This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Integrated Oversampling for Imbalanced Time Series Classification

Hong CAO, Xiao-Li LI, Yew-Kwong WOON and See-Kiong NG

Abstract—This paper proposes a novel Integrated Oversampling (INOS) method that can handle highly imbalanced time series classification. We introduce an enhanced structure preserving oversampling (ESPO) technique and synergistically combine it with interpolation-based oversampling. ESPO is used to generate a large percentage of the synthetic minority samples based on multivariate Gaussian distribution, by estimating the covariance structure of the minority-class samples and by regularizing the unreliable eigen spectrum. To protect the key original minority samples, we use an interpolation-based technique to oversample a small percentage of synthetic population. By preserving the main covariance structure and intelligently creating protective variances in the trivial eigen dimensions, ESPO effectively expands the synthetic samples into the void area in the data space without being too closely tied with existing minority-class samples. This also addresses a key challenge for applying oversampling for imbalanced time series classification, i.e. maintaining the correlation between consecutive values through preserving the main covariance structure. Extensive experiments based on seven public time series datasets demonstrate that our INOS approach, used with Support Vector Machines (SVM), achieved better performance over existing oversampling methods as well as state-of-the-art methods in time series classification.

Index Terms— Oversampling, learning, imbalanced data, time series, SVM, structure preserving, classification.

1 Introduction

Data imbalance is a key source of performance degradation [1, 2] in machine learning and data mining. Existing algorithms either explicitly or implicitly assume a balanced class distribution, with sufficient and roughly equal number of learning samples for each class, as illustrated in Fig. 1(a). In many real world applications, the data available for learning are highly imbalanced, where one class can severely out-represent another class. Moreover, the minority class often represents the class of interest. For instance, the known instances of earthquakes are rare but certainly of greater interest for earthquake prediction than the abundant normal instances. Similar scenarios can be found in fraud detection in financial data analysis, intrusion detection in network forensics, cancer detection in biomedical diagnosis, object detection in computer vision, diagnosis and prognosis of machine failures, etc. In these scenarios, the positive (Pos) class, containing only a few sparsely distributed samples, is outnumbered by the negative (Neg) class, as shown in Fig. 1(b). Consequently, the learning algorithms tend to bias towards the less important Neg class with the larger population.

Existing solutions attempt to address the imbalanced learning issue at various levels: the data level [3-8], the algorithm level [2, 9], or a combination of both levels [10-14]. The methods that address the problem at the data level re-establish the class balance through data resampling, such as oversampling of the minority class, undersampling of the majority class, or both. The methods that address the problem at the algorithm level enforce emphasis on the minority class by manipulating and incorporating learning parameters such as dataspace weighting [9-11, 13], class-dependant cost matrix, and receiver operating characteristics (ROC) threshold [15] into conventional learning paradigms. While some recent algorithm-level methods reviewed in [2] have achieved good results on certain imbalanced datasets, we believe the data-level approaches hold good potential in solving the data imbalance issue. In particular, we investigate the oversampling approach in this paper as it addresses the imbalance issue at the most fundamental data level and can serve as a generic preprocessing step. In addition, unlike undersampling, oversampling suffers no risk of losing important learning samples.

Data mining in many domains such as finance, aerospace, entertainment, network security, and medicine, involve time series data [16-20]. As defined in [18], a time series data sample is an ordered set of real-valued variables that are sampled or extracted from a continuous signal, which can be either in time or spatial domain. Due to its sequential nature, variables that are close by in a time series are often highly correlated. This observation can be exploited for time series classification. For example, we can compute the distance between two samples, known as warping distance, by searching for the optimal mapping path that realigns the two time series sequences. We can then classify a given test sample based on the label of the top-one nearest training neighbor. This is the algorithm for the one nearest neighbor (1NN) classifier with dynamic time warping (DTW) [16], which is one best-known learning method for time series classification.

Clearly, the imbalanced learning problem for time series classification is much more daunting than typical imbalanced classification problems because of its high dimensionality. Very often, the number of available samples in the minority class is few as compared with the dimensionality. The inherent data complexity of time series classification suggests that it is sensible to address the

xxxx-xxxx/0x/$xx.00 © 2012 IEEE

Digital Object Indentifier 10.1109/TKDE.2013.37 1041-4347/13/$31.00 © 2013 IEEE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

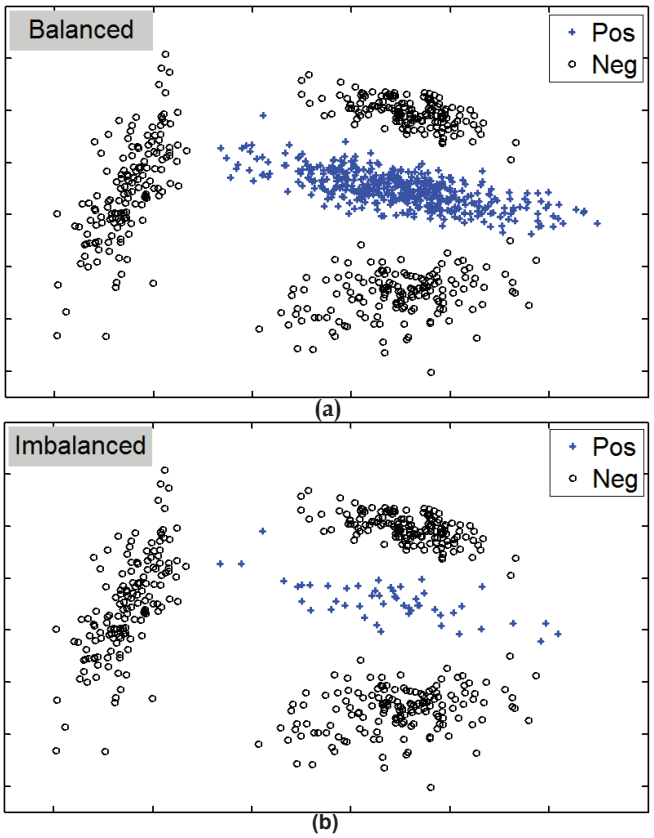


Fig. 1. Comparison of the balanced class population in (a) and the practical imbalanced distribution in (b) in 2D (two-dimensional) feature space. The data is synthetically generated for illustration purpose.

imbalance problem at the data level using oversampling, as oversampling has been found to be effective for re-establishing the class balance at the data level for generic imbalanced classification problems. However, as far as we know, oversampling has not been well-explored for imbalanced time series classification due to the complexity of the problem. In this paper, we focus on the task of oversampling for learning from highly imbalanced two-class time-series data.

2 Oversampling for Imbalanced Time Series Learning

2.1 Prior Oversampling Methods

For learning from imbalanced two-class datasets, oversampling is to balance the class distribution through augmenting the minority-class (or positive class) with synthetically generated samples. One naïve approach is to repeat the existing positive samples as many times as necessary to balance the two classes. Clearly, this would lead to overfitting. As such, the current oversampling solutions often introduce some degree of variations when generating the synthetic samples. This can be done in one of the following two approaches. The first approach interpolates between selected positive samples and their random positive nearest neighbors for generating the synthetic samples. Well-known oversampling methods that adopt this approach are SMOTE [3], Borderline-SMOTE [6] and ADASYN [7]. Particularly, SMOTE selects all positive samples and evenly generates the synthetic samples from each selected seed sample. Borderline-SMOTE identifies a set of hard and non-outlier positive samples at the class border and evenly generates the synthetic samples from this border set. ADASYN adopts an adaptive approach where the number of synthetic samples to be generated for each positive sample is determined by the percentage of negative samples in its neighborhood. For detailed description and technical differences of these interpolation-based methods, one can refer to Sec. 3 of the survey [2]. While SMOTE has been extended [20] for oversampling in the distance space where the imbalanced data are represented in the special format of distance matrix with pairwise distance, it has not been targeted for imbalanced time-series classification.

The second class of oversampling approaches generates the features of the synthetic samples *individually*. A well-known method is DataBoost [13] which generates each feature value based on Gaussian distribution within an empirical range [min, max].

2.2 Our Solution via Structure Preservation

As the interpolation-based approaches have been shown to work fairly well for various generic imbalanced learning tasks, we opine that they will not be adequate for the task of oversampling highly imbalanced time series datasets. As we have noted earlier, the adjacent variables in the time series are usually not independent but *highly correlated*. The random data variances introduced by both conventional oversampling approaches will weaken the inherent correlation structures within the original time series data. This will lead to the generation of non-representative synthetic training samples with excessive noise that confound the classification learning.

As such, we have designed our proposed INOS method with two objectives in mind: one is to preserve the regularized eigen covariance structure which can be estimated using the limited positive time series samples, the other is to be able to provide adequate emphasis on the key minority samples with the remaining oversampling capacity. For the first objective, we propose a new enhanced structure preserving oversampling (ESPO). ESPO performs oversampling in the transformed signal space in the following steps: 1) Generating the synthetic samples by estimating and maintaining the main covariance structure in the reliable eigen subspace; 2) inferring and fixing the unreliable eigen spectrum, which is insufficiently estimated due to the limited number of positive samples, using a regularization procedure. In this way, some buffer variances of the synthetic data are created in the trivial eigen subspace to improve the generalization performance on the unseen data. For the second objective, we use an interpolation-based method to generate a small percentage of synthetic samples so as to emphasize on the border set of existing samples, which are critical for building accurate classifiers in the subsequent steps.

Compared with our previous preliminary SPO work [26], the current proposed INOS method differs and improves in the following aspects: 1) The oversampling is performed in the signal space with improved efficiency

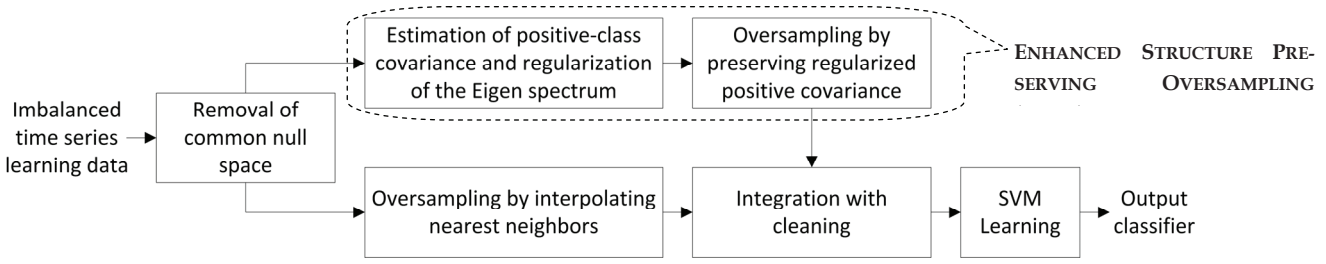


Fig. 2. Block diagram of the proposed integrated oversampling framework

and no risk of artificially introducing variances in the common null space; 2) The cleaning mechanism is redesigned to remove the "noise links" or pairs of positive and negative samples on the classification border with good efficiency; 3) We further reserve a small percentage of oversampling capacity for protective interpolation-based oversampling on the positive-class boundary, which produces better classification performance. For evaluation, we apply INOS with Support Vector Machines (SVM) classification to show that INOS outperforms current oversampling methods and that our classification results are highly competitive with other state-of-the-art methods for time series classification.

3 The Proposed Learning Framework

Fig. 2 shows the workflow of our proposed INOS framework. Given an imbalanced two-class time series dataset, we first remove the common null space by transforming the learning data into the *signal space*. Then, we perform a large portion of oversampling using enhanced structure preserving oversampling (ESPO): 1) Estimate the positive-class covariance and perform eigen decomposition and spectrum regularization; 2) Conduct oversampling by conforming to the regularized covariance structure. In parallel, a small portion of the remaining synthetic samples are generated using a nearest-neighbor interpolation method for oversampling. The resulting collective synthetic positive samples are then integrated to balance the original learning dataset. Here, the word "balance" refers to equal number of learning samples from each class and previously the work in [7] shows that the test error performance reduces when the learning data become more and more balanced through oversampling. After oversampling, we employ Support Vector Machines to learn a time series classifier from the balanced dataset. The details of the procedures are described in the following sections.

3.1 Removal of Common Null Space

Given the positive and the negative learning datasets, $P=\{\boldsymbol{x}_{11}, \boldsymbol{x}_{12}, \ldots, \boldsymbol{x}_{1|P|}\}$ and $N=\{\boldsymbol{x}_{01}, \boldsymbol{x}_{02}, \ldots, \boldsymbol{x}_{0|N|}\}$, where $|N| \gg |P|$, $\boldsymbol{x}_{ij} \in \mathbb{R}^{n\times 1}$ and n denotes the time series length or dimension, our oversampling algorithm first computes the total covariant matrix using

$$\mathbf{W}_T = \frac{\sum_{i=1}^{|P|}(\boldsymbol{x}_{1i}-\bar{\boldsymbol{x}})(\boldsymbol{x}_{1i}-\bar{\boldsymbol{x}})^T + \sum_{j=1}^{|N|}(\boldsymbol{x}_{0j}-\bar{\boldsymbol{x}})(\boldsymbol{x}_{0j}-\bar{\boldsymbol{x}})^T}{|P|+|N|} \quad (1)$$

where $\bar{\boldsymbol{x}} = \frac{1}{|P|+|N|}\left(\sum_{i=1}^{|P|}\boldsymbol{x}_{1i} + \sum_{j=1}^{|N|}\boldsymbol{x}_{0j}\right)$ is the mean vector. We perform eigen decomposition on $\mathbf{W}_T$ using

$$\mathbf{A} = \mathbf{L}^T\mathbf{W}_T\mathbf{L} \quad (2)$$

where $\mathbf{L}=[\boldsymbol{l}_1, \ldots, \boldsymbol{l}_j, \ldots, \boldsymbol{l}_n]$, $\boldsymbol{l}_j$ is the *j-th* eigenvector, and $\mathbf{A}$ is a diagonal matrix with the corresponding eigenvalues $a_1 \geq \ldots \geq a_j \geq \ldots \geq a_n$ organized in descending order. Very often, we can observe a string of trailing zeros among the eigenvalues, i.e. $a_j=0$ for $m<j\leq n$. This allows us to decompose the eigenvector space into two subspaces as:

$$\mathbf{L} = \begin{bmatrix}\mathbf{L}_s & \mathbf{L}_{nu}\end{bmatrix} \quad (3)$$

where $\mathbf{L}_s=[\boldsymbol{l}_1, \ldots, \boldsymbol{l}_m]$ consists of eigenvectors in the signal space and $\mathbf{L}_{nu}=[\boldsymbol{l}_{m+1}, \ldots, \boldsymbol{l}_n]$ forms the null space such that $\mathbf{W}_T\boldsymbol{l}_i = 0$ for all $i>m$. Previous works [24, 25] have pointed out that the common null space of the total covariance matrix does not contain useful information and can be effectively removed for enhanced discriminant performance. We therefore remove the null space through feature transformation. From Eqn (2) and (3), we write the following transformation for a training time series vector $\boldsymbol{x}_{ij}$,

$$\mathbf{L}^T\boldsymbol{x}_{ij} = \begin{bmatrix}\mathbf{L}_s & \mathbf{L}_{nu}\end{bmatrix}^T\boldsymbol{x}_{ij} = \begin{bmatrix}\mathbf{L}_s{}^T\boldsymbol{x}_{ij} \\ \mathbf{L}_{nu}{}^T\boldsymbol{x}_{ij}\end{bmatrix} \quad (4)$$

As $\mathbf{L}_{nu}{}^T\boldsymbol{x}_{ij}$ contains no data variance in the null space, it is easy to map $\mathbf{L}_{nu}{}^T\boldsymbol{x}_{ij} = \mathbf{L}_{nu}{}^T\bar{\boldsymbol{x}}$ and Eqn (4) becomes

$$\mathbf{L}^T\boldsymbol{x}_{ij} = \begin{bmatrix}\mathbf{L}_s{}^T\boldsymbol{x}_{ij} \\ \mathbf{L}_{nu}{}^T\bar{\boldsymbol{x}}\end{bmatrix} \quad (5)$$

By removing the constant term $\mathbf{L}_{nu}{}^T\bar{\boldsymbol{x}}$, we can simply use

$$\boldsymbol{q}_{ij} = \mathbf{L}_s{}^T\boldsymbol{x}_{ij} \quad (6)$$

to fully represent $\boldsymbol{x}_{ij}$ in a lower-dimensional signal space. Derived from Eqn (5), $\boldsymbol{x}_{ij}$ can also be fully recovered from $\mathbf{q}_{ij}$ using

$$\boldsymbol{x}_{ij} = \mathbf{L}\begin{bmatrix}\boldsymbol{q}_{ij} \\ \mathbf{L}_{nu}{}^T\bar{\boldsymbol{x}}\end{bmatrix} \quad (7)$$

Our removal of the null space is to use Eqn (6) to transform each of our training time series vectors to a low dimensional signal space. Note that in the scenario of high feature dimensionality, i.e. a large n, the number of zero eigenvalues can be significant. The transformation in Eqn (6) has two advantages: first, we can perform the oversampling computation more efficiently in the lower di-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

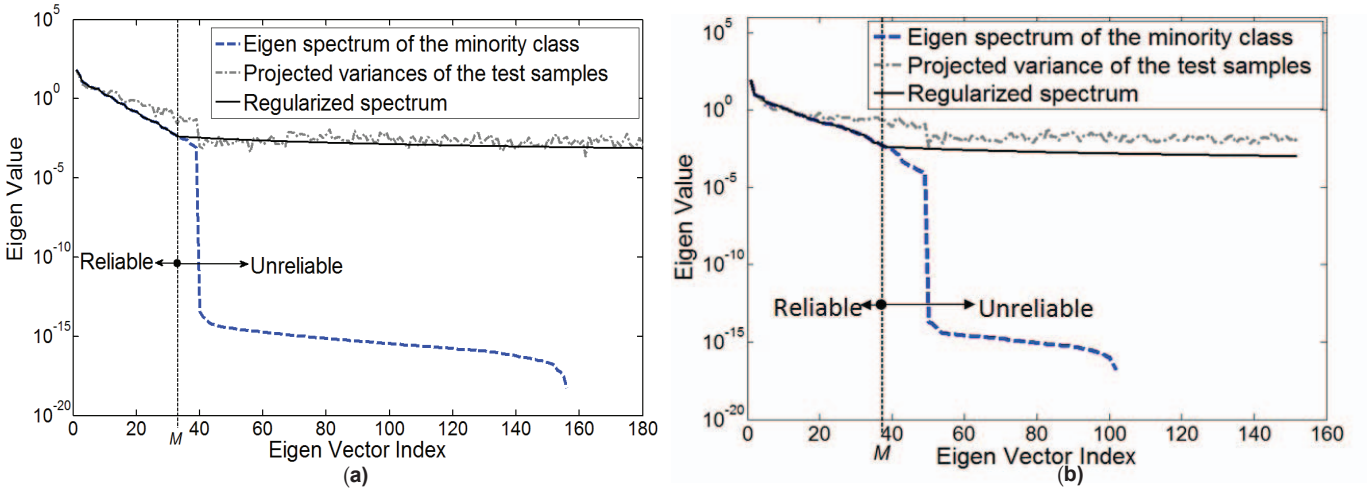


Fig. 3. Comparison of estimated minority-class spectrum, projection variance spectrum of test data and the regularized spectrum for Yoga and Wafer datasets in (a) and (b) respectively. One can refer to Table I for description of the datasets

mensional space; second, it eliminates the potential risk that the oversampling method artificially introduces data variance in the common null space, which originally contains no data variance.

3.2 Enhanced Structural Preserving Oversampling

3.2.1 Positive-Class Eigen Spectrum Regularization

After removing the common null space, we further compute the positive-class covariance matrix using

$$\mathbf{W}_P = \frac{1}{|P|}\sum_{j=1}^{|P|}\left(\boldsymbol{q}_{1j} - \bar{\boldsymbol{q}}_1\right)\times\left(\boldsymbol{q}_{1j} - \bar{\boldsymbol{q}}_1\right)^T \tag{8}$$

$\bar{\boldsymbol{q}}_1 = \frac{1}{|P|}\sum_{j=1}^{|P|}\boldsymbol{q}_{1j}$ denote the corresponding positive-class mean vector. On the positive training data covariance matrix $\mathbf{W}_P$, we then perform eigen decomposition using

$$\mathbf{D} = \mathbf{V}^T\mathbf{W}_P\mathbf{V} \tag{9}$$

where $\mathbf{D}$ is a diagonal matrix with the eigenvalues $d_1 \geq \ldots \geq d_j \geq \ldots \geq d_n$ organized in descending order and $\mathbf{V}=[v_1, \ldots, v_j, \ldots, v_m]$ is the corresponding eigenvector matrix. The eigenvectors in $\mathbf{V}$ satisfy $v_i^T\mathbf{W}_P v_j = 0$ for $i \neq j$. Here, covariance matrix $\mathbf{W}_P$ is a symmetric positive semi-definite matrix. In other words, $d_j \geq 0$ must be satisfied for $1 \leq j \leq m$ and d_j represents the projection variance on the j^{th} eigenvector for the positive training data. Suppose we have a large number of positive samples, all the eigenvalues $\{d_j\}$ shall be greater than zero and from Eqn (9), we derive

$$\mathbf{I}_m = \left(\mathbf{V}\mathbf{D}^{-1/2}\right)^T\mathbf{W}_P\left(\mathbf{V}\mathbf{D}^{-1/2}\right) = \mathbf{F}^T\mathbf{W}_P\mathbf{F} = \frac{1}{|P|}\sum_{j=1}^{|P|}\left(\mathbf{F}^T\boldsymbol{q}_{1j} - \mathbf{F}^T\bar{\boldsymbol{q}}_{1j}\right)\times\left(\mathbf{F}^T\boldsymbol{q}_{1j} - \mathbf{F}^T\bar{\boldsymbol{q}}_{1j}\right) \tag{10}$$

where $\mathbf{F} = \mathbf{V}\mathbf{D}^{-1/2} = [v_1/\sqrt{d_1}, \ldots, v_j/\sqrt{d_j}, \ldots, v_n/\sqrt{d_n}]$ represents a scaled transformation so that our transformed feature vectors $\{\mathbf{F}^T\boldsymbol{q}_{1j}, 1 \leq j \leq |P|\}$ of the positive samples have a covariance structure of an identity matrix $\mathbf{I}_m \in \mathbb{R}^{m\times m}$. Here, the transformation $\mathbf{F}$ will turn an arbitrary full-rank covariance matrix into a simple and well-known covariance structure of an identity matrix. This one-to-one mapping thus can be exploited to simplify the oversampling process; that is, we can generate the new samples based on an identity-matrix covariance structure and then map them into the targeted covariance structure using the inverse transformation of $\mathbf{F}$.

In many real world scenarios, the number of positive training samples ($|P|$) can be significantly less than the time series dimension n and even our reduced dimension m. Our estimated covariance structure cannot be fully trusted in such a case as it could over-adapt to the small set of positive time series data samples for learning. This can be observed in Fig. 3(a) and (b), where we find that the large eigenvalues are often good approximations to the projected variances of the test spectrum, but not the remaining portion of very small eigenvalues. Here, we compute the test spectrum by projecting a set of test positive samples onto each of eigenvectors $\{v_j\}$ and then calculating its corresponding projection variance. Given that the inverse of an eigenvalue is commonly used as a multiplicative term in feature scaling as seen in Eqn (5), the unreliable portion of the eigen spectrum is a major source of learning instability and poor generalization performance in discriminant feature extraction [22, 23] and in machine learning.

We aim to generate random synthetic samples that not only maintain the positive dataset's major covariance structure but also generalize well on the test dataset (so as to ensure accurate classification results subsequently). Therefore, we bring in a regularization step to fix the covariance eigen spectrum $\{d_j\}$ of the positive dataset. To achieve this, we need to divide the eigen spectrum into two regions, namely, the *reliable* and the *unreliable* subspace, and then perform regularization on the unreliable spectrum. Here, we perform a *c*-fold cross validation (CV) to determine the division location between the two subspaces (marked as M in Fig. 3). To do this, we randomly

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

divide the positive data into *c* equal partitions. One partition is reserved for testing and the remaining *c*-1 partitions are used for computing the eigen vectors and the eigen spectrum. The testing partition is then projected onto the eigen vectors to measure the test variance spectrum. We repeat this *c* times to use each partition as a test partition once. By averaging the *c* eigen spectrums and the *c* testing spectrums, a good *M* is located where the average eigen spectrum departs from the test spectrum. In this work, given that we our dataset can have as few as about ten positive samples in the evaluation experiments, a small *c*=2 is chosen to avoid tiny partitions.

With a fixed *M*, we then compute the regularized eigen spectrum using

$$\hat{d}_j = \begin{cases} \rho/(j+\theta), & \text{for } j > M \\ d_j, & \text{otherwise} \end{cases} \tag{11}$$

where $T(j) = \rho/(j+\theta)$ is a smooth eigen spectrum model in [22]. We use $T(1)=d_1$ and $T(M)=d_M$ to ensure the smooth transition and compatibility of the two spectrum regions with the two parameters determined as follows:

$$\rho = \frac{d_1 d_M (M-1)}{d_1 - d_M}, \quad \theta = \frac{M d_M - d_1}{d_1 - d_M} \tag{12}$$

3.2.2 Oversampling with Preserved Covariance

We generate a total of $(|N|-|P|)r$ synthetic positive samples (to contribute to a balanced population of positive and negative classes) using the regularized positive dataset's covariance structure and based on multivariate Gaussian distribution (MGD). Here, $r \in [0,1]$ is the integration percentage of synthetic samples contributed by ESPO, which is chosen empirically. The remaining (1-*r*) synthetic samples are generated by the interpolation-based oversampling method.

We chose MGD here because it is the most natural distribution. It is also well-known that summation of a large number of random and independent distributions obeys the Gaussian distribution [23]. Suppose $\hat{\mathbf{F}} = [\boldsymbol{v}_1/\sqrt{\hat{d}_1}, \ldots, \boldsymbol{v}_j/\sqrt{\hat{d}_j}, \ldots, \boldsymbol{v}_n/\sqrt{\hat{d}_n}]$ and assume that $\boldsymbol{z} = \hat{\mathbf{F}}(\boldsymbol{b} - \bar{\boldsymbol{q}}_1)$, the transformed version of the synthetic positive sample $\boldsymbol{b}$ to be generated, follows two MGDs of $\mathsf{N}(\mathbf{0}_M, \mathbf{I}_M)$ and $\mathsf{N}(\mathbf{0}_{m-M}, \mathbf{I}_{m-M})$ catering for the *reliable* and the *unreliable* eigen spectrum regions, respectively. Here, $\mathbf{0}_M$ denotes a vector of *M* zeros. We generate the two portions z_1 and z_2 separately. z_1 is generated from the MGD of $\mathsf{N}(\mathbf{0}_M, \mathbf{I}_M)$ and z_2 from $\mathsf{N}(\mathbf{0}_{m-M}, \mathbf{I}_{m-M})$. The two portions z_1 and z_2 are concatenated to form z. The synthetic sample in the signal space is then computed using

$$\boldsymbol{b} = \hat{\mathbf{D}}^{1/2} \mathbf{V}^T \boldsymbol{z} + \bar{\boldsymbol{q}}_1 \tag{13}$$

where $\hat{\mathbf{D}}$ is the diagonal matrix of regularized eigen values $\{\hat{d}_1, \ldots, \hat{d}_n\}$.

The above oversampling procedure is repeated until all $(|N|-|P|)r$ required synthetic samples are generated.

3.3 Oversampling through Interpolation Nearest Neighbor Pair

The synthetic samples generated by ESPO follow multivariate Gaussian distribution and are not tied closely with the original positive learning samples. Despite its many advantages (to be discussed in Sect. 3), it is still lacking in the aspect that original set of positive samples, especially their key (hard-to-classify) samples, are not well protected by the synthetic data generated. This is particularly critical for highly imbalanced learning tasks, where there would be a large capacity for oversampling.

To complement the synthetic population generated by ESPO, we propose to use interpolation-based method to oversample a second portion of $(|N|-|P|)(1-r)$ synthetic positive samples to provide additional protection of the existing positive learning samples. The interpolation-based methods, e.g. SMOTE [3], Borderline-SMOTE [6] or ADASYN [7], yield synthetic samples that are distributed closely around the seed samples from original positive set by generating synthetic samples through interpolating selected positive seed samples with their nearest neighbors found within the same class. We choose ADASYN [7] for its good adaptiveness in allocating quota for each seed sample based on the number of samples belonging to the opposite class (i.e. the Neg class) in its nearest *k*-neighbourhood. The larger this number, the harder the seed positive sample is classified correctly.

For improved efficiency, we execute the interpolation-based method in the transformed signal space rather than in the original feature space. Given our transformed datasets $P_t=\{\boldsymbol{q}_{1i}\}$ and $N_t=\{\boldsymbol{q}_{0j}\}$, and the remaining quota of $(|N|-|P|)(1-r)$ for oversampling, we perform the following:

1. For each positive sample $\boldsymbol{q}_{1i}$, find its *k* nearest neighbors $S_{i:k\text{-NN}}$ in the entire learning dataset;
2. Calculate the ratio:

$$\Gamma_i = |S_{i:k\text{-NN}} \cap N_t| / Z \tag{14}$$

   where *Z* is a normalization factor so that $\sum_{i=1}^{|P_t|} \Gamma_i = 1$.
3. Choose each sample $\boldsymbol{q}_{1i}$ as a seed in the positive class to generate $(|N|-|P|)(1-r)\Gamma_i$ synthetic positive samples. Each time for a seed sample $\boldsymbol{q}_{1i}$, we randomly selects one sample $\boldsymbol{q}_{1j}$ out of its *Q* (*Q*=5 as used in [3, 7]) nearest positive neighbors and interpolate a new sample $\boldsymbol{q}$ using,

$$\boldsymbol{q} = (1-\delta)\boldsymbol{q}_{1i} + \delta \boldsymbol{q}_{1j} \tag{15}$$

   where $\delta \in [0,1]$ is randomly chosen.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

3.4 Integration and Cleaning

We are now ready to integrate the oversampled population from ESPO and the interpolation-based method together with the original positive dataset to form an overall positive-class dataset P_o=$\{\boldsymbol{q}_{11}, \ldots, \boldsymbol{q}_{1|P|}, \boldsymbol{q}_{1|P|+1},\ldots, \boldsymbol{q}_{1|N|}\}$, where $\boldsymbol{q}_{1|P|+1},\ldots, \boldsymbol{q}_{1|N|}$ denote the synthetic samples generated. At the same time, we also have the transformed negative-class learning dataset N_o=$\{\boldsymbol{q}_{01}, \ldots \boldsymbol{q}_{0|N|}\}$. However, since oversampling could potentially generate outliers residing closely to the class boundary that unnecessarily impede the learning of a good decision boundary, we perform data cleaning as follows:

1. For a synthetic positive sample, $\boldsymbol{q}_{1i}$, where $i>|P|$, we find its nearest positive neighbor $\boldsymbol{q}_{1s}$ that satisfies $s=\text{argmin}_{j\neq i} f\left(\boldsymbol{q}_{1i},\boldsymbol{q}_{1j}\right)$, where $f(\cdot)$ denote a distance function. We use the simple Euclidean distance for this, as it is efficient in computation and also known as a suitable distance metric for time series classification [18, 19]. In addition, computing the Euclidean distance in our transformed signal space is identical to computing it in the original feature space;
2. We also find the nearest negative neighbor $\boldsymbol{q}_{0w}$ of $\boldsymbol{q}_{1i}$ that satisfies $w=\text{argmin}_j f\left(\boldsymbol{q}_{1i},\boldsymbol{q}_{0j}\right)$;
3. For $\boldsymbol{q}_{0w}$, we find its nearest positive-class neighbor $\boldsymbol{q}_{1e}$ that satisfies $e=\text{argmin}_j f\left(\boldsymbol{q}_{1j},\boldsymbol{q}_{0w}\right)$;
4. If $f\left(\boldsymbol{q}_{1i},\boldsymbol{q}_{0w}\right)<f\left(\boldsymbol{q}_{1i},\boldsymbol{q}_{1s}\right)$ and e=i, we record that $\boldsymbol{q}_{1i}$ and $\boldsymbol{q}_{0w}$ form a "noise link" on the decision boundary, which is similar to a Tomek link [5].
5. We repeat Steps 1 – 4 for each synthetic positive sample.
6. For all the recorded noise links, we remove their corresponding positive and negative samples from the oversampled learning dataset. Our concept of cleaning is similar to the previous work in [5], which removes the Tomek links to mitigate class overlapping for improved classification performance.

The above cleaning procedure differs from our previous cleaning mechanism in SPO [26] in the following aspects: 1) The distance computation and data cleaning are performed in the reduced m-dimensional signal space, making it more efficient; 2) The cleaning is performed after the process of oversampling is completed; and 3) it removes both positive and negative samples in pairs. Our previous approach performs cleaning after each sample is generated and only the positive synthetic samples are removed. In comparison, our new approach is less susceptible to those individual noisy negative samples that intrude into the positive-class territory, which are detected and removed by our new cleaning procedure. Furthermore, the new approach is more efficient in that we do not generate more than $|N|-|P|$ synthetic samples before cleaning is conducted, while previously, we generate more than this number of synthetic samples, depending on the percentage of synthetic positive samples that are cleaned. Also, as our cleaning mechanism removes positive and negative samples in pairs, it assures that the cleaned dataset is still balanced. Without these noisy links, the class boundary is usually better learned with standard learning algorithms [5].

After cleaning, we convert each of the remaining samples back into the original feature space using Eqn (7).

3.5 Learning Support Vector Machines Classifier

With the synthetically balanced dataset, we are now ready to learn our time series classifier. We chose Support Vector Machines (SVM) for learning our classifier for the following reasons: 1) SVM is known to provide excellent generalization to unseen data as it minimizes the structural risk instead of the empirical risk [28]; 2) The formulation of SVM allows users to choose different non-linear feature mapping to a high dimensional space so that a good linear separation hyper-plane can be found. Kernel tricks can also be employed in this process with acceptable computational loads; 3) In our preliminary classification test on balanced time series data, we found that the classification accuracy of SVM is comparable to the reported best results of 1NN-DTW, which is the state-of-the-art time series classification method, if the optimal learning parameters were chosen in conjunction with the common radial basis function (RBF) kernel.

Our SVM learning consists of the following steps: feature scaling, searching of best parameters, and solving an optimization problem [29]. We use

$$s_j=\frac{2\left(h_j-h_j^{(\text{Min})}\right)}{h_j^{(\text{Max})}-h_j^{(\text{Min})}}-1 \tag{16}$$

to linearly transform each feature h_j to the range of [-1, 1], where $h_j^{(\text{Max})}$ and $h_j^{(\text{Min})}$ are the maximum and minimum of the j-th feature in the training data respectively. This step helps reduce the risk that good features with relatively small numerical values being overshadowed by those having large values. The same scaling is also applied to all the test feature vectors in the testing scenario, where the same $\{h_j^{(\text{Max})}, h_j^{(\text{Min})}\}$ are used as those from the training dataset.

Next, the SVM classifier is learned by solving the optimization problem below [28]:

$$\begin{aligned}\min_{\boldsymbol{w},\boldsymbol{b},\xi}\quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}+C\sum_i\xi_i\\ \text{s.t.}\quad & \ell_i\left(\boldsymbol{w}^T\boldsymbol{\phi}\left(\boldsymbol{s}_i\right)+\beta\right)\geq 1-\xi_i\\ & \xi_i\geq 0\end{aligned} \tag{17}$$

where $\boldsymbol{w}$ defines the separation hyper-plane, C is a penalty constant for the total margin error, ξ_i is a slack variable denoting the margin error for the i-th selected support vector $\boldsymbol{s}_i$ from our scaled training set, ℓ_i is the class label of $\boldsymbol{s}_i$, and $\boldsymbol{\phi}\left(\boldsymbol{s}_i\right)$ denotes a nonlinear mapping to a high dimensional space. Only the kernel needs to be evaluated in solving the optimization problem. We use the radial basis function (RBF) kernel below

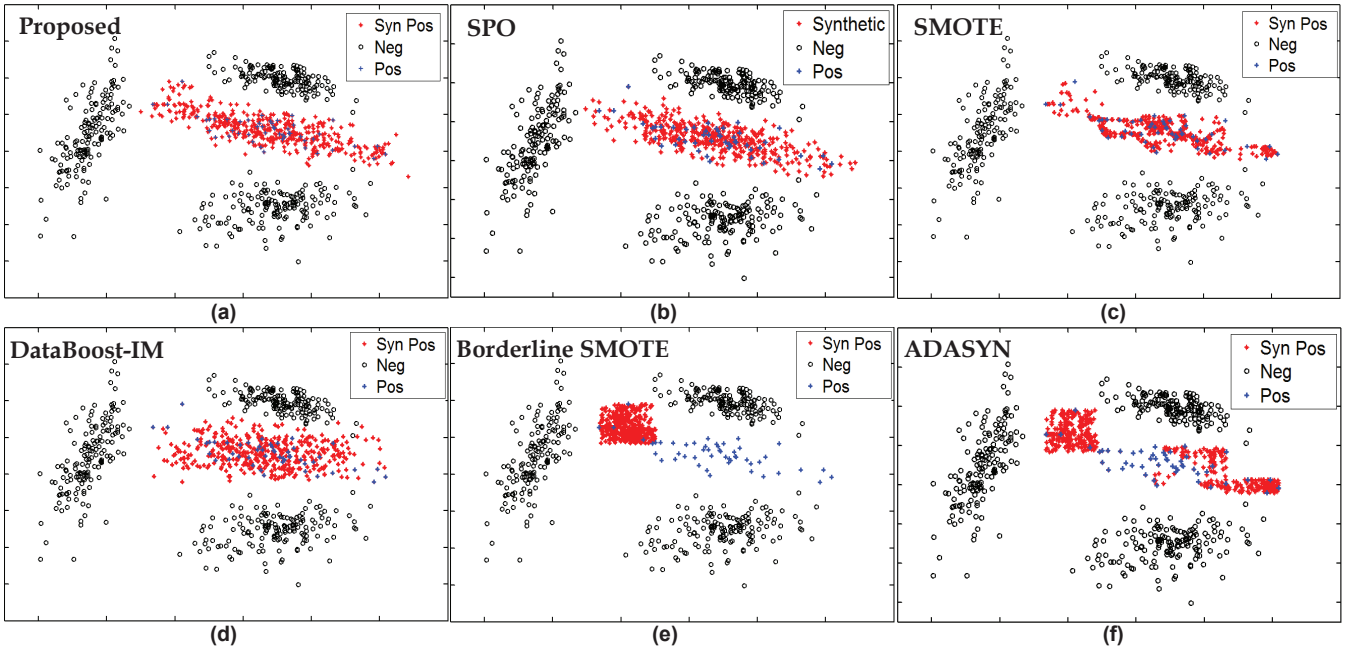


Fig. 4. Visual comparison of the proposed INOS with other oversampling techniques; (a) Proposed integrated oversampling; (b) our earlier work on structure preserving oversampling; (c) Synthetic minority over-sampling technique (SMOTE) [3]; (d) DataBoost oversampling [12]; (e) Borderline-SMOTE [6]; and (f) Adaptive synthetic sampling approach (ADASYN) [7]

TABLE 1
IMBALANCED TIME SERIES DATASETS

Datasets	#Apportions	Training			Test		Time series Length
		#Pos	*#Neg*	*IM-Ratio*	*#Pos*	*#Neg*	
Adiac	10	10	380	38	10~15	376~381	176
FaceAll	10	50	1000	20	62~223	977~1138	131
50Words	10	10~50	400	8~40	12~59	396~483	270
SLeaf	10	35	450	12.9	40~50	250~600	128
TwoPats	4	50	1800	36	1151~1256	1894~1999	128
Wafer	2	50	380-3000	7.6-60	712~6532	382~3402	152
Yoga	2	50	800~900	16~18	1480~1720	730~870	426

$$K(\boldsymbol{s}_i, \boldsymbol{s}_j) = \exp\left(-g\left\|\boldsymbol{s}_i - \boldsymbol{s}_j\right\|^2\right) \tag{18}$$

where $g \geq 0$ is a constant defining the kernel width. Note that the good choice of (C, g) is important for good SVM learning performance. As such, we search for the best (C, g) within the common ranges of these two parameters in a two dimensional log-scale grid, which gives the best cross validation performance. With the best (C, g), our SVM classifier is learned by solving Eqn (17) to classify the time series data.

4 Experimental Results and Discussions

We perform a series of experiments to evaluate our proposed integrated oversampling procedure against existing oversampling methods as well as state-of-the-art methods for imbalanced time series classification.

4.1 Visual Comparison

Based on the example in Fig. 1(b), we visually compare the oversampling effect of our proposed INOS with five methods in Fig. 4 in two-dimensional feature space. For each method, we oversample the set of 50 positive samples nine times to have a total of 500 positive samples (same size of negative examples).

Among all five existing oversampling methods, the synthetic samples generated from DataBoost [13] fills the void areas within the territory of positive samples most competently. However, the sample distribution appears square-like (i.e. in a different structure from the original positive data distribution) and intrudes into the territory of the negative class. The synthetic samples generated by the other three nearest-neighbor interpolation methods formed dense clusters with small data variances near the existing samples. Among them, SMOTE creates the most even synthetic distribution, since every existing positive sample has been selected for generating roughly the same number of new samples. Borderline-SMOTE creates a synthetic sample distribution where only a small set of existing positive samples, whose neighborhood contains more negative samples than the positive samples, are heavily emphasized. ADASYN strikes a balance between

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

TABLE II

AVERAGE SVM PERFORMANCE COMPARISON FOR DIFFERENT OVERSAMPLING METHODS. THE RESULTS ARE AVERAGED OVER MULTIPLE APPORTIONS AND TEN OVERSAMPLING EXPERIMENTS FOR EACH APPORTION

Evaluation Measure	Dataset	Oversampling Methods						
		Repeat	*SMOTE*	*B-SMOTE*	*ADASYN*	*DataBoost*	*SPO*	*Proposed INOS*
F-Value	*Adiac*	.663±.0005	.732±.0022	.718±.0018	.733±.0022	.728±.0012	.784±.0029	**.800±.0032**
	FaceAll	.903±.0002	.907±.0004	.904±.0004	.908±.0004	.925±.0005	**.936±.0004**	**.936±.0005**
	50Words	.774±.0014	.783±.0010	**.786±.0014**	.782±.0009	.769±.0008	.778±.0023	.778±.0026
	SLeaf	.897±.0003	.890±.0010	.894±.0011	.893±.0010	.888±.0007	**.906±.0015**	.904±.0014
	TwoPats	.296±.0015	.358±.0020	.352±.0018	.359±.0022	.371±.0022	.479±.0022	**.546±.0002**
	Wafer	.958±.0010	.965±.0020	.964±.0010	.966±.0015	.974±.0005	.983±.0005	**.985±.0015**
	Yoga	.653±.0015	.684±.0020	.677±.0025	.686±.0025	.678±.0025	.702±.0060	**.724±.0030**
	Average	.735±.0009	.760±.0015	.757±.0014	.761±.0015	.762±.0012	.796±.0023	**.810±.0020**
G-Mean	*Adiac*	.730±0	.807±.0001	.794±.0026	.809±.0013	.813±.0009	.872±.0014	**.882±.0023**
	FaceAll	.910±.0003	.916±.0004	.913±.0003	.917±.0004	.934±.0004	**.946±.003**	.945±.0004
	50Words	.851±.0006	.863±.0008	.862±.0008	.861±.0008	.839±.0004	.877±.0009	**.880±.0006**
	SLeaf	.931±.0004	.933±.0009	.936±.0008	.933±.0008	.938±.0004	.959±.0009	**.962±.0008**
	TwoPats	.417±.0001	.467±.0015	.462±.0001	.468±.0013	.477±.0015	.561±.0015	**.614±.0020**
	Wafer	.959±.0005	.966±.0010	.965±.0010	.967±.0005	.974±.0005	.984±.0005	**.985±.0005**
	Yoga	.695±.0015	.720±.0025	.714±.0010	.721±.0020	.713±.0010	.733±.0025	**.750±.0030**
	Average	.785±.0006	.810±.0011	.807±.0011	.811±.0010	.813±.0007	.847±.0011	**.860±.0014**
Precision	*Adiac*	**.924**±.0004	.828±.0012	.837±.0011	.824±.0012	.809±.0009	.806±.0026	.820±.0022
	FaceAll	**.995±.0001**	.992±.0003	.991±.0003	.992±.0003	.983±.0004	.982±.0003	.981±.0004
	50Words	**.857±.0010**	.846±.0007	.853±.0010	.847±.0006	.849±.0006	.806±.0017	.806±.0014
	SLeaf	**.926±.0002**	.907±.0008	.908±.0009	.910±.0007	.891±.0004	.889±.0009	.879±.0008
	TwoPats	.991±.0012	**.993±.0016**	**.993±.0015**	**.993±.0017**	.992±.0018	.978±.0017	.971±.0015
	Wafer	**.999±.0011**	**.999±.0018**	**.999±.0011**	**.999±.0014**	.998±.0005	.998±.0013	.996±.0015
	Yoga	.991±.0013	.991±.0017	**.994±.0018**	.992±.0022	.987±.0020	.989±.0039	.986±.0026
	Average	**.955±.0008**	.936±.0012	.939±.0011	.937±.0012	.930±.0010	.921±.0018	.920±.0015
Recall (or True Positive Rate)	*Adiac*	.592±.0006	.673±.0018	.657±.0016	.677±.0017	.676±.0013	.780±.0035	**.792±.0038**
	FaceAll	.833±.0003	.843±.0005	.839±.0006	.844±.0006	.876±.0007	**.898±.0007**	.896±.0007
	50Words	.741±.0017	.763±.0013	.760±.0016	.760±.0012	.722±.0010	.788±.0028	**.794±.0024**
	SLeaf	.872±.0004	.876±.0009	.883±.0014	.877±.0014	.888±.0008	.927±.0014	**.935±.0015**
	TwoPats	.175±.0010	.219±.0015	.215±.0013	.220±.0015	.228±.0018	.323±.0022	**.386±.0018**
	Wafer	.921±.0020	.934±.0035	.930±.0020	.934±.0030	.952±.0010	.970±.0015	**.975±.0020**
	Yoga	.487±.0015	.522±.0025	.513±.0025	.524±.0030	.517±.0030	.545±.0075	**.573±.0040**
	Average	.660±.0011	.690±.0018	.685±.0016	.691±.0017	.694±.0014	.747±.0028	**.764±.0024**
True Negative Rate	*Adiac*	**.998±0**	.996±.0001	.997±.0001	.996±0	.995±.0001	.995±.0002	.995±.0001
	FaceAll	**1±0**	.999±0	.999±0	.999±0	.998±.0001	.998±.0001	.998±0
	50Words	.991±0	.991±.0001	.991±.0001	.991±.0001	**.993±.0001**	.987±.0002	.986±.0003
	SLeaf	**.995±0**	.994±.0001	.994±.0001	.994±.0001	.992±.0001	.992±.0001	.991±.0001
	TwoPats	**.999±0**	**.999±.0001**	**.999±.0001**	**.999±0**	**.999±.0001**	.997±.0002	.994±.0002
	Wafer	**1±0**	**1±0**	**1±.0001**	**1±.0001**	.997±.0005	.998±.0004	.996±.0004
	Yoga	.992±.0003	.991±.0005	**.994±.0009**	.992±.0006	.985±.0005	.987±.0008	.983±.0012
	Average	**.997±.0001**	.996±.0001	.996±.0002	.996±.0001	.994±.0002	.993±.0003	.992±.0003

SMOTE and Borderline-SMOTE by adaptively emphasizing more towards the samples that are closer to the classification border. Our earlier work, SPO, shared the most similar covariance shape with the existing limited positive samples. Its synthetic samples not only adequately fill the void internal spaces within the positive class territory but also sensibly expand towards the void areas in the vicinity of positive-class territory which are not currently occupied by the abundant negative samples. However, we can also see that SPO does not place any extra emphasis in its oversampling to protect the existing hard-to-classify samples that are close to the classification border.

The effects of our proposed INOS method in integrating ESPO and interpolation–based oversampling can be seen clearly. The synthetic samples created by INOS largely shared similar covariance shape with the existing limited positive samples. In addition, there were more synthetic positive samples generated near the decision boundary to protect the existing positive samples that are hard to classify.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

TABLE III
SVM PERFORMANCE COMPARISON (ADIAC DATASET) FOR DIFFERENT OVERSAMPLING METHODS

Eval. Measure	P-CI	Oversampling Methods						
		REP	*SMO*	*BoS*	*ADA*	*DB*	*SPO*	***INOS***
F-Value	*1*	**.842**	.789	.785	.770	.643	.831	.834
	2	.600	.500	.500	.500	.571	.623	**.704**
	3	.750	.750	.75	.750	.75	**.971**	.924
	4	**.900**	**.900**	**.900**	**.900**	.842	.838	.879
	5	.182	.573	.563	.583	.500	.642	**.700**
	6	.615	.686	.600	.668	.705	.779	**.750**
	7	.778	.749	.749	.749	.632	.684	**.784**
	8	.125	.417	.382	.437	**.640**	.517	.499
	9	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	10	.842	.955	.955	**.975**	1	.952	.931
G-Mean	*1*	.893	.892	.892	.891	.831	.924	**.925**
	2	.679	.619	.619	.619	.678	.731	**.795**
	3	.775	.775	.775	.775	.775	**.999**	.927
	4	**.947**	**.947**	**.947**	**.947**	.893	.946	**.947**
	5	.316	.705	.705	.705	.680	.806	**.889**
	6	.698	.773	.712	.773	.773	.845	**.846**
	7	.836	.811	.811	.811	.772	.833	**.869**
	8	.258	.567	.502	.588	**.728**	.634	.644
	9	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	10	.893	.979	.979	.984	**1**	.999	.979

The acronyms are: REP: repeating; SMO: SMOTE; BoS: Borderline SMOTE. ADA: ADASYN; DB: DataBoost oversampling. P-CI: index of the selected positive-class in the original dataset.

TABLE IV
SVM PERFORMANCE COMPARISON (YOGA DATASET) FOR DIFFERENT OVERSAMPLING METHODS

Eval. Measure	P-CI	Oversampling Methods						
		REP	*SMO*	*BoS*	*ADA*	*DB*	*SPO*	***INOS***
F-Value	*1*	.667	.667	.663	.670	.654	.680	**.696**
	2	.639	.701	.690	.702	.701	.725	**.752**
G-Mean	*1*	.705	.706	.703	.708	.697	.705	**.729**
	2	.684	.733	.725	.734	.729	.762	**.771**

4.2 Comparison for Time Series Classification

For evaluation, we constructed a total of 48 imbalanced two-class data apportions from seven balanced benchmark datasets in the UCR time series repository [17], as tabulated in Table I. Each apportion here represents a standalone binary imbalanced learning task. Please refer to [17, 18] for the detailed descriptions of these datasets. Note that *S-Leaf* here refers to the "*Swedish Leaf*" dataset. We have chosen these datasets since they contain large number of samples to facilitate the simulation of scenarios of large class imbalance.

Out of the seven datasets, *Adiac*, *FaceAll, 50words and S-Leaf* originally contained 37, 14, 50 and 15 classes, respectively. In each of its apportions, we converted them into two-class datasets by selecting one class as the positive class and using the remaining classes to form the negative class. For each set, the training and test data are divided randomly and all available samples are included either in the training or in the test set. As shown in Table I, we chose round numbers for the positive training samples and made sure that they do not exceed 50% of the total available positive samples. For each training set, the number of positive samples is kept no more than 50 to simulate the scenario of rare positive instances. We maintained high imbalance ratios (IM-ratios), i.e. the number of negative samples divided by the number of positive samples, in the datasets, with the highest being 60 for the *Wafer* dataset and the lowest being 7.6 for the second apportion of the *Wafer* dataset. Note that for all datasets, the number of positive samples is significantly smaller than the time series length (dimension of the time series). In particular, for *Adiac*, the feature dimension is about 18 times of the number of positive samples. This sparsity of data with respect to the high dimensions in time series classification is another key challenge. For reproducibility of our results, we have shared our 48 apportioned datasets online (http://sites.google.com/site/sstarcao/).

To evaluate the effectiveness of our INOS for oversampling, we compared it with six oversampling methods (five different existing methods and our earlier SPO method) to balance the classes in the training dataset separately, and then trained a SVM classifier for each resulting balanced dataset using the steps in Sec. II. The performance of the classifiers is evaluated using various metrics shown in Eqn (19) below. In particular, *F-Value* and *G-Mean* are well-reported performance metrics for imbalanced learning.

Actual Class \ Predicted Class	***Positive***	***Negative***
Postive	*TP* (True Positives)	*FN* (False Negatives)
Negative	*FP* (False Positives)	*TN* (True Negatives)

$$\begin{aligned} &F\text{-}Value: && \frac{2\times Recall\times Precision}{Recall+Precision} \\ &G\text{-}Mean: && \sqrt{\frac{TP\times TN}{(TP+FN)\times(TN+FP)}} \\ &Precision: && TP/(TP+FP) \\ &Recall\,(True\ Positive\ Rate): && TP/(TP+FN) \\ &True\ Negative\ Rate: && TN/(FP+TN) \end{aligned} \quad (19)$$

The results presented in Table II shows that our proposed INOS achieved *F-Value* and *G-Mean* of 0.81 and 0.86, respectively, which are significantly better than all six existing oversampling methods. On a closer examination of our results, we found that our good results are largely attributed by excellent recall scores, with comparable precisions and true negative rates. The high recall rates indicate that most of the test positive samples can be classified with a good accuracy when using INOS to supplement the limited positive training datasets. In comparison, oversampling by repeating achieved the largest precision score at the expense of the poorest recall value. This is due to the tendency that a smaller set of positive predictions at a high accuracy rate is made by the corresponding SVM (i.e. overfitting). Besides our proposed INOS method and our earlier work SPO, ADASYN achieved the third best pair of *F-Value* and *G-Mean* results. This shows that the adaptive approach of ADASYN

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

TABLE V

SVM PERFORMANCE COMPARISON FOR INTEGRATING DIFFERENT PAIRS OF OVERSAMPLING METHODS. THE PERCENTAGE SHOWN INDICATES THE BEST EMPIRICAL INTEGRATION PERCENTAGE FOR THE FIRST INDICATED METHOD OF THE PAIR

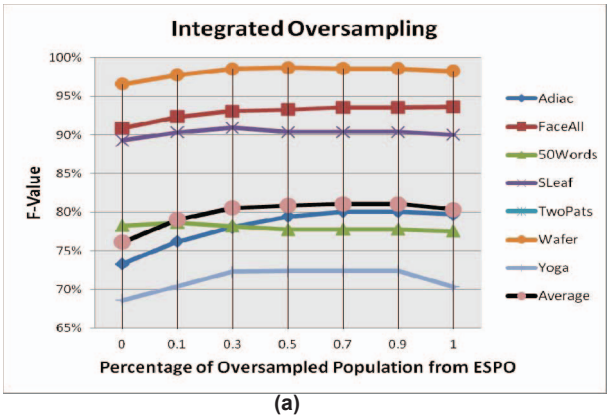
Eval. measure	Dataset	Integrated oversampling distribution				
		ESPO + REP 100%	*ESPO + SMO 70%*	*ESPO + BoS 70%*	***Proposed INOS 70%***	*DB+ ADA 30%*
F-Value	*Adiac*	.797	.799	.798	**.800**	.737
	FaceAll	**.936**	.934	.935	**.936**	.923
	50Words	.776	.775	.774	.778	**.793**
	SLeaf	.900	.901	.902	**.904**	.890
	TwoPats	.530	.544	.543	**.546**	.377
	Wafer	.982	.985	**.986**	.985	.983
	Yoga	.704	.721	.722	**.724**	.703
G-Mean	*Adiac*	.875	.876	.876	**.882**	.823
	FaceAll	**.947**	.944	.944	.945	.932
	50Words	**.880**	.877	.877	**.880**	.873
	SLeaf	**.962**	.961	.961	**.962**	.939
	TwoPats	.601	.613	.613	**.614**	.482
	Wafer	.982	**.985**	**.985**	**.985**	.983
	Yoga	.733	.748	.748	**.750**	.733

in choosing the seeds is also quite effective for dealing with highly imbalanced time series classification.

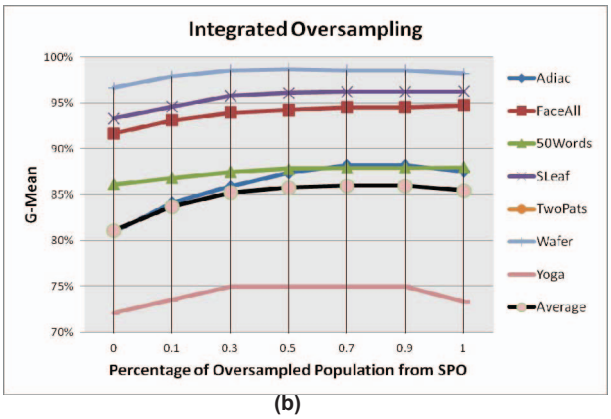
We examined our achieved results on the individual apportions for two particularly interesting datasets, *Adiac* and *Yoga*, as tabulated in Table III and IV, respectively. As we can see from Table 1, each of the *Adiac* apportions has only 10 positive samples, which is very small as compared to its time series length of 176. It thus represents a challenging (but quite common) scenario in which the positive learning samples are extremely scarce. In contrast, the *Yoga* apportions have the longest time series length of 426, which is significantly larger than that for the remaining six datasets. The classification results in Table III show that out of the ten *Adiac* apportions, our INOS's oversampling achieved the most competitive *F-value* and *G-Mean* for five and seven times, respectively, representing the most number of wins among all seven different oversampling methods. For the two *Yoga* apportions, INOS also achieved the best *F-value* and *G-Mean*, which outperform the second best set of results from SPO. These suggest that our proposed INOS is most competitive for imbalance correction for the two challenging learning tasks.

4.3 Comparison with Other Types of Integration

Table II has suggested that the oversampling methods by repeating, SMOTE, Borderline-SMOTE and ADASYN tend to give a stronger precision value, while SPO and DataBoost oversampling are clearly stronger in giving better recall results. By visual inspection of Fig. 4, we have also seen that the synthetic distributions from SPO and DataBoost are not tied closely to existing samples, while the remaining four methods tend to tie their synthetic distribution closer to the existing training positive samples. This suggests that better results may be obtained



(a)



(b)

Fig. 5. F-Value versus the integration percentage *r* in (a) and G-Mean versus the integration percentage in (b) for our proposed INOS. Here, integration percentage *r* refers to percentage of synthetic samples generated by ESPO

if the complementary characteristics for the two groups of oversampling methods can be combined in an integrative approach, and our INOS method has been designed to do so. Fig. 5 shows the performance of INOS with different integration percentages, i.e. with *r* increasing from 0 to 1 with a step of 0.1. The *F-Value* and *G-Mean* for most of the datasets continued to improve until contributions from both methods reached 50% or more. From Fig. 5, on average, the *F-Value* and the *G-Mean* peaked at an integration percentage of $r \in [0.7, 0.9]$, i.e. 70%-90% of the synthetic positive samples are from ESPO and the remaining percentage from ADASYN. In other words, INOS generates most of the synthetic positive samples following the covariance structure of existing positive samples, and generates 10-30% positive samples which protect the key original minority samples. In fact, we observed from Fig. 5 that when $r \in [0.5, 0.9]$, the experimental results are very stable across all the 7 datasets, indicating that selecting a suitable *r* value for good performance is not an issue. In all our experiments, we fixed r=0.7 for INOS.

Table V compares the results achieved by other types of integrations. We can observe that ESPO used in integration with existing oversampling methods also improves the performance in terms of *F-value* and *G-Mean*.

4.4 Evaluation of Separate INOS Procedures

We also performed experiments to evaluate how much improvement was gained by each of the sub-procedures in our INOS algorithm. We compared the full INOS with

the following cases: (i) no positive dataset oversampling being performed, (ii) ESPO without eigen spectrum regularization in Eqn (11), (iii) ESPO without cleaning in Sect. 2.3, (iv) ESPO without feature scaling by Eqn (16), (v) ESPO with uniform distribution, (vi) ESPO and (vii) proposed INOS. For Case (i), the SVM classifier was learnt from the original imbalanced training set (with feature scaling). For each case, we perform logscale search for the best combination of (C, g), which gives the highest G-Mean in five-fold cross-validation. For the non-regularization Case (ii), we replaced $\hat{\mathbf{D}}$ in Eqn (13) with $\mathbf{D}$, diagonal matrix with the unregularized eigen spectrum $\{d_j\}$, to generate the synthetic samples. For Case (ii), uniform distribution is chosen instead of multivariate Gaussian distribution to generate random synthetic samples.

Without oversampling, the average *F-Value* and *G-Mean* over the seven datasets are 72.9% and 77.7% respectively, which are apparently lower than the average *F-Value* and *G-Mean* of the various oversampling cases shown in Fig. 6 (a) and (b). INOS achieved an average *F-Value* of 0.81, which is 0.7%, 2.8%, 3.1%, 0.7% and 1.4% higher than the cases of ESPO alone, ESPO with uniform distribution, ESPO without scaling, ESPO without data cleaning and ESPO without regularization, respectively. The average *G-Mean* for our INOS is 0.860, which is also 0.5%, 2.4% 1.3%, 0.5% and 1.1% higher than the cases of ESPO alone, ESPO with uniform distribution, ESPO without scaling, ESPO without data cleaning and ESPO without regularization, respectively. The improved results showed that each of the four INOS sub-procedures, namely integration of ESPO with interpolated based method, eigen regularization, feature scaling and data cleaning, are important for enabling INOS to attain better learning outcomes for imbalanced time series datasets.

4.5 Comparison of Different Learning Methods

Finally, we compared our proposed learning framework in Fig. 2 with two recently developed imbalanced learning methods, EasyEnsemble [8] and BalanceCascade [8], as well as with two other well-known state-of-the-art methods, 1NN-DTW [16] and 1NN [18, 19], for time series classification. Like our INOS method, EasyEnsemble and BalanceCascade also addressed the data imbalance issue at the data level. However, unlike our oversampling strategy, EasyEnsemble and BalanceCascade adopted the approach of undersampling the negative class to achieve balance in the training datasets. In both methods, multiple balanced sets were constructed to learn multiple decision-tree [30] weak classifiers. These classifiers were then systematically integrated by AdaBoost into a strong ensemble classifier. The difference between EasyEnsemble and BalanceCascade is that BalanceCascade periodically removes a pre-computed percentage of relatively easier negative samples from the overall negative training set before the random undersampling takes place. This is to enforce more hard-to-classify negative samples being included in the final balanced training set. A similar method to EasyEnsemble was also proposed recently in [27], except that bagging was used to integrate the weak classifiers learnt from the undersampled rebalanced datasets. 1NN-DTW and 1NN do not explicitly address the data imbalance issue since their performance is less susceptible by data imbalance. As mentioned earlier, 1NN-DTW has been shown to be highly competent for classifying balanced time series data [16]. The 1NN classifier with a simple Euclidean distance metric, which has also been popularly suggested for semi-supervised time series classification tasks [18, 19], involves utilizing a large portion of unlabelled samples to improve the learning outcomes.

Table VI shows the comparison of our proposed INOS cum SVM learning with other time-series learning methods in terms of *F-Value* and *G-Mean*. While the performance is dependent on the individual datasets, our proposed method showed good results in terms of ranking by scoring the best performance for the most number of times, i.e. four and four times for a total of seven datasets in terms of *F-Value* and *G-Mean* respectively. Except for the *FaceAll, TwoPats* and *Yoga* datasets which INOS was

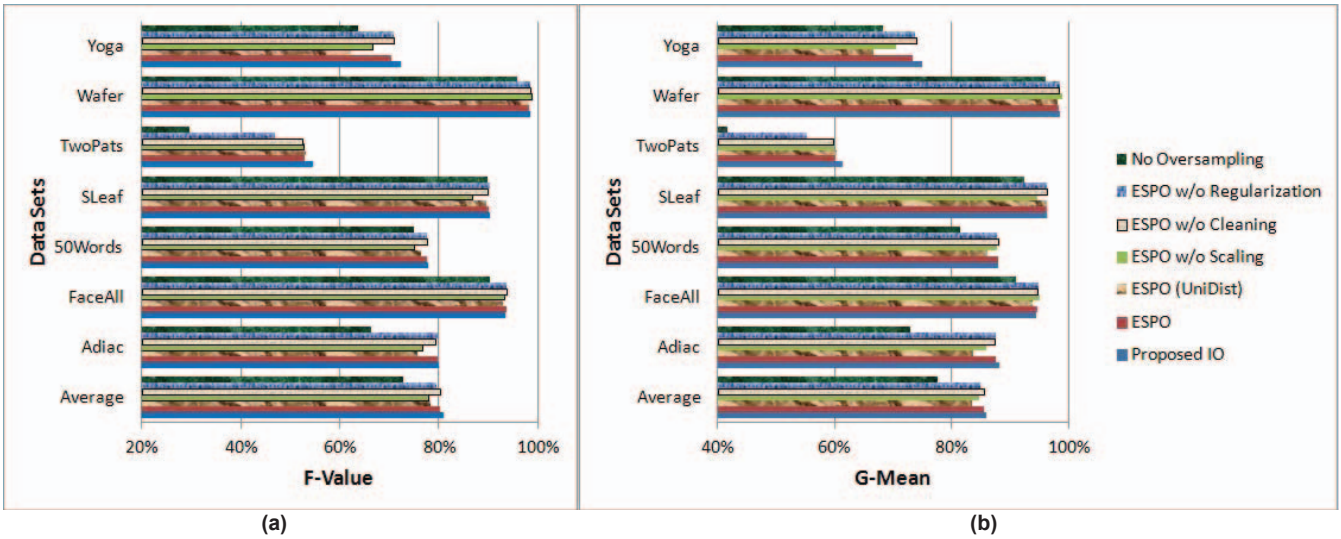


Fig. 6. INOS performance comparison with the separate cases when no data cleaning, ESPO alone with no eigenspectrum regularization, ESPO alone with no data cleaning, ESPO with no feature scaling, ESPO alone, i.e. with no integration of ESPO with ADASYN. (a) F-value comparison; (b) G-mean comparison

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

TABLE VI
PERFORMANCE COMPARISON OF SEVERAL CONVENTIONAL METHODS

Eval. measure	Dataset	Learning methods				
		Easy	*Bal.*	*1NN*	*1NN DTW*	*INOS*
F-Value	*Adiac*	.464	.297	.621	.603	**.800**
	FaceAll	.675	.806	.910	**.946**	.935
	50Words	.579	.482	.768	.696	**.778**
	SLeaf	.699	.692	.774	.787	**.904**
	TwoPats	.211	.605	.608	**1**	.546
	Wafer	.808	.962	.975	.930	**.985**
	Yoga	.322	**.760**	.599	.673	.724
	Average	.537	.658	.751	.805	**.810**
G-Mean	*Adiac*	.669	.847	.736	.752	**.882**
	FaceAll	.719	.936	.936	**.973**	.945
	50Words	.684	.864	**.890**	.881	.879
	SLeaf	.783	.938	.874	.876	**.962**
	TwoPats	.344	.657	.661	**1**	.614
	Wafer	.825	.966	.976	.925	**.985**
	Yoga	.435	.711	.651	.709	**.750**
	Average	.637	.846	.818	**.874**	.860

The acronyms are: Easy: EasyEnsemble; Bal: BalanceCascade; 1NN: One nearest neighbor classifier using Euclidean distance; 1NN DTW: One nearest neighbor classifier using dynamic time warping distance; INOS: Proposed integrated oversampling, where majority of positive synthetic samples are generated by enhanced structural preserving oversampling with support vector machines classifier

ranked as the 2nd, the 4th and the 2nd places, respectively, INOS achieved the best *F-Value* consistently for the remaining four datasets, *Adiac*, *50Words*, *SLeaf* and *Wafer*. In terms of *G-Mean*, our proposed INOS is ranked the best for *Adiac*, *SLeaf*, *Wafer* and *Yoga*, the second best for *FaceAll*, the third for *50Words,* and the fourth for *TwoPats*. These ranking results showed that our method is highly competitive compared with the existing methods on highly imbalanced time series classification.

Amongst the existing methods, we noticed that 1NN-DTW gave the best results for the *FaceAll* and *TwoPattern* datasets. In particular, for *TwoPats*, its performance (*F-Value*, *G-Mean*) of (1, 1) is remarkablely better than (0.608, 0.661), the second-best result from 1NN. Yet, despite its superiority found on such time series datasets, its good performance was not consistent and its counterpart 1NN performed better than 1NN-DTW at times, e.g. on the *50Words* and *Wafer* datasets.

BalanceCascade provided significantly better average *F-Value* and *G-Mean* than its counterpart EasyEnsemble. This is in agreement with the discussion in [8] that BalanceCascade is more suitable for highly imbalanced data than EasyEnsemble. Other than the good ranking of INOS, its average results showed that it is in a comparable range to the results of 1NN-DTW. However, the good average performance of 1NN-DTW is mainly contributed by its excellent margin on *TwoPats* dataset.

We have also conducted comparison experiments on another 7 UCR datasets including *CBF*, *Synthetic Control*, *OSU Leaf*, *Fish*, *Gun-Point*, *Trace*, and *ECG*, which contain relatively less samples for our simulation of highly imbalanced learning scenarios. The results (available at: https://sites.google.com/site/sstarcao/publication/journal-papers) demonstrate similar good performance for our proposed INOS method as it achieved the highest average *F-Value* and average *G-Mean* amongst the various oversampling-based time series classification methods and the non-oversampling based methods.

4.6 Computation Efficiency

In our current framework, the number of synthetic samples to be generated by the proposed INOS is dependent on size of the negative class. The oversampling and the subsequent learning can be inefficient when the negative training class is extremely large, e.g. with billions of samples. Nevertheless, in such a situation, we can reduce the negative class to a manageable yet representative size through undersampling using existing methods [3, 4]. Our proposed method can then be applied to the modified training set on a smaller scale for efficiency.

As tabulated in Table VII, using our most populous time series dataset, *Wafer*, our current MATLAB implementation takes an average of 2.1×10^{-2} and 1.5×10^{-2} second for SPO [26] and our proposed INOS, respectively, to create a synthetic sample of 152 dimensions using a ordinary computer with 2.79-GHz CPU. For *Yoga*, which has the longest time series length, it took 1.7×10^{-1} and 5.0×10^{-2} second, respectively, for SPO and our INOS to create a sample of 426 dimensions. For *TwoPats*, it took 1.0×10^{-1} and 2.7×10^{-2} second, respectively, for SPO and our INOS to create a sample of 128 dimensions. For all seven types of time series datasets, it took an average of 4.7×10^{-2} and 1.6×10^{-2} second to create a sample. This shows that INOS has significantly reduced the oversampling time needed for SPO at per-sample basis by 28.6% for *Wafer*, 70.6% for *Yoga*, 76.8% for *TwoPats* and 66.0% for the average case. This is largely contributed by the following factors. 1) The main oversampling computation is now conducted in the signal space with a feature dimension of m instead of the original dimension of n. The complexity of our algorithm thus reduces to $O\left(\left(|N|-|P|\right)m^2\right)$ from the original $O\left(\left(|N|-|P|\right)n^2\right)$. This reduction is particularly significant for the dataset with long time series length. For instance, m is only about 56% and 29% of n for *50words* and *Yoga* datasets, respectively, after the null-space removal. 2) For some dataset, e.g. *TwoPats*, we found that the rate of rejecting a synthetic sample is high using our previous-SPO's cleaning mechanism. Therefore, significantly more synthetic samples were needed to be created in order to fulfill a required number of $|N|-|P|$ samples that can survive through the cleaning mechanism. On the other hand, our newly designed ESPO will not generate more than $|N|-|P|$ samples.

Out of all the sub-procedures of our proposed INOS, we found that our cleaning process required the most processing time, i.e. more than 90% of the total oversampling time on average. By taking away the cleaning procedure, our INOS took only 4.7×10^{-4} second for *Wafer*, 2.5×10^{-3} second for *Yoga*, 3.6×10^{-4} second for *TwoPats,* and an average of 8.8×10^{-4} second for over the seven datasets,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

to create one synthetic sample. Thus, in situations where efficiency is the most critical factor, INOS can be used without the cleaning procedure at a slight expense on the *F-value* and *G-mean* performance as shown in our Sect. 3.4.

After oversampling, it took 1.1 seconds to train the SVM classifier using the balanced *Wafer* dataset of 6000 samples and 2.2×10^{-4} second to classify a test sample. The relatively small time requirements for oversampling and SVM classification suggest that our proposed algorithm can be used for many practical time series classification tasks.

TABLE VII
COMPARISON OF REQUIRED TIME (10^{-3} SECOND) FOR GENERATING A SYNTHETIC SAMPLE USING SPO AND INOS

Datasets	Dimension	SPO	INOS	Reduction (%)	INOS without cleaning
Adiac	176	7.5	5.1	31.9	0.71
FaceAll	131	12.4	9.1	27.3	0.42
50Words	270	13.9	7.3	47.9	1.28
SLeaf	128	6.0	5.9	2.8	0.45
TwoPats	128	102.5	23.7	76.9	0.36
Wafer	152	20.7	15.1	26.9	0.47
Yoga	426	168.5	43.7	74.1	2.47
Average	176	47.4	15.7	66.9	0.88

5 CONCLUSION

In this paper, we have proposed a novel integrated oversampling method INOS for the challenging learning problem of imbalanced time series classification. In the current work, INOS addresses the imbalanced learning issue by oversampling the minority class with a hybrid approach in the signal space. The main portion of synthetic samples is generated by ESPO in eigen decomposed subspace and is based on regularized eigen spectrum. This allows our resulting synthetic sample data to preserve the main covariance structure of the original minority-class samples and at the same time, to incorporate some protective variances in the trivial eigen dimensions. The synthetic samples are able to fill up the gaps between the minority-class samples and sensibly expand into the vicinity in a consistent shape with the existing positive samples. A smaller portion of the synthetic samples is also created with interpolation-based method with an aim to protect the existing samples that are hard to classify.

Based on seven public sets and forty eight apportions of highly imbalanced UCR time series data, our INOS with SVM achieved good average *F-Value* and *G-Mean* of 0.81 and 0.86, respectively. It outperformed an array of existing oversampling methods as well as state-of-the-art learning methods for time series data. We have also shown that each of our suggested procedures, namely, integration of ESPO with an interpolation-based method, eigen spectrum regularization, feature scaling and data cleaning, are effective in enabling INOS to attain better learning outcomes for imbalanced time series datasets.

Our results are particularly significant given that many real-world data mining applications are afflicted with data imbalance and involve time series data, but there have been relatively limited work on imbalanced time series learning. Our results with INOS showed that by taking into careful consideration of the specific issues related to time series data, such as preserving covariance structure and providing extra emphasis on distribution of the existing hard samples, the oversampling approach can be employed to effectively address the challenging problem of data imbalance in time series classification.

While our proposed INOS have achieved generally good results, we also noticed that some time series datasets are certainly better classified with 1NN-DTW algorithm (e.g. *TwoPats*). Our future work will investigate the characteristics of different time series datasets and design meta-learning algorithm that predicts the best classification methodologies for each time series dataset.

REFERENCES

[1] N.V. Chawla, N. Japkowicz and A. Kolcz, "Editorial: Special issue on learning from imbalanced data sets," ACM SIGKDD Explorations Newsletter, vol. 6(1), 2004, pp. 1-6.

[2] H. He and E.A. Garcia, "Learning from imbalanced data," IEEE Trans. on Knowledge and Data Engineering, vol. 21-9, Sept. 2009, pp. 1263-1284.

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

[4] A. Estabrooks, T. Jo and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," Computational Intelligence, vol. 20, 2004, pp. 18-36.

[5] G.E.A.P.A. Batista, R.C. Prati and M.C. Monard, "A Study of the behavior of several methods for balancing machine learning training Data," ACM SIGKDD Explorations Newsletter, vol. 6(1), 2004, pp. 20-29.

[6] H. Han, W.Y. Wang and B.H. Mao, "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning," Proc. Int. Conf. Intelligent Computing, 2005, pp. 878-887.

[7] H. He, Y. Bai, E.A. Garcia and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," Proc. Int. Conf. Neural Networks, 2008, pp. 1322-1328.

[8] X.-Y. Liu, J. Wu and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," IEEE Trans. on System, Man and Cybernetics, vol. 39 (2), Apr. 2009, pp. 539-550.

[9] Y. Sun, M.S. Kamel, A.K.C. Wong and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," Pattern Recognition, vol. 40 (12), Dec. 2007, pp. 3358-3378.

[10] H. Cao and A.C. Kot, "Manipulation detection on Image Patches using FusionBoost," IEEE Trans. on Information Forensics and Security, vol. 7 (3), June 2012, pp. 992-1002,

[11] N.V. Chawla, A. Lazarevic, L.O. Hall and K.W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," Proc. European Conf. Principles and Practice of Knowledge Discovery in Databases, 2003, pp. 107-119.

[12] B. Raskutti and A. Kowalczyk, "Extreme re-balancing for SVMs: a case study," ACM SIGKDD Explorations Newsletter, vol. 6(1), 2004, pp.60-69

[13] H. Guo and H.L. Viktor, "Learning from imbalanced data sets with boosting and data generation: The DataBoost-IM approach," ACM SIGKDD Explorations Newsletter, vol. 6(1), 2004, pp. 30-39.

[14] S. Chen, H. He and E.A. Garcia, "RAMOBoost: Ranked minor-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

ity oversampling in boosting," IEEE Trans. on Neural Networks, vol. 21 (10), Oct. 2010, pp. 1624-1642.

[15] M.A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," Proc. Int. Conf. Machine Learning, Workshop Learning from Imbalanced Data Sets II, 2003.

[16] X. Xi, E. Keogh, C. Shelton and L. Wei, "Fast time series classification using numerosity reduction," Proc. Int. Conf. on Machine Learning, 2006, pp. 1033-1040.

[17] E. Keogh, X. Xi, L. Wei and C.A. Ratanamahatana (2006), UCR time series classification/clustering page, www.cs.ucr.edu/~eamonn/time_series_data

[18] L. Wei and E.J. Keogh, "Semi-supervised time series classification," Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Aug. 2006, pp. 748-753

[19] M.N. Nguyen, X.-L. Li and S.-K. Ng, "Positive unlabeled learning for time series classification," Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'11), Jul. 2011

[20] S. Koknar-Tezel and L.J. Latecki, "Improving SVM classification on imbalanced time series data sets with ghost points," Knowledge and Information Systems – KAIS, vol. 24(2), 2010, DOI: 10.1007/s10115 -010-0310-3

[21] R. Pearson, G. Goney and J. Shwaber, "Imbalanced clustering for microarray time-series," Proc. Int. Conf. Machine Learning, Workshop Learning from Imbalanced Data Sets II, 2003

[22] X. Jiang, B. Mandal and A.C. Kot, "Eigenfeature regularization and extraction in face recognition," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 30(3), Mar. 2008, pp. 383-394.

[23] X. Jiang, "Linear subspace learning-based dimensionality reduction," IEEE Signal Processing Magazine, vol. 28(2), Mar. 2011, pp. 16-26.

[24] W. Liu, Y. Wang, S.Z. Li and T. Tan, "Null space-based kernel Fisher discriminant analysis for face recognition," Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition, 2004, pp. 369-374.

[25] R. Huang, Q.S. Liu, H.Q. Lu and S.D. Ma, "Solving the small sample size problem of LDA," Proc. of Int. Conf. on Pattern Recognition, vol. 3, 2002, pp. 29-32.

[26] H. Cao, X.-L. Li, Y.-K. Woon and S.-K. Ng, "SPO: Structure preserving oversampling for imbalanced time series classification," Proc. Int. Conf. on Data Mining (ICDM'11), pp. 1008-1013

[27] B.C. Wallace, K. Small, C.E. Brodley and T.A. Trikalinos, "Class imbalance, redux," Proc. Int. Conf. on Data Mining (ICDM'11), pp. 754-763

[28] V. Vapnik, The Nature of Statistical Learning Theory. Springer-Verlag, New York, 1995.

[29] C.-W. Hsu, C.-C. Chang and C.-J. Lin, "A pratical guide to support vector classification," 2008.

[30] J.R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1(1), 1986, pp. 81-106.

Dr. Hong CAO (S'08-M'11) is currently a data analytics scientist in Institute for Infocomm Research (I^2R) of Singapore's Agency of Science, Technology and Research (A*STAR), since 2011. He received the first-class honors B.Eng, and Ph.D degrees from Nanyang Technological University, Singapore, in 2001 and 2010, respectively. His current research interest includes machine learning, mobile data analytics, time series data mining, and multimedia forensics. His previous work in image forensics received the best paper award in IWDW 2010 and the honorary mention in ISCAS 2010. Recently, he also won international and local benchmarking challenges such as Opportunity Activity Recognition Challenge 2011 and Up-Singapore Hackathon 2012. He currently serves as secretary for IEEE signal processing society, Singapore section.

Dr. Xiaoli LI is currently a machine learning lab head in the Data Analytics Department at the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore. He also holds an appointment of adjunct assistant professor in the School of Computer Engineering, Nanyang Technological University. His research interests include data mining, machine learning and bioinformatics. He has been serving as a member of technical program committees in the leading data mining and machine learning conferences KDD, ICDM, SDM, PKDD/ECML, PAKDD, WWW, ACML, AAAI, and CIKM etc. He is the co-Editor-in-Chief of the International Journal of Knowledge Discovery in Bioinformatics (IJKDB). He received a number of best paper awards in international conferences (DASFAA 2011, GIW 2005) and won some global benchmarking awards (DREAM 2007, Opportunity Activity Recognition Challenge 2011). To learn more about Dr. Xiao-Li Li, please visit his Web page: http://www1.i2r.a-star.edu.sg/~xlli/.

Dr. David Yew-Kwong WOON is the Head of Operations of EADS Innovation Works South Asia (IW-SA) since July 2012, looking after operational aspects of the centers in Singapore and India as well as managing partnerships and governmental relations in the South Asia region. He is also the Head of the Energy Cluster, charting future directions and growing research activities in the areas of smart energy management systems and alternative energy. He started his career in 2004 as a Senior Officer at Singapore's Agency for Science, Technology & Research (A*STAR), assisting on the development of R&D strategies and partnerships for the various research institutes under the Science and Engineering Research Council (SERC) and managing research grants. In May 2006, he joined EADS as the Deputy Chief Operating Officer of IW-SA, the first research centre of EADS outside of Europe. After helping to set up the centre, he went back to research as Project Leader (Advanced Data Analysis) in January 2009 while still being partially involved in management work, focusing on establishing strategic and research partnerships in Singapore. Dr. Woon graduated from the Nanyang Technological University (NTU) in Singapore in 2000 with a bachelor's degree in Applied Science with Honors (First Class). He went on to obtain a Ph.D. in Computer Engineering from NTU in 2004. His research interests include association rule mining, classification, cluster analysis, stream mining and Bayesian networks. He has published several papers in top journals and conferences including the IEEE Transactions on Knowledge and Data Engineering (TKDE), the IEEE International Conference on Data Mining (ICDM) and the ACM Conference on Information and Knowledge Management (CIKM). Dr. Woon is married with one son and is an avid writer. He has a diploma in Freelance Journalism and blogs at www.davidwoon.com/blog in his free time.

Dr. See-Kiong NG (Ph.D., Carnegie Mellon University) is the Programme Director of the Urban Systems Initiative for Singapore's Agency of Science, Technology and Research (A*STAR) and the Center Director of the Business Analytics Translational Center. He also holds a concurrent appointment as a Principal Scientist and the Advisor to the Data Analytics Department at A*STAR's Institute for Infocomm Research. See-Kiong has a long-standing interest in cross-disciplinary applied computer science research. His primary research is in machine learning and data mining, with diverse applications in text mining, bioinformatics, privacy-preserving data mining and social network mining.