データ科学特論

その 9: 潜在意味分析

1　今回行う分析

1. 好みの異性の性格の分析

今回は，入力として文書（文字列）の集合を考え，文書間の類似性や，語句の類似性を分析する。どちらかというと意味検索などの検索技術の一分野であるが，主成分分析などの統計の考え方も用いる.

2　意味とは

「病院」「医院」「クリニック」といった単語は，同じような意味を持つと考えられる。このように，単語間どうしの意味が似ているかどうか，ということを定量的に表すにはどうすれば良いか。

まず，それぞれの単語をベクトルで表すことにする。

そして，ベクトル間の角度が小さければ，意味が似ている，大きければ似ていないと考える。これには，角度が小さければ 1 に近づき，π に近づけば 0 に近づく，cos で表せば良い。ベクトル間の cos は一般には，

$$cos(a,b) = \frac{\sum a_i b_i}{||a|| \cdot ||b||}$$

で表される[1].

では，単語のベクトルをどのように作ればいいのだろうか。以下では，たくさんの文書を集め，その一つ一つを次元（つまり直交基底）としたベクトルとする。つまり，

「同じ文書に現れる（**共起する**）語句どうしは似ている。これをまとめていくと，似たような文書に現れる語句どうしは似ている」

[1]前回の通り，$||w||$ は w の長さ（ノルム）のことであり，$\sqrt{w \cdot w}$ で定義される。

という考え方を定量化した物となる。これを語句・共起行列として後述する。

ただしこれだけでは，語句ベクトルの次元が大きすぎて，なかなか意味をとらえることができない。そこで，特異値分解という手法を使って，語句間の違いを表す成分 (正確には主成分に相当する。) に変換しながら次元を削減して，意味をとらえやすくする。

以下ではこれらの詳細を述べる。

2.1 語句・文書共起行列

多くの文書があるとき，これらから語句の共起を以下のような行列で表す。

全ての文書に現れる語句を行に並べ，全ての文書を列に並べる。この行列の i, j 要素は，

「i 番目の語句が j 番目の文書に（どれだけ）あるかどうか」

を表す．以下ではこの行列を，**語句・文書行列** $A_{m\times n}$ と呼ぶ。m は語句数，n は文書数である。

より正確には，$A_{m\times n}$ の i, j 要素 $a_{i,j}$ は，次の式で表される。

$$a_{i,j} = L(i,j) \times G(i)$$

- $L(i,j)$: 語句と文書を入力とする局所的重み関数
- $G(i)$: 語句を入力とする大局的重み関数.

局所的重み関数

局所的重み関数 $L(i,j)$ には，次のような種類がある。

- $L(i,j) = tf(i,j)$: 文書 j における語句 i の出現頻度 $tf(i,j)$.
- $L(i,j) = log(tf(i,j)+1)$: 出現頻度に 1 を足して対数を取ったもの．大規模なデータに用いられる。1 を足すのは対数値を正にするため。
- $L(i,j) = \begin{cases} 1 & if\ tf(i,j) > 0, \\ 0 & if\ tf(i,j) = 0, \end{cases}$: 単純に語句が文書に出現するかどうかを表すもの.

大局的重み関数

大局的重み関数では，各語句が文書間にわたる影響を調整する。出現頻度がどの文書においても高くなる語句，つまり一般的な語句の影響を弱めることが行われる。

- $G(i) = \frac{1}{\sqrt{\sum_j L(i,j)^2}}$: ベクトル $(L(i,1), L(i,2)\cdots)$ の長さの逆数をとった，**標準化**と呼ばれる重み関数.

- $G(i) = \frac{gf(i)}{df(i)}$: 語句 i の全体での出現頻度 $gf(i)$ と，その語句が含まれる文書の数 $df(i)$ の比．語句が珍しい文書数で現れるほど高い値となる重み関数.

- $G(i) = 1 + \frac{\sum_j p(i,j)\log p(i,j)}{\log n}$: エントロピーと呼ばれる。$p(i,j) = \frac{tf(i,j)}{gf(i)}$ は，語句 i が出現した元での文書 j の条件付き確率．n は文書数。

- $G(i) = 1 + \log\frac{n}{df(i)}$: これもよく用いられる．$L(i,j) = tf(i,j)$ の時に組み合わせて tf-idf と言われる。

2.2 特異値分解

語句・文書行列 $A_{m\times n}$ は，以下の形に分解することが出来る。

$$A_{m\times n} = T_{m\times k} \times S_{k\times k} \times D_{k\times n}$$

この分解を**特異値分解**（一般には実数または複素数を成分とする行列に対して可能である。）と呼び，以下のような特徴がある。

- $T_{m\times k}$: 正規直交行列[2]となり，行数は m つまり語句数に対応する．$T_{m\times k}$ の i,j 要素を $語_{i,j}$ と書くことにする。

- $S_{k\times k}$: 対角行列であり，k は $min(m,n)$ とする．対角成分 $s_1\cdots s_k$ は**特異値**と呼ばれ，$s_1 \geq s_2 \geq \cdots \geq s_k \geq 0$ となる.

- $D_{k\times n}$: 転置した $D^t_{k\times n}$ は正規直交行列となり，$D_{k\times n}$ の列数は n つまり文書数に対応する。$D_{k\times n}$ の i,j 要素を $文_{i,j}$ と書くことにする。

この分解の意味を考えてみよう。行列の積の定義は

$$M_{m\times k} \times Y_{k\times n} = \left(\sum_{l=1}^{k} x_{i,l}y_{l,j}\right)$$

であるから，分解の右辺は

$$T_{m\times k} \times S_{k\times k} \times D_{k\times n} = \left(\sum_{l}^{k} 語_{i,l}s_{l,j}\right) \times D_{k\times n}$$

[2]正規直交行列 M とは，各行をベクトルと見なしたときにその大きさ (ノルム) が 1 で，どの 2 つも互いに直交するような行列．直交は $MM^t = M^t = I$（つまり単位行列）で表される。

$$= \left(\sum_{l}^{k} \sum_{h}^{k} 語_{i,l} s_{l,h} 文_{h,j} \right)$$

ただし $S_{k \times k}$ は対角行列なので $l = h$ 以外は $s_{l,h} = 0$ となり，$S_{k \times k}$ の対角要素に対応する項のみが残るので，

$$= \left(\sum_{l}^{k} 語_{i,l} s_{l} 文_{l,j} \right)$$

$$= 語_{i,1} s_1 文_{1,j} + 語_{i,2} s_2 文_{2,j} + \cdots 語_{i,k} s_k 文_{k,j}$$

となる．つまり，$A_{m \times n}$ の i, j 要素は，$T_{m \times k}$ の i 行目と $D_{k \times n}$ の j 列目を取ってきて，それぞれの l 番目の列（行）に s_l をかけたものの和である．

ここで，$T_{m \times k}$ の行数は語句数に対応し，$D_{k \times n}$ の列数が文書数に対応し，$s_1 \geq s_2 \geq \cdots \geq s_k$ であることを考えると，以下のようにとらえることが出来る。

- 特異値分解により，$T_{m \times k}$ の要素は，($T_{m \times k}$の要素 $\times$ $D_{k \times n}$の要素) の重み付き和として表現できる。その重みは，$S_{k \times k}$ の対角要素によって大きい順に与えられる。
- $T_{m \times k}$ の各行は語句の特徴を表し，各行ベクトルは，その影響の大きい順に要素が並んでいる。この意味で，$T_{m \times k}$ の各列を**左特異ベクトル**と呼ぶ。
- $D_{k \times n}$ の各列は文書の特徴を表している。各列ベクトルは，その影響の大きい順に要素が並んでいる。この意味で，$D_{k \times n}$ の各行を**右特異ベクトル**と呼ぶ。

具体的に特異値分解をどう計算するかはここでは述べないが，近年大規模な行列に対しても高速に計算できる手法が開発されている。

さて，このような特異値分解の結果の一部を用いて，行列 $S_{k \times k}$ の k' 番目 $(k' < k)$ までの特異値を用いた

$$\hat{A}_{m \times n} = T_{m \times k'} \times S_{k' \times k'} \times D_{k' \times n}$$

を考えてみよう。特異値 s_l は大きい順に並ぶので，$\hat{A}_{m \times n}$ は，影響が大きい左 (右）特異ベクトルを用いた $A_{m \times n}$ の近似と見なせる。この $\hat{A}_{m \times n}$ で表される空間を，**潜在意味空間**と呼ぶ。

2.3 分析

上記のような $\hat{A}_{m \times n}$ を用いて分析を行う際，次のような分析が考えられる。

1. 文書間の類似性

2. 語句間の類似性

3. 語句と文書の間の類似性

4. 新たな検索質問文と文書間の類似性

1 については，$\hat{A}_{m\times n}$ の各行間の類似度を調べれば良いが，本質的にはこの情報は $D_{k'\times n}$ が持っているので，$D_{k'\times n}$ の各列間の cos を計算すれば良い．2 については $T_{m\times k'}$ の行間となる．

3 については，$\hat{A}_{m\times n}$ における各要素が類似性を表していると考えることができる。

4 については，基本的には 1 と同じように比較したいのだが，検索質問文を，潜在意味空間上のベクトルとして表現する必要がある。そのために少し変形を行っておく。

$$A_{m\times n} = T_{m\times k}S_{k\times k}D^t_{k\times n}$$

を転置して，

$$A^t_{m\times n} = D_{k\times n}S_{k\times k}T^t_{m\times k}$$

両辺に右から $T_{m\times k}$ をかけて ($T^t_{m\times k}T_{m\times k} = I$ なので)，

$$A^t_{m\times n}T_{m\times k} = D_{k\times n}S_{k\times k}$$

$$\Longrightarrow D_{k\times n} = A^t_{m\times n}T_{m\times k}S^{-1}_{k\times k}$$

検索質問文をベクトル q としておくと，これはもともとの語句・文書ベクトル $A_{m\times n}$ に対応しており，潜在意味空間上の $\hat{q}$ は $D_{k\times n}$ に対応するべきだと考えると，q の k' 次元潜在意味空間上での表現は，

$$\hat{q} = q^t T_{m\times k'}S^{-1}_{k'\times k'}$$

と見なすことができる。このように変換をおこなってから，q と $D_{k'\times n}$ の類似性を見れば良い。

このような考え方は，後で文書を追加するときにも使うことが出来，行列を全て計算し直す必要はない。

3　RでLSA

R潜在意味分析を行うには，パッケージRMeCabを使うが，これは外部のmecabというプログラムを呼び出す。このどちらも事前に別途インストールしておく必要がある[3].

ライブラリ・プログラム読み込み

まずは，RMeCabライブラリと，教科書で提供されたプログラムLSA.txtを読み込む[4]. LSA.txtでは，後に使うKyoki(),dimReducShare(),dimReducNdocs(), dimReducKaiser(), dimReducFrac(), myQuery, myCosine()関数が定義されている。

```
> library(RMeCab)
> source("chap9/LSA.txt")
```

語句・文書行列の生成

```
> docterm <- docMatrix("chap9/z")
file = chap9/z/doc01.txt
file = chap9/z/doc02.txt
file = chap9/z/doc03.txt

...

file = chap9/z/doc32.txt
file = chap9/z/doc33.txt
Term Document Matrix includes 2 information rows!
whose names are [[LESS-THAN-1]] and [[TOTAL-TOKENS]]
if you remove these rows, run
result[ row.names(result) !=  "[[LESS-THAN-1]]" , ]
result[ row.names(result) !=  "[[TOTAL-TOKENS]]" , ]
```

末尾に表示されているように，最初の2行はデータに関する情報が書かれている。確かめてみる。

```
> head(docterm)
                  docs
terms              doc01.txt doc02.txt doc03.txt doc04.txt
```

[3]RMeCabについては http://rmecab.jp/wiki/ , mecabについては http://mecab.sourceforge.net/からダウンロード．私のMac OS Snow Leopardでは32bitコンパイルのために工夫が必要でした。
[4]LSA.txtおよび後述のzフォルダ以下は，Mac上ではUTF8文字コードに変換しておいた.

```
  [[LESS-THAN-1]]          0          0          0          0
  [[TOTAL-TOKENS]]        19         14         38         46
  観察                     1          0          0          0
  距離                     1          0          0          0
  周り                     1          0          0          0
  素直                     1          0          0          0
...
```

そこで以下のようにして 1，2 行目を削除する。

```
> docterm <-docterm[-(1:2),]
```

複数の文書に現れる語句のみを抽出

```
> docterm2 <- Kyoki(docterm, minDocFreq=2)
```

引数 minDocFreq で語句が現れる最小の文書数を指定する。

特異値分解

```
> svd.docterm<-svd(docterm2)
```

意味空間の次元縮小

特異値分解した結果の，特異値が大きい方から順にある基準までを採用して次元縮小する．ここでは LSA.txt で定義された dimReducShare() 関数を使う。これは特異値の大きい方からの累積和が合計に対して share になるまでの特異値を採用する。

```
> rslt <- dimReducShare(svd.docterm, share=0.5, docterm=docterm2)
```

他にも，

- 指定された ndocs の数を特異値の和が最初に超えた時点までを採用する dimReducNdocs(ndocs)
- 次元数を元の frac 割に変換する dimReducFrac(frac)
- 特異値が 1 以上なら採用する dimReducKaiser()

が LSA.txt に用意されている.
削減された結果を見ると,

```
> str(rslt)
List of 3
 $ tk: num [1:63, 1:10] 0.0478 0.0737 0.0567 0.1087 0.0265 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:63] "観察" "周り" "素直" "良い" ...
  .. ..$ : NULL
 $ dk: num [1:33, 1:10] 0.0426 0.0517 0.3699 0.329 0.2259 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:33] "doc01.txt" "doc02.txt" "doc03.txt" "doc04.txt" ...
  .. ..$ : NULL
 $ sk: num [1:10] 6.74 4.94 4.62 4.4 3.92 ...
```

10 次元で表されている。元は 33 人分の文書を 10 次元に近似していることになる。tk が $T_{m\times k'}$ に，dk が $D_{k'\times n}$ に，sk が $S_{k'\times k'}$ に対応する。

類似性の計算

myCosine() では引数に与える行列の全ての列に関して cos を計算する。

```
> myCosine(a=t(rslt$dk))
             doc01.txt   doc02.txt     doc03.txt    doc04.txt
doc01.txt  1.000000000 -0.18331428 -0.1021110442 -0.027724107
doc02.txt -0.183314283  1.00000000 -0.0578080215  0.184043512
doc03.txt -0.102111044 -0.05780802  1.0000000000  0.681758975
doc04.txt -0.027724107  0.18404351  0.6817589745  1.000000000
doc05.txt -0.052545734  0.04494544  0.0002858855 -0.039381050
...
```

検索質問文のベクトル化

RMeCabC() を使って検索質問文をベクトル化する.

```
> unmei <- RMeCabC("社交的で優しいし，俺のことを優先してくれるけど，以外とクールな面もある人")
> tmp <- unlist(unmei)
> lst <- names(tmp) %in% c("名詞", "動詞", "形容詞")
> unmei <- tmp[lst]
> unmei
    名詞       名詞    形容詞       名詞       名詞       名詞       動詞
  "社交"       "的" "優しい"       "俺"     "こと"     "優先"       "し"
    動詞       名詞       名詞       名詞       動詞       名詞
"くれる"     "以外" "クール"       "面"     "ある"       "人"
```

unlist() はリストをベクトルに変換する関数である。

%in%は，右側のベクトルの中に左側のベクトルの一つ一つが存在するかを logical 型で返す。ここでは unmei を unlist() した後の tmp の名前 names() が，名詞，動詞，形容詞のいずれかに入るかを返している。さらに次の行で，tmp[lst] としているので，結局，名詞，動詞，形容詞のいずれかを名前に持つ要素のみが抽出される。

検索と，潜在意味空間に存在しない語の除外

```
> unmei.q <- myQuery(unmei, rownames(rslt$tk))
4 番目の俺 が term.list に存在しません
6 番目の優先 が term.list に存在しません
7 番目のし が term.list に存在しません
8 番目のくれる が term.list に存在しません
9 番目の以外 が term.list に存在しません
11 番目の面 が term.list に存在しません
12 番目のある が term.list に存在しません
 以下にエラー myQuery(unmei, rownames(rslt$tk)) :
```

これらの語が $T_{m\times k'}$ に存在しないので，検索質問文 unmei から除外する。

```
> unmei2 <- unmei[-c(4,6,7,8,9,11,12)]
```

再度検索

```
> unmei.q <- myQuery(unmei2, rownames(rslt$tk))
```

潜在意味空間上に検索質問文ベクトルを変換

```
> unmei.vec <- t(unmei.q) %*% rslt$tk %*% solve(diag(rslt$sk))
> unmei.vec <- as.vector(unmei.vec)
```

diag() は対角行列を作る関数，solve(a, b) は，a %*% $x = b$ という方程式を x について解く関数である。ここでは b は省略されていて，単位行列と見なされる。つまり逆行列を求めている。%*%は行列の積だったことを思い出そう。

検索質問文との類似性

検索質問ベクトル unmei.vec と潜在意味空間上の文書 result$dk との類似性を計算。

```
> unmei.cos <- myCosine(a=t(rslt$dk), b=unmei.vec)
> names(unmei.cos) <- as.character(1:33)
> round(sort(unmei.cos, decreasing=T),3)[1:5]
   3     22    33     28    25
0.772 0.588 0.584  0.394 0.393
```

2 行目では 1〜33 を `unmei.cos` の名前とし，3 行目では降順に並べ替えて小数点第 3 位で丸め，5 位までを表示している。

これで，33,3,2,4,24 番目の文書（人）がその順に検索質問文に近いことになった。

4　宿題

今回 R で行った内容を，自分の計算機環境で再現してください。