先端データ解析論 (杉山将先生・本多淳也先生)
第 5 回レポート

ashiato45

2017 年 5 月 16 日

宿題 1

$$\widehat{\Sigma}^{-1}(\widehat{\mu}_+ - \widehat{\mu}_-) = n(X^\top X)^{-1}\left(\frac{1}{n_+}\sum_{i:\, y_i>0} x_i - \frac{1}{n_-}\sum_{y_i<0} x_i\right) \tag{1}$$

$$= n(X^\top X)^{-1}\left(X^\top(\chi_{y_i>0}/n_+)_i + X^\top(-\chi_{y_i<0}/n_-)\right) \tag{2}$$

$$= (X^\top X)^{-1}X^\top y. \tag{3}$$

ここで、χ_{P_i} は命題 P_i がなりたつときだけ 1 となり、他は 0 となる指示関数であり、$(\chi_{y_i>0}/n_+)_i$ は i でインデックスされた縦ベクトルである。$(-\chi_{y_i<0}/n_-)$ も同様。

宿題 2

Python 実装は付録にある。$h = 1, \lambda = 1$ とした。手法として、一対他法を用いた。結果、認識率は 2000/2000 となった。(大変不審だが、バグというわけでもないし見直してもテストデータを学習したりはしていないのでこのまま提出する。)

分類の結果のベクトルの上 10 行を添付する:

```
[[ 0.04262447 -0.05240391 -0.05240407 -0.0523833  -0.05240401 -0.05240226
  -0.05240403 -0.04264692 -0.05240407 -0.05240401]
 [ 0.36047768 -0.36102044 -0.36102349 -0.36054768 -0.36102312 -0.36102844
  -0.36102336 -0.3609486  -0.36102439 -0.36102312]
 [ 0.99966143 -1.00045727 -1.00040688 -0.99985568 -1.00040916 -1.00026625
  -1.00040237 -1.00035691 -1.00037088 -1.00040914]
 [ 0.54018609 -0.53807053 -0.53807618 -0.5400626  -0.53807573 -0.53808072
  -0.53807661 -0.53819459 -0.5380792  -0.53807573]
 [ 0.21946985 -0.22708963 -0.22701911 -0.22844702 -0.22701907 -0.21808369
  -0.22702193 -0.22691148 -0.22702368 -0.22702082]
 [ 0.94659635 -0.95316953 -0.95332412 -0.94480574 -0.95273149 -0.95277348
  -0.95313887 -0.9524805  -0.95329306 -0.95273149]
 [ 0.15014248 -0.14958288 -0.14958233 -0.15008022 -0.14958222 -0.1495817
  -0.14958236 -0.14966585 -0.14956047 -0.14958222]
 [ 0.93879607 -1.0222282  -1.02226578 -0.93867979 -1.0222337  -1.0223108
  -1.02223    -1.02225364 -1.02223008 -1.0222337 ]
 [ 0.86976546 -0.84741716 -0.84746422 -0.86981668 -0.8474761  -0.84739499
```

```
  -0.84749496 -0.84751297 -0.84752108 -0.8474761 ]
 [ 0.98670759 -1.01883236 -1.01868275 -0.98655863 -1.01879963 -1.01908324
  -1.01876152 -1.0188339  -1.01875307 -1.01879964]]
```

これは、行方向に分類したデータが並んでおり、行方向にそれが 0 らしいか、1 らしいか…が並んでいる。したがって、行方向に argmax を取れば分類ができることになる。

付録

プログラム中の_K1 と_KK1 は計算に時間がかかるのでファイルに保存してある。生成に使ったコードはコメントアウトしてある。

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

np.random.seed(42)

datanum = 10
datasamples = np.linspace(-3, 3, datanum)
datasamples = datasamples.reshape(len(datasamples), 1)
y = datasamples + 0.2*np.random.randn(datanum, 1)
y[datanum - 1] = -4
y[datanum - 2] = -4
y[3] = -4

eta = 1
theta = np.zeros((2, 1))
phi = np.ones((datanum, 2))
phi[:, 1] = datasamples.reshape(len(datasamples))
for i in range(10000):
    r = np.abs(np.dot(phi, theta) - y)
    w = np.zeros((datanum, 1))
    w[r <= eta] = ((1-r**2/eta**2)**2)[r <= eta]
    W = np.diag(w.reshape(len(w)))
    A = np.dot(np.dot(phi.transpose(), W), phi)
    b = np.dot(np.dot(phi.transpose(), W), y)
    theta2 = np.linalg.solve(A, b)
    if np.linalg.norm(theta2 - theta) < 0.001:
        break
    theta = theta2

graphnum = 5000
graphsamples = np.linspace(-3, 3, graphnum)
graphsamples = graphsamples.reshape(graphnum, 1)
graph = theta[0, 0] + theta[1, 0]*graphsamples

plt.axis([-3.2, 3.2, -4.2, 4.2])
plt.plot(datasamples, y, 'o')
plt.plot(graphsamples, graph, '-')
plt.show()
```