

# 超高次元データからの頻出パターン発見手法

## Frequent Pattern Mining Method from Super High Dimensional Data

森 紘一郎<sup>◆</sup>      折原 良平<sup>▲</sup>

Kouichirou MORI      Ryohei ORIHARA

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。

Traditional frequent pattern mining methods such as Apriori have a problem - the order of calculation exponentially grows with high dimensional data because of search by the combination of attribute sets. The purpose of our work is 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 efficiently search the space. As a result, we can extract the frequent patterns in real time from high dimensional datasets that cannot be handled by Apriori. We conclude that our method is efficient for frequent pattern mining from high dimensional data.

### 1. はじめに

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。

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の計算が困難

という問題があった。

そこで、本論文では、レコード数が小さく（数百のオーダー）、属性数が極めて大きい（数万のオーダー）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。本手法の特徴は、レコードの組合せを探索するレコード空間探索と最小パターン長の導入による探索空間の枝刈りである。

### 2. 関連研究

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生成するボトムアップ型のアルゴリズムである。

Aprioriでは、候補パターンの生成時に属性の組合せを生成して頻出パターンを探索する属性空間探索と呼ばれる手法を用いている。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、アイテムの組合せが爆発的に大きくなり、適用するのが難しいという問題点が指摘されている[2]。また、データが密の場合、頻出アイテム集合長Lが長くなる傾向にある。そのため長さLの頻出アイテム集合Iを求める前に2個のIの部分集合をすべて生成しなければならず、計算時間が長くなるという問題点もある。

長い頻出パターンを発見するのが困難というAprioriの問題点に対して、極大頻出アイテム集合 (MFI) を発見する Max-Miner[2]が提案されている。しかし、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言及がない。

近年、数千から数万の超高次元データから頻出パターンを発見する手法としてCARPENTER[3]とTD-Close[4]が提案されている。これらの手法は、飽和頻出アイテム集合 (CFI) を発見するCHARM[5]を元にした手法である。提案手法と同様、属性の組合せではなく、レコードの組合せを探索し、元のデータベースのレコードと属性を入れ替えた転置テーブルを作成して効率的に探索を行うのが特徴である。

### 3. 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としており、レコードの組合せを探索するという点でCARPENTERやTD-Closeと類似している。異なる点は、転置テーブルを作成する必要がないことと、最小パターン長に基づく探索空間の枝切りの導入である。また、本手法で発見するのは、ユーザが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である。一般的にデータが高次元になると抽出される頻出パターン長は長くなりやすい。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が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。

本論文では、アイテム集合 (itemset) とレコード集合 (ridset) という用語を用いている。例えば、図2のデータベースでは、{A, B, C}はアイテム集合、{1, 5}はレコード集合である。それぞれ、ABC, 15と略記する。

◆ 正会員 東芝研究開発センター  
[kouichirou1.mori@toshiba.co.jp](mailto:kouichirou1.mori@toshiba.co.jp)

▲ 東芝研究開発センター  
[ryohei.oriyara@toshiba.co.jp](mailto:ryohei.oriyara@toshiba.co.jp)

```

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]!=0; 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 = argmaxi len(i) for i in F'
F = {s|s>=minlen for s in subset(F')}
return F

```

図1 提案手法

Fig. 1 Our Method

### 3.1 アルゴリズム

図1に提案手法の擬似コードを示す。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、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 $r_1=n_1.ridset, r_2=n_2.ridset, r_1[j]$ を $r_1$ の $j$ 番目のレコードとする。 $r_1$ と $r_2$ が $(r_1[1]=r_2[1]) \wedge (r_1[2]=r_2[2]) \wedge \dots \wedge (r_1[k-2]=r_2[k-2]) \wedge (r_1[k-1] < r_2[k-1])$ を満たすとき、 $r_1$ と $r_2$ から長さ $k$ のレコード集合 $r_{new}=r_1[1] \dots r_1[k-2]r_1[k-1]r_2[k-1]$ 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[6]に類似した手法を用いてレコード集合を除去できる。Apriori Propertyは、Aprioriの枝刈り手法で「アイテム集合 $I$ のいずれかの部分集合が頻出でなければ、 $I$ もまた頻出ではない」という性質である。本手法では、「生成されたレコード集合 $r_{new}$ のいずれかの部分集合がすでに除去されていたら $r_{new}$ もまた除去できる」と変形して用いている。レコード集合の組合せにおいてこの性質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)  $k$ 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。繰り返し終了後、 $k$ 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。

### 3.2 例題

図3は、図2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。本手法では木構造を用いて探索を進める。図3には、探索木の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, ..., 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。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。以下、同様の手順で探索を進める。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。

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

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。 $F'$ 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。

RID	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

図2 トランザクションデータベース  
Fig.2 Sample transaction database

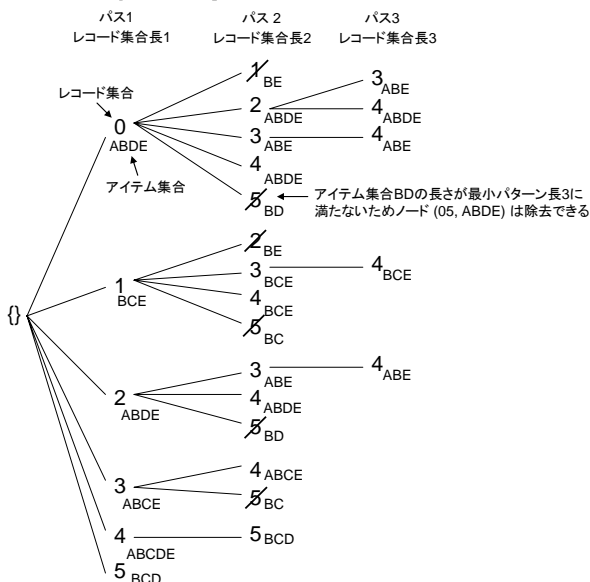


図3 提案手法の処理例

Fig.3 Example of our algorithm

### 3.3 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。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。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探索木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に残る。

## 4. 実験と考察

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Xeon 2.4GHz、メモリが2GBである。プログラムはC++で記述した。

### 4.1 テストデータセット

表1に示す2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。

T1000.D100は、IBM Quest Synthetic Data Generation Code<sup>1</sup>を用いて生成した。指定したパラメータ以外はデフォルト値である。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と呼ばれる遺伝子データセット<sup>2</sup>である。各属性値は連続値であるため離散化手法であるequidepth方式により各属性を10分割して離散化した。

### 4.2 実行時間

図4は、提案手法とApriori、CHARMの実行時間を比較した

表1 テストデータセット

Table.1 Test dataset

データ名	レコード数	平均レコード長
T1000.D100	100	1000
ALL	43	12625

<sup>1</sup>[http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data\\_mining/mining.shtml](http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/mining.shtml)

<sup>2</sup><http://www.stjudereseearch.org/data/ALL1/>

グラフである。データセットALLに関してはAprioriではメモリーオーバーでクラッシュしたため図示していない。

Aprioriはすべての頻出パターンを発見するまでの時間、CHARMはCFIを発見するまでの時間、提案手法は最長頻出パターンを求めるまでの時間を測定した。本来、各アルゴリズムの求める対象が異なるため実行時間を直接比較できないが、提案手法の特徴を明らかにするために比較データを載せた。

提案手法では、最小サポートカウントのほかに最小パターン長を設定する必要がある。最小パターン長は、頻出パターンが1つ以上見つかるという条件を満たす中で最大値を設定した。T1000.D100では、最小サポートカウント5のときに15、20のときに4と設定した。一方、ALLでは、最小サポートカウント2のときに2697、5のときに4と設定した。

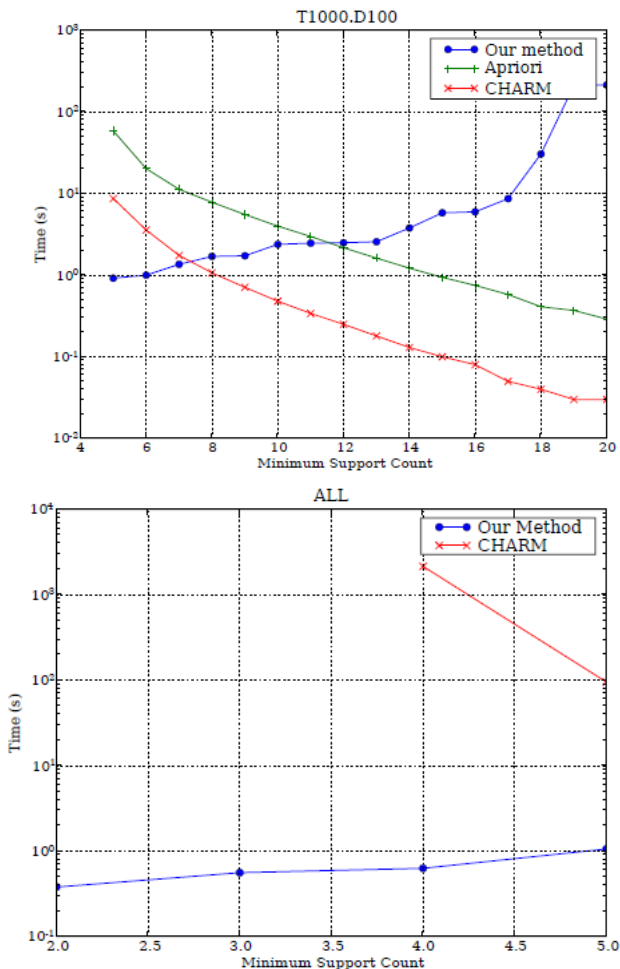


図4 実行時間

Fig.4 Execution Time

### 4.3 考察

図4は、本手法が最小サポートカウントが小さいほど実行時間が短いというAprioriやCHARMとは逆の結果を示している。これは、本手法がレコード空間探索を行っており、最小サポートカウントが小さいほど探索木の浅いところで探索を打ち切れるためである。逆に、最小サポートカウントが大きいほど計算時間が増加してしまう。

T1000.D100とALLの結果を比較すると、本手法は平均レコード長が長く、データが密で、最小サポートカウントが小さいほど有効なことがわかる。一般的に平均レコード長が長く、密なデータは、頻出パターン長が長くなりやすい。た

例えば、ALLは最小サポートカウント4のとき、長さ194の頻出パターンが見つかった。このような長い頻出パターンの発見は、Aprioriのような属性空間探索の手法では困難である。提案手法は、レコード空間探索を行い、パターンの生成は、単なる共通集合を取るだけなのでパターン長が長くても高速に計算できる。

### 5. 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、効率的な枝刈り手法の開発、具体的な問題への応用などを考えている。

### 【文献】

- [1] Agrawal, R. and Srikant, R.: "Fast Algorithms for Mining Association Rules", in Proc. of the Intl. Conf. on Very Large Data Bases, pp.487-499 (1994).
- [2] Bayardo, R. J.: "Efficiently Mining Long Patterns from Databases", in Proc. of ACM SIGMOD Intl. Conf. on Management of Data, pp.85-93 (1998)
- [3] Pan, F., Cong, G., Yang, J. and Zaki, M. J.: "CARPENTER: Finding Closed Patterns in Long Biological Datasets", in Proc. of the ACM SIGKDD Intl. Conf. on Knowledge Discovery (2003).
- [4] Liu, H., Han, J., Xin, D. and Shao, Z.: "Mining Frequent Pattern Very High Dimensional Data: A Top-Down Row Enumeration Approach", in Proc. of SIAM Conf. on Data Mining (2006).
- [5] Zaki, M. J. and Hsiao, C. J.: "CHARM: An Efficient Algorithm for Closed Association Rule Mining", in Proc. of the Intl. Conf. on Data Mining, pp.457-473 (2002).
- [6] Han, J. and Kamber, M.: "Data Mining: Concepts and Techniques", Morgan Kaufmann Pub. (2000).

### 森 紘一郎 Kouichirou MORI

2005年早稲田大学大学院理工学研究科修士課程修了。同年(株)東芝入社。現在、同社研究開発センターシステム技術ラボラトリーに勤務。主に機械学習、データマイニングの研究開発に従事。情報処理学会、日本データベース学会、人工知能学会各会員。

### 折原 良平 Ryohei ORIHARA

1988年筑波大学大学院工学研究科修士課程修了。同年、(株)東芝入社。現在、同社研究開発センターシステム技術ラボラトリーに勤務。2005年より東京工業大学連携准教授。博士(工学)。発想支援技術、類推、機械学習、データ・テキストマイニングの研究に従事。情報処理学会、日本ソフトウェア科学会、人工知能学会各会員。