CS 760: Machine Learning - Fall 2020
Homework 3: Logistic Regression

Due : 10/22/2020

Zijie Zhang

October 19, 2020

Problem 1

Proof.

(a) Step size: $\eta_t = - \left(\nabla^2 \ell(\theta_t)\right)^{-1}$

(b) Stop at the 7th iteration.

(c) $\hat{\theta} = [2.53996931, -1.17765865, 2.75728236, -0.04347367, -0.40183102, -0.10650523, 0.00278568]$

(d) $\ell(\hat{\theta}) = -390.465964754707$

(e) By **Theorem 6.2**, as $N \to \infty$,

$$\hat{\theta} \xrightarrow{d} \mathcal{N}\left(\theta^*, \frac{1}{N} \mathbf{I}_{\theta^*}^{-1}\right)$$

where $\mathbf{I}_{\theta^*} := -\nabla^2 \ell(\theta^*)$

$$\hat{\theta} \xrightarrow{d} \mathcal{N}\left(\theta^*, \frac{1}{N} \left(-\nabla^2 \ell(\theta^*)\right)^{-1}\right)$$

$\square$

Problem 2

Proof.

(a)
$$\hat{\omega} = \hat{\theta}^T \mathbf{x}$$

(b)
$$\hat{\omega} \xrightarrow{d} \mathcal{N}\left(\hat{\theta^*}^T \mathbf{x}, \frac{1}{N} \mathbf{x}^{\mathbf{T}} \mathbf{I}_{\theta^*}^{-1} \mathbf{x}\right)$$

$\square$

Problem 3

Proof.

(a)
$$\mathbf{x} = [1, 3, 0, 22, 0, 3, 7.25]$$
$$\hat{\omega} = -2.2487468892027604$$

I wouldn't, $\hat{\omega} < 0$, $\hat{y} = 0.0954596285185124 < \frac{1}{2}$.

Because I am single and have family on board. On the other hand, women and children first, my rescue priority is relatively low.

The lower passenger class are less likely to survive, and the first class have a greater chance of surviving.

(b)
$$\tau = 0.024518795784839398$$

(c) The 95% confidence interval is

$$(-2.2732656849876, -2.224228093417921)$$

The entire interval is to the left of 0. So, the answer is fairly certain.

$\square$

Problem 4

Proof.

(a) The jth feature is significant if
$$\hat{\theta}_j^2 > \nu_j^2 \Phi^{-1}(\alpha)$$

(b)
```
0  25.44164930795299
1  64.99221714290489
2  189.2765279839097
3  31.68956969534484
4  13.173407762230326
5  0.8066025744790511
6  1.3591762929269946
alpha = 0.05 , 3.841458820694124
```

Passenger Class,**Gender**, **Age**, **Siblings/Spouses** are significant features.

(c) If I change to female, $\hat{y} = 0.6244586486812256 > \frac{1}{2}$. I will survive.

$\square$

code

```python
1  import pandas as pd
2  import numpy as np
3  import math
4  import scipy.stats as st
5
6  #import data
7  data = pd.read_csv('titanic_data.csv')
8  y_label = data['Survived'].to_numpy()
9  X = data.drop(['Survived'], axis=1).to_numpy()
10 N, D = X.shape
11 X = np.append(np.array([[1] for i in range(N)]), X, axis=1)
12 N, D = X.shape
13
14 def dot(theta, X, i):
15     N, D = X.shape
16     ans = 0
17     for j in range(D):
18         ans = ans + theta[j]*X[i,j]
19     return ans
20
21 def l(theta, X):
22     ans = 0
23     N, D = X.shape
24     for i in range(N):
25         ans = ans        -y_label[i]*math.log(1+math.exp(-dot(theta, X, i)))        -(1-
       y_label[i])*math.log(1+math.exp(dot(theta, X, i)))
26     return ans
27
28 def nabla_l(theta, X):
29     N, D = X.shape
30     ans = np.zeros(D)
31     for i in range(N):
32         ans = ans +        (y_label[i] - 1/(1+math.exp(-dot(theta, X, i)))) * X[i,]
33     return ans
34
35 def hessian(theta, X):
36     N, D = X.shape
37     ans = np.zeros((D,D))
38     for i in range(N):
39         ans = ans\
40         - np.dot(np.asmatrix(X[i,]).T, np.asmatrix(X[i,]))\
41         * math.exp(-dot(theta, X, i))/ (1+math.exp(-dot(theta, X, i)))**2
42     return ans
43
44 def GradientAscent(theta, X, epsilon):
45     k = 1
46     nabla = nabla_l(theta, X)
47     nabla_norm = np.linalg.norm(nabla)
48     print("k = ", k, "\n\ttheta = ", theta)
49     print("\tnabla_norm = ", nabla_norm, "\n")
50     while nabla_norm > epsilon:
51         k=k+1
52         Hessian = hessian(theta, X)
53         alpha = np.dot(np.linalg.inv(Hessian), nabla)
54         theta = theta - np.array(alpha)[0]
55         nabla = nabla_l(theta, X)
56         nabla_norm = np.linalg.norm(nabla)
57         print("k = ", k, "\n\ttheta = ", theta)
58         print("\tnabla_norm = ", nabla_norm, "\n")
59     return theta
60
61 # init parameter
62 N, D = X.shape
63 theta = np.array([0 for i in range(D)])
64 epsilon = 1e-12
65
66 # fit
67 theta = GradientAscent(theta, X, epsilon)
```

```python
68  print("l(theta) = ", l(theta, X))
69
70  # asymptotic
71  def fisher(theta, X):
72      return -hessian(theta, X)
73  Mean = theta
74  Sigma = np.linalg.inv(fisher(theta, X))/N
75  print("mean = ", Mean)
76  print("sigma = ", Sigma)
77
78  # Example
79  x = np.array([1,3,0,22,0,3,7.25])
80  omega = np.dot(np.asmatrix(theta), np.asmatrix(x).T).tolist()[0][0]
81  print('omega = ', omega)
82
83  # derive tau
84  alpha = 0.05
85  std_omega = math.sqrt(np.dot(np.dot(np.asmatrix(x), np.linalg.inv(fisher(theta, X))),np.
        asmatrix(x).T).tolist()[0][0]/N)
86  tau = st.norm.interval(1-alpha, loc = omega, scale = std_omega)
87  print('tau = ', tau[1] - omega)
88  print(tau)
89
90  # Learning Significant Features
91  df = 1
92  st.chi2.ppf(1-alpha, df)
93  Sigma = np.linalg.inv(fisher(theta, X))
94  for j in range(D):
95      v = (Sigma[j,j])
96      print(j,theta[j]*theta[j]/v)
97  print('alpha =', alpha, ',', st.chi2.ppf(1-alpha, df))
98
99  # Example
100 x = np.array([1,3,1,22,0,3,7.25])
101 omega = np.dot(np.asmatrix(theta), np.asmatrix(x).T).tolist()[0][0]
102 y_ = 1/(1+math.exp(-omega))
103 print('omega = ', omega)
104 print('y_ = ', y_)
```

code result

```
 1 k =  1
 2   theta =  [0 0 0 0 0 0 0]
 3   nabla_norm =   4043.297274997574
 4
 5 k =  2
 6   theta =  [ 1.29729102e+00 -7.20135362e-01  2.03094224e+00 -2.47886747e-02
 7  -2.01165383e-01 -7.71336514e-02  1.61359036e-03]
 8   nabla_norm =   851.9838354569285
 9
10 k =  3
11   theta =  [ 2.24795287e+00 -1.07328272e+00  2.59294853e+00 -3.90996306e-02
12  -3.45601865e-01 -1.03378863e-01  2.50249418e-03]
13   nabla_norm =   153.6420958940281
14
15 k =  4
16   theta =  [ 2.52222201 -1.17135233  2.74707544 -0.04320928 -0.39757009 -0.10654932
17   0.00276689]
18   nabla_norm =   8.46634411645958
19
20 k =  5
21   theta =  [ 2.5398976  -1.17763361  2.75724004 -0.04347261 -0.40180868 -0.10650616
22   0.00278559]
23   nabla_norm =   0.03091722885180926
24
25 k =  6
26   theta =  [ 2.53996931 -1.17765865  2.75728235 -0.04347367 -0.40183102 -0.10650523
27   0.00278568]
28   nabla_norm =   4.425564116334014e-07
29
30 k =  7
31   theta =  [ 2.53996931 -1.17765865  2.75728236 -0.04347367 -0.40183102 -0.10650523
32   0.00278568]
33   nabla_norm =   9.434469596878176e-13
34
35 l(theta) =   -390.465964754707
36 mean =  [ 2.53996931 -1.17765865  2.75728236 -0.04347367 -0.40183102 -0.10650523
37   0.00278568]
38 sigma =  [[ 2.85882819e-04 -7.25571210e-05  9.56069677e-06 -3.32103305e-06
39   -1.04593654e-05  2.59710472e-06 -5.96893717e-07]
40  [-7.25571210e-05  2.40576874e-05 -7.81199155e-06  5.75795416e-07
41    5.18198617e-07 -1.87877997e-06  1.92163542e-07]
42  [ 9.56069677e-06 -7.81199155e-06  4.52837206e-05 -2.00732961e-07
43   -4.26792360e-06 -5.28298601e-06 -5.21377294e-09]
44  [-3.32103305e-06  5.75795416e-07 -2.00732961e-07  6.72376664e-08
45    2.41735298e-07  3.96443409e-08  7.11842803e-10]
46  [-1.04593654e-05  5.18198617e-07 -4.26792360e-06  2.41735298e-07
47    1.38186358e-05 -3.77743192e-06 -3.89866365e-08]
48  [ 2.59710472e-06 -1.87877997e-06 -5.28298601e-06  3.96443409e-08
49   -3.77743192e-06  1.58547228e-05 -6.77235360e-08]
50  [-5.96893717e-07  1.92163542e-07 -5.21377294e-09  7.11842803e-10
51   -3.89866365e-08 -6.77235360e-08  6.43668174e-09]]
52 omega =   -2.2487468892027604
53 tau =   0.024518795784839398
54 (-2.2732656849876 , -2.224228093417921)
55 0 22566.7429361543
56 1 57648.09660575663
57 2 167888.28032172789
58 3 28108.648319770873
59 4 11684.812685098299
60 5 715.4564835629184
61 6 1205.5893718262441
62 alpha = 0.05 , 3.841458820694124
63 omega =   0.5085354659355167
64 y_ =   0.6244630907922954
```