Benchmarking Multi-instance Learning for Multivariate Time Series Analysis

Rufat Babayev(✉) and Lena Wiese

Fraunhofer ITEM, Nikolai-Fuchs-Strasse 1, 30625 Hannover, Germany
{rufat.babayev,lena.wiese}@item.fraunhofer.de

Abstract. Successful incorporation of Electronic Health Records to the data mining tools created new frontiers in digital clinical data analysis. One of the well-known applications of clinical data analysis is the mortality prediction of patients in intensive care units (ICUs). One important aspect of mortality prediction is the analysis of multivariate time series of observations after 24 or 48 h of ICU admission. Recent mortality prediction models for ICU patients are based on either recurrent neural networks or traditional machine learning algorithms using statistical summaries of timestamped observations. Instead of using complex neural network architectures and statistical summaries, we transform multivariate time series into multi-instance representation by keeping the expressiveness of the original observations. We then perform mortality prediction using multi-instance machine learning algorithms. Our empirical study shows that multi-instance representation achieves comparable or better (in some configurations) performance in various experiments.

Keywords: Multi-instance learning · Multivariate time series analysis · Machine learning using statistical summaries · Descriptive statistics · Ensemble methods

1 Introduction

Electronic Health Records (EHRs) contain an electronic medical history of different patients collected over time, namely, the key clinical data related to the health routine of patients. Clinicians need to examine these data and prepare treatment options for patients in a short period of time [21,22,37]. A fast diagnosis and treatment is especially important for the patients staying in intensive care units (ICUs), because they are admitted to these units in extreme situations. Hospitals and health care providers are also interested in a status of patients in intensive care units such as how long a patient is going to stay in ICU, whether or not a patient is going to die after certain number of hours (e.g. 24 or 48 h after ICU admission). With this grounding, they are able to organize future actions to save time, cost and other resources required for each patient.

This work was supported by the Fraunhofer Internal Programs under Grant No. Attract 042-601000.

To determine future patient status, monitoring data are utilized. These data are mostly collected as multivariate time series containing values for each key health indicator (e.g. heart rate, respiratory rate, Creatinine level, etc.) in a temporal order. In our study, we focus on mortality prediction after 48 h of ICU admission through multivariate time series classification.

A lot of research has been carried out for multivariate time series analysis of health monitoring data [3,8,19,32,40]. In their experiments, the authors either use deep neural networks (especially recurrent neural networks) for sequence/temporal modeling or create feature spaces from time series variables digested by traditional machine learning algorithms such as random forests, logistic regression classifiers, support vector machines, etc. Recurrent neural networks are useful tools to learn sequential or temporal relationships from time series data [32]. However, there are still some open questions remaining about the possible effectiveness of deep learning models for health care data. For example, the size of the data in health care applications is often modest relative to the complexity of deep learning models [19]. More specifically, these models can easily overfit on small-scale data. Moreover, complex architectures and parameter configurations need to be maintained for training. In contrast, traditional machine learning algorithms are not as complex as deep neural networks, however, the vast majority of them are not designed in a way that they can handle sequential/temporal data. To cope with this issue, the straightforward approach is to map multivariate time series data to the data with a single instance (or a propositional) feature space. As an example, assume that a multivariate time series contains T observations of D variables in a temporal order. One can generate the following statistical summaries for each variable $d \in \{1, \ldots, D\}$ from that time series by considering all T observations; *Maximum*, *Minimum*, *Mean*, *Median*, *Mode*, *Standard deviation*, *Variance*, *Range*, *Geometric center*, *Kurtosis*, *Skewness*, *Averaged power*, *Energy spectral density* [35]. Then a feature space can be generated by all of these summaries or a subset of them. Now assume that the *Maximum* and *Minimum* are selected, then a single instance obtained from the respective time series becomes a part of *Minimax* feature space containing $2 \cdot D$ variables (features). In this approach, the time order is not considered and instead of using raw features and their values, generated features and their corresponding values are applied for training machine learning models. This kind of approach does indeed show decent predictive performance [3,8,19,35]. In our work, we focus on using as many raw features as possible to keep the expressiveness of the original dataset. To do so, we represent multivariate time series data as multi-instance data in the context of an MIL (Multi-Instance Learning) framework. In this representation, each observation of variables in a time series is considered as one instance and a collection of such instances is denoted as a *bag* with a corresponding class label. Finally, we benchmark multivariate time series classification by classifying bags with multi-instance versions of traditional machine learning algorithms in the MIL framework. Our results show that the multi-instance representation yields comparable or sometimes better (in some configurations) performance as compared to the statistical summary representation.

2 Related Work

Multi-Instance Learning (MIL) is a notable topic in machine learning proposed in 1997 [9] as a variation of supervised learning (weakly supervised) for drug activity prediction. Since then, MIL frameworks are adapted for many other areas. For instance, [2] incorporated SVM (support vector machines) to the MIL framework and proposed MISVM to generate instance-level and bag-level predictions effectively. The work by [45] expanded multi-instance SVM approaches through MIMLSVM (multi-instance multi-label SVM) for solving multi-label classification tasks. MIGraph [46] is proposed to model multi-instance bag structures. Generative mixture models – MIMM (multi-instance mixture model) [11], and DPMIL (dirichlet process mixture of gaussians) [27] are adapted to tackle binary multiple-instance classification problems. Deep multi-instance learning methods are also introduced in an MIL setting [23,30,31,42,44,47]. In-depth surveys of MIL frameworks are given in works such as [1,7,20,39]. Time series analysis is adopted into the MIL framework by [17] which utilizes an autoregressive hidden markov model for an activity recognition in time series data. The work by [36] proposes a multi-instance learning method for a sound event detection from time series. Multi-instance learning approach based on the time series modeling for EEG (Electroencephalogram) identification is proposed by [24].

Despite some applications, we have not noticed any work which benchmarks multi-instance learning on multivariate time series data. The goal of our work is to achieve this in the context of multivariate time series classification.

3 Preliminaries

In this section, we present mathematical notations for a multivariate (multi-dimensional) time series and briefly discuss background for the multi-instance learning in this context.

Following the notations from [8], we specify a multivariate time series with D variables (also known as a D-dimensional time series) of length T as $X = (x_1, x_2, \ldots, x_T)^\intercal \in \mathbb{R}^{T\times D}$, where $\forall t = \{1, 2, \ldots, T\}$, $x_t \in \mathbb{R}^D$ is a vector which represents the t-th measurements (observations) of all variables and x_t^d is the observation of d-th variable of x_t. In this paper, we focus on time series classification to predict a label $l_n \in \{1, \ldots, L\}$ for each of N multivariate time series collected in a dataset $\mathcal{D}$, where $\mathcal{D} = \{(X_n)\}_{n=1}^{N}$ and $X_n = \left[x_1^{(n)}, x_2^{(n)}, \ldots, x_{T_n}^{(n)}\right]$.

3.1 Multi-instance Learning

Multi-instance learning (MIL) is a type of supervised learning where the data points are collected in multisets called bags, and the entire bag has a label – either discrete or real-valued. The data points of each bag are called instances. The main purpose is to learn a model from the instances of the bag and the label of the bag such that bag-level and instance-level predictions can be generated. Our focus in this work is a classification task (i.e. discrete-valued labels), more specifically binary classification. In general, there are two types of assumptions that can be used to model relationships between the bag label and labels of instances inside the bag. The first assumption is called the *standard MI assumption* [9]. In this assumption, the bag label is considered negative if all instances inside the bag are negative and it is considered positive if at least one instance inside the bag has a positive label. We follow the notations provided by [7] to explain the assumptions. Let B be a bag of M instances with static features (a.k.a propositional instances or feature vector instances), namely, $B = \{z_1, z_2, \ldots, z_M\}$. Assume that $\forall m \in \{1, \ldots, M\}$, an instance z_m in a feature space $\mathcal{Z}$ is mapped to a class by some imaginary function $f : \mathcal{Z} \rightarrow \Omega$, where $\Omega = \{0, 1\}$, and where 0 and 1 represent negative and positive labels correspondingly. Then, the bag classifier (a.k.a the aggregator function), $g(B)$ is given by:

$$g(B) = \begin{cases} 1 & \text{if } \exists z \in B : f(z) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

This standard assumption might be viewed as too strict for some cases where positive bags cannot be determined by a single instance of a bag. Therefore, this assumption is relaxed to the *collective MI assumption* [10,43] which treats the contribution of each instance to the bag label separately. In contrast to the standard assumption, the collective assumption considers a bag B as a distribution $P(z|B)$ (the probability of an instance z given a bag B) over the feature space $\mathcal{Z}$, and similarly considers labels as a distribution $P(c|z)$ over instances, where $c \in \Omega = \{0, 1\}$. The collective assumption then models the distribution

$$P(c|B) = \int_{\mathcal{Z}} P(c|z)P(z|B)dz. \tag{2}$$

To calculate this, the probability distribution $P(z|B)$ for the bag must be known. Generally, this probability distribution is not known in practice; hence, an empirical version over the instances in the bag is calculated instead:

$$\hat{P}(c|B) = \frac{1}{M_B} \sum_{m=1}^{M_B} P(c|z_m), \tag{3}$$

where M_B is the number of instances inside the bag B. Since $P(c|z_m)$, $\forall m = \{1, \ldots, M_B\}$ is also unknown, most methods based on the collective assumption learn this distribution as in a single-instance dataset [10,43]. The probability distribution in (3) is also called an *arithmetic average of posterior probabilities* of instances in the bag. In this probability distribution, the instance-level class label is modeled by $P(c|z_m)$ for each $m \in \{1, \ldots, M_B\}$ for the bag B. It can also be modeled by a logit transformation, namely, the log-odds function

$\log \frac{P(c=1|z_m)}{P(c=0|z_m)}$ [10]. When the logit transformation is substituted in (3), the following equation is obtained:

$$\begin{aligned}
&\log \frac{P(c=1|B)}{P(c=0|B)} = \frac{1}{M_B} \sum_{m=1}^{M_B} \log \frac{P(c=1|z_m)}{P(c=0|z_m)} \\
&= \frac{1}{M_B} \log \left[\frac{P(c=1|z_1)}{P(c=0|z_1)} \cdot \ldots \cdot \frac{P(c=1|z_{M_B})}{P(c=0|z_{M_B})} \right] \\
&\Rightarrow \frac{1-P(c=0|B)}{P(c=0|B)} = \frac{[\prod_{m=1}^{M_B} P(c=1|z_m)]^{1/M_B}}{[\prod_{m=1}^{M_B} P(c=0|z_m)]^{1/M_B}} \\
&\Rightarrow \begin{cases} P(c=1|B) \\ = \frac{[\prod_{m=1}^{M_B} P(c=1|z_m)]^{1/M_B}}{[\prod_{m=1}^{M_B} P(c=1|z_m)]^{1/M_B} + [\prod_{m=1}^{M_B} P(c=0|z_m)]^{1/M_B}} \\ \\ P(c=0|B) \\ = \frac{[\prod_{m=1}^{M_B} P(c=0|z_m)]^{1/M_B}}{[\prod_{m=1}^{M_B} P(c=1|z_m)]^{1/M_B} + [\prod_{m=1}^{M_B} P(c=0|z_m)]^{1/M_B}} \end{cases}
\end{aligned} \tag{4}$$

Equation (4) is called a (normalized) *geometric average of posterior probabilities* (or an arithmetic mean of log-posterior) [10] of instances in the bag.

The collective assumption weights every instance inside a bag equally. The paper [12] presents a collective assumption with instance weights. It is called the (arithmetic) *weighted collective MI assumption* and simply utilizes weights of instances inside a bag to calculate the probability distribution

$$\breve{P}(c|B) = \frac{1}{w_B} \sum_{m=1}^{M_B} w(z_m) \cdot P(c|z_m), \tag{5}$$

where $w : \mathcal{Z} \rightarrow \mathbb{R}^+$ is a weight function over instances and $w_B = \frac{1}{M_B} \sum_{z \in B} w(z)$ [10]. The probability distribution for a geometric weighted collective assumption can be calculated similarly. Finally, an aggregator function for the collective assumptions can be defined as follows:

$$g(B) = \begin{cases} 1 & \text{if } P(c=1|B) \geq P(c=0|B) \\ 0 & \text{otherwise} \end{cases}. \tag{6}$$

3.2 Multivariate Time Series in the MIL Framework

We incorporated multivariate time series into the MIL framework as follows; we consider $\forall n = \{1, \ldots, N\}$ a multivariate time series X_n as one bag, and the observations of all D variables at each time step $t \in \{1, \ldots, T_n\}$ as an instance of this bag. More formally, the bag B_n of X_n is defined as $B_n = \{x_1^{(n)}, x_2^{(n)}, \ldots, x_{T_n}^{(n)}\}$, where the class label of X_n becomes the bag label of B_n. With this in mind, various multivariate time series having different lengths can be considered inside their own encapsulating bags, so that the expressiveness of the original values is maintained without rescaling the time series data.

4 Empirical Evaluation

4.1 Dataset and Task Description

We evaluate the performance of a classification in the MIL framework on multivariate time series data using several robust machine learning approaches specifically used for this kind of classification tasks. We evaluate our models for different settings such as for the data obtained by different imputations methods, by boosting and bagging.

PhysioNet Challenge 2012 Dataset (PhysioNet). This dataset is from PhysioNet Challenge 2012 [38] which is a publicly available[1] collection of multivariate clinical time series records of 12000 intensive care unit (ICU) patients. Each record is a multivariate time series of 48 h after ICU admission of a corresponding patient and contains 36 variables such as *mean arterial blood pressure*, *heart rate*, *respiratory rate*, etc. The dataset is divided into three sets (Set-A, Set-B, Set-C) each having 4000 multivariate time series. Set-C is designated for the reviews of the challenge, so we did not use it. We used Set-A and Set-B in our experiments. We perform the mortality prediction task on this dataset to predict whether a patient dies in a hospital after 48 h. We designate the class of death as a positive class (with the label of 1) and the class of survival as a negative class (with the label of 0). This is a binary classification task. There are 554 positively labeled multivariate time series, and 3446 negatively labeled multivariate time series in Set-A. For Set-B and Set-C, these numbers are 568–3432, and 585–3415 respectively. The class imbalance for each of these sets is roughly 14% (positive)–86% (negative). We test different approaches to handle the class imbalance, i.e. through undersampling or oversampling.

Because the PhysioNet dataset is collected from Electronic Health Records, it has missing values. We replace missing values with **Mean** and **Forward** methods. Apart from that (similarly to [8]), we combine the invasive blood pressure variables DiasABP (diastolic arterial blood pressure), SysABP (systolic arterial blood pressure) and MAP (mean arterial blood pressure) with noninvasive ones, i.e., NIDiasABP, NISysABP and NIMAP respectively which effectively reduces the number of variables to 33. The combination of variables enables us to reduce the number of missing values as well. More formally, it is possible to obtain one value from the other using the following formula [6]:

$$\mathrm{MAP} = \frac{(2 \cdot \mathrm{DiasABP} + \mathrm{SysABP})}{3}. \tag{7}$$

[1] https://physionet.org/content/challenge-2012/.

In this case, if a MAP value is not present for some time step, then it can be calculated from the existing DiasABP and SysABP value of that time step. The similar approach is also used for missing DiasABP and SysABP values. After that, it is possible to replace missing values with the computed values. The main benefit here is the replacement of missing values with the real values, instead of imputed ones. During the combination, if there are existing values for invasive and noninvasive counterparts for the same time step, then we prefer an invasive measurement instead of noninvasive one [33]. Our combination of variables differs from [8] in a few more nuances. For example, instead of using raw timestamps in each multivariate time series, they take hourly samples of observations. Moreover, they perform forwarding through hourly/2-hourly samples in their forward imputation method. In terms of expressiveness, we do not apply such stages.

4.2 Machine Learning Approaches

To benchmark the multivariate time series classification in the MIL framework, we used different multi-instance learners from the WEKA machine learning workbench (version 3.7.2) [18]. The multi-instance learners are available under the ***weka.classifiers.mi*** package. We explicitly tested the following learners:

- ***weka.classifiers.mi.MILR*** uses either the standard or collective multi-instance assumption, but within a logistic regression. We picked the collective assumption with the geometric average of posterior probabilities which outperformed other assumptions for this learner.
- ***weka.classifiers.mi.MIWrapper*** [13] is a simple Wrapper class for applying standard propositional (feature vector) learners to multi-instance data. As the first step, MIWrapper gathers instances from all bags, and labels each instance with the label of its bag. This step creates a propositional (i.e. single-instance) version of the multi-instance dataset. Then, it weights all instances such that each bag has equal cumulative total weight. Different weighting schemes are available; we used unit weighting for propositional instances. After weighting, a single-instance (feature vector) learner is utilized for this propositional dataset. During the learning phase, the single-instance learner estimates class probabilities for all instances inside the bag for which the bag label should be generated. The generated bag label is simply the mean (arithmetic or geometric) of the estimated class probabilities of the corresponding instances [10]. For MIWrapper, we selected
  - ***weka.classifiers.trees.RandomForest***
  - ***weka.classifiers.functions.Logistic***

  as propositional learners and used the collective assumption with the geometric average which performed better than other collective assumptions. The latter class refers to the logistic regression (LR). Both LR and the random forest (RF) are widely used in health care applications [3,8,19,40].
- ***weka.classifiers.mi.SimpleMI*** reduces multi-instance data into single-instance data by taking an arithmetic or geometric average of variable (feature) values of instances or by creating a minimax feature space from instances

inside each bag. After reducing each bag into a single instance or feature vector, single-instance learners such as the random forest or the logistic regression can be used for modeling. In our experiments, we make use of the arithmetic average of variable values which provides higher performance. This scheme is equivalent to the feature space obtained by *Mean* statistical summary having the same number of variables (features) as of multi-instance data (see Sect. 1).

- ***weka.classifiers.meta.RealAdaBoost*** is a class for boosting a binary classifier using the Real Adaboost method [16]. We utilized this class to boost binary classifiers which are wrapped by MIWrapper and SimpleMI.
- ***weka.classifiers.meta.Bagging*** [5] is a class for bagging a classifier to reduce variance. It can perform a classification and regression depending on the base learner. We used this class as a meta learner for MILR and for the logistic regression wrapped by SimpleMI.

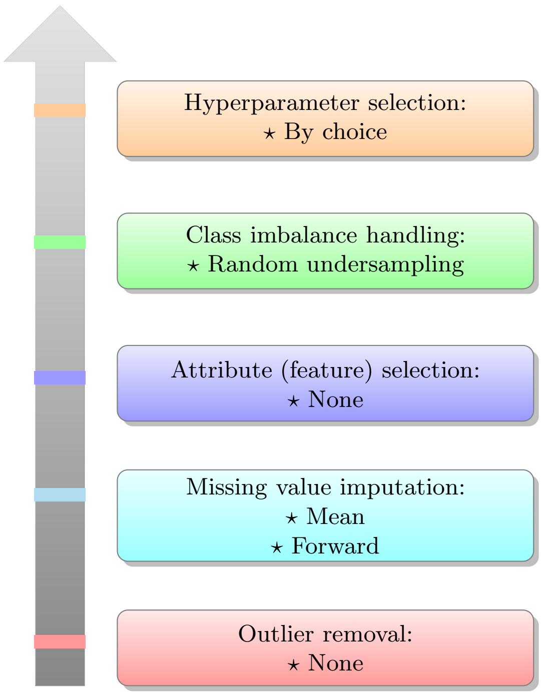
Hyperparameter selection:
⋆ By choice
Class imbalance handling:
⋆ Random undersampling
Attribute (feature) selection:
⋆ None
Missing value imputation:
⋆ Mean
⋆ Forward
Outlier removal:
⋆ None

Fig. 1. Multi-instance learning pipeline

Remember that a bagging (**b**ootstrap **agg**regat**ing**) [5] and boosting are ensemble machine learning methods to enhance base classifiers for a better predictive performance. Thus, there is a clear distinction between an ensemble method *bagging* and bags used in the MIL setting where each bag is a multiset instance.

In our study, we compare the performance of learners in the multi-instance context with the learners using the data obtained by SimpleMI (i.e. statistical summary representation). We use short names of learners for the demonstration of results such as SMI for SimpleMI, MIW for MIWrapper, RB for RealAdaBoost, BG for Bagging, RF for random forest and LR for logistic regression.

4.3 Multi-instance Learning Pipeline

To make the data ready for multi-instance learners, the data preprocessing is performed through the pipeline presented in Fig. 1. More comprehensively:

- Physionet dataset is a dataset containing the first 48 h of recordings after ICU admission. After 48 h, either a patient died or survived. Patients are admitted to the intensive care unit (ICU) in extreme circumstances. In this case, their health recordings may contain values deviating from the rest of the population in the ICU. However, these values are still valid and might not be taken as outliers. From the statistical point of view, traditional outlier removal methods (e.g. standard deviation based or median absolute deviation based) can easily strip out this information from the patients' data. For the sake of expressiveness, we do not explicitly perform outlier removal. We also observed that the authors of Physionet dataset successfully removed the medically implausible values during the dataset creation [25].
- For imputing missing values we used mean and forward methods which show decent results [8,32].
  - **Mean** (shortly M) [8] – a mean value for each of 33 variables is computed from the existing measurements in all multivariate time series in Set-A. Then, missing values for each variable in Set-A and Set-B are replaced by the corresponding mean value.
  - **Forward** (shortly F) [32] – in this strategy, we impute the missing value x_d^t of a variable d at a time step t as follows; if there is at least one measurement which is recorded previously for a variable d at a time $t' < t$, we perform a forward imputation by $x_d^t \leftarrow x_d^{t'}$. If there is no measurement that is recorded previously (or if the variable is completely empty), then we compute the median over all existing measurements in Set-A and replace the missing values by the respective median in both Set-A and Set-B.
- We do not perform an attribute selection for Physionet dataset. Our purpose is to make the version of the dataset comparable to the dataset used in [8] which also applies a variable combination we explained in Sect. 4.1.
- The class imbalance for each subset (Set-A, Set-B, Set-C) of Physionet dataset is around 14% (positive)–86% (negative). To handle this problem, we make use of SpreadSubsample class (shortly SS) from the package ***weka.filters.supervised.instance*** with a distribution 1.0 which is a random undersampling effectively reducing the negative class to the size of the positive class (a class imbalance of 50%-50%). We also checked random oversampling of the positive class to the size of negative class using the respective

WEKA class. However, this caused higher false negatives in our experiments. Oversampling is a data generation process and we think that more sophisticated oversampling strategy is required to mimic the existing multivariate time series data (especially for the health-related data).
- Finally, we set hyperparameters for machine learning algorithms by choice. For instance, for the RF, we set the number of trees to 100 (which demonstrated better performance and is tolerable in terms of runtime). For boosting algorithms, the number of boosting iterations is selected to be 10. For runtime constraints, we do not apply hyperparameter tuning through the cross-validation (CV) or grid search, because the learners are wrapped by classes where each wrapper has its own parameters. The other parameters of learners are WEKA defaults.

4.4 Experimental Setup

In our experiments, two different setups are tested. For each setup, an AUROC value of the classification is reported. The AUROC is a standard metric for evaluating the performance of classifiers. The ***weka.classifiers.Evaluation*** class is applied for the evaluation.

1. Learners are trained on Set-A and tested on Set-B in 10 runs. In each run, Set-A is randomly shuffled with a different random seed and then a train/test procedure applied. Finally, the results are averaged.
2. Predictive models are built using a stratified CV on Set-A and then an average AUROC is reported. Some papers [8] only used this setup, since at their time of writing, class labels were not available for Set-B and Set-C.

4.5 Interpretation of Results

Results are generated for different configurations. Each configuration is titled by the short names of learners and short names of pipeline stages. For instance, RB-MIW-RF-M-SS means that the data are mean imputed (M), a class imbalance is handled by SpreadSubsample (SS) (undersampled) and the resulting data are learned by the random forest (RF) which is wrapped by MIWrapper (MIW) and boosted by RealAdaBoost (RB). For BG-SMI-LR-F-SS, the data are forward imputed (F), a class imbalance is handled by SpreadSubsample (SS) (undersampled), then the data are transformed into a single-instance format (using *Mean* statistical summary) by SimpleMI (SMI), and the resulting data are learned by LR which is enhanced by Bagging (BG). The other configurations can be understood similarly.

The bagging is a technique to reduce the complexity of models that overfit during training, whereas boosting is used to increase the complexity of models subject to high bias, thus, handles underfitting during training. We observe that the RF is better enhanced by RealAdaBoost and LR is by Bagging through systematically testing different meta learners that WEKA provides. These boosting and bagging combinations yield more balanced learners for our experiments where we compare them to non-bagged and non-boosted variants. In this manner, the RF is boosted by RealAdaBoost in both multi-instance and propositional settings. MILR and the propositional logistic regression (SMI-LR) are enhanced by Bagging. The only exception is a logistic regression in the multi-instance setting which is wrapped by MIWrapper; we find that RealAdaBoost improves the base learner better than Bagging (especially for the mean imputation) so that we provide the results for the former meta learner. The main benefit of RealAdaBoost is that it improves the base learner by adapting predicted class probabilities of instances [34] which is useful in a multi-instance setting – a bag label is determined collectively by the probability distribution of labels of its instances.

Table 1. Model performances measured by average AUROC score for mortality prediction. The weight of each instance in a bag is 1. After propositional conversion unit-weighting for each single instance is still maintained. The results are generated by train/test procedure through 10 runs.

Mortality prediction on PhysioNet dataset			
MILR		BG-MILR	
MILR-M-SS	0.8051	BG-MILR-M-SS	0.8075
MILR-F-SS	0.8094	BG-MILR-F-SS	0.8098
MIW-RF		MIW-LR	
MIW-RF-M-SS	0.7736	MIW-LR-M-SS	0.7645
MIW-RF-F-SS	0.8146	MIW-LR-F-SS	0.8094
RB-MIW-RF		RB-MIW-LR	
RB-MIW-RF-M-SS	0.7808	RB-MIW-LR-M-SS	0.8035
RB-MIW-RF-F-SS	**0.8190**	RB-MIW-LR-F-SS	0.8044
SMI-RF		SMI-LR	
SMI-RF-M-SS	0.8212	SMI-LR-M-SS	0.8051
SMI-RF-F-SS	0.8224	SMI-LR-F-SS	0.8094
RB-SMI-RF		BG-SMI-LR	
RB-SMI-RF-M-SS	0.8301	BG-SMI-LR-M-SS	0.8075
RB-SMI-RF-F-SS	**0.8313**	BG-SMI-LR-F-SS	0.8098

In Table 1, we report the results of the first experimental setup (train/test) and in Table 2, we show the results for the second setup (stratified CV). In both tables, we tested all configurations for unweighted instances in the bag, namely the weight of every instance is 1 in each bag. In the experiments, a unit weighting is also maintained after propositional conversion. In multi-instance configurations, bags are also unit-weighted.

We observe that the settings we set for MILR, BG-MILR and SMI-LR, BG-SMI-LR respectively, resulted in a similar predictive performance. MILR and BIG-MILR applies a geometric average of posterior probabilities of instances inside a bag to obtain a bag label, however, SMI-LR and BG-SMI-LR uses *Mean* statistical summary of instances inside a bag during training. We notice that a bagging slightly improves MILR and SMI-LR performances for both setups.

As compared to MILR (in MILR-M-SS), MIW-LR demonstrates slightly lower performance for the mean imputation (MIW-LR-M-SS). The MIWrapper (MIW) performs a propositional conversion and generates a bag label from the estimated class probabilities of bag's instances. Remember that the mean imputation replaces all missing values with the corresponding mean value. Thus, it inherently causes the creation of more similar propositional instances (after conversion) from each bag and across the bags as compared to the forward imputation which in turn negatively affects the performance of a bag label prediction from the class probabilities of respective instances for the LR. The similar phenomenon also occurs for MIW-RF with the mean imputation (namely, MIW-RF-M-SS). The forward imputation enables MIW-RF (in MIW-RF-F-SS) and MIW-LR (in MIW-LR-F-SS) to show similar or better performance than MILR (i.e. MILR-F-SS).

Table 2. Model performances measured by average AUROC score for mortality prediction. The weight of each instance in a bag is 1. After propositional conversion unit-weighting for each single instance is still maintained. The results are generated by 10-fold CV on Set-A.

Mortality prediction on PhysioNet dataset			
MILR		BG-MILR	
MILR-M-SS	0.7540	BG-MILR-M-SS	0.7583
MILR-F-SS	0.7679	BG-MILR-F-SS	0.7685
MIW-RF		MIW-LR	
MIW-RF-M-SS	0.7537	MIW-LR-M-SS	0.7274
MIW-RF-F-SS	0.7917	MIW-LR-F-SS	0.7685
RB-MIW-RF		RB-MIW-LR	
RB-MIW-RF-M-SS	0.7563	RB-MIW-LR-M-SS	0.7518
RB-MIW-RF-F-SS	**0.7977**	RB-MIW-LR-F-SS	0.7639
SMI-RF		SMI-LR	
SMI-RF-M-SS	0.8113	SMI-LR-M-SS	0.7540
SMI-RF-F-SS	0.7939	SMI-LR-F-SS	0.7679
RB-SMI-RF		BG-SMI-LR	
RB-SMI-RF-M-SS	**0.8159**	BG-SMI-LR-M-SS	0.7583
RB-SMI-RF-F-SS	0.8101	BG-SMI-LR-F-SS	0.7685

The variants of MIWrapper (MIW) with RealAdaBoost, namely, RB-MIW-RF and RB-MIW-LR improves the performance of MIW-RF and MIW-LR respectively (except for MIW-LR-F-SS). Remember that 10 runs/folds used in both experimental setups averages the predictive performance of the boosted variants (where each variant internally does 10 iterations). Thus, even an averaging in our setups enables boosted variants enhance their non-boosted counterparts.

The multi-instance configuration with the highest performance in Table 1 is RB-MIW-RF-F-SS with **0.8190** average AUROC which improves upon MIW-RF-F-SS having 0.8146 average AUROC. We observe the similar case in Table 2.

In Table 1, the single-instance configuration with *Mean* statistical summary shows the highest performance in RB-SMI-RF-F-SS with **0.8313** average AUROC. In our tests, it is even higher than *Geometric Center* and *Minimax* statistical summaries that SimpleMI class provides for the same configuration. It seems that *Mean* statistical summary better reflects original data points than the other summaries. In our experiments, we utilize *Mean* statistical summary for all configurations containing SimpleMI (SMI).

The single-instance configuration with *Mean* statistical summary showing the highest performance is RB-SMI-RF-M-SS having **0.8159** average AUROC in Table 2. In general, the CV (Table 2) does not yield higher results as compared to the train/test setup in Table 1. The main reason of that is the number of instances used in both setups. The train/test setup has more instances to learn/test (since it trains on Set-A and tests on Set-B) than the CV setup (which does a stratified CV on Set-A). In both setups, the forward imputation generally yields better results as compared to the mean imputation.

Our experimental setups show that the multi-instance learners are capable of performing multivariate time series classification in a decent level. We expect that they can show even higher performance after more careful data preparation and in more sophisticated parameter configurations.

5 Discussion

5.1 The Other Multi-instance Learners

In our tests, we comparably examined different multi-instance learners from the ***weka.classifiers.mi*** package in terms of their heuristics, hyperparameter space and predictive performance with respect to the learners presented in Sect. 4.2. For some of them, the implementation is not relevant for a multivariate time series representation. For the others, they either need many hyperparameters to be adjusted or do not provide an adequate predictive performance in their default settings. For instance:

– ***weka.classifiers.mi.MIBoost*** is a multiple instance AdaBoost method which considers the geometric average of posteriors of instances in the bag and takes the expectation for a bag inside the loss function [15]. Analogous to MIWrapper, it is possible to wrap RF and LR by MIBoost. The default number of boosting iterations is set to 10 by WEKA.
    - The one drawback of this method is that AdaBoost adapts itself according to the amount of error on predicted classes of instances [34] rather than class probabilities of instances like RealAdaBoost. Thus, it may not be effective in a multi-instance setting.

  - Another drawback is that it internally converts all instances in the bag to the propositional format and weights each propositional instance by *total number of propositional instances after conversion*/(*total number of bags* ∗ *total number of instances in the corresponding bag*). This weighting scheme is not appropriate for a multivariate time series representation and we indeed obtained relatively low predictive performance in the tests with respect to our chosen learners (i.e. MILR, MIW-LR, MIW-RF and their boosted/bagged variants).

– ***weka.classifiers.mi.MISVM*** is a class which implements MISVM [2] (Maximum pattern Margin Formulation of MIL). It internally applies the algorithm called ***weka.classifiers.functions.SMO*** [28] to solve multiple instance problem.

  - We observed that its predictive performance was relatively low (in default settings) as compared to the chosen learners. Moreover, many hyperparameter configurations need to be maintained (such as a kernel type, a complexity constant, a cache size, a usage of lower order terms, etc.) [2].

Our findings showed that the learners we chose in our experimental setups were suitable for the multivariate time series representation in the MIL setting, required less hyperparameter space to adjust and displayed a decent classification performance in a straightforward comparison.

5.2 Hyperparameters

We investigated different hyperparameters for our learners to find the proper ones. In WEKA (version 3.7.2), the RF implementation has 10 trees by default. Instead, we used 100 trees in our experiments. We additionally observed that 1000 trees slightly improved the performance of configurations containing the RF in Sect. 4.5. However, this number brought additional runtime overhead (i.e. for the configurations which also employed RealAdaBoost).

Both MILR and the original LR have a parameter to set the ridge in the log-likelihood. The former used 10^{-6} and the latter used 10^{-8} as its ridge parameter which resulted in an adequate performance both in multi-instance and single-instance settings.

As a multi-instance hyperparameter, our selected multi-instance learners employed the collective assumption with the geometric average of posteriors in all experiments which outperformed the other assumptions including the standard assumption and the collective assumption with the arithmetic average.

5.3 Future Work

In this section, we discuss the future insights to improve the predictive capabilities of multi-instance learners for multivariate time series analysis.

Weighting Instances in the Bag. When a multivariate time series is represented in the MIL framework, each instance in a bag is treated equally without a time order (similarly to statistical summaries where the time order is also discarded). In this case, every instance in each bag has a weight of 1 by default. To incorporate the temporal order to the multi-instance learning, we checked weighted inner bag instances in one of our tests. Our weighting scheme is straightforward. In Physionet dataset every time step of a multivariate time series has its own timestamp value. That value is in *hh:mm* format, namely, numbers of hours and number of minutes after ICU admission. We converted each timestamp value to minutes. More formally, for each multivariate time series X_n, $\forall n \in \{1, \ldots, N\}$ in $\mathcal{D}$, all respective timestamp minutes are summed up. Then for each t-th observations of D variables, i.e. $x_t \in X_n$, $\forall t \in \{1, \ldots, T_n\}$, x_t is weighted as the ratio of its timestamp minutes divided by the corresponding sum. Finally, the weighted instances are put to the respective bag B_n. In this weighting scheme, instances close to the 48-h threshold gain more weight in the temporal order. For this scheme, the weighted collective assumption of the MIL framework can be utilized for generating class labels of bags (see (Eq. 5)). Our findings and future proposals for this and other custom weighting schemes are given in the next two paragraphs.

We observed that a WEKA implementation of multi-instance learners does not support such a weighting scheme. We discovered that our weighting scheme is supported by SimpleMI (which performs a single-instance transformation) where *Mean* statistical summary also averages the weights of inner bag instances so that after transformation by SimpleMI (SMI), propositional instances gain different weights. In fact, this weighting scheme improved the performance of the configurations using SMI-LR, SMI-RF, BG-SMI-LR and RB-SMI-RF presented in Sect. 4.5. It is because the original Logistic and RandomForest class from WEKA can handle weighted instances. The highest performance is obtained in RB-SMI-RF-F-SS configuration with the average AUROC performance of **0.8427** (in the train/test setup) which is a decent improvement over the unweighted configuration which displays 0.8313. This insight augments our expectations that the multi-instance learners supporting such custom weighting schemes can get a similar performance improvement.

As a future work, one can port custom weighting schemes to multi-instance learners (e.g. MILR and MIWrapper) by modifying WEKA source code. Then, the similar tests from Sect. 4.5 can be performed to reveal the effectiveness.

Propositionalisation of Multivariate Time Series Data by Sophisticated Approaches. Remember that SimpleMI class is designed to generate three different statistical summaries, namely *Geometric Center*, *Mean* and *Minimax* from the multi-instance representation of multivariate time series data. There are also approaches to propositionalise multi-instance data by decision trees [41] and more ingeniously by random forests [14]. These approaches can create more advanced feature spaces from the multi-instance representation in contrast to the statistical summaries. As a future work, every multivariate time series can be propositionalised by one of these approaches in its MIL format and then the resulting data can be fed to traditional ML algorithms for classification.

6 Conclusion

In this paper, we benchmarked multi-instance learning for clinical multivariate time series classification (on the mortality prediction task). We utilized different multi-instance learners to study the time series data in multi-instance format and then used multi-instance assumptions of the MIL framework to generate class labels for each multivariate time series. We evaluated the multi-instance learners in different experimental setups and configurations using the well-known metric named AUROC in both multi-instance and propositional settings. We compared their performance to the performance of traditional machine learning algorithms using statistical summaries. Despite the fact that we focused on the mortality prediction using time series data collected in intensive care units, we believe that the multi-instance representation will be also useful for other tasks such as a length-of-stay prediction, phenotype classification, and psychological decompensation prediction. It will be also interesting to see the generalization of multi-instance learners on the other healthcare datasets such as MIMIC-III [26], EEG database dataset [4] and ICU dataset [29].

6.1 Code Availability

For the sake of reproducing the results obtained in this work, all our source code is published in a public repository[2].

References

1. Amores, J.: Multiple instance classification: review, taxonomy and comparative study. Artif. Intell. **201**, 81–105 (2013)
2. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Advances in Neural Information Processing Systems, vol. 15, pp. 561–568. MIT Press (2003)
3. Awad, A., Bader-El-Den, M., McNicholas, J., Briggs, J., El-Sonbaty, Y.: Predicting hospital mortality for intensive care unit patients: time-series analysis. Health Inform. J. **26**(2), 1043–1059 (2019). https://doi.org/10.1177/1460458219850323
4. Begleiter, H.: UCI machine learning repository: EEG database data set (1999). https://archive.ics.uci.edu/ml/datasets/eeg+database
5. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996). https://doi.org/10.1007/BF00058655
6. Brunner, L.S.: Brunner & Suddarth's Textbook of Medical-Surgical Nursing, vol. 1. Lippincott Williams & Wilkins (2010)
7. Carbonneau, M.A., Cheplygina, V., Granger, E., Gagnon, G.: Multiple instance learning: a survey of problem characteristics and applications. Pattern Recogn. **77**, 329–353 (2018)
8. Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. Sci. Rep. **8**(1), 1–12 (2018)
9. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artif. Intell. **89**(1–2), 31–71 (1997)

[2] https://github.com/CavaJ/time-series-analysis.

10. Foulds, J., Frank, E.: A review of multi-instance learning assumptions. Knowl. Eng. Rev. **25**(1), 1–25 (2010)
11. Foulds, J., Smyth, P.: Multi-instance mixture models and semi-supervised learning. In: Proceedings of the 2011 SIAM International Conference on Data Mining, pp. 606–617. SIAM (2011)
12. Foulds, J.R.: Learning instance weights in multi-instance learning. Ph.D. thesis, The University of Waikato (2008)
13. Frank, E.T., Xu, X.: Applying propositional learning algorithms to multi-instance data. Technical report, University of Waikato, Department of Computer Science, Hamilton, NZ, June 2003
14. Frank, E., Pfahringer, B.: Propositionalisation of multi-instance data using random forests. In: Cranefield, S., Nayak, A. (eds.) AI 2013. LNCS (LNAI), vol. 8272, pp. 362–373. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03680-9_37
15. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Thirteenth International Conference on Machine Learning, pp. 148–156. Morgan Kaufmann, San Francisco (1996)
16. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Stat. **95**(2), 337–407 (2000)
17. Guan, X., Raich, R., Wong, W.K.: Efficient multi-instance learning for activity recognition from time series data using an auto-regressive hidden Markov model. In: International Conference on Machine Learning, pp. 2330–2339 (2016)
18. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. **11**(1), 10–18 (2009)
19. Harutyunyan, H., Khachatrian, H., Kale, D.C., Steeg, G.V., Galstyan, A.: Multitask learning and benchmarking with clinical time series data. arXiv preprint arXiv:1703.07771 (2017)
20. Herrera, F., et al.: Multiple instance learning. In: Herrera, F., et al. (eds.) Multiple Instance Learning, pp. 17–33. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47759-6_2
21. Hesse, B.W., Ahern, D., Beckjord, E.: Oncology Informatics: Using Health Information Technology to Improve Processes and Outcomes in Cancer. Academic Press, Cambridge (2016)
22. Howie, J.G., Heaney, D.J., Maxwell, M., Walker, J.J., Freeman, G.K., Rai, H.: Quality at general practice consultations: cross sectional survey. BMJ **319**(7212), 738–743 (1999)
23. Huang, Y., Wang, W., Wang, L., Tan, T.: Multi-task deep neural network for multi-label learning. In: 2013 IEEE International Conference on Image Processing, pp. 2897–2900. IEEE (2013)
24. Jafari, A., Gandhi, S., Konuru, S.H., Hairston, W.D., Oates, T., Mohsenin, T.: An EEG artifact identification embedded system using ICA and multi-instance learning. In: 2017 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–4. IEEE (2017)
25. Johnson, A.E., Dunkley, N., Mayaud, L., Tsanas, A., Kramer, A.A., Clifford, G.D.: Patient specific predictions in the intensive care unit using a Bayesian ensemble. In: 2012 Computing in Cardiology, pp. 249–252. IEEE (2012)
26. Johnson, A.E., et al.: MIMIC-III, a freely accessible critical care database. Sci. Data **3**, 1–9 (2016)
27. Kandemir, M., Hamprecht, F.A.: Instance label prediction by Dirichlet process multiple instance learning. In: UAI, pp. 380–389 (2014)
28. Keerthi, S., Shevade, S., Bhattacharyya, C., Murthy, K.: Improvements to Platt's SMO algorithm for SVM classifier design. Neural Comput. **13**(3), 637–649 (2001)

29. Kohane, I.: UCI machine learning repository: ICU data set (1994). https://archive.ics.uci.edu/ml/datasets/ICU
30. Kotzias, D., Denil, M., Blunsom, P., de Freitas, N.: Deep multi-instance transfer learning. arXiv preprint arXiv:1411.3128 (2014)
31. Kraus, O.Z., Ba, J.L., Frey, B.J.: Classifying and segmenting microscopy images with deep multiple instance learning. Bioinformatics **32**(12), i52–i59 (2016)
32. Lipton, Z.C., Kale, D.C., Wetzel, R.: Modeling missing data in clinical time series with RNNs. arXiv preprint arXiv:1606.04130 (2016)
33. McMahon, N., Hogg, L., Corfield, A., Exton, A.: Comparison of non-invasive and invasive blood pressure in aeromedical care. Anaesthesia **67**(12), 1343–1347 (2012)
34. Nock, R., Nielsen, F.: A real generalization of discrete AdaBoost. Artif. Intell. **171**(1), 25–41 (2007)
35. Sadeghi, R., Banerjee, T., Romine, W.: Early hospital mortality prediction using vital signals. Smart Health **9**, 265–274 (2018)
36. Salamon, J., McFee, B., Li, P., Bello, J.P.: DCASE 2017 submission: multiple instance learning for sound event detection. In: Detection and Classification of Acoustic Scenes and Events 2017 (2017)
37. Sandberg, J.G., Johnson, L.N., Robia, M., Miller, R.B.: Clinician identified barriers to clinical research. J. Marital Fam. Ther. **28**(1), 61–67 (2002)
38. Silva, I., Moody, G., Scott, D.J., Celi, L.A., Mark, R.G.: Predicting in-hospital mortality of ICU patients: the PhysioNet/computing in cardiology challenge 2012. In: 2012 Computing in Cardiology, pp. 245–248. IEEE (2012)
39. Soleimani, H., Miller, D.J.: Semisupervised, multilabel, multi-instance learning for structured data. Neural Comput. **29**(4), 1053–1102 (2017)
40. Song, H., Rajan, D., Thiagarajan, J.J., Spanias, A.: Attend and diagnose: clinical time series analysis using attention models. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
41. Weidmann, N., Frank, E., Pfahringer, B.: A two-level learning method for generalized multi-instance problems. In: Lavrač, N., Gamberger, D., Blockeel, H., Todorovski, L. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 468–479. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39857-8_42
42. Wu, J., Yu, Y., Huang, C., Yu, K.: Deep multiple instance learning for image classification and auto-annotation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3460–3469 (2015)
43. Xu, X.: Statistical learning in multiple instance problems. Ph.D. thesis, The University of Waikato (2003)
44. Yan, Z., et al.: Multi-instance deep learning: discover discriminative local anatomies for bodypart recognition. IEEE Trans. Med. Imaging **35**(5), 1332–1343 (2016)
45. Zhang, Z.L., Zhang, M.L.: Multi-instance multi-label learning with application to scene classification. In: Advances in Neural Information Processing Systems, pp. 1609–1616 (2007)
46. Zhou, Z.H., Sun, Y.Y., Li, Y.F.: Multi-instance learning by treating instances as non-IID samples. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1249–1256 (2009)
47. Zhu, W., Lou, Q., Vang, Y.S., Xie, X.: Deep multi-instance networks with sparse label assignment for whole mammogram classification. In: Descoteaux, M., Maier-Hein, L., Franz, A., Jannin, P., Collins, D.L., Duchesne, S. (eds.) MICCAI 2017. LNCS, vol. 10435, pp. 603–611. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66179-7_69