1 Abstract

The Kaggle dataset for binary classification of cats and dogs is taken and a simple neural network is used, which is programmed from scratch using **numpy** package in **Python** programming language.

2 Neural Network Architecture

The neural network used is for binary classification using **Binary Cross-Entropy** loss/cost function and **Sigmoid** activation function for output layer and **tanh** activation function for the hidden layer.

The neural network has:

1. Input layer has 12288 neurons (since 64 x 64 x 3 = 12288 matching the flattened features of training set)
2. Hidden layer has user defined number of neurons passed to the program while training
3. Output layer has a single neuron (since, this is for a binary classification task)

3 Backpropagation partial derivatives

The following equations are pertaining to backpropagation algorithm for adjusting the parameters of a neural network viz., it's weights and biases

The partial derivative of cost (J) with respect to weight vector 'W_2' (weights for the output layer) is:
$\frac{\partial J}{\partial W_2} = \frac{1}{m} * (a^2 - Y) * a_1$

The partial derivative of cost (J) with respect to weight vector 'W_1' (weights for the hidden layer) is:
$\frac{\partial J}{\partial W_1} = \frac{1}{m} * (a^2 - Y) * (1 - (a^1)^2) * X$

The partial derivative of cost (J) with respect to 'b_2' (bias for output layer) is:
$\frac{\partial J}{\partial b_2} = \frac{1}{m} * \sum_{i=1}^{m}(a_i^2 - Y_i)$

The partial derivative of cost (J) with respect to 'b_1' (biases for hidden layer) is:
$\frac{\partial J}{\partial b_1} = \frac{1}{m} * (a^2 - Y) * W_2 * (1 - (a^1)^2)$

4 Partial Derivatives and Mathematics

This section pertains to the different mathematical formulas and derivations which are used and subsequently coded in the program for the binary classification using the neural network described above.

1. Partial derivatives of weights for output layer:
   $\frac{\partial J}{\partial W_2} = \frac{\partial J}{\partial a_2} * \frac{\partial a_2}{\partial z_2} * \frac{\partial z_2}{\partial W_2}$ (using chain rule)
2. Partial derivatives of weights for hidden layer:
   $\frac{\partial J}{\partial W_1} = (\frac{\partial J}{\partial a_2} * \frac{\partial a_2}{\partial z_2} * \frac{z_2}{a_1}) * \frac{\partial a_1}{\partial z_1} * \frac{z_1}{W_1}$ (using chain rule)
3. Partial derivatives of bias for output layer:
   $\frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial a_2} * \frac{\partial a_2}{\partial z_2} * \frac{\partial z_2}{\partial b_2}$ (using chain rule)
4. Partial derivatives of biases for hidden layer:
   $\frac{\partial J}{\partial b_1} = \frac{\partial J}{\partial a_1} * \frac{\partial a_1}{\partial z_1} * \frac{\partial z_1}{\partial b_1}$ (using chain rule)
5. Partial derivative of Cost/loss with respect to output 'a_2' (binary cross-entropy cost function is used):
   $\frac{\partial J}{\partial a_2} = -\frac{1}{m} * \frac{a^2 - Y}{a^2 * (1 - a^2)}$
6. Partial derivative of output 'a_2' with respect to network input 'z_2' (output layer):
   $\frac{\partial a_2}{\partial z_2} = a^2 * (1 - a^2)$
7. Partial derivative of network input 'z_2' with respect to 'W_2' (output layer):
   $\frac{\partial z_2}{\partial W_2} = a_1$
8. Partial derivative of network input 'z_2' (output layer) with respect to 'a_1' (hidden layer):
   $\frac{\partial z_2}{\partial a_1} = W_2$
9. Partial derivative of output 'a_1' (hidden layer) with respect to 'a_1' (hidden layer):
   $\frac{a_1}{\partial z_1} = (1 - \tanh^2 x)$

   OR

   $\frac{\partial a_1}{\partial z_1} = (1 - (a_1)^2)$
10. Partial derivative of 'z_1' (hidden layer) with respect to 'W_1' (hidden layer):
    $\frac{\partial z_1}{\partial W_1} = X$