TMLS20 Machine Learning

Lecture 9 - Neural Networks

Beril Sirmacek

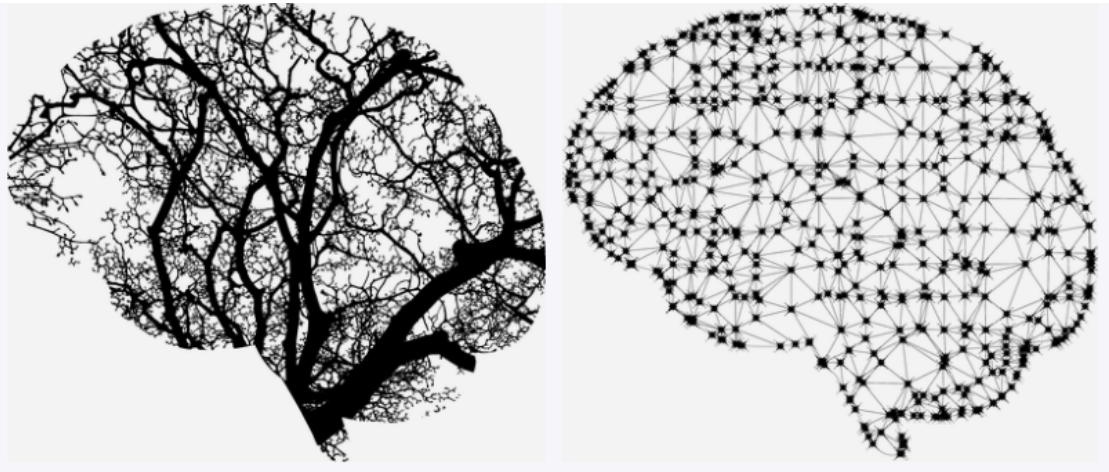
Jönköping AI Lab
Jönköping University
Beril.Sirmacek@ju.se

February, 2020

Table of Contents

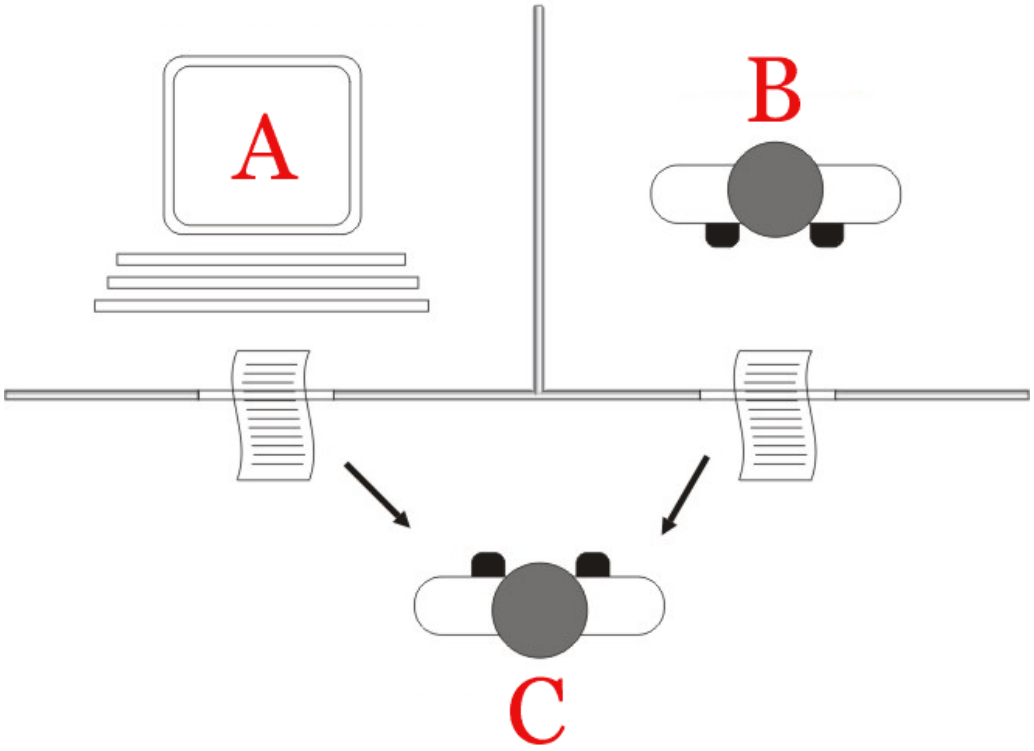
Neural networks Introduction: Intelligence

Human brain vs. Machine



- Soft tissues *(not explained)*
- Non-linear mathematical functions *(explained?)*
- Purpose: AI *(AGI)*

Neural networks Introduction: Turing test



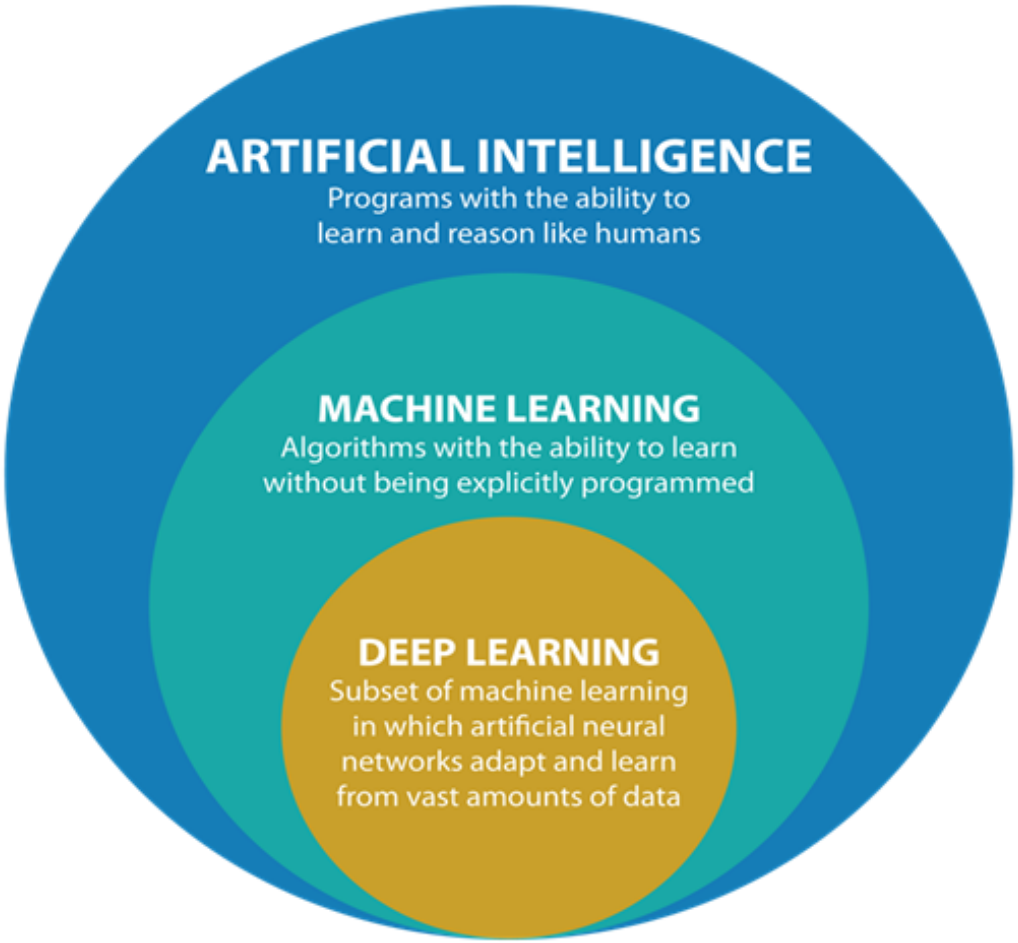
The "standard interpretation" of the Turing test, in which player C, the interrogator, is given the task of trying to determine which player – A or B – is a computer and which is a human. The interrogator is limited to using the responses to written questions to make the determination.

Saygin, A. P., Cicekli, I., Akman, V. (2000), "Turing Test: 50 Years Later" (PDF), Minds and Machines, 10 (4): 463–518.

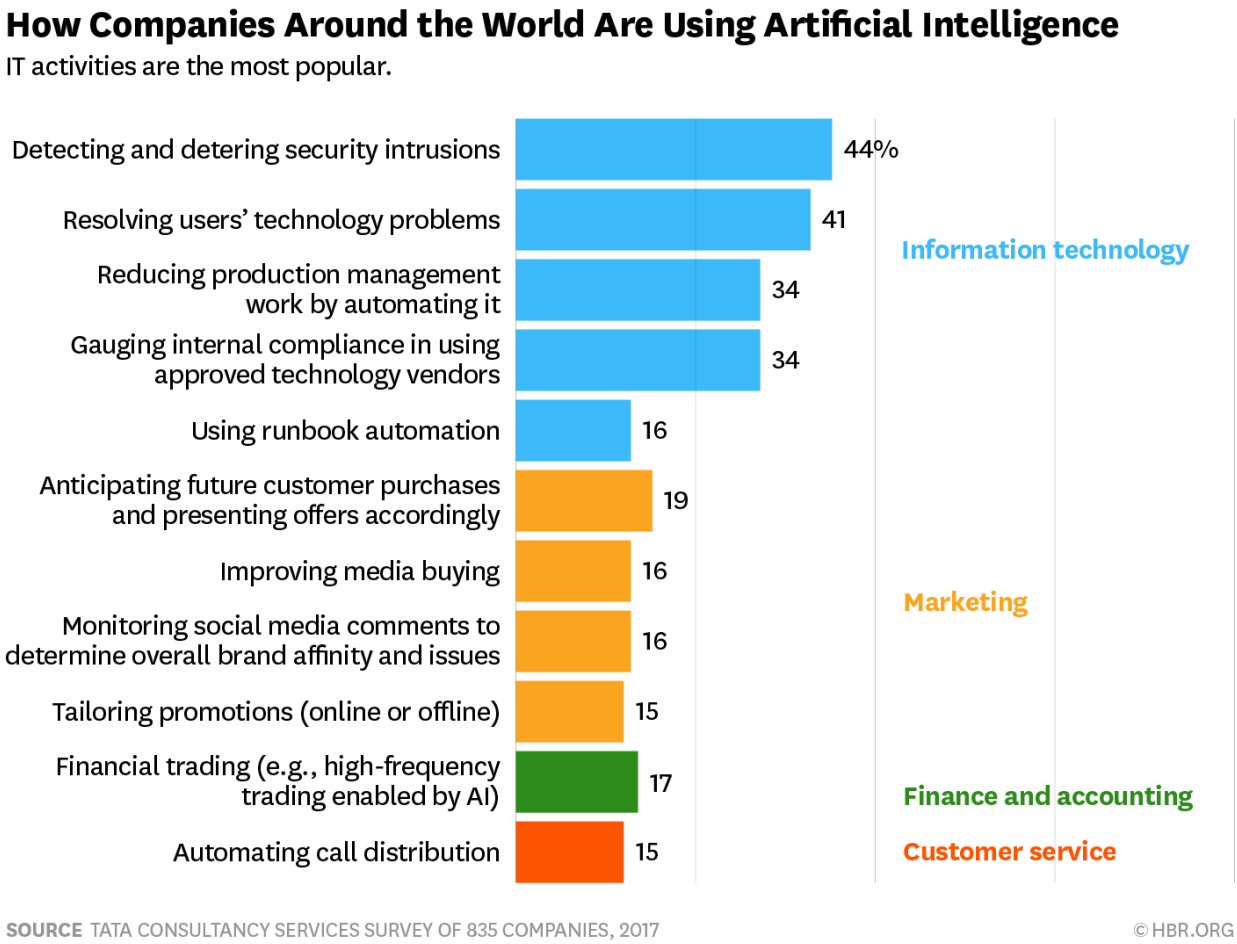
Neural networks Introduction: Turing test passed

Getting a hair dresser appointment `https://youtu.be/JvbHu_bVa_g`

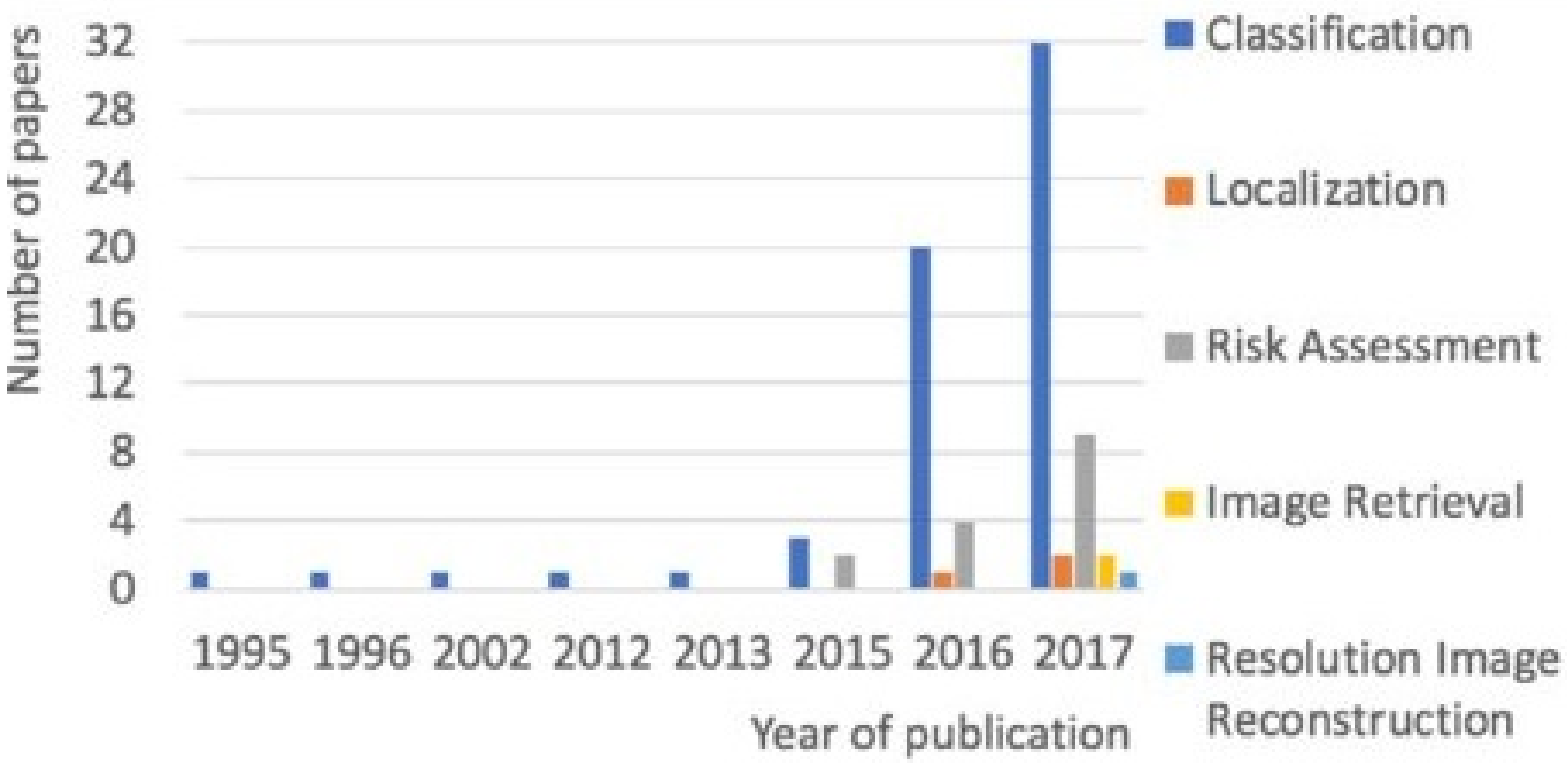
Neural networks Introduction: map



Neural networks Introduction: Why important?



Neural networks Introduction: Why now?



https://doi.org/10.1186/s12859-019-2823-4

Neural networks Introduction: Biological inspiration

A single neuron cell

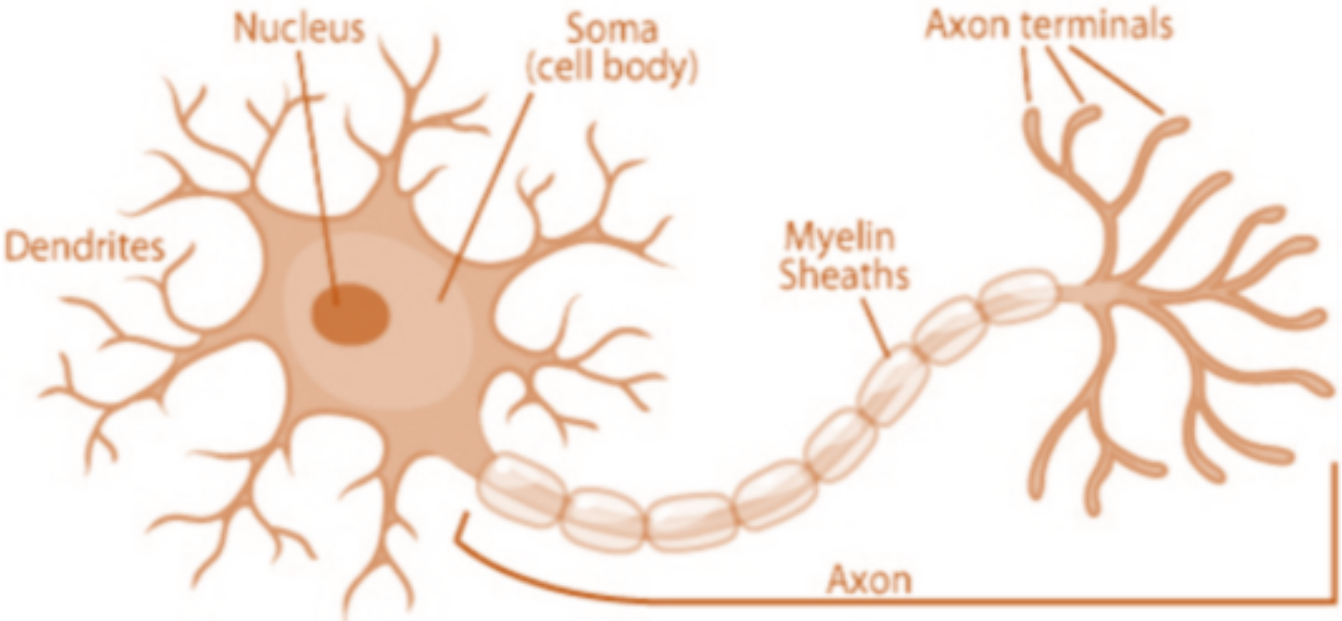
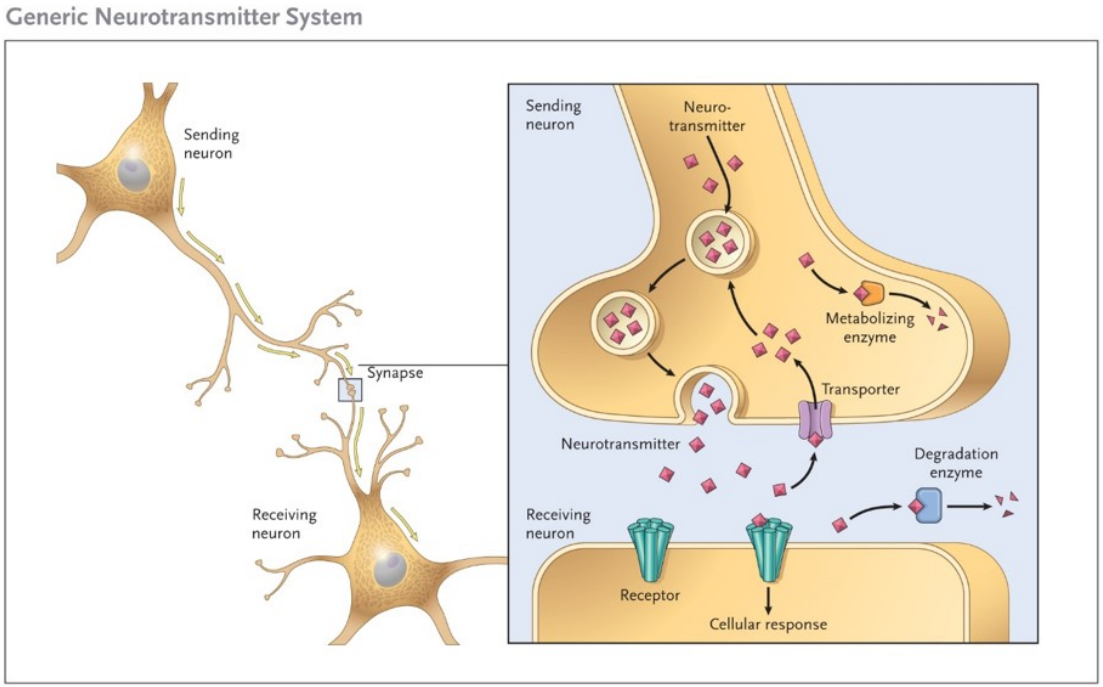


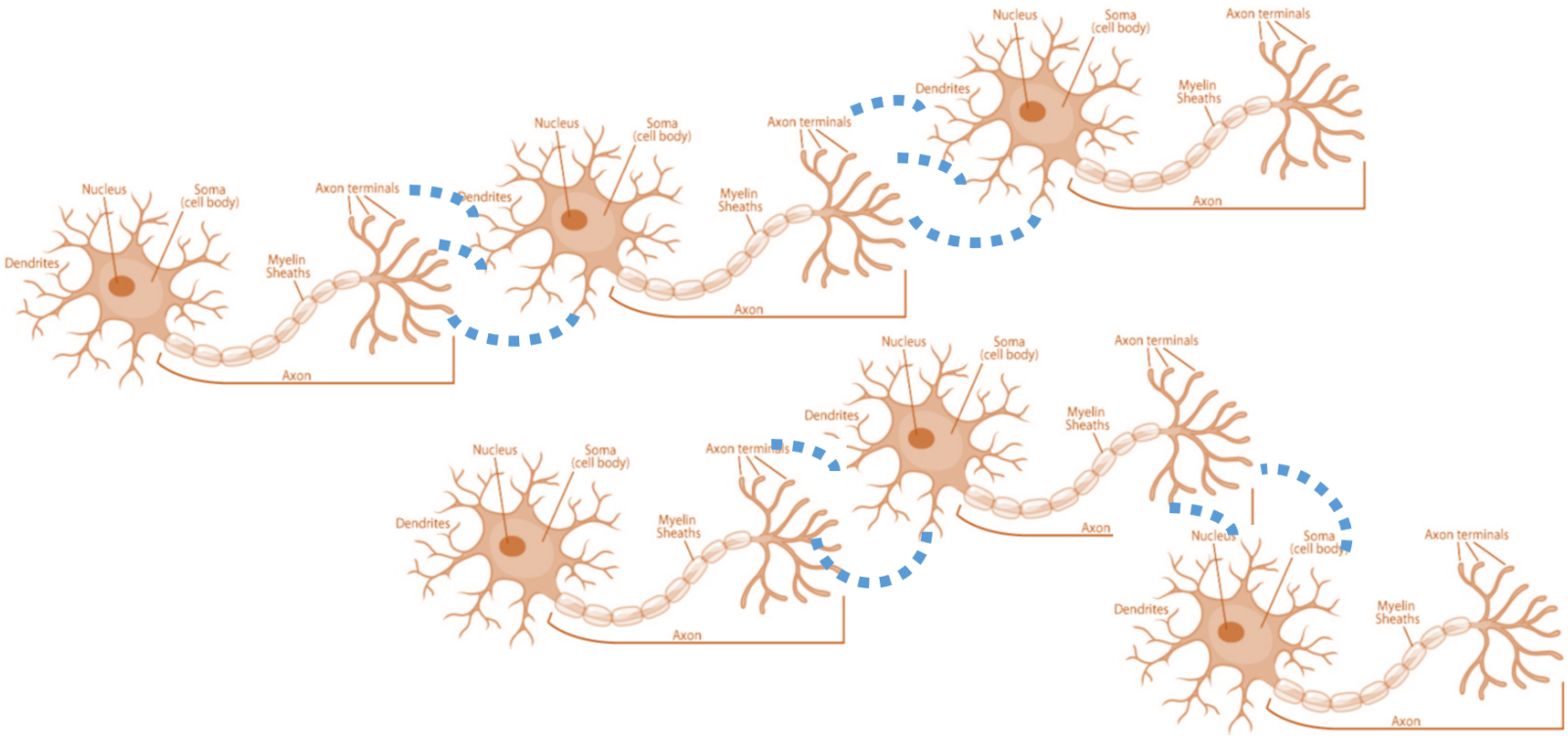
Image source: Wikipedia

Neural networks Introduction: Biological inspiration

Synapses (the area where one neuron ends and the next one comes)



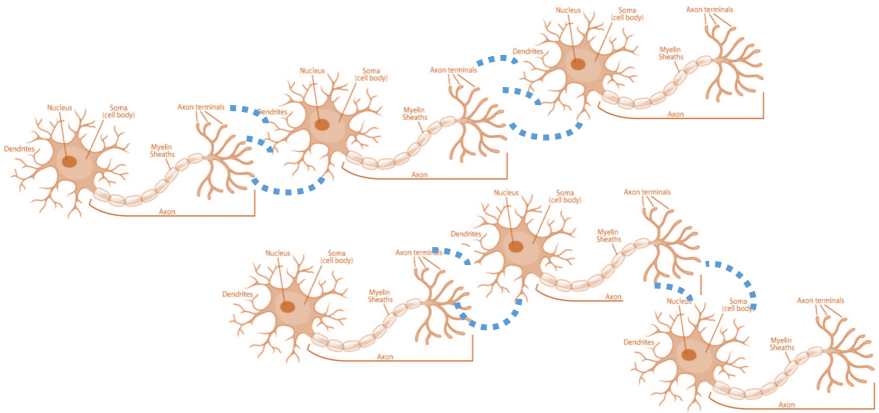
Neural networks Introduction: Biological inspiration



An illustration of some connected neurons in the human brain.

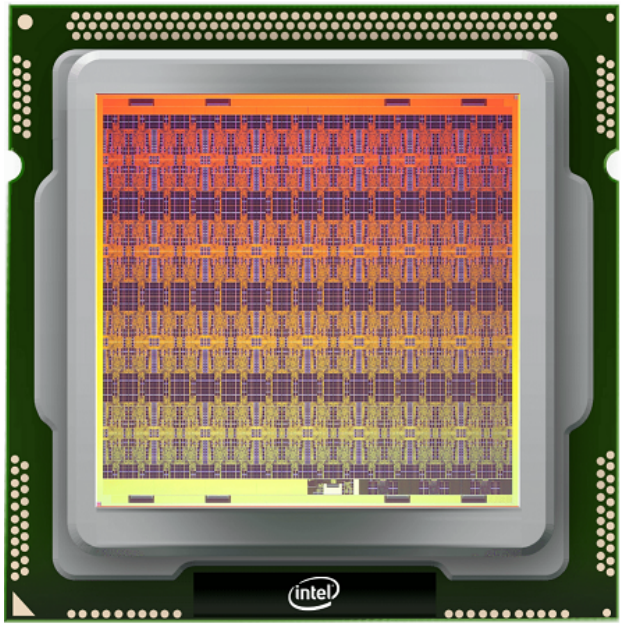
Neurons which fire together wire together.

Neural networks Introduction: Biological inspiration



The connection between two neurons can be vary between **strong** and **weak**. A strong connection allows more charge to flow. A neuron pathway which frequently transmits charge will eventually become a strong pathway.

Neural networks Introduction: Recent advances



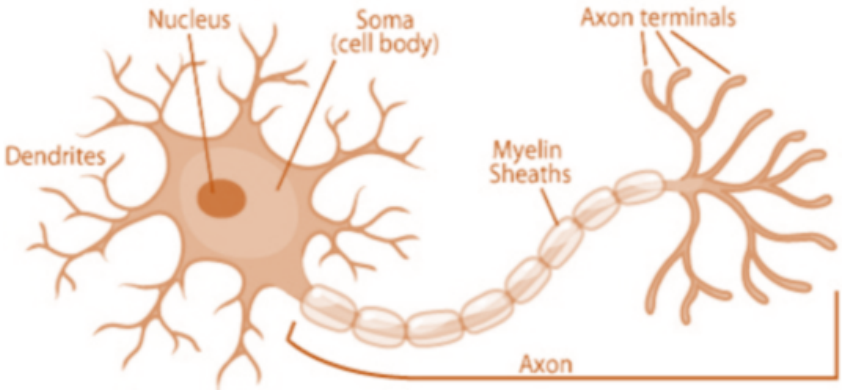
Intel Corporation's self-learning neuromorphic research chip, code-named "Loihi." (Credit: Intel Corporation)

- Neuromorphic computing research emulates the (analog) neural structure of the human brain.
- The Loihi research chip includes 130,000 neurons optimized for spiking neural networks.

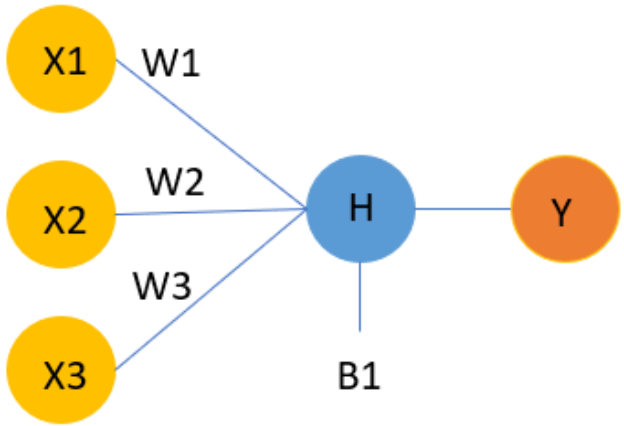
Neural networks Introduction: Recent advances

- Neuralink
  `https://www.biorxiv.org/content/10.1101/703801v1`
- Neuromorphic computing
  `https://www.nature.com/articles/s41467-019-12521-x`

Neural networks Introduction: Biological inspiration

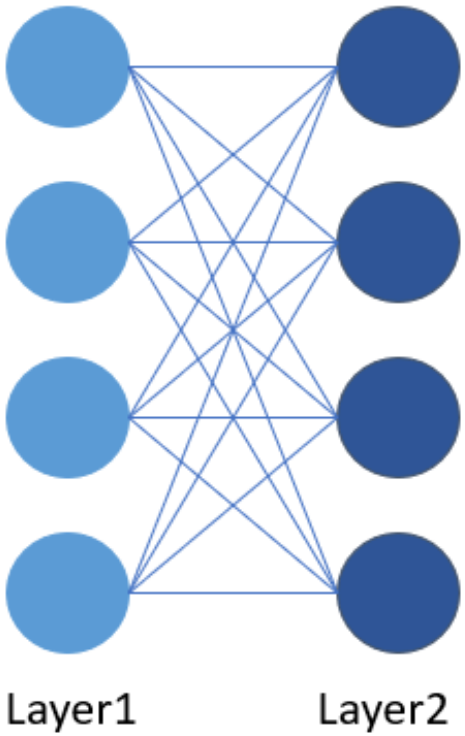


Biological representation of one neuron



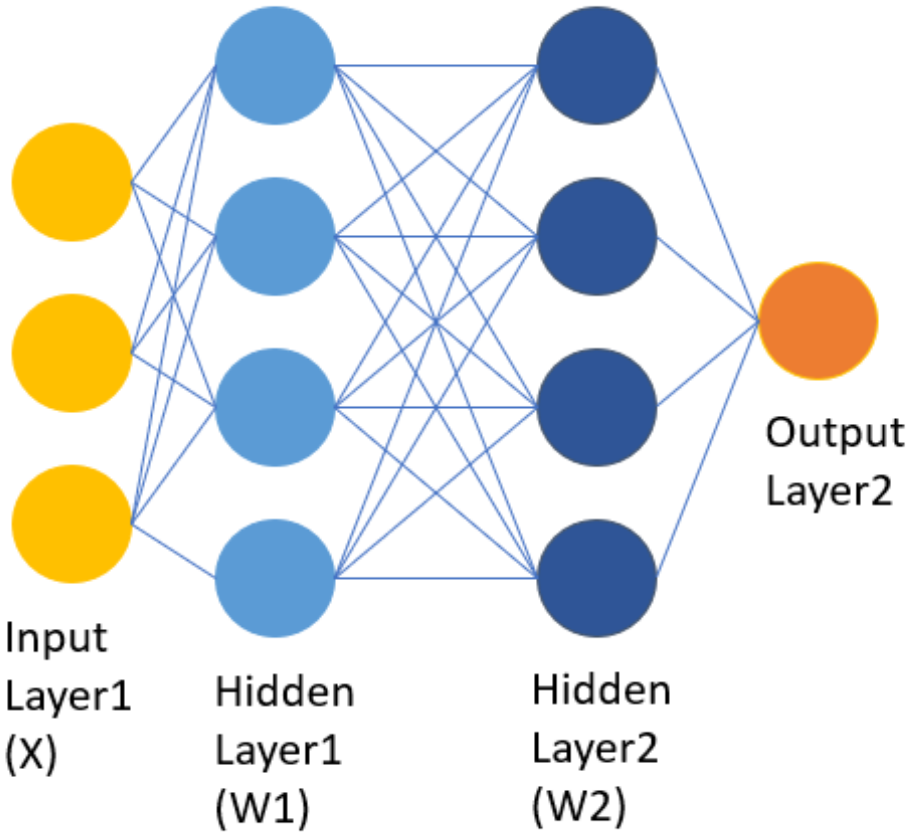
Digital representation of one neuron

Neural networks Introduction: Biological inspiration



Digital representation of a neural network

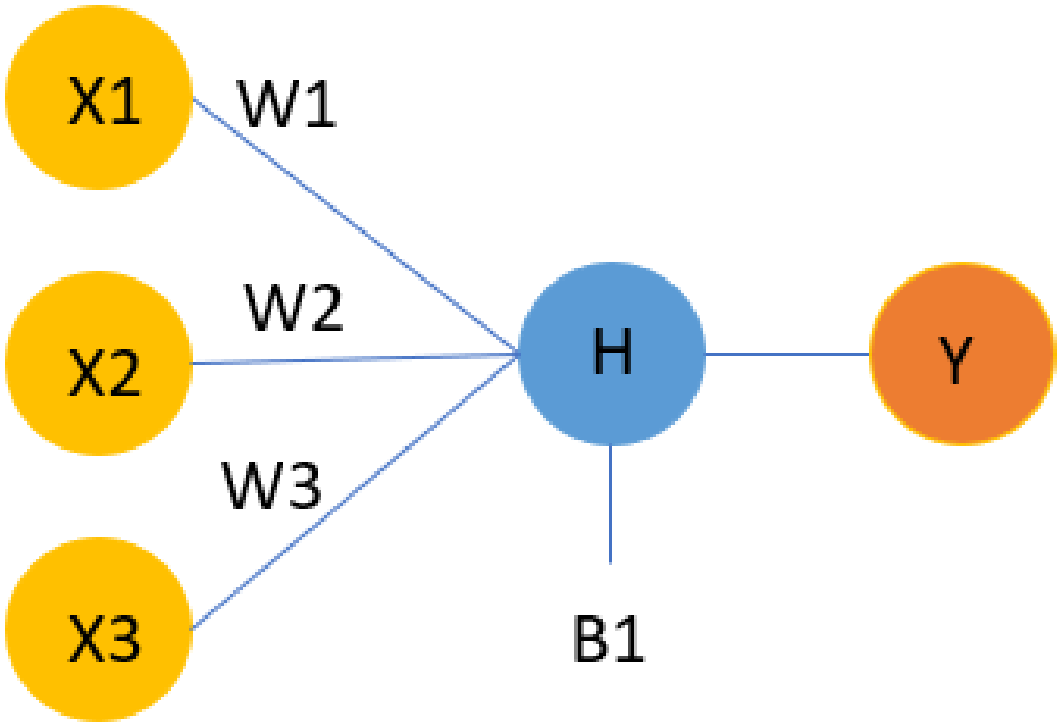
Neural networks Introduction: Biological inspiration



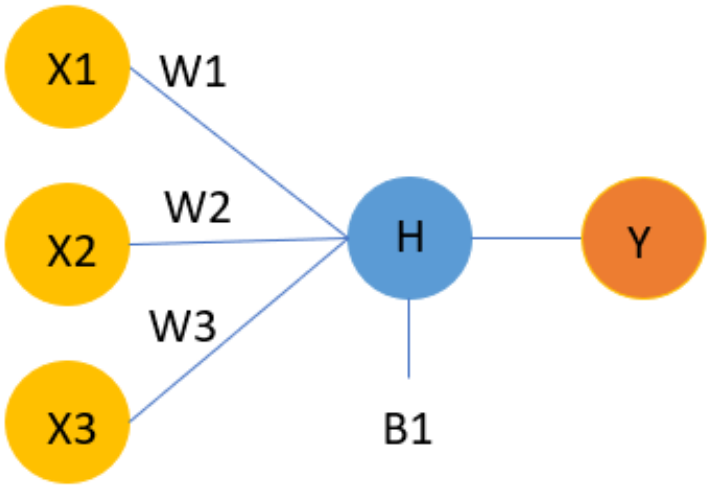
Digital representation of a neural network

Neural networks Introduction: Digital neurons

Digital representation of one neuron

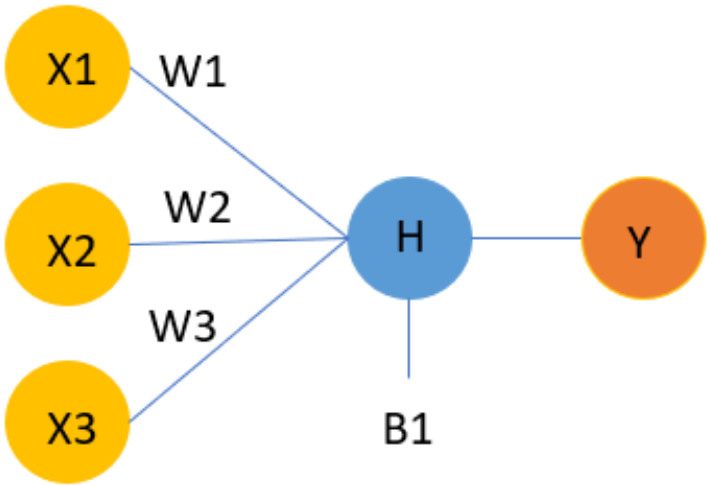


Neural networks Introduction: Digital neurons



X represents the values of the nodes of the previous layer.

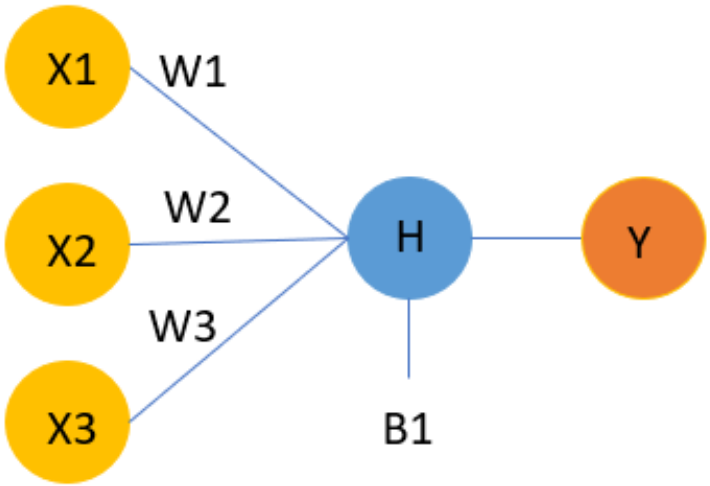
Neural networks Introduction: Digital neurons



W represents the weights between the nodes in the previous layer and the output node.

(W is sometimes called **preactivation** or **input activation**.)

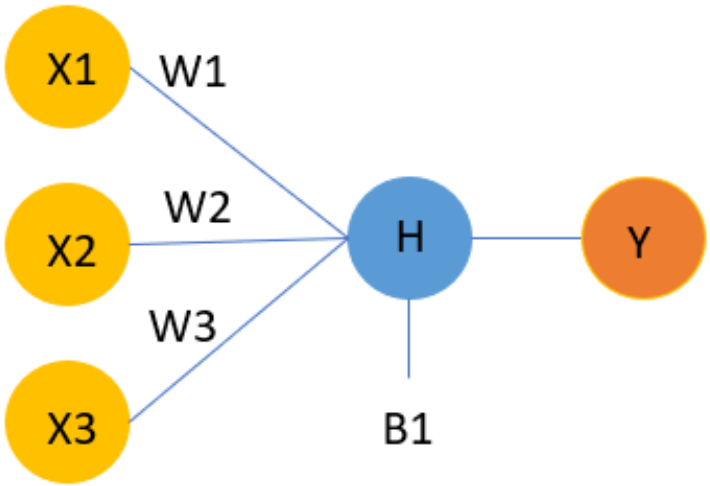
Neural networks Introduction: Digital neurons



$Y = f(H)$

H is the intermediate node value. This is not the final value of the node.

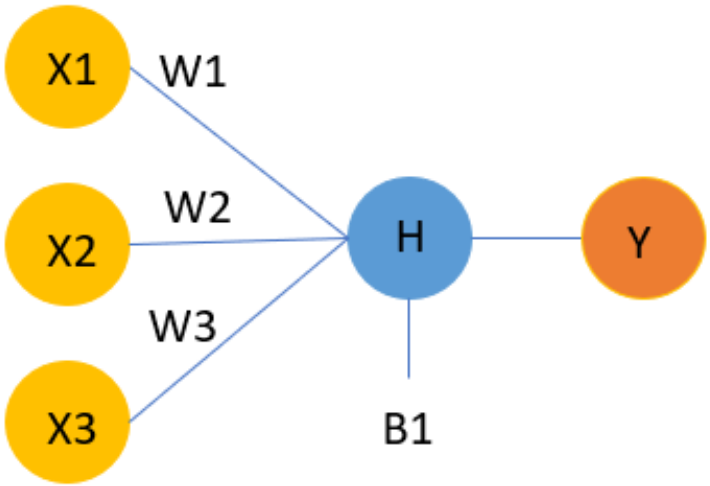
Neural networks Introduction: Digital neurons



Y is the final value of the node.

$Y = f(H)$

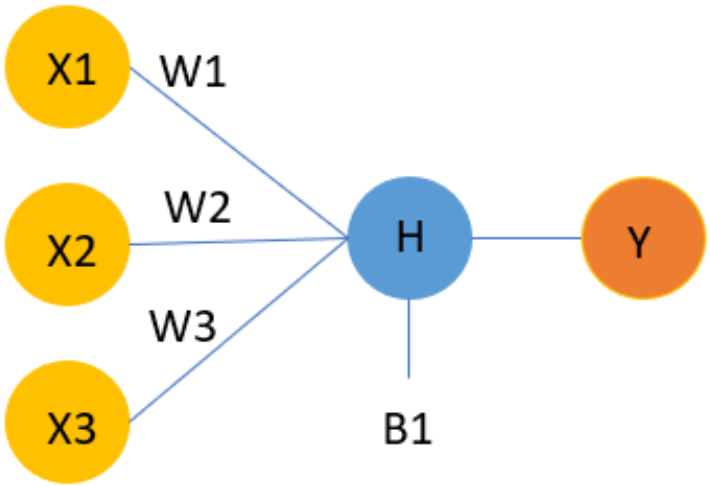
Neural networks Introduction: Digital neurons



B represents bias, which is an additional value present for each neuron. Bias is essentially a weight without an input term.

It's useful for having an extra bit of adjustability which is not dependant on previous layer.

Neural networks Introduction: Digital neurons

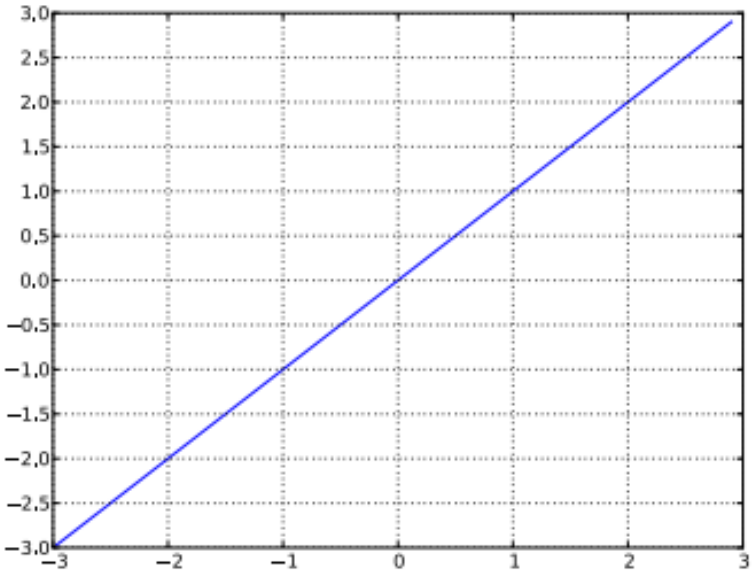


$Y = f(H)$

$f()$ is the **activation function**

It is sometimes called as **output activation** which is something which we can choose.

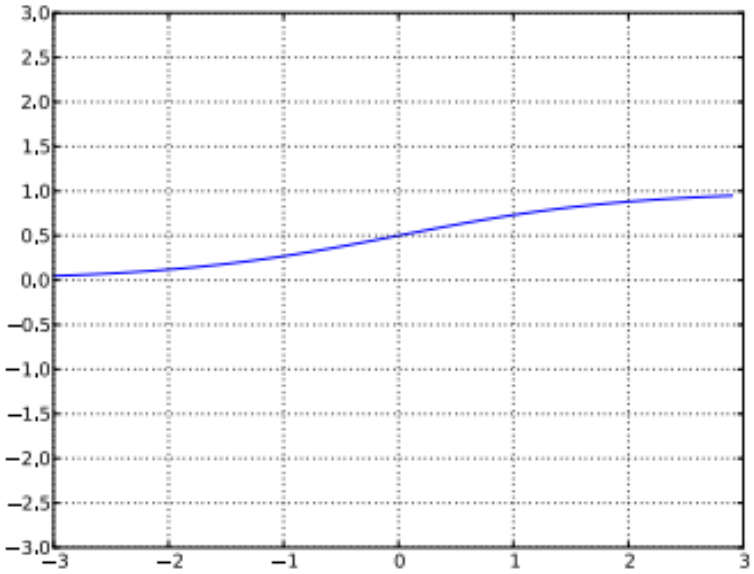
Neural networks Introduction: Digital neurons



$y = f(H)$ is a **linear activation function**

(probably not very interesting, not shaping the mapping)

Neural networks Introduction: Digital neurons

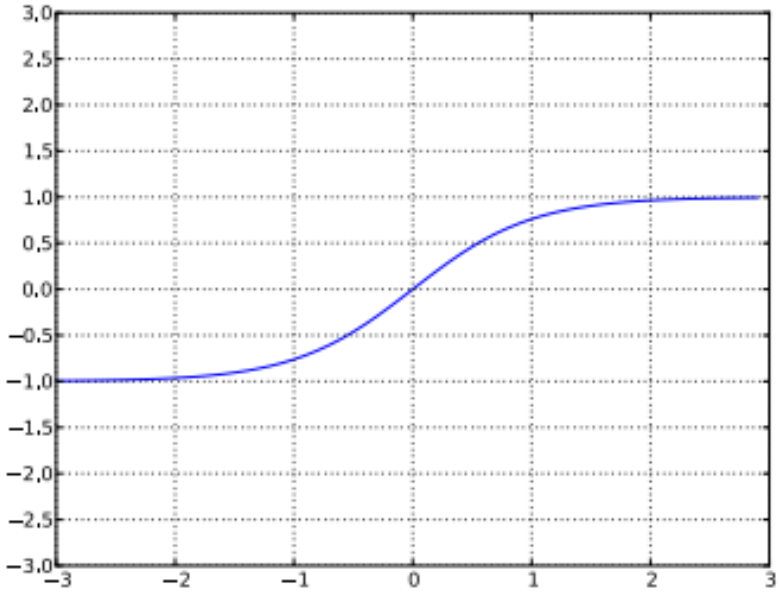


$y = sigm(H) = 1/(1 + exp(-H))$ is a **sigmoid activation function**

-squashes the H value between 0 and 1
-always positive
-bounded

-strictly increasing

Neural networks Introduction: Digital neurons



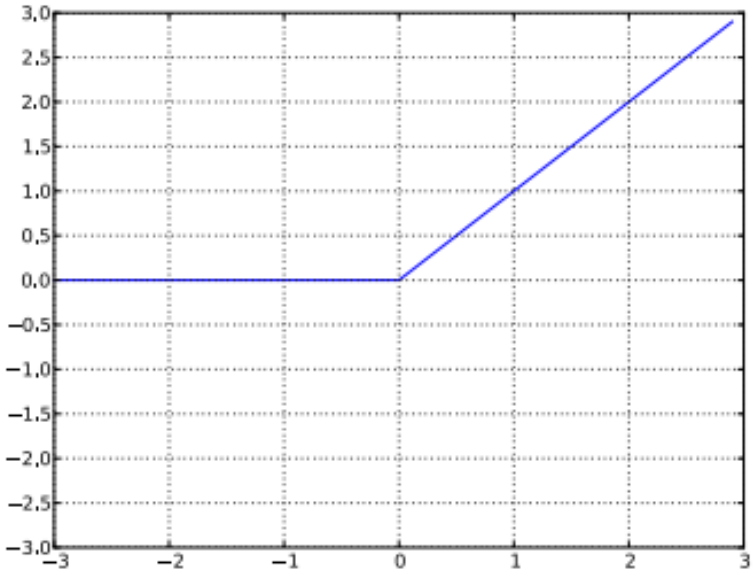
$y = tanh(H)$
$= (exp(H) - exp(-H))/(exp(H) + exp(-H))$

is a **hyperbolic tangent activation function**

-squashes the H value between -1 and 1
-can be positive or negative
-bounded

-strictly increasing

Neural networks Introduction: Digital neurons



$y = reclin(H) = max(0, H)$

is a **rectified linear activation function**

-squashes the H value between 0 and 1
-always positive
-not upper bounded

-strictly increasing

Neural networks Introduction: Digital neurons

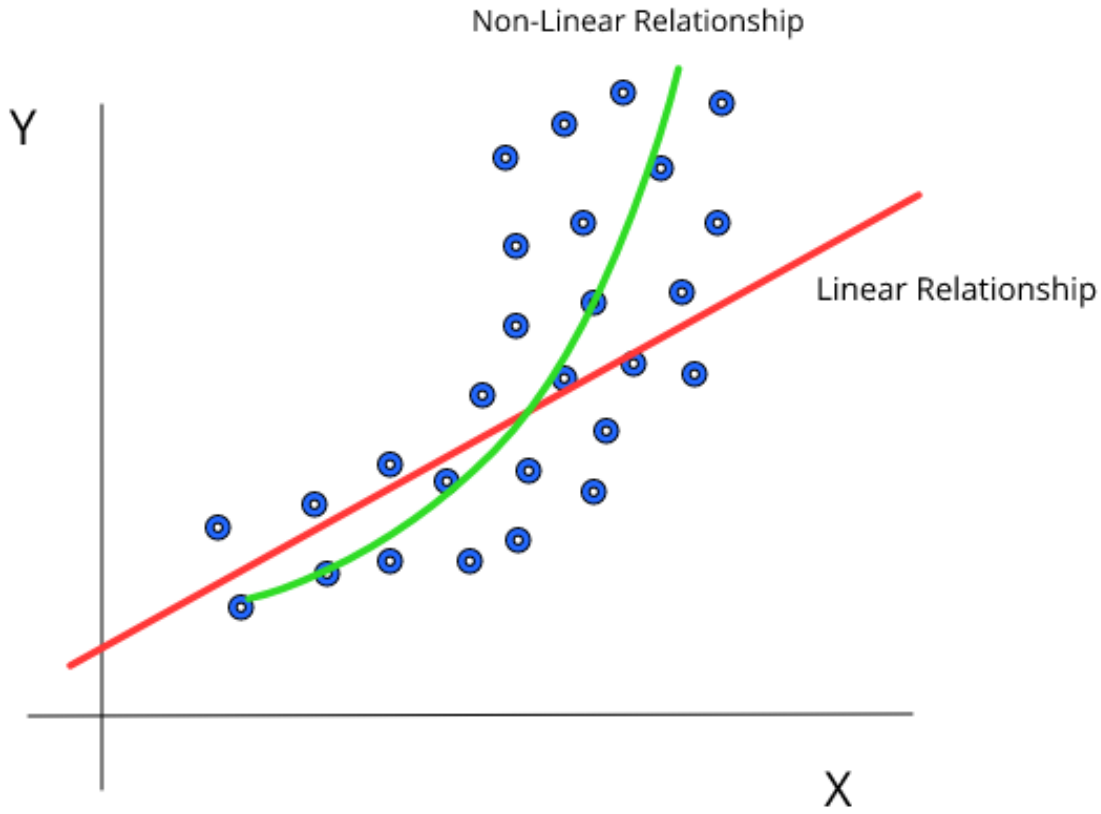
Without Activation Function

$$y = \sum_{i=0}^{n}(W_i * X_i) + B$$

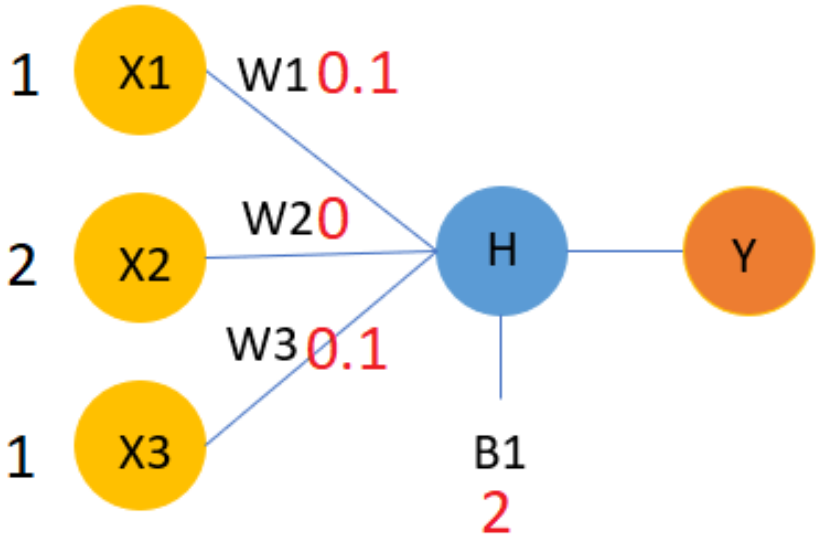
With Activation Function

$$y = f(\sum_{i=0}^{n}(W_i * X_i) + B)$$

Neural networks Introduction: Digital neurons



Neural networks Introduction: Digital neurons



What is the result?

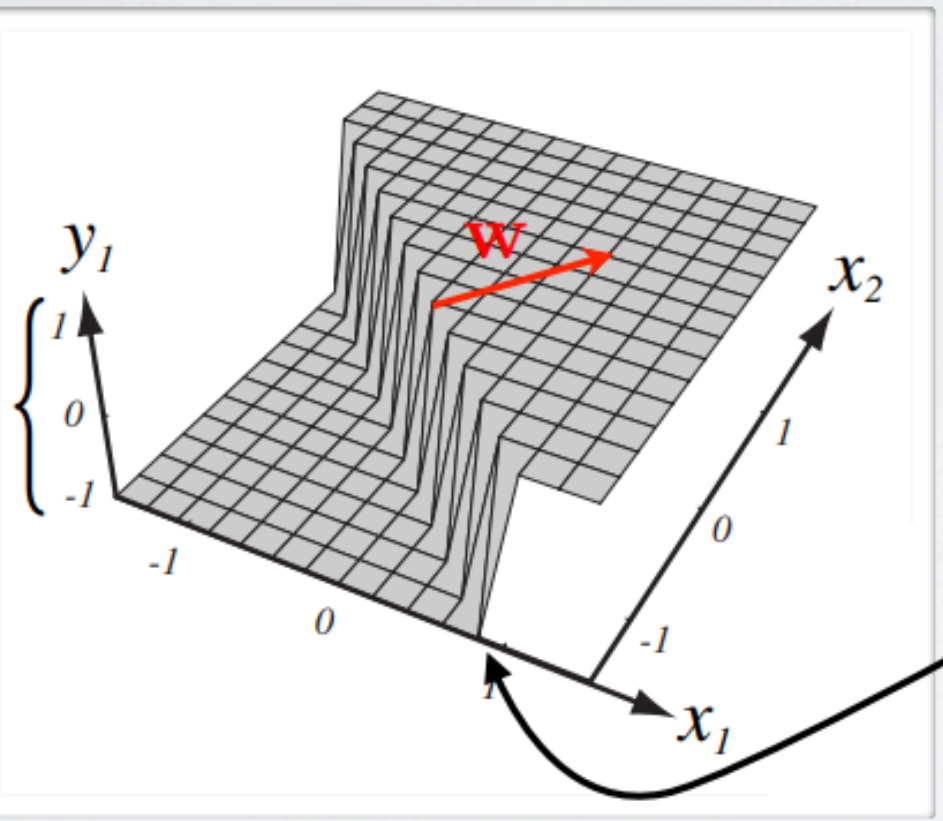
(assume we have a **rectified linear activation function**)

Can you change the result for the same input? How?

Feedforward neural networks

Capacity of a single neuron (perceptron)

range determined by $g(\cdot)$

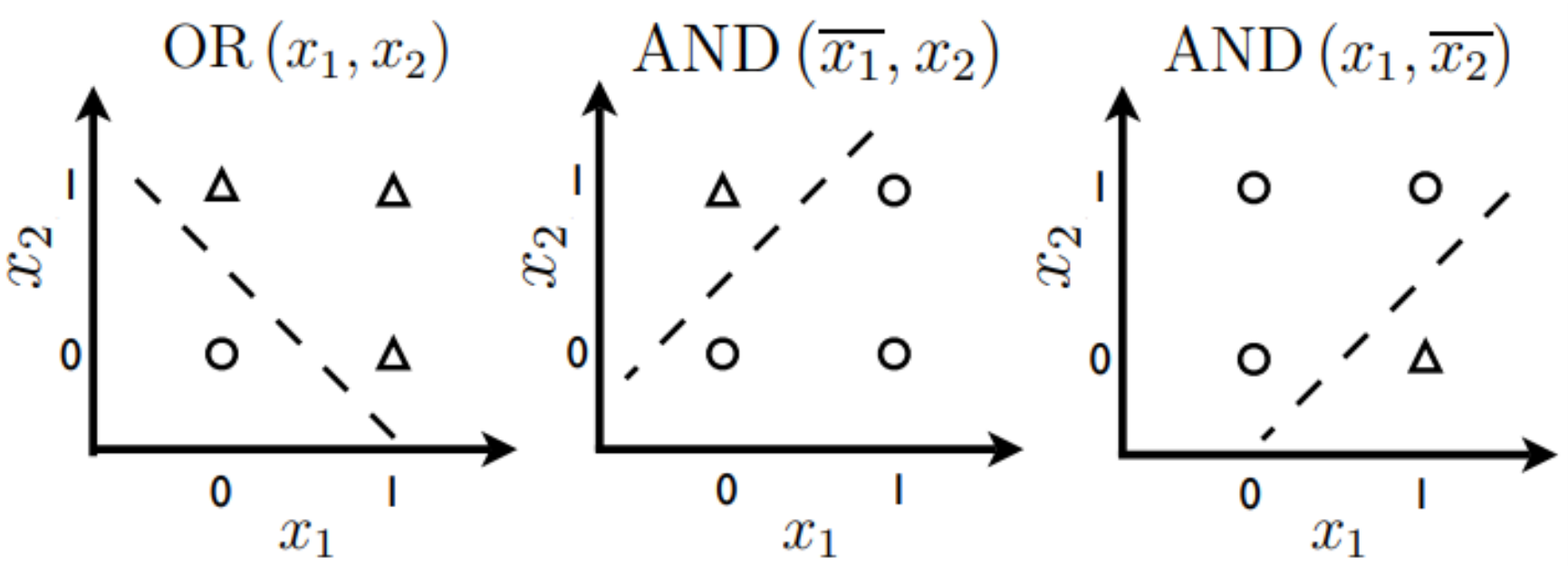


bias b only changes the position of the riff

(from Pascal Vincent's slides)

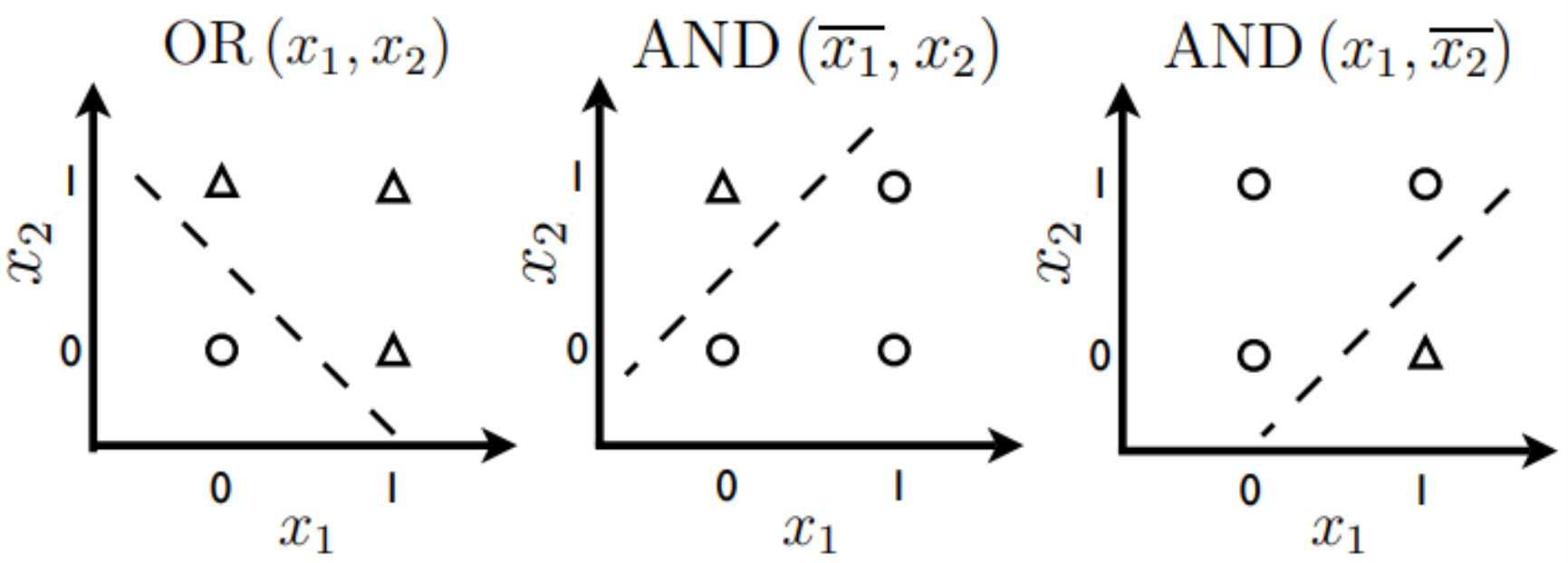
Feedforward neural networks

Capacity of a single neuron (perceptron)



Feedforward neural networks

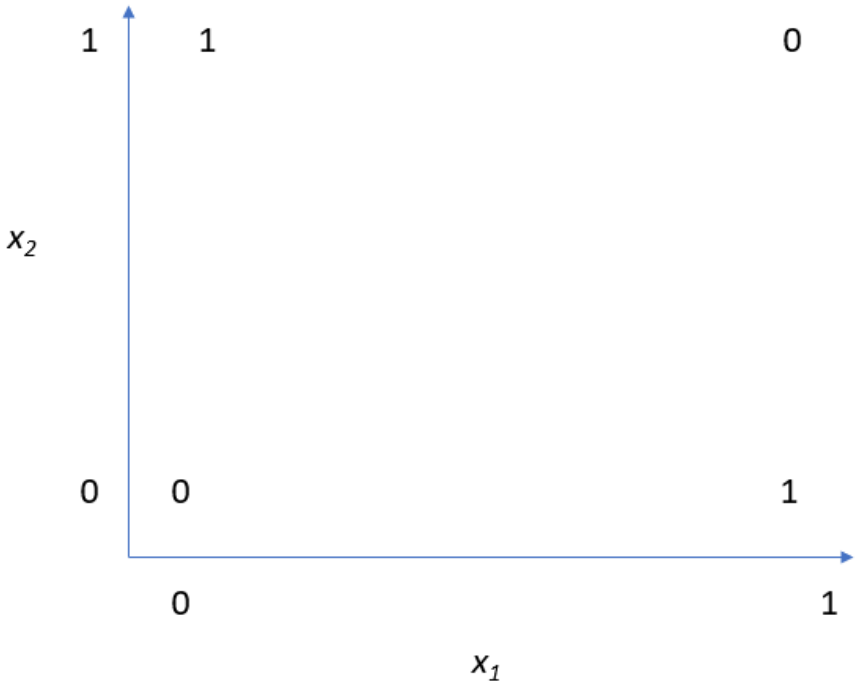
Capacity of a single neuron (perceptron)



Single neuron can only solve a linear/logistic regression problem.

Feedforward neural networks

Capacity of a single neuron (Can this be solved with 1 neuron?)



Feedforward neural networks

How can this be solved with a single neuron?

Input 1	Input 2	Output
0	0	0
0	1	1
1	1	0
1	0	1

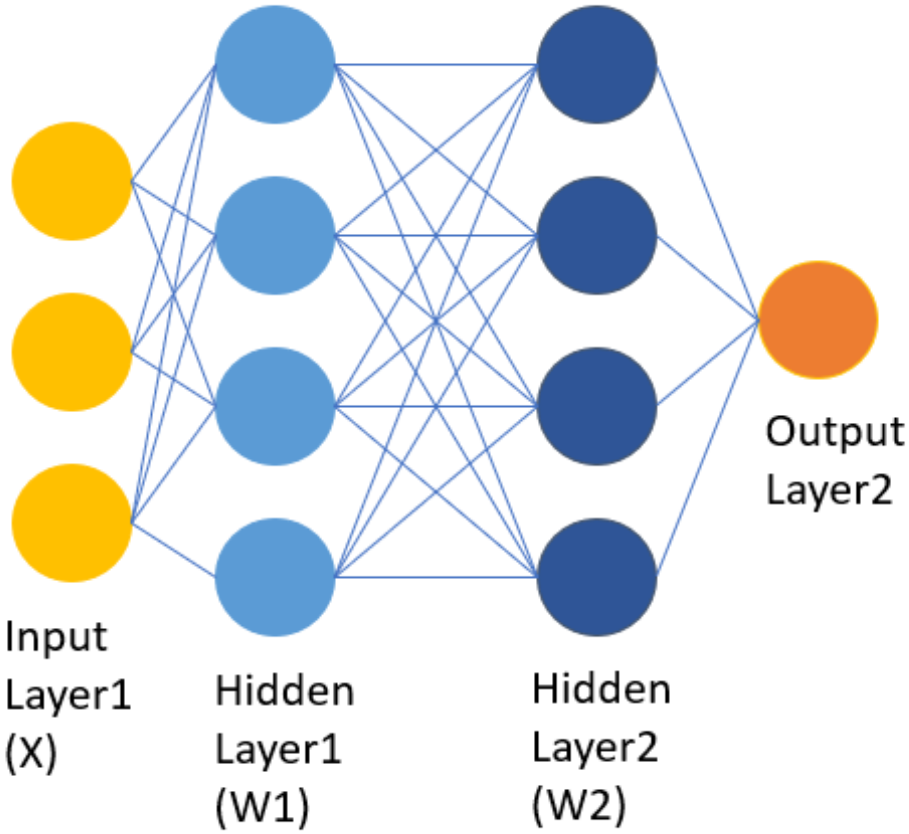
Feedforward neural networks

How can this be solved with a single neuron?

Input 1	Input 2	Output
0	0	0
0	1	1
1	1	0
1	0	1

One reason that neural networks idea died earlier is that; it was proved the XOR problem cannot be solved by a perceptron. (There is no way to solve this problem with a perceptron indeed!)

Feedforward neural networks



$y = f(x) = f^2(f^1(x))$

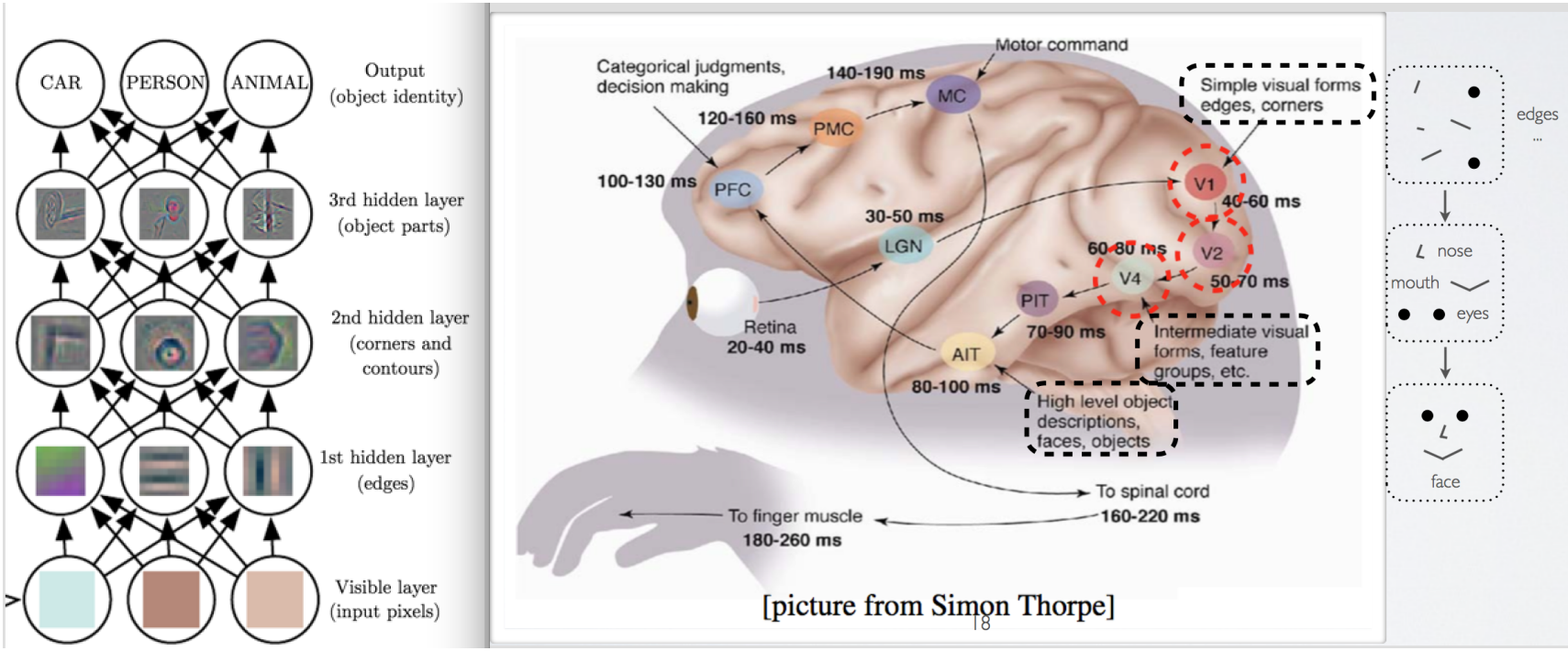
Number of $f(.)$ functions, tell about the **depth** of the neural network.

Feedforward neural networks

There are no feedback connections in which outputs of the model are fed back into itself.

When feedforward neural networks are extended to include feedback connections, they are called **recurrent neural networks (RNN)**.

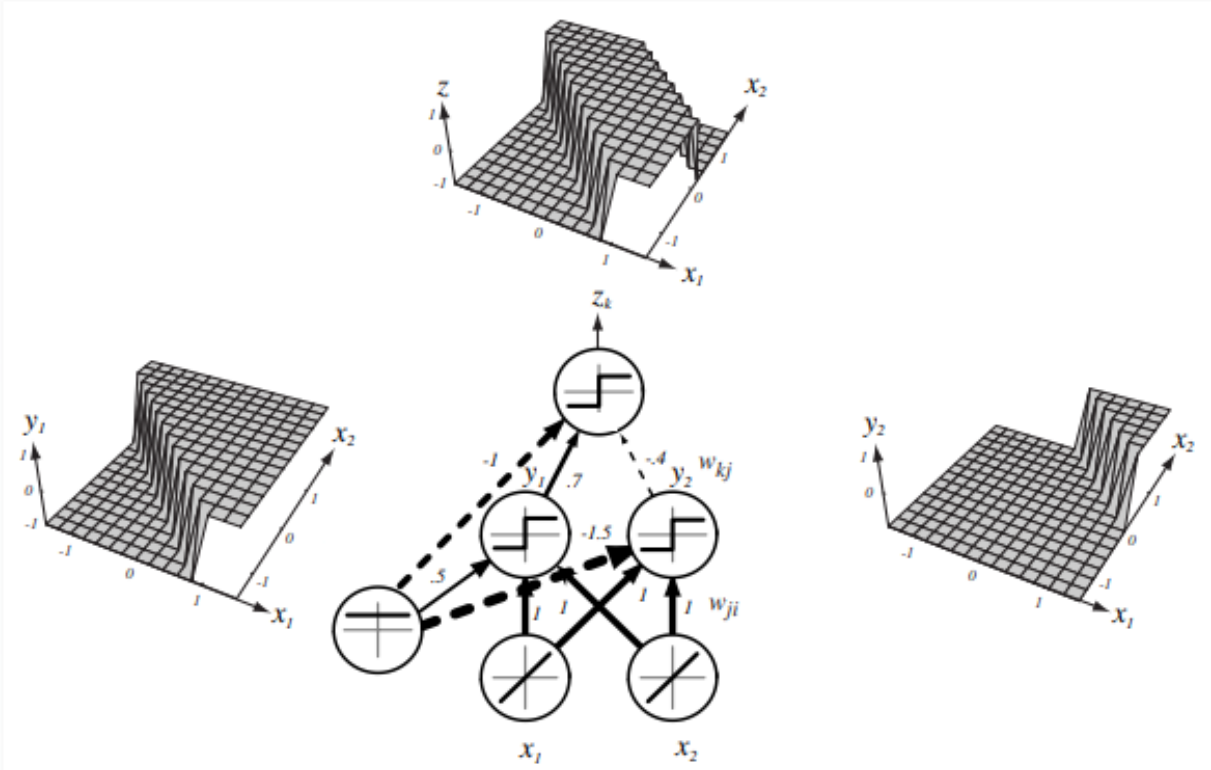
Feedforward neural networks



picture: Hugo Larochelle

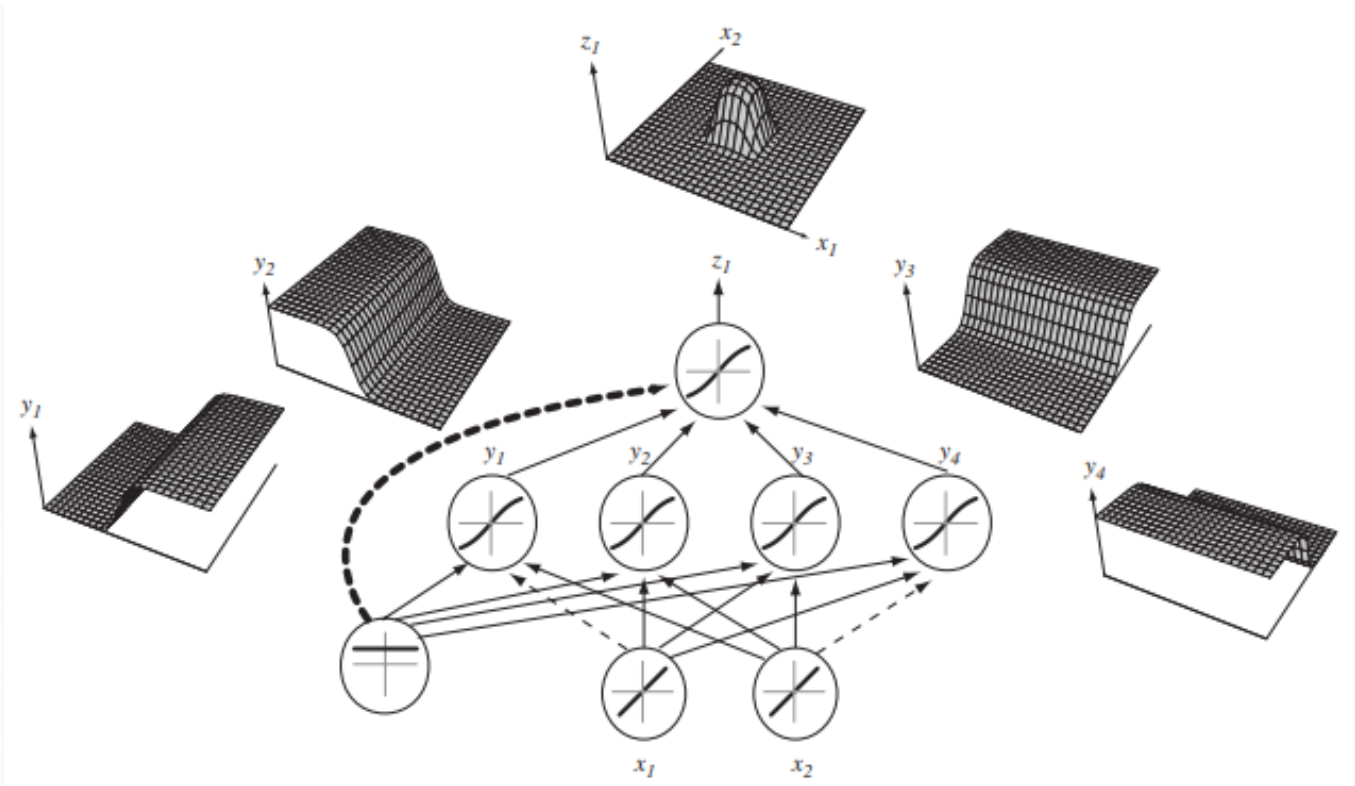
Feedforward neural networks

We need to combine neurons (just like our brains!)



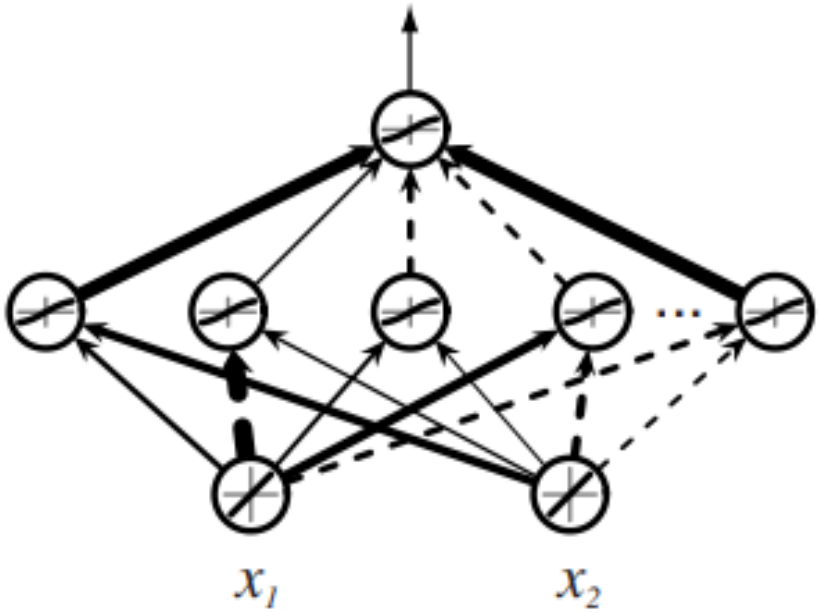
Feedforward neural networks

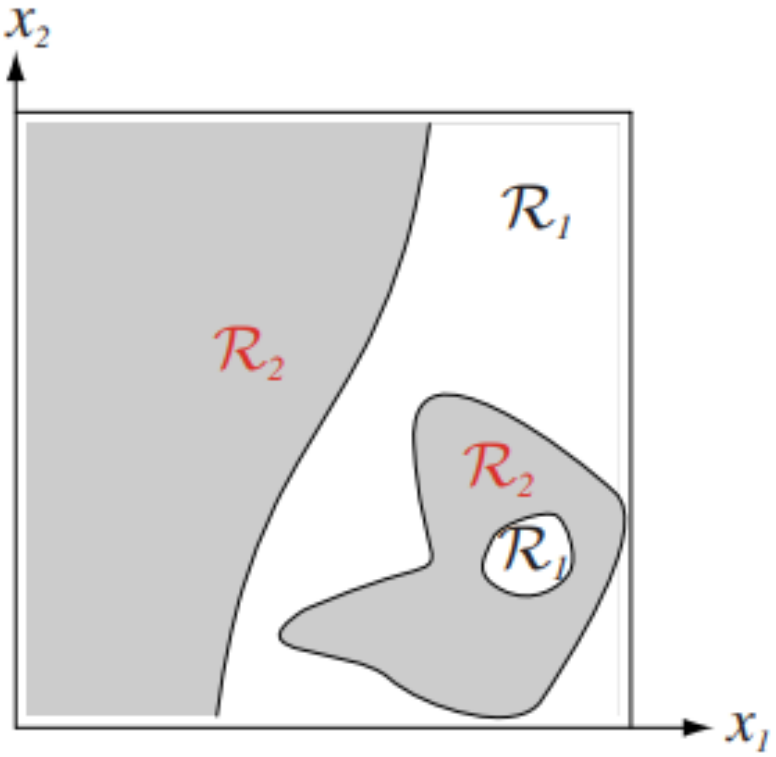
We need to combine neurons (just like our brains!)



Feedforward neural networks

We need to combine neurons (just like our brains!)





(Good result, but it doesn't mean that the neural network has learned something!)

Feedforward neural networks

"A single hidden layer neural network with a **linear output unit** can approximate **any continuous function** arbitrary well, given **enough hidden layers**."

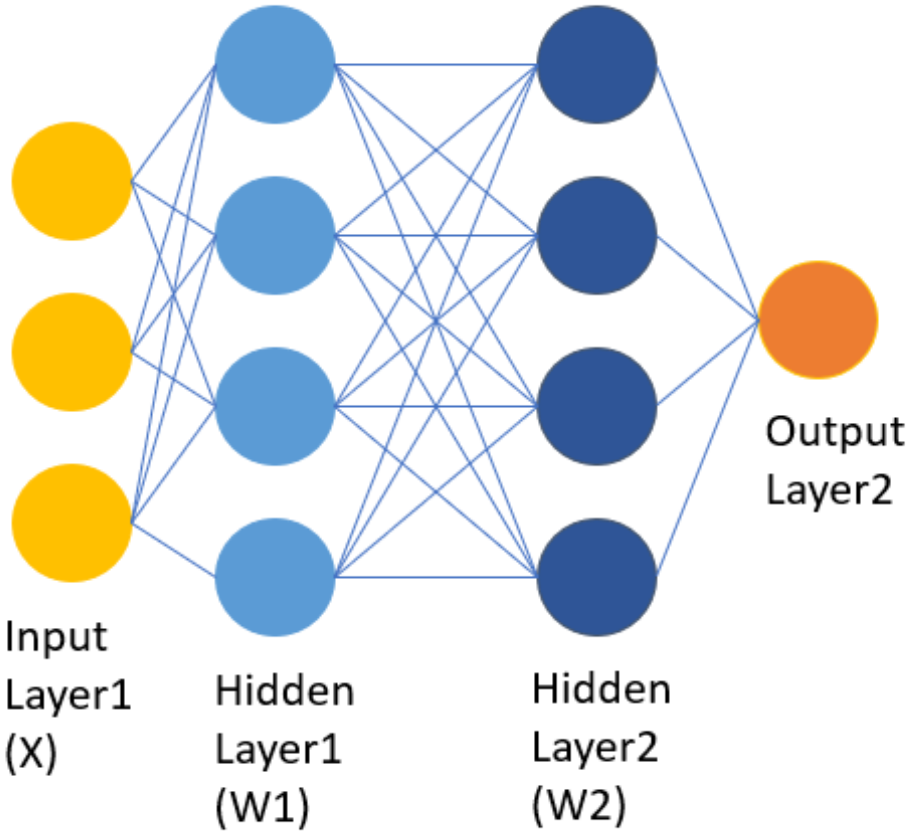
Hornik, 1991

Feedforward neural networks

Deep feedforward networks (or feedforward neural networks) are the **learning** models.

The goal of a feedforward neural network is to approximate some function for f^*, where $y = f(x)$ maps input x to a category y (if it is a classification problem).

Feedforward neural networks



$y = f(x) = f^2(f^1(x))$

Number of $f(.)$ functions, tell about the **depth** of the neural network.

Feedforward neural networks

A feedforward network defines a mapping $y = f(x; \theta)$ and **learns** the value of the parameters θ that result in the best function approximation.

Feedforward neural networks

Where is learning?

Feedforward neural networks

$$C(\theta) = \sum^{N}(f^*(x) - f(x;\theta))$$

N is the number of the training samples

learning means doing adjustments to make less errors

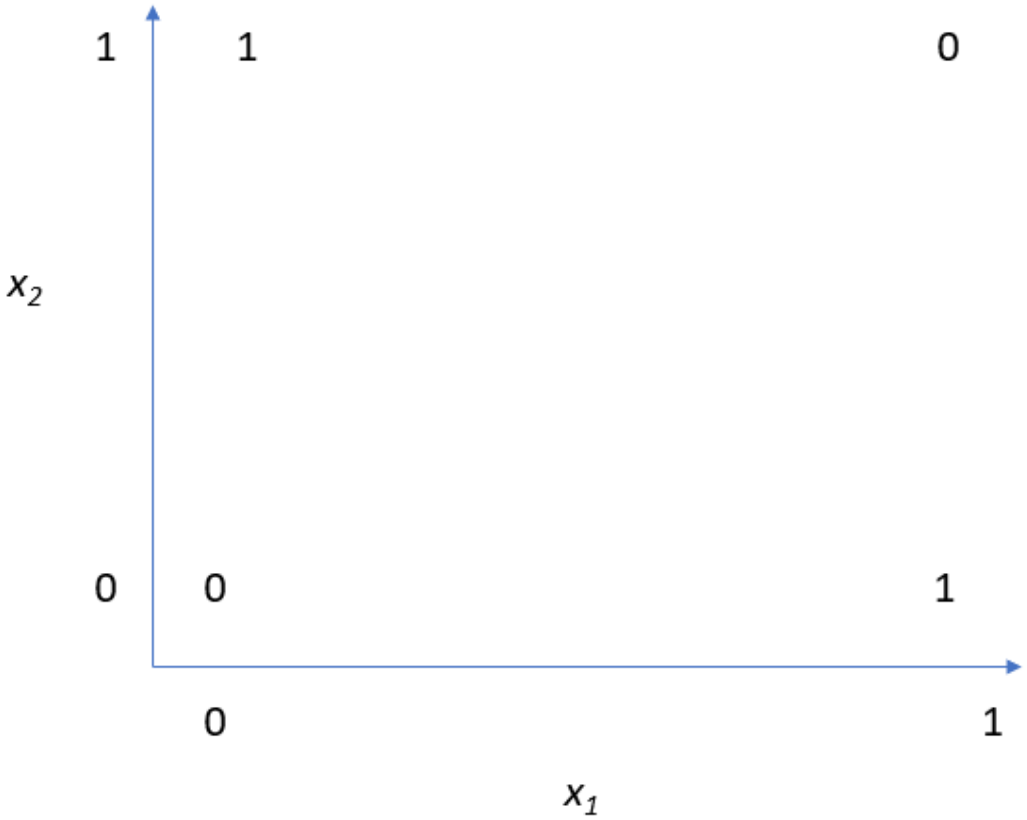
Feedforward neural networks

$$C(\theta) = \sum^{N}(f^*(x) - f(x;\theta))$$

$C(\theta)$ is called **loss function**

Feedforward neural networks

Solving XOR problem with a feedforward neural network



Feedforward neural networks

Solving XOR problem with a feedforward neural network

$X = \{[0,0]^T, [0,1]^T, [1,0]^T, [1,1]^T\}$

Feedforward neural networks

Mean Square Error (MSE) cost function is

$$C(\theta) = (1/4) \sum (f^*(x) - f(x;\theta))^2$$

Feedforward neural networks

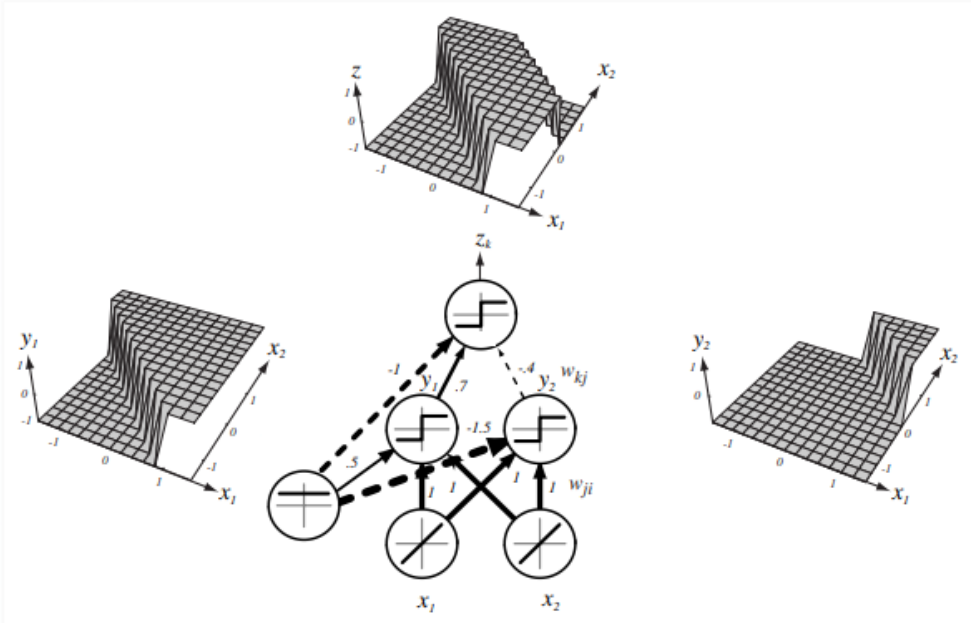
Now we have to model $f(x;\theta)$. We can choose a linear model θ consisting of w and b

$$f(x;w,b) = x^T w + b$$

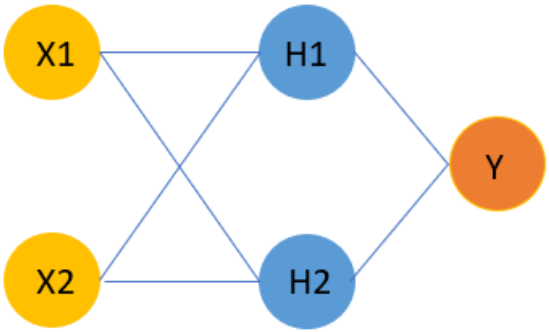
solving the equations for the training input, we find $w = 0$ and $b = 1/2$. The linear model outputs 0.5 everywhere.

Feedforward neural networks

We need to add one hidden layer to combine two linear responses;



Feedforward neural networks



Now, the linear regression model is applied to h rather than to x

Feedforward neural networks

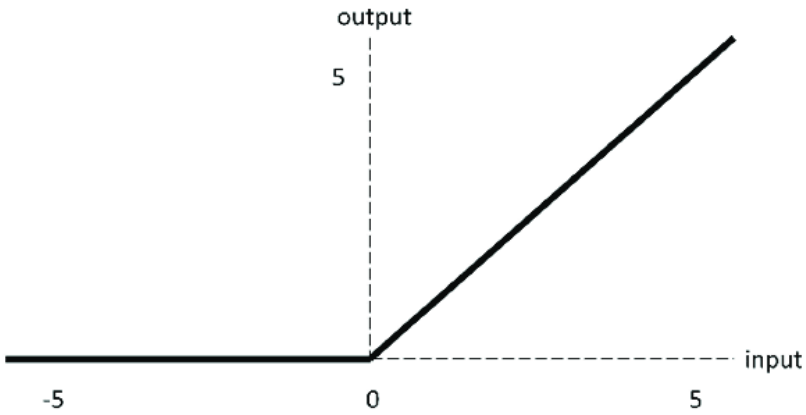
$$h_i = g(x^T W + b_i)$$

(b_i are the biases)

Now, the linear regression model is applied to h rather than to x

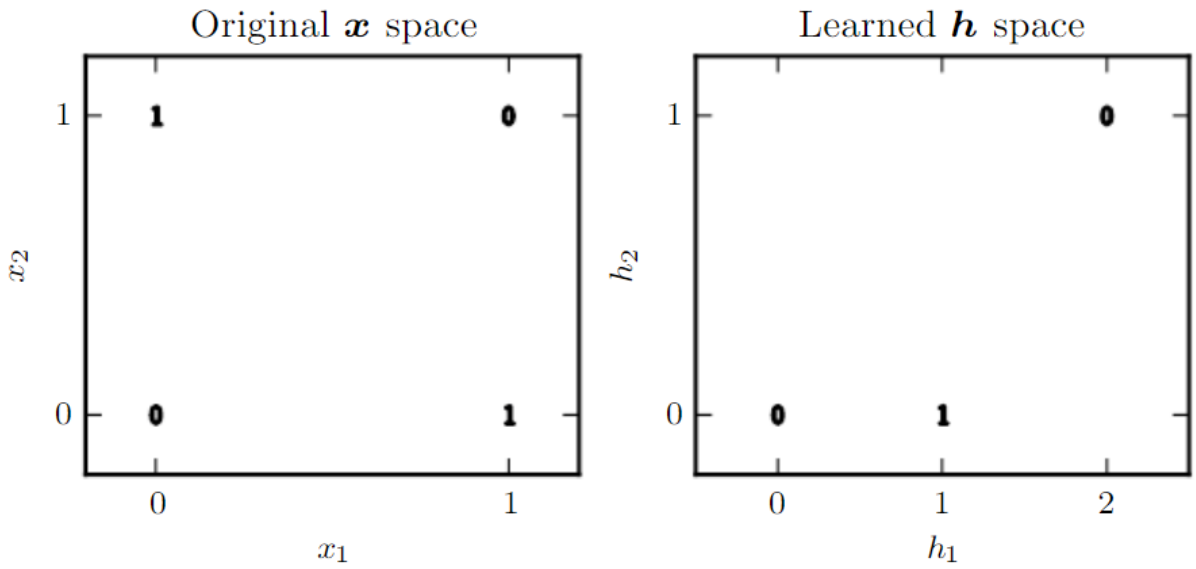
Feedforward neural networks

Default recommendation for the $g()$ is to use a **rectified linear unit (RELU)** activation function



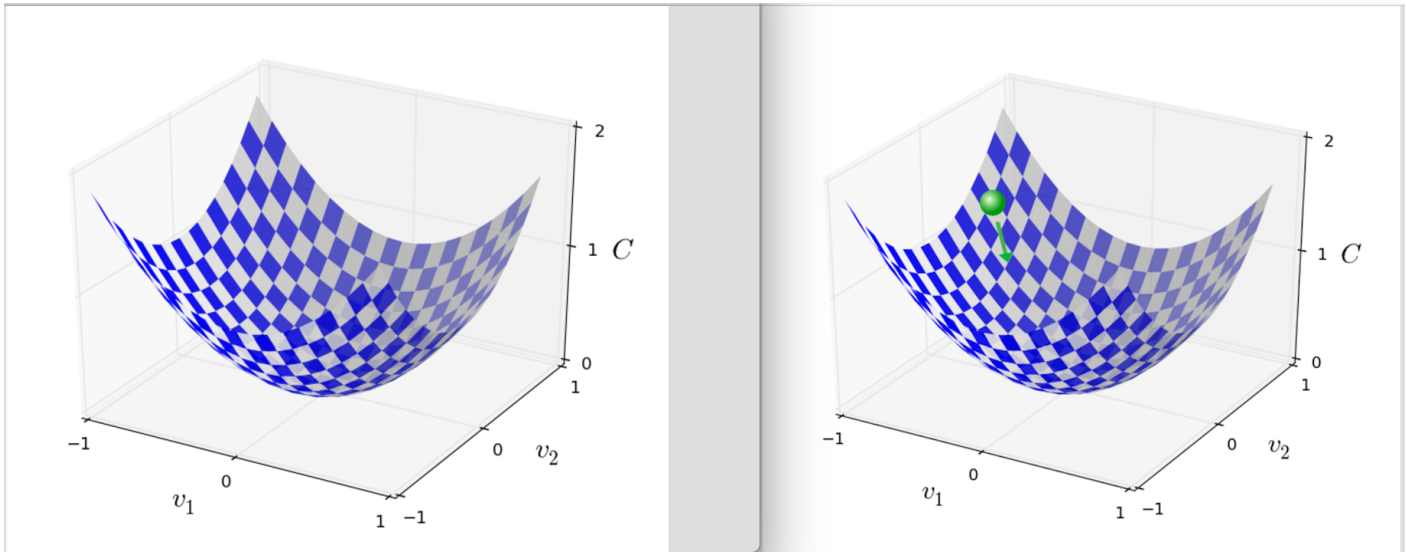
Feedforward neural networks

The learned h space looks like:



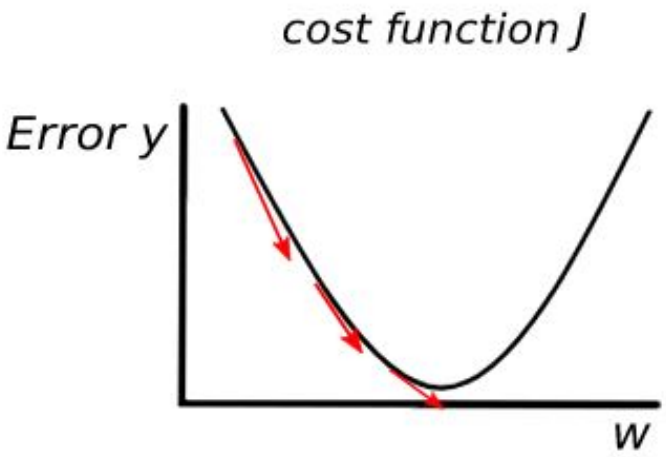
Gradient-based learning

Finding the optimal parameters
(once we have a training data set and a cost function):



Gradient-based learning

Finding the optimal parameters

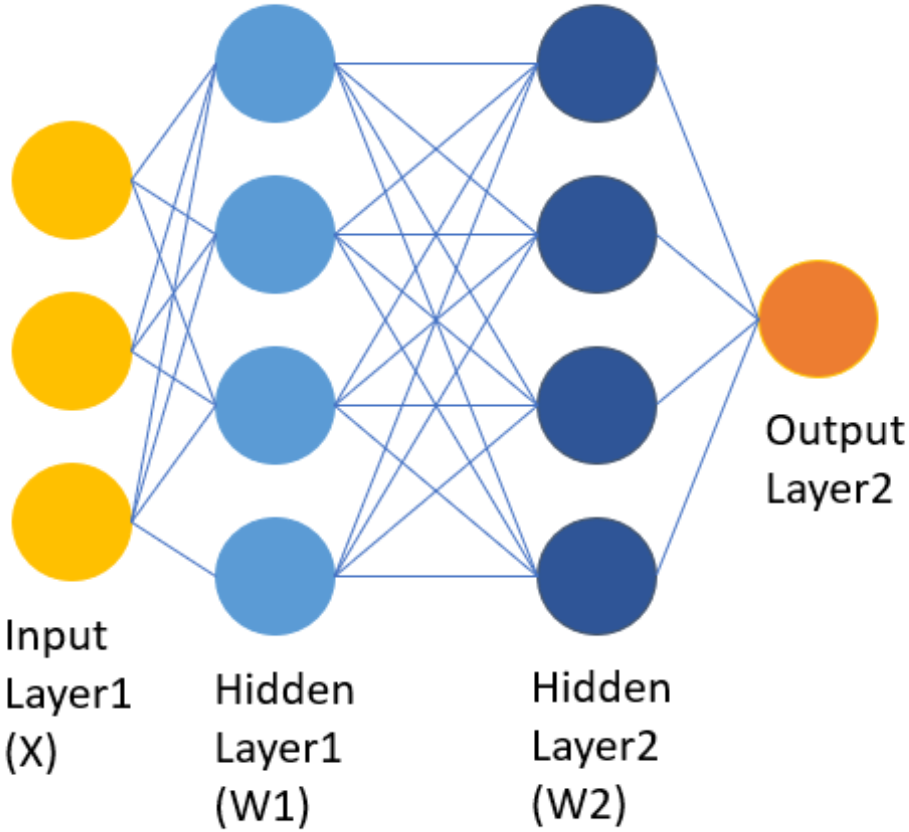


$\frac{dJ}{dW}$ *get the direction to move towards local minimum

the derivative d of the J function with respect to each weight

Back-propagation

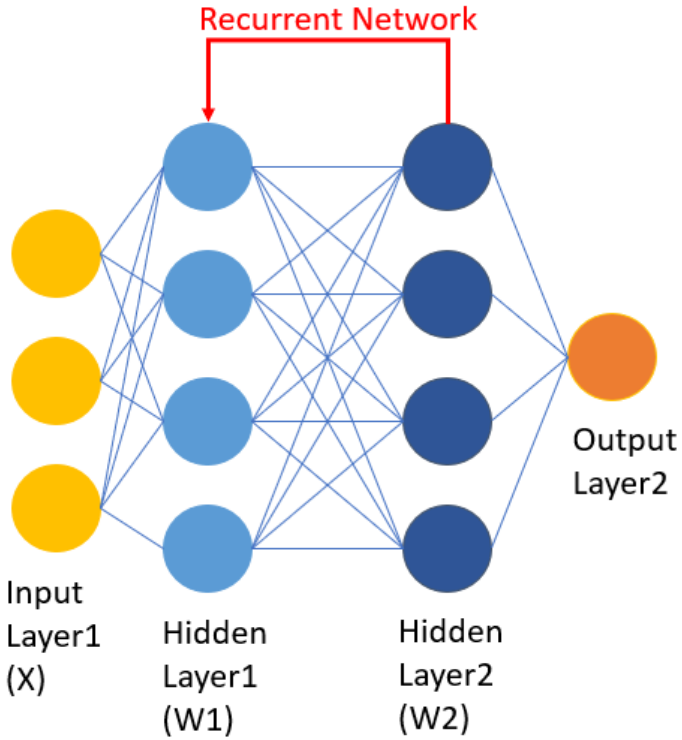
We said; we don't have cycles in a feed-forward network:



Neurons are not aware of the earlier training examples!

Back-propagation

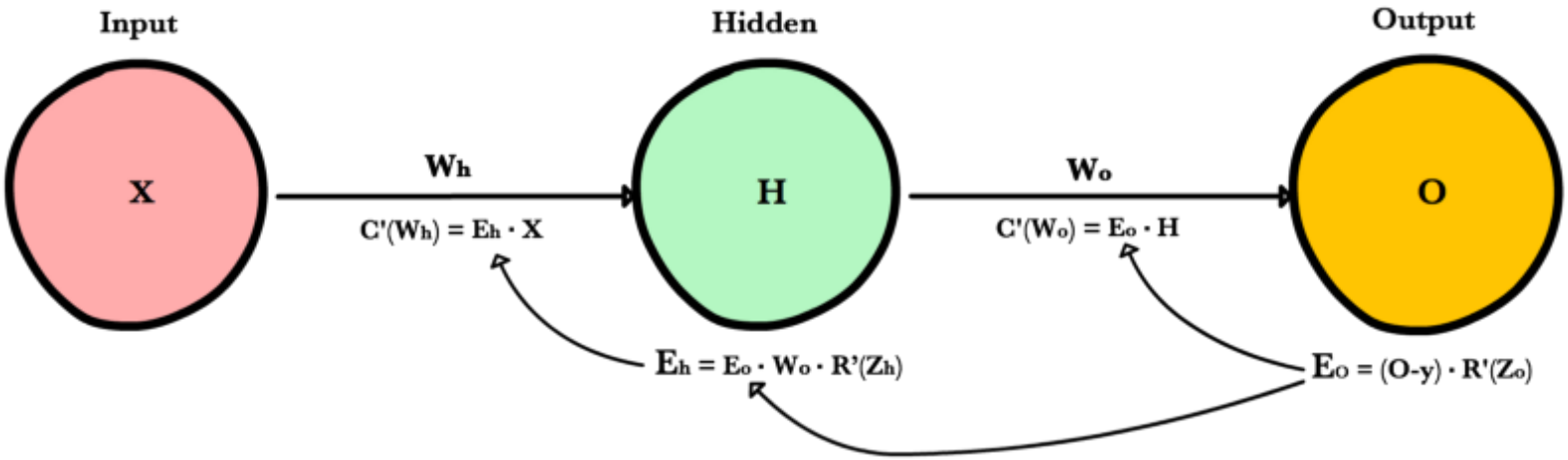
If there are feed-back cycles



This network REMEMBERS the past training examples.

1) Feed forward the values 2) calculate the error and propagate it back to the earlier layers

Back-propagation



The goals of backpropagation are straightforward: adjust each weight in the network in proportion to how much it contributes to overall error.

If we iteratively reduce each weight's error, eventually we'll have a series of weights that produce good predictions.

Back-propagation

Given a forward propagation function:

$$f(x) = A(B(C(x)))$$

$A()$, $B()$, $C()$ are the activation functions of the layers.

Using the chain rule we easily calculate the derivative of $f(x)$:

$$f'(x) = f'(A) \cdot A'(B) \cdot B'(C) \cdot C'(x)$$

Back-propagation

Using the chain rule we can easily find the derivative of Cost with respect to weight W:

$$\begin{aligned} C'(W) &= C'(R) \cdot R'(Z) \cdot Z'(W) \\ &= (\hat{y} - y) \cdot R'(Z) \cdot X \end{aligned}$$

What is the derivative of cost with respect to Wo?

$$\begin{aligned} C'(W_O) &= C'(\hat{y}) \cdot \hat{y}'(Z_O) \cdot Z'_O(W_O) \\ &= (\hat{y} - y) \cdot R'(Z_O) \cdot H \end{aligned}$$

Back-propagation

To calculate hidden layer error we need to find the derivative of cost with respect to the hidden layer input, Z_h:

$$C'(Z_h) = (\hat{y} - y) \cdot R'(Z_o) \cdot W_o \cdot R'(Z_h)$$

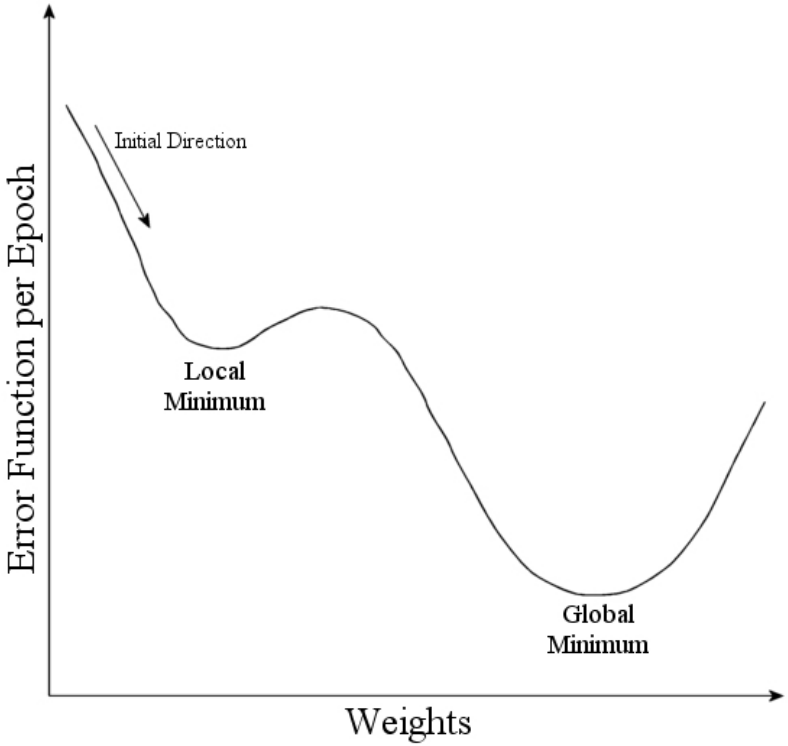
Next we can swap in the E_o term above to avoid duplication and create a new simplified equation for Hidden layer error.

$$E_h = E_o \cdot W_o \cdot R'(Z_h)$$

$$C'(w) = CurrentLayerError \cdot CurrentLayerInput$$

Back-propagation

Local vs. Global minimums:



Back-propagation

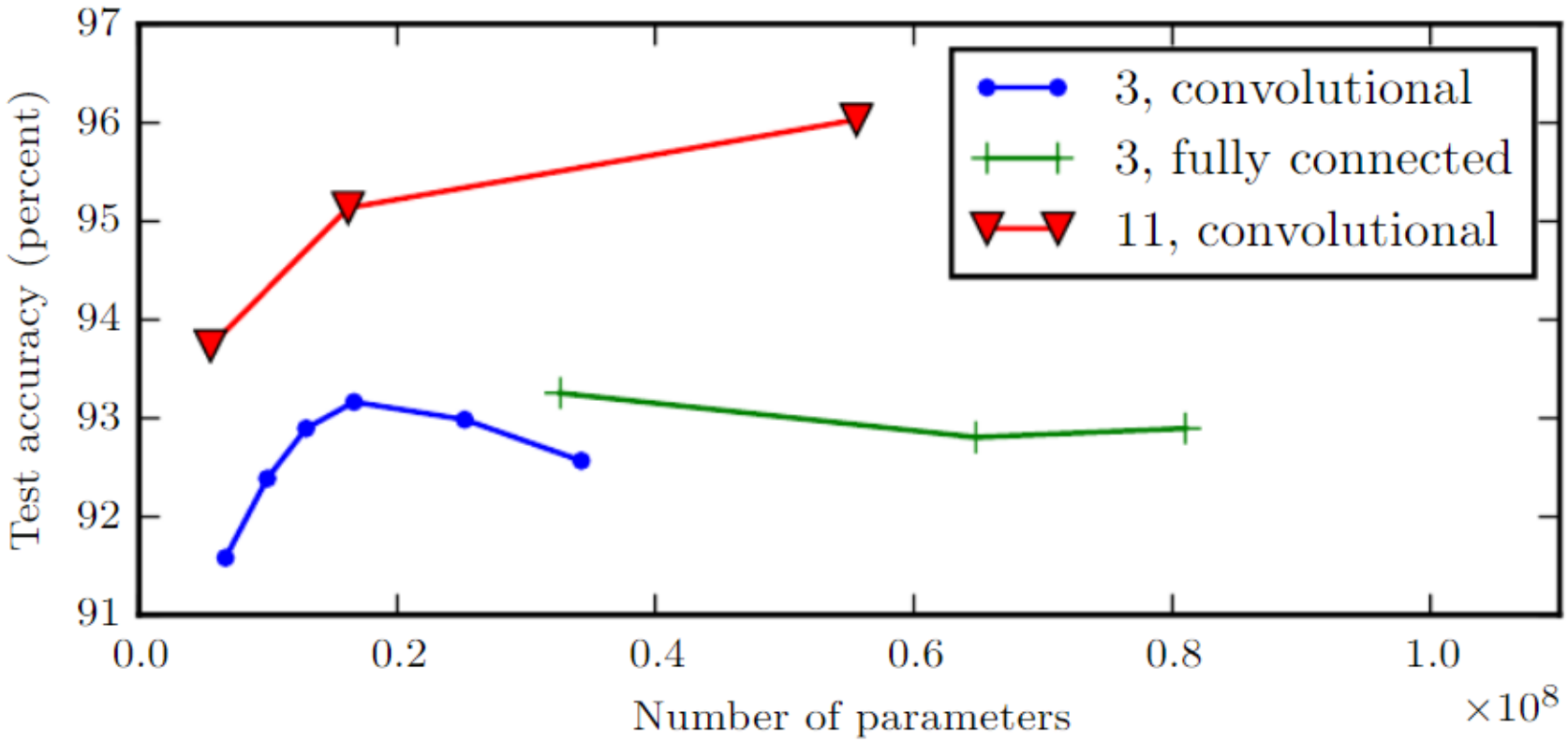
Back-propagation

```
def relu_prime(z):
    if z > 0:
        return 1
    return 0

def cost(yHat, y):
    return 0.5 * (yHat - y)**2

def cost_prime(yHat, y):
    return yHat - y

def backprop(x, y, Wh, Wo, lr):
    yHat = feed_forward(x, Wh, Wo)

    # Layer Error
    Eo = (yHat - y) * relu_prime(Zo)
    Eh = Eo * Wo * relu_prime(Zh)

    # Cost derivative for weights
    dWo = Eo * H
    dWh = Eh * x

    # Update weights
    Wh -= lr * dWh
    Wo -= lr * dWo
```

Network modelling: what is an architecture?

- How many neurons?
- How many layers?
- How are they connected?
- What are the initial weights?
- What is the activation function?
- How many output neurons?

Network modelling: how many layers



Network modelling: knowing when to stop

Network modelling: regularization

"Regularization is a subset of methods used to encourage generalization in learned models, often by increasing the difficulty for a model to learn the fine-grained details of the training data."

Network modelling: regularization

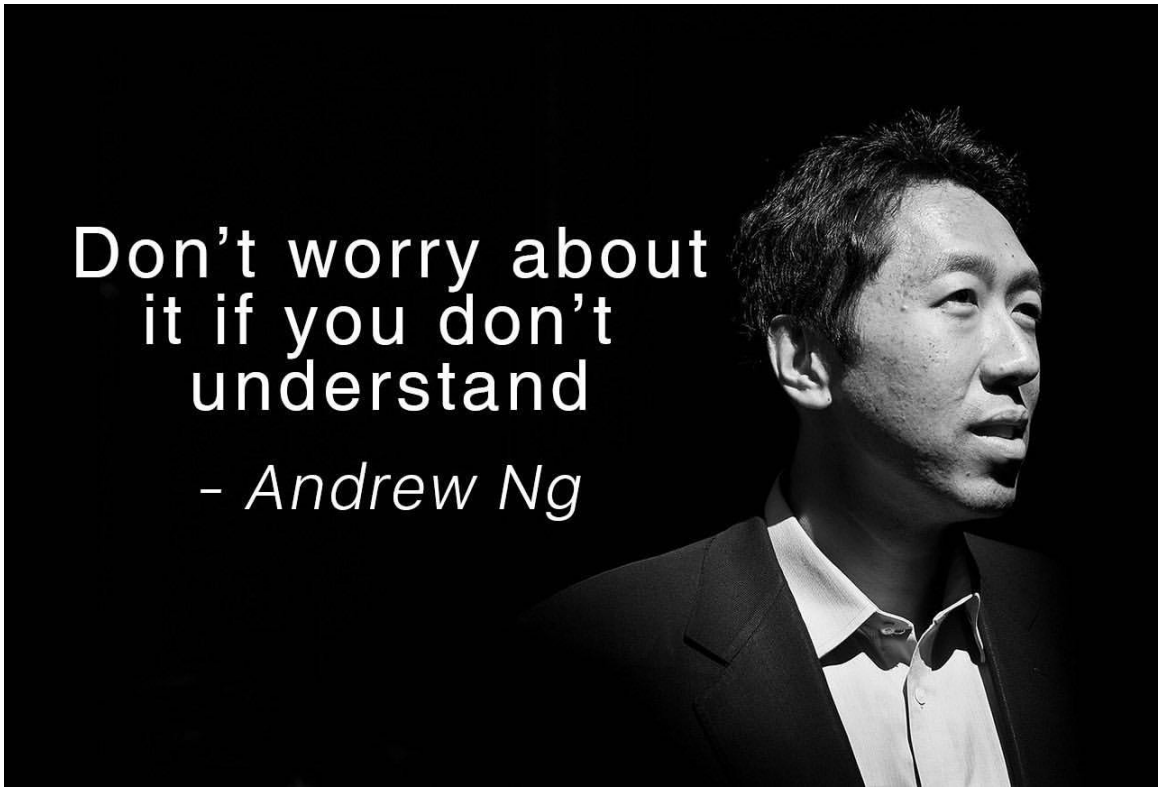
- Early stopping
- Drop out
- Adding noise to the weights
- Data augmentation

Network modelling: neural networks and deep learning

It is possible to use neural networks for;

- Supervised learning
- Semi-supervised learning
- Unsupervised learning
- Reinforcement learning

Network modelling: the end



References I