Model-Based Oversampling for Imbalanced Sequence Classification

Zhichen Gong, Huanhuan Chen*
University of Science and Technology of China, Hefei, China
zcgong@mail.ustc.edu.cn, hchen@ustc.edu.cn

ABSTRACT

Sequence classification is critical in the data mining communities. It becomes more challenging when the class distribution is imbalanced, which occurs in many real-world applications. Oversampling algorithms try to re-balance the skewed class by generating synthetic data for minority classes, but most of existing oversampling approaches could not consider the temporal structure of sequences, or handle multivariate and long sequences. To address these problems, this paper proposes a novel oversampling algorithm based on the 'generative' models of sequences. In particular, a recurrent neural network was employed to learn the generative mechanics for sequences as representations for the corresponding sequences. These generative models are then utilized to form a kernel to capture the similarity between different sequences. Finally, oversampling is performed in the kernel feature space to generate synthetic data. The proposed approach can handle highly imbalanced sequential data and is robust to noise. The competitiveness of the proposed approach is demonstrated by experiments on both synthetic data and benchmark data, including univariate and multivariate sequences.

Keywords

Imbalanced learning; Model space; Oversampling; Sequence classification

1. INTRODUCTION

Imbalanced learning aims to tackle the adverse influence on learning algorithms raised by the (relative or absolute) bias of the size of different classes [1, 2, 3]. Imbalanced dataset has at least one class where the number of examples is far less than others so that the traditional classifiers usually favor the majority, ignoring the rare yet interesting target class. With the increasing requirements of big data applications, this problem has been becoming more challenging [3].

Roughly speaking, the approaches to imbalanced learning can be divided into two main streams, i.e., cost sensitive learning [2] and resampling [4, 1]. Cost sensitive learning assigns different penalties for the misclassification of different classes so that more penalties will be given if points in minority classes are misclassified to majority classes. Resampling based learning methods are to oversample the minority and undersample the majority [4], which can be further divided into random oversampling/undersampling [3], synthetic oversampling/undersampling [4] and other variants, e.g. data cleaning approaches [3]. Oversampling has the advantage of more flexibility [4], since it alleviates the problem from the data level. This paper will investigate oversampling methods for binary sequence classification. Without loss of generality, we assume that the positive class is the minority class.

A sequence is a series of observations from a certain dynamical process. Sequence learning is ubiquitous in the field of health care [5, 6], genetics [7], engineering control [8], music [9], video prediction [10] etc. Sequences may be of high dimensionality and varying lengths. Sequences differ from normal vectorial data because they usually contain temporal dynamics over time.

When sequence classification meets imbalanced classification, it becomes more challenging for traditional classification algorithms to handle. For example, in medical diagnosis, the DNA sequence of patients with a particular disease is much more rare than that of the healthy controls [5, 6].

To re-balance the class distribution, classical interpolation-based oversampling methods are in the danger of generating synthetic minority samples in improper regions, e.g. the region of the majority class(es). This may damage the original data distribution and cause over-fitting [3]. Besides, existing oversampling approaches usually ignore the temporal information of sequences, which limits their capability in real-world applications.

The most widely used oversampling algorithms include random repetition [3], SMOTE [4], BorderSMOTE [11], ADASYN [12] and INOS [13] etc. Random repetition simply replicates the minority data points to enrich the minority class, which might lead to over-fitting [3]. SMOTE generates new samples evenly in the minority region [4]. BorderSMOTE gives more emphasis to points near the class border [11]. ADASYN makes use of the proportion of majority points in the K nearest neighbors of a minority point as its sampling probability [12]. However, these methods do not match the temporal dynamics of sequences.

*Corresponding author.

CIKM'16 , October 24 - 28, 2016, Indianapolis, IN, USA

ISBN 978-1-4503-4073-1/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2983323.2983784

INOS re-scales the original sequences in each time step to prevent the synthetic data from damaging the original covariance structure [13]. To calculate the covariance among different time steps, INOS requires that the sequences be univariate and of equal length, which might be hard to satisfy in most real-world applications. Besides, the high computational complexity of the covariance limits the scalability of INOS to long sequences.

A sequence is generally generated from a generative model. It is un-intuitive to directly sample synthetic data in the data space. This paper considers generating new synthetic data according to the generative models of sequences. Our work is motivated by representation learning [10, 8] and kernel learning [14, 15]. In this paper, we propose an algorithm that generates new synthetic data in a model-spanned representation space. In particular, we first learn representations for sequences to take advantage of the sequential order information using a special kind of recurrent neural network (RNN). Then the sequence representations are utilized to compute a kernel to preserve the similarity constraints. The representations and kernel parameter are tuned iteratively for better representation and discrimination. Finally, in the learned kernel feature space, oversampling is performed.

The rest of this paper is organized as follows. Section 2 introduces the background. Section 3 details the proposed algorithm. In Section 4, experiments are performed using both synthetic data and benchmark data. Section 5 concludes the paper.

2. BACKGROUND

Traditional classification algorithms usually have difficulty in dealing with imbalanced sequence datasets. They potentially assume that the class distributions are balanced. However, in imbalanced scenarios, the majority class benefits more than the minority class. It overwhelms the minority class and pushes the classifier boundary to the minority region. This results in high false negative rate and low recall. Oversampling gives a solution by generating new synthetic data for minority class to re-balance the class distribution. Existing oversampling approaches mainly sample the original signal space of sequences [3].

It is notable to point out that existing approaches in this line do not concern modeling the true generative mechanism of the original data [3, 4]. By contrast, the proposed approach in this work is to learn generative models for sequential data. Then it samples the model-spanned space [8], which is safer for simple yet efficient interpolation based oversampling techniques, such as SMOTE. Because our approach is based on RNN and kernel learning, we will discuss the background on these two topics.

2.1 Recurrent Neural Network and Reservoir Computing

A recurrent neural network (RNN) takes account of the combinational influence of the past input history and the current input for each time step [9]. The autocorrelation of the sequence can be effectively preserved in this manner. In contrast to Markov process [16], which assumes a hard forgetting point for the past history, RNN does not specify an explicit memory length. This implicit memory ability enables RNN a powerful tool in text processing [17], music data mining [9], video analysis [10] and speech processing [16] etc.

Reservoir model [18] is a kind of RNN. It has a nonlinear sparse recurrent network, called reservoir, which offers versatile dynamical features for the input sequence if designed properly. For more details about the reservoir model, please refer to [18].

Learning in the model space [8, 19] tries to use the generative mechanism of data to build a model for each data item. Learning algorithms are then performed in the space spanned by these models. Distance between model functions is treated as the distance of original data [8]. Our work is similar to model-based kernel [19] in terms of taking advantage of kernel learning techniques, but differs from that in the way we learn the generative model and kernel [19].

2.2 Kernel Target Alignment

Kernel learning is important in support vector machine [20, 21]. A kernel function $K : R^D \times R^D \rightarrow R$ implicitly transforms the data into the infinite dimensional Hilbert space, encapsulating more nonlinearity and separability. There are a set of possible kernel functions available, such as Gaussian kernel, polynomial kernel etc. However, to determine a proper kernel function and its parameters for a task is not easy. Instead of trying different kernel functions, directly learning a kernel has been actively invested recently [14, 15]. Kernel target alignment learns a kernel by defining a similarity between kernel matrices [22]. Centered kernel target alignment [15] has been shown to be independent of data distribution, which makes it extremely suitable for imbalanced learning.

Previous oversampling approaches in imbalanced data classification usually ignore the generative mechanism of data. So the generated synthetic data may be not consistent with the true data dynamics. As a remedy, we propose to learn generative models for sequential data as representations. Oversampling is performed in the model space instead of the original signal space.

3. LEARNING REPRESENTATIONS AND KERNEL

3.1 Learning Generative Models for Sequences

Representation learning uses a generative model, such as a neural network, to learn representations for images, text [10] etc. In this work, we choose the reservoir model as the base model because it enables the proposed approach: 1) to learn parsimonious representations for sequences. 2) to take advantage of the structure information of sequences using the relative sequential order. 3) to perform learning with fewer parameters.

Echo state network [18] (ESN) is a kind of reservoir model, which constitutes an un-trainable recurrently connected dynamical reservoir and a trainable linear readout mapping. The reservoir is specified randomly but s.t. the maximum eigenvalue of reservoir weight matrix is less than 1. For a given task, the reservoir is usually selected by restarting and cross-validation. The readout mapping is learned by linear regression, trying to approximate the target sequence.

Assume the generative mechanism of sequences is time invariant, the fading memory of an echo state network is competent to capture this generative distribution.

To justify the randomness of ESN, Rodan et al. [23] proposed a deterministically built alternative, i.e. cycle reser-

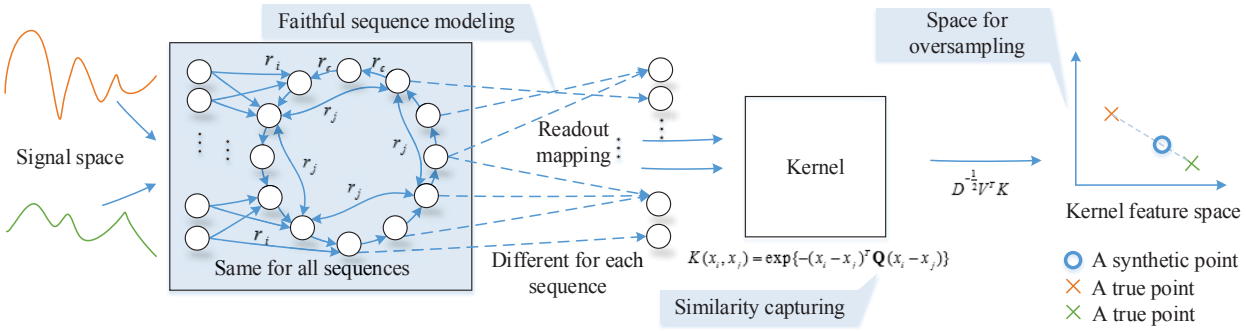

Figure 1: The graph demonstration of oversampling in the generative model-spanned space.

voir with jumps (CRJ). Different from ESN, the neurons in CRJ reservoir are uni-directional connected, forming a circle; and bi-directional connected, forming jumps on the circle. The input connections share the same weight $r_i > 0$. The sign is determined randomly [23]. All the cycle connections share a same value $r_c > 0$. Jump connections share the same weight $r_j > 0$. By constraining the network structure, the number of parameters is significantly reduced. Figure 1 demonstrates the key steps of this work.

The form of CRJ is generalized as:

$$\begin{cases} \mathbf{S}(t+1) = g(\mathbf{R}\mathbf{S}(t) + \mathbf{V}\mathbf{X}(t+1)) \\ f(t) = \mathbf{W}\mathbf{S}(t) \end{cases} \tag{1}$$

where $\mathbf{S}(t) \in R^N$ is the reservoir state, N is the number of neurons in the reservoir; $\mathbf{X}(t) \in R^{n+1}$ is the input with an additional bias term, $t \in \{1, 2, \cdots, T\}$, T is the length of the sequence, n is the number of variates; $\mathbf{R} \in R^{N \times N}$ is the reservoir weight matrix, $\mathbf{V} \in R^{N*(n+1)}$ is the input weight, $\mathbf{W} \in R^{O*N}$ is the output weights, O is the number of output neurons; g is the state transition function, which is a nonlinear function and usually taken as *tanh* or the sigmoid function.

Given a sequence $\mathbf{X} = \{\mathbf{X}(1), \mathbf{X}(2), \cdots, \mathbf{X}(T)\}$ as the input, where T is the length of the sequence, the reservoir neurons take account of the input memory to collect the sequential order information and convert the input into the reservoir state space. The reservoir state not only offers the current input information but also remembers the cumulative correlation along the temporal axis. Denote the collection of target output items as $\mathbf{Y}$ and the reservoir state as $\mathbf{S}$. The readout mapping is trained by one-step-forward prediction. Thus $\mathbf{Y}(t) = \mathbf{X}(t+1)$ in this scenario. Then

$$\mathbf{W} = \arg\min_{\mathbf{W}} E(\mathbf{W}) = \arg\min_{\mathbf{W}} \sum_{t=1}^{T-1} \left\| \hat{f}(t) - \mathbf{Y}(t) \right\|^2 + \eta ||\mathbf{W}||^2,$$

where $\hat{f}(t)$ is the prediction function (Equation 1). By taking advantage of ridge regression, $\mathbf{W}$ can be calculated as:

$$\mathbf{W} = \mathbf{Y}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T + \eta\mathbf{I})^{-1}, \tag{2}$$

where $\mathbf{I}$ is the identity matrix, $\eta > 0$ is the regularization parameter. The reservoir network (Equation 1) provides approximations for the given sequences and the readout mappings $\hat{f} = \mathbf{W}\mathbf{S}$ are models of the original sequences.

We take the model parameters as representations. The representations are used to compute a kernel matrix. The benefit of using kernel techniques here is two-fold: 1) It provides a training approach that is not sensitive to imbalanced data distribution (See the following subsection). 2) It incorporates the desired similarity constraints into the training process. We consider a Gaussian kernel in this paper:

$$K(x_i, x_j) = exp\{-(x_i - x_j)^T \mathbf{Q}(x_i - x_j)\} \tag{3}$$

where $\mathbf{Q}$ is the mahananobis distance metric. The kernel parameters are optimized to encourage the representations in the same class to stay closely and the representations in different classes to stay apart. By doing so, the data becomes more separated, and more likely to be linearly separable in the kernel feature space. Note that interpolation based oversampling algorithms potentially assume that the class distribution is linearly separable. Otherwise, the synthetic data may lie in the region of the opposite class.

3.2 Learning the Kernel

The goal is to minimize the difference between the empirical kernel matrix and an ideal kernel matrix. The ideal kernel [15] matrix is defined as:

$$K^*(x_i, x_j) = \begin{cases} 1 & \text{if } x_i, x_j \text{ are in the same class} \\ 0 & \text{otherwise} \end{cases}$$

Therefore, the ideal kernel provides desired similarity constraints of representations. Kernel target alignment learns a kernel by approximating the ideal kernel.

Centered kernel target alignment is proved to be not sensitive to class distribution [15], which makes it competent for imbalanced learning problems. In contrast to kernel target alignment [22], which only considers preserving the general similarity of classes, centered kernel target alignment weighs different classes and takes into account if a class is underrepresented in a kernel. Let $R = \mathbf{1}_{N_{tr}}\mathbf{1}_{N_{tr}}^T \times 1/N_{tr}$ be a square matrix of size $N_{tr} \times N_{tr}$, where each entry is $1/N_{tr}$ and N_{tr} is the size of training set. In particular, centering process is formed as [15]:

$$K_C = K - KR - RK + RKR.$$

The similarity of representation formed kernel matrix and the ideal kernel matrix are defined in [15]:

$$A(K_C, K_C^*) = \frac{(K_C, K_C^*)_F}{\sqrt{(K_C, K_C)_F (K_C^*, K_C^*)_F}} \in [-1, +1]$$

where $(K_C, K_C^*)_F = Tr(K_C^T K_C^*)$. It enhances the discrimination of data to the requirement of the ideal kernel by

maximizing the similarity between the kernel matrix K and ideal kernel K^*.

The sequence representation learning and centered kernel alignment is performed iteratively. The cost function is defined as:

$$C(\mathbf{r},\theta) = A(K_C, K_C^*) - \lambda \sum_{m=1}^{N_{tr}} \sum_{t=1}^{T_m-1} (f_m(t) - \mathbf{X}^m(t+1))^2 \quad (4)$$

where $\mathbf{r} = \{r_i, r_j, r_c\}$ and $\theta = \mathbf{Q}$ are the network parameters and kernel parameter respectively, X^m is the training sequence indexed by m, T_m is the length of the sequence indexed by m, N_{tr} is the size of the training set, $\lambda > 0$ is the combinational coefficient; the first term on the right-hand side is the alignment between the representation formed kernel matrix and the ideal kernel matrix, the second term is the difference between the output of learned models and the desired output. In doing so, we take a trade-off of the representations and the alignment into consideration.

3.3 Parameter Learning

To align the representation formed kernel matrix with the desired kernel matrix, we learn the representations and tune the kernel parameter iteratively.

To learn faithful representations of sequences, the kernel parameter ($\mathbf{Q}$) is initialized as an identity matrix and kept fixed. The representations learned by the reservoir model should approximate the sequence well.

The network weights are tuned to learn representations using gradient-based optimization:

$$\begin{aligned}\frac{\partial C}{\partial r} &= \frac{\partial A}{\partial r} - 2\lambda \sum_{m=1}^{N_{tr}} \sum_{t=1}^{T_m-1} (f_m(t) - \mathbf{X}^m(t+1))^T \frac{\partial f_m(t)}{\partial r} \\ &= \frac{\partial A}{\partial K_C}\frac{\partial K_C}{\partial \mathbf{W}}\frac{\partial \mathbf{W}}{\partial r} - \\ &\quad 2\lambda \sum_{m=1}^{N_{tr}} \sum_{t=1}^{T_m-1} (f_m(t) - \mathbf{X}^m(t+1))^T \frac{\partial f_m(t)}{\partial r} \end{aligned} \quad (5)$$

where $r \in \{r_i, r_c, r_j\}$ is the connection weights of CRJ.

In particular, the first term in the above equation is computed as

$$\frac{\partial A}{\partial K_C} = \frac{K_C^*}{\sqrt{Tr(K_C^T K_C)}} - \frac{Tr(K_C^T K_C^*)}{(Tr(K_C^T K_C))^{\frac{3}{2}}} \quad (6)$$

$$\frac{\partial K_C(i,j)}{\partial \mathbf{W}_i} = 2K_C(i,j)(\mathbf{W}_j - \mathbf{W}_i) \quad (7)$$

$$\begin{aligned}\frac{\partial \mathbf{W}}{\partial r} &= \mathbf{Y}\frac{\partial \mathbf{S}^T}{\partial r}(\mathbf{S}\mathbf{S}^T + \eta\mathbf{I})^{-1} + \mathbf{Y}\mathbf{S}^T\frac{\partial (\mathbf{S}\mathbf{S}^T + \eta\mathbf{I})^{-1}}{\partial r} \\ &= \mathbf{Y}\frac{\partial \mathbf{S}^T}{\partial r}(\mathbf{S}\mathbf{S}^T + \eta\mathbf{I})^{-1} - \mathbf{Y}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T + \eta\mathbf{I})^{-1} \\ &\quad \left(\frac{\partial \mathbf{S}}{\partial r}\mathbf{S}^T + \mathbf{S}(\frac{\partial \mathbf{S}}{\partial r})^T\right)(\mathbf{S}\mathbf{S}^T + \eta\mathbf{I})^{-1} \end{aligned} \quad (8)$$

where $\frac{\partial \mathbf{S}}{\partial r}$ is computed at each time step using real time recurrent training:

$$\begin{aligned}\frac{\partial \mathbf{S}(t)}{\partial r} &= \sec h^2(\mathbf{R}\mathbf{S}(t-1) + \mathbf{V}\mathbf{X}(t)) \\ &\quad .* \left(\mathbf{R}\frac{\partial \mathbf{S}(t-1)}{\partial r} + \frac{\partial \mathbf{V}}{\partial r}\mathbf{S}(t)\right)\end{aligned} \quad (9)$$

where $.*$ is the elementwise production. Then, using the above equations (Equation 8 and 9), we have

$$\frac{\partial f_m(t)}{\partial r} = \mathbf{W}\frac{\partial \mathbf{S}}{\partial r} + \frac{\partial \mathbf{W}}{\partial r}\mathbf{S} \quad (10)$$

By using Equation 6,7,8,9 and 10, the overall cost function (Equation 5) can be optimized. The reservoir network is trained to fit the data.

Then the representations are fixed. As a valid mahananobis distance, $\mathbf{Q}$ must be symmetric positive semi-definite matrix. By making use of the fact $\mathbf{Q} = \mathbf{U^T U}$, we instead optimize $\mathbf{U}$. Denote the kernel parameter as θ. θ is learned by gradient ascent:

$$\frac{\partial A(K,K^*)}{\partial \theta} = \frac{1}{Tr(K_C^*, K_C^*)} \left(\begin{array}{c} \frac{Tr(\frac{\partial K}{\partial \theta}, K_C^*)}{Tr(K_C, K_C)} - \\ \frac{Tr(K, K_C^*) Tr(K_C, \frac{\partial K}{\partial \theta})}{Tr(K_C, K_C)^3} \end{array} \right) \quad (11)$$

The derivative of the kernel matrix with respect to the kernel parameter is then:

$$\frac{\partial K(i,j)}{\partial \mathbf{U}} = -K(i,j) \times 2\mathbf{U}(x_i - x_j)(x_i - x_j)^T$$

To this end, we can optimize the objective function by alternating between Equation 5 and Equation 11 to find a compromise between the modeling and kernel learning.

In each iteration the time complexity of our method is $O(N_{tr}^2 n^2 + N_{tr} n T)$, where N_{tr} is the size of the training set, n is the dimensionality of a sequence, T is the length of sequences. This relatively high computation cost is due to the training of the CRJ network and kernel. Compared with oversampling algorithms directly operating on the signal space, our method pays additional cost on the learning process. However, our method has better performance (Section 4) and is able to deal with multivariate and varying length sequences.

3.4 Oversampling

Denote the learned kernel matrix as $K \in R^{m \times m}$. Eigenvalue decomposition is performed as $K = VDV^T$, where $D = diag([d_1, d_1, \cdots, d_r])$ has r nonzero eigenvalues on the main diagonal, V is the corresponding eigenvector matrix. We project the data into a kernel feature space spanned by eigenvectors: $D^{-\frac{1}{2}}V^T K$ [20] as final representations. The kernel feature space preserves the distance relationships of patterns in the infinite Hilbert space. New synthetic data are obtained in this kernel feature space.

There are two variants of our method:

1) Model-kernel: The neural network is optimized to learn models for sequences and the kernel parameter is omitted. Oversampling is performed in the representation formed kernel feature space [8].

2) Data-kernel: The kernel parameter is optimized. Instead of learning representations for original data, we feed the data directly to the system. Oversampling is performed in the original data formed kernel feature space.

Assume the models and kernel fit the data properly, the data are hypothesized more likely to be linearly separable[1] in

[1]However, being linearly separable is the ideal case, which may not be guaranteed in most datasets.

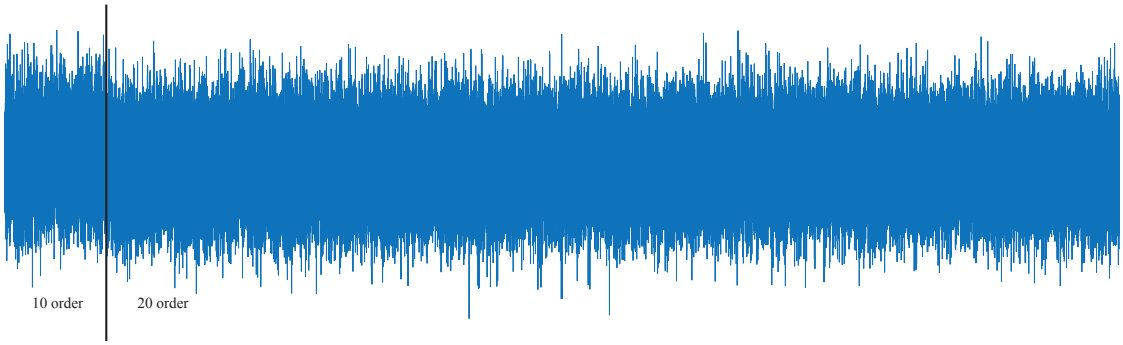

Figure 2: Illustration of the imbalanced NARMA sequences in the signal space.

the kernel feature space. This serves as a necessary condition for interpolation based oversampling.

We postulate by learning representations for sequences as well as making use of the kernel nonlinearity, the generative mechanism and similarity structure are pursued. Therefore, interpolation based oversampling can be performed more effectively in the representation induced kernel feature space. In this case, simple yet efficient oversampling approaches can be employed safely. In this paper, we use SMOTE to generate synthetic data. The whole algorithm is demonstrated in Algorithm 1.

Algorithm 1 Model-based oversampling

1: **Input:** Training set $T = \{X^1, X^2, \cdots, X^{N_{tr}}\}$; #neurons of the reservoir; jump size of reservoir; regression parameter η; trade-off λ; initial parameters for network ($\{r_i, r_c, r_j\}$) and kernel ($\mathbf{U}$).
2: **Output:** A balanced Synthetic dataset.
3: **repeat**
4: Use the CRJ network to fit models for sequences in T.
5: Compute empirical kernel according to Equation 3.
6: Tune network parameters according to Equation 5.
7: Tune kernel parameter according to Equation 11.
8: **until** Maximum number of iterations is reached or the variation of the objective value falls below a threshold.
9: Project the data into kernel feature space as final representations.
10: Perform SMOTE in the kernel feature space to enrich the minority class.

4. EXPERIMENTS

In this Section, we first evaluate the proposed approach on synthetic datasets. Then we compare the proposed approach with state-of-the-art oversampling approaches for imbalanced sequence classification on benchmark datasets. We have tested the proposed method on both univariate and multivariate sequences. Before that, we first detail our experiment settings.

4.1 Experiment Settings

We compare our approach with the majority rule (MjR), SVM without taking account of any synthetic data (SVM), DTW with the nearest neighbor classifier [24], simple replication (Rep) [3], SMOTE [4], ADASYN [12], BorderSMOTE [11] and INOS [13]. DTW computes a distance between two sequences by aligning them over temporal axis [24]. It is widely used in sequence processing. The two variants of our method, i.e. Model-kernel (learning models alone) and Data-kernel (learning the kernel alone) are considered as well. This comparison aims to show that the distance between representations approximates the underlying similarity structure well. Euclidean distance applied directly to sequences may yield suboptimal performance. In all experiments, the number of neighbors for oversampling algorithms are set to be 5 (for AUS dataset it is set 3 because the minority is too rare). The division point of reliable and unreliable subspaces in INOS is determined by 2-fold cross-validation [13].

The kernel we used is a generalized Gaussian kernel with the mahananobis distance metric. The size of reservoir network is fixed to be 100. The jump length of CRJ is set as 5 without further optimization. The network parameters are initialized randomly within the interval [0,1]. The combinational trade-off and ridge regression parameter are tuned by 5-fold cross-validation. We use SVM as the classifier and employ a widely acknowledged implementation Libsvm [25]. The parameters of SVM, such as the slack weight regularization parameter $C \in \{10^{-5}, 10^{-4}, \cdots, 10^5\}$, the kernel width $\gamma \in \{10^{-6}, 10^{-4}, \cdots, 10\}$ are selected by 5-fold cross-validation maximizing g_{means} on the training set. The classifier is trained again on the total training set after the parameters are determined by cross-validation.

We employ four univariate datasets from UCR time series repository [26] — Adiac, SLeaf, FaceAll and wafer[2]. These datasets have already been divided into training and test set. We process the datasets by selecting one class as the minority class, and grouping the rest of the classes as majority class. We also evaluate our method on three multivariate sequence datasets. They are processed to be binary classes as well. The datasets also contain varying length sequences. Existing oversampling approaches cannot handle this kind of sequences, which prevents us from comparing our method with others on these datasets.

For the performance evaluation metric, we use $f_{measure}$, g_{means} and AUC. These evaluation criteria is widely employed in imbalanced learning. Given the confusion matrix:

[2]These datasets are chosen because they have many classes originally, which can be conveniently converted to binary imbalanced datasets.

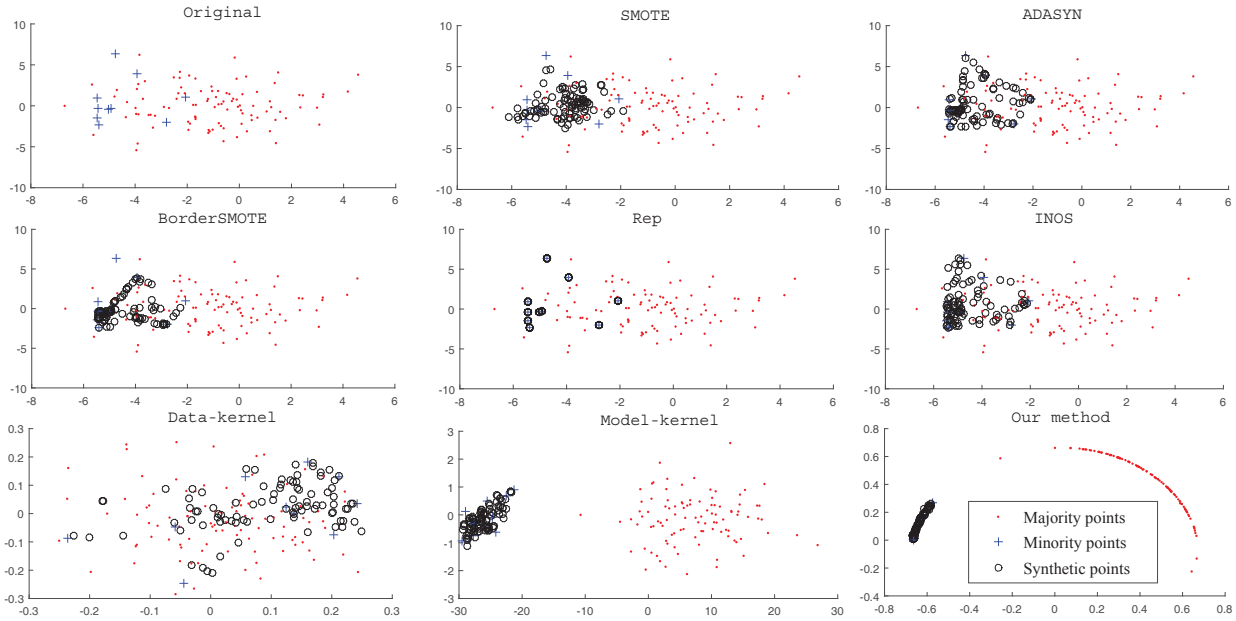

Figure 3: The original synthetic dataset and oversampling results of different algorithms. We first samples new synthetic data and then project both original data and synthetic data into 2D space for visualization. The comparative algorithms are performed in the signal space. The bottom row presents the results of our method. Data-kernel and Model-kernel are two variants of our method. Data-kernel optimizes the empirical kernel by learning the kernel parameter without learning the representations. Model-kernel optimizes the empirical kernel by learning the representations without learning the kernel parameter. Our method is a combination of the two cases. It learns the representations as well as the kernel parameter.

Table 1: Confusion Matrix

	Predictive positive	Predictive negative
True positive	TP	FN
True negative	FP	TN

Then

$$
\begin{aligned}
recall &= TPR = TP/(TP+FN) \\
precision &= TP/(TP+FP) \\
f_{measure} &= 2recall * precision/(recall + precision) \\
TNR &= TN/(TN+FP) \\
g_{means} &= \sqrt{TPR \times TNR}
\end{aligned}
$$

$f_{measure}$ considers recall and precision simultaneously. g_{means} value is determined by the product of correct classified proportion of each class. This value is large only when the classifier performs well on each class. AUC measures the difference between the predictive distribution between classes. The larger the difference, the better the learning algorithm.

4.2 Synthetic Sequence

We generate a series of 10 order and 20 order NARMA sequences:

$$
\begin{aligned}
s(t+1) &= 0.3s(t) + 0.05s(t)\sum_{i=0}^{9} s(t-i) + \\
&\quad 1.5u(t-9)u(t) + 0.1 \\
s(t+1) &= tanh(0.3y(t) + 0.05y(t)\sum_{i=0}^{19} y(t-i) + \\
&\quad 1.5u(t-19)u(t) + 0.01) + 0.2
\end{aligned}
$$

where $s(t)$ is the output sequence, $u(t)$ is the input sequence, $u(t)$ is independently and identically generated in the range [0,0.5] according to uniform distribution. These two kinds of sequences are generated using the same input sequence. The 10 order and 20 order NARMA sequences are treated as the positive and negative class respectively.

We generate 10 order NARMA sequence of 9000 length and 20 order NARMA sequence of 90000 length. These sequences are then partitioned into non-overlapping subsequences. Each subsequence is of 300 length. In this way, the positive class have 30 items and the negative class have 300 items. We randomly select 10 and 100 items from each class as a training set, and the others are used a test set. The imbalance ratio is 10:1 in this scenario. Figure 2 illustrates the synthetic sequences in the original signal space. It is imbalanced and hard to separate the two classes from their signal characteristics using our eyes.

4.2.1 Visualization

Figure 3 demonstrates the generated synthetic dataset

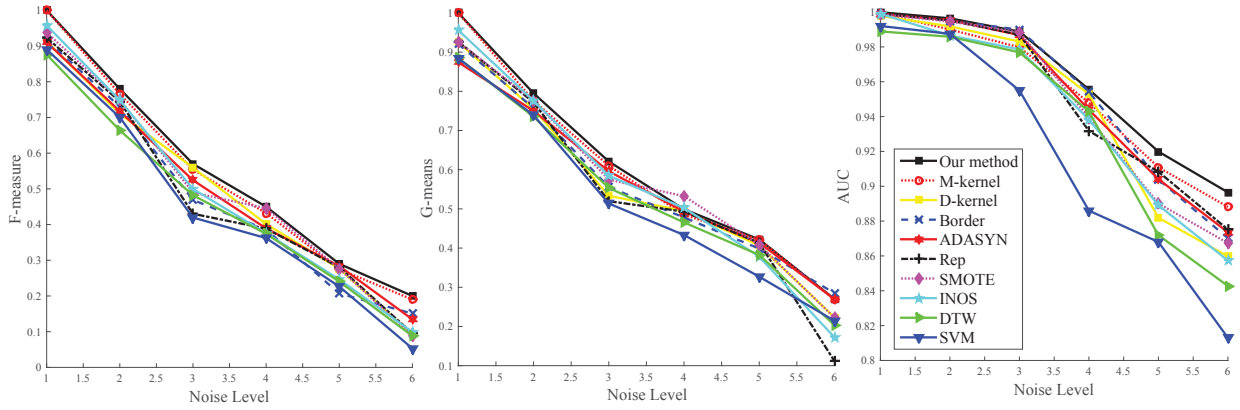

Figure 4: Illustration of the performance evolution of different oversampling algorithms. D-kernel and M-kernel are abbreviations for Data-kernel and Model-kernel.

and sampling results of different algorithms. The sequences and samples are projected to 2 dimensional space for visualization using principle component analysis[3]. It is clear that the positive and negative class severely overlapped in the signal space.

SMOTE uses each minority point with equal probability in generating new synthetic data. It is observed that SMOTE samples evenly in the region of minority class. BorderSMOTE puts more stress on the border points near the majority class. ADASYN assigns soft weights to each minority point and it samples with different probability. The border points are assigned larger probability and points in the inner region of minority class obtains smaller probability. Replication randomly repeat the minority class. It avoids generating samples in the region of the opposite class. However, it may easily cause over-fitting. INOS spreads in the positive region trying to be consistent with the covariance of original minority data. It is clear that oversampling in the signal space using interpolation strategy works poorly in this case, because the minority and majority class overlapped. Learning only the kernel does not perform well. We attribute this result to the fact that directly applying Euclidean distance to high dimensional sequential data may not approximate the underlying similarity structure well. In contrast, our strategy is to transform the original data into the generative model space. The models are hypothesized to be endowed with the ability to preserve the correlated features of sequences along temporal axis. Sampling is performed in the model space. It is observed that learning models alone can yield good performance. The distance between representations approximates the underlying similarity structure well.

4.2.2 *Noise Robustness*

Oversampling techniques have problems when noise or outliers exist in the dataset. To test the robustness of different oversampling algorithms, we also corrupt the synthetic sequences with Gaussian noise of zero mean and standard deviation varying in the interval [0,0.5]. For each noise level, we randomly generated 5 datasets. On each dataset, we repeat each sampling algorithm 10 times. Then the algorithm performance is the average of the 50 runs. Figure 4 illustrates the performance of different oversampling algorithms when facing different levels of noises. It is clear that our method is more robust to noise than other oversampling algorithms. We take the mahananobis distance metric into consideration in the kernel learning. The distance metric encourages the sequences in the same class stay closely while the sequences in different classes stay apart. Therefore, the sequence representations are more probable to be linearly separable, which is a necessary condition for oversampling based on convex combination to be safe.

For the signal space, it is observed that SMOTE in the signal space usually yields quite good performance, despite its simplicity. Replication most time yields the poorest results. However, when facing extreme noisy scenarios, it shows a little performance improvement compared to other algorithms. It is presumed that this observation is the result of the strategy of these algorithms. Because of the large noise in the dataset, interpolation based strategy may capture the noise and result in generating improper synthetic data. These "wrong" samples do not contribute much to the overall performance. By contrast, the samples of replication strategy does not fall into this problem. We also observe that INOS performs well when noise is small. However, when facing large noise, it leads to poor results. It is not surprising for this observation because INOS heavily counts on the covariance of the original data. Therefore, it is more sensitive to noise.

4.3 Univariate Sequence

The dataset information of univariate sequences is summarized in Table 2. The original datasets have multiple classes, e.g. Adiac has 37 classes, SLeaf has 15 and FaceAll has 14. Wafer is imbalanced itself. We select class 2 in Adiac dataset, class 1 in SLeaf and FaceAll datasets respectively as the minority class [13], and group the rest as the negative class. The processed datasets are highly imbalanced.

We present the results on imbalanced univariate sequence classification in Table 3. The reported results are average of 10 runs. Each time the oversampling algorithms are restarted. We perform t-test between our method and the other oversampling algorithms. The significance level is set as 0.05.

From Table 3, we can make four main observations:

1) It is observed that sampling using the generative models

[3]http://www.cad.zju.edu.cn/home/dengcai/Data/code/PCA.m

Table 2: Summarization of univariate sequence datasets.

Dataset	length	Training		Test		Positive class	Imbalance ratio
		#Positive	#Negative	#Positive	#Negative		
Adiac	176	10	350	13	408	2	35:1
SLeaf	128	30	500	45	550	1	16.7:1
FaceAll	131	40	400	72	1738	1	10:1
wafer	152	50	3000	712	3402	1	60:1

Table 3: Classification results on univariate sequence. D-kernel and M-kernel denote Data-kernel and Model-kernel. The best results are boldfaced. The superscript * indicates the difference between the algorithm and our method is statistically significant.

$f_{measure}$											
Dataset	MjR	SVM	DTW	Rep	SMOTE	ADASYN	Border	INOS	D-kernel	M-kernel	Our method
Adiac	-	0.655*	0.727*	0.663*	0.732*	0.733*	0.718*	0.800*	0.800*	0.920*	**0.963**
SLeaf	-	0.485*	0.364*	0.706*	**0.764**	0.712*	0.724	0.731	0.716	0.731	0.762
FaceAll	-	0.824*	0.802*	0.899*	0.903	0.904	0.904	**0.936**	0.906	0.926	0.929
wafer	-	0.866*	0.922*	0.958*	0.965	0.966	0.964	**0.981**	0.969	0.969	0.975
g_{means}											
Dataset	MjR	SVM	DTW	Rep	SMOTE	ADASYN	Border	INOS	D-kernel	M-kernel	Our method
Adiac	0*	0.703*	0.783*	0.730*	0.807*	0.809*	0.794*	0.882*	0.904*	0.907*	**0.999**
SLeaf	0*	0.588*	0.507*	0.800*	0.861	0.849*	0.861	0.853	0.816*	0.853	**0.873**
FaceAll	0*	0.930*	0.933*	0.903*	0.907*	0.913*	0.910*	0.936	0.922	0.933	**0.962**
wafer	0*	0.935*	0.937*	0.958*	0.965*	0.966*	0.964*	**0.985**	0.935*	0.962	**0.985**
AUC											
Dataset	MjR	SVM	DTW	Rep	SMOTE	ADASYN	Border	INOS	D-kernel	M-kernel	Our method
Adiac	0.500*	0.863*	0.806*	0.87*	0.926*	0.933*	0.918*	0.938*	0.943*	0.948*	**1.000**
SLeaf	0.500*	0.894*	0.624*	0.897*	0.890*	0.893*	0.894*	0.904*	0.885*	0.910	**0.945**
FaceAll	0.500*	0.985*	0.983*	0.970*	0.981*	0.986*	0.985*	0.984*	0.981*	0.988	**0.994**
wafer	0.500*	0.984*	0.939*	0.984*	0.965*	0.966*	0.964*	0.985*	0.989*	0.987*	**0.999**

is indeed useful. Our method achieves competitive performance on all 3 evaluation metrics. In detail, INOS achieves 2 wins in $f_{measure}$ and 1 tie in g_{means}. Our method performs slightly inferior in $f_{measure}$ 3 times, but wins in g_{means} and AUC in all datasets. As expected, the memory of dynamical reservoir is helpful given the input sequences have sequential order information. The learned generative model for a sequence encodes the correlation over temporal axis to help preserve the structure information. Therefore, the model space provides more flexible representations than signal space for oversampling. This makes our method superior to state-of-the-art methods.

2) It is obvious that Model-kernel achieves better performance on $f_{measure}$, g_{means} and AUC than Data-kernel on almost all datasets. We attribute this result to the effectiveness of making use of sequential order in learning sequence representations. By doing so, the learned representations may fit the underlying data manifold well, which explains the better performance of Model-kernel.

3) There seems to be no clear winner in performance among the five oversampling algorithms when they sample in the signal space. This is consistent with previous studies. Different dataset characteristics warrant different treatment [27].

4) Based on the experiment results, it is observed that the success of our method is largely due to high recall.

The inferior results of our method with $f_{measure}$ is because the inferior precision performance. This observation indicates that our method can recognizes most of positive examples at the cost of misclassifying some negative examples. It is important in applications where predicting the rare but interesting positive class is given more priority. Besides, our method shows superior performance on AUC. This result means that the predictive distribution of learning in the model-spanned space is more separated than that in the original sequence (signal) space.

4.4 Multivariate Sequence

Our method is evaluated on three different multivariate sequence datasets selected from UCI Machine Learning Repository[4]. The dataset information of multivariate sequences is given in Table 4.

The Libras Movement Dataset (Libras) contains 15 hand movement patterns of Brazilian signal language. Each pattern has 24 instances. The movements are video-recorded and pre-processed. In each video, 45 frames are selected evenly. Then the centroid pixels of the hand are found. The discrete version of the movement curve has 45 points, each of which represents the coordinates of a movement. The label is the hand movement types.

The Character Trajectories Dataset (Hand) comprises 2858 character trajectories written by one person. The trajectories are represented as sequences of 3 variables: 2-dimensional coordinates and pen tip force. The characters are treated as the label.

The Australian Sign Language Dataset (AUS) consists of 95 video-recorded sign language signs from a native sign-

[4]http://archive.ics.uci.edu/ml/datasets.html

Table 4: Summarization of multivariate sequence datasets.

Dataset	Dimensions	#Class	#Sample	Positive class	Imbalance ratio	Length	
						min	max
Libras	2	15	360	1	12.3:1	45	45
Hand	3	20	2858	1	13.6:1	60	182
AUS	22	95	2565	1	99:1	45	136

Table 5: The performance of our method on Libras dataset. D-kernel and M-kernel denote Data-kernel and Model-kernel. The superscript * indicates the difference between the algorithm and our method is statistically significant.

	MjR	SVM	DTW	Rep	SMOTE	ADASYN	Border	INOS	D-kernel	M-kernel	Our method
$f_{measure}$	0*	0.921*	0.919*	0.939*	0.971	0.961	0.971	0.954	0.971	0.974	**0.976**
g_{means}	-	0.938*	0.968*	0.969*	0.971*	0.971*	0.971*	0.961*	0.962*	0.971*	**0.988**
AUC	0.5*	0.952*	0.962*	0.946*	0.960*	0.953*	0.963*	0.961*	0.959*	0.981*	**0.999**

er, 27 samples per sign. The signs are represented by 22 channels of features. Each class is one type of signs.

We have selected the class 1 of all datasets to be the positive class. Note that the Hand and AUS datasets are also of variable length. Existing oversampling methods cannot directly apply to variable length sequences, which prevents the comparison of our method with the other approaches.

Because the sequences in Libras dataset does not have varying length, we have transformed the multivariate sequences into univariate ones by concatenating different channels. In this way, we can perform experiments using other oversampling algorithms on it.

We randomly partition the dataset and use 70% of data for training and 30% for test. Table 5 reports the generalization results of our method and other sampling algorithms on Libras dataset. Table 6 illustrates the classification results of our approach on Hand and AUS dataset. The results are averaged over 10 runs.

It is clear that our method achieves statistically significant better performance on multivariate sequence datasets than comparative methods. Libras dataset contains the discrete position coordinates of the movements of a hand. It is natural to believe that the two dimensions must be interactive when presenting specific patterns. Converting the original multivariate sequences into vectorial data for sampling not only loss the temporal contextual information, but also ignores the interaction of two variables. Our model is based on RNN and can handle multivariate and varying length sequences naturally. Thus it is not surprising to observe better performance of oversampling using the model space than other algorithms that sample in the signal space. On AUS and Hand datasets, we also achieve nearly perfect performance.

To this end, our method performs favorably on both synthetic and benchmark datasets. Two reasons may be involved.

1) We have learned a generative model for each sequence, which encourages sequence modeling and similarity constraints. The learned representations and similarity are presumed to follow the data manifold. Oversampling in the model space avoids the adverse influence of sampling in the signal space.

2) The nonlinearity and separation of kernel learning encapsulates more flexibility. The memory ability of reservoir model also plays a key role, which contributes in maintaining the correlation of sequential data over time.

Table 6: The performance of our method on multivariate sequence datasets. D-kernel and M-kernel denote Data-kernel and Model-kernel. The superscript * indicates the difference between the algorithm and our method is statistically significant.

$f_{measure}$				
Dataset	DTW	D-kernel	M-kernel	Our method
AUS	0.793*	0.791*	0.880*	**0.976**
Hand	0.835*	0.824*	0.915*	**0.996**
g_{means}				
Dataset	DTW	D-kernel	M-kernel	Our method
AUS	0.872*	0.900*	0.922*	**0.976**
Hand	0.877*	0.878*	0.912*	**1.000**
AUC				
Dataset	DTW	D-kernel	M-kernel	Our method
AUS	0.933*	0.944*	0.951	**0.976**
Hand	0.951*	0.981*	0.995	**1.000**

5. CONCLUSION

This paper proposes to oversample imbalanced sequence dataset in the generative model space. For each sequence, a generative model is learned as a representation[5]. The representations of sequences are used to define a kernel matrix. Alignment to desired kernel matrix is optimized by tuning the generalized Gaussian kernel parameter. In doing so, similarity constraints are also pursued. The model and kernel are learned iteratively. The proposed method has advantages in handling highly imbalanced, noisy, varying length and multivariate sequential data. Experiments on synthetic, univariate and multivariate sequences datasets show that our method can achieves competitive performance in terms of $f_{measure}$, g_{means} and AUC.

The time complexity of our method is $O(N_{tr}^2 n^2 + N_{tr} n T)$, where N_{tr} is the size of the training set, n is the dimensionality of a sequence, T is the length of sequences. This tolerable cost is balanced by the better performance and the ability to deal with multivariate and varying length sequences of our method. It is acceptable in applications where good performance is more important. Future work would be to reduce

[5]Strictly speaking, we have used the model as an intermediate representation.

the time complexity of our method and apply our method to more datasets.

6. ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China with grant number 2016YFB1000905, the National Science Foundation of China with grant numbers 91546116, 61511130083, 61503357 and the Fundamental Research Funds for the Central Universities with grant number WK2150110001.

7. REFERENCES

[1] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1356–1368, 2015.

[2] Y. H. Zhou and Z. H. Zhou, "Large margin distribution learning with cost interval and unlabeled data," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, pp. 1–1, 2016.

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

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

[5] Y. Jo, N. Loghmanpour, and C. P. Rosé, "Time series analysis of nursing notes for mortality prediction via a state transition topic model," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1171–1180, ACM, 2015.

[6] K. H. Brodersen, T. M. Schofield, A. P. Leff, C. S. Ong, E. I. Lomakina, J. M. Buhmann, and K. E. Stephan, "Generative embedding for model-based classification of fmri data," *PLoS Comput Biol*, vol. 7, no. 6, p. e1002079, 2011.

[7] J.-S. Wu and Z.-H. Zhou, "Sequence-based prediction of microrna-binding residues in proteins using cost-sensitive laplacian support vector machines," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 3, pp. 752–759, 2013.

[8] H. Chen, P. Tino, A. Rodan, and X. Yao, "Learning in the model space for cognitive fault diagnosis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 124–136, 2014.

[9] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8624–8628, IEEE, 2013.

[10] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun, "Unsupervised learning of spatiotemporally coherent metrics," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4086–4093, 2015.

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

[12] 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*, pp. 1322–1328, IEEE, 2008.

[13] H. Cao, X.-L. Li, D. Y.-K. Woon, and S.-K. Ng, "Integrated oversampling for imbalanced time series classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 12, pp. 2809–2822, 2013.

[14] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *The Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.

[15] C. Cortes, M. Mohri, and A. Rostamizadeh, "Algorithms for learning kernels based on centered alignment," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 795–828, 2012.

[16] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[18] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, p. 34, 2001.

[19] H. Chen, F. Tang, P. Tino, and X. Yao, "Model-based kernel for efficient time series analysis," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 392–400, ACM, 2013.

[20] V. N. Vapnik and V. Vapnik, *Statistical learning theory*, vol. 1. Wiley New York, 1998.

[21] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[22] N. ello Cristianini, A. Elisseeff, J. Shawe-Taylor, and J. Kandola, "On kernel-target alignment," in *Advances in Neural Information Processing Systems*, 2001.

[23] A. Rodan and P. Tiňo, "Simple deterministically constructed cycle reservoirs with regular jumps," *Neural computation*, vol. 24, no. 7, pp. 1822–1852, 2012.

[24] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping.," in *Sdm*, vol. 1, pp. 5–7, SIAM, 2001.

[25] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

[26] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015. www.cs.ucr.edu/˜eamonn/time_series_data/.

[27] G. E. 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, no. 1, pp. 20–29, 2004.