Deep Learning

Ref: https://towardsdatascience.com/deep-learning-101-for-dummies-like-me-a53e3caf31b1
https://www.forbes.com/sites/bernardmarr/2018/10/01/what-is-deep-learning-ai-a-simple-guide-with-8-practical-examples/#702b1b298d4b

Examples of deep learning

- Virtual assistants

  Whether it's Alexa or Siri or Cortana, the virtual assistants of online service providers use deep learning to help understand your speech and the language humans use when they interact with them.

- Translations

  In a similar way, deep learning algorithms can automatically translate between languages. This can be powerful for travelers, business people and those in government.

- Vision for driverless delivery trucks, drones and autonomous cars

  The way an autonomous vehicle understands the realities of the road and how to respond to them whether it's a stop sign, a ball in the street or another vehicle is through deep learning algorithms. The more data the algorithms receive, the better they are able to act human-like in their information processing—knowing a stop sign covered with snow is still a stop sign.

Examples of deep learning

- Chatbots and service bots

  Chatbots and service bots that provide customer service for a lot of companies are able to respond in an intelligent and helpful way to an increasing amount of auditory and text questions thanks to deep learning.

- Image colorization

  Transforming black-and-white images into color was formerly a task done meticulously by human hand. Today, deep learning algorithms are able to use the context and objects in the images to color them to basically recreate the black-and-white image in color. The results are impressive and accurate.

- Facial recognition

  Deep learning is being used for facial recognition not only for security purposes but for tagging people on Facebook posts and we might be able to pay for items in a store just by using our faces in the near future. The challenges for deep-learning algorithms for facial recognition is knowing it's the same person even when they have changed hairstyles, grown or shaved off a beard or if the image taken is poor due to bad lighting or an obstruction.

Examples of deep learning

- Medicine and pharmaceuticals

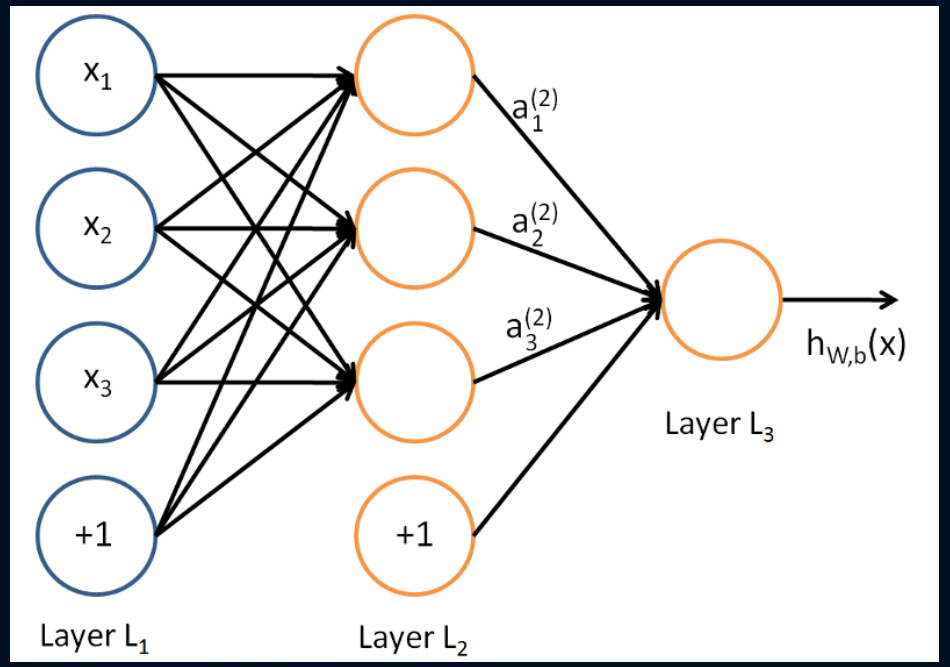
  From disease and tumor diagnoses to personalized medicines created specifically for an individual's genome, deep learning in the medical field has the attention of many of the largest pharmaceutical and medical companies.

- Personalized shopping and entertainment

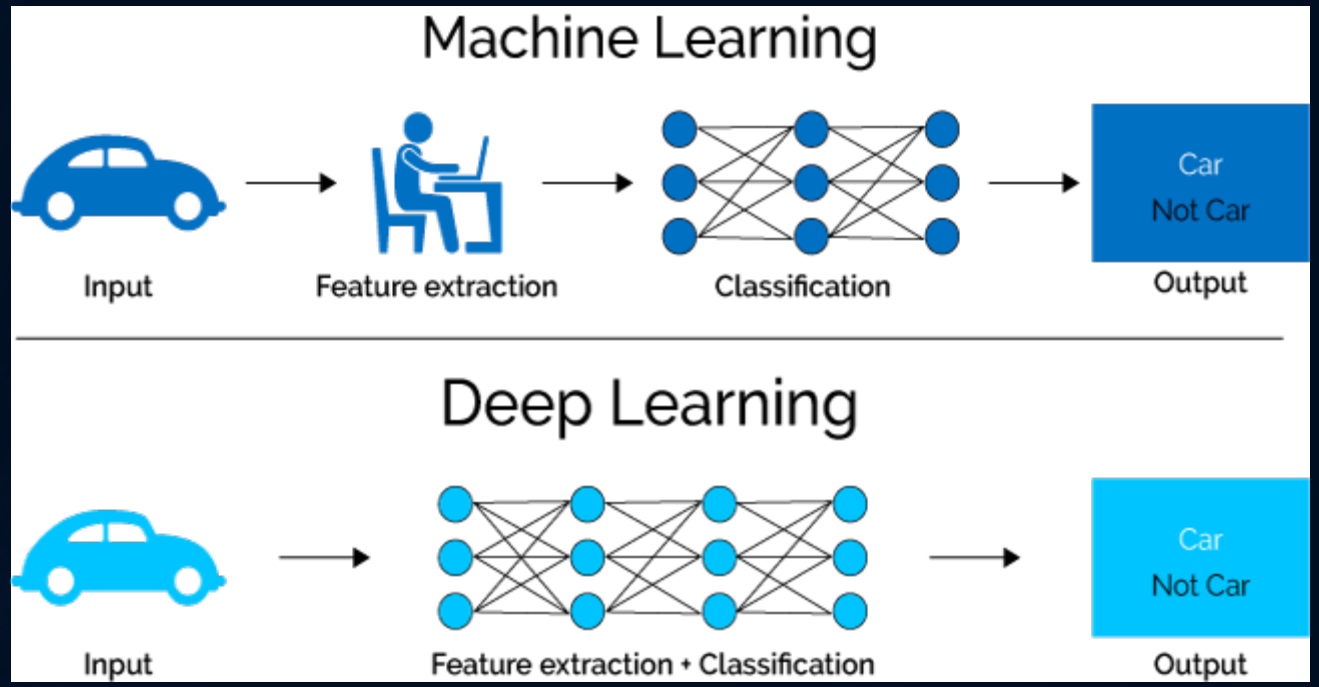
  Ever wonder how Netflix comes up with suggestions for what you should watch next? Or where Amazon comes up with ideas for what you should buy next and those suggestions are exactly what you need but just never knew it before? Yep, it's deep-learning algorithms at work.

What is Deep Learning ?

- **What is Deep Learning all about?**
- **Deep learning** is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data inspired by the structure and function of the brain called artificial neural networks.



How Deep Learning differ from Traditional Machine Learning?



What are activation functions all about?

- ***Activation functions*** are really important for an Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable. They introduce *non-linear properties to our Network*. ***Their main purpose is to convert an input signal of a node in an A-NN to an output signal.*** That output signal now is used as an input in the next layer in the stack.
- One of the simplest activation functions is the **Heaviside step function**. This function returns a **0** if the linear combination is less than 0. It returns a **1** if the linear combination is positive or equal to zero.

$$f(h) = \begin{cases} 0 & \text{if } h < 0 \\ 1 & \text{if } h \geq 0 \end{cases}$$

What is the error and loss function?

- In most learning networks, the error is calculated as the difference between the actual output and the predicted output.
- The function that is used to compute this error is known as Loss Function(J). Different loss functions will give different errors for the same prediction, and thus have a considerable effect on the performance of the model. One of the most widely used loss function is mean square error(MSE), which calculates the square of the difference between actual value and the predicted value. Different loss functions are used to deal with different type of tasks, i.e. regression and classification. Thus, loss functions are helpful to train a neural network. Given an input and a target, they calculate the loss, i.e difference between output and the target variable. Loss functions fall under four major category:
- Regressive loss functions which are used in case of regressive problems, that is when the target variable is continuous. Some examples are: Mean Square Error, Absolute Error & Smooth Absolute Error

What is the error and loss function?

- Classification loss functions used when the target variable y, is a binary variable, 1 for true and -1 for false. Some examples are: Binary Cross Entropy, Negative Log Likelihood, Margin Classifier & Soft Margin Classifier
- Embedding loss functions which deal with problems where we have to measure whether two inputs are similar or dissimilar. Some examples are:
- 1. L1 Hinge Error- Calculates the L1 distance between two inputs.
- 2. Cosine Error- Cosine distance between two inputs.

How did Gradient Descent come into play?

- *"A **gradient** measures how much the output of a function changes if you change the inputs a little bit." — **Lex Fridman (MIT)***
- **Gradient descent** is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost). Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm. Gradient descent is used to find the minimum error by minimizing a "cost" function.
- **Stochastic Gradient Descent** performs a parameter update for each training example, unlike normal Gradient Descent which performs only one update. Thus it is much faster. Gradient Decent algorithms can further be improved by tuning important parameters like **momentum**(which determines the velocity with which learning rate has to be increased as we approach the minima), **learning rate**(a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient) etc.

Forward Propagation and Back Propagation

- In neural networks, you **Forward Propagate** to get the output and compare it with the real value to get the error. Now, to minimize the error, you propagate backward by finding the derivative of error with respect to each weight and then subtracting this value from the weight value. This is called **Back Propagation**,

Forward Propagation

The input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.

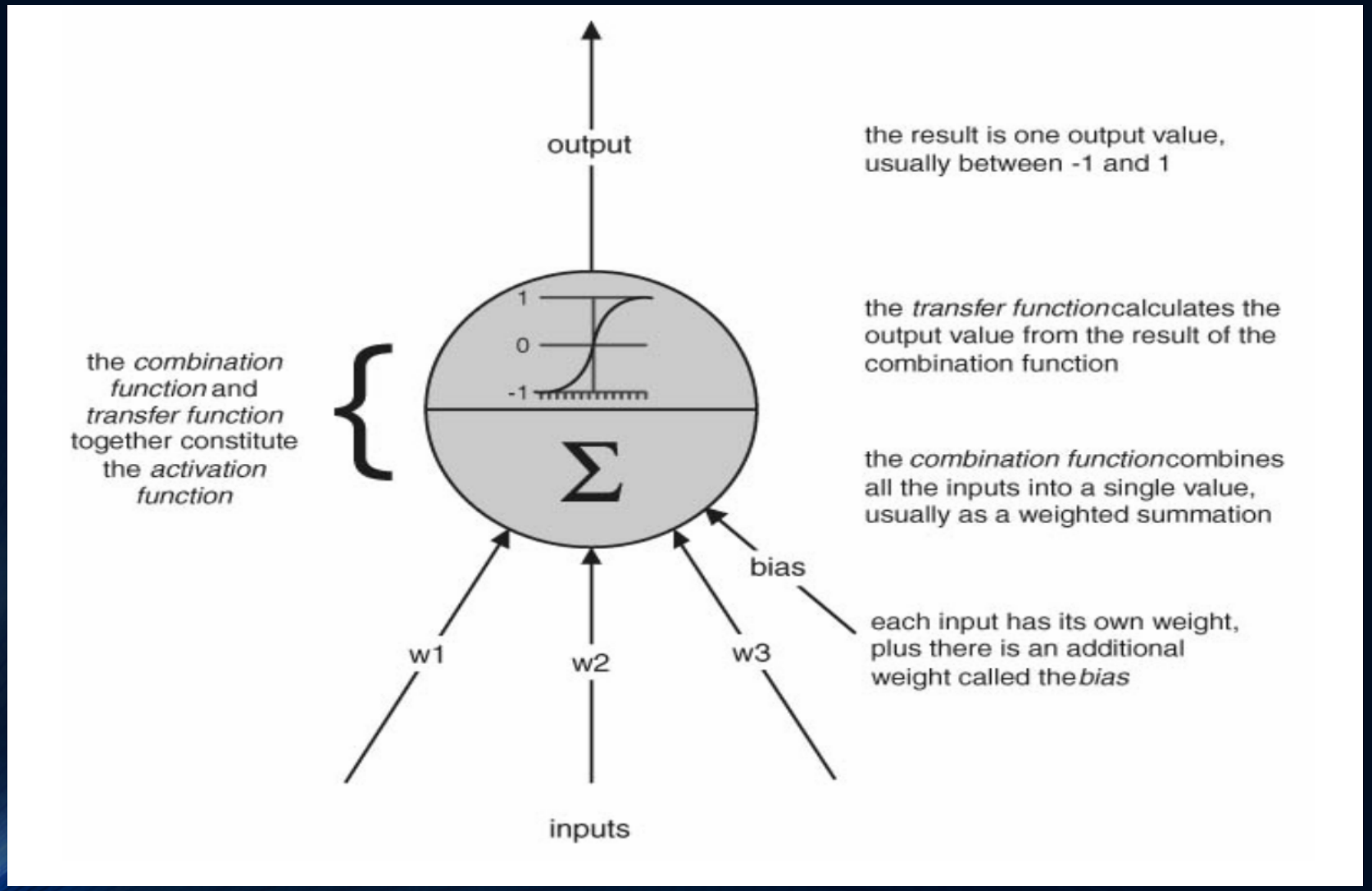
The data should not flow in reverse direction during output generation otherwise it would form a cycle and the output could never be generated. Such network configurations are known as ***feed-forward network***.

Forward Propagation

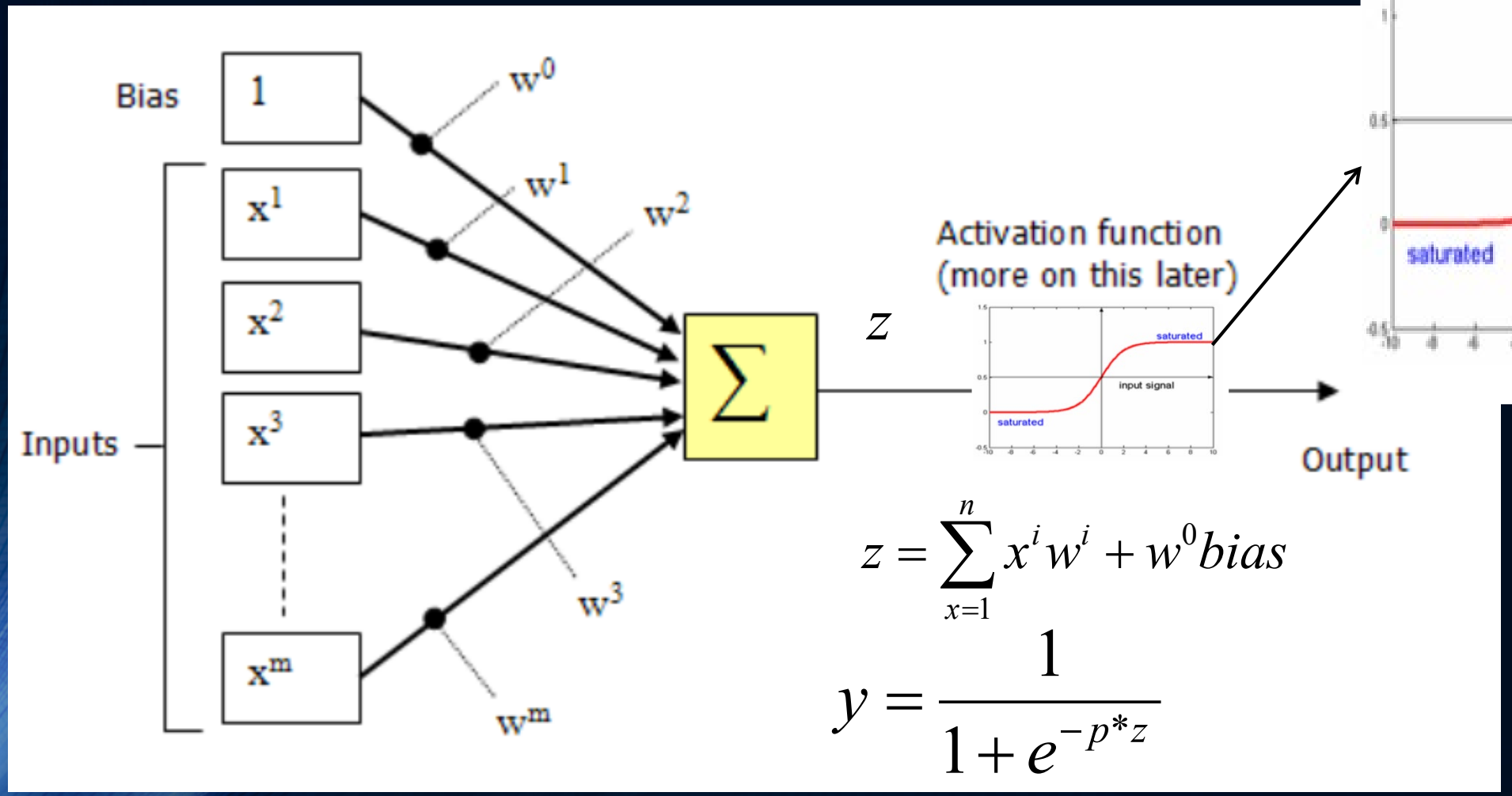
- At each neuron in a hidden or output layer, the processing happens in two steps:

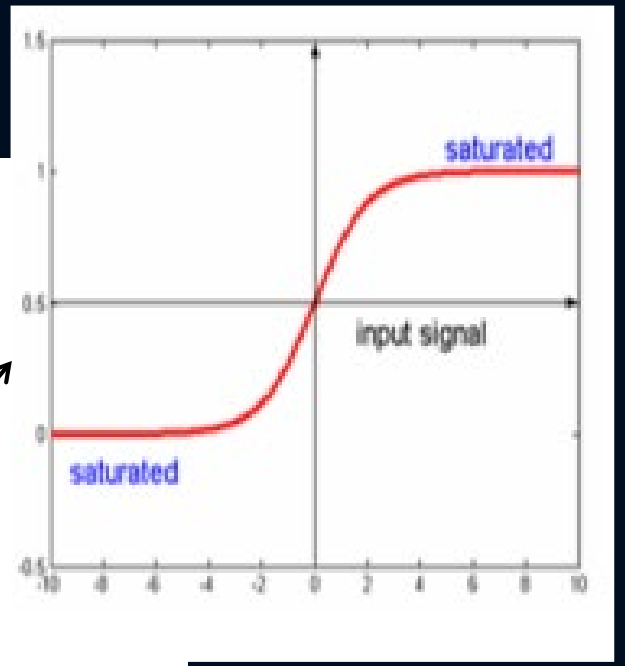
1. **Preactivation:** it is a *weighted sum of inputs* i.e. the *linear transformation of weights* w.r.t to inputs available.
2. **Activation:** the calculated weighted sum of inputs is passed to the activation function. An activation function is a mathematical function which adds non-linearity to the network. There are four commonly used and popular activation functions — sigmoid, hyperbolic tangent(tanh), ReLU and Softmax.

Artificial Neuron



Output With Step Activation function

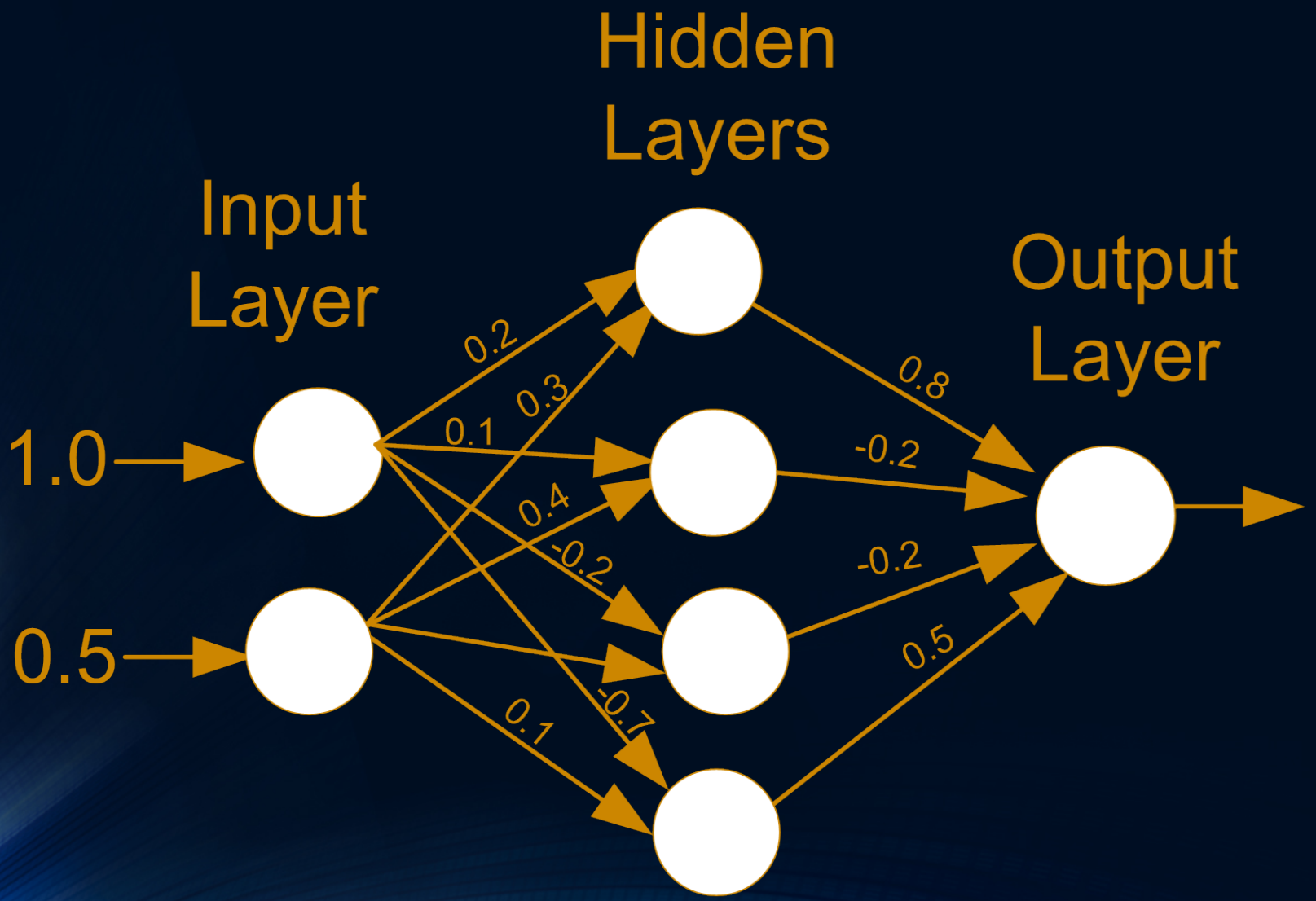




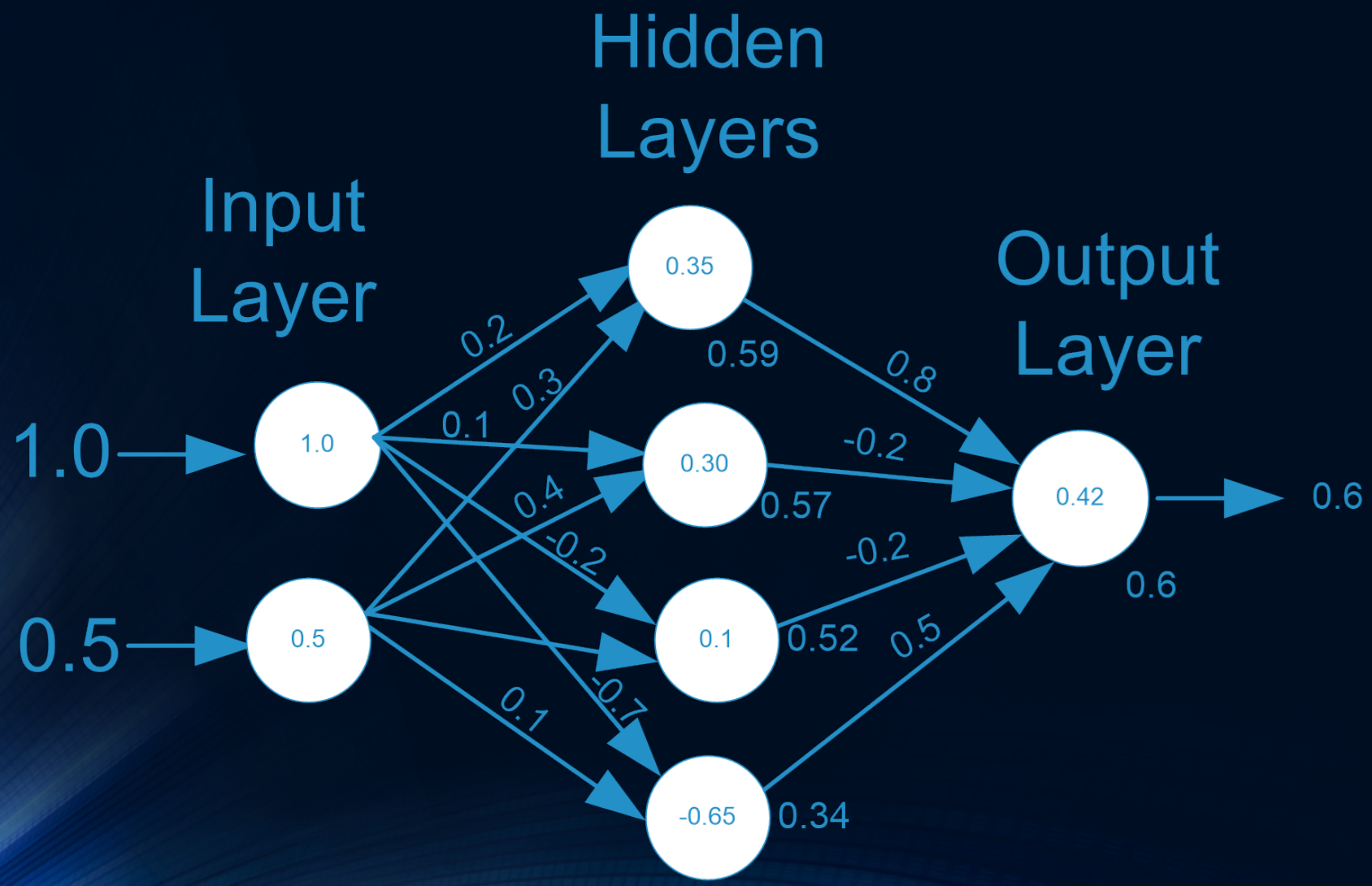
$$z = \sum_{x=1}^{n} x^i w^i + w^0 bias$$

$$y = \frac{1}{1 + e^{-p*z}}$$

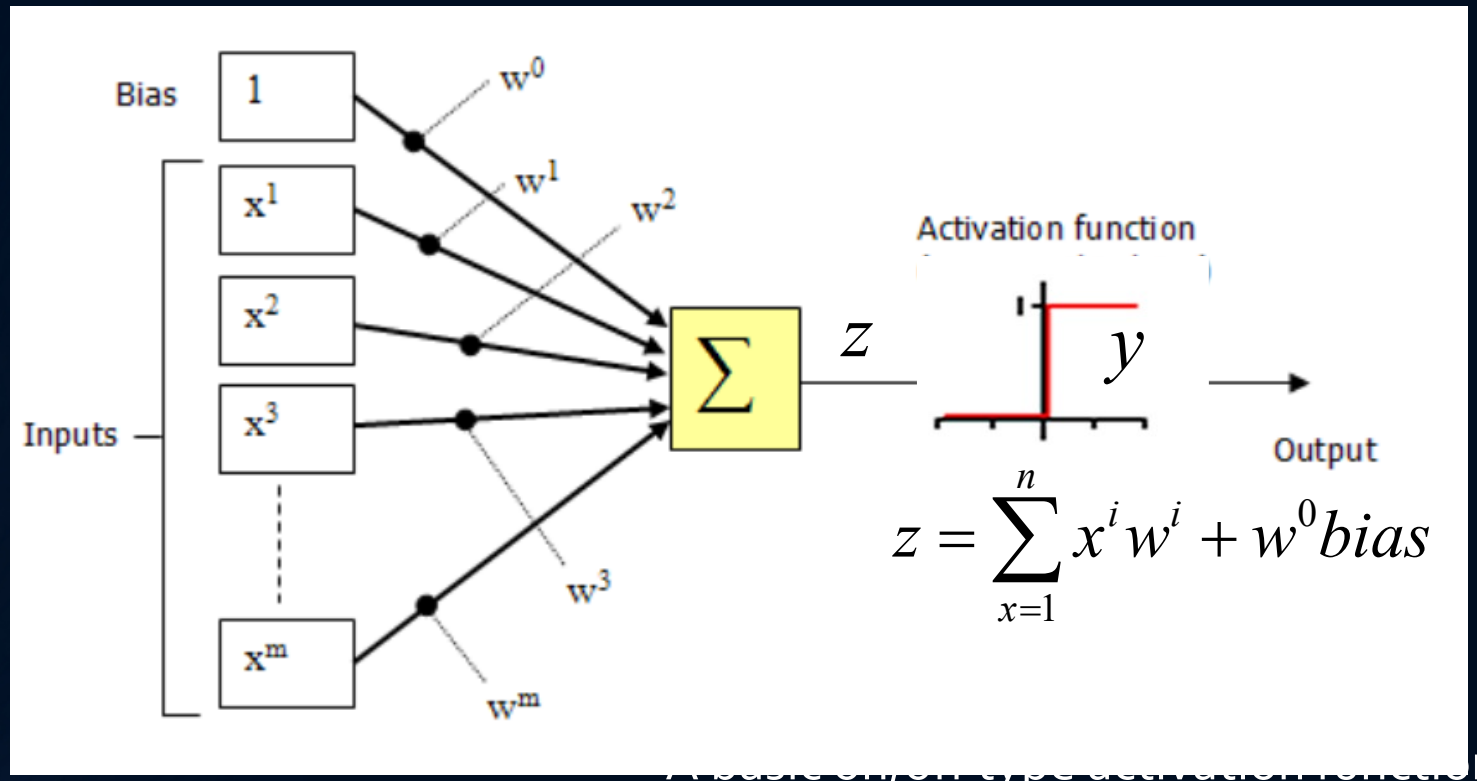
Feedforward example



Feedforward Example

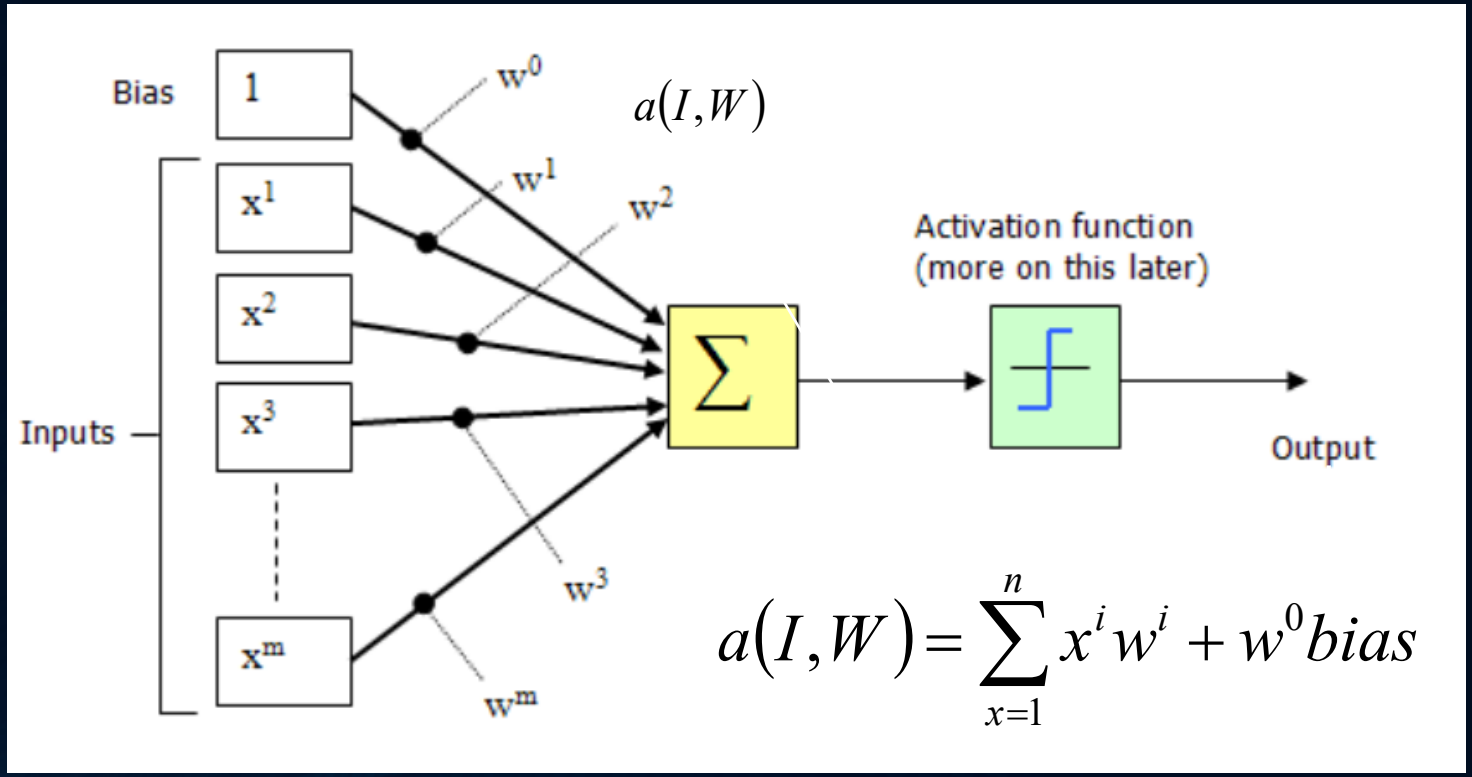


Output With Step Activation function

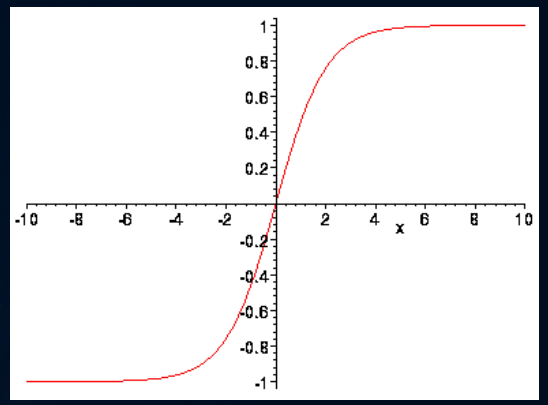


- A basic on/off type activation function
- if z<o then output y is o
- if z>= o then output y is 1

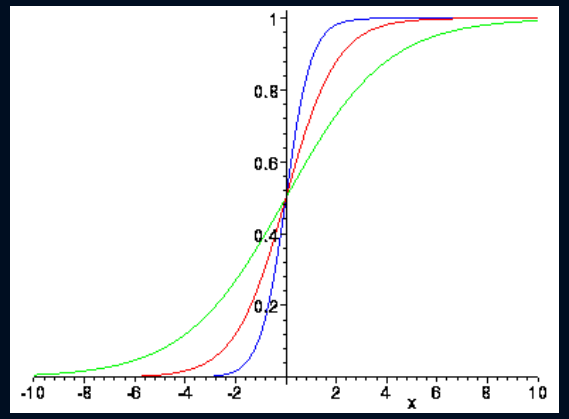
Neuron Model



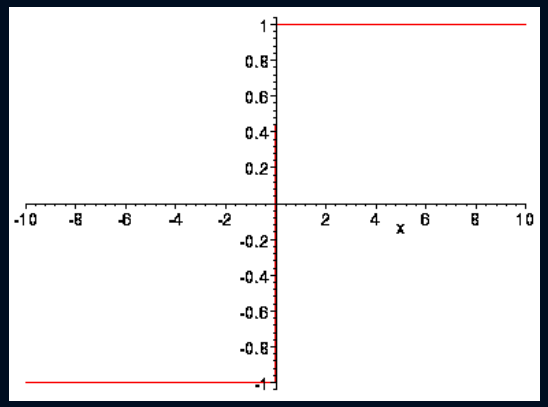
Activation functions



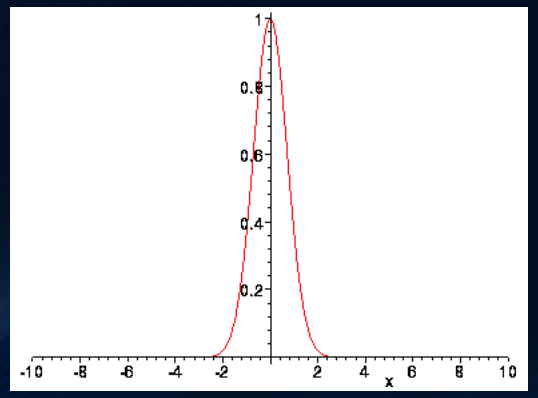
Symmetricc Sigmoid Function



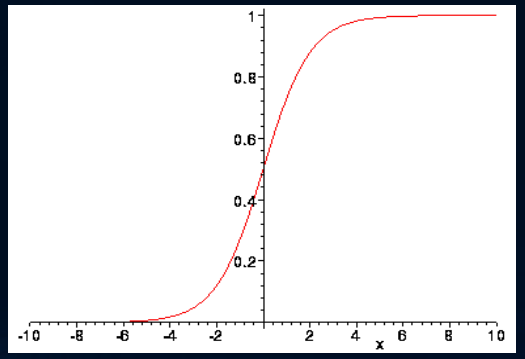
Sigmoid Functions



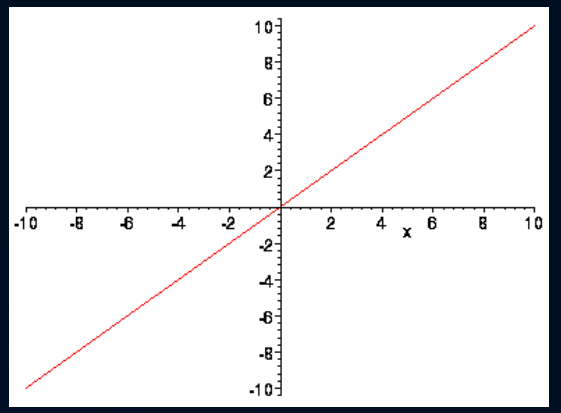
Step/Heaviside Function



Radial Basis Function



Sigmoid Function



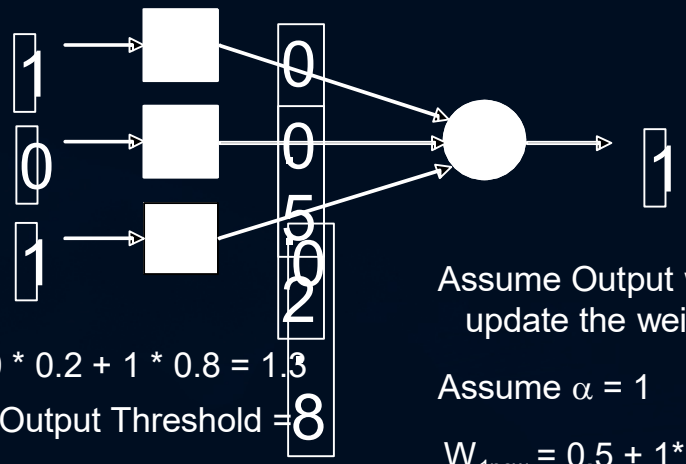
Identity Function

How Do Perceptron Learn?

- Uses supervised training
- If the output is not correct, the weights are adjusted according to the formula:

- $w_{new} = w_{old} + \alpha(desired - output)*input$

α is the learning rate



1 * 0.5 + 0 * 0.2 + 1 * 0.8 = 1.3

Assuming Output Threshold = 1.2

1.2 1.3 > 1.2

Assume Output was supposed to be 0
update the weights

Assume $\alpha = 1$

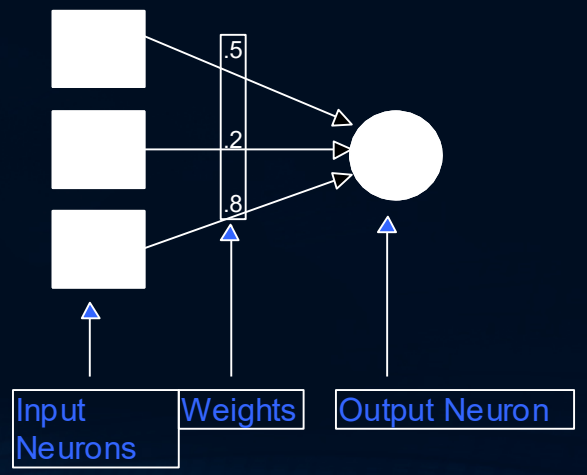
$W_{1new} = 0.5 + 1*(0-1)*1 = -0.5$

$W_{2new} = 0.2 + 1*(0-1)*0 = 0.2$

$W_{3new} = 0.8 + 1*(0-1)*1 = -0.2$

Perceptrons

- First neural network with the ability to learn
- Made up of only input neurons and output neurons
- Input neurons typically have two states: ON and OFF
- Output neurons use a simple threshold activation function
- In basic form, can only solve linear problems
  - Limited applications

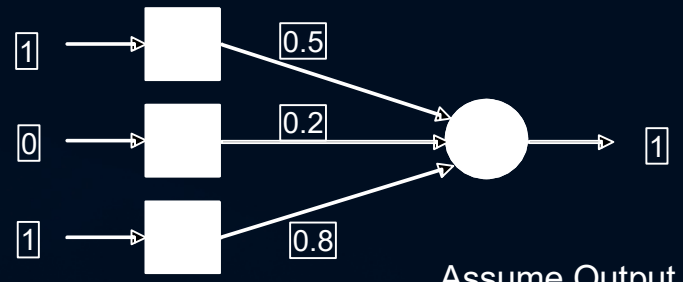


How Do Perceptrons Learn?

- Uses supervised training
- If the output is not correct, the weights are adjusted according to the formula:

$w_{new} = w_{old} + \alpha(desired - output)*input$

α is the learning rate



1 * 0.5 + 0 * 0.2 + 1 * 0.8 = 1.3

Assuming Output Threshold =

1.2 1.3 > 1.2

Assume Output was supposed to be 0
update the weights

Assume $\alpha = 1$

$W_{1new} = 0.5 + 1*(0-1)*1 = -0.5$

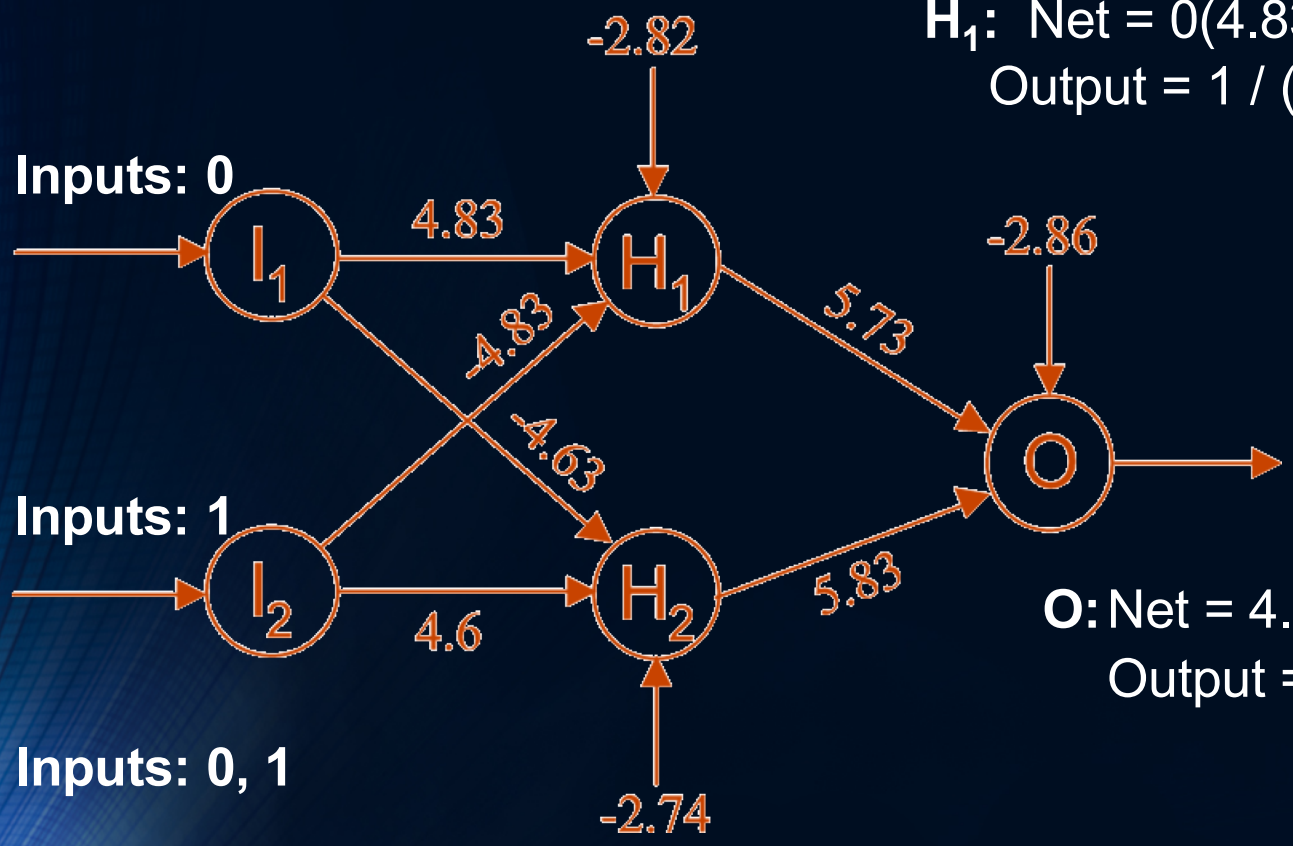
$W_{2new} = 0.2 + 1*(0-1)*0 = 0.2$

$W_{3new} = 0.8 + 1*(0-1)*1 = -0.2$

Multilayer Feedforward Networks

- Most common neural network
- An extension of the perceptron
  - Multiple layers
    - The addition of one or more “hidden” layers in between the input and output layers
  - Activation function is not simply a threshold
    - Usually a sigmoid function
  - A general function approximator
    - Not limited to linear problems
- Information flows in one direction
  - The outputs of one layer act as inputs to the next layer

XOR Example



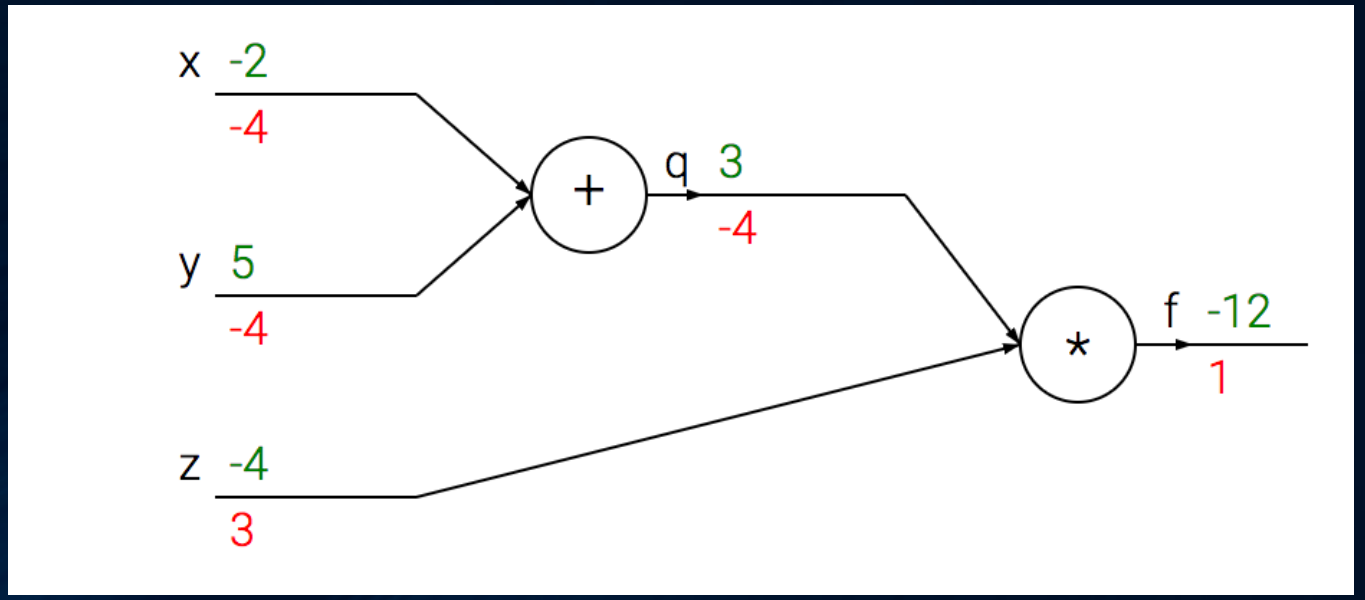
H_1: Net = 0(4.83) + 1(-4.83) – 2.82 = -7.65
Output = 1 / (1 + $e^{7.65}$) = 4.758 x 10^{-4}

O: Net = 4.758 x 10^{-4}(5.73) + 0.8652(5.83) – 2.86 = 2.187
Output = 1 / (1 + $e^{-2.187}$) = 0.8991 ≡ "1"

H_2: Net = 0(-4.63) + 1(4.6) – 2.74 = 1.86
Output = 1 / (1 + $e^{-1.86}$) = 0.8652

Back Propagation Algorithm

Lets see what Back propagation Algorithm doing?



Deep Learning: Back Propagation

- Now the problem that we have to solve is to update weight and biases such that our cost function can be minimised. For computing gradients we will use Back Propagation algorithm.

Why Back Propagation?

- Is to correct the error of the intial weights
- Given the a cost function of with weights C=C(w) alone. The weight are numbered $w_1, w_2, \ldots,$
- For Bacck propagation we want to compute $\partial C/\partial w_i$ for some particular weight w_i. An obvious way of doing that is to use the approximation

$$\frac{\partial C}{\partial w_i} \approx \frac{C(w + \epsilon e_i) - C(w)}{\epsilon},$$