3I4-2

サンプリングを用いた精度保証つき頻出パターンマイニング

Frequent pattern mining using sampling with guaranteed accuracy

馬場祥人[*1] 杉山麿人[*1*2] 鷲尾隆[*1]
Yoshito Baba Mahito Sugiyama Takashi Washio

[*1]大阪大学産業科学研究所
The Institute of Scientific and Industrial Research, Osaka University

[*2]科学技術振興機構
JST PRESTO

We propose a probabilistic approach to frequent itemset mining using sampling with accuracy guarantees. The aim of frequent itemset mining is to enumerate all frequent itemsets in a transaction database, which has been applied in various disciplines such as business and genome analysis. However, a fast and memory efficient method is required to analyze recently appearing massive databases over several terabytes, in which existing methods take too exhaustive to enumerate all frequent itemsets as they cannot even load such databases on the main memory. In contrast to such exhaustive approaches, our method finds frequent itemsets via repeating sampling and does not need to load the entire database. Moreover, we theoretically bound the ratio of false positives and false negatives of frequent itemsets.

1. 序論

頻出パターンマイニング [Agrawal 14] は，データマイニングの中でも重要な問題の 1 つであり，これまで様々な研究が進められてきた．特に，頻出アイテム集合発見問題では，データベース中に頻繁に同時に現れるアイテムの組合せを発見することを目的とし，例えば，同時に購入されやすい傾向にある商品の組合せの発見 [Agrawal 94, Han 00] や，ある特定の特徴を示す遺伝子の組合せの特定 [Llinares-López 15, Terada 13] など，多様な分野で応用されている．しかし，近年出現している高次元かつ大規模なデータでは，容量が数テラバイト以上になることもあり，全データを一度に主記憶上に読み込むことが困難である．このため，既存の頻出アイテム集合発見手法ではこれらの大規模データを扱えない場合があり，また逐次的にデータを読み込む手法（例えば [Otey 04]）を用いても解析に時間がかかってしまう．

そこで，本稿では，ランダムサンプリングを用いて大規模データから効率的に頻出アイテム集合の候補を見つける手法を提案する．提案手法では，データ中からランダムに選択したトランザクション同士の比較に基づき頻出アイテム集合を発見するため，サポートの計算が必要なく，データを全て読み込む必要がない．そのため，全データを主記憶上に保持する必要がなく，大規模データに対しての適用が可能である．提案手法では，必要となるサンプリングの回数を理論的に求めることで，頻出アイテム集合の検出力を高く維持する．さらに，頻出アイテム集合の候補からなるバッチを複数生成してそれらを統合することで，頻出でないアイテム集合を除外し，偽陽性の割合を低く維持する．最小サポートよりも小さいサポートとして下限サポートを導入して，この下限サポートよりもサポートが小さいアイテム集合を発見してしまう確率を理論的に制御する．

本論文は，以下のように構成されている．第 2 節で頻出アイテム集合発見問題を定式化し，第 3 節で関連研究について述べる．第 4 節で提案手法のアルゴリズムを紹介して，頻出アイテム集合を発見する確率を理論的に解析する．第 5 節で提案手法を実験によって評価し，結果を考察する．最後に，第 6 節でまとめと今後の課題について述べる．

連絡先: 馬場祥人，大阪大学産業科学研究所，〒 567-0047 大阪府茨木市美穂ヶ丘 8-1，06-6879-8540，baba@ar.sanken.osaka-u.ac.jp

2. 問題の定式化

ある有限集合 A について，A の各要素 $i \in A$ を**アイテム**（item）と呼び，A の部分集合 $F \subseteq A$ を**アイテム集合**（itemset）と呼ぶ．**トランザクション**（transacition）T はアイテム集合である．頻出アイテム集合発見問題において，解析の対象となるデータは，**トランザクションデータベース**（transaction database）と呼ばれるトランザクションの有限集合 $\mathcal{D} = \{T_1, T_2, \ldots, T_n\}$ である．あるアイテム集合 F について，データベース $\mathcal{D}$ における F を含むトランザクションの割合を**サポート**（support）と呼び，F のサポート $S(F)$ は以下の式(1)で定義される．

$$S(F) = \frac{|\{T \in \mathcal{D} \mid F \subseteq T\}|}{|\mathcal{D}|} \quad (1)$$

サポート $S(F)$ は，アイテム集合 F が頻出であるかどうかの判定に用いる指標である．具体的には，この値が**最小サポート**（minimum support）$minsup$ と呼ばれるユーザーが与える閾値以上となるとき，この F を**頻出アイテム集合**（frequent itemset）と呼ぶ．与えられた $\mathcal{D}$ から $minsup$ 以上のサポートを持つ頻出アイテム集合を全て列挙する問題のことを，頻出アイテム集合発見問題と呼ぶ．

本研究では，全ての頻出アイテム集合のうち，特に**極大頻出アイテム集合**（maximal frequent itemset）と呼ばれるアイテム集合を発見することを目的とする．極大頻出アイテム集合とは，任意のアイテム $a \in A$ に対して $S(F \cup \{a\}) < minsup$ となるアイテム集合 F のことを指す．極大頻出アイテム集合の部分集合は，必ず頻出アイテム集合となることが分かっている．

3. 関連手法

頻出アイテム集合マイニング手法では，まず主要な手法の 1 つとして，Apriori 法 [Agrawal 94] が挙げられる．Apriori 法は，頻出でないアイテム集合の上位集合は頻出ではないという性質に着目し，初めて効率的な頻出アイテム集合発見を可能にした手法である．この性質は，以降の頻出アイテム集合マイニング手法でも頻繁に利用されている．

既存の頻出アイテム集合発見手法の中で，最速な手法の一つに LCM [Uno 04] が挙げられる．LCM は，配列とビットマップ，接頭辞木を組み合わせて頻出アイテム集合を発見する手法で，FIMI04 と呼ばれる頻出アイテム集合マイニングのコンペティションで優勝した手法でもある．本研究では，提案手法と LCM の出力を比較し，提案手法の精度，効率について検証する．

上に述べた手法以外にも，既存の手法の多くはデータベース $\mathcal{D}$ 全体を主記憶上に格納する必要があるため，トランザクション数が非常に大規模な場合，一度に $\mathcal{D}$ 全体を読み込むことができない．このために，$\mathcal{D}$ 中のトランザクションを逐次的に繰り返し読み込む必要があり，効率性が損なわれることになる．また，既存手法は $\mathcal{D}$ が疎である，すなわち，各トランザクションに属するアイテム数がアイテムの総種類数に比べて小さいという仮定で高速化を達成しているため，高次元かつ密なデータベースに対しては効率性が著しく低下してしまう．

一方で，より高速な解析のため，データベースの一部をサンプリングにより抽出し，そこから頻出アイテム集合を発見する手法も提案されている．代表的な手法として，Random Intersection Trees [Shah 14] (以下 RIT と記述) がある．RIT では，データベースからランダムにサンプリングされたトランザクション同士を比較し，その共通部分から頻出アイテム集合を近似的に求めている．RIT におけるトランザクションの比較の際には，多数の木構造が用いられる．まず，根ノードとして $\mathcal{D}$ からランダムに選んだトランザクション T を一つ格納する．次に，新たにトランザクション $T_1, T_2, \ldots, T_k$ を $\mathcal{D}$ からランダムに抽出し，それぞれについて，T との共通部分集合 $T \cap T_i$ を子ノードに格納する．この操作を繰り返し，最後に葉ノードに格納されたアイテム集合を近似的な頻出アイテム集合として取り出す．本研究では，この手法に着想を得て，サンプリングとトランザクション同士の比較により頻出アイテム集合を取り出す手法を提案する．

4. 提案手法

本研究では，ランダムサンプリングによって抽出されたトランザクション同士の共通部分を取得することで，確率的に最小サポート以上の極大アイテム集合を求める手法を提案する．節 4.2で示すように，提案手法の計算量は $\mathcal{D}$ の密度に依存しない．このため，多くの頻出アイテム集合発見手法が疎なデータに対してのみ効率的であるのに対して，提案手法では密なデータに対しても効率的に極大頻出アイテム集合を発見することができる．

提案手法では，頻出アイテム集合が見つかる確率を理論的に求めることで，最小サポートからサンプリングの繰り返し数を自動的に計算する．さらに，頻出でないアイテム集合の割合を抑えるために，頻出アイテム集合の候補集合からなるバッチを導入する．頻出アイテム集合が見つかる確率と頻出でないアイテム集合が見つかってしまう確率の差を最大化する，という原理にもとづいて，バッチ数も自動的に求める．

提案手法では，入力パラメータとして，1 つの共通部分アイテム集合を生成するために比較するトランザクションの個数（以下サンプルサイズ）r，最小サポート $minsup$，頻出でないアイテム集合の個数を抑制するために用いる最小サポート未満のサポート（以下，**下限サポート**と呼ぶ）$lowsup$，誤差率 ϵ の 4 個のパラメータを用いる．ここで，誤差率 ϵ とは，共通部分アイテム集合を反復回数 h 回だけ取得した際に，頻出アイテム集合を見逃す確率の上限値である．これらの入力パラメータを元に，頻出アイテム集合の Recall と Precision の値を高く維持できるよう，反復回数 h ならびにバッチ数 k の値を解析的に求める．

4.1 提案手法のアルゴリズム

提案手法のアルゴリズムを Algorithm 1に示す．以下では，この動作について説明する．

まず，データベース $\mathcal{D}$ 中の全トランザクションからランダムに r 個のトランザクション $T_{i_1}\ T_{i_2}\ \ldots\ T_{i_r}$ を抽出し，これら r 個すべてに共通に含まれているアイテム集合

$$F_i = T_{i_1} \cap T_{i_2} \cap \ldots \cap T_{i_r}$$

を取得する．この操作を h 回繰り返すことで，h 個の共通部分アイテム集合 $F_1, F_2, \ldots, F_h$ を得る．ここで，これら h 個のアイテム集合の中で極大なアイテム集合のみからなる集合をバッチ $\mathcal{B}$ とする．すなわち，任意の共通部分アイテム集合 F_i に対して $F_i \subseteq F$ を満たすアイテム集合 $F \in \mathcal{B}$ が存在し，かつ，$F \subseteq G$ を満たすアイテム集合 G は $\mathcal{B}$ 中に存在しない．

上記の手順を繰り返すことで，k 個のバッチ $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$ を生成し，これら全てに共通して出現しているアイテム集合のなかで，極大なものを取り出す．具体的には，最終的に得られる極大頻出アイテム集合の候補集合を $\mathcal{F}$ とすると，各アイテム集合 $F \in \mathcal{F}$ に対して，どのバッチ $\mathcal{B}_i$ 中にも $F \subseteq F_i$ を満たす $F_i \in \mathcal{B}_i$ が存在して，かつ $F \subseteq G$ を満たす $G \in \mathcal{F}$ は存在しない．実際には，バッチを生成する度に，現在のバッチと比較することでバッチを更新していく．

提案手法では，実際のアイテム集合のサポートを計算することなく頻出アイテム集合を発見するため，$\mathcal{D}$ が巨大な場合でも，高速化及び省メモリ化の達成が期待できる．

4.2 提案手法の計算量

反復回数を h 回，バッチの個数を k 個，全アイテムの種類数を $|A| = m$ とすると，共通部分アイテム集合を取得する際に必要な空間計算量は $O(m)$，それらを格納するための配列に必要な空間計算量は $O(hm)$ となる．バッチは生成するたびに適宜比較するため，3 つ以上のバッチを同時に保持する必要がなく，アルゴリズム全体の空間計算量は $O(hm)$ となる．

また，共通部分アイテム集合の取得の計算に必要な時間計算量は $O(rmh)$，k 個のバッチ同士の比較を行うために必要な計算時間量は $O(mkh^2)$ である．したがって，全体の時間計算量は $O(mkh^2)$ となる．以上より，空間計算量と時間計算量は共にトランザクション数 $|D| = n$ に独立であることがわかる．

4.3 発見精度の解析とパラメータの自動決定

提案手法において，頻出アイテム集合が出力される確率を理論的に解析することによって，ランダムサンプリングの反復回数 h およびバッチの数 k を入力パラメータから求める．

データベース中のトランザクション数を $|\mathcal{D}| = n$ とする．あるアイテム集合 I のサポート $S(I)$ について，$s = S(I) \cdot n$ とし，サポート $S(I) = minsup$ および $S(I) = lowsup$ のときの s をそれぞれ s_{min}, s_{low} とする．ランダムサンプリングによって抽出されたトランザクション r 個の共通部分アイテム集合 F に I が含まれる確率 $p_S(s, r)$ は，二項係数から次式のように計算できる．

$$p_S(s, r) = \binom{s}{r} \Big/ \binom{n}{r}. \tag{2}$$

この操作を h 回反復して得られた h 個の共通部分アイテム集合 $F_1, F_2, \ldots, F_h$ について、少なくとも 1 つの共通部分アイテム集合 F_i が存在して，$I \subseteq F_i$ となる確率 $p_R(s, r, h)$ は，次式で与えられる．

$$p_R(s, r, h) = 1 - (1 - p_S(s, r))^h. \tag{3}$$

Algorithm 1 提案手法のアルゴリズム

入力: サンプルサイズ r，最小サポート $minsup$,
下限サポート $lowsup$，誤差率 ϵ,
出力: 頻出アイテム集合のリスト
$n \leftarrow$ データベース $\mathcal{D}$ 中のトランザクション数
$s_{min} \leftarrow minsup \cdot n$，$s_{low} \leftarrow lowsup \cdot n$
数式(4)で h を計算
数式(6)で k_{opt} を計算
for $l = 1$ to k_{opt} **do**
  for $i = 1$ to h **do**
    $\mathcal{D}$ 中からランダムに r 個のトランザクション
    $T_{i_1}, T_{i_2}, \ldots, T_{i_r}$ を取り出す
    $F_i \leftarrow T_{i_1} \cap T_{i_2} \cap \cdots \cap T_{i_r}$
    F_i をバッチ $\mathcal{B}_l$ に格納
  end for
  $\mathcal{B}_l$ 中に上位集合が存在するアイテム集合を $\mathcal{B}_l$ から削除
  if $l = 1$ **then**
    バッチ $\mathcal{F}$ を生成，$\mathcal{F}$ に $\mathcal{B}_1$ をコピー
  else
    $\mathcal{F}$ と $\mathcal{B}_l$ を比較し，両方に共通に現れている部分集合のみを $\mathcal{F}$ に上書きして更新
  end if
end for
$\mathcal{F}$ を出力

ゆえに，$\epsilon = 1 - p_R(s, r, h)$ とすると，反復回数 h を

$$h = \frac{\log \epsilon}{\log(1 - p_S(s_{min}, r))} \tag{4}$$

と定めることにより，確率 $1 - \epsilon$ 以上で I が共通部分アイテム集合のいずれかに含まれる．

ここまでの操作を k 回繰り返して k 個のバッチを生成したとき，それら k 個のバッチ全てに I が含まれている確率 $p_B(s, r, h, k)$ は

$$p_B(s, r, h, k) = p_R(s, r, h)^k \tag{5}$$

となる．そこで，提案手法では，

$$k_{opt} = \underset{k}{\operatorname{argmax}} \left(p_B(s_{min}, r, h, k) - p_B(s_{low}, r, h, k) \right) \tag{6}$$

を満たすバッチ数 k_{opt} を用いることで，サポートが $lowsup$ 未満のアイテム集合が出力される確率が低く，かつサポートが $minsup$ 以上のアイテム集合が出力される確率を高く維持して頻出アイテム集合を発見する．最終的に，頻出アイテム集合を発見する確率は $p_B(s_{min}, r, h, k_{opt})$ となり，h は数式(4)，k_{opt} は数式(6)から求まる．

5. 評価実験

5.1 実験環境

実験は，Intel Core i7-6500U CPU 2.50 GHz のプロセッサ，実装メモリ 16 GB の Windows10 Pro 64 bit OS 上で行った．提案手法は C 言語及び C++ 言語で実装し，同じく C 言語で実装されている比較対象の LCM[*1] とともに gcc (version 4.9.3) でコンパイルした．

*1 http://research.nii.ac.jp/~uno/code/lcm53.zip

5.2 実験条件

FIMI Repository [Goethals 03] で提供されているデータ chess を用いて評価実験を行った．chess は，トランザクション数が $n = 3,196$ で，各トランザクションが全 75 種類のアイテム中から 37 個のアイテムを持つ，密な実データである．このデータに対して，$minsup$，及び $lowsup$ の値を固定し，r を変えたときの頻出アイテム集合発見にかかる時間と Recall, Precision について評価した．比較対象とする解析手法には LCM ver 5.3 を採用し，頻出アイテム集合の候補を列挙した．LCM と同様の前処理として，データセット中の全トランザクションから出現回数が $minsup$ 未満のアイテムを全て削除している．なお，Recall の値は，頻出アイテム集合が，提案手法が出力したアイテム集合中にどの程度含まれているかを表し，Precision は提案手法が出力したアイテム集合がどの程度頻出アイテム集合となっているかを表している．すなわち，$minsup$ 及び $lowsup$ に関する頻出アイテム集合をそれぞれ $\mathcal{F}_{minsup}$, $\mathcal{F}_{lowsup}$ とし，提案手法が出力したアイテム集合を $\mathcal{F}$ とすると，Recall $= |\{F \in \mathcal{F}_{minsup} \mid \text{ある} I \in \mathcal{F} \text{が存在して} F \subseteq I\}|/n$, Precision $= |\{I \in \mathcal{F} \mid \text{ある} F \in \mathcal{F}_{lowsup} \text{が存在して} I \subseteq F\}|/n$ と定義される．提案手法では，$lowsup$ 未満のアイテムが出力されないことを理論的に保証しているため，Precision の値は $lowsup$ を基準にして計算を行っている．また，全ての実験を通して誤差率 $\epsilon = 0.001$ とした．

5.3 実験結果

実データ chess に対して，r を変化させたときの計算時間，Recall，Precision の三つの値を評価した．$minsup = 0.50$, $lowsup = 0.40$ となるよう $minsup$，$lowsup$ を固定し，r を 8 から 13 の範囲で変化させたときの計算時間，Recall，Precision をそれぞれ図 1， 2， 3に示す．

図 2， 3より，提案手法は Recall，Precison の値をどちらも高く保つことができており，出力するアイテム集合はそのほとんどが頻出アイテム集合となっていることがわかる．これは，理論計算に一致する結果である．計算時間は，バッチの比較に時間を要するため，LCM に比べて遅くなっている．サンプルサイズ r の値が小さい方が計算時間が遅くなっているのは，バッチの個数 k の影響によるものと考えられる．実際，今回の実験では $r = 8$ のとき $k = 15$，$r = 10$ のとき $k = 9$ となっている．

5.4 実験の考察

実験結果から，提案手法は LCM と比べて計算に時間がかかっているが，これは実データ 'chess' がトランザクション数，アイテムの総種類数がともに小さく，LCM がじゅうぶん高速に解析できるためと考えられる．提案手法は，メモリに一度に読み込むことができないような大規模データを扱うことを想定しているため，時間計算量はデータベースの大きさではなく，入力パラメータから求める反復回数に依存する．このため，より巨大なデータで実験を行えば，提案手法が LCM より高速に頻出アイテム集合を発見できると期待される．また提案手法では，サポートの計算を行わずに，理論解析どおりに Recall，Precision の値を高く維持することができた．このことからも，提案手法は，より巨大なデータベースにおいて，すべてのトランザクションを見ることなく頻出アイテム集合を発見することができると期待される．

6. 結論

本論文では，頻出アイテム集合発見問題に対して，サンプリングを用いて確率的に頻出アイテム集合を発見する手法を提案し，その性能を評価した．サポートの値を計算することなく，か

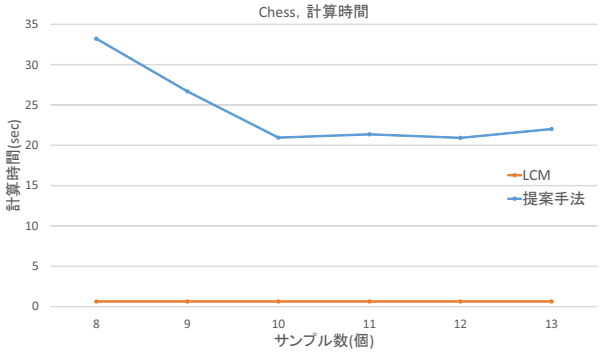

図1　Chess, 計算時間

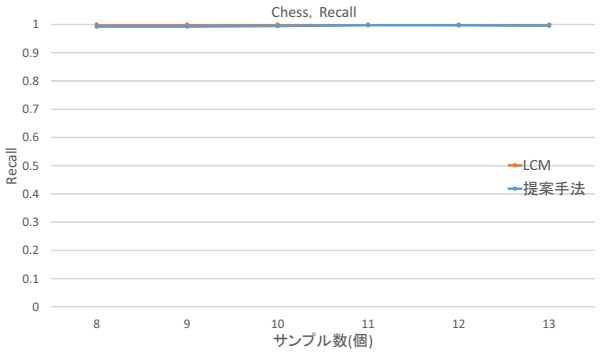

図2　Chess, Recall

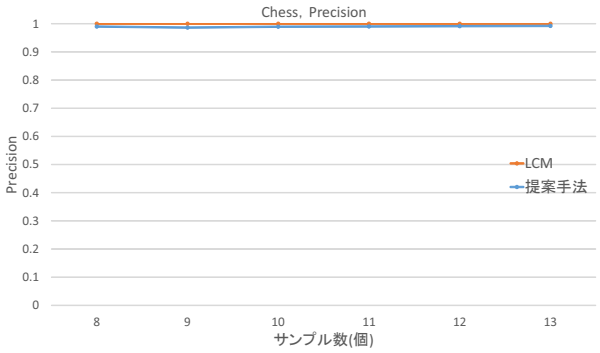

図3　Chess, Precision

つ高い確率で頻出アイテム集合を得ることができることを理論的に示し，実験で検証した．今後の課題としては，計算時間をより高速化することと，一度に主記憶上に読み込むことが困難な現実の大規模データを用いた提案手法の検証が挙げられる．

参考文献

[Agrawal 94] Agrawal, R. and Slikant, R.: Fast algorithms for mining association rules, in *Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487–499 (1994)

[Agrawal 14] Agrawal, C. C. and Han, J. eds.: *Frequent Pattern Mining*, Springer (2014)

[Goethals 03] Goethals, B. and Zaki, M. J.: *the FIMI repository*, http://fimi.cs.helsinki.fi/ (2003)

[Han 00] Han, J., Pei, J., and Yin, Y.: Mining frequent patterns without candidate generation, in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 1–12 (2000)

[Llinares-López 15] Llinares-López, F., Grimm, D. G., Bodenham, D. A., Gieraths, U., Sugiyama, M., Rowan, B., and Borgwardt, K. M.: Genome-Wide Detection of Intervals of Genetic Heterogeneity Associated with Complex Traits, *Bioinformatics*, Vol. 31, No. 12, pp. i240–i249 (2015)

[Otey 04] Otey, M. E., Parthasarathy, S., Wang, C., Veloso, A., and Meira, W.: Parallel and distributed methods for incremental frequent itemset mining, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 34, No. 6, pp. 2439–2450 (2004)

[Shah 14] Shah, R. D. and Meinshausen, N.: Random Intersection Trees, *Journal of Machine Learning Research, 15, 629-654*, pp. 629–654 (2014)

[Terada 13] Terada, A., Okada-Hatakeyama, M., Tsuda, K., and Sese, J.: Statistical significance of combinatorial regulations, *PNAS*, Vol. 110, No. 32, pp. 12996–13001 (2013)

[Uno 04] Uno, T., Asai, T., Uchida, Y., and Arimura, H.: An efficient algorithm for enumerating closed patterns in transaction databases, in *Discovery Science*, Vol. 3245 of *Lecture Notes in Computer Science*, pp. 16–31 (2004)