**Backpropagation**[**¶**](https://hub.coursera-notebooks.org/user/iotijzgpoxuikpyniyodnj/notebooks/Backpropagation.ipynb#Backpropagation)

**Instructions**[**¶**](https://hub.coursera-notebooks.org/user/iotijzgpoxuikpyniyodnj/notebooks/Backpropagation.ipynb#Instructions)

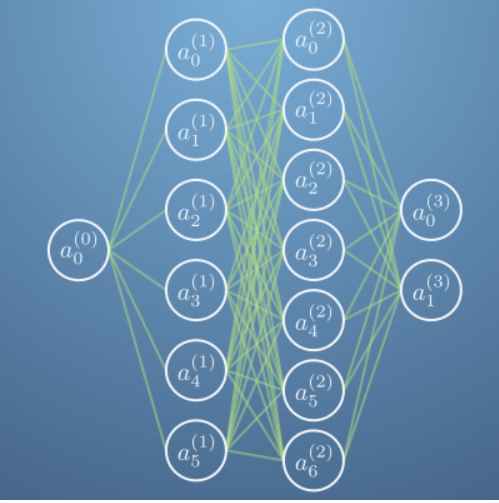
In this assignment, you will train a neural network to draw a curve. The curve takes one input variable, the amount travelled along the curve from 0 to 1, and returns 2 outputs, the 2D coordinates of the position of points on the curve.

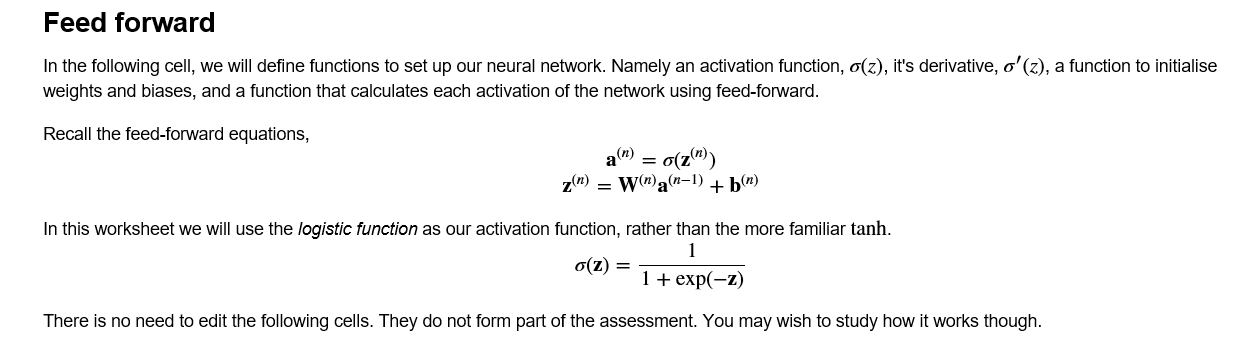
To help capture the complexity of the curve, we shall use two hidden layers in our network with 6 and 7 neurons respectively.

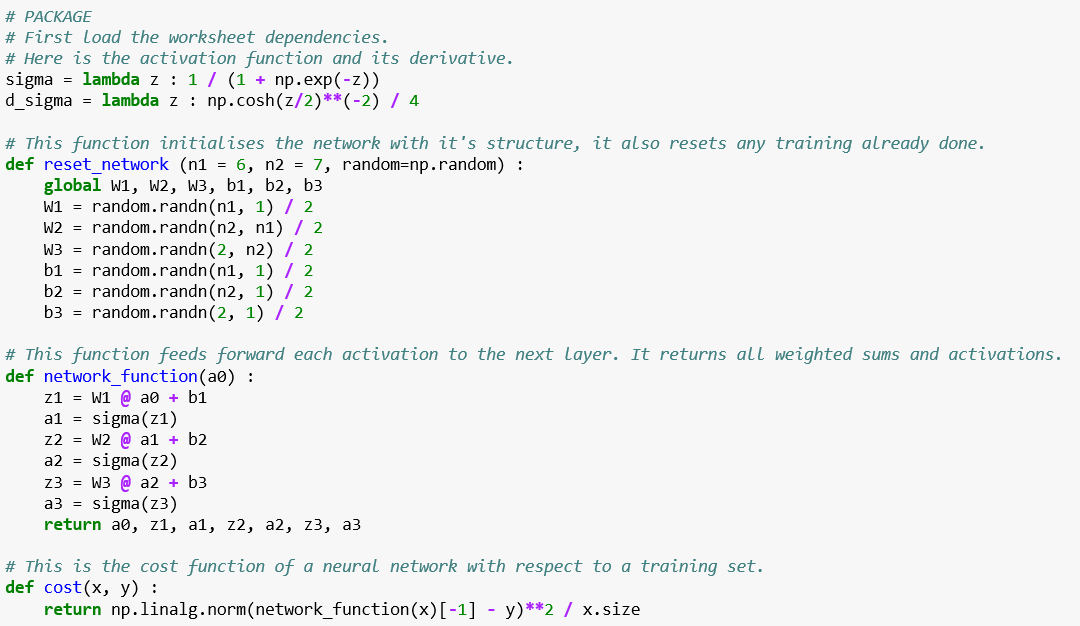
Backpropagation is used to minimise the cost function, i.e. where the difference between the actual and predicted output is small. A network model is fully trained to give the desired output when the weights and bias are adjusted when at the values where the cost is the minimum. This is done by calculating the partial derivative (or Jacobian – the vector pointing the direction of steepest slope) of Cost function with respect to the weight and bias, and setting it to zero. This called Gradient Descent.

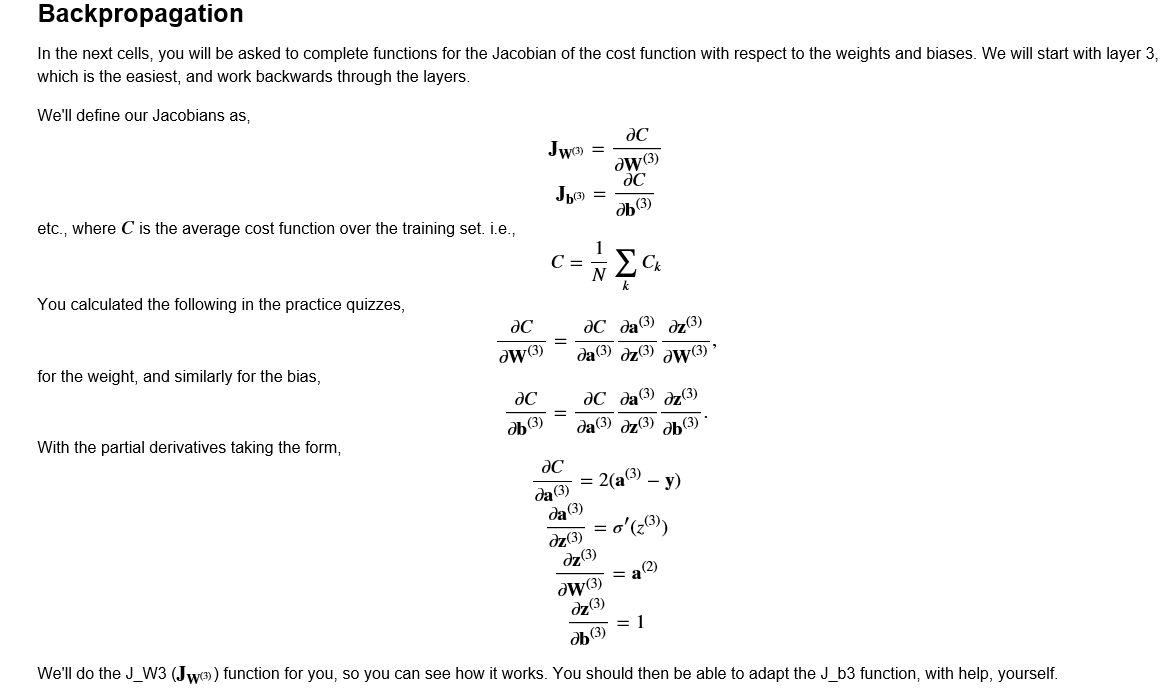
The partial derivative (Jacobian) of the cost function is the error term and is the same as delta that is backpropagated through the network to update the weights and bias.

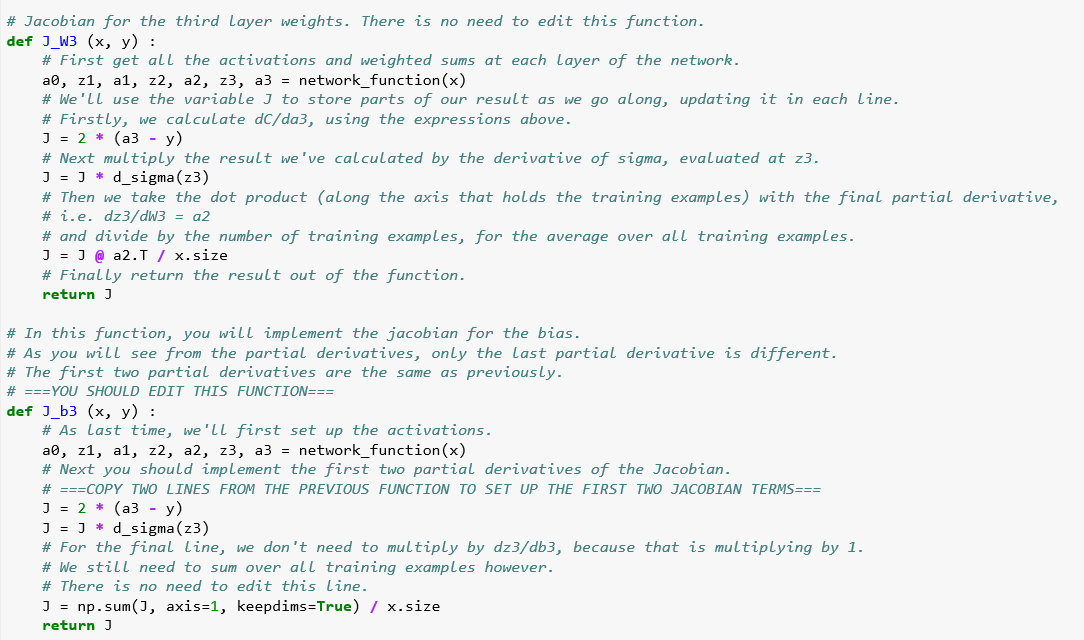
Alpha is the learning rate







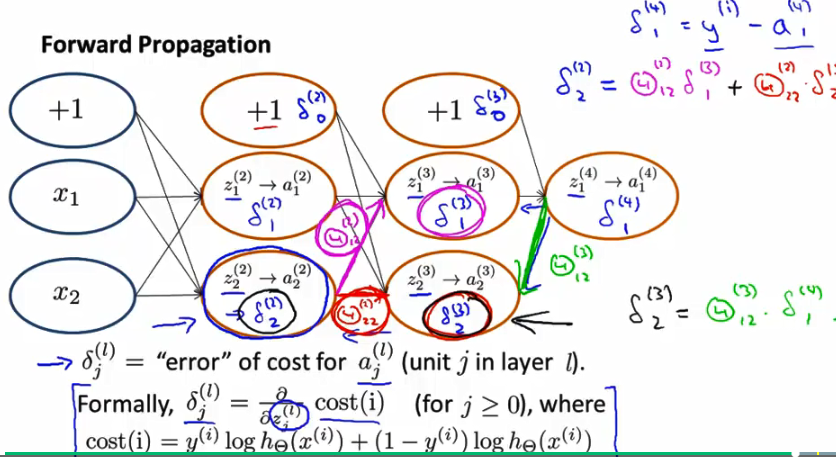


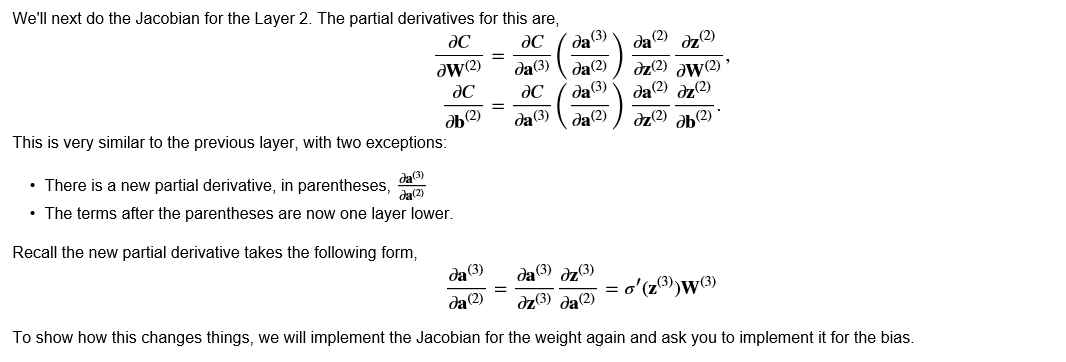


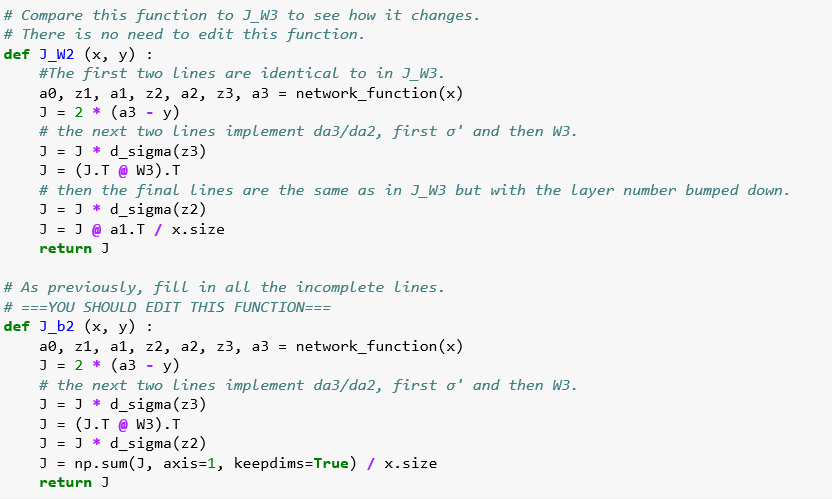
**Extra explanation**

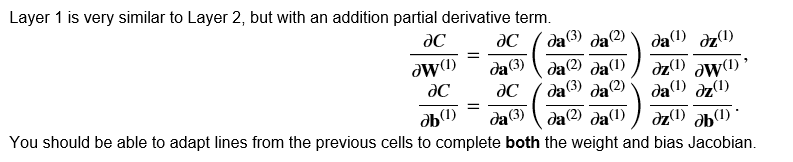
dC/dW (i) = activation(i) \* delta(i+1)

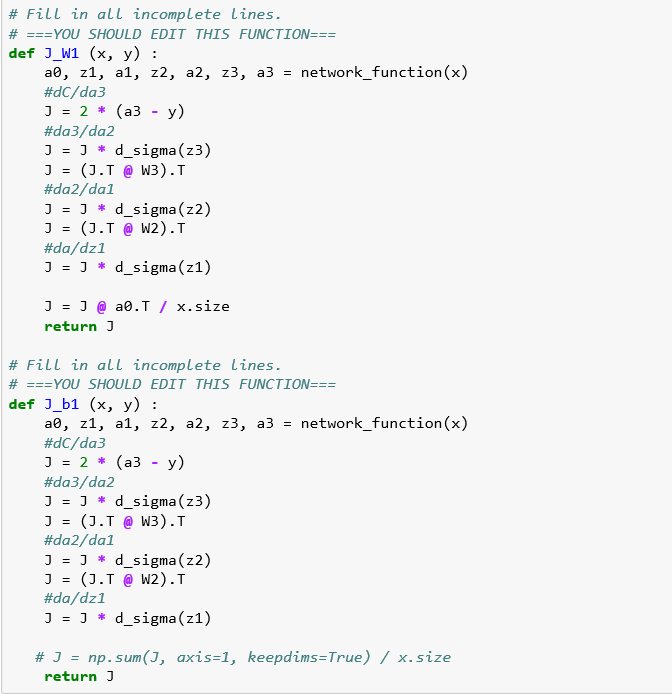
i is the layer number. So in the formula, an extra da3/da2 is added.



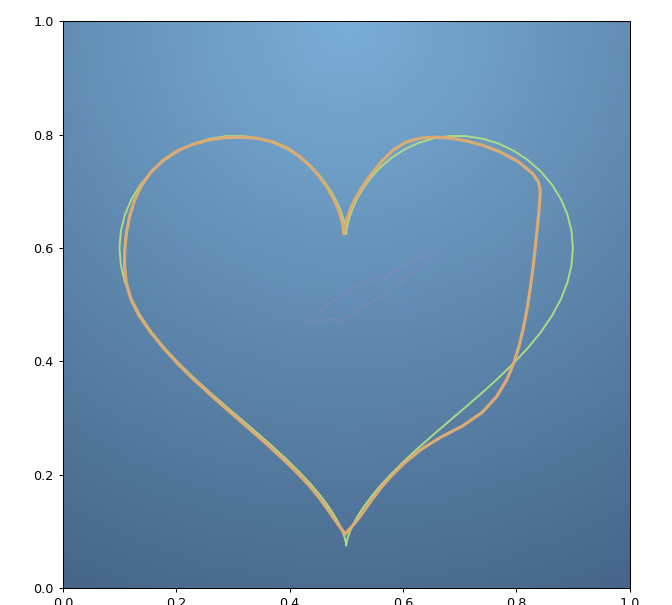








**After 100 iterations**



The green heart is the actual, the orange is the predited.

After 1000 iterations

