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


def generate_data(n=200):
    x = np.linspace(0, np.pi, n // 2)
    u = np.stack([np.cos(x) + .5, -np.sin(x)], axis=1) * 10.
    u += np.random.normal(size=u.shape)
    v = np.stack([np.cos(x) - .5, np.sin(x)], axis=1) * 10.
    v += np.random.normal(size=v.shape)
    x = np.concatenate([u, v], axis=0)
    y = np.zeros(n)
    y[0] = 1
    y[-1] = -1
    return x, y


def calc_norm(x, c, save_memory=False):
    if save_memory:
        n_x = x.shape[1]
        n_c = c.shape[0]
        d = x.shape[-1]
        norm = np.zeros((n_c, n_x))
        x = np.reshape(x, (n_x, d))
        c = np.reshape(c, (n_c, d))
        for i in range(len(x)):
            x_i = x[i]
            norm[i, :] = np.sum((x_i - c)**2, axis=-1)
    else:
        norm = np.sum((x - c) ** 2, axis=-1)
```

```
34      return norm
35
36
37  def gauss_kernel(x, c, h, save_memory=False):
38      norm = calc_norm(x, c, save_memory)
39      ker = np.exp(- norm / (2*h**2))
40      return ker
41
42
43  def calc_design_matrix(x, c, h, kernel):
44      return kernel(x[None], c[:, None], h)
45
46
47  def lrls(x, y, h=1., l=1., nu=1., kernel=gauss_kernel):
48      """
49
50      :param x: data points
51      :param y: labels of data points
52      :param h: width parameter of the Gaussian kernel
53      :param l: weight decay
54      :param nu: Laplace regularization
55      :return:
56      """
57      x_tilde = x[y!=0]
58      K = calc_design_matrix(x, x, h, kernel)
59      K_tilde = calc_design_matrix(x_tilde, x, h, kernel)
60
61      W = K
62      D = np.diag(W.sum(axis=1))
63      L = D - W
64
65      tmp = K_tilde.dot(K_tilde.T) + l*np.eye(len(K_tilde)) +
        2*nu*K.dot(L).dot(K)
66      theta = np.linalg.inv(tmp).dot(K_tilde).dot(y[y!=0])
67      return theta
68
69
70  def supervised_train(x, y, h=1., l=1., kernel=gauss_kernel):
71      K = calc_design_matrix(x, x, h, kernel)
72      tmp = K.dot(K.T) + l*np.eye(len(K))
73      theta = np.linalg.inv(tmp).dot(K).dot(y)
74      return theta
75
76
77  def visualize(x, y, theta, h=1., grid_size=100, path=None):
78      plt.xlim(-20., 20.)
79      plt.ylim(-20., 20.)
```

```
80      grid = np.linspace(-20., 20., grid_size)
81
82      X, Y = np.meshgrid(grid, grid)
83      mesh_grid = np.stack([np.ravel(X), np.ravel(Y)], axis=1)
84
85      k = np.exp(
86          -np.sum(
87              (x.astype(np.float32)[:, None] -
        mesh_grid.astype(np.float32)[None])**2,
88              axis=2).astype(np.float64)
89              / (2 * h ** 2)
90          )
91      plt.contourf(
92          X, Y,
93          np.reshape(np.sign(k.T.dot(theta)), (grid_size, grid_size)),
94          alpha=.4,
95          cmap=plt.cm.coolwarm,
96          )
97      plt.scatter(x[y==0][:, 0], x[y == 0][:, 1], marker='$.$', c='black')
98      plt.scatter(x[y==1][:, 0], x[y == 1][:, 1], marker='$X$', c='red')
99      plt.scatter(x[y==-1][:, 0], x[y == -1][:, 1], marker='$O$', c='blue')
100     if path:
101         plt.savefig(path)
102     plt.show()
103
104
105 def main():
106     # settings
107     h = 1.0
108     lamb = 1.0
109     nu = 1.0
110     n = 200
111     fig_path = '../figures/assignment1_result.png'
112     np.random.seed(0)
113
114
115     # generate data
116     x, y = generate_data(n)
117     #print(x.shape, y.shape)
118
119
120     # train
121     theta = lrls(x, y, h=h, l=lamb, nu=nu)
122
123     # supervised
124     #x, y = x[y!=0], y[y!=0]
125     #theta = supervised_train(x, y, h=h, l=lamb,)
```

```
126
127
128         # result
129         print(f'#data: {n}')
130         print(f'h = {h}  lambda = {lamb}  nu = {nu}')
131         #print('theta = \n', theta)
132
133         visualize(x, y, theta, path=fig_path)
134
135
136     if __name__ == '__main__':
137         main()
```

Listings 2: assignment3.py

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


import numpy as np
import matplotlib.pyplot as plt


def generate_data(n_total, n_positive):
    x = np.random.normal(size=(n_total, 2))
    x[:n_positive, 0] -= 2
    x[n_positive:, 0] += 2
    x[:, 1] *= 2.
    y = np.empty(n_total, dtype=np.int64)
    y[:n_positive] = 0
    y[n_positive:] = 1
    return x, y


def calc_A(y, y_tilde, x_train, y_train):
    indices_i = np.where(y_train == y)[0]
    indices_i_tilde = np.where(y_train == y_tilde)[0]

    if len(indices_i) * len(indices_i_tilde) == 0:
        return 0

    x_tilde = x_train[indices_i_tilde]
    A = 0
    for idx_i in indices_i:
        A += np.sum(np.sqrt(np.sum((x_train[idx_i] - x_tilde)**2, axis=0)))
    A /= (len(indices_i) * len(indices_i_tilde))
    return A


def calc_b(y, x_train, y_train, x_test):
    x_i = x_test[y_train == y]

    if len(x_i) * len(x_test) == 0:
        return 0

    b = 0
    for x_i_dash in x_test:
        b += np.sum(np.sqrt(np.sum((x_i_dash - x_i)**2, axis=0)))
    b /= (len(x_i) * len(x_test))
    return b


```

```
47 def estimate_pi(x_train, y_train, x_test):
48     x, y = x_train, y_train
49     A_pp = calc_A(1, 1, x, y)
50     A_pm = calc_A(1, -1, x, y)
51     A_mm = calc_A(-1, -1, x, y)
52     b_p = calc_b(1, x_train, y_train, x_test)
53     b_m = calc_b(-1, x_train, y_train, x_test)
54 
55     pi_hat = (A_pm - A_mm - b_p + b_m) / (2*A_pm - A_pp - A_mm)
56     pi_hat = min(1, max(0, pi_hat))
57     return pi_hat
58 
59 
60 def cwls(train_x, train_y, test_x, is_weighted=True):
61     n = train_y.shape[0]
62 
63     if is_weighted:
64         pi_hat = estimate_pi(train_x, train_y, test_x)
65         Pi = np.zeros(n)
66         Pi[train_y == 1] = pi_hat
67         Pi[train_y == -1] = 1 - pi_hat
68         Pi = np.diag(Pi)
69     else:
70         Pi = np.eye(n)
71 
72     Phi = np.concatenate([np.ones(n)[:, np.newaxis], train_x], axis=1)
73 
74     theta =
       np.linalg.inv(Phi.T.dot(Pi).dot(Phi)).dot(Phi.T).dot(Pi).dot(train_y)
75     return theta
76 
77 
78 def visualize(train_x, train_y, test_x, test_y, theta, is_weighted=True):
79     str_weighted = 'weighted' if is_weighted else 'unweighted'
80     for x, y, name in [(train_x, train_y, 'train'), (test_x, test_y,
       'test')]:
81         plt.xlim(-5., 5.)
82         plt.ylim(-7., 7.)
83         lin = np.array([-5., 5.])
84         plt.plot(lin, -(theta[2] + lin * theta[0]) / theta[1])
85         plt.scatter(x[y==0][:, 0], x[y==0][:, 1], marker='$O$', c='blue')
86         plt.scatter(x[y==1][:, 0], x[y==1][:, 1], marker='$X$', c='red')
87 
       plt.savefig('../figures/assignment3_result_{}_{}.png'.format(str_weighted,
       name))
88         plt.show()
89 
```

```
90
91  def main():
92      # settings
93      is_weighted = False
94      np.random.seed(0)
95
96
97      # generate data
98      train_x, train_y = generate_data(n_total=100, n_positive=90)
99      eval_x, eval_y = generate_data(n_total=100, n_positive=10)
100
101
102     # train
103     theta = cwls(train_x, train_y, eval_x, is_weighted=is_weighted)
104
105
106     # result
107     print('result')
108     visualize(train_x, train_y, eval_x, eval_y, theta,
        is_weighted=is_weighted)
109
110
111 if __name__ == '__main__':
112     main()
```