AI Research Labs

Section 2 of 11 · 3 lessons

The Neuron and Linear Separability

What a single artificial neuron computes, what its decision boundary looks like, and why it cannot solve XOR.

2.1 The Artificial Neuron

2.2 Decision Boundaries and Linear Separability

2.3 The XOR Problem



Free course with quizzes, progress tracking and Scoopy, an AI tutor:
airesearchlabs.si/learn/dl

Deep Learning Foundations · Edition November 2026 · © 2026 AI Research Labs. Free for personal study.

Where this section fits

Every lesson is a step in building a model. These are the steps this section teaches; the Big picture PDF and the online map show how they connect to the rest.

M1

BUILDING AND TRAINING A NETWORK · DENSE LANE · STEP M1

Understand the neuron

What can one neuron do, and what can it never do?

Lessons here: 2.1, 2.2, 2.3 · feeds into Stack layers

The Artificial Neuron

A neuron computes a weighted sum of its inputs, adds a bias and applies an activation function. What each part does, a worked example, and why the bias is just a weight on a constant input.

What a neuron computes

$$y = f\left(w^\top x + b\right) = f\left(\sum_i w_i x_i + b\right)$$

- x: the input vector (the features).
- w: the **weights**, one per input.
- b: the **bias**.
- $z = w^\top x + b$: the **pre-activation**, a single number.
- f: the **activation function**, which turns z into the output y.

So a neuron does two things: a weighted vote of its inputs, then a (usually non-linear) squashing of that vote.

What weights and bias mean

A **positive weight** means "more of this input pushes the output up"; a **negative weight** pushes it down; a weight near zero means the input barely matters. The **bias** shifts the threshold independently of the inputs: a large negative bias makes the neuron hard to switch on.

The bias can be treated as one more weight, $w_0 = b$, on a constant input $x_0 = 1$. Then $z = \sum_{i=0}^{n} w_i x_i$, which is why some books write the neuron without a separate bias.

WORKED EXAMPLE

$x = (2, -1)$, $w = (0.5, 1.5)$, $b = -0.2$.

$z = 0.5(2) + 1.5(-1) - 0.2 = 1 - 1.5 - 0.2 = -0.7$.

- Sigmoid: $y = \sigma(-0.7) = 1/(1 + e^{0.7}) \approx 0.332$.
- ReLU: $y = \max(0, -0.7) = 0$.
- Sign: $y = -1$.

Same pre-activation, three different outputs: the activation function decides how the neuron responds.

Common doubts

FIRST PASS **Why do we need a bias at all?**

Without a bias, $z = w^\top x$ is always 0 when $x = 0$, so the decision boundary $w^\top x = 0$ must pass through the origin. The bias lets the boundary sit anywhere. In one dimension it's the difference between a line through the origin, $y = mx$, and any line, $y = mx + c$.

FIRST PASS **Is it still a neuron if it has no activation function?**

Yes. A neuron with no activation is said to use the **identity** (or linear) activation, $f(z) = z$: it outputs its weighted sum plus bias unchanged. Output neurons for regression are usually like this. Linear regression is exactly one such neuron.

GOING DEEPER **Can a neuron have more than one activation function?**

In the standard model, no: a neuron computes one weighted sum and applies one activation to it. If you applied two functions one after the other, that composition is just a single, different activation function. Different neurons can use different activations, though, and a layer's output can go through extra operations such as softmax or normalisation.

GOING DEEPER **Can a neuron contain another neuron inside it?**

No. A neuron is the smallest unit: a weighted sum, a bias and an activation. Structure comes from **connecting** neurons: the outputs of one layer become the inputs of the next. When a "neuron" seems to hide something more complex, it is really a small network of neurons.

Cheatsheet

The neuron

$y = f(w^\top x + b)$. Pre-activation $z = w^\top x + b$; bias = weight on a constant input 1.

Online version with the interactive quiz: airesearchlabs.si/learn/dl/the-neuron/the-artificial-neuron

Decision Boundaries and Linear Separability

A neuron with a threshold activation splits the input space with a straight line (a hyperplane). Which problems that can solve: AND and OR worked through, and the definition of linear separability.

The decision boundary

With a sign activation, the neuron outputs $+1$ when $z = w^\top x + b > 0$ and -1 when $z < 0$. The **decision boundary** is where $z = 0$:

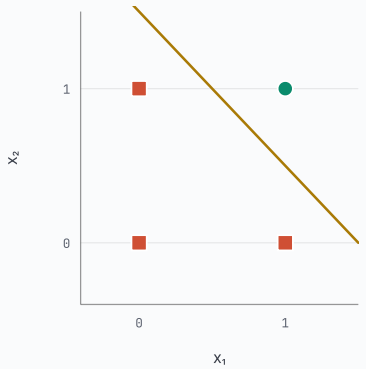
$$w^\top x + b = 0$$

In 2 dimensions this is a line, in 3 a plane, in d a **hyperplane**. The weight vector w is perpendicular (normal) to it and points towards the side classified $+1$; the bias moves it away from the origin.

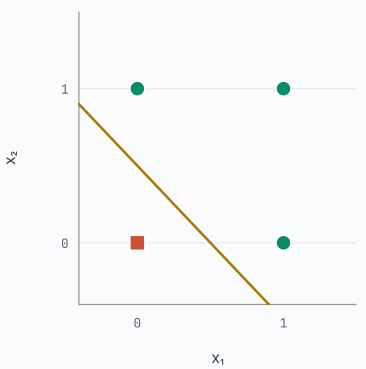
Linear separability

Two classes are **linearly separable** if some w and b put every positive example on one side and every negative example on the other: $w^\top x + b > 0$ for all positives and < 0 for all negatives. A single neuron can only solve linearly separable problems.

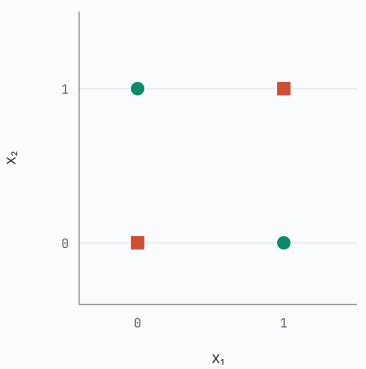
AND and OR are separable



AND: w = (1, 1), b = −1.5



OR: w = (1, 1), b = −0.5



XOR: no single line works

(X_1, X_2)	AND: $X_1 + X_2 - 1.5$	OR: $X_1 + X_2 - 0.5$
(0, 0)	−1.5 → −1	−0.5 → −1
(1, 0)	−0.5 → −1	0.5 → +1
(0, 1)	−0.5 → −1	0.5 → +1
(1, 1)	0.5 → +1	1.5 → +1

Same weights, different bias: the bias slides the line, turning AND into OR. XOR (true when exactly one input is 1) puts the two positives on opposite corners, and no line separates them. The next lesson proves it.

EXAM PHRASING

"A dataset is linearly separable if a hyperplane can separate the classes without error. AND and OR are linearly separable; XOR is not."

Common doubts

FIRST PASS **Why is w perpendicular to the boundary?**

Take two points x_a and x_b on the boundary: $w^\top x_a + b = 0$ and $w^\top x_b + b = 0$. Subtracting gives $w^\top (x_a - x_b) = 0$, so w is at right angles to every direction along the boundary. Moving in the direction of w increases z fastest, which is why w points to the positive side.

Cheatsheet

Decision boundary

$w^\top x + b = 0$: a line (2-D), plane (3-D) or hyperplane. w is normal to it and points to the positive side.

Separable or not

AND, OR: linearly separable (one neuron works).
XOR: not separable (needs a hidden layer).

Online version with the interactive quiz: airesearchlabs.si/learn/dl/the-neuron/decision-boundaries-and-linear-separability

The XOR Problem

A proof that no single neuron can compute XOR, what happens when you try to train one anyway, and two ways out: hand-made features or a hidden layer.

The impossibility proof

XOR is positive for $(1, 0)$ and $(0, 1)$ and negative for $(0, 0)$ and $(1, 1)$. Suppose one neuron with weights w_1, w_2 and bias b classified all four correctly. Then:

INPUT	CLASS	REQUIREMENT
(0, 0)	negative	$b < 0$
(1, 1)	negative	$w_1 + w_2 + b < 0$
(1, 0)	positive	$w_1 + b > 0$
(0, 1)	positive	$w_2 + b > 0$

- Adding the two **positive** rows: $w_1 + w_2 + 2b > 0$.
- Adding the two **negative** rows: $w_1 + w_2 + 2b < 0$.

The same number can't be both positive and negative, so no such w_1, w_2, b exist. **No single neuron with a linear decision boundary can compute XOR.**

> WHY IT MATTERS
>
> This is a proof of impossibility, not a failed training run. No amount of training, data or learning-rate tuning will make one neuron solve XOR.

What training does anyway

Train a single sigmoid neuron on XOR with mean squared error and it gets stuck: every output settles near 0.5 and the loss at 0.25. With targets 0 and 1, predicting 0.5 everywhere gives $\frac{1}{4}\sum(0.5 - y)^2 = 0.25$: the best a straight boundary can do is refuse to decide.

Way out 1: hand-made features

XOR can be written as $x_1 + x_2 - 2x_1x_2$. If you add the feature x_1x_2 as a third input, XOR is **linear** in (x_1, x_2, x_1x_2) and one neuron can solve it. But this needs someone to design the right feature.

Way out 2: a hidden layer

A network with a **hidden layer** can learn a useful representation itself. Section 3 builds an explicit 2–2–1 network for XOR: its hidden units compute OR and AND, and the output combines them. That is the core idea of deep learning: layers learn the features.

Common doubts

FIRST PASS **Is XOR unsolvable because we did not train long enough?**

No. The inequalities show that **no** weights work, so this is about what one neuron can represent, not about training. A failed training run could be bad luck; a contradiction in the inequalities is a proof.

GOING DEEPER **Why does the trained neuron output exactly 0.5 for every input?**

With MSE and targets 0 and 1, each pair of opposite corners pulls the boundary in opposite directions and they cancel. The point that minimises the squared error when no boundary can separate the classes is to predict the mean target, 0.5, everywhere, giving loss 0.25.

Cheatsheet

XOR proof in one line

Positives give $w_1 + w_2 + 2b > 0$; negatives give $w_1 + w_2 + 2b < 0$. Contradiction, so one neuron cannot solve XOR.

Two ways to solve XOR

Add the feature $x_1 x_2$ (XOR $= x_1 + x_2 - 2x_1 x_2$), or add a hidden layer that learns the representation.

Online version with the interactive quiz: airesearchlabs.si/learn/dl/the-neuron/the-xor-problem