密なデータベースに対する動的な探索順序を用いた高速な顕在パターンマイニング手法

Efficient Constrained Pattern Mining Using Dynamic Item Ordering for Explainable Classification

岩下 洋哲 [1*]　高木 拓也 [1]　鈴木 浩史 [1]
後藤 啓介 [1]　大堀 耕太郎 [1]　有村 博紀 [2]
Hiroaki Iwashita[1]　Takuya Takagi[1]　Hirofumi Suzuki[1]
Keisuke Goto[1]　Kotaro Ohori[1]　Hiroki Arimura[2]

[1] 株式会社富士通研究所 [2] 北海道大学
[1] Fujitsu Laboratories Ltd. [2] Hokkaido University

Abstract: Learning of interpretable classification models has been attracting much attention for the last few years. Discovery of succinct and contrasting patterns that can highlight the differences between two classes are very important. Such patterns are useful for human experts, and can be used to construct powerful classifiers. In this paper, we consider mining of minimal emerging patterns from high-dimensional data sets under a variety of constraints in supervised setting. We focus on an extension in which patterns can contain negative items that designate the absence of an item. In such a case, a database becomes highly dense, and it makes mining more challenging since popular pattern mining techniques such as fp-tree and occurrence deliver do not efficiently work. To cope with this difficulty, we present an efficient algorithm for mining minimal emerging patterns by combining two techniques: dynamic variable-ordering during pattern search for enhancing pruning effect, and the use of a pointer-based dynamic data structure, called dancing links, for efficiently maintaining occurrence lists. Experiments on benchmark data sets showed that our algorithm achieves significant speed-ups over emerging pattern mining approach based on LCM, a very fast depth-first frequent itemset miner using static variable-ordering.

1　はじめに

異なるクラスに属するデータについて，クラスを識別する特徴的なパターンの発見は，データに潜む性質・傾向の分析やクラス分類問題へ応用されるデータマイニング分野における重要な基礎問題の一つである [1,2,5].

一般的に抽出対象のパターン空間は入力サイズに対して指数的に大きく，そのような膨大なパターン空間の中から与えられた制約を満たすパターンを高速に見つけるパターン探索アルゴリズムの開発が技術的な課題となる．これまで提案されてきたアルゴリズムは抽出対象となるデータベースの疎性を利用し高速なパターン抽出を実現してきた [2,8]．データベースが疎であるとは，アイテム集合サイズに対して，データベースの各レコードが持つアイテム数の割合が小さいことを指す．反対にデータベースが密であるとは，アイテム集合サイズに対して，各レコードが持つアイテム数の割合が大きいことを指す．例えばユーザーの購入履歴をレコードとして持つ商品購買データベースでは，各ユーザーの購入商品数は全商品数に対して限りなく小さく疎なデータベースになる傾向がある．従来手法は疎なデータベースについて高速に動作するものの，密なデータベースに対しては現実的な時間で計算を行うことが難しい．

本論文ではパターン抽出の探索過程においてアイテム探索順序の動的化による新たなプルーニング手法を提案し，密なデータベースに対しても高速なパターン抽出を可能にする．このプルーニング手法を行うためには動的なアイテム探索順序で探索しつつ，探索の各ステップにおいてそのとき注目しているパターンに合致するレコード情報を効率的に保存する必要がある．レコード情報をダンシングリンク [4] で保存し，この要件を満たすデータ構造，動的縮約可能バイナリ行列 **DRMX**

*連絡先：株式会社富士通研究所
川崎市中原区上小田中 4-1-1
E-mail: iwashita.hiroak@fujitsu.com

(Dynamic Reducible MatriX) を提案する．DRMX を用いることでアイテムの効率的な動的順序探索が可能になり高速なパターン抽出が可能となる．

本手法の提案による副産物として，否定アイテムを含んだより表現力の高いパターンマイニングも可能となる．例えば商品購買データベースなどで，あるアイテムを「購入しなかった」という情報はあるアイテムを「購入した」という情報と同等に重要な場合がある．両者の特徴を捉えるためには各商品について「購入した」アイテムに加えその否定を表す「購入しなかった」否定アイテムをデータベースへ追加しパターン抽出を行えば良い．このようなデータベースは密になり従来は計算が困難であったが，提案手法を用いることでこのような表現力の高いパターンも現実的な時間で計算可能になる．

計算機実験により提案手法の有効性を検証する．まず，静的順序探索と動的順序探索をした場合のパターン抽出に要した時間を比較し，動的順序探索がデータベースが疎であるか否かに関わらず高速化に貢献し，特に密である場合に劇的に効果的であることを確認する．次に既存手法である LCM [8] および CP-tree [2] と提案手法のパターン抽出速度の比較を行い，ほとんどのデータセットにおいて提案手法がより高速，特に密なデータセットの場合その傾向が顕著であることを確認する．最後に否定アイテムを含んだパターンを基にした分類モデルがロジスティック回帰や決定木，ランダムフォレストなどの既存の分類モデルと比較し高い分類精度を達成し，否定アイテムパターンの使用が分類問題において有効であることを確認する．

2 準備

$I = \{a_1, \ldots, a_n\}$ を n 個のアイテムの集合とする．I 上の**正負ラベル付きデータベース**をそれぞれ $D_+, D_- \subseteq 2^I$ で表し [1]，各データベースの要素を正ラベルレコード，負ラベルレコードと呼ぶ．また，各レコードを行，アイテムを列に対応させ，i 行 j 列要素を i 行のレコードが j 列のアイテムを要素として含むとき 1，そうでないとき 0 とした行列をデータベースのバイナリ行列表現と呼ぶ．I 上の任意のデータベース D，パターン $P \subseteq I$ に対して，D における P の出現リストを $Occ_D(P) := \{t \in D \mid P \subseteq t\}$ と定義する．また，正負ラベルデータベース $D = (D_+, D_-)$ におけるパターン P の出現頻度をそれぞれ $Sup_+(P) := |Occ_{D_+}(P)|$ と $Sup_-(P) := |Occ_{D_-}(P)|$ とする．

正負ラベル付きデータベース $D = (D_+, D_-)$，パターン P を入力として真偽値を返す関数 $\mathbb{C}$ を**制約**と呼ぶ．$\mathbb{C}(D, P)$ が真のとき，パターン P は D 上で制約 $\mathbb{C}$ を満たす制約パターンと呼ぶ．

本論文では，入力としてラベル付きデータベース $D = (D_+, D_-)$ と制約 $\mathbb{C}$ を受け取り，制約 $\mathbb{C}$ を満たすすべての制約パターンを求める制約パターンマイニング問題を取り扱う．

本論文で対象とする制約を以下に示す．

制約 1．コントラスト制約

任意の非負整数 $\sigma_+ \in [0..|D_+|]$ および $\sigma_- \in [0..|D_-|]$ に対して，パターン P が $Sup_+(P) \geq \sigma_+$ および $Sup_-(P) \leq \sigma_-$ を満たすときに限り P はコントラスト制約 $\mathbb{CP}[\sigma_+, \sigma_-]$ を満たすとする．$\mathbb{CP}[1, 0]$ の要素は特に**ジャンピング顕在パターン**と呼ばれる．

制約 2．Growth rate 制約

任意の非負実数 $\theta \in [0, \infty]$ に対して，パターン P が $GR(P \mid D_+, D_-) := Sup_+(P)/Sup_-(P) \geq \theta$ を満たすときに限り P は Growth rate 制約 $\mathbb{GR}[\theta]$ を満たすとする．$\mathbb{GR}[\theta]$ の要素は**顕在パターン**と呼ばれる．

制約 3．カイ二乗制約

任意の非負実数 $\gamma \in [0, \infty]$, に対して，パターン P が $\chi^2(P \mid D_+, D_-) \geq \gamma$ を満たすときに限り P はカイ二乗制約 $\mathbb{CHI}[\gamma]$ を満たすとする．ここで $\chi^2(P \mid D_+, D_-)$ はカイ二乗値である．情報利得やジニ係数などの統計量も，同様な制約として取り扱うことができる [6]．

制約 4．複合制約

2 つの制約 $\mathbb{C}_1$ および $\mathbb{C}_2$ を満たす制約は，それらの論理積 $\mathbb{C}(D, P) := \mathbb{C}_1(D, P) \wedge \mathbb{C}_2(D, P)$ で表現される．

制約 5．極小制約

制約 $\mathbb{C}$ について，$\mathbb{C}$ の極小制約 $\mathbb{MIN}\,\mathbb{C}$ は，$\mathbb{C}$ を満たす制約パターンのうち，極小なパターンのみを真とする制約である．つまり，パターン P が $\mathbb{MIN}\,\mathbb{C}$ を満たすとは，(i) P が $\mathbb{C}$ を満たし，かつ (ii) P の任意の部分パターン $P' \subset P$ が $\mathbb{C}$ を満たさない場合に限る．

3 アルゴリズム

提案アルゴリズムは，既存手法の CP-tree [2] や LCM [8] などと同様に深さ優先探索アルゴリズムに基づいている．すなわち，アルゴリズムは空集合のパターンからアイテムを一つずつ追加し，事前に与えられた制約を満たすか逐次的に確認を行う．探索中のパターンが制約を満たさないならば，新たなアイテムを追加し再帰的に探索を繰り返す，もし制約を満たすならばそのパターンを出力，一つ前の状態へバックトラック（最後に追加したアイテムをパターンから削除），新たなアイテムを追加し再帰的に探索を行う．図 1 に示すような探索木と呼ばれる木構造を探索することになる．

この探索にはプルーニングと呼ばれる高速化手法が鍵となる．プルーニングとは，現在探索中のパターン

[1] 2^I は I の冪集合を表す．

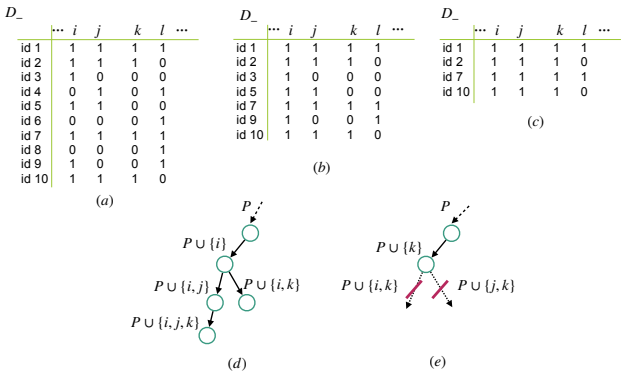

図 1: (a): パターン P が出現するレコードのみからなる負例データベース D_-．(b): パターン $P \cup \{i\}$ が出現するレコードのみからなる負例データベース D_-．(c): パターン $P \cup \{k\}$ が出現するレコードのみからなる負例データベース D_-．(d): アイテムの辞書順に従った探索木．(e): パターンの出現頻度に従った探索木．パターン $P \cup \{i, k\}$ および $P \cup \{j, k\}$ の D_- 上の出現頻度は $P \cup \{k\}$ から変化しないためプルーニング可能（斜線はそれ以上探索しないことを表す）．

にどのアイテムの組み合わせを追加しても制約を満たさないと判別できた時，それ以上の探索を中止しバックトラックすることである．効果的なプルーニング手法を複数利用することで探索の大幅な高速化が期待できる．

本章では，3.1 節にて対象とするプルーニングを紹介し，3.2 節にてプルーニング効果を促進するアイテム探索順序動的化のアイディアを紹介し，3.3 節にてアイテム探索順序動的化のためのレコード情報管理データ構造 DRMX とそれを用いたアルゴリズムの提案を行う．

3.1 プルーニング

提案アルゴリズムで使用するプルーニングを紹介する．

1：極小制約を用いたプルーニング

これは自明なプルーニングの一つである．極小制約以外の制約すべてを満たすパターン P が見つかった場合，極小性よりパターン $Q \supset P$ を探索する必要はなくプルーニングが可能である．

2: 頻度の上界と下界を用いたプルーニング

Morishita and Sese [6] によって提案されたものと同様[2]のプルーニングである．出現頻度の単調性より，$Sup_-(P \cup B)$ は負ラベルデータベース上の頻度の下界となる．また同様に $Sup_+(P)$ は正ラベルデータベース上の頻度の上界となる．したがって Growth rate 制約を例にすると，関数 $GR = Sup_+(P)/Sup_-(P)$ に対しては $Sup_+(P)/Sup_-(P \cup B)$ が上界となり，これが与えられたパラメータ θ よりも小さければプルーニングが可能である．この負ラベルデータベース上の頻度の下界と正ラベルデータベース上の頻度の上界を用いたプルーニングは，コントラスト制約やカイ二乗値制約でも同様に用いることができる．

3: 極小性と負例頻度変化量によるプルーニング

Loekito and Bailey [5] によって提案されたプルーニングである．与えられた制約が以下の条件を満たすと仮定する．

条件 C1: 任意のパターン P と任意のアイテム $a \in I$ に対して，もし $Sup_-(P) = Sup_-(P \cup \{a\})$ かつ $P \cup \{a\}$ が制約 $\mathbb{C}$ を満たすならば P も制約 $\mathbb{C}$ を満たす．

条件 C2: もし $Occ_+(P) = Occ_+(Q)$ かつ $Occ_-(P) = Occ_-(Q)$ ならば， P が制約 $\mathbb{C}$ を満たすとき，かつそのときに限り Q が制約 $\mathbb{C}$ を満たす．

このとき，以下の定理が成り立つ．

Theorem 1 ([5]) 制約 $\mathbb{C}$ が上記の条件 *C1* と *C2* を満たすと仮定する．もし $Sup_-(P) = Sup_-(P \cup \{a\})$ であるならば，任意の $(P \cup \{a\})$ の子孫パターン Z は $\mathbb{C}$ の極小性制約を満たさない．

この定理より，$Sup_-(P) = Sup_-(P \cup \{a\})$ であるときプルーニングが可能となる．

3.2 アイテムの動的探索

節 3.1 のプルーニング 2 とプルーニング 3 はそれぞれ，新たなアイテムをパターンに追加しても負ラベルデータベース上のパターンの出現頻度が大きく変わらない，もしくはまったく変化しないときに有効となる．図 1 の (c) で示すように，これはデータベースバイナリ行列の列がほとんどもしくは全て 1 であるような場合に発生する．提案手法では，負ラベルデータベース上のパターンの出現頻度がより減少するアイテムを優先して追加し，未探索の列に多くの 1 が残る状況を優先するようにアイテム探索順序の動的化を行い，早期のプルーニングの適用を狙う．たとえば，アイテム i および j が次の検索候補であり，$Sup_{D_-}(P \cup \{i\}) < Sup_{D_-}(P \cup \{j\})$ の場合，出現頻度が少ない $P \cup \{i\}$ を優先して探索する．

3.3 データベースの動的縮約とマイニングアルゴリズム

3.1 節のプルーニングを効率よく実行するための動的縮約可能バイナリ行列 DRMX（Dynamic Reducible MatriX）を提案する．3.1 節のプルーニングはパター

[2] [6] では出現頻度の下界を 0 としているが，我々はより良い下界を用いる．

ンの出現頻度や出現頻度の上界，下界を基に計算されるため，現在探索中のパターンに合致するレコード情報を保持し，アイテムの追加や復元したときのレコード情報を計算する必要がある．既存手法で用いられるデータ構造は探索順が静的であることを利用して効率的な計算を実現しており，効率を維持しつつ動的な探索順に変更することは難しい．DRMX は Knuth のダンシングリンク [4] を用いてデータベースをバイナリ行列として格納し，レコードとアイテムに対応する行と列に対して以下の操作を効率的にサポートする．

- $\mathbb{M} := DRMX.create(M)$: 指定されたトランザクションデータベース D を格納する新しい DRMX を作成する．
- $\mathbb{M}.deleteRow(i)$: i 番目の行を行列から削除する
- $\mathbb{M}.deleteColumn(j)$: j 番目の列を行列から削除する．
- $\mathbb{M}.checkpoint()$: 行列の現在の状態をアンドゥスタックにプッシュする．
- $\mathbb{M}.undo(i)$: アンドゥスタックから i 回ポップし，チェックポイント時点での行列 M の状態に復元する．
- $\mathbb{M}.countRows(P)$: 全ての $j \in P$ に対して j 番目の列の要素が 1 であるような行数を返す． $P = \emptyset$ の場合，行列 M の行数を返す．

DRMX を用いたマイニングアリゴリズムの擬似コードをアルゴリズム 1 に示す．アルゴリズム 1 では，行 3 で極小性以外の制約を満たす出力候補パターンを計算し，行 4 で候補の中から極小解を満たすパターンのみを抽出する．このパターン集合からの極小解の絞り込みは BDD を使用する手法 [7] を用いる．パターン集合から極小解を絞り込む方法は BDD を使用する手法が提案されており [7]，本論文でもその手法を用いている．アルゴリズム 2 がマイニングアルゴリズムのコア部分である．探索パターン P は空集合から始まり，行 18 で次に探索するアイテム i_* を選択し，行 21 と行 23 でそれぞれ i_* を用いないパターンと i_* を用いたパターンの場合に分けて再帰探索する．行 4，行 7 と行 10，行 12 はそれぞれ節 3.1 で示したプルーニングを行なっている．

4 実験結果

この節では次の 3 つの実験を行う．第一に静的変数順序と動的変数順をそれぞれ用いたマイニング速度の比較を行い，動的探索順序の優位性を評価する．第二に提案手法と既存のマイニングアルゴリズムとの速度比較を行う．最後に実際にマイニングしたパターンを使用して構築した分類モデルの精度を評価する．プログラムは全て C++を使用して実装した．また，実行環境は 400GB メモリと Intel Xeon E5-2680 v2 2.80GHz CPU を搭載した Linux ワークステーションで測定した．

Algorithm 1: 極小制約パターンマイニング

```
   input  : DRMX で表現された正例と負例のデータセット D+, D− ⊆ 2^I , アイテム集合 I，制約用のパラメータ Θ = (σ+, σ−, θ, γ)
   output : 極小制約パターン 𝕄𝕀ℕℂℙ
1  MiningMCP(D+, D−, I, Θ)
2      (D+, D−) ← DRMX.create(D) ;
3      CP ← FindCandidates(∅, I, 𝔻+, D−, Θ) ;
4      MCP ← ExtractMinimalPatterns(CP) ;
5      return MCP ;
```

4.1 実験 1: 動的探索順序の優位性

提案手法上で静的変数順序と動的変数順序の速度差を調査する．静的変数順序は，負ラベルデータベース全体におけるアイテム出現頻度の昇順とした．表 1 に実験に用いたデータセットを示す[3]．Mushroom と Splice-1 は比較的疎で，German-credit は中程度，残りのデータセットは全て密である．

極小ジャンピング顕在パターンのマイニング時間の比較を図 2 に示す．LB はプルーニング 1 および 2 を適用し，NC はプルーニング 1 および 3 したものである．実験では，growth rate 制約を $\theta = 9$ としている．Audiology データセットでは，LB と動的な順序付けを組み合わせない限りマイニングを 3600 秒以内に終了できなかった．特に密なデータセットでは，動的順序付けと組み合わせると LB の有効性が劇的に向上することがわかる．

4.2 実験 2: 既存手法との比較

提案手法によるジャンピング顕在パターン（JEP）と極小ジャンピング顕在パターン（SJEP）の抽出，LCM [8] による JEP の抽出 [4]，CP-tree [2] による SJEP の抽出を行い，それぞれの計算速度を比較する．

実験結果を図 3 に示す．図より，ほとんどの場合において提案手法が最も高速にマイニング可能であることがわかる．CP-tree は高密度のデータセットで 3600

[3] これらはすべて CP4IM データセット https://dtai.cs.kuleuven.be/CP4IM/datasets/ によるものである．

[4] http://research.nii.ac.jp/~uno/codes.htm で公開される LCM バージョン 5.3 を使用した．

Algorithm 2: 解候補のパターンマイニング

```
1  FindCandidates(P, B, D+, D−, Θ)
2      CP ← ∅ ;
3      (occ+, occ−) ←
         (D+.countRow(), D−.countRow())
       // プルーニング 1: 極小性によるプルーニング
4      if isCP(occ+, occ−, Θ) = true then
5          CP ← CP ∪ {P} return CP ;
       // プルーニング 2: P の子孫の負例頻度の下界
          と正例頻度の上界を用いたプルーニング
6      lb_occ− ← D+.countRow(∅);
7      if (lb_occ− > σ−) then
8          return ;        // P の子孫をプルーニング
9      ub_occ+ ← D−.countRow(B)
         ub_gr ← ub_occ+/lb_occ− ; // Growth Rate
         以外も同様にプルーニング可能
10     if (ub_gr < θ) then
11         return ;        // P の子孫をプルーニング
       // プルーニング 3: 負例頻度の変化量によるプ
          ルーニング
12     for each i ∈ B do
13         if (D−.countRow() = D−.countRow(i))
             then
14             B ← B \ i ;        // 探索候補から除外
15             Dk.deleteColumn(i), ∀k ∈ {+, −} ;
16     if B = ∅ then
17         return CP
       // P ∪ {i} の D− 上の頻度による動的探索順
          序の決定
18     i* ← arg min_{i∈B}(D−.countRow(i))
19     Dk.checkpoint(), ∀k ∈ {+, −} ;
20     Dk ← Dk.deleteColumn(i*), ∀k ∈ {+, −} ;
21     CP ← CP ∪
         FindCandidates(P, B \ {i*}, D+, D−, Θ) ;
22     Dk ← {t ∈ Dk | P ∪ {i*} ⊆ t}, ∀k ∈ {+, −} ;
23     CP ← CP ∪ FindCandidates(P ∪ {i*}, B \
         {i*}, D+, D−, Θ) ;
24     Dk.undo(1), ∀k ∈ {+, −} ;
25     return CP
```

表 1: 実験 1 と 2 で用いるデータセット．density は全てのアイテム数に対するレコードの平均アイテム割合．#JEPs および#SJEPs はそれぞれ最小サポートしきい値を正ラベルデータ数の 0.02 倍に設定した場合のジャンピング顕在パターンと極小ジャンピング顕在パターン数

name	#item	#example	density	#SJEPs	#JEPs
Mushroom	119	8124	18%	1353	21574290
Splice-1	287	3190	21%	179810	377330
German-credit	112	1000	34%	148303	2410029163
Kr-vs-kp	73	3196	49%	7283	129786095160
Hypothyroid	88	3247	50%	1966	40807701172704
Anneal	93	812	45%	3906	34803198050304
Australian-credit	125	653	41%	2057646	261786633471699
Audiology	148	216	45%	2858	*unknown*

秒以内にマイニングを実行不可能であった．それに対し LCM は SJEP マイニングよりも高価なタスクである JEP マイニングを実行可能であり，従来の CP-tree アルゴリズムよりも高速であることがわかる．特に密なデータセットである Audiology データセットでは，提案アルゴリズムの SJEP バージョンのみが 3600 秒以内に実行可能だった．また比較的密なデータセットに対する JEP の抽出において，多くの場合提案アルゴリズムは LCM の 10 から 100 倍の高速化を達成している．

表 2: 実験 3 で用いるデータセット．

name	#sample	#feature (not binarized)	#target class
Banknote Authentication	1372	5	1
Breast Tissue	106	10	6
Glass Identification	214	10	6
Iris	150	4	3
Wireless Indoor Localization (Wifi)	2000	7	4
Yeast	1484	8	9

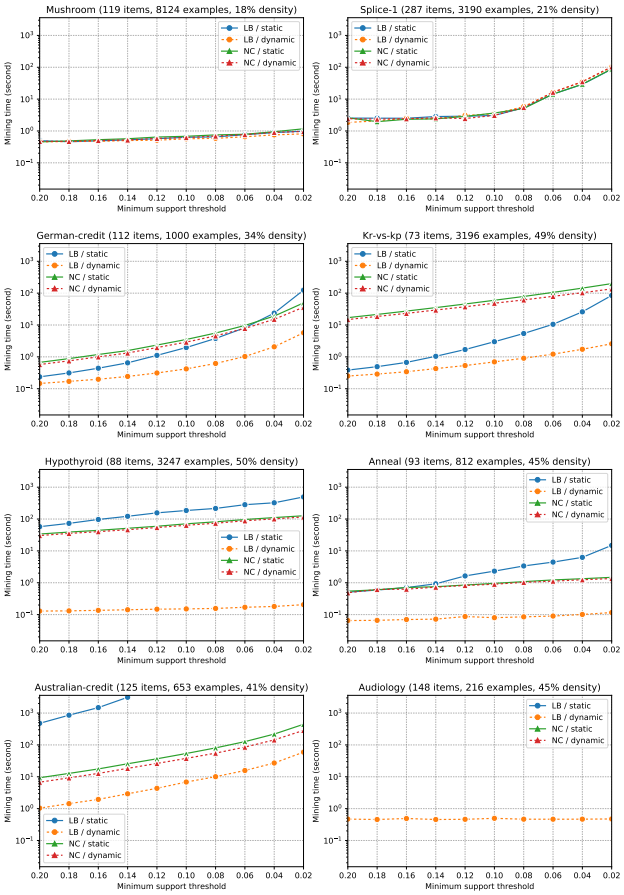

図 2: プルーニング 1 と 2（LB）および 1 と 3（NC）における静的順序と動的順序の速度比較

4.3 実験 3: 分類精度の比較

既存の分類モデルと否定アイテム含んだ極小顕在パターンを基にした分類モデルとの精度比較を行なう．具体的には，入力データベースに否定アイテムを加え，パターン長を 5 以下に制限したパターン抽出を行い，抽出したパターンを説明変数としてもつロジスティック回

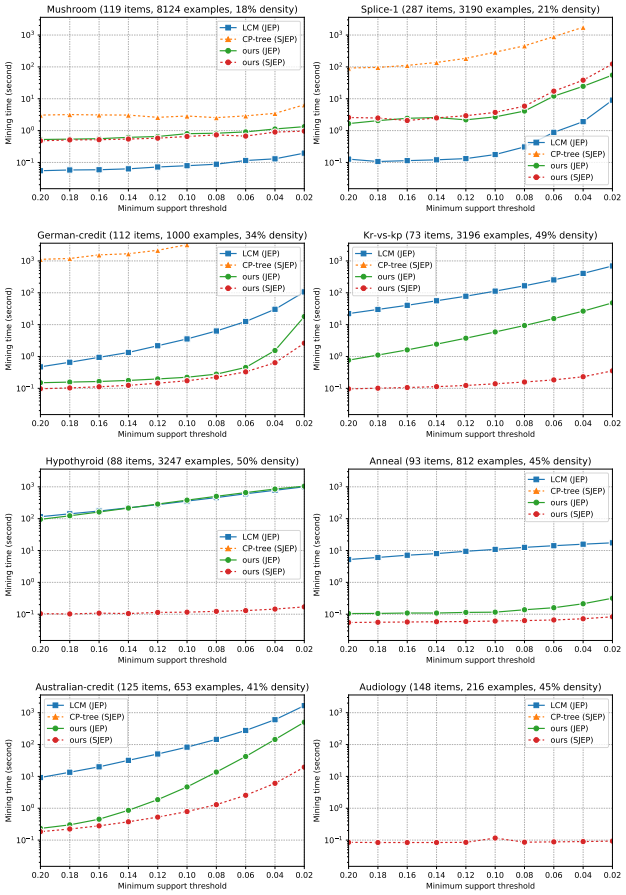

図 3: JEP と SJEP のマイニング時間の比較

帰を L1 正則化したモデルを考える．これを用いて表 2 に示すデータセットに対して二値分類問題を実行し，F 値を既存手法（ロジスティック回帰，決定木，ランダムフォレスト）と五分割交差検証で比較した．提案手法の学習には最小記述長原理を用いて離散化 [3] したデータを使用し，既存手法の学習には元の実数値データを使用した．また，全ての手法はライブラリ Optuna [5]を用いてハイパーパラメータを最適化した．実験結果を表 4.3 に示す．結果より，提案手法でマイニングしたパターンを用いたモデルはほとんどのデータセットで高い F 値を示した．さらに，否定アイテムを導入することでより高い精度を達成することがわかった．

表 3: 実験 3. 各手法との F 値の比較.

	提案手法（否定アイテムあり）	提案手法（否定アイテムなし）	決定木	ロジスティック回帰	ランダムフォレスト
banknote-class-1	**0.998**	0.992	0.989	0.991	0.994
breast-tissue-class-adi	**1.000**	**1.000**	0.935	0.931	0.971
breast-tissue-class-car	**0.937**	0.863	0.891	0.894	0.931
breast-tissue-class-con	**1.000**	**1.000**	0.891	0.740	0.900
breast-tissue-class-fad	**0.806**	0.722	0.599	0.673	0.535
breast-tissue-class-gla	**0.881**	**0.881**	0.771	0.700	0.727
breast-tissue-class-mas	**0.832**	0.770	0.523	0.482	0.474
glass-class-1	0.802	0.800	0.733	0.716	**0.830**
glass-class-2	**0.867**	0.863	0.777	0.599	0.799
glass-class-3	**0.697**	0.625	0.558	0.231	0.371
glass-class-5	0.920	**0.960**	0.777	0.658	0.865
glass-class-6	**1.000**	**1.000**	0.960	0.920	**1.000**
glass-class-7	0.942	**0.966**	0.900	0.915	0.915
iris-class-setosa	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
iris-class-versicolor	**0.962**	0.916	0.948	0.730	0.949
iris-class-virginica	0.949	0.943	0.952	**0.971**	0.952
wifi-class-1	0.993	0.993	0.989	0.989	**0.997**
wifi-class-2	**0.982**	0.961	0.979	0.977	0.978
wifi-class-3	0.974	0.943	0.953	0.598	**0.975**
wifi-class-4	**0.995**	0.956	0.991	0.994	**0.995**
yeast-class-CYT	0.632	0.604	0.604	0.606	**0.650**
yeast-class-ERL.	**1.000**	**1.000**	0.647	0.867	0.167
yeast-class-EXC	**0.661**	0.536	0.589	0.530	0.654
yeast-class-ME1	**0.785**	0.734	0.761	0.641	0.779
yeast-class-ME2	**0.591**	0.420	0.485	0.430	0.483
yeast-class-ME3	**0.823**	0.810	0.793	0.768	0.811
yeast-class-MIT	0.634	0.589	0.614	0.590	**0.645**
yeast-class-NUC	0.630	0.545	0.605	0.590	**0.634**
yeast-class-POX	**0.628**	0.614	0.614	0.614	0.560

参考文献

[1] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proc. KDD 1999*.

[2] H. Fan and K. Ramamohanarao. Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers. *IEEE TKDE*, 18(06):721–737, 2006.

[3] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI1993).*, pages 1022–1029, 1993.

[4] D. Knuth. Dancing links. *Millennial Perspectives in Computer Science*, Nov. 2000.

[5] E. Loekito and J. Bailey. Fast mining of high dimensional expressive contrast patterns using zero-suppressed binary decision diagrams. In *Proc. KDD2006*, pages 307–316. ACM, 2006.

[6] S. Morishita and J. Sese. Transversing itemset lattices with statistical metric pruning. In *Proc. PODS 2000*, pages 226–236. ACM, 2000.

[7] T. Toda. Hypergraph transversal computation with binary decision diagrams. In *Experimental Algorithms*, pages 91–102. Springer Berlin Heidelberg, 2013.

[8] T. Uno, T. Asai, Y. Uchida, and H. Arimura. LCM: an efficient algorithm for enumerating frequent closed item sets. In *Proc. FIMI '03, IEEE*, 2003.

[5] `https://optuna.org/`