Neural Network

Instructor, Nero Chan Zhen Yu

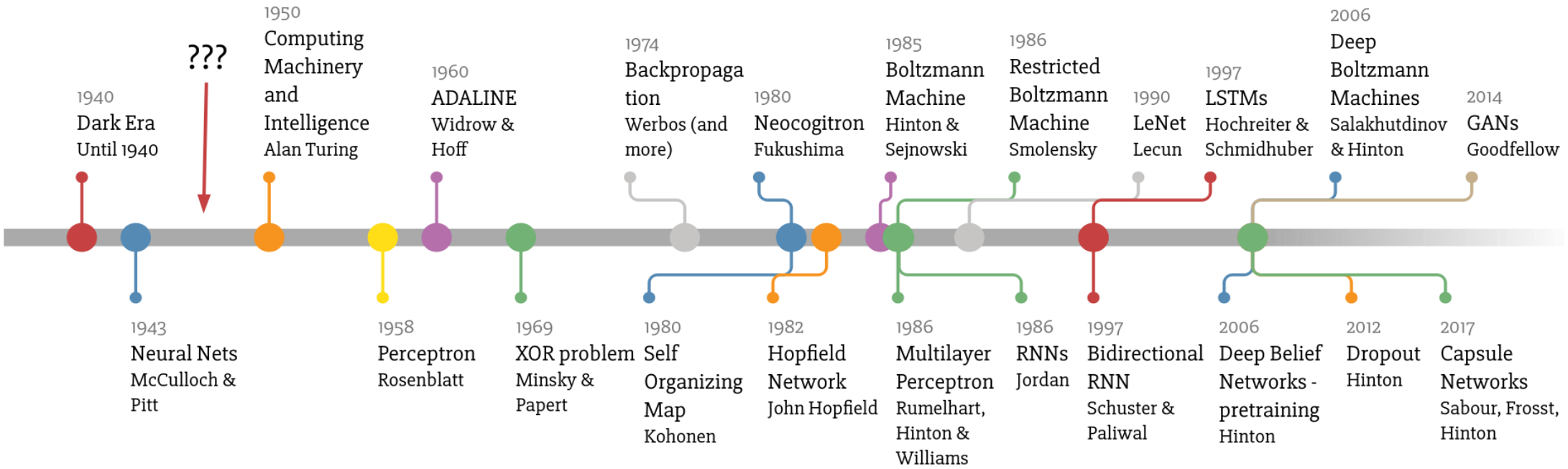
Forward School

Neural Network

- One of the hottest AI/ML techniques now
  - Especially with all the developments around self-driving cars, robots and advanced computer vision
- Why is it so “hot” now? (and not 20 years ago?)

Forward School

Deep Learning Timeline

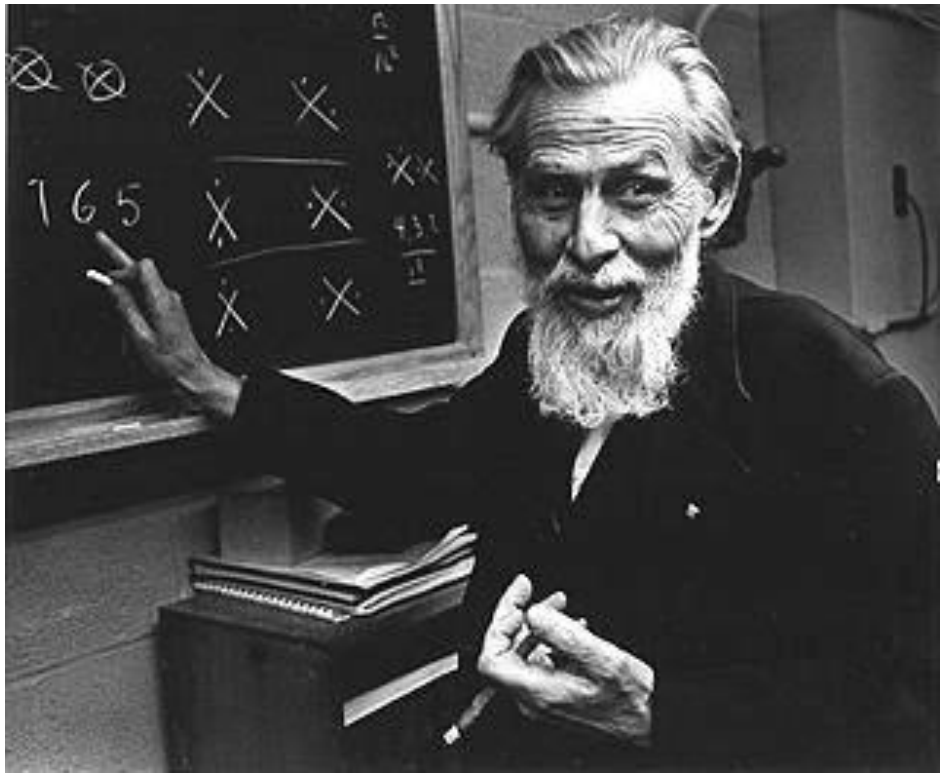


Made by Favio Vázquez

Forward School

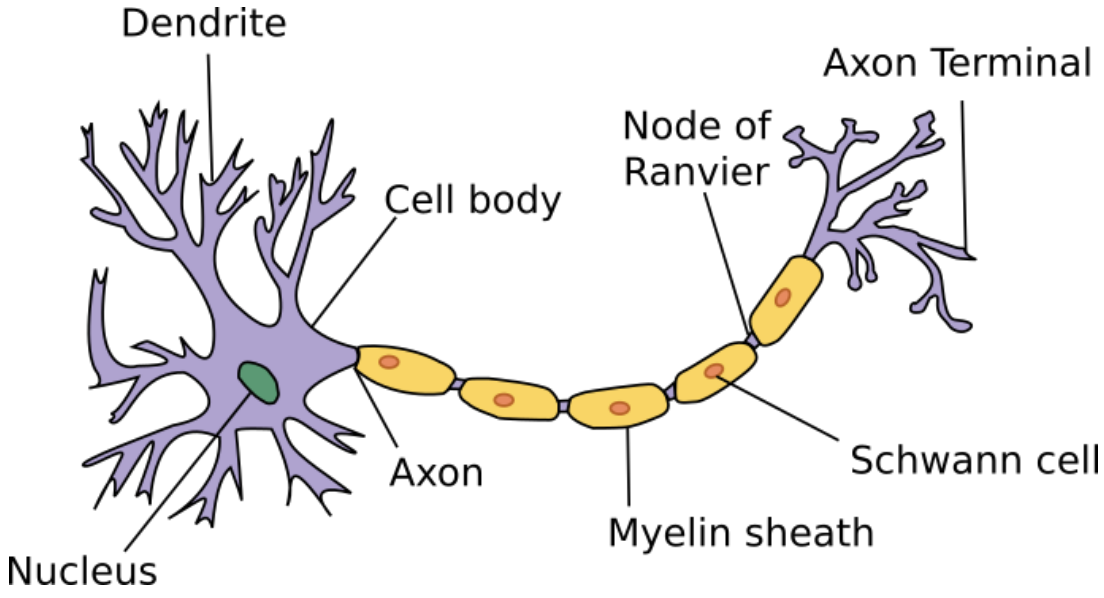
History of Neural network

- It all begun with Warren S. McCulloch (1898–1969, American neurophysiologist) and Walter H. Pitts Jr (1923–1969, American logician) portrayed a model of a human brain neuron



Forward School

Human Neuron

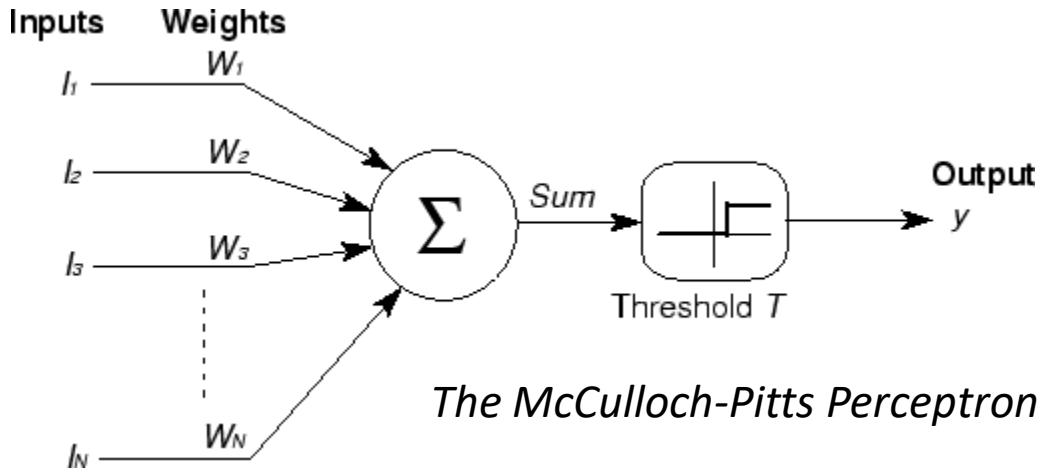


- The operating principle of a biological neuron can be summarized as follows.
  - First, it takes inputs from its dendrites (i.e. from other neurons).
  - In a second step, a weighted sum of these input is performed within the soma.
  - The result is then passed on to the axon hillock. If this weighted sum is larger than the threshold limit, the neuron will fire.
  - Otherwise, it stays at rest. The state of our neuron (on or off) then propagates through its axon and is passed on to the other connected neurons via its synapses.
- Albeit very simple, this high-level description of the operating principle of a biological neuron is sufficient to understand the mathematical model of an artificial neuron proposed by McCulloch & Pitts in 1943.

Forward School

The mathematical model

- A simple electronic circuit which took a set of inputs, multiplied them by weighted values and put them through a threshold gate which gave as output a value of 0 or 1, based on the threshold value.



The McCulloch-Pitts Perceptron

Forward School

The MCP neuron

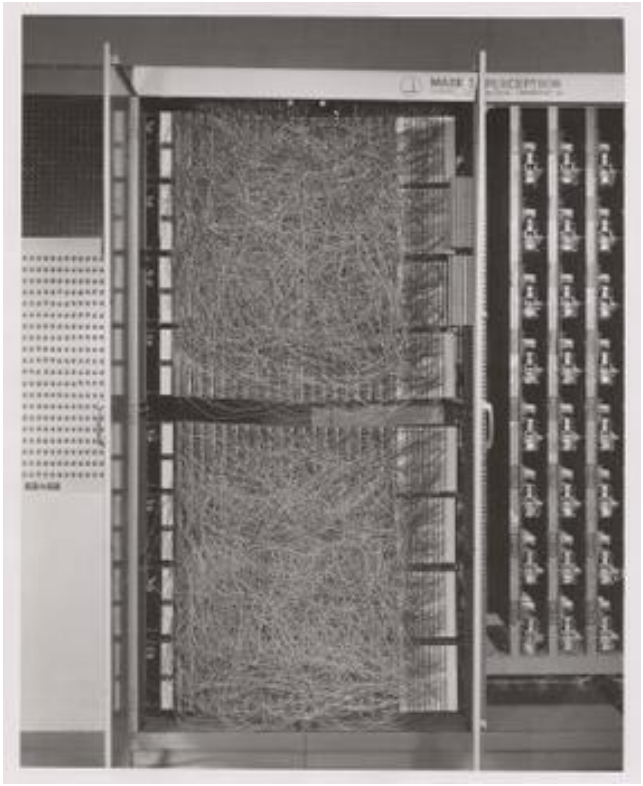
- Defined by the following rules
  - It has a binary output $y \in \{0, 1\}$, where y=1 indicates that the neuron fires and y=0 that it is at rest.
  - It has a number N of exitatory binary inputs $x_k \in \{0, 1\}$.
  - It has a single inhibitory input i. If it is on, the neuron cannot fire.
  - It has a threshold value Θ. If the sum of its inputs is larger than this critical value, the neuron fires. Otherwise it stays at rest.
- Given the input $\boldsymbol{x} = [\, x_1, x_2, x_3, \ldots, x_n \,]^T$, the inhibitory input i and the threshold Θ, the output y is computed as follows

$$\sigma(\mathbf{x}) = \begin{cases} 1 \text{ if } \sum_{k=1}^{n} x_k > \Theta \text{ and } i = 0, \\ 0 \text{ otherwise.} \end{cases}$$

Forward School

Rosenblatt's single layer perceptron (1957)

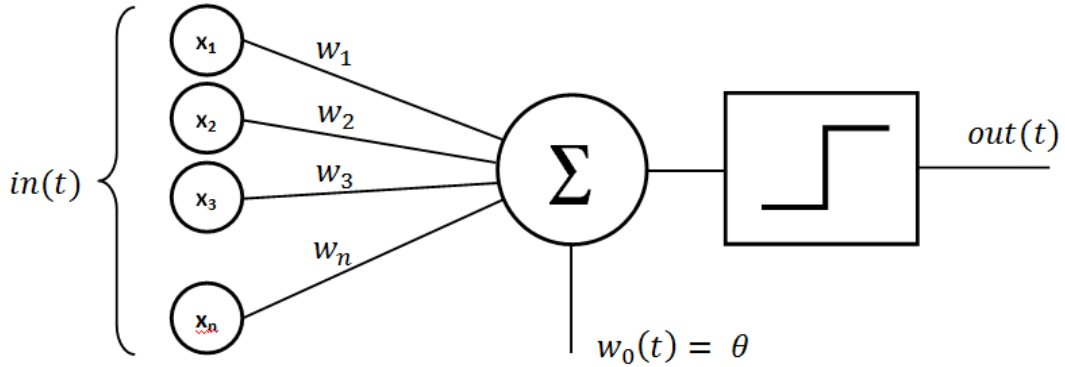
- Frank Rosenblatt (1928–1971), inspired by the Hebbian theory of synaptic plasticity (i.e. the adaptation of brain neurons during the learning process)
  - Came up with the idea of a perceptron
- By relaxing some of the MCP rules, the artificial neurons could actually learn from data
- More importantly, he came up with a supervised learning algorithm for this modified MCP neuron model that enabled the artificial neuron to figure out the correct weights directly from training data by itself



Forward School

Artificial neuron used by the perceptron

- Differences:
  - The neuron takes an extra constant input associated to a synaptic weight *b* (denoted Θ in the figure above), also known as the bias
  - The synaptic weights w_k are not restricted to unity, thus allowing some inputs to have more influence onto the neuron's output than others.
  - They are not restricted to be strictly positive either. Some of the inputs can hence have an inhibitory influence.
  - The absolute inhibition rule no longer applies



Forward School

Mathematically

- Non-linearity of the artificial neuron on which the perceptron relies is:

$$\sigma(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} 1 \text{ if } \mathbf{w}^T\mathbf{x} + b \geq 0, \\ 0 \text{ otherwise.} \end{cases}$$

- Compare to the MCP neuron:

$$\sigma(\mathbf{x}) = \begin{cases} 1 \text{ if } \sum_{k=1}^{n} x_k > \Theta \text{ and } i = 0, \\ 0 \text{ otherwise.} \end{cases}$$

Forward School

Perceptron learning algorithm

Algorithm 1: Perceptron Learning Algorithm

Input: Training examples $\{\mathbf{x}_i, y_i\}_{i=1}^{m}$.

Initialize $\mathbf{w}$ and b randomly.

while *not converged* **do**

- # # # Loop through the examples.
- **for** $j = 1, m$ **do**
  - # # # Compare the true label and the prediction.
  - $error = y_j$ - $\sigma(\mathbf{w}^T \mathbf{x}_j + b)$
  - ### If the model wrongly predicts the class, we update the weights and bias.
  - **if** *error != 0* **then**
    - ### Update the weights.
    - $\mathbf{w} = \mathbf{w} + error \times x_j$
    - ### Update the bias.
    - $b = b + error$
- Test for convergence

Output: Set of weights $\mathbf{w}$ and bias b for the perceptron.

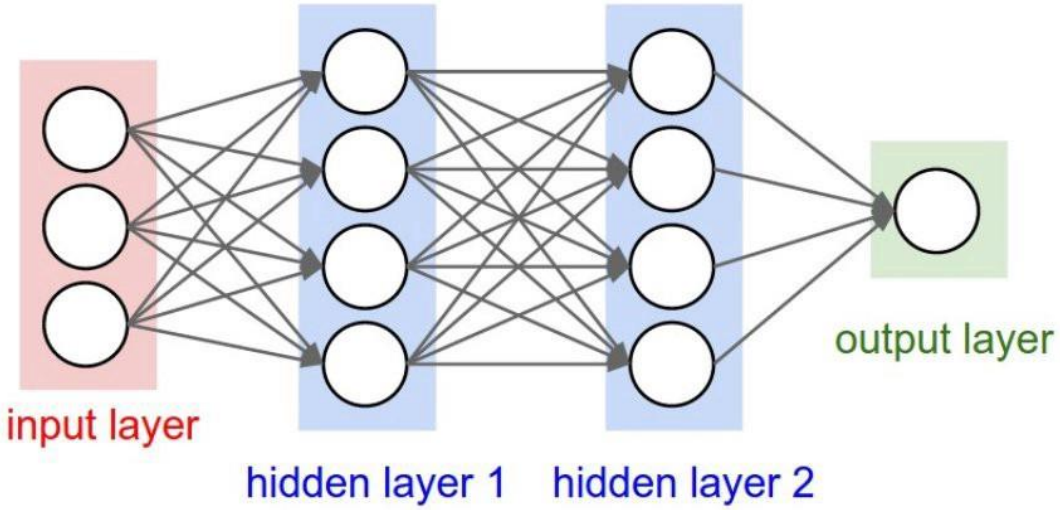
Forward School

Note: Some of these explanations are taken from: https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a

You can also find python code example of the perceptron learning here.

When one groups all the perceptrons together

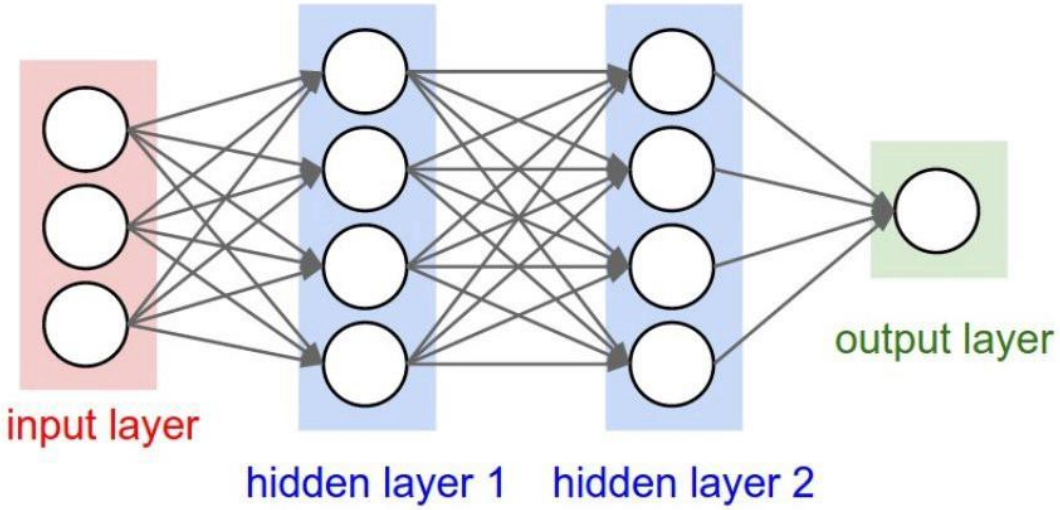
- It forms a neural network!
  - The network forms a layer made of perceptrons which are linked to the perceptrons of the previous and the next layers if such do happen to exist.
  - Every layer defines it’s own functionality and therefore serves its own purpose



Forward School

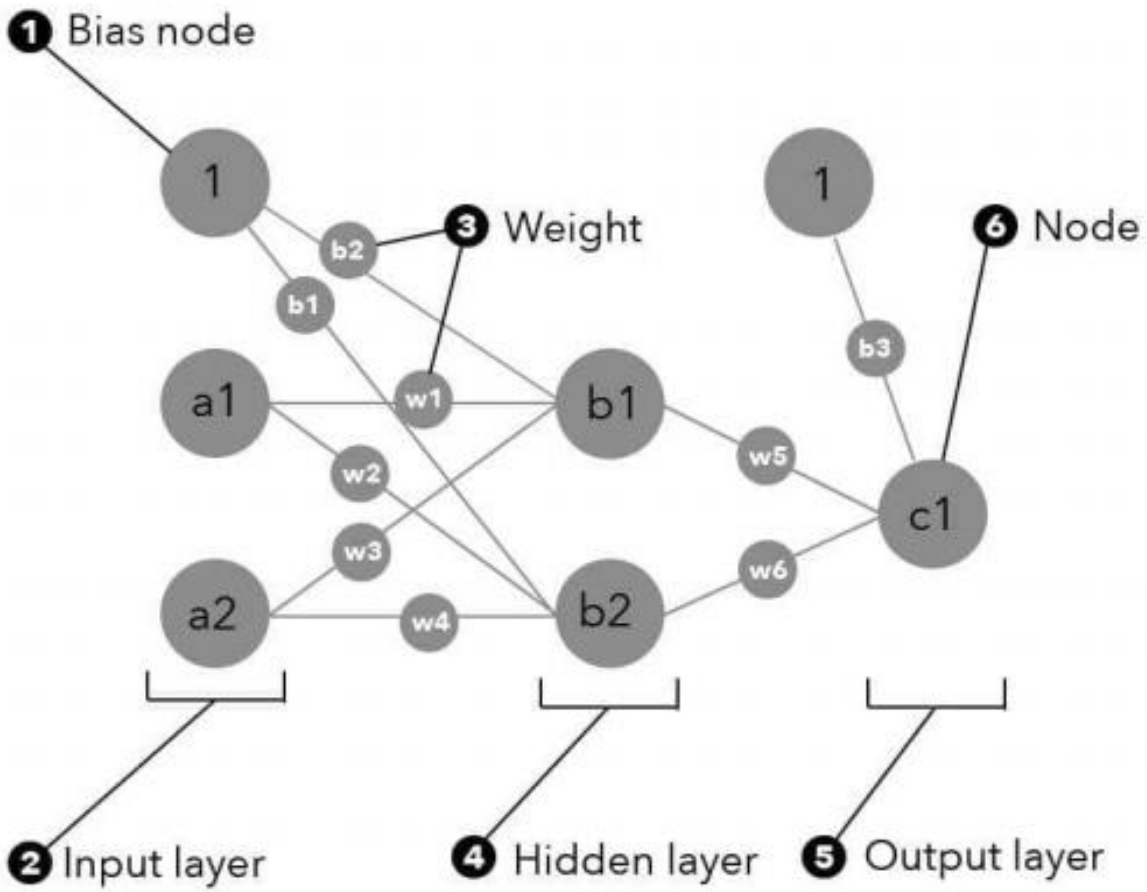
Neural networks

- Consist of an input layer(takes the initial data), an output layer(returns the overall result of the network), and hidden layers(one or many layers with different sizes(number of perceptrons) and functionality).



Forward School

Another view



Forward School

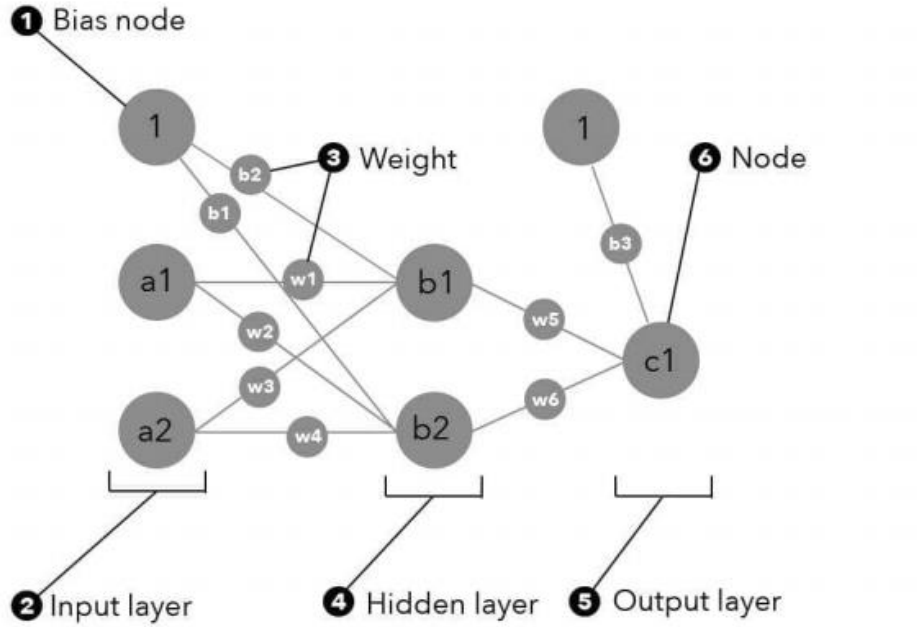
5 things you should know about Neural Networks

- #1 Neural Networks Are Specific:
- Neural networks are always built to solve a specific type of problem, although this doesn't mean they can be used as "general purpose" tools.
- For example, you will not find a "general purpose" algorithm that you can sink data into for prediction or estimation…at least not yet!
- Examples of specific uses include: prediction, forecasting, estimation, classification, and pattern recognition.

Forward School

5 things you should know about Neural Networks

- #2. Neural Networks Have Three Basic Parts: A neural network has three basic sections, or parts, and each part is composed of “nodes”
- 1. Input Layer
- 2. Hidden Layer(s)
- 3. Output Layer



Forward School

5 things you should know about Neural Networks

- #3 Neural Networks Are Built Two Ways:
- On a high level, neural networks can be built two ways:

1. Feedforward: With a feedforward neural network, signals travel only one way, from input to output. These types of networks are more straightforward and used extensively in pattern recognition. A convolutional neural network (CNN or ConvNet) is a specific type of feedforward network that is often used in image recognition.
2. Feedback (or recurrent neural networks, RNNs): With an RNN, signals can travel both directions and there can be loops. Feedback networks are more powerful and complex than CNNs, and they are always changing. Despite this, RNNs have been less influential than feedforward networks, in part because the learning algorithms for recurrent nets are (at least to date) less powerful. However, RNNs are still extremely interesting and are much closer in spirit to how our brains work than feedforward networks.

Forward School

5 things you should know about Neural Networks

- #4. Neural Networks Are Either Fixed or Adaptive:
- Neural networks can be either fixed or adaptive.
  - Fixed: The weight values in a fixed network remain static. They do not change.
  - Adaptive: The weight values in an adaptive network are not static and can change.

Forward School

5 things you should know about Neural Networks

- #5. Neural Networks Use Three Types of Datasets:
- Neural networks are often built and trained using three datasets:
  - Training dataset: The training dataset is used to adjust the weights of the neural network.
  - Validation dataset: The validation dataset is used to minimize a problem known as overfitting, which we will cover in more detail.
  - Testing dataset: The testing dataset is used as a final test to gauge how accurately the network has been trained

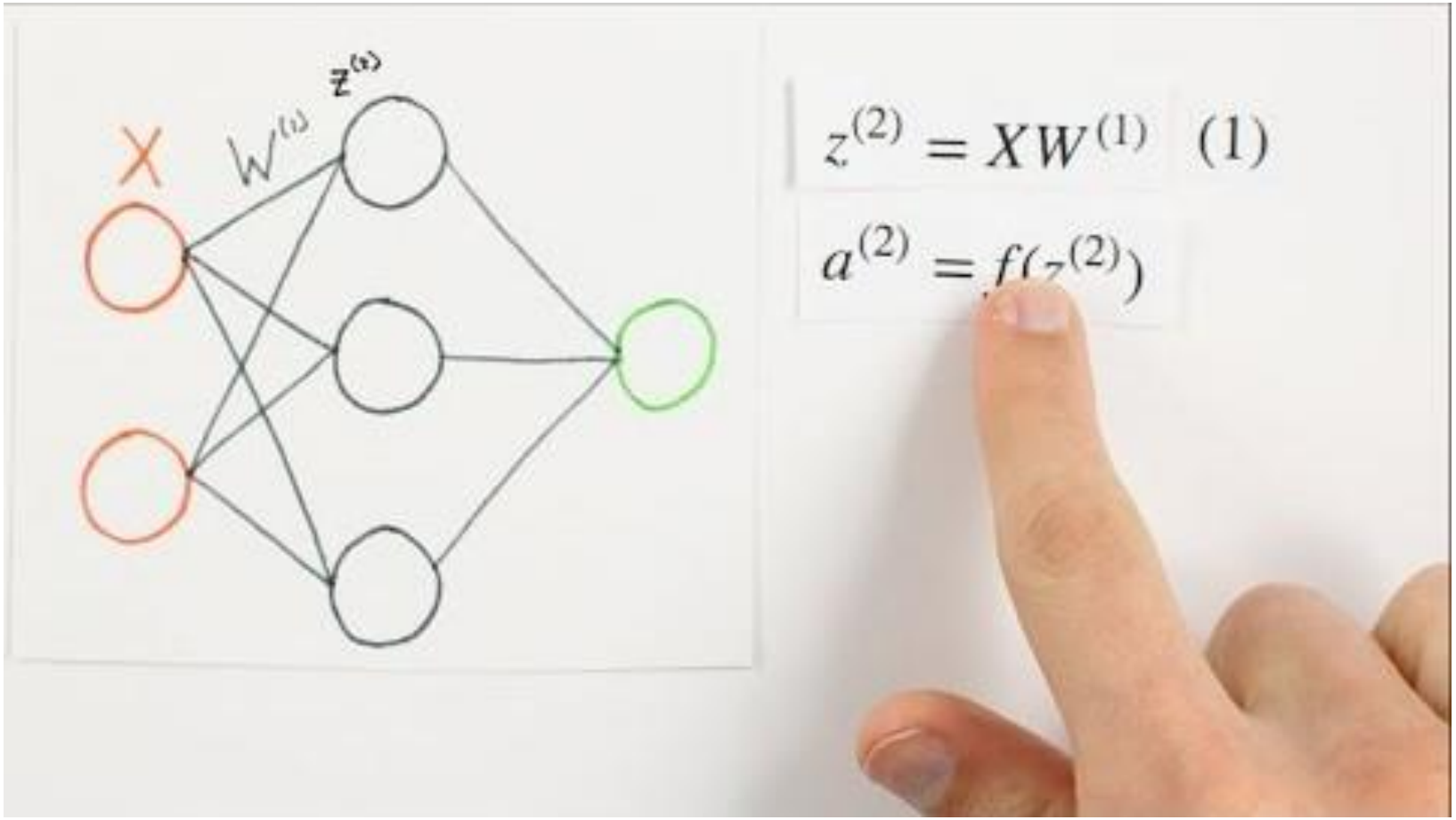
Forward School

Learning to produce results

- Each layer has to implement forward propagation and backward propagation (backpropagation)
- Example:
  - Imagine a train travelling between point A(input) and point B(output) which changes direction each time it reaches one of the points. The A to B course takes one or more samples from the input layer and carries it through the forward propagation functions of all hidden layers consecutively, until point B is reached(and a result is produced).
  - Backpropagation is basically the same thing only in the opposite direction — the course takes the data through the backpropagation methods of all layers in a reverse order until it reaches point A. What differs the two courses though is what happens inside of these methods.

Forward School

Time for some simple video explanations



Forward School

The whole series

- Data and Architecture: https://www.youtube.com/watch?v=bxe2T-V8XRs
- Forward Propagation: https://www.youtube.com/watch?v=UJwK6jAStmg
- Gradiant Descent: https://www.youtube.com/watch?v=5u0jaA3qAGk
- Backpropagation: https://www.youtube.com/watch?v=GlcnxUlrtek
- Numerical Gradient Checking: https://www.youtube.com/watch?v=pHMzNW8Agq4

Forward School