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
matplotlib.use(’TkAgg’)
import matplotlib.pyplot as plt


def generate_data(sample_size):
    """Generate training data.

    Since
    f(x) = w^{T}x + b
    can be written as
    f(x) = (w^{T}, b)(x^{T}, 1)^{T},
    for the sake of simpler implementation of SVM,
    we return (x^{T}, 1)^{T} instead of x

    :param sample_size: number of data points in the sample
    :return: a tuple of data point and label
    """

    x = np.random.normal(size=(sample_size, 3))
    x[:, 2] = 1.
    x[:sample_size // 2, 0] -= 5.
    x[sample_size // 2:, 0] += 5.
    y = np.concatenate([np.ones(sample_size // 2, dtype=np.int64),
                        -np.ones(sample_size // 2, dtype=np.int64)])
    x[:3, 1] -= 5.
    y[:3] = -1
    x[-3:, 1] += 5.
    y[-3:] = 1
    return x, y


def calc_subgrad(x, y, w):
    f = x.dot(w)
    z = y * f
    yx = y[:, np.newaxis] * x

    indices_over_1 = (z > 1)
    indices_equals_1 = (z == 1)
    indices_under_1 = (z < 1)

    subgrads = np.zeros_like(x)
    subgrads[indices_over_1] = 0
    subgrads[indices_under_1] = - yx[indices_under_1]
    subgrads[indices_equals_1] = 0

    subgrad = subgrads.sum(axis=0)
    return subgrad


def calc_grad(x, y, w, c):
    subgrad = calc_subgrad(x, y, w)
    grad_w = 2*w
    grad_w[2] = 0
    grad = grad_w + c*subgrad
    return grad


def update(x, y, w, c, lr):
    grad = calc_grad(x, y, w, c)
    w_new = w - lr * grad
    return w_new


def svm(x, y, c, lr, max_iter=1e4, eps=1e-3):
    """Linear SVM implementation using gradient descent algorithm.

    f_w(x) = w^{T} (x^{T}, 1)^{T}

    :param x: data points
    :param y: label
    :param l: regularization parameter
    :param lr: learning rate
    :return: three-dimensional vector w
    """
    d = x.shape[1]
    w = np.zeros(d)
    prev_w = w.copy()
    for i in range(int(max_iter)):
        w = update(x, y, w, c, lr)

        # convergence condition
        if np.linalg.norm(w - prev_w) < eps:
            break
        prev_w = w.copy()
    n_iter = i + 1
    return w, n_iter


def visualize(x, y, w, path=None):
    plt.clf()
    plt.xlim(-10, 10)
    plt.ylim(-10, 10)
    plt.scatter(x[y == 1, 0], x[y == 1, 1])
    plt.scatter(x[y == -1, 0], x[y == -1, 1])
    plt.plot([-10, 10], -(w[2] + np.array([-10, 10]) * w[0]) / w[1])
    if path:
        plt.savefig(path)
    plt.show()


def main():
    # settings
    n_sample = 200
    fig_path = ’../figures/assignment2_result.png’
    np.random.seed(0)

    # load data
    x, y = generate_data(n_sample)

    # train
    w, n_iter = svm(x, y, c=.1, lr=0.05, max_iter=1e4, eps=1e-4)

    # result
    print(f’#Sample: {n_sample}’)
    print(f’#Iter: {n_iter}’)
    print(f’w: {w}’)
    visualize(x, y, w, fig_path)


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