Lecture 1.2: Feedforward Networks

Module 1: Foundations of Neural Networks
Deep and Reinforcement Learning

Srikanth Pai

Madras School of Economics

Why Linear Models Are Not Enough

A linear model $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ treats each feature independently.

Business examples where interactions matter:

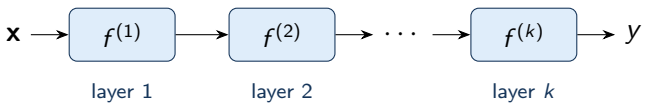
- **Demand** $= f($**price**$)$: fine if demand responds only to price, but what about price $\times$ advertising interaction?
- **Churn** $= f($**days since login, transactions**$)$: a customer who seldom logs in but rarely transacts is at high risk; the individual features mislead.

Three responses:

1. **Kernels**: high-dimensional feature map, risk of overfitting
2. **Feature engineering**: experts hand-craft ϕ, does not scale
3. **Representation learning**: learn ϕ from data

Feedforward Networks: Composing Functions

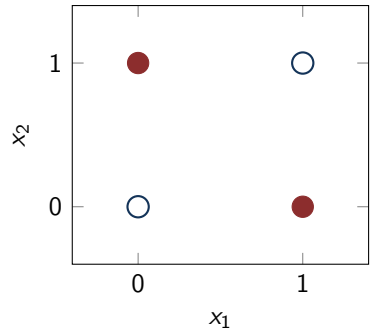
A **feedforward network** computes $f = f^{(k)} \circ f^{(k-1)} \circ \cdots \circ f^{(1)}$



- **Depth** $= k$: number of layers
- **Width** of layer ℓ: dimension of $f^{(\ell)}$'s output
- Layers $1, \ldots, k-1$ are **hidden**; layer k is the **output layer**

The XOR Problem: Can a Linear Model Learn It?

x_1	x_2	$f^*(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0



$\circ$ XOR $= 0$ $\bullet$ XOR $= 1$

Question: Is there w_1, w_2, b such that $f(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$ matches the table?

Classwork: Linear XOR?

Try it (5 minutes)

Suppose $f(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$. Write out the four constraints:

$$f(0,0) = 0, \quad f(0,1) = 1, \quad f(1,0) = 1, \quad f(1,1) = 0.$$

Solve the resulting linear system for w_1, w_2, b. What goes wrong?

[Hint: The first equation pins b; the next two pin w_1 and w_2; then check the fourth.]

No Linear Model Represents XOR

Theorem (Goodfellow et al., Ch. 6.1)

No $f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + b$ satisfies all four XOR constraints.

Proof sketch:

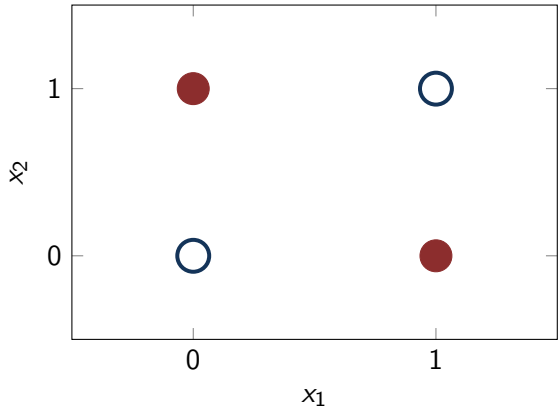
1. $f(0,0) = 0 \implies b = 0$
2. $f(0,1) = 1 \implies w_2 = 1$
3. $f(1,0) = 1 \implies w_1 = 1$
4. $f(1,1) = 0 \implies w_1 + w_2 + b = 2 \neq 0$ **Contradiction!**

What about MSE? Minimising $J = \frac{1}{4}\sum(y_i - \hat{y}_i)^2$ gives:

$$w_1 = w_2 = 0, \quad b = \tfrac{1}{2} \quad \implies \quad \hat{f}(\mathbf{x}) \equiv \tfrac{1}{2} \text{ for all } \mathbf{x}.$$

The optimal linear model predicts the **mean label**, ignoring the input entirely.

No Straight Line Separates XOR

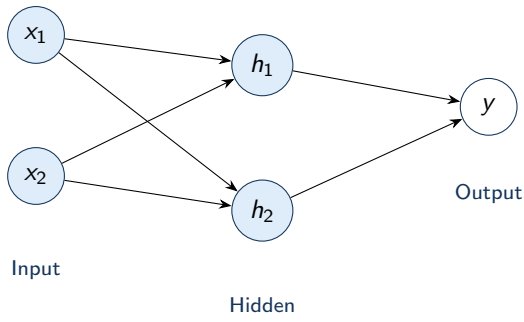


○ XOR = 0 ● XOR = 1

Label-0 at opposite corners; label-1 at the other two. Any separating line would need to cut both diagonals simultaneously: impossible.

The Fix: Transform to a New Feature Space

Idea: introduce a hidden layer $\mathbf{h} = f^{(1)}(\mathbf{x}; W, \mathbf{c})$ and apply a linear output $y = f^{(2)}(\mathbf{h}; \mathbf{w}, b)$.



But: if $h_i = \mathbf{w}_i^\top \mathbf{x} + c_i$ (linear), then $y = \mathbf{w}^\top \mathbf{h} + b$ is still an **affine function of x**.

The Linearity Trap

Proposition

Let $f^{(1)}(\mathbf{x}) = W^\top \mathbf{x} + \mathbf{c}$ and $f^{(2)}(\mathbf{h}) = \mathbf{w}^\top \mathbf{h} + b$. Then:

$$f^{(2)}(f^{(1)}(\mathbf{x})) = \underbrace{(W\mathbf{w})^\top}_{\text{new weight}} \mathbf{x} + \underbrace{(\mathbf{w}^\top \mathbf{c} + b)}_{\text{new bias}}.$$

This is again **linear** in $\mathbf{x}$.

Depth without nonlinearity buys nothing.

Conclusion: the hidden units must apply a **nonlinear activation function**.

Activation Functions

Each hidden unit computes:

$$h_i = g\left(\mathbf{w}_i^\top \mathbf{x} + c_i\right)$$

where $g : \mathbb{R} \to \mathbb{R}$ is **nonlinear**.

Name	Formula	Range
ReLU	$g(z) = \max(0, z)$	$[0, +\infty)$
Sigmoid	$g(z) = \dfrac{1}{1 + e^{-z}}$	$(0, 1)$
Tanh	$g(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}$	$(-1, 1)$

ReLU is the default for hidden layers. Sigmoid and tanh are used as output units for probability estimates.

XOR Solved: The ReLU Network

Parameters: $W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $\mathbf{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, $b = 0$.

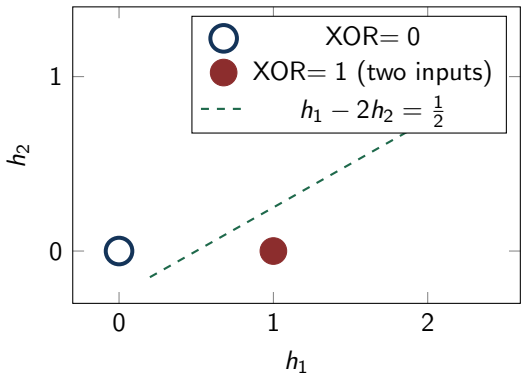
Hidden units: $h_1 = \max(0, x_1 + x_2)$, $h_2 = \max(0, x_1 + x_2 - 1)$. Output: $y = h_1 - 2h_2$.

x_1	x_2	h_1	h_2	$y = h_1 - 2h_2$
0	0	0	0	**0**
0	1	1	0	**1**
1	0	1	0	**1**
1	1	2	1	**0**

Perfect. The network learned to represent XOR exactly.

The Learned Feature Space

In h-space, the XOR inputs are linearly separable.



Inputs $(0, 1)$ and $(1, 0)$ both map to $\mathbf{h} = (1, 0)$: the network identified them as equivalent.

Architecture Design: Three Key Choices

1. **Architecture (depth and width):** deeper networks build increasingly abstract features; depth matters more than width for most tasks.

2. **Output unit and loss function:**
   - Linear output + MSE for regression
   - Sigmoid output + binary cross-entropy for classification
   - Softmax output + categorical cross-entropy for multiclass

3. **Optimiser:** gradient descent + backpropagation computes the required gradients through the composition of layers (Lecture 1.3).

Classwork: Verify and Explore

Try it (5 minutes)

1. Using the parameters $W, \mathbf{c}, \mathbf{w}, b$ from the previous slide, **compute h_1, h_2, and y for input $(1, 1)$ step by step**. Confirm $y = 0$.
2. (If time permits) What happens if you change c_2 from -1 to 0? Recompute the output for all four inputs. Which entries are now wrong?

Key Takeaways

- **Linear models cannot represent XOR**: the constraints are inconsistent; MSE gives a useless constant predictor.
- **Linear composed with linear is linear**: any stack of affine layers is equivalent to a single affine map, regardless of depth.
- **Nonlinear activations** (ReLU, sigmoid, tanh) break the linearity trap and give feedforward networks their expressive power.
- **Feature transformation:** the hidden layer maps inputs to a new representation where the problem becomes linearly separable.
- **Universal Approximation Theorem:** a single hidden layer of sufficient width can approximate any continuous function, though deep networks generalise better in practice.