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

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


import numpy as np
import matplotlib.pyplot as plt


def generate_data(n=50):
    x = np.random.randn(n, 3)
    x[:n // 2, 0] -= 15
    x[n // 2:, 0] -= 5
    x[1:3, 0] += 10
    x[:, 2] = 1
    y = np.concatenate((np.ones(n // 2), -np.ones(n // 2)))
    index = np.random.permutation(np.arange(n))
    return x[index], y[index]


def phi(x):
    return x


def update(x, y, gamma, theta):
    #import pdb; pdb.set_trace()
    mu, sigma = theta
    n_sample = x.shape[0]
    phi_x = phi(x)

    beta = gamma + (phi_x.dot(sigma) * phi_x).sum(axis=1)

    hinge = 1 - phi_x.dot(mu) * y
    hinge[hinge < 0] = 0
    d_mu = (y * hinge / beta)[:, np.newaxis] * sigma.dot(phi_x.T).T
    d_mu = d_mu.mean(axis=0)

    d_sigma = np.zeros_like(sigma)
    for i in range(n_sample):
        _phi_x = phi_x[i, :]
        _beta = beta[i]
        tmp = sigma.dot(_phi_x)[:, np.newaxis]
        d_sigma += tmp.dot(tmp.T) / _beta
    d_sigma /= n_sample

    mu_new = mu + d_mu
    sigma_new = sigma - d_sigma
    return mu_new, sigma_new


def compute_loss(x, y, gamma, theta, theta_old):
    mu, sigma = theta
    mu_old, sigma_old = theta_old

    d = mu.shape[0] - 1
    sigma_old_inv = np.linalg.inv(sigma_old)
    phi_x = phi(x)

    loss_main = 1 - phi_x.dot(mu) * y
    loss_main[loss_main < 0] = 0
    loss_main = loss_main ** 2
    loss_main = loss_main.mean()

    loss_var = (phi_x.dot(sigma) * phi_x).sum(axis=1)
    loss_var = loss_var.mean()

    loss_KL = (0
        + np.log(np.linalg.det(sigma_old) / np.linalg.det(sigma))
        + np.trace(sigma_old_inv.dot(sigma))
        + (mu - mu_old).T.dot(sigma_old_inv).dot(mu - mu_old)
        - d
        )
    loss_KL.mean()

    loss = loss_main + loss_var + gamma * loss_KL
    return loss, loss_main, loss_var, loss_KL


def train(x, y, gamma, epochs=1, batch_size=1, shuffle=True):
    d = x.shape[1]
    n_sample = x.shape[0]
    mu = np.random.random(d)
    sigma = np.diag(np.random.random(d) + 0.1)
    theta = (mu, sigma)
    print(’train’)
    for epoch in range(1, 1+epochs):
        if shuffle:
            idx = np.random.permutation(np.arange(n_sample))
            x = x[idx]
            y = y[idx]
        loss_list = []
        for i in range(0, n_sample, batch_size):
            x_mini = x[i:i+batch_size]
            y_mini = y[i:i+batch_size]

            theta_new = update(x_mini, y_mini, gamma, theta)

            losses = compute_loss(x_mini, y_mini, gamma, theta_new, theta)
            loss_list.append(losses)

            theta = theta_new

        losses = np.array(loss_list).mean(axis=0)
        loss, loss_main, loss_var, loss_KL = tuple(losses)
        print(f’Epoch: {epoch}  Loss: {loss:.4f}  (main: {loss_main:.4f}  var: {loss_var:.4f}  KL: {loss_KL:.4f})’)
    print()
    return theta


def visualize(x, y, theta, num=100, offset=1.0, path=None):
    x1_max, x1_min = x[:, 0].max(), x[:, 0].min()
    x2_max, x2_min = x[:, 1].max(), x[:, 1].min()

    X = np.linspace(x1_min, x1_max, num=num)
    mu, sigma = theta

    plt.xlim(x1_min-offset, x1_max+offset)
    plt.ylim(x2_min-offset, x2_max+offset)

    plt.scatter(x[(y==1), 0], x[(y==1), 1] , c=’blue’, marker=’o’)
    plt.scatter(x[(y==-1), 0], x[(y==-1), 1] , c=’red’, marker=’x’)

    if abs(mu[0]) > abs(mu[1]):
        plt.plot(
            [x1_min, x1_max],
            [-(mu[2]+mu[0]*x1_min)/mu[1], -(mu[2]+mu[0]*x1_max)/mu[1]]
            )
    else:
        plt.plot(
            [-(mu[2]+mu[1]*x2_min)/mu[0], -(mu[2]+mu[1]*x2_max)/mu[0]],
            [x2_min, x2_max]
            )

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


def main():
    # settings
    gamma = 1.0
    n_sample = 50
    batch_size = 10
    epochs = 50
    fig_path = '../figures/assignment2_result.png'
    np.random.seed(0)

    # load data
    x, y = generate_data(n_sample)
    #print(x)
    #print(y)

    # train
    theta = train(x, y, gamma, epochs=epochs, batch_size=batch_size)
    mu, sigma = theta

    # result
    print('result')
    print(f'#Sample: {n_sample}')
    print(f'gamma: {gamma}')
    print(f'mu: {mu}')
    print(f'sigma: \n{sigma}')

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


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