# 統計的機械学習
# 第二回　レポート

37-196360　森田涼介

2019 年 4 月 23 日

# 宿題 1

入力 $x \in \mathbb{R}^d$，期待値 $\mu \in \mathbb{R}^d$，共分散行列 $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ のガウスモデルは次のように表される。

$$q(\boldsymbol{x};\, \boldsymbol{\mu},\, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}\det(\boldsymbol{\Sigma})^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{1}$$

標本 $\{x_i\}_{i=1}^n$ に対して，このモデルの最尤推定量 $\hat{\boldsymbol{\mu}}_{\mathrm{ML}}$, $\hat{\boldsymbol{\Sigma}}_{\mathrm{ML}}$ を求める。対数尤度は次のようになる。

$$\begin{aligned}
\log L(\boldsymbol{\mu},\, \boldsymbol{\Sigma}) &= \sum_{i=1}^{n} q(\boldsymbol{x}_i;\, \boldsymbol{\mu},\, \boldsymbol{\Sigma}) & (2)\\
&= \sum_{i=1}^{n}\left(-\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(\det(\boldsymbol{\Sigma})) - \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})\right) & (3)\\
&= -\frac{nd}{2}\log(2\pi) - \frac{n}{2}\log(\det(\boldsymbol{\Sigma})) - \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}) & (4)\\
& & (5)
\end{aligned}$$

ここで，各要素の分散が等しく，共分散が 0 となる，すなわち $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$ であるガウスモデルを考えると，

$$q(\boldsymbol{x};\, \boldsymbol{\mu},\, \boldsymbol{\Sigma}) = \frac{1}{(2\pi\sigma^2)^{d/2}}\exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{\mu})}{2\sigma^2}\right) \tag{6}$$

となる。このとき，$\det(\boldsymbol{\Sigma}) = \sigma^{2d}$，$\Sigma^{-1} = (1/\sigma^2)\boldsymbol{I}$ から，対数尤度は，

$$\begin{aligned}
\log L(\boldsymbol{\mu},\, \sigma) &= -\frac{nd}{2}\log(2\pi) - nd\log(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})^{\mathrm{T}}(\boldsymbol{x}_i-\boldsymbol{\mu}) & (7)\\
& & (8)
\end{aligned}$$

$\boldsymbol{\mu}, \sigma$ でそれぞれ偏微分して，

$$\frac{\partial}{\partial \mu}(\log L(\boldsymbol{\mu},\, \sigma)) = \frac{1}{\sigma^2}\left(-n\boldsymbol{\mu} + \sum_{i=1}^{n}\boldsymbol{x}_i\right) \tag{9}$$

$$\frac{\partial}{\partial \sigma}(\log L(\boldsymbol{\mu},\, \sigma)) = -\frac{nd}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})^{\mathrm{T}}(\boldsymbol{x}_i-\boldsymbol{\mu}) \tag{10}$$

これらをそれぞれ $\boldsymbol{0}$, 0 と置くことで，$L$ に最大値を与える $\boldsymbol{\mu}$, $\sigma$ が得られる。つまり，

$$\hat{\boldsymbol{\mu}}_{\mathrm{ML}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \tag{11}$$

$$\begin{aligned}
\hat{\sigma}^2_{\mathrm{ML}} &= \frac{1}{nd}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})^{\mathrm{T}}(\boldsymbol{x}_i-\boldsymbol{\mu}) & (12)\\
&= \frac{1}{nd}\sum_{i=1}^{n}\sum_{j=1}^{d}\left(x_i^{(j)} - \hat{\mu}_{\mathrm{ML}}^{(j)}\right)^2 & (13)
\end{aligned}$$

# 宿題 2

入力次元 $d=2$，カテゴリ数 $c=2$，各カテゴリの事前分布 $p(y=1)=p(y=2)=1/2$ の分類問題を考える。また，各カテゴリの条件付き確率 $p(x|y)$ は正規分布であるとし，その期待値と共分散行列は次のようになるとする。

$$\mu_1=\begin{bmatrix}2\\0\end{bmatrix},\qquad \mu_2=\begin{bmatrix}-2\\0\end{bmatrix} \tag{14}$$

$$\Sigma_1=\Sigma_2=\Sigma=\begin{bmatrix}9-8\cos^2\beta & 8\sin\beta\cos\beta\\ 8\sin\beta\cos\beta & 9-8\sin^2\beta\end{bmatrix} \tag{15}$$

線形判別分析に基づいて決定境界を求める。決定境界は，

$$p(y=1|x)=p(y=2|x) \tag{16}$$

で与えられる。ここで，各カテゴリの分散共分散行列が等しいので，

$$\log\ p(y|x)=\mu_y{}^{\mathrm{T}}\Sigma^{-1}x-\frac{1}{2}\mu_y{}^{\mathrm{T}}\Sigma^{-1}\mu_y+\log p_y+C'' \tag{17}$$

となる。よって

$$\log p(y=1|x)=\mu_1{}^{\mathrm{T}}\Sigma^{-1}x-\frac{1}{2}\mu_1{}^{\mathrm{T}}\Sigma^{-1}\mu_1+\log p_1+C'' \tag{18}$$

$$\log p(y=2|x)=\mu_2{}^{\mathrm{T}}\Sigma^{-1}x-\frac{1}{2}\mu_2{}^{\mathrm{T}}\Sigma^{-1}\mu_2+\log p_2+C'' \tag{19}$$

式 (16) と $p(y=1)=p(y=2)=1/2$ とから，辺々引いて，

$$(\mu_1{}^{\mathrm{T}}-\mu_2{}^{\mathrm{T}})\Sigma^{-1}x-\frac{1}{2}(\mu_1{}^{\mathrm{T}}\Sigma^{-1}\mu_1-\mu_2{}^{\mathrm{T}}\Sigma^{-1}\mu_2)=0 \tag{20}$$

を得る。いま，$\Sigma$ の逆行列を求めると，

$$\Sigma^{-1}=\frac{1}{9}\begin{bmatrix}9-8\sin^2\beta & -8\cos\beta\sin\beta\\ -8\cos\beta\sin\beta & 9-8\cos^2\beta\end{bmatrix} \tag{21}$$

となるので，

$$(\mu_1{}^{\mathrm{T}}-\mu_2{}^{\mathrm{T}})\Sigma^{-1}=\begin{bmatrix}4 & 0\end{bmatrix}\cdot\frac{1}{9}\begin{bmatrix}9-8\sin^2\beta & -8\cos\beta\sin\beta\\ -8\cos\beta\sin\beta & 9-8\cos^2\beta\end{bmatrix} \tag{22}$$

$$=\frac{4}{9}\begin{bmatrix}9-8\sin^2\beta & -8\cos\beta\sin\beta\end{bmatrix} \tag{23}$$

$$\mu_1{}^{\mathrm{T}}\Sigma^{-1}\mu_1=\begin{bmatrix}2 & 0\end{bmatrix}\cdot\frac{1}{9}\begin{bmatrix}9-8\sin^2\beta & -8\cos\beta\sin\beta\\ -8\cos\beta\sin\beta & 9-8\cos^2\beta\end{bmatrix}\cdot\begin{bmatrix}2\\0\end{bmatrix}=\frac{4}{9}(9-8\sin^2\beta) \tag{24}$$

$$\mu_2{}^{\mathrm{T}}\Sigma^{-1}\mu_2=\begin{bmatrix}-2 & 0\end{bmatrix}\cdot\frac{1}{9}\begin{bmatrix}9-8\sin^2\beta & -8\cos\beta\sin\beta\\ -8\cos\beta\sin\beta & 9-8\cos^2\beta\end{bmatrix}\cdot\begin{bmatrix}-2\\0\end{bmatrix}=\frac{4}{9}(9-8\sin^2\beta) \tag{25}$$

となる。これらを式 (20) に代入して，

$$\frac{4}{9}\Big((9-8\sin^2\beta)x^{(1)}-8\cos\beta\sin\beta x^{(2)}\Big)-\frac{1}{2}\left(\frac{4}{9}(9-8\sin^2\beta)-\frac{4}{9}(9-8\sin^2\beta)\right)=0 \tag{26}$$

$$(9-8\sin^2\beta)x^{(1)}-8\cos\beta\sin\beta x^{(2)}=0 \tag{27}$$

$$x^{(1)}=-\frac{8\cos\beta\sin\beta}{9-8\sin^2\beta}x^{(2)} \tag{28}$$

よって，決定境界は，

$$x^{(1)} = -\frac{8\cos\beta\sin\beta}{9-8\sin^2\beta}x^{(2)} \tag{29}$$

である。

# 宿題 3

各カテゴリの分散共分散行列が等しいので，対数事後確率は以下のように表される。

$$\log p(y|x) = {\mu_y}^{\mathrm{T}}\Sigma^{-1}x - \frac{1}{2}{\mu_y}^{\mathrm{T}}\Sigma^{-1}\mu_y + \log p_y + C'' \tag{30}$$

いま，2 値分類問題を考えるから，ある $x$ について $\log p(y=1|x) > \log p(y=2|x)$ となるときに $x$ はカテゴリ $\hat{y}=1$ に，$\log p(y=1|x) < \log p(y=2|x)$ となるときに $x$ はカテゴリ $\hat{y}=2$ に分類される。

これを実装した結果が 6 ページの listing 1 である。この実行結果より，$x_1$ と $x_2$ は，それぞれ 100% の正解率で分類できることがわかる。

# 宿題 4

線形判別分析に基づき 0–9 までの 10 クラスの手書き文字認識を行う。プログラムは 8 ページの listing2 に示した。

以下に結果を示す。混同行列は表 1 のようになり，また，各カテゴリごとの正解率等は表 2 のようになった。

なお，マハラノビス距離に基づく分類を行ってみたところ，train データの分散行列の行列式が 10 となり逆行列が計算できなかったため，こちらは断念した。

表 1: 混同行列

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 192 | 0 | 0 | 3 | 0 | 0 | 4 | 0 | 1 | 0 |
| 1 | 0 | 199 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 169 | 8 | 8 | 1 | 2 | 4 | 8 | 0 |
| 3 | 1 | 0 | 0 | 182 | 1 | 5 | 0 | 2 | 8 | 1 |
| 4 | 0 | 2 | 2 | 0 | 182 | 0 | 1 | 0 | 3 | 10 |
| 5 | 4 | 0 | 0 | 21 | 4 | 162 | 1 | 0 | 4 | 4 |
| 6 | 3 | 1 | 2 | 0 | 1 | 5 | 185 | 0 | 3 | 0 |
| 7 | 1 | 2 | 0 | 1 | 5 | 1 | 0 | 181 | 0 | 9 |
| 8 | 3 | 0 | 1 | 16 | 6 | 6 | 0 | 1 | 164 | 3 |
| 9 | 0 | 1 | 0 | 0 | 8 | 0 | 0 | 7 | 2 | 182 |

表 2: 各カテゴリごとの結果

| Category | # Data | # Correct | Accuracy |
|---|---|---|---|
| 0 | 200 | 192 | 0.960 |
| 1 | 200 | 199 | 0.995 |
| 2 | 200 | 169 | 0.845 |
| 3 | 200 | 182 | 0.910 |
| 4 | 200 | 182 | 0.910 |
| 5 | 200 | 162 | 0.810 |
| 6 | 200 | 185 | 0.925 |
| 7 | 200 | 181 | 0.905 |
| 8 | 200 | 164 | 0.820 |
| 9 | 200 | 182 | 0.910 |

# プログラム

実行環境と用いた言語・ライブラリを以下の表 3 に示す。

表 3: プログラムの実行環境

| | | |
|---|---|---|
| OS | : | Microsoft Windows 10 Pro (64bit) |
| CPU | : | Intel(R) Core(TM) i5-4300U |
| RAM | : | 4.00 GB |
| 使用言語 | : | Python3.6 |
| 可視化 | : | matplotlib ライブラリ |

Listings 1: assignment3.py

```
# -*- coding: utf-8 -*-


import numpy as np
import matplotlib.pyplot as plt


def generate_sample(n, alpha):
    n1 = sum(np.random.rand(n) < alpha)
    n2 = n - n1
    mean1, mean2 = np.array([2, 0]), np.array([-2, 0])
    cov = np.array([[1, 0], [0, 9]])
    x1 = np.random.multivariate_normal(mean1, cov, n1).transpose()
    x2 = np.random.multivariate_normal(mean2, cov, n2).transpose()
    return x1, x2


def sampling_normal(mean, cov, n):
    return np.random.multivariate_normal(mean, cov, n)


def main():
    np.random.seed(0)

    # settings
    n = 100
    alpha = 0.3

    mu_1 = np.array([2, 0])
    mu_2 = np.array([-2, 0])
    sigma = np.array([[1, 0], [0, 9]])

```

```

    # variables
    n_1 = sum(np.random.rand(n) < alpha)
    n_2 = n - n_1

    p_1 = alpha
    p_2 = 1 - alpha


    # generate data
    x_1 = sampling_normal(mu_1, sigma, n_1)
    x_2 = sampling_normal(mu_2, sigma, n_2)
    # print(x_1.shape)


    # calc
    constant = 0.0
    sigma_inv = np.linalg.inv(sigma)


    # log probs
    def log_probabilities(x):
        logp_1x = mu_1.T.dot(sigma_inv).dot(x.T) - (1/2) * mu_1.T.dot(sigma_inv).dot(mu_1) + np.log(p_1) + constant
        logp_2x = mu_2.T.dot(sigma_inv).dot(x.T) - (1/2) * mu_2.T.dot(sigma_inv).dot(mu_2) + np.log(p_2) + constant
        return logp_1x, logp_2x

    x = x_1
    logp_1x, logp_2x = log_probabilities(x)
    is_1 = (logp_1x > logp_2x)
    print('1: #Data: {}\t#Correct: {}\tAcc: {:.3f}'.format(len(x), is_1.sum(), is_1.sum()/len(x)))

    x = x_2
    logp_1x, logp_2x = log_probabilities(x)
    is_2 = (logp_1x < logp_2x)
    print('2: #Data: {}\t#Correct: {}\tAcc: {:.3f}'.format(len(x), is_2.sum(), is_2.sum()/len(x)))


    # coeffs of decision boundary
    a = (mu_1.T - mu_2.T).dot(sigma_inv).T
    b = -(1/2) * ((mu_1.T).dot(sigma_inv).dot(mu_1) - (mu_2.T).dot(sigma_inv).dot(mu_2)) + np.log(p_1/p_2)
    # _x = np.arange(-5, 5, 0.1)

```

```
    plt.title(r'$\alpha = {}$'.format(alpha))
    plt.scatter(x_1[:, 0], x_1[:, 1], marker='o')
    plt.scatter(x_2[:, 0], x_2[:, 1], marker='x')
    #plt.plot(-a[0]/a[1]*_x + b/a[1], _x)
    plt.show()


if __name__ == '__main__':
    main()
```

Listings 2: `assignment4.py`

```
# -*- coding: utf-8 -*-


import pathlib
import numpy as np
import matplotlib.pyplot as plt


def fisher(x, mean, cov_inv, p_y):
    logp = mean.T.dot(cov_inv).dot(x.T) - (1/2)*mean.T.dot(cov_inv).dot(mean) + np.log(p_y)
    return logp


def mahalanobis(x, mean, cov, p_y, eps=1e-6):
    cov_inv = np.linalg.inv(cov + eps*np.eye(len(cov)))
    logp = -(1/2)*np.diag((x - mean.T).dot(cov_inv).dot((x - mean.T).T))
    logp += - (1/2)*np.log(np.linalg.det(cov)) + np.log(p_y)
    return logp


def main():
    np.random.seed(0)

    datadir = pathlib.Path().cwd().parent / 'data'

    n_category = 10
    categories = list(range(10))

    # train
    data = []
    means = []
    covs = []
    for category in categories:
        data_path = datadir / 'digit_train{}.csv'.format(category)
        _data = np.loadtxt(str(data_path), delimiter=',')
        mean = np.mean(_data, axis=0)
        cov = np.cov(_data.T)
        data.append(_data)
        means.append(mean)
        covs.append(cov)
    cov_train = np.zeros_like(covs[0])
    for i in range(n_category):
        cov_train += covs[i]
    cov_train /= n_category
    cov_train_inv = np.linalg.inv(cov_train + 1e-8*np.eye(len(cov_train)))
```

```


    # test
    n_test = 0
    data_test = []
    for category in categories:
        data_path = datadir / 'digit_test{}.csv'.format(category)
        _data = np.loadtxt(str(data_path), delimiter=',')
        n_test += len(_data)
        data_test.append(_data)

    confusion_matrix = np.zeros((n_category, n_category))
    for y, data in enumerate(data_test):
        print('Category: {}\t'.format(y), end='')
        n_data = len(data)
        p_y = n_data / n_test
        preds = []
        for category in categories:
            mean = means[category]
            cov = covs[category]
            logp = fisher(data, mean, cov_train_inv, p_y)
            # logp = mahalanobis(data, mean, cov, p_y)
            preds.append(logp)
        preds = np.array(preds).T
        flag = np.argmax(preds, axis=1)
        for category in categories:
            n = (flag == category).sum()
            confusion_matrix[y, category] = n
        n_correct = (flag == y).sum()
        acc = n_correct / n_data
        print('#Data: {}\t#Crr: {}\tAcc: {:.3f}'.format(n_data, n_correct,
    acc))

    print()
    print('Confusion Matrix\n', confusion_matrix)
    print()

    n_crr_all = np.diag(confusion_matrix).sum()
    n_data_all = 200 * 10
    acc_all = n_crr_all / n_data_all
    print('All\t#Data: {}\t#Crr: {}\tAcc: {:.3f}'.format(n_data_all,
    n_crr_all, acc_all))


if __name__ == '__main__':
    main()
```