先端データ解析論
第三回　レポート

37-196360　森田涼介

2019 年 4 月 26 日

宿題 1

$$T(z) = \lambda |z| + u(\theta - z) + \frac{1}{2}(\theta - z)^2 \qquad (\lambda \geq 0) \tag{1}$$

について，

$$\arg\min_z T(z) = \max(0,\ \theta + u - \lambda) + \min(0,\ \theta + u + \lambda) \tag{2}$$

であることを証明する。

$$\frac{\partial T}{\partial z} = \lambda \mathrm{sign}(z) - u + (z - \theta) \tag{3}$$

$$= z - (\theta + u - \lambda \mathrm{sign}(z)) \tag{4}$$

$$\frac{\partial^2 T}{\partial z^2} = 1 \quad (> 0) \tag{5}$$

より，T は下に凸な関数である。以下では z の正負について場合分けをし，T に最小を与える z を求める。

$z \geq 0$ のとき，

$$\frac{\partial T}{\partial z} = z - (\theta + u - \lambda) \tag{6}$$

1. $\theta + u \leq \lambda$ のとき
   $\forall z \geq 0$ に対して $\partial T / \partial z \geq 0$ が成立し，T は増加関数となるので，T に最小を与える $z = 0$ となる。
2. $\theta + u \geq \lambda$ のとき
   T は下に凸な関数であるから，$\partial T / \partial z = 0$ を与える z が T に最小を与える。よって $z = \theta + u - \lambda$ となる。

これをまとめると，

$$\arg\min_z T(z) = \max(0,\ \theta + u - \lambda) \tag{7}$$

$z \leq 0$ のとき，

$$\frac{\partial T}{\partial z} = z - (\theta + u + \lambda) \tag{8}$$

1. $\theta + u \geq -\lambda$ のとき
   $\forall z \leq 0$ に対して $\partial T / \partial z \leq 0$ が成立し，T は減少関数となるので，T に最小を与える $z = 0$ となる。
2. $\theta + u \leq -\lambda$ のとき
   $\partial T / \partial z = 0$ を与える $z = \theta + u + \lambda$ が T に最小を与える。

これをまとめると，

$$\arg\min_z T(z) = \min(0,\ \theta + u + \lambda) \tag{9}$$

以上をグラフで表すと，図 1 のようになる。

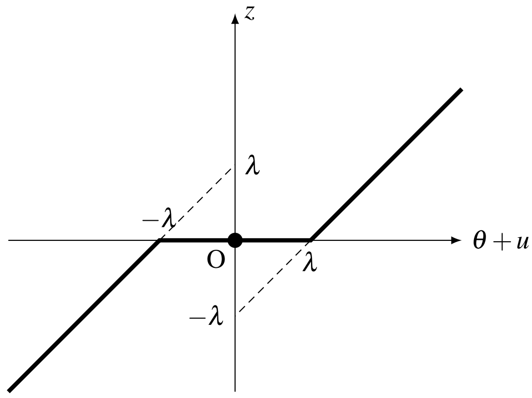

図 1: $z - \theta + u$ 曲線

以上より，

$$\arg\min_z T(z) = \max(0,\ \theta + u - \lambda) + \min(0,\ \theta + u + \lambda) \tag{10}$$

であることがわかる。

宿題 2

ガウスカーネルモデルに対して，スパース回帰の交互方向乗数法による反復式を求め，また適当なモデルに対してスパース回帰を実行する。

ガウスカーネルモデルは次のように表される。

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x},\, \boldsymbol{x}_j) \tag{11}$$

$$K(\boldsymbol{x}, \boldsymbol{c}) = \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{c}||^2}{2h^2}\right) \tag{12}$$

このとき，L1 正則化を用いた最小二乗誤差は，

$$J(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{K}\boldsymbol{\theta}-\boldsymbol{y}||^2 + \lambda||\boldsymbol{\theta}||_1 \tag{13}$$

$$\boldsymbol{K} = \begin{bmatrix} K(\boldsymbol{x}_1,\, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1,\, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n,\, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_n,\, \boldsymbol{x}_n) \end{bmatrix} \tag{14}$$

と表される。このとき，最適なパラメータ $\hat{\boldsymbol{\theta}}$ は次のように求まる。

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}}(J) \tag{15}$$

J に最小を与える $\boldsymbol{\theta}$ を，以下の設定で交互方向乗数法を用いることで求める。すなわち，

$$\boldsymbol{\theta} - \boldsymbol{z} = \boldsymbol{0} \tag{16}$$

$$l(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{K}\boldsymbol{\theta}-\boldsymbol{y}||^2 \tag{17}$$

$$g(\boldsymbol{z}) = \lambda||\boldsymbol{z}||_1 \tag{18}$$

の下で，

$$\min_{\boldsymbol{\theta},\, \boldsymbol{z}}[l(\boldsymbol{\theta}) + g(\boldsymbol{z})] \tag{19}$$

を与える $\boldsymbol{\theta}$, $\boldsymbol{z}$ を求める。このとき，拡張ラグランジュ関数は，

$$\begin{aligned} L(\boldsymbol{\theta},\, \boldsymbol{z},\, \boldsymbol{u}) &= l(\boldsymbol{\theta}) + g(\boldsymbol{z}) + \boldsymbol{u}^{\mathrm{T}}(\boldsymbol{\theta}-\boldsymbol{z}) + \frac{1}{2}||\boldsymbol{\theta}-\boldsymbol{z}||^2 & (20) \\ &= \frac{1}{2}||\boldsymbol{K}\boldsymbol{\theta}-\boldsymbol{y}||^2 + \lambda||\boldsymbol{z}||_1 + \boldsymbol{u}^{\mathrm{T}}(\boldsymbol{\theta}-\boldsymbol{z}) + \frac{1}{2}||\boldsymbol{\theta}-\boldsymbol{z}||^2 & (21) \end{aligned}$$

であるから，$\boldsymbol{K}$ は対称行列であることに注意すると，

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{\theta}} &= \boldsymbol{K}^2\boldsymbol{\theta} - \boldsymbol{K}\boldsymbol{y} + \boldsymbol{u} + (\boldsymbol{\theta}-\boldsymbol{z}) & (22) \\ &= (\boldsymbol{K}^2 + \boldsymbol{I})\boldsymbol{\theta} - \boldsymbol{z} + \boldsymbol{u} - \boldsymbol{K}\boldsymbol{y} & (23) \\ \frac{\partial L}{\partial \boldsymbol{z}} &= \lambda\frac{\partial||\boldsymbol{z}||_1}{\partial \boldsymbol{z}} - \boldsymbol{u} + (\boldsymbol{z}-\boldsymbol{\theta}) & (24) \\ &= \boldsymbol{z} - (\boldsymbol{\theta} + \boldsymbol{u} - \lambda\,\mathrm{sign}(\boldsymbol{z})) & (25) \end{aligned}$$

となる。よって，更新式は，

$$\boldsymbol{\theta}^{(t+1)} = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta},\ \boldsymbol{z}^{(t)},\ \boldsymbol{u}^{(t)}) \tag{26}$$

$$= (\boldsymbol{K}^2 + \boldsymbol{I})^{-1}(\boldsymbol{K}\boldsymbol{y} + \boldsymbol{z}^{(t)} - \boldsymbol{u}^{(t)}) \tag{27}$$

$$\boldsymbol{z}^{(t+1)} = \max(\boldsymbol{0},\ \boldsymbol{\theta}^{(t+1)} + \boldsymbol{u}^{(t)} - \lambda) + \min(\boldsymbol{0},\ \boldsymbol{\theta}^{(t+1)} + \boldsymbol{u}^{(t)} + \lambda) \tag{28}$$

$$\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} + \boldsymbol{\theta}^{(t+1)} - \boldsymbol{z}^{(t+1)} \tag{29}$$

となる。

いま，真のデータが

$$f(x) = \sin(\pi x)/(\pi x) + 0.1x \quad (-3 \leq x < 3) \tag{30}$$

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 L1 正則化付き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, λ 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 L1 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。

h, λ の値の候補は次の通りに設定した。

$$h = 1\times10^{-2},\ 1\times10^{-1},\ 1,\ 1\times10^{1} \tag{31}$$

$$\lambda = 1\times10^{-6},\ 1\times10^{-5},\ 1\times10^{-4},\ 1\times10^{-3},\ 1\times10^{-2},\ 1\times10^{-1} \tag{32}$$

また，ラグランジュ関数の値の変化（最大値と最小値の差）が直近 10 イタレーションで 1×10^{-3} より小さくなったときに更新を止めるように設定した。これは，前回の課題からテスト誤差は最適解付近で 1 のオーダーであることが分かっていて，誤差 0.1% 以内ならば収束しているとみなせると考えられるからである。その結果，最適な h, λ 及びこれらが与えるテスト誤差 L は次のように求まった。

$$h = 1.0 \tag{33}$$

$$\lambda = 1\times10^{-4} \tag{34}$$

$$L = 1.31 \tag{35}$$

交差確認法の結果を図 2 に示す。また，プログラムは 6 ページの Listing 1 に記載した。

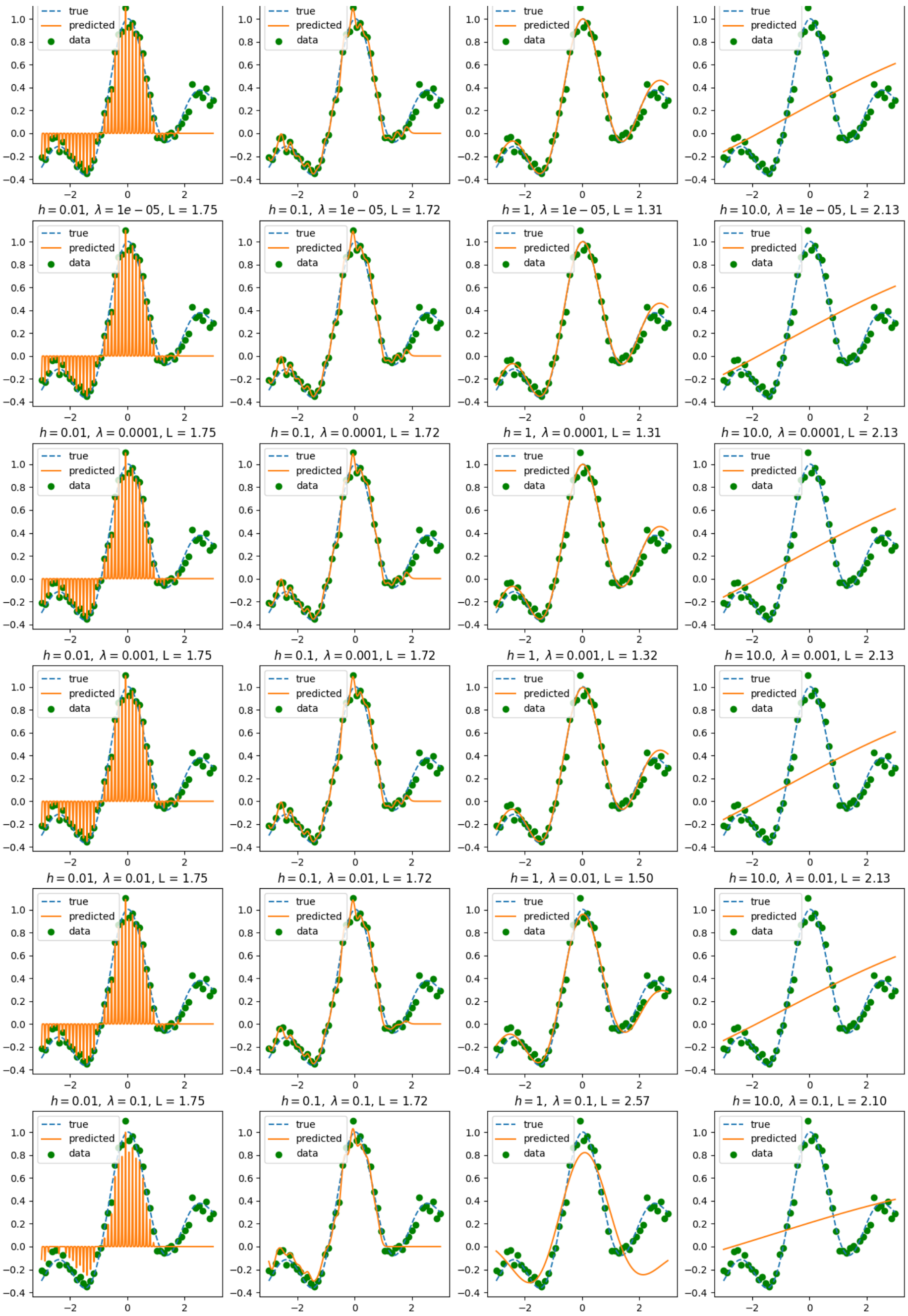
true
predicted
data
h = 0.01, λ = 1e − 05, L = 1.75
h = 0.1, λ = 1e − 05, L = 1.72
h = 1, λ = 1e − 05, L = 1.31
h = 10.0, λ = 1e − 05, L = 2.13
h = 0.01, λ = 0.0001, L = 1.75
h = 0.1, λ = 0.0001, L = 1.72
h = 1, λ = 0.0001, L = 1.31
h = 10.0, λ = 0.0001, L = 2.13
h = 0.01, λ = 0.001, L = 1.75
h = 0.1, λ = 0.001, L = 1.72
h = 1, λ = 0.001, L = 1.32
h = 10.0, λ = 0.001, L = 2.13
h = 0.01, λ = 0.01, L = 1.75
h = 0.1, λ = 0.01, L = 1.72
h = 1, λ = 0.01, L = 1.50
h = 10.0, λ = 0.01, L = 2.13
h = 0.01, λ = 0.1, L = 1.75
h = 0.1, λ = 0.1, L = 1.72
h = 1, λ = 0.1, L = 2.57
h = 10.0, λ = 0.1, L = 2.10

図 2: 交差確認法の結果

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 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 solve_gauss_kernel_sparse_model(x, y, h, lamb, eps=1e-2, iter_max=100, n_look=5):
    def update(theta, z, u, lamb, K, y):
        theta = np.linalg.inv(K.T.dot(K)+ np.eye(len(K))).dot(K.dot(y[:, None]) + z - u)

        z_plus = theta + u - lamb
        z_plus[z_plus < 0] = 0
        z_minus = theta + u + lamb
        z_minus[z_minus > 0] = 0
        z = z_plus + z_minus
```

```
        u = u + theta - z
        return theta, z, u

    def compute_lagrange_func(theta, z, u, lamb, K, y):
        loss = (
            (1/2) * np.linalg.norm(K.dot(theta) - y)
            + lamb * np.linalg.norm(z, ord=1)
            + u.T.dot(theta - z)
            + (1/2) * np.linalg.norm(theta - z)
            )
        return loss

    K = calc_design_matrix(x, x, h, gauss_kernel)
    theta = np.linalg.solve(K.T.dot(K), K.T.dot(y[:, None]))
    z = np.random.rand(*theta.shape) * 0.1
    u = np.random.rand(*theta.shape) * 0.1
    lagrange_list = []
    for i in range(iter_max):
        theta, z, u = update(theta, z, u, lamb, K, y)
        lagrange = compute_lagrange_func(theta, z, u, lamb, K, y)
        if i < n_look:
            lagrange_list.append(lagrange)
        else:
            lagrange_list = lagrange_list[1:] + [lagrange]
            if max(lagrange_list) - min(lagrange_list) < eps:
                break
    return theta, z, u, i+1


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,]

    # 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_iter_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, z, u, n_iter = solve_gauss_kernel_sparse_model(
                    x_train, y_train, h, lamb,
                    eps=1e-3, iter_max=200, n_look=10,
                    )
                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
                n_iter_best = n_iter

```

```
            # 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('Best Model')
    print('\th = {}'.format(h_best))
    print('\tlambda = {}'.format(lamb_best))
    print('\tloss = {}'.format(loss_min))
    print('\tn_iter: {}'.format(n_iter_best))
    print('\ttheta: \n', theta_best)

    plt.savefig('../figures/assignment2_result.png')
    plt.show()


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