

# Machine Learning HW2 Report

b02901122 電機四 劉致廷

## 1. Logistic regression function

### 1) Function Set & Cross Entropy

```
Y = 1.0 / (1.0 + np.exp(-np.dot(X, W)))  
loss = np.mean(-(Y * np.log(Y + e) + (1 - Y) * np.log(1 - Y + e)))
```

### 2) Gradient

```
Grad = -np.dot(X.T, (Y - Y)) / len(X)
```

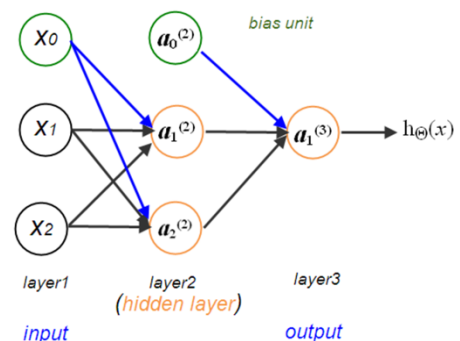
### 3) Optimizer: AdamOptimizer (Will explain below)

Best Kaggle Accuracy: 0.92667

## 2. Describe your another method, and which one is best.

I use **Neural Network** for my another method, the model is like the figure below.

The only **one** hidden layer neurons are 40, and I combine the bias unit in it, so there are **41 hidden neurons**, the input layer neurons are **58**, which are the 57 input features and one bias unit, and the output layer is a neuron used to predict two classes, all the activation function I used are **sigmoid**.



The optimizer I used was **AdamOptimizer**, because the downtrend of the loss when I used Gradient Descent is sluggish. Therefore, the loss would drop steadily. The parameters and the algorithm are as below.

**learning\_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08, g = Gradient**

```
m_0 <- 0 (Initialize initial 1st moment vector)  
v_0 <- 0 (Initialize initial 2nd moment vector)  
t <- 0 (Initialize timestep)
```

```
t <- t + 1  
lr_t <- learning_rate * sqrt(1 - beta2^t) / (1 - beta1^t)
```

```
m_t <- beta1 * m_{t-1} + (1 - beta1) * g  
v_t <- beta2 * v_{t-1} + (1 - beta2) * g * g  
variable <- variable - lr_t * m_t / (sqrt(v_t) + epsilon)
```


Best Kaggle Accuracy: 0.9600, which is much better than logistic regression.

### 3. How did I choose the nn model ?

First, I try to use the **regularization** to avoid overfitting, so it just need to add a  $\lambda W$  to the gradient. Therefore, I use **3601 training data & 400 validation data** to choose my model. After I choose the  $\lambda$ , I try to base on the  $\lambda$  and then choose the **number of hidden neurons**.

The **stop condition** when I use training data to train and use validation data to test is that when the output of the network would not change too much( less than 0.004), I would stop the training and start to use validation data to test.

The graph below is the model I chose.

$\lambda = 0.01$ (neuron=32)	Validation ACC = 0.9649		$\lambda = 0.01$ , 16 neurons	Validation ACC = 0.9525
$\lambda = 0.1$ (neuron=32)	Validation ACC = 0.9549		$\lambda = 0.01$ , 32 neurons	Validation ACC = 0.9649
$\lambda = 10$ (neuron=32)	Validation ACC = 0.9525		$\lambda = 0.01$ , 40 neurons	Validation ACC = 0.9675
$\lambda = 100$ (neuron=32)	Validation ACC = 0.9605		$\lambda = 0.01$ , 64 neurons	Validation ACC = 0.9649

Result1: fix  $\lambda = 0.01$  , start to tune the hidden neurons,

Result2: the final model:  $\lambda = 0.01$  40 neurons (hidden layer)