D-032

データマイニング実行回数軽減に関する提案

When to Stop the Mining Process for an Application Environment

李 思瑶† 王 家宏‡ 児玉 英一郎‡ 高田 豊雄‡
Siyao Li Jiahong Wang Eiichiro Kodama Toyoo Takata

1．まえがき

近年，データマイニングに関する数多くの研究が広く行われている．しかし，既存の研究は頻出パターンをいかに速く抽出するかに焦点をあてており，対象となるシステムに存在する頻出パターンのほとんど全てを得るまでにデータマイニングをどれだけ実行しなければいけないかに関する研究はまだ行われていなかった．

Apriori 法や FP-Growth 法[1]といったよく知られている既存のデータマイニング手法の大半は，与えられたデータベース全体を処理対象としており，そのなかに存在する全ての頻出パターンを抽出することを目的としている．これらの手法は，データの追加によりデータベースの更新が行われると，その度，データベース全体を処理しなければならず効率が悪い．そのため，既存のマイニング結果を利用し，追加データ分を中心に分析し，全データベースにわたる頻出パターンを抽出する手法がいくつか提案されている[2,3,4].

本研究では，ある種のシステムにおける頻出パターンマイニングを対象している．本研究でターゲットとするシステムは，頻出パターンの集合が一定の集合に収束するといった性質を持つものである．すなわち，データベースがある時点まで成長すると，頻出パターンがもうほとんど全て抽出済みとなり，その後追加されたデータは，頻出パターンの抽出に影響を与えない．このようなシステムに対し，最低限どの時点までデータマイニングを行わなければいけないかを明らかにすることが求められるが，まだ十分な解決法は知られていない．そこで本研究では，システムが安定するようになった時点を発見し，その時点までの頻出パターンを抽出する手法を提案する．

2．安定システムとは

本節では，ある建物における温度や湿度などのセンサーからなるセンサーネットワークという実例を用い，本研究の動機を明確にすると共に，安定システムの定義を述べる．

このシステムでは，建物に管理センターがあり，管理センターにデータベースが設置されている．センサーネットワークからの観測データは定期的にデータベースに保管される．建物の管理者は，そのデータベースに蓄積されたデータを分析し，建物の正常状態を示す温度や湿度に関する正常な頻出パターンを抽出し，建物の異常状態の検知に役立てる．こういった状況では，管理者は，正常な頻出パターンをいつまで抽出し続ければ良いかを知る必要性が生じる．この答えの一つとして，システムが以下の状態に入ったら中止するという方法が挙げられる．

- 各センサーの感知したデータの種類が一定となった．
- 頻繁に感知されたデータの集合が一定となった．
- 関連性のあるデータの種類の集合が一定となった．

Definition1(安定システム) 少しあけるデータが時間とともに増え続けるデータベースを持つシステムにおいて，データが追加されるたびに，そのシステムの頻出アイテムの集合と頻出パターンの集合を抽出する状況を考える．このとき，以下の２つの条件が満たされた時点で，そのシステムが安定した状態に入ったといい，安定した状態にあるシステムを安定システムと呼ぶ．

- 頻出アイテムの集合が変化しなくなった．
- 頻出パターンの集合が変化しなくなった．

3. 安定システムの頻出パターン抽出

本節では，システムが安定するようになった時点を発見し，それまでの頻出パターンを抽出する手法を提案する．本提案手法を以下に示す．

3.1 提案手法

Input：データベース B_0，データベース B_1

Output：安定したデータベース B_0，頻出アイテム FI_0，安定した頻出パターン FP_0．$<B_0, FI_0, FP_0>$

1. Initialize
   (1) B_0 (resp. B_1) ← the initial (resp. next) batch;
   (2) FI_0 (resp. FI_1) ← the set of all the frequent items in B_0 (resp. B_1);
2. While ($FI_0 \neq FI_1$) {
   (1) $B_0 \leftarrow B_0 + B_1$;
   (2) FI_0 ← the set of all the frequent items in B_0;
   (3) B_1 ← the next batch;
   (4) FI_1 ← the set of all the frequent items in B_1;}
3. FP_0 (resp. FP_1) ← the set of all the frequent patterns in B_0 (resp. B_1);
4. While ($FP_0 \neq FP_1$) {
   (1) $B_0 \leftarrow B_0 + B_1$;
   (2) Calculate the new FP_0 as follows:
      ① Identify the following three kinds of item sets:
         A) $S_{unchanged}$ ← The frequent patterns that remain to be supported;
         B) S_{NewFP} ← The newly-supported frequent patterns;
         C) S_{IFP} ← The frequent patterns that supported in the previous B_0 but not in B_1 ;
      ② Merge $S_{unchanged}$ with the corresponding ones in the previous FP_0, and put the results to the new FP_0;
      ③ If $P \in S_{NewFP}$, then calculating its support in the previous FP_0 to know whether to put it to the new FP_0;
      ④ If $P \in S_{IFP}$, then according to its support in the previous FP_0 to know whether to put it to the new FP_0;
   (3) B_1 ← the next batch;
   (4) FP_1 ← the set of all the frequent patterns in B_1;}
5. Take $< B_0, FI_0, FP_0 >$ as the stable database;

†岩手県立大学大学院ソフトウェア情報学研究科
‡岩手県立大学ソフトウェア情報学部

3.2 提案手法の分析

基本的な考えは以下の通りである：頻出パターンが抽出漏れする可能性を少なくするために，データベースへのデータの追加は，現在のデータベースサイズの倍数で行われる．また，できるだけ、前回の結果を再利用するために，３種類の頻出パターン，$S_{unchanged}$，S_{NewFP}，S_{IFP} を利用する．それで，最新の頻出パターンを計算する際に，再度データベーススキャンは必要ではなく，計算効率は高くなると考えられる．最後の結果は，$<B_0, FI_0, FP_0>$となる．$S_{unchanged}$，S_{NewFP}，S_{IFP} の意味を説明するために，以下の例をあげた．

B_0，B_1 の頻出アイテム(FI_0, FI_1)：{a,c,e,f,g,h,k,m,l}

B_0 の頻出パターン(FP_0)：{a,c,f}，{a,h,k,m}，{a,f,g,m} {f,k,m,l}，{k,m,l,p}

B_1 の頻出パターン(FP_1)：{a,c,f}，{a,h,k,m}，{f,k,m,p} {c,k,m,l}

このとき，

$S_{unchanged}$：{a,c,f}，{a,h,k,m}

S_{NewFP}：{f,k,m,p}，{c,k,m,l}

S_{IFP}：{a,f,g,m}，{f,k,m,l}，{k,m,l,p}

のようになったと仮定する．

この場合 $S_{unchanged}$ は直接に新しい FP_0 に挿入する．S_{NewFP} は新しい FP_0 に追加するかどうかは，現時点の S_{NewFP} の出現数と前の B_0 のスキャンの結果によって決まる．同様に，S_{IFP} を新しい FP_0 に追加するかどうかは，現時点の S_{IFP} の出現数と B_1 のスキャンの結果によって決まる．

4. 実験

以上提案した手法の性能を評価するために，その実装を行った．現在，性能評価に関する実験を行っている．本節はその実装の動作確認について述べる．

4.1 実験用データ

データベースへ追加されるデータは，図 1 に示す手動で作った２つのファイル（TestData1.dat – TestData2.dat）とその２つのファイルを利用して生成したファイル TestData3.dat と TestData3.dat である．

TestData1.dat

```
5 101 102 103
5 3 8 102 103
5 3 8 101 104
5 3 8 13 2 101 102 107
5 3 8 13 2 101 102 107
5 6 4 17 107
5 6 4 17 107 103
5 6 4 17 107 108
6 13 2 101 107 104
6 13 2 1 104 102 103 101
6 3 17 8 1 104 103 107
6 3 17 1 101
```

TestData2.dat

```
5 101 102 103
5 3 8 102 103
5 3 8 101 104
5 3 8 13 2 101 102 107
5 3 8 13 2 101 102 107
5 6 4 17 107
5 6 4 17 107 103
5 6 4 17 107 108
5 13 2 101 107 104
6 13 2 1 104 102 103 101
6 3 17 8 1 104 103 107
6 3 17 1 101
```

```
107 17 4 5 6 (12)
101 102 13 2 (12)
101 107 13 2 (12)
101 104 (12)
102 3 5 8 (12)
101 3 5 8 (12)
107 3 8 (12)
102 103 (12)
103 6 (12)
103 5 (12)
101 102 5 (12)
```

図3 実行結果

図1 TestData1.datとTestData2.dat

図2 データの追加とデータベースの構成

TestData1.dat は最初のデータで，TestData2.dat は TestData1.dat と１ヵ所だけが違う．TestData3.dat は TestData1.dat と TestData2.dat の合併であり，TestData4.dat はこの３つのファイルから構成された．

データベースへのデータ追加の流れ及びデータベースの構成は図 2 に示す．このデータベースの構成では，理論的に安定した頻出パターンを抽出することができる．

4.2 結果と考察

本実験では，TestData1.dat と TestData2.dat の頻出パターンを等しくなくしたから，TestData2.dat が TestData1.dat に追加され，$S_{unchanged}$、S_{NewFP} と S_{IFP} を通して，この TestData1.dat と TestData2.dat との連結でできたデータベース（TestData1+2.dat と呼ぶ）の頻出パターンが計算された．その後，TestData1+2.dat の頻出パターンと TestData3.dat の頻出パターンを比較し，違いがあったため，TestData1+2.dat の後に TestData3.dat のデータが追加され，TestData1+2+3.dat ができ，前と同じように，頻出パターンが計算された．最後，TestData1+2+3.dat の頻出パターンと TestData4.dat の頻出パターンを比較し，同じになったため，データの追加が止まった．そのときの実行結果は図 3 に示す．

TestData4.dat のデータは TestData1.dat － TestData3.dat の集合である．図 3 に示した本提案手法で抽出した頻出パターンの集合と TestData4.dat から直接に抽出したパターンと比較して，同じであることが分かった．その結果，本提案手法が確認されたと考える．TestData3.dat が追加された時点でこのシステムが安定システム状態に入った．

5. まとめ

本稿では，安定システムの概念を定義し，システムが安定状態に入った時点と，そこまでの頻出パターンを抽出する手法を提案した．本提案手法を利用することによって，ユーザはシステムがいつ安定状態に入ったかを把握できるようになり，そこまで抽出した頻出パターンを安心して利用できるようになり，データマイニング実行回数を軽減できると考える．

実験結果は，本提案手法の正しさを示した．今後，より詳しい性能評価をする予定である．

参考文献

[1]j. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proc. of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD00), pp. 1--12, 2000.

[2]C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In Proc. of the NSF Workshop on Next Generation Data Mining, November 2002.

[3]H. Cheng, X. Yan, and J. Han. Incspan: incremental mining of sequential patterns in large database. In Proc. of the 10th International Conference on Knowledge Discovery in Databases, pp. 527--532, 2004.

[4] G. Li, J. Feng, J. Wang, Y. Zhang, and L. Zhou. Incremental Mining of Frequent Query Patterns from XML Queries for Caching. In Proc. of the Sixth IEEE International Conference on Data Mining (ICDM06), pp. 350--361, 2006.