A Novel Algorithm for Online Classification of Time Series when Delaying Decision is Costly

Asma Dachraoui[12], Antoine Cornuéjols[2], et Alexis Bondu[1]

[1]EDF Lab, Paris-Saclay (France)
[2]AgroParisTech, département MMIP et INRA UMR-518
16, rue Claude Bernard, F-75231 Paris Cedex 5 (France)

July 2, 2016

Abstract

Aiding to make decisions as early as possible by learning from past experiences is becoming increasingly important in many application domains. In these settings, information can be gained by waiting for more evidences to arrive, thus helping to make better decisions that incur lower misclassification costs, but, meanwhile, the cost associated with delaying the decision generally increases, rendering the decision less attractive. Learning requires then to solve an optimization problem combining two types of competing costs.

In the growing literature on online decision making, very few works have explicitly incorporated the cost of delay in the decision procedure. One recent work [DBC15] has introduced a general formalization of this optimization problem. However, the algorithm presented there to solve it is based on a clustering step, with all the attendant necessary choices of parameters that can heavily impact the results. In this paper, we adopt the same conceptual framework but we present a more direct technique involving only one parameter and lower computational demands. Extensive experimental comparisons between the two methods on synthetic and real data sets show the superiority of our method when the classification of the incomplete time series is difficult, which corresponds to a large fraction of the applications.

Keywords: Early classification of time series, Cost estimation, Sequential decision making.

1 Introduction

There exists nowadays an increasing awareness of the importance of learning in support of online decision-making. In emergency wards of hospitals, in control rooms of national or international electrical power grids, in government councils assessing emergency situations, in all kinds of contexts, it is essential to make timely decisions in absence of complete knowledge of the true outcome. The issue facing the decision makers is that, usually, the longer the decision is delayed, the clearer is the likely outcome (e.g. should the patient undergo a risky surgical operation), but, also, the higher the cost that will be incurred if only because earlier decisions allow one to be better prepared.

This is a classical optimization problem with a trade-off between the gain of information that can be expected if one delays the decision, and the rising cost of such a delay. It has historical roots in fields such as *sequential decision making* and *optimal statistical decisions*[DeG05, Ber85]. One technique especially has gained a wide exposition: Wald's Sequential Probability Ratio Test [WW48]. The task is to classify a sequence of measurements $\mathbf{x}_t^i$ into one of two possible classes -1 or $+1$. The likelihood ratio $R_t = \frac{P(\langle x_1^i,\ldots,x_t^i\rangle \mid y=-1)}{P(\langle x_1^i,\ldots,x_t^i\rangle \mid y=+1)}$ is computed and compared with two thresholds set according to the required error of the first kind α (*false positive error*) and error of the second kind β (*false negative error*). One difficulty lies in the estimation of the conditional probabilities $P(\langle x_1^i,\ldots,x_t^i\rangle \mid y)$. And, furthermore, there is no explicit reference to the cost of delaying the decision in the choice of α and β.

These limitations are shared with many modern techniques that seek to classify incomplete sequences. Most of them define some confidence metrics in order to assess the degree of certainty of the decision if it was taken instantly, but they do not take into account the cost associated with the delay before deciding (see for instance [ISS00, XPP09, XPP09, APTG12, PAGH13, HC13, XPPW11]). In addition, these techniques implement a myopic strategy by which they examine at each new time step if a decision should be taken now, or if it seems better to wait one more time step. They do not try to look beyond the current state and estimate what future instant could yield the best trade-off between the quality of the decision and its cost.

In a recent paper [DBC15], a generic framework in which to cast the optimization of quality of decision and delaying cost was formalized. This work supposes that there exists a set $\mathcal{S}$ of m training sequences, each being a couple $(\mathbf{x}_T^i, y^i) \in \mathbb{R}^T \times \mathcal{Y}$, meaning that it is composed of T real valued measurements $\langle x_1^i, \ldots, x_T^i \rangle$, and an associated label $y^i \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite set of classes. For instance, $\langle x_1^i, \ldots, x_T^i \rangle$ could be a sequence of measurements of the arterial tension of a patient in an hospital. After learning from $\mathcal{S}$ has taken place, the goal is to classify each new incoming sequence $\mathbf{x}$ (e.g. sequence of arterial tensions on a period of time about a new patient) with as low a cost as possible, combining the cost of misclassification and the cost of delaying decision so far.

The set $\mathcal{S}$ of training sequences provides two types of information that can be learned from. First, one can extract typical patterns or evolutions in the sequences, allowing one, in the predicting phase, to try to extrapolate the most likely continuation of a new incomplete sequence. Second, it is possible to learn classifiers for each time step: h_t, that associates to incomplete sequences $\mathbf{x}_t = \langle x_1, \ldots, x_t \rangle$ a label $y \in \mathcal{Y}$.

From this, it becomes possible to express the expected cost of a decision after t time steps (t measurements) as:

$$f(\mathbf{x}_t) \;=\; \sum_{y \in \mathcal{Y}} P(y|\mathbf{x}_t) \sum_{\hat{y} \in \mathcal{Y}} P(\hat{y}|y, \mathbf{x}_t)\, \mathrm{C}_t(\hat{y}|y) \;+\; \mathrm{C}(t) \tag{1}$$

where $\mathrm{C}_t(\hat{y}|y) : \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}$ is the *misclassification cost function* that defines the cost at time t of predicting $\hat{y}$ when the true class is y, $P(\hat{y}|y, \mathbf{x}_t)$ is the probability that h_t, the learned decision function at time t, classifies $\mathbf{x}_t$ as $\hat{y} = h_t(\mathbf{x}_t)$ when it was really of the class y, and $\mathrm{C}(t)$ is the *time cost function* which is non decreasing over time. If this cost is computed for all time steps $t \in \{1, \ldots, T\}$, the optimal time t^* for the decision problem is defined as:

$$t^* \;=\; \underset{t \in \{1, \ldots, T\}}{\mathrm{ArgMin}}\; f(\mathbf{x}_t) \tag{2}$$

However, this formulation of the problem does not readily yield a method for finding, online, the optimal decision time. First, it would require to compute all the decision costs until time T before knowing what is t^*, clearly defeating the purpose of the approach. Second, the terms $P(y|\mathbf{x}_t)$ and $P(\hat{y}|y, \mathbf{x}_t)$ are difficult to estimate on a single sequence. This requires that some generalization over the space of possible sequences takes place.

In [DBC15], the authors presented an intuitively alluring solution to these problems.

First of all, they propose to capture typical evolutions of the sequences $\mathbf{x}_T$ using a clustering technique. They thus end up with K clusters $\mathfrak{c}_k$, $(1 \leq k \leq K)$, and equation (1) thus becomes:

$$f(\mathbf{x}_t) = \sum_{\mathfrak{c}_k \in \mathcal{C}} P(\mathfrak{c}_k|\mathbf{x}_t) \sum_{y \in \mathcal{Y}} P(y|\mathfrak{c}_k) \sum_{\hat{y} \in \mathcal{Y}} P_t(\hat{y}|y, \mathfrak{c}_k) \mathrm{C}(\hat{y}|y) + \mathrm{C}(t) \tag{3}$$

replacing the hard to compute conditional probabilities $P(y|\mathbf{x}_t)$ and $P(\hat{y}|y, \mathbf{x}_t)$ by the more easily available $P(y|\mathfrak{c}_k)$ and $P_t(\hat{y}|y, \mathfrak{c}_k)$.

Namely, they use a specific distance function between a incoming incomplete sequence and a cluster that allow them to compute the term $P(\mathfrak{c}_k|\mathbf{x}_t)$. The probabilities $P(y|\mathfrak{c}_k)$ are easy to compute from the training set. Finally, a classifier h_t is learned for each time step, allowing to estimate $P_t(\hat{y}|y, \mathfrak{c}_k)$, the terms of the confusion matrix associated with h_t when the sequences are recognized as belonging to each cluster $\mathfrak{c}_k$.

The second idea they introduce in order to overcome the necessity to compute $f(\mathbf{x}_t)$ for all $t \in \{1, \ldots, T\}$ is to compute *in advance*, at time t, the expected costs of decision for all future time steps. This is possible since, given an incomplete sequence $\mathbf{x}_t$, its membership to each cluster $\mathfrak{c}_k$ can be estimated, which provides information about potential futures.
Given a time step t, the expected decision cost for any future instant $t + \tau$ is:

$$f_\tau(\mathbf{x}_t) = \sum_{\mathfrak{c}_k \in \mathcal{C}} P(\mathfrak{c}_k|\mathbf{x}_t) \sum_{y \in \mathcal{Y}} P(y|\mathfrak{c}_k) \sum_{\hat{y} \in \mathcal{Y}} P_{t+\tau}(\hat{y}|y, \mathfrak{c}_k) \mathrm{C}(\hat{y}|y) + \mathrm{C}(t + \tau) \tag{4}$$

yielding the expected best future time for decision as:

$$\tau^* = \underset{\tau \in \{0, \ldots, T-t\}}{\text{ArgMin}} \; f_\tau(\mathbf{x}_t) \tag{5}$$

The proposed algorithm is then very simple. Observing an incoming sequence $\mathbf{x}_t = \langle x_1, \ldots, x_t \rangle$, the expected cost of decision is estimated for all future decision time using equations (4) and (5). If t is equal to the estimated optimal decision time t^* ($\tau = 0$), the procedure stops and a prediction $h_t(\mathbf{x}_t)$ is made[1]. Otherwise, the algorithm waits for an additional measurement x_{t+1}, unless $t = T$.

This approach yields experimental results (see [DBC15]) that remarkably agree with what one would expect from an early decision system, viz:

- The time of decision rises when the classification task is increasingly harder (for instance, if the data is increasingly noisy), up until a point when, given the difficulty of extracting information from the signal, the system knows from its experience that classification gains in the future cannot overcome the delaying cost, thus deciding that it is not worth waiting and it is better to make decision at the first possible moment even though the quality of this decision might be quite low.
- The time of decision decreases when the delay cost function increases more rapidly with time.

However, the method requires the user to make a set of choices that may be baffling. First, the choice of a clustering method with all the attending choices of parameters (e.g. distances, parameters of the method, number of clusters, ...). Second the choice of a distance between an incomplete sequence $\mathbf{x}_t$ and a cluster $\mathfrak{c}_k$ made of complete sequences. And, third, the choice of a membership function in order to compute $P(\mathfrak{c}_k|\mathbf{x}_t)$. All these choices can be tricky to make and can entail non negligible variations in the results.

In this paper, while retaining the overall framework presented in [DBC15], we introduce a competing method which avoids the burdens associated with the clustering approach. In particular, the new method uses a segmentation of the training set which is much more direct and simple, yielding more robust results with lower computational demands.

The novel algorithm is presented in Section 2 below. In order to test its properties, a set of experiments has been devised, which includes controlled experiments that allow one to check the validity of any early classification technique in a wide set of conditions, as well as experiments on real data sets. This is described in Section 3 and Section 4 where the empirical results and findings both for the method presented in [DBC15] and for the new approach are reported. The conclusion, in Section5, underlines the pros and cons of the two competing methods and highlights the domains in which each of them is better suited.

[1]Alternatively, a prediction $h_t(\mathbf{x}_t)$ can be made by combining all the predictions $h_t(\mathbf{x}_t)$ weighted by the estimated membership of $\mathbf{x}_t$ to all clusters $\mathfrak{c}_k$.

2 A new algorithm

Aside the difficulties inherent in the use of a clustering method, specially over sequences, there is another aspect that can make the approach described in [DBC15] less than optimal. Indeed, from equation (3), repeated below:

$$f(\mathbf{x}_t) = \sum_{\mathfrak{c}_k \in \mathcal{C}} P(\mathfrak{c}_k|\mathbf{x}_t) \sum_{y \in \mathcal{Y}} P(y|\mathfrak{c}_k) \sum_{\hat{y} \in \mathcal{Y}} P_t(\hat{y}|y, \mathfrak{c}_k)\, \mathrm{C}(\hat{y}|y) + \mathrm{C}(t)$$

it is apparent that the membership of $\mathbf{x}_t$ to a cluster $\mathfrak{c}_k$ is important only insofar that the associated confusion matrices, given by $P_t(\hat{y}|y, \mathfrak{c}_k)$, are different from one cluster to the other. Otherwise, there is no point in considering $P(\mathfrak{c}_k|\mathbf{x}_t)$, that is to which cluster belongs the incoming sequence. In addition, the conditional probabilities $P(y|\mathfrak{c}_k)$ should be as non uniform as possible.

If one, then, is considering an alternative way of segmenting the set of sequences, it should better lead to confusion matrices that differ as widely as possible from one category to another. It should also lead to terms alike $P(y|\mathfrak{c}_k)$ as different as possible.

It is not easy to devise directly such a segmentation of the sequences, but there exists an approach that naturally favors these properties. The idea is to use the confidence level of the prediction $h_t(\mathbf{x}_t)$ to make that segmentation. (Remark: The following directly applies to the binary classification case, it is more involved, but possible to extend it to the multiple classification case).

Most binary classifiers, like neural networks, SVM, Naïve Bayes, decision trees, a.s.o. can easily be made to output a real number $g(\mathbf{x}_t)$ in the range $[0, 1]$ such that $h_t(\mathbf{x}_t) = -1$ if $g(\mathbf{x}_t) \leq 0.5$ and $h_t(\mathbf{x}_t) = +1$ otherwise (the threshold 0.5 depends on the calibration of the function g_t). The value $g(\mathbf{x}_t)$ can be interpreted as expressing a confidence level in the prediction of the class to which belongs $\mathbf{x}_t$, and when some care is taken over the choice of the loss function used to learn g, $g(\mathbf{x}_t)$ can even be interpreted as a probability to belong to class +1. (See [Pla99] for instance, that shows how to associate a confidence level to the prediction of a SVM).

What is interesting is that the confusion matrices over examples that are predicted with a confidence level close to 1 and over examples that are predicted with a confidence level close to 0.5 are generally quite different, which is natural if the confidence level $g(\mathbf{x}_t)$ somewhat reflects the probability that the class +1 has been predicted for $\mathbf{x}_t$. Hence, the idea to use confidence intervals to differentiate classes of sequences.

A sequence $\mathbf{x}_t$ will thus be recoded as a sequence $\langle \gamma_1, \ldots, \gamma_t \rangle$ of confidence intervals with each $\gamma_{s(1 \leq s \leq t)} \in \{1, N\}$ one of the N intervals of confidence. N is the only parameter in the method, in addition to the size of the memory of the past taken into account, as is explained below.

Therefore the segmentation method used in our algorithm is the following. For each time step t, a function g_t is learned using the training set $\mathcal{S}$ (and hence a decision function $h_t(\cdot) = \text{sign}\big(g_t(\cdot) - 0.5\big)$). More specifically, g_t is learnt using the training sequences in $\mathcal{S}$ reduced to their first t components $\langle x_1^i, \ldots, x_t^i \rangle$ and their label y^i.

Then, a discretization of the confidence interval $[-1, +1]$ for each time step is learnt. For each time step t, the associated function g_t induces an ordering of the sequences in $\mathcal{S}$ from the sequence with the highest confidence value $g_t(\mathbf{x}_t)$, to the sequence with the lowest one. It is easy to compute a set of $N-1$ thresholds on $[0, 1]$ for time step t such that each of the induced N sub-intervals is associated with approximately $|\mathcal{S}|/N$ sequences. This segmentation method automatically corrects any bias in the calibration of g_t.

Given this discretization scheme, resulting in varying discretization thresholds depending on the time steps $t \in \{1, \ldots, T\}$, each sequence $\mathbf{x}_t$ can be recoded as a sequence of t confidence intervals γ_t where $\gamma_t = \ell$ if $g_t(\mathbf{x}_t)$ is in the sub-interval corresponding to $\ell \in \{1, \ldots, N\}$. (See Figure 1).

This recoding provides a way to compute the likely future outlines of a given incomplete sequence $\mathbf{x}_t$. Given the code $\langle \gamma_1, \ldots, \gamma_t \rangle$ of a sequence $\mathbf{x}_t$, the objective is to compute the probability for each future time step $t+s$, $(1 \leq s < T-t)$, that $\gamma_{t+s} = \ell$ with $\ell \in \{1, \ldots, N\}$. Let us note $\overrightarrow{\gamma}_{t+s}$ the vector made of the N corresponding values: $[p(\gamma_{t+s} = 1), \ldots, p(\gamma_{t+s} = N)]^\top$. Then, in all generality, we want to compute $\langle \overrightarrow{\gamma}_{t+1}, \ldots, \overrightarrow{\gamma}_T \rangle | \langle \gamma_1, \ldots, \gamma_t \rangle$.

This would entail learning a dependency matrix of $(T-t) \times t$ values, and these dependency matrices should be learned for all possible $t \in \{1, T-1\}$. The number of possible sequences in the code is N^T, and it is required to estimate the probability of each one of them. With $N = 5$ and $T = 100$, limiting sequences to 100 time steps, this is already approximately 7×10^{72} numbers to estimate. This is why we introduce a Markov condition, namely that only $\overrightarrow{\gamma}_{t+1}$ will be computed given as input the coded sequence: $\overrightarrow{\gamma}_{t+1} | \langle \gamma_1, \ldots, \gamma_t \rangle$, and all probability vectors after time $t+1$, i.e. $\overrightarrow{\gamma}_{t+s}$ with $1 \leq s \leq T-1$ are computed using a one order dependency: $\overrightarrow{\gamma}_{t+s+1} | \overrightarrow{\gamma}_{t+s}$.

In words, within this hypothesis, only the first future time step $\overrightarrow{\gamma}_{t+1}$ is estimated given the whole past history of $\mathbf{x}_t$ coded as $\langle \gamma_1, \ldots, \gamma_t \rangle$, and thereupon all future time steps are supposed to depend only on the previous one.

Let us note $\mathbf{M}_t^{t+1}$ the $N \times N$ transition matrix from time step t to time step $t+1$ with elements $m_{u,v}^t = p(\gamma_{t+1} = v | \gamma_t = u)$, where u, v are confidence intervals at time steps t and $t+1$ respectively, that is $u, v \in \{1, \ldots, N\}^2$. With $N = 5$ different confidence intervals, the transition matrices have each 25 elements.

Let us note $\mathbf{M}_{t,\ldots,t-\delta}^{t+1}$ the $(t \times N) \times N$ transition matrix from a coded sequence $\langle \gamma_{t-\delta}, \ldots, \gamma_t \rangle$ represented as $t \times N$ probability vector $\mathbf{P}_{\langle \gamma_1, \ldots, \gamma_t \rangle}$ to the probability vector $\overrightarrow{\gamma}_{t+1}$. For illustration, suppose that $N = 5$ and we only look at a length 2 sequence coded as $\langle \gamma_1 = 2, \gamma_2 = 4 \rangle$. Then the corresponding probability vector has 10 components: $\mathbf{P}_{\langle \gamma_1, \gamma_2 \rangle} = [0, 1, 0, 0, 0, 0, 0, 1, 0]^\top$.

Using these notations, and given an input coded sequence $\langle \gamma_1, \ldots, \gamma_t \rangle$, one can estimate the future probability vector using equation:

$$\overrightarrow{\gamma}_{t+\tau} | \langle \gamma_1, \ldots, \gamma_t \rangle \; = \; \left[\prod_{s=1}^{\tau-1} \mathbf{M}_{t+s}^{t+s+1} \right] \mathbf{M}_{0,\ldots,t}^{t} \; \mathbf{P}_{\langle \gamma_1, \ldots, \gamma_t \rangle} \tag{6}$$

In words, given the past coded history $\langle \gamma_1, \ldots, \gamma_t \rangle$, one computes the next probability vector $\overrightarrow{\gamma}_{t+1}$ using $\mathbf{M}_{0,\ldots,t}^{t} \; \mathbf{P}_{\langle \gamma_1, \ldots, \gamma_t \rangle}$, and then, the probability vector at horizon $t+\tau$ is computed thanks to a product of one order transition matrices $\prod_{s=1}^{\tau-1} \mathbf{M}_{t+s}^{t+s+1}$.

However, even this simplified scheme necessitates to learn large transition matrices $\mathbf{M}_{0,\ldots,t}^{t}$ with N^t elements to be learnt, and this for all possible values of $t \in \{1, \ldots, T-1\}$, which amounts to $\frac{N^{T+1}-1}{N-1}$ probability values to be estimated, which, for $N = 5$ and $T = 100$ gives approximately 10^{71} values to learn.

Suppose then that only the last δ codes $\langle \gamma_{t-\delta+1}, \ldots, \gamma_t \rangle$ of an input sequence $\mathbf{x}_t$ be used

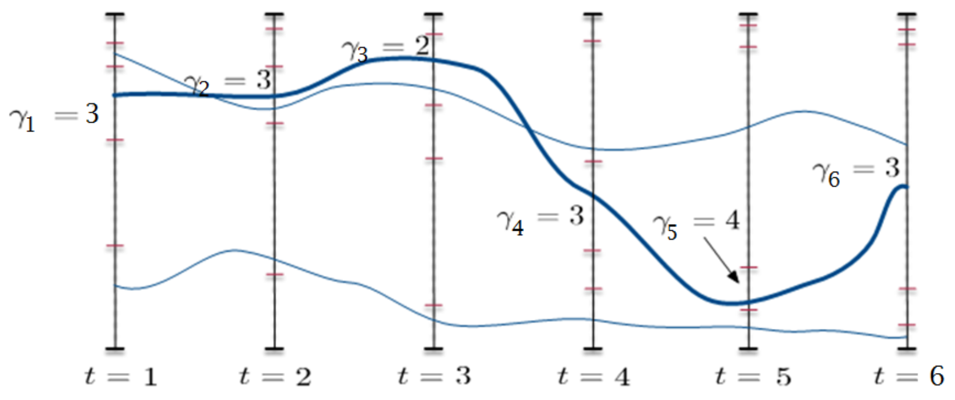

Figure 1: How sequences are coded using confidence intervals. Here, the thick curve is coded as $\langle \gamma_1 = 3, \gamma_2 = 3, \gamma_3 = 2, \gamma_4 = 3, \gamma_5 = 4, \gamma_6 = 3 \rangle$. Actually, the sequences here depicted as curves in order to better visualize them are sequence of points $\langle x_1, x_2, x_3, x_4, x_5, x_6 \rangle$ with no curves in between. The confidence intervals vary from one time step to another as explained in the text.

to compute the future states, we get the following expression:

$$\overrightarrow{\gamma}_{t+\tau} | \langle \gamma_{t-\delta+1}, \ldots, \gamma_t \rangle \; = \; \left[\prod_{s=1}^{\tau-1} \mathbf{M}_{t+s}^{t+s+1} \right] \mathbf{M}_{t-\delta+1,\ldots,t}^{t} \; \mathbf{P}_{\langle \gamma_{t-\delta+1},\ldots,\gamma_t \rangle} \qquad (7)$$

where $\mathbf{M}_{t-\delta+1,\ldots,t}^{t}$ which requires $\mathcal{O}(N^{\delta})$ probability values to be learned. Even with $\delta = 2$, and $N = 5$, $5^3 = 125$ probabilities must be estimated, and this for all values of t.

In the experiments reported below, and because we had only a few thousands training time sequences, we have radically simplified the approach and used a one order memory, yielding the equation:

$$\overrightarrow{\gamma}_{t+\tau} | \langle \gamma_t \rangle \; = \; \left[\prod_{s=1}^{\tau-1} \mathbf{M}_{t+s}^{t+s+1} \right] \mathbf{P}_{\langle \gamma_t \rangle} \qquad (8)$$

which computes the vector $\overrightarrow{\gamma}_{t+\tau} = [\gamma_{t+\tau} = \ell]_{1 \le \ell \le N}^{\top}$.

This requires only the estimation of $T \times N^2$ probability elements using the training set. This first order Markov model provides a baseline with which to assess the minimal capacity of the method.

We can now return to the computation of the expected decision cost for future time steps:

$$f_\tau(\mathbf{x}_t) \; = \; \sum_{y, \hat{y} \in \mathcal{Y}} \sum_{\ell=1}^{N} (\gamma_{t+\tau} = \ell | \langle \gamma_t \rangle) \; P_{t+\tau}(\hat{y} | y, \gamma_{t+\tau} = \ell) \times \mathrm{C}(\hat{y} | y, \gamma_{t+\tau} = \ell) + \mathrm{C}(t + \tau) \qquad (9)$$

Using equation (9), one obtains an estimation of the optimal decision time to come: $t^* = t + \mathrm{ArgMin}_{\tau \in \{0,\ldots,T-t\}} \, f_\tau(\mathbf{x}_t)$.

The whole method can be described by two algorithms. One for learning from a set $\mathcal{S}$ of training sequences (see Algorithm 1), and one for making decision (see Algorithm 2).

Compared to the algorithm presented in [DBC15], the method described above has several advantages:

1. Aside the choice of the class of prediction functions g (and hence of decision functions h) that must be made whatever the approach, there are two parameters to set. The first is N, the number of confidence intervals one is willing to consider. Higher values of N may seem preferable because they would yield higher precision. But this is illusory since what matters is the difference in the confusion matrices. In addition, one obtains a better precision on the estimation of these matrices if the number of training sequences used to compute them is large. As will be seen below, a good choice seems to be $N = 5$. The second parameter

Algorithm 1 Learning algorithm for early classification of time series

```
Input:
    • A training set $\mathcal{S}$ of $m$ labeled time series $(\mathbf{x}_T^i, y^i) \in \mathbb{R}^T \times \mathcal{Y}$ $(1 \leq i \leq m)$ ;
    • A validation set $\mathcal{S}'$ of $m'$ labeled time series $(\mathbf{x}_T^j, y^j) \in \mathbb{R}^T \times \mathcal{Y}$ $(1 \leq j \leq m')$ ;
    • a set $\mathcal{G} = \{\mathcal{G}_{\min} \cup \ldots \cup \mathcal{G}_T\}$ where each $\mathcal{G}_t$ is itself a set of scoring functions: $g_t : \mathbb{R}^t \rightarrow [0,1]$;
1: for $t \in \{\min, \ldots, T\}$ do
2:     Use a learning algorithm that takes as input $\mathcal{S}$ and $\mathcal{G}_t$ and returns a function $g_t$
3:     Using $g_t$ and $\mathcal{S}$: compute $N$ confidence intervals on $[0,1]$ as explained in Section 2
4: end for
5: for $t \in \{1, \ldots, T-1\}$ do
6:     Compute the transition matrices $\mathsf{M}_t^{t+1}$
7: end for
```

Algorithm 2 Prediction algorithm for early classification of time series

```
Input:
    • An incomplete sequence $\mathbf{x}_t$ with $1 \leq t \leq T$
1: for $t \in \{1, \ldots, T\}$ do
2:     Compute the sequence $\langle \gamma_1, \ldots, \gamma_t \rangle$ coding for $\mathbf{x}_t$
3: end for
4: for $\tau \in \{0, \ldots, T-t\}$ do
5:     Compute the expected cost $f_\tau(\mathbf{x}_t)$ using equation (9)
6: end for
7: return $t^\star = t + \mathrm{ArgMin}_{\tau \in \{0,\ldots,T-t\}} f_\tau(\mathbf{x}_t)$
```

is the order of the dependency taken into account, similarly to Markov chain models that can depend on the past to various degrees.

2. The confusion matrices that appear in equation 9 tend naturally to differ, leading to better estimates of the future decision costs.

3. The conditional probabilities $P_{t+\tau}(\hat{y}|y, \gamma_{t+\tau} = \ell)$ tend also to differ for different values of the confidence interval ℓ, which favors better predictions.

In the following, we compare the two methods: one we call "clustering-based" and the other called "confidence-based" which is reduced here to its baseline version with a first order time dependency. For this, we use both synthetic data that allow a fine control of the parameters influencing the difficulty of the task: possible gain of information with time, noise level of the measurements and cost of delaying the decision (see Section 3), and real-like data from the UCR repository (see Section 4).

In addition, when possible, that is on synthetic data, we look at how close the methods come to the optimal omniscient algorithm.

3 Experimental Evaluation on Synthetic Data Sets

The goal of the experimental evaluation is to measure how the methods behave when the (i) difficulty of the decision task varies, and (ii) faced with varying levels of increasing costs when delaying decision.

3.1 The generation of the synthetic data sets

The difficulty of the decision task can naturally be determined using two types of controlling parameters. One that controls the information that can be gained about the class of the incoming sequence with each new measurement. And one that controls the noise level of the measurements. The two parameters are not independent as more noise decreases the information that can be gained, but they still are complementary as the noise level is supposed to be constant over time when the gain of information can vary.

These types of control parameters were used in the experimental setting described in [DBC15], and since we want to compare the new method to the one proposed in [DBC15], we generated data sets along the same protocol.

The overall idea is to generate sets of time series according to two class models, one for the +1 class and one for the −1 class. In addition, within each class, there are sub-classes, some of them that can share a strong similarity with sub-classes of the other class.

Specifically, the time series have been generated according to the following equation:

$$\mathbf{x}_t = \underbrace{t \times \text{slope} \times \text{class}}_{\text{information gain}} + \underbrace{\mathrm{x}_{max} \sin(\omega_i \times t + \varphi_j)}_{\text{sub shape within class}} + \underbrace{\eta(t)}_{\text{noise factor}} \quad (10)$$

The higher the value of the *slope* factor (noted m below), the higher the gain at each time step. At the same time, x_{max} controls the importance of the sub-classes within each class. If $\mathrm{x}_{max} = 0$ there are no sub-classes, and little possible confusion between the classes, except for the noise factor $\eta(t)$. If x_{max} has a large value, the sub-shape tend to dominate the information gain factor, at least for not large enough time step t. Figure 2 illustrates what can be obtained for three classes of time series, one with slope $m = 0.01$ (class $y = +1$), one with $\mathrm{m} = -0.01$ (class $y = -1$) and one with $\mathrm{m} = 0$ (a confusing class). Here $\mathrm{x}_{max} = 5$ and the subclasses are determined by the period ω_i and phase φ_j. The noise factor $\eta(t)$ is randomly chosen from a Gaussian distribution with a mean μ and a standard deviation σ.

We conducted experiments using two disjoint training sets $\mathcal{S}_1$ and $\mathcal{S}_2$ each containing $2,500$ time series and a testing set $\mathcal{T}$ containing $5,000$ time series. Each set is equally divided between the two classes $y = -1$ and $y = +1$.

The set $\mathcal{S}_1$ is used for learning the functions g_t, while the set $\mathcal{S}_2$ is used to compute the confidence intervals for each time step t and the confusion matrices $P_{t+\tau}(\hat{y}|y, \gamma_{t+\tau} = \ell)$.

Table 1 displays the range of parameters used for generating the synthetic data set.

In addition to the various types of data sets generated, we varied also the cost function $\mathrm{C}(t)$ that expresses how costly it is to delay making a decision. The cost function is a non decreasing function of time. In our experiments, we used linear cost functions: $\mathrm{C}(t) = \text{constant} \times t$.

3.2 Experimental results

For each method, and for each experimental condition determined by the three controlling factors: information gain m, noise level $\eta(t)$ and cost of delaying decision $\mathrm{C}(t)$, we measured the following quantities:

The real costs are obtained for each test time series $\mathbf{x}_t$ by computing the predicted class $\hat{y} = h_t(\mathbf{x}_t)$ and comparing it with the real label y and evaluating $\mathrm{C}(\hat{y}|y) + \mathrm{C}(t)$.

The $\overline{\mathrm{C}}_{\mathrm{PRCM}}$ value is an optimistic optimal value. It is the cost (or gain if this is a negative value) that the system would endure if it made a decision as soon as the prediction is correct, $h_t(\mathbf{x}_t) = y$, which can happen accidentally even though the decision function h_t is bad. We still report this value since it gives an idea of how far is the method to this (unrealistic) optimal early decision method.

Table 8 reports the results for a slope $\mathrm{m} = 0.07$. Other results for the values $\mathrm{m} = 0.005$, $\mathrm{m} = 0.01$, $\mathrm{m} = 0.05$ and $\mathrm{m} = 0.1$ are available at: `https://github.com/asma-dachraoui/ECTS-Costly-delaying-decision`.

3.3 Comparison of the methods and interpretation

A first look at the table of results shows that:

- For both methods, when the cost of relying decision increases (from $0.001 \times t$ to $0.1 \times t$), the algorithms decrease the waiting time (if one looks for the same noise level $\eta(t)$).
- As the difficulty of the task increases, with mounting noise level (from 0.1 to 20) the algorithms tend to first increase the time of decision, because it is more difficult to make a good prediction early on, before deciding that it is not worth waiting, and making a prediction after 4 time steps, which is the minimum amount of time set in our experiments.
- However, the confidence-based method tends to delay the "discouragement" phase more than the clustering-based method. This is advantageous for small and medium values of $\eta(t)$ and tend to be slightly disadvantageous when the noise level is high.

In order to compare the two algorithms, we performed paired statistical t-tests, computed using $\frac{\overline{d}}{s_d/\sqrt{N}}$, where $\overline{d}$ is the difference between the two observations on each pair, s_d is the standard deviation of the differences and $N = 225$, the number of examples. We compared the cost incurred by the two systems when following their decision policies: $\overline{\mathrm{C}}_{\mathrm{RCM}}$. The question was: is one algorithm significantly superior to the other in the experimental setting? Table 3 gives the results for the paired t-tests when we consider the difference $\overline{\mathrm{C}}_{\mathrm{RCM}}(\text{clustering-based}) - \overline{\mathrm{C}}_{\mathrm{RCM}}(\text{confidence-based})$. For small and medium cost of delaying decision, the confidence-based method is significantly better than the clustering-based method (largely above the significance level for $\alpha = 0.05\%$). Nothing can be said one way or the other for $\mathrm{C}(t) = 0.07$, while the clustering-based method is better for $\mathrm{C}(t) = 0.1$, because it does not wait to make a decision.

These results show that the confidence-based method, even in the baseline implementation, is clearly superior to the clustering-based method.

We also compared the proximity of the real cost $\overline{\mathrm{C}}_{\mathrm{RCM}}$ incurred by each algorithm with the optimal cost $\overline{\mathrm{C}}_{\mathrm{ICM}}$ given by the omniscient algorithm. In Table 4, we report the results for the paired t-test when

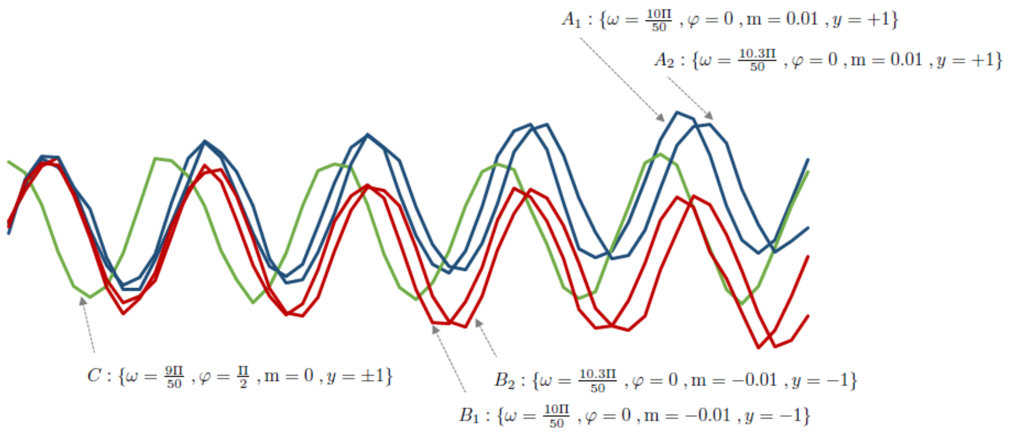

Figure 2: An example from synthetic data set $\mathcal{S}$ where $\eta : (\mu = 0, \sigma = 0.2)$.

Parameter	Description	Value(s)
T	number of data points	50
x_{max}	sine amplitude	5
$\{\omega_1, \omega_2, \omega_3\}$	sine period	$\{\frac{9\Pi}{T}, \frac{10\Pi}{T}, \frac{10.3\Pi}{T}\}$
$\{\varphi_1, \varphi_2\}$	sine phase	$\{0, \frac{\Pi}{2}\}$
m	slope	$\mathrm{m} \in \{0.005, 0.01, 0.05, 0.07, 0.1\}$
η	noise as Gaussian distribution	$\mu = 0$, $\sigma \in \{1 : 200\}$
K	Number of sub-classes in each class	$K = 3$ (see sine period values)

Table 1: The set of parameters used for the generation of the data sets.

we consider the differences $\overline{\mathrm{C}}_{\mathrm{RCM}}$(clustering-based)-$\overline{\mathrm{C}}_{\mathrm{ICM}}$(perfect algorithm) and $\overline{\mathrm{C}}_{\mathrm{RCM}}$(confidence-based)-$\overline{\mathrm{C}}_{\mathrm{ICM}}$(perfect algorithm). Again, the confidence-based method significantly better approximates the optimal decision time .

Another mean to compare the methods is to look at the highest level of noise $\eta(t)$ for which a method yields a cost that is better than the cost incurred when deciding at the first possible moment (4 in these experiments) with a margin of at least 0.1. For instance, (see Table 8), for $\mathrm{m} = 0.07$ and $\mathrm{C}(t) = 0.001$, $\overline{\mathrm{C}}_{\mathrm{ECM}} \leq 0.004 - 0.1$ up to noise level $\eta(t) = 5$ (for which $\overline{\mathrm{C}}_{\mathrm{ECM}} = -0.1$ for the clustering-based method), while this is true up to $\eta(t) = 20$ for the confidence-based method. Thus, for $\mathrm{m} = 0.07$ and $\mathrm{C}(t) = 0.001$, the clustering-based method is significantly winning, according to our rule, for 5 values of noise levels, while the confidence-based methods is winning for all the 7 noise levels reported in the experiments.

This comparison, for all values of m and values of $\mathrm{C}(t)$ can be expressed as the histogram of Figure 3. It is apparent that the Confidence-based method is winning in all the situations in which the Clustering-based method wins, plus others. It thus brings significant gains in a wider spectrum of situations than the Clustering-based method.

4 Experimental Evaluation on Real-like Data Sets

The goal here was to test the methods with real like data. For this, we chose data sets from the UCR Time-Series Classification/Clustering archive [CKH+15].

The data sets correspond to real binary classification problems. For example, the data set *FordA* includes time series collected from an automotive subsystem. Each example in the data set consists of a time series composed of 500 data points recording the engine noise and a label describing the diagnostic result according to a certain symptom. Class +1 is associated with the diagnosis that the symptom exists and −1 with the diagnosis that the symptom does not exist.

Out of the 31 data sets involving two classes in the UCR archive, we selected 11 data sets: DistalPhalanxOC, ECGFiveDays, ItalyPowerDemand, Mid-

Quantity	Description
$\overline{\tau}^{\star}_{\text{ETM}}$	mean of the decision time computed by the method $\pm$ standard deviation
$\overline{\text{C}}_{\text{RCM}}$	mean real cost using decision time $\overline{\tau}^{\star}_{\text{ETM}}$
$\overline{\tau}^{\star}_{\text{PETM}}$	decision time $= t^* = \text{ArgMin}_{t\in\{1,\ldots,T\}} f(\mathbf{x}_t)$ (using the knowledge of the complete series).
$\overline{\text{C}}_{\text{PRCM}}$	mean real cost using decision time $\overline{\tau}^{\star}_{PETM}$
$\overline{\tau}^{\star}_{\text{ITM}}$	mean time before $h_t(\mathbf{x}_t) = y$ (perfect algorithm)
$\overline{\text{C}}_{\text{ICM}}$	mean real cost when deciding the first time that $h_t(\mathbf{x}_t) = y$ (perfect algorithm)

Table 2: Quantities measured in the experiments.

Paired t-test statistic	C(t)				
	0.001	0.01	0.05	0.07	0.1
Clustering-based approach vs. Confidence-based approach	6.6795	5.0266	2.7667	1.3091	-2.4248

Table 3: Results of the paired t-test over the real cost $\overline{\text{C}}_{\text{RCM}}$ incurred by each algorithm using synthetic data sets for the 5 different cost functions $\text{C}(t) = \{0.001, 0.01, 0.05, 0.07, 0.1\}$.

Paired t-test statistics	C(t)				
	0.001	0.01	0.05	0.07	0.1
Clustering-based approach vs. perfect algorithm	-12.4027	-16.7838	-16.6993	-14.3468	-11.6398
Confidence-based approach vs. perfect algorithm	-9.5060	-11.9130	-16.5332	-14.8554	-12.0682

Table 4: Results of the paired t-test over the real cost $\overline{\text{C}}_{\text{RCM}}$ incurred by the Clustering-based approach (respectively, the Confidence-based approach) and the optimal cost $\overline{\text{C}}_{\text{ICM}}$ for the 5 different cost functions $\text{C}(t) = \{0.001, 0.01, 0.05, 0.07, 0.1\}$.

dlePhalanxOC, MoteStrain, PhalangesOC, ProximalPhalanxOC, SonyAIBORobotS, SonyAIBORobotSII, Strawberry, TwoLeadECG. The others were excluded because of their small sizes (less than 500 time series), or their large lengths.

Originally, the data sets were provided with two separate train and test data sets. As we need three sets in our algorithm ($\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{T}$), we combined the train and test data and then randomly divided the total into three sets of the same size. The misclassification costs are set to $\text{C}(\hat{y}|y) = +1$ if $\hat{y} = y$ and -1 if $\hat{y} \neq y$

To make the early-decision tasks, we set cost functions for delaying decisions as $\text{C}(t) = d \times t$, where $d \in \{0.001, 0.01, 0.05, 0.07, 0.1\}$ ranging the delay cost from low to high values.

The complete tables of results are available on: `https://github.com/asma-dachraoui/ECTS-Costly-delaying-decision`.

An abbreviated table is given here for the two data sets: DistalPhalanxOC, ECGFiveDays in Table **??**.

Table 6 shows the impact of varying the number of clusters K (Clustering-based approach) and the number of intervals N (confidence-based approach) over the ItalyPowerDemand real data set. Results of the optimal time decision and the cost incurred by the two methods are given when varying $k \in \{\in 5, 8, 12, 15, 16\}$ and $N \in \{5, 10\}$.

One important observation is that, for the clustering-based method, the number of clusters k chosen heavily influences the results. By contrast, the number N of confidence intervals (here $N = 5$ and $N = 10$) has no noticeable effect on the results.

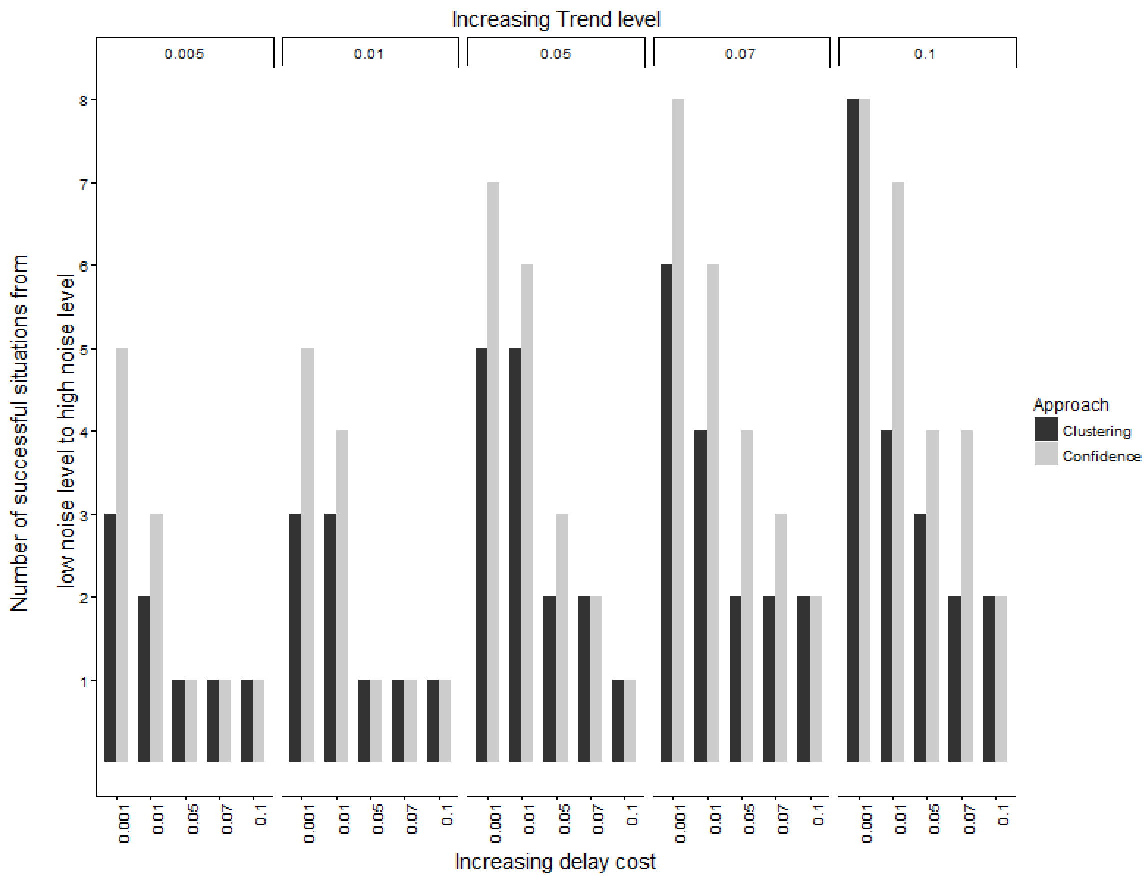

Figure 3: Histogram showing the number of noise levels for which each method brings a significant gain as compared to the earliest possible decision.

In order to compare the two methods, we used the Wilcoxon Signed-Rank test since we had only 11 data sets over which the performances of the methods were measured.

C(t)	0.001	0.01	0.05	0.07	0.1
z	63	51	23	10	10

Table 7: Wilcoxon Signed-Rank Test over the real data sets with $\alpha = 0.05$ and $n - 1 = 10$ degrees of freedom.

Table 7 provides the results for the Wilcoxon Signed-Rank Test over the real-like 11 data sets with $\alpha = 0.05$ and $n - 1 = 10$ degrees of freedom. Similarly to the results obtained using the synthetic data set, the Confidence-based method is remarkably better than the Clustering-based method (largely above the significance level for $\alpha = 0.05$).

5 Conclusion and future works

In this paper, we revisited the problem of early classification of time series when delaying decision incurs a rising cost. Analyzing the work reported in [DBC15], we have presented a new method that diminishes the number of parameters to set, while providing a general scheme to capture generalized patterns in the training sequences.

We have conducted extensive experiments in controlled situations with synthetic data sets under a wide variety of parameter values. The method we propose, even in its baseline implementation, significantly outperforms the clustering-based approach described in [DBC15]. This means that the new algorithm is able to exploit the information in the time series more quickly and with more precision. This has been confirmed with experiments using real-like data sets.

It is expected that using the same algorithm with higher order of time dependencies taken into account

Quantity	Clustering-based approach					Confidence-based approach	
	5	8	12	15	16	5	10
$\overline{\tau}^{\star}_{\text{ETM}}$	5.7±1.0	13.0±0.0	14.3±2.5	19.1±2.0	17.8±2.9	16.6±4,3	17.1±3,8
$\overline{\text{C}}_{\text{RCM}}$	-0.13	-0.73	-0.76	-0.91	-0.90	-0.907	-0.901
$\overline{\tau}^{\star}_{\text{PETM}}$	14.5±7.1	12.2±2.3	10.9±5.7	10.0±7.7	11.8±6.1	17.3±3.48	18.3±3.9
$\overline{\text{C}}_{\text{PRCM}}$	-0.64	-0.65	-0.58	-0.48	-0.58	-0.82	-0.85

Table 6: Impact of varying the number of clusters (Clustering-based approach) and the number of intervals (Confidence-based approach) over the ItalyPowerDemand real data set.

would further improve the performances. These richer models should indeed be able to extract the useful information in the training set and new incoming time series, and come near the optimal decision time and optimal cost. However, only very large training sets can allow a learning algorithm to reach this type of performance, by enabling the learning of the large number of conditional dependencies involved in these higher order models.

References

[APTG12] Hyrum S Anderson, Nathan Parrish, K Tsukida, and MR Gupta. Early time-series classification with reliability guarantee. *Sandria Report*, 2012.

[Ber85] James O Berger. *Statistical decision theory and Bayesian analysis.* Springer Science & Business Media, 1985.

[CKH+15] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[DBC15] 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*, pages 433–447. Springer, 2015.

[DeG05] Morris H DeGroot. *Optimal statistical decisions*, volume 82. John Wiley & Sons, 2005.

[HC13] Nima Hatami and Camelia Chira. Classifiers with a reject option for early time-series classification. In *Computational Intelligence and Ensemble Learning (CIEL), 2013 IEEE Symposium on*, pages 9–16. IEEE, 2013.

[ISS00] Katsuhiko Ishiguro, Hiroshi Sawada, and Hitoshi Sakano. Multi-class boosting for early classification of sequences. *Statistics*, 28(2):337–407, 2000.

[PAGH13] Nathan Parrish, Hyrum S Anderson, Maya R Gupta, and Dun Yu Hsiao. Classifying with confidence from incomplete information. *J. of Mach. Learning Research*, 14(1):3561–3589, 2013.

[Pla99] John C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, volume 10, pages 61–74, 1999.

[WW48] Abraham Wald and Jacob Wolfowitz. Optimum character of the sequential probability ratio test. *The Annals of Mathematical Statistics*, pages 326–339, 1948.

[XPP09] Zhengzheng Xing, Jian Pei, and S Yu Philip. Early prediction on time series: A nearest neighbor approach. In *IJCAI*, pages 1297–1302. Citeseer, 2009.

[XPPW11] Zhengzheng Xing, Jian Pei, S Yu Philip, and Ke Wang. Extracting interpretable features for early classification on time series. In *SDM*, volume 11, pages 247–258. SIAM, 2011.

C(t)	$\eta(t)$	Clustering-based approach				Confidence-based approach					
		$\overline{\tau}^{\star}_{\text{ETM}}$	$\overline{\text{C}}_{\text{RCM}}$	$\overline{\tau}^{\star}_{\text{PETM}}$	$\overline{\text{C}}_{\text{PRCM}}$	$\overline{\tau}^{\star}_{\text{ETM}}$	$\overline{\text{C}}_{\text{RCM}}$	$\overline{\tau}^{\star}_{\text{PETM}}$	$\overline{\text{C}}_{\text{PRCM}}$	$\overline{\tau}^{\star}_{\text{ITM}}$	$\overline{\text{C}}_{\text{ICM}}$
	0.1	5.9±0.4	-0.76	7.1±2.8	-0.76	8.4±8.6	-0.76	8.6±4.8	-0.76	4.0±0.2	-0.77
	0.2	5.1±0.4	-0.64	6.8±3.1	-0.65	17.4±14.6	-0.74	10.0±4.5	-0.74	4.2±0.9	-0.8
	0.5	9.0±4.2	-0.45	12.2±4.0	-0.66	16.3±10.5	-0.74	18.8±10.0	-0.74	5.1±2.4	-0.87
	1.0	14.4±2.3	-0.72	12.3±4.4	-0.6	20.8±10.0	-0.74	23.2±8.3	-0.74	5.3±3.0	-0.92
0.001	1.5	16.1±1.5	-0.65	9.6±4.8	-0.27	23.6±13.2	-0.68	30.3±10.4	-0.7	6.5±4.5	-0.96
	5.0	10.9±7.1	-0.1	9.3±5.4	-0.01	32.8±5.2	-0.65	42.3±6.1	-0.67	9.7±7.8	-0.96
	10.0	13.1±11.5	-0.08	11.1±7.5	-0.02	41.4±5.7	-0.52	45.2±4.4	-0.5	11.4±10.0	-0.95
	15.0	10.9±6.7	-0.03	11.4±6.4	-0.02	34.5±16.6	-0.25	39.2±14.9	-0.27	12.2±10.7	-0.95
	20.0	12.4±10.6	-0.01	12.6±9.2	0.0	44.5±3.9	-0.24	46.3±3.2	-0.24	11.3±9.9	-0.96
	0.1	4.0±0.0	-0.71	5.5±1.3	-0.71	5.3±1.4	-0.71	6.4±2.2	-0.71	4.0±0.2	-0.74
	0.2	5.1±0.4	-0.6	5.9±2.1	-0.59	6.7±1.9	-0.67	7.8±2.7	-0.67	4.2±0.9	-0.77
	0.5	5.1±0.6	-0.24	7.0±3.4	-0.27	11.1±4.5	-0.64	12.0±2.8	-0.64	5.1±2.4	-0.83
	1.0	6.7±2.5	-0.26	8.1±3.8	-0.33	15.0±4.0	-0.62	16.3±4.6	-0.59	5.3±3.0	-0.88
0.01	1.5	7.2±3.8	-0.09	8.8±4.3	-0.13	17.0±9.2	-0.51	19.1±5.5	-0.53	6.5±4.5	-0.9
	5.0	6.2±3.7	0.03	8.3±4.2	0.08	26.9±6.6	-0.34	33.3±5.5	-0.33	9.7±7.8	-0.87
	10.0	4.2±1.0	0.04	8.5±4.3	0.09	18.1±8.4	0.01	38.1±10.3	-0.08	11.4±10.0	-0.85
	15.0	4.9±1.8	0.04	6.4±2.4	0.05	5.4±0.5	0.05	8.3±9.7	0.06	12.2±10.7	-0.84
	20.0	4.1±1.1	0.04	6.9±3.9	0.07	9.7±1.6	0.08	10.2±2.0	0.08	11.3±9.9	-0.86
	0.1	4.0±0.0	-0.55	4.5±0.9	-0.53	4.6±0.8	-0.54	5.4±1.4	-0.5	4.0±0.2	-0.58
	0.2	5.0±0.2	-0.4	5.3±0.4	-0.38	4.8±0.7	-0.42	5.8±1.8	-0.44	4.2±0.9	-0.6
	0.5	5.0±0.2	-0.04	5.5±1.2	0.01	8.4±1.5	-0.3	9.3±1.8	-0.27	5.0±2.0	-0.62
	1.0	5.0±0.2	0.04	5.6±1.2	0.02	8.8±2.8	-0.16	10.2±3.3	-0.13	5.2±2.7	-0.66
0.05	1.5	5.3±1.0	0.23	5.7±1.3	0.26	7.2±4.1	0.08	11.0±3.3	0.0	6.3±3.6	-0.64
	5.0	4.0±0.0	0.2	5.8±2.1	0.29	4.0±0.0	0.2	4.0±0.0	0.2	9.6±7.5	-0.48
	10.0	4.0±0.0	0.2	5.2±1.6	0.26	4.0±0.0	0.2	4.0±0.0	0.2	11.0±9.4	-0.39
	15.0	4.0±0.0	0.2	4.5±0.8	0.22	5.1±0.7	0.25	5.2±0.6	0.26	11.6±9.9	-0.36
	20.0	4.0±0.0	0.2	4.3±0.7	0.22	4.0±0.0	0.2	4.0±0.0	0.2	10.8±9.2	-0.41
	0.1	4.0±0.0	-0.47	4.5±0.9	-0.44	4.6±0.8	-0.45	4.7±1.1	-0.44	4.0±0.2	-0.5
	0.2	5.0±0.2	-0.3	5.3±0.4	-0.28	4.6±0.8	-0.33	5.2±1.5	-0.35	4.2±0.9	-0.51
	0.5	5.0±0.2	0.06	5.5±1.2	0.12	7.5±1.3	-0.11	8.2±2.1	-0.11	5.0±2.0	-0.52
	1.0	4.0±0.0	0.13	5.4±0.8	0.13	7.1±2.1	0.03	8.3±2.6	0.07	5.2±2.2	-0.56
0.07	1.5	4.1±0.6	0.28	5.7±1.2	0.38	4.0±0.0	0.27	7.7±3.4	0.23	6.2±3.2	-0.52
	5.0	4.0±0.0	0.28	5.4±1.4	0.38	4.0±0.0	0.28	4.0±0.0	0.28	9.2±7.0	-0.29
	10.0	4.0±0.0	0.28	4.8±1.0	0.34	4.0±0.0	0.28	4.0±0.0	0.28	9.8±8.0	-0.18
	15.0	4.0±0.0	0.28	4.3±0.6	0.3	5.1±0.7	0.36	4.9±0.7	0.34	10.1±8.4	-0.14
	20.0	4.0±0.0	0.28	4.2±0.5	0.29	4.0±0.0	0.28	4.0±0.0	0.28	9.7±7.8	-0.2
	0.1	4.0±0.0	-0.35	4.5±0.9	-0.31	4.0±0.0	-0.35	4.1±0.6	-0.34	4.0±0.2	-0.37
	0.2	4.0±0.0	-0.18	4.5±0.9	-0.14	4.6±0.8	-0.2	4.6±1.1	-0.2	4.2±0.9	-0.38
	0.5	5.0±0.2	0.21	5.3±0.5	0.25	4.7±1.0	0.26	5.4±1.7	0.27	5.0±1.7	-0.37
	1.0	4.0±0.0	0.25	5.4±0.5	0.28	5.0±0.0	0.29	6.2±2.0	0.26	5.1±2.0	-0.41
0.1	1.5	4.0±0.1	0.39	5.5±0.8	0.53	4.0±0.0	0.39	4.3±1.1	0.39	6.1±3.0	-0.33
	5.0	4.0±0.0	0.4	5.1±0.8	0.51	4.0±0.0	0.4	4.0±0.0	0.4	8.1±5.7	-0.03
	10.0	4.0±0.0	0.4	4.6±0.8	0.46	4.0±0.0	0.4	4.0±0.0	0.4	7.6±5.6	0.07
	15.0	4.0±0.0	0.4	4.1±0.4	0.41	5.0±0.6	0.5	4.6±0.5	0.46	8.0±6.4	0.14
	20.0	4.0±0.0	0.4	4.1±0.4	0.41	4.0±0.0	0.4	4.0±0.0	0.4	7.6±5.6	0.05

Table 8: Comparison table: results of clustering approach vs. confidence approach 2 (with trend = 0.07, over simulated sine data)