Introduction to Neural Networks

From Brain to Business: How Machines Learn to Predict

Neural Networks Course

November 23, 2025

Learning Goals

After this lecture, you will be able to:

- Understand the biological inspiration behind neural networks
- Explain how an artificial neuron processes information mathematically
- Describe the architecture of a multilayer neural network
- Understand how networks make predictions (forward propagation)
- Grasp the concept of learning through error minimization
- Apply neural network concepts to business prediction problems

The Prediction Challenge: Can We Predict Markets?

The Business Question:

- Can we predict if a stock price will rise or fall tomorrow?
- Traditional methods: Statistical analysis, expert intuition
- Challenge: Markets are complex, non-linear systems
- Many interacting factors: price history, volume, sentiment, volatility

Why This Matters:

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

What We Need

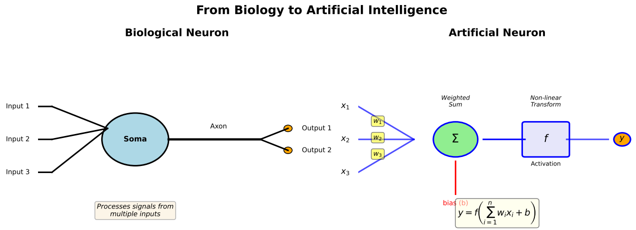
A system that can:

1. Process multiple inputs simultaneously
2. Learn patterns from historical data
3. Handle non-linear relationships
4. Improve predictions over time

Inspiration from Nature

The human brain solves complex pattern recognition tasks every day. Can we mimic this for business predictions?

Nature's Computer: How Your Brain Makes Predictions



The Biological Neuron:

- **Dendrites:** Receive signals from other neurons
- **Soma:** Processes and integrates signals
- **Axon:** Transmits output to other neurons
- **Synapses:** Connection points with varying strengths

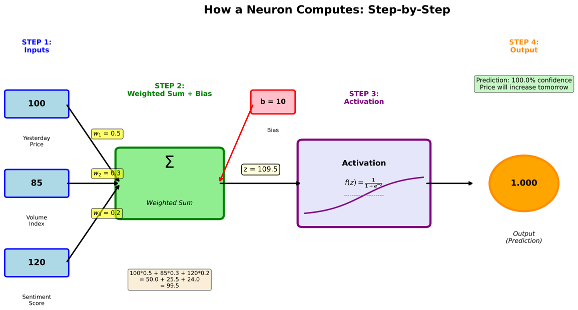
Key Insights:

- Multiple inputs combined
- Weighted connections (some inputs matter more)
- Non-linear activation (threshold behavior)
- Output propagates to next layer

From Biology to Mathematics

We can model this process with equations!

The Artificial Neuron: From Biology to Mathematics



The Mathematical Model:

Step 1: Weighted Sum

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

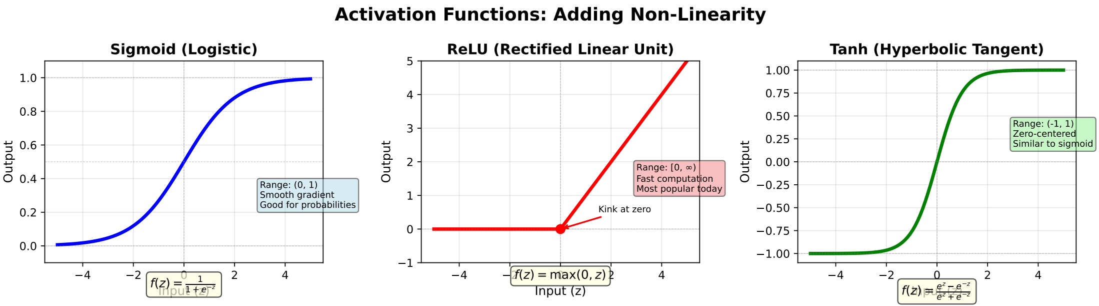
- x_i: Input features
- w_i: Weights (learned)
- b: Bias term

Step 2: Activation

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

- f: Activation function
- Introduces non-linearity
- Output: probability (0 to 1)

Activation Functions: Why Non-Linearity Matters



Sigmoid

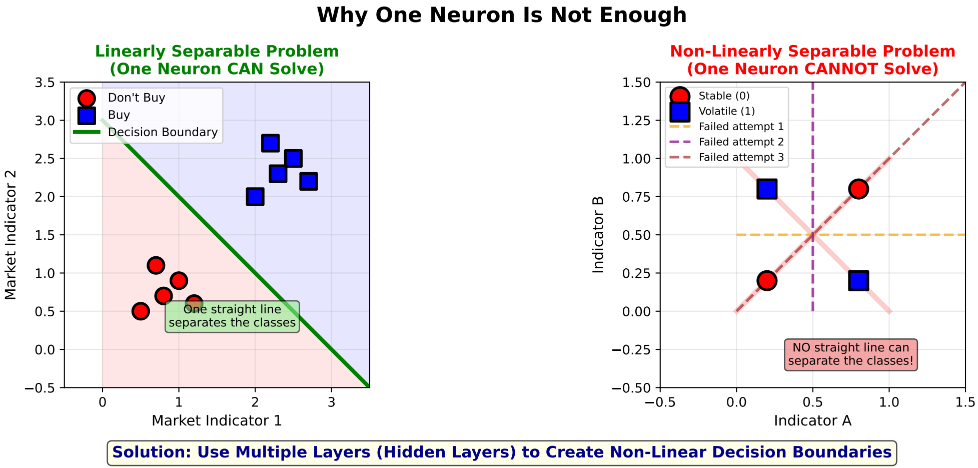
- Smooth, bounded (0,1)
- Good for probabilities
- Used in output layers

ReLU

- Fast to compute
- Most popular today
- Used in hidden layers

Tanh

- Zero-centered (-1,1)
- Stronger gradients
- Alternative to sigmoid



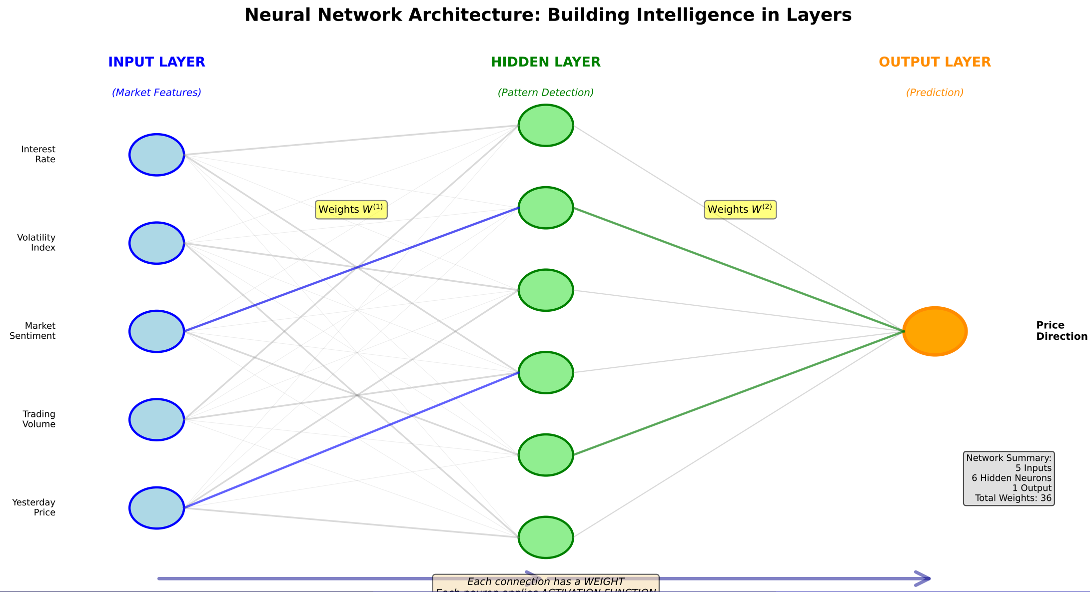
What One Neuron Can Do

- Draw a single straight line (hyperplane)

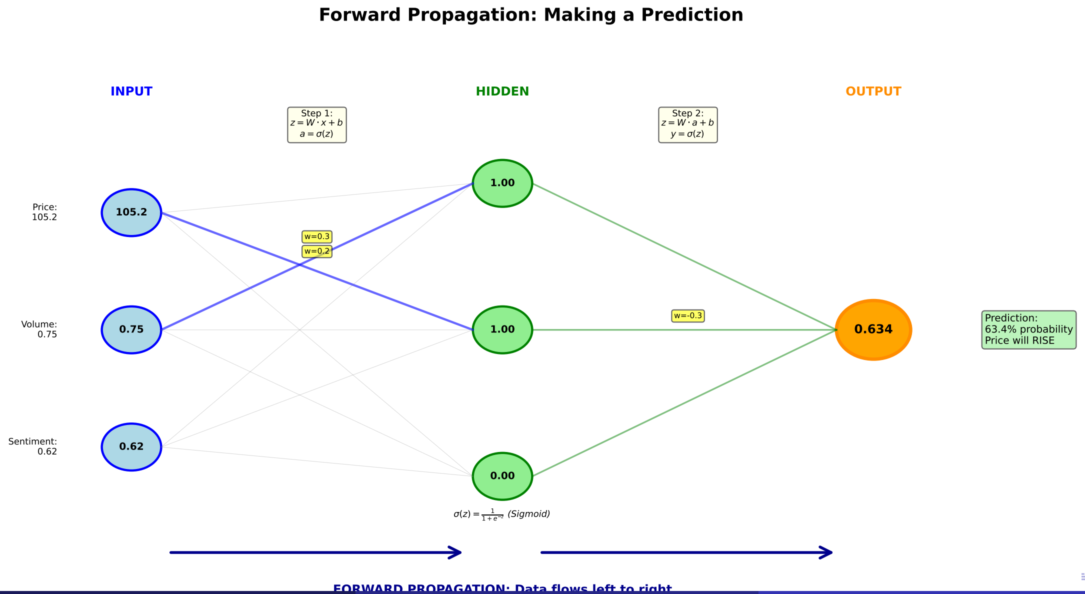
What One Neuron Cannot Do

- Complex, curved decision boundaries

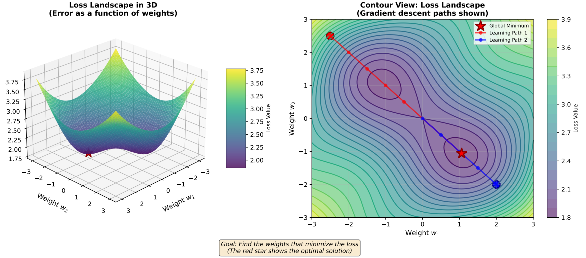
Building the Network: Layers of Intelligence

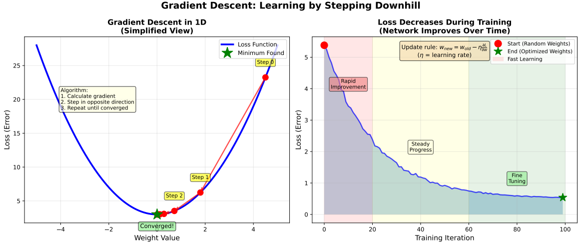


Forward Propagation: How Networks Make Predictions



Learning from Mistakes: How Networks Improve





The Learning Process

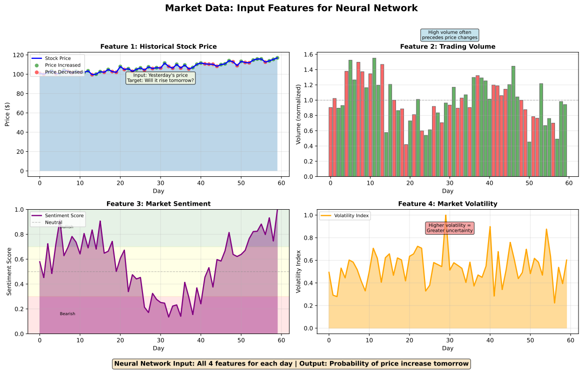
1. **Loss Function:** Measures prediction error

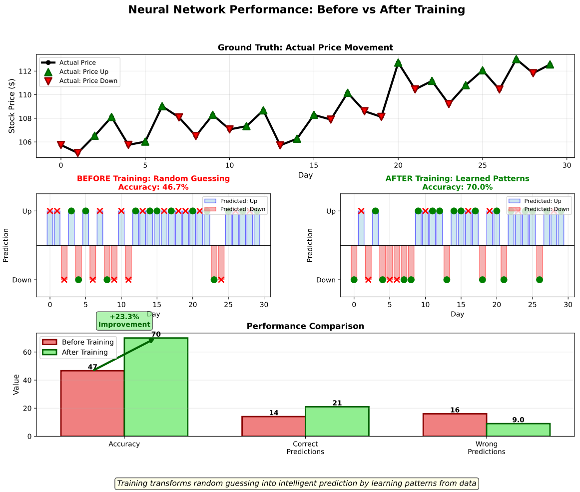
$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2. **Gradient Descent:** Iteratively adjust weights to minimize loss

$$w_{new} = w_{old} - \eta\frac{\partial L}{\partial w}$$

3. **Backpropagation:** Efficiently compute gradients (chain rule)





Key Takeaways

Summary: From Neurons to Predictions

What We Learned:

1. Biological inspiration: neurons in the brain
2. Artificial neurons: mathematical model
3. Activation functions: non-linearity
4. Network architecture: layers of neurons
5. Forward propagation: making predictions
6. Learning: gradient descent & backpropagation
7. Application: market prediction

Important Limitations

- **Data hungry:** Need large datasets
- **Black box:** Hard to interpret
- **Overfitting:** May memorize, not generalize
- **No guarantees:** Markets are unpredictable
- **Computational cost:** Training is expensive

Ethical Considerations

- Fairness: Biased data $\rightarrow$ biased predictions
- Transparency: Explain decisions
- Responsibility: Who is accountable?

Appendix: Mathematical Details

Backpropagation Derivation (Chain Rule):
For a simple 2-layer network, the gradient of the loss with respect to weights:

$$\frac{\partial L}{\partial w^{(2)}} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w^{(2)}}$$

$$\frac{\partial L}{\partial w^{(1)}} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial a^{(1)}} \cdot \frac{\partial a^{(1)}}{\partial z^{(1)}} \cdot \frac{\partial z^{(1)}}{\partial w^{(1)}}$$

Common Loss Functions:

- **Mean Squared Error (Regression):** $L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- **Binary Cross-Entropy (Classification):** $L = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$

Appendix: Further Reading & Resources

Books:

- "Deep Learning" by Goodfellow, Bengio, and Courville (Free online)
- "Neural Networks and Deep Learning" by Michael Nielsen (Free online)
- "Hands-On Machine Learning" by Aurelien Geron (Practical focus)

Online Courses:

- Andrew Ng's Machine Learning Course (Coursera)
- Fast.ai Deep Learning for Coders (Free)
- MIT 6.S191 Introduction to Deep Learning (Free)

Tools & Frameworks:

- TensorFlow & Keras (Python)
- PyTorch (Python)
- scikit-learn (Python, simpler models)

Datasets for Practice:

- Yahoo Finance API (Market data)
- Kaggle Competitions (Various business problems)
- UCI Machine Learning Repository