# 先端データ解析論レポート第四回

48196635 桑原亮介

2019 年 5 月 10 日

## 1　宿題 1

目的：重み付き最小二乗法

$$min_{\theta}\frac{1}{2}\sum_{i=1}^{n}\tilde{w}_i\left(f_{\theta}(\boldsymbol{x_i}-\boldsymbol{y_i})^2\right)$$

の解が次式で与えられることの証明

$$\hat{\boldsymbol{\theta}}=\left(\boldsymbol{\Phi^T}\tilde{\boldsymbol{W}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi^T}\tilde{\boldsymbol{W}}\boldsymbol{y}$$

証明:
最小二乗誤差の式を行列で表すと、

$$E=\frac{1}{2}\left(\boldsymbol{\Phi\theta}-\boldsymbol{y}\right)^T\tilde{\boldsymbol{W}}\left(\boldsymbol{\Phi\theta}-\boldsymbol{y}\right)$$

となり、$\boldsymbol{\theta}$ について微分し 0 と置くと、

$$\begin{aligned}\frac{\partial E}{\partial\boldsymbol{\theta}}&=\frac{1}{2}\boldsymbol{\Phi}^T\left(2\tilde{\boldsymbol{W}}\right)\left(\boldsymbol{\Phi\theta}-\boldsymbol{y}\right)\\&=0\end{aligned}$$

$\boldsymbol{\theta}$ を含む項を左辺に移行し、整理すると

$$\boldsymbol{\Phi^T}\tilde{\boldsymbol{W}}\boldsymbol{\Phi\theta}=\boldsymbol{\Phi^T}\tilde{\boldsymbol{W}}\boldsymbol{y}$$

したがって、

$$\hat{\boldsymbol{\theta}}=\left(\boldsymbol{\Phi^T}\tilde{\boldsymbol{W}}\boldsymbol{\Phi}\right)^{\boldsymbol{-1}}\boldsymbol{\Phi^T}\tilde{\boldsymbol{W}}\boldsymbol{y}$$

## 2　宿題 2

目的：微分可能で対称な損失 $\tilde{\rho}(r)$ に対して $\tilde{r}$ で接する二次上界は（存在するなら）次式で与えられることを示す。

$$\tilde{\rho}(r) = \frac{\tilde{w}}{2}r^2 + const$$

証明:

対称な損失関数 $\tilde{\rho}(r)$ は次のように表すことができる

$$\tilde{\rho}(r) = ar^2 + const \tag{1}$$

ここで、$\tilde{r}$ で $\tilde{\rho}(r)$ に接する接線を考えると、その式は次のように書ける。

$$(\tilde{\rho}(\tilde{r}))^{'} = 2a\tilde{r}$$

これを a について整理すると、

$$a = \frac{(\tilde{\rho}(\tilde{r}))^{'}}{2\tilde{r}} \tag{2}$$

(2) を (1) に代入すると下記のように表せる。

$$\tilde{\rho}(r) = \frac{(\tilde{\rho}(\tilde{r}))^{'}}{2\tilde{r}}r^2 + const$$

ここで $\tilde{w} = \frac{(\tilde{\rho}(\tilde{r}))^{'}}{\tilde{r}}$ とおくと、

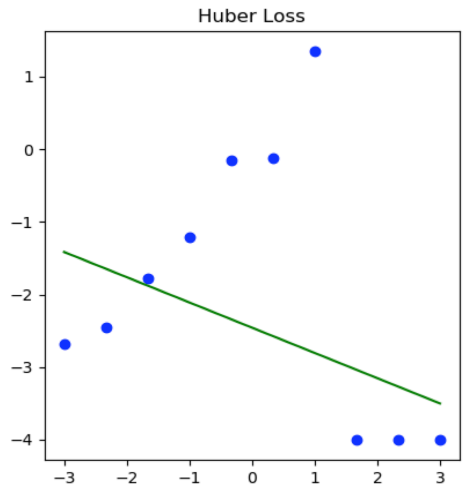
$$\tilde{\rho}(r) = \frac{\tilde{w}}{2}r^2 + const$$

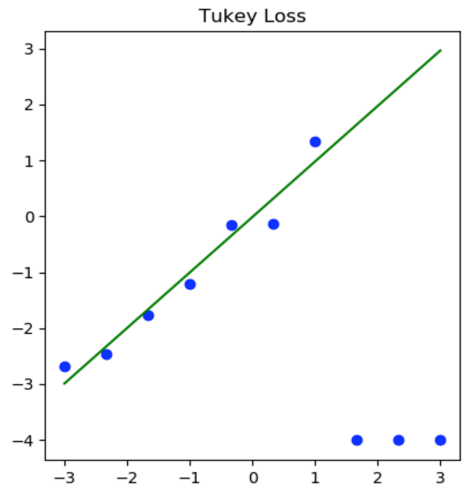
## 3　宿題 3

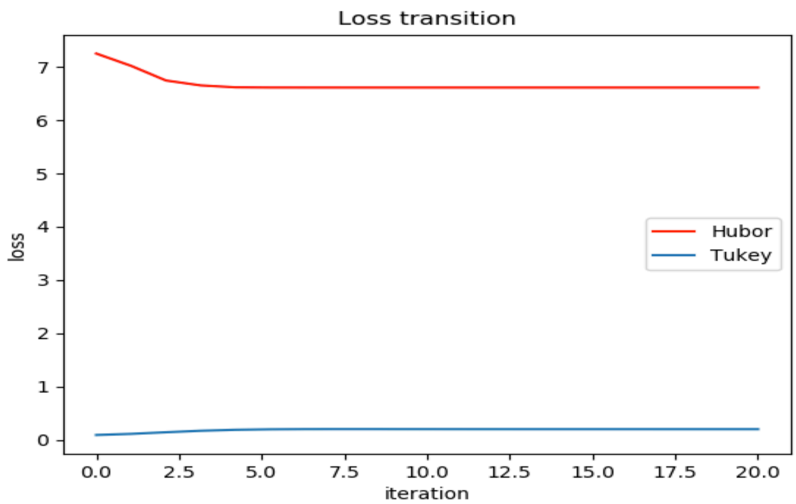
目的：直線モデル $f_\theta(x) = \theta_1 + \theta_2 x$ に対して，テューキー回帰の繰り返し最小二乗アルゴリズムを実装

方法：テューキー回帰を実装。データに対する当てはまり度合いを観察する。

結果：フーバー回帰に対して大幅に当てはまり度合いが良くなった。




実装コード：

```
import numpy as np
import matplotlib

matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

np.random.seed(1)
x_min=-3.
```

```
x_max=3.
n_iter=20

def generate_sample(x_min=-3., x_max=3., sample_size=10):
    x = np.linspace(x_min, x_max, num=sample_size)
    y = x + np.random.normal(loc=0., scale=.2, size=sample_size)
    y[-1] = y[-2] = y[-3] = -4  # outliers
    return x, y


def build_design_matrix(x):
    phi = np.empty(x.shape + (2,))
    phi[:, 0] = 1.
    phi[:, 1] = x
    return phi


def iterative_reweighted_least_squares(x, y, eta=1., n_iter=20):
    phi = build_design_matrix(x)
    # initialize theta using the solution of regularized ridge regression
    theta = theta_prev = np.linalg.solve(
        phi.T.dot(phi) + 1e-4 * np.identity(phi.shape[1]), phi.T.dot(y))
    fig1=plt.figure(figsize=(12,5))
    fig1.subplots_adjust(hspace=0.6, wspace=0.6)

    error_transition_hu=[]
    error_transition_tu=[]

#Hubor
    for _ in range(n_iter):
        r = np.abs(phi.dot(theta_prev) - y)
        w = np.diag(np.where(r > eta, eta / r, 1.))
        phit_w_phi = phi.T.dot(w).dot(phi)
        phit_w_y = phi.T.dot(w).dot(y)
        theta = np.linalg.solve(phit_w_phi, phit_w_y)
        theta_prev = theta
        error_transition_hu.append(np.sum((w/2).dot(r**2)))
```

```
ax1=fig1.add_subplot(1,2,1)
ax1.scatter(x, y, c='blue', marker='o')
ax1.set_title('Huber Loss')
X = np.linspace(x_min, x_max)
Y = predict(X, theta)
ax1.plot(X, Y, color='green')

theta = theta_prev = np.linalg.solve(
    phi.T.dot(phi) + 1e-4 * np.identity(phi.shape[1]), phi.T.dot(y))

#T
for _ in range(n_iter):
    r = np.abs(phi.dot(theta_prev) - y)
    w = np.diag(np.where(r > eta, 0, (1-r**2/eta**2)**2))
    phit_w_phi = phi.T.dot(w).dot(phi)
    phit_w_y = phi.T.dot(w).dot(y)
    theta = np.linalg.solve(phit_w_phi, phit_w_y)
    theta_prev = theta
    error_transition_tu.append(np.sum((w/2).dot(r**2)))
ax2=fig1.add_subplot(1,2,2)
ax2.scatter(x, y, c='blue', marker='o')
ax2.set_title('Tukey Loss')
X = np.linspace(x_min, x_max)
Y = predict(X, theta)
ax2.plot(X, Y, color='green')
plt.show()

plt.figure()
x = np.linspace(0,n_iter,20)
plt.plot(x,error_transition_hu,color="red",label="Hubor")
plt.plot(x,error_transition_tu,label="Tukey")
plt.xlabel('iteration')
plt.ylabel('loss')
plt.title('Loss transition')
plt.legend(loc="best")
plt.show()
```

```python
    return theta

def predict(x, theta):
    phi = build_design_matrix(x)
    return phi.dot(theta)


x, y = generate_sample()
theta = iterative_reweighted_least_squares(x, y, eta=1.)
```