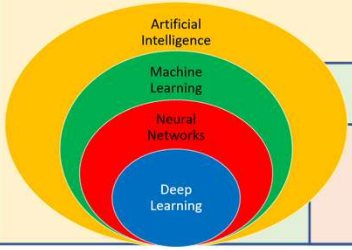


Artificial Intelligence (AI)

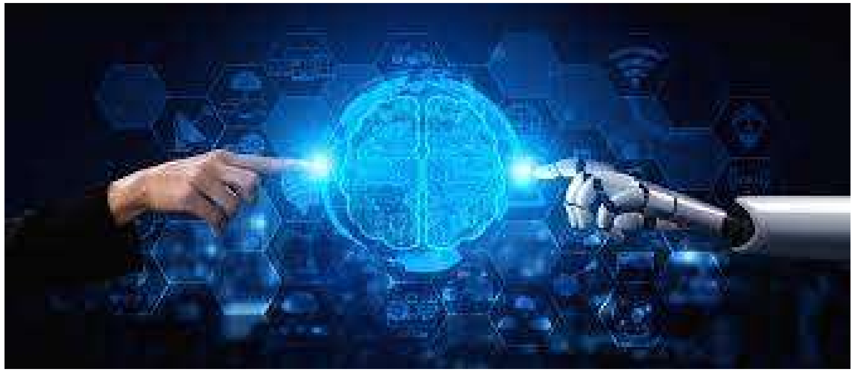
Machine Learning (ML)

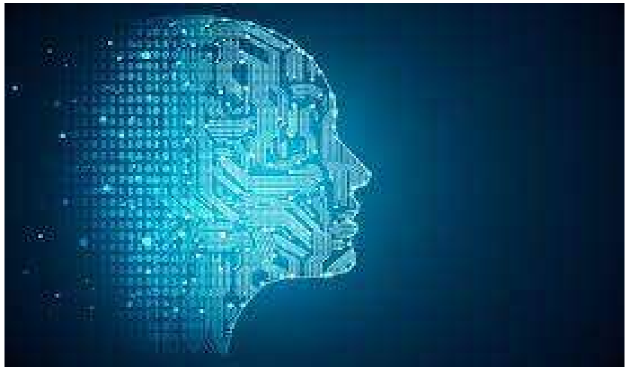
Neural Networks (NNs)

Deep Learning (DL)

Artificial Neural Networks and Deep Learning

Dr. Rastgoo





Outline

- Introduction to Backpropagation Algorithm
- Introduction to Neural Networks (NNs).
- Introduction to Deep Learning (DL).
- Programming.

Backpropagation Algorithm

- At its core, a neural network is an algorithm that was designed to learn patterns in real-life data and make predictions.
- An important part of this learning is done using the backpropagation algorithm.
- The backpropagation attempts to correct errors at each layer to make a better prediction.
- We can do this by fine-tuning the weights of the neural network.
- This fine-tuning mechanism is done by the partnership of gradient descent and backpropagation algorithms.

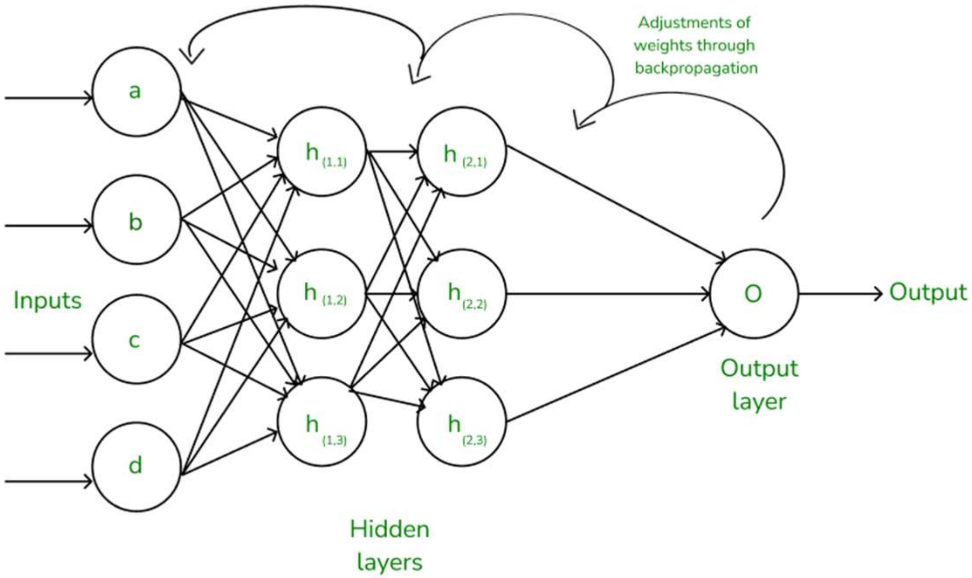
The point of backpropagation is based on the intuition that we can improve the final choice (output) by correcting every small decision that leads to it.

Backpropagation Algorithm

- Backpropagation is an iterative algorithm, that helps to minimize the cost function by determining which weights and biases should be adjusted.

- During every epoch, the model learns by adapting the weights and biases to minimize the loss by moving down toward the gradient of the error. Thus, it involves the two most popular optimization algorithms, such as gradient descent or stochastic gradient descent.

- Computing the gradient in the backpropagation algorithm helps to minimize the cost function and it can be implemented by using the mathematical rule called chain rule from calculus to navigate through complex layers of the neural network.



Advantages of Using the Backpropagation Algorithm in Neural Networks

- Backpropagation, a fundamental algorithm in training neural networks, offers several advantages that make it a preferred choice for many machine learning tasks. Here, we discuss some key advantages of using the backpropagation algorithm:
- ✓ Ease of Implementation: Backpropagation does not require prior knowledge of neural networks, making it accessible to beginners. Its straightforward nature simplifies the programming process, as it primarily involves adjusting weights based on error derivatives.

- ✓ Simplicity and Flexibility: The algorithm’s simplicity allows it to be applied to a wide range of problems and network architectures. Its flexibility makes it suitable for various scenarios, from simple feedforward networks to complex recurrent or convolutional neural networks.

Advantages of Using the Backpropagation Algorithm in Neural Networks

- ✓ Efficiency: Backpropagation accelerates the learning process by directly updating weights based on the calculated error derivatives. This efficiency is particularly advantageous in training deep neural networks, where learning features of a function can be time-consuming.

- ✓ Generalization: Backpropagation enables neural networks to generalize well to unseen data by iteratively adjusting weights during training. This generalization ability is crucial for developing models that can make accurate predictions on new, unseen examples.

- ✓ Scalability: Backpropagation scales well with the size of the dataset and the complexity of the network. This scalability makes it suitable for large-scale machine learning tasks, where training data and network size are significant factors.

How does backpropagation work?

- Once the inputs fed to the neural network pass through all the layers and make a prediction, we calculate a loss for it.

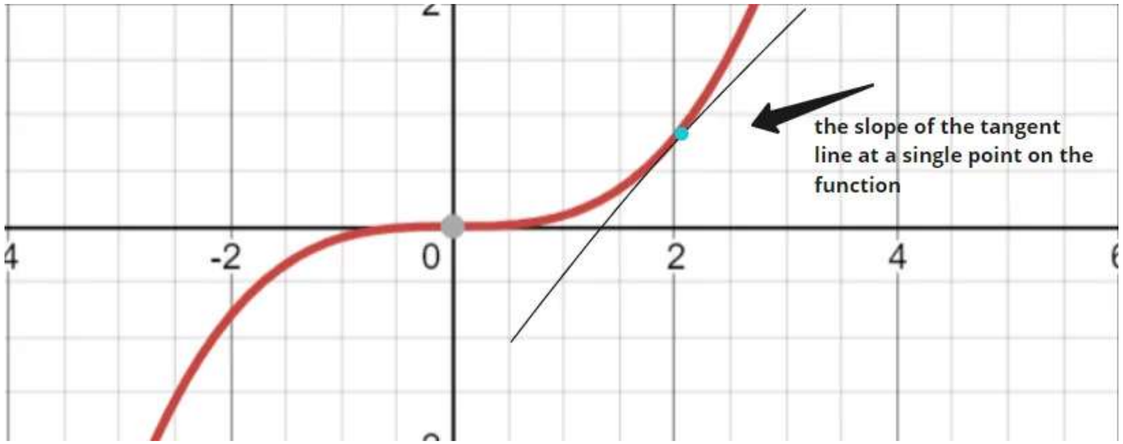
- The job of backpropagation is to then find how much each weight contributes to the final loss.

- It does this by computing the negative gradient of the loss with respect to each weight.

- Gradient descent now uses this information to update the weight to efficiently reduce the loss.

How does backpropagation work?

- Backpropagation finds the negative gradient uses two important mathematical functions from calculus :
- 1. Partial Derivative: In deep learning, we compute the partial derivative to find the slope of the loss function, i.e. the gradient.
- Partial derivative gives us the derivative of a function concerning a single variable.
- This allows us to find how much a change in a single weight affects the final loss.
- The higher the absolute magnitude of the weight, the more it affects the output.



How does backpropagation work?

- Backpropagation finds the negative gradient uses two important mathematical functions from calculus :

- 2. Chain Rule: The chain rule in mathematics is used to find the derivative of a composite function (a function of a function).

- In a neural network the output of each layer is a function of the output from the preceding layer.

- Hence we need the chain rule to find the gradient of each weight.

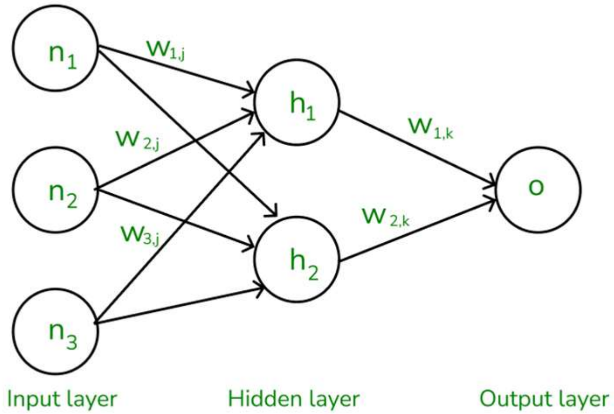
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} * \frac{\partial y}{\partial x}$$

How does backpropagation work?

- ➢ The Backpropagation algorithm works by two different passes, they are:

- ✓ Forward pass,
- ✓ Backward pass.

How does backpropagation work?



- In forward pass, initially the input is fed into the input layer.
- Since the inputs are raw data, they can be used for training our neural network.
- The inputs and their corresponding weights are passed to the hidden layer.
- The hidden layer performs the computation on the data it receives.
- If there are two hidden layers in the neural network, the output of the first hidden layer can be used as an input of the second hidden layer.
- Before applying it to the activation function, the bias is added.

How does backpropagation work?

- To the weighted sum of inputs, the activation function is applied in the hidden layer to each of its neurons.

- One such activation function that is commonly used is ReLU can also be used, which is responsible for returning the input if it is positive otherwise it returns zero. By doing this so, it introduces the non-linearity to our model, which enables the network to learn the complex relationships in the data.

- And finally, the weighted outputs from the last hidden layer are fed into the output to compute the final prediction, this layer can also use the activation function called the softmax function which is responsible for converting the weighted outputs into probabilities for each class.

How does backpropagation work?

- In the backward pass process shows, the error is transmitted back to the network which helps the network, to improve its performance by learning and adjusting the internal weights.

- To find the error generated through the process of forward pass, different commonly used methods, such as mean squared error, can be used.

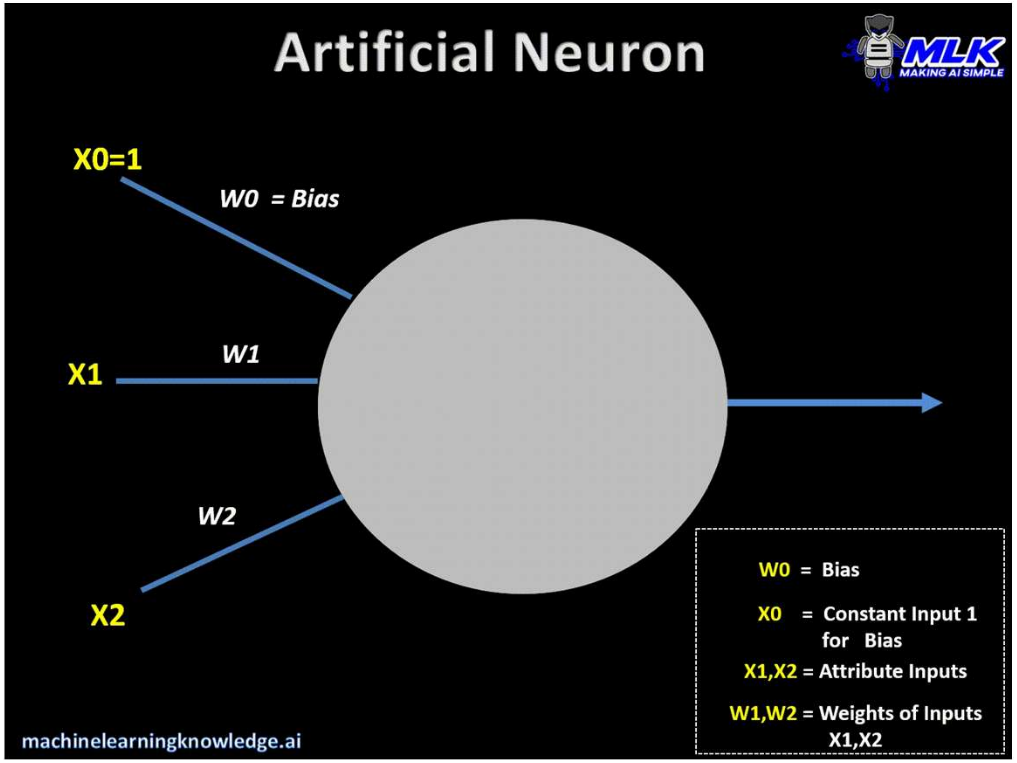
- Once we have done the calculation at the output layer, we then propagate the error backward through the network, layer by layer.

How does backpropagation work?

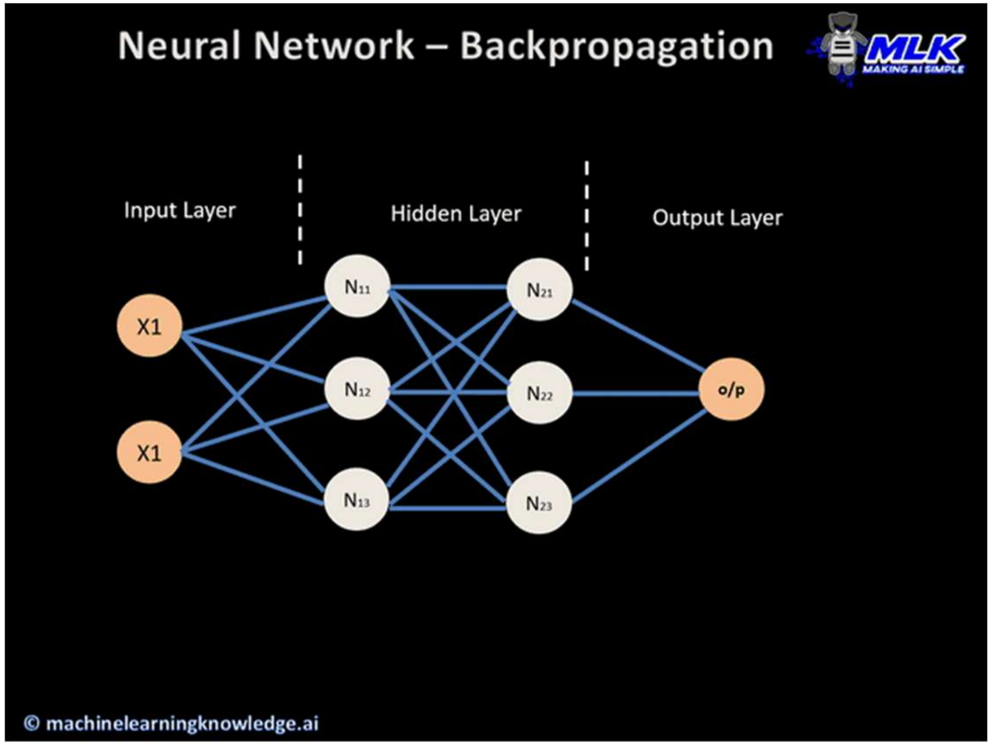
- The key calculation during the backward pass is determining the gradients for each weight and bias in the network. This gradient is responsible for telling us how much each weight/bias should be adjusted to minimize the error in the next forward pass. The chain rule is used iteratively to calculate this gradient efficiently.

- In addition to gradient calculation, the activation function also plays a crucial role in backpropagation, it works by calculating the gradients with the help of the derivative of the activation function.

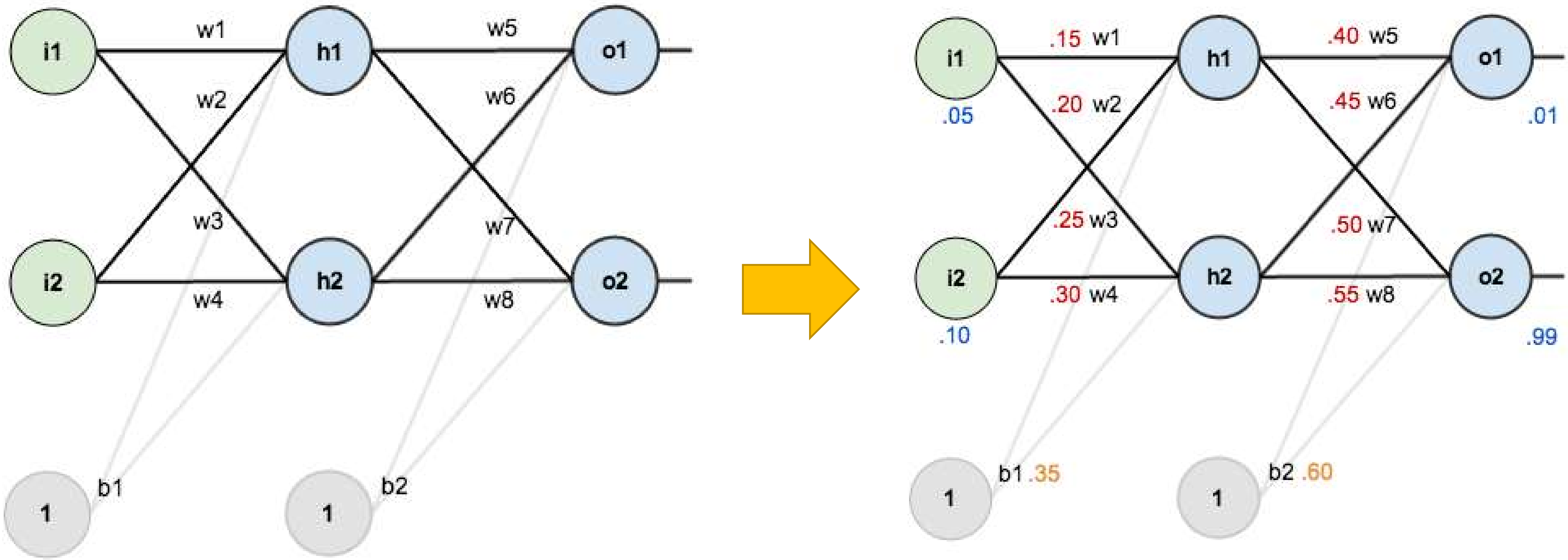
Backpropagation: Example



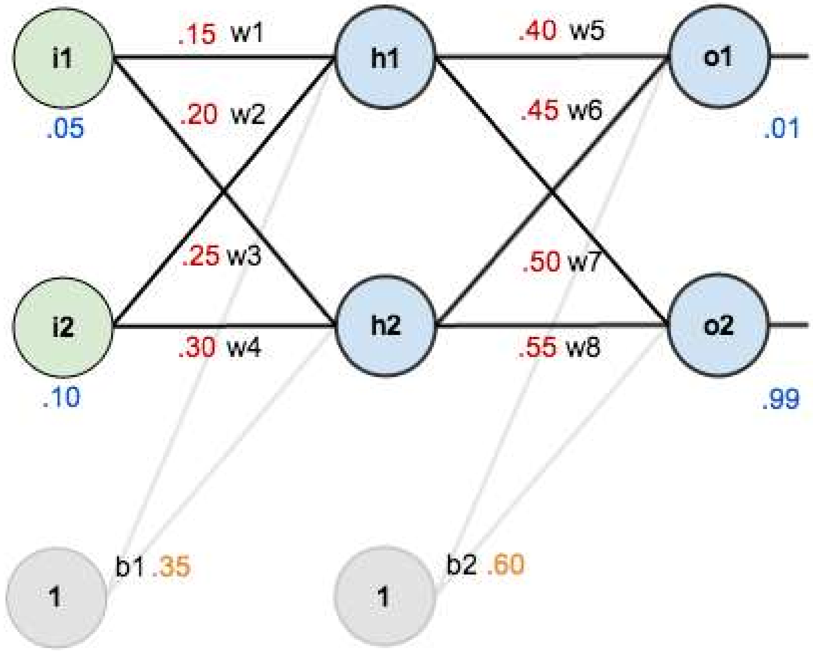
Backpropagation: Example



Backpropagation: Example



Backpropagation: Example



$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

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

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

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

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

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

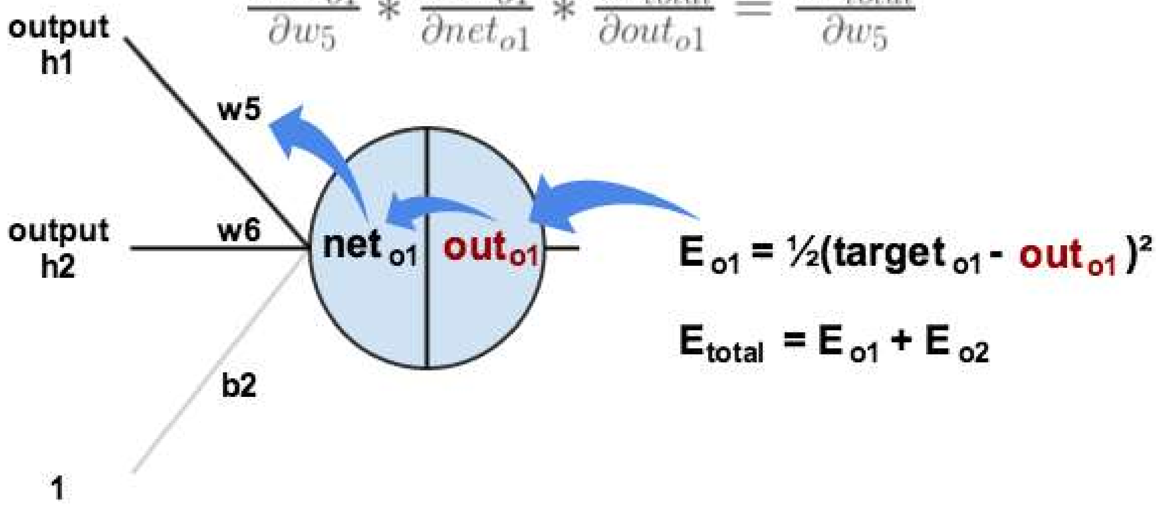
$$E_{total} = E_{o1} + E_{o2}$$

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

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5}$$

Backpropagation: Example

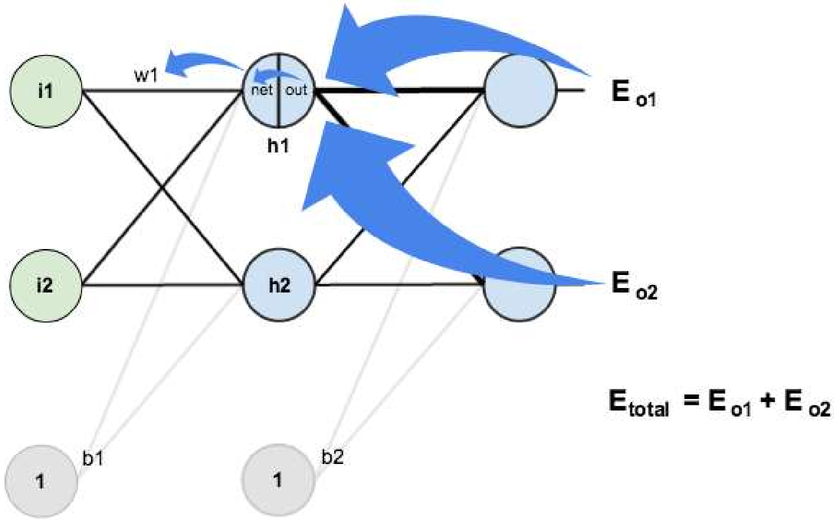
$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



Backpropagation: Example

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



Backpropagation: Example

- The key calculation during the backward pass is determining the gradients for each weight and bias in the network. This gradient is responsible for telling us how much each weight/bias should be adjusted to minimize the error in the next forward pass. The chain rule is used iteratively to calculate this gradient efficiently.

- In addition to gradient calculation, the activation function also plays a crucial role in backpropagation, it works by calculating the gradients with the help of the derivative of the activation function.