Cost-Aware Early Classification of Time Series

Romain Tavenard, Simon Malinowski

Cost-Aware Early Classification of Time Series

Romain Tavenard[1] and Simon Malinowski[2]

[1] LETG-Rennes COSTEL / IRISA – Univ. Rennes 2
[2] IRISA – Univ. Rennes 1

Abstract. In time series classification, two antagonist notions are at stake. On the one hand, in most cases, the sooner the time series is classified, the more rewarding. On the other hand, an early classification is more likely to be erroneous. Most of the early classification methods have been designed to take a decision as soon as sufficient level of reliability is reached. However, in many applications, delaying the decision with no guarantee that the reliability threshold will be met in the future can be costly. Recently, a framework dedicated to optimizing a trade-off between classification accuracy and the cost of delaying the decision was proposed, together with an algorithm that decides online the optimal time instant to classify an incoming time series. On top of this framework, we build in this paper two different early classification algorithms that optimize a trade-off between decision accuracy and the cost of delaying the decision. These algorithms are non-myopic in the sense that, even when classification is delayed, they can provide an estimate of when the optimal classification time is likely to occur. Our experiments on real datasets demonstrate that the proposed approaches are more robust than existing methods.

Keywords: time series classification ; early classification

1 Introduction

Time series classification has received a lot of attention from researchers in the last years due to the increasing amount of data available and its applicability in many domains like medicine, environment, business, *etc.* Classical methods for time series classification have been designed to make a decision based on complete time series. However, in many applications where observations arrive one at a time, it is rewarding to be able to classify a time series even without knowing it entirely. In this case, each time a new sample arrives, one can decide either to perform classification or to wait for more data. Consequently, two antagonist notions are at stake. It is more valuable to make a decision as soon as possible, but premature decisions are more likely to be erroneous. For example, in applications related to business, higher profits can be made thanks to early classification of customer behaviours, provided that the classification is accurate. Hence, two different costs clearly appear from this point of view : the cost of delaying the decision and the cost of misclassification. Most of the approaches for early classification in the literature focus on the design of algorithms that

make a reliable decision from incomplete time series, without explicitly accounting for the cost of delaying the decision [1, 4, 5, 6, 7, 10, 13, 14]. In these works, sufficient guarantee of accuracy triggers the decision.

In this paper, we focus on cost-aware early classification of time series. Let us illustrate the advantages of cost-aware early classification of time series on a concrete example. In a surveillance scenario, one would want the system to warn the police as soon as a crime scene is detected. The optimal time to make a decision depends on two antagonist notions: accuracy and earliness. For this scenario, warning the police too early would induce a cost if the scene is actually not a crime scene, but waiting too long would prevent the police from stepping in the crime scene. It hence becomes desirable to allow end-users to set a trade-off between earliness and accuracy, which is the goal of cost-aware early classification of time series

Recently, Dachraoui *et al.* [3] introduced a framework for cost-aware early classification of time series that is dedicated to optimizing a trade-off between the accuracy of the prediction and the time at which it is performed. This framework hence involves both costs defined above: a misclassification cost and a cost of delaying the prediction. The authors of [3] derive an algorithm that decides, at test time, whether the decision should be made at a given time instant t or if more data should be waited for (based on both costs defined above). In addition, the method of [3] has two other interesting properties : it is adaptive and non-myopic. It is adaptive in the sense that the optimal time of decision depends on the time series to be classified. It is non-myopic because at every time instant t, the algorithm not only decides if t is the optimal time instant to classify, but it also estimates when this optimal time is likely to happen, if not at time t. Such a property is essential since it can help end users of these systems to get prepared for action when the decision should soon be made (*e.g.* for medical applications).

In this paper, we propose two new early-classification schemes built upon the framework defined in [3]. This framework is based on approximating an expected cost through clustering, which tends to bring vagueness in the process. Our schemes improve on this existing work by removing the clustering step. They optimize a trade-off between a misclassification cost and a cost of delaying the decision, they are adaptive and non-myopic.

The rest of this paper is organized as follows. Section 2 reviews the related work on early classification of time series. Section 3 presents both proposed approaches and how they differ from [3]. Experiments on real time series data sets are presented in Section 4. Notations used throughout this paper are summarized in Table 1.

2 Related work

A wide range of methods tackle the early classification problem without explicitly accounting for the cost of delaying the decision [1, 4, 5, 6, 7, 10, 13, 14]. These methods differ in (i) the design of the early classifiers and (ii) the estimation of the optimal time to make a decision. They mostly wait for sufficient

Notation	Description
$\mathbf{x} = (x_1, \ldots, x_T)$	The time series to be classified
$\mathbf{x_t} = (x_1, \ldots, x_t)$	Time series $\mathbf{x}$ truncated after t observations, $t \leq T$
$\tau(\mathbf{x})$	Time index at which early classification of $\mathbf{x}$ is performed, also referred to as classification time
$\tau^*(\mathbf{x})$	Optimal time index for early classification of $\mathbf{x}$
$\hat{\tau}(\mathbf{x_t})$	Time index (greater than current time t) at which classification is most likely to occur (for non-myopic methods only)
$y \in \mathcal{Y}$	Class label of time series $\mathbf{x}$
$\mathcal{T} = \{(\mathbf{x^i}, y^i)\}_{1 \leq i \leq N}$	A training set of N time series and their labels
$\mathcal{T}_t = \{(\mathbf{x_t^i}, y^i)\}_{1 \leq i \leq N}$	A training set of N truncated time series and their labels
$C_m(\hat{y}, y)$	Cost of misclassifying a time series of class y into class $\hat{y}$
$C_d(t)$	Cost of delaying classification after t time instants
$C(\mathbf{x}, y)$	Overall early classification cost
$E_t(\mathbf{x})$	Expected cost of performing classification of time series $\mathbf{x}$ at time t
$\mathcal{H} = \{h_t\}_{1 \leq t \leq T}$	A set of classifiers for predicting the class of time series of length t
$\mathcal{D} = \{d_t\}_{1 \leq t \leq T}$	A set of classifiers for predicting whether early classification should be performed or not at time t (for *2Step* method only)
$\mathcal{M} = \{m_t\}_{1 \leq t \leq T}$	A set of regressors predicting when early classification is likely to happen (for *2Step* method only)

Table 1: Mathematical notations used throughout the paper.

confidence to make their decision, using varied procedures to define this confidence. The authors of [7] design an early classification scheme based on the boosting procedure for classification, where one weak classifier is built at each time stamp. This method is able to predict the class of a test time series at any time, but the issue of estimating the optimal time to make the decision is not addressed. Hatami *et al.* [5] propose an ensemble classifier with a reject option. A decision is made as soon as the agreement between all classifiers is above a certain threshold. Otherwise, the reject option is activated and more data is waited for. In [14], the authors rely on feature extraction for early classification. They first extract, from a training set of time series, a set of features called shapelets with a high discriminating power. Shapelets having high utility are then selected, where utility is an extension of the F-measure which also takes earliness into account. At test time, classification is performed as soon as one of the selected shapelets is found in the testing time series. Ghalwash *et al.* [4] extend this work so that it also provides an estimation of uncertainty associated with the decision. An alternative of these works designed for multivariate time series is proposed in [6]. Antonucci *et al.* [1] use Hidden Markov Model Classifiers with set valued parameters to determine whether the model is confident enough about its prediction (if not, several possible outputs are returned, which illustrates the uncertainty of the model). Early classification happens when there exists no ambiguity anymore about the prediction.

These methods do not attempt to infer whether future observations could improve classification accuracy. For difficult classification problems, they might tend to delay the decision even if future observations are unlikely to help. To bypass this limitation, Xing *et al.* [13] present a method that relies on neighborhood properties. For a given training time series $\mathbf{x^i} \in \mathcal{T}$, optimal prediction time $\tau(\mathbf{x^i})$ is set to the smallest τ^i such that the set of reverse nearest neighbors for $\mathbf{x^i}$ is stable for all $t \geq \tau^i$. At test time, if a time series has $\mathbf{x^i_t}$ as nearest neighbor after $t \geq \tau(\mathbf{x^i})$ time steps, classification is performed at this point. To improve on this setting, clustering can be performed on training time series so that stability can be robustly evaluated at the cluster level. Parrish *et al.* [10] design a probabilistic approach based on Quadratic Discriminant Analysis which aims at maximizing the *reliabilty*, *i.e.* the probability that the early decision leads to the same classification result as the classification of the complete time series.

One drawback of these methods is that they do not explicitly optimize a trade-off between earliness and accuracy, hence leading to sub-optimal solutions in this regard. In order to overcome this limitation, Dachraoui *et al.* [3] propose a framework for early classification, where the cost of delaying the decision is included in the optimization function. This framework considers a time series classification task for which two cost functions are given:

- $C_m(\hat{y}, y)$ is the misclassification cost function, *i.e.* the cost of predicting class label $\hat{y}$ whereas the effective class label was y;
- $C_d(t)$ is the cost of delaying the decision up to time instant t.

In this framework, the cost of classifying a time series $\mathbf{x}$ of class y using a set $\mathcal{H} = \{h_t\}_{1 \leq t \leq T}$ of classifiers is given by:

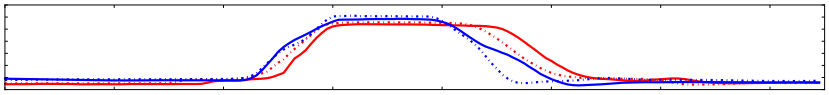
$$C(\mathbf{x}, y) = C_m(h_{\tau(\mathbf{x})}(\mathbf{x}), y) + C_d(\tau(\mathbf{x})), \tag{1}$$

where $\tau(\mathbf{x})$ is the time index at which classification is performed and $h_{\tau(\mathbf{x})}(\mathbf{x})$ is the class predicted at time $\tau(\mathbf{x})$ by the *ad hoc* classifier. To optimize on Eq. (1), the authors compute an expected cost for current and future time instants. If the current expected cost is lower than all future ones, classification is performed. This method is further discussed in Section 3.

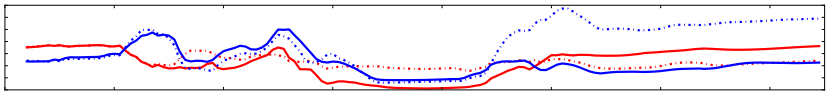
The authors of [9] also design an early classification scheme that takes the cost of delaying the decision into account. They define a stopping rule that depends on 4 parameters (to be fine tuned), on the time of the decision and on the posterior probabilities output by the classifiers. The optimal parameters for the stopping rule are learned through cross-validation so that they minimize a cost function that is similar in spirit to Eq. (1):

$$C'(\mathbf{x}, y) = \alpha \times C_m(h_{\tau(\mathbf{x})}(\mathbf{x}), y) + (1 - \alpha) \times C_d(\tau(\mathbf{x})), \tag{2}$$

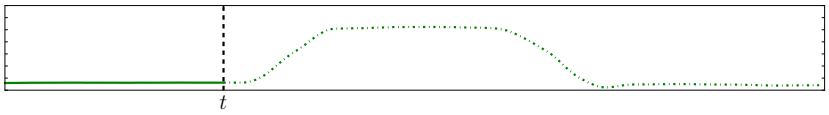
where α is a parameter that designs a trade-off between C_m and C_d. Unlike the method proposed in [3], the one from [9] is myopic in the sense that at a given time t, if the algorithm decides that it is better to wait to perform classification, it does not provide an estimate $\hat{\tau}(\mathbf{x_t}) > t$ of the optimal classification time.



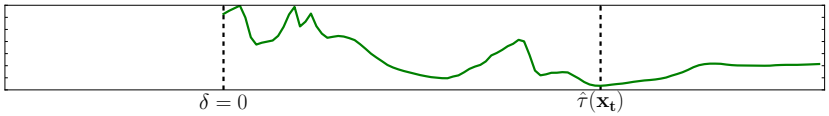
(a) Sample training time series



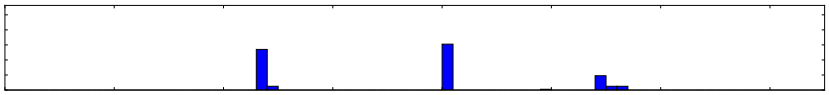
(b) Sample training expected cost functions



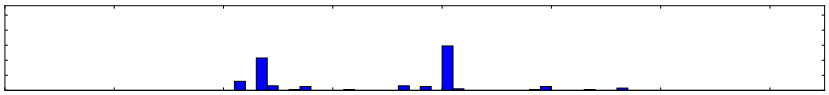
(c) Incoming test time series $\mathbf{x_t}$



(d) Expected cost $f_\delta(\mathbf{x_t})$ as computed by *NoCluster*



(e) Histogram of τ values using *NoCluster*



(f) Histogram of τ values using *2Step*

Fig. 1: Illustration of proposed methods on *Gun_Point* time series.

In this paper, we propose two non-myopic early classification methods taking into account the cost of delaying the decision. These methods are improvements over the framework defined in [3]. In the following section, we detail this framework and position the methods we propose with respect to it.

3 Proposed early classification schemes

The problem tackled in this paper is to estimate, for an incoming time series $\mathbf{x}$, the optimal time $\tau^*(\mathbf{x})$ at which the classification of $\mathbf{x}$ should be done in order to minimize the cost given by Eq. (1). In Section 3.1, we first give a detailed

review of the method proposed in [3] and its limitations. We then introduce an improved version of this approach from which we derive two early classification methods presented in Sections 3.2 and 3.3.

3.1 Computing expected costs for training time series

As an attempt to optimize on Eq. (1), Dachraoui *et al.* [3] intend to perform classification at a time t such that the expected cost:

$$f(\mathbf{x_t}) = \sum_{y \in \mathcal{Y}} P(y|\mathbf{x_t}) \sum_{\hat{y} \in \mathcal{Y}} P(\hat{y}|y, \mathbf{x_t})\, C_m(\hat{y}, y) + C_d(t) \tag{3}$$

is minimized. The term $P(y|\mathbf{x_t})$ is unknown by definition of the learning problem, which prevents us from computing this expected cost directly.

In order to approximate Eq. (3), authors rely on a clustering $\mathcal{C} = \{\mathbf{c^1}, \ldots, \mathbf{c^K}\}$ of the set $\mathcal{T}$ of complete training time series. At any given instant t, the expected cost of classifying $\mathbf{x}$ at future time instants $t + \delta$ (for $\delta \geq 0$) is computed as:

$$\begin{aligned} f_\delta(\mathbf{x_t}) &= \sum_{k=1}^{K} P(\mathbf{c^k}|\mathbf{x_t}) \sum_{y \in \mathcal{Y}} P(y|\mathbf{c^k}) \sum_{\hat{y} \in \mathcal{Y}} P_{t+\delta}(\hat{y}|y, \mathbf{c^k})\, C_m(\hat{y}, y) + C_d(t+\delta) \\ &= \sum_{k=1}^{K} P(\mathbf{c^k}|\mathbf{x_t})\, E_{t+\delta}(\mathbf{c^k}), \end{aligned} \tag{4}$$

where $E_{t+\delta}(\mathbf{c^k})$ is the expected cost of performing classification at time $t+\delta$ for series in cluster $\mathbf{c^k}$, that is computed as follows. At every time t, probabilities $P_t(\hat{y} \mid y, \mathbf{c^k})$ are estimated using cross-validation on the training set. Each of these probabilities correspond to a bin of the confusion matrix associated to cluster $\mathbf{c^k}$ at time t. The computation of the expected cost function $f_\delta(\mathbf{x_t})$ also depends on the probabilities $P(\mathbf{c^k} \mid \mathbf{x_t})$. In [3], these probabilities are computed as:

$$P(\mathbf{c^k}|\mathbf{x_t}) = \frac{s^k}{\sum_{i=1}^{K} s^i}, \text{ where } s^k = \frac{1}{1 + \exp^{-\lambda \Delta_t^k}}, \tag{5}$$

λ is some positive constant and $\Delta_t^k = (\bar{D}_t - d_t^k)/\bar{D}_t$ is the normalized difference between the average $\bar{D}_t$ of the distances between $\mathbf{x_t}$ and all the clusters, and the distance d_t^k between $\mathbf{x_t}$ and the cluster $\mathbf{c^k}$. We refer the interested reader to [3] for more details on these equations.

In other words, the expected cost function $f_\delta(\mathbf{x_t})$ for a time series to be classified is a weighted sum of the per-cluster expected cost functions $E_{t+\delta}(\mathbf{c^k})$ computed from the training set. Classification is finally performed at time t if, for any $\delta > 0$, $f_\delta(\mathbf{x_t}) \geq f_0(\mathbf{x_t})$.

This approach hence relies on the assumption that intra-cluster variability is sufficiently low for distance to centroid to be an acceptable proxy for the distance between time series. In practice, such an assumption is unlikely to hold. Furthermore, the clustering $\mathcal{C}$ is obtained using complete time series whereas

distances between $\mathbf{x_t}$ and the centroids are computed from incomplete series. This surely impacts the estimation of $\tau^*(\mathbf{x})$.

In order to overcome these two weaknesses, we propose to get rid of the clustering step used in [3]. Training expected costs are then computed on a per-series basis as:

$$\begin{aligned} E_t(\mathbf{x^i}) &= \sum_{y \in \mathcal{Y}} P(y|\mathbf{x^i}) \sum_{\hat{y} \in \mathcal{Y}} P_t(\hat{y} \,|\, \mathbf{x_t^i})\, C_m(\hat{y}, y^i) + C_d(t) \\ &= \sum_{\hat{y} \in \mathcal{Y}} P_t(\hat{y} \,|\, \mathbf{x_t^i})\, C_m(\hat{y}, y^i) + C_d(t) \end{aligned} \tag{6}$$

Indeed, as we do not tackle the multilabel case in this work, for every individual time series $\mathbf{x^i}$ (of class y^i) in the training set, we have $P(y \,|\, \mathbf{x^i}) = 1$ if $y = y^i$ and 0 otherwise, which removes the summation over y. Then, $P_t(\hat{y} \,|\, \mathbf{x_t^i})$ can be obtained from any classifier with probabilistic outputs.[1] To do so, we learn a set of classifiers $\mathcal{H}^{-i} = \{h_t^{-i}\}_{1 \leq t \leq T}$. In order to get reliable estimations of $P_t(\hat{y} \,|\, \mathbf{x_t^i})$, classifiers h_t^{-i} are built on subsets of $\mathcal{T}_t$ that do not contain $\mathbf{x_t^i}$. In practice, we use a standard cross-validation procedure to make sure that this condition is fulfilled. In this case, reliable estimation of $P_t(\hat{y} \,|\, \mathbf{x_t^i})$ is given by the probabilities output by h_t^{-i}.

An example of 4 sample time series together with their expected cost functions is given respectively in Fig. 1a and 1b.

3.2 NoCluster

Our first proposed method, which we denote *NoCluster*, consists in computing the expected cost for a test time series as a weighted sum of training expected costs calculated using Eq. (6). In other words, present and future expected costs for a time series to be classified are computed, at time t, as:

$$\forall \delta \geq 0, \;\; f_\delta(\mathbf{x_t}) = \sum_{i=1}^{N} \frac{1}{K_0} \frac{1}{1 + \exp^{-\lambda \Delta_t^i}} E_{t+\delta}(\mathbf{x^i}), \tag{7}$$

where $K_0 = \sum_{i=1}^{N} \frac{1}{1+\exp^{-\lambda \Delta_t^i}}$ and Δ_t^i is computed the way Δ_t^k is in Eq. (5), where distances to centroids are replaced by distances to training time series. As in [3], classification is performed at the smallest time t such that, for any $\delta > 0$, we get $f_\delta(\mathbf{x_t}) \geq f_0(\mathbf{x_t})$. This time instant is denoted $\tau(\mathbf{x_t})$. At test time, the Δ_t^i are computed for all times t up to $\tau(\mathbf{x})$, while when clustering was used, only K such normalized distance computations were required per time t. As this method is derived from [3], it preserves its non-myopia and adaptiveness properties.

Fig. 1c shows a sample test time series of length t, and Fig. 1d shows its associated expected cost from time t, and the estimation of the optimal decision time $\hat{\tau}(\mathbf{x_t})$. Fig. 1e depicts the histogram of τ values obtained by *NoCluster*

[1] Note that even when using classifiers that do not inherently compute probability estimates, cross validation can be used to get such estimates, as done in [2].

for all the test time series of the *Gun_Point* dataset. Note that peaks of this histogram (which also correspond to valleys of the expected cost presented in Fig. 1d) are related to regions of high interest in the time series for this dataset: a first increasing part of the time series, followed by a plateau and a final decrease. Algorithmic description of this method is provided in Algorithms 1 and 2.

Algorithm 1 Offline training of *NoCluster* method

Input: $\mathcal{T} = \{(\mathbf{x^i}, y^i)\}_{i \in \{1..N\}}$, $\{\mathcal{H}^{-i}\}_{i \in \{1..N\}}$
Output: $\{E_t(\mathbf{x^i})\}_{i \in \{1..N\}, t \in \{1..T\}}$
  for $i \in \{1..N\}$ **do**
    for $t \in \{1..T\}$ **do**
      Compute $P_t(\hat{y} \mid \mathbf{x_t^i})$ for all $\hat{y} \in \mathcal{Y}$ using h_t^{-i}
      Compute $E_t(\mathbf{x^i})$ using $P_t(\hat{y} \mid \mathbf{x_t^i})$ according to Eq. (6)
    end for
  end for
  return $\{E_t(\mathbf{x^i})\}_{i \in \{1..N\}, t \in \{1..T\}}$

Algorithm 2 Classification using *NoCluster* method

Input: $\{E_t(\mathbf{x^i})\}_{i \in \{1..N\}, t \in \{1..T\}}, \mathbf{x}, \mathcal{H} = \{h_t\}_{t \in \{1..T\}}, t_{\min}$
Output: $\hat{y}, \tau(\mathbf{x})$
  for $t \in \{t_{\min}..T\}$ **do**
    for $i \in \{1..N\}$ **do**
      $\Delta_t^i \leftarrow \|\mathbf{x_t} - \mathbf{x_t^i}\|_2$
    end for
    for $\delta \in \{0..T - t + 1\}$ **do**
      Compute $f_\delta(\mathbf{x_t})$ using Δ_t^i and $E_t(\mathbf{x^i})$, according to Eq. (7)
    end for
    if $\forall \delta \geq 0,\ f_0(\mathbf{x_t}) \leq f_\delta(\mathbf{x_t})$ **then**
      break
    end if
  end for
  return $h_t(\mathbf{x_t})$, t

3.3 2Step

As explained above, the computational complexity of *NoCluster* at test time is higher than that of the baseline, which can be a limitation for some applications. In this section, we present another early classification method that we call *2Step*. This method has a lower complexity at test time, at the cost of possibly higher training complexity. *2Step* also relies on the computation of the expected

Algorithm 3 Offline training of *2Step* method

Input: $\mathcal{T} = \{(\mathbf{x^i}, y^i)\}_{i\in\{1..N\}}$, $\{\mathcal{H}^{-i}\}_{i\in\{1..N\}}$
Output: $\mathcal{D} = \{d_t\}_{t\in\{1..T\}}$
  for $i \in \{1..N\}$ **do**
    for $t \in \{1..T\}$ **do**
      Compute $P_t(\hat{y} \,|\, \mathbf{x_t^i})$ for all $\hat{y} \in \mathcal{Y}$ using h_t^{-i}
      Compute $E_t(\mathbf{x^i})$ using $P_t(\hat{y} \,|\, \mathbf{x_t^i})$ according to Eq. (6)
    end for
    for $t \in \{1..T\}$ **do**
      Compute $\gamma_t(\mathbf{x_t^i})$ according to Eq. (8).
    end for
  end for
  for $t \in \{1..T\}$ **do**
    Learn classifier d_t with training set $\mathcal{T}_t$ and target variables $\{\gamma_t(\mathbf{x_t^i})\}_{i\in\{1..N\}}$
  end for
  return $\{d_t\}_{t\in\{1..T\}}$

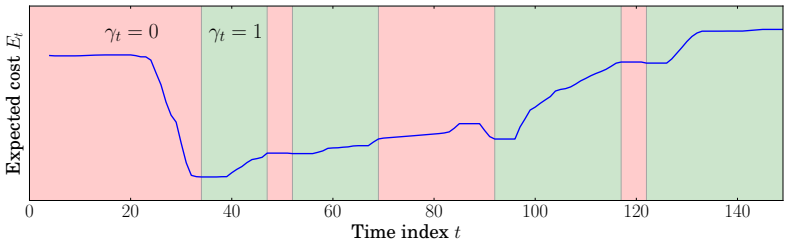


Fig. 2: Expected early classification cost for a training time series from the *Gun_Point* dataset, together with the corresponding γ_t values. For all time instants that are depicted in pink, $\gamma_t = 0$, and for time instants depicted in green, $\gamma_t = 1$. Best viewed in color.

costs for every training time series (Eq. (6)), but uses them in a different manner. A set of classifiers $\mathcal{D} = \{d_t\}_{1\leq t\leq T}$ is built as follows. The classifier d_t is built on the training set $\mathcal{T}_t$. Its target variable is not the class of the time series but a binary variable γ_t indicating if t is the expected optimal time to classify the time series. In other words,

$$\forall\, \mathbf{x_t^i} \in \mathcal{T}_t,\ \gamma_t(\mathbf{x_t^i}) = \begin{cases} 1 \text{ if } E_t(\mathbf{x^i}) = \min\limits_{\delta\geq 0} E_{t+\delta}(\mathbf{x^i}) \\ 0 \text{ otherwise.} \end{cases} \tag{8}$$

A graphical example showing how γ_t can be computed from the expected cost of classification of a time series is given in Fig. 2. In particular, this figure illustrates that γ_t is not monotonically increasing with t.

At test time, time series $\mathbf{x_t}$ is given to classifier d_t. If the output of d_t is 1, then $\tau(\mathbf{x}) = t$ and h_t is used to classify $\mathbf{x_t}$. Otherwise, the classification of $\mathbf{x_t}$ is delayed. Fig. 1f depicts the histogram of τ values obtained by *2Step* for all

the test time series of the *Gun_Point* dataset. One can note from Fig. 1e and 1f that both proposed methods are adaptive, in the sense that predicted timings τ vary across test time series. Algorithmic description of this method is provided in Algorithms 3 and 4.

Algorithm 4 Classification using *2Step* method

Input: $\mathcal{D} = \{d_t\}_{t\in\{1..T\}}, \mathcal{H} = \{h_t\}_{t\in\{1..T\}}, \mathbf{x}, t_{\min}$
Output: $\hat{y}, \tau(\mathbf{x})$
  for $t \in \{t_{\min}..T\}$ **do**
    if $d_t(\mathbf{x_t}) = 1$ **then**
      break
    end if
  end for
  return $h_t(\mathbf{x_t})$, t

The early classification method as presented in this section is myopic. In order to keep the non-myopic property, a set of regressors $\mathcal{M} = \{m_t\}_{1\leq t\leq T}$ should be learned in addition to the set $\mathcal{D}$ of classifiers. The target variable of regressor m_t should represent the time difference between t and the expected time of minimal cost, that is $\arg\min_{\delta\geq 0} (E_{t+\delta}(\mathbf{x_t}))$.

4 Experimental results

In this section, we design extensive experiments in order to evaluate the performance of both *NoCluster* and *2Step* early classification methods. The method presented in [3] is used as our main baseline, as (i) our proposed methods are built on top of it and (ii) all three methods share the desirable properties of non-myopia and adaptiveness. However, comparison with other state-of-the-art methods is also provided in Sections 4.4 and 4.5. The cost function defined in Eq. (1) (which defines a trade-off between accuracy and earliness of the decision) is used as our main performance indicator. Other criteria such as classification accuracy, earliness (expressed as the average prediction time $\bar{\tau}$) and timings are also considered.

4.1 Experimental setup

Experiments are conducted on all 76 datasets from the UCR archive [8] that have sufficient training data for cross-validation on the method parameters to be conducted (as a rule of thumb, we keep all datasets with at least 10 training time series per class). The complete list of datasets on which experiments are run is provided in the Supplementary material, which details per-dataset performance of the methods. Datasets from this archive cover a wide range of application domains. They are also diverse in terms of the number and the lengths of time

series in each dataset. The datasets are split into a training and a test set in the archive and we keep this separation in our experiments. For this set of experiments, the cost of delaying the decision is chosen linear in t:

$$C_d(t) = \beta \times t. \tag{9}$$

So as not to focus on a single trade-off between accuracy and earliness, we vary β in the set $\{0.0005, 0.001, 0.005, 0.01\}$ in all experiments. In real-world applications, β should be set according to expert knowledge and effective cost of waiting for more observations. For instance, in remote sensing applications where buying new images can be costly, high β values shall be used, unlike applications such as ECG monitoring for which new observations are made at almost no cost.

Results are reported for all setups (i.e., all datasets and β values) for which any of the compared methods finishes in less than 7 days. Presented results (for all criteria) are medians over 4 runs for each method.

Classifiers from sets $\mathcal{D}$ and $\mathcal{H}$ are linear Support Vector Machines trained using scikit-learn and LIBSVM Python binding [2, 11]. Parameters C from the SVM and λ from the expected cost functions are learned through cross-validation on the training set. Five values for parameter C are sampled regularly from a logarithmic scale between 10^{-1} and 10^{1}. Five values for λ are sampled in a similar way between 10^{1} and 10^{3}. As done in [3], we do not perform early classification before time $t_{\min} = 4$. For the baseline method, the number of clusters is chosen so as to maximize the silhouette factor [12], as suggested in [3].

Following the principle of reproducible research, the Python code used in these experiments (including both proposed method and our implementation of the baseline) is made publicly available for download.[2]

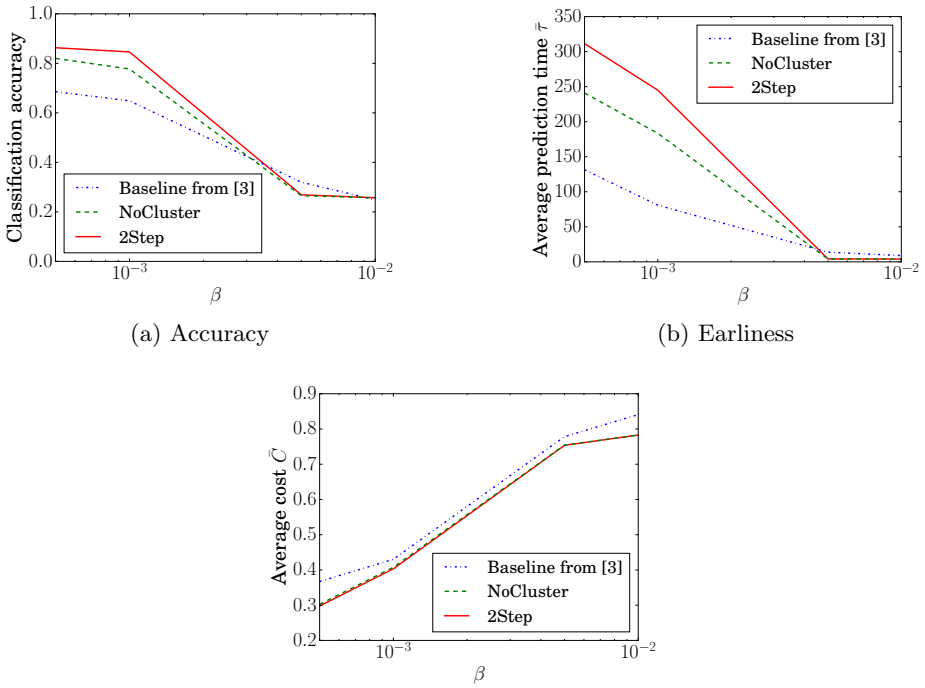
4.2 Sensitivity to β

Fig. 3 presents results obtained for dataset *FISH* in terms of accuracy, earliness and overall early classification cost function ($C(\mathbf{x}, y)$) for different values of β. As expected, when parameter β increases, accuracy and earliness both drop, for all methods. Indeed, high values of β indicate that it is more costly to wait for more data. Decision is hence made earlier, leading to worse accuracy. On the contrary, overall early classification costs increases with β, whatever the method. For this dataset (see Section 4.5 for more results) and for the whole range of β values considered here, *2Step* and *NoCluster* methods both outperform the baseline from [3] in terms of early classification cost, which is the indicator all three methods optimize on (remember that the lower cost, the better).

4.3 Timings

Fig. 4 presents both training and test timings of the two proposed methods using cumulative density functions (CDF), and compare them to the baseline method

[2] `https://github.com/rtavenar/CostAware_ECTS`



(a) Accuracy

(b) Earliness

(c) Early classification cost

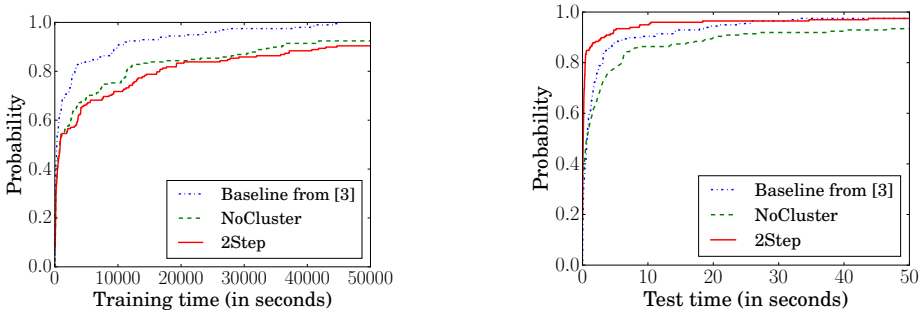
Fig. 3: Early classification performance as a function of β for dataset *FISH*.

from [3]. At a given probability level, a smaller execution time would correspond to a faster method, hence leftmost CDF curves are the most desirable. This figure shows that the baseline method runs faster than *NoCluster* for both training and test. The difference in test timings is due to the fact that *NoCluster* requires to compute more distances than the baseline, as explained in Section 3.2. For training timings, the difference is due to the number of expected cost function calculations that is lower for the baseline (one per cluster for the baseline versus one per training time series for *NoCluster*).

Concerning *2Step* method, training consists in computing the same expected cost functions as for *NoCluster* and then building classifiers on top of them, so it is expected that this method has higher training timings. Yet, it is important to notice that the use of classifiers has a positive impact on test timings, for which *2Step* even outperforms the baseline. As explained in Section 3, this feature can be important for some applications where decisions need to be made quickly.

4.4 Comparisons with classical early classification methods

We first compare *2Step* and *NoCluster* methods with classical early classification methods that do not directly optimize on a mixed cost function. To do so, we



(a) Training (b) Test

Fig. 4: Cumulative density functions of training (a) and test (b) times over all datasets and β values.

	EDSC [14]			ECTS1 [13]			ECTS2 [13]			RelClass1 [10]			RelClass2 [10]			RelClass3 [10]			RelClass4 [10]		
	W	T	L	W	T	L	W	T	L	W	T	L	W	T	L	W	T	L	W	T	L
2Step	49	0	21	52	0	21	51	0	22	54	0	19	56	0	17	57	0	16	54	0	19
NoCluster	54	0	17	56	0	19	56	0	19	56	0	19	58	0	17	58	0	17	57	0	18

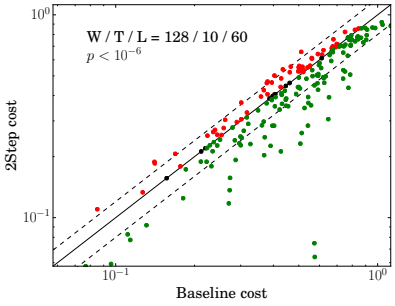
Table 2: Win / Tie / Lose scores based on early classification costs. Results from all datasets using all β values are used. For these scores, "Win" means that proposed method outperforms the baseline in terms of cost minimization.

use earliness and accuracy scores published in the supplementary material associated to [9]. Such scores are available for EDSC [14], strict and loose variants of ECTS [14] (denoted ECTS1 and ECTS2 respectively) and RelClass [10] with four different reliability thresholds (0.001, 0.1, 0.5, 0.9 for methods RelClass1 to RelClass4 respectively). Based on these scores, we can compute early classification costs for all methods with varying β values. Win/Tie/Lose scores computed from these costs are presented in Table 2. These results show that both proposed methods outperform all these baselines. To assess statistical significance, we perform one-sided Wilcoxon signed rank tests between each proposed method and each baseline. All tests show significance with p-values lower than 10^{-6}.

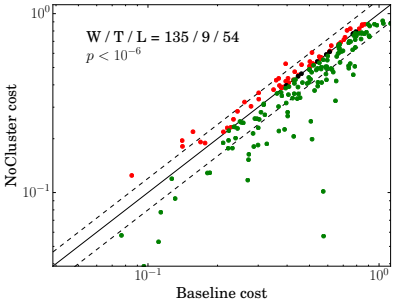
This performance improvement was expected, as baseline methods considered in this Section do not optimize on the cost function that mixes earliness and accuracy. To further evaluate performance of our proposed methods, we compare them to other cost-aware methods in the following Section.

4.5 Comparisons with cost-aware methods

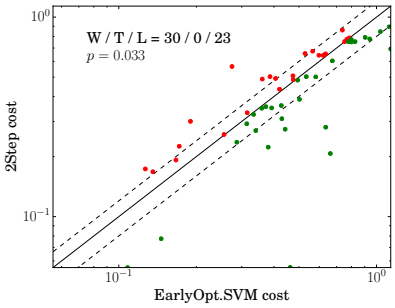
Fig. 5 presents comparisons of early classification costs between both our proposed methods and cost-aware baselines. Presented results show that proposed



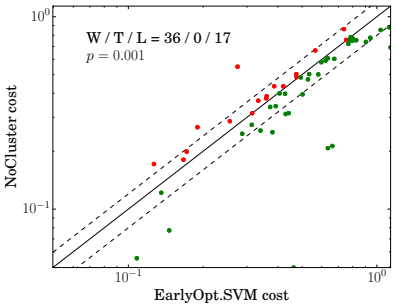
(a) *2Step* vs. Baseline from [3]



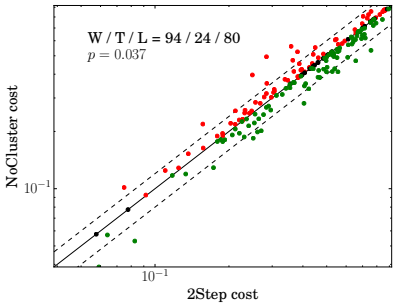
(b) *NoCluster* vs. Baseline from [3]



(c) *2Step* vs. EarlyOpt.SVM



(d) *NoCluster* vs. EarlyOpt.SVM



(e) *NoCluster* vs. *2Step*

Fig. 5: Comparison of early classification costs between proposed methods and the baselines from [3] and [9]. Results from all datasets using all β values are gathered here. Solid line indicates equal costs and dashed lines correspond to the +/-20% cost interval.

methods enable to reach lower early classification cost than the baseline from [3] in most cases: *2Step* outperforms the baseline in 128 out of the 198 cases (i.e., in 64.6% of the cases) , while *NoCluster* has better results in 135 cases (68.2%).

Moreover, the number of setups for which the baseline is improved by more than 20% is another indicator showing the benefit of both proposed methods (cf. points outside the dashed lines in Fig. 5a and 5b).

One-sided Wilcoxon signed rank tests between each proposed method and the baseline show significance with a p-value lower than 10^{-6}, which confirms that both our methods outperform the baseline in terms of cost minimization.

We then compare both proposed methods to EarlyOpt.SVM [9], that optimizes on a slightly different trade-off between accuracy and earliness, in Fig. 5 When doing so, evaluation can be performed with respect to two different cost functions: the one on which EarlyOpt.SVM optimizes or the one used for our methods. Fig. 5c and 5d present results obtained when using the cost function from EarlyOpt.SVM, that favors the latter in the comparison[3]. In spite of this, both proposed methods improve on EarlyOpt.SVM performance and these improvements are statistically significant at the 5% significance level.

Finally, when comparing *2Step* and *NoCluster* costs as shown in Fig. 5e, *NoCluster* gets slightly better results that are considered significant when tested at the 5% significance level ($p = 0.037$).

5 Conclusion

In this paper, we build upon the framework introduced in [3]. This framework deals with the problem of early classification from a new point of view: it aims at minimizing a cost function that includes a cost of misclassification and a cost of delaying the decision. Following this framework, our proposition consists in designing two different methods (called *NoCluster* and *2Step*) that optimize on such a cost function. These methods decide online for an incoming time series if the current time instant is the optimal moment to classify the time series (with respect to the cost function) or if it is more valuable to wait for more samples of the time series before classifying it. In the latter case, they also give an estimate of when the optimal moment is more likely to occur. Both methods are hence adaptive and non myopic. We have performed extensive experiments on a large and widely used set of datasets in order to evaluate the appropriateness of these methods, in terms of performance and timings. It turns out that (i) *2Step* outperforms the state-of-the art in terms of both cost minimization and classification time and (ii) *NoCluster* is an even better option in terms of early classification cost at the expense of slower classification. As future work, we will consider the use of time-series specific classifiers, which should improve the overall performance of these approaches.

Acknowledgements

This work has been partly funded by ANR project ASTERIX (ANR-13-JS02-0005-01) and CNES-TOSCA project VEGIDAR. Authors would like to thank Antoine Cornuéjols for his insight on the state-of-the-art, as well as data donors.

[3] See Supplementary material for more details on this comparison.

References

[1] Alessandro Antonucci et al. "Early Classification of Time Series by Hidden Markov Models with Set-Valued Parameters". In: *Proceedings of the NIPS Time Series Workshop*. 2015.

[2] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 27:1–27:27.

[3] Asma Dachraoui, Alexis Bondu, and Antoine Cornuéjols. "Early Classification of Time Series as a Non Myopic Sequential Decision Making Problem". In: *Machine Learning and Knowledge Discovery in Databases*. 2015, pp. 433–447.

[4] Mohamed F. Ghalwash, Vladan Radosavljevic, and Zoran Obradovic. "Utilizing Temporal Patterns for Estimating Uncertainty in Interpretable Early Decision Making". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2014, pp. 402–411.

[5] Nima Hatami and Camelia Chira. "Classifiers With a Reject Option for Early Time-Series Classification". In: *Proceedings of the IEEE Symposium on Computational Intelligence and Ensemble Learning*. 2013.

[6] Guoliang He et al. "Early classification on multivariate time series". In: *Neurocomputing* 149, Part B (2015), pp. 777–787.

[7] Katsuhiko Ishiguro, Hiroshi Sawada, and Hitoshi Sakano. "Multi-class Boosting for Early Classification of Sequences". In: *Proceedings of the British Machine Vision Conference*. 2010, pp. 24.1–10.

[8] Eamonn Keogh et al. *The UCR Time Series Classification/Clustering Homepage*. `http://www.cs.ucr.edu/~eamonn/time_series_data/`. 2011.

[9] Usue Mori et al. "Early classification of time series from a cost minimization point of view". In: *Proceedings of the NIPS Time Series Workshop*. 2015.

[10] Nathan Parrish et al. "Classifying With Confidence From Incomplete Information". In: *Journal of Machine Learning Research* 14 (2013), pp. 3561–3589.

[11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[12] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[13] Zhengzheng Xing, Jian Pei, and Philip S. Yu. "Early Classification on Time Series". In: *Knowledge And Information Systems* 31.1 (2012), pp. 105–127.

[14] Zhengzheng Xing et al. "Extracting Interpretable Features for Early Classification on Time Series". In: *Proceedings of the SIAM International Conference on Data Mining*. 2011, pp. 247–258.