CP8318 Machine Learning - Assignment 3

Student: Nebojsa Djosic, nebojsa.djosic@torontomu.ca

Lecturer: Elodie Lugez, Ph.D., cps8318@cs.torontomu.ca

Contents

Introduction

In this report, we show detailed, step-by-step calculations as described and denoted in [1], for both forward and backward propagation for a Neural Network shown in Figure 1

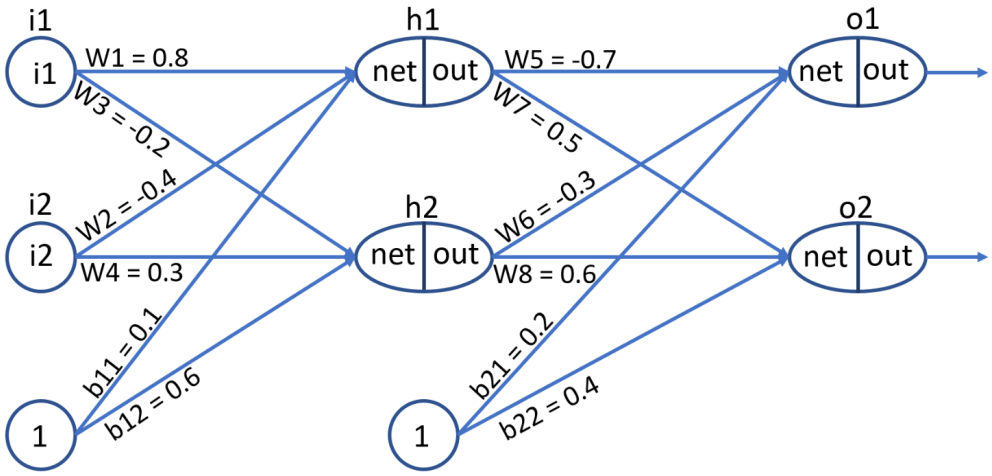

Figure 1: Neural Network Diagram

The input values are 0.9 and 0.3 for inputs 1 and 2 respectively. The target output values are 0.01 and 0.99 for outputs 1 and 2 respectively. The learning rate value is 0.5. The calculations will show the following:

1. The total error of the network given by the formula: $E_{total} = \sum \frac{1}{2}(target - output)^2$
2. Optimized neural network weights after the first round of backpropagation.
3. The total error after the first round of backpropagation.

Forward Pass

The forward pass computes the output of the network by passing inputs through layers one by one all the way to the output. We use inputs, weights and biases, and an activation (non-linear) function one layer at a time going forward through the network finishing with the output nodes.

The pre-activation value $z_i^{(l)}$, sometimes denoted as $net_i^{(l)}$ is the sum of weighted inputs and a bias:

$$z_i^{(l)} = \sum_j W_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)} \tag{1}$$

where:

- $W_{ij}^{(l)}$ is the weight for the connection between node j in layer $l-1$ to node i in layer l.
- $a_j^{(l-1)}$ is the activation output of node (neuron) j in the previous layer.
- $b_i^{(l)}$ is the bias term for node i in layer l.

Using the notation from Figure 1, and the approach in [1] for the first hidden node h_1, the pre-activation value is calculated using Eq. 1 as follows:

$$net_{h_1} = w_1 * i_1 + w_2 * i_2 + b_{11} * 1$$

where w_1 is weight for input i_1, w_2 is weight for input i_2, and b_{11} is bias. The net_{h_1} is the pre-activation value, usually denoted as $z_i^{(l)}$, sometimes referred to as *logit*, or more generally as *weighted sum*. Using the values from Figure 1 and Eq. 1 we can calculate as:

$$net_{h1} = 0.8 * 0.9 + (-0.4) * 0.3 + 0.1 * 1 = 0.7$$

The calculated pre-activation value is put through the logistic, sigmoid, activation function to get the output of the node. The Sigmoid function used is:

$$out_i = \frac{1}{1 + e^{-net_i}} \tag{2}$$

Using Eq. 2 we can calculate the output for out_{h1} as:

$$out_{h1} = \frac{1}{1 + e^{-0.7}} = 0.6681877721681662$$

For the second hidden neuron, we compute the output using the same input with different weights and biases and pass it through the same sigmoid activation function just like we did for the first hidden node:

$$net_{h2} = (-0.2) * 0.9 + 0.3 * 0.3 + 0.6 * 1 = 0.51$$

$$out_{h2} = \frac{1}{1 + e^{-0.51}} = 0.6248064744684293$$

The output of the hidden layer nodes becomes the input for the next layer in this case the output layer nodes as shown in Figure 1.

The same set of calculations is performed again at this layer. First, we calculate values for the output node 1:

$$net_{o1} = (-0.7) * 0.6681877721681662 + (-0.3) * 0.6248064744684293 + 0.2 * 1 = -0.455173382858245$$

$$out_{o1} = \frac{1}{1 + e^{-0.455173382858245}} = 0.3881314542973173$$

Next, the same calculation is applied to the output node 2:

$$net_{o2} = 0.5 * 0.6681877721681662 + 0.6 * 0.6248064744684293 + 0.4 * 1 = 1.1089777707651405$$

$$out_{o2} = \frac{1}{1 + e^{-0.51}} = 0.7519384871981952$$

At this point, we reach the last, output layer and have the outputs for each neuron, thus the forward pass is completed.

We can now calculate the total error using the formula:

$$\begin{gathered} E_{total} = \sum \frac{1}{2}(target - output)^2 \\ \text{or the same formula rearanged:} \\ E_{total} = \frac{1}{2} \sum (target - output)^2 \end{gathered} \tag{3}$$

The target value given for the output node 1 is 0.01 and for the output node 2 is 0.99. Using the Eq. 3 we calculate as:

$$E_1 = (0.01 - 0.3881314542973173)^2 = 0.14298339672900418$$

$$E_2 = (0.99 - 0.7519384871981952)^2 = 0.056673283877483854$$

$$E_{total} = \frac{1}{2}(0.14298339672900418 + 0.056673283877483854) = 0.09982834030324401$$

Answer to the first question

The answer to the first question "How much is the error function E_{total} of this network?" is 0.09982834030324401

Backward Pass

Backward Pass, is a part of backpropagation, sometimes used as a synonym, and is a general term used for the process of applying the chain rule of calculus to compute gradients of the loss function with respect to each weight in the network propagating errors backward layer by layer through the network. The gradient is a vector consisting of all the partial derivatives with respect to weights and biases. In this way, the total error is minimized and thus the performance of the network is gradually improved as the process moves through each layer and is repeated during the model training phase.

Typically for the backward pass processing we need to define a loss function L, in this work we use E_{total}, the activation $a^{(l-1)}$, in this work we use out_i, from the previous layer $(l-1)$, which serves as the input to the layer l, and the weights $W^{(l)}$ and biases $b^{(l)}$ for layer l. The backward pass process calculates the gradient of the loss function L with respect to each weight like $W_{ij}^{(l)}$, which is the weight connecting neuron j in layer l-1 to neuron i in layer l.

To calculate the gradient for the loss function L, in this report, we also use E_{total}, with respect to all weights going backward from the output, we use the following formula:

$$\frac{\partial L}{\partial W_{ij}^{(l)}} = \delta_i^{(l)} \cdot a_j^{(l-1)}$$

where:

- $\delta_i^{(l)}$ is the error for node i in layer l.

- $a_j^{(l-1)}$ is the activation output of node j in the previous layer.

To optimize the weight and biases of a network we need to find out how much a small change to each weight and bias affects the total error, as we keep making small changes to minimize the total error. This means we need to keep calculating the partial derivative for each weight, for each neuron, for each layer, adjust weights and biases, calculate the output through the forward pass, calculate the total error after each pass, and keep repeating the process until we minimize the error.

Since the partial derivative of the total error with respect to a single weight is a composite, we must apply the chain rule of calculus. Going backwards from the output i out_i to calculate the partial derivative of the total error E_{total} with respect to the weight $w_{l,j}$ we would use the following chain rule sequence:

$$\frac{\partial E_{total}}{\partial w_{l,j}} = \frac{\partial E_{total}}{\partial out_i} * \frac{\partial out_i}{\partial net_i} * \frac{\partial net_i}{\partial w_{l,j}} \tag{4}$$

Output Layer

We will start with $\frac{\partial E_{total}}{\partial out_i}$ calculation. Using the total error Eq. 3 and adopting it for a single node out_{o1} we can simplify the calculation of the partial derivative as:

$$\frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial}{\partial out_{o1}} \left(\frac{1}{2} (target_{o1} - out_{o1})^2 \right)$$

Applying the chain rule we get the first part:

$$= \frac{1}{2} \cdot 2(target_{o1} - out_{o1}) = (target_{o1} - out_{o1})$$

Next, differentiating with respect to out_{o1} we get the second part as:

$$\frac{\partial}{\partial out_{o1}} (target_{o1} - out_{o1}) = -1$$

We combine the two parts into the chain:

$$\frac{\partial E_{total}}{\partial out_{o1}} = (target_{o1} - out_{o1}) \cdot (-1) = -(target_{o1} - out_{o1})$$

Finally, we can complete the calculation for the first term in Eq. 4:

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(0.01 - 0.3881314542973173) = 0.3781314542973173$$

We continue with the calculation of the second term in Eq. 4, the partial derivative of out_{o1} with respect to the pre-activation value denoted as net_{o1}.

The out_{o1} is the result of the activation function, the sigmoid function Eq. 2 which we will rewrite as:

$$out_{o1} = (1 + e^{-net_{o1}})^{-1}$$

Using the chain rule we first differentiate the $(1 + e^{-net_{o1}})^{-1}$ with respect to $(1 + e^{-net_{o1}})$:

$$\frac{d}{d(1 + e^{-net_{o1}})}(1 + e^{-net_{o1}})^{-1} = -(1 + e^{-net_{o1}})^{-2}$$

Next, we differentiate $1 + e^{-net_{o1}}$ with respect to net_{o1}:

$$\frac{d}{d(net_{o1})}(1 + e^{-net_{o1}}) = -e^{-net_{o1}}$$

Combining these using the chain rule, we get:

$$\frac{\partial out_{o1}}{\partial net_{o1}} = -(1 + e^{-net_{o1}})^{-2} \cdot (-e^{-net_{o1}})$$

or we can rewrite it as:

$$\frac{\partial out_{o1}}{\partial net_{o1}} = \frac{e^{-net_{o1}}}{(1 + e^{-net_{o1}})^2}$$

Using Eq. 2 we know that the output value is:

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}}$$

thus we can expand it to:

$$1 - out_{o1} = 1 - \frac{1}{1 + e^{-net_{o1}}} = \frac{e^{-net_{o1}}}{1 + e^{-net_{o1}}}$$

and if we multiply it by out_{o1} we get the same expression as in the partial derivative:

$$out_{o1} \cdot (1 - out_{o1}) = \left(\frac{1}{1 + e^{-net_{o1}}}\right) \cdot \left(\frac{e^{-net_{o1}}}{1 + e^{-net_{o1}}}\right) = \frac{e^{-net_{o1}}}{(1 + e^{-net_{o1}})^2}$$

thus, we can express the partial derivative of the output with respect to pre-activation only in terms of output:

$$\frac{\partial out_{o1}}{\partial net_{o1}} = \frac{e^{-net_{o1}}}{(1 + e^{-net_{o1}})^2} = out_{o1} \cdot (1 - out_{o1}) \tag{5}$$

We can now complete the calculation of the second term in Eq. 4:

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1} \cdot (1 - out_{o1}) = 0.3881314542973173 \cdot (1 - 0.3881314542973173) = 0.23748542848236678$$

The last term in Eq. 4 is the partial derivative of the pre-activation net_{o1} with respect to the weight w_5. We start by recalling the equation used to calculate the pre-activation value which is:

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

The partial derivative of net_{o1} with respect to w_5 is:

$$\frac{\partial net_{o1}}{\partial w_5} = \frac{\partial}{\partial w_5}(w_5 \cdot out_{h1} + w_6 \cdot out_{h2} + b_2 \cdot 1)$$

All variables other than w_5, namely w_6, out_{h2}, and b_2, are constants for the particle derivative calculation, and since they don't depend on w_5 their derivatives are zeros. As a result, we only

differentiate with respect to w_5 as follows:

$$\frac{\partial net_{o1}}{\partial w_5} = out_{h1} \tag{6}$$

So, the partial derivative of net_{o1} with respect to w_5 is equal to the output out_{h1} that is:

$$\frac{\partial net_{o1}}{\partial w_5} = 0.6681877721681662$$

Finally, using Eq. 4 to calculate the partial derivative of E_{total} with respect of w_5 using the chain rule we get:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

Substituting with numbers we get the final result:

$$\frac{\partial E_{total}}{\partial w_5} = 0.3781314542973173 \cdot 0.23748542848236678 \cdot 0.6681877721681662$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.06000373665233793$$

To get a new value for w_5, the initial value of w_5, -0.7, as illustrated in Figure 1, must be decreased by the partial derivative we just calculated, 0.06000373665233793, but we also typically adjust it by a learning rate, usually denoted as α, in our case $\alpha = 0.5$:

$$w_5^+ = w_5 - \alpha \cdot \frac{\partial E_{total}}{\partial w_5}$$

or:

$$w_5^+ = -0.7 - 0.5 \cdot 0.06000373665233793 = -0.7300018683261689$$

With this, we concluded the calculation for w_5. To proceed we must repeat the same calculation for the rest of the weights and biases in the last, output, layer. We also need to complete the same

calculations for all hidden layers to fully complete the first iteration of backpropagation. After this, we can repeat the forward pass using the newly calculated values.

Using Figure 1 and Eq. 4 and the method already shown we can calculate the w_6^+:

$$\frac{\partial E_{total}}{\partial w_6} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_6}$$

The first, and the second term in the chain rule we already calculated and can reuse: 0.3781314542973173, and 0.23748542848236678 respectively. The third term is the output value from the node from which the weight originates in this case out_{h2} which we already calculated to be 0.6248064744684293. Using these numbers we can calculate the partial derivative of the total error, or the loss, E_{total} with respect to w_6 as:

$$\frac{\partial E_{total}}{\partial w_6} = 0.3781314542973173 \cdot 0.23748542848236678 \cdot 0.6248064744684293$$

$$\frac{\partial E_{total}}{\partial w_6} = 0.05610806529881223$$

Thus, the new value w_6^+ is:

$$w_6^+ = w_6 - \alpha \cdot \frac{\partial E_{total}}{\partial w_6}$$

or:

$$w_6^+ = -0.3 - 0.5 \cdot 0.05610806529881223 = -0.3280540326494061$$

The calculate the adjustment for bias b_{21} used as input to the output node $o1$, we can also reuse the first two terms from the Eq. 4, and the third term is simply 1, since the bias is contributing in a linear way, without any transformation, the partial derivative of anything with respect to the bias will always be 1, since any infinitesimal change to the bias linearly, directly shifts the output by the same amount.

$$\frac{\partial E_{total}}{\partial b_{21}} = 0.3781314542973173 \cdot 0.23748542848236678 \cdot 1 = 0.08980071044645889$$

Thus the new value b_{21}^{+} is:

$$b_{21}^{+} = b_{21} - \alpha \cdot \frac{\partial E_{total}}{\partial b_{21}}$$

or:

$$b_{21}^{+} = 0.2 - 0.5 \cdot 0.08980071044645889 = 0.15509964477677057$$

Using Figure 1 and Eq. 4 we can calculate w_7^{+} in the same way:

$$\frac{\partial E_{total}}{\partial w_7} = \frac{\partial E_{total}}{\partial out_{o2}} * \frac{\partial out_{o2}}{\partial net_{o2}} * \frac{\partial net_{o2}}{\partial w_7}$$

As shown before, we calculate the first term as:

$$\frac{\partial E_{total}}{\partial out_{o2}} = -(target_{o2} - out_{o2})$$

$$\frac{\partial E_{total}}{\partial out_{o2}} = -(0.99 - 0.7519384871981952) = -0.23806151280180476$$

The second term is calculated as:

$$\frac{\partial out_{o2}}{\partial net_{o2}} = out_{o2} \cdot (1 - out_{o2}) = 0.7519384871981952 \cdot (1 - 0.7519384871981952) = 0.18652699866828482$$

The third term is the output out_{h1} which we already calculate to be 0.6681877721681662.

$$\frac{\partial E_{total}}{\partial w_7} = -0.23806151280180476 \cdot 0.18652699866828482 \cdot 0.6681877721681662 = -0.029670810857796024$$

Thus, the new value w_7^{+} is:

$$w_7^{+} = w_7 - \alpha \cdot \frac{\partial E_{total}}{\partial w_7}$$

or:

$$w_7^{+} = 0.5 - 0.5 \cdot 0.029670810857796024 = 0.514835405428898$$

Using Figure 1 and Eq. 4 we can calculate w_8^+:

$$\frac{\partial E_{total}}{\partial w_8} = \frac{\partial E_{total}}{\partial out_{o2}} \cdot \frac{\partial out_{o2}}{\partial net_{o2}} \cdot \frac{\partial net_{o2}}{\partial w_8}$$

We will be reusing the first two terms from the calculations for w_7^+, and the third term is the output of the $h2$ which we calculated to be 0.6248064744684293.

We can calculate the partial derivative as follows:

$$\begin{aligned}\frac{\partial E_{total}}{\partial w_8} &= -0.23806151280180476 \cdot 0.18652699866828482 \cdot 0.6248064744684293 \\ &= -0.027744468694068596\end{aligned}$$

Thus, the new value w_8^+ is:

$$w_8^+ = w_8 - \alpha \cdot \frac{\partial E_{total}}{\partial w_8}$$

or:

$$w_8^+ = 0.6 - 0.5 \cdot -0.027744468694068596 = 0.6138722343470343$$

The calculate the adjustment for bias b_{22} we will reuse the first two terms only

$$\frac{\partial E_{total}}{\partial b_{22}} = -0.23806151280180476 \cdot 0.18652699866828482 = -0.04440489948135211$$

Thus the new value b_{22}^+ is:

$$b_{22}^+ = b_{22} - \alpha \cdot \frac{\partial E_{total}}{\partial b_{22}}$$

or:

$$b_{22}^+ = 0.4 - 0.5 \cdot -0.04440489948135211 = 0.42220244974067606$$

Hidden Layer

To calculate the adjustments for weights and biases in the hidden layer we will be using the same approach as for the output layer but we need to include both output layer nodes since the network

is fully connected and each hidden layer affects both output neurons. The chain rule would consist of the following:

1. Calculate the partial derivative of the total error with respect to the output layer activations.
2. Calculate the partial derivative of the output layer activations with respect to the output layer inputs.
3. Calculate the partial derivative of the output layer inputs with respect to the hidden layer activations.
4. Calculate the partial derivative of the hidden layer activations with respect to the hidden layer inputs.
5. Calculate the partial derivative of the hidden layer inputs with respect to the hidden layer weights.

We can represent this as follows:

$$\frac{\partial E_{total}}{\partial w_h^k} = \left(\sum_{j=1}^{n} \frac{\partial E_{total}}{\partial out_{o_j}} \cdot \frac{\partial out_{o_j}}{\partial net_{o_j}} \cdot \frac{\partial net_{o_j}}{\partial out_{h_h}} \right) \cdot \frac{\partial out_{h_h}}{\partial net_{h_h}} \cdot \frac{\partial net_{h_h}}{\partial w_h^k} \tag{7}$$

where w_h^k is the weight connecting to the hidden neuron w_h, and in our case $n = 2$ since we have two output neurons in our network in Figure 1.

For instance, for the weight W1:

$$\frac{\partial E_{\text{total}}}{\partial W1} = \left(\frac{\partial E_{\text{total}}}{\partial out_{o1}} \cdot \frac{\partial out_{o1}}{\partial net_{o1}} \cdot \frac{\partial net_{o1}}{\partial h1} + \frac{\partial E_{\text{total}}}{\partial out_{o2}} \cdot \frac{\partial out_{o2}}{\partial net_{o2}} \cdot \frac{\partial net_{o2}}{\partial h1} \right) \cdot \frac{\partial h1}{\partial net_{h1}} \cdot \frac{\partial net_{h1}}{\partial W1} \tag{8}$$

Since we already calculated the first two terms during the backward pass through the output layer, we can reuse them.

$$\frac{\partial E_{total}}{\partial out_{o1}} = 0.3781314542973173$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = 0.23748542848236678$$

$$\frac{\partial E_{total}}{\partial out_{o2}} = -0.23806151280180476$$

$$\frac{\partial out_{o2}}{\partial net_{o2}} = 0.18652699866828482$$

The third term is the partial derivative of the pre-activation of the output neuron with respect to the output of the hidden layer neuron. As we have shown above the equation used to calculate the pre-activation value of the output neurons is a sum of hidden layer neurons' outputs multiplied by weights. To calculate the partial derivative with respect to the output value of a specific hidden neuron we need to isolate only the term, and there will be only one, that involves the output of this neuron, and thus the partial derivative will be equal to the weight connecting this hidden neuron the the output neuron. In other words, the rate of change of the pre-activation depends directly on the weight:

$$\frac{\partial net_{o_j}}{\partial out_{h_h}} = w_{h_j}$$

Since the values for all weights are given we can write the third term for all hidden layers as:

$$\frac{\partial net_{o1}}{\partial out_{h1}} = W5 = -0.7$$

$$\frac{\partial net_{o1}}{\partial out_{h2}} = W6 = -0.3$$

$$\frac{\partial net_{o2}}{\partial out_{h1}} = W7 = 0.5$$

$$\frac{\partial net_{o2}}{\partial out_{h2}} = W8 = 0.6$$

The fourth term is the partial derivative of the sigmoid function with respect to pre-activation value. We have already shown the proof for Eq. 5 and here we will calculate the values:

$$\frac{\partial out_{h1}}{\partial net_{h1}} = 0.6681877721681662 \cdot (1 - 0.6681877721681662) = 0.22171287329310904$$

$$\frac{\partial out_{h2}}{\partial net_{h2}} = 0.6248064744684293 \cdot (1 - 0.6248064744684293) = 0.2344233439307613$$

Finally, using the Eq. 6 we can calculate the fifth and final term for all hidden layer weights. The only difference is that we will be using inputs 1 and 2 instead of hidden neuron outputs

$$\frac{\partial net_{h1}}{\partial W1} = i1 = 0.9$$
$$\frac{\partial net_{h1}}{\partial W2} = i2 = 0.3$$
$$\frac{\partial net_{h2}}{\partial W3} = i1 = 0.9$$
$$\frac{\partial net_{h2}}{\partial W4} = i2 = 0.3$$

We can now calculate all partial derivatives with respect to weights for the hidden layer:

$$\frac{\partial E_{total}}{\partial W1} = 0.01699330154632806$$
$$\frac{\partial E_{total}}{\partial W2} = -0.005657865120583191$$
$$\frac{\partial E_{total}}{\partial W3} = -0.011305027676755635$$
$$\frac{\partial E_{total}}{\partial W4} = -0.0037683425589185447$$

Finally, we can calculate the adjusted values for each weight in the hidden layer:

$$W1^{+} = W1 - \alpha \cdot \frac{\partial E_{total}}{\partial W1} = 0.8084867976808748$$

$$W2^{+} = W2 - \alpha \cdot \frac{\partial E_{total}}{\partial W2} = -0.3971710674397084$$
$$W3^{+} = W3 - \alpha \cdot \frac{\partial E_{total}}{\partial W3} = -0.1943474861616222$$
$$W4^{+} = W4 - \alpha \cdot \frac{\partial E_{total}}{\partial W4} = 0.30188417127945927$$

To calculate the adjusted values for hidden layer biases we will be reusing the same Eq. 7. In the case of biases, the last term is always 1 as we have shown before. For each of the two biases we can reuse values we already calculated, for instance using Eq. 8 without the last term. The adjusted

values for biases are:

$$b11^{+} = b11 - \alpha \cdot \frac{\partial E_{total}}{\partial b11} = 0.109429775200972$$

$$b12^{+} = b12 - \alpha \cdot \frac{\partial E_{total}}{\partial b12} = 0.6062805709315309$$

Answer to the second question

The following are optimize network weights after the first round of backpropagation:

- $W1^{+} = 0.8084867976808748$
- $W2^{+} = -0.3971710674397084$
- $W3^{+} = -0.1943474861616222$
- $W4^{+} = 0.30188417127945927$
- $W5^{+} = -0.7300018683261689$
- $W6^{+} = -0.3280540326494061$
- $W7^{+} = 0.514835405428898$
- $W8^{+} = 0.6138722343470343$
- $b11^{+} = 0.109429775200972$
- $b12^{+} = 0.6062805709315309$
- $b21^{+} = 0.15509964477677057$
- $b22^{+} = 0.42220244974067606$

Answer to the third question

To calculate the total error after the first round of backpropagation we have to take the adjusted (optimized) weights and biases and perform forward propagation. We will use Eq. 1 to calculate

activation and apply the sigmoid function Eq. 2 to calculate the output. Once we calculate the output for the last layer we will use Eq. 3 to calculate the total error. This process has been demonstrated in the section Forward Pass.

After the forward pass calculations, we get the following values:

- *h1*: 0.6721480673334459
- *h2*: 0.6275996749712447
- *o1*: 0.36784887381481113
- *o2*: 0.7601521938672593
- *error for o1*: 0.1280558164905286
- *error for o2*: 0.052830013984033954
- **Total Error: 0.09044291523728128**

References

[1] Matt Mazur. *A Step by Step Backpropagation Example*. 2015. URL: `https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/` (visited on 10/31/2024).