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: assignment2.py

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


import numpy as np
import scipy.linalg

import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


def generate_data(n=1000):
    a = 3. * np.pi * np.random.rand(n)
    x = np.stack([
        a*np.cos(a),
        30*np.random.random(n),
        a*np.sin(a),
        ], axis=1)
    return a, x


def similarity_matrix(x, k=1):
    n = x.shape[0]
    W = np.zeros((n, n))
    for i in range(n):
        x_i = x[i, :]
        W[i, i] = 1

        norms_i = np.sum((x_i - x)**2, axis=1)
        idx_nearest = np.argsort(norms_i)[1:1+k]

        W[i, idx_nearest] = 1
```

```
            W[idx_nearest, i] = 1
    return W


def train(x, d, k=1, eps=1e-8):
    W = similarity_matrix(x)
    D_diag = W.sum(axis=0)
    D = np.diag(D_diag)
    L = D - W

    eigen_values, eigen_vectors = scipy.linalg.eig(L, D)
    eigen_vectors = eigen_vectors.T

    indices = np.argsort(eigen_values)[::-1]
    eigen_values = eigen_values[indices]
    eigen_vectors = eigen_vectors[indices]
    #eigen_values[(-eps < eigen_values) & (eigen_values < eps)] = 0

    #eigen_vectors_reduced = eigen_vectors[(eigen_values>0), :][::-1]
    eigen_vectors_reduced = eigen_vectors[::-1][1:]

    Psi_T = eigen_vectors_reduced[:d]
    Psi = Psi_T.T
    return Psi


def visualize(x, z, a, path=None):
    fig = plt.figure(figsize=(12, 6))

    ax = fig.add_subplot(1, 2, 1, projection='3d')
    ax.scatter3D(x[:, 0], x[:, 1], x[:, 2], c=a, marker='o')

    ax = fig.add_subplot(1, 2, 2)
    ax.scatter(z[:, 1], z[:, 0], c=a, marker='o')

    if path:
        plt.savefig(str(path))
    plt.show()


def main():
    # settings
    n = 1000
    k = 1
    n_components = 2
    fig_path = f'../figures/assignment2_result.png'
    np.random.seed(1)
```

```
81
82
83      # generate data
84      a, x = generate_data(n)
85      #print(a.shape, x.shape)
86
87
88      # preprocess
89      #mu = x.mean(axis=0)
90      #x = x - mu
91
92
93      # train
94      z = train(x, d=n_components, k=k)
95
96
97      # result
98      print(f'#Data: {n}')
99      print(f'z shape: {z.shape}')
100     print(f'z = \n{z}')
101
102     visualize(x, z, a, path=fig_path)
103
104
105 if __name__ == '__main__':
106     main()
```