Mining distinguishing subsequence patterns with nonoverlapping condition

Youxi Wu, Yuehua Wang, Jingyu Liu, Ming Yu, Jing Liu & Yan Li

Cluster Computing
The Journal of Networks, Software Tools and Applications

ISSN 1386-7857

Cluster Comput
DOI 10.1007/s10586-017-1671-0

Springer

Cluster Computing
https://doi.org/10.1007/s10586-017-1671-0

Mining distinguishing subsequence patterns with nonoverlapping condition

Youxi Wu[1,2] · Yuehua Wang[1,2] · Jingyu Liu[1,2] · Ming Yu[1,2] · Jing Liu[1,2] · Yan Li[3]

Received: 27 October 2017 / Revised: 20 December 2017 / Accepted: 28 December 2017

Abstract
Distinguishing subsequence patterns mining aims to discover the differences between different categories of sequence databases and to express characteristics of classes. It plays an important role in biomedicine, feature information selection, time-series classification, and other areas. The existing distinguishing subsequence patterns mining only focuses on whether a pattern appears in a sequence, regardless of the number of occurrences of the pattern in the sequence and the proportion of the pattern in the entire sequence database, which affects the discovery of the distinguishing patterns when there are a large number of irrelevant occurrences. Therefore, the nonoverlapping conditional distinguishing subsequence patterns mining algorithm is proposed. In this paper, we focus on the number of nonoverlapping occurrences that effectively reduce the number of irrelevant or redundant occurrences, and in this way, the number of occurrences can be better grasped. At the same time, we use a specially designed data structure, namely, a Nettree, to avoid backtracking. In addition, we use the distinguishing patterns as classification features, and carry out classification experiments on DNA sequences and time-series data with two classes. Extensive experimental results and comparisons demonstrate the efficiency of the proposed algorithm and the correctness of the feature extraction.

Keywords Nonoverlapping occurrences · Distinguishing subsequence pattern · Nettree · Feature extraction

1 Introduction

Data mining, as an important means of data analysis and information extraction [1], has received extensive attention from researchers. As an important branch of data mining, sequence pattern mining [2] is widely used in mining information [3], time-series analysis and prediction [4], and other fields [5], and has become one of the focuses of various studies. Distinguishing subsequence patterns [6] and other new types of subsequence patterns proposed provide a new direction for sequential pattern mining. To satisfy user demand better, a gap constraint has been introduced in traditional sequential pattern mining. A gap constraint exists in sequence pattern mining [7] that makes the occurrences of patterns more flexible. The wide use of a gap constraint enhances the expressiveness and applicability of patterns. Sequential pattern mining with a gap constraint plays an important role in information retrieval [8], computational biology [9], feature extraction [10], and emergency medical recognition [11], among others.

Distinguishing subsequence patterns mining aims to discover differences between different categories or conditions, and has become a hotspot in the field of data mining, and is commonly applied in real life [12], such as shareholders determining whether to buy a stock by comparing changes in the stock market index. By comparing the price and quality of products, customers choose products with a high cost-performance. In an enterprise, an HR chooses applicants who meet the requirements by comparing their abilities.

Ji et al. [6] first proposed the concept of distinguishing patterns and used the ConSGapMiner algorithm to solve the problem of the minimum distinguishing pattern mining with a gap constraint, which laid an important foundation for future studies. Based on [6], Wang et al. [13] proposed the concept of a density constraint, and introduced it into dis-

✉ Yan Li
wuc567@163.com

1 School of Computer Science and Engineering, Hebei University of Technology, Tianjin 300401, China

2 Hebei Province Key Laboratory of Big Data Calculation, Tianjin 300401, China

3 School of Economics and Management, School of Hebei University of Technology, Tianjin 300401, China

Published online: 09 January 2018

Table 1 Occurrences of pattern P in sequence S

S	s_1 G	s_2 T	s_3 G	s_4 T	s_5 G
1st occurrence	G	T	G	·	·
2nd occurrence	G	T	·	·	G
3rd occurrence	G	·	·	T	G
4th occurrence	·	·	G	T	G

tinguishing pattern mining to mine patterns that satisfy the density and gap constraints. Yang et al. [14] added the top-k concept to distinguishing pattern mining to avoid pattern loss caused by an improper threshold. Wang et al. [15] proposed a minimum distinguishing pattern mining algorithm with a compact gap constraint without a pre-set gap constraint to avoid an improper threshold owing to insufficient prior knowledge. Our previous studies showed that nonoverlapping subsequence patterns mining with a gap constraint can make full use of the characteristics of the elements in the sequence. In addition, Wu et al. [16] proposed a nonoverlapping subsequence patterns mining with a gap constraint algorithm that balances completeness with the Apriori property [17].

Example 1 Suppose that the sequence $S = s_1s_2s_3s_4s_5 =$ GTGTG and pattern $P =$ G[0, 2]T[0, 2]G are given. The occurrences of P in S are shown in Table 1. The elements of the same position for any two occurrences cannot be the same in a sequence with the nonoverlapping condition. There are thus two occurrences, ⟨1,2,3⟩ and ⟨3,4,5⟩, and the support is 2.

Suppose that a positive database $D+$ is $\{S_1 =$ GTGTG, $S_2 =$ GTGGTG, $S_3 =$ GGTTGGGTG$\}$, negative database $D-$ is $\{S_4 =$ GTATC, $S_5 =$ ACGTC$\}$, positive support threshold is 2, and negative support threshold is 1. A distinguishing subsequence pattern is a pattern whose support in a positive database is greater than the positive support threshold, whereas support in a negative database is less than the negative support threshold, and that satisfies the gap constraint. Because the supports of G[0,2]T[0,2]G are 3 and 0 in $D+$ and $D-$, respectively, pattern P is a distinguishing subsequence pattern.

This will cause a loss of distinguishing patterns if we adopt the density threshold method in [13]. Let us follow the same example. In [13], the density calculation method is $density = \frac{\text{the number of occurrences in sequence}}{\text{the number of possible occurrences in sequence}}$. G[0,2]T[0,2]G occurs four times in S_1, namely, ⟨1,2,3⟩, ⟨1,2,5⟩, ⟨1,4,5⟩, and ⟨3,4,5⟩, and there are nine possible occurrences, ⟨1,2,3⟩, ⟨1,2,4⟩, ⟨1,2,5⟩, ⟨1,3,4⟩, ⟨1,3,5⟩, ⟨1,4,5⟩, ⟨2,3,4⟩, ⟨2,3,5⟩, and ⟨3,4,5⟩. If we agree that the density threshold is 0.5, S_1 cannot satisfy the *gd-support* count of pattern P, which should be greater than the threshold. In addition, S_2, and S_3 are also not satisfied in this way. Continuing to assume that the positive support threshold is 0.5, according to the calculation method $sup = \frac{\text{the number of sequences whose density is greater than threshold in database}}{\text{the number of sequences in datasbase}}$, then $sup(P, D+) = 0$, which is less than the positive class support threshold of 0.5, and thus P will not be a distinguishing pattern. It can be seen that P occurs frequently in a positive class, and the cause of a loss of P is due to the fact that a large number of irrelevant patterns under no-condition have a reduced pattern density, and thus, the sequence cannot be a *gd-support* count. In addition, when calculating the support, a sequence that satisfies the density threshold is simply used as a *gd-support* count, weakening the relationship between the patterns and database, and ignoring the proportion of patterns in the entire database.

To solve the above problems, we conducted the following in this study:

(1) We used a Nettree to solve distinguishing subsequence patterns with the nonoverlapping condition to improve the algorithm efficiency, and proposed the nonoverlapping conditional distinguishing subsequence patterns mining (NOCM) algorithm to reduce redundancy.
(2) We employed the average support rate, which takes the number of occurrences in a sequence and the ratio in the entire database into account, making the mining results more reasonable.
(3) We designed an approach for applying distinguishing subsequence patterns as classification features, and verified its effectiveness and efficiency on some binary datasets.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 provides a definition of the problem and describes it through concrete examples. Section 4 proposes the NOCM algorithm based on a Nettree, and illustrates the working principle of NOCM through some examples. Section 5 shows the mining effect of NOCM and the effect of distinguishing subsequence patterns for sequence classification. Finally, we conclude the paper and discuss the future work in Sect. 6.

2 Related work

Subsequence pattern mining has laid the foundation for research in many fields and plays an important role in practical applications. For example, Zhang et al. [18] addressed sequence pattern mining with periodic gap constraints, to apply in mining DNA sequence. Zhang et al. [19] proposed a method of mining frequent patterns with occupancy, which is used in the print-area recommendation and the travel-landscape recommendation. The two methods do not take the classification issue into account. Li et al. [10] proposed

an efficient algorithm for mining the closed frequent patterns with gap constraints and applied them to the feature selection of classification or clustering [20,21]. This research adopted the frequent patterns as the features. However, it is difficult to treat these features as distinguishing features. The following researches focused on mining distinguishing patterns for the classification issues. Wang et al. [13] proposed a new method of distinguishing pattern mining based on density-aware, and used it to identify human genetic variations. Yang et al. [14] and Wang et al. [15] also mined distinguishing subsequence patterns. And [14] introduced top-k to reduce the output of the patterns, while [15] adopted an idea which is without a predefined gap constraint. Therefore, these methods can achieve better performance for the classification issues. However, all these above methods employ no-condition to find the frequent patterns or the distinguishing patterns.

Our previous research showed that subsequence pattern mining with nonoverlapping condition is a better strategy than that with no-condition since the former method meets the Apriori property while the latter is difficult to meet the Apriori property [16]. One of our previous works [22] is a pattern matching task which is one of the key issues of sequence pattern mining task [16] since calculating the support of a pattern actually is pattern matching task. In research [16], we investigated subsequence pattern mining with nonoverlapping condition which does not focus on feature selection. However, this paper focuses on mining the distinguishing patterns.

Considering that the relevance and differences of this paper with these papers, we therefore make a comparison of related research in Table 2.

It can be seen from Table 2 that the research work in this paper is the closest to the research [13]. The difference is that, research [13] focused on sequence pattern mining with no-condition, while this paper studies sequence pattern mining with nonoverlapping condition.

It is not difficult to find through Table 2 that this paper is characterized by the mining of distinguishing subsequence patterns with nonoverlapping condition. Such mining is of great importance in practice. For example, the study of [23] stipulated the order of occurrence of events, but from the background analysis, the same event can only occur in one event sequence at a time, but not in several different event sequences. If there is nonoverlapping sequence of events, there will be more significance in the practical application.

Table 2 The comparison of related work

Algorithm	Mining/Matching	Type of mining	Type of condition	Length constraints	Database type
Zhang et al. [18]	Pattern mining	Frequent pattern mining	No-condition	No	Single class database
Li et al. [10]	Pattern mining	Frequent pattern mining	No-condition	No	Multi class database
Wang et al. [13]	Pattern mining	Distinguishing pattern mining	No-condition	No	Binary class database
Wu et al. [22]	Pattern matching	–	Nonoverlapping	Yes	Single class database
Wu et al. [16]	Pattern mining	Frequent pattern mining	Nonoverlapping	Yes	Single class database
This paper	Pattern mining	Distinguishing pattern mining	Nonoverlapping	Yes	Binary class database

– indicates that it does not involve in this

3 Problem definition

In this section, the definitions about distinguishing subsequence patterns with nonoverlapping condition are presented

and several examples are given to illustrate the definitions clearly.

Definition 1 (*sequence*) A sequence S with length n is an ordered list of events, which can be denoted as $S = s_1 s_2 \ldots s_i \ldots s_n$, where $1 \leq i \leq n$ and $s_i \in \Sigma$. Here, Σ represents a set of events, where the number of events in the set is denoted as $|\Sigma|$.

For example, in the DNA sequences, Σ is {A,T,C,G} and $|\Sigma|$ is 4.

Definition 2 (*pattern with gap constraints*) A pattern with gap constraints can be denoted as $P = p_1[a_1, b_1]p_2 \ldots p_j[a_j, b_j]p_{j+1} \ldots [a_{m-1}, b_{m-1}]p_m (1 \leq j \leq m)$, where $p_j \in \Sigma$, a_j and b_j are two given non-negative integers that represent the minimal and maximal gap constraints between p_j and p_{j+1}, respectively. The length of P is m.

Definition 3 (*occurrence*) If there exists a group of m integers $I = \langle i_1, i_2, \ldots i_m \rangle$ that satisfies: $1 \leq i_1 \leq i_2 < \ldots < i_m \leq n$, $a_j \leq i_{j+1} - i_j - 1 \leq b_j$, $p_1 = S_{i_1}$, $p_2 = S_{i_2}$, ..., $p_m = S_{i_m}$, then I is an occurrence of pattern P in sequence S.

Definition 4 (*nonoverlapping occurrences*) Let $I = \langle i_1, i_2, \ldots i_m \rangle$ and $I' = \langle i'_1, i'_2, \ldots i'_m \rangle$ be two occurrences. If and only if $\forall 1 \leq j \leq m$: $i_j \neq i'_j$, I and I' are two nonoverlapping occurrences.

In Example 1, $I_1 = \langle 1, 2, 3 \rangle$ and $I_2 = \langle 3, 4, 5 \rangle$ are two nonoverlapping occurrences of P in sequence S_4. Although I_1 and I_2 both use a character at position 3, I_1 and I_2 are two nonoverlapping occurrences because position 3 differs in I_1 and I_2.

Definition 5 (*support*) With the nonoverlapping condition, the support of pattern P in sequence S is the total number of occurrences of P in S, denoted by $sup(P, S)$.

Definition 6 (*offset occurrence*) If there exists a sequence $L = \langle l_1, l_2, \ldots l_n \rangle$ that satisfies $1 \leq l_1 \leq l_2 < \cdots < l_m \leq n$, $a \leq l_{j+1} - l_j - 1 \leq b$, then L is an offset occurrence of pattern P in sequence S.

Definition 7 (*nonoverlapping offset occurrences*) Let $L =< l_1, l_2, \ldots l_j >$ and $L' =< l'_1, l'_2, \ldots l'_j >$ be two offset occurrences. If and only if $\forall 1 \leq j \leq m$: $l_j \neq l'_j$ when $j = j'$, I and I' are two nonoverlapping offset occurrences. Using $ofs(P, S)$ represents the number of nonoverlapping offset occurrences of pattern P in sequence S.

Example 2 Following Example 1, occurrences not only need to satisfy the gap constraints, they also need to ensure that the corresponding position in the sequence of elements is the same as the pattern, whereas the offset occurrences that only need to satisfy the gap constraint, regardless of whether the element at the position is the same as the pattern. Therefore, the nonoverlapping offset occurrences of P in sequence S are: $\langle 1, 2, 3 \rangle$, $\langle 2, 3, 4 \rangle$, and $\langle 3, 4, 5 \rangle$. In addition, its $ofs(P, S)$ is 3.

Lemma 1 *Under the nonoverlapping condition, given sequence $S = s_1 s_2 \cdots s_n$ and pattern $P = p_1 p_2 \cdots p_m$, the total number of nonoverlapping offset occurrences of pattern P in sequence S is $n - m + 1$.*

Proof Compared with the definition of occurrence, p_j can be different with S_{i_j}. Therefore, the nonoverlapping offset occurrences of pattern P in S should be $\langle 1, 2, \ldots, m \rangle$, $\langle 2, 3, \ldots, m+1 \rangle$, $\langle 3, \ldots, m+1, m+2 \rangle$, ..., $\langle n - m + 1, \ldots, n-1, n \rangle$, for a total of $n - m + 1$. □

Definition 8 (*support rate*) The support rate of pattern P in sequence S can be written as $rate(P, S) = sup(P, S)/ofs(P, S)$.

Definition 9 (*binary classification sequence database*) A binary classification sequence database is a collection of sequences in two mutually exclusive classes database, which can be expressed as $D = \{S_1, S_2, \ldots, S_M, S_{M+1}, \ldots, S_N\}$, where $S_1 \cdots S_M$ comes from the database of $D+$, and $S_{M+1} \cdots S_N$ comes from the database of $D-$. The number of sequences contained in the sequence database can be denoted as $|D|$.

Definition 10 (*nonoverlapping occurrences between multiple sequences*) Let $D = \{S_1, S_2, \ldots, S_M\}$ be a multi-sequence database, the occurrences $I = \langle i_1, i_2, \ldots i_m \rangle$ and $I' = \langle i'_1, i'_2, \ldots i'_m \rangle$ of pattern P in any two sequences S_j and $S_{j'}$ can be two nonoverlapping occurrences between multiple sequences.

Example 3 Suppose sequence database $D = \{S_1 = \text{GTGTG}, S_2 = \text{GGTTGG}\}$. The nonoverlapping occurrences of pattern P= G[0,2]T[0,2]G in S_1 are then ⟨1,2,3⟩ and ⟨3,4,5⟩, and are ⟨1,3,5⟩ and ⟨2,4,6⟩ in S_2. According to Definition 10, the occurrence ⟨1,2,3⟩ in S_1 and the occurrence ⟨1,3,5⟩ in S_2 are two nonoverlapping occurrences between multiple sequences. Similarly, so are ⟨2,3,4⟩ and ⟨2,4,6⟩. The supports $sup(P, S_1)$ and $sup(P, S_2)$ of pattern P = G[0, 2]T[0, 2]G in S_1 and S_2, respectively, are both 2.

Definition 11 (*average support rate*) Let sequence database $D = \{S_1, S_2, \ldots, S_M\}$, then $|D| = M$. The average support rate can be denoted as $r(P, D)$, and $r(P, D) = \frac{\sum_{i=1}^{n} rate(P,S)}{M}$. Given the average support rate threshold $minsup(D)$, if r is not less than the given $minsup(D)$, then pattern P is said to be frequent.

Example 4 Continuing to example 3, the average support rate of P in D is $r(P, D) = 0.583$. Supposing that the average support rate threshold $minsup(D) = 0.3$, because $r > minsup(D)$, then P is frequent.

Table 3 Example of sequence database

ID	Sequence	Class
1	GTGTG	$D+$
2	GTGGTG	$D+$
3	GGTTGGGTG	$D+$
4	GTATC	$D-$
5	ACGTC	$D-$

Definition 12 (*nonoverlapping conditional distinguishing subsequence pattern*) Given the positive sequence database $D+$ and the negative sequence database $D-$, the positive average support rate threshold $minsup(D+)$ and negative average support rate threshold $minsup(D-)$, gap constraints is $[a, b]$. The pattern $P = p_1[a, b]p_2 \cdots [a, b]p_m$ is the nonoverlapping conditional distinguishing subsequence pattern (NOCP) that satisfies the gap constraint if P satisfies the following conditions:

(1) P is frequent in positive sequence database $D+$, that is, $r(P, D+) \geq minsup(D+)$.
(2) P is not frequent in negative sequence database $D-$, that is, $r(P, D-) \leq maxsup(D-)$.

The average support rate r is used as the threshold criterion because the average support rate takes into account the number of occurrences of patterns in each sequence and the proportion of the entire sequence database at the same time, and will not be affected by the number of positive and negative sequences. Let us combine the following examples:

Example 5 Given the sequence database shown in Table 3, the positive support rate threshold is 0.5, and the negative support rate threshold is zero. Here, P = G[0, 2]T[0, 2]G is not a distinguishing subsequence pattern when adopting the method in [13]. Because $sup(P, D+) = 0$ and $sup(P, D-) = 0$, $sup(P, D+)$ is less than 0.5 and $sup(P, D-)$ is equal to zero. It is clearly unreasonable to state that G[0,2]T[0,2]G is not a distinguishing subsequence pattern because G[0,2]T[0,2]G occurs frequently in a positive sequence database, and P occurs in a positive sequence database far more than a negative sequence database.

Using the method proposed in this paper, we first calculate the support rate of the pattern in each sequence, and then calculate the ratio in the positive and negative databases separately, that is, the average support rate, $r(P, D+) = 0.532$, $r(P, D-) = 0$. It can be seen that the average support ratio of P in a positive database is much larger than in a negative database. This method reduces redundant occurrences and avoids interference with patterns of excessive possible occurrences, and takes into account the contribution of occurrences in the individual sequence and the entire database. This, therefore, makes the evaluation criteria more reasonable.

Definition 13 (*maximum prefix and maximum suffix*) Given a pattern $R = r_1 r_2 \cdots r_m (m \geq 2)$, if $P = r_1 r_2 \cdots r_{m-1}$, then P is the maximum prefix sub-pattern of R. This can be denoted as $prefix(R) = P$, and has $R = P \cup r_{m-1}$. If $Q = r_2 \cdots r_{m-1} r_m$, then Q is the maximum suffix sub-pattern of R, and is denoted as $suffix(R) = Q$, and $R = r_1 \cup Q$.

Definition 14 (*prefix–suffix connection method*) Suppose that pattern $A = a r_1 r_2 \cdots r_{m-1}$ and $B = r_1 r_2 \cdots r_{m-1} b$, from Definition 16 we know that $suffix(A) = prefix(B) = R$, and thus, A and B can be connected to generate a super-pattern T whose length is $m + 1$ using the operator $\oplus$, and $T = A \oplus B = aRb$.

Example 6 Given pattern P = ACTA and Q = CTAC, the maximum prefix sub-pattern and maximum suffix sub-pattern of P are the same as CTA. Therefore, super-pattern $T = suffix(A) \oplus prefix(B)$ = ACTAC can be generated.

4 Mining distinguishing subsequence patterns with the nonoverlapping condition

In this section, we introduce the special designed data structure at first. Then we present our algorithm. Finally, we analyze the space and time complexities of our algorithm.

Under the nonoverlapping condition, there are two problems: (1) how to effectively identify the character in a sequence that can be reused by applying a nonoverlapping support, and (2) how to quickly prune out candidate patterns that cannot exist to avoid calculating a large number of irrelevant patterns. To solve these two problems, we use a feature of a Nettree to calculate the support, that is, we allow the same node label to appear many times at different levels, which can effectively identify whether characters in the sequence can be reused. We use the pattern growth approach to minimize the support calculation of irrelevant patterns.

Definition 15 (*Nettree*) A Nettree [24–26] is a type of data structure that is similar to a tree, and consists of a root, leaf, level, parent, child, and so on. Nevertheless, a Nettree is clearly different from a tree data structure based on the following characteristics:

(1) A Nettree may have one or more roots, where n $\geq$ 1.
(2) Any node except a root may have more than one parent, but all parents must appear at the same level.
(3) The same node label can appear at different levels. Node i in the jth level can be described as n_j^i.

(4) There may be more than one path from one node to its ancestor.

Definition 16 (*root-leaf* path) A path from a root to the termination of the leaf is called a root-leaf path.

Lemma 2 *An occurrence of pattern P in sequence S can be expressed as a root-leaf path in a Nettree.*

Proof Our previous work [24] proved that each occurrence of pattern P in sequence S can be transformed into a root-leaf path in a Nettree, that is, each occurrence can be expressed as a root-leaf path. □

Lemma 3 *Let A and B be two root-leaf paths that do not contain any of the same nodes. The corresponding occurrences of A and B are two nonoverlapping occurrences.*

Proof A and B are $< n_1^{a_1}, n_2^{a_2}, \ldots, n_m^{a_m} >$ and $< n_1^{b_1}, n_2^{b_2}, \ldots, n_m^{b_m} >$, respectively. For all $i(1 \leq i \leq m)$, a_i is not equal to b_i, because A and B do not contain the same node. Therefore, $\langle a_1, a_2, \ldots, a_n \rangle$ and $\langle b_1, b_2, \ldots, b_n \rangle$ are two nonoverlapping occurrences. □

Lemma 4 *If a non-leaf node does not have any children, it can be safely pruned.*

Proof Supposing node n_j^i is a non-leaf node in a Nettree without any children, then node n_j^i cannot reach the leaf. Therefore, n_j^i is an invalid node and should be pruned. After pruning n_j^i, we should also check whether its parent nodes have other children. A node with no child should also be pruned. □

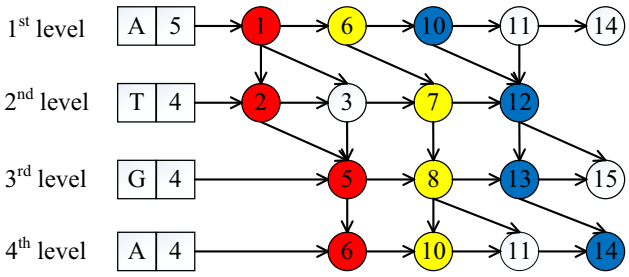


Fig. 1 The corresponding Nettree of Example 7

We use the following example to illustrate how the algorithm works.

Example 7 Given sequence $S = s_1s_2s_3s_4s_5s_6s_7s_8s_9s_{10}s_{11}s_{12}s_{13}s_{14}s_{15}$ = ATTCGATGCAATGAG and pattern P = A[0, 2]T[0, 2]G[0, 2]A, the Nettree shown in Fig. 1 can be created.

Using the Nettree in Fig. 1, we can obtain three root-leaf paths: $\langle n_1^1, n_2^2, n_3^5, n_4^6 \rangle$, $\langle n_1^6, n_2^7, n_3^8, n_4^{10} \rangle$ and $\langle n_1^{10}, n_2^{12}, n_3^{13}, n_4^{14} \rangle$. According to Lemma 3, there are three nonoverlapping occurrences. As can be seen from Fig. 1, node n_2^3 has no child and is an invalid node. Hence, n_2^3 can be pruned by combining with Lemma 4. As we can see, the character at position 6 are reused in $\langle n_1^1, n_2^2, n_3^5, n_4^6 \rangle$ and $\langle n_1^6, n_2^7, n_3^8, n_4^{10} \rangle$ since position 6 is different in the two occurrences with nonoverlapping condition. Namely, if a character appears at different levels in the Nettree, then it can be reused.

An algorithm, called Support, which computes the support of P in S, $sup(P, S)$, is shown in Algorithm 1. Because Support is used for calculation support, in other words, it is used for mining occurrences, we prune the invalid nodes while mining the occurrences of P.

Algorithm 1 Support

Input: Sequence S, Pattern P, $gap = [a,b]$, $len = [minlen,maxlen]$
Output: $sup(P,S)$

```
1: Create a nettree of P in S;
2: Prune the invalid nodes according to Lemma 4;
3: for each n_j^i in nettree do
4:     node[1] ← n_j^i;
5:     for j = 1 to nettree.level-1 step 1 do
6:         node[j+1] ← node[j]'s leftmost child with length constraint;
7:     end for
8:     sup(P,S) + +;
9:   Prune the invalid nodes according to Lemma 4;
10: end for
11: return sup (P,S);
```

The NOCM algorithm first scans a sequence at a particular time to generate the event set Σ. It then generates candidate patterns in positive database using the prefix-suffix connection method on the basis of Σ, and uses Nettree to calculate the support. Based on the nature of the Nettree, it is known that the number of levels in the Nettree is equal to the length of the patterns, and a complete path from the root to the leaf represents an occurrence of P in the sequence with the root to the leaf satisfying the gap constraint. Frequent patterns generated in the positive database are used as the candidate patterns of the negative mining, and the distinguishing sequence patterns are finally generated.

Theorem 1 *During the process of frequent pattern mining, if pattern P is infrequent, then any of its super patterns is infrequent.*

Proof Mining in a positive database is the mining of frequent patterns of the nonoverlapping condition. Suppose that $sup(P)$ is the support of pattern P, $minsup(D)$ is the average support rate threshold. If P is infrequent, namely, $sup(P) < minsup(D)$, then when element e is added to pattern P, the generated super pattern (eP or Pe) occurrences cannot be more than the original pattern P, and must be less than or equal to the occurrences of the original model. □

The following example illustrates the principle of the NOCM algorithm.

Example 8 We mine the distinguishing subsequence patterns with the nonoverlapping condition in sequence database D shown in Table 3 and the selected parameters are the gap constraint = [0,2], $minsup(D+) = 0.25$, and $maxsup(D-) = 0.1$.

First, we compute the average support rate for the patterns in Σ. We know that $r(\mathrm{A}, D+) = 0.22$ is less than $minsup(D+)$. According to Theorem 1, all super patterns of "A" are also infrequent, thereby pruning "A" and its super patterns. Since $r(\mathrm{T}, D+) = 0.37$ is greater than $minsup(D+)$, pattern "T" which is stored is a frequent pattern in positive class. Similarly, we prune pattern "C" and its super patterns, and store pattern "G" since $r(\mathrm{C}, D+)$ and $r(\mathrm{G}, D+)$ are 0 and 0.63, respectively. Then we can generate the candidate patterns in positive class {TG, TT, GG, GT} with length of 2 by "T" and "G". We know that $r(\mathrm{TG}, D+) = 0.40$, $r(\mathrm{TT}, D+) = 0.20$, $r(\mathrm{GG}, D+) = 0.56$, and $r(\mathrm{GT}, D+) = 0.44$. Therefore, pattern "TT" and its supper patterns are pruned since $r(\mathrm{TT}, D+)$ is less than $minsup(D+)$. Repeat this process until there is no new frequent pattern in positive class generated. After that, we compute the support rate for these negative candidate patterns. The pattern will be a distinguishing subsequence pattern if its average support rate is less than $maxsup(D-)$. Here, $r(\mathrm{G}, D-)$ and $r(\mathrm{T}, D-)$ are 0.2 and 0.3 separately which are more than $maxsup(D-)$, so they are not distinguishing patterns. Both $r(\mathrm{TG}, D-)$ and $r(\mathrm{GG}, D-)$ are 0 which are less than $maxsup(D-)$, thus "TG" and "GG" are distinguishing patterns. Using this method, we find eight distinguishing patterns: "GG", "TG", "GGG", "GGT", "GTG", "TGG", "GGTG" and "GTGG".

We describe the detailed procedures of the NOCM algorithm in Algorithm 2 which employs the Apriori property to prune candidate patterns in $D+$.

Algorithm 2 NOCM: Mining distinguishing subsequence patterns

Input: Database $D+$, $D-$, $gap = [a,b]$, $minsup(D+)$, $maxsup(D-)$
Output: Distinguishing Subsequence Patterns set DSP

```
Scan sequence S in turn, calculate the average support rate of each pattern with length 1, and put the frequent patterns with length 1 into a queue F[1];
len ← 1;
C = GFC(F[len]);
while C <> NULL && D+ <> NULL do
    for each candidate in C do
        if r(candidate,D+) ≥ minsup(D+) then
            F[len] ← candidate;
        else
            Prune the candidate and its super patterns according to Theorem 1;
        end if
    end for
    C = GCP(F[len]);
end while
for each pattern P in F do
    if r(P,D-) ≤ maxsup(D-) then
        DSP.enqueue(P);
    end if
end for
return DSP;
```

Algorithm 3 is proposed to generate candidate patterns in positive class.

Algorithm 3 GCP(F): Generating candidate patterns

Input: Frequent pattern set F
Output: Frequent candidates set C
1: $begin \leftarrow 1$;
2: **for** $i = 1$ to $|F|$ **do**
3: P = suffix($F[i]$);
4: Q = prefix($F[begin]$);
5: **while** $Q == R$ **do**
6: C.enqueue($suffix(P) \oplus prefix(Q)$);
7: $begin \leftarrow begin + 1$;
8: **end while**
9:**end for**
10:**return** C;

Theorem 2 *Let $m, n, w, l_{pos}, l_{neg}$ and r be the maximal length of a pattern, the maximal length of a sequence in the database SDB, $b-a+1$ (where a and b are the minimal and maximal gap constraint), the number of the positive class candidate patterns, the number of the negative class candidate patterns, and the size of Σ, respectively. Then the space complexity of NOCM is $O(m \times (n \times w + l_{pos}))$ in the worst case and $O(m \times (n \times w/r/r + l_{pos}))$ in the average case.*

Proof The maximal length of candidate and mined pattern is $O(m)$. The space complexity of frequent and candidate patterns in positive class is $O(m \times l_{pos})$. *Algorithm* 1 employs a Nettree to calculate the support of a pattern in a sequence. In the worst case, the Nettree has no more than m levels, each level has no more than n nodes, and each node has no more than w children. Thus it can be concluded that the space and time complexities of creating a Nettree are both $O(m \times n \times w)$. So the space complexity of NOCM mined in positive class is $O(m \times (n \times w + l_{pos}))$ in the worst case. The space complexity of mined patterns in negative class which is $O(m \times l_{neg})$ can be neglected compared with the space complexity of frequent patterns in positive class since l_{neg} is less than l_{pos}. Therefore, the space complexity of NOCM is $O(m \times (n \times w + l_{pos}))$. In addition, each level has no more than n/r nodes and each node has no more than w/r children in the average case. Hence, the space complexity of NOCM is $O(m \times (n \times w/r/r + l_{pos}))$ in the average case. □

Theorem 3 *The time complexity of NOCM is $O(m \times m \times n \times w \times l_{pos})$ in the worst case and $O(m \times m \times n \times w \times l_{pos}/r/r/r)$ in the average case, where m, n, w, l_{pos}, and r are given above.*

Proof The complexity of *Algorithm* 3 is $O(l_{pos} \times \log l_{pos})$ since it employs binary search. We know that the time complexity of creating Nettrees is $O(m \times n \times w)$ from the above theorem and the nonoverlapping occurrences are no more than n. Apparently, the depth of the Nettree is m. According to *Lemma* 4, in the $(m-1)$th level we may prune w nodes at most since each node has at most w parents. Similarly, at most $2 \times w$ nodes in the $(m-2)$th level could be pruned. Thus, there are $O(m \times m \times w)$ nodes could be deleted at most. Therefore, the time complexity of lines 5 to 12 in NOCM is $O(m \times n \times w + m \times m \times n \times w) = O(m \times m \times n \times w)$. There are l_{pos} candidate patterns in positive class, so the time complexity is $O((n \times m \times n \times \mathrm{w} + \log l_{pos}) \times l_{pos}) = O(n \times m \times n \times w \times l_{pos})$ in the worst case. Similarly, the time complexity of computing negative candidate patterns which is $O(n \times m \times n \times w \times l_{neg})$ can be neglected compared with $O(n \times m \times n \times w \times l_{pos})$ since l_{neg} is less than l_{pos}. Hence, the complexity of NOCM is $O(n \times m \times n \times w \times l_{pos})$. Moreover, as mentioned in the space complexity, the time complexity of NOCM is $O(m \times m \times n \times w \times l_{pos}/r/r/r)$ in the average case. □

5 Experimental results and analysis

In this section, we will demonstrate the correctness and efficiency of our algorithm with extensive experiments. All experiments were conducted on a computer with an Intel(R) Core(TM) i5-3210M 2.50 GHz CPU and 8.0 GB of RAM running the 64-bit version of Windows 7. VC++ 6.0 was used in the development of all algorithms applied, including NOCM and NOSEP. The data used in the experiments are DNA sequences and time-series data. The real DNA sequences were extracted from http://dbtss.hgc.jp/, and original time-series data can be obtained from www.cs.ucr.edu/~eamonn/time_series_data/.

5.1 Efficiency

We conducted several experiments on five groups of time-series data and a set of DNA sequences. NOCM is compared with NOSEP based on the number of patterns and the running time. Each group of experiments uses the same length

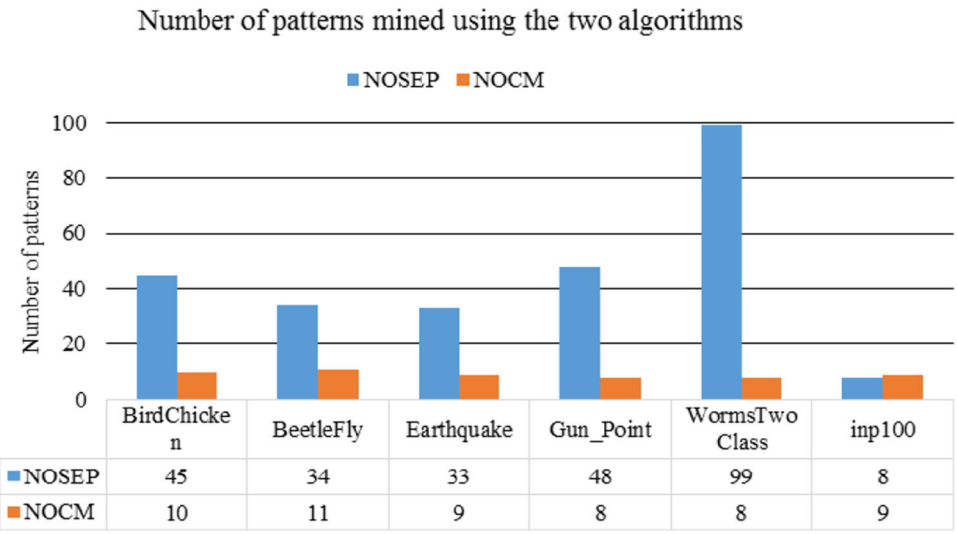


Fig. 2 Number of patterns mined using the two algorithms

constraint. Because NOSEP mines frequent patterns, for all sequences under a single class, only the minimum average support threshold *minsup* needs to be set. NOCM needs to set the positive average support rate threshold *minsup*($D+$) and negative maximum support rate threshold *maxsup*($D-$). To unify the standard and make the experiment comparable, it is necessary to keep *minsup* and *minsup*($D+$) consistent.

Figure 2 shows the number of patterns mined using NOSEP and NOCM with gap constraint [0,2], using the threshold of each parameter shown in Table 4.

From Fig. 2, we can see that the number of distinguishing patterns is generally much less than the number of frequent patterns. In addition, with a decrease in *minsup*($D+$), the number of distinguishing patterns gradually increases. For WormsTwoClass, NOSEP mines 99 frequent patterns when *minsup* is 0.01, whereas NOCM mines eight distinguishing patterns at the corresponding threshold. Under a condition in which the parameter setting is reasonable, the number of patterns mined by NOCM is less than NOSEP. Thus, redundant patterns can be effectively reduced, and the secondary selection of features in the sequence classification is avoided.

It is possible for the number of distinguishing patterns to be greater than or equal to the number of frequent patterns under certain threshold conditions. For dataset inp100 in Fig. 2, NOCM mines eight distinguishing patterns, whereas NOSEP mines nine frequent patterns when *minsup*/*minsup*($D+$) is 0.12 and *maxsup*($D-$) is 0.1. This is due to patterns appearing in a positive class far more frequently than in a negative class when the difference between the positive and negative classes is obvious, and thus its support will be averaged in frequent pattern mining, leading to a frequent decrease in support. Thus, the pattern cannot be a frequent pattern. However, it is precisely in line with a distinguishing pattern, resulting in the above phenomenon. In most cases, NOCM can be very good at finding significant distinguishing patterns.

Figure 3 shows the running time of NOSEP and NOCM with a gap constraint of [0, 2], and the threshold parameters shown in Table 4.

From Fig. 3, we can see that NOCM is fast when the gap constraint and *minsup*/*minsup*($D+$) are the same, and takes far less time than NOSEP. As *minsup*($D+$) increases, the time required for mining becomes increasingly shorter. This illustrates that the speed of NOCM has a significant advantage owing to the use of a Nettree, which avoids backtracking in the mining process and reduces the time consumption.

Table 4 Threshold parameter table used in the comparison experiment

Dataset	len_1	gap_1	*minsup*	len_2	gap_2	*minsup*($D+$)	*maxsup*(D-)
BirdChicken	1–15	[0,2]	0.033	1–15	[0,2]	0.033	0.03
BeetleFly	1–15	[0,2]	0.015	1–15	[0,2]	0.015	0.01
Earthquake	1–15	[0,2]	0.055	1–15	[0,2]	0.055	0.05
Gun_Point	1–15	[0,2]	0.03	1–15	[0,2]	0.03	0.02
WormsTwoClass	1–15	[0,2]	0.01	1–15	[0,2]	0.01	0.009
inp100	1–15	[0,2]	0.12	1–15	[0,2]	0.12	0.1

len_1 and gap_1 are the length and gap constraints of NOSEP respectively, and len_2 and gap_2 are the length and gap constraints of NOCM

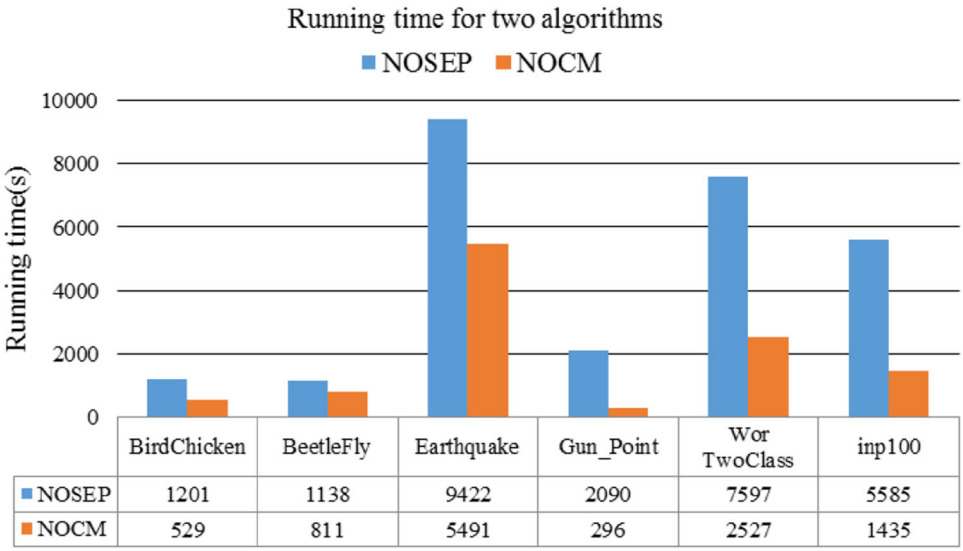


Fig. 3 Running time for two algorithms

Table 5 Corresponding number of features of different feature extraction approaches

Dataset	Extraction approach	*len*	*gap*	*minsup*($D+$)	*maxsup*($D-$)	Number of features
inp100	NOSEP	1–15	[0,2]	0.12	–	8
	NOCM	1–15	[0,2]	0.12	0.1	8
BirdChicken	NOSEP	1–15	[0,2]	0.033	–	45
	NOCM	1–15	[0,2]	0.033	0.03	10
BeetleFly	NOSEP	1–15	[0,2]	0.015	–	34
	NOCM	1–15	[0,2]	0.015	0.01	11
Earthquake	NOSEP	1–15	[0,2]	0.03	–	148
	NOCM	1–15	[0,2]	0.03	0.015	13
Gun_Point	NOSEP	1–15	[0,2]	0.07	–	14
	NOCM	1–15	[0,2]	0.07	0.06	9
WormsTwoClass	NOSEP	1–15	[0,2]	0.012	–	66
	NOCM	1–15	[0,2]	0.012	0.011	7

– indicates that this threshold does not need to be set

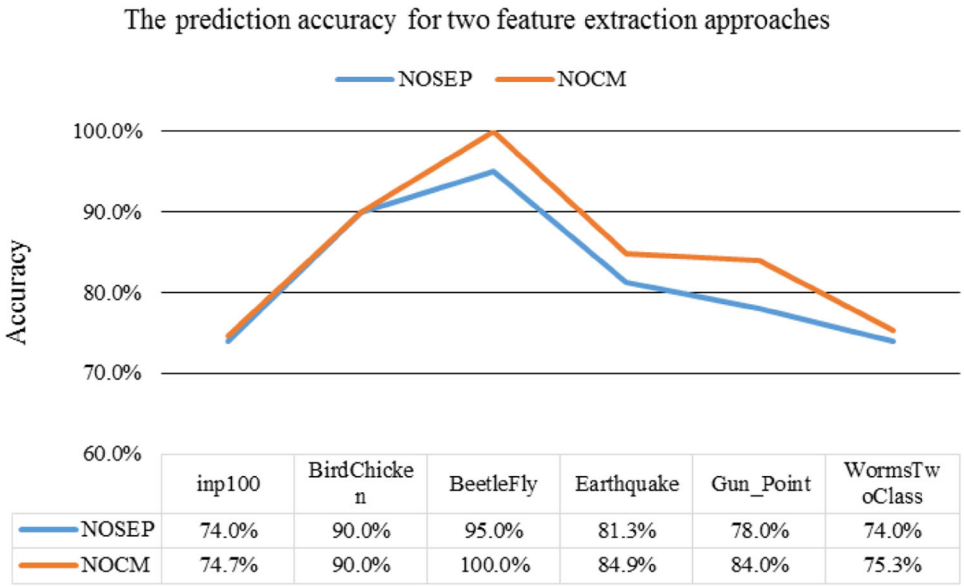
5.2 Classification effect

A distinguishing subsequence pattern can show well how things change under the different classes or conditions, and focuses on the differences in different datasets. Therefore, in this paper, we propose a feature extraction approach based on a distinguishing subsequence pattern. Breaking from a single situation of frequent patterns as the classification features [27–29], a new direction for feature extraction is provided.

The number of features used in the experiment and the corresponding feature extraction thresholds are shown in Table 5.

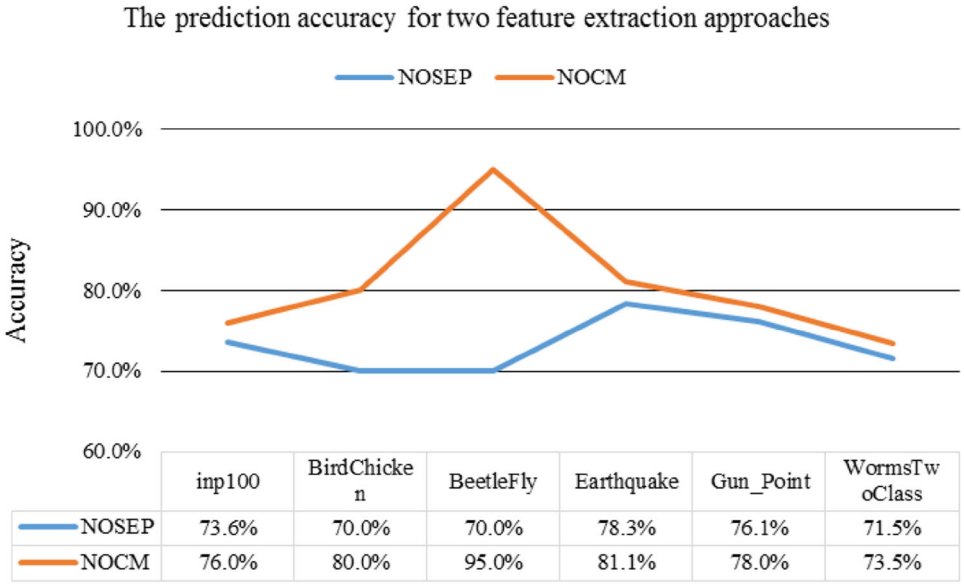
To verify the classification effect, we use the K-NN classifier to conduct experiments on a binary classification sequence database and compare it with the frequent patterns as features. The classification effect is evaluated based on the training and prediction accuracies, shown in Figs. 4 and 5, respectively. Clearly, the feature extraction approach with NOCM obtains a better classification effect and has a higher accuracy in different datasets. In general, the accuracy of its classification is better than feature extraction with NOSEP.

It can be seen from Figs. 4 and 5 that the classification accuracy of the two feature extraction approaches is basically the same, but training accuracy and prediction accuracy have significantly improved with choosing distinguishing subsequence patterns as classification features. For example, from the BeetleFly dataset in Figs. 4 and 5, we can know that the training accuracy of the two feature extraction methods of NOSEP and NOCM is 95.0 and 100.0% respectively, and the prediction accuracy is 70.0 and 95.0% respectively. Meanwhile, combined with Table 5, we can see that NOCM only used 11 features, whereas NOSEP used 34 features. Experimental results on each data set show that a relatively small number of distinguishing subsequence patterns can be used



	inp100	BirdChicken	BeetleFly	Earthquake	Gun_Point	WormsTwoClass
NOSEP	74.0%	90.0%	95.0%	81.3%	78.0%	74.0%
NOCM	74.7%	90.0%	100.0%	84.9%	84.0%	75.3%

Fig. 4 Comparison of the classification accuracy in training sets



	inp100	BirdChicken	BeetleFly	Earthquake	Gun_Point	WormsTwoClass
NOSEP	73.6%	70.0%	70.0%	78.3%	76.1%	71.5%
NOCM	76.0%	80.0%	95.0%	81.1%	78.0%	73.5%

Fig. 5 Comparison of the classification accuracy in testing sets

to achieve higher accuracy. Therefore, NOCM which can find the most distinguishing patterns is an effective feature extraction than the competitive method.

6 Conclusion

Existing distinguishing subsequence patterns mining algorithms pay less attention to the sequences in a database, and if long sequences appear in a database, or if the number of positive and negative sequences is not equal, the mining result will be unreasonable. In view of this problem, an effective distinguishing mining algorithm with the nonoverlapping condition, called NOCM, was proposed in this paper. NOCM not only can reduce redundant patterns, it can also pay attention to the proportion of patterns in each sequence and the entire database. This prevents the number or length of the sequences to have impact on the patterns, and makes distinguishing subsequence patterns more significant and reasonable. We also proposed a new feature extraction approach that employs nonoverlapping distinguishing subsequence patterns as features of a sequence classification. Extensive experimental results demonstrate the efficiency of NOCM, and a good classification effect was shown on a standard binary classification database.

As future work, more effective mining algorithm could be studied and better pattern mining method for feature extraction could be investigated, so as to improve the performance of classification.

Acknowledgements The work was supported in part by the National Natural Science Foundation of China under Grant 61673159, in part by the Natural Science Foundation of Hebei Province under Grant F2016202145, in part by the Science and the Technology Project of Hebei Province under Grant 15210325, and in part by the Graduate Student Innovation Program of Hebei Province under Grant CXZZSS2017037.

References

1. Malarvizhi, S.P., Sathiyabhama, B.: Frequent pagesets from web log by enhanced weighted association rule mining. Clust. Comput. **19**(1), 269–277 (2016)
2. Ding, B., Lo, D., Han, J., et al.: Efficient mining of closed repetitive gapped subsequences from a sequence database. In: IEEE 25th International Conference on Data Engineering, pp. 1024–1035 (2009)
3. Zhang, S., Du, Z., Wang, J.T.: New techniques for mining frequent patterns in unordered trees. IEEE Trans. Cybern. **45**(6), 1113–1125 (2015)
4. Tan, C., Min, F., Wang, M., et al.: Discovering patterns with weak-wildcard gaps. IEEE Access **4**, 4922–4932 (2016)
5. Feng, Y., Ji, M., Xiao, J., et al.: Mining spatial-temporal patterns and structural sparsity for human motion data denoising. IEEE Trans. Cybern. **45**(12), 2693–2706 (2015)
6. Ji, X., Bailey, J., Dong, G.: Mining minimal distinguishing subsequence patterns with gap constraints. Knowl. Inf. Syst. **11**(3), 259–286 (2007)
7. Wu, Y., Wang, L., Ren, J., et al.: Mining sequential patterns with periodic wildcard gaps. Appl. Intell. **41**(1), 99–116 (2014)
8. Chou, C., Jea, K., Liao, H.: A syntactic approach to twig-query matching on XML streams. J. Syst. Softw. **84**(6), 993–1007 (2011)
9. Cole, J., Chai, B., Farris, R., et al.: The Ribosomal Database Project (RDP-II): sequences and tools for high-throughput rRNA analysis. Nucleic Acids Res. **33**(suppl_1), D294–D296 (2005)
10. Li, C., Yang, Q., Wang, J., et al.: Efficient mining of gap-constrained subsequences and its various applications. ACM Trans. Knowl. Discov. Data **6**(1), 2 (2012)
11. Ghosh, S., Feng, M., Nguyen, H., et al.: Risk prediction for acute hypotensive patients by using gap constrained sequential contrast patterns. In: AMIA Annual Symposium Proceedings, pp. 1748–1757. American Medical Informatics Association (2014)
12. Drory Retwitzer, M., Polishchuk, M., Churkin, E., et al.: RNAPattMatch: a web server for RNA sequence/structure motif detection based on pattern matching with flexible gaps. Nucleic Acids Res. **43**(W1), W507–W512 (2015)
13. Wang, X., Duan, L., Dong, G., et al.: Efficient mining of density-aware distinguishing sequential patterns with gap constraints. In: International Conference on Database Systems for Advanced Applications, pp. 372–387. Springer, Cham (2014)
14. Yang, H., Duan, L., Hu, B., et al.: Mining Top-k distinguishing sequential patterns with gap constraint. J. Softw. **26**(11), 2994–3009 (2015). (in Chinese)
15. Wang, H., Duan, L., Zuo, J., et al.: Efficient mining of distinguishing sequential patterns without a predefined gap constraint. Chin. J. Comput. **39**(10), 1979–1991 (2016). (in Chinese)
16. Wu, Y., Tong, Y., Zhu, X., et al.: NOSEP: nonoverlapping sequence pattern mining with gap constraints. IEEE Trans. Cybern. (2017). https://doi.org/10.1109/TCYB.2017.2750691
17. Min, F., Wu, Y., Wu, X.: The Apriori property of sequence pattern mining with wildcard gaps. Int. J. Funct. Inform. Pers. Med. **4**(1), 15–31 (2012)
18. Zhang, M., Kao, B., Cheung, D., et al.: Mining periodic patterns with gap requirement from sequences. ACM Trans. Knowl. Discov. Data **1**(2), 7 (2007)
19. Zhang, L., Luo, P., Tang, L., et al.: Occupancy-based frequent pattern mining. ACM Trans. Knowl. Discov. Data (TKDD) **10**(2), 14 (2015)
20. Wu, Y., Liu, D., Jiang, H.: Length-changeable incremental extreme learning machine. J. Comput. Sci. Technol. **32**(3), 630–643 (2017)
21. Egho, E., Gay, D., Boulle, M., et al.: A parameter-free approach for mining robust sequential classification rules. Knowl. Inf. Syst. **52**(1), 53–81 (2017)
22. Wu, Y., Shen, C., Jiang, H., et al.: Strict pattern matching under non-overlapping condition. Sci. China Inf. Sci. **60**(1), 012101 (2017)
23. Yen, S., Lee, Y.: Mining non-redundant time-gap sequential patterns. Appl. Intell. **39**(4), 727–738 (2013)
24. Wu, Y., Wu, X., Min, F., et al.: A Nettree for pattern matching with flexible wildcard constraints. In: International Conference on Information Reuse and Integration, pp. 109–114 (2010)
25. Wu, Y., Tang, Z., Jiang, H., et al.: Approximate pattern matching with gap constraints. J. Inf. Sci. **42**(5), 639–658 (2016)
26. Wu, Y., Fu, S., Jiang, H., et al.: Strict approximate pattern matching with general gaps. Appl. Intell. **42**(3), 566–580 (2015)
27. Fradkin, D., Mörchen, F.: Mining sequential patterns for classification. Knowl. Inf. Syst. **45**(3), 731–749 (2015)
28. Zhou, C., Cule, B., Goethals, B.: Pattern based sequence classification. IEEE Trans. Knowl. Data Eng. **28**(5), 1285–1298 (2016)
29. Fong, S., Wong, R., Vasilakos, A.: Accelerated PSO swarm search feature selection for data stream mining big data. IEEE Trans. Serv. Comput. **9**(1), 33–45 (2016)

Youxi Wu received the Ph.D. degree in School of Electrical Engineering, Hebei University of Technology. He is currently a Ph.D. supervisor and a professor with School of Computer Science and Technology, Hebei University of Technology. His current research interests include data mining and machine learning.

Yuehua Wang is a master candidate in School of Computer Science and Technology and a Ph.D. candidate in Economics and Management, Hebei University of Technology. Her current research interests include data mining.

Jingyu Liu received the Ph.D. degree in School of Computer Science, Beijing Institute of Technology. He is an associate professor with School of Computer Science and Technology, Hebei University of Technology. His main research interests include network storage and storage security.

Jing Liu received the Ph.D. degree in School of Communication Engineering, Communication University of China. She is a professor with School of Computer Science and Technology, Hebei University of Technology. Her current research interests include big data analytics, data mining, and digital manufacturing.

Ming Yu received the Ph.D. degree in School of Computer Science, Beijing Institute of Technology. He is currently a Ph.D. supervisor and a professor with School of Computer Science and Technology, Hebei University of Technology. His main research interests include image processing and machine learning.

Yan Li received the Ph.D. degree in School of Management Science and Engineering, Tianjin University. She is an associate professor with School of Economics and Management, Hebei University of Technology. Her current research interests include supply chain management and data mining.