Backpropagation and Gradient-Based Optimization in Multi-Layer Perceptrons

1 Introduction

Multi-Layer Perceptrons (MLPs) are one of the most fundamental architectures in neural networks. Despite their apparent simplicity, their training relies on a carefully structured mathematical framework involving calculus, linear algebra, and numerical optimization. At the core of this framework lie *backpropagation* and *gradient-based optimization*, which together enable the network to learn from data.

This report presents a detailed and mathematically grounded discussion of backpropagation, gradient descent and its variants, loss functions, activation functions, and common training pathologies such as vanishing and exploding gradients.

2 Structure of a Multi-Layer Perceptron

An MLP consists of a sequence of layers indexed by $l = 1, 2, \dots, L$. Each layer performs an affine transformation followed by a nonlinear activation:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \quad \mathbf{a}^{(l)} = \sigma^{(l)}(\mathbf{z}^{(l)})$$

where:

- $\mathbf{W}^{(l)}$ is the weight matrix,
- $\mathbf{b}^{(l)}$ is the bias vector,
- $\mathbf{a}^{(l)}$ is the activation output,
- $\sigma^{(l)}$ is the activation function.

The input layer is defined as $\mathbf{a}^{(0)} = \mathbf{x}$, and the final layer produces the network output $\hat{\mathbf{y}} = \mathbf{a}^{(L)}$.

3 Loss Functions

The loss function quantifies the discrepancy between the predicted output $\hat{\mathbf{y}}$ and the true target $\mathbf{y}$. Common choices include:

3.1 Mean Squared Error (MSE)

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|^2$$

Primarily used in regression tasks.

3.2 Cross-Entropy Loss

For classification with softmax outputs:

$$\mathcal{L}_{\text{CE}} = -\sum_k y_k \log(\hat{y}_k)$$

This loss arises naturally from maximum likelihood estimation under a categorical distribution.

4 Backpropagation

Backpropagation is an efficient application of the chain rule to compute gradients of the loss with respect to all parameters.

4.1 Error Signal

Define the error at layer l as:

$$\boldsymbol{\delta}^{(l)} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l)}}$$

For the output layer:

$$\boldsymbol{\delta}^{(L)} = \nabla_{\mathbf{a}^{(L)}} \mathcal{L} \odot \sigma'(\mathbf{z}^{(L)})$$

For hidden layers:

$$\boldsymbol{\delta}^{(l)} = \left(\mathbf{W}^{(l+1)}\right)^T \boldsymbol{\delta}^{(l+1)} \odot \sigma'(\mathbf{z}^{(l)})$$

4.2 Gradients of Parameters

Once $\boldsymbol{\delta}^{(l)}$ is known:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)} \left(\mathbf{a}^{(l-1)}\right)^T, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \boldsymbol{\delta}^{(l)}$$

This backward pass has computational complexity linear in the number of parameters.

5 Gradient Descent

Gradient descent updates parameters by moving in the direction of steepest descent:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}$$

where η is the learning rate.

5.1 Variants

- **Batch Gradient Descent**: Uses the full dataset.
- **Stochastic Gradient Descent (SGD)**: Uses a single sample.
- **Mini-batch Gradient Descent**: Compromise between the two.

6 Adaptive Optimization Algorithms

6.1 RMSProp

RMSProp normalizes gradients using an exponentially weighted average:

$$v_t = \rho v_{t-1} + (1-\rho)g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} g_t$$

6.2 Adam

Adam combines momentum and RMSProp:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$$

Bias-corrected estimates:

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

Update rule:

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

7 Activation Functions

7.1 Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Prone to vanishing gradients.

7.2 Tanh

Zero-centered but still saturates.

7.3 ReLU

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

Efficient and widely used, but can suffer from dead neurons.

7.4 Variants

Leaky ReLU, ELU, and GELU aim to mitigate ReLU's shortcomings.

8 Vanishing and Exploding Gradients

Repeated multiplication of Jacobians during backpropagation can lead to:

- **Vanishing gradients**: Gradients approach zero, slowing learning.
- **Exploding gradients**: Gradients grow uncontrollably, causing instability.

8.1 Mitigation Strategies

- Proper weight initialization (Xavier, He)
- Non-saturating activations
- Gradient clipping
- Batch normalization

9 Learning Rate Considerations

The learning rate controls convergence behavior:

- Too small: Slow convergence
- Too large: Divergence or oscillation

Learning rate schedules and adaptive optimizers are commonly employed to balance stability and speed.

10 Conclusion

Backpropagation and gradient-based optimization form the mathematical backbone of training MLPs. While the core ideas are conceptually simple, practical training requires careful consideration of optimization algorithms, activation functions, and numerical stability issues. Understanding these mechanisms at a mathematical level is essential for designing reliable and efficient neural networks.