# 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)

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

$\square$

## 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.

□

## Problem 4

*Proof.*

(a) The $j$th feature is significant if

$$\hat{\theta}_j^2 > \nu_j^2 \Phi^{-1}(\alpha)$$

(b) Every feature is significant if $\alpha = 0.05$.

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

□

## code

```
import pandas as pd
import numpy as np
import math
import scipy.stats as st

#import data
data = pd.read_csv('titanic_data.csv')
y_label = data['Survived'].to_numpy()
X = data.drop(['Survived'], axis=1).to_numpy()
N, D = X.shape
X = np.append(np.array([[1] for i in range(N)]), X, axis=1)
N, D = X.shape

def dot(theta, X, i):
    N, D = X.shape
    ans = 0
    for j in range(D):
        ans = ans + theta[j]*X[i,j]
    return ans

def l(theta, X):
    ans = 0
    N, D = X.shape
    for i in range(N):
        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)))
    return ans

def nabla_l(theta, X):
    N, D = X.shape
    ans = np.zeros(D)
    for i in range(N):
        ans = ans +         (y_label[i] - 1/(1+math.exp(-dot(theta, X, i)))) * X[i,]
    return ans

def hessian(theta, X):
    N, D = X.shape
    ans = np.zeros((D,D))
    for i in range(N):
        ans = ans\
        - np.dot(np.asmatrix(X[i,]).T, np.asmatrix(X[i,]))\
        * math.exp(-dot(theta, X, i))/ (1+math.exp(-dot(theta, X, i)))**2
    return ans

def GradientAscent(theta, X, epsilon):
    k = 1
    nabla = nabla_l(theta, X)
    nabla_norm = np.linalg.norm(nabla)
    print("k = ", k, "\n\ttheta = ", theta)
    print("\tnabla_norm = ", nabla_norm, "\n")
    while nabla_norm > epsilon:
        k=k+1
        Hessian = hessian(theta, X)
        alpha = np.dot(np.linalg.inv(Hessian), nabla)
        theta = theta - np.array(alpha)[0]
        nabla = nabla_l(theta, X)
        nabla_norm = np.linalg.norm(nabla)
        print("k = ", k, "\n\ttheta = ", theta)
        print("\tnabla_norm = ", nabla_norm, "\n")
    return theta

# init parameter
N, D = X.shape
theta = np.array([0 for i in range(D)])
epsilon = 1e-12

# fit
theta = GradientAscent(theta, X, epsilon)
```

```
print("l(theta) = ", l(theta, X))

# asymptotic
def fisher(theta, X):
    return -hessian(theta, X)
Mean = theta
Sigma = np.linalg.inv(fisher(theta, X))/N
print("mean = ", Mean)
print("sigma = ", Sigma)

# Example
x = np.array([1,3,0,22,0,3,7.25])
omega = np.dot(np.asmatrix(theta), np.asmatrix(x).T).tolist()[0][0]
print('omega = ', omega)

# derive tau
alpha = 0.05
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)
tau = st.norm.interval(1-alpha, loc = omega, scale = std_omega)
print('tau = ', tau[1] - omega)
print(tau)

# Learning Significant Features
df = 1
st.chi2.ppf(1-alpha, df)
Sigma = np.linalg.inv(fisher(theta, X))/N
for j in range(D):
    v = (Sigma[j,j])
    print(j,theta[j]*theta[j]/v)
print('alpha =', alpha, ',', st.chi2.ppf(1-alpha, df))

# Example
x = np.array([1,3,1,22,0,3,7.25])
omega = np.dot(np.asmatrix(theta), np.asmatrix(x).T).tolist()[0][0]
y_ = 1/(1+math.exp(-omega))
print('omega = ', omega)
print('y_ = ', y_)
```

## code result

```
k =  1
  theta =  [0 0 0 0 0 0 0]
  nabla_norm =  4043.297274997574

k =  2
  theta =  [ 1.29729102e+00 -7.20135362e-01  2.03094224e+00 -2.47886747e-02
 -2.01165383e-01 -7.71336514e-02  1.61359036e-03]
  nabla_norm =  851.9838354569285

k =  3
  theta =  [ 2.24795287e+00 -1.07328272e+00  2.59294853e+00 -3.90996306e-02
 -3.45601865e-01 -1.03378863e-01  2.50249418e-03]
  nabla_norm =  153.6420958940281

k =  4
  theta =  [ 2.52222201 -1.17135233  2.74707544 -0.04320928 -0.39757009 -0.10654932
  0.00276689]
  nabla_norm =  8.46634411645958

k =  5
  theta =  [ 2.5398976  -1.17763361  2.75724004 -0.04347261 -0.40180868 -0.10650616
  0.00278559]
  nabla_norm =  0.03091722885180926

k =  6
  theta =  [ 2.53996931 -1.17765865  2.75728235 -0.04347367 -0.40183102 -0.10650523
  0.00278568]
  nabla_norm =  4.425564116334014e-07

k =  7
  theta =  [ 2.53996931 -1.17765865  2.75728236 -0.04347367 -0.40183102 -0.10650523
  0.00278568]
  nabla_norm =  9.434469596878176e-13

l(theta) =  -390.465964754707
mean =  [ 2.53996931 -1.17765865  2.75728236 -0.04347367 -0.40183102 -0.10650523
  0.00278568]
sigma =  [[ 2.85882819e-04 -7.25571210e-05  9.56069677e-06 -3.32103305e-06
  -1.04593654e-05  2.59710472e-06 -5.96893717e-07]
 [-7.25571210e-05  2.40576874e-05 -7.81199155e-06  5.75795416e-07
   5.18198617e-07 -1.87877997e-06  1.92163542e-07]
 [ 9.56069677e-06 -7.81199155e-06  4.52837206e-05 -2.00732961e-07
  -4.26792360e-06 -5.28298601e-06 -5.21377294e-09]
 [-3.32103305e-06  5.75795416e-07 -2.00732961e-07  6.72376664e-08
   2.41735298e-07  3.96443409e-08  7.11842803e-10]
 [-1.04593654e-05  5.18198617e-07 -4.26792360e-06  2.41735298e-07
   1.38186358e-05 -3.77743192e-06 -3.89866365e-08]
 [ 2.59710472e-06 -1.87877997e-06 -5.28298601e-06  3.96443409e-08
  -3.77743192e-06  1.58547228e-05 -6.77235360e-08]
 [-5.96893717e-07  1.92163542e-07 -5.21377294e-09  7.11842803e-10
  -3.89866365e-08 -6.77235360e-08  6.43668174e-09]]
omega =  -2.2487468892027604
tau =  0.024518795784839398
(-2.2732656849876, -2.224228093417921)
0 22566.7429361543
1 57648.09660575663
2 167888.28032172789
3 28108.648319770873
4 11684.812685098299
5 715.4564835629184
6 1205.5893718262441
alpha = 0.05 , 3.841458820694124
omega =  0.5085354659355167
y_ =  0.6244630907922954
```