Borderline Over-sampling in Feature Space for Learning Algorithms in Imbalanced Data Environments

Kittipat Savetratanakaree, *Member, IAENG*
Kingkarn Sookhanaphibarn, Sarun Intakosum and Ruck Thawonmas

***Abstract*—In this paper, we propose a new approach to over-sample new minority-class instances along the borderline using the Euclidean distance in the feature space to improve support vector machine (SVM) performance in imbalanced data environments. SVM has been an outstandingly successful classifier in a wide variety of applications where balanced class data distribution is assumed. However, SVM is ineffective when coping with imbalanced datasets whereby the majority-class instances far outnumber the minority-class instances. Our new approach, called Borderline Over-sampling in the Feature Space, can deal with imbalanced data to effectively recognize new minority-class instances for better classification with SVM. The results of our class prediction experiments using the proposed approach demonstrate better performance than the existing SMOTE, Borderline-SMOTE and borderline over-sampling methods in terms of the g-mean and F-measure.**

***Index Terms*—Borderline Over-sampling in the Feature Space, Imbalanced Dataset, Over-sampling, SVM in Imbalanced Data Environments**

I. Introduction

SUPPORT vector machine (SVM) is an extremely successful classifier proposed by Vapnik [1] under the presumed condition of balanced data distributions among classes. However, SVM is ineffective when mining data with imbalanced classes. An imbalanced dataset in which the representation between classes is not approximately equal. There are many applications in real-world domains that have innately imbalanced datasets including fraudulent telephone call detection [2], oil spill detection in satellite images [3], telecommunications risk management [4], credit card fraud detection [5], IVF embryos implantation [6], balancing class in clinical dataset[7], text categorization, and unusual disease diagnosis [8].

By mining a large amount of balanced data, SVM classifiers can extract valuable knowledge for decision making support and other objectives. Hidden valuable knowledge sometimes resides in minority-class instances. Minority-class instances are thus often more useful than the majority-class instances and are also called positive instances. Majority-class instances are also called negative instances. On imbalanced datasets, the positive instances are generally misclassified by SVM classifiers because they can be treated as noise.

In some cases, the issue of class imbalance is critical and cannot be ignored. One example is the classification of pixels in mammogram images for possible breast cancer [9]. In this application, the majority-class of normal pixels might contain 98% of the data, whereas the minority-class of abnormal pixels may contain only 2%. If the machine learning algorithm ignores the abnormal pixels, patients' lives could be threatened. Classification algorithms often perform worse in the detection of such unusual cases, which tend to be the most important ones.

There are several methods [10] for overcoming the imbalanced class problem in SVM. The methods are classified into two main groups. The first group comprises external methods: data preprocessing methods that adjust the distribution of class datasets before training SVM classifiers. The second group comprises internal methods: algorithmic modifications to SVM to decrease its sensitivity to imbalanced classes.

In this paper, we propose an over-sampling method called Borderline Over-sampling in the Feature Space (BOSFS) that fits into the first group of data preprocessing methods. BOSFS conducts over-sampling by generating new synthetic minority-class instances with the nearest existing neighbors focused on the borderline in the feature space. These new synthetic instances are combined with the original imbalanced training dataset to form a new training dataset. SVM is trained using the new training dataset, and then assessed on an independent testing dataset. With this new BOSFS method, the SVM classifier achieves higher recognition performance for the minority-class instances in the imbalanced testing dataset.

II. Background and related work

In the external methods category, there are two different approaches: resampling and ensemble learning. First, resampling methods [11] consist of random, focused under- or over-sampling methods. These methods balance the minority-class instances and majority-class instances in the datasets before training SVM models. In the under-sampling approach, the random instances of the majority-class are removed until the datasets are balanced. In the over-sampling approach, the minority-class instances are randomly duplicated to achieve an approximately one-to-one ratio with the majority-class instances. Some research [12] [13] [14]

Manuscript received February 12, 2016, revised May 30, 2016.

K. Savetratanakaree and S. Intakosum are with the Computer Science Department, School of Science, King Mongkut's Institute of Technology, Ladkrabang, Bangkok, Thailand.

K. Sookhanaphibarn is with the Computer Science and Software Engineering Department, School of Science and Technology, Bangkok University, Bangkok, Thailand.

R. Thawonmas is with Ritsumeikan University, Shiga, Japan.

Part of this work by the fourth author was supported in part by Grant-in-Aid for Scientific Research (C), Number 26330421, JSPS. Corresponding author is K. Savetratanakaree, Email: kittipatsavet@gmail.com.

has pointed out that information might be lost by using the under-sampling method because important minority-class information might be randomly removed. In contrast, there is no information loss when using the over-sampling method, which may lead to better results. Over-sampling methods include synthetic data generation methods such as SMOTE [15], Borderline-SMOTE [16], and borderline over-sampling (BOS) [17] [18].

Second, ensemble learning methods divide the majority-class dataset into many subdatasets. The number of majority-class instances in each of these subdatasets is equal to the number of minority-class instances. These approaches may use clustering methods or random sampling with or without replacement (bootstrapping). Different SVM classifiers are used for each training dataset, which consists of the same positive dataset combined with a different negative subdataset.

In this paper, we focus on the external (data preprocessing) over-sampling methods. SMOTE [15] is one of the most popular methods that over-samples the minority class by generating new synthetic minority-class instances rather than by over-sampling with replacement. This is done interpolating the k nearest neighbors of randomly-chosen existing minority-class instances. Borderline-SMOTE [16], a variant of SMOTE, performs over-sampling by interpolating the k nearest neighbors of each minority-class instance focused on the borderline. Concentrating on instances in the borderline area has been proven to achieve higher SVM performance. The rest of the minority-class instances in areas other than the borderline are removed from consideration.

BOS [17] [18], yet another variant of SMOTE, focuses on using both interpolation and extrapolation techniques to generate synthetic minority-class instances along the borderline from existing minority-class instances that are minority-class support vectors of SVM and a number of their nearest neighbors. All of these algorithms use the over-sampling method to generate new synthetic minority-class instances, but the BOS Algorithm in [17] [18] outperforms both SMOTE and Borderline-SMOTE.

When SVM is used as the classifier, SMOTE, Borderline-SMOTE, and BOS find nearest neighbors of interest in the input space, not feature space, of SVM. Our proposed BOSFS uses the kernel function as a mapping function of those neighbors or minority-class instances in the input space to the feature space. BOSFS uses specific equations of Euclidean distance with the kernel function to find such nearest neighbors to the borderline directly in the feature space. BOSFS also applies a combination of interpolation and extrapolation techniques with the Euclidean distance in the feature space to create new synthetic minority-class instances or positive instances along the borderline. Thus, we expect that such synthetic positive instances could become new support vectors contributing to better SVM performance.

III. Imbalanced data classification with support vector machines

A. Support Vector Machines

The aim of support vector machines [19][20] is to find the optimal boundary that separates the negative and positive instances with the largest margin. Consider the training dataset $\{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_i, y_i)\}$ where $\mathbf{x}_i$ represents the training instance and y_i represents the label of the instance: $y_i \in \{-1, +1\}$. Using the training dataset, SVM creates an optimal boundary. This boundary can be computed by minimizing the objective function as follows:

$$\min_{w,b,\xi} \frac{1}{2}\mathbf{w} \cdot \mathbf{w}^T + C \sum_{i=1}^{N} \xi_i \tag{1}$$

subject to

$$\begin{cases} \forall_i y_i(\mathbf{w}^T \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ \forall_i \xi_i \geq 0 \end{cases} \tag{2}$$

where $\mathbf{w}$ is the weight vector, y_i are the labels, b is the offset or bias of the hyperplane, $\Phi(\cdot)$ is the mapping function from a point in the input space to a corresponding point in the feature space, ξ_i are the slack variables (considering the non-separable case by admitting misclassification of training instances), and C is the user-specified parameter for the penalty on training instances on the wrong side of the boundary. If the C parameter is very small, SVM classifies all instances as negative to maximize the margin. Clearly, this causes more training errors with respect to positive instances. [21] proposed a solution to this problem by using different parameters C, such that C^+ corresponds to the minority-class instances and C^- corresponds to the majority-class instances. The tradeoff C^+ is chosen to be larger than C^- according to the data imbalance ratio.

In Equation (1), minimizing the objective function by minimizing the first and second terms corresponds to maximizing the margin and minimizing the training errors, respectively. The dual representation of Equation 1 is as follows:

$$\max W(\alpha) \equiv \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{3}$$

subject to

$$\begin{cases} \forall_i 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{4}$$

where y_i are the labels, $K(\mathbf{x}_i, \mathbf{x}_j)$ represents a Kernel function (as shown in (7) and (9), also known as the Kernel trick to compute dot products in the feature space without knowing the real Φ mapping), and α_i's are the Lagrange multipliers which are nonzero only for the training instances that fall within the margin. These training instances are called support vectors [19] and they are crucial instances of the training dataset. They define the optimal hyperplane of the decision boundary to separate positive and negative instances.

B. SVM and imbalanced class data

There are two main problems with using SVM dealing to classify imbalanced datasets [22]. First, the SVM classifier is biased towards the majority-class instances and thereby achieves a poor classification rate on minority-class instances. Fig. 1 shows that, in imbalanced data environments, the borderline (solid line) is skewed towards the minority class instances. In other words, the positive instances (white triangles) are further from the ideal boundary. This problem occurs because the number of negative instances is higher than the number of positive instances around

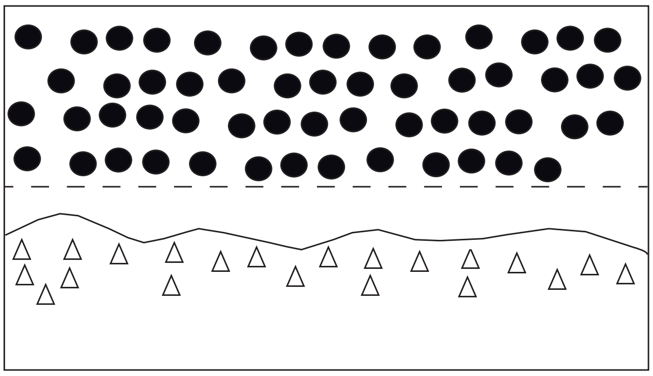


Fig. 1. The borderline (solid line) when training with SVM is skewed towards the minority instances. The dashed line shows an ideal borderline.

the ideal boundary. This may cause the misclassification of positive instances by the SVM such that the prediction results yield all negative instances. In this case, the example of an ideal borderline (dashed line in Fig. 1) is much less skewed and should have more generalization ability in classifying unknown instances.

To overcome this problem in classifying imbalanced datasets using SVM, the main target is to expand the borderline derived by SVM by over-sampling new synthetic minority instances (new positive instances) shifted towards the ideal borderline, as shown in Fig. 1. This has been approached by using an interpolation technique as proposed by [23], and by a combination of interpolation and extrapolation techniques in BOS [17] [18], as further explained in Section IV C.

Second, soft-margins [22] minimize the objective function in (1) by minimizing the first and second terms corresponding to maximizing the margin and minimizing the training errors, respectively. The weakness of soft-margins lies in the fact that the C parameter in the second term ($C\sum_{i=1}^{N}\xi_i$) is a constant chosen by the user. The C parameter specifies the tradeoff that the user is willing to allow between maximizing the margin and minimizing the training errors. If C is too high or low, it may cause over- or under-fitting problem, respectively. We solve this problem by controlling the sensitivity of SVM using different parameters C, as proposed by [21]. The sensitivity of SVM is the ratio between the number of true positive predictions (TP) and the number of positive instances in the test set:

$$\textit{sensitivity} = \frac{TP}{TP + FN} \tag{5}$$

where FN is the number of false negatives.

The specificity of SVM is the ratio between the number of true negative predictions (TN) and the number of negative instances in the test set:

$$\textit{specificity} = \frac{TN}{TN + FP} \tag{6}$$

where FP is the number of false positives.

We must control the sensitivity of SVM with different parameters C by using the cross-validation method. We thus determine the optimal C to solve this problem.

C. Main drawbacks of existing methods

Before over-sampling by generating new synthetic positive instances with interpolation or extrapolation techniques, there is the crucial step of finding the nearest neighbors of the positive instances along the borderline to determine the optimal boundary of SVM. Fig. 2 provides an illustration of the difference between our proposed method and the previous methods.

In Fig. 2a, the lighter line is a border line in the input space. The negative instances (black circles) and positive instances (white triangles) are in the original imbalanced training dataset or in the input space. $\mathbf{x}_i, \mathbf{x}_j$, and $\mathbf{x}_k$ are examples of any positive instances (white triangles) in the input space. The arrows depict the distances between each positive instance for each $\mathbf{sv}_i$ (stars) in the input space. $\mathbf{x}_j$ (white triangle with circle) is selected to be the nearest neighbor for $\mathbf{sv}_i$ in the input space. In Fig. 2b the standard SVM creates the borderline by using the original imbalanced training dataset. The borderline (solid line) classifies the negative instances (black circles) and positive instances (white triangles) in the feature space. The $\mathbf{sv}_i$ (star) is any positive instance on the borderline in the feature space. Some positive instances in the input space are individually associated to their corresponding positions in the feature space by double-sided arrows.

In Fig. 2a the Borderline-SMOTE [16] and BOS [17], [18] algorithms find the nearest neighbors for $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$ which are in the input space to each $\mathbf{sv}_i$ (a corresponding point) in the input space. The algorithms consider only the positive instances that are closest to the $\mathbf{sv}_i$ (star) corresponding point in the input space rather than the $\mathbf{sv}_i$ (star) on the borderline in the feature space. The white triangle with circle ($\mathbf{x}_j$) is selected to be the nearest neighbors to $\mathbf{sv}_i$ in the input space. In Fig. 2b The real nearest neighbor to $\mathbf{sv}_i$ in the feature space is ($\mathbf{x}_k$), which is not selected by Borderline-SMOTE or BOS. The SMOTE [15] algorithm also finds the nearest neighbors in the same way as Borderline-SMOTE and BOS, but it considers the existing positive instances of the entire area rather than just in the borderline area.

The main issue in SMOTE, Borderline-SMOTE, and BOS is that the nearest neighbor of $\mathbf{x}_i$ and $\mathbf{sv}_i$ should be found in the feature space rather than the input space. The proposed BOSFS uses the kernel function in Equations (7) and (9), and the Euclidean distance in Equation (8) between any $\mathbf{sv}_i$ and $\mathbf{x}_i$ to find the new nearest neighbors of any $\mathbf{x}_i$ for each $\mathbf{sv}_i$ in the feature space. [24] showed that the Euclidean distance using the kernel function in the feature space can be used effectively with the SVM classifier. These new nearest neighbors are combined with interpolation and extrapolation techniques to generate new synthetic positive instances for the new borderline. Our BOSFS method thus achieves the main target of expanding the new borderline for improved SVM performance when dealing with imbalanced class datasets.

IV. Proposed method

A. Main concept of the BOSFS algorithm

We propose BOSFS to allow SVM to better deal with imbalanced data environments. It has been shown that focusing on positive instances in the borderline area [16] [25] is important to achieve better SVM performance. There are several kernel functions used in SVM including the linear kernel, polynomial kernel, and radial basis function (RBF)

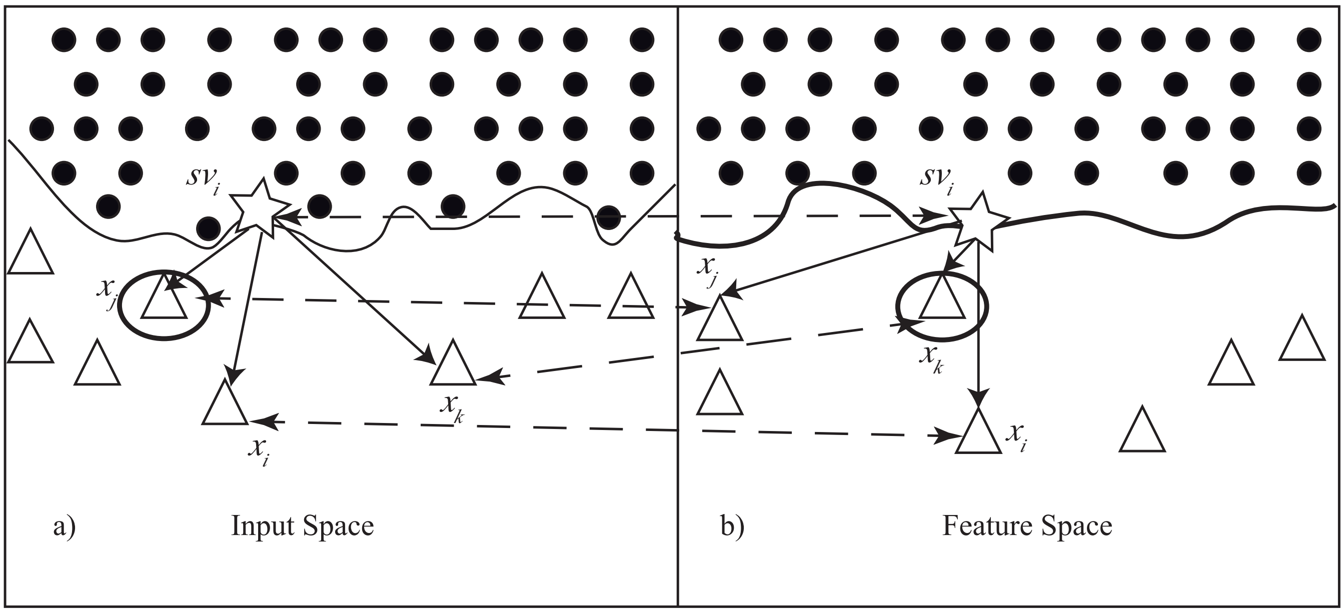


Fig. 2. Illustration of the difference between the previous methods and our proposed BOSFS method. a) The arrows depict the distances between positive instances $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$ to each $\mathbf{sv}_i$ on the borderline in the input space. This figure illustrates that the SMOTE, Borderline-SMOTE, and BOS algorithms find their nearest neighbors to $\mathbf{sv}_i$ (star) in the input space. $\mathbf{x}_j$ (white triangle with circle) is selected to be the nearest neighbor for $\mathbf{sv}_i$ in the input space. b) There is a dashed double-sided arrow between $\mathbf{sv}_i$ (star) in the input space and its corresponding point in the feature space. The arrows depict the distances between corresponding positive instances $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$ to each $\mathbf{sv}_i$ on the borderline in the feature space. The real nearest neighbor $\mathbf{x}_k$ (white triangle with circle) is selected for $\mathbf{sv}_i$ (star) in the feature space. This figure illustrates that the new BOSFS algorithm finds the real nearest neighbors to $\mathbf{sv}_i$ (stars) directly in the feature space. This process is an important step in generating new synthetic positive instances for the new borderline to improve the performance of SVM.

kernel. We conducted several experiments on UCI datasets using different kernel methods. The best result was obtained when using the RBF kernel. The RBF kernel may be more suitable for nonlinear relationships between class labels and attributes [26] if the number of features is not too large. Hence, we use the RBF kernel function in SVM and in the kernel functions in Equations (7), (8), and (9).

BOSFS creates a borderline after training an RBF kernel SVM on the original training dataset. BOSFS introduces a new way of computing the nearest neighbor by using the kernel function in Equations (7) and (9) and the Euclidean distance in Equation (8) to select the k nearest neighbors with the nearest distances of positive instances along the borderline in the feature space. New synthetic positive instances are generated by these new k nearest neighbors with a combination of interpolation and extrapolation techniques, as in BOS. The selection between the interpolation or extrapolation technique depends on the density of negative instances along the borderline[17] [18]. The BOSFS approach finds the density of negative instances along the borderline by nonlinear mapping of these negative instances of the input space to their corresponding points in the feature space. Then, our algorithm over-samples to obtain new synthetic positive instances and adds these new instances to the original training dataset. Using the new training dataset, the improved performance of SVM is assessed by testing on an independent testing dataset. We explain these steps in detail below.

B. Euclidean distance in the feature space

Kernel methods [1],[27],[28] are performed on the feature space H that is generated from the input space X by using a nonlinear map $\Phi(\mathbf{x}_i)$. BOSFS uses the following kernel function (K) representation.

$$K : X \times X \to \mathbb{R}, \qquad K(\mathbf{a}, \mathbf{b}) = \mathbf{\Phi}(\mathbf{a})'\mathbf{\Phi}(\mathbf{b}) \tag{7}$$

The Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space [24] [29] is d_{ij} and can be computed as follows:

$$\begin{aligned}(d_{ij})^2 &= \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 \\ &= \|\Phi(\mathbf{x}_i)\|^2 + \|\Phi(\mathbf{x}_j)\|^2 + 2\|\Phi(\mathbf{x}_i)\|\|\Phi(\mathbf{x}_j)\| \\ &= K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)\end{aligned} \tag{8}$$

With Equation (8) we can now find the Euclidean distance in the feature space of any $\mathbf{x}_i$, and $\mathbf{x}_j$ in the input space. The nearest neighbor $\mathbf{x}_j$ for each $\mathbf{sv}_i$ on the borderline is determined by the nearest Euclidean distance in the feature space.

Fig. 3 illustrates how our BOSFS algorithm finds the nearest neighbor of each $\mathbf{sv}_i$ on the borderline in the feature space. The k nearest neighbors $(\mathbf{x}_k)$ in the input space for each $\mathbf{sv}_i$ on the borderline are selected by BOSFS to create new synthetic positive instances for the new training dataset. The black triangle is an example of a new synthetic positive instance created by the interpolation technique, while the black triangle within white triangle is an example of a new synthetic positive instance created by the extrapolation technique. These new synthetic positive instances in the new training dataset serve to improve the performance of SVM for imbalanced datasets.

We have conducted several experiments with different k = 3, 5, and 7 for k nearest neighbors. There is no significant difference between the results achieved by using k = 3, 5, and 7. Hence, we set k to 5 in all experiments, as was done in SMOTE, Borderline-SMOTE, and BOS.

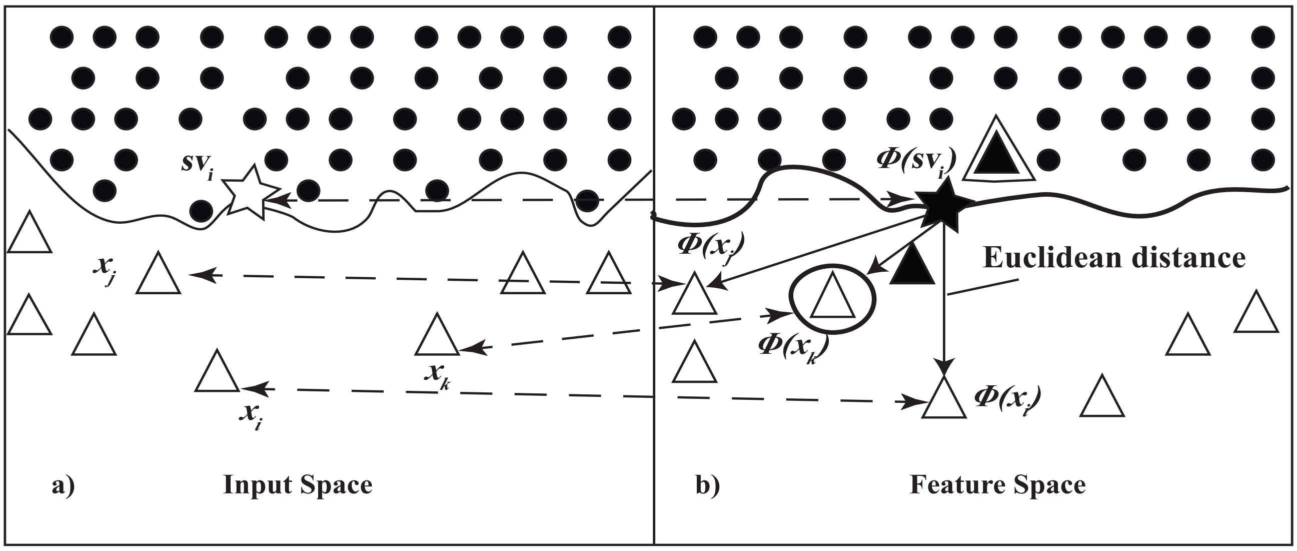
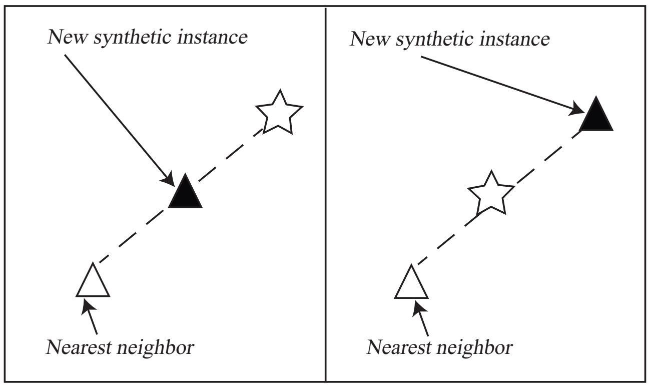


Fig. 3. Illustration of creating the new synthetic positive instances by using the proposed BOSFS. a) $\mathbf{sv}_i$ (white star) is an example of any positive instance in the input space and its corresponding point (black star) is on the borderline created by SVM in the feature space. b) BOSFS uses the RBF kernel function to map any $\mathbf{x}_i, \mathbf{x}_j$, $\mathbf{x}_k$, and $\mathbf{sv}_i$ of positive instances in the input space to their corresponding points $\Phi(\mathbf{x}_i)$, $\Phi(\mathbf{x}_j)$, $\Phi(\mathbf{x}_k)$, and $\Phi(\mathbf{sv}_i)$ in the feature space. The dashed double-sided arrows show the non-linear mapping from points in the input space to their corresponding points in the feature space. The solid arrows show the Euclidean distance between $\Phi(\mathbf{sv}_i)$ and any other positive instances $\Phi(\mathbf{x}_i)$, $\Phi(\mathbf{x}_j)$, and $\Phi(\mathbf{x}_k)$ in the feature space. The nearest neighbor of $\Phi(\mathbf{sv}_i)$ is $\Phi(\mathbf{x}_k)$ (white triangle with circle) in the feature space. $\mathbf{x}_k$ is selected for the corresponding point in the input space for the real nearest neighbors of $\Phi(\mathbf{sv}_i)$ on the borderline in the feature space. The black triangle is an example of a new synthetic positive instance created by the interpolation technique. Its position remains in between the black star and the nearest neighbor $\Phi(\mathbf{x}_k)$. The black triangle within the white triangle is an example of a new synthetic positive instance created by the extrapolation technique. Its position is further from the black star and its nearest neighbor $\Phi(\mathbf{x}_k)$.

C. Interpolation and extrapolation techniques in BOSFS

We used an interpolation technique as used in SMOTE [15] to create new data points for a discrete set of known data points in Fig. 4a. A new synthetic positive instance (black triangle) is randomly created using the interpolation technique within the area between the positive instance on the borderline (star) and the nearest neighbor positive instance (white triangle). We determine the difference between the white triangle and star where the minuend is white triangle and the subtrahend is star. A new synthetic positive instance is created by multiplying this difference by a random number between 0 to 1 and adding it to a positive instance (star) under consideration. This computation step of the interpolation technique is shown in step 4 in the BOSFS algorithm. The interpolation technique is applied when there is a crowd density of negative instances near the borderline. This technique then increases the number of positive instances in the crowd density area of negative instances near the borderline.



a) Interpolation like in SMOTE b) Extrapolation like in BOS

Fig. 4. Illustration of the difference between interpolation and extrapolation technique. a) The star is the instance under consideration. The white triangle is the nearest neighbor of the positive instance to the star. The new synthetic instance (black triangle) found by the interpolation technique will be located between the star and white triangle. b) The new synthetic instance found by the extrapolation technique will be located further from the star and white triangle.

In addition, we used an extrapolation technique as in BOS[17] [18] to create new data points by the extension process of estimating beyond the original observation range in Fig. 4b. The extrapolation technique performs the same calculation process as the interpolation technique but the aforementioned difference between the white triangle and star is reversed such that the minuend is the star and the subtrahend is the white triangle. The extrapolation technique can expand the area of the new synthetic positive instance (black triangle) further from the positive instance (star) and its nearest neighbor (white triangle) to the ideal borderline. This technique is applied when there is a lower density of negative instances near the borderline. A low density of negative instances is determined when the number of negative instances is less than half of the number of its nearest neighbors in the feature space. It has been shown in BOS that the new borderline with the extrapolation technique can be shifted towards the ideal borderline. This computation step of the extrapolation technique is shown in step 4 in the BOSFS algorithm.

The BOSFS algorithm uses a combination of Interpolation and extrapolation techniques, as in SMOTE and BOS. The main difference between BOSFS and SMOTE, Borderline-SMOTE and BOS is that BOSFS considers the density of negative instances near the borderline in the feature space rather than in the input space. BOSFS maps all of the negative instances near the borderline to their corresponding points in the feature space. It computes the Euclidean distance to find the nearest negative instances to the borderline in the feature space. This can be seen in Fig. 3 but the dashed double-sided arrows map the negative instances (black circles) instead of the positive instances (white triangles). The Euclidean distance in Equation (8) using the kernel function in Equations (7) and (9) is recomputed to find the nearest negative instances to the borderline in the feature space.

If the total number of nearest negative instances to the borderline in the feature space is less than half of the number of its nearest neighbors (lower density case), we apply the extrapolation technique. Otherwise, in the crowd density case, we apply the interpolation technique. We must determine the total number of nearest negative instances to the borderline to select the appropriate technique between interpolation and extrapolation to create the new synthetic positive instance.

D. BOSFS algorithm

BOSFS is described as follows:

The notations used for our BOSFeatureSpace algorithm are as follows.

Main variables:

· X	: Training dataset
· P^+	: Set of positive instances in X
· P^-	: Set of negative instances in X
· N	: Over-sampling rate (N is a percent rate such as 100, 200, $\cdots$)
· SV^+	: Set of positive instances on the borderline in support vectors (SVs).
· k	: Number of nearest neighbors of positive instances to the borderline.
· ϖ	: Maximum number of negative instances indicating the threshold density of negative instances to select the interpolation or extrapolation technique.
· np	: Total number of new synthetic positive instances to create.
· nn	: Total number of negative instances nearest to each $\mathbf{sv}_i$ in SV^+ for the borderline.
· §	: Array containing the k nearest neighbors of the positive instances for each $\mathbf{sv}_i^+ \in SV^+$.
· $\mathfrak{D}$	: Array of Euclidean distances (d_{ij}) for each $\mathbf{sv}_i^+ \in SV^+$ and its nearest neighbors $\mathbf{x}_j$ in P^+ in the feature space.
· K	: RBF kernel function computed as in Equation (9).
· p^+_{new}	: New synthetic positive instances for new borderline.
· X_{new}	: New over-sampled training dataset

Algorithm: BOSFeatureSpace

Input Parameters: X, N, k, ϖ

Output Parameters: X_{new}

Begin

1)
$$np = \left(\frac{N}{100} \times |X|\right)$$
where $|X|$ is the size of the training dataset.

2) Compute SV^+ by training RBF kernel SVMs on X.

3) For each $\mathbf{sv}_i^+ \in SV^+$ and each $\mathbf{x}_j \in P^+$
Perform the mapping function ($\mathbf{x}$, $\mathbf{x}'$) to map $\mathbf{x}$, $\mathbf{x}'$ into the feature space by using the RBF kernel function as follows:
$$K(\mathbf{x}, \mathbf{x}') = exp\left(\frac{-\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{9}$$
where $Sigma$ (σ) is a user-specified parameter.
Compute $\mathfrak{D}$ to keep the Euclidean distance d_{ij} in the feature space as follows:
$$d_{ij} = \sqrt{K(\mathbf{sv}_i, \mathbf{sv}_j) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{sv}_i, \mathbf{x}_j)}$$
where $i \neq j$, i = 1, 2, $\cdots$ $|SV^+|$ and j = 1, 2, $\cdots$ $|P^+|$, $|SV^+|$ is the size of the set of positive instances on the borderline, and $|P^+|$ is the size of the set of positive instances in X.
d_{ij} are sorted in ascending order to find $\S[i][k]$ where k signifies the nearest neighbors of the positive instances for each $\mathbf{sv}_i^+$ in the feature space.

4) For each $\mathbf{sv}_i^+ \in SV^+$ and each $\mathbf{x}_j \in P^-$,
repeat step 3 to find the nn negative instances nearest to $\mathbf{x}_j$ for each $\mathbf{sv}_i$ in SV^+ for the borderline.

If nn is less than half of the ϖ nearest neighbors of the negative instances (extrapolation case as in BOS)

Create p^+_{new} according to np with each $\S[i]$ for synthetic positive instances i using the following formula:
$$p^+_{new} = \mathbf{sv}_i^+ + \rho(\mathbf{sv}_i^+ - \S[i][j])$$
where $\S[i][j]$ is the j-th positive nearest neighbor of $\mathbf{sv}_i^+$ and ρ is a random number in the range [0,1].

else (interpolation case as in SMOTE)
$$p^+_{new} = \mathbf{sv}_i^+ + \rho(\S[i][j] - \mathbf{sv}_i^+)$$

5) Combine $\{p^+_{new}\}$, a set of new synthetic positive instances along the borderline, with the original training dataset to form the new training dataset X_{new} as follows:
$$X_{new} = X \cup \{p^+_{new}\} \tag{10}$$

End

V. Experiments and results

A. Data description

In our experiments, we use a total of five datasets from the UCI machine learning repository [8]: Abalone (5), Glass (7), Page-blocks (4), Spect (0), and Yeast (5). The numbers in the parentheses indicate the class numbers that are selected as positive instances whereby the remaining classes become the negative instances. The dataset statistics and the over-sampling rates performed in our experiments are shown in Table I. The reason we use these five datasets is because

TABLE I
FIVE UCI DATASETS WITH OVER-SAMPLING RATE (%).

Dataset	Attributes	No of Instances	Imbalance ratio	Over-Sampling rate(%)
Abalone	8	4177	35	100, 500, 1000, 1500, 2000, 2500, 3000, 3400
Glass	9	214	6	100, 200, 300, 400, 500
Page-blocks	10	5473	61	100, 1000, 2000, 3000, 4000, 5000, 6000
Spect	22	267	4	100, 200, 300
Yeast	8	1484	28	100, 500, 1000, 1500, 2000, 2500, 2700

they cover a variety of imbalance ratios (Minority:Majority): Spect (1:4), Glass (1:6), Yeast (1:28), Abalone (1:35) and Page (1:61).

B. Experimental setting

We compare our BOSFS algorithm with the SMOTE, Borderline-SMOTE and BOS algorithms. It has already been shown in [17] [18] that BOS outperforms SMOTE [15], Borderline-SMOTE [16], and standard SVM. The borderline instances are derived by the support vectors after training an SVM on the original training set. The number k of nearest neighbors to positive instances along the borderline is to five in all experiments, as was done in SMOTE, Borderline-SMOTE and BOS. We use the RBF kernel SVM as described in Section IV.

We select the over-sampling rates in each experiment according to the imbalance ratio of each dataset in Table I. For example, the imbalance ratio of the Abalone dataset is 1:35 (Minority:Majority). Suppose there is only one minority-class instance in the original training dataset. If we use an over-sampling rate of 100% for the minority-class instances, we will randomly over-sample one minority-class instance, and the new imbalance ratio (Minority:Majority) of the new training dataset after adding the new synthetic minority-class instances will be (2:35). Hence, the over sampling rates of Abalone dataset are varied as follows: 100% (2:35), 500% (6:35), $\cdots$ to the highest rate of 3400% (35:35) to achieve the balanced ratio (1:1).

C. Performance metrics

Using accuracy as a metric to evaluate SVM classification performance is practically useless when coping with significantly imbalanced datasets. This is because if a dataset has an imbalance ratio of 95:5, an SVM classifier that classifies all instances as negative achieves 95% accuracy but is absolutely useless for the task at hand. Several publications use the g-means metric [30], [31], [32] to evaluate classifiers on imbalanced datasets. The g-means metric is defined as follows in [31]:

$$\textit{g-means} = \sqrt{acc^{+} \cdot acc^{-}} \tag{11}$$

where *sensitivity* (5) is acc^{+} and *specificity* (6) is acc^{-}.

Another useful performance metric, the F-measure [33], is the harmonic mean of the precision and recall and is defined as:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{12}$$

We use these two performance metrics, g-means and F-measure, to compare SMOTE, Borderline-SMOTE, BOS, and BOSFS. Tables II through VI show our experimental results.

D. Experimental process

First, we use a holdout method to separate each dataset in Table I into two sets: the training dataset and the test dataset.

Second, in Section IV, we conduct the experiments comparing our BOSFS algorithm to SMOTE [15], Borderline-SMOTE [16] and BOS algorithms [17], [18] using MATLAB with the same over-sampling rate or imbalance ratio listed in Table I for each training dataset. we add the set of new synthetic positive instances $\{p^{+}_{new}\}$ determined by BOSFS, SMOTE, Borderline-SMOTE or BOS to the original training dataset X to form the new training dataset X_{new}, as in Equation (10).

Third, we perform k-fold cross-validation where k = 5 for each dataset with a variety of values for the C parameter in the range of [0,1] and a variety of values for the $Sigma$ (σ) parameter in the range of [0,1] to determine the suitable C and $Sigma$ parameters to achieve the best g-means (Equation 11) and F-measure (Equation 12) using SMOTE, Borderline-SMOTE, BOS, or BOSFS. This step solves SVMs soft-margin problem by using different C and $Sigma$ parameters in the RBF kernel function (Equation 9), as explained above.

Fourth, we train the SVM on the new training dataset X_{new} with the suitable C and $Sigma$ parameters computed in the previous step.

Fifth, we execute SVM with the suitable C and $Sigma$ parameters on the testing dataset previously prepared using the holdout method in the first step to finally obtain the acc^{+}, acc^{-}, g-means (Equation 11), and F-measure (Equation 12) for each experiment. These metrics are recorded and averaged for each imbalance ratio.

To reduce the effect of randomness in the data division and sampling, we perform the first step through the fifth step 10 times for each over-sampling rate and dataset combination. The values of acc^{+}, acc^{-}, g-means, and F-measure are averaged after 10 experiments.

E. Experimental results and evaluation

In Tables II through VI, the values in bold correspond to the best values attained in each experiment. It is clear that BOSFS achieves the greatest number of superior values. For the Glass dataset with imbalance ratios of (3:6) and (4:6), we observe that the BOSFS method is only inferior to SMOTE according to the g-means and F-measure but still better than BOS. The acc^{+} values attained by our BOSFS method are generally better than those of SMOTE, Borderline-SMOTE and BOS, especially for the Abalone, Glass, Spect, and Yeast datasets. We observe that the acc^{+} values of all of the methods can reach 100% for the low degrees of imbalance shown in the Glass and Spect datasets. This means that all of the methods can perform well for low degrees of imbalance, while the BOSFS method outperforms the other methods for higher degrees of imbalance such as Abalone (2:35), Page (2:61), and Yeast (2:28). The values of acc^{+} and acc^{-} attained by SMOTE, Borderline-SMOTE, BOS, and BOSFS vary because of the dependency on the tradeoff between the C^{+} and C^{-} parameter and the imbalance ratio of each dataset. In summary, our BOSFS method accomplishes better g-means and F-measure performance than the SMOTE, Borderline-SMOTE and BOS methods at almost all over-sampling rates and imbalance ratios.

TABLE II
ABALONE DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS,AND BOSFS ALGORITHM USING k = 5 NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)								
	2:35	6:35	11:35	16:35	21:35	26:35	31:35	35:35	Average
SMOTE									
acc^+	31.97	60.18	75.59	82.75	85.91	87.79	89.69	90.39	73.41
acc^-	**98.62**	99.08	97.85	98.15	**98.6**	98.21	98.7	98.89	98.46
g-means	56.08	77.2	85.98	90.08	92.03	92.85	94.09	94.54	84.04
F-measure	44.81	74.42	84.86	89.74	91.95	92.98	94.25	94.69	81.86
Borderline-SMOTE									
acc^+	37.11	63.08	74.16	80.29	85.08	86.98	**93.19**	89.61	76.19
acc^-	95.97	95.33	98.45	98.95	98.41	**98.79**	91.8	**99.58**	97.16
g-means	59.32	77.19	85.44	89.13	91.49	92.69	92.48	94.46	85.28
F-measure	38.94	65.89	84.37	88.67	91.44	92.7	94.04	94.43	81.31
BOS									
acc^+	35.13	61.24	74.79	81.67	85.04	87.75	89.47	90.13	75.66
acc^-	97.61	**99.53**	**99.19**	**99.01**	98.06	98.34	**98.87**	99.42	**98.66**
g-means	58.48	78.06	86.12	89.93	91.31	92.89	94.05	94.66	85.63
F-measure	45.1	75.6	85.14	89.53	91.28	93.02	94.17	94.68	83.51
BOSFS									
acc^+	**47.61**	**72.41**	**81.64**	**86**	**89.09**	**90.93**	91.58	**92.31**	**79.89**
acc^-	97.05	96.16	98.02	98.44	98.31	98.51	97.75	98.86	97.75
g-means	**67.94**	**83.41**	**89.45**	**92.01**	**93.59**	**94.64**	**94.61**	**95.53**	**87.95**
F-measure	**55.54**	**79.22**	**88.7**	**91.79**	**93.64**	**94.8**	**94.99**	**95.75**	**85.53**

TABLE III
PAGE DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS,AND BOSFS ALGORITHM USING k = 5 NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)							
	2:61	11:61	21:61	31:61	41:61	51:61	61:61	Average
SMOTE								
acc^+	94.02	86.55	89.66	92.44	94.72	95.26	96.36	92.72
acc^-	95.94	93.06	96.29	98.44	98.09	96.61	97.08	96.50
g-means	94.92	89.60	92.87	95.39	96.38	95.93	96.71	94.54
F-measure	38.47	78.28	91.48	95.33	96.57	96.49	97.33	84.85
Borderline-SMOTE								
acc^+	88.58	93.54	93.50	95.67	96.94	96.30	97.31	94.09
acc^-	97.55	91.31	91.31	93.60	93.96	96.22	96.55	93.99
g-means	92.84	92.40	92.38	94.62	95.43	96.26	96.93	93.99
F-measure	67.78	81.73	89.74	94.34	95.80	96.86	97.64	87.71
BOS								
acc^+	97.67	92.55	**94.80**	**97.42**	**96.99**	**97.53**	97.43	**96.34**
acc^-	97.35	**96.05**	94.25	98.36	98.47	98.13	97.81	97.20
g-means	97.48	94.26	94.48	**97.88**	97.72	97.82	97.62	96.75
F-measure	68.42	**90.03**	92.31	**97.85**	97.88	98.13	98.10	91.82
BOSFS								
acc^+	**98.85**	**95.35**	93.37	94.55	96.67	96.96	**97.78**	96.13
acc^-	**97.55**	95.56	**97.72**	**98.62**	**98.98**	**99.09**	**99.08**	**98.16**
g-means	**98.19**	**95.41**	**95.52**	96.56	**97.81**	**98.01**	**98.42**	**97.12**
F-measure	**70.63**	90.02	**94.88**	96.51	**97.91**	**98.16**	**98.62**	**92.38**

TABLE IV
GLASS DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS,AND BOSFS ALGORITHM USING k = 5 NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)					
	2:6	3:6	4:6	5:6	6:6	Average
SMOTE						
acc^+	**100.00**	**100.00**	99.60	92.44	94.72	97.35
acc^-	94.60	**97.28**	**99.33**	**98.44**	98.09	**97.55**
g-means	97.24	**98.62**	**99.46**	95.39	96.38	97.42
F-measure	85.29	**97.28**	**99.19**	95.33	96.57	94.73
Borderline-SMOTE						
acc^+	99.00	96.41	96.83	97.72	96.52	97.30
acc^-	94.41	92.69	88.86	89.09	90.15	91.04
g-means	96.67	94.49	92.72	93.26	93.13	94.05
F-measure	86.35	91.56	89.40	92.60	94.27	90.84
BOS						
acc^+	**100.00**	**100.00**	**100.00**	99.00	96.82	99.16
acc^-	94.00	90.57	90.18	90.93	**98.21**	92.78
g-means	96.92	95.15	94.94	94.78	**97.47**	95.85
F-measure	87.57	89.83	92.54	93.09	97.56	92.12
BOSFS						
acc^+	**100.00**	**100.00**	**100.00**	**99.66**	**100.00**	**99.93**
acc^-	**97.06**	94.46	93.55	95.98	94.93	95.20
g-means	**98.51**	97.17	96.71	**97.79**	97.41	**97.52**
F-measure	**94.80**	93.95	95.23	**97.67**	**97.67**	**95.86**

TABLE V
YEAST DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS,AND BOSFS ALGORITHM USING k = 5 NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)							
	2:28	6:28	11:28	16:28	21:28	26:28	28:28	Average
SMOTE								
acc^+	**100.00**	92.66	85.61	89.40	92.49	92.79	93.23	92.31
acc^-	93.35	86.58	93.58	92.28	92.78	91.60	92.51	91.81
g-means	96.61	89.37	89.50	90.82	92.61	92.17	92.86	91.99
F-measure	39.86	66.82	88.47	90.48	93.53	93.85	94.43	81.06
Borderline-SMOTE								
acc^+	96.11	94.25	89.16	89.11	91.18	92.65	93.63	92.30
acc^-	92.77	85.04	91.45	95.57	96.71	97.67	97.99	93.89
g-means	94.34	89.34	90.26	92.25	93.89	95.13	95.78	93.00
F-measure	34.56	63.75	88.11	92.33	94.28	95.53	96.20	80.68
BOS								
acc^+	**100.00**	91.86	92.65	93.82	95.45	95.58	96.07	95.06
acc^-	93.71	94.15	96.56	95.83	95.07	95.04	93.93	94.90
g-means	96.80	92.96	94.58	94.80	95.26	95.31	94.99	94.96
F-measure	55.52	86.59	93.65	94.88	95.85	96.13	96.20	88.40
BOSFS								
acc^+	**100.00**	**95.17**	**96.19**	**96.47**	**97.27**	**97.89**	**97.99**	**97.28**
acc^-	**94.96**	**97.18**	**96.71**	**97.13**	**97.57**	**97.39**	**97.79**	**96.96**
g-means	**97.44**	**96.15**	**96.44**	**96.80**	**97.42**	**97.64**	**97.89**	**97.11**
F-measure	**61.00**	**93.70**	**95.74**	**96.88**	**97.67**	**98.13**	**98.35**	**91.64**

TABLE VI
SPECT DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS, AND BOSFS ALGORITHM USING k = 5 NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)			
	2:4	3:4	4:4	Average
SMOTE				
acc^+	99.33	94.83	90.41	94.86
acc^-	87.35	64.91	51.88	68.05
g-means	93.11	77.60	67.09	79.27
F-measure	90.92	76.47	69.97	79.12
Borderline-SMOTE				
acc^+	**100.00**	94.29	93.18	95.82
acc^-	65.61	53.21	49.49	56.10
g-means	80.52	70.28	67.50	72.77
F-measure	60.54	67.37	67.85	65.25
BOS				
acc^+	**100.00**	98.61	96.18	98.26
acc^-	87.16	82.75	**84.04**	84.65
g-means	93.28	90.18	89.62	91.03
F-measure	90.35	93.01	92.08	91.81
BOSFS				
acc^+	**100.00**	**100.00**	**98.66**	**99.55**
acc^-	**89.99**	**86.77**	83.04	**86.60**
g-means	**94.79**	**93.07**	**90.40**	**92.75**
F-measure	**93.30**	**94.19**	**93.74**	**93.74**

VI. CONCLUSION

The imbalance class data problem has impacted the prediction performance of SVM classifiers. In this paper, we proposed a new over-sampling method called BOSFS that focuses on the k nearest neighbors of the positive instances along the borderline. BOSFS finds these nearest neighbors using the Euclidean distance and kernel function in the feature space, rather than in the input space as is done in SMOTE, Borderline-SMOTE, and BOS. We also introduce a new way to find the density of the nearest negative instances in the feature space along the borderline. We determine the appropriate technique to generate new synthetic instances (i.e., interpolation or extrapolation) by considering the density of negative instances in the feature space. Thus, the BOSFS algorithm achieves superior SVM classification performance in terms of g-means and the F-measure for imbalanced datasets, as shown in Tables II through VI. We conclude that our BOSFS algorithm is better suited than the existing SMOTE, Borderline-SMOTE, and BOS algorithms for effectively determining new positive instances to improve SVM prediction in imbalanced data environments.

REFERENCES

[1] V. N. Vapnik, "The nature of statistical learning theory," 1995.

[2] T. Fawcett and F. J. Provost, "Combining data mining and machine learning for effective user profiling.," in *KDD*, pp. 8–13, 1996.

[3] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine learning*, vol. 30, no. 2-3, pp. 195–215, 1998.

[4] K. J. Ezawa, M. Singh, and S. W. Norton, "Learning goal oriented bayesian networks for telecommunications risk management," in *ICML*, pp. 139–147, 1996.

[5] P. K. Chan and S. J. Stolfo, "Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection.," in *KDD*, vol. 1998, pp. 164–168, 1998.

[6] A. Uyar, A. Bener, H. Ciracy, and M. Bahceci, "Handling the imbalance problem of ivf implantation prediction," *IAENG International Journal of Computer Science*, vol. 37, pp. 164–170, 2010.

[7] N. Poolsawad, C. Kambhampati, and J. Cleland, "Balancing class for performance of classification with a clinical dataset," *Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering 2014, WCE 2014*, 2-4 July, 2014, London, U.K., pp 237-242.

[8] M. Lichman, "UCI machine learning repository," 2013.

[9] K. S. Woods, C. C. Doss, K. W. Bowyer, J. L. Solka, C. E. Priebe, and W. P. Kegelmeyer JR, "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 06, pp. 1417–1436, 1993.

[10] R. Batuwita and V. Palade, "Class imbalance learning methods for support vector machines," *Imbalanced learning: Foundations, algorithms, and applications*, pp. 83–99, 2013.

[11] R. Batuwita and V. Palade, "Efficient resampling methods for training support vector machines with imbalanced datasets," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pp. 1–8, IEEE, 2010.

[12] H. He, E. Garcia, *et al.*, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, 2009.

[13] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 2, pp. 539–550, 2009.

[14] I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of Workshop on Learning from Imbalanced Datasets*, 2003.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, pp. 321–357, 2002.

[16] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *Advances in intelligent computing*, pp. 878–887, Springer, 2005.

[17] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," in *Proc. IEEE SMC Hiroshima Chapter Fifth International Workshop on Computational Intelligence and Applications*, (Hiroshima University, Japan), pp. 24–29, 2009.

[18] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *International Journal of Knowledge Engineering and Soft Data Paradigms*, vol. 3, no. 1, pp. 4–21, 2011.

[19] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[20] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the border: active learning in imbalanced data classification," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 127–136, ACM, 2007.

[21] K. Veropoulos, C. Campbell, N. Cristianini, *et al.*, "Controlling the sensitivity of support vector machines," in *Proceedings of the international joint conference on AI*, pp. 55–60, 1999.

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

[23] G. Wu and E. Y. Chang, "Kba: Kernel boundary alignment considering imbalanced data distribution," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 786–795, 2005.

[24] H. Zhang, "Distance-based classifier via the kernel trick," in *International Journal Software Informatics*, Vol.2, pp. 121–133, 2010.

[25] H.-Y. Wang, "Combination approach of smote and biased-svm for imbalanced datasets," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pp. 228–231, IEEE, 2008.

[26] C.-W. Hsu, C.-C. Chang, C.-J. Lin, *et al.*, "A practical guide to support vector classification," 2003.

[27] R. Herbrich, "Learning kernel classifiers," MIT Press, Cambridge, 2002.

[28] B. Schölkopf and A. J. Smola, "Learning with kernels. 2002.", MIT Press, Cambridge, 2002.

[29] Y. Li, Z. Hu, Y. Cai, and W. Zhang, "Support vector based prototype selection method for nearest neighbor rules," in *Advances in Natural Computation*, pp. 528–535, Springer, 2005.

[30] M. Kubat and R. Holte, "S. matwin,learning when negative example abound," in *Proceedings of the 9th European Conference on Machine Learning, ECML*, vol. 97, 1997.

[31] M. Kubat, S. Matwin, *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *ICML*, vol. 97, pp. 179–186, Nashville, USA, 1997.

[32] G. Wu and E. Y. Chang, “Class-boundary alignment for imbalanced dataset learning,” in *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*, pp. 49–56, 2003.
[33] D. M. Powers, “Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation,” *Journal of Machine Learning Technologies*, vol. 2, pp. 37–63, 2011.