# Two Novel Techniques to Improve MDL-Based Semi-Supervised Classification of Time Series

Vo Thanh Vinh[1(✉)] and Duong Tuan Anh[2]

[1] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam
vtvinh@it.tdt.edu.vn

[2] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam
dtanh@cse.hcmut.edu.vn

**Abstract.** Semi-supervised classification problem arises in the situation that we just have a small amount of labeled instances in the training set. One method to classify the new time series in such situation is that; firstly we need to use self-training to classify the unlabeled instances in the training set. Then, we use the output training set to classify the new time series. In this paper, we propose two novel improvements for Minimum Description Length-based semi-supervised classification of time series: an improvement technique for Minimum Description Length-based stopping criterion and a refinement step to make the classifier more accurate. Our first improvement applies the non-linear alignment between two time series when we compute Reduced Description Length of one time series exploiting the information from the other. The second improvement is a post-processing step that aims to identify the class boundary between positive and negative instances accurately. For the second improvement, we propose an algorithm called Refinement that attempts to identify the wrongly classified instances in the self-training step; then it reclassifies these instances. We compare our method with some previous methods. Experimental results show that our two improvements can construct more accurate semi-supervised time series classifiers.

**Keywords:** Time series · Semi-supervised classification · Stopping criterion · MDL principle · X-Means

## 1 Introduction

In time series data mining, classification is a crucial problem which has attracted lots of research works in the last decade. However, most of the current methods assume that the training set contains a great number of labeled data. Such an assumption is unrealistic in the real world where we have a small set of labeled data, in addition to abundant unlabeled data. In such circumstances, semi-supervised classification is a suitable paradigm.

To the best of our knowledge, most of the studies about semi-supervised classification of time series follow two directions: the first approach bases on Wei and Keogh

DOI: 10.1007/978-3-662-53580-6_8

framework [8] as in [1, 6, 8], and the second approach bases on a clustering algorithm such as in [10–12].

For the former approach, Semi-supervised classification (SSC) method will train itself by trying to expand the set of labeled data with the most similar unlabeled data until reaching a stopping criterion. Though several semi-supervised approaches have been proposed, only a few could be used for time series data, due to its special characteristic within. Most of the time series SSC methods have to suggest a good stopping criterion. The SSC approach for time series proposed by Wei et al. in 2006 [8] uses a stopping criterion which is based on the minimal nearest neighbor distance, but this criterion can not work correctly in some situations. Ratanamahatana and Wanichsan, in 2008 [6], proposed a stopping criterion for SSC of time series which is based on the historical distances between candidate instances from the set of unlabeled instances to the initial positive instances. The most well-known stopping criterion so far is the one using Minimum Description Length (MDL) proposed by Begum et al., 2013 [1]. Even though this newest state-of-the-art stopping criterion gives a breakthrough for SSC of time series, it is still not effective to be used in some situations where time series may have some distortion along the time axis and the computation of Reduced Description Length for them becomes so rigid that the stopping point for the classifier can not be found precisely.

For the latter approach, Nhut et al. in 2011 proposed a method called LCLC (Learning from Common Local Cluster) [11]. This method firstly apply K-means clustering algorithm to obtain the clusters. Then, it considers all the instances in a cluster belong to a class. According to Begum et al. [1], this method depends too much on the clustering algorithm and it wrongly classifies many instances. In order to improve LCLC, Nhut et al. in 2012 [12] proposed an extended version of LCLC called En-LCLC (Ensemble based Learning from Common Local Clusters). This method attempts to identify probability that a time series belong to a class. Since, the authors proposed a fuzzy classification algorithm called AFNN (Adaptive Fuzzy Nearest Neighbor) based on these probabilities. According to Begum et al. [1], this method needs to be set up many initial constants. Marussy and Buza in 2013 [10] proposed a semi-supervised classification method based on single-link hierarchical clustering accompanying with must-link constraint and cannot-link constraint. Different from the other methods, Marussy and Buza applied graph theory to tackle the semi-supervised classification problem. In this method, the authors showed that semi-supervised classification problem is equivalent to finding the minimal spanning tree problem in a graph. However, this method required to know all the classes before hand. For example, in binary classification, we need to classify into two classes. Marussy and Buza's method requires that there must be two types of instances labeled positive and negative as seeds at the beginning whereas the other methods only require one type of instances (positive instances only).

In this work, we propose two novel improvements for binary SSC of time series in the spirit of the first approach direction: an improvement technique for MDL-based stopping criterion and a refinement step to make the classifier more accurate. Our first improvement applies the non-linear alignment between two time series when we compute Reduced Description Length of one time series exploiting the information from the other. In order to obtain the non-linear alignment between two time series, we apply the Dynamic Time Warping distance. For the second improvement, we propose a post-processing step that aims to identify the class boundary between positive and negative instances accurately. Experimental results reveal that our two improvements can construct more accurate semi-supervised time series classifiers.

The rest of this paper is organized as follows. Section 2 reviews some background. Section 3 gives details of the two proposed improvements, followed by a set of experiments in Sect. 4. Section 5 concludes the work and gives suggestions for future work. Section Appendix shows some more experimental results.

## 2 Background

In this section, we review briefly Time Series and 1-Nearest Neighbor Classifier, Euclidean Distance, Dynamic Time Warping, and the framework of semi-supervised time series classification as well as some stopping criterion such as MDL-based stopping criterion, Ratanamahatana and Wanichsan's Stopping Criterion, and lastly we introduce X-means clustering algorithm.

### 2.1 Time Series and 1-Nearest Neighbor Classifier

A time series $T$ is a sequence of real numbers collected at regular intervals over a period of time: $T = t_1, t_2, \ldots, t_n$. Furthermore, a time series can be seen as an $n$-dimensional object in metric space. In 1-Nearest Neighbor Classifier (1-NN), the data object is classified the same class as its nearest object in the training set. The 1-NN has been considered hard to beat in classification of time series data among many other methods such as Artificial Neural Network, Bayesian Network [16].

### 2.2 Euclidean Distance

The Euclidean Distance (ED) between two time series $Q = q_1, q_2, \ldots, q_n$ and $C = c_1, c_2, \ldots, c_n$ is a similarity measure defined as follows:

$$ED(Q, C) = \sqrt{\sum\nolimits_{i=1}^{n} (q_i - c_i)^2}$$

Euclidean distance is one of the most widely used distance measure in time series, its computational complexity is O($n$). In this work, Euclidean Distance is applied only in the X-means clustering algorithm which is used to support the Refinement process described in Subsect. 3.2.

### 2.3 Dynamic Time Warping Distance

One problem with time series data is the distortion in the time axis, making Euclidean distance unsuitable. However, this problem can be effectively addressed by Dynamic

Time Warping (DTW), a distance measure that allows non-linear alignment between the two time series to accommodate sequences that are similar in shape but out of phase [2].

Now we would like to show how to calculate DTW. Given two time series $Q$ and $C$ which have length $n$ and $m$ respectively: $Q = q_1, q_2 ..., q_n$ and $C = c_1, c_2 ..., c_m$. DTW is a dynamic programming technique which calculates all possible warping paths between two time series for finding minimum distance. To calculate DTW between the two above time series, firstly we construct a matrix $D$ with size $m \times n$. Every element in matrix $D$ is cumulative distance defined as:

$$\gamma(i,j) = d(i,j) + \min \begin{cases} \gamma(i-1, j) \\ \gamma(i, j-1) \\ \gamma(i-1, j-1) \end{cases}$$

where $\gamma(i, j)$ is $(i, j)$ element of matrix that is a summation between $d(i, j) = (q_i - c_j)^2$, a square distance of $q_i$ and $c_j$, and the minimum cumulative distance of three adjacent elements to $(i, j)$.

Next, we choose the optimal warping path which has minimum cumulative distance defined as:

$$DTW(Q, C) = \min \sum_{k=1}^{K} w_k$$

where $w_k$ is $(i, j)$ at $k^{th}$ element of the warping path, and $K$ is the length of the warping path.

In addition, for a more accurate distance measure, some global constraints were suggested to DTW. A well-known constraint is Sakoe-Chiba band [7], shown in Fig. 1. The Sakoe-Chiba band constrains the indices of the warping path $w_k = (i, j)_k$ such that $j - r \leq i \leq j + r$, where $r$ is a term defining the allowed range of warping, for a given point in a sequence. Much more detail about DTW is beyond the scope of this paper, interested readers may refer to [3, 7].

Due to evident advantages of DTW for time series data, we incorporate DTW distance measure into our proposed algorithm.

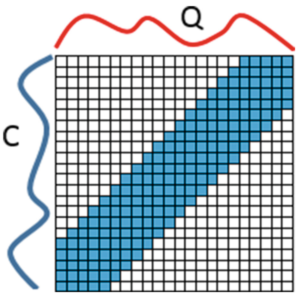


**Fig. 1.** DTW with Sakoe-Chiba band

## 2.4 Semi-Supervised Classification of Time Series

SSC technique can help build better classifiers in situations where we have a small set of labeled data, in addition to abundant unlabeled data. The main ideas of SSC of time series are summarized as follows. Given a set $P$ of positive instances and a set $N$ of unlabeled instances, the algorithm iterates the following two steps:

- *Step* 1: We find the nearest neighbor of any instance of our training set from the unlabeled instances.
- *Step* 2: This nearest neighbor instance, along with its newly acquired positive label, will be added into the training set.

Note that the above algorithm has to be coupled with the ability to stop adding instances at the correct time. This important issue will be addressed later. The algorithm for SSC of time series [1, 8] is given as follows:

```
Self_Training_Classifier (P, N)
  // P: Positive/Labeled set and N: Negative/Unlabeled set
while (the stopping criterion)
        nearest_obj = One_Nearest_Neighbor (P, N)
        $P = P \cup$ {nearest_obj}
        $N = N \setminus$ {nearest_obj}
End
```

Figure 2 illustrates the Semi-Supervised Learning process. The circled instances are the initial positive/labeled instances. The triangle instances are the positive/unlabeled instances, and the rectangle instances are the negative/unlabeled instances. Initially, there are three positive labeled instances (circled instances); the process will assign all the other unlabeled instances as well as their newly acquired labels into the positive set. As we can see, the positive/unlabeled will be added into the training set in a chain which is called the chain effect of this algorithm.

In this semi-supervised classification framework, to identify the point where negative instances are taken into the positive set is an important task as it affects the quality of the final training set. There are some stopping criterions were proposed such as

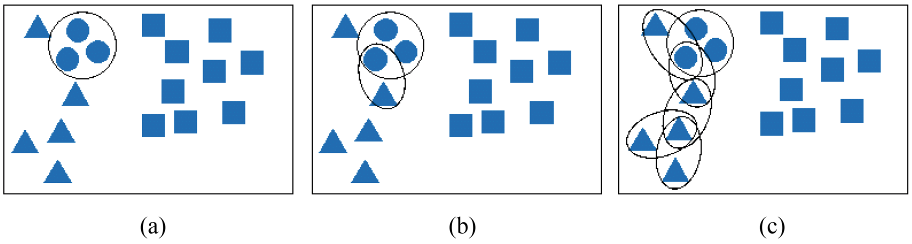


**Fig. 2.** Semi-Supervised Learning on time series data, (a) Initial positive/labeled instances (circled instances), (b) Select one nearest neighbor from unlabeled data (triangle instance) to added in to positive/labeled set, (c) Continue taking more unlabeled instances into positive/labeled set

Ratanamahatana and Wanichsan's Stopping Criterion [6] and Stopping Criterion based on MDL Principle [1], which are depicted in the next two subsections.

## 2.5 Ratanamahatana and Wanichsan's Stopping Criterion

In 2008, Ratanamahatana and Wanichsan [6] proposed a stopping criterion called SCC (Stopping Criterion Confidence) for semi-supervised classification of time series data which is based on the following formula:

$$SCC(i) = \frac{|Mindist(i) - Mindist(i-1)|}{Std\{Mindist(1), Mindist(2), \ldots, Mindist(i)\}} \times \frac{NumInitialUnlabeled - (i-1)}{NumInitialUnlabeled}$$

- *Mindist*: minimum distance in the positive/labeled set after each step of adding one more instance into positive/labeled set.
- *Std*: standard deviation.
- *NumInitialUnlabeled*: the number of unlabeled data at the beginning of the learning phase.

At the point, the value of *SCC* is maximal, i.e. at iteration *i*, the stopping criterion is chose at $i - 2$.

In this work, we use this stopping criterion in order to test the effect of our Refinement process (described later in Subsect. 3.2) for Semi-Supervised Learning.

## 2.6 Stopping Criterion Based on MDL Principle

The Minimum Description Length (MDL) principle is a formalization of Occam's razor in which the best hypothesis for a given set of data is the one that leads to the best compression of the data. The MDL principle was introduced by Rissanen in 1978 [17]. This principle is a crucial concept in information theory and computational learning theory.

The MDL principle is a powerful tool which has been applied in many time series data mining tasks, such as motif discovery [18], criterion for clustering [19], semi-supervised classification of time series [1, 15], discovery rules in time series [21], Compression Rate Distance measure for time series [14]. In this work, we improve a version of MDL for semi-supervised classification of time series which was firstly proposed by Begum et al, in 2013 [1]. The MDL principle is described as follows:

- **Definition 1.** *Discrete Normalization Function*: A discrete function *Dis_Norm* is the function to normalize a real-value subsequence *T* into *b*-bit discrete value of range [1, $2^b$]. The maximum of the discrete range value $2^b$ is also called the *cardinality*. The *Dis_Norm* function is described as follows:

$$Dis\_Norm(T) = round\left(\frac{T - \min}{\max - \min} \times (2^b - 1)\right) + 1$$

where *min* and *max* are the minimum and maximum value in *T* respectively.
After casting the original real-valued data to discrete values, we are interested in determining how many bits are needed to store a particular time series *T*. It is called the *Description Length* of *T*.

- **Definition 2.** *Description Length*: A description length *DL* of a time series *T* is the total number of bits required to represent it.

$$DL(T) = w \times \log_2 c$$

where $w$ is the length of $T$ and $c$ is the cardinality (the number of values we discretize the time series).

- **Definition 3.** *Hypothesis*: A hypothesis *H* is a subsequence used to encode one or more subsequences of the same length.
We are interested in how many bits are required to encode *T* given *H*. It is called the *Reduced Description Length* of *T*.
- **Definition 4.** *Reduced Description Length*: A reduced description length of a time series *T* given hypothesis *H* is the sum of the number of bits required in order to encode *T* exploiting the information in *H*. i.e. *DL*(*T* | *H*), and the number of bits required for *H* itself, i.e. *DL*(*H*). Thus, the reduced description length is defined as:

$$DL(T, H) = DL(H) + DL(T|H)$$

One simple approach of encoding *T* using *H* is to store a *difference vector* between *T* and *H*. Therefore: *DL*(*T* | *H*) = *DL*(*T* – *H*).

Example: Given *A* and *H*, two time series of length 20 as follows:

$$A = [6\,7\,9\,9\,10\ \ 11\,13\,13\,14\,15\,16\,18\,18\,19\,22\,21\,22\,23\,24\,24]$$
$$H = [6\,7\,8\,9\,10\ \ 11\,12\,13\,14\,15\,16\,17\,18\,19\,20\ \ 21\,22\,23\,24\,25]$$

Without encoding, the bit requirement to store *A* and *H* is $2 \times 20 \times \log_2 20 = 173$ bits. The difference vector *A'* = |*A* – *H*| = [0 0 1 0 0 0 1 0 0 0 0 1 0 0 2 0 0 0 0 1]. And in the difference vector, there are 5 mismatches. The bit requirement is now just $20 \times \log_2 20 + 5 \times (\log_2 20 + \lceil \log_2 20 \rceil) = 134$ bits, which brings out a good data compression.

Assume that there exists only a single positive instance as the initial training set [1]. The SSC procedure using MDL-based stopping criterion can be outlined as follows.

First, it selects the seed positive instance as hypothesis. It selects the nearest neighbor of any of the instance(s) in the labeled training set from the unlabeled dataset. It encodes this instance in terms of the hypothesis and keeps the rest of the dataset uncompressed. Then it computes the reduced description length of the whole dataset. If it can achieve a data compression then the instance in question is a true positive. It continues to test to see if unlabeled instances can be added to the positive pool by this data compression criterion. Once the SSC module starts including instances dissimilar to the hypothesis, it no longer achieves data compression and the first occurrence of such an instance is the point where the SSC module should *stop*.

Even though this stopping criterion is the best one for SSC of time series so far, it is still not effective to be used in some situations where time series may have some distortion along the time axis and the way of computing Difference Vector for them becomes so rigid that the stopping point for the classifier can not be found precisely.

In this work, we improve this stopping criterion by applying a non-linear alignment between two time series when calculating their Reduce Description Length (described in Subsect. 3.1).

### 2.7 X-Means Clustering Algorithm

X-means was proposed by Pelleg and Moore in 2000 [5], which is an extended clustering algorithm of K-means. X-means can identify the best number of clusters *k* by itself based on the Bayesian Information Criterion (BIC) [20]. This clustering algorithm requires setting up a more flexible *k* cluster than in K-means. At the beginning, we need to specify a maximal value *max_k* and minimal value *min_k* of *k* clusters. X-means will identify which value of *k* in the range [*min_k*, *max_k*] should be selected. In Fig. 3, we show the outline of X-means which includes two steps. Step 1, called *Improve-Params*, runs K-means until converging. Step 2, called *Improve-Structure*, decides whether a cluster should be split into two sub-clusters or not basing on BIC. The algorithm stops when the number of clusters reaches the maximum number of cluster *max_k* which was set at the beginning.

In this work, we use X-means as a semi-supervised classification method, called *X-means-classifier*. We apply X-means-classifier to support our refinement step to identify the ambiguous instances which will be depicted later in Subsect. 3.2. For more information about X-means algorithm, interested reader can refer to [5].

| | **X-means** |
|---|---|
| 1 | ***Improve-Params*** |
| 2 | ***Improve-Structure*** |
| 3 | If $K > K_{max}$, return the best-scoring model. Otherwise, go to step 1. |

**Fig. 3.** Outline of X-means clustering algorithm [5]

## 3 The Proposed Method

This work aims to improve the MDL-based stopping criterion and at the same time improve the accuracy of the classifier. We devise an improvement technique for the MDL-based stopping criterion and propose a Refinement step to make the classifier more accurate.

## 3.1 New Stopping Criterion Based on MDL Principle

The original MDL-based stopping criterion is really simple, which finds mismatch points by one-to-one alignment between two time series and then calculates Reduced Description Length using the number of mismatch points. In fact, it is hard to find bit saves in this method because the time series may have some distortion in the time axis and a lot of mismatches will be found and there are not many bit saves.

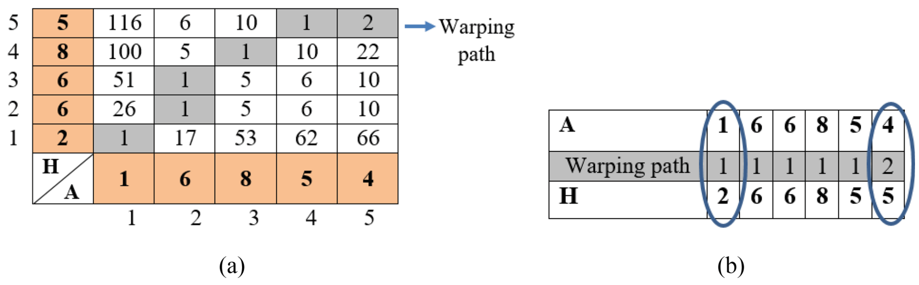
We propose a more flexible technique for finding mismatch points. Instead of linear alignment, we use a non-linear alignment when finding mismatch points. This method attempts to find an optimal matching between two time series for determining as fewer mismatch points as possible.

The principle of our proposed method is in the same spirit of the main characteristic of Dynamic Time Warping (DTW). Therefore, we can modify the algorithm of computing DTW distance between two time series in order to include the finding of mismatch points between them.

Given an example, suppose we have two discrete time series $H$ and $A$ as follows:

$$H = [2\,6\,6\,8\,5]$$
$$A = [1\,6\,8\,5\,4]$$

By original method, the number of mismatch points is 4 because they have different values at 4 positions (2 vs. 1, 6 vs. 8, 8 vs. 5, and 5 vs. 4). On the other hand, by using our *Count_Mismatch* algorithm, the number of mismatch points is 2, less than in the original method. This result can be easily seen in Fig. 4. The alignment between $A$ and $H$ is shown in Fig. 4(a) through the warping path and the number of mismatch points between them is shown in Fig. 4(b).


Warping path
(a)
(b)

| | | | | | | |
|---|---|---|---|---|---|---|
| 5 | **5** | 116 | 6 | 10 | 1 | 2 |
| 4 | **8** | 100 | 5 | 1 | 10 | 22 |
| 3 | **6** | 51 | 1 | 5 | 6 | 10 |
| 2 | **6** | 26 | 1 | 5 | 6 | 10 |
| 1 | **2** | 1 | 17 | 53 | 62 | 66 |
| | **H** / **A** | **1** | **6** | **8** | **5** | **4** |
| | | 1 | 2 | 3 | 4 | 5 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **A** | **1** | **6** | **6** | **8** | **5** | **4** |
| Warping path | 1 | 1 | 1 | 1 | 1 | 2 |
| **H** | **2** | **6** | **6** | **8** | **5** | **5** |

**Fig. 4.** Example of counting mismatch points in our proposed method

Figure 6 shows our proposed mismatch count algorithm based on the calculation of DTW distance. There are two phases in this algorithm. At first phase, we calculate the DTW distance. The second phase goes backward along the found warping path and finds the number of mismatch points. In addition, at first phase, we use Sakoe-Chiba band constraint (through the user-specified parameter $r$) for limiting the meaningless warping paths between the two time series.

In addition, for finding an efficient warping path, we also propose a method to calculate the suitable value of Sakoe-Chiba band $r$ with the algorithm given in Fig. 5. At the beginning, the positive/labeled set must have at least *two* time series. We will calculate the value of $r$ by finding the lowest value of $r$ that satisfies the condition whereby one time series (seed) will accept the other as a positive instance. This condition results in the following inequality that must be satisfied:

$$mismatch\_count \le TS\_length \times \frac{\log_2 card}{\log_2 card + \lceil \log_2 TS\_length \rceil}$$

where *mismatch_count* is the number of mismatch points between two positive/labeled time series, *TS_length* is the length of two time series and *card* is the cardinality.

```
r = Find_Match_Range (T1, T2, card)
// T1, T2: positive/labeled sample time series,
// card: the cardinality
// TS_length: the length of two time series
1  value = TS_length × log2(card)/(log2(card) + ceil(log2(TS_length)))
2  for i = 0 to TS_length
3      mismatch_count = Count_Mismatch (T1, T2, i)
4      if mismatch_count <= value
5          break
6      end
7  end
8  r = i
```

**Fig. 5.** The outline of Refinement process in SSC

Now we will prove the above-mentioned condition.

Proof:

Time series $T1$ accept time series $T2$ as a positive/labeled instance, if and only if the following inequality is satisfied:

$$DL(T1, T2) \le DL(T1) + DL(T2)$$
$$\Leftrightarrow DL(T1) + DL(T2|\,T1) \le DL(T1) + DL(T2)$$

We can derive:

$$DL(T2 \,|\, T1) \le DL(T2)$$
$$\Leftrightarrow mismatch\_count \times (\log_2 card + \lceil \log_2 TS\_length \rceil) \le TS\_length \times \log_2 card$$

So we can rewrite:

$$mismatch\_count \le TS\_length \times \log_2 card / (\log_2 card + \lceil \log_2 TS\_length \rceil) \quad \square$$

```
**mismatch_count = Count_Mismatch (x, y, r)**
  // x: Time series,  y: Time series,  r: Sakoe-Chiba band constraint

// Phase 1: Calculate DTW with Sakoe-Chiba band constraint
$matrix[1,1] = \text{square}(x[1] - y[1])$
for $i = 2$ to length(y) do
    $matrix[1, i] = matrix[1, i - 1] + \text{square}(x[1] - y[i])$
end
for $i = 2$ to length(x) do
    $matrix[i, 1] = matrix[i - 1, 1] + \text{square}(x[i] - y[1])$
end
for $i = 2$ to length(x) do
    for $j = 2$ to length(y) do
        if $|i - j| <= r$ then
            $min\_val = \text{MIN}(matrix[i - 1, j], matrix[i, j - 1], matrix[i - 1, j - 1])$
            $matrix[i, j] = min\_val + \text{square}(x[i] - y[j])$
        else
            $matrix[i, j] = +INFINITY$
        end
    end
end
// Phase 2: Finding minimum number of mismatch points
$i$ = length(x); $j$ = length(y)
$mismatch\_count = 0$
if $x[i] \; != y[j]$ then
    $mismatch\_count = mismatch\_count + 1$
end
while $i > 1$ OR $j > 1$ do
    $value = \text{MIN}(matrix[i - 1, j], matrix[i, j - 1], matrix[i - 1, j - 1])$
    if $i > 1$ AND $j > 1$ AND $value = matrix[i - 1, j - 1]$ then
        $i = i - 1$;  $j = j - 1$
    else if $j > 1$ AND $value = matrix[i, j - 1]$ then
        $j = j - 1$
    else if $i > 1$ AND $value = matrix[i - 1, j]$ then
        $i = i - 1$
    end
    if $x[i] \; != y[j]$ then
        $mismatch\_count = mismatch\_count + 1$
    end
end
```

**Fig. 6.** Mismatch-count algorithm between two time series with Sakoe-Chiba band constraint

Based on the above inequality, we proposed the algorithm for finding the suitable value for the Sakoe-Chiba band $r$, which is given in Fig. 5. Line 3 of the algorithm in Fig. 5 invokes the procedure *Count_Mismatch* which is given in Fig. 6. This algorithm can be easily extended for finding $r$ with more than two initial positive/labeled samples. One solution on this situation is to choose $r$ as the average value of Match Range between any two pairs of positive/labeled time series.

## 3.2 Refinement Step

In this work, we include to the framework of semi-supervised time series classification algorithm given in Subsect. 3.2 a process called *Refinement*. The aim of this process is to check again the training set and modify it in order to obtain a more accuracy classifier. This process is based on the finding of *ambiguous labeled instances*, and these ambiguous instances will be classified again using the confident true labeled instances. The refinement process is iterated until the training set becomes stable, i.e. the training set before and after a refinement iteration are the same.

Figure 7 shows our proposed refinement algorithm. In this algorithm, *AMBI* is the set of ambiguous labeled instances, *P* is the positive set and *N* is the negative set. The set *AMBI* consists of the instances which are near the positive and negative boundary. This algorithm classifies the instances in *AMBI* basing on the current *P* and *N*. The process of detecting *AMBI* and classifying the instances in *P* is repeated until *P* and *N* are unchanged. Finally, the instances in *AMBI* that cannot be labeled will be classified the last time.

```
Refinement (P, N)
   // P: positive/labeled set (output of  Improved MDL method)
   // N: negative/unlabeled set (output of Improved MDL method)
1  AMBI = Find ambiguous instances in P and N
2  P = P – AMBI; N = N – AMBI
3  repeat
4     Classify AMBI by new training set P and N and then add each classified instance
      to P and N.
5     AMBI = Find ambiguous instances in P and N
6     P = P – AMBI;  N = N – AMBI
7  until (P and N are unchanged)
8  Classify AMBI by new training set P and N and then add each classified instance to
   P and N.
```

**Fig. 7.** The outline of Refinement process in SSC

The ambiguous instance detection process is done under the following rules:

1. The instances in *P* which were classified as positive by SSC but their nearest neighbors are in the negative set *N*, they and their nearest neighbors are ambiguous.
2. The instances in *N* which were classified as negative by SSC but their nearest neighbors are in the positive set *P*, they and their nearest neighbors are ambiguous.
3. The instances which were classified as positive by *X-means-classifier* (explained later) but are classified as negative by SSC, these are considered ambiguous.

The process of classifying instances in *AMBI* is done using One-Nearest-Neighbor (1-NN) in which the instance in *AMBI* which is nearest to *P* or *N* will be labeled first.

In this work, we propose a method called *X-means-Classifier* that can be used as SSC method for time series. This is a clustering-based approach which applies *X-means* algorithm, an extended variant of *k-means* which was proposed by Pelleg and Moore in

2000 [5]. One outstanding feature of *X-means* is that it can automatically estimate the suitable number of clusters during the clustering process. The SSC method based on *X-means* consists of the following steps. First, we use *X-means* to cluster the training set (including positive and unlabeled instances). Then, if there exists one cluster which contains the positive instance, all the instances in it will be classified as positive instances, and all the rest are classified as negative. *X-means-Classifier* will be used to initialize the *AMBI* in the Refinement process (Line 1 in the algorithm in Fig. 7).

In Fig. 8, we show an example to illustrate how the Refinement process works. In Fig. 8(a), the circled/positive instances and squared/negative instances are obtained from the Self-Learning process. The separate line which split the space into two areas *P* and *N* indicates the true boundary between two classes *P* and *N*. As we can see from Fig. 8(a), there are three wrongly classified instances, two squared instances indicate that they belong to negative set but their true class is positive (they stand in area *P*), and one circled instance indicates that it belong to positive set but their true class is negative (because it locates in *N* area). When applying the Refinement process, some ambiguous instances are identified because their nearest neighbors belong to another class as shown in Fig. 8(b). Since, they are reclassified as shown in Fig. 8(c). In Fig. 8(d), the Refinement process is continued, two more instances are identified as ambiguous instances. They are finally reclassified as in Fig. 8(e). The Refinement step repeats until there is no change in the positive set and the negative set.

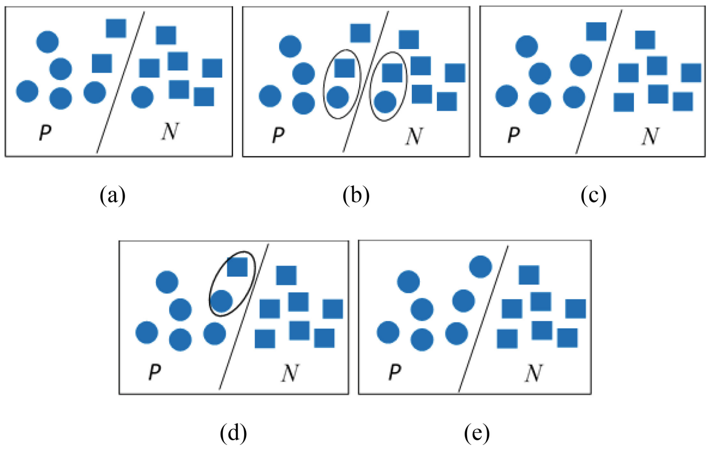


**Fig. 8.** An example of Refinement step, (a) positive set *P* and negative set *N* after applying Self-Learning with improved MDL-based stopping criterion, (b) ambiguous instances are identified (the two pair of instances marked), (c) the ambiguous instances are reclassified, (d) continuing to identify ambiguous instances, (e) the final training set after Refinement step

## 4 Experimental Evaluation

We implemented our proposed method and previous methods with Matlab 2012 and conducted the experiments on the Intel Core i7-740QM 1.73 GHz, 4 GB RAM PC. After the experiments, we evaluate the classifier by measuring the precision, recall and

F-measure of the retrieval. The precision is the ratio of the correctly classified positive test data to the total number of test instances classified as positive. The recall is the ratio of the correctly classified positive test data to the total number of all positive instances in the test dataset. An F-measure is the ratio defined by the formula:

$$F = \frac{2 \times p \times r}{p + r}$$

where $p$ is precision and $r$ is recall.

$$p = \frac{\#\,of\ correct\,positive\,predictions}{number\ of\ positive\,predictions}$$
$$r = \frac{\#\,of\ correct\,positive\,predictions}{number\ of\ positive\ examples}$$

In general, the higher the F-measure is, the better the classifier is.

## 4.1 Datasets

Our experiments were conducted over the datasets from UCR Time Series Classification Archive [4]. Details of these datasets are shown in Table 1. Besides, we also use two other datasets: MIT-BIH Supraventricular Arrhythmia Database, and St. Petersburg Arrhythmia Database that are used to compare the stopping criteria. These two datasets are available in [9] and featured as follows:

- *MIT-BIH Supraventricular Arrhythmia Database*: This database includes many ECG signals and a set of beat annotations by cardiologists. Record 801 and signal ECG1 were used in our experiments as in [1] because we compared our method with [1]. The target class in the 2-class classification problem is abnormal heartbeats.

**Table 1.** Datasets used in the evaluation experiments

| Datasets | Number of classes | Size of dataset | Time series length |
|---|---|---|---|
| Yoga | 2 | 300 | 426 |
| Words synonyms | 25 | 267 | 270 |
| Two patterns | 4 | 1000 | 128 |
| MedicalImages | 10 | 381 | 99 |
| Synthetic control | 6 | 300 | 60 |
| TwoLeadECG | 2 | 23 | 82 |
| Gun-Point | 7 | 50 | 150 |
| Fish | 7 | 175 | 463 |
| Lightming-2 | 2 | 60 | 637 |
| Symbols | 6 | 25 | 398 |

- *St. Petersburg Arrhythmia Database*: This database contains 75 annotated readings extracted from 32 Holter records. Record I70 and signal II were used in our experiments as in [1] because we compared our method with [1]. The target class in the 2-class classification problem is R-on-T Premature Ventricular Contraction.

## 4.2 Parameters Setup

Cardinality for the MDL principle (described in Subsect. 2.6) is set to 8 (3-bit discrete values). For all the methods, we use DTW as distance measure. Euclidean Distance is applied only in X-means-classifier.

## 4.3 Comparing Two MDL-Based Stopping Criteria

We perform a comparison between our improvement technique and the previous MDL-based stopping criteria [1] on four datasets: MIT-BIH Supraventricular Arrhythmia Database, St. Petersburg Arrhythmia Database, Gun Point Training Set and Fish Training Set in Figs. 9, 10, 11 and 12 respectively. In order to compare the stopping criteria, we record the point when the truly negative instance is added into the positive set of Self-Learning process, this point is consider as expected stopping point. We compare the stopping criteria based on this expected stopping point as a baseline.

From Figs. 9, 10, 11 and 12, we can see that our improvement technique suggests a better stopping point in most of the datasets. Detecting a good stopping point is very crucial in SSC of time series. We attribute this desirable advantage of our improvement technique to the flexible way of determining mismatches between two time series when computing Reduced Description Length of one time series exploiting the information in the other.

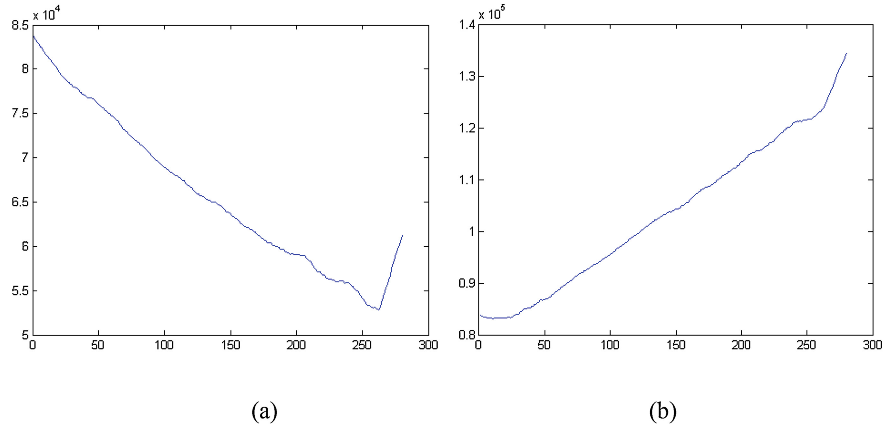
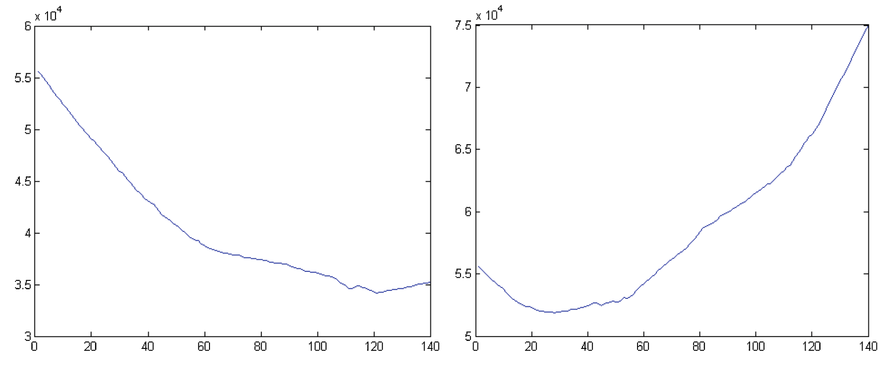
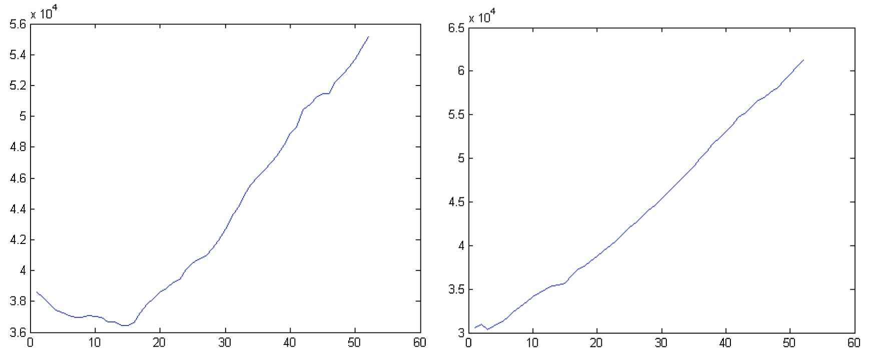


**Fig. 9.** In MIT-BIH Supraventricular Arrhythmia Database, the expected stopping point is 268. (a) Stopping point by our MDL (Proposed Method) at iteration 262 (Nearly perfect). (b) Stopping point by MDL (Previous Method) at iteration 10 (too early).



(a) (b)

**Fig. 10.** In St. Petersburg Arrhythmia Database, the expected stopping point is 126. (a) Stopping point by our MDL (Proposed Method) at iteration 121 (Nearly perfect). (b) Stopping point by MDL (Previous Method) at iteration 28 (too early)



(a) (b)

**Fig. 11.** In Gun Point Training Set, the expected stopping point is $14^{th}$. (a) Stopping point by our MDL (Proposed Method) at iteration $15^{th}$ (Nearly perfect). (b) Stopping point by MDL at iteration $3^{rd}$ (too early).

Figure 9 shows the experimental results of our proposed MDL based stopping criterion compared with the previous MDL based stopping criterion MIT-BIH Supraventricular Arrhythmia Database. Our proposed stopping point is 268 which is nearly the same as expected stopping point 262, and much better than that of the previous method at 10.

Figures 10, 11 and 12 also reveal that our improvement can produce a more accurate stopping point than the previous stopping criterion. In St. Petersburg Arrhythmia Database (Fig. 10), the expected stopping point is 126; our proposed method gives result 128, whereas the previous method gets 28 as stopping point.

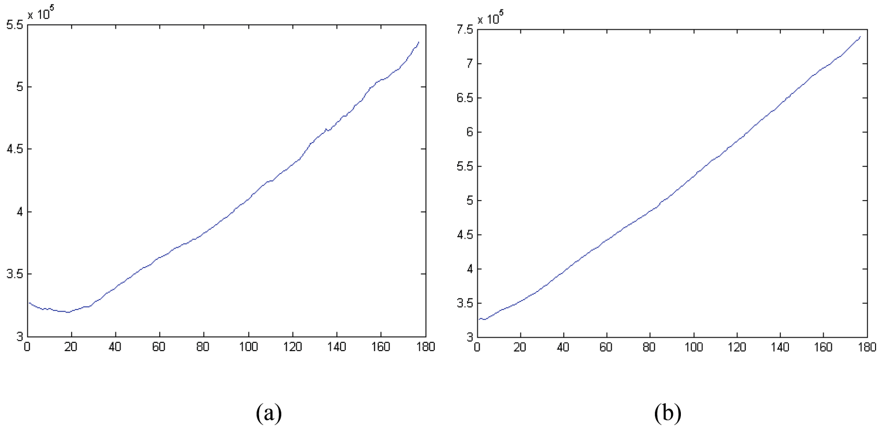


**Fig. 12.** In Fish Training Set, the expected stopping point is 18th. (a) Stopping point by our MDL (Proposed Method) at iteration 19th (Nearly perfect). (b) Stopping point by MDL at iteration 3rd (too early).

In Gun Point (Fig. 11), the expected stopping point is 14; our proposed method gives result 15, whereas the previous method gets 3 as stopping point. And in Fish dataset (Fig. 12), the expected stopping point is 18; our proposed method gives result 19, whereas the previous method gets 3 as stopping point.

## 4.4 Effects of Refinement Step

In this subsection, we compare SSC by our new MDL-based stopping criterion with and without Refinement step. Table 2 reports the experimental results (precision, recall and F-measure) of this comparison. The results show that our proposed Refinement step brings out better performance in all the datasets. In most of datasets, the

**Table 2.** Experiment results with and without Refinement (used proposed stopping criterion)

| Datasets | Without Refinement | | | With Refinement | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F-measure | Precision | Recall | F-measure |
| Yoga | 0.64 | 0.35036 | 0.45283 | 0.57609 | 0.38686 | **0.46288** |
| WordsSynonyms | 0.94737 | 0.3 | 0.4557 | 0.625 | 0.41667 | **0.5** |
| Two patterns | 1.0 | 0.41328 | 0.58486 | 1.0 | 0.68635 | **0.814** |
| MedicalImages | 0.57276 | 0.91133 | 0.70342 | 0.56587 | 0.93103 | **0.70391** |
| Synthetic control | 1.0 | 0.08 | 0.14815 | 1.0 | 0.98 | **0.9899** |
| TwoLeadECG | 0.88889 | 0.66667 | 0.7619 | 0.75 | 1.0 | **0.85714** |
| Gun-Point | 0.93333 | 0.58333 | 0.71795 | 1.0 | 0.625 | **0.76923** |
| Fish | 0.94737 | 0.81818 | 0.87805 | 1.0 | 0.86364 | **0.92683** |
| Lightning-2 | 0.7619 | 0.4 | 0.52459 | 0.6875 | 0.55 | **0.61111** |
| Symbols | 1.0 | 0.75 | 0.85714 | 0.88889 | 1.0 | **0.94118** |

performance of the proposed method is better, for example, on Two-Paterns F-measure = 81.4 %, on Synthetic-Control F-measure = 98.99 %, on TwoLeadECG F-measure = 85.714 %, on Fish F-measure = 92.683 %, on Symbol F-measure = 94.118 %. Specially, on the Synthetic-Control dataset, SSC without Refinement gives F-measure = 14.815 %, while with Refinement, F-measure reaches to 98.99 %, a perfect result. These experimental results show that the Refinement step in SSC can improve the accuracy of the classifier remarkably.

Now we show the effect of Refinement step by using Ratanamahatana and Wanichsan's Stopping Criterion [6]. Table 3 indicates the precision, recall and F-measure with and without Refinement. The results also reveal that our Refinement step helps to bring better classifier. On Two Paterns dataset F-measure = 100 %, on Synthetic Control F-measure = 98.99 %, on Fish F-measure = 88.372. On Yoga, WordsSynonyms, and Symbols training set, the F-measure decreases with an insignificant amount.

**Table 3.** Experiment results with and without Refinement (used Ratanamahatana and Wanichsan's Stopping Criterion [6])

| Datasets | Without Refinement | | | With Refinement | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F-measure | Precision | Recall | F-measure |
| Yoga | 0.6383 | 0.43796 | 0.51948 | 0.58654 | 0.44526 | 0.50622 |
| WordsSynonyms | 0.58696 | 0.9 | 0.71053 | 0.45669 | 0.96667 | 0.62032 |
| Two patterns | 0.99267 | 1.0 | 0.99632 | 1.0 | 1.0 | **1.0** |
| MedicalImages | 0.96078 | 0.24138 | 0.38583 | 1.0 | 0.24138 | **0.38889** |
| Synthetic control | 0.95918 | 0.94 | 0.94949 | 1.0 | 0.98 | **0.9899** |
| TwoLeadECG | 1.0 | 0.41667 | 0.58824 | 0.71429 | 0.83333 | **0.76923** |
| Gun-Point | 0.63636 | 0.58333 | 0.6087 | 0.68182 | 0.625 | **0.65217** |
| Fish | 0.9 | 0.81818 | 0.85714 | 0.90476 | 0.86364 | **0.88372** |
| Lightning-2 | 0.62162 | 0.575 | 0.5974 | 0.69388 | 0.85 | **0.76404** |
| Symbols | 0.53333 | 1.0 | 0.69565 | 0.5 | 1.0 | 0.66667 |

## 5 Conclusions

Existing semi-supervised learning algorithms for time series classification still have less than satisfactory performance. In this work, we have proposed two novel improvements for semi-supervised classification of time series: an improvement technique for MDL-based stopping criterion and a refinement step to make the classifier more accurate. Our former improvement applies the Dynamic Time Warping to find a non-linear alignment between two time series when computing their Reduced Description Length. The latter improvement attempts to identify wrongly classified instances by self-learning process and reclassify these instances. Experimental results reveal that our two improvements can construct more accurate semi-supervised time series classifiers.

As for future work, we plan to generalize our method to the case of multiple classes and adapt it to some other distance measures such as Complexity-Invariant Distance [13] or Compression Rate Distance [14]. Compression Rate Distance is a powerful distance measure for time series data which we recently proposed. We also plan to include some constraint in the Semi-Supervised Learning process as in [15] and extend our method to perform semi-supervised classification for streaming time series. Besides, we intend to apply another version of MDL such as in [14, 15, 19] which computes the Description Length of a time series by its entropy. Although our method helps to improve the F-measure of the output training set, there are still many instances which were wrongly classified in the training set. This weakness could be solved by removing the wrongly classified instances.

**Acknowledgment.** We would like to thank Prof. Eamonn Keogh and Nurjahan Begum for kindly sharing the datasets which help us in constructing the experiments in this work.

## Appendix A: Some More Experimental Results

This section shows the experimental results of X-means-classifier which was used to support the Refinement step shown in Subsect. 4.4. Table 4 illustrates the precision, recall and F-measure of X-means classifier. The experiments reveal that X-means classifier gives good results in some datasets such as in Synthetic Control F-measure = 100 %, in Symbols F-measure = 94.118 %, in Gun Point F-measure = 71.795 %, in Fish F-measure = 71.795 %. Specially, in Synthetic Control, the result is perfect F-measure = 100 % (totally exact).

**Table 4.** Semi-supervised classification of time series by X-means-Classifier

| Datasets | Precision | Recall | F-measure |
|---|---|---|---|
| Yoga | 0.48421 | 0.33577 | 0.39655 |
| WordsSynonyms | 0.35632 | 0.51667 | 0.42177 |
| Two patterns | 0.28676 | 0.28782 | 0.28729 |
| MedicalImages | 0.71277 | 0.33005 | 0.45118 |
| Synthetic control | 1.0 | 1.0 | 1.0 |
| TwoLeadECG | 0.6 | 0.75 | 0.66667 |
| Gun-Point | 0.93333 | 0.58333 | 0.71795 |
| Fish | 0.82353 | 0.63636 | 0.71795 |
| Lightning-2 | 0.75 | 0.525 | 0.61765 |
| Symbols | 0.88889 | 1.0 | 0.94118 |

In Table 5, we show the execution time (seconds) of some algorithms: Refinement with Improved MDL based stopping criterion, Refinement with Ratanamahatana and Wanichsan's stopping criterion, and X-means-classifier. Note that these figures do not include the execution time of Self-Learning process.

**Table 5.** The execution time (seconds) of each algorithm

| Datasets | Improved MDL based stopping criterion with Refinement | Ratanamahatana and Wanichsan's stopping criterion with Refinement | *X*-means |
|---|---|---|---|
| Yoga | 32.158997 | 5.398214 | 5.349535 |
| WordsSynonyms | 17.201075 | 2.3947555 | 2.313145 |
| Two patterns | 23.461983 | 10.6709155 | 7.588841 |
| MedicalImages | 5.4952995 | 1.2302655 | 1.124553 |
| Synthetic control | 1.8147455 | 0.623597 | 0.564281 |
| TwoLeadECG | 0.3780895 | 0.038229 | 0.030232 |
| Gun-Point | 0.4166765 | 0.1593195 | 0.147925 |
| Fish | 6.669053 | 2.237528 | 2.202647 |
| Lightning-2 | 4.60413 | 0.777428 | 0.763867 |
| Symbols | 0.964603 | 0.2640985 | 0.258735 |

## References

1. Begum, N., Hu, B., Rakthanmanon, T., Keogh, E.J.: Towards a minimum description length based stopping criterion for semi-supervised time series classification. In: IEEE 14th International Conference on Information Reuse and Integration, IRI 2013, San Francisco, CA, USA, August 14–16, 2013, pp. 333–340 (2013)
2. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, July 1994. Technical report WS-94-03, pp. 359–370 (1994)
3. Keogh, E.J., Ratanamahatana, C.A.: Exact indexing of dynamic time warping. Knowl. Inf. Syst. **7**(3), 358–386 (2005)
4. Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The UCR time series classification archive, July 2015. www.cs.ucr.edu/∼eamonn/time_series_data/
5. Pelleg, D., Moore, A.W.: X-means: extending k-means with efficient estimation of the number of clusters. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29–July 2, 2000, pp. 727–734 (2000)
6. Ratanamahatana, C.A., Wanichsan, D.: Stopping criterion selection for efficient semi-supervised time series classification. In: Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 1–14 (2008)
7. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Signal Process. **26**(1), 43–49 (1978)
8. Wei, L., Keogh, E.J.: Semi-supervised time series classification. In: Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20–23, 2006, pp. 748–753 (2006)
9. Begum, N.: Minimum description length based stopping criterion for semi-supervised time series classification (2013). www.cs.ucr.edu/∼nbegu001/SSL_myMDL.htm
10. Marussy, K., Buza, K.: SUCCESS: a new approach for semi-supervised classification of time-series. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013, Part I. LNCS, vol. 7894, pp. 437–447. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38658-9_39

11. Nguyen, M.N., Li, X.L., Ng, S.K.: Positive unlabeled learning for time series classification. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI 2011, vol. 2, pp. 1421–1426. AAAI Press (2011)
12. Nguyen, M., Li, X.-L., Ng, S.-K.: Ensemble based positive unlabeled learning for time series classification. In: Lee, S., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 243–257. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29038-1_19
13. Batista, G.E.A.P.A., Keogh, E.J., Tataw, O.M., de Souza, V.M.A.: CID: an efficient complexity-invariant distance for time series. Data Min. Knowl. Discov. **28**(3), 634–669 (2014)
14. Vinh, V.T., Anh, D.T.: Compression rate distance measure for time series. In: 2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19–21, 2015, pp. 1–10 (2015)
15. Vinh, V.T., Anh, D.T.: Constraint-based MDL principle for semi-supervised classification of time series. In: 2015 Seventh International Conference on Knowledge and Systems Engineering, KSE 2015, Ho Chi Minh City, Vietnam, October 8–10, 2015, pp. 43–48 (2015)
16. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.J.: Querying and mining of time series data: experimental comparison of representations and distance measures. PVLDB **1**(2), 1542–1552 (2008)
17. Rissanen, J.: Modeling by shortest data description. Automatica **14**(5), 465–471 (1978)
18. Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of time-series motif from multidimensional data based on MDL principle. Mach. Learn. **58**(2–3), 269–300 (2005)
19. Rakthanmanon, T., Keogh, E.J., Lonardi, S., Evans, S.: MDL-based time series clustering. Knowl. Inf. Syst. **33**(2), 371–399 (2012)
20. Schwarz, G.E.: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)
21. Shokoohi-Yekta, M., Chen, Y., Campana, B.J.L., Hu, B., Zakaria, J., Keogh, E.J.: Discovery of meaningful rules in time series. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10–13, 2015, pp. 1085–1094 (2015)