1　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
from scipy.io import loadmat


def load(path):
    data = loadmat(path)
    # print(data.keys())
    x1 = data['x1']
    x2 = data['x2']
    return x1, x2


def plot_data(x1, x2):
    # x1
    plt.hist(x1, bins=50)
    plt.savefig('../figures/x1.png')
    plt.show()

    # x2
    plt.hist(x2, bins=50)
    plt.savefig('../figures/x2.png')
    plt.show()


def normal_distribution(x, mu, sigma):
    p = (1 / np.sqrt(2*np.pi*sigma**2)) * np.exp(-(1/2) * ((x-mu)/sigma)**2)
    return p


```

```
34 def hypercube(x, mu, h):
35     p = 1/h * (np.abs((x - mu)/h) < 1/2)
36     return p
37
38
39 def conditional_probability_parzen(x, h, x_axis, kernel):
40     n = len(x)
41     hn = h/np.sqrt(n)
42     prob = np.zeros(len(x_axis))
43     for x_i in x:
44         prob += kernel(x_axis, x_i, hn)
45     prob = prob / n
46     return prob
47
48
49 def conditional_probability_kmeans(x, k, x_axis, kernel):
50     n = len(x)
51     # k = np.sqrt(n)
52     # kn = k/np.sqrt(n)
53     prob = np.zeros(len(x_axis))
54
55     for i in range(len(x_axis)):
56         # r: sorted list by the distance to x[j]
57         r = sorted(abs(x - x_axis[i]))
58
59         # r[int(k)-1]: k-th distance
60         prob[i] = k / (n * 2 * r[int(k)-1])
61
62     return prob
63
64
65 def _nonparametric_method(
66         x1, x2, p1, p2,
67         param_candidates, param_str,
68         kernel,
69         conditional_probability,
70         offset=1.0, num=100,
71         path=None):
72     x_min = min(x1.min(), x2.min()) - offset
73     x_max = max(x1.max(), x2.max()) + offset
74     x_axis = np.linspace(x_min, x_max, num)
75
76     n_params = len(param_candidates)
77     n_row = 3
78     n_col = n_params + 1
79     fig = plt.figure(figsize=(n_col*4, n_row*6))
80     fig_idx = 0
```

```
    for i, text in enumerate(['prior probability (x1)', 'prior probability
    (x2)', 'posterior probability']):
        fig_idx = 1 + i*n_col
        ax = fig.add_subplot(n_row, n_col, fig_idx)
        ax.tick_params(
            labelbottom=False,
            labelleft=False,
            labelright=False,
            labeltop=False,
            bottom=False,
            left=False,
            right=False,
            top=False,
            )
        for pos in ['bottom', 'left', 'right', 'top']:
            ax.spines[pos].set_visible(False)
        ax.text(0.5, 0.5, text, ha='center', va='bottom', fontsize=12)

    fig_idx = 0
    for i, param in enumerate(param_candidates):
        # calc conditional probability
        p1_cond = conditional_probability(
            x1, param, x_axis, kernel
            )
        p2_cond = conditional_probability(
            x2, param, x_axis, kernel
            )

        # calc post prob
        p1_joint = p1_cond * p1
        p2_joint = p2_cond * p2
        p_sum = p1_joint.sum() + p2_joint.sum()
        p1_post = p1_joint / p_sum
        p2_post = p2_joint / p_sum

        # plot
        ax_1 = fig.add_subplot(n_row, n_col, (i+1)+1)
        title = ''
        if param_str:
            title = '${} = {}$'.format(param_str, param)
        else:
            title = '{}'.format(param)
        ax_1.set_title(title)
        ax_1.hist(x1, bins=50, normed=True)
        ax_1.plot(x_axis, p1_cond)
        ax_1.set_ylim([0, 1.0])

```

```
127         ax_2 = fig.add_subplot(n_row, n_col, n_col+(i+1)+1)
128         ax_2.hist(x2, bins=50, normed=True)
129         ax_2.plot(x_axis, p2_cond)
130         ax_2.set_ylim([0, 1.0])
131 
132         ax = fig.add_subplot(n_row, n_col, 2*n_col+(i+1)+1)
133         ax.plot(x_axis, p1_post, label='1')
134         ax.plot(x_axis, p2_post, label='2')
135         ax.legend()
136         # ax.set_ylim([0, 1.0])
137 
138     plt.savefig(str(path))
139     plt.show()
140 
141 
142 def parzen(x1, x2, p1, p2, h_list, kernel, offset=1.0, num=100, path=None):
143     _nonparametric_method(
144         x1, x2, p1, p2,
145         h_list, 'h',
146         kernel,
147         conditional_probability_parzen,
148         offset, num, path
149         )
150 
151 
152 def kmeans(x1, x2, p1, p2, k_list, kernel, offset=1.0, num=100, path=None):
153     _nonparametric_method(
154         x1, x2, p1, p2,
155         k_list, 'k',
156         kernel,
157         conditional_probability_kmeans,
158         offset, num, path
159         )
160 
161 
162 def main():
163     # settings
164     data_path = '../data/data.mat'
165     offset = 1.0
166     num = 100
167     np.random.seed(0)
168 
169 
170     # load data
171     x1, x2 = load(data_path)
172     #print(x1.shape, x2.shape)
173     x1, x2 = x1[0], x2[0]
```

```
174
175     #plot_data(x1, x2)
176     n1, n2 = len(x1), len(x2)
177     n = n1 + n2
178     p1, p2 = n1/n, n2/n
179
180
181     # parzen
182     """
183     # kernel = normal_distribution
184     kernel = hypercube
185     h_list = [1.0, 3.0, 5.0, 10.0]
186     fig_path = '../figures/parzen_hypercube_result.png'
187
188     parzen(
189         x1, x2, p1, p2,
190         h_list, kernel,
191         offset, num,
192         fig_path
193         )
194     """
195
196
197     # kmeans
198     k_list = [2, 5, 10, 14, 20]
199     fig_path = '../figures/kmeans_result.png'
200
201     kmeans(
202         x1, x2, p1, p2,
203         k_list, None,
204         offset, num,
205         fig_path
206         )
207     #"""
208
209
210 if __name__ == '__main__':
211     main()
```