先端データ解析論 (杉山将先生・本多淳也先生)
第 3 回レポート

ashiato45

2017 年 5 月 1 日

宿題 1

$$T_+(z) = z^2/2 + (\lambda - u - \theta)z + (u\theta + \theta^2/2), \tag{1}$$
$$T_-(z) = z^2/2 + (-\lambda - u - \theta)z + (u\theta + \theta^2/2) \tag{2}$$

と定義する。両方とも最高次が正な 2 次関数であり、$T'_+(z) = z + (\lambda - u - \theta), T'_-(z) = z + (-\lambda - u - \theta)$ なので、

$$\operatorname*{argmin}_{z \geq 0} T(z) = \max\left(\operatorname*{argmin}_z T_+(z), 0\right) = \max(\theta + u - \lambda, 0), \tag{3}$$
$$\operatorname*{argmin}_{z \leq 0} T(z) = \min\left(\operatorname*{argmin}_z T_-(z), 0\right) = \min(\lambda + u + \theta, 0). \tag{4}$$

よって、

$$\operatorname*{argmin}_z T(z) = \operatorname*{argmin}_{z \geq 0} T(z) + \operatorname*{argmax}_{z \leq 0} T(z) = \max(\theta + u - \lambda, 0) + \min(\lambda + u + \theta, 0). \tag{5}$$

宿題 2

$$\Phi = \begin{pmatrix} K(x_1, x_1) & \dots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_1) & \dots & K(x_n, x_n) \end{pmatrix} \tag{6}$$

として、以下の反復

$$\theta \leftarrow (\Phi^\top \Phi + I)^{-1}(\Phi^\top y + z - u), \tag{7}$$
$$z \leftarrow \max(0, \theta + u - \lambda) - \max(0, -\theta - u - \lambda), \tag{8}$$
$$u \leftarrow u + \theta - z \tag{9}$$

を行えばよい。今回は停止条件として「更新の幅が十分小さくなったら」というものを用いた。計算に用いたプログラムは付録に記す。パラメタは、$h = 0.3, \lambda = 0.2$ とした。

結果、図 1 を得た。50 個のパラメタのうち、38 個は 10^{-8} 以下であり、ほぼ 0 と見做せた。誤差のノルムは 1.14 であった。

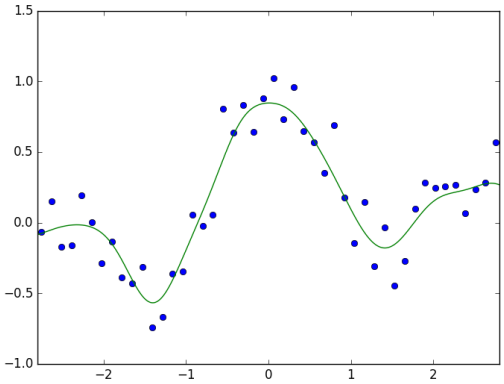

図 1

付録

```
import sys
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)

h = float(sys.argv[1])
lam = float(sys.argv[2])

datanum = 50
datasamples = np.linspace(-3, 3, datanum)
datasamples = datasamples.reshape(len(datasamples), 1)
y = np.sin(np.pi*datasamples)/(np.pi*datasamples) + 0.1*datasamples
y += np.random.randn(datanum, 1)*0.2 # normal distribution
# print(y)

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

phi = np.fromfunction(lambda i, j: gk(datasamples[i, 0], datasamples[j, 0]), (datanum, datanum), dtype=int)

theta = np.zeros((datanum, 1))
z = np.zeros((datanum, 1))
u = np.zeros((datanum, 1))

while True:
    theta2 = np.linalg.solve(np.dot(phi.transpose(), phi) + np.eye(datanum, dtype=int), np.dot(phi.transpose(), y) + z - u)
    z2 = np.maximum(0, theta2+u-lam) + np.minimum(0, theta2+u+lam)
    u2 = u + theta2 - z2
    dt = theta-theta2
    dz = z-z2
    du = u-u2
    theta = theta2
    z = z2
    u = u2
    if np.all(abs(dt) < 1e-9) and np.all(abs(dz) < 1e-9) and np.all(abs(du) < 1e-9):
        break


graphnum = 5000
graphsamples = np.linspace(-3, 3, graphnum)
graphsamples = graphsamples.reshape(graphnum, 1)
calckernel = lambda x: np.dot(theta.transpose(), np.fromfunction(lambda i, j:gk(x, datasamples[i, 0]), (datanum, 1), dtype=int))[0, 0]
km = np.fromfunction(lambda i,j: gk(graphsamples[i, 0], datasamples[j, 0]), (graphnum, datanum), dtype=int)
graph = np.dot(km, theta)

em = np.fromfunction(lambda i,j: gk(datasamples[i, 0], datasamples[j, 0]), (datanum, datanum), dtype=int)
eout = np.dot(em, theta)
er = eout - y

print(theta)
print(np.sum(abs(theta) < 1e-8))
print(np.linalg.norm(er))

plt.axis([-2.8, 2.8, -1, 1.5])
plt.plot(datasamples, y, 'o')
plt.plot(graphsamples, graph, '-')
# plt.plot(datasamples, eout, 'o')
plt.show()
```