テンソルデータモデル実現のための局所鋭敏型ハッシュを用いた順序処理

横林　亮平†　　三浦　孝夫†

† 法政大学大学院 理工学研究科システム理工学専攻　〒 184–8584 東京都小金井市梶野町 3-7-2
E-mail: †ryohei.yokobayashi.2j@stu.hosei.ac.jp, ††miurat@hosei.ac.jp

あらまし　テンソルデータモデルは，詳細なデータ表現が可能でありデータマイニングなどと相性が良い一方，データ量が膨大となり処理時間がかかるなどの問題がある．2 次記憶領域におけるデータの挿入・消去・探索など一連のデータ処理の効率化を図ることは，この問題の解決策の 1 つと言える．本稿では，局所性鋭敏型ハッシュを用いた順序探索法を提案する．ハッシュ技術は，データ量の影響を受けにくい探索方法であるが順序探索ができないという問題がある．局所性鋭敏型ハッシュを用いることで類似データを同ブロックにシーケンシャルに並べ，ブロックごとの並列処理を行うことで効率化を図る．また，テンソルデータ操作 (特にデータマイニング操作) の効率化についても述べる．局所性鋭敏型ハッシュを使用し、部分的なデータに絞った高速な多次元同時関係抽出を実現する．

キーワード　Tensor，Data Model，Data Minig，Hash

1. 前　書　き

近年，インターネットの利用拡大とセンサデバイスや半導体メモリの技術発展に伴い，取得可能なデータ量と多様性が増加している．これら膨大なデータから有用な特徴を効率的に抽出する技術はますます重要性を帯びている．これまでの研究 [11] で，テンソルデータモデル [5] [10] はデータの直観性に加え，詳細な表現が可能であることからデータマイニングなどと非常に相性のよいモデルであることが示されている．しかし，テンソルデータモデルは新たに加わる軸の次元数分データ量が増加し，疎なデータとなることから処理時間に関する問題が存在する．2 次記憶領域におけるデータの挿入・消去・探索など一連のデータ処理の効率化を図ることは，この問題の解決策の 1 つと言える．ハッシュ技術は，データ量に影響されない探索手法であるが順序処理の実行が困難となる．本研究では，局所鋭敏型ハッシュ [2] [6] [12] を用いることで上記の問題を解決する．

また，テンソルデータ操作（特にデータマイニング操作）[11] における高速化を図る．データマイニングの有名な手法の 1 つに同時関係抽出 [1] がある．現在，トランザクションデータベースから複数のトランザクションにおいて頻出するアイテム集合を見つける手法が一般的となっている．しかし，より有益な情報はトランザクションに跨って頻出する同時関係 [8] であると言える．例えば一般的な同時関係は，

A 社の株価が上がれば，同時に B 社の株価が上がる

といった関係ルール抽出となる．しかし，テンソルモデルでは

A 社の株価が上がれば，2 日後に B 社の株価が上がる

という時間などの距離概念を追加し，トランザクションを越えたルール (多次元同時関係) 抽出 [8] が容易に可能となる．

このとき，全データを対象に共起性を考えることは計算コストを大幅に上げることとなる．そこで，類似する次元を近い位置にまとめた構造をとることで局所データに絞った高速な関係抽出を行う．

本研究の貢献点は以下の点である．

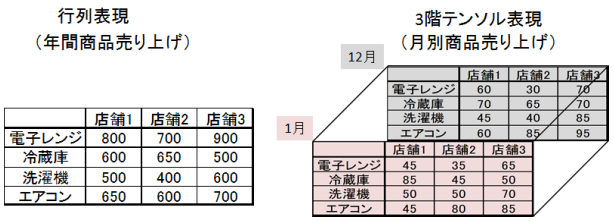
- 局所鋭敏型ハッシュによる高速な順序処理の実現
- テンソルデータ操作の効率化

2 章以降では，以下のように構成される．2 章ではテンソルデータモデルのデータ記述とその操作について述べる．3 章では同時関係抽出とデータマイニング操作について述べ，4 章でデータマイニング操作の効率化について述べる．5 章ではテンソルのデータ挿入・探索方法について言及する．6 章では，実験により提案手法の有用性を示し，7 章で結論とする．

2. テンソルデータモデル

2.1 高階データ表現

高階（テンソル）データ [10] とは，階数が 3 以上となるデータ構造を指す．テンソルは $X \in \mathcal{T}[I_1 \times I_2 \times ... \times I_N]$ で表し，N をこのテンソルの階数という．また，ベクトルと行列はそれぞれ 1 階のテンソル，2 階のテンソルと呼びテンソルで一般化することで同様に扱える．高階テンソルを用いることで，行列データよりも詳細な情報表現を行うことができ，またデータ操作が単純になる．例えば，ある店の商品売上情報をみる場合，行列では 1 年ごとといったまとまったデータ情報を表現することになる．しかし，テンソル表現をする場合，年間売上を月別に表示することができる．図 1 にテンソルのデータ例を示す．



	店舗1	店舗2	店舗3
電子レンジ	800	700	900
冷蔵庫	600	650	500
洗濯機	500	400	600
エアコン	650	600	700

	店舗1	店舗2	店舗3
電子レンジ	60	30	70
冷蔵庫	70	65	70
洗濯機	45	40	85
エアコン	60	85	95

	店舗1	店舗2	店舗3
電子レンジ	45	35	65
冷蔵庫	85	45	50
洗濯機	50	50	70
エアコン	45	80	85

図 1　店舗 1,2 の商品売り上げ

2.2 テンソルデータ操作

テンソルデータ操作 [5] [10] には，テンソル射影と選択，テンソルどうしの和・差・積演算，階変換，次元縮小と復元，検索

操作，ベクトル化，シフト操作，カウント操作などがある．以下，個々の操作を定義する．

射影は π と記述し，指定した条件に従い部分的なテンソルを取り出す．例えば，図 1 の月別売上データにおいて，電子レンジと洗濯機が 12 月以外に売れた店舗情報を取り出す．この操作は，$X' = \pi_{(\sharp 1(\text{電子レンジ, 洗濯機})\sharp 2(*)\sharp 3(12\ \text{月})^{-1})}(X)$ で表す．

選択は σ で記述し，限定的にデータの一部分を抽出する．例えば，図 1 のテンソルデータより洗濯機が 70 台以上売れた店を抽出する．データ操作は，$X'' = \sigma_{\sharp 2.\text{洗濯機}>70}(X)$ となる．射影，選択によって得られるデータ例を図 2 に示す．

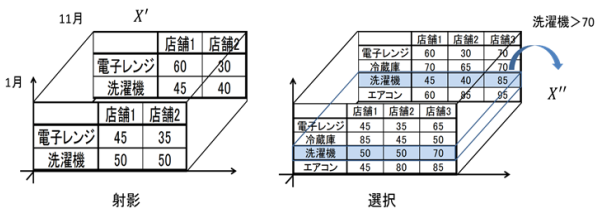


図 2　射影と選択操作

次に，テンソルの演算操作について述べる．階数と次元数の等しい 2 つのテンソル $X, Y \in \mathcal{T}[I_1 \times I_2 \times \ldots \times I_N]$ についてそれぞれの要素を $x_{i_1,i_2,\ldots,i_N}$，$y_{i_1,i_2,\ldots,i_N}$ と表す．2 テンソルの和と差は，$X+Y$，$X-Y$ と表し，その要素は

$$X + Y = x_{i_1,i_2,\ldots,i_N} + y_{i_1,i_2,\ldots,i_N}$$
$$X - Y = x_{i_1,i_2,\ldots,i_N} - y_{i_1,i_2,\ldots,i_N}$$

となる．2 つのテンソルの積は

$$\langle X, Y \rangle = \sum_{i=1}^{I_1} \sum_{i=2}^{I_2} \ldots \sum_{i=N}^{I_N} x_{i_1,i_2,\ldots,i_N} y_{i_1,i_2,\ldots,i_N}$$

となる．

テンソルは，行列やベクトルの集合であり，ベクトル表現ができる．ベクトル表現変換を形式化したファイバとモードについて言及する [5]．ファイバは，テンソルをベクトル表現したものである．3 階のテンソルを例に挙げる．このとき，ベクトル表現方法は 3 種類存在する．3 種の表現を図 3 に示す．固定したテンソルに対して，列方向 (column)，行方向 (row)，奥行き方向 (tube) にファイバをとり，それぞれモード 1 ファイバ，モード 2 ファイバ，モード 3 ファイバと呼ぶ．テンソルを行列

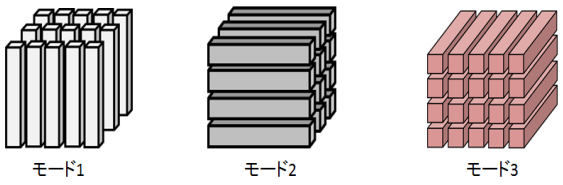


図 3　ファイバとモード

で表現するときは，n-モード行列化と呼びモード n のファイバ

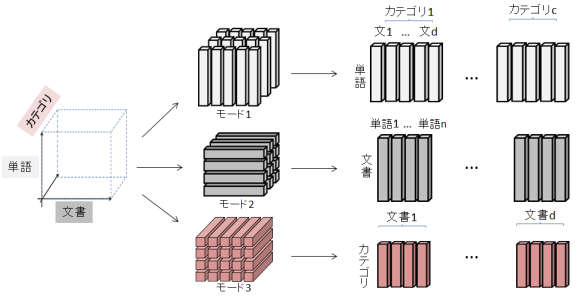


図 4　n-モード行列化

を用いて行列化する．n-モード行列化を図 4 に示す．モード n のファイバから構成された n-モード行列は，$X_{(n)}$ で表現する．

テンソルに対する情報検索は，モード n ファイバまたは n-モード行列化を用いて，検索質問との類似度を求めることにより行う．検索質問は，索引語の集合からなり，類似度は一般的に内積や余弦類似度を用いる．したがって，テンソルとベクトルの内積 (ベクトルテンソル積) は類似度操作の意味を持つ．テンソル $X \in R^{I_1 \times I_2 \times \ldots \times I_N}$ とベクトル $V \in R^{I_n}$ とすると，ベクトルテンソル積は $Y_{(n)} = X \times_n V \in \mathcal{T}[I_1 \times I_2 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N]$ であり，

$$X \times_n V = ((Y_{i_1,..,i_{n-1},i_{n+1},..,i_N})) \tag{1}$$

$$Y_{i_1,..,i_{n-1},i_{n+1},..,i_N} = \sum_{i_n=1}^{I_n} X[i_1,..,i_{n-1},i_n,i_{n+1},..,i_N] V[i_n] \tag{2}$$

となる．このとき，Y は X に対して階数が 1 つ低くなる．

テンソル (図 6) に対してベクトルテンソル積を行う．ベクトル $(1,\ 2,\ 3)^t$ のとき，ベクトルテンソル積による類似度操作を以下に示す．

$$\begin{bmatrix} 45 & 35 & 65 & 60 & 30 & 70 \\ 85 & 45 & 50 & 70 & 65 & 70 \\ 50 & 50 & 70 & 45 & 40 & 85 \\ 45 & 80 & 85 & 60 & 85 & 95 \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 310 & 210 \\ 325 & 410 \\ 360 & 380 \\ 1160 & 535 \end{bmatrix}$$

要素が内積となるため，要素の高い部分の属性データをランキングし表示することがテンソル情報検索となる．テンソルにおける情報検索操作の例を図 5 に示す．

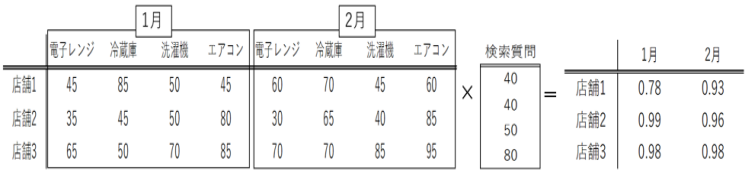


図 5　テンソル情報検索

次元縮小はデータを低次元に射影し，データの空間量と複雑さを大幅に削減することができる．テンソルデータモデルにおいて，行列とテンソルの内積 (行列テンソル積) 操作は

次元縮小を意味する．テンソル $X \in \mathcal{T}[I_1 \times I_2 \times ... \times I_N]$ ，行列 $M \in R^{J_n \times I_n}$ とし，行列テンソル積を Y とする．$Y_{(n)} = X \times_n M \in \mathcal{T}[I_1 \times I_2 \times ... \times I_{n-1} \times J_n \times I_{n+1} \times ... \times I_N]$, $j_n = 1, .., J_n$ とおくと，

$$X \times_n M = ((Y_{i_1,..,i_{n-1},j_n,i_{n+1},..,i_N})) \tag{3}$$

$$Y_{i_1,..,i_{n-1},j_n,i_{n+1},..,i_N} = \sum_{i_n=1}^{I_n} X[i_1,..,i_{n-1},i_n,i_{n+1},..,i_N]M[j_n,i_n] \tag{4}$$

となる．$J_n \leqq I_n$ であれば次元数は小さくなる．$X \in \mathcal{T}[2\times3\times2]$ のテンソル (図 6) を次元縮小する場合について述べる．行列

	1月			2月		
	店舗1	店舗2	店舗3	店舗1	店舗2	店舗3
電子レンジ	45	35	65	60	30	70
冷蔵庫	85	45	50	70	65	70
洗濯機	50	50	70	45	40	85
エアコン	45	80	85	60	85	95

図 6　1,2 月店舗売り上げ

$M \in R^{3\times4}$ を用いると，

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 45 & 35 & 65 & 60 & 30 & 70 \\ 85 & 45 & 50 & 70 & 65 & 70 \\ 50 & 50 & 70 & 45 & 40 & 85 \\ 45 & 80 & 85 & 60 & 85 & 95 \end{bmatrix}$$

$$= \begin{bmatrix} 320 & 385 & 445 & 340 & 400 & 525 \\ 335 & 245 & 365 & 365 & 260 & 435 \end{bmatrix}$$

となり，次元数が小さくなる．

$Y_{(n)} = X \times_n M$ は以下の特性を持つ．

$$X \times_m A \times_n B = X \times_n B \times_m A (m \neq n)$$

$$X \times_n A \times_n B = X \times_n (BA)$$

ベクトル化，シフト操作，カウント操作について言及する．テンソル $X \in \mathcal{T}[I_1 \times I_2 \times ... \times I_N]$ をベクトル化する場合，$I_k = \{1, ..., |I_k|\}$ とし，$Y = (\vec{X})$ $y_{i_1 \ldots i_N} = 1$ if $x_{i_1 \ldots i_N} \neq 0$, otherwise $y_{i_1 \ldots i_N} = 0$ と定義する．テンソル $X \in \mathcal{T}[I_1 \times I_2 \times ... \times I_N]$，各要素 $x_{i_1} = x_{i_{1,1},i_{1,2},...,i_{1,N}}$ においてベクトル化を行うと,$\vec{x}_{i_{N,1}} = x_{i_{1,1},i_{2,1},...,i_{N,1}}$ となる．

テンソルシフト操作は，条件に従い要素をシフトする操作である．$x_{i_1,i_2,..,i_N}$ に対して右シフトを行うと $x_{0,i_1,i_2,..,i_{N-1}}$ が得られる．テンソルをベクトル化や行列化を用いることで各ベクトル・行列に対してもシフト演算が可能となる．右シフト操作の様子を図 7 に示す．次に要素が全て 1 のテンソル Y との内積について言及する．例として 2 階のテンソル X と 1 階のテンソル Y の内積を考える．$X^{n\times m} \times Y^{m\times1} = (Z^{n\times1})$ であるため，

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^t = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

を得る．全要素 1 のベクトルとの内積計算を行う場合，各次元のカウント操作を行うことと等価となる．

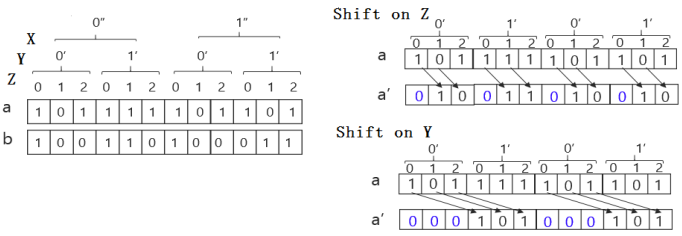


図 7　テンソルシフト操作

3. 同時関係抽出

同時関係 [3] を積極的に利用するアプリケーションに，スーパーマーケットの売り上げデータを分析するマーケットバスケット分析などがある．これらのデータベースはトランザクションデータベースと呼ばれ，トランザクションレコードが含まれる．同時関係は，同じトランザクション中で同時に起こる (出現する) 集合の内，その集合が別の複数トランザクションでも含まれる高出現頻度集合をルールとして抽出する．Agrawal [1] らにより提案された Apriori では，このとき候補となる集合の絞り込みを行うことで効率的にルールを取得する．

3.1 Apriori

アイテム集合を $I = \{i_1, i_2, \cdots, i_s\}$，トランザクションデータベース D をトランザクション $\{t_1, t_2, \cdots t_n\}$ の集合とする．トランザクション t_i $(i = 1, 2, \cdots, n)$ は I のサブセットからなる．トランザクションとアイテム集合の様子を表 1 の左に示す．トランザクション内に同時に出現するアイテム項目集合の内，高出現頻度のものをルールとして取得する．例えば a,b が共起するとき，

$a \Rightarrow b$

といった同時関係ルールを得る．出現頻度の多さは支持度 (support) と確信度 (confident) をユーザ定義の閾値と比較することにより計る．

$$support = \frac{\text{条件 } X, Y \text{ を同時に満たすトランザクション}}{\text{全トランザクション}}$$

$$confident = \frac{\text{条件 } X, Y \text{ を同時に満たすトランザクション}}{\text{条件 } X \text{ を満たすトランザクション}}$$

ユーザ定義の閾値は最小支持度 (min-support)，最小確信度 (min-confidence) としこれらを上回るものを頻出項目集合とし抽出する．Apriori は全てのアイテムの組み合わせを考えるのではなく，長さ $(k-1)$ の頻出項目集合から長さ k の候補集合 C_k を作成することにより候補を絞り込み効率的に同時関係抽出を行うことができる．

3.2 多次元同時関係

多次元同時関係 [7] [8] では，距離概念を加え次元 (トランザクション) を越えたルールの抽出を行う．$n_i = \langle v \rangle$ と $n_j = \langle u \rangle$ を階 1 の空間上の 2 点 (アドレス) としたとき，n_i と n_j の相対距離は $(n_i, n_j) = \langle u - v \rangle$ となる．基準点 0 における相対距離 (相対アドレス) は $(0,n_j)=(n_j)=\langle v \rangle$ と記述される．階 1 空間上の点 Δ_j におけるアイテム i_k は拡張アイテムと呼び $\Delta_j(i_k)$ と表す．同様に点 Δ_j におけるトランザクション T_K は拡張ト

ランザクションセットと呼び $\Delta_j(T_K)$ と表す．I_e は階 1 空間内に出現し得る全ての拡張アイテムセット，D_e を出現し得る全拡張トランザクションセットとする．

Lu [7] [8] らは，従来のデータベースから拡張トランザクションデータベースに変換することで距離概念を追加している．表 1 に拡張トランザクションの変換例を示す．

表 1　拡張トランザクションと拡張アイテム

トランザクション	アイテム	拡張トランザクション	拡張アイテム
T1	a,b,c,d	Δ 0(T1)	Δ 0(a) , Δ 0(b) , Δ 0(c), Δ 0(d)
T2	a,b,d	Δ 1(T2)	Δ 1(a), Δ 1(b), Δ 1(d)
T3	a,c,d	Δ 2(T3)	Δ 2(a), Δ 2(c), Δ 2(d)
T4	b,d	Δ 3(T4)	Δ 3(b), Δ 3(d)
T5	b,c,d	Δ 4(T5)	Δ 4(b), Δ 4(c), Δ 4(d)

この拡張トランザクションに含まれる拡張アイテムはどのトランザクションを基準とするかにより距離が変化する．そのため，拡張トランザクションには拡張アイテムリストと起こり得る全ての相対アドレスを保持しておく．多次元同時関係の摘出アルゴリズム E-Apriori [8] を Algoritm1 に示す．全ての距離

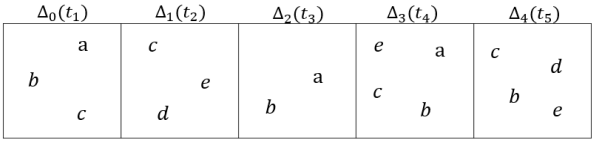


図 8　多次元同時関係

のアイテム集合をカウントし，最小支持度，最小確信度と比較することで 1 項目頻出集合を得る．次に 2 項目集合では，頻出項目集合から基準 $0''0'0$ のアイテムとのペアを全て候補とする．データベースを読み込み候補集合と同じアイテム・相対アドレスを持つものをカウントし，最小支持度，最小確信度を越えるものを頻出項目とする．3 項目集合以降も同様に候補を作成し，カウントする．

3.3　テンソルによる多次元同時関係抽出操作

テンソルデータモデルの枠組みで多次元同時関係を抽出する方法 [11] について述べる．多次元同時関係抽出操作は，ベクトル化，シフト操作，カウント操作を用いることで実現する．テンソルデータ操作による多次元同時関係を抽出の様子を Algoritm2 に示す．

ここでは，3 階のテンソルについて言及する．まず，ベクトル化により各要素を 1,0 のベクトルに変換する．

1 項目集合 $\Delta_{0''0'0}\{i_n\}$ では，各ベクトルに対して全要素 1 のベクトルとの内積カウント操作を行い，結果が最小支持度･最小確信度以上となるものを抽出する．$\Delta_{s''s's}\{i_n\}\,(s''s's \neq 0''0'0)$ では，ベクトルを $x''=s'', x'=s', x=s$ だけ右シフトし，得られたベクトルに対して内積によるカウント操作を行う．全ての距離について内積をとり，1 項目頻出項目集合とする．

2 項目集合以上の場合を考える．簡単のためここでは 2 項目集合について述べる．1(k-1) 項目頻出集合から候補集合 $C_2(C_k)$ を作成する．C_2 の項目集合と一致するアイテムを要素別ベクトル化から取り出す．$\Delta_{0''0'0}\{a\} \Rightarrow \Delta_{0''0'1}\{b\}$ では，a ベクトルと b ベクトルを抽出し，a ベクトルを $0''0'0$ から $0''0'1$ となるように右シフト操作を行う．これにより違う距離のアイテムを同じ次元としてみることができ，AND をとることで同時関係をとることが可能となる．図 9 にこの様子を示す．最後に全要素 1 のベクトルと内積を行う内積カウント操作を実行し，最小支持度最小確信度と比較することで頻出項目集合抽出とする．

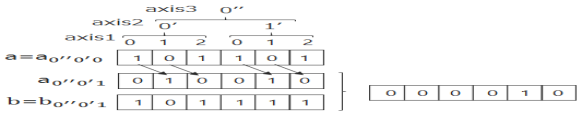


図 9　$2-itemsets$

4.　局所鋭敏型ハッシュを用いた同時関係抽出

既存のテンソル操作による方法では，全データに対してマイニングを行う必要がある為，関係ルールの候補が膨大で計算コストがかかる．そこでテンソルを構築する際，データの次元の共起性を考えることで影響の受ける範囲を限定しデータ操作を高速に行う．ここでは共起性を考える為，情報検索手法の 1 つである局所鋭敏型ハッシュを用いる方法について述べる．

4.1　局所鋭敏型ハッシュ

局所鋭敏型ハッシュ [2] [6] [12] は，データの類似性を考慮した情報検索などで用いられる技術である．1 つのデータに対し複数回のハッシュを行うことで，類似データをなるべく近いまたは同じバケットにハッシュする．例えば，図 10 のようなアイテム集合 S1，S2，S3，S4 があるとする．このとき，S の個

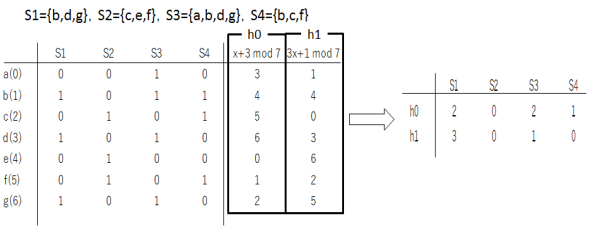


図 10　局所鋭敏型ハッシュ

数が増えると類似度を示す Jaccard 係数などの計算が高価となる．局所鋭敏型ハッシュは，各アイテムに複数のハッシュ関数 (h_0と h_1) を適用し最小のハッシュ値を求める．ここでは簡単のためアイテムをそれぞれ番号で置換している．例えば，S1 ではビットが立っているのがアイテム b,d,g の部分でハッシュ関数 h0 での値が 4,6,2 となっている．よって最小値の 2 を得る．これを複数のハッシュ関数に対して行うことで，7 次元あったものを 2 回の比較のみでジャッカード係数を推定できる．

4.2　多次元同時関係の高速化

局所鋭敏型ハッシュは類似データが類似するアドレスを持つ．よって，それぞれの軸の各次元に対し局所鋭敏型ハッシュを用いると次元の類似性を簡単に考えることが可能である．類似する次元を近い位置におきテンソルを構成することで，共起する次元は近い次元にのみに限定される．従って，データマイニング操作ではデータの全部分を対象とするのではなく局所的な

データに絞ることで効率化を図ることができる．図 11 にテンソルの次元構成の仕方を示す．

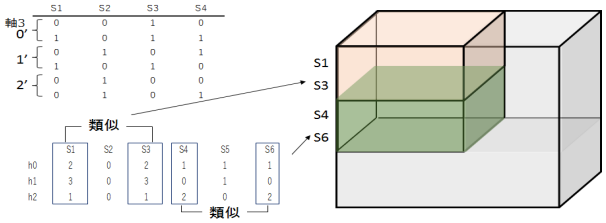


図 11　局所鋭敏型ハッシュによる次元構成

5.　テンソルの探索

一般に扱われるデータは膨大であり，主記憶・２次記憶などと分けてデータを保存する．このとき，低速な入出力装置の使用はコンピュータ全体の大幅な性能低下につながる．特にテンソルでは，各軸のデータ対してデータ構造を適用し情報を保持する．これはテンソル構築が仮想的に行われるためである．よって，データ量に独立した効率的な探索方法が必要となる．ここではハッシュ，Btree，局所鋭敏型ハッシュを用いる提案手法について述べる．また，テンソル操作と探索の関係について言及する．

5.1　ハッシュ

ハッシュ技術はデータ量に影響を受けず挿入・消去・探索を O(1) で行える手法である．与えられるデータを d，ハッシュ関数 h としキー値 C から得られる h(C) により対応するバケットを一意に決定することができる．したがって，キーの呼び出しのみで高速に探索が可能となる．しかし，順序処理ができないため連続データに対して探索を行うとヘッドの移動が起こりアクセスに時間がかかる．

5.2　Btree

Btree ２次記装置上で効率よく操作ができるよう設計された平行探索木で探索効率は，木の高さ・ブロックのばらつきなどに影響を受ける．木の高さがブロックの読み取り回数となっており，与えられるデータから生成される木は分岐数によって複数存在する．B-tree は乱順探索は高価となるが，順序処理が効率的に行える．

5.3　局所鋭敏型ハッシュによる順序探索

局所鋭敏型ハッシュを用いた順序処理について述べる．k 個のハッシュ関数から得られる $h_0(C0)h_1(C1)\cdots h_k(Ck)$ をまとめて１つのアドレスと考えると類似データを同じバケットに入れることが可能である．そのバケット内でデータを主キーによりシーケンシャルに並べることで順序処理を可能とする．図 12 を例に述べる．

データは主キーの学籍番号と副次キーの出身地が存在し，副次キーを用いてハッシュを行う．これにより副次キーに関して似たデータ同士が同じバケットに挿入される．ハッシュ関数 h_0 , h_1 とし，得られた $h_0(C)$ と $h_1(C)$ を合わせてアドレスとする．アドレス 2-3 には長崎，宮崎が含まれるためアドレス 2-3 内で主キーによるソート順にデータを保存する．よって，連続

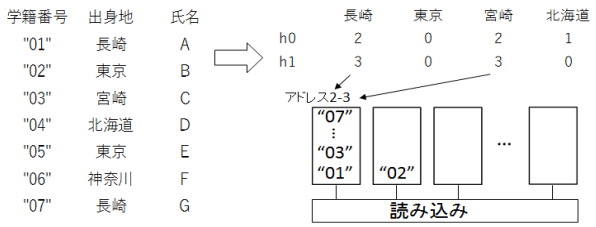


図 12　局所鋭敏型ハッシュによる処理

してアクセスされる類似データが同じ空間または隣接した空間に存在するという局所性を持つ．複数のバケットを並べ，データが読み込まれたら並列にデータをたどることで高速な順序処理を可能とする．

5.4　テンソル操作と探索

テンソルでは，各軸のデータに対して Btree や提案手法を適用する．例えば，選択操作を行う際の探索との関係を図 13 に示す．データは X，Y，Z，V から構成される．出身地が広島で最終学歴が博士，年齢 32 の年収を取り出す場合を考える．各軸における構造から選択を行うため，$\sigma_{Z=32}\sigma_{Y=博士}\sigma_{X=広島}(D) = 850$ となる．

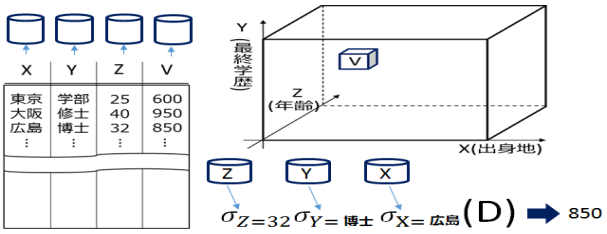


図 13　選択操作と探索

6.　実　　験

実験では，探索の速度比較とデータマイニング操作に関する実験を行う．Btree と局所鋭敏型ハッシュによるデータ構造を用い乱順序と順序探索の両方を行う．また，既存の全データに対するデータマイニング操作と局所的に絞った操作とを比較し，読み込みデータ量と得られる同時関係数を考察する．

6.1　実 験 準 備

１つ目の実験では日本の郵便 (全国一括) データを用いる．ここには，市番号・郵便番号・県名・都市名・町名などが含まれる．郵便番号 122897 件からランダムに抽出した 10000 件を用いて Btree と提案手法を構築する．次に 10000 件からランダムに抽出した 1000 件を用いて探索を行う．乱順序探索では 1000 件をそのまま用い，順序探索ではソートをかけた後探索を行う．

２つ目の実験に用いる気象データひまわり 2000 では，アメダス (自動気象データ収集システム) により収集された日単位データが収録されている．本実験のテンソルデータモデルはアイテム，地域，時間の３つを距離概念として用い，３階のテンソルを構築する．地域は 45 個のグループ (次元) に分けられておりデータの欠損のあるさいたまと大津を除く県庁所在地となっている．時間は季節的な天候の変化を考慮して１月１日から 31

日までのデータを使用する．アイテム属性はそれぞれの観測地で観測された日降水量，日照時間，平均気温，最高気温，最低気温，平均湿度，最小湿度，最大瞬間風速の 8 属性をさらに定量化した 32 個を用いる．日降水量は降水がなければ 0 あれば 1 で表す．日照時間は 0〜2 時間未満，2 時間以上 4 時間未満，4 時間以上 6 時間未満，6 時間以上 8 時間未満，8 時間以上の当てはまる部分に 1 それ以外 0 としている．表 2 にその他の要素を定量化したものを示す．以上により得られた要素，地域，時間に基づきテンソルデータの構築を行う．

表 2　要素定量化

平均気温 (deg.C)	∼0, 0∼4, 4∼8, 8∼
最高気温 (deg.C)	∼5, 5∼10, 10∼15, 15∼
最低気温 (deg.C)	∼2, 2∼5, 5∼8, 8∼
最大瞬間風速 (m/s)	∼10, 10∼15, 15∼20, 20∼
平均湿度 (%)	∼50, 50∼60, 60∼70, 70∼80, 80∼
最小湿度 (%)	∼30, 30∼40, 40∼50, 50∼60, 60∼

6.2　評価と考察

1 つ目の実験では，10000 件のデータから 2 層の BTree と 367 個のバケットを持つ提案手法が構築される．提案手法では，局所鋭敏型ハッシュにより得られるアドレス (バケット) が 367 個存在する．Btree の分岐数と最大要素数を変化させたときの様子を表 3 と表 4 に示す．

表 3　分岐と順序処理のタッチ回数

分岐数	113	96	85	69	68	64	62	60	54	49
最大要素数	130	150	171	186	209	228	248	269	289	311
タッチ回数	8970	9172	9099	9381	9464	9419	9441	9424	9492	9637

表 4　分岐と乱順序処理のタッチ回数

分岐数	113	96	85	69	68	64	62	60	54	49
最大要素数	130	150	171	186	209	228	248	269	289	311
タッチ回数	107182	103978	105629	109473	110051	110619	115906	118530	126555	135144

順序処理では，提案手法のタッチ回数は 8912 回であり，2 層で平行を保つ最大の分岐数の Btree と同程度となる．乱順序では，45475 回と B-tree の半分以下の速度である．順序処理では，両手法共に一度読んだ部分以降のみを探索するため乱順序より早く探索が行える．乱順序探索の提案手法では，アドレスによるデータの絞り込みと B-tree のノード数よりも 1 バケットあたりの要素数が少ないことからより高速にデータを探索できていると考えられる．

2 つ目の実験では，支持度 40%，確信度 80% で同時関係抽出を行い，既存手法・提案手法ともに 265 件の同時関係数が得られる．抽出過程での候補数は全件マイニングで 15105 件，提案手法では 9999 件となる．よって，既存の 66% の候補数で全く同じマイニングを行える．以下に抽出された同時関係を示す．

- 札幌の最高気温が 10 ∼ 15 度
  ⇒ 次の日の盛岡の最小湿度は 50∼ 60%
- 札幌の最高気温が 10 ∼ 15 度，
  札幌の平均気温が 0 度未満
  ⇒ 次の日の秋田の最小湿度は 50∼60%

取得された同時関係が北海道・東北に関するものに偏っていたことから 1 月の北に位置する地域の気温と湿度は安定していることが言える．また，時間的なルールは同日，1 日後，2 日後の 3 種の組み合わせからできており天候が影響する範囲 (予測できる範囲) は 2 日間までと考えられる．

7.　結　　論

テンソルデータモデル実現のため，局所鋭敏型ハッシュを用いた 2 次記憶領域におけるデータの I/O 操作の提案を行った．これによりハッシュでは行えない順序処理を可能とし，Btree と同程度の高速な順序探索が可能であることを示した．また，ハッシュの特性である高速な乱順序処理が行える性質も併せ持つことを示した．さらに実データを用いてテンソルの多次元同時関係抽出を行い，複数の意味を有するトランザクションを跨いだ同時関係 265 件を取得した．局所鋭敏型ハッシュを用いることで従来の 66%のデータ量で多次元同時関係抽出が可能であることを示した．

文　　献

[1] Agrawal, Rakesh, Tomasz Imielinski, and Arun Swami. "Database mining: A performance perspective." IEEE transactions on knowledge and data engineering 5.6 (1993): 914-925.

[2] Broder, Andrei Z., et al. "Min-wise independent permutations." Journal of Computer and System Sciences 60.3 (2000): 630-659.

[3] Han, Jiawei, Jian Pei, and Micheline Kamber. "Data mining: concepts and techniques. " Elsevier, 2011.

[4] De Lathauwer, Lieven, Bart De Moor, and Joos Vandewalle. "A multilinear singular value decomposition." SIAM journal on Matrix Analysis and Applications 21.4 (2000): 1253-1278.

[5] Kolda, Tamara G., and Brett W. Bader. "Tensor decompositions and applications." SIAM review 51.3 (2009): 455-500.

[6] Li, Ping, and Christian Knig. "b-Bit minwise hashing." Proceedings of the 19th international conference on World wide web. ACM, 2010.

[7] Lu, Hongjun, Jiawei Han, and Ling Feng. "Stock movement prediction and n-dimensional inter-transaction association rules." 1998 ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Seattle, WA, USA, ACM, New York, USA. 1998.

[8] Lu, Hongjun, Ling Feng, and Jiawei Han. "Beyond intra-transaction association analysis: mining multidimensional intertransaction association rules." ACM Transactions on Information Systems (TOIS) 18.4 (2000): 423-454.

[9] Sanguansat, P., "Higher-Order Random Projection for Tensor Object Recognition.", Intn'l Symp. on Communications and Information Technologies (ISCIT). IEEE, 2010.

[10] Yokobayashi, Ryohei, and Takao Miura. "Modeling random projection for tensor objects." Proceedings of the International Conference on Web Intelligence. ACM, 2017

[11] Yokobayashi, Ryohei, and Takao Miura. "Multidimensional Data Mining Based on Tensor." The 2nd International Workshop on Social Computing (IWSC). IEEE, 2018.

[12] 岩野和生, 浦本直彦. "大規模データのマイニング." 共立出版.

$\bullet k = 1$

$C_1 = \{\{\Delta_x\,(i_n)\}\} \mid (i_n \in D) \wedge (0 \leqq x \leqq maxspan)$

foreach *extended transaction* $\Delta_s\,(t)$ **do**

  foreach *candidate* $c:\ \Delta_x\,(i_n) \in C_1$ **do**

    c.count++;

  end

end

$L_1 = \{c : \{\Delta_x\,(i_n)\}\} \mid (c \in C_1) \wedge (c.count \geqq support)$

$\bullet k = 2$

$C_2 = \{\{\Delta_0\,(i_m)\,, \Delta_x\,(i_n)\} \mid (\Delta_0\,(i_m) \in L_1) \wedge (\Delta_x\,(i_n) \in L_1) \wedge ((x \neq 0) \vee (x = 0 \wedge i_m < i_n))\}$

foreach *extended transaction* $\Delta_s\,(t)$ **do**

  foreach *candidate* $c:\ \Delta_0\,(i_m)\,, \Delta_x\,(i_n) \in C_2$ **do**

    c.count++;

  end

end

$L_2 = \{c : \{\Delta_0\,(i_m)\,, \Delta_x\,(i_n)\}\} \mid (c \in C_2) \wedge (c.count \geqq support)$

$\bullet k > 2$

for $(k = 3; L_{k-1} \neq \emptyset; k++)$ **do**

  $C_k = E - Apriori - Gen\,(L_{k-1})$

  for $(o = 1; o \neq k; o++)$ **do**

    $G_\emptyset = E - Group\,(C_k, o)$;

    foreach *extended transaction* $\Delta_s\,(t)$ **do**

      $G_{\Delta_s(t)} = E - Group\,(C_o, \Delta_s)$;

      foreach *candidate* $c:\ \{\Delta_{x_1}\,(i_1)\,, \ldots, \Delta_{x_k}\,(i_k)\} \in G_{\Delta_s(t)}$ **do**

        c.count++;

      end

    end

  end

  $L_k = \{c : \{\Delta_{x_1}\,(i_1)\,, \ldots, \Delta_{x_k}\,(i_k)\}\} \mid (c \in C_k) \wedge (c.count \geqq support)$

end

$L = \bigcup_k L_k$

Algorithm 1: E-Apriori

$\bullet k = 1$

$C_1 = \{\{\Delta_x\,(i_n)\}\} \mid (i_n \in D) \wedge (0 \leqq x \leqq maxspan)$

elemental vectorization

foreach *element* $\Delta_{s''s's}\,(N)$ **do**

  foreach *candidate* $c:\ \Delta_{x''x'x}\,(i_n) \in C_1$ **do**

    c.dotproduct++;

  end

end

$L_1 = \{c : \{\Delta_{x''x'x}\,(i_n)\}\} \mid (c \in C_1) \wedge (c.dotproduct \geqq support)$

$\bullet k \geqq 2$

$C_2 = \{\{\Delta_{0''0'0}\,(i_m)\,, \Delta_{x''x'x}\,(i_n)\} \mid (\Delta_{0''0'0}\,(i_m) \in L_1) \wedge (\Delta_{x''x'x}\,(i_n) \in L_1) \wedge ((x''x'x \neq 0''0'0)) \vee (x''x'x = 0''0'0 \wedge (i_m < i_n))\}$

foreach $(k = 2; L_{k-1} \neq \varnothing; k++)$ **do**

  foreach *element* $\Delta_{s''s's}\,(N) \mid (0 < s'' < x''_{maxspan}, 0 < s' < x'_{maxspan}, 0 < s < x_{maxspan})$ **do**

    for *candidate* $c:\ \Delta_0\,(i_m)\,, \Delta_x\,(i_n) \in C_2$ **do**

      right shift$\{\{\Delta_{0''0'0}\,(i_m)\,, \Delta_{x''x'x}\,(i_n)\} \Rightarrow \{\Delta_{max(0,x'')max(0,x')max(0,x)}\,(i_m)\,, \Delta_{max(0,x'')max(0,x')max(0,x)}\,(i_n)\}\}$;

      and operation$\{\Delta_{max(0,x'')max(0,x')max(0,x)}\,(i_m)\,, \Delta_{max(0,x'')max(0,x')max(0,x)}\,(i_n)\}$;

      c.dotproduct;

    end

  end

  $L_k = \{c : \{\Delta_{x_1}\,(i_1)\,, \ldots, \Delta_{x_k}\,(i_k)\}\} \mid (c \in C_k) \wedge (c.dotproduct \geqq support)$

end

$L = \bigcup_k L_k$

Algorithm 2: テンソル多次元同時関係抽出操作