Machine Learning 2016
系級：電機四
姓名：鍾勝隆
學號：B02901001

HW2 Report – Spam Classification

1.**Logistic Regression** function by Gradient Descent.

```
#intial coefficient(import numpy as np)
weight = np.zeros((1, 57))
learning_time = 50000
bias = 0
#Adadelta
G_w = np.zeros((1, 57))
G_b = 0
t_w = np.zeros((1, 57))
t_b = 0
T_w = np.zeros((1, 57))
T_b = 0
gamma = 0.9
epsilon = 10 ** -8

t = 1
while(True):
  z = np.sum(data * weight, axis=1) + bias
  f_wb = 1 / (1+ math.e ** (-z))
  change = answer - f_wb
  gradient_b = -1 * (change.sum())
  gradient_w = -1 * (np.sum(np.transpose(data) * change, axis=1)

  #gradient adadelta
  G_w = gamma * G_w + (1 - gamma) * (gradient_w ** 2)
  G_b = gamma * G_b + (1 - gamma) * (gradient_b ** 2)
  t_w = -(((T_w + epsilon) ** 0.5) / ((G_w + epsilon) ** 0.5))  * gradient_w
  t_b = -(((T_b + epsilon) ** 0.5) / ((G_b + epsilon) ** 0.5))  * gradient_b
  T_w = gamma * T_w + (1 - gamma) * (t_w ** 2)
  T_b = gamma * T_b + (1 - gamma) * (t_b ** 2)
  weight += t_w
```

```
    bias += t_b

    if ( t > learning_time):
        print ("Logistic Regression training is done.")
        break
    t += 1
```

2.**Neural Network** function by Backpropagation.

```
    #intial coefficient  (import numpy as np)
    #intial coefficient
    node = 50
    weight_1 = np.zeros((57, node))
    bias_1 = np.zeros((1, node))
    weight_2 = np.zeros((1, node))
    bias_2 = 0
    learning_time = 5000
    #Adadelta
    G_w1 = np.zeros((57, node))
    G_b1 = np.zeros((1, node))
    t_w1 = np.zeros((57, node))
    t_b1 = np.zeros((1, node))
    T_w1 = np.zeros((57, node))
    T_b1 = np.zeros((1, node))
    G_w2 = np.zeros((1, node))
    G_b2 = 0
    t_w2 = np.zeros((1, node))
    t_b2 = 0
    T_w2 = np.zeros((1, node))
    T_b2 = 0
    gamma = 0.9
    epsilon = 10 ** -8

    t = 1
    while(True):
        z_1 = np.dot(data, weight_1) + bias_1
        a_1 = 1 / (1+ math.e ** (-z_1))
        z_2 = np.sum( a_1 * weight_2, axis=1) + bias_2
        y = 1 / (1+ math.e ** (-z_2))
```

```
change = answer - y
gradient_b2 = -1 * (change.sum())
gradient_w2 = -1 * (np.sum(np.transpose(a_1) * change, axis=1))

delta_1 = (a_1*(1-a_1)) * np.dot(np.transpose([change]), weight_2)
gradient_b1 = -1 * np.sum(delta_1, axis=0)
gradient_w1 = -1 * np.dot(np.transpose(data), delta_1)

#gradient_1 adadelta
G_w1 = gamma * G_w1 + (1 - gamma) * (gradient_w1 ** 2)
G_b1 = gamma * G_b1 + (1 - gamma) * (gradient_b1 ** 2)
t_w1 = -(((T_w1 + epsilon) ** 0.5) / ((G_w1 + epsilon) ** 0.5))  * gradient_w1
t_b1 = -(((T_b1 + epsilon) ** 0.5) / ((G_b1 + epsilon) ** 0.5))  * gradient_b1
T_w1 = gamma * T_w1 + (1 - gamma) * (t_w1 ** 2)
T_b1 = gamma * T_b1 + (1 - gamma) * (t_b1 ** 2)
weight_1 += t_w1
bias_1 += t_b1

#gradient_2 adadelta
G_w2 = gamma * G_w2 + (1 - gamma) * (gradient_w2 ** 2)
G_b2 = gamma * G_b2 + (1 - gamma) * (gradient_b2 ** 2)
t_w2 = -(((T_w2 + epsilon) ** 0.5) / ((G_w2 + epsilon) ** 0.5))  * gradient_w2
t_b2 = -(((T_b2 + epsilon) ** 0.5) / ((G_b2 + epsilon) ** 0.5))  * gradient_b2
T_w2 = gamma * T_w2 + (1 - gamma) * (t_w2 ** 2)
T_b2 = gamma * T_b2 + (1 - gamma) * (t_b2 ** 2)
weight_2 += t_w2
bias_2 += t_b2
if (t % 1 == 0):
   print("The", t, "times__Cross Entropy:", \
            E_function(weight_1, bias_1, weight_2, bias_2, answer, data) )
if ( t > learning_time):
   print ("Neural Network training is done.")
   break
t += 1
```

3. Describe method1 (Logistic Regression).

將 spam_train.csv 中的資料去除掉 id，取出爲一筆資料：

$$((x_1^n, x_{2,..}^n, x_{57}^n), y^n)$$

共可以產生 4001 筆 data，使用 numpy 的二維 array(shape 爲(4001*57))儲存，設定參數一維 array，每個 feature 都有對應的 weight，故我初始化 57 個 0 起始值以及設 bias 也爲 0，由於 feature 之間數值大小差異非常大，有的是幾乎都爲 0，有的則是到 1000 以上，故將全部的 feature 都各自算出 mean 和 variance，把每個 feature 都標準化，減少 weight 調整上的不易。

在 training 中，，由於是 classification 的問題，feature 經過 weight、bias 以及 sigmoid function 的運算，得出的結果爲 0 到 1 的值，巧好此次分類問題爲兩類，視爲機率，並可以-ln(likelihood)，即是計算出的 $f(x^n)=\sigma(\sum w_i x_i^n+b)$ 和 y^n 的 Cross Entropy，作爲 loss function：

$$\sum_{x^n, y^n} -[y^n \ln f(x^n)+(1-y^n)\ln(1-f(x^n))]$$

計算對應各個 feature 的 gradient，因爲我有使用 Adadelta，故將每次計算的 gradient 平方後，依照比例（1:9）累積在 G_w 和 G_b ，與極小值的 ϵ ，以 $1/\sqrt{(G_w+\epsilon)}$ 和 $1/\sqrt{(G_b+\epsilon)}$ 再乘上之前的 weight 和 bias 變化量參數 $\sqrt{(T_{w,b}+\epsilon)}$ 和 gradient，加上原本的 weight 和 bias，即完成 Adadelta 的 Gradient Descent。($\delta t_{w,b}$ 爲 weight 和 bias 每次 train 完後的變化量)

$$G_{w,b}=\gamma\times G_{w,b}+(1-\gamma)\times gradient_{w,b}^2$$

$$T_{w,b}=\gamma\times T_{w,b}+(1-\gamma)\times \delta t_{w,b}^2$$

$$weight, bias = weight, bias + \frac{\sqrt{(T_{w,b}+\epsilon)}}{\sqrt{(G_{w,b}+\epsilon)}}\times gradient_{w,b}$$

如何設定 training time(次數) ，我是透過 **K-fold cross-validation(交互驗證)** 去評估各個參數的最佳值，將全部的資料**分成 10 等分**，其中 1 份當作 Validation set，剩下 9 份爲 Training set，故產生 10 組 Training set 和 Validation set，平行 train 10 個獨立的 Training set，使用 Validation set 算 train 出資料的 loss function，再由 10 個 loss 的 **mean** 和 **variance** 評估該參數建立的 model 是否較好，調整 training times ，

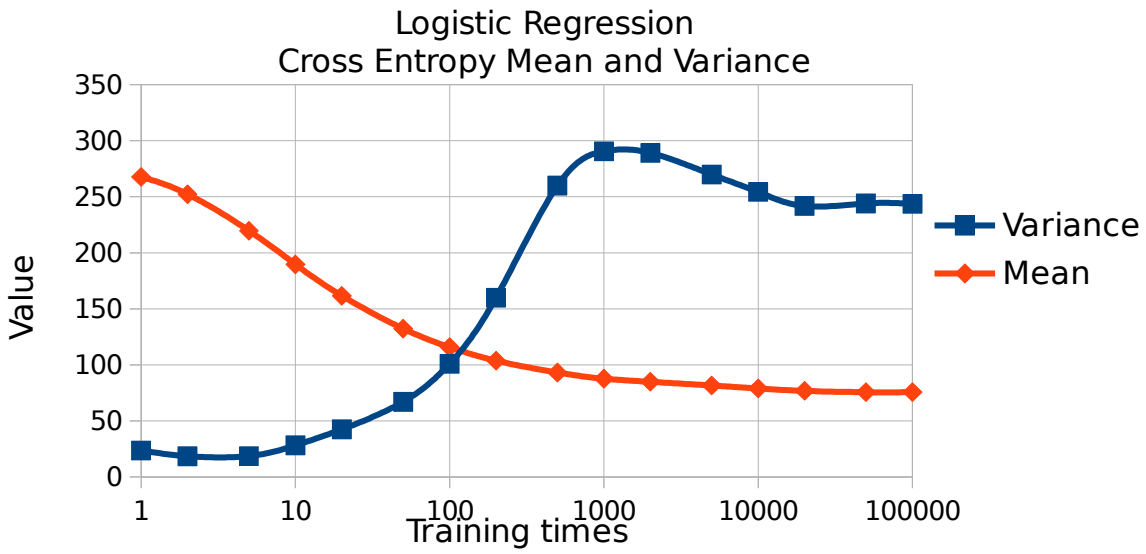
以達到最低的 mean 和 variance，並再輸入 logistic_regression.py 中，把全部的 4001 筆資料重新 train 一遍，使用 pickle 輸出 model。再用 test.py 讀入 model 和 spam_test.csv，計算出答案。

4.Describe method2（Neural Network – 只有與 method1 不同處)

Neural Network 是將 sigmoid function 疊層化，我在這裏的做法是將一開始的 57 個 feature 通過一個 layer 轉換成 50 個維度的資料（經過 validation 評估，50 效果較佳），透過與兩個矩陣 W 大小爲（57,50）和 b 大小爲（1,50）進行運算，再通過 sigmoid function，由剩下的 50 個值作 logistic regression。另外，Loss function 跟 method1 相同，但是需要注意的是，由於還是使用 gradient descent 的方法，所以 loss function 還是需要對各個 weight、bias、 W 和 b 作 gradient，後半 logistic regression 的部分與 method1 相同，但是 W 和 b 是透過 Backpropagation 的方法，計算 gradient。

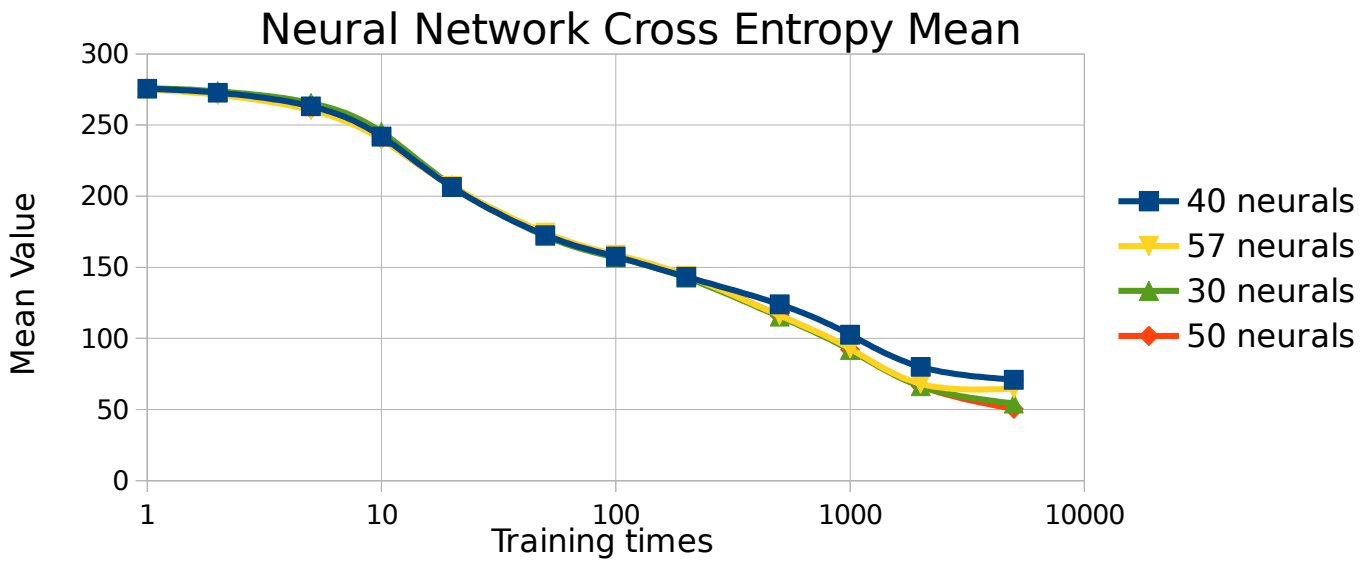
5.參數調整影響

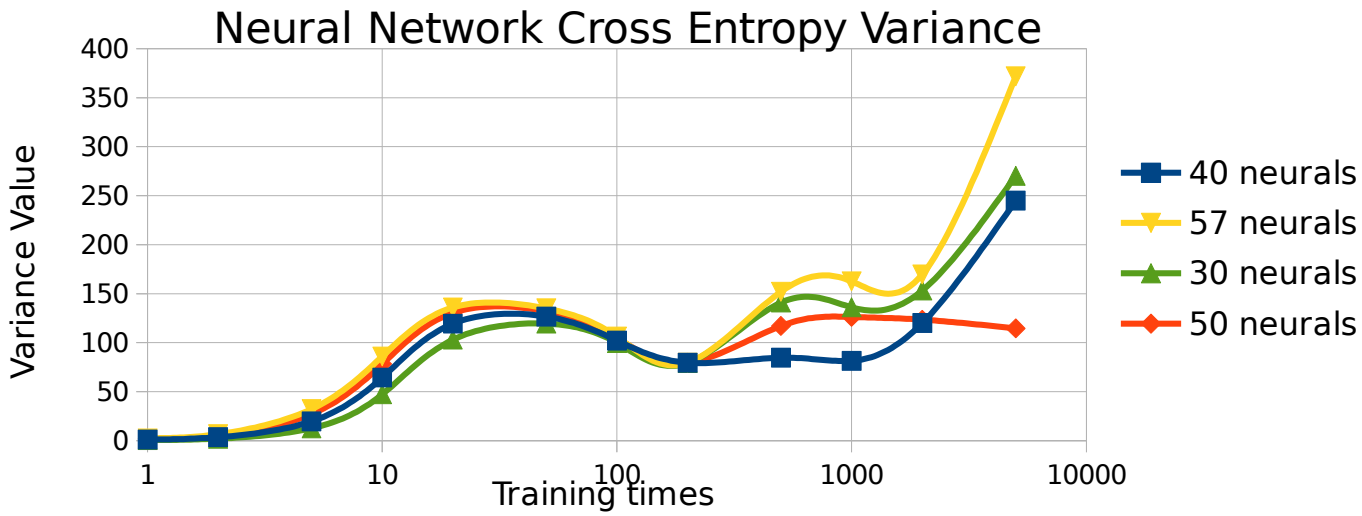
透過 Cross Validation 評估參數大小，在 method1 中 Logistic Regression 由於使用 adadelta，只剩下 training time 這個主要變因，故針對 training 次數多寡計算 10 個 Cross Entropy 的 Mean 和 Variance：



可以觀察到 training 次數增加，Mean 的減少和 Variance 的增加，代表 model 的 fitting 程度增加。由於當超過一萬次時 Variance 沒有顯著的增加，最後取 training time 爲 20000 次，作爲 model。

在 Neural Network 則多了 layer 中要幾個 neural 這個變量，故以不同數量的 neural 和 training time 作爲變因進行實驗：





觀察到在使用 50 個 neural 和 training time 爲值 5000 次時，Mean 和 Variance 都爲最低值，故以此參數進行 training。

6. Reference

NTUEE – ML2016Fall 上課投影片（Logistic Regression）
http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html
NTUEE – ML2015Fall 上課投影片（Neural Network）
http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html
An overview of gradient descent optimization algorithms (Adadelta)
http://sebastianruder.com/optimizing-gradient-descent/