On Principal Curve-Based Classifiers and Similarity-Based Selective Sampling in Time-Series

Aref Hakimzadeh  
School of Electrical and  
Computer Engineering  
Shiraz University  
Shiraz, Fars  
Email: sa.hakimzadeh@shirazu.ac.ir

Koorush Ziarati  
School of Electrical and  
Computer Engineering  
Shiraz University  
Shiraz, Fars  
Email: ziarati@shirazu.ac.ir

Mohammad Taheri  
School of Electrical and  
Computer Engineering  
Shiraz University  
Shiraz, Fars  
Email: mtaheri@cse.shirazu.ac.ir

Abstract—Considering the concept of time-dilation, there exist some major issues with recurrent neural Architectures. Any variation in time spans between input data points causes performance attenuation in recurrent neural network architectures. Principal curve-based classifiers have the ability of handling any kind of variation in time spans. In other words, principal curve-based classifiers preserve the relativity of time while neural network architecture violates this property of time. On the other hand, considering the labeling costs and problems in online monitoring devices, there should be an algorithm that finds the data points which knowing their labels will cause in better performance of the classifier. Current selective sampling algorithms have lack of reliability due to the randomness of the proposed algorithms. This paper proposes a classifier and also a deterministic selective sampling algorithm with the same computational steps, both by use of principal curve as their building block in model definition.

I. INTRODUCTION

Label prediction in time-series is equivalent to event recognition. A time-series classifier should be able to recognize events instead of trend. A particular event can happen in different time spans; which means two events with the same label may occur in different time duration. Classifiers that are based on neural networks, e.g., recurrent neural networks (RNN), lack the ability of event recognition in a manipulated time stamp. In other words, RNNs are particular solutions to the event recognition problem with a specified time-span.

The ability to recognize an event is the ability to detect that event in several time steps. Several sources endorse this statement, as Jeffrey M.Zacks says, "From a range of sources comes evidence that people perceive and conceive of events in terms of representations that span time-scales", "People spontaneously segment activity at different timescales in correspondence with a partonomic hierarchy"(Jeffrey M.Zacks, 2001, Psychological bulletin). He later mentioned that, "The theory asserts that people do not perceive event boundaries on only one timescale. Rather, they perceive event boundaries on multiple timescales simultaneously but selectively attend to one timescale in response to instructions or other experimental manipulations" (Jeffrey M.Zacks, 2007, Psychological bulletin).

For instance, if a child is able to count integer numbers, he would also be able to count with time step 2 (counting just even or odd numbers). Also, if a child recognizes a specific word in English, he can also recognize it with a different accent. This problem includes any variation in the time parameter and not just a regular change in the sampling rate. An example in the AI world can be a device which is not real time and sends sampled data in irregular intervals. How this problem affects prediction systems and how we can handle it, are the subjects of current paper.

In many real-world problems, human should do the labeling as a costly task. In this case, an algorithm to decide which data points are more worthy to for labeling can be beneficial. In other words, considering a limited budget for querying label of a data point, it is preferred to query the labels of the points that will cause better performance of the classifier. This problem is known as selective sampling and the budget for querying the label of a data point is known as the query budget. Selective sampling has similarities with anomaly detection problem.

In this paper, recurrent and principal curve-based classifier have been compared. Also, a deterministic selective sampling method is proposed with an ignorable time consumption in comparison with the task of label prediction.

II. LITERATURE REVIEW

In this paper, the literature review is partitioned into methods based on principal curves, recurrent structures in neural networks and selective sampling. In each part several prominent algorithms in that field are reviewed.

A. Principal Curves

Principal curve was defined to be a non-parametric curve that passes through the cloud of data [1]. More specifically, each point in a principal curve is the expected value of the data points that have been projected on that specific point. The original objective function is hard to reach and applicable implementations of principal curves try to find a sub-optimum solution. One of the promising modifications of the original objective function breaks the curve to K-1 linear forms separated by K points, which represent the model of data cloud. Each of the K-1 lines will be optimized considering all data points that have been projected on that specific line-segment. However, the K principal curve accomplished in
fitting the curve to different data distributions, is much time consuming because the algorithm surfs all data points in each step of optimization. To address time consumption problem in principal curves, Constraint Local Principal Curve (CLPC) method proposed an algorithm which excludes a batch of data points from optimization process in each step [3]. CLPC method determines the first and last data points as the first and last principal curve points. Then, in each step, it appends a curve point in between until the total error does not exceed a predefined threshold. Assuming that $P_i$ is the $i_{th}$ point of the extracted principal curve, the first and second data points are considered as $P_1$ and $P_K$ respectively. For appending each new point in the curve, a circle with $P_{n-1}$ as the center and radius equal to $\|P_n - P_{n-1}\|$ is drawn. The expected value of the data points in a narrow bar around drawn circle’s perimeter will be enacted as the new curve point if the error is less than the predefined threshold. After determining each curve point, all the data points that have been projected on the former parts of the curve will not participate in future calculations. As far as we know, there are few papers on classifiers using principal curves. A classifier was proposed in [4] that trains one principal curve for each class of data. Then in the test phase, each sample is assigned to its nearest principal curve and corresponding class. That method is not able to handle sequential data.

B. Recurrent Structures

The pioneer of prediction in time-series includes Recurrent Neural Networks (RNN). These structures are meant to preserve the effect of each data point in later time steps by passing a history unit through the network flow. Recurrent structures have shown promising results in many real-world problems like natural language processing, genomic analysis, image captioning and etc. [5]–[7]. Early efforts included supervised learning on sequential data with simple recurrent structures [8]. However, RNN structures were unable to preserve long term effects and lacked the long-term memory characteristics. To address this problem, Long-Short-Term-Memory (LSTM) cells were proposed to handle both long and short-term effects [9]. Adding forget gates to LSTM cells made them more powerful memory units for predicting sequential data [10]. LSTM cells have excessive parameters which demands numerous data points for training the network. Another proposal for preserving long and short-term effects, and handling vanishing gradients, is Gated Recurrent Units (GRU) by adding reset and forget gates to RNN structures [11]. Compared to LSTM cells, GRU was simpler and has fewer parameters. Another approach to improve the generalization capacity of recurrent structures is dropout [12]. Some other approaches tried to implement a convolutional layer on top of LSTM cells [13] which paved the way to implement attention mechanisms based on recurrent structures [14]. Affected by deep learning, deep RNN was introduced to be able to learn more complex dependencies between data points by adding recurrent layers on top of each other [15].

Non-recurrent models like Gaussian models with latent variables are some other solutions for sequential data analysis [16], [17]. However, recurrent structures have better performance in preserving long-term effects. Temporal Convolutional Network (TCN) is a neural network architecture that is able to preserve long-term dependencies in sequential data [18]. Unlike the former methods, TCN is not a recurrent structure but has achieved promising performance in voice recognition. The main drawback of RNN models in event prediction is their assumption on having data points with equal time steps.

C. Selective Sampling

Selective sampling is about finding data points that knowing their label will cause better training for a specific classifier. A usual approach in selective sampling is finding distribution or decision boundaries which determine the corresponding regions of the classes. Each input data point belongs to each class based on the defined distribution or the distance to the boundaries [19]. Most of the selective sampling algorithms, sample from data points by a Bernoulli distribution extracted from the class probabilities. For example, in perceptron selective sampling, the class probabilities are determined by one or more perceptron(s) to determine the selection threshold for Bernoulli sampling [20]. Winnow and second-order-perceptron selective sampling algorithms use the same query strategy with a different method of probability estimation [21]. Some of the sampling algorithms are known as active learning, where a Gaussian distribution is used to model the class probabilities. The generated output of the Gaussian distribution is used to compute the query probability of the corresponding data point and sampling from Bernoulli distribution with inferred threshold makes the query decision [21]. Considering imbalanced datasets, Cost-Sensitive Online Active Learning (CSOAL) was proposed to assume different probability distributions for different classes of data [22]. Defining an asymmetric update rule for each class of data led to Online Asymmetric Active Learning (OAAL) algorithm [23]. Combining the asymmetric update rule and asymmetric query probability was proposed in [24]. Online Adaptive Asymmetric Active Learning (OA3L) was designed for a binary-class imbalanced data where two distinct Bernoulli distributions with different hyper-parameters make the query decision considering the computed probability and asymmetric update rules updates the boundaries of each class of data differently.

Selective sampling is, somehow, finding the most informative data points in the training phase. Some proposed algorithms try to rank the data points considering an uncertainty evaluation. For example, in [25], data points with highest entropy are found and the most distant samples among them are recommended as queries. Information theoretical approaches form another set of selective sampling methods; but most of them are not designed to sample in online manner. Furthermore, these methods do not preserve the sequential characteristics of the data.

As far as we know, all selective sampling algorithms have a randomness inside the algorithm which causes unreliability of
the sampling algorithm in applications with high sensitivity. Also, randomness leads to more misclassification for imbalanced datasets.

III. PROPOSED ALGORITHM

The following paragraph summarizes the notation used in this paper. In the following sections, it is considered that there are \( n \) training data points \( \{X_1, X_2, \ldots, X_n\} \). The \( i_{th} \) training data point is denoted by an ordered pair \( X_i = (t_i, x_i) \) where \( t_i \) is the time of observing the data point \( x_i \in \mathbb{R}^d \) with \( d \) features. Also, \( P_k \) is the \( k_{th} \) point of the principal curves assuming that the principal curves of all classes form a unit principal curve (by concatenating extracted principal curves). The stated united principal curve is called governing pattern in this paper. It is notable that the \( P_{k,0} \) represents the time variable of the \( k_{th} \) principal curve point \( P_k \). Parameter \( H \) is the history length of the classifier which determines the length of the input stream stored for predicting the next input. In other words, in each step, classifier uses last \( H \) input data points to predict the label of the current input point. Parameter \( C \) is the number of classes in the input data. In addition, \( t_o \) is the time shift (offset) of the input data stream in the matching step. \( X_{i,t} \) is used to show that \( X_i \) is shifted in time with value of \( t_o \). In case of projection on the principal curves, \( f_i(.) \) denotes the projection on the \( \tau^{th} \) principal curve (corresponding to the \( \tau^{th} \) class). Also, \( F(.) \) is the projection on the nearest principal curve, Eq. (1) shows the stated relation.

\[
F(X_{n,t_o}) = f_\beta(X_{n,t_o})
\]

where \( \beta = \arg \min_\tau \|f_\tau(X_{n,t_o}) - X_{n,t_o}\| \) (1)

This section consists of two parts. In the first part, the modeling and optimization steps of the proposed classifier is explained and in the second part, the extra step for selective sampling is explained. Both the proposed classifier and sampling algorithms have common steps for initialization, matching process and update rules. Fig. 1 demonstrates the flowchart of the proposed algorithm.

As shown in Fig. 1, after modeling the input data in initialization step, matching step finds the most similar state between the input data stream and the governing pattern. Label prediction step finds the nearest curve to the matched data and query strategy decides the query considering the similarity history. If query strategy decides to ask the true label, online learning will enhance governing pattern with queried label.

A. Classifying

In this section, the proposed classifier is explained in details. At the initialization step, the input data points are modeled with principal curves and then the governing pattern is extracted. This type of modeling considers the time as an informative feature and also preserves relativity characteristics of time in all subsequent steps. Matching process is the next step to search for a time shift with the minimum projection error. It is also able to handle the variant time spans between the data points. In the next step, the classifier predicts the label of the input data as the corresponding label of the matched input stream. In the last step, the classifier enhances the extracted model considering the true label of the data and the predicted label.

1) Initialization: In initialization step, algorithm is meant to extract a model from an initial labeled dataset. In the proposed method, it is desired to fit several K-principal curves to data, one curve per class. Obviously, each point is presented with the same number of features as the input data. CLPC method is used, in this paper, for extracting the principal curves. In addition, after finding all curve points with CLPC method, the curve points with angle (between two connected line-segments) less than a predefined threshold are removed. This reform in curve points causes removing less informative points and reducing computational cost in matching step.

Fig. 2 shows this step applied on one feature of ”Sleep” dataset. Pale points are the input points and each class is represented by a different color. Therefore, the proposed model has \( K \times (d+1) \) parameters where \( K \) is the number of points in the curve and \( d \) is the feature dimension of data. In this paper, the extracted principal curve is called as governing pattern. To ease calculations in subsequent steps, the governing pattern is shifted in time to place the beginning of the curve in the time origin.

2) Matching: Matching process is meant to find the most similar part of the governing pattern (called state) with the input data sequence. The last \( H \) input data points are shifted in time within the time boundaries of the governing pattern and the most similar state is chosen. To move the input sequence inside the time boundaries of the governing pattern, the input
Fig. 2: Initialization step. Each color defines a class of data and pale points are the original data points.

sequence is shifted to the origin of time.

\[ t_i \leftarrow t_i - t_{n-H+1}, \quad i = n - H + 1, \ldots, n. \]  

In Eq. (2), \( t_i \) is the time variable of the \( i^{th} \) input data point and \( H \) is the length of the input stream and \( t_n \) is the time variable of the last input data point. This step places the input stream in the origin of time. Next, input data sequence will be shifted in time to find the most similar state.

\[
\arg\max_{t_o} \sum_{i=n-H+1}^{n} \text{similarity}(X_{i,t_o}, \phi_{pc})
\]

subject to: \( P_{i,0} \leq t_o \leq P_{i,K}; \)

In Eq. (3), the purpose is to find a \( t_o \) as the beginning of the state in the governing pattern having the maximum similarity with the data points in that state. \( X_{i,t_o} = (t_i + t_o, x_i) \) and \((t_i, x_i)\) is the \( i^{th} \) data point of the input sequence. Also, \( t_n \) is the time upper bound of the input sequence and \( \phi_{pc} \) represents the governing pattern. \( P_{i,0} \) and \( P_{i,K} \) are the time variables of the first and the last points of the governing pattern.

The method of computing similarity value determines the optimization method of the classifier. In the current paper, similarity value is computed through a modified KL-divergence. Eq. (4) implements the similarity function used in Eq. (3).

\[
\arg\min_{t_o} \sum_{i=n-H+1}^{n} \left( F(X_{i,t_o}) \right) \log \left( \frac{\|F(X_{i,t_o}) - X_{i,t_o}\|}{\|F(X_{i,t_o})\|} \right)
\]

subject to: \( P_{i,0} \leq t_o \leq P_{i,K}; \)

where \( P_{i,K} \) is the time value of last principal curve point and \( P_{i,0} \) is the time value of first principal curve point. \( X_{i,t_o} = (t_i + t_o, x_i) \) for each \( t_o \). \( F(X_{i,t_o}) \) is the projection of shifted data point on the nearest principal curve and \( \tau \) is the iterator on the number of classes of data, which is determined by \( C \).

By defining potential areas, we are able to decrease the number of computations by searching potential timespans. We aim to find areas that the input stream is possible to have the highest similarity with the governing pattern. The potential points are defined as Eq. (5) defines.

\[
M = E\{x_{n-H+1}, \ldots, x_n\} \\
\theta = \{P_i \mid \|P_i - M\|/\|P_i\| \leq 0.2\}
\]

where \( x_n \) is the last input data and \( P_i \) is the \( i^{th} \) point of the principal curve. \( M \) is the expected value of the input data stream with \( H \) data points, \( \theta \) is the potential area for matching process. By applying potential areas to the Eq. (3), the objective function changes as follows.

\[
\arg\max_{t_o} \sum_{i=n-H+1}^{n} \text{similarity}(X_{i,t_o}, \phi_{pc})
\]

subject to: \( t_o \in \theta \)

Where \( X_{i,t_o} = (t_i + t_o, x_i) \) and \((t_i, x_i)\) is the \( i^{th} \) data point of the input sequence, \( t_n \) is the time upper bound of the input sequence and \( \phi_{pc} \) represents the governing pattern. This process decreases computations with a considerable ratio.

Fig. 3 demonstrates matching process. In each figure, four principal curves shown in cyan, green, red and magenta colors represent the different classes of the data and blue sequence is the input data sequence. Horizontal axis is the time and vertical axis is one of the features of the initialization data. The correct answer demonstrated in the figure has the most similarity with the principal curves.

As it is obvious, matching process makes no assumption about the time intervals between input data points and time intervals between initialization data points. This ability makes the proposed classifier able to recognize events when time span of the event changes.

3) Label Prediction: As stated before, one principal curve represents each class of the data. After Matching process and finding the most similar state, the last data point of the input sequence will be projected to all principal curves. The nearest principal curve will determine the predicted label. Eq. (7) formulates this process.

\[
\arg\min_{\tau} \|f_{\tau}(X_{n,t_o}) - X_{n,t_o}\|, \quad \tau = 1, \ldots, C
\]

where \( X_{n,t_o} \) is the last input data with optimized time shift and \( f_{\tau}(X_{n,t_o}) \) is the projection of \( X_{n,t_o} \) on \( \tau^{th} \) principal curve and \( C \) is the number of classes.

4) Online Learning: Online learning helps classifier to modify the learned model and leads to better performance in prediction. Online learning is necessary when input data has intrinsic drift. The proposed algorithm draws a vector between the input data point(end of vector) and its projection on the nearest principal curve and moves the two neighboring principal curve points along drawn vector with a predefined step size(\( \alpha \)). If predicted label is the same as the true label, the corresponding segment of the principal curve moves in the
Fig. 3: Example of matching process. This process finds a time shift which causes the most similarity between governing pattern and the input sequence. Red colored points are the last H input data points and other colors define the curve of each class of data. Subfigure (a) is the desired time shift and the most similar situation where Subfigure (b) and Subfigure (c) are two wrong examples of matching process.

(a) Correct Example  
(b) Incorrect Example  
(c) Incorrect Example

positive direction of the error vector. Eq. (8) demonstrates this process.

\[ k = \text{argmin}_i \{ \|P_i - f_r(X_{n,t_i})\| \mid P_{i,0} > t_n \} \]

\[ P_k^- = \alpha \ast (X_{n,t_o} - f_r(X_{n,t_o})) \]

\[ P_{k-1}^- = \alpha \ast (X_{n,t_o} - f_r(X_{n,t_o})) \]

where \( P_i \) is the \( i^{th} \) point of principal curve and \( P_{i,0} \) is the time variable of the corresponding curve point. Hence, \( k \) is the first point of the principal curve which has a bigger time variable compared to \( f_r(X_{n,t_o}) \). \( X_{n,t_o} \) is the input data point with optimized time shift and \( \alpha \) is the step size of the learning process.

If the predicted label is different from the true label, the corresponding segment of the principal curve moves in the opposite direction of the error vector. Eq. (9) demonstrates this process.

\[ k = \text{argmin}_i \{ \|P_i - f_r(X_{n,t_i})\| \mid P_{i,0} > t_n \} \]

\[ P_k^- = \alpha \ast (X_{n,t_o} - f_r(X_{n,t_o})) \]

\[ P_{k-1}^- = \alpha \ast (X_{n,t_o} - f_r(X_{n,t_o})) \]

Eq. (9) has the same parameters as Eq. (8) except that this process moves the curve points against the error vector.

**B. Selective sampling**

In addition to proposed steps, a deterministic selective sampling system can be built by defining a query strategy. After matching process, a similarity value will be produced which is the most similar state between input sequence and governing pattern. By arrival of each data point, the produced similarity value will be added to a similarity history. Hence, by analyzing the variations in similarity values, system is able to decide which data is more worthy to query the label.

1) **Linear Query Strategy**: This strategy is a simple counting problem where the algorithm count the times that the similarity value has been decreased. By normalizing the value by the length of history, the produced value can be compared with a threshold to decide on querying the label.

In Algorithm 1 parameter \( q \) is the return value of the algorithm and it will be initialized as 1. Starting from second value in similarity history, if current similarity value is less than the former value, \( q \) will be incremented by one. This process will continue till the end of similarity history, then, \( q \) will be divided by \( \text{Length(history)} \) for normalization. Algorithm will query the label if \( q > tr \) where \( tr \) is the threshold for query process.

2) **Exponential Query Strategy**: In exponential strategy, the goal is to perform the counting with extra penalty for consecutive decrease in similarity values; such that after each decrease, penalty coefficient will be increase with ratio of 2 and increase in similarity value will decrease penalty coefficient with ratio of \( \frac{1}{2} \). This strategy punishes consecutive decreases and rewards consecutive increases. Hence, in each step, penalty coefficient will be modified considering the similarity values and considering its value, system will be punished and at last, punishment value will be normalized and compared with a threshold to decide query the label.

In Algorithm 2 \( q \) is the return value of query function and
Algorithm 2: Exponential query strategy

Input: similarity\(^{1\times h}\)

Parameters: Threshold

Result: queries label if return value is 1

c = 1;
q = 1;

for \(i \leftarrow 2\) to \(Length(similarity)\) do
\[
q^* = c;
\]
if \(similarity[i] < similarity[i - 1]\) then
\[
c = c \ast 2;\]
else
\[
c = c / 2;\]
end

if \(\frac{q}{2 \cdot Length(similarity)} - 1 \geq \text{Threshold}\) then
\[
\text{return 1};\]
else
\[
\text{return 0};\]
end

\(c\) is penalty coefficient; both these values are initialized as \(c,q = 1\). Starting from second element of the history, \(q\) will be multiplied in \(c\). If current similarity is less than the former one, \(c\) will be multiplied in \(2\), otherwise, in \(\frac{1}{2}\) till the end of the similarity history. For normalizing \(q\) value, it will be divided by \(2 \cdot Length(history) - 1\). Considering that \(tr\) is the threshold for query process, algorithm will query the label if \(q > tr\).

IV. DATA DESCRIPTION

A. WESAD dataset

WESAD is a dataset for wearable stress and affect detection [26]. This dataset contains samples from 15 patients each wearing one wrist-worn device, Empatica E4, and one chest-worn device, RespiBAN. Here we just use data from the wrist-worn device to predict labels. The wrist-worn dataset contains the following features. Blood volume pulse (BVP) with 64 Hz sampling rate, electrodermal activity (EDA) with 4 Hz sampling rate, body temperature with 4 Hz sampling rate, and three-axis acceleration with 32 Hz sampling rate. Labels were also provided with 64 Hz frequency.

Given the different sampling rates, a dataset was constructed from the data corresponding to the least frequent points, which has no missing values. It is notable that no preprocessing was applied to the dataset. In terms of the occurrence of each label, this dataset is a balanced dataset.

B. Sleep dataset

The second dataset contains motion and heart rate features from an apple watch and was labeled according to sleep steps obtained from polysomnography data [27]. Three acceleration axes and heart rate are the features that we want to use to predict the sleep stage label in the dataset. Each subject wearing a device has spent one night in-lab for an 8-hour sleep and the labels were assigned using polysomnography. 5 stages naming wake, N1, N2, N3 and REM are demonstrated in dataset.

One approach for this dataset is the REM/Non-REM prediction which will be highly imbalanced. Other approaches like Wake/Sleep have been tested on this dataset, which are easier tasks. Because none of the features are linearly separable, for predicting REM/None-REM labels analyzing the changes in features over time is the only way to predict labels. Patient 4018081 dataset is used for experiments.

C. PPG dataset

PPG-Dalia dataset is a multimodal dataset which features psychological and motion data[28]. The devices used in PPG-Dalia is the same as the devices used in WESAD dataset. As with WESAD, same features would be chosen to predict performed activities. Sitting, ascending or descending stairs, table soccer, cycling, driving a car, lunch break, walking and working are close to real-life actions which is performed by each subject. This dataset is assumed a balanced dataset. Compared to the previous datasets, PPG has a sensible drift in the data, which will be helpful in testing the online learning process of the proposed algorithm. The drift in data is due to the fact that two data batches from two different patients with similar parameters are used as our dataset for experiments.

D. PAMAP dataset

PAMAP dataset is a physical activity monitoring dataset [29]. This dataset is sampled from 9 subjects via wearable datasets. Each subject wears three inertial measurement units (IMU) and a heart rate sensor. Subjects have some predefined activities to do in sequence, which are lying, sitting, standing, walking, running, cycling, Nordic walking, watching TV, computer work, car driving, ascending stairs, descending stairs, vacuum cleaning, ironing, folding laundry, house cleaning, playing soccer, rope, jumping and transient activities. Each data point has 52 features, a heart rate and 17 features for each IMU sensor. Taking into account the number of occurrences per label, this dataset is considered balanced.

V. EXPERIMENTAL RESULTS

A. Classifier

In this section, the proposed algorithm has been compared with three state-of-the-art algorithms for time-series prediction. Implementation details of these three neural networks are reported in Table I. In addition to accuracy, considering imbalanced datasets, F-score and G-score have been used for evaluating classifiers. F-score is calculated as follows.

\[
F^\beta_\ell = \frac{\text{precision}^\ell \cdot \text{recall}^\ell}{\beta^2 \cdot \text{precision}^\ell + \text{recall}^\ell} \geq (1 + \beta^2) \cdot TP^\ell \tag{10}
\]

\[
F_\beta = \frac{1}{N} \sum_{\ell=1,...,N} C^\ell \cdot F^\ell_\beta \tag{11}
\]
\( F^\ell_\beta \) is the F-score of class \( \ell \) with hyper parameter \( \beta \) and \( C^\ell \) is the importance of class \( \ell \) while \( N \) is the number of classes. \( TP^\ell \), \( FP^\ell \) and \( FN^\ell \) represent true positive, false positive and false negative numbers of the class \( \ell \). In experiments, \( C_1 = 1 \) and \( \beta = 1 \).

G-score is the generalization of Jaccard measure and is calculated as follows.

\[
G^\ell_\beta = \frac{TP^\ell}{TP^\ell + FP^\ell + \beta FN^\ell} \tag{12}
\]

\[
G_\beta = \frac{1}{N} \sum_{\ell=1,...,N} C^\ell . G^\ell_\beta \tag{13}
\]

\( G^\ell_\beta \) is the G-score of class \( \ell \) with hyper parameter \( \beta \) and \( C_\ell \) is the importance of class \( \ell \) while \( N \) is the number of classes. In experiments, \( C_1 = 1 \) and \( \beta = 1 \).

As demonstrated in Eq. (10) and Eq. (12), G-score penalizes false predicted instances more than F-score. In other words, for imbalanced data, F-score reports the average performance while G-score reports the worst-case performance. Hence, both metrics have been reported in result tables.

Sampling rate is the parameter to control the time span of an event. Changing sampling rate allows us to simulate time dilation phenomenon. Simply put, \( \text{samplingrate} = \frac{1}{c} \) means that we only sample first data point out of each \( c \) consecutive data points. For instance, if \( c \) is chosen greater than 1, the sampled event has same time span with less data points which their time difference between each two data points has been multiplied by \( c \). In other words, we intend to test classifiers when data points have been reduced with ratio of \( c \).

1) WESAD dataset: First experiment is on the WESAD dataset. 60% of data points are used as training data and the rest is used as test data. History length is set to 10 for all classifiers and \( \text{samplingrate} = 1 \) in current experiment. All tables report time in seconds.

| TABLE II: WESAD results with sampling rate = 1 |
|-----------------------------------------------|
| Accuracy | F-score | G-score | Time  |
|----------|---------|---------|-------|
| Proposed Alg. | 1.0     | 1.0     | 1.0   | 2434  |
| LSTM      | 0.9926  | 0.9945  | 0.9892| 1853  |
| GRU       | 0.9460  | 0.9563  | 0.9181| 1663  |
| TCN       | 0.8743  | 0.8635  | 0.7626| 1403  |

In next experiment, by choosing \( \text{samplingrate} = \frac{1}{10} \), we aim to test classifiers under reduction of data points. For feeding reduced dataset into the neural networks, we used two settings. In 1\(^{st}\) setting, reduced dataset will be fed with no modification and in 2\(^{nd}\) setting, a linear function between each two data points approximates value of omitted data points and at last number of data points will be identical to the original (not reduced) dataset. History length will be 10 for classifiers.

| TABLE III: WESAD results with sampling rate = \frac{1}{10} |
|--------------------------------------------------------|
| Accuracy | F-score | G-score | Time  |
|----------|---------|---------|-------|
| Proposed Alg. | 0.9969  | 0.9976  | 0.9952| 2524  |
| LSTM-1\(^{st}\)setting | 0.7822  | 0.6194  | 0.5081| 2766  |
| LSTM-2\(^{nd}\)setting | 0.9850  | 0.9703  | 0.9438| 2815  |
| GRU-1\(^{st}\)setting | 0.8189  | 0.8178  | 0.6923| 2174  |
| GRU-2\(^{nd}\)setting | 0.9799  | 0.9793  | 0.9539| 2364  |
| TCN       | 0.5592  | 0.5562  | 0.3924| 2218  |

In last experiment on WESAD dataset, intervals between each two samples is determined by a random number. In other words, \( \text{samplingrate} \) changes randomly in each data point. Here, \( \text{samplingrate} \) is chosen by random sampling from integer number between 1 and 30. Other parameters of the classifiers are the same.

| TABLE IV: WESAD results with random sampling rate between 1 and \frac{1}{30} |
|---------------------------------------------------------------------|
| Accuracy | F-score | G-score | Time  |
|----------|---------|---------|-------|
| Proposed Alg. | 0.9973  | 0.9977  | 0.9958| 2328  |
| LSTM-1\(^{st}\)setting | 0.6436  | 0.5211  | 0.4039| 2362  |
| LSTM-2\(^{nd}\)setting | 0.9803  | 0.9714  | 0.9697| 2395  |
| GRU-1\(^{st}\)setting | 0.8327  | 0.7983  | 0.6692| 2116  |
| GRU-2\(^{nd}\)setting | 0.9588  | 0.9537  | 0.9505| 2294  |
| TCN       | 0.6337  | 0.5935  | 0.4631| 1949  |

As shown in Tab.\[II\] Tab\[III\] and Tab\[IV\], changing sampling rates causes more wrong predictions in the classifier even if the skipped data points were approximated with a linear equation. Whereas the proposed algorithm is slightly affected by changes in the sampling rate. However results show that the proposed algorithm is more time consuming compared to other algorithms.

2) Sleep dataset: The second dataset is sleep stage prediction dataset which is highly imbalanced. 60% of data points is used as training data and the rest is used as test data. History length is 10 and \( \text{samplingrate} = 1 \). Table \[V\] reports the results.

| TABLE VI: Sleep results with sampling rate = 1 |
|------------------------------------------------|
| Accuracy | f-score | G-score | Time  |
|----------|---------|---------|-------|
| Proposed Alg. | 1.0     | 1.0     | 1.0   | 2389  |
| LSTM-1\(^{st}\)setting | 0.9419  | 0.9248  | 0.8629| 1946  |
| LSTM-2\(^{nd}\)setting | 0.9886  | 0.9866  | 0.9736| 1958  |
| GRU-1\(^{st}\)setting | 0.9451  | 0.9331  | 0.8774| 1745  |
| GRU-2\(^{nd}\)setting | 0.9803  | 0.9762  | 0.9539| 1803  |
| TCN       | 0.8258  | 0.8185  | 0.7010| 1493  |
TABLE I: details of implemented neural network architectures used for comparison in experimental results

| Neural network | Description |
|----------------|-------------|
| LSTM           | 80 LSTM cells with a fully connected layer on top of LSTM network. Network is trained 100 epochs on initialization data and 40 epochs on each 10 data points batch for online learning. Optimizer is Adam and loss function is MAE. History length is set to 10. |
| GRU            | 80 GRU cells form recurrent structure. A fully connected layer connects GRU outputs to output. Network is trained 100 epochs on initialization data and 40 epochs on each batch of data during online learning. Optimizer is Adam and loss function is MAE. History length is set to 10. |
| TCN            | A TCN network with 64 filters and dilation of powers of 2. Network is trained 100 epochs on initialization data and 40 on each batch of online learning data. A fully connected layer connects TCN output to one-hot-encoded output. Optimizer is Adam and loss function is MAE. History length is set to 10. |

|   | Accuracy | f-score | G-score | Time |
|---|----------|---------|---------|------|
| Proposed Alg. | **0.9707** | **0.8423** | **0.7289** | 3218 |
| LSTM          | 0.8957   | 0.6883  | 0.5717  | 1694 |
| GRU           | 0.9358   | 0.7851  | 0.6488  | 1531 |
| TCN           | 0.8520   | 0.2105  | 0.1176  | **1204** |

By changing sampling rate to $\frac{1}{2}$, results will be as follows.

**TABLE VII:** Sleep results with sampling rate = $\frac{1}{2}$

|   | Accuracy | f-score | G-score | Time |
|---|----------|---------|---------|------|
| Proposed Alg. | **0.9680** | **0.8849** | **0.7935** | 3315 |
| LSTM-1$^{st}$setting | 0.8876 | 0.1428 | 0.0769 | 1189 |
| LSTM-2$^{nd}$setting | 0.9561 | 0.6551 | 0.4871 | 1326 |
| GRU-1$^{st}$setting | **0.9785** | 0.7104 | 0.5508 | 1173 |
| GRU-2$^{nd}$setting | 0.9618 | 0.7642 | 0.6184 | 1262 |
| TCN            | 0.7280   | 0.1987  | 0.1103  | **1081** |

Results with the sampling rate=$\frac{1}{4}$ is as follows.

**TABLE VIII:** Sleep results with sampling rate = $\frac{1}{4}$

|   | Accuracy | f-score | G-score | Time |
|---|----------|---------|---------|------|
| Proposed Alg. | **0.9778** | **0.8686** | **0.7678** | 3104 |
| LSTM-1$^{st}$setting | 0.8913 | 0.0153 | 0.0077 | 1719 |
| LSTM-2$^{nd}$setting | **0.9802** | 0.4960 | 0.3298 | 1761 |
| GRU-1$^{st}$setting | 0.9770 | 0.2189 | 0.1229 | 1549 |
| GRU-2$^{nd}$setting | 0.9666 | 0.7213 | 0.5641 | 1691 |
| TCN            | 0.3197   | 0.0403  | 0.0206  | **1381** |

In last experiment, in each step, a random number between 1 and 4 determines the sampling rate. The results are demonstrated in Table IX.

**TABLE IX:** Sleep results with random sampling rate between 1 and $\frac{1}{4}$

|   | Accuracy | f-score | G-score | Time |
|---|----------|---------|---------|------|
| Proposed Alg. | **0.9855** | **0.8926** | **0.8061** | 2881 |
| LSTM-1$^{st}$setting | 0.9008 | 0.0176 | 0.0089 | 1541 |
| LSTM-2$^{nd}$setting | 0.9261 | 0.4917 | 0.3489 | 1533 |
| GRU-1$^{st}$setting | 0.9127 | 0.1184 | 0.0629 | 1472 |
| GRU-2$^{nd}$setting | 0.8693 | 0.7152 | 0.5601 | 1362 |
| TCN            | 0.9180   | 0.2130  | 0.1192  | **1194** |

Results of sleep stage prediction dataset confirm that the proposed algorithm can handle any variation in the time difference of the input data points. In other words, the relative concept of time is involved in label prediction process of the proposed algorithm. The significant difference between results of the proposed algorithm and other neural networks is because that sleep stage prediction dataset is highly imbalanced and the proposed algorithm does not loose accuracy on imbalanced data. The proposed algorithm is immune to imbalanced data because that no matter how many of the data points belong to the minority class, the occupied time span of the minority data points will not be violated by the majority class. As shown in results, in some situations, neural network classifiers have higher accuracy but lower F-score and G-score which confirms the stated assertion about the imbalanced data.

3) PPG dataset: The third experiment is on PPG dataset. 60% of data points is used as training data and the rest is used as test data. History length is 10 and sampling rate = 1. Table X reports the results.

**TABLE X:** PPG results with sampling rate = 1

|   | Accuracy | f-score | G-score | Time |
|---|----------|---------|---------|------|
| Proposed Alg. | **0.7737** | **0.5896** | **0.4904** | 5306 |
| LSTM          | 0.8078   | 0.6175  | 0.5152  | 2178 |
| GRU           | **0.8296** | 0.6172  | 0.5255  | 1994 |
| TCN           | 0.7482   | **0.7287** | **0.5784** | **1762** |

In following table, the sampling rate = $\frac{1}{60}$.

**TABLE XI:** PPG results with sampling rate = $\frac{1}{60}$

|   | Accuracy | f-score | G-score | Time |
|---|----------|---------|---------|------|
| Proposed Alg. | 0.7737 | 0.5896 | 0.4904 | 5306 |
| LSTM          | 0.8078   | 0.6175  | 0.5152  | 2178 |
| GRU           | **0.8296** | 0.6172  | 0.5255  | 1994 |
| TCN           | 0.7482   | **0.7287** | **0.5784** | **1762** |
B. Analysis of Online Learning

In this experiment, for analyzing online learning step of the proposed algorithm, we aim to compare classifiers without learning from the test data. In other words, classifier only learns from initialization data points. Sampling rate is equal to 1 in this experiment and other parameters are same as the previous experiments.

### TABLE XIV: Results of the proposed classifier without online learning on test data. sampling rate is set to 1 for this experiment.

| Dataset | Accuracy | f-score | G-score | Time |
|---------|----------|---------|---------|------|
| WESAD   | TPM 0.9642 | 0.9639  | 0.9619  | 2016 |
|         | LSTM 0.9661 | 0.9570  | 0.9505  | 1628 |
|         | GRU 0.9179  | 0.8992  | 0.8879  | 1408 |
|         | TCN 0.8287  | 0.8149  | 0.7238  | 1284 |
| Sleep   | TPM 0.9625  | 0.7945  | 0.7018  | 2817 |
|         | LSTM 0.8186  | 0.3972  | 0.2610  | 1192 |
|         | GRU 0.9103  | 0.4913  | 0.3172  | 1092 |
|         | TCN 0.8017  | 0.1983  | 0.1147  | 1028 |
| PPG     | TPM 0.7162  | 0.5162  | 0.4261  | 4719 |
|         | LSTM 0.7028  | 0.4961  | 0.4285  | 1863 |
|         | GRU 0.7168  | 0.5219  | 0.4381  | 1639 |
|         | TCN 0.6838  | 0.6139  | 0.5261  | 1529 |

Comparing results of Tab.XIV with Tab. II, Tab.VI and Tab.X shows that in all situation, the proposed classifier has closer accuracy, F-score and G-score with the situation that classifiers are rained with the test data in online learning. Highlighted values in Tab.XIV show the least difference between current setting and the experiments of part A. In other words, TPM is more vulnerable to drifts in data. These results confirms the results of Tab. X where on sampling rate 1, the proposed classifier had weaker performance than neural networks; because PPG dataset has sensible drift in data compared to WESAD and Sleep datasets.

To evaluate the effect of step size, $\alpha$, in online learning, in following experiment, $\alpha$ varies from $[0.2, 2]$ by footstep equal to 0.1. Tab. XV reports best and mean accuracy of the proposed classifier while alpha is changing in $[0.2, 2]$.

### TABLE XV: Results of the proposed classifier while $\alpha$ parameter in online learning changes in range $[0.2, 2]$ with footstep 0.1. Values are extracted from 10 runs of the algorithm.

| Dataset | Accuracy | f-score | G-score |
|---------|----------|---------|---------|
|         | best     | mean    | best    | mean   |
| WESAD   | 1.0      | 0.9832  | 1.0     | 0.9515 |
|         | 0.9810   | 0.9263  | 0.8493  | 0.7318 |
| Sleep   | 0.7759   | 0.6818  | 0.5886  | 0.4962 |

As shown in Tab.XV considering the difference between mean value and best value of the metrics, online learning in the proposed classifier is highly dependent to $\alpha$ parameter. Nevertheless, online learning step in the proposed algorithm has not the expected functionality considering its time consumption.
C. Selective sampling

Following experiment will compare the proposed algorithm for selective sampling with winnow selective sampling and OA3L algorithm. The classifier used for all sampling algorithms in following experiment is the proposed classifier. All experiments will simulate an online sampler device where there exists an initialization data with all corresponding labels and unlabeled test data points which will be fed to the classifier in an online manner. In arrival of each data point, classifier predicts the label and query strategy decides the query of the corresponding label.

Each reported value in result tables is mean value ± variance of data from 10 runs of each query strategy. Table XVI explains the metrics used in tables.

| metrics   | Description                                      |
|-----------|--------------------------------------------------|
| apt query | Queried labels that predicted label is different from true label |
| inapt query | Queried labels that predicted label is equal to true label |
| ratio     | ratio of good queries to total queries           |
|           | \[\frac{\text{aptQuery}}{\text{aptQuery} + \text{inaptQuery}}\] |

TABLE XVI: used metrics

Implementation details are as follows. For winnow algorithm, \(\eta = 2\) and \(b = 2\). In OA3L algorithm, \(\delta_+ = 0.2\) and \(\delta_- = 0.2\) for PPG, PAMAP and WESAD datasets; and considering that sleep dataset is imbalanced, \(\delta_+ = 0.2\) and \(\delta_- = 0.4\).

1) Quality of queried labels: In first experiment, the proposed algorithm with linear and exponential strategies will be compared with OA3L and winnow algorithms. 10% of data points are used as initialization data, where all corresponding labels are provided, and 90% of data is used as test data where no labels are provided and there exists a budget equal to 30 queries for each 100 input data points. Table XVII contains the results.

In second experiment, the budget for query is equal to 90 for every 1000 input data points. Table XVIII contains the results.

Results of Tab. XVII and Tab. XVIII show that the proposed algorithm has more query requests in situations that the predicted label is not the same true label, which we call an apt query. Also, results show that the proposed algorithms have less variance in all situations compared to two other sampling algorithms. This low variance of the proposed algorithm is caused by removing the randomness from the query strategy which accelerates reliability of the sampling strategy in intensive risk environments. Even in situations that other algorithms had better ratio than one of the proposed algorithms, the lower variance belongs to the proposed algorithm. Results of Tab. XVII are summarized in Figs. 4 and 5.

2) Accuracy tests: In these experiments, effect of different query algorithms on accuracy of the classifier will be evaluated. In first experiment, 50% of data points with corresponding labels are used as initialization data and the rest of data points are used as test data with query budget equal to 90 for each 1000 data points. Initialization accuracy reports the accuracy of the classifier on initialization data and test accuracy reports the accuracy of classifier on test data while query budget is limited. Table XIX reports the results of this experiment.

In second accuracy test, accuracy of classifier will be compared in two situations. In first situation, all data points will have their corresponding labels which is reported as full budget in Table XX. In second situation, 50% of data points with their labels are used as initialization data and other 50% are used as test data with no labels provided unless 90 labels can be queried per each 1000 data points.

Confirming the results on quality of labels, Tab. XIX shows that the apt query requests causes better performance in classifier. Comparing the exponential and linear proposed algorithms, exponential strategy has the upper hand in three datasets while linear strategy has better performance in one dataset. Results of Tab. XX show that the proposed algorithms have caused better accuracy compared to OA3L and winnow selective sampling. However, results of Tab. XX show that linear strategy has better performance in three datasets and the exponential strategy is better in one dataset. Also, in Tab. XIX and Tab. XX, the proposed algorithm with linear and exponential strategies have lower variance compared to OA3L and winnow.

VI. Discussion

Results of the experiments show that neural networks are a specific solution for time series label prediction. Neural networks do not consider relativity of time, thus, they are not general optimizer for pattern recognition in time series. Also, the proposed selective sampling algorithm solves selective sampling problem from a different point of view which makes if more reliable and stable by omitting randomness from the
TABLE XVII: Results on quality of query attempts with budget 30

| Dataset       | PPG         | WESAD       | Sleep       | PAMAP       |
|---------------|-------------|-------------|-------------|-------------|
| Proposed algorithm Linear | apt query | 5.46 ± 0.57 | 5.92 ± 0.73 | 7.79 ± 1.05 | 7.91 ± 0.49 |
|                | inapt query| 13.84 ± 1.14| 18.81 ± 1.97| 16.78 ± 1.54| 15.05 ± 1.61|
|                | ratio      | 0.28 ± 0.03 | 0.23 ± 0.07 | 0.32 ± 0.04 | **0.34 ± 0.02** |
| Proposed algorithm Exponential | apt query | 5.80 ± 0.75 | 6.23 ± 0.64 | 7.14 ± 0.58 | 6.01 ± 0.54 |
|                | inapt query| 14.20 ± 1.32| 18.65 ± 2.10| 14.57 ± 1.17| 16.28 ± 2.21 |
|                | ratio      | **0.29 ± 0.04** | **0.25 ± 0.07** | **0.33 ± 0.03** | **0.27 ± 0.05** |

| winnow algorithm | apt query | 4.14 ± 1.64 | 3.46 ± 1.96 | 7.46 ± 2.64 | 6.14 ± 2.46 |
|                  | inapt query| 13.28 ± 1.83| 16.28 ± 2.98| 16.18 ± 2.63| 13.94 ± 1.92 |
|                  | ratio      | 0.23 ± 0.07 | 0.17 ± 0.08 | 0.32 ± 0.07 | 0.31 ± 0.09 |

| OA3L algorithm  | apt query | 6.8 ± 4.30 | 5.70 ± 4.58 | 4.80 ± 2.45 | 6.58 ± 2.83 |
|                 | inapt query| 18.40 ± 3.44| 19.30 ± 4.10| 17.20 ± 2.82| 17.79 ± 3.06 |
|                 | ratio      | 0.26 ± 0.15 | 0.23 ± 0.17 | 0.26 ± 0.22 | 0.27 ± 0.10 |

TABLE XVIII: Results on quality of query attempts with budget 90

| Dataset       | PPG         | WESAD       | Sleep       | PAMAP       |
|---------------|-------------|-------------|-------------|-------------|
| Proposed algorithm Linear | apt query | 17.41 ± 1.01 | 23.01 ± 1.14 | 27.35 ± 2.67 | 24.31 ± 2.07 |
|                | inapt query| 32.29 ± 1.36| 34.85 ± 1.92| 49.73 ± 2.83| 49.12 ± 2.79 |
|                | ratio      | 0.35 ± 0.06 | 0.39 ± 0.06 | 0.35 ± 0.07 | 0.33 ± 0.06 |
| Proposed algorithm Exponential | apt query | 19.41 ± 1.05 | 23.73 ± 1.45 | 26.65 ± 2.07 | 27.35 ± 1.72 |
|                | inapt query| 30.52 ± 1.82| 32.05 ± 1.76| 47.03 ± 2.74| 48.28 ± 3.12 |
|                | ratio      | **0.38 ± 0.07** | **0.42 ± 0.03** | **0.36 ± 0.05** | **0.36 ± 0.06** |
| winnow algorithm | apt query | 16.03 ± 2.30 | 20.70 ± 2.24 | 26.15 ± 2.73 | 29.15 ± 2.43 |
|                  | inapt query| 37.50 ± 2.83| 34.05 ± 3.06| 56.83 ± 3.41| 51.83 ± 3.01 |
|                  | ratio      | 0.29 ± 0.07 | 0.38 ± 0.07 | 0.32 ± 0.09 | **0.36 ± 0.08** |
| OA3L algorithm  | apt query | 16.42 ± 6.28 | 27.92 ± 7.28 | 22.08 ± 4.70 | 18.15 ± 4.28 |
|                 | inapt query| 39.80 ± 5.63| 54.06 ± 8.40| 59.41 ± 5.16| 58.37 ± 4.71 |
|                 | ratio      | 0.29 ± 0.15 | 0.34 ± 0.19 | 0.25 ± 0.17 | 0.23 ± 0.16 |

Fig. 6: Mean and Variance ratio of query quality with budget 30

algorithm and making decision considering similarity of input stream and governing pattern.

Considering concept of “time dilation” derived from the "General Relativity" of Einstein, time should be assumed as a milieu of dilating patterns. Because that a neural network classifier has a static architecture, relativity can not be implemented with current architectures. In a classifier which predicts labels for time series, the behavior of time should be justified in "General Relativity" theorem. Hence, any pattern should be predictable when “time dilation” occurs. Despite the implementation details of the proposed algorithm, the stated key difference between the proposed algorithm and neural networks (Connectionist Architecture) is what this paper aims to demonstrate.

VII. CONCLUSION

This paper proposes a classifier with particular characteristics for time-series label prediction and also a deterministic selective sampling algorithm which uses the same computational steps as the classifier. The proposed classifier is able to handle variant time spans between data points. Furthermore, the proposed algorithm is able to handle class imbalanced data with no further constraint or computation. Comparing the results of the classifiers, the proposed classifier outperform other time-series prediction classifiers in case of varying sampling rate. Considering that all neural networks are bounded to the initial architecture design, they can be assumed as a special case in time series prediction.

The proposed selective sampling algorithm has better performance in finding the data points that have been predicted in wrong class. In this paper, instead defining a decision line for query strategy, a similarity-based algorithm is proposed to have both classifier and the query strategy working on same principles. Also, the lack of randomness in the proposed selective sampling algorithm, and thus lower variance in reported results, improves the reliability of algorithm in intensive risk applications.
TABLE XIX: Accuracy results on Initialization data with full budget and test data with limited budget

| Dataset   | PPG     | WESAD   | Sleep   | PAMAP   |
|-----------|---------|---------|---------|---------|
| Proposed algorithm Linear |         |         |         |         |
| Init. Acc. | 63.50 ± 1.76 | 93.36 ± 1.87 | 73.30 ± 2.01 | 78.25 ± 1.97 |
| Test Acc. progress | -3.75 | +0.80 | +1.56 | +1.73 |
| Proposed algorithm Exponential |         |         |         |         |
| Init. Acc. | 64.52 ± 1.93 | 91.64 ± 1.99 | 71.78 ± 2.16 | 76.47 ± 1.35 |
| Test Acc. progress | +3.95 | +1.51 | +1.17 | +3.36 |
| winnow algorithm |         |         |         |         |
| Init. Acc. | 64.2 ± 2.48 | 95.22 ± 2.88 | 74.20 ± 3.13 | 77.85 ± 2.93 |
| Test Acc. progress | -1.55 | -1.98 | +0.09 | -4.79 |
| OA3L algorithm |         |         |         |         |
| Init. Acc. | 61.37 ± 4.87 | 91.92 ± 6.80 | 75.94 ± 5.17 | 79.64 ± 5.10 |
| Test Acc. progress | -5.87 | -1.81 | -4.90 | -7.47 |

TABLE XX: Accuracy results on comparing full budget and limited budget on test data

| Dataset   | PPG     | WESAD   | Sleep   | PAMAP   |
|-----------|---------|---------|---------|---------|
| Proposed algorithm Linear |         |         |         |         |
| limited budget progress | 67.24 ± 1.83 | 94.16 ± 1.36 | 74.86 ± 1.88 | 79.98 ± 2.53 |
| Proposed algorithm Exponential |         |         |         |         |
| limited budget progress | 68.47 ± 1.39 | 93.15 ± 1.89 | 72.95 ± 2.21 | 79.83 ± 2.61 |
| winnow algorithm |         |         |         |         |
| limited budget progress | 62.65 ± 2.06 | 93.29 ± 3.31 | 74.29 ± 3.39 | 73.06 ± 2.83 |
| OA3L algorithm |         |         |         |         |
| limited budget progress | 55.5 ± 5.91 | 90.11 ± 7.05 | 71.04 ± 5.58 | 72.17 ± 4.92 |

Analysis of online learning step shows that this step is not functioning as expected and neural networks are better learners in case of online learning. Also, online learning in the proposed algorithm is consumes more computational power than neural networks and thus more processing time. In general, results of the experiments show that the proposed algorithm has higher processing time in all situations. Furthermore, when a new feature is added to the input data, the processing time of the proposed algorithm increases faster than neural networks.

This paper proposes an algorithms that preserves all characteristics of the time and thus is able to handle any variant time spans between data points. Also, Considering the differences between Empiricism and the Piagetian oeuvre, besides counting wrong and correct predictions, the Matching step creates an opportunity to investigate why the label is predicted in this way. This point of view has weaknesses but paves the way which features new opportunities in case of time series prediction. The proposed sampling strategy is an example of the potential of this point of view.

REFERENCES

[1] Trevor Hastie and Werner Stuetzle. “Principal curves”. In: Journal of the American Statistical Association 84.406 (1989), pp. 502–516.

[2] Balázs Kégl, Adam Krzyzak, Tamás Linder, et al. “A polygonal line algorithm for constructing principal curves”. In: Advances in Neural Information Processing Systems. 1999, pp. 501–507.

[3] Dewang Chen, Jiateng Yin, Shiyings Yang, et al. “Constraint local principal curve: Concept, algorithms and applications”. In: Journal of Computational and Applied Mathematics 298 (2016), pp. 222–235.

[4] Yunsong Qi, Lei Pan, and Huaijiang Sun. “Microarray data classification based on principal curves”. In: 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery. Vol. 5. IEEE. 2010, pp. 2199–2202.

[5] Michael Auli, Michel Galley, Chris Quirk, et al. “Joint language and translation modeling with recurrent neural networks”. In: (2013).

[6] Zachary C Lipton, David C Kale, Charles Elkan, et al. “Learning to diagnose with LSTM recurrent neural networks”. In: arXiv preprint arXiv:1511.03677 (2015).

[7] Oriol Vinyals, Alexander Toshev, Samy Bengio, et al. “Show and tell: A neural image caption generator”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015, pp. 3156–3164.

[8] Jeffrey L Elman. “Finding structure in time”. In: Cognitive science 14.2 (1990), pp. 179–211.
Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: Neural computation 9.8 (1997), pp. 1735–1780.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. “Learning to forget: Continual prediction with LSTM”. In: (1999).

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: arXiv preprint arXiv:1406.1078 (2014).

Vu Pham, Théodore Bluche, Christopher Kermorvant, et al. “Dropout improves recurrent neural networks for handwriting recognition”. In: 2014 14th International Conference on Frontiers in Handwriting Recognition. IEEE, 2014, pp. 285–290.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, et al. “LSTM fully convolutional networks for time series classification”. In: IEEE access 6 (2017), pp. 1662–1669.

Fazle Karim, Somshubra Majumdar, and Houshang Darabi. “Insights into LSTM fully convolutional networks for time series classification”. In: IEEE Access 7 (2019), pp. 67718–67725.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, et al. “How to construct deep recurrent neural networks”. In: arXiv preprint arXiv:1312.6026 (2013).

Marzyeh Ghassemi, Marco AF Pimentel, Tristan Naumann, et al. “A multivariate timeseries modeling approach to severity of illness assessment and forecasting in ICU with sparse, heterogeneous clinical data”. In: Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

Ioan Stanculescu, Christopher KI Williams, and Yvonne Freer. “A Hierarchical Switching Linear Dynamical System Applied to the Detection of Sepsis in Neonatal Condition Monitoring.” In: UAI. 2014, pp. 752–761.

Aaron van den Oord, Sander Dieleman, Heiga Zen, et al. “Wavenet: A generative model for raw audio”. In: arXiv preprint arXiv:1609.03499 (2016).

Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. “Learning probabilistic linear-threshold classifiers via selective sampling”. In: Learning Theory and Kernel Machines. Springer, 2003, pp. 373–387.

Nicolo Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. “Worst-case analysis of selective sampling for linear classification”. In: Journal of Machine Learning Research 7.Jul (2006), pp. 1205–1230.

Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. “Get another label? improving data quality and data mining using multiple, noisy labelers”. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008, pp. 614–622.

Peilin Zhao and Steven CH Hoi. “Cost-sensitive online active learning with application to malicious URL detection”. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. 2013, pp. 919–927.

Xiaoxuan Zhang, Tianbao Yang, and Padmini Srinivasan. “Online asymmetric active learning with imbalanced data”. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016, pp. 2055–2064.

Yifan Zhang, Peilin Zhao, Jie Zhang Cao, et al. “Online adaptive asymmetric active learning for budgeted imbalanced data”. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018, pp. 2768–2777.

Asim Smailagic, Pedro Costa, Alex Gaudio, et al. “O-MedAL: Online active deep learning for medical image analysis”. In: Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 10.4 (2020), e1353.

Olivia Walch, Yitong Huang, Daniel Forger, et al. “Sleep stage prediction with raw acceleration and photoplethysmography heart rate data derived from a consumer wearable device”. In: Sleep 42.12 (2019), zsz180.

A. Reiss, I. Indlekofer, Philip Schmidt, et al. “Deep PPG: Large-Scale Heart Rate Estimation with Convolutional Neural Networks”. In: Sensors 19.14 (2019), p. 3079.

A. Reiss and D. Stricker. “Introducing a new benchmarked dataset for activity monitoring”. In: 2012 16th international symposium on wearable computers. IEEE, 2012, pp. 108–109.

Albert Einstein. “Zusammenfassung der Mitteilung ‘Erklärung der Perihelbewegung des Merkur aus der allgemeinen Relativitätstheorie.’” In: Königlich Preußische Akademie der Wissenschaften (Berlin). Sitzungsberichte 47 (1915), p. 803.