Learning in Neural Networks: From Intuition to Backpropagation

Sayan Chaki]

March 20, 2025

Outline

The Mystery of Learning

- How do we humans learn?
  - Our brains contain approximately 86 billion neurons
  - Learning involves forming and strengthening connections
  - No explicit programming required
- We observe, make mistakes, and adapt
  - Example: Learning to ride a bicycle
  - Fall $\rightarrow$ Adjust $\rightarrow$ Try again $\rightarrow$ Improve
- Learning is an iterative process of improvement
- Can we enable machines to learn in similar ways?

Learning in Biological vs. Artificial Systems

Biological Learning

- Synaptic connections strengthen or weaken
- Neurons that fire together, wire together (Hebb's Law)
- Experience-dependent plasticity
- Reinforcement through neurotransmitters

Artificial Neural Networks

- Weight parameters increase or decrease
- Connections adjusted based on error
- Data-dependent adaptation
- Reinforcement through error signals

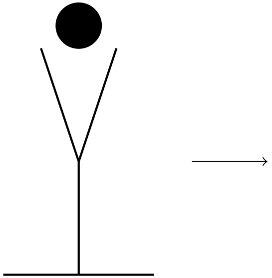
Key Insight

Both systems learn through adjustment of connections based on feedback.

Learning Metaphors

The Child Learning to Walk

- Starts with no knowledge of balance or coordination
- Tries to stand, falls down (error signal)
- Brain processes what went wrong
- Adjusts muscle control (parameter updates)
- With each attempt, gradually improves
- Eventually masters walking without conscious thought



Neural Network Parallel

Initial random weights → Forward pass → Error calculation → Weight adjustment → Improved performance

Finding the Valley in Fog

- You're on a mountain in thick fog
- Goal: Reach the lowest point (valley)
- Can only feel the slope under your feet
- You move in the direction of steepest descent
- Take small steps to avoid overshooting
- Each step brings you closer to the valley

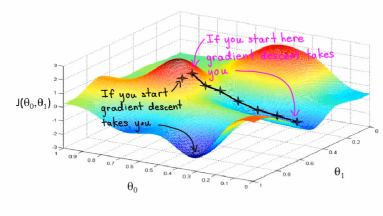


Figure: Descent

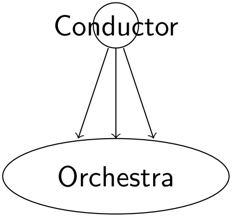
Network Parallel

The error landscape has many dimensions (one for each weight), and gradient descent helps navigate this landscape to find the minimum error.

Learning Metaphors

The Orchestra Conductor

- Orchestra plays a piece (forward pass)
- Conductor hears the overall sound (output)
- Identifies which instruments are out of tune (error attribution)
- Provides specific feedback to each musician (backpropagation)
- Musicians adjust their playing (weight updates)
- Overall performance improves



Neural Network Parallel

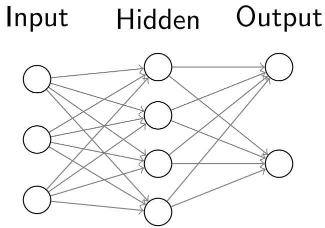
Backpropagation distributes error corrections throughout the network, telling each weight how much it contributed to the error.

Real-World Examples of Learning Systems

- **Computer Vision**
  - Image classification (Is this a cat or a dog?)
  - Object detection (Where is the pedestrian in this image?)
  - Facial recognition (Who is this person?)
- **Natural Language Processing**
  - Sentiment analysis (Is this review positive or negative?)
  - Machine translation (English to French)
  - Text generation (Complete this paragraph)
- **Reinforcement Learning**
  - Game playing (Chess, Go, video games)
  - Robotics (Learning to walk, grasp objects)
  - Autonomous driving

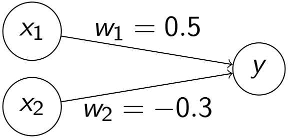
Neural Networks: A Brief Overview

- Inspired by biological neural networks (our brains)
- Composed of interconnected nodes (neurons)
- Organized in layers:
  - Input layer: Receives data
  - Hidden layer(s): Processes information
  - Output layer: Provides predictions
- Each connection has a weight parameter
- Neurons apply an activation function to their input



What Does a Neural Network Need to Learn?

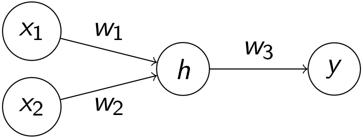
- A neural network is a collection of connections
- Each connection has a **weight** (a number)
- Learning means finding the right weights
- Goal: Find weights that make predictions match reality



Example: If $y = w_1 x_1 + w_2 x_2$ and we want y to be 1 when $x_1 = 2$ and $x_2 = 1$, we need:

$$w_1 \times 2 + w_2 \times 1 = 1 \tag{1}$$

Learning as Weight Adjustment



Example: We want y to be 1 when $x_1 = 1, x_2 = 0$

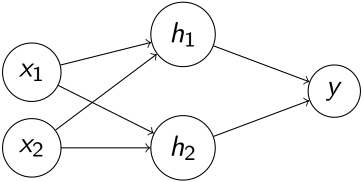
- Current weights: $w_1 = 0.3, w_2 = 0.4, w_3 = 0.5$
- Forward pass:
  $h = \sigma(w_1 x_1 + w_2 x_2) = \sigma(0.3 \times 1 + 0.4 \times 0) = \sigma(0.3) = 0.574$
- Output: $y = \sigma(w_3 h) = \sigma(0.5 \times 0.574) = \sigma(0.287) = 0.571$
- Error: y should be 1, but it's 0.571
- Question: Which weights should we adjust, and by how much?

A Concrete Example: XOR Problem

The XOR function:

x_1	x_2	$\text{XOR}(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

- XOR is not linearly separable
- Requires at least one hidden layer
- Network must learn appropriate weights



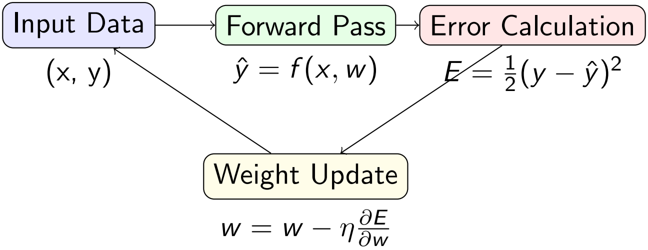
How Learning Works - Intuitively

1. Present an input to the network (e.g., $x_1 = 1, x_2 = 0$)
2. Calculate the output using current weights
3. Compare output to desired target (e.g., $y_{target} = 1$)
4. Calculate the error (difference): $E = \frac{1}{2}(y - y_{target})^2$
5. Adjust weights to reduce error
6. Repeat with many examples

Key Questions:

- How much should we adjust each weight?
- In which direction should we adjust?
- How do we distribute the error across all weights?

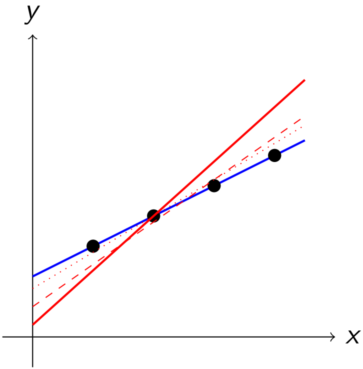
The Learning Loop



- This loop constitutes one training iteration
- Training typically requires thousands to millions of iterations
- Each iteration improves the weights slightly
- Eventually, the weights converge to good values

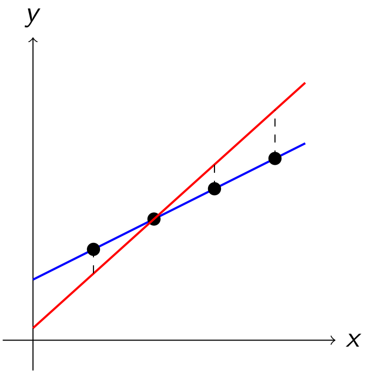
The Learning Process: Visualization

Linear Regression Example:



- Blue line: True relationship ($y = 0.5x + 1$)
- Black dots: Training data
- Red lines: Model predictions during training
  - Solid: Initial prediction
  - Dashed: After some training
  - Dotted: Final prediction

Learning as Error Minimization



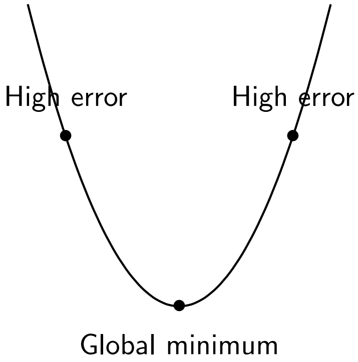
- Errors (dashed lines) are the differences between predictions and targets
- We want to minimize these errors
- Total error: $E = \sum_i \frac{1}{2}(y_i - \hat{y}_i)^2$
- Learning means finding weights that minimize E

Key Insight

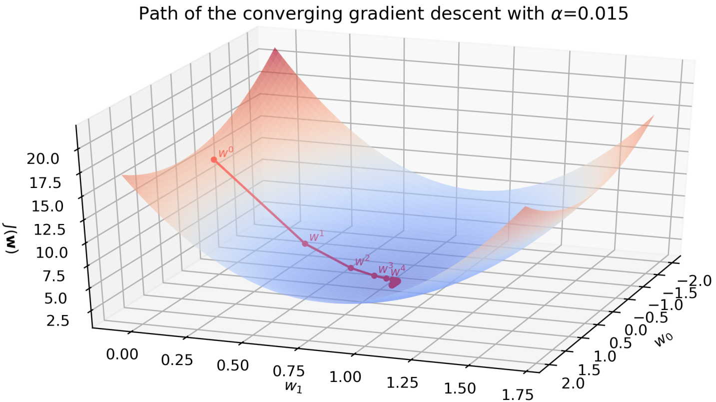
Learning is an optimization problem.

The Error Landscape

- For each set of weights, there's an associated error
- This creates an "error landscape" over weight space
- Valleys represent low error (good predictions)
- Hills represent high error (poor predictions)
- Goal: Find the lowest point in this landscape

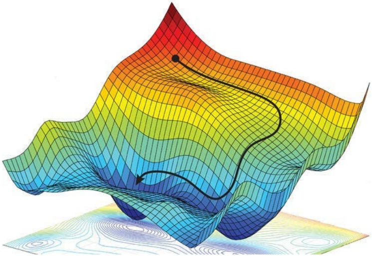


Error Landscape: 2D Visualization



- x-axis: Weight 1
- y-axis: Weight 2
- z-axis (color): Error value
- Blue regions: Low error
- Red regions: High error

Gradient Descent Intuition



- If we're on a slope, which way should we move?
- Answer: In the direction of steepest descent
- This direction is given by the negative gradient
- The gradient points "uphill"
- The negative gradient points "downhill"

Key Insight

Follow the negative gradient to reduce error.

What is a Gradient?

- The gradient is a vector of partial derivatives
- For a function $f(x, y)$, the gradient is:

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \tag{2}$$

- For error function $E(w_1, w_2, \ldots, w_n)$:

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n} \right) \tag{3}$$

- The gradient points in the direction of steepest increase
- The negative gradient points in the direction of steepest decrease

The Gradient

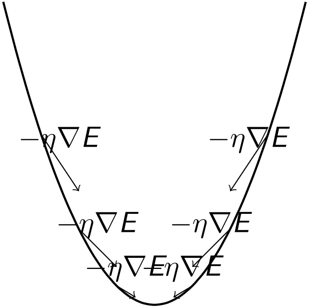
The gradient of the error E with respect to a weight w tells us:

- Direction of steepest increase in error
- How sensitive the error is to changes in w

Mathematically: $\nabla E = \frac{\partial E}{\partial w}$
To minimize error, move in opposite direction:

$$w_{new} = w_{old} - \eta \frac{\partial E}{\partial w} \qquad (4)$$

where η is the learning rate



Simple Gradient Descent Example

Linear regression with one weight:

$$\hat{y} = wx \tag{5}$$

Error function:

$$E = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - wx)^2 \tag{6}$$

Gradient:

$$\frac{\partial E}{\partial w} = \frac{\partial}{\partial w}\frac{1}{2}(y - wx)^2 = -(y - wx)x = -xy + wx^2 \tag{7}$$

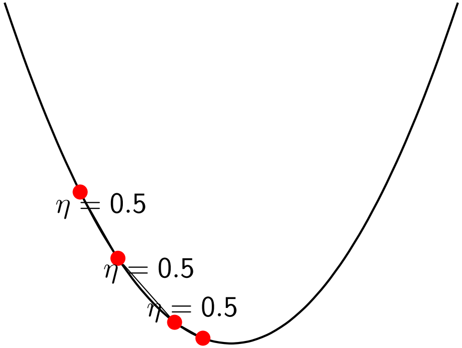
Update rule:

$$w_{new} = w_{old} - \eta(-xy + wx^2) = w_{old} + \eta(xy - wx^2) \tag{8}$$

Interpretation

If prediction is too low, increase w; if too high, decrease w.

Learning Rate



Learning rate too small:

- Very slow convergence
- Many iterations needed
- May get stuck in local minima

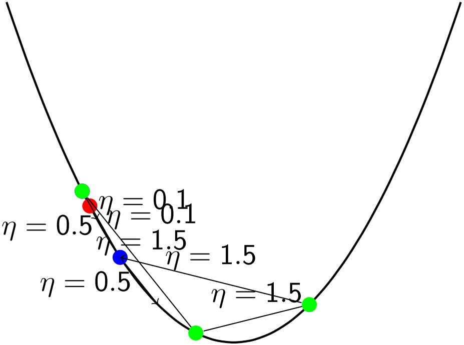
Learning rate too large:

- May overshoot minimum
- Can oscillate or diverge
- Learning becomes unstable

Key Insight:

Finding the right learning rate is crucial for efficient learning.

Learning Rate Comparison



- Red path: Learning rate too small ($\eta = 0.1$)
- Blue path: Good learning rate ($\eta = 0.5$)
- Green path: Learning rate too large ($\eta = 1.5$)