3C4-2

サンプリングを用いた高速頻出パターンマイニング

Fast frequent itemset mining using sampling

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

In this paper, we propose an efficient sampling-based method for frequent itemset mining, which is one of the central tasks in data mining. To date, frequent itemset mining have been successfully applied in various disciplines such as business and genome analysis. However, a fast and memory efficient method is required because databases on a broad scale over several terabytes are recently appearing, in which existing methods take too long time to enumerate frequent itemsets as they cannot even load such databases on the main memory. Our method repeats sampling of a small part of a given database and does not need to load the entire database. We analytically give the number of repetition of sampling needed to retrieve frequent itemsets given a true positive rate. Experiments show that our method is faster than the state-of-the-art frequent itemset mining method on dense benchmark datasets and can control the desired false negative rate.

1. 序論

データベースからの頻出アイテム集合発見 [Aggarwal 14] は，データマイニングにおける最も基本的な問題の 1 つであり，これまで盛んに研究されてきた．アイテム集合とは，同時に出現する対象の組み合わせであり，例えば，購買データベースにおける一緒に売れた商品の組み合わせや，遺伝子データベースにおける同時に発現する遺伝子の組み合わせなどが発見の対象となる．特に，より頻出するアイテム集合，すなわちそのアイテム集合を含むトランザクション数（サポート）が大きいアイテム集合が，多くの応用領域で重要な対象として解析される．例えば，マーケットバスケット分析では，より多くの消費者に購入されている商品の組み合わせが，収益の向上にとって重要となる．

しかし，最近では数テラバイト以上の高次元かつ大規模なデータが出現しており、最先端の高速なアイテム集合発見手法（例えば LCM [Uno 04]）を用いても解析が困難な場合がある．特に，ほとんどの既存手法では，最初にデータを全て読み込み主記憶上に保持する必要があるため，データが膨大になるとこれらの既存手法を適用できない．また，逐次的にトランザクションを読み込む手法を用いても，計算時間が大きくなってしまい，適用が困難である．このため，大規模なデータにおいて，データ全てを読み込むことなくサポートの大きいアイテム集合を効率的に発見する手法が求められている．

本稿では，ランダムサンプリングを用いて効率的に頻出アイテム集合の候補を見つけるための手法を提案する．提案手法では，ランダムに取得したトランザクションのみから頻出アイテム集合を発見する．このため，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も適用が可能である．さらに，必要となるサンプリング数を理論的に求めることで，Recall を高い値に保ったまま既存の全列挙型手法よりも高速に頻出アイテム集合が発見できることを実験的に示す．

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 節で提案手法とその理論解析について述べ、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

頻出アイテム集合発見問題において，A をある有限集合としたとき，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**（transaction）T はアイテム集合であり，与えられるデータは**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 $\mathcal{D} = \{T_1, \ldots, T_n\}$ である．以降，トランザクションデータベースを単にデータベースと呼ぶ．

あるアイテム集合 F が頻出であるかどうかを判定するには、**サポート** (support) $S(F)$ を用いる．$S(F)$ は，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であり，以下の式 (1) によって定義される．

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

この値が，ユーザーが与える，**最小サポート** (minimum support) $minsup$ と呼ばれる閾値以上である場合，この F を**頻出アイテム集合**（frequent itemset）と呼ぶ．頻出アイテム集合発見問題とは、$\mathcal{D}$ 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である．

頻出アイテム集合の中でも特に，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 を**極大頻出アイテム集合**（maximal frequent itemset）と呼ぶ．頻出アイテム集合は，必ず極大頻出アイテム集合の部分集合となる．

3. 関連手法

Apriori 法 [Agrawal 94] 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することで，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．Apriori

法で使われているこの性質は，以降の頻出アイテム集合マイニング手法でも頻繁に採用されている．

既存の頻出アイテム集合発見手法のなかで，最速の手法の一つが LCM [Uno 04] である．LCM は，配列とビットマップ，接頭辞木を組み合わせた手法であり，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．本研究では，提案手法と LCM の速度を比較することで，提案手法の効率性を検証する．

これらの既存手法は，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，$\mathcal{D}$ が巨大な場合は，一度に主記憶上に格納できない．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入力する必要があり，列挙の効率性が失われてしまう．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が保持するアイテム数がアイテム総種類数に比べて小さいという性質を利用して高速化を達成しているため，遺伝子解析などで登場しつつある密なデータベースにおいては，効率性が著しく損なわれてしまう．

一方，近年，より高速な解析のため，サンプリングを用いてデータベースの一部分を抽出し，そこから頻出アイテム集合を発見するための手法が提案されている．その中で代表的な手法である Random Intersection Trees [Shah 14] (以下 RIT と記述) は，トランザクションのランダムなサンプリングによって，頻出アイテム集合を近似的に求める手法である．RIT では，ランダムに選び出したトランザクションから多数の木構造を構成する．まず根ノードに $\mathcal{D}$ 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 $T_{i_1}, \ldots, T_{i_k}$ をサンプリングで取り出し，それぞれに対して T との共通部分集合 $T \cap T_{i_j}$ を子ノードとする．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間の共通アイテム集合の取り出しに基づく手法を提案する．

4. 提案手法

本研究では，ランダムサンプリングによって選ばれたトランザクション同士を比較することで，近似的にサポートが大きい頻出アイテム集合を求める手法を提案する．さらに，提案手法では，サポートの値が大きい極大頻出アイテム集合の発見を対象とする．このため，本提案手法が扱う問題では，列挙すべきアイテム集合数がある程度限られる．一方，本提案手法は，$\mathcal{D}$ が疎という仮定を置かないため，密なデータに対しても高速に極大頻出アイテム集合を見つけることができる．

また，提案手法では頻出アイテム集合を見逃す確率を解析的に求めることで，各頻出アイテム集合が見つかる確率を理論的に保証し，最小サポートからその確率を達成するために必要となるサンプリングの繰り返し数を求める．

提案手法では，入力パラメータとして，1 つの頻出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（サンプルサイズ）r，最小サポート $minsup$，誤差率 ϵ の 3 個のパラメータを用いる．ここで，誤差率 ϵ とは，頻出アイテム集合を見逃す確率の上限値である．提案手法では，4.3 節で述べるように, 頻出アイテム集合を見逃す確率を誤差率 ϵ 以下にするために必要となるサンプリングの反復回数 h を，入力パラメータから解析的に求める．

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．

Algorithm 1 提案手法

入力: サンプルサイズ r, 最小サポート $minsup$, 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
$s \leftarrow minsup \cdot n$
$p \leftarrow \binom{s}{r} / \binom{n}{r}$ // 数式 (2)
$h \leftarrow \log \epsilon / \log(1-p)$ // 数式 (4)
for $i = 1$ to h **do**
  $\mathcal{D}$ 中からランダムに $T_{i_1}, T_{i_2}, \ldots, T_{i_r}$ を取り出す
  $F_i \leftarrow T_{i_1}$
  for $j = 2$ to r **do**
    $F_i \leftarrow F_i \cap T_{i_j}$
  end for
  F_i の重複をチェックするため, トライ木を探索
  if トライ木に F_i と同じアイテム集合が存在しない **then**
    F_i をトライ木に格納
  end if
end for
全ての極大アイテム集合を出力

まず，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 $T_{i_1}, T_{i_2}, \ldots, T_{i_r}$ を選び，この r 個のトランザクション全てに共通に含まれるアイテム集合

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

を取り出す．ここで，得られたアイテム集合の重複チェック，及びそれらの格納のために，トライ木を採用する．このトライ木は，各ノードが 1 つのアイテムを保持する．サンプリングで取り出したアイテム集合 F に含まれるアイテムを辞書式順序で並べ替え，それらを $a_1, a_2, \ldots, a_k$ とすると，根ノードの子ノードに a_1 を挿入，a_1 の子ノードに a_2 を挿入という操作を繰り返して，a_k までトライ木に格納する．末端のノード a_k には，このノードまでがアイテム集合 F であることを示すためのキーも同時に挿入する．

サンプリングを繰り返し，新たにアイテム集合 F を得るたびに，F のアイテムを辞書式順序で並べ替え，トライ木をたどり，既に F を含むアイテム集合があるかどうかを確認する．もし既に存在する場合は，その F を破棄する．

以上の処理を h 回繰り返し，h 個のアイテム集合 $F_1, F_2, \ldots, F_h$ について，格納と重複のチェックを行う．最後に，トライ木に格納されているアイテム集合の中で，極大なものを全て出力する．ここで，得られたアイテム集合の $\mathcal{D}$ 中における実際のサポートは計算しないことに注意されたい．これにより，$\mathcal{D}$ が巨大な場合でも，高速化及び省メモリ化が達成できる．

4.2 提案手法の計算量

反復回数を h 回，全アイテムの種類数を $|A| = m$ とする．サンプリングしたアイテム集合からの共通部分の計算に必要な空間計算量は $O(m)$，それらの格納のためのトライ木に必要な空間計算量は $O(hm)$ なので，アルゴリズム全体の空間計算量は $O\,(hm)$ となる．

また，サンプリングしたアイテム集合からの共通部分の計算に必要な時間計算量は $O(rmh)$，トライ木への格納に必要な時間計算量は $O(mh)$ である．従って，全体では $O(rmh)$ となる．空間計算量と時間計算量は共にトランザクション数 $|D| = n$ に無関係である．

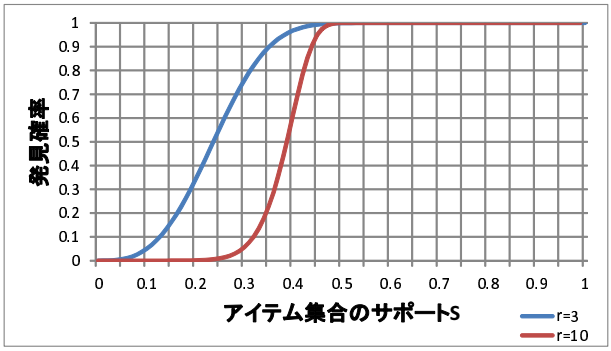


図 1: r の値に対する，各サポートのアイテム集合が共通部分として現れる確率 $1-(1-p)^h$.

4.3 ランダムサンプリング操作の反復回数

ランダムサンプリングの操作を行う回数 h は，入力パラメータから計算する．トランザクション数 $|\mathcal{D}| = n$ とし，アイテム集合 I について，$s = S(I) \cdot n$ とする．ランダムサンプリングの操作を 1 回行ったときに，選ばれたトランザクション r 個から得た共通部分のアイテム集合 F に，I が含まれる確率 p は 2 項係数を用いて次式で与えられる．

$$p = \binom{s}{r} \Big/ \binom{n}{r}. \tag{2}$$

このとき，ランダムサンプリングを h 回繰り返して得られたアイテム集合のいずれにもアイテム集合 I が含まれない確率（誤差率）ϵ は

$$\epsilon = (1-p)^h \tag{3}$$

となる．したがって，サンプリングの繰り返し回数 h を

$$h = \frac{\log \epsilon}{\log(1-p)} \tag{4}$$

と定めると，確率 $1-\epsilon$ 以上で I を発見することができる．そこで，提案手法では，逆に $s = minsup \cdot n$ として必要なサンプリング回数 h を計算する．これによって，サポートの値が $minsup$ 以上の頻出アイテム集合を $1-\epsilon$ 以上の確率で発見することができる．

4.4 サンプルサイズ r の解析

頻出でないアイテム集合，すなわち偽陽性のアイテム集合が出力されてしまう確率は，サンプルサイズ r に依存する．図 1 に，全トランザクション数 $n = 10,000$ とし，$minsup$ を 0.5，誤差率 $\epsilon = 0.001$ とした場合に，反復回数 h で各サポート S のアイテム集合が出力される確率 $1-(1-p)^h$ を示す．図の青の線は $r = 3$，$minsup = 0.5$ のときの反復回数 $h = 52$ のグラフであり，赤の線は $r = 10$，$minsup = 0.5$ のときの反復回数 $h = 7,103$ のグラフである．サンプルサイズ r が小さいときは，反復回数 h が小さいので解析が高速に終了することが期待されるが，その代わり頻出ではないアイテム集合が出力されやすいことがわかる．例えば，$s = 0.3$ のアイテム集合は，$r = 3$ のときは約 0.75 の確率で出力されてしまうが，$r = 10$ のときは約 0.05 となり，誤発見が起こりにくい．

5. 評価実験

5.1 実験環境

実験は，Intel Core i7-4790 CPU 3.60GHz のプロセッサ，実装メモリ 8.00GB の Windows 7 Enterprise 64 bit OS 上でおこなった．提案手法は 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 4.8.3) でコンパイルした．

5.2 実験条件

人工的に生成した密なデータと，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．人工データは，トランザクション数 $n = 10,000$，アイテムの種類数 $m = 10,000$ で，各々のトランザクションが各アイテムを確率 0.5 で保持する密なデータである．chess は，トランザクション数が $n = 3,916$ 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 2 つのデータに対して，$minsup$ 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，極大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，提案手法が出力したアイテム集合中に頻出アイテム集合が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．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

まず，人工データに対して，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．また，出力すべき頻出アイテム集合の総数の変化がこれらの値に与える影響を確認するため，$minsup$ の値が 0.45 と 0.50 の 2 つの場合について評価を行った．得られた結果を，$minsup = 0.45$ のときの計算時間（図 2a），Precision（図 2d）と，$minsup = 0.50$ のときの計算時間（図 2b），Precision（図 2e）について，図に示した．ただし，LCM は r とは無関係であるため，各 r の値について全て同じ値であり，かつ Recall と Precision は常に 1 である．

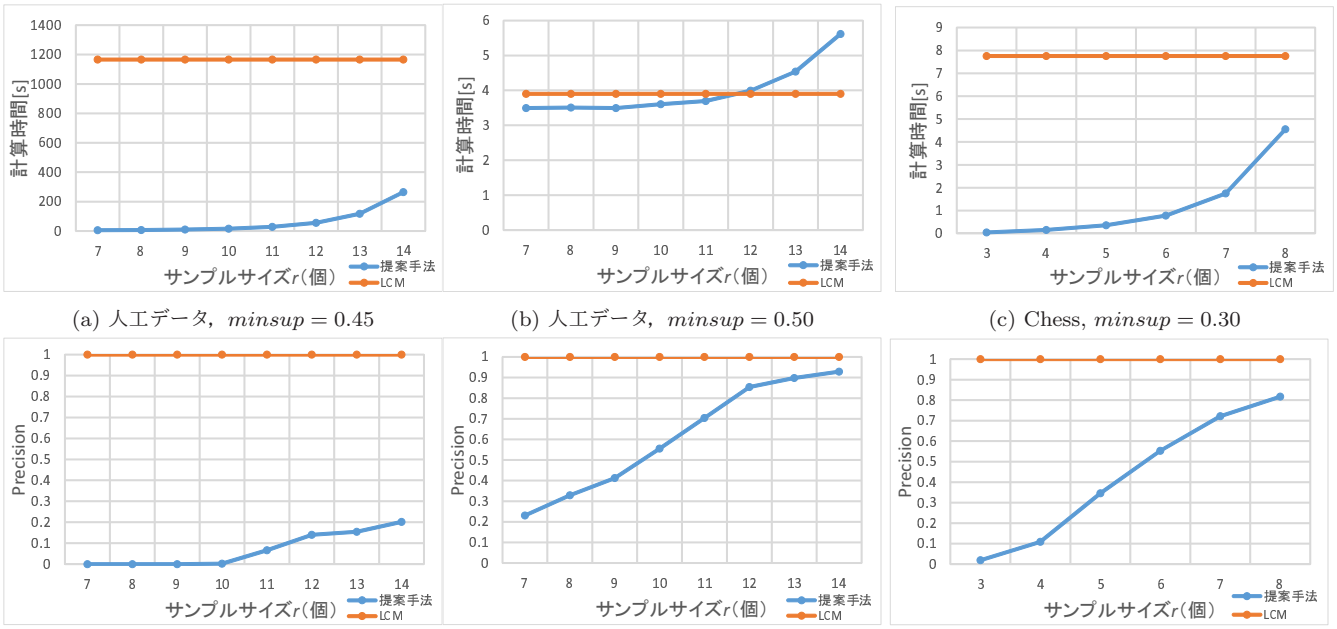
提案手法の Recall は，すべての場合において，理論計算の通りほぼ 1 となった．実際の解析においては，頻出なアイテム集合を見逃すことのほうがリスクが高いため，高い Recall が求められるが，提案手法はそれを満たしている．また，$minsup = 0.45$ のとき，$r = 14$ の場合でも，4 倍以上の高速化を達成している（図 2a）．また，r が大きくなるにつれて Precision の値も大きくなるが，$r = 14$ でも Precision は 0.20 と，低い値になっている（図 2d）．一方，$minsup = 0.50$ のときは，例えば $r = 11$ では LCM より高速かつ Precision が 0.70 と比較的高くなった（図 2b, 2e）．

次に，実データ chess に対して，同様の評価実験を行った．$minsup$ の値を 0.30 に固定し，r を変化させたときの計算時間と Precision をそれぞれ図 2c, 2f に示した．図 2c より，r が 8 以下のとき，提案手法は LCM よりも高速となっている．また，図 2f では，r の増加につれて，Precision の値が大きく増加している．

5.4 実験の考察

人工データでは，$minsup = 0.45$ のときは提案手法の方が LCM よりも解析が高速に終了しているが，$minsup = 0.50$ では $r = 12$ 以上の場合に LCM より提案手法の方が計算に時間がかかってしまっている．これは，(2) 式より，r や n が大きい場合や s の値が小さい場合は，p の値が小さくなり，式 (4)

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



(a) 人工データ，$minsup = 0.45$　(b) 人工データ，$minsup = 0.50$　(c) Chess, $minsup = 0.30$

(d) 人工データ，$minsup = 0.45$　(e) 人工データ，$minsup = 0.50$　(f) Chess, $minsup = 0.30$

図 2: 計算時間（上段）および Precision（下段）.

により反復回数 h の値が非常に大きくなってしまうためである．実際に，人工データにおいて，$minsup = 0.50$ の場合では $r = 7$ のときの反復回数は $h = 883$ であり，$r = 14$ のときは $h = 114,210$ となる．また，$minsup = 0.50$ の場合は，頻出アイテム集合の総数が少なくなり，その結果 LCM の計算時間が高速に終了したことも LCM より提案手法の方が計算に時間がかかる原因に挙げられる．

また，提案手法ではいずれの結果においても，r が大きくなるにつれて Precision の値が大きくなるという結果が得られた．これは，r が大きくなると，頻出ではないアイテム集合が出力されにくくなることを表しており，図 1 で示した理論解析と一致している．

以上の考察より，提案手法は，密なデータにおいて，$r = 10$ 程度の小さいサンプルサイズで，高速に極大頻出アイテム集合を列挙できることがわかった．特に，列挙すべき極大頻出アイテム集合の総数が大きいとき，実用性が高い．ただし，効率性を保ちつつ高い Precision を得るためのサンプルサイズ r はデータセットによって異なるため，最適な r の計算は今後の課題である．

6. 結論

本論文では，頻出アイテム集合発見問題に対して，ランダムサンプリングを用いた手法を提案し，その性能を評価した．その結果，アイテムの種類数が多く，頻出アイテム集合の個数が非常に多いために解析が困難である密なデータの解析において，提案手法は既存の手法より高速であることが判った．特に，このようなデータに対して，Recall をほぼ 1 に保ちつつ，サンプルサイズ r を大きくすることで，Precision を上げることができることを示した．

提案手法において，サンプルサイズ r が小さい場合はアルゴリズムが高速であるが，r が大きすぎると，反復回数 h が爆発的に増加してしまい，アルゴリズムが遅くなってしまう．アルゴリズムを改良し，高速なままに高い Precision を確保することは，今後の課題である．例えば，LCM で用いられている効率的なアイテム集合の探索法を提案手法に部分的に取り込むことなどに取り組む予定である．一方，ランダムサンプリングの性質上，データベース全てを保持する必要がないため，主記憶に格納することができないような大規模なデータベースからも頻出アイテム集合を効率的に発見することができる．この利点について実データを用いて実証する必要がある．

参考文献

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

[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)

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

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

[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 *LNCS*, pp. 16–31 (2004)