平成24年度茨城大学工学部情報工学科卒業研究論文

ディリクレ混合過程を利用した
単語用例の語義別クラスタリング

平成24年度2月8日

工学部情報工学科

執筆者：倉持辰洋(07T4024L)

指導教員：新納浩幸 准教授

平成25年度 茨城大学工学部情報工学科 卒業研究論文

ディリクレ混合過程を利用した
単語用例の語義別クラスタリング

著者：倉持辰洋 (07T4024L)
指導教員：新納浩幸　准教授

論文要旨

ある単語の用例集を対象に、その単語の語義に基いてクラスタリングをすることを語義推定 (Word Sense Induction, WSI) という。本論文ではディリクレ混合過程を利用して語義推定を行う。

自然言語処理の中心課題の１つとして、語義推定曖昧性解消 (Word Sense Disambiguation, WSD) がある。これは文中の多義語の語義を推定するタスクである。例えば「ボタン」には 1. 洋服のボタン、2. スイッチのボタン、3. 花のボタン、の少なくとも 3 つの語義がある。文中に単語「ボタン」が現れたときに 1、2、3 のどの語義かを推定する処理が WSD である。WSD の 1 つの特徴として、語義が予め与えられていることがある。しかし WSD が必要とされるタスクに応じて語義の粒度は異るし、時代の変化に伴って語義が変化したり、新しい語義が生れることもある。そのために WSD の処理以前に対象単語の語義セット自体を設定する必要がある。この処理が WSI であり、基本的に単語の語義別クラスタリングを行う。

WSI はクラスタリングのタスクであり、一般のクラスタリングアルゴリズムを利用して解決できる。ただし、一般にクラスタリングアルゴリズムは分割するクラスタ数を予め与えておく必要がある。しかし WSI のタスクの性格上、クラスタ数（語義数）を予め与えておくことはできない。

一方、近年、ベイズ理論を利用した文書の生成モデルとしてディリクレ混合過程が提案された。これは文書のトピックを潜在変数としているため、文書群をトピックに基いてクラスタリングする手法として捉えられる。ディリクレ混合過程の最大の特徴はトピック数自体も推定することである。つまりディリクレ混合過程を利用して、クラスタ数を指定することなしに文書群をトピックに基いてクラスタリングすることができる。そこで本論文では、文書を用例に置き換えてベクトルを作ることにより、ディリクレ混合過程を利用して WSI を試みる。ただしディリクレ混合過程は計算コストが高いので、ここでは最初にある程度の大きさのクラスタ数にクラスタリングを行っておき、それをディリクレ混合過程の初期値とすることで計算の負荷を減らすことを行う。

実験では単語「見る」を対象にコーパスから「見る」の用例 621 文を抽出した。辞書の定義では「見る」の定義は 6 つある。また上記 621 文の用例中の「見る」にはその用例内での「見る」の語義が付与されている。上記 621 文の用例に対してディリクレ混合過程を利用して語義別クラスタリングを行った結果、クラスタ数は 8 となった。ディリクレ混合過程の処理中にクラスタ数が 6 つなった際のクラスタリング結果からクラスタリングの評価を行った。クラスタ数を与える一般のクラスタリング手法 (Ward 法と k-means 法) と比較すると、クラスタリングの精度は若干悪かったが、初期値としてクラスタリング結果を使うことで計算時間を 「約 2 割」軽減できた。

目 次

表 目 次

図 目 次

第1章　はじめに

1.1　概要

ある単語の用例集を対象に、その単語の語義に基いてクラスタリングをすることを語義推定 (Word Sense Induction, WSI) という。本論文ではディリクレ混合過程を利用して語義推定を行う。

自然言語処理の中心課題の１つとして、語義推定曖昧性解消 (Word Sense Disambiguation, WSD) がある。これは文中の多義語の語義を推定するタスクである。例えば「ボタン」には 1. 洋服のボタン、2. スイッチのボタン、3. 花のボタン、の少なくとも 3 つの語義がある。文中に単語「ボタン」が現れたときに 1、2、3 のどの語義かを推定する処理が WSD である。WSD の 1 つの特徴として、語義が予め与えられていることがある。しかし WSD が必要とされるタスクに応じて語義の粒度は異るし、時代の変化に伴って語義が変化したり、新しい語義が生れることもある。そのために WSD の処理以前に対象単語の語義セット自体を設定する必要がある。この処理が WSI であり、基本的に単語の語義別クラスタリングを行う。

WSI はクラスタリングのタスクであり、一般のクラスタリングアルゴリズムを利用して解決できる。ただし、一般にクラスタリングアルゴリズムは分割するクラスタ数を予め与えておく必要がある。しかし WSI のタスクの性格上、クラスタ数（語義数）を予め与えておくことはできない。

一方、近年、ベイズ理論を利用した文書の生成モデルとしてディリクレ混合過程が提案された。これは文書のトピックを潜在変数としているため、文書群をトピックに基いてクラスタリングする手法として捉えられる。ディリクレ混合過程の最大の特徴はトピック数自体も推定することである。つまりディリクレ混合過程を利用して、クラスタ数を指定することなしに文書群をトピックに基いてクラスタリングすることができる。そこで本論文では、文書を用例に置き換えてベクトルを作ることにより、ディリクレ混合過程を利用して WSI を試みる。ただしディリクレ混合過程は計算コストが高いので、ここでは最初にある程度の大きさのクラスタ数にクラスタリングを行っておき、それをディリクレ混合過程の初期値とすることで計算の負荷を減らすことを行う。

1.2　論文の構成

初めに WSI を行う上で必要なクラスタリングについての説明。次に今回行うクラスタリング方法で利用するディリクレ混合過程、及びその前提知識となるベイズモデルやディリクレ分布を述べる。そしてディリクレ混合過程を推定するために用いる CRP(Chinese Restaurant Process) を解説し、実験方法へと持っていく。

第2章　クラスタリング

データの集まりをデータ間の類似度（あるいは非類似度）従って、いくつかのグループに分ることをクラスタリングという。この際データは n 次元ベクトルで表現されていることが前提である。また、ベクトルで表現されてることによりどの観点に着目したかも与えられてることになる。

クラスタリングの手法は通常 2 つの観点、「階層的手法」と「非階層的手法（分割最適化手法）」に分類される。

2.1　階層的クラスタリング

クラスタリング手法の中で、階層的手法は古典的であり、直感的にも自然な手法である。

与えられたデータセットの各データが 1 つのクラスタとなっている状態を初期状態として、クラスタ間の距離や類似度に基いて、2 つのクラスタを逐次的に併合してゆく手法。

次に群平均法による例を交えて詳しい方法を説明していく。

2.1.1　群平均法 (group average method)

以下のような 5 つのデータを階層的手法でクラスタリングする。

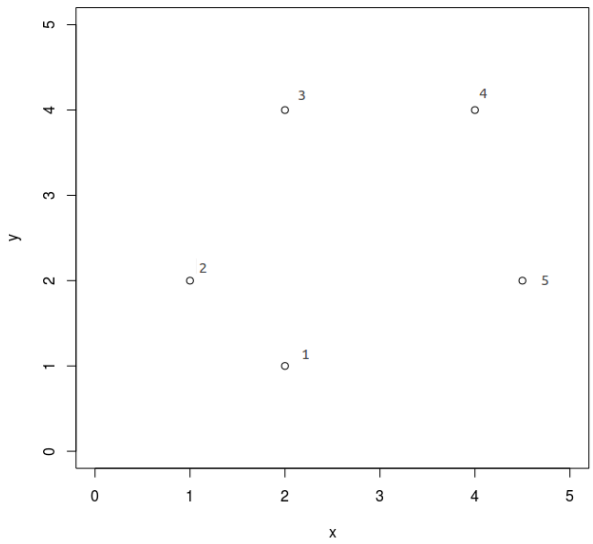

図 2.1: クラスタリングのデータ

表 2.1: データの座標

データ	座標
データ 1	(2,1)
データ 2	(1,2)
データ 3	(2,4)
データ 4	(4,4)
データ 5	(4.5,2)

階層的手法では、まず各データがそれ自身のデータにのみからなるクラスタだと考え、データ数分のクラスタを作る。上記の例では 5 つのクラスタが作成される。今、データ $x_1, X_2, \cdots, x_n$ を含むクラスタを $C(x_1, x_2, \cdots, x_n)$ と書くことにする。そしてクラスタ間の距離を測り、クラスタ間の距離行列を作成する。初期状態ではクラスタ内のデータは 1 つだけなので、クラスタ間の距離は各クラスタ内のデータ間の距離になり、距離行列は以下のようになる。

表 2.2: 距離行列 (1)

	$C(1)$	$C(2)$	$C(3)$	$C(4)$
C(2)	1.4142			
C(3)	3.0000	2.2361		
C(4)	3.6056	3.6056	2.0000	
C(5)	2.6926	3.5000	3.2016	2.0616

ここで最も短いクラスタどうしを併合する。例の場合、クラスタ $C(1)$ とクラスタ $C(2)$ が併合される。併合されたクラスタは $C(1,2)$ となる。

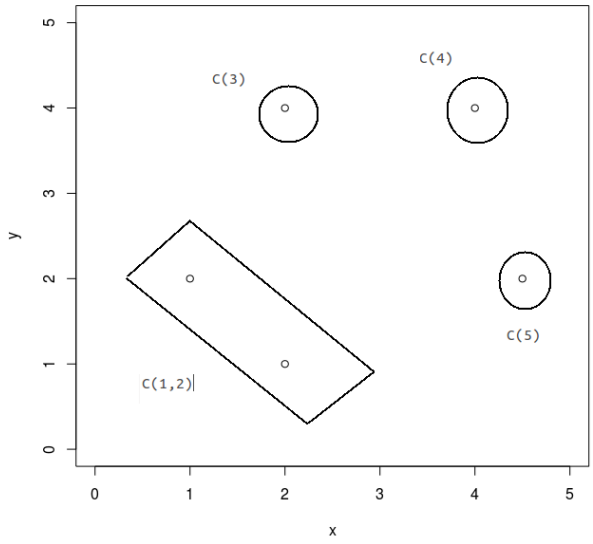

図 2.2: クラスタの併合 (1)

次に、再びクラスタ間の距離行列を作成する。このときクラスタ $C(1,2)$ には複数個のデータが入っている。群平均法ではクラスタ間の距離の測り方は次のようになる。

クラスタ A とクラスタ B の距離を $D(A,B)$ として

$$D(A,B) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} d(a_i, b_j) \tag{2.1}$$

となる。ここで $A = \{a_1, a_b, \cdots, a_m\}$、$B = \{b_1, b_2, \cdots, b_n\}$ でありデータ a とデータ b 間の距離を $d(a,b)$ で表している。つまりクラスタ A とクラスタ B 間のすべてのデータの組み合わせで距離を求め、その平均でクラス間の距離を定義している。式 (2.1) より求められる距離行列は次のようになる。

表 2.3: 距離行列 (2)

	$C(1,2)$	$C(3)$	$C(4)$
C(3)	2.6180		
C(4)	3.6056	2.0000	
C(5)	3.0963	3.2016	2.0616

再び、最も距離の短いクラスどうしを併合する。ここでは、クラスタ $C(3)$ とクラスタ $C(4)$ が併合され、クラスタ $C(4)$ が作成できる。

以下、距離行列の更新と最も距離が短いクラスタ同士の併合の処理を、クラスタ数が 1 になるまで繰り返す。

例からわかるように、階層的手法では距離行列が更新されてゆくことで処理が進む。距離行列の更新箇所は新たに作られたクラスタと既存のクラスタとの距離だけとなる。一般に、その距離を測るには、クラスタの大きさと現在の距離行列の情報だけでよいので、各データの座標の情報は必要ない。

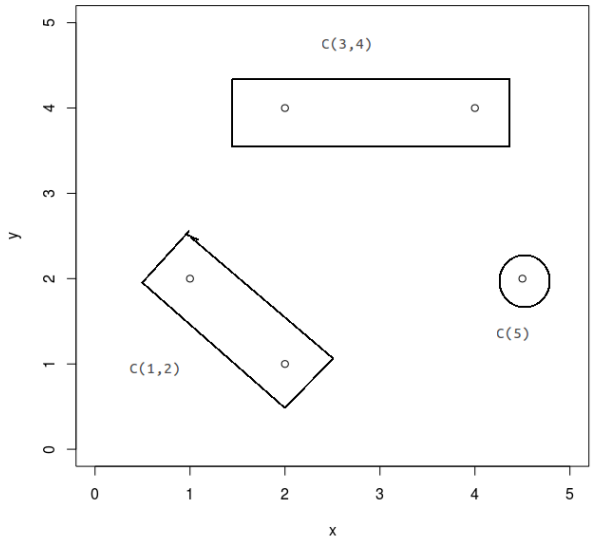

図 2.3: クラスタの併合 (2)

このため階層的手法のアルゴリズムは、クラスタ A とクラスタ B が併合されてクラスタ C できるときに、その他のクラスタ X とクラスタ C との距離を現在の距離行列から求めることに対応する。

群平均法であれば、クラスタ C とクラスタ X の距離は次式で作成できる。

$$D(C,X) = \frac{D(A,X)}{|B|} + \frac{D(B,X)}{|A|} \tag{2.2}$$

2.1.2 その他の手法

群平均法の他にもクラスタ間の距離を測る考え方があり、その測り方によって手法に名前がつけられている。

単連結法 (single linkage method)

単連結法ではクラスタ間の最小距離を与えるデータ対を選び、その距離をクラスタ間の距離とする手法である。式で書くと次のようになる。

$$D(A,B) = \min_{a_i \in A, b_j \in B} d(a_i, b_j) \tag{2.3}$$

距離行列の更新式は以下のとおりとなる。

$$D(C,X) = \min\{D(A,X), D(B,X)\} \tag{2.4}$$

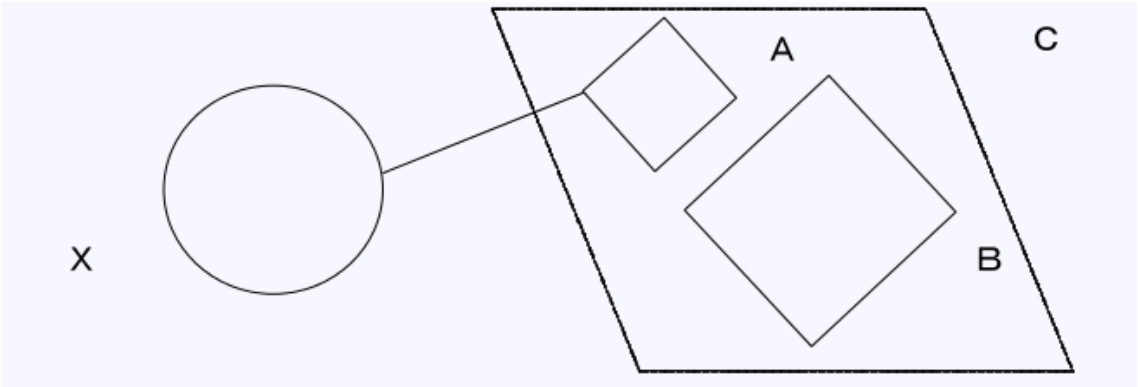

図 2.4: 単連結法のクラスタ間距離

完全連結法 (complete linkage method)

完全連結法は、単連結法とは逆に、クラスタ間の最大の距離を与えるデータ対を選び、その距離をクラスタ間の距離とする手法である。式で書くと次のようになる。

$$D(A,B) = \max_{a_i \in A, b_j \in B} d(a_i, b_j) \tag{2.5}$$

距離行列の更新式は次のとおりとなる。

$$D(C,X) = \max\{D(A,X), D(B,X)\} \tag{2.6}$$

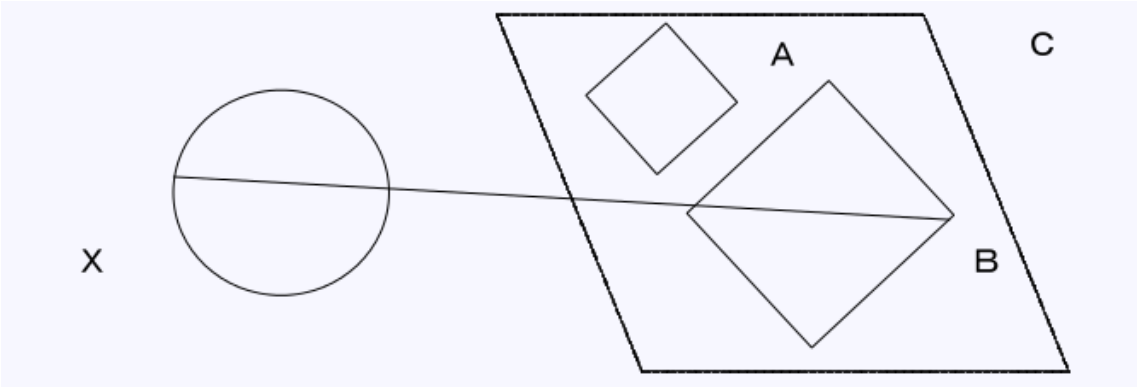

図 2.5: 完全連結法のクラスタ間距離

ウォード法 (Ward method)

ウォード法では、クラスタ A と B を併合したときに、クラスタ内の平方和の増加分が最小のものを併合してゆく。つまり、クラスタ A と B の距離をクラスタ内の平方和の増加分にとる。式で書くと次のようになる。

$$D(A,B) = E(A \cup B) - E(A) - E(B) \tag{2.7}$$

ここで、$E(X)$ はクラスタ X の平方和を意味する。平方和とはクラスタ X 内の各データ x に対して、クラスタ X の重心 $center(X)$ との距離 $d(x, center(X))$ の二乗の和である。定義は次のとおりとなる。

$$E(X) = \sum_{x \in X} d(x, center(X))^2 \tag{2.8}$$

距離行列の更新式は次のとおりとなる。

$$D(C,X) = \frac{|A| + |X|}{|A| + |B| + |X|} D(A,X) + \frac{|B| + |X|}{|A| + |B| + |X|} D(B,X) - \frac{|X|}{|A| + |B| + |X|} D(A,B) \tag{2.9}$$

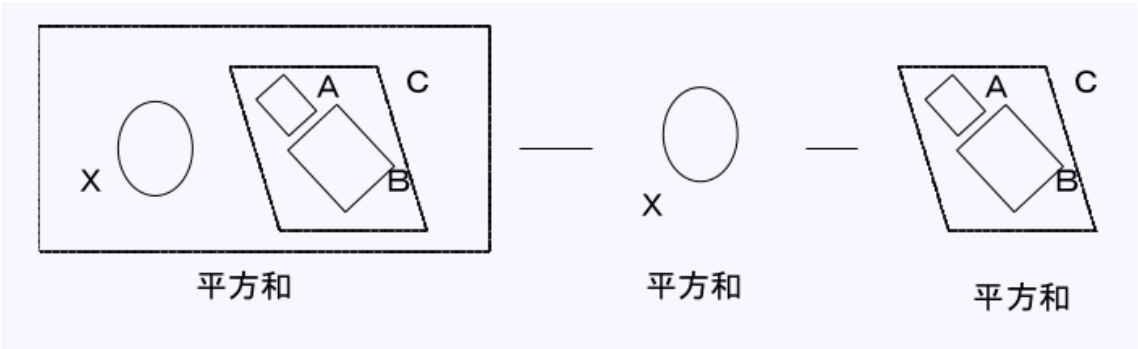

図 2.6: ウォード法のクラスタ間距離

一般に、階層的手法ではウォード法が最も精度が良いといわれている。

重心法 (centroid method)

重心法ではクラスタ A と B の距離をクラスタ A の重心とクラスタ B の重心間の距離の二乗で定義します。式で書くと次のとおりとなる。

$$D(A,B) = ||center(A) - center(B)||^2 \tag{2.10}$$

距離行列の更新式は以下のとおりとなる。

$$D(C,X) = \frac{|A|}{|A| + |B|} D(A,X) + \frac{|B|}{|A| + |B|} D(B,X) - \frac{|A||B|}{(|A| + |B|)^2} D(A,B) \tag{2.11}$$

$$D(C,X) = \frac{1}{2} D(A,X) + \frac{1}{2} D(B,X) - \frac{1}{4} D(A,B) \tag{2.12}$$

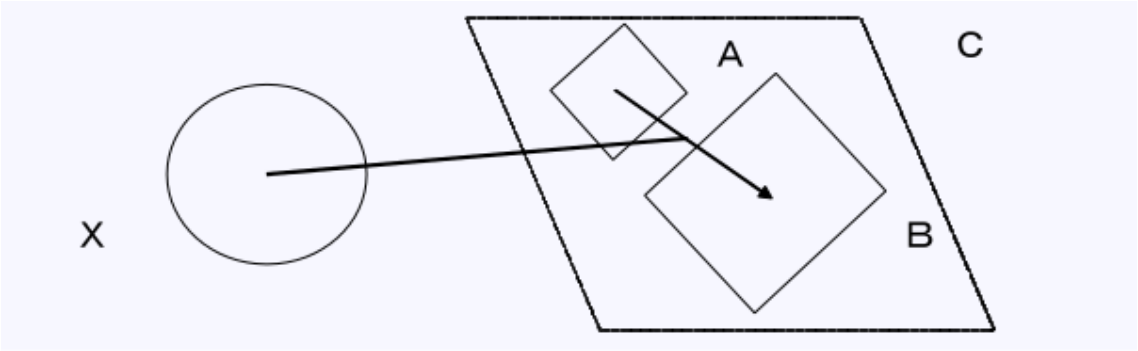

図 2.7: 重心法のクラスタ間距離

また重心法の注意として、距離の単調性がないことがあある。通常、階層的手法では、クラスタを併合する際のクラスタ間の距離は単調に増えてゆくが、重心法の場合はその単調性がない。そのために重心法によるデンドログラムでは木が逆向きになる場合がある。

メディアン法 (median method)

メディアン法は重心法とほとんど同じである。違いは重心のとり方となっている。階層的手法では 2 つのクラスタ A と B が併合されてクラスタ C が形成されるが、クラスタ A の重心と B の重心の中点をとることで、クラスタ C の重点を定義するのがメディアン法である。

距離行列の更新式は次のとおりである。

またメディアン法も重心法と同様、距離の単調性がない。

2.1.3 デンドログラム

階層的手法ではクラスタが 1 つずつ併合されてゆき、最終的に 1 つのクラスタにまとまる。この併合されてゆく過程は、デンドログラムというグラフで表現することができる。2.1.1 で例に出した 5 つのデータのクラスタリングでは、以下の過程で一つのクラスタにまとまる。

●距離 1.4142 の $C(1)$ と $C(2)$ を併合して $C(1,2)$ を作成
●距離 2.0000 の $C(3)$ と $C(4)$ を併合して $C(3,4)$ を作成
●距離 2.6316 の $C(3,4)$ と $C(5)$ を併合して $C(3,4,5)$ を作成
●距離 3.1066 の $C(3,4,5)$ と $C(1,2)$ を併合して $C(1,2,3,4,5)$ を作成

これは次のような図で表現でき、このグラフをデンドログラムという。

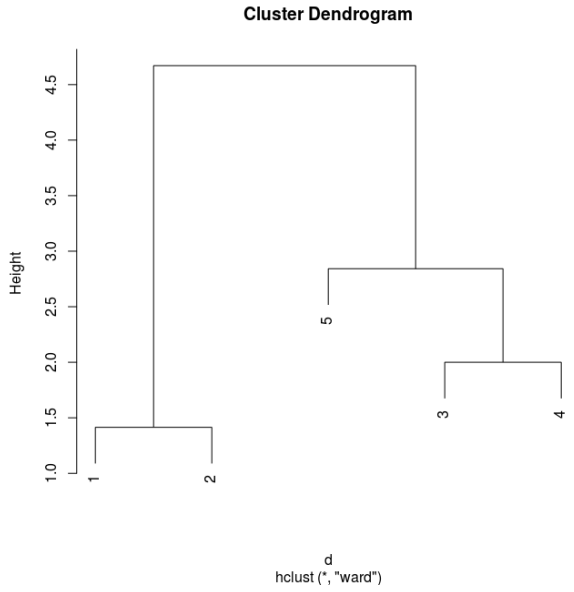

図 2.8: デンドログラム

デンドログラムの縦軸は距離を表している。クラスタどうしが併合された場合に、デンドログラムでは横棒でそれらクラスタを併合されたことを示すが、そのときの横棒の縦軸の目盛が、クラスタ間の距離を表している。

また、デンドログラムの縦軸のある位置で、グラフをカッとすることにより、目的の数のクラスタ数を得ることができる。例えば、図 (2.8) のデンドログラムを次の横線の位置ででカットすると、次のように 2 つのクラスタが得られる。

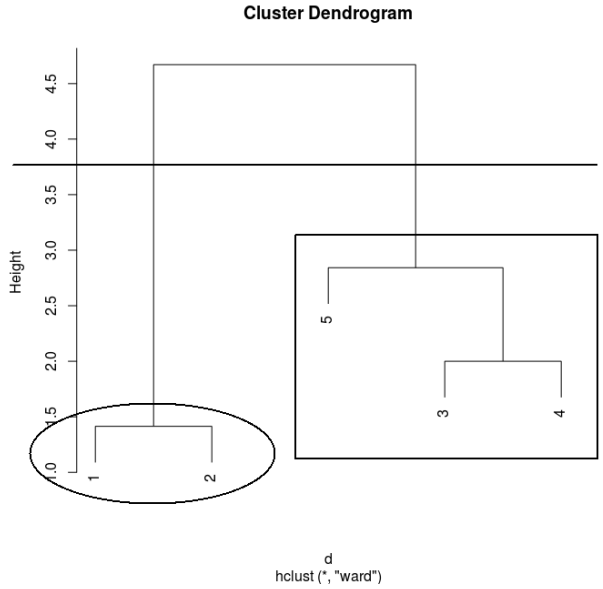

図 2.9: デンドログラムのカット

2.2 k-means 法

k-means 法は非階層的手法の代表的手法であり、現実のクラスタリングの問題で使われることが多い実用的な手法である。

アルゴリズム

注意する事として、k-means 法では分割後のクラスタの数 K を予め与える必要がある。つまり k-means の入力はデータセットとクラスタ数 K が与えられ、以下の手順で実行される。

[step 1] K 個のクラスタの代表点 $c_1, c_2, \cdots, c_K$ を適当に作成する。
[step 2] 各データ x に対して、x と c_i との距離を測り、最も距離の短い c_i に対するクラスタを x のクラスタにに設定する。
[step 3] step2 により各データ x のクラスタが変化しなければ終了。変化があったときは、各クラスタの重心を代表点 $c_1, c_2, \cdots, c_K$ に設定し step2 に戻る。

k-means 法はクラスタの重心をそのクラスタの代表点 $c_1, c_2, \cdots, c_K$ni 設定して、以下の評価関数を最小化するようなデータのクラスタへの割り当て（クラスタリング結果）を求めることに対応する。

$$\sum_{i=1}^{K} \sum_{x \in C_i} \|x - c_i\|^2 \tag{2.13}$$

実は先のアルゴリズム (step1　step3) によって式 (2.13) の値は単調に減少してゆくことが示せる。概略を説明すると、各 C_i を固定して式 (2.13) を最小化する x のクラスタへの割り当てを考えると、それは先のアルゴリズムの step2 を実効することに対応する。次に x のクラスタへの割り当てを固定して式 (2.13) を最小化 c_i を求めると、それはクラスタの重心なので、アルゴリズムの step3 を実効することに対応する。この step2 と step3 を繰り返すので、先のアルゴリズムで式 (2.13) の値は単調に減少してゆくことになる。

しかし、先のアルゴリズムによって式 (2.13) の値は単調に減少してゆくが、それが式 (2.13) の本当の最小値（大域解）であるとは限らない。式 (2.13) を最小にする解を求める問題は、NP 困難な問題であるため、大域解を求めることは非常に難しい問題である。先のアルゴリズムによって求まる解は局所解となっている。

また step1 の初期値の選び方によって、得られる解が異なる。このためにどのように初期値を選ぶかが重要となる。通常、初期値をいろいろ変化させて、解を複数個得て、それらの中から式 (2.13) の値の最も良い解を選択する。

2.3　評価法

クラスタリングの結果を評価するには、正解集合が用意されているかどうか、クラスタの数が与えられているかどうかによって、様々な方法がある。ただしどの評価方法にしても長所と短所があり、標準的な評価方法は確立されていない。

ここでは正解集合が存在し、しかもクラスタの数が既知である場合の評価方法を 4 つ（F 値、エントロピー、純度、精度）紹介する。またこれらの評価値を算出するためのクロス表についても解説する。

2.3.1　クロス表

クラスタ数 K が既知である場合、クラスタリングの結果 C は、得られた各クラスタを C_i として、以下のように表現できる。1

$$C = \{C_1, C_2, \ldots, C_K\}$$

また正解となるクラスタリング結果 A を、

$$A = \{A_1, A_2, \ldots, A_K\}$$

とする。

すると C と A を利用して、以下のような表を作成することができる。この表がクロス表である。

マス目に入ってる x_{ij} は、$|C_i \cap A_j|$ つまり C_i と A_j に共通に属するデータの個数を意味する。

表 2.4: クロス表

	A_1	A_2	…	A_j	…	A_K
C_1	x_{11}	x_{12}	…	x_{1j}	…	x_{1K}
C_2	x_{21}	x_{22}	…	x_{2j}	…	x_{2K}
…	…	…	…	…	…	…
C_i	x_{i1}	x_{i2}	…	x_{ij}	…	x_{iK}
…	…	…	…	…	…	…
C_K	x_{K1}	x_{K2}	…	x_{Kj}	…	x_{KK}

2.3.2 エントロピー (entropy)

最も標準的に用いられるクラスタリングの評価尺度はエントロピーである。

各クラスタ C_i に対するエントロピー E_i を求めて、クラスタのデータ数による重み付き平均をとることで全体のエントロピーが定義される。つまり、以下の式がクラスタリング結果のエントロピーである。この値は 0 から 1 の値をとり、値が低いほどクラスタリング結果が良好であることを意味する。

$$\sum_{i=1}^{K} \frac{|C_i|}{N} E_i = \sum_{i=1}^{K} \frac{\sum_{j=1}^{K} x_{ij}}{N} E_i \tag{2.14}$$

ここで、N はクラスタリング対象のデータ数を表す。また、E_i は以下で定義される。

$$E_i = -\sum_{h=1}^{K} P(A_h|C_i) \log P(A_h|C_i) \tag{2.15}$$

ここで確率 $P(A_h|C_i)$ がでているが、これは、

$$\frac{|A_h \cap C_i|}{|C_i|} = \frac{x_{ih}}{\sum_{j=1}^{K} x_{ij}} \tag{2.16}$$

によって推定する。

また式 (2.15) には対数 log が出ている。対数の底はいくつでもかまわないが、クラスタ数 K を使うことが多い。

2.3.3 純度 (purity)

エントロピーと同様、純度も標準的な評価尺度である。

クラスタ C_i に対する純度 P_i とは、ある正解のクラスのデータをどの程度含むかという指標であり、以下で定義される。

$$P_i = \frac{1}{|C_i|} \max_h |C_i \cap A_h| \tag{2.17}$$

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、以下が定義となる。

$$\sum_{i=1}^{K} \frac{|C_i|}{N} P_i = \frac{1}{N} \sum_{i=1}^{K} \max_h |C_i \cap A_h| \tag{2.18}$$

この値は 0 から 1 の値をとり、値が高いほどクラスタリング結果が良好であることを意味する。

2.3.4 F 尺度 (F-measure)

F 尺度は情報検索における標準的な評価方法であり、再現率と精度の調和平均で求められる。クラスタリングにおける F 尺度はそれを応用したものである。

まず、正解クラスタ A_h と得られたクラスタ C_k に対する再現率 R_{hk} と精度 P_{hk} を以下のように求める。

$$R_{hk} = \frac{|A_h \cap C_k|}{|A_h|}$$

$$p_{hk} = \frac{|A_h \cap C_k|}{|C_k|}$$

A_h と C_k に対する F 尺度 F_{hk} は R_{hk} と P_{hk} の調和平均である。

$$F_{hk} = \frac{2R_{hk}P_{hk}}{R_{hk} + p_{hk}}$$

クラスタリング結果に対する F 尺度 F は、A_h に対して、F_{hk} が最大になるような k を求めて F_{hk} を算出し、各 h に対して重み付き平均をとったものとなる。

$$F = \sum_{h=1}^{K} \frac{|A_h|}{N} \max_k F_{hk} \tag{2.19}$$

2.3.5 精度 (accuracy)

精度（あるいは正解率）は最も厳密な評価尺度といれるが、精度を測るためには、クラスタとクラスタのラベルを対応させる必要がある。クラスタの番号をクラスタのラベルに対応させる関数を f とすると、各クラスタ C_i の精度 AC_i は以下で測れる。

$$AC_i = \frac{|A_{f(i)} \cap C_i|}{|A_{f(i)}|} = \frac{x_{if(i)}}{\sum_{j=1}^{K} X_{f(i)j}} \tag{2.20}$$

各クラスタに対する精度のデータ数による重み付き平均をとることで全体の精度が定義される。

$$\sum_{i=1}^{K} \frac{|C_i|}{N} AC_i \tag{2.21}$$

ただし、クラスタとクラスタのラベルを対応させるのはそれほど簡単ではない。単純にはすべての対応づけに対して精度を測り、最も精度が高い対応づけを選べばよいのだが、クラスタの数が K だと組み合わせは $K!$ なので、K が大きい場合は経産コストが高い。K が大きい場合は組み合わせ最適化手法を利用して、良さそうな対応付けを探す。

第3章　ディリクレ分布

3.1　ノンパラメトリックベイズモデル

ノンパラメトリックモデル法 [1] とはベイズ統計の新しいベイズ統計の新しい統計もデルであり、簡単に言うと「データに応じてモデル自体の複雑さをも自動的に学習する」ことのできる統計モデルである。ノンパラメトリックベイズには以下のような特徴がある

●事前分布が確率分布でなく、確率過程（無限次元）
●パラメータがないという意味ではなく、生成や観測に応じてパラメータが増えていくモデル
●数学的に厳密に理解するには、測度論が必要

混合モデルを用いて通常のパラメトリックベイズモデルについて説明する。混合モデルでは、n 個の観測データ $x_1, x_2, \ldots, x_n$ の生成過程は以下のようにモデル化される。

$$\theta_{(K)} \sim G_0, fork = 1, \ldots, K \tag{3.1}$$

$$\pi = (\pi_1, \ldots, \pi_K) \sim Dirichlet(\pi; \gamma_1, \ldots, \gamma_K) \tag{3.2}$$

$$z_i|\pi \sim Multinomial(z; \pi), and x_i|\theta_{(z_i)} \sim p(x|\theta_{(zi)}, fori = 1, ;n \tag{3.3}$$

ここで、表記 $y \sim p$ は y が分布 p から生成されることを意味する。また $y|x \sim p$ は、x が既知の下で y が分布 p から生成されることを意味する。

まず式 (3.1) で K 個の要素分布のパラメータ $\theta_{(K)}, k = 1, \ldots, K$ が共通の事前分布 G_0 から生成される。例えば, 要素分布が正規分布の場合、$\theta_{(K)}$ はその平均と分散である。

次に式 (3.2) で、要素分布の選択確率 $\pi = (\pi_1, \ldots, \pi_K)$ がディリクレ (*Dirichlet*) 分布により生成される。π_k は第 k 要素分布が選択される確率を表す。ディリクレ分布は, 非負で総和が 1($\sum_k \pi_k = 1$) となる K 個の確率変数 (K) 次元単体上の確率変数 (K 次元単体上の確率ベクトル) の同時分布で、

$$Dirichlet(\pi; \gamma_1, \ldots, \gamma_K) = Z(\gamma)^{-1} \Pi_{k=1}^{K} \pi_k^{\gamma_k - 1}$$

で定義される. ただし,$Z(\gamma)$ は $\gamma_1, \ldots, \gamma_K$ にのみ依存する正規化項を表す。具体的には, ガンマ関数 $\Gamma(x)$ を用いて $Z(\gamma) = (\Pi_k \Gamma(\gamma_k))/\Gamma(\sum_k \gamma_k)$。また、一般には、$K$ 個の異なるパラメータ $\gamma_1, \ldots, \gamma_K$ とするが、通常、それらが全て等しい (γ) とする対称ディリクレ分布が用いられる。

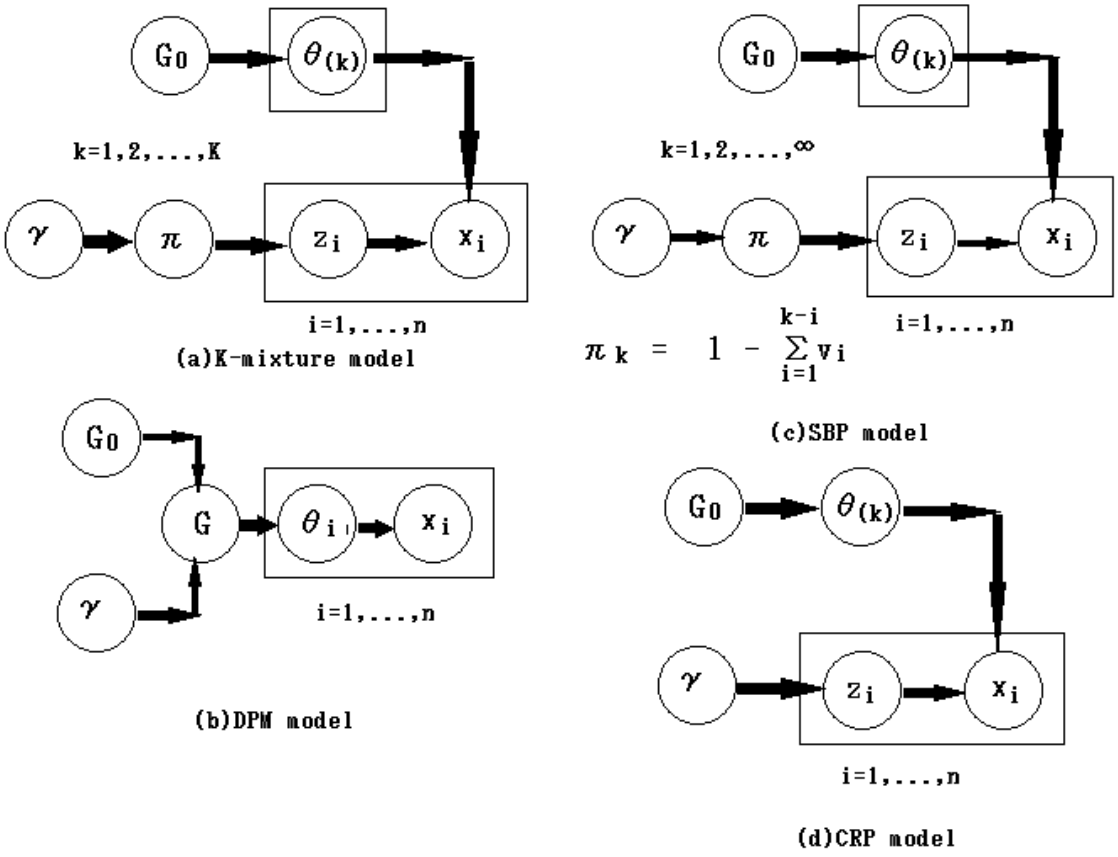

図 3.1: パラメトリック／ノンパラメトリックベイズモデル混合モデルのグラフィカルモデル

式 (3.3) では、x_i がどの要素分布から生成されるかを決める潜在変数 z_i が、π をパラメータとする多項分布 $Multinomial(z;\pi)$ で確率的に決定される (π_k は $z_i = k$ となる確率に相当し、π_k の期待値は $gamma_k/(\gamma_1 + ... + \gamma_K)$ である)。そして、z_i が決まれば対応する $\theta(z_i)$ が決まり、x_i は要素分布 $p(x|\theta(z_i))$ から生成される。上記は、一般の有限混合モデルのデータ生成過程で、応用に応じて、要素分布モデルを具体的に定め、また、それに応じ、パラメータの事前分布 G_0 も具体化される。なお式 (3.3) は θ が分布 $G(\theta) = \sum_{k=1}^{k} \pi_k \delta_{\theta_k}$ に従って生成されると言い換えることができる。ただし、$\delta_u(v)$ はデルタ関数で、$u = v$ の時は 1、それ以外は 0 となる。

まとめると、

式 (3.1) は

3.2 ディリクレ分布

ここではディリクレ分布 [1] 及び、ディリクレ過程について述べる。先にそれぞれの特徴について簡単に述べておく。

ディリクレ分布

ディリクレ分布は多項分布の事前分布としてよく使われ、パラメトリックベイズモデルにおいて事前分布を現すのによく使われる。その理由としては

●全ての確率変数の定義域が 0 以上 1 以下という前提条件を満たしている
●多項分布の共役事前分布なので、数学的に扱いやすい
●パラメータによって一様分布やデルタ分布など様々な分布を表現できる
●しかし、ディリクレ分布だけが多項分布の事前分布になれるというわけではない
●もっと自由度が欲しいときは、混合ディリクレ分布とかを使う

ディリクレ過程

ディリクレ過程 [1] は、ディリクレ分布を無限拡張したもので、ノンパラメトリックモデルの最も基本となるモデルとなっている。その特徴は

●無限次元に拡張されたディリクレ分布のようなもの
●理論的には無限次元まで増やせるという意味で、最初から無限次元を仮定するわけではない
●逐次的に次元を増やしていく仕組みが組み込まれている
●逐次的に生成するモデルだが、出てくる確率は生成した順番に無関係（交換可能）
●自然言語処理においては、語彙を無限に増やせる unigram の確率モデルとして使える
●混合ディリクレ過程を使うと、混合分布においてクラスタ数を自動的に決定できる

一番最後に述べられた混合分布においてクラスタ数を自動的に決定で切るという特徴が、こんかい WSI に活用する上で一番重要なことである。

では次より詳しく述べていくことにする

式 (3.1) の G_0 は各サイコロの目 k に対応するパラメータ $\theta_{(k)}$ を生成する分布である。即ち式 (3.1) で、まずあらかじめ $\theta_{(k)}(1 \leq k \leq K)$ が G_0 より生成される。

式 (3.2) のディリクレ分布は、「K 面サイコロ生成器」に相当し、k の目が出る確率、つまりこの目の出やすさが π_k である不平等で偏った K 面サイコロを生成する。この不平等さはパラメータ $\gamma_1, \ldots, \gamma_K$ で制御される。このサイコロと各目に対応するパラメータ $\theta_{(k)}$ をあわせたものが上述の $G_{(\theta)}$ に対応する。

式 (3.3 では、式 (3.2) で生成されたサイコロを複数回振って目を出す。第 i 回目に出た目が $\theta_{(k)}$ の時、$\theta_{(k)}$ をパラメータとして持つ要素分布 $p(x|\theta_{(k)})$ から x_i が生成される。

上記式 (3.1),(3.2),(3.3) による観測データ x_i の生成過程のグラフィカルモデルを図 (3.1 に示す。グラフィカルモデルとは、変数間の依存関係を有向グラフで表現したものである。例えば、図 (3.1)(a) で x_i は z_i からの矢印があるが、π からの矢印はない。もちろん、z_i は pi に依存するため、x_i も間接的に π に依存するが、z_i が既知 (例えば、$z_i = k$)

であれば、上記生成過程より明らかなように、x_i はもはや π に依存しない。即ち、x_i は z_i 既知の下で π と条件付き独立である。条件付き独立性より、x_i の確率分布は、矢印の起点の変数を条件として、$p(x_i|z_i = k, \theta_{(k)})$ と書ける。明らかに、これは、式 (3.3) の $p(x_i|\theta_{(z_i)})$ と等価表現である.

前節の式 (3.1) から明らかなように、パラメトリックベイズモデルでは、K 個のモデルパラメータ $\{\theta_{(k)}\}_{k=1}^{K}$ はデータを生成する前に K 種類全てが決められている。ただし、K の値は、モデル構造の尤度 $p(D|K)$ の最大化により選択される (ベイズ推定の枠組みで、尤度関数の最大化としてパラメータ（本例では K の値）を点推定する方法を経験ベイズ法と呼ぶ)。ここに D は観測データ集合を表す。実際には、候補となる幾つかの K の各々に対して学習を行った後、それらの中で $p(D|K)$ の一番大きな K を選択することになる。各データ x_i をどの要素分布から生成するかは、多項分布に従う確率 $\pi = (\pi_1, \ldots, \pi_K)$ で決まる。

これに対し、ノンパラメトリックベイズモデル、即ち、図 (3.1)(b) に示すディリクレ過程 (Dirichlet Process: DP) では、x_i 毎に θ_i が対応づけられている。つまり、データ数分の n 個のパラメータ、即ち最大 n 混合モデルまでが実現できる。もちろん、一つのデータに一つの要素分布というのは非現実的であり、実際、DP では、データが観測される毎に、要素分布数が必要に応じて増える柔軟なデータ生成過程となっている。換言すれば、$\theta_1, \ldots, \theta_n$ は全て異なるわけではなく、適応的に値の異なる K 個のパラメータ $\theta_{(1)}, \ldots, \theta_{(K)}$ から成る。即ち、$\theta_i \in \{\theta_{(1)}, \ldots, \theta_{(K)}\}$ (ここで表記に関し、$i \neq j$ に対し $\theta_i = \theta_j$ となり得るが、$\theta_{(i)} \neq \theta_{(j)}$ に注意)。つまり、 $x_1, \ldots, x_n$ が有限個 (K 個) のクラスタに分割される。DP の場合、K の値が予め固定されているわけではなく観測データに応じて定まる. この性質を用いて混合モデルを構成するモデリングがディリクレ混合過程 (Dirichlet ProcessMixture: DPM) モデルである.

第4章　ディリクレ混合過程

4.1　ディリクレ混合過程

DP[1] は確率分布に対する分布で、基底分布 G_0 と正のパラメータ γ で定義される。G が DP に従う時、$G \in DP(\gamma, G_0)$ と表記する。そして、確率変数 *theta* が G から生成される時、$\theta \in G$(即ち $P(\theta|G) = G(\theta)$) と書く。図 (3.1)(b) に、ディリクレ混合過程 (Dirichlet Process Mixture: DPM) に従うデータ生成過程のグラフィカルモデルを図示する。DPM モデルによるデータ生成過程は以下で与えられる。

$$G \in DP(\gamma, G_0) \tag{4.1}$$

$$\theta_i|G \in G, for i = 1, \ldots, n \tag{4.2}$$

$$x_i \in p(x|\theta_i), for i = 1, \ldots, n \tag{4.3}$$

DP の厳密な定義等は紙面の都合上, 参考文献に譲り, 以下では, 上記 DPM によるデータ生成過程について, 再び「サイコロ」の例を用いてより直観的に説明する.

前 2.1 節で説明したように、ディリクレ分布は多項分布のパラメータの事前分布として用いられ,「K 面サイコロ生成器」を実現するものであった. 一方,DP は直観的には K の値を固定しない, どんな面数のサイコロでも生成できる, ただし, 面数の多いサイコロほど生成されにくい「任意面サイコロ生成器」を実現する

まず、基底分布 G_0 からサイコロの目 k に対応するパラメータ $\theta_{(k)}$ として、十分多くの $\theta_{(1)}, \theta_{(2)}, \ldots, \theta(k), \ldots$ を生成する。基底分布 G_0 は離散分布でも連続分布でもよい。次に、「任意面サイコロ生成器」によって不平等で偏ったサイコロが生成される。ここで、k の目の出る確率、つまりこの目の出やすさを π_k とする。このサイコロと、対応するパラメータ $\{\theta_{(k)}\}$ で定義される離散分布が G である。すなわち、以上が式 (4.1) による、確率分布に対する分布である DP からの、確率分布 G の生成に相当する。次節で説明するように、DP においては、このようなサイコロをいくつも生成すると、これらは平均して、k が大きいほど対応する面の出やすさ π_k が指数的に小さくなるという傾向を持ち、この傾向はパラメータ γ で制御される。したがって式 (4.1) の任意面サイコロ生成においては、各サイコロのいわば実効的な面数はサイコロごとに異なり、面数の多いサイコロほど生成されにくいことになる。次に式 (4.2) により、このサイコロを振って出た目を θ_i とし、式 (4.3) により、前節同様、x_i がパラメータ θ_i を持つ要素分布 $p(x|\theta_i)$ から生成される。

4.2 Stick-breaking 過程

前節式 (4.1) の DP の直観的な説明として用いた「サイコロ生成器」は、どんな多くの面数を持つサイコロでも生成可能であった。理論的には、DP は、G_0 から生成される加算無限個の $\{\theta_{(k)}\}_{k=1}^{\infty}$ と、第 k 面の出やすさが π_k である加算無限面のサイコロを生成しているといえる。特に後者に着目すると、DP とは無限次元のディリクレ分布と見なすことができ、実際 G は、$G(\theta) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_{(k)}}$ と書けることが示されている。ただし、$\sum_{k=1}^{\infty} \pi_k = 1$ であり、したがって、有限個を除いたほとんどの k に対する π_k は限りなくゼロに近い。即ち G を生成することは G_0 から生成される $\{\theta_{(k)}\}_{k=1}^{\infty}$ と、前節での面数の多いサイコロほど生成されにくいことに対応する。以下に示す一定の条件を満たす $\{\pi_k\}_{k=1}^{\infty}$ を生成することに相当する。以下、具体的な $\{\pi_k\}_{k=1}^{\infty}$ の構成法について説明する。

$\pi_1, \pi_2, \ldots$ の構成法として、$v_k \sin Beta(v; 1, \gamma)$, および $\pi_k = v_k \Pi_{j=1}^{k-1}(1 - v_j)$ からなる Stick-breaking 過程 (SBP) を示した。ここで、$Beta()$ は区間 $[0, 1]$ で定義されるベータ分布で、ディリクレ分布における $K = 2$ の特別な場合に相当する。ベータ分布は 2 つのパラメータ $(1, \gamma)$ を持ち (一つは 1 に固定)、γ の値が小さい (大きい) 程、v_k は 1(0) に近い値が生成され易くなる。$\gamma = 1$ の時、ベータ分布は $[0, 1]$ に渡る一様分布となる.

図 (3.1)(c) に示すように、SBP では、長さ 1 の棒 (stick) を考える。まず、v_1 をベータ分布より生成し、v_1 対 $1 - v_1$ の比で棒を折り、折り取った v_1 の比の方の棒の長さを π_1 とする。明らかに、$\pi_1 = v_1$ である。次いで、v_2 を生成し、残っている長さ $1 - v_1$ の棒に対し、v_2 対 $1 - v_2$ の比で棒を折り、v_2 の比の方の棒の長さ $v_2(1 - v_1)$ を π_2 とする。以下この操作を繰り返すと、第 k 操作では、$\pi_k = v_k \Pi_{j=1}^{k-1}(1 - v_j)$ となる。長さ 1 の棒を逐次折り取った長さを π_k としているので、無限回の操作による pi_k の総和は 1 となり、$\sum_{k=1}^{\infty} \pi_k = 1$ が保証される。この過程は DP の stick-breaking 過程と呼ばれる。前節におけるサイコロの目の出やすさ π_k が、ここでの折り取った棒の長さに対応することに注意されたい。k が大きいほど棒の残りは短くなり、平均的に π_k は小さくなる。実際、SBP では $\pi_k = v_k \Pi_{j=1}^{k-1}(1 - v_j)$ で、各 v_k は独立かつ同じベータ分布に従うので、π_k の期待値は $E[\pi_k] = (1/(1 + \gamma))(\gamma/(1 + gamma))^{k-1}$ となり、k に対し指数的に小さくなる。すなわち、前節で述べたように、面数の多いサイコロほど生成されにくい。

SBP は DP の一構成法を与えるが、無限個のパラメータを予め生成しておくというのは現実的ではない。より自然で扱いやすい DP 構成法として、次節で述べる Chineserestaurant 過程 (CRP) がある。

4.3 CRP(Chinese Restaurant Process)

CRP[3] は DRM を簡単に推定する方法である。隠れ分布 G を積分消去すると、このクラスタリングは、次の CRP(Chinise Restaurant Process) と呼ばれる方法によって順番に生成されたと考えても等価なことが知られている。

CRP では、データ X_n の属するクラスタの事前確率は、これまでのデータ $x_1, x_2, \ldots, x_{n-1}$ に依存し、

$$p(z_n = k | z_1^{n-1}) = \begin{cases} \frac{n_k}{\alpha + n - 1} & (k = 1, \ldots, K) \\ \frac{\alpha}{\alpha + n - 1} & (k = K + 1) \end{cases} \tag{4.4}$$

で与えられる。ここで n_k は $z_1^{n-1} = z_1, z_2, \ldots, z_{n-1}$ の中で k が現れた回数、K は現在までのクラスタ数を表す。また、この確率は $x_1, x_2, \ldots, x_{n-1}$ の順番によらない（交換可能性）。式 (4.4) は、どのクラスタが選ばれるかの事前確率は

●そのクラスタの「人気度」n_k に比例し、かつ α に比例する確率で新しいクラスタがせいせいされることもあることを意味している。

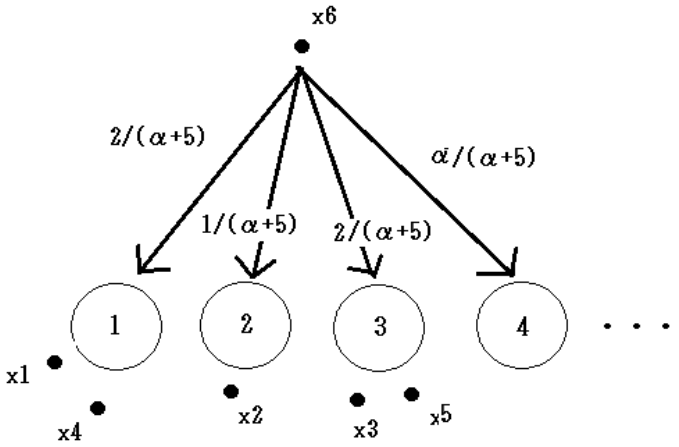

図 4.1: CRP に基づく x のクラスタの選択

いま、丸テーブルが無限にある中華料理店に「客」$x_1, x_2, \ldots, x_n$ が順番に入り、テーブルに分かれて着席する状況を考えてみよう (図 4.1)。入った客 x は式 (4.4) に従い、各テーブルの人数 n_k に比例した確率で座るテーブルを選び、α に比例した確率で誰もいない新しいテーブルに着席する。最初は必ずテーブル 1 が選ばれるが、しだいに使われるテーブル数 K が増えていくことになる。このメタファーが、CRP という名前の源になっている。整理すると

1. 最初の客 1(= データ x_1) は、テーブル 1(= クラスタ z_1) に座る
2. 次の客は [テーブルに座ってるすべての客:定数 α] の比率で既存のテーブルに座るか、新しいテーブルに座るかを決める
3. 既存のテーブルに座る場合それぞれのテーブルに座ってる客の数の比率で座る席を決める
4. すべての客 (= データ) に対し 2,3 を繰り返す

上で説明した CRP はデータ x をこれから生成するモデルであり, 事前確率だけを与えている場合の過程である。

逆に、データ x_n だけが与えられてその属するクラスタ z_n が未知のときどうなるかを考える、その事後確率はベイズの定理から、

$$p(z_n = k|x_n, z_1, z_2, \ldots, z_{n-1}) \propto p(x_n|z_n)p(z_n|z_1, z_2, \ldots, z_{n-1}) \tag{4.5}$$

となり、この式の第 2 項は式 (4.4) の事前確率そのものである。 第 1 項はクラスタ k から x_n が生成される尤度で、これは k に属する他のデータで決まり、 結局式 (4.5) は、

$$p(z_n = k|x_1, x_2, \ldots, x_{n-1}, z_1, z_2, \ldots, z_{n-1}) \propto \begin{cases} p(x_n|k) \cdot \frac{n_k}{\alpha+n-1} & (k = 1, 2, \ldots, K) \\ p(x_n|k) \cdot \frac{\alpha}{\alpha+n-1} & (k = K+1) \end{cases} \tag{4.6}$$

で求めることができる。

第5章　実験

5.1　実験概要

5.1.1　提案手法

持橋大地の 2.4 CRP に基づく DPM の学習 [3] によれば

...式 (4.6) を用いることによって、データ $X = x_1, x_2, \ldots, x_N$が与えられたとき、その属するクラスタ番号 $Z = z_1, z_2, \ldots, z_N (z_n \in 1, \ldots, K)$ をギブスサンプリングによって求めることができる。...

とあり CRP を用いた手法が提案されている。今回の実験ではそれを用いさせてもらう。

データ数を N、クラスタ数を K とする。式 (4.6) を用いることによって、データ $X = x_1, x_2, \ldots, x_N$ が与えられたとき、その属するクラスタ番号 $Z = z_1, z_2, \ldots, z_N (z_n \in 1, \ldots, K)$ をギブスサンプリングによって求めることができる。隠れ変数 z_n を適当な初期値から始めて, それ以外を条件とした条件つき確率

$$z_n \sim p(z_n | X, Z_{-n})$$

からサンプリングすることをランダムな順番ですべての n について繰り返せば、真の分布 $p(Z|X)$ に従うサンプル Z が得られる、 という方法のことである。ここで、$Z_{-n}n$ は z_n を除いた Z を表している.

今の場合、 これは「今の値 z_n をいったん忘れて」新しい z_n を式 (4.5)、すなわち式 (4.6) からサンプリングしていけば、真の z_n が求まることを意味する。これは、式 (4.4) は X の順番によらないので、x_n をいつでも最後のデータとみなすことができる (交換可能性) からである。

なのでアルゴリズムの流れとしては

(1). まず最初のデータ (今回は x_1) を適当なクラスタ (今回は z_1) に割り当てる
(2).(1) の初期値と式 (4.6) を元に他のデータをクラスタに割り当てる
(3). すべてのデータに対し (3-1)(3-2) の処理を行う。これを任意の回数繰り返す
(3-1). 選んだでデータをクラスタから削除し、クラスタ情報を更新する
(3-2). 式 (4.6) より追加先のクラスタを決定し追加する、クラスタ情報を更新する。

また改良したアルゴリズムとして (1),(2) の初めのクラスタ割り当てを適当な初期値（これ以上のクラスタ数になることはないだろうと思える値、今回は 40 を与え k-means 法でおこなったあと CRP を用いたアルゴリズムで繰り替えしたものも利用した。

5.1.2 実験方法

使用するデータ

実験では単語「見る」を対象にコーバスから「見る」の用例 621 文を抽出した。辞書の定義では「見る」の定義は 6 つある。また上記 621 文の用例中の「見る」にはその用例内での「見る」の語義が付与されている。

利用するアルゴリズム

以下の４つのアルゴリズムで WSI を行い、その結果を比較する。

(1) 提案手法で述べた方法 ： CRP

(2)CRP の初期値を k-means によって与えたもの : CRP+k-means

(3)k-means 法でクラスタリングしたもの ： k-means

(4)ward 法でクラスタリングしたもの ： ward

(1),(2) についてはそれぞれ繰り返し回数を 150、ハイパラメーター α を 5 とした。

k-mean 法は初期値に影響されるクラスタリング方法なので (3) は 10 回おこないその平均値をとる。

5.2　実験結果

得られた結果

クロス表を作成するためにクラスタ数が 6 となった時のデータを利用する。(1) のほうは８１回めを、(2) のほうは 63 回めのデータを使用した。それぞれのクロス表は表 (5.1),(5.2) のようになった。

表 5.1: (1) を 81 回処理を繰り返したクラスタリング結果のクロス表

	1	2	3	4	5	6
Z_1	19	1	1	0	0	4
Z_2	124	23	8	2	1	26
Z_3	66	7	7	0	1	12
Z_4	145	20	13	1	0	37
Z_5	72	3	5	1	0	12
Z_6	8	0	0	0	0	2

表 5.2: (2) で 63 回処理を繰り返したクラスタリング結果のクロス表

	1	2	3	4	5	6
Z_1	18	4	0	0	1	7
Z_2	21	1	3	0	0	4
Z_3	338	43	26	4	1	72
Z_4	7	0	0	0	0	1
Z_5	8	1	0	0	0	0
Z_6	42	5	5	0	0	9

それぞれのクラスタリング方法で行った結果をエントロピーと純度によって評価し比較する。それぞれの値を表 (5.3) にまとめた。

(1),(2) でのクラスタリングを行った結果、クラスタ数の推移は図 (5.1) のようになった。また繰り返し回数＝ N が 50 以上の部分を拡大したものは図 (5.1) とする。図は次のページに載せる。

表 5.3: 各クラスタリングでのエントロピーと純度

	Entropy	Purity
CRP	0.5248078	0.6988728
CRP+k-means	0.5246923	0.6988728
k-mean	0.4985224	0.7025765
ward	0.4999596	0.6988728

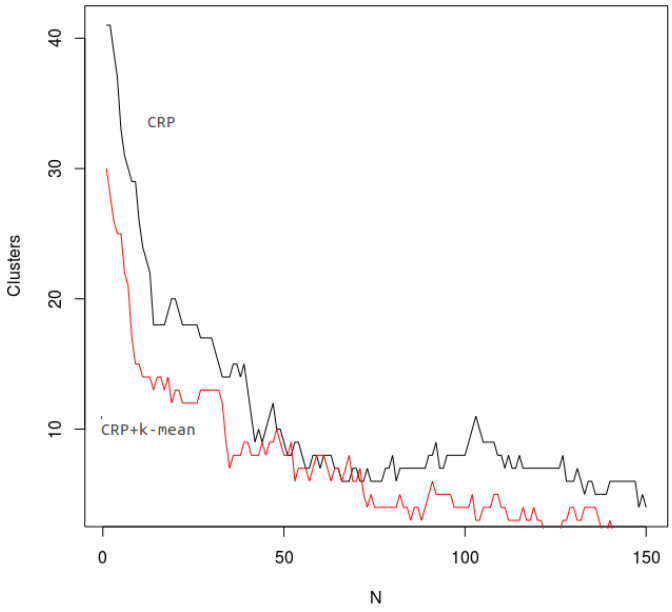

図 5.1: (CRP を用いたクラスタリングによるクラスタ数の推移

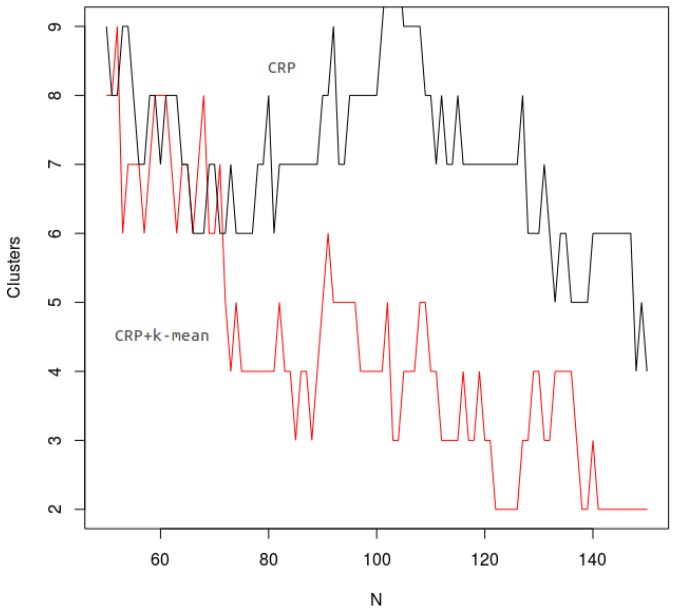

図 5.2: 図 (5.2) の N≥ 50 を抜き出したもの

実験結果について

表 (5.3) を元により。エントロピーの比較では CRP を用いたものは k-meas 法やウォード法を用いたものよりもわずかに低かった。次に純度の比較ではほぼ同じ値となっている。これらよりわずかではあるが CRP を用いたものは k-mean やが若干悪い結果となった。

CRP でクラスタリングした場合と k-mena の結果を比較すると、最初のクラスタ分けを行った場合でエントロピー、純度ともにほぼ同じ値となっている事がわかる。ただ、(1) は 81 回、(2) は 63 回繰り返しを行った後のクラスタリング結果から求めているので、(2) の方が約 2 割ほど処理時間が短い事になる。

クラスタ数の推移については図 (5.1 をみると、どちらの方法でも繰り返し回数を増やしていくと、最終的に 1 つのクラスタへと収束する結果となった。なのでともにグラフの推移が横（＝クラスタ数の変化が少ない部分）の N の値でクラスタリング結果を決めることにした。繰り返し回数 $N = 61$ の値をとることにし両方共クラスタリング結果は 8 とした。

第6章　考察

6.1　CRPによるクラスタリング

CRPを用いたクラスタリング結果の評価はk-means法やward法と比べると若干精度が悪いものの、ある程度の結果が得られたと考えられる。なのでディリクレ混合過程を利用することによりパラメータ数を自動で推定したいという目的はある程度は達成できた。

6.2　CRPの初期値としてk-meansを用いた方法

k-meansを用いた方法の方がCRPのみの場合よりも少ない計算時間で目的のクラスタ数へとなった。これより、より適切な初期値を与えることが可能となればさらに処理時間を軽減できる可能性があるのではと考えられる。

6.3　クラスタリング終了の決定方法

ただ、今回のクラスタリング結果の決定方法は図(5.1)を元に、クラスタ数の変化が安定してきてる部分から繰り返し回数を決定する方法で行った。ディリクレ混合過程を用いることによって自動に推定させることが目的なので、処理の終了条件もなんらかの基準をもうける事が望ましい。例えば、

・繰り返し回数の適切な値を求める

今回適当な繰り返し回数を決めたががある程度の結果がでたので、1へ収束していく過程である部分をとると望ましい結果となるものがとれるというのが求められる可能性があるのでは。

・クラスタ数の変化が安定していく

ある程度変化が安定したら、例えば一定回数同じクラスタ数となったり決まったクラスタ数を行き来した場合に処理を終了するようにする。

・適切なクラスタ数に収束していくようにする

といった事が考えられる。

6.4 パラメータ α

また式 (4.6) にあるようにパラメータ α を事前に決めておく必要があるが、今回は適当に 5 という値で始めた。パラメータ α はアルゴリズムを繰り返す上で新しいクラスが発生する頻度に影響するので、今回クラスタ数が 1 へ収束していった結果は α が適切では無かった可能性も考えられる。今後の課題として適切な α を求める方法がある。

第7章　おわりに

本論文ではディリクレ混合過程を用いたWSIについての手法を提案した。実験では十分な結果ではなかったものの、トピック数を自動で求めたいとう目的はある程度達成できたと考えられる。これにより、語義の変化や新しい語義について柔軟に対応でき、WSIへ役立てると思われる。今後の課題としてクラスタリングの終了条件の設定、パラメータαの設定があげられる。

参考文献

[1] 上田修功, 山田武士,” ノンパラメトリックベイズモデル”, 応用数理 VOL.17(3),pp196-214,2007.

[2] 久保拓弥,” 最近のベイズ理論の進展と応用 [I] 階層ベイズモデルの基礎”, 電子情報通信学会誌 2009 月 10 月号,pp881-885,2009.

[3] 持橋大地,” 最近のベイズ理論の進展と応用 [III] ノンパラメトリックベイズ”, 電子情報通信学会誌 2010 年 1 月号,pp73-79,2010.

[4] 持橋大地,” 生きた言葉をモデル化する-自然言語処理と数学の接点”, 月刊「数学セミナー」2007 月 11 月号 pp.33-43, 特集” 統計科学のすすめ [その 2],2007.

[5] 新納浩幸,”R で学ぶクラスタ解析”, オーム社,2007.

付 録A　ソースコード

Listing A.1: CRP.pl

```
use strict;
use warnings;

#my $INDEX_NAME = "INDEX";
my $INDEX_COUNT = 2275; #単語の種類数

my $syoki_class ="KEKKA"; #最初のくらす情報

my @files; #ふぁいる名
my %files_data; #各ふぁいるの内容 $files_data{$x_$w} ふぁいる x.の単語の頻度情報 tfw
my $files_count; #文書ふぁいるの数
my %files_class; #がどのくらすに属するか file 所属してない 0: 以外その番号のくらす 0:
my $file_bango;
my $key_files;

my $all_words_count = 0; #すべての単語のかず
my $number; #単語の種類
my $count; #単語の頻度

my %classes; #くらす $classes{$z_$w}:の単語の頻度 classzw
my %class_sonzai; #$class{$z}:にいくつでーたがあるか classz
my $key_class;

my @outfiles;
my $class_count = 0; #の数 class

#my $p; 確率#
my %p; #p{$z_$w} くらすの単語の頻度についての確率 zw

my %p_CRP;
my %p_log;

my $roop = 100; #繰返し回数
my $alpha = 5; #で用いる値α CRP;
my $tsuikasaki; #による追加先くらすた番号 CRP
my $class_MAX = 200; #くらすの上限

my $fh; my $line;

if(@ARGV < 1){
    print "引数を一つ以上与えてください\n";
    exit(0);
}
@files = sort { $a cmp $b }@ARGV;
$files_count = @ARGV;
undef @ARGV;
#print "$files_count\n";

```

```
#読込 INDEX
#open $fh, "<", "$INDEX_NAME" or die "Cannot open INDEX: $!";
#while($line = <$fh>){
# $INDEX_COUNT++;
#}
#close $fh;
#print "$INDEX_COUNT\n";

#読込 syoki_class
#open $fh, "<", "$syoki_class" or die "Cannot open $syoki_class: $!";
#while($line = <$fh>){
# if($line =~ /(\d+)\s(\d+)/){
# $files_class{$1} = $2;
# }
#}
#close $fh;

#各ふぁいるの読込
foreach my $i (1 .. $files_count){
    open $fh, "<", $files[$i-1] or die "Cannot open $files[$i]: $!";
    #print "$files[$i-1]\t$files_class{$i}\n";
    if($files[$i-1] =~ /(\d+)/){
        $file_bango = $1;
    }
    while($line = <$fh>){
        #単語をとりだす
        if($line =~ /(\d+)\s(\d+)/){
            $number = $1;
            $count = $2;
        }
        $key_files = "${file_bango}_${number}";
        $files_data{$key_files} = $count;
        $all_words_count += $count;

        #print "$key_files $files_data{$key_files}\n";
    }
    close $fh;
}
undef @files;

#文書でーたの初期くらす割当
foreach my $i (1 .. $files_count){
    keisanP();
    if($i == 1){
        tsuika2class($i,1);
        $files_class{$i} = 1;
    }else{
        $tsuikasaki = CRP($i,$i);
        #print "add to $tsuikasaki\n";
        tsuika2class($i,$tsuikasaki);
        $files_class{$i} = $tsuikasaki;
    }
    #print "$i\n";
}

#の計算 p
keisanP();

#ここまで初期設定

#でくりかえす CRP
```

```
foreach my $i (1 .. $roop){
    print "ループ回目$i\n";
    foreach my $j (1 .. $files_count){
        print "$j.tf␣class_count␣=␣$class_count\n";
        #くらすからでーた削除
        sakuzyo2class($j, $files_class{$j});
        #print "test_CRP1\n";
        print "del␣fr␣class$files_class{$j}\n";
        #を計算しなおす p
        keisanP();
        #print "test_CRP2\n";
        #により選択 CRP
        $tsuikasaki = CRP($j,$files_count−1);
        #print "test_CRP3\n";
        #print "class:$tsuikasaki\n";
        #選んだくらすに追加
        $files_class{$j} = $tsuikasaki;
        tsuika2class($j, $files_class{$j});
        #print "test_CRP4\n";
        #を計算しなおす p
        keisanP();
        #print "test_CRP5\n";
        #再推定

    }
}

#でばっく用すべてのくらすのデータ数:
foreach my $key ( sort keys %class_sonzai ){
    print "class$key␣:␣$class_sonzai{$key}\n";
}

#ふぁいるに出力
open $fh, ">", "kekka.txt" or die "Cannot␣open␣kekka.txt␣:$!";
#すべての単語について書き出す
foreach my $i ( 1 .. $INDEX_COUNT){
    print $fh "$i";
    while( my ($key, $val) = each %class_sonzai){
        $key_class = "${key}_${i}";

        #print"$key_class\n";

        print $fh "\t$p{$key_class}";

        #print $fh "\t$classes{$key_class}";
    }
    print $fh "\n";
}

close $fh;

#各でーたの所属くらす情報を書き出す
open $fh, ">", "class.txt" or die "Cannot␣open␣class.txt␣:$!";
foreach my $i ( 1 .. $files_count ){
    print $fh "$i\t␣$files_class{$i}\n";
}
close $fh;

#くらすにでーたを追加するさぶるーちん
#追加するでーたのふぁいる番号と、追加先のくらす情報
sub tsuika2class{
```

```
171     my ( $file_number, $class_number) = @_;
172
173     if(exists $class_sonzai{$class_number}){
174         #追加さきのくらすが存在する場合
175         $class_sonzai{$class_number} += 1;
176         foreach my $i (0 .. $INDEX_COUNT){
177             $key_class = "${class_number}_${i}";
178             $key_files = "${file_number}_${i}";
179             if(exists $files_data{$key_files}){
180                 $classes{ $key_class } += $files_data{$key_files};
181
182                 #print "$C{ $key_C }\n";
183             }
184         }
185     }else{
186         #存在しない場合
187         $class_sonzai{$class_number} = 1;
188         $class_count++;
189         foreach my $i (0 .. $INDEX_COUNT){
190             $key_class = "${class_number}_${i}";
191             $key_files = "${file_number}_${i}";
192             if(exists $files_data{$key_files}){
193                 $classes{ $key_class } = $files_data{$key_files};
194
195                 #print "$C{ $key_C }\n";
196             }else{
197                 $classes{ $key_class } = 0;
198
199                 #print "$C{ $key_C }\n";
200             }
201         }
202     }
203 }
204
205 #くらすからでーたを削除するさぶるーちん
206 sub sakuzyo2class{
207     my ( $file_number, $class_number ) = @_;
208     $class_sonzai{$class_number} -= 1;
209     if($class_sonzai{$class_number} == 0){
210         #そのくらすに一つもでーたが無くなった場合
211         foreach my $i (0 .. $INDEX_COUNT){
212             $key_class = "${class_number}_${i}";
213             if(exists $classes{ $key_class }){
214                 delete $classes{ $key_class };
215             }
216         }
217         delete $class_sonzai{$class_number};
218         $class_count--;
219     }else{
220         #そのくらすにまだでーたが或る場合
221         foreach my $i (0 .. $INDEX_COUNT){
222             $key_class = "${class_number}_${i}";
223             $key_files = "${file_number}_${i}";
224             if(exists $files_data{$key_files}){
225                 #でーた分の頻度を引く
226                 $classes{ $key_class } -= $files_data{$key_files};
227                 if($classes{ $key_class } == 0){
228                     delete $classes{ $key_class };
229                 }
230                 #print "$C{ $key_C }\n";
231             }
```

```
232             }
233         }
234 }
235
236 sub CRP{
237         my $x=0; #分母
238         my $ransu; my $sum=0;
239         #my $kaerichi=0;
240         my ( $file_number, $files_kazu ) = @_;
241         my $new_class = 0;
242         my $min = 0;
243         my %kakuritsu; #各くらすへ追加される確率
244         while( my ($key, $val) = each %class_sonzai){ #すべてのくらすで実行
245             #$p_log{$key} = 0;
246             #を求める p_CRP
247             $p_CRP{$key} = log( $class_sonzai{$key} / ($alpha+$files_kazu-1) );
248             #を求める p_log
249             foreach my $i ( 1 ... $INDEX_COUNT ){
250                 $key_files = "${file_number}_${i}";
251                 if(exists $files_data{$key_files}){ #に番目の単語が存在するか fileji
252                     #存在する
253                     $key_class = "${key}_${i}";
254                     $p_log{$key} += log($p{$key_class});
255                     #print "p_log = $p_log{$key}\n";
256                 }
257             }
258             #一番ちいさい値かどうか
259             if( $min > ($p_log{$key} + $p_CRP{$key}) ) { $min = $p_log{$key} + $p_CRP{$key
                    }; }
260         }
261
262         #新しいくらすについて
263         foreach my $i ( 1 .. $class_MAX){ #まだ使われていないくらす番号を探す
264             if(!(exists $class_sonzai{$i})){
265                 $new_class = $i;
266                 #print "new_class_number = $i\n";
267                 last;
268             }
269         }
270 # $p_CRP{$new_class} = log($alpha/($alpha + $files_kazu -1));
271 # foreach my $i ( 1 ... $INDEX_COUNT ){
272 # $key_files = "${file_number}_${i}";
273 # if(exists $files_data{$key_files}){ #に番目の単語が存在するか fileji
274 # 存在する#
275 # #print "1 / $INDEX_COUNT \n";
276 # #print "p_log{$new_class} : $p_log{$new_class}\n";
277 # $p_log{$new_class} += log( 1 / (1+$INDEX_COUNT) );
278 #
279 # }
280 # }
281 # #print "new_class = $p_CRP{$new_class} + $p_log{$new_class}\n";
282 # if( $min > ($p_log{$new_class} + $p_CRP{$new_class}) ){
283 # $min = $p_log{$new_class} + $p_CRP{$new_class};
284 # }
285         if( $min > 0 ) { $min = 0;}
286
287         #もとの値に戻す
288         while( my ($key, $val) = each %p_CRP){ #すべてのくらすで実行
289             $kakuritsu{$key} = exp( $p_CRP{$key} + $p_log{$key} - $min + 1 );
290             $x += $kakuritsu{$key};
291             #print("kakuritsu{$key}:$kakuritsu{$key}\n");
```

```
292     }
293     $kakuritsu{$new_class} = exp(log(1/$alpha+2) + $min);
294     $x += $kakuritsu{$new_class};
295     #print "bunbo : $x\n";
296
297     #追加先を求める
298     $ransu = rand($x);
299     #print "ransu : $ransu\n";
300     while ( my( $key, $val ) = each %class_sonzai ){
301         $sum += $kakuritsu{$key};
302         #print "sum : $sum\n";
303         if ($sum>=$ransu){ #既存のくらすへ
304             print "add to class$key\n";
305             #と p_CRPp_log,を初期化 x
306             undef %p_CRP;
307             undef %p_log;
308             #あたいを返す
309             return $key;
310         }
311     }
312     #返さずに縷ーぷをぬけたら新しいくらすへ
313     #と p_CRPp_log,を初期化 x
314     undef %p_CRP;
315     undef %p_log;
316     #追加さきのくらすを返す
317     print "add to $new_class\n";
318     return $new_class;
319 }
320
321 sub keisanP{
322     while( my ($key, $val) = each %class_sonzai){ #すべてのくらすで実行
323         #print "class$key\n";
324         foreach my $i (1 .. $INDEX_COUNT){
325             #print "$i\n";
326             $key_class = "${key}_${i}";
327
328             #print"$key_class\n";
329
330             if(exists $classes{$key_class}){
331                 #頻度が 1 以上のとき
332                 $p{$key_class} = ($classes{$key_class} + 1) / ($val + $INDEX_COUNT);
333             }else{
334                 #頻度が 0 の時
335                 $p{$key_class} = 1 / ($val + $INDEX_COUNT);
336             }
337         }
338     }
339 }
```