Chapter 7: Derivatives

The Problem with Random Weight and Bias Adjustments

- Randomly adjusting weights and biases is ineffective due to the infinite number of combinations.
- Each weight and bias influences the loss differently, depending on the parameter values and the current input sample.
- Loss is calculated separately for each sample since each sample affects the neuron outputs and thus the loss differently.
- To optimize, we need a systematic method to understand and adjust the impact of weights and biases.

Understanding the Impact on Loss

- **Loss Function vs. Parameters**: The loss function itself doesn't contain weights and biases; instead, these parameters influence the model's output, which is the input to the loss function.
- **Gradient Descent Goal**: The goal is to minimize the loss by adjusting weights and biases intelligently. This adjustment is driven by understanding the impact of each parameter on the loss.

Numerical Differentiation

- **Derivatives** help measure how much a parameter affects the output.
- For a simple function y=2xy = 2x, the slope (derivative) is constant (2).
- For non-linear functions like y=2x2y = 2x^2, the slope changes at different points, requiring tangent lines to approximate the slope at specific points.
- **Numerical Differentiation**: By selecting two points very close to each other, we approximate the slope, which represents the instantaneous rate of change.

Approximation of the Derivative

- **Tangent Line Method**: The slope is measured using two very close points. The slope of a line is:

  - $$\frac{Change\ in\ y}{Change\ in\ x} = \frac{\Delta y}{\Delta x}$$

- Using a small delta (like 0.0001), we compute:

$$\text{Approximate Derivative} = \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

- This slope is the **instantaneous rate of change** for that point.
- For smooth curves, this method provides a good approximation.
- Tangent lines indicate the slope and how rapidly the function value changes with a change in x.

Limitations of Numerical Differentiation in Neural Networks

- Calculating derivatives for each weight and bias for every sample using this method is computationally expensive.
- Neural networks have complex, multi-dimensional loss functions, making brute-force differentiation impractical.
- This introduces the need for more efficient methods like **backpropagation** and **gradient descent**.

The Analytical Derivative

- The analytical approach to derivatives involves calculus and finding the exact derivative function.

Using **limits**, the derivative of $f(x)$ at $x = a$ is expressed as:

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

- The result is an exact formula representing the rate of change at any point on the curve.

The derivative of a constant equals 0 (m is a constant in this case, as it's not a parameter that we are deriving with respect to, which is *x* in this example):

$$\frac{d}{dx}1 = 0$$

$$\frac{d}{dx}m = 0$$

The derivative of *x* equals 1:

$$\frac{d}{dx}x = 1$$

The derivative of a linear function equals its slope:

$$\frac{d}{dx}mx + b = m$$

$$\frac{d}{dx}[k \cdot f(x)] = k \cdot \frac{d}{dx}f(x)$$

$$\frac{d}{dx}[f(x) + g(x)] = \frac{d}{dx}f(x) + \frac{d}{dx}g(x) = f'(x) + g'(x)$$

$$\frac{d}{dx}[f(x) - g(x)] = \frac{d}{dx}f(x) - \frac{d}{dx}g(x) = f'(x) - g'(x)$$

$$\frac{d}{dx}x^n = n \cdot x^{n-1}$$

The second derivative f''(x) describes the rate of change of the rate of change (concavity).

Higher-order derivatives provide more insights into the behavior of the function.