# Lesson 34: MLPs and Activations

Data Science with Python – BSc Course

45 Minutes

**The Problem:** Single perceptrons can only learn linear boundaries. How do we build networks that learn complex, non-linear patterns?

**After this lesson, you will be able to:**

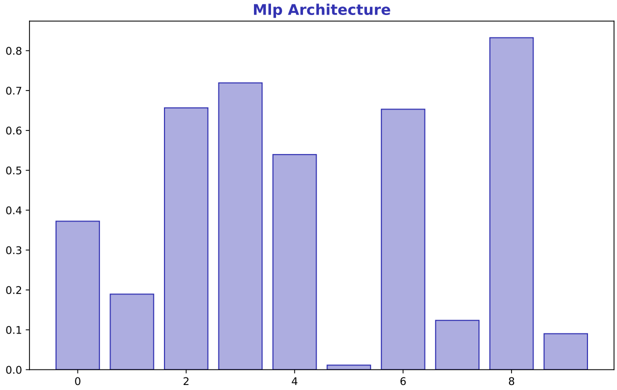
- Design multi-layer perceptron (MLP) architectures
- Choose appropriate activation functions (ReLU, sigmoid, softmax)
- Build neural networks with Keras
- Apply MLPs to non-linear classification problems

**Finance Application: Non-linear regime detection and pattern recognition**

# MLP Architecture

**Adding Hidden Layers**

- Input layer $\rightarrow$ Hidden layer(s) $\rightarrow$ Output layer
- Hidden layers learn intermediate representations

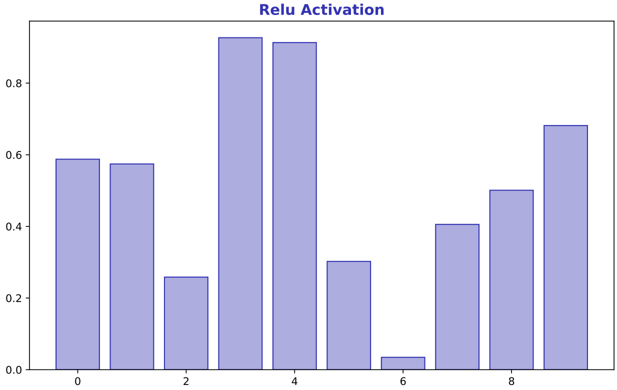



**More layers = more complex patterns, but also more parameters to train**

# ReLU Activation

**The Modern Default**

- $\text{ReLU}(x) = \max(0, x)$ – simple, fast, effective
- Solves vanishing gradient problem of sigmoid

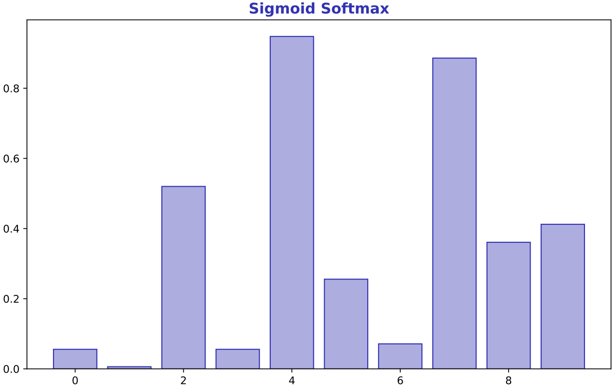



**Use ReLU for hidden layers. It's the default choice in 2024.**

# Sigmoid and Softmax

**Output Layer Activations**

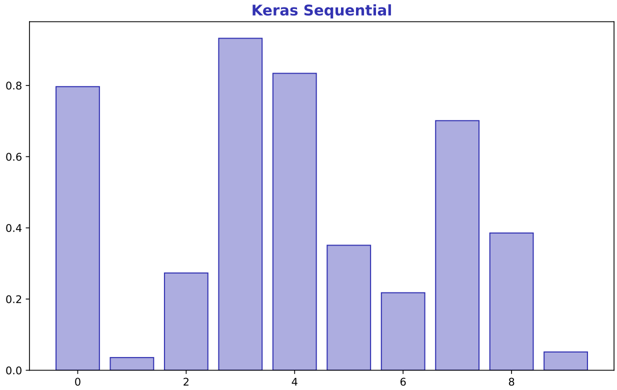
- Sigmoid: binary classification (output in [0,1])
- Softmax: multi-class (outputs sum to 1)




**Rule: ReLU for hidden layers, sigmoid/softmax for output layer**

## Building Networks in Python

- `model = Sequential([Dense(64, activation='relu'), Dense(1, activation='sigmoid')])`
- `model.compile(optimizer='adam', loss='binary_crossentropy')`

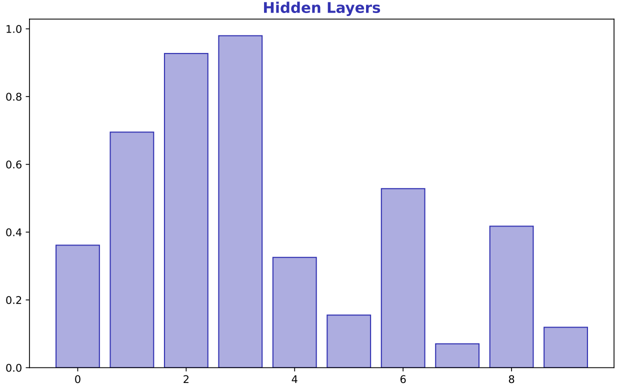



**Keras pattern: add layers sequentially, compile, fit, predict**

# Choosing Hidden Layers

**How Many Neurons? How Many Layers?**

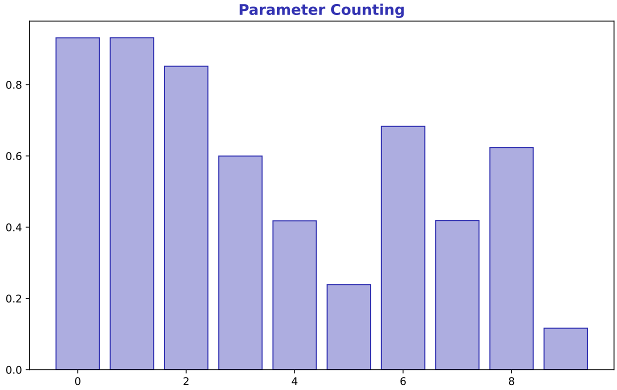
- Start simple: 1-2 hidden layers, 32-128 neurons
- More complex patterns need deeper networks




**Rule of thumb: start small, increase if underfitting**

**How Many Weights to Learn?**

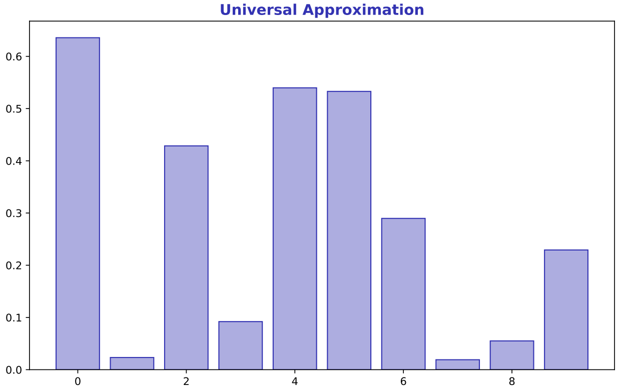
- Dense layer: (inputs $+$ 1) $\times$ outputs parameters
- More parameters $=$ more expressive but slower, prone to overfit




**Check** `model.summary()` **to see parameter count**

## Why Neural Networks Are Powerful

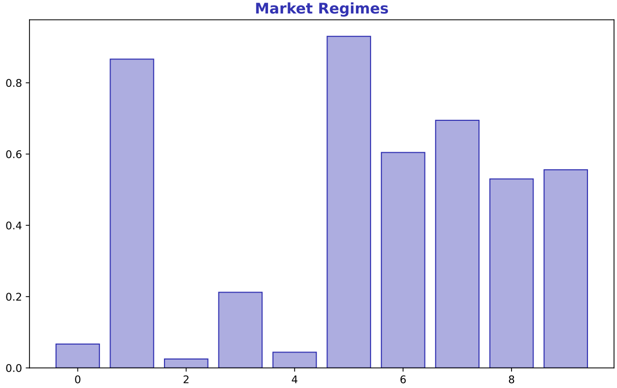
- Theorem: MLP with one hidden layer can approximate any continuous function
- Caveat: may need exponentially many neurons




**Theory says possible; practice says depth often works better than width**

**Finance Application**

- Inputs: volatility, momentum, correlation features
- Output: regime class (bull, bear, sideways)




**MLPs can capture non-linear regime boundaries that linear models miss**

# Hands-On Exercise (25 min)

**Task: Build MLP for XOR and Beyond**

1. Create XOR dataset and verify perceptron fails
2. Build MLP: 2 inputs $\rightarrow$ 4 hidden (ReLU) $\rightarrow$ 1 output (sigmoid)
3. Train and verify 100% accuracy on XOR
4. Visualize decision boundary – observe non-linearity
5. Apply to 3-class classification with softmax output

**Deliverable:** XOR decision boundary plot + 3-class accuracy.

**Extension: Try different hidden layer sizes – how does decision boundary change?**

# Lesson Summary

**Problem Solved:** Multi-layer perceptrons overcome the linear limitation and learn complex patterns.

**Key Takeaways:**

- Hidden layers enable non-linear decision boundaries
- ReLU for hidden layers, sigmoid/softmax for output
- Keras: Sequential API makes building networks easy
- Universal approximation: MLPs can learn any pattern (in theory)

**Next Lesson:** Backpropagation (L35) – how networks actually learn

**Memory: ReLU = max(0,x). Hidden layers = non-linear power. Keras = easy MLPs.**