# Machine Learning (ML) with Python

## Artificial Neural Network (Deep Learning)

Dr. Aeshah Alsughayyir

Collage of Computer Science and Engineering

Taibah University

2021-2022

# Outline

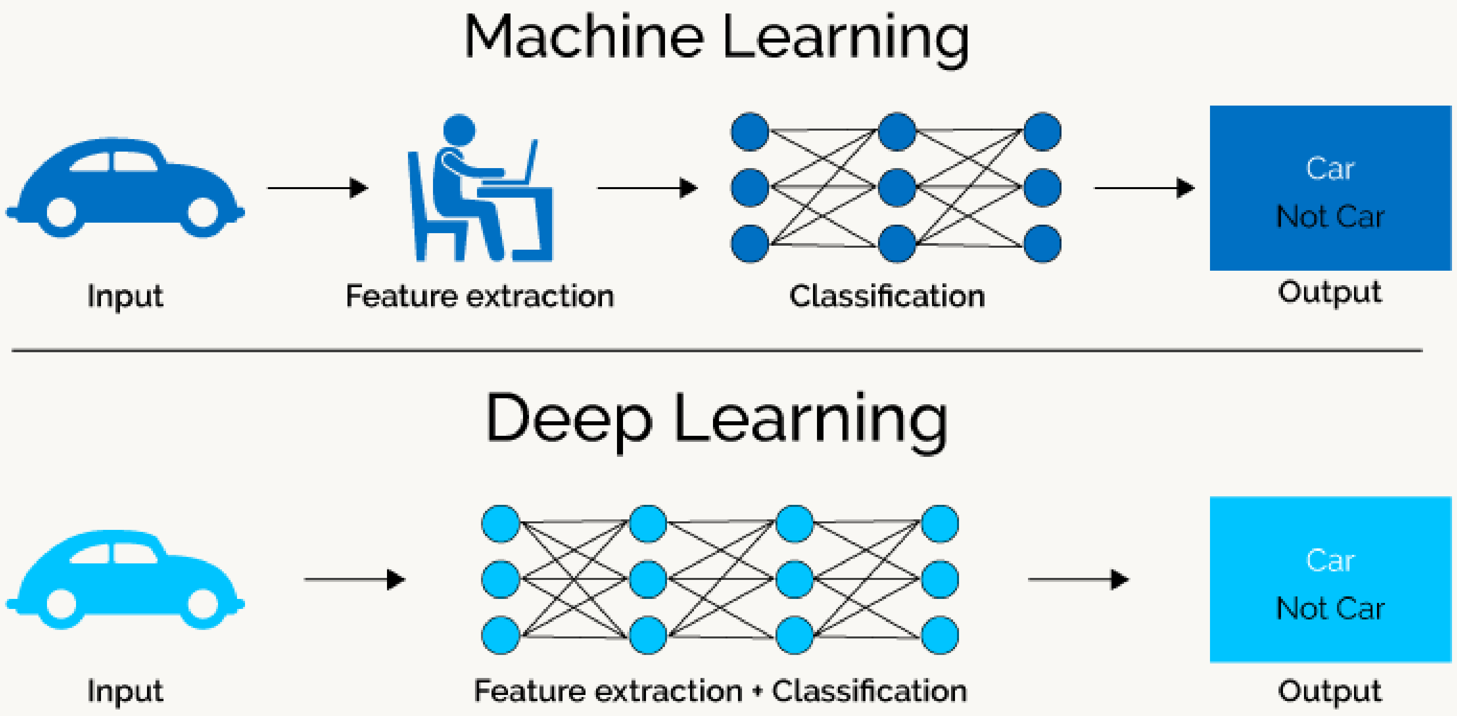
- What is Deep Learning (DL) ?
- How do our brains work?
- Artificial Neural Network: *what is it*?
- How do ANNs work?
- Model of an artificial neuron
- NN Hidden Layers and Learning
- Learning by *trial* and *error*
- Main Issues in Designing NN
  - Activation Functions
    - *Sigmoid*
    - *ReLU*
  - Error Estimation
  - Weights Adjusting
  - Back Propagation
  - Number of Neurons
  - Data Representation
  - Size of Training-set
- Learning Paradigms or Approaches (*recall*)
- Advantages / Disadvantages
- Example: Voice Recognition

# What is Deep Learning (DL) ?

A machine learning subfield of learning **representations** of data. Exceptional effective at **learning patterns**.

Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**

If you provide the system **tons of information**, it begins to understand it and respond in useful ways.




*https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png*

# How do our brains work?

- The Brain is a massively parallel information processing system.
- Our brains are a huge network of processing elements. A typical brain contains a network of 10 billion neurons.

# Artificial Neural Network: *what is it?*

An artificial neural network consists of a pool of simple processing units which communicate by sending signals to each other over a large number of **weighted** connections.

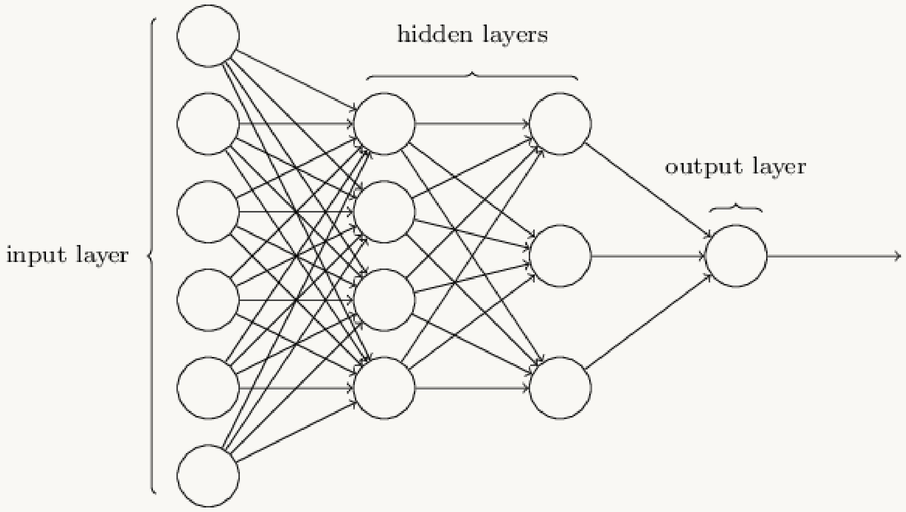
# Artificial Neural Network: *what is it?*

- Models of the brain and nervous system
- Highly parallel
  - Process information much more like the brain than a serial computer
- Learning
- Very simple principles
- Very complex behaviours
- Applications
  - As powerful problem solvers
  - As biological models

# Artificial Neural Network: *what is it?*

The "**building blocks**" of neural networks are the **neurons**.

- In technical systems, we also refer to them as **units** or **nodes**.

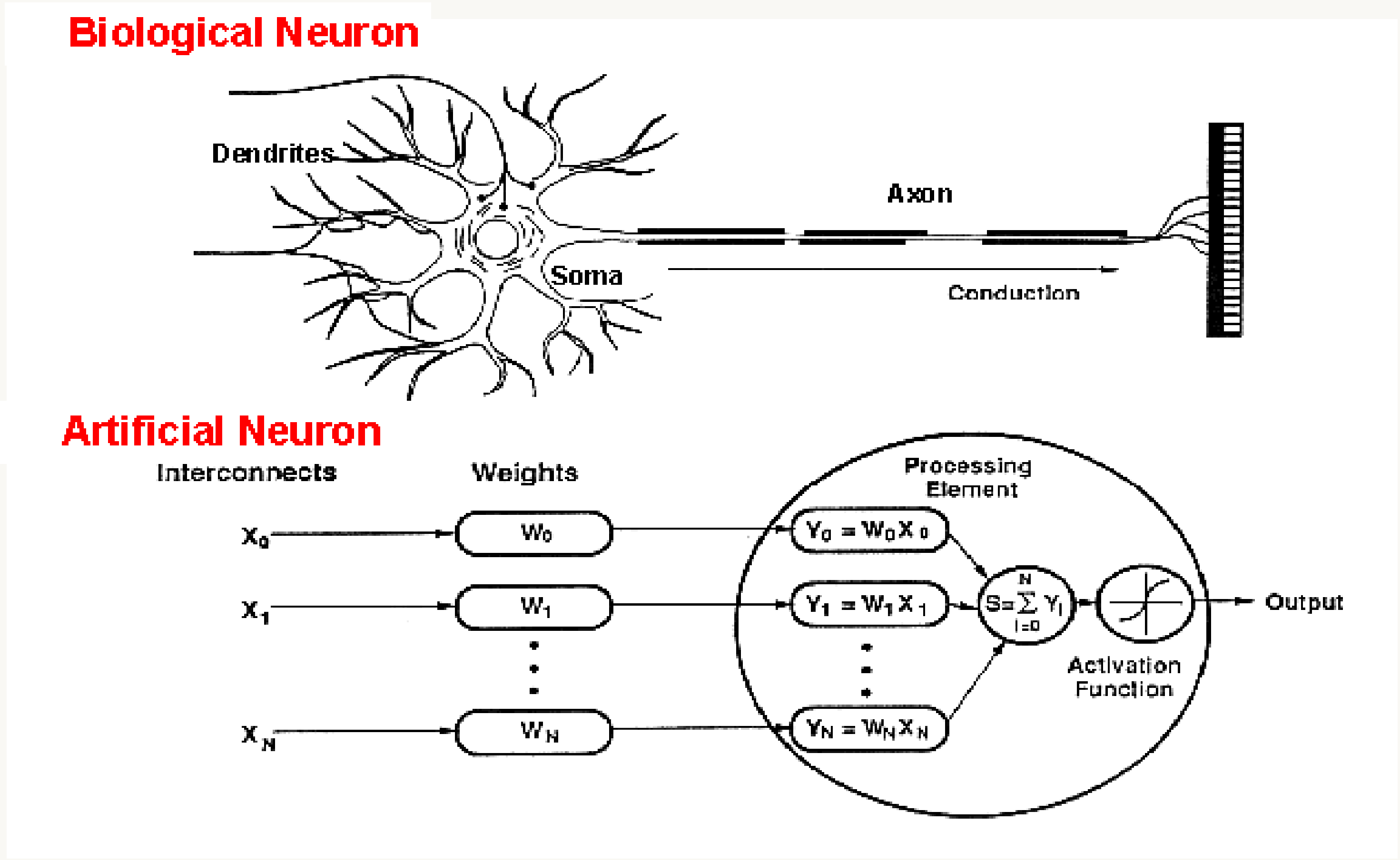



## Basically, each neuron

- receives **input** from many other neurons.
- changes its internal state (**activation**) based on the current input.
- sends **one output signal** to many other neurons, possibly including its input neurons (**recurrent network**).

# Artificial Neural Network: *what is it?*

An artificial neuron is an imitation of a human neuron



# Artificial Neural Network: *what is it?*

A neuron looks like this...

$f(x) = m\,x + b$

could also be represented like

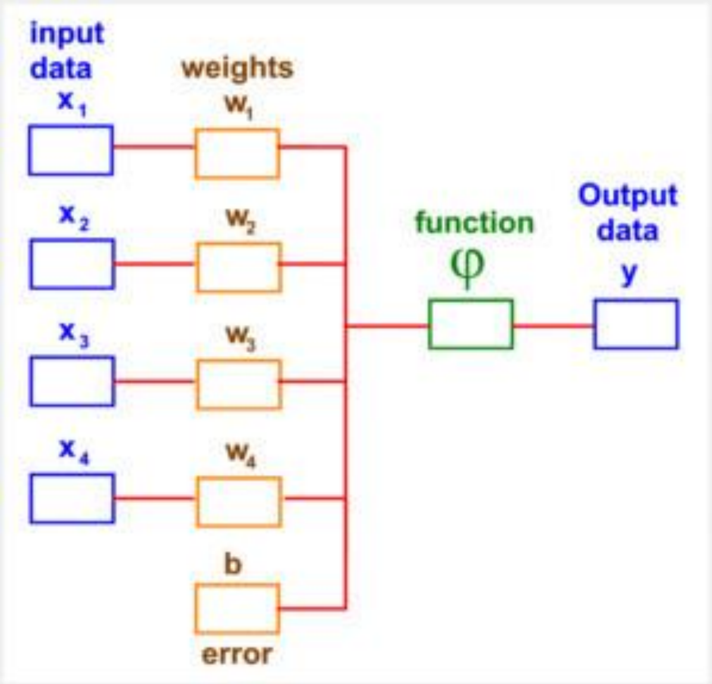
$y = f(x)$

$f(x) = w1 * x1 + b$

where w is the weight, and **b** is the bias

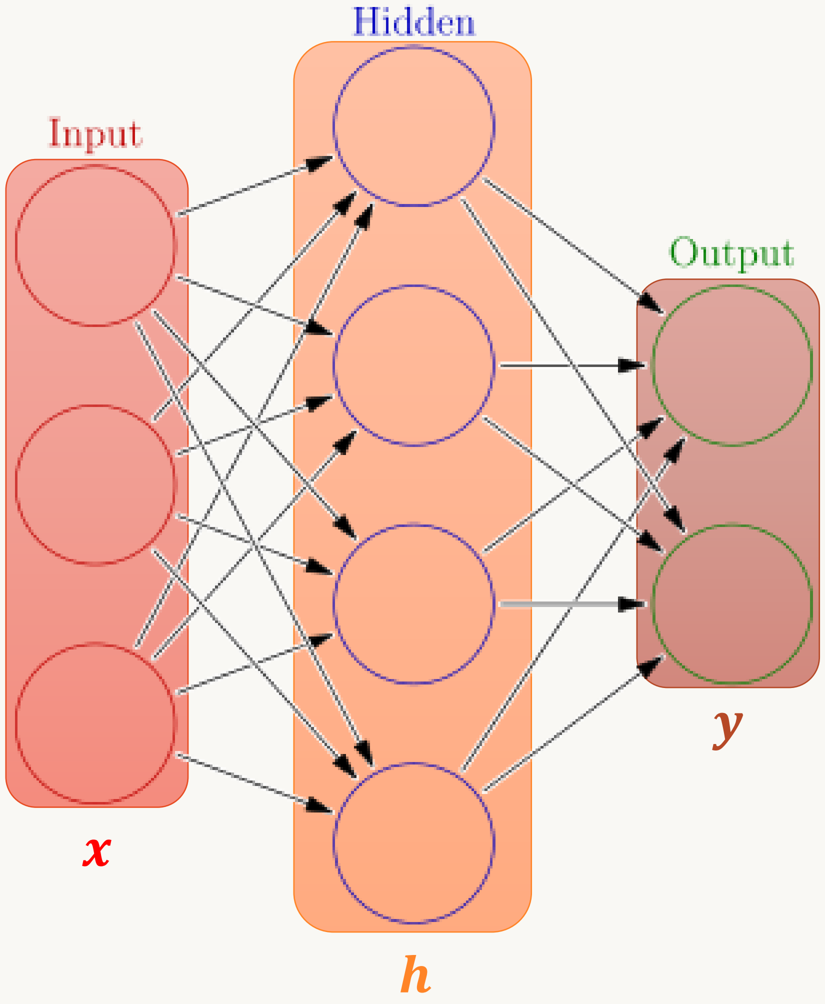
A general form to represent a neuron is:

$y = f(x1 \cdot w1 + x2 \cdot w2 + \ldots + b)$




The trick of machine learning is to find values of **w** and **b** coefficients (degree) that bring the best final results for the entire neuron network.

# How do ANNs work?




Weights

$$h = \sigma(W_1 x + b_1)$$

$$y = \sigma(W_2 h + b_2)$$

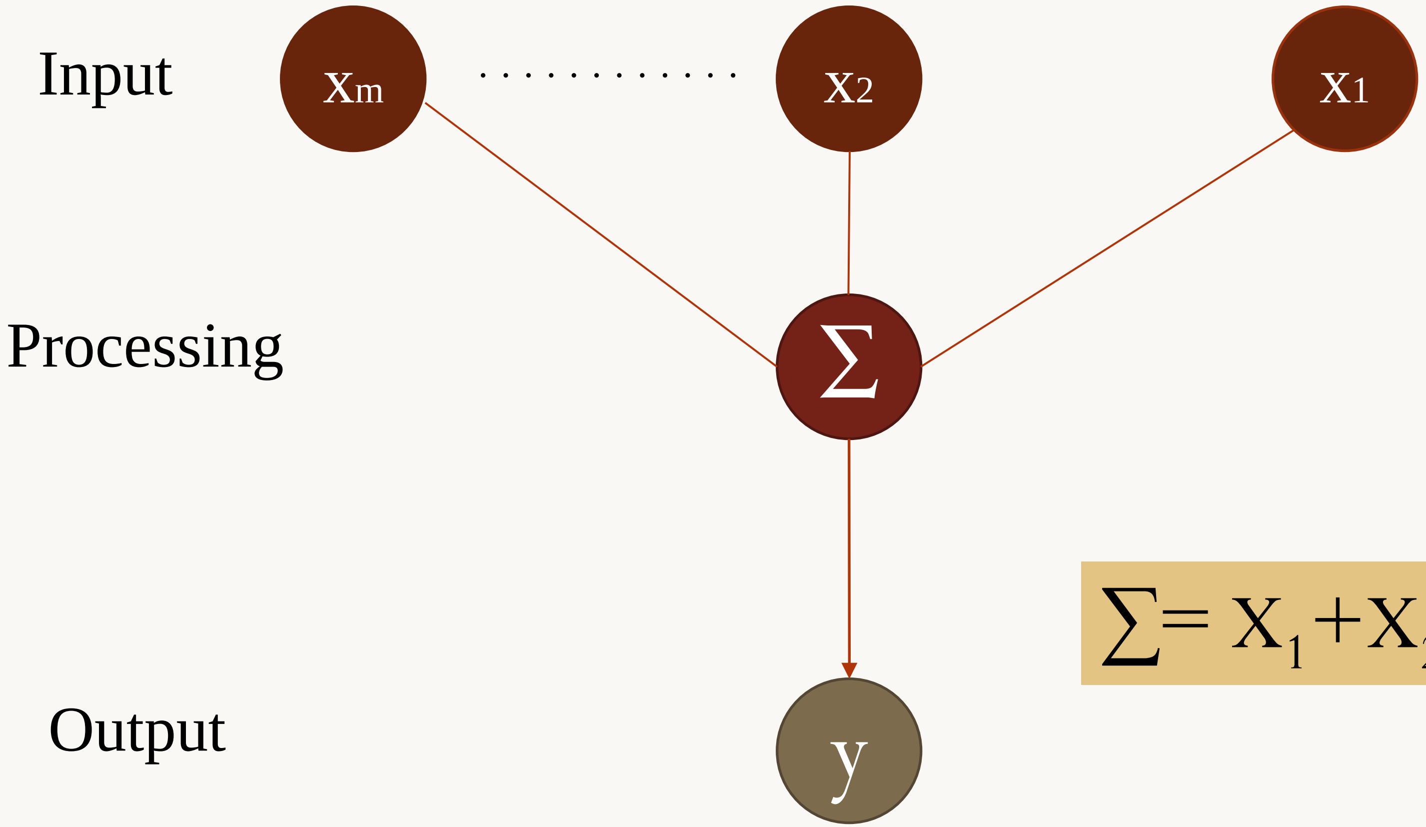
Activation functions

How do we train?

4 + 2 = 6 neurons (not counting inputs)
[3 x 4] + [4 x 2] = 20 weights
4 + 2 = 6 biases

26 learnable **parameters**

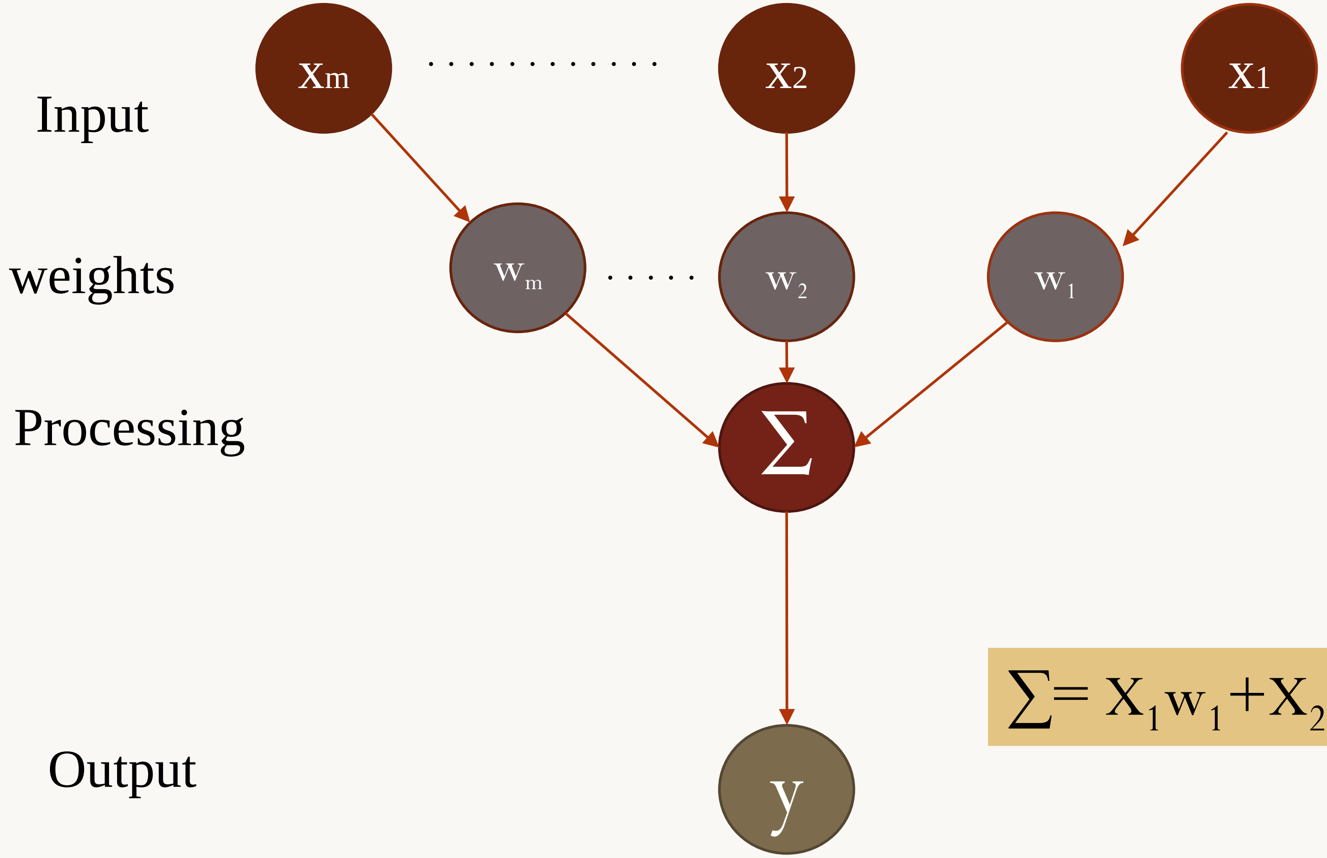
# Model of an Artificial Neuron




$$\sum = X_1 + X_2 + \ldots + X_m = y$$

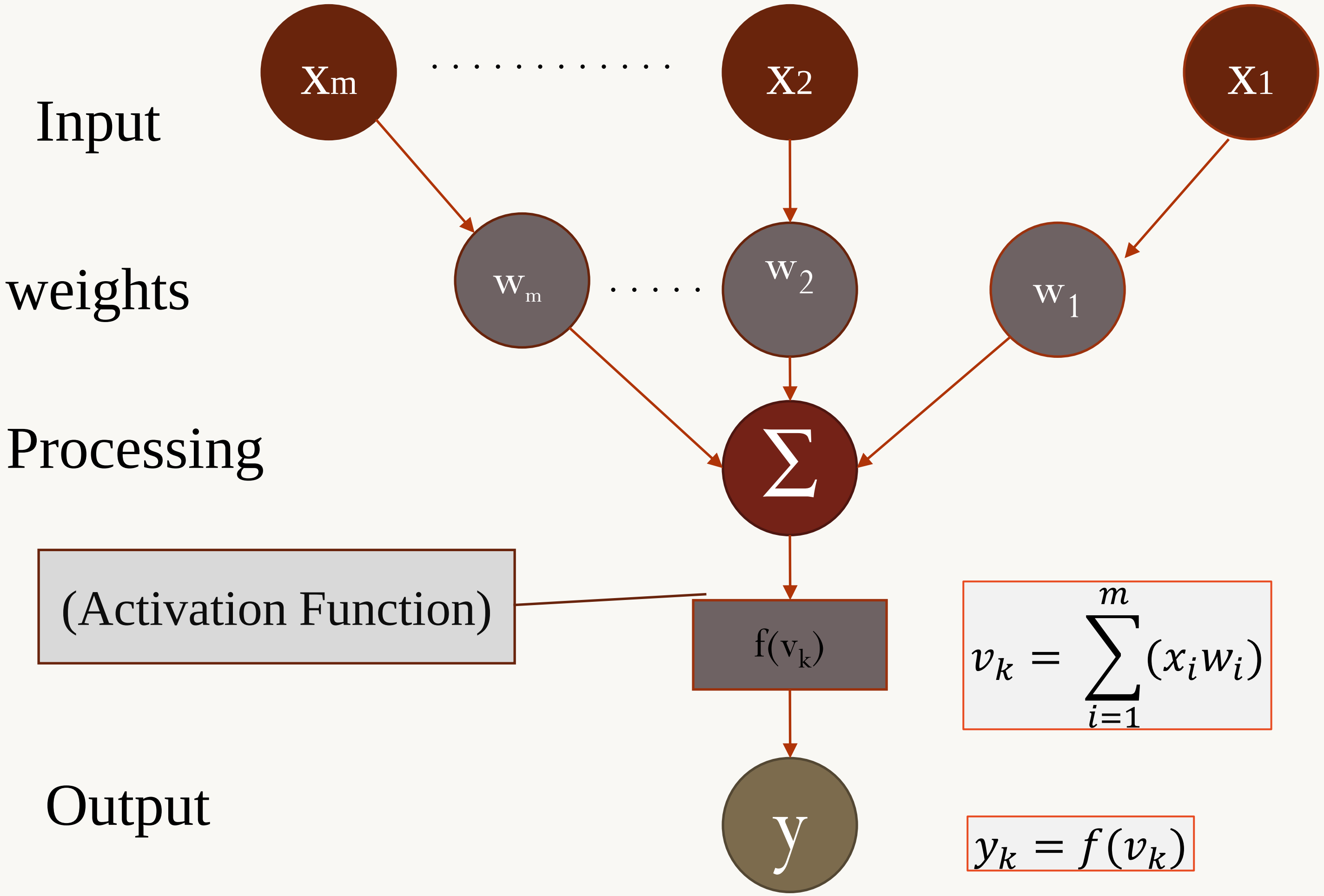
# Model of an Artificial Neuron

Not all inputs are equal




$$\sum = X_1 w_1 + X_2 w_2 + \ldots + X_m w_m = y$$

# Model of an Artificial Neuron

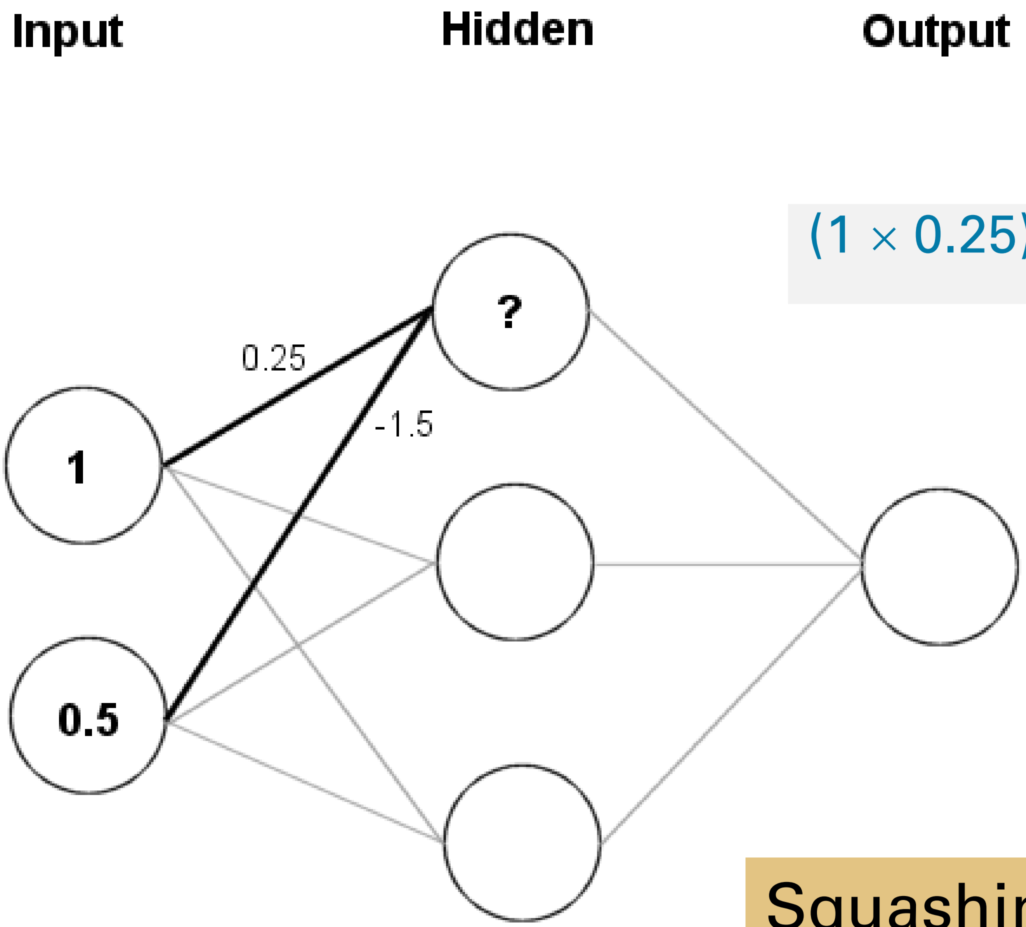



$$v_k = \sum_{i=1}^{m} (x_i w_i)$$

$$y_k = f(v_k)$$

- The signal is not passed down to the next neuron directly.
- The output is a **function** of the input, that is affected by the weights, and the **activation functions**

# Feed-forward nets

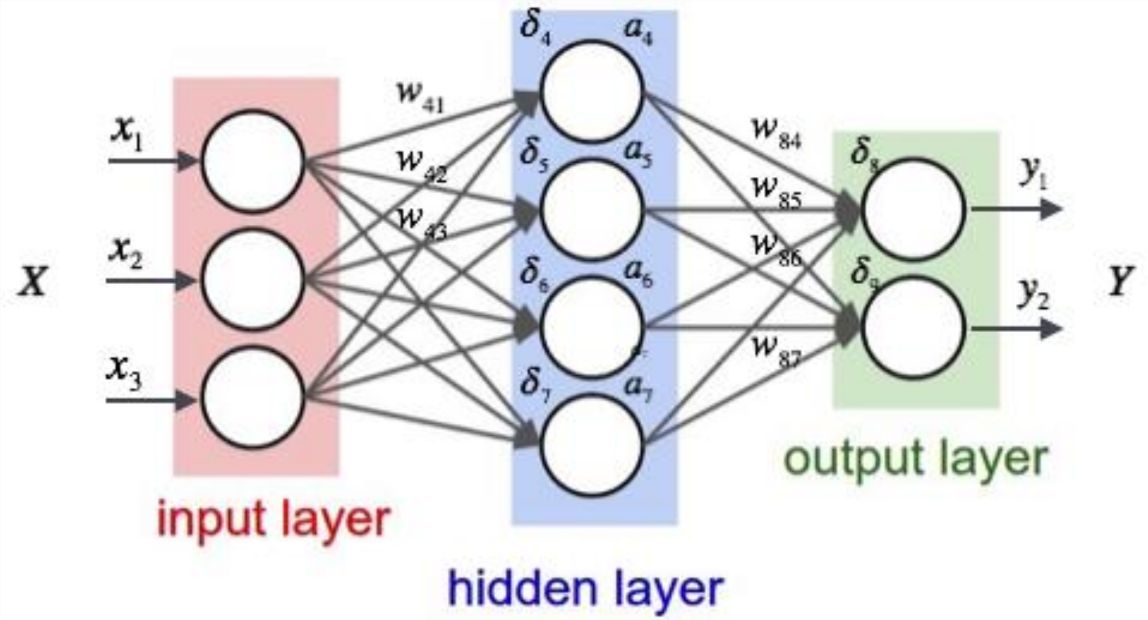



$(1 \times 0.25) + (0.5 \times (-1.5)) = 0.25 + (-0.75) \quad = \ \mathbf{-\ 0.5}$

Squashing:

$$\frac{1}{1+e^{0.5}} = 0.3775$$

# NN hidden Layers and Learning



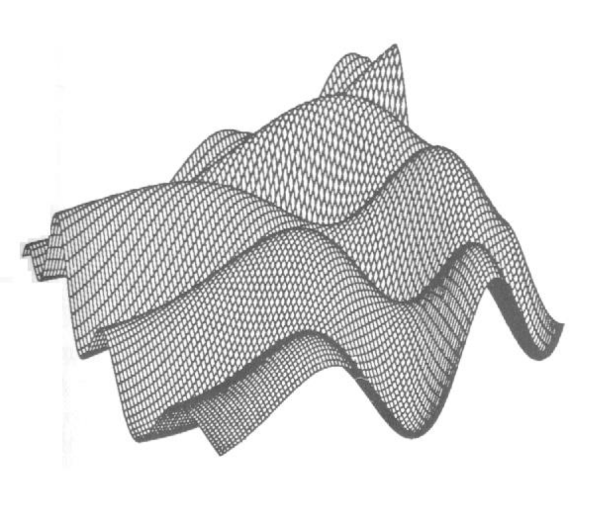

**An ANN can:**

- compute *any computable* function, by the appropriate selection of the network topology and weights values.
- learn from experience!
- Specifically, by trial-and-error

# Weight settings determine the behavior of a network



How can we find the right weights?

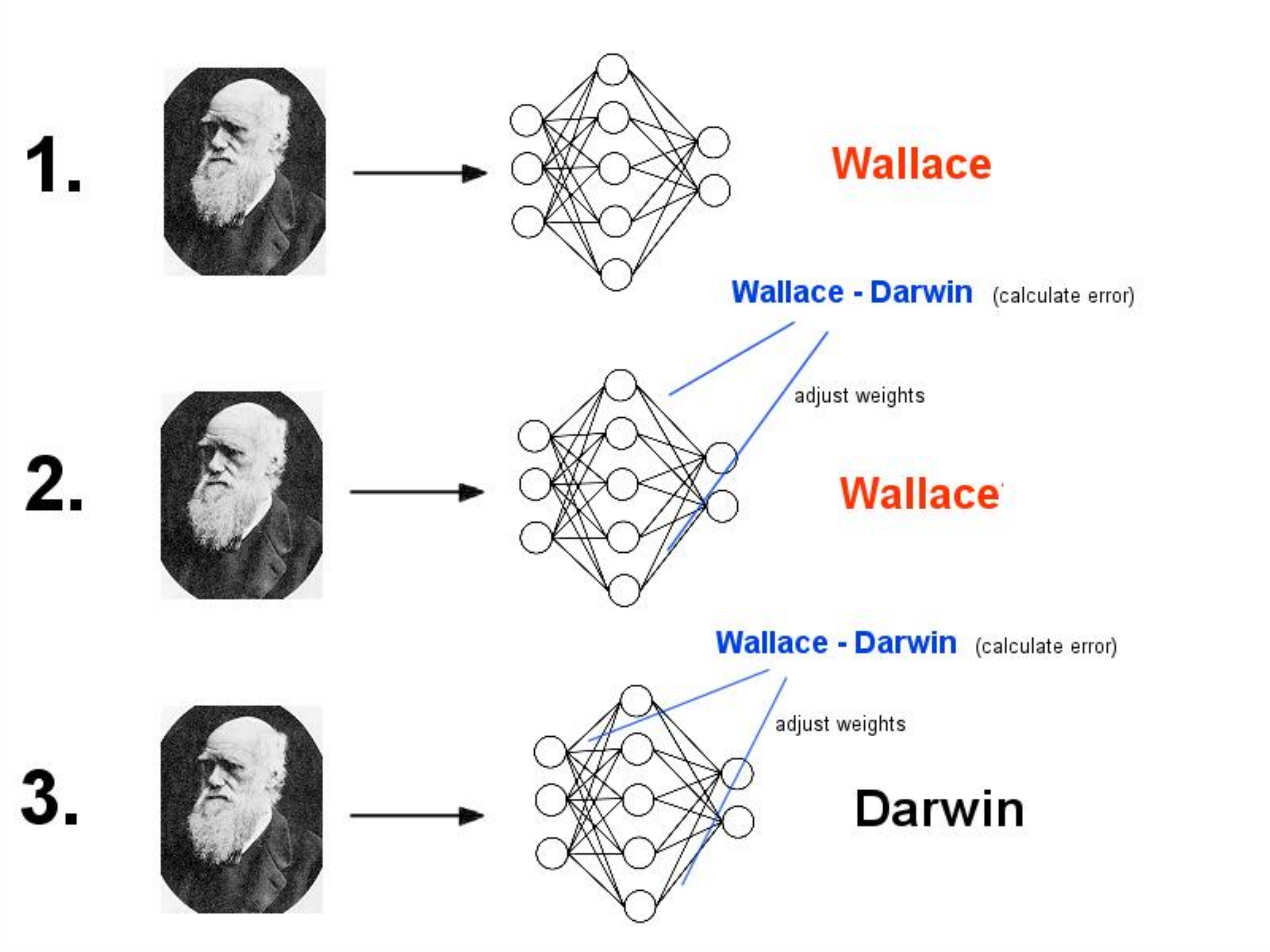


**Training the Network - Learning**

- Backpropagation
  - Requires training set (input / output pairs)
  - Starts with small random weights
  - Error is used to adjust weights (supervised learning)

  → **Gradient descent on error landscape**

# For Example:


1.
Wallace
Wallace - Darwin (calculate error)
adjust weights
2.
Wallace
Wallace - Darwin (calculate error)
adjust weights
3.
Darwin

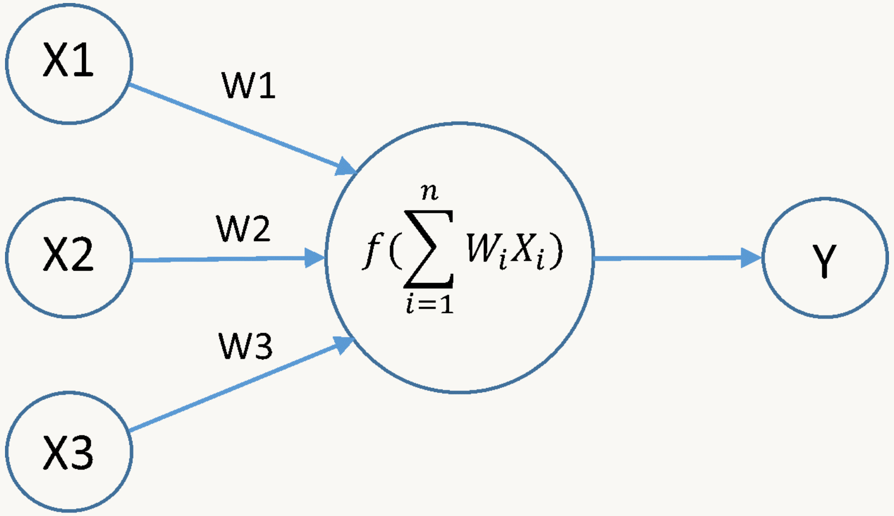
# Learning by trial-and-error

Continuous process of

- Trial:
  - Processing an input to produce an output (In terms of ANN: Compute the output function of a given input)
- Evaluate:
  - Evaluating this output by comparing the actual output with the expected output.
- Adjust:
  - Adjust the weights.

# Main issues in designing NN

- Initial weights
- Activation (Transfer) function (How the inputs and the weights are combined to produce output?)
- Error estimation
- Weights adjusting
- Number of neurons
- Data representation
- Size of training set



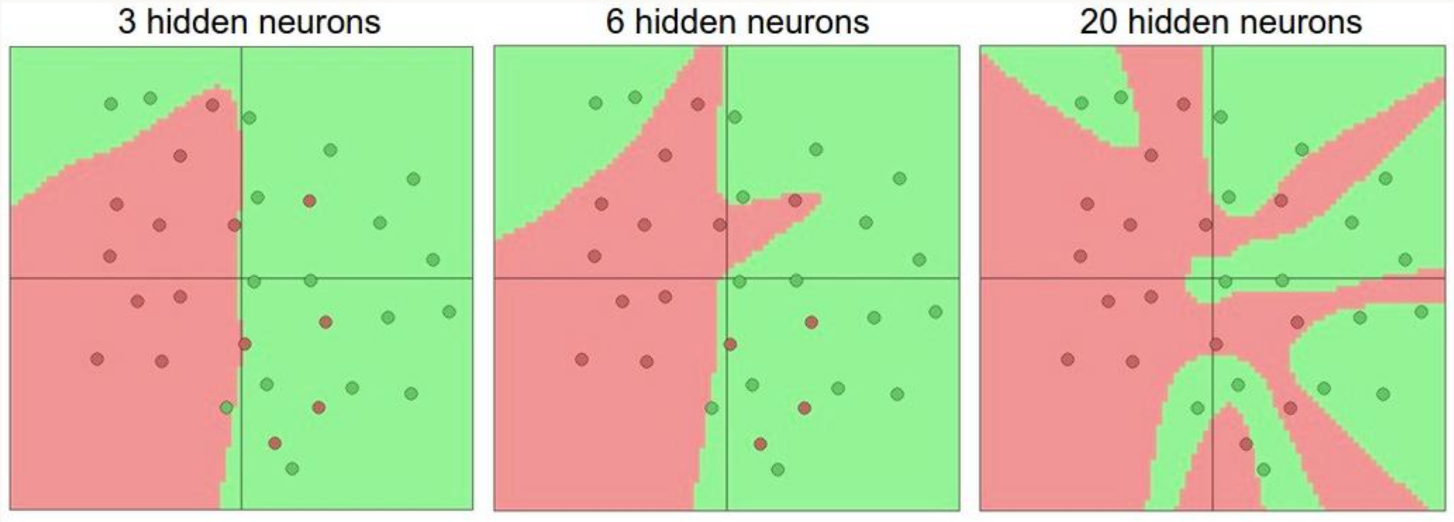
# Activation Functions

- **Linear:** The output is proportional to the total weighted input.
- **Threshold:** The output is set at one of two values, depending on whether the total weighted input is greater than or less than some threshold value.
- **Non-linear:** The output varies continuously but not linearly as the input changes.

# Activation Functions

Non-linearities needed to learn complex (non-linear) representations of data, otherwise the NN would be just a linear function

$y = f(x1 \cdot w1 + x2 \cdot w2 + \ldots + b)$



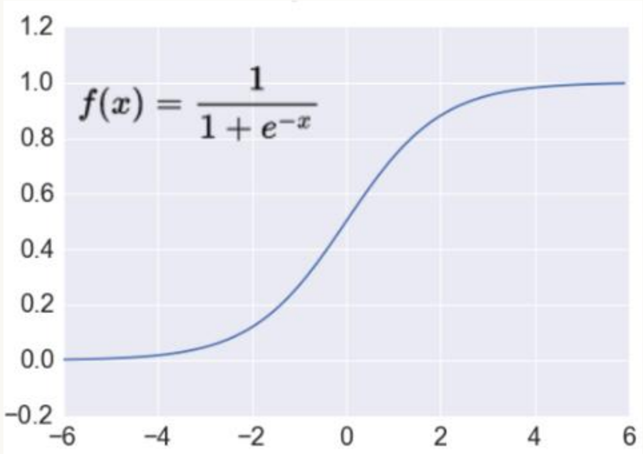

*http://cs231n.github.io/assets/nn1/layer_sizes.jpeg*

More layers and neurons can approximate more complex functions

# Activation: Sigmoid

+ Nice interpretation as the **firing rate** of a neuron
  - 0 = not firing at all
  - 1 = fully firing

Takes a real-valued number and "squashes" it into range between 0 and 1.




*http://adilmoujahid.com/images/activation.png*

- Sigmoid neurons **stick or kill gradients**, thus NN will hardly learn
  - when the neuron's activation are 0 or 1 (stick)
    - gradient at these regions almost zero
    - almost no signal will flow to its weights
    - if initial weights are too large then most neurons would stick

# Activation: ReLU

Most Deep Networks use ReLU nowadays

☺ Trains much **faster**

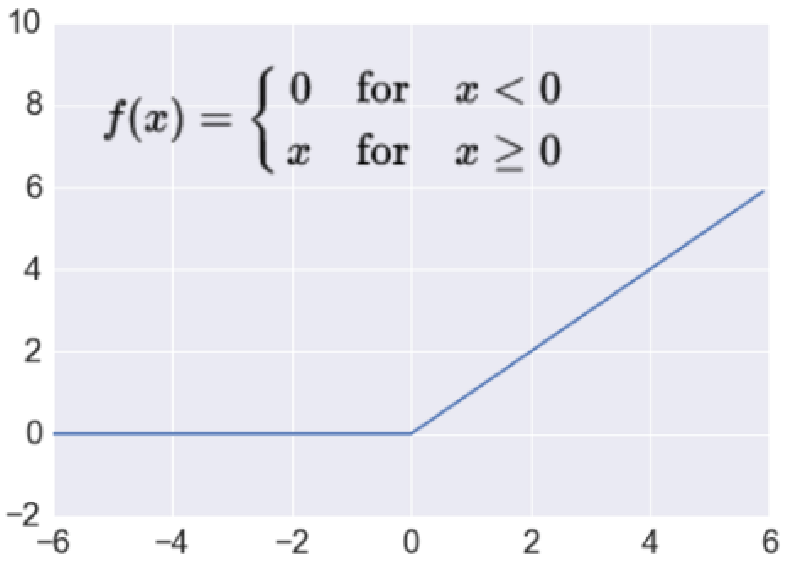
☺ Less expensive operations

- compared to sigmoid/tanh (exponentials etc.)
- implemented by simply thresholding a matrix at zero

☺ More **expressive**

☺ Prevents the **gradient vanishing problem**

Takes a real-valued number and thresholds it at zero $f(x) = \max(0, x)$




*http://adilmoujahid.com/images/activation.png*

# Error Estimation

## The root mean square error (RMSE)

is a frequently-used measure of the differences between values predicted by a model or an estimator and the values actually observed from the thing being modelled or estimated.

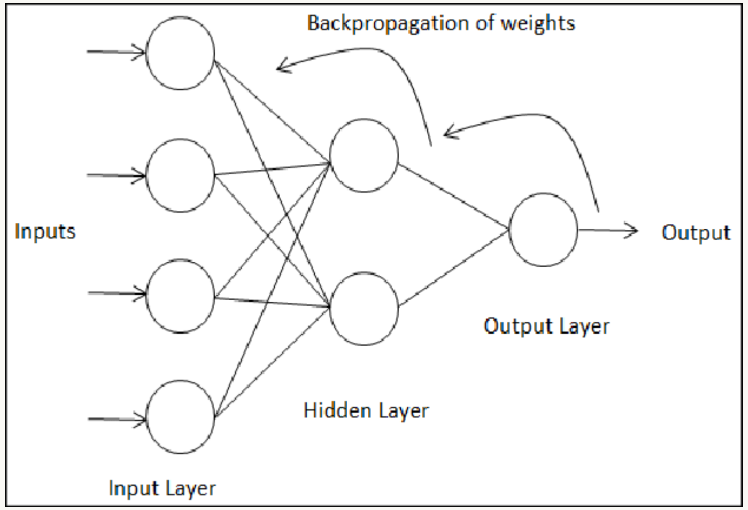
## Weights Adjusting

After each iteration, weights should be adjusted to minimize the error.

- All possible weights
- Back propagation

# Back Propagation

- Back-propagation is an example of supervised learning is used at each layer to minimize the error between the layer's response and the actual data
- The error at each hidden layer is an average of the evaluated error
- Hidden layer networks are trained this way.
- The poplar algorithm used here is *gradient descent.*



# Back Propagation

- N is a neuron.
- $N_w$ is one of N's inputs weights
- $N_{out}$ is N's output.
- $N_w = N_w - \alpha \nabla N_w$
- $\nabla N_w = N_{out} * (1 - N_{out}) * N_{ErrorFactor}$
- $N_{ErrorFactor} = N_{ExpectedOutput} - N_{ActualOutput}$

This works only for the last layer, as we can know the actual output, and the expected output.

# Number of neurons

- Many neurons:
  - Higher accuracy
  - Slower
  - Risk of over-fitting
    - Memorizing, rather than understanding
    - The network will be useless with new problems.
- Few neurons:
  - Lower accuracy
  - Inability to learn at all
- Optimal number!

# Data Representation

- Usually input/output data needs pre-processing
- Pictures
  - Pixel intensity
- Text:
  - A pattern

    0-0-1 for “Asma”
    0-1-0 for “Abrar”
  - Encoding mechanism

# Size of Training-set

- Overfitting can occur if a "good" training set is not chosen


- What constitutes a "good" training set?
  - Samples must represent the general population.
  - Samples must contain members of each class.
  - Samples in each class must contain a wide range of variations or noise effect.

- The size of the training set is <u>related</u> to the number of hidden neurons

# Learning Paradigms (recall)

- Supervised learning (our focus on this lecture)
- Unsupervised learning
- Reinforcement learning

# Advantages / Disadvantages

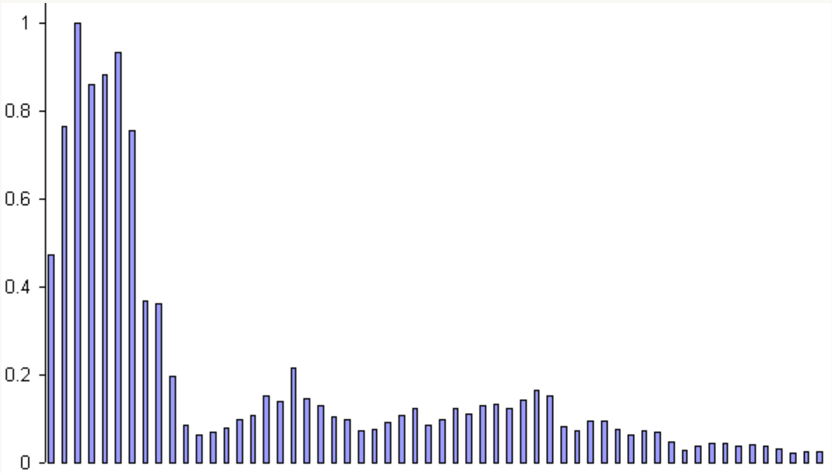
- Advantages
  - Adapt to unknown situations
  - Powerful, it can model complex functions.
  - Ease of use, learns by example, and very little user domain-specific expertise needed

- Disadvantages
  - Not exact
  - Large complexity of the network structure

# Example: Voice Recognition

- Task: Learn to differentiate between two different voices saying "Hello"

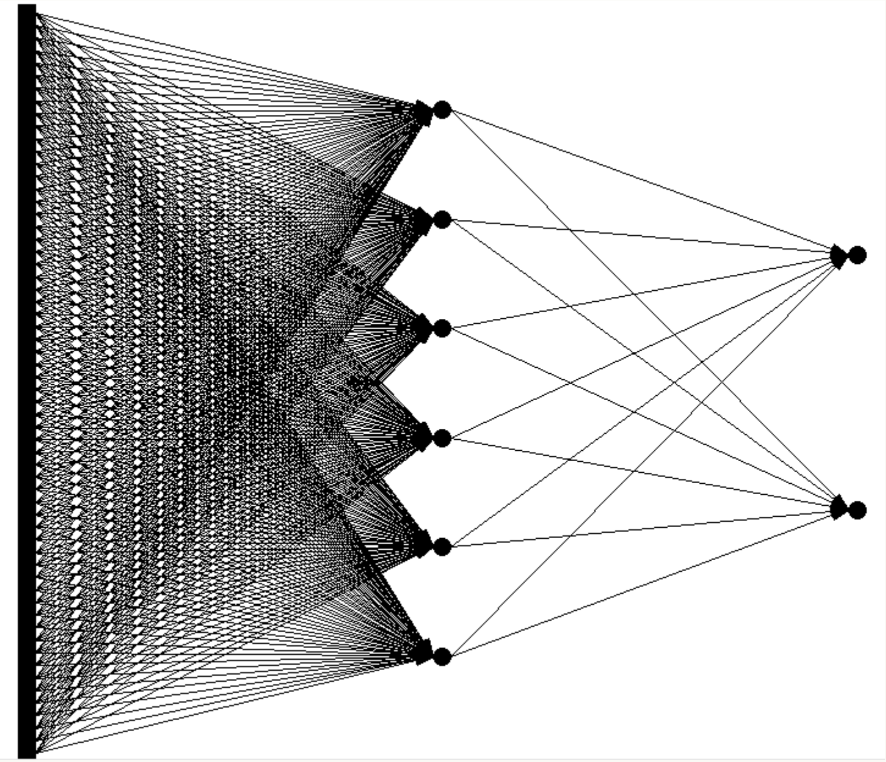
- Data
  - Sources
    - Steve
    - David
  - Format
    - Frequency distribution (60 bins)



# Network architecture

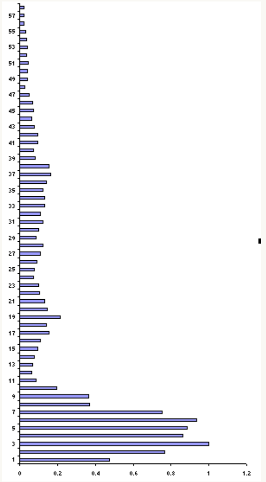
Feed forward network

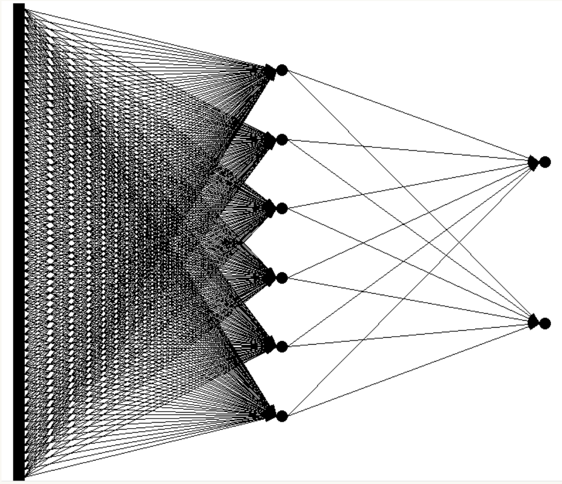
- 60 input (one for each frequency bin)
- 1 hidden with 6 neurons
- 2 output (0-1 for “Steve”, 1-0 for “David”)



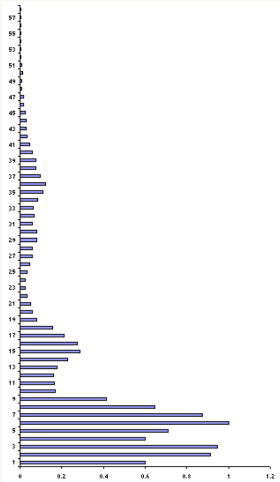
# Presenting the data

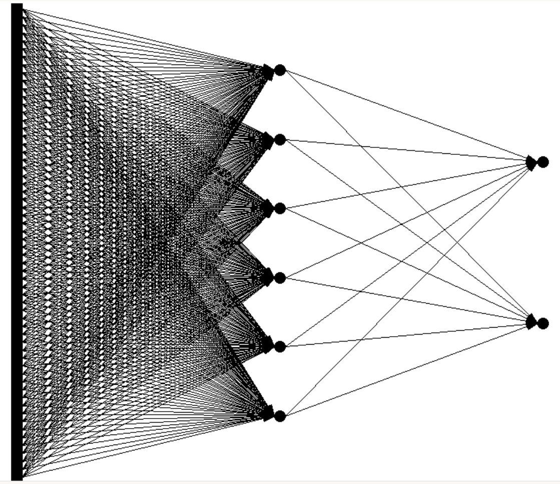
**Steve**





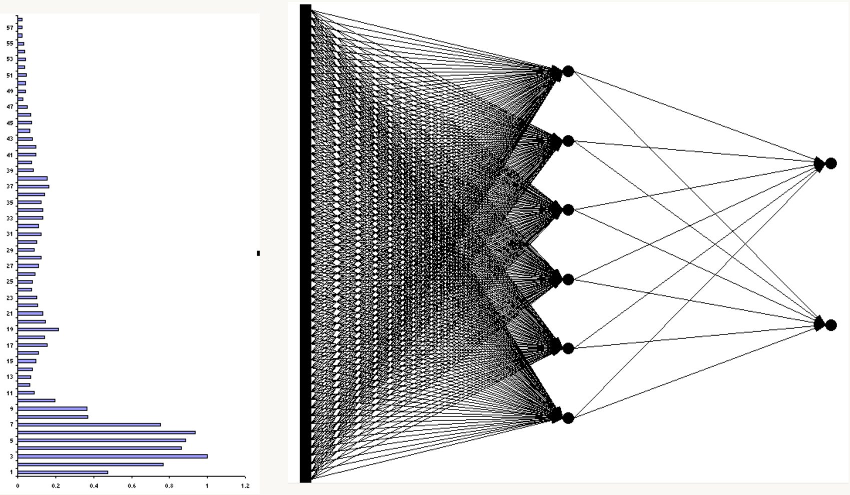
**David**





# Presenting the data (untrained network)

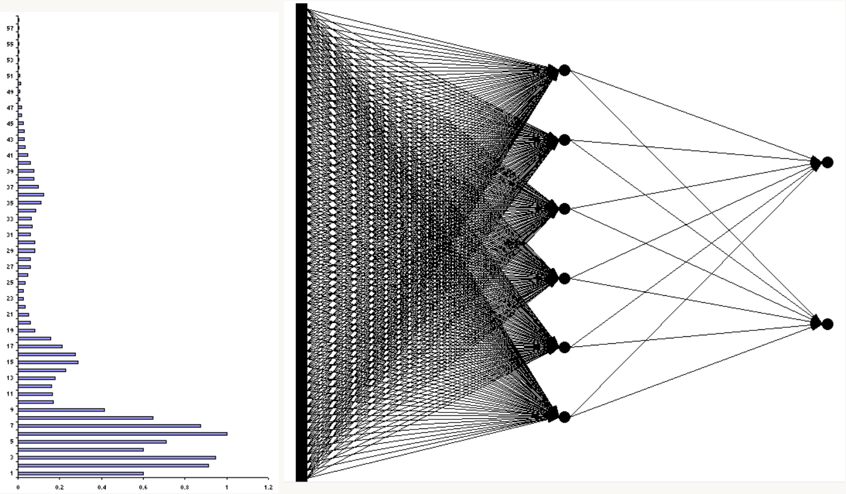
**Steve**



0.43

0.26

**David**

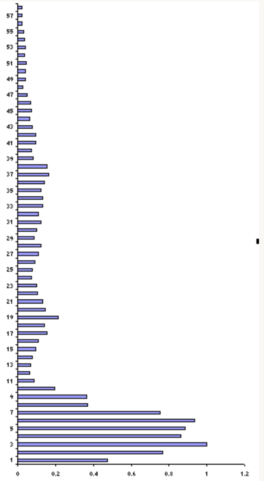


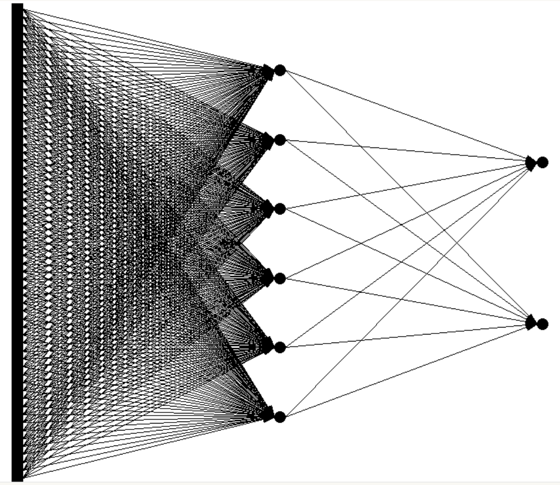
0.73

0.55

# Calculate error

**Steve**

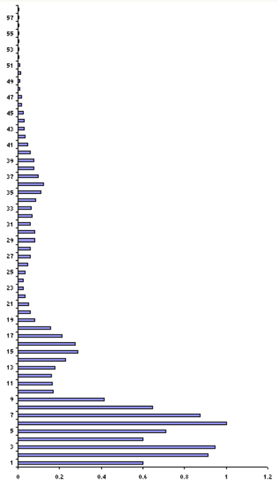


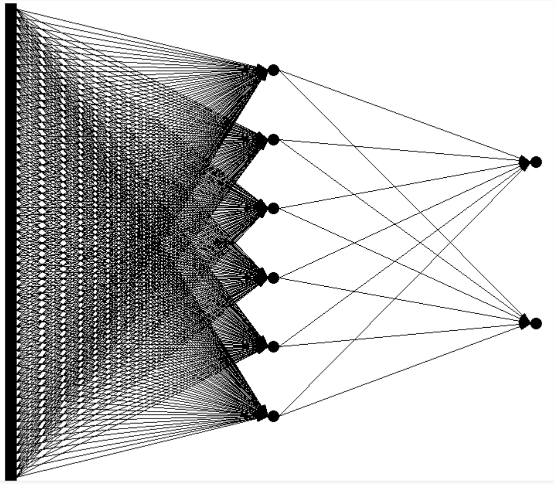


$0.43 - 0 = 0.43$

$0.26 - 1 = 0.74$

**David**



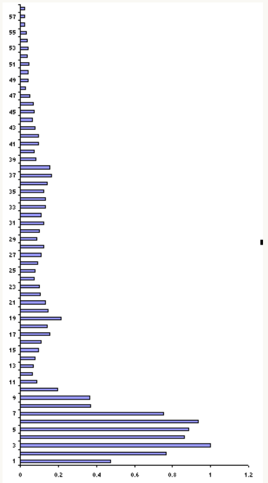


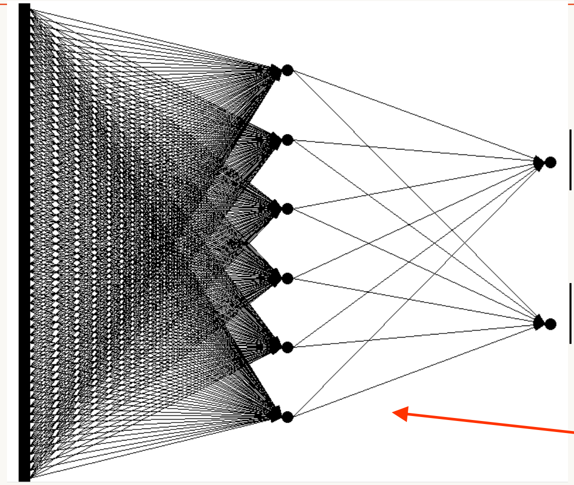
$0.73 - 1 = 0.27$

$0.55 - 0 = 0.55$

# Backprop error and adjust weights

**Steve**



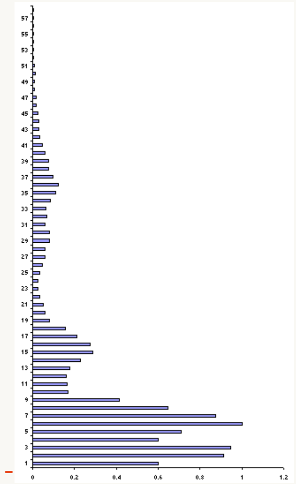


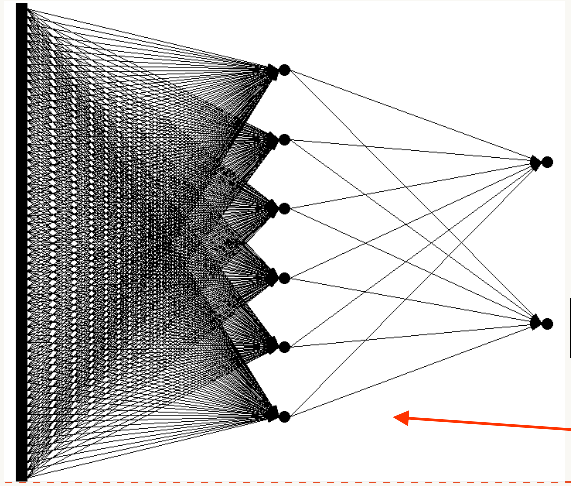
$|0.43 - 0|$ $= 0.43$

$|0.26 - 1|$ $= 0.74$

1.17

**David**





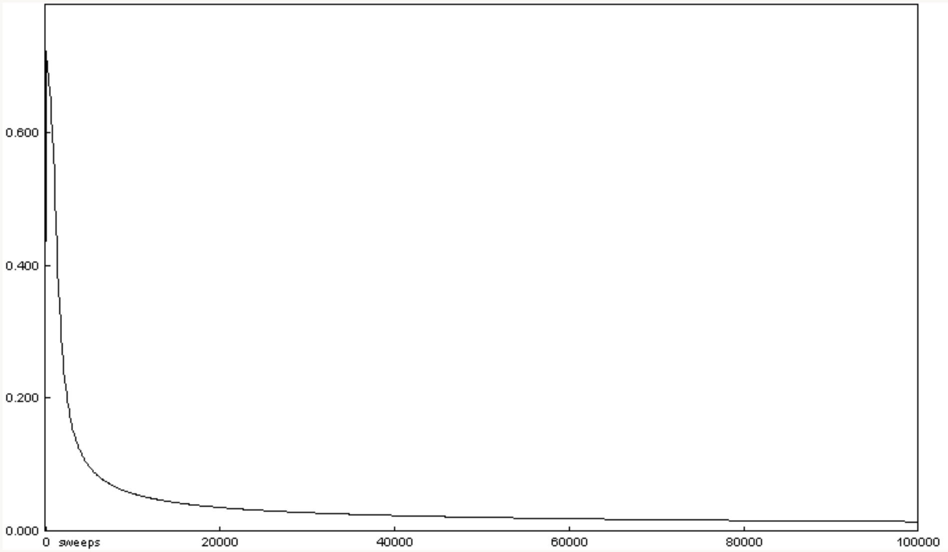
$|0.73 - 1|$ $= 0.27$

$|0.55 - 0|$ $= 0.55$

0.82

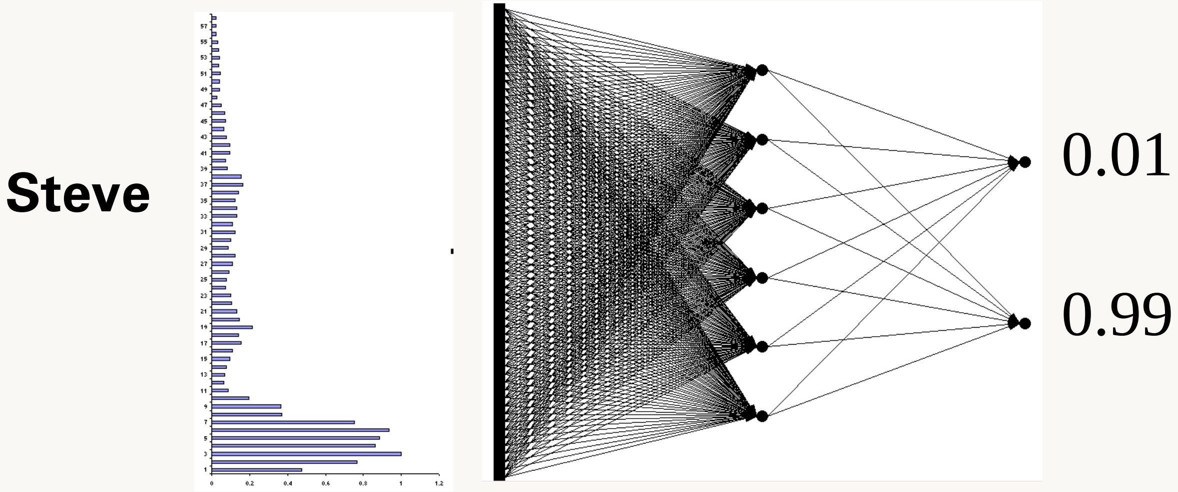
# Backprop error and adjust weights

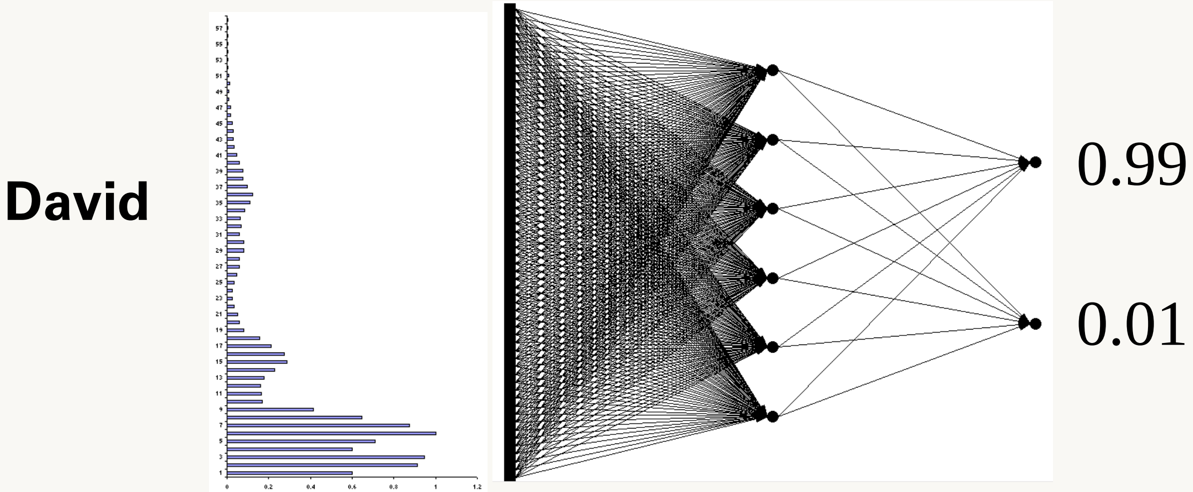
- Repeat process (sweep) for all training pairs
  - Present data
  - Calculate error
  - Backpropagate error
  - Adjust weights




- Repeat process multiple times

# Presenting the data (trained network)



# Results – Voice Recognition

Performance of trained network

- Recognition accuracy between known “Hello”s
  - 100%

- Recognition accuracy between new “Hello”’s
  - 100%

# Any questions?