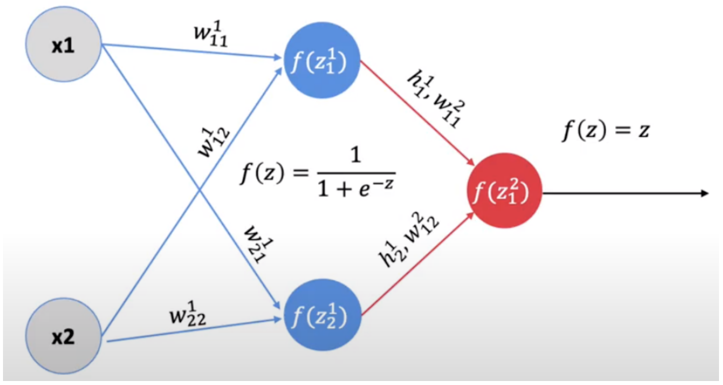
Session 3: Backpropagation in Neural Networks

For a neural network, you will now learn how the loss function is minimised using the gradient descent function by finding the optimum values of weights and biases using **backpropagation**.

The gradient descent algorithm presents us with the following parameter update equation:

$$w^{l}_{kj} = w^{l}_{kj} - \eta\frac{\partial L}{\partial w^{l}_{kj}}$$

where k and j are the indices of the weight in the weight matrix and l is the index of



the layer to which it belongs.

Given the neural network in the diagram, for the **output** layer, the following weights and bias terms will be updated using the gradient descent update equation:

$$w^2_{11} = w^2_{11} - \eta\frac{\partial L}{\partial w^2_{11}}$$
$$w^2_{12} = w^2_{12} - \eta\frac{\partial L}{\partial w^2_{12}}$$
$$b^2_1 = b^2_1 - \eta\frac{\partial L}{\partial b^2_1}$$

Now, for the **hidden** layer, the following weights and biases will be updated:

$$w_{11}^1 = w_{11}^1 - \eta \frac{\partial L}{\partial w_{11}^1} \qquad w_{21}^1 = w_{21}^1 - \eta \frac{\partial L}{\partial w_{21}^1}$$

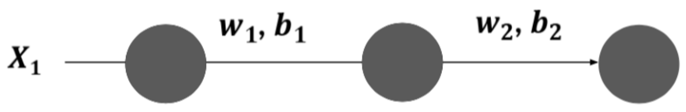
$$w_{12}^1 = w_{12}^1 - \eta \frac{\partial L}{\partial w_{12}^1} \qquad w_{22}^1 = w_{22}^1 - \eta \frac{\partial L}{\partial w_{22}^1}$$

$$b_1^1 = b_1^1 - \eta \frac{\partial L}{\partial b_1^1} \qquad b_2^1 = b_2^1 - \eta \frac{\partial L}{\partial b_2^1}$$

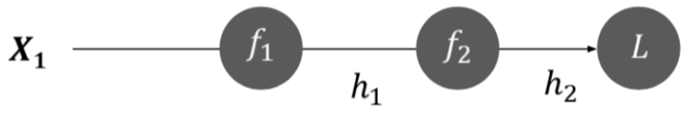
To compute these gradients, you use an algorithm called backpropagation.

As you can see in the above-mentioned formulas, there exist partial derivatives of the loss function L with respect to the weights and biases in these equations. To compute these, you use the chain rule. You can observe the dependencies of different layers in the gradient computation.

Now, let's simplify the neural network given above and represent it in a condensed format as shown below.



In this case, the loss function is a function of $w1$, $b1$, $w2$ and $b2$.

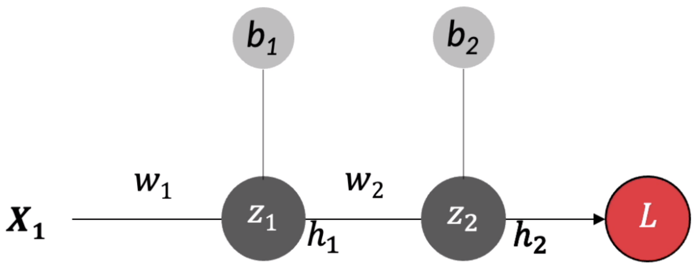


The loss function, the activation function and the cumulative inputs are shown in the following expressions:

$$Loss = \frac{1}{2}(y - h_2)^2$$
$$h_i = tanh(z_i)$$
$$z_i = w_i h_{i-1} + b_i$$

Now, let’s compute the gradient of the loss function with respect to one of the weights to understand how **backpropagation** works.

Suppose you want to calculate $\frac{\partial L}{\partial w_2}$, that is, the gradient of the loss function, with respect to the weight w_2.



Using the chain rule, you can say that:

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$

Based on the definition of the loss function, L is a direct function of h_2, h_2 is a function of z_2 and z_2 is a function of w_2.

Now let’s see how each of the three terms in the RHS of the equation are computed:

$$L = \frac{1}{2}(y - h_2)^2 \rightarrow\rightarrow \frac{\partial L}{\partial h_2} = \frac{\partial}{\partial h_2}\frac{1}{2}(y - h_2)^2 =- (y - h_2)$$

$$h_2 = tanh(z_2) \rightarrow\rightarrow \frac{\partial h_2}{\partial z_2} = 1 - tanh^2(z_2) = 1 - (h_2)^2$$

$$z_2 = w_2 h_1 + b_2 \rightarrow\rightarrow \frac{\partial z_2}{\partial w_2} = h_1$$

Hence, you get the gradient of the loss function with respect to w_2, which is shown below.

$$\frac{\partial L}{\partial w_2} = [- (y - h_2)][1 - (h_2)^2][h_1]$$

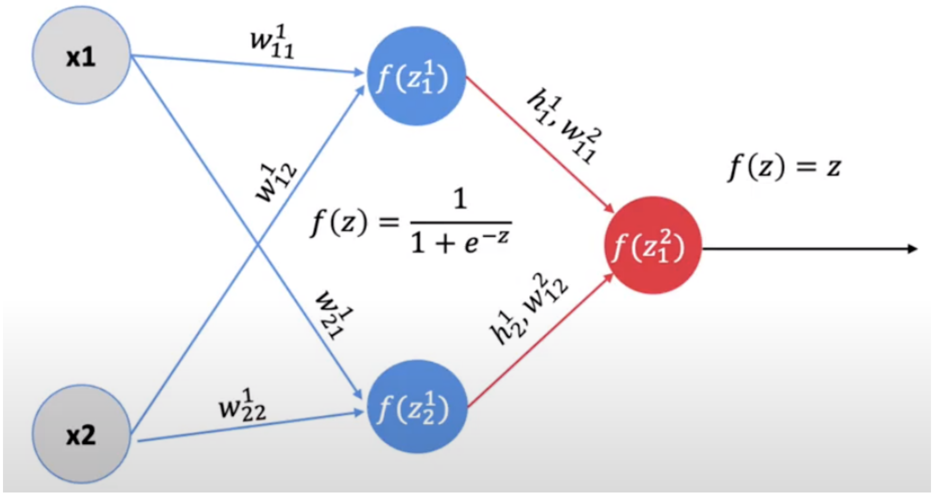
With this, you have completed the computation of the gradient of the loss function with respect to the weight w_2 for backpropagation.

Numerical Example Demonstrating Backpropagation

You gained an understanding of the backpropagation technique on a very simple neural network. You will now learn how the weights and biases, that is, the parameters of the neural network considered for the house price prediction example, change.

The housing data set has two inputs, which are the size of the house and the number of rooms available, and one output, which is the price of the house.

As seen in the computation of the forward pass, we randomly initialise the weights and biases in the network. Let's now take the same initialisation and the same input observation that you used earlier while doing forward propagation.



So, we have:

$$W^1, W^2 = \begin{bmatrix} w^1_{11} & w^1_{12} \\ w^1_{21} & w^1_{22} \end{bmatrix}, \begin{bmatrix} w^2_{11} \\ w^2_{12} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.15 \\ 0.5 & 0.6 \end{bmatrix}, \begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix}$$

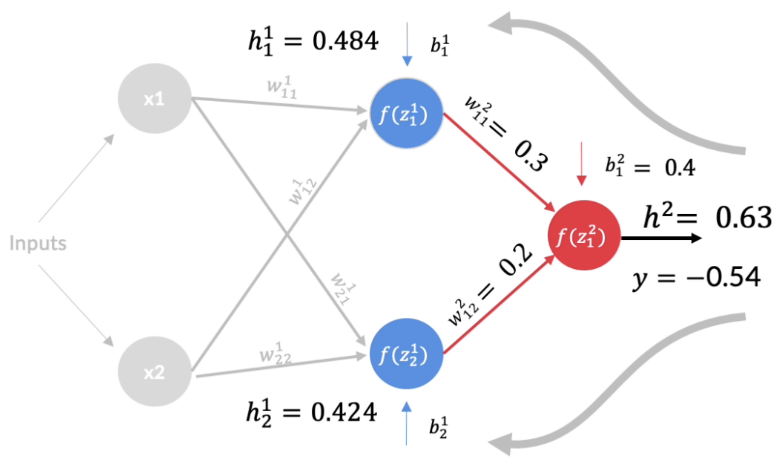
$$b^1, b^2 = \begin{bmatrix} b^1_1 \\ b^1_2 \end{bmatrix}, [b^2_1] = \begin{bmatrix} 0.1 \\ 0.25 \end{bmatrix}, [0.4]$$

And the input data is as follows:

$$X^1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -0.32 \\ -0.66 \end{bmatrix}$$

As previously calculated, the output prediction h^2 obtained is 0.63, whereas the actual output y is −0.54. Using backpropagation, let's update the weights and biases such that this difference between the predicted and the actual output gets minimised.

The steps taken to update the weights and biases between the hidden layer and the output layer are as follows.



First, you will focus on the weights for the output layer. Let's take the gradient of L with respect to w_{11}^2.

You know that:

$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial h_1^2}\frac{\partial h_1^2}{\partial z_1^2}\frac{\partial z_1^2}{\partial w_{11}^2} \text{ (using chain rule)}$$

1) $\frac{\partial L}{\partial h_1^2} = \frac{\partial}{\partial h_1^2}\frac{1}{2}(y - h_1^2)^2 =- (y - h_1^2)$

$\frac{\partial L}{\partial h_1^2} =- (- 0.54 - 0.63) = 1.17$

2) $\frac{\partial h_1^2}{\partial z_1^2} = 1$ as $h_1^2 = z_1^2$ (linear activation)

3). $\frac{\partial z_1^2}{\partial w_{11}^2} = \frac{\partial}{\partial w_{11}^2}(b_1^2 + w_{11}^2 h_1^1 + w_{12}^2 h_2^1)$

$\frac{\partial z_1^2}{\partial w_{11}^2} = (h_1^1) = 0.484$

Hence, this evaluates to $\frac{\partial L}{\partial w_{11}^2} = 1.17 * 1 * 0.484 = 0.5663$.

Now, using the update rule for gradient descent and considering the learning rate η as 0.2:

$$w_{11(updated)}^2 = w_{11}^2 - \eta\frac{\partial L}{\partial w_{11}^2} = 0.3 - (0.2 * 0.5663) = 0.1867$$

Similarly, $\frac{\partial L}{\partial w^2_{12}} = \frac{\partial L}{\partial h^2_1}\frac{\partial h^2_1}{\partial z^2_1}\frac{\partial z^2_1}{\partial w^2_{12}}$.

Since you have already computed the first two derivatives, let's now compute the third one.

$$\frac{\partial z^2_1}{\partial w^2_{12}} = \frac{\partial}{\partial w^2_{12}}(b^2_1 + w^2_{11}h^1_1 + w^2_{12}h^1_2)$$

$$\frac{\partial z^2_1}{\partial w^2_{12}} = (h^1_2) = 0.424$$

Hence, this evaluates to $\frac{\partial L}{\partial w^2_{12}} = 1.17 * 1 * 0.424 = 0.4961.$

Now,

$$w^2_{12(updated)} = w^2_{12} - \eta\frac{\partial L}{\partial w^2_{12}} = 0.2 - (0.2 * 0.4961) = 0.1008.$$

Similarly, for the bias **term**, you know that:

$$\frac{\partial L}{\partial b^2_1} = \frac{\partial L}{\partial h^2_1}\frac{\partial h^2_1}{\partial z^2_1}\frac{\partial z^2_1}{\partial b^2_1}$$

You have computed the first two derivatives already and the third one can be computed as shown below.

$$\frac{\partial z^2_1}{\partial b^2_1} = \frac{\partial}{\partial b^2_1}(b^2_1 + w^2_{11}h^1_1 + w^2_{12}h^1_2)$$

$$\frac{\partial z^2_1}{\partial b^2_1} = 1$$

Hence, this evaluates to $\frac{\partial L}{\partial b^2_1} = 1.17 * 1 * 1 = 1.17.$

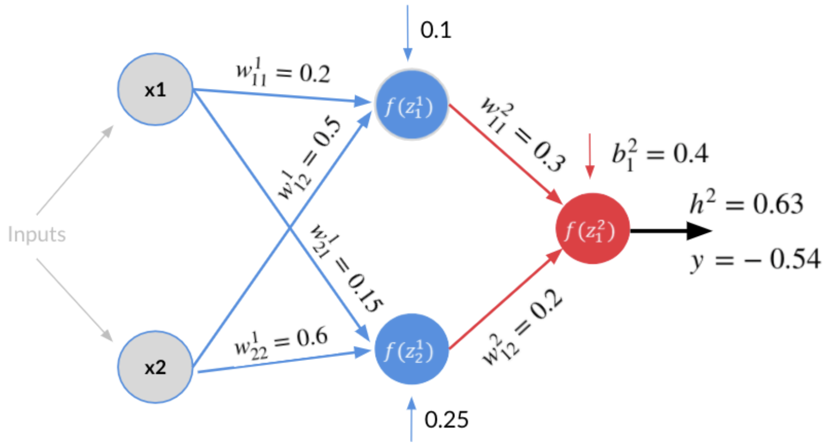
Now,

$$b^2_{1(updated)} = b^2_1 - \eta\frac{\partial L}{\partial b^2_1} = 0.4 - (0.2 * 1.17) = 0.166.$$

So, you have updated values of weights and biases of the output layer from a single iteration.

$$w^2_{11(updated)} = 0.1867, w^2_{12(updated)} = 0.1008, b^2_{1(updated)} = 0.166$$

The steps involved in computing the updated weights and biases in the hidden layer are shown in this image given below.



Now, let's start with computing the weights and biases corresponding to the **first neuron** of the hidden layer.

Taking the gradient of L with respect to w^1_{11}, you can say that:

$$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial h^2_1}\frac{\partial h^2_1}{\partial h^1_1}\frac{\partial h^1_1}{\partial w^1_{11}} = \frac{\partial L}{\partial h^2_1}\frac{\partial h^2_1}{\partial h^1_1}[\frac{\partial h^1_1}{\partial z^1_1}\frac{\partial z^1_1}{\partial w^1_{11}}]$$

You already know the first derivative term.

$$\frac{\partial L}{\partial h^2_1} = \frac{\partial}{\partial h^2_1}\frac{1}{2}(y - h^2_1)^2 =- (y - h^2_1)$$

$$\frac{\partial L}{\partial h^2_1} =- (- 0.54 - 0.63) = 1.17$$

Now, let's compute the second, third and fourth derivative terms.

1) $\frac{\partial h^2_1}{\partial h^1_1} = \frac{\partial}{\partial h^1_1}(b^2_1 + w^2_{11}h^1_1 + w^2_{12}h^1_2) = w^2_{11} = 0.30$

2). $\frac{\partial h^1_1}{\partial z^1_1} = \sigma(z^1_1)(1 - \sigma(z^1_1)) = h^1_1(1 - h^1_1) = 0.484(1 - 0.484)$

3) $\frac{\partial z^1_1}{\partial w^1_{11}} = \frac{\partial}{\partial w^1_{11}}(b^1_1 + w^1_{11}x_1 + w^1_{12}x_2) = x_1 =- 0.32$

Hence, this evaluates to $\frac{\partial L}{\partial w^1_{11}} = 1.17 * 0.30 * 0.484 * (1 - 0.484) * (- 0.32) =- 0.028.$

Now,

$w^1_{11(updated)} = w^1_{11} - \eta\frac{\partial L}{\partial w^1_{11}} = 0.2 - 0.2 * (-0.028) = 0.2056.$

Similarly, $\frac{\partial L}{\partial w^1_{12}} = \frac{\partial L}{\partial h^2_1}\frac{\partial h^2_1}{\partial h^1_1}[\frac{\partial h^1_1}{\partial z^1_1}\frac{\partial z^1_1}{\partial w^1_{12}}]$.

Since you have already computed the values of the first three terms, you simply need to calculate the pending derivative term.

$\frac{\partial z^1_1}{\partial w^1_{12}} = \frac{\partial}{\partial w^1_{12}}(b^1_1 + w^1_{11}x_1 + w^1_{12}x_2) = x_2 =- 0.66$

Hence, this evaluates to $\frac{\partial L}{\partial w^1_{12}} = 1.17 * 0.30 * 0.484 * (1 - 0.484) * (-0.66) =- 0.058.$

Now,

$w^1_{12(updated)} = w^1_{12} - \eta\frac{\partial L}{\partial w^1_{12}} = 0.15 - 0.2 * (-0.058) = 0.1616.$

Similarly, $\frac{\partial L}{\partial b^1_1} = \frac{\partial L}{\partial h^2_1}\frac{\partial h^2_1}{\partial h^1_1}[\frac{\partial h^1_1}{\partial z^1_1}\frac{\partial z^1_1}{\partial b^1_1}]$.

Consider the last term on the RHS of the above equation:

$\frac{\partial z^1_1}{\partial b^1_1} = \frac{\partial}{\partial b^1_1}(b^1_1 + w^1_{11}x_1 + w^1_{12}x_2) = 1$

Hence, this evaluates to $\frac{\partial L}{\partial b^1_1} = 1.17 * 0.30 * 0.484 * (1 - 0.484) * 1 = 0.088.$

Now,

$b^1_{1(updated)} = b^1_1 - \eta\frac{\partial L}{\partial b^1_1} = 0.1 - 0.2 * (0.088) = 0.0824.$

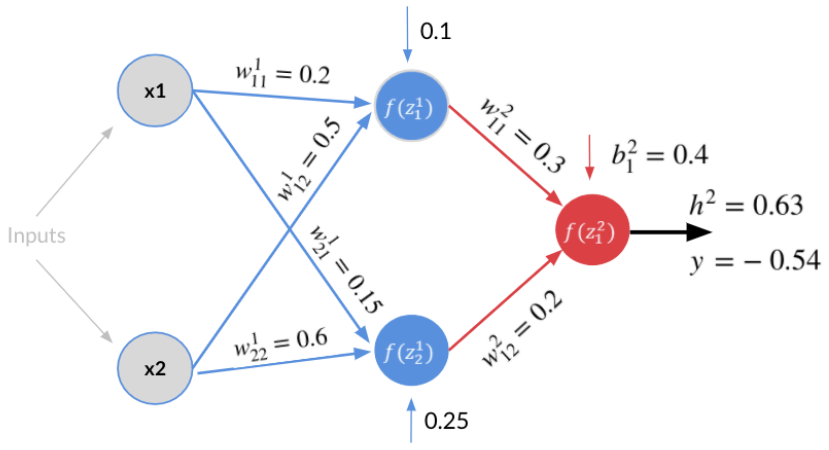
Hence, for the **first node**, let's compute the updated values of the weights and biases using **gradient descent** and a learning rate of 0.2 (η).

$w^1_{11(updated)} = w^1_{11} - \eta\frac{\partial L}{\partial w^1_{11}} = 0.2 - 0.2 * (-0.028) = 0.2056$

$w^1_{12(updated)} = w^1_{12} - \eta\frac{\partial L}{\partial w^1_{12}} = 0.15 - 0.2 * (-0.058) = 0.1616$

$b^1_{1(updated)} = b^1_1 - \eta\frac{\partial L}{\partial b^1_1} = 0.1 - 0.2 * (0.088) = 0.0824$

In the same manner, you calculate the weights and biases corresponding to the **second neuron** in the hidden layer.



Starting with finding the derivative of the loss function L with respect to w_{21}^1:

$$\frac{\partial L}{\partial w_{21}^1} = \frac{\partial L}{\partial h_1^2}\frac{\partial h_1^2}{\partial h_2^1}\frac{\partial h_2^1}{\partial w_{21}^1} = \frac{\partial L}{\partial h_1^2}\frac{\partial h_1^2}{\partial h_2^1}\left[\frac{\partial h_2^1}{\partial z_2^1}\frac{\partial z_2^1}{\partial w_{21}^1}\right]$$

You have already computed the first derivative term.

$$\frac{\partial L}{\partial h_1^2} = \frac{\partial}{\partial h_1^2}\frac{1}{2}(y - h_1^2)^2 =- (y - h_1^2)$$

$$\frac{\partial L}{\partial h_1^2} =- (- 0.54 - 0.63) = 1.17$$

Let's now compute the second, third and fourth ones.

1) $\frac{\partial h_1^2}{\partial h_2^1} = \frac{\partial}{\partial h_2^1}(b_1^2 + w_{11}^2 h_1^1 + w_{12}^2 h_2^1) = w_{12}^2 = 0.20$

2). $\frac{\partial h_2^1}{\partial z_2^1} = \sigma(z_2^1)(1 - \sigma(z_2^1)) = h_2^1(1 - h_2^1) = 0.424(1 - 0.424)$

3) $\frac{\partial z_2^1}{\partial w_{21}^1} = \frac{\partial}{\partial w_{21}^1}(b_2^1 + w_{21}^1 x_1 + w_{22}^1 x_2) = x_1 =- 0.32$

Also, for w_{22}^1 and b_2^1, the first three terms will remain the same. Only the last term will change. Hence, you will compute only the last term.

$$\frac{\partial z_2^1}{\partial w_{21}^1} = x_2 =- 0.66$$

$$\frac{\partial z_2^1}{\partial b_2^1} = 1$$

Hence, for the **second node**:

$$\frac{\partial L}{\partial w^1_{21}} = \frac{\partial L}{\partial h^2_1}\frac{\partial h^2_1}{\partial h^1_1}[\frac{\partial h^1_1}{\partial z^1_1}\frac{\partial z^1_2}{\partial w^1_{21}}] = 1.17 * 0.20 * 0.424 * (1 - 0.424) * (-0.32) =- 0.018$$

$$\frac{\partial L}{\partial w^1_{22}} = \frac{\partial L}{\partial h^2_1}\frac{\partial h^2_1}{\partial h^1_1}[\frac{\partial h^1_1}{\partial z^1_1}\frac{\partial z^1_2}{\partial w^1_{22}}] = 1.17 * 0.20 * 0.424 * (1 - 0.424) * (-0.66) =- 0.038$$

$$\frac{\partial L}{\partial b^1_2} = \frac{\partial L}{\partial h^2_1}\frac{\partial h^2_1}{\partial h^1_1}[\frac{\partial h^1_1}{\partial z^1_1}\frac{\partial z^1_2}{\partial b^1_2}] = 1.17 * 0.20 * 0.424 * (1 - 0.424) * 1 = 0.057$$

Now, computing the updated values of weights and biases using gradient descent and a learning rate of 0.2 (η):

$$w^1_{21(updated)} = w^1_{21} - \eta\frac{\partial L}{\partial w^1_{21}} = 0.5 - 0.2 * (-0.018) = 0.5036$$
$$w^1_{22(updated)} = w^1_{22} - \eta\frac{\partial L}{\partial w^1_{22}} = 0.6 - 0.2 * (-0.038) = 0.6076$$
$$b^1_{2(updated)} = b^1_2 - \eta\frac{\partial L}{\partial b^1_2} = 0.25 - 0.2 * (0.057) = 0.2386$$

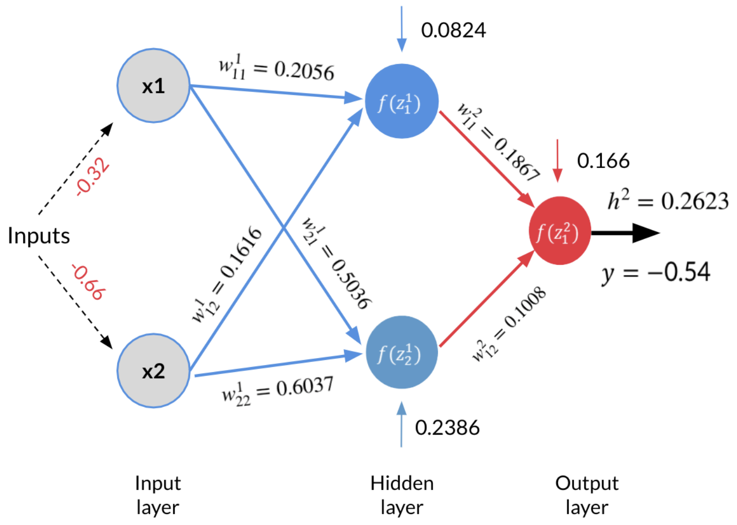
Given below are the new values for weights and biases after one step of gradient descent for the hidden and the output layers.

$$W^1_{(updated)}, W^2_{(updated)} = \begin{bmatrix} w^1_{11(updated)} & w^1_{12(updated)} \\ w^1_{21(updated)} & w^1_{22(updated)} \end{bmatrix}, \begin{bmatrix} w^2_{11(updated)} \\ w^2_{12(updated)} \end{bmatrix} = \begin{bmatrix} 0.2056 & 0.1616 \\ 0.5036 & 0.6076 \end{bmatrix}, \begin{bmatrix} 0.1867 \\ 0.1008 \end{bmatrix}$$

$$b^1_{(updated)}, b^2_{(updated)} = \begin{bmatrix} b^1_{1(updated)} \\ b^1_{2(updated)} \end{bmatrix}, \begin{bmatrix} b^2_{1(updated)} \end{bmatrix} = \begin{bmatrix} 0.0824 \\ 0.2386 \end{bmatrix}, [0.166]$$

Forward Pass with Updated Parameters

Now, let's perform another **forward pass** and check if performing backpropagation and updating the weights and biases once has helped in reducing the loss.



You can see that the loss function computed with the updated weights and biases is lower than earlier, which is what we want. By repeatedly performing backpropagation to get optimum values of weights and biases, you can continue reducing the loss. This, eventually, will help you obtain the predicted output that is as close as possible to the actual expected output. This is how a neural network learns using backpropagation.

Algorithm to Train a Neural Network

The pseudocode/pseudo-algorithm to train a neural network is given as follows:

Point 1: Initialise h0 with the inputs

Forward Propagation - Make a Prediction

Point 2: For each layer, compute the cumulative input and apply the non-linear activation function on each neuron of each layer to get the prediction.

Point 3: For classification, get the probabilities of the observation belonging to a class. For regression, compute the numerical output.

Point 4: Assess the performance of the neural network through a loss function, for example, a cross-entropy loss function for classification and RMSE for regression.

Backpropagation - Update the Model Parameters to Reduce Loss

Point 5: Consider the output neuron and compute the derivative of the loss function.

Point 6: Compute the derivative of the loss function with respect to the weights in the output layer.

Point 7: From the last layer to the first layer, for each layer, compute the gradient of the loss function with respect to the weights at each layer and all the intermediate gradients.

Updating the Model Parameters Using an Optimisation Algorithm such as Gradient Descent

Point 8: Once all the gradients of the loss with respect to the weights and biases are obtained, use the gradient descent update equation to update the values of the weights and biases.

Repeat This Process Until the Model Gives Acceptable Predictions

Point 9: Repeat the process for a specified number of iterations or until the predictions made by the model are acceptable.

This algorithm is used to train a neural network.