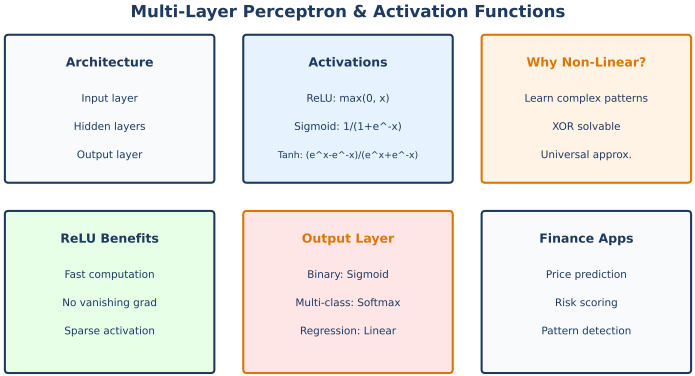
Lesson 34 Summary: MLP & Activations

Data Science with Python – Key Concepts

Data Science Program

MLP & Activations Overview



MLPs use non-linear activations to learn complex patterns

Multi-Layer Perceptron

Network architecture:

- **Input layer:** Receives features
- **Hidden layers:** Learn representations
- **Output layer:** Produces predictions

Each layer transforms data for the next

Why Non-Linear Activations?

The key insight:

- **Linear only:** Collapses to single layer
- **Non-linear:** Enables complex functions
- **Universal approximation:** Can learn any function

Without non-linearity, depth provides no benefit

ReLU: Rectified Linear Unit

Most popular hidden activation:

- **Formula:** $f(x) = \max(0, x)$
- **Benefit:** Fast, no vanishing gradient
- **Issue:** Dead neurons (always 0)

Leaky ReLU fixes dead neuron problem

Sigmoid and Tanh

Classic activations:

- **Sigmoid:** $\sigma(x) = 1/(1 + e^{-x})$ outputs 0-1
- **Tanh:** Outputs -1 to 1, zero-centered
- **Issue:** Vanishing gradients at extremes

Sigmoid still used for binary output layers

Output Layer Activations

Depends on task:

- **Binary classification:** Sigmoid
- **Multi-class:** Softmax (probabilities sum to 1)
- **Regression:** Linear (no activation)

Match activation to your prediction task

Universal Approximation

Theoretical foundation:

- **Theorem:** MLP can approximate any function
- **Requirement:** Sufficient hidden neurons
- **Practice:** Finding right architecture is art

More layers often better than more neurons

Finance Applications

Deep learning in finance:

- **Price prediction:** Regression with linear output
- **Credit scoring:** Classification with sigmoid
- **Portfolio:** Multi-output with softmax

Choose architecture based on prediction task

Quick Reference

Activation Functions:

Function	Range	Use
ReLU	$[0, \infty)$	Hidden layers
Sigmoid	$(0, 1)$	Binary output
Tanh	$(-1, 1)$	Hidden (legacy)
Softmax	$(0, 1)$ sum=1	Multi-class output
Linear	$(-\infty, \infty)$	Regression output

ReLU for hidden, task-specific for output