CS 760 HW3 Lingyun Xiao

1(a) The step size is $\eta = 0.005$.

(b) It takes about 30 seconds to converge.

(c) $\hat{\theta} = (-0.53,\ 2.75,\ -0.016,\ -0.34,\ -0.15,\ 0.010)^T$

(d) $\ell(\hat{\theta}) = -403$.

(e) Following the theorem, $\hat{\theta} \overset{d}{\sim} N(\theta^*, I_{\theta^*}^{-1})$

See the attached code to get $I_{\theta^*}^{-1}$, which is 6×6 matrix.

2.(a) We know that the MLE of w^* is: $\hat{w} := \hat{\theta}^T x$.

(b) $\hat{w} \overset{d}{\sim} N(w^*, x^T I_{\theta^*}^{-1} x) = N(\hat{\theta}^T x, x^T I_{\theta^*}^{-1} x)$

3.(a) Let $x = (3, 1, 30, 0, 0, 20)$

$\hat{w} = \hat{\theta}^T x = 1.63 > 0 \Rightarrow$ Has survived.

(b) $\tau = \Phi_N^{-1}(0.025 \mid 0, x^T I_{\theta^*}^{-1} x) = 0.17$

$\Rightarrow$ 95% CI is $(\hat{w}-\tau, \hat{w}+\tau) = (1.46, 1.80)$.

(c) Since the entire CI is above 0, we can say it is certain that the person <u>has survived</u>.

4.(a) We can perform a χ^2 test on $\left(\frac{\theta_j}{v_j}\right)^2$:

$$\left(\frac{\theta_j}{v_j}\right)^2 \underset{H_0}{\overset{H_1}{\gtrless}} \Phi_{\chi^2}^{-1}(0.05) = 3.84$$

After calculation, we get:

(b) $\left(\frac{\theta_j}{v_j}\right)^2 = (57.5, 206, 9.70, 10.6, 1.64, 12.9)$

$\Rightarrow j = 1, 2, 3, 4, 6$ are significant, $j=5$ is <u>NOT</u> significant.

(c) The most significant feature is the gender.

If $X = (3, 0, 30, 0, 0, 20)$, $\hat{w} = -1.89 < 0$

In this case, the person has <u>NOT</u> survived.

In [159…

```
import numpy as np
import pandas as pd

data = pd.read_csv("titanic_data.csv")
df = pd.DataFrame(data)
```

In [160…

```
learning_rate = 0.005
likelihoods = []
epsilon = 1e-7

X = np.array(data.iloc[:,1:])
Y = np.array(data.iloc[:,0])

# Define the sigmoid function
def sigmoid(z):
    sigmoid_z = 1/(1+np.exp(-z))
    return sigmoid_z

# Define the log likelihood
def log_likelihood(y, y_pred):
    likelihood = np.sum(y*np.log(y_pred+epsilon)+(1-y)*np.log(1-y_pred+epsilon)
    return likelihood

theta = np.zeros((X.shape[1]))

# Perform Gradient Ascent

for i in range(100000):

    # Calculate z as the product of theta and x
    z  = np.dot(X,theta)

    # Output probability value by applying sigmoid on z
    y_pred = sigmoid(z)

    # Calculate gradient values
    gradient = np.mean((Y-y_pred)*X.T, axis=1)

    # Update theta
    theta += learning_rate*gradient

theta
```

Out[160]:

```
array([-0.5336015 ,  2.7526719 , -0.01618139, -0.33772808, -0.14603545,
        0.0096048 ])
```

In [161…

```
gradient
```

Out[161]:

```
array([-1.54159442e-09,  1.22123718e-08, -1.24244846e-11, -1.01674532e-09,
       -2.10736705e-09,  5.23267062e-12])
```

In [162…

```
z  = np.dot(X,theta)
y_pred = sigmoid(z)
log_likelihood(Y,y_pred)
```

Out[162]:

```
-403.4565786931986
```

In [163…

```
Fisher = np.zeros((6,6))
```

```
for i in range(len(X)):
    Fisher += (np.exp(-z[i]))/(1+np.exp(-z[i]))**2 * np.outer(X[i], X[i].T)
Fisher_inv = np.linalg.inv(Fisher)
Fisher_inv
```

Out[163]:

```
array([[ 4.95112514e-03, -4.56236633e-03, -2.51765841e-04,
        -1.92464589e-03, -1.07141836e-03,  5.16816843e-05],
       [-4.56236633e-03,  3.67201703e-02, -4.68958426e-05,
        -2.96899562e-03, -4.71854286e-03,  2.56776269e-06],
       [-2.51765841e-04, -4.68958426e-05,  2.69839953e-05,
         1.26973008e-04,  7.71811393e-05, -7.71071694e-06],
       [-1.92464589e-03, -2.96899562e-03,  1.26973008e-04,
         1.07846789e-02, -2.49545935e-03, -7.91951861e-05],
       [-1.07141836e-03, -4.71854286e-03,  7.71811393e-05,
        -2.49545935e-03,  1.30003013e-02, -7.97791024e-05],
       [ 5.16816843e-05,  2.56776269e-06, -7.71071694e-06,
        -7.91951861e-05, -7.97791024e-05,  7.13447288e-06]])
```

In [164…

```
omega_var = np.dot(Fisher_inv, X.T).dot(X)
omega_var
```

Out[164]:

```
array([[ 8.49186311e+00, -1.83411549e-01,  2.00562141e+01,
         3.31440033e-01,  2.12234841e-01,  3.03100879e+01],
       [-9.11299467e+00,  6.45470697e+00, -4.53322902e+01,
        -3.93140326e+00, -1.31397770e+00,  1.13284097e+02],
       [-5.77210560e-03, -1.48217545e-02,  5.79927955e+00,
        -7.00765513e-03, -1.73747347e-02, -7.24622863e+00],
       [ 1.57472518e+00, -3.96107634e-01, -2.49331802e+00,
         9.60501443e+00,  1.09873699e+00, -6.14602453e+01],
       [-7.40950837e-01, -5.42455958e-01, -2.02934421e+01,
        -6.11814568e-02,  5.25988462e+00, -1.07751613e+02],
       [ 1.70581590e-02,  3.48376687e-02,  7.41627833e-01,
         2.74980199e-03,  1.60485123e-02,  1.43282855e+01]])
```

In [165…

```
theta**2/(np.diag(Fisher_inv))
```

Out[165]:

```
array([ 57.50825312, 206.34987636,   9.70343625,  10.57613861,
         1.64045069,  12.93048068])
```

In [166…

```
x = np.array([3, 1, 30, 0, 0, 100])
```

In [167…

```
theta.dot(x)
```

Out[167]:

```
1.6269055204442842
```

In [168…

```
from scipy.stats import norm
sd = x.dot(Fisher_inv).dot(x)
tau = norm.ppf(0.025, scale=sd)
tau
```

Out[168]:

```
-0.169856521055011
```

In [169…

```
x2 = np.array([3, 0, 30, 0, 0, 20])
theta.dot(x2)
```

Out[169]:

```
-1.8941503013762506
```