JAPLA 研究会資料 2016/1/16

固有値計算の Power 法と Deflation 法の J プログラム
－主成分分析からの理工学とは違った固有値の理解－

西川　利男

はじめに

　先月、鳥邊氏から「コレスポンデンス分析のプログラムをスマホのアプリへ」という発表があった。私にとっては、「コレスポンデンス分析」とは初めてのことばであり、質問したところ、「主成分分析」と似ているが、少しちがう、とのことであった。実は、主成分分析の語は知ってはいるが、私自身これまで実際に使ったことはなかった。

　鈴木義一郎氏の著書「J 言語による統計解析」[1]に「主成分分析」なる一章があり、あらためて読み返した。これに関連して、基礎となる「固有値、固有ベクトル」について、理工学と主成分分析など経済数学での理解のしかたが、かなり違うのではないか、と思った。

　同時に線形数学の基礎理論としてではなく、もっと身近な計算の道具(computation literacy)としての「固有値、固有ベクトル」への理解に対して私なりの考えについて多少述べたいとおもう。

[1] 鈴木義一郎「J 言語による統計解析」p.119-135 森北出版 (1996).

１．理工学と経済数学における固有値、固有ベクトルの理解の違い

　これまで、私は理工学に携わる一人として、固有値、固有ベクトルに対して次のように接してきた。つまり、固有振動を頭に描いて、振動や量子力学における微分方程式を解く道具として使ってきた。

　しかし、主成分分析に代表される経済数学では、考え方はこれとは違うように見える。統計分析では、多数のデータの集まりから、将来の予想を見通すことが求められる。

　これらを箇条書きで対比してみよう。

・理工学……たとえば原子配置の力の場において、安定した振動＝固有振動は何か、これは連立微分方程式を解く。……このための固有値、固有ベクトル

・主成分分析…多数のデータのさまざまな性質の違いをもとに、第一に主要な性質、傾向は何なのだろうか、その次ぎに主要なものは、……という指標をうるための道具としての、固有値、固有ベクトル

　ところで、私はこれまで、数学の行列(matrix、ちなみに英語の本来の意味には、タテヨコという考えはない)とは、一種の状態を示すベクトルという名詞に対して、行列演算(とくに記号は入れないが)を行う動詞(ちょうど平方根や三角関数のように)という見方をしてきた。関数型言語の J でも、そう考えてきた。しかし見方を変えて、行列は名詞と見て、左引数にこの行列をとり、右引数にはベクトルをとる両側動詞（たとえば内積(inner product)など）と見て、プログラミングしたほうが良いかもしれない。

　さらに、経済数学で扱う行列はデータがタテヨコに並んだ名詞であり、行列というより配列(array)である。固有値のいろいろな計算では、その要素の間で行う演算であるから配列演算といったほうがよいのではないか。

　いずれにせよ、J は固有値、固有ベクトルの計算において、極めて効率的にそのプログラムをこなしてくれるまたとない環境である。

２．固有値、固有ベクトルについて－計算法などいろいろな文献

固有値、固有ベクトルの計算法はそれだけで、数値計算の一つの分野をなすもので、いろいろなアルゴリズムがあるが、主なものは次のようである。

Power 法（繰り返し、収束法）＋ Deflation 法（減次元法、Wielandt/Hotelling）
Jacobi 法
QR 分解法
LF 法（Le Verier & Faddeev）

これまで、Ｊの分野では、志村正人、中野嘉弘両氏の詳細な解説の文献がある。

[2] 志村正人「マトリックスの数学と数値計算(I)」行列式と行列の固有値
JAPLA 2010/8/3
[3] 中野嘉弘「固有ベクトル計算法」JAPLA 2010/9/25, JAPLA 2010/10/23

いま、私の手元には次のような本がある。

[4] 戸田英雄、小野令美「入門数値計算」オーム社(1983).
[5] 洲之内治男、寺田文行、四条忠雄「FORTRAN による演習数値計算」サイエンス社 (1980).
[6] Brian Bradie, "A Friendly Introduction to Numerical Analysis", Pearson(2006).
[7] Bill Jacob, "Linear Functions and Matrix Theory", Springer-Verlag(1995).
[8] G.ストラング、山口昌也監訳、井上昭訳「線形代数とその応用」産業図書(1986).

ここでも、理工学と経済数学とで好みがあるように思われる。私の感じたところでは、すべての固有値を求める理工学ではJacobi 法が、主要なものから求める主成分分析では、Power 法を説明している。

３．固有値、固有ベクトルの位置づけ

いまでは、ちょうど平方根が√ ルートキーを押すだけで得られるように、固有値、固有ベクトルもその値をどうやって計算するかなどには気を使うことなく、有効に使えばよい、という時代になったのだろう。その代わりに、どう意味を持つかを真に理解することが大切であろうと思う。

とくにわれわれＪユーザにとっては、固有値の計算に対して 128!:0（QR 分解）という強力な武器が与えられているのである。

４．固有値、固有ベクトル計算のためのPower 法とDeflation 法

私自身、Jacobi 法のプログラムは経験があるが、Power 法とDeflation 法のプログラミングは実は初めてである。手法は、上の文献[4], [5], [6]によった。

Power 法とDeflation 法はまさに主成分分析のためにあるといえる。
Power 法は繰り返し、収束法であり、絶対値最大の固有値を１つ求めるだけである。そしてそのアルゴリズムは、一回の計算では近似値が得られるだけである。しかし、その近似値をもとに計算を同じ計算アルゴリズムを繰り返す（Power）ことによって固有値と固有ベクトルを得る計算法である。固有値は１つしか得られない。

つぎにDeflation 法とは固有値を求めるのではなく、さきの固有値と行列を元に、行列（配列）の次元を１つ下げるアルゴリズムである。そして、これには非対称行列でもよ

いWielandt法と対称行列に限るHotelling法とがある。なお、Deflationという語はある要素を取り除き、行列の次数を下げるという意味である。

４．１　Power法のアルゴリズムとＪプログラムの実行

アルゴリズムは次のようである。文献[5]，p.171，　文献[4]，p.60

$$x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$y_i = \frac{1}{l^{(i)}} x_i\,,\quad l^{(i)} = \max(x_i)$$

$$x_{i+1} = A\,y_i$$

$$r_{i+1} = \frac{(y_i\ ip\ \ x_{i+1})}{(y_i\, ip\ y_i)}\,,\quad ip = inner\ product$$

を繰り返していくと、

$l^{(i)} \to \lambda_1, r^{(i+1)} \to \lambda_1$ 固有値, $y_i \to \lambda_1$ に対する固有ベクトル

として、固有値、固有ベクトルが求められる。

つぎのように、数値計算例によって示す。すべてのＪプログラムは巻末に示す。

文献[4]，問題4.3(p.65)，問題解答(p.160-161)

```
   DS43 =: 4 4$_2 1 _3 2 1 1 1 0 4 _2 5 _3 5 _1 5 6
   DS43
_2  1 _3  2
 1  1  1  0
 4 _2  5 _3
 5 _1  5  6
```

計算途中を表示しながら実行する。

```
   DS43 pow0 ^:(i.14) 1 1 1 1;1
x: 1 1 1 1
l: 1
y: 1 1 1 1
x: _2 3 4 15
r: 5
--------------------
x: _2 3 4 15
l: 15
y: _0.13333 0.2 0.26667 1
x: 1.6667 0.33333 _2.6 6.4667
r: 4.9764
--------------------
x: 1.6667 0.33333 _2.6 6.4667
l: 6.4667
y: 0.25773 0.051546 _0.40206 1
```

```
x： 2.7423 _0.092784 _4.0825 5.2268
r： 6.1509
-------------------
```

（途中省略）

```
------------------
```

（以下まとめたもの）

```
+--------------------------------+------+------+
|1 1 1 1                         |1     |      |
+--------------------------------+------+------+
|_2 3 4 15                       |5     |1     |
+--------------------------------+------+------+
|1.6667 0.33333 _2.6 6.4667      |4.9764|15    |
+--------------------------------+------+------+
|2.7423 _0.092784 _4.0825 5.2268|6.1509|6.4667|
+--------------------------------+------+------+
|3.2761 _0.27416 _4.7712 4.7357 |5.4019|5.2268|
+--------------------------------+------+------+
|3.5544 _0.37081 _5.1162 4.4461 |4.8745|4.7712|
+--------------------------------+------+------+
|3.2761 _0.37775 _4.6832 3.7603 |4.5715|5.1162|
+--------------------------------+------+------+
|3.1261 _0.38111 _4.4493 3.396  |4.3884|4.6832|
+--------------------------------+------+------+
|3.0356 _0.38305 _4.308 3.1783  |4.2713|4.4493|
+--------------------------------+------+------+
|2.9773 _0.38427 _4.2168 3.0387 |4.1932|4.308 |
+--------------------------------+------+------+
|2.938 _0.38508 _4.1554 2.945   |4.1395|4.2168|
+--------------------------------+------+------+
|2.9107 _0.38564 _4.1127 2.8802 |4.1017|4.1554|
+--------------------------------+------+------+
|2.8914 _0.38603 _4.0825 2.8344 |4.0747|4.1127|
+--------------------------------+------+------+
|2.8775 _0.38631 _4.0607 2.8014 |4.0552|4.0825|
+--------------------------------+------+------+
```

最終結果を得るプログラム

```
   pow DS43
+---------------------------+-+------+
|1 _0.13636 _1.4091 0.95455|4|1.4091|
+---------------------------+-+------+
   22 * > {. pow DS43
22 _3 _31 21
```

第１固有値　4
第１固有ベクトル　　22 _3 _31 21

４．２　Deflation法のアルゴリズムとＪプログラムの実行

Power 法で求めた固有ベクトルを $\boldsymbol{v}_1$ 、最初の行列を A として、次の操作をおこなう。

文献[5], p.175,　文献[4], p.61

$$B = A - \frac{1}{(v_1\text{の第１成分})} v_1 \ ip \ \ (A\text{の第１行}),\quad ip = inner\ product$$

このとき、B の第１行は全ての要素が０になる。そして、第１行、第１列を除いた次元が１つ下がった行列が B_1 として得られる。この B_1 に対して、Power 法を用いて、２番目の固有値、固有ベクトルを求める。

Ｊプログラムによる途中経過をふくめた計算結果は次のようである。

```
   defpow DS43
j: 0
A
_2  1 _3  2
 1  1  1  0
 4 _2  5 _3
 5 _1  5  6
v1
      1
_0.13636
 _1.4091
 0.95455
lambda1 = 4
Xt
 _0.5
 0.25
_0.75
  0.5
lambda1 * v1 * Xt
     _2        1      _3        2
0.27273 _0.13636 0.40909 _0.27273
 2.8182  _1.4091  4.2273  _2.8182
_1.9091  0.95455 _2.8637   1.9091
B
      0        0       0        0
0.72727   1.1364 0.59091  0.27273
 1.1818 _0.59091 0.77272 _0.18182
 6.9091  _1.9546  7.8637   4.0909
B1
  1.1364 0.59091  0.27273
_0.59091 0.77272 _0.18182
 _1.9546  7.8637   4.0909
+------+-+-+
|1 _1 9|3|9|
+------+-+-+
```

```
u2x
0.11111 _0.11111 1
lambda2 = 3
---
       0
 0.11111
_0.11111
       1
v2 ----
v2a
       0
_0.11112
 0.11112
      _1
v2b
_0.5 0.25 _0.75 0.5
***
_2 1 _3 2
       0
 0.11111
_0.11111
       1
2.4444
---
  2.4444
_0.33333
 _3.4444
  2.3334
v2(result:)
  2.4444
_0.44445
 _3.3333
  1.3333
```

　計算結果だけを得るプログラムの実行は次のようになる。

```
   dpow DS43
lambda1 = 4
+--------------------------+-+
|1 _0.13636 _1.4091 0.95455|4|
+--------------------------+-+
lambda2 = 3
+------------------------------+-+
|2.4444 _0.44445 _3.3333 1.3333|3|
+------------------------------+-+
```

第２固有値　3

第２固有ベクトル　11 _2 _15 6

５．QR法による固有値－Ｊユーザに向けて

Ｊ言語にはQR分解を行うプリミティブ 128!:0が備えられている。

```
NB. QR method ==============================
qr =: 3 : 0
A1 =. (128!:0) y.
((>@{:) (+/ . *) (>@{.)) A1
)
diag =: i.@# {"_1 ]
eigen_value =: 3 : 0
40 eigen_value y.
:
diag qr^:x. y.
)
```

上のように、使い易くして、実行すると次のように、すべての固有値が一瞬で得られる。

```
   DS43
_2  1 _3  2
 1  1  1  0
 4 _2  5 _3
 5 _1  5  6
   eigen_value DS43
4.00002 2.99998 0.999998 2
```

６．鈴木先生の「美人のパターンを求める」主成分分析の章から

これらのデータは、ＪのSTYLEという名詞に入れられている。準備として、まず相関行列Rを計算する。それを元に固有値、固有ベクトルを求めて、主成分分析をおこなう。

今回のＪプログラムdpowでの結果は次のようであった。

```
   R =: corm STYLE
   R
        1 0.542494  0.326565   0.372425  0.0115011
 0.542494        1    0.3131   0.449056   0.241926
 0.326565   0.3131         1  0.0605705   0.424285
 0.372425 0.449056 0.0605705          1 _0.0649716
0.0115011 0.241926  0.424285 _0.0649716          1
   dpow R
lambda1 = 2.149
+--------------------------------+-----+
|1 1.0958 0.80605 0.7669 0.49502|2.149|
+--------------------------------+-----+
lambda2 = 1.3073
+-----------------------------------------+------+
|0.25038 0.11038 _0.51063 0.51575 _0.71769|1.3073|
```

+--+-------+

７．固有値、固有ベクトル計算のPower法とDeflation法ーJプログラム

```
NB. QR method ===============================
qr =: 3 : 0
A1 =. (128!:0) y.
((>@{:) (+/ . *) (>@{.)) A1
)

diag =: i.@# {"_1 ]

eigen_value =: 3 : 0
40 eigen_value y.
:
diag qr^:x. y.
)

NB. Roger Hui / M. Shimura 2010/8/3
rheval =: +/ . * & >/@ |. &(128!:0)
rh =: [: (<0 1)&|: rheval ^:_
rhn =: [: (<0 1)&|: rheval ^:(50)   NB. 50 repeat safely ! by TN

NB. Eigen Calc. / Power Iteration and Deflation ============
NB. 戸田、小野「入門数値計算」p.170-2

DT0 =: 2 2$1 8 2 1  NB. p.170
NB. Usage:
NB.   DT0 pow^:(4) 1 1;1 => x; r; el

DT1 =: 3 3$6 1 1 1 6 1 3 3 6

wr =: 1!:2&2

amax =: >./ @: |

ip =: +/ . *

cleanz =: * | >: 1e_4"_

pow0 =: 3 : 0
:
A =. x.
 x0 =. > {. y.
 el =. amax x0
 y0 =. x0 % el
 x1 =. A ip y0
 r =. (y0 ip x1) % (y0 ip y0)
```

```
wr 'x: ', ": x0
wr 'l: ', ": el
wr 'y: ', ": y0
wr 'x: ', ": x1
wr 'r: ', ": r
wr '-------------------'
x1; r; el
)

pow1 =: 3 : 0
:
A =. x.
 x0 =. > {. y.
 el =. amax x0
 y0 =. x0 % el
 x1 =. A ip y0
 r =. (y0 ip x1) % (y0 ip y0)
xx =. x1 % ({. x1)
xx; r; el
)

pow =: 3 : 0
A =. y.
9!:11 (5)  NB. floating 5 digits
cleanz L:0 A pow1^:(40) (({. $A)#1);1
)

max =: ({.@¥:) { ]

defpow =: 3 : 0
0 defpow y.
:
j =. x.
wr 'j: ', ": j
A =. y.
N =. {. $ A
wr 'A'
wr A
'v1 lam1 one' =: pow A
wr 'v1'
wr V1 =: (N, 1)$v1
wr 'lambda1 = ', ": lam1

wr 'X'
lam1 * (0{v1)
```

```
AT =: (N, 1) $ {. A
wr X =. AT % (lam1 * (0{v1))
wr 'lam1*v1*Xt'
wr C =. lam1 * ((N, 1)$ , v1) ip ((1, N)$ , X)
wr 'B'
wr cleanz B =. A -  C
wr 'B1'
wr BB =. }. }."(1) B

wr pow BB

'uu2 lam2 one' =. pow BB
wr 'u2x'
u2x =. uu2

wr u2x =. cleanz u2x % (max u2x)
wr 'lambda2 = ', ": lam2
wr '---'
wr u2 =: (N, 1)$0, u2x
wr 'v2 ----'
wr 'v2a'
wr v2a =. (lam2 - lam1) * u2
wr 'v2b'
wr ((1, N)$ , X)
wr '*** '
wr lam1 * ((1, N)$ , X)
wr ((N, 1)$,u2)
wr v2b1 =.  lam1 * ((1, N)$ , X) ip ((N, 1)$,u2)
wr '---'
wr v2b =. ({. , v2b1) * (N, 1)$v1
wr 'v2(result:)'
wr v2 =: cleanz v2a + v2b
'** end **' return.
)

dpow =: 3 : 0
0 dpow y.
:
j =. x.
A =. y.
N =. {. $ A
'v1 lam1 one' =: pow A
V1 =: (N, 1)$v1
wr 'lambda1 = ', ": lam1
wr v1;lam1
```

```
lam1 * (0{v1)
AT =: (N, 1) $ {. A
X =. AT % (lam1 * (0{v1))
C =. lam1 * ((N, 1)$ , v1) ip ((1, N)$ , X)
B =. A -  C
BB =. }. }."(1) B
'uu2 lam2 one' =. pow BB
u2x =. uu2
u2x =. cleanz u2x % (max u2x)
wr 'lambda2 = ', ": lam2
u2 =: (N, 1)$0, u2x
v2a =. (lam2 - lam1) * u2
v2b1 =.  lam1 * ((1, N)$ , X) ip ((N, 1)$,u2)
v2b =. ({. , v2b1) * (N, 1)$v1
v2 =: , cleanz v2a + v2b
v2;lam2
)
```