Incremental and Adaptive Feature Exploration over Time Series Stream

Jingwei Zuo[1,2], Karine Zeitouni[1,2], and Yehia Taher[1,2]

[1] DAVID Lab. University of Versailles, France
[2] University of Paris-Saclay, France
{jingwei.zuo, karine.zeitouni, yehia.taher}@uvsq.fr

Abstract. Over past years, various attempts have been made at analysing Time Series (TS) which has been raising great interest of Data Mining community due to its special data format and broad application scenarios. An important aspect in TS analysis is Time Series Classification (TSC), which has been applied in medical diagnosis, human activity recognition, industrial troubleshooting, etc. Typically, all TSC work trains a stable model from an off-line TS dataset, without considering potential Concept Drift in streaming context. Domains like healthcare look to enrich the database gradually with more medical cases, or in astronomy, with human's growing knowledge about the universe, the theoretical basis for labelling data will change. The techniques applied in a stable TS dataset are then not adaptable in such dynamic scenarios, that said streaming context. Classical data stream analysis are biased towards vector or row data, where each attribute is independent to train an adaptive learning model, but rarely considers Time Series as a stream instance. Processing such type of data, requires combining techniques in both communities of Time Series (TS) and Data Streams. To this end, we conduct the first attempt to extract the adaptive features from Time Series Stream, which is capable of emphasizing the trade-off between past and present information by different attention mechanisms, and taking into account the Concept Drift in streaming context.

1 Introduction

Time Series (*TS*) is a sequence of real-valued data, which can be collected from various sources, such as ECG data in medicine, IoT data in smart cities, light-curves in astronomy, GPS or accelerometer data in activity recognition, etc. Time Series Classification (*TSC*) is intended to predict the label of a newly input TS instance by extracting the knowledge from collected data. Various TSC approaches have been proposed by researchers in recent years which are suitable for different contexts along with dissimilar TS features. One Nearest Neighbor (*1-NN*) classifier for whole series similarity measure is a typical baseline of TSC research, which is usually combined with various distance measures [7,8,16,22]. Other than considering the global feature of entire series, summary statistic features (e.g., mean, deviation, slope, etc.) can be extracted from every sub-series to build diverse ensemble classifiers [3,20,6]. With the emergence

of TS dimensionality reduction techniques (e.g., PAA [12], SAX [13], etc.), TS instances can be represented by high-dimensional vectors so that various techniques from classic data analysis can be adapted into TS context. For instance, in case that motifs or frequent patterns are what characterize a given class, the dictionary based approaches [14,24,23] borrowed from Text Mining and Information Retrieval community can be adopted. As for the scenario that the occurrence of specific sub-series determines a class, TS can then be represented by such shape-based features, namely Shapelet [27]. Various Shapelet-based approaches have been proposed to optimize both the accuracy [17,9] and the efficiency [21,4] of the classification. Another remarkable attempt [18,19] adopting ensemble approaches on several TS representations (e.g., time, auto-correlation, power spectrum, Shapelet domain, etc.) shows a superior accuracy to one single representation classifiers, where TS features are from different representation domains, and can not be presented in a single form.

The optimization of TS feature extraction and model construction process allows us to strive for a low prediction error, and stay as close as possible to Time Series' nature Concept [1], which refers to the target variable that the learning model is trying to predict. Most *TSC* approaches are biased towards learning from an off-line Time Series dataset, with the assumption that data instances are independently and identically distributed (i.i.d) within a particular concept, but rarely consider the streaming context, where a gradual change of the concept happens along with the input of TS stream, that says *Concept Drift*. For instance, the most accurate ensemble classifiers [18,19] are not good options in streaming context due to their complex architecture. Lazy classifiers on Time Series such as Nearest Neighbor (1-NN) [25] and dictionary based approaches [14] are applicable for streaming context. However, every input instance will be considered to adjust the inner concept, which requires potentially a large buffer space and will bring a huge computation cost. Recent Deep Neural Network (DNN) approaches [10,11,26] on TSC are capable of tuning the model incrementally, but stay always in an awkward position for the lack of explainability, which is required by domains like healthcare where questions of accountability and transparency are particularly important.

Shapelet, as a shape-based feature in TS, which is widely adopted by the community for its reliability and interpretability, provides a possibility to fulfil the aforementioned requirements. With an advanced work in [29], the explainability of Shapelet Extraction process is ensured even to non-expert, which offers us an option to further explore the streaming context conserving the current goodness. To build the gap between Time Series Classification and data streams processing, in this paper, we propose a TS stream learning algorithm, where TS features and models can be updated with consideration of Concept Drift.

The main contributions of this paper are as follows:

1. **Evaluation Procedure:** We propose an evaluation strategy for TS learning model, which is capable of not only accelerating the incremental extraction of Shapelet in stable-concept dataset, but also detecting Time Series' Concept Drift in streaming context.

2. **Explainability:** The algorithm is capable of presenting the Shapelet Evolution in streaming context with concept drift, which gives a possibility to supervise the system and back up the historical features.
3. **Scalability:** The algorithm conserves the scalability of TS proposition [29] in streaming context. The computation of Shapelet Extraction is always parallelizable in a remote Spark cluster with a minimum communication cost between distributed nodes.

The rest of this paper is organized as follows. In Section 2, we review the background and state-of-the-art approaches which are useful for our research problems. Then, we present our scalable engine in both context of stable concept and Time Series Stream with drifting concept in Section 3. Finally, we give our conclusions and perspectives for future work in Section 4.

2 Background

2.1 Definitions and Notations

We start with defining the notions used in the paper:

Definition 1: *A Time Series* T is a sequence of real-valued numbers T=(t_1, t_2, ..., t_i, ..., t_n), where n is the length of T.

Definition 2: *Time Series Stream* S_{TS} is a continuous input data stream where each instance is a Time Series: S_{TS}=(T_1, T_2, ..., T_N, ...). Notice that N increases with each new time-tick.

Definition 3: *Time Series Chunk* $C_{t,w}$ is a Time Series micro-batch at time-tick t with window size w in S_{TS}: $C_{t,w}$=(T_{t-w+1}, T_{t-w+2}, ..., T_t).

Definition 4: *Cached Dataset* D_t is a set of time series T_i, and class label c_i, collected after time tick t. Formally, D_t= $\langle T_t, c_{j_t}\rangle$, $\langle T_{t+1}, c_{j_{t+1}}\rangle$, ..., $\langle T_N, c_{j_N}\rangle$, where N is the time-tick of the most recent input instance. $C = c_1, c_2, ..., c_{|C|}$ is a collection of class labels, where $|C|$ denotes the number of labels.

Definition 5: *A subsequence* $T_{i,m}$ of Time Series T is a continuous subset of values from T of length m starting from position i. $T_{i,m} = (t_i, t_{i+1}, ..., t_{i+m-1})$, where $i \in [0, n-m+1]$.

Definition 6: *Shapelet* $\hat{s}$ is a time series subsequence which is particularly representative of a class. As such, it shows a shape which can distinguish one class from the others.

Definition 7: *Euclidean Distance(ED)* between two time series $T_{x,m}$, $T_{y,m}$ is expressed as follows:

$$ED_{x,y} = \sqrt{\sum_{i=1}^{m}(t_{x,i} - t_{y,i})^2} \tag{1}$$

Definition 8: *Distance Profile* DP_i is a vector which stores the Euclidean Distance between a given subsequence/query $T_{i,m}$ and every subsequences $T'_{j,m}$ of a target Time Series T'. Formally, $DP^m_{i,j} = dist(T_{i,m}, T'_{j,m}), \forall j \in [0, n'-m+1]$.

We assume that the distance is measured by Euclidean Distance between z-normalized subsequences [29]. Authors in [28] propose *MASS* which is considered as the fastest distance measure between two Time Series. *MASS* computes Distance Profile based on Fast Fourier Transform (FFT), which requires only $\mathcal{O}(nlogn)$ time and is independent of query's length, other than $\mathcal{O}(nm^2)$ with several orders of magnitude higher.

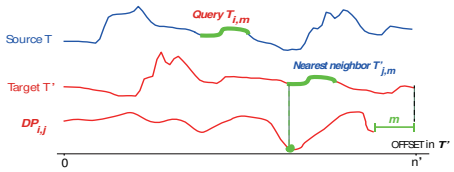

Fig. 1: Distance Profile between Query $T_{i,m}$ and target time series T', where n' is the length of T'. Obviously, $DP_{i,j}$ can be considered as a meta TS annotating target T'

Definition 9: *Matrix Profile* MP is a vector of distance between subsequence $T_{i,m}$ in source T and its nearest neighbor $T'_{j,m}$ in target T'. Formally, $MP_i^m = min(DP_i^m)$, where $i \in [0, n-m+1]$.

Unlike the distance profile, the matrix profile is a meta TS annotating the source time series. The highest point on MP corresponds to the TS discord, the lowest points correspond to the position of a query which has a similar matching in target TS.

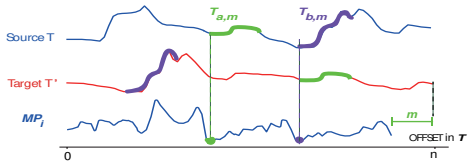

Fig. 2: Matrix Profile between Source time series T and Target time series T', where n is the length of T. Intuitively, MP_i shares the same offset as source T

Definition 10: *Representative Profile* RP_T^C is a vector of representative power of subsequences in T for class C: $RP_T^C = (RP_{T_1}^C, RP_{T_2}^C, ..., RP_{T_i}^C, ...RP_{T_{n-m+1}}^C)$

The representative power of subsequence T_i in class C is defined by the formula:

$$RP_{T_i}^C = avg(MP_{T_i,T_j}) \tag{2}$$

where $T_j \in D_t^C$. Intuitively, $RP_{T_i}^C$ is a normalized distance between T_i and global TS instance cluster of class C, it represents the relevance between the subsequence T_i (i.e., the candidate Shapelet) and the class. As shown in **Fig.** 3 (b)(d), a threshold can be set to show the starting index area in T, where the subsequences are representative for class C.

Definition 11: *Discriminative Profile* $DiscmP_T$ is a vector of discriminative power of subsequences in T:

$$DiscmP_T = RP_T^{NonC} - RP_T^C \tag{3}$$

The discriminative power of T_i in dataset shows the difference of representative power of candidate Shapelet from its own class to the others *(OVA, one-vs-all)*. Intuitively, Discriminative Profile can give a global view of the important patterns' positions over a Time Series. As shown in **Fig.** 3 (f)(g), the highest point in the profile shows the position of the sub-series in T, which has the biggest skewing of relevance between class C and other classes. Through setting a power threshold, the discriminative patterns, that says the candidate Shapelets, can be visually identified in T by their discriminative power.

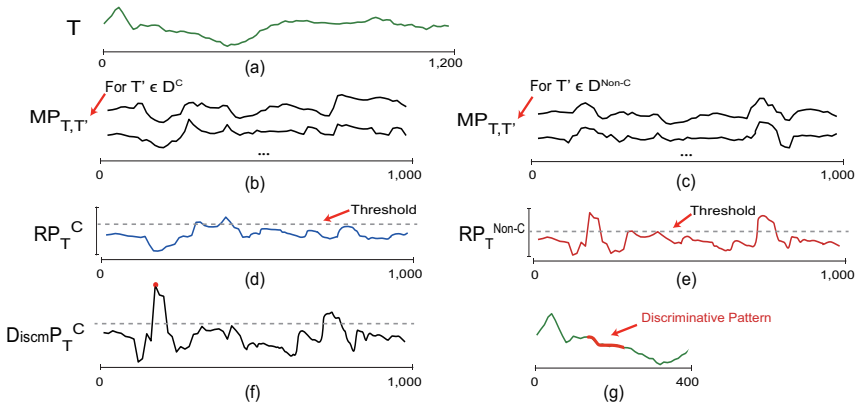

Fig. 3: *(a)* Time Series T with class C. *(b,c)* Matrix Profile set for T with TS instance in different class. *(d,e)* Representative Profile of T in different class. A threshold can determine the representative area of T. *(f)* Discriminative Profile of T in dataset. The highest point in $DiscmP_T^C$ represents the most discriminative pattern's position in T.

2.2 A Brief Review of the SMAP algorithm for TSC

Matrix profile [28] provides a meta-data which facilitates the representation of a complex correlation between two time series. Authors in [29] propose *SMAP*, a scalable algorithm for Shapelet Extraction on Matrix Profile, which ensures meanwhile the parallelism in Spark cluster, and the explainability of extraction process to a non-expert.

As shown in *Algorithm* 1, considering time series as the smallest processing unit between Spark nodes, *SMAP* firstly broadcasts the dataset to distributed nodes in order to reduce the communication cost from repetitive access of common data. Then, each cluster partition shares the computing tasks for a set of TS, and extracts the most discriminative sub-series of various length in each processing unit. Each extracted sub-series can be considered as a candidate Shapelet, which is assigned a distance threshold defined by its representative power in its own class. The threshold can determine the inclusion between the candidate Shapelet and a TS. A strategy to check if T contains a candidate Shapelet $\hat{s}$ can be defined as the following:

$$Inclusion(T, \hat{s}) = \begin{cases} true, & if\ dist(T, \hat{s}) \leq \hat{s}.Thresh \\ false, & otherwise \end{cases} \tag{4}$$

Each TS unit is assigned an unique Hash ID to reduce the volume of transferred data between nodes. The TS ID, as well as the discriminative power and threshold distance of its contained candidate Shapelets, will be output as the computing results of the partition. Finally, a single aggregation process between nodes is required to obtain the Shapelet result of different classes. The whole extraction process can be visualized with a strong explainability, and generates high interpretable results.

Algorithm 1: SMAP (Shapelet extraction on MAtrix Profile on Spark)

Input: Dataset D, **classSet** $\hat{C}$, k
Output: $\hat{S}$

$l_{min} \leftarrow 2$, $l_{max} \leftarrow getMinLen(D)$, $DiscmP \leftarrow []$, $Thresh \leftarrow []$, $\hat{S} \leftarrow \emptyset$
D.broadcast(); //each TS has an unique ID
MapPartition *(Set of* $\langle ID, T\rangle : T_{set}$*)*
  for $\langle ID, T\rangle \in T_{set}$ **do**
    for $m \leftarrow l_{min}$ *to* l_{max} **do**
      $DiscmP[m], Thresh[m] \leftarrow computeDiscmP(T, D, m)$
      $DiscmP[m] \leftarrow DiscmP[m] * \sqrt{1/l}$
    $DiscmP, Thresh \leftarrow pruning(DiscmP, Thresh)$
    emit$(ID, DiscmP, Thresh)$
MapAggregation $(class, (ID, DiscmP, Thresh))$
  for $c \in \hat{C}$ **do**
    $\hat{S}' \leftarrow getTopk(DiscmP[c], Thresh[c], k)$
    $\hat{S} \leftarrow \hat{S} \cup \hat{S}'$
return $\hat{S}$

To sum up this section, Discriminative Profile provides a possibility to extract the interpretable patterns in an explainable manner. The adoption of *MASS* essentially accelerates the extraction process compared to using pruning techniques based on brute force approach[27]. *SMAP* provides a parallel processing mechanism to conduct the extraction in a minimum communication cost on Spark cluster. In the next section, we will show an advanced algorithm which, when applied on Spark cluster, is capable of updating Shapelet results by adopting an incremental model in dynamic source context.

3 Algorithms

In this section, we start by studying the incrementality of SMAP, which is a necessary condition for learning in streaming context. Then we propose the evaluation strategies to accelerate incremental learning process and adapting it to streaming context considering Concept Drift.

3.1 Incremental SMAP (ISMAP)

Typically, a non-incremental algorithm requires to re-pass the existing dataset and conduct a large amount of redundant computations. In *Algorithm* 2, we show Incremental Shapelet extraction on Matrix Profile on Spark (*ISMAP*),

which avoids essentially the repetitive computations on existing dataset. As in Spark environment, the communication cost between distributed nodes is a key factor of system's efficiency. The computing task in each Spark partition should be relatively independent without frequent exchange of intermediate results with other partitions. In light of this, we need to make use of the parallel mechanism to well manage the allocation of computing tasks.

Algorithm 2: ISMAP(Incremental Shapelet extraction on MAtrix Profile on Spark)

Input: Partition $[ID, T, DiscmP, Thresh]$, **New input** T_N, **classSet** $\hat{C}$, k
Output: $\hat{S}$

1 $l_{min} \leftarrow 2$, $l_{max} \leftarrow getMinLen(D)$, $DiscmP \leftarrow []$, $Thresh \leftarrow []$, $\hat{S} \leftarrow \emptyset$
2 $\langle ID_N, T_N \rangle$.broadcast();
3 **MapPartition** $([ID, T, DiscmP, Thresh])$
4 /* 1. compute the Matrix Profile between T_N and all TS in dataset */
5 /* 2. update the current DiscmP of all TS in dataset */
6 /* 3. prepare MP_{T_N} elements to compute $DiscmP_{T_N}$ */
7 **for** $m \leftarrow l_{min}$ to l_{max} **do**
8 $MP_T[m] \leftarrow computeMP(T, T_N, m)$
9 $MP_{T_N}[m] \leftarrow computeMP(T_N, T, m)$
10 $DiscmP[m], Thresh[m] \leftarrow updateDiscmP(DiscmP[m], Thresh[m], MP_T[m])$
11 $DiscmP, Thresh \leftarrow pruning(DiscmP, Thresh)$
12 **emit**$(ID, T, DiscmP, Thresh, MP_{T_N})$
13 **MapAggregation** $(*, (ID, T, DiscmP, Thresh, MP_{T_N}))$
14 $DiscmP_{T_N}, Thresh_{T_N} = computeDiscmP(collect(MP_{T_N}))$
15 $DiscmP_{T_N} \leftarrow DiscmP * \sqrt{1/l}$
16 $DiscmP_{T_N}, Thresh_{T_N} \leftarrow pruning(DiscmP_{T_N}, Thresh_{T_N})$
17 **cache**$(ID_{T_N}, DiscmP_{T_N}, Thresh_{T_N})$
18 **MapAggregation** $(class, (ID, DiscmP, Thresh))$
19 **for** $c \in \hat{C}$ **do**
20 $\hat{S}' \leftarrow getTopk(DiscmP[c], Thresh[c], k)$
21 $\hat{S} \leftarrow \hat{S} \cup \hat{S}'$
22 **return** $\hat{S}$

As shown in *Algorithm* 2, we assume that each Spark partition keeps a set of Time Series with their Discriminative Profiles and corresponding Threshold Distance sets. The newly input Time Series T_N will be broadcast to each distributed node. Information in T_N should be extracted and merged to existing knowledge base, which can be carried out into two steps:

1. **Update existing Shapelets**: With newly input instance T_N, existing candidate Shapelets should update their representative power in each class, and discriminative power in current dataset.
2. **Evaluate new candidate Shapelets**: T_N will introduce new candidate Shapelets of various length, which should be evaluated and placed into Shapelet ranking list by their discriminative power.

Step (1) is shown in *line 8,10*, from the *Formula* 2 and 3, we can observe that the linearity of Discriminative Profile makes the fact that each existing TS only need one single Matrix Profile computation with T_N to update the candidate

Shapelets. As for Step (2), the Discriminative Profile computing of T_N is shared on different Spark partitions, where Matrix Profiles with existing TS instances are computed in *line 9*, an aggregation process in *line 13-17* extracts the discriminative patterns in T_N, which will be aggregated with existing candidate Shapelets and update the output results.

Like classical incremental algorithms, *ISMAP* takes all input instances into account, which means every input TS instance will be imported into system to update the Shapelet, even if the computing imports no valuable information into the system, that says, the information contained in the instance is repetitive with that in knowledge base. Evidently, we are capable of avoiding the redundant information's computation by an interleaved *Test-Then-Train* strategy [5] with an extra evaluation process.

3.2 Evaluation Procedure

The intuition behind the evaluation procedure is that once we have a bad evaluation result, we need to import the instance batch into Shapelet Extraction process, to update the output Shapelet result. As the evaluation time $\mathcal{O}(n-m+1)$ for a TS instance is much less than that of extraction computing ($\mathcal{O}(Nn^3logn)$), then an evaluation module can improve system's efficiency by preventing the computation of certain valueless instances. However, how to define that an instance is valueless stays a problem to resolve.

The classical Shapelet-based approach [27] supposes that a Time Series T can be classified by the inclusion of a class-specified Shapelet $\hat{s}$. (i.e. if $dist(T, \hat{s}) \leq \hat{s}.dist_{thresh}$, then $T.class = \hat{s}.class$). The threshold distance of Shapelet gives a split point to decide the TS-Shapelet inclusion. As shown in [15], various approaches (e.g, Information Gain (IG), Kruskal-Wallis (KW) and Mood's Median (MM)) can be applied for both Shapelet assessment and split point decision. Representative Profile and Discriminative Profile achieve the same effect with these techniques but in a more interpretable manner. Intuitively, we are capable of deciding whether to import a TS instance into Shapelet Extraction process by evaluating its prediction results on current learning model. The Loss Measure is intended to detect the shift between the learning model and the inner concept of data source. In the context of Shapelet, the distance between the learned Shapelets and input instance is able to represent the loss to some extent. Typically, the distance is compared with Shapelets' threshold distance, which derives the *0-1 Loss Function*:

$$L(Y, h(TS)) = \begin{cases} 0, Y = h(TS) \\ 1, Y \neq h(TS) \end{cases}, \text{ where } h(TS) = \begin{cases} C, & if\ dist(T, \hat{s}) \leq \hat{s}.Thresh \\ nonC, & otherwise \end{cases} \tag{5}$$

However, by TS-Shapelet inclusion technique, two Time Series with similar distance to a Shapelet may obtains evidently different classes. In addition, a good prediction result of input TS instance with current Shapelets doesn't mean that the instance contains no useful information for adjusting the learning model. The *0-1 Loss Function* analyzed the surface phenomenon of the prediction but

ignored the deep information behind the arbitrary split point technique. A loss measured by a crisp *0-1 Loss Function* is then ill-adapted.

When $dist(T^C, \hat{s}^C) \leq \hat{s}^C.Thresh$, the prediction result is relatively acceptable. The problem then becomes how to find a balance between time efficiency and TS information checking (i.e., for exhaustive information extraction). The distance can describe the shift between learned Shapelets and real concept, which should be as small as possible. As the distance measure is usually data-dependant, and the absolute distance value varies with datasets, then a normalized measure describing the shift scale is required. To this end, can we just convert the TS-Shapelet inclusion problem to the possibility that a TS contains the Shapelet? We assume that $dist(T, \hat{s}^C)$ satisfies Gaussian distribution, where $T \in D$, the split point of $\hat{s}^C$ is the expectation of the distribution. *Sigmoid* function can be adopted to represent the possibility that T contains $\hat{s^C}$.

For instance, the loss distribution of TS in dataset towards a Shapelet of class C is shown in **Fig.** 4. The smaller the loss, the greater the possibility that T will contain the Shapelet. Intuitively, a loss threshold δ can be set by user for *ISMAP* to control the extraction from input instance, and update incrementally the Shapelet to approach the real concept of data source. When δ is set to 0.5, it has the same effect as *0-1 Loss Function*.

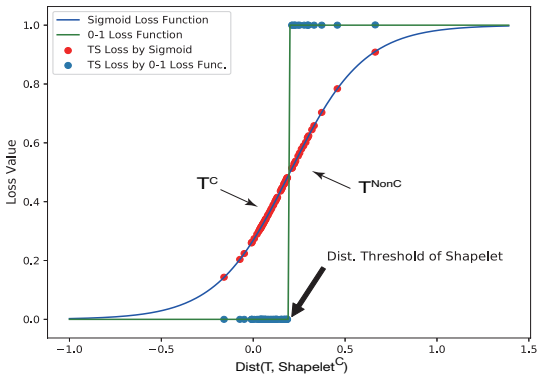

Fig. 4: Loss measure of Time Series by Sigmoid Function and 0-1 Loss Function

However, a stable concept does not hold in several real-life scenarios. For instance, with the soundness of the knowledge in a particular domain, the labeling of newly input instances may evolve gradually, leading to a concept drift. Therefore, the most recent training instances should contribute more than the oldest ones to the target prediction. Then the problem becomes the Concept Drift detection in a Time Series Stream by monitoring the loss function. Conserving the interpretability and explainability of the algorithm, *ISMAP* can be extended to the context of TS Stream by extracting adaptive features.

3.3 Adaptive feature extraction from Time Series Stream

As shown in **Fig.** 5, the system of extracting adaptive Shapelets from Time Series Stream is composed by Shapelet Extraction block and Evaluation Process.

We take TS Chunk $C_{t,w}$ as minimum input unit which contains a number of continuous TS instances: $C_{t,w}$=(T_{t-w+1}, T_{t-w+2}, ..., T_t), where t is the time-tick, w is the window size. The main idea is to evaluate continuously the shift between learned concept and real concept in data source. Once a Concept Drift is detected, the input chunk will be imported into Shapelet Extraction bloc to update the learning model. Both Shapelet initialization and updating process are parallelizable on Spark cluster, which makes use of RAM as caching unit to lower the I/O cost.

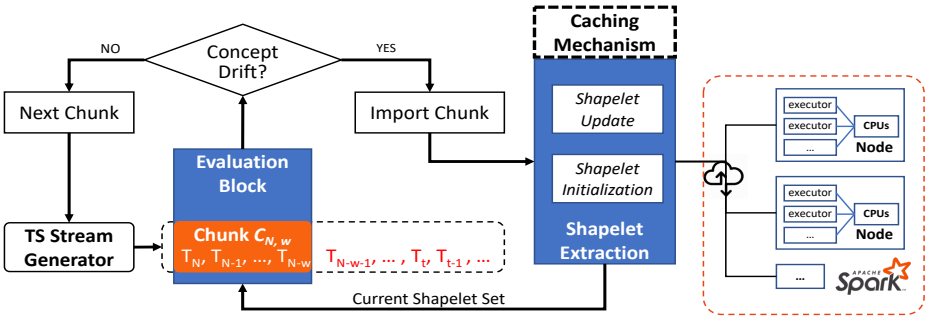

Fig. 5: System Structure in TS Stream context with Concept Drift

1) Shapelet Extraction: The computing process follows the same methodology with *ISMAP*, which allows TS instances in the input chunk to be partitioned on various Spark nodes, the discriminative patterns in each partition will be extracted individually and merged between partitions by their ranking power. The ranking list of Shapelets is then composed by power-updated existing Shapelets and newly imported candidates.

2) Concept Drift Detection: As aforementioned, the loss of a Shapelet on input instances can describe its shift to real concept of data source. With the same methodology, when there is a concept drift in data stream, the analysis tends to be more complicated. The challenge here is to distinguish the measured loss from two aspects:

1. **Incomplete Extraction:** Insufficient training instances introduce the main constraint of Shapelet Extraction, which will bring a relative high loss. More data will make the learning model approach more to the inner concept.
2. **Concept Drift:** The measured shift can only reflect the distance with a stable concept, a big shift will be measured from the evaluation on input instance using out-of-date concept.

In light of these challenges, an advanced analysis on detecting the Concept Drift from measured loss is required. That says, not only to measure the loss from

each TS Chunk, but also to propose a strategy to analyse the loss. Based on the loss definition in section 3.2, we define the average loss for a TS chunk $C_{N,w}$:

$$L_C(N) = \frac{1}{w}\sum_{k=1}^{w} L(y_k, h(T_k)), \text{ where } L(y, h(x)) = |h(x)|, h(x) = \frac{1}{1+e^{-\theta x}} \quad (6)$$

First attempt: A simple technique can be adopted by comparing the average loss of current caching TS chunk $C_{N,w}$ and that of all historical chunks cached in memory:

$$DriftDetection = \begin{cases} True, & if\ L_{avg}(N) \leq L_C(N) \\ False, & otherwise \end{cases} \quad (7)$$

Advanced detection: Page-Hinkley test (PH) [5] is a typical technique used for change detection in signal processing. It allows a loss tolerance for the signal. A cumulative difference between the observed loss and their mean up untim the current time:

$$m_N = \sum_{t=0}^{N}(L_C(t) - L_{avg}(N) - \delta) \quad (8)$$

where $L_{avg}(T)$ is the average loss until the most recent time tick N, δ specifies the tolerable magnitude of changes. The minimum m_N is defined as $M_N = min(m_t, t = 1...N)$. PH test will measure the difference between M_N and m_N:

$$PH_N = m_N - M_N \quad (9)$$

Intuitively, the difference reflects the degree of Concept Drift, when it pass a threshold λ defined by user, then a drift is detected.

3) Caching Mechanism: As the discriminative power of a candidate Shapelet is based on the its global distribution in dataset, the fact that TS instances should be cached in memory is then a necessary condition of Shapelet Extraction. Which is the main difference compared to Concept Drift detection in classical data streams, where it's possible to have one single pass on input instance. Then the main challenge here is to bridge the gap between TS and data stream analysis, that says, to consider the nature of Shapelet in Time Series Stream, and propose a caching mechanism in streaming context which has no contradiction with Shapelet nature, meanwhile the caching volume will not increase indefinitely along with the input the never-ending TS Stream.

As Concept Drift is the fact that the prediction targets at different time tick are different, the previous learned concept is inapplicable to current input data. Conversely speaking, the fresh extracted concept doesn't answer previous prediction target. This fact opens a path to optimize proactively the data caching procedure in memory. The caching mechanism is shown in **Fig.** 6. When there is a state transition of Concept Drift detection, the extraction of a fresh concept is then finished which is applicable for stream instances coming afterwards. The detection of this transition will trigger a cache elimination procedure.

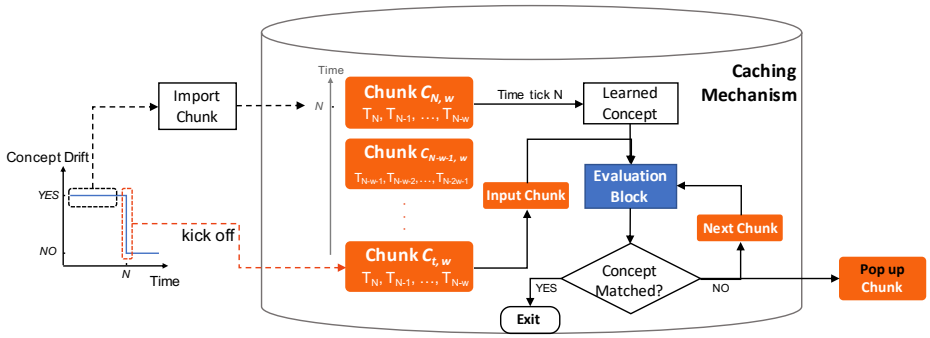

Fig. 6: Caching mechanism of streaming instance chunk in memory

The elimination procedure is based on the assumption that the prediction target of an old TS Chunk is not compatible with the fresh updated concept. By evaluating the cached chunks chronologically, we aim at finding the transition border where historical chunk starts to match the fresh updated concept, which is a reverse process to the detection of cache elimination trigger. We assume that $C_{t,w}$ is the oldest chunk cached in memory, after a trigger is detected, the evaluation will be conducted from $C_{t,w}$ to more recent chunks using the fresh learned concept. If the prediction target in $C_{t,w}$ matches the fresh concept, that means $C_{t,w}$ is in the frame of the fresh concept, the chunks in later time ticks also contributes to the concept's tuning, which can be conserved in memory. Otherwise, $C_{t,w}$ should be removed from cache to eliminate the negative effect to fresh learned concept. The process will not stop until a transition border is detected. By this proactive mechanism, the system is capable of caching a stable volume of data in TS Stream context, and generating adaptive Shapelets in the frame of drifting concept.

4 Experiments and Preliminary Results

All the programs are implemented under Python 3.6. The code source can be found in our project page[3]. The Shapelet Exploration process can be either conducted at local or on a remote Spark cluster. We provide also an 1-click cluster based on Docker, to facilitate the replay of the distributed test offline by the user[4].

We show in **Table** 1 the datasets from UCR/UEA archive [2] used in the experiments. Based on the explainable approach proposed in [29], we test firstly the incremental feature of our algorithm by adopting interleaved Test-Then-Train strategy with an evaluation procedure. The loss threshold $\delta \in [0.2, 0.25, 0.3, 0.35, 0.4, 0.5]$, which controls the sensitivity of system for importing TS instance into Shapelet Extraction process. The first four datasets in **Table** 1 are adopted for the incremental test.

[3] https://github.com/JingweiZuo/ISETS/
[4] https://github.com/JingweiZuo/ISETS/tree/master/Spark_Cluster_Docker

Table 1: Datasets from UCR/UEA Archive used in the experiments

Name	Type	Train	Test	Class	Length
GunPoint	Motion	50	150	2	150
Trace	Sensor	100	100	4	275
ECG5000	ECG	500	4500	5	140
Two_Patterns	Simulated	1000	4000	4	128
ElectricDevices	Device	8926	7711	7	96

From the results shown in **Fig.** 7, we can know that a high loss threshold δ leads to a high efficiency at the expense of certain accuracy. However, for certain datasets (e.g., GunPoint, ECG5000), the accuracy stays on a relative stable stage even with the increase of δ, which can be explained by the fact that the instances from the same class are highly consistent, and share the common Shapelet features. Where the system efficiency can be largely improved without affecting the reliability of learning model. The poor accuracy result for the dataset *Two_Patterns* shows the constraint of Shapelet that is inapplicable for the scenarios where the classes are characterised by global series features or frequent items.

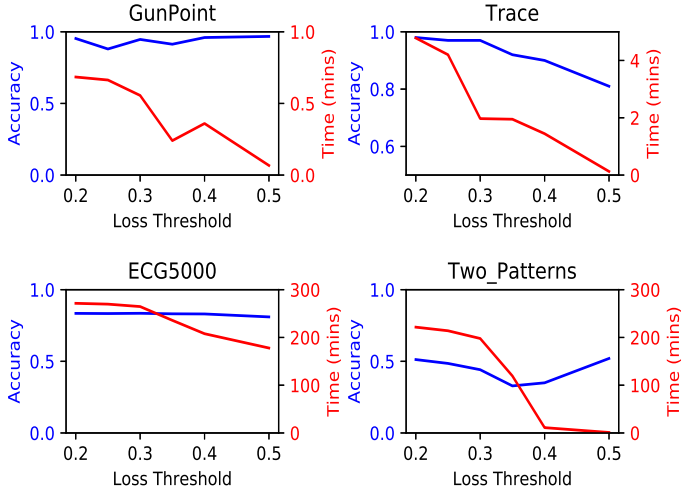

Fig. 7: Results of incremental test by adopting an extra evaluation procedure

When we are in the context of Time Series Stream, the current dataset doesn't reflects a such Concept Drift nature. Then we need to adjust manually the data source to comply the test scenario. Here we choose the dataset *ElectricDevices*, which contains a large number of instances and is eligible for simulating the test scenario by shifting the class of dataset on different time

ticks. We set 3 Concept Drifts equally distributed on time axis (with the limit of training instance number). As shown in **Fig.** 8, The training/testing set is sampled individually into 3 equal-sized subsets with different concepts. Due to the page limit here, further experiment results can be found in our project page[5].

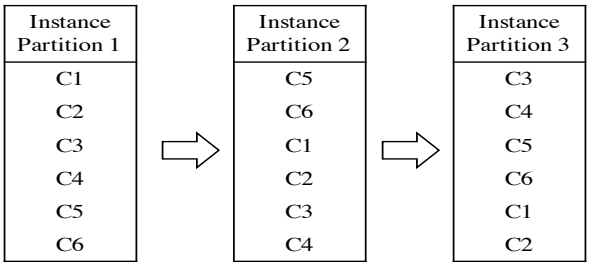

Fig. 8: The simulation of TS Stream context with Concept Drift on different time ticks, the class labels are changed by human intervention

5 Conclusion

In this paper, we studied the dynamic feature exploration over Time Series Stream, which is based on the interpretable Shapelet features and an explainable Shapelet extraction process. An incremental Shapelet extraction with a novel evaluation procedure on a stable concept is proposed which is applicable to adjust program's efficiency by different shift tolerance degree. As for a non-stable concept data source, we adjust the conventional strategies of Concept Drift detection into the context of Time Series Stream, which opens the path for a proactive optimization of caching data in memory. The system can be applied in the scenario where an existing dataset should be enriched with new knowledge without human loop in the middle.

However, there are still constraints for Shapelet-based approaches in the context of Time Series Stream. Time efficiency and dependence on caching instances are two mains aspects to be improved in the future.

Acknowledgements

This research was supported by DATAIA convergence institute as part of the *Programme d'Investissement d'Avenir*, (ANR-17-CONV-0003) operated by DAVID Lab, University of Versailles Saint-Quentin, and MASTER project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie-Slodowska Curie grant agreement N. 777695.

[5] https://github.com/JingweiZuo/ISETS/

References

1. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. Tech. rep. (2010), http://www.jmlr.org/papers/volume11/bifet10a/bifet10a.pdf
2. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The ucr time series classification archive (2018), https://arxiv.org/pdf/1810.07758.pdf
3. Deng, H., Runger, G., Tuv, E., Vladimir, M.: A time series forest for classification and feature extraction (2013). https://doi.org/10.1016/j.ins.2013.02.030, http://dx.doi.org/10.1016/j.ins.2013.02.030
4. Fang, Z., Wang, P., Wang, W.: Efficient learning interpretable shapelets for accurate time series classification. Proceedings - IEEE 34th International Conference on Data Engineering, ICDE 2018 pp. 497–508 (2018). https://doi.org/10.1109/ICDE.2018.00052
5. Gama, J., Zliobait E, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A Survey on Concept Drift Adaptation. A Survey on Concept Drift Adap-tation. ACM Comput. Surv. 1, 1, Article **1** (2013). https://doi.org/10.1145/0000000.0000000, http://doi.acm.org/10.1145/0000000.0000000
6. Gokce Baydogan, M., Runger, G., Keogh Mustafa Gokce Baydogan, E.B.: Time series representation and similarity based on local autopatterns. Data Mining and Knowledge Discovery **30**, 476–509 (2016)
7. Górecki, T., Łuczak, M.: Using derivatives in time series classification. Data Mining and Knowledge Discovery **26**(2), 310–331 (2013). https://doi.org/10.1007/s10618-012-0251-4
8. Górecki, T., Łuczak, M.: Non-isometric transforms in time series classification using DTW. Knowledge-Based Systems **61**, 98–108 (5 2014). https://doi.org/10.1016/j.knosys.2014.02.011, https://linkinghub.elsevier.com/retrieve/pii/S0950705114000604
9. Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning Time-Series Shapelets (2014). https://doi.org/10.1145/2623330.2623613, http://dx.doi.org/10.1145/2623330.2623613
10. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Tech. rep. (2019). https://doi.org/10.1007/s10618-019-00619-1, https://arxiv.org/pdf/1809.04356.pdf
11. Karim, F., Majumdar, S., Darabi, H., Chen, S.: LSTM Fully Convolutional Networks for Time Series Classification. IEEE Access **6**, 1662–1669 (2017). https://doi.org/10.1109/ACCESS.2017.2779939
12. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Tech. rep., http://www.cs.ucr.edu/~eamonn/kais_2000.pdf
13. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '03 p. 2 (2003)
14. Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems **39**(2), 287–315 (2012). https://doi.org/10.1007/s10844-012-0196-5, https://link.springer.com/content/pdf/10.1007%2Fs10844-012-0196-5.pdf

15. Lines, J., Bagnall, A.: Alternative Quality Measures for Time Series Shapelets. Tech. rep., http://www.uea.ac.uk/cmp
16. Lines, J., Bagnall, A., Lines, J., Bagnall, A., Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Min Knowl Disc **29**, 565–592 (2015). https://doi.org/10.1007/s10618-014-0361-2, https://link.springer.com/content/pdf/10.1007%2Fs10618-014-0361-2.pdf
17. Lines, J., Davis, L.M., Hills, J., Bagnall, A.: A Shapelet Transform for Time Series Classification. Tech. rep. (2012), http://wan.poly.edu/KDD2012/docs/p289.pdf
18. Lines, J., Hills, J., Bagnall, A.: The Collective of Transformation-Based Ensembles for Time-Series Classification. IEEE Transactions on Knowledge and Data Engineering **27**(9), 2522–2535 (2015). https://doi.org/10.1109/TKDE.2015.2416723
19. Lines, J., Taylor, S., Bagnall, A.: HIVE-COTE: The Hierarchical Vote Collective of Transformation-based Ensembles for Time Series Classification. Tech. rep., www.timeseriesclassification.com/icdm2016.php
20. Mustafa Gokce Baydogan, G.R., Tuv, E.: A Bag-of-Features Framework to Classify Time Series. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE **39**(10), 2104–2111 (2017)
21. Rakthanmanon, T., Keogh, E.: Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In: Proceedings of the 2013 SIAM International Conference on Data Mining, pp. 668–676 (2013). https://doi.org/10.1137/1.9781611972832.74, http://epubs.siam.org/doi/abs/10.1137/1.9781611972832.74
22. Ratanamahatana, C.A., Keogh, E.: Three Myths about Dynamic Time Warping Data Mining. In: Proceedings of the 2005 SIAM International Conference on Data Mining, pp. 506–510 (2005). https://doi.org/10.1137/1.9781611972757.50, http://epubs.siam.org/doi/abs/10.1137/1.9781611972757.50
23. Schäfer, P.: The BOSS is concerned with time series classification in the presence of noise. Data Min Knowl Disc **29**, 1505–1530 (2015). https://doi.org/10.1007/s10618-014-0377-7, http://www.physionet.org/physiobank/database/
24. Senin, P.: Interpretable Time Series Classification Using SAX and Vector Space Model
25. Ueno, K., Xi, A., Keogh, E., Lee, D.J.: Anytime classification using the nearest neighbor algorithm with applications to stream mining. Proceedings - IEEE International Conference on Data Mining, ICDM (December), 623–632 (2006). https://doi.org/10.1109/ICDM.2006.21
26. Wang, J., Wang, Z., Li, J., Wu, J.: Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis. KDD p. 10 (2018). https://doi.org/10.1145/3219819.3220060, https://doi.org/10.1145/3219819.3220060
27. Ye, L., Keogh, E.: Time series shapelets: A New Primitive for Data Mining. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09 p. 947 (2009)
28. Yeh, C., Zhu, Y., Ulanova, L., Begum, N.: Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets. Ieee (2016). https://doi.org/10.1109/ICDM.2016.0179, https://pdfs.semanticscholar.org/6152/3cfe6f51859e00aa8ce320114c03151208fa.pdf
29. Zuo, J., Zeitouni, K., Taher, Y.: Exploring Interpretable Features for Large Time Series with SE4TeC (2019). https://doi.org/10.5441/002/edbt.2019.67, https://openproceedings.org/2019/conf/edbt/EDBT19_paper_353.pdf