negative* when referring to majority instances.

Fig. 2 exhibits the boundary distortion between the two left quadrants in the checkerboard under two different negative/positive training data ratios, where a black dot with a circle represents a support vector, and its radius represents the weight value α_i of the support vector. The bigger the circle, the larger the α_i. Fig. 2a shows the SVM class boundary when the ratio of the number of negative instances (in the quadrant above) to the number of positive instances (in the quadrant below) is $10:1$. Fig. 2b shows the boundary when the ratio increases to $10,000:1$. The boundary in Fig. 2b is much more skewed toward the positive quadrant than the boundary in Fig. 2a, thus causing a higher incidence of false negatives.

Although, in a theoretical sense, the Bayesian framework gives the optimal results (in terms of the smallest average error rate), we must be careful in applying it to real-world applications. In a real-world application such as security surveillance or disease diagnosis, the risk (or consequence) of mispredicting a positive event (a false negative) far outweighs that of mispredicting a negative event (a false positive). It is well-known that, in a binary classification problem, Bayesian risks are defined as:

• *The authors are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106. E-mail: gwu@engineering.ucsb.edu, echang@ece.ucsb.edu.*

Manuscript received 8 June 2004; revised 12 Nov. 2004; accepted 6 Dec. 2004; published online 20 Apr. 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0165-0604.

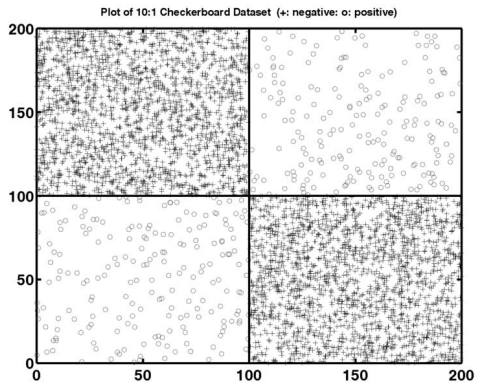

Fig. 1. Checkerboard experiment.

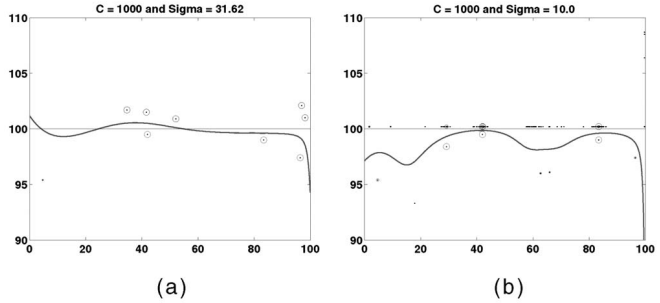

Fig. 2. Boundaries of different imbalanced ratios. We use a Gaussian RBF kernel for training. For better illustration, we zoom into the area around the ideal boundary (y = 100) between left two quadrants. Only support vectors are shown in the figures. (a) 10:1. (b) 10,000:1.

$$R(\alpha_p|\mathbf{x}) = \lambda_{pp}P(\omega_p|\mathbf{x}) + \lambda_{pn}P(\omega_n|\mathbf{x})$$
$$R(\alpha_n|\mathbf{x}) = \lambda_{np}P(\omega_p|\mathbf{x}) + \lambda_{nn}P(\omega_n|\mathbf{x}),$$

where p refers to the positive events and n to the negative, λ_{np} refers to the risk (or cost) of a false negative, and λ_{pn} refers to the risk of a false positive. The decisions about which action (α_p or α_n) to take—or which action has a smaller risk—are affected not just by the event likelihood (which directly influences the misclassification error), but also by the risk of mispredictions (λ_{np} and λ_{pn}).

How can we factor risk into SVMs to compensate for the effect caused by $P(\omega_n|\mathbf{x}) >> P(\omega_p|\mathbf{x})$? Examining the class prediction function of SVMs,

$$sgn\left(f(\mathbf{x}) = \sum_{i=1}^{n} y_i\alpha_i K(\mathbf{x}, \mathbf{x_i}) + b\right), \quad (1)$$

we see that three parameters can affect the decision outcome: b, α_i, and K. Our theoretical analysis, backed up by empirical study, will show that the only effective method for improving SVMs is through adaptively modifying K based on the training data distribution. To modify K, we propose in this paper the kernel-boundary-alignment (KBA) algorithm, which addresses the imbalanced training data problem in three complementary ways:

1. *Improving class separation*. KBA increases intraclass similarity and decreases interclass similarity through changing the similarity scores in the kernel matrix. Therefore, instances in the same class are better clustered in the feature space $\mathcal{F}$ away from those in the other classes.
2. *Safeguarding overfitting*. To avoid overfitting, KBA uses the existing support vectors to guide its boundary-alignment procedure.
3. *Improving imbalanced ratio*. By properly adjusting the similarity scores between majority instances, KBA can reduce the number of support vectors on the majority side and, hence, improve the imbalanced support-vector ratio.

Our experimental results on both UCI and real-world image/video data sets show the kernel-boundary-alignment algorithm to be effective in correcting a skewed boundary caused by imbalanced training data.

The rest of this paper is organized as follows: Section 2 discusses related work. In Section 3, we describe the kernel-boundary-alignment algorithm for addressing the imbalanced training-data problem. Section 4 presents the setup and the results of our empirical studies. We offer our concluding remarks in Section 5.

2 Related Work

Approaches for addressing the imbalanced training data problem can be divided into two main categories: the data processing approach and the algorithmic approach. The data processing approach can be further subdivided into two methods: undersample the majority class or oversample the minority class. The one-sided selection proposed by Kubat and Matwin [17] is a representative undersampling approach which removes noisy, borderline, and redundant majority training instances. However, these steps typically can remove only a small fraction of the majority instances, so they might not be very helpful in a scenario with a majority-to-minority ratio of more than $100:1$ (which is becoming common in many emerging pattern-recognition applications). Multiclassifier training [6] and Bagging [3] are two other undersampling methods. These methods do not deal with noisy and borderline data directly, but use a large ensemble of subclassifiers to reduce prediction variance.

Oversampling [7], [26] is the opposite of the undersampling approach. It duplicates or interpolates minority instances in the hope of reducing the imbalance. The oversampling approach can be considered as a "phantom-transduction" method. It assumes the neighborhood of a positive instance to be still positive and the instances between two positive instances positive. The validity of assumptions like these, however, can be data-dependent.

The algorithmic approach, which is traditionally[1] orthogonal to the data-processing approach, is the focus of this paper. Nugroho et al. [20] suggest combining a competitive learning network and a multilayer perceptron as a solution for the class imbalance problem. Cardie and Howe [5], [10], [17] modify the decision-tree generator to improve its learning performance on imbalanced data sets. For SVMs, few attempts [16], [19], [24], [29] have dealt with the imbalanced training-data problem. Basically, all of those

1. Although our algorithmic approach focuses on aligning the class boundary, it can effectively remove redundant majority instances as a by-product.

aim to incorporate into the SVMs the prior knowledge of the risk factors of false negatives and false positives. Karakoulas and Taylor [16] propose an approach to modify the bias (or parameter b) in the class prediction function (1). Lin et al. [19], [24], [29] use different predefined penalty constants (based on some prior knowledge) for different classes of data. The effectiveness of this method is limited since the Karush Kuhn Tucker (KKT) conditions [18] use the penalty constants as the upper bounds, rather than the lower bounds, of misclassification costs. Moreover, the KKT condition $\sum_{i=1}^{n} \alpha_i y_i = 0$ imposes an equal total influence from the positive and negative support vectors. The increases in some α_is at the positive side will inadvertently increase some α_is at the negative side to satisfy the constraint. These constraints can make the increase of C^+ on minority instances ineffective. (Validation is presented in Section 4.)

Another algorithmic approach to improve the SVMs for imbalanced training is to modify the employed kernel function K or kernel matrix[2] $\mathbf{K}$. In kernel-based methods, such as SVMs, the kernel K represents a pairwise similarity measurement among the data. Because of the central role of the kernel, a poor K will lead to a poor performance of the employed classifier [8], [21]. Our prior work ACT [27] falls into this category by modifying the K using (quasi) conformal transformation so as to change the spatial resolution around the class boundary. However, ACT works only when data have a fixed-dimensional vector-space representation since the algorithm relies on information in the input space. The kernel-boundary alignment algorithm (KBA) that we propose in this paper is a more general approach, which does not require the data to have a vector-space representation. This relaxation is important so that we can deal with a large class of sequence data (motion trajectories, DNA sequences, sensor-network data, etc.), which may have a different length. Furthermore, KBA provides greater flexibility in adjusting the class boundary. (We present details in Section 3.2).

Recently, several kernel alignment algorithms [8], [15] have been proposed in the Machine Learning community to learn a kernel function or a kernel matrix from the training data. The motivation behind these methods is that a good kernel should be data dependent and a systematic method for learning a good kernel from the data is useful. All these methods are based on the notion of the kernel target alignment proposed by Cristianini et al. [8]. The alignment score is used for measuring the quality of a given kernel matrix. To address the imbalanced training data problem, Kandola and Shawe-Taylor [15] propose an extension to kernel-target alignment by giving the alignment targets of $\frac{1}{n^+}$ to the positive instances and $-\frac{1}{n^-}$ to the negative instances. (We use n^+ and n^- to denote the number of minority and majority instances, respectively.) Unfortunately, when $\frac{n^+}{n^-}$ is small (when n^+ does not remain $O(n^+ + n^-)$), the concentration property upon which that kernel-target alignment relies may no longer hold. In other words, the proposed method can deal only with uneven data that are not very uneven. Our proposed KBA algorithm is based on maximizing the separation margin of the SVMs and is more effective in its solution.

2. Given a kernel function K and a set of instances $\mathcal{X}_{train} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$, the kernel matrix (Gram matrix) is the matrix of all possible inner-products of pairs from X_{train}, $\mathbf{K} = (k_{ij}) = K(\mathbf{x}_i, \mathbf{x}_j)$.

3 Kernel Boundary Alignment

Let us consider a two-class classification problem with training data set $\mathcal{X}_{train} = \{x_i, y_i\}_{i=1}^{n}$, where $x_i \in \Re^m$ and $y \in \{-1, +1\}$. The basic idea of kernel methods is to map $\mathcal{X}$ from its input space $\mathcal{I}$ to a feature space $\mathcal{F}$, where the data can be separated by applying a linear procedure [23]. The attractiveness of kernel methods is that the mapping from $\mathcal{I}$ to $\mathcal{F}$ can be performed efficiently through the inner product defined in $\mathcal{F}$, or $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$. Common choices for kernels are polynomial functions $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot x_j + 1)^p$ and Gaussian radial basis functions (RBF) $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}\right)$.

More generally, $K(\mathbf{x}_i, \mathbf{x}_j)$ can be considered as a similarity measure between instances $\mathbf{x}_i$ and $\mathbf{x}_j$. (Theoretical justifications are presented in [21].) For instance, when an Gaussian RBF function is employed, the value of $K(\mathbf{x}_i, \mathbf{x}_j)$ ranges from 0 to 1, where $K(\mathbf{x}_i, \mathbf{x}_j) = 0$ when x_i and x_j are infinitely far away (dissimilar) in input space and $K(\mathbf{x}_i, \mathbf{x}_j) = 1$ when $\mathbf{x}_i$ and $\mathbf{x}_j$ are infinitely close (almost identical). Thus, the choice of a good kernel is equivalent to the choice of a good distance function for measuring similarity.

To tackle the imbalanced training data set problem, we propose to modify the kernel by considering the imbalanced data distribution as the prior information. There are two approaches to modify the kernel. The first approach is to modify the kernel function K directly in input space $\mathcal{I}$. The second approach is to modify the kernel matrix $\mathbf{K}$ generated by a kernel function (for vector data) or a similarity measurement (for nonvector data) on the training set $\mathcal{X}$ in feature space $\mathcal{F}$. The first approach relies on the data information in $\mathcal{I}$ and, hence, the fixed-dimensional input space must exist. However, the second approach to modify the kernel matrix in $\mathcal{F}$ can bypass this limitation by only relying on the mapped data information in the feature space. Indeed, as long as the resulting kernel matrix $\mathbf{K}$ maintains the positive (semi) definite property, the modification is mathematically valid.

In the remainder of this section, we first summarize ACT [27], our prior function-modification approach, to set up the context for discussing KBA. (KBA must obey the theoretical justification on which ACT is explicitly founded.) We then propose the kernel-boundary-alignment (KBA) algorithm. This algorithm generalizes the work of ACT, by modifying the kernel matrix in $\mathcal{F}$, to deal with data that have a fixed-dimensional vector-space representation and also data that do not (e.g., sequence data). At the end of Section 3, we will discuss the differences between KBA and ACT and, in particular, the additional flexibility that KBA enjoys in adjusting similarity measures. Table 1 lists key notations used in this section.

3.1 Conformally Transforming Kernel K

Kernel-based methods, such as SVMs, introduce a mapping function Φ which embeds the $\mathcal{I}$ into a high-dimensional $\mathcal{F}$

TABLE 1
Notations Used in ACT and KBA

Symbol	Meaning
$D(\mathbf{x})$	Conformal transformation function
$\tau_k^2,\ \tau_b^2$	Parameters of $D(\mathbf{x})$
M	Nearest neighborhood range
$\lvert\mathbf{SI}\rvert$	Number of support instances
$\lvert\mathbf{SI}^+\rvert$	Number of minority support instances
$\lvert\mathbf{SI}^-\rvert$	Number of majority support instances
$\mathbf{x}^+,\ \Phi(\mathbf{x}^+)$	A minority support instance
$\mathbf{x}^-,\ \Phi(\mathbf{x}^-)$	A majority support instance
$\mathbf{x}_b,\ \Phi(\mathbf{x}_b)$	An interpolated boundary instance
α	Weight parameter of interpolation
$\mathcal{X}_{train}$	Set of training instances
$\mathcal{X}_b^*$	Sets of interpolated boundary instances
$\mathcal{X}_{mis}^+$	Set of misclassified minority test instances
$\mathcal{X}_{mis}^-$	Set of misclassified majority test instances

as a curved Riemannian manifold $\mathcal{S}$ where the mapped data reside [4]. A Riemannian metric $g_{ij}(\mathbf{x})$ is then defined for $\mathcal{S}$, which is associated with the kernel function $K(\mathbf{x}, \mathbf{x}')$ by

$$g_{ij}(\mathbf{x}) = \left(\frac{\partial^2 K(\mathbf{x}, \mathbf{x}')}{\partial x_i \partial x'_j}\right)_{\mathbf{x}'=\mathbf{x}}. \tag{2}$$

The metric g_{ij} shows how a local area around $\mathbf{x}$ in $\mathcal{I}$ is magnified in $\mathcal{F}$ under the mapping of Φ. The idea of conformal transformation in SVMs is to enlarge the margin by increasing the magnification factor $g_{ij}(\mathbf{x})$ along the boundary (represented by support vectors) and to decrease it around the other points. This could be implemented by a conformal transformation[3] of the related kernel $K(\mathbf{x}, \mathbf{x}')$, according to (2), so that the spatial relationship between the data would not be affected too much [1]. Such a (quasi) conformal transformation can be depicted as

$$\tilde{K}(\mathbf{x}, \mathbf{x}') = D(\mathbf{x})D(\mathbf{x}')K(\mathbf{x}, \mathbf{x}'). \tag{3}$$

In (3), $D(\mathbf{x})$ is a properly defined positive (quasi) conformal function. $D(\mathbf{x})$ should be chosen in such a way that the new Riemannian metric $\tilde{g}_{ij}(\mathbf{x})$, associated with the new kernel function $\tilde{K}(\mathbf{x}, \mathbf{x}')$, has larger values near the decision boundary. Furthermore, to deal with the skew of the class boundary caused by imbalanced classes, we magnify $\tilde{g}_{ij}(\mathbf{x})$ more in the boundary area close to the minority class. In [27], we demonstrate that an RBF distance function such as

$$D(\mathbf{x}) = \sum_{k \in \mathbf{SV}} \exp\left(-\frac{|\mathbf{x} - \mathbf{x}_k|}{\tau_k^2}\right) \tag{4}$$

is a good choice for $D(\mathbf{x})$.

In (4), we can see that if τ_k^2s are fixed (equal) for all support vectors $\mathbf{x}_k$s, $D(\mathbf{x})$ would be very dependent on the density of support vectors in the neighborhood of $\mathbf{x}$. To alleviate this problem, we adaptively tune τ_k^2 according to the spatial distribution of support vectors in $\mathcal{F}$. This goal can be achieved by the following equation:

$$\tau_k^2 = \mathrm{AVG}_{i \in \{\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_k)\|^2 < M,\ y_i \neq y_k\}} \left(\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_k)\|^2\right). \tag{5}$$

In the above equation, the average on the right-hand side comprises all support vectors in $\Phi(\mathbf{x}_k)$'s neighborhood within a radius of M but having a different class label. If we choose a large M, such as the maximum distance $\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_k)\|^2$, we might not be able to achieve the local spatial distribution of the support vectors in the neighborhood of $\Phi(\mathbf{x})$. On the contrary, if we choose a small M, we might not be able to find enough support vectors in $\Phi(\mathbf{x}_k)$'s neighborhood for density calculation. To alleviate this problem, ACT automatically calculates M as the average distance of support vectors that are nearest and farthest from $\Phi(\mathbf{x}_k)$. Setting τ_k^2 in this way takes into consideration the spatial distribution of the support vectors in $\mathcal{F}$. Moreover, since ACT aims to further increase the margin of SVMs, in (5), we only take into account the support vectors which have different class labels with $\Phi(\mathbf{x}_k)$ while computing τ_k^2. With this method, we could expect to achieve higher magnification around the margin area, compared to the method of counting the support vectors without the constraint $y_i \neq y_k$.

Although the mapping Φ is unknown, we can use the kernel trick to calculate the distance in $\mathcal{F}$:

$$\begin{aligned} &\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_k)\|^2 \\ &= K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_k, \mathbf{x}_k) - 2 * K(\mathbf{x}_i, \mathbf{x}_k). \end{aligned} \tag{6}$$

Substituting (6) into (5), we can then calculate the τ_k^2 for each support vector, which can adaptively reflect the spatial distribution of the support vector in $\mathcal{F}$, not in $\mathcal{I}$.

When the training data set is very imbalanced, the class boundary tends to be skewed toward the minority class in the input space $\mathcal{I}$. We hope that the new metric $\tilde{g}_{ij}(\mathbf{x})$ would further magnify the area far away from a minority support vector $\mathbf{x}_i$ so that the boundary imbalance could be alleviated. Our algorithm thus assigns a multiplier for the τ_k^2 in (5) to reflect the boundary skew in $D(\mathbf{x})$. We tune $\tilde{\tau}_k^2$ as $\eta_p \tau_k^2$ if $\mathbf{x}_k$ is a minority support vector; otherwise, we tune it as $\eta_n \tau_k^2$. Examining (4), we can see that $D(\mathbf{x})$ is a monotonically increasing function of τ_k^2. To increase the metric $\tilde{g}_{ij}(x)$ in an area which is not very close to the support vector $\mathbf{x}_k$, it would be better to choose a larger η_p for the τ_k^2 of a minority support vector. For a majority support vector, we can choose a smaller η_n, so as to minimize influence on the class boundary. We empirically demonstrate that η_p and η_n are proportional to the skew of support vectors, or η_p as $O\left(\frac{|\mathbf{SV}^-|}{|\mathbf{SV}^+|}\right)$, and η_n as $O\left(\frac{|\mathbf{SV}^+|}{|\mathbf{SV}^-|}\right)$, where $|\mathbf{SV}^+|$ and $|\mathbf{SV}^-|$ denote the number of minority and majority support vectors, respectively. (Please refer to [27] for more details on the theoretical justification of ACT.)

3.2 Modifying Kernel Matrix $\mathbf{K}$

For data that do not have a fixed-dimensional vector-space representation (e.g., sequence data), it may not be feasible to

3. Usually, it is difficult to find a totally-conformal mapping function to transform the kernel. As suggested in [1], we can choose a quasi-conformal mapping function for kernel transformation.

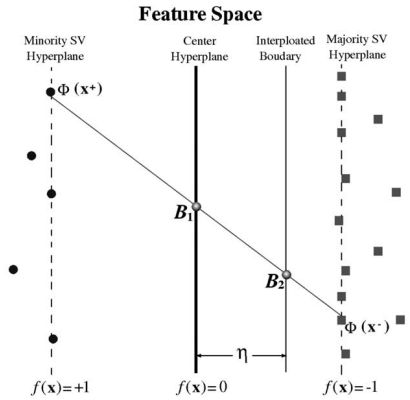

Fig. 3. Estimate boundary instances in $\mathcal{F}$.

transform kernel function K conformally directly in a vector space. In this situation, KBA modifies kernel matrix K based on the training-data distribution in $\mathcal{F}$. Kernel matrix $\mathbf{K}$ encodes all pairwise-similarity information between the instances in the training data set. Hence, modifying the kernel matrix transforms the kernel function indirectly. (Notice that KBA is certainly applicable to data that *do* have a vector-space representation, since $\mathbf{K} = (k_{\mathbf{xx}'} = K(\mathbf{x}, \mathbf{x}'))$.) Now, because a training instance $\mathbf{x}$ might not be a vector, we introduce a more general term, *support instance*,[4] to denote $\mathbf{x}$ if its embedded point via $\mathbf{K}$ is a support vector in $\mathcal{F}$.

In the following sections, we will first propose a data-dependent way to estimate the "ideal" class boundary in $\mathcal{F}$ (Section 3.2.1). We then choose a feasible conformal function $D(\mathbf{x})$, which can assign a larger spatial resolution along the estimated "ideal" boundary in $\mathcal{F}$ (Section 3.2.2). Finally, we present KBA's iterative training procedure (Section 3.2.3).

3.2.1 Estimation of Boundary

Performing transformation on K or $\mathbf{K}$ aims to magnify the spatial resolution along the decision boundary, thereby improving the class separation. According to the work of [1], [27], maximal magnification should be performed along the class boundary. Unfortunately, locating the class boundary in input space $\mathcal{I}$ is difficult [1]. (When the data do not have a fixed-dimensional vector-space representation, locating the class boundary in $\mathcal{I}$ is impossible.) Instead, KBA locates the class boundary in feature space $\mathcal{F}$ through interpolation. In $\mathcal{F}$, the class boundary learned from the training data is the center hyperplane in the margin. When the training data set is balanced, the center hyperplane approximates the "ideal" boundary well. However, when the training data set is imbalanced, the decision boundary is skewed toward the minority class. To compensate for this skew, KBA gives the maximal magnification to an interpolated boundary between the center hyperplane and the hyperplane formed by the majority support instances in $\mathcal{F}$.

Fig. 3 illustrates how the interpolation procedure works. Let $\Phi(\mathbf{x}^+)$ and $\Phi(\mathbf{x}^-)$ denote a minority support instance and a majority support instance, respectively. A boundary instance $\Phi(\mathbf{x}_b)$ on the "ideal" boundary should reside between the center hyperplane (the thick line in the middle of Fig. 3) and the majority support-instance hyperplane (the dash line on the right-hand side of the figure). We can thus estimate the location of $\Phi(\mathbf{x}_b)$ by interpolating the positions of $\Phi(\mathbf{x}^+)$ and $\Phi(x^-)$ as follows:

$$\Phi(\mathbf{x}_b) = (1-\beta)\Phi(\mathbf{x}^+) + \beta\Phi(\mathbf{x}^-),\ \frac{1}{2} \le \beta \le 1. \tag{7}$$

When the training data set is balanced, β is $\frac{1}{2}$ and $\Phi(\mathbf{x}_b)$ lies on the center hyperplane (e.g., point B_1 in the figure). In this balanced case, the estimated "ideal" boundary coincides with the learned boundary. When the training data set is imbalanced, however, we need to adjust β to estimate the "ideal" boundary. The key research question to answer is: "How to determine β in a data-dependent way?"

We propose a cost function to measure the loss caused by false negatives and false positives when different values of β are introduced. We then choose the β which can achieve the minimal cost. Let $\mathcal{X}^+_{mis}$ denote the set of the misclassified minority test-instances and $\mathcal{X}^-_{mis}$ the set of the misclassified majority test-instances. We define the cost functional $C(\cdot)$ for any scalar decreasing loss functions $c_p(\cdot)$ and $c_n(\cdot)$ as follows:

$$\begin{aligned} C(\eta) &= \sum_{i=1}^{|\mathcal{X}^+_{mis}|} c_p(y_i f'(\mathbf{x}_i)) + \sum_{i=1}^{|\mathcal{X}^-_{mis}|} c_n(y_i f'(\mathbf{x}_i)), \\ &where\ f'(\mathbf{x}_i) = f(\mathbf{x}_i) + \eta,\ 0 \le \eta \le 1. \end{aligned} \tag{8}$$

In the equation above, $f(\mathbf{x}_i)$ is the SVM predication score for test instance $\mathbf{x}_i$, η is the offset of the interpolated boundary from the center hyperplane, as shown in Fig. 3, and $y_i f'(\mathbf{x}_i)$ is the associated margin in $\mathcal{F}$ for instance $\mathbf{x}_i$ with respect to the interpolated class boundary. The loss functions $c_p(\cdot)$ and $c_n(\cdot)$ are used to penalize the misclassified[5] minorities (false negative) and majorities (false positive), respectively. Each loss function, $c_p(\cdot)$ or $c_n(\cdot)$, can be chosen as any scalar decreasing function of the margin $y_i f'(\mathbf{x}_i)$ according to the prior knowledge. When no prior knowledge is available, usually, we can choose the exponential loss function as $c_p(\cdot)$ and the log-likelihood loss function as $c_n(\cdot)$, i.e,

$$\begin{aligned} c_p(y_i f'(\mathbf{x}_i)) &= \exp(-y_i f'(\mathbf{x}_i)), \\ c_n(y_i f'(\mathbf{x}_i)) &= \ln(1 + \exp(-y_i f'(\mathbf{x}_i))). \end{aligned}$$

The justification of choosing them as the loss functions comes from boosting [13], where the exponential loss criterion concentrates much more influence (exponentially) on observations with large negative margins ($y_i f'(\mathbf{x}_i) < 0$), and the log-likelihood loss concentrates relatively less influence (linearly) on such observations. Since KBA aims to concentrate on false negatives, we use the exponential loss as $c_p(\cdot)$ and the log-likelihood loss as $c_n(\cdot)$. Notice that, since both exponential and log-likelihood loss functions are convex [13], our cost formulation in (8) is also convex with respect to η.

The optimal η^* is then chosen by minimizing the total loss induced by all test instances falling into the margin of SVMs,

$$\eta^* = \arg\min_{\eta} C(\eta),\ \ 0 \le \eta \le 1.$$

The optimal η^* can be calculated from $\frac{\partial C(\eta)}{\partial \eta} = 0$ and truncated between 0 and 1. The above optimization

4. In the KBA algorithm, if $\mathbf{x}$ is a support instance, we call both $\mathbf{x}$ and its embedded support vector via $\mathbf{K}$ in $\mathcal{F}$ *support instance*.

5. In KBA, we only consider the misclassified test instances among the margin so as to reduce the influence from the outliers. Their SVM scores $f(\mathbf{x})$ range from -1 to $+1$.

procedure involves with only one unknown variable η. It can thus be efficiently solved using many numerical analysis methods such as the conjugate gradient algorithm.

After the optimal position η^* of the interpolated boundary is calculated, we can obtain β in (7) as follows:

$$\beta = \frac{1+\eta^*}{2}.$$

3.2.2 Selection of $\mathbf{D}(\mathbf{x})$

After interpolating a boundary in the margin, we then magnify the spatial resolution along the boundary by modifying the Riemannian metric $g_{ij}(\mathbf{x})$ according to (2) and (3). When given a prior kernel, $g_{ij}(\mathbf{x})$ is determined by the conformal function $D(\mathbf{x})$. As what we discussed in Section 3.1, a good $D(\mathbf{x})$ function should be larger when $\mathbf{x}$ is closer to the boundary in $\mathcal{F}$ so as to achieve a larger spatial resolution around the boundary. According to this criteria, we choose $D(\mathbf{x})$ as a set of Gaussian functions:

$$D(\mathbf{x}) = \frac{1}{|\mathcal{X}_b^*|}\sum_{\mathbf{x}_b \in \mathcal{X}_b^*} \exp\left(-\frac{\|\Phi(\mathbf{x}) - \Phi(\mathbf{x}_b)\|^2}{\tau_b^2}\right), \tag{9}$$

where τ_b^2 is a parameter controlling the magnitude of each exponential function in $D(\mathbf{x})$. For a given instance $\mathbf{x}$, $D(\mathbf{x})$ is calculated as the average of all exponential functions, each of which is related with one interpolated boundary instance $\Phi(\mathbf{x}_b)$ in $\mathcal{X}_b^*$ set. In addition, $\|\Phi(\mathbf{x}) - \Phi(\mathbf{x}_b)\|^2$ is calculated via the kernel trick as follows:

$$\begin{aligned} &\|\Phi(\mathbf{x}) - \Phi(\mathbf{x}_b)\|^2 \\ &= \|\Phi(\mathbf{x}) - (1-\beta)\Phi(\mathbf{x}^+) - \beta\Phi(\mathbf{x}^-)\|^2 \\ &= k_{\mathbf{xx}} + (1-\beta)^2 k_{\mathbf{x}^+\mathbf{x}^+} + \beta^2 k_{\mathbf{x}^-\mathbf{x}^-} - 2(1-\beta)k_{\mathbf{xx}^+} \\ &\quad - 2\beta k_{\mathbf{xx}^-} + 2\beta(1-\beta)k_{\mathbf{x}^+\mathbf{x}^-}, \end{aligned} \tag{10}$$

where $k_{\mathbf{xx}'}$ is from the kernel matrix $\mathbf{K}$. When the instance $\mathbf{x}$ is an unseen test instance, $k_{\mathbf{xx}'}$ is computed using the predefined similarity measurement which generates the kernel matrix $\mathbf{K}$.

According to [1], [21], we have the following corollary to guarantee the kernel transformation induced by $D(\mathbf{x})$, as defined in (9), performs a mathematically valid conformal transformation.

Corollary 1. *The function $D(\mathbf{x})$ defined in (9) gives a valid conformal transformation on feature space $\mathcal{F}$ induced by the predefined kernel matrix $\mathbf{K}$.*

Proof. Please see Appendix A. □

In KBA, we adaptively choose τ_b^2 in a data-dependent way as

$$\tau_b^2 = \mathrm{AVG}_{i \in \{Dist^2(\mathbf{x}_i, \mathbf{x}_b) < M\}}\big(Dist^2(\mathbf{x}_i, \mathbf{x}_b)\big), \tag{11}$$

where the neighborhood range M is a constant. We choose the threshold M as the margin[6] value of SVMs. The distance $Dist^2(\mathbf{x}_i, \mathbf{x}_b)$ between two interpolated boundary instances $\mathbf{x}_i$ and $\mathbf{x}_b$ is $\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_b)\|^2$ and can be computed using (7) and (10). Notice that we do not need to scale τ_b^2 as in Section 3.1 for dealing with the imbalanced training-data problem, since we have considered this factor when interpolating the class boundary and selecting $D(\mathbf{x})$. Compared to (5) in ACT, (11) does not include the constraint $y_i \neq y_b$ since the interpolated boundary instance $\Phi(\mathbf{x}_b)$ does not have a label attribute.

We believe that our adjusted interpolation procedure and selection of $D(\mathbf{x})$ enjoy two benefits:

1. *Improved class-prediction accuracy*. In the imbalanced situation, most of misclassified minority instances fall into the margin area between the center hyperplane and the majority support-vector hyperplane. By maximizing the spatial resolution in this area, we expect to move those ambiguous instances as far away from the decision boundary as possible, so as to improve class-prediction accuracy.
2. *Improved imbalance ratio*. Since the majority support instances are located nearer the interpolated boundary than the minority support instances ($\frac{1}{2} \leq \beta \leq 1$ in (7)), by choosing a proper form of $D(\mathbf{x})$ as in (9), we can increase the degree of similarity between majority support instances and make them close each other in feature space after kernel transformation. This increase can lead to a reduction of the number of majority support instances and, hence, improve the imbalanced support-instance ratio.

3.2.3 Retraining

After choosing $D(\mathbf{x})$, KBA modifies the given kernel matrix $\mathbf{K} = (k_{ij})$ in the following way:

$$\tilde{k}_{ij} = D(\mathbf{x}_i) \times D(\mathbf{x}_j) \times k_{ij}. \tag{12}$$

The new kernel matrix $\tilde{\mathbf{K}}$ after modification is then put back into the regular SVMs algorithm for retraining. We have the following corollary, supported by the work of [21], to guarantee that the new kernel matrix after transformation in (12) is a valid kernel matrix.

Corollary 2. *When given a positive (semi) definite kernel matrix $\mathbf{K}$, the kernel transformation defined in (12) results in a new kernel matrix $\tilde{\mathbf{K}}$ which is also positive (semi) definite.*

Proof. Please see Appendix B. □

Fig. 4 summarizes the KBA algorithm. We apply KBA on the training data set X_{train} for several iterations or until the imbalanced support-instance ratio cannot be further decreased. In each iteration, KBA adaptively calculates τ_b^2 for each interpolated boundary instance (steps 8 to 10), based on the distribution in $\mathcal{F}$. Then, KBA updates the training data set X_{train} using the support-instance set (step 11). Why do we use the support-instance set as the training data in the next iteration? We do so because the decision boundary of SVMs will not change if we just use the support-instance set for retraining [23]. One benefit of doing so is that we can reduce the computational cost of training and we can also reduce the ratio of the majority-over-minority support-instances. Finally, KBA updates the kernel matrix and performs retraining on X_{train} (steps 16 to 18).

6. In ACT, we use the locations of support vectors to approximate the decision boundary. Empirically, we found that selecting different Ms for different support vectors works better than using a fixed M, though it incurs higher computational cost. In KBA, we approximate the "ideal" boundary by a set of interpolated boundary instances $\mathbf{x}_b$s. Since $\mathbf{x}_b$s are already located on the decision boundary, our empirical study showed that KBA is not very sensitive to M. We thus fix M as the margin value of SVMs in KBA.

Input:
X_{train}, $\mathbf{K}$;
θ; /* *stopping threshold* */
T; /* *maximum running iterations* */

Output:
$\mathcal{C}$; /* *output classifier* */

Variables:
$\mathbf{SI}$; /* *support-instance set* */
$\mathcal{X}_b$; /* *interpolated boundary-instance set* */
$\mathcal{X}_b^*$; /* *subset of* $\mathcal{X}_b$ */
M; /* *neighborhood range* */
$\mathbf{x}$; /* *a training instance* */
$\mathbf{s}$; /* *an interpolated boundary instance* */
$\mathbf{s}.\tau$; /* *parameter of* $\mathbf{s}$ */
$\varepsilon_{old}, \varepsilon_{new}$; /* *support-vector ratios* */

Function Calls:
SVMTrain(X_{train}, $\mathbf{K}$); /* *train classifier* $\mathcal{C}$ */
ExtractSI($\mathcal{C}$); /* *obtain* $\mathbf{SI}$ *from* $\mathcal{C}$ */
InterpolateBI($\mathbf{SI}$); /* *compute interpolated* $\mathcal{X}_b$ */
ComputeM($\mathbf{s}, \mathcal{X}_b$); /* *compute M* */
ExtractBI($\mathbf{x}$, $\mathcal{X}_b$, M); /* *sample* $\mathcal{X}_b$ *in range* M *of* $\mathbf{x}$ */

Begin
1) $\mathcal{C} \leftarrow$ SVMTrain($X_{train}, \mathbf{K}$);
2) $\varepsilon_{old} \leftarrow \infty$;
3) $\varepsilon_{new} \leftarrow \frac{|\mathbf{SI}^-|}{|\mathbf{SI}^+|}$;
4) $t \leftarrow 0$;
5) **while** $((\varepsilon_{old} - \varepsilon_{new} > \theta) \&\& (t < T))$ {
6) $\quad \mathbf{SI} \leftarrow$ ExtractSI($\mathcal{C}$);
7) $\quad \mathcal{X}_b \leftarrow$InterpolateBI($\mathbf{SI}$);
8) $\quad$ **for** each $\mathbf{s} \in \mathcal{X}_b$ {
9) $\quad\quad$ M $\leftarrow$ComputeM($\mathbf{s}, \mathcal{X}_b$);
10) $\quad\quad \mathbf{s}.\tau \leftarrow \sqrt{\text{AVG}_{i \in \{Dist^2(\mathbf{s}_i, \mathbf{s}) < \text{M}\}} \left(Dist^2(\mathbf{s}_i, \mathbf{s})\right)}$;}
11) $\quad X_{train} \leftarrow \mathbf{SI}$;
12) $\quad$ **for** each $\mathbf{x} \in X_{train}$ {
13) $\quad\quad$ M $\leftarrow$ComputeM($\mathbf{x}, \mathcal{X}_b$);
14) $\quad\quad \mathcal{X}_b^* \leftarrow$ ExtractBI($\mathbf{x}$, M, $\mathcal{X}_b$);
15) $\quad\quad D(\mathbf{x}) = \frac{1}{|\mathcal{X}_b^*|} \sum_{\mathbf{s} \in \mathcal{X}_b^*} \left(\exp\left(- \frac{\|\Phi(\mathbf{x}) - \Phi(\mathbf{s})\|^2}{\mathbf{s}.\tau^2} \right) \right)$; }
16) $\quad$ **for** each k_{ij} in $\mathbf{K}$ {
17) $\quad\quad k_{ij} \leftarrow D(\mathbf{x}_i) \times D(\mathbf{x}_j) \times k_{ij}$;}
18) $\quad \mathcal{C} \leftarrow$ SVMTrain($X_{train}, \mathbf{K}$);
19) $\quad \varepsilon_{old} \leftarrow \varepsilon_{new}$;
20) $\quad \varepsilon_{new} \leftarrow \frac{|\mathbf{SI}^-|}{|\mathbf{SI}^+|}$;
21) $\quad t \leftarrow t + 1$; }
22) **return** $\mathcal{C}$;
End

Fig. 4. The KBA algorithm.

4 Experimental Results

Our empirical study examined the effectiveness of the kernel-boundary-alignment algorithm in two aspects:

1. *Vector-space evaluation*. We compared KBA with other algorithms for imbalanced-data learning. We used six UCI data sets and an image data set to conduct this evaluation. (We present the data sets shortly.)
2. *Nonvector-space evaluation*. We evaluated the effectiveness of KBA on a set of video surveillance data, which are represented as spatio-temporal sequences that do not have a vector-space representation.

In our experiments, we used C-SVMs as our yardstick to measure how other methods perform. We employed Laplacian kernels of the form $\exp(-\gamma|\mathbf{x} - \mathbf{x}'|)$ as $K(\mathbf{x}, \mathbf{x}')$ of C-SVMs. Then, we used the following procedure: The data set was randomly split into training and test subsets generated in a certain ratio which was empirically chosen to be optimal on each data set for the regular C-SVMs. Hyperparameters (C and γ) of $K(\mathbf{x}, \mathbf{x}')$ were obtained for each run using 7-fold crossvalidation. All training, validation, and test subsets were sampled in a stratified manner ensuring each of them had the same negative/positive ratio [17]. We repeated this procedure seven times, computed average class-prediction accuracy, and compared the results. For ACT and KBA, we chose the maximum running iterations T as 5. The detailed choices of parameters are presented in Sections 4.1.1 and 4.1.2.

4.1 Vector-Space Evaluation

For this evaluation, we used six UCI data sets and a 116-category image data set. The six UCI data sets we experimented with are *abalone* (19), *car* (3), *segmentation* (1), *yeast* (5), *glass* (7), and *euthyroid* (1). The class-label in the parentheses indicates the target class we chose. Table 2 shows the characteristics of these six data sets organized according to their negative-to-positive training-instance ratios. The top three data sets (segmentation, glass, and euthyroid) are not-too-imbalanced. The middle two (car and yeast) are mildly imbalanced. The bottom data set (abalone) is the most imbalanced (the ratio is about $130:1$).

The image data set contains 20K images in 116 categories collected from the Corel Image CDs.[7] Each image is represented by a vector of 144 dimensions including color, texture, and shape features [22]. To perform class prediction, we employed the one-per-class (OPC) ensemble [9], which trains 116 classifiers, each of which predicts the class membership for one class. The class prediction on a testing instance is decided by voting among the 116 classifiers.

4.1.1 Results on UCI Benchmark Data Sets

Tables 2 and 3 report the experimental results with the six UCI data sets. In addition to conducting experiments with SVMs, ACT, and KBA, we also implemented and tested one popular minority-oversampling strategy SMOTE [7]. We used the L_2-norm RBF function for $D(\mathbf{x})$ in ACT. In each run, the training and test subsets were generated in the ratio $6:1$. For SMOTE,[8] the minority class was oversampled at 200 percent, 400 percent, and 1,000 percent for each of three groups of UCI data sets in Table 2, respectively.

We report in Table 2 using the Kubat's g-means metric defined as $\sqrt{a^+ \cdot a^-}$, where a^+ and a^- are positive (the

7. We exclude from our testbed those categories that cannot be classified automatically, such as "industry," "Rome," and "Boston." (for example, the Boston category contains various subjects, e.g., architectures, landscapes, and people, of Boston.)

8. For the data sets in Table 2 from top to bottom, for SMOTE, the optimal γ was 0.002, 0.003, 0.085, 0.3, 0.5, and 0.084, respectively. For SVMs, ACT, and KBA, the optimal γ was 0.004, 0.003, 0.08, 0.3, 0.5, and 0.086, respectively. All optimal Cs were $1,000$.

TABLE 2
Mean and Standard Deviation of *g*-Means Prediction Accuracy on UCI Data Sets

DATASET	# ATTRIB	# POS	# NEG	$\frac{\lvert SI^- \rvert}{\lvert SI^+ \rvert}$	SVMs	SMOTE	ACT	KBA
SEGMENTATION	19	30	180	1.8:1	$\mathbf{98.1 \pm 5.1}$	$\mathbf{98.1 \pm 5.1}$	$\mathbf{98.1 \pm 5.1}$	$\mathbf{98.1 \pm 5.1}$
GLASS	10	29	185	2.0:1	89.9 ± 6.3	91.8 ± 6.5	93.7 ± 6.7	$\mathbf{93.7 \pm 6.6}$
EUTHYROID	24	238	1762	1.5:1	92.8 ± 3.6	92.4 ± 4.3	94.5 ± 3.0	$\mathbf{94.6 \pm 2.9}$
CAR	6	69	1659	1.8:1	99.0 ± 2.2	99.0 ± 2.3	$\mathbf{99.9 \pm 0.2}$	$\mathbf{99.9 \pm 0.2}$
YEAST	8	51	1433	3.0:1	59.0 ± 12.1	69.9 ± 10.0	78.5 ± 4.5	$\mathbf{82.2 \pm 7.1}$
ABALONE	8	32	4145	9.0:1	0.0 ± 0.0	0.0 ± 0.0	51.9 ± 7.6	$\mathbf{57.8 \pm 5.4}$

target class) and negative testing accuracy, respectively [17]. Means and standard deviations of the experimental results are both reported in the table. In all of the six data sets, KBA achieves the highest or ties for the highest accuracy. (The best results are marked in bold.) When the data is very imbalanced (the last row abalone of Table 2), both SVMs and SMOTE cannot make accurate predictions. KBA achieves 57.8 percent mean class-prediction accuracy (in *g*-means) and shows 5.9 percentile points improvement over ACT.

We also report in Table 3 using AUC [2] defined as the area under an ROC curve to compare the four strategies on the six UCI data sets. Means and standard deviations of the AUC scores are reported in the table. For readability, we report AUCs as percentages between 0 percent and 100 percent, instead of between 0 and 1. Again, KBA achieves the highest mean AUCs in all six UCI data sets. Compared to ACT, KBA generated better results especially for the last data sets (yeast and abalone), with 1.4 and 7.2 percentile points improvement, respectively. Such gains bear out the flexibility and superiority of KBA working in feature space $\mathcal{F}$. Statistically, the higher AUCs from KBA means that our KBA algorithm will favor in classifying a positive (target) instance with a higher probability than other algorithms and, hence, could well tackle the imbalanced training data set problem.

Finally, we report in Table 4 the total training time of each method. The time is reported in seconds by averaging seven runs of training on different subsets of the training data. All experiments were done on a Pentium III 1GHZ workstation with 1GB DRAM. Compared to $SVMs$ and $SMOTE$, both ACT and KBA took longer time to train. This was because some computational costs were spent on modifying the kernel of SVMs in a data-dependent way to deal with the imbalanced-training problem. However, we can see that, for ACT, the training time increases only linearly compared to $SVMs$. For *euthyroid* and *abalone*, which have the largest number of training instances among the six UCI data sets, ACT's training time is about 12 and 9 times longer than that of SVMs. For all six data sets, the average increase in training time is about seven times. In addition, compared to ACT, KBA takes shorter time to train. For the six UCI data sets, KBA's average training time is 16.1 percent shorter than ACTs. This is expected, since KBA only used the support-instance set from the last iteration as the new training set in the current iteration, as described in Section 3.2.

4.1.2 Results on 20K Image Data Set

The image data set is more imbalanced than the UCI data sets. We first set aside $4K$ images to be used as the test subset; the remaining $16K$ images were used for training and validation. We compared five schemes: SVMs, BM (the boundary movement method by changing the parameter b in *C*-SVMs), BP (the biased penalty method of assigning different C to penalize different class in *C*-SVMs), ACT, and KBA. (The details of BM and BP have been presented in Section 2.) Notice that, in this experiment, we used the L_1-norm RBF function for $D(\mathbf{x})$ in ACT, since the L_1-norm RBF works best for the image data set [22].

Table 5 presents the prediction accuracy for 12 representative categories out of 116, sorted by their imbalance ratios. KBA improves the accuracy over SVMs by 5.3, 5.9, and 15.5 percentile points on the three subgroup data sets, respectively. KBA achieves the best prediction accuracy for seven out of 12 categories among all schemes (marked by bold font). BM is inferior to SVMs for almost all categories. Finally, BP outperforms SVMs, but only slightly. (We have predicted BP's ineffectiveness, due to the KKT conditions, in Section 2.)

Remark. From Table 5, we can see that, on this challenging data set of several diversified classes, the results of all algorithms, including KBA, are not stellar (class-predication accuracy is less than 50 percent for almost all classes). This low accuracy is caused partly by a large number of classes (116), and partly by not-so-perfect image-feature extraction. Nevertheless, a 50 percent prediction accuracy is far better than that of a random predication, which is $1/116 = 0.86$ percent.

TABLE 3
Mean and Standard Deviation of AUCs (in %) on UCI Data Sets

DATASET	SVMs	SMOTE	ACT	KBA
SEGMENTATION	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
GLASS	96.9 ± 3.0	97.1 ± 3.1	98.5 ± 2.5	$\mathbf{98.9 \pm 2.6}$
EUTHYROID	96.6 ± 2.2	96.0 ± 2.8	98.2 ± 1.8	$\mathbf{98.8 \pm 1.5}$
CAR	99.8 ± 0.2	99.8 ± 0.2	$\mathbf{99.9 \pm 0.1}$	$\mathbf{99.9 \pm 0.1}$
YEAST	89.2 ± 5.4	91.1 ± 5.0	93.8 ± 2.2	$\mathbf{95.2 \pm 2.5}$
ABALONE	62.5 ± 12.1	62.5 ± 12.1	80.2 ± 7.1	$\mathbf{87.4 \pm 6.8}$

TABLE 4
Training Time (in second) on UCI Data Sets

DATASET	SVMs	SMOTE	ACT	KBA
SEGMENTATION	1.0	1.7	4.7	4.3
GLASS	1.1	1.6	4.7	4.3
EUTHYROID	1.7	4.7	22.1	17.1
CAR	1.3	1.4	7.9	6.4
YEAST	1.1	1.4	7.4	5.7
ABALONE	1.5	3.6	15.6	11.9

TABLE 5
Image Data Set Prediction Accuracy

Category	Ratio	SVMs	BM	BP	ACT	KBA
MOUNTAIN	34 : 1	24.8	21.2	24.8	33.3	**34.5**
SNOW	37 : 1	46.4	47.5	47.8	**54.6**	52.3
DESERT	39 : 1	33.7	31.8	34.3	**39.1**	36.8
DOG	44 : 1	32.9	28.5	35.2	41.5	**42.7**
WOMAN	54 : 1	27.9	25.3	26.2	35.3	**39.1**
CHURCH	66 : 1	**21.8**	19.4	**21.8**	20.0	20.6
LEAF	80 : 1	26.1	27.2	24.8	32.6	**37.2**
LIZARD	101 : 1	13.9	11.8	15.1	22.2	**25.4**
PARROT	263 : 1	7.1	3.5	7.1	14.3	**18.4**
HORSE	264 : 1	14.3	10.4	14.3	28.6	**32.9**
LEOPARD	283 : 1	7.7	5.6	7.7	**23.1**	**23.1**
SHARK	1232 : 1	0.0	0.0	0.0	**16.6**	**16.6**

4.2 Nonvector-Space Evaluation

For our multicamera video-surveillance project, we recorded video data at a campus parking lot. We collected trajectories depicting five motion patterns: *circling* (30 instances), *zigzag-pattern* or *M-pattern* (22 instances), *back-forth* (40 instances), *go-straight* (200 instances), and *parking* (3,161 instances). We divided these events into benign and suspicious categories and aimed to detect suspicious events with high accuracy. The benign-event category consists of patterns *go-straight* and *parking*, and the suspicious-event category consists of the other three patterns.

For each experiment, we chose 60 percent of the data as the training set, keeping the remaining 40 percent to use as our testing data. We employed a sequence-alignment kernel to compare similarity between two trajectories (see [28] for details). Fig. 5a reports the sensitivities of using SVMs and three methods of improving the SVMs. All three methods—BM, BP, and KPA—improve sensitivity. Among the three, KBA achieves the largest magnitude of improvement over SVMs, around 30 percentile points. Fig. 5b shows that all methods maintain high specificity. We note that BM method performs well for detecting *M-pattern* and *back-forth*; however, it does not do well consistently over all patterns. The performance of the BM method can be highly dependent on the data distribution. Overall, BP does not work effectively, which bears out our prediction in Section 2.

5 Conclusion

We have proposed the kernel-boundary-alignment algorithm for tackling the imbalanced training data challenge. Through theoretical justifications and empirical studies, we show this method to be effective. We believe that kernel-boundary alignment is attractive, not only because of its accuracy, but also because it can be applied to learning both vector data and sequence data (e.g., DNA sequences and spatio-temporal patterns) through modifying the kernel matrix directly. Future research includes studies on formulating a robust way of incorporating prior knowledge of the imbalanced data sets to estimate the "ideal" boundary. Some prior work has been done in incorporating the prior knowledge into the optimization formulation of SVMs, such as [30]. However, the incorporation is usually not robust and depends on the prediction rules of the prior knowledge. We plan to research a more robust way and apply it on KBA.

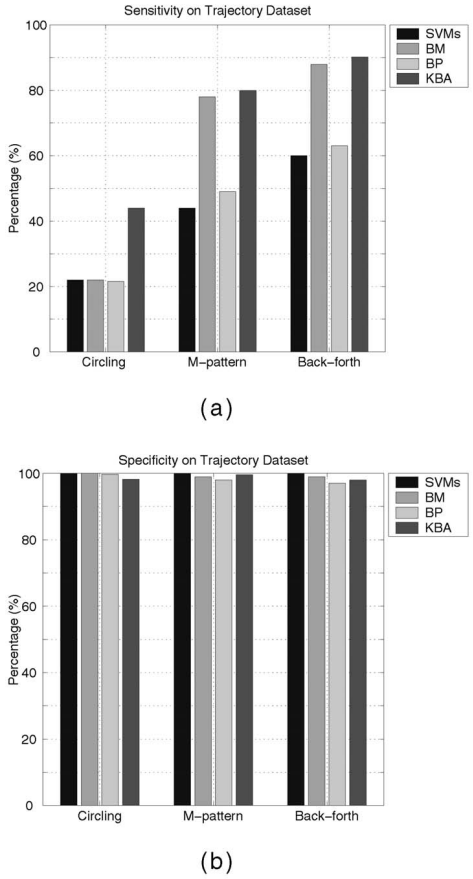

Fig. 5. Sensitivity versus specificity on the trajectory data set.

Appendix A
Proof of Conformal Transformation

Corollary 1. *The function $D(\mathbf{x})$ defined in (9) gives a valid conformal transformation on feature space $\mathcal{F}$ induced by the predefined kernel matrix $\mathbf{K}$.*

Proof. Suppose the mapped vector of an input instance $\mathbf{x}$ is $\Phi(\mathbf{x})$ before transformation and $\Psi(\mathbf{x})$ after transformation. Equation (12) defines the kernel transformation in $\tilde{k}_{\mathbf{x}\mathbf{x}'} = D(\mathbf{x})D(\mathbf{x}')k_{\mathbf{x}\mathbf{x}'}$. Thus, the cosine value of the angle between two mapped vectors $\Psi(\mathbf{x})$ and $\Psi(\mathbf{x}')$ can be written as follows [21]:

$$\begin{aligned}\cos(\angle(\Psi(\mathbf{x}),\Psi(\mathbf{x}'))) &= \frac{<\Psi(\mathbf{x}),\Psi(\mathbf{x}')>}{\|\Psi(\mathbf{x})\|\cdot\|\Psi(\mathbf{x}')\|}\\ &= \frac{D(\mathbf{x})D(\mathbf{x}')k_{\mathbf{x}\mathbf{x}'}}{\sqrt{D(\mathbf{x})D(\mathbf{x})k_{\mathbf{x}\mathbf{x}}}\sqrt{D(\mathbf{x}')D(\mathbf{x}')k_{\mathbf{x}'\mathbf{x}'}}}\\ &= \frac{k_{\mathbf{x}\mathbf{x}'}}{\sqrt{k_{\mathbf{x}\mathbf{x}}}\sqrt{k_{\mathbf{x}'\mathbf{x}'}}}\\ &= \cos(\angle(\Phi(\mathbf{x}),\Phi(\mathbf{x}'))),\end{aligned}$$

where we use the fact that $D(\mathbf{x})$ defined in (9) is a positive function. We can see that the kernel transformation by $D(\mathbf{x})$ defined in (9) does not affect pairwise-angles between the mapped data in feature space and, hence, is a valid conformal transformation. □

APPENDIX B
PROOF OF POSITIVE SEMIDEFINITENESS

Corollary 2. *When given a positive (semi) definite kernel matrix* $\mathbf{K}$*, the kernel transformation defined in (12) results in a new kernel matrix* $\tilde{\mathbf{K}}$ *which is also positive (semi) definite.*

Proof. Since $D(\mathbf{x})$ is a scalar function, we have $D(\mathbf{x})D(\mathbf{x}') =< D(\mathbf{x}), D(\mathbf{x}')$, which is a positive (semi) definite (psd) kernel function, which is the so-called one-rank kernel in [21]. Denoting $\mathbf{d} = (d_i)$ as an n-dimensional vector with $d_i = D(\mathbf{x}_i)$, where n is the number of training instances, we have a matrix $\mathbf{d}\mathbf{d}^T$ which is associated with the psd function $D(\mathbf{x})D(\mathbf{x}')$ for the training set $\mathbf{X}_{train}$. Hence, $\mathbf{d}\mathbf{d}^T$ is a psd matrix. On the other hand, (12) can be rewritten as

$$\tilde{\mathbf{K}} = \mathbf{d}\mathbf{d}^T \otimes \mathbf{K}.$$

Since the prior kernel $\mathbf{K}$ is also psd, using the closure property of kernels under tensor product $\otimes$ [21], the new kernel matrix $\tilde{\mathbf{K}}$ is a psd matrix and, hence, a valid kernel matrix. □

ACKNOWLEDGMENTS

This work was supported by the US National Science Foundation under two NSF grants: NSF Career IIS-0133802 and NSF ITR IIS-0219885.

REFERENCES

[1] S. Amari and S. Wu, "Improving Support Vector Machine Classifiers by Modifying Kernel Functions," *Neural Networks,* vol. 12, no. 6, pp. 783-789, 1999.
[2] A.P. Bradley, "The Use of the Area under the Roc Curve in the Evaluation of Machine Learning Algorithms," *Pattern Recognition,* vol. 30, no. 7, pp. 1145-1159, 1997.
[3] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, no. 2, pp. 123-140, 1996.
[4] C. Burges, "Geometry and Invariance in Kernel Based Methods," *Advances in Kernel Methods: Support Vector Learning,* Cambridge, Mass.: MIT Press, 1999.
[5] C. Cardie and N. Howe, "Improving Minority Class Prediction Using Case-Specific Feature Weights," *Proc. 14th Int'l Conf. Machine Learning,* pp. 57-65, 1997.
[6] P. Chan and S. Stolfo, "Learning with Non-Uniform Class and Cost Distributions: Effects and a Distributed Multi-Classifier Approach," *Proc. Workshop Notes KDD-98 Distributed Data Mining,* pp. 1-9, 1998.
[7] N. Chawla, K. Bowyer, L. Hall, and W.P. Kegelmeyer, "Smote: Synthetic Minority Over-Sampling Technique," *J. Artificial Intelligence and Research,* vol. 16, pp. 321-357, 2002.
[8] N. Cristianini, J. Shawe-Taylor, and J. Kandola, "On Kernel Target Alignment," *Proc. Neural Information Processing Systems,* pp. 367-373, 2001.
[9] T. Dietterich and G. Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *J. Artifical Intelligence Research,* vol. 2, pp. 263-286, 1995.
[10] C. Drummond and R. Holte, "Exploiting the Cost (in)Sensitivity of Decision Tree Splitting Criteria," *Proc. 17th Int'l Conf. Machine Learning,* pp. 239-246, 2000.
[11] T. Fawcett and F. Provost, "Adaptive Fraud Detection," *Data Mining and Knowledge Discovery,* vol. 1, no. 3, pp. 291-316, 1997.
[12] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* second ed. Boston: Academic Press, 1990.
[13] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York: Springer, 2001.
[14] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proc. 10th European Conf. Machine Learning,* pp. 137-142, 1998.
[15] J. Kandola and J. Shawe-Taylor, "Refining Kernels for Regression and Uneven Classification Problems," *Proc. Ninth Int'l Workshop Artificial Intelligence and Statistics,* 2003.
[16] G. Karakoulas and J.S. Taylor, "Optimizing Classifiers for Imbalanced Training Sets," *Advances in Neural Information Processing Systems,* 1999.
[17] M. Kubat and S. Matwin, "Addressing the Curse of Imbalanced Training Sets: One-Sided Selection," *Proc. 14th Int'l Conf. Machine Learning,* pp. 179-186, 1997.
[18] H.W. Kuhn and A.W. Tucker, "Non-Linear Programming," *Proc. Second Berkeley Symp. Math. Statistics and Probability,* 1961.
[19] Y. Lin, Y. Lee, and G. Wahba, "Support Vector Machines for Classification in Nonstandard Situations," *Machine Learning,* vol. 46, pp. 191-202, 2002.
[20] A. Nugroho, S. Kuroyanagi, and A. Iwata, "A Solution for Imbalanced Training Sets Problem by Combnet-ii and Its Application on Fog Forecasting," *IEICE Trans. Information and Systems,* vol. E85-D, no. 7, pp. 1165-1174, July 2002.
[21] B. Scholkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, Mass.: MIT Press, 2002.
[22] S. Tong and E. Chang, "Support Vector Machine Active Learning for Image Retrieval," *Proc. ACM Int'l Conf. Multimedia,* pp. 107-118, 2001.
[23] V. Vapnik, *The Nature of Statistical Learning Theory.* New York: Springer, 1995.
[24] K. Veropoulos, C. Campbell, and N. Cristianini, "Controlling the Sensitivity of Support Vector Machines," *Proc. Int'l Joint Conf. Artificial Intelligence,* pp. 55-60, 1999.
[25] G.M. Weiss, "Mining with Rarity: A Unifying Framework," *SIGKDD Explorations,* vol. 6, no. 1, pp. 7-19, June 2004.
[26] G.M. Weiss and F. Provost, "Learning When Training Data Are Costly: The Effect of Class Distribution on Tree Induction," *J. Artificial Intelligence Research,* vol. 19, pp. 315-354, 2003.
[27] G. Wu and E. Chang, "Adaptive Feature-Space Conformal Transformation for Imbalanced Data Learning," *Proc. 20th Int'l Conf. Machine Learning,* pp. 816-823, 2003.
[28] G. Wu, Y. Wu, L. Jiao, Y.-F. Wang, and E. Chang, "Multi-Camera Spatio-Temporal Fusion and Biased Sequence-Data Learning for Security Surveillance," *Proc. ACM Int'l Conf. Multimedia,* Nov. 2003.
[29] X. Wu and R. Srihari, "New ν-Support Vector Machines and Their Sequential Minimal Optimization," *Proc. 20th Int'l Conf. Machine Learning,* 2003.
[30] X. Wu and R. Srihari, "Incorporating Prior Knowledge with Weighted Margin Support Vector Machines," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 2004.

Gang Wu received the MS degree in electrical engineering from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 2001. He is currently working toward the PhD degree at the University of California at Santa Barbara (UCSB). His research interests are statistical learning theory and its application in multimedia retrieval.

Edward Y. Chang received the PhD degree in electrical engineering from Stanford University, Stanford, California, in 1999. He is currently an associate professor in the Department of Electrical and Computer Engineering, University of California at Santa Barbara. His research interests include content-based image and video retrieval, machine learning, and sensor networks. He is a senior member of the IEEE.