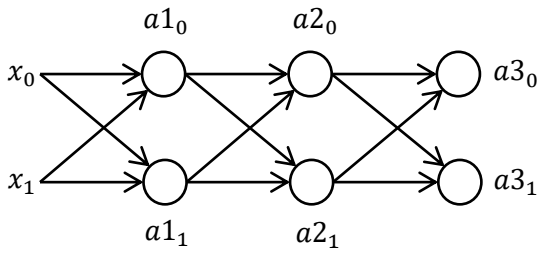
Topic 1: The backpropagation algorithm

In this topic, we will see how to manually compute the gradients of a feedforward neural network with two hidden layers with two neural units in each layer, like this:



We will use nothing but NumPy in order to train this neural network to output the same thing it receives as input.

1. Perturbation based

https://github.com/mtanti/deeplearningtutorial/blob/master/tensorflow_v1/numpy/01_-_Backpropagation_algorithm/01_-_Perturbation_based.py

The simplest way to find the gradient of a function with respect to a variable is to approximate it by increasing or decreasing the variable by a tiny amount and checking how the function's output changes as a result. This is called a perturbation and it calculated using the following equation:

$$\frac{\partial}{\partial x} f(x) \approx \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon}$$

where ε (epsilon) is a very tiny positive number. As ε gets closer to zero, the equation comes closer to the true gradient.

We apply this to every individual number in the weight and bias tensors (instead of x) and then use the approximate gradient in the gradient descent algorithm.

2. Full gradient equations

https://github.com/mtanti/deeplearningtutorial/blob/master/tensorflow_v1/numpy/01_-_Backpropagation_algorithm/02_-_Full_gradient_equations.py

Approximating gradients is easy but also very slow and requires a lot of code to train a tiny neural network as you need to handle each individual number in the weights and bias tensors separately. You also need to find a suitable epsilon which is pretty much another learning rate hyperparameter. The first step to solving these problems is to use the actual gradient equations to compute the gradients. The code shows the full derivation of every gradient in the neural network from error function to parameter. The derivatives require access to intermediate values that are computed whilst calculating the output of the neural network from the inputs. We can reduce some code by first making a forward pass (calculate the

outputs from the inputs and save the intermediate values in separate variables) and then making a backward pass (using intermediate values obtained in the forward pass to calculate the gradients).

3. The backpropagation algorithm

https://github.com/mtanti/deeplearningtutorial/blob/master/tensorflow_v1/numpy/01_-_Backpropagation_algorithm/03_-_The_backpropagation_algorithm.py

This is the real deal, the backpropagation algorithm. This algorithm is an efficient way to compute gradients for a neural network without recomputing intermediate any intermediate values that were already computed for other gradients. It works by exploiting a recurrent pattern found in the gradient equations in the previous program. It also works using matrix notation to compute the gradient for entire tensors at once given the previous layers' intermediate values which makes it very fast and very terse.