RICE UNIVERSITY

COMP 540

Statistical Machine Learning

Honor Pledge:

On my honor, I have neither given nor received any unauthorized aid on this assignment.

Mingrui Liang (ml86)
Yuetong Yang (yy72)

HW # 2

Problem 1: Gradient and Hessian of $J(\theta)$ for logistic regression

Please see attached handwritten page (next page). The implementation of the last part is following the manuscript.

problem 1

- derivative of $g(z)$

since $g(z) = (1+e^{-z})^{-1}$

$$\frac{\partial g(z)}{\partial z} = -(1+e^{-z})^{-2}(-e^{-z})$$

$$= (1+e^{-z})^{-2} e^{-z}$$

$$= \frac{1}{1+e^{-z}} \quad \frac{e^{-z}}{1+e^{-z}} = g(z)\left(1-g(z)\right)$$

- gradient of L2 penalized cost function

we first calculate $\frac{\partial h_\theta(x)}{\partial \theta_j}$ $\quad j = 0, \cdots, d$

$$\frac{\partial h_\theta(x)}{\partial \theta_j} = \frac{\partial g(\theta^T x)}{\partial \theta_j} = \frac{\partial g(\theta^T x)}{\partial \theta^T x} \cdot \frac{\partial \theta^T x}{\partial \theta_j}$$

(using previous result) $= g(\theta^T x)\left(1-g(\theta^T x)\right) x_j$

$$= h_\theta(x)\left(1-h_\theta(x)\right) x_j$$

then

$$\frac{\partial}{\partial \theta_j} J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left\{ y^{(i)} \frac{1}{h_\theta(x^{(i)})} h_\theta(x^{(i)})(1-h_\theta(x^{(i)})) x_j^{(i)} + \right.$$

$$\left. (1-y^{(i)}) \frac{1}{1-h_\theta(x^{(i)})} \left[-h_\theta(x^{(i)})(1-h_\theta(x^{(i)})) x_j^{(i)} \right] \right\} + d$$

where $d = \begin{cases} 0 & \text{if } j = 0 \\ \frac{\lambda}{m}\theta_j & \text{if } j \neq 0 \end{cases}$

$$= -\frac{1}{m} \sum_{i=1}^{m} \left[\left(y^{(i)}(1-h_\theta(x^{(i)})) - \left(1-y^{(i)}\right) h_\theta(x^{(i)}) \right) x_j^{(i)} \right] + d$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} + d$$

hence $\frac{\partial}{\partial \theta} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)} + \frac{\lambda}{m} [0, \theta_1, \cdots, \theta_d]^T$

- vector form of $\frac{\partial}{\partial \theta} J(\theta)$

By observation we can see that

$\frac{\partial}{\partial \theta} J(\theta) = \frac{1}{m} X^T H_0 + \frac{\lambda}{m} [0, \theta_1, \cdots, \theta_d]^T$, where

$$X^T = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad H_0 = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

To verify this is the vector form of $\frac{\partial}{\partial \theta} J(\theta)$, we first check that the expression has a dimension of $(d+1) \times 1$. This can be easily shown since X^T is $(d+1) \times m$ and H_0 is $m \times 1$.

For the j^{th} element of $X^T H_0$, we can see that it equals to

$$x_j^{(1)} \left(h_\theta(x^{(1)}) - y^{(1)} \right) + x_j^{(2)} \left(h_\theta(x^{(2)}) - y^{(2)} \right) + \cdots + x_j^{(m)} \left(h_\theta(x^{(m)}) - y^{(m)} \right),$$

this is $\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$ in the previous expression.

Hessian of $J(\theta)$

The $(j,k)^{th}$ element of the hessian can be calculated by taking the j^{th} component of the gradient $\frac{\partial}{\partial \theta_j} J(\theta)$, then take derivative with respect to θ_k. That is

$$\frac{\partial^2}{\partial \theta_j \partial \theta_k} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)})(1 - h_\theta(x^{(i)})) \right) x_j^{(i)} x_k^{(i)} + \frac{\lambda d}{m}$$

$$\text{where } d = \begin{cases} 0 & \text{if } j = 1 \text{ or } j \neq k \\ 1 & \text{if } j = k \end{cases}$$

it matches with the expression $\frac{1}{m}\left(X^T S X + \lambda \begin{bmatrix} 0 & 0 \\ 0 & I_d \end{bmatrix}\right)$

Now show it's positive definite:

let $q \in \mathbb{R}^{d+1}$, we can see that

$$q^T H q = \frac{1}{m}\left[\sum_{i=1}^{m} h_\theta(x^{(i)})(1 - h_\theta(x^{(i)})) q^T x^{(i)} x^{(i)T} q + \lambda q^T \begin{bmatrix} 0 & 0 \\ 0 & I_d \end{bmatrix} q \right] > 0 \text{ since}$$

all the elements in this expression are strictly greater than 0.

Newton's method

The update formula would be

$$\theta_{t+1} = \theta_t - H^{-1} \nabla_\theta J(\theta)$$

$$= \theta_t - \left(X^T S X + \lambda \begin{bmatrix} 0 & 0 \\ 0 & I_d \end{bmatrix} \right)^{-1} \left(X^T H_0 + \lambda [0, \theta_1, \cdots, \theta_d]^T \right)$$

where $H_0 = \left[h_\theta(x^{(1)}) - y^{(1)}, \cdots, h_\theta(x^{(m)}) - y^{(m)} \right]^T$

Implementing Newton's method

My Python script is as follows:

```
import numpy as np

x = np.array([0,3,1,3,0,1,1,1])
x = x.reshape((4,2))
m,d = x.shape
X = np.hstack([np.ones((m,1)),x])
y = np.array([1,1,0,0])
theta = np.array([0,-2,1])
lam = 0.07

def h(theta,x):
    return 1/(1+np.exp(-theta.T@x))

for ind in range(2):
    H0 = np.array([])

    for i in range(m):
        H0 = np.append(H0,h(theta,X[i,:]))

    H0 = H0 - y

    S = np.array([])
    for i in range(m):
        S = np.append(S, h(theta,X[i,:])*(1-h(theta,X[i,:])) )

    S = np.diag(S)

    theta = theta - np.linalg.inv(X.T @ \
        S @ X + lam* np.diag( np.append([0],np.repeat(1,d)) ) ) \
            @ (X.T @ H0 + lam * np.append([0],theta[1:]).T )
    print(theta)
```

After 1 iteration, $\theta = [-4.67100739 \quad -0.00823255 \quad 2.34035274]^T$
After 2 iterations, $\theta = [-5.63804815e+00 \quad -1.73933112e-03 \quad 2.82056371e+00]^T$

Problem 2: Overfitting and unregularized logistic regression

Please see attached handwritten page (next page).

problem2

It will be easier to use -1 and 1 as labels here. By doing so, we get

$$P(y|x) = \left[1 + \exp\left(-y\theta^T x\right)\right]^{-1},$$

our loss function (negative log likelihood) is:

$$J(\theta) = -\sum_{i=1}^{m} \log\left(1 + \exp\left(-y^{(i)}\theta^T x^{(i)}\right)\right)$$

If we consider a hyperplane $\theta^T x = 0$ that linearly separates the dataset we will see that $y^{(i)}\theta^T x^{(i)} > 0$ for any i. (This is because the label will be 1 for $\theta^T x > 0$ and -1 for $\theta^T x < 0$)

Now after such hyperplane is found, its position (or direction in a 2-dimensional case) is fixed. To make our loss function smaller, all we can do is increasing the magnitude of θ.

When $\theta \to \infty$, the logistic function becomes a step function. When $x \geq 0$ $h_\theta(x) = 1$, when $x < 0$, $h_\theta(x) = -1$, it is no longer continuous.

To avoid such singular solution, we only need to add penalized term to our cost function. In that way the optimized θ will not go to infinity.

Problem 4: Implementing logistic regression

Problem 4B3: Varying λ

We compared situations where $\lambda = 0$ and 100, the result is as follows:

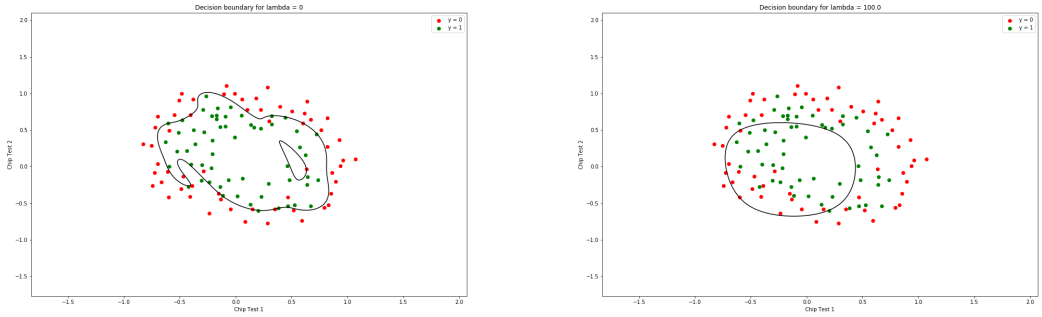

Figure 1: Comparison between $\lambda = 0$ and $\lambda = 100$. Left: $\lambda = 0$, right: $\lambda = 100$

We can see that when $\lambda = 0$ there is heavy overfitting issue. It wanted to fit the so well that the decision boundary actually ruled out the single red dot within the green dots. When $\lambda = 100$ it appears to have underfitting issue.

Problem 4B4: Exploring L1 and L2 penalized logistic regression

We compared 3 scenarios: $\lambda = 0.1, 1.0$ and 5.0.
When $\lambda = 0.1$, we can barely see any difference between L1 and L2 since the penalization is very weak.

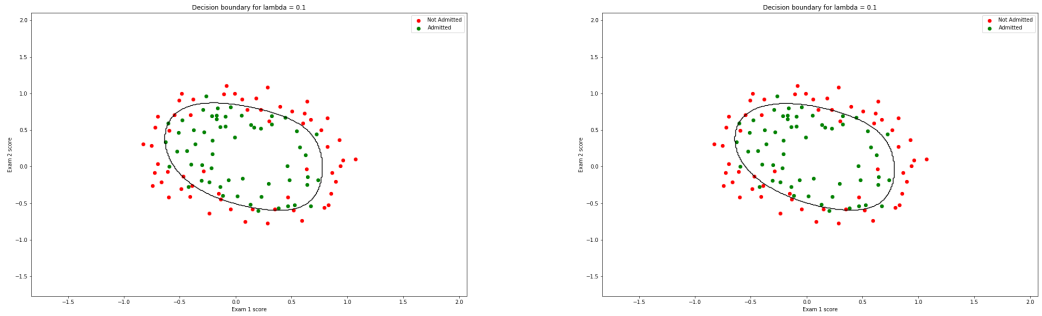

Figure 2: Comparison between L1 and L2 penalized logistic regresssion, $\lambda = 0.1$. Left: L2 penalized, right: L1 penalized

When $\lambda = 1.0$, it looks like the L1 regularization is slightly stronger since it includes fewer green points in the circle.

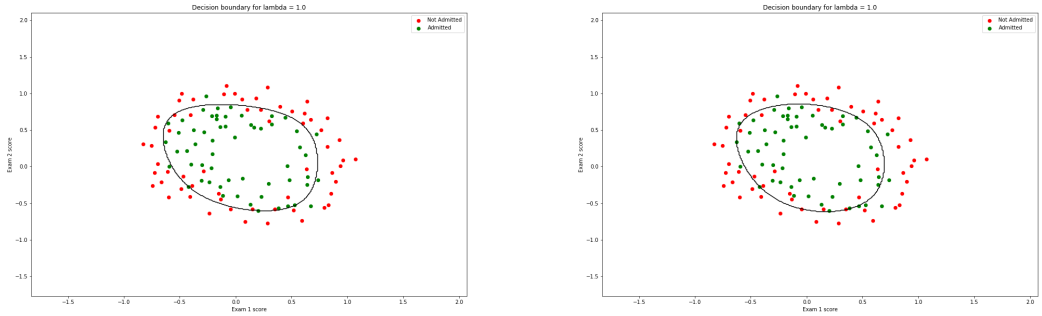

Figure 3: Comparison between L1 and L2 penalized logistic regresssion, $\lambda = 1.0$. Left: L2 penalized, right: L1 penalized

When $\lambda = 5.0$, the difference is huge between L1 and L2 regularization. There is actually no decision boundary for the L1 situation. This is because L1 is so strong that it almost penalizes all coefficients to 0.

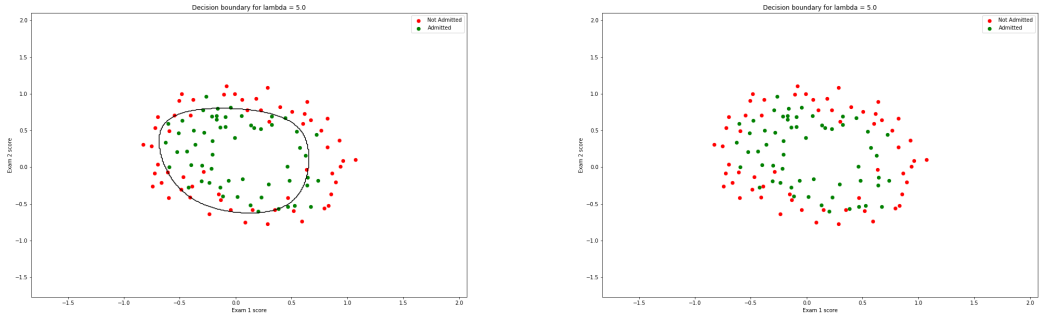

Figure 4: Comparison between L1 and L2 penalized logistic regresssion, $\lambda = 5.0$. Left: L2 penalized, right: L1 penalized

Problem 4 Part C: Logistic regression for spam classification

The summarization of the result is as follows:

Standarize features:
L2 penalty: best lambda-4.1, accuracy-0.9219
L1 penalty: best lambda-4.1, accuracy-0.9225;
Log transform features:
L2 penalty: best lambda-0.6, accuracy-0.9434
L1 penalty: best lambda-0.6, accuracy-0.9453;
Binarize features:
L2 penalty: best lambda-1.6, accuracy-0.9284
L1 penalty: best lambda-1.6, accuracy-0.9251;

As of **model sparsities**, we can see that L1 regularization tends to penalize terms to 0 while L2 rarely does so. The recommandation would be using L1 regularization because it has (slightly) higher accuracy. Also, using L1 we can oftentimes have a simpler model.