Weight Initialization and Training Stability in Multi-Layer Perceptrons

1 Introduction

Training deep Multi-Layer Perceptrons (MLPs) is fundamentally a problem of numerical stability. Even with correct model architecture and optimization algorithms, poor weight initialization or unsuitable activation functions can prevent learning altogether. Issues such as vanishing gradients, exploding gradients, and dead neurons often arise before optimization has a chance to succeed.

This report examines the role of weight initialization, activation function choice, and gradient control techniques in ensuring stable and efficient training of MLPs.

2 The Role of Weight Initialization

At initialization, weights determine the scale of activations and gradients throughout the network. If weights are too small, signals shrink as they propagate; if too large, signals grow uncontrollably.

For a neuron:

$$z = \sum_{i=1}^{n} w_i x_i$$

Assuming x_i and w_i are independent and zero-mean, the variance of z is:

$$\mathrm{Var}(z) = n\,\mathrm{Var}(w)\,\mathrm{Var}(x)$$

To maintain stable signal propagation across layers, the variance of activations should remain approximately constant.

3 Xavier (Glorot) Initialization

Xavier initialization was designed for activation functions such as sigmoid and tanh. It chooses weights such that:

$$\mathrm{Var}(w) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$

Common implementations sample:

$$w \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}\right)$$

This initialization balances forward and backward signal flow, reducing early saturation in sigmoidal activations.

4 He Initialization

For ReLU-based networks, Xavier initialization underestimates the required variance because ReLU zeroes out half of its inputs. He initialization compensates for this effect:

$$\mathrm{Var}(w) = \frac{2}{n_{\text{in}}}$$

Weights are typically sampled as:

$$w \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right)$$

He initialization significantly improves convergence speed and stability in deep ReLU networks.

5 Choice of Activation Functions

5.1 Sigmoid and Tanh

Sigmoid and tanh activations saturate for large magnitude inputs:

$$\sigma'(x) \approx 0 \quad \text{for large } |x|$$

This leads to vanishing gradients and slow learning in deep networks. While tanh is zero-centered and preferable to sigmoid, both are largely unsuitable for deep MLPs.

5.2 ReLU

The Rectified Linear Unit is defined as:

$$\mathrm{ReLU}(x) = \max(0, x)$$

ReLU avoids saturation for positive inputs and enables efficient gradient propagation. Its simplicity and effectiveness have made it the default activation for hidden layers.

5.3 Dying ReLU Problem

If a neuron's weights cause its input to remain negative, ReLU outputs zero and its gradient becomes zero permanently:

$$\frac{d}{dx}\mathrm{ReLU}(x) = 0 \quad \text{for } x < 0$$

This issue is exacerbated by poor initialization or large learning rates.

5.4 ReLU Variants

Leaky ReLU introduces a small slope in the negative region:

$$\text{Leaky ReLU}(x) = \max(\alpha x, x)$$

which mitigates neuron death while preserving ReLU's advantages.

6 Softmax and Output Stability

For classification tasks, the softmax function converts logits into probabilities:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

To prevent numerical overflow, softmax is computed using a stabilized form:

$$\text{softmax}(z_i) = \frac{e^{z_i - \max_j z_j}}{\sum_k e^{z_k - \max_j z_j}}$$

Softmax is typically paired with cross-entropy loss, yielding well-conditioned gradients.

7 Exploding and Vanishing Gradients

Repeated multiplication of weight matrices during backpropagation can cause gradients to shrink or grow exponentially:

$$\prod_{l=1}^{L} \mathbf{W}^{(l)}$$

- **Vanishing gradients**: Early layers learn slowly.
- **Exploding gradients**: Training becomes unstable.

Proper initialization and activation choice significantly reduce these effects.

8 Gradient Clipping

Gradient clipping constrains the magnitude of gradients:

$$\mathbf{g} \leftarrow \mathbf{g} \cdot \min\left(1, \frac{\tau}{\|\mathbf{g}\|}\right)$$

This technique is particularly effective when gradients occasionally spike, preventing catastrophic parameter updates without altering gradient direction.

9 Interplay Between Initialization and Optimization

Initialization determines the starting geometry of the loss surface. Adaptive optimizers can compensate for poor scaling to some extent, but cannot fully overcome pathological initial conditions. Stable training requires a coherent choice of:

- Initialization scheme
- Activation function
- Learning rate
- Gradient control mechanisms

10 Conclusion

Weight initialization and training stability are foundational to the success of deep MLPs. Techniques such as Xavier and He initialization, combined with appropriate activation functions and gradient clipping, enable reliable gradient flow across many layers. Without these considerations, optimization algorithms may fail regardless of their theoretical guarantees.