7K-06

シーケンス・マイニング・アルゴリズムの機能と性能について

氏 名† 荒井 広太 † 宇田川 佳久
所 属† 東京工芸大学工学部 コンピュータ応用学科

1. はじめに

近年,多種多様なデータを大量に蓄積することが可能になり,そのデータを有効利用することが研究テーマ(データマイニング)になってきている. 大量のデータの中から頻繁に発生するパターンを見つけ出すことは,ルールや規則性を見つけ出すため基本である.頻繁に発生するパターンを見つけ出す方法として, シーケンス（シーケンス）マイニングがある. 本文では, 公開されている シーケンス・マイニング・アルゴリズムである PrefixSpan （頻出 シーケンス）, CloSpan（飽和頻出 シーケンス）, MaxSp（極大頻出 シーケンス）アルゴリズム[1]について、機能と性能について報告する.

2. シーケンスマイニングについて

1993 年に Agrawal らが提起した頻出する集合を列挙する「頻出集合列挙問題」の研究から開発された apriori アルゴリズムがある. Apriori アルゴリズムでは,要素の集合を対象とし,要素が発生する順番は考慮していない.

Apriori アルゴリズムの原理は、要素の シーケンス(要素の順番)を考慮した シーケンスマイニングにも適用できる。探索の方法としては,要素 1 回から始めて,頻出パターンか否かを判定し, 頻出パターンであればさらに要素を 1 つ追加し, 頻出パターンか否かを判定するという処理を繰り返す.

3. 頻出・飽和・極大頻出パターンについて

図 1 に示したシーケンスデータを例にして、頻出・飽和・極大頻出パターンについて,以下に説明する. 図 2 は、Minsup を 50%(2 回以上)と設定した場合の lattice(束)構造で表している.

頻出(PrefixSpan)とは,Minsup(頻出か否かを判断する値)を設定し,条件を満たしているシーケンスを出力する. 図 1 の例では, 2 回以上出現した要素 A, B, C, AB, BC が頻出パターンとなる. 要素 AC と要素 ABC は出現回数が 1 のため頻出パターンではない.

頻出パターン fp に要素を 1 つ加えたパターンの集合を{fp'}と表記する。また、出現回数を、それぞれ、|fp|、|fp'|と表記する。すべての x ∈{fp'} に対し|fp|＞|x|であるとき、fp は飽和頻出パターン(closed frequent sequence pattern)と定義される[2]。

図 1 の例では,要素 A と要素 AB では出現回数が同じため,要素 A は飽和にならない. 次に,要素 AB と要素 ABC の出現回数に着目すると要素 AB は 2 回,要素 ABC は 1 回であり,出現回数が異なる.この場合,要素 AB は飽和(AB)になる.要素 B と要素 C も同じ処理をすると要素 B, C, AB, BC が飽和となる.

頻出パターン fp に要素を 1 つ加えたパターンの集合{fp'}のすべてが頻出パターンでないとき、fp は極大頻出パターン(maximal frequent sequence pattern)と定義される[2]。図 1 の例では,要素 ABC は出現回数が 1 回のため頻出パターンではない.つまり極大頻出パターンではない. 要素 ABC が極大頻出ではないため,1 つ前の要素 AB, BC に着目する.要素 AB, BC の出現回数は 2 回であり,頻出である.つまり,要素 AB, BC が極大頻出(*AB*, *BC*)となる.また,要素 AB, BC は飽和でもあるため,飽和かつ極大頻出(*AB*, *BC*)となる.

TID	Sequence data
1	A→B→C
2	A→B
3	C
4	B→C

図 1.シーケンスデータの例

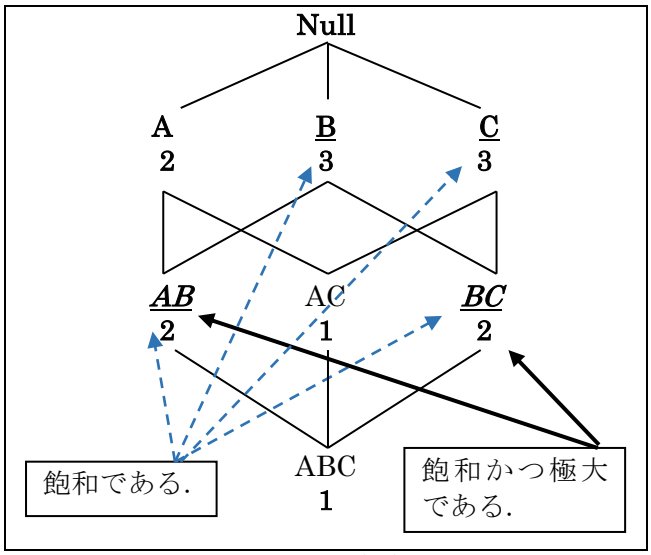

図 1. Lattice(束)構造

4.データ変換

頻出　シーケンス抽出アルゴリズムである PrefixSpan, CloSpan, MaxSp は汎用的なアルゴリズムであり、　シーケンスデータを構成する要素は番号で表記する。また、１つの　シーケンスデータの終わりは -1、データの終わりは -2 で表記する。このため、実験では、このフォーマットに変換プログラムを作成した.

今回の実験で使用した　シーケンスデータは、*Java SDK 1.7.0.45* の *lang* パッケージのソースプログラム（67,677 行）から制御文とメソッドの呼び出しを、メソッド単位に抽出したものである。制御文またはメソッドの呼び出しは、シーケンスデータの１要素に対応しており、要素の総数は 1,8205 個である. ソースプログラムを構成するメソッド数は、　シーケンスデータの行数に対応し、2,522 行である。また、　シーケンスデータを構成する要素の種類は、制御文とメソッドの呼び出しの種類に対応し、1,286 種類である。

図 2 はフォーマット変換前の　シーケンスデータの例である. 図 2 のメソッドを 10 進数の数値に変換した　シーケンスデータが図 3 である.

```
AbstractStringBuilder::expandCapacity(int minimumCapacity) → { → if{ → } → if{ → if{ → OutOfMemoryError()→}→}→Arrays.copyOf()→}
```

図 2. フォーマット変換前のシーケンスデータ

```
5 -1 3 -1 5 -1 5 -1 8 -1 3 -1 3 -1 9 -1 -2
```

図 3. フォーマット変換後のシーケンスデータ

5.実験・結果

実験として,PrefixSpan,CloSpan,MaxSp の性能検証をした.実験では,Minsup を 0.09(227 個)から 0.02(50 個)と設定している。

図 4 は PrefixSpan,CloSpan,MaxSp の処理時間の実験結果である.実験から,CloSpan の処理時間が 1 秒未満であり,PrefixSpan と MaxSp より処理が早いという結果が出た.また,PrefixSpan と MaxSp の処理時間は近似しており,処理時間は,ほぼ同じ処理時間という結果が出た.

図 5 は PrefixSpan,CloSpan,MaxSp から抽出されたパターン数の実験結果である.実験から,MaxSp から抽出されたパターン数は CloSpan の約 5 分の 1 の数という結果が出た.

Minsup	PrefixSpan	CloSpan	MaxSp
50	162s	0.272s	146s
75	40s	0.207s	33s
100	14s	0.141s	12s
126	5s	0.098s	4s
152	2s	0.073s	2s
177	1s	0.083s	1s
202	0.936s	0.075s	0.756s
227	0.515s	0.073s	0.426s

図 4.処理時間

Minsup	PrefixSpan	CloSpan	MaxSp
50	2062	1672	290
75	741	660	113
100	375	342	59
126	204	195	36
152	139	133	28
177	99	94	19
202	74	69	13
227	49	49	12

図 5.抽出されたパターン数

6. おわりに

本文では,頻出　シーケンス抽出アルゴリズムである PrefixSpan,CloSpan,MaxSp の抽出機能と性能検証について実験し,その結果について述べた.実験結果からそれぞれの処理時間では,CloSpan が一番早く,抽出されたパターン数の大きさは,PrefixSpan>CloSpan>MaxSp ということも確認できた.

参考文献

[1] “An Open-Source Data Mining Library,” http://www.philippe-fournier-viger.com/spmf/index.php, August 2015.

[2] Tan, P. N., Steinbach M., and Kumar V.: Introduction to Data Mining, pp.1-769, Pearson Addison - Wesley, 2006.

[3] 宇田川佳久:極大頻出　シーケンスマイニングを用いた Java コードクローンの検出,信学技報, vol.115, no.353, pp.11-18, 2015 年 12 月.

Functionality and performance comparison of sequence mining algorithms

† Kouta Arai: Tokyo Polytechnic University
† Yoshihisa Udagawa: Tokyo Polytechnic University