Okay, let us go through the whole process step by step in this example.

Forward propagation

Taking $sample_1$ ($x_1 = 0.04$,$x_2 = 0.42$, target=0) as the example.

1. **Step1**: Get the values of nodes after the activation operation f in the hidden layer.

   - The input value $N_i^{input}, i = 1, 2, 3$ of the node N_i on $sample_1$.

$$\begin{aligned} N_1^{input} &= x_1 w_{1,N_1} + x_2 w_{2,N_1} + b_{N_1 2} \\ &= 0.04 \times (-2.5) + 0.42 \times 0.6 + 1.6 \\ &= 1.752 \end{aligned} \tag{1}$$

$$\begin{aligned} N_2^{input} &= x_1 w_{1,N_2} + x_2 w_{2,N_2} + b_{N_1} \\ &= 0.04 \times (-1.5) + 0.42 \times 0.4 + 0.7 \\ &= 0.808 \end{aligned} \tag{2}$$

   - The output value $N_i^{output}, i = 1, 2$ of the node N_i on $sample_1$ with ReLU activation function $f(x) = max(0, x)$ where x is $N_i^{input}, i =$the number of nodes in the hidden layer.

$$\begin{aligned} N_1^{output} &= max(0, N_1^{input}) \\ &= max(0, 1.752) \\ &= 1.752 \end{aligned} \tag{3}$$

$$\begin{aligned} N_2^{output} &= max(0, N_2^{input}) \\ &= max(0, 0.808) \\ &= 0.808 \end{aligned} \tag{4}$$

2. **Step2**: Get the values of nodes after the softmax function operation g in the output layer.

   - The input value $O_i^{input}, i = 1, 2, 3$ of the node O_i in the hidden layer.

$$\begin{aligned} O_1^{input} &= N_1^{output} w_{N_1,O_1} + N_2^{output} w_{N_2,O_1} + b_{O_1} \\ &= 1.752 \times (-0.1) + 0.808 \times 1.5 + 0 \\ &= 1.0368 \end{aligned} \tag{5}$$

$$\begin{aligned} O_2^{input} &= N_1^{output} w_{N_1,O_2} + N_2^{output} w_{N_2,O_2} + b_{O_2} \\ &= 1.752 \times 2.4 + 0.808 \times (-5.2) + 0 \\ &= 0.0032 \end{aligned} \tag{6}$$

$$\begin{aligned} O_3^{input} &= N_1^{output} w_{1,N_3} + N_2^{output} w_{2,N_3} + b_{O_3} \\ &= 1.752 \times (-2.2) + 0.808 \times 3.7 + 1 \\ &= 0.1352 \end{aligned} \tag{7}$$

   - The output value $O_i^{output}, i = 1, 2, 3$ of the node O_i in the hidden layer.

$$\begin{aligned} O_1^{output} &= softmax(O_1^{input}) = \frac{e^{O_1^{input}}}{e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}}} \\ &= \frac{e^{1.0368}}{e^{1.0368} + e^{0.1352} + e^{0.0032}} \\ &= 0.568 \end{aligned} \tag{8}$$

$$O_2^{output} = softmax(O_2^{input}) = \frac{e^{O_2^{input}}}{e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}}} \tag{9}$$
$$= \frac{e^{0.1352}}{e^{1.0368} + e^{0.1352} + e^{0.0032}}$$
$$= 0.202$$

$$O_3^{output} = softmax(O_3^{input}) = \frac{e^{O_3^{input}}}{e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}}} \tag{10}$$
$$= \frac{e^{0.0032}}{e^{1.0368} + e^{0.1352} + e^{0.0032}}$$
$$= 0.23$$

Till now, we know that $sample_1$ ($x_1 = 0.04$,$x_2 = 0.42$, target=0) goes through the forward propagation of the neural network and generates three predictions that are

- $Pred_1 = O_1^{output} = 0.948$ means the 'probability' that $sampe_1$is assigned with Target=0.
- $Pred_2 = O_2^{output} = 0.042$ means the 'probability' that $sampe_1$is assigned with Target=1.
- $Pred_3 = O_3^{output} = 0.009$ means the 'probability' that $sampe_1$is assigned with Target=2.

3. **Step3**: Calculate the 'difference' between the prediction and the actual value via Cross Entropy. The actual target observation of $sample_1$ is

   - $Act_1 = 1$ means the 'probability' that $sampe_1$is assigned with Target=0.
   - $Act_2 = 0$ means the 'probability' that $sampe_1$is assigned with Target=1.
   - $Act_3 = 0$ means the 'probability' that $sampe_1$is assigned with Target=2.

   Therefore, the Cross-Entropy(CE) of $sample_1$ with Target=0 is

$$CE_{sample_1} = -\sum_{i}^{M} Act_i log(Pred_i), M = \text{number of nodes in the hidden layer} \tag{11}$$
$$= -Act_1 \times log(Pred_1) - Act_2 \times log(Pred_2) - Act_3 \times log(Pred_3)$$
$$= -1 \times log(Pred_1) - 0 \times log(Pred_2) - 0 \times log(Pred_3)$$
$$= -1 \times log(Pred_1) = -1 \times log(0.568) = 0.5656$$

We repeat **Step1**,**Step2** and **Step3** on $sample2$ ($x_1 = 1$,$x_2 = 0.54$, target=1) and $sample3$ ($x_1 = 0.5$,$x_2 = 0.65$, target=2), and we get followings:

- $sample2$ ($x_1 = 1$,$x_2 = 0.54$, target=1)

Backward Propagation

Taking b_{O_1} as an example. The updated b'_{O_1} is the previous b_{O_1} subtract the learning rate$r = 0.5$ times the 'difference', which is $b'_{O_1} = b_{O_1} - r\times$'difference', where 'difference'= $\frac{\mathrm{d}CE^{total}}{db_{O_1}}$. Next, I will show you how to calculate $\frac{\mathrm{d}CE^{total}}{db_{O_1}}$.

1. Derivative of the sum is the sum of derivatives:

$$\frac{\mathrm{d}CE^{total}}{\mathrm{d}b_{O_1}} = \frac{\mathrm{d}CE^{sample1}}{\mathrm{d}b_{O_1}} + \frac{\mathrm{d}CE^{sample2}}{\mathrm{d}b_{O_1}} + \frac{\mathrm{d}CE^{sample3}}{\mathrm{d}b_{O_1}}$$

   - Taking $\frac{\mathrm{d}CE^{sample1}}{\mathrm{d}b_{O_1}}$ as an example. The key is to find the connection between the numerator $CE^{sample1}$ and the dominator b_{O_1}, and then solve it by the chain rule.

(a) $CE^{sample1} = -\log(Pred_1^{sample1})$ according to Equation.11, therefore,

$$\frac{\mathrm{d}CE^{sample1}}{\mathrm{d}Pred_1^{sample1}} = \frac{\mathrm{d}}{\mathrm{d}Pred_1^{sample1}} - \log(Pred_1^{sample1}) = \frac{-1}{Pred_1^{sample1}} \tag{12}$$

(b) $Pred_1^{sample1} = \mathrm{softmax}(O_1^{input})$ according to Equation.8, therefore,

$$\begin{aligned}
\frac{\mathrm{d}Pred_1^{sample1}}{\mathrm{d}O_1^{input}} &= \frac{\mathrm{d}}{\mathrm{d}O_1^{input}}\mathrm{softmax}(O_1^{input}) \\
&= \frac{\mathrm{d}}{\mathrm{d}O_1^{input}}\left(\frac{e^{O_1^{input}}}{e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}}}\right) \\
&= \frac{\frac{\mathrm{d}}{\mathrm{d}O_1^{input}} e^{O_1^{input}}(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}}) - e^{O_1^{input}} \frac{\mathrm{d}}{\mathrm{O_1^{input}}}(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}})}{(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}})^2} \\
&= \frac{e^{O_1^{input}}(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}}) - (e^{O_1^{input}})^2}{(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}})^2} \\
&= \frac{e^{O_1^{input}}[(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}}) - e^{O_1^{input}}]}{(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}})^2} \\
&= \frac{e^{O_1^{input}}}{(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}})} \times \frac{(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}}) - e^{O_1^{input}}}{(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}})} \\
&= \frac{e^{O_1^{input}}}{(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}})} \times \left(1 - \frac{e^{O_1^{input}}}{(e^{O_1^{input}} + e^{O_2^{input}} + e^{O_3^{input}})}\right) \\
&= Pred_1^{sample1} \times (1 - Pred_1^{sample1})
\end{aligned} \tag{13}$$

note: Quotient rule.

(c) $O_1^{input} = N_1^{output} w_{N_1,O_1} + N_2^{output} w_{N_2,O_1} + b_{O_1}$ according to Equation.5, therefore,

$$\begin{aligned}
\frac{\mathrm{d}O_1^{input}}{\mathrm{d}b_{O_1}} &= \frac{\mathrm{d}}{\mathrm{d}b_{O_1}}(N_1^{output} w_{N_1,O_1} + N_2^{output} w_{N_2,O_1} + b_{O_1}) \\
&= 0 + 0 + 1 = 1
\end{aligned} \tag{14}$$

note: treating other parameters as their current values when updating O_1^{input}.

Therefore,

$$\begin{aligned}
\frac{\mathrm{d}CE^{sample1}}{\mathrm{d}b_{O_1}} &= \frac{-1}{Pred_1^{sample1}} \times Pred_1^{sample1} \times (1 - Pred_1^{sample1}) \times 1 \\
&= Pred_1^{sample1} - 1
\end{aligned} \tag{15}$$

A common neural network architecture involves:

1. Three layers and each layer has a number of nodes/neurons:
   - Input layer has 2 nodes x_1 and x_2. The number of nodes in the Input layer is the number of features in the dataset. In this example, each sample has 2 features.
   - Hidden layer has 2 nodes N_1 and N_2. The number of nodes in the Hidden layer is customized by us. In this example, We specify that there are two neurons.
   - Output layer has 3 nodes O_1, O_2 and O_3. The number of nodes in the Output layer is the number of unique targets. In this example, the dataset has 3 targets $(0, 1, 2)$.
2. Parameters (Weights and bias): In this example, parameters exist:
   - between Input layer and Hidden layer:
     - W_{1,N_1}, W_{2,N_1}, b_{N_1}

- W_{1,N_2}, W_{2,N_2}, b_{N_2}

- between Hidden layer and Output layer:
  - W_{N_1,O_1}, W_{N_2,O_1}, b_{O_1}
  - W_{N_2,O_1}, W_{N_2,O_2}, b_{O_2}
  - W_{N_2,O_3}, W_{N_2,O_3}, b_{O_3}

3. Two kinds of functions (one in the Hidden layer, and one in the Output layer):
   - Activation function f in the Hidden layer for each node. In this example, $f(x) = log(1 + e^x)$.
   - Softmax function g in the Output layer for each node. In this example, $g(x_i) = \frac{e^{x_i}}{sum(e^{x_i})}$.