超高次元データからの頻出パターン発見手法

森　紘一郎†　　折原　良平†

† 株式会社 東芝 研究開発センター　〒 212–8582 神奈川県川崎市幸区小向東芝町 1
E-mail: †{kouichirou1.mori, ryohei.orihara}@toshiba.co.jp

あらまし　従来の Apriori に代表される頻出パターン発見手法は，属性の組合せを探索するために属性数が大きいデータでは計算量が指数関数的に大きくなるという問題があった．本研究の目的は超高次元データから頻出パターンを効率的に発見する手法を開発することにある．本論文ではレコード空間探索を用いることでこの問題を解決した．レコード空間探索とは，レコードの組合せを探索する手法であり，レコードの組合せに共通する属性を求めることで頻出パターンを探索する．また探索を効率化するために最小パターン長と呼ぶパラメータを導入した．本手法を用いることで，Apriori では処理できないような超高次元データセットから頻出パターンを実時間で抽出することができた．
キーワード　データマイニング，相関ルール，頻出パターン，高次元データ，レコード空間探索

Frequent Pattern Mining from Super-High-Dimensional Data

Kouichirou MORI† and Ryohei ORIHARA†

† Research & Development Center, Toshiba Corporation
1 Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, Kanagawa, 212–8582, Japan
E-mail: †{kouichirou1.mori, ryohei.orihara}@toshiba.co.jp

Abstract　Traditional frequent pattern mining methods such as Apriori have a problem in that the order of calculation exponentially grows with high-dimensional data because of search by the combination of attribute sets. The purpose of our work was to develop a method that efficiently extracts frequent patterns from very high-dimensional data. We can solve the problem by a record space search. The record space search means the search by the combination of record sets. We can find frequent patterns from attributes common to the combination of record sets. Moreover, we introduce a parameter called minimum pattern length to search efficiently. As a result, we can extract the frequent patterns in real time from high-dimensional datasets that cannot be handled by Apriori. In conclusion, our method is efficient for frequent pattern mining from high-dimensional data.
Key words　Data Mining, Association Rule, Frequent Pattern, High Dimensionality, Record Space Search

1. はじめに

大量のデータから有用な知識を発見する技術はデータマイニングと呼ばれている．頻出パターン発見はデータマイニングの主要テーマのひとつであり，データベース中に頻出する属性の組合せを発見するのが目的である．

頻出パターン発見のアルゴリズムは，Apriori[1] をはじめとしてさまざまな手法が提案されてきた．従来手法の多くは，レコード数が極めて大きく，属性数が小さいデータを効率的に処理することに重点が置かれていた．これらのデータの特徴をトランザクション分析に例えると，バスケットの数が極めて多く，バスケット内の商品数は少ない場合に相当する．

しかし，近年，レコード数は小さいが，属性数が極めて大きいデータを対象とした分析への期待が高まっている．たとえば，遺伝子のマイクロアレイデータやセンサーデータなどである．これらのデータの特徴をトランザクション分析に例えると，バスケットの数は少なく，バスケット内の商品数は極めて多い場合に相当する．

このような属性数が極めて大きいデータは，従来の Apriori に代表される属性の組合せを探索していく手法では計算量が指数関数的に大きくなり実時間内での計算が困難である．

そこで，本論文では，レコード数が小さく（数百のオーダ），属性数が極めて大きい（数万のオーダ）超高次元データから頻出パターンを効率的に発見する手法を提案する．本手法は，Apriori と類似しているが，レコードの組合せを探索する点で本質的に異なっている．

超高次元データの頻出パターン長は長くなる傾向にあるため短いパターンより長いパターンの方が重要だと考える．そのた

RID	属性 A	属性B	属性C	属性D	属性E
0	True	True	False	True	True
1	False	True	True	False	True
2	True	True	False	True	True
3	True	True	True	False	True
4	True	True	True	True	True
5	False	True	True	True	False

(a)

TID	アイテム
0	A B D E
1	B C E
2	A B D E
3	A B C E
4	A B C D E
5	B C D

(b)

図 1　(a) 関係データベース (b) トランザクションデータベース

め，本手法では最小パターン長と呼ぶパラメータを導入し，最小パターン長より長い頻出パターン，もしくは最長の頻出パターンのみ求める．

2 章では，本論文で用いる用語を定義する．3 章では，本手法と比較検討する Apriori について詳説する．また，超高次元データを対象とした近年の研究について簡単にまとめる．4 章では，提案手法を詳説する．5 章では，本手法をテストデータセットに適用した結果について述べる．6 章では，実験結果について考察する．7 章では，全体をまとめる．

2. 用 語 定 義

本論文で用いる用語を定義する．図 1 (a) で表されるデータベースを関係データベースと呼ぶ．関係データベースはレコードと属性からなり，各レコードはいくつかの属性を含む．各レコードにはレコード ID（RID）が振られている．ここでは属性が 2 値（True または False）と仮定する．True の場合はその属性を持ち，False の場合はその属性を持たないことを意味する．多値属性または連続値属性の場合でも離散化することによって 2 値属性に変換可能である．

図 1 (b) で表されるデータベースをトランザクションデータベースと呼ぶ．トランザクションデータベースはトランザクションとアイテムからなり，各トランザクションは複数のアイテムを含む．各トランザクションにはトランザクション ID（TID）が振られている．

関係データベースとトランザクションデータベースは相互に変換可能である．レコードはトランザクション，属性はアイテムに対応する．本論文ではレコードとトランザクション，属性とアイテムを同じ意味で用いる．

データが高次元であるとは，属性数が多いことを指す．本論文における超高次元は属性数が数千から数万のオーダーであることを想定している．またデータが密であるとは，関係データベース形式において各レコードで True となる属性が多いこと，トランザクションデータベース形式において各トランザクションのアイテム数（トランザクションサイズまたはレコード長と呼ぶ）が多いことを指す．

アイテム集合（itemset）とはアイテムの組合せのことであり，レコード集合（ridset）とはレコードの組合せのことである．たとえば，{A,B,C} はアイテム集合，{1,5} はレコード集合である．特にアイテム集合のことをパターンと呼ぶ．以後，アイテム集合 {A,B,C} を ABC，レコード集合 {1,5} を 15 のように略記する．また，アイテム集合長（アイテム集合に含まれるアイテム数）が k のとき，長さが k のアイテム集合（k-itemset）と呼ぶ．同様に，レコード集合長（レコード集合に含まれるレコード数）が k のとき，長さが k のレコード集合（k-ridset）と呼ぶ．

アイテム集合のサポートカウントとは，そのアイテム集合を含むレコードの数である．たとえば，図 1 において ABC はレコード 3 と 4 に含まれるためサポートカウントは 2 となる．

頻出パターン（頻出アイテム集合）とは，ユーザがあらかじめ設定した最小サポートカウント（minsup）以上のサポートカウントを持つアイテム集合である．たとえば，図 1 において最小サポートカウントを 3 とした場合，B（サポートカウントは 6），AB（サポートカウントは 4），ABE（サポートカウントは 4）などは頻出パターンである．

3. 関 連 研 究

本章では，代表的な頻出パターン発見手法である Apriori [1] と高次元データの頻出パターン発見に関する近年の研究 [4], [6] について述べる．

3.1 Apriori

Apriori [1] は与えられたトランザクションデータベースからすべての頻出アイテム集合を効率よく発見するアルゴリズムである．Apriori は短い頻出パターンから順に長い頻出パターンを見つけていくボトムアップ型のアルゴリズムであり，効率化を図った多数の派生アルゴリズムが提案されている．以下，Apriori の手順について述べる．

（1） 長さ 1 の頻出アイテム集合の抽出

データベース中に出現するすべてのアイテムを列挙する．データベースを走査して各アイテムのサポートカウントを計算し，最小サポートカウントに満たないアイテムは除去する．残ったアイテムを長さ 1 の頻出アイテム集合 L_1 とする．

（2） 候補アイテム集合の生成

Apriori では，長さ $k-1$ の頻出アイテム集合 L_{k-1} から長さ k の頻出アイテム集合 L_k を順に求めることによってすべての長さの頻出アイテム集合を生成する．まず，L_{k-1} から長さ k の候補アイテム集合 C_k を生成する．候補アイテム集合の生成は Join と呼ぶ操作を用いる．頻出アイテム集合 L_{k-1} に含まれるアイテム集合を l_1，l_2 とし，$l_i[j]$ を l_i の j 番目のアイテムとする．l_1 と l_2 が $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ を満たすとき，l_1 と l_2 から長さ k の候補アイテム集合 $l_1[1] \cdots l_1[k-2] l_1[k-1] l_2[k-1]$ を生成する．ここで，$<$ は辞書順を意味する．たとえば，ABC と ABD からは ABCD が生成できる．

（3） 候補の枝刈り

Apriori では，Apriori Property [2] と呼ばれる性質を用いて候補アイテム集合を削減している．Apriori Property とは，「F が頻出アイテム集合であるならば，F の空でない部分集合もまた頻出である」という性質である．対偶を取ると，「F の空でない部分集合が頻出でなければ，F は頻出アイテム集合でない」となる．つまり，先ほど生成した長さ k の候補アイテム集合の部分集合が頻出であるか調べ，頻出でない部分集合がある場合，生成した候補アイテム集合は頻出ではないためあらかじめ候補

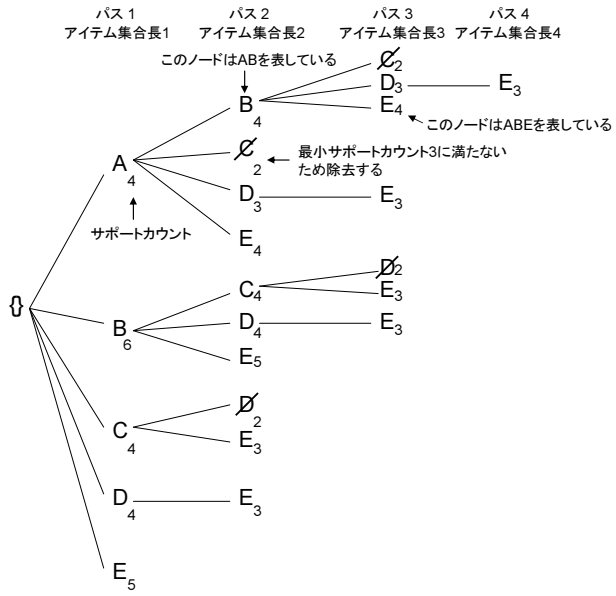

図 2　Apriori の処理例

から除去できる．

（4） サポートカウントの計算

候補アイテム集合 C_k を生成したら，データベースを走査し，候補アイテム集合のサポートカウントを計算する．データベースの各レコードを構成するアイテム集合から長さ k の全部分集合を求め，対応する候補アイテム集合がある場合カウントする．最後に最小サポートカウントに満たないアイテム集合を候補アイテム集合から除去し，残った候補アイテム集合を L_k とする．

以上の（2）から（4）の手順（パスと呼ばれる）を k を 1 ずつ増やしながら繰り返し実行し，頻出アイテム集合 L_k が空集合になった時点で処理を終了する．この段階ですべての長さの頻出アイテム集合が得られている．

図 2 は，図 1 のデータベースを用い，最小サポートカウントを 3 とした場合の Apriori の具体例である．ここでは，トライというデータ構造を用いている．トライは節点と枝から構成される．たとえば，A，B，C などは節点であり，節点と節点をつなぐ線が枝である．Apriori では，節点がアイテムになるのが特徴である．図 2 には，トライ全体を図示しているが，実際はアイテム集合長 1 から順に枝を伸ばして探索を進める．

まず，長さ 1 の頻出アイテム集合 L_1 である A，B，C，D，E を求め，データベースを走査してサポートカウントを計算する．次に L_1 をもとに長さ 2 の候補アイテム集合 C_2 である AB，AC，AD，AE，･･･，DE を Join で生成し，データベースを走査してサポートカウントを計算する．このとき，AC のサポートカウントは 2 であり，指定した最小サポートカウント 3 より小さいため AC を除去する．この処理を繰り返し，残ったアイテム集合が頻出アイテム集合となる．

Apriori では，候補アイテム集合の生成はアイテムの組合せによって求める．このようなアイテム（属性）の組合せによる頻出アイテム集合の探索はアイテム（属性）空間探索と呼ばれる．

高次元または密なデータベースに Apriori を適用するのは困難である．Apriori はアイテムの組合せを探索するため，データが高次元の場合，アイテムの組合せ数が爆発的に大きくなるという問題がある．また，データが密の場合，頻出アイテム集合長 L が大きくなる傾向にあり，長さ L の頻出アイテム集合 I を求める前に 2^L 個の I の部分集合をすべて生成しなければならないため計算時間が長くなる．

3.2　MFI/CFI の抽出

長い頻出パターンを発見するのが困難という Apriori の欠点に対して，極大頻出アイテム集合（Maximal Frequent Itemsets: MFI）を効率よく発見できる手法として Max-Miner [3] が提案されている．ある頻出アイテム集合 I が，頻出である superset を持たないとき，I を MFI と呼ぶ．MFI の部分集合もまた頻出であるため，すべての MFI が求まればそこからすべての頻出アイテム集合を生成することができる．Max-Miner は Apriori と同様に属性空間のボトムアップ探索を行うが，長い頻出アイテム集合を早めに同定する先読みを行い，その頻出アイテム集合の部分集合を枝刈りすることによって処理を効率化している．Max-Miner が対象としているのは非常に密なデータベースであり，数千から数万次元の超高次元データについては [3] では言及されていない．

近年，数千から数万次元の超高次元データから頻出パターンを発見する手法が提案されている．超高次元データの頻出パターンは長くなる傾向にあるため出力される頻出パターンは冗長で非常に多くなる傾向にある．そのため，これから述べる 2 つの手法では冗長なパターンを除去した飽和頻出アイテム集合（Closed Frequent Itemsets: CFI）のみ抽出している．

CARPENTER [4] は，レコード数が少なく，属性数が非常に大きいデータから CFI を効率的に発見する手法である．CARPENTER は，本手法と同じくレコードの組合せを探索するレコード空間探索を行って CFI を発見する．効率よく探索空間を枝刈りするために転置テーブルと呼ぶ元のデータベースを変換したデータベースを用いるのが特徴である．転置テーブルは，表 2 のデータベースにおいてトランザクションとアイテムを入れ替えたデータベースであり，あるアイテムがどのトランザクションに出現しているかを表している．[4] では，レコード数が 181，属性数が 12533 の遺伝子データを対象に実験を行い，CFI 発見の代表的な手法である CHARM [5] に比べて高速であることを示している．

TD-Close [6] は，CARPENTER を改良した手法である．レコード空間探索，転置テーブルを用いるのは CARPENTER と同じだが，トップダウン探索を行う点が異なる．CARPENTER では，レコード集合が短い方から長い方へとボトムアップに探索を進めるが，TD-Close では長い方から短い方へとトップダウンに探索を進める．[6] では，レコード数が 38，属性数が 7129 の遺伝子データを対象に実験を行い，CARPENTER に比べて高速であることを示している．

4. 提 案 手 法

提案手法は，レコード数が少なく，属性数が非常に大きいデータを対象にし，レコード空間探索を行うという点で CARPENTER と似ている．また Apriori と同様に最小サポートカ

```
Input:
  D:database
  minsup:minimum support count
  minlen:minimum pattern length
Output:
  LFP: longest frequent patterns
  F: frequent patterns
Method:
  N[1] = gen_1-ridsets(D)
  for (k=2; k<=minsup and N[k]!=∅; k++) {
      C[k] = gen_ridset(N[k-1])
      N[k] = {n|len(n.itemset)>=minlen for n in C[k]}
  }
  F' = {n.itemset for n in N[minsup]}
  LFP = arg max_i len(i) for i in F'
  F = {s|s>=minlen for s in subset(F')}
  return F
```

図 3　提案手法の擬似コード

ウントを指定し，データベース中から最小サポートカウント以上のサポートカウントを持つパターンを抽出する点も同じである．

異なる点は，最小パターン長に基づく探索空間の枝刈り，最長頻出パターンまたは指定した最小パターン長以上の長さを持つ頻出パターンのみ発見する点である．一般的にデータが高次元だと抽出される頻出パターン長は長く，冗長になりやすい．このような場合，短い頻出パターンよりは長い頻出パターンの方がより重要だと考えている．

本手法を用いると，最小パターン長以上の長さを持つすべての頻出パターンを発見できることを示すが，超高次元データの場合，これでもまだ冗長であるため本論文では最長頻出パターンを求めることに重点を置いている．

4.1　アルゴリズム

図 3 に擬似コードを示す．本手法では，$(ridset, itemset)$ のようにレコード集合とアイテム集合を対にして管理し，これをノードと呼ぶ．ノード n のレコード集合は $n.ridset$，アイテム集合は $n.itemset$ と表記する．以下，アルゴリズムの手順について述べる．

（1） 長さ 1 のレコード集合生成

長さ 1 のレコード集合を生成し，各レコードに含まれるアイテム集合を求める．生成したレコード集合とアイテム集合は対にしてノードとして管理する．このときアイテム集合長が最小パターン長に満たない場合，そのアイテム集合を含むノードは除去する．除去されずに残ったノードの集合を N_1 とする．

（2） 長さ k のレコード集合生成

次に，長さ k-1 のレコード集合から長さ k のレコード集合を生成する．生成手段は Apriori の Join 操作と類似しているが，アイテム集合ではなくレコード集合を生成するところが異なる．ノード集合 N_{k-1} に含まれるノードを n_1, n_2，ノードに含まれるレコード集合を $r_1 = n_1.ridset$, $r_2 = n_2.ridset$, $r_i[j]$ を r_i の j 番目のレコードとする．r_1 と r_2 が $(r_1[1] = r_2[1]) \wedge (r_1[2] = r_2[2]) \wedge \cdots \wedge (r_1[k-2] = r_2[k-2]) \wedge (r_1[k-1] < r_2[k-1])$ を満たすとき，r_1 と r_2 から長さ k のレコード集合 $r_{new} = r_1[1] \cdots r_1[k-2] r_1[k-1] r_2[k-1]$ を生成する．ここで，$<$ は辞書順を意味する．たとえば，012 と 013 からは 0123 が生成できる．

ここで，Apriori Property を用いてレコード集合を除去できる．生成されたレコード集合 r_{new} の空でないいずれかの部分集合がすでに除去されていたら r_{new} は除去できる．レコード集合の組合せにおいても Apriori Property が成り立つことは後ほど示す．

r_{new} に対応するアイテム集合 i_{new} は，共通集合操作を用いて $i_{new} = n_1.itemset \cap n_2.itemset$ で求める．生成した r_{new} と i_{new} はノード (r_{new}, i_{new}) として C_k に追加する．

（3） 最小パターン長によるノード除去

新しいノードを生成した後，最小パターン長に基づいてノードを除去する．ステップ (2) で生成したノード n に含まれるアイテム集合 $n.itemset$ の長さを計算する．アイテム集合長が指定した最小パターン長（minlen）より短い場合，そのアイテム集合を含むノードは除去する．アイテム集合長が指定した最小パターン長以上の場合，そのアイテム集合を含むノードを N_k に追加する．最小パターン長に基づいてノードを除去しても最小パターン長以上の長さを持つすべての頻出パターンが生成できることは後ほど示す．

以上の（2）と（3）の手順（パスと呼ばれる）を k が最小サポートカウント以下であり，かつ N_k が空集合でない限り繰り返す．

ここで，k が最小サポートカウントと等しいパスのノード N_k に含まれるアイテム集合を F' とおく．レコード空間探索では，パス k のノード N_k に含まれるアイテム集合のサポートカウントは k 以上である．よって，パス $minsup$ のノード N_{minsup} に含まれるアイテム集合のサポートカウントは minsup 以上，つまり頻出であるため F' は頻出パターンであることが保証されている．特に F' の中で最長のアイテム集合が最長頻出パターンである．F' から最小パターン長以上の長さを持つすべての部分集合を取り出したものを F とする．もし F に重複がある場合は除去する．このとき，F は最小パターン長以上の長さを持つすべての頻出パターンである．この理由は後ほど示す．

4.2　例　　題

図 4 は，図 1 のデータベースを用い，最小サポートカウントを 3，最小パターン長を 3 とした場合の提案手法の処理例である．Apriori と同様に，本手法でもトライを用いて探索を進める．Apriori の節点がアイテムであったのに対し，本手法では節点がレコードとなるのが特徴である．図 4 には，トライ全体を図示しているが，実際はレコード集合長 1 から順に枝を伸ばして探索を進める．

まず，長さ 1 のレコード集合である 0，1，2，3，4，5 を列挙し，各レコードに含まれるアイテム集合とアイテム集合長を求める．ここでは，最小パターン長 3 より短いアイテム集合はないため除去するノードはない．

次に，長さ 1 のレコードから長さ 2 のレコード集合である 01，02，03，04，$\cdots$，45 を生成する．このとき Join 操作によって生成したレコード集合のいずれかの部分集合がすでに除去されている場合は Apriori Property により生成しなくてよ

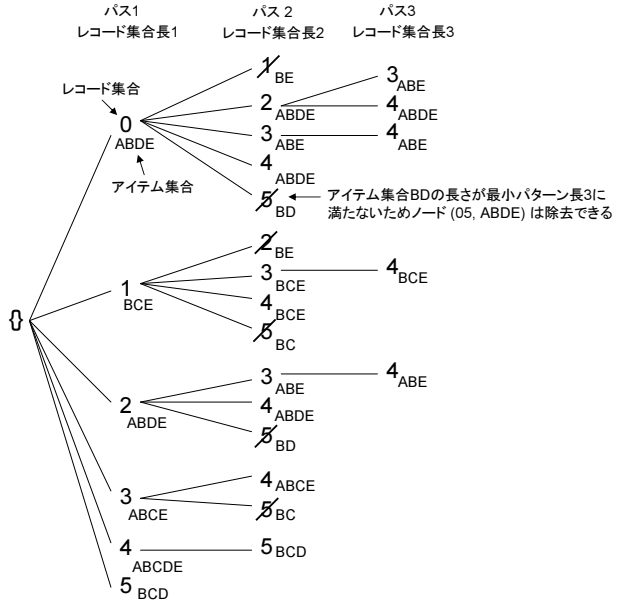

図 4　提案手法の処理例

い．今回の例では Apriori Property によって除去できるノードはないが，たとえば，03 を生成したとき，03 の部分集合 0，3 のいずれかが前のパスですでに除去されていた場合，03 は生成する必要はない．

生成したレコード集合に対応するアイテム集合は個々のレコードに対応するアイテム集合の共通集合をとる．たとえば，レコード集合 0 に対応するアイテム集合が ABDE，レコード集合 1 に対応するアイテム集合が BE なのでレコード集合 01 に対応するアイテム集合は ABDE と BE の共通集合である BE となる．

次に，長さ 2 のレコード集合に対応するアイテム集合長を求める．このとき，01 に対応するアイテム集合 BE の長さは 2 であり，最小パターン長 3 に満たないため対応するレコード集合 01 はノードごと除去する．レコード集合 05，12，15，25，35 も同様の理由でノードごと除去する．

次に，長さ 2 のレコード集合から長さ 3 のレコード集合を生成する．このとき，Apriori と同様 Join 操作が可能である．たとえば，02，03，04 からは 023，024，034 が生成される．ここでも Apriori Property によって生成するかしないかを判断する．たとえば，034 を生成したとき，034 の部分集合 03，04，34 のいずれかのノードが前のパスですでに除去されていた場合，034 は生成する必要はない．ここで，034 の部分集合として 0，3，4 については除去されているか調べなくてよい．なぜなら，1 つ前のパスで 03 を生成するときに 0，3 が除去されているか，04 を生成するときに 0，4 が除去されているかすでに調べているためである．

023 に対応するアイテム集合は，02 と 03 の共通アイテム集合をとる．02 に対応するアイテム集合が ABDE，03 に対応するアイテム集合が ABE なので 023 に対応するアイテム集合は ABDE と ABE の共通集合である ABE となる．

この例では，最小サポートカウントが 3 であるためパス 3 で探索を打ち切る．

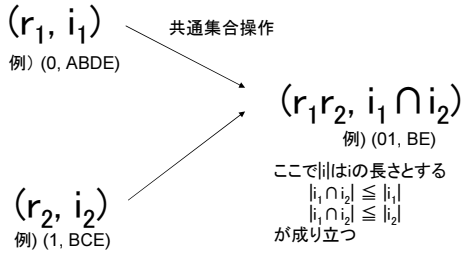

図 5　共通集合操作の性質

最後に探索結果から最長頻出パターンと最小パターン長 3 以上の長さを持つすべての頻出パターンを生成する．

最小サポートカウントが 3 であるためレコード集合長が 3 のレコード集合に対応するアイテム集合 F ’ = {ABE, BCE, ABDE} を抽出する．F’ に重複がある場合は除去する．このとき，F’ に含まれるアイテム集合の最小サポートカウントは 3 以上であることが保証されている．たとえば，ABE について考えてみる．ABE に対応するレコード集合は 023 である．アイテム集合はレコード集合に対応するアイテム集合の共通集合をとってきたことから，レコード 0，2，3 に共通するアイテム集合が ABE といえる．つまり，ABE は少なくともレコード 0，2，3 に含まれており，ABE のサポートカウントは少なくとも 3 であることがわかる．

ここで，ABE のサポートカウントが 3 であるとは言い切れない．実際，ABE は，レコード 0，2，3，4 に含まれておりサポートカウントは 4 である(注1)．しかし，ここでは，アイテム集合が頻出であるか，頻出でないかわかればよいので ABE のサポートカウントが最小サポートカウントの 3 以上であることが保証されていればよい．

F’ の中で最も長い ABDE が最長頻出パターンである．最小パターン長以上の長さを持つすべての頻出パターンは，長さが 3 以上の F' の全部分集合 F = {ABE, BCE, ABD, ADE, BDE, ABDE} である．

F の各アイテム集合の部分集合，たとえば，A，B，C，E，AB，BC なども頻出パターンではあるが，本手法では長さが最小パターン長より短い「すべての」頻出パターンが抽出できる保証はない．

4.3　証　　明

提案手法のいくつかの点について根拠を示す．

(1) レコード空間探索において Apriori Property が成り立つ理由

レコード空間探索における Apriori Property とは，レコード集合 A のある部分集合がすでに除去されていれば A も除去できるという性質である．

図 5 のようにアイテム集合はレコード集合間の共通集合操作によって生成するためレコード集合長が長くなると対応するアイテム集合長は操作の元になるアイテム集合長と同じかより短くなる．

(注1)：トライをパス 4 まで伸ばせばノード (0234, ABE) が生成されることがわかる．

この共通集合操作の性質からレコード集合 A のある部分集合 B がすでに最小パターン長を満たさず除去されていたなら，B よりもレコード集合長が長い A に対応するアイテム集合長が B に対応するアイテム集合長より長くなることはない．このことから，A は最小パターン長を満たすことがないため必然的に除去される．ゆえに，レコード空間探索においても Apriori Property が成り立つ．

(2) 最小パターン長によるノード除去で最小パターン長以上の長さを持つすべての頻出パターンが残る理由

ノードの除去を行わない場合，レコード空間探索によってすべての長さのレコード集合が列挙されることから，すべての頻出パターンは必ずいずれかのレコード集合に対するアイテム集合に含まれていると考えられる．なぜなら，頻出パターンのサポートカウントはそのアイテム集合が出現するレコードの数であり，対応するレコード組合せはレコード空間のいずれかに存在するからである．

すべての頻出パターンがトライ上に必ず出現するため，最小パターン長によるノードの除去を行っても最小パターン長以上の長さを持つ頻出パターンが除去されないことを示せばよい．

(1) で示したレコード空間探索における Apriori Property の性質から，最小パターン長より短いアイテム集合を持つノードの先を探索してもこれ以上アイテム集合長が長くなるノードが出現することはないことがわかる．ゆえに，最小パターン長より短いアイテム集合を持つノードを除去しても最小パターン長以上の長さを持つ頻出パターンのノードが除去されることはない．

すべての頻出パターンが探索空間に出現し，かつ最小パターン長以上の長さを持つ頻出パターンのノードが除去されないことから，最小パターン長以上の長さをもつすべての頻出パターンが除去されず残る．

(3) 最小サポートカウントのパスで探索を打ち切ってよい理由

(2) で示したように最小パターン長以上の長さをもつすべての頻出パターンが除去されず残る．また，レコード空間探索ではレコード集合長がサポートカウントにあたり，パス minsup に含まれるアイテム集合のサポートカウントは minsup 以上，つまり頻出であることが保証されている．よって，最小パターン長以上の長さをもつすべての頻出パターンが除去されず残り，かつパス minsup のアイテム集合は頻出なのでパス minsup に最小パターン長以上の長さを持つすべての頻出パターンが存在するはずである．

次にパス minsup+1 まで探索しなくてよいことを示す．(1) で示したように，パス minsup+1 まで枝を伸ばした場合，minsup+1 のアイテム集合の長さはパス minsup のアイテム集合の長さと同じかより短くなる．よって，最小パターン長以上の長さをもつすべての頻出パターンを見つけるためにパス minsup+1 まで探索する必要はなく，パス minsup までの探索で十分である．以上のことから，最小サポートカウント，最小パターン長の両条件を満たす最小パターン長以上の長さを持つすべての頻出パターンはすべてパス minsup に含まれている．ゆえに，パス minsup で探索を打ち切ってよい．

表 1　テストデータセット

データ名称	レコード数	平均レコード長
sub chess	50	37
T1000.D50	50	1000
T1000.D100	100	1000
ALL	43	12625

5. 実験と結果

本章では，提案手法をテストデータセットに適用し，実行時間を測定した結果について述べる．実験に用いた計算機は，CPU が Pentium4 3.2GHz，メモリが 1GB である．プログラムは C++で記述した．

5.1 テストデータセット

表 1 に示す 4 つのデータセットに対して本手法を適用して評価した．平均レコード集合長は，1 つのレコードに含まれるアイテム数の平均値である．

sub chess は，UCI Machine Learning Repository (注2) に含まれる chess データを元にした．chess は非常に密なデータであり，かつ発見される頻出パターン長が長いため Apriori で頻出パターンを発見するのは困難なデータである．元の chess データはレコード数が 3,196 と提案手法の適用対象としては大きいため最初の 50 レコードのみ抽出した．T1000.D50 と T1000.D100 は IBM Quest Synthetic Data Generation Code (注3) を用いて生成した．レコード数と平均レコード長は表 1 の値を用い，それ以外のパラメータはデフォルト値とした．

Synthetic Data では，アイテム数を一定にし，平均レコード長を大きくすることによって高次元データを表現した．これは，多値属性は必ず離散化したどれか 1 つの値を取るため属性数に比例してレコード長が大きくなると考えたためである．平均レコード長が長いほどデータが密になり，かつ発見される頻出パターン長も長くなる傾向にあるためより難しい課題となる．

ALL は，Acute Lymphoblastic Leukemia (ALL) と呼ばれる遺伝子データセット(注4) である．各属性の値は連続値であるため離散化手法の一種である equidepth 方式 [2] により各属性を 10 分割して離散化した．ALL は，先の 3 つのデータに比べて高次元で密なデータである．

予備実験としてこれらのデータセットから Apriori を用いて頻出パターンの抽出を試みたが T1000.D50（最小サポート 10%，20%）を除いて頻出パターンを実時間で発見することはできなかった．

5.2 実行時間

各データセットにおいて最長頻出パターンを抽出するまでの実行時間を図 5 に示す．実行時間は 5 回計測したうちの最速のものを選んだ．

図 6 においてグラフ上の数値は設定した最小パターン長を表

(注2)：http://www.ics.uci.edu/ mlearn/MLRepository.html
(注3)：http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/mining.shtml
(注4)：http://www.stjuderesearch.org/data/ALL1/

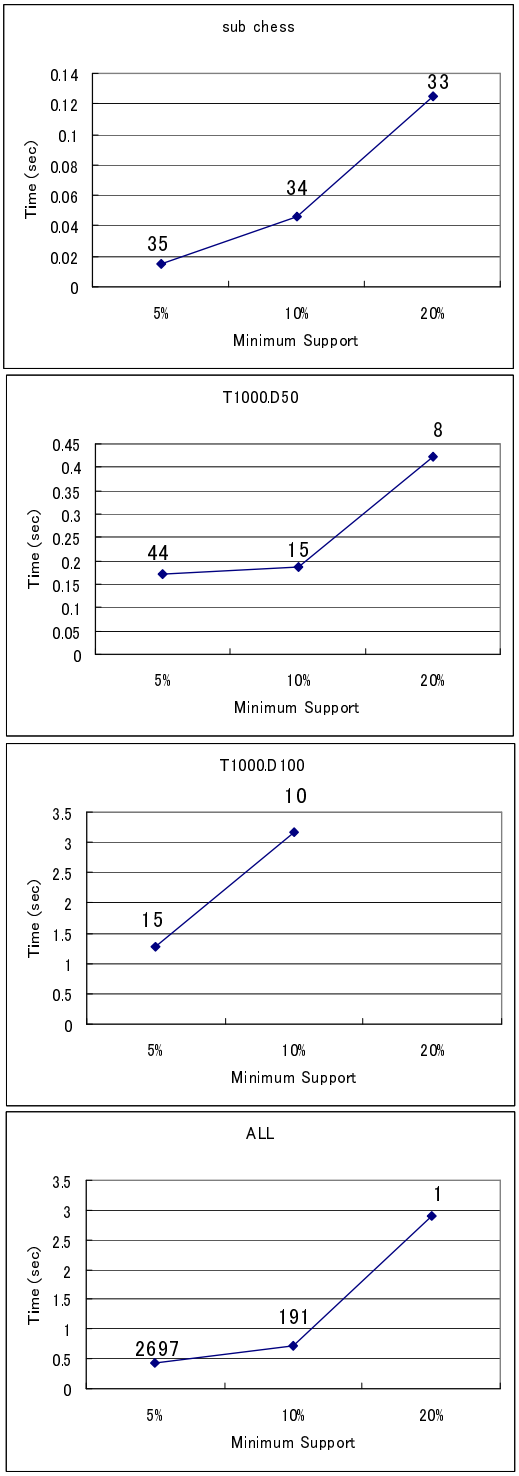

図 6　実行時間

している．各最小サポートにおいて最小パターン長を変えているのは，本手法が主に最長頻出パターン集合を求めることに焦点を当てているためである．本実験では，最小パターン長は，最長頻出パターン集合が 1 つ以上見つかるという条件でもっとも大きい値に設定している．つまり，実行時間は最長頻出パターンを見つけるのに必要な最短時間を表している．

最長頻出パターンを見つけたい場合，最小パターン長はできるだけ大きくしたほうがよいが，大きくしすぎると頻出パターンが見つからなくなるため適切に設定する必要がある．一般に最小サポートが大きいほど頻出パターンは短くなるため最小パターン長を小さくせざるを得ない．そのため，最小サポートが大きいほど最小パターン長は小さく設定している．

6. 考　　察

本章では，実験結果によって明らかになった本手法の特徴と問題点について考察する．

最小サポートが小さいほど実行時間が短い

図 6 において，最小サポートが小さいほど実行時間が短いという結果は，Apriori の結果とは逆であることを示している．これには 2 つの理由が考えられる．

1 つめの理由は，本手法がレコード空間探索を行っているためである．レコード空間探索では，トライの深さがサポートカウントにあたる．最小サポートカウントが小さいほどトライの浅いところで探索を打ち切ることができるため実行時間が短くなる．最小サポートカウントが大きいとトライの深いところまで探索を進める必要があり，レコードの組合せが指数関数的に大きくなるため計算時間が長くなる．

2 つめの理由は，Apriori がすべての頻出パターンを発見しているのに対し，本手法は主に最長頻出パターンのみ発見することを目的としているためである．Apriori と直接実行時間を比較しなかったのはこのためである．

本手法は，最長頻出パターンだけではなく，最小パターン長以上の長さを持つすべての頻出パターンも求められるが，計算時間が非常に長くなる場合がある．たとえば，図 6 の ALL データでは，長さが 2697 の最長頻出パターンが 1 つ見つかっている．最小パターン長を 2697 とした場合，最小パターン長以上の長さを持つすべての頻出パターンは 1 つだが，最小パターン長を 2690 とした場合，最小パターン長以上の長さを持つすべての頻出パターンを生成しようとすると長さが 2697 の最長頻出パターン 1 つからだけでも ${}_{2697}C_{2690} \approx 10^{20}$ 個の頻出パターンが生成される．このような莫大な数の頻出パターンを現実的な時間ですべて生成することは不可能であるし，たとえ生成できたとしても解釈できない．このような理由から本論文では最長頻出パターンのみ発見することに重点を置いている．

一方，Apriori で最長頻出パターンを発見したい場合，すべての長さの頻出パターンを列挙しなければならない．長さが 2697 の最長頻出パターンを見つけるためには長さが 2697 より小さいすべての頻出パターンを求める必要がある．図 2 のトライの例で言うと，木の深さが 2697 の場所まで探索を進める必要があり現実的ではない．このような点から頻出パターン長が長いデータほど本手法が有効だと考えられる．

最小パターン長が大きいほど高速

図 6 において，最小パターン長が大きいほど高速なのは，トライ上の探索においてノードが枝刈りされやすくなるためである．パスの速い段階でノードが枝刈りされるほど探索の効率はよくなる．

現状では，適切な最小パターン長を決定する方針がないこと

が問題となっている．最小パターン長は大きければ大きいほど枝刈りが効率よく行われるため望ましいが，あまりに大きくしすぎると頻出パターンが見つからなくなってしまう．本論文の実験では，頻出パターンが発見できるように大きい値から小さい値へと試行錯誤的に最小パターン長を見つけている．

IBM Quest Synthetic Data Generation Code で生成する人工データセットはあらかじめ頻出パターンの平均長を指定できるため最小パターン長を設定する手がかりになる．しかし，一般的なデータでは，頻出パターンの平均長があらかじめわからないため最小パターン長の適切な設定は難しい問題となる．頻出パターンの平均長，または最長頻出パターン長を概算する手法の開発は今後の課題である．

レコード数が大きいほど計算時間が長い

図 6 において，T1000.D100 のようにレコード数を大きくすると計算時間が長くなるのは本手法がレコードの組合せを探索しているためである．レコード数が大きいほどレコード組合せの数は大きくなるため計算時間が長くなる．Apriori はレコード数が大きく属性数が小さい場合に有効な手法であるが，本手法はこれと逆の特徴を持ち，レコード数が少なく属性数が大きい場合に有効な手法となっている．

現在，ノードの除去に用いている方法は，最小パターン長と Apriori Property のみである．レコード空間探索において有効な枝刈り手法の開発は今後の課題である．

密なデータに対しても高速

図 6 において，sub chess や ALL のように密なデータに対しても高速なのは，候補アイテム集合の生成に共通集合操作を用いており，アイテムの組合せを生成しないためである．共通集合操作は高速に行えるためアイテム集合長が数万のオーダーでも十分対応できる．2 つのアイテム集合の共通集合操作を並列化することによってさらに高速化が可能だと考えている．

7. おわりに

本論文では，レコード数が少なく，属性数が極めて大きなデータから効率的に頻出パターンを発見する手法を提案した．本手法は，レコード空間探索，最小パターン長による枝刈り，レコード空間探索における Apriori Property を用いて最長頻出パターンを高速に発見することができる．実験用のデータセットで評価したところ高次元で密なデータに対して有効にはたらくことが確認できた．

本手法を用いることによって従来手法で扱えなかった高次元データを分析できる可能性がある．たとえば，項目数が多いアンケートデータ，センサーデータ，テキストデータ，画像データ，音楽データ，遺伝子データなどが候補として考えられる．

今後の課題として，並列分散化，時系列データへの拡張，最小パターン長の設定方針の検討，具体的な問題への応用などを考えている．

文　献

[1] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, Proc. of Intl. Conf. on Very Large Data Bases, pp.487-499, 1994.

[2] J. Han and M. Kamber, Data Mining Concepts and Techniques, Morgan Kaufmann Pub, 2000.

[3] R. J. Bayardo Jr, Efficiently Mining Long Patterns from Databases, Proc. of Intl. Conf. on Management of Data, pp.85-93, 1998.

[4] F. Pan, G. Cong, J. Yang and M. J. Zaki, CARPENTER: Finding Closed Patterns in Long Biological Datasets, Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery, 2003.

[5] M. J. Zaki and C. J. Hsiao, CHARM: An Efficient Algorithm for Closed Association Rule Mining, Proc. of Intl. Conf. on Data Mining, pp.457-473, 2002.

[6] H. Liu, J. Han, D. Xin and Z. Shao, Mining Frequent Pattern from Very High Dimensional Data: A Top-Down Row Enumeration Approach, Proc. of SIAM Conf. on Data Mining, 2006.

[7] R. J. Bayardo Jr, Efficient Mining Long Patterns from Databases, Proc. of Intl. Conf. on Management of Data, pp.85-93, 1998.