次元数のばらつきに対応した一般射影クラスタリング

古瀬一隆[†] 石川雅弘[††]
陳漢雄[†] 大保信夫[†]

本稿では，高次元空間中の実データに対して有効な一般射影クラスタリングの手法を提案する．クラスタリングは傾向分析や近傍検索などのさまざまな分野に用いられる重要なツールの 1 つであるが，高次元データについては次元の呪いの問題により既存の手法が有効でないことが知られている．この問題を解決する有効な方法の 1 つとして一般射影クラスタリングがある．本稿ではこれまでに提案された一般射影クラスタリングの手法の「すべてのクラスタの次元数が同一でなければならない」という制限を取り除いた新たな手法を提案する．提案手法では，クラスタリングの処理の過程で得られる固有値を利用してそれぞれのクラスタの次元数を決定する．これにより，高次元データについてより良いクラスタを発見することが可能となる．提案手法の有効性は，合成データを用いたシミュレーションの結果によって検証されている．

A Generalized Projected Clustering Method for Finding Clusters of Different Dimensionality

Kazutaka Furuse,[†] Masahiro Ishikawa,[††] Hanxiong Chen[†] and Nobuo Ohbo[†]

This paper proposes a novel method of generalized projected clustering which is effective for high dimensional data. Although clustering is an important tool for various kinds of fields such as trend analysis and similarity search, it is well known that most of existing mothods are not effective for high dimensional data because of the curse of dimensionality. One of the promising approaches for this problem is the generalized projected clustering. However, formerly proposed method is not so effective when clusters have different dimensionality. The method proposed in this paper exploits the eigenvalues which are aquired during the process of clustering, and determines how many dimensions are meaningful for each cluster. With this mechanism, we can canstruct a method which is effectively applicable to real datasets in high dimensional space. The results from experimental simulations verify the effectiveness of the proposed method.

1. 序　　論

クラスタリング（clustering）は，多次元空間中の点として表現されるデータ集合から，互いの距離が近いような点の集合（これを**クラスタ**（cluster）という）を発見する手法である．クラスタリングは，傾向分析や近傍検索，パターン認識といったさまざまな分野に用いられる重要なツールの 1 つとなっている．

クラスタリングについては，これまでに多くの手法が開発されている[1)~3)]．低次元空間のデータ集合に対しては，これらの手法により，効率良くクラスタを発見することができる．しかし，これらを高次元空間のデータ集合に対して適用すると，一般に良い結果が得られない．これは高次元データの性質そのものに由来する問題である．近年の理論的な研究により，次元が高くなるに従ってデータ空間が疎になると同時に，空間中のすべての 2 点間の距離がほとんど等しくなることが明らかになっている[4)~7)]．これらの研究の結果は，高次元データのクラスタリングが本質的に難しい（あるいは，そもそも高次元空間中には通常の意味でのクラスタが存在しない）ということを示している．このような高次元データ特有の問題を一般に**次元の呪い**（curse of dimensionality）という[8),9)]．

こういった問題を解決する有効な手段の 1 つとして，射影クラスタリング（projected clustering）があ

† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
†† 農業生物資源研究所
National Institute of Agrobiological Sciences

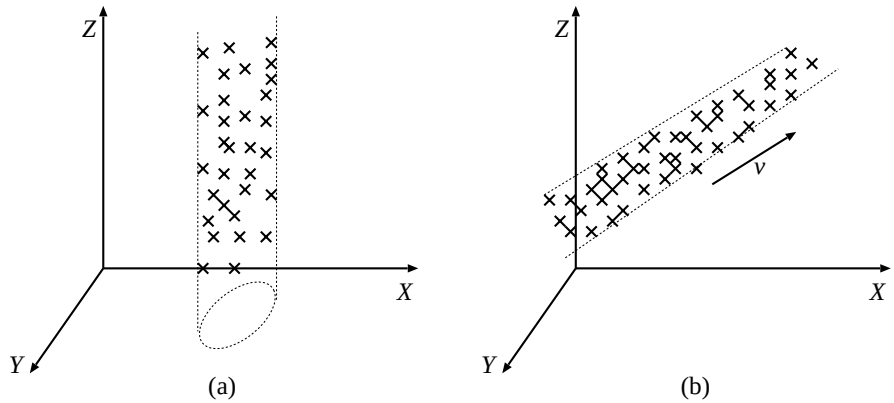

図 1 射影クラスタリング
Fig. 1 Projected clustering.

る[10)~12)]．これは，元の高次元空間から，クラスタが存在するような低次元の部分空間を発見するという手法である．発見された部分空間中では，元の高次元空間から射影されたデータ集合の一部（部分集合）がクラスタを形成する．

図 1 は，射影クラスタリングの例である（個々の“x”がそれぞれ 1 つのデータを表している）．図 1 (a) のデータ集合は，Z 軸の方向に一様分布しており，3 次元空間中ではクラスタとなっていない．しかし，これを X-Y 平面に射影したものはクラスタを形成している．高次元空間のデータ集合の場合，ある部分集合が，ほとんどの次元では一様に分布し，限られたいくつかの次元からなる部分空間ではクラスタを形成するということがしばしば起こりうる．射影クラスタリングはそのようなクラスタを発見する手法であり，これまでに PROCLUS[10)] およびそれを改良したもの[12)] が提案されている．

当然のことながら，実際の高次元データを考えた場合，すべてのクラスタが座標軸に平行に分布するということは考えられない．これに対処するため，任意の方向に分布するクラスタを発見する手法が，一般射影クラスタリング（generalized projected clustering）である．図 1 (b) は，ベクトル v の方向に分布するデータ集合であり，v を法線ベクトルとする平面に射影したときにクラスタを形成する．これはすなわち，この平面によって定められる 2 次元の部分空間中でデータ集合がクラスタを形成するということである．このように，一般射影クラスタリングは，与えられた高次元データ集合から，いくつかの直交したベクトル（本稿ではこれを**クラスタリングベクトル**という）によって定められる低次元の部分空間と，その部分空間中でクラスタを形成するデータ集合を発見する．

これまでに提案された一般射影クラスタリングの手法として，ORCLUS[11)] がある．ORCLUS は，d 次元空間中のデータを段階的に l 次元まで落としながら射影クラスタリングを行い，最終的に k 個のクラスタを発見する（l および k はユーザが与える）．ORCLUS が発見するクラスタはすべて l 次元の部分空間中でクラスタを形成することになるが，実際の高次元データでは，すべてのデータが同じ次元数の部分空間において最も良いクラスタを形成するとは限らない．むしろ，クラスタごとに次元数が異なる（それぞれのクラスタの部分空間を定めるクラスタリングベクトルの数が異なる）のが一般的である．

本稿では，このような ORCLUS の問題を解決する新しい一般射影クラスタリングの手法を提案する．この手法では，ORCLUS と同様の方法で部分空間の次元数を段階的に絞り込むが，固有値を利用してそれぞれのクラスタの部分空間の次元数を定めるため，ORCLUS の持つ「発見するすべてのクラスタの部分空間が同じ次元数となってしまう」という制限が取り除かれている．これにより，より一般的な高次元データ集合に対して適切にクラスタを発見することが可能となる．シミュレーションによる結果は，次元数が異なるクラスタを持つデータ集合に対して，本稿の提案手法が ORCLUS よりも優れていることを示している．

本稿の構成は以下のとおりである．まず，2 章において，関連研究の概略を示す．3 章では，本稿の提案手法について，その考え方とアルゴリズムの詳細を記述する．4 章には，提案手法の有効性を検証するために行ったシミュレーションの結果を示す．最後に，結論と今後の課題を 5 章で述べる．

2. 関連研究

これまでに考案された射影クラスタリングの手法としては，PROCLUS と ORCLUS がよく知られている．

PROCLUS [10] は，座標軸に平行な分布を持つクラスタを発見する手法である．この手法では，山登り法によって medoid を交換しながら良い medoid 候補を見つけ出し，最終的にそれぞれの medoid について特徴的な座標軸（座標値の分散が小さい座標軸）を用いて部分空間を作る．

PROCLUS ではそれぞれの medoid に対して定められる locality と呼ばれる領域を用いてデータ集合を分配しているが，この方法では本来分配されるべき medoid とは異なる medoid にデータが分配されてしまうことがある．次元数を段階的に絞り込みながらデータを再分配することでより精度を高めた射影クラスタリングの手法が，文献 12) で提案されている．

ORCLUS [11] は，PROCLUS よりも一般化された射影クラスタリングの手法である．PROCLUS が元の高次元空間の座標軸と同じ座標軸を用いて部分空間を定めるのに対し，ORCLUS は良いクラスタを形成する任意の方向のベクトルを見つけ出し，そのベクトル集合によって部分空間を定める．

ORCLUS では，ユーザが指定するパラメータとして，以下を用いる．

- 見つけ出すクラスタ数 k
- 各クラスタの部分空間の次元数 l

ORCLUS は，元の高次元空間中のデータから $k_0\,(>k)$ 個のクラスタを作り，それを繰り返しマージしながら最終的に k 個のクラスタを作る．このとき，繰返しの各段において，クラスタを形成する次元数を元のデータ集合の次元数 d から段階的に減らしていき，最終的に l 次元の部分空間を作る．

最終的な結果として，ORCLUS は以下を出力する．

- クラスタ $\{C_1,\ldots,C_k\}$
- クラスタリングベクトル集合 $\{E_1,\ldots,E_k\}$

クラスタ C_i に対するクラスタリングベクトル集合 E_i には，l 個のベクトルが含まれる．C_i 中のデータは，E_i によって定められる部分空間に射影されたときにクラスタとなる（すなわち，そのクラスタに含まれるデータ間の距離が短くなる）．

ORCLUS が出力するクラスタは，部分空間の次元数がどれも等しくなるので，それぞれのクラスタに対する部分空間の次元数にばらつきがあるようなデータ集合に対しては，良い結果を出力することができない．

3. 提案手法

ここでは，本稿で提案する新しい一般射影クラスタリングの手法について述べる．

3.1 記法と定義

まずはじめに，以降の説明に必要となる記法と定義を導入する．

本稿が提案する一般射影クラスタリングの手法が対象とする個々のデータは d 次元空間中の 1 つの点であり，ベクトル $\boldsymbol{x}=(x_1,\ldots,x_d)$ によって表される．全データ集合を D とし，D に含まれるデータの数を N とする．すなわち，$|D|=N$ である☆．

データ集合 $C=\{\boldsymbol{x_1},\ldots,\boldsymbol{x_n}\}$ をクラスタとするとき，クラスタの重心（centroid）を $cent(C)$ で表す．重心は，以下のように定義される．

$$cent(C)=\frac{\sum_{i=1}^{n}\boldsymbol{x_i}}{n} \tag{1}$$

2 つのデータ $\boldsymbol{x_1}$，$\boldsymbol{x_2}$ 間の距離は，$dist(\boldsymbol{x_1},\boldsymbol{x_2})$ で表す．本稿ではユークリッド距離を用いる．

$\boldsymbol{x}=(x_1,\ldots,x_d)$ を d 次元空間中の点とし，$E=\{\boldsymbol{e_1},\ldots,\boldsymbol{e_l}\}$ を $l\,(\leq d)$ 個のベクトルからなる d 次元空間における正規直交ベクトル集合とすると，E は元の d 次元空間の部分空間となる．点 $\boldsymbol{x}$ の部分空間 E への射影 $P(\boldsymbol{x},E)$ は，l 次元空間内の点 $(\boldsymbol{x}\cdot\boldsymbol{e_1},\ldots,\boldsymbol{x}\cdot\boldsymbol{e_l})$ として与えられる．ここで，$\boldsymbol{x}\cdot\boldsymbol{e_i}$ は $\boldsymbol{x}$ と $\boldsymbol{e_i}$ の内積を表す．

d 次元空間中の 2 点 $\boldsymbol{x_1}$，$\boldsymbol{x_2}$ 間の，l 次元部分空間 E における**射影距離**（projected distance）とは，2 点の E への射影 $P(\boldsymbol{x_1},E)$，$P(\boldsymbol{x_2},E)$ 間の E におけるユークリッド距離のことを指し，$Pdist(\boldsymbol{x_1},\boldsymbol{x_2},E)$ と表す．

C をクラスタ，E を l 次元部分空間とするとき，以下のように定義される量 $R(C,E)$ を**射影エネルギー**（projected energy）という．

$$R(C,E)=\frac{\sum_{i=1}^{n}\{Pdist(\boldsymbol{x_i},cent(C),E)\}^2}{n}$$

射影エネルギーが小さいほど，クラスタ C は部分空間 E において互いに近い距離にあることになり，したがってそのようなクラスタは良いクラスタであるといえる．このことから，射影エネルギーは一般射影クラスタリングにおいてクラスタの評価値として用いられる．ただし，本稿で提案する手法では，次元数の異なるクラスタの比較を行うので，上記のように定義される射影エネルギーをそのまま使うことはできない．

☆ 本稿では，集合 S の濃度（cardinality）を $|S|$ によって表す．

この点についての詳細は後述する．

3.2 分散共分散行列の対角化

ORCLUS および本稿の提案手法では，分散共分散行列（covariance matrix）の対角化の性質を利用している．その概略は以下のとおりである．

d 次元空間中の点集合 C に対して，i 番目の次元と j 番目の次元の座標値の共分散を (i,j) 要素とする（対角要素は各次元の分散となる）分散共分散行列を M_C とすると，この行列は $M_C = P\Delta P^t$ の形で表すことができる．ここで，Δ は非負の要素を持つ対角行列であり，その対角要素 $\lambda_1,\ldots,\lambda_d$ は M_C の固有値となる．また，P の各列 $\boldsymbol{p_1},\ldots,\boldsymbol{p_d}$ は M_C の正規直交固有ベクトルとなり，それぞれ対応する固有値が $\lambda_1,\ldots,\lambda_d$ となる．このとき，固有ベクトルを新たな軸（基底）とした座標系を考えると，Δ の対角要素は，元の空間における各点をこの座標系に射影した場合の各座標の分散を表すことになる．

ここで，Δ の対角要素の固有値 λ_i のうち最も小さい $l\,(\leq d)$ 個を選び，それぞれの固有値に対する固有ベクトルによって作られる l 次元の部分空間を $\hat{E}$ とする．このとき，$\hat{E}$ は，M_C の固有ベクトル $\boldsymbol{p_1},\ldots,\boldsymbol{p_d}$ から l 個を選んで作る l 次元の部分空間の中で座標系の分散の合計が最も小さいものとなる．

先に示したように，点集合 C の射影エネルギー $R(C,E)$ が小さいほど，C は部分空間 E において良いクラスタとなると考えられる．したがって，点集合 C に対して上に述べたような方法で l 次元部分空間 $\hat{E}$ を求めると，$\hat{E}$ は E の l 次元部分空間の中で C の最も良いクラスタを形成する部分空間となる．

3.3 クラスタリングアルゴリズム

ORCLUS は任意の軸の方向に分布するクラスタを見つけるのに有効な手段であるが，先にも示したとおり，実際の高次元データに適用した場合にはその精度が問題となることがしばしば起こりうる．

ORCLUS では段階的に次元を絞り込んでいくが，すべてのクラスタを（クラスタリングの過程においても，また，クラスタリングの最終的な出力においても）同じ次元数の部分空間に縮小していくので，結果としてすべてのクラスタが同じ次元数を持つことになる．しかし，実データではすべてのクラスタが必ず同じ次元数を持つとは考えにくい．

この問題に対処するため，本稿では，この制限を取り除いた新しい一般射影クラスタリングの手法を提案する．この手法は，ORCLUS と同様に，クラスタを繰り返しマージしながら段階的に次元数を絞り込んでいく．ORCLUS と異なるのは，マージするクラスタの次元数が必ずしも等しくないという点である．これを実現するため，各クラスタの部分空間を決定するベクトルの選び方や，クラスタの評価の方法などに，ORCLUS とは異なる手法を用いている．

提案手法においてユーザが指定するパラメータは，以下に示すとおり，ORCLUS とほぼ同様である．

- 見つけ出すクラスタ数 k
- 各クラスタの部分空間の次元数の平均値 l

出力される結果は，以下のとおりである．

- クラスタ $\{C_1,\ldots,C_k\}$
- クラスタリングベクトル集合 $\{E_1,\ldots,E_k\}$

ORCLUS では出力されるクラスタリングベクトル集合が同じ濃度を持つ（$|E_1| = \cdots = |E_k|$）のに対し，提案手法ではベクトル集合の濃度の平均がユーザによって与えられたパラメータ l に一致する．すなわち，$(\sum_{i=1}^{k}|E_i|)/k = l$ となる．

提案手法のアルゴリズムは，以下のとおりである（**図 2** に疑似コードを示す）．まず，適当に定められた $k_0\,(>k)$ 個の seed $s_1,\ldots,s_{k_0}$ をデータ集合 D から任意に選ぶ．次に，すべてのデータを分類してクラスタ $C_1,\ldots,C_{k_0}$ を作成する．C_i には，最も近い seed が s_i であるデータを収める．また，それぞれのクラスタ C_i に対応したクラスタリングベクトル集合 E_i を作成する．E_i の初期値には，元の座標系の d 個の軸をそのまま収める．

以下，繰返しにより，クラスタ数とそれぞれのクラスタの次元数を段階的に減らしていく．1 回の繰返しの中で次に求めるクラスタ数と次元数は，定数 $\alpha\,(<1)$ および $\beta\,(<1)$ によって定まる．すなわち，現在のクラスタ数を k_c，次元数の平均を l_c とするとき，次のクラスタ数と次元数の平均はそれぞれ $\alpha\cdot k_c$ および $\beta\cdot l_c$ となる．本稿では，α を**クラスタ数の絞り込み定数**，β を**次元数の絞り込み定数**という．

繰返しの中で行う処理は以下のとおりである．

- まず，それぞれのクラスタ $C_1,\ldots,C_{k_c}$ について分散共分散行列を作り，対角化して固有値と固有ベクトルを求める．
- つづいて，濃度の平均が $\beta\cdot l_c$ になるようにクラスタ $C_1,\ldots,C_{k_c}$ に対するクラスタリングベクトル集合 $E_1,\ldots,E_{k_c}$ を決定する．
- k_c 個のクラスタを $\alpha\cdot k_c$ 個になるまでマージする．
- すべてのデータを最も近い seed を持つクラスタに再分配し，新しいクラスタ $C_1,\cdots,C_{\alpha\cdot k_c}$ を作る．

クラスタリングベクトル集合 $E_1,\ldots,E_{k_c}$ の決定方法，クラスタのマージの方法，および，データの再分配の方法の詳細については後述する．

procedure Clustering(k, l)
  {Initialization}
  $\{s_1, \ldots, s_{k_0}\} \leftarrow$ pick $k_0 (> k)$ points from the data set
  $k_c \leftarrow k_0$; $l_c \leftarrow d$
  for all $i \in \{1, \ldots, k_c\}$ **do**
    $E_i \leftarrow$ original axis system
  end for
  $(s_1, \ldots, s_{k_c}, C_1, \ldots, C_{k_c}) \leftarrow Assign(s_1, \ldots, s_{k_c}, E_1, \ldots, E_{k_c})$

  {Iteration}
  while $k_c > k$ **do**
    $k_{new} \leftarrow max(k, \alpha \cdot k_c)$; $l_{new} \leftarrow max(l, \beta \cdot l_c)$
    for all $i \in \{1, \ldots, k_c\}$ **do**
      $E_i \leftarrow FindVector(C_i)$
    end for
    $(E_1, \ldots, E_{k_c}) \leftarrow DetermineVector(E_1, \ldots, E_{k_c}, l_{new})$
    $(s_1, \ldots, s_{k_{new}}, C_1, \ldots, C_{k_{new}}, E_1, \ldots, E_{k_c}) \leftarrow Merge(C_1, \ldots, C_{k_c}, k_{new}, l_{new})$
    $(s_1, \ldots, s_{k_{new}}, C_1, \ldots, C_{k_{new}}) \leftarrow Assign(s_1, \ldots, s_{k_{new}}, E_1, \ldots, E_{k_{new}})$
    $k_c \leftarrow k_{new}$; $l_c \leftarrow l_{new}$
  end while
end procedure

図 2 クラスタリングアルゴリズム
Fig. 2 The clustering algorithm.

以上の処理を，クラスタ数が k，クラスタリングベクトル数の平均が l になるまで繰り返し行う．文献 11) で述べられているとおり，クラスタ数およびクラスタリングベクトル数の平均がそれぞれ k_0 および $l_0 (= d)$ から始まり，順次それぞれ α と β を乗じながら同じ繰返し回数で k と l となるためには，α と β の間に

$$\log_{\frac{1}{\alpha}} \left(\frac{k_0}{k}\right) = \log_{\frac{1}{\beta}} \left(\frac{l_0}{l}\right) \qquad (2)$$

という関係が成り立っている必要がある．

一般に，α の値が小さいほど，クラスタ数が最終的な値 k に早く収束することになるので処理時間は短縮されるが，早い段階で多くのクラスタがマージされることになるため適切なクラスタが発見できず，精度が落ちるおそれがある．α の値と結果の精度の関係については，シミュレーションを行った結果を次章に示す．

3.4 次元数のばらつきへの対応

図 2 の疑似コードでは，以下の 4 つの関数が使われている．クラスタごとの次元数のばらつきについての対応を行っているのは，これらの関数である．

- *Assign*
- *FindVector*
- *DetermineVector*
- *Merge*

Assign は，クラスタの seed $s_1, \ldots, s_{k_c}$ およびそれぞれのクラスタの部分空間を定義するクラスタリングベクトル集合 $E_1, \ldots, E_{k_c}$ を引数として受け取り，データ集合中のすべてのデータを分割してクラスタとして返す関数である．それぞれのデータがどのクラスタに所属するかは，その点と距離が最も近い seed がどれかによって決定する．

通常の場合，この seed と各点との距離にはユークリッド距離を用いるのが一般的であるが，射影クラスタリングでは部分空間中で互いの距離が近い点を集めてクラスタとするので，一般の場合と同様の方法は採用できない．ORCLUS では，それぞれのクラスタの部分空間での射影距離 *Pdist* を用いて seed との間の距離を計算し，その値が最も小さい seed のクラスタに分類するという手法を採用している．しかし，本稿の提案手法では，クラスタによって部分空間の次元数が異なるので，ORCLUS のように単純に射影距離を用いて各 seed までの距離を比較することができない．このため，本稿の手法では，*Pdist* の代わりに以下のように定義される距離 *NPdist* を用いる．

```
function Assign(s_1, ..., s_{k_c}, E_1, ..., E_{k_c})
  for all i ∈ {1, ..., k_c} do
    C_i ← ∅
  end for
  for all data point p in the data set do
    i ← find i ∈ {1, ..., k_c} such that NPdist(p, s_i, E_i) is the minimum
    C_i ← C_i ∪ {p}
  end for
  return (cent(C_1), ..., cent(C_{k_c}), C_1, ..., C_{k_c})
end function
```

図 3 クラスタへのデータの分配
Fig. 3 Distributing data points.

```
function DetermineVector(E_1, ..., E_{k_c}, l_new)
  for all i ∈ {1, ..., k_c} do
    {v_1, v_2} ← find two eigenvectors in E_i such that corresponding eigenvalues are minimum
    E'_i ← {v_1, v_2}
  end for
  l_determined ← 2k_c
  while l_determined < k_c · l_new do
    for all i ∈ {1, ..., k_c} do
      v_i ← find eigenvector v_i ∈ E_i − E'_i such that corresponding eigenvalue is the minimum
      λ_i ← corresponding eigenvalue of v_i
    end for
    i ← find i ∈ {1, ..., k_c} such that λ_i is the minimum in {λ_1, ..., λ_{k_c}}
    E'_i ← E'_i ∪ {v_i}; l_determined ← l_determined + 1
  end while
  return (E'_1, ..., E'_{k_c})
end function
```

図 4 クラスタの部分空間の決定
Fig. 4 Determination of clusters' subspaces.

$$NPdist(\boldsymbol{x_1}, \boldsymbol{x_2}, E) = \frac{Pdist(\boldsymbol{x_1}, \boldsymbol{x_2}, E)}{\sqrt{|E|}} \qquad (3)$$

これは，射影距離を部分空間 E の次元数によって正規化したものである．$\sqrt{|E|}$ で除するのは，d 次元の単位空間における最も離れた 2 点（対角頂点）間の距離が $\sqrt{d}$ であることによる．

すべてのデータについてクラスタへの分配が終わったら，それぞれのクラスタについて重心を計算し，それをそのクラスタの新しい seed とする．関数 *Assign* は，返り値としてこの seed とクラスタを返す．**図 3** に *Assign* の疑似コードを示す．

関数 *FindVector* は，1 つのクラスタを引数として受け取り，そのクラスタに属するデータ集合の分散共分散行列を作って対角化を行い，結果としてそのクラスタの固有ベクトルの集合を返す．*FindVector* の疑似コードは省略する．

DetermineVector は，濃度の平均が l_{new} になるようにクラスタ $C_1, \ldots, C_{k_c}$ に対するクラスタリングベクトル集合 $E_1, \ldots, E_{k_c}$ を決定する関数である（**図 4**）．

それぞれのクラスタリングベクトル集合は，以下のようにして決定する．

- まず，それぞれのクラスタ C_i について，固有値が最も小さい固有ベクトルと，その次に固有値が小さい固有ベクトルの 2 つを選び，クラスタリングベクトル集合 E'_i に収める．
- 以降，全体の合計が l_{new} 個になるまで，すべてのクラスタの残りの固有値の中から最も小さい固有値を順に選び，それに対応する固有ベクトルを

```
function Merge(C_1, ..., C_{k_c}, k_new, l_new)
  while k_c > k_new do
    for each pair i, j ∈ {1, ..., k_c} satisfying i < j do
      E'_ij ← FindVector(C_i ∪ C_j)
    end for
    (i, j) ← find a pair i, j ∈ {1, ..., k_c} such that i < j and NR(C_i ∪ C_j, E'_ij) is the best
    (C_1, ..., C_{k_c−1}) ← (C_1, ..., C_{i−1}, C_{i+1} ..., C_{j−1}, C_{j+1}, ..., C_{k_c}, C_i ∪ C_j)
    (E_1, ..., E_{k_c−1}) ← (E_1, ..., E_{i−1}, E_{i+1} ..., E_{j−1}, E_{j+1}, ..., E_{k_c}, E'_ij)
    (E_1, ..., E_{k_c−1}) ← DetermineVector(E_1, ..., E_{k_c−1}, l_new)
    k_c ← k_c − 1
  end while
  return (cent(C_1), ..., cent(C_{k_new}), C_1, ..., C_{k_new}, E_1, ..., E_{k_c})
end function
```

図 5 クラスタのマージ
Fig. 5 Merging clusters.

E_i に収める．

以上のようにしてできあがったクラスタリングベクトル集合の列 $E'_1, \ldots, E'_{k_c}$ が，関数 *DetermineVector* の結果となる．

クラスタのマージは，関数 *Merge* によって行う．この関数は，k_c 個のクラスタを引数として受け取り，それを k_{new} になるまで順次マージする．

マージするクラスタの選択は，以下のようにして行う．まず，すべてのクラスタの組について，それをマージしたときの評価値を計算する☆．すべての組合せの中で最も評価値の良いものを選び，それらを実際にマージする．このようにしてクラスタの数を 1 つずつ減らしていき，k_{new} 個になったら終了する．

クラスタの評価は，ORCLUS と同様，射影エネルギーを用いて行う．ただし，先にも述べたように本稿の手法ではそれぞれのクラスタの次元数が異なるので，射影エネルギーの計算にも正規化した射影距離 *NPdist* を用いる．すなわち，本稿の提案手法で用いる射影エネルギーは，以下の式によって定義される．

$$NR(C, E) = \frac{\sum_{i=1}^{n} \{NPdist(\boldsymbol{x_i}, cent(C), E)\}^2}{n}$$

関数 *Merge* の疑似コードを**図 5** に示す．このコードでは，$NR(C, E)$ の値が最小になるようなクラスタの組 C_i，C_j を見つけ出してマージするという処理を，クラスタ数が k_{new} 個になるまで繰り返している．繰返しの各回の最後のところで，マージ後のクラスタ集合の次元数の平均が l_{new} 個になるように関数 *DetermineVector* を呼び出している．このようにしてマージを繰り返し，クラスタ数が k_{new} 個になったら，それぞれのクラスタの重心を計算し直し，最終的な結果を返す．

☆ ORCLUS では，クラスタをマージしたときの評価値を効率良く求めるために ECF-vector と呼ばれるデータ構造を利用しているが，これは本稿で提案する手法においても同様に利用することができる．

4. シミュレーション

本稿の提案手法の有効性を検証するため，合成データによるシミュレーションを行い，ORCLUS との比較を行った．ここでは，その結果を示す．

4.1 合成データの作成

シミュレーションに用いる合成データは，文献 11) で用いられた方法と同様の方法によって作成した．その概略は以下のとおりである．

まず，各クラスタの部分空間を決定するため，任意の方向のベクトルを以下の方法によって作成する．はじめに，$(-1, 1)$ の範囲の値を持つ $d \times d$ 対称行列を乱数によって作成する．この行列を固有値分解すると d 個の正規直交固有ベクトルが得られるので，その中から適当な個数のベクトルを選んでそのクラスタのクラスタリングベクトル集合とする．このベクトル集合は，個々のクラスタについてそれぞれ異なる $d \times d$ 対称行列を用いて生成する．それぞれのクラスタのクラスタリングベクトルの個数は，平均 l，分散 v となるように乱数によって決定する．

次に，それぞれのクラスタに含まれる点の座標値を決定する．座標値は，それぞれのクラスタについて決定されたクラスタリングベクトルの方向に対しては，そのクラスタの seed を中心に正規分布によって決定する．また，それ以外の方向に対しては，一様分布に

よって決定する．それぞれのクラスタの seed は，一様分布によって決定する．

以上の処理を生成するクラスタの数だけ繰り返し実行し，得られたデータをまとめて 1 つのデータセットとした．

データ生成およびシミュレーションで用いた各種パラメータとそのデフォルト値を**表 1** に示す．表にデフォルト値が示されていない β の値は，式 (2) に示した式によって求まる値を用いた．

4.2 ORCLUS との比較

ここでは，上に述べた方法により合成したデータに対して ORCLUS および本稿の提案手法でクラスタリングを行った結果を示す．

図 6 および**図 7** は，クラスタリングベクトル数の分散の値が異なる 2 つの合成データセットについて，それぞれの手法によるクラスタリングを行った結果を示した混乱行列（confusion matrix）である．この行列は，上述の方法で生成した 5 つのクラスタ（I1〜I5）に属するそれぞれのデータが，ORCLUS および提案手法によるクラスタリングを行った結果発見されたクラスタ（O1〜O5）のどこに含まれているかを数え上げたものである．したがって，この行列の非対角成分の値が小さいほどクラスタリングの精度が高いことを意味することになる．

図 6 および図 7 を見ると，ORCLUS よりも本稿の提案手法の方が非対角成分の値が小さくなっている．また，ORCLUS ではうまく発見できていないクラスタも，提案手法では発見できている．このことより，提案手法の方がより高い精度でクラスタを発見できているといえる．この差は，クラスタリングベクトル数の分散が大きい図 7 でより顕著となっている．

それぞれのアルゴリズムで用いられている手法を考慮すれば，クラスタリングベクトル数の分散が大きいほど ORCLUS と比較した場合の本稿の提案手法の優位性が増すと考えられる．この点について調べた結果が，**図 8** および**図 9** である．これは，クラスタリングベクトル数の平均の値が異なる 2 つのデータセットについて，クラスタリングベクトル数の分散を 1〜5 まで変化させたときの ORCLUS および本稿の提案手法による処理結果である．縦軸は，それぞれの手法による結果出力されたクラスタの射影エネルギーの総和である．この値が小さいほど，より精度良くクラスタ

表 1 シミュレーションで用いたパラメータ
Table 1 Parameters used in the simulations.

記号	説明	値
d	データ空間の次元数	20
N	データ数	5,000
k	クラスタ数	5
k_0	クラスタリング開始時の初期クラスタ数	50
l	クラスタリングベクトル数の平均	6
v	クラスタリングベクトル数の分散	3
α	クラスタ数の絞り込み定数	0.7
β	次元数の絞り込み定数	—

	I1	I2	I3	I4	I5
O1	667	0	2	0	0
O2	0	1027	0	542	0
O3	1	0	941	0	0
O4	0	0	426	0	0
O5	0	0	0	1	1393

(a) ORCLUS

	I1	I2	I3	I4	I5
O1	665	0	2	0	0
O2	1	1027	2	1	0
O3	2	0	1365	0	0
O4	0	0	0	541	0
O5	0	0	0	1	1393

(b) 提案手法

図 6 混乱行列（$v = 3$）
Fig. 6 Confusion matrix ($v = 3$).

	I1	I2	I3	I4	I5
O1	1054	0	0	804	0
O2	0	1399	5	0	1149
O3	0	0	238	0	0
O4	0	0	115	0	0
O5	0	0	236	0	0

(a) ORCLUS

	I1	I2	I3	I4	I5
O1	1054	0	0	0	37
O2	0	1354	0	0	0
O3	0	0	593	0	0
O4	0	2	0	804	29
O5	0	43	1	0	1083

(b) 提案手法

図 7 混乱行列（$v = 5$）
Fig. 7 Confusion matrix ($v = 5$).

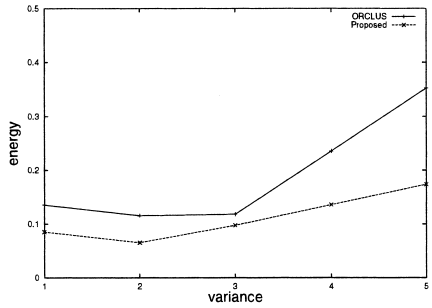

図 8 分散の値と精度の関係（$l = 6$）
Fig. 8 Effects of variance on precision ($l = 6$).

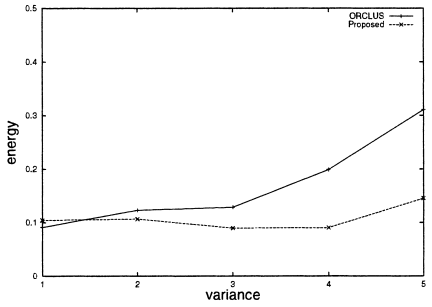

図 9 分散の値と精度の関係（$l = 8$）
Fig. 9 Effects of variance on precision ($l = 8$).

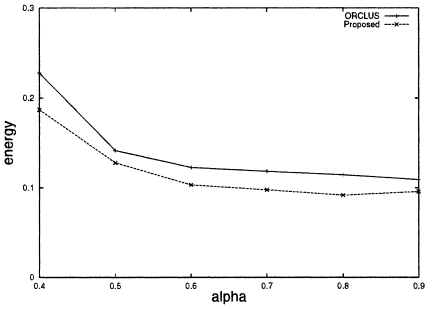

図 10 α の値と精度の関係
Fig. 10 Effects of α on precision.

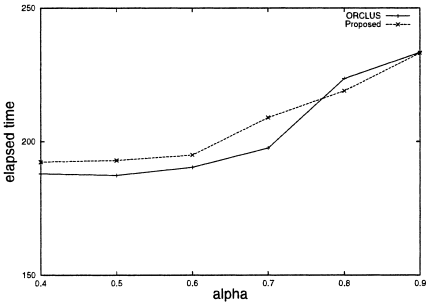

図 11 α の値と処理時間の関係
Fig. 11 Effects of α on elapsed time.

が発見できていることを意味する．図には，それぞれの分散値についてデータセットを 5 セット用意し，得られた結果の射影エネルギーの総和の平均を示した．

これらの図が示すように，どちらのデータセットについても，クラスタリングベクトル数の分散値が大きくなるにつれて ORCLUS と提案手法の精度の差が大きくなるという結果となった．また，すべての分散値について，ORCLUS より提案手法の方が安定的に精度の良い結果が得られた．

以上のシミュレーション結果により，ORCLUS では良い結果が得られないクラスタごとに部分空間の次元数が大きく異なるようなデータセットに対するクラスタリングの手法として，本稿の提案手法が有効であることが検証された．

4.3 α の値と精度の関係

ORCLUS および本稿の提案手法で用いているクラスタ数の絞り込み定数 $\alpha\,(< 1)$ は，クラスタを発見する 1 回の繰返しごとに，どれだけクラスタをマージして数を減らすかを定める係数である．当然のことながら，α を小さくするほど早く目的のクラスタ数に達することになるので処理時間は短くなるが，その分精度が下がる可能性がある．

この点について調べた結果が，**図 10** である．この図は，α の値を変化させたときに発見されたクラスタの射影エネルギーの総計がどのように変化するかを示したものである．この図より，α の値が 0.5 を下回る（すなわち，クラスタの数を繰返しのたびに半分未満にする）と精度が急激に悪くなることが分かる．また，α の値が 0.6 を超えたあたりからは，安定した結果が得られている．

図 11 は，処理時間について調べた結果である．α が大きくなるにつれ，処理時間が増す傾向にあることが分かる．

5. 結　　論

本稿では，次元のばらつきに対応した一般射影クラスタリングの手法を提案した．この手法では，ORCLUS と同様の方法で部分空間の次元数を段階的に絞り込む際に，それぞれのクラスタの分散共分散行列を固有値分解して得られる固有値を利用してそれぞれのクラスタの次元数を定めるという手法を採っている．これにより，より一般的な高次元データ集合に対して適切にクラスタを発見することが可能としている．シミュレーションによる結果は，次元数が異なるクラスタを持つデータ集合に対して，本稿の提案手法が ORCLUS よりも優れていることを示している．

今後の課題としては，さまざまなパラメータ設定のシミュレーションにより提案手法の性質をより詳細に

調査するとともに，索引機構などの各種の高次元データに対する処理への本手法の適用を検討している．

参 考 文 献

1) Jain, A. and Dubes, R.: *Algorithms for Clustering Data*, Prentice Hall (1998).
2) Ng, R.T. and Han, J.: Efficient and Effective Clustering Methods for Spatial Data Mining, *Proc. 20th International Conference on Very Large Data Bases*, pp.144–155 (1994).
3) Zhang, T., Ramakrishnan, R. and Livny, M.: BIRCH: An Efficient Data Clustering Method for Very Large Databases, *Proc. 1996 ACM SIGMOD International Conference on Management of Data*, pp.103–114 (1996).
4) Berchtold, S., Böhm, C., Keim, D.A. and Kriegel, H.-P.: A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space, *Proc. 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp.78–86 (1997).
5) Beyer, K.S., Goldstein, J., Ramakrishnan, R. and Shaft, U.: When Is "Nearest Neighbor" Meaningful?, *Proc. ICDT '99, 7th International Conference*, Lecture Notes in Computer Science, Vol.1540, pp.217–235, Springer-Verlag (1999).
6) Böhm, C.: A Cost Model for Query Processing in High Dimensional Data Spaces, *ACM Trans. Database Syst.*, Vol.25, No.2, pp.129–178 (2000).
7) Aggarwal, C.C., Hinneburg, A. and Keim, D.A.: On the Surprising Behavior of Distance Metrics in High Dimensional Spaces, *Proc. ICDT 2001, 8th International Conference*, Lecture Notes in Computer Science, Vol.1973, pp.420–434, Springer-Verlag (2001).
8) Indyk, P. and Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality, *Proc. ACM Symposium on Theory of Computing*, pp.604–613 (1998).
9) Hinneburg, A. and Keim, D.A.: Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering, *Proc. 25th International Conference on Very Large Data Bases*, pp.506–517 (1999).
10) Aggarwal, C.C., Procopiuc, C., Wolf, J.L., Yu, P.S. and Park, J.S.: Fast Algorithms for Projected Clustering, *Proc. 1999 ACM SIGMOD International Conference on Management of Data*, pp.61–72 (1999).
11) Aggarwal, C.C. and Yu, P.S.: Finding Generalized Projected Clusters In High Dimensional Spaces, *Proc. 2000 ACM SIGMOD International Conference on Management of Data*, pp.70–81 (2000).
12) 黒木　薫，古瀬一隆，大保信夫：次元の段階的な絞り込みによる射影クラスタリングの機構と評価，情報処理学会研究報告，2000-DBS-123, pp.15–22 (2001).

(平成 13 年 9 月 20 日受付)
(平成 13 年 11 月 13 日採録)

(担当編集委員　牧之内 顕文)

古瀬　一隆（正会員）

1993 年筑波大学大学院工学研究科修了.（株）リコーソフトウェア研究所勤務，茨城大学工学部情報工学科助手を経て，1999 年筑波大学電子・情報工学系助手．博士（工学）.

石川　雅弘（正会員）

2001 年筑波大学大学院工学研究科修了．同年農業生物資源研究所研究員．博士（工学）.

陳　　漢雄（正会員）

1993 年筑波大学大学院工学研究科修了．同年同大学電子・情報工学系助手．1994 年つくば国際大学産業情報学科講師．2001 年筑波大学電子・情報工学系講師．博士（工学）.

大保　信夫（正会員）

1968 年東京大学大学院修士課程修了．同年同大学理学部助手．1980 年筑波大学電子・情報工学系講師．1995 年同大学電子・情報工学系教授．理学博士.