Exploding Gradient Problem in Deep Neural Networks

Introduction

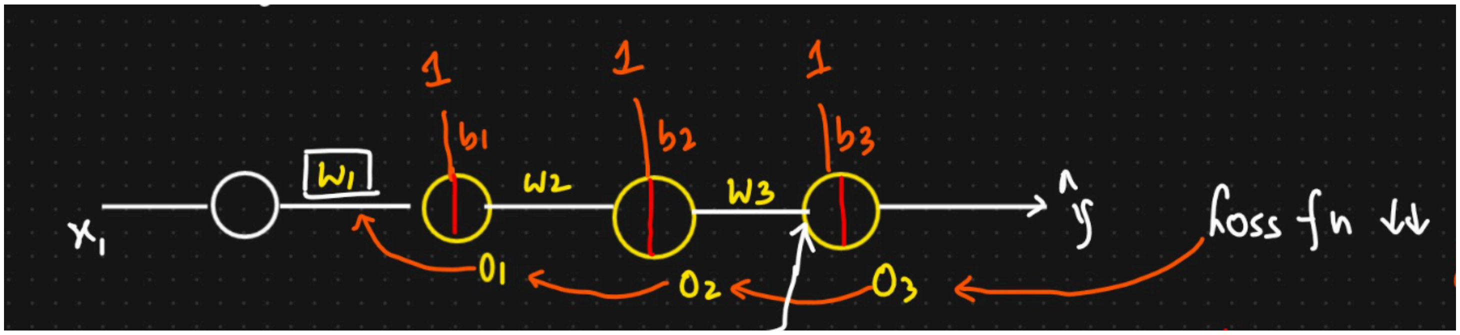
The exploding gradient problem is a critical issue in training deep neural networks that occurs when gradients become exponentially large during backpropagation. This problem can prevent the network from converging to an optimal solution and is closely related to weight initialization techniques.

Neural Network Architecture

Consider a simple deep neural network with:

- Input layer
- Hidden layer 1 (with one neuron)
- Hidden layer 2 (with one neuron)
- Output layer

The network processes information through forward propagation using weights w_1, w_2, w_3 and biases b_1, b_2, b_3.



Forward Propagation

The forward pass through the network can be described as:

Hidden Layer 1:

$$o_1 = \text{ReLU}(w_1 \cdot x + b_1)$$

where:

- x = input value
- w_1 = weight connecting input to hidden layer 1
- b_1 = bias term for hidden layer 1
- o_1 = output of hidden layer 1

Hidden Layer 2:

$$o_2 = \mathrm{ReLU}(w_2 \cdot o_1 + b_2)$$

where:

- w_2 = weight connecting hidden layer 1 to hidden layer 2
- b_2 = bias term for hidden layer 2
- o_2 = output of hidden layer 2

Output Layer (Binary Classification):

$$o_3 = \hat{y} = \mathrm{sigmoid}(w_3 \cdot o_2 + b_3)$$

where:

- w_3 = weight connecting hidden layer 2 to output
- b_3 = bias term for output layer
- $\hat{y}$ = predicted output

Weight Update Rule

During training, weights are updated using gradient descent:

$$w_1^{\text{new}} = w_1^{\text{old}} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial w_1^{\text{old}}}$$

where:

- w_1^{new} = updated weight value
- w_1^{old} = current weight value
- α = learning rate (step size)
- $\frac{\partial \mathcal{L}}{\partial w_1^{\text{old}}}$ = gradient of loss with respect to the weight

Chain Rule Application

To compute the gradient $\frac{\partial \mathcal{L}}{\partial w_1^{\text{old}}}$, we apply the chain rule:

$$\frac{\partial \mathcal{L}}{\partial w_1^{\text{old}}} = \frac{\partial \mathcal{L}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_1^{\text{old}}}$$

where each term represents:

- $\frac{\partial \mathcal{L}}{\partial o_3}$ = gradient of loss with respect to final output
- $\frac{\partial o_3}{\partial o_2}$ = gradient of output layer with respect to hidden layer 2
- $\frac{\partial o_2}{\partial o_1}$ = gradient of hidden layer 2 with respect to hidden layer 1
- $\frac{\partial o_1}{\partial w_1^{\text{old}}}$ = gradient of hidden layer 1 with respect to weight w_1

Detailed Gradient Computation

Step 1: Output Layer Gradient

For the output layer with sigmoid activation:

$$\frac{\partial o_3}{\partial o_2} = \frac{\partial}{\partial o_2} \text{sigmoid}(z)$$

where $z = w_3 \cdot o_2 + b_3$

Using chain rule:

$$\frac{\partial o_3}{\partial o_2} = \frac{\partial \text{sigmoid}(z)}{\partial z} \cdot \frac{\partial z}{\partial o_2}$$

Sigmoid Derivative:

$$\frac{\partial \text{sigmoid}(z)}{\partial z} = \text{sigmoid}(z) \cdot (1 - \text{sigmoid}(z))$$

This value ranges between 0 and 0.25 (maximum at $z = 0$).

Linear Component:

$$\frac{\partial z}{\partial o_2} = \frac{\partial}{\partial o_2}(w_3 \cdot o_2 + b_3) = w_3$$

Therefore:

$$\frac{\partial o_3}{\partial o_2} = \text{sigmoid}(z) \cdot (1 - \text{sigmoid}(z)) \cdot w_3$$

The Exploding Gradient Problem

Problem Mechanism

When weights are initialized with very large values (e.g., $w_3 = 500$ or $w_3 = 1000$), the gradient computation becomes:

$$\frac{\partial \mathcal{L}}{\partial w_1^{\text{old}}} = \underbrace{\frac{\partial \mathcal{L}}{\partial o_3}}_{\text{reasonable}} \cdot \underbrace{[0, 0.25] \times \text{LARGE}}_{\text{can be large}} \cdot \underbrace{\frac{\partial o_2}{\partial o_1}}_{\text{depends on } w_2} \cdot \underbrace{\frac{\partial o_1}{\partial w_1^{\text{old}}}}_{\text{depends on input}}$$

Cascading Effect

If multiple weights are initialized with large values:

- Each gradient term becomes large
- The chain rule multiplies these large terms together
- The final gradient becomes exponentially large

Weight Update Impact

With large gradients, the weight update becomes:

$$w_1^{\text{new}} = w_1^{\text{old}} - \alpha \cdot \text{VERY LARGE NUMBER}$$

This can result in:

1. $w_1^{\text{new}} \gg w_1^{\text{old}}$ (weights increase dramatically)
2. $w_1^{\text{new}} \ll w_1^{\text{old}}$ (weights decrease dramatically)

Consequences on Training

Optimization Landscape

In the gradient descent optimization process:

- **Goal**: Reach global minimum of loss function $\mathcal{L}(w)$

- **Problem**: Large weight updates cause oscillations around the minimum
- **Result**: Training becomes unstable and may diverge

Mathematical Representation

If the gradient magnitude is too large:

$$\left| \frac{\partial \mathcal{L}}{\partial w} \right| \gg \text{optimal range}$$

Then the weight update step:

$$\Delta w = -\alpha \cdot \frac{\partial \mathcal{L}}{\partial w}$$

becomes too large, causing the optimizer to "overshoot" the minimum.

Root Cause: Weight Initialization

The exploding gradient problem primarily stems from poor weight initialization:

Problematic Initialization

$$w_i \sim \mathcal{U}(\text{large negative}, \text{large positive})$$

where weights are sampled from a uniform distribution with large bounds.

Impact on Gradient Flow

With large initial weights, the gradient magnitude grows as:

$$\left| \frac{\partial \mathcal{L}}{\partial w_1} \right| \propto \prod_{i=2}^{L} |w_i|$$

where L is the number of layers.

Solution Preview

The solution involves using proper weight initialization techniques that:

1. **Control gradient magnitude**: Keep gradients in a reasonable range

2. **Maintain signal flow**: Prevent both exploding and vanishing gradients
3. **Enable stable training**: Allow convergence to optimal solutions

Common techniques include:

- Xavier/Glorot initialization
- He initialization
- LeCun initialization

Conclusion

The exploding gradient problem is a fundamental challenge in deep learning that occurs when:

- Weights are initialized with large values
- Gradients become exponentially large during backpropagation
- Weight updates become unstable, preventing convergence

Understanding this problem is crucial for training stable and effective deep neural networks. The solution lies in proper weight initialization techniques that maintain gradient magnitudes within reasonable bounds throughout the network depth.