Matrix Clustering：CRM向けの新しいデータマイニング手法

小柳 滋† 久保田 和人† 仲瀬 明彦†

ネットワークビジネスにおけるマーケティングでは，CRMが重要である．本稿ではCRM向けの新しいデータマイニング手法としてMatrix Clusteringを提案する．Matrix Clusteringでは顧客と商品より構成される2値行列の行や列を入れ替えることにより密な部分行列を抽出するが，この手法の高速化を目指してピンポン法と呼ぶ新しいアルゴリズムを開発した．ピンポン法では行と列の間でマーカ伝播を繰り返しながら枝刈りを行う．これにより大規模疎行列では行や列を入れ替える方法と比べて大幅な高速化が達成され，さらに見つかる解の品質も優れていることを実験により確認した．また，Matrix ClusteringをWWWアクセスログ分析に応用し，有効なクラスタの発見が可能であることを確認した．

Matrix Clustering: A New Data Mining Algorithm for CRM

Shigeru Oyanagi,† Kazuto Kubota† and Akihiko Nakase†

A new data mining method named Matrix Clustering is proposed for CRM (Customer Relationship Management). Matrix clustering is defined to generate a dense sub-matrix from a sparse binary matrix by exchanging rows and columns. We also propose a new fast algorithm named ping-pong algorithm for clustering large sparse matrix. The ping-pong algorithm is 1,000 to 10,000 times faster than a naive algorithm, and the quality of solution is better. This algorithm is also applicable to WWW access log analysis.

1. はじめに

インターネットの急速な進展により，ECが広く普及しつつある．これにより各種の商取引データの電子化が進むと，そのマーケティングへの応用がビジネスにおいて重要になると認識されている．CRM（Customer Relationship Management）とは，さまざまな販売チャネルを通じた顧客のコンタクトや取引の履歴情報を一元管理し，個々の顧客に最適な対応を実施することにより顧客維持率を高め，長期的な企業利益を高めることを表すマーケティングのキーワードである．適切なCRMのためには，データマイニング技術が顧客の購買履歴から顧客の特性を把握するうえで強力なツールとなりうるものとして期待されている[1),2),5)]．

従来のデータマイニング技術は，マクロな傾向の分析（POSデータ分析）や，バスケット分析に威力を発揮してきた．しかし，個々の顧客を識別し，その購買履歴から個々の顧客の特徴を抽出する手法として適切なものはない．

† 新情報処理開発機構並列応用東芝研究室
RWCP Parallel Application TOSHIBA Lab

従来のマーケティング手法では，あらかじめ顧客ごとにプロファイルを作成し，特定の商品を購入した顧客セグメントを求める方法がとられている．しかしながら，顧客の嗜好や商品の属性が多様化している状況では，顧客や商品の正確なプロファイルの作成は困難であり，また次元が大きくなるため従来手法によるプロファイルを用いたセグメント化は容易でなく，CRMのためにはより強力な手法が求められている．

本論文ではMatrix Clusteringと呼ぶCRM向けの新しいクラスタリング手法と，その高速なアルゴリズムを提案する．本手法では基本的な商取引データのみを用いて顧客と商品の双方のクラスタリングを同時に行う．すなわち，顧客の特性と商品の特性を双対的なものと考え，顧客の特性は購入した商品から把握でき，商品の特性は購入した顧客から把握できると考える．

分析手法としては，商取引データを顧客と商品より構成される2値行列として表現し，2値行列の中から密な部分行列を抽出することにより関連の強い顧客集合と商品集合を求める．これより，求まった顧客集合に属する顧客は求まった商品集合に属する商品を好むと解釈する．言い換えると，「顧客は顧客特性の類似した他の顧客が購入した商品を買う可能性が高い」，お

よび「商品は商品特性の類似した他の商品を購入した顧客が買う可能性が高い」という仮説に基づいて，顧客特性と商品特性とを同時に求める手法ということができる．

本稿は次のように構成される．2 章では Matrix Clustering の概要について説明する．3 章では Matrix Clustering と従来手法とを比較し，その特徴を明らかにする．4 章では Matrix Clustering の基本アルゴリズムについて述べる．ナイーブな行・列置換法と高速なピンポン法について説明する．5 章ではこれらを用いた評価実験について述べる．6 章では実データを用いた評価実験について述べる．7 章ではまとめと今後の課題について述べる．

2. Matrix Clustering 手法の概要

本稿で提案する Matrix Clustering は，**図 1** に示すような 2 値行列による表現を対象とする．図 1 の行は顧客を示し，列は商品を示す．行列要素 A_{ij} は顧客 i による商品 j の購入状況を表す．すなわち，$A_{ij} = 1$ ならば顧客 i は商品 j を購入したことを表し，$A_{ij} = 0$ ならば購入していないことを表す．一般に顧客数，商品数は多く A_{ij} は大規模となり，行列は疎行列となる．

2 値行列 A_{ij} の行・列である顧客や商品の並びかたは任意であるため，行および列の順序を入れ替えてもよい．**図 2** では，図 1 の行・列を入れ替えることにより，密な部分行列を抽出する過程を示している．図 2 において，顧客 A，B，C と商品 a，b，c より成る密な部分行列が抽出される．この密な部分行列の表す意味を次のように解釈する．「顧客 A，B，C は商品 a，b，c に関して共通の性質を持つ」，あるいは，「商品 a，b，c は顧客 A，B，C に関して共通の性質を持つ」．また，密な部分行列内に 0 となる要素は，将来 1 となる可能性が高いと解釈する．たとえば，「顧客 B は商品 b を購入する可能性が高い」と解釈する．Matrix Clustering では，以上のように疎行列の行・列を入れ替えることにより密な部分行列を抽出し，部分行列内の 0 となる要素より将来の購買可能性を抽出する方法と定義する．

Matrix Clusterig では，抽出される部分行列に関して統計的な意味付けが必要となる．ここでは相関規則発見における Apriori Algorithm[6] で提案されたサポート値，コンフィデンス値と同様の考え方により統計的な意味付けを行う．Apriori Algorithm では，バスケット分析における商品間の組の出現頻度をサポート値，商品の組より規則を生成するときの規則の成立する確率をコンフィデンス値と定義している．Matrix

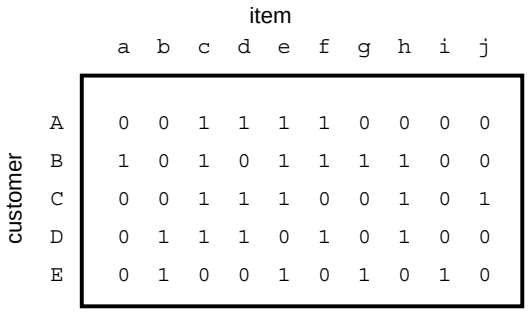

図 1 対象とする行列表現
Fig. 1 Target matrix representation.

Clustering においては，抽出された密行列の大きさ（面積）をサポート値と定義し，密行列における 1 の要素の比率（密度）をコンフィデンス値と定義する．これにより，Matrix Clustering とは与えられた疎行列の行・列を入れ替えることにより，指定された面積（サポート値）以上，指定された密度（コンフィデンス値）以上の密な部分行列を抽出することと定義する．さらに，アルゴリズムの評価を明確にするため，面積と密度の一方を拘束条件，他方を評価値とすることにより，次の 2 つの形式で問題設定する．

(1) 密度最大化：指定された面積以上で密度が最大となる部分行列を見つける．
(2) 面積最大化：指定された密度以上で面積が最大となる部分行列を見つける．

なお利用場面から考えると，特定の行あるいは列を含む密な部分行列を見つける問題，あるいは行列全体の中からサポート値，コンフィデンス値の条件を満たす部分行列を求める問題の 2 つが存在する．後者の問題は前者の問題をすべての行や列について繰り返し行うことで対処できるので，ここでは前者の問題を基本として考察する．

一般に，特定の行あるいは列を含む密な部分行列は複数個存在する．条件を満たすすべての部分行列を求めるには，探索に後戻りを必要とするため，効率的な処理が困難であり，大規模な行列において実用的な時間で解くことは非常に困難である．そこで本稿では，良質の解を高速に求める greedy なアルゴリズムについて考察する．すなわち，解の最適性を保証するのではなく，大規模な問題においても適用できるアルゴリズムの開発を目的とする．

3. 他手法との比較

3.1 クラスタリング手法との比較

データベースの中から互いに類似したレコードの集合を抽出してクラスタを形成する手法はクラスタリ

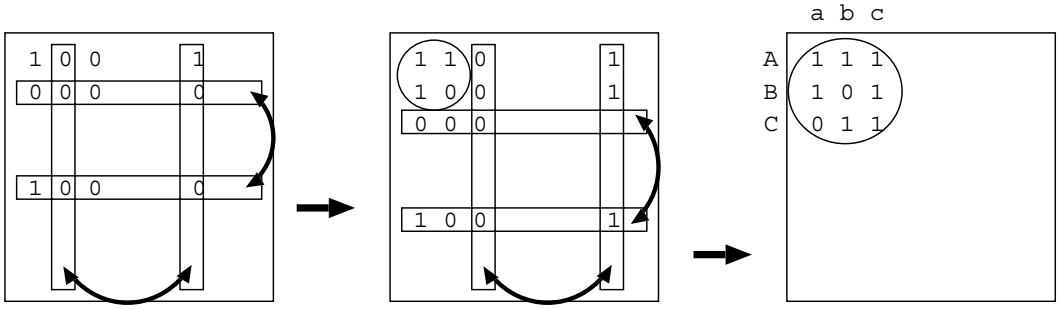

図 2 行・列の入れ替え
Fig. 2 Exchange of rows and columns.

ングと呼ばれている[1),2),5),9)]．代表的なクラスタリング手法として，K-means 法，凝縮法（agglomeration method）等が用いられている．これらのクラスタリング手法ではレコードをその属性空間上の点ととらえ，レコード間の距離やウェイトを定義することにより，距離の近いレコードをまとめてクラスタを形成する手法である．

従来のクラスタリング手法を用いて本稿で取り扱っているような顧客と商品の関連に適用するには，顧客を点として商品を属性とする方法，あるいは商品を点として顧客を属性とする方法が考えられる．いずれの場合においても，空間を張る属性を指定し，そのうえで距離やウェイトの定義をクラスタリングの前にあらかじめ指定する必要がある．

たとえば顧客を点としてクラスタリングする場合，顧客の年齢や年収等の数値属性は距離を定義しやすいが，購入商品のようなクラスタ値をとる属性に関して距離を定義することは一般に困難である．1 つの方法として個別の購入商品をそれぞれ異なる属性とし，その商品を購入したか否かを 0/1 の属性値とすることが考えられる．この場合，顧客間の距離として多数の購入商品に対応する 2 値ベクトル間のハミング距離をとることが考えられる．このような方法では，すべての属性を一律に扱うため，商品全体にわたって購入傾向が類似している顧客のクラスタを抽出することはできても，部分的な顧客と部分的な商品の間に存在する関係を抽出することはできない．顧客や商品には多様な特性があり，商品全体や顧客全体にわたって購買傾向が類似することはきわめて稀なケースと考えられるため，部分的な特徴抽出ができないことは実用的に大きな欠点であると考えられる．

以上のように，従来のクラスタリング手法と比較した Matrix Clustering の特徴は，クラスタリングにおける属性や距離の指定が不要であり，かつ行および列に関する部分的な特徴を自動的に抽出することができる点である．

従来のクラスタリング手法と Matrix Clustering の違いをデータベース設計の観点から論じる．データベースの設計手法として実体関連モデル（Entity Relationship Model）が広く知られている[3),4)]．実体関連モデルでは実体（Entity）と関連（Relationship）の 2 つを基本構成要素としてデータベースを設計する．実体関連モデルを用いて顧客と商品の関係を表現すると，顧客や商品を実体として扱い，顧客と商品の関係を関連として扱うのが一般的である．従来のクラスタリング手法は実体関連モデルにおける実体のクラスタリングに相当し，一方 Matrix Clustering は関連のクラスタリングに相当する．すなわち，顧客や商品等の実体をその属性によりクラスタリングする場合は従来のクラスタリング手法が適しており，顧客と商品の関係等の実体間の関連をクラスタリングする場合は Matrix Clustering が適している．このようにデータベースの設計論に立ち返ると，従来のクラスタリング手法と Matrix Clustering との使い分けが明確になる．

3.2 相関規則発見との比較

データマイニングの代表的な手法の 1 つとして相関規則発見[6)]が著名であり，その実用的な応用例として，バスケット分析が知られている．また，相関規則発見の高速アルゴリズムとして Apriori Algorithm が知られている．バスケット分析ではトランザクションとアイテムという 2 つの実体を用いて，同時に購入される比率の高いアイテムの組を見つける問題である．バスケット分析において，トランザクションを行，アイテムを列とした 2 値行列表現を想定すると，顧客と商品を分析するための Matrix Clustering と比較して論じることができる．

バスケット分析では，まず列数を 2 とし，行数がサポート値以上で密度が 100%の部分行列を求める．次

に，この結果を用いて SELF JOIN により列数を増やす方向で処理が進む．各段階において行数がサポート値以下のものを枝刈りすることにより，効率良く処理できる点が特徴である．しかし，列数の大きい部分行列を求める場合には組の要素数だけ SELF JOIN を繰り返し実行する必要があり，効率が低下する．

一方，Matrix Clustering では行数×列数をサポート値として部分行列を求める．また部分行列内の密度は 100%である必要はなく，コンフィデンス値以上であればよい．すなわち，バスケット分析は Matrix Clustering においてコンフィデンス値を 100%と設定し，列数を徐々に増やしながら，行数の大きい部分行列を求める問題と考えることができる．

3.3 OLAP との比較

多次元データベースを分析対象として取り扱う手法として OLAP が注目されている[7),8)]．OLAP では，各次元に関する階層的な構造をあらかじめ定義しておくことにより，ドリルダウン（drill down）やロールアップ（roll up）と呼ばれる操作を容易に実行することができ，多次元データベースの階層的な分析機能を提供している．しかし，あらかじめ設定された階層表現と異なる特徴を OLAP で把握することは困難である．

一般に商品や顧客は多くの属性を持つ．Matrix Clustering では，生の販売データから商品と顧客のクラスタを生成する方法であり，生成されたクラスタに関して，商品や顧客の属性を分析することにより，多くの属性の中でどの属性が特徴的であるかを見つけることができる．この点ではあらかじめ特定の属性に従って階層化する OLAP には不得意な機能が実現できる．すなわち，Matrix Clustering は OLAP とうまく融合させると強力な分析ツールとなりうる可能性がある．

3.4 スライス&ダイス分析との比較

我々は多次元データベースから特徴を抽出するマイニングアルゴリズムとして，スライス&ダイス分析を提案している[11),12)]．スライス&ダイス分析では多次元データベースをデータキューブに分割し，与えられた条件を満たすレコードの分布が特徴的なデータキューブ上の部分空間を探索するアルゴリズムである．また，隣接する部分空間を併合することにより，解を一般化することもできる．

スライス&ダイス分析と Matrix Clustering の大きな相違点は多次元データベースの軸に関する取扱いである．スライス&ダイス分析では軸が数値の場合は値により自動的に分割し，隣接する部分空間が共通の性質を持つときこれらを併合することにより一般化を行うことができるが，クラスタ軸に関しては隣接という概念がないため一般化できない．一方，Matrix Clustering の場合はクラスタ軸の順番を入れ替えることにより一般化を行うが，数値軸では行・列の入れ替えができないため一般化は行えない．このように，スライス&ダイス分析と Matrix Clustering は相補的な手法であり．これらを統合することにより，強力な分析機能の実現が期待される．

4. 基本アルゴリズム

4.1 行・列置換法

Matrix Clustering のナイーブなアルゴリズムとして，疎行列の行と列を入れ替えることにより，行列の左上に密な部分行列を生成するアルゴリズムについて説明する．アルゴリズムの概要を以下に示す．

```
while（停止条件）{
  x = select_row(); /* 行の選択 */
  y = select_row();
  xx = evaluate_row(x); /* 行の評価 */
  yy = evaluate_row(y);
  if (x<y && xx>yy) exchange_row(x,y);
  else if (x>y && xx<yy) exchange_row(x,y);
  x = select_col(); /* 列の選択 */
  y = select_col();
  xx = evaluate_col(x); /* 列の評価 */
  yy = evaluate_col(y);
  if (x<y && xx>yy) exchange_col(x,y);
  else if (x>y && xx<yy) exchange_col(x,y);
}
```

本アルゴリズムは 2 つの行を比較して 1 となる要素が上に集まるように入れ替えを行い，2 つの列を比較して 1 となる要素が左に集まるように入れ替えを行う．上記のアルゴリズムでは行および列の入れ替えを交互に行っている．一方を固定して他方のみを入れ替える問題ならば，評価値に基づくソーティングを行えばよい．しかし双方を入れ替えるため，一方を固定して他方のソーティングを繰り返すことが必ずしも効率良いアルゴリズムではないと考えられる．

本アルゴリズムにおいてポイントとなるのは，操作対象となる行や列を選択するための選択関数（select_row，select_col），および行や列を入れ替えるか否かを決定するための評価関数（evaluate_row，evaluate_col）である．

選択関数はランダムに選択する方法や，順番に選択する方法がありうる．いずれの場合も，充分大きな回数繰り返すことにより収束する（行や列の入替えにより評価値が改善しない状態となる）ことを実験により

確認した．すなわち，行列のサイズを $n \times n$ とすると，繰返し回数はいずれの方法でも $O(n^2)$ であった．なお，この収束値はローカルミニマムであり，最適値ではないことは明らかである．したがってシミュレーテッドアニーリング等の手法により，解の改善を図ることが可能である．

評価関数に関しては，多数回繰り返されるためできるだけ簡単である方が望ましい．そこで，以下のような評価関数を設定して実験した．いずれも，行の左側に 1 となる要素がより多く存在する行の評価値が高くなるものである．

(1) 行において，1 となる要素の右端からの位置の総和を評価値とする．
(2) 行において，1 となる要素の右端からの位置の 2 乗総和を評価値とする．
(3) 求めるべき密行列のサイズをあらかじめ指定し，行において列のサイズ内に存在する 1 となる要素の総数を評価値とする．

たとえば，行列サイズが 8×8 として行 A が (1,0,1,0,0,0,0,0) 行 B が (0,0,1,1,1,0,0,0) とすると，評価法 (1) では行 A の評価値は右端から 8 番目と 6 番目の要素が 1 であるので，8+6=14，行 B の評価値は $4+5+6=15$ となり，行 B が上となる．評価法 (2) では行 A の評価値は $64+36=100$，行 B の評価値は $16+25+36=78$ となり，行 A が上となる．評価法 (3) で求めるべき密行列のサイズを 3×3 とすると，行 A の評価値は 2，行 B の評価値は 1 となり，行 A が上となる．

なお，以下では，選択関数（select_row，select_col）としてランダムに選択する方法と隣接する行・列を調べる方法を組み合わせ，評価関数（evaluate_row，evaluate_col）として上記の (3) を採用し，収束判定としては，すべての隣接する行・列において入れ替えが生じない状態になっている状態として実験した．

4.2 ピンポン法

行・列入れ替え法ではすべての行・列を比較対象として入れ替えを行うため，行列のサイズが大きくなると実行時間が膨大となる．ここで提案するピンポン法は活性化された行と列の間でマーカ伝播を繰り返し，操作対象の行・列を小さく抑えることにより高速化を図った手法である．マーカ伝播とは，処理単位をノードで表し，処理単位間の関係をリンクで表すグラフにおいて，活性化されたノードがリンクを経由して 1 ビットのマーカを他のノードに伝播し，マーカを受信したノードが活性化される処理方式である．ピンポン法においては，2 値行列の行と列をノードとし，1 となる要素の存在する行と列を双方向リンクで結んだグラフ構造とする．このグラフ上にマーカを伝播し，受信したマーカ数に対する枝刈りを加えてノードの活性化を制御する処理モデルである．ピンポン法のアルゴリズムの概要を以下に示す．

```
while ( 停止条件 ) {
   row_to_col(); /* active な行から接続されて
                    いる列にマーカを伝播 */
   prune_col();  /* 各列の受信マーカ数より枝刈
                    りを行い，列を活性化する */
   col_to_row(); /* active な列から接続されて
                    いる行にマーカを伝播 */
   prune_row();  /* 各行の受信マーカ数より枝刈
                    りを行い，行を活性化する */
}
```

ピンポン法の処理過程を例とともに**図 3** に示す．図 3 では，与えられた行列の開始行 a を指定して処理を始める．まず，ステップ 1 で行 a を活性化し，行 a において値が 1 の要素に対応する列 (A,D,F,H,L,N) にマーカを伝播する．次にステップ 2 でマーカを受け取った列 (A,D,F,H,L,N) はすべて受信マーカ数が 1 であるので枝刈りを行わずすべて活性化され，それぞれの対応する列において値が 1 の要素に対応する行にマーカを伝播する．たとえば，列 A は行 (a,b,d,f,i,l) に，列 D は行 (a,d,f,i,k,o) にマーカを伝播する．すべての活性化された列からのマーカ伝播が終了すると（ステップ 3），各行は受信したマーカの合計を数える．ここで，合計値が閾値以下の行を枝刈りし，閾値以上の行のみを活性化する．図 3 において括弧内の数値は受信したマーカ数を表す．たとえばステップ 3 の a(6) は行 a が 6 個のマーカを受信したことを表す．また，ワクで囲まれた行が枝刈りに生き残り，活性化されたことを示す．上記のアルゴリズムでその時点に活性化されている行・列により構成される部分行列が解となる．また，活性化された行あるいは列が前回と同じならば，これ以上繰り返しても同一状態の繰返しとなるため，その時点で繰返しを終了する．

このアルゴリズムでポイントとなるのは，枝刈り戦略である．枝刈りが緩やかだと，多くの行・列が活性化されるため演算量が増え，解の面積は大きくなるが密度は低下する．一方枝刈りが厳しいと，解の密度は高くなるが面積は小さくなる．本稿では枝刈りの閾値はその時点での面積，および密度を計算することにより決める．たとえば，ステップ 3 において閾値を 4 とすると，活性化される行は a のみとなり，直前に活性化されている列数は 6 であるので，面積は $6 \times 1 = 6$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
a	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	0
b	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1
c	0	1	0	0	1	0	0	1	0	0	1	0	0	1	1	0
d	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1
e	0	1	0	0	1	1	0	0	1	0	0	0	1	0	1	0
f	1	0	1	1	0	0	1	0	0	1	0	0	0	1	0	0
g	0	1	0	0	0	1	0	1	1	0	0	0	1	0	1	0
h	0	0	0	0	1	0	0	1	0	1	0	1	0	1	0	1
i	1	0	1	1	0	0	0	0	1	0	1	0	0	0	1	0
j	0	0	1	0	1	0	0	0	0	1	0	1	1	0	0	1
k	0	1	0	1	0	0	1	0	0	0	0	1	0	1	0	1
l	1	0	1	0	0	1	0	1	0	0	1	0	0	0	1	0
m	0	1	0	0	1	0	1	0	1	0	0	1	1	0	0	0
n	0	0	1	0	0	1	1	0	0	1	0	1	0	0	1	0
o	0	0	0	1	0	0	0	1	1	0	1	0	0	1	0	1
p	0	1	0	0	1	1	0	0	0	1	1	0	1	0	0	0

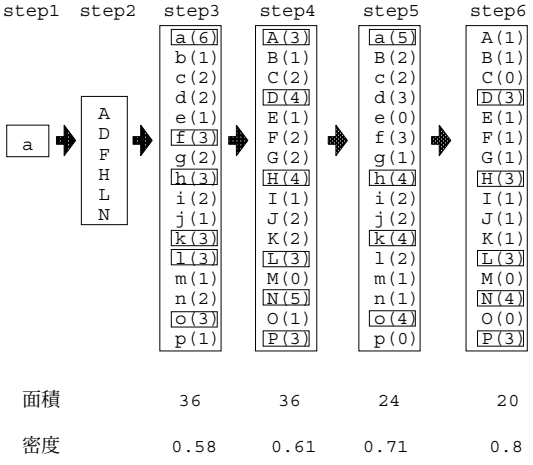

図 3 ピンポン法の処理過程
Fig. 3 Example of ping-pong algorithm.

となり，1 となる要素数は行 a が受信するマーカ数が 6 であるため，密度は 6/6 = 1 となる．また閾値を 3 とすると，a，f，h，k，l，o の 6 行が活性化される．面積は 6 × 6 = 36 となり，1 となる要素数は a，f，h，k，l，o の 6 行が受信するマーカ数の合計である 21 となるので，密度は 21/36 = 0.58 となる．このように閾値を高くすると面積は小さくなり，密度は高くなる．また，適度な閾値で本操作を繰り返すことにより，図 3 に示すように徐々に良質の解を生成することができる．

前述のように枝刈り戦略として，部分行列の面積や密度を計算しながら閾値を設定することができる．たとえば，指定された密度以上で面積最大の部分行列を求める問題と，指定された面積以上で密度最大の部分行列を求める問題は枝刈りにおける閾値を対応するように設定することにより容易に対処することができる．また，枝刈りにおいて最小の行数，列数を指定することにより，求める部分行列の形状をコントロールすることも可能となる．

なお，2 値行列表現を行（顧客）と列（商品）の 2 つの属性から構成されるリレーションと考えることができる．ピンポン法は，このリレーションに対して，SELF JOIN と GROUP BY 演算に枝刈りを加えて繰り返すことにより実行できる．

リレーショナルデータベースの基本演算の組合せを用いたマイニングアルゴリズムとして，相関規則発見のための Apriori アルゴリズム[6] がよく知られている．ここで，ピンポン法と Apriori アルゴリズムは，いずれも基本演算として SELF JOIN と GROUP BY を用いており，枝刈りにより計算量の削減を行う点で共通している．異なる点は SELF JOIN のとり方である．Apriori アルゴリズムは（transaction, item）という 2 つの属性からなるリレーションの transaction で必ず SELF JOIN をとるが，ピンポン法では（顧客，商品）という 2 つの属性からなるリレーションにおいて SELF JOIN をとる属性が交互に変わる．

5. 試作と評価

5.1 試　　作

Matrix Clustering の効果，および行・列置換法とピンポン法の 2 つのアルゴリズムの比較のため，2 種類のプログラムを試作した．いずれのプログラムも実行時のパラメータとして，以下のものを指定する．

- 実行モード：面積最大化/密度最大化を指定する．
- 最小サポート値：密度最大化の場合，満たすべき面積を指定する．
- 最小コンフィデンス値：面積最大化の場合，満たすべき密度を指定する．
- 最小行数/列数：部分行列の形状を指定する．
- 初期行/列：出発点となる行/列を指定する．これは必ず解に含まれる．
- 抑止行/列：活性化を抑止する行/列を指定する．

Matrix Clustering では一般に大規模な疎行列を対象とするため，2 値行列のデータ構造が重要となる．行・列置換法，ピンポン法とも行列に対する操作は，指定された行あるいは列における 1 となる要素の存在する位置を求める点で共通している．2 値行列は疎行列であることを考慮し，行列は**図 4** に示すようなデータ構造でメモリ上に格納した．

図 4 において，各行，各列はエントリを有し，1 となる行列要素は要素領域に格納される．要素領域は行列の位置 (x,y)，同一行の次の要素へのポインタ (next_x)，同一列の次の要素へのポインタ (next_y) により構成される．これにより，指定された行あるいは列における 1 となる要素のみをエントリから直接ポインタをたどって求めることができ，疎行列の場合に高速なアク

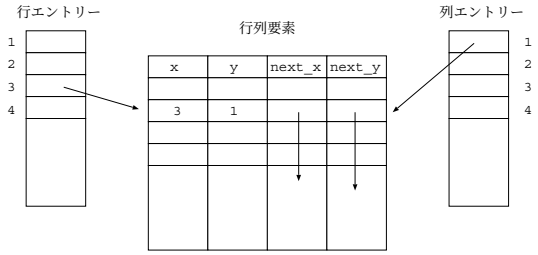

図 4 データ構造
Fig. 4 Data structure.

セスが可能となる．また領域サイズは 1 となる要素の数のみに依存するため，疎行列の場合はるかに小規模である．必要な領域サイズは 1 レコードあたり 16 バイトであり，行列のサイズの影響はエントリ部分のみであるので，たとえば行列のサイズが 10 万 × 10 万でも，要素数 100 万の規模ならメモリ上に充分展開可能である．なお，行列要素がメモリに入りきらない場合には，行あるいは列単位でディスクにアクセスする必要があるが，繰返しにおいて同一の行や列にアクセスすることが多いため，適切なバッファ管理を行うことが重要と思われる．

5.2 評　　価

Matrix Clustering における行・列置換法とピンポン法の性能を比較するため，ランダムデータを用いて評価実験を行った．以下では，実行時間，パターン検出の 2 つの観点から比較評価を行う．

5.2.1 実 行 時 間

ピンポン法と行・列置換法について実行時間を比較する．実行時間については，行列の大きさ，および密度が影響するため，両者を変化させて実行時間を測定した．それぞれの方法について測定した結果を**表 1**, **2** に示す．

- ピンポン法
  ピンポン法については，単一の行を開始行として指定し，その行を含み指定された密度以上となる面積最大の密な部分行列を求める問題において測定を行った．ピンポン法の実行時間は解により異なる．すなわち，中間状態で大きなクラスタを取り扱う場合は多大な処理時間を要する．ここでは初期値を変えた数回の実行の計算について平均実行時間を求めた．
  ピンポン法の計算量について検討する．計算量は活性化される行や列の数により変わる．行列のサイズを $n \times n$，密度を d とすると，最初のステップで活性化される行あるいは列数の平均は dn であり，次のステップでマーカが伝播される行あるいは列数は d^2n^2 となる．ここで，枝刈りにより活性化される行あるいは列数が一定（dn）であると仮定すると，全体の計算量はこの繰返しとなる．ピンポン法では収束が早く，繰返しは一定回数と考えられるので，トータルの計算量は $O(d^2n^2)$ と予測できる．実験結果では密度の高い場合にこの予測を若干上回っているが，これは中間状態で大きなクラスタとなることが原因と思われる．
- 行・列置換法
  行・列置換法についても，ピンポン法と同様に単一の行を開始行として指定し，その行を含む密な部分行列を求める問題において測定を行った．行・列置換法の実行時間はピンポン法に比べて解のサイズにそれほど依存しない．
  行・列置換法の計算量について検討する．行列のサイズを $n \times n$，密度を d とすると，行・列の入れ替え回数は前述のように n^2 のオーダであり，入れ替えに要する計算量は行内で 1 となる要素の数，すなわち dn のオーダであるので，全体の計算量は $O(dn^3)$ と予測される．表 2 より，ほぼこの予測に沿っていることが分かる．
- ピンポン法と行・列置換法の比較
  ピンポン法と行・列置換法の実行時間の比較を**図 5**，**6** に示す．図 5 では密度一定（1%）で行列サイズを変化させたときの実行時間を，図 6 では行列のサイズ一定（1024 × 1024）で密度を変化させたときの実行時間を表す．
  図 5，6 より，行列サイズが大きく（512×512 以上），かつ密度が低い（5%以下）場合にピンポン法は行・列置換法に比べて 1,000 倍以上高速であることが分かる．図 5 より，グラフの傾きは行・

表 1 ピンポン法の実行時間（秒）
Table 1 Execution time of ping-pong algorithm (sec).

	1%	2%	5%	10%	20%
128 × 128	0.00044	0.00069	0.0010	0.0028	0.017
256 × 256	0.00063	0.0016	0.0038	0.019	0.11
512 × 512	0.002	0.0029	0.024	0.12	1.097
1024 × 1024	0.0075	0.0156	0.163	1.29	5.44

表 2 行・列置換法の実行時間（秒）
Table 2 Execution time of exchange algorithm (sec).

	1%	2%	5%	10%	20%
64 × 64	0.009	0.014	0.020	0.051	0.087
128 × 128	0.051	0.083	0.20	0.41	0.67
256 × 256	0.42	0.53	1.35	2.28	5.32
512 × 512	2.55	4.43	9.12	22.8	54.4
1024 × 1024	13.0	37.3	107.4	198.1	538.4

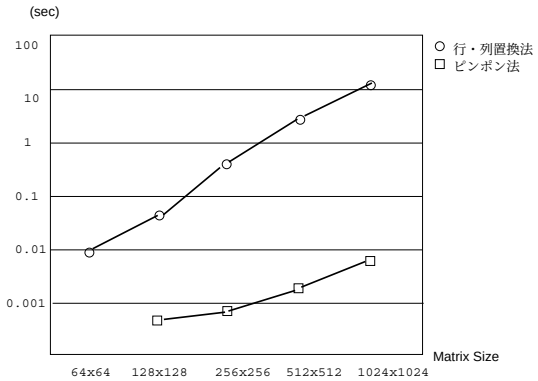

図 5 実行時間の比較
〔密度一定（1%）でサイズを変化させた場合〕
Fig. 5 Comparison of execution time with changing matrix size.

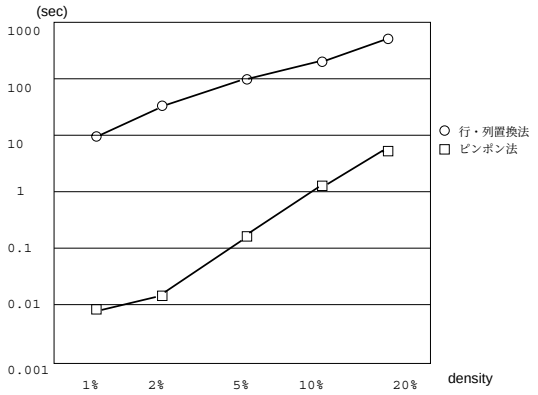

図 6 実行時間の比較
〔サイズ一定（1024 × 1024）で密度を変化させた場合〕
Fig. 6 Comparison of execution time with changing density.

列置換法の方が大きい．一方，図 6 より，グラフの傾きはピンポン法の方が大きい．これは両者の計算量の予測からも読み取れる．これらより，行列サイズが大きく，かつ密度が低い場合にピンポン法は高速なアルゴリズムであると結論づけられる．

5.2.2 パターン検出

解の品質を比較評価するため，ランダムな行列から抽出された部分行列を比較すると 2 つのアルゴリズムで異なる部分行列が抽出されるため，評価が困難であることが分かった．そこで，あらかじめ最適解が予測できる例を用いて実験を行った．すなわち，サイズ 100×100，密度 10%の行列において，5×5 のパターンをあらかじめ埋め込み，このパターン中の 1 行を開始行とすることにより，埋め込まれたパターンが検出できるか否かの実験を行った．パターンとしては，次のような密度 80%の行列を用いた．

1 1 1 1 0
1 1 1 0 1
1 1 0 1 1
1 0 1 1 1
0 1 1 1 1

1 パターンについて 5 個の行を開始行として 5 回の実験を行い，5 つのパターンについて合計 25 回の実験を行った．結果を，5×5 パターン全体が検出された場合（全体検出），5×5 パターンの半分以上が検出された場合（部分検出），半分以下しか検出されなかった場合（検出不能）に分類したものを，以下に示す．

	全体検出	部分検出	検出不能
行・列置換法	0	2	23
ピンポン法	15	6	4

これより明らかに埋め込みパターンの検出に関してはピンポン法が行・列置換法より優れていることが分かる．なお，行・列置換法においてはほとんどの場合極小解に収束し，埋め込みパターンが検出できなかったが，シミュレーテッドアニーリング等を用いることにより解の品質の向上が可能である．しかし，行・列置換法の計算量をこれ以上増加させるとピンポン法との実行時間の差がますます広がるため，本稿ではこれらの改良については考察しない．

6. WWW アクセスログを用いた評価

6.1 クラスタの抽出

Matrix Clustering の実データによる評価として WWW アクセスログを用いた実験を行った[10]．実験に用いたログにはユーザを特定する情報（cookie）と WWW ページを特定する情報（URL）が含まれており，これらを抽出して URL を行，cookie を列とする Matrix を生成し，これから密な部分行列を抽出する．

実験に用いたログデータは，2,251 種類の URL と 1,223 種類の Cookie より構成される 66,101 レコードである．Web ページはトップページをルートとする階層構造となっている．ログ中の URL には gif，jpg ファイル等が含まれ，また cgi により生成されるページも含まれている．本実験では，ユーザと Web ページとの関係を分析することを主眼とするため，URL は HTML ファイルに限定し，また明らかにアクセス頻度の高いトップページは本手法には適さないため除外した．また，検索のようにアクセス時に生成されるページは cookie ごとに異なる URL が用いられるのでクラスタリングが困難となる．このため，これらを使

われ方によりグルーピングした．その結果，対象とする URL は合計 480 種類となり，cookie は 1,223 種類すべてを用い，ログデータはこれらに該当するもののみとして，合計 14,416 レコードとなった．また，この中で同一ユーザが同一ページをアクセスする場合が 7,480 回含まれるため，行列中の 1 となる要素数は 6,936 となり，行列全体の密度は 1.4%となった．

まず，各行を開始行としてすべての行についてピンポン法を適用し，クラスタを求めた．実行パラメータは密度 80%以上で面積最大とする問題とし，最小行数，列数を 3 とした．その結果，以下のような大きなクラスタが見つかった．

- 行・列とも大きなクラスタ

クラスタ	面積（行 × 列）	密度
A	$38 \times 25 = 950$	86%
B	$24 \times 57 = 1392$	86%
C	$20 \times 46 = 920$	80%

- 列の大きなクラスタ

クラスタ	面積（行×列）	密度
D	$5 \times 195 = 975$	90%
E	$3 \times 229 = 687$	90%
F	$6 \times 111 = 666$	100%

- 行の大きなクラスタ

クラスタ	面積（行×列）	密度
G	$18 \times 3 = 54$	100%
H	$16 \times 3 = 48$	100%

例としてクラスタ B のパターンを**図 7** に示す．

なお，上記のような大きなクラスタには階層の上位の URL が多く含まれている．このような URL は WWW ページの構成上アクセス頻度が高くなるのは明らかであり，Web ページのアクセログの分析においては，アクセス頻度のそれほど高くない階層的に下位のページに関する分析が重要である．さらに，大きなクラスタの存在は小さなクラスタの発見を妨害する可能性が強い．そこでアクセス頻度の高い上位の階層のページの活性化を抑止して実験を行うことにより，自明ではない階層的に下位のページ間のクラスタを見つけることができた．その例を以下に示す．

クラスタ	面積（行 × 列）	密度
P	$18 \times 3 = 54$	100%
Q	$16 \times 3 = 48$	100%
R	$4 \times 6 = 24$	90%
S	$11 \times 3 = 33$	100%

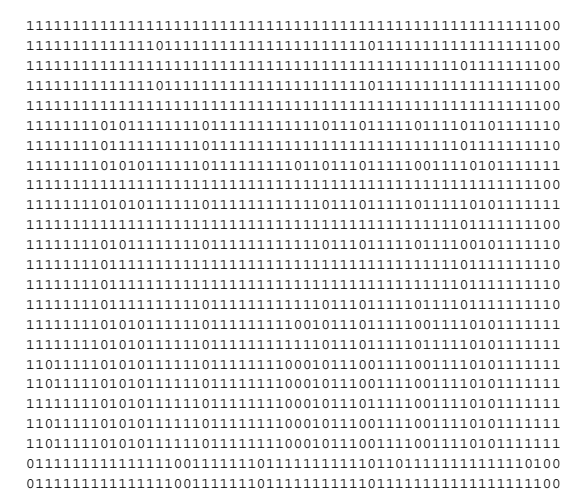

図 7 密な部分行列（クラスタ B）のパターン例
Fig. 7 Example of a pattern of dense sub-matrix (Cluster B).

6.2 Apriori アルゴリズムとの比較

前述のように，Apriori アルゴリズムは高速な相関規則発見アルゴリズムとして著名なアルゴリズムである．Web アクセスログの分析において，cookie をトランザクション，URL をアイテムと考えると，Apriori アルゴリズムにより出現頻度がサポート値以上の URL の組をすべて抽出することができる．ここでは，Matrix Clustering の実験データと同一のデータを用いて Apriori アルゴリズムにより抽出される URL の組と，Matrix Clustering により抽出されるクラスタの比較を行うことにより Matrix Clustering の性質を明らかにする．

Apriori アルゴリズムの実行時間はサポート値に依存する．ここでは Matrix Clustering に要した実行時間と同等程度の時間で得られる結果を比較することとし，サポート値を 44 として比較を行った．なお，Apriori アルゴリズムで求まる URL の組において出力される組の部分集合もまた解として含まれる．すなわち，(A,B,C) が解ならば (A,B)，(A,C)，(B,C) もすべて解に含まれる．本実験で求めたいものは他の解の部分集合になっていない組のみであるので，出力される組の中からこれらを抽出した結果を**表 3** に示す．

表 3 より，Apriori アルゴリズムにより求まる解の大部分は他の解の部分集合であり，ここで求めたいクラスタに相当する解は数個程度しかないことが分かる．この中で，長さ 9 以上の URL の組はすべて 6.1 節で得られたクラスタ A，B に含まれていることが分かった．すなわち，クラスタ A，B より 0 を含む行・列を削除すれば Apriori アルゴリズムにより求まった大き

表 3 Apriori アルゴリズムによる URL の組
Table 3 Set of URLs obtained by Apriori algorithm.

組の長さ	総組数	上位の解の部分集合でない組数
2	177	5
3	705	4
4	2066	1
5	4465	2
6	7278	0
7	9104	1
8	8819	0
9	6623	1
10	3828	0
11	1673	0
12	535	2
13	118	0
14	16	1
15	1	1

な組がすべて得られる．

Apriori アルゴリズムはサポート値以上の出現頻度を持つ解をすべて出力する．しかし，サポート値を低くすると実行時間は膨大となり，適切なサポート値の設定が困難である．2 値行列の操作として考えると，行と列との取扱いが対称ではないため，行あるいは列の大きなクラスタは得られるが，行および列の大きなクラスタを得ることは困難である．また，得られた解はそれぞれ独立であり，それらを合成してさらに大きなクラスタを形成するには．十分な情報が含まれていない．

一方，Matrix Clustering における面積や密度の指定はサポート値の設定に比べて容易であり，パラメータ設定による実行時間の違いは少ない．また，行と列を対称的に取り扱うため，行の大きなクラスタ，列の大きなクラスタ等のさまざまなクラスタを得ることができる．さらに，出力されたクラスタより Apriori アルゴリズムで得られる解を生成することも可能である．以上より，Web アクセスログのように行・列とも大きなクラスタが抽出できる場合には，Matrix Clustering が Apriori アルゴリズムに比べてより有効な情報が得られると考えられる．

7. おわりに

EC における CRM（Customer Relationship Management）向けのデータマイニング手法として考案した Matrix Clustering について述べた．Matrix Clustering は，大規模な疎行列の行と列を入れ替えることにより密な部分行列を抽出する手法であり，その高速アルゴリズムとしてピンポン法を提案した．ピンポン法は単純な行・列置換法と比較して行列のサイズが大きく（512 × 512 以上），密度が低い（5%以下）場合に千倍以上の高速化が可能であり，さらに見つかる解の品質も優れていることを実験により確認した．また，Matrix Clustering を WWW アクセスログ分析に応用し，有効なクラスタの発見が可能であることを確認した．

なお，本稿で述べたピンポン法は Greedy アルゴリズムである．すなわち最適性の保証はない．今後の課題として，なんらかの理論的保証を与えるアルゴリズムの開発が望まれる．また，実用面での改良として，品質の良い解を確実に抽出するためのパラメータの調整方法や，メモリに格納できない大規模データへの適用方法等があげられる．さらに，多次元行列，多値行列が取り扱えるようにアルゴリズムを拡張することも考えられる．

今後，これらの研究課題に加えて，他のデータマイニング手法との融合[11)~13)]．や，並列化による大規模データの高速化を目標として研究を進める予定である．

参考文献

1) Chen, M-S., Han, J. and Yu, P.S.: Data Mining: An Overview from a Database Perspective, *IEEE Trans.Knowledge and Data Engineering*, Vol.8, No.6, pp.866–883 (1996).
2) Fayyad, U. and Simoudis, E.: Data Mining and KDD: An Overview, *3rd Intl. Conf. on Knowledge Discovery & Data Mining* (1997).
3) Ullman, J.D. and Widom, J.: *A First Course in Database Systems*, Prentice Hall (1997).
4) Chen, P.P.: The entity-relationship model: Toward a unified view of data, *ACM Trans. Database Syst.*, Vol.1, No.1, pp.9–36 (1966).
5) マイケル J.A. ベリー，ゴードン・リノフ：データマイニング手法，海文堂 (1999).
6) Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules, *Proc. 20th VLDB Conf.*, pp,487–499 (1994).
7) Chaudhuri, S. and Dayal, U.: An Overview of Data Warehousing and OLAP Technology, *SIGMOD Record*, Vol.26, No.1, pp.63–74 (1997).
8) 豊島，木村：OLAP 実践データウェアハウス，日本経営科学研究所 (1997).
9) Hinneburg, A. and Kelm, D.A.: Clustering Techniques for Large Data Sets, *KDD-99* (1999).
10) Wu, K.-L., Yu, P.S. and Ballman, A.: SpeedTracer: A Web Usage Mining and Analysis Tool, *IBM Systems Journal*, Vol.37, No.1 (1998).

11) Nakase, A., Kubota, K., Sakai, H. and Oyanagi, S.: Parallel Classification Methods on PC Clusters, *2000 RWC Symposium*, pp.53–59 (2000).
12) 仲瀬，久保田，酒井，小柳：表データからの傾向抽出方式と並列化手法．情報処理学会研究報告，99-HPC-77（SWoPP'99），pp.155–160 (1999).
13) 久保田，仲瀬，酒井，小柳：決定木の並列化とその評価，情報処理学会研究報告，99-HPC-77（SWoPP'99），pp.161–166 (1999).
14) 久保田，仲瀬，酒井，小柳：PC クラスタを用いた決定木生成，情報処理学会研究報告，2000-HPC-80（HOKKE2000），pp.113–118 (2000).
15) Kubota, K., Nakase, A., Sakai, H. and Oyanagi, S.: Parallelization of Decision Tree Algorithm and its Performance Evaluation, *Proc. 4th HPC-ASIA2000*, pp.574–579 (2000).
16) Kubota, K., Nakase, A. and Oyanagi, S.: Implementation and Performance Evaluation of Dynamic Scheduling for Parallel Decision Tree Generation, *Proc. 4th Workshop on PDDM 2001*, p.157 (2001).
17) 小柳，久保田，仲瀬：Matrix Clustering：CRM 向けの新しいデータマイニング手法．信学技報，DE2000-94, pp.25–32 (2000).

(平成 12 年 9 月 25 日受付)
(平成 13 年 5 月 10 日採録)

小柳　滋（正会員）

昭和 24 年生．昭和 52 年京都大学大学院工学研究科数理工学専攻博士課程修了．同年（株）東芝入社．現在（株）東芝研究開発センターコンピュータ・ネットワークラボラトリ研究主幹．北陸先端科学技術大学院大学客員教授．並列処理，データベース，データマイニング等の研究開発に従事．工学博士．IEEE-CS，ACM，電子情報通信学会各会員．

久保田和人（正会員）

昭和 39 年生．昭和 63 年早稲田大学理工学部電子通信学科卒業．平成 5 年同大学大学院理工学研究科博士課程修了．同年（株）東芝入社．平成 7 年 10 月より平成 10 年 9 月まで技術研究組合新情報処理開発機構に出向．工学博士．並列計算機のプログラミング環境，性能評価環境，計算機クラスタシステム，データマイニング等の研究開発に従事．電子情報通信学会会員．

仲瀬　明彦（正会員）

昭和 36 年生．昭和 61 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．平成 5 年 4 月より平成 7 年 3 月まで（財）新世代コンピュータ技術開発機構に出向．並列処理，データマイニングに関する研究に従事．電子情報通信学会会員．