HW 3

YI NIAN
MPCS Machine learning, University of Chicago

4-26

I discussed some of the problem with Hanqi Zhang.

Problem 1:

Let the perceptron rule be: predict positive when h(x) $>$0 and predict negative otherwise.

Let the intercept column be all 1s and let the initial weight be [0,0,1]. Then the linear part of the model is now:

$0 * intercept + 0 * x1 + x2$

First, Pass the first data into the model:

$\hat{y} = \sum(1 * 0 + 1 * 0 + 1 * 1)$ = 1

$\triangle w = \alpha(y - \hat{y} * x)$ = 0.5*(0-1)*[1,1,1] = [-0.5,-0.5,-0.5]

w = w+$\triangle w$ = [-0.5,-0.5,0.5]

Next use the data (3,4):

we get w = w +$\alpha(y - \hat{y} * x)$ =[-0.5,-0.5,0.5]+ 0.5*(0-1)*[1,3,4]= $[-1, -2, -1.5]$

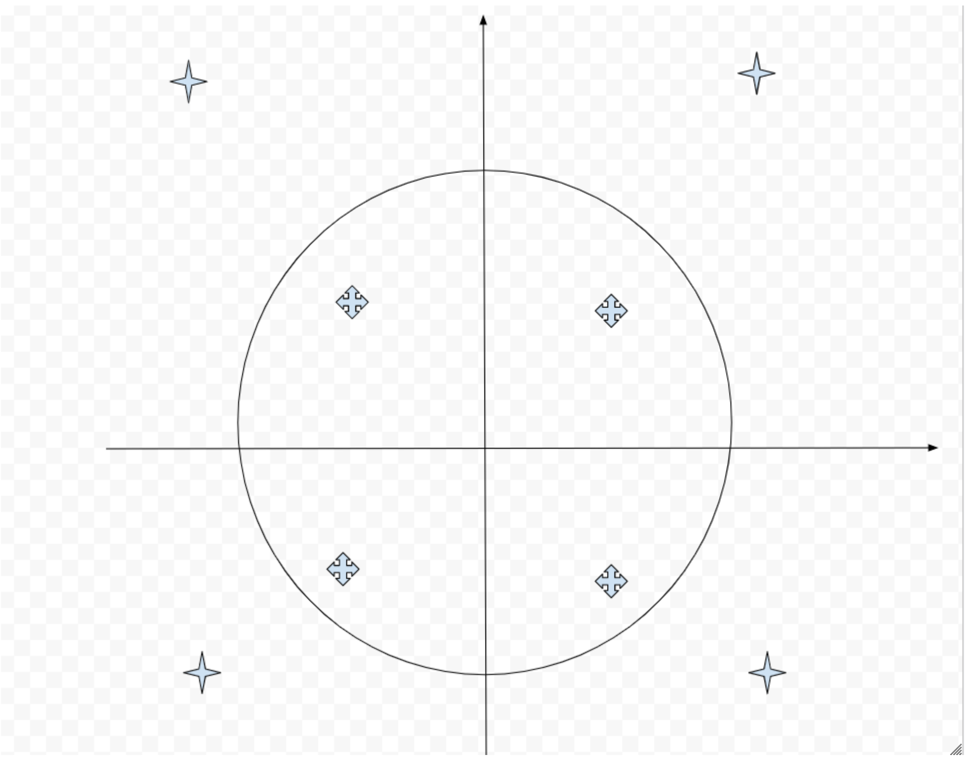
Next use the data(2,4.5)

we get w = w + $\alpha(y - \hat{y} * x) = [-1, -2, -1.5] + 0.5 * (1 - 0) * [1, 2, 4.5] = [-0.5, -1, 3/4]$

Now this model : $-0.5 * intercept - x1 + 0.75 * x2$ correctly predict every input.

Problem 2:

a:



b:

Assume we predict positive when

P(class = 1 | x1,x2) $>$ 0.5 and negative otherwise.

Then we actually get:

1/(1+exp-(W_0+$W_1 x_1^2$ + $W_2 x_2^2$)) $>$ 0.5

which is equivalent to

W_0+$W_1 x_1^2$ + $W_2 x_2^2 > 0$

Using the data we have,

W_0+0.25W_1 + 0.25$W_2 > 0$

W_0+4W_1 + 4$W_2 < 0$

Solve this, we will get infinite many solutions. One of them would be:

W1 = -1, W2 = -1, W0 = 5

c:

For the first data:

-0.5-0.5+5 will give 4, which implies P(class = 1 | x1,x2) = 0.98 $>$0.5, and this gives a positive class label.

For the last data:

The linear part is -4-4+5 = -3, which implies P(class = 1 | x1,x2) = 0.047 $<$ 0.5, and this gives a negative label.

Problem 3:

The assumption of multinomial logistic regression is actually $\log\frac{P(Y=k|x)}{P(Y=j|x)}$ and usually we take some J as a "base case" .

But here, we just need to show that for each pair of k,j, we have a linear boundary.

So, $\log\frac{P(Y=k|x)}{P(Y=j|x)}$

$\implies$ log(exp(w$_k^T x$)/$exp(w_j^T x)$) = $(W_k - W_j)$X using the equation of given information.

And this equation is a linear combination of input x.

Problem 4:

See the python code

Problem 5:

The hyperparameters used as follows:

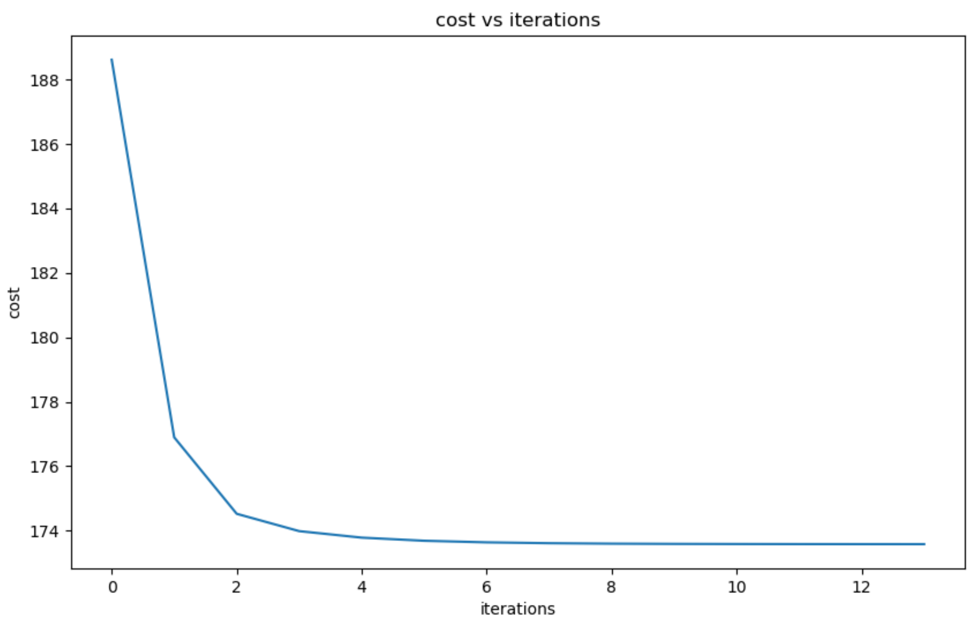
$LogisticRegression(random_state = 1, penalty =' l2', C = 0.01, max_iter = 10000, solver =' liblinear')$

The gradient function will stop once the cost converge, so the number of iterations is actually less than 10000.

The theta output is as follows:

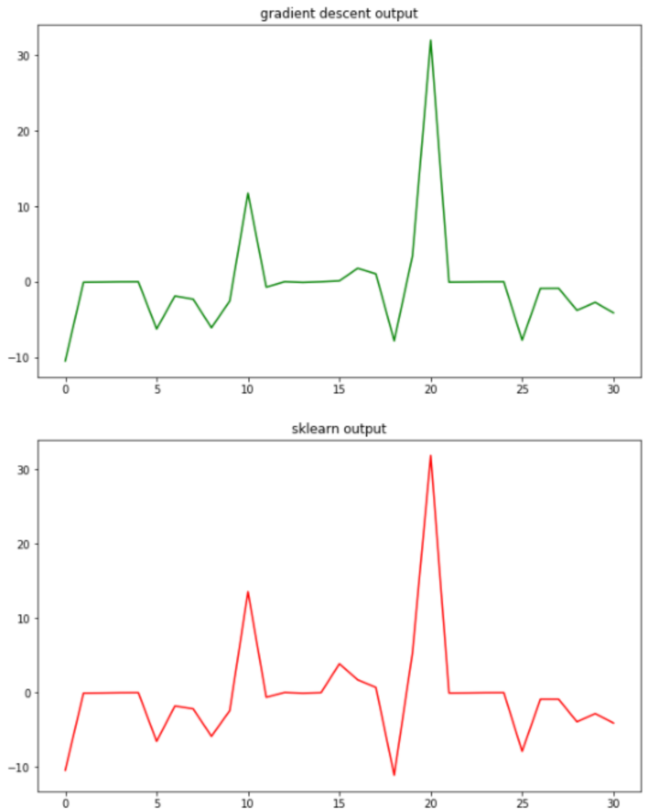
```
sklearn theta output is :
[-1.04252569e+01 -6.44899287e-02 -4.37711692e-02 -9.28431675e-03
 -6.26617271e-04 -6.25020288e+00 -1.79977227e+00 -2.27505655e+00
 -6.02202418e+00 -2.47719241e+00  1.22113602e+01 -6.89558770e-01
  1.05246480e-02 -8.17596126e-02 -3.78478545e-03  4.77256908e-01
  1.87925324e+00  9.51839054e-01 -8.64587370e+00  3.94476306e+00
  3.18046247e+01 -5.46726924e-02 -3.73475356e-02 -7.59911049e-03
 -4.28664726e-04 -7.79212044e+00 -8.73963000e-01 -8.73777347e-01
 -3.81334990e+00 -2.76444422e+00 -4.16199268e+00]
my gradient descent theta output is
[-8.22203946e+00 -5.15072725e-02 -3.16984189e-02 -7.45910968e-03
 -5.01933629e-04 -5.08245412e+00 -1.76728094e+00 -1.82587844e+00
 -4.75928488e+00 -2.15690184e+00  7.84423074e+00 -5.21806509e-01
  6.61747659e-03 -6.44297145e-02 -2.96585346e-03  2.78511792e+00
  3.45625328e-01  4.51806119e-01 -7.44417838e+00  2.27128344e+00
  2.02169309e+01 -4.24425922e-02 -2.63001730e-02 -5.96670686e-03
 -3.36067878e-04 -5.62853583e+00 -7.47752579e-01 -6.83459967e-01
 -2.94599018e+00 -2.01495016e+00 -3.32584735e+00]
[Finished in 9.4s]
```

The cost function plot is as follows:



Problem 6**:**

A comparison between gradient descent coefficient and sklearn coefficient:



Why θ is different:

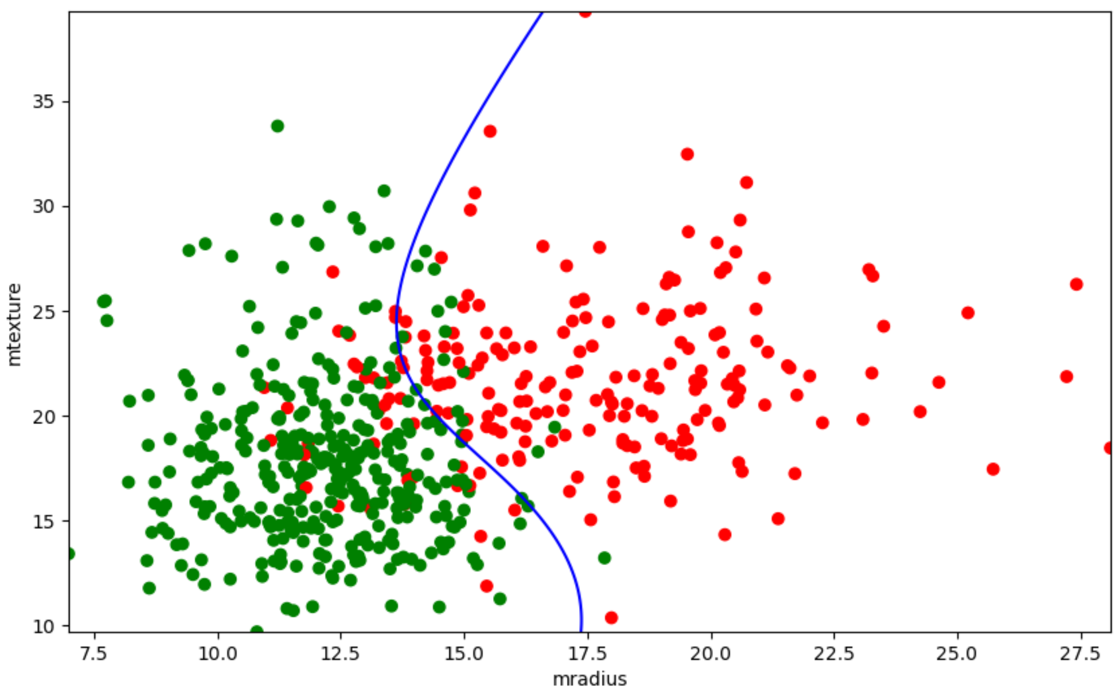
I think the main reason is how I define the cost function and the gradient update function. For example, sometimes we can divide m (number of examples) in calculating the cost or we can divide 2m in the formula in P-1.And I did not divide them by m, but this is also valid.

$$-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log h_{\theta}(\mathbf{x}^{(i)}) + (1-y^{(i)})\log(1-h_{\theta}(\mathbf{x}^{(i)}))\right) + \lambda\sum_{j=1}^{n}\theta_j^2,$$

And also, when updating the parameters, we could multiply by 2 before the lambda as the formula shows. But the sklearn did not have the 2 here.

$$\theta_j \leftarrow \theta_j + \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - h_{\theta}(\mathbf{x}^{(i)})\right)x_j^{(i)} - 2\lambda\theta_j\right].$$

Problem 7:



35
30
25
20
15
10
mtexture
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
mradius