EE6310

Problem Set 8

In this problem set, we will develop the logistic regression. Here we will predict whether a material is metallic or not using the fractional composition of elements. It looks like below:

Metallic	**H%**	**Li%**	**Be%**	**…**	**U%**
y	x_1	x_2	x_3		x_{79}
1	0.41	0	0	…	0
0	0	0	0	…	0
0	0	0.04	0	…	0
...	...	...	...	…	

The full data can be downloaded from the Assignments menu "MetalbyComp.csv" or in the weekly learning as the "HW8data.csv." You can recycle codes from the previous work. That's the beauty of coding.

(a) Write the code to load the HW8data.csv. You can refer to the lecture 5 or Google it. Then convert the pandas data frame into numpy and separate the data into x and y.

(b) Let's write the logistic regression function. We will use linear model to calculate the *z*. So you need to implement:

$$f_{\boldsymbol{w},b}(\boldsymbol{x}) = g(z) = g(\boldsymbol{w}\cdot\boldsymbol{x}+b) = \frac{1}{1+exp\big(-(\boldsymbol{w}\cdot\boldsymbol{x}+b)\big)}$$

The input to this function is $\boldsymbol{w}$, b, and $\boldsymbol{x}$. Use the matrix multiplication from the previous homework to vectorize over the entire data set, $\boldsymbol{X} \in \mathbb{R}^{m\times n}$.

$$z = \boldsymbol{X}\boldsymbol{w} + b$$

```
def logistic(w,b,x):
    # your code
    return f
```

If the code is correct, you should get the result in the comment

```
print(logistic(np.ones(79),1,x).sum()) # 999.300698
```

(c) Implement the cost function. Let's vectorize over the training examples as well.

$$J(\boldsymbol{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log\left(f_{\boldsymbol{w},b}\left(\boldsymbol{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_{\boldsymbol{w},b}\left(\boldsymbol{x}^{(i)}\right)\right)]$$

In order to vectorize this function, we can introduce the vector version *y*, $\boldsymbol{Y} \in \mathbb{R}^m$. Then, the $\boldsymbol{L} \in \mathbb{R}^m$ the vector of loss function value is:

$$\boldsymbol{L} = \boldsymbol{Y} \odot \log\left(f_{\boldsymbol{w},b}(\boldsymbol{X})\right) \oplus (1 - \boldsymbol{Y}) \odot \log\left(1 - f_{\boldsymbol{w},b}(\boldsymbol{X})\right)$$

where $\odot$ and $\oplus$ is the element wise multiplication and addition. The cost function is:

$$J = -\bar{\boldsymbol{L}}$$

where the bar indicates the average of the Loss vector. Remember the numpy broadcasting!

```
def cost_function(w,b,x,y):
    # your code
    return J
```

If the code is correct, you should get the result in the comment

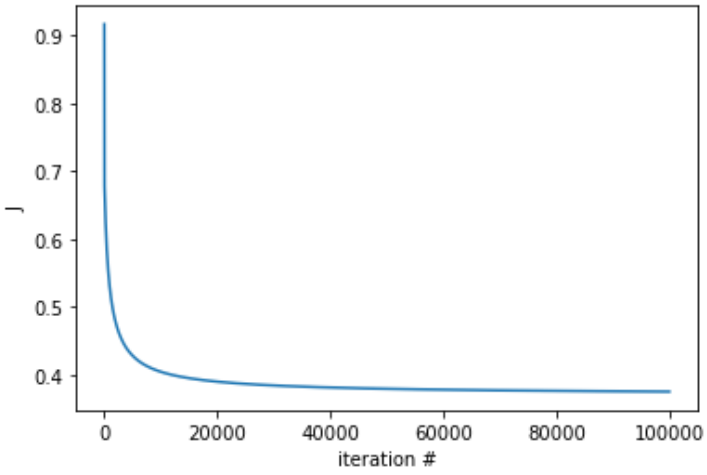
```
print(cost_function(np.ones(79),1,x,y).sum()) #  1.064981
```

(d) Finally, let's implement the gradient descent. Note that the gradient descent looks very similar to the gradient descent of the multiple linear regression. Just the function needs to be changed. Utilize the code from the previous homework to implement the gradient descent.

```
def gradient_descent(w,b,x,y,alpha):
    # your code
    return w,b
```

If your code is correct, you should get a figure shown below up running the code below:

```
import matplotlib.pyplot as plt
w = np.ones(79)
b = 1
J = []
for i in range(100000):
    w,b = gradient_descent(w,b,x,y,1)
    J.append(cost_function(w,b,x,y))
    if len(J) > 2 and J[-2] - J[-1] < 1E-11:
        break
plt.plot(np.arange(len(J)),J)
plt.xlabel('iteration #')
plt.ylabel('J')
plt.show()
```



(e) With the trained model, let's calculate the accuracy of the model. The accuracy is defined as:

$$\text{Accuracy} = \frac{TP + TN}{m}$$

where TP is the number of true positive, and the TN is the number of true negative. Remember that $\hat{y} = 1$ if $g(\boldsymbol{z}) \geq 0.5$ and $\hat{y} = 0$ if $g(\boldsymbol{z}) < 0.5$. A datum is true positive if $\hat{y} = \text{y} = 1$ and a datum is true negative if $\hat{y} = \text{y} = 0$. Can you predict whether a material is metallic or not with just the composition alone?