Early classification of time series by simultaneously optimizing the accuracy and earliness

Usue Mori [*], Alexander Mendiburu [†],
Sanjoy Dasgupta [‡], and Jose A. Lozano [§]

October 18, 2017

Abstract

The problem of early classification of time series appears naturally in contexts where the data, of temporal nature, is collected over time, and early class predictions are interesting or even required. The objective is to classify the incoming sequence as soon as possible, while maintaining suitable levels of accuracy in the predictions. Thus, we can say that the problem of early classification consists in optimizing two objectives simultaneously: accuracy and earliness. In this context, we present a method for early classification of time series based on combining a set of probabilistic classifiers together with a stopping rule. This stopping rule will act as a trigger and will tell us when to output a prediction or when to wait for more data, and it's main novelty lies in the fact that it is built by explicitly optimizing a cost function based on accuracy and earliness. We have selected a large set of benchmark datasets and 4 other state-of-the-art early classification methods and we have evaluated and compared our framework obtaining superior results in terms of both earliness and accuracy.

1 Introduction

Time series classification [1, 2] is a supervised learning task which deals with predicting the class labels of time series as accurately as possible by using a

[*]U. Mori is with the Department of Applied Mathematics, Statistics and Operations Research, University of the Basque Country UPV/EHU, e-mail: (usue.mori@ehu.eus).

[†]A. Mendiburu is with the Department of Computer Arquitecture and Technology, University of the Basque Country UPV/EHU.

[‡]S. Dasgupta is with the Department of Computer Science and Engineering, University of California, San Diego.

[§]J.A. Lozano is with the Departament of Computer Science and Artifical Inteligence, University of the Basque Country UPV/EHU, and the Basque Center for Applied Mathematics BCAM.

training set of completely labeled full length time series. An example application of this task is trying to identify which household device is working at a given time by using the electricity usage profiles [3]. Also, Kadous and Sammut [4] use electrocardiography (ECG) data to predict whether a patient has heart disease or not and also to recognize sign language signs. Pei et al. [2] use image sequences for facial expression detection, Ye and Keogh [5] use time series classification methods to identify coffee and wheat varieties by using spectrography series, and Pei et al. [2] and Li et al. [6] perform motion identification by using multivariate sensor readings, which can also be interpreted as time series.

Due to its applicability, and because time series databases are increasingly common in many real world domains, this problem has been largely studied in the past decade. Consequently, some extensions of the classical problem have also been proposed and worked on. Among them, semi-supervised classification [7], positive unlabeled classification [8] and early classification [9] are the most notable, and in this work we will focus on the latter one.

The problem of early classification of time series appears when the unlabeled time series are collected over time, and it is desirable to obtain the class label predictions as early as possible. For example, Evans et al. [10] show that the monitoring of patients and early identification of physiologic deterioration can be used to raise alerts and prevent crises in hospitalized patients. Also, Ghalwash et al. [11] mention early stock crisis identification; Bregón et al. [12] apply early classification to classify different types of faults in a simulated industrial plant; Hatami and Chira [13] attempt to classify a set of different odors as early as possible by using odor signals obtained from a set of sensors with the aim of identifying chemical leaks. Finally, in Mori et al. [14], an early classification approach is applied to detect and identify bird songs as early as possible, with the objective of saving memory and battery life of the recording devices.

It is obvious that the earliness of the predictions has an influence on their accuracy but, can this influence be quantified or modeled? Can we build models that are able to classify time series as early as possible while maintaining a suitable level of accuracy? This is exactly the aim of early classification, finding a trade-off between two conflicting objectives: the accuracy of the predictions and their earliness [9]. However, the degree of conflict between these two objectives changes drastically from database to database and, as more data becomes available, the accuracy evolves differently in each database [14]. So, the problem of early classification can be described as an optimization problem in which two conflicting objectives must be optimized at the same time.

In this paper, we propose an early classification framework, departing from our preliminary work in [15], which is based on combining a set of probabilistic classifiers and an optimized stopping rule. In this previous paper, we presented an initial version of the method, considering only a basic stopping rule and only one specific cost function. In this paper, we propose several different stopping rules and cost functions, we improve on the definition and design of the method, we extend our experimentation to analyze the characteristics of our methodology, including the influence of the parameters more in detail, and we include a new state-of-the-art method in the comparative analysis.

The rest of the paper is organized as follows. In Section 2, we present the problem of early classification and summarize the previous work on the topic. Section 3 presents the main contribution of this paper. In Section 4, the experimentation and validation of this method is carried out and in Section 5 we analyze the results in detail. Finally, in Section 6, we summarize the main conclusions and propose some future research directions.

2 Early classification of time series

A *time series* is an ordered sequence of pairs of finite length L [9]:

$$TS = \{(t_1, x_1), (t_2, x_2), \ldots, (t_L, x_L)\}, \tag{1}$$

Note that timestamps $\{t_i\}_{i=1}^{L}$ take positive and ascending real values, so we are dealing with an ordered sequence. Indeed, although in the most common examples the $\{t_i\}_{i=1}^{L}$ values refer to temporal references (timestamps), other types of orderings can also be defined [1]. The values of the time series ($\{x_i\}_{i=1}^{L}$) are usually real numbers [1]. Finally, a *database of time series* is a set of time series with no order, which can all be of the same length, or can have different lengths.[2]

Considering these definitions, *time series classification* is formally defined as a supervised data mining task in which, given a training set of complete time series and their respective class labels $X = \{(TS_1, CL_1), (TS_2, CL_2), \ldots, (TS_n, CL_n)\}$, the aim is to build a model that is able to predict the class label of new unlabeled time series as accurately as possible [1, 2, 9, 14].

As an extension of time series classification, **early classification of time series** also departs from a completely labeled training set of time series. However, this problem appears when the new unlabeled time series are collected over time. In these contexts, early class predictions are sometimes important, for example, when collecting the data is expensive or due to the consequences associated with making late decisions [16]. However, maintaining a suitable level of accuracy is also usually an important requirement. To sum up, the key to early classification of time series is not just to maximize accuracy, but rather to find a trade-off between two conflicting objectives: accuracy and earliness.

This problem has been compared to other classical problems in machine learning, such as optimal stopping, feature selection, learning with incomplete data, etc. [16]. However, early classification has its own peculiarities, such as the temporal correlation in the data, so, its appearance is fairly recent in the literature. Xing et al. [9] formally defined the problem of early classification for

[1]The values can be univariate or multivariate, but, in this work, we will only deal with univariate time series

[2]In the following sections, we will focus on datasets conformed with time series of the same length L. However, with a few changes, our proposed method could also be applied to sequences that take values from a finite set, databases with series of different lengths, and even to classify series of finite but unknown lengths. The necessary modifications will be commented on as we introduce the method.

the first time and proposed a method called Early Classification on Time Series (ECTS). This work analyzes the stability of the nearest neighbor relationships in the training set over time. Based on this information, the method selects the training time series that can reliably be used at each timestamp within a 1NN classifier.

Another intuitive approach is to simply learn a model for each early timestamp and design different mechanisms to decide which predictions are reliable and which are not. Different types of models and reliability conditions result in a wide variety of early classification methods such as those proposed in [13, 16, 17].

A new method, denominated Early Classification of Time Series based on Discriminating Classes Over Time (ECDIRE)[14], lies in between these two methods. The main objective of this method is to analyze the evolution of the accuracy of a set of probabilistic classifiers over time, with the aim of identifying the timestamps from whence it is safe to make predictions. Predictions will only be made in these timestamps or later and, so, a large number of predictions are avoided. As with the second type of method introduced in this section, ECDIRE also incorporates a reliability condition which must also be met, and is useful to discard outlier series that do not belong to any class.

Finally, a completely different strategy can be found in Xing et al. [18], Ghalwash et al. [11] and He et al. [19], where the authors propose methods based on shapelets, which refer to subsequences of time series that can be used to discriminate a given class from the others. In the context of early classification, the trick relies on finding a library of shapelets that is useful to discriminate the classes as early as possible.

Note that, even if the problem of early classification is clearly an optimization problem, neither of the approaches introduced above deal with the problem of early classification from a cost minimization point of view. Only one recent work [20] deals with the problem from this perspective. These authors propose a complex meta-algorithm based on calculating the expectation of the cost of misclassification at time t, together with the cost of making the decision at time t. Unfortunately, this method has only been validated on one benchmark time series, it has not been compared with other state-of-the-art methods and its code is not available for comparison.

In this paper, we propose an early classification method based on a very intuitive idea: we will combine a set of probabilistic classifiers and a stopping rule which, when optimized by a suitable cost function, will act as a reliability test. The novelty of the approach lies in the construction of the stopping rule, which is based on an optimization process that will aim to find a trade-off between earliness and accuracy.

As it will be seen, this approach is easy to understand for non-expert users because it only implies using a basic rule. However, it shows improved performance over other much more complex approaches in the literature and it is capable of adapting its behavior to the different shapes that the evolution of the accuracy over time can take, and to the different degrees of conflict between the objectives.

3 Early classification of time series by minimizing a cost function

In this section we present our early classification method. The proposed framework will consist of a learning phase and a prediction phase, which will be explained in detail in the following sections.

3.1 Learning phase

The goal of this phase is to train a model which provides early and accurate class predictions for new unlabeled time series. For this purpose, we will use a training set $X = \{(TS_1, CL_1), (TS_2, CL_2), ..., (TS_n, CL_n)\}$ of labeled full length time series of finite length, and we will proceed as follows:

Step 1: Learn probabilistic classifiers

In this first step, we will train two sets of probabilistic classifiers.

- On the one hand, we will train a set of classifiers $\{h_t\}_{t=1}^{L}$ for all timestamps $t \in \{1, 2, ..., L\}$, or for a user-defined subset of timestamps. In our experimentation, we use many different databases, of which we have no specific domain knowledge, a priori. As such, based on a percentage of the series lengths, we have selected a sequence of equidistant time points, in which we carry out the classification. Nevertheless, the user could choose any other subset of timestamps, based on domain knowledge or other information of the shape of the series or even by using specific time series sampling methods. This is especially important if the series are of different or unknown lengths or if they are unevenly sampled.

  To train these classifiers, we will use the whole training set X. Each classifier h_t will receive the first t points of a series, and will output the posterior probabilities for each class at that time. These classifiers will be used to obtain the posterior class membership probabilities for the new unlabeled (test) time series at each time t. We can see an illustrative example of the construction of the h_t classifiers in Figure 1, for a training set of 5 time series.

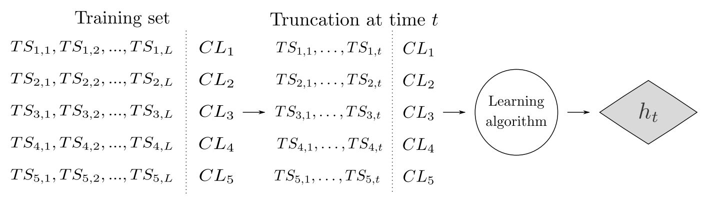
Training set
Truncation at time t
$TS_{1,1}, TS_{1,2}, ..., TS_{1,L}$ CL_1
$TS_{2,1}, TS_{2,2}, ..., TS_{2,L}$ CL_2
$TS_{3,1}, TS_{3,2}, ..., TS_{3,L}$ CL_3
$TS_{4,1}, TS_{4,2}, ..., TS_{4,L}$ CL_4
$TS_{5,1}, TS_{5,2}, ..., TS_{5,L}$ CL_5
$TS_{1,1}, \ldots, TS_{1,t}$ CL_1
$TS_{2,1}, \ldots, TS_{2,t}$ CL_2
$TS_{3,1}, \ldots, TS_{3,t}$ CL_3
$TS_{4,1}, \ldots, TS_{4,t}$ CL_4
$TS_{5,1}, \ldots, TS_{5,t}$ CL_5
Learning algorithm
h_t

Figure 1: Construction of the h_t classifiers.

- On the other hand, we will build another set of classifiers $\{h'_{t,i} \mid t = 1, ..., L \ , \ i = 1, 2, ...5.\}$, similar to h_t but trained using a sort of 5-fold cross-validation scheme. We will use these classifiers to obtain the posterior probabilities of the training examples, but with the intention of avoiding the overfitting phenomenon. We can see an illustrative example of the construction of the $h'_{t,i}$ classifiers in Figure 2, for a training set of 5 time series. Based on this contruction, to obtain the posterior probabilities of a given training series TS at time t, we will use the $h'_{t,i}$ classifier, which has been trained without using series TS.

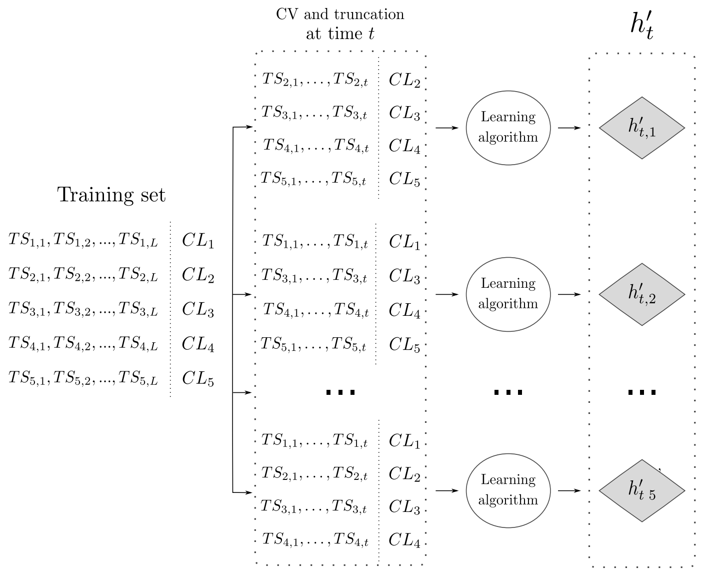
CV and truncation at time t
h'_t
Training set
$TS_{1,1}, TS_{1,2}, ..., TS_{1,L}$ CL_1
$TS_{2,1}, TS_{2,2}, ..., TS_{2,L}$ CL_2
$TS_{3,1}, TS_{3,2}, ..., TS_{3,L}$ CL_3
$TS_{4,1}, TS_{4,2}, ..., TS_{4,L}$ CL_4
$TS_{5,1}, TS_{5,2}, ..., TS_{5,L}$ CL_5
$TS_{2,1}, \ldots, TS_{2,t}$ CL_2
$TS_{3,1}, \ldots, TS_{3,t}$ CL_3
$TS_{4,1}, \ldots, TS_{4,t}$ CL_4
$TS_{5,1}, \ldots, TS_{5,t}$ CL_5
$TS_{1,1}, \ldots, TS_{1,t}$ CL_1
$TS_{3,1}, \ldots, TS_{3,t}$ CL_3
$TS_{4,1}, \ldots, TS_{4,t}$ CL_4
$TS_{5,1}, \ldots, TS_{5,t}$ CL_5
$TS_{1,1}, \ldots, TS_{1,t}$ CL_1
$TS_{2,1}, \ldots, TS_{2,t}$ CL_2
$TS_{3,1}, \ldots, TS_{3,t}$ CL_3
$TS_{4,1}, \ldots, TS_{4,t}$ CL_4
Learning algorithm
Learning algorithm
Learning algorithm
$h'_{t,1}$
$h'_{t,2}$
$h'_{t\ 5}$
...
...
...

Figure 2: Construction of the h'_t classifiers.

Step 2: Definition of stopping rules

In the second step, we define the stopping rules and the associated cost functions. Specifically, we propose two different stopping rules, of different characteristics, whose performance will be analyzed in the experimental section.

As a first simple stopping rule, we analyze a basic stopping rule $SR1_\gamma(\cdot)$, based on intuition and defined by the following linear rule, proposed previously by Mori et al. [15]:

$$SR1_\gamma(\mathbf{p^t}, t) = \begin{cases} 0 & if \ \ \gamma_1 p^t_{1:k} + \gamma_2 (p^t_{1:k} - p^t_{2:k}) + \gamma_3 \frac{t}{L} \le 0 \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

$\mathbf{p^t} = (p^t_1, p^t_2, ..., p^t_k)$ is the vector of posterior probabilities for the k possible classes issued by the corresponding h_t for a given time series, and $p^t_{1:k}$ and $p^t_{2:k}$ are the first and second largest posterior probability values obtained at time t.

$\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \gamma_3)$ is a vector of parameters that takes real values between -1 and 1.

The interpretation of this stopping rule is the following: if the rule outputs a value of 1, we conclude that the prediction is reliable enough, and thus, the class corresponding to the maximum posterior probability value is provided. On the contrary, a value of 0 indicates that the prediction is not yet reliable, and so, we should wait until a larger part of the time series is available. If the entire time series is available and the stopping rule has not triggered, the class prediction obtained at $t = L$ is used.

As indicated above, the shape of the stopping rule has been chosen based on intuition. We assume that the value of $p_{1:k}$ and the difference between the two largest posterior probabilities are indicators of the goodness and reliability of the prediction obtained. As such, we include these two terms in the stopping rule. Furthermore, the time in which the prediction is made can also be an influencing factor, so we also include this parameter in the stopping rule. Note that in posterior steps, the γ_i parameters in this stopping rule will be optimized using a suitable cost function and so, each of the components of the stopping rule will be given more or less relevance, depending on their corresponding γ_i value.

As a second and novel stopping rule, we avoid defining the shape of the stopping rule a priori and we introduce all the posterior probabilities ($SR2_\gamma$):

$$SR2_\gamma(\mathbf{p^t}, t) = \begin{cases} 0 & if \;\; \gamma_1 p^t_{1:k} + \cdots + \gamma_k p^t_{k:k} + \gamma_{k+1} \frac{t}{L} \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

Step 3: Definition of the cost function

Recall that the final aim is to find an optimized stopping rule, which takes earliness and accuracy into account. For this, departing from the general stopping rules defined in the previous step, we will try to find the $\boldsymbol{\gamma}$ parameters that minimize a given cost function, based on accuracy and earliness. In this section we will propose various cost functions of different characteristics, which will be compared later in the experimental section.

To begin with, we propose the following basic cost function proposed initially in [15]:

$$CF1(X, SR_\gamma) = \sum_{x \in X} CF1(x, SR_\gamma) = \sum_{x \in X} (\alpha C_a(x, SR_\gamma) + (1-\alpha) C_e(x, SR_\gamma)) \tag{4}$$

where $\alpha \in [0, 1]$ is a user-defined parameter that represents the weight associated to each of the objectives, and C_a and C_e are the cost of accuracy and earliness, and will be presented at the end of this section.

Additionally, when we are using the SR2 stopping rule, we would like our optimization process to select some of the posterior probabilities from the stopping rule automatically, based on their relevance. This means that we want to reward simpler models that will have as many 0-s in the $\boldsymbol{\gamma}$ vector as possible.

In optimization, this is denominated regularizing the cost function for sparsity in the parameter vector.

The most direct manner of rewarding this sparsity is introducing a non-convex penalization term to the basic cost function, based on the 0-norm of the chosen parameters. This term will penalize the vectors of parameters that have many non-zero components and will result in the following new cost function (CF2):

$$CF2(X, SR_\gamma) = \sum_{x \in X} CF2(x, SR_\gamma) = \sum_{x \in X} (\alpha C_a(x, SR_\gamma) + (1-\alpha) C_e(x, SR_\gamma)) - \lambda \left\| \gamma \right\|_0 \quad (5)$$

Note that this penalization term is non-convex and its application results in NP-hard optimization problems. As such, in practice, it is common to use a relaxed but convex penalization term for sparsity, based on the $\|\gamma\|_1$ norm. This is typically done in the basic LASSO method, where this penalization term is added to the classic least squares formulation for regression [21]. The introduction of this penalization term in the cost function will result in the third cost function that will be considered in this work:

$$CF3(X, SR_\gamma) = \sum_{x \in X} CF3(x, SR_\gamma) = \sum_{x \in X} (\alpha C_a(x, SR_\gamma) + (1-\alpha) C_e(x, SR_\gamma)) - \lambda \left\| \gamma \right\|_1 \quad (6)$$

The functions C_a and C_e, which appear in all the cost functions, can be defined in several ways. In this case:

$$C_e(x, s_\gamma) = \frac{t_x^*}{L} \quad (7)$$

t_x^* being the earliest timestamp in which $s_\gamma(\cdot)$ outputs a value of 1 (halt) for series x, and

$$C_a(x, \gamma) = \mathbb{I}(\text{argmax}_{i=1,\ldots,k}\{p_{x,i}^{t_x^*}\} \neq CL_x) \quad (8)$$

where $\text{argmax}_{i=1,\ldots,k}\{p_{x,i}^{t_x^*}\}$ is the class with highest predicted probability value at instant t_x^* for time series x, and CL_x its true class value. $\mathbb{I}(\cdot)$ takes a value of 1 if the condition is true, and 0 otherwise. In essence, this is the classical 0-1 cost function, based on comparing the true label with the label obtained at time t_x^*.

Based on the two stopping rules and the three cost functions we obtain the following combinations: SR1-CF1, SR2-CF1, SR2-CF2, SR2-CF3.

Step 4: Optimization process

The $\boldsymbol{\gamma}$ vector that minimizes this cost function can be found by using several different optimization algorithms. Of course, we must take into account the nature of the defined cost function. Specifically, the non-convexity of the cost function and its lack of differentiability, among other features, will be determinant when choosing the optimization algorithm. The specific algorithms chosen in this work will be introduced in the experimental section, but note that other optimization algorithms which do not require convexity or differentiability of the cost function could be used without loss of generality.

In order to apply any optimization algorithm, we have to be able to, at least, evaluate the objective function at different points. Note that the objective function is defined as the cost associated to different $\boldsymbol{\gamma}$ values. Given a training set of time series (X) for which the true class value is known, in Figure 3, we can see an example of how the cost associated to a given training example (x) would be computed for a certain stopping rule (SR) and a specific parameter vector ($\boldsymbol{\gamma}$). Consequently, the cost of the whole training set is calculated by summing the costs of all its conforming time series.

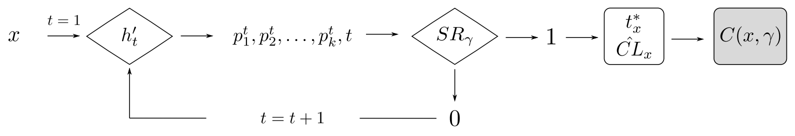

Figure 3: Evaluation of the cost function for a certain training example x.

As can be seen, starting from timestamp $t = 1$, the posterior probabilities for the training examples are obtained by using the corresponding h'_t probabilistic classifiers, as explained previously. These posterior probabilities are introduced into the stopping rule, which is completely defined by the chosen $\boldsymbol{\gamma}$ vector. If the stopping rule returns a value of 1, the process terminates and the current class prediction ($\hat{CL}_x = \text{argmax}_{i=1,\ldots,k}\{p_{x,i}^{t_x^*}\}$) and timestamp ($t_x^*$) will be used directly to calculate the cost. If the stopping rule outputs a 0, then we must add the next data point to the time series and repeat the process.

3.2 Prediction step

Finally, once the framework has been trained, we can use it to predict the class labels of new time series. As can be seen in Figure 4, each time a new data point of the new time series becomes available, we will introduce the truncated series into the corresponding h_t classifier specific for that t and trained with all the training set X as explained before. This classifier, in turn, will output a vector of posterior probabilities for that time series and that specific time stamp t. Next, this probability vector will be introduced into the optimized stopping rule, together with the current time stamp. The stopping rule will provide an answer of 0 or 1, and, based on this, we will halt and provide a class prediction, or, on the contrary, wait until more data is available.

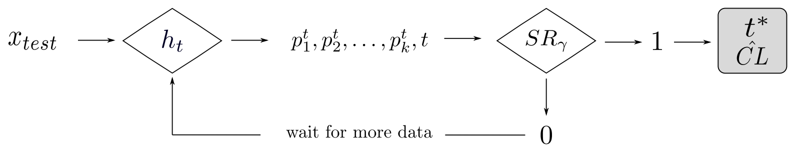

Figure 4: Early classification of a test example x_{test}.

Note that, for each database, the accuracy evolves differently as more data becomes available. As shown in [14], in some databases, the accuracy is strictly increasing. In others, in contrast, the accuracy becomes stable after some point, and in some databases, the accuracy even drops after some point due to the noise that the additional data incorporates. The proposed method has been designed to be very flexible, and does not require the stability or convergence of the stopping rules, as do other methods in the literature such as [9, 14]. This enables a better adjustment to the different forms that the accuracy can take over time and allows the method to profit more from early class predictions.

4 Experimental setup

In this section we detail the setting of the experiments carried out to evaluate our method [3].

4.1 The data

To evaluate our proposal, we have considered the 45 databases available at the time of experimentation from the UCR time series database archive [22]. This archive collects the majority of the publicly available synthetic and real time series databases designed for clustering and classification purposes, and is the classical benchmark used for evaluation in this study area.

4.2 Parameter selection

As aforementioned, in Section 4, the h_t and h'_t classifiers can be trained in every timestamp, $t = \{1, ..., L\}$, but a smaller set of timestamps can also be chosen. In this case, due to the large number of databases considered and because we want to control the computational cost of the experimentation, we build the classifiers only with a granularity of 5% of the length of time series. However, some additional experiments that consider other granularities and analyze the effect of this parameter are provided in Section 5.2.

To build these two sets of probabilistic classifiers we have used Gaussian Process (GP) classifiers [23], because they have shown good performance in similar frameworks [14, 15]. Nevertheless, any other probabilistic model could be used

[3]The complete code of the proposed methodology will be made available in our web page when the last version of the paper is ready.

Method	**Variants**	**Parameter name**	**Values**
ECTS [9]	-Strict -Loose	Minimum support	0, 0.05, 0.1, 0.2, 0.4, 0.8
EDSC [18]	Chebyshev Inequality version	Chebyshev bound	2.5, 3, 3.5
RelClass [16]	-Naive Gaussian Quadratic set -Gaussian Naive Bayes box	Reliability threshold τ	0.001, 0.1, 0.5, 0.9
ECDIRE		*acc_perc*	100%

Table 1: Combinations and variants of the comparison methods.

equivalently within our framework. Indeed, in order to obtain some additional insights, and to analyze the effect of using another alternative probabilistic classifier, in Section 5.4, we show the results of some experiments using Support Vector Machines (SVM) combined with Platt's scaling [24].

The GPs are implemented using an extension of the *vbmp* package of R [25] previously used in [14]. Similarly to this work, the parameters for the *vbmp* function have all been set to their default values except for the convergence threshold (set to 10^{-8}), and the maximum iteration number (set to 24). Finally, after preliminary experiments, the inner product kernel has been chosen as the covariance function. On the contrary, the SVM classifiers have been trained using the *e1071* R package [26], using the Gaussian kernel and leaving the remaining parameters in their default values.

In previous studies, the use of specific distance measures for time series, such as Dynamic Time Warping or Edit distances has shown to improve the results of classification algorithms when dealing with time series classification [27]. As such, following the example of [14] and [28], we enable the use of different distance measures within the probabilistic classifiers, by using a suitable distance matrix as input to the classifiers, instead of the raw time series. This distance matrix will save the pairwise distances between all the series in the training set, and can be built using any distance measure of choice which enables us to deal with discrete series and series of different lengths, simply by choosing a suitable distance matrix. However, since the objective is not to evaluate the performance of different distance measures, in this experimentation we always use the standard Euclidean Distance.

Regarding the α parameter, which appears in the cost functions defined in Section 3.1, we analyze 4 weight values ($\alpha \in \{0.6, 0.7, 0.8, 0.9\}$). The reason why we only choose α values higher that 0.5 is that the other early classification methods that we will use for comparison (see next Section) aim at obtaining the same accuracy that is obtained when the full series is available. In this context, we will also lean towards this objective. However, in Section 5.1 we will analyze the effect of modifying this parameter.

As mentioned previously, different optimization algorithms can be used within the presented framework, always taking into account the properties of the defined cost function. In our case, since we can not calculate an analytic expression of the gradient of the cost function, and we do not always have information about its convexity, we have chosen to use Genetic Algorithms [29]. These algorithms make few assumptions about the search space and are suitable to solve many kinds of complex optimization problems. As such, we can also use this optimization algorithm when the shape of the cost function is unknown or non-convex and, so, the optimization problem can not be solved efficiently by standard algorithms. We implement the Genetic Algorithm using the *ga* function of the *GA* package written in R [30] using the default parameter values of this function. The population size is 50 and the initial population is chosen uniformly at random. With regard to the genetic operators, at each iteration, a whole new population is created by first applying selection, carried out by using fitness proportional selection with linear scaling [31]. Then, using the local arithmetic crossover operator, these solutions are recombined with a probability of 0.8 [32], and mutated with a probability of 0.1 using the nonuniform random mutation operator [33]. The individuals created after applying these operations will conform the new population, but 5% of the best fitted individuals from the initial population will survive at each generation and will replace the worst 5% obtained from the genetic operations. Finally, the algorithm is halted after 100 iterations.

Since this algorithm is a randomized heuristic, for each database and α value, we carry out 30 executions and select the $\boldsymbol{\gamma}$ value that obtains the median of the costs obtained in the 30 repetitions.

As special cases, in combinations SR2-CF2 and SR2-CF3 when we apply regularization for sparsity, we experiment with the values $\lambda \in \{0, 0.1, 0.5, 1, 5, 10, 50\}$ in the first step. Then, for each λ we save the result ($\boldsymbol{\gamma}$) that obtains the median of the costs obtained in the 30 repetitions. Then, we choose the λ that minimizes the cost within the training set. We are aware that this may cause overfitting, but, for the sake of simplicity, we choose this selection method to avoid adding another external cross-validation process to the framework.

4.3 Comparison to other early classification methods

To validate our proposal, we have compared its performance with the four state-of-the-art early classification methods which, to the best of our knowledge, have available source codes [4] [5] [6]. A summary of the methods chosen and the parameter configurations experimented with are summarized in Table 1. Note that this selection has been made based on the experimentation carried out by the corresponding authors.

Note that, additionally, in the RelClass method, the local discriminative Gaussian dimensionality reduction has been enabled because it reduces computational costs and can reduce noise and yield higher accuracy values [16], and the joint Gaussian estimation method has been chosen because it is more

[4]**ECTS and EDSC:** http://zhengzhengxing.blogspot.com.es/p/research.html
[5]**Rel.Class.**: http://www.mayagupta.org/publications/Early_Classification_For_Web.zip
[6]**ECDIRE**: http://www.sc.ehu.es/ccwbayes/members/umori/ECDIRE/ECDIRE.html

efficient and obtains similar results to those obtained by the other estimation methods considered.

4.4 Evaluation method

The databases from the UCR archive are provided with pre-specified training and testing sets, which have been used directly to enable reproducibility and comparison. Furthermore, the two objectives of the early classification problem are evaluated separately based on the previously defined cost of accuracy (C_a) and cost of earliness (C_e):

$$Accuracy = \frac{1}{|X_{test}|} \sum_{x \in X_{test}} \mathbb{I}(\text{argmax}_{i=1,\dots,k}\{p_{x,i}^{t_x^*}\} = CL_x) \tag{9}$$

$$Earliness = \frac{1}{|X_{test}|} \sum_{x \in X_{test}} \frac{t_x^*}{L} \cdot 100\% \tag{10}$$

In addition to measuring and comparing these two objectives separately, we must consider the multi-objectiveness of the problem. As such, we use the Pareto optimality criterion as in [14], which states that a solution dominates another if it obtains better results in at least one of the objectives while not degrading any of the others. We compare the methods pairwise and calculate the number of times in which our method dominates the other and vice versa.

Based on the domination counts, we also calculate two summary indices: *Summary 1* is calculated by assigning one point for each database in which our method wins and substracting one point for each time it loses; *Summary 2* is calculated by assigning one point for winning, half point for each draw and 0 points for losing.

5 Results

In Table 2, we show the domination counts for each configuration of our method (SR1-CF1, SR2-CF1, SR2-CF2, SR2-CF3) compared to the other state-of-the-art methods (ECDIRE, Rel.Class., ECTS, EDSC) [7]. We also show the scores *Summary 1* and *Summary 2*, for each combination of our method and, also, average values for each α value.

Note that, for ECDIRE, RelClass, ECTS and EDSC, from all the parameter combinations enumerated in Table 1, which we have considered in the experimentation, we only show the results for the configuration that dominates our method the most times. If there are ties within a method, the configuration that is dominated a lower number of times by our method is chosen. Anyhow,

[7] After a month of computation, the EDSC method has not provided any solutions for the *StarLightCurves*, *NonInvasiveFatalECG_Thorax1* and *NonInvasiveFatalECG_Thorax2* databases, so these results are not considered in the domination counts calculated for this method.

the raw accuracy and earliness values obtained by all the methods and all the parameter configurations can be found in our website [8].

As we can see, all four proposed methods dominate the rest of the methods by a large margin, proving that our methodology is an effective way of performing early classification. Furthermore, if we analyze the statistical significance of these results using a signed permutation test [34] (corrected with the Holm-Bonferroni post-hoc test to control the family-wise error), all the p-values that we obtain except SR2-CF1 or SR2-CF3 with $\alpha = 0.6$ compared to Rel.Class., indicate significant differences for a significance level of 0.05, supporting the previous statement further. The p-values issued from these statistical tests are

[8]http://www.sc.ehu.es/ccwbayes/members/umori/EarlyClassification/EarlyClassification.html

$\alpha = 0.6$	ECDIRE	Rel.Class.	ECTS	EDSC	Summary1	Summary2
SR1-CF1	15/30/0	11/33/1	21/23/1	29/16/0	18.50	31.75
SR2-CF1	18/27/0	7/37/1	18/26/1	29/16/0	17.50	31.25
SR2-CF2	16/29/0	9/35/1	18/26/1	29/16/0	17.50	31.25
SR2-CF3	18/27/0	7/37/1	18/26/1	29/16/0	17.50	31.25
					17.75	31.38

$\alpha = 0.7$	ECDIRE	Rel.Class.	ECTS	EDSC	Summary1	Summary2
SR1-CF1	18/27/0	12/31/2	24/20/1	31/13/1	20.25	32.63
SR2-CF1	20/25/0	14/29/2	23/21/1	30/14/1	20.75	32.88
SR2-CF2	21/24/0	15/28/2	25/19/1	30/14/1	21.75	33.38
SR2-CF3	20/25/0	14/29/2	23/21/1	30/14/1	20.75	32.88
					20.88	32.94

$\alpha = 0.8$	ECDIRE	Rel.Class.	ECTS	EDSC	Summary1	Summary2
SR1-CF1	28/17/0	22/21/2	28/15/2	31/14/0	26.25	35.63
SR2-CF1	25/20/0	20/23/2	29/15/1	30/15/0	25.25	35.13
SR2-CF2	27/18/0	22/21/2	29/15/1	31/14/0	26.50	35.75
SR2-CF3	25/20/0	20/23/2	29/15/1	30/15/0	25.25	35.13
					25.81	35.41

$\alpha = 0.9$	ECDIRE	Rel.Class.	ECTS	EDSC	Summary1	Summary2
SR1-CF1	28/17/0	23/19/3	29/15/1	28/17/0	26.00	35.50
SR2-CF1	24/20/1	23/19/3	32/11/2	28/17/0	25.25	35.13
SR2-CF2	22/22/1	24/19/2	32/11/2	28/17/0	25.25	35.13
SR2-CF3	24/20/1	23/19/3	32/11/2	28/17/0	25.25	35.13
					25.44	35.22

Table 2: Domination counts for our method with GP-s in comparison to ECDIRE, ECTS, EDSC and Rel.Class. The first number corresponds to the number of times our method dominates the other method, the second number refers to the draws and the third number counts the times the comparison method dominates our method.

available at [9]. As such, these results indicate that our method, by means of the posterior probabilities obtained from the classifiers and the optimization process, is able to analyze how the accuracy evolves as more data becomes available, and is capable of identifying when the best moment to stop and provide a prediction is.

Additionally, if we study the domination counts and also the values obtained by the two summary indexes (*Summary1* and *Summary2*), we can see that, overall, $\alpha = 0.8$ seems to achieve the most impressive differences when compared to the other state-of-the-art methods.

If we compare the different combinations of our method, combinations SR1-CF1 and SR2-CF2 obtain the highest values for *Summary1* and *Summary2*, being the former the winner when α is 0.6 or 0.9 and the latter when it is 0.7 or 0.8.

After analyzing these general results, in the next sections we will provide a more detailed analysis and discussion of the effect of modifying the different parameters of the framework: the α parameter, the sampling granularity of the series, the stopping rule, the cost function and the underlying classifier.

5.1 Effect of modifying α

Based on the formulation of the cost functions, when we increase α, we expect the accuracy to increase whereas the earliness values should become worse. To show that this is indeed so, in Figure 5, we plot the mean value (averaged over all databases) of the accuracy and earliness for different α-s.

As we can see, the trend of the plots is clearly increasing, for all four proposed methods. In this sense, if accuracy requirements were lower, we could use other smaller α values.

In any case, as commented previously, if we analyze the domination counts obtained by different α values in comparison to the rest of the state-of-the-art methods (see Table 2), it seems that the best results are obtained by larger α-s. This is somewhat expected, since all the rest of the methods aim at obtaining 100% accuracy. Specifically, if we consider the values of the performance scores *Summary1* and *Summary2*, both considering each stopping rule and cost function combination separately, and also averaging over them, the best results are obtained by $\alpha = 0.8$. As such, from now on, we will only take into account this value of α.

However, note that the selection of the most suitable α can be made following other criteria and, it strongly depends on the database and also on the interest or requirements of the user in terms of accuracy and earliness.

[9]http://www.sc.ehu.es/ccwbayes/members/umori/EarlyClassification/pvalues.pdf

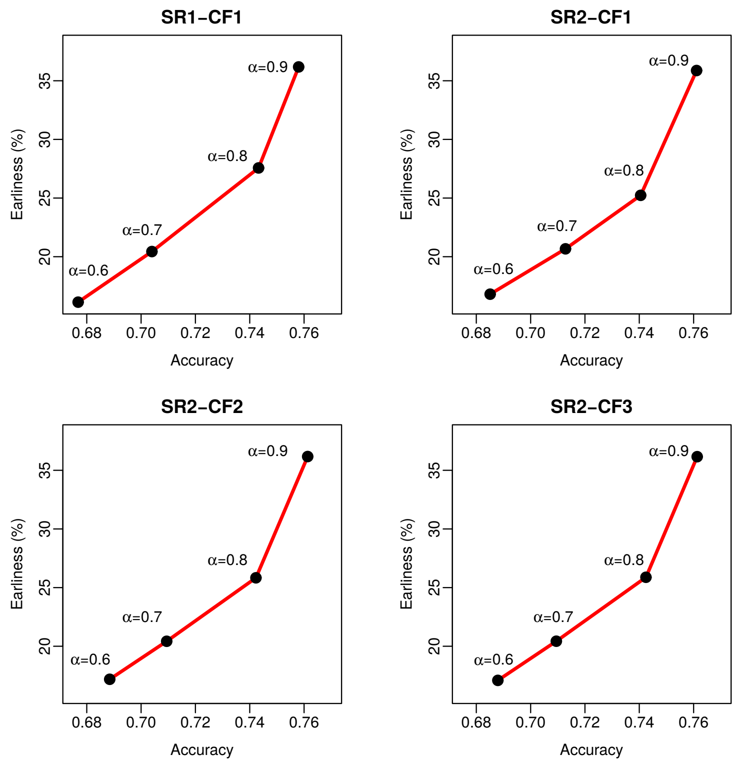

Figure 5: Evolution of accuracy and earliness according to α.

5.2 Effect of modifying the granularity of the sampling

The sampling granularity of the time series, which has been set to 5%, is also a parameter of the model. As such, with the aim of analyzing to what extent the results are affected by this parameter, we have carried out some experiments based on the SR1-CF1 combination and $\alpha = 0.8$, but using different granularities. The accuracy and earliness summaries can be found in Figure 6, and the domination counts when compared to the other state-of-the-art methods are shown in Table 3.

As can be seen, there are no large differences between the obtained accuracies and earliness values. When the granularity is increased, the earliness results become a bit better, of course, in detriment of the accuracy. However, the domination results clearly show that our method still outperforms the rest by a large margin with all four granularity values we have chosen.

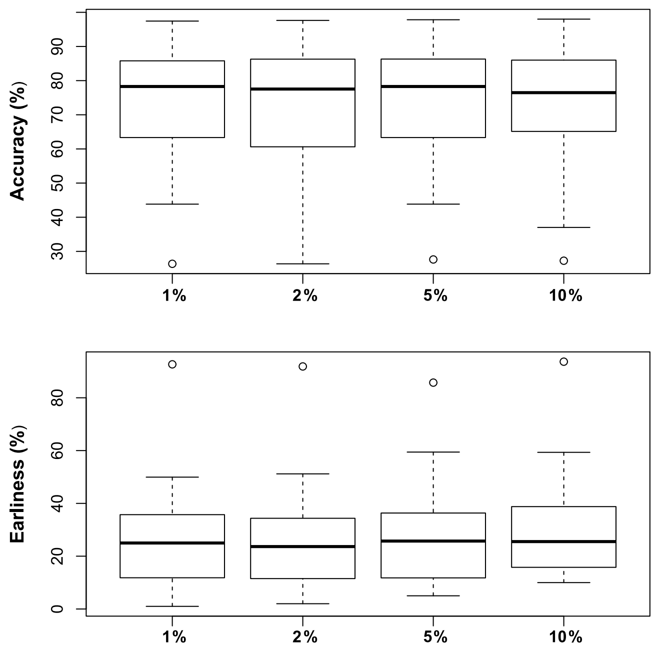

Figure 6: Accuracy and earliness values for the considered databases, depending on the sampling granularity.

$\alpha = 0.8$	ECDIRE	Rel.Class.	ECTS	EDSC
granularity=1%	26/19/0	22/22/1	28/15/2	31/14/0
granularity=2%	27/18/0	23/21/1	27/16/2	32/13/0
granularity=5%	27/18/0	23/20/2	30/14/1	32/13/0
granularity=10%	21/23/1	18/25/2	28/15/2	32/13/0

Table 3: Domination counts for SR1-CF1 with GP-s and different sampling granularities in comparison to ECDIRE, ECTS, EDSC and Rel.Class.

5.3 Effect of modifying the cost function and the stopping rule

In order to analyze the effect of modifying the cost function and stopping rule, we select $\alpha = 0.8$ as explained previously and, in Table 4, we compare the results obtained by the 4 configurations of our method.

Firstly, if we analyze the effect of the different stopping rules, we must emphasize that the simplest stopping rule (SR1) obtains quite good results when comparing it with other state-of-the-art methods (see Table 2). However, it performs a little bit worse than the other combinations of our own method. In Table 4, we can see that the SR2 rule obtains slightly higher domination counts than SR1, for all three cost functions considered. Even if these differences

$\alpha = 0.8$	SR1-CF1	SR2-CF1	SR2-CF2	SR2-CF3
SR1-CF1	-	7 /28/10	7/28/10	7 /28/10
SR2-CF1	10/28/7	-	12/23/10	0/43/2
SR2-CF2	10/28/7	10/23/12	-	10/21/14
SR2-CF3	10/28/7	2/43/0	14/21/10	-

Table 4: Domination counts for SR1-CF1, SR2-CF1, SR2-CF2 and SR2-CF3. The first number corresponds to the times the method in the corresponding row wins, the second number counts the draws and the third number the number of times the method in the corresponding column wins.

are not statistically significant, it seems that using SR2 better adjusts to the characteristics of some databases and is thus a better choice, in general.

Secondly, if we analyze the different cost functions by comparing the results obtained by SR2-CF1, SR2-CF2 and SR2-CF3, we conclude that applying regularization improves the results slightly. With $\alpha = 0.8$, SR2-CF2 obtains the best results when compared to other state-of-the-art algorithms (see Table 2). This is because it obtains very good earliness values, while maintaining competitive results on accuracy. Contrarily, SR2-CF3 obtains results almost identical to those obtained by its unregularized version (SR2-CF1), but it improves on the earliness in two datasets. SR2-CF3 tends more towards accuracy than SR2-CF2, which results in higher domination counts for the former when we compare these two configurations among each other (see Table 4). However, SR2-CF2 seems to benefit more from the regularization that SR2-CF3.

5.4 Effect of modifying the underlying classifier

In order to compare the performance of GP and the SVM classifiers within our framework, we choose the most simple approach (SR1-CF1) and, in Table 5, we show the domination counts of our method, using SVMs as underlying classifiers instead of GPs. Based on these results, it is evident that the GP classifiers obtain much better results.

SR1-CF1	ECDIRE	Rel.Class.	ECTS	EDSC
$\alpha = 0.6$	3/41/1	4/39/2	6/37/2	22/23/0
$\alpha = 0.7$	3/41/1	4/38/3	6/37/2	24/21/0
$\alpha = 0.8$	5/34/6	4/36/5	7/36/2	25/17/3
$\alpha = 0.9$	5/31/9	2/36/7	10/33/2	20/23/2

Table 5: Domination counts for our method using SVM-s with Platt's scaling in comparison to ECDIRE, Rel.Class., ECTS and EDSC. The first number corresponds to the number of times our method dominates the other method, the second number refers to the draws and the third number counts the times the comparison method dominates our method.

As expected, our framework is limited to the prediction ability of the chosen underlying classifiers, since they will set bounds to the maximum accuracy that can be obtained. In this case, the SVMs do not yield high accuracy results, and as such, even when we increase the value of α, the domination values remain quite low, especially when we compete with methods that obtain high accuracy values. However, note that even when using SVM classifiers, our method dominates the rest of the comparison methods more times that it is dominated in most cases, which proves the usefulness of our method.

6 Conclusions and Future Work

In this work we have proposed an early classification framework based on combining a set of probabilistic classifiers and a stopping rule, designed by minimizing the cost in earliness and accuracy. The method is conceptually simple and does not require complex parameter settings. Furthermore, it is one of the few approaches that tackles the problem of early classification from an optimization point of view.

We have experimented with different cost functions conformed with these two objectives and also with different stopping rules by using 45 benchmark databases from the UCR archive. We have also compared our results to other early classification methods from the state-of-the-art, showing superior results in terms of earliness and accuracy.

As future work, we propose designing a more complex structure for the stopping rule, in which a different stopping rule is associated to each class or more common reliability measures are used. The idea is to capture the different behaviors of the classes, if they exist. Additionally, more complex methods, such as genetic programming, could be used to learn more suitable stopping rules in a more automatic manner.

Finally, the problem of early classification has two conflicting objectives. In this paper, the balance between these two objectives has been sought by means of the α parameter, and we have emphasized that the choice of a suitable α strongly depends on the database at hand, and also on the requirements and needs of the user. In this sense, providing strategies to optimize this parameter in some specific context could be an interesting future research line. Additionally, and in this same line, it could be interesting to approach this problem as a multi-objective optimization problem, in which no α parameter would be necessary.

Aknowledgements

We would like to thank the UCR archive for providing access to the data used in this study. This work has been partially supported by the Research Groups 2013-2018 (IT-609-13) programs (Basque Government), TIN2016-78365-R (Spanish Ministry of Economy, Industry and Competitiveness) and Severo Ochoa Program SEV-2013-0323 (Spanish Ministry of Economy, Industry and Competitiveness).

References

[1] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 12(1):40–48, 2010.

[2] W. Pei, H. Dibekliolu, D. M. J. Tax, and L. van der Maaten. Multivariate time-series classification using the hidden-unit logistic model. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–12, 2017.

[3] Jason Lines, Anthony Bagnall, Patrick Caiger-Smith, and Simon Anderson. Classification of household devices by electricity usage profiles. In *Intelligent Data Engineering and Automated Learning - IDEAL 2011*, volume 6936 LNCS, pages 403–412, 2011.

[4] Mohammed Waleed Kadous and Claude Sammut. Classification of multivariate time series and structured data using constructive induction. *Machine Learning*, 58(2-3):179–216, 2005.

[5] Lexiang Ye and Eamonn Keogh. Time series shapelets: A novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 22(1-2):149–182, 2011.

[6] Chuanjun Li, Latifur Khan, and Balakrishnan Prabhakaran. Feature selection for classification of variable length multiattribute motions. *Knowledge and Information Systems*, 10(2):163–183, 2006.

[7] Li Wei and Eamonn Keogh. Semi-Supervised Time Series Classification. In *KDD 06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 748–753, 2006.

[8] Mabel González, Christoph Bergmeir, Isaac Triguero, Yanet Rodríguez, and José M Benítez. On the stopping criteria for k-nearest neighbor in positive unlabeled time series classification problems. *Information Sciences*, 328(C):42–59, January 2016.

[9] Zhengzheng Xing, Jian Pei, and Philip S. Yu. Early classification on time series. *Knowledge and Information Systems*, 31(1):105–127, apr 2011.

[10] R Scott Evans, Kathryn G Kuttler, Kathy J Simpson, Stephen Howe, Peter F Crossno, Kyle V Johnson, Misty N Schreiner, James F Lloyd, William H Tettelbach, Roger K Keddington, Alden Tanner, Chelbi Wilde, and Terry P Clemmer. Automated detection of physiologic deterioration in hospitalized patients. *Journal of the American Medical Informatics Association : JAMIA*, 22(2):350–60, 2015.

[11] Mohamed F. Ghalwash, Vladan Radosavljevic, and Zoran Obradovic. Utilizing temporal patterns for estimating uncertainty in interpretable early decision making. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '14*, pages 402–411, New York, New York, USA, 2014. ACM Press.

[12] Aníbal Bregón, M Aránzazu Simón, Juan José Rodríguez, Carlos Alonso, Belarmino Pulido, and Isaac Moro. Early Fault Classification in Dynamic Systems Using Case-Based Reasoning. In *Proceeding CAEPIA'05 Proceedings of the 11th Spanish association conference on Current Topics in Artificial Intelligence*, pages 211–220, 2006.

[13] Nima Hatami and Camelia Chira. Classifiers With a Reject Option for Early Time-Series Classification. In *IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, pages 9–16, 2013.

[14] Usue Mori, Alexander Mendiburu, Eamonn Keogh, and Jose A Lozano. Reliable early classification of time series based on discriminating the classes over time. *Data Mining and Knowledge Discovery*, pages 1–31, 2016.

[15] Usue Mori, Alexander Mendiburu, Sanjoy Dasgupta, and Jose A. Lozano. Early classification of time series from a cost minimization point of view. In *Neural Information Processing Systems. Time Series Workshop*, NIPS 2015, 2015.

[16] Nathan Parrish, Hyrum S Anderson, and Dun Yu Hsiao. Classifying With Confidence From Incomplete Information. *Journal of Machine Learning Research*, 14:3561–3589, 2013.

[17] Mohamed F. Ghalwash, Dusan Ramljak, and Zoran Obradovic. Early classification of multivariate time series using a hybrid HMM/SVM model. In *IEEE International Conference on Bioinformatics and Biomedicine*, pages 1–6, oct 2012.

[18] Zhengzheng Xing, Philip S Yu, and Ke Wang. Extracting Interpretable Features for Early Classification on Time Series. In *Proceedings of the Eleventh SIAM International Conference on Data Mining*, pages 247–258, 2011.

[19] Guoliang He, Yong Duan, Rong Peng, Xiaoyuan Jing, Tieyun Qian, and Lingling Wang. Early classification on multivariate time series. *Neurocomputing*, 149:777–787, feb 2015.

[20] Asma Dachraoui and Alexis Bondu. Early Classification of Time Series as a Non Myopic Sequential Decision Making Problem. In *ECML PKDD 2015*, volume Part I, pages 433–447, 2015.

[21] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Monthly Weather Review*, 78:1–3, 1996.

[22] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullahand Mueen, and Gustavo. Batista. The UCR Time Series Classification/Clustering Homepage. URL `www.cs.ucr.edu/~eamonn/time_series_data/`.

[23] Carl Edward Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.

[24] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, pages 61–74. MIT Press, 1999.

[25] Nicola Lama and Mark Girolami. vbmp: Variational Bayesian Multinomial Probit Regression. R package version 1.34.0., 2014. URL `http://bioinformatics.oxfordjournals.org/cgi/content/short/btm535v1`.

[26] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, Friedrich Leisch, Chih-Chung Chang, and Chih-Chen Lin. e1071: Misc Functions of the Department of Statistics. R package version 1.6-4. 2014. URL `http://cran.r-project.org/web/packages/e1071`.

[27] P.-F. Marteau and S Gibet. On Recursive Edit Distance Kernels With Applications To Time Series Classification. *IEEE Transactions on Neural Networks and Learning Systems*, PP(6):1–13, 2014.

[28] Rohit J. Kate. Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, pages 283–312, 2015.

[29] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, USA, 1975.

[30] Luca Scrucca. GA: A Package for Genetic Algorithms in R. *Journal of Statistical Software*, 53(4), 2013.

[31] Back T. and Fogel D. *Evolutionary Computation 1: Basic Algorithms and Operators.* Decision Engineering. IOP Publishing Ltd., Bristol and Philadelphia, 2000.

[32] X. Yu and Gen M. *Introduction to Evolutionary Algorithms.* Decision Engineering. Springer-Verlag, London, 2010.

[33] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing.* Natural Computing. Springer-Verlag, Berlin, 2015.

[34] Stefano Bonnini, Livio Corain, Marco Marozzi, and Luigi Salmaso. *Nonparametric Hypothesis Testing.* Wiley.