k-means の結果を初期値に用いた NMF による文書クラスタリング

執筆者：直江 宗紀

指導教官：新納 浩幸

平成１９年３月１日

目次

図目次

表目次

第1章　はじめに

1.1　概要

文書クラスタリングとは，記述された内容が似ている文書をまとめていき，分類させることをいう[1].

文書は，文章の集まりである．しかし，クラスタリングは，数値データを元にして分類を行うため，文章のままでは，文書間の類似性を知ることはできない．よって，文書のベクトル化をしなければならない(以下，このベクトル化されたベクトルを文書ベクトルと呼ぶ)．そこで，索引語文書行列と呼ばれる行列を用いる．この行列は，特定の文書内の単語の頻度数を要素とした行列であり，文書と単語頻度の関係となるベクトル行列となるためである．しかし，その行列のままでは，特定の文書の単語数が多いほど，その文書の特徴があるとしているため，ほかの文書でも同様に同じ単語数が多い場合には，特定の文書を特徴付けてるとはいえなくなり，他の文書でも出現する単語に対して，なんらかの方法によって，特徴付けをさせなくしなければならない．よって，文書ベクトルに対して，tf-idf と呼ばれる計算方法を取り入れ，他の文書でも同様に同じ単語が多い場合の対処を行う．さらに，ベクトルの長さをそろえるため正規化を行う[2].

これまで，様々なデータ抽出手法や解析手法を用いたクラスタリングの研究が行われてきた．そのクラスタリング研究の過程において，近年 NMF(non-Negative Matrix Factorization)を用いた手法が考案された．この手法の特徴としては，データとして与えられた実数値で構成された行列から，効率よく次元削減を行うためのアルゴリズムである[3].

しかし，NMF は，初期値によって結果がばらついてしまう．よって，既存手法から得られるクラスタリング結果から，分類の評価をあらかじめ初期値として設定すれば，正解率が安定すると考えられる．また，本研究では，この手法を試す．

NMF と本手法との結果の違いや問題点を考察する．NMF の初期値に利用するための既存手法として，k-means[4]を用いる．

本実験では，ネットニュースの文書群，及び，英語の文書群から作られた評価用データセット2種類の，計3種類を用意する．ネットニュースは元々の文書群のカテゴリであった国際関係，政治関係，社会関係，スポーツ関係，経済関係の5クラス，残りの評価用データセット2つは，それぞれ8クラスのカテゴリ，10クラスのカテゴリに分類される．3種類の文書群から，それぞれ索引語文書行列を作成し，また，作成された行列を tf-idf 計算及び，正規化する．それらのデータを，k-means, NMF，CLUTO，及び，本手法の4種類のクラスタリングを行う．CLUTO とは，ミ

ネソタ大学の Gcorge Karypis 氏の開発したクラスタリングツールアプリケーションソフトである．それぞれのクラスタリング結果から正解率を比較し，本手法の有効性について，検討する．

1.2 本論文の構成

第２章において，基本的な文書クラスタリングについて述べる．ここでは，文書と単語のベクトル空間モデルとして用いる，索引語文書行列の作成について説明する．また，この行列の要素として，各文書の単語頻度だけを重みとするだけでは，不都合が生じる理由，tf-idf を採用する理由を説明する．さらに，既存のクラスタリング手法としてどのような物が存在するのか，基本的手法について説明する．

第３章において，本研究の核となる，NMF についての概要，及び，アルゴリズムについて説明をする．

第４章では，本研究において，新しい手法の提案と，その概要について説明する．

第2章　文書クラスタリング

2.1 索引語文書行列

文書をクラスタリングするためには，文書の内容をなんらかの形で表して比較しなければならない．その比較を行うために文書をベクトルとして表す必要がある[3].

そこで，文書のベクトル化を考える．このとき，文書から，各単語の頻度をベクトル要素とする．そのためには，まず文書を単語などに分解する必要がある．ここで，形態素解析と呼ばれる，文章を単語や助詞などに分解するプログラムを利用する．この形態素解析により，文書から使われている名詞句を抽出できる．抽出された全名詞句から，同じ名詞句をまとめ，頻度をカウントしていく．そうして，カウントされた単語の頻度を要素とした，ベクトルを作成する．

ここで，N 個の文書があるとする．さらに，N 個の文書のうち，j 番目の文書をベクトル化した X_j を考える．また，全名詞数が M 個であったとき，ベクトル Xj を次のように表す．

$$X_j = \begin{bmatrix} x_{1j} \\ \vdots \\ x_{ij} \\ \vdots \\ x_{Mj} \end{bmatrix} \tag{2.1}$$

この時，x_{ij} は j 番目の文書に含まれる i 番目の単語の頻度を示している．各文書のベクトルを持つ索引語文書行列は，次のように表すことができる．

$$X = [X_1, X_2, \cdots, X_j, \cdots, X_N] \tag{2.2}$$

式(2) を展開すると，次のようになる．

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1N} \\ \vdots & \ddots & \vdots & & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{iN} \\ \vdots & & \vdots & \ddots & \vdots \\ x_{M1} & \cdots & x_{Mj} & \cdots & x_{MN} \end{bmatrix} \tag{2.3}$$

式(2.3) に示される索引語文書行列 X がクラスタリング可能なデータとなる．

2.2 TF-IDF と正規化

2.1 の式(3)で示すような，行列の要素は，各文書の単語頻度を要素としている．

この単語頻度を tf(term frequency)と呼び，これは，特定の文章における頻度が高ければ，その文書における特徴付けとして，大きな意味となるという，重み付けベクトルである．しかし，他に出現する文書の数が多い場合，その単語が特定の文書の特徴を表すとは言い難くなる．そこで，idf(inverse document frequency)というものを考える．これは，その単語が出現する文書数の少ないほど，その単語が特定の文書における特徴を付けていると考えるものである[2]．

その２つの積を値として用いることにより，tf，idf の両面から見たベクトル要素を得ることが可能となる．

重み付けを行う tf 及び idf は，それぞれ，さまざまな重み評価の計算方法があり，目的にあった計算方法を選択する必要がある．

そこで，以下の式に示す tf 及び idf の評価計算を採用した．

$$tf = \log(tfw + 1)$$
$$idf = \log(\frac{N}{df} + 1) \tag{2.4}$$

ここで tfw は文書内の単語頻度数(Term Frequency of Word)，N は全文書数，df は文書頻度数(Document Frequency)を示す．文書頻度数とは，全文書のうち単語が出現した文書数を指す．

このとき，tfw 及び N/df に，それぞれ 1 を加えているのは，tfw は頻度数が 0 の場合に log(0)=1 になることを防ぎ，N/df は，文書頻度数が，全文書と同数の場合に log(1)=0 となることを防ぐためである．

しかし，この時，各文書ベクトルの長さが一定であるとはいえないため，各文書ベクトルの長さを 1 にそろえる必要がある．

よって，式(3)の行列 X の各要素は，以下に示す式で決定することにする．

$$x_{ij} = \frac{x_{ij}'}{\sqrt{\sum_{k=1}^{M}(x_{ik}')^2}}$$
$$x_{ij}' = tf \times idf = \log(tfw_{ij} + 1) \times \log(\frac{N}{df_i} + 1) \tag{2.5}$$

ここで，x_{ij}'は，j 番目の文書の i 番目の単語についての tf-idf を示す．また，tfw_{ij} は j 番目の文書の i 番目の単語についての単語頻度数，df_i は i 番目の単語についての文書頻度数を示す．

2.3 クラスタリング手法

クラスタリングとは，さまざまな対象が混ざり合っている集合体の中から，互いに似たものを集めて集合を作り，対称を分類しようという方法を総称したものである．

クラスタリング手法には，大きく，階層的手法(hierarchical)と分割最適化手法(partitioning-optimization)とに分けることが出来る[1].

ここでは，基本的なクラスタリング手法について紹介する．

まず，階層的手法は，さらに分枝型(divisive)と凝集型(agglomerative)に分けられるが，ここでは後者のみについて紹介する．

凝集型は，まず，全部でN個のデータが与えられた時，1個の対象だけを含むクラスタがN個あるとする(データ1個が1クラスタ)．次に，対象 x_1 と x_2 の間の距離 $D(x_1,x_2)$ (非類似度)からクラスタ間の距離 $D(C_1,C_2)$を計算し，最もこの距離の近い二つのクラスタを逐次的に併合する．そして，この併合を，全ての対象が一つのクラスタに併合されるまで繰り返すことで階層構造を獲得する．また，クラスタ C_1 と C_2 の距離関数 $D(C_1,C_2)$の違いにより様々な手法が存在する．距離関数から得られた階層構造から，樹形図あるいはデンドログラム(dendrogram)と呼ばれる樹状の分類構造によって表示される．凝集型の代表的な手法として，最短距離法，最長距離法，郡平均法，さらにWard法とがある．それぞれの距離関数を示す．

・最短距離法

最短距離法は，対象と対象の単純距離を調べ，お互いが一番近い対象から順にクラスタとして形成していき，階層構造を得る手法である．距離関数は，以下に示す式によって表される．

$$D(C_1,C_2)=\min_{x_1\in C_1,x_2\in C_2} D(x_1,x_2) \tag{2.6}$$

・最長距離法

最長距離法は，最短距離法とは逆に，対象間の距離が一番長い物から順に，分割していく方法である．つまり，距離が一番長い物は，同一クラスタではないと見なし，分割しながら，階層構造を得る手法である．距離関数は，以下に示す式によって表される．

$$D(C_1,C_2)=\max_{x_1\in C_1,x_2\in C_2} D(x_1,x_2) \tag{2.7}$$

・群平均法

郡平均法は，最短距離法や最長距離法のように，極端に最大，最小という値に基づいて比較を行っている．しかし，この手法は，クラスタ間の距離を平均的な値で定義する考え方に基づいた方法の一つである．具体的には，対象となる 2 つのクラスタがあるとすると，お互いが持っている対象全ての対の距離の平均により，両クラスタ間の距離を比較する方法である．距離関数は，以下に示す式によって表される．

$$D(C_1,C_2)=\frac{1}{n_1 n_2}\sum_{x_1\in C_1}\sum_{x_2\in C_2}D(x_1,x_2) \tag{2.8}$$

ここで，n_1,n_2 は，それぞれ，クラスタ C_1,C_2 を構成する対象の数を示す．

・ Ward 法

Ward 法は，各対象からその対象を含むクラスタのセントロイドまでのユークリッド距離である． Ward 法は，この距離関数を最小化する．距離関数は以下に示す式によって表される．

$$\begin{aligned} &D(C_1,C_2)=E(C_1\cup C_2)-E(C_1)-(C_2)\\ &(\text{ただし } E(C_i)=\textstyle\sum_{x\in C_i}\left(D(x,c_i)\right)^2) \end{aligned} \tag{2.9}$$

ここで，c_i はクラスタのセントロイド，x はクラスタ C_i に含まれるデータを示す．

・ k-means

分割最適化手法についてであるが，分割最適化手法とは，分割の良さの評価関数を定め，その評価関数を最適にする分割を探索する手法である．可能な分割の総数は全データ数 N に対して指数的なので，実際は局所最適解を求めることになる． k-means は，その分割最適化手法の中でも代表的な手法である．

k-means は，クラスタの重心点 ci をクラスタの代表点とし，各データとの距離を以下の式で評価する．

$$\sum_{i=1}^{k}\sum_{x\in C_i}\left(D(x,c_i)\right)^2 \tag{2.10}$$

式(2.10) に示す，評価関数を最小化するように，k 個のクラスタを分割する．最適解の探索は下図にしめすように，対象のクラスタへの割り当てと代表点の再計算を交互に繰り返して行う．この手法は山登り法で，局所最適解しか求められないため，ランダムに初期値を変更して，評価関数を最小にする結果を選択するのが一般的である．

1. k 個の代表点 $c_1,\cdots,c_k$ をランダム選択
2. X の全ての対象 x を $c^* = \arg\min_{c_i} D(x,c_i)$ なる代表点に割り当て
3. もし代表点への割り当てが変化しないならば終了，
   そうでなければ各クラスタのセントロイドを代表点にして 2.に戻る

図 2.1　k-means 法アルゴリズム

・ユークリッド距離

ユークリッド距離とは，ベクトル空間に表現されたベクトルの大きさまたは距離の概念のことである．クラスタリングにおいて用いられるユークリッド距離は，データ個体間の非類似度を表す量として用いられる．ここで，式(2.3)に示されるような，N 個体のM個の変量があったとすると，個体 i と個体 j との非類似度 d_{ij} とすると，以下のような式に表される[4]．

$$d_{ij} = \sum_{k=1}^{m} (x_{ki} - x_{kj})^2 \qquad (2.11)$$

・クラスタリングの問題点

文書クラスタリングなどの，高次元空間の対照を扱う場合，その高次元に起因した問題として，「次元の呪い」という物がある．これは，次元の増加により，対象間の距離が急速に大きくなるため，距離関数で正確に求めることが出来なくなるためである．元々，距離関数が小さい物から扱うクラスタリングにとっては，有意な解が得られなくなるという問題である．

この次元の呪いに対する抜本的な解決法としては，外的知識によって不要と考えられる属性を排除し，次元数を小さくすることである．

また，手法固有の問題点もある．階層的手法では，数学的に優れたある種の性質を持つ最短距離法は，空間収縮という性質のため，外乱にきわめて弱く，実データではあまり良い結果が得られないことが知られている．空間収縮とは，併合されて出来た新しいクラスタは，以後の併合の対象として選ばれる可能性が加速度的に増加する現象である．これは，外乱となっているデータが，クラスタ形成を大きな物にしていってしまうため，デンドログラムなどで階段状構造となってしまう場合がある．これをチェイニング効果といい，最短距離法の特性として生じる構造であることが多い．

また，k-means 法にも固有の問題点も存在する．先ほども示したように，k-means は欲張り探索により，局所解を求める手法であるため，初期状態によって最終結果は大きく影響されてしまう．乱数により，初期状態を決定しているのは，複数回実行し，いくつかの分割を獲得してから，それらの分割の中で評価関数を最小にする

ものを選ぶためである．また，同じクラスタにならないことが事前に予測される対象があれば，初期クラスタでそれらの対象を別のクラスタの代表点とする方法もある．

・ CLUTO

上記の既存のクラスタリング手法とは他の最近のクラスタリングとして，現在公開されている，クラスタリングツールに CLUTO と呼ばれるアプリケーションがある．ミネソタ大学情報工学科の George Karypis 氏が製作，公開をしているソフトである．このソフトの特徴として，優秀な正解率を求めることが出来るため，新手法による正解率を比較検討する指標としても度々用いられる．この CLUTO では，評価関数や手法を，ユーザーが指定することにより，さまざまな組み合わせでクラスタリングさせることが可能である．

CLUTO では，主に，k-way 解法を用いた手法で構成されている．また，距離評価として，ユークリッド距離のほかに，コサイン，囲いこみ係数， Jaccard 係数などがある．

第３章　NMF

3.1　概要

NMF(non-Negative Matrix Factrization)とは，非負値行列に対する行列分解アルゴリズムである．このアルゴリズムは，ベクトル空間として表した行列から，特徴抽出し，分解するのに優れており，自然言語処理分野だけにとどまらず，様々な分野において利用されている[3].

具体的な処理は，入力された行列 X を次に示すような行列 U 及び V の積に分離する．

$$X = U \cdot V^T \tag{3.1}$$

ここで，全文書数を N，全単語数を M とした時，行列 X は M×N の索引語文書行列，行列 U は M×k の索引語とクラスの関係を示す行列，また，行列 V は N×k の文書とクラスの関係を示す行列であり，文書クラスタリングの結果を示す行列でもある．K は，求めたいクラスタの数を示す．

NMF では，入力となる行列 X の各要素と，行列 U 及び V の積の値が限りなく同じになるように，計算していく必要がある．

そこで，行列 X 及び行列 U と行列 V の積とのユークリッド距離を以下のようにする[5].

$$E = \sum_{ij} \left(X_{ij} - (UV^T)_{ij} \right)^2 \tag{3.2}$$

ここで，Eはユークリッド距離関数を示す．

式(3.2)を展開させていく[1].

$$E = \sum_{ij} \left(X_{ij} - (UV^T)_{ij} \right)\left(X_{ij} - (UV^T)_{ij} \right) \tag{3.3}$$

$$E = \sum_{ij} \left((XX^T)_{ij} - 2(XU^TV)_{ij} - (UV^TVU^T)_{ij} \right) \tag{3.4}$$

ここで，Lagrangc 乗数法により，ユークリッド距離関数 E について，Lagrangc L を以下のように立てる．

$$L = E + \sum_{ij} (\alpha U)_{ij} + \sum_{ij} (\beta V)_{ij} \tag{3.5}$$

ここで，α, β は任意の定数とする．

未知変数である α, β，及び，未知行列要素数 U_{ij}, V_{ij} について，それぞれ偏微分を行った式を連立させる．

$$\begin{cases} \dfrac{\partial L}{\partial \alpha} = \sum_{ij} U_{ij} = 0 \\ \dfrac{\partial L}{\partial \beta} = \sum_{ij} V_{ij} = 0 \\ \dfrac{\partial L}{\partial U} = -2\sum_{ij}(XV)_{ij} + 2\sum_{ij}(UV^{T}V)_{ij} + \alpha = 0 \\ \dfrac{\partial L}{\partial V} = -2\sum_{ij}(X^{T}U)_{ij} + 2\sum_{ij}(VU^{T}U)_{ij} + \beta = 0 \end{cases} \tag{3.6}$$

この連立方程式に，Kuhn-Tucker 条件を当てはめる．ここで，連立方程式から，$\partial L/\partial \alpha = 0, \partial L/\partial \beta = 0$ から，$\alpha u_{ij}=0, \beta v_{ij}=0$ となる．よって，以下のような連立方程式が成り立つ．行列と行列要素との区別をつけるために，$U_{ij}=u_{ij}, V_{ij}=v_{ij}$ とする．

$$\begin{cases} (XV)_{ij}u_{ij} - (UV^{T}V)_{ij}u_{ij} = 0 \\ (X^{T}U)_{ij}v_{ij} - (VU^{T}U)_{ij}v_{ij} = 0 \end{cases} \tag{3.7}$$

よって，この連立方程式から，行列 U 及び V の各要素を更新する，更新式が求まる．

$$\begin{aligned} u_{ij} &\leftarrow u_{ij} \cdot \frac{(XV)_{ij}}{(UV^{T}V)_{ij}} \\ v_{ij} &\leftarrow v_{ij} \cdot \frac{(X^{T}U)_{ij}}{(VU^{T}U)_{ij}} \end{aligned} \tag{3.8}$$

また，要素を更新後，行列 U および行列 V を正規化する必要がある．よって，更新式によって更新した要素に対して，行列 U に基づく正規化を，以下に示す式によって行う．

$$\begin{aligned} u_{ij} &\leftarrow \frac{u_{ij}}{\sqrt{\sum_{k=1}^{M} u_{kj}}} \\ v_{ij} &\leftarrow v_{ij} \cdot \sqrt{\sum_{k=1}^{M} u_{kj}} \end{aligned} \tag{3.9}$$

この正規化によって，行列 U 及び V の各ベクトルの長さを 1 にする．

3.2 アルゴリズムの流れ

NMF のアルゴリズムは，以下の図の様に示される[1].

1. 入力となる行列 X の行数，列数及び，クラスタ数 k から，行列 U 及び行列 V を用意する.
   このとき，行列 U 及び行列 V の要素は，乱数により設定される.
2. 更新式である式(3.8)により，行列 U 及び V の各要素を更新する.
3. 式(3.9)により，行列 U 及び行列 V を正規化する.
4. 各要素の変量が限りなく 0 になるまで，2.に戻り，繰り返す.

図 3.1 NMF アルゴリズム

3.3 NMF の欠点

NMF は，式(3.2)で示されるような，ユークリッド距離関数が，最小になるように，行列 U 及び行列 V の要素を更新するとともに，両行列の整合をとるアルゴリズムである.

しかし，NMF は局所最適解しか求めることしか出来ないため，初期値によって，正解率にばらつきが出てくる.

第４章　クラスタリング結果を用いる NMF

第３章の 3.3 で述べたような NMF の欠点である，正解率のばらつきや精度を，改善する必要がある．

行列 V に対する初期値が乱数で設定されている場合，行列 V の各要素のベクトルの向きに統一性がないため，効率よくクラスタリングを行える可能性はある．しかし，完全な乱数とは言い難く，ベクトルの向きに偏りが生じてしまった場合に，NMF の正解率が下がるという可能性もあると考えられる．

そこで，他のクラスタリング手法によって得られた結果から，行列 V の要素を推定し，その値を初期値として，NMF クラスタリングを行わせる方法を提案する．

これは，他のクラスタリング手法から得られた評価のベースを，予め初期値として与えておくことにより，NMF の正解率が改善されるのではないかと推測したためである．

よって，新たな手法として，k-means によるクラスタリング結果を参考にした初期値を作成し，その初期値を用いて NMF クラスタリングを行うという方法を提案する．

k-means を採用した理由として，NMF クラスタリングと同様に，ユークリッド距離による評価関数を用いるため，計算方法が似ていてかつ，あらかじめ評価のベースを与えることにより，正解率が改善されるのではないかと考える．

ここで，全文書が N 個のデータのうち，i 番目の文書ベクトルに当てられたクラスタ番号を c_i とすると，k-means による全文書ベクトルの各クラスタ番号は，以下の式に示すことが出来る．

$$Class_{k-means} = [c_1, \cdots, c_i, \cdots, c_N] \tag{4.1}$$

式(4.1)から，行列 V の作成を行う．c_i によって指定されたクラスタ番号の要素のみ，他の要素より値が高くなるように，以下に示す式(4.2)で初期値を決定する．

$$V = [V_1, \cdots, V_i, \cdots, V_N], V_i = \begin{bmatrix} v_{1i} \\ \vdots \\ v_{c_i i} \\ \vdots \\ v_{ki} \end{bmatrix}$$

$$(\text{ただし } v_{c_i i} = \frac{2}{k}, v_{ji(j \neq c_i)} = \frac{1 - v_{c_i i}}{k-1}) \tag{4.2}$$

ここで，k は全クラスタ数を示す．c_i によって指定されたクラスタの文書ベクト

ルの要素は，その要素が c_i のクラスタに分類される可能性が高いと見て，クラスタ数分の平均確率に 2 倍した値を採用している．文書ベクトル内の要素の合計が 1 になるように，残りの要素は，1 から平均確率を 2 倍した値を引き，残りを同じベクトル要素に分配している．

第5章　実験

5.1　実験の概要

実際に，第4章で提唱した手法が，どれほど有効性のあるものなのか，調べる必要がある．

よって，第4章での手法を本手法とし，クラスタリングを行い，また，比較のために他のクラスタリング手法及び，元の NMF クラスタリングについても同様にクラスタリングを行う．正解率を比較することにより，クラスタリングの有効性を確認することを目的とする．

本実験では，k-means，NMF，CLUTO，及び本手法により文書クラスタリングを行う．クラスタリングを行うデータは，以下に示す 3 種類について扱う．

- 5 種類のカテゴリから取得したネットニュース記事
  全 395 文書，全単語数 4798 個
  (以降，News と呼ぶ)
- 8 分野に分類可能な評価用データセット(英語)
  全 313 文書，全単語数 5804 個
  (以降，tr12 と呼ぶ)
- 10 分野に分類される評価用データセット(英語)
  全 878 文書，全単語数 7454 個
  (以降，tr41 と呼ぶ)

図 5.1　文書クラスタリングを行うデータの内容

なお，クラスタリングの正解率を導くため，文書群の元々のカテゴリや分野が既知となっている．

また，News 文書群の元々の 5 種類のカテゴリは，それぞれ，国際関係 53 文書，政治関係 56 文書，社会関係 182 文書，スポーツ関係 40 文書，経済関係 63 文書，計 395 文書となっている．この文書は，毎日新聞社サイト(http://www.mainichi.co.jp/)より，2003 年 11 月 25 日から，同年の 12 月 5 日までの期間に，それぞれのカテゴリに掲載されたニュースを抜粋した．

具体的な文書一例として，図 5.2 に，使用した文書の一部を抜粋した．

tr12 文書群及び tr41 文書群は，クラスタリング手法の評価用のデータセットのため，文書という形ではなく，既に単語頻度数を要素とした索引語文書行列として提供されている．また，索引語文書行列の形として提供されているだけでなく，正解となる分類項目を番号で提示，また，どのような単語が使われているかの単語一覧

も共に提供している.

tr12 の 8 種類のカテゴリは，それぞれ，54 番 34 文書，58 番 29 文書，77 番 9 文書，78 番 30 文書，82 番 35 文書，94 番 29 文書，95 番 54 文書，100 番 93 文書，計 313 文書となっている.

同様に tr41 の 10 種類のカテゴリは，それぞれ，352 番 174 文書，357 番 162 文書，351 番 26 文書，354 番 243 文書，359 番 18 文書，360 番 83 文書，358 番 33 文書，355 番 35 文書，353 番 95 文書，356 番 9 文書，計 878 文書となっている.

CLUTO を除く，各クラスタリング手法でのクラスタリングを 5 回行い，それぞれのベーススコア，クラスタリング結果，及び，正解率を求める.

CLUTO は，安定した正解率を誇るため，出力結果が変わらないためである.

また，CLUTO に付属されているオプションは一切使用せずに，スタンダードのままプログラムを動かすことにするため，計算手法は，k-way 解法の一つである，k-1 repeated bisection を用いる.

実験に用いるクラスタリング手法プログラムとして，k-means は，統計解析ソフト R を用い，NMF は Java において，プログラミングして作成した物，k-way 解法を用いる CLUTO の 3 種類のプログラムを利用した.

・ 経済：2003 年 11 月 25 日

中国首相：高速鉄道計画は「検討作業中」　訪中代表団と会談

【北京・浦松丈二】中国の温家宝首相は２４日，訪中した奥田碩・日本経団連会長を最高顧問とする日中経済協会訪中代表団と北京の人民大会堂で会談した．温首相は，日独仏が受注を争う上海－北京の高速鉄道計画について「検討作業中」と述べ，採用方式の最終決定を下していないことを明らかにした．

温首相は「高速鉄道は路線が長く，コストも高い．膨大な資金がかかるため，政府として科学的に市場要素を調べ，広く意見を聞く必要がある．（専門家らによる）検討作業中だ」と説明した．検討の終了時期などについても明らかにしなかった．

代表団には奥田会長ほか経済界首脳ら約９０人が参加して２３日に北京入りしている．

図 5.2(a)　ネットニュース記事一例(抜粋)

・ 国際：2003 年 11 月 25 日

一時閉鎖：ＵＡＥの衛星テレビの支局を　イラク統治評議会が

【バグダッド竹之内満】イラク統治評議会は２４日，アラブ首長国連邦（ＵＡＥ）のアラビア語衛星テレビ「アルアラビーヤ」のイラクでの活動を禁止し，バグダッドの支局を一時閉鎖すると発表した．統治評議会のタラバニ議長は，活動禁止の理由について「フセイン元大統領による，人殺しを先導する声明テープを放送したため」などと説明している．

ＡＦＰ通信によると，アルアラビーヤは「バグダッドの支局はイラク警察当局に閉鎖された」と短いコメントを出した．またバグダッド支局の記者は「イラク警察は暴力を助長せないと確約するまで支局を閉鎖し，機材を差し押さえるという統治評議会の決定を伝えてきた」と語った．

アルアラビーヤは今月１６日，占領軍に対する徹底抗戦を呼び掛けるフセイン元大統領の肉声とみられる声明テープを放送した．ラムズフェルド米国防長官は，アルアラビーヤとカタールのアルジャジーラのアラブ圏衛星テレビ２局を名指しし，「猛烈に敵対的」だと非難していた．

図 5.2(b)　ネットニュース記事一例(抜粋)

・ 政治：2003 年 11 月 26 日

公益法人：「制度改革に関する有識者会議」設置へ

「公益法人制度改革に関する有識者会議」（座長・福原義春資生堂名誉会長）が２５日設置され，２８日に初会合を開く．民間の非営利法人活動を促進する狙いで始めた同改革だが，法人への課税問題で政府・与党内から異論が噴出し，今年３月に予定していた改革大綱の閣議決定は先送りされた．有識者会議は来春をめどに新法人の設立方法などの論点を整理し，来年中に最終報告をまとめる．

図 5.2(c)　ネットニュース記事一例(抜粋)

・ 社会：2003 年 11 月 25 日

養殖ニシキゴイ：４０００匹が大量死　KIIVか　新潟県中部

新潟県内のニシキゴイ養殖業者でつくる全日本錦鯉振興会新潟地区（事務局・小千谷市）は２４日，県中部の養殖業者のニシキゴイ約４０００匹が今月１７～２３日，大量死したと発表した．報告を受けた県はコイヘルペスウイルス（KIIV）感染の疑いが強いとみて，独立行政法人・水産総合研究センター養殖研究所（三重県）に１０匹を送り，検査を頼んだ．

振興会によると，まず１７日に約２００匹が死んでいるのが見つかり，その後も次々と死んだという．死んだコイは複数の池と水槽に分かれていたが，同じ水が循環する仕組みになっていたという．大半は既に焼却処分された．振興会は「損害額や仕入れ先は分かっていない」と話している．

ニシキゴイの感染例は青森県と岡山県で確認されている．全国有数のニシキゴイ産地の新潟県では今月１４日，茨城県・霞ケ浦産マゴイの感染が県西部で確認された．このためニシキゴイ業者の間でも，被害を防ぐためセリやニシキゴイの移動を自粛していた．【田苗学】

図 5.2(d)　ネットニュース記事一例(抜粋)

・ スポーツ：2003 年 11 月 25 日

大相撲：一夜明けた栃東　初場所へ決意新たに

大相撲九州場所で１１場所ぶり２回目の優勝を果たした西大関・栃東（２７）＝本名・志賀太祐，玉ノ井部屋＝が，優勝から一夜明けた２４日朝，福岡県須恵町の同部屋で記者会見し，「まだ夢を見ているみたい．本当にうれしい」と改めて喜びを語った．

栃東は終始リラックスした表情だった．優勝を決めた朝青龍との結びの一番で，勝った直後に土俵上でガッツポーズをしたことに質問が飛ぶと，「本当はあまりああいうことをしてはいけないんだけど，思わず出てしまった」と苦笑した．

また，昨年初場所で初優勝した後，けがなどで苦しんだ日々を振り返って，「もう相撲を取りたくない，辞めたいと思って師匠に相談したこともある」と打ち明けた．その上で，「完全復活か」という質問に「そうですね．もう大丈夫」ときっぱり．２度目の綱取り場所となる来年初場所についても「いい形で結果を残せるように頑張りたい」と語った．【井上俊樹】

図 5.2(e)　ネットニュース記事一例(抜粋)

5.2 k-means 文書クラスタリング

データごとの k-means のクラスタリング結果を示す.

News について k-means でクラスタリングした結果を以下に示す.

表 5.1(a) News データの k-means クラスタリング 1 回目結果表

1 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	3	34	0	0	26	63	1	0.539683	
国際	31	0	0	14	8	53	0	0.584906	
政治	20	13	0	21	3	57	3	0.368421	0.60153
社会	5	2	5	11	159	182	4	0.873626	
スポーツ	0	0	25	0	14	39	2	0.641026	

表 5.1(b) News データの k-means クラスタリング 2 回目結果表

2 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	29	0	31	0	3	63	0	0.460317	
国際	0	10	33	0	10	53	1	0.188679	
政治	0	7	21	0	29	57	4	0.508772	0.52679
社会	8	20	152	1	1	182	2	0.835165	
スポーツ	0	0	14	25	0	39	3	0.641026	

表 5.1(c) News データの k-means クラスタリング 3 回目結果表

3 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	1	24	2	0	36	63	4	0.571429	
国際	0	33	0	18	2	53	3	0.339623	
政治	0	21	10	24	2	57	2	0.175439	0.54257
社会	52	114	0	11	5	182	0	0.626374	
スポーツ	0	39	0	0	0	39	1	1.000000	

表 5.1(d)　News データの k-means クラスタリング 4 回目結果表

4 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	39	0	23	0	1	63	0	0.619048	
国際	22	20	0	11	0	53	3	0.207547	
政治	21	23	0	13	0	57	1	0.403509	0.46690
社会	113	16	2	4	47	182	4	0.258242	
スポーツ	6	0	33	0	0	39	2	0.846154	

表 5.1(e)　News データの k-means クラスタリング 5 回目結果表

5 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	0	2	0	0	61	63	4	0.968254	
国際	7	3	16	0	27	53	2	0.301887	
政治	11	0	22	0	24	57	0	0.192982	0.56295
社会	3	92	11	4	72	182	1	0.505495	
スポーツ	0	0	0	33	6	39	3	0.846154	

News を k-means で 5 回クラスタリングした正解率平均 Average は,

Average=0.5402=54.02(%)

であった.

次に, tr12 を k-means で 5 回クラスタリングした結果を示す.

表 5.2(a)　tr12 データの k-means クラスタリング 1 回目結果表

1 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	0	23	0	0	3	1	0	3	30	1	0.766667	
54	0	0	30	0	0	2	0	2	34	2	0.882353	
82	29	0	0	0	1	1	0	4	35	0	0.828571	
58	0	0	0	0	0	0	29	0	29	6	1.000000	0.68063
100	0	0	0	62	0	4	0	27	93	3	0.666667	
95	0	0	0	0	38	1	0	15	54	7	0.277778	
94	0	0	0	0	20	6	0	3	29	4	0.689655	
77	0	0	0	0	6	3	0	0	9	5	0.333333	

表 5.2(b)　tr12 データの k-mcans クラスタリング 2 回目結果表

2 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	4	26	0	0	0	0	0	0	30	1	0.866667	0.62192
54	4	0	0	0	0	30	0	0	34	5	0.882353	
82	4	0	0	29	0	0	2	0	35	3	0.828571	
58	0	0	29	0	0	0	0	0	29	2	1.000000	
100	32	0	0	0	11	0	0	50	93	7	0.537634	
95	15	0	0	0	0	0	39	0	54	6	0.722222	
94	4	0	0	0	0	0	25	0	29	0	0.137931	
77	0	8	0	0	0	0	1	0	9	4	0.000000	

表 5.2(c)　tr12 データの k-means クラスタリング 3 回目結果表

3 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	0	1	24	0	0	3	2	0	30	2	0.800000	0.4739
54	22	0	0	0	8	0	4	0	34	0	0.647059	
82	0	0	31	0	0	1	3	0	35	6	0.085714	
58	0	0	0	29	0	0	0	0	29	3	1.000000	
100	0	3	0	0	15	1	38	36	93	7	0.387097	
95	0	28	1	0	0	21	4	0	54	5	0.388889	
94	0	14	0	0	0	14	1	0	29	1	0.482759	
77	0	3	1	0	0	5	0	0	9	4	0.000000	

表 5.2(d)　tr12 データの k-means クラスタリング 4 回目結果表

4 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	0	4	0	0	25	0	1	0	30	0	0.000000	0.66585
54	0	0	0	34	0	0	0	0	34	3	1.000000	
82	0	2	0	0	33	0	0	0	35	4	0.942857	
58	14	0	0	0	0	15	0	0	29	5	0.517241	
100	1	0	0	3	4	0	85	0	93	6	0.913978	
95	2	50	0	0	2	0	0	0	54	1	0.925926	
94	0	22	4	0	0	0	1	2	29	2	0.137931	
77	0	0	0	0	1	0	0	8	9	7	0.888889	

表 5.2(c)　tr12 データの k-means クラスタリング 5 回目結果表

5 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	5	17	0	0	1	0	7	0	30	6	0.233333	0.50336
54	2	0	0	0	4	28	0	0	34	5	0.823529	
82	7	1	0	0	2	25	0	0	35	4	0.057143	
58	0	13	0	0	0	0	1	15	29	7	0.517241	
100	21	0	0	46	4	1	21	0	93	3	0.494624	
95	23	27	0	0	1	3	0	0	54	0	0.425926	
94	5	17	2	0	4	1	0	0	29	1	0.586207	
77	0	0	8	0	0	1	0	0	9	2	0.888889	

tr12 を k-means で 5 回クラスタリングした正解率平均 Average は,

Average= 0.5891 =58.91(%)

であった.

次に, tr41 を k-means で 5 回クラスタリングした結果を示す.

表 5.3(a)　tr41 データの k-means クラスタリング 1 回目結果表

1 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	31	0	1	0	0	0	141	0	1	0	174	6	0.810345	0.52795
357	12	0	0	51	42	31	0	0	21	5	162	3	0.314815	
351	14	0	1	0	0	0	0	0	11	0	26	5	0.000000	
354	21	0	98	25	1	5	0	0	3	90	243	9	0.370370	
359	18	0	0	0	0	0	0	0	0	0	18	4	0.000000	
360	2	0	32	0	0	0	0	0	49	0	83	2	0.385542	
358	0	32	1	0	0	0	0	0	0	0	33	1	0.969697	
355	1	0	0	0	0	0	0	0	34	0	35	8	0.971429	
353	1	0	0	0	0	0	0	54	40	0	95	7	0.568421	
356	8	0	1	0	0	0	0	0	0	0	9	0	0.888889	

表 5.3(b)　tr41 データの k-means クラスタリング 2 回目結果表

2 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	30	0	1	0	0	0	0	142	1	174	8	0.816092	0.68033
357	0	8	0	3	0	146	0	2	0	3	162	5	0.901235	
351	0	15	0	0	0	10	0	0	0	1	26	0	0.000000	
354	35	1	4	87	9	1	0	1	0	105	243	9	0.432099	
359	0	18	0	0	0	0	0	0	0	0	18	1	1.000000	
360	0	0	0	0	83	0	0	0	0	0	83	4	1.000000	
358	0	0	32	0	0	0	0	0	0	1	33	2	0.969697	
355	0	0	0	0	0	0	0	35	0	0	35	7	1.000000	
353	0	0	0	0	0	0	23	65	0	7	95	6	0.684211	
356	0	4	0	0	0	0	0	1	0	4	9	3	0.000000	

表 5.3(c)　tr41 データの k-means クラスタリング 3 回目結果表

3 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	122	0	5	0	3	0	0	43	1	174	1	0.701149	0.68425
357	133	0	0	20	0	8	0	0	0	1	162	0	0.820988	
351	11	0	0	14	0	0	0	0	0	1	26	3	0.538462	
354	23	0	35	31	0	1	0	9	0	144	243	9	0.592593	
359	0	0	0	1	0	17	0	0	0	0	18	5	0.944444	
360	0	0	0	1	0	0	0	82	0	0	83	7	0.987952	
358	0	0	0	0	32	0	0	0	0	1	33	4	0.969697	
355	1	0	0	0	0	34	0	0	0	0	35	5	0.971429	
353	2	0	0	43	0	20	30	0	0	0	95	6	0.315789	
356	0	0	0	6	0	3	0	0	0	0	9	2	0.000000	

表 5.3(d)　tr41 データの k-means クラスタリング 4 回目結果表

4 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	0	96	41	0	35	1	0	1	0	174	2	0.551724	0.46455
357	2	50	0	0	0	11	7	0	2	90	162	9	0.555556	
351	0	0	0	0	0	14	11	0	1	0	26	6	0.423077	
354	1	25	0	0	0	11	87	0	117	2	243	8	0.069959	
359	0	0	0	0	0	18	0	0	0	0	18	5	0.833333	
360	0	0	0	0	0	1	49	0	33	0	83	3	0.000000	
358	0	0	0	0	0	0	0	32	1	0	33	7	0.969697	
355	35	0	0	0	0	0	0	0	0	0	35	0	1.000000	
353	71	0	0	0	23	1	0	0	0	0	95	4	0.242105	
356	0	0	0	0	0	7	0	0	2	0	9	1	0.000000	

表 5.3(e)　tr41 データの k-means クラスタリング 5 回目結果表

5 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	1	16	0	0	0	0	0	113	0	44	174	7	0.649425	0.49264
357	0	14	58	8	24	1	57	0	0	0	162	2	0.358025	
351	1	25	0	0	0	0	0	0	0	0	26	9	0.000000	
354	80	44	4	68	21	26	0	0	0	0	243	3	0.279835	
359	0	18	0	0	0	0	0	0	0	0	18	1	1.000000	
360	31	3	0	49	0	0	0	0	0	0	83	6	0.000000	
358	33	0	0	0	0	0	0	0	0	0	33	0	1.000000	
355	1	0	5	0	0	29	0	0	0	0	35	5	0.828571	
353	0	2	5	0	0	11	0	0	77	0	95	8	0.810526	
356	1	8	0	0	0	0	0	0	0	0	9	4	0.000000	

tr41 を k-means で 5 回クラスタリングした正解率平均 Average は,

$$Average = 0.5699 = 56.99(\%)$$

であった.

以上, 3 種類のデータによるクラスタリング結果について以下にまとめる.

5 回行ったクラスタリング結果から得られた正解率について以下の表に示す.

表 5.4　k-means クラスタリング各データ正解率比較表

データ	1 回目	2 回目	3 回目	4 回目	5 回目	正解率平均
News	0.60153	0.52679	0.54257	0.46690	0.56295	0.5401
tr12	0.68063	0.62192	0.47394	0.66585	0.50336	0.5891
tr41	0.52795	0.68033	0.68425	0.46455	0.49264	0.5699

また，1 回目から 5 回目までのクラスタリング正解率の比較グラフを以下に示す．

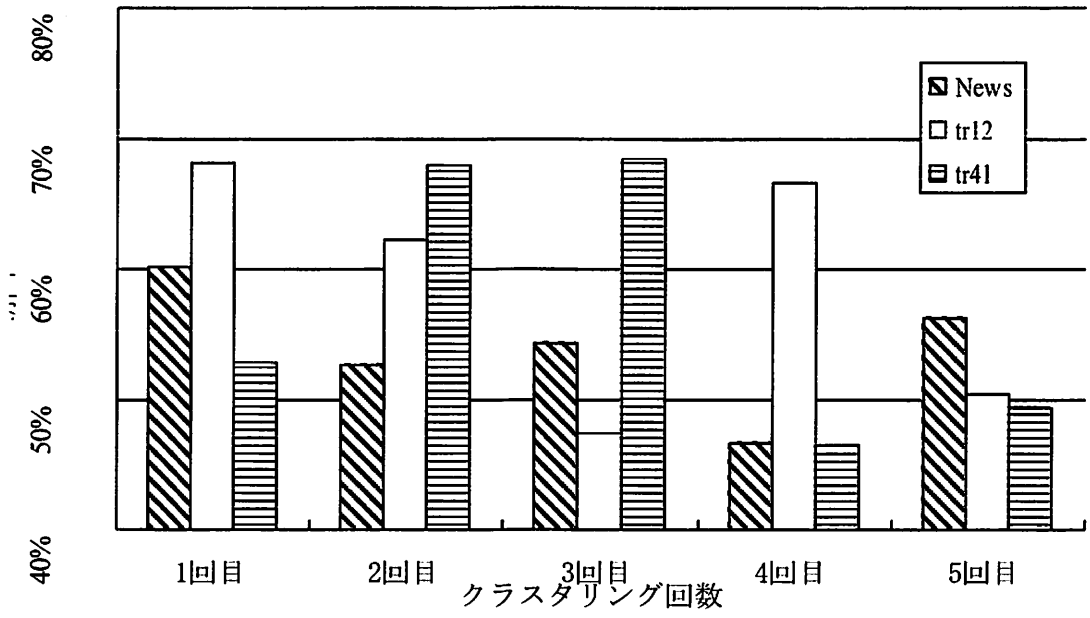

図 5.2　各データの k-means クラスタリング正解率比較グラフ

5.3 NMF 文書クラスタリング

NMF によるクラスタリング結果を，データごとに示す．まず，News データのクラスタリング結果を示す．

表 5.5(a)　News データの NMF クラスタリング 1 回目結果表

1 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	11	9	1	42	0	63	0	0.174603	
国際	11	3	18	3	18	53	4	0.339623	
政治	19	0	30	0	8	57	2	0.526316	0.545965
社会	16	130	6	19	11	182	1	0.714286	
スポーツ	0	0	1	39	0	40	3	0.975	

表 5.5(b)　News データの NMF クラスタリング 2 回目結果表

2 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	0	48	8	4	3	63	1	0.761905	
国際	25	2	18	3	5	53	2	0.339623	
政治	26	18	13	0	0	57	0	0.45614	0.59324
社会	14	32	18	88	30	182	3	0.483516	
スポーツ	0	0	3	0	37	40	4	0.925000	

表 5.5(c)　News データの NMF クラスタリング 3 回目結果表

3 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	11	39	10	3	0	63	2	0.158730	
国際	9	3	1	24	16	53	4	0.301887	
政治	9	0	19	21	8	57	3	0.368421	0.43471
社会	90	54	9	4	25	182	0	0.494505	
スポーツ	5	34	0	1	0	40	1	0.850000	

表 5.5(d)　News データの NMF クラスタリング 4 回目結果表

4 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	15	1	42	5	0	63	0	0.238095	
国際	6	24	1	4	18	53	1	0.452830	
政治	20	27	2	0	8	57	4	0.140351	0.46213
社会	22	1	110	15	34	182	2	0.604396	
スポーツ	0	0	5	35	0	40	3	0.875000	

表 5.5(e)　News データの NMF クラスタリング 5 回目結果表

5 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	1	1	4	1	56	63	4	0.888889	
国際	13	19	3	8	10	53	0	0.245283	
政治	3	27	3	0	24	57	1	0.473684	0.60498
社会	43	3	85	9	42	182	2	0.467033	
スポーツ	0	0	2	38	0	40	3	0.950000	

News を NMF で 5 回クラスタリングした正解率平均 Average は,

Average= 0.5282=52.82(%)

であった.

次に, tr12 を NMF でクラスタリングした結果を示す.

表 5.6(a)　tr12 データの NMF クラスタリング 1 回目結果表

1 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	12	6	5	2	1	1	3	0	30	7	0.000000	
54	2	7	0	7	0	18	0	0	34	5	0.529412	
82	3	1	0	1	30	0	0	0	35	4	0.857143	
58	0	0	29	0	0	0	0	0	29	2	1.000000	0.50343
100	27	21	0	3	0	0	33	9	93	6	0.354839	
95	44	2	2	6	0	0	0	0	54	0	0.814815	
94	23	0	1	4	0	0	1	0	29	3	0.137931	
77	3	3	1	1	0	0	1	0	9	1	0.333333	

表 5.6(b)　tr12 データの NMF クラスタリング 2 回目結果表

2 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	1	4	0	4	0	17	2	2	30	5	0.566667	0.44975
54	1	2	0	4	26	0	0	1	34	4	0.764706	
82	1	14	0	1	18	1	0	0	35	1	0.400000	
58	0	0	0	0	0	0	29	0	29	6	1.000000	
100	2	18	36	11	2	0	0	24	93	7	0.258065	
95	7	17	1	5	2	22	0	0	54	0	0.129630	
94	4	6	1	3	1	14	0	0	29	2	0.034483	
77	4	1	0	4	0	0	0	0	9	3	0.444444	

表 5.6(c)　tr12 データの NMF クラスタリング 3 回目結果表

3 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	0	4	3	19	0	2	0	2	30	3	0.633333	0.61506
54	0	0	3	0	28	0	0	3	34	4	0.823529	
82	28	1	0	1	0	2	0	3	35	0	0.800000	
58	0	0	0	0	0	0	29	0	29	6	1.000000	
100	2	16	42	8	1	8	0	16	93	2	0.451613	
95	0	16	0	21	2	8	1	6	54	1	0.296296	
94	1	10	0	8	0	6	0	4	29	7	0.137931	
77	0	1	0	0	0	7	0	1	9	5	0.777778	

表 5.5(d)　tr12 データの NMF クラスタリング 4 回目結果表

4 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	1	0	4	0	2	17	0	6	30	5	0.566667	0.53611
54	1	0	0	0	5	3	24	1	34	6	0.705882	
82	4	28	0	0	0	0	3	0	35	1	0.800000	
58	0	0	1	28	0	0	0	0	29	3	0.965517	
100	14	0	26	0	5	24	3	21	93	2	0.279570	
95	7	25	0	0	1	2	4	15	54	7	0.277778	
94	4	12	0	1	3	0	1	8	29	0	0.137931	
77	0	0	0	0	5	1	1	2	9	4	0.555556	

表 5.6(c)　tr12 データの NMF クラスタリング 5 回目結果表

5 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	1	5	11	0	1	6	3	3	30	2	0.366667	0.48503
54	0	0	2	26	4	0	0	2	34	3	0.764706	
82	29	0	3	0	2	0	1	0	35	0	0.828571	
58	0	0	0	0	0	29	0	0	29	5	1.000000	
100	0	42	5	1	4	0	8	33	93	1	0.451613	
95	0	0	19	3	5	0	16	11	54	6	0.296296	
94	0	0	6	2	4	1	11	5	29	7	0.172414	
77	0	0	4	0	0	0	4	1	9	4	0.000000	

tr12 を NMF で 5 回クラスタリングした正解率平均 Average は,

Average= 0.5179=51.79(%)

であった.

次に, tr41 を NMF でクラスタリングした結果を示す.

表 5.7(a)　tr41 データの NMF クラスタリング 1 回目結果表

1 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	9	0	0	0	0	0	0	1	164	0	174	8	0.942529	0.58295
357	10	5	33	60	28	0	5	20	1	0	162	3	0.37037	
351	0	0	24	0	0	1	0	0	1	0	26	2	0.923077	
354	12	0	28	16	7	64	1	93	4	18	243	7	0.382716	
359	1	0	0	0	0	0	3	0	14	0	18	5	0.000000	
360	19	0	1	0	44	3	0	1	0	15	83	4	0.53012	
358	0	0	0	0	0	0	0	0	0	33	33	9	1.000000	
355	0	0	0	0	0	0	35	0	0	0	35	6	1.000000	
353	32	33	0	0	0	1	26	1	2	0	95	1	0.347368	
356	3	0	0	0	0	0	0	0	6	0	9	0	0.333333	

表 5.7(b)　tr41 データの NMF クラスタリング 2 回目結果表

2 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	1	0	0	0	47	0	25	101	0	174	8	0.58046	0.60257
357	6	63	0	77	1	0	2	5	7	1	162	1	0.388889	
351	1	0	1	20	0	0	0	3	1	0	26	3	0.769231	
354	16	98	22	0	0	0	1	6	14	86	243	9	0.353909	
359	0	0	0	0	0	0	0	18	0	0	18	7	1.000000	
360	51	0	31	0	0	0	0	0	1	0	83	0	0.614458	
358	0	0	33	0	0	0	0	0	0	0	33	2	1.000000	
355	0	1	0	0	0	0	34	0	0	0	35	6	0.971429	
353	0	2	0	0	33	0	57	1	2	0	95	4	0.347368	
356	0	0	0	0	0	0	2	5	2	0	9	5	0.000000	

表 5.7(c)　tr41 データの NMF クラスタリング 3 回目結果表

3 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	84	0	0	0	2	0	68	20	0	174	1	0.482759	0.54397
357	0	0	4	0	85	1	56	7	5	4	162	4	0.524691	
351	0	0	0	0	17	1	0	2	5	1	26	0	0.000000	
354	0	0	0	31	0	97	79	21	3	12	243	3	0.127572	
359	0	0	0	0	0	0	0	0	18	0	18	8	1.000000	
360	0	0	0	0	0	13	0	1	0	69	83	9	0.831325	
358	0	0	0	0	0	33	0	0	0	0	33	5	1.000000	
355	2	0	0	0	0	1	32	0	0	0	35	6	0.914286	
353	21	0	32	1	0	8	17	13	3	0	95	2	0.336842	
356	0	0	0	0	0	1	0	2	6	0	9	7	0.222222	

表 5.7(d)　tr41 データの NMF クラスタリング 4 回目結果表

4 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	53	1	1	0	108	0	0	11	0	174	5	0.620690	0.62912
357	0	0	1	123	0	0	20	3	15	0	162	3	0.759259	
351	0	1	1	0	0	0	24	0	0	0	26	6	0.923077	
354	0	0	150	28	16	0	17	20	12	0	243	2	0.617284	
359	0	4	0	0	0	0	1	0	13	0	18	1	0.222222	
360	0	0	1	0	11	0	1	69	1	0	83	7	0.831325	
358	0	0	0	0	33	0	0	0	0	0	33	4	1.000000	
355	2	0	0	2	0	0	0	0	31	0	35	8	0.885714	
353	7	0	0	2	0	1	31	0	13	41	95	9	0.431579	
356	0	0	0	0	0	0	2	0	5	2	9	0	0.000000	

表 5.7(e)　tr41 データの NMF クラスタリング 5 回目結果表

5 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	0	1	0	1	79	0	89	4	0	174	7	0.511494	0.57477
357	0	3	12	60	0	5	4	0	8	70	162	3	0.370370	
351	1	0	24	0	0	1	0	0	0	0	26	2	0.923077	
354	55	7	22	0	27	8	1	0	34	89	243	9	0.366255	
359	0	0	0	0	1	10	3	3	1	0	18	5	0.555556	
360	0	41	0	0	0	1	0	0	41	0	83	1	0.493976	
358	33	0	0	0	0	0	0	0	0	0	33	0	1.000000	
355	0	0	0	0	0	0	34	0	0	1	35	6	0.971429	
353	4	0	10	1	0	2	76	0	1	1	95	4	0.000000	
356	0	0	1	0	0	3	0	0	5	0	9	8	0.555556	

tr41 を NMF で 5 回クラスタリングした正解率平均 Average は,

$$\text{Average}= 0.5867=58.67(\%)$$

であった.

以上，3 種類のデータによるクラスタリング結果について以下にまとめる.

5 回行ったクラスタリング結果から得られた正解率について以下の表に示す.

表 5.8　NMF クラスタリングによる正解率比較表

データ	1 回目	2 回目	3 回目	4 回目	5 回目	正解率平均
News	0.54597	0.59324	0.43471	0.46213	0.60498	0.5282
tr12	0.50343	0.44975	0.61506	0.53611	0.48503	0.5179
tr41	0.58295	0.60257	0.54397	0.62912	0.57477	0.5867

また，1 回目から 5 回目までのクラスタリング正解率比較グラフを以下に示す．

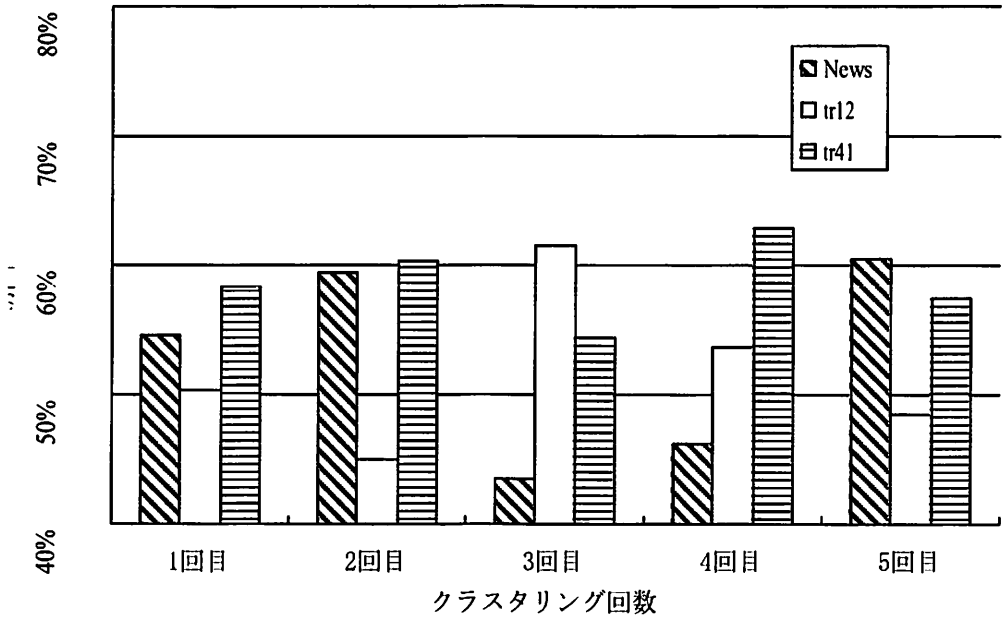

図 5.3　各データの NMF クラスタリング正解率比較グラフ

5.4 k-means の結果を用いた NMF による文書クラスタリング

各データの本手法によるクラスタリング結果について示す．本手法に用いた k-means クラスタリング結果は，5 回中，それぞれで一番正解率が良いものを使用した．まず News を本手法でクラスタリングした結果を示す．

表 5.9(a)　News データの本手法クラスタリング 1 回目結果表

1 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	12	46	2	0	3	63	1	0.730159	
国際	30	0	0	20	3	53	0	0.566038	
政治	18	16	0	23	0	57	3	0.403509	0.64752
社会	31	23	8	13	107	182	4	0.587912	
スポーツ	2	0	38	0	0	40	2	0.950000	

表 5.9(b)　News データの本手法クラスタリング 2 回目結果表

2 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	12	45	3	0	3	63	1	0.714286	
国際	29	0	1	20	3	53	0	0.547170	
政治	19	16	0	22	0	57	3	0.385965	0.63817
社会	29	24	8	13	108	182	4	0.593407	
スポーツ	2	0	38	0	0	40	2	0.950000	

表 5.9(c)　News データの本手法クラスタリング 3 回目結果表

3 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	13	44	3	0	3	63	1	0.698413	
国際	30	0	0	20	3	53	0	0.566038	
政治	19	16	0	22	0	57	3	0.578947	0.67626
社会	31	23	8	13	107	182	4	0.587912	
スポーツ	2	0	38	0	0	40	2	0.950000	

表 5.9(d)　News データの本手法クラスタリング 4 回目結果表

4 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	13	44	3	0	3	63	1	0.698413	
国際	30	0	0	20	3	53	0	0.566038	
政治	17	17	0	23	0	57	3	0.403509	0.64227
社会	30	23	8	13	108	182	4	0.593407	
スポーツ	2	0	38	0	0	40	2	0.950000	

表 5.9(e)　News データの本手法クラスタリング 5 回目結果表

5 回目	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	12	45	3	0	3	63	1	0.714286	
国際	29	0	1	20	3	53	0	0.547170	
政治	19	16	0	22	0	57	3	0.385965	0.63817
社会	28	25	8	13	108	182	4	0.593407	
スポーツ	2	0	38	0	0	40	2	0.950000	

News を本手法で 5 回クラスタリングした正解率平均 Average は,

Average= 0.6485=64.85(%)

であった.

次に, tr12 を本手法でクラスタリングした結果を示す.

表 5.10(a)　tr12 データの本手法クラスタリング 1 回目結果表

1 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	2	28	0	0	0	0	0	0	30	1	0.933333	
54	4	0	0	0	0	29	0	1	34	5	0.852941	
82	4	0	0	28	1	0	2	0	35	3	0.800000	
58	0	0	28	0	1	0	0	0	29	2	0.965517	
100	35	0	0	0	13	1	0	44	93	7	0.473118	0.64855
95	20	0	0	1	0	2	31	0	54	0	0.370370	
94	5	0	0	0	1	0	23	0	29	6	0.793103	
77	0	8	0	1	0	0	0	0	9	4	0.000000	

表 5.10(b)　tr12 データの本手法クラスタリング 2 回目結果表

2回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	4	26	0	0	0	0	0	0	30	1	0.866667	0.64836
54	3	0	0	0	0	31	0	0	34	5	0.911765	
82	4	0	0	29	0	0	2	0	35	3	0.828571	
58	0	0	29	0	0	0	0	0	29	2	1.000000	
100	32	0	0	0	23	1	0	37	93	7	0.397849	
95	21	0	0	0	0	2	31	0	54	0	0.388889	
94	5	0	0	0	1	0	23	0	29	6	0.793103	
77	4	2	0	0	0	0	3	0	9	4	0.000000	

表 5.10(c)　tr12 データの本手法クラスタリング 3 回目結果表

3回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	2	28	0	0	0	0	0	0	30	1	0.933333	0.65943
54	3	0	0	0	0	31	0	0	34	5	0.911765	
82	2	0	0	31	1	0	1	0	35	3	0.885714	
58	0	0	29	0	0	0	0	0	29	2	1.000000	
100	32	0	0	0	11	0	0	50	93	7	0.537634	
95	13	0	0	0	9	2	30	0	54	0	0.240741	
94	5	0	0	0	5	0	19	0	29	6	0.655172	
77	2	3	0	0	1	0	3	0	9	4	0.111111	

表 5.10(d)　tr12 データの本手法クラスタリング 4 回目結果表

4回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	4	26	0	0	0	0	0	0	30	1	0.866667	0.70020
54	3	0	0	0	0	31	0	0	34	5	0.911765	
82	4	0	0	28	1	0	2	0	35	3	0.800000	
58	0	0	29	0	0	0	0	0	29	2	1.000000	
100	28	0	0	0	14	3	0	48	93	7	0.516129	
95	22	0	0	0	0	1	31	0	54	0	0.407407	
94	5	0	1	0	4	0	19	0	29	6	0.655172	
77	3	1	0	1	4	0	0	0	9	4	0.444444	

表 5.10(d)　tr12 データの本手法クラスタリング 5 回目結果表

5 回目	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	5	25	0	0	0	0	0	0	30	1	0.833333	0.62863
54	1	0	0	0	1	29	0	3	34	5	0.852941	
82	3	0	0	26	4	0	2	0	35	3	0.742857	
58	0	0	29	0	0	0	0	0	29	2	1.000000	
100	24	1	0	0	20	1	0	47	93	7	0.505376	
95	20	0	0	1	1	2	30	0	54	0	0.370370	
94	6	0	0	0	1	1	21	0	29	6	0.724138	
77	4	2	0	0	0	0	3	0	9	4	0.000000	

tr12 を本手法で 5 回クラスタリングした正解率平均 Average は,

Average=0.6570=65.70(%)

であった.

次に, tr41 を本手法でクラスタリングした結果を示す.

表 5.11(a)　tr41 データの本手法クラスタリング 1 回目結果表

1 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	24	0	0	0	0	0	0	150	0	174	8	0.862069	0.63230
357	1	13	0	6	3	132	0	2	5	0	162	5	0.814815	
351	1	20	0	0	0	4	0	0	1	0	26	0	0.038462	
354	60	10	13	98	9	1	0	1	15	36	243	3	0.403292	
359	0	18	0	0	0	0	0	0	0	0	18	1	1.000000	
360	3	3	5	0	72	0	0	0	0	0	83	4	0.86747	
358	0	0	33	0	0	0	0	0	0	0	33	2	1.000000	
355	0	0	0	0	0	0	0	35	0	0	35	7	1.000000	
353	1	0	0	0	0	0	32	59	2	1	95	6	0.336842	
356	0	4	0	0	0	0	0	2	3	0	9	9	0.000000	

表 5.11(b)　tr41 データの本手法クラスタリング 2 回目結果表

2 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	26	0	0	0	0	0	0	148	0	174	8	0.850575	0.62994
357	1	13	0	7	3	130	1	2	5	0	162	5	0.802469	
351	1	17	0	0	0	7	0	0	1	0	26	0	0.038462	
354	61	11	11	101	9	0	0	1	13	36	243	3	0.415638	
359	0	18	0	0	0	0	0	0	0	0	18	1	1.000000	
360	3	4	5	0	71	0	0	0	0	0	83	4	0.855422	
358	0	0	33	0	0	0	0	0	0	0	33	2	1.000000	
355	0	0	0	0	0	0	0	35	0	0	35	7	1.000000	
353	1	0	0	0	0	0	32	59	2	1	95	6	0.336842	
356	0	4	0	0	1	0	0	1	3	0	9	9	0.000000	

表 5.11(c)　tr41 データの本手法クラスタリング 3 回目結果表

3 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	27	0	0	0	0	0	0	147	0	174	8	0.844828	0.63430
357	1	13	0	4	3	134	0	2	5	0	162	5	0.82716	
351	1	16	0	0	0	7	0	0	2	0	26	0	0.038462	
354	59	9	11	107	9	1	0	1	13	33	243	3	0.440329	
359	0	18	0	0	0	0	0	0	0	0	18	1	1.000000	
360	3	4	5	0	71	0	0	0	0	0	83	4	0.855422	
358	0	0	33	0	0	0	0	0	0	0	33	2	1.000000	
355	0	0	0	0	0	0	0	35	0	0	35	7	1.000000	
353	1	0	0	0	0	0	32	59	2	1	95	6	0.336842	
356	0	4	0	0	0	0	0	2	3	0	9	9	0.000000	

表 5.11(d)　tr41 データの本手法クラスタリング 4 回目結果表

4 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	31	0	0	0	0	0	0	143	0	174	8	0.821839	0.63274
357	1	9	0	2	3	143	0	4	0	0	162	5	0.882716	
351	1	15	0	0	0	10	0	0	0	0	26	0	0.038462	
354	62	11	9	107	9	1	0	1	10	33	243	3	0.440329	
359	0	18	0	0	0	0	0	0	0	0	18	1	1.000000	
360	9	3	4	0	67	0	0	0	0	0	83	4	0.807229	
358	0	0	33	0	0	0	0	0	0	0	33	2	1.000000	
355	0	0	0	0	0	0	0	35	0	0	35	7	1.000000	
353	1	0	0	0	0	0	32	59	2	1	95	6	0.336842	
356	1	5	0	0	1	0	0	1	1	0	9	9	0.000000	

表 5.11(e)　tr41 データの本手法クラスタリング 5 回目結果表

5 回目	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	34	0	0	0	0	0	0	140	0	174	8	0.804598	0.63495
357	1	10	0	4	3	139	0	3	2	0	162	5	0.858025	
351	1	14	0	0	0	11	0	0	0	0	26	0	0.038462	
354	60	10	11	105	10	1	0	1	11	34	243	3	0.432099	
359	0	18	0	0	0	0	0	0	0	0	18	1	1.000000	
360	3	3	4	0	73	0	0	0	0	0	83	4	0.879518	
358	0	0	33	0	0	0	0	0	0	0	33	2	1.000000	
355	0	0	0	0	0	0	0	35	0	0	35	7	1.000000	
353	1	0	0	0	0	0	32	59	2	1	95	6	0.336842	
356	0	5	0	0	1	0	0	1	2	0	9	9	0.000000	

tr41 を本手法で 5 回クラスタリングした正解率平均 Average は,

$$\text{Average} = 0.6328 = 63.28(\%)$$

であった.

以上，3 種類のデータによるクラスタリング結果について以下にまとめる.

表 5.12　各データの本手法クラスタリング正解率比較表

データ	1 回目	2 回目	3 回目	4 回目	5 回目	正解率平均
News	0.64752	0.63817	0.67626	0.64227	0.63817	0.6485
tr12	0.64855	0.64836	0.65943	0.70020	0.62863	0.6570
tr41	0.63229	0.62994	0.63430	0.63274	0.63495	0.6328

また，1 回目から 5 回目までのクラスタリング正解率比較グラフを以下に示す．

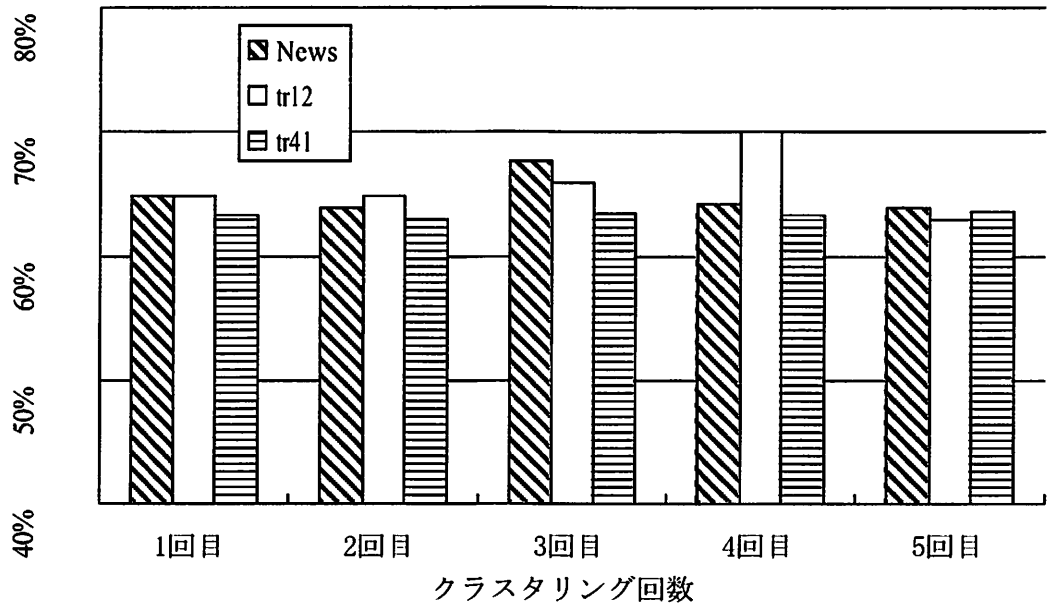

図 5.4　各データの本手法クラスタリングによる正解率比較グラフ

5.5 CLUTO による文書クラスタリング

CLUTO によるクラスタリング結果を各データごとに示す．まず News データのクラスタリング結果を示す．

表 5.13　News データの CLUTO クラスタリング結果表

カテゴリ	0	1	2	3	4	文書数合計	結果	クラスタの正解率	平均正解率
経済	0	1	46	12	4	63	2	0.730159	0.67497
国際	22	0	1	26	4	53	0	0.415094	
政治	25	0	2	30	0	57	3	0.526316	
社会	11	11	14	18	128	182	4	0.703297	
スポーツ	0	40	0	0	0	40	1	1.000000	

次に，tr12 のクラスタリング結果を示す．

表 5.14　tr12 データの CLUTO クラスタリング結果表

カテゴリ番号	0	1	2	3	4	5	6	7	文書数合計	結果	クラスタの正解率	平均正解率
78	0	1	0	0	24	1	3	1	30	4	0.800000	0.62599
54	30	0	0	1	0	0	3	0	34	0	0.882353	
82	0	0	0	0	0	33	0	2	35	5	0.942857	
58	0	29	0	0	0	0	0	0	29	1	1.000000	
100	0	0	35	33	16	3	6	0	93	2	0.376344	
95	0	2	0	0	0	4	16	32	54	7	0.592593	
94	0	0	0	0	0	0	12	17	29	6	0.413793	
77	0	0	0	0	0	1	8	0	9	3	0.000000	

さらに，tr41 のクラスタリング結果を示す．

表 5.15　tr41 データの CLUTO クラスタリング結果表

カテゴリ番号	0	1	2	3	4	5	6	7	8	9	文書数合計	結果	クラスタの正解率	平均正解率
352	0	0	0	0	151	0	0	0	23	0	174	4	0.867816	0.64373
357	0	53	2	0	0	0	1	93	11	2	162	1	0.329193	
351	0	0	0	0	0	0	1	12	12	1	26	7	0.461538	
354	35	0	0	9	0	1	51	9	20	118	243	9	0.485597	
359	0	0	0	0	0	0	0	0	18	0	18	8	1.000000	
360	0	0	0	81	0	0	1	0	1	0	83	3	0.975904	
358	0	0	0	0	0	0	33	0	0	0	33	6	1.000000	
355	1	1	2	0	0	31	0	0	0	0	35	5	0.885714	
353	2	0	41	0	0	48	0	1	2	1	95	2	0.431579	
356	0	0	0	3	0	0	0	0	6	0	9	0	0.000000	

5.6 結果

以上の 5.2 から 5.5 までのクラスタリング結果をまとめる.

各クラスタリングの正解率について，以下の表に示す. CLUTO 以外のクラスタリングは，計 5 回のクラスタリングを行った時の正解率の平均である.

表 5.16　全クラスタリング正解率比較表

	k-means	NMF	本手法	CLUTO
News	54.02	52.82	64.85	67.50
tr12	58.91	51.79	65.70	62.60
tr41	56.99	58.67	63.28	64.37

(単位:%)

表 5.13 を参考に，比較グラフを以下に示す.

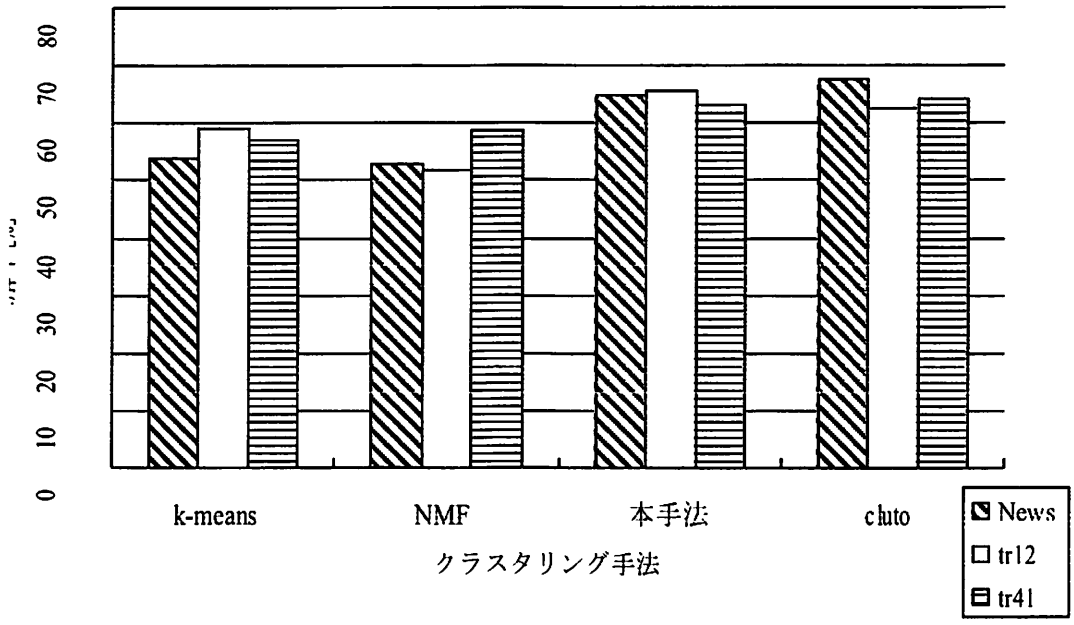

図 5.5　手法別クラスタリング正解率比較グラフ

図 5.5 を見ても分かるように，本手法における正解率は，既存の NMF だけのクラスタリングに比べて，どのデータでも同程度の正解率が出ていることが分かる．さらに，他の既存クラスタリング手法では，データによって，また，クラスタリングを数回行った時，非常に正解率にばらつきが出ていることも，第 5 章 5.2 及び 5.3 の結果から明らかであった．しかし，本手法においては，第 5 章 5.4 の結果から，ばらつきの幅が大幅に減少，さらに，高い正解率を出していることが分かる．

さらに，既存のクラスタリングの正解率と比較した時，NMF だけだとばらつきが大きくても最大で正解率が 60%に届かなかった正解率が，改善後には全て 6 割以上出ていること，さらに，ばらつきの大きい物で最大 70%の正解率を出している．しかし，まだまだばらつきがあり，このばらつきの幅をさらに狭めることが出来れば，クラスタリングの精度がまたさらに上がるのではないかと推測される．

また，CLUTO と比較をしても，ほとんど同程度のクラスタリング精度があることから，本手法の考え方も強力な手法であると考えられる．

第6章　考察

本手法により，NMF クラスタリングの正解率の改善を図ることができた．しかし，図 5.4 を見ても分かるように，ばらつきがほとんどない場合，大きい場合と分かれてしまっている．これは，初期値によって，正解率が安定，不安定になるのではないかと考えられる．そのため，初期値として与える k-means の結果内容，または，正解率が，本手法クラスタリングの正解率にどのような影響があるのかという疑問が出る．

初期値の内容によって，安定，不安定が変わるのであれば，k-means 以外のクラスタリング結果を用いた場合でも，同様に安定する場合があるのか，逆に不安定になる場合があるのかどうかを調査する必要がある．

また，距離計算として，今回はユークリッド距離のみを採用した．しかし，距離計算手法としては，他に重み付ユークリッド距離，マハラノビス距離，ミンコフスキー距離などがある．

よって，同一手法でも，距離計算手法により，結果がどのように変化するのか，また，索引語文書行列の tf-idf 値に採用した対数関数を用いた手法以外の方法など，さまざまな組み合わせを試行し，調査することによって，さらに細かく比較する必要もある．

今回の結果は，本研究において非常に良い結果であるが，同時に新たな疑問点が生じたため，以上のことについても深く調査する必要がある．

第7章　まとめ

特徴抽出に優れるNMFを，クラスタリングに用いる方法による手法について，以前から考案されていたが，この手法では，正解率にばらつきがあり，あまり良い結果が出ていなかった．そこで，今回は，正解率を改善させる方法として，既存のクラスタリングの結果を元に，NMFの初期値を設定させるという方法を提案し，実験を行い，その有効性について調べた．

実際にデータとして用意したNews，tr12，tr41の3種類を，本手法を含む5種類手法でクラスタリングを行い，その結果を比較する実験を行った．

本研究は，この実験から，本手法の正解率が，他の手法と比較しても高く維持され，また，ばらつきも少なくなっていたことから，この方法の有効性を確認することができ，目的を達成させることができた．

しかしながら，距離の計算方法を統一させ，また，k-meansの結果が良いものだけを採用したため，初期値による，正解率の安定については不明確となった．

よって，採用する初期値の内容や，初期値の正解率，または，他のクラスタリング手法結果を用いた時に，NMFのクラスタリングにどのような変化がでるか，また，距離計算として用いたユークリッド距離の計算以外の計算方法では結果はどう変わるのかについて調査する必要がある．

謝辞

本研究の遂行及び論文の作成に多大なご助言及び指導を賜った新納浩幸教官(茨城大学工学部システム工学科)と岩崎唯史教官(茨城大学工学部システム工学科)及び，佐々木実教官(茨城大学情報工学科)に深い感謝の意を表します．

最後に，本研究を進めるにあたり助言，協力を戴きました，同研究室の南慶典 氏(茨城大学工学部システム工学科５回生)，ならびに岩崎研究室の神津啓太 氏(茨城大学工学部システム工学科４回生)，佐々木研究室の加藤友宏 氏(茨城大学工学部情報工学科４回生)，鈴木朋央 氏(茨城大学工学部情報工学科５回生)，田島勇樹 氏(茨城大学工学部情報工学科４回生)にも深く感謝します．

参考文献

[1] 神嶌敏弘:データマイニング分野のクラスタリング手法(1) ークラスタリングを使ってみよう!ー", 人工知能学会誌, Vol.18, No.1, pp.59-65(2003).
[2]北研二, 津田和彦, 獅子堀正幹 : 情報検索アルゴリズム, 共立出版(2002).
[3] Wei Xu , Xin Liu , hong Gong : Document Clustering Based On Non-negative Matrix Factorization, SIGIR '03:Proceedings of the 26th Annual international ACM SIGIR Conference on Research and Development in information retrieval,pp.267-273,(2003).
[4]永田靖, 棟近雅彦 : 多変量解析入門, サイエンス社(2001).
[5]Daniel D.Lee , H.Sebastian Seung : Algorithms for Non-negative Matrix Factorization : Neural Information Proccessing Systems , volume 13 , pp556-562,(2001).

付録

NMF アルゴリズム
プログラム
(Java 言語)

```
//-----------------------------------------
//          NMF_main.java(プログラム動作メイン)
//-----------------------------------------
import java.util.*;
import java.io.*;

class Test_data{
          static void test(){
                    double testA[][]=new double[3][3] ;
                    double testB[][]=new double[3][3] ;
                    for(int i=0;i<testA.length;i++){
                              for(int j=0;j<testA[i].length;j++){
                                        testA[i][j]=i+j+1 ;
                              }
                    }
                    for(int i=0;i<testB.length;i++){
                              for(int j=0;j<testB[i].length;j++){
                                        testB[i][j]=i+j+1 ;
                              }
                    }
                    double testC[][]=new double[testA.length][testB[0].length];
                    testC=MatrixCalc.product(testA,testB);

                    for(int i=0;i<testA.length;i++){
                              for(int j=0;j<testA[i].length;j++){
                                        if(j!=0)System.out.print(",");
                                        System.out.print(""+testA[i][j]);
                              }
                              System.out.println();
                    }
                    System.out.println();
                    for(int i=0;i<testB.length;i++){
                              for(int j=0;j<testB[i].length;j++){
                                        if(j!=0)System.out.print(",");
                                        System.out.print(""+testB[i][j]);
                              }
                              System.out.println();
                    }
                    System.out.println();
                    for(int i=0;i<testC.length;i++){
```

```
            for(int j=0;j<testC[i].length;j++){
                        if(j!=0)System.out.print(",");
                        System.out.print(""+testC[i][j]);
            }
            System.out.println();
}
System.out.println();

NMF nmf=new NMF(3,30) ;
int check=nmf.euclidian_calc(testC);
if(check==-1)check=nmf.divergence_calc(testC);
testA=nmf.MatW ;
testB=nmf.MatH ;
System.out.println("check= "+check);

if(check==-1){
            System.out.println("err...");
            return ;
}

for(int i=0;i<testA.length;i++){
            for(int j=0;j<testA[i].length;j++){
                        if(j!=0)System.out.print(",");
                        System.out.print(""+testA[i][j]);
            }
            System.out.println();
}
System.out.println();
for(int i=0;i<testB.length;i++){
            for(int j=0;j<testB[i].length;j++){
                        if(j!=0)System.out.print(",");
                        System.out.print(""+testB[i][j]);
            }
            System.out.println();
}

System.out.println();
System.out.println("----------------");
System.out.println("check...");
```

```
                testC=MatrixCalc.product(testA, testB);

                for(int i=0;i<testC.length;i++){
                        for(int j=0;j<testC[i].length;j++){
                                if(j!=0)System.out.print(",");
                                System.out.print(""+testC[i][j]);
                        }
                        System.out.println();
                }
        }
}

public class NMF_main {
        static public void main(String args[]){
                //tests
                //Test_data.test();

                // datas
                Calendar c=Calendar.getInstance();

                Exe_data exe=new Exe_data(args[0]);
                exe.Tf_Idf();
                exe.Normalize();

                // type
                // 0:Random - Multiple type
                // 1:Random - Additional type
                // 2:Static - Multiple type
                // 3:Static - Additional type

                // NewsDocuments - class 5
                File f=new File(args[0]);
                String filename=f.getParent();
                filename+="¥¥NMF_class¥¥all2shote2_061120_";

                int filenumbers=5;
                for(int i=0;i<filenumbers;i++){
                        System.out.println("-----Random- Multiple type");
                        int check=exe.euclid_nmf(5,30);
                        if(check==-1)check=exe.divergence_nmf(5,30);
```

```
                    if(check!=-1){
                            exe.Print() ;
                            exe.Anser_print(args[2]) ;

        exe.Anser_save(filename+"nodata_nmf1_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH+1)+c.get(Calendar.DAY_
OF_MONTH)+"__"+i+".csv") ;
                    }

                    System.out.println("-----Random- Additional type") ;
                    check=exe.euclid_nmf(5,30,1);
                    if(check==-1)check=exe.divergence_nmf(5,30,1);
                    if(check!=-1){
                            exe.Print() ;
                            exe.Anser_print(args[2]) ;

        exe.Anser_save(filename+"nodata_nmf2_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH+1)+c.get(Calendar.DAY_
OF_MONTH)+"__"+i+".csv") ;
                    }

                            // MatV
                    System.out.println("-----AppointMatrix V - Multiple type") ;
                    check=exe.euclid_nmf(5,30,args[1]);
                    if(check==-1)check=exe.divergence_nmf(5,30,args[1]);
                    if(check!=-1){
                            exe.Print() ;
                            exe.Anser_print(args[2]) ;

        exe.Anser_save(filename+"kmeans_nmf1_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH)+c.get(Calendar.DAY_OF
_MONTH)+"__"+i+".csv") ;
                    }

                    System.out.println("-----AppointMatrix V - Additional type") ;
                    check=exe.euclid_nmf(5,30,args[1],1);
                    if(check==-1)check=exe.divergence_nmf(5,30,args[1],1);
                    if(check!=-1){
                            exe.Print() ;
                            exe.Anser_print(args[2]) ;

        exe.Anser_save(filename+"kmeans_nmf2_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH)+c.get(Calendar.DAY_OF
_MONTH)+"__"+i+".csv") ;
```

```
                    }
                }


                // tr12 - class 8
/*              File f=new File(args[0]) ;
                String filename=f.getParent() ;
                filename+="¥¥tr12_" ;

                int filenumbers=5 ;
                for(int i=0;i<filenumbers;i++){
                    System.out.println("-----Random- Multiple type") ;
                    int check=exe.euclid_nmf(8,30) ;
                    if(check==-1){check=exe.divergence_nmf(8,30);}
                    if(check!=-1){
                        exe.Print() ;
                        exe.Anser_print(args[2]) ;

        exe.Anser_save(filename+"nodata_nmf1_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH+1)+c.get(Calendar.DAY_
OF_MONTH)+"__"+i+".csv") ;
                    }

                    System.out.println("-----Random- Additional type") ;
                    check=exe.euclid_nmf(8,30,1) ;
                    if(check==-1){check=exe.divergence_nmf(8,30,1);}
                    if(check!=-1){
                        exe.Print() ;
                        exe.Anser_print(args[2]) ;

        exe.Anser_save(filename+"nodata_nmf2_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH+1)+c.get(Calendar.DAY_
OF_MONTH)+"__"+i+".csv") ;
                    }

                        // MatV
                    System.out.println("-----AppointMatrix V - Multiple type") ;
                    check=exe.euclid_nmf(8,30,args[1]) ;
                    if(check==-1){check=exe.divergence_nmf(8,30,args[1]);}
                    if(check!=-1){
                        exe.Print() ;
```

```
                                        exe.Anser_print(args[2]) ;

            exe.Anser_save(filename+"kmeans_nmf1_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH)+c.get(Calendar.DAY_OF
_MONTH)+"__"+i+".csv") ;
                                    }

                                    System.out.println("-----AppointMatrix V - Additional type") ;
                                    check=exe.euclid_nmf(8,30,args[1],1) ;
                                    if(check==-1){check=exe.divergence_nmf(8,30,args[1],1);}
                                    if(check!=-1){
                                            exe.Print() ;
                                            exe.Anser_print(args[2]) ;

            exe.Anser_save(filename+"kmeans_nmf2_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH)+c.get(Calendar.DAY_OF
_MONTH)+"__"+i+".csv") ;
                                    }
                            }
*/

                            // tr41 - class 10
/*                          File f=new File(args[0]) ;
                            String filename=f.getParent() ;
                            filename+="¥¥tr41_" ;

                            int filenumbers=5 ;
                            for(int i=0;i<filenumbers;i++){
//                                  System.out.println("-----Random- Multiple type") ;
//                                  int check=exe.euclid_nmf(10,30) ;
//                                  if(check==-1)check=exe.divergence_nmf(10,30);
//                                  if(check!=-1){
//                                          exe.Print() ;
//                                          exe.Anser_print(args[2]) ;
//
            exe.Anser_save(filename+"nodata_nmf1_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH+1)+c.get(Calendar.DAY_
OF_MONTH)+"__"+i+".csv") ;
//                                  }
//
//                                  System.out.println("------Random- Additional type") ;
//                                  check=exe.euclid_nmf(10,30,1) ;
//                                  if(check==-1)check=exe.divergence_nmf(10,30,1);
```

```
//                              if(check!=-1){
//                                      exe.Print() ;
//                                      exe.Anser_print(args[2]) ;
//
        exe.Anser_save(filename+"nodata_nmf2_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH+1)+c.get(Calendar.DAY_
OF_MONTH)+"__"+i+".csv") ;
//                              }

                                int check=0 ;
                                        // MatV
                                System.out.println("-----AppointMatrix V - Multiple type") ;
                                check=exe.euclid_nmf(10,30,args[1]);
                                if(check==-1)check=exe.divergence_nmf(10,30,args[1]);
                                if(check!= 1){
                                        exe.Print() ;
                                        exe.Anser_print(args[2]) ;

        exe.Anser_save(filename+"kmeans_nmf1_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH)+c.get(Calendar.DAY_OF
_MONTH)+"__"+i+".csv") ;
                                }

                                System.out.println("-----AppointMatrix V - Additional type") ;
                                check=exe.euclid_nmf(10,30,args[1],1);
                                if(check==-1)check=exe.divergence_nmf(10,30,args[1],1);
                                if(check!=-1){
                                        exe.Print() ;
                                        exe.Anser_print(args[2]) ;

        exe.Anser_save(filename+"kmeans_nmf2_ans_"+c.get(Calendar.YEAR)+c.get(Calendar.MONTH)+c.get(Calendar.DAY_OF
_MONTH)+"__"+i+".csv") ;
                                }
                        }
*/

                        //exe.Save(args[1]) ;
                }
}
```

```
//-----------------------------------------
//          NMF.java(NMFアルゴリズム)
//-----------------------------------------
public class NMF {
          int classes ;
          int renew ;
          double MatW[][] ;
          double MatH[][] ;
          public NMF(int classes){
                    this.classes=classes ;
                    this.rcnew=1 ;
          }
          public NMF(int classes, int renew){
                    this.classes=classes ;
                    this.renew=renew ;
          }
          int divergence_calc(doublesource[][]){
                    return this.divergence_calc(source, false);
          }
          int divergence_calc(doublesource[][], boolean sw){
                    return this.divergence_calc(source, MatrixCalc.Random(source[0].length, this.classes), sw)
          }
          int divergence_calc(doublesource[][], double MatH[][], boolean sw){
                    return this.divergence_calc(source, MatH, MatrixCalc.Random(source.length, this.classes), sw)
          }
          int divergence_calc(doublesource[][], double MatH[][], double MatW[][], boolean sw){

                    // MatH(Matrix V)は結果表示がしやすいように transaction状態で計算

                    // Kullback-Leibler divergence より最急降下法による計算

                    // format Matrix and more there...
                    double checkW, checkH ;
                    double WH[][], SumW[], SumH[], SumWV, SumHV;
                    WH=new double[MatW.length][MatH.length];
                    SumW=new double[this.classes] ;
                    SumH=new double[this.classes] ;
                    SumWV=0.0 ;
                    SumHV=0.0 ;
```

```
for(int replace=0;replace<this.renew;replace++){
        System.out.println("Now...RenewCount..."+(replace+1));

        // MatW * MatH
        WH=MatrixCalc.product(MatW,MatrixCalc.transaction(MatH));

        // SumW , SumH
        for(int k=0;k<this.classes;k++){
                SumW[k]=0.0 ;
                SumH[k]=0.0 ;
                for(int j=0;j<MatW.length;j++)SumW[k]+=MatW[j][k];
                for(int j=0;j<MatH.length;j++)SumH[k]+=MatH[j][k];
        }

        double old_dataW=MatrixCalc.ElementAverage(MatW);
        double old_dataH=MatrixCalc.ElementAverage(MatH);

        // renew process
        // MatW renew
        for(int i=0;i<MatW.length;i++){
                for(int j=0;j<MatW[i].length;j++){
                        for(int k=0;k<MatH.length;k++){
                                SumHV+=(MatH[k][j]*source[i][k]/WH[i][k]);
                        }
                        MatW[i][j]*=(SumHV/SumH[j]) ;
                }
        }
        // MatH renew
        for(int i=0;i<MatH.length;i++){
                for(int j=0;j<MatH[i].length;j++){
                        for(int k=0;k<MatW.length;k++){
                                SumWV+=(MatW[k][j]*source[k][i]/WH[k][i]);
                        }
                        if(sw){
                                // additive

MatH[i][j]+=(MatH[i][j]/SumW[j])*Math.abs(SumWV-SumW[j]);
                        }
                        else{
                                // multiple
```

```
                                        MatH[i][j]*=(SumWV/SumW[j]);
                                }
                        }
                }

                // Normarize
                double SumU[]=new double[this.classes];
                for(int j=0;j<this.classes;j++){
                        for(int i=0;i<MatW.length;i++){
                                SumU[j]+=Math.pow(MatW[i][j],2.0);
                        }
                        SumU[j]=Math.sqrt(SumU[j]);
                }
                for(int i=0;i<MatW.length;i++){
                        for(int j=0;j<MatW[i].length;j++){
                                MatW[i][j]/=SumU[j];
                        }
                }
                for(int i=0;i<MatH.length;i++){
                        for(int j=0;j<MatH[i].length;j++){
                                MatH[i][j]*=SumU[j];
                        }
                }

                double tempW=MatrixCalc.ElementAverage(MatW);
                double tempH=MatrixCalc.ElementAverage(MatH);
                checkW=tempW-old_dataW;
                checkH=tempH-old_dataH;
                // 全要素平均の変動量表示
                System.out.println("Ave.MatW -> "+tempW);
                System.out.println("Ave.MatH -> "+tempH);
                System.out.println("SubtractW -> "+checkW);
                System.out.println("SubtractH -> "+checkH);

                checkW=Math.abs(checkW);
                checkH=Math.abs(checkH);

                if((checkW<1.0e-3)&(checkH<1.0e-3))break;
```

```
                if(Double.isNaN(tempW)|Double.isNaN(tempH))return-1 ;
                if(Double.isInfinite(tempW)|Double.isInfinite(tempH))return-1 ;
        }

        this.MatW=MatW ;
        this.MatH=MatH ;

        return 0 ;
}
int euclidian_calc(double source[][]){
        return this.euclidian_calc(source, false);
}
int euclidian_calc(double source[][], boolean sw){
        return this.euclidian_calc(source, MatrixCalc.Random(source[0].length, this.classes), sw);
}
int euclidian_calc(double source[][], double MatH[][], boolean sw){
        return this.euclidian_calc(source, MatH, MatrixCalc.Random(source.length, this.classes), sw);
}
int euclidian_calc(double source[][], double MatH[][], double MatW[][], boolean sw){

        // MatH(Matrix V)は結果表示がしやすいように transaction状態で計算

        // format Matrix and more there...
        double checkW, checkH ;
        double XV[][], UVV[][], XU[][], VUU[][];
        XV=new double[MatW.length][this.classes];
        UVV=new double[MatW.length][this.classes];
        XU=new double[MatH.length][this.classes];
        VUU=new double[MatH.length][this.classes];

        for(int replace=0;replace<this.renew;replace++){
                System.out.println("Now...RenewCount..."+(replace+1));

                // X*V
                XV=MatrixCalc.product(source, MatH);

                // U*V*V
                UVV=MatrixCalc.product(MatW, MatrixCalc.transaction(MatH));
                UVV=MatrixCalc.product(UVV, MatH);
```

```
                                        // X*U
                                        XU=MatrixCalc.product(MatrixCalc.transaction(source),MatW);

                                        // V*U*U
                                        VUU=MatrixCalc.product(MatH,MatrixCalc.transaction(MatW));
                                        VUU=MatrixCalc.product(VUU,MatW);


                                        // Debug
                                        for(int i=0;i<MatW.length;i++){
//                                                  for(int j=0;j<MatW[i].length;j++){
//                                                          if(Double.isNaN(XV[i][j]))System.out.println("XV"+i+",
                                                          if(Double.isNaN(UVV[i][j]))System.out.println("UVV"+i+",
//"+j+" - NaN");
//                                                          if(Double.isNaN(UVV[i][j]))System.out.println("UVV"+i+",
//"+j+" - NaN");
//                                                          if(XV[i][j]==0.0)System.out.println("XV"+i+", "+j+" - 0")
//;
//                                                          if(UVV[i][j]==0.0)System.out.println("UVV"+i+", "+j+" -
//0");
//                                                  }
//                                          }
//                                          for(int i=0;i<MatH.length;i++){
//                                                  for(int j=0;j<MatH[i].length;j++){
//                                                          if(Double.isNaN(XU[i][j]))System.out.println("XU"+i+",
//"+j+" - NaN");
                                                          if(Double.isNaN(VUU[i][j]))System.out.println("VUU"+i+",
//"+j+" - NaN");
//                                                          if(XU[i][j]==0.0)System.out.println("XU"+i+", "+j+" - 0")
//;
//                                                          if(VUU[i][j]==0.0)System.out.println("VUU"+i+", "+j+" -
//0");
//                                                  }
//                                          }


                                        double old_dataW=MatrixCalc.ElementAverage(MatW);
                                        double old_dataH=MatrixCalc.ElementAverage(MatH);

                                        // renew process
                                        // MatW renew
```

```
for(int i=0;i<MatW.length;i++){
        for(int j=0;j<MatW[i].length;j++){
                MatW[i][j]*=(XV[i][j]/UVV[i][j]);
        }
}
// MatH renew
for(int i=0;i<MatH.length;i++){
        for(int j=0;j<MatH[i].length;j++){
                if(sw){
                        // additive
                        double cta=MatH[i][j]/VUU[i][j];
                        MatH[i][j]+=(eta*Math.abs(XU[i][j]-VUU[i][j]));
                }
                else{
                        // multiple
                        MatH[i][j]*=(XU[i][j]/VUU[i][j]);
                }
        }
}


// Debug
for(int i=0;i<MatW.length;i++){
        for(int j=0;j<MatW[i].length;j++){
                if(Double.isNaN(MatW[i][j]))System.out.println("MatW+i+" ,
"+j+" - NaN") ;
        }
}
for(int i=0;i<MatH.length;i++){
        for(int j=0;j<MatH[i].length;j++){
                if(Double.isNaN(MatH[i][j]))System.out.println("MatH+i+" ,
"+j+" - NaN") ;
        }
}

// Normarize
double SumW[]=new double[this.classes];
for(int j=0;j<this.classes;j++){
        for(int i=0;i<MatW.length;i++){
                // Debug
```

```
//                                      if(Double.isNaN(MatW[i][j])){
//                                              continue;
//                                      }
                                        SumW[j]+=Math.pow(MatW[i][j],2.0);
                                }
                                SumW[j]=Math.sqrt(SumW[j]);
                        }
                        for(int i=0;i<MatW.length;i++){
                                for(int j=0;j<MatW[i].length;j++){
                                        MatW[i][j]/=SumW[j];
                                }
                        }
                        for(int i=0;i<MatH.length;i++){
                                for(int j=0;j<MatH[i].length;j++){
                                        MatH[i][j]*=SumW[j];
                                }
                        }


                        double tempW=MatrixCalc.ElementAverage(MatW);
                        double tempH=MatrixCalc.ElementAverage(MatH);
                        checkW=tempW-old_dataW;
                        checkH=tempH-old_dataH;
                        // 全要素平均の変動量表示
                        System.out.println("Ave.MatW -> "+tempW);
                        System.out.println("Ave.MatH -> "+tempH);
                        System.out.println("SubtractW -> "+checkW);
                        System.out.println("SubtractH -> "+checkH);


                        checkW=Math.abs(checkW);
                        checkH=Math.abs(checkH);

                        if((checkW<1.0e-3)&(checkH<1.0e-3))break;
                        if(Double.isNaN(tempW)|Double.isNaN(tempH))return-1;
                        if(Double.isInfinite(tempW)|Double.isInfinite(tempH))return1;
                }


                this.MatW=MatW;
```

```
                this.MatH=MatH ;

                return 0 ;
        }
}

//---------------------------------------
//      MatrixCalc.java(行列計算プログラム)
//---------------------------------------
public class MatrixCalc {
        static double[][] product(double[][] MatA, double[][]MatB){
                if(MatA[0].length!=MatB.length){
                        double result[][]=new double[1][1] ;
                        result[0][0]= 0.0 ;
                        return result ;
                }

                double result[][]=new double[MatA.length][MatB[0].length];
                for(int i=0;i<result.length;i++){
                        for(int j=0;j<result[i].length;j++)result[i][j]=0.0;
                }

                for(int i=0;i<result.length;i++){
                        for(int j=0;j<result[i].length;j++){
                                double ret=0.0 ;
                                for(int k=0;k<MatA[i].length;k++){
                                        ret+=MatA[i][k]*MatB[k][j] ;
                                }
                                result[i][j]=ret ;
                        }
                }

                return result ;
        }
        static double[][] sum(double[][] MatA, double[][]MatB){
                if((MatA.length!=MatB.length)|(MatA[0].length!=MatB[0].length)){
                        double result[][]=new double[1][1] ;
                        result[0][0]=-0.0 ;
                        return result ;
                }
```

```
		double result[][]=new double[MatA.length][MatB[0].length];
		for(int i=0;i<result.length;i++){
			for(int j=0;j<result[i].length;j++)result[i][j]=0.0;
		}

		for(int i=0;i<result.length;i++){
			for(int j=0;j<result[i].length;j++){
				result[i][j]=MatA[i][j]+MatB[j][i];
			}
		}

		return result ;
	}
	static double[][] multiple(double mul,double Mat[][]){
		for(int i=0;i<Mat.length;i++){
			for(int j=0;j<Mat[i].length;j++){
				Mat[i][j]*=mul ;
			}
		}
		return Mat ;
	}
	static double[][] transaction(double Mat[][]){
		double result[][]=new double[Mat[0].length][Mat.length];
		for(int i=0;i<result.length;i++){
			for(int j=0;j<result[i].length;j++)result[i][j]=Mat[j][i];
		}
		return result ;
	}
	static double[][] Random(int column,int row){
		double result[][]=new double[column][row];

		for(int i=0;i<result.length;i++){
			for(int j=0;j<result[i].length;j++){
				result[i][j]=Math.random();
			}
		}
		return result ;
	}
	static double[][] allElementOne(int column,int row){
```

```
                double result[][]=new double[column][row];

                for(int i=0;i<result.length;i++){
                        for(int j=0;j<result[i].length;j++){
                                result[i][j]=1.0/(double)result[i].length;
                        }
                }
                return result ;
        }
        static double EllementAverage(double[][]Mat){
                double result=0.0 ;
                for(int i=0;i<Mat.length;i++){
                        for(int j=0;j<Mat[i].length;j++){
                                result+=Mat[i][j] ;
                        }
                }
                result=result/(double)(Mat.length*Mat[0].length);
                return result ;
        }
}
//---------------------------------------
//      Exe_data.java(tf-id計算及びNMFプログラム呼び出し)
//---------------------------------------
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.StringTokenizer;


class Check{
        String TrueClass[] ;
        String classes ;
        String Savedata ;
        int c_num ;
        public Check(String file){
                try{
                        String data="" ;
                        FileReader fr=new FileReader(file) ;
                        BufferedReader br=new BufferedReader(fr) ;
```

```
                String temp="" ;
                this.classes="" ; boolean sw=false ;
                while((temp=br.readLine())!=null){
                        if(!data.equals(""))data+=",";
                        data+=temp ;
                        sw=true ;
                        StringTokenizer t_st=new StringTokenizer(this.classes,",");
                        while(t_st.hasMoreTokens()){
                                String numb=t_st.nextToken() ;
                                if(temp.equals(numb))sw=false;
                        }
                        if(sw){
                                if(!this.classes.equals(""))this.classes+=",";
                                this.classes=temp;
                        }
                }
                fr.close() ;
                StringTokenizer st=new StringTokenizer(data,",");
                this.TrueClass=new String[st.countTokens()];
                for(int i=0;i<this.TrueClass.length;i++){
                        this.TrueClass[i]="" ;
                }
                for(int i=0;st.hasMoreTokens();i++){
                        this.TrueClass[i]=st.nextToken();
                }
        }
        catch(Exception e){
                e.printStackTrace() ;
        }
}

void compare(double result[][]){
        StringTokenizer st=new StringTokenizer(this.classes,",");
        int classNum=st.countTokens() ;
        this.c_num=st.countTokens() ;
        String classes[]=new String[st.countTokens()];
        int list[]=new int[classNum*classNum];
        for(int i=0;st.hasMoreTokens();i++){
                classes[i]="" ;
                classes[i]=st.nextToken();
```

```
}

int res_class[]=new int[result.length];
for(int i=0;i<res_class.length;i++){
        double Max=0.0;
        int number=0;
        for(int j=0;j<classNum;j++){
                if(Max<result[i][j]){
                        Max=result[i][j];
                        number=j;
                }
        }
        res_class[i]=number;
}

for(int i=0;i<this.TrueClass.length;i++){
        int number=0;
        for(int j=0;j<classNum;j++){
                if(this.TrueClass[i].equals(classes[j]))number=j;
        }
        list[(number*classNum)+res_class[i]]++;
}

this.Savedata="";

System.out.print("class"); this.Savedata="class";
for(int i=0;i<classNum;i++){
        System.out.print(","+i);
        this.Savedata+=","+i;
}
System.out.println(); this.Savedata+="¥n";
StringTokenizer test=new StringTokenizer(this.classes,",");
for(int i=0;i<classNum;i++){
        String temp="" ; temp=test.nextToken();
        System.out.print(temp);
        this.Savedata+=temp;
        for(int j=0;j<classNum;j++){
                System.out.print(",");
                System.out.print(list[j+(i*classNum)]);
                this.Savedata+=","+(list[j+(i*classNum)]);
```

```
                        }
                        System.out.println();
                        this.Savedata+="¥n";
                }
        }
        void save(String file){
                try{
                        FileWriter fw=new FileWriter(file);
                        BufferedWriter bw=new BufferedWriter(fw);
                        StringTokenizer st=new StringTokenizer(this.Savedata,"¥n");
                        while(st.hasMorcTokens()){
                                String temp="" ; temp=st.nextToken();
                                bw.write(temp);
                                bw.newLine();
                        }
                        bw.flush();
                        fw.close();
                }
                catch(Exception e){
                        e.printStackTrace();
                }
        }
}

public class Exe_data{
        double data[][];
        double result[][];
        String anser ;
        public Exe_data(String file){
                try{
                        FileReader fr=new FileReader(file);
                        BufferedReader br=new BufferedReader(fr);
                        String temp="" ;
                        // 最初の行読み込み
                        temp=br.readLine();
                        StringTokenizer st=new StringTokenizer(temp,"") ;
                        //列数
                        temp=st.nextToken();
                        int row=Integer.valueOf(temp).intValue();
                        //行数
```

```
                    temp=st.nextToken() ;
                    int column=Integer.valueOf(temp).intValue();

                    this.data=new double[column][row] ;

                    for(int i=0;i<this.data.length;i++){
                            for(int j=0;j<this.data[i].length;j++)this.data[i][j]=0.0
                    }

                    // 行列の要素入れ込み
                    for(int i=0;(temp=br.readLine())!=null;i++){
                            st=new StringTokenizer(temp,"") ;
                            while(st.hasMoreTokens()){
                                    temp=st.nextToken() ;
                                    int number=Integer.valueOf(temp).intValue();
                                    number-- ;
                                    temp=st.nextToken() ;
                                    this.data[number][i]=Double.valueOf(temp).doubleValue();
                            }
                    }
                    /*
                    for(int i=0;i<this.data.length;i++){
                            for(int j=0;j<this.data[i].length;j++){
                                    if(j!=0)System.out.print(",");
                                    System.out.print(this.data[i][j]);
                            }
                            System.out.println() ;
                    }*/

                    System.out.println("FinishedLoading...") ;

                    fr.close() ;
            }
            catch(Exception e){
                    e.printStackTrace() ;
            }
    }
    void Tf_Idf(){
            double wor[]=new double[this.data.length] ;
            for(int i=0;i<this.data.length;i++){
```

```
                int sum=0 ;
                for(int j=0;j<this.data[i].length;j++){
                        if(this.data[i][j]!=0.0)sum++;
                        //TF        :log(1+i,j番目の頻度数)
                        this.data[i][j]=Math.log(1.0+this.data[i][j]);
//                      this.data[i][j]=Math.log10(1.0+this.data[i][j]);
                }
                if(sum==0)continue;
                //IDF       :log((N/文書頻度数)+1)
                wor[i]=Math.log((this.data[i].length/(double)sum)+1.0);
//              wor[i]=Math.log10((this.data[i].length/(double)sum)+1.0);
        }
        for(int i=0;i<this.data.length;i++){
                for(int j=0;j<this.data[i].length;j++){
                        this.data[i][j]=this.data[i][j]*wor[i];
                }
        }
        System.out.println("FinishedTf-Idf Calclating...");
        return ;
}
void Normalize(){
        for(int j=0;j<this.data[0].length;j++){
                double temp=0.0,nor ;
                for(int i=0;i<this.data.length;i++){
                        temp+=(Math.pow(this.data[i][j],2.0));
                }
                nor=Math.sqrt(temp) ;
                for(int i=0;i<this.data.length;i++){
                        this.data[i][j]/=nor ;
                }
        }
        System.out.println("FinishedNormalize Calclating...");
        return ;
}
int euclid_nmf(int classes,int renew,String dataV){
        return this.euclid_nmf(classes, renew, dataV,0) ;
}
int euclid_nmf(int classes,int renew,String dataV,int type){
        double MatV[][]=new double[this.data[0].length][classes];
        try{
```

```
                    FileReader fr=new FileReader(dataV) ;
                    BufferedReader br=new BufferedReader(fr) ;
                    String temp="" ;
                    for(int i=0;(temp=br.readLine())!=null;i++){
                            StringTokenizer st=new StringTokenizer(temp,",");
                            for(int j=0;st.hasMoreTokens();j++){
                                    temp=st.nextToken() ;
                                    MatV[i][j]=Double.valueOf(temp).doubleValue();
                            }
                    }
                    fr.close() ;
            }
            catch(Exception e){
                    e.printStackTrace() ;
            }
            NMF nmf=new NMF(classes,renew) ;
            int check=0 ;
            switch(type){
            case(0) :
                    check=nmf.euclidian_calc(this.data,MatV,false);
            break ;
            case(1) :
                    check=nmf.euclidian_calc(this.data,MatV,true);
            break ;
            case(2) :

check=nmf.euclidian_calc(this.data,MatrixCalc.allElementOne(this.data[0].length,classes),false)
                    break ;
                    case(3) :

check=nmf.euclidian_calc(this.data,MatrixCalc.allElementOne(this.data[0].length,classes),true)
                    break ;
                    }
                    if(check==-1){
                            System.out.println("NMFeuclidean distance calclating error...") ;
                            return -1 ;
                    }
                    this.result=nmf.MatH;
                    return 0 ;
}
```

```
int euclid_nmf(int classes, int renew){
        return this.euclid_nmf(classes, renew,0) ;
}

int euclid_nmf(int classes, int renew,int type){
        NMF nmf=new NMF(classes,renew) ;
        int check=0 ;
        switch(type){
        case(0) :
                check=nmf.euclidian_calc(this.data);
        brcak ;
        case(1) :
                check=nmf.euclidian_calc(this.data,true);
        break ;
        case(2) :

check=nmf.euclidian_calc(this.data,MatrixCalc.allElementOne(this.data[0].length,classes),false)
        break ;
        case(3) :

check=nmf.euclidian_calc(this.data,MatrixCalc.allElementOne(this.data[0].length,classes),true)
        break ;
        }
        if(check==-1){
                System.out.println("NMFeuclidean distance calclating error...") ;
                return -1 ;
        }
        this.result=nmf.MatH;
        return 0 ;
}
int divergence_nmf(int classes, int renew,String dataV){
        return this.divergence_nmf(classes, renew,dataV, 0) ;
}
int divergence_nmf(int classes, int renew,String dataV,int type){
        double MatV[][]=new double[this.data[0].length][classes];
        try{
                FileReader fr=new FileReader(dataV) ;
                BufferedReader br=new BufferedReader(fr) ;
                String temp="" ;
                for(int i=0;(temp=br.readLine())!=null;i++){
```

```
                                StringTokenizer st=new StringTokenizer(temp,",");
                                for(int j=0;st.hasMoreTokens();j++){
                                        temp=st.nextToken();
                                        MatV[i][j]=Double.valueOf(temp).doubleValue();
                                }
                        }
                        fr.close();
                }
                catch(Exception e){
                        e.printStackTrace();
                }
                NMF nmf=new NMF(classes,renew);
                int check=0;
                switch(type){
                case(0):
                        check=nmf.divergence_calc(this.data,MatV,false);
                break;
                case(1):
                        check=nmf.divergence_calc(this.data,MatV,true);
                break;
                case(2):

check=nmf.divergence_calc(this.data,MatrixCalc.allElementOne(this.data[0].length,classes),false)
                break;
                case(3):

check=nmf.divergence_calc(this.data,MatrixCalc.allElementOne(this.data[0].length,classes),true)
                break;
                }
                if(check==-1){
                        System.out.println("NMFdivergence distance calclating error...");
                        return -1;
                }
                this.result=nmf.MatH;
                return 0;
}
int divergence_nmf(int classes,int renew){
                return this.divergence_nmf(classes,renew,0);
}
int divergence_nmf(int classes,int renew,int type){
```

```
            NMF nmf=new NMF(classes,renew) ;
            int check=0 ;
            switch(type){
            case(0) :
                        check=nmf.divergence_calc(this.data);
            break ;
            case(1) :
                        check=nmf.divergence_calc(this.data,true);
            break ;
            case(2) :

check=nmf.divergence_calc(this.data,MatrixCalc.allElementOne(this.data[0].length,classes),false)
            break ;
            case(3) :

check=nmf.divergence_calc(this.data,MatrixCalc.allElementOne(this.data[0].length,classes),true)
            break ;
            }
            if(check==-1){
                        System.out.println("NMFdivergence distance calclating error...") ;
                        return -1 ;
            }
            this.result=nmf.MatH;
            return 0 ;
}
void Print(){
            for(int i=0;i<this.result.length;i++){
                        for(int j=0;j<this.result[i].length;j++){
                                    if(j!=0)System.out.print(",");
                                    System.out.print(""+this.result[i][j]);
                        }
                        System.out.println() ;
            }
            return ;
}
void Save(String file){
            try{
                        FileWriter fw=new FileWriter(file) ;
                        BufferedWriter bw=new BufferedWriter(fw) ;
                        for(int i=0;i<this.result.length;i++){
```

```
                                        for(int j=0;j<this.result[i].length;j++){
                                                if(j!=0)bw.write(",");
                                                bw.write(""+this.result[i][j]);
                                        }
                                        bw.newLine();
                                }
                                bw.flush();
                                fw.close();
                        }
                        catch(Exception e){
                                e.printStackTracc();
                        }
                        System.out.println("FinishedFile Writing ...");
                }
                void Anser_print(String file){
                        Check check=new Check(file);
                        check.compare(this.result);
                        this.anser=check.Savedata;
                }
                void Anser_save(String file){
                        try{
                                FileWriter fw=new FileWriter(file);
                                BufferedWriter bw=new BufferedWriter(fw);
                                StringTokenizer st=new StringTokenizer(this.anser,"¥n");
                                while(st.hasMoreTokens()){
                                        String temp="";temp=st.nextToken();
                                        bw.write(temp);
                                        bw.newLine();
                                }
                                bw.flush();
                                fw.close();
                                System.out.println("FinishedSaving file - "+file);
                        }
                        catch(Exception e){
                                e.printStackTrace();
                        }
                }
        }
```