Comprehensive Review: Backpropagation in Neural Networks

DSC 255 - Machine Learning Fundamentals

Contents

1 Overview

Backpropagation is the key algorithm used for training feedforward neural networks. It computes the gradients of a loss function with respect to each parameter in the network via the chain rule of calculus.

2 Learning as Optimization

Let W denote the set of all weights and biases in a neural net. Training is posed as minimizing a loss function $L(W)$ based on a labeled dataset $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$.

Loss Function for Classification

Assuming k possible class labels:

$$L(W) = -\sum_{i=1}^{n} \log \Pr_W(y^{(i)} \mid x^{(i)})$$

This is known as the **cross-entropy loss**. The network outputs a probability distribution over labels for each input.

3 Gradient-Based Optimization Methods

Gradient Descent Variants

- **Batch Gradient Descent**: Uses the full dataset for each update.
- **Stochastic Gradient Descent (SGD)**: Updates using one data point at a time.
- **Mini-batch SGD**: Updates using a small batch of points. Commonly used in practice.

4 Gradient Computation with Backpropagation

We want the derivative of the loss L with respect to every parameter in the network. The method proceeds in two passes:

1. **Forward Pass**: Compute outputs of all nodes layer by layer.
2. **Backward Pass**: Apply the chain rule recursively to compute gradients.

5 Chain Rule of Calculus

Single Variable Case

If $h(x) = g(f(x))$, then:

$$h'(x) = g'(f(x)) \cdot f'(x)$$

General Form

If $z = g(y)$ and $y = f(x)$, then:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

6 Example: Single Chain Neural Net

Assume a deep net where each hidden layer has only one node:

$$x = h_0, \quad h_1 = \sigma(w_1 h_0 + b_1), \quad h_2 = \sigma(w_2 h_1 + b_2), \quad \ldots, \quad h_\ell$$

Gradient with Respect to Weights

Using the chain rule:

$$\frac{dL}{dw_i} = \frac{dL}{dh_i} \cdot \sigma'(w_i h_{i-1} + b_i) \cdot h_{i-1}$$

Recursive Gradient Propagation

$$\frac{dL}{dh_i} = \frac{dL}{dh_{i+1}} \cdot \sigma'(w_{i+1} h_i + b_{i+1}) \cdot w_{i+1}$$

7 The Backpropagation Algorithm

- Perform a forward pass to compute all intermediate h_i and final output.
- Use the chain rule in a backward pass to compute all derivatives $\frac{dL}{dw_i}$.
- Update each parameter w_i using gradient descent.

8 Remarks on Non-Convexity

The loss surface of a neural net is highly non-convex:

- Contains many local optima.
- Final result depends heavily on initialization and randomization.

Despite non-convexity, neural nets often perform well due to overparameterization and high model flexibility.

9 Practical Implementation: Automatic Differentiation

Modern frameworks like PyTorch and TensorFlow handle backpropagation automatically. The user specifies:

- The architecture of the network
- The loss function

The framework computes gradients and applies updates internally.

10 Summary

- Neural network training minimizes a loss via gradient descent.
- Gradients are computed efficiently via backpropagation using the chain rule.
- Variants of SGD are widely used due to large dataset sizes.
- Despite non-convexity, training often converges to useful solutions.