離散データの
確率的トピックモデル

持橋大地
統計数理研究所
daichi@ism.ac.jp

生物に見られる時空間パターンと統計
数理：同調・認知・行動
2015-1-6 (Tue)

Research Organization of Information and Systems
The Institute of Statistical Mathematics

序

本冊子は、2015 年 1 月 5－6 日に行われた統計数理研究所共同利用研究集会

「生物に見られる時空間パターンと統計数理：同調・認知・行動」

において行われた統計手法チュートリアル

「離散データの確率的トピックモデル」（統計数理研究所 持橋大地）

の講義録です。

チュートリアル講義を録音し、テープを起こし、講義で用いた該当するスライドと合わせて編集しました。なお、講義は一般に話し言葉で語られますが、講義録として読む場合、往々にして同義の書き言葉に変換したほうが読みやすいので、気付いた部分は書き言葉風に直してあります。また、講義中は指さしで「この式」となっていた部分などは、対応する数式などを明記し、読み物として通用するよう編集しました。

本冊子は、持橋氏の講義録音を丹野夕輝（岐阜大学）がテープ起こしした上で読みやすいように編集を行った文書を、三村喬生（精神・神経セ）、長田穣（東京大）、深谷肇一（統数研）がスライドを加えながら再編集し、さらに島谷健一郎（統数研）が編集・加筆した上で、持橋氏に内容を確認・修正してもらって完成となりました。

確率的トピックモデルは、広範な応用可能性を秘めながら、未だ和文教科書はおろか review paper や紹介記事も著しく不足しており、統計数理研究者以外には近寄りがたい手法となっています。トピックモデル適用に適した実データを有する非統計数理研究者が独習するさいの、いくらかの補助になるよう、本冊子を編集しました。

なお、冊子中に不適切な表現等が見受けられた場合、それはすべて本研究集会代表の島谷の責任です。

目次

自己紹介

- 統計数理研究所 数理・推論研究系
- 専門：
  統計的自然言語処理、ベイズ統計
- 略歴：
  東大基礎二→NAIST情報・松本研
  →ATR音声研→NTT CS基礎研→統数研

私の専門は統計的自然言語処理という、言葉をあつかう分野です。言葉というのはデータとしては離散データで、その扱いにはその特性に応じた統計が必要になります。そして、トピックモデルというのは、もともとは言語処理という統計数理で生まれた手法なのですが、今日では、様々な分野の様々なデータへ適用されるようになっています。生態学でも、

Valle *et al.*, "Decomposing biodiversity data using the Latent Dirichlet Allocation model, a probabilistic multivariate statistical method". Ecology Letters, 17(12), 1591-1601, 2014

のように使われ始めています。なお、私も東大の基礎二という学科にいたせいもあって生態学には関心があり、この講義をひとつのきっかけにトピックモデルが日本の生物分野へ適用されていくことになりましたら、望外の喜びというものです。

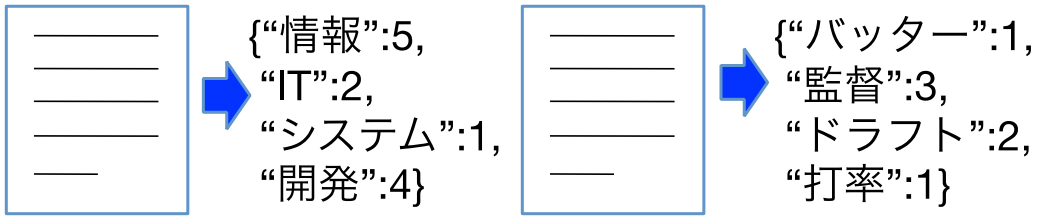
離散(計数)データは重要

- 動物、植物、昆虫、‥の観測数
- 言葉の頻度データ
- 相互に相関がある
  → ある生物種が観測されれば、他の生物種も存在する可能性が高い
  – ヒストグラムは独立ではない！

離散データは大事です。実際、動物，植物，昆虫、バイオ関連の話題など、解析対象に離散データが出てくることは本当にたくさんあります。しかし普通の統計解析ではガウス分布などを仮定するため、最初から連続値として取り扱うことが多い。そうではなく、離散データそのものに対してまともな統計モデルを作りたいという要求が結構多いのではないかと思います。とりわけ、データ相互に相関があるとき、例えば生物の例ですと、ある種のクジラが観測された場合、見えてはいないが他の種の魚も生息していると考えられます。この場合、観測種ごとの観測数のヒストグラムの間は独立でなく、相関があります。それをどうやってモデル化していくかというのが、高次元の離散データを扱ううえでの本質的な問題です。

“Bag of words” データ

- 文書の中に同時に現れる単語群には相関がある
  → 文書の中に現れた“単語”とその頻度



- 生態学では、文書＝観測ユニット、単語＝生物
  と読み替えてOK

統計的言語処理の分野では、データを Bag of words として表します。言葉というのは、テキストがあり、テキストに単語が書いてあるというものです。Bag of words の考え方では、単語の順番よりも、どんな単語が出てくるかの方に着目します。例えば「情報」が 5 回、「IT」が 2 回、「システム」が 1 回と出てきたら、なんとなく IT 系の文章だとわかります。また、「バッター」、「監督」、「ドラフト」、「打率」などが出てくると野球の話だとわかります。この中の「バッター」と「監督」の間には高い相関があって、それをどうとらえていくかが問題になります。言語処理で扱うデータでは、単語の種類は 1 万ではおさまらなくて、Google などのデータでは 1 千万を簡単に超えるような、超高次元です。ここでは言語データを実例として話しますが、例えば生態学では、文書は観測ユニット、単語は生物種で、単語の出現回数はどの種が何回出てきたかという頻度、のように自分が関わっているデータの事例に読み替えて実例を見ていただけると、理解もしやすいことと思います。

Bag of Words の行列表現

$$d_1 \to \{w_1 : 1, w_5 : 2, w_{27} : 1, w_{105} : 1\}$$
$$d_2 \to \{w_2 : 1, w_3 : 1, w_{15} : 4, w_{32} : 3\}$$

のような情報を行列で表してみる

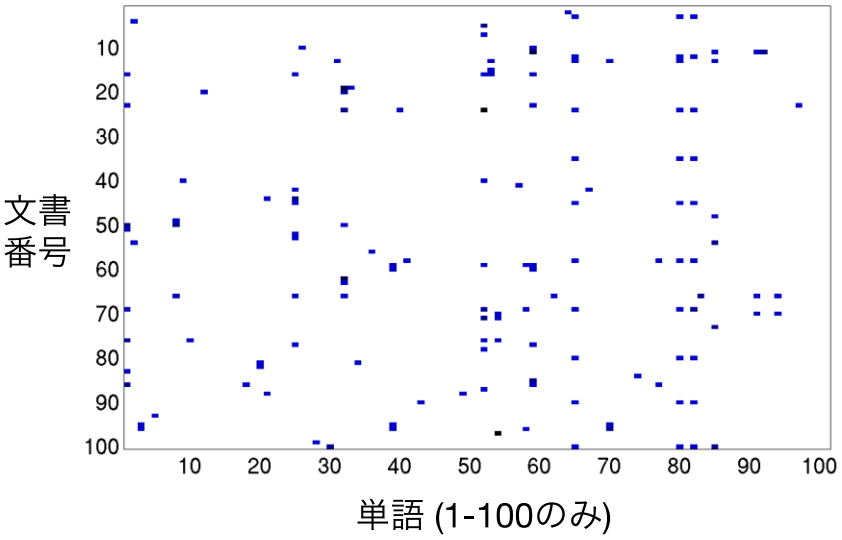
$$N \begin{pmatrix} & 1 & 1 & & & & \\ 1 & & & & & 1 & 1 \\ 2 & 4 & & & & & \\ & 2 & 1 & 1 & & & 1 \\ & & & & 1 & & \end{pmatrix} = D$$

W

文書を Bag of words としてデータ化するという作業は、行列で表すと簡単にできます。横方向の列、つまり図の W に単語１、単語２・・と単語の種類をとり、縦方向の行、図の N に一個一個の文書の番号をとり、一個の文書を行、すなわち単語を表す W 次元ベクトルに対応させます。例えば、最初の文書には２番目の単語が１回、３番目の単語が一回出てきています。次の文書には最初の単語が１回、しばらく０回で、最後にまた１回ずつです。概して文書を行列表現すると０が多くなります。０が多い行列をスパース(疎)な行列といいますが、そんな行列の扱いが、この分野での一つの大きな問題となります。

Bag of Words (実際のデータ)

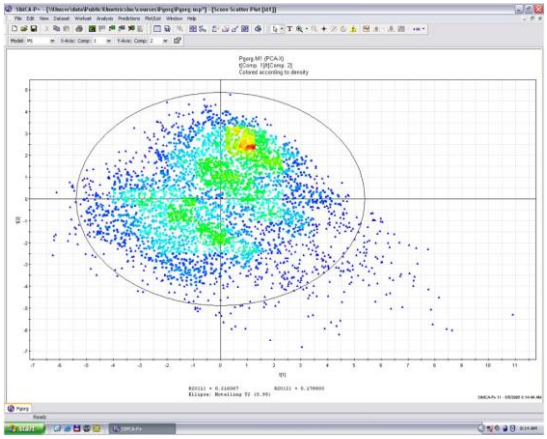
- DailyKOS (新聞)のよく使われるデータセット (一部)

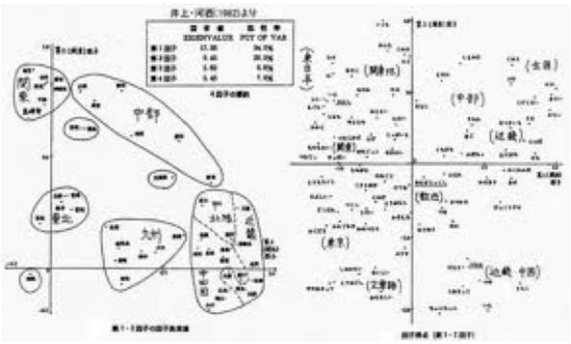


- ほとんどの値は0
- 非負の値も1か、
  非常に小さい値

上の図は実際のデータの例ですが、ご覧のとおり、スパースになっています。この図は DailyKOS という有名なデータセットの一部で、新聞記事のデータから単語を百個、文章を百個選び、その単語の出てきた文書の行に点を入れてあるのですが、このように、ぽつぽつしたゴマ塩みたいなスパース行列が出てきます。縦の列の中に多く出ている単語がありますが、それは例えば日本語の場合では、「は」や「が」など、どの文章にも出てくる単語が対応します。一方、一般の内容的な言葉は出てきたり出てこなかったりするため、ほとんどの値は 0 です。また、出現したとしても、1 などの小さな数値になりがちです。カウントデータとして、ポアソン分布のような特徴を示す場合が多くなります。

古典的な分析法: 多変量解析





- 相関分析、回帰分析、判別分析、・・・・
  - 一部の変量を抽出して分析
  - 暗黙にガウス分布を仮定 (離散で非負なのに!)

Research Organization of Information and Systems
The Institute of Statistical Mathematics

このようなデータには、古典的には多変量解析という伝統があると思います。しかし、相関分析や回帰分析や判別分析には、問題があります。一つには、一部の変数だけを取り出して分析すること。もう一つは、暗黙のうちにこういう解析手法ではガウス分布を仮定することです。本当のデータは離散でかつ非負なのに、無理やりガウス分布に合わせて使ってしまいます。それでも、カウント数が多ければあまり問題は起こらないのですが、生態学などでは、レアな種が 1 回だけ観察されたり 2 回だけ観察されることは普通に見られ、かつ、そうした情報は重要な意味を持っています。そういう情報を、普通の多変量解析では全く無視してしまいます。そうではなく、1 回や 2 回という情報までしっかり全部とらえたいというのが、確率的トピックモデルの一番の目標です。

何がだめなのか?

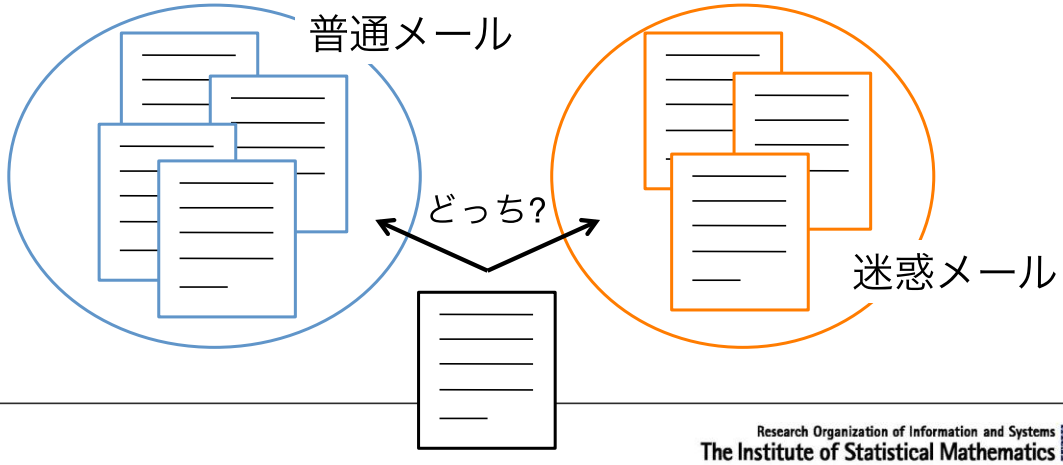
- 観測データの「由来」を説明していない
  - 回帰分析、PCA等は「後付け」の解析手法
  - 手法に発展性がない　性能も低い
- 離散、正の観測データを生み出す確率モデルを先に考えよう
  - モデルに従って、カウントが確率的に生成
  - モデルのパラメータ θ を推定
  - 複雑なモデルが作れる

従来の多変量解析の何が最大の問題かというと、観測データが来たときに、後付けの解析手法になっているという点です。回帰分析や主成分分析は、とりあえずガウシアンだという間違った仮定を置いて適当にやるという後付けの解析手法でしかありません。そのため、そのあとの発展性がない。つまり、モデルがないので、それで解析をやりましたという以上のものがない。それに対して、そもそも離散で非負の整数の観測データを生み出すような確率モデルを先に考えましょう。そのモデルに従って、カウントデータは確率論的に出てきたと思うのです。そして、そのモデルの未知パラメーターを推定します。そうすれば、より複雑なモデルも作れるし、それにより性能も上がるということになります。

こういう言語処理で統計処理をやるようになったのは大ざっぱに言うと、90 年代の終わりから 2000 年ぐらいで、この話の最後に出てくる Latent Dirichlet Allocation model は、2003 年ぐらいに出た話です。今それが非常に複雑なレベルになっていますが、たかだかここ 10 年ぐらいの話と思ってかまいません。

簡単な例: ナイーブベイズ

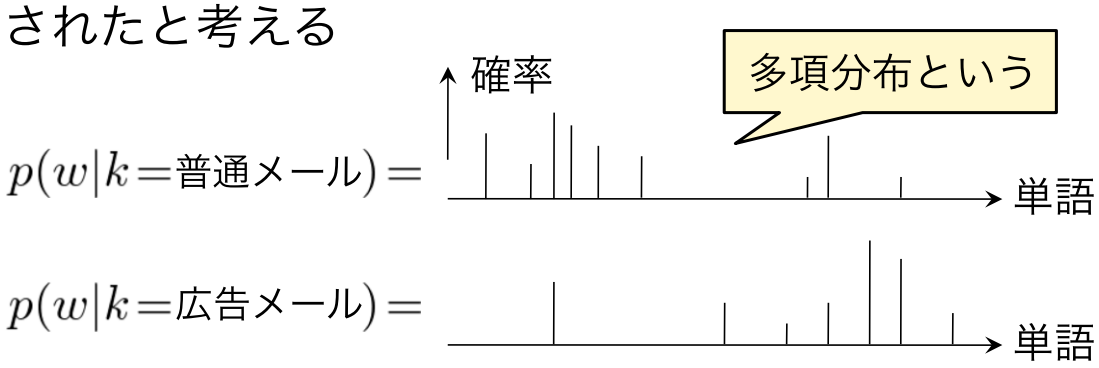
- ナイーブベイズ (Naïve Bayes)
  ‥テキストの非常に簡単な生成モデル
- いま、普通のメールと迷惑メールを分類する問題を考えよう



確率的トピックモデルの一番簡単な例は、ナイーブベイズです。これは有名な例なので、皆さんもネットなどで見たことがある人も多いと思います。電子メールでは、普通のメールに混じって広告などの迷惑メールもいっぱい来るのですが、それを分類するという問題を考えます。すなわち、普通メールの集合と迷惑メールの集合があったときに、新しいメールをどちらに分類するかという問題です。

ナイーブベイズ (2)

- いま、カテゴリ$k \in$ {普通メール, 広告メール}毎にメールの単語が確率分布$p(w|k)$ から生成されたと考える



$$p(\mathbf{w}|k) = \prod_{n=1}^{N} p(w_n|k) \quad \text{(BOWの仮定)}$$

いま、カテゴリは、普通メールと迷惑メール（広告メール）の 2 つです。そして、それぞれで単語の出方が違います。例えば、普通メールでは単語の分布が上の図のようになっているとします。縦軸はある単語が出現する確率を表し、すべての単語についての和は 1 になっています。このような確率分布を、多項分布といいます。それに対して、広告メールでは、例えば「激安」や「特価」や「バイアグラ」などの単語の出現確率が高く、全然別の分布をしているでしょう。単語の分布が違っており、広告メールは広告メール側の分布から単語をランダムにサンプルしてできたと思い、普通メールは普通メール側の分布から単語をランダムにサンプリングしてできたと思います。ここで、単語の選ばれ方は独立と仮定します。そしてデータを、単語の順番を無視して、単なる使われた単語のカウント数というベクトルと考えます。すると、カテゴリ k が決まれば、このベクトルが出てくる確率は、一個一個の単語が選ばれる確率を単にかけ算した値になります。これが、Bag of words (BOW)という仮定です。

ナイーブベイズ (3)

- 生成モデルとしては、
  (1) $k \sim p(k)$ からクラス $k \in \{0,1\}$ を選択
    - たとえば、$p(k) = [0.9, 0.1]$
  (2) for n = 1 … N,
    n番目の単語 $w_n \sim p(w|k)$ を生成.

$$p(d,k) = p(k)p(d|k) = p(k)\prod_{n=1}^{N} p(w_n|k)$$

- パラメータ $p(k), p(w|k)$ は簡単に推定できる.

BOW の仮定の下では、普通メールと迷惑メールの生成モデルを簡単に作ることができます。まず、普通メールか迷惑メールかの割合をあらわす確率分布、例えば普通メール(k=0)は確率 0.9、迷惑メール(k=1)は確率 0.1 で出てくるという確率分布があり、そこから、どちらのメールを作るかというカテゴリ k をサンプリングします。k として普通メールが選ばれたら、p(w|0)という多項分布からランダムにこの単語、この単語・・、と順に n = 1, 2, …, N 番目の単語をサンプリングしてメールを作ります。迷惑メール(k=1)が選ばれたら、迷惑メールの確率分布 p(w|1)から単語を選びます。結局、あるメール（文書: 以下、単語の集合は文書と呼ぶようにします）が生成される確率はどうなるかというと、それは、ある文書 d とそのラベル k の対の同時確率となり、そこでは、まずラベルが決まり、それからそのラベルのもとで文書が作られます。そのラベルが選ばれる確率を p(k)とし(ここでは、p(k)=[0.9, 0.1]でした)、その k のもとで単語 w が選ばれる確率を p(w|k)とすると、それらは n 回独立に出てくるので、全体の同時確率は以下の式になります。ここで、p(k)と p(w|k)は本当は未知のパラメータです。

$$p(d,k) = p(k)\prod_{n=1}^{N} p(w_n|k)$$

ナイーブベイズ (例)

$$D = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 \end{array} \\ \begin{pmatrix} 1 & 2 & 1 & & 1 & & \\ & 2 & & & 1 & 1 & 1 \\ 1 & & 1 & 1 & & 2 & \end{pmatrix} \end{array}$$

普通メール

広告メール

- これから、
  - $p(k=0) = 2/3,\ p(k=1) = 1/3$
  - $p(w|k=0) = [0.1\ 0.4\ 0.1\ 0\ 0.2\ 0.1\ 0.1]$
    $p(w|k=1) = [0.2\ 0\ 0.2\ 0.2\ 0\ 0.4\ 0]$
- ex. $p(d_1, k=0) = \frac{2}{3}\left(\frac{1}{10}\cdot\frac{4}{10}\cdot\frac{4}{10}\cdot\frac{1}{10}\cdot\frac{2}{10}\right) = 2.13\times10^{-4}$

データがあれば、未知パラメータの値は以下のように推定できます。単純には、単に数えるだけです。とても簡単な例ですが、上のようなデータ D があったとします。すなわち、メールが 3 つ(d_1, d_2, d_3)あり、単語の種類は 7 個(w_1~w_7)です。最初の二つ(d_1, d_2)が普通メールで、最後(d_3)が広告メールです。文章のカテゴリを選ぶ確率 p(k)は、一番ナイーブには、普通メール(k=0)が 2/3、迷惑メール(k=1)が 1/3 と推定できます。普通メールの中でどういう単語が出てくるかの推定は、やはり一番ナイーブには、普通メールの中での単語の頻度を単に足して 1 になるように正規化した、以下のような分布になります。

$$p(w|k=0) = [0.1, 0.4, 0.1, 0, 0.2, 0.1, 0.1]$$

迷惑メールのほうの p(w|k) は、3 番目のメールの単語のカウントを正規化して、以下のような分布になります。

$$p(w|k=1) = [0.2, 0, 0.2, 0.2, 0, 0.4, 0]$$

そうすると、1 個 1 個のメールが生成される確率を計算することができ、例えばメール d_1 の確率は、最初に普通メールである確率が 2/3 で、かつ、最初の単語が出てくる確率は 1/10、次に確率 4/10 の単語が 2 回出てくるので 4/10×4/10、次に 1/10、最後に 2/10 が一回掛け算され、2.13×10^{-4} という確率が求まります。

ナイーブベイズ (5)

- 新しいメール d をどちらに分類?
  → 確率 $p(k|d)$ が高い方に分類すればよい。
- $p(k|d)$ の計算?

さて、こんな計算がなんの役に立つかというと、それは、新しいメールのカテゴリを分類するときです。与えられた文書 d のカテゴリ k が 0 なのか 1 なのか、その確率 p(k|d)を、p(k)と p(w|k)から、計算するのです。

ところで、さきほど求めたのは、同時確率 p(d,k)や p(w|k)や p(k)です。p(k|d)では、k が条件を表す記号 | の左側にあります。k が条件の方にいる条件つき確率ではなく、ある d の下で k である確率を知りたいわけです。k の位置がひっくり返っているのですが、これは最近だとどこかで見たことのある人も多いのではないでしょうか。そう、ベイズの定理です。

ベイズの定理

- $p(A,B) = p(A|B)p(B) = p(B|A)p(A)$ より、

$$p(A|B) = \frac{p(A,B)}{p(B)}$$

$$\propto p(A,B) = p(B|A)p(A)$$

$p(A|B)$ を $p(B|A)$ で書ける！！

  - $p(B)$は、Aについての和を1にする正規化定数
  - 条件つき確率を引っくり返せる!!

ベイズの定理では、A と B の同時確率は、まず B が出て、それから B の下で A が出てくるという確率で書けます。

$$p(A,B) = p(A|B)p(B)$$

この式から、p(A|B)は p(A,B)を p(B)で割り算した形で書くことができます。

$$p(A|B) = \frac{p(A,B)}{p(B)}$$

p(A,B)を、先に A が出てそのもとで B が出てくる確率と考えても同じなので、

$$p(A,B) = p(B|A)p(A)$$

とも書けます。この p(A, B)を上に代入します。さらに、今は p(A|B)では B の下での A の確率の話をしており、B の確率 p(B)には関心がないので、これは単なる定数ですから、比例式の形にして p(A|B)を、A と B をひっくり返した p(B|A)と p(A) の積で書けることになります。

$$p(A|B) \propto p(B|A)p(A)$$

このように、ベイズの定理で、条件付き確率を逆にすることができるわけです。

ナイーブベイズ (5)

- 新しいメール d をどちらに分類?
  → 確率 $p(k|d)$ が高い方に分類すればよい。
- $p(k|d)$ の計算?

$$p(k|d) \propto p(d|k)p(k)$$

  - $p(k)$ も $p(d|k) = \prod_{w \in d} p(w|k)$ もわかっている!

今知りたいのは p(k|d)ですが、先ほど出てきたモデルのパラメータは p(k)と p(w|k)で、k の位置が逆です。それでも p(k|d)を求めたかったら、このベイズの定理を使えばよいわけです。

$$p(k|d) \propto p(d|k)p(k)$$

いま、カテゴリの出現確率である p(k)はわかっているし、カテゴリ k から文書 d が生成される確率 p(d|k) は、単語は独立だと思っているので(BOW の仮定)、単語の出てくる確率 p(w|k)を単にかけ算すれば求まります。

ナイーブベイズ (6)

- 前の例で、d=$\{w_1, w_6\}$ のとき、これは何メール?

$$p(k|d) \propto p(d|k)p(k)$$

$$= \begin{cases} \frac{2}{3} \cdot \left(\frac{1}{10} \cdot \frac{1}{10}\right) = \frac{2}{3} \times 10^{-2} \\ \frac{1}{3} \cdot \left(\frac{2}{10} \cdot \frac{4}{10}\right) = \frac{8}{3} \times 10^{-2} \end{cases} \propto \begin{cases} 0.2 \\ 0.8 \end{cases}$$

- よって、$p(k|d) = [0.2, 0.8]$
  → dは広告メール.

例えば、さきほどからの例で、単語 1 と単語 6 が 1 回だけ出てくる新しい文書が来た場合、これが普通メールか迷惑メールか知りたかったら、それぞれの確率を計算します。k=0 だという確率は 2/3×(1/10×1/10)に比例しますし、k=1 であるという確率は、1/3×(2/10×4/10)に比例します。これらを和が 1 になるように正規化すればそれぞれの確率となり、それらは 0.2 と 0.8 となっています。これが、この文書が持っている k の確率分布、事後分布と呼ばれるものです。両者を比べると、これはおそらく迷惑メールだろうと推定できます。こういうものが実際のメーラーにも入っていて、よく使われています。

さて,

- 今までは、テキスト(メール)に {普通,広告} のようなカテゴリが与えられている場合を考えてきた

  ⇩

  実際にはそんなことはほとんどない!

これまでの方法では、文書とラベル k の一対一対応があるデータセットを使って、k の出現確率 p(k)と、ラベルが k の文書に含まれる単語の出現確率分布 p(w|k)を推定しておかないと、p(k|d)が計算できません。しかし、一般の文書では、こうしたラベル k は与えられていません。文書に最初からこれは「政治」、これは「スポーツ」、これは「家庭」というラベルは貼られていたりしないのが普通ですね。次は、この場合、どうするか？という話になります。

Unigram Mixtures (Nigam+ 2000)

- ナイーブベイズの式

$$p(d,k) = p(k)\prod_{n=1}^{N} p(w_n|k)$$

において、kを推定すべき変数(潜在変数)とする

$$p(d) = \sum_{k} p(k)\prod_{n=1}^{N} p(w_n|k)$$

  - kに関して和をとって周辺化

Unigram Mixtures (ユニグラム混合モデル)という手法があります。先ほどのナイーブベイズの式では、ある文書とそのラベル k ではまずラベル k が決まり、それからそのラベルのもとでの単語がそれぞれかけ合わされて、確率が決まっているのでした。

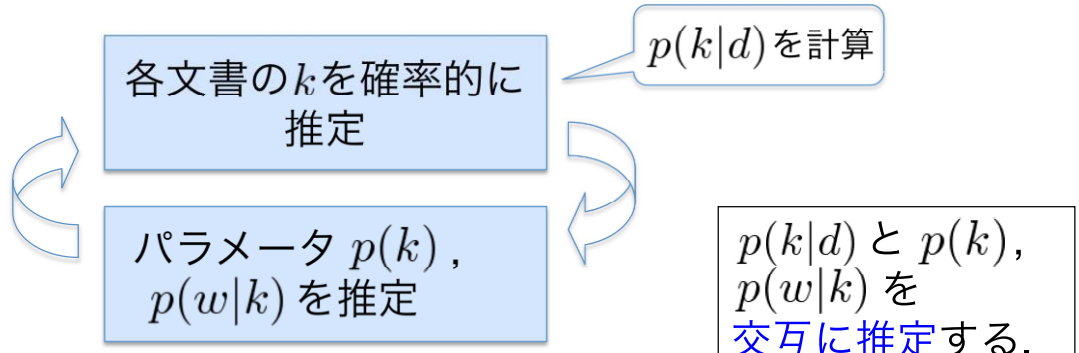
$$p(d,k) = p(k)\prod_{n=1}^{N} p(w_n|k)$$

ここで、もしそもそも k がわからないのなら、すべての k について確率の和をとってしまえばよいではないか、と考えます。これを周辺化といいますが、そういう確率モデルを作るのです。

$$p(d) = \sum_{k} p(k)\prod_{n=1}^{N} p(w_n|k)$$

この p(d)は「文書 d が、全てのカテゴリ k を通して生成される確率」です。

Unigram Mixtures (2)



- 手順:
  (0) p(k|d) を乱数で設定.
  (1) p(k|d) から、p(k), p(w|k) を推定.
  (2) p(k) p(w|k) から、p(k|d) を再推定.
  (3) 収束していなければ、goto (1).

ナイーブベイズに比べて、Unigram Mixtures は一般的にあまり知られていないのですが、それは、これを解こうとすると単に数を数えるだけでは無理で、例えば EM アルゴリズムなどを使わないと解けないからです。EM アルゴリズムについてここでは詳しくは説明しませんが、大ざっぱに言うと、まず､各文書に初期値p(k|d)を与えます｡ある文書がどういうkをラベルとして持っているかという、今まさに知りたい値を、最初は適当に設定しておきます。それらから、確率分布に関する計算式を使って、パラメータ p(k), p(w|k)を計算します。つまり、先ほどは一個一個の文書が持っている確率分布は既知、言い換えるとデルタ関数のように迷惑メールか普通メールかで 1 か 0 かのどちらかになっているものを仮定したわけですが、一般にはデルタ関数ではなく、わからないので何か大ざっぱな初期値を与え、そこからパラメータ p(k), p(w|k)を推定します。すると、それらから、先ほどのようにベイズの定理で p(k|d)を再推定できます。このような操作をぐるぐる回して、交互に推定を繰り返すのです。すると、ほどなくすべての値が一定の値に収束していきます。こういう手法を、EM アルゴリズムといいます。

パラメータ p(k), p(w|k)を計算する部分は、基本的に先ほどと同じでデータのカウント数に関する割り算ですが、これは多項分布における最尤推定（データの確率を最大にするパラメータを求める作業）を実行していることに相当します。一般の EM アルゴリズムでは確率の最大化を行うので、ここは M-step と呼ばれます。一方、これらのパラメータによる p(k|d)の計算は、期待値を求めているので、EM アルゴリズムでは E-step と呼ばれます。

Unigram Mixtures (3)

	d_1		d_2		d_3	
Step	k=0	k=1	k=0	k=1	k=0	k=1
1	0.4000	0.6000	0.6000	0.4000	0.4000	0.6000
2	0.4642	0.5358	0.6902	0.3098	0.2520	0.7480
3	0.7034	0.2966	0.8746	0.1254	0.0304	0.9696
4	0.9797	0.0203	0.9924	0.0076	0.0000	1.0000
5	1.0000	0.0000	1.0000	0.0000	0.0000	1.0000

- 計算を続けると、
  - ステップ t=5 程度で収束
  - ほぼ何も教えていないのに分類できた!!

最初の例では、最初の二つが普通メールで最後が迷惑メールだとわかっているとしましたが、それを知らずにデータだけ与えられたときに、一個一個のメールをどちらに分類するかという問題を考えます。EM アルゴリズムを用いる方法では、d_1, d_2, d_3 という 3 つの文書について、文書 1 は k=0 か k=1 かわからないので、ここでは 0.5 を中心に適当に 0.4、0.6 としました（Step 1）。d_2 は 0.6 と 0.4、d_3 は 0.4 と 0.6 と、やはり適当に置きました。この初期値をもとに一回計算し直すと、確率分布が変わりますが、k=1 の方が少し大きい値のままなので、どうも d_1 は k=1 の確率が高いように思えます（Step 2）。逆に、d_2 では、k=0 がさらに増えて k=1 が減るので、k=0 らしい。d_3 では、k=0 が減って k=1 が増えるので k=1 らしい。こういう風に、少しずつ変わるわけです。さらにもう一回計算すると、d_1 では大小が逆転し、k=0 が増えて k=1 より大きくなっています（Step 3）。他では元から大きかったところがさらに大きく、小さかったところはさらに小さくなっています。この操作をぐるぐる回すとこの例では 5 回くらいで収束して、文書 1 はカテゴリ 0 つまり普通メールである確率がほぼ 1、文書 2 が普通メールである確率もほぼ 1、文書 3 は迷惑メールである確率が 1 となっています。

Unigram Mixtures (9)

- Unigram Mixtures‥1つの文書に潜在トピックが1つある、最も簡単なトピックモデル
  - 離散データのクラスタリング
- パラメータ:
  - *p(k)*：トピックkの事前分布
  - *p(w|k)*：トピックkから出る単語の分布

最初ラベルは与えなかったのですが、まるで与えたかのように分類できてしまうのです。ただし、あくまで各文書があるラベルを持つ確率が 1 に収束するというだけで、そのラベルに「普通メール」という名前が自動的につくわけではありません。

以上が Unigram Mixtures と呼ばれているもので、いわゆる K-means という手法とほぼ等価です。何も教えていないのですが、一個一個のテキストあるいは観測データが持っている確率分布を推定できてしまいます。これは一番簡単なトピックモデルで、一個一個の文書に潜在トピック、つまりこのテキストはこういう話題ですよと決め打ちする、言い換えると真の値があってそれを推定するという、一番簡単なトピックモデルです。先ほどのメールの例でいえば、普通メール、迷惑メールがトピックです。一般的には「政治」とか「経済」とか「コンピューター」とかトピックがいろいろあるわけです。パラメータは 2 種類あり、トピックの事前分布 p(k) と、そこからどんな単語が出てくるか p(w|k)、です。

Unigram Mixtures (例)

- 毎日新聞2001年度のテキスト(一部)から計算したUMのトピック別単語分布p(w|k)の上位特徴語

Topic 2	Topic 3	Topic 4
の,円,億,する,を,は, 生産,に,など,年度, 兆,万,約,#,削減,事業, 予算,や,化,計画,販売, いる,費,旅行,国内,工場, なる,減,グループ,から, 機,月,USJ,向け,会社, 同社,開業,年間,発表, 赤字,統合	社長,を,た,さ,月, 発泡,酒,容疑,年,者, れ,相,藤,氏,首相,会, は,化,秋山,検出, 市原,石川,辞任,社, 取締役,出身,就任, から,灯油,アサヒ	の,を,米,テロ,米国, する,パキスタン,同時, インド,アフガニスタン, タリバン,し,支援,へ, 多発,政府,アフガン, 国,いる,ドル,政権, 経済,組織,国際,金融, 資金,攻撃,IMF,など, 協議

毎日新聞の 2001 年の新聞記事のデータでは 1 千万個くらい単語があるのですが、その 1/10 くらいを使い、トピックは 100 個と限定して、Unigram Mixtures を適用した場合の結果の例を上に挙げています。分類された各トピックの単語分布の上位特徴語です。特徴語とは、大ざっぱに言うと確率の高いものですね。そうすると、例えばトピック 2 は、大体会社経営みたいな内容で、トピック 3 は人事に関する内容が目立ちます。トピック 4 では国際的な内容でしょうか。これは何も経営とか国際に関する言葉を集めるようにプログラムを組んだのではなく、Unigram Mixtures というモデルが勝手に学習した結果だという点を留意しておいてください。

Unigram Mixtures (例)

- 毎日新聞2001年度のテキスト(一部)から計算したUMのトピック別単語分布p(w|k)の上位特徴語

Topic 5	Topic 10	Topic 100
の,を,に,する,細胞,	た,し,に,と,が,て,	た,さん,て,で,容疑,
など,船,レーザー,	者,い,処分,こと,	い,調べ,ごろ,と,捜査,
ブロック,し,こと,	は,生徒,れ,人,さ,	署,市,れ,時,者,事件,
靴,から,や,な,型,	を,教委,問題,府,	が,いる,し,逮捕,午後,
銀河,融合,核,足,	女子,男性,被害,	男,み,県,県警,分,
研究,状,宇宙,評価,	保護,県,生活,保険,	男性,本部,殺人,いう,
が,方法,サイズ,不審,	など,ない,浪人,	から,午前,町,車,
物質,高速,なる,ず,	よる,あっ,教職員	同署,人,員,死亡,疑い,
意見,建造,グループ,星		乗用車,女性,府警

もう少し見てみると、トピック 5 では科学のこと、トピック 10 では教育問題、トピック 100 は警察の話をしていることがわかります。

なお、ここではトピック数はあらかじめ決めてあり、いくつのトピックに分けるべきかという個数を推定しているわけではありません。分けるべき個数自体を推定することもできますが、それは難しい話題なので、ここでは触れないことにします。今の場合は、100 個と決めてやりました。

最尤推定とベイズ推定

$$D(3,:) = [0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 2 \quad 0]$$

広告メール中の単語生起回数

- 頻度を単純に割り算した

$$\hat{p}(w) = \frac{n(w)}{N}$$

は、データの確率を最大にするので、最尤推定量とよばれる (n(w)=0なら確率は0)

- 実際には、単語の次元は数万次元‥ほとんどのn(w)は0
  - それらの確率は本当に0か?

以上で基本的なトピックモデルはできており、これを生態データや RNA のシーケンスデータに使うこともできるはずです。ただ、そういうときしばしば問題になるのが、例えば 100 や 200 といったカウントはいいのですが、カウントデータには 1 や 0 などの頻度の低いものも多く、それらがそのままでは信用できない場合が多いという点です。カウント数 0 の単語に先ほどの手法を使うと当然出る確率は 0 になるのですが、そのデータではたまたま 0 だっただけで、本当は確率は 0 ではないのかもしれません。つまり、データが 0 回だから確率も 0 としたのでは、推定した確率分布自体が信用できなくなるのです。例えば、先ほど迷惑メールでは、出現回数は、単語 1 が 1 回、単語 2 が 0 回、単語 3 が 1 回、数列としては 1、0、2、0, …となっていますが、これらを足して割った数値が本当に確率なのでしょうか。カウント数を単純に割り算した確率（p̂(w)=n(w)/N）はデータの尤度を最大にするので最尤推定量と呼ばれます。しかし、データのカウント数はたまたま 0 だっただけで、確率が本当に 0 かというと、おそらくそういうことではない。カウントデータから割り算で算出する最尤推定量は、特にデータが少ない場合、あまり信用できません。言語データでは単語という空間の次元が数万次元あり、ほとんどのカウントは 0 で、一個の文書に出てくる単語の数はたかだか数百か多くて 1000 くらいです。そんなデータだけからこの数万次元を推定できるわけがなく、もっとまじめに確率を考える必要があります。

最尤推定とベイズ推定 (2)

- NB/UMでは、
  $p_0 = \{p(w|k=0)\}, p_1 = \{p(w|k=1)\}$
  などの複数の多項分布を考える
  - ある分布で頻度が0でも、他では現れている ことが多い (共通性がある)

- $p_0, p_1, \cdots$ の 多項分布を生成する分布 を考えるべきなのでは?
  - 最も簡単な分布: ディリクレ(Dirichlet)分布.

そのためにベイズ推定をします。先ほどもナイーブベイズといって、ベイズの定理を使った推定をしましたが、それとは別に、今一度、もう少し深くベイズの定理を用います。ナイーブベイズでは、k=0 のときにはどんな単語が出て、k=1 のときにはどんな単語が出るか、という風に複数個の多項分布を考えました。すると、あるカテゴリーのデータの中でカウントが 0 のとき、そのまま割り算で確率を求めてしまうとその多項分布では確率 0 になってしまいます。しかし、他のカテゴリーでは現れている場合が多い。例えば、3 番目の迷惑メールでは、単語 2、単語 5、単語 7 の出現回数はいずれも 0 回です。しかし、迷惑メールにこれらの単語が絶対に出ないかというと、例えば単語 2 は普通メールでは文書 1、文書 2 ともに 2 回出現しているのですから、迷惑メールでもそれなりの確率で出現するだろうと思えます。逆に、単語 4 は迷惑メールには 1 回出ていますが、普通メールには出てきていません。しかし普通メールに全く出ないと考えるより、低い確率で出てきてもよいとするほうが自然に思えます。

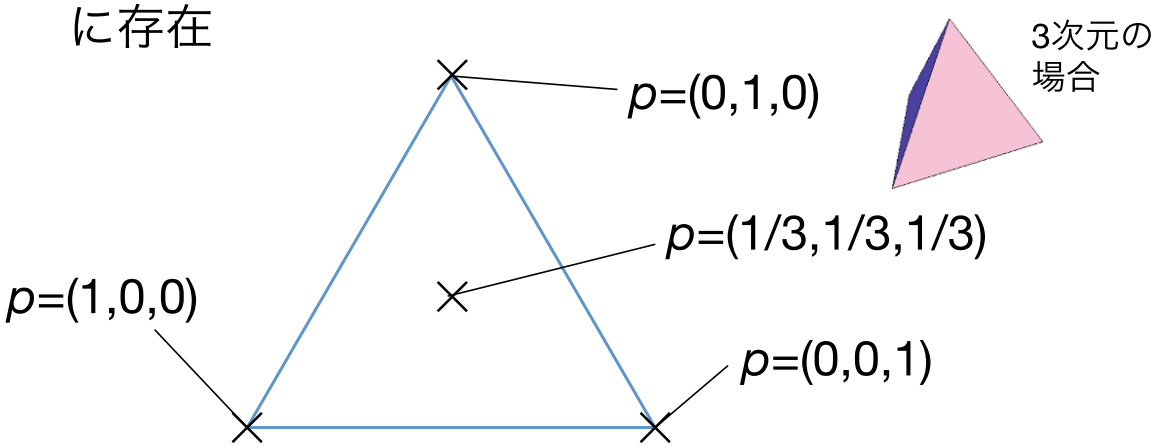
そこで、確率分布を独立に割り算して決めてしまうのでなく、ベイズ的な考え方をしましょう。つまり、今、単語の出現頻度という多項分布を考えているのですが、その多項分布を生成する分布 [事前分布] を考えて、それによって、多項分布つまりそれぞれの単語の出現確率を直接推定するのではなく、多項分布自体の確率分布 [事後分布] を推定しようとするのです。そのために事前分布として用意する一番簡単な分布が、ディリクレ（Dirichlet）分布と呼ばれるものです。

多項分布と単体

- K次元の多項分布

$$\mathbf{p}=(p_1,p_2,\cdots,p_K)\ (p_k\geq 0,\ \sum_k p_k=1)$$

は、単体(Simplex)とよばれるK-1次元の図形の中に存在



例えば、カテゴリーが三つの多項分布の場合、まず三角形の内部の点のそれぞれが、この多項分布の 3 個の確率を表していると考えます。足して 1 になるような 0 以上の数値の 3 組の集合は、三角形の中に含まれます。それを平面に描いたのが上の図です。つまり、三面サイコロを考え、1 しか出ないサイコロが三角形の左下の頂点 p=(1,0,0)に相当し、1,2,3 が均等に出るサイコロが三角形の中心 p=(1/3, 1/3, 1/3)に相当すると考えているわけですね。右下の頂点は 3 だけが出るサイコロ、上の頂点は 2 しか出ないサイコロに対応します。この 3 角形の中を点が動くことで、全てのサイコロを表現することができます。

今あるデータがどのサイコロらしいか、の確率分布がディリクレ分布です。

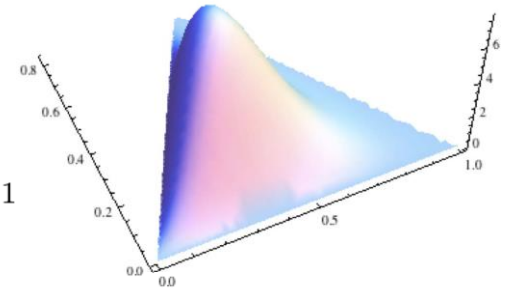
文書の単語を出すために、単語の種類と同じ数の面を持つ巨大なサイコロがあると考えると、「ほぼ 1 しか出ないけれど、少ない確率で 2 や 3 も出る」という多項分布や、「どの値も同じくらいの頻度で出る」多項分布の、どれがどのくらいもっともらしいかをディレクレ分布で表現できるわけです。

カテゴリーが四つの多項分布では、正四面体を考えることになります。

ディリクレ分布

- $\mathbf{p}=(p_1,p_2,\cdots,p_K)$ $(p_k \geq 0, \sum_k p_k=1)$ のとき、ディリクレ分布

$$p(\mathbf{p}|\boldsymbol{\alpha}) \propto \prod_{k=1}^{K} p_k^{\alpha_k-1}$$
$$= \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^{K} p_k^{\alpha_k-1}$$



  - $\boldsymbol{\alpha}=(\alpha_1,\alpha_2,\cdots,\alpha_K)$：パラメータ $(\alpha_k > 0)$
  - 期待値：$E[p_k|\boldsymbol{\alpha}] = \dfrac{\alpha_k}{\sum_k \alpha_k}$

多項分布 p に対して、以下の式で確率密度関数が定義される確率分布をディリクレ分布といいます。

$$p(p|a) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^{K} p_k^{\alpha_k-1}$$

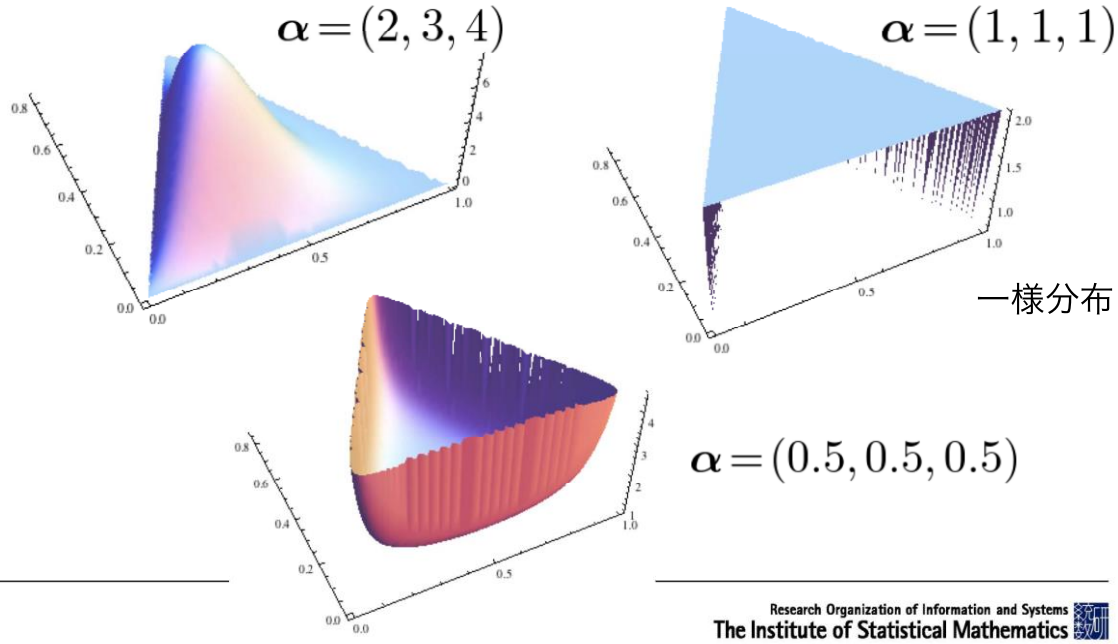
最初にガンマ関数を含む謎の定数が付いていますが、これは定数なので、確率密度関数の形は後の部分に比例しています。

$$p(p|a) \propto \prod_{k=1}^{K} p_k^{\alpha_k-1}$$

今、確率変数は $p=(p_1,p_2,\ldots,p_K)$で、それらは足すと 1 になる数の K 個の組で、それらは p_k の α_k-1 乗の巾乗という形になっています。ディリクレ分布は、離散分布におけるガウシアンのようなものと思っていいかもしれません。α はハイパーパラメータと呼ばれます。期待値は、この α を和が 1 になるよう正規化した形$\frac{\alpha_k}{\sum_k \alpha_k}$になり、α の総和が、広がりを表す分散のようなものになっています。

ディリクレ分布 (2)

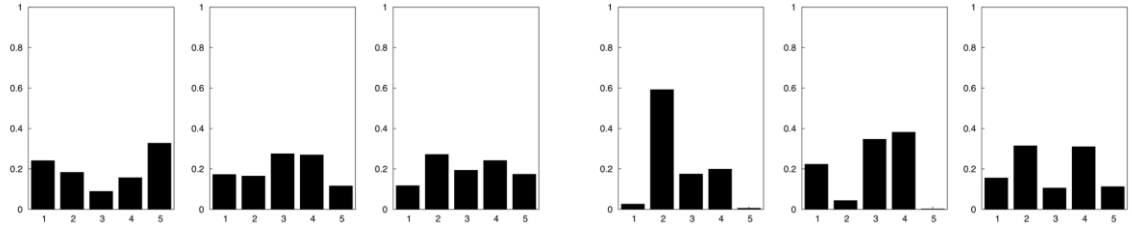
- ディリクレ分布のパラメータ$\boldsymbol{\alpha}$と分布の形



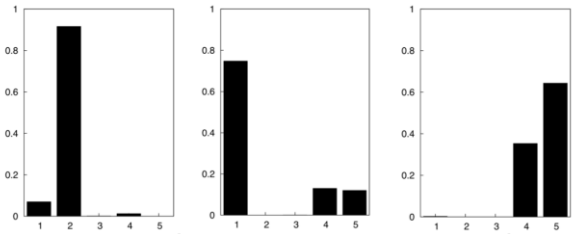
K 個のパラメータ α を変えると、いろいろな分布が出てきます。例えば、α=(1,1,1)とおくと、α_k-1=0ですが、p_k を 0 乗したら必ず 1 となるため、Uniform な分布が作られます。α が 1 より大きい値をとると、どこかに集中します。逆に、1 より小さい値をとると、くぼみが出て端で高くなるような分布になります。

ディリクレ分布 (3)

- ディリクレ分布からのサンプル $\mathbf{p}$



$\boldsymbol{\alpha} = (10, 10, \cdots, 10)$ $\quad$ $\boldsymbol{\alpha} = (1, 1, \cdots, 1)$



$\boldsymbol{\alpha} = (0.1, 0.1, \cdots, 0.1)$

実際にディリクレ分布から p_k たちをランダムにとってみると、例えばすべての α_k が 1 に等しい Uniform な分布からとると、得られる多項分布は右上のようにいろいろな形になります。一方、10,10,10, …のような中央にピークを持つ分布からとると、得られる多項分布は大ざっぱには一緒で、一様な多項分布になります（左上)。それに対して、こういう 0.1,0.1, …のときは、どこかの値が大きく、他はほとんど 0 のようなものが出てくるはずで、実際やってみると、どこかが大きくて他が 0 に近いという多項分布が出てきます。

最尤推定とベイズ推定 (3)

- $\mathbf{p}$ がディリクレ事前分布から生まれたとき、観測頻度 $X = n_1, n_2, \cdots, n_K$ による事後分布?
- ベイズの定理によれば、

$$p(\mathbf{p}|X) \propto p(X|\mathbf{p})\,p(\mathbf{p})$$
$$\propto \prod_k p_k^{n_k} \cdot \left(\prod_k p_k^{\alpha_k - 1}\right) = \prod_k p_k^{n_k + \alpha_k - 1}$$

  - これは $\mathrm{Dir}(\boldsymbol{\alpha}+\mathbf{n})$ なので、期待値は

$$E[p_k|X] = \frac{n_k + \alpha_k}{\sum_k (n_k + \alpha_k)}$$

ディレクレ分布をどう使うかというと、X が何かの観測頻度とします。例えば、ある植物 1 が観測されたのが n_1 回、植物 2 が観測されたのが n_2 回, …, 一般に植物 k が n_k 回とします (X = (n_1, n_2, …, n_k, …, n_K)。この裏には、これらの真の確率を表すベクトル p が隠れているはずです。この p の、X の観察データを踏まえた事後確率分布 p(p|X)は、これを逆転して、まず事前分布 p(p)があったと考えます。この事前分布のもとでデータが生成され、それらから事後分布 p(p|X)を求めます。これは簡単で、一個一個の確率が p_k で、それが n_k 回出てきているので、p_k の n_k 乗がその確率になります。それを事前分布のディレクレ分布に掛け算します。掛け算をすると k 番目は p_k の(n_k+α_k−1)乗という形になるので、結局全体では、(n_k＋α_k)という新しいパラメータを持つディリクレ分布になっているだけです。その期待値は、α_k の和が 1 になるように正規化したもので、以下のようになります。

$$E[p_k|X] = \frac{n_k + \alpha_k}{\sum_k (n_k + \alpha_k)}$$

最尤推定とベイズ推定 (4)

ディリクレスムージングという

- 最尤推定　$\hat{p}_k|X = \frac{n_k}{N}$
- ベイズ推定　$E[p_k|X] = \frac{n_k+\alpha_k}{\sum_k (n_k+\alpha_k)} = \frac{n_k+\alpha_k}{N+\sum_k \alpha_k}$
  - 頻度にα_kを足して正規化することは、ディリクレ事前分布$\mathrm{Dir}(\boldsymbol{\alpha})$を考えていることに相当する
  - $\alpha_k \equiv 1$：事前分布に一様分布を仮定
    - ラプラススムージングとよばれる (が、これが最良なわけではない)

こうすると、たまたま n_k が 0 だったとしても、$E[p_k|x]$には α_k が残るので、確率が 0 になることはなく、すべてについて穏やかな確率の値が得られます。言い換えると、偶然 k 番目の項目の出現回数が 0 だった場合（n_k=0）でも、期待値（$n_k+\alpha_k$）／ $\sum_k(n_k+\alpha_k)$は 0 になりません。このため、全体的に穏やかな確率が得られます。こうした方法をスムージングといいます。

まとめると、最尤推定では、観察されたカウント数をすべてそのまま正規化して和を 1 にするのですが、ここでは、k 番目のカウント数 n_k に α_k というハイパーパラメーターを与え、それらを $n_k+\alpha_k$ のように足し、その和が 1 になるように正規化しています。実はそれがベイズ推定になっていて、これをディリクレスムージングと言います。そのほうが、より頑健な推定値になります。

ナイーブなやり方はカウントに全部 1 を足すというやり方で、1 を足すというのは α_k=1 に対応し、さきほど示したように、これは事前分布が一様分布であることを意味します。言語処理の世界でも、最初の頃（どれくらい昔かというと、ラプラスが言ったので、18 世紀の終わりくらいです）は 1 としてたりしたのですが、もちろん 1 とすることは良くなくて、正しい値があります。それを推定したい場合は、この α_k 自体にガンマ分布を置いて、さらにベイズでサンプリングをしたりすると正しい値を求められます。カテゴリごとに何か小さな値を推定することで、推測性能が良くなることがわかっています。

UMの実装

- p(k)もp(w|k)もディリクレスムージング
  - EMが過学習しにくくなる

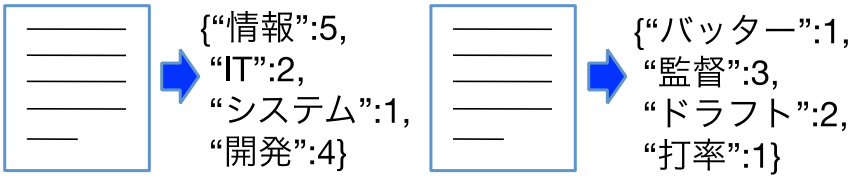
```
mondrian:~/work/um/src% ./um –h
um, Unigram Mixtures.
Copyright (C) 2012 Daichi Mochihashi, all rights reserved.
$Id: um.c,v 1.4 2013/01/05 06:33:55 daichi Exp $
usage : um -M mixtures [-e eta] [-g gamma] [-d epsilon] [-I emmax]
train model
eta       = Dirichlet prior for beta (default 0.01)
gamma = Dirichlet prior for lambda (default 0)
epsilon  = relative difference for convergence (default 0.0001)
```

  - http://www.ism.ac.jp/~daichi/dist/um/um-0.1.tar.gz

質問「今、ディリクレ分布を使ったのは、多項分布のパラメータ p_k の分布を求めたいからですか？」
答「そうです。もともとディリクレ分布の形から、それに多項分布の尤度をかけると同じ形で出るわけです。p_k のべき乗という形です。事前分布が Dir(α)で、事後分布が Dir(α＋n)と、同じ形が出てきます。なお、絶対にディリクレ分布でないといけないというわけではなく、もっと複雑な分布を使ってもかまいません。」
質問「今のディリクレ分布では$\hat{p}$を表していたと思うのですが、ちょっと前の話でカテゴリを迷惑メールとそうでないメールのように、いくつか分けていましたね。あれをディリクレ分布で表しているのですか？それとも同じカテゴリの中での p のばらつきをディリクレ分布で表しているのでしょうか？」
答「両方できます。先ほどのナイーブベイズでは、分布が p(k)と p(w|k)の 2 種類あり、どちらも多項分布でした。どちらでやってもかまいません。片方は最尤推定でもかまいません」
質問「モデルを選ぶ p(k)のほうにディリクレ分布を使ってもいいのですか？」
答「はい、そして、そこから単語を出すほうに使ってもいいわけです」
これはいい質問で、それをやらないと Unigram Mixture を動かしてもなかなかうまくいかないという場合は結構あります。
ディリクレスムージングを入れたソフトを私が書いて、プログラムとして公開しています。これをダウンロードすると、普通に使えるようになっています。

UMとBag of wordsの復習

- Bag of words: テキストを単語の集合と頻度で表現



- Unigram Mixturesの生成モデル
  - For d = 1 … D,
    (1) $z \sim p(z)$ でテキストdのトピック z を選択
    (2) For n = 1 .. N_d,
    $w_n \sim p(w|z)$ でn番目の単語 w_n を生成.

これは復習です。

Bag of Words

・ テキスト＝単語の集合とし、その前後関係は考慮に入れず、単語の出現頻度がテキストの特徴を表現しているという仮定。

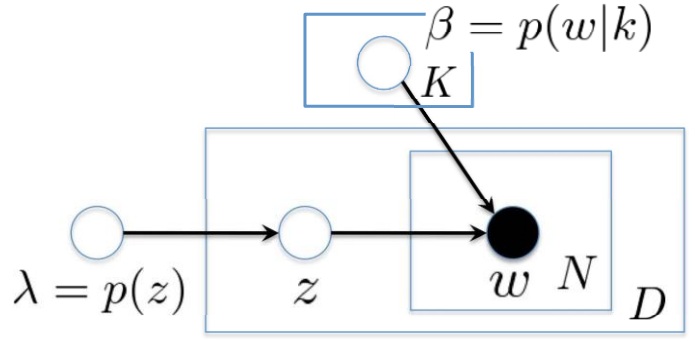
Unigram Mixtures の生成モデル

テキスト d = 1, 2, …, D に対し、

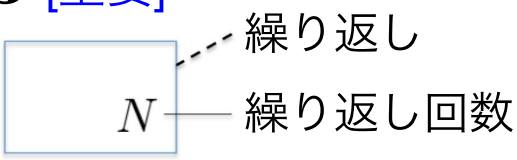
・ 確率 p(z)でトピック z を採用

・ トピック z を特徴づける単語の確率分布 p(w|z)に従って、n 個の単語 $w_1, w_2, \ldots, w_n$ を抽出

グラフィカルモデル表現



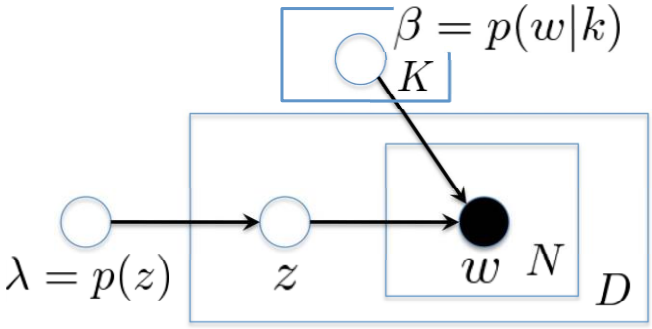
- UMのモデルは、上のようなグラフィカルモデル(プレート表現)で表せる [重要]
  - ●：観測された変数
  - ○：未知の潜在変数

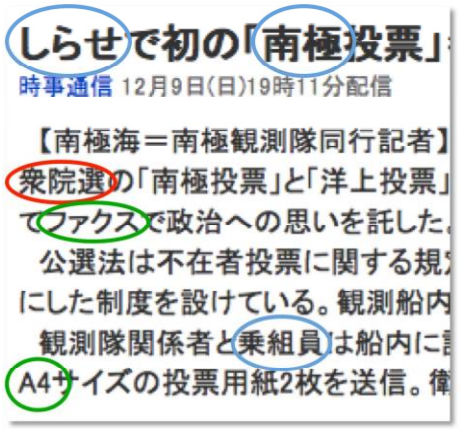


さて、Unigram mixture の次のステップに進む前に、グラフィカルモデルについて説明しておきます。上の図は Unigram Mixture のグラフィカルモデル表現で、黒丸は観測値を表しています。今観測されているのは単語ベクトル $w=w_1, \ldots, w_N$ だけで、その裏に、一個一個の文書ごとにカテゴリ z があり（先ほどまでは k と書きましたが、いわゆる潜在変数なので z にするほうが多いです)、そのカテゴリから、単語が全部で N 回抽出されています。図の箱は繰返しを表していて、単語 w_1、単語 w_2、単語 w_3・・とランダムにサンプリングされているのが全部で N 回あるという意味です。そもそも文書が D 個ありますので、同じことが全体で D 回行われています。どういうトピック z を選ぶかというパラメーター($\lambda=p(z)$)が一個あり、そこから z を選ぶ操作を D 回やり、各 z の中で単語を N 回出す。こういうモデルになっています。一方、全ての文書に共通な、カテゴリ k からどんな単語が出てくるのですかという分布 p(w|k)があり、この z が決まったら、この分布を使って(k=z として)単語 w を出力します。

箱は繰返しを表していて、箱の右下に繰り返し回数が書いてあります。こういうのがグラフィカルモデルです。なお、ユニグラムとかナイーブベイズとかいう言葉は最近よく聞かれるのですが、本屋でよく売っているベイズ統計の本は大体このあたりまでで終わっていて、それ以上の話題に触れておりません。しかし、この段階では、まだまだ問題があります。

UM/NBの問題



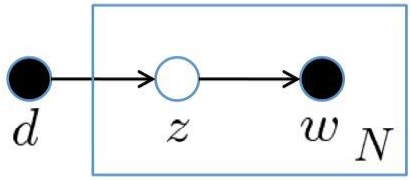

しらせで初の「南極投票」
時事通信 12月9日(日)19時11分配信
【南極海＝南極観測隊同行記者】
衆院選の「南極投票」と「洋上投票」
でファクスで政治への思いを託した。
公選法は不在者投票に関する規
にした制度を設けている。観測船内
観測隊関係者と乗組員は船内に
A4サイズの投票用紙2枚を送信。

- 1つの文書は、すべて同じトピックに属する
  - 実際の文書は、複数のトピックが入り混じっている!
- UM/NBは制限の強い、単純すぎるモデル

何が問題かというと、例えばこれは Yahoo ニュースから取ってきたのですが、「しらせ」とか「南極」などは多分、南極観測のようなトピックから出てきていますが、この記事はしらせで投票をしたという話なので、「衆院選」とか「ファックス」とか「A4」とかも出てきています。ところで、「A4」と「衆院選」と「しらせ」が一個の確率分布から出てきたとは到底思えない。そうではなくて、この記事に関する話題は、南極船の話題が 0.6 くらいで、選挙の話題が 0.3 くらいで、他の話題が 0.1 くらい入っているという感じに思えるわけです。一個一個の文書が、複数のトピックにおける単語の確率分布の混合分布になっていると考えられるわけです。

PLSI (Hofmann 1999): 「トピックモデル」

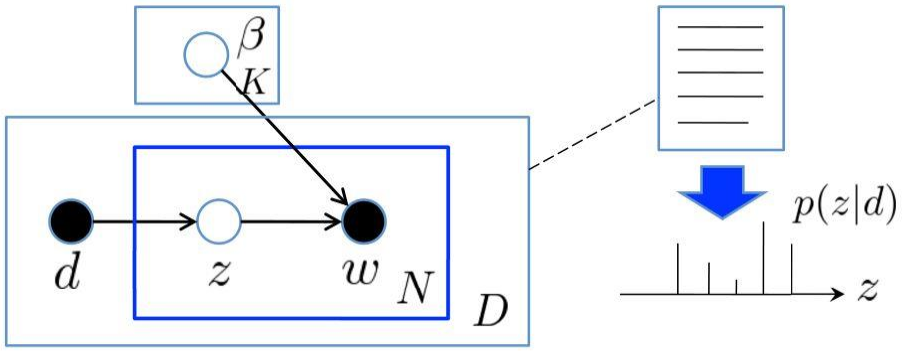
- Probabilistic Latent Semantic Indexing:
  複数のトピック、LSIの確率化
  - PLSA (Probabilistic Latent Semantic Analysis)
    ともよばれるが、以下PLSI



- 1単語ごとに、違うトピックz
  - 文書には1つのzではなく, “トピック分布” $p(z|d)$

それが、いわゆるトピックモデルというもので、最初に提案されたときには、上のような簡単な形になっていました。これは一見簡単に見えますが、文書ごとに、それがどういう話題 z を持つかが確率分布になっています。つまり、文書 d からどういうトピック z が出てくるかというトピックの分布が文書ごとに一個一個決まっていて、その確率分布からまず話題 z を選び、次にそこから単語を出します。この操作を全体で N 回繰り返します。つまり、一単語ごとに違うトピックがあるのです。先ほどは、一個の文書があったらその話題は全部同じだったのですが、このモデルからは一単語ごとに違う話題があるわけです。そうすると、先ほどの例では、「しらせ」を出すときには、まず「南極トピック」を選び、次にそこから「しらせ」という単語が選ばれたと考える。次に「衆院選」を出した時には、まず「選挙トピック」が選ばれ、そこから「衆院選」という単語が出てきた。このように考えられるわけです。

PLSIのグラフィカルモデル



- 文書dには、トピック分布 $p(z|d)$ が存在
  - 一単語ごとに、$p(z|d)$ からトピックzを生成
    → z から単語を生成する

グラフィカルモデル表現も違ってきます。さっきと似ていますが、d から z の部分が、図の右に示したような確率分布 p(z|d)になっています。これは未知のパラメータなので、推定することになります。それからもちろん、p(d)や p(w|k)も推定します。

PLSIの学習結果 (1)

- トピック別単語分布 $p(w|z)$ の上位語

Topic 1	Topic 2	Topic 3	Topic 4
先,後,#,歩,銀,四, 五,六,同,二,飛, 八,成,玉,七,三, 金,九,桂,角,と, 谷川,が,た,手,は, 丸山,一,香,の,で, 局,図,戦,段	の,号,事故,機,が, た,に,安全,#,部分, を,原発,原因,は, 基,水,運転,装置, 爆発,器,原子力, 炉,作業,し,燃料, で,漏れ,発生,と, 配管,原子,ガス	#,勝,敗,戦, イチロー,日,回, リーグ,大リーグ, マリナーズ,新庄, 試合,安打,点,ス, で,手,共同,メッツ, 外野,は,大,投手, 第,米,の,打席, ソックス, ヤンキース,記録, バックス,打率, ニューヨーク	研究,細胞, 遺伝子,移植, の,治療,物質, 教授,を,患者, 科学,脳,医療, 病院,ローン, ヒト,実験,薬, グループ,遺伝, が,臓器,体,ク, 病,する,に,学会, さ,DNA,開発, 臨床,人間,神経

こうしたアイデアに基づく最初のトピックモデルは PLSI と呼ばれるもので、これについてはすぐ後で説明しますが、先にその効果を見ておきます。これは、前に 1 個の文書はすべて同じ分布から出てきたと考える Unigram Mixture のときの例と同じデータに PLSI を適用した結果の一部なのですが、以前は、頻度の高い上位語の中に、「を」とか「の」とか「する」とかいう機能語の類がたくさん入っていました。それが PLSI を使うとほとんど除かれるのです。実は Unigram mixture の例でも結構頑張って機能語を除いたのですが、手作業で除くのには限界があり、ちゃんと統計的にやらないとうまく動きませんでした。それが、PLSI を用いると、Topic 3 は見事に野球の話ばかり集まっていますし、Topic 1 は将棋ですね。べつにテキストに「トピックは将棋」と書いてあるわけではないのですが、モデルが勝手にこうした単語を見つけてまとめてくれるわけです。Topic 2 も、原発とか事故とか配管とか、原発事故に関するものですね。Topic 4 は、「研究」とか「細胞」とか「遺伝子」とか、医学や生物学に関する単語です。

PLSIの学習結果 (2)

- トピック別単語分布 $p(w|z)$ の上位語

Topic 5	Topic 10	Topic 50	Topic 100
イスラエル,	#,回,た,を,勝,	た,#,を,容疑	の,を,#,に,
パレスチナ,	の,で,監督,は,	し,の,者,に,	は,環境,化,
自治,議長,	が,敗,点,に,	で,は,て,逮捕,	が,で,する,
和平,中東,派,	投手,登板,試合,	と,月,い,同,	し,など,量,
攻撃,の,	巨人,一,近鉄,	など,捜査,れ,	と,書,や,
エルサレム,	本塁打,安打,	疑い,元,さ,	削減,開発,
と,延期,人,	打,死,球,戦,	県警,が,違反,	た,年,て,いる,
政府,を,過激,	ヤクルト,	事件,日,万,	地球,も,生産,
日,アラファト,	先発,ダイエー,	ら,処分,人,	国,議定,温暖,
ユダヤ,イラク,	阪神,西武,初,	として,課,	ガス,エネルギー,
軍,テロ,小倉,	チーム,今季,番	地検,調べ	効果,技術,さ,
は,シャロン			国内,計画

他のトピックも、何となくどういう話題なのか、想像がつくことと思います。

PLSIの生成モデル

- PLSIでは、次のようにして文書群が生成されたと仮定する
  - For i = 1 … D,
    (1) 文書 $d \sim p(d)$ を選択.
    (2) For n = 1 … N,
    (a) トピック $z \sim p(z|d)$ を選択.
    (b) 単語 $w \sim p(w|z)$ を生成.
- 確率で書くと、

$$p(d,w) = \sum_z p(d,w,z)$$
$$= \sum_z p(d)p(w,z|d) = p(d)\sum_z p(w|z)p(z|d)\,.$$

40

さて、生成モデルとして PLSI はどうなっているかというと、全部で D 個の文書があり、文書一個一個について、まずインデックスを作り（d~p(d)）、このインデックス、すなわち文書ごとに単語が N 個あるのですが、まず文書が持っている話題分布からトピック z を作り（z~p(z|d)）、その話題から単語を出す（w~p(w|z)）ということをぐるぐる繰り返します。

つまり確率で書くと、ある文書とある単語が共起したのは、その裏に z という見えない話題があり、それに対して周辺化されているわけですが、それは文書をまず選び、その文書から話題を選び、話題から単語を選ぶ、ということを z に対して和をとったという確率モデルになります。

PLSIの生成モデル (2)

- PLSIによる文書とコーパス全体の確率
  - PLSIでは、文書のインデックスdを観測値として考える

$$p(d, \mathbf{w}) = \prod_n p(d, w_n) = \prod_n p(d) \sum_z p(w_n|z_n) p(z_n|d)$$

$$p(D, W) = \prod_i p(d_i, \mathbf{w}_i)$$

今までは一つの文書と一つの単語が共起する確率を考えていましたが、ここでは、一つの文書と複数の単語が共起する確率について考えています。それは単に、各文書ごとにすべての単語との共起確率をかけあわせ、さらにそれらをすべての文書についてかけ合せるだけです。

PLSIの生成モデル (3)

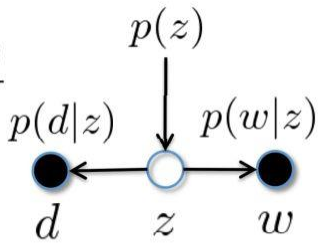
- ベイズの定理を使って計算すると、単語の確率は

$$p(d,w) = p(d)\sum_z p(w|z)p(z|d)$$

意味が謎

$$= p(d)\sum_z p(w|z)\frac{p(d|z)p(z)}{p(d)}$$

$$= \sum_z p(z)p(d|z)p(w|z)\,.$$



  - 文書dと単語wの共起にトピックzが存在
- 新しいパラメータ: $p(z), p(w|z), p(d|z)$

こうして p(z|d)や p(w|z)が与えられたら確率的に文書を生成するモデルはできたのですが、問題はこの逆で、文書が沢山与えられたとき、それからどうやって p(z|d)や p(w|z)を推定するかにあります。いわゆる逆問題ですね。ここで、いささかテクニカルになりますが、またベイズの定理を使って式を書き直します。特に難解な計算をするわけではありませんが、それで Unigram Mixture と同様に、EM アルゴリズムを使うことで推定が可能となってしまうのです。

さきほどから、文書を選ぶ確率 p(d)というのが出ていますが、これの意味になんとなく謎めいた印象を受けている人はいませんか？p(d)がちょっと怪しいと気付いた人は鋭い人だと思います。そこで、条件付き確率とベイズの定理を使って上のように式を変形します。最後の式はどう解釈されるかというと、ある文書 d とその中の単語 w が共起する確率は、まず話題 z が先に選ばれます。まず z を選び、その z から文書と単語が出てきたと思いましょう。最初の p(z)は、どんな話題が出やすいかという確率分布を表していて、その話題からどんな文書が出やすいのか（p(d|z)）、あるいはその話題からどんな単語が出やすいのか(p(w|z))が決まってくる。ちょっと変形しただけなのですが、これでモデルが EM アルゴリズムで解ける形になってしまうのです。

PLSIの学習

- 文書-単語行列Wと文書インデックスD、各単語の潜在トピックZについて、

$$
\begin{aligned}
p(D, W, Z) &= \prod_d p(d, W_d, Z_d) \\
&= \prod_d \prod_n p(d, w_{dn}, z_{dn}) \\
&= \prod_d \prod_n p(z_{dn}) p(d|z_{dn}) p(w_{dn}|z_{dn})
\end{aligned}
$$

- よって、対数尤度は

$$
\log p(D, W, Z) = \sum_d \sum_n \left[\log p(z_{dn}) + \log p(d|z_{dn}) + \log p(w_{dn}|z_{dn}) \right]
$$

EM アルゴリズムの手順の詳細は説明しませんが、ある程度のイメージを持つために、このモデルの下でのデータの対数尤度の式は必要です。D は文書の集合、W は単語ベクトルの集合で、これらはデータとして与えられます。一方、Z は話題ですが、これは未知の潜在変数で、問題はこの推定にあるわけです。ですから、推定したいのは p(z), p(d|z), p(w|z)というパラメータと、それぞれの文書の中の各単語がどの話題から選ばれたものなのか、という潜在変数 Z です。

PLSIの学習 (2) : EMアルゴリズム

- Eステップ:

$$p(Z|W,\theta) = p(z|d,w)$$
$$\propto p(z,d,w_n) = p(z)p(d|z)p(w|z) \text{ を計算.}$$

  - 文書dの単語wが、どの潜在トピックに属するかの確率分布

- Mステップ: Q関数 $Q(\theta) = \langle \log p(W,Z|\theta) \rangle_{p(Z|W,\theta)}$ をθについて最大化.

$$Q(\theta) = \sum_d \sum_n \sum_z p(z|d,w_{dn}) \Big[\log p(z_{dn}) + \log p(d|z_{dn}) + \log p(w_{dn}|z_{dn}) \Big]$$

EM アルゴリズムは、E-step と M-step に分けられます。

E-step では、推定したい未知の潜在変数について、その期待値を求めます。いまのモデルでは、文書 d の中のある単語 w がどのトピックかという確率分布 p(z|d,w)の形で求めます。これは上のようにして、3 つのパラメータが与えられれば計算できます。

M ステップでは、この期待値の下で対数尤度を最大にするパラメータを求めます。そこで、テクニカルな話ですが、Q 関数というものを作ります。

PLSIの学習 (3)

- $\delta Q/\delta\theta = 0$ を計算すると、

$$\frac{\delta Q}{\delta p(z)} = \frac{\sum_d \sum_n p(z|d, w_{dn})}{p(z)} + \lambda = 0$$

よって

$$p(z) \propto \sum_d \sum_n p(z|d, w_{dn}) \propto \sum_d \sum_n p(z, d, w_{dn})$$
$$\propto \sum_d \sum_n p(z|d, w_{dn}) p(d, w_{dn}) \propto \sum_d \sum_w p(z|d, w) n(d, w)$$

- 同様にして、

$$p(w|z) \propto \sum_d p(z|d, w) n(d, w), \quad p(d|z) \propto \sum_w p(z|d, w) n(d, w)$$

Q を最大化したいので、これを微分して 0 とおくことで、3 つのパラメータの計算式が得られます。ここで、 n(d, w) は、文書 d の中に単語 w が何回入っているか、そのカウント数、要するにデータを表しています。

PLSIの学習 (4)

- まとめ: PLSIのEMアルゴリズム
  - 初期化: $p(z), p(w|z), p(d|z)$ を乱数で設定.
  - Eステップ:
    各文書dの各単語wについて、トピック分布
    $$p(z|d,w) \propto p(z)p(d|z)p(w|z)$$
    を計算.
  - Mステップ:
    今計算した $p(z|d,w)$ を用いて、パラメータ $p(z), p(w|z), p(d|z)$ を更新. Eステップに戻る.
- 実装: http://chasen.org/~taku/software/plsi/plsi-0.03.tar.gz

以上の EM アルゴリズムをまとめてみます。

E-step では、各文書 d の各単語 w について、その単語 w はどういう話題から選ばれたらしいかを確率分布の形で推定します。

この「一個一個の単語がどういう話題に割り当てられたのか」がわかっていると（期待値という意味で)、それを逆に使うことで、ある話題からどういう単語が出やすいかがわかります。それが M ステップで、それを使って、３つのパラメーターを更新します。

そうしたら、E ステップに戻って、もう一回その新しいパラメーターを使って、単語が本当はどういう話題なのかを計算する。これをぐるぐる繰り返します。

この計算は R のようなソフトでもできると思いますが、言語処理では単語の種類の数が数万あり、文書の数も数万とか数十万とか数百万とかあるので、R とかでは一年たっても終わらないようです。それで普通、C 言語とかでプログラムをゴリゴリ書きます。うまく工夫すると、10 時間とか一晩くらいで終わるくらいの計算量です。4 日くらいかかる研究も普通にあります。

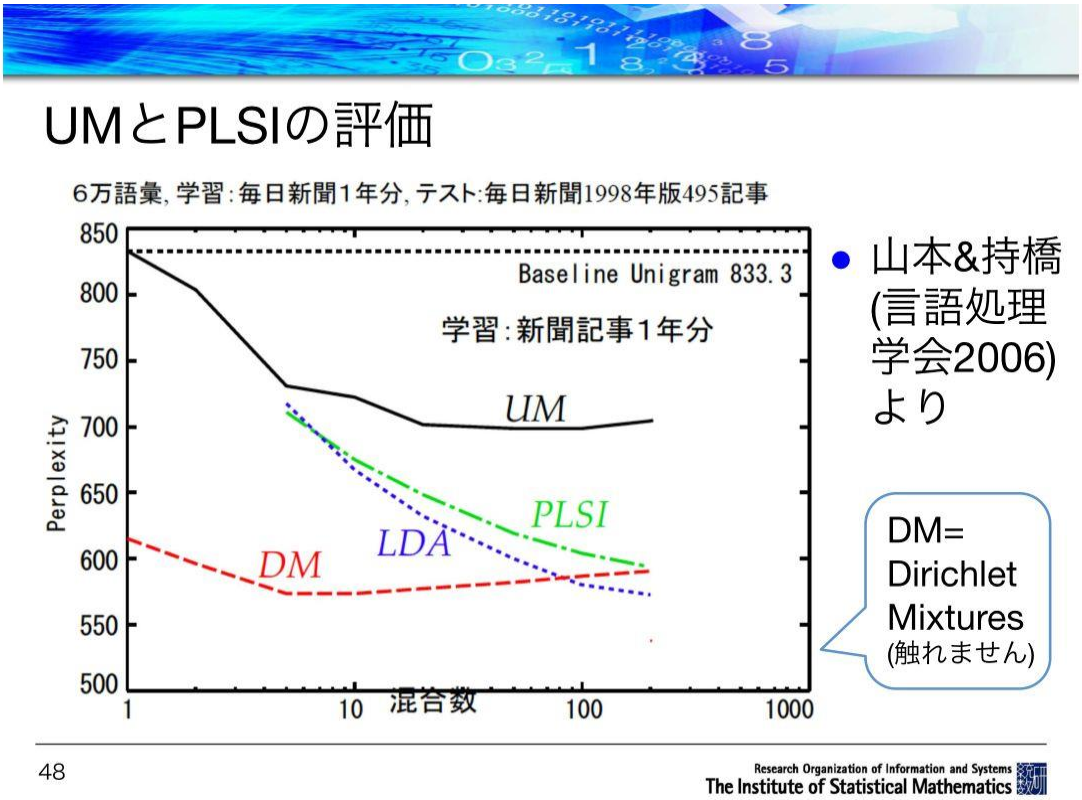
PLSI: retrospective

- 単語ごとに潜在トピックを考えることで、UMよりずっと良いトピック分布
- UMは、普通の混合モデル
  - 混合比からトピックzを選ぶ→zから文書を生成
- PLSIは、文書ごとの混合モデル
  - 文書ごとに混合比を選ぶ
  - 混合比からトピックzを選ぶ→zから単語を生成

PLSIは、混合モデルの混合モデル

こうして文書 d の各単語 w ごとに潜在的な話題 z を考えるので、いわゆるナイーブベイズなどより、はるかによいモデルが作れます。Unigram Mixture も混合モデルの一種で、つまり、まず混合比からある話題を選び、その話題からテキスト全体を作っているわけです。一方このトピックモデル PLSI では、一個一個の文章ごとにまず混合比 p(z|d)を選び、この混合比から各単語ごとにトピックを選んで、そこから単語を出力します。結局、混合モデルの混合モデルというのがトピックモデルの本質です。

ただし、p(z|d)は生成モデルには登場しますが、EM を適用する段階で、これは逆の p(d|z)になっています。p(z|d)はベイズの定理を使えば、p(z)p(d|z)に比例するので求められますね。この p(d|z)というのも少々謎の確率分布ですが、この問題はすぐ後で述べます。それがそのまま次のステップのモデル、LDA へつながっていきます。



LDA へ進む前に、PLSI の性能について、少し触れておきます。

トピックモデルは言語処理の人にはそれなりに知られてきていますが、他の分野では、多分まだ教科書がないこともあって[1]、あまり知られていないと思います。

Unigram Mixture (UM)と比較してみると PLSI のほうが性能がよいことを示しているのが上の図です。縦軸について、しっかりした説明は省きますが、モデルの性能の評価の一つで、下に行くほど性能が良いモデルを意味します。赤の DM=Dirichlet Mixtures は今はおいておいて、いわゆるナイーブベイズ（教師なしナイーブベイズ）でやると、Unigram Mixture は黒い実線ですが、話題の数を増やしていくと性能は良くはなるものの、図のように向上が止まってきます。一方、PLSI や、すぐ後で説明するそのベイズ版 LDA では、はるかによい性能が出ています。

ちなみに、DM=Dirichlet Mixtures というモデルは、話題は文書ごとに一個ですが、単語にキャッシュがあって、一回出た単語が二回以上出やすくなるという性質をうまくモデル化するものです。それを使うと、つまり Unigram Mixture にキャッシュを付けると、ものすごく性能が良くなるわけです。現状では、LDA にキャッシュを入れるともっと性能がよくなるみたいなことがわかりつつあるというのが、今の言語処理の状況です。

[1] この講演の時点(2015 年 1 月)では、まだ日本語の教科書は出版されていなかった。

PLSIの欠点

- PLSIのパラメータ: $p(z), p(w|z), p(d|z)$

学習データに比例して増大!
(D×K=数万～数千万×数百)

簡単に学習データにオーバーフィット.

- アドホックな解決: Tempered EM (焼きなまし)

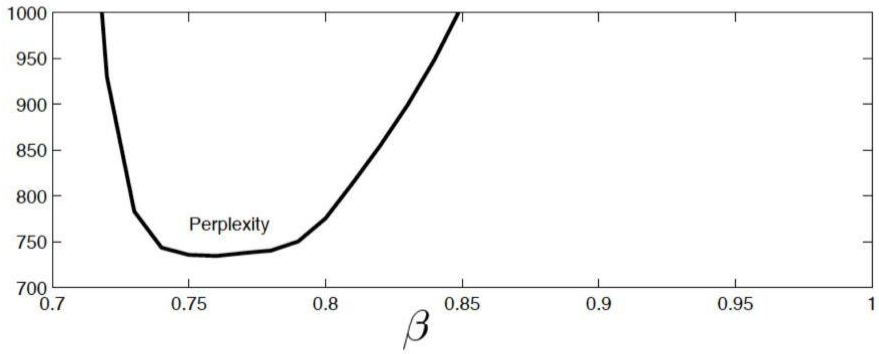
$$p(z|w,d) \propto \{ p(z)p(w|z)p(d|z) \}^{\beta}$$

  - $0<\beta\leq 1$ ととって、確率分布p(z|w,d)を「なめらか」にする

PLSI は 99 年頃に出て、統計を知っている人はすごいと思ったわけですが、もちろん問題もあります。何が問題かというと、話題の事前分布 p(z)とか、話題ごとの単語の分布 p(w|z)を推定するという部分はいいのですが、p(d|z)がパラメータに入っています。これは話題 z からどんな文書 d が選ばれるかの確率ですが、この文書 d というよくわからない謎の確率変数が入っています。そしてこれは、学習データに比例して増えていきます。これはモデルの中に含まれていて、未知パラメータとして推定しないといけません。未知パラメータを導入したことで EM で解けるようになったのですが、推定すべきパラメータ数が膨大になってしまいました。そのため、しばしば学習データにオーバーフィットして、性能がどんどん悪くなってしまうのです。

それを解決するためには、焼きなまし(Tempered)法というのがあって、確率分布の計算をしたときにオーバーフィットしないようにβ乗します。このβが 1 より小さい値、例えば 1/2 だと√をとる形になるので、確率分布がなめらかになるわけです。

Tempered EMの効果 (Hofmann99)



- $\beta=0.75$程度で最良 ($\beta=1$では大幅に悪化)
- アドホック。。。
  - そもそも、なぜオーバーフィットするか?

焼きなましでこのβをだんだん1に近づけるといった、ある意味姑息な方法を使うと、βが1だとだめなものが、βを0.75くらいにするとうまくいく、みたいな方法が最初のころ取られていました。でも、こんなアドホックでいいの？という疑問がありました。ここでいう「アドホック」とは、オーバーフィットする原因自体を解決することなく、対処療法的に学習し過ぎないようにしていることです。

PLSIからLDAへ

- PLSIの問題点: $p(z|d)$ が学習データについてだけ定義されている
(由来がない)

  ⇩

  真の生成モデルではない！
  - 学習データについての最尤推定
- $\theta = p(z|d)$ 自体を確率的に生成するモデルを考えるべきなのでは?

  ⇨ LDA (Latent Dirichlet Allocation)
  (Blei+ 2001,2003)

そもそも問題は、EM で解くということからわかるように、PLSI というトピックモデルはベイズではなく、真の生成モデルでもなく、最尤推定をやっているだけなのです。文書ごとの p(d|z)をパラメータと思って最尤推定していて、p(z|d)はパラメータの最尤推定値から求められるのですが、これを確率的に生成するようなモデルを考えましょう。それには、全体をベイズで書きましょう。これが、一番基本的なトピックモデルである Latent Dirichlet Allocation です。これが 2001 年から 2003 年ごろの話です。

LDAとは

- Blei+ (NIPS 2001, JMLR 2003)で提案
- トピックモデルの最も基本となるモデル
  - PLSIの完全なベイズ化
- 基本的な考え方は、PLSIと同じ
  - 単語ごとに潜在トピックがある

基本的な考え方は、PLSI と同じで、単語ごとに見えないトピックがあるというものです。

トピック分布の由来

- PLSIでは、文書dにトピック分布 $\theta = p(z|d)$ があった

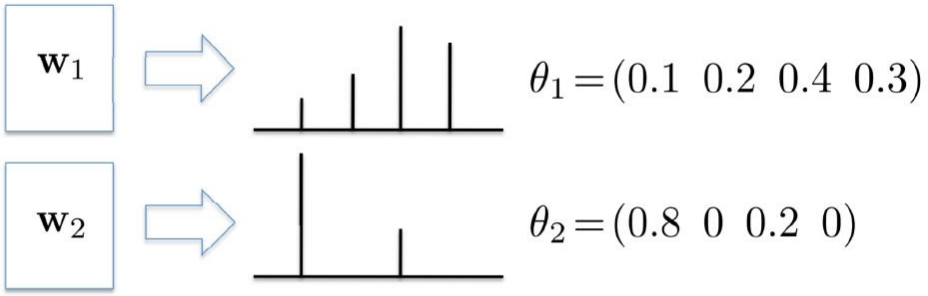
  これを確率変数(×パラメータ)とみて、生成したい
- ディリクレ分布を使えばいい!
  - $\theta = (\theta_1, \theta_2, \cdots, \theta_K)$ を生成する、K次元のディリクレ分布 $\mathrm{Dir}(\theta|\boldsymbol{\alpha})$

$$p(\theta|\boldsymbol{\alpha}) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$

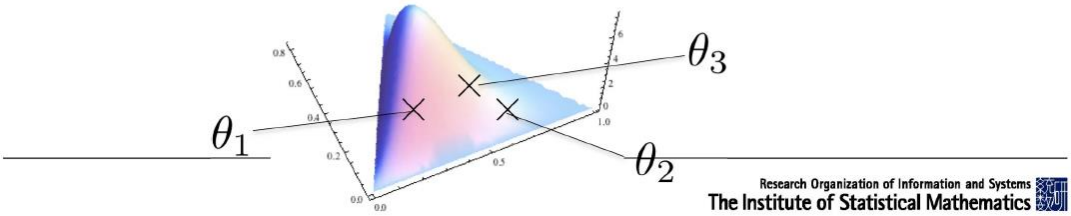
ではどうすればいいかというと、ある文書が持っている話題の分布というものは離散分布なので、ディリクレ分布を使ってこれを作る。基本的にはこれだけです。このことを簡単に θ と書いていますが、ディリクレ分布を使って作る、それだけです。

LDAの生成モデル

- 文書 $\mathbf{w}$ を話題(トピック)の混合で表現



- 混合比 θ をディリクレ事前分布から生成



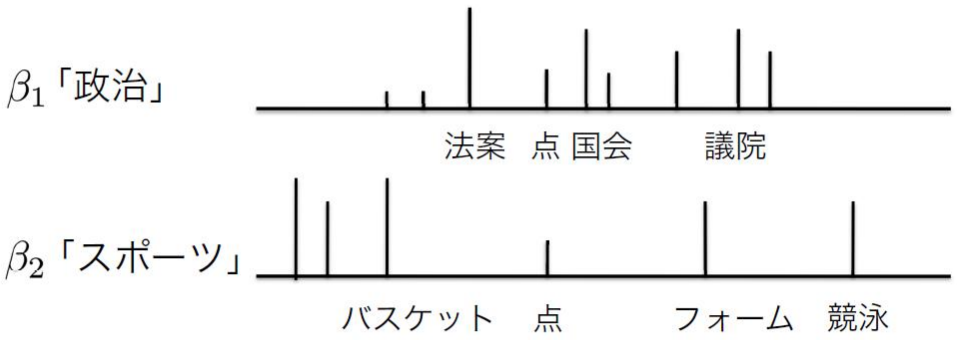
Research Organization of Information and Systems
The Institute of Statistical Mathematics

そうすると、あるテキスト $\mathbf{w}_1$ が持っている話題の分布が上のほうのような感じになっていて、それはこのディリクレ分布からランダムにとってきたのだけど、それは上の図で θ_1 で指されているあたりからだった。$\mathbf{w}_2$ だと下のほうのような感じの分布になっていて、それも同じディリクレ分布からランダムにとってきたのだけど、こっちは θ_2 で指されているあたりからだった。こういうモデルを考えます。

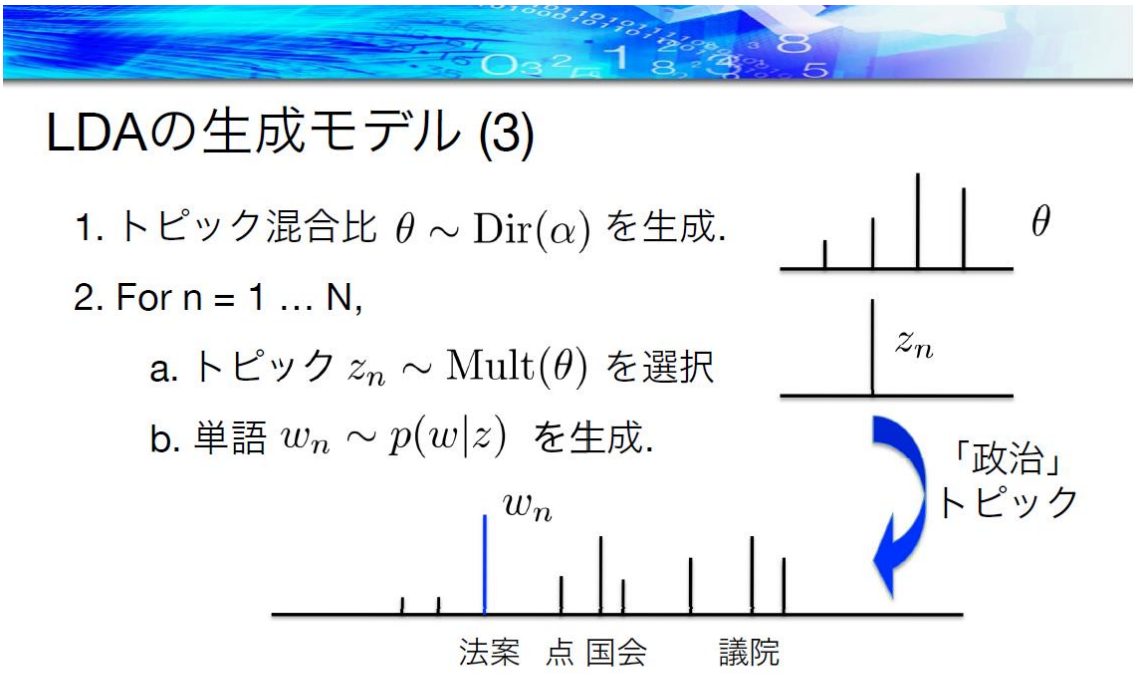
LDAの生成モデル (2)

- 「話題」とは?→単語の生起確率分布 $\beta_k = \{ p(w|k) \}$

$(w = 1 \cdots V)$



話題から単語を選ぶ方は、こっちは最尤推定をやっても良いのですが、これも多項分布なので、やはりディリクレ分布から作ることにしましょう。たいていの場合、話題より高次元のディリクレ分布になります。

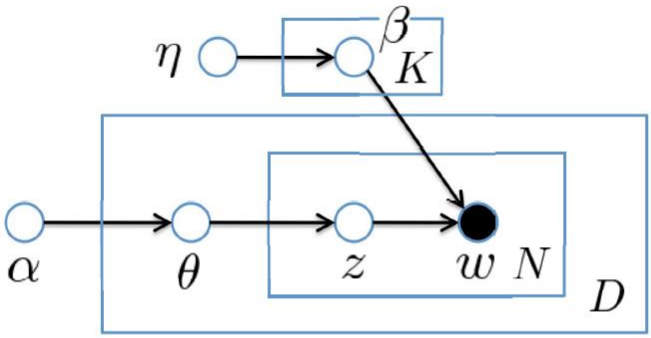


全体の生成モデルは、一個一個の文書に対して、どういう話題を持っているかという混合比 θ という確率分布、これをまずディリクレ分布からのランダムサンプリングで作ります。もし全部で話題が 4 個あるなら、各文書について、4 次元のディリクレ分布からのランダムサンプリングで話題の分布を決めます。

次にその多項分布の中から、話題 z をランダムに選びます。それから、その話題からはどういう単語が出やすいかという多項分布から、ランダムに単語を選びます。２番目の単語についても、同じようにまず混合比の多項分布から話題を決め、その話題の多項分布から単語を選びます。もし話題「政治」が選ばれたなら、「政治」トピックの確率分布から単語をランダムサンプリングします。おそらく「法案」とか「国会」とか「議院」などが出やすい確率分布なのでしょう。今とり出した単語が「法案」だったとします。次にまた同じことをして、今度は話題「経済」が選ばれ、「経済」トピックから単語を出したとします。この操作を、N 個ある単語ごとに行います。

次の文書に対しては、混合比をディリクレ分布からランダムにサンプリングする作業から始めます。これが生成モデルですが、もちろん、現実には文書、つまり単語のカウント数というデータがあるだけで、この生成過程はわかっていません。それをデータから求めるのです。

LDAのグラフィカルモデル



- $\theta \to z \to w$ の順で単語wを生成
- θ はD回(文書数)、z はN回(単語数) 生成
  - $\beta=(\beta_1,\cdots,\beta_K)$ はトピック別単語分布 $p(w|k)$

グラフィカルモデルでいうと、我々にわかっているのは、出現した単語 w という黒丸だけです。この単語の黒丸の裏に見えない話題があって、この話題を作る見えない話題分布がある。これら全てを推定するという一見無茶に見えることをします。なお、話題分布を支配するハイパーパラメーターもあり、これも頑張れば推定できます。

LDAの確率モデル

$$p(w, z, \theta) = p(w|z)p(z|\theta)p(\theta|\alpha)$$

観測単語　トピック　トピック分布

よって、文書 $\mathbf{w} = w_1 w_2 \cdots w_N$ について

$$p(\mathbf{w}, z, \theta) = p(\theta|\alpha) \prod_n p(w_n|z_n)p(z_n|\theta)$$

$$p(\mathbf{w}) = \int \sum_z p(\mathbf{w}, z, \theta) d\theta$$

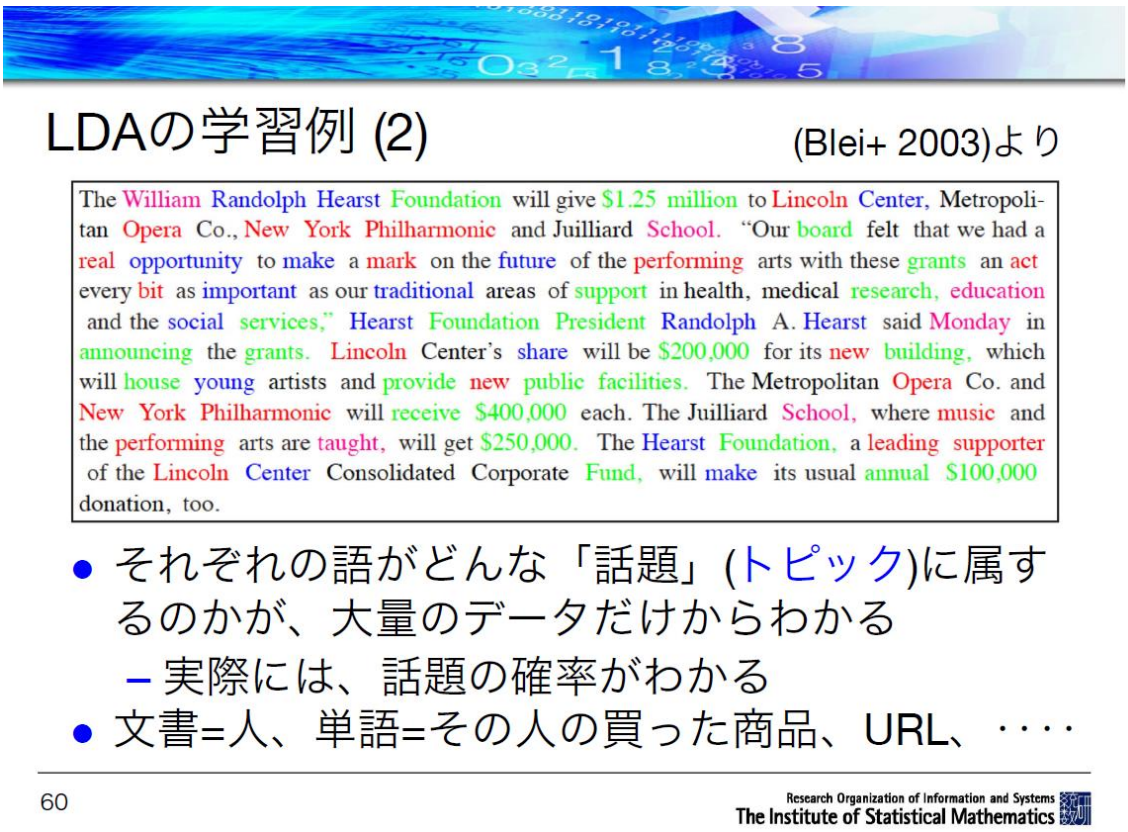
$$= \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \int \left(\prod_k \theta_k^{\alpha_k - 1} \right) \prod_n \sum_k p(w_n|k)\theta_k d\theta$$

- パラメータは α と $\beta = \{ p(w|k) \}$

数式で書くと、**w** というのが単語です。その裏に一個一個の単語が持っている話題 $\mathbf{z}_n$ と、文書が持っている話題分布 θ があります。ある単語ベクトル **w** が生成される確率 **p(w)**は、n 個の単語についての n 個の積についての、すべての話題の和（Σ_z）とすべての話題分布の和（θ についての積分）になります。θ の分布がディリクレ分布で、そこから θ_k という話題 k の出る確率が現れ、θ_k から単語が出てくる。パラメーターはディリクレ分布の α と、話題 k から単語 w の出る確率（上では β となっています)、これが全体のパラメーターです。

こんな尤度をもつモデルの最適化など無理なように思えますが、これが実は解けるのです。

LDA モデルの計算法を説明する前に、LDA を適用した時に得られる結果の事例を先に示しておきます。縦の列は、それぞれ LDA が求めた話題とその話題の下で選ばれる確率の高い単語です。話題には、単なる番号しか付いていません。この結果を見てどういう話題かは人が見て、話題の名前を決めています。例えば話題 1 には Art に関する言葉が多いから Art と名付けました。３番目には、子供に関する言葉が多いから children にしています。４番目は教育関係が並んでいるから education にしました。”Art”、"Budgets”、”Children”、”Education”は、モデルではなく図示のために与えた各話題の名前です。



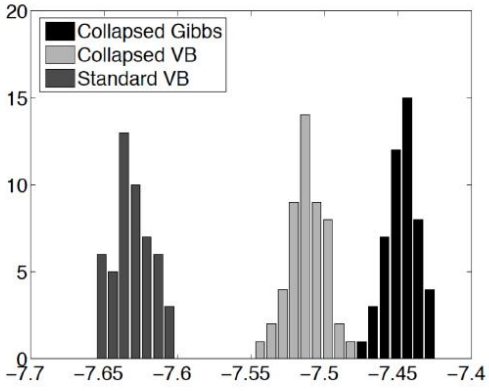
ここまでですと PLSI の結果と変わらないのですが、実際のテキストの単語ごとに話題があるので、テキストを話題で色分けできるのです。例えば、この文書の冒頭には William Randolph Hearst Fundation will give $1.25 million to と書いてあります。New York Philharmonic と、Juilliard School に Foundation をあげましたという話で、音楽関係の話とお金関係の話が混じっているのですが、New York Philharmonic とか performing とか Lincoln は音楽関係（赤）、緑色のところはお金関係、というように、単語ごとに（それを選ばせた）話題をあてることができるわけです。本当はどの話題かは確率分布として推定されますが、ここでは一番事後確率の高かった話題に単語を色づけしています。

LDAの解法

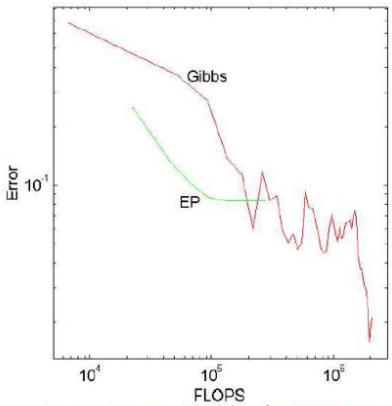
- どうやって解くか?
  - 変分ベイズ法 (Blei+ 2001,2003)　(オリジナル)
  - Gibbs サンプリング (Griffiths&Steyvers 2004)
  - 期待値伝播法 (Minka&Lafferty 2002)
  - Collapsed 変分ベイズ法 (Teh+ 2006)
  - 固有値計算 (!!) (Anandkumar+, arXiv 2012)

この複雑なモデルをどうやって解くかというと、実はモデルの提唱以来、たくさんの方法が提唱されています。

各学習法の比較



Teh+(2006)より (KOSコーパス)



EPとGibbsの比較 (無限混合モデル)

- 性能はGibbs>CVB>VB (Gibbsは局所解に陥り難い)
  - 計算の速さではVB>CVB>Gibbs、数倍程度

一番簡単なのは Gibbs サンプラーと呼ばれているもので、これだけ紹介します。

LDAの学習: Gibbs Sampler

- 導出や実装が簡単で、高性能
  - 最近は並列化も研究されている (Ihler+09など)
- Gibbs Samplerとは
  ‥マルコフ連鎖モンテカルロ法 (MCMC)の
  最も簡単な場合
  - 潜在変数を、分布ではなく条件つき分布から
    実際にサンプリング
    ＝単語の潜在トピックを次々とサンプリング
  - EMと違い、原理的に無限回繰り返せば、
    真の分布からのサンプル

Gibbs Sampler

- 潜在変数 $z_1, z_2, \cdots, z_N$ を持つ確率モデル

$$p(X, z_1, z_2, \cdots, z_N)$$

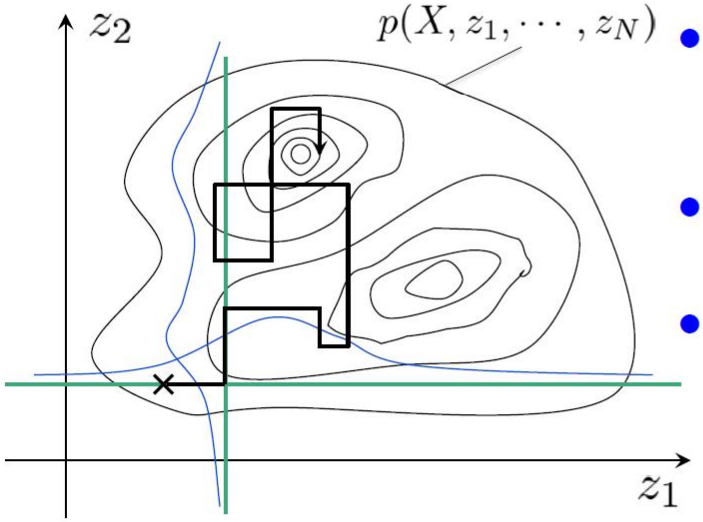
があるとき、各z_iを「考え直す」、つまり、
条件付き分布

$$z_i \sim p(z_i | X, z_1, \cdots, z_{i-1}, z_{i+1}, \cdots, z_N)$$

からランダムにサンプリングすることを繰り返す

真の分布に収束.

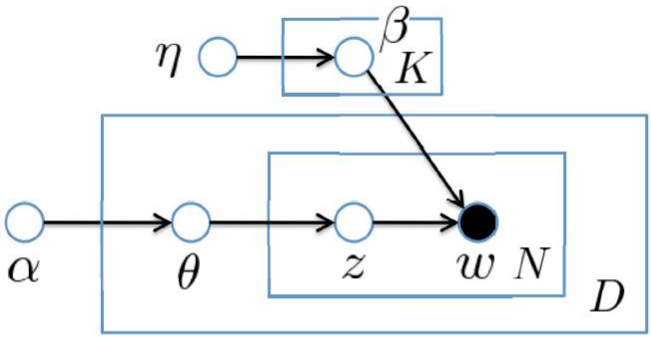
Gibbs Samplerのイメージ



- 条件つき確率に基づく、確率的な山登り
- 局所最適に陥らない
- 確率最大の一点に収束するわけではない
  - 分布全体からのサンプル

図の横軸と縦軸は、2 つの潜在変数 z_1 と z_2 です。z_1 と z_2 が特定の値のときにデータが得られる確率を、等高線状に示しています。データの確率が高くなるパラメーターの値を推定するために、マルコフ連鎖モンテカルロ法というものを使います。

LDAのGibbs Sampler



- LDAの潜在変数: θ (文書のトピック分布)と z (各単語のトピック)→実は z だけでよい
  - $z_i \sim p(z_i|\mathbf{w}, z_{-i}, \alpha, \eta)$ から、z_i を次々とサンプルして更新.

LDAのGibbs Sampler (2) (Griffiths+ 2004)

$$
\begin{aligned}
p(z_i = k|\mathbf{w}, z_{-i}) &\propto p(z_i = k, w_i|\mathbf{w}_{-i}, z_{-i}) \\
&= p(w_i|z_i = k, w_{-i}, z_{-i})\, p(z_i = k|\mathbf{w}_{-i}, z_{-i}) \\
&= \frac{\eta + n_{-i,k}^{(w_i)}}{\sum_w \left(\eta + n_{-i,k}^{(w)}\right)} \cdot \frac{\alpha_k + n_{-i,k}^{(d)}}{\sum_k \left(\alpha_k + n_{-i,k}^{(d)}\right)}
\end{aligned}
$$

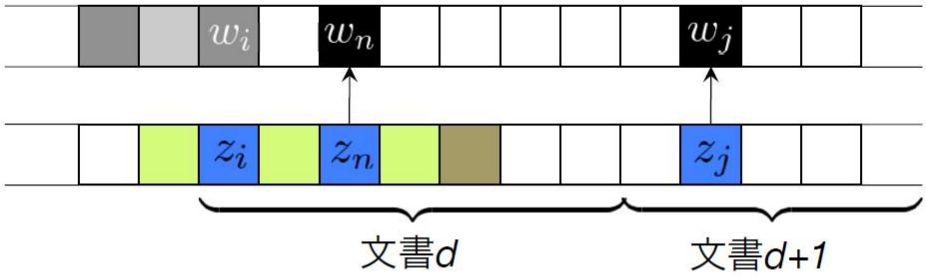
$n_{-i,k}^{(w)}$ データ全体で単語wがトピックkに割り当てられた回数 (w_i除く)

$n_{-i,k}^{(d)}$ 文書d中でトピックkに割り当てられた単語数 (w_i除く)

- $p(z|w,d) \propto p(z,w|d) = p(w|z)p(z|d)$ のような意味
  - 第2項では、$\int p(z_i = k|\theta)p(\theta|\alpha, \mathbf{w}_{-i}, z_{-i})d\theta$ として θ を積分消去

一個一個の単語、つまりある単語ベクトル $\mathbf{w}$ の中の i 番目の単語が持っている話題が k である確率は、ベイズでひっくり返すと、上のようになります。それが何かというと、ある単語が持っている話題の分布は、その単語が含まれている文章の中で、その話題がどのくらい出てくるかという事前分布と、その話題からその単語がどのくらい出てくるかという尤度をかけ合わせた形になります。それに従って、一個一個の単語がどういう話題なのかをサンプリングします。これがギブスサンプリングです。

LDAのGibbs Sampler (3) : イメージ



$$p(z_n = \blacksquare) \propto p(\blacksquare \mid \blacksquare)\, p(\blacksquare \mid d)$$
$$= p(w_n = \blacksquare \mid z_n = \blacksquare)\, p(z_n = \blacksquare \mid d)$$

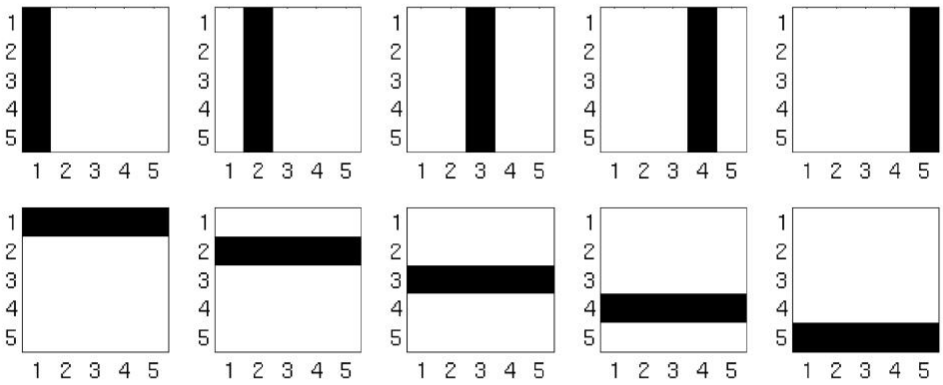
- 色の確率 ∝ (その文書内での色の割合)
  × (その色から単語の出る確率)

Research Organization of Information and Systems
The Institute of Statistical Mathematics

一個一個の単語が観測値で、その裏に見えない話題があります。話題を色で分類してあります。この黒の単語が持っている話題が青である確率を計算するのですが、青になる確率は、この文書の中で青がどのぐらい出てきたかという確率と、青という話題からこの黒い単語はどのぐらい出てくるか、この尤度を掛け合わせた形になります。それをこの文書という単語の配列に対してサンプリングを回します。通常、計算には一晩くらいかかります。

LDAのGibbs Sampler (4) : 学習例

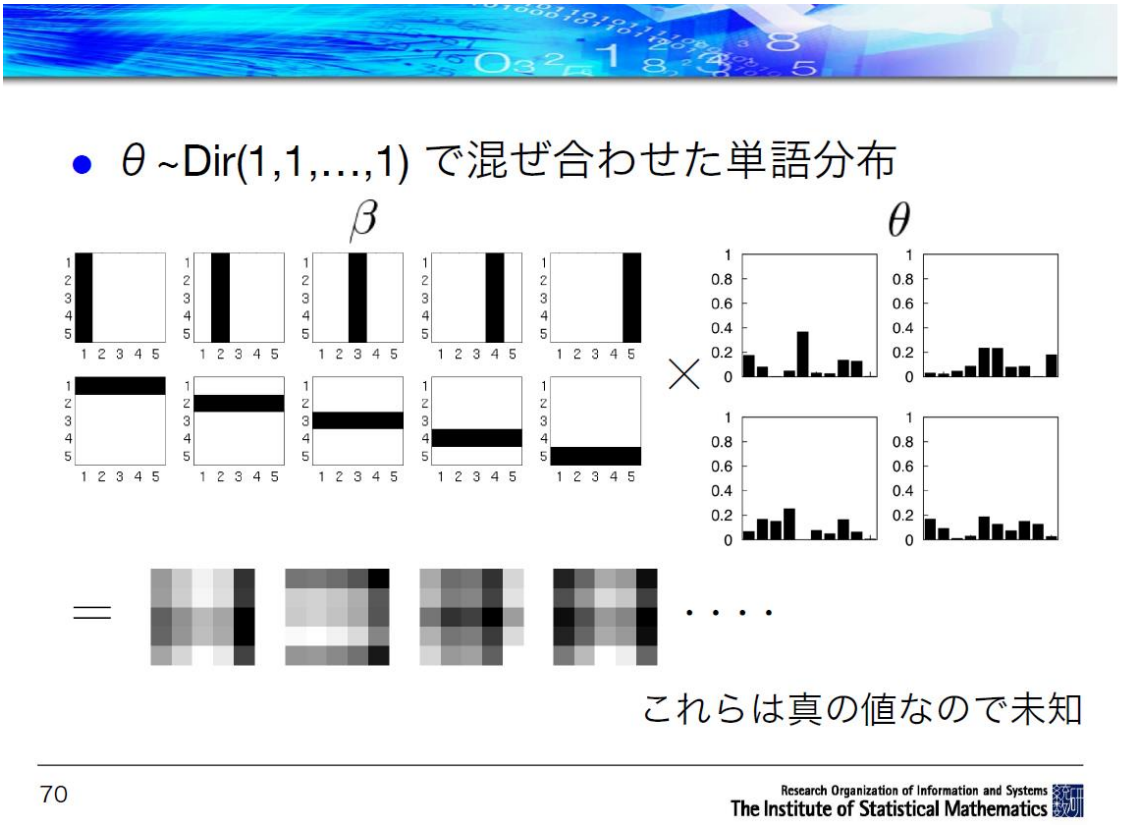
- 5x5=25単語上に10個の「トピック」β



  - 白は、その単語が出る確率が0
  - 黒は、その単語が出る確率が0.2

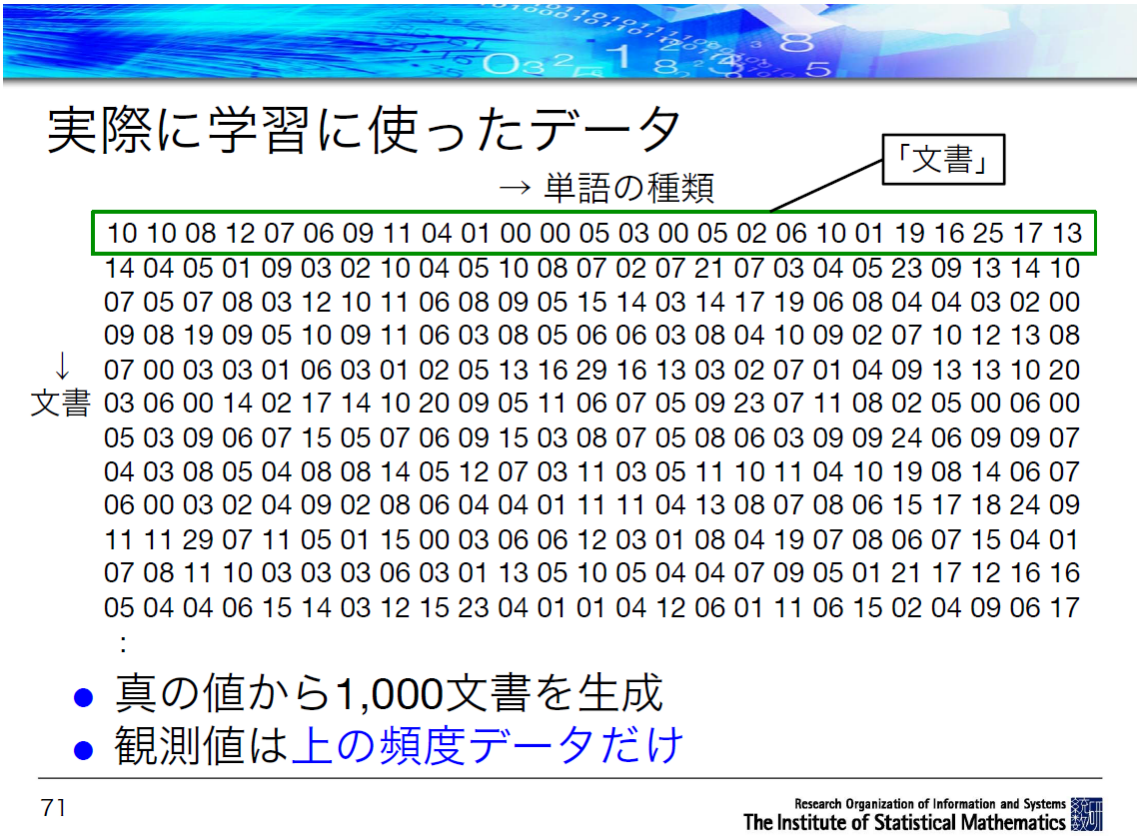
人工的に作った文書にギブスサンプリングを実際に行った例をお見せしましょう。まず、単語の種類を 25 とし、単語 1，単語 2，…，単語 25 を横に 5 つずつ、5 行に並べて表示することにします。トピックは 10 個としました。

10 個のトピックからそれぞれどんな単語が出てくるかを表すパラメータ $\beta_k = \{p(\mathbf{w}|k)\}$ を表すのが 10 個の正方形です。図の黒の部分は単語の出る確率が 0.2、それ以外は 0 を意味します。つまり左上のトピックでは、黒の部分は単語 1，6，11，16, 21 なので、それらが確率 0.2 ずつ出て、他は 0 という意味です。



次に、このトピックをどういう風に混ぜ合わせて文書を作るかを表す多項分布を、ディリクレ分布からランダムに選びます。今、文書を 4 つ作ることにし、一様なディリクレ分布(Dir(1,1,…,1))からランダムにサンプリングしたところ、右上のような 4 つの多項分布が選ばれたとしましょう。この割合に従って 10 個のトピックを混ぜ合わせて単語をサンプリングするので、10 個のトピックをそれぞれの多項分布で混ぜた分布を下に図示しました。一個一個の文書 1、文書 2、文書 3、文書 4 が持つ、各単語の出やすさを黒の濃さで表しています。

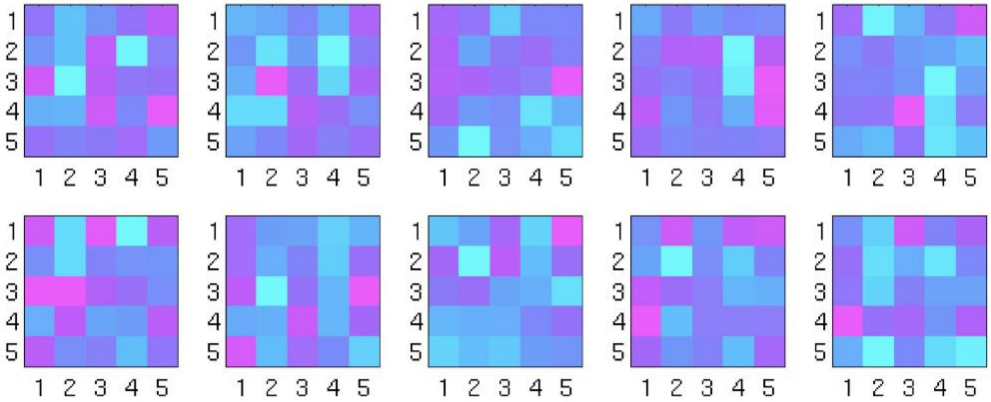
もちろん、我々はこの確率分布自体を知っているわけではなくて、ここからさらにサンプリングした単語の回数というデータを持っているだけです。



上がそうやって作った文書の例です。この例では 1000 個の文書を作っていて、図にはその最初の 12 個について、25 個の単語がそれぞれ何回出現したかを示しています。横軸が 1 個の文書です。文書 1 では、単語 1 が 10 回、単語 2 が 10 回、単語 3 が 8 回、単語 4 が 12 回、単語 5 が 7 回・・という感じで、単語 25 が 13 回出てきています。そういった文書が、文書 1、文書 2、文書 3・・とたくさん（全部で 1000 個）あります。我々が知っているのはこれだけです。β は教えません。そして、このカウントデータだけを使って、β が本当に復元できるのかを試します。つまり、β の真の分布は、ある 5 個が確率 0.2 で出て他は確率 0 という分布でした。もちろん我々は真の β を知りません。β を θ で混ぜ合わせ（混ぜ合わせ方 θ も知らない）、そこからさらに単語をランダムサンプリングして作ったカウントデータから、β を復元できるかテストするのです。

トピック分布 β の学習経過

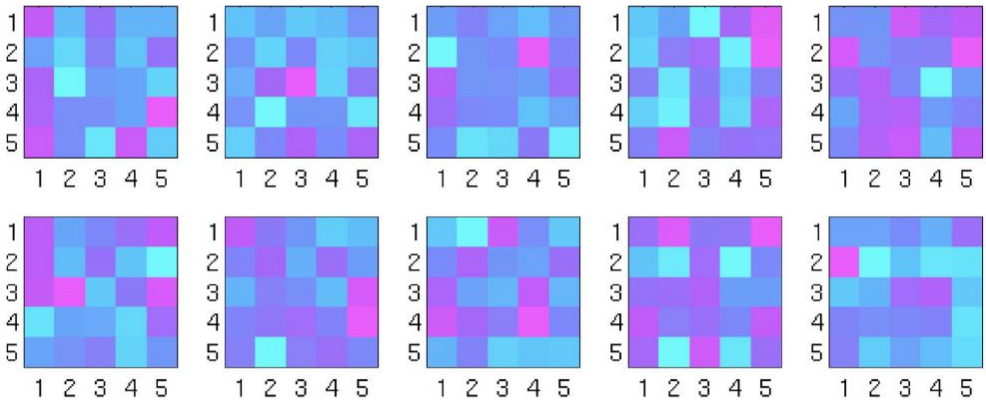
- Gibbs iteration = 1 (乱数で初期化)



最初はさっぱりわからないので、ひとまずランダムに β を定め、そこから少しずつ学習します。

トピック分布 β の学習経過

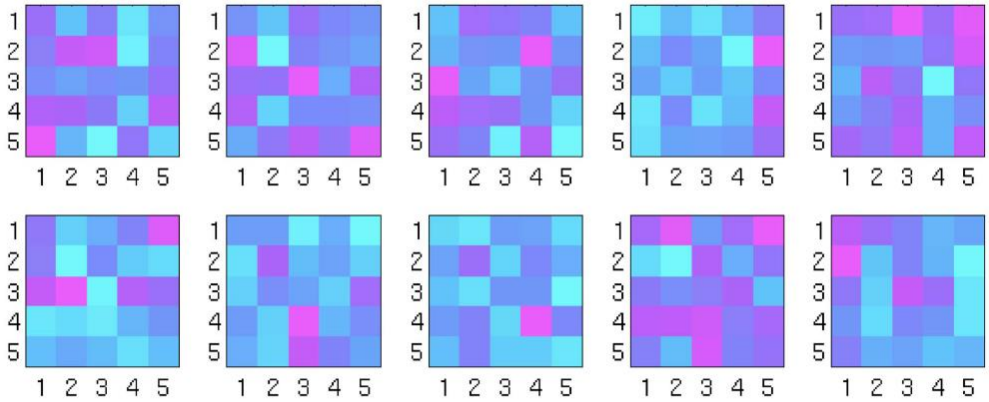
- Gibbs iteration = 2



それで、ギブスサンプラーを一回やります。少しきれいになったような、たいして変わらないような。

トピック分布 β の学習経過

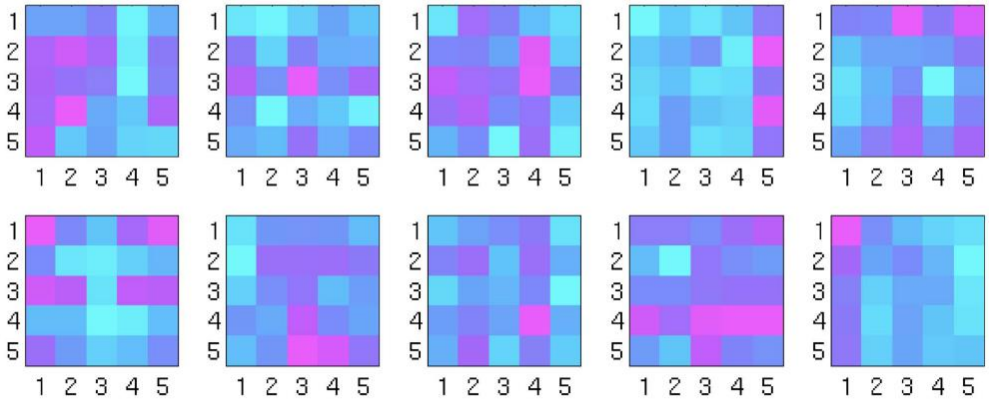
- Gibbs iteration = 4



ギブズサンプラーを 3 回やったところです。

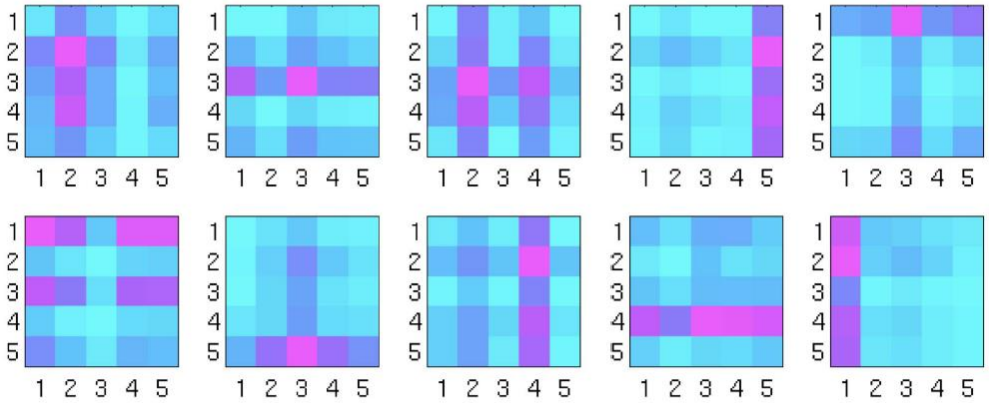
トピック分布 β の学習経過

- Gibbs iteration = 8



トピック分布 β の学習経過

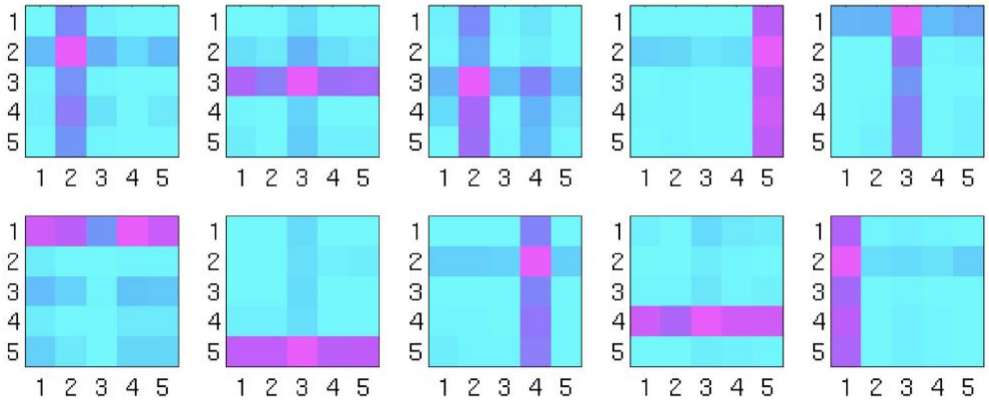
- Gibbs iteration = 16



Research Organization of Information and Systems
The Institute of Statistical Mathematics

トピック分布 β の学習経過

- Gibbs iteration = 32

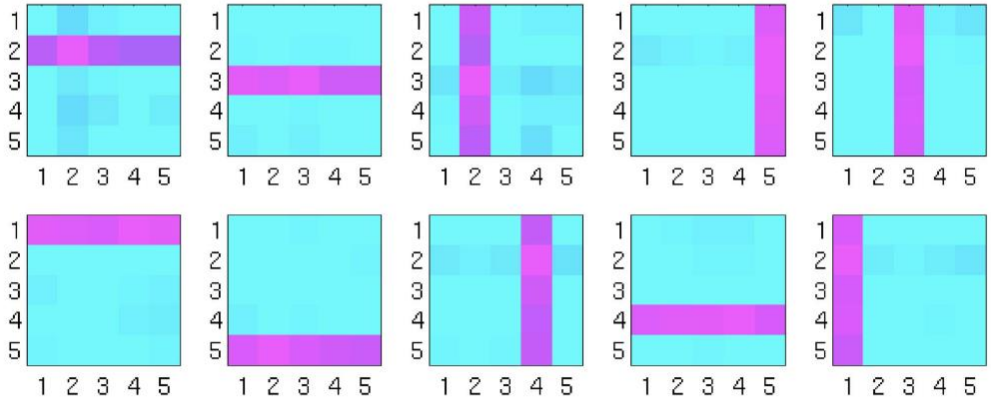


16 回で結構きれいになっていて、32 回でかなり真の分布に近づいています。

64 回だと、もうほぼ復元できています。

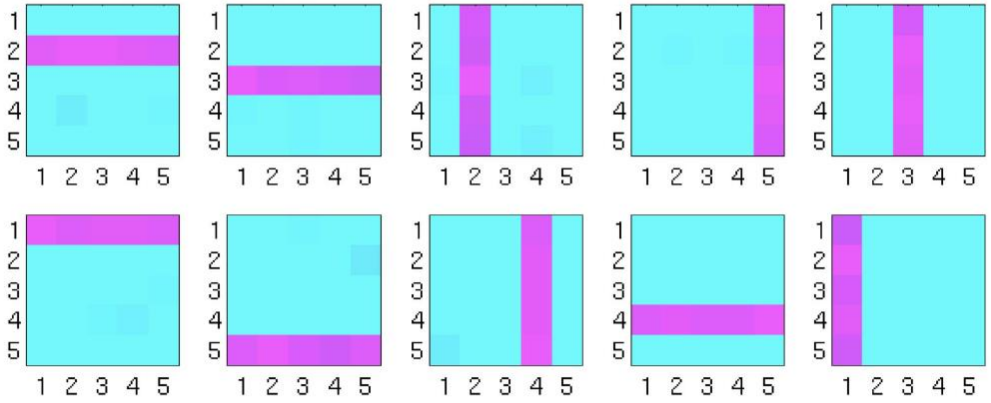
トピック分布 β の学習経過

- Gibbs iteration = 64



トピック分布 β の学習経過

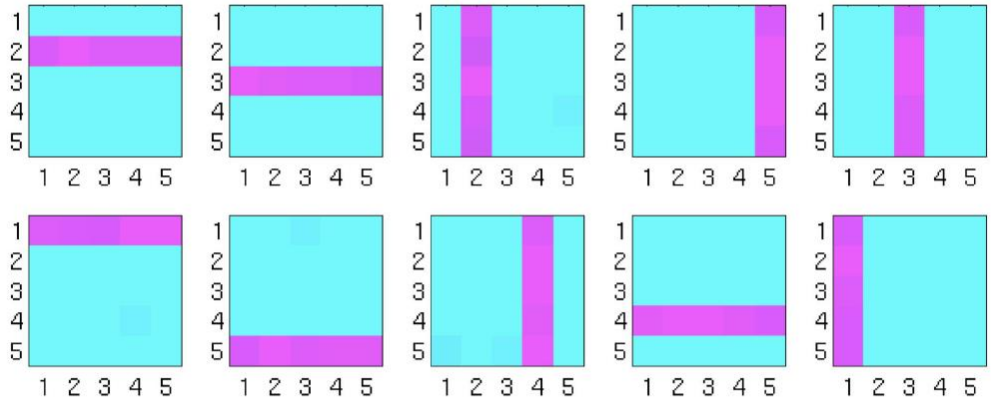
- Gibbs iteration = 128



Research Organization of Information and Systems
The Institute of Statistical Mathematics

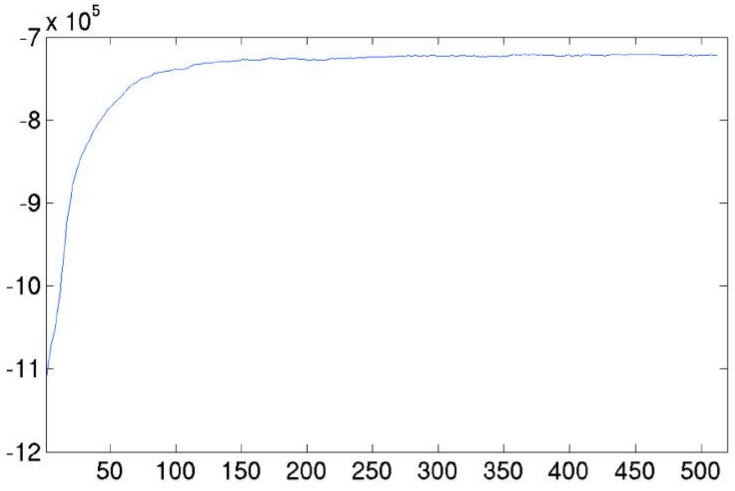
トピック分布 β の学習経過

- Gibbs iteration = 256



10 個のトピックの順番は関係ないので、この段階で真の分布はほぼ復元できたといえます。

対数尤度の変化



↑
データの
対数尤度

- 200 iterationあたりでほぼ収束

データの対数尤度も、最初はだーっと上がっていき、先ほどほぼ正解だった 64 あたりから横ばいになっています。

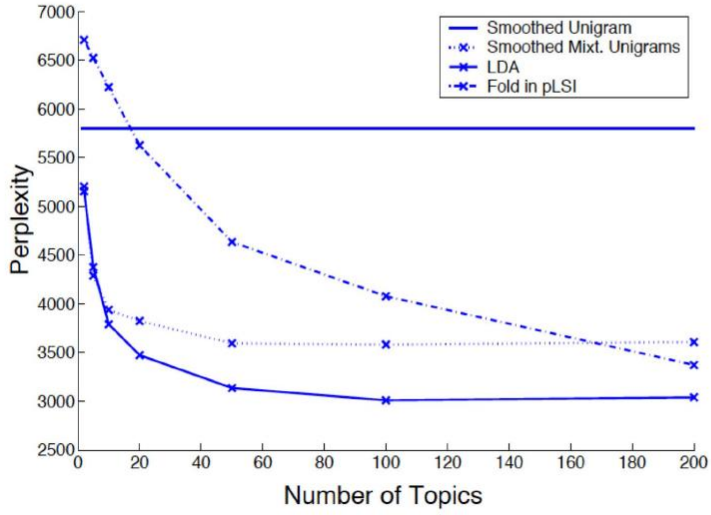
LDAとPLSIの比較

- LDAは、PLSIのベイズ化
- 様々な良い点 (デファクトスタンダード)
  - オーバーフィットしない
  - 完全な階層ベイズ生成モデル
    → 様々な拡張が可能 (明日の講義)
    - PLSIでは p(z|d) に由来がないため、これ以上手をつけることができない
  - 計算量のオーダーはほとんど同じ
    - モデルが完全なので、色々なオンライン推定法も提案されている

以上が LDA というものです。オーバーフィットしないトピックモデルで、かつ、完全な生成モデルなので、いろいろと拡張することができます。それでいて、計算オーダーはほとんどナイーブベイズと同じくらいです。

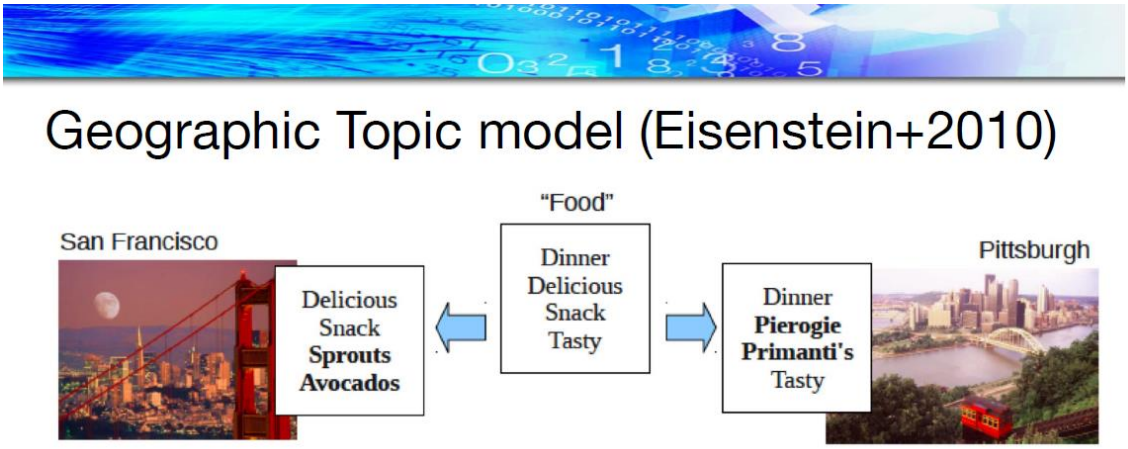
LDAとPLSIの比較 (2)

- PPLの比較 (Blei+ 2003より)



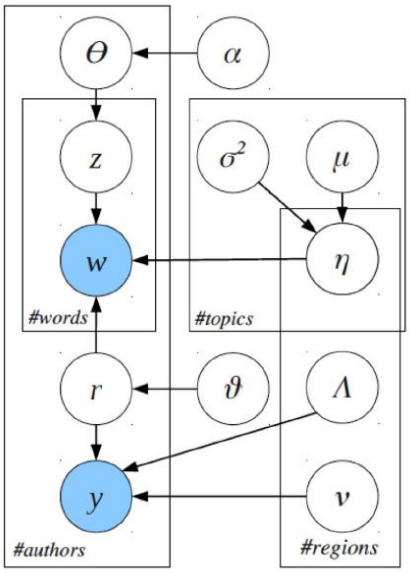
現在では、単純なLDAよりさらに性能の良いモデルが開発されている

モデルの性能の比較ですが、今回は特に説明しないことにします。



最後に、Geographic Topic model というのが面白かったので、紹介しようと思います。
同じトピックでも、地域によって言葉というものは微妙に違っていて、例えばこの例で Food とは Dinner や Delicious、Snack などですけれど、サンフランシスコに行くと Food というと Sprouts や Avocados などのほうが結構確率が高い。逆にピッツバーグだと、この単語の意味は知らないですが、レストランの名前などが多かったりします。全体としては、多分別々のものではなく、あくまで Food ですね。
大事なのは、地域がサンフランシスコやピッツバーグなどと分かっていれば、地域ごとに単語が出現した回数を数えればよいのですが、そもそも、どこまでが地域かということがよくわかりません。例えば北海道地域という風に区分できればよいのですが、北海道から東北のここまでが一個のまとまりをもった地域なのかもしれません。そうした地域自体をデータから見つけたい。そのような問題です。

Geographic Topic model (2)



- データ：緯度経度(y)つきTwitterツイート(w)
- yは、領域の混合正規分布
- 領域ごとに、基本のトピック分布に正規ノイズを加える
- 学習はMCMC

方言みたいなものですが、データとしては、Twitter のツイートを使っているので緯度経度がわかっています。緯度経度からでは領域が見えないわけです。

まず、地域をガウス混合分布で表現します。つまり、その地域の中心があり、そこから離れるに従ってその地域性は薄れていく。このとき、地域ごとのガウス分布は基本となるガウス分布にノイズが加わったものだと仮定します。このガウス分布をロジスティック変換して和を 1 にした分布がトピックで、それを使って MCMC で学習を動かします。どういう地域があるかということと、そこからどういう単語が出てくるかの両方を知ろうというのです。

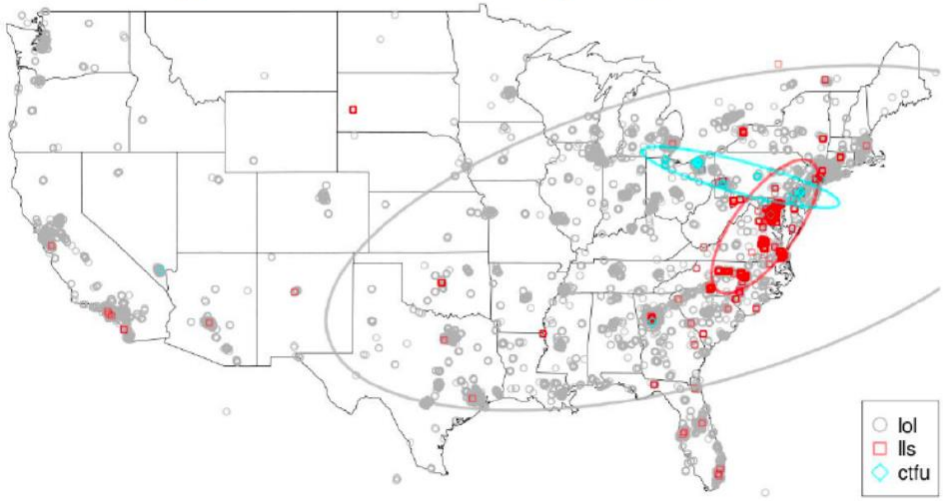
Geographical lexical variation (3)

	"basketball"	"popular music"	"daily life"	"emoticons"	"chit chat"
	PISTONS KOBE LAKERS game DUKE NBA CAVS STUCKEY JETS KNICKS	album music beats artist video #LAKERS ITUNES tour produced vol	tonight shop weekend getting going chilling ready discount waiting iam	:) haha :d :(;) :p xd :/ hahaha hahah	lol smh jk yea wyd coo ima wassup somethin jp
Boston 	CELTICS victory BOSTON CHARLOTTE	playing daughter PEARL alive war comp	BOSTON	;p gna loveee	*ese* exam suttin sippin
N. California 	THUNDER KINGS GIANTS pimp trees clap	SIMON dl mountain seee	6am OAKLAND	*pues* hella koo SAN fckn	hella flirt hut iono OAKLAND
New York 	NETS KNICKS	BRONX	iam cab	oww	wasssup nm
Los Angeles 	#KOBE #LAKERS AUSTIN	#LAKERS load HOLLYWOOD imm MICKEY TUPAC	omw tacos hr HOLLYWOOD	af *papi* raining th bomb coo HOLLYWOOD	wyd coo af *nada* tacos messin fasho bomb
Lake Erie 	CAVS CLEVELAND OHIO BUCKS od COLUMBUS	premiere prod joint TORONTO onto designer CANADA village burr	stink CHIPOTLE tipsy	;d blvd BIEBER hve OHIO	foul WIZ salty excuses lames officer lastnight

これは food ではなく basketball 関連らしいのですが、全体としては、PISTONS や LAKERS などが出てきます。でも、Boston だと CELTICS が出てくるし、New York だと KNICKS が出てきます。他の例で、popular music だと、playing, daughter, PEARL が Boston では出てきますが、New York では BRONX が出てきます。Emoticons (顔文字)というのは面白くて、この haha という笑いマークがアメリカ全土で同じかというとそんなことはなくて、ボストンでは図のようなものを使うけれど California では別なものを使い、New York ではまた別なものが使われています。

Geographical lexical variation (4)

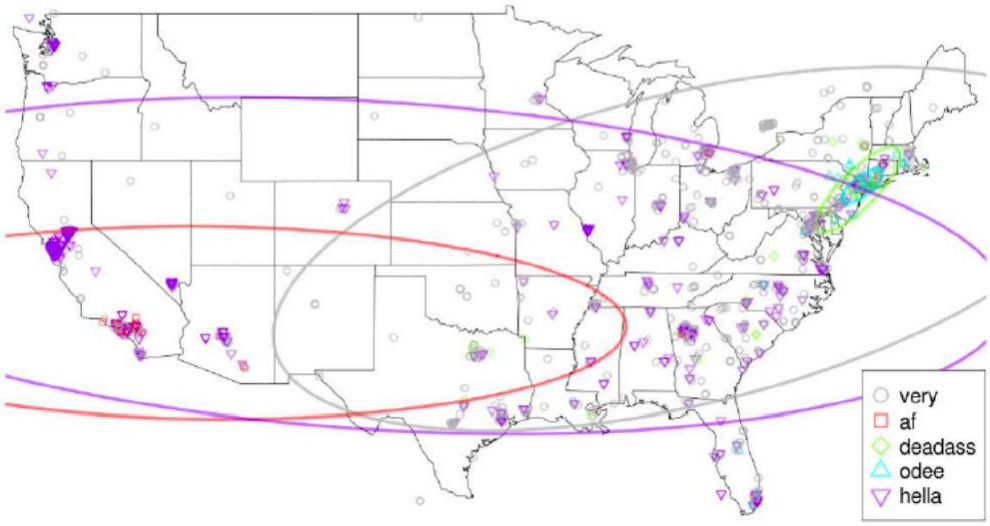
- lol=laugh out loud, lls=laughing like shit



ガウス混合分布なので、この辺り、この辺りとたくさんあるんですが、そのうちの一部をとってきた例で、lol というのは、日本語で言うと（笑）です。（笑）は、全体的に散らばっています。全国に散らばっているのですが、lls（laughing like shit という少し下品な表現）があるらしく、それはごく一部にしかありません。

Geographical lexical variation (5)

- hella=very,



Very は普通の表現で全国にあるのですが、カリフォルニアには hella というのがあるらしく、それがこういった感じのガウス分布で説明できるようです。af というのは、書こうと思いましたが品がないのでやめました。odee はなにかの四文字言葉のようです。ほかにも色々最近の話を用意してきましたが、今回はこのあたりにしておくのがよいかと思います。

編集後記

実際のチュートリアルは、この後活発な質疑が続き、そこからはチュートリアルならではの躍動感が感じ取れると期待できますが、テープ起こしの負担は重く、また、入門講義としては、ナイーブベイズ、Unigram Mixture, PLSI, LDA と 1 つずつ 4 つのステップを登ったところがひとつの切れ目になると考え、本講義録はここで打ち切ることにしました。

トピックモデルについては、佐藤一誠著「トピックモデルによる統計的潜在意味解析」(2015 年)、岩田具治著「トピックモデル」 (機械学習プロフェッショナルシリーズ、2015 年) のように和文教科書も出版され始めています。ただ、統計数理系の読者を満足させる書物になっているため、非統計数理系の研究者が独習するには数式が多く難儀するかもしれません。本講義録は、生物系研究者中心のチュートリアルだったため、計算アルゴリズムの詳細を大胆に割愛し、モデルの生成過程やアウトプットの解釈など、エンドユーザーにとって最も重要な部分に重点を置いたものとなっています。専門書と合わせて活用することで、ぜひこの新しい統計モデルの活用法を覚え、新しいデータ科学の世界を切り拓いていきましょう。