Topic Models AO:Chapter 4.3:PLSA

@anemptyarchive

2019/12/06-2019/12/07

Contents

4.3 PLSA

・コード全体

```
# 利用パッケージ
library(RMeCab)
library(tidyverse)
```

・テキスト処理

```
## 抽出する単語の指定
# 品詞（大分類）を指定
PoS_1 <- "名詞|^動詞|形容詞"

# 品詞（細分類）を指定
PoS_2 <- "一般|^自立"

# 最低出現頻度を指定
Freq <- 5

# 抽出しない単語を指定
stop_words <- "[a-z]"

# 形態素解析
mecab_df <- docDF("フォルダ名", type = 1) # テキストファイルの保存先を指定する

# 文書dの語彙vの出現回数（N_dv）の集合
N_dv <- mecab_df %>%
        filter(grepl(PoS_1, POS1)) %>%         # 指定した品詞（大分類）を取り出す
        filter(grepl(PoS_2, POS2)) %>%         # 指定した品詞（細分類）を取り出す
        filter(!grepl(stop_words, TERM)) %>%  # ストップワードを除く
        select(-c(TERM, POS1, POS2)) %>%       # 数値列のみを残す
        filter(apply(., 1, sum) >= Freq) %>%  # 指定した頻度以上の語彙を取り出す
        t()                                    # 転置

# 確認用の行列名
dimnames(N_dv) <- list(paste0("d=", 1:nrow(N_dv)), # 行名
                       paste0("v=", 1:ncol(N_dv))) # 列名

# 文書dの単語数（N_d）のベクトル
N_d <- apply(N_dv, 1, sum) # 行方向に和をとる
```

```r
# 文書数 (D)
D <- nrow(N_dv)

# 総語彙数 (V)
V <- ncol(N_dv)
```

・パラメータの初期設定

```r
# トピック数 (K)
K <- 4   # 値を指定する

# 負担率 (q_dvk) の集合
q_dvk <- array(0, dim = c(D, V, K),
               dimnames = list(paste0("d=", 1:D),
                               paste0("v=", 1:V),
                               paste0("k=", 1:K))) # 確認用

## トピック分布 (theta_dk) の集合
# 値をランダムに付ける
theta_dk <- sample(seq(0.1, 1, 0.01), D * K, replace = TRUE) %>%  # ランダムな値を生成
            matrix(nrow = D, ncol = K,
                   dimnames = list(paste0("d=", 1:D),  # 行名
                                   paste0("k=", 1:K))) # 列名

# 初期値の正規化
theta_dk <- theta_dk / apply(theta_dk, 1, sum)

## 単語分布 (phi_kv) の集合
# 値をランダムに付ける
phi_kv <- sample(seq(0, 1, 0.01), K * V, replace = TRUE) %>%  # ランダムな値を生成
          matrix(nrow = K, ncol = V,
                 dimnames = list(paste0("k=", 1:K),  # 行名
                                 paste0("v=", 1:V))) # 列名

# 初期値の正規化
phi_kv <- phi_kv / apply(phi_kv, 1, sum)
```

・PLSA

```r
# 推定回数を指定
Iter <- 50   # 回数を指定する

# 推移の確認用の受け皿を用意
trace_theta <- array(0, dim = c(D, K, Iter + 1),
                     dimnames = list(paste0("d=", 1:D),
                                     paste0("k=", 1:K),
                                     paste0("Est", 1:(Iter + 1))))
trace_phi   <- array(0, dim = c(K, V, Iter + 1),
                     dimnames = list(paste0("k=", 1:K),
                                     paste0("v=", 1:V),
                                     paste0("Est", 1:(Iter + 1))))

# 初期値を代入
```

```
trace_theta[, , 1] <- theta_dk
trace_phi[, , 1]   <- phi_kv

# 推定
for(i in 1:Iter) {

  # パラメータを初期化
  next_theta_dk <- matrix(0, nrow = D, ncol = K,
                          dimnames = list(paste0("d=", 1:D),  # 行名
                                          paste0("k=", 1:K))) # 列名

  next_phi_kv <- matrix(0, nrow = K, ncol = V,
                        dimnames = list(paste0("k=", 1:K),  # 行名
                                        paste0("v=", 1:V))) # 列名

  for(d in 1:D) { ## (各文書)

    # 負担率を計算
    tmp_q_vk <- t(theta_dk[d, ] * t(t(phi_kv) ^ N_dv[d, ])) ## V*KdimMat <- t(KdimVec * t(V*KdimMat ^ V
    q_dvk[d, , ] <- tmp_q_vk / apply(tmp_q_vk, 1, sum)      ## V*KdimMat <- V*KdimMat / VdimVec

    # theta_dk を更新
    next_theta_dk[d, ] <- apply(q_dvk[d, , ] * N_dv[d, ], 2, sum) ## KdimVec <- apply(V*KdimMat * VdimV

    # phi_kv を更新
    next_phi_kv <- next_phi_kv + t(q_dvk[d, , ] * N_dv[d, ]) ## K*VdimMat <- K*VdimMat + t(V*KdimMat *

  } ## (/各文書)

  # パラメータを正規化して更新
  theta_dk <- next_theta_dk / apply(next_theta_dk, 1, sum)
  phi_kv   <- next_phi_kv / apply(next_phi_kv, 1, sum)

  # 推移の確認用
  trace_theta[, , i + 1] <- theta_dk
  trace_phi[, , i + 1]   <- phi_kv
}
```

・コードの解説

文書ごとにパラメータ推定を行っていきます。

R ではベクトルとマトリクスとを計算するとき、ベクトルの各要素をマトリクスの 1 行 1 列目の要素から列方向に順番に対応させて計算していきます。つまり、ベクトルの要素の数とマトリクスの列の要素の数 (行数) を一致させると、ベクトルの 1 つ目の要素をマトリクスの 1 行目の各要素に対応させることができます。なので、適宜転置して計算していきます。

・負担率の更新

```
# 負担率を計算
tmp_q_vk <- t(theta_dk[d, ] * t(t(phi_kv) ^ N_dv[d, ])) ## V*KdimMat <- t(KdimVec * t(V*KdimMat ^ VdimV
q_dvk[d, , ] <- tmp_q_vk / apply(tmp_q_vk, 1, sum)      ## V*KdimMat <- V*KdimMat / VdimVec
```

負担率の計算式は

$$q_{dvk} = \frac{\theta_{dk}\phi_{kv}^{N_{dv}}}{\sum_{k'=1}^{K}\theta_{dk'}\phi_{k'v}^{N_{dv}}} \tag{4.1}$$

なので、基本的な計算は $\theta_{dk} * \phi_{kv}^{N_{dv}}$ になります。

$\phi_{kv}^{N_{dv}}$ の計算は、phi_kv を転置して 1 から V までを列とし V 次元ベクトル N_dv[d,] と対応させて行います。
それを更に、K 次元ベクトル theta_dk[d,] と掛けるために、再度転置して行数を K に戻して計算します。
最終的に計算結果を q_dvk[d, ,] に代入するため、最後にもう一度転置して V 行 K 列のマトリクスとして tmp_q_vk に代入します。

続いて正規化の計算を行います。分母の $\sum_{k'=1}^{K}$ の計算は、tmp_q_vk を apply() で行方向に (1 から K まで)sum() で行います。

この処理を D 回繰り返すことで、q_dvk の全ての要素を計算できます。

・トピック分布の更新

```
# theta_dk を更新
next_theta_dk[d, ] <- apply(q_dvk[d, , ] * N_dv[d, ], 2, sum) ## KdimVec <- apply(V*KdimMat * VdimVec,
```

Θ の各要素の計算式は

$$\theta_{dk} = \frac{\sum_{v=1}^{V} q_{dvk}N_{dv}}{\sum_{k'=1}^{K}\sum_{v=1}^{V} q_{dvk'}N_{dv}} \tag{4.2}$$

なので、基本的な計算は $q_{dvk} * N_{dv}$ になります。

ここでは分子の計算のみを行い、next_theta_dk としておきます。

q_dvk[d, ,] は V 行 K 列のマトリクスで、N_dv[d,] は V 次元ベクトルなので、そのまま q_dvk[d, ,] * N_dv[d,] で計算します。
その計算結果を apply(., 2, sum) で列方向に合計することで、$\sum_{v=1}^{V}$ の計算を行います。

この処理を D 回繰り返すことで、next_theta_dk に 1 行目から D 行目まで順番に代入していきます。

全ての計算が行われた後で、正規化処理 (1 から K まで (各列ごとに) 和をとったもので割る) を行い theta_dk を更新します。

・単語分布の更新

```
# phi_kv を更新
next_phi_kv <- next_phi_kv + t(q_dvk[d, , ] * N_dv[d, ]) ## K*VdimMat <- K*VdimMat + t(V*KdimMat * Vdim
```

Φ の各要素の計算式は

$$\phi_{kv} = \frac{\sum_{d=1}^{D} q_{dvk}N_{dv}}{\sum_{v'=1}^{V}\sum_{d=1}^{D} q_{dv'k}N_{dv}} \tag{4.3}$$

なので、基本的な計算は $q_{dvk} * N_{dv}$ になります。

ここでは分子の計算のみを行い、next_phi_kv としておきます。

q_dvk[d, ,] は V 行 K 列のマトリクスで、N_dv[d,] は V 次元ベクトルなので、そのまま q_dvk[d, ,] * N_dv[d,] で計算します。
次に next_phi_kv と足すために、計算結果を転置して K 行 V 列のマトリクスとして計算します。

$\sum_{d=1}^{D}$ の計算は、ループの中で N 回繰り返して足していくことで行われます。

全ての計算が行われた後で、正規化処理 (1 から V まで (各列ごとに) 和をとったもので割る) を行い phi_kv を更新します。

・推定結果の確認

・作図用関数の作成

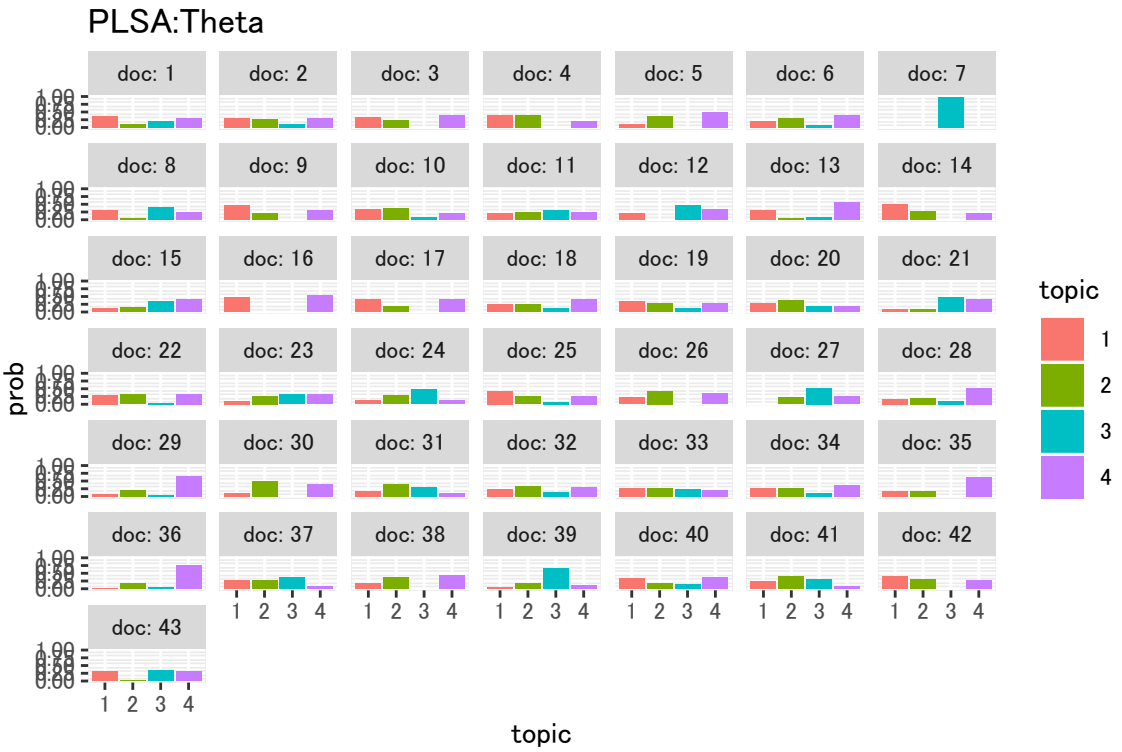
```
### トピック分布
fn_plotTheta <- function(theta_dk){

  # データフレームを作成
  theta_WideDF <- cbind(as.data.frame(theta_dk),
                        doc = as.factor(1:D))

  # データフレームを long 型に変換
  theta_LongDF <- pivot_longer(
    theta_WideDF,
    cols = -doc,           # 変換せずにそのまま残す現列名
    names_to = "topic",    # 現列名を格納する新しい列の名前
    names_prefix = "k=",   # 現列名から取り除く文字列
    names_ptypes = list(topic = factor()),  # 現列名を要素とする際の型
    values_to = "prob"     # 現要素を格納する新しい列の名前
  )

  # 描画
  ggplot(data = theta_LongDF, mapping = aes(x = topic, y = prob, fill = topic)) +
    geom_bar(stat = "identity", position = "dodge") +  # 棒グラフ
    facet_wrap( ~ doc, labeller = label_both) +        # グラフの分割
    labs(title = "PLSA:Theta") # タイトル
}


### 単語分布
fn_plotPhi <- function(phi_kv){

  # データフレームを作成
  phi_WideDF <- cbind(as.data.frame(phi_kv),
                      topic = as.factor(1:K))

  # データフレームを long 型に変換
  phi_LongDF <- pivot_longer(
    phi_WideDF,
    cols = -topic,         # 変換せずにそのまま残す現列名
```

```
    names_to = "word",     # 現列名を格納する新しい列の名前
    names_prefix = "v=",   # 現列名から取り除く文字列
    names_ptypes = list(word = factor()),  # 現列名を要素とする際の型
    values_to = "prob"     # 現要素を格納する新しい列の名前
  )

  # 描画
  ggplot(data = phi_LongDF, mapping = aes(x = word, y = prob, fill = word)) +
    geom_bar(stat = "identity", position = "dodge") +  # 棒グラフ
    facet_wrap( ~ topic, labeller = label_both) +      # グラフの分割
    scale_x_discrete(breaks = seq(1, V, by = 10)) +    # x軸目盛
    theme(legend.position = "none") +                  # 凡例
    labs(title = "PLSA:Phi") # タイトル
}


### トピック分布の推移
fn_plotTraceTheta <- function(trace_theta, DocNum = 1){

  if(DocNum > D){
    return("ERROR:DocNum > D")
  }

  # 文書番号を指定
  DocNum <- DocNum

  # データフレームを作成
  trace_theta_WideDF <- cbind(as.data.frame(trace_theta[DocNum, , ]),
                              topic = as.factor(1:K))

  # データフレームを long 型に変換
  trace_theta_LongDF <- pivot_longer(
    trace_theta_WideDF,
    cols = -topic,           # 変換せずにそのまま残す現列名
    names_to = "Iteration",  # 現列名を格納する新しい列の名前
    names_prefix = "Est",    # 現列名から取り除く文字列
    names_ptypes = list(Iteration = numeric()),  # 現列名を要素とする際の型
    values_to = "prob"       # 現要素を格納する新しい列の名前
  )

  # 描画
  ggplot(data = trace_theta_LongDF, mapping = aes(x = Iteration, y = prob, color = topic)) +
    geom_line() +  # 折れ線グラフ
    labs(title = "PLSA:theta",
         subtitle = paste0("d=", DocNum)) # タイトル
}


## 単語分布の推移
fn_plotTracePhi <- function(trace_phi, TopicNum = 1){

  if(TopicNum > K){
    return("ERROR:TopicNum > K")
```

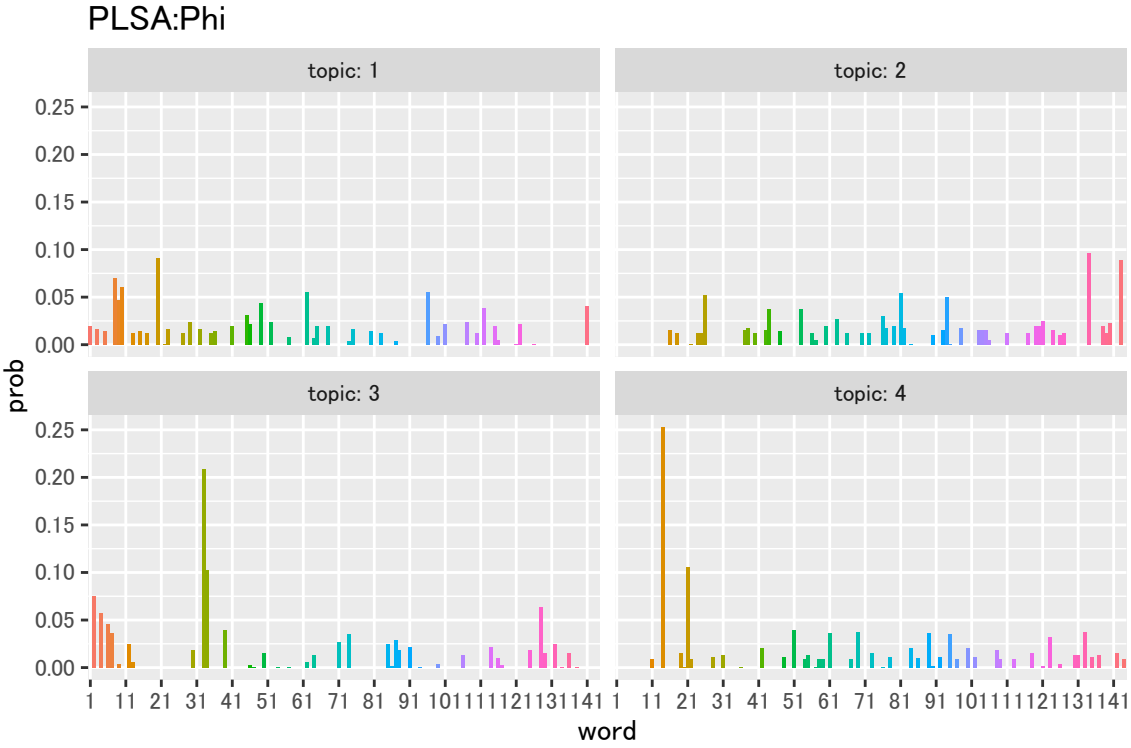
```
  }

  # トピック番号を指定
  TopicNum <- TopicNum

  # データフレームを作成
  trace_phi_WideDF <- cbind(as.data.frame(trace_phi[TopicNum, , ]),
                            word = as.factor(1:V))

  # データフレームを long 型に変換
  trace_phi_LongDF <- pivot_longer(
    trace_phi_WideDF,
    cols = -word,           # 変換せずにそのまま残す現列名
    names_to = "Iteration", # 現列名を格納する新しい列の名前
    names_prefix = "Est",   # 現列名から取り除く文字列
    names_ptypes = list(Iteration = numeric()),  # 現列名を要素とする際の型
    values_to = "prob"      # 現要素を格納する新しい列の名前
  )

  # 描画
  ggplot(data = trace_phi_LongDF, mapping = aes(x = Iteration, y = prob, color = word)) +
    geom_line(alpha = 0.5) +          # 折れ線グラフ
    theme(legend.position = "none") + # 凡例
    labs(title = "PLSA:phi",
         subtitle = paste0("k=", TopicNum)) # タイトル
}
```

・描画

```
# トピック分布
fn_plotTheta(theta_dk)
```

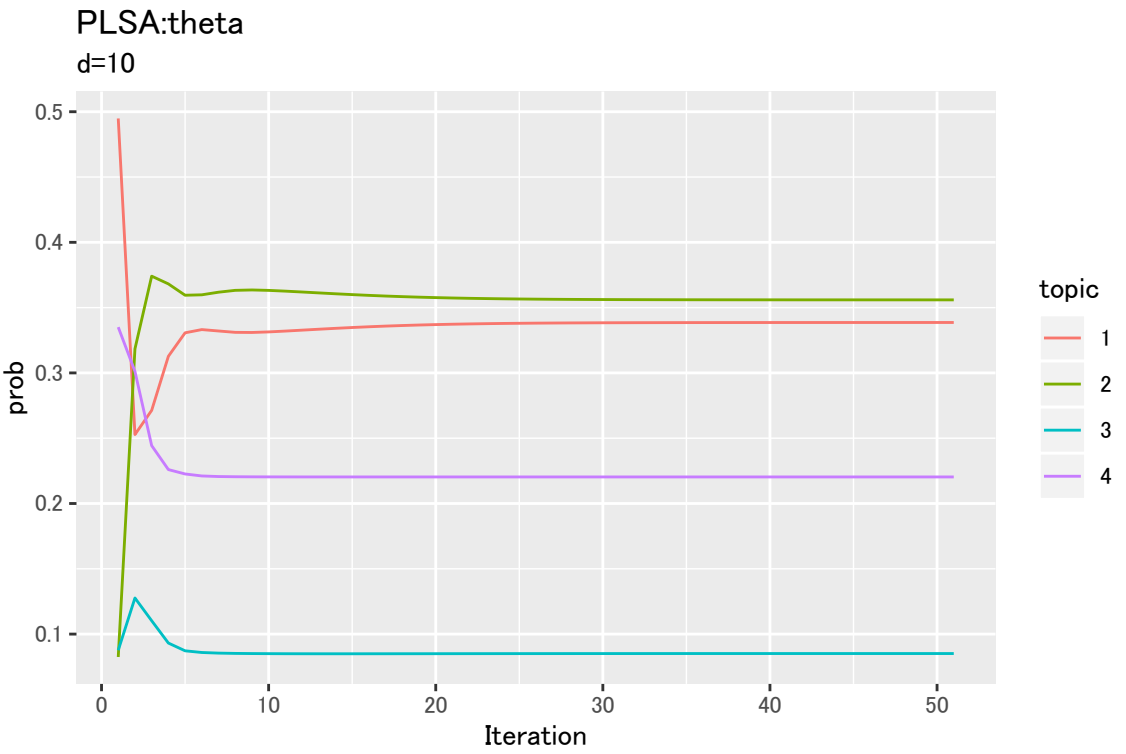



```
# 単語分布
fn_plotPhi(phi_kv)
```




```
# トピック分布の推移
```

```
fn_plotTraceTheta(trace_theta, DocNum = 10)
```




```
# 単語分布の推移
fn_plotTracePhi(trace_phi, TopicNum = 4)
```