自然言語処理プログラミング勉強会 7 - トピックモデル

Graham Neubig
奈良先端科学技術大学院大学 (NAIST)

文章のトピック

- 文章には様々なトピックが存在する

Cuomo to Push for Broader Ban on Assault Weapons … … … …	2012 Was Hottest Year in U.S. History … … … …

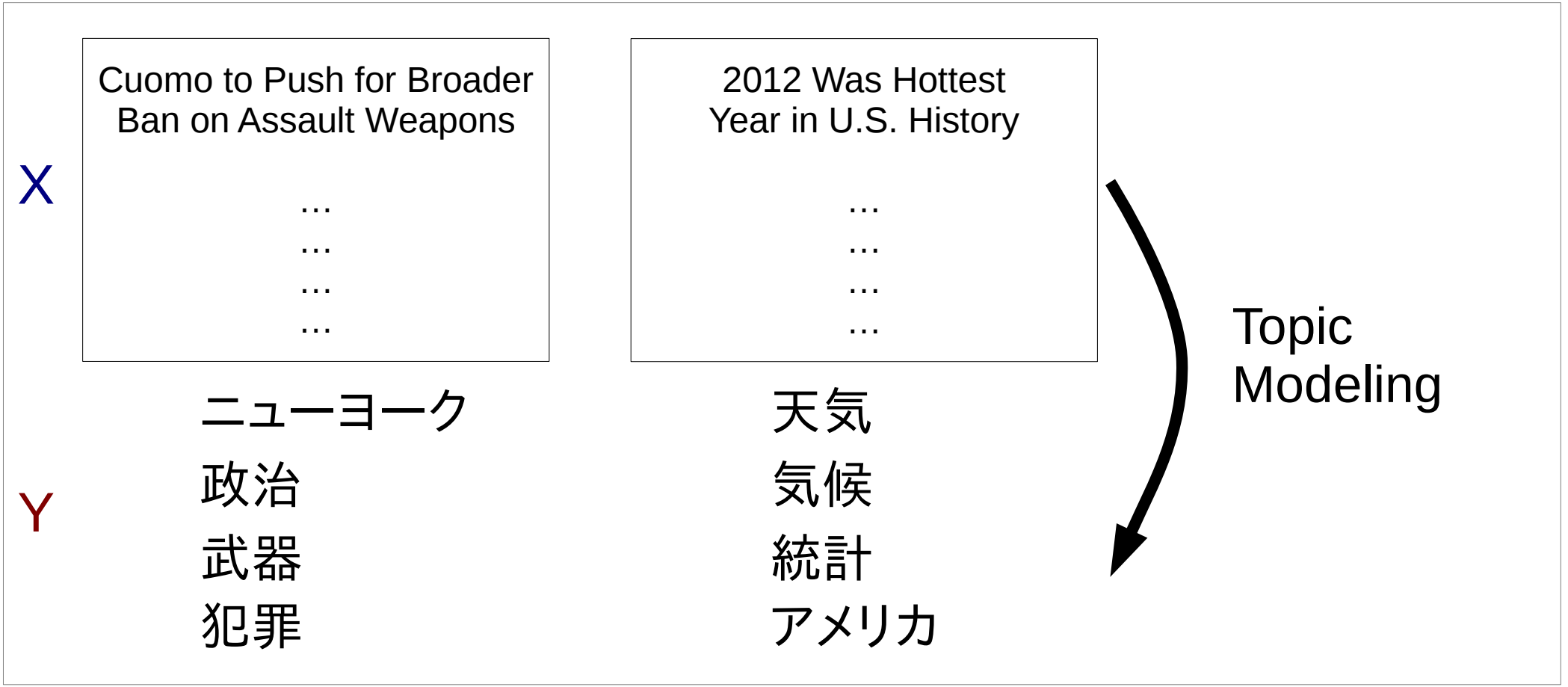
文章のトピック

- 文章には様々なトピックが存在する

Cuomo to Push for Broader Ban on Assault Weapons … … … …	2012 Was Hottest Year in U.S. History … … … …
ニューヨーク	天気
政治	気候
武器	統計
犯罪	アメリカ

トピックモデル

- トピックモデルでは文章 X に対してトピック Y を発見



確率的生成モデル

- 文章 X とトピック Y が何かの過程によって同時に生成されたとする

$$P(Y, X)$$

- 同時確率が高ければ、条件付き確率も高い：

$$\underset{Y}{\operatorname{argmax}}\, P(Y|X) = \underset{Y}{\operatorname{argmax}}\, P(Y, X)$$

トピックを考慮した文の生成モデル

- 単語列 X とトピック列 Y:

X = Cuomo to Push for Broader Ban on Assault Weapons

Y = NY 機能 政治 機能 政治 政治 機能 犯罪 犯罪

- まずトピックを独立に生成:

$$P(Y)=\prod_{i=1}^{I} P(y_i)$$

- その次、各単語をトピックに基づいて生成:

$$P(X|Y)=\prod_{i=1}^{I} P(x_i|y_i)$$

$$P(X,Y)=P(X|Y)P(Y)=\prod_{i=1}^{I} P(x_i|y_i)P(y_i)$$

トピックが付与された場合の確率学習

X = Cuomo to Push for Broader Ban on Assault Weapons

Y = NY 機能 政治 機能 政治 政治 機能 犯罪 犯罪

- 最尤推定で学習可能

トピック確率

$$P(y=NY) = c(y=NY)/|Y| = 1/9$$

$$P(y=\text{政治}) = c(y=\text{政治})/|Y| = 3/9$$

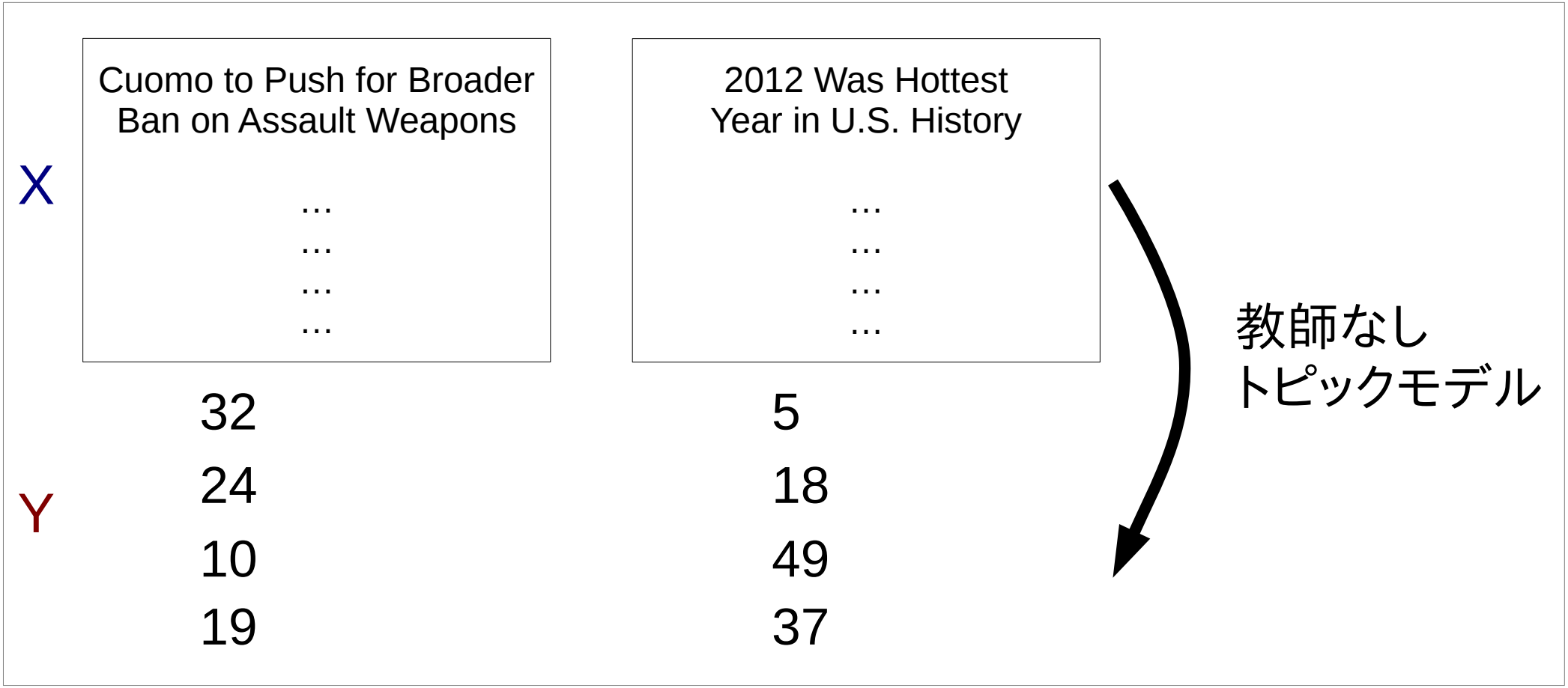
単語確率

$$P(x=\text{Assault}|y=\text{犯罪}) = c(x=\text{Assault}, y=\text{犯罪})/c(y=\text{犯罪}) = 1/2$$

- (実際は文ではなく、文章)

教師なしトピックモデル

- 文章 X のみからトピックらしいクラス Y を発見



- 前と違って Y の記された学習データがない！
- どうしよう…

教師なし学習

- 観測変数 X、隠れ変数 Y、パラメータ θ に対する分布を定義

$$P(\Theta, Y, X)$$

  - (θ はモデル確率を定義、Y はある X に対応、という差)
- これを使って、例えば最尤推定、で θ と Y を推定

$$\hat{\Theta}, \hat{Y} = \underset{\Theta, Y}{\operatorname{argmax}} P(\Theta, Y, X)$$

潜在的ディリクレ配分法 (Latent Dirichlet Allocaton: LDA)

- トピックモデルの中で最も一般的
- まずモデルのパラメータ θ を生成: $P(\theta)$
- 各文章に対して X:
  - 文章のトピック分布 T_i を生成: $P(T_i|\theta)$
  - X_i の各単語 $x_{i,j}$ に対して:
    - トピック $y_{i,j}$ を生成: $P(y_{i,j}|T_i,\theta)$
    - 単語 $x_{i,j}$ を生成: $P(x_{i,j}|y_{i,j},\theta)$

$$P(X,Y)=\int_\theta P(\theta)\prod_i P(T_i|\theta)\prod_j P(y_{i,j}|T_i,\theta)P(x_{i,j}|y_{i,j},\theta)$$

最尤推定

- 単語 X とトピック Y が与えられたとしたら:

X_1 = Cuomo to Push for Broader Ban on Assault Weapons

Y_1 = 32 7 24 7 24 24 7 10 10

- 各文章のトピック分布を決定:

$$P(y|Y_i) = c(y, Y_i)/|Y_i| \quad \text{e.g.: } P(y=24|Y_1) = 3/9$$

- 各トピックの単語分布を決定:

$$P(x|y) = c(x, y)/c(y) \quad \text{e.g.: } P(x=\text{assault}|y=10) = 1/2$$

隠れ変数

- 問題 : $y_{i,j}$ の値は与えられていない
- 解決策 : 教師なし学習を利用
- 教師なし学習の手法例:
  - EM アルゴリズム
  - 変分ベイズ
  - **サンプリング**

サンプリングの例

- ある分布に従ってサンプルを生成：

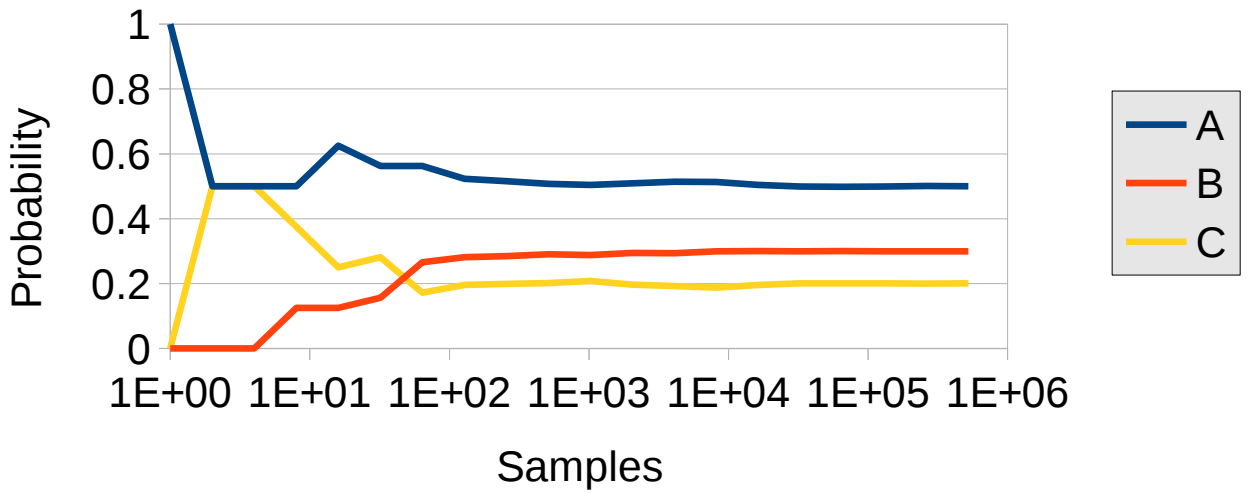
分布：　　P(A)=0.5　P(B)=0.3　P(C)=0.2

サンプル：　　B　B　C　A　A　C　A　B　B　A ...

- サンプルを数え上げて割ったら確率が近似可能：

P(A)= 4/10 = 0.4,　　P(B)= 4/10 = 0.4,　　P(C) = 2/10 = 0.2

- サンプルが増えれば近似の精度も増える：



アルゴリズム

```
SampleOne(probs[])
    z = Sum(probs)
    remaining = Rand(z)
    for each i in 0 .. probs.size-1
        remaining -= probs[i]
        if remaining <= 0
            return i
```

確率の和 (正規化項) を計算

[0,z) の乱数を一様分布によって生成

probs の各項目を検証

現在の確率を引く

0 より小さい場合、返す

全ての確率が終わっても返さない場合はバグでエラー終了！

ギブスサンプリング

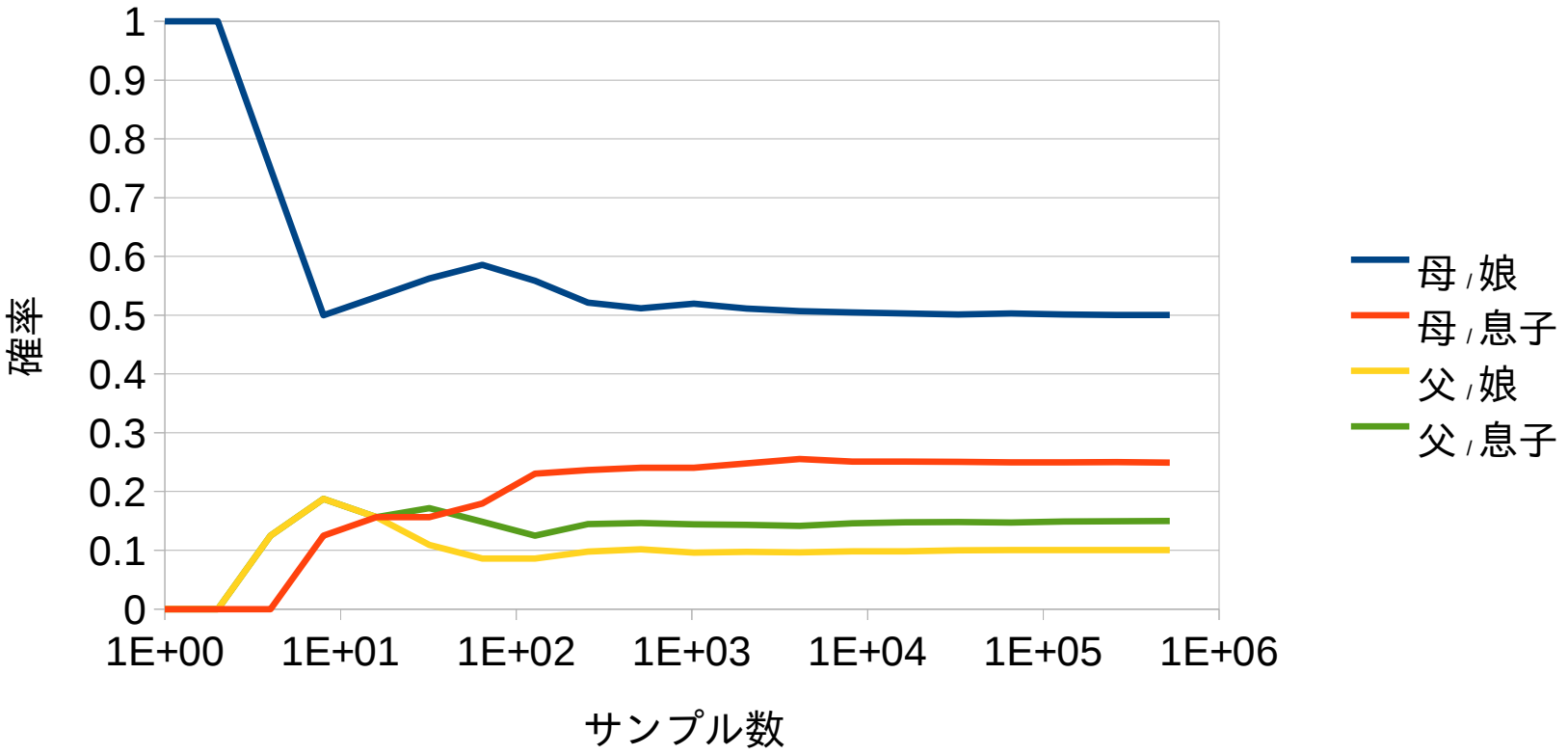
- 2 つの変数を分布 P(X,Y) からサンプルしたい
  - … P(X,Y) からサンプリングすることが不可
  - … P(X|Y) と P(Y|X) からサンプリングすることが可
- ギブスサンプリングでは、変数を 1 個ずつサンプリングする
- 各イタレーション:
  X を固定し、Y を P(Y|X) に従ってサンプリング
  Y を固定し、X を P(X|Y) に従ってサンプリング

ギブスサンプリングの例

- 親 A と子 B は買い物している、それぞれの性別は？
  P(母 | 娘) = 5/6 = 0.833　P(母 | 息子) = 5/8 = 0.625
  P(娘 | 母) = 2/3 = 0.667　P(娘 | 父) = 2/5 = 0.4
- 初期状態：母 / 娘
  A をサンプル：P(母 | 娘)=0.833，母を選んだ！
  B をサンプル：P(娘 | 母)=0.667，息子を選んだ！
  c(母 , 息子)++
  A をサンプル：P(母 | 息子)=0.625，母を選んだ！
  B をサンプル：P(娘 | 母)=0.667，娘を選んだ！
  c(母 , 娘)++

…

実際にやってみると



- 同時確率の式を手で解いてこの結果を確認できる

トピックモデルのサンプリング (1)

- $y_{i,j}$ を１つずつ:

X_1 = Cuomo to Push for Broader Ban on Assault Weapons
Y_1 = 5 7 4 7 3 4 7 6 6

- まず $y_{i,j}$ をカウントから削除、確率を再計算

{0, 0, 1/9, 2/9, 1/9, 2/9, 3/9, 0}

{0, 0, 1/8, 2/8, 1/8, 2/8, 2/8, 0}

トピックモデルのサンプリング (2)

- $y_{i,j}$ を１つずつ:

X_1 = Cuomo to Push for Broader Ban on Assault Weapons
Y_1 = 5 7 4 ??? 3 4 7 6 6

- トピック確率と単語確率を掛け合わせる:

コーパス全体から計算

$P(y_{i,j} \mid Y_i)$ = { 0, 0, 0.125, 0.25, 0.125, 0.25, 0.25, 0}

*

$P(x_{i,j} \mid y_{i,j}, \theta)$ ={0.01, 0.02, 0.01, 0.10, 0.08, 0.07, 0.70, 0.01}

=

$P(x_{i,j}\ y_{i,j} \mid Y_i, \theta)$={ 0, 0,0.00125,0.01,0.01,0.00875,0.175, 0}/Z

正規化係数

トピックモデルのサンプリング (3)

- 確率分布から１つの値をサンプリング:

$$P(x_{i,j}, y_{i,j} \mid T_i, \theta) = \{0, 0, 0.00125, 0.01, 0.01, 0.00875, 0.175, 0\}/Z$$

- トピックを更新:

X_1 = Cuomo to Push for Broader Ban on Assault Weapons
Y_1 = 5 7 4 6 3 4 7 6 6

- カウントと確率を更新:

{0, 0, 1/8, 2/8, 1/8, 2/8, 2/8, 0}

{0, 0, 1/9, 2/9, 1/9, 3/9, 2/9, 0}

ディリクレ分布による平滑化:

- 問題: 多くのカウントが 0→ 多くの確率が 0
  → 局所解に陥る
- 解決策: 確率の平滑化

平滑化なし

$$P(x_{i,j}|x_{i,j}) = \frac{c(x_{i,j}, y_{i,j})}{c(y_{i,j})}$$

平滑化有り

$$P(x_{i,j}|y_{i,j}) = \frac{c(x_{i,j}, y_{i,j}) + \alpha}{c(y_{i,j}) + \alpha * N_x}$$

$$P(y_{i,j}|Y_i) = \frac{c(y_{i,j}, Y_i)}{c(Y_i)} \rightarrow P(y_{i,j}|Y_i) = \frac{c(y_{i,j}|Y_i) + \beta}{c(Y_i) + \beta * N_y}$$

- N_x と N_y はそれぞれ単語とトピックの異なり数
- 確率に対してディリクレ分布に基づく事前分布の利用と等しい(LDA の論文を参照)

実装:初期化

```
make vectors xcorpus, ycorpus   # 各 x, y を格納
make map xcounts, ycounts       # カウントの格納
for line in file
    docid = size of xcorpus     # この文章の ID を獲得
    split line into words
    make vector topics          # 単語のトピックをランダム初期化
    for word in words
        topic = RAND(NUM_TOPICS)    # [0,NUM_TOP) の間
        append topic to topics
        ADDCOUNTS(word, topic, docid, 1)  # カウントを追加
    append words (vector) to xcorpus
    append topics (vector) to ycorpus
```

実装:カウントの追加

AddCounts(*word, topic, docid, amount*)

xcounts["topic"] += amount
xcounts["word|topic"] += amount

$$P(x_{i,j}|y_{i,j}) = \frac{c(x_{i,j}, y_{i,j}) + \alpha}{c(y_{i,j}) + \alpha * N_x}$$

ycounts["docid"] += amount
ycounts["topic|docid"] += amount

$$P(y_{i,j}|Y_i) = \frac{c(y_{i,j}, Y_i) + \beta}{c(Y_i) + \beta * N_y}$$

バグチェック
　＜０の場合はエラー終了

実装:サンプリング

```
for many iterations:
  for i in 0:Size(xcorpus):
    for j in 0:Size(xcorpus[i]):
      x = xcorpus[i][j]
      y = ycorpus[i][j]
      AddCounts(x, y, i, -1)          #  各カウントの減算 (-1)
      make vector probs
      for k in 0 .. NUM_TOPICS-1:
        append P(x|k) * P(k|Y) to probs    #  トピック k の確率
    new_y = SampleOne(probs)
    ll += log(probs[new_y])           #  対数尤度の計
    AddCounts(x, new_y, i, 1)         #  各カウントの加算
    ycorpus[i][j] = new_y
  print ll

print out xcounts and ycounts
```

演習課題

Exercise

- 実装 learn-lda
- テスト (NUM_TOPICS=2)
  - 入力 : `test/07-train.txt`
  - 正解 :
    - 正解はない！ (サンプリングはランダムなので)
    - しかし、「ａｂｃｄ」と「ｅｆｇｈ」に分かれる確率が高い
- 学習 `data/wiki-en-documents.word` を使って
- 検証 発見されたトピックは直感に沿うのか？(機能語を削除して、各トピックで頻度の高い内容語を見ると良い)
- チャレンジ トピック数を事前に決めなくても良いようにモデルを変更 (ノンパラメトリックベイズで検索)

Thank You!