Deep Neural Network for Image Classification: Application

1 Introduction

In this report, we present a method for classifying images using a deep L-layer neural network. Image classification is a fundamental task in computer vision, where the goal is to categorize images into predefined classes based on their visual content. Our focus will be on developing a binary classifier to distinguish between images of cats and non-cats.

2 Steps Involved

2.1 Initialization of Parameters

Parameters for the neural network, specifically weights (W) and biases (b), are initialized before training begins. There are two common methods for initialization:

- **Random Initialization**: Weights are initialized to small random values, often drawn from a normal distribution multiplied by a small scaling factor. This prevents symmetry in the network which could impede learning.
- **Zero Initialization**: Biases are initialized to zero. Initializing weights to zero is not recommended as it can lead to poor convergence.

Mathematically, for each layer l:

$$W^{[l]} = \text{random values from a normal distributin} \times 0.01$$

$$b^{[l]} = \mathbf{0}$$

2.2 Forward Propagation

Forward propagation is the process of computing the output of the neural network given an input. This involves computing the linear and activation components for each layer.

For each layer l:

2.2.1 Linear Step

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

where $A^{[l-1]}$ is the activation from the previous layer (or input data for the first layer).

2.2.2 Activation Step

- **ReLU Activation**:

$$A^{[l]} = \text{ReLU}(Z^{[l]}) = \max(0, Z^{[l]})$$

- **Sigmoid Activation**:

$$A^{[L]} = \sigma(Z^{[L]}) = \frac{1}{1 + e^{-Z^{[L]}}}$$

2.2.3 Caching

Cache $Z^{[l]}$ and $A^{[l]}$ for use in backpropagation.

2.3 Compute Cost

The cost function quantifies the difference between the predicted output and the actual output. For binary classification, the cost function used is binary cross-entropy.

$$J = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(a^{[L](i)}) + (1 - y^{(i)}) \log(1 - a^{[L](i)}) \right]$$

where:

- m is the number of examples
- $y^{(i)}$ is the true label of the i-th example
- $a^{[L](i)}$ is the predicted probability for the i-th example

2.4 Backward Propagation

Backward propagation computes the gradient of the cost function with respect to the parameters of the network. These gradients are then used to update the parameters.

For each layer l, starting from the output layer and moving backwards:

2.4.1 Output Layer

Gradient of the cost with respect to $A^{[L]}$:

$$dA^{[L]} = -\left(\frac{Y}{A^{[L]}} - \frac{1 - Y}{1 - A^{[L]}} \right)$$

2.4.2 Gradients for Activation

- For Sigmoid:

$$dZ^{[L]} = dA^{[L]} \times \sigma'(Z^{[L]}) = dA^{[L]} \times A^{[L]} \times (1 - A^{[L]})$$

- For ReLU:

$$dZ^{[l]} = dA^{[l]} \times \mathbf{1}_{Z^{[l]}>0}$$

2.4.3 Gradients for Linear Step

$$dW^{[l]} = \frac{1}{m} dZ^{[l]} A^{[l-1]T}$$

$$db^{[l]} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[l](i)}$$

$$dA^{[l-1]} = W^{[l]T} dZ^{[l]}$$

2.5 Update Parameters

The parameters are updated using the gradients computed during backpropagation. The update rule is based on gradient descent.

$$W^{[l]} = W^{[l]} - \alpha dW^{[l]}$$

$$b^{[l]} = b^{[l]} - \alpha db^{[l]}$$

where α is the learning rate.

2.6 Training the Model

Training involves iteratively performing forward propagation, computing the cost, performing backward propagation, and updating the parameters. This process continues for a specified number of iterations or until the cost converges.

3 Conclusion

By systematically initializing parameters, performing forward and backward propagation, and updating parameters, a deep L-layer neural network can be trained to classify images with high accuracy. This process leverages the power of deep learning to capture complex patterns in the data, thereby improving classification performance.