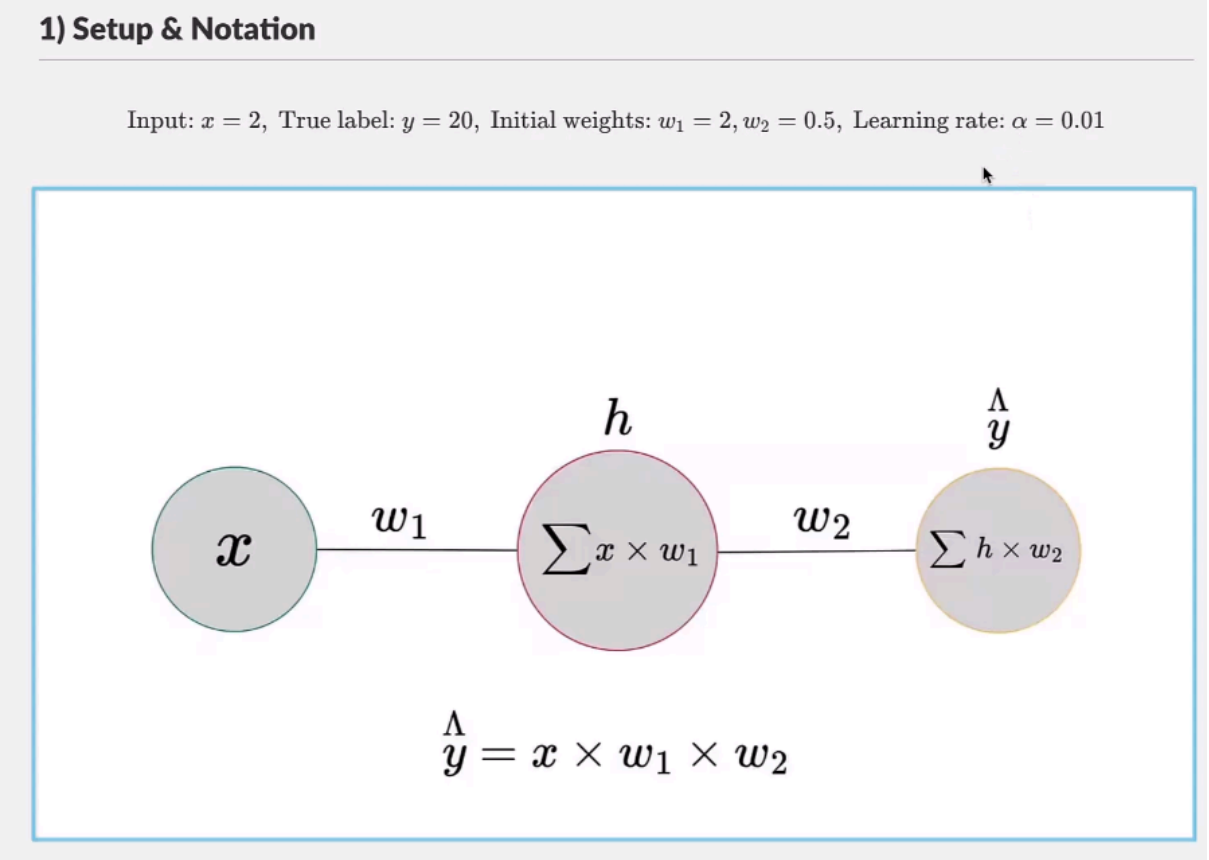
Build a Simple Neural Network and Learn Backpropagation

Introduction to our simple Neural Network



Forward Pass (Before Update)

2) Forward Pass (Before Update)

Remember this is the entire network

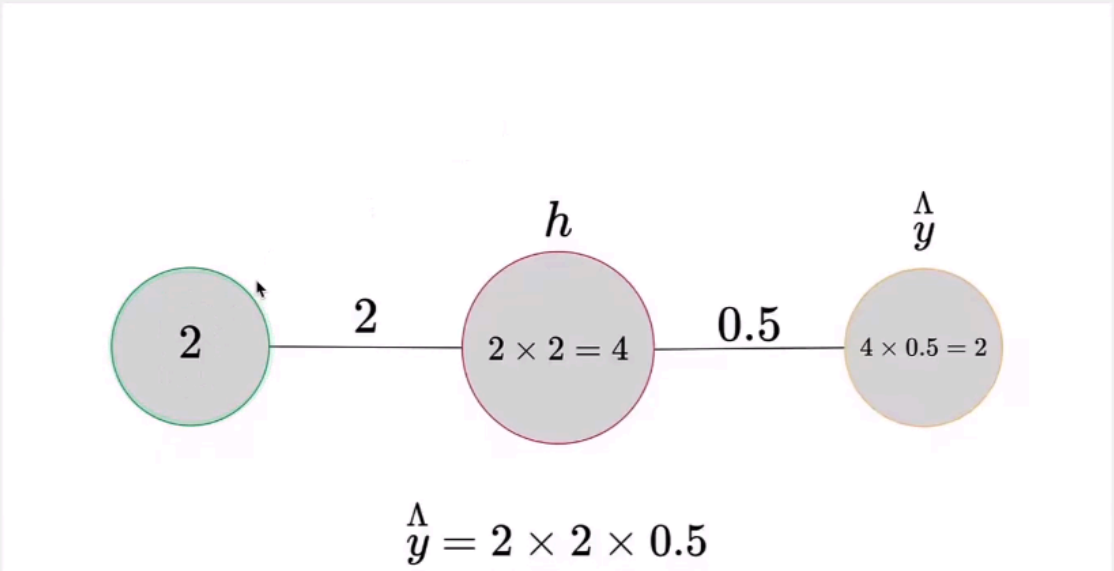
$$\hat{y} = x \times w_1 \times w_2,$$

1. **Hidden node output**:

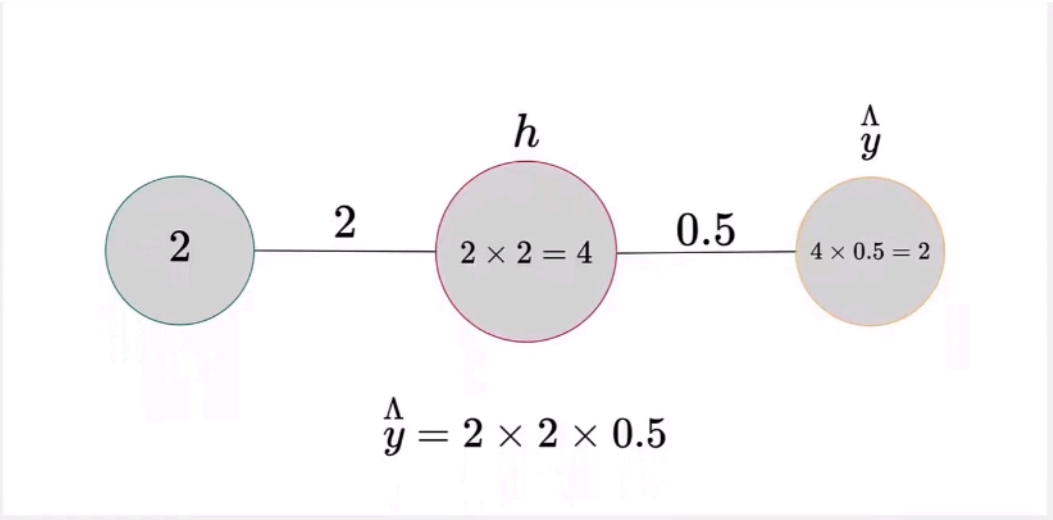
$$h = x \times w_1 = 2 \times 2 = 4.$$

2. **Predicted output**:

$$\hat{y} = h \times w_2 = 4 \times 0.5 = 2.$$



$$\hat{y} = 2 \times 2 \times 0.5$$



3. **Calculate Initial Loss**

$$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(20 - 2)^2 = \frac{1}{2}(18)^2 = \frac{1}{2} \times 324 = 162.$$

Initial loss is 162.

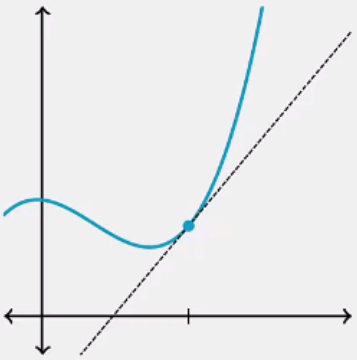
Derivates Theory

Understanding the Derivative, Partial Derivative, and Gradient

What is a Derivative?

A derivative measures how a function changes as its input changes. For example, it tells us the rate of change of one quantity with respect to another.

The **derivative** of a function at a given point represents the **slope of the tangent line** to the function's graph at that point.



Numerical Example of Derivates

Equation of the Curve:

Let the curve be defined as:

$$f(x) = x^3 - 3x^2 + 2x + 1.$$

Derivative of the Function:

To find the slope of the tangent line, compute the derivative (f'(x)):

$$f'(x) = \frac{d}{dx}(x^3 - 3x^2 + 2x + 1).$$

Using the power rule:

$$f'(x) = 3x^2 - 6x + 2.$$

Slope(Derivative) at a Specific Point:

Suppose we want to find the slope of the tangent at (x = 2). Substitute (x = 2) into (f'(x)):

$$f'(2) = 3(2)^2 - 6(2) + 2 = 2$$

The slope at x = 2 is 2

Small Change:

original function:

$$f(x) = x^3 - 3x^2 + 2x + 1.$$

let $x = 2$ and apply a small increase of $\Delta x = 0.1$. Calculate the change in $f(x)$:

The function at (x = 2) is:

$$f(2) = (2)^3 - 3(2)^2 + 2(2) + 1 = 1$$

The function at (x = 2.1) is:

$$f(2.1) = (2.1)^3 - 3(2.1)^2 + 2(2.1) + 1 = 1.231$$

The change is

$$\Delta f = f(2.1) - f(2) = 1.231 - 1 = 0.231$$

Let's see what the Derivative guesses(0.231 is the correct verified answer)

Remember we found the slope of the tangent line at (x = 2). We did that by plugging in 2 into the derivative:

$$f'(x) = 3x^2 - 6x + 2.$$

$$f'(2) = 3(2)^2 - 6(2) + 2 = 2$$

2 is the slope(rate of change, or multiplier) at x=2

The derivative at (x = 2) predicted that the change in (f(x)) output would be approximately:

$$\Delta f \approx f'(2) \cdot \Delta x = 2 \cdot 0.1 = 0.2$$

The actual change (0.231) is very close to the predicted change (0.2), confirming that the derivative gives a good approximation for small changes.

Partial Derivatives

Introduction to Partial Derivatives:

When dealing with functions of multiple variables, such as

$$f(x, y, z)$$

the rate of change of the function with respect to one variable while keeping others constant is called a partial derivative.

The notation for partial derivatives is similar to regular derivatives but uses a different symbol:

$$\frac{\partial}{\partial x}$$

For example:

1. If $f(x, y) = x^2 + y^3$, then $\frac{\partial f}{\partial x} = 2x$ (treating y as a constant), and $\frac{\partial f}{\partial y} = 3y^2$ (treating x as a constant)

Gradients

What is a Gradient?

The gradient is just a fancy term used when you take the derivative of a function with respect to all of its variables. In essence, it is a collection of all the partial derivatives of a function.

- For a function

$$f(x, y, z)$$

, the gradient vector is written as:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}$$

- The gradient points in the direction of the steepest increase of the function and is widely used in optimization problems, such as in machine learning.

For example If

$$f(x, y) = x^2 + y^3$$

then:

$$\nabla f = \begin{bmatrix} 2x \\ 3y^2 \end{bmatrix}$$

Understanding What Partial Derivatives Do

Example to show What Partial Derivatives do

- Remember: the rate of change of the function with respect to one variable while keeping others constant(frozen) is called a partial derivative. This is done to see how one variable affects the output of the function when all other variables are constant/frozen.

Step 1: Initial Output

We have the function:

$$f(x, y) = x^2 + y^2.$$

Suppose (x = 3) and (y = 4). The initial output of f(x, y) is:

$$f(3, 4) = 3^2 + 4^2 = 9 + 16 = 25.$$

Step 2: Small Changes in (x) and (y)

Let x increase by $\Delta x = 0.1$ and y increase by $\Delta y = 0.2$. The new values are:

$$x = 3 + 0.1 = 3.1, \quad y = 4 + 0.2 = 4.2.$$

Step 3: New Output

The new output of $f(x, y)$ is:

$$f(3.1, 4.2) = (3.1)^2 + (4.2)^2 = 9.61 + 17.64 = 27.25.$$

Step 4: Change in Output

The change in the output of (f(x, y)) is:

$$\Delta f = f(3.1, 4.2) - f(3, 4) = 27.25 - 25 = 2.25.$$

Step 5: Approximation Using Gradients

original function:

$$f(x, y) = x^2 + y^2.$$

x increases by $\Delta x = 0.1$ and y increases by $\Delta y = 0.2$

$$\Delta f \approx \frac{\partial f}{\partial x} \cdot \Delta x + \frac{\partial f}{\partial y} \cdot \Delta y,$$

where:

$$\frac{\partial f}{\partial x} = 2x, \quad \frac{\partial f}{\partial y} = 2y.$$

At (x = 3) and (y = 4), we calculate:

$$\frac{\partial f}{\partial x} = 2(3) = 6, \quad \frac{\partial f}{\partial y} = 2(4) = 8.$$

Using the gradient, the approximate change in (f) is:

$$\Delta f \approx \frac{\partial f}{\partial x} \cdot \Delta x + \frac{\partial f}{\partial y} \cdot \Delta y,$$

$$\Delta f \approx 6 \cdot 0.1 + 8 \cdot 0.2 = 0.6 + 1.6 = 2.2.$$

- The **actual change** in (f(x, y)) is:

$$\Delta f = 2.25.$$

- The **approximate change** using the gradient is:

$$\Delta f \approx 2.2.$$

What This Shows

This example demonstrates that partial derivatives reveal how changes in each input affect the function's output:

1. For x, the partial derivative is 6, meaning the output changes 6 times the increment in x. A 0.1 increase in x results in a

$$\Delta f = 6 \cdot \Delta x = 6 \cdot 0.1 = 0.6$$

change in the output of the function.

2. 1. For y, the partial derivative is 8, meaning the output changes 8 times the increment in y. A 0.2 increase in y results in a

$$\Delta f = 8 \cdot \Delta y = 8 \cdot 0.2 = 1.6$$

Partial derivatives provide a clear rate of change for each variable.

Chain Rule

Chain rule

$$\frac{d}{dx} f(g(x)) = f'(g(x)) \cdot g'(x)$$

The chain rule states that if z depends on y and y depends on x, then: $\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$

$y = 2x + 3$, Derivative: $\frac{dy}{dx} = 2$

.

$z = y^2$, Derivative: $\frac{dz}{dy} = 2y$

Using the chain rule,

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

$$\frac{dz}{dx} = (2y) \cdot (2)$$

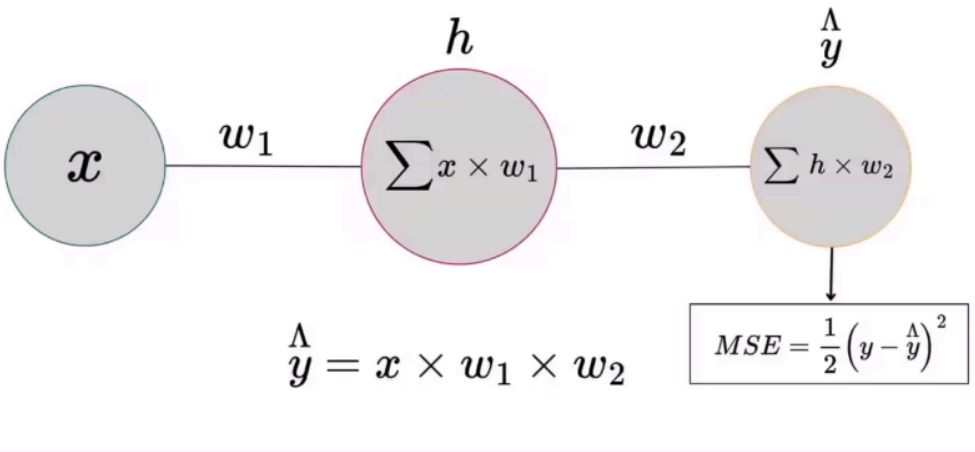
$$\frac{dz}{dx} = 4y$$

If you substitute $y = 2x + 3$, then: $\frac{dz}{dx} = 4(2x + 3) = 8x + 12$

So, $\frac{dz}{dx}$ represents the total rate of change of z with respect to x. It combines the effects of x on y and y on z.

Translation of the Chain Rule to the Neural Network:

$\frac{dz}{dx}$ in the original example corresponds to how the loss changes with respect to a weight.



For w_1, for instance: w_1 affects h, h affects $\hat{y}$, and $\hat{y}$ affects the loss.
The chain rule "chains" these effects together to find how w_1 impacts the loss.

Intuition:

Each weight affects the final loss by influencing the network's predictions step by step.
The chain rule lets us calculate these effects layer by layer, multiplying them together to get the total gradient.

Introduction to Backpropagation

The main objective of backpropagation is **to adjust the parameters of a neural network** (such as weights and biases) **to minimize the error between model predictions and actual values**.

It is the fundamental algorithm that **allows a neural network to learn from data**.

The objective of backpropagation is **to calculate the gradient of the loss function with respect to each weight in the neural network**, to update those weights and improve model performance.

Backpropagation is divided into two key phases within the training process:

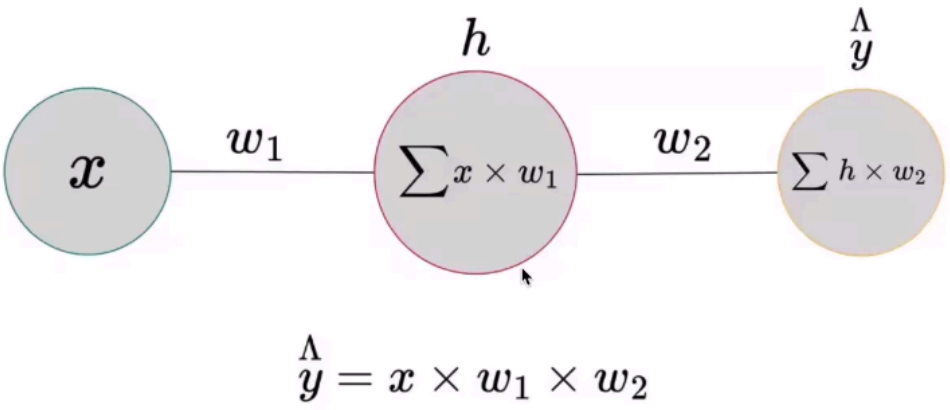
1. **Forward pass (forward propagation)**
   -
   An input is fed to the network.
   - The output is calculated and compared to the actual label using a loss function (such as MSE or Cross-Entropy).
2. **Backward pass (backpropagation)**
   -
   The chain rule of differential calculus is applied to calculate how the loss function changes with respect to each weight in the network.
   - The errors are propagated from the output to the previous layers.
   - The gradient of the loss with respect to each weight is obtained.

Why is it key in Machine Learning?

Backpropagation allows a multi-layered neural network:

- Learn complex and nonlinear patterns.
- Automatically adjust its parameters with training data.
- Improve its accuracy in tasks such as image classification, natural language processing, time series prediction, etc.

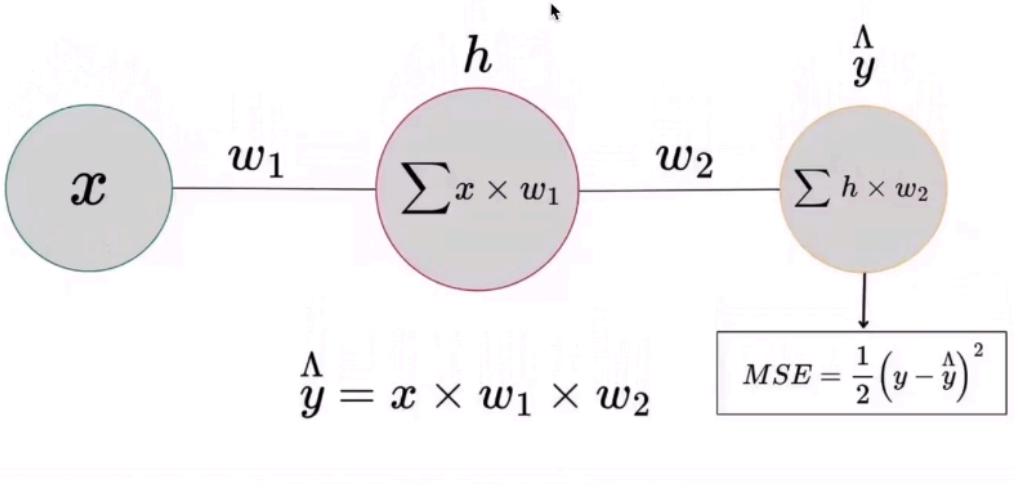
3) Backpropagation



We need to find the tuneable parameters $\frac{\partial \mathcal{L}}{\partial w_1}$ and $\frac{\partial \mathcal{L}}{\partial w_2}$.

3.1) First we must Find

$$\frac{\partial \mathcal{L}oss}{\partial \hat{y}}$$



Gradient Derivation of Mean Squared Error Loss Function

1) Mean Squared Error (MSE) Gradient Derivation

For **one** training example, the MSE is written as:

$$\text{MSE} = \frac{1}{2}(y - \hat{y})^2.$$

Derivation of the Gradient Step-by-Step

Loss Function Definition:
The loss is defined as:

$$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2.$$

Chain Rule Definition

The chain rule in calculus is defined as:

$$\frac{d}{dx} f(g(x)) = f'(g(x)) \cdot g'(x).$$

To compute the gradient of the loss function with respect to the model's output $\hat{y}$, we use the chain rule. Start by expanding the derivative:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \left(\frac{1}{2}(y - \hat{y})^2 \right).$$

Apply the power rule:

The term $(y - \hat{y})^2$ is a power of 2, so the derivative becomes:

$$\frac{\partial}{\partial \hat{y}} \left(\frac{1}{2}(y - \hat{y})^2 \right) = \frac{1}{2} \cdot 2 \cdot (y - \hat{y}) \cdot \frac{\partial}{\partial \hat{y}}(y - \hat{y}).$$

Simplify the constants:

The factor $\frac{1}{2} \cdot 2$ cancels out, leaving:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = (y - \hat{y}) \cdot \frac{\partial}{\partial \hat{y}}(y - \hat{y}).$$

Calculate the derivative of $(y - \hat{y})$:

$$\frac{\partial}{\partial \hat{y}}(y - \hat{y}) = -1.$$

Final Simplification:

Simplify the terms to get:

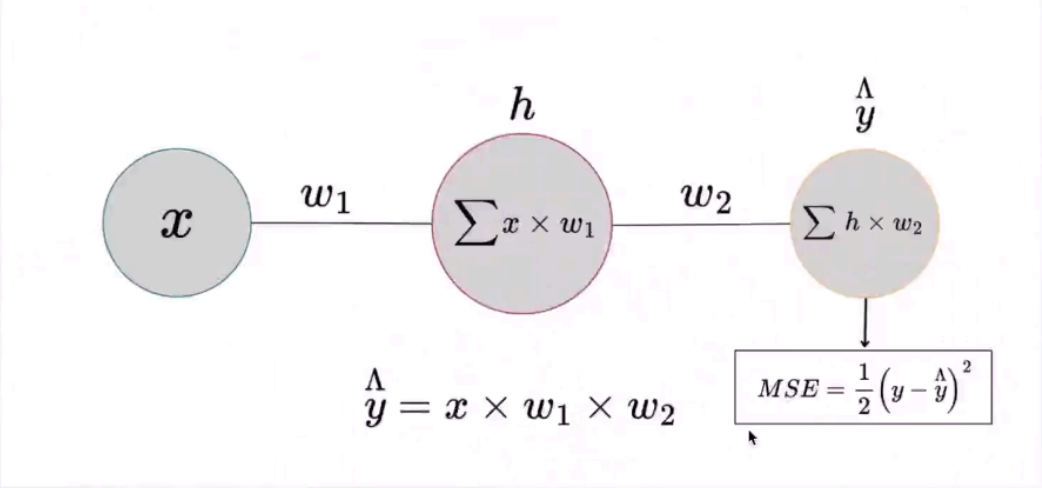
$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y.$$

Final Gradient Expression:

The gradient of the Mean Squared Error with respect to the output $\hat{y}$ is:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

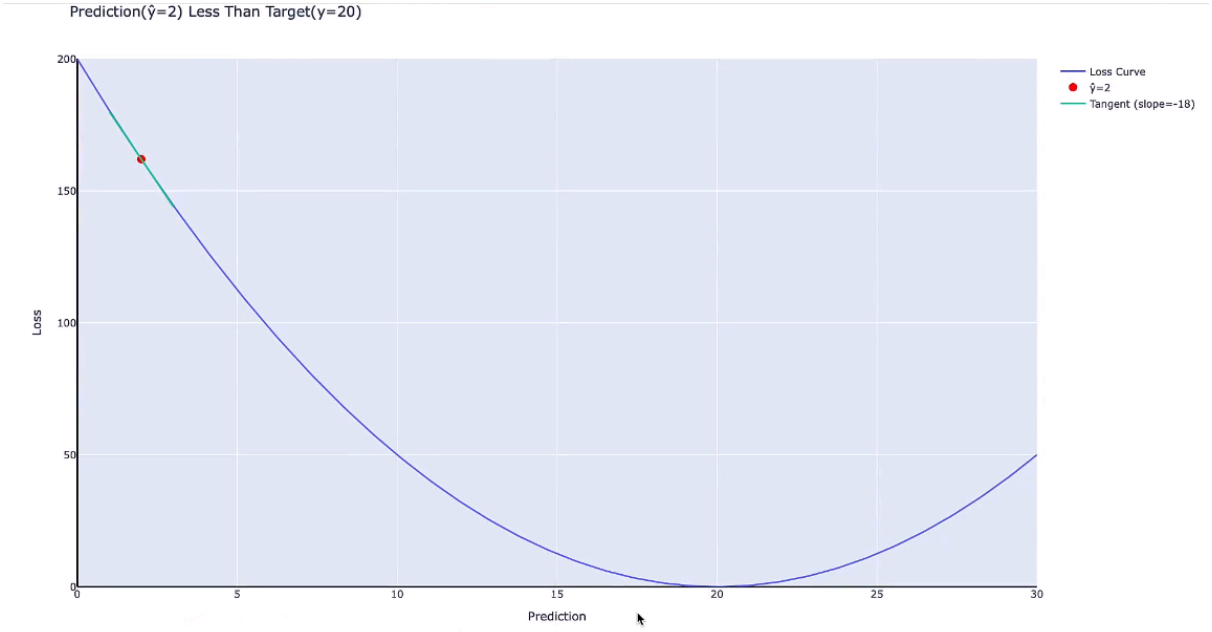
Visualizing the Loss Function and Understanding Gradients

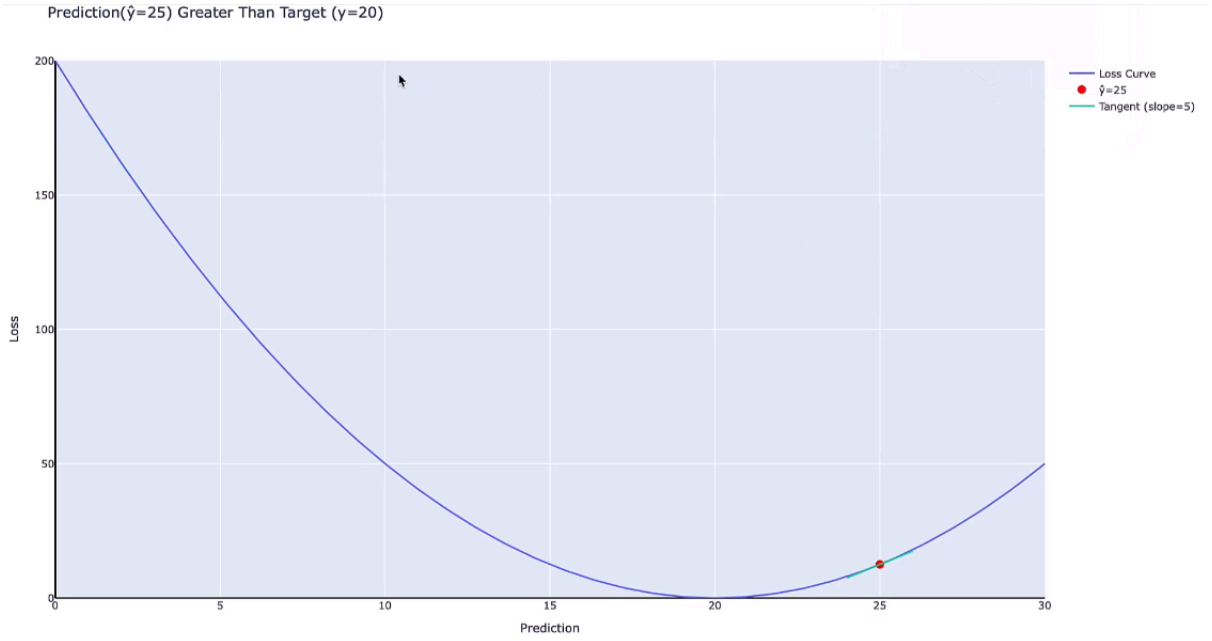


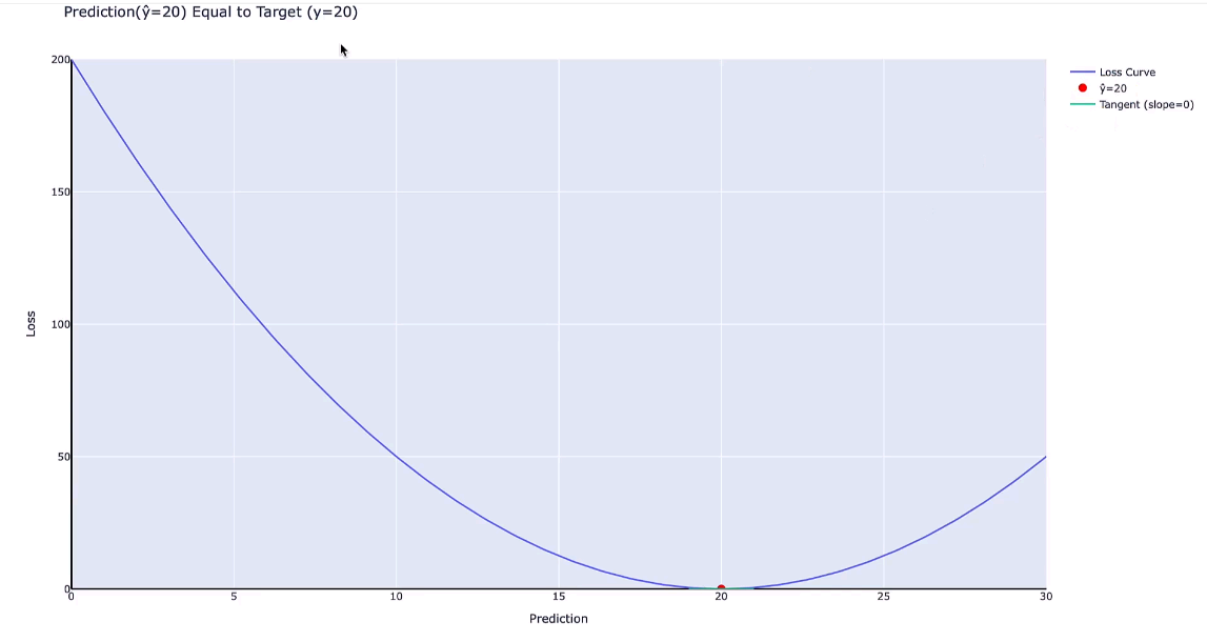
$$\mathcal{L}oss = \frac{1}{2}(y - \hat{y})^2 \implies \frac{\partial \mathcal{L}oss}{\partial \hat{y}} = \hat{y} - y.$$

Here, $\hat{y} = 2$, $y = 20$, so $\frac{\partial \mathcal{L}oss}{\partial \hat{y}} = (2 - 20) = -18.$

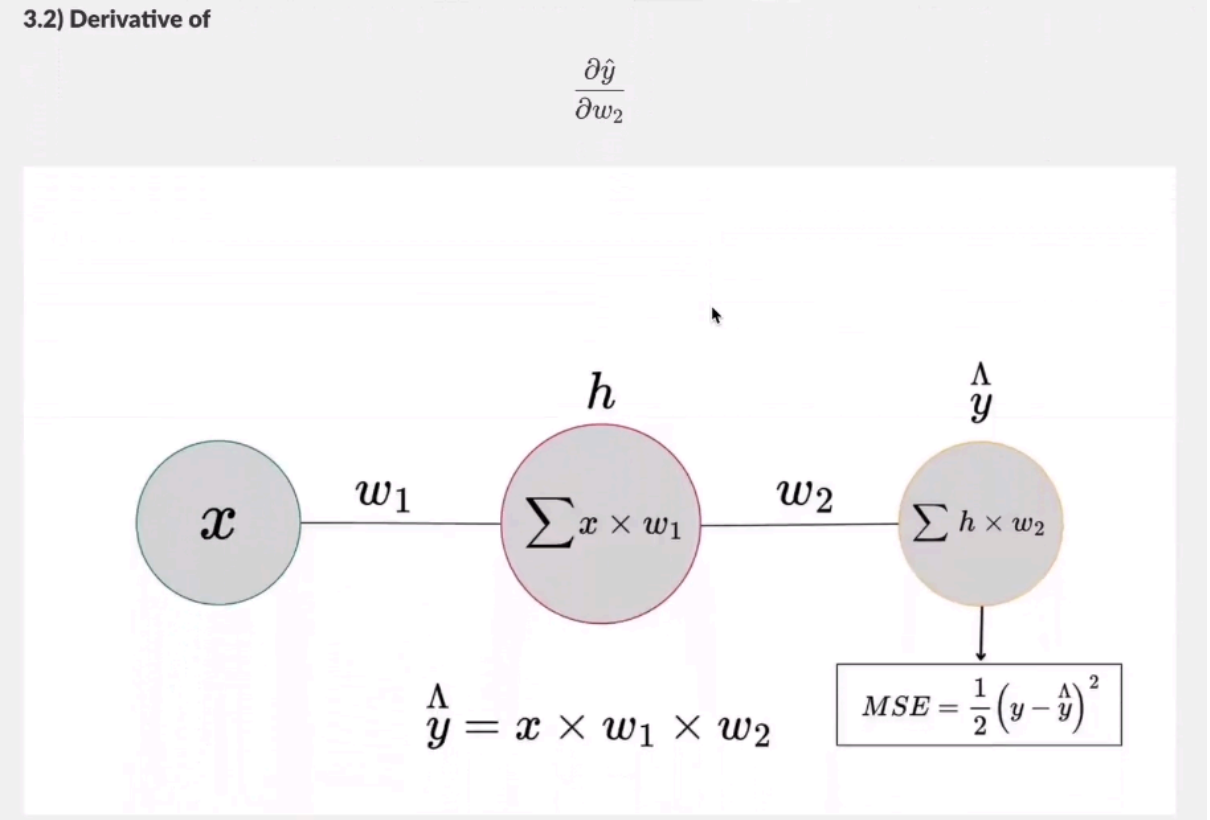
The gradient -18 indicates the loss decreases steeply as $\hat{y}$ increases; to minimize the loss, $\hat{y}$ needs to increase significantly.


Prediction(ŷ=2) Less Than Target(y=20)
Loss Curve
ŷ=2
Tangent (slope=-18)
Loss
Prediction
0
50
100
150
200
0
5
10
15
20
25
30


Prediction(ŷ=25) Greater Than Target (y=20)
Loss Curve
ŷ=25
Tangent (slope=5)
Loss
Prediction
0
50
100
150
200
0
5
10
15
20
25
30



Using the Chain Rule to See how w2 Affects the Final Loss



Remember this is the entire network

$$\hat{y} = x \times w_1 \times w_2,$$

1. **Hidden node output**:

$$h = x \times w_1 = 2 \times 2 = 4.$$

2. **Predicted output**:

$$\hat{y} = h \times w_2 = 4 \times 0.5 = 2.$$

$$\frac{\partial \hat{y}}{\partial w_2} = \frac{\partial}{\partial w_2}(h \cdot w_2)$$

Since h is multiplied by w_2, it is treated as a constant during differentiation. Therefore:

$$\frac{\partial \hat{y}}{\partial w_2} = h \cdot 1$$

Finally, substituting $h = 4$:

$$\frac{\partial \hat{y}}{\partial w_2} = 4 \cdot 1 = 4.$$

Note: If h were added to w_2 (e.g., $\hat{y} = h + w_2$), then:

$$\frac{\partial \hat{y}}{\partial w_2} = 1.$$

This value of 4 means that a small change in w_2 affects $\hat{y}$ by a magnitude of 4. However, we still need to determine how a small char

For that, we need to apply the chain rule.

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_2} = (-18) \times 4 = -72.$$

What does a partial derivative of -72 with respect to w_2, it means:

1. Direction
2. Magnitude

In short, a derivative of -72 says: "If you increase w_2 by a tiny step, the loss will drop quickly (because the slope is negative and si

Let's prove this

$$\text{Given:} \quad \frac{\partial \mathcal{L}}{\partial w_2} = -72.$$

If w_2 increases by $\Delta w_2 = 0.1,$ then the change in the loss $\Delta \mathcal{L}$ is given by:

$$\Delta \mathcal{L} = \frac{\partial \mathcal{L}}{\partial w_2} \cdot \Delta w_2.$$

Substitute the values:

$$\Delta \mathcal{L} = -72 \cdot 0.1 = -7.2.$$

Therefore, increasing w_2 by 0.1 results in the loss decreasing by 7.2.

$$\text{Given:} \quad x = 2,\ w_1 = 2,\ w_2 = 0.6.$$

1. Hidden node output:

$$h = x \cdot w_1 = 2 \cdot 2 = 4.$$

2. Predicted output:

$$\hat{y} = h \cdot w_2 = 4 \cdot 0.6 = 2.4.$$

3. Mean Squared Error (MSE) loss:

$$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2.$$

Assume the target $y = 20$.

Substitute:

$$\mathcal{L} = \frac{1}{2}(20 - 2.4)^2 = \frac{1}{2} \cdot (17.6)^2 = \frac{1}{2} \cdot 309.76 = 154.88.$$

Previously, with $w_2 = 0.5$, $\hat{y} = 2.0$, and the MSE was:

$$\mathcal{L}_{\text{original}} = \frac{1}{2}(20 - 2.0)^2 = \frac{1}{2} \cdot (18.0)^2 = \frac{1}{2} \cdot 324 = 162.0.$$

4. Loss reduction:

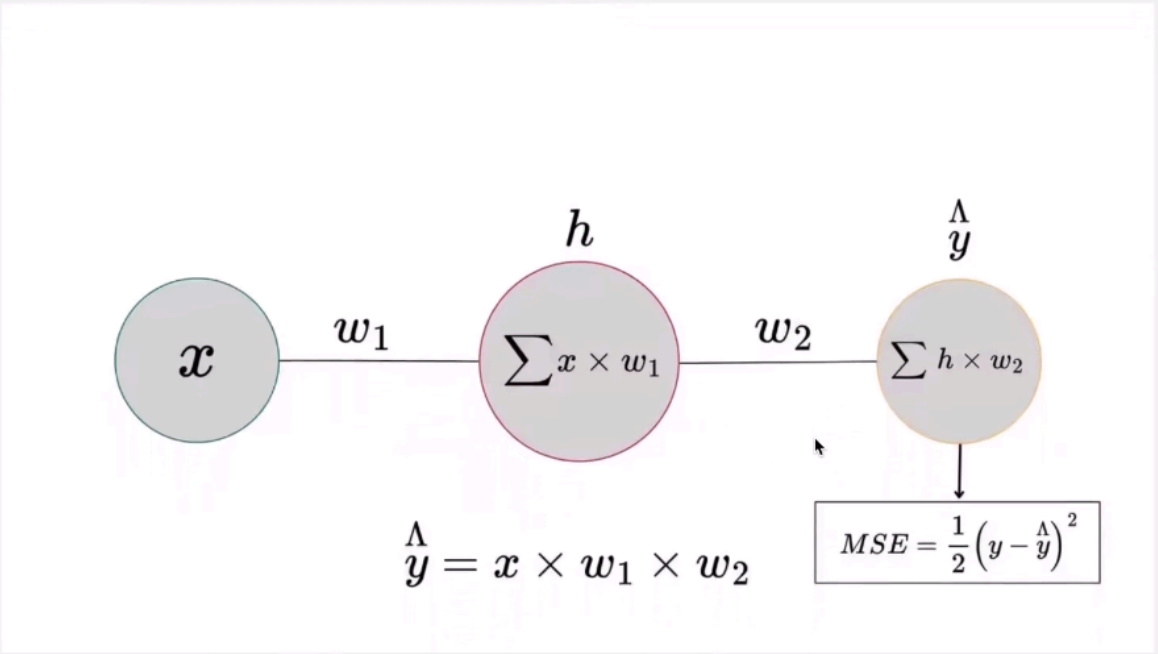
$$\Delta\mathcal{L} = \mathcal{L}_{\text{original}} - \mathcal{L} = 162.0 - 154.88 = 7.12.$$

Thus, increasing w_2 to 0.6 decreases the loss by approximately 7.12.

Backpropagation of w1

3.3) Derivative of

$$\frac{\partial \hat{y}}{\partial w_1}$$



$$\hat{y} = x \times w_1 \times w_2$$

$$\frac{\partial \hat{y}}{\partial w_1} = x \times 1 \times w_2$$

Given $x = 2$ and $w_2 = 0.5$, $\frac{\partial \hat{y}}{\partial w_1} = 2 \times 0.5 = 1.$

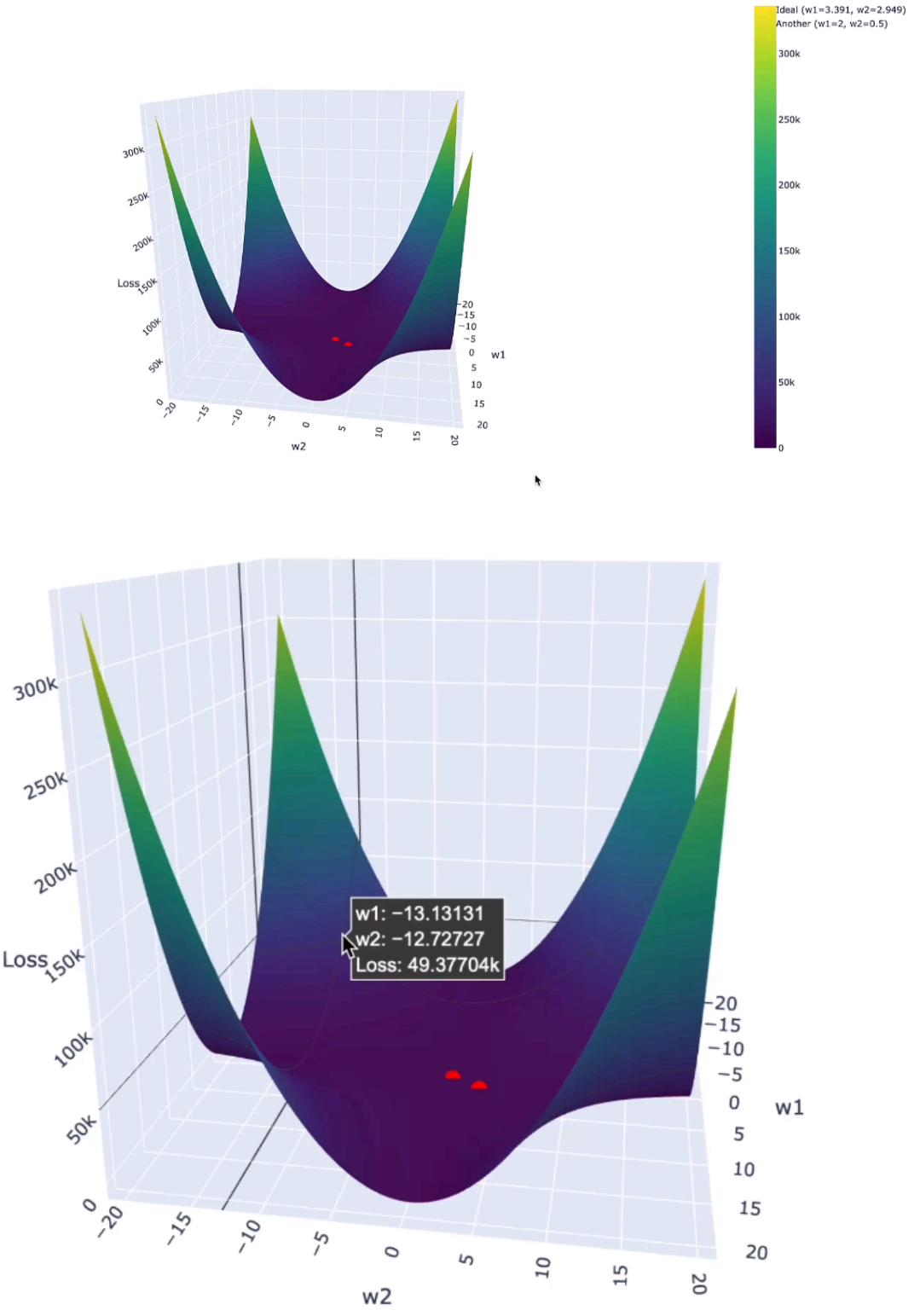
Thus,

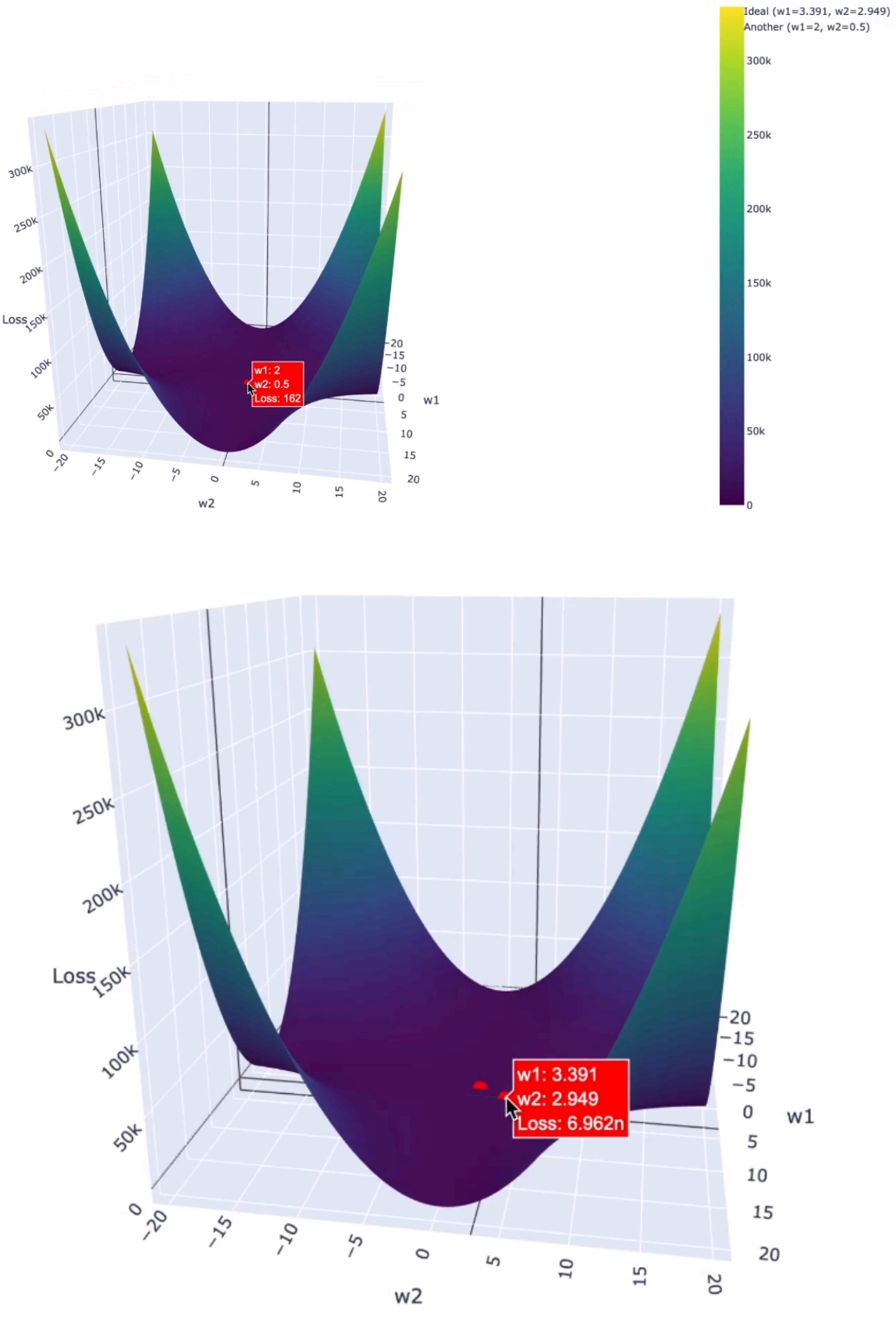
$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_1} = (-18) \times 1 = -18.$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = -18$$

Introduction to Gradient Descent Visually

3D Surface: Loss vs. (w1, w2)




Ideal (w1=3.391, w2=2.949)
Another (w1=2, w2=0.5)
300k
250k
200k
150k
100k
50k
0
Loss
w1: 2
w2: 0.5
Loss: 162
w1
w2
w1: 3.391
w2: 2.949
Loss: 6.962n

4) Weight Updates using Gradient Descent

Gradient Descent Visually

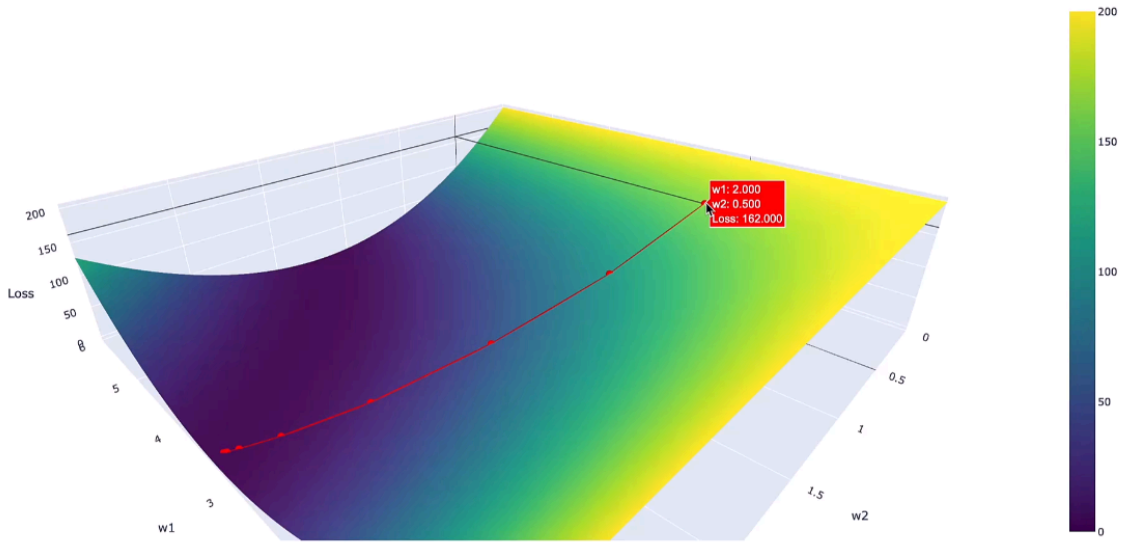
$$\text{target: } y = 20$$

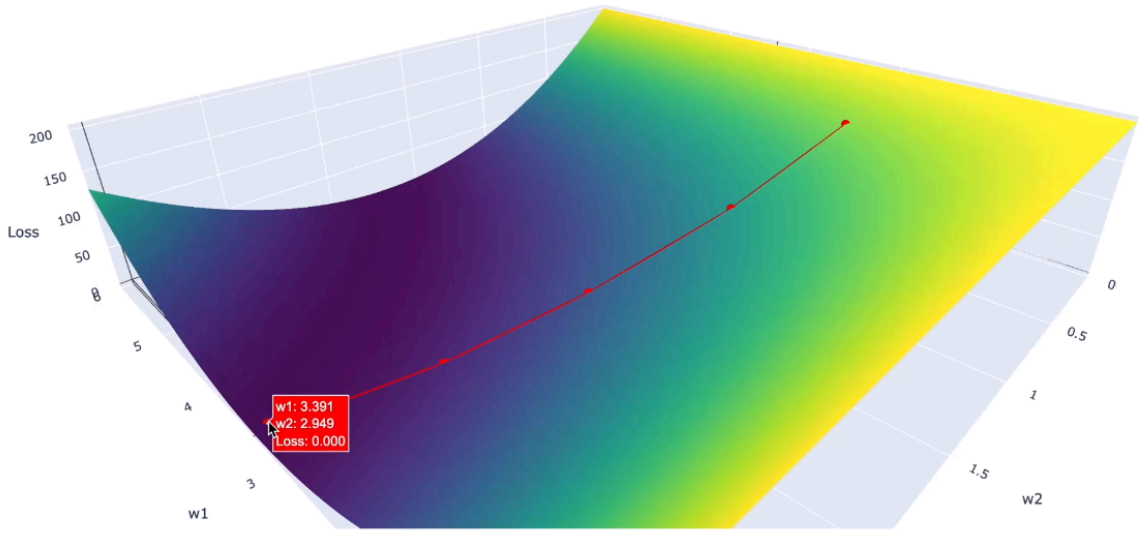
$$w_1 = 3.391, \quad w_2 = 2.949$$

$$\text{Predicted output: } \quad \hat{y} = 2 \cdot w_1 \cdot w_2$$

$$\text{Predicted output: } \quad \hat{y} = 2 \cdot 3.391 \cdot 2.949 = 20.000118$$

attachment:a9278236-961f-42e4-a4a3-4693d1c77016:Vdeo_sin_ttulo__Hecho_con_Clipchamp.mp4





Gradient Descent

Gradient Descent Explanation

The gradient (e.g., $\frac{\partial \mathcal{L}}{\partial w_2}$) points in the direction of the steepest increase of the loss $\mathcal{L}$.

1. If we want to minimize the loss, we need to move in the opposite direction of the gradient.

2. Gradient descent uses this idea and updates w as follows:

$$w \leftarrow w - \alpha \frac{\partial \mathcal{L}}{\partial w},$$

$\frac{\partial \mathcal{L}}{\partial w}$ tells us the slope of the loss with respect to w

α is the learning rate (step size)

Understanding the Learning Rate (Alpha)

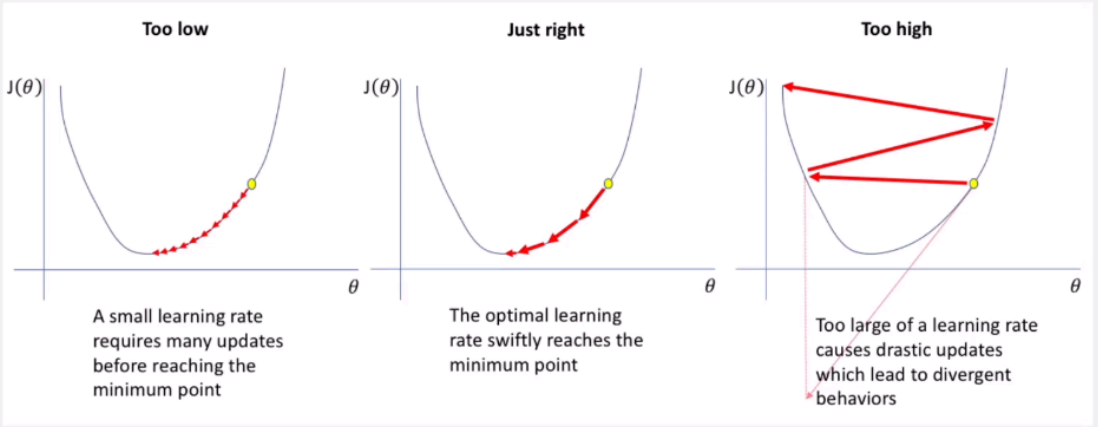
Learning Rate (α):

α, or the learning rate, controls the size of the steps taken during gradient descent.

1. A small α makes the learning process slow but ensures stability.

2. A large α speeds up learning but risks overshooting the minimum.

Balance is key: α must be chosen carefully to ensure efficient and stable convergence.



Moving in the Opposite Direction of the Gradient

Simple Example of Going in the Opposite Direction of the Gradient

Numerical Example: Weight Updates in Gradient Descent

Initial weight: $w = 1.0$, Learning rate: $\alpha = 0.1$

Case 1: Positive Gradient

$$\text{Gradient of the loss: } \frac{\partial \mathcal{L}}{\partial w} = +2.0$$

$$\text{Update rule: } w = w - \alpha \cdot \frac{\partial \mathcal{L}}{\partial w}$$

$$w = 1.0 - 0.1 \cdot 2.0 = 1.0 - 0.2 = 0.8$$

Explanation: Since the gradient is positive, moving in the opposite direction means decreasing the weight.

Case 2: Negative Gradient

Gradient of the loss: $\frac{\partial \mathcal{L}}{\partial w} = -3.0$

Update rule: $w = w - \alpha \cdot \frac{\partial \mathcal{L}}{\partial w}$

$$w = 1.0 - 0.1 \cdot (-3.0) = 1.0 + 0.3 = 1.3$$

Explanation: Since the gradient is negative, moving in the opposite direction means increasing the weight.

Summary:

If the gradient is positive, subtracting it decreases the weight.

If the gradient is negative, subtracting it increases the weight.

Calculating Gradient Descent by Hand

Numerically Calculating Gradient Descent

$\frac{\partial \mathcal{L}}{\partial w}$ is the gradient of the loss function $\mathcal{L}$ with respect to w.

Remember we calculated these:

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_1} = (-18) \times 1 = -18$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_2} = (-18) \times 4 = -72$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = -18$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = -72$$

Using gradient descent with $\alpha = 0.01$

Update w_2

$$w_2 \leftarrow w_2 - \alpha \frac{\partial \mathcal{L}}{\partial w_2} = 0.5 - 0.01 \times (-72) = 0.5 + 0.72 = 1.22$$

2. Update w_1

$$w_1 \leftarrow w_1 - \alpha \frac{\partial \mathcal{L}}{\partial w_1} = 2 - 0.01 \times (-18) = 2 + 0.18 = 2.18$$

So after one update step:

- $w_1 = 2.18$
- $w_2 = 1.22$

5) Forward Pass (After Update)

1. **Hidden node output**:

$$h = x \times w_1 = 2 \times 2.18 = 4.36$$

2. **New predicted output**:

$$\hat{y}_{\text{new}} = h \times w_2 = 4.36 \times 1.22 \approx 5.3192$$

3. **New Loss**:

$$\mathcal{L}_{\text{new}} = \frac{1}{2}(y - \hat{y}_{\text{new}})^2 = \frac{1}{2}(20 - 5.3192)^2 = \frac{1}{2}(14.6808)^2 \approx \frac{1}{2} \times 215.76 \approx 107.88$$

Compare Old vs. New Loss after One Step

Old loss: 162, New loss: ≈ 107.88

Summary

1. **Initial weights**:

$$w_1 = 2,\ w_2 = 0.5$$

2. **Forward pass**:

$$\text{Compute hidden output } h, \text{ then final output } \hat{y}.$$

3. **Loss (MSE)**:

$$\frac{1}{2}(y - \hat{y})^2$$

4. **Backpropagation**:

$$\text{- Get } \partial\mathcal{L}/\partial w_2 \text{ and } \partial\mathcal{L}/\partial w_1 \text{ via the chain rule.}$$

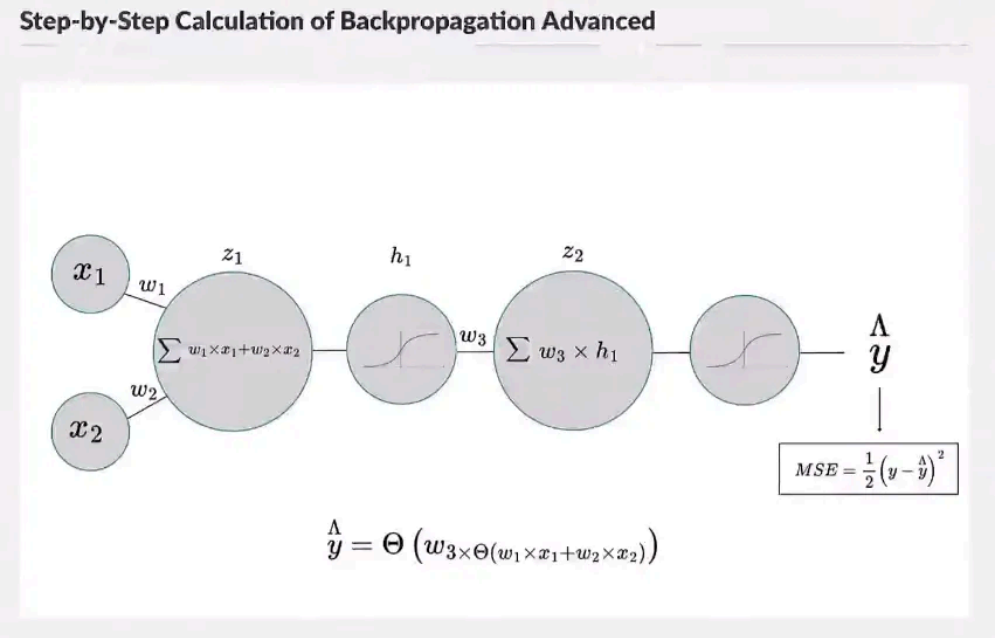
$$\text{- Update each weight: } w \leftarrow w - \alpha\frac{\partial\mathcal{L}}{\partial w}$$

5. **Loss after update is lower**, showing that learning is proceeding in the right direction.

Google Colab

https://colab.research.google.com/drive/1gfhmkojTj2mlKMGZD5zlOLbQ9lcsBXT6?usp=sharing

Introduction to Our Complex Neural Network



We have two inputs (x_1 and x_2), a single hidden neuron (h_1), and a single output neuron ($\hat{y}$).

Below are the initial parameters we will work with.

$$x_1 = 1, \quad x_2 = 2$$

Weights (from inputs to hidden, and hidden to output):

$$w_1 = 0.10 \quad , \quad w_2 = -0.30 \quad , \quad w_3 = 0.50 \quad)$$

We will use the Sigmoid activation function for both neurons, which is defined as follows:

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

Finally, our target (desired) output for this forward pass is given by:

$$y = 1$$

Conducting the Forward Pass

1.1. Hidden neuron h_1

$$z_1 = w_1 x_1 + w_2 x_2 = 0.10 \times 1 + (-0.30) \times 2 = 0.10 - 0.60 = -0.50$$

$$h_1 = \sigma(z_1) = \frac{1}{1 + e^{-(-0.50)}} = \frac{1}{1 + e^{0.50}}$$

For a rough numeric approximation:

$$e^{0.50} \approx 1.65 \quad \Longrightarrow \quad h_1 \approx \frac{1}{1 + 1.65} = \frac{1}{2.65} \approx 0.3775$$

1.2. Output neuron $\hat{y}$

$$z_2 = w_3 \cdot h_1 = 0.50 \times 0.3775 \approx 0.18875$$

$$\hat{y} = \sigma(z_2) \approx \frac{1}{1 + e^{-0.18875}} \approx 0.54705$$

1.3. Error (using Mean Squared Error for a single example)

$$\text{Loss} = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(1 - 0.54705)^2 = \frac{1}{2}(0.45295)^2 \approx \frac{1}{2} \times 0.20516 \approx 0.10258$$

Getting Started with Backpropagation

We'll compute partial derivatives of the error/loss with respect to each weight and then do a simple **gradient descent update**:

$$w := w - \alpha \frac{\partial \text{Loss}}{\partial w}$$

where α is the learning rate (let's pick $\alpha = 0.1$ for example).

2.1. Derivatives at the Output Neuron

1. **Loss derivative with respect to $\hat{y}$:**

$$\frac{\partial \text{Loss}}{\partial \hat{y}} = (\hat{y} - y) = (0.54705 - 1) = -0.45295$$

2. **Output neuron uses the Sigmoid function**, so the derivative of $\hat{y} = \sigma(z_2)$ with respect to z_2 is:

$$\frac{d\hat{y}}{dz_2} = \hat{y} \cdot (1 - \hat{y}) = 0.54705 \cdot (1 - 0.54705) \approx 0.54705 \cdot 0.45295 \approx 0.24758$$

3. **Chain rule for $\frac{\partial \text{Loss}}{\partial z_2}$:**

$$\frac{\partial \text{Loss}}{\partial z_2} = \frac{\partial \text{Loss}}{\partial \hat{y}} \cdot \frac{d\hat{y}}{dz_2} = -0.45295 \cdot 0.24758 \approx -0.11222$$

Getting the Derivative of the Sigmoid Activation Function(Optional)

Step-by-Step Derivation of the Sigmoid Function's Derivative

Below is a **step-by-step derivation** of the sigmoid function's derivative. We start with the definition of the sigmoid (a.k.a. logistic) function:

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

and show in detail why

$$\frac{d}{du}\sigma(u) = \sigma(u)\,[1 - \sigma(u)]$$

or

$$\hat{y} \cdot (1 - \hat{y})$$

1. Write Sigmoid in a Form Convenient for Differentiation

$$\sigma(u) = \frac{1}{1 + e^{-u}} = \left(1 + e^{-u}\right)^{-1}.$$

To differentiate, we can use the **chain rule**.

2. Apply the Chain Rule

2.1. Outer function

Let's denote

$$f(x) = x^{-1},$$

so

$$f'(x) = -x^{-2} = -\frac{1}{x^2}.$$

2.2. Inner function

Let

$$x = 1 + e^{-u}.$$

Its derivative with respect to

$$u$$

is

$$\frac{dx}{du} = \frac{d}{du}\left(1 + e^{-u}\right) = 0 + \frac{d}{du}\left(e^{-u}\right)$$

Since

$$\frac{d}{du}\left(e^{g(u)}\right) = e^{g(u)}g'(u),$$

we have

$$\frac{d}{du}\left(e^{-u}\right) = e^{-u} \cdot (-1) = -e^{-u}.$$

Hence,

$$\frac{dx}{du} = -e^{-u}$$

2.3. Combine the pieces

By the chain rule,

$$\frac{d}{du}\left[(1+e^{-u})^{-1}\right] = f'(x)\cdot\frac{dx}{du} = \left[-\frac{1}{(1+e^{-u})^2}\right]\cdot\left[-e^{-u}\right].$$

The two minus signs cancel out, so we get

$$\frac{d\sigma(u)}{du} = \frac{e^{-u}}{(1+e^{-u})^2}.$$

3. Rewrite to Show

$$\sigma(u)\left[1-\sigma(u)\right]$$

We now have:

$$\frac{d\sigma(u)}{du} = \frac{e^{-u}}{(1+e^{-u})^2}.$$

3.1. Factor out

$$\sigma(u)$$

Recall,

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

So let's multiply-and-divide cleverly to make it appear:

$$\frac{e^{-u}}{(1+e^{-u})^2} = \frac{1}{1+e^{-u}}\times\frac{e^{-u}}{1+e^{-u}}.$$

Notice that

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

Thus the first part is exactly

$$\sigma(u)$$

3.2. Rewrite the Second Factor

Given a fraction of the form $\frac{a+b}{c}$, it can be separated into $\frac{a}{c}+\frac{b}{c}$.

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

Hence,

$$\frac{e^{-u}}{(1+e^{-u})^2} = \frac{1}{1+e^{-u}} \times \left(1-\frac{1}{1+e^{-u}}\right) = \sigma(u)\left[1-\sigma(u)\right].$$

4. Final Expression

Therefore,

$$\frac{d}{du}\sigma(u) = \sigma(u)\left[1-\sigma(u)\right].$$

gives us the gradient of the sigmoid function **with respect to its input** u.
Tells us how the input u influences the output of the sigmoid function

Implementing Backpropagation with the Chain Rule

2.1. Derivatives at the Output Neuron

1. **Loss derivative with respect to $\hat{y}$**:

$$\frac{\partial \text{Loss}}{\partial \hat{y}} = (\hat{y}-y) = (0.54705-1) = -0.45295$$

2. **Output neuron uses the Sigmoid function**, so the derivative of $\hat{y}=\sigma(z_2)$ with respect to z_2 is:

$$\frac{d\hat{y}}{dz_2} = \hat{y}\cdot(1-\hat{y}) = 0.54705\cdot(1-0.54705) \approx 0.54705\cdot 0.45295 \approx 0.24758$$

3. **Chain rule for $\frac{\partial \text{Loss}}{\partial z_2}$**:

$$\frac{\partial \text{Loss}}{\partial z_2} = \frac{\partial \text{Loss}}{\partial \hat{y}}\cdot\frac{d\hat{y}}{dz_2} = -0.45295\cdot 0.24758 \approx -0.11222$$

This means that an increase in z2 will decrease the loss

Understanding How w3 Affects the Final Loss

$$z_2 = w_3 \cdot h_1$$

$$\frac{\partial z_2}{\partial w_3} = h_1$$

Given that (h1 = 0.3775), we substitute:

$$\frac{\partial z_2}{\partial w_3} = 0.3775$$

$$\frac{\partial \text{Loss}}{\partial w_3} = \frac{\partial \text{Loss}}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_3} = -0.11222 \times h_1 = -0.11222 \times 0.3775 \approx -0.04239$$

To determine how the loss changes when (w_3) increases by (0.1):

$$\Delta \text{Loss} \approx \frac{\partial \text{Loss}}{\partial w_3} \cdot \Delta w_3$$

Substitute the values:

$$\Delta \text{Loss} \approx -0.04239 \cdot 0.1 = -0.004239$$

Thus, the loss would decrease by approximately (0.004239)

Calculating Gradients for Z1

Find

$$\frac{\partial z_2}{\partial h_1}$$

$$z_2 = w_3 \cdot h_1$$

$$\frac{\partial z_2}{\partial h_1} = w_3$$

Given that (w3 = 0.5):

$$\frac{\partial z_2}{\partial h_1} = 0.5$$

1. **Hidden neuron also uses Sigmoid.** We need $\frac{\partial h_1}{\partial z_1} = h_1(1 - h_1)$:

$$\frac{\partial h_1}{\partial z_1} = 0.3775 \times (1 - 0.3775) = 0.3775 \times 0.6225 \approx 0.2350$$

2. **Chain rule for $\frac{\partial \text{Loss}}{\partial z_1}$:**

- We already have $\frac{\partial \text{Loss}}{\partial z_2} \approx -0.11222$
- $\frac{\partial z_2}{\partial h_1} = w_3 = 0.50$

$$\frac{\partial \text{Loss}}{\partial z_1} = \frac{\partial \text{Loss}}{\partial z_2} \cdot \frac{\partial z_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial z_1} = (-0.11222) \times (0.50) \times 0.2350 \approx -0.01318$$

Understanding How w1 and w2 Affect the Loss

$$z_1 = w_1 \cdot x_1 + w_2 \cdot x_2$$

$$\frac{\partial z_1}{\partial w_1} = x_1$$

$$\frac{\partial z_1}{\partial w_2} = x_2$$

$$\frac{\partial \text{Loss}}{\partial w_1} = \frac{\partial \text{loss}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} = (-0.01318) \times x_1 = -0.01318 \times 1 = -0.01318$$

$$\frac{\partial \text{Loss}}{\partial w_2} = \frac{\partial \text{Loss}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_2} = (-0.01318) \times x_2 = -0.01318 \times 2 = -0.02636$$

Implementing Gradient Descent by Hand

$$w_3 \leftarrow w_3 - \alpha \frac{\partial \text{Loss}}{\partial w_3} = 0.50 - 0.1 \times (-0.04239) = 0.50 + 0.004239 \approx 0.50424$$

$$w_1 \leftarrow w_1 - \alpha \frac{\partial \text{Loss}}{\partial w_1} = 0.10 - 0.1 \times (-0.01318) = 0.10 + 0.001318 \approx 0.10132$$

$$w_2 \leftarrow w_2 - \alpha \frac{\partial \text{Loss}}{\partial w_2} = -0.30 - 0.1 \times (-0.02636) = -0.30 + 0.002636 \approx -0.29736$$

3. Summary of Updated Weights

After **one** simple iteration of backprop:

- w_1 from 0.10 to 0.10132
- w_2 from -0.30 to -0.29736
- w_3 from 0.50 to 0.50424

New Error Calculation After Weight Update

1. Forward Pass with Updated Weights

Hidden Neuron h_1:

$$z_1 = w_1 \cdot x_1 + w_2 \cdot x_2 = 0.10132 \cdot 1 + (-0.29736) \cdot 2 = -0.4934$$

$$h_1 = \sigma(z_1) = \frac{1}{1 + e^{-z_1}} = \frac{1}{1 + e^{0.4934}} \approx 0.3791$$

Output Neuron $\hat{y}$:

$$z_2 = w_3 \cdot h_1 = 0.50424 \cdot 0.3791 = 0.1912$$

$$\hat{y} = \sigma(z_2) = \frac{1}{1 + e^{-z_2}} = \frac{1}{1 + e^{-0.1912}} \approx 0.5476$$

2. New Error Calculation:

Using the Mean Squared Error (MSE) formula:

$$\text{Error} = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(1 - 0.5476)^2 = \frac{1}{2}(0.4524)^2 \approx \frac{1}{2} \cdot 0.2046 \approx 0.1023$$

Final Result:

- **New Error**:

$$\text{Error} \approx 0.1023$$

- **Previous Error**:

$$\text{Error} \approx 0.10258$$

Error Decrease Calculation

The initial error was

$$0.10258$$

and the new error is

$$0.1023$$

The decrease in error is:

$$\text{Error Decrease} = \text{Initial Error} - \text{New Error} = 0.10258 - 0.1023 = 0.00028$$

Final Result:

The error decreased (after the first iteration) by

$$0.00028$$

Google Colab

https://colab.research.google.com/drive/1qDfFlPlOYCC9okV-xAOwjW51mA4yYbAt?usp=sharing