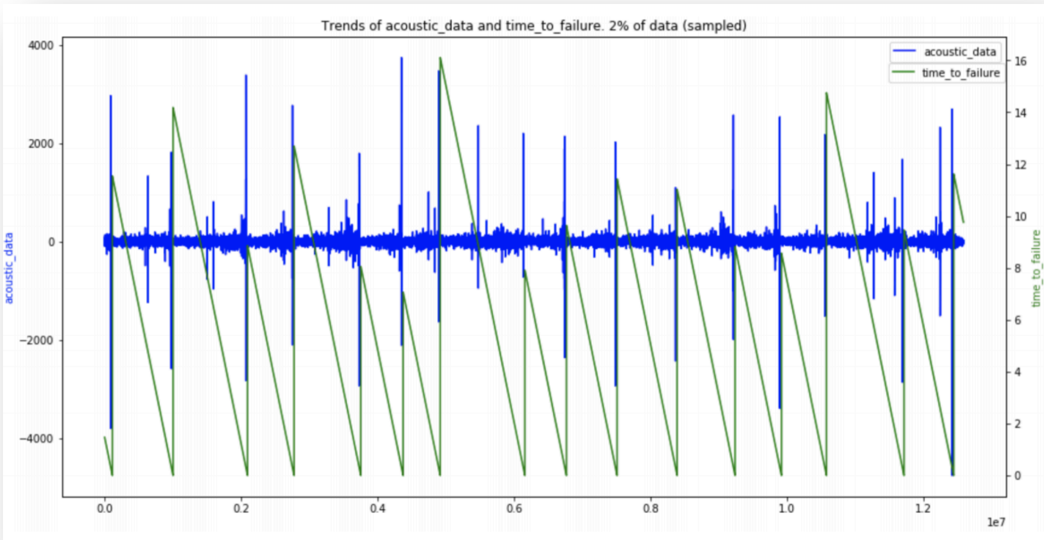
Data Mining
資料探勘

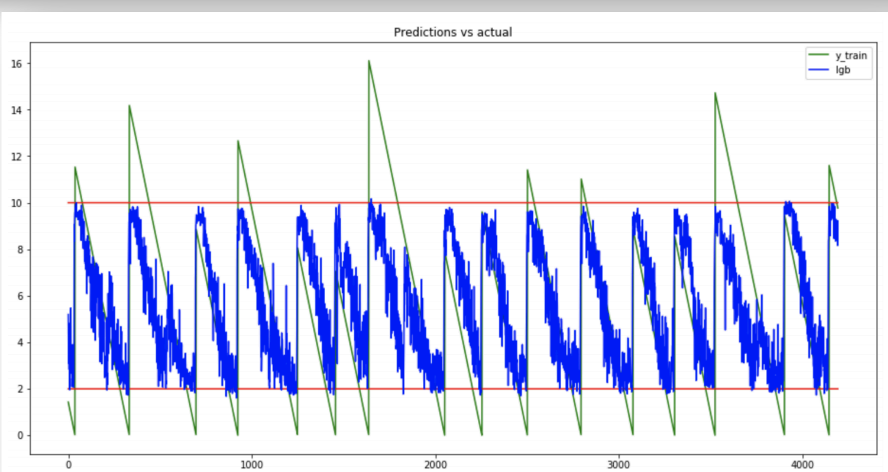
Sequential Pattern

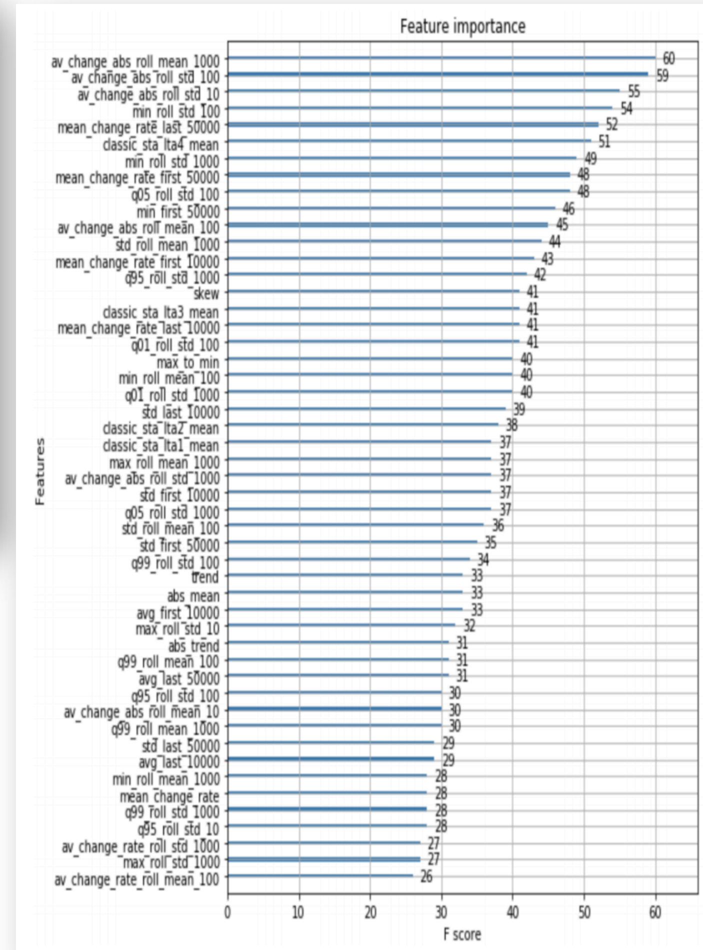
Hung-Yu Kao, Fall 2019

General time-series data

(https://www.kaggle.com/c/LANL-Earthquake-Prediction)



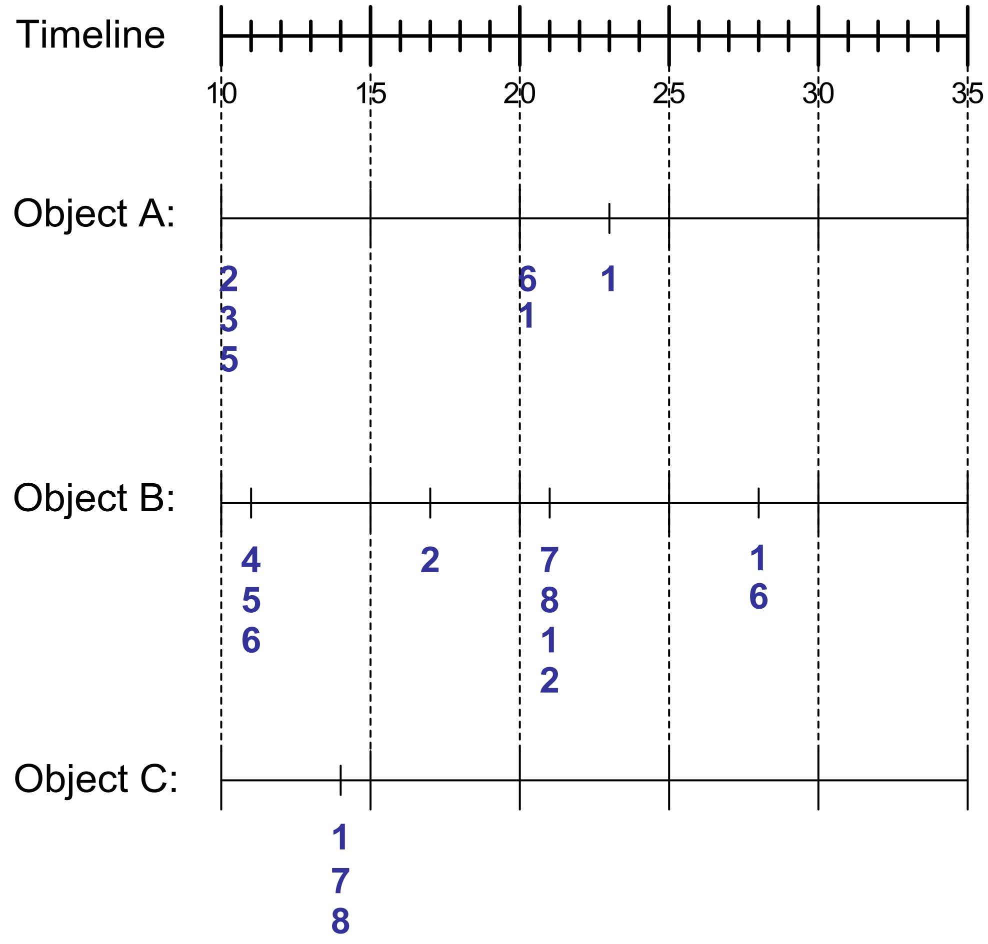




Sequence Data

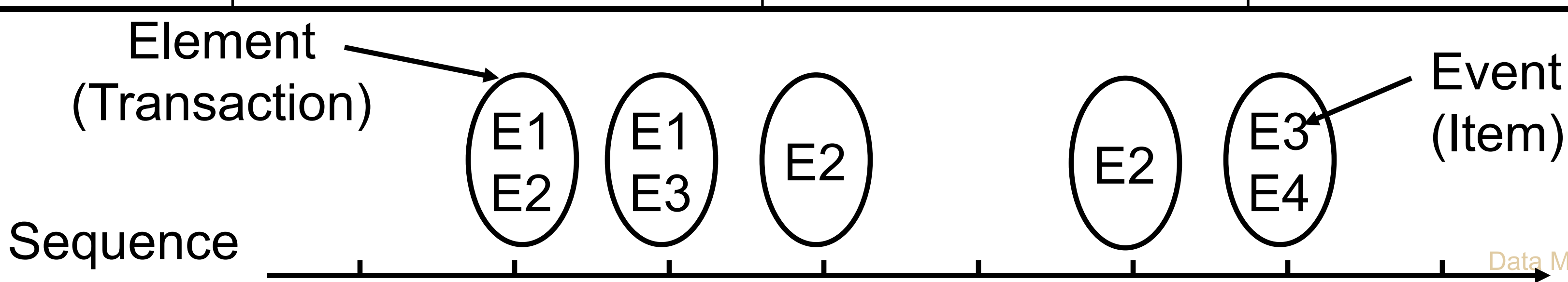
Sequence Database:

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



Examples of Sequence Data

Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	An element of the DNA sequence	Bases A,T,G,C



Formal Definition of a Sequence

- A sequence is an ordered list of elements (transactions)

$$s = < e_1 \, e_2 \, e_3 \ldots >$$

  - Each element contains a collection of events (items)

$$e_i = \{i_1, i_2, \ldots, i_k\}$$

  - Each element is attributed to a specific time or location
- Length of a sequence, |s|, is given by the number of elements of the sequence
- A k-sequence is a sequence that contains k events (items)
  - a 8-sequence of length 5 for the example in the last slide

Examples of Sequence

- Web sequence:

  < {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >

- Sequence of initiating events causing the nuclear accident at 3-mile Island:

  (http://stellar-one.com/nuclear/staff_reports/summary_SOE_the_initiating_event.htm)

  < {clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps trip} {main waterpump trips} {main turbine trips} {reactor pressure increases}>

- Sequence of books checked out at a library:

  <{Fellowship of the Ring} {The Two Towers} {Return of the King}>

Formal Definition of a Subsequence

- A sequence $<a_1\ a_2 \dots a_n>$ is contained in another sequence $<b_1\ b_2 \dots b_m>$ ($m \geq n$) if there exist integers $i_1 < i_2 < \dots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}$, …, $a_n \subseteq b_{in}$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Yes
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Yes

- The support of a subsequence w is defined as the fraction of data sequences that contain w
- ***A sequential pattern*** is a frequent subsequence (i.e., a subsequence whose support is $\geq$ *minsup*)

i1=1 i2=2 i3=4

b = {Milk,Bread}{Apples}{Sausages}{Beer,Bread}

a = {Milk}{Apples}{Bread}

What Is Sequential Pattern Mining?

- Given a set of sequences, find the complete set of frequent subsequences

A *sequence database*

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

A *sequence* : < (ef) (ab) (df) c b >

An element may contain a set of items. Items within an element are unordered and we list them alphabetically.

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

Given *support threshold* *min_sup* =2, <(ab)c> is a *sequential pattern*

Sequential Pattern Mining: Definition

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, *minsup*
- Task:
  - Find all subsequences with support ≥ *minsup*

Extracting Sequential Patterns

- Given n events: $i_1, i_2, i_3, \ldots, i_n$
- Candidate 1-subsequences:

  $<\{i_1\}>, <\{i_2\}>, <\{i_3\}>, \ldots, <\{i_n\}>$
- Candidate 2-subsequences:

  $<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots, <\{i_2\} \{i_1\}>, <\{i_2\} \{i_2\}>, \ldots, <\{i_{n-1}\} \{i_n\}>$
- Candidate 3-subsequences:

  $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots, <\{i_1, i_2\} \{i_1\}>, <\{i_1, i_2\} \{i_2\}>, \ldots,$
  $<\{i_1\} \{i_1, i_2\}>, <\{i_1\} \{i_1, i_3\}>, \ldots, <\{i_1\} \{i_1\} \{i_1\}>, <\{i_1\} \{i_1\} \{i_2\}>, \ldots$

Sequential Pattern Mining: Challenge

- Given a sequence: <{a b} {c d e} {f} {g h i}>
  - Examples of subsequences:

    <{a} {c d} {f} {g} >, < {c d e} >, < {b} {g} >, etc.

- How many k-subsequences can be extracted from a given n-sequence?

<{a b} {c d e} {f} {g h i}> n = 9

k=4: Y _ _ Y Y _ _ _ Y

<{a} {d e} {i}>

Answer :

$$\binom{n}{k} = \binom{9}{4} = 126$$

Mining Sequential Patterns

- Sequential Patterns [Agrawal, Shrikant ICDE1995]
  - Rakesh Agrawal and Ramakrishnan Srikant. "*Mining sequential patterns*". IEEE Intern'l Conf. on Data Eng.,, Mar. 1995, pp. 3-14.
  - Customer, time sequenced, not transaction
  - rents "Star Wars", then "Empire Strikes Back", then "Return of the Jedi" in that order
  - transform transactional to customer-sequenced
  - mine maximal sequence using Apriori* (AprioriAll, AprioriSome,...)
- Apriori-based SP algorithm
  - GSP (R. Srikant, R. Agrawal, "Mining quantitative association rules in large relation tables", SIGMOD 1996.)

Issues in Apriori-like sequential pattern mining methods

- A huge set of candidate sequences could be generated in a large sequence database.
- Many scan of databases in mining.
- Encountering difficulty when mining long sequential patterns.

Algorithm

- Sort phase
  - customer id (primary key), time (second primary key)
- Litemset (large itemset) phase
  - support: the fraction of customers who bought the itemset in any one of their possibly many tx's

Transaction Time	Customer Id	Items Bought
June 10 '93	2	10, 20
June 12 '93	5	90
June 15 '93	2	30
June 20 '93	2	40, 60, 70
June 25 '93	4	30
June 25 '93	3	30, 50, 70
June 25 '93	1	30
June 30 '93	1	90
June 30 '93	4	40, 70
July 25 '93	4	90

Customer Id	TransactionTime	Items Bought
1	June 25 '93	30
1	June 30 '93	90
2	June 10 '93	10, 20
2	June 15 '93	30
2	June 20 '93	40, 60, 70
3	June 25 '93	30, 50, 70
4	June 25 '93	30
4	June 30 '93	40, 70
4	July 25 '93	90
5	June 12 '93	90

Figure 1: Database Sorted by Customer Id and Transaction Time

Algorithm (cont'd)

- Transformation phase
  - each tx is replaced by the set of all litemsets contained in that tx
- Sequence phase
- Maximal phase

Customer-Sequence Version of the Database

Customer Id	Customer Sequence
1	< (30) (90) >
2	< (10 20) (30) (40 60 70) >
3	< (30 50 70) >
4	< (30) (40 70) (90) >
5	< (90) >

Large itemset Phase (support:2)

Large Itemsets	Mapped To
(30)	1
(40)	2
(70)	3
(40 70)	4
(90)	5

Transformation Phase

Customer Id	Original Customer Sequence	Transformed Customer Sequence	After Mapping
1	< (30) (90) >	< {(30)} {(90)} >	< {1} {5} >
2	< (10 20) (30) (40 60 70) >	< {(30)} {(40) (70) (40 70)} >	< {1} {2, 3, 4} >
3	< (30 50 70) >	< {(30), (70)} >	< {1, 3} >
4	< (30) (40 70) (90) >	< {(30)} {(40) (70) (40 70)} {(90)} >	< {1} {2, 3, 4} {5} >
5	< (90) >	< {(90)} >	< {5} >

Sequence Phase

- Apriori-like algorithm
- An example of Apriori candidate generation

Sequence	Support
<1 2 3>	2
<1 2 4>	2
<1 3 4>	3
<1 3 5>	2
<2 3 4>	2

→

<1 2 3 4>
<1 2 4 3>
<1 3 4 5>
<1 3 5 4>

Example

<{1 5} {2} {3} {4}>
<{1} {3} {4} {3 5}>
<{1} {2} {3} {4}>
<{1} {3} {5}>
<{4} {5}>

Customer Sequences

Sequence	Support
<1>	4
<2>	2
<3>	4
<4>	4
<5>	4

Large 1-Sequences

Sequence	Support
<1 2>	2
<1 3>	4
<1 4>	3
<1 5>	2
<2 3>	2
<2 4>	2
<3 4>	3
<3 5>	2
<4 5>	2

Large 2-Sequences

Sequence	Support
<1 2 3>	2
<1 2 4>	2
<1 3 4>	3
<1 3 5>	2
<2 3 4>	2

Large 3-Sequences

Sequence	Support
<1 2 3 4>	2

Large 4-Sequences

Sequence	Support
<1 2 3 4>	2
<1 3 5>	2
<4 5>	2

Maximal Large Sequences

Maximal Sequence

- <(3) (4 5) (8)> is contained by <(7) (3 8) (9) (4 5 6) (8)>
- <(3) (5)> is not contained in <(35)>, and vice versa
- In a set of sequences, a sequence s is maximal if s is not contained in any other sequences in the set

Notes

Step 1: SORT phase : into customer sequence

DB sorted major(customer_id), minor(transaction_time)

Step 2: Litemset Phase: support+1 / per satisfied customer

Association rule problem with support count increment difference

Step 3: Transformation Phase: transform by Litemset

Transaction transformed into contained Litemset sequence, drop useless Ti.

Step 4: Sequence Phase : **Key Algorithm**

Step 5: Maximal Phase : find max. from large sequences

for (k=n; k > 1; k--) do

for each k-sequence sk do delete from S all subsequence of sk

Rule Discovery from Time Sequences

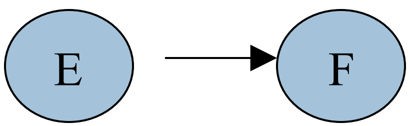
- (Das, Lin, Mannila, Renganathan, Smyth 98)
- Algorithm:
  - Cluster sliding windows
  - Label the windows in the same cluster with their cluster id
  - Generate association rule-like rules

Sequential Patterns (cont'd)

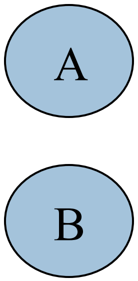
- Discovering Episodes [Mannila, Toivonen; KDD 1995 and KDD 1996]
  - Collection of ordered events within an interval
  - Web page C is accessed 2 min after A, B

Episode Mining

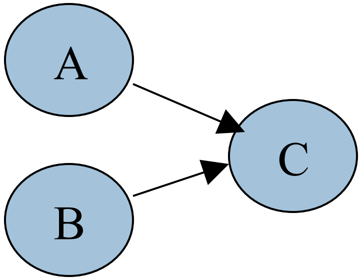
- Episode
  - A partially ordered collection of events occurring together
  - Can be described as DAG
- Serial Episode



- Parallel Episode

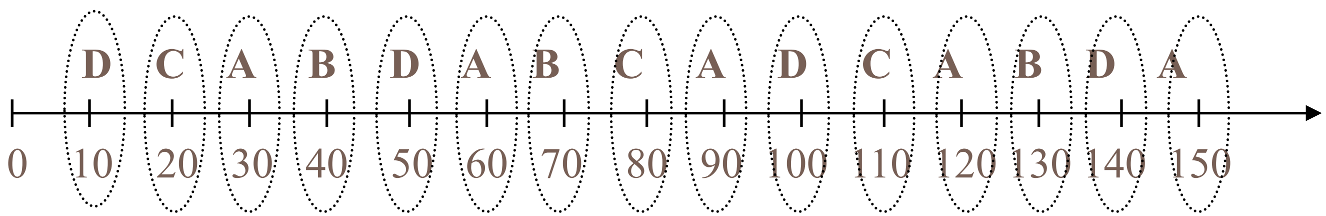


- Non-Serial and Non-Parallel



Example of Episode Mining

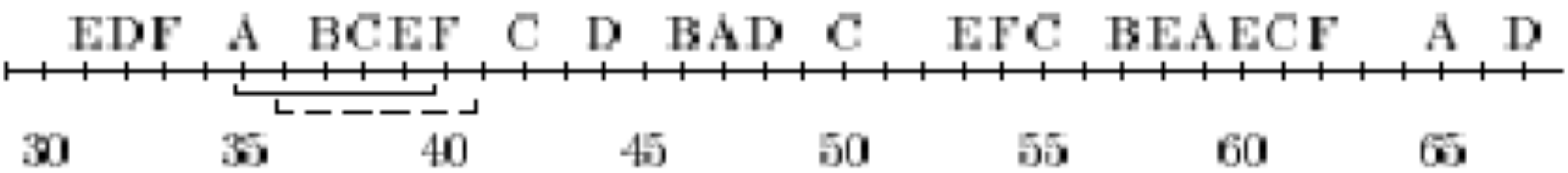
- **Alarm data sequence:**



- **Here:**
  - *A, B, C* and *D* are event (or here alarm) types
  - *10…150* are occurrence times
  - $s = \langle (D, 10), (C, 20), \ldots, (A, 150) \rangle$
  - T_s (starting time) = 10 and T_e (ending time) = 150
- **Note: There needs <u>not</u> to be events on every time slot!**

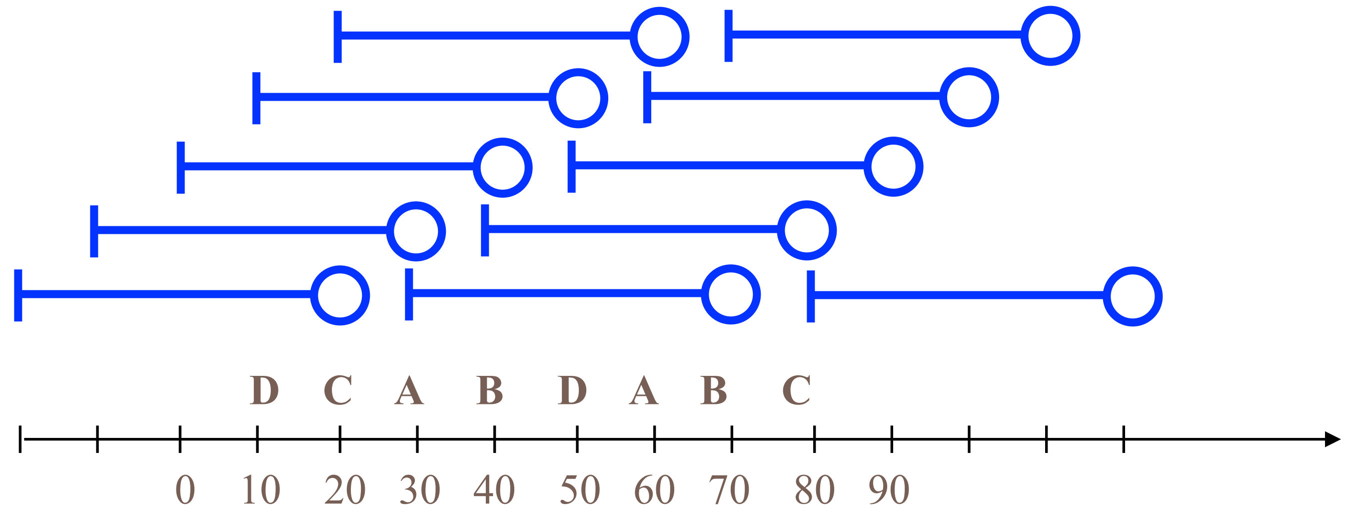
Event Sequence

- Event Sequence S=(s,29,68) s=<(E,31),(D, 32),(F33),….,(D,67)>
- Window W=(w,35,40) w=<(A,35),(B,37),(C,38), (E,39)>



Sliding Window

- **Example alarm data sequence:**



- **The window width is 40 seconds**

Frequency of an Episode

- The fraction of windows in which the episode occurs
- An episode is frequent if its frequency >= min_fr a given frequency threshold

$$fr(\alpha, S, W) = \frac{|\, S_w \in \mathcal{W}(S, W) \mid \alpha \text{ occurs in } S_w \,|}{|\, \mathcal{W}(S, W) \,|}$$

where $\mathcal{W}(S, W)$ is the set of all windows S_w of sequence S such that the window width is W

- Once the frequent episodes are known, they can be used to obtain rules

Find Frequent Episodes

- Task: discover all frequent episodes from a given class(ex. all parallel or all serial) of episodes
  - Start from the episodes with one event
  - Do a level-wise search in the episode lattice
  - On each level, compute the candidates and check their frequencies

FreeSpan

- Frequent pattern-projected Sequential pattern mining (KDD'00)
- Main Idea
  - project sequence databases into a set of smaller projected databases
  - grow subsequence fragments in each projected database
  - Divide-and-conquer approach
  - Complete set of sequential patterns can be divided into several subsets without overlaps

Example of FreeSpan

Example database: min support = 2

Sequence id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

f_list = a:4,b:4,c:4,d:3,e:3,f:3 (frequent item list, sorted)

g is deleted because of support of g <2.

Example of FreeSpan (cont'd)

- **Finding sequential patterns containing only item a**

Sequence id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<e(af)cbc>

=>

{a}-projected database

10	<aaa>
20	<aa>
30	<a>
40	<a>

Frequent Patterns
<a> <aa>

Example of FreeSpan (cont'd)

- **Finding sequential patterns containing item b but no item after b in f_list**

{b}-projected database

Sequence id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<e(af)cbc>

=>

10	<a(ab)a>
20	<aba>
30	<(ab)b>
40	<ab>

Frequent Patterns
<b> <ab> <ba> <(ab)>

Example of FreeSpan (cont'd)

- **Finding other subsets of sequential patterns.**

Sequence id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<e(af)cbc>

=>

{c}-projected database

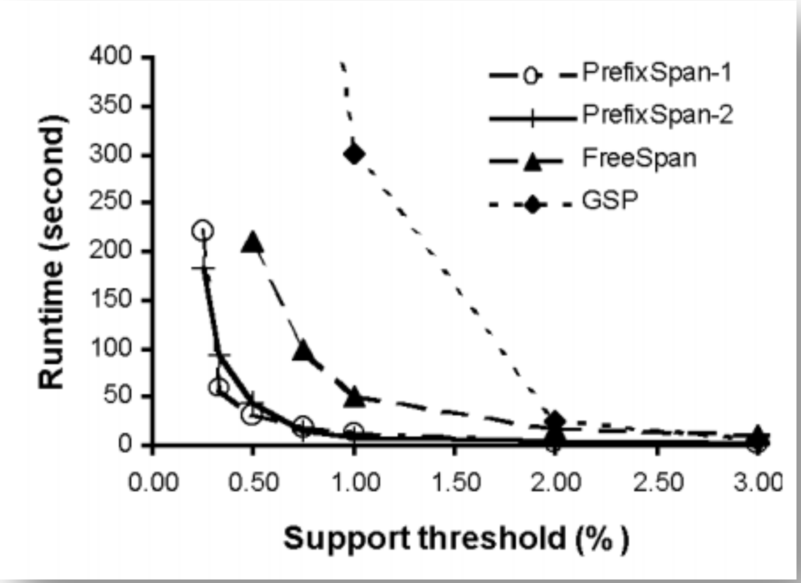
10	<a(abc)(ac)c>
20	<ac(bc)a>
30	<(ab)cb>
40	<acbc>

Frequent Patterns
<c> <ac> <bc> <(bc)> <ca> <cb>
<(ab)c> <acc> <acb>

How about {f}-projected database?

PrefixSpan

- Prefix-projected Sequential pattern mining (Jian Pei, ICDE'01)
  - Projection-based
  - Prefix-based projection: less projections and quickly shrinking sequences



PrefixSpan - Concepts

- Prefix
  - e.g. s1 = <a(abc)(ac)d(cf)>
    - The prefixes of s1 are <a>,<aa>,<a(ab)>,<a(abc)>…
    - but <ab> and <a(bc)> are not
- Projection
  - e.g. s1= <a(abc)(ac)d(cf)>
    - the projection of s1 w.r.t <bd> is <bd(cf)>

PrefixSpan – Concepts

- Postfix
  - e.g. s1= <a(abc)(ac)d(cf)>
    - The postfix of s1 w.r.t <aa> is <(_bc)(ac)d(cf)>
    - The postfix of s1 w.r.t <bd> is <(cf)>

Example of PrefixSpan

Sequence id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<e(af)cbc>

=>

<a>-projected database

10	<(abc)(ac)d(cf)>
20	<(_d)c(bc)(ae)>
30	<(_b)(df)cb>
40	<(_f)cbc>

By scanning <a>-projected database once, all the length-2 sequential patterns having prefix <a> can be found.

<aa>:2 <ab>:4 <(ab)>:2 <ac>:4 <ad>:2 <af>:2

Recursively, patterns with prefix <a> can be partitioned into 6 subsets.

Example of PrefixSpan (cont'd)

Sequence id	Sequence
10	<(abc)(ac)d(cf)>
20	<(_d)c(bc)(ae)>
30	<(_b)(df)cb>
40	<(_f)cbc>

=>

<aa>-projected database

10	<(_bc)(ac)d(cf)>
20	<(_e)>

=>

<ab>-projected database

10	<(_c)(ac)d(cf)>
20	<(_c)(ae)>
40	<c>

Sequential patterns of <ab>-projected db:
<(_c)>,<(_c)a>,<a><c>

Example of PrefixSpan (cont'd)

Sequence id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<e(af)cbc>

=>

<b>-projected database

10	<(_c)(ac)d(cf)>
20	<(_c)(ae)>
30	<(df)cb>
40	<c>

Sequential patterns
<b> <ba> <bc> <(bc)> <(bc)a> <bd> <bdc> <bf>

References

- Rakesh Agrawal and Ramakrishnan Srikant. "*Mining sequential patterns*". IEEE Intern'l Conf. on Data Eng.,, Mar. 1995, pp. 3-14.
- R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generations and Performance Improvements," Proc. 5th Int'l Conf. Extending Database Technology, Mar. 1996, pp. 3-17
- Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, Padhraic Smyth: Rule Discovery from Time Series. KDD 1998: 16-22
- Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, Meichun Hsu: PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. ICDE 2001: 215-224
- Ioan Alfred Letia, Florin Craciun, Zoltan Köpe, Alexandru Lelutiu: First Experiments for Mining Sequential Patterns on Distributed Sites with Multi-Agents. IDEAL 2000: 187-192
- Shiby Thomas, Sunita Sarawagi: Mining Generalized Association Rules and Sequential Patterns Using SQL Queries. KDD 1998: 344-348

References (cont'd)

- C.-R. Lin and M.-S. Chen, ``On the Optimal Clustering of Sequential Data," Proc. of the 2nd SIAM Intern'l Conference on Data Mining (SDM-02), April 11-13, 2002, pp. 141-157.
- Takahiko Shintani, Masaru Kitsuregawa: Mining Algorithms for Sequential Patterns in Parallel: Hash Based Approach. PAKDD 1998: 283-294
- F. Masseglia, Fabienne Cathala, Pascal Poncelet: The PSP Approach for Mining Sequential Patterns. PKDD 1998: 176-184
- P.-H. Wu, W.-C. Peng and M.-S. Chen, ``Mining Sequential Alarm Patterns in a Telecommunication Database," Workshop on Databases in Telecommunications (in conjunction with VLDB 2001), September 10, 2001