Introduction to Neural Networks

From Linear Models to Multi-Layer Perceptrons

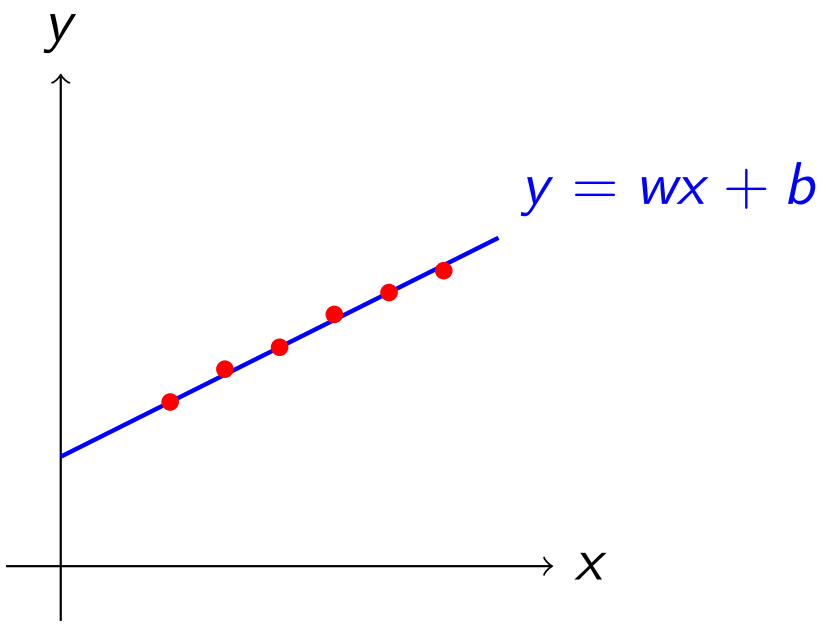
Sayan CHAKI

March 19, 2025

Outline

Linear Regression Recap

- Models a straight-line relationship: $y = wx + b$
- Minimizes the sum of squared errors
- Works well for simple, linearly separable problems

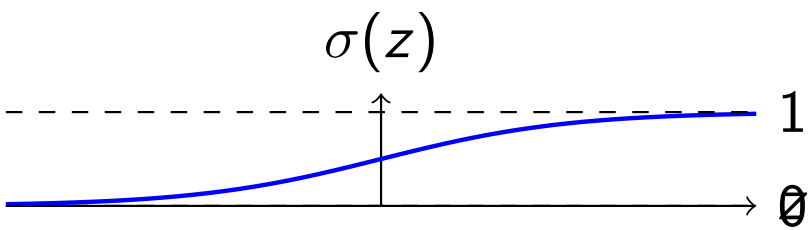


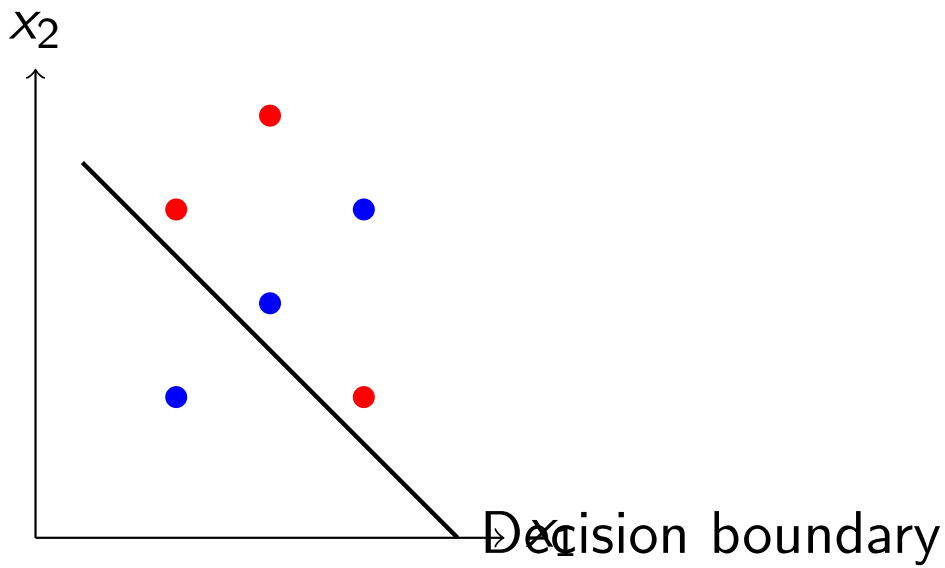
Limitation

Cannot model complex, non-linear relationships

Logistic Regression for Classification

- Uses the sigmoid function to map outputs between 0 and 1:
- $\sigma(z) = \frac{1}{1+e^{-z}}$ where $z = wx + b$
- Used for binary classification problems





Limitation

Cannot handle non-linearly separable problems like XOR

The XOR Problem

x_1	x_2	XOR Output
0	0	0
0	1	1
1	0	1
1	1	0

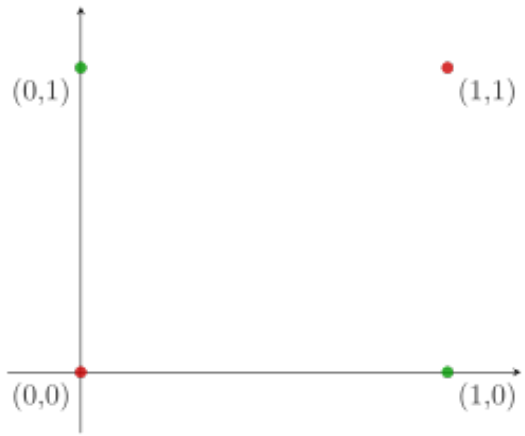


Figure: XOR Problem

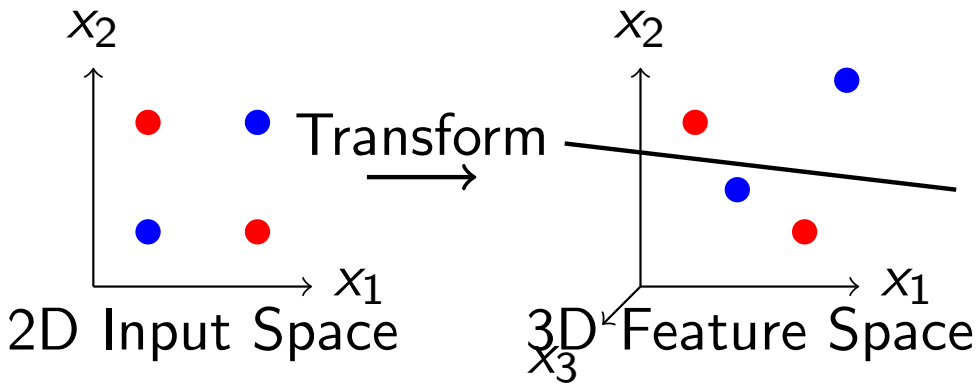
XOR Definition:
Outputs 1 when inputs differ, 0 otherwise

Why logistic regression fails

XOR is not linearly separable - no straight line can separate the blue and red points

Need for a More Complex Model

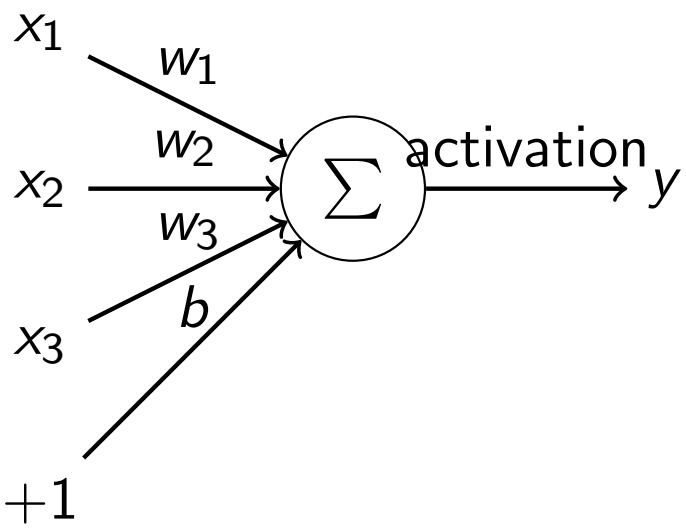
- We need to transform the input space
- One approach: map inputs to higher dimensions where they become linearly separable



Key Idea

We need a model that can learn non-linear decision boundaries

The Perceptron Model



- Inspired by biological neurons
- Takes multiple inputs x_i with weights w_i
- Computes: $z = \sum w_i x_i + b$
- Applies activation: $y = f(z)$

$$\text{Step activation: } f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Limitation

A single perceptron is still a linear classifier and cannot solve XOR

Perceptron as a Regression Model

- Perceptron computes a weighted sum:
- $z = w_1 x_1 + w_2 x_2 + ... + w_n x_n + b$
- This is identical to the linear component in linear/logistic regression
- The activation function determines the output type:
  - Step function $\rightarrow$ binary classification
  - Identity function $\rightarrow$ linear regression
  - Sigmoid function $\rightarrow$ logistic regression

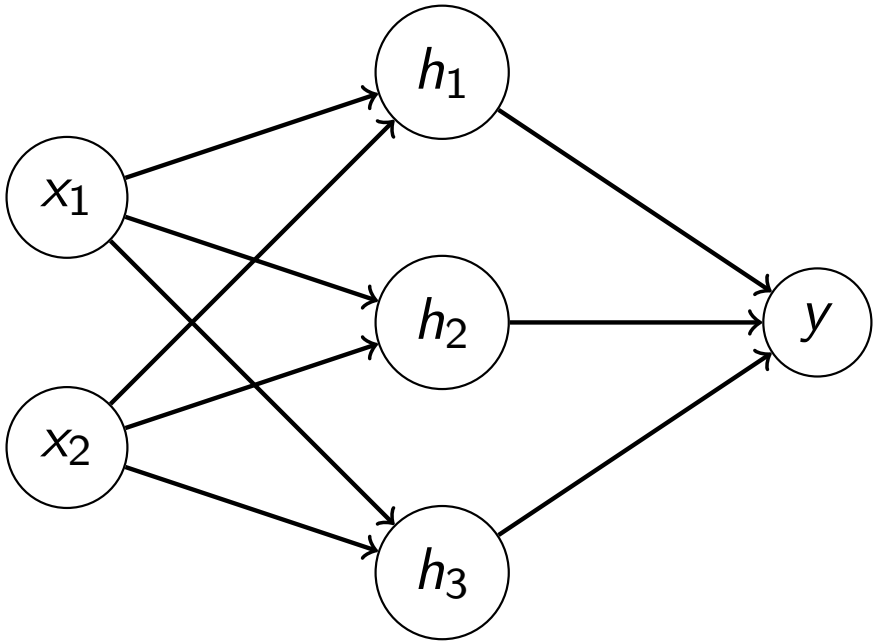
Limitation

The perceptron is still a linear classifier

Need a solution for non-linearity

How can we solve problems like XOR?

Introducing Hidden Layers

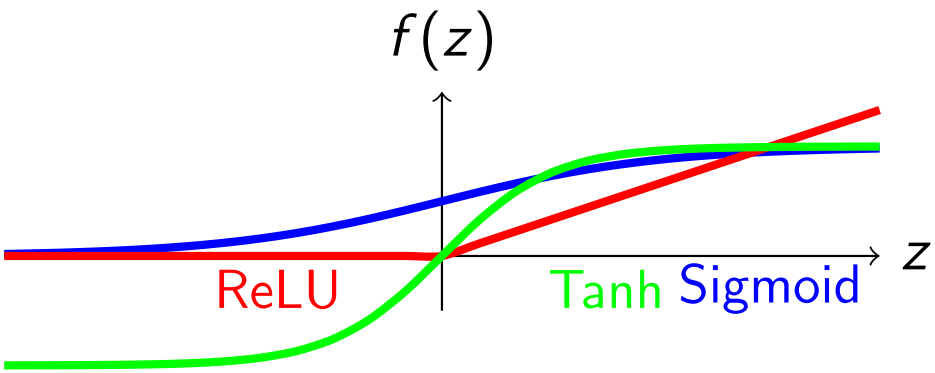


- Hidden layers transform inputs into a new feature space
- Multi-layer networks can approximate any continuous function
- Each hidden neuron creates its own decision boundary
- Combined, they can model complex decision boundaries

Key insight

Hidden layers allow the network to learn feature transformations

The Role of Activation Functions



- **Sigmoid:** $\sigma(z) = \frac{1}{1+e^{-z}}$
  - Outputs between 0 and 1
  - Good for probability estimates
- **ReLU:** $f(z) = \max(0, z)$
  - Simple, computationally efficient
  - Helps with vanishing gradient problem
- **Tanh:** $f(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$
  - Zero-centered outputs (-1 to 1)
  - Often better than sigmoid for hidden layers

Key concept

Non-linear activation functions allow networks to learn complex patterns

Motivation behind activation

Non-linear activation functions are essential for neural networks to learn complex patterns for several important reasons:
1. **Breaking the linearity limitation**: Without non-linear activation functions, neural networks would simply be a series of linear transformations. No matter how many layers you stack, the composition of linear functions is still linear. This means a deep network would be mathematically equivalent to a single-layer network, unable to learn complex data representations.
2. **Creating decision boundaries**: Non-linear activations enable networks to approximate arbitrary decision boundaries. For example, in the XOR problem we discussed, it's impossible to separate the classes with a straight line. Non-linear activations allow the network to create curved or complex boundaries.

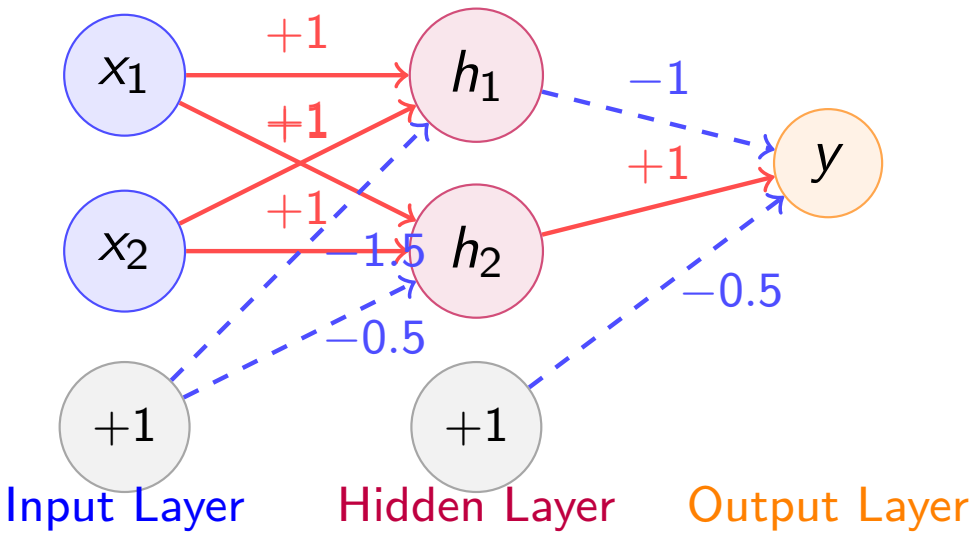
Motivation(contd.)

3. **Feature transformation**: Each layer in a neural network transforms its inputs. With non-linear activations, each neuron can project inputs into different dimensional spaces where previously non-separable patterns become separable. This is essentially what happens in the hidden layers when solving the XOR problem.
4. **Universal approximation**: The universal approximation theorem states that a neural network with just one hidden layer and non-linear activations can approximate any continuous function to arbitrary precision (given enough neurons). This theoretical foundation confirms their pattern-learning capability.

Motiv(contd.)

6. **Biological inspiration**: Neurons in the brain fire in a non-linear fashion - they have thresholds and saturation points. Non-linear activation functions like sigmoid and ReLU capture this biological inspiration, allowing artificial networks to mimic the brain's ability to recognize complex patterns.
To demonstrate this practically: if you tried to train a neural network with linear activations on the XOR problem, it would fail regardless of network depth. However, with non-linear activations like sigmoid or ReLU, even a simple network with one hidden layer can solve XOR by creating the necessary complex decision boundary.

Solving XOR with a Neural Network



- XOR is not linearly separable because no single line can separate $(0, 0)$ and $(1, 1)$ from $(0, 1)$ and $(1, 0)$
- Hidden neurons create two regions:
- h_1 activates when $x_1 + x_2 > 1.5$ (only when both inputs are 1)
- h_2 activates when $x_1 + x_2 > 0.5$ (when at least one input is 1)

Geometric Interpretation

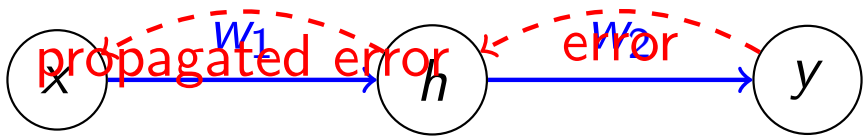
The two hidden neurons create two half-planes in input space. Their intersection forms regions that correspond exactly to the XOR function.

Output activates when: $y = h_2 - h_1 - 0.5 > 0$
This creates the XOR logic: $y = (x_1 \text{ OR } x_2) \text{ AND NOT } (x_1 \text{ AND } x_2)$

x_1	x_2	h_1 (AND)	h_2 (OR)	y (XOR)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Backpropagation: Learning Network Weights

- Problem: How do we learn the optimal weights?
- Solution: Backpropagation algorithm
  1. Forward pass: Calculate outputs for given inputs
  2. Compare with desired outputs: Calculate error
  3. Backward pass: Update weights to minimize error

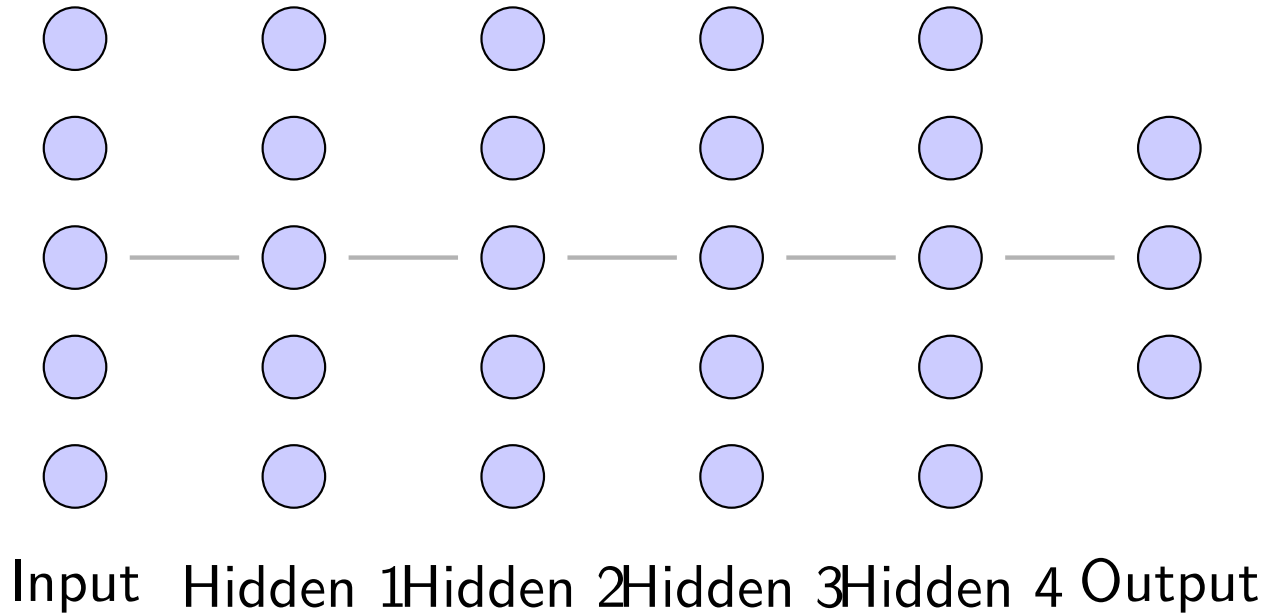


Gradient Descent

- Adjust weights in proportion to their contribution to the error
- Small adjustments iteratively improve network performance

From Simple Networks to Deep Learning

- Modern neural networks have many hidden layers
- Each additional layer allows more complex feature extraction
- Deep networks can learn hierarchical representations:
  - Early layers: Simple features (edges, corners)
  - Middle layers: Component parts (shapes, textures)
  - Later layers: Complex concepts (objects, scenes)



Beyond simple perceptrons

Deep networks have enabled breakthrough performance in image

Summary

- Linear and logistic regression are limited to linear decision boundaries
- The XOR problem illustrates the need for non-linear models
- Perceptrons combine inputs with weights but are still linear classifiers
- Adding hidden layers enables modeling of complex non-linear relationships
- Non-linear activation functions are crucial for network expressiveness
- Backpropagation enables automatic learning of optimal weights
- Deep networks with many layers can learn hierarchical representations

Key takeaway

Neural networks overcome the limitations of linear models by learning complex feature transformations automatically through multiple layers of non-linear processing