Green University of Bangladesh

Department of Computer Science and Engineering

The Backpropagation Algorithm

An Animated Explanation

Course: **Machine Learning**

Submitted by:

Ibrahim Refat
ID: 221902333
Section: 221D9

Submitted to:

Dr. Muhammad Abul Hasan
Associate Chairperson

September 1, 2025

Contents

Abstract **2**

1 Introduction **3**

2 Project Objective **4**

3 The Backpropagation Cycle **5**
3.1 Step 1: The Forward Pass 5
3.2 Step 2: Error Calculation 5
3.3 Step 3: The Backward Pass 5
3.4 Step 4: Weight Update 5

4 Implementation **7**

5 Figures **8**

6 Challenges **10**

7 Conclusion **11**

References **12**

Abstract

This report details a project focused on creating an animated visualization of Backpropagation. How Backpropagation works at each step, and the calculation is shown by animation.

The animation will break down this learning process into four clear steps:

1. **Forward Pass:** The network makes a guess or prediction.
2. **Error Calculation:** The system figures out how wrong the prediction was.
3. **Backward Pass:** The error is sent back through the network to identify which weights need to be changed.
4. **Weight Update:** The weights are slightly adjusted to reduce the error, using a method called gradient descent.

Chapter 1

Introduction

Back Propagation is also known as "Backward Propagation of Errors" is a method used to train neural network . Its goal is to reduce the difference between the model's predicted output and the actual output by adjusting the weights and biases in the network.

It works iteratively to adjust weights and bias to minimize the cost function. In each epoch the model adapts these parameters by reducing loss by following the error gradient. It often uses optimization algorithms like gradient descent or stochastic gradient descent. The algorithm computes the gradient using the chain rule from calculus allowing it to effectively navigate complex layers in the neural network to minimize the cost function.

Chapter 2

Project Objective

This project aims to create an animated video that visually demonstrates the step-by-step learning process of a neural network through backpropagation.

Chapter 3

The Backpropagation Cycle

Training a neural network is a cycle that repeats itself over and over. The animation will show this cycle in four main steps.

3.1 Step 1: The Forward Pass

The cycle starts with the **forward pass**. In this step, the network takes a piece of data and makes a prediction using its current weights. This is how information flows forward through the network.

3.2 Step 2: Error Calculation

After the network makes a guess, it compares it to the correct answer to find the **error**. The bigger the number, the bigger the mistake. A common way to measure this is with a formula like this one for a single prediction:

$$E = \frac{1}{2}(y_{\text{true}} - y_{\text{pred}})^2 \tag{3.1}$$

The goal of the network is to make this error value as small as possible.

3.3 Step 3: The Backward Pass

This is the most important part of backpropagation. The algorithm now works backward, sending the error signal through the network to figure out how much each weight contributed to the mistake. This uses a rule from calculus called the **chain rule** to find the direction of change needed for each weight, or its "gradient," represented as $\frac{\partial E}{\partial w}$.

3.4 Step 4: Weight Update

With the "blame" assigned, the final step is to adjust the weights to improve the network's next guess. This is done with an algorithm called **Gradient Descent**. The gradient tells the network the direction to change to make the error bigger, so it moves in the opposite direction to make the error smaller.

The update rule for each weight is:

$$w_{\text{new}} = w_{\text{old}} - \eta \frac{\partial E}{\partial w_{\text{old}}} \tag{3.2}$$

Here, (eta) is the **learning rate**, which controls how big of a step the network takes. This simple adjustment is applied to every weight, completing one full learning cycle. This whole process is then repeated many times until the network is well-trained.

Chapter 4

Implementation

The complete source code for the Manim animation can be found on GitHub.

```
https://github.com/ibrahimrifats/ML-with-Manim/blob/main/
   backpropagation.py
```

Listing 4.1: GitHub Repository URL

The project can be accessed via the hyperlink: https://github.com/ibrahimrifats/ML-with-Manim

Chapter 5

Figures

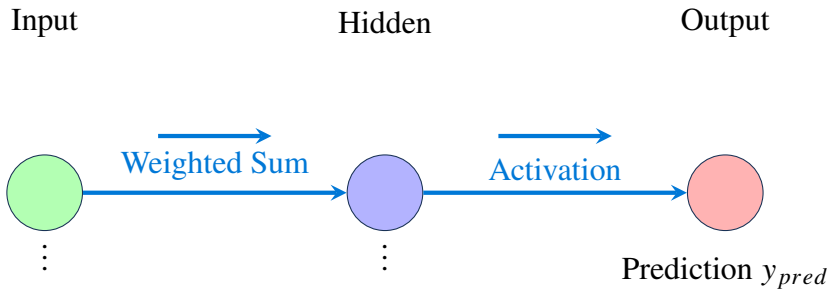

Figure 5.1: Step 1: Forward Pass. Data flows from input to output to generate a prediction.

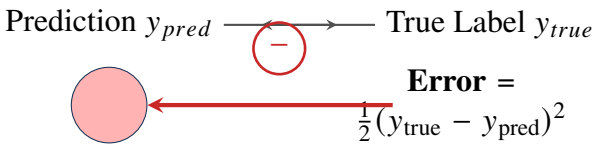

Figure 5.2: Step 2: Error Calculation. The network's prediction is compared to the true label to calculate the error.

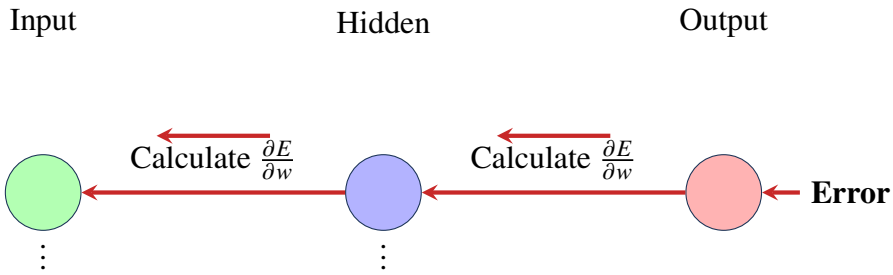

Figure 5.3: Step 3: Backward Pass. The error is propagated backward using the Chain Rule to calculate the gradient for each weight.

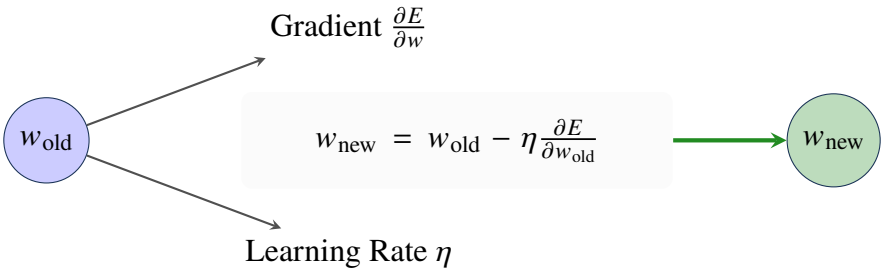

Figure 5.4: Step 4: Weight Update. The old weight is adjusted using the calculated gradient and the learning rate via Gradient Descent.

Chapter 6

Challenges

The main challenge in this project was learning the Manim library. Making accurate and well-timed animations to represent complex mathematical processes like backpropagation took a lot of effort, especially with limited time.

Chapter 7

Conclusion

This project effectively simplifies the intricate process of training neural networks into a clear, four-step animated narrative centered on the Backpropagation algorithm. By visualizing the progression from prediction (the forward pass) to correction (which includes error calculation, the backward pass, and weight updates), the animation brings abstract concepts to life, making them more understandable.

References

1. Manim Community Documentation. (2023). Retrieved from https://docs.manim.community/en/stable/

2. GeeksforGeeks. (n.d.). *Multi-Layer Perceptron Learning in TensorFlow*. Retrieved from https://www.geeksforgeeks.org/deep-learning/multi-layer-perceptron-learning-in