MOGT: OVERSAMPLING WITH A PARSIMONIOUS MIXTURE OF GAUSSIAN TREES MODEL FOR IMBALANCED TIME-SERIES CLASSIFICATION

John Z. F. Pang[‡] *Hong Cao*[⋆] *Vincent Y. F. Tan*[⋆ †]

[‡] School of Physical and Mathematical Sciences, Nanyang Technological University
[⋆] Data Analytics Department, Institute for Infocomm Research (I^{2}R), A*STAR, Singapore
[†] Department of Electrical and Computer Engineering, National University of Singapore

ABSTRACT

We propose a novel framework of using a parsimonious statistical model, known as *mixture of Gaussian trees*, for modelling the possibly *multi-modal* minority class to solve the problem of imbalanced time-series classification. By exploiting the fact that close-by time points are highly correlated, our model significantly reduces the number of covariance parameters to be estimated from $O(d^2)$ to $O(Ld)$, L denotes the number of mixture components and d is the dimension. Thus our model is particularly effective for modelling high-dimensional time-series with limited number of instances in the minority positive class. We conduct extensive classification experiments based on several well-known time-series datasets (both single- and multi-modal) by first randomly generating synthetic instances from our learned mixture model to correct the imbalance. We then compare our results to several state-of-the-art oversampling techniques and the results demonstrate that when our proposed model is used, the same support vector machines classifier achieves much better classification accuracy across the range of datasets. In fact, the proposed method achieves the best average performance 27 times out of 30 multi-modal datasets according to the F-value metric.

***Index Terms*—** Imbalanced dataset, Time-series, Oversampling, Gaussian graphical models, Mixture models, Multi-modality

1. INTRODUCTION

In binary classification of time-series, *class imbalance* is the situation where one class has significantly fewer samples than the other. It is important to solve the imbalance as standard classification methods tend to bias toward the class with the large number of training samples. In many real-world applications, the minority class is often the more informative of the two and the class of significant interest for modelling. For instance, the known failures for aeroplanes are very rare but are of greater importance for predicting the next failure than the abundant number of normal instances. Motivated by the data paucity in the minority class and to model multi-modality, this paper proposes an oversampling method based on a parsimonious *mixture of Gaussian trees model* for imbalanced time-series classification. Such a model is shown to (i) compare excellently with other methods in terms of classification accuracy, (ii) require the estimation of far fewer parameters than existing methods, (iii) model time-series dependence explicitly and (iv) have low algorithmic complexity.

Current techniques for solving class imbalance can be segmented into (i) those at the algorithm-level [1–3] and (ii) those at the data-level [4–14]. Methods in (i) correct the imbalance by assigning a pre-determined weight to each class [1]. This class of methods include uneven dataspace weighting [1, 2, 11] or modifying the operating point on the receiver operating characteristics curve [3]. Though some methods achieves excellent performance with theoretical bounds [15], the pre-determination of the weight of each class is required, and this is not always trivial. Methods in (ii) correct the imbalance by undersampling the majority class [10] or by oversampling from the minority class [4–9, 11, 12], or a combination of both [6]. Undersampling is efficient because of the reduced size of the resulting dataset but it suffers from the risk of discarding important samples. Oversampling has the advantage of retaining all existing training samples at the expense of an enlarged learning dataset. It also enhances representativeness of the minority class by introducing rich data variations. Oversampling is the approach we adopt in this paper as it has been found to be effective for re-establishing the class balance for conventional imbalanced classification problems. To deal with the curse-of-dimensionality involved in modelling high-dimensional time-series data, we adopt *sparse graphical models* [16] to model dependence across time of the positively-labelled time-series data statistically. This has also been done previously [17].

We now summarize our main contributions. First, by using mixtures of sparse (or parsimonious) Gaussian graphical models [18,19], we reduce the number of parameters of the model compared to structure preserving oversampling (SPO) [4] by exploiting the correlations of nearby time points in a time-series instance. This ameliorates overfitting and circumvents the need for regularization of the learned model. Second, there is only one parameter to tune, the number of mixture components L, which we typically set to one or two in this study. Thirdly, the model is well-adapted to the problem at hand because time-series data is highly correlated and our model explicitly takes this into account. Fourth, the model is able to deal with *multi-modal* time-series data since we use a *mixture* of Gaussian trees, each of which has one mode. This addresses the issue of large within-class variations of time-series data. See Fig. 1 for an illustration.

2. PRELIMINARIES

2.1. Notation

Upper case (e.g., X) and lower case (e.g., x) letters denote random variables and their realizations respectively. Sets are denoted by calligraphic font (e.g., $\mathcal{X}$) and the cardinality of finite sets by $|\cdot|$. The probability distribution of X denoted as $p_X(x)$ or simply as $p(x)$ if X is clear from the context. A random vector is a finite collection of random variables and is denoted by bold-face upper case as $\mathbf{X} = (X_1, \ldots, X_d)$, where d is the dimension of the random vector in context. A realization of this random vector, a deterministic vector in $\mathbb{R}^d$, is denoted as $\mathbf{x} = (x_1, \ldots, x_d)$. Given a subset of indices $\mathcal{S} \subset \{1, \ldots, d\}$, $\mathbf{X}_\mathcal{S} = \{X_i : i \in \mathcal{S}\}$ denotes the set of random variables indexed by $\mathcal{S}$. Correspondingly, $\mathbf{x}_\mathcal{S} = \{x_i : i \in \mathcal{S}\}$. Matrices are also denoted using bold-face upper case (e.g., $\mathbf{A}$).

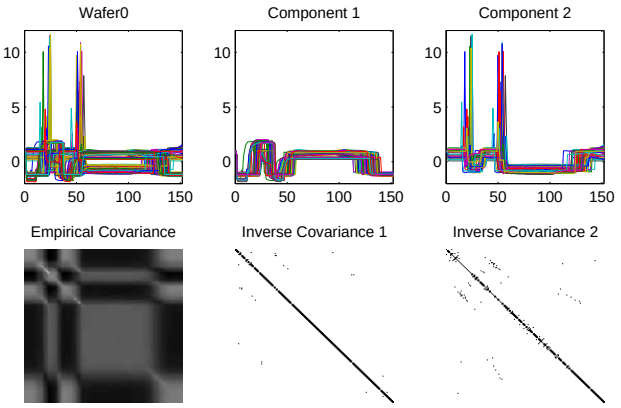

Fig. 1. The Wafer0 dataset and the two constituent components separated by our algorithm are shown on the top row. The empirical covariance matrix have no sparsity structure but the learned inverse covariance matrices do.

Standard notations for information-theoretic quantities are used in this paper [20]. For example, *mutual information* between random variables X and Y is denoted as $I(X;Y)$ and the *Kullback-Leibler (KL) divergence* between distributions p and q is denoted as $D(p\,||\,q)$. If random variables X and Y are *conditionally independent* given another random variable Z, we denote this by the Markov chain $X - Z - Y$. The *correlation coefficient* between two random variables X and Y, a number between -1 and 1, is defined as $\rho(X,Y) = \mathsf{Cov}(X,Y)/\sqrt{\mathsf{Var}(X)\,\mathsf{Var}(Y)}$.

The multivariate Gaussian probability density function with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{S}_+^d$ ($\mathbb{S}_+^d$ is the cone of positive semi-definite $d \times d$ matrices) is denoted as

$$\mathsf{N}(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{2\pi\det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right). \tag{1}$$

2.2. The Binary Classification Problem

We consider a binary classification problem where the training data is divided into two disjoint subsets $\mathcal{D}_+ := \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n_+)}\}$ and $\mathcal{D}_- := \{\bar{\mathbf{x}}^{(1)}, \ldots, \bar{\mathbf{x}}^{(n_-)}\}$, a small set of positively-labelled samples and a large set of negatively-labelled samples. The union $\mathcal{D}_+ \cup \mathcal{D}_-$ is the set of *training samples*. We would like to use $\mathcal{D}_+$ and $\mathcal{D}_-$ to determine the class label of a set of samples that are unlabelled. To do so, we propose a statistical model for the samples in $\mathcal{D}_+$ and then augment to $\mathcal{D}_+$ by sampling from the learned distribution so that the resulting cardinality of both classes is the same. Finally, we perform classification on the augmented dataset.

2.3. Graphical Models

A *graphical model* [16, 21] is a probability distribution for which a graph encodes the conditional independence relations among a collection of random variables $\mathbf{X} = (X_1, \ldots, X_d)$. We associate vertex $i \in \mathcal{V}$ to random variable X_i, which takes on values in some alphabet $\mathcal{X}_i$. In the sequel, we use the terms *variable*, *vertex* and *node* interchangeably. We say that random vector $\mathbf{X}$ is *Markov on* graph $\mathcal{G}$ if conditioned on the set of neighbors of i, X_i is independent of all the other random variables in the graph.

By the Hammersley-Clifford theorem [21], if the random vector $\mathbf{X}$ is Markov on the undirected graph $\mathcal{G}$, then the joint distribution $p_{\mathbf{X}}(\mathbf{x})$ can be factorized according to the maximal cliques $\mathscr{C}$ of the graph $\mathcal{G}$, i.e.,

$$p_{\mathbf{X}}(\mathbf{x}) = \prod_{\mathcal{C}\in\mathscr{C}} p_{\mathbf{X}_\mathcal{C}}(\mathbf{x}_\mathcal{C}), \tag{2}$$

Equation (2) shows that the number of parameters describing the joint distribution is reduced since they only depend on the joint distributions on the maximal cliques now. If $\mathbf{X}$ is a Markov chain, the factorization in (2) specializes in this case to

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)\ldots p(x_d|x_{d-1}). \tag{3}$$

The number of parameters given the factorization in (3) is *linear* in d. Thus, overfitting can be ameliorated if the high-dimensonal data can be modelled adequately by a sparse graphical model.

Assuming that $\mathbf{X}$ is jointly Gaussian, then the *inverse covariance matrix* $\mathbf{J} := \boldsymbol{\Sigma}^{-1}$ has a sparsity structure corresponding to that of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Such a probabilistic model is known as a *Gaussian graphical model*. Then $J_{i,j} \neq 0$ if and only if $(i,j) \in \mathcal{E}$.

Tree-structured graphical models [22] (or trees) admit efficient and exact inference [23] and learning [24] algorithms. We choose to model the data using trees because the learning is conceptually easier and the set of tree models is more flexible and does not incur any penalty in terms of the number of parameters as we shall see.

A *tree-structured graphical model* is one that is Markov on a tree. The joint distribution of a random vector Markov on a tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ admits the factorization as the maximal cliques are precisely the edges of the graph:

$$p(\mathbf{x}) = \prod_{i\in\mathcal{V}} p(x_i) \prod_{(i,j)\in\mathcal{E}} \frac{p(x_i,x_j)}{p(x_i)p(x_j)}. \tag{4}$$

Thus, all we need to describe the joint distribution of a tree model are the pairwise statistics on the edges, i.e., $p(x_i, x_j)$ for all $(i,j) \in \mathcal{E}$. It can be easily verified [19] that if the Gaussian tree model is normalized such that the variances of X_i are unity for every $i \in \mathcal{V}$, then all elements of the covariance matrix $\boldsymbol{\Sigma}$ are functions of the set of correlation coefficients on the edges of the graph, i.e., $\{\rho_{i,j} : (i,j) \in \mathcal{E}\}$. Indeed, for general models with variances $\sigma_i^2 := \Sigma_{i,i}$, by exploiting the conditional independences between the random variables $(X_1, \ldots, X_d)$ which are Markov on the tree $\mathcal{T}$, it can easily be shown [19] that

$$\Sigma_{i,j} = \begin{cases} \rho_{i,j}\sigma_i\sigma_j & (i,j) \in \mathcal{E} \\ \left(\prod_{(k,l)\in\mathrm{Path}(i,j)} \rho_{k,l}\right)\sigma_i\sigma_j & (i,j) \notin \mathcal{E} \end{cases}. \tag{5}$$

where $\mathrm{Path}(i,j) \subset \mathcal{E}$ is the set of edges joining i and j. In general, the sets $\{\sigma_i : i \in \mathcal{V}\}$ and $\{\rho_{i,j} : (i,j) \in \mathcal{E}\}$ completely characterize the covariance matrix of $\mathbf{X}$. The construction of $\boldsymbol{\Sigma}$ in (5) results in a positive definite matrix if $|\rho_{i,j}| \in (0,1)$ for all $(i,j) \in \mathcal{E}$ [19]. Furthermore $\mathbf{J} := \boldsymbol{\Sigma}^{-1}$ is sparse according to $\mathcal{E}$.

Since there are only $d-1$ edges in a tree model, $2d-1$ parameters ($d-1$ correlation coefficients and d variances) suffice to describe the covariance matrix. Including the d elements in the mean of $\mathbf{X}$, $3d-1$ parameters suffice to describe the entire distribution. This is in contrast to an unstructured model which would require $d(d+3)/2$ parameters to describe.

2.4. Learning Tree-Structured Graphical Models

Chow and Liu [24] used the factorization in (4) to derive an algorithm to learn trees from data, assuming that the variables are discrete. Here we describe an extension of their algorithm to

continuous-valued data.

The setup is as follows: Given that samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)} \in \mathbb{R}^d$ are i.i.d. from a tree-structured graphical model $p_{\mathbf{X}}(\mathbf{x})$, we aim to reconstruct the set of edges of the underlying graph of p. To do so, consider the following optimization:

$$\min_{q \in \mathscr{T}_d} D(\hat{p} \,||\, q) \tag{6}$$

where $\mathscr{T}_d$ denotes the family of tree models and $\hat{p}$ is an estimate of the distribution from $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$. Chow and Liu showed that (6) reduces to the following max-weight spanning tree problem [25]:

$$\hat{\mathcal{E}} := \arg\max_{\mathcal{E} \in \mathfrak{T}_d} \sum_{(i,j) \in \mathcal{E}} \hat{I}(X_i; X_j), \tag{7}$$

where the maximization in (7) is over the set of (spanning) trees $\mathfrak{T}_d$ and $\hat{I}(X_i; X_j)$ is the empirical mutual information between X_i and X_j computed with respect to $\hat{p}$, which in the jointly Gaussian case, can be expressed in closed-form as [20]

$$\hat{I}(X_i; X_j) = -\frac{1}{2} \log(1 - \hat{\rho}_{i,j}^2), \tag{8}$$

where $\hat{\rho}_{i,j}$ is an estimate of the correlation coefficient between random variables X_i and X_j. This quantity can be computed via a maximum likelihood(ML) procedure. Since $\hat{I}(X_i; X_j)$ in (8) is monotonically increasing in $|\hat{\rho}_{i,j}|$, we can search for the edge set with the highest sum of $|\hat{\rho}_{i,j}|$ instead of the mutual informations in (7). In this way, we circumvent the need to perform the computation in (8).

3. THE MIXTURE OF GAUSSIAN TREES MODEL FOR THE POSITIVE CLASS

In this section, we describe the mixture of trees model [18] and the Expectation Maximization-based algorithm used to learn Gaussian Tree models from data. However, because our data are *continuous-valued*, it is necessary to modify the algorithm to handle such non-categorical data. Thus, we suggest the *Gaussian* mixture of trees model and also a learning algorithm for such models by combining the EM-framework in [18] and the learning algorithm for Gaussian tree models [19] described in Section 2.4.

3.1. Mixture of Gaussian Trees

Even though trees and their variants have been employed successfully in a variety of machine learning tasks such as classification [26], it suffers from the fact that in the Gaussian case, the Gaussian tree model can only model data that is unimodal. Thus a natural generalization to effectively model data with more than one mode (or multi-modal data) is to consider a convex combination of two or more Gaussian trees. Such a model is characterized by a set of non-negative *mixing coefficients* $\{\alpha_l : 1 \leq l \leq L\}$ which sum up to one (i.e., $\sum_{l=1}^{L} \alpha_l = 1$) and a collection of *constituent Gaussian tree models* $\{p_l(\mathbf{x}) = \mathsf{N}(\mathbf{x}; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l) : 1 \leq l \leq L\}$. Since each p_l is a Gaussian tree model, $\mathbf{J}_l := \boldsymbol{\Sigma}_l^{-1}$ has a sparsity pattern that corresponds to a tree and, in particular, $\mathbf{J}_l$ only has $O(d)$ non-zero elements.

The probability density function of the random vector $\mathbf{X}$, following a mixture of Gaussian trees model $\mathcal{M}_L$, satisfies

$$\mathcal{M}_L : \quad p^{(L)}(\mathbf{x}; \boldsymbol{\theta}) = \sum_{l=1}^{L} \alpha_l p_l(\mathbf{x}), \tag{9}$$

where the set of parameters is denoted as $\boldsymbol{\theta} := \{\alpha_l, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l : 1 \leq l \leq L\}$. Because each mixture component p_l is a Gaussian tree model, the convex combination $p^{(L)}$ can have up to L modes. Since the positive class, which we are modelling using mixture of Gaussian trees, has few samples, we typically set $L = 2$ to avoid overfitting. Note that the number of parameters required to describe the model is still very low. Indeed, only $3dL - 1$ parameters suffice. This number is *linear* in d and hence the model is parsimonious. We are thus able to ameliorate overfitting and represent multi-modal data well.

3.2. Learning Mixture of Gaussian Trees

A central question in the study of mixture models is the ML learning of the model parameters from data. As in Section 2.4, we assume that we are given samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)} \subset \mathbb{R}^d$, each sampled i.i.d. from a probability distribution $p(\mathbf{x})$ that is either an exact mixture of Gaussian trees or can be well-modeled by such a mixture. We would like to learn the parameters of the model $\boldsymbol{\theta}$ assuming that L is given, which will then be used for oversampling. The usual way of learning mixture models is to regard cluster assignments as latent variables and employ an EM-procedure [27] that alternates between an Expectation step (E-step) and a Maximization step (M-step). A canonical example is in Gaussian mixture modelling [16, Section 9.2.2]. However, since we want to take the time-series correlations into account we have to perform further projections onto the set of Gaussian trees using the technique described in Section 2.4.

First, we initialize the parameters $\alpha_l, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l$ for $1 \leq l \leq L$ to feasible values. We can use the k-means algorithm to obtain good estimates of $\boldsymbol{\mu}_l$. Finally, $\boldsymbol{\Sigma}_l \in \mathbb{S}_+^d$ can be set to any positive definite covariance matrix via $\boldsymbol{\Sigma} = \mathbf{A}\mathbf{A}^\top$ for a random matrix $\mathbf{A}$. Based on these initial parameters, compute the log-likelihood of the data.

Second, for the E-step, compute the *responsibilities* [16] as

$$\gamma_{kl} := \frac{\alpha_l \mathsf{N}(\mathbf{x}^{(k)}; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}{\sum_{j=1}^{L} \alpha_j \mathsf{N}(\mathbf{x}^{(k)}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{10}$$

where $k \in \{1, \ldots, n\}$ indexes samples and $l \in \{1, \ldots, L\}$ indexes mixture components. Roughly speaking, γ_{kl} represents the probability that sample $\mathbf{x}^{(k)}$ belongs to mixture component l and γ_{kl} are the latent variables in the EM procedure. Note that $\sum_{l=1}^{L} \gamma_{kl} = 1$.

Third, for the M-step, we set $n_l := \sum_{k=1}^{n} \gamma_{kl}$ to be the *effective number of samples in mixture component* l. Note that $\sum_{l=1}^{L} n_l = n$. We then re-estimate the *intermediate parameters* consisting of *mixing coefficients* and the *means*,

$$\alpha_l^{\text{new}} = \frac{n_l}{n}, \qquad \boldsymbol{\mu}_l^{\text{new}} = \frac{1}{n_l} \sum_{k=1}^{n_l} \gamma_{kl} \mathbf{x}^{(k)}, \tag{11}$$

and the *intermediate covariances*

$$\tilde{\boldsymbol{\Sigma}}_l^{\text{new}} = \frac{1}{n_l} \sum_{k=1}^{n_l} \gamma_{kl} (\mathbf{x}^{(k)} - \boldsymbol{\mu}_l)(\mathbf{x}^{(k)} - \boldsymbol{\mu}_l)^\top, \tag{12}$$

for each $l \in \{1, \ldots, L\}$. However, note from (12) that the Gaussians $\tilde{p}_l(\mathbf{x}) := \mathsf{N}(\mathbf{x}; \boldsymbol{\mu}_l^{\text{new}}, \tilde{\boldsymbol{\Sigma}}_l^{\text{new}})$ are not necessarily tree-structured (because $(\tilde{\boldsymbol{\Sigma}}_l^{\text{new}})^{-1}$ is, in general, not sparse with respect to a tree). Hence, to spasify the model, we adopt the tree-learning method described in section 2.4 on each of the covariance matrices to obtain a model with only $O(d)$ parameters. We depart from the traditional procedure used to learn the parameters of a Gaussian mixture model and project $\tilde{\boldsymbol{\Sigma}}_l^{\text{new}}$ onto the set of trees by (6), resulting in the l-th

edge set being updated as follows:

$$\hat{\mathcal{E}}_l^{\text{new}} := \arg\max_{\mathcal{E}\in\mathfrak{T}_d} \sum_{(i,j)\in\mathcal{E}} \hat{I}_{\tilde{p}_l}(X_i; X_j), \tag{13}$$

where $\hat{I}_{\tilde{p}_l}(X_i; X_j)$ is the empirical mutual information between X_i and X_j computed with respect to the Gaussian $\tilde{p}_l$ which has mean and covariance given by (11) and (12) respectively. The empirical mutual information $\hat{I}_{\tilde{p}_l}(X_i; X_j)$ is given by the formula in (8) where the correlation coefficient is computed using $\hat{\tilde{\mathbf{\Sigma}}}_l^{\text{new}}$ in place of $\hat{\mathbf{\Sigma}}$. The updated covariance matrix $\mathbf{\Sigma}_l^{\text{new}}$ is then given by (5) with edge set $\hat{\mathcal{E}}_l^{\text{new}}$. In addition, the correlation coefficients along the edges and the variances are computed based on the intermediate covariance $\tilde{\mathbf{\Sigma}}_l^{\text{new}}$. To find the path between two nodes efficiently, we employ Dijkstra's shortest path algorithm. $\mathbf{\Sigma}_l^{\text{new}}$ is guaranteed to have an inverse whose sparsity structure corresponds to the tree with edge set $\hat{\mathcal{E}}_l^{\text{new}}$. After the M-step, we recompute the log-likelihood of the n samples under the new parameters $\{\alpha_l^{\text{new}}, \boldsymbol{\mu}_l^{\text{new}}, \mathbf{\Sigma}_l^{\text{new}} : 1 \leq l \leq L\}$.

We iterate and recompute the responsibilities in the E-step in (10) until the difference between the log-likelihoods of the samples between two iterations is below a pre-defined small threshold. In practice, for our datasets, the number of iterations taken for the algorithm to converge is small (< 10) because we provide good initializations using k-means.

4. THE OVERSAMPLING AND CLASSIFICATION FRAMEWORK USING A MIXTURE OF GAUSSIAN TREES

4.1. Details of the Oversampling and Classification Procedure

First, we use the procedure described in Section 3.2 to learn a mixture of Gaussian trees from $\mathcal{D}_+$. For simplicity, we usually set $L = 2$ so that we can model multi-modal data from a limited number of samples and the number of parameters is also small.

Second, we generate $n_- - n_+$ samples from $p^{(L)}$. Suppose that $L = 2$, then we sample from a Bernoulli random variable $B \in \{1, 2\}$ with bias α_1. If $B = 1$ (resp. $B = 2$), then we sample a vector randomly from $\mathsf{N}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1)$ (resp. from $\mathsf{N}(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$). For sampling vectors from a multivariate Gaussian $\mathsf{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, one usually performs the Cholesky decomposition of the covariance matrix $\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^\top$, where $\mathbf{L}$ is lower triangular. One then samples a vector $\mathbf{z}$ from $\mathsf{N}(\mathbf{0}, \mathbf{I})$ and performs a linear transformation $\mathbf{L}\mathbf{z} + \boldsymbol{\mu}$ to obtain the desired sample. Since each component is a tree, we can implement this step more judiciously by sampling from any node in the tree (the root) and direct all undirected edges away from the root. The complexity of such a procedure exploiting the learned tree structure is linear in d.

Having augmented to the samples in $\mathcal{D}_+$ to form a new dataset $\mathcal{D}'_+$, we can then use a standard Support Vector Machines (SVM) [28] on $\mathcal{D}_-$ and $\mathcal{D}'_+$ to classify the unlabelled samples.

4.2. Advantages of the Proposed Method

In this section, we highlight competencies of the proposed method.

First, the number of parameters required to describe the parsimonious model is small (of order $O(Ld)$). This is in contrast to other oversampling methods, such as SPO [4] which is of order $O(d^2)$. As there are fewer parameters, it is less likely that overfitting occurs (Occam's razor [16]).

Second, the mixture of Gaussian trees model has no parameters to tune once L is fixed. Other methods such as SPO [4] require an additional post-processing step on the eigenspectrum of the learned covariance matrix $\mathbf{\Sigma}$ to truncate small eigenvalues of $\mathbf{\Sigma}$, requiring cross-validation which is computationally expensive. As such, our model alleviates this problem due to its parsimony.

Third, the model is well-adapted to time-series data since each component is Markov on a tree [17]. Time-series data are characterized by the fact that each sample in time is highly correlated to its neighbors and the correlation decays as the distance between the two nodes increases. By learning a tree, we explicitly incorporate this important feature of the dataset into our model.

Finally, we are able to handle multi-modal data because the model is a *mixture* of trees.

5. NUMERICAL RESULTS

We constructed 48 imbalanced two-class datasets from 8 multi-class datasets from the UCR time-series repository [29]. These datasets are selected as they contain large numbers of samples to simulate the highly skewed binary classification task. We systematically converted the datasets into multiple imbalanced two-class datasets by selecting one or merging several classes to form the positive class and using the remaining classes to form the negative class. The indices of the selected positive classes are appended to the name of the dataset as in Table 1. Note that we have distinguished two types of imbalanced datasets, which we call "multi-modal" and "unimodal". "Multi-modal" here refers to the case that *multiple* original classes are selected to form the new positive class, while "unimodal" means that only *one* is selected. By merging original classes, we simulate the situation where the positive class is multi-modal, and it is this situation that our proposed algorithm is expected to perform well since it takes the multi-modality into account explicitly. The training and test data are divided randomly and all available samples are included in either sets with *imbalance ratios* between 1.9 and 15.

The figures of merit that we use to assess the accuracy of the algorithms are the *F-value* and *G-mean* [15, 30]. These are simply functions of the precision and recall. Due to space constraints, we only tabulate the F-values. We observed that the behavior of the G-means are similar to the F-Values. All the imbalanced datasets and tables for the G-means can be found at the following permanent link: `https://sites.google.com/site/mlsp13mogt/`.

The comparisons of our mixture of Gaussian trees method for oversampling to other oversampling methods are detailed in Tables 1, and 2. During the SVM-based classification with radial basis function kernel, we perform log-scale grid search and choose the best combination of SVM parameters (C, g) for each oversampling method and each dataset through cross-validation. Our methods are termed 1MoGT and 2MoGT where numeral denotes the number of mixture components. In the 1MoGT case, we are simply modeling the positively-labeled data using one Gaussian tree model. We ran all experiments over 10 independent runs and computed the means and standard deviations of the F-values and G-means. This is to confirm that the variability across different oversampled datasets is not large and our results are consistent.

For the multi-modal datasets, we observe from Table 1 that 2MoGT outperformed all the remaining oversampling methods in good consistency. Here, 2MoGT achieves the best average performance 27 times (according to the F-value). In particular, we found for some difficult-to-classify datasets (e.g., TwoPattern4_1, SLeaf1_2, 50words3_4 and Adiac7_8_9_10) that 2MoGT achieves large margins of improvement over all other oversampling methods. The good consistency and large improvement margin of 2MoGT suggest that it is important to model the distribution of such datasets with more than one mixture component using 2MoGT.

Methods	None	SPO [4]	Repeat	SMOTE [5]	BorSMOTE [8]	ADASYN [9]	DB [11]	1MoGT	2MoGT
FaceAll2_3	.955±.000	.960±.004	.955±.000	.952±.002	.951±.003	.952±.003	.954±.003	**.965±.002**	.963±.005
FaceAll3_4	.912±.000	.940±.003	.912±.000	.904±.004	.910±.004	.903±.005	.946±.004	.969±.004	**.970±.003**
FaceAll4_5	.973±.000	.964±.006	.941±.000	.943±.004	.944±.002	.945±.003	.964±.003	.960±.005	**.980±.005**
FaceAll5_8	.952±.000	.950±.002	.952±.000	.951±.003	.951±.002	.952±.002	.947±.002	.962±.001	**.963±.003**
FaceAll8_2	.677±.000	.812±.002	.940±.000	.941±.002	.941±.002	.939±.002	.760±.001	.891±.002	**.951±.002**
TwoPatterns1_2	.263±.000	.914±.004	.746±.000	.769±.002	.769±.003	.769±.006	.605±.003	.854±.007	**.961±.002**
TwoPatterns2_3	.163±.000	.742±.003	.163±.000	.325±.005	.310±.004	.334±.003	.328±.002	.672±.006	**.798±.005**
TwoPatterns3_4	.768±.000	.911±.003	.856±.000	.860±.001	.860±.002	.860±.002	.812±.001	.904±.004	**.922±.002**
TwoPatterns4_1	.456±.000	.634±.008	.087±.000	.291±.003	.286±.003	.297±.002	.459±.002	.554±.005	**.832±.005**
SLeaf1_2	.818±.000	.804±.013	.768±.000	.787±.004	.775±.011	.785±.006	.825±.009	.728±.011	**.892±.011**
SLeaf2_3	.816±.000	.796±.014	.800±.000	.803±.010	.801±.007	.806±.009	.818±.007	.761±.008	**.841±.011**
SLeaf3_4	.707±.000	.826±.014	.588±.000	.659±.010	.624±.012	.656±.011	.808±.010	.823±.016	**.841±.014**
SLeaf4_5	.839±.000	.838±.014	.630±.000	.682±.000	.677±.014	.686±.010	.837±.010	.828±.011	**.896±.014**
SLeaf5_1	.845±.000	.835±.011	.845±.000	.840±.006	.850±.008	.839±.007	.821±.008	.840±.016	**.892±.013**
50words1_2	.935±.000	.914±.005	.913±.000	.915±.002	.916±.002	.914±.005	.921±.004	.925±.002	**.977±.002**
50words2_3	.870±.000	.854±.012	.865±.000	.842±.008	.847±.010	.839±.008	.861±.000	.871±.002	**.934±.003**
50words3_4	.680±.000	.668±.011	.648±.000	.670±.006	.657±.018	.670±.008	.659±.002	.650±.008	**.903±.008**
50words4_5	.711±.000	.677±.020	.756±.000	.739±.009	.737±.008	.736±.008	.697±.003	.700±.006	**.814±.008**
50words5_1	.867±.000	.862±.014	.867±.000	.864±.006	.843±.000	.860±.004	.875±.000	.867±.000	**.883±.007**
SynCtrl1_2	1.00±.000	1.00±.000	1.00±.000	1.00±.000	1.00±.000	1.00±.000	1.00±.000	1.00±.000	1.00±.000
SynCtrl2_3	.995±.000	.992±.004	.995±.000	.996±.002	.995±.000	**.998±.003**	.985±.004	.996±.002	.996±.003
SynCtrl3_4	.980±.000	.980±.004	.980±.000	.980±.000	.978±.002	.979±.002	.987±.003	.982±.003	**.988±.003**
SynCtrl4_5	.964±.000	.969±.006	.964±.000	.964±.000	.964±.000	.964±.000	.963±.009	.968±.006	**.978±.006**
SynCtrl5_1	**.990±.000**	.985±.000	.975±.000	.977±.003	.976±.002	.977±.002	.985±.000	.985±.002	**.990±.000**
Adiac1_2_3_4	.709±.000	.723±.007	.739±.000	.785±.011	**.789±.007**	.785±.005	.686±.009	.685±.005	.781±.009
Adiac4_5_6_7	.448±.000	.522±.027	.520±.000	.514±.017	.491±.006	.507±.018	.514±.016	.566±.021	**.578±.016**
Adiac7_8_9_10	.658±.000	.584±.023	.600±.000	.699±.018	.700±.016	.698±.021	.621±.000	.568±.021	**.790±.021**
Adiac10_11_12_13	.819±.000	.872±.017	.760±.000	.779±.013	.775±.121	.780±.120	.753±.002	.843±.008	**.877±.007**
Adiac13_14_15_16	.829±.000	.835±.011	.790±.000	.796±.017	.836±.004	.797±.018	.683±.007	.794±.022	**.910±.015**
Adiac16_17_18_19	.769±.000	.775±.019	.667±.000	.698±.022	.698±.016	.716±.023	.746±.008	.772±.029	**.857±.008**

Table 1. Comparison of F-values for various oversampling methods on multi-modal positive class

We have also benchmarked 2MoGT with a recently-developed integrated oversampling (INOS) method [30], which enhances and integrates SPO with interpolation-based oversampling techniques. Using the suggested integration ratio of 70%, we find for the multi-modal dataset, INOS achieves better performance than SPO, but is not comparable with 2MoGT. INOS achieves average F-values of 92.5%, 85.6%, 81.6%, 78.2%, 98.6% and 71.5% for FaceAll, TwoPatterns, SLeaf, 50words, SynCtrl and Adiac, respectively. Correspondingly, 2MoGT scores 96.5%, 87.8%, 87.2%, 90.2%, 99.0% and 79.9% with clear winning margins for four of the six datasets.

For the unimodal datasets, we observe from Table 2 that 1MoGT and 2MoGT perform competitively across all 18 imbalanced time-series datasets compared to the other oversampling methods. Since the positive classes of these datasets are unimodal, we notice that SPO performs the best on many occasions since it fits data better. Moreover, SPO also incorporates regularization and cleaning mechanisms to boost its performance. Without such additional measures, our 1MoGT performed slightly worse than SPO, but it requires significantly fewer parameters for modeling time-series. Other than SPO, our 1MoGT and 2MoGT perform the best by achieving the best F-value for 7 datasets and the best G-mean for 8 datasets out of a total of 18 datasets. We also noticed when 1MoGT outperforms 2MoGT, the differences in performance are not very pronounced because when we set $L = 2$ for a strongly unimodal positive class, then either the inferred misture components α_1 or α_2 will be small (close to zero), thus that component is automatically de-emphasized.

6. CONCLUSION

We proposed oversampling from a parsimonious mixture of Gaussian trees to correct the imbalance in time-series classification. Our model is able to capture the dependencies among neighboring time points and learning is computationally efficient. The number of parameters that are required to be stored is linear in the dimensionality of the data. We validated the proposed methods on 18 unimodal and 30 multimodal imbalanced datasets and observed that combining trees with the multi-modal aspect results in excellent classification performance.

In the future, especially for oversampling minority classes which are multi-modal in nature, we will compare the results obtained using our modeling technique to non-oversampling methods for time-series classification such as 1NN-DTW [31].

References

[1] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–78, Dec 2007.

[2] H. Cao and A. C. Kot, "Manipulation detection on image patches using FusionBoost," *IEEE Trans. on Info. Forensics and Security*, vol. 7, no. 3, pp. 992–1002, Jan 2012.

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

[4] H. Cao, X.-L. Li, Y.-K. Woon, and S.-K. Ng, "SPO: Structure preserving oversampling for imbalanced time series classification," in *Proc. IEEE Int. Conf. on Data Mining*, 2011, pp. 1008–13.

[5] 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, pp. 321–57, 2002.

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

[7] G. Batista, R. C. Prati, and M. C. Monard, "A study of the

Methods	None	SPO [4]	Repeat	SMOTE [5]	BorSMOTE [8]	ADASYN [9]	DB [11]	1MoGT	2MoGT
FaceAll1	.968±.000	.964±.009	.948±.000	.953±.006	.951±.005	.958±.005	.959±.008	.963±.005	**.975±.005**
FaceAll2	.922±.000	**.938±.010**	.831±.000	.883±.005	.881±.007	.888±.008	.922±.004	.908±.005	.868±.008
FaceAll3	.923±.000	.955±.006	**.978±.000**	.967±.006	.976±.002	.967±.004	.919±.004	.959±.006	.948±.006
FaceAll4	.944±.000	.940±.005	**.963±.000**	.956±.006	.962±.005	.961±.005	.908±.011	.950±.005	.950±.005
FaceAll5	.975±.000	.965±.004	**.979±.000**	**.979±.001**	.976±.001	.977±.002	.955±.003	.973±.002	.952±.003
SLeaf1	.817±.000	.835±.008	.817±.000	.794±.014	.799±.016	.798±.016	.816±.016	**.870±.021**	.838±.017
SLeaf2	.949±.000	.963±.010	.949±.000	.955±.007	.963±.000	.962±.004	.963±.000	**.980±.007**	.953±.010
SLeaf3	.838±.000	.883±.016	.836±.000	.835±.010	.836±.011	.837±.011	.829±.005	.842±.007	**.849±.014**
SLeaf4	.849±.000	**.934±.010**	.610±.000	.724±.024	.719±.007	.733±.004	.894±.015	.834±.029	.888±.019
SLeaf5	.883±.000	**.902±.011**	.883±.000	.894±.007	.892±.007	.891±.006	.870±.008	.892±.010	.897±.008
TwoPatterns1	.883±.000	**.926±.003**	.718±.000	.776±.002	.776±.002	.780±.003	.897±.002	.842±.006	.842±.006
TwoPatterns2	.728±.000	.783±.007	.653±.000	.688±.003	.687±.001	.689±.002	.739±.002	.755±.005	**.819±.006**
TwoPatterns3	.344±.000	.869±.003	.659±.000	.708±.004	.710±.005	.712±.007	.571±.002	.825±.006	**.896±.003**
TwoPatterns4	.852±.000	**.908±.005**	.681±.000	.732±.002	.737±.002	.735±.002	.736±.002	.824±.007	.875±.004
Wafer1	.996±.000	.996±.000	.996±.000	.996±.000	.996±.000	.996±.000	.996±.000	.996±.000	.996±.000
Wafer-1	.990±.000	**.992±.001**	.991±.000	.991±.001	.991±.000	.991±.001	**.992±.001**	.991±.001	.991±.001
Yoga1	.878±.000	.896±.002	.886±.000	.902±.001	.903±.002	**.905±.002**	.883±.000	.880±.001	.885±.002
Yoga2	.907±.000	**.917±.002**	.905±.000	.912±.001	.913±.001	.912±.001	.914±.000	.905±.001	.911±.001

Table 2. Comparison of F-values for various oversampling methods on unimodal positive class

behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, pp. 20–29, 2004.

[8] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intelligent Computing*, 2005, pp. 878–87.

[9] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. Int. Conf. Neural Networks*, 2008, pp. 1322–28.

[10] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. on System, Man and Cybernetics*, vol. 39, no. 2, pp. 539–50, Apr 2009.

[11] 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, pp. 30–39, 2004.

[12] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *ACM SIGKDD Explorations Newsletter*, vol. 6, pp. 40–49, 2004.

[13] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "SVMs modeling for highly imbalanced classification," *IEEE Trans. on System, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 1, pp. 281–88, 2009.

[14] S. Koknar-Tezel and L. J. Latecki, "Improving SVM classification on imbalanced time series data sets with ghost points," *Knowledge and Information Systems*, vol. 28, no. 1, pp. 1–23, Jul 2011.

[15] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, pp. 1263–84, Sept 2009.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2008.

[17] F. Bach and M. I. Jordan, "Learning graphical models for stationary time series," *IEEE Trans. on Sig. Proc.*, vol. 52, no. 8, pp. 2189–99, 2004.

[18] M. Meilă and M. I. Jordan, "Learning with mixtures of trees," *Journal of Machine Learning Research*, vol. 1, pp. 1–48, Oct 2000.

[19] V. Y. F. Tan, A. Anandkumar, and A. S. Willsky, "Learning Gaussian tree models: Analysis of error exponents and extremal structures," *IEEE Trans. on Sig. Proc.*, vol. 58, no. 5, pp. 2701–2714, May 2010.

[20] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience, 2nd edition, 2006.

[21] S. Lauritzen, *Graphical Models*, Oxford University Press, USA, 1996.

[22] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 2nd edition, 1988.

[23] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Inf. Th.*, vol. 47, no. 2, pp. 498–519, Feb 2001.

[24] C. K. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees.," *IEEE Trans. on Inf. Th.*, vol. 14, no. 3, pp. 462–467, May 1968.

[25] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, McGraw-Hill Science/Engineering/Math, 2nd edition, 2003.

[26] V. Y. F. Tan, S. Sanghavi, J. W. Fisher, and A. S. Willsky, "Learning Graphical Models for Hypothesis Testing and Classification," *IEEE Trans. on Sig. Proc.*, vol. 58, no. 11, pp. 5481–95, Nov 2010.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, no. 1–38, 1977.

[28] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Berlin: Springer-Verlag, 1995.

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

[30] H. Cao, X.-L. Li, Y.-K. Woon, and S.-K. Ng, "Integrated oversampling for imbalanced time series classification," *IEEE Trans. on Knowledge and Data Engineering*, Apr 2013.

[31] X. Xi, E. Keogh, C. Shelton, and L. Wei, "Fast time series classification using numerosity reduction," in *Proc. Int. Conf. Machine Learning*, 2006, pp. 1033–40.