パターン認識
2019-05-07 授業分 レポート

37-196360　森田涼介

2019 年 5 月 11 日

宿題 1

2 つのクラス c_1, c_2 からそれぞれ得た標本 x_1, x_2 について，条件付確率密度 $p(x|c_i)$ を，パルゼンウィンドウ法 (カーネル関数として正規分布と超立方体両方) と k 近傍法（様々な k について）で求め，図示する。また，その事後確率 $p(c_i|x)$ も図示する。

図 1，2 にそれぞれ x_1, x_2 のヒストグラムを示す。

プログラム

プログラムの本体は 9 ページの Listing 1 に示した。以下に簡単なプログラムの説明を示す。

- load
  .mat ファイルからデータを読み出し，x_1, x_2 のデータを返す。
- plot_data
  x_1, x_2 それぞれのヒストグラムをプロットする。
- normal_distribution

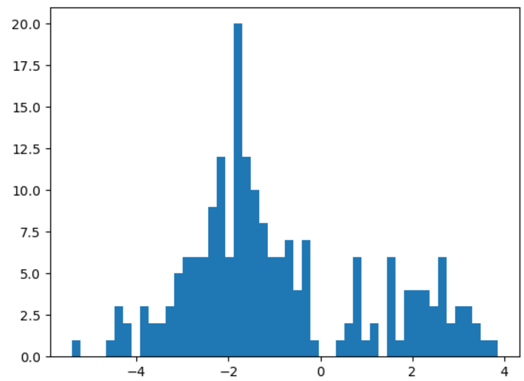

図 1: x_1 のヒストグラム

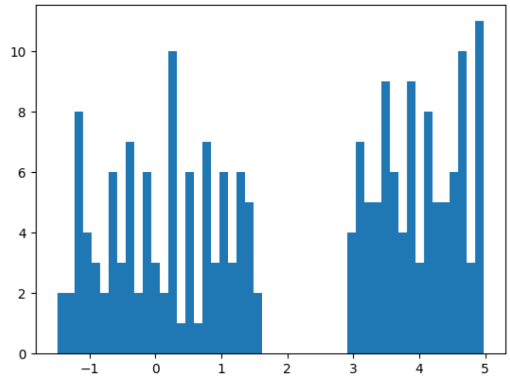

図 2: x_2 のヒストグラム

正規分布の確率密度を返す。parzen 法におけるカーネル関数として用いる。

- hypercube
  超立方体の確率密度を返す。parzen 法におけるカーネル関数として用いる。
- conditional_probability_parzen
  parzen 法における条件付確率を求める。
- conditiona_probability_kmeans
  k 近傍法における条件付確率を求める。距離関数には絶対値を用いる。
- _nonparametric_method
  parzen 法と k 近傍法で x_1, x_2 の条件付確率と事後確率を求めてプロットするときの，共通する部分をまとめた関数。
- parzen
  parzen 法で x_1, x_2 の条件付確率と事後確率を求めてプロットする。
- kmeans
  k 近傍法で x_1, x_2 の条件付確率と事後確率を求めてプロットする。
- main
  上記の関数をまとめて実行する関数。

結果

parzen 法にて正規分布をカーネル関数として用いたときの結果を図 3 に，超立方体をカーネル関数として用いたときの結果を図 4 に，k 近傍法の結果を図 5 に示す。バンド幅 h とクラスタに含まれるサンプル数 k は，図に示した値について実験した。

考察

parzen 法において正規分布をカーネル関数として用いた場合は，バンド幅 $h=3$ でもかなり滑らかな確率密度関数が得られていたが，超立方体を用いた場合は $h=10$ でもあまり滑らかにはならなかった。これは，正

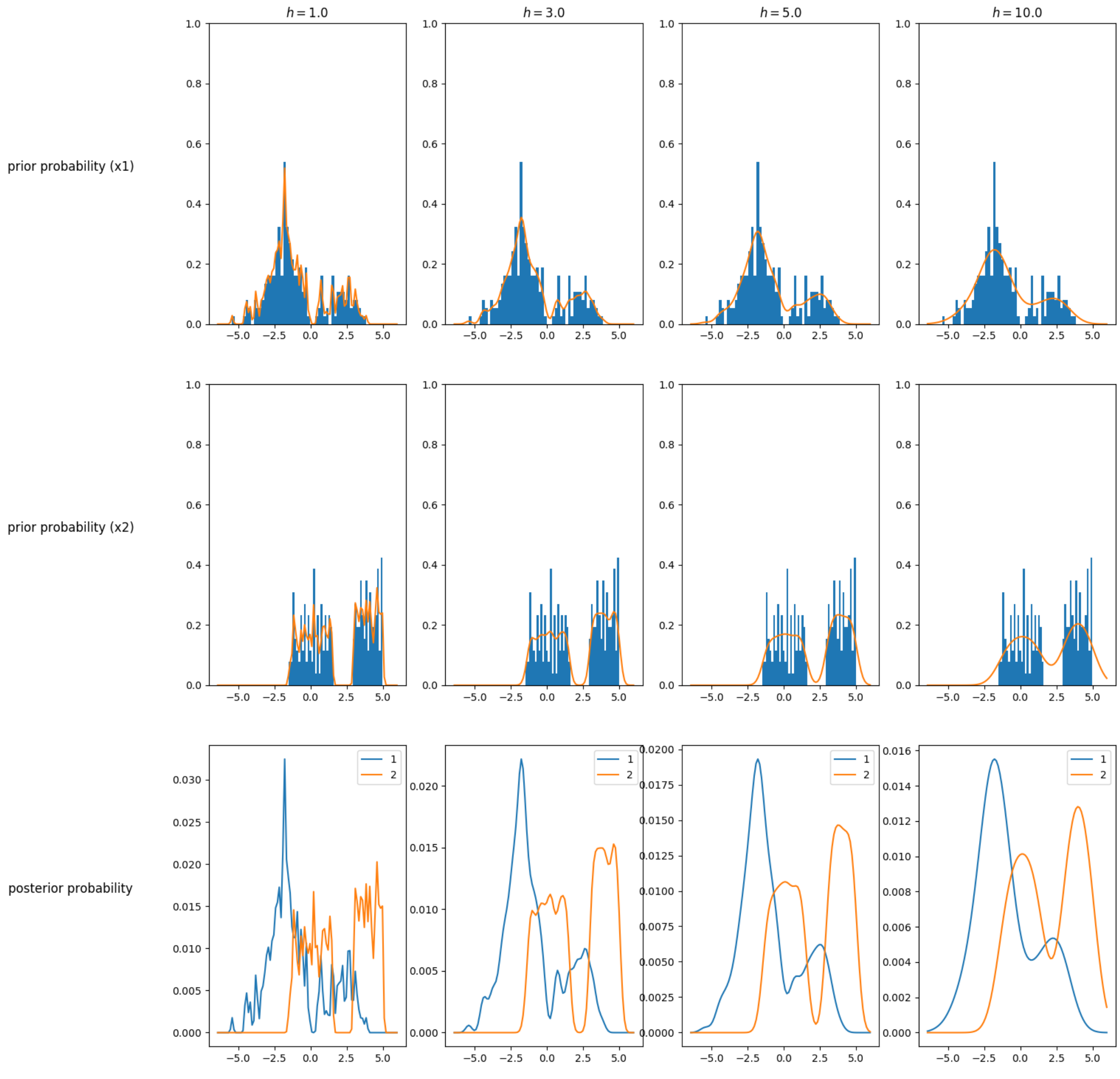

図 3: parzen 法にて正規分布をカーネル関数として用いたときの結果

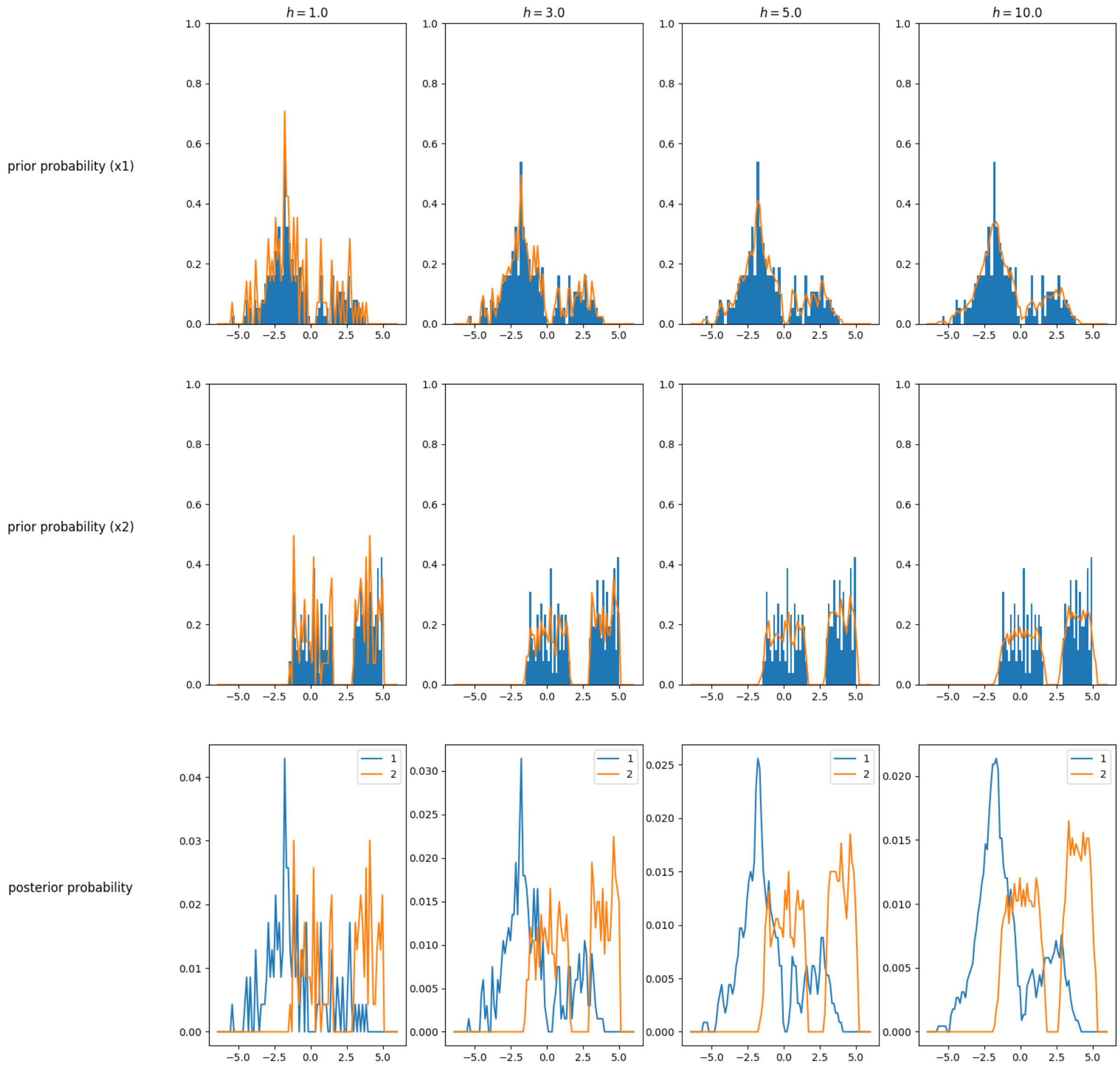

図 4: parzen 法にて超立方体をカーネル関数として用いたときの結果

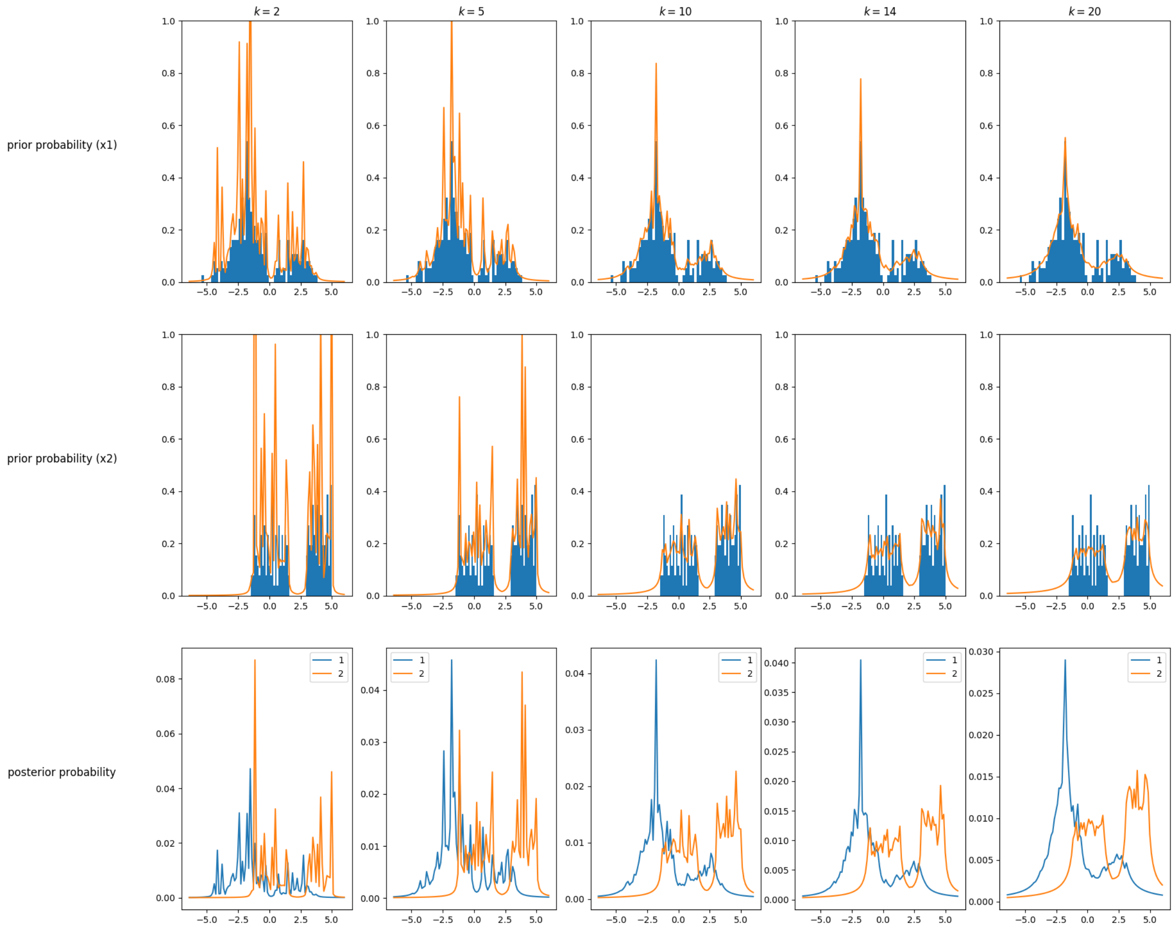

図 5: k 近傍法の結果

規分布の領域は分散の影響で x に対し滑らかにしか変化しないのに対し，超立方体はその領域の内と外をきっぱりと分けてしまう，つまり x に対し不連続に変化するため，当然ともいえる。実際，x_2 のような，不連続（に見えるよう）な分布について，バンド幅 3 のものを見れば，正規分布ではデータが存在しない部分でも確率密度関数の値が大きくなってしまっているが，超立方体ではそのような部分はほとんど見られない。逆に，x_1 のような，x に対し滑らかに変化しているような分布については，正規分布はうまく真の分布を近似しているように見えるが，超立方体ではノイズの影響が大きく出てしまっている。

k 近傍法については，k が小さいときは値が発散してしまっていて，ある程度大きくすると（$k \geq 10$），真の分布に近づけている。

宿題 2

次式を導出する。ここで，$\mathbf{Z} \in \mathbb{R}^d$, $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$, $\mu \in \mathbb{R}$ である。

$$\frac{\partial}{\partial \mathbf{Z}} \left\{ \mathbf{Z}^{\mathrm{T}} \mathbf{\Sigma}^{-1} \mathbf{Z} - \mu (\mathbf{Z}^{\mathrm{T}} \mathbf{Z} - 1) \right\} = 2\mathbf{\Sigma}^{-1} \mathbf{Z} - 2\mu \mathbf{Z} \tag{1}$$

$\mathbf{Z}$, $\mathbf{\Sigma}$ を次のようにおく。

$$\mathbf{Z} = \begin{bmatrix} z_1 \\ \vdots \\ z_d \end{bmatrix} \tag{2}$$

$$\mathbf{\Sigma}^{-1} = \mathbf{\Lambda} = \begin{bmatrix} \lambda_{11} & \cdots & \lambda_{1d} \\ \vdots & \ddots & \vdots \\ \lambda_{d1} & \cdots & \lambda dd \end{bmatrix} \tag{3}$$

このとき，

$$\begin{aligned} \mathbf{Z}^{\mathrm{T}} \mathbf{Z} &= \begin{bmatrix} z_1 & \cdots & z_d \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_d \end{bmatrix} \\ &= {z_1}^2 + \cdots + {z_d}^2 \\ &= \sum_{i=1}^{d} {z_i}^2 \end{aligned} \tag{4}$$

$$\begin{aligned} \mathbf{\Sigma}^{-1} \mathbf{Z} &= \mathbf{\Lambda} \mathbf{Z} \\ &= \begin{bmatrix} \lambda_{11} & \cdots & \lambda_{1d} \\ \vdots & \ddots & \vdots \\ \lambda_{d1} & \cdots & \lambda_{dd} \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_d \end{bmatrix} \\ &= \begin{bmatrix} \lambda_{11} z_1 + \cdots + \lambda_{1d} z_d \\ \vdots \\ \lambda_{d1} z_1 + \cdots + \lambda_{dd} z_d \end{bmatrix} \\ &= \begin{bmatrix} \sum_{j=1}^{d} \lambda_{1j} z_j \\ \vdots \\ \sum_{j=1}^{d} \lambda_{dj} z_j \end{bmatrix} \end{aligned} \tag{5}$$

$$
\begin{aligned}
\mathbf{Z}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{Z} &= \mathbf{Z}^{\mathrm{T}}\mathbf{\Lambda}\mathbf{Z} \\
&= \begin{bmatrix} z_1 & \cdots & z_d \end{bmatrix} \begin{bmatrix} \sum_{j=1}^{d} \lambda_{1j} z_j \\ \vdots \\ \sum_{j=1}^{d} \lambda_{dj} z_j \end{bmatrix} \\
&= z_1 \sum_{j=1}^{d} \lambda_{1j} z_j + \cdots + z_d \sum_{j=1}^{d} \lambda_{dj} z_j \\
&= \sum_{i=1}^{d} z_i \sum_{j=1}^{d} \lambda_{ij} z_j && (6) \\
&= \sum_{i=1}^{d} \sum_{j=1}^{d} \lambda_{ij} z_i z_j && (7) \\
&= \sum_{i=1}^{d} \lambda_{ii} {z_i}^2 + \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \lambda_{ij} z_i z_j + \sum_{j=1}^{d-1} \sum_{i=j+1}^{d} \lambda_{ij} z_i z_j && (8) \\
&= \sum_{i=1}^{d} \lambda_{ii} {z_i}^2 + 2 \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \lambda_{ij} z_i z_j && (9)
\end{aligned}
$$

である。ここで，$\mathbf{\Sigma}$ は対称行列であるから，$\mathbf{\Lambda} = \mathbf{\Sigma}^{-1}$ も対称行列であり，$\lambda_{ij} = \lambda_{ji}$ となることを用いた。このとき，

$$
\frac{\partial}{\partial z_k}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\right) = \frac{\partial}{\partial z_k}\left(\sum_{i=1}^{d} {z_k}^2\right) = 2 z_k \tag{10}
$$

$$
\begin{aligned}
\frac{\partial}{\partial z_k}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{Z}\right) &= \frac{\partial}{\partial z_k}\left(\sum_{i=1}^{d} \sum_{j=1}^{d} \lambda_{ij} z_i z_j\right) && (11) \\
&= \frac{\partial}{\partial z_k}\left(\sum_{i=1}^{d} \lambda_{ii} {z_i}^2 + 2 \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \lambda_{ij} z_i z_j\right) && (12) \\
&= 2\lambda_{kk} z_k + 2\left(\sum_{j=k+1}^{d} \lambda_{kj} z_j + \sum_{i=1}^{k-1} \lambda_{ik} z_i\right) && (13) \\
&= 2 \sum_{l=1}^{d} \lambda_{kl} z_l && (14)
\end{aligned}
$$

であるから，

$$
\frac{\partial}{\partial \mathbf{Z}}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\right) = \begin{bmatrix} \frac{\partial}{\partial z_1}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\right) \\ \vdots \\ \frac{\partial}{\partial z_d}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\right) \end{bmatrix} = \begin{bmatrix} 2z_1 \\ \vdots \\ 2z_d \end{bmatrix} = 2\mathbf{Z} \tag{15}
$$

$$
\begin{aligned}
\frac{\partial}{\partial \mathbf{Z}}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{Z}\right) &= \begin{bmatrix} \frac{\partial}{\partial z_1}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{Z}\right) \\ \vdots \\ \frac{\partial}{\partial z_d}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{Z}\right) \end{bmatrix} && (16) \\
&= \begin{bmatrix} 2\sum_{l=1}^{d}\lambda_{1l}z_l \\ \vdots \\ 2\sum_{l=1}^{d}\lambda_{dl}z_l \end{bmatrix} && (17) \\
&= 2\boldsymbol{\Sigma}^{-1}\mathbf{Z} && (18)
\end{aligned}
$$

となる。また，当然 $\partial\mu\big/\partial\mathbf{Z} = \mathbf{0}$ である。以上より，

$$
\begin{aligned}
\frac{\partial}{\partial \mathbf{Z}}\left\{\mathbf{Z}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{Z} - \mu(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} - 1)\right\} &= \frac{\partial}{\partial \mathbf{Z}}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{Z}\right) - \mu\frac{\partial}{\partial \mathbf{Z}}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\right) && (19) \\
&= 2\boldsymbol{\Sigma}^{-1}\mathbf{Z} - 2\mu\mathbf{Z} && (20)
\end{aligned}
$$

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


```

```
def hypercube(x, mu, h):
    p = 1/h * (np.abs((x - mu)/h) < 1/2)
    return p


def conditional_probability_parzen(x, h, x_axis, kernel):
    n = len(x)
    hn = h/np.sqrt(n)
    prob = np.zeros(len(x_axis))
    for x_i in x:
        prob += kernel(x_axis, x_i, hn)
    prob = prob / n
    return prob


def conditional_probability_kmeans(x, k, x_axis, kernel):
    n = len(x)
    # k = np.sqrt(n)
    # kn = k/np.sqrt(n)
    prob = np.zeros(len(x_axis))

    for i in range(len(x_axis)):
        # r: sorted list by the distance to x[j]
        r = sorted(abs(x - x_axis[i]))

        # r[int(k)-1]: k-th distance
        prob[i] = k / (n * 2 * r[int(k)-1])

    return prob


def _nonparametric_method(
        x1, x2, p1, p2,
        param_candidates, param_str,
        kernel,
        conditional_probability,
        offset=1.0, num=100,
        path=None):
    x_min = min(x1.min(), x2.min()) - offset
    x_max = max(x1.max(), x2.max()) + offset
    x_axis = np.linspace(x_min, x_max, num)

    n_params = len(param_candidates)
    n_row = 3
    n_col = n_params + 1
    fig = plt.figure(figsize=(n_col*4, n_row*6))
    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])

```

```
        ax_2 = fig.add_subplot(n_row, n_col, n_col+(i+1)+1)
        ax_2.hist(x2, bins=50, normed=True)
        ax_2.plot(x_axis, p2_cond)
        ax_2.set_ylim([0, 1.0])

        ax = fig.add_subplot(n_row, n_col, 2*n_col+(i+1)+1)
        ax.plot(x_axis, p1_post, label='1')
        ax.plot(x_axis, p2_post, label='2')
        ax.legend()
        # ax.set_ylim([0, 1.0])

    plt.savefig(str(path))
    plt.show()


def parzen(x1, x2, p1, p2, h_list, kernel, offset=1.0, num=100, path=None):
    _nonparametric_method(
        x1, x2, p1, p2,
        h_list, 'h',
        kernel,
        conditional_probability_parzen,
        offset, num, path
        )


def kmeans(x1, x2, p1, p2, k_list, kernel, offset=1.0, num=100, path=None):
    _nonparametric_method(
        x1, x2, p1, p2,
        k_list, 'k',
        kernel,
        conditional_probability_kmeans,
        offset, num, path
        )


def main():
    # settings
    data_path = '../data/data.mat'
    offset = 1.0
    num = 100
    np.random.seed(0)


    # load data
    x1, x2 = load(data_path)
    #print(x1.shape, x2.shape)
    x1, x2 = x1[0], x2[0]
```

```

    #plot_data(x1, x2)
    n1, n2 = len(x1), len(x2)
    n = n1 + n2
    p1, p2 = n1/n, n2/n


    # parzen
    """
    # kernel = normal_distribution
    kernel = hypercube
    h_list = [1.0, 3.0, 5.0, 10.0]
    fig_path = '../figures/parzen_hypercube_result.png'

    parzen(
        x1, x2, p1, p2,
        h_list, kernel,
        offset, num,
        fig_path
        )
    """


    # kmeans
    k_list = [2, 5, 10, 14, 20]
    fig_path = '../figures/kmeans_result.png'

    kmeans(
        x1, x2, p1, p2,
        k_list, None,
        offset, num,
        fig_path
        )
    #"""


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