

先端データ解析論
第五回 レポート

37-196360 森田涼介

2019 年 5 月 14 日

宿題 1

線形モデル

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} \quad (1)$$

を用いた最小二乗分類を考える。いま、訓練標本の平均 $\boldsymbol{\mu}$ について次式が成立する。

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i = \mathbf{0} \quad (2)$$

また、最小二乗誤差は、

$$L(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^n (f_{\boldsymbol{\theta}}(\mathbf{x}) - y_i)^2 \quad (3)$$

$$= \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 \quad (4)$$

と表されるので、これの $\boldsymbol{\theta}$ による偏微分が $\mathbf{0}$ になる条件から、

$$(\mathbf{X}^T \mathbf{X}) \hat{\boldsymbol{\theta}} = \mathbf{X}^T \mathbf{y} \quad (5)$$

が成立する。ここで、

$$\mathbf{X} = [\mathbf{x}_1 \quad \cdots \quad \mathbf{x}_n]^T \quad (6)$$

$$\mathbf{y} = [y_1 \quad \cdots \quad y_n]^T \quad (7)$$

である。

いま、正例を下付きの $+$ で、負例を下付きの $-$ で表すこととすると、次が成立する。

$$n = n_+ + n_- \quad (8)$$

$$\boldsymbol{\mu}_+ = \frac{1}{n_+} \sum_{\mathbf{x}_i \in D_+} x_i \quad (9)$$

$$\boldsymbol{\mu}_- = \frac{1}{n_-} \sum_{\mathbf{x}_i \in D_-} x_i \quad (10)$$

$$(11)$$

全標本の和を考えると、

$$\sum_{i=1}^n x_i = \sum_{\mathbf{x}_i \in D_+} x_i + \sum_{\mathbf{x}_i \in D_-} x_i \quad (12)$$

$$n\boldsymbol{\mu} = n_+\boldsymbol{\mu}_+ + n_-\boldsymbol{\mu}_- = \mathbf{0} \quad (13)$$

$$\boldsymbol{\mu}_- = -\frac{n_+}{n_-} \boldsymbol{\mu}_+ \quad (14)$$

が得られる。

また、二値分類問題を考えているので、 y は正例のとき 1 、負例のとき -1 となる。このことから、

$$\mathbf{X}^T \mathbf{y} = \sum_{\mathbf{x}_i \in D_+} x_i - \sum_{\mathbf{x}_i \in D_-} x_i \quad (15)$$

$$= n_+\boldsymbol{\mu}_+ - n_-\boldsymbol{\mu}_- \quad (16)$$

$$= 2n_+\boldsymbol{\mu}_+ \quad (17)$$

を得る。

また、平均 $\boldsymbol{\mu} = \mathbf{0}$ より、標本の共分散行列の推定値は次で表される。

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \quad (18)$$

これを用いると、次が成立する。

$$\mathbf{X}^T \mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \quad (19)$$

$$= \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \quad (20)$$

$$= n \hat{\boldsymbol{\Sigma}} \quad (21)$$

以上、式 (5), (17), (21) より、

$$n \hat{\boldsymbol{\Sigma}} \hat{\boldsymbol{\theta}} = 2n_+ \boldsymbol{\mu}_+ \quad (22)$$

$$\hat{\boldsymbol{\theta}} = \frac{2n_+}{n} \hat{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\mu}_+ \quad (23)$$

を得る。一方、フィッシャー判別分析による境界の垂線方向は、

$$\hat{\boldsymbol{\Sigma}}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-) = \hat{\boldsymbol{\Sigma}} \left(1 + \frac{n_+}{n_-} \right) \boldsymbol{\mu}_+ \quad (24)$$

$$= \frac{n}{n_-} \hat{\boldsymbol{\Sigma}} \boldsymbol{\mu}_+ \quad (25)$$

となる。式 (23), (25) から、最小二乗分類により得られる識別境界の垂線 $\hat{\boldsymbol{\theta}}$ は、フィッシャー判別分析による境界の垂線 $\hat{\boldsymbol{\Sigma}}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)$ と同じ方向になる。

宿題 2

MNIST の手書き数字分類を，ガウスカーネルモデルに対する最小二乗回帰（一対他法）を用いて行う。訓練標本には各 500 文字ずつ計 5,000 文字，テスト標本には各 200 文字ずつ計 2,000 文字が含まれている。

プログラムは 4 ページの Listing 1 に示した。以下に結果を示す。混同行列は表 1 のようになり，各カテゴリごとの正解率等は表 2 のようになった。

ここで，ガウスカーネルのバンド幅 h と L2 正則化項の係数 λ は次のようにした。

$$h = 1.0 \quad (26)$$

$$\lambda = 1 \times 10^{-4} \quad (27)$$

表 1: 混同行列

	0	1	2	3	4	5	6	7	8	9
0	198	0	1	1	0	0	0	0	0	0
1	0	200	0	0	0	0	0	0	0	0
2	0	0	193	1	0	0	0	2	3	1
3	0	0	0	192	0	3	0	2	2	1
4	0	5	0	0	184	1	3	0	0	7
5	2	0	3	4	0	187	0	1	1	2
6	1	0	2	0	0	2	195	0	0	0
7	0	1	0	0	3	0	0	191	1	4
8	3	2	1	4	0	3	0	0	185	2
9	0	1	0	0	2	0	0	2	0	195

表 2: テストデータに対する各カテゴリごとの結果

Category	# Data	# Correct	Accuracy
0	200	198	0.990
1	200	200	1.000
2	200	193	0.965
3	200	192	0.960
4	200	184	0.920
5	200	187	0.935
6	200	195	0.975
7	200	191	0.955
8	200	185	0.925
9	200	195	0.975
All	2,000	1,920	0.960

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 3 に示す。

表 3: 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

```
1  # -*- coding: utf-8 -*-
2
3
4  import pathlib
5  import numpy as np
6  import matplotlib.pyplot as plt
7
8
9  def load_data(n_label=None, n_train=None, n_test=None):
10     data_dir = '../data/'
11     data_dir = pathlib.Path(data_dir)
12     categories = list(range(10))
13     train_X = []
14     test_X = []
15     for category in categories[:n_label]:
16         # train data
17         data_path = data_dir / 'digit_train{}.csv'.format(category)
18         data = np.loadtxt(str(data_path), delimiter=',')[:n_train]
19         train_X.append(data)
20
21         # test data
22         data_path = data_dir / 'digit_test{}.csv'.format(category)
23         data = np.loadtxt(str(data_path), delimiter=',')[:n_test]
24         test_X.append(data)
25     labels = categories[:n_label]
26     return train_X, test_X, labels
27
28
29 def make_train_data(train_data, labels, label):
30     train_X = []
31     train_y = []
32     for i in labels:
```

```

33     data = train_data[i]
34     train_X.extend(data)
35     n_data = len(data)
36     if i == label:
37         train_y.extend([1] * n_data)
38     else:
39         train_y.extend([-1] * n_data)
40     train_X = np.array(train_X)
41     train_y = np.array(train_y)
42     return train_X, train_y
43
44
45 class GaussKernelModel(object):
46     def __init__(self, n, bandwidth):
47         self.n = n
48         self.bandwidth = bandwidth
49         self.theta = np.empty(n)
50
51     def __call__(self, x, c):
52         K = self.design_matrix(x, c)
53         y_hat = K.T.dot(self.theta)
54         return y_hat
55
56     def kernel(self, x, c, save_memory=False):
57         if save_memory:
58             n_x = x.shape[1]
59             n_c = c.shape[0]
60             d = x.shape[-1]
61             norm = np.zeros((n_c, n_x))
62             x = np.reshape(x, (n_x, d))
63             c = np.reshape(c, (n_c, d))
64             for i in range(len(x)):
65                 x_i = x[i]
66                 norm[i, :] = np.sum((x_i - c)**2, axis=-1)
67         else:
68             norm = np.sum((x - c) ** 2, axis=-1)
69             ker = np.exp(- norm / (2*self.bandwidth**2))
70             return ker
71
72     def design_matrix(self, x, c, save_memory=False):
73         mat = self.kernel(x[None], c[:, None], save_memory)
74         return mat
75
76     def train(self, x, y, lamb=1e-4, save_memory=False):
77         K = self.design_matrix(x, x, save_memory)
78         self.theta = np.linalg.solve(
79             K**2 + lamb*np.identity(len(K)),

```

```

80         K.T.dot(y[:, None]),
81     )
82     return
83
84
85 def train_one_vs_others(train_data, labels, h, lamb=1e-4, save_memory=False):
86     model = []
87     print('Train: ', end='')
88     for label in labels:
89         print(f'{label} ', end='')
90         train_X, train_y = make_train_data(train_data, labels, label)
91         n_data = len(train_y)
92         model_i = GaussKernelModel(n_data, h)
93         model_i.train(train_X, train_y, lamb, save_memory)
94         model.append(model_i)
95     print('\ndone\n')
96     return model
97
98
99 def test(model, train_data, test_data, labels):
100     n_label = len(labels)
101     confusion_matrix = np.zeros((n_label, n_label), dtype=int)
102     n_data_all = 0
103     result = {}
104     print('Test')
105     for label in labels:
106         print(f'Label: {label}\t', end='')
107
108         # load
109         test_X = test_data[label]
110         n_data = len(test_X)
111         n_data_all += n_data
112         train_X = []
113         for i in labels:
114             data = train_data[i]
115             train_X.extend(data)
116         train_X = np.array(train_X)
117
118         # predict
119         preds = []
120         for i in labels:
121             pred = model[i](test_X, train_X).flatten() # (n_data,)
122             preds.append(pred)
123         preds = np.array(preds).T # (n_data, n_label)
124         preds = np.argmax(preds, axis=1) # (n_data,)
125
126         # make confusion matrix

```

```

127         for i in labels:
128             n = (preds == i).sum()
129             confusion_matrix[label, i] = n
130
131         # calc accuracy
132         n_correct = confusion_matrix[label, label]
133         acc = n_correct / n_data
134         print(f'#Data: {n_data}\t#Correct: {n_correct}\tAcc: {acc:.3f}')
135
136         result[label] = {
137             'data': n_data,
138             'correct': n_correct,
139             'accuracy': acc,
140         }
141     result['confusion_matrix'] = confusion_matrix
142
143     # overall score
144     n_crr_all = np.diag(confusion_matrix).sum()
145     acc_all = n_crr_all / n_data_all
146     result['all'] = {
147         'data': n_data_all,
148         'correct': n_crr_all,
149         'accuracy': acc_all,
150     }
151     print(f'All\t#Data: {n_data_all}\t#Correct: {n_crr_all}\tAcc: {acc_all:.3f}')
152     print()
153     print('Confusion Matrix:\n', confusion_matrix)
154     print()
155     return result
156
157
158 def print_result_in_TeX_tabular_format(result):
159     labels = list(range(10))
160     print('Scores')
161     for label in labels:
162         print('{} & {} & {} & {:.3f} \\\\'
163             .format(
164                 label,
165                 int(result[label]['data']),
166                 int(result[label]['correct']),
167                 result[label]['accuracy']
168             ))
169     print()
170     print('Confusion Matrix')
171     for i in labels:
172         print('{} \t'.format(i), end='')
173     for j in labels:

```



```

173         print(' & {}'.format(int(result['confusion_matrix'][i, j])),
end='')
174     print(' \\\\'')
175     return
176
177
178 def main():
179     # settings
180     bandwidth = 1.0
181     lamb = 1e-4
182     np.random.seed(0)
183
184     print('Settings')
185     print(f'bandwidth: {bandwidth}\tlambda (L2): {lamb}\n')
186
187     # load data
188     train_X, test_X, labels = load_data(n_label=10, n_train=None,
n_test=None)
189
190     # train
191     model = train_one_vs_others(train_X, labels, bandwidth, lamb,
save_memory=True)
192
193     # test
194     result = test(model, train_X, test_X, labels)
195     print_result_in_TeX_tabular_format(result)
196
197 if __name__ == '__main__':
198     main()

```