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 pathlib
import numpy as np
import matplotlib.pyplot as plt


def load_data(n_label=None, n_train=None, n_test=None):
    data_dir = '../data/'
    data_dir = pathlib.Path(data_dir)
    categories = list(range(10))
    train_X = []
    test_X = []
    for category in categories[:n_label]:
        # train data
        data_path = data_dir / 'digit_train{}.csv'.format(category)
        data = np.loadtxt(str(data_path), delimiter=',')[:n_train]
        train_X.append(data)

        # test data
        data_path = data_dir / 'digit_test{}.csv'.format(category)
        data = np.loadtxt(str(data_path), delimiter=',')[:n_test]
        test_X.append(data)
    labels = categories[:n_label]
    return train_X, test_X, labels


def make_train_data(train_data, labels, label):
    train_X = []
    train_y = []
    for i in labels:
        data = train_data[i]
```

```
        train_X.extend(data)
        n_data = len(data)
        if i == label:
            train_y.extend([1] * n_data)
        else:
            train_y.extend([-1] * n_data)
    train_X = np.array(train_X)
    train_y = np.array(train_y)
    return train_X, train_y


class GaussKernelModel(object):
    def __init__(self, n, bandwidth):
        self.n = n
        self.bandwidth = bandwidth
        self.theta = np.empty(n)

    def __call__(self, x, c):
        K = self.design_matrix(x, c)
        y_hat = K.T.dot(self.theta)
        return y_hat

    def kernel(self, 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)
        ker = np.exp(- norm / (2*self.bandwidth**2))
        return ker

    def design_matrix(self, x, c, save_memory=False):
        mat = self.kernel(x[None], c[:, None], save_memory)
        return mat

    def train(self, x, y, lamb=1e-4, save_memory=False):
        K = self.design_matrix(x, x, save_memory)
        self.theta = np.linalg.solve(
            K**2 + lamb*np.identity(len(K)),
            K.T.dot(y[:, None]),
```

```
            )
        return


def train_one_vs_others(train_data, labels, h, lamb=1e-4, save_memory=False):
    model = []
    print('Train: ', end='')
    for label in labels:
        print(f'{label}  ', end='')
        train_X, train_y = make_train_data(train_data, labels, label)
        n_data = len(train_y)
        model_i = GaussKernelModel(n_data, h)
        model_i.train(train_X, train_y, lamb, save_memory)
        model.append(model_i)
    print('\ndone\n')
    return model


def test(model, train_data, test_data, labels):
    n_label = len(labels)
    confusion_matrix = np.zeros((n_label, n_label), dtype=int)
    n_data_all = 0
    result = {}
    print('Test')
    for label in labels:
        print(f'Label: {label}\t', end='')

        # load
        test_X = test_data[label]
        n_data = len(test_X)
        n_data_all += n_data
        train_X = []
        for i in labels:
            data = train_data[i]
            train_X.extend(data)
        train_X = np.array(train_X)

        # predict
        preds = []
        for i in labels:
            pred = model[i](test_X, train_X).flatten()  # (n_data,)
            preds.append(pred)
        preds = np.array(preds).T  # (n_data, n_label)
        preds = np.argmax(preds, axis=1)  # (n_data,)

        # make confusion matrix
        for i in labels:
```

```
128             n = (preds == i).sum()
129             confusion_matrix[label, i] = n
130 
131         # calc accuracy
132         n_correct = confusion_matrix[label, label]
133         acc = n_correct / n_data
134         print(f'#Data: {n_data}\t#Correct: {n_correct}\tAcc: {acc:.3f}')
135 
136         result[label] = {
137             'data': n_data,
138             'correct': n_correct,
139             'accuracy': acc,
140             }
141     result['confusion_matrix'] = confusion_matrix
142 
143     # overall score
144     n_crr_all = np.diag(confusion_matrix).sum()
145     acc_all = n_crr_all / n_data_all
146     result['all'] = {
147         'data': n_data_all,
148         'correct': n_crr_all,
149         'accuracy': acc_all,
150         }
151     print(f'All\t#Data: {n_data_all}\t#Correct: {n_crr_all}\tAcc:
    {acc_all:.3f}')
152     print()
153     print('Confusion Matrix:\n', confusion_matrix)
154     print()
155     return result
156 
157 
158 def print_result_in_TeX_tabular_format(result):
159     labels = list(range(10))
160     print('Scores')
161     for label in labels:
162         print('{} & {} & {} & {:.3f} \\\\'.format(
163             label,
164             int(result[label]['data']),
165             int(result[label]['correct']),
166             result[label]['accuracy']
167             ))
168     print()
169     print('Confusion Matrix')
170     for i in labels:
171         print('{}    '.format(i), end='')
172         for j in labels:
173             print(' & {}'.format(int(result['confusion_matrix'][i, j])),
```

```
        end='')
174         print(' \\\\')
175     return
176
177
178 def main():
179     # settings
180     bandwidth = 1.0
181     lamb = 1e-4
182     np.random.seed(0)
183
184     print('Settings')
185     print(f'bandwidth: {bandwidth}\tlambda (L2): {lamb}\n')
186
187     # load data
188     train_X, test_X, labels = load_data(n_label=10, n_train=None,
        n_test=None)
189
190     # train
191     model = train_one_vs_others(train_X, labels, bandwidth, lamb,
        save_memory=True)
192
193     # test
194     result = test(model, train_X, test_X, labels)
195     print_result_in_TeX_tabular_format(result)
196
197 if __name__ == '__main__':
198     main()
```