# SML HW6

29-176004 奥村 恭平 *†

June 9, 2018

## 宿題 1: カーネル密度推定法の実装

ガウスカーネルに対するカーネル密度推定法を実装した．(言語は python．) バンド幅は尤度交差確認により決定 ($b^* := 0.1$) した．以下はシミュレーション結果の図．上がオリジナルのサンプルのヒストグラム，下が推定された密度関数．うまく推定できていることがわかる．

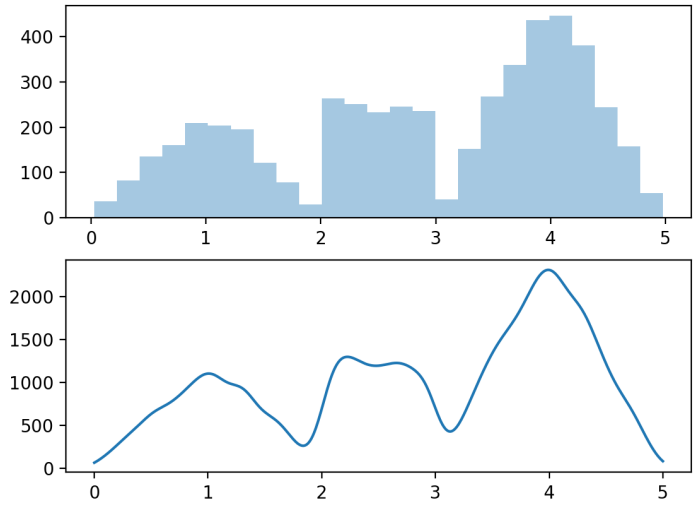


Figure 1: シミュレーション実行結果

```
import numpy as np
import scipy as sp
from numpy.random import randn, rand
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm


def myrand(n=5000):
    x = np.zeros(n)
    u = rand(n)
    flag = (0 <= u) * (u < 1/8)
    x[flag] = np.sqrt(8*u[flag])
    flag = (1/8 <= u) * (u < 1/4)
    x[flag] = 2 - np.sqrt(2 - 8*u[flag])
    flag = (1/4 <= u) * (u < 1/2)
    x[flag] = 1 + 4*u[flag]
```

*E-mail: kyohei.okumura@gmail.com
†東京大学大学院 経済学研究科 M2

```
    flag = (1/2 <= u) * (u < 3/4)
    x[flag] = 3 + np.sqrt(4*u[flag] - 2)
    flag = (3/4 <= u) * (u <= 1)
    x[flag] = 5 - np.sqrt(4 - 4*u[flag])
    return x


def gkernel_est(sample, x=np.linspace(0, 5, 501), bandwidth=0.1):
    pxh = np.zeros_like(x)
    n = sample.shape[0]
    for i in range(n):
        pxh = pxh + norm.pdf(x, loc=sample[i], scale=bandwidth)
    return x, pxh


def cross_validation(sample, n_split=5, params=[0.01, 0.1, 0.5]):
    n_params = len(params)
    likelihoods = np.zeros(n_params)
    group = np.split(sample, n_split)
    for j in range(n_params):
        for i in range(n_split):
            if i==0:
                sample_temp = np.hstack(group[i+1:][0])
            elif i==n_split-1:
                sample_temp = np.hstack(group[0:i])
            else:
                sample_temp = np.hstack([np.hstack(group[0:i]), group[i
                    +1:][0]])
            _, pxh = gkernel_est(sample_temp, group[i], bandwidth=params[j])
            likelihoods[j] += np.sum(np.log(pxh))
    opt_param = params[np.argmax(likelihoods)]
    #print(likelihoods)
    return opt_param


if __name__ == '__main__':
    np.random.seed(1)
    sample = myrand()
    opt_b = cross_validation(sample=sample)
    x, pxh = gkernel_est(sample=sample, bandwidth=opt_b)

    # plot original samples
    fig = plt.figure()
    ax1 = fig.add_subplot(2,1,1)
    sns.distplot(sample, kde=False, rug=False, bins=25)
    None

    # plot estimated distribution
    ax2 = fig.add_subplot(2,1,2)
    ax2.plot(x, pxh)
    plt.show()
```

## 宿題 2: kNN 法による文字識別

kNN 法による手書き文字認識を行った．言語は python．最近傍数 $k$ は交差確認により決定した．(正答率の計算部分のみ，scikit-learn のモジュールを用いた．) 訓練データを用いて $k \in \{1,2,3,4,5,10,20\}$ について交差確認を行ったところ，$k^* = 1$ が最適になった．(交差確認におけるそれぞれの平均正答率は，$[0.9676, 0.9676, 0.9674, 0.9688, 0.965, 0.959, 0.9454]$ となった．) $k := 1$ としてテストデータを識別したところ，正答率は，0.965 となった．

```
import numpy as np
from collections import Counter
from sklearn.metrics import accuracy_score


class kNN(object):
    def __init__(self, k=1):
        self._train_data = None
        self._target_data = None
        self._k = k


    def fit(self, train_data, target_data):
        self._train_data = train_data
        self._target_data = target_data


    def predict(self, x):
        distances = np.array([np.linalg.norm(p - x) for p in self.
            _train_data])
        nearest_indices = distances.argsort()[:self._k]
        nearest_labels = self._target_data[nearest_indices]
        c = Counter(nearest_labels)
        return c.most_common(1)[0][0]


def load_train_data():
    for i in range(10):
        if i==0:
            train_feature = np.loadtxt('data/digit_train{}.csv'.format(i),
                delimiter=',')
            train_label = np.array([i]*train_feature.shape[0])
        else:
            temp_feature = np.loadtxt('data/digit_train{}.csv'.format(i),
                delimiter=',')
            train_feature = np.vstack([train_feature, temp_feature])
            temp_label = np.array([i]*temp_feature.shape[0])
            train_label = np.hstack([train_label, temp_label])

    return train_feature, train_label


def load_test_data():
    for i in range(10):
        if i==0:
            test_feature = np.loadtxt('data/digit_test{}.csv'.format(i),
                delimiter=',')
            test_label = np.array([i]*test_feature.shape[0])
        else:
            temp_feature = np.loadtxt('data/digit_test{}.csv'.format(i),
                delimiter=',')
            test_feature = np.vstack([test_feature, temp_feature])
            temp_label = np.array([i]*temp_feature.shape[0])
            test_label = np.hstack([test_label, temp_label])

    return test_feature, test_label

```

```
53
54 def calc_accuracy(train_feature, train_label, test_feature, test_label, k=1)
       :
55     model = kNN(k)
56     model.fit(train_feature, train_label)
57     predicted_labels = []
58     for feature in test_feature:
59         predicted_label = model.predict(feature)
60         predicted_labels.append(predicted_label)
61     return accuracy_score(test_label, predicted_labels)
62
63
64 def load_train_data_cv(n_split=5):
65     for i in range(10):
66         if i==0:
67             train_feature = np.loadtxt('data/digit_train{}.csv'.format(i),
                   delimiter=',')
68             train_label = np.array([i]*train_feature.shape[0])
69             group_feature = np.split(train_feature, n_split)
70             group_label = np.split(train_label, n_split)
71         else:
72             temp_feature = np.loadtxt('data/digit_train{}.csv'.format(i),
                   delimiter=',')
73             temp_group_feature = np.split(temp_feature, n_split)
74             temp_label = np.array([i]*temp_feature.shape[0])
75             temp_group_label = np.split(temp_label, n_split)
76
77             for m in range(n_split):
78                 group_feature[m] = np.vstack([group_feature[m],
                       temp_group_feature[m]])
79                 group_label[m] = np.hstack([group_label[m], temp_group_label
                       [m]])
80
81     return group_feature, group_label
82
83
84 def cross_validation(n_split=5, params=[1,2,3,4,5,10,20]):
85     n_params = len(params)
86     score_list = np.zeros(n_params)
87     group_feature, group_label = load_train_data_cv(n_split)
88
89     for j in range(n_params):
90         for i in range(n_split):
91             temp_group_feature = group_feature.copy()
92             temp_test_feature = temp_group_feature.pop(i)
93             temp_train_feature = np.vstack(temp_group_feature)
94
95             temp_group_label = group_label.copy()
96             temp_test_label = temp_group_label.pop(i)
97             temp_train_label = np.hstack(temp_group_label)
98
99             score_list[j] += calc_accuracy(temp_train_feature,
                   temp_train_label, temp_test_feature, temp_test_label, k=
                   params[j])/n_split
100
101    opt_param = params[np.argmax(score_list)]
102    print(score_list)
103    return opt_param
104
105
106 def main():
107    k_opt = cross_validation(n_split=5, params=[1,2,3,4,5,10,20])
108    train_feature, train_label = load_train_data()
109    test_feature, test_label = load_test_data()
110    score = calc_accuracy(train_feature, train_label, test_feature,
           test_label, k=k_opt)
```

```
111     print(score)
112
113
114 if __name__ == '__main__':
115     main()
```