1

Logistic Regression Function

The function used in logistic regression is as follows: In the first step, we pass the data through the regression layer and denote the output of the $\mathbf{i^{th}}$ logistic regression output unit as

$\mathbf{z^i = w^i x + b}$ where $\mathbf{i} \in [\mathbf{0}, \mathbf{K}]$.

Here, **K** is the number of classes amd W indicates weight and b is the bias term.

After the values of the logistic units are calculated, the outputs are passed through a softmax layer to calculate the probability of each class

$\mathbf{y^{(i)}} = \frac{\mathbf{e^{z^{(i)}}}}{\sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{e^{z^{(k)}}}}$

$\mathbf{y^{(i)}}$ denotes the probability of class **i**.

Graditent Derivation

We have been using cross entropy loss function for Logistic Regression and the loss function is

$$L = - \sum_{\mathbf{i=1}}^{\mathbf{K}} \mathbf{t^i} * \mathbf{log}(\mathbf{y^i})$$

where t^i is true label for the unit i.

derivate of Loss w.r.t Weights = $\frac{\partial \mathbf{L}}{\partial \mathbf{W}} = \frac{\partial \mathbf{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{w}}$ – (based on chain rule)

$\frac{\partial \mathbf{L^i}}{\partial \mathbf{y^i}} \frac{\partial \mathbf{y^i}}{\partial \mathbf{z^i}} = \mathbf{y^i - t^i}$

$\frac{\partial \mathbf{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{w}} = \mathbf{x^i} * (\mathbf{y^i - t^i})$

Based on gradient derivation the weight update will be defined as

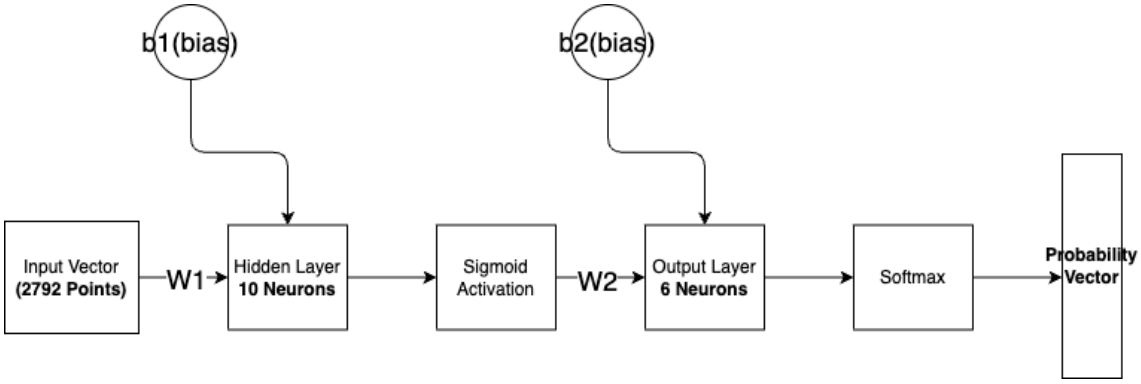
$\mathbf{W^{t+1}} = \mathbf{W^t}$ - α * $\frac{\partial \mathbf{L}}{\partial \mathbf{W}}$

$\mathbf{W^{t+1}} = \mathbf{W^t}$ - $\alpha * \mathbf{x^i} * (\mathbf{y^i - t^i})$

2

Neural Network Architecture Details

From a higher level the Neural network architecture is depicted as follows.



Number of hidden Layers = 1
Number of Neurons in the hidden layer = 10
Activation in Hidden Layer = Sigmoid
Number of Neurons in output layer = 6
Activation in output layer = softmax

Loss Function L = - $\sum_{\mathbf{i=1}}^{\mathbf{K}} \mathbf{t^i} * \mathbf{log}(\mathbf{y^i})$
$\mathbf{y^{(i)}}$ denotes the probability of class i and where t^i is true label for the unit i.

Forward pass equation

$$y_i = \frac{e^{W_2\sigma(W_1x+b_1)+b_2}}{\sum_{i=1}^{k} e^{W_2\sigma(W_1x+b_1)+b_2}}$$

x is the input vector
W_1 is the weight matrix for the hidden layer
b_1 is the bias vector for the hidden layer.
$\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid activation function.
W_2 is the weight matrix for the output layer.
b_2 is the bias vector for the output layer.
k is the number of output classes.

Gradient Calculation and Weight Update

Gradient Calculation with respect to W2

$$\frac{\partial L}{\partial W_2} = (\hat{y} - y)h^T$$

$$\frac{\partial L}{\partial b_2} = \hat{y} - y$$

where y is the target output, $\hat{y}$ is the predicted output, and h is the output of the hidden layer.

The derivation is same as solved in Logistic Regression with inputs as hidden layers outputs

Gradient Calculation with respect to W1

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial W_1}$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial b_1}$$

$$\frac{\partial L}{\partial h} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial h}$$

$$\frac{\partial \hat{y}}{\partial z} = W_2^T \frac{\partial \sigma(z)}{\partial z} = W_2^T \sigma(z) \odot (1 - \sigma(z))$$

$$\frac{\partial z}{\partial W_1} = x^T$$

$$h = \sigma(W_1 x + b_1)$$

where σ is the sigmoid activation function and $z = W_1 x + b_1$
$\odot$ indicates dot product

Based one Above equation*s

$$\frac{\partial L}{\partial W_2} = (\hat{y} - y)(\sigma(W_1 x + b_1))^T$$

$$\frac{\partial L}{\partial b_2} = \hat{y} - y$$

$$\frac{\partial L}{\partial W_1} = ((\hat{y} - y)W_2^T \odot (\sigma(W_1 x + b_1) \odot (1 - \sigma(W_1 x + b_1))))x^T$$

$$\frac{\partial L}{\partial b_1} = ((\hat{y} - y)W_2^T \odot (\sigma(W_1 x + b_1) \odot (1 - \sigma(W_1 x + b_1))))^T$$

Update the weights and biases using the gradients and a learning rate α:

$$W_2 \leftarrow W_2 - \alpha \frac{\partial L}{\partial W_2}$$

$$b_2 \leftarrow b_2 - \alpha \frac{\partial L}{\partial b_2}$$

$$W_1 \leftarrow W_1 - \alpha \frac{\partial L}{\partial W_1}$$

$$b_1 \leftarrow b_1 - \alpha \frac{\partial L}{\partial b_1}$$

3

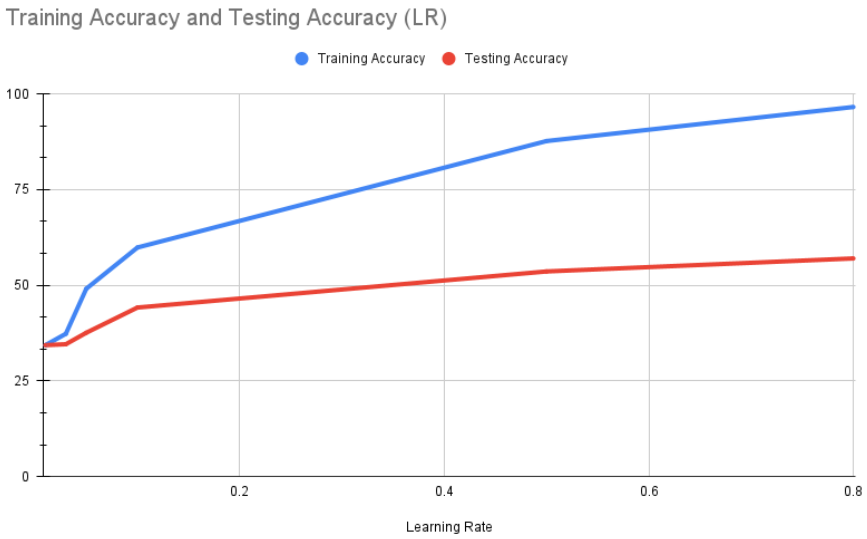
Hyper-Parameter Optimization for Logistic Regression

Hyper-parametrs for Logistic Regression.

1. Learning Rate
2. Epochs
3. L2 norm (lambda)

I have performed 5 fold cross-validation splitting the data into 80 20 randomly and tuned all the three hyper parameters.

Example plot showing training and testing accuracy while adjusting the learning rate, here as we are increasing the learning rate from a smaller number to nearly 0.8 the model start giving the best accuracries for both training and testing data-sets. At a learning rate of 0.8 the model gives a accuracy of 96.8 and testing accuracy of 58 percent since the testing accuracy is also increasing along with training accuracy we can infer that model is not over fitting for these parameters.



Hyper-parametrs for Neural Network

1. Learning Rate
2. Epochs

I have performed 5 fold cross-validation splitting the data into 80 20 randomly and tuned all the two hyper parameters.

Example plot showing training and testing accuracy while adjusting the epochs, here as we are increasing the number of epochs from a smaller number to nearly 2000 epochs the model start giving the best accuracries for both training and testing data-sets. For 2000 epochs model yields a training accuracy of 97 percent and a testing accuracy of 63 percent, since the testing accuracy is also increasing along with training accuracy we can infer that model is not over fitting for these parameters.