Adaptive Feature-Space Conformal Transformation for Imbalanced-Data Learning

Gang Wu GWU@ENGINEERING.UCSB.EDU
Edward Y. Chang ECHANG@ECE.UCSB.EDU
Department of Electrical & Computer Engineering, University of California, Santa Barbara

Abstract

When the training instances of the target class are heavily outnumbered by non-target training instances, SVMs can be ineffective in determining the class boundary. To remedy this problem, we propose an adaptive conformal transformation (ACT) algorithm. ACT considers feature-space distance and the class-imbalance ratio when it performs conformal transformation on a kernel function. Experimental results on UCI and real-world datasets show ACT to be effective in improving class prediction accuracy.

1. Introduction

Support Vector Machines (SVMs) are a core machine learning technology. They have strong theoretical foundations and excellent empirical successes in many pattern recognition applications such as handwriting recognition (Cortes & Vapnik, 1995), image retrieval (Tong & Chang, 2001), and text classification (Joachims, 1998). However, for many emerging applications, such as gene profiling, image understanding, and fraud detection (Fawcett & Provost, 1997), where 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 towards the target class. As a result, the false-negative rate can be excessively high in identifying target objects (e.g., a diseased gene or a suspicious event), and hence can render the classifier ineffective. (We will discuss the details of this problem in Section 3.)

Several attempts have been made to improve class-prediction accuracy of SVMs (Amari & Wu, 1999; Breiman, 1996; Chan & Stolfo, 1998; Karakoulas & Taylor, 1999; Kubat & Matwin, 1997; Li et al., 2002; Lin et al., 2002). Given the class prediction function of SVMs,

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

we can identify three parameters that affect the decision outcome: b, α_i, and K. (We will discuss them in Section 2.) Our empirical study shows that the only effective method for improving SVMs is through modifying the kernel function K. As indicated by (Amari & Wu, 1999), by conformally spreading the area around the class-boundary outward on the Riemannian manifold S where all mapped data are located in feature space F, we can adapt K locally to data distribution to improve class-prediction accuracy.

In this paper, we propose an adaptive conformal transformation (ACT) algorithm. ACT improves upon Amari and Wu's method for tackling the imbalanced-training-classes problem in two respects.

1. We conduct the transformation based on the spatial distribution of the support vectors in feature space F, instead of in input space I (Wu & Amari, 2002). Using feature-space distance to conduct conformation transformation takes advantage of the new information learned by SVMs in every iteration, whereas input-space distance remains unchanged.

2. We adaptively control the transformation based on the skew of the class-boundary. This transformation gives the neighborhood of minority support vectors a higher spatial resolution, and hence achieves better separation between the classes.

Our experimental results on both UCI and real-world image datasets show ACT to be very effective in correcting the skewed boundary.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 uses a 2-D example to explain the problem caused by imbalanced training data for SVMs. In Section 4 we describe the ACT algorithm for tackling the imbalanced training-data problem. Section 5 presents the setup and the results of our empirical studies. We offer our concluding remarks in Section 6, as well as suggestions for further studies.

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

2. Related Work

Researchers have modified the learning algorithm of SVMs for remedying the imbalanced training-data problem. Let us revisit the class prediction function in Eq. 1. We can identify three parameters that affect the decision outcome: b, α_i, and K. Related algorithmic work can be summarized by changing these parameters.

- *Boundary Movement* (BM). The b parameter is the intercept. We can change b to shift the decision boundary. However, this naive boundary-movement is a post-processing method. Intuitively, we can see that changing b trades a higher false positive count for a lower false negative count.
- *Biased Penalties* (BP). The α_i determines the magnitude of influence of training instance $\mathbf{x}_i$. According to Eq. 1, the larger the α_i, the larger the influence of $\mathbf{x}_i$ in class prediction. Based on this idea, Veropoulos et al. (Veropoulos et al., 1999; Lin et al., 2002) use different penalty constraints for different classes to tune the α_i's. It turns out that this biased-penalty method does not help SVMs as much as expected. According to the KKT conditions, the value of α_i has three ranges (Cristianini & Shawe-Taylor, 2000):

$$\alpha_i = 0 \Rightarrow y_i f(\mathbf{x}_i) > 1 \quad \text{and} \quad \xi_i = 0 \tag{2}$$

$$0 < \alpha_i < C \Rightarrow y_i f(\mathbf{x}_i) = 1 \quad \text{and} \quad \xi_i = 0 \tag{3}$$

$$\alpha_i = C \Rightarrow y_i f(\mathbf{x}_i) < 1 \quad \text{and} \quad \xi_i \geq 0. \tag{4}$$

  In addition, α_i's are constrained by

$$\sum_{i=1}^{n} \alpha_i y_i = 0. \tag{5}$$

  We can see that C imposes only an upper bound on α_i, not a lower bound. Increasing C does not necessarily affect α_i. Moreover, the constraint in Eq. 5 imposes an equal total influence from the positive and negative support vectors. The increases in some α_i's at the positive side will inadvertently increase some α_i's at the negative side to satisfy the constraint. These constraints can make the increase of C^+ on minority instances ineffective.
- *Kernel Modification*. The kernel function affects the distance computation between a support vector $\mathbf{x}_i$ and a testing instance $\mathbf{x}$. (We assume that K is an RBF function in this paper.) Based on this idea, Amari and Wu (Amari & Wu, 1999; Wu & Amari, 2002) propose to change the kernel using a conformal transformation method. In this paper, we further Amari and Wu's work to deal with imbalance-data training. Another method is proposed by Kandola et al. (Kandola & Shawe-Taylor, 2003), which is based on the kernel-alignment idea but with a simple transformation of the "ideal" target kernel, to adapt the kernel in the imbalanced training-data problem. Compared to (Kandola & Shawe-Taylor, 2003), our method deals with just the class-boundary data, not the entire training dataset. Furthermore, the solution we introduce here is to modify the kernel function, instead of the kernel matrix as in (Kandola & Shawe-Taylor, 2003).

An entirely orthogonal approach to the algorithmic approach that we have discussed so far, is data-processing, which under-samples the majority class or over-samples the minority class in the hope of reducing the skew of the training dataset. One-sided selection by (Kubat & Matwin, 1997) is a representative under-sampling approach, which removes noisy, borderline, and redundant majority training instances. However, these steps typically can remove only a small fraction of the majority instances, and may 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 applications. Multi-classifier training (Chan & Stolfo, 1998) and Bagging (Breiman, 1996), are two other under-sampling methods. These methods do not deal with noisy and borderline data directly, but use a large ensemble of subclassifiers to reduce prediction variance.

Over-sampling (Chawla et al., 2000) is the opposite of the under-sampling approach. It duplicates or interpolates minority instances in the hope of reducing the imbalance. The over-sampling 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. However, assumptions like these can be data-dependent and unreliable. We believe that an effective data-processing approach can complement an algorithmic approach. Our focus in this paper is on developing an algorithmic approach.

3. Boundary Bias in SVMs

A subtle but severe problem that an SVM classifier faces is the skewed class boundary caused by imbalanced training data. To illustrate this problem, Figure 1 depicts a 2D checkerboard example. The checkerboard divides a 200×200 square into four quadrants. The top-left and bottom-right quadrants contain negative (majority) instances while the top-right and bottom-left quadrants contain positive (minority) instances. The lines between the classes are the "ideal" boundary that separates the two classes. In the rest of the paper, we will use *positive* when referring to minority instances, and *negative* when referring to majority instances.

Figure 2 exhibits the boundary distortion between the two left quadrants in Figure 1 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. Figure 2(a) 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 $10:1$. Figure 2(b) shows the boundary when the ratio increases to $10,000:1$. The boundary in Figure 2(b) is much more skewed towards the positive quadrant than the boundary in Figure 2(a), and hence causes a higher incidence of false negatives.

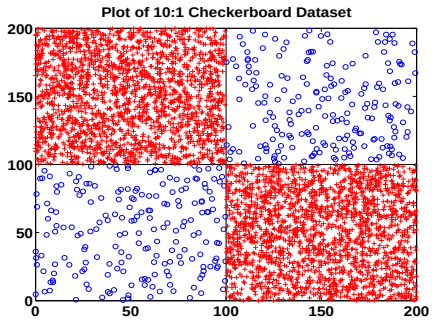
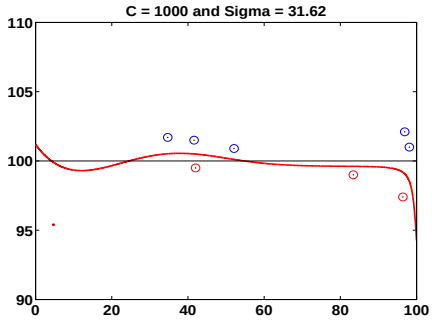
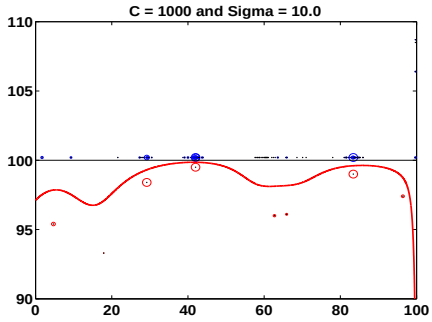


Figure 1. Checkerboard Experiment.



(a) 10:1



(b) 10,000:1

Figure 2. Boundaries of Different Ratios.

4. Adaptive Conformal Transformation

In this section, we first explain the geometry of the feature space, from which we obtain information to perform conformal transformation. We then propose the Adaptive Conformal Transformation (ACT) algorithm. We close by pointing out the differences between ACT and the prior work (Amari & Wu, 1999; Wu & Amari, 2002).

4.1. Geometry of Feature Space F

A support vector machine can use the *kernel mapping* (Vapnik, 1995; Cristianini & Shawe-Taylor, 2000) to map the data from a Euclidean input space I to a high-dimensional Hilbert feature space F, in which a classification or regression problem becomes linear. The mapped data lie on a surface S in F. In general, S has the same dimensionality as that of the input space I (Burges, 1999). The shape of S is determined by the associated mapping Φ. If we assume that Φ has all continuous derivatives, as in the case of an RBF function, the surface S in F is smooth, and thus can be considered as a Riemannian manifold, which enables us to define a Riemannian metric for S.

The relationship between I and S can be imagined from a simple Earth example. The local spot where we stand is perceived as flat, but actually the Earth itself is a globe. Figure 3 illustrates another example. Surface 1 shows that a $\Re^2$ input space I is mapped to an irregular surface in feature space F under one mapping function Φ_1. In this case, for one point P in this irregular surface, its neighborhood is locally $\Re^2$. One can also choose a different Φ_2 to map $\Re^2$ to the surface of a globe in F, where the neighborhood of one point P is still locally $\Re^2$.

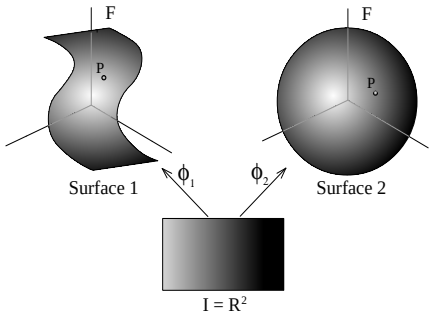


Figure 3. A Riemannian-Manifold Example.

4.1.1. Riemannian Metric

Ordinarily, we use two schemes to measure the distance between two points on S, as schematically illustrated in Figure 4. The first one considers the distance between two points along a straight line in F, which is the so-called Euclidean distance; the second one measures the distance between two points along a path on S by integration. This distance, called the Riemannian distance (Burges, 1999), is computed by a metric induced on S. This second metric is thus called the Riemannian metric, denoted as g_{ij} in this paper.

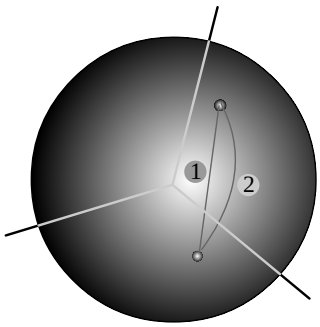
A Riemannian metric tells us how to compute the distance between any two points on S. The components of a Riemannian metric can be viewed as multiplication factors which are placed before the differential displacements dx_i in I to compute the distance ds of an element dz in F in a generalized Pythagorean theorem,

$$ds^2 = \sum_{i,j} g_{ij} dx_i dx_j. \tag{6}$$

The Euclidean distance can be also regarded as some format of the Riemannian distance, where g_{ij} is a discrete delta function δ_{ij}[1], and hence

$$ds^2 = \sum_{i} {dx_i}^2. \tag{7}$$

[1] A discrete delta function δ_{ij} equals 1 only when $i = j$, else equals 0.



1: Euclidean distance

2: Riemannian distance

Figure 4. Two Distance Measures in a 3-D Feature Space.

4.1.2. From Kernel to Riemannian Metric

Although the kernel mapping Φ is unknown, one can still calculate the Riemannian metric induced on S by playing the kernel trick.

Let $\mathbf{z}$ denote the mapped pattern of $\mathbf{x}$ in F. We have $\mathbf{z} = \Phi(\mathbf{x})$. Under this mapping function, a small vector $d\mathbf{x}$ is mapped to

$$d\mathbf{z} = \Phi(\mathbf{x} + d\mathbf{x}) - \Phi(\mathbf{x}). \tag{8}$$

Then, the Riemannian distance ds is defined as

$$\begin{aligned} ds^2 &= \|d\mathbf{z}\|^2 \\ &= \|\Phi(\mathbf{x} + d\mathbf{x}) - \Phi(\mathbf{x})\|^2 \\ &= K(\mathbf{x} + d\mathbf{x}, \mathbf{x} + d\mathbf{x}) - 2K(\mathbf{x}, \mathbf{x} + d\mathbf{x}) + K(\mathbf{x}, \mathbf{x}) \\ &= \sum_{i,j} \left(\frac{\partial^2 K(\mathbf{x}, \mathbf{x}')}{\partial x_i \partial x'_j} \right)_{\mathbf{x}'=\mathbf{x}} dx_i dx_j. \end{aligned}$$

From Eq. 6, we can see that the Riemannian metric induced on S can be defined as

$$g_{ij}(\mathbf{x}) = \left(\frac{\partial^2 K(\mathbf{x}, \mathbf{x}')}{\partial x_i \partial x'_j} \right)_{\mathbf{x}'=\mathbf{x}}. \tag{9}$$

It shows how a local area in I is magnified in F under the mapping $\Phi(\mathbf{x})$ (Burges, 1999; Amari & Wu, 1999).

4.2. Adaptive Conformal Transformation

A conformal transformation, also called a conformal mapping, is a transformation T which maps the elements $X \in D$ to elements $Y \in T(D)$ while preserving the local angles between the elements after mapping, where D is a domain in which the elements X reside (Cohn, 1980). Figure 5 shows an example using two different conformal transformations on a grid structure, where $\mathbf{z}$ is a complex variable. We can see that the local grid structure is almost invariant after such a transformation (that is, "*conformal*"). Usually, an analytic function[2] is conformal at any point where it has a nonzero derivative. In this paper, we use the term *conformal function*. Some commonly used conformal functions are represented as $\mathbf{z}^2$, $e^{-\mathbf{z}}$, and $e^{-\mathbf{z}^2}$.

[2]An analytic function is a complex function which is complex differentiable at every point in a region R. Please refer to mathematics books for the details.

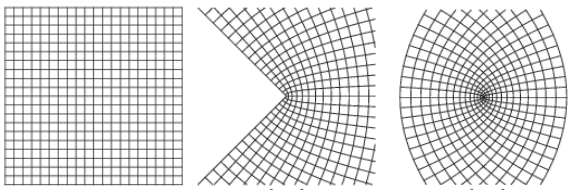


Figure 5. A Conformal Transformation Example.

In Section 4.1.2, we noted that a Riemannian metric $g_{ij}(\mathbf{x})$ induced on S shows how a local area around $\mathbf{x}$ in I is magnified in F under the mapping Φ. The idea of conformal transformation in SVMs is to enlarge the magnification factor $g_{ij}(\mathbf{x})$ around the boundary but reduce it around other points by modifying the related function K, according to Eq. 9. This can be implemented by introducing a conformal transformation of the kernel (Amari & Wu, 1999; Wu & Amari, 2002)

$$\tilde{K}(\mathbf{x}, \mathbf{x}') = D(\mathbf{x}) D(\mathbf{x}') K(\mathbf{x}, \mathbf{x}'), \tag{10}$$

where $D(\mathbf{x})$ is a properly defined positive function.

4.2.1. Selection of $D(\mathbf{x})$

Substituting Eq. 10 into Eq. 9 with the form of a new kernel $\tilde{K}(\mathbf{x}, \mathbf{x}')$, the new Riemannian metric $\tilde{g}_{ij}(\mathbf{x})$ is

$$\tilde{g}_{ij}(\mathbf{x}) = D(\mathbf{x})^2 g_{ij}(\mathbf{x}) + D'_i(\mathbf{x}) D'_j(\mathbf{x}) + 2D'_i(\mathbf{x}) D(\mathbf{x}) K'_i(\mathbf{x}, \mathbf{x}), \tag{11}$$

where $K'_i(\mathbf{x}, \mathbf{x}) = \partial K(\mathbf{x}, \mathbf{x}')/\partial x_i|_{\mathbf{x}'=\mathbf{x}}$ and $D'_i(\mathbf{x}) = \partial D(\mathbf{x})/\partial x_i$. To further increase the margin of SVMs in F, $D(\mathbf{x})$ should be chosen in a way such that $\tilde{g}_{ij}(\mathbf{x})$ has greater values around the decision boundary. In addition to dealing with the skew of the class-boundary, $\tilde{g}_{ij}(\mathbf{x})$ is greater especially for the boundary area close to the minority class. An RBF distance function is a good choice for $D(\mathbf{x})$.

In practice, since it is unknown exactly where the optimal boundary is located, the positions of support vectors[3] are used to approximate the class-boundary (Amari & Wu, 1999). Conformal transformation then increases the metric in the neighborhood of the support vectors. The degree of such magnification depends on the selection of conformal function $D(\mathbf{x})$.

Suppose the feature vector $\mathbf{x}$ has been normalized to make all its elements lie between 0 and 1. In addition, suppose that the conformal function $D(\mathbf{x})$ is chosen as $\exp(-\frac{\|\mathbf{x}-\mathbf{x}_k\|_n^n}{\tau_k^2})$, where $\mathbf{x}_k$ is a support vector, and the initial kernel $K(\mathbf{x}, \mathbf{x}')$ chosen as $\exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|_m^m)$. Due to Eq. 11, the new Riemannian metric in the neighborhood of the support vector $\mathbf{x}_k$ can be written[4] as

[3]Notice that in the case that soft margin SVMs are employed, a support vector can reside in the area of the other class. We consider such support vectors as outliers and do not consider them as support vectors in our algorithm.

[4]When $K(\mathbf{x}, \mathbf{x}')$ is a L_1-norm RBF function, it is not derivable

$$\tilde{g}_{ij}(\mathbf{x}) = D(\mathbf{x})^2 g_{ij}(\mathbf{x}) + D(\mathbf{x})^2 \frac{n^2}{\tau_k^4}(x_i - x_{ki})^{n-1}(x_j - x_{kj})^{n-1}. \quad (12)$$

From Eq. 12, we can see the new metric $\tilde{g}_{ij}(\mathbf{x})$ is dominated by the exponential function $D(\mathbf{x})$, which is associated with the norm number n, the parameter τ_k^2, and the distance $\|\mathbf{x} - \mathbf{x}_k\|$. Since we have normalized $\|x_i - x_{ki}\|$ to be ≤ 1, $D(\mathbf{x})$ becomes greater when n is increased at the points $\mathbf{x}$ distant from the support vectors $\mathbf{x}_k$, and thereby $\tilde{g}_{ij}(\mathbf{x})$ becomes greater. However, we hope that $\tilde{g}_{ij}(\mathbf{x})$ is smaller for the area far away from the class-boundary. We therefore prefer $D(\mathbf{x})$ with a smaller norm n, such as L_1 or L_2. In the remainder of this section, for simplification, we will discuss only the case when $D(\mathbf{x})$ is an L_1-norm RBF function.

4.2.2. SELECTION OF τ_k^2 FOR $D(\mathbf{x})$

Let $D(\mathbf{x})$ be an L_1-norm RBF function. Since the positions of support vectors are used to approximate the class-boundary, $D(\mathbf{x})$ can be chosen as

$$D(\mathbf{x}) = \sum_{k \in \mathbf{SV}} \exp(-\frac{|\mathbf{x} - \mathbf{x}_k|}{\tau_k^2}). \quad (13)$$

If we fix τ_k^2, $D(\mathbf{x})$ can be very large in areas where a large number of support vectors are present, and very small in areas where few support vectors are present. In other words, $D(\mathbf{x})$ is dependent on the density of support vectors in the neighborhood of $\Phi(\mathbf{x})$ (if we fix τ_k^2 for all $\mathbf{x}_k$). To alleviate this problem, we adaptively tune τ_k^2 according to the spatial distribution of support vectors in F. This goal can be achieved by the following equation:

$$\tau_k^2 = \mathrm{AVG}_{i \in \{\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_k)\|^2 < M,\ y_i \neq y_k\}} \left(\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_k)\|^2 \right). \quad (14)$$

In the above equation, the average on the right-hand side comprises all the support vectors in $\Phi(\mathbf{x}_k)$'s neighborhood within the radius of M but having a different class label. Here, M is the average distance of the nearest and the farthest support vector from $\Phi(\mathbf{x}_k)$. Setting τ_k^2 in this way takes into consideration the spatial distribution of the support vectors in F.

Though the mapping function Φ is unknown, we can play the kernel trick to calculate the distance in F:

$$\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_k)\|^2 = K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_k, \mathbf{x}_k) - 2K(\mathbf{x}_i, \mathbf{x}_k). \quad (15)$$

Substituting Eq. 15 into Eq. 14, we then can calculate a τ_k^2 for each support vector, which can adaptively reflect the spatial distribution of the support vector in the feature space, not in the input space.

at point $\mathbf{x} = \mathbf{x}'$. However, it is possible to define its "subderivative" at $\mathbf{x} = \mathbf{x}'$, which corresponds here to the average value of the right and left derivatives (d'Alche Buc et al., 2002). The equation when n is an odd number is very similar to Eq. 12.

4.2.3. TUNING τ_k^2 FOR IMBALANCED CLASSES

When the training dataset is very imbalanced, the class-boundary would be skewed toward the minority class in input space I. We then hope that the new metric $\tilde{g}_{ij}(\mathbf{x})$ in Eq. 12 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 coefficient for the τ_k^2 in Eq. 14 to reflect the boundary skew in $D(\mathbf{x})$. The new $\tilde{\tau}_k^2$ is as follows:

If $\mathbf{x}_k$ is a minority support vector,

$$\tilde{\tau}_k^2 = \eta_p \tau_k^2, \quad (16)$$

else

$$\tilde{\tau}_k^2 = \eta_n \tau_k^2. \quad (17)$$

Examining Eq. 12 and Eq. 13, we can see that $D(\mathbf{x})$ is a monotonously increasing function of τ_k^2. To increase the metric $\tilde{g}_{ij}(\mathbf{x})$ 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 $O(\frac{|\mathbf{SV}^-|}{|\mathbf{SV}^+|})$, and η_n as $O(\frac{|\mathbf{SV}^+|}{|\mathbf{SV}^-|})$, where $|\mathbf{SV}^+|$ and $|\mathbf{SV}^-|$ denote the number of minority and majority support vectors, respectively.

Figure 6 summarizes the ACT algorithm. We apply ACT on the training dataset X_{train} until the testing accuracy on X_{test} cannot be further improved. In each iteration, ACT adaptively calculates τ_k^2 for each support vector (step 10), based on the distribution of support vectors in feature space F. ACT scales the τ_k^2 according to the negative-to-positive support-vector ratio (steps 11 to 14). Finally, ACT updates the kernel and performs retraining on X_{train} (steps 15 to 17).

4.3. Comparison with Traditional Conformal Transformation

Compared to the traditional conformal transformation (Amari & Wu, 1999; Wu & Amari, 2002), our ACT algorithm has two differences:

1. The selection of τ_k^2 is dynamic in the tradition conformal transformation, but not adaptive, since the distance between two support vectors is calculated in input space I, not in feature space F. Calculating τ_k^2 in I does not reflect the spatial distribution of the Riemannian manifold S. Furthermore, the input-space distance is unchanged throughout the transformation iterations. In ACT, the τ_k^2 for the conformal function is calculated based on its distribution in F.

2. The traditional conformal transformation does not address the problem of boundary bias in the imbalanced training data scenario. In such a case, the decision boundary locates much closer to the minority class than to the majority one. We assign a larger metric $g_{ij}(\mathbf{x})$ for the boundary area

Input:
X_{train}, X_{test}, K;
θ; /* *stopping threshold* */
T; /* *maximum running iterations* */
Output:
$\mathcal{C}$; /* *output classifier* */
Variables:
$\mathbf{SV}$; /* *support vector set* */
M; /* *neighborhood range* */
$\mathbf{s}$; /* *a support vector* */
$\mathbf{s}.\tau$; /* *parameter of* $\mathbf{s}$ */
$\mathbf{s}.y$; /* *class label of* $\mathbf{s}$ */
Function Calls:
SVMTrain(X_{train}, K); /* *train SVM classifier* $\mathcal{C}$ */
SVMClassify($X_{test}, \mathcal{C}$); /* *classify* X_{test} *by* $\mathcal{C}$ */
ExtractSV($\mathcal{C}$); /* *obtain* $\mathbf{SV}$ *from* $\mathcal{C}$ */
ComputeM($\mathbf{s}, \mathbf{SV}$); /* *compute neighborhood range for* $\mathbf{s}$ */

Begin
1) $\mathcal{C} \leftarrow$ SVMTrain(X_{train}, K);
2) $\varepsilon_{old} \leftarrow \infty$;
3) $\varepsilon_{new} \leftarrow$ SVMClassify($X_{test}, \mathcal{C}$);
4) $t \leftarrow 0$;
5) **while** $((\varepsilon_{old} - \varepsilon_{new} > \theta) \&\& (t < T))$ {
6) $\mathbf{SV} \leftarrow$ExtractSV($\mathcal{C}$);
7) $\eta_p \leftarrow O(\frac{|\mathbf{SV}^-|}{|\mathbf{SV}^+|})$, $\eta_n \leftarrow O(\frac{|\mathbf{SV}^+|}{|\mathbf{SV}^-|})$;
8) **for** each $\mathbf{s} \in \mathbf{SV}$ {
9) $M \leftarrow$ComputeM($\mathbf{s}, \mathbf{SV}$);
10) $\mathbf{s}.\tau \leftarrow sqrt(\text{AVG}_{i \in \{\|\Phi(\mathbf{s}_i) - \Phi(\mathbf{s})\|^2 < M,\ \mathbf{s}_i.y \neq \mathbf{s}.y\}} \left(\|\Phi(\mathbf{s}_i) - \Phi(\mathbf{s})\|^2\right))$;
11) **if** $\mathbf{s} \in \mathbf{SV}^+$ **then** /* *a minority support vector* */
12) $\mathbf{s}.\tau \leftarrow \sqrt{\eta_p} \times \mathbf{s}.\tau$;
13) **else** /* *a majority* */
14) $\mathbf{s}.\tau \leftarrow \sqrt{\eta_n} \times \mathbf{s}.\tau$; }
15) $D(\mathbf{x}) = \sum_{\mathbf{s} \in SV} \exp\left(-\frac{|\mathbf{x}-\mathbf{s}|}{\mathbf{s}.\tau^2}\right)$;
16) $K \leftarrow D(\mathbf{x}) \times D(\mathbf{x}') \times K$;
17) $\mathcal{C} \leftarrow$ SVMTrain(X_{train}, K);
18) $\varepsilon_{old} \leftarrow \varepsilon_{new}$;
19) $\varepsilon_{new} \leftarrow$ SVMClassify($X_{test}, \mathcal{C}$);
20) $t \leftarrow t + 1$;}
21) **return** $\mathcal{C}$;
End

Figure 6. Algorithm ACT.

close to the minority class than for that close to the majority class, so as to reduce the skew of the class-boundary.

5. Experimental Results

Let CT_I denote Wu and Amari's conformal transformation algorithm based on input-space distance, ACT_I denote the ACT that uses input-space distance to transform $D(\mathbf{x})$, and ACT_F the ACT using feature-space distance. Our empirical study examined the effect of algorithm ACT in three aspects.

1. CT_I *versus* ACT_I. We evaluated the marginal effect of using adaptive τ_k^2 for shaping the conformal kernel $D(\mathbf{x})$. Both methods shape $D(\mathbf{x})$ in I, and ACT_I in addition takes the imbalance-class ratio into consideration.

2. ACT_I *versus* ACT_F. We evaluated the marginal effect of using feature-space distance for shaping the conformal kernel $D(\mathbf{x})$ compared to using input-space distance.

Dataset	attributes	positives	negatives
seg1	19	30	180
g7	10	29	185
euth1	24	238	1762
car3	6	69	1659
yeast5	8	51	1433
ab19	8	32	4145

Table 1. Six UCI Benchmark Datasets.

3. ACT_F *versus others*. We evaluated the effect of ACT_F for dealing with the imbalanced-training problem compared to other commonly used strategies, such as minority over-sampling (SMOTE (Chawla et al., 2000)), boundary movement (BM), and biased penalties (BP), which are discussed in Section 2.

In our experiments, we employed Laplacian kernels of the form $\exp(-\gamma|\mathbf{x} - \mathbf{x}'|)$ as $K(\mathbf{x}, \mathbf{x}')$. Then we used the following procedure. The dataset was randomly split into training and test subsets generated in a best ratio, which was empirically chosen for each dataset. Hyper-parameters (C and γ) of $K(\mathbf{x}, \mathbf{x}')$ were obtained for each run using 7-fold cross-validation. All training, validation, and test subsets were sampled in a stratified manner that ensured each of them had the same negative/positive ratio (Kubat & Matwin, 1997). We repeated this procedure ten times, computed average class-predication accuracy, and compared the results. The parameter setting for CT_I was achieved in the way suggested in (Wu & Amari, 2002). For ACT, we chose the stopping threshold θ as 0.001 and maximum running iteration T as 10. Our experiments were performed on six UCI datasets and a 20K, 116-category image dataset, which are described as follows:

Six UCI datasets The six UCI datasets we experimented with are, *segmentation* (seg1), *glass* (g7), *euthyroid* (euthy1), *car* (car3), *yeast* (yeast5), and *abalone* (abalone19). The number in the parentheses depicts the target class we chose. Table 1 shows the characteristics of these six datasets organized according to their negative-to-positive training-instance ratios. The top three datasets (seg1, g7, and euth1) are not-too-imbalanced. The middle two (car3 and yeast5) are mildly imbalanced. The bottom dataset (ab19) is the most imbalanced (the ratio is $130:1$).

20K-image dataset The image dataset contains 20K, 116 categories of images collected from the Corel Image CDs[5]. Each image is represented by a vector of 144 dimensions including color, texture, and shape features (Tong & Chang,

[5]We exclude from our testbed categories that are not possible to classify automatically, such as "industry", "Rome", and "Boston". (The Boston category contains various subjects, e.g., architectures, landscapes, and people, of Boston.)

Dataset	SVMs	SMOTE	$\mathrm{CT_I}$	$\mathrm{ACT_I}$	$\mathrm{ACT_F}$
seg1	98.1	98.1	96.2	98.1	**98.1**
g7	89.9	91.8	85.6	93.7	**93.7**
euth1	92.8	92.4	94.1	94.4	**94.5**
car3	99.0	99.0	99.1	99.1	**99.9**
yeast5	59.1	69.9	71.8	75.6	**78.5**
ab19	0.0	0.0	44.2	50.9	**51.9**

Table 2. UCI-dataset Prediction Accuracy.

Data		SVMs	SMOTE	$\mathrm{CT_I}$	$\mathrm{ACT_I}$	$\mathrm{ACT_F}$
seg1	a^+	96.4	96.4	92.9	96.4	**96.4**
	a^-	100.0	100.0	100.0	100.0	100.0
g7	a^+	82.1	85.7	75.0	89.3	**89.3**
	a^-	98.9	98.9	98.4	98.9	98.9
euth1	a^+	87.8	87.4	90.8	91.6	**92.0**
	a^-	98.2	97.8	97.7	97.3	97.1
car3	a^+	98.4	98.4	98.4	98.4	**100.0**
	a^-	99.6	99.6	99.9	99.9	99.8
yeast5	a^+	36.7	51.0	53.1	61.2	**63.3**
	a^-	98.4	97.5	97.6	96.4	97.6
ab19	a^+	0.0	0.0	25.0	28.6	**28.6**
	a^-	99.5	99.2	94.0	92.5	94.0

Table 3. UCI-dataset Prediction Accuracy.

2001; Chang et al., 2003). To perform class prediction, we employed the one-per-class (OPC) ensemble (Dietterich & Bakiri, 1995), 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. Table 4 presents results of 12 selected categories also organized into three groups according to the imbalance ratio (the imbalance ratios are listed in the second column).

5.1. Results on UCI Benchmark Datasets

We first report the experimental results with the six UCI datasets in Tables 2 and 3. In addition to conducting experiments with SVMs, CT_I, ACT_I, and ACT_F, we also implemented and tested SMOTE.

We used the L_2-norm RBF function for $D(\mathbf{x})$ to be consistent with (Wu & Amari, 2002). In each run, the training and test subsets are generated in the ratio $6:1$, which was empirically proven to be optimal. For SMOTE[6], the minority class was over-sampled at 200%, 400% and 1000% for each of three groups of UCI datasets in Table 1, 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 target class) and negative test accuracy, respectively (Kubat & Matwin, 1997). The table shows that ACT_I outperforms CT_I by a significant margin, especially when the imbalance ratio is large (ab19). ACT_F performs about the same or slightly better than ACT_I. ACT_F achieves the best accuracy in all of the six datasets. When the data is very imbalanced (the last row of Table 2), only ACT_F works reasonably effectively. Table 3 presents a^+ and a^- separately. ACT_F achieves the best a^+ while maintaining a good a^-.

[6]For the datasets 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 all other methods, the optimal γ was $0.004, 0.003, 0.08, 0.3, 0.5$, and 0.086 respectively. All optimal C's were $1,000$.

Category	Ratio	SVMs	BM	BP	$\mathrm{CT_I}$	$\mathrm{ACT_I}$	$\mathrm{ACT_F}$
Mountain	34 : 1	24.8	21.2	24.8	23.9	25.7	**33.3**
Snow	37 : 1	46.4	47.5	47.8	49.5	51.2	**54.6**
Desert	39 : 1	33.7	31.8	34.3	35.9	37.1	**39.1**
Dog	44 : 1	32.9	28.5	35.2	37.8	38.4	**41.5**
Woman	54 : 1	27.9	25.3	26.2	30.9	32.2	**35.3**
Church	66 : 1	21.8	19.4	21.8	21.8	**23.6**	20.0
Leaf	80 : 1	26.1	27.2	24.8	30.4	**33.6**	32.6
Lizard	101 : 1	13.9	11.8	15.1	16.0	20.0	**22.2**
Parrot	263 : 1	7.1	3.5	7.1	7.1	14.3	**14.3**
Horse	264 : 1	14.3	10.4	14.3	21.4	28.6	**28.6**
Leopard	283 : 1	7.7	5.6	7.7	0.0	14.3	**23.1**
Shark	1232 : 1	0.0	0.0	0.0	0.0	8.4	**16.6**

Table 4. Image-dataset Prediction Accuracy.

5.2. Results on $20k$ Image Dataset

The image dataset is more imbalanced than the UCI datasets. In each run, we first randomly set aside $4K$ images to be used as the test subset; the remaining $16K$ images were used for training and validation. We compared six schemes: SVMs, BM, BP, CT_I, ACT_I, and ACT_F. Notice that in this experiment, we use the L_1-norm RBF function for $D(\mathbf{x})$, since the L_1-norm RBF works the best for the image dataset (Tong & Chang, 2001; Chang et al., 2003). (Please see (Chang et al., 2003) for the kernel parameter settings.)

Table 4 presents the classification accuracy for twelve representative categories out of 116 ones, organized by their imbalance ratios. First, there was no significant increase in the average classification accuracy for the BP and CT_I schemes over that of SVMs. Second, compared to SVMs, BM obtained poor results for almost all categories. It proves that this naive method is very data-dependent, and thus cannot work well in the scenario of multi-class classification. Third, compared to SVMs, the ACT_I scheme improves prediction accuracy by 3.8%, 5.0%, and 9.1% on the three subgroup datasets, respectively. Finally, ACT_F's improvement is even more significant: it outperforms SVMs by 7.6%, 4.9%, and 13.4%, respectively. In addition, ACT_F achieves the best classification accuracy for ten out of twelve categories among all schemes (marked in bold). ACT (i.e., ACT_F) is especially effective when training classes are imbalanced.

6. Conclusion and Future Work

We have proposed an adaptive feature-space conformal transformation (ACT) algorithm, for tackling the imbalanced training-data challenge. ACT adaptively changes the shape of Riemannian manifold S where all mapped training instances reside, based on the spatial distribution of all support vectors in feature space F and their imbalance ratio. Through example, theoretical justifications, and empirical studies, we show ACT to be effective.

We plan to enhance ACT by closely examining the tuning of its parameters. First, when ACT decides the degree of magnification for a support vector, it needs to locate other support vectors in its neighborhood. Currently, ACT uses the average distance M of the nearest and the farthest support vector from $\Phi(\mathbf{x}_k)$ as the nearest-neighbor radius. We believe that the nearest neighborhood could be more adaptive to local data density. Second, Eqs. 16 and 17 set the scaling factors η_p and η_n based on the support-vector ratio. Although this rule-of-thumb has provided ACT good performance through empirical validation, we plan to investigate theoretical justification and experiment with other methods for setting these scaling parameters.

References

Amari, S., & Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks, 12*, 783–789.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123–140.

Burges, C. (1999). *Geometry and invariance in kernel based methods. in adv. in kernel methods: Support vector learning*. MIT Press.

Chan, P., & Stolfo, S. (1998). Learning with non-uniform class and cost distributions: Effects and a distributed multi-classifier approach. *Workshop Notes KDD-98 Workshop on Distributed Data Mining*, 1–9.

Chang, E., Goh, K., Sychay, G., & Wu, G. (2003). Content-based soft annotation for multimodal image retrieval using bayes point machine. *IEEE Transactions on Circuits and Systems for Video Technology Special Issue on Conceptual and Dynamical Aspects of Multimedia Content Description, 13*, 26–38.

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. P. (2000). Smote:synthetic minority over-sampling technique. *International Conference on Knowledge Based Computer Systems*.

Cohn, H. (1980). *Conformal mapping on riemann surfaces*. Dover Pubns.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning, 20*, 273–297.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines*. Cambridge University Press.

d'Alche Buc, F., Grandvalet, Y., & Ambroise, C. (2002). Semi-supervised marginboost. In *Advances in Neural Information Processing Systems*.

Dietterich, T., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artifical Intelligence Research, 2*, 263–286.

Fawcett, T., & Provost, F. (1997). Adaptive fraud detection. *In Data Mining and Knowledge Discovery, 1*, 291–316.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 137–142.

Kandola, J., & Shawe-Taylor, J. (2003). Refining kernels for regression and uneven classification problems. *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*.

Karakoulas, G., & Taylor, J. S. (1999). Optimizing classifiers for imbalanced training sets. *Advanced in Neural Information Processing Systems*.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. *Proceedings of the Fourteenth International Conference on Machine Learning*, 179–186.

Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J., & Kandola, J. (2002). The perceptron algorithm with uneven margins. *Proceedings of the Nineteeth International Conference on Machine Learning*, 379–386.

Lin, Y., Lee, Y., & Wahba, G. (2002). Support vector machines for classification in nonstandard situations. *Machine Learning, 46*, 191–202.

Tong, S., & Chang, E. (2001). Support vector machine active learning for image retrieval. *Proceedings of ACM International Conference on Multimedia*, 107–118.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer.

Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of support vector machines. *Proceedings of the International Joint Conference on Artificial Intelligence*, 55–60.

Wu, S., & Amari, S. (2002). Conformal transformation of kernel functions: A data-dependent way to improve the performance of support vector machine classifiers. *Neural Processing Letter, 15*.