pLSI を用いた文書クラスタリングにおける初期値設定手法

執筆者：阿部 竜之介

指導教官：新納 浩幸

平成 20 年 2 月 28 日

目次

第1章　はじめに

1.1 研究概要

近年、デジタル化されたテキストデータが増えたことで、それらを計算機上で扱う自然言語処理技術が実用化されてきている。人間が膨大な情報群から必要な情報を取り出し、利用することは困難である。そのため、テキストマイニング技術などのテキストデータから人間にとって利用性のある知識を効率よく獲得する技術の研究がなされている。

テキストマイニングの要素技術である文書クラスタリングは、文書集合をトピックの類似性にもとづいてグループ分け（分割）することである。文書クラスタリングは長年にわたって研究が行われてきており、検索性能の向上や検索支援のために用いられてきた。

本研究では、確率モデルに基づいた文書単語行列の次元圧縮法—pLSI（Probabilistic Latent Semantic Indexing）による文書クラスタリングを扱う。pLSIは、次元を減らす計算にEMアルゴリズムが用いられており、その初期値は設定する必要がある。pLSIによるクラスタリングでは、この初期値に依存してクラスタリングの結果が異なるという問題がある。そこで、本研究ではクラスタリング結果がより良くなるような初期値の設定手法を提案し、実験を通してその効果を確認する。

1.2 本論文の構成

第2章では文書クラスタリングについて述べる。ここではクラスタリング手法の類型や特徴を紹介する。

第3章ではpLSIについての説明をする。pLSIによる文書クラスタリングの特徴とそのクラスタリング方法、EMアルゴリズムについて解説を行う。

第4章では初期値設定手法について述べる。従来の手法と提案する手法それぞれの設定方法と特徴を述べる。

第5章では本研究で行った実験について述べる。従来の手法と提案する手法とで実際にクラスタリングを行う。実験結果として、エントロピーと純度を示す。

第6章では実験結果の比較、提案する手法の利点などについて考察を行う。

第2章　文書クラスタリング

クラスタリングとは異質なもののまざり合っている対象の中で、互いに似たものを集めて集合（クラスタ）をつくり、対象を分類しようという方法を総称したもので、数値分類法（numerical classification, numerical taxonomy）とも呼ばれる。

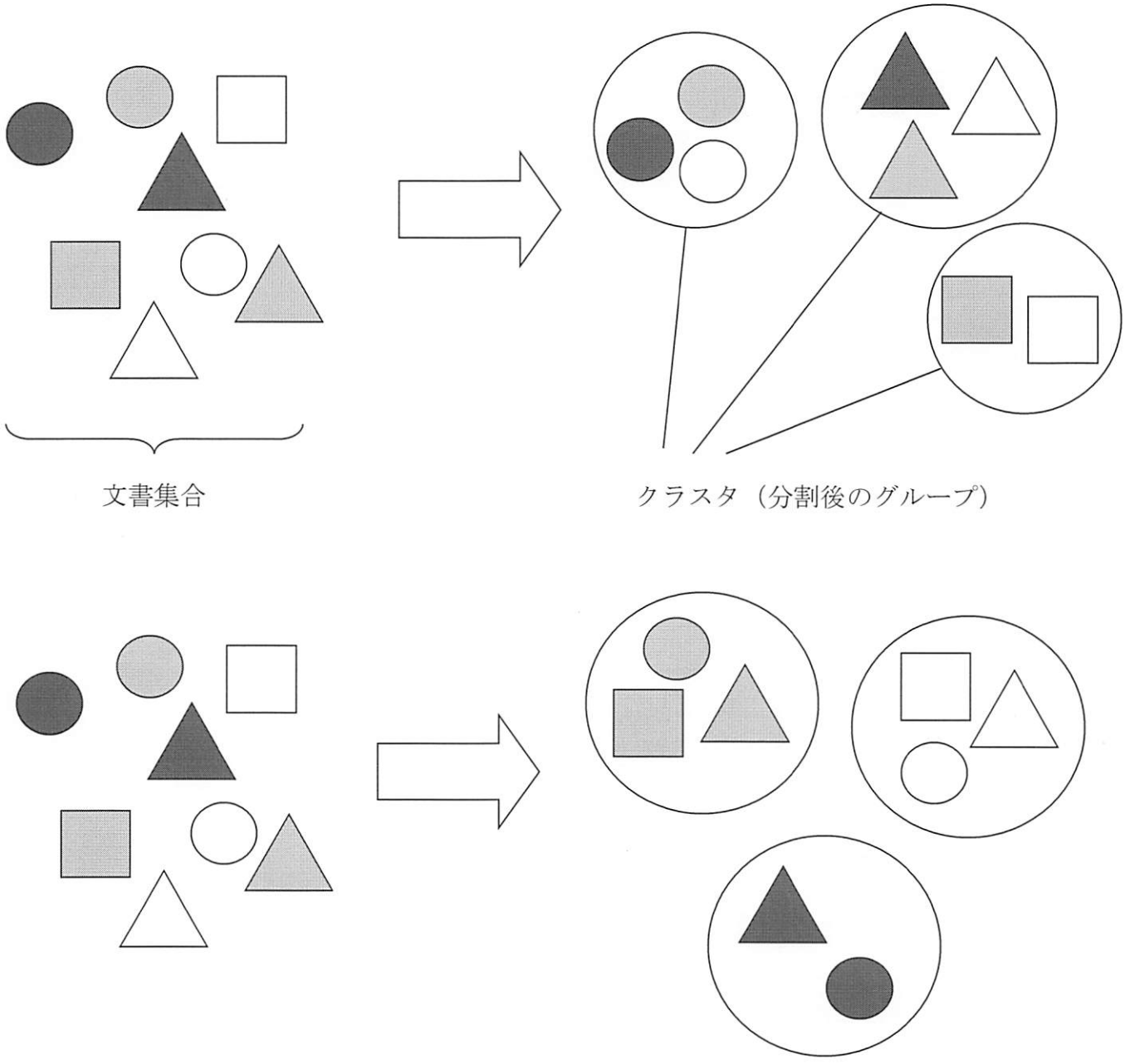

図.1　文書クラスタリング

情報検索の分野では、図書や雑誌論文などの文書の集合を内容的に均質ないくつかの群に分けるための、文書クラスタリングの研究が、長年にわたって試みられてきた。その応用目的としては以下のものが挙げられる。

1. 従来の情報検索にクラスタリングの結果を直接適用することによって検索性能を向上させる
2. 検索結果として文書集合をグループ化してわかりやすく提示する
3. キーワード検索とは異なった、ブラウジングに基づく検索様式を提供する

ここでは文書クラスタリングの特徴と類型について紹介する。

文書クラスタリングの一般的特徴

文書クラスタリングを実行する場合、一般に、文書は各語の重みから構成されるベクトル

$$\mathrm{d_i} = (t_{i1}, t_{i2}, \ldots, t_{iM})^T$$

として表現される。ここで t_{ij} は、i 番目の文書における単語 w_j の重みである $(j = 1, \ldots, M)$。なお、M は文書集合に含まれる語の異なり数である。ベクトル空間モデルに基づけば、2件の文書間の類似度は、それらのベクトルの成す角度の余弦として定義される。その類似度に基づいて、クラスタリングを実行できる。このように余弦係数に基づいて計算されることが多い。

また、クラスタリングの対象数（文書数）N がかなり大きいこと、特徴の数（異なり語数、単語数）M がクラスタリングの対象数よりも大きくなることなどが特徴として挙げられる。このような特徴のために、文書クラスタリングは一種独特な工夫が必要となる。

文書クラスタリング技法の類型

文書クラスタリングの方法は以下のように類型化される。

1. 非階層的クラスタ分析法
2. 階層的クラスタ分析法
3. 次元縮約法を利用した方法
4. 確率モデルに基づく方法

上記1.～4.の他に、グラフ理論に基づく方法なども挙げられるがここでは省略する。

非階層的クラスタ分析法

情報検索の分野では、早い時期から非階層的な方法の適用が探求されてきた。この方法は、文書が記録されたファイルを1度走査するだけでクラスタを構成しようと試みるので、結果、その計算量が抑えられ、大規模文書集合のクラスタリングに適しているといえる。非階層的クラスタ分析法の代表例としてk-means法が挙げられる。

一般的な k-means アルゴリズム

1. クラスタの個数を決め、初期的なクラスタのベクトルを生成する
2. 分類対象を、それぞれ、最も近いベクトルにしたがって分類し、ベクトルを更新する
3. もしベクトルが変化しなければ処理を終了し、そうでなければ2.に戻る

階層的クラスタ分析法

階層的なクラスタリングには、凝集型と分割型とがある。凝集型として単連結法などの階層的クラスタ分析法があり、これは個々の対象（文書）から出発し、類似度行列を用いて、それらを次第に大きなクラスタに組み上げていく。一方、分割型の場合は、全文書集合から出発して、その分割を再帰的に繰り返すことによって、階層を構成する。

階層的な方法では非階層的な方法とは異なり、クラスタが上位・下位に構造化されるわけであり、これは、情報の組織化・検索の観点からはより望ましいと考えられる。

階層的クラスタ分析法を大規模な文書集合に適用する場合の最大の問題点はその計算量にある。また、類似度行列が大きければ、それを保存する領域についても注意を払う必要

がある。一般的には、階層的クラスタ分析法のアルゴリズムの研究において、例えばSLINKなどの効率的な方法も考案されている。

次元縮約法を利用した方法

行列に対して主成分分析のような次元縮約の方法を適用して、その結果から文書をクラスタに分割する。情報検索の分野では、LSI (Latent Semantic Indexing) において特異値分解 (Singular Value Decomposition: SVD) を利用した次元縮約の方法が利用されている。

具体的に語×文書の行列 W^T に対する特異値分解は

$$\mathrm{W}^T = \mathrm{UQV}^T$$

と書ける（ただし、$M > N$ を仮定)。ここで、U は$M \times N$ の直行行列、Q は$N \times N$ の対角行列、V は$N \times N$ の直行行列である。ただし、Wのランクを$r(\leq N)$ とすると、Q における $N-r$ 個の対角要素は0である。ここで、Q における0でない r 個の対角要素に対応するVの列ベクトルを取り出して、

$$\mathrm{v}_{(1)}, \mathrm{v}_{(2)}, \cdots, \mathrm{v}_{(r)}$$

と表記する。ここで、各$\mathrm{v}_{(k)}$ はN次元ベクトルであり $(k = 1,\ldots,r)$ 、SVDによって抽出された　個の次元に対する各文書の値を示している。したがって、ある閾値を超えた文書のみがその次元に属すると仮定すれば、本来の語数Mよりも少ない r 次元での文書ベクトルを構成することができ、これを一種の特徴抽出として利用する。

確率モデルに基づく方法

1つの文書ベクトルが与えられたときのクラスタの確率を求めることによって、文書集合をクラスタに分割する。ただし、確率分布のパラメータを推定するために、EMアルゴリズムなどの近似的な方法を使わざるを得ないことや、文書集合における語の分布を、数学的に扱いやすい確率分布で近似しなければならないなどの問題がある。

第3章　pLSI

3.1 pLSIによる文書クラスタリング

pLSIは、確率の尤度を最大にするような次元圧縮を行う手法である。pLSIはAspectモデルと呼ばれるモデルを利用している。Aspectモデルとは文書と単語を結びつける潜在的なクラスを想定したモデルであり、文書d と単語wの出現を潜在的なクラスz を用いて以下のようにモデル化する。

$$P(w|d)=\sum_{z}P(w|z)p(z|d)$$

ベイズの定理から、

$$P(z|d)=\frac{P(z)P(d|z)}{P(d)}$$

であり、$P(d,w)=P(d)P(w|d)$ なので、

$$P(d,w)=P(d)\sum_{z}P(w|z)\frac{P(z)P(d|z)}{P(d)}=\sum_{z}P(z)P(w|z)P(d|z)$$

となる。

K 次に次元縮約する場合は、潜在的クラスを K 個設定する。

$$z_1,z_2,\cdots,z_K$$

データ（文書）d に対して、

$$\left(P(z_1,d),P(z_2,d),\cdots;P(z_K,d)\right)$$

が縮約されたベクトルとなる。

また、潜在的なクラスをそのままクラスタリングにおけるクラスタだと捉えれば、次元縮約した結果自体がクラスタリングを表す。

クラスタリング手順

1. クラスタの数Kと初期値を設定する
2. 尤度を最大にするようにパラメータを学習させる
3. 学習後のパラメータを元に文書をクラスタに分割する

1. クラスタの数Kと初期値を設定する

クラスタの数Kは任意に設定する。

ここで設定する初期値は$P(z)$、$P(w|z)$、$P(d|z)$の3つのパラメータである。それぞれクラスタzの現れる確率、単語wがクラスタzの中で現れる確率、文書dがクラスタzの中で現れる確率を意味する。このパラメータを何らかの方法で決定する。この初期値の設定方法が本研究の主題である。

2. 尤度を最大にするようにパラメータを学習させる

EMアルゴリズムを用いて、最初に設定した3つのパラメータを学習させる。EMアルゴリズムはEステップとMステップの処理を繰り返し行うもので、詳細は後述する。尤度が最大になるようにこの処理を繰り返す。今、文書dに含まれている単語wの個数を$n(d,w)$で表すと、尤度Lは以下の式で表される。

$$L=\sum_{d}\sum_{w}n(d,w)\log\sum_{z}P(z)P(w|z)P(d|z)$$

3. 学習後のパラメータから文書をクラスタに分割する

2.で学習したパラメータを用いて、文書dの属するクラスタ$\hat{z}$を以下の式で求める。

$$\hat{z}=\arg\max_{z}P(d|z)$$

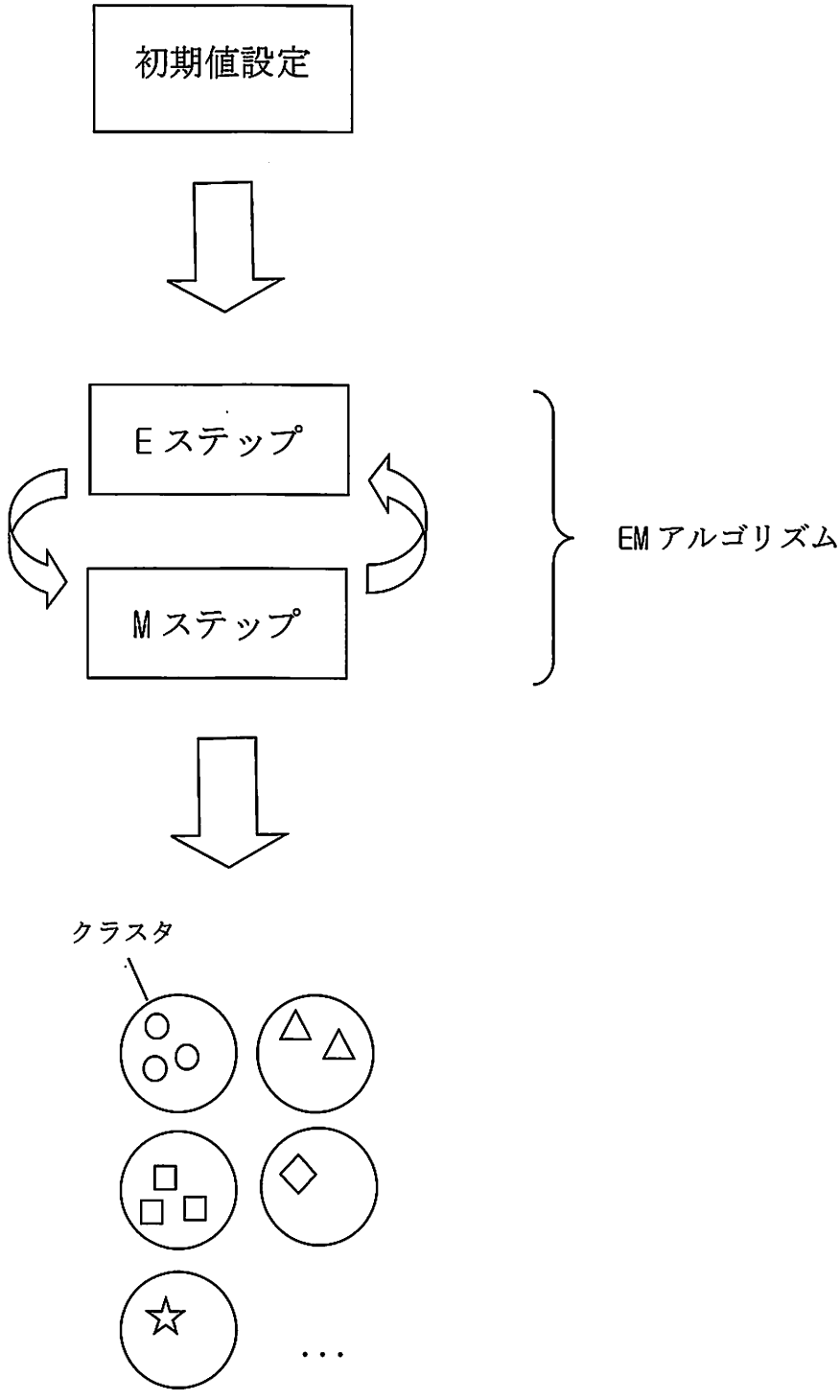

図.2　pLSIによる文書クラスタリング手順

3.2 EMアルゴリズム

EMアルゴリズムは隠れ変数が存在し、その隠れ変数の値を得ることができれば、その他のパラメータが推定でき、しかもその他のパラメータが推定できれば、隠れ変数の値も推定できるような場合に用いる。EMアルゴリズムはEステップとMステップの2つのステップからなり、この2つのステップを収束するまで交互に繰り返す。pLSIでは隠れ変数はzに当たる。

Eステップ

z以外のパラメータを固定したときのzの分布を求める。これは$P(z|d,w)$を表す。Aspectモデルでは、

$$P(d,w,z)=P(z)P(w|z)P(d|z)$$

が仮定されているので、結局$P(z_K|d,w)$は以下のようになる。

$$P(z_k|d,w)=\frac{P(d,w,z_k)}{P(d,w)}=\frac{P(z_k)P(w|z_k)P(d|z_k)}{\sum_{k=1}^{K}P(z_k)P(w|z_k)P(d|z_k)}$$

簡単のために$P(z_K|d_i,w_j)=Q_{ijk}$とおいておく。EステップではこのQ_{ijk}を求めることになる。更新式の形で書くと以下のとおり。

$$Q_{ijk}^{(t+1)}=\frac{P(z_k)^{(t)}P(w_j|z_k)^{(t)}P(d_i|z_k)^{(t)}}{\sum_{k=1}^{K}P(z_k)^{(t)}P(w_j|z_k)^{(t)}P(d_i|z_k)^{(t)}}$$

Mステップ

zを固定した場合、つまりQ_{ijk}を固定した場合のその他のパラメータを求める。

$$
\begin{aligned}
L &= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \log\left(\sum_{k=1}^{K} P(z_k)P(w_j \mid z_k)P(d_i \mid z_k)\right) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \log\left(\sum_{k=1}^{K} Q_{ijk} \frac{P(z_k)P(w_j \mid z_k)P(d_i \mid z_k)}{Q_{ijk}}\right) \\
&\leq \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} Q_{ijk} \log \frac{P(z_k)P(w_j \mid z_k)P(d_i \mid z_k)}{Q_{ijk}}
\end{aligned}
$$

最後の式の変形はJensenの不等式から得られる。そして最後の不等式では、実は、等号が成立していることが簡単に確認できる。

そのため、さらに変形して以下が成立する。

$$
\begin{aligned}
L = &\sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} Q_{ijk} \log(P(z_k)P(w_j \mid z_k)P(d_i \mid z_k)) \\
&- \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} Q_{ijk} \log Q_{ijk}
\end{aligned}
$$

上式の第2項はMステップでは定数なので、Lの最大化には関係ない。第2項を省いたものを再びLとおく。

$$
L = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} Q_{ijk} \log(P(z_k)P(w_j \mid z_k)P(d_i \mid z_k))
$$

ここから、Lの最大化は$\sum_{i=1}^{N} P(d_i \mid z_k) = 1$、$\sum_{j=1}^{M} P(w_j \mid z_k) = 1$、$\sum_{k=1}^{K} P(z_k) = 1$の関係があるので、ラグランジュの未定乗数法を利用して解ける。

$$
L + \sum_{k=1}^{K} \alpha_k \left(1 - \sum_{i=1}^{N} P(d_i \mid z_k)\right) + \sum_{k=1}^{K} \beta_k \left(1 - \sum_{j=1}^{M} P(w_j \mid z_k)\right) + \gamma\left(1 - \sum_{k=1}^{K} P(z_k)\right)
$$

$P(d_i \mid z_k) = u_{ik}$ 、$P(w_j \mid z_k) = v_{jk}$ 、$P(z_k) = w_k$ とおいて、上記の式をu_{ik} 、v_{jk} 、w_k でそれぞれ偏微分して極値問題を解く。ここでQ_{ijk} は正確に書くと$Q_{ijk}^{(t)}$ であり、Mステップの時点で定数になっている。

まず、u_{ik} の極値問題を解く。

$$\frac{\sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}}{u_{ik}} - \alpha_k = 0$$

よって、

$$u_{ik} = \frac{\sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}}{\alpha_k}$$

両辺を i に関して和をとると、

$$\sum_{i=1}^{N} u_{ik} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}}{\alpha_k}$$

以上より、

$$u_{ik} = P(d_i \mid z_k) = \frac{\sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}}{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}}$$

更新式の形で書くと以下となる。

$$P(d_i \mid z_k)^{(t)} = \frac{\sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}$$

左辺の繰り返し回数がt となっているが、これは右辺の$Q_{ijk}^{(t)}$ がすでに更新されている形なので、間違いではない。

同様に、v_{jk} とw_k でそれぞれ偏微分して極値問題を解くことで、以下が得られる。

$$P(w_j \mid z_k)^{(t)} = \frac{\sum_{i=1}^{N} n(d_i, w_j) Q_{ijk}^{(t)}}{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}$$

$$P(z_k)^{(t)} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}{\sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) Q_{ijk}^{(t)}}$$

実際にEMアルゴリズムでパラメータを求める際には、初期値が必要である。つまり、$P(d_i \mid z_k)^{(0)}$ 、$P(w_j \mid z_k)^{(0)}$ 、$P(z_k)^{(0)}$ を設定しておかなければならない。その設定方法については次章で述べる。

第4章　初期値設定手法

4.1 従来の手法

従来の初期値の設定方法としては、乱数を用いていた。実際には初期値を変化させ、各パラメータを求める実験を複数回繰り返し、それらの結果の中から尤度が最大のものを選択する。もちろんこの方法では、毎回安定した解は得られない。また、何度も実験を繰り返すので、計算時間がかかるという問題点も挙げられる。

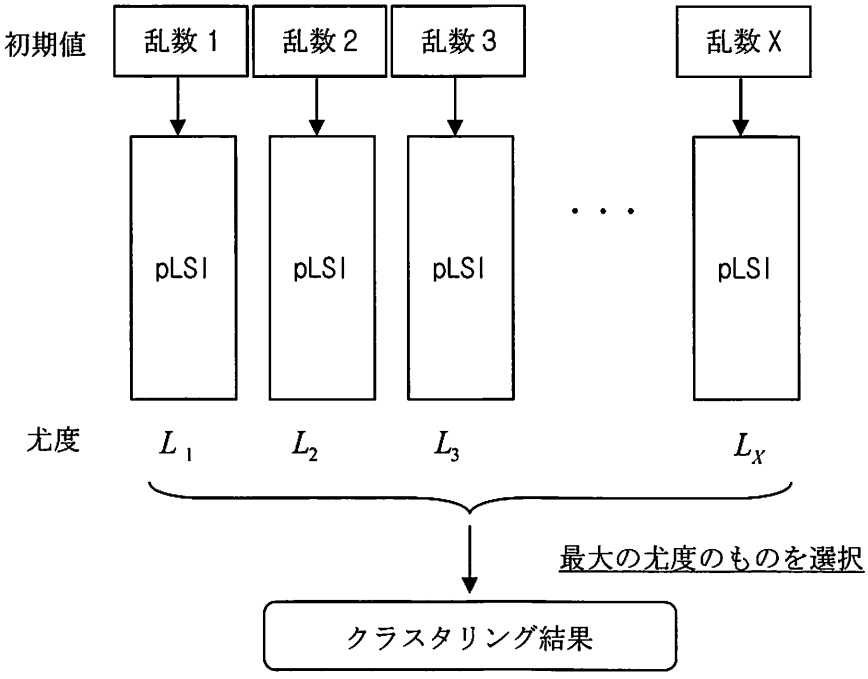

図.3　従来の初期値設定手法

4.2 KKZを利用した手法

本研究で提案する手法が、KKZを利用した設定方法である。KKZはk-means法で用いられる初期値設定手法であるが、そのKKZをpLSIに適用し、初期値設定を行うのが本手法である。ここでは、KKZについての解説とpLSIへの適用方法を述べる。

KKZ

KKZは、初期値となる代表点を選択する手法である。クラスタの代表となる点、ここでは文書を選択する。それぞれの代表点はお互いから遠いもの、似ていないものが選択されるという特徴がある。具体的なKKZによる初期値計算法を以下に述べる。

1. 文書集合の i 行の要素の総和を計算し、最大のものを選び、代表点 $\mathbf{c}_0$ に行ベクトル $\mathbf{x}_i$ を代入
2. $n=1$
3. 各行ベクトルごとに既にある代表点 $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_K$ との距離 g_{ij} を計算する

$$g_{ij} = \left\| \mathbf{x}_i - \mathbf{c}_j \right\|$$

4. 各行毎に g_{ij} の中で最小のもの gm_{ij} を選ぶ

$$gm_{ij} = \min_j \{ g_{ij} \}$$

5. gm_{ij} の中で最も値の大きいものを選ぶ
6. 選んだ gm_{ij} の i 番目の行ベクトルを $\mathbf{c}_n$ に代入する
7. $n+=1$
8. 設定したクラスタの数だけ代表点が作成されたなら終了
9. 3. に戻る

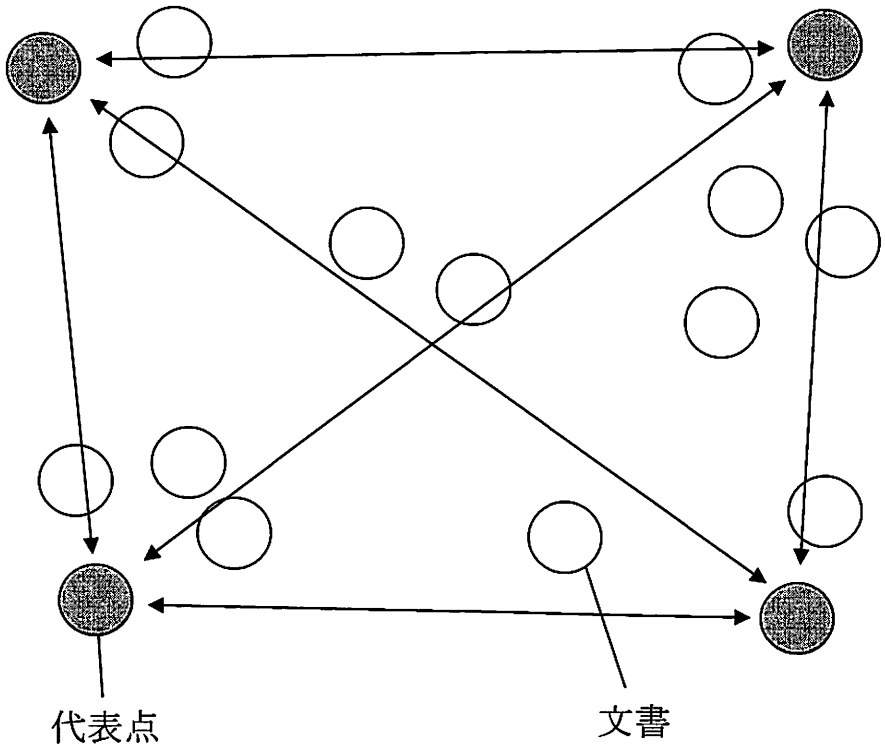

図.4　KKZ による代表点選択

pLSIへの適用方法

KKZにより代表点となる文書を選択したら、pLSIの初期値への変換を行う。計算式は以下のとおり。

$$P(w_j \mid z) = \frac{f_j + \alpha}{\sum_{j'}(f_{j'} + \alpha)} \qquad (1)$$

$$P(d \mid z) = 1/N$$

$$P(z) = 1/K$$

f_j はクラスタ j の代表点で、$P(w_j \mid z)$ の値が0にならないように補正パラメータ α を用いている。N は文書数、K はクラスタ数である。

KKZを利用した手法では、初期値を設定してしまえば、あとはクラスタリングを行うだけである。何度も実験を繰り返さず、pLSIによる処理は一度で済む。

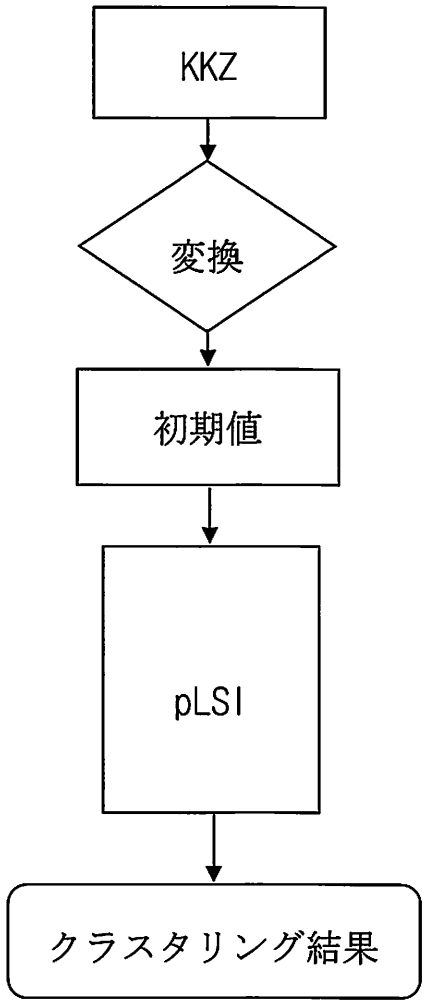

図.5　KKZ を利用した初期値設定手法

第5章　実験

5.1 従来の手法

5.1.1　実験方法

まず、従来の手法を用いて実際に文書クラスタリングを行う。

クラスタリングを行う対象は、データセットtr11（文書数414、単語数6429、クラスタ数9)、tr12（文書数313、単語数5804、クラスタ数8)、tr23（文書数204、単語数5832、クラスタ数6)、tr41（文書数878、単語数7454、クラスタ数10)、tr45（文書数690、単語数8261、クラスタ数10）の5つを使用する。

pLSIについては、下記アドレスからダウンロードできるpLSIのツールを用いる。

http://chasen.org/~taku/software/plsi/　ファイル名 ： plsi-0.03.tar.gz

このファイルを展開後、cygwinに実装し、使用する。cygwinについては下記アドレスを参照のこと。

http://cygwin.com/

また、ツールを正常に動かすためにgetopt.hの106行目を

```
extern int getopt (int argc, char *const *argv, const char *shortopts);
```

と書き換えて使用した。

ツールについて簡単に解説をしておくと、外部ファイル *.pz , *.pwz , *.pdz （それぞれが$P(z)$ 、$P(w|z)$ 、$P(d|z)$ に当たる）という拡張子のファイル（*部の名前は統一しておく必要がある）を用意するとそれらを初期値として読み込み、pLSIを実行してくれるというものである。結果は各パラメータ毎にファイルが出力される。

評価方法

クラスタリング結果の評価として、エントロピーと純度を用いる。

エントロピーはクラスタリング結果と正解データとの違いのバラツキの度合いを表していて、値が小さいほどよいクラスタリング結果と言える。

エントロピーは以下の式で定義される。

$$\sum_{i=1}^{K}\frac{|C_i|}{N}E_i=\sum_{i=1}^{K}\frac{\sum_{j=1}^{K}x_{ij}}{N}E_i$$

上記はクラスタリング結果のエントロピーを表している。以下は各クラスタに対するエントロピーの定義である。

$$E_i=-\sum_{h=1}^{K}P(A_h\mid C_i)\log P(A_h\mid C_i)$$

ここで確率$P(A_h\mid C_i)$が出ているが、これは、

$$\frac{|A_h\cap C_i|}{|C_i|}=\frac{x_{ih}}{\sum_{j=1}^{K}x_{ij}}$$

によって推定する。

ここで、Cはクラスタリングの結果、Aは正解となるデータを表している。

また、$x_{ij}=|C_i\cap A_j|$　つまりクラスタリングの結果と正解とで共通に属するデータの個数を表す。

純度はクラスタリング結果が正解データをどの程度含むかという度合いを表していて、値が大きいほどよいクラスタリング結果であると言える。

純度は以下の式で定義される。

$$P_i = \frac{1}{|C_i|} \max_h |C_i \cap A_h|$$

クラスタリング結果の純度は、各クラスタのデータ数による重み付き平均をとることで定義される。つまり、以下が定義である。

$$\sum_{i=1}^{K} \frac{|C_i|}{N} P_i = \frac{1}{N} \sum_{i=1}^{K} \max_h |C_i \cap A_h|$$

初期値の設定方法

初期値として、自作のプログラム（詳細は付録A）により、乱数で10個の初期値〈1〉～〈10〉を作成して用いる。

5.1.2 実験結果

従来の手法による初期値を用いたクラスタリング結果を以下に示す。

太字が尤度が最大のもの、下線はエントロピー、純度それぞれで最もよい値が出ていたものを表す。また、エントロピー、純度は小数点第4位で四捨五入した値で、尤度は小数点第1位で四捨五入した値である。

表.1 tr11 従来の手法 実験結果

	尤度	エントロピー	純度
〈1〉	-5727801	0.508	0.606
〈2〉	-5776082	<u>0.403</u>	<u>0.693</u>
〈3〉	-5685444	0.418	0.606
〈4〉	**-5414421**	**0.526**	**0.565**
〈5〉	-5524797	0.437	0.638
〈6〉	-5720161	0.461	0.601
〈7〉	-5484517	0.473	0.609
〈8〉	-5815605	0.431	0.618
〈9〉	-5759625	0.405	0.676
〈10〉	-5529573	0.459	0.635

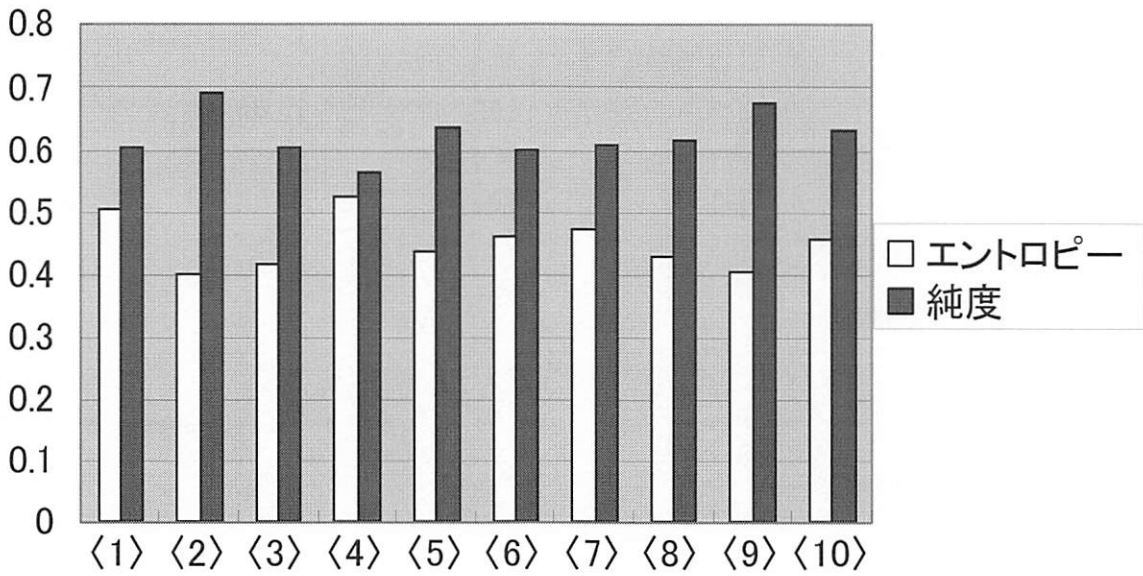

図.6 tr11 従来の手法 実験結果

表.2 tr12 従来の手法 実験結果

	尤度	エントロピー	純度
〈1〉	-5426729	0.702	0.415
〈2〉	-5395411	0.718	0.390
〈3〉	-5426473	0.525	0.559
〈4〉	**-5368323**	**0.538**	**0.521**
〈5〉	-5416084	0.829	0.364
〈6〉	-5376613	0.546	0.578
〈7〉	-5505197	0.638	0.450
〈8〉	-5455271	0.723	0.399
〈9〉	-5384895	0.656	0.393
〈10〉	-5380689	0.788	0.339

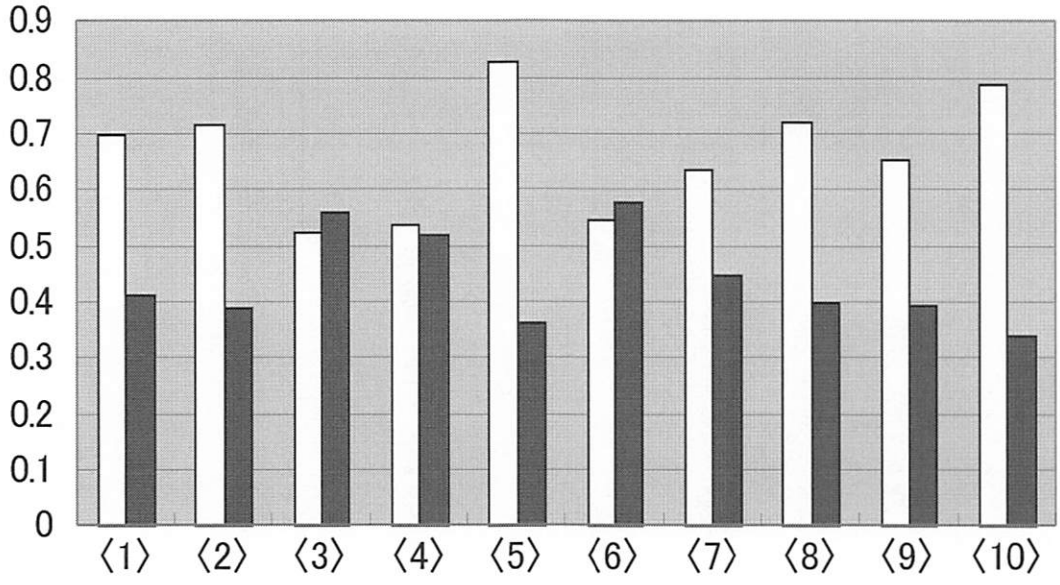

図.7 tr12 従来の手法 実験結果

表.3 tr23 従来の手法 実験結果

	尤度	エントロピー	純度
〈1〉	-9081493	0.732	0.588
〈2〉	-9296144	0.795	0.446
〈3〉	-9087684	0.617	0.603
〈4〉	-9332048	0.716	0.446
〈5〉	-9221804	0.750	0.471
〈6〉	-9363485	0.714	0.500
〈7〉	-9357264	0.806	0.451
〈8〉	-9232258	0.594	0.539
〈9〉	-9210324	0.726	0.578
〈10〉	-9352852	0.781	0.461

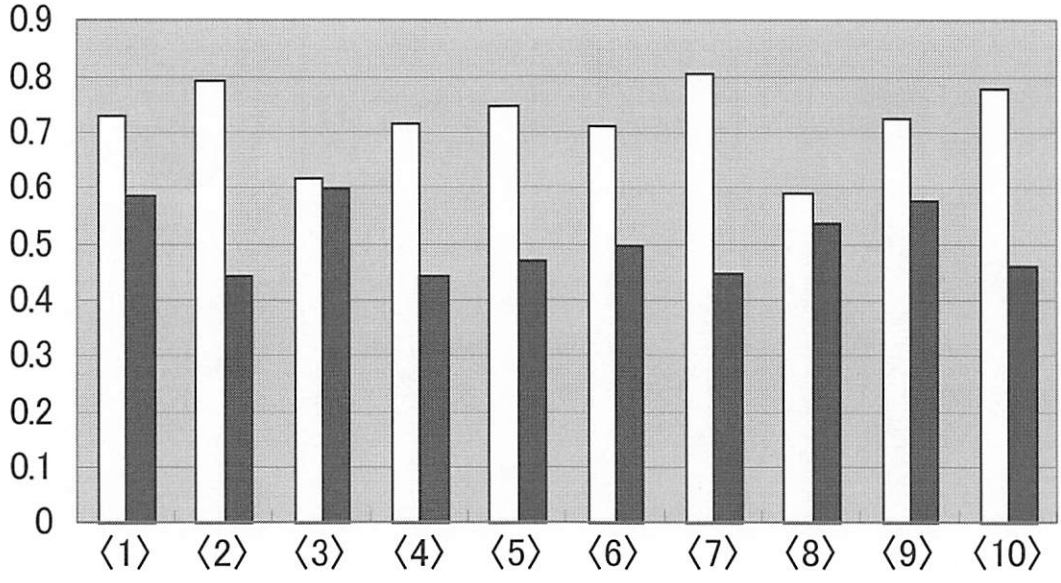

図.8 tr23 従来の手法 実験結果

表.4 tr41 従来の手法 実験結果

	尤度	エントロピー	純度
〈1〉	-5615190	0.366	0.641
〈2〉	-5647109	0.330	0.726
〈3〉	-5612051	0.325	0.747
〈4〉	-5643570	0.360	0.685
〈5〉	-5685484	0.351	0.682
〈6〉	-5650109	0.268	0.774
〈7〉	-5631578	0.353	0.679
〈8〉	-5629808	0.336	0.686
〈9〉	-5651333	0.280	0.741
〈10〉	-5084072	0.757	0.342

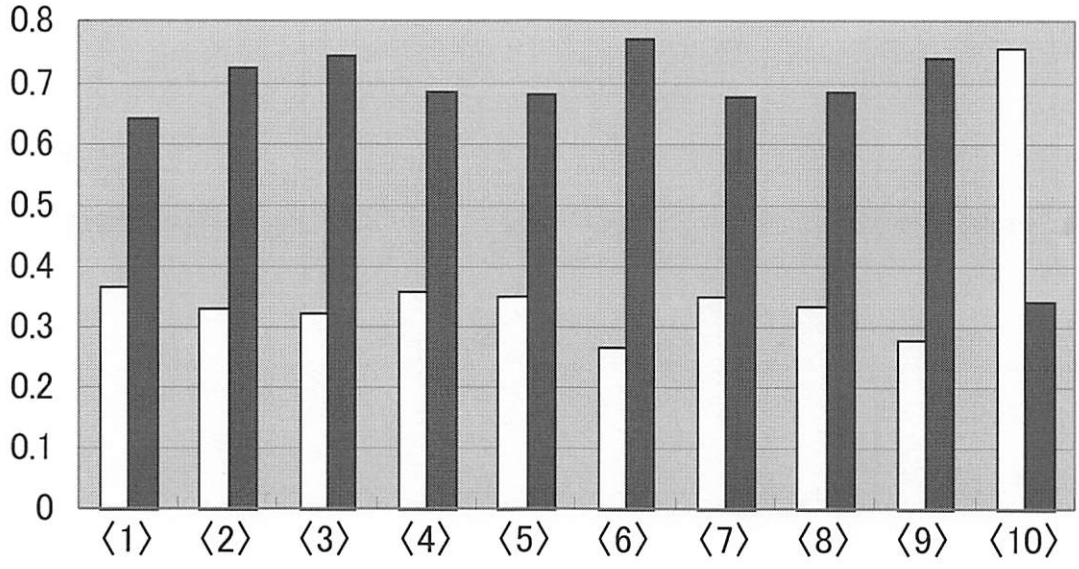

図.9 tr41 従来の手法 実験結果

表.5 tr45 従来の手法 実験結果

	尤度	エントロピー	純度
〈1〉	-10379292	0.633	0.461
〈2〉	-10326093	0.564	0.526
〈3〉	-10272983	0.606	0.461
〈4〉	-10456813	0.570	0.483
〈5〉	-10293308	0.624	0.459
〈6〉	-10256456	0.638	0.459
〈7〉	-10306061	0.564	0.533
〈8〉	-10395481	0.577	0.504
〈9〉	-10191850	0.589	0.483
〈10〉	-10255547	0.641	0.448

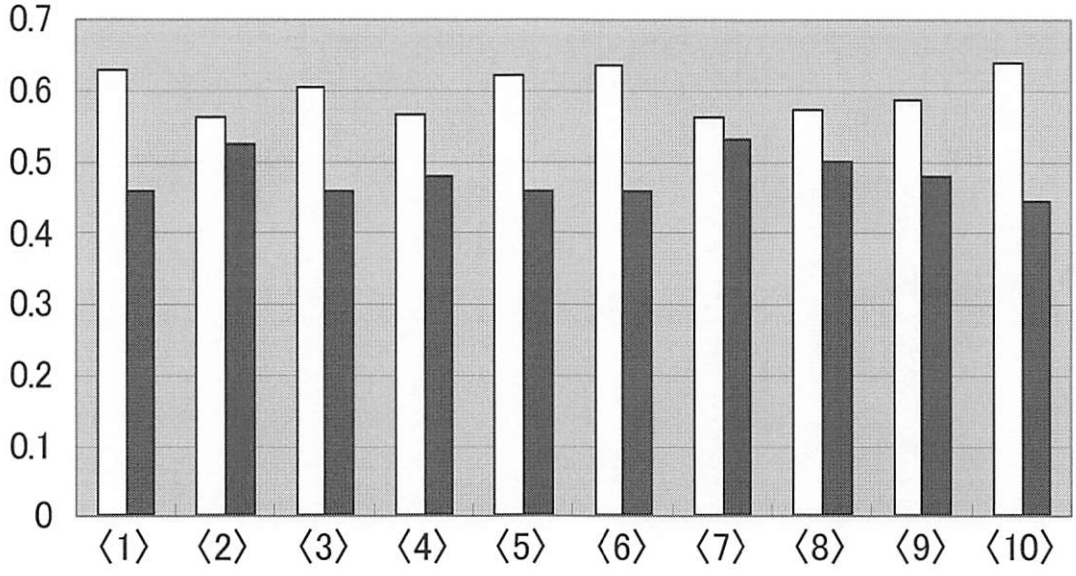

図.10 tr45 従来の手法 実験結果

tr11のエントロピーは最大値と最小値の差が0.123、純度では0.128となっている。同様にtr12のエントロピーは0.304、純度は0.239、tr23のエントロピーは0.212、純度は0.157、tr41のエントロピーは0.489、純度は0.432、tr45のエントロピーは0.077、純度は0.085となっている。

従来手法の中では、tr41の尤度が最大のものはかなり悪い結果になっている。逆にtr12の尤度が最大のものはそこそこの結果になっていることも分かる。

データセットによるが、初期値によって結果がばらついているのが分かる。

5.2 KKZを利用した手法

5.2.1 実験方法

次に、KKZを利用した手法で同様の実験を行う。

初期値を除き、他の条件は5.1.1で述べたものと同様である。

初期値として、KKZを利用した手法により（1）式の補正パラメータαの値を1.0にしたものと0.5にしたものの2つ〈K1.0〉と〈K0.5〉を用意する。

5.2.2 実験結果

KKZを利用した手法による初期値を用いたクラスタリング結果を以下に示す。

aveは従来の手法で設定した10個の結果の平均値で、太字は従来の手法で尤度が最大だったものである。下線は4つの中で最もよい値が出ていたものを表す。

表.6 tr11実験結果

	尤度	エントロピー	純度
〈K1.0〉	-5829758	0.345	0.732
〈K0.5〉	-5866849	0.352	0.729
〈4〉	-5414421	0.526	0.565
ave	-5643803	0.452	0.625

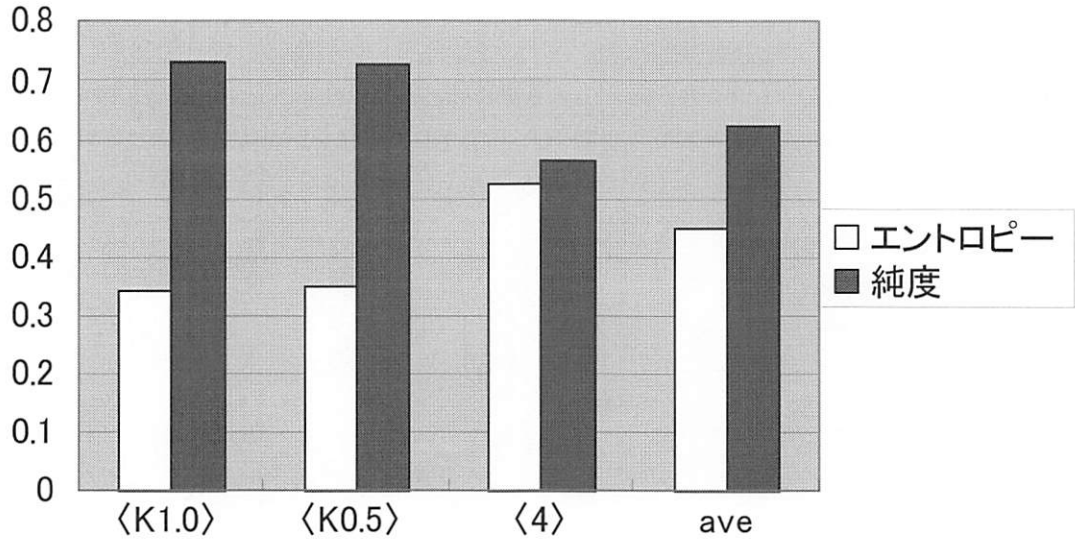

図.11 tr11実験結果

表.7 tr12実験結果

	尤度	エントロピー	純度
〈K1.0〉	-5493318	0.551	0.569
〈K0.5〉	-5399720	0.512	0.543
〈4〉	-5368323	0.538	0.521
ave	-5413569	0.666	0.441

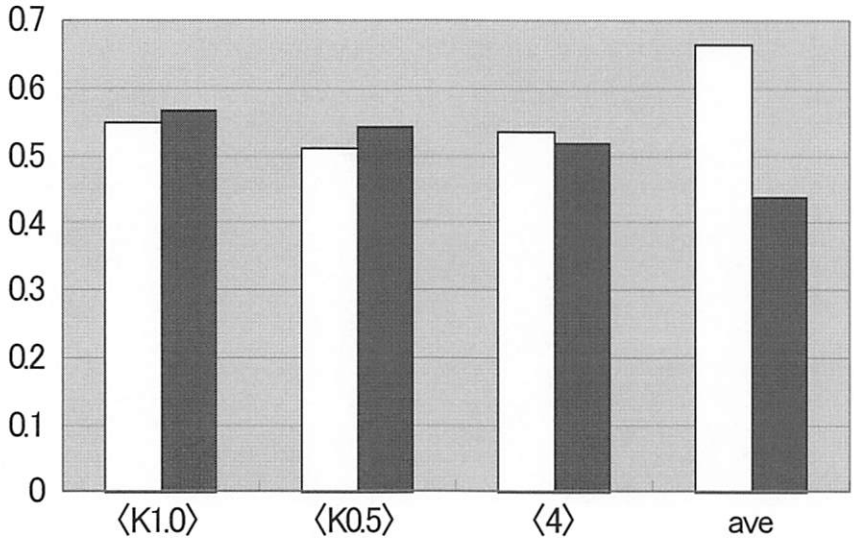

図.12 tr12実験結果

表.8 tr23実験結果

	尤度	エントロピー	純度
〈K1.0〉	-9250381	0.629	0.603
〈K0.5〉	-9279940	0.670	0.564
〈1〉	-9081493	0.732	0.588
ave	-9253536	0.723	0.508

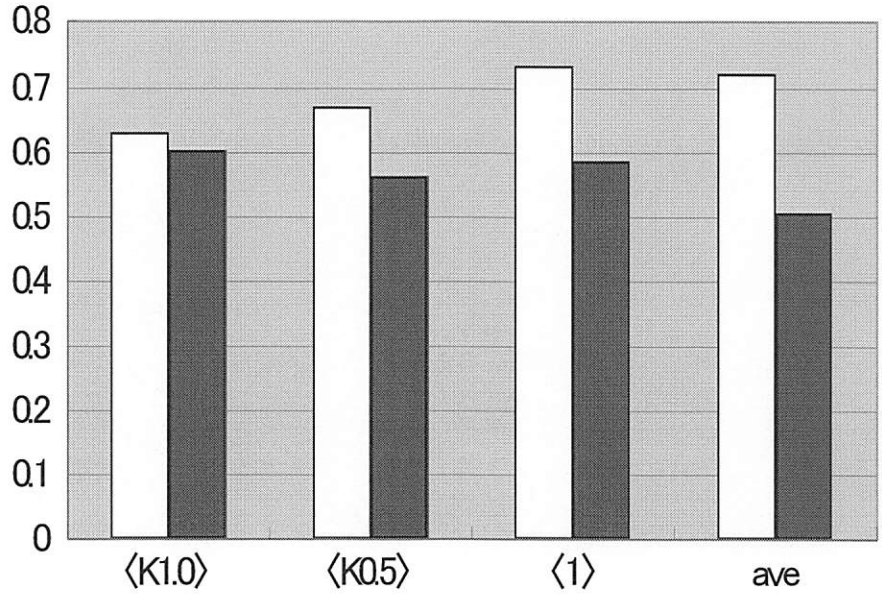

図.13 tr23実験結果

表.9 tr41実験結果

	尤度	エントロピー	純度
〈K1.0〉	-5645490	0.257	0.770
〈K0.5〉	-5646219	0.256	0.771
〈10〉	-5084072	0.757	0.342
ave	-5585030	0.373	0.670

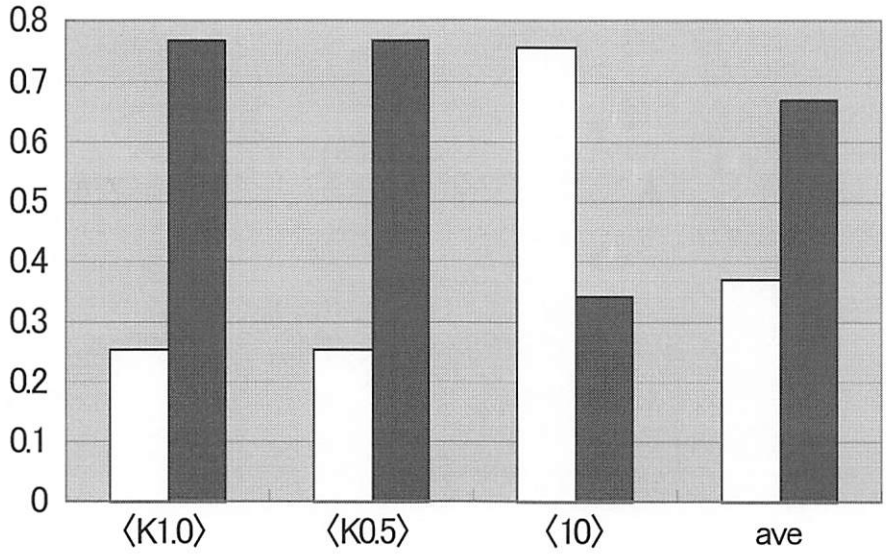

図.14 tr41実験結果

表.10 tr45実験結果

	尤度	エントロピー	純度
〈K1.0〉	-10247828	0.592	0.468
〈K0.5〉	-10255518	0.556	0.546
〈9〉	-10191850	0.589	0.483
ave	-10313388	0.601	0.482

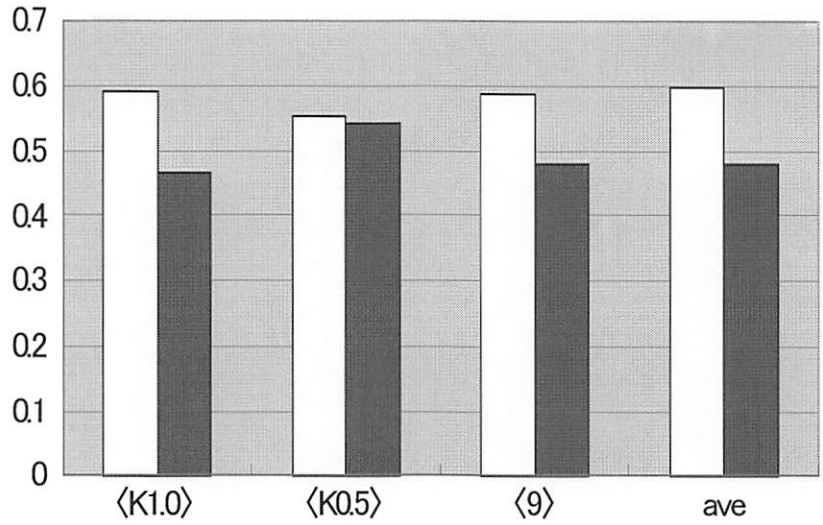

図.15 tr45実験結果

KKZを利用した手法の結果は概ね良好であると言えるが、tr23の〈K0.5〉の純度や、tr45の〈K1.0〉の純度などは、従来の手法よりも悪い結果が出ている。

〈K1.0〉と〈K0.5〉とではデータセットによってよい結果になるものが入れ替わるが、それほど大きな差は開いていない。

尤度に関しては、全てのデータセットで従来手法の尤度最大だったものが大きな値を持っている。

第6章　考察

まず、従来の手法について考える。実験結果から尤度が最大のものを選んだ場合、必ずしも良い結果が出るわけではないということが分かった。また、かなり飛びぬけたクラスタリング結果を示すものも確認でき、乱数による初期値設定の場合、結果のバラツキに対して十分考慮する必要があると考える。

次に、KKZを利用した手法について考える。

まず、〈K1.0〉と〈K0.5〉の2つの初期値を比較する。データセット毎に見ていくと、どちらか片方が常によい結果であるわけでもなく、また、差は大きく開かないように見受けられる。よって、代表点から初期値への変換時の補正パラメータは結果にそれほど大きく影響しないと考えられる。

従来の手法（尤度が最大のもの）と比較すると、KKZを利用した手法の方がよい結果を得られていると言える。しかし、従来の手法で尤度が小さかったために選ばれなかったものそれぞれと比較してみると、KKZを利用した手法よりもよい結果が得られているものもあった。これは、従来の手法が乱数を用いているので当然のことではある。そこで、従来の手法で設定した全ての初期値の結果の平均をとって比べてみる。

KKZを用いた手法と従来の手法の平均とで比べると、それでもなおKKZを用いた手法の方がよいクラスタリング結果を得られている。しかし、一部ではKKZを用いた手法よりもよい結果を得ている部分もある。それに加え、両者はエントロピーに関しても純度に関しても、それほど大きく値を離しているわけではない。よって、KKZを用いた手法は、従来の手法と同程度か、それ以上の精度のクラスタリング結果が望める。

また、KKZを用いた手法は、従来の手法と異なり、何度も実験を行う必要が無く、計算時間において優れていることが分かっている。これらのことから、KKZを利用した手法は、従来の手法よりも優秀な手法であると言える。

第7章　おわりに

本研究ではpLSIによる文書クラスタリングにおいて、より良い結果を得るために、初期値設定の手法について研究を行った。

そこで、クラスタリング技法のk-means法で用いられる初期値設定手法KKZを利用することを考えた。

従来の手法と、提案するKKZを利用した初期値設定手法の比較のために実際に文書クラスタリングを行う実験をした。

KKZを利用した手法のクラスタリング結果は良好であったが、従来手法と比べて同程度か少し良い程度の結果であった。しかし、従来手法と比べて提案する手法は計算時間が短くて済むという利点もあった。

従来手法と同程度の結果が、より短い時間で得られるという点で、本研究で提案したKKZを利用した手法は優れているということが言えた。

参考文献

[1]Thomas Hofmann: "Probabilistic Latent Semantic Indexing", Proc. SIGIR-99, pp. 35-44(1999).

[2]新納浩幸: "R で学ぶクラスタ解析", オーム社,(2007).

[3]岸田和明: "文書クラスタリングの技法: 文献レビュー", Proc. Library and Information Science No. 49, pp. 33-75(2003).

[4]J. He, M. Lan, C-L. Tan, S-Y. Sung and H-B. Low:"Initialization of Cluster Refinement Algorithms: A Review and Comparative Study", Proc. IEEE Int. Joint Conf. Neural Networks, pp. 297-302(2004).

[5]伊藤潤, 石田崇, 後藤正幸, 平澤茂一: "PLS I を利用した文書からの知識発見", FIT 論文集vol. 2, pp. 83-84(2003).

付録A

プログラムソースリスト

※全てJavaのプログラムソース

```
//乱数による初期値ファイルを出力するプログラム
import java.io.*;

class rndj {
        public static void main( String args[] ) {
                try {

                        int i = 0; int j = 0;
                        float x = 0.0f;
                        int d = Integer.valueOf( args[3] ).intValue();
                        int w = Integer.valueOf( args[4] ).intValue();
                        int z = Integer.valueOf( args[5] ).intValue();

                        FileWriter fw = new FileWriter( args[0] );  //出力ファイ
ル名 .pdz
                        BufferedWriter bw = new BufferedWriter( fw );
                        for( i = 0; i < z; i++ ) {
                                for( j = 0; j < d; j++ ) {
                                        x = (float)Math.random();
                                        bw.write( x + " " );
                                }
                                bw.write("¥n");
                        }
                        bw.close();

                        FileWriter fw2 = new FileWriter( args[1] );  //出力ファ
イル名 .pwz
                        BufferedWriter bw2 = new BufferedWriter( fw2 );
```

```
                for( i = 0; i < z; i++ ) {
                        for( j = 0; j < w; j++ ) {
                                x = (float)Math.random();
                                bw2.write( x + " " );
                        }
                        bw2.write("¥n");
                }
                bw2.close();

                FileWriter fw3 = new FileWriter( args[1] );  //出力ファ
イル名 .pz
                BufferedWriter bw3 = new BufferedWriter( fw3 );
                for( i = 0; i < z; i++ ) {
                        x = (float)Math.random();
                        bw3.write( x + " " );
                }
                bw3.close();

        }
        catch( Exception e ) {
                System.out.println("Exception: " + e );
        }
    }
}
```

```
//データセットを読み取りKKZにより代表点を表示するプログラム
import java.io.*;

class cx {
	public static void main( String args[] ) {  //引数  文書数d 単語数w クラスタ数z データセット名
		try {
			int i = 0; int j = 0; int k = 0; int l = 0; int m = 0; int n = 1; int p = 0;
			int gh = 1; int mh = 1;
			int d = Integer.valueOf( args[0] ).intValue();  //文書数
			int w = Integer.valueOf( args[1] ).intValue();  //単語数
			int z = Integer.valueOf( args[2] ).intValue();  //クラスタ数
			int c[] = new int [z];
			double dc[][] = new double [z][d];      //各クラスタとの距離
			double dk[] = new double [d];
	//最小値一時代入用
			double tr[][] = new double [d][w];      //データセットの要素
			double dw = 0.0;                                    //各行の和
			double mdw = 0.0;                                   //各行の和（各クラスタとの距離）の最大

			for( i = 0; i < d; i++ ) {
				for( j = 0; j < w; j++ ) {
					tr[i][j] = 0.0;
				}
				for( l = 0; l < z; l++ ) {
					dc[l][i] = 0.0;
				}
			}

			FileReader fr = new FileReader( args[3] );  //args[3] =
```

データセット名

```
BufferedReader br = new BufferedReader( fr );
StreamTokenizer st = new StreamTokenizer( br );
st.eolIsSignificant(true);

while( st.nextToken() != StreamTokenizer.TT_EOF ) {
        switch( st.ttype ) {
                case StreamTokenizer.TT_NUMBER :
                        switch( k ) {
                                case 0 :
                                        m = (int)st.nval
- 1;
                                        k += 1;
                                        break;
                                case 1 :
                                        p = (int)st.nval
- 1;
                                        k += 1;
                                case 2 :
                                        tr[m][p]        =
st.nval;
                        }
                        break;
                case StreamTokenizer.TT_EOL :
                        k = 0;
                        break;
                default :
                        break;
        }
}
fr.close();
k = 0;

//c[0]
for( i = 0; i < d; i++ ) {
        for( j = 0; j < w; j++ ) {
```

```
                dw += tr[i][j];
            }
            if( dw > mdw ) {
                mdw = dw;
                mh = i;
            }
            dw = 0.0;
        }

        c[0] = mh;

//c[n]

        for( n = 1; n < z; n++ ) {

            for( i = 0; i < d; i++ ) {
                for( j = 0; j < w; j++ ) {
                    dc[ n - 1 ][i] += Math.pow( tr[i][j] - tr[ c[ n - 1 ] ][j], 2.0 );
                }
            }

            for( i = 0; i < d; i++ ) {
                dk[i] = dc[0][i];
                for( l = 0; l < n; l++ ) {
                    if( dk[i] > dc[l][i] ) dk[i] = dc[l][i];
                }
            }

            mh = 0;
            mdw = 0.0;
            for( i = 0; i < d; i++ ) {
                if( mdw < dk[i] ) {
```

```
                                        mdw = dk[i];
                                        mh = i;
                                }
                        }

                        c[n] = mh;

                }

                FileWriter fw = new FileWriter( "kkz.txt" );
                BufferedWriter bw = new BufferedWriter( fw );
                for( i = 0; i < z; i++ ) {
                        x = (float)Math.random();
                        bw.write( x + " " );
                }
                bw.close();

        }
        catch( Exception e ) {
                System.out.println("Exception: " + e );
        }
    }
}
```

```
//代表点の行番号ファイルとデータセットを読み取りpLSIの初期値に変換し、test.pz、
test.pdz、test.pwzファイルを出力するプログラム
import java.io.*;

class kp {
        public static void main( String args[] ) {  //引数  文書数d 単語数w クラ
スタ数z kkz.txt データセット名
                try {
                        int i = 0; int j = 0; int k = 0; int l = 0; int m = 1;
                        double alpha = 1.0; //補正パラメータ  1.0 or 0.5
                        int d = Integer.valueOf( args[0] ).intValue();  //文書数
                        int w = Integer.valueOf( args[1] ).intValue();  //単語数
                        int z = Integer.valueOf( args[2] ).intValue();  //クラス
タ数
                        double pz[] = new double [z];
                        double pdz[][] = new double [z][d];
                        double pwz[][] = new double [z][w];
                        double iwz[][] = new double [z][w];
                        int c[] = new int [z];
                        double beta[] = new double [z];

                        for( i = 0; i < z; i++ ) {
                                pz[i] = 0;
                                beta[i] = 0;
                                for( j = 0; j < w; j++ ) {
                                        pwz[i][j] = 0;
                                        iwz[i][j] = 0;
                                }
                                for( l = 0; l < d; l++ ) {
                                        pdz[i][l] = 0;
                                }
                        }

                        i = 0;

                        FileReader fr = new FileReader( args[3] );  //args[3] =
```

kkz.txt

```
BufferedReader br = new BufferedReader( fr );
StreamTokenizer st = new StreamTokenizer( br );
st.eolIsSignificant(true);

while( st.nextToken() != StreamTokenizer.TT_EOF ) {
        switch( st.ttype ) {
                case StreamTokenizer.TT_NUMBER :
                        c[i] = (int)st.nval;
                        i++;
                        break;
                default :
                        break;
        }
}
fr.close();
```

//pz

```
for( i = 0; i < z; i++ ) pz[i] = 1.0 / z;
```

//pdz

```
for( i = 0; i < z; i++ ) {
        for( j = 0; j < d; j++ ) {
                        pdz[i][j] = 1.0 / d;
        }
}
```

//pwz

```
i = 1; j = 0;

FileReader fr2 = new FileReader( args[4] );  //args[4] =
```

データセット名

```
BufferedReader br2 = new BufferedReader( fr2 );
StreamTokenizer st2 = new StreamTokenizer( br2 );
```

```
                    st2.eolIsSignificant(true);
                    st2.whitespaceChars( ':' , ':' );

                    while( st2.nextToken() != StreamTokenizer.TT_EOF ) {
                            switch( st2.ttype ) {
                                    case StreamTokenizer.TT_NUMBER :
                                            if( m < 0 ) {
                                                    for( j = 0; j < z; j++ )
{
                                                            if( i == c[j] )
iwz[j][k] = st2.nval;
                                                    }
                                            } else {
                                                    k = (int)st2.nval;
                                            }
                                            m *= -1;
                                            break;
                                    case StreamTokenizer.TT_EOL :
                                            i++;
                                            break;
                                    default :
                                            break;
                            }
                    }
                    fr2.close();

                    for( i = 0; i < z; i++ ) {
                            for( j = 0; j < w; j++ ) {
                                    beta[i] += iwz[i][j] + alpha;
                            }
                    }
                    for( i = 0; i < z; i++ ) {
                            for( j = 0; j < w; j++ ) {
                                    pwz[i][j] = ( iwz[i][j] + alpha ) /
beta[i];
                            }
```

```
                }


                FileWriter fw = new FileWriter( "test.pz" );
                BufferedWriter bw = new BufferedWriter( fw );
                for( i = 0; i < z; i++ ) {
                        bw.write( pz[i] + " " );
                }
                bw.close();

                FileWriter fw2 = new FileWriter( "test.pdz" );
                BufferedWriter bw2 = new BufferedWriter( fw2 );
                for( i = 0; i < z; i++ ) {
                        for( j = 0; j < d; j++ ) {
                                bw2.write( pdz[i][j] + " " );
                        }
                        bw2.write( "¥n" );
                }
                bw2.close();

                FileWriter fw3 = new FileWriter( "test.pwz" );
                BufferedWriter bw3 = new BufferedWriter( fw3 );
                for( i = 0; i < z; i++ ) {
                        for( j = 0; j < w; j++ ) {
                                bw3.write( pwz[i][j] + " " );
                        }
                        bw3.write( "¥n" );
                }
                bw3.close();

        }
        catch( Exception e ) {
                System.out.println("Exception: " + e );
        }
    }
}
```

```
//データセットと学習後の.pz、.pwz、.pdzファイルを読み込み、尤度を計算して表示する
プログラム
import java.io.*;

class yuudo {
        public static void main( String args[] ) {  //引数  クラスタの数 データ
セット名 .pz .pwz .pdz
                try {
                        int i = 0; int j = 0;
                        int z = Integer.valueOf( args[0] ).intValue();  //クラス
タの数

                        FileReader frz = new FileReader( args[2] );  //args[2] =
 .pz
                        BufferedReader brz = new BufferedReader( frz );
                        StreamTokenizer stz = new StreamTokenizer( brz );
                        stz.eolIsSignificant(true);
                        double tz[] = new double [z];
                        while( stz.nextToken() != StreamTokenizer.TT_EOF ) {
                                switch( stz.ttype ) {
                                        case StreamTokenizer.TT_NUMBER:
                                                tz[i] = stz.nval;
                                                i++;
                                                break;
                                        default:
                                                break;
                                }
                        }
                        frz.close();
                        i = 0;

                        FileReader frwz = new FileReader( args[3] );  //args[3]
= .pwz
                        BufferedReader brwz = new BufferedReader( frwz );
                        StreamTokenizer stwz = new StreamTokenizer( brwz );
                        stwz.eolIsSignificant(true);
```

```
                    double twz[] = new double [z];

                    FileReader frdz = new FileReader( args[4] );  //args[4]
= .pdz
                    BufferedReader brdz = new BufferedReader( frdz );
                    StreamTokenizer stdz = new StreamTokenizer( brdz );
                    stdz.eolIsSignificant(true);
                    double tdz[] = new double [z];

                    double yuudo = 0.0;  double pdw = 0.0;
                    int k = 1;  int tng = 0;

                    FileReader fr = new FileReader( args[1] );  //args[1] =
tr
                    BufferedReader br = new BufferedReader( fr );
                    StreamTokenizer st = new StreamTokenizer( br );
                    st.whitespaceChars( ':' , ':' );

                    while( st.nextToken() != StreamTokenizer.TT_EOF ) {
                            switch( st.ttype ) {
                                    case StreamTokenizer.TT_NUMBER :
                                            if( k > 0 ) {  //単語No.取得
                                                    tng = (int)st.nval;
                                            } else {

                                                    while(   stwz.nextToken()
 != StreamTokenizer.TT_EOF ) {

        switch( stwz.ttype ) {

                                                                    case
StreamTokenizer.TT_NUMBER:

        if( j == ( tng - 1 ) ) twz[i] = stwz.nval;

        j++;
```

```
        break;
                                                        case
StreamTokenizer.TT_EOL:
        i++; j = 0;
        break;
                                                        default:
        break;
                                                    }
                                                }
                                                i = 0; j = 0;
                                                while(   stdz.nextToken()
 != StreamTokenizer.TT_EOF ) {
        switch( stdz.ttype ) {
                                                        case
StreamTokenizer.TT_NUMBER:
        if( j == ( st.lineno() - 1 ) ) tdz[i] = stdz.nval;
        j++;
        break;
                                                        case
StreamTokenizer.TT_EOL:
        i++; j = 0;
        break;
                                                        default:
        break;
                                                    }
                                                }
```

```
                                                    i = 0; j = 0;

                                                    for( i = 0; i < z; i++ )
{
                                                            pdw += tz[i] *
twz[i] * tdz[i];
                                                    }
                                                    yuudo += st.nval *
Math.log( pdw );
                                                    pdw = 0.0;
                                                    i = 0; j = 0;
                                            }
                                            k *= -1;
                                            break;
                                    default :
                                            break;
                                }
                        }
                        fr.close();
                        frwz.close();
                        frdz.close();

                        System.out.println( "尤度 : " + yuudo );

                }
                catch( Exception e ) {
                        System.out.println("Exception: " + e );
                }
        }
}
```