Problem Set 3: Neural Networks

Abhishek Velusamy

November 24, 2024

1 Introduction

This report explores key concepts in neural networks, focusing on activation functions, forward propagation, and regularization. By implementing and analyzing these components, we aim to understand their impact on model performance.

2 Task 1: Neural Network Architecture

Neural networks consist of three main components:

- **Input Layer:** Receives input features.
- **Hidden Layers:** Transforms inputs using learned weights and activation functions.
- **Output Layer:** Produces final predictions (e.g., classification or regression results).

The architecture implemented includes:

- Three layers: Input layer, two hidden layers, and an output layer.
- Initialization of weights and biases using small random values.

For example, initializing parameters for a simple network:

$$W^{[1]} \in R^{4\times 3}, \quad b^{[1]} = \mathbf{0}, \quad W^{[2]} \in R^{4\times 4}, \quad b^{[2]} = \mathbf{0}$$

This structure allows the network to learn complex patterns in data.

3 Task 2: Activation Functions

Activation functions introduce non-linearity, enabling networks to learn complex relationships. Below are the commonly used functions:

- **Sigmoid:** $\sigma(x) = \frac{1}{1+e^{-x}}$
- **ReLU:** $f(x) = \max(0, x)$
- **Leaky ReLU:** $f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$

The following graph compares these activation functions and highlights their behavior over a range of input values:

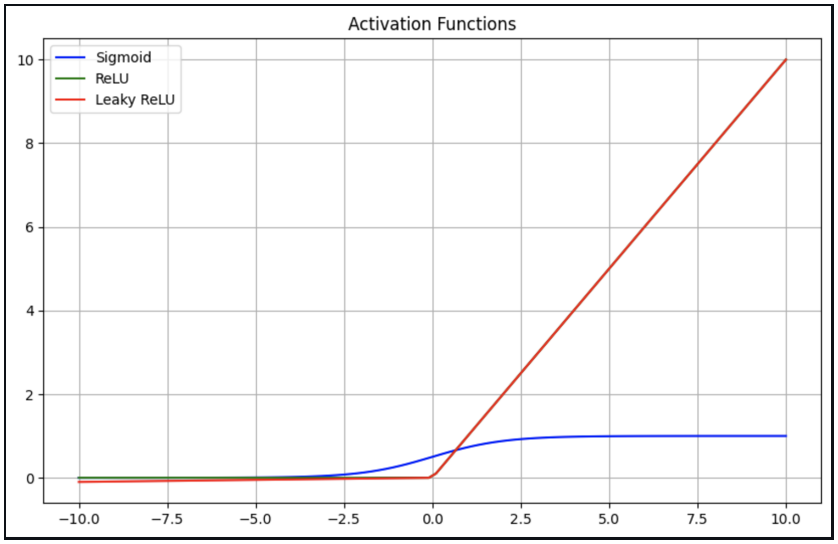

Figure 1: Comparison of Sigmoid, ReLU, and Leaky ReLU Activation Functions

4 Task 3: Forward Propagation

Forward propagation computes the outputs for a network layer-by-layer. For a 3-layer neural network:

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}, \quad A^{[l]} = f(Z^{[l]})$$

where f is the activation function. This process allows the network to compute predictions from inputs.

5 Task 4: Regularization

Regularization helps control overfitting, improving model generalization. L2 regularization adds a penalty to the loss:

$$J(\theta) = J_{\text{original}}(\theta) + \frac{\lambda}{2m} \sum_{i=1}^{n} \theta_i^2$$

This technique penalizes large weight values, encouraging simpler models that generalize better to unseen data.

6 Task 5: Training the Neural Network

Training adjusts weights and biases to minimize the loss function using techniques like gradient descent. The weight update rule is:

$$w := w - \alpha \frac{\partial J(w)}{\partial w}$$

where α is the learning rate, and $\frac{\partial J(w)}{\partial w}$ is the gradient of the loss function with respect to w.

7 Task 6: Adaptive Learning Rates

The Adam optimizer computes adaptive learning rates for each parameter using:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_t = \theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

This method stabilizes training by adjusting the step size based on past gradients.

8 Conclusion

Through this problem set, we implemented and analyzed neural network components, gaining insights into activation functions, forward propagation, and regularization. These concepts are foundational for improving model accuracy and robustness.