# Minority Oversampling in Kernel Adaptive Subspaces for Class Imbalanced Datasets

Chin-Teng Lin, *Fellow, IEEE*, Tsung-Yu Hsieh, Yu-Ting Liu, Yang-Yin Lin, Chieh-Ning Fang, Yu-Kai Wang, *Member, IEEE*, Gary Yen, *Fellow, IEEE*, Nikhil R. Pal, *Fellow, IEEE*, Chun-Hsiang Chuang, *Member, IEEE*

**Abstract**—The class imbalance problem in machine learning occurs when certain classes are underrepresented relative to the others, leading to a learning bias toward the majority classes. To cope with the skewed class distribution, many learning methods featuring minority oversampling have been proposed, which are proved to be effective. To reduce information loss during feature space projection, this study proposes a novel oversampling algorithm, named minority oversampling in kernel adaptive subspaces (MOKAS), which exploits the invariant feature extraction capability of a kernel version of the adaptive subspace self-organizing maps. The synthetic instances are generated from well-trained subspaces and then their pre-images are reconstructed in the input space. Additionally, these instances characterize nonlinear structures present in the minority class data distribution and help the learning algorithms to counterbalance the skewed class distribution in a desirable manner. Experimental results on both real and synthetic data show that the proposed MOKAS is capable of modeling complex data distribution and outperforms a set of state-of-the-art oversampling algorithms.

**Index Terms**—Class Imbalance, Adaptive Subspace Self-Organizing Maps, Kernel, Minority Oversampling

## 1 Introduction

ONE of the ten challenging problems in data mining research [1] is the imbalance learning problem which has been widely reported in various real-world applications, such as medical diagnosis [2], detection of fraudulent financial activities [3, 4], detection of oil spills [5], brain-computer interface applications [6], and customer churn prediction [7]. The class-imbalance learning problem occurs when highly unequal distribution of data exists among different classes in a learning task [8]. The majority class having a relatively large number of data points can overwhelm the data distribution of the minority class. As a result, the minority class with a relatively small number of data points may be severely underrepresented during the learning process and this may make the machine learning algorithms fail to accurately learn the minority class concepts [9, 10].


- C.-T. Lin, C.-N. Fang, Y.-K. Wang, C.-H. Chuang are with the Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia (e-mail: chintenglin@gmail.com; fang1992119@gmail.com; yukai.wang@uts.edu.au and chchuang@ieee.org)
- T.-Y. Hsieh is with the College of Information Sciences and Technology, University of Pennsylvania, University of Pennsylvania, USA (e-mail: supreme7911@hotmail.com)
- Y.-T. Liu is wih the Department of Multimedia Technology Deveopment, Mediatek Inc., Hsinchu, Taiwan (e-mail: tingting76319@gmail.com)
- Y.-Y. Lin is with the Electronic Systems Research Division, National Chung-Shan Institute of Science and Technology, Taiwan (e-mail: oliver.yylin@gmail.com)
- G. Yen is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA (e-mail: gyen@okstate.edu)
- N. R. Pal is with the Electronics and Communication Sciences Unit, Indian Statistical Institute, India (e-mail: nrpal59@gmail.com)


With great influx of attention devoted to the imbalance learning problem, several strategies have been proposed, which can be roughly divided into two categories: algorithm-level methods and data-level methods [11]. Some of the algorithm-level methods [12-14] use cost-sensitive learning [15], in which the imbalance present in the dataset is counterbalanced by assigning higher cost to misclassification of the minority class instances and lower cost to that of the majority class instances [16]. On the other hand, data-level methods establish class balance through data resampling techniques such as undersampling of the majority class [17], oversampling of the minority class [11, 18-25], or a combination of both [26]. This study lays emphasis on oversampling techniques since these methods do not disregard informative and important instances, which undersampling algorithms may during rejection of majority class instances.

Oversampling algorithms enforce emphasis on the minority class by augmenting it with generated instances. The most straightforward approach is to generate instances in the data space (i.e., the original input space in which the training data are available) such as duplicating the existing minority class instances as much as necessary to balance the dataset or performing operations like rotation and skew to perturb the data [27]. However, this strategy will inevitably lead to over-fitting [28] and cannot significantly improve minority class recognition. To address this problem, several advanced methods have been proposed including interpolation-based [18-20] and structure-preserving approaches [11, 21] that tend to introduce certain variations in the synthetic instances from the existing

instances. In the interpolation-based approach, minority class instances are selected as seed instances in turn, and synthetic instances are generated by interpolating between a seed instance and one of its random nearest minority class neighbors. The interpolation-based methods include synthetic minority over-sampling technique (SMOTE) [18] and its extensions, adaptive synthetic sampling approach (ADASYN) [19], majority weighted minority oversampling technique (MWMOTE) [20], and certainty guided minority oversampling (CGMOS) [29]. In SMOTE, the minority class is over-sampled by taking each minority class instance to generate synthetic instances along the line segments joining any/all of the minority class nearest neighbors. The synthetic instances can be generated in a less application-specific manner by operating in the feature space rather than data space. In ADASYN and MWMOTE, advanced mechanisms to determine the hard-to-learn minority class instances are proposed to improve the classifier learning efficiency. The CGMOS [28] aims at improving the Bayesian classification performance for both majority and minority class by examining the certainty change while adding synthetic instances to the dataset. On the other hand, structure preserving oversampling (SPO) aims at allowing the resulting synthetic instance data to preserve the main structure of the original minority-class instances. For example, Mahalanobis distance-based oversampling technique (MDO) [22] maintains covariance structure of target instances in the minority class and generates synthetic data along with the probability contours of the selected instances. A hybrid oversampling technique called Integrated Oversampling (INOS) [11] is proposed to address imbalanced learning issue by synergistically combining enhanced structure preserving oversampling (ESPO) [11] with interpolation-based oversampling. In this way, INOS can inherit the main covariance structure of the original minority-class instances and at the same time can protect the existing hard-to-classify instances that are close to the class boundary.

To consider the probability distribution of data, several methods have been developed recently. For example, distribution random oversampling (DRO) [23] extends the original vector representation of the input data with additional features that are generated by some stochastic function to oversample the minority class. Synthetic instances are generated exploiting the distributional characteristics of the training data. These additional features are called latent features and the associated feature space as latent space. RACOG and wRACOG [24] both generate instances for the minority class by considering joint probability distribution of data features. The probability distribution of the minority class is learnt using a dependence tree algorithm and the Gibbs sampler is used to generate instances from the distribution. RACOG and wRACOG differ only in the strategy to select instances generated by the Gibbs sampler.

To address nonlinear boundary problem, the absent data generator (ADG) [25] algorithm, which applies kernel Fisher discriminant analysis, ensures that the synthetic instances are close to the border of the majority class and their projections to a lower dimension are near to that of exiting minority points. As opposed to working in the original data space (or input space), mapping data into a (usually high-dimensional) feature space, where linear separation may exist among different classes, and generating synthetic minority class instances in such a feature space are expected to enhance the performance of classifiers. However, the information loss during feature space projection and reconstruction is usually significant. The characteristics of the synthetic instances generated by such a mechanism may differ from that of the input instances considerably. To cope with class imbalance problem in the abovementioned circumstance, this study proposes an innovative oversampling technique using a kernel based adaptive subspace self-organizing map that is expected to model complex data distribution of the minority class while generating synthetic instances. There are two important features of the proposed algorithm. First, the synthetic instances bear significant resemblance to the existing instances, but introduces necessary variation and, therefore, holds great potential in generating synthetic instances. Second, multiple subspaces are exploited to model different characteristics of the input data distribution, that the synthetic instances generated by different subspaces will inherit.

The rest of the paper is organized as follows. Section 2.1 is a revisit of previous works. In Section 2.2, the proposed oversampling algorithm, named minority oversampling in kernel adaptive subspace (MOKAS), is introduced in detail. Experimental design and results are presented in Section 3. In Sections 4 and 5, discussion and conclusions are included, respectively.

## 2 Minority Oversampling in Adaptive Subspaces

### 2.1 Adaptive Subspace Self-Organizing Map

A special type of self-organizing map (SOM) [30], the adaptive subspace self-organizing map (ASSOM) algorithm, proposed by Kohonen *et al.,* [31], [32] consists of different modules where each module learns to recognize invariant patterns that are subjected to simple transformation (each module represents a subspace). If weight vectors can be trained in a way such that they act like invariant feature-filters, then such weight vectors can be used to generate input representation invariant to certain transformations and hence can be very effectively used for pattern recognition.

Let $\boldsymbol{B}_m = (\boldsymbol{b}_{m1}, \boldsymbol{b}_{m2}, \dots, \boldsymbol{b}_{mH})$ be a set of orthonormal basis vectors defining a subspace. Then the projection operator matrix for orthogonal projection of a data point $\mathbf{x}$ on the subspace spanned by $B_{\boldsymbol{m}}$ is $\boldsymbol{P} = \boldsymbol{B}_m \boldsymbol{B}_m^T = \sum_{h=1}^{H} \boldsymbol{b}_{mh} \boldsymbol{b}_{mh}^T$. So the orthogonal projection $\hat{\boldsymbol{x}}$ of $\mathbf{x}$ on the subspace is given by

$$\hat{\boldsymbol{x}} = \boldsymbol{P}\boldsymbol{x} = (\sum_{h=1}^{H} \boldsymbol{b}_{mh} \boldsymbol{b}_{mh}^T)\boldsymbol{x} = \sum_{h=1}^{H} \boldsymbol{b}_{mh} (\boldsymbol{b}_{mh}^T \boldsymbol{x}) = \sum_{h=1}^{H} (\boldsymbol{b}_{mh}^T \boldsymbol{x})\, \boldsymbol{b}_{mh} = \sum_{h=1}^{H} o_{mh} \boldsymbol{b}_{mh}, \quad (1)$$

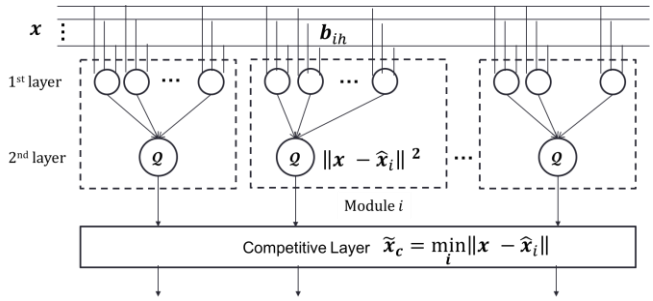


Fig. 1. ASSOM original structure. Each dotted square box distinguishes a processing unit in the ASSOM structure and is called a module. Each module can be regarded as a linear subspace. The structure can be trained to represent transformations, including translation, scaling and rotation.

where

$$o_{mh} = (\boldsymbol{b}_{mh}^T \boldsymbol{x}). \quad (2)$$

The vector x may not exactly lie in the subspace spanned by $B_{\boldsymbol{m}}$ and in that case $\hat{\boldsymbol{x}} \neq \boldsymbol{x}$ and there will be some error in the projected or reconstructed vector $\hat{\boldsymbol{x}}$. The error vector is $\tilde{\boldsymbol{x}} = (\boldsymbol{x} - \hat{\boldsymbol{x}})$ and a good measure of error can be defied as $\|\tilde{\boldsymbol{x}}\|^2 = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$.

An ASSOM [31, 32]. is a neural network that uses (1) and (2) to adaptively find different subspaces, where each subspace characterizes some invariant aspect of the data used to train the network. An ASSOM (Fig. 1) is realized using three layers of neurons: input layer, first layer and the second layer. In the second layer, each neuron represents a module. Each module is connected with a number of neurons in the first layer. Each module represents a preprocessing unit in a special SOM array. Let the m$^{th}$ module be connected with H neurons in the first layer. Each neuron in the first layer is connected with all input nodes. Therefore, a neuron in the first layer is associated with a weight vector in the same dimension as that of the input. Consequently, a module is associated with a set of such weight vectors. For module m, let us denote the weight vector of neuron $h$ in the first layer by $b_{mh}$. We orthonormalize the set weight vectors $\boldsymbol{B}_{\boldsymbol{m}} = (\boldsymbol{b}_{m1}, \boldsymbol{b}_{m2}, \dots, \boldsymbol{b}_{mH})$ associated with the module m. Thus $\boldsymbol{B}_{\boldsymbol{m}}$ forms an orthonormal basis vectors of a linear subspace $\mathcal{L}^{(m)}$. The h$^{th}$ neuron in the first layer that is associated with the m$^{th}$ module computes the inner products of an input instance and its associated basis vector using (2). In other words, it computes the similarity between the basis and the input. The second-layer neurons $(m)$ form quadratic functions of the first-layer neurons' outputs. As mentioned earlier, a given input pattern $\boldsymbol{x}$ may not exactly lie in any of the linear subspaces. The quadratic neurons compete among themselves to find the best matching subspace, i.e., winner, $\boldsymbol{c} = \text{argmin}_m \|\boldsymbol{x} - \hat{\boldsymbol{x}}_m\|^2$. The network then updates the basis vectors associated with the winner as well as its topological (spatial) neighbors so that the associated subspaces capture some invariant characteristics of the data [31, 32]. In case of a SOM, the weight vectors quantize the data, where each weght vector represent a set of similar data points (cluster). But in case of an ASSOM, each quadratic neuron represents a subspace which represents a subset of the data (i.e., a subset of the data that lies on the subspace). Thus, each suspace represents some invariant characteristics of a subset of the data. It is therefore, reasonable to assume that if we can generate synthetic instances from each of these subspaces, these instances will follow the distribution of the original data. We shall exploit this property to deal with classification with imbalanced data.

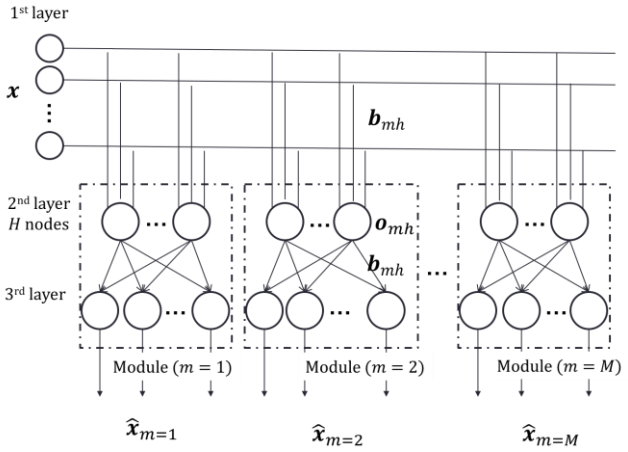


Fig. 2. Three-layer structure of the minority oversampling in adaptive subspace.

Since the set of basis vectors associated with a module is required twice, once to compute (1) and then to compute (2), to make an efficient network implementation, a quadratic neuron representing a module in Fig. 1 has been expanded in Fig. 2 to have another layer of neurons so that a copy of the basis vectors is available for computation of (2).

Note that, if the number of first-layer nodes is smaller than the dimension of input instances, the first-layer neurons will not form a full-rank orthonormal basis. As a result, the reconstructed vector from the second-layer neurons will not be identical to the input vector. However, with proper training of the subspaces, this variation becomes beneficial and results in synthetic instances that bear strong resemblance to the existing instances. This mechanism introduces adequate and apropriate variation for generation of samples implicitly from the original data distribution. Therefore, this holds a great potential in generating synthetic instances for minority oversampling. Another major feature of the minority oversampling in adaptive subspaces is that multiple subspaces are exploited to model different characteristics of data distribution. This suggests that the synthetic instances generated by each subspace will inherit different characteristics of the input data.

## 2.2 Minority Oversampling in Kernel Adaptive Subspaces

In this section, the proposed oversampling method, named minority oversampling in kernel adaptive subspace (MOKAS), is introduced. First, we discuss the kernel version of ASSOM proposed by Kawano *et al.*, [33], which enhances the capability of modeling complex data distribution with invariant features. Second, we explain generation of synthetic instances by projecting existing minority instances onto these kernel adaptive subspaces to cope with the class imbalance problem. In this method, the obtained synthetic instances that preserve characteristics of the existing instances and yet possess certain variations might improve the classifier learning efficiency and avoid the over-fitting problem. Then, an inverse transformation, i.e., the pre-image process [34], is performed to map the synthetic data in the feature space back to the original input space.

The system flowchart in Fig. 3 depicts that there are five major phases in the proposed algorithm. In the first phase, certain system parameters are initialized according to the target task. In the second phase, input instances are mapped to a feature space and projected onto different subspaces of the feature space. Next, weight vectors of each subspace are trained using a competitive learning scheme [35] in the third phase. In the fourth phase, the Gram-Schmidt process is applied in the feature space to generate orthonormal bases. Training epoch is repeated until a pre-defined termination criterion is reached. Finally, in the fifth phase, synthetic instances are generated by projecting existing instances onto the well-trained feature subspaces and then performing an inverse transformation of the synthetic instances back to the input space. Thus, each training instance can be used to generate as many synthetic instances as the number of subspaces. Details will be discussed in the following subsections.

### 2.2.1 Initialization

Two major parameters, i.e., the number of modules $M$ (each module represents a subspace) and the number of orthogonal basis $H$, have to be determined for initialization. The number of modules $M$ reflecting the imbalance ratio is defined as

$$M = \text{round}\left(\frac{N_{maj}}{N_{min}}\right) - 1, \tag{3}$$

where $N_{min}$ and $N_{maj}$ are the numbers of instances in the minority and majority classes, respectively. The value is defined such that the dataset after oversampling will make the imbalance ratio approach one. That is, for each existing instance, we can generate $M$ synthetic instances. The value of $H$ can be randomly selected as long as the value is not greater than $N_{min}$ for initialization. This is because the basis vectors are represented by the minority class instance vectors and therefore we can guarantee orthonormal basis only if the number of basis vectors is not greater than $N_{min}$. The number of basis vectors has a close connection to the similarity between original data and synthetic data. The larger the value, the more similar synthetic data are expected to be to the original data. When two parameters have been determined, the subspaces can be initialized as

$$\boldsymbol{V}_m = [\boldsymbol{b}_{m1}, \boldsymbol{b}_{m2}, \dots, \boldsymbol{b}_{mH}], \tag{4}$$

where $m = 1, 2, \dots, M$ and each basis vector in the feature space can be represented as [33]

$$\boldsymbol{b}_{mh} = \sum_{n=1}^{N_{min}} \alpha_{mhn}\, \phi(\boldsymbol{x}_n), \tag{5}$$

where $h = 1,2, \dots, H$. Here $\boldsymbol{b}_{mh}$ is the $h^{\text{th}}$ basis vector of the

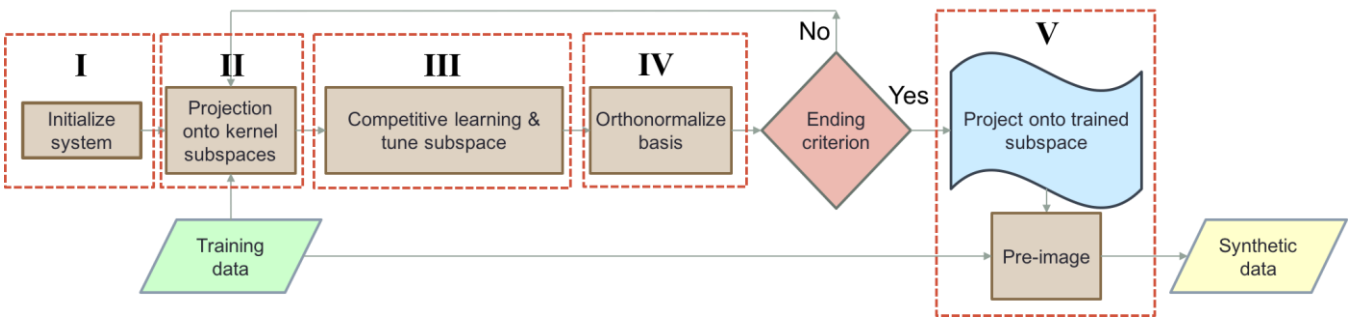


Fig. 3. Procedure of the proposed MOKAS algorithm. The proposed algorithm is composed of five major parts: I) Initialization, II) Projection on subspaces in feature space, III) Competitive learning and subspace training, IV) Basis vectors orthonormalization, V) Synthetic instance generation.

$m^{\text{th}}$ subspace in the feature space. $\phi(\boldsymbol{x}_n)$ is the feature space representation of the $n^{\text{th}}$ minority class instance, and $\alpha$ is a subspace parameter. Here $b_{mh}$ is written as a liner combination of $\phi(\boldsymbol{x}_n)$.

### 2.2.2 Transformation and projection

As stated earlier, $\boldsymbol{b}_{mh}$s are the basis vectors in the $m^{\text{th}}$ subspace in the feature space. The projection of input $\phi(\boldsymbol{x})$ on the feature subspace can be derived by

$$\hat{\phi}_m(\boldsymbol{x}) = \sum_{h=1}^{H} \left( \boldsymbol{b}_{mh}{}^{\mathrm{T}} \phi(\boldsymbol{x}) \right) \boldsymbol{b}_{mh}. \tag{6}$$

which can then be rewritten as

$$\hat{\phi}_m(\boldsymbol{x}) = \sum_{h=1}^{H} \left( \sum_{n=1}^{N_{min}} \alpha_{mhn} \phi^{\mathrm{T}}(\boldsymbol{x}_n) \right) \phi(\boldsymbol{x}) \sum_{j=1}^{N_{min}} \alpha_{mhj}\, \phi(\boldsymbol{x}_j)$$

$$= \sum_{h=1}^{H} \left[ \left( \sum_{n=1}^{N_{min}} \alpha_{mhn} \kappa(\boldsymbol{x}_n, \boldsymbol{x}) \right) \cdot \sum_{j=1}^{N_{min}} \alpha_{mhj}\, \phi(\boldsymbol{x}_j) \right]. \tag{7}$$

Thus, the projection length can be evaluated by

$$\left\| \hat{\phi}_m(\boldsymbol{x}) \right\|^2 = \hat{\phi}_m(\boldsymbol{x})^T \hat{\phi}_m(\boldsymbol{x})$$

$$= \left( \sum_{h=1}^{H} \left( \boldsymbol{b}_{mh}{}^{\mathrm{T}} \phi(\boldsymbol{x}) \right) \boldsymbol{b}_{mh} \right)^{\mathrm{T}} \left( \sum_{h=1}^{H} \left( \boldsymbol{b}_{mh}{}^{\mathrm{T}} \phi(\boldsymbol{x}) \right) \boldsymbol{b}_{mh} \right)$$

$$= \sum_{h=1}^{H} \left( \boldsymbol{b}_{mh}{}^{\mathrm{T}} \phi(\boldsymbol{x}) \right)^2$$

(As $\boldsymbol{b}_{mh}; h = 1, \ldots, H$ form a set of orthonormal basis vectors.)

$$= \sum_{h=1}^{H} \left( \sum_{n=1}^{N_{min}} \alpha_{mhn} \phi^{\mathrm{T}}(\boldsymbol{x}_n) \phi(\boldsymbol{x}) \right)^2$$

$$= \sum_{h=1}^{H} \left( \sum_{n=1}^{N_{min}} \alpha_{mhn} \kappa(\boldsymbol{x}_n, \boldsymbol{x}) \right)^2. \tag{8}$$

Note that, the above equation is solely dependent on the kernel function. Kawano *et al.,* [33] call Eq. (8) as the response of the $m^{th}$ neuron. One can easily select from a wide variety of commonly used kernel functions, such as linear, polynomial, sigmoid, or, radial basis function (RBF) kernel, depending on the target application to yield useful results. Here we shall use the RFB kernel.

### 2.2.3 Competitive learning

Using a competitive learning algorithm, the ASSOM finds the basis vectors of the subspaces by minimizing the expected weighted squared projection error. This is equivalent to maximizing the expected weighted squared projection lengths. The output of a module is invariant to limited linear transformation that occurs within the subspace [31]. Here each module learns to represent a different subset of the data that are invariant to some linear transformation. In other words, a module captures some invariant characteristics of a subset of the data. The output of a module (quadratic neuron) can be viewed as the degree of match between the input and the subspace.

In case of a competitive learning environment like SOM or for a clustering algorithm, a prototype represents a set of points (i.e., a cluster) and each such cluster represents a different substructure present in the data. Similarly, here also, different subspaces model different subsets of the data. In case of clustering, for every data point there is a best representative cluster, here too for a data point there is a best matching subspace. The set of points, that best matches a subspace, enjoys some invariant characteristics

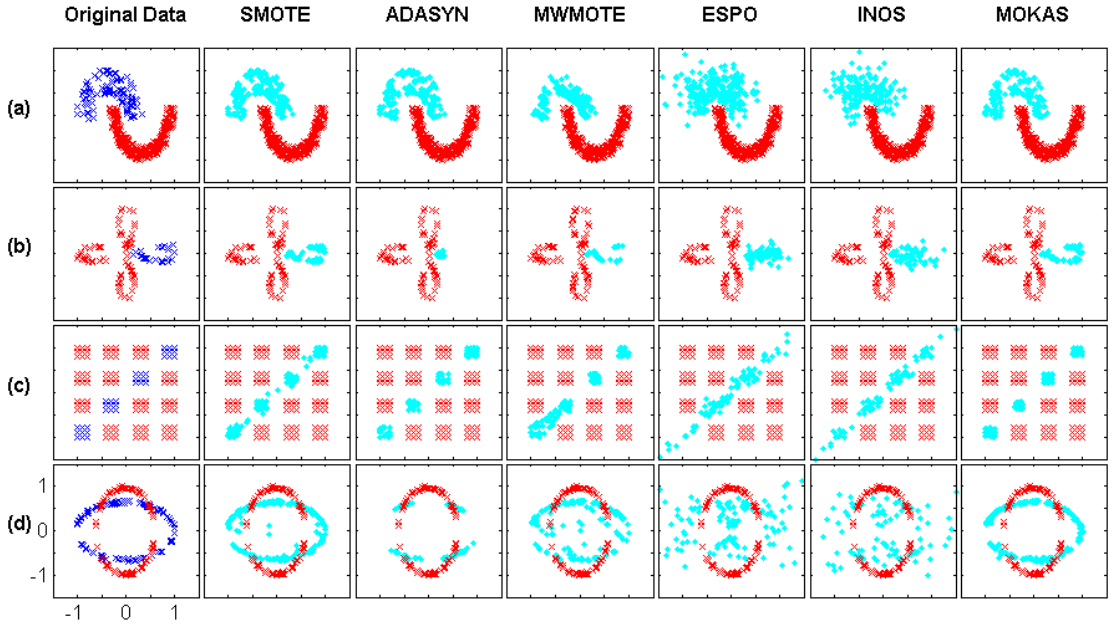


Fig. 4. Scatter plots of the original data and synthetic data of (a) Half ring, dataset, (b) Petals dataset, (c) Regular dataset, and (d) Saturn dataset generated by SMOTE, ADASYN, MWMOTE, ESPO, INOS, and MOKAS. The cyan dots represent synthetic minority class instances, blue crosses represent existing minority class instances and red crosses represent existing majority class instances.

TABLE 1
UTILITY OF THE SYNTHETIC DATA MEASURED BY THE ACCURACY OF A 1-NN CLASSIFIER TRAINED WITH THE ORIGINAL DATA

| | SMOTE | ADASYN | MWMOTE | ESPO | INOS | MOKAS |
|---|---|---|---|---|---|---|
| Half ring | 100.0 | 100.0 | 100.0 | 98.3 | 98.1 | 100.0 |
| Petals | 100.0 | 100.0 | 100.0 | 98.8 | 98.8 | 100.0 |
| Regular | 98.0 | 78.0 | 94.8 | 76.0 | 76.3 | 100.0 |
| Saturn | 98.2 | 96.2 | 96.8 | 86.0 | 87.7 | 98.4 |

represented by the subspace. When we do the training in a feature space (after projecting the data to a high dimensional space), the same is true in the feature space. In this case, an additional advantage is that when the subset of data corresponding to a module is projected back to the input space, the invariance is likely to be with respect to some nonlinear characteristics.

The minimum length of an orthogonal projection of a vector on a subspace is zero when the vector does not lie on the subspace and the maximum length of the projection is the length of the vector itself, and it happens when the vector lies exactly on the subspace. In all other cases, the length of the projection will be between zero and the length of the vector. Hence, larger the length of the projection, the more similar the projected vector to the input vector. Consequently, the winner module for every training iteration can be identified according to the following criterion:

$$c = \text{argmax}_{m=1,..,M} \left\| \hat{\phi}_m(\boldsymbol{x}) \right\|^2. \quad (9)$$

After the winner module is identified, the following neighborhood function with respect to the winner module $c$ is applied to evaluate the relationship between the winner module, $c$, and any other module, $m$,

$$G_m^c = \exp\left( -\frac{\left( \hat{\phi}_m(\boldsymbol{x}) - \hat{\phi}_c(\boldsymbol{x}) \right)^2}{2\sigma^2} \right), \quad (10)$$

where $m = 1, 2, .., M$. For the conventional ASSOM, the neighborhood function is defined such that a module closer to the winner module will result in a higher value and vice versa. This is realized considering a neighborhood function on the ASSOM array. This helps to realize the toplogy preservation property, which is important for SOM. However, in our application, this property is not relevant. There is no utility of having two subsaces associated with two neighboring neurons to be similar. On the contrary, we want each subspace to model some invariant structure of the data as good as possible. Hence, use of neighborhood function in the feature space is more useful. For this we use (10) to define the strength of association between the winner module c and some other mdule $m$. The learning momentum is thus dissipated according to the neighborhood relation in the feature space, in which the winner module receives the largest momentum and as the distance of a neighbor increases, the learning strength decreases.

Following the same strategy as done in [33] the learning objective function to be maximized is defined as

$$D(\boldsymbol{x}) = \sum_{m=1}^{M} G_m^c \sum_{h=1}^{H} \left( \sum_{n=1}^{N_{min}} \alpha_{mhn} \kappa(\boldsymbol{x}_n, \boldsymbol{x}) \right)^2. \quad (11)$$

The objective function states that the weighted summation of the projection length on the subspaces should be maximized, so that the projected vectors are as similar as possible to the original data distribution. This is using search. Taking the derivative of the objective function with respect to the subspace parameter $\alpha$ yields

$$\frac{\partial D(x)}{\partial \alpha_{mhn}} = 2 \cdot G_m^c \left( \sum_{a=1}^{N_{min}} \alpha_{mha} \kappa(\boldsymbol{x}_a, \boldsymbol{x}) \right) \kappa(\boldsymbol{x}_n, \boldsymbol{x}). \quad (12)$$

Thus the subspace parameters are updated by

$$\alpha_{mhn}{}^{(t+1)} = \alpha_{mhn}{}^{(t)} + \lambda(t) \frac{\partial D(x)}{\partial \alpha_{mhn}{}^{(t)}}, \quad (13)$$

where the superscript $(t)$ denotes training iteration and $\lambda(t)$ is a learning-rate factor that diminishes with $t$.

### 2.2.4 Orthonormalization

After updating the subspace parameters $\alpha$, the Gram-Schmidt process [36] is performed in the feature space to ensure that the basis vectors are orthonormalized. Consider a basis $B = [b_1, b_2, \ldots, b_M]$, where $b_h = \sum_{n=1}^{N_{min}} \alpha_{hn} \phi(x_n)$, and $a_h = \left[ \alpha_{h1}, \alpha_{h2}, \ldots, \alpha_{hN_{min}} \right]^T$. An orthonormal basis $[\bar{a}_1', \bar{a}_2', \ldots, \bar{a}_H']$ can be found using Gram-Schmidt process, where

$$\bar{a}'_h = \begin{pmatrix} \frac{\alpha_{h1}}{\sqrt{{a'_h}^{\mathrm{T}} K a'_h}} \\ \vdots \\ \frac{\alpha_{hN_{min}}}{\sqrt{{a'_h}^{\mathrm{T}} K a'_h}} \end{pmatrix}, \quad (14)$$

$h = 1,2, \ldots, H$, and $K$ is the Gram matrix.

### 2.2.5 Inverse transformation

To this point, we are able to train the subspaces so that they approximate the distribution of the original data in the feature space, and are able to generate an orthonormal basis for each subspace. Therefore, the synthetic data can now be generated by projecting the original data onto the trained subspaces in the feature space as

$$\hat{\phi}_m(\boldsymbol{x}) = \sum_{h=1}^{H} \left( \boldsymbol{b}_{mh}^{\mathrm{T}} \phi(\boldsymbol{x}) \right) \boldsymbol{b}_{mh}. \quad (15)$$

However, the formula in (15) only provides representation of the synthetic instances in the feature space. An inverse transformation, i.e., a pre-image process [34], that exploits the concept of multi-dimensional scaling [37] is used to reconstruct the input space representation of the synthetic instances. The distances with neighbors are the most important in determining the location of synthetic data points. To preserve the local neighborhood structure, this pre-image method directly finds the location of the pre-image based on distance constraints, in which the $n$-neighbors are used in locating the pre-image. This method involves only linear algebra and does not suffer from numerical instabilities or the local minimum problem [34].

## 3 Experimental Design and Results

This section compares the performance of the proposed minority oversampling in kernel adaptive subspaces (using an RBF kernel function with a width parameter gamma) with that of five selected state-of-the-art algorithms including interpolation-based and structure-preserving methods from two perspectives. First, the proposed MOKAS is compared with other state-of-the-art oversampling algorithms by visualizing the data distribution with scatter plots. Additionally, the similarity between original data and synthetic data is examined by the nearest neighbor method. Second, to further assess the quality of the samples generated by MOKAS, we have made an extensive comparison of classfier performance using the data generated by different state-of-the-art oversampling methods on ten real-word benchmark imbalanced datasets from the UCI machine learning repository [38] and one electroencephalographic (EEG) datasets [39]. For interpolation-based methods, SMOTE, ADASYN, and MWMOTE are selected. On the other hand, ESPO and INOS are selected to be representatives of the structure-preserving group.

TABLE 2
THE DATASETS USED FOR COMPARING THE CLASSIFICATION PERFORMANCE ARE BENCHMARK IMBALANCED LEARNING DATASETS AND ARE OBTAINED FROM THE UCI MACHINE LEARNING DEPOSITORY AND AN EEG EXPERIMENT. MINOR MODIFICATIONS WERE MADE ON THE DATASETS IN ORDER TO FIT OUR RESEARCH INTEREST.

| Data set name | # of total instances | # of attributes | Minority class | Majority class | # of minority instances | # of majority instances | Imbalance ratio |
|---|---|---|---|---|---|---|---|
| Abalone | 731 | 7 | Class of '18' | Class of '9' | 42 | 689 | 16.4 |
| Breast cancer | 683 | 9 | Class of 'malignant' | Class of 'benign' | 239 | 444 | 1.9 |
| Ecoli | 336 | 7 | Class of 'im' | All other classes | 77 | 259 | 3.4 |
| Glass | 214 | 9 | Class of '5,6,7' | All other classes | 51 | 163 | 3.2 |
| Pima | 768 | 8 | Class of '1' | Class of '0' | 268 | 500 | 1.9 |
| Vehicle | 846 | 18 | Class of 'van' | All other classes | 199 | 647 | 3.3 |
| Yeast | 1484 | 8 | Class of 'ME3', 'ME2', 'EXC', 'VAC', 'POX', 'ERL' | All other classes | 304 | 1180 | 3.9 |
| Libras | 360 | 90 | Class of '1' | All other classes | 24 | 336 | 14.0 |
| Ozone | 1848 | 72 | Class of '1' | Class of '0' | 57 | 1791 | 31.4 |
| Letter | 20000 | 16 | Class of 'Z' | All other classes | 734 | 19266 | 26.2 |
| VEP (EEG; 14 subjects) | ~264 | 10 | Class of 'enemy combatant (target)' | Class of 'US soldier (non-target)' | ~33 | ~231 | ~7.0 |

TABLE 3
ASSESSMENT METRICS ADAPTED TO EVALUATE CLASSIFICATION PERFORMANCE IN IMBALANCE LEARNING TASKS.

| | | **Intrinsic class** | | |
|---|---|---|---|---|
| | **Total Population (T)** | **Positive** | **Negative** | |
| **Test Outcome** | Positive | True Positive (TP) | False Positive (FP) | *Precision* $\frac{TP}{TP+FP}$ |
| | Negative | False Negative (FN) | True Negative (TN) | |
| | | *Recall* $\frac{TP}{TP+FN}$ | *Specificity* $\frac{TN}{FP+TN}$ | *F-value* $\frac{2 \times recall \times precision}{recall + precision}$ |
| | | | | *G-mean* $\sqrt{recall \times specificity}$ |

### 3.1 Data Distribution

To test the performance of the proposed oversampling algorithm, MOKAS, and the selected state-of-the-art algorithms, 2D datasets named *Half ring*, *Petals*, *Regular* and *Saturn* (as shown in the left panel of Fig. 4) [40] were chosen to consider non-Gaussian and arbitrary shaped data distributions. Note that each algorithm was required to generate $3 \times N_{min}$ synthetic instances for a predefined class. All the instances generated by different oversampling algorithms are visualized with 2D scatter plots for comparison. As shown in Fig. 4, red crosses symbolize majority class instances, blue crosses stand for minority class instances, and cyan dots represent synthetic minority class instances generated by oversampling algorithms. The effectiveness of the synthetic data is measured by the classification accuracy of a nearest neighbor (1-NN) classifier (as shown in Table 1), where the 1-NN classifier is trained with the original data (two-class: minority class vs. majority class) and is tested using the synthetic data generated by MOKAS.

In *Half ring* (Fig. 4(a)) and *Petals* (Fig. 4(b)) datasets, we observe that the synthetic instances generated by MWMOTE are more densely located near the class boundary. In Saturn dataset (Fig. 4(d)), some of the synthetic instances (generated by MWMOTE) are placed at erroneous positions. Originally, existing minority class instances are located on a horizontal ellipse. However, some of the synthetic instances are found around the central region. On the other hand, synthetic instances generated by ADASYN, ESPO, and INOS for *Half ring* (Fig. 4(a)), *Petals* (Fig. 4(b)), and *Saturn* (Fig. 4(d)) datasets are distributed with lower similarity to the original minority class compared to the instances generated by SMOTE and MWMOTE. Also, notice that for *Half ring* and *Petals* datasets, the boundaries expand considerably after synthetic instances are appended to the minority class.

TABLE 4
CLASSIFICATION RESULTS. THE RESULTS USING NEURAL NETWORK ARE ORGANIZED IN THE LEFT COLUMN AND THAT USING SVM (WITH BALANCED MODE* AND SVMLIGHT†) ARE IN THE RIGHT COLUMN. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD FACE.

| | Method | NN | | | | SVM | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Rec. | Pre. | F | G | Rec. | Pre. | F | G |
| Abalone | | | | | | 0.202 | 0.599 | 0.293 | 0.418 |
| | w/o | 0.401 | 0.414 | 0.394 | 0.606 | **0.769*** | 0.370* | **0.500*** | **0.840*** |
| | | | | | | 0.143† | **0.750†** | 0.240† | 0.327† |
| | SMOTE | **0.765** | 0.355 | 0.483 | **0.832** | 0.567 | 0.353 | 0.431 | 0.716 |
| | ADASYN | 0.511 | 0.345 | 0.407 | 0.687 | 0.422 | 0.336 | 0.369 | 0.626 |
| | MWMOTE | 0.683 | 0.426 | 0.518 | 0.797 | 0.447 | 0.440 | 0.434 | 0.652 |
| | ESPO | 0.634 | 0.299 | 0.404 | 0.753 | 0.655 | 0.363 | 0.458 | 0.773 |
| | INOS | 0.666 | 0.306 | 0.416 | 0.771 | 0.587 | 0.298 | 0.390 | 0.725 |
| | MOKAS | 0.734 | **0.447** | **0.546** | 0.827 | 0.568 | 0.377 | 0.446 | 0.721 |
| Breast cancer | | | | | | 0.961 | 0.945 | 0.952 | 0.965 |
| | w/o | 0.862 | 0.937 | 0.896 | 0.913 | 0.986* | 0.932* | 0.958* | 0.958* |
| | | | | | | 0.967† | **0.947†** | 0.957† | 0.957† |
| | SMOTE | 0.940 | 0.934 | 0.937 | 0.952 | 0.962 | 0.932 | 0.947 | 0.962 |
| | ADASYN | 0.902 | 0.936 | 0.918 | 0.933 | 0.984 | 0.931 | 0.956 | 0.972 |
| | MWMOTE | 0.949 | 0.929 | 0.938 | 0.954 | **0.988** | 0.935 | **0.961** | 0.975 |
| | ESPO | **0.969** | 0.936 | **0.952** | **0.966** | **0.988** | 0.928 | 0.957 | 0.973 |
| | INOS | 0.947 | 0.917 | 0.931 | 0.950 | 0.983 | 0.931 | 0.956 | 0.971 |
| | MOKAS | 0.966 | **0.938** | 0.951 | 0.965 | **0.988** | 0.936 | **0.961** | **0.976** |
| Ecoli | | | | | | 0.761 | **0.873** | **0.811** | 0.857 |
| | w/o | 0.714 | 0.645 | 0.674 | 0.791 | 0.773* | 0.586* | 0.667* | 0.673* |
| | | | | | | 0.779† | 0.811† | 0.795† | 0.795† |
| | SMOTE | **0.864** | 0.664 | 0.746 | **0.863** | 0.776 | 0.683 | 0.722 | 0.829 |
| | ADASYN | 0.734 | 0.655 | 0.689 | 0.803 | 0.773 | 0.726 | 0.744 | 0.837 |
| | MWMOTE | 0.818 | **0.716** | **0.761** | 0.858 | 0.843 | 0.727 | 0.775 | 0.869 |
| | ESPO | 0.863 | 0.608 | 0.710 | 0.846 | **0.925** | 0.631 | 0.747 | 0.878 |
| | INOS | 0.818 | 0.654 | 0.722 | 0.841 | 0.892 | 0.681 | 0.765 | 0.877 |
| | MOKAS | 0.845 | 0.658 | 0.736 | 0.854 | 0.876 | 0.716 | 0.784 | **0.883** |
| Glass | | | | | | 0.789 | 0.840 | 0.799 | 0.860 |
| | w/o | 0.817 | 0.800 | 0.800 | 0.870 | 0.667* | 0.800* | 0.727* | 0.730* |
| | | | | | | 0.882† | **0.918†** | 0.900† | 0.900† |
| | SMOTE | 0.863 | 0.842 | 0.849 | 0.903 | 0.876 | 0.850 | 0.857 | 0.910 |
| | ADASYN | 0.790 | 0.857 | 0.817 | 0.867 | 0.890 | 0.835 | 0.855 | 0.914 |
| | MWMOTE | 0.871 | 0.843 | 0.850 | 0.906 | 0.885 | 0.874 | 0.875 | 0.919 |
| | ESPO | 0.859 | **0.889** | **0.867** | 0.908 | 0.862 | 0.843 | 0.850 | 0.904 |
| | INOS | 0.875 | 0.835 | 0.851 | 0.908 | **0.986** | 0.876 | **0.927** | **0.971** |
| | MOKAS | **0.914** | 0.828 | 0.862 | **0.923** | 0.945 | 0.873 | 0.905 | 0.950 |
| Pima | | | | | | 0.536 | **0.691** | 0.602 | 0.681 |
| | w/o | 0.556 | **0.604** | 0.577 | 0.667 | 0.685* | 0.625* | 0.654* | 0.654* |
| | | | | | | 0.571† | 0.662† | 0.613† | 0.615† |
| | SMOTE | **0.739** | 0.596 | 0.657 | 0.730 | 0.613 | 0.573 | 0.590 | 0.679 |
| | ADASYN | 0.634 | 0.551 | 0.589 | 0.677 | 0.560 | 0.555 | 0.553 | 0.648 |
| | MWMOTE | 0.708 | 0.603 | 0.649 | 0.726 | 0.643 | 0.607 | 0.623 | 0.706 |
| | ESPO | 0.677 | 0.568 | 0.617 | 0.700 | **0.735** | 0.616 | **0.667** | **0.740** |
| | INOS | 0.663 | 0.573 | 0.613 | 0.697 | 0.708 | 0.608 | 0.652 | 0.728 |
| | MOKAS | 0.734 | 0.601 | **0.659** | **0.734** | 0.652 | 0.609 | 0.626 | 0.708 |
| Vehicle | | | | | | 0.952 | 0.939 | **0.946** | 0.966 |
| | w/o | 0.898 | 0.904 | 0.900 | 0.933 | 0.983* | 0.862* | 0.918* | 0.968* |
| | | | | | | **1.000†** | 0.765† | 0.867† | 0.875† |
| | SMOTE | 0.949 | 0.907 | 0.926 | 0.959 | 0.960 | 0.934 | **0.946** | 0.969 |
| | ADASYN | 0.935 | 0.899 | 0.917 | 0.951 | 0.935 | **0.941** | 0.935 | 0.957 |
| | MWMOTE | 0.962 | **0.920** | **0.940** | **0.968** | 0.962 | 0.906 | 0.931 | 0.964 |
| | ESPO | 0.967 | 0.849 | 0.903 | 0.956 | 0.991 | 0.872 | 0.927 | **0.973** |
| | INOS | 0.965 | 0.849 | 0.903 | 0.955 | 0.991 | 0.866 | 0.924 | 0.971 |
| | MOKAS | **0.979** | 0.879 | 0.926 | **0.968** | 0.973 | 0.913 | 0.941 | 0.972 |
| Yeast | | | | | | 0.670 | **0.810** | **0.733** | 0.801 |
| | w/o | 0.674 | **0.730** | 0.700 | 0.793 | **0.847*** | 0.537* | 0.658* | 0.837* |
| | | | | | | 0.641† | 0.809† | 0.716† | 0.720† |
| | SMOTE | 0.806 | 0.636 | 0.710 | 0.842 | 0.714 | 0.678 | 0.695 | 0.807 |
| | ADASYN | 0.782 | 0.558 | 0.650 | 0.809 | 0.658 | 0.591 | 0.621 | 0.761 |
| | MWMOTE | 0.809 | 0.641 | 0.714 | **0.845** | 0.746 | 0.646 | 0.690 | 0.815 |
| | ESPO | 0.775 | 0.649 | 0.706 | 0.831 | 0.804 | 0.624 | 0.702 | 0.838 |
| | INOS | **0.828** | 0.603 | 0.696 | 0.842 | 0.817 | 0.622 | 0.705 | **0.843** |
| | MOKAS | 0.798 | 0.653 | **0.717** | 0.842 | 0.720 | 0.723 | 0.719 | 0.816 |
| Libras | | | | | | 0.621 | **0.988** | 0.744 | 0.779 |
| | w/o | 0.609 | 0.917 | 0.718 | 0.769 | 0.737* | 0.980* | 0.836* | 0.855* |
| | | | | | | 0.686† | 0.986† | 0.786† | 0.817† |
| | SMOTE | 0.629 | 0.755 | 0.662 | 0.770 | 0.783 | 0.818 | 0.788 | 0.876 |
| | ADASYN | 0.629 | **0.934** | **0.731** | 0.779 | 0.816 | 0.831 | 0.811 | 0.893 |
| | MWMOTE | 0.634 | 0.819 | 0.682 | 0.767 | 0.740 | 0.848 | 0.772 | 0.846 |
| | ESPO | **0.751** | 0.631 | 0.662 | **0.844** | 0.783 | 0.750 | 0.753 | 0.869 |
| | INOS | 0.714 | 0.727 | 0.690 | 0.816 | 0.829 | 0.633 | 0.682 | 0.882 |
| | MOKAS | 0.674 | 0.815 | 0.708 | 0.803 | **0.849** | 0.855 | **0.836** | **0.912** |

TABLE 5
CLASSIFICATION RESULTS OF DATASETS WITH HIGH IMBALANCE RATIO (>20) AND VEP DATASET. THE RESULTS USING NEURAL NETWORK ARE ORGANIZED IN THE LEFT COLUMN AND THAT USING SVM (WITH BALANCED MODE* AND SVMLIGHT†) ARE IN THE RIGHT COLUMN. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD FACE.

| | Method | NN | | | | | | SVM | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | REC. | PRE. | F | G | AUC | ACC | REC. | PRE. | F | G | AUC | ACC |
| Ozone | w/o | 0.035 | 0.089 | 0.047 | 0.094 | **0.840** | **0.967** | 0.197<br>0.287*<br>0.181† | 0.25<br>0.186*<br>0.217† | 0.209<br>0.223*<br>0.212† | 0.421<br>0.516*<br>0.398† | 0.860<br>0.845*<br>0.843† | 0.956<br>0.939*<br>0.955† |
| | SMOTE | 0.360 | 0.173 | 0.228 | 0.572 | 0.698 | 0.925 | 0.238 | 0.259 | 0.243 | 0.452 | 0.845 | 0.957 |
| | ADASYN | 0.358 | 0.172 | 0.228 | 0.570 | 0.707 | 0.928 | 0.199 | 0.283 | 0.225 | 0.434 | 0.884 | **0.958** |
| | MWMOTE | 0.362 | 0.173 | 0.228 | 0.566 | 0.730 | 0.928 | 0.311 | **0.288** | 0.293 | 0.54 | 0.882 | 0.954 |
| | ESPO | **0.499** | 0.139 | 0.214 | **0.659** | 0.779 | 0.889 | 0.327 | 0.167 | 0.219 | 0.542 | 0.847 | 0.931 |
| | INOS | 0.482 | 0.149 | 0.227 | 0.657 | 0.793 | 0.901 | 0.235 | 0.169 | 0.223 | 0.459 | **0.899** | 0.945 |
| | MOKAS | 0.366 | **0.177** | **0.235** | 0.572 | 0.746 | 0.924 | **0.347** | 0.287 | **0.311** | **0.576** | 0.856 | 0.951 |
| Letter | w/o | 0.8729 | **0.936** | **0.9093** | 0.9330 | 0.996 | **0.993** | 0.9691<br>0.6364*<br>0.9560† | **0.9831**<br>0.9204*<br>0.9776† | **0.9760**<br>0.7520*<br>0.9666† | 0.9841<br>0.7967*<br>0.9773† | **0.999**<br>0.973*<br>**0.999†** | **0.998**<br>0.985*<br>**0.998†** |
| | SMOTE | 0.9722 | 0.7788 | 0.8642 | **0.9807** | **0.997** | 0.989 | 0.9778 | 0.9715 | 0.9746 | **0.9883** | **0.999** | **0.998** |
| | ADASYN | 0.9640 | 0.7382 | 0.8348 | 0.9752 | 0.996 | 0.986 | 0.9669 | 0.9709 | 0.9688 | 0.9827 | **0.999** | **0.998** |
| | MWMOTE | 0.9709 | 0.6992 | 0.8119 | 0.9773 | **0.997** | 0.983 | 0.9713 | 0.9491 | 0.9598 | 0.9845 | **0.999** | 0.997 |
| | ESPO | **0.9804** | 0.5467 | 0.7016 | 0.9746 | 0.996 | 0.969 | 0.9802 | 0.8745 | 0.9240 | 0.9873 | **0.999** | 0.994 |
| | INOS | 0.9782 | 0.5465 | 0.7006 | 0.9735 | 0.996 | 0.969 | **0.9825** | 0.8563 | 0.9148 | 0.9881 | **0.999** | 0.993 |
| | MOKAS | 0.9469 | 0.7896 | 0.8603 | 0.9683 | 0.995 | 0.988 | 0.9698 | 0.9775 | 0.9736 | 0.9844 | **0.999** | 0.970 |
| VEP | w/o | 0.106 | 0.225 | 0.132 | 0.189 | 0.636 | **0.873** | 0.198<br>**1.000***<br>0.206† | 0.270<br>0.124*<br>**0.293†** | 0.211<br>0.220*<br>0.225† | 0.352<br>0.000*<br>0.372† | 0.622<br>0.487*<br>0.633† | 0.825<br>0.124*<br>**0.828†** |
| | SMOTE | 0.461 | 0.226 | 0.295 | 0.564 | 0.639 | 0.706 | 0.320 | 0.245 | 0.271 | 0.487 | 0.618 | 0.775 |
| | ADASYN | 0.468 | 0.225 | 0.295 | 0.568 | 0.640 | 0.704 | 0.296 | 0.235 | 0.256 | 0.455 | 0.615 | 0.785 |
| | MWMOTE | 0.446 | **0.235** | 0.299 | 0.564 | 0.652 | 0.729 | 0.332 | 0.257 | 0.281 | 0.491 | 0.627 | 0.781 |
| | ESPO | 0.502 | 0.222 | 0.299 | 0.575 | 0.649 | 0.697 | 0.421 | 0.217 | 0.281 | 0.538 | 0.629 | 0.723 |
| | INOS | 0.511 | 0.218 | 0.297 | 0.582 | 0.642 | 0.679 | 0.425 | 0.230 | **0.292** | **0.549** | 0.628 | 0.725 |
| | MOKAS | **0.513** | 0.234 | **0.309** | **0.591** | **0.660** | 0.694 | 0.371 | 0.249 | 0.288 | 0.517 | **0.636** | 0.756 |

Overall, as shown in Fig. 4 and Table 1, synthetic instances generated by MOKAS better fit existing minority class instances. This is also revealed by the fact that for each data set, the performance of a 1-NN classifier on the data generated by MOKAS is better than or equal to the best of the other five synthetic instance generation schemes. Regardless of the distinct characteristics of different datasets, MOKAS, utilizing the kernel adaptive subspaces, demonstrated its robustness to describe the distribution of the minority class.

### 3.2 Classification performance

The effectiveness of each oversampling algorithm in assisting classification algorithms in an imbalanced learning task is thoroughly compared in this subsection. The experiment is conducted on 10 real-world datasets (as listed in Table 2), taken from the UCI machine learning repository [38]. To fit our research interests, some minor modifications were made on the datasets. For example, manual selection of two classes as majority and minority classes or given a selected class as the minority class, combining all other classes as the majority class. Additionally, to verify the performance of the proposed algorithm on data with high noise and variation, EEG datasets collected from 14 human subjects in a VEP oddball task [39] are used. Regarding classification algorithms, three-layer neural networks (with an input layer, a 10-node hidden layer, and an output layer) and support vector machines (with RBF kernel function) [41] are used in this study. The penalty parameter $C$ of SVM and the spread parameter $\gamma$ for the RBF kernel in SVM are selected by a grid search [42]. For $\gamma$ we consider the set $\{2^{-15}, 2^{-13}, \cdots, 2^{3},\}$and for $C$ we consider the set $\{2^{-5}, 2^{-3}, \cdots, 2^{15},\}$. Note that, these parameters are obtained directly from the training data. To find the separating hyperplane for imbalanced data, the "balanced" mode of SVM is also applied. The "balanced" mode automatically adjusts weights inversely proportional to class frequencies in the input data [43].

When the classification problem is a balanced one, accuracy or error rate is a good measure of performance. However, for highly imbalanced data, accuracy is not at all a good performance index as a classifier may learn the majority class completely ignoring the minority class, yet yielding a high accuracy. Hence, in this study, in addition to Accuracy (ACC) or Area Unver Curve (AUC), four commonly used metrics, recall, precision, f-value, and g-mean , are calculated to assess the effectiveness of imbalance learning techniques. The confusion matrix provided in Table 3, interprets the physical meaning of such metrics. Recall is often regarded as a measure of completeness, indicating faction of correctly classified positive (minority) instances among the actual positive instances. On the other hand, precision is a measure of exactness that states the fraction of true positive instance among positively classified instances. Thus, precision and recall, two together provides a better picture, but each of them separately gives a one sided picture. The F-value and G-mean act as overall assessment metrics that reflect compromised performance between the two classes.

TABLE 6
AVERAGE RANKS GATHERED FROM NEURAL NETWORK RESULTS ARE PRESENTED IN THE UPER HALF AND THAT FROM SVM ARE IN THE LOWER HALF. BEST RESULTS (SMALLEST RANKS) ARE HIGHLIGHTED IN BOLD FACE.

| | OVERSAMPLING METHOD | REC. | PRE. | F | G | OVERALL |
|---|---|---|---|---|---|---|
| NN | w/o | 6.82 | 3.30 | 5.82 | 6.80 | 5.68 |
| | SMOTE | 3.45 | 3.60 | 3.36 | 2.80 | 3.30 |
| | ADASYN | 5.64 | 4.10 | 4.91 | 5.50 | 5.04 |
| | MWMOTE | 3.64 | 3.00 | 2.91 | 3.20 | 3.19 |
| | ESPO | 2.82 | 5.20 | 4.27 | 3.10 | 3.85 |
| | INOS | 2.82 | 5.80 | 4.73 | 3.70 | 4.26 |
| | MOKAS | **2.55** | **2.80** | **2.00** | **2.40** | **2.44** |
| SVM | w/o | 8.09 | 2.09 | 5.36 | 6.91 | 5.61 |
| | Balanced mode | 4.45 | 5.91 | 5.73 | 6.36 | 5.61 |
| | SVM*light* | 6.82 | **2.55** | 5.36 | 8.36 | 5.77 |
| | SMOTE | 5.64 | 4.91 | 5.09 | 5.09 | 5.18 |
| | ADASYN | 6.55 | 5.36 | 5.82 | 6.09 | 5.95 |
| | MWMOTE | 4.27 | 3.82 | 4.18 | 4.45 | 4.18 |
| | ESPO | **2.55** | 6.36 | 5.00 | 2.64 | 4.14 |
| | INOS | 2.73 | 6.55 | 5.00 | 2.64 | 4.23 |
| | MOKAS | 3.18 | 3.36 | **2.27** | **2.45** | **2.82** |

For a fair comparison, each oversampling algorithm is required to generate as much synthetic instances as necessary to balance the datasets. Here, the number of synthetic instances generated, $M$, is determined by Eq. (3). In MOKAS, the gamma value of the RBF kernel function is set to 0.5 for data generation. In the pre-image process, five neighbors are selected to reconstruct the input space representation. After some preliminary experiments, the number of basis vectors is empirically set to $N_{min} \times 0.35$.

The classification results are obtained via a training-test partition based validation scheme consisting of an outer iteration and an inner iteration. In the outer iteration, 70% of randomly selected instances and the remaining 30% instances are used to form the training and the testing sets, respectively. Notice that the selection process is separately performed on each class data so that the imbalance ratio remains the same in the training and test sets. The chosen oversampling algorithms are trained using the training set, and then synthetic instances are generated to balance the training set. Afterwards, in the inner iteration, classifiers are trained using the balanced training set and the trained classifiers are applied on the test dataset. For each outer iteration, the inner iteration is repeated 50 times. The outer iteration is repeated five times. In short, the classification results are obtained by averaging the results of 250 iterations to account for randomness in oversampling algorithms as well as in the generated data partition.

Tables 4 and 5 show the comparison of classification results among SMOTE, ADASYN, MWMOTE, ESPO, INOS, and MOKAS. Classification results obtained without applying any oversampling algorithm are labeled "w/o". The asterisk and cross denote classification results obtained using SVM with 'balanced' mode and SVM*light* [44], respectively. To further interpret the results, as done in [14], for each data set and each validation measure we rank the performance of different instance generation methods. In each assessment metric, the method that exhibits the best result is assigned a rank one; the second one is given a rank two and so on. This is done separately for the neural networks and SVMs and the average ranks for all algorithms considered are summarized

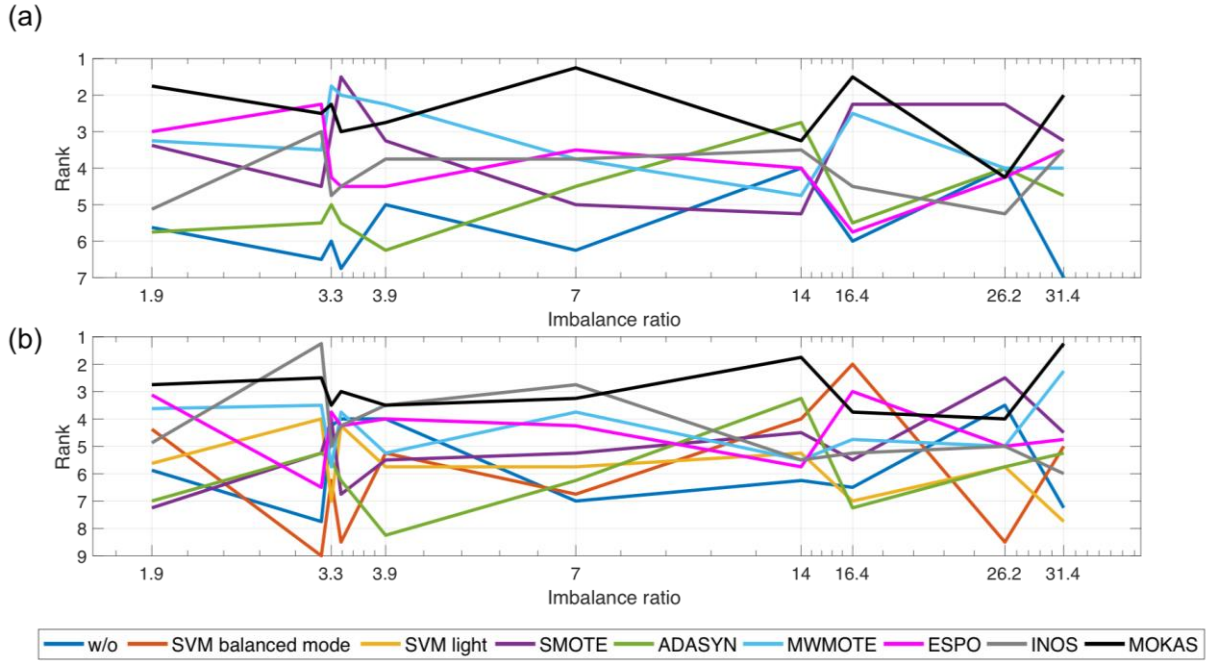


Fig. 5. Overall performance (average of performance matrices) of different oversampling methods against different imbalance ratios. (a) NN. (b) SVM.

in Table 6. The overall performance (average of performance matrices) against different imbalance ratios is shown in Fig. 5. Overall, MOKAS (black trace) performs better than other oversampling methods in most of cases.

When neural networks are applied, MOKAS is ranked the number one in terms of recall (2.55 rank value), precision (2.80 rank value), f-value (2.00 rank value), g-mean (2.40 rank value), and overall rank (2.44 rank value). When SVMs are applied, MOKAS is ranked the topmost one in terms of f-value (2.27 rank value), g-mean (2.45 rank value), and overall rank (2.82 rank value). In terms of recall, MOKAS (3.18 rank value) falls behind ESPO (2.55 rank value) and INOS (2.73 rank value). These results suggest that, in terms of precision, the oversampling method may not help much.

## 4 Discussion

### 4.1 Classification performance

Referring to the experimental results provided in the previous section, we find that whenever an oversampling algorithm is applied, with a few exceptions, recall rates are improved. On the contrary, precision rates are often degraded after oversampling algorithms are applied. This phenomenon usually arises when there are overlapped classes in the datasets, and overlapped classes are commonly observed in real-world data sets. When an oversampling algorithm acts on a dataset with some overlap between classes, oversampling may push the boundary towards the majority class. Consequently, classifiers are able to detect more minority class instances but might also misclassify a portion of the majority class instances around the boundary, resulting in an improved recall rate but a degraded precision rate.

The results in Table 4, reveal that the structure-preserving methods improve the performance in terms of Recall rate but degrade the results in terms of precision rate. On the other hand, MWMOTE, one of the interpolation-based methods used in this study, tends to generate better results in terms of precision rate but worse results in terms of recall rate when compared with MOKAS. The oversampling mechanisms account for these differences. In INOS algorithm, although principal components are extracted from the training data, a normal distribution is assumed when producing synthetic instances. Such an assumption may fail to reflect the actual data distribution and may result in a larger boundary for the minority class. Hence, recall rate can be expected to be improved but at the cost of degradation in precision rate. On the other hand, interpolation-based methods generate synthetic instances on the line segment between a seed instance and its neighboring instances. So we can expect that the boundary of the minority class will not expand considerably. In addition, MWMOTE exploits an advanced mechanism to identify hard-to-learn instances. This mechanism further avoids over-fitting problem. However, such a conservative strategy may also limit the improvement on recall rate. In the subspace-projection methods such as the one proposed here, MOKAS, the synthetic instances are generated from subspaces that are trained to appropriately describe the minority class distribution. In this category, no specific distribution type is presumed and certain variations are introduced when generating synthetic instances. This results in a more useful oversampling strategy, which strikes a balance between recall and precision rates.

### 4.2 Number of Basis Vectors in MOKAS

A major factor determining the outcome of MOKAS is the number of basis vectors. In Fig. 6, several scatter plots are presented to visualize the differences when adopting varying number of basis vectors to model data distribution. When the number of basis vectors increases, the synthetic instances are located closer to the existing instances. This is straightforward since the information loss is reduced when more basis vectors are exploited to model the data distribution and hence results in more similar synthetic instances.

### 4.3 Limitation of MOKAS

A limitation of the proposed method is that it does scale well with number of data points in the training set. Like SVM, the size of kernel matrix increases quadratically with the number of data points making memory storage and computation requirement difficult, as show in Table 7.

## 5 Conclusions

Inspired by the kernel-based ASSOM structure, we de-

TABLE 7
COMPARISON OF COMPUTATION TIME OF DIFFERENT METHODS FOR DIFFERENT DATA SETS (IN SECOND).

| | | Dataset | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Ozone | Libras | Letter | abalone | Breast cancer | Ecoli | Glass | Pima | Vehicle | Yeast | VEP |
| Method | SMOTE | 0.207 | 0.082 | 6.692 | 0.15 | 0.118 | 0.077 | 0.066 | 0.126 | 0.171 | 0.312 | 0.066 |
| | ADASYN | 0.362 | 0.102 | 16.701 | 0.185 | 0.273 | 0.11 | 0.077 | 0.302 | 0.343 | 0.79 | 0.075 |
| | MWMOTE | 0.283 | 0.108 | 40.473 | 0.243 | 2.019 | 0.197 | 0.117 | 2.813 | 1.482 | 4.03 | 0.075 |
| | ESPO | 5.214 | 0.301 | 529.323 | 1.171 | 0.495 | 0.214 | 0.127 | 0.581 | 0.998 | 2.978 | 0.142 |
| | INOS | 5.384 | 0.376 | 508.599 | 1.292 | 0.606 | 0.231 | 0.136 | 0.499 | 1.049 | 3.158 | 0.164 |
| | MOKAS | 12.89 | 2.355 | 106340 | 5.109 | 110.096 | 5.473 | 1.798 | 156.326 | 176.577 | 745.274 | 0.269 |

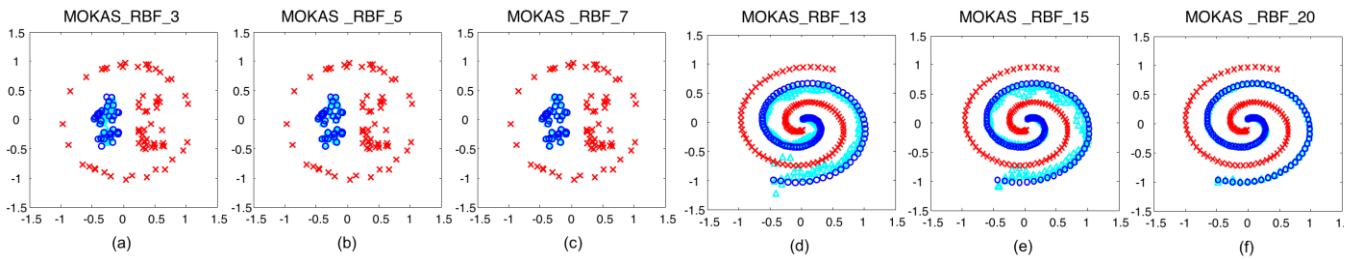


Fig. 6. Scatter plots of the original data and synthetic data of Boat dataset generated by MOKAS using (a) 3, (b) 5, and (c) 7 basis vectors and Two spirals dataset generated by MOKAS using (d) 13, (e) 15, and (f) 20 basis vectors

velop a novel oversampling algorithm, MOKAS, which exploits subspaces to model data distribution in a high-dimensional feature space. The subspaces are trained in a competitive learning framework like SOM and they adapt to different characteristics of the minority class instances. Synthetic instances are generated by projecting existing minority class instances onto different subspaces. The synthetic instances bear much resemblance to existing instances but each possesses different characteristic inherited from the subspace. A pre-image method is utilized to transform the synthetic instances from the feature space back to the input space.

Visualization results show that MOKAS is capable of modeling complex data distribution. Classification results also show that the proposed algorithm effectively copes with imbalanced distribution of instances over different classes and assists classifiers in learning from imbalanced data. MOKAS exhibits superior overall performance compared to selected state-of-the-art interpolation-based and structure-preserving oversampling algorithms.

## ACKNOWLEDGMENT

This work was supported in part by the Australian Research Council (ARC) under discovery grant DP180100670 and DP180100656. Research was also spon-sored in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0022 and W911NF-10-D-0002/TO 0023. The views and the conclusions contained in this docu-ment are those of the authors and should not be inter-preted as representing the official policies, either ex-pressed or implied, of the Army Research Laboratory or the U.S Government. The U.S Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

## REFERENCES

[1] Q. Yang and X. Wu, "10 challenging problems in data mining research," *International Journal of Information Technology and Decision Making,* vol. 5, pp. 597-604, 2006.

[2] H. Parvin, B. Minaei-Bidgoli, and H. Alizadeh, "Detection of cancer patients using an innovative method for learning at imbalanced datasets," in *Rough Sets and Knowledge Technology*. vol. 6954, J. Yao, S. Ramanna, G. Wang, and Z. Suraj, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 376-381.

[3] M. D. Martino, F. Decia, J. Molinelli, and A. Fernandez, "Improving electric fraud detection using class imbalance strategies," in *International Conference on Pattern Recognition Applications and Methods*, 2012, pp. 135-141.

[4] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen, "Effective detection of sophisticated online banking fraud on extremely imbalanced data," *World Wide Web,* vol. 16, pp. 449-475, 2013.

[5] M. Kubat, R. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine Learning,* vol. 30, pp. 195-215, 1998.

[6] G. Xu, F. Shen, and J. Zhao, "The effect of methods addressing the class imbalance problem on P300 detection," in *The 2013 International Joint Conference on Neural Networks*, Dallas, TX, 2013.

[7] A. Amin, S. Anwar, A. Adnan, M. Nawaz, N. Howard, J. Qadir, *et al.*, "Comparing oversampling techniques to handle the class imbalance problem: a customer churn prediction case study," *IEEE Access,* vol. 4, pp. 7940-7957, 2016.

[8] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering,* vol. 21, pp. 1263-1284, 2009.

[9] G. M. Weiss, "Mining with rarity: a unifying framework," *ACM SIGKDD Explorations Newsletter,* vol. 6, pp. 7-19, 2004.

[10] R. Prati, G. A. P. A. Batista, and M. Monard, "Class Imbalances versus Class Overlapping: An Analysis of a Learning System Behavior," in *MICAI 2004: Advances in Artificial Intelligence*. vol. 2972, R. Monroy, G. Arroyo-Figueroa, L. Sucar, and H. Sossa, Eds., ed: Springer Berlin Heidelberg, 2004, pp. 312-321.

[11] H. Cao, X.-L. Li, D.-K. Woon, and S.-K. Ng, "Integrated oversampling for imbalanced time series classification," *IEEE Transactions on Knowledge and Data Engineering,* vol. 25, pp. 2809-2822, 2013.

[12] T. Imam, K. Ting, and J. Kamruzzaman, "z-SVM: An SVM for improved classification of imbalanced data," in *AI 2006: Advances in Artificial Intelligence*. vol. 4304, A. Sattar and B.-h. Kang, Eds., ed: Springer Berlin Heidelberg, 2006, pp. 264-273.

[13] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," in *The 2010 International Joint Conference on Neural Networks*, 2010, pp. 1-8.

[14] R. Batuwita and V. Palade, "FSVM-CIL: Fuzzy support vector machines for class imbalance learning," *IEEE Transactions on Fuzzy Systems,* vol. 18, pp. 558-571, Jun 2010.

[15] A. Tayal, T. F. Coleman, and Y. Li, "Rankrc: Large-scale nonlinear rare class ranking," *IEEE Transactions on Knowledge and Data Engineering,* vol. 27, pp. 3347-3359, Jul 2015.

[16] P. Cao, D. Zhao, and O. Zaiane, "An optimized cost-sensitive SVM

for imbalanced data learning," in *Advances in Knowledge Discovery and Data Mining*. vol. 7819, J. Pei, V. Tseng, L. Cao, H. Motoda, and G. Xu, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 280-292.

[17] A. Manukyan and E. Ceyhan, "Classification of Imbalanced Data with a Geometric Digraph Family," *Journal of Machine Learning Research,* vol. 17, pp. 1-40, Jan 2016.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research,* vol. 16, pp. 321-357, 2002.

[19] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE International Joint Conference on Neural Networks*, 2008, pp. 1322-1328.

[20] S. Barua, M. M. Islam, Y. Xin, and K. Murase, "MWMOTE--Majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Transactions on Knowledge and Data Engineering,* vol. 26, pp. 405-425, Feb 2014.

[21] C. Hong, L. Xiao-Li, W. Yew-Kwong, and N. See-Kiong, "SPO: Structure preserving oversampling for imbalanced time series classification," in *IEEE 11th International Conference on Data Mining (ICDM)*, Vancouver, Canada, 2011, pp. 1008-1013.

[22] L. Abdi and S. Hashemi, "To combat multi-class imbalanced problems by means of over-sampling techniques," *IEEE Transactions on Knowledge and Data Engineering,* vol. 28, pp. 238-251, Jul 2016.

[23] A. Moreo, A. Esuli, and F. Sebastiani, "Distributional Random Oversampling for Imbalanced Text Classification," presented at the Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, Pisa, Italy, 2016.

[24] B. Das, N. C. Krishnan, and D. J. Cook, "RACOG and wRACOG: Two probabilistic oversampling techniques," *IEEE Transactions on Knowledge and Data Engineering,* vol. 27, pp. 222-234, May 2015.

[25] A. Pourhabib, B. K. Mallick, and Y. Ding, "Absent data generating classifier for imbalanced class sizes," *Journal of Machine Learning Research,* vol. 16, pp. 2695-2724, Jan 2015.

[26] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering,* vol. 30, pp. 25-36, 2006.

[27] T. M. Ha and H. Bunke, "Off-Line, Handwritten Numeral Recognition by Perturbation Method," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 19, pp. 535-539, 1997.

[28] R. C. Prati, G. E. Batista, and M. C. Monard, "A study with class imbalance and random sampling for a decision tree learning system," in *Artificial Intelligence in Theory and Practice II*, ed: Springer, 2008, pp. 131-140.

[29] X. Zhang, D. Ma, L. Gan, S. Jiang, and G. Agam, "CGMOS: Certainty Guided Minority OverSampling," presented at the Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, Indiana, USA, 2016.

[30] T. Kohonen, "The self-organizing map," *Neurocomputing,* vol. 21, pp. 1-6, 1998.

[31] T. Kohonen, S. Kaski, and H. Lappalainen, "Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM," *Neural Computation,* vol. 9, pp. 1321-1344, 1997.

[32] T. Kohonen, "Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map," *Biological Cybernetics,* vol. 75, pp. 281-291, Nov. 1996.

[33] H. Kawano, T. Yamakawa, and K. Horio, "Kernel-based adaptive-subspace self-organizing map as a nonlinear subspace pattern recognition," in *Proceedings of the World Automation Congress*, 2004, pp. 267-272.

[34] J. T.-Y. Kwok and I. W.-H. Tsang, "The pre-image problem in kernel methods," *IEEE Transactions on Neural Networks,* vol. 15, pp. 1517-1525, Nov. 2004.

[35] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science,* vol. 11, pp. 23-63, 1987.

[36] K.-J. Bathe and E. L. Wilson, *Numerical methods in finite element analysis*. Englewood Cliffs, New Jersey: Prentice-Hall, 1976.

[37] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling*, 2 ed. Florida: Chapman and Hall/CRC, 2000.

[38] M. Lichman. UCI Machine Learning Repository [Online]. Available: http://archive.ics.uci.edu/ml

[39] A. Ries, J. Touryan, J. Vettel, K. McDowell, and W. D. Hairston, "A Comparison of Electroencephalography Signals Acquired from Conventional and Mobile Systems," *Journal of Neuroscience and Neuroengineering,* vol. 3, pp. 10-20, 2014.

[40] L. Kuncheva. (1 July 2015 ). *Artificial data sets*. Available: http://pages.bangor.ac.uk/~mas00a/activities/artificial_data.htm

[41] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning,* vol. 20, pp. 273-297, 1995.

[42] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. (2010, A practical guide to support vector classification. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[43] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for support vector machines," *ACM Transactions on Intelligent Systems and Technology,* vol. 2, pp. 1-27, April 2011.

[44] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in kernel methods*, S. Bernhard, lkopf, J. C. B. Christopher, and J. S. Alexander, Eds., ed: MIT Press, 1999, pp. 169-184.

**Chin-Teng Lin** received the B.S. degree from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1986, and the master's and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1989 and 1992, respectively. He is currently the Distinguished Professor with the University of Technology Sydney, Sydney, NSW, Australia, and the Honorary Professorship with the University of Nottingham, Nottingham, U.K. He has co-authored the book entitled Neural Fuzzy Systems (Prentice Hall) and has authored the book entitled Neural Fuzzy Control Systems with Structure and Parameter Learning (World Scientific). He has authored over 260 journal papers (Total Citation: 17403, H-index: 60, and i10-index: 213) in the areas of neural networks, fuzzy systems, brain computer interface, multimedia signal processing, and cognitive neuroengineering, including approximately 110 IEEE journal papers.
Dr. Lin is a fellow of IEEE and the International Fuzzy Systems Association. He served as the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS from 2011 to 2017. He served on the Board of Governors at the IEEE Circuits and Systems (CAS) Society from 2005 to 2008, the IEEE Systems, Man, Cybernetics Society from 2003 to 2005, and the IEEE Computational Intelligence Society from 2008 to 2010, and the Chair of the IEEE Taipei Section from 2009 to 2010. He was the Distinguished Lecturer of the IEEE CAS and CIS Societies. He served as the Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II from 2006 to 2008. He was the Program Chair of the IEEE International Conference on Systems, Man, and Cybernetics in 2005 and the General Chair of the 2011 IEEE International Conference on Fuzzy Systems.

**Tsung-Yu Hsieh** received the B.S. degree from the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan, in 2015. He is pursuing his Ph.D. degree at the College of Information Sciences and Technology, University of Pennsylvania, University of Pennsylvania. His research interests include fuzzy logic theory, machine learning, biomedical signal processing, and brain-computer interface.

**Yu-Ting Liu** received the B.S. and Ph.D. degrees from the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan, in 2010 and 2016. He was a Lecturer with the University of Technology Sydney, Australia. Currently, he is a Senior Engineer with MediaTek Inc., Taiwan. His research interests include fuzzy logic theory, machine learning, computer vision, EEG analysis, and application.

**Yang-Yin Lin** received the M.S. degree from the Institute of Electrical Engineering, National Chung Hsing University, Taichung, Taiwan, in 2008, and the Ph.D. degree from the Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan, in 2013. He is currently with the Electronic Systems Research Division, National Chung-Shan Institute of Science and Technology, Taoyuan, Taiwan, as an Assistant Researcher. His current research interests include evolutionary computation, type-2 neural fuzzy systems, deep learning, transfer fuzzy learning, and field-programmable gate array chip design based on neural network architecture.

**Chieh-Ning Fang** received the B.S. degree from the Department of Electrical and Computer Engineering in 2014 and M.S. degree from Institute of Electrical and Control Engineering in 2016, National Chiao-Tung University, Hsinchu, Taiwan. She currently works as research assistant in Computational Intelligence and Brain Computer Interface Centre, University of Technology Sydney. Her research interests cover deep learning, neural network, image recognition and pattern recognition. Future research aims at developing a brain computer interface associated with deep learning algorithm to improve its performance.

**Yu-Kai Wang** received the B.S. degree in mathematics education from National Taichung University of Education, Taichung, Taiwan, in 2006, the M.S. degree in biomedical engineering from National Chiao Tung University (NCTU), Hsinchu Taiwan, in 2009, and the Ph.D. degree in computer science from NCTU, Hsinchu Taiwan, in 2015. He was a Visiting Scholar with the Swartz Center for Computational Neuroscience, University of California at San Diego, La Jolla, CA, USA, from 2013 to 2014. He is currently a postdoctoral researcher with the Centre for Artificial Intelligence in University of Technology Sydney, Australia. His current research interests include machine learning, computational neuroscience, biomedical signal processing, and brain–computer interface.

**Gary G. Yen** received the Ph.D. degree in electrical and computer engineering from University of Notre Dame, Notre Dame, IN, USA, in 1992. He is currently a Professor in the School of Electrical and Computer Engineering, Oklahoma State University (OSU), Stillwater, OK, USA. Before he joined OSU in 1997, he was with the U.S. Air Force Research Laboratory, Albuquerque, NM, USA. His research is supported by the DoD, DoE, EPA, NASA, NSF, and Process Industry. His research interests include intelligent control, computational intelligence, conditional health monitoring, signal processing, and their industrial/defense applications. Dr. Yen was an Associate Editor of the IEEE Control Systems Magazine, IEEE Transactions on Control Systems Technology, Automatica, Mechantronics, IEEE Transactions on Systems, Man and Cybernetics, Part A and Part B, and IEEE Transactions on Neural Networks. He is currently serving as an Associate Editor for the IEEE Transactions on Evolutionary Computation and International Journal of Swarm Intelligence Research. He served as the General Chair for the 2003 IEEE International Symposium on Intelligent Control held in Houston, TX, and 2006 IEEE World Congress on Computational Intelligence held in Vancouver, Canada. He served as the Vice President for the Technical Activities in 2005–2006 and the President in 2010–2011 of the IEEE Computational Intelligence Society and is the founding Editor-in-Chief of the IEEE Computational Intelligence Magazine 2006–2009. In 2011, he received the Andrew P. Sage Best Transactions Paper award from IEEE Systems, Man and Cybernetics Society. In 2013, he received the Meritorious Service award from the IEEE Computational Intelligence Society. He is a Fellow of IET.

**Nikhil R. Pal** is a Professor in the Electronics and Communication Sciences Unit of the Indian Statistical Institute. His current research interest includes bioinformatics, brain science, fuzzy logic, pattern analysis, neural networks, and evolutionary computation. He was the Editor-in-Chief of the IEEE Transactions on Fuzzy Systems (January 2005-December 2010). He has served/been serving on the editorial/advisory board/ steering committee of several journals including the International Journal of Approximate Reasoning, Applied Soft Computing, Fuzzy Sets and Systems, Fuzzy Information and Engineering: An International Journal, IEEE Transactions on Fuzzy Systems and the IEEE Transactions on Systems Man and Cybernetics B. He is a recipient of the 2015 Fuzzy Systems Pioneer Award. He has given many plenary/keynote speeches in different premier international conferences in the area of computational intelligence. He has served as the General Chair, Program Chair, and co-Program chair of several conferences. He was a Distinguished Lecturer of the IEEE Computational Intelligence Society (CIS) and was a member of the Administrative Committee of the IEEE CIS. At present he is the Vice President for Publications of the IEEE CIS.
He is a Fellow of the IEEE, the National Academy of Sciences, India; the Indian National Academy of Engineering; the Indian National Science Academy, the International Fuzzy Systems Association (IFSA), and The World Academy of Sciences (TWAS).

**Chun-Hsiang Chuang** received the B.S. degree in mathematics education from Taipei Municipal Teachers College, Taipei, Taiwan, in 2004, the M.S. degree in educational measurement and statistics from National Taichung University, Taichung, Taiwan, in 2009, and the Ph.D. degree in electrical engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2014. He was a Visiting Scholar with the Swartz Center for Computational Neuroscience, University of California at San Diego, La Jolla, CA, USA, from 2012 to 2013. From 2014-2016, he was a postdoctoral researcher and an assistant researcher with the Brain Research Center, National Chiao Tung University, Taiwan. He is currently a Lecturer with the University of Technology Sydney, Australia. His current research interests include machine learning, computational neuroscience, biomedical signal processing, and brain–computer interface.