PadhAI: Backpropagation - the light math version

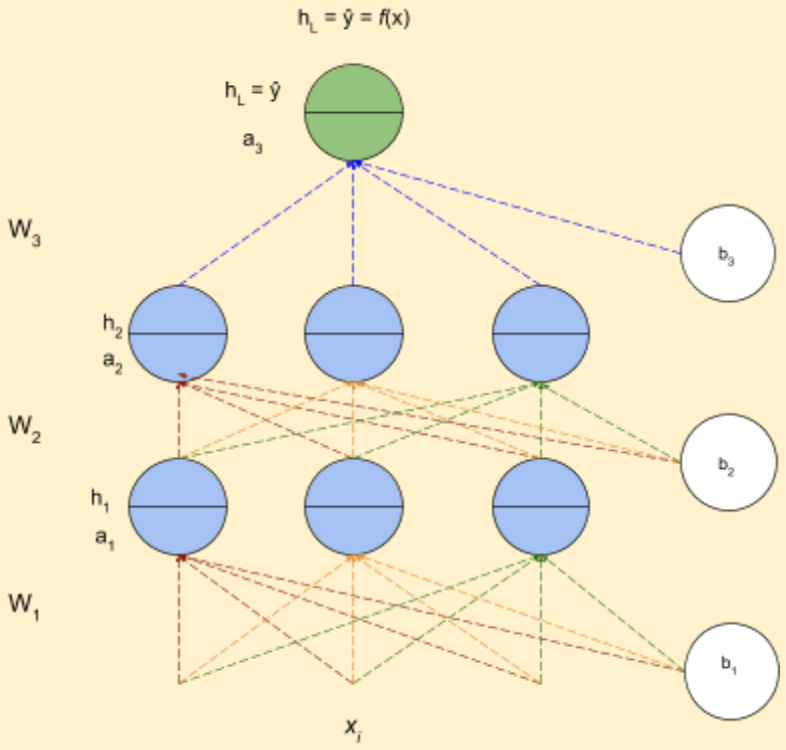
One Fourth Labs

Backpropagation (Light Math)

Setting the context

Can we use the same learning algorithm as before?

1. Here is the learning algorithm as discussed in the previous chapter, the no-math version
2. Consider the Neural Network with the following configuration



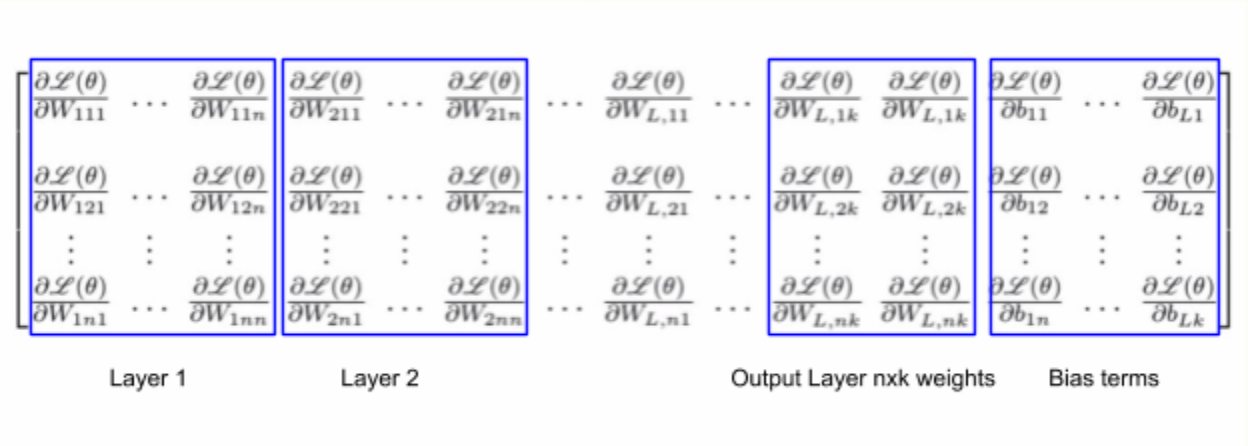
3. The algorithm
   a. **Initialise:** $w_{111}, w_{112}, \ldots w_{313}, b_1, b_2, b_3$ randomly
   b. **Iterate over data**
      i. Compute $\hat{y}$
      ii. Compute L(w,b) Cross-entropy loss function
      iii. $w_{111} = w_{111} - \eta \Delta w_{111}$
      iv. $w_{112} = w_{112} - \eta \Delta w_{112}$
      ...
      v. $w_{313} = w_{111} - \eta \Delta w_{313}$
      vi. $b_i = b_i + \eta \Delta b_i$
      vii. Pytorch/Tensorflow have functions to compute $\frac{\delta l}{\delta w}$ *and* $\frac{\delta l}{\delta b}$
   c. **Till satisfied**
      i. Number of epochs is reached (ie 1000 passes/epochs)
      ii. Continue till Loss < ε (some defined value)

PadhAI: Backpropagation - the light math version

One Fourth Labs

---

4. In this section, we will be looking at the light-math version, where we will be computing the derivatives
5. Derivatives for all layers from 1 to L



6. Once we know the gradients, we can use them in the Gradient Descent algorithm to compute the weights of the network