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 1 に示す。

表 1: 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()
```