International Journal of Computational Intelligence Systems (2024) 17:89
https://doi.org/10.1007/s44196-024-00461-4

RESEARCH ARTICLE

Finding Discriminative Subsequences Via a Coverage Measure and Mutual Information Selection Strategy for Multi-Class Time Series Classification

Jun Yang[1,2,3] · **Siyuan Jing**[1,3]

Received: 11 September 2023 / Accepted: 17 March 2024

Abstract
Time series classification (TSC) has attracted considerable attention from the data mining community over the past decades. One of the effective ways to handle this task is to find discriminative subsequences in time series to train a classifier. Obviously, how to measure the discriminative power of subsequences and find the optimal combination of subsequences is crucial to the accuracy of TSC. In this paper, we introduce a new method, CRMI, to find high-quality discriminative subsequences for multi-class time series classification (MC-TSC). Different from existing methods, there are two significant innovations in the work. At first, we propose a novel measure, named *coverage ratio*, to evaluate the discriminative power of a subsequence based on a coverage matrix which is figured out by the clustering technique. Second, a heuristic algorithm based on mutual information (MI) is proposed to find the optimal combination of subsequence candidates. The calculation of MI is also based on the coverage matrix. Extensive experiments were conducted on 54 UCR time series datasets with at least 3 categories, and the results show that (1) the proposed algorithm achieves the highest average accuracy and outperforms most of the existing shapelet-based TSC algorithms; (2) compared with existing methods, the proposed algorithm performs better on datasets with a large number of categories.

Keywords Multi-class time series classification · Discriminative subsequence · Coverage ratio · Mutual information · Clustering

1 Introduction

Time series classification (TSC) has attracted considerable attention from the data mining community over the past decades because of the increase of time series data from various domains, such as the Internet of Things, finance, medicine, etc. Similar to the classification task in machine learning, TSC aims to build a classifier based on a time series training dataset and predict labels of target time series [1, 2]. Up to now, the technologies of TSC can be divided into four categories. The first is the whole-series comparison algorithm which combines classifiers, such as 1-Nearest Neighbor with similarity metrics, such as Euclidean Distance (ED) or Dynamic Time Warping (DTW) distance. The second category is the Deep Neural Network based TSC algorithm (DNN-TSC) which has been a popular topic in the field of machine learning recently [3]. The third category is the ensemble algorithm which combines two or more popular TSC algorithms and employs a voting strategy to determine the label of the target time series. These two categories of algorithms, such as ResNet [4], InceptionTime [5], TapNet [6], Elastic Ensemble [7], COTE [8, 9], Proximity Forest [10], etc., can achieve competitive results on most datasets. However, most of them cannot explain why a target time series is assigned to a particular label. Besides, training a DNN-TSC model or an ensemble classifier for TSC requires massive training data as well as considerable computing resources. The last category is the pattern-based

✉ Siyuan Jing
syjing628@126.com

[1] Sichuan Provincial Key Laboratory of Philosophy and Social Science for Language Intelligence in Special Education, Leshan Normal University, Leshan 614000, China

[2] Chengdu Computer Application Institute CAS, Chengdu Information Technology of Chinese Academy of Sciences Co. Ltd., Chengdu 610041, China

[3] Key Lab of Internet Natural Language Intelligent Processing of Sichuan Provincial Education Department, Leshan Normal University, Leshan 614000, China

Published online: 11 April 2024

TSC algorithm which aims to find out some local patterns that can discriminate time series from different classes. The advantages of the pattern-based TSC algorithm are that it not only achieves high classification accuracy but also provides good interpretability. In this work, we focus on the pattern-based TSC algorithm.

For TSC, a high-quality pattern is essentially a discriminative subsequence which is helpful to improve the accuracy of classification. There are two approaches to the pattern-based TSC, which are the shapelet-based algorithm [11] and the dictionary-based algorithm [12]. Generally, the shapelet-based algorithm extracts a number of subsequences from the raw numeric time series and employs information gain (IG) to measure their classification ability. The subsequences with the highest IG will be regarded as shapelets and be used to build a classifier. For the dictionary-based algorithm, a classic example is the bag-of-patterns (BOP) algorithm [13] which involves converting a time series into a discrete series using Symbolic Aggregate approXimation (SAX), creating a set of SAX words for each series through the application of a short sliding window, and then using the frequency count of the words in a series as the pattern. Senin et al. [14] extend the BOP algorithm in SAX-VSM thereby computing TF-IDF weights for each word and label. Schäfer [15] proposes the BOSS algorithm which converts time series using Symbolic Fourier Approximation (SFA) and then creates a dictionary of SFA symbols represented patterns.

Obviously, there are two key issues in pattern-based TSC, i.e., how to measure the discriminative power of subsequences and how to find the optimal combination of subsequences. Ye and Keogh [15] who first proposed shapelets employ a brute force algorithm to enumerate subsequences from the raw time series dataset and use IG to measure the quality of subsequences. Since the number of subsequences is quadratic to the length of the time series and the IG requires calculating the distance between all the subsequences and time series, the complexity of the algorithm is $O\left(n^2 \cdot m^4\right)$ where n is the number of time series and m is the length of time series. Although Keogh et al. [16, 17] proposed some techniques, e.g., candidate pruning, random masking, etc., to accelerate the discovery of shapelets, they sacrifice the accuracy of the classification. Moreover, their method only selects the subsequence with the highest IG score to be the shapelet to build a decision tree classifier which affects the accuracy of the TSC. Hills et al. [18] proposed a new algorithm named shapelet transformation (ST) which finds the top-k shapelets to produce a transformed dataset. Since the transformed dataset removes the temporal relation from the time series, the ST method can be combined with any machine learning classifiers, such as SVM, random forest, neural network, etc., which significantly improves the accuracy of the shapelet-based classifier. However, the ST algorithm requires setting some parameters to limit the final shapelet lengths which are sensitive to both datasets and algorithms. Moreover, it also needs to evaluate all the subsequences which is computationally expensive. To handle this problem, some shapelet-based methods replace the brute force algorithm with random sampling to improve the efficiency of the algorithm, such as gRSF [19], CRSF [20], ELIS [21], BSPCover [22], etc.

To address the two issues mentioned above, we propose a new method, Coverage Ratio and Mutual Information (CRMI), to find discriminative subsequences for multi-class time series classification (MC-TSC). The first step of CRMI is random sampling for several minutes that obtain a large number of subsequence candidates. Then, it calculates a distance matrix in which each cell is the distance between a subsequence candidate and a time series. Based on the distance matrix, a clustering technique is employed to figure out a coverage matrix which is used to measure the discriminative power of subsequence candidates. In this way, it can efficiently determine subsequence candidates that maximally represent each time series class. Next, a heuristic algorithm based on mutual information (MI) is designed to find the optimal combination of subsequence candidates. Finally, we also adopt the ST method to produce a transformed dataset and build a TSC classifier. The major contributions of this work can be summarized as follows:

(1) We propose an efficient algorithm named CRMI to find discriminative subsequences for MC-TSC. Different from the shapelet-based algorithms and the dictionary-based algorithms, we exploit the clustering technique to build a coverage matrix and propose a new measure, named *coverage ratio* (CR), to evaluate the discriminative power of subsequence candidates. Moreover, we consider the effect of feature combinations and propose a heuristic algorithm based on MI to find the optimal combination of subsequences.

(2) Extensive experiments were performed on 54 UCR [23] time series datasets with at least three categories to evaluate the effectiveness of the proposed algorithm. First, we explore the parameter setting in the CRMI algorithm although it has only two parameters. Second, ablation experiments were performed to demonstrate the effectiveness of our methods. Finally, we compare the CRMI algorithm with seven classic algorithms, including the 1NN + DTW and six state-of-the-art shapelet-based TSC algorithms. The efficiency of the algorithm is also discussed.

The rest of the paper is organized as follows. In Sect. 2, we recall some related works. Section 3 gives the symbols, concepts, and definitions in the paper. In Sect. 4, we introduce the CRMI algorithm in detail. In Sect. 5, the design of the experiment and analysis of experimental results are presented. Finally, we summarize the findings of this work in Sect. 6.

2 Related Works

In this section, we review some related works of TSC. A classic approach to TSC is 1NN + DTW, which has been proven by Wang et al. [24] to be hard to beat. Given a target time series, it calculates its distance to all the training time series using DTW and adopts the 1NN algorithm to determine its label.

It is known to all that there are enormous works on TSC, however, the shapelet-based methods are close to our work which also aims to find discriminative subsequences for building classifier models. Therefore, we review some important works about shapelet-based TSC. Ye and Keogh first proposed the concept of shapelets in [16], and shapelets are time series subsequences that are maximally representative of a class.

In this groundbreaking work, the IG is employed to measure the quality, i.e., discriminative power, of subsequence candidates, and a brute force algorithm is used to enumerate subsequences from the raw time series. Since the number of subsequences is quadratic to the length of the time series and the IG requires calculating the distances between all the subsequences and time series, the complexity of the algorithm is $O\left(n^2 \cdot m^4\right)$ where n is the number of time series and m is the length of time series. To improve the efficiency of the algorithm, Keogh et al. proposed fast shapelets (FS) [25] which exploit a random projection technique on the SAX representation of time series to find potential shapelet candidates. Its time complexity is $O\left(n \cdot m^2\right)$ which outperforms the previous work by two or three orders of magnitude. However, the FS algorithm cannot guarantee finding the best shapelet and it sacrifices the accuracy of the classification.

Grabocka et al. proposed a Scalable shapelet Discovery algorithm (SD) [26], which used the Piecewise Aggregate Approximation (PAA) to represent time series and adopted an online clustering/pruning technique to avoid measuring the prediction accuracy of similar candidates in Euclidean distance space and incorporated a supervised shapelet selection to improve classification accuracy. Since the SD algorithm prunes 99% of shapelet candidates, it was three -four orders of magnitudes faster than FS.

Tight coupling of shapelet discovery and training of decision tree is another factor that hinders the accuracy improvement. Hill et al. proposed a ST [18] technique, which separates the shapelet discovery from the classifier fitting, which can cooperate with any existing classifiers, e.g., SVM, kNN, neural network, etc. The ties are resolved by constructing a new feature space based on the top-k best shapelets and then transforming the original time series data into the new feature space in which each data point is the similarity (i.e., distance) between the corresponding shapelet and the time series. However, ST is one of the slowest shapelet-based algorithms since it needs to evaluate all possible candidates.

Karlsson et al. proposed a generalized random shapelet forest (gRSF) [19] algorithm which consists of a set of shapelet-based decision trees, where both the choice of instances used for building a tree and the choice of shapelets are random. The experiments prove its effectiveness and efficiency. However, some techniques in gRSF still decrease the efficiency of the algorithm, e.g., the growth of a decision tree needs to repetitively sample shapelets for each tree node. Yang et al. proposed a compressed random shapelet tree (CRSF) [20] algorithm to improve the gRSF algorithm by three techniques that are the SAX representation, an innovative SAX distance measuring, and a shapelet-pool strategy for generating shapelet-based decision trees.

Li et al. proposed BSPCover [22] which focuses on the discovery of a set of high-quality shapelet candidates for model building. BSPCover prunes identical and highly similar shapelet candidates then uses the p-cover algorithm to determine discriminative shapelet candidates, and finally applies the existing shapelet-learning technique to build the classifier. They reported that BSPCover achieves very competitive performance compared with the existing TSC methods.

3 Preliminaries

A time series is an ordered sequence of m real values, denoted as $T_i = t_{i,1}t_{i,2}t_{i,3}\cdots t_{i,m}$, where m is the length of the time series. Usually, the m real values are obtained by observing a target object at a fixed time interval. A time series dataset contains n time series instances which can be denoted as $D = \langle T, Y\rangle, T = \left\{T_1, T_2, \cdots, T_n\right\}, Y = \left\{y_1, y_2, \cdots, y_Q\right\}$, where T is the time series set, Y is the set of the time series labels, and Q is number of distinct labels. The label of T_i can be obtained by a function $label\left(T_i\right)$. The proposed algorithm is appropriate for MC-TSC with $Q \geq 3$.

A time series subsequence $T_i^{j,k} = t_{i,j}t_{i,j+1}\cdots t_{i,j+k-1}$ is a consecutive sequence obtained from a time series T_i where j and k are the starting position and the length of the sequence, respectively. A discriminative subsequence is a specific time series subsequence that can strongly represent a class of time series. For simplicity, we also use symbol s to denote a time series subsequence. A discriminative subsequence is a specific time series subsequence that can strongly represent a class of time series. Finding the discriminative subsequence, requires calculating the distance between a subsequence and a time series. Regarding a subsequence as another time series with a shorter length, the distance between two time series with different length is defined below.

Definition 1 (Distance between two time series) The distance between time series T_1 with length m_1 and time series T_2 with length m_2 can be calculated by Formula (1). Without loss of generality, we assume $m_1 < m_2$.

$$d(T_1, T_2) = \min_{j=0\ldots m_2-m_1} \sqrt{\frac{1}{m_1}\sum_{i=1}^{m_1}\left(t_{1,i} - t_{2,j+i}\right)^2} \quad (1)$$

It is not difficult to find that the distance is a variation of Euclidean distance. The proposed method adopts transformation, just like the ST algorithm, that converts the raw time series dataset to a distance matrix, also called transformation matrix.

Definition 2 (Subsequence transformation) Assuming a set of subsequences $S = \left\{s_1, s_2, \cdots, s_{n_S}\right\}$ has been extracted from a time series dataset $D = \langle T, Y\rangle$, it can transforms the time series dataset to a distance matrix, $D' = \left\{d_{i,j}\right\}$, where $d_{i,j}$ is the distance between the i-th time series and the j-th subsequence, i.e., $d\left(s_j, T_i\right)$. Note that all distances need to be normalized.

4 Finding Discriminative Subsequences for TSC

4.1 Framework of Algorithm

This paper proposes an algorithm, CRMI, for finding discriminative subsequences from time series to handle the TSC problem. The framework of the algorithm is shown in Fig. 1. The CRMI algorithm performs random sampling on raw time series dataset and extracts a large number of subsequence candidates at first. This step is the same with some existing works. Then, it calculates the distances between each subsequence candidate and all the time series based on Eq. 1 and forms a distance matrix. The third step is clustering performed on the distance matrix which aims to reveal the coverage relation of a subsequence on time series. The details of this step will be explained in Sect. 4.2. After that, a coverage matrix is produced and a new measure, named *CR*, is employed to evaluate the discriminative power of each subsequence candidate. In step 5, a heuristic algorithm based on MI tries to find the optimal combination of subsequences. In the last step, a set of discriminative subsequences is used to produce a transformed dataset for classifier training.

The pseudo code of the proposed algorithm is given in Algorithm 1. In lines 2 to 4, a set of subsequences are randomly sampled from time series dataset D according to the minimum length of subsequence *minLen*, the maximum length of subsequence *maxLen*. The sampling stops when the sampling time is higher than the parameter *samplingTime*. In Lines 5–7, it calculates the distance between each subsequences and time series. The distance is stored in a matrix M. In line 8, a K-means algorithm is performed on the distance matrix and figures out the coverage matrix $\mathbb{C}$. From lines 9–10, it calculates the CR for each subsequence based on the coverage matrix. The algorithm employs a MI-based heuristic algorithm to find the optimal combination of subsequences S' in Line 11. The algorithm transforms the time series data set D to a new data set D' based on S' in Line 12 and trains the classifier Ψ on D' in Line 13. More details of selection of subsequences will be elaborated in Sects. 4.2 and 4.3.

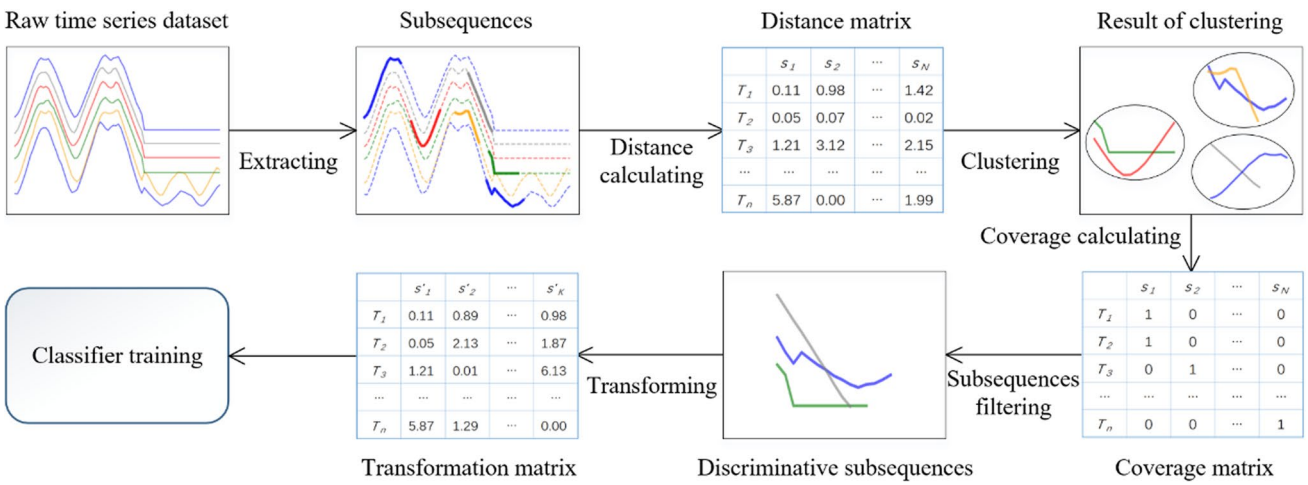

Fig. 1 Framework of the CRMI algorithm

Algorithm 1: Training a time series classifier based on Coverage Ratio and Mutual Information

Input: D : a training dataset of time series, Y : a label set, $samplingTime$: time limitation of sampling subsequences, $minLen$: the minimal length of discriminative subsequences, $maxLen$: the maximal length of discriminative subsequences

Output: Ψ : a classifier for time series classification

$S \leftarrow \varnothing$;
while $getRunTime() < samplingTime$ **do**
  $s \leftarrow randExtractSubsequence(D, minLen, maxLen)$;
  $S \leftarrow S \cup s$;
for each T_i **in** D **do**
  for each s_j **in** S **do**
    $M_{i,j} = d(s_j, T_i)$;
$\mathbb{C} \leftarrow Kmeans(M)$;
for each s_j **in** S **do**
  $s_j.cr = CR(s_j, \mathbb{C})$;
$S' \leftarrow selectSubsequencesByMutualInformation(S, \mathbb{C}, \boldsymbol{Y})$; // see Algorithm 2
$D' \leftarrow transformTimeSeries(D, S')$;
$\Psi \leftarrow trainClassifier(D')$;
return Ψ;

4.2 A Measure for Estimating Discriminative Power of Subsequences

IG is a popular measure for evaluating the discriminative power of subsequences at present. However, IG has several shortcomings, including (1) it is easy to be disturbed by the noise; (2) the computation of IG is time-consuming. To overcome these shortcomings, a new measure named CR is proposed to estimate the discriminative power of subsequences.

Given a subsequence s_j sampled from time series T_j and $D = \langle T, Y \rangle$ is a time series dataset with Q distinct labels. At first, the method calculates each distance $d(s_j, T_i)$, $1 \le i \le n$. Obviously, $d(s_j, T_i) = 0$ if $label(s_j) = label(T_i)$, otherwise $d(s_j, T_i) \ge 0$. After that, the distance values are normalized by min–max strategy and a clustering algorithm e.g., K-Means, is exploited to group the distances into Q categories.

Since a small distance value means that there exists a subsequence of time series whose shape is similar to the target subsequence, we construct a vector in which the j-th value will be set to 1 if $d(s_j, T_i)$ falls into the clustering set which contains the distance value 0, and otherwise, the j-th value of vector will be set to 0. In other words, the cover relation between a subsequence and a time series is revealed by the clustering of their distance, which is the main difference between our method and the existing works. Finally, the vectors of all subsequences form a Boolean matrix, called *coverage matrix*.

The advantages of our method involve: (1) it is a parameter-free technique that does not need to set any parameters, which is different from some works that require a distance threshold to determine whether a time series contains or does not contain a subsequence; (2) it does not require any transformation on the raw time series, e.g., SAX or SFA, which may lose some information and increase computing overhead. Moreover, the number of categories for clustering

is naturally set to Q which is the category size of a time series dataset. A toy example is given below.

Example 1 Assume a time series dataset that contains 5 time series T_1, T_2, T_3, T_4, T_5 and has 3 distinct labels. Let s be a subsequence extracted from T_1. The distance between s and the 5 time series forms a vector [0.00, 0.56, 0.37, 0.62, 0.08], e.g., $d(T_5, s) = 0.08$. Then, we perform the K-Means algorithm on the distance vector where $K = 3$ (which is same with the number of the distinct labels) and obtain the clustering result {{0.00, 0.08}, {0.37}, {0.56, 0.62}}. Based on the result, the time series are divided into three groups: {T_1 ,T_5}, {T_3} and {T_2,T_4}. It can be seen that s is taken from T_1 and the group containing T_1 also contains another time series T_5, therefore the values correspondent T_1 and T_5 in a coverage vector are set to 1 and the others are set to 0, that is $[1, 0, 0, 0, 1]^T$. It indicates that the time series T_1, T_5 are covered by the subsequence s. In another word, there exist some subsequences in T_1, T_5 similar to s in shape.

Based on the coverage matrix, this paper proposed an innovative measure CR, for estimating the discriminative power of a subsequence. The definition of CR is given below.

Definition 3 (Coverage ratio, CR): Given a coverage matrix $\mathbb{C}$ and a subsequence s_j with the label $label(s_j)$, the CR of s_j, can be calculated by the following formula.

$$CR(s_j, \mathbb{C}) = \frac{\#(c_{i,j} = 1 \wedge label(T_i) = label(s_j))}{\#(label(T_i) = label(s_j))} - \lambda \cdot \frac{\#(c_{i,j} = 1 \wedge label(T_i) \neq label(s_j))}{\#(label(T_i) \neq label(s_j))} \tag{2}$$

where T_i and s_j represent the i-th time series and the j-th subsequence, respectively, and $label(T_i)$ and $label(s_j)$ represent the labels of T_i and s_j, respectively. In the coverage matrix $\mathbb{C}$, $c_{i,j} = 1$ when the j-th subsequence covers the i-th time series. The symbol #() is a cardinal operator, and λ is a parameter to control the weight of two components. The range of λ is $(0, 1]$. It is not difficult to figure out that the range of $CR(s_j, \mathbb{C})$ is $[-1, 1]$. The rationale behind the CR is simple that we hope a subsequence s_j with great discriminative power covers as many time series with the label $label(s_j)$ as possible and covers as few time series with other labels as possible. A toy example for explanation is shown below.

Example 2 There are 12 time series and $Y = \{1, 2, 3\}$ in the time series dataset (as shown in Table 1). Two subsequences s_1, s_2 are extracted from time series with the label *1*. The coverage vectors of the two subsequences are shown in the 3rd and the 4th row of Table 1, respectively. The parameter λ is set to 0.5. We can calculate that the CR of s_1 is 0.6875 and the CR of s_2 is 0.3125 based on Formula (2).

It demonstrates that s_1 is more discriminative than s_2. Let us analyze the coverage vector of both subsequences, it is easy to find that most of the time series covered by s_1 belong to the same category (i.e., the first category), and in contrast, the time series covered by s_2 are distributed across three categories. It demonstrates that the new measure, i.e., *CR*, can reveal the discriminative power of subsequences.

4.3 A MI-Based Algorithm for Discriminative Subsequence Selection

Although a new measure has been proposed to evaluate the discriminative power of subsequences, selecting a set of subsequences that can maximally represent the time series is still an issue. Some works simply choose the top-k subsequences with the highest score for classifier training. However, it cannot work well on some the datasets because it does not consider the correlation among the subsequences.

MI is an important tool in information theory that can handle the correlation between two variables. Moreover, MI-based feature selection has been proven to be effective in many works. In this section, an MI-based algorithm is proposed to handle the problem of discriminative subsequence selection. First, the definition of MI is given below.

Definition 4 (Mutual information, MI) Given a coverage matrix $\mathbb{C}$ of a subsequence set $S = \{s_1, s_2, \cdots, s_{n_s}\}$ on time series dataset $D = \langle T, Y \rangle$, the MI between S and Y is denoted as follows.

$$MI(S;Y) = \sum_{y \in Y} \sum_{s \in S} p(s, y) log_2 \left(\frac{p(s, y)}{p(s)p(y)} \right) = H(Y) - H(Y|S) \tag{3}$$

where $p(s, y)$ is the joint probability distribution function of s and y in $\mathbb{C}$, while $p(s)$ and $p(y)$ is the probability distribution function of s and y in $\mathbb{C}$, respectively. $H(Y)$ is the information entropy of Y, and $H(Y|S)$ is the conditional entropy of Y with

Table 1 Illustration for coverage ratio

Time series	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
Label	1	1	1	1	2	2	2	2	3	3	3	3
$\mathbb{C}_1$	1	0	1	1	0	1	0	0	0	0	0	0
$\mathbb{C}_2$	0	1	1	0	0	0	1	1	0	1	0	0

respect to S. Similar to the existing work, the algorithm aims to find a subset S' of S such that $MI(S;Y)$ is equal to $MI(S';Y)$.

To find the optimal combination of subsequences, we first group the subsequences into Q groups by their labels, and sort them by their CR values in descending order within the groups. Then, the subsequences with the highest CR in each group are chosen to form a *core* set. The core set is regarded as the starting set of the subsequence selection. Next, a round-robin strategy is employed that successively selects a subsequence from a group and adds it to S' until $MI(S';Y)$ no longer increases. The process of the MI-based subsequence selection is shown in Fig. 2.

The pseudo-code of the algorithm is shown in Algorithm 2. In the first line, the algorithm groups all the subsequences by their labels and sorts them by CR score in descending order within each group. In Line 2, the result set is set to be empty. From Lines 3 to 4, a core set is built using the first subsequence (i.e., the subsequences with the highest CR score in each group) in each group, where the function *group.getAndRemove*(0) removes the first element from the group and returns the element. The MI of the core set is calculated in Line 5 and assigned to a variable *lastMI* in Line 6. From Lines 8 to 19, a round-robin strategy is employed to traverse all groups to find the best subsequence in each group which can achieve the maximum MI value. The loop ends when the MI value does not increase (Lines 20–23). At the end of the algorithm, the algorithm returns the final subsequences S'.

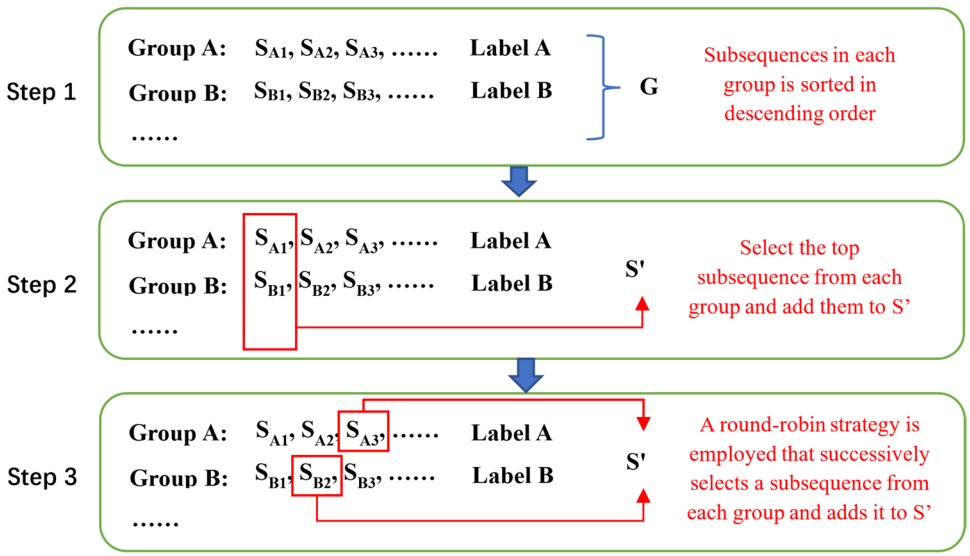

Fig. 2 Process of the MI-based subsequence selection

Algorithm 2: Select Subsequences By MutualInformation (S,£,Y)

Input: S : a subsequences set; $\mathbb{C}$: the coverage matrix; Y : label set of time series dataset
Output: S' : a final subsequences set

```
1  groups ← groupByLabelAndDescendByCR(S) ;        // Step 1
2  S' ← {} ;
3  for each group in groups do                     // Step 2: From lines 3 to 4
4      S' ← S' ⋃ group.getAndRemove(0) ;
5  bestMI ← MI(S';Y) ;
6  lastMI ← bestMI ;
7  while true do
8      for each group in groups do                 // Step 3: From lines 8 to 19
9          bestGrpMI ← 0 ;
10         bestS ← NULL ;
11         for each s in group do
12             S'' ← S' ⋃ s ;
13             mi ← MI(S'';Y) ;
14             if mi > bestGrpMI do
15                 bestGrpMI ← mi ;
16                 bestS ← s ;
17         if bestGrpMI > bestMI then
18             bestMI ← bestGrpMI ;
19             S' ← S' ⋃ group.getAndRemove(bestS) ;
20     if bestMI > lastMI then
21         lastMI ← bestMI ;
22     else
23         break;
24 return S' ;
```

4.4 Analysis of Time Complexity

The proposed algorithm is composed of five steps which involve random sampling time series subsequences, calculating the distances between each subsequence and time series, clustering based on the distance matrix, calculating the CR for each subsequence, and finding the optimal subset of subsequences. The time of random sampling is a parameter and we will discuss it in the experiments. The worst time complexity of the second step is $O(m \cdot n \cdot n_S)$ where m is the length of the time series, n and n_S represent the number of time series and the number of subsequences to estimate, respectively. The worst time complexity of the third step, e.g., K-means clustering, is $O(Q \cdot n \cdot n_S)$, Q is the number of distinct classes. The key operation of the fourth step is counting the time of coverage and its worst time complexity is $O(n \cdot n_s)$. The final step requires calculating the MI for many times and its worst time complexity is $O(n \times Q \times (Q + (Q+1) + (Q+2) + \cdots + n_S)) \approx O(n \cdot Q \cdot n_s^2)$. Since $Q \cdot n_s$ is obviously higher than the length of the time series in most cases, the worst time complexity of the proposed algorithm is $O(n \cdot Q \cdot n_s^2)$.

5 Experiments

5.1 Experiment Setup

In this section, we perform extensive experiments to evaluate the performance of the proposed algorithm. First, the datasets will be introduced in this paragraph. Second, we explore

Table 2 Descriptions of datasets

ID	Name	#Train	#Test	#Class	Length	ID	Name	#Train	#Test	#Class	Length
1	Adiac	390	391	37	176	28	MedicalImages	381	760	10	99
2	ArrowHead	36	175	3	251	29	MiddlePhalanxO	400	154	3	80
3	Beef	30	30	5	470	30	MiddlePhalanxTW	399	154	6	80
4	Car	60	60	4	577	31	NonInvasiveFetal1	1800	1965	42	750
5	CBF	30	900	3	128	32	NonInvasiveFetal2	1800	1965	42	750
6	ChlorineCon	467	3840	3	166	33	OliveOil	30	30	4	570
7	CinCECGTorso	40	1380	4	1639	34	OSULeaf	200	242	6	427
8	CricketX	390	390	12	300	35	Phoneme	214	1896	39	1024
9	CricketY	390	390	12	300	36	Plane	105	105	7	144
10	CricketZ	390	390	12	300	37	ProximalPhalanxO	400	205	3	80
11	DiatomSizeR	16	306	4	345	38	ProximalPhalanxT	400	205	6	80
12	DistalPhalanxOAG	400	139	3	80	39	RefrigerationD	375	375	3	720
13	DistalPhalanxTW	400	139	6	80	40	ScreenType	375	375	3	720
14	ECG5000	500	4500	5	140	41	ShapesAll	600	600	60	512
15	ElectricDevices	8926	7711	7	96	42	SmallKitchen	375	375	3	720
16	FaceAll	560	1690	14	131	43	StarLightCurves	1000	8236	3	1024
17	FaceFour	24	88	4	350	44	SwedishLeaf	500	625	15	128
18	FacesUCR	200	2050	14	131	45	Symbols	25	995	6	398
19	FiftyWords	450	455	50	270	46	SyntheticControl	300	300	6	60
20	Fish	175	175	7	463	47	Trace	100	100	4	275
21	Haptics	155	308	5	1092	48	TwoPatterns	1000	4000	4	128
22	InlineSkate	100	550	7	1882	49	UWaveGestureA	896	3582	8	945
23	InsectWingBS	220	1980	11	256	50	UWaveGestureX	896	3582	8	315
24	LargeKitchenA	375	375	3	720	51	UWaveGestureY	896	3582	8	315
25	Lightning7	70	73	7	319	52	UWaveGestureZ	896	3582	8	315
26	Mallat	55	2345	8	1024	53	WordSynonyms	267	638	25	270
27	Meat	60	60	3	448	54	Worms	181	77	5	900

the parameter setting strategies. Third, ablation experiments are performed to evaluate the effectiveness of the methods in CRMI, including the CR measure and the MI-based heuristic algorithm for discriminative subsequence selection. Finally, the performance of the proposed algorithm is evaluated by comparing it with six shapelet-based TSC algorithms as well as the 1NN + DTW algorithm which is claimed to be difficult to defeat.

The experiments were carried out on a server equipped with Intel Xeon Gold 5215 CPU (2.5 GHz) and 64 GB memory. The algorithm is coded in Java with toolkits Weka 3.4.3 and Time Series Machine Learning (TSML) [27]. Since the proposed method is for MC-TSC, 54 datasets that have at least three categories are selected from the UCR repository. The details of the datasets are shown in Table 2. The last four columns in the table are the number of training instances, the number of testing instances, the number of categories, and the length of the time series.

5.2 Influence of Parameter Settings

In this section, we conduct experiments to study how to set the parameters in the CRMI algorithm, including the time of subsequence sampling and the parameter λ in Formula (2).

First, we investigate the impact of the time of subsequence sampling by setting its value from 2 to 5 min with a step value of 1 min. The value of λ is set to 1.0. For fairness, the experiments were performed 50 times on each dataset because the algorithm is stochastic. The accuracy of classification, as well as the training time and the testing time on each dataset, are recorded. For simplicity, we calculate the average value of the classification accuracy, the training time, and the testing time on all datasets. The results are shown in Fig. 3, in which the abscissa is the time of sampling and the left ordinate is the average accuracy of classification on all datasets and the right ordinate is the time cost of the algorithm. From the figure, it is obvious that

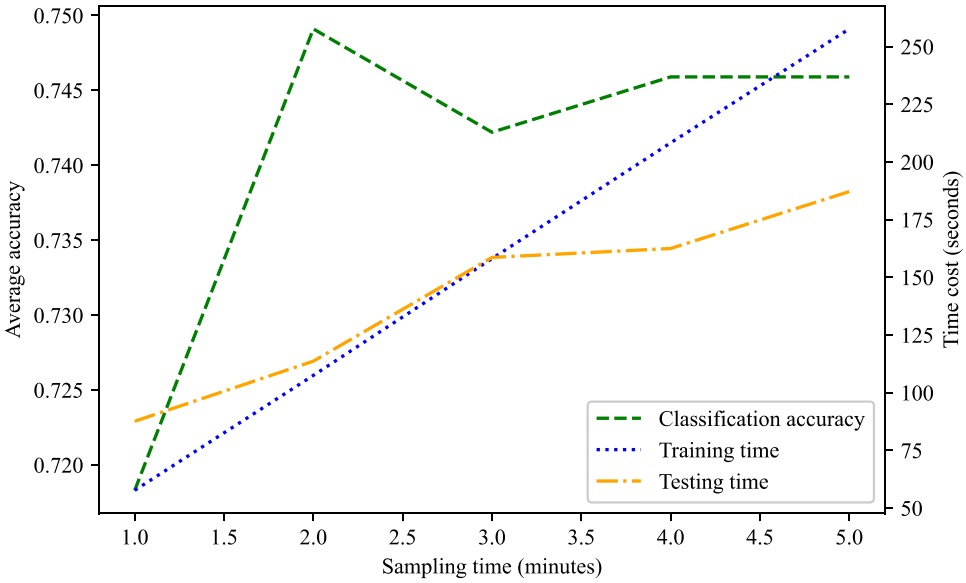

Fig. 3 Influence of sampling time setting

the classification accuracy achieves the highest value, i.e., 74.91%, when the sampling time is 2 min. The second highest classification accuracy is 74.59% which is achieved when the sampling time is set to 4 and 6, respectively. Obviously, 2 min is a good choice for the sampling time.

Next, we conduct experiments to investigate the influence of the setting of λ. The value of λ is set from 0.2 to 1.0 with a step value of 0.2. The sampling time is set to 2 min. Similarly, the experiments were performed 50 times on each dataset, and the average value of classification accuracy is calculated. The results are shown in Fig. 4. As can be seen from the figure, the highest classification accuracy is 74.91% when the λ is set to 1.0, and the lowest classification accuracy is 73.70% when the λ is set to 0.8. Therefore, the value of λ in the experiment is set to 1.0. We also change the order of the parameter selection, i.e., first determine the setting of λ and then determine the sampling time, the results show that it will not influence the performance of the algorithm.

5.3 Comparison of Different Strategies in MI-Based Discriminative Subsequence Discovery

In Algorithm 2, an MI-based algorithm for discovering the discriminative subsequences is proposed in which two strategies are employed including a *core* set consisting of the subsequence with the highest CR value in each group and a *round-robin* selection strategy. In this section, we evaluate three different strategies.

(1) Core + Round-Robin.

This strategy is employed in the proposed algorithm. After the calculation of the CR value of all subsequences, the algorithm groups the subsequences according to their labels. Moreover, the subsequences in each group are ranked in descending order according to the CR value. Then, it constructs a core set that consists of subsequences with the highest CR value in each group. Finally, it successively selects a subsequence with the highest CR value in each group and adds it to the set. The algorithm ends until the value of MI does not grow.

(2) Core + Ranking.

The difference between this strategy and Core + Round-Robin is that the former does not successively select a subsequence from each group, but from a collection of all subsequences. After the construction of the core set, this strategy ranks all the subsequences according to their CR values in descending order and successively selects a subsequence with the highest CR value. The algorithm ends until the value of MI does not grow.

(3) Ranking.

This strategy sorts all subsequences in descending order according to their CR values and then adds them to the result set one by one. It calculates the MI of the selected subsequences and the algorithm ends until the MI no longer increases.

In the experiments, the sampling time is set to 2 min and the value of λ is set to 1.0. Similarly, each strategy was performed 50 times on each dataset, and the average value of classification accuracy as well as the training time was calculated. The experimental results are shown in Fig. 5. It is easy to find that the Core + Ranking strategy performs slightly better than the Ranking strategy. It demonstrates that the core set is helpful to improve classification accuracy. Furthermore, the Core + Round-Robin strategy performs much better than the Core + Ranking strategy in

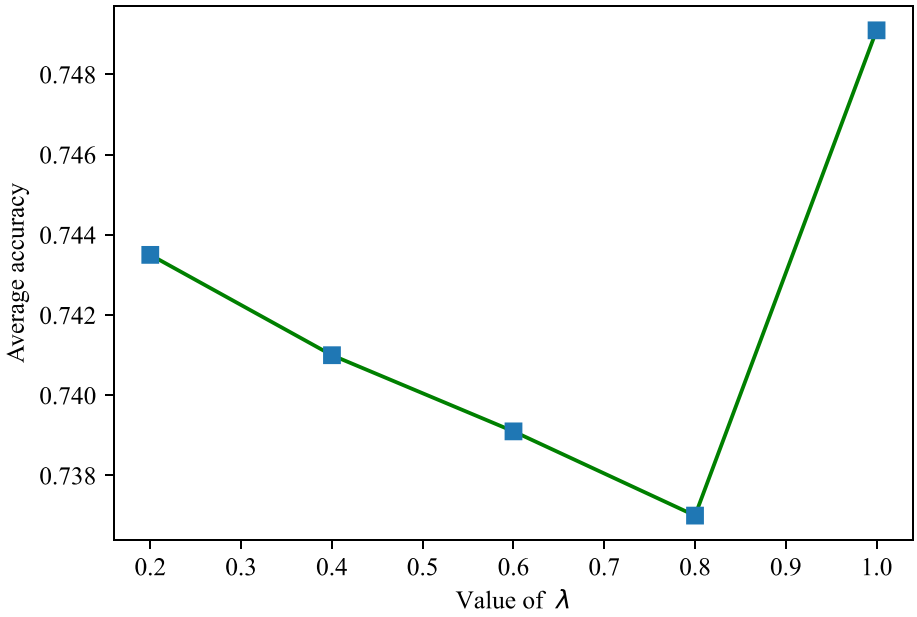

Fig. 4 Average classification accuracy achieved by different values of λ

terms of classification accuracy. It proves the efficiency of the Round-Robin strategy. However, the training time of the Core + Round-Robin strategy is slightly longer than the Core + Ranking strategy. The only reason behind the phenomenon is that the former selects more subsequences than the latter because the number of subsequences from each group is balanced under the Core + Round-Robin strategy. The experimental results prove that balancing the number of subsequences (features) from different categories of time series is helpful for TSC.

5.4 Comparison Against the State-of -the-Art Algorithms

We compared the CRMI algorithm with seven TSC classifiers, including 1NN + DTW and six state-of-art algorithms based on shapelets, which are ST [18], gRSF [19], CRSF [20], BSPCover [22], Fast Shapelet (FS) [25], and SD [26]. The reasons that we only select shapelet-based algorithms for comparison are twofold. On the one hand, existing works have shown that state-of-the-art shapelet-based algorithms perform much better than most dictionary-based algorithms. On the other hand, the proposed algorithm is similar to the shapelet-based algorithm in that all of them work on the raw time series data. This is different from the dictionary-based methods which need to convert the raw time series data by SAX or SFA etc. The parameters of algorithms for comparison were set according to the corresponding references. If datasets do not appear in the references, we use the default parameters in the open-source code.

The experiments were conducted on 54 datasets. All the stochastic algorithms were run 50 times on each dataset,

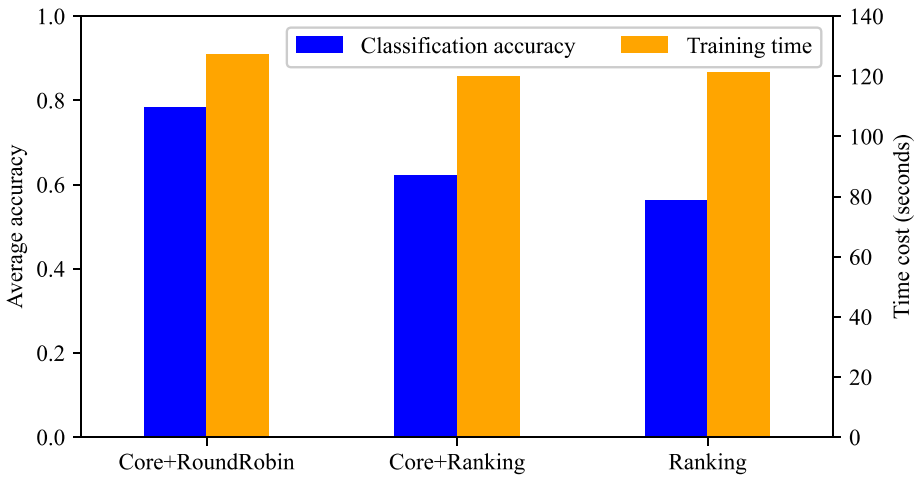

Fig. 5 Comparison of average accuracy and training time on different selection strategy

Table 3 Accuracy comparison of 8 classifiers on 54 datasets

ID	1NN + DTW	FS	SD	ST	BSPCover	gRSF	CRSF	CRMI
1	0.7829	0.6000	0.6990	0.7714	**0.8000**	0.7697	0.7169	0.7126
2	0.9911	0.9578	0.9807	**0.9933**	0.8627	0.9919	0.9927	0.9767
3	0.6469	0.5828	0.5377	0.7046	0.5326	0.6514	0.6249	**0.7073**
4	0.7266	0.6691	0.6667	0.7770	0.4676	**0.8172**	0.7363	0.7914
5	0.7893	0.3333	0.6517	0.8160	0.8560	**0.8643**	0.8229	0.8230
6	**1.0000**	0.7833	0.8800	0.9500	0.7167	0.9567	0.9392	0.9667
7	0.5844	0.5325	0.5182	0.5844	0.4883	**0.7078**	0.6347	0.5909
8	0.8049	0.8244	0.7668	0.8342	0.4878	0.8420	0.8463	**0.8488**
9	0.4640	0.5106	0.5104	0.5440	**0.8182**	0.6883	0.5639	0.5573
10	0.3973	0.4213	0.3701	**0.6760**	0.4271	0.6475	0.4517	0.4759
11	0.6427	0.3333	0.6347	0.8020	0.6432	**0.8224**	0.7123	**0.8224**
12	0.9600	0.9102	0.9341	0.9770	0.8400	**0.9793**	0.9576	0.9757
13	0.7167	0.7000	0.6556	**0.9020**	0.6167	0.8167	0.7175	0.8833
14	0.6812	0.7123	0.7778	0.9180	0.4457	0.8919	0.6953	**0.9717**
15	**0.9804**	0.8856	0.9139	0.8791	0.7745	0.8654	0.9109	0.8725
16	0.8296	0.9205	0.8045	0.7614	0.8182	0.8955	0.9659	**0.9773**
17	0.8333	0.7333	0.7733	0.8810	0.8621	0.8600	0.8333	**0.9333**
18	**1.0000**	**1.0000**	0.9580	**1.0000**	0.8600	**1.0000**	**1.0000**	**1.0000**
19	**1.0000**	0.2588	0.9813	0.9520	0.9512	0.9959	0.9833	0.9515
20	0.4333	0.5000	0.4444	0.7333	0.5667	0.5467	0.5683	**0.8000**
21	0.9293	0.9042	0.9129	**0.9456**	0.9257	0.9405	0.9316	0.9440
22	0.4156	0.3506	0.3532	**0.5162**	0.3896	0.4721	0.4705	**0.5162**
23	0.5844	0.5325	0.5377	0.7190	0.4796	**0.7299**	0.6273	0.6753
24	0.6043	0.6331	0.6091	0.6900	0.3022	**0.6950**	0.6921	0.6906
25	0.4870	0.5000	0.4610	0.5488	0.2727	0.5727	**0.6019**	0.5648
26	0.5909	0.6983	0.5397	**0.9546**	0.5866	0.8624	0.6855	0.8636
27	0.7561	0.7171	0.7141	0.8030	0.7574	0.7961	0.7966	**0.8098**
28	0.9498	0.9246	0.8563	**0.9648**	0.9166	0.8341	0.9475	0.9487
29	**0.9933**	0.9233	0.9760	0.9832	0.9754	0.9930	0.9908	0.9833
30	0.7544	0.5488	0.5928	**0.8950**	0.6852	0.8773	0.7279	0.5111
31	0.7829	0.8171	0.7291	0.9547	0.8215	0.9092	0.8637	**0.9657**
32	0.3836	0.2946	0.3982	**0.4455**	0.3834	0.4240	0.3892	0.4092
33	0.6849	0.5343	0.6247	0.7397	0.5890	0.7123	**0.7459**	0.7260
34	**1.0000**	0.9810	0.9848	**1.0000**	0.9520	0.9962	0.9881	**1.0000**
35	0.9348	0.9582	0.9308	0.9395	0.9346	0.9339	0.9709	**0.9889**
36	**0.9630**	0.9021	0.9078	0.8060	0.8025	0.9526	0.8996	0.9389
37	0.7275	0.6963	0.7603	0.7370	0.6963	**0.8106**	0.7799	0.7457
38	0.6340	0.6108	0.6709	**0.8060**	0.7012	0.7360	0.7025	0.6778
39	0.6583	0.6374	0.6754	**0.9420**	0.7542	0.7548	0.7348	0.7044
40	**0.7211**	0.6408	0.6595	0.6910	0.5145	0.6974	0.6497	0.7026
41	0.3576	0.4904	0.4947	0.4382	0.5697	0.6301	0.5779	**0.6348**
42	0.7590	0.5154	0.6829	0.7639	0.5544	0.7623	0.6753	**0.7641**
43	0.6974	0.4821	0.6590	0.7495	0.7254	0.7495	0.6855	**0.7531**
44	**0.8000**	0.4103	0.6897	0.7980	0.5026	0.7982	0.7142	0.7900
45	**0.9456**	0.6237	0.7125	0.7485	0.7324	0.9387	0.7349	0.7420
46	0.8859	0.6737	0.8487	**0.9117**	0.7025	0.8928	0.8383	0.8805
47	0.7920	0.7696	0.8614	0.9239	0.9239	0.9037	0.8737	**0.9248**
48	**0.6489**	0.4671	0.6201	0.5820	0.5412	0.5978	0.5679	0.6442
49	0.6215	0.5601	0.5865	0.7680	0.7681	0.7235	0.4737	**0.7951**
50	0.2284	0.2214	0.1459	**0.3402**	0.2275	0.2970	0.2275	0.3012
51	0.7903	0.7984	0.8084	**0.9470**	0.8354	0.9091	0.7645	0.8372

Table 3 (continued)

ID	1NN + DTW	FS	SD	ST	BSPCover	gRSF	CRSF	CRMI
52	0.8646	0.8345	0.8560	**0.9540**	0.8554	0.9331	0.7982	0.8651
53	0.6615	0.5055	0.6923	0.7130	0.7142	0.7059	0.6388	**0.7341**
54	0.7683	0.6100	0.7860	**0.8540**	0.7426	0.8405	0.8536	0.7467
Wins	11	1	0	18	2	9	3	**19**
Ave. Rank	4.50	6.61	6.06	2.74	5.94	2.81	4.15	**2.70**
1v1 Wins	37	51	48	28	48	27	38	–
1v1 Draws	2	1	0	3	0	2	1	–
1v1 Loses	15	2	6	23	6	25	15	–

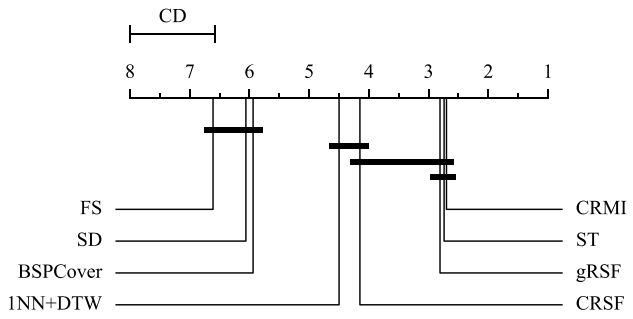

Fig. 6 Nemenyi tests for 8 classifiers ($p = 0.05$)

including FS, SD, BSPCover, gRSF, and CRSF. The experimental results are shown in Table 3 in which the average accuracy of classification on 54 datasets for each algorithm is listed. The highest accuracy for each dataset is marked in bold. To describe the experimental results more intuitively, some comparisons are provided in the last five rows of the table. The "Wins" row shows the number of datasets on which the corresponding algorithm won the gold medal. The second row shows the average ranking of the algorithm on 54 datasets. The last three rows show the number of datasets that CRMI wins, draws, and loses across all datasets compared with other algorithms, respectively. For example, compared with CRSF, CRMI won 38 times, drew 1 time, and lost 15 times on 54 datasets. As shown in Table 3, the CRMI algorithm achieves the best results on 19 datasets which is slightly better than the ST algorithm which won on 18 datasets. The number of datasets won on by other algorithms is far less than the proposed algorithm. In terms of the average ranking, the top three algorithms are CRMI, ST, and gRSF which get 2.70, 2.74, and 2.81, respectively. Comparing CRMI with other algorithms one-to-one, it can also be found that the CRMI algorithm outperforms other algorithms.

We also exploit the Nemenyi test to detect whether there exist significant differences among the eight algorithms. A critical difference diagram is shown in Fig. 6. Classifiers that are not significantly different at p = 0.05 are connected. It can be easily found that CRMI is significantly superior to the five algorithms, which are 1NN + DTW, SD, FS, BSPCover, and CRSF. Since the difference among CRMI, ST, and gRSF is not significant, the empirical findings indicate that when aiming for the highest accuracy, any one of them can be safely recommended. It is known to us that the ST algorithm is too time-consuming because it needs to evaluate all the subsequences in a time series dataset. Based on the prior analysis, we can conclude that the accuracy achieved by CRMI is slightly better than gRSF and ST, and obviously better than the rest methods. It proves the effectiveness of the proposed algorithm.

5.5 Cross Validation

To further analyze the effectiveness of the CRMI algorithm, fivefold cross-validation experiments were conducted on all 54 datasets. Specifically, the training set data and testing set data in the UCR archive were merged, and the data with the same label was divided into five groups. One group was designated as the validation set, while the remaining four groups were served as the training set. We record the accuracy in the cross-validation experiments and figured out the average score. Same with the previous experiments, some algorithms which are stochastic were repeated 50 times on each dataset to ensure the fairness of results. Pairwise comparisons were performed on these experimental results which illustrated in Fig. 7. Each subplot represents the comparative analysis between the CRMI algorithm and another algorithm, where the y-axis represents the average accuracy of CRMI, and the x-axis represents the average accuracy of the compared algorithm. Each point represents a dataset. If a point falls above the diagonal, it suggests that the CRMI algorithm performs better than the compared algorithm. Conversely, the CRMI algorithm is inferior to the compared algorithm.

From figures (a), (b), (c), (e), and (g), it is obvious that the CRMI algorithm are superior to the 1NN-DTW, FS, SD, BSPCover, and CRSF algorithms because most of the points fall on the upper side of the diagonal in those figures. In Figures (d) and (f), most points are near the diagonal,

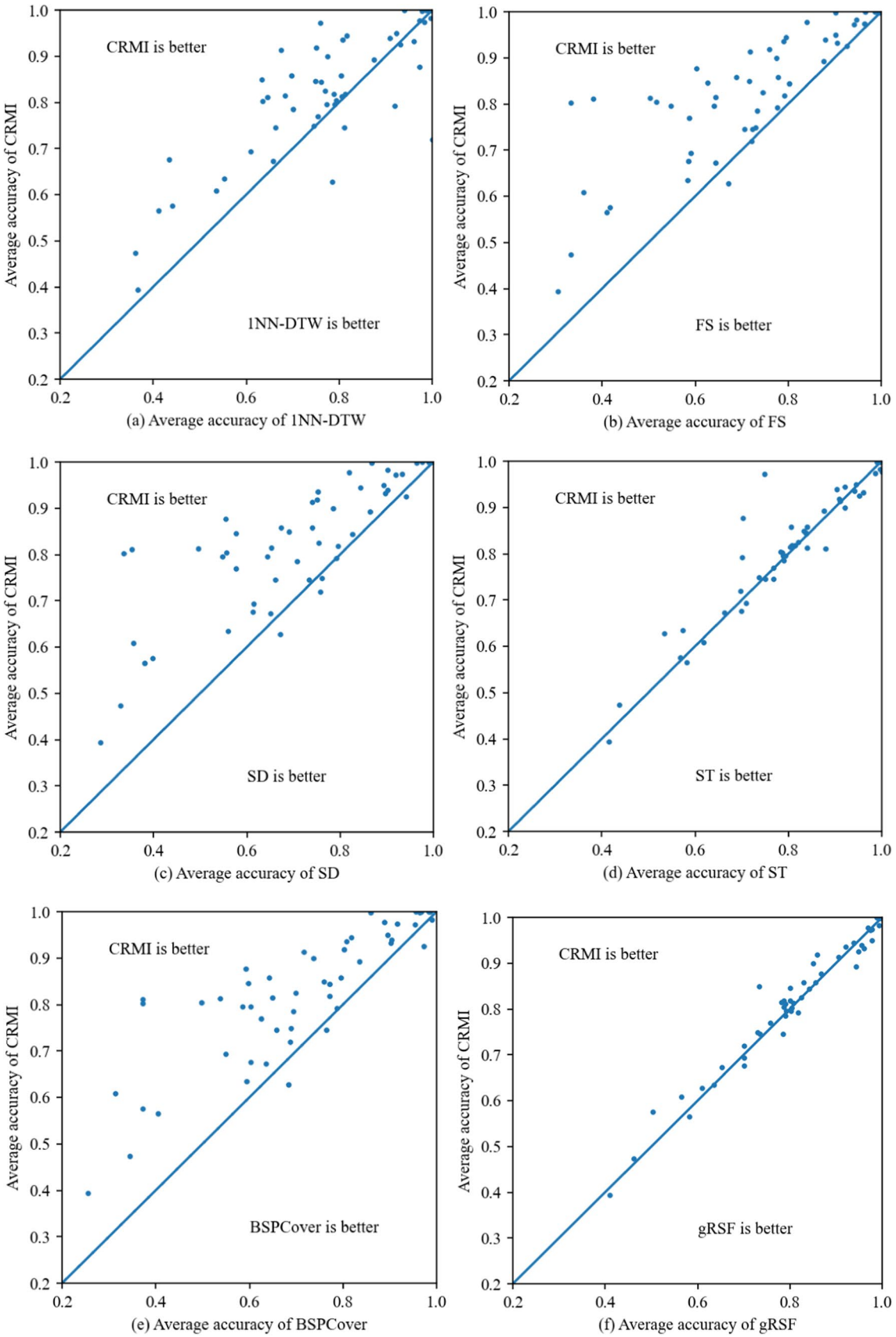

Fig. 7 Results of cross validation on 54 datasets

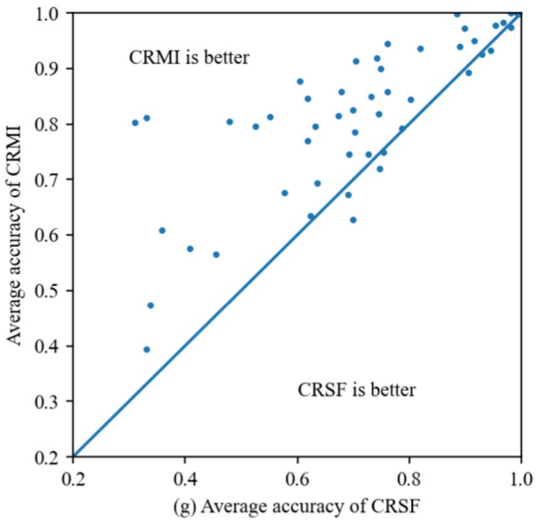

Fig. 7 (continued)

however it can be also found that the CRMI algorithm is also slightly superior to the two algorithms, i.e., the ST algorithm and the gRSF algorithm. The results of the cross-validation also prove the effectiveness of the proposed algorithm.

5.6 Analysis of the Impact of the Number of Categories

In this section, we investigate the impact of the number of categories in a dataset on CRMI in terms of classification accuracy. In terms of the distribution of category numbers, the datasets are categorized into three groups. The dataset category numbers in 3 groups are no more than 6, no less than 7, and no less than 10, respectively.

The experimental results are re-analyzed as listed in Table 4. From the table, it can be seen that the number of datasets won by the CRMI algorithm is 9 which is equal to that won by the ST algorithm when the number of categories is no more than 6. Although the CRMI algorithm is tied with the ST algorithm for first place in this metric, the average ranking of the former is 2.83 which is inferior to that of the ST as well as the gRSF which is 2.48. When the category number is no less than 7, it can be seen that the CRMI algorithm wins the gold medals on two metrics, i.e., Ave. Rank and Wins, which are 2.56 and 10, respectively. The second place is the ST algorithm which can achieve 9 and 3.04 on the two metrics, respectively. Furthermore, it can be seen that the CRMI algorithm obtains 2.60 and 7 on the two metrics when the category number is no less than 10. The second place of the two metrics is 3.13 and 4 which are obtained by the gRSF and the ST, respectively. Although the proposed algorithm achieves first place in both groups, it is easy to find that the advantage of the CRMI algorithm is apparent along with the increase in the category number of the dataset.

The analyses show that CRMI, ST, and gRSF have good classification ability for multi-class time series classification. However, both the ST and the gRSF require calculating the IG to evaluate the discriminative ability of the shapelet. Moreover, the former will evaluate all shapelet candidates. Therefore, both of them are too time-consuming. For example, the time costs of the ST and the gRSF are 6752 s and 1872s on dataset NonInvasiveFetalECGThorax1, respectively, which is far higher than that of the CRMI which is 129 s. The experiments prove the effectiveness and efficiency of the proposed algorithm.

Table 4 Experimental results of different number of categories in datasets

		1NN + DTW	FS	SD	ST	BSPCover	gRSF	CRSF	CRMI
$Q \leq 6$	Ave. rank	4.34	7.24	5.93	**2.48**	6.28	**2.48**	4.10	2.83
	Wins	6	0	0	**9**	0	7	1	**9**
	1v1 Wins	19	28	25	17	25	14	19	–
	1v1 Draws	1	0	0	1	0	1	0	–
	1v1 Loses	9	1	4	11	4	14	10	–
$Q \geq 7$	Ave. rank	4.68	5.88	6.20	3.04	5.56	3.20	4.20	**2.56**
	Wins	5	1	0	9	2	2	2	**10**
	1v1 Wins	18	23	23	11	23	13	19	–
	1v1 Draws	1	1	0	2	0	1	1	–
	1v1 Loses	6	1	2	12	2	11	5	–
$Q \geq 10$	Ave. rank	4.67	5.87	6.40	3.33	5.40	3.13	3.60	**2.60**
	Wins	3	1	0	4	2	2	1	**7**
	1v1 Wins	10	14	14	7	13	7	11	–
	1v1 Draws	1	1	0	1	0	1	1	–
	1v1 Loses	4	0	1	7	2	7	3	–

5.7 Other Advantages

The experiments show the effectiveness and efficiency of the CRMI algorithm, especially in multi-class classification tasks. Furthermore, the ease of use of the proposed algorithm is superior to most of the pattern-based algorithms of TSC because it is nearly a parameter-free algorithm. It is easy to find that the CRMI algorithm only requires determining the minimum and maximum lengths of the subsequences. We always set the two parameters to three and the length of the time series, respectively. The existing pattern-based algorithms of TSC, such as CRSF, ST, etc., must set a distance threshold to determine whether a time series contains or does not contain a subsequence which may greatly decrease the accuracy of the classification. Besides, the CRMI algorithm does not require any transformation of the raw time series. Some work, such as CRSF, BSPCover, BOSS, etc., require the representation of the time series by SAX or SFA which may lose some information and introduce extra computing overhead. Finally, the CRMI algorithm employs an MI-based heuristic for looking for the optimal combination of subsequences. Feature selection has been studied for a long time and lots of efficient techniques can be exploited in our work, such as the meta-heuristic algorithms. It may further improve the effectiveness and efficiency of the proposed algorithm.

6 Conclusions

In this paper, we present a new algorithm named CRMI to find discriminative subsequences for MS-TSC. A new measure for evaluating the discriminative power of subsequences, called CR, is proposed which reveals the cover relation between a subsequence and time series via clustering their distances. Besides, an MI-based heuristic algorithm for looking for the optimal combination of subsequences is also presented. We perform extensive experiments on 54 datasets to evaluate the effectiveness and efficiency of the CRMI algorithm. Some conclusions can be drawn. (1) Compared with the 1NN + DTW classifier and six well-known shapelet-based classifiers, the CRMI algorithm can achieve the best average accuracy on 54 datasets. Although there is no significant difference in terms of the accuracy between CRMI and ST statistically, CRMI is much more efficient than the two algorithms. (2) Extensive experiments were conducted to evaluate the proposed strategies and explore the strategies of parameter setting. The experimental results show that the proposed strategies, i.e., employing a core set to be the start point of subsequence selection and the round-robin selection for subsequence selection, are effective. (3) The CRMI algorithm only requires setting two parameters, i.e., the time of subsequence sampling and the λ in Formula (2). Experimental results show that 2 min and 1.0 are good choices for the two parameters, respectively. (4) The CRMI algorithm performs well on time series datasets with a large number of categories.

However, this study still has limitations and faces several challenges: (1) the UCR datasets are balanced, while many real-life data are imbalanced. Therefore, the CRMI algorithm which is trained on balanced datasets needs to consider imbalanced data; (2) although it has been proven the effectiveness of the algorithm in this paper, it can still be enhanced by utilizing representations such as SAX or SFA, or employing parallel computing to improve its efficiency; (3) it is necessary to consider the unseen time series data in real-life to improve the generalization of the algorithm.

Acknowledgements The authors thank the reviewers for their work and the contributors of the UCR archive.

Author Contributions Siyuan Jing helped design the algorithm and he is a director of this work. Jun Yang contributed to the algorithm design and performed the experiments.

Funding This work is supported by the open project fund of Key Lab of Internet Natural Language Intelligent Processing of Sichuan Provincial Education Department (Grant No. INLP202304), the Project Fund of Sichuan Tourism Development Research Center (LY22-14), the Research and Cultivation Project of Leshan Normal University (No. KYPY2024-0002), the Research and Innovation Team Cultivation Project of Leshan Normal University (No. KYCXTD2023-9), the Ministry of Education Humanities and Social Sciences Planning Project (Grant No. 23YJA740013), open project fund of Intelligent Terminal Key Laboratory of Sichuan Province (Grant no. SCITLAB-1002).

Data Availability The datasets used to support the findings of this study are from the UCR Time Series Classification Archive which can be downloaded from https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

Declarations

Conflict of Interest All authors declare that there are no conflicts of interest.

References

1. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Disc. **31**(3), 606–660 (2017)
2. Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.: The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Disc. **35**(2), 401–449 (2021)
3. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.-A.: Deep learning for time series classification: a review. Data Min. Knowl. Disc. **33**(4), 917–963 (2019)
4. Wang Z, Yan W, Oates T (2017) Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp 1578–85
5. Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.-A., Petitjean, F.: InceptionTime: finding AlexNet for time series classification. Data Min. Knowl. Disc. **34**(6), 1936–1962 (2020)
6. Zhang X, Gao Y, Lin J, Lu C-T (2020) TapNet: Multivariate Time Series Classification with Attentional Prototypical Network. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 6845–52
7. Lines, J., Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Min. Knowl. Disc. **29**, 565–592 (2015)
8. Lines J, Taylor S, Bagnall A (2018) Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles. ACM Transactions on Knowledge Discovery from Data 12(5):52:1–52:35
9. Middlehurst M, Large J, Flynn M, Lines J, Bostrom A, Bagnall A (2021) HIVE-COTE 2.0: a new meta ensemble for time series classification. Machine Learning 110(11):3211–3243
10. Lucas, B., Shifaz, A., Pelletier, C., O'Neill, L., Zaidi, N., Goethals, B., Petitjean, F., Webb, G.: Proximity Forest: an effective and scalable distance-based classifier for time series. Data Min. Knowl. Disc. **33**(3), 607–635 (2019)
11. Gordon, D., Hendler, D., Kontorovich, A., Rokach, L.: Local-shapelets for fast classification of spectrographic measurements. Expert Syst. Appl. **42**(6), 3150–3158 (2015)
12. Bai, B., Li, G., Wang, S., Wu, Z., Yan, W.: Time series classification based on multi-feature dictionary representation and ensemble learning. Expert Syst. Appl. **169**, 114162 (2021)
13. Lin J,Khade R, LiY(2012) Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems 39(2):287–315
14. Senin P, Malinchik S (2013) SAX-VSM: interpretable time series classification using sax and vector space model. In: Proceedings of the 13th IEEE international conference on data mining (ICDM)
15. Schäfer, P.: The BOSS is concerned with time series classification in the presence of noise. Data Min. Knowl. Disc. **29**(6), 1505–1530 (2015)
16. 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, pp 947–56
17. Ye L, Keogh E (2011) Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Mining and Knowledge Discovery 22): 149–82
18. Hills, J., Lines, J., Baranauskas, E., Mapp, J., Bagnall, A.: Classification of time series by shapelet transformation. Data Min. Knowl. Disc. **28**(4), 851–881 (2013)
19. Karlsson I, Papapetrou P, Boström H (2016) Generalized random shapelet forests. Data Mining and Knowledge Discovery 30(5): 1053–85 (gRSF)
20. Yang J, Jing S, Huang G (2023) Accurate and fast time series classification based on compressed random Shapelet Forest. Applied Intelligence 53(5): 5240–5258 (CRSF)
21. Fang Z, Wang P, Wang W (2018) Efficient Learning Interpretable Shapelets for Accurate Time Series Classification. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp 497–508 (ELIS)
22. Li G, Choi B, Xu J, Bhowmick S S, Chun K-P, Wong G L-H (2022) Efficient Shapelet Discovery for Time Series Classification. IEEE Transactions on Knowledge and Data Engineering 34(3):1149–1163 (BSPCover)
23. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.-C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The UCR time series archive. IEEE/CAA J Automatica Sinica **6**(6), 1293–1305 (2019)
24. Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., Keogh, E.: Experimental comparison of representation methods and distance measures for time series data. Data Min. Knowl. Disc. **26**(2), 275–309 (2013)
25. Keogh E, Rakthanmanon T (2013) Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In: Proceedings of the 2013 SIAM International Conference on Data Mining, pp 668–676 (FS)
26. Grabocka J, Wistuba M, Schmidt-Thieme L (2015) Fast classification of univariate and multivariate time series through shapelet discovery. Knowledge and Information Systems 49(2): 429–54 (SD)
27. Bagnall A, Bostrom A, Lines J. UEA Time Series Classification. https://github.com/time-series-machine-learning/tsml-java