We initialize the Feed-Forward algorithm by computing

$$\begin{aligned} z_j^1 &= w_{jk}^1 X_k + b_j^1 \\ a_j^1 &= \sigma(z_j^1) \end{aligned} \tag{1}$$

where we adopt the Einstein summation convention by summing over repeated indices. We then compute

$$\begin{aligned} z_j^l &= w_{jk}^l a_k^{l-1} + b_j^l \\ a_j^l &= \sigma(z_j^l) \end{aligned} \tag{2}$$

for hidden layers $l = 2, \ldots, L$. Then, for the ouput layer we compute

$$\begin{aligned} z_j^L &= w_{jk}^L a_k^{L-1} + b_j^L \\ a_j^L &= \tilde{\sigma}(z_j^L) \end{aligned} \tag{3}$$

where a_j^L is the predicted response, and $\tilde{\sigma}$ is the activation function for the output layer, which may differ from the activation function of the hidden layers.

We then compute the error of the output as

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \odot \tilde{\sigma}'(z_j^L) \tag{4}$$

and backpropogate the error like

$$\delta_j^l = (\delta_k^{l+1} w_{kj}^{l+1}) \odot \sigma'(z_j^l) \tag{5}$$

for all $l = L-1, \ldots, 2$ where we make particular note that $w_{kj} = (w_{jk})^T$.

We may then easily compute the gradients of the cost function wrt. the weights & biases as

$$\begin{aligned} \frac{\partial C}{\partial w_{jk}^l} &= \delta_j^l a_k^{l-1} \\ \frac{\partial C}{\partial b_j^l} &= \delta_j^l \end{aligned} \tag{6}$$

which are then used to update the weights and biases via gradient descent.

Backpropogation with minibatches

We have input and output matrices $X \in [M \times P], Y \in [M \times Q]$, with elements X_{mp}, Y_{mq}. Since we wish to perform the Backpropogation effectively using the highly optimized linear algebra libraries available in Numpy, we must therefore slightly alter the way in which we perform the backpropogation. Since our input vectors X now comes in the form

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_P \end{bmatrix} \rightarrow X = \begin{bmatrix} X_{11} & \dots & X_{1P} \\ & \vdots & \\ X_{M1} & \dots & X_{MP} \end{bmatrix} \tag{7}$$

we let $z_j^l \rightarrow z_{mj}^l$ and $a_j^l \rightarrow a_{mj}^l$ such that they are in accordance with X_{mp}, Y_{mq}.

In order to adhere to this new form, we must transpose Eqn. 1 which yields

$$w_{jk} X_K \rightarrow (w_{jk} X_k)^T = X_k^T w_{kj} \tag{8}$$

Thus, we may write the initial step as

$$\begin{aligned} z_{mj}^1 &= X_{mk} w_{kj} + \left(b_j^1\right)^T \\ a_{mj}^1 &= \sigma\left(z_{mj}^1\right) \end{aligned} \tag{9}$$

where the transposed bias is implicitly added element-wise to each row in the resultant matrix. In a similar fashion, we feed forward for $l = 2, \dots L$

$$\begin{aligned} z_{mj}^l &= a_{mk}^{l-1} w_{kj} + \left(b_j^L\right)^T \\ a_{mj}^l &= \sigma(z_{mj}^L) \end{aligned} \tag{10}$$

and for the output layer

$$\begin{aligned} z_{mj}^L &= a_{mk}^{L-1} w_{kj} + \left(b_j^L\right)^T \\ a_{mj}^L &= \tilde{\sigma}\left(z_{mj}^L\right) \end{aligned} \tag{11}$$

$$\delta_j^l = (w_{jk})^T \delta_j^{l+1} \tag{12}$$

$$\delta_{mj}^l = \delta_{mk}^{l+1} w_{kj}^{l+1} \tag{13}$$