SML HW2

29-176004 奥村 恭平 *†

May 12, 2018

宿題1

$\Sigma := \sigma^2 I$ のガウスモデルの最尤推定量が，次のようになることを示す．

$$\widehat{\mu}_{ML} = \frac{1}{n}\sum_{i=1}^{n} x_i,\ \widehat{\sigma}^2_{ML} = \frac{1}{nd}\sum_{i=1}^{d}\sum_{j=1}^{n}(x_j^{(i)} - \widehat{\mu}_{ML}^{(i)})(x_j^{(i)} - \widehat{\mu}_{ML}^{(i)}) \tag{1}$$

対数尤度関数 $l(\mu, \sigma^2)$ は，以下のようになる．

$$l(\mu, \sigma^2) := \sum_{i=1}^{n} \log q(x_i; \mu, \sigma^2) \propto \frac{nd}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^\top (x_i - \mu)$$

最尤推定量は以下の方程式系を満たす．

$$\begin{cases} \dfrac{\partial l}{\partial \mu} = 0 & (2) \\ \dfrac{\partial l}{\partial \sigma^2} = 0 & (3) \end{cases}$$

これを解くと，

$$\begin{aligned} (2) &\iff \frac{1}{2\sigma^2}\sum_{i=1}^{n} 2(x_i - \mu) = 0 \\ &\iff \sum_{i=1}^{n}(x_i - \mu) = 0 \\ &\therefore\ \widehat{\mu}_{ML} = \frac{1}{n}\sum_{i=1}^{n} x_i \end{aligned}$$

$$\begin{aligned} (3) &\iff -\frac{nd}{2}\frac{1}{\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_{j=1}^{n}(x_j - \mu)(x_j - \mu)^\top = 0 \\ &\iff \sigma^2 = \frac{1}{nd}\sum_{j=1}^{n}(x_j - \mu)(x_j - \mu)^\top \\ &\therefore\ \widehat{\sigma}^2_{ML} = \frac{1}{nd}\sum_{i=1}^{d}\sum_{j=1}^{n}(x_j^{(i)} - \widehat{\mu}_{ML}^{(i)})(x_j^{(i)} - \widehat{\mu}_{ML}^{(i)}) \end{aligned}$$

となり，所定の結果 (1) を得る． □

*E-mail: kyohei.okumura@gmail.com
†東京大学大学院 経済学研究科 M2

宿題2

$\mu_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$, $\mu_2 = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$, $\Sigma_1 = \Sigma_2 = \Sigma = \begin{pmatrix} 9 - 8\cos^2\beta & 8\sin\beta\cos\beta \\ 8\sin\beta\cos\beta & 9 - 8\sin^2\beta \end{pmatrix}$, $p(y=1) = p(y=2) = 1/2$

まず,

$$\Sigma^{-1} = \begin{pmatrix} 1 - \frac{8}{9}\sin^2\beta & -\frac{8}{9}\sin\beta\cos\beta \\ -\frac{8}{9}\sin\beta\cos\beta & 1 - \frac{8}{9}\cos^2\beta \end{pmatrix}$$

である．データから決定境界を推定した場合の推定された決定境界の式は以下のようになる．

$$\widehat{a}^\top x + \widehat{b} = 0, \text{where } \widehat{a} := \widehat{\Sigma}^{-1}(\widehat{\mu_1} - \widehat{\mu_2}), \widehat{b} := -\frac{1}{2}(\widehat{\mu_1}^\top \widehat{\Sigma}^{-1}\widehat{\mu_1} - \widehat{\mu_2}^\top \widehat{\Sigma}^{-1}\widehat{\mu_2}) + \log\frac{n_1}{n_2}$$

サンプルサイズ n が十分大きいとき，大数の法則・最尤推定量の一致性より，

$$\widehat{a} \to^p \Sigma^{-1}(\mu_1 - \mu_2) = \begin{pmatrix} 4 - \frac{32}{9}\sin^2\beta \\ -\frac{32}{9}\sin\beta\cos\beta \end{pmatrix}$$

$$\widehat{b} \to^p -\frac{1}{2}(\mu_1^\top \Sigma^{-1}\mu_1 - \mu_2^\top \Sigma^{-1}\mu_2) + \log\frac{n \cdot p(y=1)}{n \cdot p(y=2)} = 0$$

よって，決定境界の式は，漸近的に

$$\left(4 - \frac{32}{9}\sin^2\beta\right)x_1 - \left(\frac{32}{9}\sin\beta\cos\beta\right)x_2 = 0$$

となる．

宿題3

Python を用いて実装した．$n := 600, \alpha := 0.1$ としてシミュレーションを行った際の図を載せる．黒い直線が，データから推定された決定境界を表している．

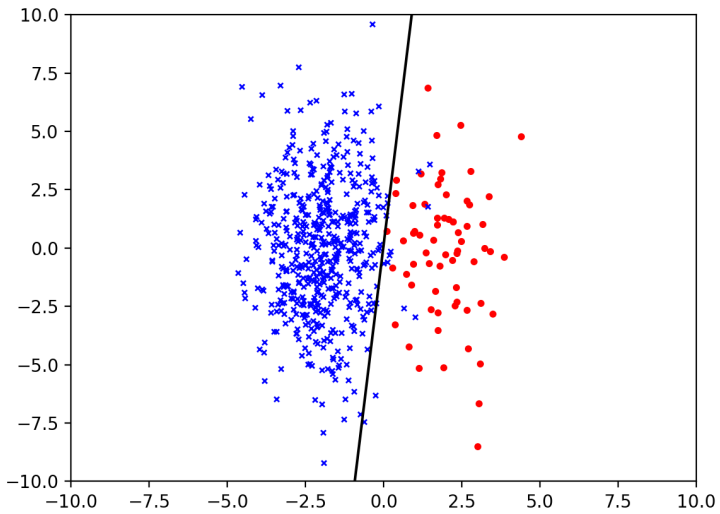

Figure 1: $n := 600, \alpha := 0.1$

以下，用いたコード．

ソースコード 1: Fisher

```
import numpy as np
from numpy.random import randn, rand
import matplotlib.pyplot as plt


def gen(n=600, alpha=0.1):
# data generation
    n1 = np.sum(rand(n, 1) < alpha)
    n2 = n - n1
    x1 = np.concatenate([randn(1, n1) + 2, 3 * randn(1, n1)])
    x2 = np.concatenate([randn(1, n2) - 2, 3 * randn(1, n2)])

    return x1, x2


def plot(x1, x2, line=None):
    x, y = x1
    plt.plot(x, y, 'ro', ms=3, label='class1')
    x, y = x2
    plt.plot(x, y, 'bx', ms=3, label='class2')

    if not (line is None):
            plt.plot(line[0], line[1], 'k-', ms=5)

    plt.xlim(-10, 10)
    plt.ylim(-10, 10)

    plt.show()


def fisher(x1, x2):
    x1 = x1.T
    x2 = x2.T
    n1 = x1.shape[0]
    n2 = x2.shape[0]
    n = n1 + n2

    # dimension
    DIM = x1.shape[1]


    # average
    mean1 = np.mean(x1, axis=0)
    mean1 = mean1.reshape(DIM, 1)
    mean2 = np.mean(x2, axis=0)
    mean2 = mean2.reshape(DIM, 1)

    # covariance
    sample_cov_1 = np.zeros((DIM, DIM))
    sample_cov_2 = np.zeros((DIM, DIM))
    for x_i in x1:
        x_i = x_i.reshape(DIM, 1)
        sample_cov_1 += np.dot((x_i - mean1), (x_i - mean1).T)
        sample_cov_1 = sample_cov_1 / n1
    for x_i in x2:
        x_i = x_i.reshape(DIM, 1)
        sample_cov_2 += np.dot((x_i - mean2), (x_i - mean2).T)
        sample_cov_2 = sample_cov_2 / n2
    sample_cov = (n1/n) * sample_cov_1 + (n2/n) * sample_cov_2

    sample_cov_inv = np.linalg.inv(sample_cov + 0.000001 * np.eye(DIM))

    a = np.dot(sample_cov_inv, (mean1 - mean2))
    if (n1 - n2 > 1e-10):
        b = -0.5 * (np.dot(mean1.T, np.dot(sample_cov_inv, mean1)) - np.dot(
```

```
            mean2.T, np.dot(sample_cov_inv, mean2))) + np.log(n1/n2)
    else:
        b = -0.5 * (np.dot(mean1.T, np.dot(sample_cov_inv, mean1)) - np.dot(
            mean2.T, np.dot(sample_cov_inv, mean2)))
    b = b.reshape(1)

    return a, b



def line_est(a, b, x):
    if abs(a[1]) > 1e-10:
        c = - a[0]/a[1]
        d = - b/a[1]
    return [x, c*x+d]


if __name__ == '__main__':
    x1, x2 = gen(n=600, alpha=0.1)
    a, b = fisher(x1, x2)
    x = np.linspace(-10, 10, 3000)
    l = line_est(a, b, x)
    plot(x1, x2, l)
```

宿題 4

Octave で末尾のコードを実行し，コンソールで混同行列を表示した．その画像を添付する．

```
>> C
C =

   199     0     0     0     1     0     0     0     0
     0   168    10     6     0     3     4     8     1
     0     0   184     0     6     0     2     7     1
     2     2     0   181     0     2     0     3    10
     0     1    20     4   168     0     0     3     4
     1     3     0     2     3   188     0     3     0
     2     0     1     4     1     0   182     0    10
     0     0    17     4     7     0     1   168     3
     1     0     0     6     0     0     7     2   184
```

Figure 2: 混同行列

(python のコードを貼る設定で matlab のコードを貼っているため，latex での表示が少しおかしくなっています．)

ソースコード 2: digits

```
clear all
load digit.mat X T
[d,n,nc] = size(X);

nc = 9;
S = zeros(d,d);
for c = 1:nc
  mu(:,c) = mean(X(:,:,c), 2);
  S = S + cov(X(:,:,c)')/nc;
end
invS␣=␣inv(S);

for␣ct␣=␣1:nc
␣␣for␣c␣=␣1:nc
```

```
    muc = mu(:,c);
    t=T(:,:,ct);
    p(ct,:,c) = t' * invS * muc - muc' * invS * muc/2;
  end
end

[pmax P] = max(p,[],3);
for ct = 1:nc
  for c = 1:nc
    C(ct, c) = sum(P(ct,:)==c);
  end
end
```