Linear Separators

Perceptron:

$w \cdot x > t$ for positive inputs

$w \cdot x < t$ for negative inputs

$wx - t > 0$

$w \cdot x - t < 0$

Write b for $-t$

"bias" vs threshold

$wx + b > 0$

$wx + b < 0$

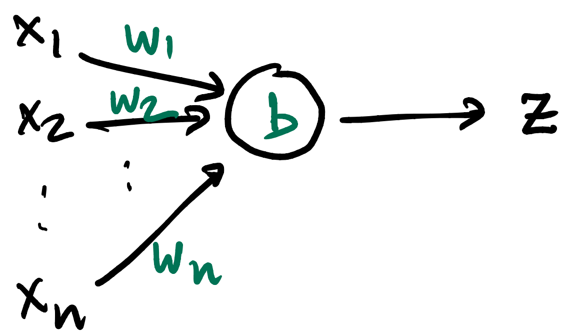
Obvious problem: limited to linearly separable data

Solution: geometric transformation

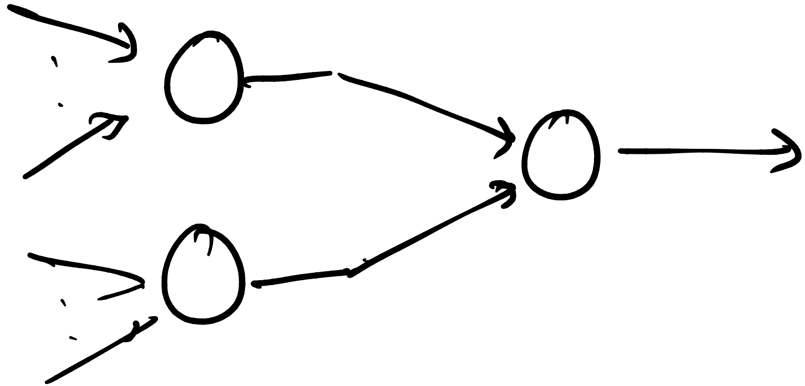
Implicitly using kernel functions

Alternative: Cascade perceptrons

$$wx + b > 0$$



Network of perceptron



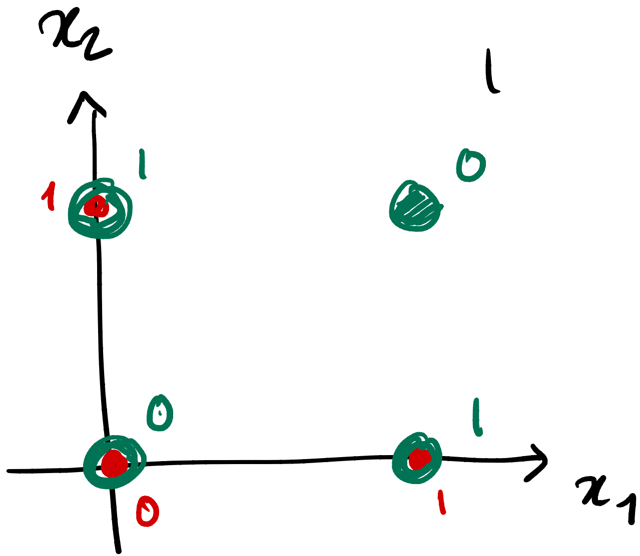
Does this help improve expressiveness?

Yes, but ...

Output is still a (complex) linear function

XOR function

x_1	x_2	$x_1 \oplus x_2$	$x_1 \vee x_2$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1



$x_1 = 0$, output increases with x_2

$x_1 = 1$, output decreases with x_2

But w_2 is fixed!

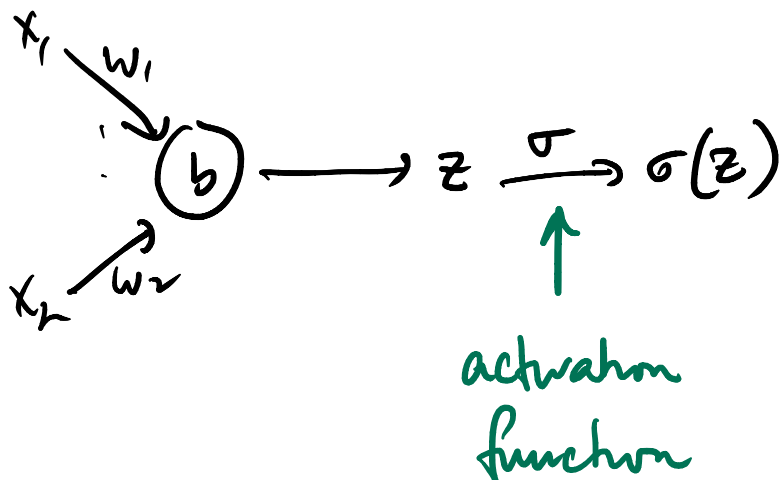
Minsky & Papert $\longrightarrow$ first "AI winter"

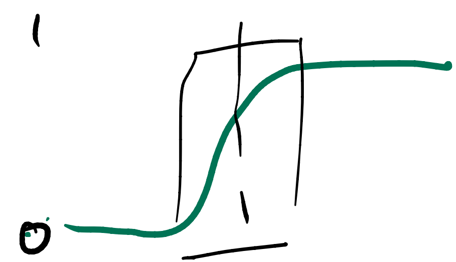
But – introduce non-linearity

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$z \to \infty$, $\sigma(z) \to 1$

$z \to -\infty$, $\sigma(z) \to 0$

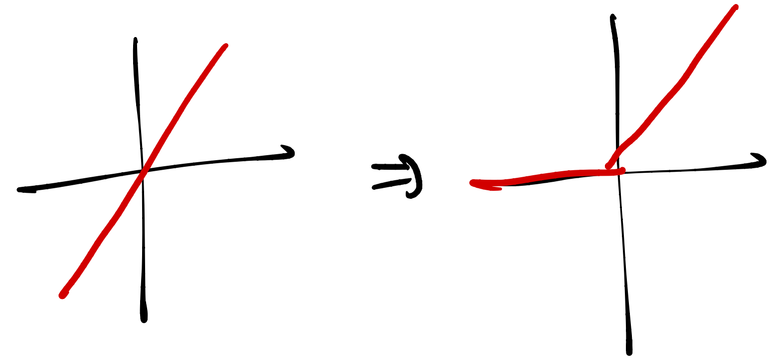




Other nonlinear activation.

RELU

Rectified Linear Unit

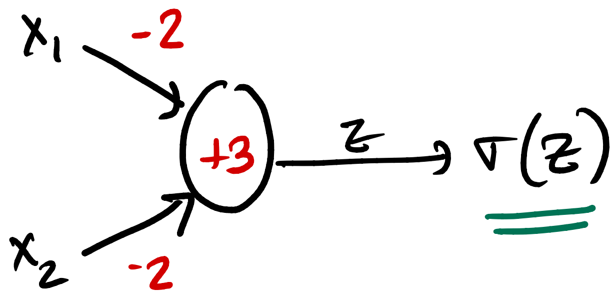


Softmax

⋮

Networks of perceptrons with non-linear activation function — (Artificial) Neural Network

Neural Networks



$$-2x_1 - 2x_2 + 3 > 0$$

x_1	x_2	output
0	0	$3 \Rightarrow 1$
0	1	$1 \Rightarrow 1$
1	0	$1 \Rightarrow 1$
1	1	$-1 \Rightarrow 0$

NAND alone is universal

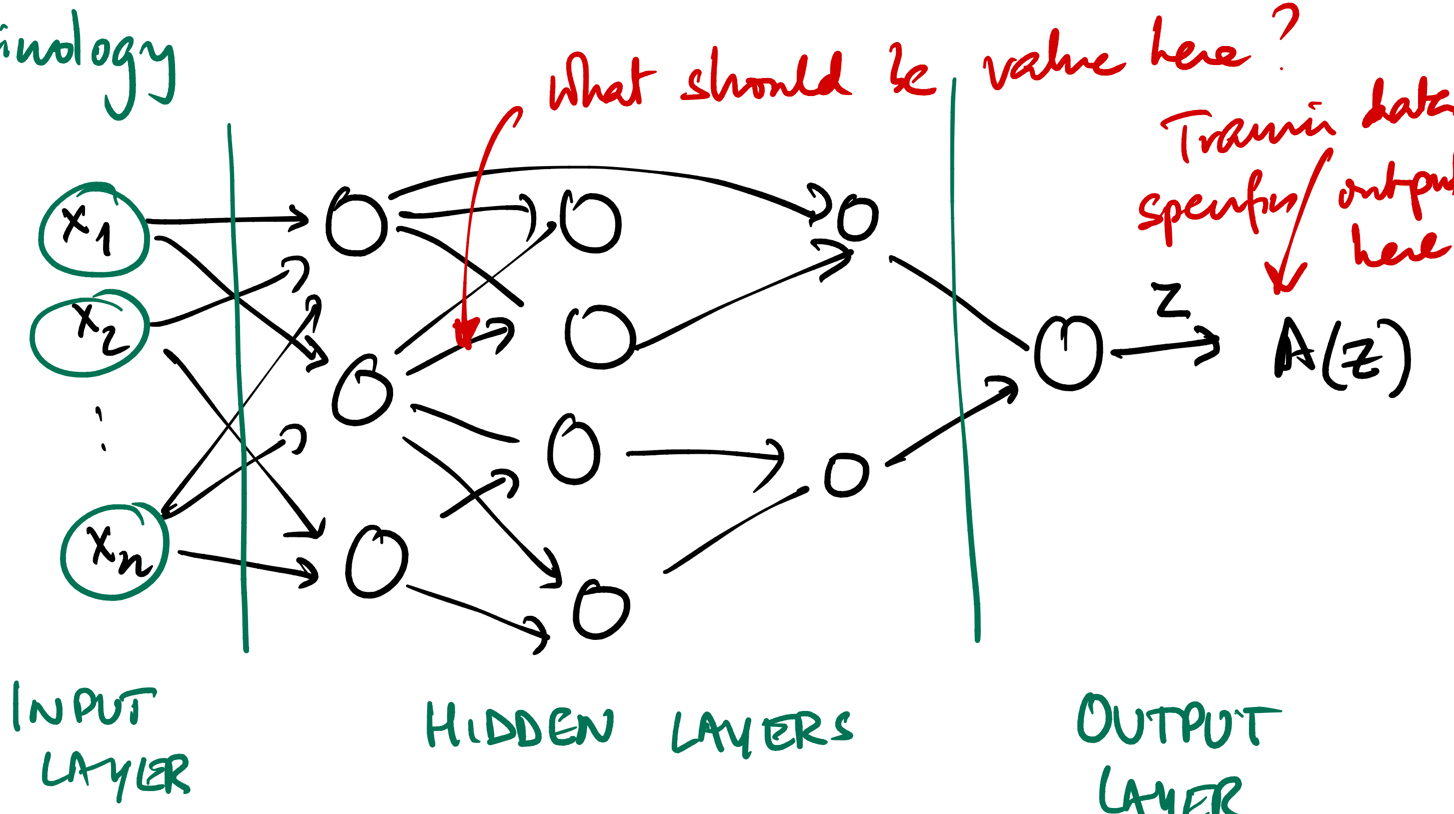
Any boolean function $f(x_1, x_2)$ can be expressed using a network of $\mathrm{NAND}(x_1, x_2)$

$x_1 \wedge x_2$ AND

$\neg(x_1 \wedge x_2)$ NAND

Boolean function → neural networks are "universal"

Terminology



Need to

1. Fix architecture — network connectivity of hidden layers, plus activation functions

2. Determine weights and biases of all nodes

→ Via Gradient Descent — More complicated than for a single node ⇒ "2nd AI Winter"

Determining architecture?

No obvious procedure

- Generally, "layers", no long distance connection

- Wlog, complete connectivity

Set $w_i = 0$ if no connection needed

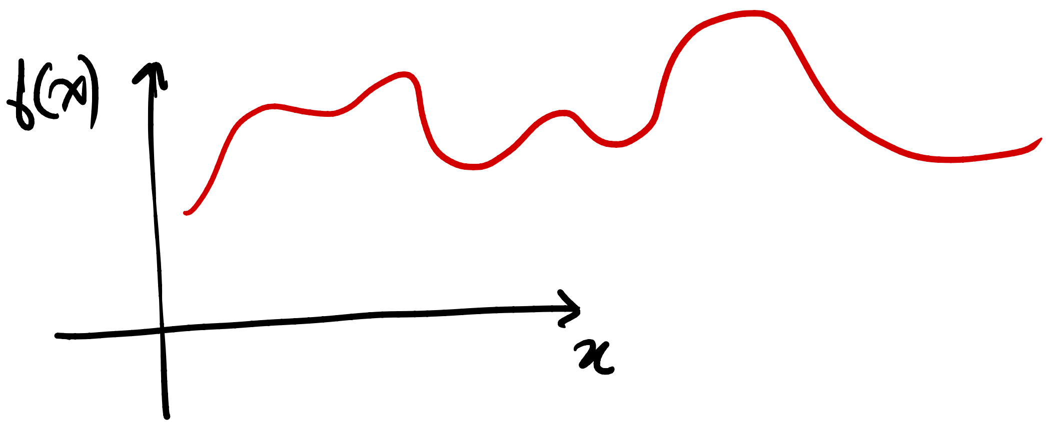
"Deep" neural networks = "Deep learning"

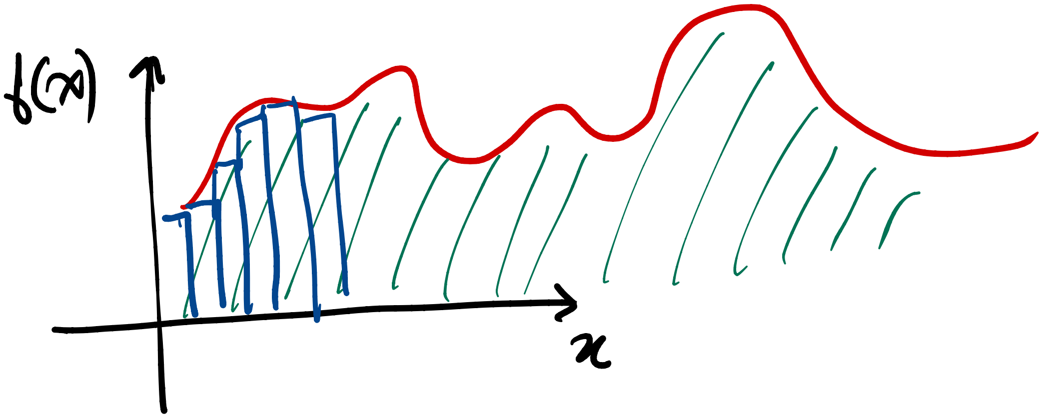
Many hidden layers

Was ≥ 4, now ≥ 100

Why neural networks "always" work

In general, assume classification boundary is some function $f(x)$





Recall integration

Limit of rectangular approximation

Similar idea

$$wx + b > 0$$

$$wx + b = 0$$

$$x = -\frac{b}{w}$$

$$x \xrightarrow{w} (b) \longrightarrow \sigma(z)$$

Changing b shifts step left or right

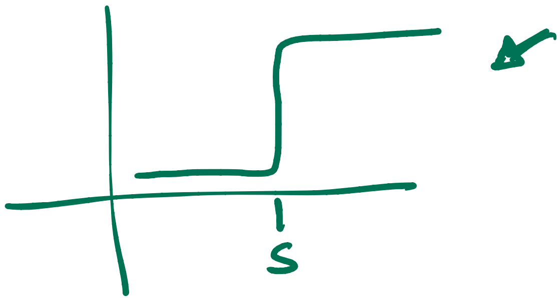
Increasing w sharpens the step

Let $s = -\frac{b}{w}$

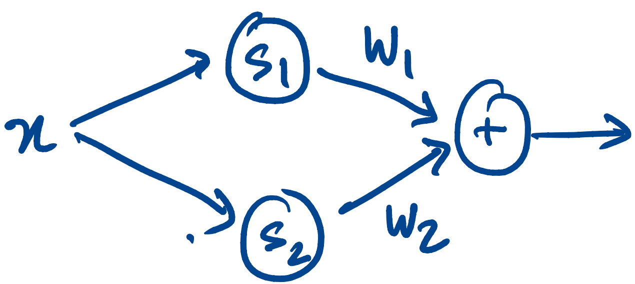
$$x \xrightarrow{w} s \rightarrow \sigma(z)$$

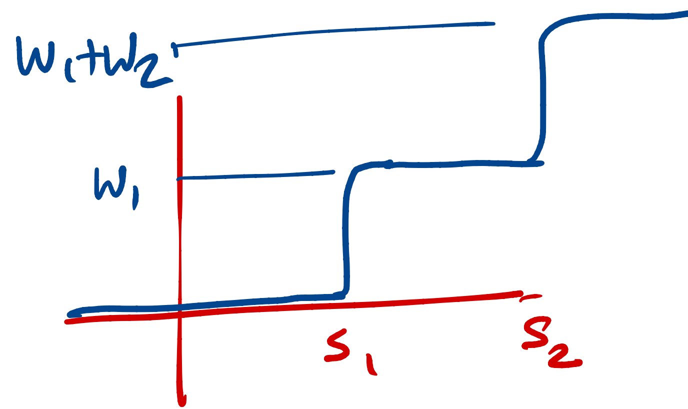
transition point

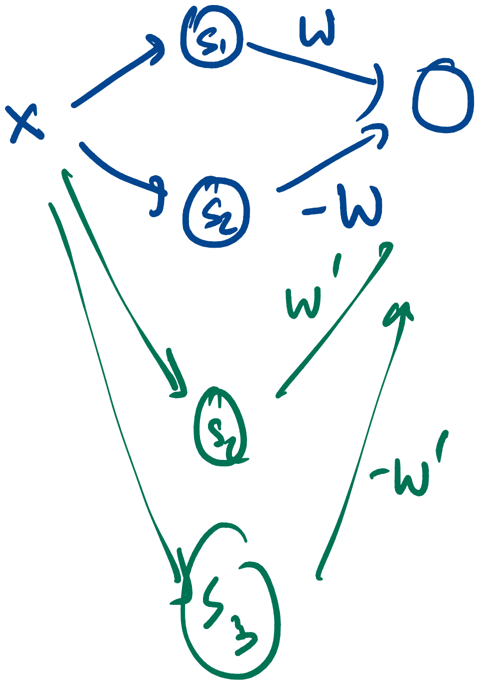
assumed large, to get sharp step

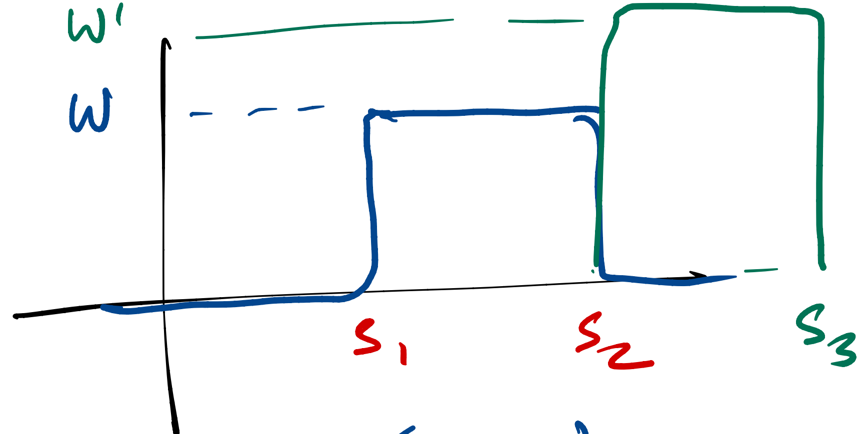


$s_1 < s_2$



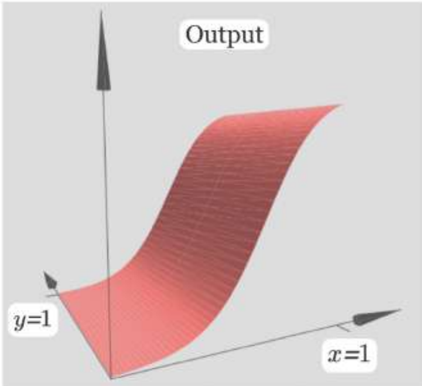


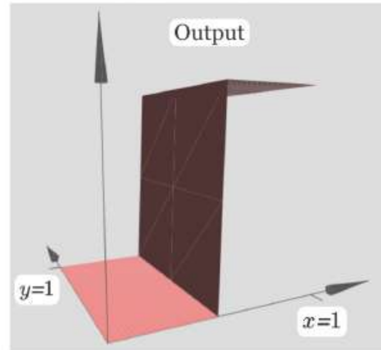


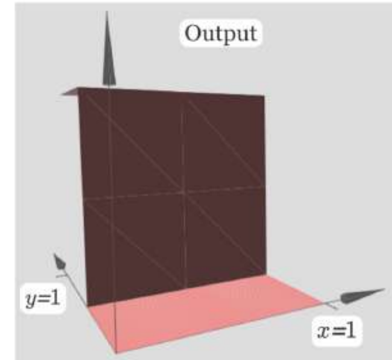


adjust this

via b_1, b_2





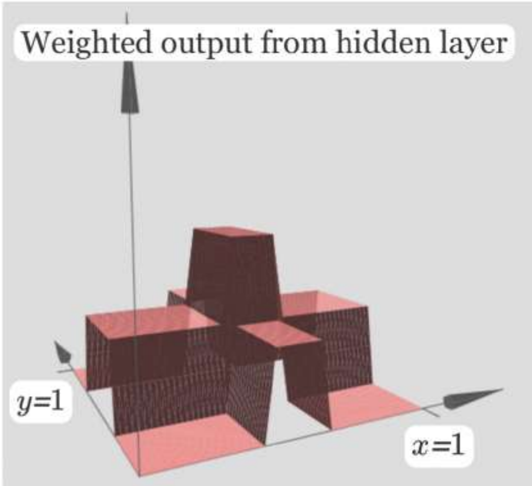


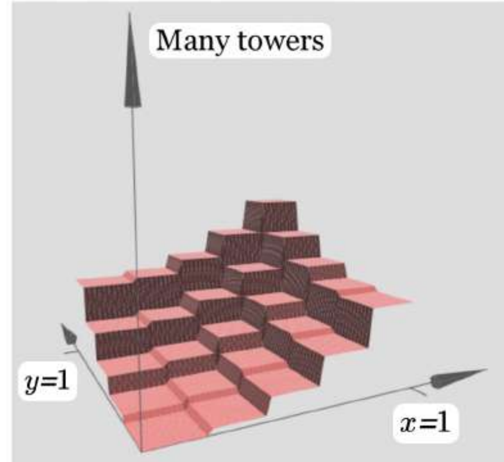
Surface to approximate

$f(x_1, x_2)$

Create steps in

x_1, x_2 direction





Create boxes

Notice we have only one hidden layer!