先端データ解析論
第二回　レポート

37-196360　森田涼介

2019 年 4 月 25 日

期限内提出では宿題 2 ができていなかったのですが，遅ればせながら解けましたので提出します。よろしくお願いいたします。

宿題 1

ガウスカーネルモデルに対する L2 正則化を用いた最小二乗回帰の交差確認法を実装し，正則化パラメータとガウス幅を決定する。

ガウスカーネルモデルは次のように表される。

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x},\, \boldsymbol{x}_j) \tag{1}$$

$$K(\boldsymbol{x}, \boldsymbol{c}) = \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{c}||^2}{2h^2}\right) \tag{2}$$

このとき，L2 正則化を用いた最小二乗誤差は，

$$L(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{K\theta}-\boldsymbol{y}||^2 + \frac{\lambda}{2}||\boldsymbol{\theta}||^2 \tag{3}$$

$$\boldsymbol{K} = \begin{bmatrix} K(\boldsymbol{x}_1,\, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1,\, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n,\, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_n,\, \boldsymbol{x}_n) \end{bmatrix} \tag{4}$$

と表される。ここで，L の $\boldsymbol{\theta}$ による偏微分が $\boldsymbol{0}$ となるような $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ が L に最小値を与えることから，

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \boldsymbol{K}^{\mathrm{T}}\boldsymbol{K\theta} - \boldsymbol{K}^{\mathrm{T}}\boldsymbol{y} + \lambda\boldsymbol{\theta} \tag{5}$$

$$= (\boldsymbol{K}^2 + \lambda\boldsymbol{I})\boldsymbol{\theta} - \boldsymbol{Ky} \tag{6}$$

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{K}^2 + \lambda\boldsymbol{I})^{-1}\boldsymbol{Ky} \tag{7}$$

ここで，$\boldsymbol{K}$ が対称行列であることを用いた。

いま，真のデータが

$$f(x) = \sin(\pi x)/(\pi x) + 0.1x \quad (-3 \leq x < 3) \tag{8}$$

から生成されていて，そのサンプルが 50 個ある場合を考える。このデータに対しガウスカーネルモデルを適用し，交差確認法によってそのハイパーパラメータ $h,\, \lambda$ を推定する。いま，交差確認法のために，データを 5 分割し，それぞれのテスト誤差の平均をそのモデルのテスト誤差とした。また，テスト誤差には L2 正則化項は含まず，y の推定値と真の値の二乗和誤差のみを用いた。

まず，大域的に探索するため，

$$h = 1\times10^{-2},\ 1\times10^{-1},\ 1,\ 1\times10^{1} \tag{9}$$

$$\lambda = 1\times10^{-6},\ 1\times10^{-5},\ 1\times10^{-4},\ 1\times10^{-3},\ 1\times10^{-2},\ 1\times10^{-1} \tag{10}$$

として，各 $(h,\, \lambda)$ の組について交差確認法を用いて誤差を計算したところ，

$$h = 1 \tag{11}$$

$$\lambda = 1\times10^{-5} \tag{12}$$

$$L = 1.106 \tag{13}$$

と求まった。次に，求まった範囲の付近で最適なものを探したところ，

$$h = 1.1 \tag{14}$$
$$\lambda = 8 \times 10^{-7} \tag{15}$$
$$L = 1.04 \tag{16}$$

と求まった。

大域的に探索したときの結果を図 1 に示す。これより，ガウス幅は 0.1–1.0 あたりに設定する必要があることがわかる。また，プログラムは 6 ページの Listing 1 に記載した。

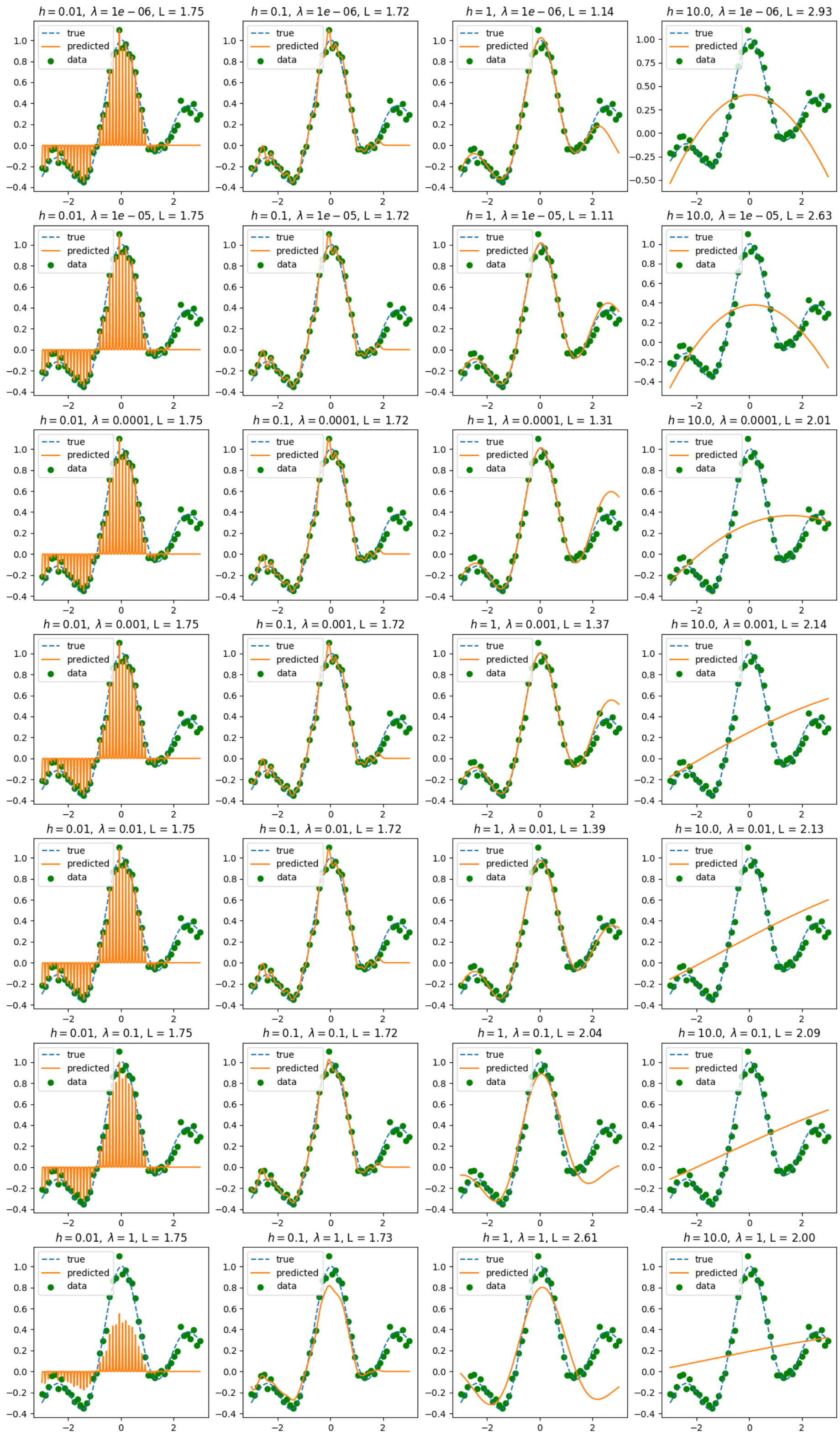
$h = 0.01,\ \lambda = 1e-06$, L = 1.75
$h = 0.1,\ \lambda = 1e-06$, L = 1.72
$h = 1,\ \lambda = 1e-06$, L = 1.14
$h = 10.0,\ \lambda = 1e-06$, L = 2.93
$h = 0.01,\ \lambda = 1e-05$, L = 1.75
$h = 0.1,\ \lambda = 1e-05$, L = 1.72
$h = 1,\ \lambda = 1e-05$, L = 1.11
$h = 10.0,\ \lambda = 1e-05$, L = 2.63
$h = 0.01,\ \lambda = 0.0001$, L = 1.75
$h = 0.1,\ \lambda = 0.0001$, L = 1.72
$h = 1,\ \lambda = 0.0001$, L = 1.31
$h = 10.0,\ \lambda = 0.0001$, L = 2.01
$h = 0.01,\ \lambda = 0.001$, L = 1.75
$h = 0.1,\ \lambda = 0.001$, L = 1.72
$h = 1,\ \lambda = 0.001$, L = 1.37
$h = 10.0,\ \lambda = 0.001$, L = 2.14
$h = 0.01,\ \lambda = 0.01$, L = 1.75
$h = 0.1,\ \lambda = 0.01$, L = 1.72
$h = 1,\ \lambda = 0.01$, L = 1.39
$h = 10.0,\ \lambda = 0.01$, L = 2.13
$h = 0.01,\ \lambda = 0.1$, L = 1.75
$h = 0.1,\ \lambda = 0.1$, L = 1.72
$h = 1,\ \lambda = 0.1$, L = 2.04
$h = 10.0,\ \lambda = 0.1$, L = 2.09
$h = 0.01,\ \lambda = 1$, L = 1.75
$h = 0.1,\ \lambda = 1$, L = 1.73
$h = 1,\ \lambda = 1$, L = 2.61
$h = 10.0,\ \lambda = 1$, L = 2.00
true
predicted
data

図 1: 交差確認法の結果

宿題 2

線形モデル $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x})$ を用いた L2 正則化回帰に，一つ抜き交差確認法を適用したときの二乗誤差 L が次式のように表されることを示す。

$$L = \frac{1}{n}||\tilde{\boldsymbol{H}}^{-1}\boldsymbol{H}\boldsymbol{y}||^2 \tag{17}$$

$$\boldsymbol{H} = \boldsymbol{I} - \boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}} \tag{18}$$

$$\tilde{\boldsymbol{H}} : \boldsymbol{H} \text{ と同じ対角成分を持ち，非対角成分は } 0 \tag{19}$$

訓練誤差 L_{train} は，

$$L_{\mathrm{train}} = ||\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y}||^2 + \frac{\lambda}{2}||\boldsymbol{\theta}||^2 \tag{20}$$

これを最小にする $\boldsymbol{\theta}$ は，上式の $\boldsymbol{\theta}$ による偏微分が 0 である条件から得られて，

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} \tag{21}$$

となる。これより，標本 $(\boldsymbol{x}_i, y_i)$ を除いた標本群を用いて学習したときに得られるパラメータ $\boldsymbol{\theta}$ は，

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_i &= (\boldsymbol{\Phi}_i^{\mathrm{T}}\boldsymbol{\Phi}_i + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}_i^{\mathrm{T}}\boldsymbol{y}_i && (22)\\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I} - \boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}})^{-1}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\phi}_i y_i) && (23)\\
&= (\boldsymbol{U} - \boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}})^{-1}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\phi}_i y_i) && (24)
\end{aligned}$$

と表される。ここで，

$$\boldsymbol{U} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I} \tag{25}$$

とおいた。また，ShermanMorrison-Woodbury 公式を用いると，

$$(\boldsymbol{U} - \boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}})^{-1} = \boldsymbol{U}^{-1} + \frac{\boldsymbol{U}^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}}{1 - \boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\phi}_i} \tag{26}$$

となることから，結局，

$$\hat{\boldsymbol{\theta}}_i = \left(\boldsymbol{U}^{-1} + \frac{\boldsymbol{U}^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}}{1 - \boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\phi}_i}\right)(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\phi}_i y_i) \tag{27}$$

これより，$\alpha_i = \boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\phi}_i$ とおくと，y_i の予測値 $\hat{y}_i$ は，

$$\begin{aligned}
\hat{y}_i &= \boldsymbol{\phi}_i^{\mathrm{T}}\hat{\boldsymbol{\theta}}_i && (28)\\
&= \boldsymbol{\phi}_i^{\mathrm{T}}\left(\boldsymbol{U}^{-1} + \frac{\boldsymbol{U}^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}}{1 - \alpha_i}\right)(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\phi}_i y_i) && (29)\\
&= \frac{1}{1-\alpha_i}\left(\boldsymbol{\phi}_i^{\mathrm{T}}(1-\alpha_i)\boldsymbol{U}^{-1} + \boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\right)(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\phi}_i y_i) && (30)\\
&= \frac{1}{1-\alpha_i}\left(\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1} - \alpha_i\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1} + \alpha_i\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\right)(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\phi}_i y_i) && (31)\\
&= \frac{\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}}{1-\alpha_i}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\phi}_i y_i) && (32)\\
&= \frac{\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - \alpha_i y_i}{1-\alpha_i} && (33)
\end{aligned}$$

よって，テスト誤差 L は，

$$
\begin{aligned}
L &= \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 && (34)\\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - \alpha_i y_i}{1-\alpha_i} - y_i\right) && (35)\\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - y_i}{1-\alpha_i}\right) && (36)
\end{aligned}
$$

となる。

また，$\boldsymbol{H}$ と $\tilde{\boldsymbol{H}}$ について，

$$
\begin{aligned}
\boldsymbol{H} &= \boldsymbol{I} - \boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}} && (37)\\
&= \boldsymbol{I} - \boldsymbol{\Phi}\boldsymbol{U}^{-1}\boldsymbol{\Phi}^{\mathrm{T}} && (38)\\
&= \boldsymbol{I} - \begin{bmatrix} \boldsymbol{\phi}_1^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_1^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\phi}_n \\ \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_n^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_n^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\phi}_n \end{bmatrix} && (39)
\end{aligned}
$$

より，$\boldsymbol{H}$（及び $\tilde{\boldsymbol{H}}$）の i 番目対角成分は $1-\alpha_i$，$\tilde{\boldsymbol{H}}^{-1}$ の i 番目対角成分は $1/(1-\alpha_i)$ となる。また，

$$
\begin{aligned}
\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - y_i &= \sum_{j=1}^{n}\left(\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\phi}_j y_j\right) - y_i && (40)\\
&= -(\boldsymbol{H}\boldsymbol{y})_i && (41)
\end{aligned}
$$

である。以上より，テスト誤差は，

$$
\begin{aligned}
L &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{U}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} - y_i}{1-\alpha_i}\right) && (42)\\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{-(\boldsymbol{H}\boldsymbol{y})_i}{1-\alpha_i}\right)^2 && (43)\\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{\boldsymbol{H}}_i^{-1}(\boldsymbol{H}\boldsymbol{y})_i\right)^2 && (44)\\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{H}\boldsymbol{y}\right)_i^2 && (45)\\
&= \frac{1}{n}||\tilde{\boldsymbol{H}}^{-1}\boldsymbol{H}\boldsymbol{y}||^2 && (46)
\end{aligned}
$$

となる。

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 true_model(x):
    pix = np.pi * x
    target = np.sin(pix) / pix + 0.1 * x
    return target


def gauss_kernel(x, c, h):
    return np.exp(-(x - c)**2 / (2*h**2))


def generate_sample(xmin, xmax, sample_size):
    x = np.linspace(start=xmin, stop=xmax, num=sample_size)
    target = true_model(x)
    noise = 0.05 * np.random.normal(loc=0., scale=1., size=sample_size)
    return x, target + noise


def split(x, y, n_split=5):
    n_data = len(y)
    n_data_in_one_split = int(n_data / n_split)
    idx = np.arange(n_data)
    np.random.shuffle(idx)

    x_split = []
    y_split = []
```

```
    for i in range(n_split):
        idx_start = i * n_data_in_one_split
        idx_end = (i+1) * n_data_in_one_split
        if idx_end == n_data:
            idx_end = None
        x_split.append(x[idx_start:idx_end])
        y_split.append(y[idx_start:idx_end])
    return x_split, y_split


def split_train_test(x_split, y_split, k):
    n_split = len(y_split)
    x_test, y_test = x_split[k], y_split[k]
    x_train, y_train = [], []
    for _k in range(n_split):
        if _k != k:
            x_train.extend(x_split[_k])
            y_train.extend(y_split[_k])
    x_train = np.array(x_train)
    y_train = np.array(y_train)
    return x_train, y_train, x_test, y_test


def calc_design_matrix(x, c, h, kernel):
    return kernel(x[None], c[:, None], h)


def solve_gauss_kernel_model(x, y, h, lamb):
    k = calc_design_matrix(x, x, h, gauss_kernel)
    theta = np.linalg.solve(
        k.T.dot(k) + lamb*np.identity(len(k)),
        k.T.dot(y[:, None]),
        )
    return theta


def compute_loss(x_train, x_test, y, h, theta, lamb):
    k = calc_design_matrix(x_train, x_test, h, gauss_kernel)
    loss = (1/2)*np.linalg.norm(k.dot(theta) - y)
    # loss += (lamb/2)*np.linalg.norm(theta)
    return loss


def main():
    np.random.seed(0)  # set the random seed for reproducibility

    # create sample
```

```
    xmin, xmax = -3, 3
    sample_size = 50
    n_split = 5
    x, y = generate_sample(xmin=xmin, xmax=xmax, sample_size=sample_size)
    # print(x.shape, y.shape)

    x_split, y_split = split(x, y, n_split=n_split)
    # print(x_split[0].shape, y_split[0].shape)

    # global search
    h_cands = [1e-2, 1e-1, 1, 1e1,]
    lamb_cands = [1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1]

    # local search
    #searched_range_base = np.arange(0.5, 1.5, 0.1)
    #h_cands = 1.0 * searched_range_base
    #lamb_cands = 1e-6 * searched_range_base

    loss_min = 1e8
    h_best = None
    lamb_best = None
    theta_best = None

    n_row = len(lamb_cands)
    n_col = len(h_cands)
    fig = plt.figure(figsize=(n_col*4, n_row*4))
    fig_idx = 0

    for lamb in lamb_cands:
        for h in h_cands:
            losses = []
            for k in range(n_split):
                x_train, y_train, x_test, y_test = split_train_test(x_split, y_split, k)
                # print(x_train.shape, y_train.shape)

                theta = solve_gauss_kernel_model(x_train, y_train, h, lamb)
                loss_k = compute_loss(x_train, x_test, y_test, h, theta, lamb)
                losses.append(loss_k)
            loss = np.mean(losses)

            if loss < loss_min:
                loss_min = loss
                h_best = h
                lamb_best = lamb
                theta_best = theta
```

```
            # for visualization
            X = np.linspace(start=xmin, stop=xmax, num=5000)
            true = true_model(X)
            K = calc_design_matrix(x_train, X, h, gauss_kernel)
            prediction = K.dot(theta)

            # visualization
            fig_idx += 1
            ax = fig.add_subplot(n_row, n_col, fig_idx)
            ax.set_title('$h = {},\ \lambda = {}$, L = {:.2f}'.format(h, lamb, loss))
            ax.scatter(x, y, c='green', marker='o', label='data')
            ax.plot(X, true, linestyle='dashed', label='true')
            ax.plot(X, prediction, linestyle='solid', label='predicted')
            ax.legend()

    print('h = {}'.format(h_best))
    print('lambda = {}'.format(lamb_best))
    print('loss = {}'.format(loss_min))

    plt.savefig('../figures/assignment1_result.png')
    plt.show()


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