

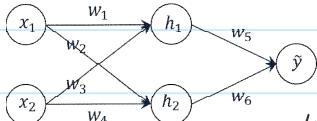
# MLP with Backpropagation

Monday, September 19, 2022 08:39

For the glory of God

## Getting started

- This hand-written note provides a simple example to understand the process of the Backpropagation algorithm.
- The example entails the following Multi-Layer Perception (MLP) model structure.



; where  $\hat{y}$  represents a predicted value

↳ Here, we will use the linear function as an activation function.

- Assumption related to input-out mapping  $s(x_1, x_2) = (2, 3)$  ; Input

$\bar{s} = 1$  ; Actual output

Difference (Actual - Predicted)  
↑

- In the subsequent sections, we will find the best weight parameters in a way that an error is minimized.

- To achieve the goal (i.e., finding weight parameters), we'll implement the following steps :

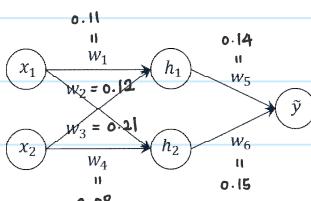
- STEP 1. Initialize weight parameters
- STEP 2. Perform forward propagation
- STEP 3. Calculate an error
- STEP 4. Conduct backpropagation
- STEP 5. Iterate the steps (STEP 2 ~ STEP 4) until it meets a user-defined criterion.

e.g.,  $\begin{cases} \text{number of iterations} \\ \text{error} \leq 10^{-6} \end{cases}$

## STEP 1. Initialization

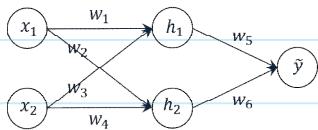
- The algorithm typically starts by initializing a set of randomly generated weight parameters.
- Keep in mind that initialization with zero values is not recommended as hidden units become identical.
- In this example, let us suppose that we initialize weight parameters shown in the Table below.

Weight Parameter	Value
$w_1$	0.11
$w_2$	-0.21 0.12
$w_3$	0.12 0.21
$w_4$	0.08
$w_5$	0.14
$w_6$	0.15



## STEP 2. Forward propagation

- We will see how the process of Forward propagation is performed, especially for the first iteration.



- $h_1 = x_1 w_1 + x_2 w_3$  ; Inputs are multiplied by the initialized weights.  
 $= (2 \times 0.11) + (3 \times 0.21)$   
 $= 0.85$  ; This result is passed forward to the next layer (i.e., output layer)

In a similar way, when it comes to  $h_2$ , we have :

$$h_2 = x_1 w_2 + x_2 w_4$$

$$= (2 \times 0.12) + (3 \times 0.08)$$

$$= 0.48$$

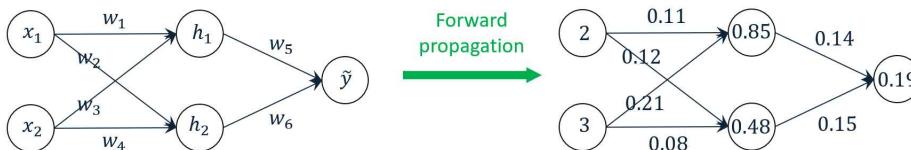
Thus, in terms of  $\tilde{y}$ , we have :

$$\tilde{y} = h_1 w_5 + h_2 w_6$$

$$= (0.85 \times 0.14) + (0.48 \times 0.15)$$

$$= 0.19$$

In Summary, through the process of Forward propagation, we have the following results :



### STEP 3. Error calculation

- The following squared error function is generally used to calculate an error between the actual output and the predicted one :

$$\text{Error} = \frac{1}{2} \sum_{i=1}^n (\tilde{y} - \bar{y})^2$$

; where   
 $\tilde{y}$  represents the predicted value  
 $\bar{y}$  refers to the actual value

- For this example, we will then have :

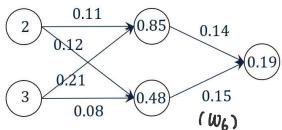
$$\text{Error} = \frac{1}{2} (0.19 - 1)^2$$

; Note that this is an error after the **first** iteration.

Forward propagation only at this point

### STEP 4. Backpropagation

- The Backpropagation algorithm allows us to update weight parameters in a way that an error is minimized.
- Particularly, the Gradient Descent algorithm is utilized for minimizing an error function.
- We will see how the process of Backpropagation is conducted, especially for the first iteration.



; Note that the actual value is equal to 1.

Suppose that we would like to update the weight parameter ( $w_6$ ).

- Then, we are going to leverage the Gradient Descent algorithm to achieve the objective (i.e., updating  $w_6$ )

$$w_{6,\text{new}} = w_{6,\text{old}} - \alpha \frac{\partial \text{Error func}}{\partial w_6} ; \text{ where } \alpha \text{ represents a learning rate (or a step size)}$$

- Here, let us take a deep dive into the term  $\frac{\partial \text{Error func}}{\partial w_6}$  more specifically;

↳ The derivative of the error function with respect to  $w_6$

; Using the chain rule, we have

$$\begin{aligned} \frac{\partial \text{Error}}{\partial w_6} &= \frac{\partial \text{Error func}}{\partial \text{Prediction}} \times \frac{\partial \text{Prediction}}{\partial w_6} ; \text{ where Prediction} = \hat{y} \\ &= \frac{\partial \left( \frac{1}{2} (\text{prediction} - \text{actual})^2 \right)}{\partial \text{Prediction}} \times \frac{\partial ((x_1 w_1 + x_2 w_3) w_5 + (x_1 w_2 + x_2 w_4) w_6)}{\partial w_6} \\ &= \left[ 2 \cdot \frac{1}{2} (\text{Prediction} - \text{actual}) \right] \cdot \frac{\partial (\text{Prediction} - \text{actual})}{\partial \text{Prediction}} \times \left[ \frac{\partial ((x_1 w_1 + x_2 w_3) w_5)}{\partial w_6} + \frac{\partial ((x_1 w_2 + x_2 w_4) w_6)}{\partial w_6} \right] \end{aligned}$$

This is something like  $[f(g(x))]' = f'(g(x)) \cdot g'(x)$  ; chain rule

$$= (\text{prediction} - \text{actual}) \cdot 1 \times [0 + (x_1 w_2 + x_2 w_4)]$$

$$= (\text{prediction} - \text{actual}) \cdot h_2 ; \text{ where } h_2 = x_1 w_2 + x_2 w_4$$

$$= \Delta h_2 ; \text{ where } \Delta = \text{Prediction} - \text{actual}$$

- To sum up, we have;

$$w_{6,\text{new}} = w_{6,\text{old}} - \alpha \Delta h_2 \quad (\text{Here, let's say that we smartly guess the learning rate} = 0.05)$$

$$= 0.15 - 0.05 \cdot (0.19 - 1) \cdot 0.48$$

$$\approx 0.17$$

- In a similar way, regarding  $w_5$ , we have;

$$w_{5,\text{new}} = w_{5,\text{old}} - \alpha \Delta h_1$$

$$= 0.14 - 0.05 (0.19 - 1) \cdot 0.85$$

$$\approx 0.17$$

- Unlike updating  $w_5$  and  $w_6$ , we need to pay attention to updating  $w_1, w_2, w_3$ , and  $w_4$  as they exist between input and hidden layers.

$$= 0.11$$

- Unlike updating  $w_5$  and  $w_6$ , we need to pay attention to updating  $w_1, w_2, w_3$ , and  $w_4$  as they exist between input and hidden layers.
- Assume that we are interested in updating the weight parameter ( $w_1$ ).
  - Likewise, we will leverage the Gradient Descent algorithm to update  $w_1$  in a way that the error function is minimized.

$$w_{1,new} = w_{1,old} - \alpha \frac{\partial \text{ErrorFunc}}{\partial w_1}$$

- Let's take a look at the component  $\partial \text{ErrorFunc} / \partial w_1$  more specifically:

$$\begin{aligned} \frac{\partial \text{ErrorFunc}}{\partial w_1} &= \frac{\partial \text{ErrorFunc}}{\partial \text{Prediction}} \cdot \frac{\partial \text{Prediction}}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1} ; \text{ using the chain rule} \\ &= \frac{\partial (\frac{1}{2} (\text{prediction} - \text{actual})^2)}{\partial \text{Prediction}} \cdot \frac{\partial (h_1 w_5 + h_2 w_6)}{\partial h_1} \cdot \frac{\partial (x_1 w_1 + x_2 w_3)}{\partial w_1} ; \text{ where prediction } (\hat{y}) = h_1 w_5 + h_2 w_6 \\ &= \left[ (\text{prediction} - \text{actual}) \cdot \frac{\partial (\text{prediction} - \text{actual})}{\partial \text{Prediction}} \right] \cdot (w_5 + 0) \cdot (x_1 + 0) \\ &= (\text{prediction} - \text{actual}) \cdot w_5 \cdot x_1 \\ &= \Delta w_5 x_1 \end{aligned}$$

- To sum up, we have:

$$\begin{aligned} w_{1,new} &= w_{1,old} - \alpha \Delta w_{1,old} x_1 \\ &= 0.11 - 0.05 \cdot (0.19 - 1) \cdot 0.14 \cdot 1 \\ &= 0.12 \end{aligned}$$

- In a similar way, we have:

$$\begin{aligned} w_{2,new} &= w_{2,old} - \alpha \Delta w_{2,old} x_1 \\ w_{3,new} &= w_{3,old} - \alpha \Delta w_{3,old} x_2 \\ w_{4,new} &= w_{4,old} - \alpha \Delta w_{4,old} x_2 \end{aligned}$$

Old Weight Parameter	Value	New Weight Parameter	Value
$w_{1,old}$	0.11	$w_{1,new}$	0.12
$w_{2,old}$	0.12	$w_{2,new}$	0.13
$w_{3,old}$	0.11	$w_{3,new}$	0.12
$w_{4,old}$	0.08	$w_{4,new}$	0.10
$w_{5,old}$	0.14	$w_{5,new}$	0.17
$w_{6,old}$	0.15	$w_{6,new}$	0.17

#### STEP 5. Iterations until convergence

• Basically, the algorithm iterates the steps (2 to 4) until it meets a convergence criterion.

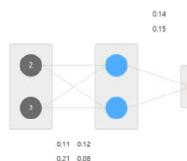
• Play with the following demo if you are interested :

<https://hmkcode.com/netflow/>

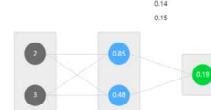
Note that this is not the squared error

but just the difference ↗

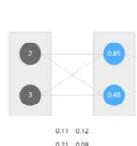
STEP 1. Weight initialization



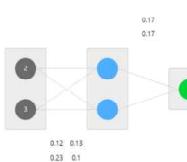
STEP 2. Forward propagation



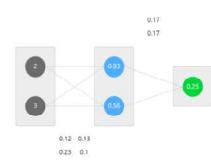
STEP 3. Calculating an error



STEP 4. Backpropagation



STEP 5. Iterations until convergence



N iterations →

