**CSE4205 – Neutral Network**

**Zamil-8B-483**

**Lecture 04 - Backpropagation in Neural Network**

**What is Backpropagation?**

* Backpropagation is a way of propagating the total loss back into the neural network to know how much of the loss every node is responsible for, and subsequently updating the weights.
* It is a process involved in training a neural network and is only used during the training phase of neural network
* It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration).
* Proper tuning of the weights allows for reduced error rates and makes the model reliable.

**How Backpropagation Works?**

Here's a step-by-step explanation of how the backpropagation algorithm works:

* **Forward Pass:**

- Input data moves through the neural network, layer by layer, producing predictions.

- Each layer applies weights and an activation function to generate an output.

* **Loss Calculation:**

- The model's output is compared to the actual target values to compute the loss or error.

- Common loss functions like MSE for regression or Cross-Entropy Loss for classification are used.

* **Backward Pass (Backpropagation):**

- *Gradients of the loss with respect to the weights are calculated*.

- The ***chain rule of calculus*** is employed to determine how changes in each weight affect the loss.

* **Weight Update:**

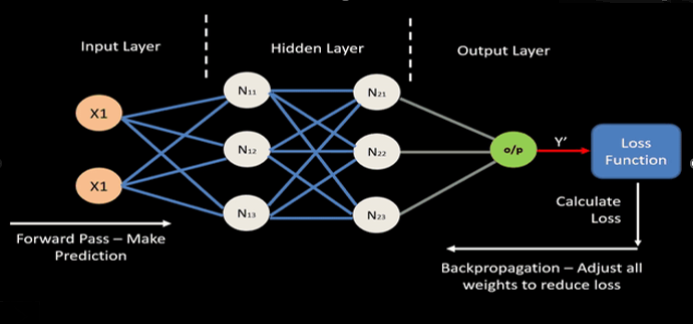
- Gradients guide updates to the weights in the direction that minimizes the loss.

- Learning rate controls the size of these updates.

* **Iteration:**

- Steps 1-4 are repeated iteratively with batches of training data over multiple epochs.

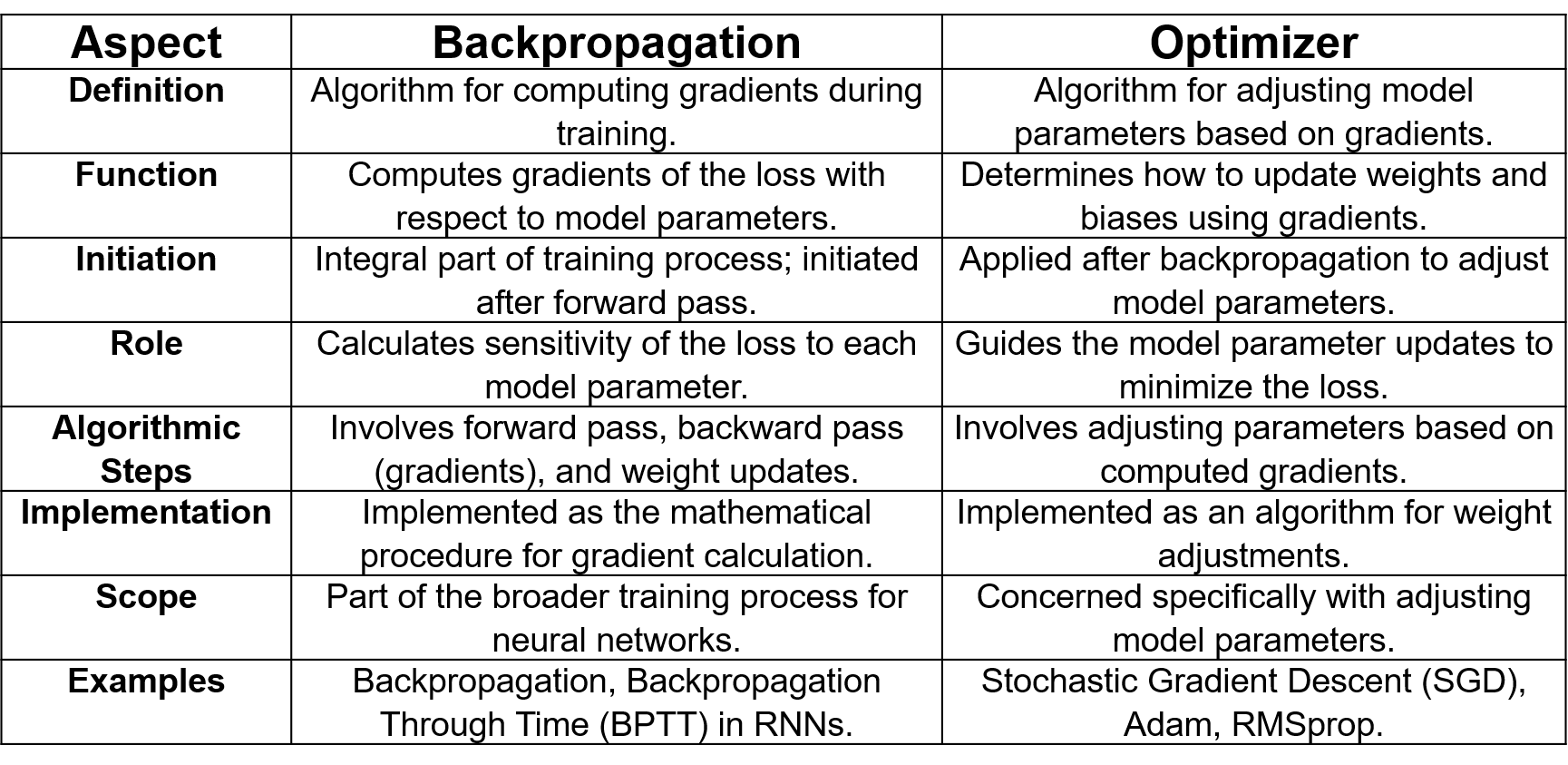
- The objective is to find optimal weights that minimize the loss, enabling accurate predictions on new data.



**Backpropagation (Don’t get confused !!)**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Backpropagation** | **Feed Forward** |
| **Function** | Computes gradients of the loss with respect to model parameters. | Processes input data through the network to generate predictions. |
| **Initiation** | Part of the training process; follows the forward pass. | Initiates the neural network's prediction process. |
| **Calculation Direction** | Propagates errors backward from output to input layers. | Propagates input forward through the network. |
| **Role** | Adjusts weights to minimize loss during training. | Generates predictions based on current weights. |
| **Algorithmic Steps** | Involves both forward and backward passes. | Involves passing input data through layers sequentially. |
| **Implementation** | Implemented as a procedure for gradient computation and weight updates. | Implemented as the process of passing data through the network layers. |
| **Scope** | Part of the training phase in the overall neural network workflow. | Part of the neural network's prediction process. |
| **Examples** | Backpropagation, Backpropagation Through Time (BPTT) in RNNs. | Initial phase in both training and prediction processes. |

**Backpropagation VS Optimizer**



**Types of Backpropagation**

There are two types of backpropagation networks.

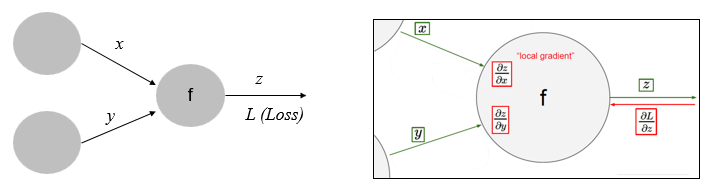
* **Static backpropagation:**Static backpropagation is a network designed to map static inputs for static outputs. These types of networks are capable of solving static classification problems such as OCR (Optical Character Recognition).
* **Recurrent backpropagation:** Recursive backpropagation is another network used for fixed-point learning. Activation in recurrent backpropagation is feed-forward until a fixed value is reached. Static backpropagation provides an instant mapping, while recurrent backpropagation does not provide an instant mapping.

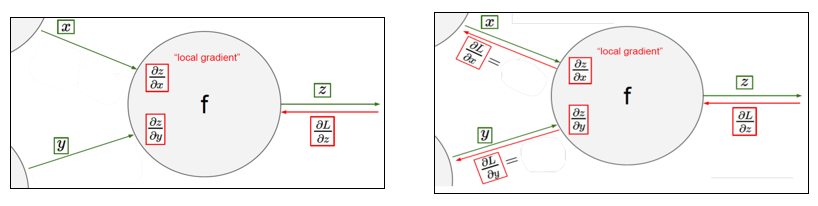
**Chain Rule in Backpropagation**

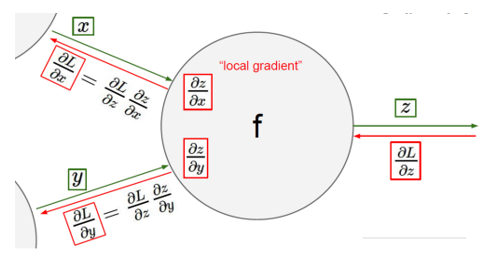
* The chain rule is a fundamental concept in calculus that describes how to find the derivative of a composite function.
* In backpropagation, the chain rule is applied to compute the derivative of the loss with respect to the weights in each layer.
* The chain rule allows us to decompose the overall derivative into products of derivatives for each intermediate step in the computation.
* Mathematically, for a neural network layer, the chain rule can be expressed as follows:

=

* The first term on the right side is the derivative of the loss with respect to the output of the layer.
* The second term is the derivative of the layer's output with respect to its weights.

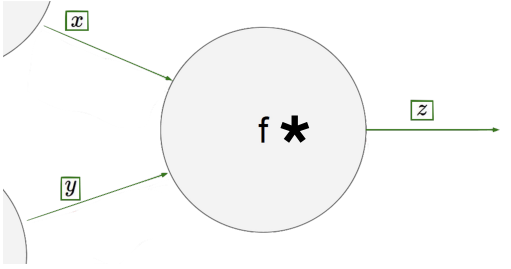




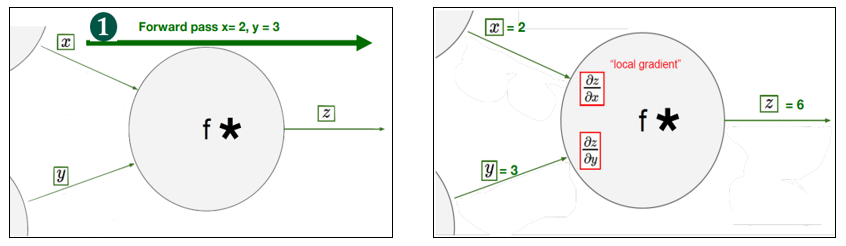


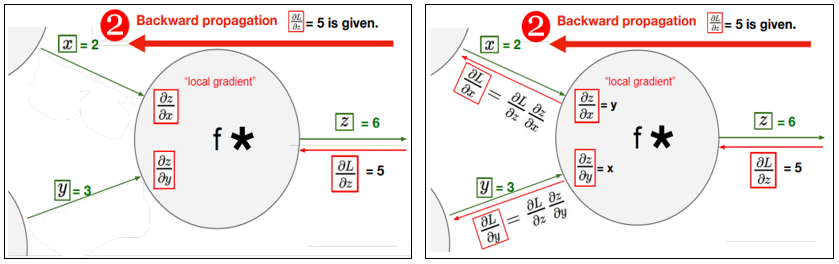
**Example:**

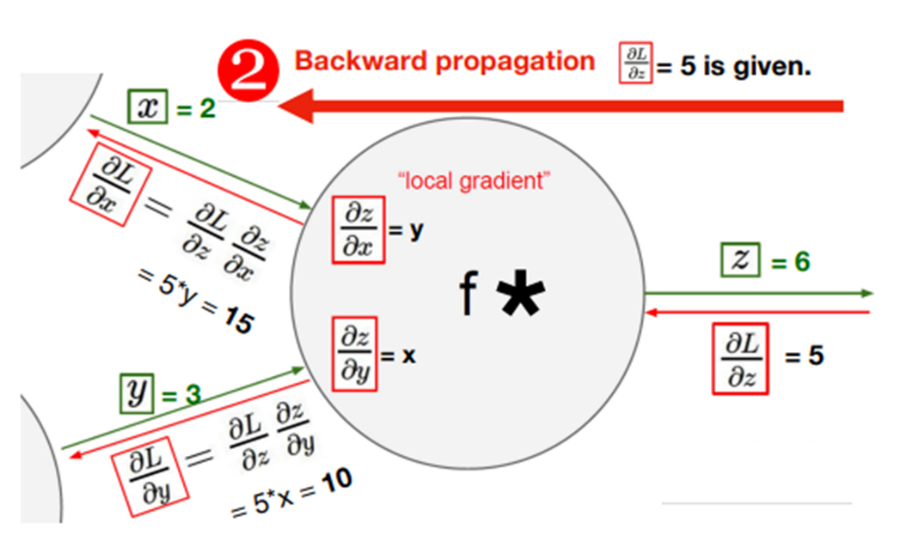
Let, x= 2, y=3 and = 5 what will be the loss with respect to x and y.



**Solution:**





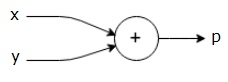


**Computational Graph**

* A computational graph is defined as a directed graph where the nodes correspond to mathematical operations.
* Computational graphs are a way of expressing and evaluating a mathematical expression.
* For example, here is a simple mathematical equation-

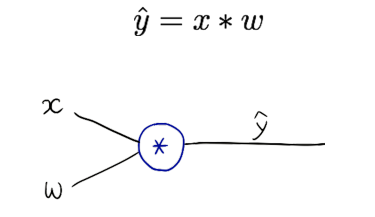
**p=x+y**

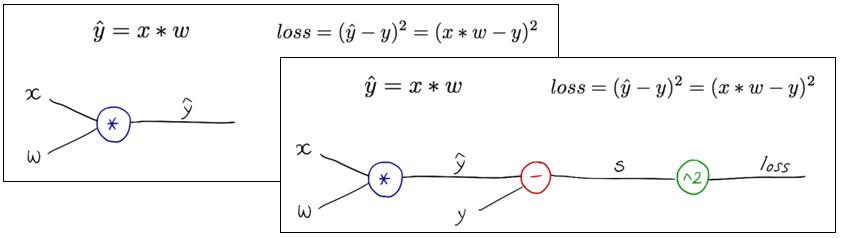
* We can draw a computational graph of the above equation as follows.

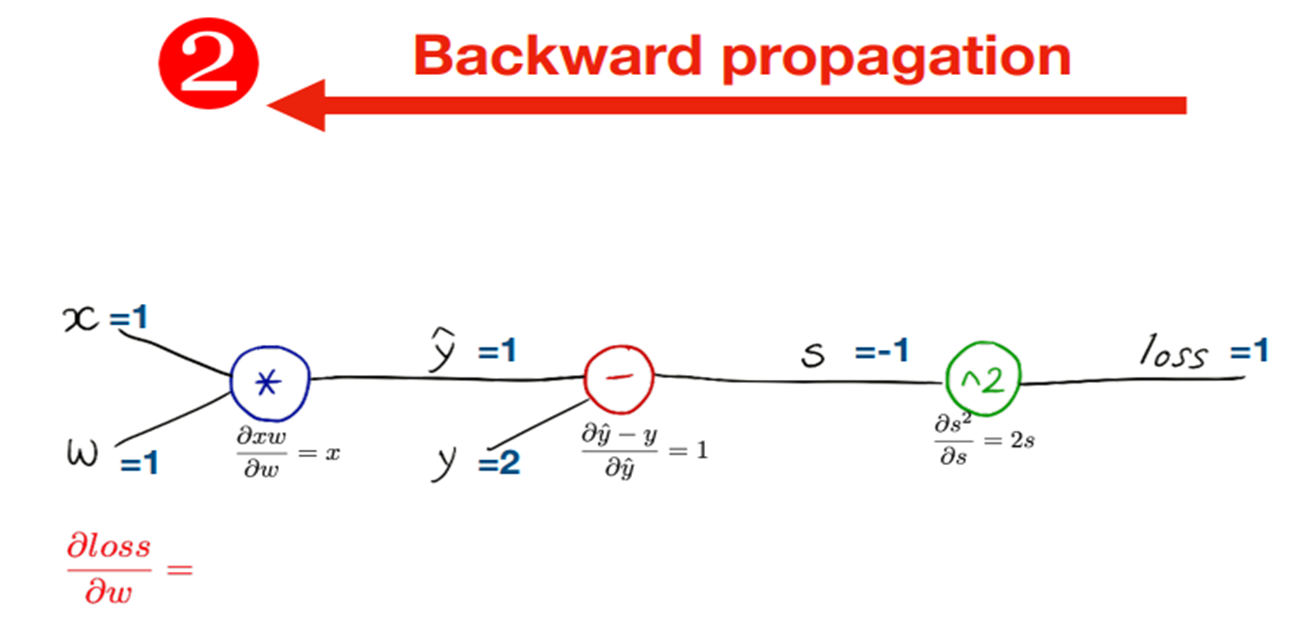
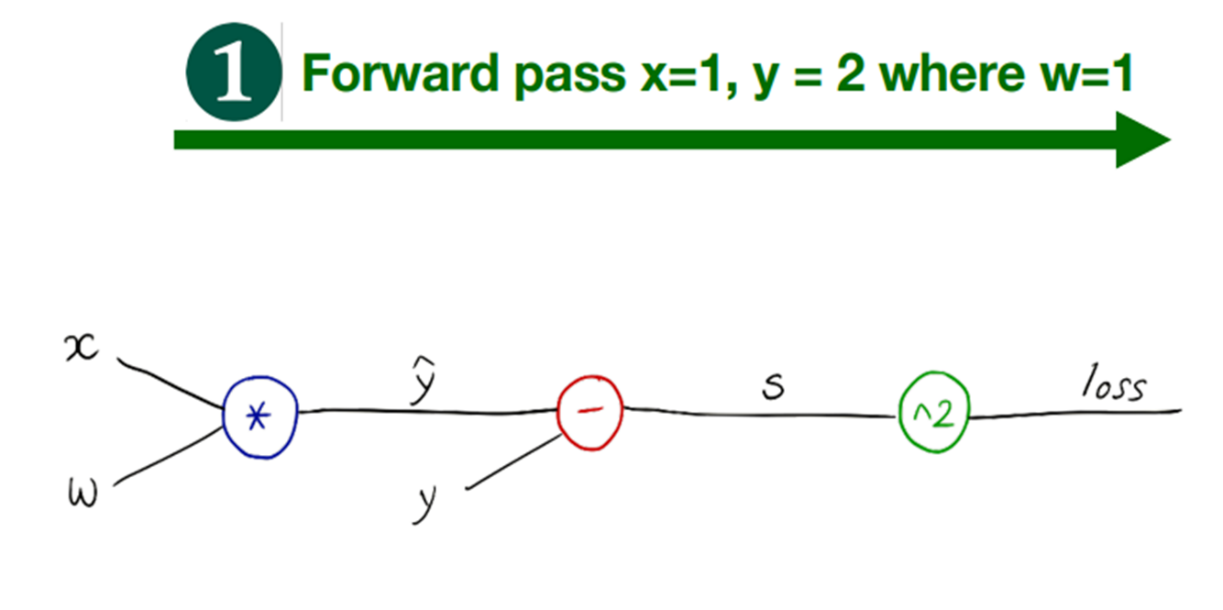


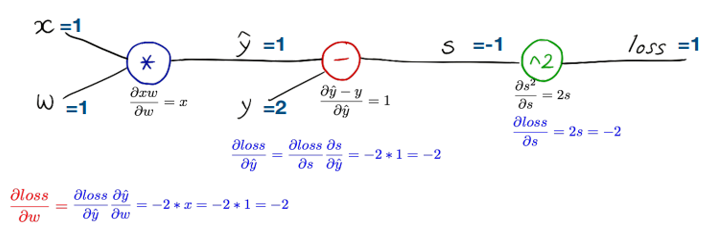
**Computational Graph of Back Propagation**

Let the model for a dataset, = x \* w

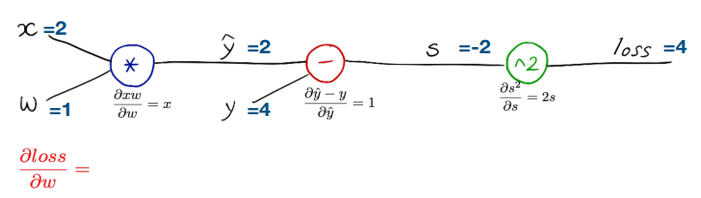








**Exercise 4-1:** x = 2, y=4, w= I



**Exercise 4-2:** x = l, y=2, w: l, b=2



**Advantages of Using Backpropagation in Neural Network**

* No previous knowledge of a neural network is needed, making it easy to implement.
* It’s straightforward to program since there are no other parameters besides the inputs.
* It doesn’t need to learn the features of a function, speeding up the process.
* The model is flexible because of its simplicity and applicable to many scenarios.

**Limitation of Using Backpropagation in Neural Network**

* Training data can impact the performance of the model, so high-quality data is essential.
* Noisy data can also affect backpropagation, potentially tainting its results.
* It can take a while to train backpropagation models and get them up to speed.
* Backpropagation requires a matrix-based approach, which can lead to other issues.