N-layer perceptron

May 2025

1 Introduction

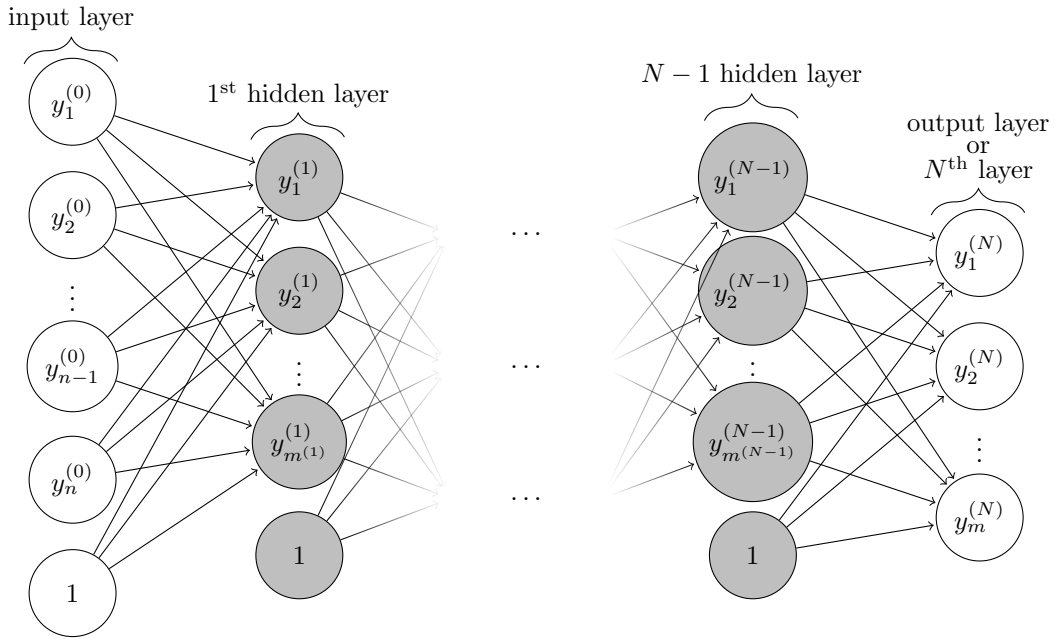

Figure 1: Network graph of a N-layer perceptron with n input units and m output units. The N^{th} hidden layer contains $m^{(N)}$ hidden units. The nodes with a 1 represent the bias nodes.

The perceptron is the simplest deep learning neural network. It is a multi-class classificator that takes an n-dimensional input to return an estimation of the probability that the input belongs to any of the m classes it has been trained to detect. Its simplest form is the single layered perceptron, only capable of separating linearly separable data. By adding new layers to the perceptron we can separate more complex data. At three layers, the perceptron should be able to separate any kind of data.

A layer in a perceptron is akin to a vector, with n components or nodes. The first layer is always the input layer, which is the input we hope to classify. Every successive layer in a perceptron introduces a linear transformation of the previous layer composite with an activation function. The activation function is a non-linear function that takes the value of each node and squishes between 0 and 1 or -1 and 1, or other similar intervals, depending on the function used. Its purpose is to add non-linearity between layers, since any composition of linear transformations is also linear and thus unable to separate non-linear data.

Some examples of this activation functions are the softmax function and the hyperbolic tangent.

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad , \quad tanh(x_i) = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{-x_i}}$$

The softmax has the interesting property of returning probabilities. For any input vector, the sum of its components after being transformed are 1, so they can be thought of as probabilities.

Our network is an N-layer perceptron, which means it has an input layer plus N transformation layers. Each layer is named $y^{(n)}$ with $n = 0$ for the input layer and $n = 1, \ldots, N$ for the rest. Same for the dimension of each layer, which is given by $m(n)$. In our case we have an input layer of n nodes and an output layer of m nodes, so $m(0) = n, \quad m(N) = m$. Additionally, each layer from 1 to N has its own weights, the matrix $W^{(n)}$ that serves as the linear transformation between layers and activation function, $a^{(n)}$. For the Nth layer, the activation function is a softmax since we want our final output to be probabilities.

Mathematically, the steps to go from one layer to the other are as follows.

$$z_i^{(n)} = \sum_{j=1}^{m(n-1)} W_{ij}^{(n)} y_j^{(n-1)} \tag{1}$$

$$y_i^{(n)} = a^{(n)}[z_i^{(n)}] \qquad\qquad n = 1, \ldots, N \tag{2}$$

2 Gradient descent and Backpropagation

Gradient descent is a method of training the network. It requires the correct label for each instance of the training data and the definition of a loss function $\mathcal{L}$. The loss function is an estimation of how bad the prediction was, and our goal is to minimize it, so each iteration of backpropagation aims to update the weights in a way that minimizes the loss. The method is using the negative direction of the gradient, since it points in the direction of maximum growth. For an iteration t and the next, t+1

$$W^{t+1} = W^t - \eta \frac{\partial \mathcal{L}}{\partial W} \tag{3}$$

Where η is a learning constant, a number chosen appropriately to fine-tune how fast the weights change and $\frac{\partial \mathcal{L}}{\partial W}$ is the gradient of the loss with respect to the weights.

The loss function chosen for this study is the Kullback–Leibler divergence, a measure of how different a model probability distribution Q(X) is from a true distribution P(X).

$$D_{K-L} = \sum_{x \in X} P(X) \ln \frac{P(X)}{Q(X)} \tag{4}$$

The model distribution in our case is the output of our network, $y_\gamma^{(N)}$, which we will rename p_γ for simplicity. The true distribution is the label, which can be understood as a vector that is 0 in every component except the one corresponding to the correct label l_γ. However the loss function is not calculated over a single instance of the training data but a subset of it, which we will denote adding a sum over α. Now our loss looks like

$$\mathcal{L} = \sum_\alpha \sum_\gamma l_\gamma^\alpha \ln \frac{l_\gamma^\alpha}{p_\gamma^\alpha} \tag{5}$$

Where α is a sum over a subset of training data and γ over the m possible classes. And since $l_\gamma^\alpha = 1$ if γ is the correct class and 0 otherwise

$$\mathcal{L} = \sum_\alpha - \ln p_{\hat{\gamma}(\alpha)}^\alpha = \sum_\alpha L^\alpha \tag{6}$$

Where $\hat{\gamma}(\alpha)$ is the index of the correct class for instance α. This loss is also known as cross-entropy loss.

The derivatives are calculated with a method called backpropagation, which uses the chain rule to calculate the derivatives, but works starting from the N^{th} instead of the input layer, hence the name.

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N)}} = \sum_{\alpha}\sum_{\beta}\sum_{i} \frac{\partial L^{\alpha}}{\partial p_{\beta}} \frac{\partial p_{\beta}}{\partial z_i^{(N)}} \frac{\partial z_i^{(N)}}{\partial W_{\mu\nu}^{(N)}} \tag{7}$$

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-1)}} = \sum_{\alpha}\sum_{\beta}\sum_{i}\sum_{j}\sum_{k} \frac{\partial L^{\alpha}}{\partial p_{\beta}} \frac{\partial p_{\beta}}{\partial z_i^{(N)}} \frac{\partial z_i^{(N)}}{\partial y_j^{(N-1)}} \frac{\partial y_j^{(N-1)}}{\partial z_k^{(N-1)}} \frac{\partial z_k^{(N-1)}}{\partial W_{\mu\nu}^{(N-1)}} \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-2)}} = \sum_{\alpha}\sum_{\beta}\sum_{i}\sum_{j}\sum_{k}\sum_{l}\sum_{q} \frac{\partial L^{\alpha}}{\partial p_{\beta}} \frac{\partial p_{\beta}}{\partial z_i^{(N)}} \frac{\partial z_i^{(N)}}{\partial y_j^{(N-1)}} \frac{\partial y_j^{(N-1)}}{\partial z_k^{(N-1)}} \frac{\partial z_k^{(N-1)}}{\partial y_l^{(N-2)}} \frac{\partial y_l^{(N-2)}}{\partial z_q^{(N-2)}} \frac{\partial z_q^{(N-2)}}{\partial W_{\mu\nu}^{(N-2)}} \tag{9}$$

$$\dots$$

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(2)}} = \sum_{\alpha}\sum_{\beta}\sum_{i}\sum_{j}\cdots\sum_{r}\sum_{s} \frac{\partial L^{\alpha}}{\partial p_{\beta}} \frac{\partial p_{\beta}}{\partial z_i^{(N)}} \frac{\partial z_i^{(N)}}{\partial y_j^{(N-1)}} \cdots \frac{\partial y_r^{(3)}}{\partial z_s^{(3)}} \frac{\partial z_s^{(2)}}{\partial W_{\mu\nu}^{(2)}} \tag{10}$$

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(1)}} = \sum_{\alpha}\sum_{\beta}\sum_{i}\sum_{j}\cdots\sum_{s}\sum_{t} \frac{\partial L^{\alpha}}{\partial p_{\beta}} \frac{\partial p_{\beta}}{\partial z_i^{(N)}} \frac{\partial z_i^{(N)}}{\partial y_j^{(N-1)}} \cdots \frac{\partial y_s^{(2)}}{\partial z_t^{(2)}} \frac{\partial z_t^{(1)}}{\partial W_{\mu\nu}^{(1)}} \tag{11}$$

Fortunately most of these sums disappear because the derivatives are 0. In the (N) case:

$$z_i^{(N)} = \sum_j W_{ij}^{(N)} y_j^{(N-1)} \quad , \quad \frac{\partial z_i^{(N)}}{\partial W_{\mu\nu}^{(N)}} = \delta_{i\mu}\ y_{\nu}^{(N-1)}$$

$$\frac{\partial p_{\beta}}{\partial z_i^{(N)}} = \frac{\partial}{\partial z_i^{(N)}} \left(\frac{e^{z_{\beta}^{(N)}}}{\sum_{j=1}^{m} e^{z_j^{(N)}}} \right) = \frac{\delta_{\beta i} e^{z_{\beta}^{(N)}} \sum_{j=1}^{m} e^{z_j^{(N)}} - e^{z_{\beta}^{(N)}} e^{z_i^{(N)}}}{(\sum_{j=1}^{m} e^{z_j^{(N)}})^2} = \delta_{\beta i}\ p_{\beta} - p_{\beta} p_i$$

$$\frac{\partial L^{\alpha}}{\partial p_{\beta}} = \frac{\partial}{\partial p_{\beta}} \left(-\ln p_{\hat{\gamma}(\alpha)}^{\alpha} \right) = -\delta_{\beta\hat{\gamma}(\alpha)} \frac{1}{p_{\hat{\gamma}(\alpha)}}$$

$$\delta_{ij} = 1 \quad i = j$$

$$\delta_{ij} = 0 \quad i \neq j$$

We can see that Kronecker deltas appear because for an output z_i only weights with components where the first index is i intervene, $\frac{\partial z_i}{\partial z_j} = \delta_{ij}$ and $\frac{\partial p_i}{\partial p_j} = \frac{\partial y_i}{\partial y_j} = \delta_{ij}$. Using these results, now (7) looks like

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N)}} = \sum_{\alpha}\sum_{\beta}\sum_{i} -\delta_{\beta\hat{\gamma}(\alpha)} \frac{1}{p_{\hat{\gamma}(\alpha)}} (\delta_{\beta i}\ p_{\beta} - p_{\beta} p_i) \delta_{i\mu}\ y_{\nu}^{(N-1)} = \sum_{\alpha}\sum_{\beta} -\delta_{\beta\hat{\gamma}(\alpha)} \frac{1}{p_{\hat{\gamma}(\alpha)}} (\delta_{\beta\mu}\ p_{\beta} - p_{\beta} p_{\mu}) y_{\nu}^{(N-1)} \tag{12}$$

$$= \sum_{\alpha} -\frac{1}{p_{\hat{\gamma}(\alpha)}} (\delta_{\hat{\gamma}(\alpha)\mu}\ p_{\hat{\gamma}(\alpha)} - p_{\hat{\gamma}(\alpha)} p_{\mu}) y_{\nu}^{(N-1)} = \sum_{\alpha} -(\delta_{\hat{\gamma}(\alpha)\mu} - p_{\mu}) y_{\nu}^{(N-1)}$$

Repeating the same process for $(N-1)$, the new derivatives are

$$\frac{\partial z_k^{(N-1)}}{\partial W_{\mu\nu}^{(N-1)}} = \delta_{k\mu}\ y_{\nu}^{(N-2)}$$

$$\frac{\partial y_j^{(N-1)}}{\partial z_k^{(N-1)}} = \frac{\partial}{\partial z_k^{(N-1)}} \left(a^{(N-1)}[z_j^{(N-1)}] \right) = \delta_{jk}\ a'^{(N-1)}[z_j^{(N-1)}]$$

Where $a'^{(N-1)}[z_j^{(N-1)}]$ is the derivative of the j component of the activation function used in layer $(N-1)$. In the case of

$$a[z_i] = \tanh(z_i) \quad , \quad a'[z_i] = 1 - (\tanh(z_i))^2 = 1 - y_i^2$$

$$\frac{\partial z_i^{(N)}}{\partial y_j^{(N-1)}} = \frac{\partial}{\partial y_j^{(N-1)}} \left(\sum_n W_{in}^{(N)} \; y_n^{(N-1)} \right) = W_{ij}^{(N)}$$

Again, some Kronecker deltas appear, so using these results, now (8) looks like

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-1)}} &= \sum_\alpha \sum_\beta \sum_i \sum_j \sum_k -\delta_{\beta\hat{\gamma}(\alpha)} \frac{1}{p_{\hat{\gamma}(\alpha)}} (\delta_{\beta i} \; p_\beta - p_\beta p_i) W_{ij}^{(N)} \delta_{jk} \; a'^{(N-1)}[z_j^{(N-1)}] \delta_{k\mu} \; y_\nu^{(N-2)} \qquad (13) \\
&= \sum_\alpha \sum_\beta \sum_i \sum_j -\delta_{\beta\hat{\gamma}(\alpha)} \frac{1}{p_{\hat{\gamma}(\alpha)}} (\delta_{\beta i} \; p_\beta - p_\beta p_i) W_{ij}^{(N)} \delta_{j\mu} \; a'^{(N-1)}[z_j^{(N-1)}] y_\nu^{(N-2)} \\
&= \sum_\alpha \sum_\beta \sum_i -\delta_{\beta\hat{\gamma}(\alpha)} \frac{1}{p_{\hat{\gamma}(\alpha)}} (\delta_{\beta i} \; p_\beta - p_\beta p_i) W_{i\mu}^{(N)} a'^{(N-1)}[z_\mu^{(N-1)}] y_\nu^{(N-2)} \\
&= \sum_\alpha \sum_i -(\delta_{\hat{\gamma}(\alpha) i} \; - p_i) W_{i\mu}^{(N)} a'^{(N-1)}[z_\mu^{(N-1)}] y_\nu^{(N-2)}
\end{aligned}$$

In all derivatives, the first two terms can be simplified to $-(\delta_{\hat{\gamma}(\alpha)i} \; - p_i)$ and remove the sum over β.

The following expression is the dot product between the i component vector $-(\delta_{\hat{\gamma}(\alpha)} \; - p_i)$ and the other i component vector $(W_{i\mu}^{(N)})^T$ where μ is given by which element of the matrix we are trying to optimize.

$$\sum_i -(\delta_{\hat{\gamma}(\alpha)i} \; - p_i) W_{i\mu}^{(N)}$$

For the case of $(N-2)$ the new derivatives are

$$\frac{\partial z_k^{(N-1)}}{\partial y_l^{(N-2)}} = W_{kl}^{(N-1)}$$

$$\frac{\partial y_l^{(N-2)}}{\partial z_q^{(N-2)}} = \delta_{lq} a'^{(N-2)}[z_l^{(N-2)}]$$

$$\frac{\partial z_q^{(N-2)}}{\partial W_{\mu\nu}^{(N-2)}} = \delta_{q\mu} \; y_\nu^{(N-3)}$$

We can see that these new derivatives have the same structure as the previous ones, so now (9) would look like

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-2)}} &= \sum_\alpha \sum_\beta \sum_{i,j,k,l,q} \frac{\partial L^\alpha}{\partial p_\beta} \frac{\partial p_\beta}{\partial z_i^{(N)}} W_{ij}^{(N)} \delta_{jk} \; a'^{(N-1)}[z_j^{(N-1)}] W_{kl}^{(N-1)} \delta_{lq} a'^{(N-2)}[z_l^{(N-2)}] \delta_{q\mu} \; y_\nu^{(N-3)} \qquad (14) \\
&= \sum_\alpha \sum_{i,j,k,l,q} -(\delta_{\hat{\gamma}(\alpha)} \; - p_i) W_{ij}^{(N)} \delta_{jk} \; a'^{(N-1)}[z_j^{(N-1)}] W_{kl}^{(N-1)} \delta_{lq} a'^{(N-2)}[z_l^{(N-2)}] \delta_{q\mu} \; y_\nu^{(N-3)} \\
&= \sum_\alpha \sum_i \sum_j \sum_k \sum_l -(\delta_{\hat{\gamma}(\alpha)} \; - p_i) W_{ij}^{(N)} \delta_{jk} \; a'^{(N-1)}[z_j^{(N-1)}] W_{kl}^{(N-1)} \delta_{l\mu} a'^{(N-2)}[z_l^{(N-2)}] y_\nu^{(N-3)} \\
&= \sum_\alpha \sum_i \sum_j \sum_k -(\delta_{\hat{\gamma}(\alpha)} \; - p_i) W_{ij}^{(N)} \delta_{jk} \; a'^{(N-1)}[z_j^{(N-1)}] W_{k\mu}^{(N-1)} a'^{(N-2)}[z_\mu^{(N-2)}] y_\nu^{(N-3)} \\
&= \sum_\alpha \sum_i \sum_k -(\delta_{\hat{\gamma}(\alpha)} \; - p_i) W_{ik}^{(N)} a'^{(N-1)}[z_k^{(N-1)}] W_{k\mu}^{(N-1)} a'^{(N-2)}[z_\mu^{(N-2)}] y_\nu^{(N-3)}
\end{aligned}$$

Doing a small change in notation we reach a more iterative formula

$$p_\beta \longrightarrow y_\beta^{(N)}$$

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-2)}} = \sum_\alpha \sum_\beta^{m(N)} \sum_i^{m(N)} \sum_j^{m(N-1)} \sum_k^{m(N-1)} \sum_l^{m(N-2)} \sum_q^{m(N-2)} \frac{\partial L^\alpha}{\partial y_\beta^{(N)}} \frac{\partial y_\beta^{(N)}}{\partial z_i^{(N)}} \frac{\partial z_i^{(N)}}{\partial y_j^{(N-1)}} \frac{\partial y_j^{(N-1)}}{\partial z_k^{(N-1)}} \frac{\partial z_k^{(N-1)}}{\partial y_l^{(N-2)}} \frac{\partial y_l^{(N-2)}}{\partial z_q^{(N-2)}} \frac{\partial z_q^{(N-2)}}{\partial W_{\mu\nu}^{(N-2)}} \tag{15}$$

And renaming the indices to

$$\beta \to i_N \quad i \to j_N \quad j \to i_{N-1} \quad k \to j_{N-1} \quad l \to i_{N-1} \dots$$

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-n)}} = \sum_\alpha \sum_{i_N} \sum_{j_N} \sum_{i_{N-1}} \sum_{j_{N-1}} \cdots \sum_{i_{N-2}} \sum_{j_{N-2}} \frac{\partial L^\alpha}{\partial y_{i_N}^{(N)}} \frac{\partial y_{i_N}^{(N)}}{\partial z_{j_N}^{(N)}} \frac{\partial z_{j_N}^{(N)}}{\partial y_{i_{N-1}}^{(N-1)}} \frac{\partial y_{i_{N-1}}^{(N-1)}}{\partial z_{j_{N-1}}^{(N-1)}} \cdots \frac{\partial z_{j_{N-n+1}}^{(N-n+1)}}{\partial y_{i_{N-n}}^{(N-n)}} \frac{\partial y_{i_{N-n}}^{(N-n)}}{\partial z_{j_{N-n}}^{(N-n)}} \frac{\partial z_{j_{N-n}}^{(N-n)}}{\partial W_{\mu\nu}^{(N-n)}} \tag{16}$$

For $n = 1, ..., N-1$ Now with this new notation, the results obtained look like

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N)}} = \sum_\alpha -(\delta_{\hat{\gamma}(\alpha)\mu} - p_\mu) y_\nu^{(N-1)} \tag{17}$$

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-1)}} = \sum_\alpha \sum_{j_N} -(\delta_{\hat{\gamma}(\alpha)j_N} - y_{j_N}^{(N)}) W_{j_N\mu}^{(N)} a'^{(N-1)}[z_\mu^{(N-1)}] y_\nu^{(N-2)}$$

We are going to define a new quantity that will be used later on

$$A_\mu^{(N)}(\alpha) = \sum_{j_N} -(\delta_{\hat{\gamma}(\alpha)j_N} - y_{j_N}^{(N)}) W_{j_N\mu}^{(N)}$$

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-1)}} = \sum_\alpha A_\mu^{(N)}(\alpha)\ a'^{(N-1)}[z_\mu^{(N-1)}] y_\nu^{(N-2)} \tag{18}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-2)}} &= \sum_\alpha \sum_{j_N} \sum_{j_{N-1}} -(\delta_{\hat{\gamma}(\alpha)} - y_{j_N}^{(N)}) W_{j_N j_{N-1}}^{(N)} a'^{(N-1)}[z_{j_{N-1}}^{(N-1)}] W_{j_{N-1}\mu}^{(N-1)} a'^{(N-2)}[z_\mu^{(N-2)}] y_\nu^{(N-3)} \\ &= \sum_\alpha \sum_{j_{N-1}} A_{j_{N-1}}^{(N)}(\alpha) a'^{(N-1)}[z_{j_{N-1}}^{(N-1)}] W_{j_{N-1}\mu}^{(N-1)} a'^{(N-2)}[z_\mu^{(N-2)}] y_\nu^{(N-3)}\end{aligned}$$

We can see we obtained a similar structure to (18), so

$$A_\mu^{(N-1)}(\alpha) = \sum_{j_{N-1}} A_{j_{N-1}}^{(N)}(\alpha) a'^{(N-1)}[z_{j_{N-1}}^{(N-1)}] W_{j_{N-1}\mu}^{(N-1)}$$

And now $(N-2)$ looks like

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-2)}} = \sum_\alpha A_\mu^{(N-1)}(\alpha) a'^{(N-2)}[z_\mu^{(N-2)}] y_\nu^{(N-3)} \tag{19}$$

And so we can generalize to obtain a general solution to the derivatives for the weights at each layer

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-n)}} = \sum_\alpha A_\mu^{(N-n+1)}(\alpha) a'^{(N-n)}[z_\mu^{(N-n)}] y_\nu^{(N-n-1)} \tag{20}$$

Where $n = 1, \dots, N-1$ And the case for $n = 0$ is an exception

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N)}} = \sum_\alpha -(\delta_{\hat{\gamma}(\alpha)\mu} - p_\mu) y_\nu^{(N-1)}$$

The coefficients $A(\alpha)$ can be obtained recursively

$$A_{\mu}^{(N)}(\alpha) = \sum_{j_N} -(\delta_{\hat{\gamma}(\alpha)j_N} - y_{j_N}^{(N)})W_{j_N\mu}^{(N)} \tag{21}$$

$$A_{\mu}^{(N-n)}(\alpha) = \sum_{j_{N-n}} A_{j_{N-n}}^{(N-n+1)}(\alpha)a'^{(N-n)}[z_{j_{N-n}}^{(N-n)}]W_{j_{N-n}\mu}^{(N-n)} \tag{22}$$

For $n = 1, \ldots, N-1$ although $A^{(1)}(\alpha)$ is not needed since it would calculate the derivatives for $W^{(0)}$ which does not exist.

Finally, remember that the derivatives are used to modify the weights following

$$W^{t+1} = W^t - \eta\frac{\partial\mathcal{L}}{\partial W}$$

3 Introducing a bias term

When introducing a bias term, now (1) becomes

$$z_i^{(n)} = \sum_{j=1}^{m(n-1)} W_{ij}^{(n)}y_j^{(n-1)} + b_i^{(n)} \tag{23}$$

This has no effect on the previously calculated derivatives for the weights because the bias can be thought of as an additional column in the weights as long you add a 1 at the end of the input, since the result of that will be an additional term added for every component of the output, which is the bias.

$$\begin{pmatrix} W_{11} & W_{12} & b_1 \\ W_{21} & W_{22} & b_2 \\ W_{31} & W_{32} & b_3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = \begin{pmatrix} W_{11}x_1 + W_{12}x_2 + b_1 \\ W_{21}x_1 + W_{22}x_2 + b_2 \\ W_{31}x_1 + W_{32}x_2 + b_3 \end{pmatrix}$$

Still, we will proceed to calculate the derivative of the loss function with respect to the bias for completeness.

$$\frac{\partial\mathcal{L}}{\partial b_{\mu}^{(N)}} = \sum_{\alpha}\sum_{\beta}\sum_{i}\frac{\partial L^{\alpha}}{\partial p_{\beta}}\frac{\partial p_{\beta}}{\partial z_i^{(N)}}\frac{\partial z_i^{(N)}}{\partial b_{\mu}^{(N)}} \tag{24}$$

$$\frac{\partial\mathcal{L}}{\partial b_{\mu}^{(N-n)}} = \sum_{\alpha}\sum_{i_N}\sum_{j_N}\sum_{i_{N-1}}\sum_{j_{N-1}}\cdots\sum_{i_{N-2}}\sum_{j_{N-2}}\frac{\partial L^{\alpha}}{\partial y_{i_N}^{(N)}}\frac{\partial y_{i_N}^{(N)}}{\partial z_{j_N}^{(N)}}\frac{\partial z_{j_N}^{(N)}}{\partial y_{i_{N-1}}^{(N-1)}}\frac{\partial y_{i_{N-1}}^{(N-1)}}{\partial z_{j_{N-1}}^{(N-1)}}\cdots\frac{\partial z_{j_{N-n+1}}^{(N-n+1)}}{\partial y_{i_{N-n}}^{(N-n)}}\frac{\partial y_{i_{N-n}}^{(N-n)}}{\partial z_{j_{N-n}}^{(N-n)}}\frac{\partial z_{j_{N-n}}^{(N-n)}}{\partial b_{\mu}^{(N-n)}} \tag{25}$$

We can see the only changes are to the last term

$$\frac{\partial z_{j_{N-n}}^{(N-n)}}{\partial b_{\mu}^{(N-n)}} = \delta_{j_{N-n}\mu}$$

Which is almost identical to the result for the derivative with respect to the weights except for a

$$y_{\nu}^{(N-n-1)}$$

Which means the derivatives for the bias will have the same structure as the ones for the weights except for the last term, which will disappear.

$$\frac{\partial\mathcal{L}}{\partial b_{\mu}^{(N-n)}} = \sum_{\alpha}A_{\mu}^{(N-n+1)}(\alpha)a'^{(N-n)}[z_{\mu}^{(N-n)}] \tag{26}$$

Where the coefficients $A(\alpha)$ are the same, $n = 1, \ldots, N-1$ and the case for $n = 0$ is an exception

$$\frac{\partial \mathcal{L}}{\partial b_\mu^{(N)}} = \sum_\alpha -(\delta_{\hat{\gamma}(\alpha)\mu} - p_\mu) \tag{27}$$

Finally, the bias term is updated identically to the weights, following

$$b^{t+1} = b^t - \eta \frac{\partial \mathcal{L}}{\partial b}$$

4 Implmentation into code

For layer N,

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N)}} = \sum_\alpha -(\delta_{\hat{\gamma}(\alpha)\mu} - p_\mu) y_\nu^{(N-1)}$$

$$\frac{\partial \mathcal{L}}{\partial b_\mu^{(N)}} = \sum_\alpha -(\delta_{\hat{\gamma}(\alpha)\mu} - p_\mu)$$

So the only operation needed is a matrix multiplication between two vectors, that will output a matrix. The vectors in question are the values of the $(N-1)$ layer and the probability of every class, except the correct class which is one minus the probability. We will call this the probability of failure pf.
The final formula is

$$\text{WN} = \text{WN} - \text{eta} * \text{np.matmul(pf,yN1.T)}$$

$$\text{bN} = \text{bN} - \text{eta} * \text{pf}$$

Assuming the input vectors are arrays of shape (N,), we need to transpose the second vector to obtain a matrix. Finally we will calculate the $A(\alpha)$ coefficient for layer N.

$$A_\mu^{(N)}(\alpha) = \sum_{j_N} -(\delta_{\hat{\gamma}(\alpha)j_N} - y_{j_N}^{(N)}) W_{j_N\mu}^{(N)}$$

This is simply a matrix multiplication of the transposed matrix of weights for the Nth layer with the pf vector.

$$\text{AN} = \text{np.matmul(WN.T,pf)}$$

For layer N-1 onwards we also have to calculate the $A(\alpha)$ coefficient, now with a recursive formula

$$A_\mu^{(N-n)}(\alpha) = \sum_{j_{N-n}} A_{j_{N-n}}^{(N-n+1)}(\alpha) a'^{(N-n)}[z_{j_{N-n}}^{(N-n)}] W_{j_{N-n}\mu}^{(N-n)} \qquad n = 1, \ldots, N-1$$

It is also a matrix multiplication between a transposed matrix and a vector, but first we have to add an element-wise vector multiplication between the A coefficient vector of the previous layer (moving backwards) and the derivative of the activation function vector of the same layer.

$$\text{AN1} = \text{np.matmul(WN1.T,np.multiply(AN,a'N1))}$$

And the derivative is calculated by

$$\frac{\partial \mathcal{L}}{\partial W_{\mu\nu}^{(N-n)}} = \sum_\alpha A_\mu^{(N-n+1)}(\alpha) a'^{(N-n)}[z_\mu^{(N-n)}] y_\nu^{(N-n-1)} \qquad n = 1, \ldots, N-1$$

$$\frac{\partial \mathcal{L}}{\partial b_\mu^{(N-n)}} = \sum_\alpha A_\mu^{(N-n+1)}(\alpha) a'^{(N-n)}[z_\mu^{(N-n)}]$$

$$\text{WN1} = \text{WN1} - \text{eta} * \text{np.matmul(np.multiply(AN,a'N1),yN2.T)}$$

$$\text{bN1} = \text{bN1} - \text{eta} * \text{np.multiply(AN,a'N1)}$$