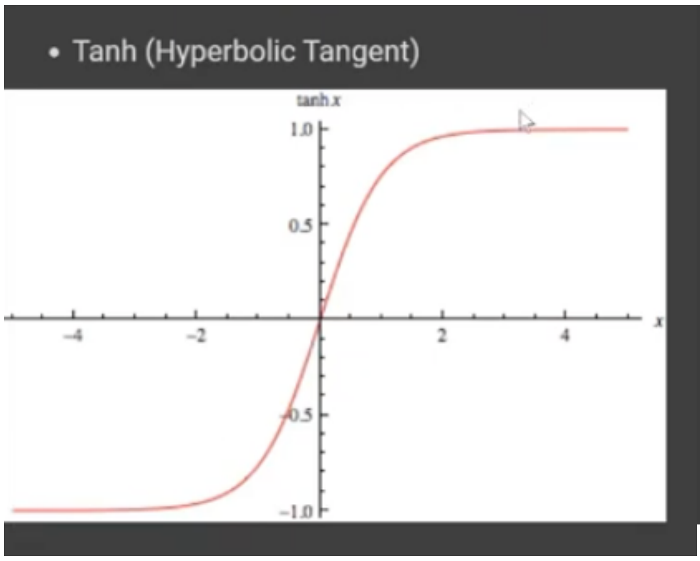
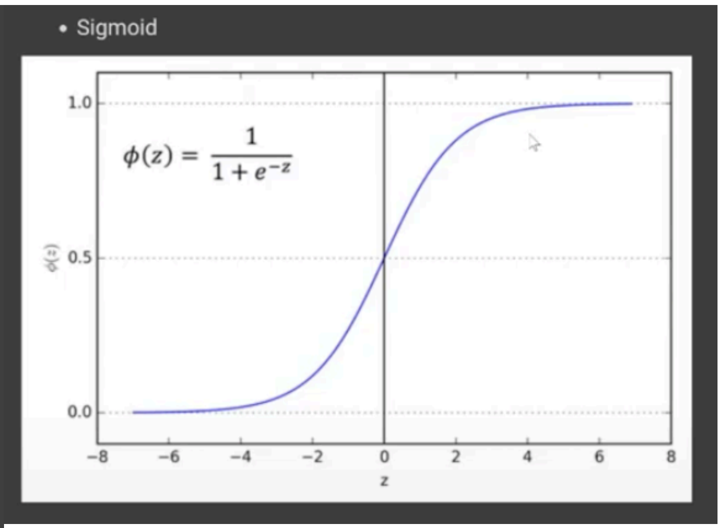
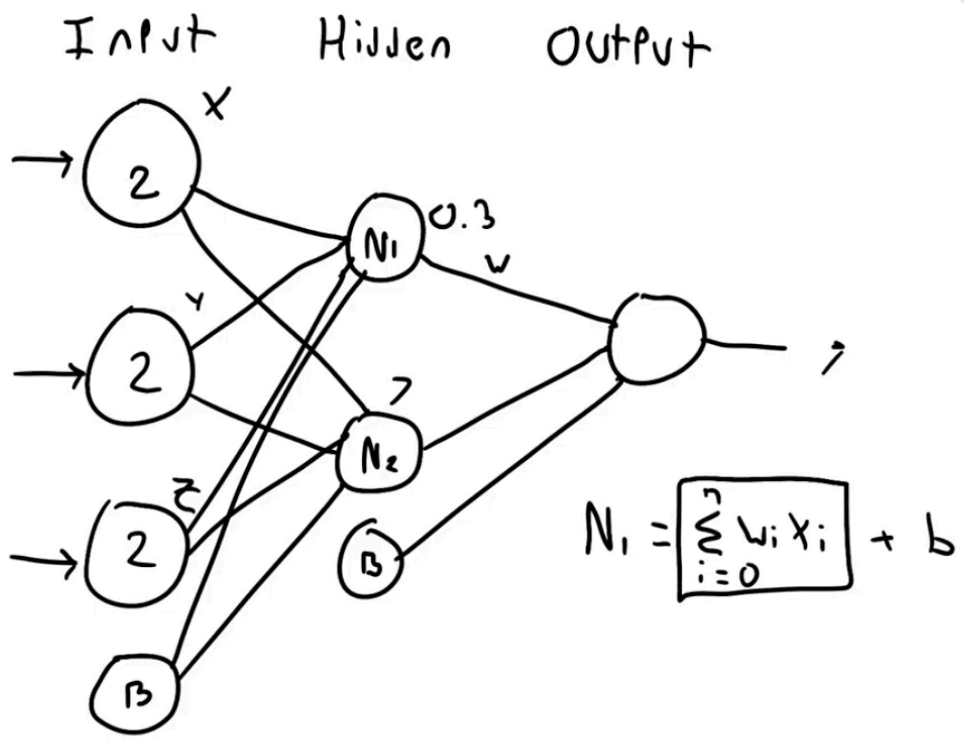
- ***Activation functions <- functions which transform the output of a neural network***
  - -> you want the output to be in between 0 and 1 -> as a probability that the outcome which is associated with that node is true
  - -> activation functions are used to normalise the outputs at each node
    - making any negative value 0 <- it's taking the norm
    - then tanh -> to normalise the outputs between -1 and 1





    - -> we also have sigmoid -> which makes the outputs in between 0 and 1
    - -> we are taking the outputs of the neural networks and normalising them using an activation function
      - -> so they are all positive
      - -> and either all between -1 and 1, or between 0 and 1
        - -> you can also have them between ± infinity
      - -> this is to make sure that they resemble probability distributions
  - ***-> how those activation functions are used***
    - -> not just for the outputs of the entire model -> after each transformation / layer in the neural network
    - -> we need to know which activation function to use
    - -> it gives some value between 0 and 1
    - -> to introduce complexity into the neural network -> those can be complex functions
      - -> complex activation functions
      - -> so we can predict more different outputs
    - -> activation functions increase the dimensionality of the model <- when training the neural network
      - -> it's spreading out the predictions more / increasing the number of dimensions
      - -> when you use activation functions it makes the predictions more pronounced
      - -> increasing the dimensionality of the model tells you more information / allows you to make more predictions



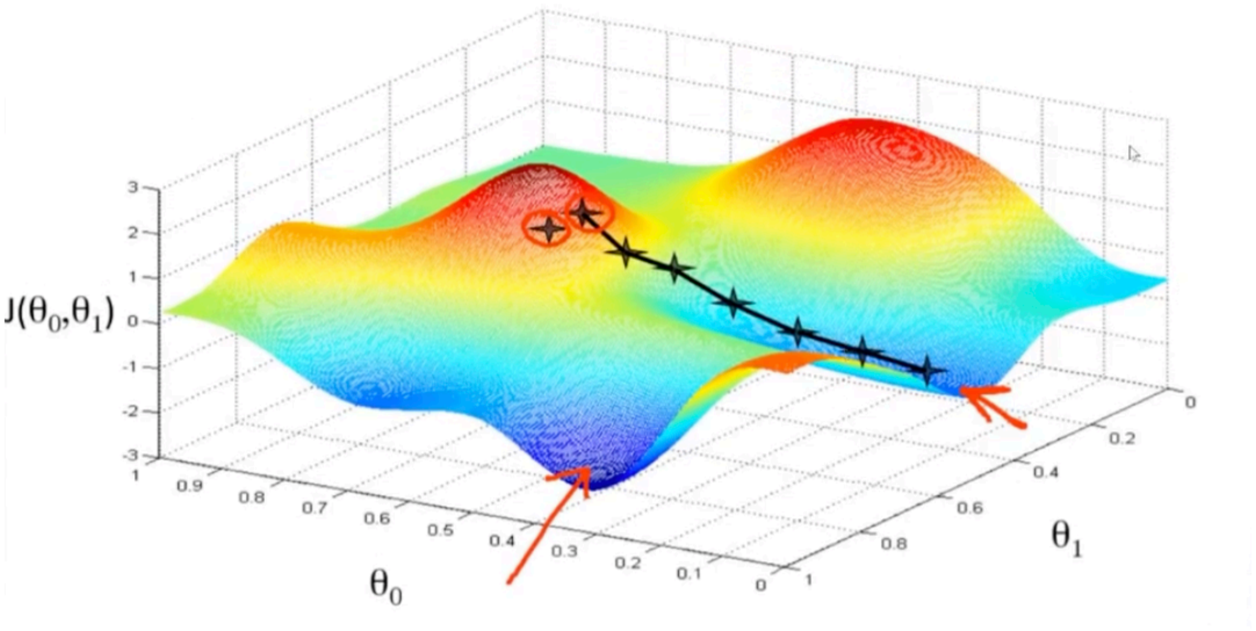
        - <u>-> b in this image is the bias of the model</u>

- ***How neural networks train***
    - -> weights and biases
    - <u>-> these are the coefficients and the constants which the model comes up with as it's trained</u>
    - ***-> loss functions***
        - -> we give the model data and it trains
        - -> <u>the loss function tells us how good or how bad the model is</u>
        - <u>-> the model needs tweaking</u>
        - <u>-> this is gradient descent</u>
        - <u>-> if the model is tripe them the points need to be tweaked a lot more -> it's calculating how bad the model is - and the worse it is the more tweaks need to be made</u>
        - -> mean absolute error

$$mae = \frac{\sum_{i=1}^{n} abs\left(y_i - \lambda(x_i)\right)}{n}$$

        - ***-> there are thee loss functions***
            - -> mean squared error
            - -> mean absolute error
            - -> huge error
            - <u>-> they are also called cost functions / loss functions <- you want the network to cost the least</u>
- ***Using gradient descent to update the weights and parameters in the model***
    - -> it determines if the function is getting better or not
    - -> gradient descent
    - -> the neural network function is moving in a probability landscape
    - -> when the model is trained it is moving along that landscape
    - -> <u>the activation functions change howit moves in that landscape</u>
    - **-> <u>minimising the loss function</u>**

- ***-> we are looking for a global minimum -> the point at the least possible loss of the function***
- -> calculating the gradient and moving in that direction
  - -> back propagation then goes through the model and updates the weights and constants in the model



- ***-> the biases move the function left and right e.g***
  - -> it's adding another parameter to the function
  - -> the weighted sum of a neurons -> and the inputs are all of the different neurons which are feeding into it from the previous layer
    - with activation functions -> to normalise its value
    - -> then we add a bias
    - -> then we look at the loss function at that value and apply an algorithm which minimises it
    - -> we are making predictions and comparing them to the expected ones using the loss functions
    - -> then back propagation goes back through the network and updates its weights and biases according to the ones which were garner at the end of the last iteration of the model
    - -> you carry on repeating the process until the loss / cost function is minimised -> in other words the predictions are more accurate
- -> tanh activation functions move the transform the output values to be between -1 and 1