数理工学実験
最小二乗法

滝本亘（学籍番号 1029-33-1175）

2023 年 1 月 9 日

1　課題 6(重回帰問題)

1.1　はじめに

各入力 $x_i \in \mathbb{R}^2$, $i = 1, 2, \cdots, 10000$ に対し、次の式で観測データ $y_i \in \mathbb{R}$ が生成されているとする。

$$y_i = x_i^T \theta + w_i \tag{1.1}$$

ここで観測誤差は $\mathcal{N}(0, 1)$ に従って発生している（ただし、平均の情報のみ知っているとし、分散も分布も知らないものとする）。このとき、次の問に答えよ。

1. θ の最小二乗誤差推定量を、与えられた全てのデータ $\{(x_i, y_i)\}_{i=1}^{10000}$ を使って求めよ。また、そのときの推定誤差共分散行列を求めよ。
2. 用いるデータ $\{(x_i, y_i)\}_{i=1}^{N}$ を増やしたとき、$\hat{\theta}_N$ の各要素が収束していくことを片対数グラフでプロットして確認せよ。ただし、$N = 2, 4, 8, \cdots, 2^{13} = 8192$ のときのみでよい。
3. 全てのデータを用いたときの決定変数を求めよ。

1.2　原理・方法の詳細

1.2.1　回帰問題

まずは θ が f に対して線形である場合を扱う。このとき、適当な行列置関数 $\varphi : \mathbb{R}^p \to \mathbb{R}^{m \times n}$ が存在して、

$$f(\theta, x) = \varphi(x)\theta \tag{1.2}$$

となる場合を考えていることになる。ここで、$\sum_{i=1}^{N} \varphi(x_i)^T \varphi(x_i)$ が正則ならば、最適化問題

$$\min_{\theta \in \mathbb{R}^n} \sum_{i=1}^{N} \|y_i - f(\theta, x_i)\|^2 \tag{1.3}$$

の最適解は次で求められる。

$$\hat{\theta}_N = \left(\sum_{i=1}^{N} \varphi(x_i)^T \varphi(x_i) \right)^{-1} \sum_{j=1}^{N} \varphi(x_j)^T y_j \tag{1.4}$$

この $\hat{\theta}_N$ を最小二乗誤差推定値あるいは最小二乗推定値という。このデータをもらったときの処理の仕方（推定規則）のことを、最小二乗誤差推定量という。また、このとき観測誤差 $w_i, i = 1, \cdots, N$ の共分散行列の推定値は、

$$\hat{V}_N = \frac{1}{N - n} \sum_{i=1}^{N} \left(y_i - \varphi(x_i)\hat{\theta}_N \right) \left(y_i - \varphi(x_i)\hat{\theta}_N \right)^T \tag{1.5}$$

で与えられる。以降、表記の簡単のために、

$$\Phi_N := \left(\sum_{i=1}^{N} \varphi_i^T \varphi_i \right)^{-1}, \quad \varphi_i := \varphi(x_i) \tag{1.6}$$

とおく。

得られた推定値がどれくらいよいか、検討する。真のパラメータ θ と $\hat{\theta}_N$ の間には

$$\begin{aligned}\theta - \hat{\theta}_N &= \Phi \left(\Phi_N^{-1}\theta - \sum_{j=1}^{N} \varphi_j^T y_j \right) \\ &= \Phi_N \sum_{j=1}^{N} \varphi_j^T (\varphi_j \theta - y_j) \\ &= \Phi_N \sum_{j=1}^{N} \varphi_j^T w_j \end{aligned} \tag{1.7}$$

の関係がある。入力のデータは既知の数であるので、確率的な変動は w_j のみに依存することに注意する。従って、w_j が平均ゼロであることは既知なので、推定が不偏的であるとき性質

$$\mathbb{E}[\theta - \hat{\theta}_N] = 0 \tag{1.8}$$

が成り立つ。また、もし観測誤差の共分散行列 V が既知ならば、推定誤差の共分散行列は

$$\mathbb{E}\left[(\theta - \hat{\theta}_N)(\theta - \hat{\theta}_N)^T\right] = \Phi_N \sum_{i=1}^{N} \varphi_i^T V \varphi_i \Phi_N \tag{1.9}$$

を得る。とくに $V = \sigma^2 I_n$ ならば、

$$\mathbb{E}\left[(\theta - \hat{\theta}_N)(\theta - \hat{\theta}_N)^T\right] = \sigma^2 \Phi_N \tag{1.10}$$

となり、Φ_N は N に対して（正定値行列の意味で）単調減少なので、漸近的に推定誤差がゼロに近づくことが示唆される。観測誤差の推定値がわからない場合は、式 (1.9) 右辺の V を $\hat{V}_N$ に置き換えればよい。

1.2.2 決定変数

観測誤差の確率分布が不明であるならば、得られたパラメータがどれほどよくデータを説明するかの指標として、決定変数と呼ばれる量が用いられる。

$$C := \frac{\sum_{i=1}^{N} \|\varphi_i \hat{\theta}_N - \bar{y}\|^2}{\sum_{i=1}^{N} \|y_i - \bar{y}\|^2} \tag{1.11}$$

ここで $\bar{y} := \frac{1}{N}\sum_{i=1}^{N} y_i$ とした。最小二乗推定値 $\hat{\theta}_N$ の決定変数は $[0, 1]$ の範囲に値を取り、その値が 1 に近いほどよい推定であるとみなされる。逆に 0 に近い場合は得られたパラメータ $\hat{\theta}_N$ が出力 y の推定に役に立たないことを意味する。

1.3 実験方法

コード 1: 課題 6-1

```
1  import numpy as np
2  import pandas as pd
3
4  df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\6\\mmse_kadai1.xls',header=
       None)
```

```
5
6   x=df.iloc[:,0:2].to_numpy()
7   y=df.iloc[:,2:3].to_numpy()
8
9   def X(i):
10      a=np.array([x[i]])
11      return a
12
13  def Y(i):
14      a=np.array([y[i]])
15      return a
16
17  def phi(N):
18      a=0
19      for i in range(N):
20          a+=np.dot(X(i).T,X(i))
21      a=np.linalg.inv(a)
22      return a
23
24  def theta(N):
25      a=phi(N)
26
27      b=0
28      for j in range(N):
29          b+=np.dot(X(j).T,Y(j))
30
31      c=np.dot(a,b)
32
33      return c
34
35  def V(N):
36      a=theta(N)
37
38      b=0
39      for i in range(N):
40          c=Y(i)-np.dot(X(i),a)
41          b+=np.dot(c,c.T)
42
43      d=b/(N-n)
44
45      return d
46
47  N=10000
48  n=2
49  print('最小二乗誤差推定量は\n',theta(N))
50  print('推定誤差共分散行列は\n',V(N))
```

コード 2: 課題 6-2

```
1   import numpy as np
2   import pandas as pd
3   import matplotlib.pyplot as plt
4
5   df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\6\\mmse_kadai1.xls',header=
        None)
6
7   x=df.iloc[:,0:2].to_numpy()
8   y=df.iloc[:,2:3].to_numpy()
9
10  def X(i):
11      a=np.array([x[i]])
12      return a
13
14  def Y(i):
15      a=np.array([y[i]])
16      return a
17
18  def phi(N):
```

```
19          a=0
20          for i in range(N):
21              a+=np.dot(X(i).T,X(i))
22          a=np.linalg.inv(a)
23          return a
24
25      def theta(N):
26          a=phi(N)
27
28          b=0
29          for j in range(N):
30              b+=np.dot(X(j).T,Y(j))
31
32          c=np.dot(a,b)
33
34          return c
35
36
37      n=[]
38      a1=[]
39      a2=[]
40      for i in range(1,14):
41          N=2**i
42          n.append(N)
43
44          a=theta(N)
45          a1.append(a[0])
46          a2.append(a[1])
47
48      plt.plot(n,a1,marker='o')
49      plt.axhline(y=1.50655081,linestyle='--',color='red')
50      plt.xscale('log',base=2)
51      plt.xlabel('N')
52      plt.ylabel(r'$\theta$[0]')
53      plt.show()
```

コード 3: 課題 6-3

```
1       import numpy as np
2       import pandas as pd
3
4       df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\6\\mmse_kadai1.xls',header=
            None)
5
6       x=df.iloc[:,0:2].to_numpy()
7       y=df.iloc[:,2:3].to_numpy()
8
9       def X(i):
10          a=np.array([x[i]])
11          return a
12
13      def Y(i):
14          a=np.array([y[i]])
15          return a
16
17      def phi(N):
18          a=0
19          for i in range(N):
20              a+=np.dot(X(i).T,X(i))
21          a=np.linalg.inv(a)
22          return a
23
24      def theta(N):
25          a=phi(N)
26
27          b=0
28          for j in range(N):
29              b+=np.dot(X(j).T,Y(j))
```

```
30
31          c=np.dot(a,b)
32
33          return c
34
35      def C(N):
36          a=theta(N)
37
38          b=0
39          for i in range(N):
40              b+=Y(i)
41          b/=N
42
43          c=0
44          for i in range(N):
45              c0=np.dot(X(i),a)−b
46              c+=np.dot(c0,c0)
47
48          d=0
49          for i in range(N):
50              d0=Y(i)−b
51              d+=np.dot(d0,d0)
52
53          e=c/d
54
55          return e
56
57      N=10000
58      print('決定変数は',C(N))
```

1.4 結果

(1)

最小二乗誤差推定量は、

$$\hat{\theta}_N = (1.50655081, 1.99769567)^T \tag{1.12}$$

推定誤差共分散行列は、

$$\mathbb{E}\left[(\theta - \hat{\theta}_N)(\theta - \hat{\theta}_N)^T\right] = \begin{pmatrix} 9.86649107 \times 10^{-5} & -4.08165714 \times 10^{-7} \\ -4.08165714 \times 10^{-7} & 1.00524760 \times 10^{-4} \end{pmatrix} \tag{1.13}$$

(2)

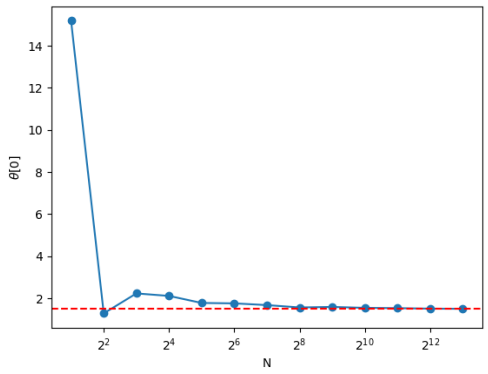


図 1: $\theta[0]$

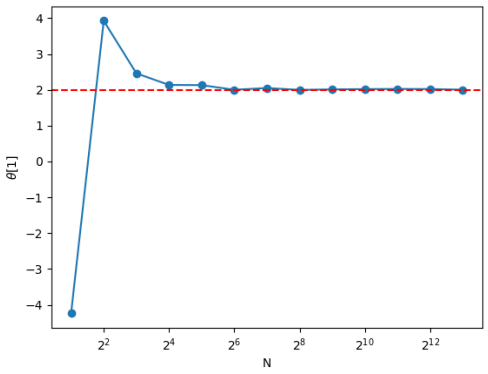


図 2: $\theta[1]$

(3)

決定変数は、

$$C = 0.86297344 \tag{1.14}$$

2　課題 7(多項式回帰)

2.1　はじめに

各入力 $x_i \in \mathbb{R}$, $i = 1, 2, \cdots, 10000$ に対し、次の式で観測データ $y_i \in \mathbb{R}$ が生成されている。

$$y_i = \varphi(x_i)\theta + w_i, \quad \varphi(x) := \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \tag{2.1}$$

ここで観測誤差は $\mathcal{N}(0, 9)$ に従って発生している（ただし、平均の情報のみ知っているとし、分散も分布も知らないものとする)。このとき、次の問いに答えよ。

1. θ の最小二乗誤差推定量を、与えられた全てのデータ $\{(x_i, y_i)\}_{i=1}^{10000}$ を使って求めよ。また、そのときの推定誤差共分散行列を求めよ。

2. 用いるデータ $\{(x_i, y_i)\}_{i=1}^{N}$ を増やしたとき、$\hat{\theta}_N$ の各要素が収束していくことを片対数グラフでプロットして確認せよ。ただし、$N = 4, 8, \cdots, 2^{13} = 8192$ のときのみでよい。

3. 全てのデータを用いたときの決定変数を求めよ.

2.2　実験方法

コード 4: 課題 7-1

```
1  import numpy as np
2  import pandas as pd
3
4  df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\7\\mmse_kadai2.xls',header=
       None)
5
6  x=df.iloc[:,0:1].to_numpy()
7  y=df.iloc[:,1:2].to_numpy()
8
9  def X(i):
10     a=np.array([[1,x[i][0],x[i][0]**2,x[i][0]**3]])
11     return a
12
13 def Y(i):
14     a=np.array([y[i]])
15     return a
16
17 def phi(N):
18     a=0
19     for i in range(N):
20         a+=np.dot(X(i).T,X(i))
21     a=np.linalg.inv(a)
22     return a
23
24 def theta(N):
25     a=phi(N)
26
27     b=0
28     for j in range(N):
29         b+=np.dot(X(j).T,Y(j))
30
31     c=np.dot(a,b)
32
33     return c
34
35 def V(N):
36     a=theta(N)
```

```
37
38          b=0
39          for i in range(N):
40              c=Y(i)−np.dot(X(i),a)
41              b+=np.dot(c,c.T)
42
43          d=b/(N−n)
44
45          return d
46
47      def E(N):
48          a=V(N)
49
50          b=0
51          for i in range(N):
52              b+=np.dot(np.dot(X(i).T,a),X(i))
53
54          c=np.dot(np.dot(phi(N),b),phi(N))
55
56          return c
57
58      N=10000
59      n=4
60      print('最小二乗誤差推定量は\n',theta(N))
61      print('推定誤差共分散行列は\n',E(N))
```

コード 5: 課題 7-2

```
1   import numpy as np
2   import pandas as pd
3   import matplotlib.pyplot as plt
4
5   df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\7\\mmse_kadai2.xls',header=
        None)
6
7   x=df.iloc[:,0:1].to_numpy()
8   y=df.iloc[:,1:2].to_numpy()
9
10  def X(i):
11      a=np.array([[1,x[i][0],x[i][0]**2,x[i][0]**3]])
12      return a
13
14  def Y(i):
15      a=np.array([y[i]])
16      return a
17
18  def phi(N):
19      a=0
20      for i in range(N):
21          a+=np.dot(X(i).T,X(i))
22      a=np.linalg.inv(a)
23      return a
24
25  def theta(N):
26      a=phi(N)
27
28      b=0
29      for j in range(N):
30          b+=np.dot(X(j).T,Y(j))
31
32      c=np.dot(a,b)
33
34      return c
35
36  n=[]
37  a1=[]
38  a2=[]
39  a3=[]
```

```
a4=[]
for i in range(2,14):
    N=2**i
    n.append(N)

    a=theta(N)
    a1.append(a[0])
    a2.append(a[1])
    a3.append(a[2])
    a4.append(a[3])

plt.plot(n,a1,marker='o')
plt.axhline(y=-0.50902942,linestyle='--',color='red')
plt.xscale('log',base=2)
plt.xlabel('N')
plt.ylabel(r'$\theta$[0]')
plt.show()
```

コード 6: 課題 7-3

```
import numpy as np
import pandas as pd

df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\7\\mmse_kadai2.xls',header=
    None)

x=df.iloc[:,0:1].to_numpy()
y=df.iloc[:,1:2].to_numpy()

def X(i):
    a=np.array([[1,x[i][0],x[i][0]**2,x[i][0]**3]])
    return a

def Y(i):
    a=np.array([y[i]])
    return a

def phi(N):
    a=0
    for i in range(N):
        a+=np.dot(X(i).T,X(i))
    a=np.linalg.inv(a)
    return a

def theta(N):
    a=phi(N)

    b=0
    for j in range(N):
        b+=np.dot(X(j).T,Y(j))

    c=np.dot(a,b)

    return c

def C(N):
    a=theta(N)

    b=0
    for i in range(N):
        b+=Y(i)
    b/=N

    c=0
    for i in range(N):
        c0=np.dot(X(i),a)-b
        c+=np.dot(c0,c0)

```

```
48          d=0
49          for i in range(N):
50              d0=Y(i)−b
51              d+=np.dot(d0,d0)
52
53          e=c/d
54
55          return e
56
57      N=10000
58      print('決定変数は',C(N))
```

2.3 結果

(1)

最小二乗誤差推定量は、

$$\hat{\theta}_N = (-0.50902942, 1.97586067, 0.19774405, -0.09866691)^T \tag{2.2}$$

推定誤差共分散行列は、

$$\mathbb{E}\left[(\theta - \hat{\theta}_N)(\theta - \hat{\theta}_N)^T\right]$$

$$= \begin{pmatrix} 2.02344200 \times 10^{-3} & -1.18649745 \times 10^{-5} & -1.34831217 \times 10^{-4} & 3.97116548 \times 10^{-7} \\ -1.18649745 \times 10^{-5} & 6.75301451 \times 10^{-4} & -1.89854517 \times 10^{-7} & -3.79471936 \times 10^{-5} \\ -1.34831217 \times 10^{-4} & -1.89854517 \times 10^{-7} & 1.60391021 \times 10^{-5} & 6.35182005 \times 10^{-8} \\ 3.97116548 \times 10^{-7} & -3.79471936 \times 10^{-5} & 6.35182005 \times 10^{-8} & 2.52871068 \times 10^{-6} \end{pmatrix} \tag{2.3}$$

(2)

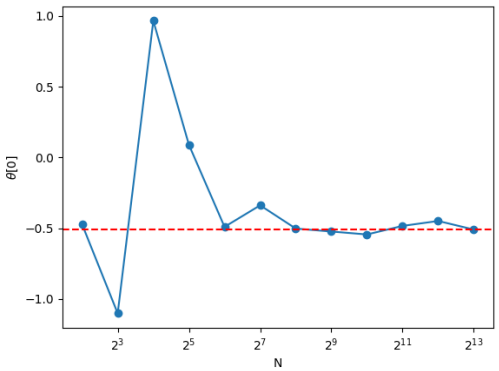


図 3: $\theta[0]$

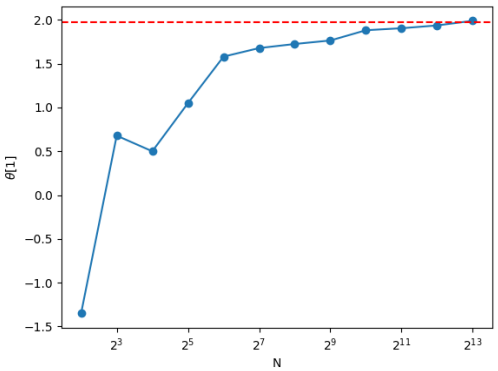


図 4: $\theta[1]$

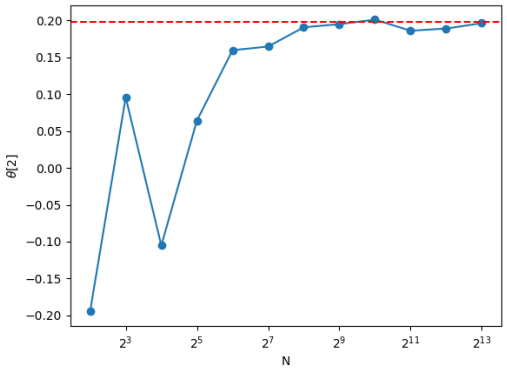


図 5: $\theta[2]$

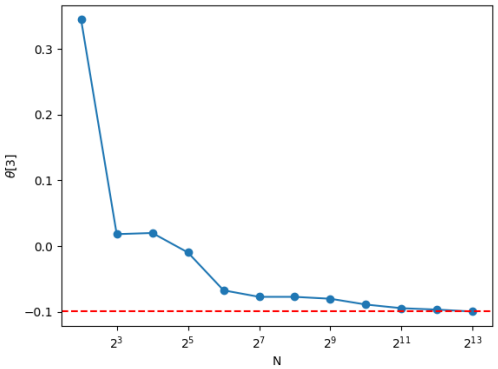


図 6: $\theta[3]$

(3)

決定変数は、

$$C = 0.461855 \tag{2.4}$$

3　課題 10(重み付き最小二乗法)

3.1　はじめに

$x_i \in \mathbb{R},\ i = 1, \cdots, 10000$ に対し、観測値の次元が 2 次元となる場合を考える。

$$y_i = \varphi(x_i)\theta + w_i \tag{3.1}$$

ここで w_i は $\mathcal{N}(0, V)$ の独立同一分布に従うとし、

$$\varphi = \begin{bmatrix} 1 & x \\ 1 & x^2 \end{bmatrix}, \quad V = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.2}$$

とする。このとき、

$$\hat{\theta}_N = \left(\sum_{i=1}^{N} \varphi(x_i)^T \varphi(x_i) \right)^{-1} \sum_{j=1}^{N} \varphi(x_j)^T y_j \tag{3.3}$$

$$\hat{\theta}_N = \left(\sum_{i=1}^{N} \varphi_i^T Q_i \varphi_i \right)^{-1} \sum_{j=1}^{N} \varphi_j^T Q_j y_j \tag{3.4}$$

をそれぞれ用いて推定値を求め、それぞれの推定誤差共分散を求めよ。ただし、$Q_i = V^{-1}$ とする。また、$\hat{\theta}_N$ の各要素の収束の仕方をプロットせよ。

3.2　原理・方法の詳細

3.2.1　重み付き最小二乗法

これまでは観測誤差は独立同一分布であることや、共分散行列の値が未知であることを仮定していたが、観測誤差は独立であっても同じ統計量をもつとはいえない場合も出てくる。データ $\{(x_i.y_i)\}_{i=1}^{N}$ を得たとき、それぞれのデータの信用度を考えて最小二乗法を解きたい場合もある。このとき、一般には重み行列 $Q_i \geq 0, i = 1, 2, \cdots, N$ を用いた次の重み付き最小二乗法

$$\min_{\theta \in \mathbb{R}^n} \sum_{i=1}^{N} (y_i - \varphi_i\theta)^T Q_i (y_i - \varphi_i\theta) \tag{3.5}$$

を考えればよい。もちろん、全ての重みがゼロ行列の場合は意味がないので、そのような例は考えない。この場合、$\sum_{i=1}^{N} \varphi_i^T Q_i \varphi_i$ が正則ならば、最適な推定値は

$$\hat{\theta}_N = \left(\sum_{i=1}^{N} \varphi_i^T Q_i \varphi_i \right)^{-1} \sum_{j=1}^{N} \varphi_j^T Q_j y_j \tag{3.6}$$

となる。重み付き最小二乗法とは通常、Q_i は対角行列とすることが多いが、ここでは重み行列の場合も重み付き最小二乗法と呼ぶ。

観測誤差の共分散行列 V が既知である場合、推定誤差共分散行列 (1.9) は最小の推定誤差共分散行列ではないことが知られている。V は正定値対称行列であるので、$V = W^2$ を満たす正定値行列 W が一意に存在する（V の平方根行列)。これを用いると、

$$y = z + w \tag{3.7}$$

は

$$W^{-1}y = W^{-1}\varphi(x)\theta + W^{1}w \tag{3.8}$$

と書き直すことができる。ここで新たな観測誤差 $w'_i = W^{-1}w$ は単位行列を分散としてもつ観測誤差である。この解は

$$\hat{\theta}_N = \left(\sum_{i=1}^{N} \varphi_i^T W^{-2} \varphi_i\right)^{-1} \sum_{j=1}^{N} \varphi_j^T W^{-2} y_j \tag{3.9}$$

となるので、$Q_i = aW^{-2} = aV^{-1}$ と選んだものに相当する（$a \in \mathbb{R}$ は正定数）。この観測誤差の統計量を用いて推定値 $\hat{\theta}_N$ を得る上記の手法は、最良線形不偏推定量と呼ばれる推定量と一致する。「最良」の名前の通り、これは最も推定誤差および訓練誤差の分散を最小にする推定規則である。正しい観測誤差の情報を持っていれば、素朴な定式化から得られる推定規則よりもいい結果が得られることを意味する。

3.3 実験方法

コード 7: 課題 10-1

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4
5  df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\10\\mmse_kadai5.xls',header=
       None)
6
7  x=df.iloc[:,0:1].to_numpy()
8  y=df.iloc[:,1:3].to_numpy()
9
10 def X(i):
11     a=np.array([[1,x[i][0]],[1,x[i][0]**2]])
12     return a
13
14 def Y(i):
15     a=np.array([[y[i][0]],[y[i][1]]])
16     return a
17
18 def phi(N):
19     a=0
20     for i in range(N):
21         a+=np.dot(X(i).T,X(i))
22     a=np.linalg.inv(a)
23     return a
24
25 def theta(N):
26     a=phi(N)
27
28     b=0
29     for j in range(N):
30         b+=np.dot(X(j).T,Y(j))
31
32     c=np.dot(a,b)
33
34     return c
35
36 def V(N):
37     a=theta(N)
38
39     b=0
40     for i in range(N):
41         c=Y(i)-np.dot(X(i),a)
42         b+=np.dot(c,c.T)
```

```
43
44          d=b/(N-n)
45
46          return d
47
48      N=1000
49      n=2
50      print('最小二乗誤差推定量は\n',theta(N))
51      print('推定誤差共分散行列は\n',V(N))
52
53      n=[]
54      a1=[]
55      a2=[]
56      for i in range(1,N):
57          n.append(i)
58
59          a=theta(i)
60          a1.append(a[0])
61          a2.append(a[1])
62
63      plt.plot(n,a1)
64      plt.axhline(y=2.99456713,linestyle='--',color='red')
65      plt.xscale('log')
66      plt.xlabel('N')
67      plt.ylabel(r'$\theta$[0]')
68      plt.show()
```

コード 8: 課題 10-2

```
1       import numpy as np
2       import pandas as pd
3       import matplotlib.pyplot as plt
4
5       df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\10\\mmse_kadai5.xls',header=
            None)
6
7       x=df.iloc[:,0:1].to_numpy()
8       y=df.iloc[:,1:3].to_numpy()
9
10      def X(i):
11          a=np.array([[1,x[i][0]],[1,x[i][0]**2]])
12          return a
13
14      def Y(i):
15          a=np.array([[y[i][0]],[y[i][1]]])
16          return a
17
18      def Q():
19          v=np.array([[100,0],[0,1]])
20          q=np.linalg.inv(v)
21          return q
22
23      def phi(N):
24          a=0
25          for i in range(N):
26              a+=np.dot(np.dot(X(i).T,Q()),X(i))
27          a=np.linalg.inv(a)
28          return a
29
30      def theta(N):
31          a=phi(N)
32
33          b=0
34          for j in range(N):
35              b+=np.dot(np.dot(X(j).T,Q()),Y(j))
36
37          c=np.dot(a,b)
38
```

```
39          return c
40
41      def V(N):
42          a=theta(N)
43
44          b=0
45          for i in range(N):
46              c=Y(i)−np.dot(X(i),a)
47              b+=np.dot(c,c.T)
48
49          d=b/(N−n)
50
51          return d
52
53      N=1000
54      n=2
55      print('最小二乗誤差推定量は\n',theta(N))
56      print('推定誤差共分散行列は\n',V(N))
57
58      n=[]
59      a1=[]
60      a2=[]
61      for i in range(1,N):
62          n.append(i)
63
64          a=theta(i)
65          a1.append(a[0])
66          a2.append(a[1])
67
68      plt.plot(n,a1)
69      plt.axhline(y=2.93908407,linestyle='--',color='red')
70      plt.xscale('log')
71      plt.xlabel('N')
72      plt.ylabel(r'$\theta$[0]')
73      plt.show()
```

3.4 結果

(1)

最小二乗誤差推定量は、

$$\hat{\theta}_N = (2.99456713, -2.06897079) \tag{3.10}$$

推定誤差共分散行列は、

$$\hat{V}_N = \begin{pmatrix} 0.03591211 & -0.01277071 \\ -0.01277071 & 0.01108541 \end{pmatrix} \tag{3.11}$$

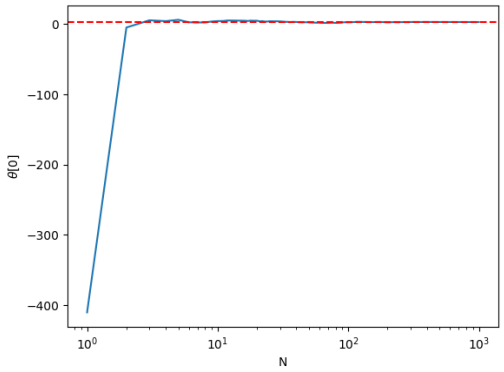


図 7: $\theta[0]$

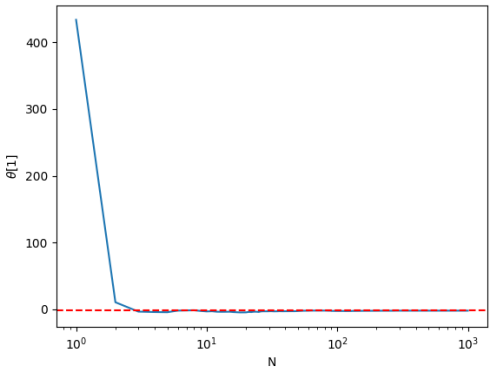


図 8: $\theta[1]$

(2)

最小二乗誤差推定量は、

$$\hat{\theta}_N = (2.93908407, -1.98646467) \tag{3.12}$$

推定誤差共分散行列は、

$$\hat{V}_N = \begin{pmatrix} 0.24522728 & -0.09870736 \\ -0.09870736 & 0.05095924 \end{pmatrix} \tag{3.13}$$

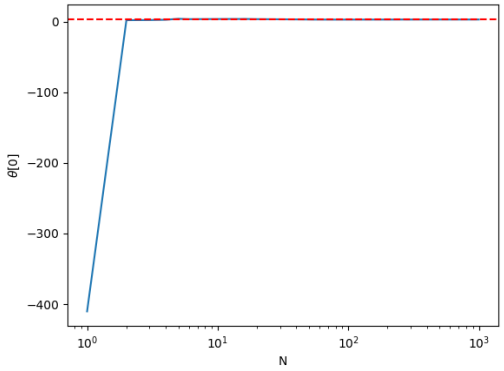


図 9: $\theta[0]$

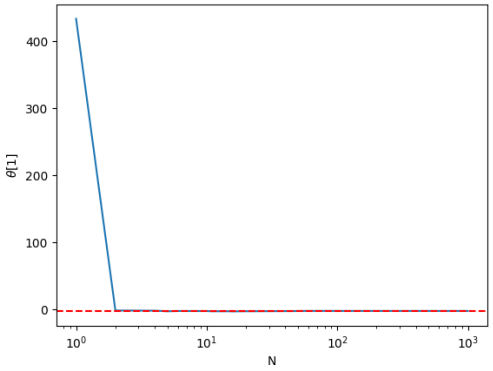


図 10: $\theta[1]$

4　課題 12(推定値の合成)

4.1　はじめに

$x_i \in \mathbb{R},\ i = 1, \cdots, 10000$ に対し、次の式で観測データ $y_i \in \mathbb{R}$ が生成されているとする。

$$y_i = \varphi(x_i)\theta + w_i \tag{4.1}$$

ここで

$$\varphi(x) := \begin{bmatrix} 1 & \exp\left(-\dfrac{(x_i-1)^2}{2}\right) & \exp(-(x_i+1)^2) \end{bmatrix} \tag{4.2}$$

であり、w_i は平均 0、分散有限の同一分布から生成されているもとする。10000 組のデータの組 $\{(x_i, y_i)\}_{i=1}^{10000}$ のうち、最初の 6000 組と残りの 4000 組のデータそれぞれで $\theta \in \mathbb{R}^3$ の推定値を求め、それらから推定値を合成せよ。また、全てデータを使った場合の推定値と比較し、一致することを確かめよ。

4.2　原理・方法の詳細

4.2.1　異なるデータセットからの推定値の合成

データ $\{(x_i, y_i)\}_{i=1}^{N}$ を得たとき、$f(\theta, x) = \varphi(x)\theta$ の場合の式 (1.3) の問題を解くと、式 (1.4) で解が得られることがわかったが、実際のデータ初期では、別のデータセッと $\{(x'_j, y'_j)\}_{j=1}^{M}$ を用いた推定値 $\hat{\theta}_M$ が独立に存在する場合もある。それらは、適切な仮定の下で、最小二乗法を用いて以下のように推定値を得ることができる。

$$\hat{\theta}_M = \left(\sum_{i=1}^{M} \varphi(x'_i)^T \varphi(x'_i)\right)^{-1} \sum_{j=1}^{M} \varphi(x'_j) y'_j \tag{4.3}$$

2 組のデータセッとからそれぞれの推定値が得られているが、より多くのデータセットから推定する方が推定精度は改善することが期待される。このとき、データを合わせて改めて解き直してもよいが、2 つの推定値を用いて、推定値を更新することができる。便宜的に $\{(x'_j, y'_j)\}_{j=1}^{M}$ のラベルをつけ直して $\{(x_i, y_i)\}_{i=N+1}^{N+M} = \{(x'_j, y'_j)\}_{j=1}^{M}$ とすると、

$$\begin{aligned}
\hat{\theta}_{N+M} &= (\Phi_N^{-1} + \Phi_M'^{-1})^{-1} \sum_{i=1}^{N+M} \varphi_i^T y_i \\
&= (\Phi_N^{-1} + \Phi_M'^{-1})^{-1} \sum_{i=1}^{N} \varphi_i^T y_i + (\Phi_N^{-1} + \Phi_M'^{-1})^{-1} \sum_{i=N+1}^{N+M} \varphi_i^T y_i \\
&= (\Phi_N^{-1} + \Phi_M'^{-1})^{-1} \Phi_N^{-1} \Phi_N \sum_{i=1}^{N} \varphi_i^T y_i + (\Phi_N^{-1} + \Phi_M'^{-1})^{-1} \Phi_M'^{-1} \Phi'_M \sum_{i=N+1}^{N+M} \varphi_i^T y_i \\
&= (\Phi_N^{-1} + \Phi_M'^{-1})^{-1} (\Phi_N^{-1} \hat{\theta}_N + \Phi_M'^{-1} \hat{\theta}'_M)
\end{aligned} \tag{4.4}$$

となる。したがって、新たなデータを得ることが想定されるならば、推定値 $\hat{\theta}$ のみでなく Φ_N も保存しておけば、元のデータを保存しておく必要はない。

4.3 実験方法

コード 9: 課題 12

```
import math
import numpy as np
import pandas as pd

df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\12\\mmse_kadai7.xls',header=None)

x=df.iloc[:,0:1].to_numpy()
y=df.iloc[:,1:2].to_numpy()

def X(i):
    a=np.array([[1,math.exp(-(x[i][0]-1)**2/2),math.exp(-(x[i][0]+1)**2)]])
    return a

def Y(i):
    a=np.array([y[i]])
    return a

def phi(N,M):
    a=0
    for i in range(N,M):
        a+=np.dot(X(i).T,X(i))
    a=np.linalg.inv(a)
    return a

def theta(N,M):
    a=phi(N,M)

    b=0
    for j in range(N,M):
        b+=np.dot(X(j).T,Y(j))

    c=np.dot(a,b)

    return c

def theta2(N,M):
    a=np.linalg.inv(phi(0,N))+np.linalg.inv(phi(N,N+M))
    a=np.linalg.inv(a)

    b=np.dot(np.linalg.inv(phi(0,N)),theta(0,N))+np.dot(np.linalg.inv(phi(N,N+M)),theta(N,N+M))

    c=np.dot(a,b)

    return c

N=6000
M=4000
print('最初の組の最小二乗誤差推定量は 6000\n',theta(0,N))
print('残りの組の最小二乗誤差推定量は 4000\n',theta(N,N+M))
print('合成値は\n',theta2(N,M))

print('全データでは\n',theta(0,N+M))
```

4.4 結果

最初の 6000 組の最小二乗誤差推定量は

$$\hat{\theta}_N = (-0.00387938, 3.0107149, -1.98943435)^T \tag{4.5}$$

残りの 4000 組の最小二乗誤差推定量は

$$\hat{\theta}_M = (-0.02537589, 3.03489937, -1.97772731)^T \tag{4.6}$$

合成値は

$$\hat{\theta}_{N+M} = (-0.0124955, 3.02042997, -1.98486916)^T \tag{4.7}$$

全データでは

$$\hat{\theta}_{N+M} = (-0.0124955, 3.02042997, -1.98486916)^T \tag{4.8}$$

5　課題 13(異なる雑音の場合の推定値)

5.1　はじめに

$x_i \in \mathbb{R}$, $i = 1, \cdots, 10000$ に対し、次の式で観測データ $y_i \in \mathbb{R}$ が生成されているとする。

$$y_i = \varphi(x_i)\theta + w_i \tag{5.1}$$

ここで w_i は最初の 6000 組のと残りの 4000 組のデータで、平均ゼロの異なる独立な分布に従うものとする（$i = 1, \cdots, 6000$ では独立同一分布、$i = 6001, \cdots, 10000$ では異なる独立同一分布）。

$$\varphi(x) := \begin{bmatrix} 1 & \exp\left(-\dfrac{(x_i-1)^2}{2}\right) & \exp(-(x_i+1)^2) \end{bmatrix} \tag{5.2}$$

である。10000 組のデータの組 $\{(x_i, y_i)\}_{i=1}^{10000}$ のうち、最初の 6000 組と残りの 4000 組のデータそれぞれで $\theta \in \mathbb{R}^3$ の推定値と偶然誤差 w_i の分散の推定値を求め、推定された分散を用いて推定値を合成せよ。また、全データを使い、式 (1.4) を用いて得た $\hat{\theta}_{10000}$ と、どちらが優れているか比較考察せよ（比較の仕方を考えること）。

5.2　原理・方法の詳細

5.2.1　異なるデータセットからの推定値の合成（つづき）

観測誤差の大きさが異なる場合、重み付きの最小二乗法を使った方が推定結果がよくなることが期待される。データ数がそれぞれ N_1, N_2 のデータセットが 2 組あたえられているとき、重み行列 Q_1 と Q_2 を使ったときのそれぞれのでデータセットにおける推定値を $\hat{\theta}_{N_1}$, $\hat{\theta}_{N_2}$ とおくと、

$$\hat{\theta}_{N_1+N_2} = \left(\Phi_{Q_1,N_1}^{-1} + \Phi_{Q_2,N_2}^{-1}\right)^{-1} \times \left(\Phi_{Q_1,N_1}^{-1}\hat{\theta}_{N_1} + \Phi_{Q_2,N_2}^{-1}\hat{\theta}_{N_2}\right) \tag{5.3}$$

となる。ここで

$$\Phi_{Q,N} := \left(\sum_{i+1}^{N} \varphi_i^T Q \varphi_i\right)^{-1} \tag{5.4}$$

とした。

5.3　実験方法

コード 10: 課題 13

```
1  import math
2  import numpy as np
3  import pandas as pd
4
5  df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\13\\mmse_kadai8.xls',header=
       None)
6
7  x=df.iloc[:,0:1].to_numpy()
8  y=df.iloc[:,1:2].to_numpy()
9
10 def X(i):
11     a=np.array([[1,math.exp(-(x[i][0]-1)**2/2),math.exp(-(x[i][0]+1)**2)]])
12     return a
13
```

```
14  def Y(i):
15      a=np.array([y[i]])
16      return a
17
18  def phi(N,M):
19      a=0
20      for i in range(N,M):
21          a+=np.dot(X(i).T,X(i))
22      a=np.linalg.inv(a)
23      return a
24
25  def theta(N,M):
26      a=phi(N,M)
27
28      b=0
29      for j in range(N,M):
30          b+=np.dot(X(j).T,Y(j))
31
32      c=np.dot(a,b)
33
34      return c
35
36  def V(N,M):
37      a=theta(N,M)
38
39      b=0
40      for i in range(N,M):
41          c=Y(i)-np.dot(X(i),a)
42          b+=np.dot(c,c.T)
43
44      d=b/(N-n)
45
46      return d
47
48  def phi2(N,M):
49      a=1/V(N,M)
50
51      b=0
52      for i in range(N,M):
53          b+=np.dot(np.dot(X(i).T,a),X(i))
54      b=np.linalg.inv(b)
55      return b
56
57  def theta2(N,M):
58      a=np.linalg.inv(phi2(0,N))+np.linalg.inv(phi2(N,N+M))
59      a=np.linalg.inv(a)
60
61      b=np.dot(np.linalg.inv(phi2(0,N)),theta(0,N))+np.dot(np.linalg.inv(phi2(N,N+M)),theta(
            N,N+M))
62
63      c=np.dot(a,b)
64
65      return c
66
67  N=6000
68  M=4000
69  n=3
70  print('最初の組の最小二乗誤差推定量は 6000\n',theta(0,N),'\偶然誤差の共分散行列の推定値は
        n',V(0,N))
71  print('残りの組の最小二乗誤差推定量は 4000\n',theta(N,N+M),'\偶然誤差の共分散行列の推定
        値は n',V(N,N+M))
72  print('合成値は\n',theta2(N,M))
73
74  print('全データでは\n',theta(0,N+M))
```

5.4 結果

最初の 6000 組の最小二乗誤差推定量は

$$\hat{\theta}_N = (0.00707684, 3.28054335, -2.1908997)^T \tag{5.5}$$

偶然誤差の共分散行列の推定値は

$$\hat{V}_N = \Big(-193629.72380112\Big) \tag{5.6}$$

残りの 4000 組の最小二乗誤差推定量は

$$\hat{\theta}_M = (0.09848311, 3.10040485, -2.09100212)^T \tag{5.7}$$

偶然誤差の共分散行列の推定値は

$$\hat{V}_M = \Big(0.00686778\Big) \tag{5.8}$$

合成値は

$$\hat{\theta}_{N+M} = (0.09848311, 3.10040484, -2.09100211)^T \tag{5.9}$$

全データでは

$$\hat{\theta}_{N+M} = (0.04373209, 3.20866562, -2.15117856)^T \tag{5.10}$$

6　課題 14(システム同定)

6.1　はじめに

次のバネ・マス・ダンパ系を表す直線上の運動方程式を考える。

$$M\frac{d^2}{dt^2}y(t) + D\frac{d}{dt}y(t) + Ky(t) = F(t) \tag{6.1}$$

ここで M, K, D は正定数で、$F(t)$ は外力である。$y(t)$ は観測でき、$F(t)$ はこちらで設定できるものとする。このとき、得られるデータから (M, K, D) を決定したい。微少な δt では

$$\frac{d}{dt}y(t) \simeq \frac{y((k+1)\delta t) - y(k\delta t)}{\delta t} \tag{6.2}$$

$$\frac{d^2}{dt^2}y(t) \simeq \frac{y((k+2)\delta t) - 2y((k+1)\delta t) + y(k\delta t)}{\delta t^2} \tag{6.3}$$

と近似できることを利用し、$y_k := y(k\delta t)$, $F_k := F(k\delta t)$ とすると、

$$y_k = \left(2 - \frac{D}{M}\delta t\right) y_{k-1} + \left(1 + \frac{D}{M}\delta t - \frac{K}{M}\delta t^2\right) y_{k-2} + \frac{\delta t^2}{M}F_{k-2} + w_k \tag{6.4}$$

を得る。ここで w_k は近似の際に生じる誤差である。したがって、問題は

$$y_k = x_k^T\theta + w_k, \quad x_k := \begin{bmatrix} y_{k-1} \\ y_{k-2} \\ F_{k-2} \end{bmatrix} \tag{6.5}$$

の θ を求める問題にすることができる。パラメータの真値を $(M, D, K) = (2, 1, 3)$ とし、$\delta t = 0.01$、w_k は $[-1, 1]$ の一様分布に従う各時刻で独立な確率変数であるとして、以下の問いに答えよ。

1. $y_{-2} = 0$, $y_{-1} = 0$ とし、$F_k = 1$, $k = -2, -1, 0, 1, 2, \cdots$ として一定の外力を加えたとき、パラメータ θ を逐次最小二乗法で求めよ。ただし、パラメータの初期値は 0 とし、$k = 10000$ までよい。また正則化項は工夫せよ。

2. $y_{-2} = 0$, $y_{-1} = 0$ とし、$F_k = \sin(\pi k/5)$, $k = -2, -1, 0, 1, 2, \cdots$ として一定の外力を加えたとき、パラメータ θ を逐次最小二乗法で求めよ。ただし、パラメータの初期値は 0 とし、$k = 10000$ までよい。また正則化項は工夫せよ。

3. $y_0 = 0$, $y_1 = 0$ とする。このとき、どのような外力を加えればパラメータの推定が上手くいくか考え実装し、逐次最小二乗法で求めよ。ただし、パラメータの初期値は 0 とし、$k = 10000$ までよい。

6.2　原理・方法の詳細

6.2.1　逐次最小二乗法

最小二乗法を実時間処理で解くことを考える。具体的には、N 個のデータから推定された $\hat{\theta}_N$ を新たなデータ (x_{N+1}, y_{N+1}) を用いて再帰的に $\hat{\theta}_{N+1}$ を求めることを考える。最小二乗誤差推定

(1.4) より、次のように式変形できる。

$$\begin{aligned}\hat{\theta}_N &= \Phi_{N+1}\sum_{j=1}^{N+1}\varphi_j^T y_j \\ &= \left(\Phi_N^{-1}+\varphi_{N+1}^T\varphi_{N+1}\right)^{-1}\times\left(\varphi_{N+1}^T y_{N+1}+\sum_{j=1}^{N}\varphi_j^T y_j\right) \\ &= \left\{\Phi_N-\Phi_N\varphi_{N+1}^T\times\left(I_m+\varphi_{N+1}\Phi_N\varphi_{N+1}^T\right)^{-1}\varphi_{N+1}\Phi_N\right\}\times\left(\varphi_{N+1}^T y_{N+1}+\sum_{j=1}^{N}\varphi_j^T y_j\right) \\ &= \hat{\theta}_N+K_{N+1}\left(y_{N+1}-\varphi_{N+1}\hat{\theta}_N\right) \qquad (6.6)\end{aligned}$$

ただし

$$K_{N+1}:=\Phi_N\varphi_{N+1}^T\left(I_m+\varphi_{N+1}\Phi_N\varphi_{N+1}^T\right)^{-1} \qquad (6.7)$$

ここで式変形に逆行列補題を用いて、

$$\Phi_{N+1}=\Phi_N-\Phi_N\varphi_{N+1}^T\times\left(I_m+\varphi_{N+1}\Phi_N\varphi_{N+1}^T\right)^{-1}\varphi_{N+1}\Phi_N \qquad (6.8)$$

を得た。したがって、新たなデータ (x_{N+1},y_{N+1}) が得られるたびに、新たな推定値 $\hat{\theta}_{N+1}$ はそれまでのデータによる推定値 $\hat{\theta}_N$ と Φ_N を用いることで更新できるので、改めてバッチ処理をする必要がない。オンラインで行う場合、重要なのは Φ_0 の選び方である。Φ_N の定義から、Φ_0 は定義できない。そこで Φ_N そのものを用いずに、小さな正の実数 ϵ を用いて $\tilde{\Phi}_0=\frac{1}{\epsilon}I_n$ とし、次のように更新すればよい。

$$\tilde{\theta}_{N+1}=\tilde{\theta}_N+\tilde{K}_{N+1}\left(y_{N+1}-\varphi_{N+1}\tilde{\theta}_N\right) \qquad (6.9)$$

$$\tilde{K}_{N+1}=\tilde{\Phi}_N\varphi_{N+1}^T\left(I_m+\varphi_{N+1}\tilde{\Phi}_N\varphi_{N+1}^T\right)^{-1} \qquad (6.10)$$

$$\tilde{\Phi}_{N+1}=\tilde{\Phi}_N-\tilde{K}_{N+1}\varphi_{N+1}\tilde{\Phi}_N \qquad (6.11)$$

ただし、

$$\tilde{\theta}_0=0,\quad \tilde{\Phi}_0=\frac{1}{\epsilon}I_n \qquad (6.12)$$

とした。

ここでこの $\tilde{\Phi}_N$ と Φ_N の関係は、Φ_N が正則になる N に対して、

$$\tilde{\Phi}_N^{-1}=\Phi_N^{-1}+\epsilon I_n=\sum_{i=1}^{N}\varphi_i^T\varphi_i+\epsilon I_n \qquad (6.13)$$

であることに注意する。Φ_N^{-1} はデータを得るたびに正定値対称行列の意味で大きくなるため、$\tilde{\Phi}_N$ と Φ_N は Taylor 展開すると、

$$\tilde{\Phi}_N=\Phi_N-\epsilon\Phi_N^2+O(\epsilon^2) \qquad (6.14)$$

であり、適切なデータ数 N が増えるにつれて Φ_N は正定値対称行列の意味で小さくなるので、ϵ の 1 次の項がゼロに近づくことが期待される。実際に ϵ の 1 次以上の項は N が増えると小さくなる項である。

6.3 実験方法

コード 11: 課題 14-1

```
import numpy as np

def X(i):
    a=np.array([[y[i+1],y[i],F]])
    return a

def Y(i):
    a=np.array([y[i+2]])
    return a

def theta(N):
    theta=np.array([[0],[0],[0]])
    phi=np.identity(n)/e

    for i in range(N+1):
        k=np.dot(np.dot(phi,X(i).T),1/(np.identity(m)+np.dot(np.dot(X(i),phi),X(i).T)))
        phi=phi-np.dot(np.dot(k,X(i)),phi)
        theta=theta+np.dot(k,Y(i)-np.dot(X(i),theta))

    return theta

M=2
D=1
K=3
dt=0.01
e=10**-6
n=3
m=1

N=10000
y=[0,0]
F=1
for k in range(N+1):
    y_k=(2-(D/M)*dt)*y[k+1]+(-1+(D/M)*dt-(K/M)*dt**2)*y[k]+(dt**2/M)*F+np.
        random.uniform(-1,1)
    y.append(y_k)

print(theta(N))
```

コード 12: 課題 14-2

```
import numpy as np
import math

def X(i):
    a=np.array([[y[i+1],y[i],F(i-2)]])
    return a

def Y(i):
    a=np.array([y[i+2]])
    return a

def theta(N):
    theta=np.array([[0],[0],[0]])
    phi=np.identity(n)/e

    for i in range(N+1):
        k=np.dot(np.dot(phi,X(i).T),1/(np.identity(m)+np.dot(np.dot(X(i),phi),X(i).T)))
        phi=phi-np.dot(np.dot(k,X(i)),phi)
        theta=theta+np.dot(k,Y(i)-np.dot(X(i),theta))

    return theta

```

```
23  M=2
24  D=1
25  K=3
26  dt=0.01
27  e=10**-6
28  n=3
29  m=1
30
31  N=10000
32  y=[0,0]
33
34  def F(k):
35      a=math.sin(math.pi*k/5)
36      return a
37
38  for k in range(N+1):
39      y_k=(2-(D/M)*dt)*y[k+1]+(-1+(D/M)*dt-(K/M)*dt**2)*y[k]+(dt**2/M)*F(k-2)+
            np.random.uniform(-1,1)
40      y.append(y_k)
41
42  print(theta(N))
```

6.4 結果

(1)

最小二乗推定誤差は

$$\hat{\theta}_N = (1.9959838, -0.9961456, -0.0031066)^T \tag{6.15}$$

(2)

$$\hat{\theta}_N = (1.99410106, -0.99426818, -0.0158544)^T \tag{6.16}$$

7　課題 16(Kalman フィルタ)

7.1　はじめに

次の離散時間ダイナミクスおよび観測方程式が与えられているとする。

$$\theta_k = 0.9\theta_{k-1} + v_k \tag{7.1}$$

$$y_k = 2\theta_k + w_k \tag{7.2}$$

ここで $\theta_k, y_k \in \mathbb{R}$ であり、v_k, w_k は互いに独立な $\mathcal{N}(0,1)$ に従う確率変数である。また、θ_0 は平均 3、分散 2 をもつ確率分布に従うとする。このとき、$\{y_l\}_{l=1}^{k}$ までを使った $\theta_k, k = 1, \cdots, 100$ の推定値 $\hat{\theta}_k$ を求め、θ_k を共に時間変化をプロットせよ。

7.2　原理・方法の詳細

7.2.1　Kalman フィルタ

次の線形方程式を考える。

$$\theta_k = a_k\theta_{k-1} + v_k \tag{7.3}$$

$$y_k = c_k\theta_k + w_k, \quad k = 1, 2, \cdots \tag{7.4}$$

ここで $a_k, c_k \in \mathbb{R}$ であり、$v_k, w_k \in \mathbb{R}$ は互いに独立な平均ゼロ、分散がそれぞれ σ_v^2, σ_w^2 の Gauss 分布に従って発生し、更に異なる時刻で独立である（白色性をもつ）とする。また、$c_k \neq 0, k = 1, 2, \cdots$ とする。θ_0 は平均 $\bar{\theta}$、分散 P を持つ分布に従って発生するとする（Gauss 性は不要）。ここで v_k や w_k および初期値 θ_0 は観測できないが、それらの分布の情報は既知であり、a_k, c_k も既知であるとする。観測値 y_k もセンサから各時刻で得られるものとするが、θ_k は潜在変数で直接知ることはできないとする。ここで扱う内容は各時刻 k までの観測値 $\{y_k\}_{l=1}^{k}$ を用いて θ_k を推定することである。これまでは推定すべきパラメータが確定的だったが、ここでは θ_k も確率変数になっている。

時刻 k までのデータによる推定値を $\hat{\theta}_k \in \mathbb{R}$ で表すとする。式 (6.9) の逐次最小二乗法では、1 ステップ前の推定値に、新たなデータ $(k, y_k) \equiv y_k$ による修正項が加えられる。式 (7.3) と式 (7.4) より、

$$y_k = c_k(a_k\theta_{k-1} + v_k) + w_k \tag{7.5}$$

となるため、前の時刻の推定値 $\hat{\theta}_{k-1}$ を a_k 倍したものに新たなデータから修正する項を考え、次の規則で推定値を更新することを考える。

$$\begin{aligned}\hat{\theta}_k &= a_k\hat{\theta}_{k-1} + F_k(y_k - c_k a_k\hat{\theta}_{k-1}) \\ &= a_k\hat{\theta}_{k-1} + a_k c_k F_k(\theta_{k-1} - \hat{\theta}_{k-1}) + F_k(w_k + c_k v_k)\end{aligned} \tag{7.6}$$

ここで $F_k \in \mathbb{R}$ は自由に設定できるパラメータである。このとき、推定誤差 $\theta_k - \hat{\theta}_k$ を最小二乗平均の意味で最小化する推定器を設計したい。推定誤差分散を $V_k := \mathbb{E}[(\theta_k - \hat{\theta}_k)^2]$ とすると、

$$\begin{aligned}V_k &= a_k^2(1 - c_k F_k)^2 V_{k-1} + (1 - c_k F_k)^2\sigma_v^2 + \sigma_w^2 F_k^2 \\ &= (a_k^2 c_k^2 V_{k-1} + c_k^2\sigma_v^2 + \sigma_w^2) \times \left(F_k - \frac{a_k^2 c_k V_{k-1} + c_k\sigma_v^2}{a_k^2 c_k^2 V_{k-1} + c_k^2\sigma_v^2 + \sigma_w^2}\right)^2 + a_k^2 V_{k-1} + \sigma_v^2 + \frac{(a_k^2 c_k V_{k-1} + c_k\sigma_v^2)^2}{a_k^2 c_k^2 V_{k-1} + c_k^2\sigma_v^2 + \sigma_w^2}\end{aligned} \tag{7.7}$$

より、

$$F_k = \frac{a_k^2 c_k V_{k-1} + c_k \sigma_v^2}{a_k^2 c_k^2 V_{k-1} + c_k^2 \sigma_v^2 + \sigma_w^2} \tag{7.8}$$

が推定誤差の分散 V_k の増加を最も抑えるパラメータになる。見やすくなるように変数 X_k を入れて整理すると、$\hat{\theta}_0 = \bar{\theta}$, $V_0 = P$ として、次の推定規則を導くことができる。

$$\hat{\theta}_k = a_k \hat{\theta}_{k-1} + F_k(y_k - c_k a_k \hat{\theta}_{k-1}) \tag{7.9}$$

$$X_k = a_k^2 V_{k-1} + \sigma_v^2 \tag{7.10}$$

$$V_k = X_k - \frac{c_k^2 X_k^2}{c_k^2 X_k + \sigma_w^2} = \frac{\sigma_w^2 X_k}{c_k^2 X_k + \sigma_w^2} \tag{7.11}$$

$$F_k = \frac{c_k X_k}{c_k^2 X_k + \sigma_w^2} \tag{7.12}$$

この推定則は Kalman フィルタと呼ばれ、システム制御や時系列解析、信号所処理等で広く用いられている。θ_k が n 次元、y_k が m 次元であっても、行列やベクトルの平方完成を使って同様に求められる。

7.3 実験方法

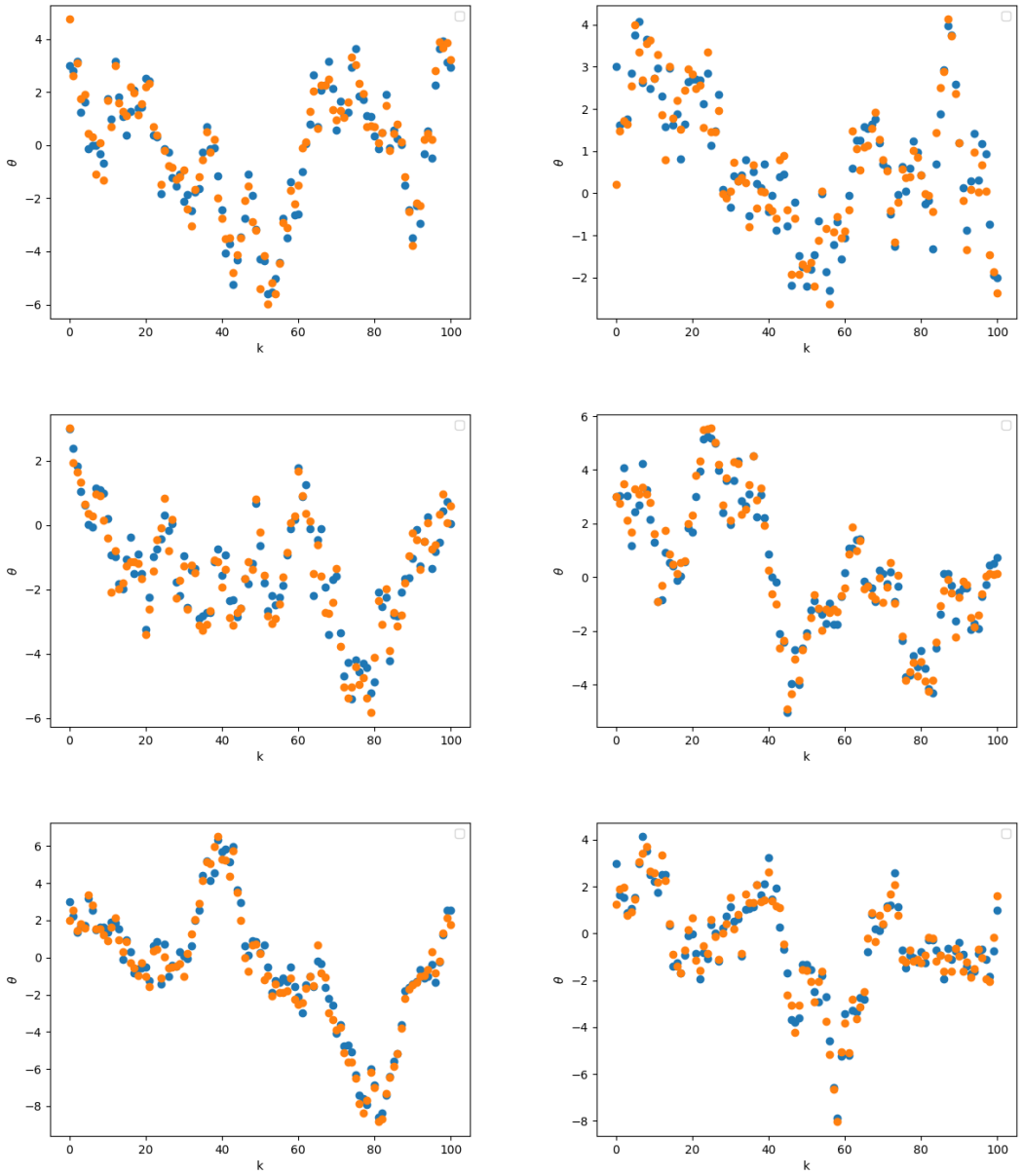
コード 13: 課題 16

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  N=100
5  ak=0.9
6  ck=2
7  sigma_v=1
8  sigma_w=1
9  mean=3
10 sigma=2
11
12 theta0=[np.random.normal(loc=3,scale=np.sqrt(2),size=None)]
13 y=[0]
14 for k in range(N):
15     theta_k1 = ak*theta0[k]+np.random.randn()
16     theta0.append(theta_k1)
17
18     y_k = ck*theta0[k+1]+np.random.randn()
19     y.append(y_k)
20
21 theta=[mean]
22 X=[0]
23 V=[sigma]
24 F=[0]
25 for k in range(N):
26     X_k=ak**2*V[k]+sigma_v
27     X.append(X_k)
28
29     V_k=(sigma_w*X[k+1])/(ck**2*X[k+1]+sigma_w)
30     V.append(V_k)
31
32     F_k=(ck*X[k+1])/(ck**2*X[k+1]+sigma_w)
33     F.append(F_k)
34
35     theta_k=ak*theta[k]+F[k+1]*(y[k+1]-ck*ak*theta[k])
36     theta.append(theta_k)
37
38 k=[]
```

```
39    for i in range(N+1):
40        k.append(i)
41
42    plt.scatter(k,theta)
43    plt.scatter(k,theta0)
44    plt.legend()
45    plt.xlabel('k')
46    plt.ylabel(r'$\theta$')
47    plt.show()
```

7.4 結果

オレンジが真値、青が推定値である。



8　課題 18(交互最小二乗法)

8.1　はじめに

$x_i \in [-1,1]$, $i = 1, \cdots, 10000$ に対して、次の方程式で定められる入出力データが与えられているとする。

$$y_i = \alpha^T \varphi(x_i)\beta + w_i \tag{8.1}$$

ここで $\alpha \in \mathbb{R}^2$, $\beta \in \mathbb{R}^3$, w_i は区間 $[-1,1]$ の一様分布であるとし、

$$\varphi(x) = \begin{bmatrix} 1 & x & x^2 \\ x & x^2 & x^3 \end{bmatrix} \tag{8.2}$$

の場合を考える。10000 組の $\{(x_i, y_i)\}_{i=1}^{10000}$ から、交互最小二乗法を用いて α, β を推定せよ。このとき、10 組の異なる初期値を用意し、最もよかったパラメータとそのときに使った初期値を示せ。

8.2　原理・方法の詳細

8.2.1　交互最小二乗法

関数 $f(\theta, x)$ が $\theta = (\alpha, \beta)^T$ に関して双線形であるとは、例えば

$$f(\theta, x) = \alpha\beta x + \beta \tag{8.3}$$

のように、推定したいパラメータ同士がかけ算の形になってしまっているというものである。$\alpha\beta$ を改めて別の変数に置き換えて推定してもよいが、それぞれのパラメータに意味のある場合もある。このような関数 $f(\theta, x)$ は、パラメータ $\theta \in \mathbb{R}^n$ を 2 つに分割すれば、分割したパラメータ $\alpha \in \mathbb{R}^{n_1}$, $\beta \in \mathbb{R}^{n_2}$, $n_1 + n_2 = n$ それぞれに対して線形になる。そこで、$f(\theta, x)$ を改めて $g(\alpha, \beta, x)$ とおいて、次のようにパラメータを求める。

Step 1 初期値 $\hat{\alpha}_0$, $\hat{beta}_0$ を設定する。

Step 2 $j = 1, \cdots$ に対し、

Step 2-1 $\beta = \hat{\beta}_{j-1}$ に固定し、α を最小化する。

$$\hat{\alpha}_j = \underset{\alpha \in \mathbb{R}^{n_1}}{\arg\min} \sum_{i=1}^{N} \|y_i - g(\alpha, \hat{\beta}_{j-1}, x_i)\|^2 \tag{8.4}$$

Step 2-2 $j = 1, \cdots$ に対し、$\alpha = \hat{\alpha}_j$ を固定し、β を最小化する。

$$\hat{\beta}_j = \underset{\beta \in \mathbb{R}^{n_2}}{\arg\min} \sum_{i=1}^{N} \|y_i - g(\hat{\alpha}_j, \beta, x_i)\|^2 \tag{8.5}$$

Step 3 収束判定条件（例えば事前に設定した $\varepsilon > 0$ に対して $(\|\alpha_{j-1} - \alpha_j\|^2 + \|\beta_{j-1} - \beta_j\|^2) < \varepsilon$）を満たすかを判定し、満たさなければ j を 1 つ繰り上げて Step 2 に戻る。

同じデータを使ったまま、最小二乗法を繰り返して解く問題なる。各最小化問題は式 (1.4) の形で解けばよい。交互に最初か門田を解くため、交互最小二乗法と呼ばれる。アルゴリズムの性質から、$j = 1, \cdots$ に対して

$$\sum_{i=1}^{N} \|y_i - g(\hat{\alpha}_j, \hat{\beta}_j, x_i)\|^2 \geq \sum_{i=1}^{N} \|y_i - g(\hat{\alpha}_{j+1}, \hat{\beta}_{j+1}, x_i)\|^2 \tag{8.6}$$

なる関係も成り立つ。したがって交互最小二乗法は「悪くならない」手法である。しかし、一般には局所最適解への収束は必ずしも保証されるとは陰らないため、初期値の工夫などが必要である。

8.3 実験方法

コード 14: 課題 18

```
1  import numpy as np
2  import pandas as pd
3
4  df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\18\\mmse_kadai13.xls',header
       =None)
5
6  x=df.iloc[:,0:1].to_numpy()
7  y=df.iloc[:,1:2].to_numpy()
8
9  theta_a=np.array([[100,100]])
10 theta_b=np.array([[0.5,-1,2]])
11
12 def XA(i,j):
13     a=np.array([[1,x[i][0],x[i][0]**2],[x[i][0],x[i][0]**2,x[i][0]**3]])
14     a=np.dot(np.reshape(theta_a[j],(1,2)),a)
15     return a
16
17 def XB(i,j):
18     a=np.array([[1,x[i][0],x[i][0]**2],[x[i][0],x[i][0]**2,x[i][0]**3]])
19     a=np.dot(np.reshape(theta_b[j],(1,3)),a.T)
20     return a
21
22 def Y(i):
23     a=np.array([y[i]])
24     return a
25
26 def phiA(j):
27     a=0
28     for i in range(N):
29         a+=np.dot(XA(i,j).T,XA(i,j))
30     a=np.linalg.inv(a)
31     return a
32
33 def phiB(j):
34     a=0
35     for i in range(N):
36         a+=np.dot(XB(i,j).T,XB(i,j))
37     a=np.linalg.inv(a)
38     return a
39
40 def A(j):
41     a=phiB(j-1)
42
43     b=0
44     for i in range(N):
45         b+=np.dot(XB(i,j-1).T,Y(i))
46
47     c=np.dot(a,b)
48
```

```
49          global theta_a
50          theta_a=np.append(theta_a,np.reshape(c,(1,2)),axis=0)
51
52      def B(j):
53          a=phiA(j)
54
55          b=0
56          for i in range(N):
57              b+=np.dot(XA(i,j).T,Y(i))
58
59          c=np.dot(a,b)
60
61          global theta_b
62          theta_b=np.append(theta_b,np.reshape(c,(1,3)),axis=0)
63
64      N=10000
65      e=10**-10
66      j=1
67      while True:
68          A(j)
69          B(j)
70          if np.linalg.norm(theta_a[j-1]-theta_a[j])**2+np.linalg.norm(theta_b[j-1]-theta_b[j])
                **2<e:
71              print(theta_b[0],theta_a[j],theta_b[j])
72              break
73          else:
74              j+=1
```

8.4 結果

表 1: 交互最小二乗法

$\hat{\beta}_0$	$\hat{\alpha}$	$\hat{\beta}$
$[1, 0, 0]$	$[2.06719861, -4.13556619]$	$[0.24499178, -0.48215047, 0.96763471]$
$[0, 1, 0]$	$[-3.20033281, 6.40244898]$	$[-0.1582484, 0.31143611, -0.62502996]$
$[0, 0, 1]$	$[3.93930386, -7.88060664]$	$[0.12856403, -0.25301161, 0.50779759]$
$[0.5, 0, 0]$	$[4.13441813, -8.27110759]$	$[0.12249568, -0.24107327, 0.48382]$
$[0, -1, 0]$	$[3.20033281, -6.40244898]$	$[0.1582484, -0.31143611, 0.62502996]$
$[0, 0, 2]$	$[1.96966309, -3.9402901]$	$[0.25712757, -0.50601859, 1.01560139]$
$[0.5, -1, 0]$	$[1.89153249, -3.7839904]$	$[0.26774835, -0.5269199, 1.05755136]$
$[0.5, 0, 2]$	$[1.41594647, -2.83257667]$	$[0.35767974, -0.70390011, 1.41276603]$
$[0, -1, 2]$	$[1.24477876, -2.49016226]$	$[0.40686351, -0.80069285, 1.60703033]$
$[0.5, -1, 2]$	$[1.0017542, -2.00407958]$	$[0.50555933, -0.99495634, 1.99678423]$

最もよかったパラメータは

$$\hat{\alpha} = [1.0017542, -2.00407958] \tag{8.7}$$

$$\hat{\beta} = [0.50555933, -0.99495634, 1.99678423] \tag{8.8}$$

このときの β の初期値は

$$\hat{\beta}_0 = [0.5, -1, 2] \tag{8.9}$$

9　課題 19(K-クラスタリング)

9.1　はじめに

与えられたデータ $x_i \in \mathbb{R}^2, i = 1, \cdots, 1000$ を 3 つのクラスターに分類せよ。異なる初期値を 10 組以上選び、最も良さそうな分類をその理由とともに示せ。

9.2　原理・方法の詳細

9.2.1　K-平均法

K-平均法は、与えられたデータを K 個のクラスタに分類する手法であり、似たようなデータを集める代表的な方法である。$\{x_i\}_{i=1}^N \subset \mathbb{R}^p$ がデータとして与えられているものとする。これを K 個のクラスタに分類したい。ここで K はこちらで与える正整数とする。K をどのように決めるかの議論はここで行わず、与えられているものとする。$\nu := \{1, \cdots, N\}$ とし、ν の空でない部分集合 $\nu_1, \cdots, \nu_K$ を、$\cup_{l=1}^K \nu_l = \nu$ かつ $\nu_l \cap \nu_k = \emptyset, l \neq k$ を満たすようにとる。この ν_l がクラスタを表し、そのクラスタを特徴づける量を次で定める。

$$\mu(\nu_l) := \arg\min_{\mu \in \mathbb{R}^p} \sum_{i_l \in \nu_l} \|x_{i_l} - \mu\|^2, \quad l = 1, \cdots, K \tag{9.1}$$

これは、次のようにクラスタの中心を表す。

$$\min_{\{\nu_1, \cdots, \nu_K\}} \sum_{l=1}^K \sum_{i_l \in \nu_l} \|x_{i_l} - \mu(\nu_l)\|^2 \tag{9.2}$$

ここで

$$r_{i,l} := \begin{cases} 1, & i \in \nu_l \\ 0, & i \notin \nu_l \end{cases} \tag{9.3}$$

を導入すれば、式 (9.2) は

$$\sum_{l=1}^K \sum_{i=1}^n r_{i,l} \|x_i - \mu(\nu_l)\|^2 \tag{9.4}$$

となる。x_i がどのクラスタに入るかを更新するには、ν_l をどう変更するかの問題になるが、これは $r_{i,l}$ と $\mu(\nu_l)$ の両方を変更しなければならないので難しい。そこで現在のクラスタ中心 $\mu(\nu_l)$ を固定して μ_l と表し、$r_{i,l}$ を更新する。$r_{i,l} = 1$ は、x_i が ν_l に所属することを意味したが、改めて x_i がどこのクラスタに所属するかは、クラスタの代表点の中で最も近いものに変更する。

$$r_{i,l} = \begin{cases} 1, & l = \arg\min_{k=1,\cdots,K} \|x_i - \mu_k\|^2 \\ 0, & otherwise \end{cases} \tag{9.5}$$

各クラスタに所属するデータが変更されたので、$\mu(\nu_l)$ も計算し直す必要があり、交互に最適化問題を解くことになる。アルゴリズムをまとめると、次の通りである。

Step 1 初期値 $\mu_l \in \mathbb{R}^p, l = 1, \cdots, K$ を設定する。

Step 2 $j = 1, \cdots$ に対し、

Step 2-1 式 (9.5) より $r_{i,l}^{(j)}, i = 1, \cdots, N, l = 1, \cdots, K$ を定める。

Step 2-2 次の式より $\mu_l^{(j)}, l = 1, \cdots, K$ を定める。

$$\mu_l^{(j)} = \frac{1}{\sum_{i=1}^{N} r_{i,l}} \sum_{i=1}^{N} r_{i,l} x_i \tag{9.6}$$

Step 3 収束判定条件（例えば事前に設定した $\varepsilon > 0$ に対して $\max_l \|\mu_l^{(j-1)} - \mu_l^{(j)}\|^2 < \varepsilon$）を満たすかを判定し、満たさなければ j を 1 つ繰り上げて Step 2 に戻る。

$U := [\mu_1, \cdots, \mu_K]$, $R = \{r_{i,l}\}$, $X = [x_1, \cdots, x_N]$ と置くと、K 平均法は次のコスト関数の最小化問題となる。

$$\|X = UR\|^2 \tag{9.7}$$

R が 0 か 1 の要素しか持たないことから、連続最適化と離散最適化の両方の性質を持った難しめの最小化問題になっており、大局的最適解を求められる保証はない。また、初期値の選び方が非常に重要になる。

9.3 実験方法

コード 15: 課題 19

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  import sys
5
6  df=pd.read_excel('C:\\Users\\Takimoto\\Desktop\\kadai\\19\\mmse_kadai14.xls',header
       =None)
7
8  x=df.iloc[:,0:2].to_numpy()
9
10 mu=np.array([[2.0,0.0],[0.0,−2.0],[−5.0,0.0]])
11 N=1000
12 e=10**−6
13
14 while True:
15     mu0=mu
16     r=np.ones((N,3))
17     for l in range(3):
18         a=0
19         b=0
20         for i in range(N):
21             for k in range(3):
22                 if np.linalg.norm(x[i]−mu[l])<=np.linalg.norm(x[i]−mu[k]):
23                     continue
24                 else:
25                     r[i][l]=0
26                     break
27             a+=r[i][l]
28             b+=r[i][l]*x[i]
29         mu[l]=b/a
30
31     c=1
32     for k in range(3):
33         if np.linalg.norm(mu0[k]−mu[k])**2<e:
34             continue
35         else:
36             c=0
37             break
```

```
38
39          if c==1:
40              x1_0=[]
41              x1_1=[]
42              x2_0=[]
43              x2_1=[]
44              x3_0=[]
45              x3_1=[]
46              for i in range(N):
47                  if r[i][0]==1:
48                      x1_0.append(x[i][0])
49                      x1_1.append(x[i][1])
50                  elif r[i][1]==1:
51                      x2_0.append(x[i][0])
52                      x2_1.append(x[i][1])
53                  elif r[i][2]==1:
54                      x3_0.append(x[i][0])
55                      x3_1.append(x[i][1])
56          else:
57              continue
58
59          break
60
61      print(len(x1_0),len(x2_0),len(x3_0))
62
63      plt.scatter(x1_0,x1_1,label='$\mu$1')
64      plt.scatter(x2_0,x2_1,label='$\mu$2')
65      plt.scatter(x3_0,x3_1,label='$\mu$3')
66      plt.legend()
67      plt.show()
```

9.4 結果

各クラスタに所属するデータの数が均等になるとき、良い分類であると考える。

表 2: K-クラスタリング

μ_1^0	μ_2^0	μ_3^0	μ_1 の要素数	μ_2 の要素数	μ_3 の要素数
$(0,0)$	$(0,-2)$	$(-5,0)$	301	313	324
$(2,0)$	$(0,0)$	$(-5,0)$	218	251	323
$(2,0)$	$(0,-2)$	$(0,0)$	208	343	303
$(2,1)$	$(0,-2)$	$(-5,0)$	300	357	340
$(2,0)$	$(1,-2)$	$(-5,0)$	307	340	343
$(2,0)$	$(0,-2)$	$(-5,1)$	300	357	342
$(2,-1)$	$(0,-2)$	$(-5,0)$	270	383	341
$(2,0)$	$(-1,-2)$	$(-5,0)$	311	356	332
$(2,0)$	$(0,-2)$	$(-5,-1)$	310	354	335
$(2,0)$	$(0,-2)$	$(-5,0)$	304	353	343

最も良さそうな分類は $\mu_1^0 = (2,0)$, $\mu_2^0 = (1,-2)$, $\mu_3^0 = (-5,0)$

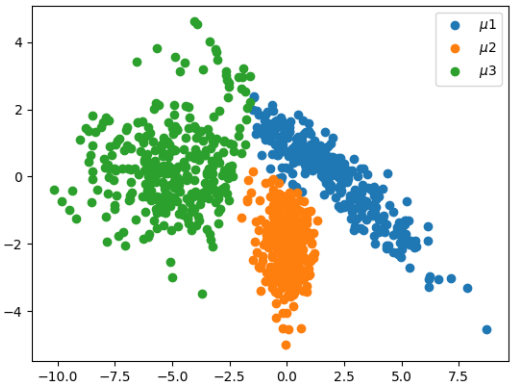


図 11: K-クラスタリング