Introduction to Neural Networks

From Brain to Business: How Machines Learn to Predict

Neural Networks for Business Applications

November 24, 2025

Learning Objectives

- **Explain** how biological neurons inspire artificial neural networks
- **Calculate** the output of an artificial neuron given inputs and weights
- **Design** a simple multilayer network architecture
- **Trace** information flow through forward propagation
- **Describe** how networks learn by minimizing prediction errors
- **Evaluate** when neural networks are appropriate for business

The Prediction Challenge: Can We Predict Markets?

The Business Question
Can we predict if a stock price will rise or fall tomorrow?

- Traditional: Statistical analysis, expert intuition
- Challenge: Markets are complex, non-linear
- Many factors: price, volume, sentiment, volatility

The Limitation
Rule-based systems cannot capture all interactions

Why This Matters

- Better investment decisions
- Risk management
- Portfolio optimization
- Automated trading strategies

What We Need
A system that learns patterns from data, not explicit rules

Our journey begins with understanding how nature solved similar prediction problems

What We Need: A Learning System

System Requirements

1. Process multiple inputs simultaneously
2. Learn patterns from historical data
3. Handle **non-linear** relationships
4. Improve predictions over time
5. Generalize to new conditions

Key Insight
We need a system that learns, not one we program

Inspiration from Nature
The human brain solves complex pattern recognition every day

Brain Capabilities

- Processes millions of inputs
- Learns from experience
- Handles ambiguity
- Generalizes to new situations

Can we mimic this for business predictions?

Next: Understanding biological neurons as the foundation

Part 1: Foundations

From biological neurons to artificial intelligence

[1] – [2] – [3] – [4] – [5]

Nature's Computer: How Your Brain Makes Predictions

Biological Neuron Structure

- **Dendrites:** Receive signals
- **Soma:** Integrates weighted signals
- **Axon:** Transmits output
- **Synapses:** Variable connection strengths

Key Principle

Fire when weighted sum exceeds threshold

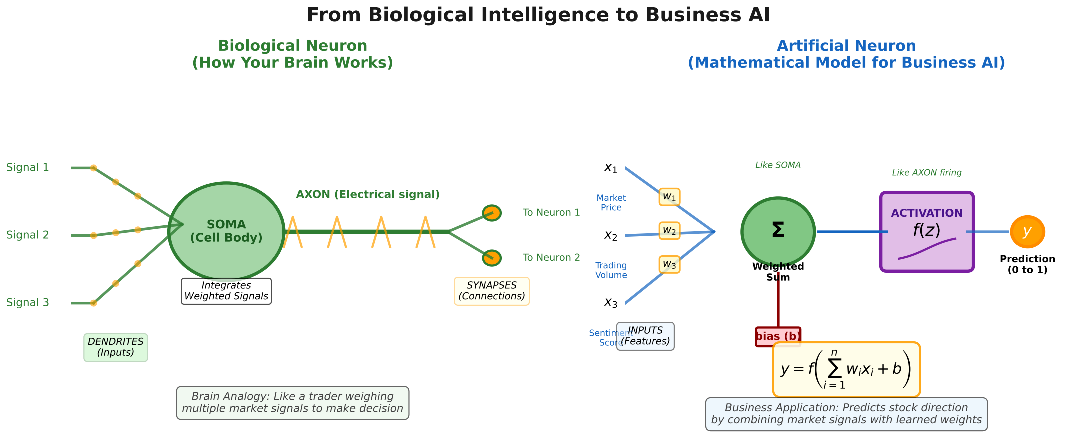
Business AI Insights

1. Multiple inputs combined
2. Weighted connections (importance)
3. Non-linear activation (thresholds)
4. Layered processing (abstraction)

Mathematical models can learn the same way!

Next: See the visual comparison of biological vs artificial neurons

From Biology to Artificial Intelligence



Observe: Which biological components map directly to mathematical operations?

The Artificial Neuron: Mathematical Model

Step 1: Weighted Sum

$$z = \sum_{i=1}^{n} w_i x_i + b$$

- x_i: Inputs (market data)
- w_i: Weights (learned)
- b: Bias (baseline)

Step 2: Activation

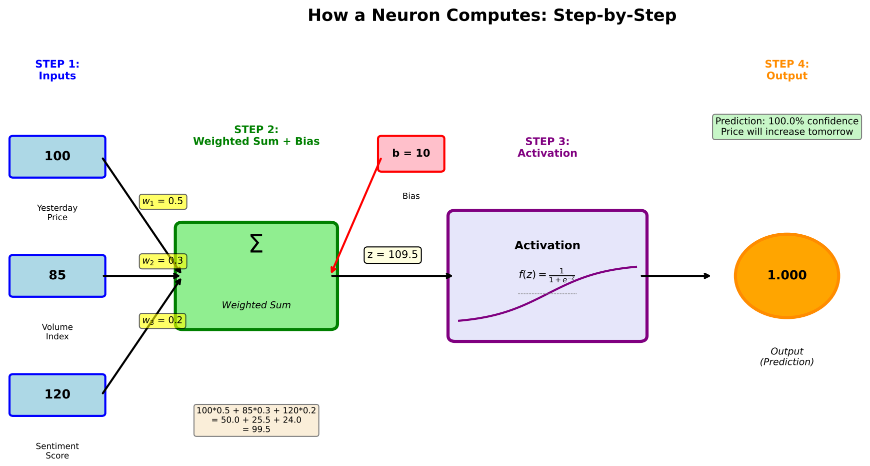
$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$

- Adds non-linearity
- Output: probability (0 to 1)
- Mimics neuron firing

Complete: $y = \sigma\left(\sum w_i x_i + b\right)$

Next: See a concrete example with real market numbers

Single Neuron Computation: Step-by-Step Example



Observe: How would changing the weights affect the final output probability?

Discussion: Part 1 Reflection

Think – Pair – Share

What other business processes might benefit from 'learning from data' instead of following explicit rules?

1. Think (1 min)
Reflect individually on the question

2. Pair (2 min)
Discuss with a neighbor

3. Share (2 min)
Share insights with class

Part 2: Building Blocks

Activation functions and their role in learning

[1] – **[2]** – [3] – [4] – [5]

Activation Functions: Why Non-Linearity Matters

The Problem
Without activation functions:

- Networks = linear regression
- Cannot learn complex patterns

Three Common Functions

- **Sigmoid:** (0,1) for probabilities
- **ReLU:** Fast, efficient
- **Tanh:** Zero-centered (-1,1)

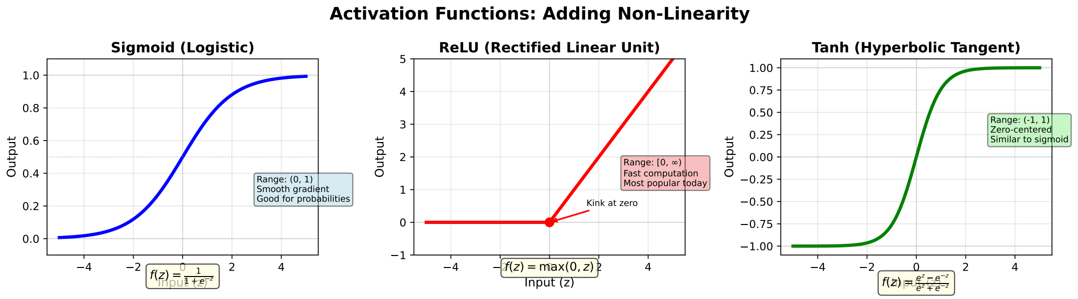
Business Non-Linearity

1. Diminishing returns
2. Threshold effects
3. Saturation points
4. Network effects

Activation functions capture these patterns!

Next: Visual comparison of these three activation functions

Activation Functions: Visual Comparison



Observe: Where does each function's output change most rapidly? Why does this matter?

The Limitation: Why One Neuron Is Not Enough

What One Neuron Can Do

- Single straight decision boundary
- Separate linearly separable patterns
- Simple rules only

Analogy: One rule for decisions

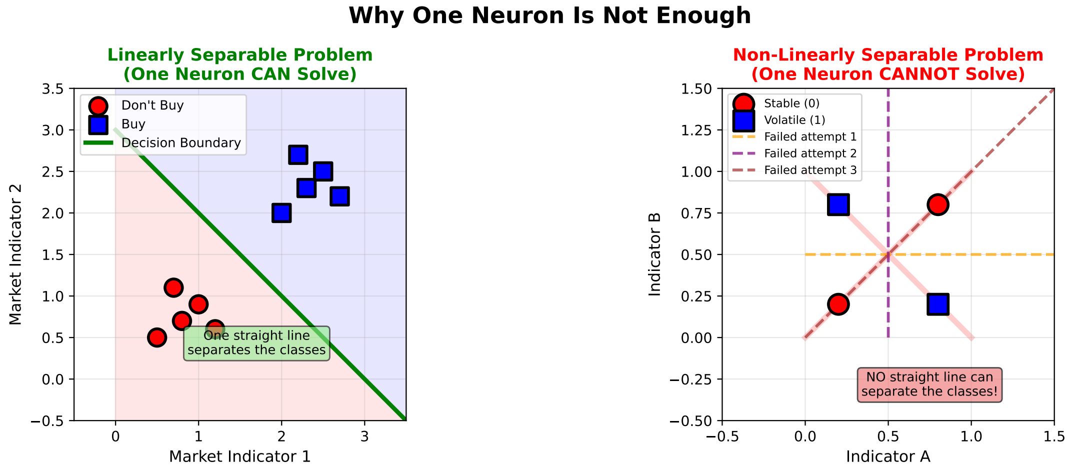
What One Neuron Cannot Do

- Complex, curved boundaries
- XOR-like patterns
- Real-world market interactions

Solution: Multiple Layers!

Next: See the XOR problem that proves one neuron's limitation

Visual Proof: The XOR Problem



Solution: Use Multiple Layers (Hidden Layers) to Create Non-Linear Decision Boundaries

Observe: Why is it impossible to draw a single straight line separating orange from blue?

Discussion: Part 2 Reflection

Think – Pair – Share

Can you think of a business metric that shows diminishing returns or threshold effects?

1. Think (1 min)
Reflect individually on the question

2. Pair (2 min)
Discuss with a neighbor

3. Share (2 min)
Share insights with class

Part 3: Network Architecture

Building layers of intelligence

[1] – [2] – **[3]** – [4] – [5]

Building the Network: Layers of Intelligence

Multi-Layer Architecture

- **Input:** Raw features (no computation)
- **Hidden:** Pattern detection
- **Output:** Final prediction

Result: Buy/Sell decision

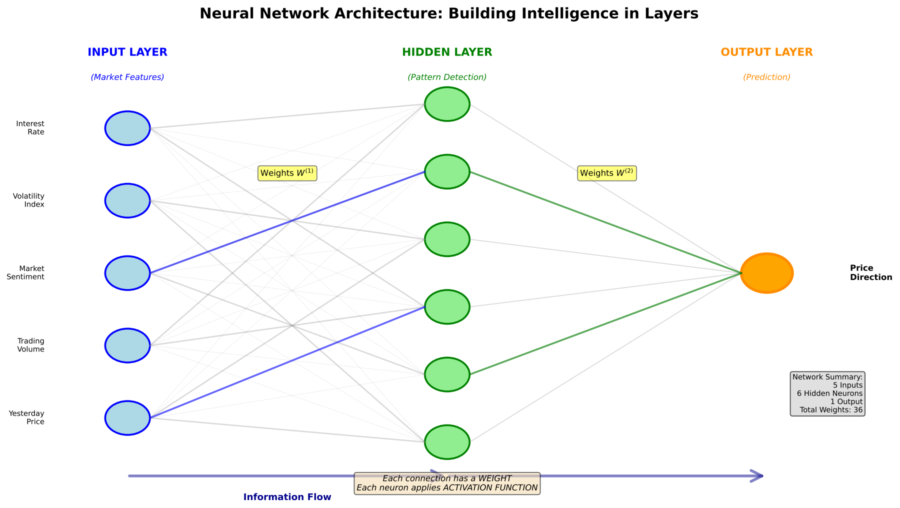
Hierarchical Learning

- Layer 1: Simple patterns
- Layer 2: Complex patterns
- Layer 3: Strategic decisions

Each layer builds on previous abstractions

Next: See the full network architecture with all connections

Neural Network Architecture Diagram



Observe: Count the connections. Why are there 36 weights total?

Forward Propagation: How Networks Make Predictions

The Forward Pass

1. **Input:** Feed market features
2. **Hidden:** $a = \sigma(Wx + b)$
3. **Output:** $y = \sigma(Wa + b)$

All neurons compute in parallel!

Example

Input: price=105.2, volume=0.75

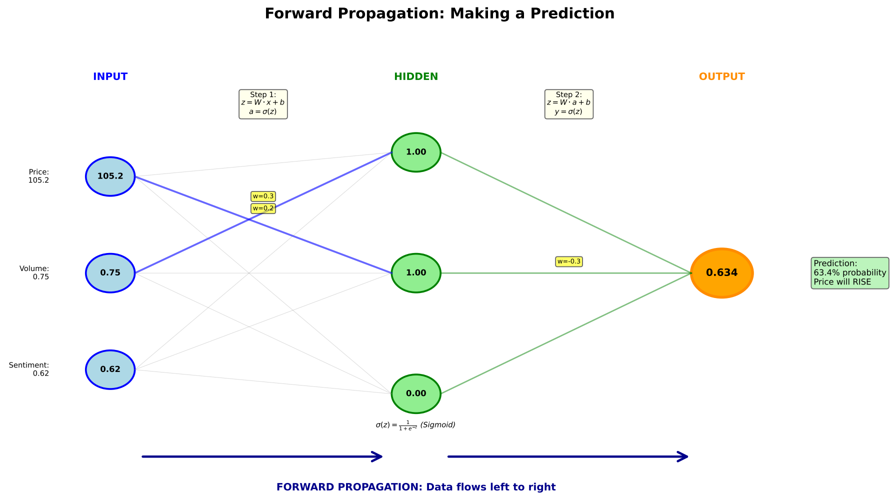
Output: $y = 0.742$

Interpretation:

- 74.2% confidence price rises
- $y > 0.5$: **BUY**
- $y < 0.5$: **SELL**

Next: See forward propagation with actual numbers and calculations

Forward Propagation: Detailed Example


Forward Propagation: Making a Prediction
INPUT
HIDDEN
OUTPUT
Step 1: $z = W \cdot x + b$ $a = \sigma(z)$
Step 2: $z = W \cdot a + b$ $y = \sigma(z)$
Price: 105.2
Volume: 0.75
Sentiment: 0.62
105.2
0.75
0.62
w=0.3
w=0.2
1.00
1.00
0.00
w=-0.3
0.634
Prediction: 63.4% probability Price will RISE
$\sigma(z) = \frac{1}{1+e^{-z}}$ (Sigmoid)
FORWARD PROPAGATION: Data flows left to right

Observe: How do the hidden layer values combine to produce the final 0.742 output?

Discussion: Part 3 Reflection

Think – Pair – Share

For your industry, what would be the 'inputs' and 'outputs' of a useful neural network?

1. Think (1 min)
   Reflect individually on the question

2. Pair (2 min)
   Discuss with a neighbor

3. Share (2 min)
   Share insights with class

Part 4: Learning Process

How networks learn from mistakes

[1] – [2] – [3] – **[4]** – [5]

Learning from Mistakes: The Training Process

Learning Steps

1. **Predict** with random weights
2. **Measure error:**

$$L = \frac{1}{n}\sum(y - \hat{y})^2$$

3. **Adjust weights:**

$$w_{new} = w_{old} - \eta \nabla L$$

4. **Repeat** until convergence

Example

Predicted: 55% rise, Actual: fell
Error: $(0 - 0.55)^2 = 0.30$

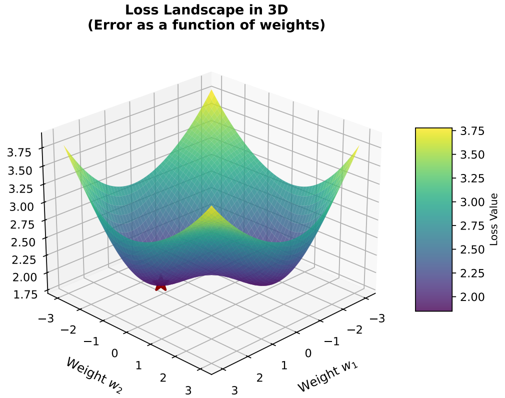
Learning:

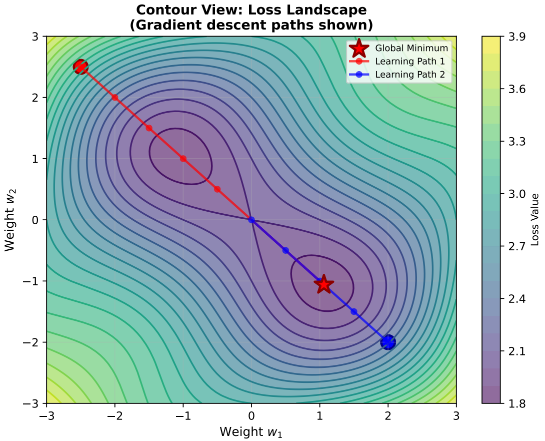
- Calculate gradient direction
- Move weights to reduce error

Like a trader learning from mistakes

Next: Visualize the loss landscape that we're trying to navigate

Loss Landscape: The Error Surface





Goal: Find the weights that minimize the loss
(The red star shows the optimal solution)

Observe: What happens if we start from different random initial weights?

Gradient Descent: Learning by Stepping Downhill

Algorithm

1. Calculate gradient (slope)
2. Step opposite direction
3. Repeat until convergence

Learning Rate Trade-offs

- Too small: Slow
- Too large: Unstable
- Just right: Steady

Business Analogy

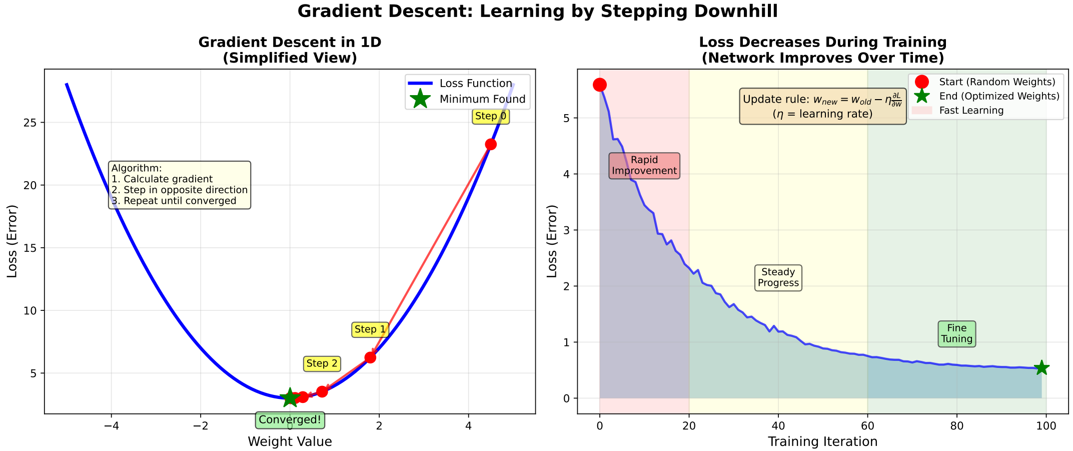
Like a trader learning:

- Fast learning from obvious patterns
- Steady fine-tuning
- Convergence to optimal rules

Gradient shows fastest error reduction

Next: See how loss decreases over training iterations

Gradient Descent: Optimization in Action



Observe: How does the step size (learning rate) affect how quickly we reach the minimum?

Discussion: Part 4 Reflection

Think – Pair – Share

How is gradient descent similar to how businesses optimize through trial and error?

1. Think (1 min)
Reflect individually on the question

2. Pair (2 min)
Discuss with a neighbor

3. Share (2 min)
Share insights with class

Part 5: Application

Putting it all together with market prediction

[1] – [2] – [3] – [4] – **[5]**

Putting It Together: Market Prediction Case Study

Business Application

- **Goal:** Predict price direction
- **Data:** 60 days market data

Input Features

1. Stock Price
2. Trading Volume
3. Market Sentiment
4. Volatility Index

Target Variable

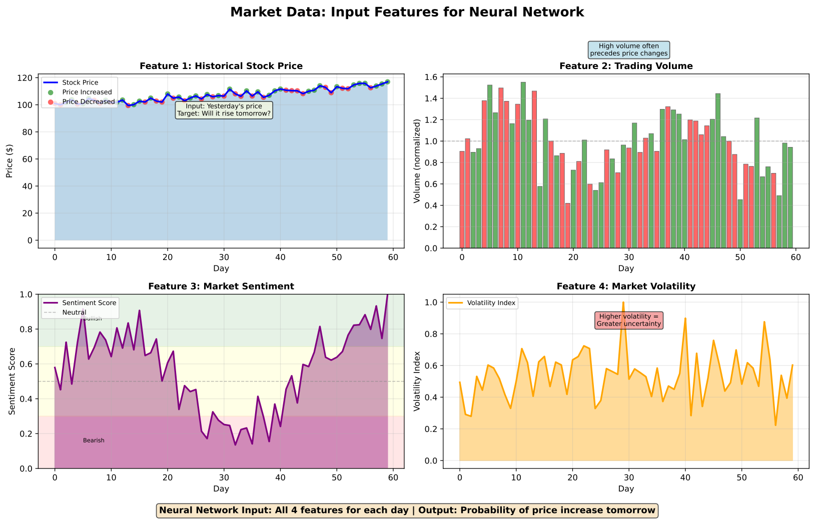
Binary: $1 =$ up, $0 =$ down
Network outputs: $p(\text{rise})$

Setup

- Train: 45 days
- Test: 15 days
- Network: 4-6-1

Next: See the actual market data used for training

Market Data: Input Features for Neural Network



Observe: Which features seem most correlated with the price direction markers?

Training Results: Before vs After

The Experiment

- Before: Random weights (coin flip)
- After: Learned weights
- Test: 30 days unseen data

Results

- Before: 50% accuracy
- After: 70% accuracy
- **Gain:** +20 points

What Network Learned

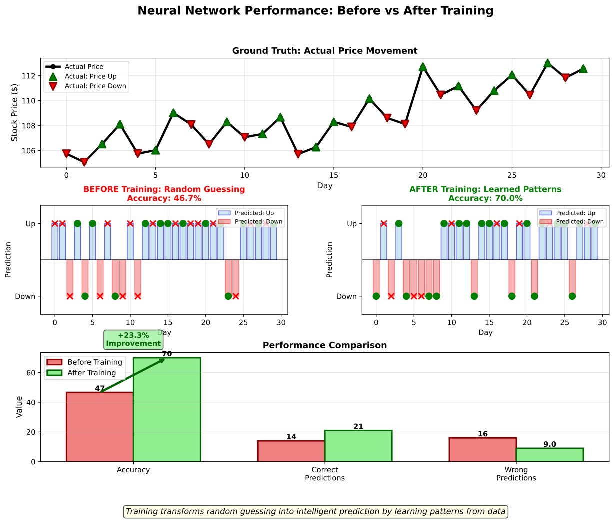
- Volume + price + sentiment patterns
- Volatility indicates uncertainty
- Sentiment confirms trends

Discovered from data alone!

70% is good for markets (100% impossible)

Next: See detailed before/after comparison with prediction accuracy

Prediction Results: Before vs After Training



Observe: Where does the trained model still make errors? What might explain these?

Discussion: Part 5 Reflection

Think – Pair – Share

What data would you need to predict customer behavior in your domain?

1. Think (1 min)	2. Pair (2 min)	3. Share (2 min)
Reflect individually on the question	Discuss with a neighbor	Share insights with class

Summary: From Neurons to Predictions

Our Journey

1. Biological inspiration
2. Mathematical model
3. Activation functions
4. Network architecture
5. Forward propagation

Learning Process

6. Loss functions
7. Gradient descent
8. Training iterations
9. Pattern discovery
10. Improved predictions

Key Takeaway

Neural networks **learn patterns from data** rather than following explicit rules. This enables them to discover complex relationships that would be impossible to program manually.

Each concept was paired with a visualization to reinforce understanding

When to Use Neural Networks

Use Neural Networks When

- Large dataset (thousands+ examples)
- Complex patterns
- Difficult to specify rules
- Pattern recognition tasks
- Black-box acceptable

Applications
Churn, fraud, recommendations, images, NLP

Do NOT Use When

- Small dataset
- Simple relationships
- Need interpretability
- Rules are known
- Real-time constraints

Alternatives
Regression, decision trees, expert systems

Choose the right tool - neural networks are powerful but not always appropriate

Important Limitations & Ethical Responsibilities

Technical Limitations

- Data hungry
- Black box decisions
- Overfitting risk
- No guarantees
- Computational cost

Ethical Concerns

- **Fairness:** Biased data leads to biased predictions
- **Transparency:** GDPR requires explanations
- **Accountability:** Who is responsible?
- **Impact:** Job displacement, market stability

With great predictive power comes great responsibility!

Always consider ethical implications before deploying AI systems

Appendix: Mathematical Details (Backpropagation)

Output Layer Gradient

$$\frac{\partial L}{\partial w^{(2)}} = (\hat{y} - y) \cdot \sigma'(z) \cdot a$$

Hidden Layer Gradient

$$\frac{\partial L}{\partial w^{(1)}} = \delta^{(2)} \cdot w^{(2)} \cdot \sigma'(z^{(1)}) \cdot x$$

Loss Functions
MSE (Regression):

$$L = \frac{1}{n} \sum (y - \hat{y})^2$$

Cross-Entropy (Classification):

$$L = -\sum [y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

Backpropagation efficiently computes how each weight contributed to the error

Appendix: Practical Tips & Resources

Practical Tips

- Start simple (baseline first)
- Feature engineering matters
- Avoid overfitting (validation, dropout)
- Tune hyperparameters
- Monitor training curves

Books
Goodfellow (Deep Learning), Nielsen, Geron

Courses

- Andrew Ng (Coursera)
- Fast.ai
- MIT 6.S191

Tools
PyTorch, TensorFlow, scikit-learn

Practice
Kaggle, Yahoo Finance, UCI Repository

Best way to learn: Build real projects with real data!

Appendix: Practice Problem for Business Students

Design Challenge
You are a data scientist at a retail company.
Problem: Predict customer churn
Data Available:

- Demographics
- Purchase history
- Service interactions
- Website engagement

Your Tasks

1. Design network architecture
2. Select input features
3. Choose activation functions
4. Select loss function
5. Define evaluation metrics
6. Identify ethical concerns
7. Plan stakeholder explanation

Discuss in groups - there's no single right answer!