1.)Forward and Backward Propagation

Forwards Propagation -

Our aim is to make predictions on the basis of given data and forward propagation does that. We give data to our neural network, then it assigns weights and biases to it and passes it to an activation function. Output of previous layer is used as input of next layer. This assignment of weights and biases and passing to activation function repeats in different layers and finally neural network outputs a prediction.

Backward Propagation –

Backward Propagation is our method of trying to minimize error in our prediction. We calculate cost from correct output and our output, then we use cost function to calculate derivatives of various parameters of our neural network. Next we use these derivatives to update parameters in such a way that cost is minimized.

2.)

$Z_1 = W_1@X + B_1$

$A_1 = g_1(Z_1)$

$Z_2 = W_2@A_1 + B_2$

$A_2 = g_2(Z_2)$

Where X is a vector with entries x[i] where I is from 0 to 3. g_1 and g_2 are activation function for first and second layer respectively. W_1 and B_1 are weights and biases for first layer and W_2 and B_2 are weights and biases for second layer.
@ is the python notation of matrix multiplication.
In general

$Z_i = W_i@A_{i-1} + B_i$

$A_i = g_i(Z_i)$

3.)

a.) Sigmoid

Function - $\frac{1}{1+e^{-x}}$

Derivative - $\frac{1}{1+e^{-x}} * (1 - \frac{1}{1+e^{-x}})$

b.) ReLU

Function – max(0,x)

Derivative – If x $\geq$ 0 then 1

Else if x < 0 then 0

c.) Leaky ReLU

Function – max(α*x, x), where α is a small number(generally ~0.01 or smaller)

Derivative – If x $\geq$ 0 then 1

Else if x < 0 then α

d.) Tanh

Function – $(e^x - e^{-x})/(e^x + e^{-x})$

Derivative –

1 – $((e^x - e^{-x})/(e^x + e^{-x}))$^2

e.) Softmax

Function – e^x_i/($\sum e$^x_i)

Derivative – (e^x_i/($\sum e$^x_i))*(1 - e^x_i/($\sum e$^x_i))