Incremental and Adaptive Feature Exploration over Time Series Stream

Jingwei Zuo, Karine Zeitouni, and Yehia Taher

DAVID Lab. University of Versailles - Université 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 (i.e. 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, by adopting the concept *Shapelet* and *Matrix Profile*, we conduct the first attempt to extract the adaptive features from Time Series Stream based on the *Test-then-Train* strategy, which is applicable in both contexts: a) under stable concept, learning model will be updated incrementally; b) for data source with Concept Drift, previous concepts that do not represent to current stream behavior will be discarded from the model.

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 [8,9,19,24]. Instead of 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 [3,22,7]. With the emergence of TS dimensionality reduction techniques (e.g., PAA [13], SAX [15], etc.), TS instances can be represented by high-dimensional vectors so that various techniques [14] 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 [16,26,25] 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 [29]. Various Shapelet-based approaches have been proposed to optimize both the accuracy [20,10] and the efficiency [23,4] of the classification. Another remarkable attempt [21,17] 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 is *Concept Drift*. For instance, the most accurate ensemble classifiers [21,17] are not good options in streaming context due to their complex architecture. Lazy classifiers on Time Series such as Nearest Neighbor (1-NN) [27] and dictionary based approaches [16] 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 [11,12,28] 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. An advanced work in [32] ensures the explainability of Shapelet Extraction process even to non-expert, which allows us to further explore the streaming context conserving the current goodness.

To fill 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 incremental version of previous Shapelet work [32] under Spark framework allows us to further explore the *Test-then-Train* strategy, to evaluate the learning model constantly on newly input instance, then update the model regarding to the evaluation result. The cached information under old concept will be elimi-

nated gradually by an elastic caching mechanism, which deals with the challenge of infinite input of streaming instances.

The main contributions of this paper are as follows:

1. **Test-then-Train**: The novel strategy for feature exploration on TS stream, not only accelerates the incremental Shapelet extraction in stable-concept context, but also helps with detecting 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 Shapelet Extraction [32] in streaming context, which 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, Section 3 shows our system in both TS stream contexts of stable concept and drifting concept. Section 4 shows an empirical evaluation of our method on real-life and synthetic datasets. Finally, we give our conclusions and perspectives for future work in Section 5.

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 $i \in [0, n - m + 1]$.

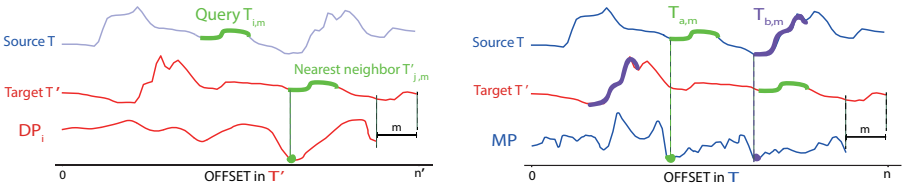
Definition 6: *Shapelet* $\hat{s}$ is a time series subsequence which is particularly representative of a class. As such, it shows a shape which can distinguish one class from the others.

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, $DP_i^m = (DP_{i,1}^m, ..., DP_{i,j}^m, ..., T_{i,n'-m+1}^m)$, where $DP_{i,j}^m = dist(T_{i,m}, T'_{j,m}), \forall j \in [0, n' - m + 1]$, n' is the length of T'.

We assume that the distance is measured by Euclidean Distance between z-normalized subsequences [31]. Authors in [30] propose *MASS* which is considered as the fastest exact 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, instead of $\mathcal{O}(nm^2)$ [29] with several orders of magnitude higher.



(a) Distance Profile DP_i (b) Matrix Profile MP

Fig. 1: (a) 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'; (b) 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 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, $MP^m = (MP_1^m, ..., MP_i^m, ..., MP_{n-m+1}^m)$, where $MP_i^m = min(DP_i^m)$, where $i \in [0, n - m + 1]$, n is the length of T. A visual presentation of MP is shown in **Fig.** 1.

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

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

The representative power of subsequence T_i in class C is defined as:

$$RP_{T_i}^C = avg(MP_{T_i,T'}) \tag{2}$$

where $T' \in D_t^C$. 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.** 2 (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.** 2 (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 pattern, that is the candidate Shapelet, can be visually identified in T by their discriminative power.

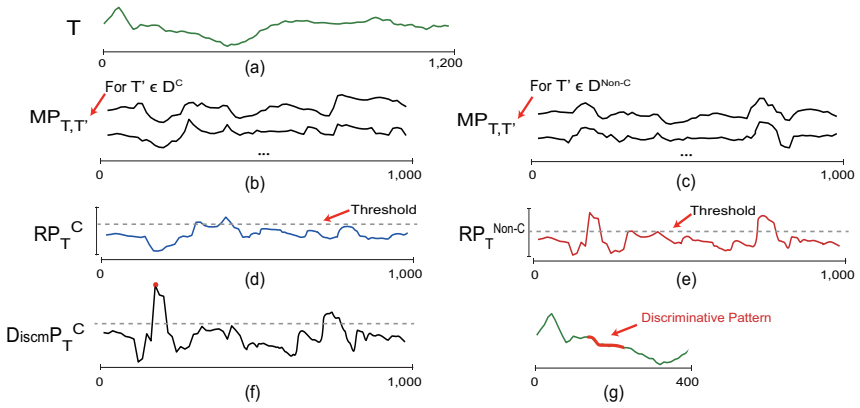


Fig. 2: *(a)* Time Series T with class C. *(b,c)* Matrix Profile set for T with TS instance in different class. *(d,e)* Representative Profile of T in different class. A threshold can determine the representative area of T. *(f)* Discriminative Profile of T in dataset. The highest point in $DiscmP_T^C$ represents the most discriminative pattern's position in T.

2.2 A Brief Review of the SMAP algorithm for TSC

Matrix profile [30] provides a meta-data which facilitates the representation of a complex correlation between two time series. Authors in [32] 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: $\hat{S}$

$l_{min} \leftarrow 0.1 * getMinLen(D)$, $l_{max} \leftarrow 0.5 * getMinLen(D)$,
$DiscmP \leftarrow []$, $dist_{Thresh} \leftarrow []$, $\hat{S} \leftarrow \emptyset$
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**
      $DiscmP[m], dist_{Thresh}[m] \leftarrow computeDiscmP(T, D, m)$
      $DiscmP[m] \leftarrow DiscmP[m] * \sqrt{1/l}$
    $DiscmP, dist_{Thresh} \leftarrow pruning(DiscmP, dist_{Thresh})$
    emit$(ID, DiscmP, dist_{Thresh})$
MapAggregation *(*$class, (ID, DiscmP, dist_{Thresh})$*)*
  for $c \in \hat{C}$ **do**
    $\hat{S}' \leftarrow getTopk(DiscmP[c], dist_{Thresh}[c], k)$
    $\hat{S} \leftarrow \hat{S} \cup \hat{S}'$
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[29]. *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 Algorithm and System Structure

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

1 $l_{min} \leftarrow 0.1 * getMinLen(D)$, $l_{max} \leftarrow 0.5 * getMinLen(D)$,
2 $DiscmP \leftarrow []$, $dist_{Thresh} \leftarrow []$, $\hat{S} \leftarrow \emptyset$
3 $\langle ID_N, T_N \rangle$.broadcast();
4 **MapPartition** *(*$[ID, T, DiscmP, dist_{Thresh}]$*)*
5 /* 1. compute the Matrix Profile between T_N and all TS in dataset */
6 /* 2. update the current DiscmP of all TS in dataset */
7 /* 3. prepare MP_{T_N} elements to compute $DiscmP_{T_N}$ */
8 **for** $m \leftarrow l_{min}$ to l_{max} **do**
9 $MP_T[m] \leftarrow computeMP(T, T_N, m)$
10 $MP_{T_N}[m] \leftarrow computeMP(T_N, T, m)$
11 $DiscmP[m], dist_{Thresh}[m] \leftarrow$ $updateDiscmP(DiscmP[m], dist_{Thresh}[m], MP_T[m])$
12 $DiscmP, dist_{Thresh} \leftarrow pruning(DiscmP, dist_{Thresh})$
13 **emit**$(ID, T, DiscmP, dist_{Thresh}, MP_{T_N})$
14 **MapAggregation** *(*$*, (ID, T, DiscmP, dist_{Thresh}, MP_{T_N})$*)*
15 $DiscmP_{T_N}, dist_{Thresh T_N} = computeDiscmP(collect(MP_{T_N}))$
16 $DiscmP_{T_N} \leftarrow DiscmP * \sqrt{1/l}$
17 $DiscmP_{T_N}, dist_{Thresh T_N} \leftarrow pruning(DiscmP_{T_N}, dist_{Thresh T_N})$
18 **cache**$(ID_{T_N}, DiscmP_{T_N}, dist_{Thresh T_N})$
19 **MapAggregation** *(class,* $(ID, DiscmP, dist_{Thresh})$*)*
20 **for** $c \in \hat{C}$ **do**
21 $\hat{S}' \leftarrow getTopk(DiscmP[c], dist_{Thresh}[c], k)$
22 $\hat{S} \leftarrow \hat{S} \cup \hat{S}'$
23 **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 in *line 18-21*.

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

3.2 Shapelet Evaluation

The intuition behind the evaluation procedure is that once we have a bad evaluation result, we need to import the instance 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 [29] 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 [18], 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*:

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

When $dist(T, \hat{s}) \leq \hat{s}.dist_{Thresh}$, the prediction result is relatively acceptable. The problem then becomes how to find a balance between time efficiency and TS information checking (i.e., the exhaustive information extraction). The distance between TS and Shapelets describes the shift between real and learned concept, a small distance leads to a reliable prediction result. 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?

As extracted Shapelets try to separate one class to others, TSs in different classes tend to be concentrated on the split point, which causes the main error in prediction. Then we assume that $dist(T, \hat{s})$ satisfies Gaussian distribution, as shown in Equation 6, the loss can be smoothed by *Sigmoid* function by considering distance distribution. The split point of $\hat{s}$ defines the expectation σ of the distribution.

$$\begin{aligned} L(Y, h(T)) &= \frac{1}{1 + e^{-(x-\sigma)}}, \quad \sigma = \hat{s}.dist_{Thresh} \\ x &= min(dist(T^C, \hat{s})), \quad \hat{s} \in \hat{S}^C \end{aligned} \tag{6}$$

As shown in **Fig.** 3, 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*.

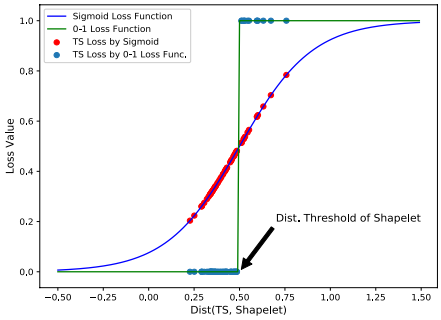


Fig. 3: Loss measure of Time Series by Sigmoid Function and 0-1 Loss Function, Time Series in different classes are distributed around the split point of Shapelet

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.** 4, 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. By adopting the *Test-then-Train* strategy, the main idea here is to evaluate continuously the shift between learned concept and real concept in

data source (i.e., *Test*). Once a Concept Drift is detected, the input chunk will be imported into Shapelet Extraction bloc to update the learning model (i.e., *Train*). 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.

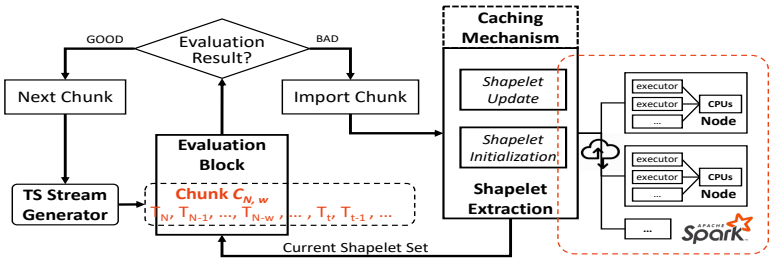


Fig. 4: System Structure in TS Stream context with Concept Drift

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:** As main constraint in Shapelet Extraction, insufficient training instances (i.e., under-fitting) will bring a relative high loss. More data will make the learning model approach more the inner concept.
2. **Concept Drift:** The measured shift can only reflect the distance to a stable concept, a big shift will be observed using out-of-date learning model.

In light of these challenges, an advanced analysis on detecting the Concept Drift from measured loss is required. That is, 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 Equation 6, we define the average loss for a TS chunk $C_{N,w}$:

$$L_C(N) = \frac{1}{w} \sum_{k=1}^{w} L(Y_{N-w+k}, h(T_{N-w+k})) \tag{7}$$

Concept Enrichment: As mentioned in 3.2, an user-defined loss threshold Δ can be set to decide whether to import the chunk into the system to enrich the concept. That is: $Import_{Chunk} = True$ if $L_C(N) \leq \Delta$.

Concept Drift 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. The sequential test on the variance which considers that normal operation corresponds to a certain variance and a drift being characterized by an increase

in this variance. Here we define a cumulative difference between the observed loss and their mean up until the current time:

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

where $L_{avg}(t)$ is the average loss until the current time tick t, 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:

$$PH_N = m_N - M_N \tag{9}$$

Intuitively, the difference reflects the degree of Concept Drift, when it exceeds a user-specified threshold λ, then the Concept Drift is detected.

3) Caching Mechanism: As the discriminative power of a candidate Shapelet is based on its global distribution in the dataset, the fact that TS instances should be cached in memory is then a necessary condition of Shapelet Extraction. This 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 consider the nature of Shapelet in Time Series Stream, and propose a Shapelet-based caching mechanism in streaming context, meanwhile the caching volume should 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.** 5. 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.

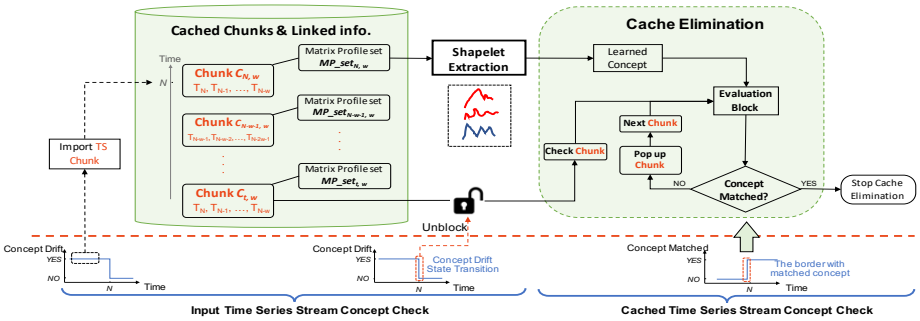


Fig. 5: Caching mechanism of streaming instance chunk in memory

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 kept in memory. Otherwise, $C_{t,w}$ should be removed from cache to eliminate the negative effect to the 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 Results

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

4.1 Experimental design

The experiment is conducted by two steps: A). We test the incremental feature and reliability of ISMAP after adopting Shapelet Evaluation process in Test-then-Train strategy. We evaluate the improvement of Shapelet Extraction in both efficiency and accuracy on dataset with stable concept; B). We check the reliability of adaptive Shapelet Extraction from TS stream with Concept Drift.

[**Datasets**] We conduct our first incremental experiments on 14 Shapelet datasets in UCR Archive [2], instead of testing all datasets under various problem types. As for the streaming scenario with *Concept Drift*, we generate synthetic dataset from *Trace* (see Table 1), which has a high reliability in Shapelet-based approaches and eligible for simulating the scenario by changing the class within some chunks on different time ticks. As a larger number of instances is required for Concept Drift assessment in the streaming TS context, we augment the data volume by randomly putting noise [6] in TS instances with a random duration. The augmentation degree is set to *10* times of original volume. The total labelled instances are sampled into 3 equal-sized subsets with different concepts.

[**Parameters**] The initial Shapelet Extraction algorithm is based on [32], where the Shapelet length $m \in [0.1n, 0.5n]$ with a step of $0.25m$, n is TS length. As for the loss threshold for Shapelet Evaluation, $\Delta \in [0.20, 0.50]$, with a step of 0.05. For Concept Drift Detection, we take the loss threshold Δ which brings the highest accuracy in raw dataset. The tolerance $\delta \in \{0.15, 0.30\}$, PH threshold $\lambda = 0.4$. The TS chunk size is fixed at *5*.

4.2 Results on ISMAP

We focus on the feature itself. Therefore, rather than considering classifiers learned over shapelet-transformed data[20], we take the Shapelet Tree methods as baselines, which utilize different quality measures to extract the Shapelet

[1] https://github.com/JingweiZuo/TSStreamMining

Table 1: Shapelet Datasets in UCR Archive used for Incremental Test (ISMAP)

Type	Name	Train/Test	Class	Length	IG	KW	MM	ISMAP(best)	Para. (Δ)	Comp. Ratio
Simulated	SyntheticControl	300/300	6	60	**0.9433**	0.9000	0.8133	0.7007	0.35	46.7%
Sensor	Trace	100/100	4	275	0.9800	0.9400	0.9200	**1**	0.5, 0.45	26.0%
	MoteStrain	20/1252	2	84	0.8251	0.8395	0.8395	**0.9169**	0.45	60.0%
	SonyAIBO.I	20/601	2	70	0.8453	0.7281	0.7521	**0.9151**	0.4	95.0%
	SonyAIBO.II	27/953	2	65	0.8457	-	-	**0.8583**	0.4	63.0%
	ItalyPower.	67/1029	2	24	0.8921	0.9096	0.8678	**0.9466**	0.45	25.4%
ECG	ECG5000	500/4500	5	140	0.7852	-	-	**0.9109**	0.4	9.4%
	ECGFiveDays	23/861	2	136	0.7747	0.8721	0.8432	**0.9826**	0.4	51.2%
	TwoLeadECG	23/1189	2	82	0.8507	0.7538	7657	**0.9337**	0.5	47.8%
Images	Symbols	25/995	6	398	0.7799	0.5568	0.5799	**0.8113**	0.35	96.0%
	Coffee	28/28	2	286	**0.9643**	0.8571	0.8671	0.9286	0.4	78.6%
	FaceFour	24/88	4	350	0.8409	0.4432	0.4205	**0.9886**	except 0.45	62.5%
	DiatomSize.	16/306	4	345	0.7222	0.6111	0.4608	**0.8758**	0.5	50.0%
Motion	GunPoint	50/150	2	150	0.8933	0.9400	0.9000	**0.9733**	0.45	42.0%

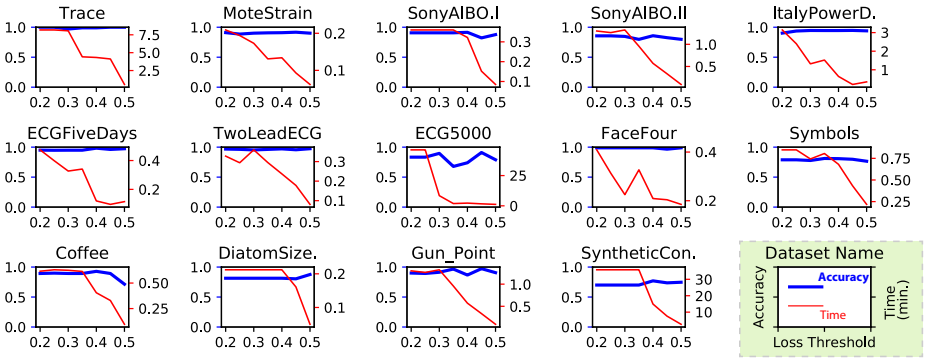


Fig. 6: Results of Incremental Test (ISMAP) by adopting Shapelet Evaluation

and predict target instance: a) Information Gain (IG)[29], b) Kruskall-Wallis (KW) [18], c) Mood's Median (MM) [18]. As quality measure's calculation is negligible compared to the total time cost, the computation time should remain at the same level when they adopt the same distance measure (e.g., *MASS*), and when *ISMAP* doesn't adopt a Test-then-Train strategy.

Table 1 shows the accuracy performance comparison between baselines and our approach. Obviously, *ISMAP* achieved the top performance on accuracy metric on more datasets than any other classifier (12 of 14). Specifically for sensor, motion and ECG data, *ISMAP* performs no doubt better than other approaches, and achieved more than 20% accuracy improvement in *ECG5000*. **Table** 1 shows as well the parameter Δ which brings the best accuracy performance. The Compression Ratio is defined by the proportion of imported valuable instances over total training instances: $Comp.Ratio = \frac{nbr.instance_{imported}}{nbr.instance_{training}}$, the ratio below 1 brings a better performance in both time and memory cost.

Fig. 6 shows a global view of accuracy and time cost tested by *ISMAP* under different loss thresholds. Most of the time the accuracy keeps 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. Therefore, the system efficiency can be largely improved with an exchange of a negligible decrease of accuracy.

4.3 Results on adaptive features in TS Stream

Fig. 7 shows the Concept Drift detection process on the *Trace* dataset under different loss tolerance level θ. The Concept Drifts were detected by the system within the time periods [345, 380]/[350, 365] and [670/790]/[675, 700] under different θ, which are in strong accordance with the true drifts in the dataset. Different θ leads to different time periods for adjusting the learned concept. A high tolerance is capable of relieving the effect of outliers or excessive feedback, and allows only a continuous high loss to be considered as a Concept Drift. Therefore, less chunks are imported into the system which leads to less time cost. From the memory and time plot on right **Fig.** 7, at the end of each drift adjustment area, the cached information is largely eliminated, finally only 100 of 1000 ($\delta = 0.15$) or 50 of 1000 ($\delta = 0.30$) instances of *Aug. Trace* dataset are cached in the memory. The proactive caching elimination mechanism shows its elastic feature. Besides, the later imported chunks requires usually longer calculation time (the step becomes longer in the time plot), as more chunks have been cached.

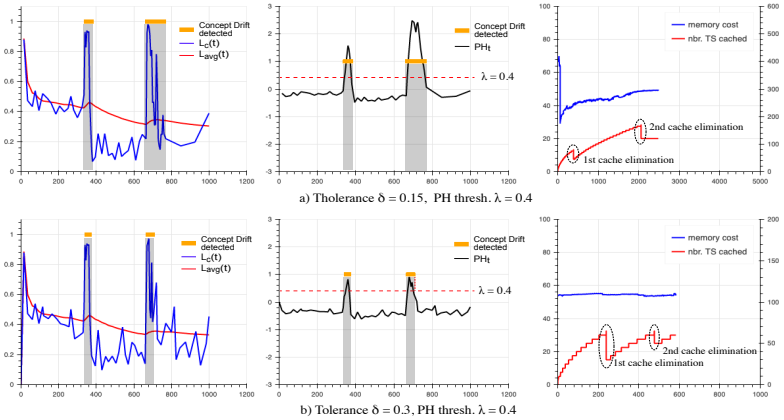


Fig. 7: Results of Concept Drift Detection on augmented *Trace* dataset

Table 2: Evaluation in datasets with manually added drift

Dataset	-	**i**(Con. 1)	**ii**(Con. 2)	**iii**(Con. 2)	**iv**(Con. 3)
$Aug.Trace(\delta = 0.15)$	Time tick	*345*	*380*	*670*	*790*
	Test Accu.	0.9600	0.9900	0.9900	0.9800
$Aug.Trace(\delta = 0.30)$	Time tick	*350*	*365*	*675*	*700*
	Test Accu.	0.9600	0.9800	0.9800	0.9700

In **Table** 2, we show the reliability of the Extracted Shapelets on 4 time ticks at the beginning/end of each drift area[2]. The extracted Shapelets perform the same accuracy at the two middle time ticks, which can be explained by the fact that no chunks were imported since the learned *Concept 2* was deemed enough reliable. Globally, the adaptive Shapelets show a high accuracy in such a streaming context with Concept Drift, although the accuracy is a little lower

[2] We recall that the *Trace* testing dataset were augmented in the same manner, where drifts were manually added.

than that in **Table** 1, as the training on the subsets gets less information than that on the entire dataset.

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 under stable concept with a novel Shapelet evaluation process is proposed, which improved largely the system's efficiency with an exchange of a negligible decrease of accuracy. 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 elimination of data cached in the memory. The system can be applied in the scenario where an existing dataset should be enriched with new knowledge but 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.

References

1. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. Tech. rep. (2010)
2. Dau, H.A., Keogh, E., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The ucr time series classification archive (October 2018), `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`
3. Deng, H., Runger, G., Tuv, E., Vladimir, M.: A time series forest for classification and feature extraction. Information Sciences **239**, 142–153 (2013)
4. Fang, Z., Wang, P., Wang, W.: Efficient learning interpretable shapelets for accurate time series classification. Proc. ICDE 2018 pp. 497–508
5. Gama, J., Zliobait E, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A Survey on Concept Drift Adaptation. ACM Comput. Surv. 1, 1, Article **1** (2013)
6. Gharghabi, S., Imani, S., Bagnall, A., Darvishzadeh, A., Keogh, E.: An Ultra-Fast Time Series Distance Measure to allow Data Mining in more Complex Real-World Deployments. ICDM'18
7. 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** (2016). https://doi.org/10.1007/s10618-015-0425-y
8. Górecki, T., Łuczak, M.: Using derivatives in time series classification. Data Mining and Knowledge Discovery **26**(2), 310–331 (2013)

9. Górecki, T., Łuczak, M.: Non-isometric transforms in time series classification using dtw. Know.-Based Syst. **61**, 98–108 (2014)
10. Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. pp. 392–401. KDD '14, ACM, New York, NY, USA (2014)
11. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Tech. rep. (2019)
12. Karim, F., Majumdar, S., Darabi, H., Chen, S.: LSTM Fully Convolutional Networks for Time Series Classification. IEEE Access **6**, 1662–1669 (2017)
13. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Tech. rep.
14. Lei, Q., Yi, J., Vaculin, R., Wu, L., Dhillon, I.S.: Similarity Preserving Representation Learning for Time Series Clustering. Tech. rep. (2019)
15. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In Proc. DMKD@SIGMOD'03
16. Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems **39** (2012)
17. Lines, J., Taylor, S., Bagnall, A.: Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In: ICDM (2016)
18. Lines, J., Bagnall, A.: Alternative Quality Measures for Time Series Shapelets. Tech. rep., `http://www.uea.ac.uk/cmp`
19. Lines, J., Bagnall, A., Lines, J., Bagnall, A., Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Min Knowl Disc **29** (2015)
20. 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`
21. Lines, J., Hills, J., Bagnall, A.: The Collective of Transformation-Based Ensembles for Time-Series Classification. IEEE TKDE **27**(9), 2522–2535 (2015)
22. Mustafa Gokce Baydogan, G.R., Tuv, E.: A Bag-of-Features Framework to Classify Time Series. IEEE Trans Pattern Anal Mach Intell **39**(10), 2104–2111 (2017)
23. Rakthanmanon, T., Keogh, E.: Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In: Proc. SDM, pp. 668–676 (2013)
24. Ratanamahatana, C.A., Keogh, E.: Three Myths about Dynamic Time Warping Data Mining. In: Pro. SDM, pp. 506–510 (2005)
25. Schäfer, P.: The BOSS is concerned with time series classification in the presence of noise. Data Min Knowl Disc **29**, 1505–1530 (2015)
26. Senin, P., Malinchik, S.: Sax-vsm: Interpretable time series classification using sax and vector space model. In: ICDM'13. pp. 1175–1180 (2013)
27. Ueno, K., Xi, A., Keogh, E., Lee, D.J.: Anytime classification using the nearest neighbor algorithm with applications to stream mining. Proc. ICDM (2006)
28. Wang, J., Wang, Z., Li, J., Wu, J.: Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis. KDD'18 p. 10
29. Ye, L., Keogh, E.: Time series shapelets: A New Primitive for Data Mining. Proc. KDD '09 p. 947 (2009). https://doi.org/10.1145/1557019.1557122
30. 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
31. Zhu, Y., Zimmerman, Z., Senobari, N.S., Yeh, C.C.M., Funning, G., Mueen, A., Brisk, P., Keogh, E.: Matrix profile II: Exploiting a novel algorithm and GPUs to break the one hundred million barrier for time series motifs and joins. Proceedings - IEEE International Conference on Data Mining, ICDM pp. 739–748 (2017)
32. Zuo, J., Zeitouni, K., Taher, Y.: Exploring interpretable features for large time series with se4tec. In: EDBT 2019, Lisbon, Portugal. pp. 606–609 (2019)