Deep Learning

(Artificial Neural Networks)

Machine Learning vs Deep Learning

Scenario:

A machine needs to identify, from a given photograph, whether it is a **car** or a **plane**

ML technique

Identify a list of features for both cars/planes

Algorithm identifies class based on features

DL technique

Identifies edges for both cars/planes

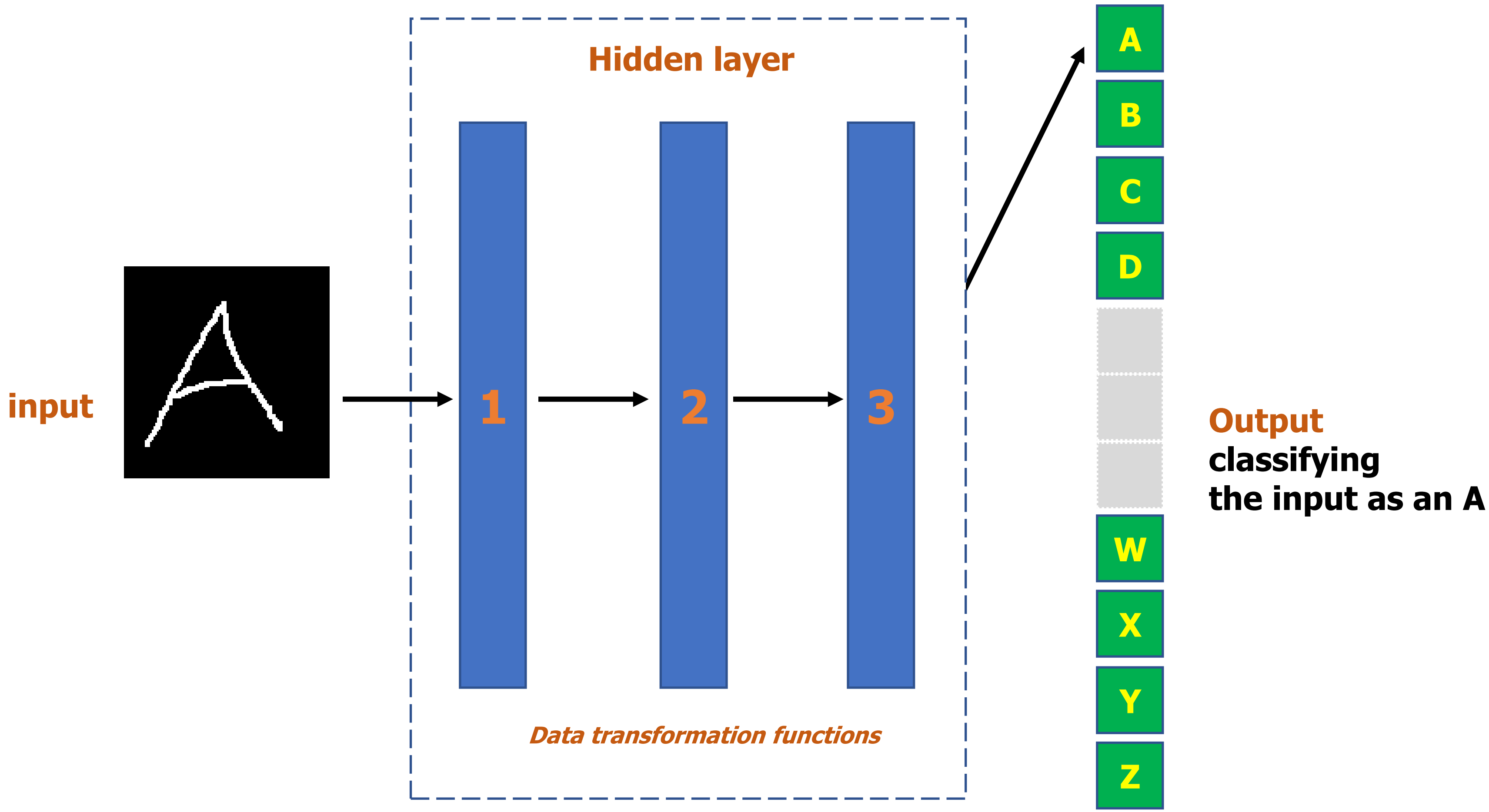
Builds consecutive hierarchical combination of edges for classification

Machine Learning vs Deep Learning

Machine learning	Deep learning
Performs very well even with small data	Performs best on a huge dataset
Works reasonably well on small/mid configuration computers	Requires high-end machines – hardware dependent (GPU)
Features are provided in the data	Features are learnt from the data
Training time is usually less	Training time is more
Easy to interpret	Not easy to interpret

Deep Learning = Machine Learning

- Deep Learning, at a high level, is taking an input, transforming it into an output, through successive/multiple layers of transformation.
- Transformation is done to get useful information regarding data
- Nested hierarchy of concepts



Introduction

- One of the most powerful machine learning algorithms today
- Used in
  - Classification (binary class, multiclassification etc.)
  - Regression (predict multidimensional Y)
- Artificial Neural Networks **(ANN)** models the functionality of the human brain
- Consists of number of neurons (nodes)
- Nodes receive inputs and pass it for further processing – either serves as input to the next node or the final result

Activation Functions

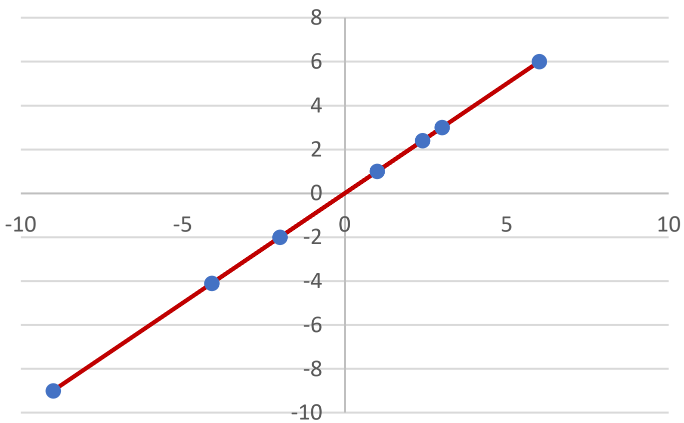
- A function that decides the output of a node, based on the inputs for that node
- **Activation function** is applied to the inputs
- Output can either be
  - Any value between 0 and 1
  - Any value
- Also used to impart non-linearity
- Activation functions greatly impact the results / accuracy in ANN
- Some activation functions suffer from ***"vanishing gradients".*** So, choice of activation functions is important to get the best results
- Hidden layers and Output layers use different Activation functions

Some common Activation Functions

- Identity
- Binary Step
- Logistic / Sigmoid
- Hyperbolic tangent (**Tanh**) (goes below 0 [-1,0,1])
- Rectified Linear Unit (**ReLU**)
- Leaky ReLU
- SoftMax (for multi-classification)
- Gaussian
- Linear, etc.

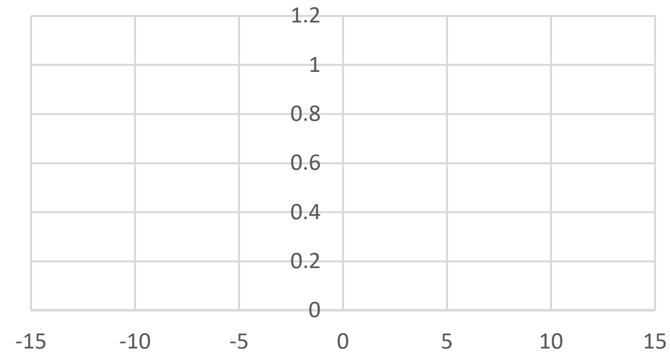
- **Identity**

$y = x$



- **Sigmoid**

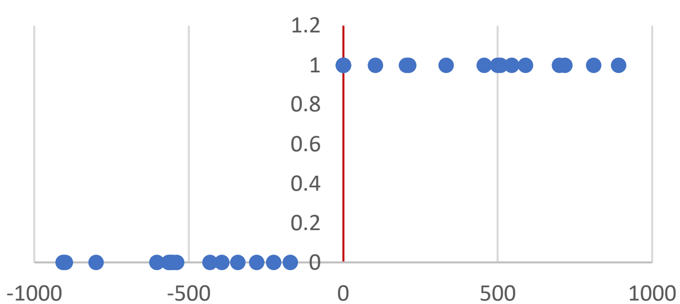
$y = 1/\ (1+e^{-x})$



- **Binary Step**

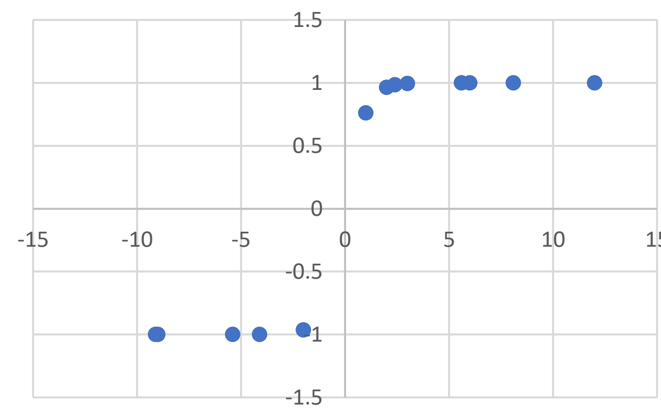
$y = 0 : x < 0$

$y = 1 : x >= 0$



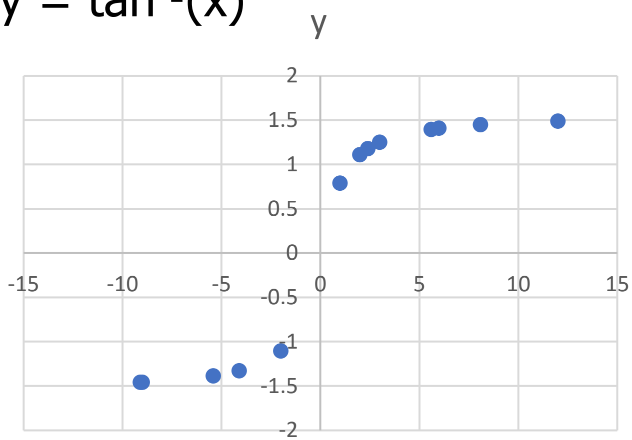
- **Tanh**

$y = [2/\ (1+e^{-2x})]-1$



- **ArcTan**

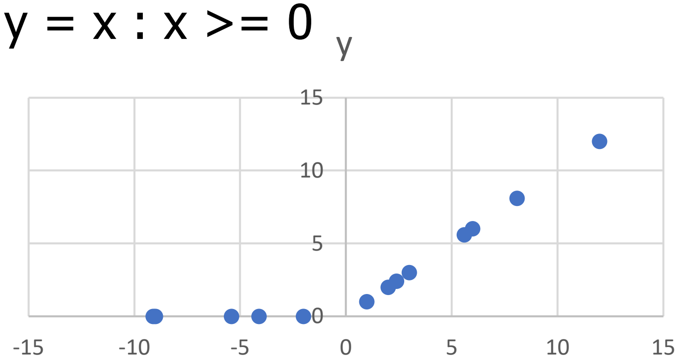
$y = \tan^{-1}(x)$



- **ReLU**

$y = 0 : x < 0$

$y = x : x >= 0$



- **Softmax**

$y = e^x / \Sigma e^{xk}$

x	e^x (num)	Σ num	y	Total
1	2.718		0.00427	
2	7.389		0.011606	
3	20.086		0.03155	
4	54.598		0.085761	
5	148.413		0.233122	
6	403.429		0.633691	
		636.633		
				1

Layers

- Neurons are organised in Layers

- Layers can be of different types
  - Dense (Fully Connected)
  - Convolutional
  - Pooling
  - Recurrent
  - Normalization

- Layers perform different transformations on the inputs

- Different layers are used for solving different sets of problems

Dense (Fully Connected)

- Most commonly used layer
- Uses the ***Sequential model*** to stack layers where each layer has one input and output tensor
- Output = **Activation** ($\sum_{i=1}^{n}$ (input $_n$ $*$ input_weight $_n$) + bias **)**

Types of Neural Network

Simple NN
For classification and regression

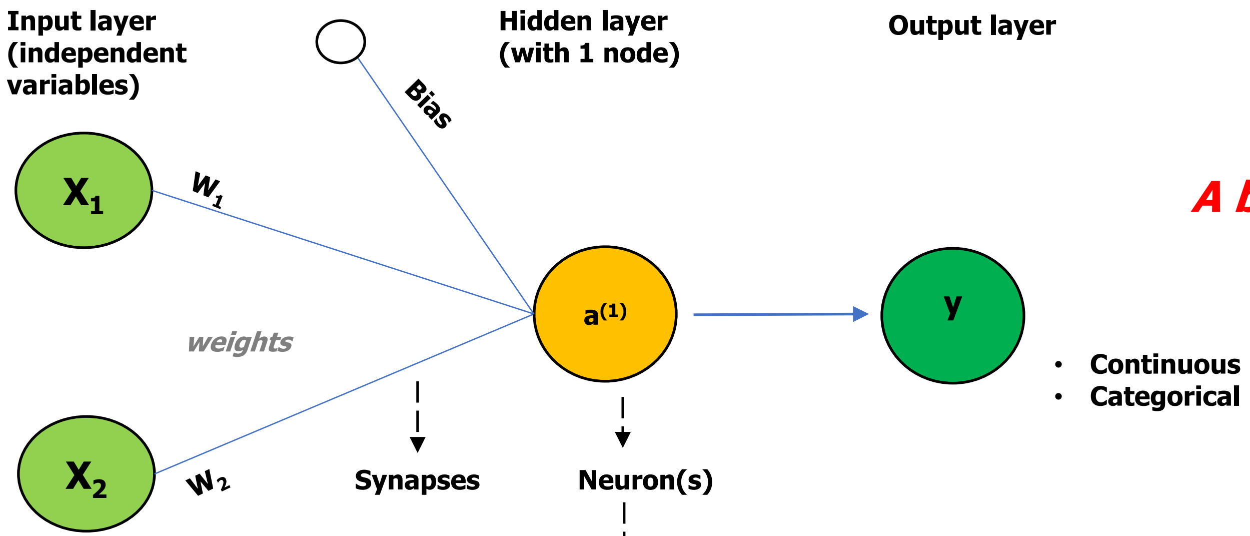
Convolutional NN
For image processing and recognition

Long short-term memory network
For speech recognition

Artificial Neural Network

- ANN can learn patterns in data
- ANN can approximate any non-linear function – a handy tool for engineering problems
- Designing a network is more of an art than science – requires lots of trial and error to come up with the best model
- All nodes are identical and contains:
  - sum unit
  - function unit
- Input data needs to be standardized / normalize

Nodes – a perspective



Perceptron

A basic architecture

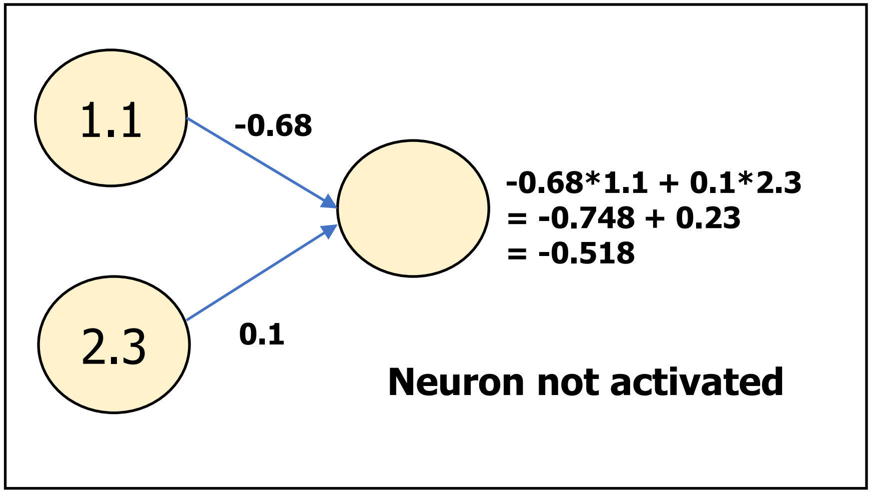
$X_1.W_1 + X_2.W_2 + b$ (bias)

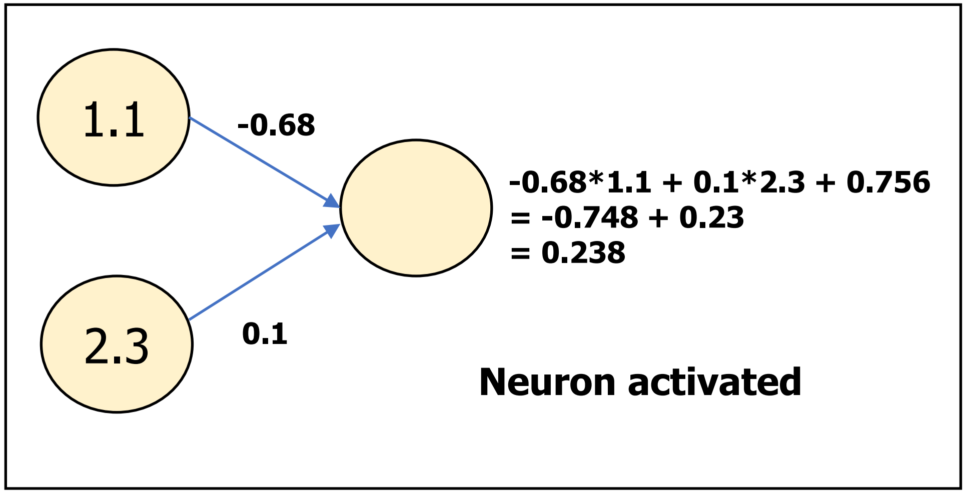
Activation needs to be a number between 0 and 1

Sigmoid ($X_1.W_1 + X_2.W_2 + b$ (bias) **)** $\longrightarrow$ $\dfrac{1}{1 + e^{-x}}$

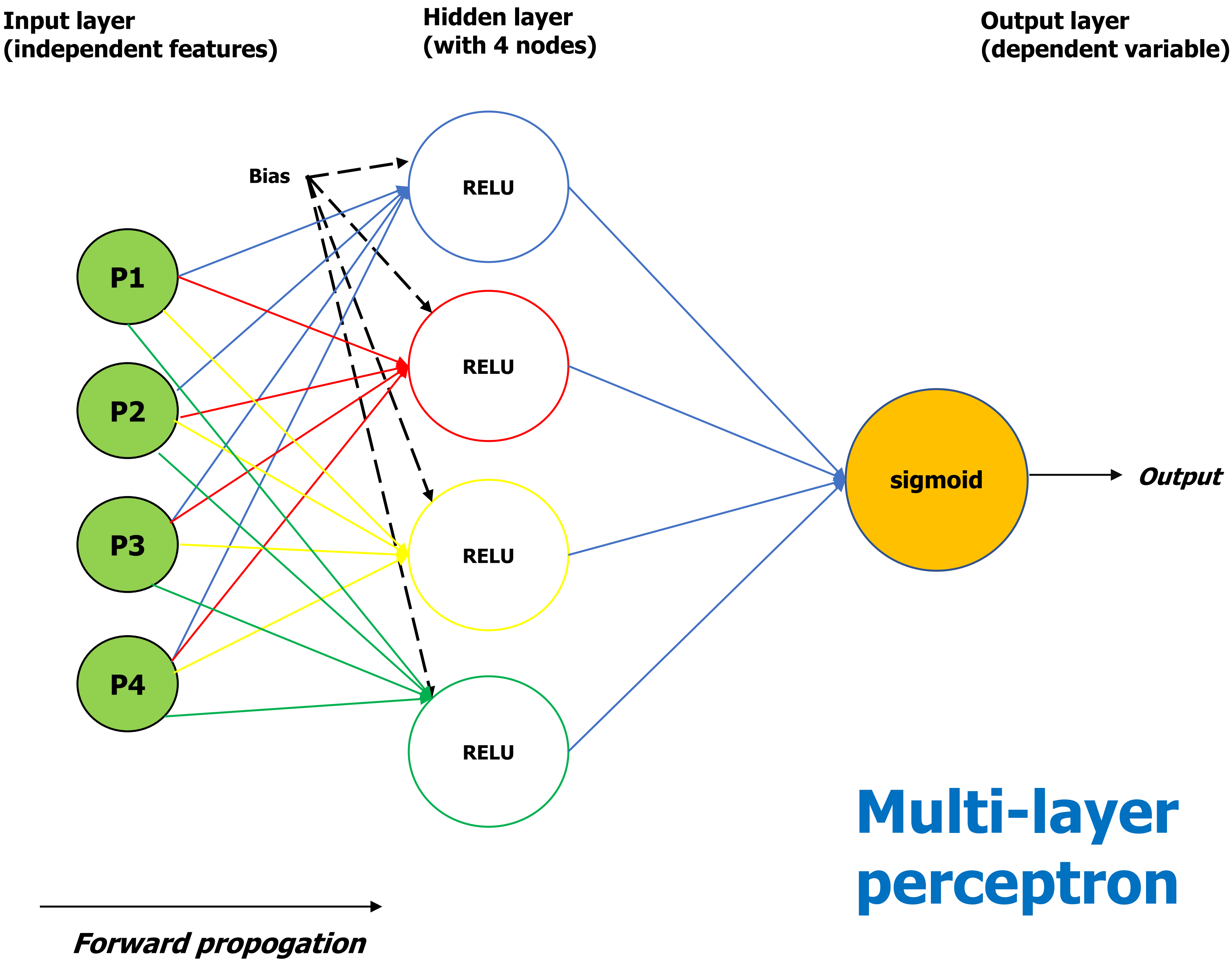
Bias

- Each Neuron has a bias
- It can be learnt – just like the weights
  - During the model building process, the biases are initialised with random values
- Makes the model flexible
- Determines if a neuron is activated



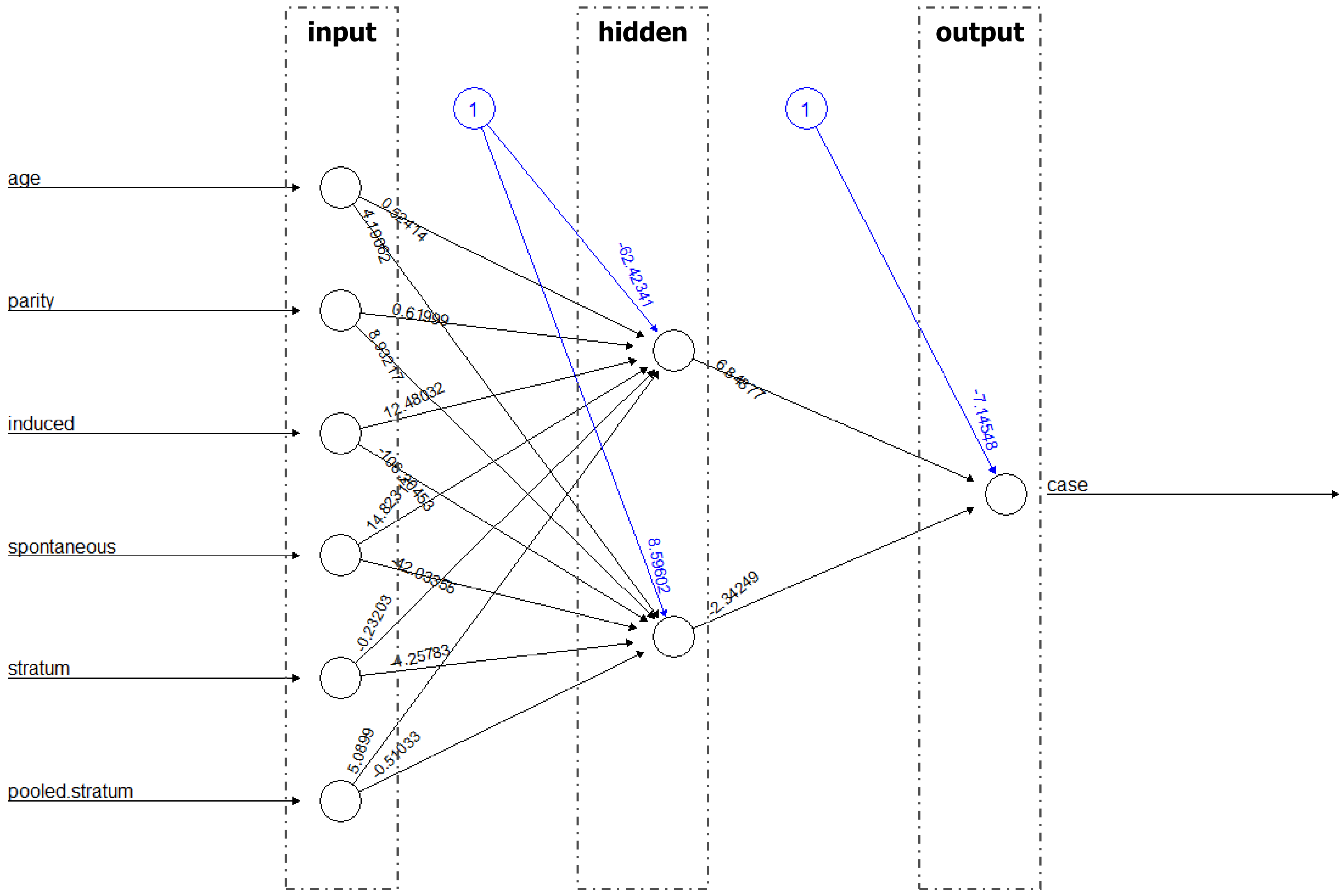


So, Bias is important in an Artificial Neural Network



Multi-layer perceptron

Actual representation of a Neural Network



Nodes

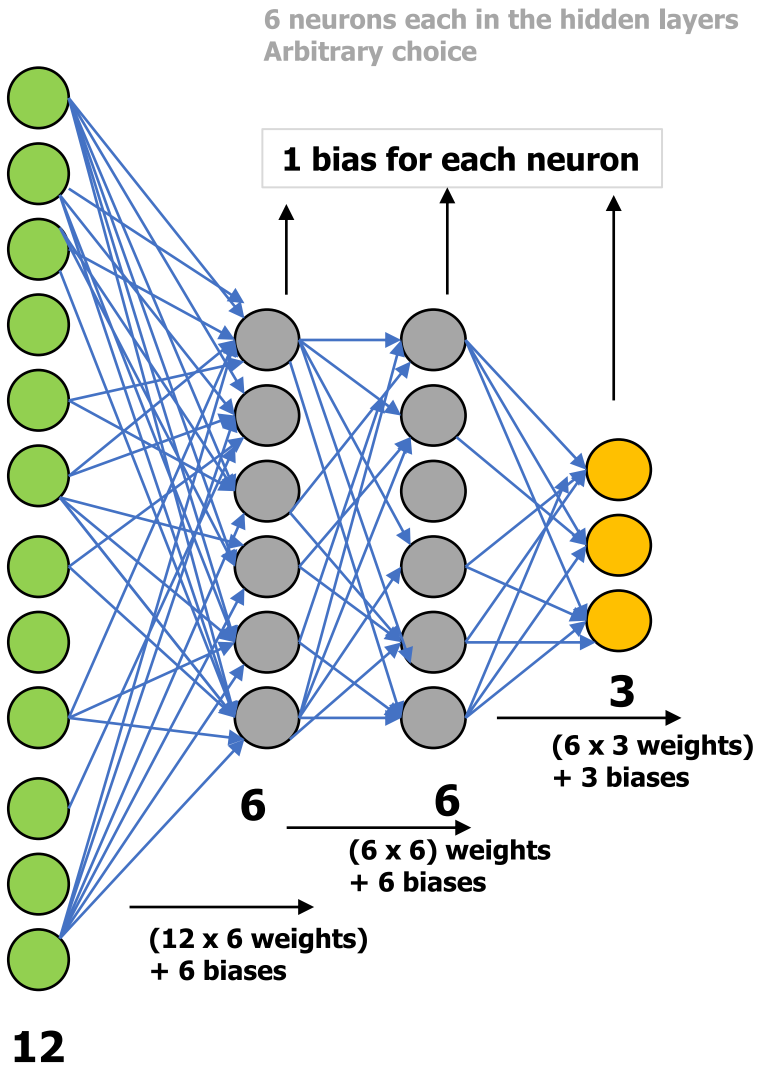
- Well defined input and output nodes
- Well directed connections that tells the direction of information
- Connections need to have different values –
  - more important vs less important
- Achieved by a concept called "connection weights"
  - "Connection Weights" decide which information in a node is important
  - Represented by the "Errors"

- Activation of one layer determines activation of the next layer

- **Transfer functions**
  - Nodes decide what to do with information
  - It is a maths equation
- Sends value to the next node and so on till it reaches output node

Weights

Weights are strength of connections between units, usually between 0 and 1



(72 + 36 + 18) = 126 weights
(6 + 6 + 3) = 15 biases
Total = 141

Learning rate
Finding the right Weights and Biases

Loss/Cost function

- It is a function that tells how good or bad the Neural Network is for a certain task
- Intuitive way
  - Take each training example
  - Pass it through the NN to get the number
  - Subtract it from the actual number
  - Square it (to ensure all error terms are positive)

$$L(y,\hat{y})=\frac{1}{m}\sum_{i=1}^{m}(y_i-\hat{y}_i)^2$$

y – number wanted from NN
y_i – training example
ŷ – predicted value of training

- Loss function should be as small as possible

Gradient Descent Optimization

- Optimization method used to find the values of the parameters (a, b_n) [coefficients] of a function Ŷ that minimises the cost function

- Gradient descent is used when the parameters cannot be calculated analytically

- Searched using an optimization algorithm
- Regression uses Gradient Descent to minimise the Error terms

- By taking small / big steps, we get closer to the global minimum, by adjusting the **learning rate**
  - Too small a value for learning rate → more number of iterations to arrive at the minimum value
    - The difference between Learning rate 0.1 and 0.01 is huge, though both are small numbers

  - Too big a value for learning rate → overshoot the minimum value
    - Need to go back and forth and keep readjusting the rates

$$E^2 = \sum_{1}^{N} (acty - pre\,\mathrm{d}y)^2$$

$$E^2 = (y - \hat{y})^2$$

$$\hat{y} = \omega x + b$$

$$E^2 = (y - \mathrm{wx} \; - \mathrm{b})^2$$

$$\frac{\partial E}{\partial w} = -2(y - wx - b)(x)$$

$$\frac{\partial E}{\partial b} = -2(y - wx - b)$$

$$\frac{\partial E}{\partial w} = -\frac{2}{N} \Sigma (actx)(E)$$

$$\frac{\partial E}{\partial b} = -\frac{2}{N} \Sigma (E)$$

Train Errors
Validation Errors
Test Errors

- Use a validation set to measure the ability of the model to generalize on unseen data.

- Don’t bother to look at accuracy on the train set itself, unless to change hyperparameters such as learning rate.

- If your goal is to achieve the best model on unseen data, then you should pick the model that has the best accuracy on validation.

- You expect train to overfit, and sometimes you need train to overfit a lot before validation achieves a desired performance.

Training the Neural Network

- Connection Weights are determined by learning
- NN are very slow learners
- Learning is done using a technique called "back propogation"

Back propogation process

- Random connection weights are assigned
- For a set of inputs, pre-decide on some outputs
- Using random weights, calculate some outputs
- Compare output with desired outputs
- Chances of the 2 outputs being equal is less
- Find the difference (Errors [y- ŷ])
- Adjust connection weights to minimise the errors
- Uses old weight, error, input node, learning rate
- The node with the maximum error is adjusted most

Cycle repeats .....

- This is done for each input set
- Can change the number of nodes
- Can change the learning rate

Training the Neural Network - 2

- Training iteration (epoch) → when the network is shown all the training data, one at a time
- Training continues over multiple iterations, until the weights reach a steady value / maximum iterations reached
- Overfitting → Network memorises data rather than generalising data
- To overcome this problem, split data into training and testing

Problems with deep learning

- Black-box processing; hard to interpret
- Requires a large amount of data to learn -> hence it is a slow learner
- Computationally very expensive
- Model can easily crack
    - Tweaking a photo a little bit makes no difference to a human eye, but a DL is most likely to misinterpret

Libraries for Deep Learning

- **Tensorflow**
- **Keras**
- theano
- torch