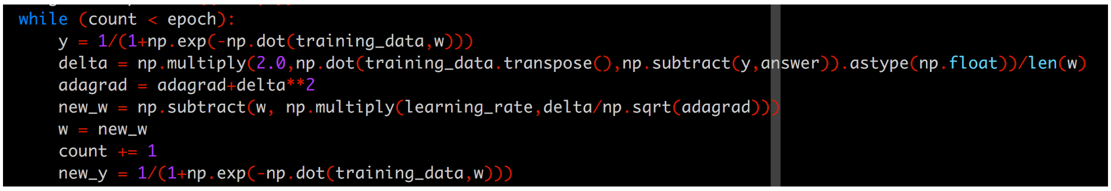
Machine Learning
Spam Classification

B02901031 陳緯哲

- ## Logistic Regression Function

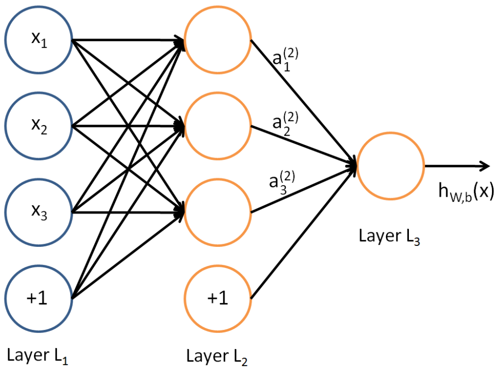


```
while (count < epoch):
    y = 1/(1+np.exp(-np.dot(training_data,w)))
    delta = np.multiply(2.0,np.dot(training_data.transpose(),np.subtract(y,answer)).astype(np.float))/len(w)
    adagrad = adagrad+delta**2
    new_w = np.subtract(w, np.multiply(learning_rate,delta/np.sqrt(adagrad)))
    w = new_w
    count += 1
    new_y = 1/(1+np.exp(-np.dot(training_data,w)))
```

- ## Neural Network

1. Method

Using a **one layer neural network** as my second method to classify.



Input : **57** attributes + **1** bias

Hidden layer : **30 + 1** nodes

Output : **1** node

Activate function : **Sigmoid** function

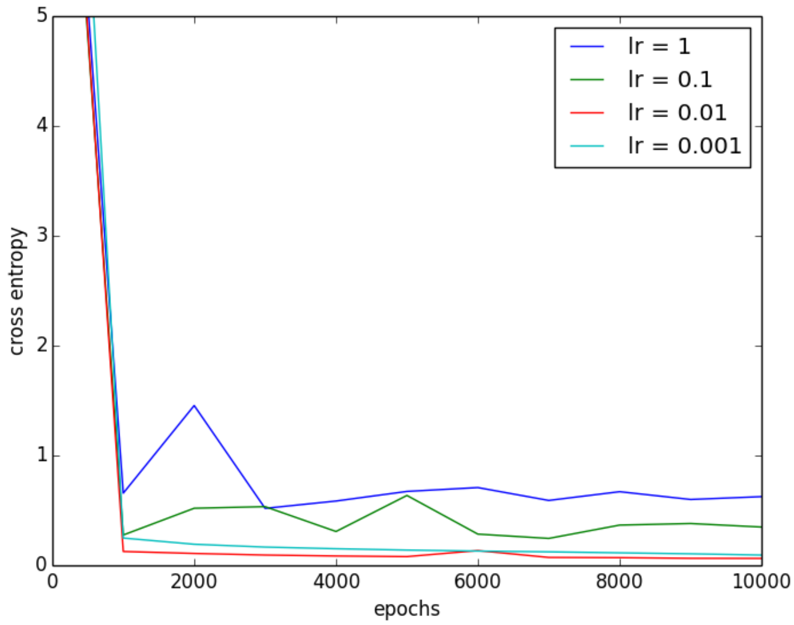
Loss Function : **Cross-entropy**

Optimize strategy : **Adam**

Best kaggle score : 0.9566

2. Experiment

a. Finding the best learning rate:



λ = 0.5 , epochs = 10000

It is obvious that when learning rate = 0.01 or 0.001 has the best performance. In the latter experiments, I choose 0.001.

b. In this experiment, I randomly split the data into 2 parts, half for training, and half for testing. By comparing the cross entropy of test data with that of train data, we can find the best model to avoid overfitting.

Finding the best λ :

λ	10	1	0.5	0.1	0.01
train	0.1983	0.1511	0.1374	0.1156	0.085
test	0.2218	0.1638	0.1560	0.1426	0.1965

epochs = 5000 , learning rate = 0.001

The best model is λ = 0.1 , however, the kaggle best score is obtained with λ = 0.5. I finally choose λ = 0.5 as the kaggle best in that it provides smoother weights that can prevent from overfitting.

c. **Compare different hidden layer nodes**

Nodes	10	20	30	40	50
Cross entropy	0.1597	0.1390	0.1303	0.1245	0.1258
Training time(s)	12.13	20.0	27.65	35.56	51.81

epochs = 5000

- ## Summary

Since the kaggle score separate into private set and public set, it is important to takes strategies to avoid overfitting. As result, I applied Regularization and Validation to find out the best model that help avoid overfitting. Based on the experiments, I found that model with λ = 0.1 and nodes = 40 get the best performance. However, I choose the parameters that will get smoother weights. That is, I choose λ = 0.5, epochs = 5000, nodes = 31, and learning rate = 0.001 as my kaggle best, and that reach the kaggle score 0.95333.