Knowl Inf Syst (2012) 31:105–127
DOI 10.1007/s10115-011-0400-x

REGULAR PAPER

Early classification on time series

Zhengzheng Xing · Jian Pei · Philip S. Yu

Received: 6 July 2010 / Revised: 9 March 2011 / Accepted: 2 April 2011 /
Published online: 26 April 2011

Abstract In this paper, we formulate the problem of early classification of time series data, which is important in some time-sensitive applications such as health informatics. We introduce a novel concept of MPL (minimum prediction length) and develop ECTS (early classification on time series), an effective 1-nearest neighbor classification method. ECTS makes early predictions and at the same time retains the accuracy comparable with that of a 1NN classifier using the full-length time series. Our empirical study using benchmark time series data sets shows that ECTS works well on the real data sets where 1NN classification is effective.

Keywords Time series · Classification · Instance-based learning

1 Introduction

Early classification of time series data is critical in some time-sensitive applications. For example, a retrospective study of the clinical data of infants admitted to a neonatal intensive care unit [8] found that the infants, who were diagnosed with sepsis disease, had abnormal heart beating time series patterns 24 hours preceding the diagnosis. Monitoring the heartbeat time series data and classifying the time series data as early as possible may lead to earlier diagnosis and effective therapy. As another example, [1] showed that by only observing the

Z. Xing
Amazon.com Inc., Seattle, WA, USA

J. Pei (✉)
School of Computing Science, Simon Fraser University,
Burnaby, BC, Canada
e-mail: jpei@cs.sfu.ca

P. S. Yu
Department of Computer Science, University of Illinois at Chicago,
Chicago, IL, USA

first five packages of a TCP connection, the application associated with the traffic flow can be classified. The applications of online traffic can be identified without waiting for the TCP flow to end, since the most relevant information about the types of flow may be represented in the application protocol headers. Waiting for more byes may hurt the classification effectiveness due to over fitting noise. In other words, in many applications, capturing the right information is important, either the complete time series or not does not matter. Generally, early classification of sequences may have applications in anomaly detection, intrusion detection, health informatics, and process control.

However, constructing classifiers that are capable of early prediction is far from trivial. Most of the existing classification methods on sequences and time series extract features from full-length sequences and time series and do not consider the earliness of prediction. Moreover, in those traditional sequence/time series classifiers, the optimization goal is often set to maximize classification accuracy. However, in early prediction, the goal is to optimize the earliness as long as the classification accuracy is satisfactory. Therefore, in order to achieve early prediction, we have to reexamine the existing classification methods and develop new earliness-aware techniques.

An early classifier is expected to meet two requirements. First, an early classifier should be able to affirm the earliest time of reliable classification so that the early predictions can be used for further actions. Second, an early classifier should retain an accuracy comparable with that of a classifier using the full-length time series or some user-specified accuracy threshold.

In this paper, we tackle the problem of early prediction on time series data and develop a native method. Specially, we adopt the 1-nearest neighbor (1NN) approach, which has been well recognized as an effective classification method for time series data [11,16]. We introduce a novel concept of MPL (minimum prediction length) and develop ECTS (early classification on time series), an effective 1-nearest neighbor classification method that makes prediction early and at the same time retains an accuracy comparable with that of a 1NN classifier using the full-length time series. Our empirical study using benchmark time series data sets shows that ECTS works well where 1NN classification is effective.

The rest of the paper is organized as follows. In Sect. 2, we formulate the problem. We summarize related work in Sect. 3. We present a basic 1NN early classifier in Sect. 4 and develop the ECTS method in Sect. 5. We evaluate ECTS empirically in Sect. 6. Section 7 concludes the paper.

2 Problem definition

A *time series* s is a sequence of pairs $(timestamp, value)$. The data values are ordered in timestamp ascending order. We assume that all timestamps take positive integer values. We denote by $s[i]$ the value of time series s at timestamp i.

To keep our discussion simple, in this paper, we assume that all time series in question are of length L, i.e., each time series s has a value $s[i]$ at timestamp $1 \leq i \leq L$. L is called the *full length* of the time series. In practice, there are many cases the time series are of the same length, such as the real data sets in the UCR time series archive [12]. When the time series are not of the same length, alignments or dynamic warping can be applied as the preprocessing step.

For a time series s of length L, $s[i, j] = s[i]s[i + 1] \cdots s[j](1 \leq i < j \leq L)$ is the *subsequence* at timestamp interval $[i, j]$. Subsequence $s[1, l](l \leq L)$ is the *length-l prefix* of s.

For two time series s and s', $dist(s, s')$ denotes the *distance* between them. In this paper, we use the Euclidean distance

$$dist(s, s') = \sqrt{\sum_{i=1}^{L} (s[i] - s'[i])^2},$$

which is a simple yet effective and popularly adopted choice.

The set of all possible time series of length L is $\mathcal{R}^L$ and is called the *full-length space*, where $\mathcal{R}$ is the set of real numbers. The *prefix space of length-l*, denoted by $\mathcal{R}^l$, is the set of length-l prefixes of all possible time series.

In time series classification, a *training set* T contains a set of time series and a set of *class labels* $\mathcal{C}$ such that each time series $t \in T$ carries a class label $t.c \in \mathcal{C}$. The *time series classification problem* is to learn from T a classifier $C : \mathcal{R}^L \to \mathcal{C}$ such that for any time series s, C predicts the class label of s by $C(s)$. The performance of a classifier is often evaluated using a *test set* T', which is a set of time series such that each time series $t' \in T'$ also carries a class label $t'.c \in \mathcal{C}$. The *accuracy* of a classifier C on the test set T' is $Accuracy(C, T') = \frac{|\{C(t')=t'.c | t' \in T'\}|}{|T'|}$. Often, we want the classifier C as accurate as possible.

For a time series s, an early classifier C can identify an integer l_0 and make classification based on $s[1, l_0]$. An early classifier is *serial* [19] if $C(s[1, l_0]) = C(s[1, l_0 + i])$ for any $i > 0$. In other words, C can classify s based on only the prefix $s[1, l_0]$, and the classification remains the same by using any longer prefixes. An early classifier is preferred to be serial so that the early classification is reliable and consistent. The minimum length l_0 of the prefix based on which C makes the prediction is called the *cost* of the prediction, denoted by $Cost(C, s) = l_0$. Trivially, for any finite time series s, $Cost(C, s) \leq |s|$. The *cost* of the prediction on a test set T' is $Cost(C, T') = \frac{1}{|T'|} \sum_{t' \in T'} Cost(C, t')$.

Among many methods that can be used in time series classification, the 1-nearest neighbor (1NN) classifier has been found often accurate in practice [11,16]. The 1NN classification method is parameter free and does not require feature selection and discretization. Theoretically, [3] showed that the error rate of the 1NN classifier is at most twice that of the optimal Bayes probability when an infinite sample set is used.

Due to the effectiveness and the simplicity of the 1NN classifier on time series data, in this paper, we focus on extending the 1NN classifier for early classification on time series data. We use the 1NN classifier on full-length time series as the baseline for comparison. Ideally, we want to build a classifier which is as accurate as the baseline method and minimizes the expected prediction cost.

In general, it is possible to achieve earlier classification at the expense of accuracy. The trade-off between earliness and accuracy is an interesting and fundamental problem for early classification. As the first step to tackle this problem, the ECTS method to be developed in this paper stays at one end of the trade-off, that is, retaining the accuracy the same as that of the 1NN classifier. The other end, classification without accuracy, can be trivially achieved by random guesses such as a naïve Bayesian method. Such a method may even have a prediction length of 0. However, such a method is not very useful or meaningful in practice. How to achieve controllable trade-off between earliness and accuracy is a very challenging open problem beyond the scope of this paper.

We summarize the frequently used notations and abbreviations in Table 1.

Table 1 The frequently used notations and abbreviations

Abbreviation	Explanation
1NN	1-nearest neighbor
ECTS	Early Classification on Time Series
MLHC	multilevel hierarchical clustering
MPL	minimum prediction length
MPP	minimum prediction prefix
$NN^l(s)$	the set of the nearest neighbors of s in T in prefix space $\mathcal{R}^l$
$\mathcal{R}^l$	prefix space
$RNN^l(t)$	the set of reverse nearest neighbors in prefix space $\mathcal{R}^l$ of $t(1, l)$
$Sib(s)$	the sibling cluster of s

3 Related work

To the best of our knowledge, [15] are the first to mention the term "early classification of time series". They segment a time series into intervals and then described intervals using relative predicates, such as increases, decreases, and stays, and region-based predicates, such as sometime, always, true-percentage. Those predicates are used as features to construct simple base classifiers, each containing only one predicate. Ada boost [7] is used to ensemble the base classifiers. The ensemble classifier is capable of making predictions on incomplete data by viewing unavailable suffixes of sequences as missing features.

Bregon et al. (2006) applied a case-based reasoning method to classify time series to monitor the system failure in a simulated dynamic system. The KNN classifier is used to classify incomplete time series using various distances, such as Euclidean distance and Dynamic time warping (DTW) distance [17]. The simulation studies show that, by using case-based reasoning, the most important increase in classification accuracy occurs on the prefixes through thirty to fifty percent of the full length.

Although in the study by [2,15] the importance of early classification on time series is identified and some encouraging results are shown, the study only solved early classification as a problem of classifying prefixes of sequences. [19] pointed out the challenge of early classification is to study the trade-off between the earliness and the accuracy of classification. The methods proposed in study by [2,15] only focus on making predictions based on partial information but do not address how to select the shortest prefix to provide a reliable prediction. This makes the result of early classification not easily used by users for further actions

In our previous work [19], we formulated the early classification problem on symbolic sequence data as classifying sequences as early as possible while maintaining an expected accuracy. The major idea is to first select a set of features that are frequent, distinctive, and early and then build an association rule classifier or a decision tree classifier using those features. In the process of building the classifier, the user expected accuracy is satisfied by each association rule or each branch in the decision tree. In the classification step, an oncoming sequence is matched with all rules or branches simultaneously until on a prefix, a matching is found and the sequence is classified. In this way, a sequence is classified immediately once the user expected accuracy is achieved. The methods proposed in the study by [19] show some successes in handling symbolic sequences by achieving competitive accuracies using only less than half of the length of the full sequences.

Time series are numeric. To use our symbolic methods, time series have to be discretized properly. However, appropriate discretization often heavily relies on good background knowledge. Moreover, affected by the discretization granularity, the discretization-based methods may lose important information in time series data. Our previous study [19] shows that the symbolic methods do not work well on numeric time series data. Thus, early prediction on time series data, a type of data prevalent in time-sensitive applications, remains open at large.

Generally, most existing time series classification methods transform a time series into a set of features and then apply conventional classification methods on the feature vectors. To apply feature-based methods on simple time series, usually, before feature selection, time series data need to be transformed into symbolic sequences through discretization or symbolic transformation [14]. [13] proposed criteria for selecting features for symbolic sequence classification. [9] proposed an efficient algorithm to mine features from symbolic sequences with gap constraints. Without discretization, [22] proposed a method to find time series shapelets and use a decision tree to classify time series. [10] extracted a set of user-defined meta-features from time series. Some universal meta-features include the features to describe the trends of increases and decreases and local max or min values.

Different from feature-based classifiers, instance-based classifiers [11,16,18], such as 1NN classifiers, make predictions based on the similarities between the time series to be classified and the ones in the training set. A 1NN classifier is chosen as the benchmark classifier in the study by [6] to evaluate the performance of various time series distances due to its competent performance and simplicity. The choice of distance measures is critical to the performance of 1NN classifiers. The Euclidean distance is shown surprisingly competitive in terms of accuracy, compared with other more complex similarity measures [11,16]. [18] showed that, on small data sets, elastic measures such as dynamic time warping (DTW) can be more accurate than the Euclidean distance. However, the empirical results [18] strongly suggest that on large data sets, the accuracy of elastic measures converges with Euclidean distance. In this paper, we focus on extending the 1NN classifier with the Euclidean distance to achieve early classification. However, our principle can be applied to other instance-based methods using different distance metrics.

This paper is a substantial extension of [20] on some important aspects. First, we propose a new method *relaxed ECTS*, which is as accurate as ECTS on all the experiments, and can find significantly shorter MPLs than ECTS on some data sets. Second, we present new techniques to speed up the learning process. Third, we provide more experimental results including those on the new methods. Last, we provide proofs for all theoretical results.

Recently, we [21] developed a feature extraction method for early classification of time series, which can be applied on even the cases where the 1NN methods are not effective.

4 1NN early classification

1NN classifier is a lazy learning method and does not require any training. To extend 1NN classifier to achieve early classification, we need to add a training process to learn more information from the training data. In the training stage, we want to find out how early a time series can be used as the nearest neighbor to make accurate prediction.

Example 1 Consider a training set of time series T in Table 2, where each time series, written as a sequence of values in timestamp ascending order, has a length $L = 3$.

To classify a time series $s_1 = (1, 2, 1)$, using full-length time series, the 1NN classifier finds t_1 as the 1NN of s_1 in space $\mathcal{R}^3$ and outputs c_1, the class label of t_1.

Table 2 A training set T as the running example

t-id	Time series	Class
t_1	(1, 1, 1)	c_1
t_2	(2, 1, 2)	c_1
t_3	(5, 2, 1)	c_1
t_4	(6, 1, 2)	c_1
t_5	(5, 8, 7)	c_2
t_6	(5, 9, 9)	c_2
t_7	(6, 8, 9)	c_2

The prediction on s_1 can be made much earlier, since t_1 is also the 1NN of prefixes $s_1[1] = (1)$ and $s_1[1, 2] = (1, 2)$. In other words, using the 1NN of the prefixes, we may make early prediction. □

For a time series s and a training data set T, let

$$NN^l(s) = \arg\min_{t \in T}\{dist(s[1, l], t[1, l])\}$$

be the set of the nearest neighbors of s in T in prefix space $\mathcal{R}^l$. In Example 1, $NN^1(s_1) = \{t_1\}$ and $NN^2(s_1) = \{t_1\}$.

In the full-length space $\mathcal{R}^L$, using the 1NN classifier, a time series s is classified by the dominating label in $NN^L(s)$. Consider prefix space $\mathcal{R}^{L-1}$. Let $T[1, L-1] = \{t[1, L-1] | t \in T\}$ be the set of length-$(L-1)$ prefixes of all time series in the training set T. If the time series in $NN^L(s)$ are still the nearest neighbors of $s[1, L-1]$ in $T[1, L-1]$, i.e., $NN^L(s) = NN^{L-1}(s)$, then we can use $NN^{L-1}(s)$ to make prediction on the class label of s at timestamp $(L-1)$ without compromise in accuracy. This immediately leads to early prediction, as observed in Example 1. The question is when the data points of s arriving in time ascending order, how we can know starting from which prefix length l, the nearest neighbors of s will remain the same.

Generally, the set of length-l prefixes $(1 \leq l \leq L)$ of the time series in T is $T[1, l] = \{t[1, l] | t \in T\}$. For $t \in T$, the *set of reverse nearest neighbors in prefix space* $\mathcal{R}^l$ of $t(1, l)$ is $RNN^l(t) = \{t' \in T \mid t \in NN^l(t')\}$. That is, $RNN^l(t)$ is the set of time series in T that treat t as its nearest neighbor. In Example 1, since $RNN^1(t_1) = RNN^2(t_1) = RNN^3(t_1) = \{t_2\}$, $RNN^1(t_2) = RNN^2(t_2) = RNN^3(t_2) = \{t_1\}$.

Suppose training set T is a sufficiently large and uniform sample of the time series to be classified. For a time series $t \in T$, if there exists a timestamp $l < L$ such that $RNN^l(t) = RNN^{l+1}(t) = \cdots = RNN^L(t)$, then we can use t to make prediction early at timestamp l without loss in expected accuracy, since every time series s which uses t in $\mathcal{R}^L$ to predict the class label also likely has t as the 1NN in prefix spaces $\mathcal{R}^l, \mathcal{R}^{l+1}, \ldots, \mathcal{R}^{L-1}$. In Example 1, t_1 can be used to make prediction at timestamp 1 since $RNN^1(t_1) = RNN^2(t_1) = RNN^3(t_1)$.

Definition 1 (*minimum prediction length*) In a training data set T with full-length L, for a time series $t \in T$, the *minimum prediction length* (*MPL* for short) of t, $MPL(t) = k$ if for any $l (k \leq l \leq L)$, (1) $RNN^l(t) = RNN^L(t) \neq \emptyset$ and (2) $RNN^{k-1}(t) \neq RNN^L(t)$. Specifically, if $RNN^L(t) = \emptyset$, then $MPL(t) = L$. We denote by $MPP(t) = t[1, MPL(t)]$ the *minimum prediction prefix* of t. □

Example 2 In Example 1, $MPL(t_1) = 1$ and $MPP(t_1)$=(1). Moreover, $MPL(t_2)$=1, $MPL(t_3) = MPL(t_4) = 2$. $MPL(t_5) = MPL(t_6) = MPL(t_7) = 3$. □

Given a training set T, a simple *1NN early classification method* works as follows.

Training Phase For each time series $t \in T$, we calculate the minimum prediction length $MPL(t)$.
Classification Phase For a time series s to be classified, the values of s arrive in timestamp ascending order. At timestamp i, we return the dominating class label of time series in $NN^i(s)$ that have an MPL at most i. If no such a time series is found, then we cannot make reliable prediction at the current timestamp and have to wait for more values of s.

Example 3 In Example 1, it is easy to verify that $s_1 = (1, 2, 1)$ is classified as c_1 using t_1 at timestamp 1. Consider $s_2 = (6, 2, 3)$. $NN^1(s_2) = \{t_4, t_7\}$, but the MPLs of t_4 and t_7 are greater than 1. Thus, s_2 cannot be classified at timestamp 1. $NN^2(s_2) = \{t_3, t_4\}$. The MPLs of t_3 and t_4 are 2. Thus, s_2 can be assigned label c_1 at timestamp 2. □

The intuition behind the simple RNN method can be understood from a classification model compression angle. A time series t in the training set T can be compressed as its minimum prediction prefix $MPP(t)$ since based on the information in T, any longer prefix of t does not give a different prediction on a time series that uses t as the nearest neighbor. Thus, those longer prefixes of t are redundant in 1NN classification and can be removed to achieve early prediction. We assume that the training set is a sufficiently large and uniform sample of the data to be classified. Then, for new time series to be classified, the early 1NN classifier can classify new time series as accurate as full-length 1NN.

The 1NN early classification method has two drawbacks. First, to make early prediction using a time series t in a training set T, the RNNs of t must be stable after timestamp $MPL(t)$. This requirement is often too restrictive and limits the capability of finding shorter prediction length. In Example 1, the RNNs of t_5, t_6, t_7 are not stable, and the MPLs of them are all 3. We cannot use those time series to classify $s_3 = (5, 8, 8)$ at timestamp 2.

Second, the 1NN early classification method may overfit a training set. The MPP of a time series is obtained by only considering the stability of its RNN, which often consist of a small number of time series. The learned $MPL(t)$ may not be long and robust enough to make accurate classification if the training set is not big enough or is not a uniform sample of the time series to be classified.

5 The ECTS method

To overcome the drawbacks in the early 1NN classification method, we develop the ECTS method that extends the early 1NN classification method by finding the MPL for a cluster of similar time series instead of a single time series. In this section, we first develop the idea of the ECTS classifier. Then, we present the algorithm of ECTS method in detail. At last, we discuss a variation of computing MPLs in the framework of ECTS.

5.1 ECTS

When we cluster the training data in each prefix space, $\mathcal{R}^l$, where $l = 1, 2, \ldots, L$, we can observe the formations of clusters along the time. When l is small, the clusters in $\mathcal{R}^l$ may be very different from the ones in $\mathcal{R}^L$. When l increases and approaches L, some clusters in $\mathcal{R}^l$ may become stable and similar to some in $\mathcal{R}^L$. Those clusters can be used for early prediction.

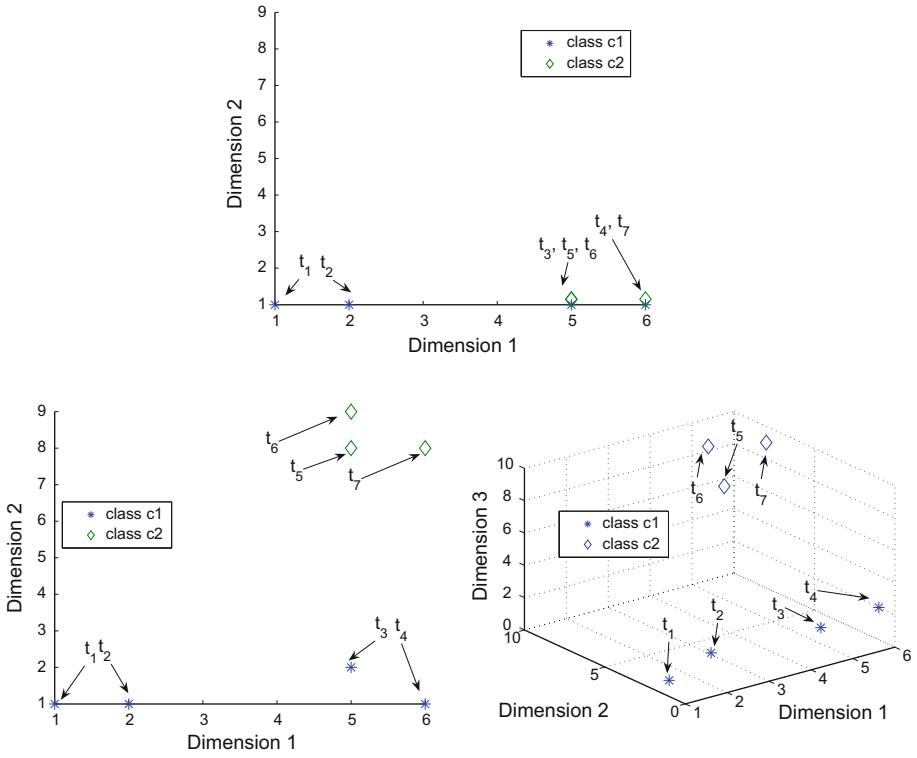


Fig. 1 Plots of data in Table 2. We can observe the natural clusters in each prefix space, which will be summarized in Table 3

Table 3 The clusters in different prefix spaces, as shown in Fig. 1

Prefix space	Clusters
$\mathcal{R}^1$	$S_1 = \{t_1, t_2\}, S_2 = \{t_3, t_4, t_5, t_6, t_7\}$
$\mathcal{R}^2$	$S_1 = \{t_1, t_2\}, S_2 = \{t_3, t_4\} S_3 = \{t_5, t_6, t_7\}$
$\mathcal{R}^3$	$S_1 = \{t_1, t_2\}, S_2 = \{t_3, t_4\} S_3 = \{t_5, t_6, t_7\}$

Example 4 Figure 1 shows the distribution of the time series in Table 2 in each prefix space, $\mathcal{R}^3$, $\mathcal{R}^2$ and $\mathcal{R}^1$. From Fig. 1, we can observe the natural clusters in each prefix space. We summarize the clusters in Table 3. Cluster S_1 is stable in all three spaces and thus can be used at early classification as early as timestamp 1. Clusters S_2 and S_3 are stable at timestamps 2 and 3 and thus can be used as early as at timestamp 2. □

We consider the clusters in the full-length space $\mathcal{R}^L$ as the ground truth. To learn when a cluster can be used for early classification, we need to address two issues. First, we need to use a clustering approach to obtain the clusters in the full-length space. Second, we need to compute the MPLs of clusters.

We adopt single link MLHC [4], an agglomerative hierarchical clustering method to cluster the training data set in the full-length space. It builds a hierarchical clustering tree level by level by merging all mutual nearest neighbor pairs of clusters at each level. Two clusters form a mutual nearest neighbor pair if they consider each other as the nearest neighbor.

The distance between two clusters is measured by the minimum among all inter-cluster pair-wise distances. In the output hierarchical clustering tree (i.e., a dendrogram), a cluster represented by a leaf node is called a *leaf cluster*, and a cluster represented by an internal node is called a *subcluster*. The whole training set is represented by the root and thus is called the *root cluster*.

A cluster S is called *1NN consistent* [5] if for each object (a time series in our case) $o \in S$, the 1NN of o also belongs to S. Immediately, we have the following.

Lemma 1 *The subclusters and the root cluster generated by single link MLHC are 1NN consistent.*

Proof In the dendrogram generated by MLHC, we call a cluster represented by a leaf node a leaf cluster, a cluster represented by an internal node a subcluster, and the cluster represented by the root the root cluster.

A leaf cluster contains only one time series. For a leaf cluster s in the dendrogram generated by MLHC on space R^l, let $Sib(s)$ be the sibling cluster of s, which is another time series. Now, we show $NN^l(s) \cap Sib(s) \neq \emptyset$ by contradiction.

Suppose $NN^l(s) \cap Sib(s) = \emptyset$. Then, we can find $q \in NN^l(s)$. According to the assumption, $q \notin Sib(s)$. Then, for all $s' \in Sib(s)$, we have $dist(s, s') > dist(s, q)$, which means $dist(s, Sib(s)) > dist(s, q)$. Then, s cannot be merged with $Sib(s)$ in the process of MLHC. This contradicts with the assumption that $Sib(s)$ is the sibling of s.

For any subcluster S in the dendrogram and all $s \in S$, we have $Sib(s) \subseteq S$. Because $NN^l(s) \cap Sib(s) \neq \emptyset$, we have $NN^l(s) \cap S \neq \emptyset$. So, S is a 1NN consistent cluster. The lemma holds for the root cluster trivially since the root cluster contains all time series in question. □

If all time series in a subcluster S carry the same label, then S is called a *discriminative cluster*. We denote by $S.c$ the common class label of the time series in S. A discriminative cluster can be used in classification.

Example 5 S_1, S_2, and S_3 in space $\mathcal{R}^3$ in Table 3 are discriminative clusters and can be used in classification. For example, $s_3 = (5, 8, 8)$ finds $t_5 \in S_3$ as the 1NN in $\mathcal{R}^3$. $S_3.c = c_2$ can be assigned to s_3. □

To explore the potential of a discriminative cluster S in early classification, we find the earliest prefix space in which S is formed and becomes stable since then. The corresponding prefix length is the minimum prediction length (MPL) of S.

One straight forward way is to apply MLHC on space $\mathcal{R}^{L-1}$ and check whether S is preserved as a subcluster and continue this process on the previous prefix spaces until the subcluster is not preserved. This can be very costly in computation.

An efficient way is to check whether the 1NN consistence property holds for the cluster in the previous prefix spaces and the stability of the reverse neighbors of S.

For a subcluster S, in space $\mathcal{R}^l (1 \leq l \leq L)$, we define the *reverse nearest neighbors* of S as $RNN^l(S) = \cup_{s \in S} RNN^l(s) \setminus S$. If S is well separated from other clusters, then $RNN^l(S)$ is empty. Often, some subclusters in a training set may not be well separated from others. In such a case, $RNN^l(S)$ is the boundary area of S.

Definition 2 (*MPLs of clusters*) In a training data set T with full-length L, for a discriminative cluster S, $MPL(S) = k$ if for any $l \geq k$, (1) $RNN^l(S) = RNN^L(S)$; (2) S is 1NN consistent in space $\mathcal{R}^l$, and (3) for $l = k - 1$, properties (1) and (2) cannot be both satisfied. □

Example 6 For discriminative clusters S_1, S_2 and S_3, $MPL(S_1) = 1$, $MPL(S_2) = 2$ and $MPL(S_3) = 2$. Take S_3 as an example, in spaces $\mathcal{R}^3$ and $\mathcal{R}^2$, S_3 is 1NN consistent and $RNN^3(S_3) = RNN^2(S_3) = \emptyset$. In space $\mathcal{R}^1$, S_3 is not 1NN consistent. Thus, $MPL(S_3) = 2$. □

In a discriminative cluster S, there may be another discriminative cluster $S' \subset S$. $MPL(S)$ may be longer or shorter than $MPL(S')$. Moreover, for a time series $t \in T$, t itself is a leaf cluster with $MPL(t)$ (Definition 1). Among all the discriminative or leaf clusters containing t, we can use the shortest MPL to achieve the earliest prediction using t.

As one drawback of the 1NN early classification method, the MPL obtained from a very small cluster may cause overfitting. To avoid overfitting, a user can specify a parameter *minimum support* to control the size of a cluster. We calculate the support of a cluster in a way that is aware of the population of the corresponding class.

Definition 3 (*Support*) In a training set T, let $T_c = \{s \in T | s.c = c\}$ be the set of time series that have label c. For a discriminative cluster or a leaf cluster S such that $S.c = c$, $Support(S) = \frac{|S|}{|T_c|}$. □

We only use clusters passing the user-specified minimum support threshold for early prediction. As in many other data mining problems such as association rule mining, the support here is a parameter. The optimal value of this parameter depends on data sets.

To summarize, given a training set T, the ECTS method works in two phases as follow.

Training phase Given a *minimum support* threshold p_0, for a time series $t \in T$, let $SS = \{S | t \in S \wedge \text{S is a discriminative cluster or a leaf cluster}\}$ be the set of discriminative or leaf clusters containing t.

$$MPL(t) = \min_{S \in SS, Support(S) \geq p_0} MPL(S).$$

The training process is to compute MPLs for all $t \in T$.

Classification phase Same as the 1NN early classification method in Sect. 4.

Section 2 states that we want to build a classifier as accurate as the 1NN classifier using the full-length time series. The following result answers the requirement.

Theorem 1 *In a training data set T of full-length L, assuming for any $t \in T$ and $1 \leq l \leq L$, $NN^l(t)$ contains only one time series*[1]*, ECTS has the same accuracy in leave-one-out testing on T as the 1NN classifier using the full-length time series. Moreover, ECTS is a serial classifier, defined in Sect. 2, on the time series in T.*

Proof In the leave-one-out classification, if a sequence $t \in T$ is classified by sequence q on prefix $t(1, k)$ by the ECTS classifier, then $MPL(q) \leq k$. Since t is classified by q, $t \in RNN^k(q)$ in prefix space $\mathcal{R}^k$. From the learning process, we know that there is a cluster Q containing q, and $MPL(Q) = MPL(q)$. Since $t \in RNN^k(q)$, t is either a member of cluster Q or a member of $RNN^k(Q)$. We consider the two cases one by one.

If $t \in Q$, because $MPL(Q) \leq k$, starting from length k to the full-length L, Q is always 1NN consistent. Since $t \in Q$, in any prefix space R^l, where $k \leq l \leq L$, we have $NN^l(t) \in Q$. Because all members in Q have the same class label, by the ECTS classifier, we have $C(t, k) = C(t, k + 1) = \cdots = C(t, L)$. For t, the classifier is serial. Since $C(t, L)$

[1] If multiple 1NNs exist, then we can select the 1NN of the smallest index.

Early Classification on Time Series

Input: a training data set T;
Output: MPL(t) for each $t \in T$;
Method:
1: pre-compute 1NNs for each $t \in T$ in all prefix spaces;
2: compute the MPLs of leaf clusters and update the MPL for each $t \in T$;
3: $n = |T|$, the number of time series in T;
4: **while** $n > 1$
5: compute the mutual nearest neighbor pairs;
6: **for** each mutual nearest neighbor pair (S_1, S_2)
7: merge S_1 and S_2 into a parent cluster S, $n = n - 1$;
8: **if** all time series in S carry the same label **then**
9: compute the MPL of S;
10: update the MPL for each time series in S;
end if
end for
11: **if** no new discriminative clusters are generated in this round **then break**;
end while

Fig. 2 The algorithm of the training phase in ECTS

by ECTS is actually the 1NN classifier using the full length, so for t, ECTS gives the same prediction as the 1NN classifier using the full-length time series.

Let $t \in RNN^k(Q)$. Since $RNN^k(Q) = RNN^{k+1}(Q) = \cdots = RNN^L(Q)$, we have $t \in RNN^l(Q)$ for any $k \leq l \leq L$. It means $NN^l(t) \in Q$ for any $k \leq l \leq L$. Because all members in Q have the same class label, by the ECTS classifier, $C(t, k) = C(t, k + 1) = \cdots = C(t, L)$. For t, the classifier is serial and ECTS gives the same prediction as the 1NN classifier using the full-length time series.

Based on the above analysis, for any $t \in T$, the class label predicted by ECTS will be the same as the class label predicted by the 1NN classifier using the full-length time series, and the ECTS is serial. □

Theorem 1 indicates that the learned MPLs by ECTS are long enough to classify all time series in the training data set as accurately as the full-length time series by a 1NN classifier. If the training data set is large enough and provides a good sample of the time series space, then, for an unlabeled time series, the ECTS classifier is expected to give the same accuracy as the full-length 1NN classifier' and is expected to be serial.

5.2 The algorithm

To train an ECTS classifier, we integrate the computation of MPLs for clusters into the framework of MLHC. The algorithm of training the ECTS classifier is shown in Fig. 2.

The training process requires many nearest neighbor queries and reverse nearest neighbor queries in various prefix spaces. To share the computation, we precompute the nearest neighbors for every time series $t \in T$ in all prefix spaces, from $\mathcal{R}^1$ to $\mathcal{R}^L$, (line 1 in Fig. 2). This precomputation step takes time $O(|T|^2 \cdot L)$ where L is the full length.

Then, we apply the single link MLHC [4] to space $\mathcal{R}^L$. We implement the MLHC framework in lines 4-11. As stated before, in each iteration, MLHC merges all mutual nearest neighbor (MNN) pairs of clusters. When a new cluster S is formed in the process of MLHC, we compute its MPL (line 9). In line 10, we update the MPL of every time series in $s \in S$ if

$MPL(S) < MPL(s)$. When the iteration terminates, the MPL of each time series s is the shortest MPL of all the clusters containing s.

All time series in a discriminative cluster should have the same label. In MLHC, if a cluster formed is not pure, that is, the time series in the cluster are inconsistent in class label, then we do not need to merge the cluster further to other clusters in the next iteration. MLHC in our method terminates when no pure clusters are generated after the current round of iteration. In other words, our adaption of MLHC generates a cluster forest instead of a cluster tree.

To compute the MPL of a discriminative cluster S, according to Definition 2, we check whether the 1NN consistency and the RNN stability hold in space $\mathcal{R}^L$ and the prefix spaces $\mathcal{R}^l$. The checking is conducted in the prefix length descending order. If $MPL(S) = k$, to compute the MPL of S, we need to find $RNN^l(S)$ and check 1NN consistency in spaces $\mathcal{R}^l(L \geq l \geq (k-1))$. If the MPL of every discriminative cluster is computed from the scratch, then it is very costly. Fortunately, we do not need to compute the MPL for every discriminative cluster from scratch. Instead, we can use the information provided by the clusters formed in the previous iterations. When computing the MPL of a discriminative cluster S, which is formed by merging clusters $S1$ and $S2$ in an iteration, two situations may arise.

First, $|S1| > 1$ and $|S2| > 1$. Without loss of generality, let us assume $MPL(S1) \leq MPL(S2)$. Cluster S must be 1NN consistent and RNN stable in spaces $\mathcal{R}^l$ for $L \geq l \geq MPL(S2)$. Thus, we only need to check whether the 1NN consistency and RNN stability hold for S in space $\mathcal{R}^l$ for $MPL(S2) \geq l \geq 1$.

Second, if $|S1| = 1$ and $|S2| > 1$, then S is always 1NN consistent in spaces $\mathcal{R}^l$ for $L \geq l \geq MPL(S2)$. For ECTS, to test the RNN stability of S, we only need to check whether $RNN(S1) \setminus S2$ is stable in spaces $\mathcal{R}^l$ for $L \geq l \geq MPL(S2)$. This is because we already know that the RNNs of cluster $S2$ are stable in spaces $\mathcal{R}^l$ for $L \geq l \geq MPL(S2)$. If for some $l(L \geq l \geq MPL(S2))$ that S passes the test in prefix space $\mathcal{R}^l$ but fails in $\mathcal{R}^{l-1}$, then $MPL(S) = l$. If S passes the test, then $MPL(S) \leq MPL(S2)$. We need to check the 1NN consistency and the RNN stability in the prefix spaces of shorter prefix lengths.

The complexity of building the dendrogram in the full-length space by MLHC is $O(|T|^2)$. By integrating the MPLs computation in MLHC, the complexity of is $O(|T|^3 L)$.

5.3 Relaxed ECTS

If we apply the 1NN classifier in the full-length space $\mathcal{R}^L$ to classify the training data T using leave-one-out method, then we may get some time series miss classified. We denote $T_{mis} \subseteq T$ as the set of time series that are not classified correctly in the leave-one-out process. One important reason of misclassification is that some time series in T_{mis} are on the decision boundary. The nearest neighbors of those time series may have a considerable chance to fall into other clusters in the prefix spaces. To compute the MPL for a cluster, Definition 2 requires the RNN stability from prefix space $\mathcal{R}^{MPL(S)}$ till the full-length space. The time series in T_{mis} may hinder many discriminative clusters from meeting the stability requirement. Can we relax the requirement of MPL by ignoring the instability caused by the time series in T_{mis}?

Definition 4 (*Relaxed MPL*) In a training data set T with full-length L, for a discriminative cluster S, $MPL(S) = k$ if for any $l \geq k$, (1) $RNN^l(S) \cap (T - T_{mis}) = RNN^L(S) \cap (T - T_{mis})$; (2) S is 1NN consistent in space $\mathcal{R}^l$, and (3) for $l = k - 1$, (1) and (2) cannot be both satisfied.

Specifically, if S is a leaf cluster and $RNN^L(S) \cap (T - T_{mis}) = \emptyset$, then we define $MPL(S) = L$. □

We call the ECTS classifier using the above relaxed MPL the relaxed version of ECTS. In the relaxed version, we relax the condition of RNN stability to partial stability. The relaxed ECTS has the following property.

Theorem 2 *In a training data set T of full-length L, assuming for any $t \in T$ and $1 \leq l \leq L$, $NN^l(t)$ contains only one time series,*[2] *the relaxed ECTS has the same leave-one-out accuracy on T as the 1NN classifier using the full-length time series.*

Proof In the leave-one-out classification process, if a time series t is classified by sequence s on prefix $t(1, k)$ by the relaxed ECTS classifier, then t either is a member of a cluster Q or belongs to $RNN^k(Q)$.

In the case where $t \in Q$, same as the proof in Theorem 1, we can prove that the relaxed ECTS makes prediction as accurate as the 1NN classifier using the full-length time series.

In the case that $t \in RNN^k(Q)$, we need to consider two subcases.

In the first subcase, $t \in RNN^k(Q) \cap (T - T_{mis})$, same as the proof in Theorem 1, we can prove the that relaxed ECTS makes prediction as accurate as 1NN classifier using the full-length time series.

In the second subcase, $t \in RNN^k(Q) \cap T_{mis}$. Although $C(t, k) = C(s, L)$ cannot be guaranteed, since t cannot be correctly classified by the 1NN classifier using the full-length time series, the subcase does not make the accuracy of the relaxed ECTS lower than that of the 1NN classifier using the full-length time series.

Based on the above analysis, we conclude, by using the relaxed ECTS, the leave-one-out accuracy is at least the same as using the 1NN classifier using the full-length time series.

□

Comparing with ECTS, the average MPLs learned by the relaxed ECTS is expected to be shorter because for a cluster, we only require a subset of its RNN to be stable. When classifying unlabeled time series, the relaxed ECTS is expected to give the same prediction as ECTS and make the prediction on a shorter prefix. We will verify the expected property in our experiments.

6 Experimental results

The UCR time series archive [12] provides 23 time series data sets, which are widely used to evaluate time series clustering and classification algorithms. In each data set, the time series have a fixed length. Each data set contains a training set and a testing set. The 1NN classification accuracies using the Euclidean distance on the testing sets are provided in the archive as well.

Table 4 lists the results on all the 7 data sets in the archive where the full-length 1NN classifier using Euclidean distance achieves an accuracy of at least 85%. The 1NN classifier can be regarded effective on those data sets. The seven data sets cover cases of 2-class and more-than-two-class.

Table 4 compares 6 methods. We use the 1NN classifier using the full length (denoted by full 1NN) as the baseline. In addition to ECTS and relaxed ECTS, we also report the results of the 1NN early classification method (Sect. 4, denoted by 1NN Early), the 1NN fixed method, which will be introduced in Sect. 6.2, and the SCR method [19]. ECTS and relaxed ECTS use only an optional parameter, *minimum support*, to avoid overfitting. All results of ECTS and relaxed ECTS in Table 4 are obtained by setting *minimum support* = 0.

[2] Again, if multiple 1NNs exist, then we can select the 1NN of the smallest index.

Table 4 Results on seven data sets from UCR Time Series Archive, where the full-length 1NN classifier using Euclidean distance achieves an accuracy of at least 85%

Data set		ECTS	RelaxedECTS	Early1NN	1NNFixed	Full1NN	SCR
Wafer	Accuracy	99.08%(−0.47%)	99.08%(−0.47%)	99.16%(−0.39%)	99.32% (−0.23%)	99.55%	93% (−6.58%)
2 classes	Ave. len.	67.39 (44.34%)	67.39 (44.34%)	122.05 (80.30%)	110 (72.37%)	152	55.36 (36.42%)
1000 training inst.	Train. time	152.90 sec	161.08 sec	3.22 sec	1.58 sec	0 sec	164.59 sec
6174 testing inst.	Class. time	0.053 sec	0.038 sec	0.103 sec	0.004 sec	0.004 sec	0.204 sec
Gun-point	Accuracy	86.67%(−5.1%)	86.67%(−5.1%)	87.3%(−4.41%)	91.3%(−0.03%)	91.33%	62.67% (−31.36%)
2 classes	Ave. len.	70.387 (46.92%)	70.387(46.92%)	107.24 (71.49%)	140 (92.67%)	150	116.17 (77.45%)
50 training inst.	Train. time	0.39 sec	0.406 sec	0.09 sec	<0.01 sec	0 sec	0.86 sec
150 testing inst.	Class. time	0.002 sec	0.003 sec	0.004 sec	0.002 sec	0.002 sec	0.11 sec
Two patterns	Accuracy	86.48%(−4.97%)	86.35%(−5.11%)	87.25%(−4.12%)	90.72%(−0.8%)	91%	94.75% (+4.12%)
4 classes	Ave. len.	111.097 (86.79%)	110.743 (86.52%)	113.24 (88.5%)	124 (96.88%)	128	86.13(67.29%)
1000 training inst.	Train. time	48.76 sec	47.46 sec	2.18 sec	1.34 sec	0 sec	65.23 sec
4000 testing inst.	Class. time	0.101 sec	0.07 sec	0.077 sec	0.003 sec	0.003 sec	0.125 sec
ECG	Accuracy	89% (+1.17%)	0.89% (+1.17%)	89% (+1.17%)	89% (+1.17%)	88%	73% (−17.05%)
2 classes	Ave. len.	74.04 (77.13%)	57.71 (60.11%)	83.12 (86.58%)	92 (92.71%)	96	37.5 (39.06%)
100 training inst.	Train. time	0.83 sec	1.20 sec	0.1 sec	0.02 sec	0 sec	4.22 sec
100 testing inst.	Class. time	0.004 sec	0.004 sec	0.004 sec	0.001 sec	0.001 sec	0.062 sec
Syn. control	Accuracy	89% (+1.17%)	88.3% (+0.34%)	88.33% (+0.38%)	88% (+0%)	88%	58.33% (−33.72%)
6 classes	Ave. len.	53.98 (89.97%)	52.73 (87.9%)	55.09 (91.82%)	60 (100%)	60	29.39 (50%)
300 training inst.	Train. time	4.64 sec	4.992 sec	0.12 sec	0.06 sec	0 sec	21.09 sec
300 testing inst.	Class. time	0.004 sec	0.006 sec	0.004 sec	0.001 sec	0.001 sec	0.02 sec

Table 4 continued

Data set		ECTS	RelaxedECTS	Early1NN	1NNFixed	Full1NN	SCR
OliveOil	Accuracy	90% (+3.8%)	90% (+3.8%)	90% (+3.8%)	83.33% (−3.92%)	86.7%	36.7% (−57.68%)
4 classes	Ave. len.	497.83 (82%)	497.83 (82%)	526 (92.28%)	406 (71.23%)	570	500 (87.72%)
30 training inst.	Train. time	0.22 sec	0.28 sec	0.08 sec	0.02 sec	0 sec	2.03 sec
30 testing inst.	Class. time	0.058 sec	0.0378 sec	0.043 sec	0.006 sec	0.006 sec	0.016 sec
CBF	Accuracy	85.2% (+0%)	85.2% (+0%)	86.89% (+1.98%)	83.2%(−2.35%)	85.2%	55.22% (−35.18%)
3 classes	Ave. len.	91.73 (71.5%)	91.52 (71.50%)	103.20 (80.63%)	54 (42.19%)	128	46 (35.93%)
30 training inst.	Train. time	0.09 sec	0.109 sec	0.02 sec	<0.01 sec	0 sec	0.24 sec
900 testing inst.	Class. time	0.001 sec	0.001 sec	0.001 sec	0.001 sec	0.001 sec	0.015 sec

The data sets cover cases of two-class and more-than-two-class. Six methods are compared. We use the 1NN classifier using the full length (denoted by full 1NN) as the baseline. We also report the results of the 1NN early classification method (Sect. 4, denoted by 1NN Early), the 1NN fixed method, which will be introduced in Sect. 6.2, and the SCR method [19]. ECTS and relaxed ECTS use only an optional parameter, *minimum support*, to avoid overfitting. All results of ECTS and relaxed ECTS are obtained by setting *minimum support* = *0*

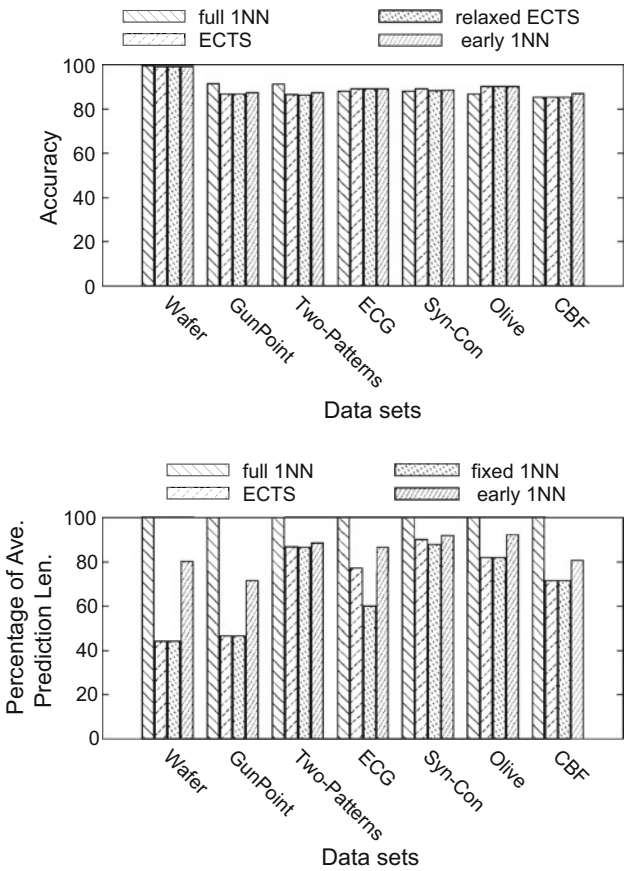


Fig. 3 Comparison between full 1NN, ECTS, relaxed ECTS, and early 1NN in terms of classification accuracy and average prediction length ($Cost(C, T')$ defined in Sect. 2. ECTS can preserve accuracy consistently as the baseline method. At the same time, ECTS can achieve considerable saving in prediction length

All the experiments were conducted using a PC computer with an AMD 2.2 GHz CPU and 1GB main memory. The algorithms were implemented in C++ using Microsoft Visual Studio 2005.

6.1 Results of ECTS methods

In Fig. 3, we compare the performance of full 1NN, ECTS, relaxed ECTS, and early 1NN in terms of classification accuracy and average prediction length ($Cost(C, T')$ defined in Sect. 2.

On data sets Wafer and Gun-point, the average prediction lengths of ECTS are shorter than half of the full lengths. On the other five data sets, the average prediction lengths of ECTS are from 71% to 89% of the full lengths. In terms of accuracy, except for data sets Gun-point and Two-patterns, ECTS has an accuracy almost the same or even slightly better than that obtained by the full-length 1NN method. On Gun-point and Two-patterns, ECTS is about 5% lower in accuracy. The results on the seven data sets show that ECTS can preserve accuracy

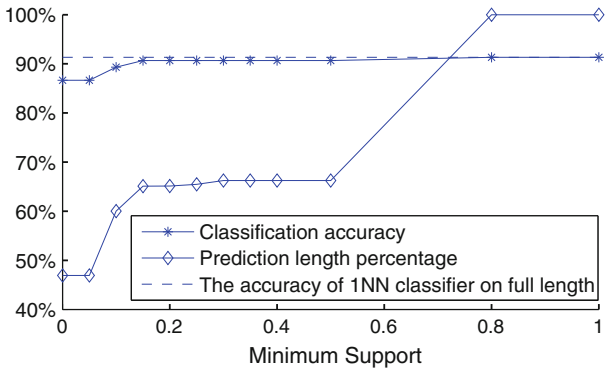


Fig. 4 Accuracy and average length vs. minimum support on the Gun-point data set. When *minimum support* increases, the accuracy of ECTS approaches the accuracy of the full-length 1NN classifier quickly. As the trade-off, the average prediction length increases, too. By using parameter *minimum support*, ECTS can reduce overfitting effectively

consistently as the baseline method. At the same time, ECTS can achieve considerable saving in prediction length.

Similar to ECTS, the results of relaxed ECTS confirm that relaxed ECTS can also achieve early classification while retaining the accuracy as the baseline method. Especially, on data set ECG, relaxed ECTS uses an average prediction length of 57.71,(60.11% of the full length) and ECTS uses an average prediction length of 74.04 (77.13% of the full length). Relaxed ECTS achieves significantly earlier classification than ECTS. Also, on the Synthetic Control data set, relaxed ECTS uses shorter prediction than ECTS. The results are consistent with our discussion in Section 4.2. The relaxed ECTS may be able to use a shorter prefix length to obtain nearly the same accuracy as ECST. In Table 4, the runtime of the training algorithms in ECTS and the relaxed ECTS (i.e., computing the MPLs for all training time series) and the average runtime of the classification phase of the ECTS methods are also reported. The relaxed ECTS takes slightly longer time in training. Both methods are very fast in classification.

From Fig. 3, we can also see that on all the 7 data sets, the early 1NN method achieves almost the same accuracies as the baseline method. In terms of accuracy, the early 1NN method, ECTS, and relaxed ECTS are all quite reliable. ECTS and relaxed ECTS always achieve a shorter average prediction length than the early 1NN method. This result confirms that finding MPLs through clusters helps to obtain shorter MPLs.

ECTS and relaxed ECTS have an optional parameter, *minimum support*. If the user does not set the parameter, then the default value is 0. As shown in the experiments, in most cases, the performance of ECTS is satisfactory by using the default value. On data sets Gun-point and Two-patterns, ECTS and relaxed ECTS are about 5% lower in accuracy comparing with the full-length 1NN. As explained before, when *minimal support* = 0, ECTS may over fit a training set and thus may slightly lose accuracy. By increasing *minimum support*, overfitting can be reduced. Figure 4 shows the effect on the Gun-point data set. When *minimum support* increases, the accuracy of ECTS approaches the accuracy of the full-length 1NN classifier quickly. As the trade-off, the average prediction length increases, too.

By comparing ECTS, relaxed ECTS, and the early 1NN method against the baseline method, we can conclude that ECTS, relaxed ECTS and the early 1NN method have a common property that they can all make early classification while retain an accuracy comparable

with using full-length 1NN classifier. Relaxed ECTS is the best at finding short prediction length. ECTS has the similar performance as relaxed ECTS in most cases. Early 1NN cannot achieve early classification as well as ECTS and relaxed ECTS, but it requires significantly less training time.

6.2 Comparison with 1NN fixed

Is learning different MPLs for different time series necessary? Can we just learn a fixed MPL for all time series in a training set? Let us consider the following simple early classification method called *1NN fixed*. Given a training set T of full-length L, we calculate the 1NN classification accuracy p in the full space $\mathcal{R}^L$. Then, we check the prefix spaces $\mathcal{R}^{L-1}, \mathcal{R}^{L-2}, \ldots$ until prefix space $\mathcal{R}^k$ such that the accuracies in spaces $\mathcal{R}^{L-1}, \ldots, \mathcal{R}^{k+1}$ are at least p, and the accuracy in space $\mathcal{R}^k$ is lower than p. We use $(k+1)$ as the MPL for all time series. That is, in classification, we always read the length-$(k+1)$ prefix of a time series s to be classified and find the 1NNs of s among the length-$(k+1)$ prefixes of the time series in T to classify s.

Interestingly, 1NN fixed is a simplified special case of ECTS in two-class situations under the assumption that each class forms one discriminative cluster in the full-length space $\mathcal{R}^L$. However, since 1NN fixed does not consider the different early classification capabilities of the clusters in the hierarchy, it may use longer prefixes for classifications. Furthermore, when there are multiple classes or multiple large discriminative clusters, the 1NN fixed method may not work well since it requires the overall accuracy to be high and cannot identify clusters that are separated from other clusters earlier than the overall accuracy is satisfied.

We compare 1NN fixed with ECTS in Fig. 5. On data sets Wafer, ECG and synthetic-control, ECTS using *minimum support*$=0$ achieves a shorter average prediction length than the 1NN fixed method. The two methods have similar accuracies. Especially, for the 6 classes synthetic control data set, the 1NN fixed method has to use the full length. ECTS uses a shorter prediction length. The saving mainly comes from the class cyclic, which is classified using an average length of 45. To handle multiclass situations, the 1NN fixed method cannot classify one class earlier if the other classes are mixed together in the prefix spaces.

On data set Gun-point, by setting *minimum support*$=15\%$ to reduce overfitting, ECTS obtains an accuracy comparable with the 1NN fixed method, but uses a remarkably shorter average prediction length. Similar results are observed on data set Two-patterns.

From Table 4, we can see that on data sets OliveOil and CBF, interestingly, the 1NN fixed method is less accurate than ECTS but can obtain shorter average prediction length. For example, on data set OliveOil, ECTS obtains an accuracy of 90% and 1NN fixed method makes only 83.33%. The 1NN fixed method obtains an average prediction length of 406, and ECTS gives a length of 497.63. By analyzing the training set of 30 time series, we find that, training samples 7 and 9 are the cause of the dropping accuracy in the 1NN fixed method, which means the MPLs (406) of those two samples learned by the 1NN fixed method are not long enough to make accurate classification. In ECTS, the learned MPLs vary from 117 to 570 for the training samples. For samples 7 and 9, the learned MPLs are 567 and 570, respectively. Why does ECTS learn longer MPLs for samples 7 and 9? In the full-length space, training sample 9 has an empty RNN set. The RNN set of training sample 7 consists of samples from two classes. Those RNN sets suggest that samples 7 and 9 are likely on the decision boundary. In contrast to the 1NN fixed method, ECTS can find longer MPLs for samples likely on the decision boundary to reduce the possible misclassification led by such a sample.

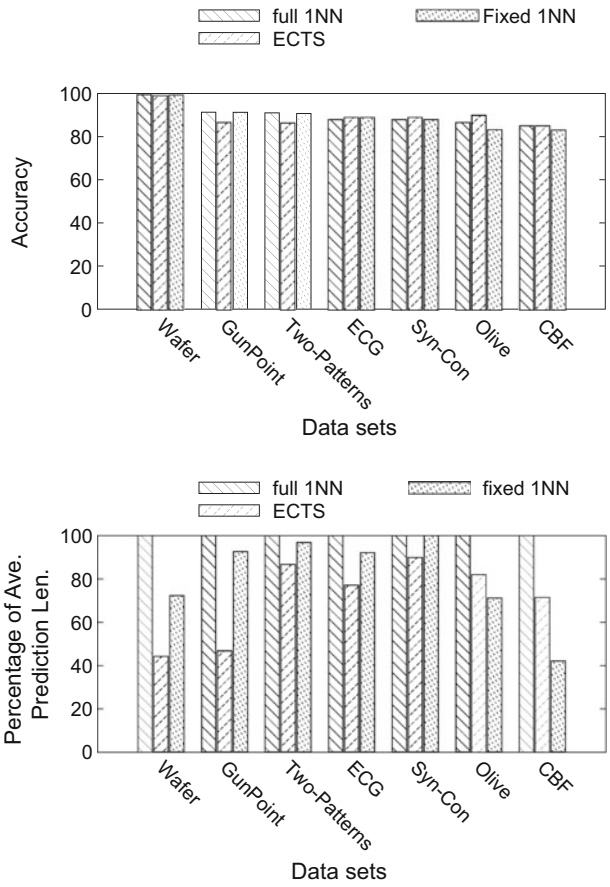


Fig. 5 Comparison between ECTS and fixed 1NN. ECTS and the 1NN fixed method are both as accurate as the full-length 1NN method, but ECTS can make prediction significantly earlier than 1NN fixed on the first 5 data sets

From the above analysis, we can conclude that in most cases, the ECTS method can use significantly shorter prediction to achieve very similar accuracy as the fixed 1NN method. ECTS is especially suitable for more-than-two-class early classification task.

6.3 Comparison with SCR

We also compare ECTS with our previous symbolic method SCR [19], a rule-based classifier proposed to solve the problem of early prediction for symbolic sequences.

Since SCR can only handle discrete values, k-means ($k = 3$) is used to discretize values in the time series into 3 values, following the recommendation in [19]. In the classification, we do the online discretization using learned thresholds on the training data set. SCR requires a parameter, expected classification accuracy, which is set to the full-length 1NN accuracy. The other parameter of SCR, *minimal support*, is set to 0.

We compare SCR with ECTS in Fig. 6. Although SCR sometimes uses a shorter average prediction length, the accuracies are significantly lower than the expected values. Comparing

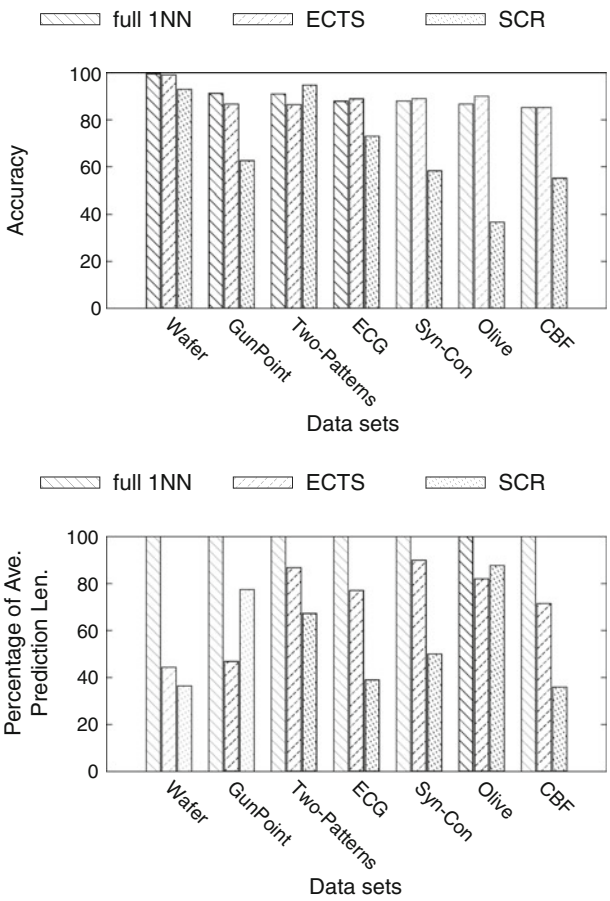


Fig. 6 Comparison between ECTS and SCR. Although SCR sometimes uses a shorter average prediction length, the accuracies are significant lower than the expected values. Comparing with SCR, ECTS makes early classification reliable in accuracy. In terms of efficiency, ECTS is much faster than SCR in training

with SCR, ECTS makes early classification reliable in accuracy. In terms of efficiency, ECTS is much faster than SCR in training.

7 Conclusions

In this paper, we propose the ECTS classifier to tackle the problem of early classification of numerical time series data. ECTS extends the 1NN classifier to achieve early classification while retains nearly the same accuracy as that of the 1NN classifier using the full-length time series. In the experiments, we show that the ECTS methods are effective and superior to other existing early classification methods.

Early classification is a new direction in data mining with many important applications. In general, early classification is concerned about the trade-off between earliness and accuracy in classification. Although we make good progress in this paper, the ETCS methods are only the first step. One important insight disclosed in this paper is that, when 1NN is effective,

it is feasible to retain the classification accuracy and shorten the prediction length substantially. This paper also leads to the fundamental question for early classification, that is, how to control the trade-off between earliness and accuracy in classification. This fundamental question is open for future study.

In the future, we also want to extend the work in a few directions. For example, we want to explore how to extend ECTS to handle multivariate time series data and complex temporal sequence data with both numerical and symbolic values. Moreover, we plan to explore how to extend the ECTS methods for streaming time series data, which may have multiple class labels.

Acknowledgments We thank the anonymous reviewers and Dr. Aristides Gionis, the handling associate editor, for their very constructive and thoughtful comments and suggestions. This research is supported in part by NSERC through an NSERC Discovery Grant and an NSERC Discovery Accelerator Supplements Grant, and by US NSF through grants IIS 0905215, DBI-0960443, OISE-0968341 and OIA-0963278. All opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

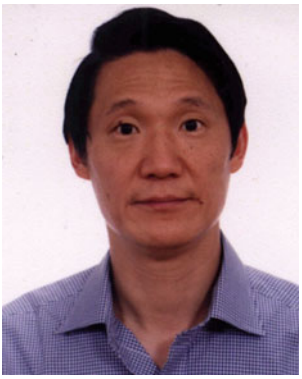
1. Bernaille L, Teixeira R, Akodkenou I, Soule A, Salamatian K (2006) Traffic classification on the fly. SIGCOMM Comput Commun Rev 36:23–26
2. Bregon A, Simon MA, Rodriguez JJ, Alonso C, Pulido B, Moro I (2006) Early fault classification in dynamic systems using case-based reasoning. In: Current topics in artificial intelligence, vol 4177 of lecture notes in computer science. Springer, Berlin, pp 211–220
3. Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inform Theor 13(1):21–27
4. Ding CHQ, He X (2005) Cluster aggregate inequality and multi-level hierarchical clustering. In: PKDD'05: proceedings of the 9th European conference on principles and practice of knowledge discovery in databases, pp 71–83
5. Ding C, He X (2004) K-nearest-neighbor consistency in data clustering: incorporating local information into global optimization. In: SAC'04: proceedings of the 2004 ACM symposium on applied computing, ACM, New York, NY, USA, pp 584–589
6. Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. Proc VLDB Endow 1(2):1542–1552
7. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139
8. Griffin MP, Moorman JR (2001) Toward the early diagnosis of neonatal sepsis and sepsis-like illness using novel heart rate analysis. Pediatrics 107(1):97–104
9. Ji X, Bailey J, Dong G (2007) Mining minimal distinguishing subsequence patterns with gap constraints. Knowl Inf Syst 11(3):259–286
10. Kadous MW, Sammut C (2005) Classification of multivariate time series and structured data using constructive induction. Mach Learn 58(2–3):179–216
11. Keogh E, Kasetty S (2002) On the need for time series data mining benchmarks: a survey and empirical demonstration. In: KDD'02: proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, NY, USA, pp 102–111
12. Keogh E, Xi X, Wei L, Ratanamahatana CA (2006) The UCR time series classification and clustering homepage: http://www.cs.ucr.edu/~eamonn/time_series_data/
13. Lesh N, Zaki MJ, Ogihara M (1999) Mining features for sequence classification. In: KDD'99: proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, NY, USA, pp 342–346
14. Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing sax: a novel symbolic representation of time series. Data Min Knowl Discov 15(2):107–144
15. Rodríguez JJ, Alonso CJ, Boström H (2001) Boosting interval based literals. Intell Data Anal 5:245–262
16. Wei L, Keogh E (2006) Semi-supervised time series classification. In: KDD'06: proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, NY, USA, pp 748–753

17. Wei L, Keogh E, Van Herle H, Mafra-Neto A, Abbott RJ (2007) Efficient query filtering for streaming time series with applications to semisupervised learning of time series classifiers. Knowl Inf Syst 11(3):313–344
18. Xi X, Keogh E, Shelton C, Wei L, Ratanamahatana CA (2006) Fast time series classification using numerosity reduction. In: ICML'06: proceedings of the 23rd international conference on Machine learning. ACM, New York, NY, USA, pp 1033–1040
19. Xing Z, Pei J, Dong G, Yu PS (2008) Mining sequence classifiers for early prediction. In: SDM'08: proceedings of the 2008 SIAM international conference on data mining, pp 644–655
20. Xing Z, Pei J, Yu PS (2009) Early prediction on time series: a nearest neighbor approach. In: 'Proceedings of the 21st international jont conference on Artifical intelligence', Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 1297–1302
21. Xing Z, Pei J, Yu PS, Wang K (2011) Extracting interpretable features for early classification on time series. In: SDM'11: proceedings of the 2011 SIAM international conference on data mining
22. Ye L, Keogh E (2009) Time series shapelets: a new primitive for data mining. In: proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'09. ACM, New York, NY, USA, pp 947–956

Author Biographies

Zhengzheng Xing received the Ph.D. degree in computer science from Simon Fraser University, Canada, the M.Sc. degree in computer science from University of Windsor, Canada, and the B.E. degree in computer engineering from Beijing Institute of Technology, China. She is currently a software development engineer in Amazon.com, USA. Her research interests include data mining, machine learning, information retrieval, and natural language processing.

Jian Pei is an Associate Professor at the School of Computing Science at Simon Fraser University, Canada. His research interests can be summarized as developing effective and efficient data analysis techniques for novel data intensive applications. He is currently interested in various techniques of data mining, Web search, information retrieval, data warehousing, online analytical processing, and database systems, as well as their applications in social networks, health informatics, business, and bioinformatics. His research has been extensively supported in part by many government funding agencies and industry partners. He has published prolifically and served regularly for the leading academic journals and conferences in his fields. He is an associate editor of ACM Transactions on Knowledge Discovery from Data (TKDD) and IEEE Transactions of Knowledge and Data Engineering (TKDE). He is a senior member of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE). He is the recipient of several prestigious awards.



Philip S. Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University. He is currently a Professor in the Department of Computer Science at the University of Illinois at Chicago and also holds the Wexler Chair in Information Technology. He spent most of his career at IBM Thomas J. Watson Research Center and was manager of the Software Tools and Techniques group. His research interests include data mining, privacy preserving data publishing, data stream, Internet applications and technologies, and database systems. Dr. Yu has published more than 620 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. Dr. Yu is a Fellow of the ACM and the IEEE. He is an associate editor of ACM Transactions on Knowledge Discovery from Data. He is on the steering committee of the IEEE Conference on Data Mining and ACM Conference on Information and Knowledge Management and was a member of the IEEE Data Engineering steering committee. He was the Editor-in-Chief of IEEE Transactions on Knowledge and Data Engineering (2001-2004). He had also served as an associate editor of ACM Transactions on the Internet Technology and Knowledge and Information Systems. He had received several IBM honors including 2 IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, 2 Research Division Awards and the 94th plateau of Invention Achievement Awards. He was an IBM Master Inventor. Dr. Yu received a Research Contributions Award from IEEE Intl. Conference on Data Mining in 2003 and also an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999.