

Mining Frequent Patterns in Data Streams at Multiple Time Granularities

Chris Giannella, Jiawei Han, Jian Pei, Xifeng Yan, Philip S.Yu
2002

www-faculty.cs.uiuc.edu/~hanj/pdf/fpstm03.pdf

2008年10月08日

論文ゼミ

M1 武智環

概要

- 頻出パターンのマイニングは、広く研究されてきているが、流動的なデータだと難しい。
- 最初ほとんど確認されないパターンでも、データが集まっていけば、頻出パターンとなったりする。
- FP-Streamを用いている

既往の研究

- 頻出パターンのマイニングは多くのアルゴリズムがある
 - **Apriori** [Agrawal & Srikant 1994]
 - **FP-growth** [Han, Pei & Yin 2000]
 - **FP-tree** [Han, Pei & Yin 2000]
 - **CLOSET** [Pei, Han & Mao 2000]
 - **CHARM** [Pei, Han & Yin 2002]

パターンの定義

- パターンのカテゴリを3つに分ける
 - 頻出パターン $> \sigma$
 - $\sigma >$ 時々起こるパターン $> \varepsilon$
 - 不定期パターン

Support = 出現パターン / 扱うデータ
min_support = σ
maximum support error = ε

-
- FP-Streamを構成する2つのもの
 - メインの記録から得られる全体的な頻出パターン
 - Pattern-treeに埋め込まれた「tilted-time window」

時間の影響を受ける頻出パターンのマイニング

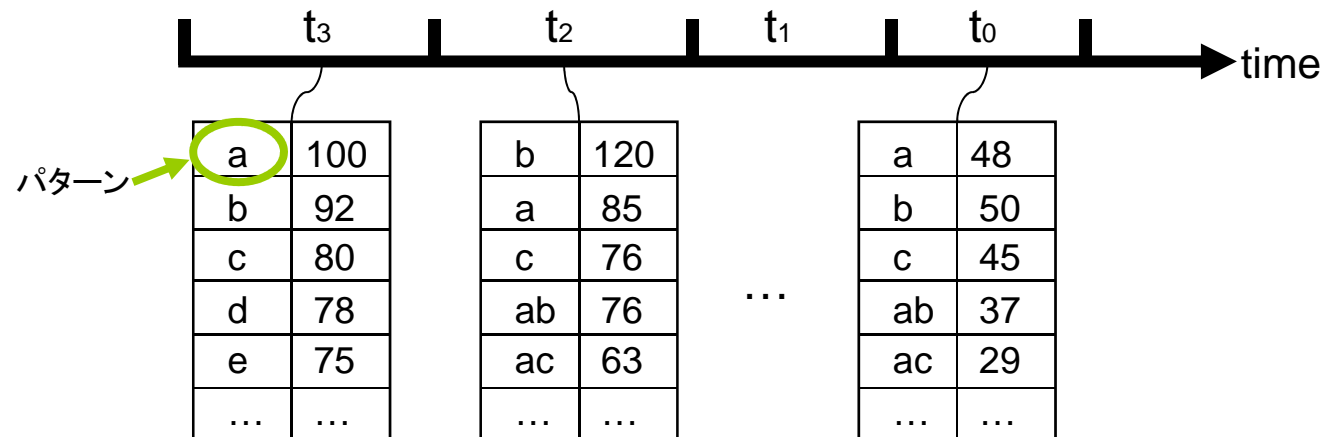
- 比較的最近の詳細なデータへの興味
- 長期間の変化をみるとデータが粗くなる
- 「今」に近いほど精度がいい
- 精度を求めると、データの期間が短く、データの長さを求めると、精度が落ちる



Natural Tilted-Time Windows Frames

Frequent Pattern for Tilted-Time Windows

- 先の時系列の中に、頻出パターンを組み合わせることで、過去の情報を保持したまま分析に使える

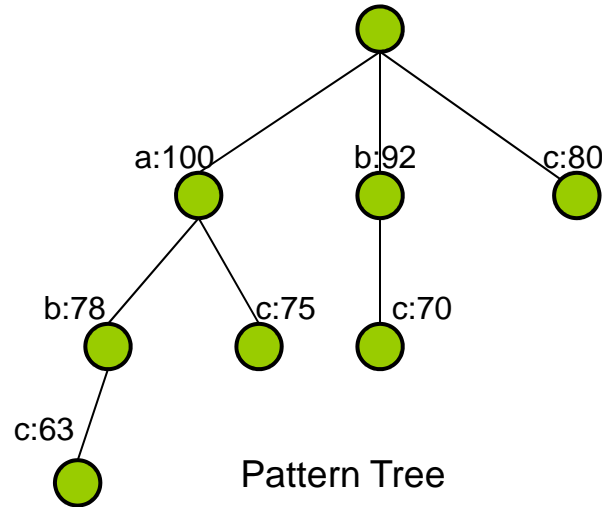


- t_2 と t_3 で頻出のパターンは何か？
- [a,b]が頻出なのはいつか？
- t_3 と t_0 の変化を裏付けるものは何か？
 - 過去のデータ同士の比較が可能
 - 頻出パターン毎に重み付けが可能

Pattern-treeの記述方法

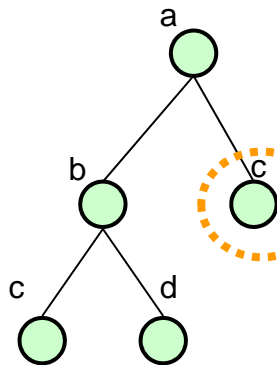
pattern	support
a	100
b	92
c	80
ab	78
ac	75
bc	70
abc	63

Frequent Pattern



Pattern Tree

- ノード毎にパターンを示している
- 頻出パターンの記述が可能



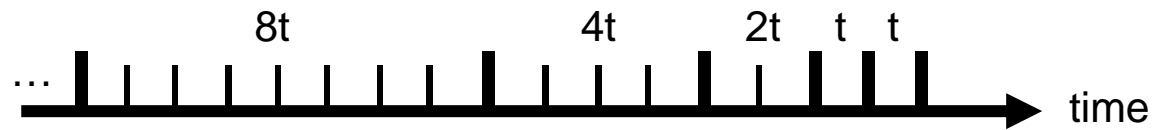
Pattern Tree

tilt window	support
t ₃	75
t ₂	63
t ₁	32
t ₀	29

Tilted-time Window Table

- それぞれのノードにTilted-time Windowsを組み込むと簡潔

Logarithmic Tilted-time Window



$$\dots\dots 8 \times \frac{1}{8}, \quad 4 \times \frac{1}{4}, \quad 2 \times \frac{1}{2}$$



$$366 \times 24 \times 4 = 35136$$



$$\log_2(365 \times 24 \times 4) + 1 \approx 17$$

$$\log_2(n) + 1$$

•大量のデータであった場合も、
対数をとることで、分析に用い
やすくなる

$$i \geq j, B(i, j)$$

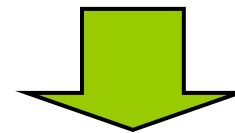
$$U_{k=j}^i B_k \quad f_I(i, j)$$

itemset Iが与えられたとき、

$B(i, j)$ の中のIの頻度

B : fixed sized batches

I : given itemset



$$f(n, n); f(n-1, n-1); f(n-2, n-3); f(n-4, n-7), \dots$$

計算方法

$$f(8,8); f(7,7); f(6,5); f(4,1)$$

Level0 Level1 Level2 Level3

B₉

$$f(9,9); f(8,8); [f(7,7)]; f(6,5); f(4,1)$$

Level0 Level1 Level2 Level3

$$f(8,7) = f(8,8) + f(7,7)$$

B₁₀

$$f(10,10); f(9,9); f(8,7); [f(6,5)]; f(4,1)$$

Level0 Level1 Level2 Level3

B₁₁

$$f(11,11); f(10,10); [f(9,9)]; f(8,7); [f(6,5)]; f(4,1)$$

Level0 Level1 Level2 Level3

B₁₂

$$f(12,12); f(11,11); f(10,9); f(8,5); [f(4,1)]$$

入力データがなくなったら終了

Algorithm

- 1. FP-treeのなかを空にする
- 2. 頻度を下げるようなitemの集合 (f_list)
 - f_listによって得られるデータを途切れることなく、FP-treeに入れていく
- 3. 全てのデータが入ったら、FP-treeを更新する
 - FP-treeは、FP-streamを使ってる

結果

Data

- Number of pattern: **10000**
- Average length of the maximal pattern: **4**
- Correlation coefficient between pattern: **0.25**
- Average confidence in a rule: **0.75**
- Support threshold: **σ** (サポート閾値)

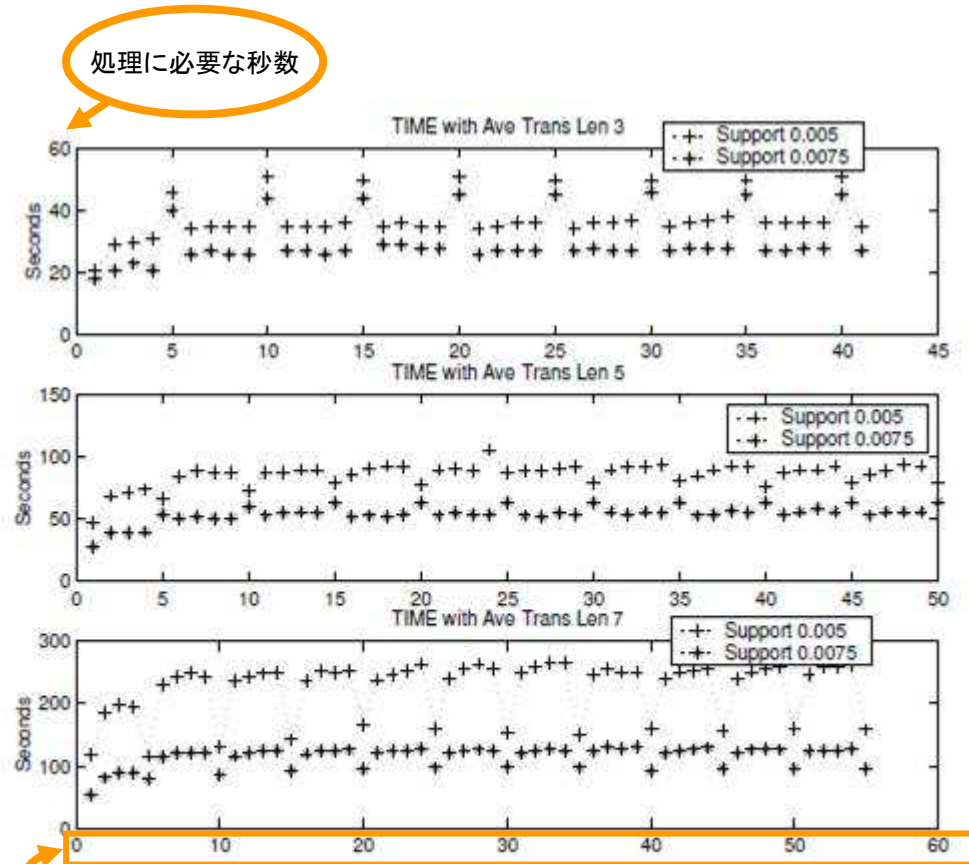


Figure 3.6: FP-stream time requirements

Batch number

まとめ

- 過去の頻度の高いパターンの情報を保持したまま、現在のデータとの分析を行った.
- FP-treeベースの、頻出パターンの抽出法を示した.