School of Electronic Engineering and Computer Science
Queen Mary University of London

Machine Learning
Neural Networks and Deep Learning

Dr Chao Liu

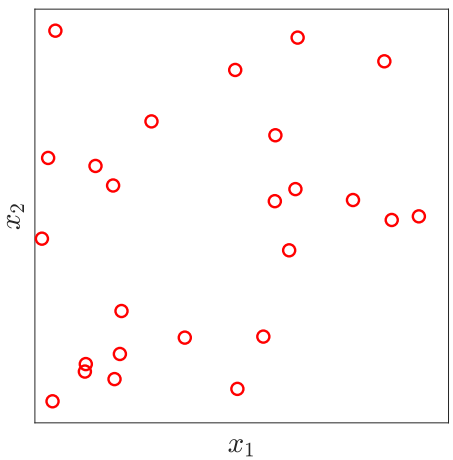
Credit to Dr Jesus Requena

Nov 2023

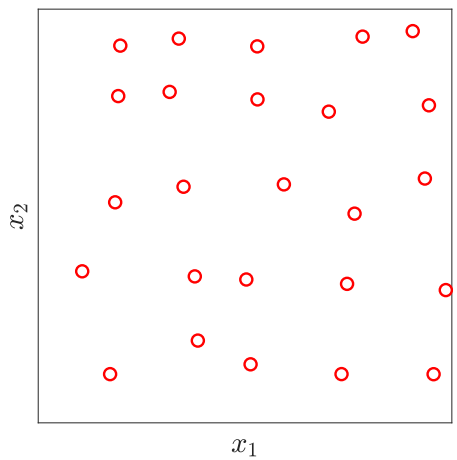
Patterns and structure

Patterns are regularities in our data and **structure** is a regularity in our target population. Machine learning project relies on discovering the underlying structure by identifying patterns in data.

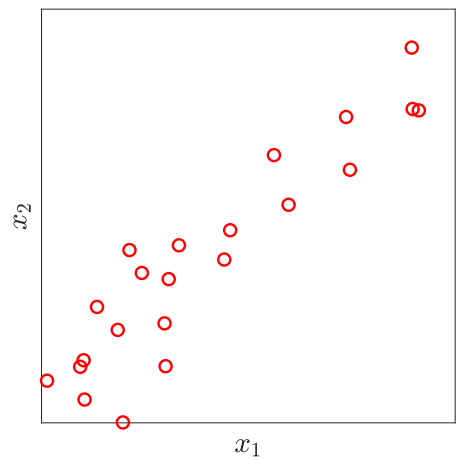
The main challenge is to distinguish **relevant** from **spurious patterns**. Which of the patterns below suggests the **least underlying structure**?



(a)



(b)



(c)

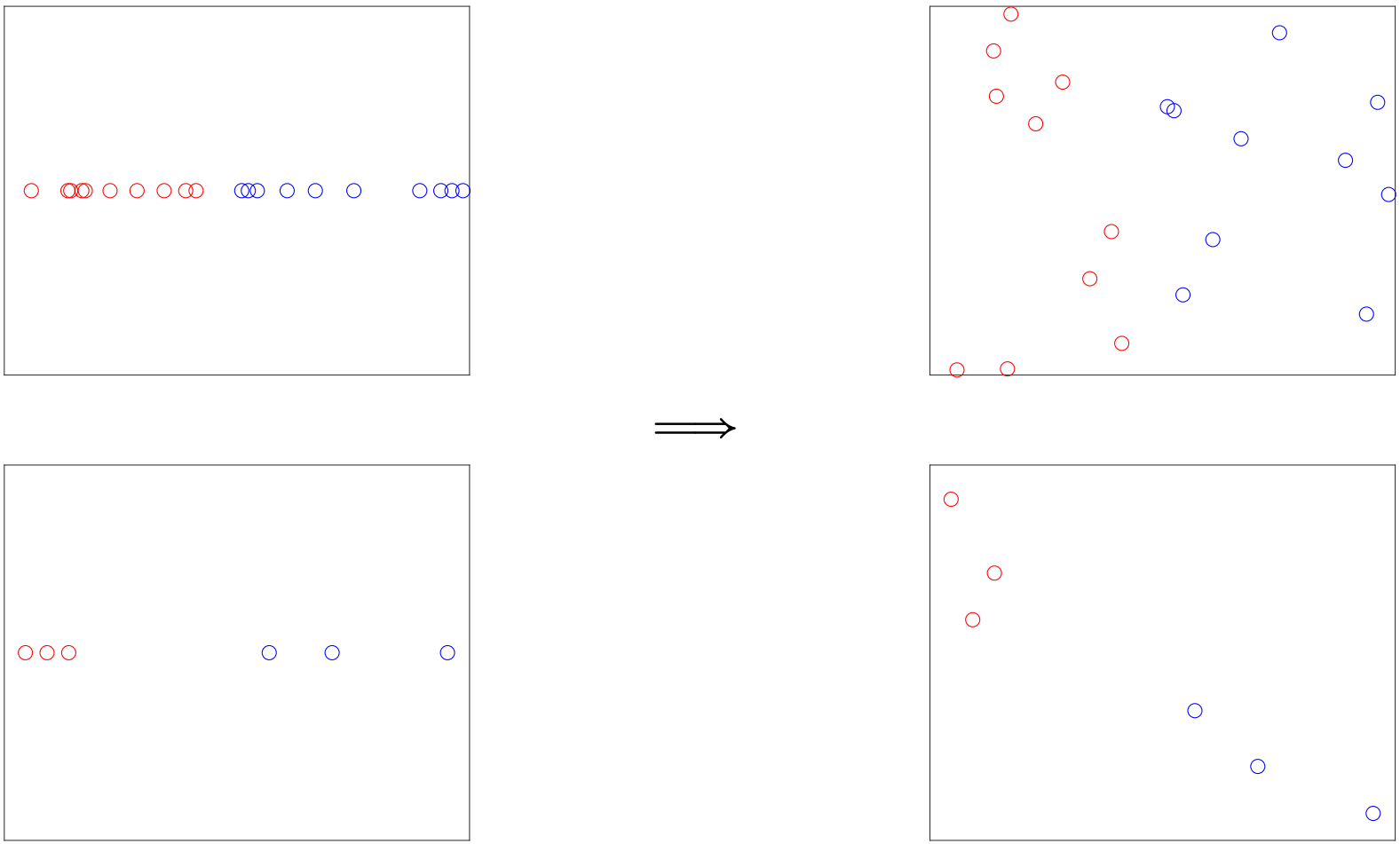
Short and wide datasets

Using the dataset below, train a model that predicts the salary of an individual based on their age, height, weight, gender and postcode.

ID	Age	Height [m]	Weight [kg]	Gender	Postcode	Salary
0	30	1.68	75	M	EC2Y 8DS	100
1	40	1.78	70	F	E14 0QR	200
2	35	1.7	85	M	WC2H 8LH	150

The Curse of Dimensionality

As the dimensionality increases, **data becomes sparser**. What happens if we add irrelevant attributes? What if we have fewer samples?



The Curse of Dimensionality

We find high dimensional datasets in many scenarios:

- **Complex** data types, such as grid data (pictures, audio files).
- Situations where we have **little prior knowledge** and we record everything, just in case.

Collecting as many attributes as possible seems like a good idea. Is it? Not necessarily: it **gives Randomness more opportunities to fool you**.

The **curse of dimensionality** is a warning. Irrelevant attributes **do not cancel each other out**, they show up as spurious patterns. Adding more attributes (*just in case*) can actually result in worse-performing models.

We can use **feature selection** techniques to reduce the dimensionality of the problem but first, let's use our **domain knowledge**.

Don't let randomness fool you!

Agenda

Neural networks: Not quite new

Neural networks were first proposed in the 1940s, became very popular in the 70s and 80s and fell out of favour in the 90s, as:

- Other algorithms performed better.
- They have a large number of parameters and are hard to train.
- Analysing them is difficult.

From 2010 onwards there has been a resurgence of interest in neural networks because of:

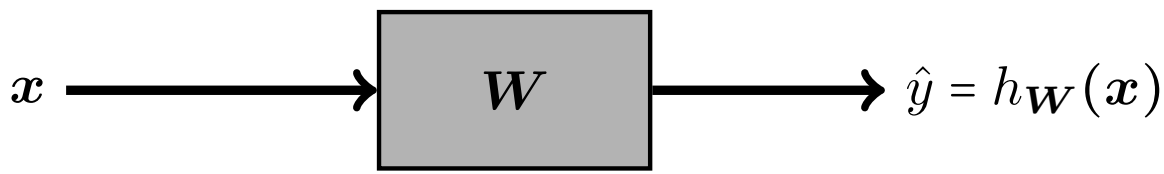
- Improved **optimisation** algorithms, increased **computational power** and **larger datasets**.
- Careful definition of **architectures**.
- Application of **transfer learning**.

What is a neural network?

A neural network is a **computing system** loosely inspired by the human nervous system, commonly used as a **family** of Machine Learning models. We will focus on their application to **supervised problems**.

From a **functional** point of view, a neural network:

- Produces an **output** (label) given an **input** (predictors), .
- Has **weights** (parameters) that can be **tuned** (trained).



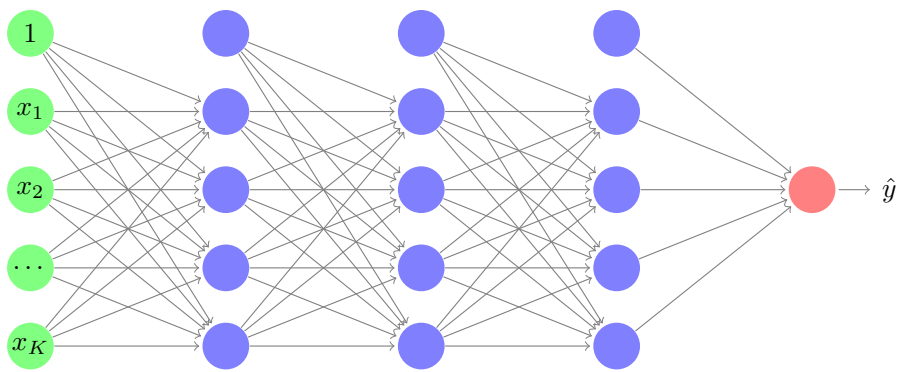
$\boldsymbol{W}$ denotes all the weights of the neural network and $h_{\boldsymbol{W}}(\boldsymbol{x})$ indicates explicytely that the prediction $\hat{y}$ depends on $\boldsymbol{W}$.

What is a neural network?

From an **computational** angle, neural networks consist of interconnected units that (loosely) mimmic **neurons**. This architecture is appealing since:

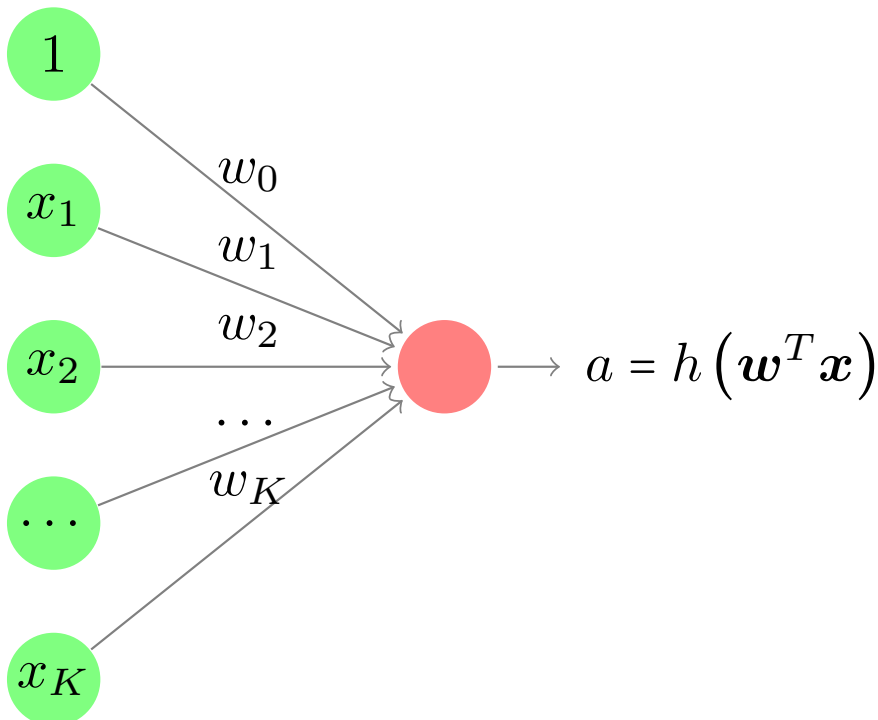
- **Neuroscience** suggests biological neural networks can solve any problem.
- **Mathematics** suggests artificial neural networks can reproduce any input/output relationship, provided they are complex enough.

Hence, the family of neural network models can be seen as a **universal machine**.



The perceptron

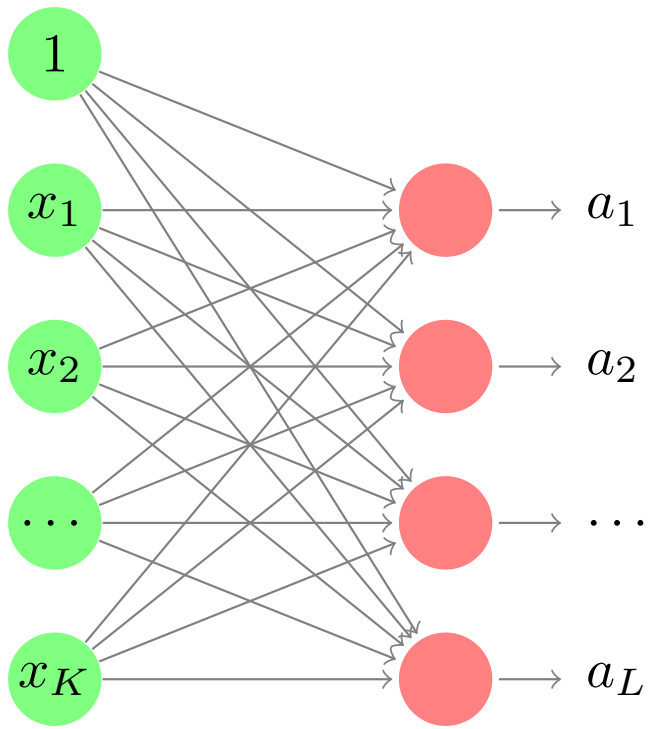
The perceptron is the **basic unit** of a neural network. It is defined by a **weight vector** $\boldsymbol{w}$ and an **activation function** $h(\cdot)$ that map an extended vector $\boldsymbol{x}$ to an output a. The coefficient w_0 is known as the **bias**.



The activation function is **non-linear**.

Layers

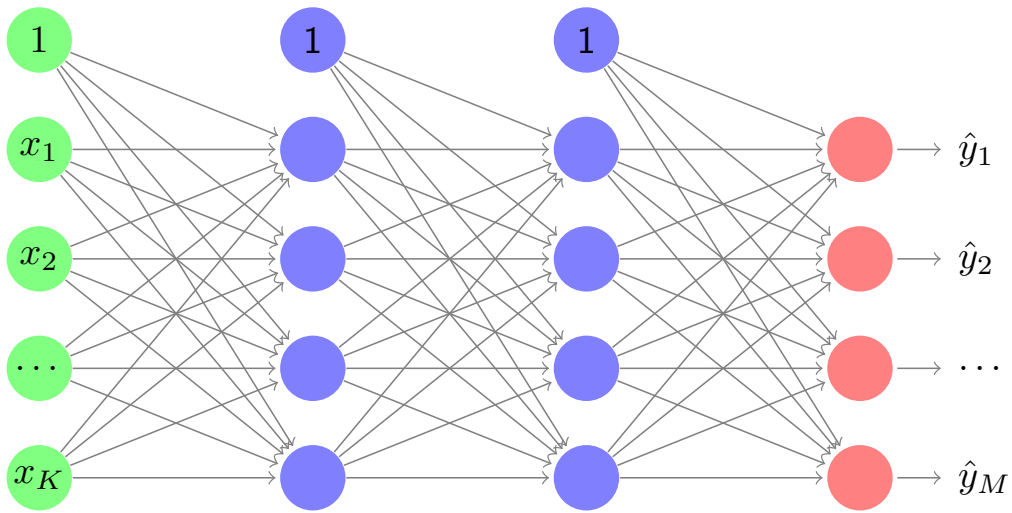
A **layer** is a collection of perceptrons that use the **same input**. Each perceptron within a layer produces a separate output.



A layer consisting of L perceptrons and K inputs has $L \times (K+1)$ weights.

The architecture

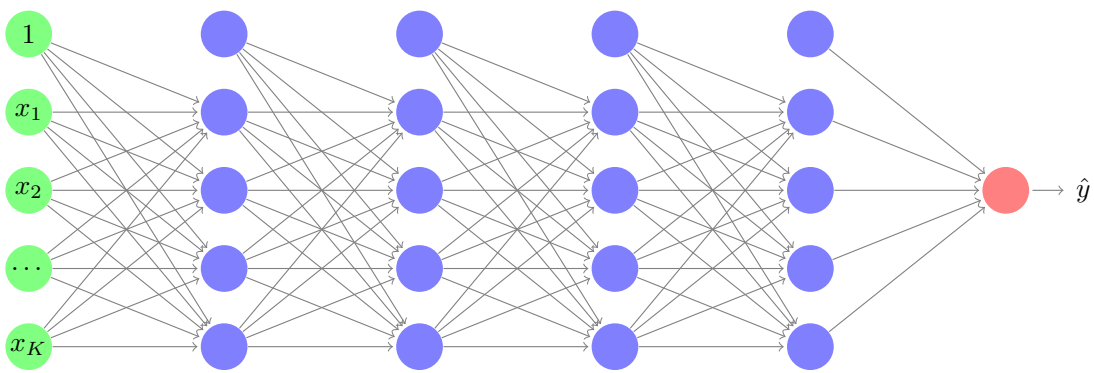
The **architecture** of a neural network describes how layers are connected. The **input layer** is the predictor vector, **hidden layers** produce internal features and the **output layer** produces the prediction.



Vector $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_M]^T = h_{\boldsymbol{W}}(\boldsymbol{x})$ is the output of the neural network.
(*Note that here $\hat{y}_j$ is not the prediction for the j-th sample!*)

The neural network

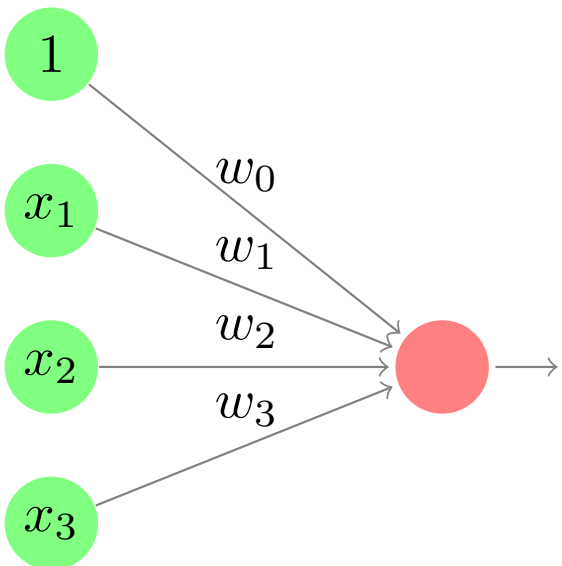
- A neural network consists of **perceptrons** arranged in **layers** that are connected according to an **architecture**.
- There is one **parameter** per connection (the connections's **weight**).
- Each layer produces intermediate **features**.
- The number of layers determines the depth of the neural network (hence the distinction between **shallow** and **deep** neural networks).



Agenda

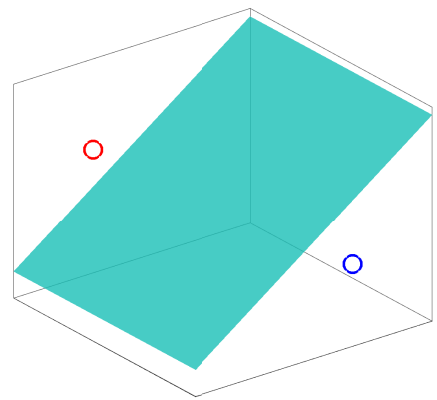
The perceptron as a linear classifier

A perceptron that uses a **step** activation function is actually a **linear classifier**. What if it uses a **logistic function**?



$$a = h_{step}\left(\boldsymbol{w}^T \boldsymbol{x}\right)$$

$$h_{step}(d) = \begin{cases} 1, & \text{if } d > 0. \\ 0, & \text{otherwise.} \end{cases}$$

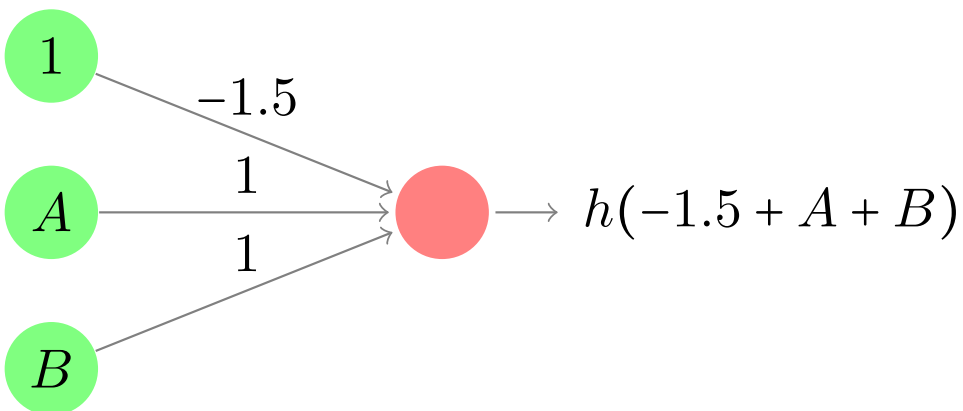


In a 3D predictor space, a perceptron with a step activation function separates two decision regions by a plane!

The perceptron as a basic logical function

We can use perceptrons to implement logical functions.

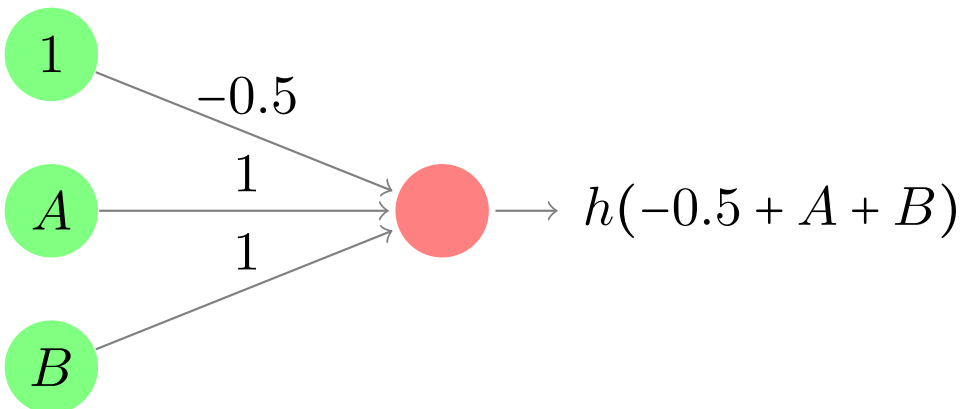
In the preceptron below, assuming that the activation function is a step and A and B can be either 0 or 1, the output $f(A, B)$ corresponds to a logical AND of A and B.



A	B	$f(A, B)$
0	0	0
0	1	0
1	0	0
1	1	1

The perceptron as a basic logical function

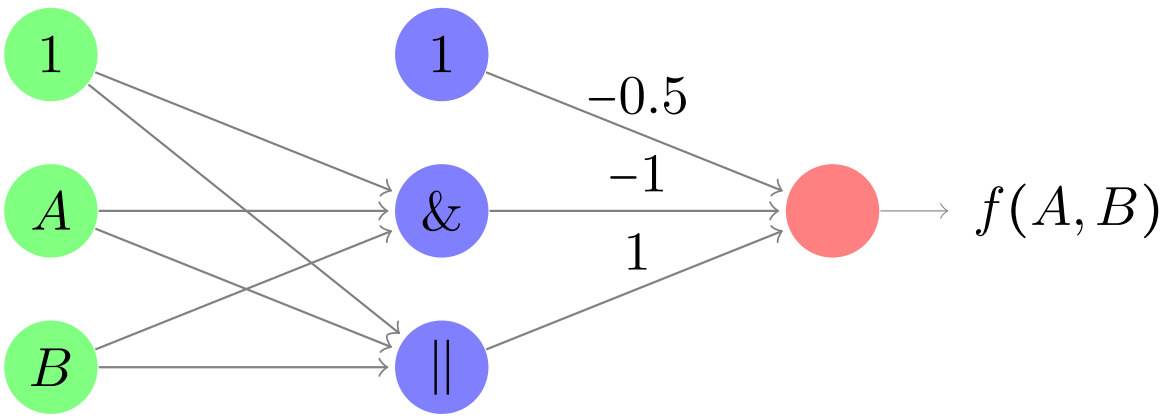
What logical function would you obtain if the coefficients were $w_0 = -0.5$, $w_A = 1$ and $w_B = 1$ instead?



A	B	$f(A, B)$
0	0	
0	1	
1	0	
1	1	

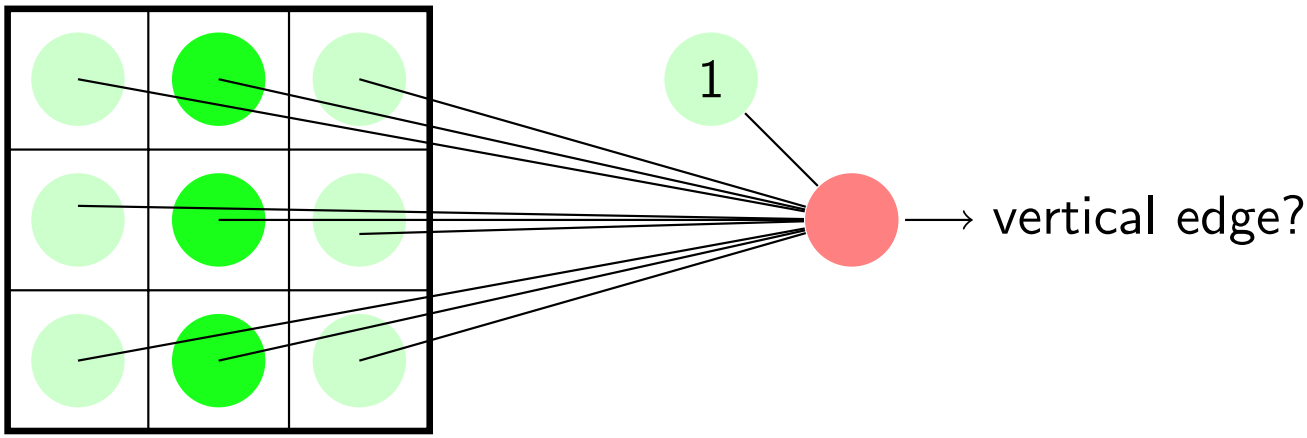
Adding depth to derive new logical functions

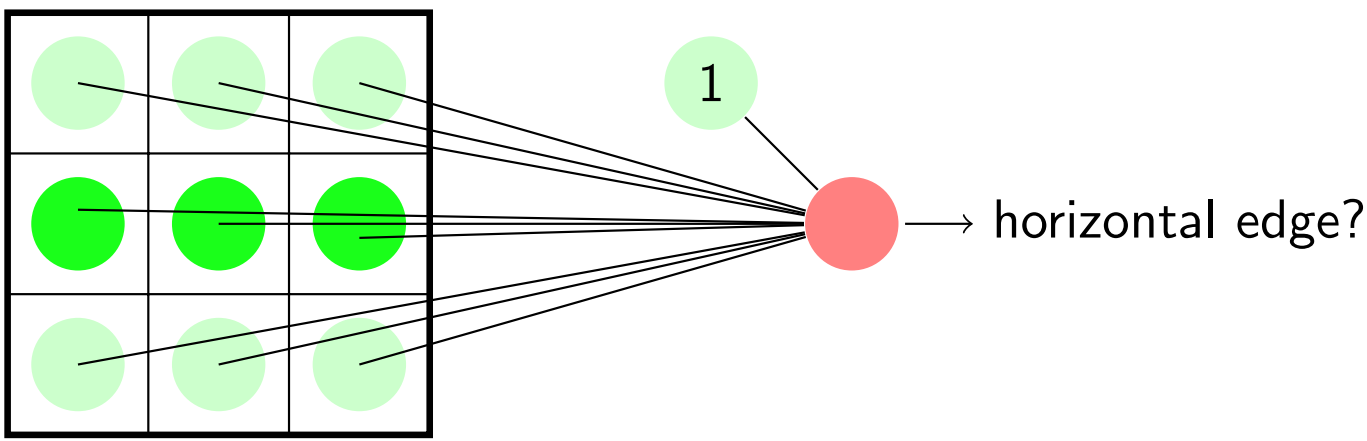
What about the following neural network? In this diagram, & denotes an AND perceptron, and || an OR perceptron.



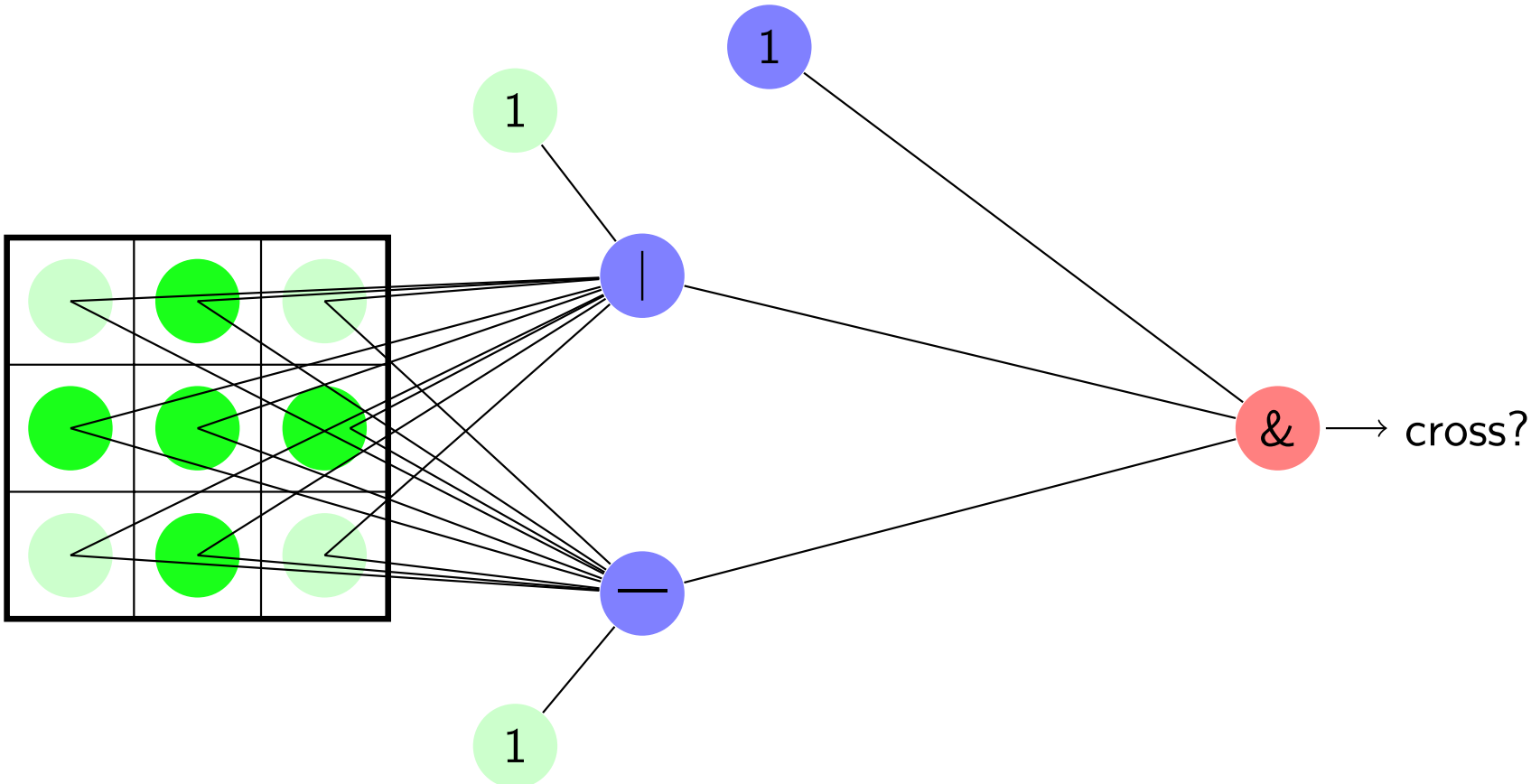
A	B	&	$\|\|$	f
0	0			
0	1			
1	0			
1	1			

The perceptron as a grid pattern detector



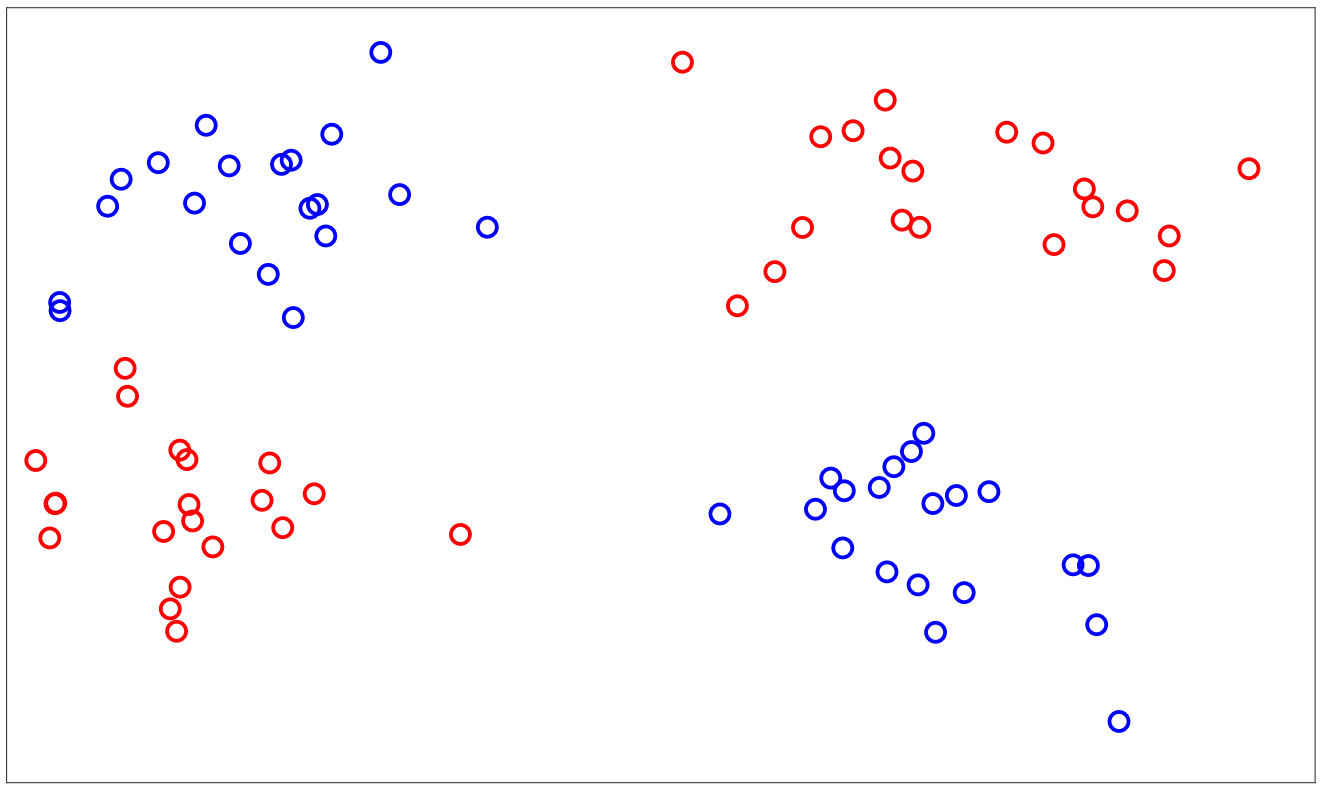


Adding depth to detect derived grid patterns



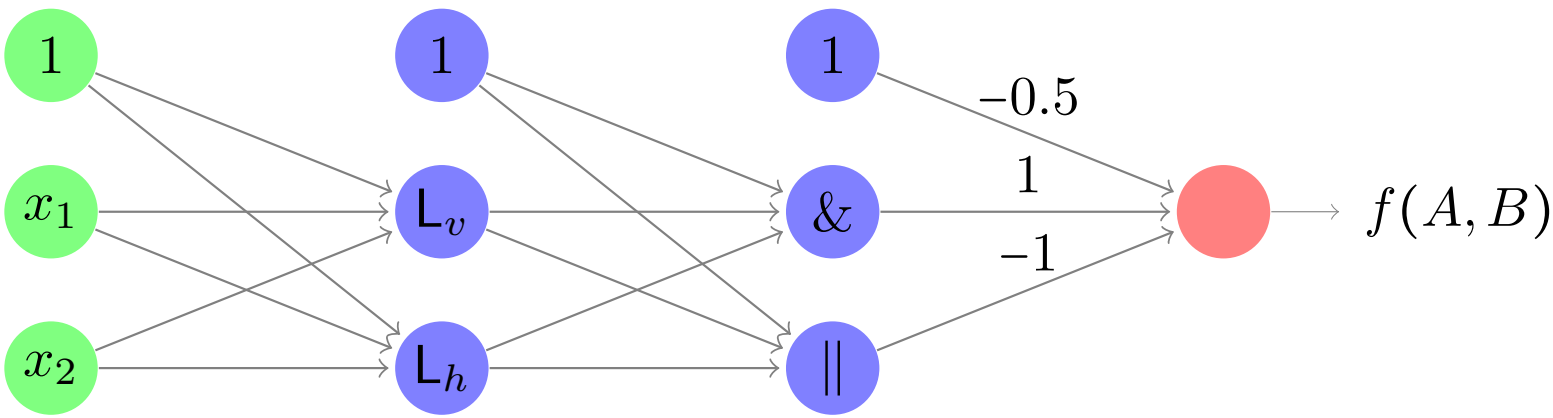
Combining linear classifiers and logical functions

Build a neural network classifier for the following dataset.



Combining linear classifiers and logical functions

In this diagram, L_v and L_h represent perceptrons that implement a vertical and horizontal linear boundaries, respectively. Their outputs indicate whether samples are on one side of the boundary or another.



Versatility of neural networks as a computing system

Using a single perceptron we can implement:

- Linear boundaries.
- Logical functions.
- Grid pattern detectors.

Connecting perceptrons allows us to implement more complex operations, e.g. complex boundaries, complex logical functions and complex grid patterns. This can be seen as building complexity from basic operations.

So far we have seen neural networks from a computation angle and have specified each network by ourselves. Note this is **not** machine learning!