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: assignment1.py

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


import numpy as np
import matplotlib.pyplot as plt
import scipy.linalg


def generate_data(sample_size=100, pattern=’two_cluster’):
    if pattern not in [’two_cluster’, ’three_cluster’]:
        raise ValueError(’Dataset pattern must be one of ’
                         ’[two_cluster, three_cluster].’)
    x = np.random.normal(size=(sample_size, 2))
    if pattern == ’two_cluster’:
        x[:sample_size // 2, 0] -= 4
        x[sample_size // 2:, 0] += 4
    else:
        x[:sample_size // 4, 0] -= 4
        x[sample_size // 4:sample_size // 2, 0] += 4
    y = np.ones(sample_size, dtype=np.int64)
    y[sample_size // 2:] = 2
    return x, y


def scatter_matrices(x, y):
    n = x.shape[0]
    d = x.shape[1]

    labels = np.unique(y)
    C, Sw, Sb = np.zeros((d, d)), np.zeros((d, d)), np.zeros((d, d))
    for label in labels:
        x_y = x[(y == label), :]
        n_y = x_y.shape[0]
        mu_y = x_y.mean(axis=0)[:, np.newaxis]
        Sb += n_y * mu_y.dot(mu_y.T)

        for i in range(n_y):
            x_i = x_y[i][:, np.newaxis]
            diff = (x_i - mu_y)
            Sw += diff.dot(diff.T)

            C += x_i.dot(x_i.T)

    return C, Sw, Sb


def train(x, y, n_components):
    """Fisher Discriminant Analysis.
    Implement this function

    Returns
    -------
    T : (1, 2) ndarray
        The embedding matrix.
    """

    C, Sw, Sb = scatter_matrices(x, y)
    eigen_values, eigen_vectors = scipy.linalg.eig(Sb, Sw)

    # normalize
    for i in range(len(eigen_vectors)):
        eigen_vectors[i] = eigen_vectors[i]/np.linalg.norm(eigen_vectors[i])

    T = eigen_vectors[:n_components]
    return T


def visualize(x, y, T, path=None):
    plt.figure(1, (6, 6))
    plt.xlim(-7., 7.)
    plt.ylim(-7., 7.)
    plt.plot(x[y == 1, 0], x[y == 1, 1], ’bo’, label=’class-1’)
    plt.plot(x[y == 2, 0], x[y == 2, 1], ’rx’, label=’class-2’)
    plt.plot(
        np.array([-T[:, 0], T[:, 0]]) * 9,
        np.array([-T[:, 1], T[:, 1]]) * 9,
        ’k-’,
        )
    plt.legend()
    if path:
        plt.savefig(str(path))
    plt.show()


def main():
    # settings
    n = 100
    n_components = 1
    #mode = ’two_cluster’
    mode = ’three_cluster’
    fig_path = f’../figures/assignment1_result_{mode}.png’
    np.random.seed(10)


    # generate data
    x, y = generate_data(sample_size=n, pattern=mode)
    #print(x.shape, y.shape)


    # train
    T = train(x, y, n_components)


    # result
    print(f’data: {mode}  (#sample = {n})’)
    print(f’T = {T}’)

    visualize(x, y, T, path=fig_path)



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