Weight Initialization Techniques in Neural Networks

Introduction

Weight initialization is a crucial step in training neural networks effectively. Proper initialization helps prevent common training problems and ensures stable gradient flow throughout the network.

Key Principles of Weight Initialization

Before diving into specific techniques, it's essential to understand the fundamental principles:

1. **Weights should be small**: Large initial weights can lead to exploding gradient problems
2. **Weights should not be identical**: If all weights are the same, neurons will perform identical computations, reducing the network's learning capacity
3. **Good variance is essential**: Weights should have appropriate variance to maintain signal propagation through layers

Neural Network Notation

Consider a neural network where:

- **Input layer**: Contains n_{input} neurons
- **Hidden layers**: Intermediate processing layers
- **Output layer**: Contains n_{output} neurons

Weights are denoted as w_{ij} where:

- i represents the input layer index
- j represents the hidden layer index

1.Uniform Distribution Initialization

Method

All weights w_{ij} are initialized from a uniform distribution:

$$w_{ij} \sim \mathcal{U}(a, b)$$

where the range parameters are:

$$a = -\frac{1}{\sqrt{n_{input}}}, \quad b = \frac{1}{\sqrt{n_{input}}}$$

Formula Components

- $\mathcal{U}(a, b)$: Uniform distribution between values a and b
- n_{input}: Number of input neurons
- $\sqrt{n_{input}}$: Square root of input neurons for scaling

Example

For a network with 3 input neurons:

$$w_{ij} \sim \mathcal{U}\left(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$$

2.Xavier (Glorot) Initialization

Xavier initialization, developed by Xavier Glorot, addresses the vanishing/exploding gradient problem by maintaining the variance of activations and gradients across layers.

2.1 Xavier Normal Initialization

Weights are initialized from a normal distribution:

$w_{ij} \sim \mathcal{N}(0, \sigma)$

where the standard deviation is:

$\sigma = \sqrt{\frac{2}{n_{input} + n_{output}}}$

Formula Components

- $\mathcal{N}(0, \sigma)$: Normal distribution with mean 0 and standard deviation σ
- n_{input}: Number of input neurons
- n_{output}: Number of output neurons
- The factor of 2 in the numerator helps maintain proper variance scaling

Example

For a layer with 3 inputs and 3 outputs:

$$\sigma = \sqrt{\frac{2}{3+3}} = \sqrt{\frac{2}{6}} = \sqrt{\frac{1}{3}}$$

2.2 Xavier Uniform Initialization

Weights are initialized from a uniform distribution:

$$w_{ij} \sim \mathcal{U}(a, b)$$

where the range parameters are:

$$a = -\sqrt{\frac{6}{n_{input} + n_{output}}}, \quad b = \sqrt{\frac{6}{n_{input} + n_{output}}}$$

Formula Components

- The factor of 6 maintains the same variance as the normal version
- $\sqrt{\frac{6}{n_{input}+n_{output}}}$: Scaling factor based on fan-in and fan-out

Example

For a layer with 3 inputs and 3 outputs:

$$w_{ij} \sim \mathcal{U}\left(-\sqrt{\frac{6}{6}}, \sqrt{\frac{6}{6}}\right) = \mathcal{U}(-1, 1)$$

3.He Initialization

He initialization, developed by Kaiming He, is particularly effective for networks using ReLU activation functions.

3.1 He Normal Initialization

Weights are initialized from a normal distribution:

$w_{ij} \sim \mathcal{N}(0, \sigma)$

where the standard deviation is:

$\sigma = \sqrt{\frac{2}{n_{input}}}$

Formula Components

- $\mathcal{N}(0, \sigma)$: Normal distribution with mean 0 and standard deviation σ
- n_{input}: Number of input neurons (fan-in)
- The factor of 2 accounts for the ReLU activation function's properties

Example

For a layer with 3 input neurons:

$$\sigma = \sqrt{\frac{2}{3}}$$

3.2 He Uniform Initialization

Weights are initialized from a uniform distribution:

$$w_{ij} \sim \mathcal{U}(a, b)$$

where the range parameters are:

$$a = -\sqrt{\frac{6}{n_{input}}}, \quad b = \sqrt{\frac{6}{n_{input}}}$$

Formula Components

- The factor of 6 maintains equivalent variance to the normal version
- $\sqrt{\frac{6}{n_{input}}}$: Scaling factor based on fan-in only

Example

For a layer with 3 input neurons:

$$w_{ij} \sim \mathcal{U}\left(-\sqrt{\frac{6}{3}}, \sqrt{\frac{6}{3}}\right) = \mathcal{U}(-\sqrt{2}, \sqrt{2})$$

Summary of Initialization Techniques

Method	Distribution Type	Parameters
Uniform	Uniform	$\mathcal{U}\left(-\frac{1}{\sqrt{n_{input}}}, \frac{1}{\sqrt{n_{input}}}\right)$
Xavier Normal	Normal	$\mathcal{N}\left(0, \sigma\right)$ where $\sigma = \sqrt{\frac{2}{n_{input}+n_{output}}}$
Xavier Uniform	Uniform	$\mathcal{U}\left(-\sqrt{\frac{6}{n_{input}+n_{output}}}, \sqrt{\frac{6}{n_{input}+n_{output}}}\right)$
He Normal	Normal	$\mathcal{N}\left(0, \sigma\right)$ where $\sigma = \sqrt{\frac{2}{n_{input}}}$
He Uniform	Uniform	$\mathcal{U}\left(-\sqrt{\frac{6}{n_{input}}}, \sqrt{\frac{6}{n_{input}}}\right)$

When to Use Each Method

- **Xavier Initialization**: Best for sigmoid and tanh activation functions
- **He Initialization**: Optimal for ReLU and its variants
- **Uniform Distribution**: Simple baseline method, less commonly used in practice

Implementation Notes

Most deep learning frameworks (TensorFlow, PyTorch, etc.) provide built-in implementations of these initialization methods, so manual calculation is rarely necessary. The choice of initialization can significantly impact training stability and convergence speed.