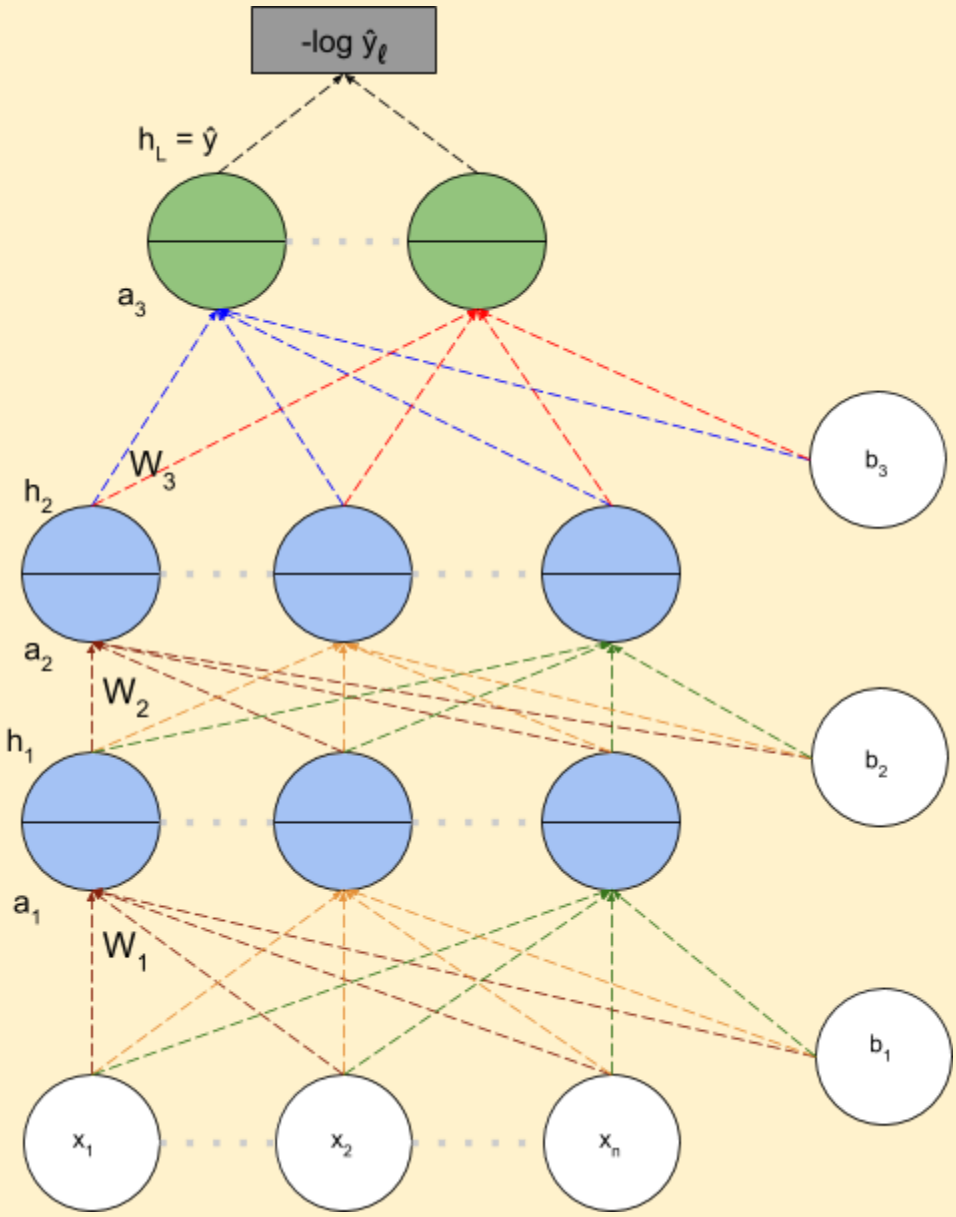
PadhAI: Backpropagation - the full version

One Fourth Labs

Intuition behind backpropagation

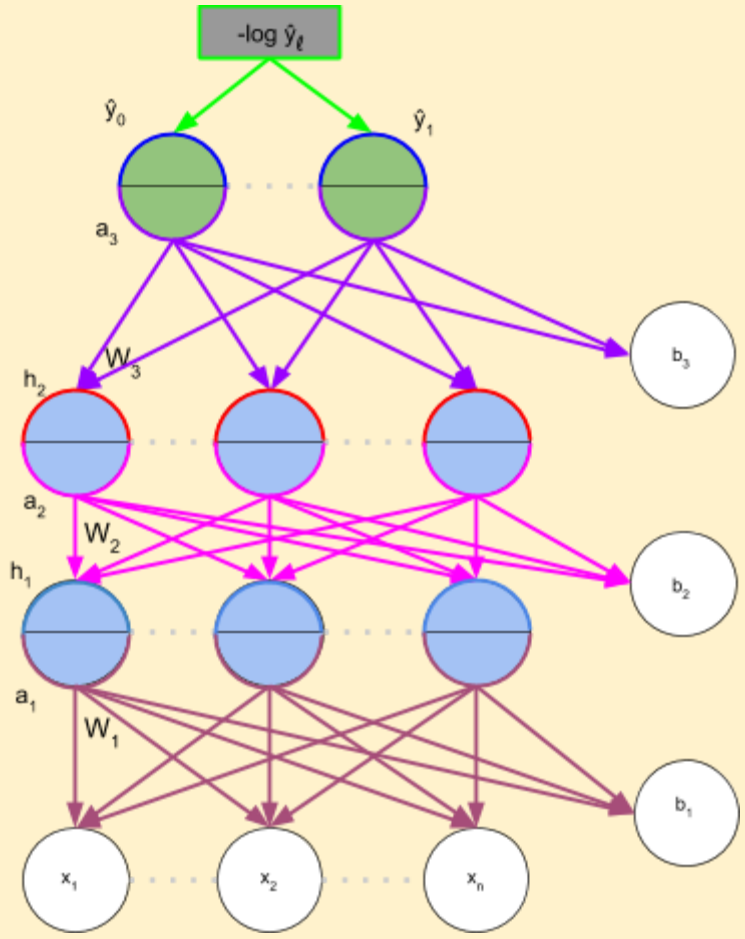
Let us look at an intuitive explanation of backpropagation

1. Let's have a quick calculus recap
   a. $f(x) = x^2$, $Derivative: \frac{df(x)}{dx}$
   b. $f(x) = x^2 + y^2$, $Partial\ Derivatives: \frac{\partial f(x)}{\partial x}\ and\ \frac{\partial f(x)}{\partial y}$ a
   c. $f(\theta) = [x, y]$, $Gradient\ \nabla_\theta = [\frac{\partial f(x)}{\partial x}, \frac{\partial f(x)}{\partial y}]$
2. Consider the following sample network



3. Now, let us assume our network has undergone some training, so we have a Loss value in hand.
4. W_1, W_2, W_3, b_1, b_2, b_3 have been updated and reflect accordingly in the Loss function

5. Now, we want to scrutinise how each parameter is responsible for the loss. So we move backwards from the Loss in a step-by-step manner



6. Stepwise calculation (while personifying the neurons and layers 😄)
   a. Step 1: The loss talks to the output layer, saying "You better take responsibility for the poor output!"
   b. Step 2: The output activation layer says, "Hey, I'm simply applying the softmax function to the input given to me by the output preactivation layer"
   c. Step 3: The output preactivation layer says, "I take responsibility for my part, but I am only as good as the hidden layer and the weights below me." After all $f(x) = \hat{y} = O(W_L h_{L-1} + b_L)$
   d. Step 4: The parameters W_3 and b_3 say "It is our mistake, **please update our values**"
   e. Step 5: However, the second hidden activation layer says "Hey, I'm simply applying the sigmoid function to the input given to me by the second hidden preactivation layer"
   f. Step 6: The second hidden preactivation layer says, "I am only as good as the hidden layer and weights below me"
   g. Step 7: The parameters W_2 and b_3 say "It is our mistake, **please update our values**"
   h. Step 8: However, the first hidden activation layer says "Hey, I'm simply applying the sigmoid function to the input given to me by the first hidden preactivation layer"
   i. Step 9: The first hidden preactivation layer says, "I am only as good as the weights below me, we cannot blame the input layer."
   j. Step 10: The last set of parameters W_1 and b_1 say "It is our mistake, please update us"

7. Thus, to arrive at the derivative of the Loss function w.r.t any of the weights, we must proceed downwards from the top. We cannot simply calculate it without knowing the preceding values.
8. Our roadmap for the rest of the module
   a. To calculate the desired gradient, we need to compute
   b. Gradient w.r.t output units
   c. Gradient w.r.t hidden units
   d. Gradient w.r.t weights and biases
   e.

$$\frac{\partial L(\theta)}{\partial W_{111}} = \frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_3} \frac{\partial a_3}{\partial h_2} \frac{\partial h_2}{\partial a_2} \frac{\partial a_2}{\partial h_1} \frac{\partial h_1}{\partial a_1} \frac{\partial a_3}{\partial W_{111}}$$

Talk to the weight directly	Talk to the output layer	Talk to the previous hidden layer	Talk to the previous hidden layer	Talk to the weights
		works for any number of output layers		

   f. Our aim is to do these calculations for any of the possible weights using notation i, j, k instead of numbers
   g. For the rest of this exercise, our focus is on *Cross Entropy loss* and *Softmax* output.
9. To reduce the tediousness of applying the chain rule each time to get the desired gradient, we will look to re-use common values and pathways to more efficiently calculate any gradient.