# データサイエンス概論 レポート

2016 年 12 月 4 日

学籍番号 201621639 山田純也
協力者： 関根 吉紀 長尾 悠真 中田 周育

```
In [1]: import numpy as np
        import numpy.linalg as la
        import pandas as pd
        %pylab inline
        from sympy import *
        init_printing()
        X = pd.read_csv('datas.csv', names=('v', 'v1', 'v2', 'v3'))
        Y = pd.read_csv('datas2.csv', names=('v1', 'vv'))
```

```
Populating the interactive namespace from numpy and matplotlib
```

## 1 問 1

### 1.1 問 1(1) 積和行列を係数とする正規方程式を作って解く

まず，積和行列$ X と Y $を求める．$x$ が説明変数のデータ行列，$y$ が目的変数のデータ行列だとすると，
$X = x^T x$
$Y = x^T y$
となる．

```
In [2]: x_11 = np.asarray(X)
        y_11 = np.asarray(Y)[:, 0][:, np.newaxis]
        xx = x_11.T.dot(x_11)
        yy = x_11.T.dot(y_11)
        A_11 = la.inv(xx).dot(yy)
```

求めた $X$ は，次の通り．

```
In [3]: Matrix(xx)
```

Out[3]:

$$\begin{bmatrix} 15.0 & 500.0 & 2349.0 & 711.0 \\ 500.0 & 17348.0 & 78674.0 & 24451.0 \\ 2349.0 & 78674.0 & 368585.0 & 111981.0 \\ 711.0 & 24451.0 & 111981.0 & 34857.0 \end{bmatrix}$$

求めた $Y$ は，次の通り．

```
In [4]: Matrix(yy)
```

Out[4]:

$$\begin{bmatrix}393.0\\13395.0\\61824.0\\19033.0\end{bmatrix}$$

求める成分 $a_0, a_1, a_2, a_3$ を $A = (a_0 a_1 a_2 a_3)^T$ とおくと，積和行列を用いて
$XA = Y$
となる． $X$ の逆行列 $X^{-1}$ を左から乗じると，
$X^{-1}XA = X^{-1}Y$
$A = X^{-1}Y$
となり，$A$ を求めることができる．実際に求めた $A$ が以下のとおりである．

```
In [5]: Matrix(A_11)
```

Out[5]:

$$\begin{bmatrix}-13.2172983163764\\0.201376887707632\\0.171024571169944\\0.124942775265382\end{bmatrix}$$

## 1.2 問 1-2 偏差積和行列を係数とする正規方程式を作って解く

まず，偏差積和行列$ X' と Y' を求める．また, x'がx$の偏差行列だとすると，
$X' = x'^T x'$
$Y' = x'^T y$
となる．

```
In [6]: x_ = np.asarray(X)[:, [1, 2, 3]]
        x_mean = x_.mean(axis=0)
        x_ = x_ - x_mean
        x_12 = np.ones((x_.shape[0], x_.shape[1] + 1))
        x_12[:, 1:] = x_
        y_12 = np.asarray(Y)[:, 0][:, np.newaxis]
        xx = x_12.T.dot(x_12)
        yy = x_12.T.dot(y_12)
        A_12 = np.linalg.inv(xx).dot(yy)
```

求めた $X'$ は，次の通り．

```
In [7]: Matrix(xx)
```

Out[7]:

$$\begin{bmatrix} 15.0 & -3.5527136788005 \cdot 10^{-14} & 8.5265128291212 \cdot 10^{-14} & 2.1316282072803 \cdot 10^{-14} \\ -3.5527136788005 \cdot 10^{-14} & 681.333333333333 & 374.0 & 751.0 \\ 8.5265128291212 \cdot 10^{-14} & 374.0 & 731.6 & 638.4 \\ 2.1316282072803 \cdot 10^{-14} & 751.0 & 638.4 & 1155.6 \end{bmatrix}$$

求めた $Y'$ は，次の通り．

```
In [8]: Matrix(yy)
```

Out[8]:

$$\begin{bmatrix} 393.0 \\ 294.999999999999 \\ 280.200000000002 \\ 404.8 \end{bmatrix}$$

この問で求める成分を $A' = (a_0 a_1 a_2 a_3)^T$ とおくと， 積和行列を用いて

$X'A' = Y'$

となる． $X'$ の逆行列 $X'^{-1}$ を左から乗じると，

$X'^{-1}X'A' = X'^{-1}Y'$

$A' = X'^{-1}Y'$

となり，$A'$ を求めることができる．実際に求めた $A'$ が以下のとおりである．

```
In [9]: Matrix(A_12)
```

Out[9]:

$$\begin{bmatrix} 26.2 \\ 0.201376887707657 \\ 0.171024571169932 \\ 0.124942775265373 \end{bmatrix}$$

この結果を見ればわかるが，$a_0$ 以外の要素は先程問 1(1) で求めた値と殆ど同じである．

## 2　問 2 ベクトル $y - \hat{y}$ の長さを求める

ここで $y$ は目的変数の測定値，$\hat{y}$ は重回帰分析によって求めた $A$ および $A'$ を用いて求めた値とする．

$A$ と $A'$ でそれぞれ長さを求める．

```
In [10]: yr_11 = x_11.dot(A_11)
         yr_12 = x_12.dot(A_12)
         y = np.asarray(Y)[:, 0][:, np.newaxis]
         n1 = la.norm(yr_11 - y_11)
         n2 = la.norm(yr_12 - y_12)
```

問 1(1) で求めた $A$ を用いた場合のベクトル $y - \hat{y}$ の長さは，以下の通りである．

```
In [11]: n1
```

Out[11]:

8.39618352926

問 1(2) で求めた $A'$ を用いた場合のベクトル $y - \hat{y}$ の長さは，以下の通りである．

```
In [12]: n2
```

```
Out[12]:
```

$$8.39618352926$$

従って積和行列を用いて重回帰分析を行った場合も，偏差積和行列を用いて重回帰分析を行った場合も，求めた最適な $a_0$ の値こそ異なるが，予測を行った際の誤差は同じ値となることがわかった．

## 3 問 3 主成分分析

```
In [13]: x_ = np.asarray(X)[:, [1, 2, 3]]
         x_mean = x_.mean(axis=0)
         x_3 = x_ - x_mean
```

```
In [14]: xx = x_3.T.dot(x_3)
```

```
In [15]: ef, ev = la.eig(xx)
```

```
In [16]: # 固有値と固有ベクトルのソート
         max_ef = 0.
         for i in range(len(ef)):
             max_i = None
             max_ef = 0.
             for j in range(i, len(ef)):
                 if max_ef < ef[j]:
                     max_i = j
                     max_ef = ef[j]

             tmp1 = ef[i].copy()
             tmp2 = ev[:, i].copy()
             ef[i] = ef[max_i].copy()
             ev[:, i] = ev[:, max_i].copy()
             ef[max_i] = tmp1
             ev[:, max_i] = tmp2
```

### 3.1 問 3(1) 偏差積和行列の固有値 3 つを求める

まず，偏差積和行列$ X' を求める．また, x'がx$の偏差行列だとすると，
$X' = x'^T x'$
となる．
ただし，先程の問 1(2) で用いた偏差積和行列とは違い，データ列に定数項の計算に用いる 1 を挿入していないため，偏差積和行列の大きさは $3 \times 3$ となる．
求めた $X'$ は，次の通り．

```
In [17]: Matrix(xx)
```

Out[17]:

$$\begin{bmatrix}681.333333333333 & 374.0 & 751.0\\374.0 & 731.6 & 638.4\\751.0 & 638.4 & 1155.6\end{bmatrix}$$

求めた固有値は，次の 3 つである．

```
In [18]: Matrix(ef).T
```

Out[18]:

$$\begin{bmatrix}2102.10812120784 & 355.877512637534 & 110.54769948796\end{bmatrix}$$

## 3.2 問 3(2) 各固有値に対応した固有ベクトルを求める

求めた固有ベクトルは，以下の通りである．先程の問 3(1) で求めた固有値にそれぞれ対応している．また，固有値が大きい順に並べており，順番に第一主成分，第二主成分，第三主成分となっている．

```
In [19]: print('固有値 {} に対応している固有ベクトル'.format(ef[0]))
         Matrix(ev[:, 0])
```

**固有値** 2102.1081212078398 **に対応している固有ベクトル**

Out[19]:

$$\begin{bmatrix}-0.505778333258832\\-0.473821861889441\\-0.720889118243258\end{bmatrix}$$

```
In [20]: print('固有値 {} に対応している固有ベクトル'.format(ef[1]))
         Matrix(ev[:, 1])
```

**固有値** 355.87751263753404 **に対応している固有ベクトル**

Out[20]:

$$\begin{bmatrix}0.499953851118198\\-0.842006600661878\\0.202659890441869\end{bmatrix}$$

```
In [21]: print('固有値 {} に対応している固有ベクトル'.format(ef[2]))
         Matrix(ev[:, 2])
```

**固有値** 110.5476994879596 **に対応している固有ベクトル**

Out[21]:

$$\begin{bmatrix}-0.703018082525621\\-0.257910309288812\\0.662757759671321\end{bmatrix}$$

## 3.3 問 3(3) $Wz(a_1, a_2, a_3)$ を求める

```
In [22]: z = x_3.dot(ev)
         z_mean = z.mean(axis=0)
```

```
z_dev = z - z_mean
wz = (z_dev**2).sum(axis=0)
```

$Wz(a_1, a_2, a_3) \equiv \Sigma(z_i - \bar{z})$ と定義されている．それぞれ，第一，第二，第三主成分の $Wz(a_1, a_2, a_3)$ を求める．

第一主成分に対応する $Wz(a_1, a_2, a_3)$ は次の通り．

```
In [23]: wz[0]
```

Out[23]:

$$2102.10812121$$

第二主成分に対応する $Wz(a_1, a_2, a_3)$ は次の通り．

```
In [24]: wz[1]
```

Out[24]:

$$355.877512638$$

第三主成分に対応する $Wz(a_1, a_2, a_3)$ は次の通り．

```
In [25]: wz[2]
```

Out[25]:

$$110.547699488$$

これらの値はそれぞれ第一，第二，第三主成分に対応する固有値と同じである．

# 4　問 4 全体の分散が Σ 各主成分の分散となることを確認する

先程の問 3 で求めたものをデータ数で割ったものが，各主成分の分散である．すなわちこれらの合計と，主成分全体の分散が一致すればよい．

Σ 各主成分の分散 は，以下の値となる

```
In [26]: wz.sum() / len(z)
```

Out[26]:

$$171.235555556$$

また，主成分全体の分散が，以下の値である．

```
In [27]: (z**2).sum() / len(z)
```

Out[27]:

$$171.235555556$$

従って，全体の分散が Σ 各主成分の分散 であることが確認された．