Finding Surprising Patterns in a Time Series Database in Linear Time and Space

Author: Eamonn Keogh, Stefano Lonardi, and Bill ‘Yuan-chi’ Chiu

Presented by Yaohua Wu(Tony)

Introduction

- The problem of finding a specified pattern in a time series has received much attention and is now a relatively mature field.
- In contrast, the important problem of enumerating all surprising or interesting patterns has received far less attention.

Topics of Discussion

- Definition of a surprising pattern
- Feature extraction techniques to discretize time series
- Background on string processing
- Preprocessing method
- “Tarzan” algorithm
- Experimental evaluation

Definition of a Surprising Pattern

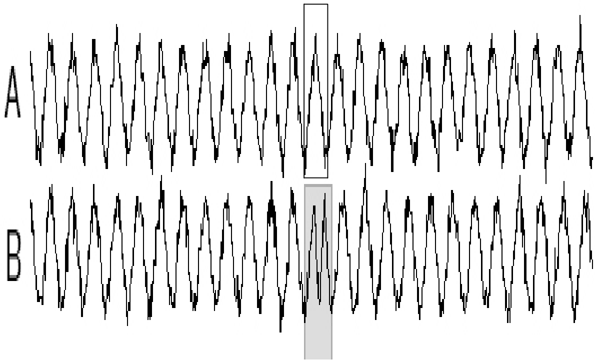
A pattern is surprising if the frequency of its occurrence differs substantially from that expected by chance, given some previously seen data

Definition of a Surprising Pattern (Continued)

Explanation (concrete definition)

- Given a reference time series database R, X is the time series database to be mined, R and X are created by the same underlying process
- A pattern P is extracted from database X
- P is surprising relative to R if the frequency of its occurrence is greatly different to that expected by chance

A Surprising Pattern



Advantages of The Definition

- An explicit definition of surprise is not required
- The user only needs to supply a collection of previously observed data, which is considered normal

Topics of Discussion

- Definition of a surprising pattern
- Feature extraction techniques to discretize time series
- Background on string processing
- Preprocessing method
- “Tarzan” algorithm
- Experimental evaluation

Paradox

The probability of a particular real number being chosen from any distribution is zero

Time Series Discretization

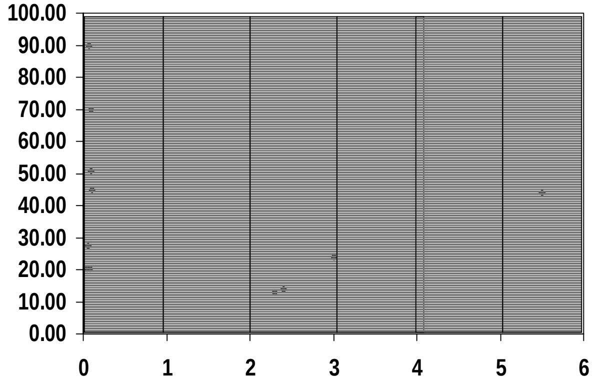
- Since a time series is an ordered list of real numbers, the paradox clearly applies
- The obvious solution to this problem is to discretize the time series into some finite alphabet Σ

How to Discretize Time Series

Given a time series database R

- First, we need the input of a feature window length and a size of the desired alphabet
- A feature window length is the length of a sliding window that moves across the time series

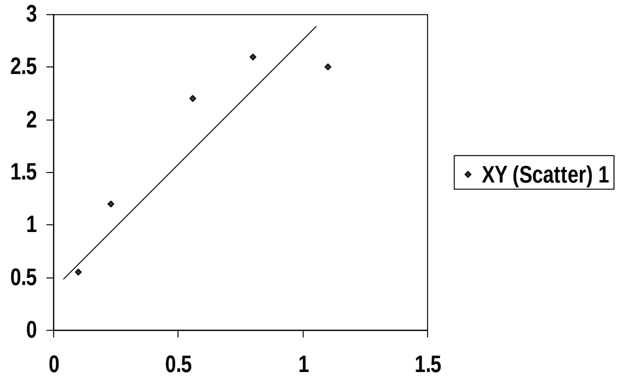
How to Discretize Time Series (Continued)



How to Discretize Time Series (Continued)

- At each time step, the portion of data falling within the window is examined, and a featured number is extracted
- In this paper, a featured number is the slope of the best-fitting line in the featured window

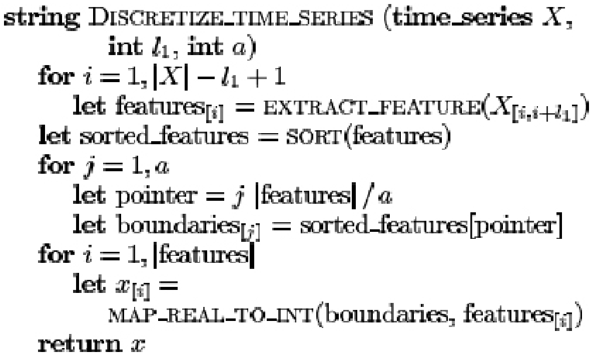
Best-fitting Line



How to Discretize Time Series (Continued)

- All the features extracted will be sorted
- Recall that the size of the alphabet is a, but the number of the features may not be equal to a
- We must make sure that each letter in the alphabet contains equal number of extracted features

Discretize_time_series



```
string DISCRETIZE_TIME_SERIES (time_series $X$,
        int $l_1$, int $a$)
  for $i = 1, |X| - l_1 + 1$
    let features$_{[i]}$ = EXTRACT_FEATURE($X_{[i, i+l_1]}$)
  let sorted_features = SORT(features)
  for $j = 1, a$
    let pointer = $j$ |features| / $a$
    let boundaries$_{[j]}$ = sorted_features[pointer]
  for $i = 1$, |features|
    let $x_{[i]}$ =
        MAP_REAL_TO_INT(boundaries, features$_{[i]}$)
  return $x$
```

Time Complexity

- The dominant step above is sorting
- Since the featured boundaries are very stable, they can be reliably estimated from a subsample of the whole data.

Time Complexity (Continued)

- For large database we can determine the feature boundaries from a subsample of size s*s = |R|
- R is the size of the database
- Since sorting takes O(s*log(s)), this feature extraction algorithm is O(|R|)

Problems

- Given two time series database r and x, we apply the discretization function to both of them, and get two strings s(r) and s(x)
- Is there any technique we can use to find the frequencies of all of the substrings of s(x) in s(r) fast?

Topics of Discussion

- Definition of a surprising pattern
- Feature extraction techniques to discretize time series
- Background on string processing
- Preprocessing method
- “Tarzan” algorithm
- Experimental evaluation

Background on String Processing

- Markov models

  Calculate the expected frequency of previously unobserved patterns

- Suffix trees

  A suffix tree is a type of digital search tree that represents a set of strings over a finite alphabet Σ

Suffix Trees

- Given a string mississippi
- How can we construct the suffix trees?

Suffix Trees (Continued)

1. Retrieve all of its suffixes

T_1 = mississippi

T_2 = ississippi

T_3 = ssissippi

T_4 = sissippi

T_5 = issippi

T_6 = ssippi

T_7 = sippi

T_8 = ippi

T_9 = ppi

T_{10} = pi

T_{11} = i

T_{12} = (empty)

Suffix Trees (Continued)

2. Sort them in order

T_{11} = i

T_8 = ippi

T_5 = issippi

T_2 = ississippi

T_1 = mississippi

T_{10} = pi

T_9 = ppi

T_7 = sippi

T_4 = sissippi

T_6 = ssippi

T_3 = ssissippi

Suffix Trees (Continued)

```
Tree →|---mississippi
      |---i→|---ssi→|---ssippi
      |     |       |---ppi
      |     |---ppi
      |---s→|---si→|---ssippi
      |     |      |---ppi
      |     |---i→|---ssippi
      |           |---ppi
      |---p→|---pi
            |---i
```

Suffix Trees in This Paper

- The leaf node of the suffix tree represents the suffix index of the substring from the root to the leaf
- The edge is labeled with a nonempty substring
- The internal node represents the frequency of occurrence of the edge label(a substring)

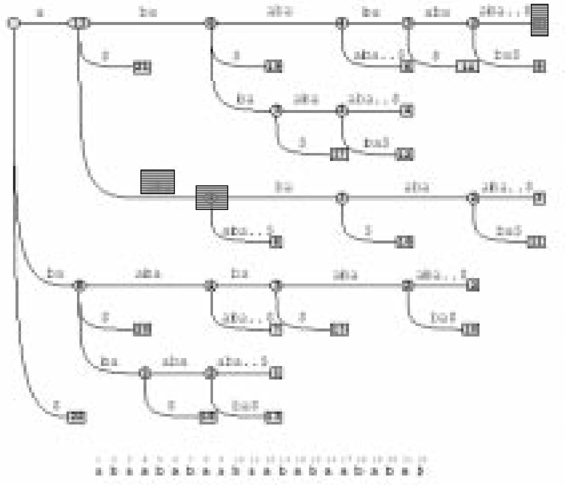
Suffix Trees in This Paper

- Consider the string abaababaabaababaababa$
- Retrieve all of its suffixes

1. abaababaabaababaababa$
2. baababaabaababaababa$
3. aababaabaababaababa$

…

Suffix Trees in This Paper



Topics of Discussion

- Definition of a surprising pattern
- Feature extraction techniques to discretize time series
- Background on string processing
- Preprocessing method
- “Tarzan” algorithm
- Experimental evaluation

Pre-process Method

- Basic idea of this method

  Build the suffix tree for the reference string and testing string, for each substring in testing string, calculate its frequency of occurrence in both reference string and testing string, and store the difference between them.

Pre-process Method (Continued)

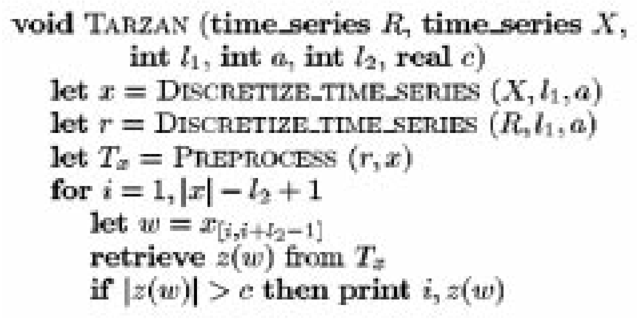
- What happen if a substring w in testing string does not occur in reference string?

  Using the Markov method to look for the longest set of strings from reference string that cover w
- Small run time and small space occupied

Topics of Discussion

- Definition of a surprising pattern
- Feature extraction techniques to discretize time series
- Background on string processing
- Preprocessing method
- “Tarzan” algorithm
- Experimental evaluation

“Tarzan” Algorithm



```
void TARZAN (time_series $R$, time_series $X$,
        int $l_1$, int $a$, int $l_2$, real $c$)
  let $x$ = DISCRETIZE_TIME_SERIES $(X, l_1, a)$
  let $r$ = DISCRETIZE_TIME_SERIES $(R, l_1, a)$
  let $T_x$ = PREPROCESS $(r, x)$
  for $i = 1, |x| - l_2 + 1$
    let $w = x_{[i, i+l_2-1]}$
    retrieve $z(w)$ from $T_x$
    if $|z(w)| > c$ then print $i, z(w)$
```

“Tarzan” Algorithm (Continued)

- R is the reference database
- X is the testing database
- L_1 is the feature window length
- a is the alphabet size for the discretization
- L_2 is the scanning window length
- c is the threshold

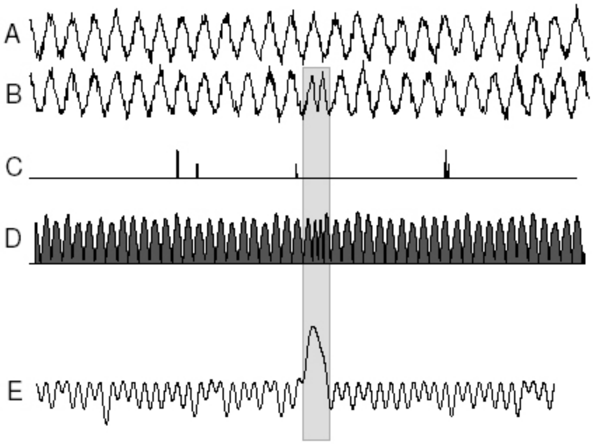
Efficiency

- Run time: O(R + X)

Topics of Discussion

- Definition of a surprising pattern
- Feature extraction techniques to discretize time series
- Background on string processing
- Preprocessing method
- “Tarzan” algorithm
- Experimental evaluation

Experimental Evaluation



Experimental Evaluation (Continued)

- A) The training data, a slightly noisy sine wave
- B) A noisy sine wave that was created with the same parameters
- C) IMM anomaly detection algorithm
- D) The TSA-Tree approach
- E) Tarzan

Conclusion

- This paper describes the definition of a surprising pattern and provides an algorithm “Tarzan” to find the surprising patterns
- It proves that “Tarzan” is efficient and has much high true positive rate and lower false positive rates

Thank you