ENCS3340 - Artificial Intelligence

Artificial Neural Network

Outline

- Introduction and Motivation

- Neural Network Architecture
  - The Perceptron
  - MLPs
  - Multi-class Perceptron

- Training MLPs

- Choosing Network Structure
  - Depth vs Width
  - Expressive Power of MLPs

Artificial Neural Networks

- A neural network can be defined as a model of reasoning based on the human brain.
- The brain consists of a densely interconnected set of nerve cells, or basic information-processing units, called neurons.
- The human brain incorporates nearly 10 billion neurons and 60 trillion connections, synapses, between them.
- By using multiple neurons simultaneously, the brain can perform its functions much faster than the fastest computers in existence today.
- Each neuron has a very simple structure, but an army of such elements constitutes a tremendous processing power.
- A neuron consists of a cell body, soma, a number of fibers called dendrites, and a single long fiber called the axon.

Artificial Neural Networks

- An artificial neural network consists of a number of very simple processors, also called neurons, which are analogous to the biological neurons in the brain.
- The neurons are connected by weighted links passing signals from one neuron to another.
- The output signal is transmitted through the neuron’s outgoing connection.
- The outgoing connection splits into a number of branches that transmit the same signal.
- The outgoing branches terminate at the incoming connections of other neurons in the network.

Artificial Neural Networks

- ANNs can learn from training data and generalize to new situations and have high expressive power.
- Neural networks have become one of the major thrust areas recently in various AI tasks including pattern recognition, prediction, and analysis problems.
- In many problems they have established the state of the art, often exceeding previous benchmarks by large margins.
- An ANN is usually characterized by
  - The way the neuruons are connected to each other.
  - The method that is used for the determinations of the connection strengths or weights.
  - The activation function.

Brain vs Computer

- There are approximately 10 billion neurons in the human cortex, compared with 10’s of thousands of processors in the most powerful parallel computers
- Each biological neuron is connected to several thousands of other neurons, similar to the connectivity in powerful parallel computers
- The typical operating speeds of biological neurons is measured in milliseconds (10-3 s), while a silicon chip can operate in nanoseconds (10-9 s)
- The human brain is extremely energy efficient, using approximately 10-16 joules per operation per second, whereas the best computers today use around 10-6 joules per operation per second

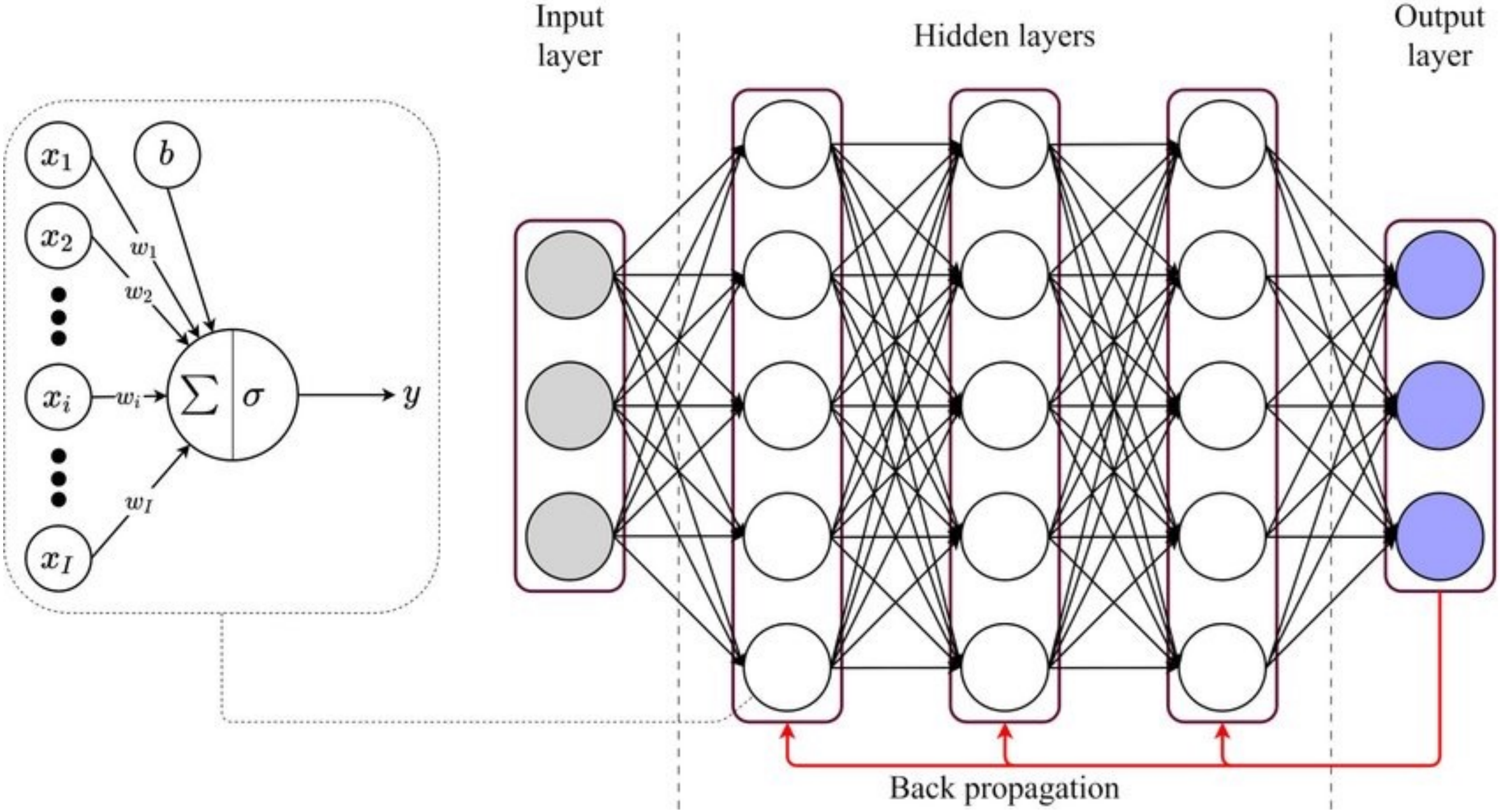
Brain vs Computer

- Tasks that are easy for brains are not easy for computers and vice versa
- Brains
  - Recognizing faces
  - Retrieving information based on partial descriptions
  - Organizing information (the more information the better the brain operates)
- Computers
  - Arithmetic
  - Deductive logic
  - Retrieving information based on arbitrary features
- Brains must operate very differently from conventional computers

Feedforward Neural Networks (FNNs)

- Feedforward neural networks (FNNs), also known as Multi-Layer Perceptrons (MLPs), are a type of artificial neural network that consists of multiple layers of neurons interconnected in a feedforward manner.
- Information flows in one direction, from the input layer through one or more hidden layers to the output layer.
- The neurons are organized into layers, where each neuron in a layer connects to every neuron in the next layer.
- Each connection is associated with a weight, and each neuron has an associated bias.
- The weights of the network are adjusted during training to minimize the error between the predicted output and the desired output. This is done using a variety of algorithms, such as backpropagation.
- FNNs are a very versatile type of neural network and can be used for a wide variety of tasks, including: Classification, Regression, Universal Function Approximators, Feature Learning,…

FNNs General Architecture



Main Components of FNNs

- **Input Layer:** The input layer consists of neurons that receive input features.
  - The number of neurons in this layer corresponds to the number of input features.
  - Each neuron in the input layer represents a specific feature of the input data.
- **Hidden Layers:** Hidden layers are intermediary layers between the input and output layers.
  - Each hidden layer consists of multiple neurons that process the information from the previous layer and pass it on to the next layer.
  - The number of hidden layers and the number of neurons in each layer are hyperparameters that you can adjust based on the complexity of the problem and the dataset.
  - Hidden layers allow FNNs to learn complex hierarchical features and patterns in the data.
- **Neurons (Nodes):** Each neuron in a hidden layer or the output layer receives inputs from the previous layer's neurons, applies weights to these inputs, and passes the result through an activation function.
  - Neurons in the hidden layers often use non-linear activation functions (e.g., ReLU, sigmoid, tanh) to introduce non-linearity to the model, enabling it to capture complex relationships in the data.

Main Components of FNNs

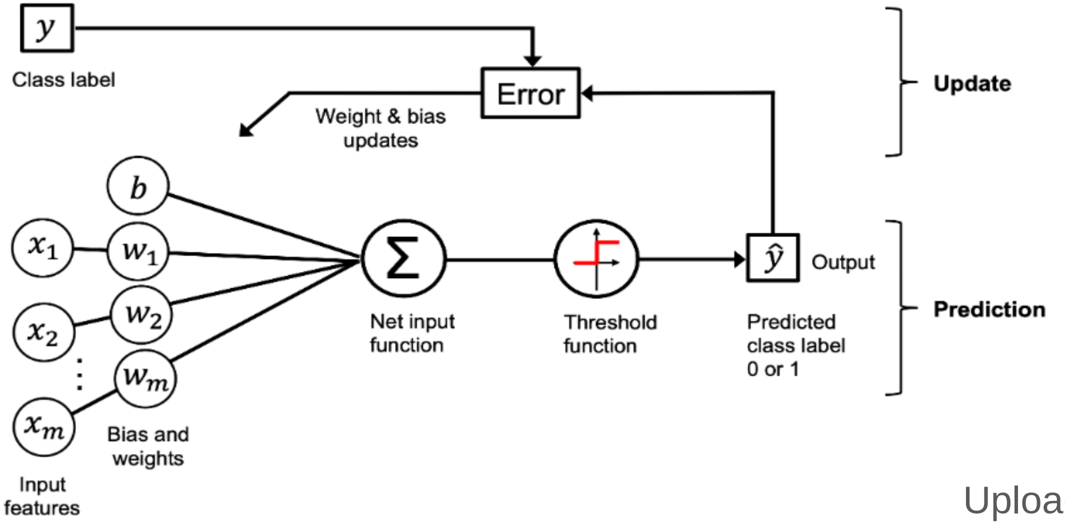
- **Weights and Biases:** Each connection between neurons has an associated weight that determines the strength of the connection.
  - These weights are learned during training.
  - Each neuron also has a bias term that influences its output. Biases are also learned during training.
- **Output Layer:** The output layer produces the final predictions or classifications based on the information processed in the hidden layers.
  - The number of neurons in the output layer depends on the type of task you're solving.
  - For binary classification, you may have a single neuron with output ranging from 0 to 1.
  - For multi-class classification, you'd have a neuron for each class, outputting the probability of that class.
- **Activation Functions:** Activation functions introduce non-linearity to the network, enabling it to model complex relationships in the data.
  - Common activation functions include ReLU (Rectified Linear Unit), sigmoid, tanh, and softmax (for multi-class classification).

FNNs Architecture

- FNNs can be categorized as Perceptrons and Multi-Layer Perceptrons (MLPs).
- A perceptron is a simple FNN that can be used to solve linearly separable problems.
- An MLP is a more complex ANN that can be used to solve both linearly separable and non-linearly separable problems.
- Choosing the right architecture for a FNN is a critical step in the machine learning process.
- The main factors to consider when choosing an FNN architecture are:
    - **The complexity of the task:** The more complex the task, the more layers and hidden neurons will need.
    - **The size of the training data:** The larger the training data, the more layers and neurons the network will need.
    - **The computational resources available:** The number of layers and neurons in an FNN will also depend on the computational resources that are available. A network with a large number of layers and neurons will require more computing power to train.

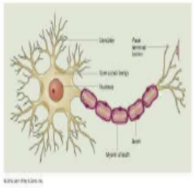
The Perceptron

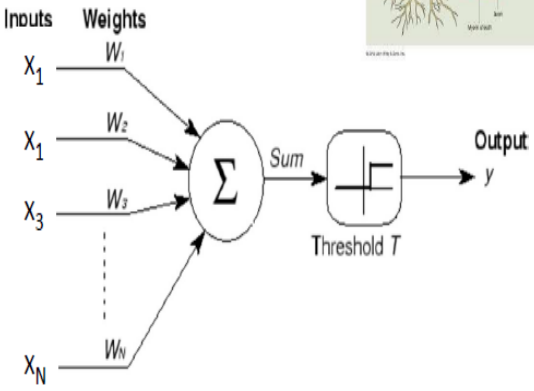
- The **perceptron** is the simplest form of a neural network.
  - The primary purpose of a perceptron is to make binary decisions, such as classifying input data into two categories (e.g., yes/no, 1/0).
  - It is made up of a single layer of neurons, each of which computes a weighted sum of its inputs and applies a linear/non-linear activation function to the result.
  - Single perceptron is limited in its capabilities and can only solve linearly separable problems.



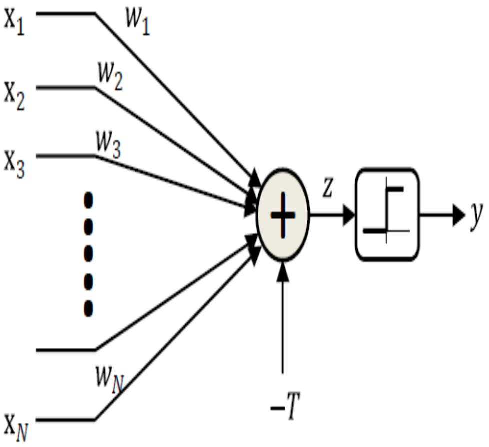
The Perceptron

- A threshold unit "Fires" if the weighted sum of inputs (linear transform) exceeds a threshold T
- A threshold unit "Fires" if the weighted sum of inputs and the bias T (affine transform) is positive





$$y = \begin{cases} 1 \; if \; \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$

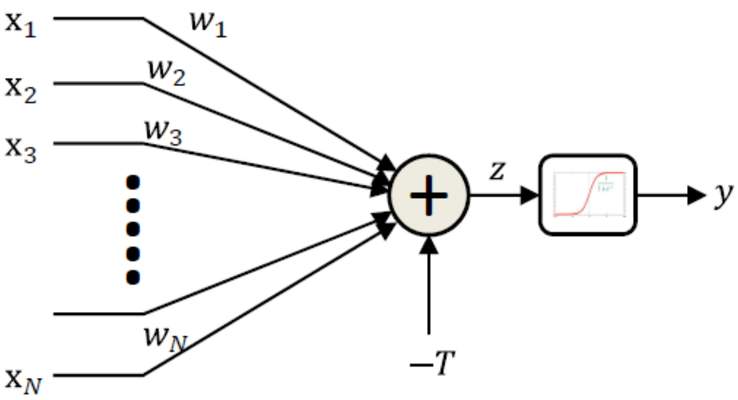


$$z = \sum_i w_i x_i - T$$

$$y = \begin{cases} 1 \; if \; z \geq 0 \\ 0 \; else \end{cases}$$

The “Soft” Perceptron

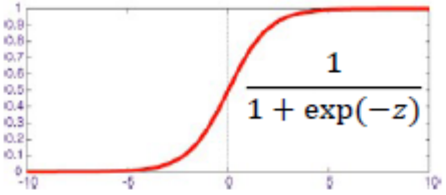
- A non-linear activation function is used at the output which normalizes the output to a range of values and helps the network learn complex data.



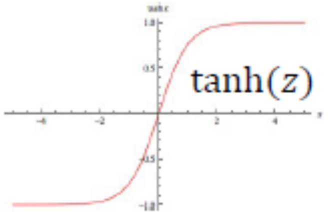
$$z = \sum_i w_i x_i - T$$

$$y = \frac{1}{1 + exp(-z)}$$

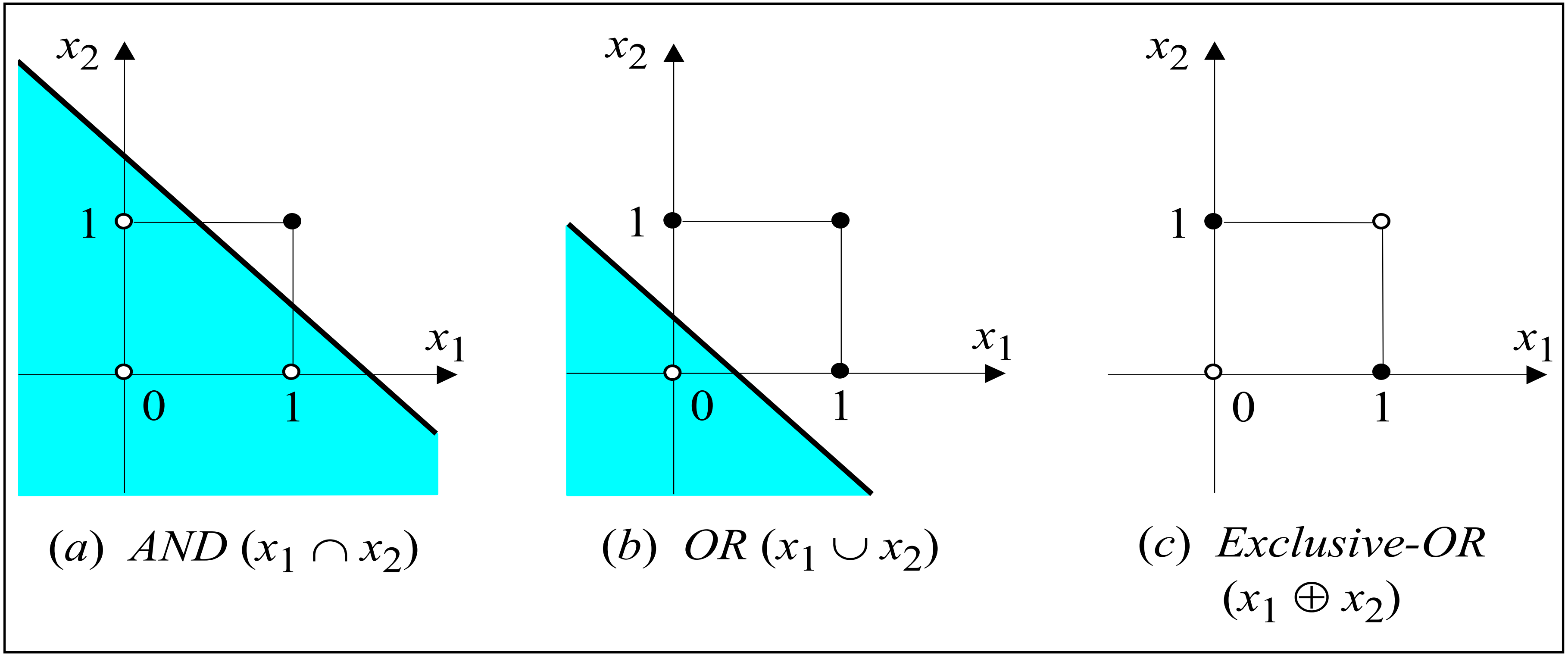
sigmoid



tanh



Performance of Perceptron



(*a*) *AND* $(x_1 \cap x_2)$

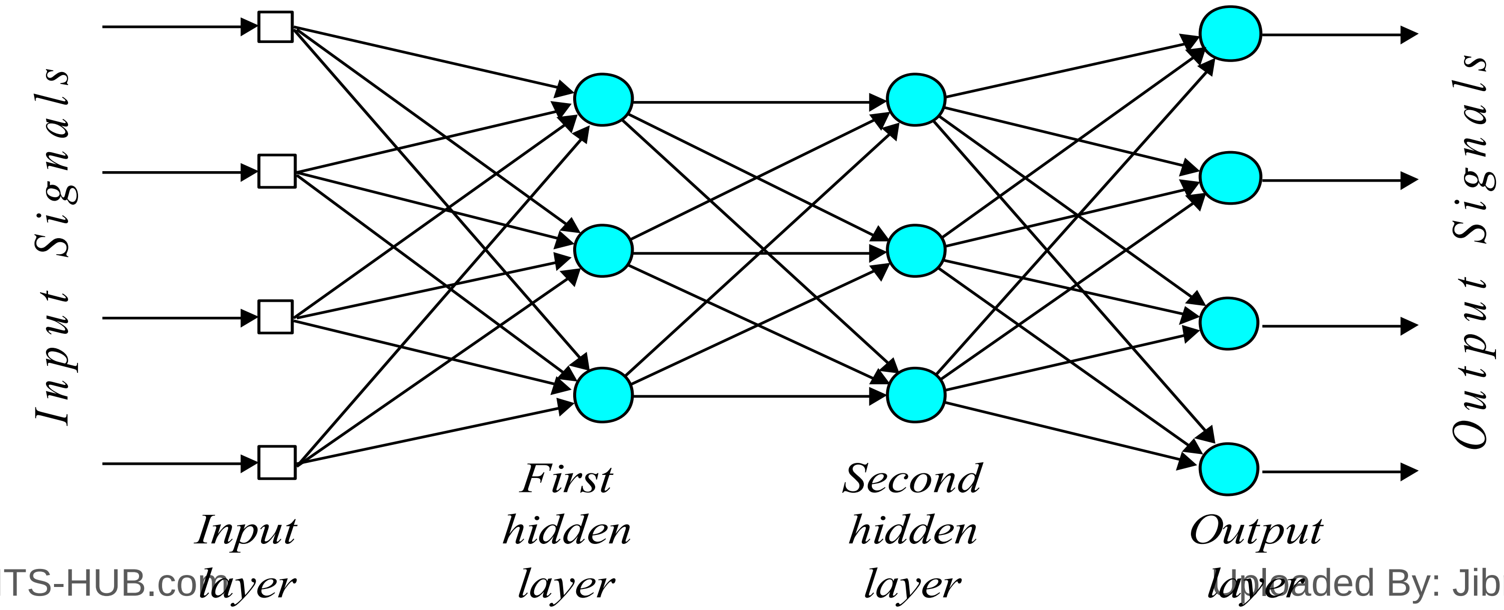
(*b*) *OR* $(x_1 \cup x_2)$

(*c*) *Exclusive-OR* $(x_1 \oplus x_2)$

A perceptron can learn the operations *AND* and *OR*, but not *Exclusive-OR*.

A Multi-Layer Perceptron's (MLPs)

- MLPs is a type of artificial neural network that consists of multiple layers of interconnected nodes (artificial neurons) arranged in a feedforward fashion.
- The neurons in the hidden layers of an MLP can learn non-linear relationships between the inputs and outputs of the network, which allows it to solve problems that a perceptron cannot.
- Their performance often depends on factors such as the architecture (number of layers and neurons), choice of activation functions, and the amount and quality of training data.



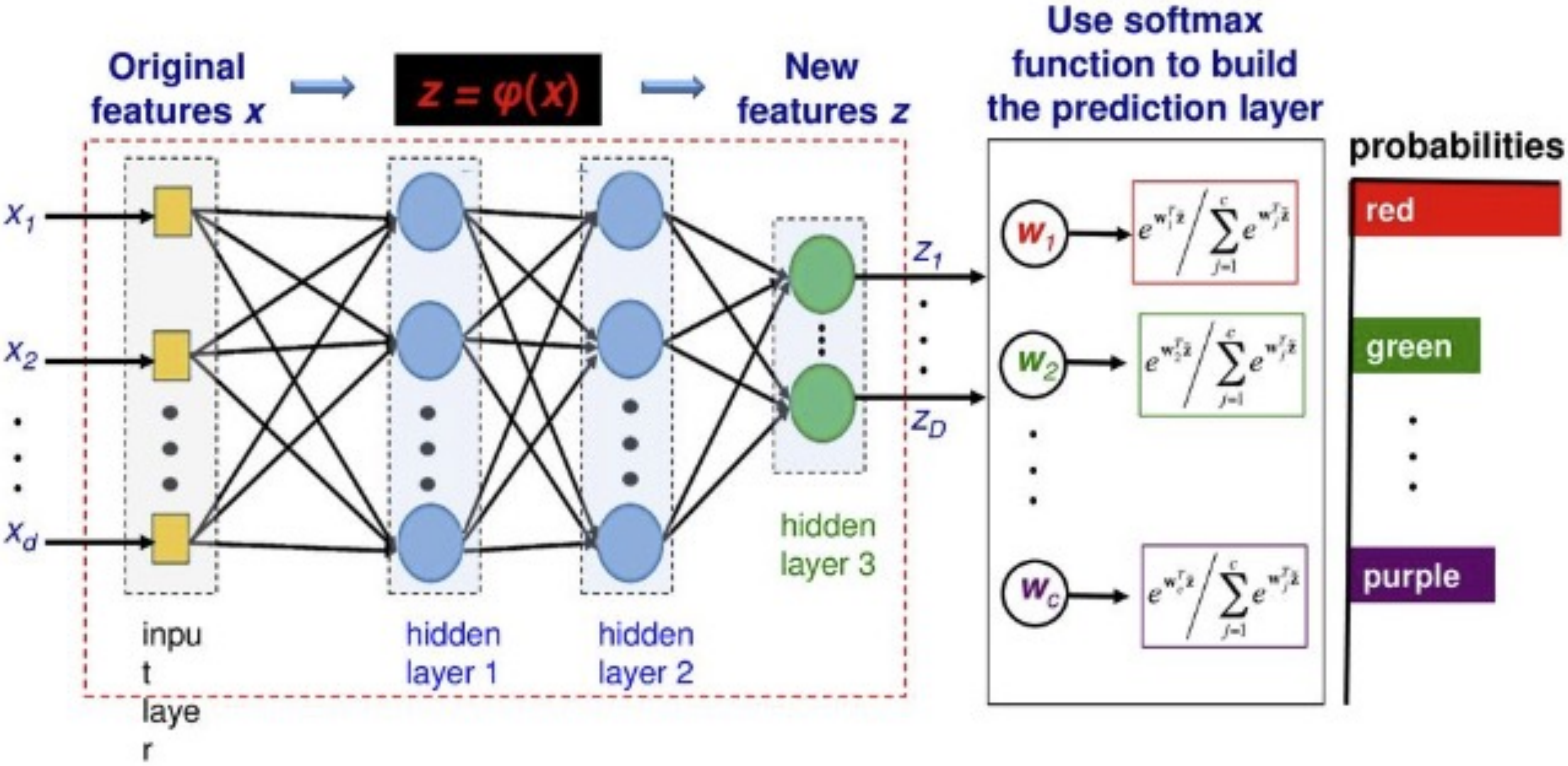
How A Multi-Layer Neural Network Works?

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward** in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function

Multi-Class MLPs

- A multi-class MLPs, is an extension of the MlP model to handle multiple classes.
- A multi-class MLPs, the number of neurons in the output layer corresponds to the number of classes in the classification problem.
- Each neuron in the output layer uses the softmax activation function to transform the weighted sum of inputs into a probability distribution over the classes.
- The softmax function is used in the output layer of a multi-class perceptron because it ensures that the output probabilities sum to 1.
- The cross-entropy loss function is commonly used for training multi-class perceptron's, where the goal is to minimize the difference between the predicted probabilities and the true class labels.
- In multi-class MLPs, input labels are often represented in one-hot encoded format, where each class corresponds to a unique index in the one-hot vector.

Multi-Class MLPs


Original features x
z = φ(x)
New features z
Use softmax function to build the prediction layer
probabilities
x1
x2
xd
input layer
hidden layer 1
hidden layer 2
hidden layer 3
Z1
ZD
w1
w2
wc
$e^{w_1^T z} / \sum_{j=1}^{c} e^{w_j^T z}$
$e^{w_2^T z} / \sum_{j=1}^{c} e^{w_j^T z}$
$e^{w_c^T z} / \sum_{j=1}^{c} e^{w_j^T z}$
red
green
purple

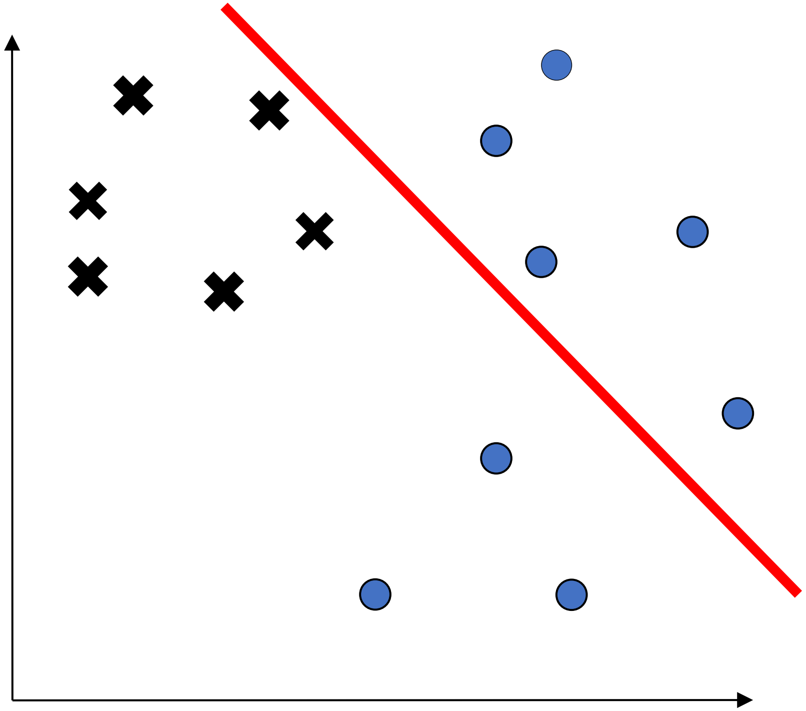
Outline

- Introduction and Motivation
- Neural Network Architecture
  - The Perceptron
  - MLPs
  - Multi-class Perceptron
- Training MLPs
- Choosing Network Structure
  - Depth vs Width
  - Expressive Power of MLPs

Training ANNs – Main Idea

- **Objective:**
  - The primary goal of training an MLP is to minimize the difference between the predicted outputs (obtained from the network) and the actual target values (ground truth) for a given dataset.
  - ANNs learn by adjusting the weights and biases of their connections based on observed data.
- The learning process involves:
  - **Forward propagation:** calculating outputs for a given input
  - and **backward propagation:** adjusting weights using gradient descent and the backpropagation algorithm.
    - **Gradient Descent:** Gradient descent is the optimization algorithm used to update the model's parameters. It involves iteratively adjusting the weights and biases in small steps (controlled by a learning rate) to minimize the loss function.
    - **Backpropagation:** Backpropagation calculates the gradient of the loss function with respect to the network's weights and biases. It essentially measures how a small change in a weight or bias would affect the loss.
- This iterative process aims to minimize a loss function that quantifies the difference between predicted and actual outcomes.

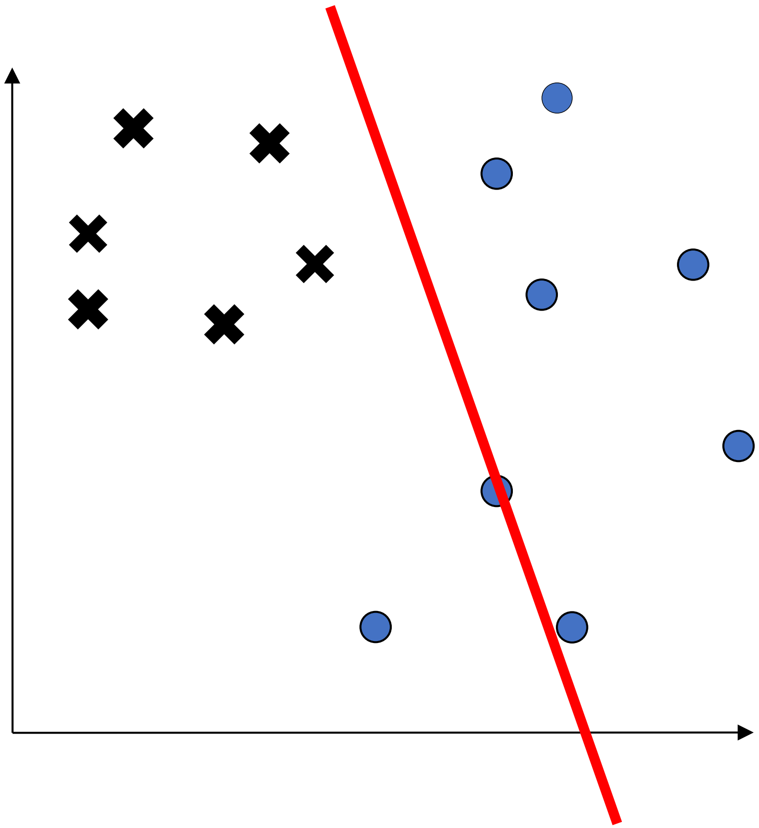
Training ANNs – Main Idea



Is this a good decision boundary?

$$\textit{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) > t \quad \text{then } output = 1, \text{ else } output = 0$$

Training ANNs – Main Idea



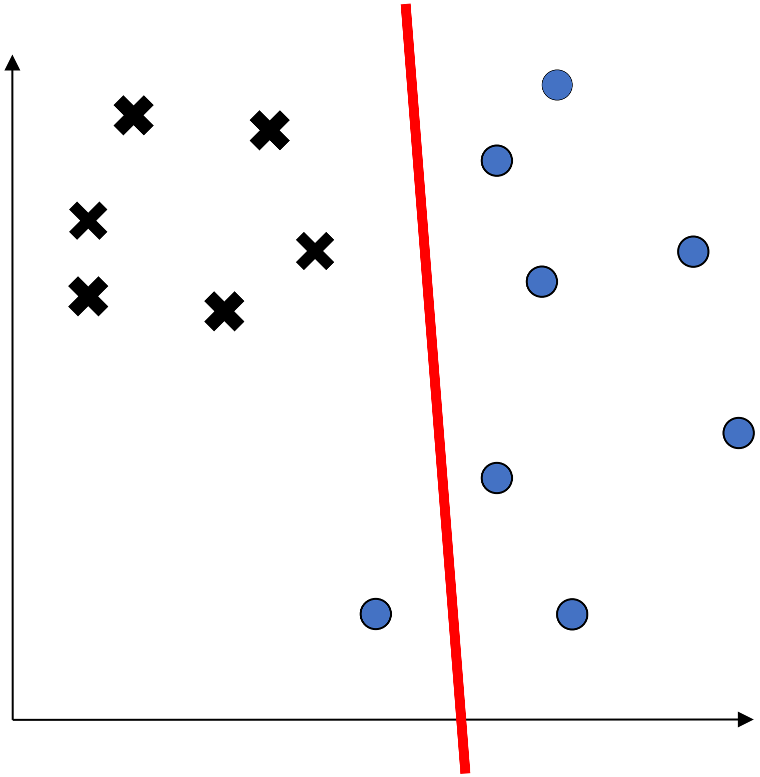
$w_1 = 2.1$

$w_2 = 0.2$

$t = 0.05$

$$\textit{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) > t \quad \text{then } output = 1, \text{ else } output = 0$$

Training ANNs – Main Idea



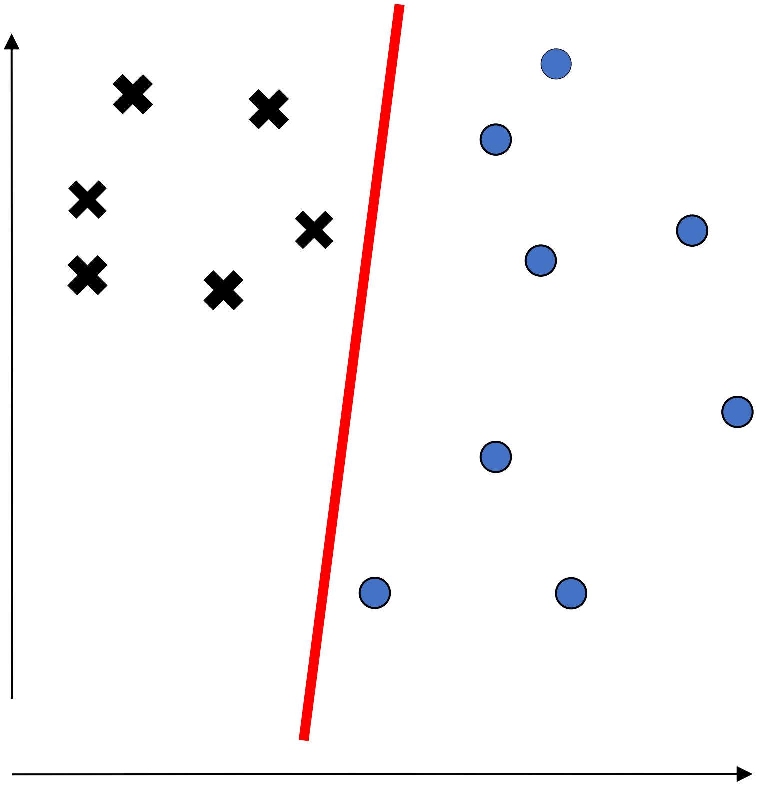
w_1 = 1.9

w_2 = 0.02

t = 0.05

$$\textit{if}\quad \left(\sum_{i=1}^{M} x_i w_i\right) \quad > \quad t \qquad \text{then } output = 1,\ \text{else } output = 0$$

Training ANNs – Main Idea



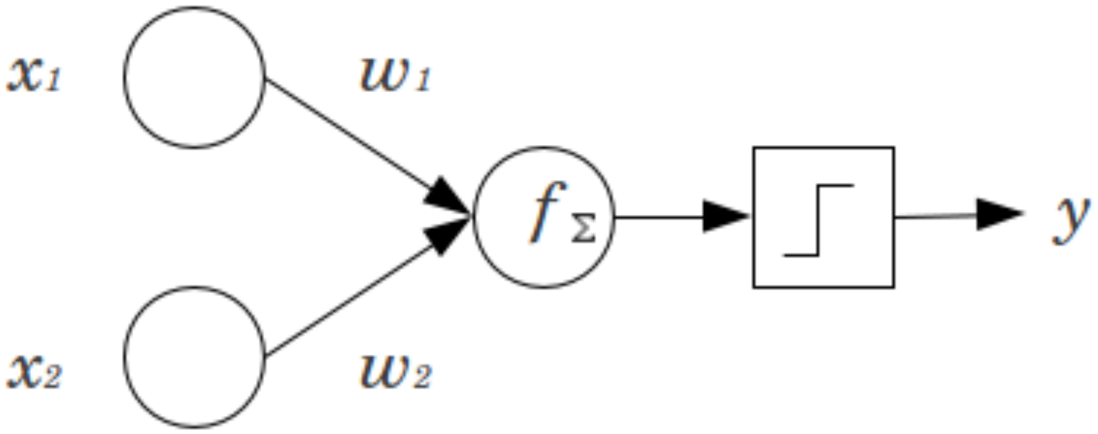
w_1 = -0.8

w_2 = 0.03

t = 0.05

- ➢ Changing the weights/threshold makes the decision boundary move.
- ➢ Pointless / impossible to do it by hand – only ok for simple 2-D case.
- ➢ We need an algorithm….

Trained Perceptron – An Example



- W1 = 0.1, W2 = 0.1, Threshold = 0.2, Step Activation Function → AND Gate
- W1 = 0.1, W2 = 0.1, Threshold = 0.1, Step Activation Function → OR Gate

MLPs Training Algorithm

1. **Preparing Network Architecture:**
   - ❑ The architecture includes the number of layers, the number of neurons in each layer, the activation functions, and the loss function.
2. **Initializations:** the initialization includes the following:
   - ❑ Weights and biases: common techniques include random initialization and using smart techniques like Xavier/Glorot initialization.
   - ❑ Hyperparameters: like learning rate, batch size, and number of epochs.
3. **Choose Optimization Algorithm:** select optimization algorithm to update the network's parameters. Gradient Descent as an example.

MLPs Training Algorithm

4. **Gradient Descent with Backpropagation Training Loop:**

   A. Present a training example: A training example is presented to the perceptron.

   - ➢ The training example consists of a set of inputs and a desired output.

   B. Forward Pass: During the forward pass, the input data is fed through the network layer by layer, and the activations are calculated at each layer.

   1. Input Layer: Initialize the input activations with the training data.
   2. Hidden Layers: For each hidden layer, calculate the weighted sum of the input activations and the layer's weights
   3. Apply the activation function to the weighted sum to compute the output activations.
   4. Output Layer: for the output layer, calculate the weighted sum and apply an appropriate activation function (e.g., sigmoid, softmax).

MLPs Training Algorithm

C. **Backward Pass (Backpropagation):** During the backward pass, gradients of the loss with respect to each parameter are calculated and propagated backward through the layers.

   1. Calculate the error at the output layer. This is the difference between the desired output and the predicted output.
   2. Compute Output Layer Gradient: the gradient of the loss function with respect to the output layer weights (dloss/dW$_{output}$). This is done using the chain rule.
   3. Use the gradient to update the output layer weights.
   4. Propagate the error to the hidden layers. This is done by multiplying the error at the output layer by the weights connecting the output layer to the hidden layer.
   5. Calculate the gradient of the loss function with respect to the hidden layer weights. This is done using the chain rule.
   6. Use the gradient to update the hidden layer weights.
   7. Repeat steps 1 to 6 until updating parameters in the input layer.

5. Repeat steps a to c for a predefined number of iterations (epochs) or error is minimized.

How Does Gradient Descent Work?

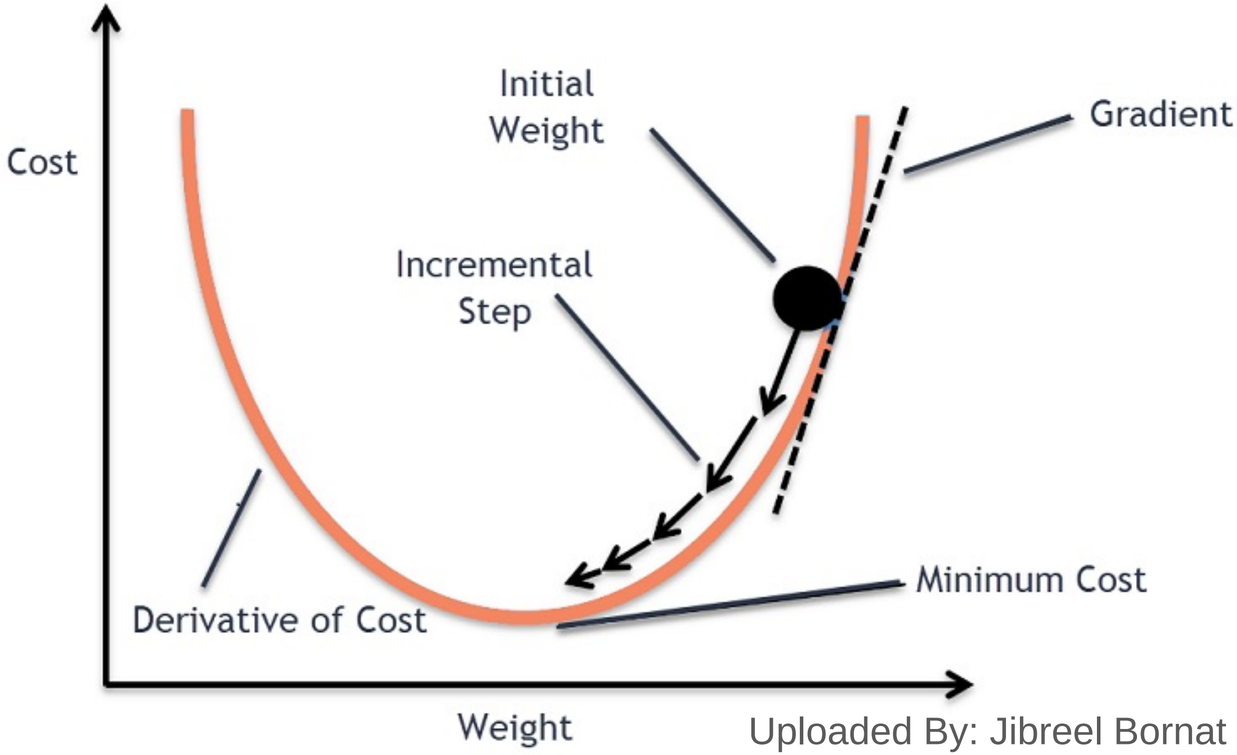
- Gradient Descent is an Iterative Solver.
  - The Iterative solver does not give the **exact** solution.
  - The iterative solvers are used to get the **approximate** solution as the purpose is to minimize the objective function.
- The algorithm starts with an initial set of parameters and updates them in small steps to minimize the cost function.
- In each iteration of the algorithm, the gradient of the cost function with respect to each parameter is computed.
- The gradient tells us the direction of the steepest ascent, and by moving in the opposite direction, we can find the direction of the steepest descent.
- The size of the step is controlled by the learning rate, which determines how quickly the algorithm moves towards the minimum.
- The process is repeated until the cost function converges to a minimum, indicating that the model has reached the optimal set of parameters.

How Does Gradient Descent Work?

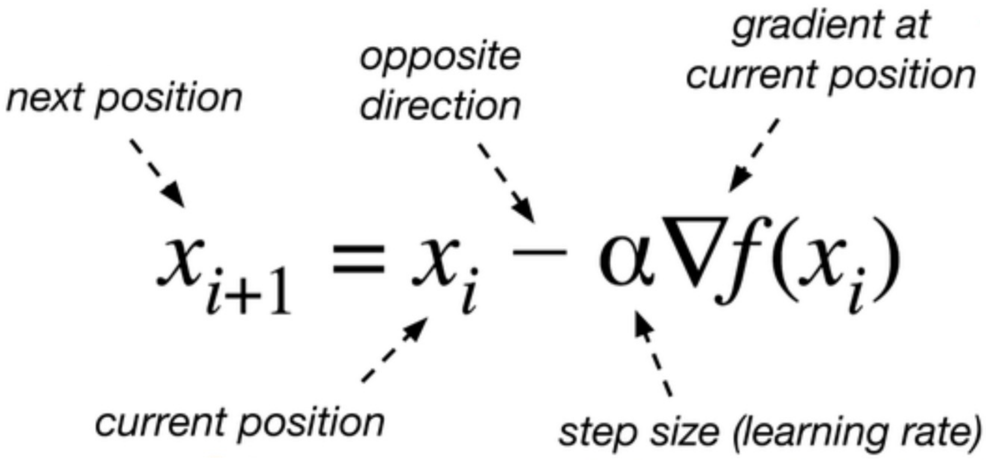
- The goal of the gradient descent algorithm is to minimize the cost function. To achieve this goal, it performs two steps iteratively:
  1. **Compute the gradient** (slope), the first order derivative of the cost function at that point
  2. **Make a step (move) in the direction opposite to the gradient**, opposite direction of slope increase from the current point by alpha times the gradient at that point

The **Step Size** will be **Very Close to 0** when the **Slope** is very close to **0**.

Step Size = $\boxed{\textbf{Slope}}$ × **Learning Rate**

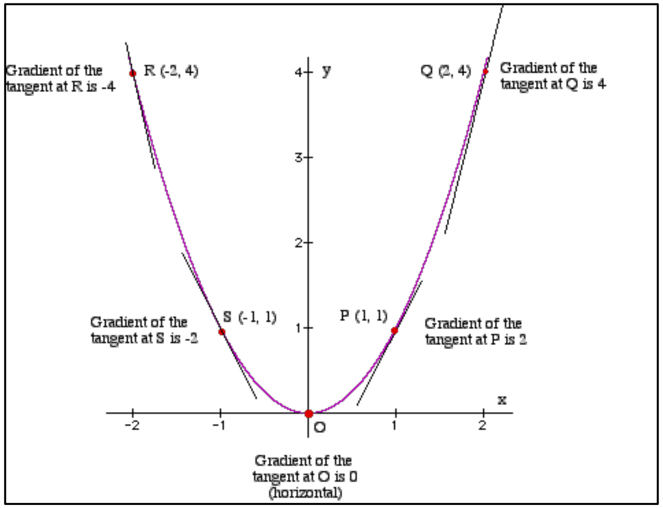


Gradient Descent Update Rules



$$x_{i+1} = x_i - \alpha \nabla f(x_i)$$

MSE cost function is equivalent to Y = f(x) = X^2



Derivative Sign

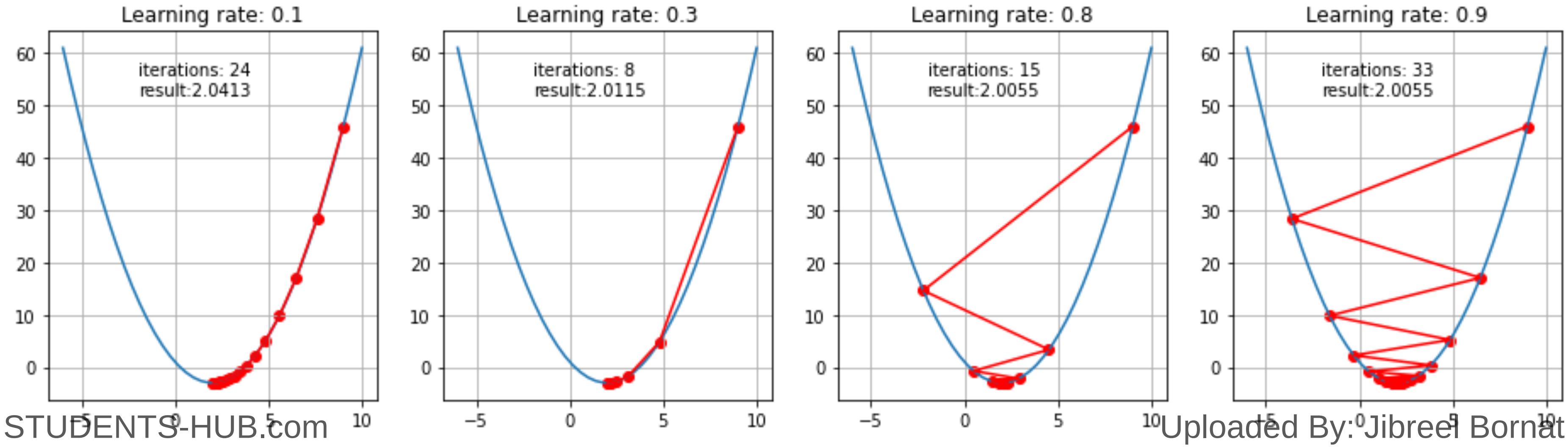
Positive	Increasing/decreasing weight increases/decreases error.
Negative	Increasing/decreasing weight decreases/increases error.

Derivative Magnitude

Positive Sign	Increasing/decreasing weight by P increases/decreases error by MAG*P.
Negative Sign	Increasing/decreasing weight by P decreases/increases error by MAG*P.

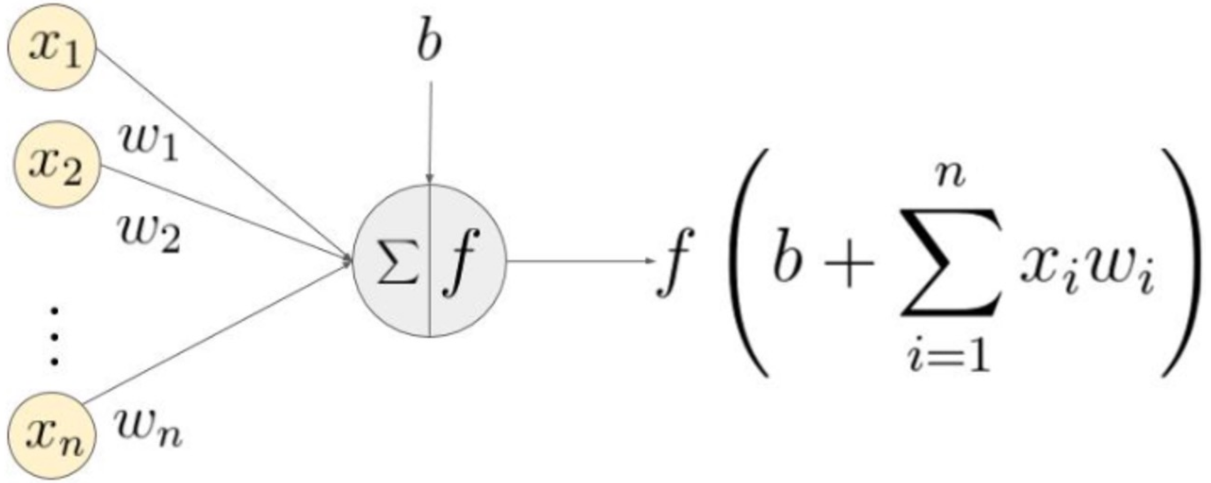
The Learning Rate

- We have the direction we want to move in, now we must decide the size of the step we must take.
- The learning rate defined as the step size taken to reach the minima or lowest point.
    - **Smaller learning rate:** the model will take too much time before it reaches minima might even exhaust the max iterations specified.
    - **Large (Big learning rate):** the steps taken will be large and we can even miss the minima the algorithm may not converge to the optimal point.



Activation Functions

- The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it.
- The purpose of the activation function is to introduce non-linearity into the output of a neuron.
- The activation function enables the MLP to capture more complex patterns and makes it capable of learning and approximating a wide variety of functions.
- Activation functions have distinct impacts on the training convergence speed, dealing with noise and outliers, handling vanishing and exploding gradient problems, and computational efficiency.



Well-Known Activation Functions

- **Sigmoid, tanh, and ReLU** are the most popular activation functions.
- **Sigmoid and tanh**
  - Suffer from the vanishing gradient problem, which can slow down training in deep networks.
  - Sensitive to outliers.
  - They are less commonly used in hidden layers of deep networks today.
- **ReLU**
  - Faster convergence, mitigation of the vanishing gradient problem for positive inputs.
  - Computational efficient
  - More robust to noise, outliers
  - Become the default choice for most hidden layers in deep networks
  - However, ReLU requires careful initialization and regularization techniques.

Loss Function

- A loss function, also known as a cost function or objective function, is a crucial component in training ANNs and other machine learning models.
- It is used during training to evaluate the performance of the neural network and to update its weights in order to minimize the loss.
- Its primary purpose is to measure how well the model's predictions match the actual target values (ground truth) during the training process.
  - The goal is to find a set of weights and biases that minimizes the cost.
- Key properties of Loss Function:
  - **Accuracy:** The loss function should be able to accurately measure the difference between the predicted output of the model and the desired output.
  - **Differentiability:** The loss function should be differentiable, so that the gradient of the loss function with respect to the model parameters can be calculated.
  - **Convexity:** The loss function should be convex, so that there is a single global minimum. This ensures that the optimization algorithm will converge to the best possible solution.
  - **Computational efficiency:** The loss function should be computationally efficient to evaluate, so that it can be used to train large and complex models in a reasonable amount of time.

Loss Function

- Here are some examples of loss functions that satisfy these key properties:
  - **Mean squared error (MSE):** MSE is a simple and efficient loss function that is often used for regression tasks. It is differentiable and convex, and it is robust to ou

$$\text{MSE} = \overset{\text{Mean}}{\boxed{\frac{1}{n}\sum_{i=1}^{n}}}\Big(\overset{\text{Error}}{\boxed{Y_i - \hat{Y}_i}}\Big)^{\overset{\text{Squared}}{\boxed{2}}}$$

  - **Cross-entropy loss:** Cross-entropy loss is a common loss function for classification tasks. It is differentiable and convex, but it is not as robust to outliers as MSE.

$$H(p,q) = -\sum_{x\in\text{classes}} \overset{\text{True probability distribution (one-hot)}}{p(x)} \log \underset{\text{Your model's predicted probability distribution}}{q(x)}$$

Gradient Descent - Mathematics

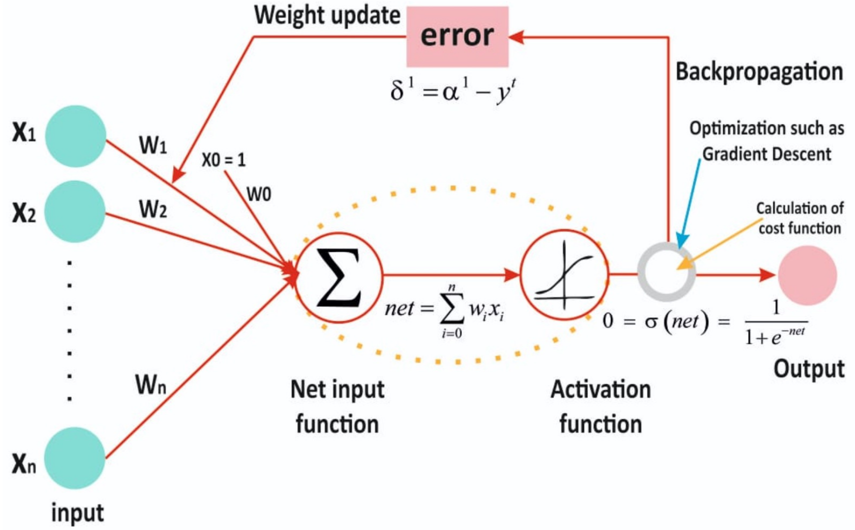
- **Forward Pass – Assume sigmoid activation function**

$\text{Output}_{\text{predicted}} = \text{Sigmoid(S)}$

Where:

$S = \sum w_i * x_i + \text{bias}$

$\text{Sigmoid} = 1/(1 + e^{-x})$


Weight update
error
$\delta^1 = \alpha^1 - y^t$
Backpropagation
Optimization such as Gradient Descent
Calculation of cost function
X1
X2
Xn
W1
W2
Wn
X0 = 1
W0
$net = \sum_{i=0}^{n} w_i x_i$
$0 = \sigma(net) = \frac{1}{1+e^{-net}}$
Output
Net input function
Activation function
input

- **Backward Pass**
  - **Calculate Error – Assume MSE loss**

$\text{MSE Loss} = ½\ (\text{Output}_{\text{desired}} - \text{Output}_{\text{predicted}})^2$

  - **Calculate Gradients**

By chain rule,

$\text{dLoss/dW}_i = [(\text{dE/dpredicted})*(\text{dpredicted/ds})*(\text{ds/dWi})]$

$\text{dE/dpredicted} = \text{Output}_{\text{predicted}} - \text{Output}_{\text{desired}}$

$\text{dpredicted/ds} = [(1/(1 + e^{-s}))(1-(1/\ (1 + e^{-s}))]$

$\text{ds/dW}_i = X_i$

$\text{dLoss/dW}_i = (\text{Output}_{\text{predicted}} - \text{Output}_{\text{desired}})*[(1/(1 + e^{-s}))(1-(1/\ (1 + e^{-s}))]*X_i$

  - **Update**

$\text{Wi}_{\text{new}} = \text{Wi}_{\text{old}} - (\text{larning\_rate} * \text{dLoss/dWi})$

Case study - Perceptron Training: Step-by-step

- Training of a two-input perceptron using stochastic gradient descent with backpropagation, sigmoid activation function, and Mean Squared Error (MSE) loss function.
- **Step 1: Initialize Weights and Bias**
  - Initialize the weights (w1 and w2) and bias (b) with small random values.

    w1 = random_initialization

    w2 = random_initialization

    b = random_initialization
- **Step 2: Forward Pass**
  - Calculate the weighted sum of the inputs and add the bias to get the linear combination (z):

    z = w1*x1 + w2*x2 + b

Perceptron training – Case study: step-by-step

- **Step 3: Apply Sigmoid Activation Function**
  - Pass the linear combination (z) through the sigmoid activation function to obtain the predicted probability of class 1 (y_pred):

    y_pred = 1 / (1 + exp(-z))
- **Step 4: Calculate Error (MSE)**
  - Compute the Mean Squared Error (MSE) between the predicted output (y_pred) and the actual target (y_true):

    MSE = (1/2) * (y_true - y_pred)^2
- **Step 5: Calculate Gradients**
  - Calculate the gradients of the MSE with respect to the weights (w1 and w2) and bias (b) using backpropagation.
    - Gradients with respect to weights:

    ∂MSE/∂w1 = -(y_true - y_pred) * y_pred * (1 - y_pred) * x1

    ∂MSE/∂w2 = -(y_true - y_pred) * y_pred * (1 - y_pred) * x2
    - Gradient with respect to bias:

    ∂MSE/∂b = -(y_true - y_pred) * y_pred * (1 - y_pred)

Perceptron training – Case study: step-by-step

- **Step 6: Update Weights and Bias**
  - Update the weights and bias using the calculated gradients and a learning rate (α):

    w1 = w1 - α * ∂MSE/∂w1

    w2 = w2 - α * ∂MSE/∂w2

    b = b - α * ∂MSE/∂b
- **Step 7: Repeat**
  - Repeat Steps 2 to 6 for each data point in the training dataset or for a mini-batch of data points.
  - Repeat this process for a fixed number of epochs or until convergence.

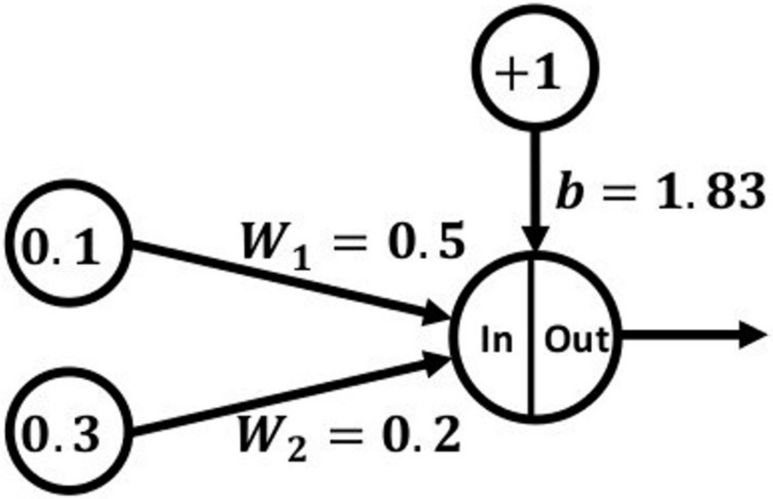
Training Perceptron – Numerical Example

Training Data

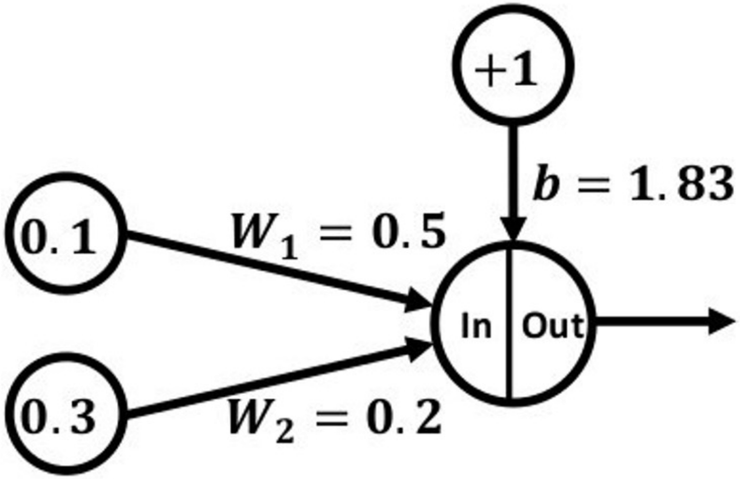
X_1	X_2	Output
0.1	0.3	0.03

Initial Weights

W_1	W_2	b
0.5	0.2	1.83



Forward Pass



- In this example, the sigmoid activation function is used.

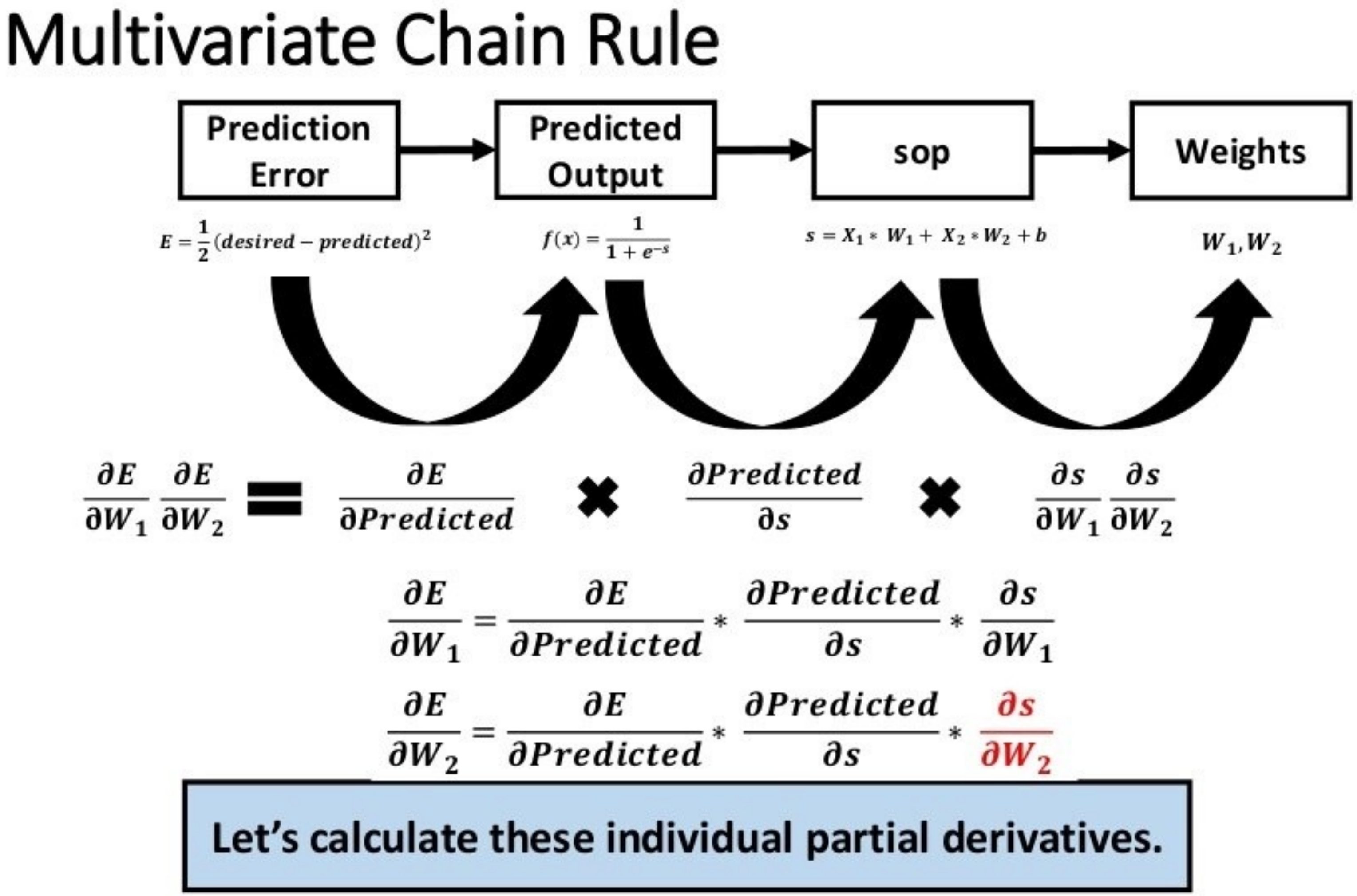
$$f(s) = \frac{1}{1 + e^{-s}}$$

- Based on the sop calculated previously, the output is as follows:

$$f(s) = \frac{1}{1 + e^{-1.94}} = \frac{1}{1 + 0.144} = \frac{1}{1.144}$$

$$f(s) = 0.874$$

Backward Pass



Multivariate Chain Rule

Prediction Error → Predicted Output → sop → Weights

$E = \frac{1}{2}(desired - predicted)^2$

$f(x) = \frac{1}{1 + e^{-s}}$

$s = X_1 * W_1 + X_2 * W_2 + b$

W_1, W_2

$$\frac{\partial E}{\partial W_1} \frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial Predicted} \times \frac{\partial Predicted}{\partial s} \times \frac{\partial s}{\partial W_1} \frac{\partial s}{\partial W_2}$$

$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial Predicted} * \frac{\partial Predicted}{\partial s} * \frac{\partial s}{\partial W_1}$$

$$\frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial Predicted} * \frac{\partial Predicted}{\partial s} * \frac{\partial s}{\partial W_2}$$

Let’s calculate these individual partial derivatives.

Backward Pass

Error-Predicted ($\frac{\partial E}{\partial Predicted}$) Partial Derivative

$$E = \frac{1}{2}(desired - predicted)^2$$

$$\frac{\partial E}{\partial Predicted} = \frac{\partial}{\partial Predicted}(\frac{1}{2}(desired - predicted)^2)$$

$$= 2 * \frac{1}{2}(desired - predicted)^{2-1} * (0 - 1)$$

$$= (desired - predicted) * (-1)$$

$$= predicted - desired$$

Substitution

$$\frac{\partial E}{\partial Predicted} = predicted - desired = 0.874 - 0.03$$

$$\frac{\partial E}{\partial Predicted} = 0.844$$

Predicted-sop ($\frac{\partial Predicted}{\partial s}$) Partial Derivative

$$Predicted = \frac{1}{1+e^{-s}}$$

$$\frac{\partial Predicted}{\partial s} = \frac{\partial}{\partial s}(\frac{1}{1+e^{-s}})$$

$$\frac{\partial Predicted}{\partial s} = \frac{1}{1+e^{-s}}(1-\frac{1}{1+e^{-s}})$$

Substitution

$$\frac{\partial Predicted}{\partial s} = \frac{1}{1+e^{-s}}(1-\frac{1}{1+e^{-s}}) = \frac{1}{1+e^{-1.94}}(1-\frac{1}{1+e^{-1.94}})$$

$$= \frac{1}{1+0.144}(1-\frac{1}{1+0.144})$$

$$= \frac{1}{1.144}(1-\frac{1}{1.144})$$

$$= 0.874(1-0.874)$$

$$= 0.874(0.126)$$

$$\frac{\partial Predicted}{\partial s} = 0.11$$

Backward Pass

Sop-W_1 ($\frac{\partial s}{\partial W_1}$) Partial Derivative

$$\mathbf{s = X_1 * W_1 + X_2 * W_2 + b}$$

$$\frac{\partial s}{\partial W_1} = \frac{\partial}{\partial W_1}(X_1 * W_1 + X_2 * W_2 + b)$$

$$= 1 * X_1 * (W_1)^{(1-1)} + 0 + 0$$

$$= X_1 * (W_1)^{(0)}$$

$$= X_1(1)$$

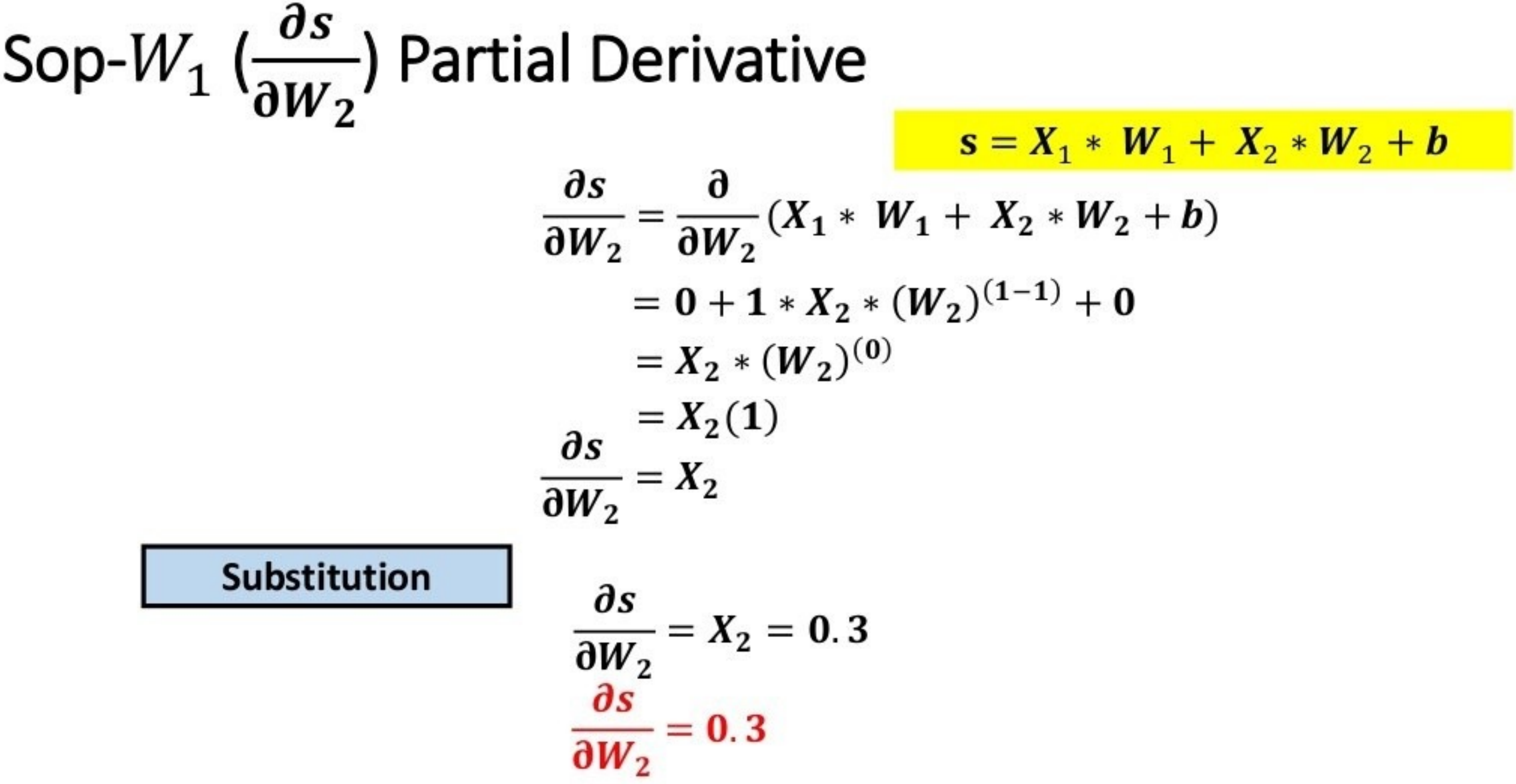
$$\frac{\partial s}{\partial W_1} = X_1$$

Substitution

$$\frac{\partial s}{\partial W_1} = X_1$$

$$\frac{\partial s}{\partial W_1} = 0.1$$

Backward Pass



Sop-W_1 ($\frac{\partial s}{\partial W_2}$) Partial Derivative

$$\mathbf{s = X_1 * \ W_1 + \ X_2 * W_2 + b}$$

$$\frac{\partial s}{\partial W_2} = \frac{\partial}{\partial W_2}(X_1 * \ W_1 + \ X_2 * W_2 + b)$$

$$= 0 + 1 * X_2 * (W_2)^{(1-1)} + 0$$

$$= X_2 * (W_2)^{(0)}$$

$$= X_2(1)$$

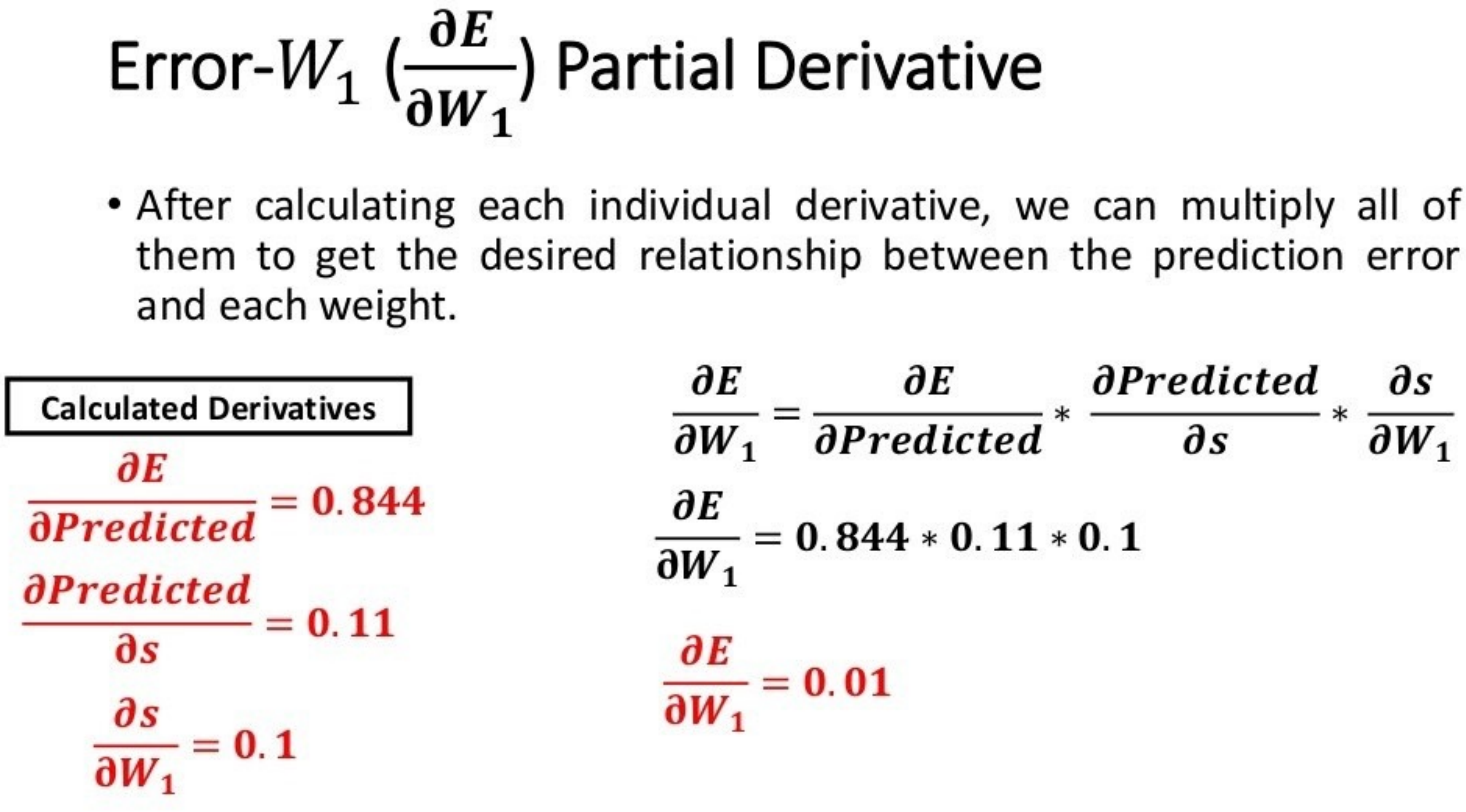
$$\frac{\partial s}{\partial W_2} = X_2$$

Substitution

$$\frac{\partial s}{\partial W_2} = X_2 = 0.3$$

$$\frac{\partial s}{\partial W_2} = 0.3$$

Backward Pass



Error-W_1 ($\frac{\partial E}{\partial W_1}$) Partial Derivative

- After calculating each individual derivative, we can multiply all of them to get the desired relationship between the prediction error and each weight.

Calculated Derivatives

$$\frac{\partial E}{\partial Predicted} = 0.844$$

$$\frac{\partial Predicted}{\partial s} = 0.11$$

$$\frac{\partial s}{\partial W_1} = 0.1$$

$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial Predicted} * \frac{\partial Predicted}{\partial s} * \frac{\partial s}{\partial W_1}$$

$$\frac{\partial E}{\partial W_1} = 0.844 * 0.11 * 0.1$$

$$\frac{\partial E}{\partial W_1} = 0.01$$

Backward Pass

Error-W_2 $(\frac{\partial E}{\partial W_2})$ Partial Derivative

Calculated Derivatives

$$\frac{\partial E}{\partial Predicted} = 0.844$$

$$\frac{\partial Predicted}{\partial s} = 0.11$$

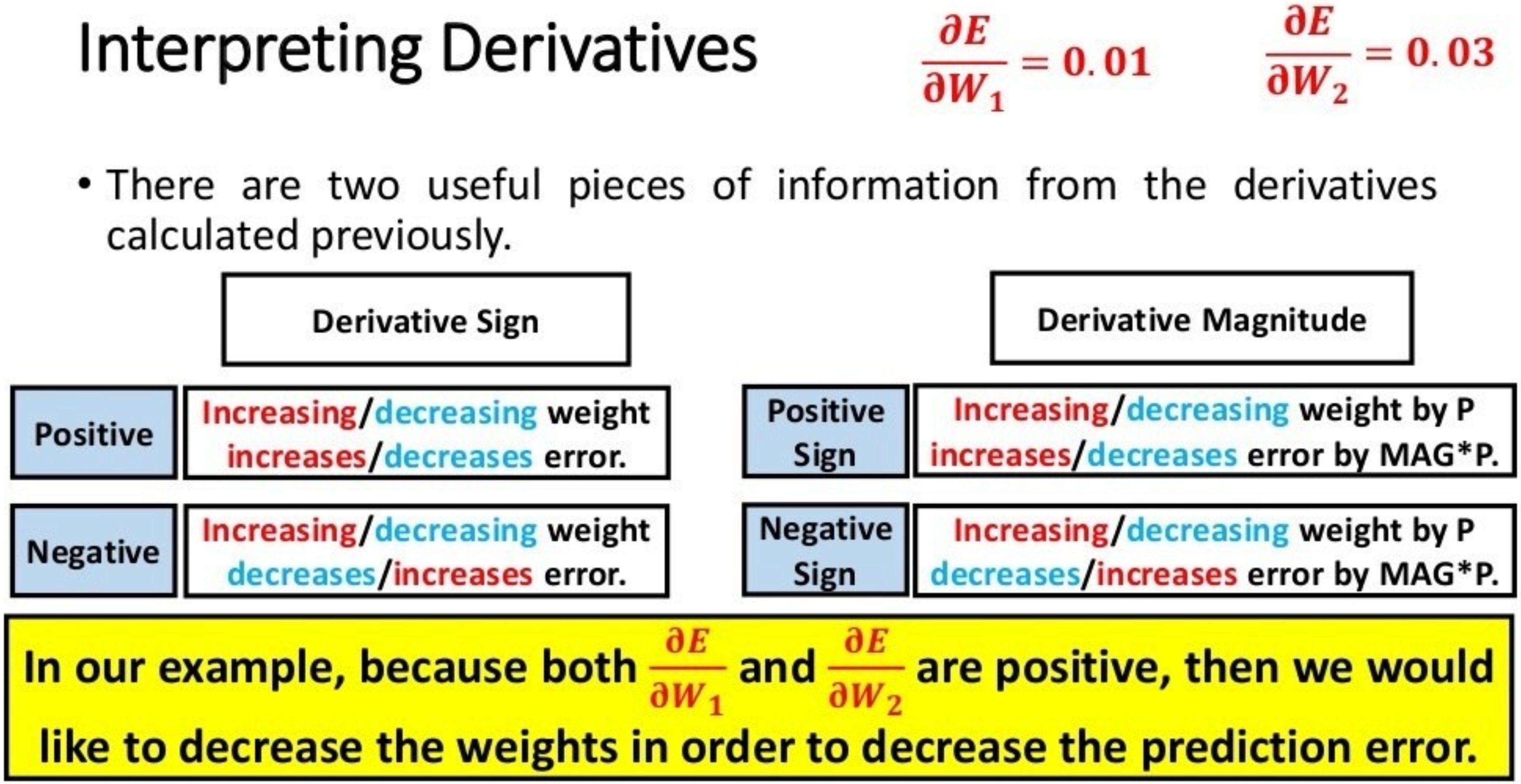
$$\frac{\partial s}{\partial W_2} = 0.3$$

$$\frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial Predicted} * \frac{\partial Predicted}{\partial s} * \frac{\partial s}{\partial W_2}$$

$$\frac{\partial E}{\partial W_2} = 0.844 * 0.11 * 0.3$$

$$\frac{\partial E}{\partial W_2} = 0.03$$

Weights Update



Interpreting Derivatives

$\frac{\partial E}{\partial W_1} = 0.01$ $\frac{\partial E}{\partial W_2} = 0.03$

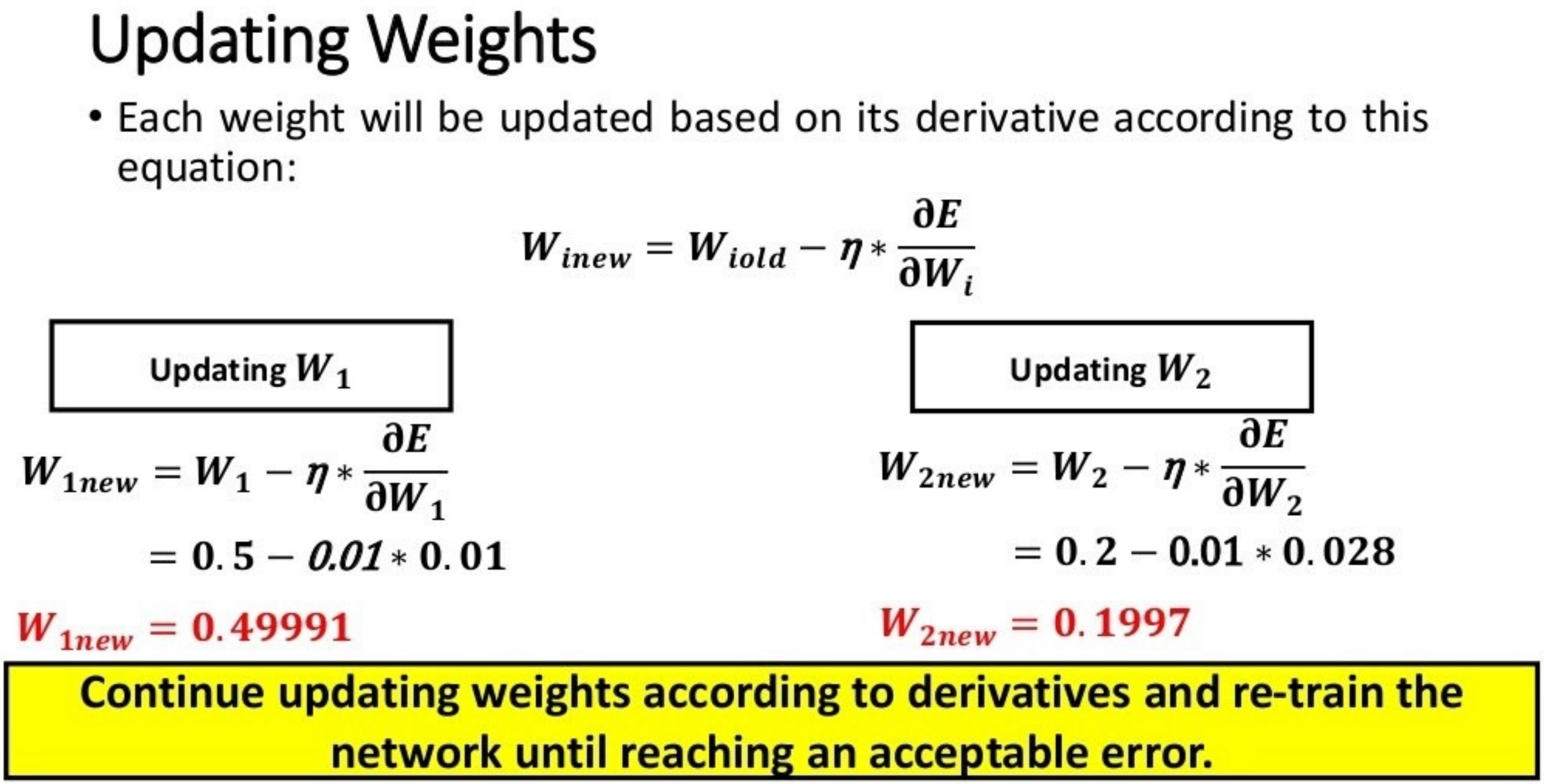
- There are two useful pieces of information from the derivatives calculated previously.

	Derivative Sign
Positive	Increasing/decreasing weight increases/decreases error.
Negative	Increasing/decreasing weight decreases/increases error.

	Derivative Magnitude
Positive Sign	Increasing/decreasing weight by P increases/decreases error by MAG*P.
Negative Sign	Increasing/decreasing weight by P decreases/increases error by MAG*P.

In our example, because both $\frac{\partial E}{\partial W_1}$ and $\frac{\partial E}{\partial W_2}$ are positive, then we would like to decrease the weights in order to decrease the prediction error.

Weights Update



Updating Weights

- Each weight will be updated based on its derivative according to this equation:

$$W_{inew} = W_{iold} - \eta * \frac{\partial E}{\partial W_i}$$

Updating W_1

$$W_{1new} = W_1 - \eta * \frac{\partial E}{\partial W_1}$$

$$= 0.5 - \mathit{0.01} * 0.01$$

$$W_{1new} = 0.49991$$

Updating W_2

$$W_{2new} = W_2 - \eta * \frac{\partial E}{\partial W_2}$$

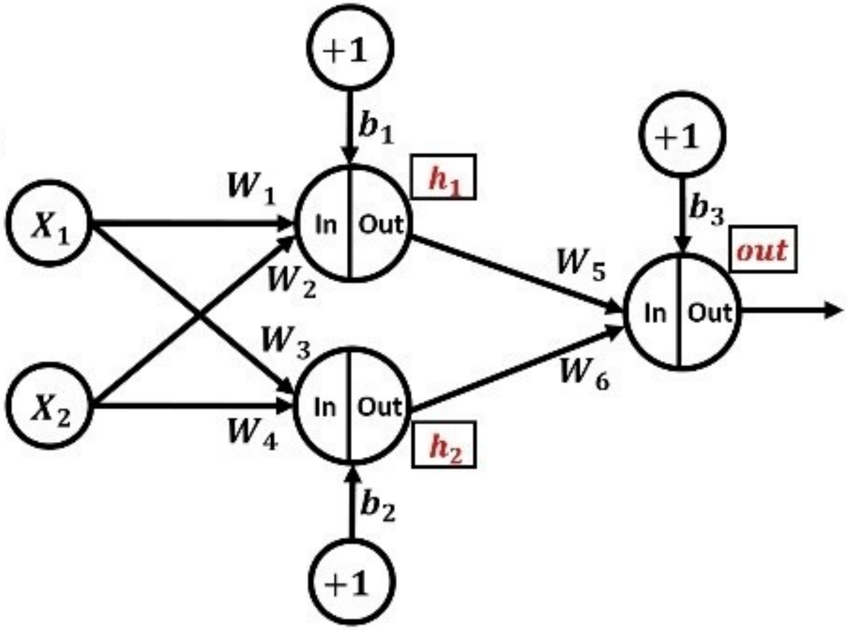
$$= 0.2 - 0.01 * 0.028$$

$$W_{2new} = 0.1997$$

Continue updating weights according to derivatives and re-train the network until reaching an acceptable error.

Training MLPs with One Hidden Layer

ANN with Hidden Layer



Training Data

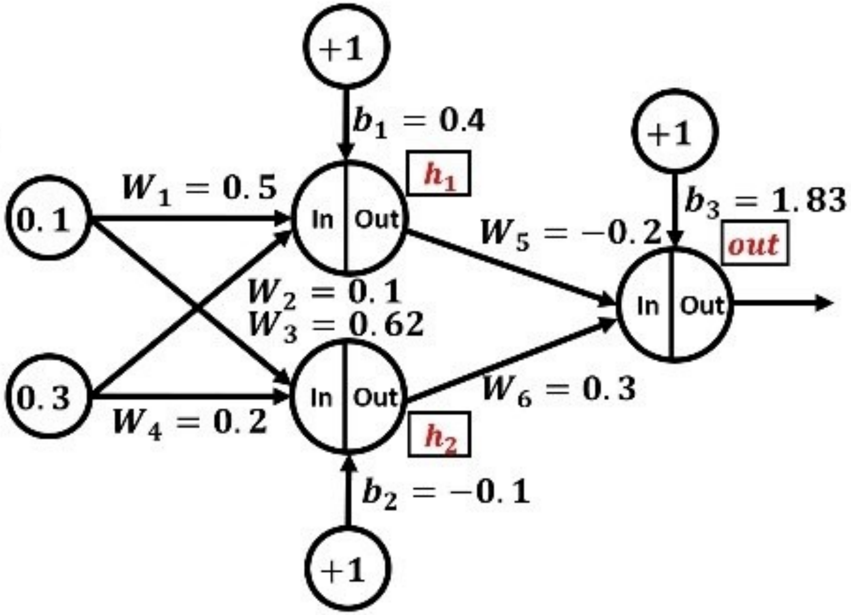
X_1	X_2	Output
0.1	0.3	0.03

Initial Weights

W_1	W_2	W_3	W_4	W_5	W_6	b_1	b_2	b_3
0.5	0.1	0.62	0.2	-0.2	0.3	0.4	-0.1	1.83

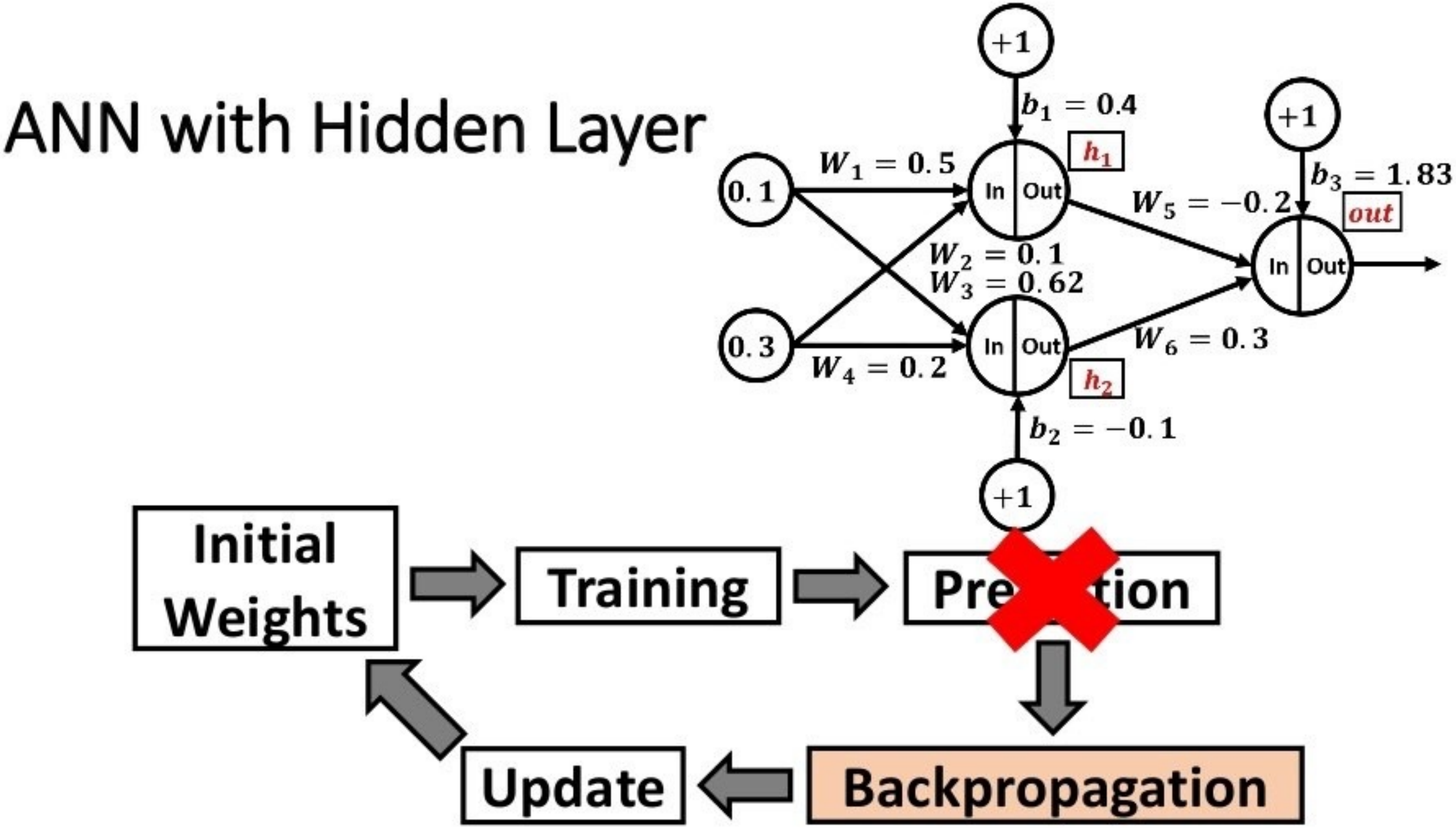
Training MLPs with One Hidden Layer

ANN with Hidden Layer



Initial Weights → Training → Prediction

Training MLPs with One Hidden Layer



ANN with Hidden Layer

Initial Weights → Training → ~~Prediction~~ → Backpropagation → Update → Initial Weights

Forward Pass

Forward Pass – Hidden Layer Neurons

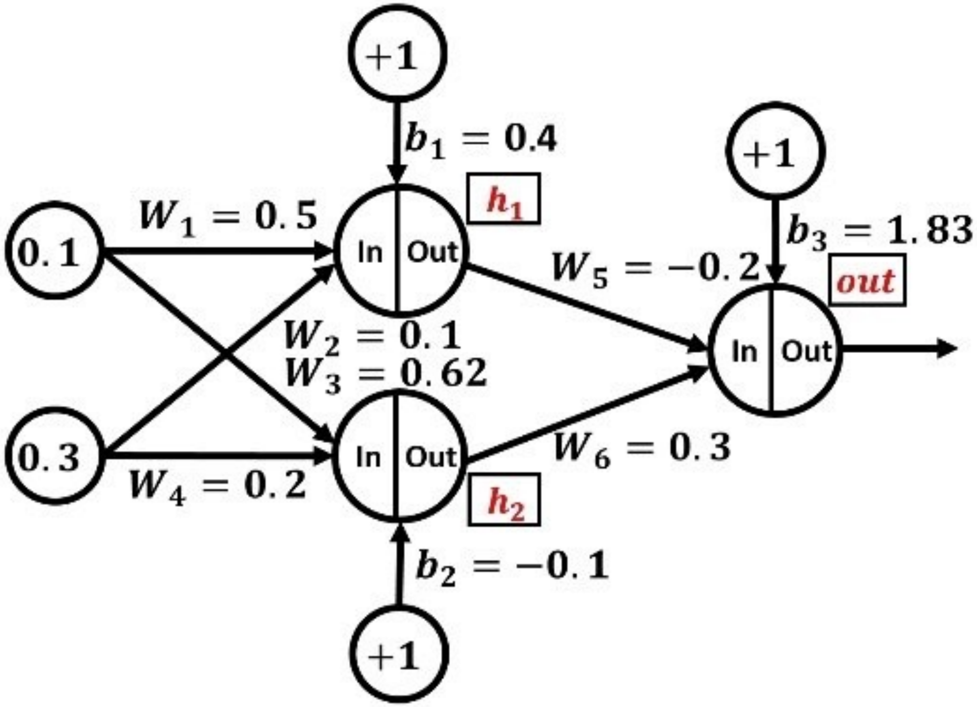
h_1

In

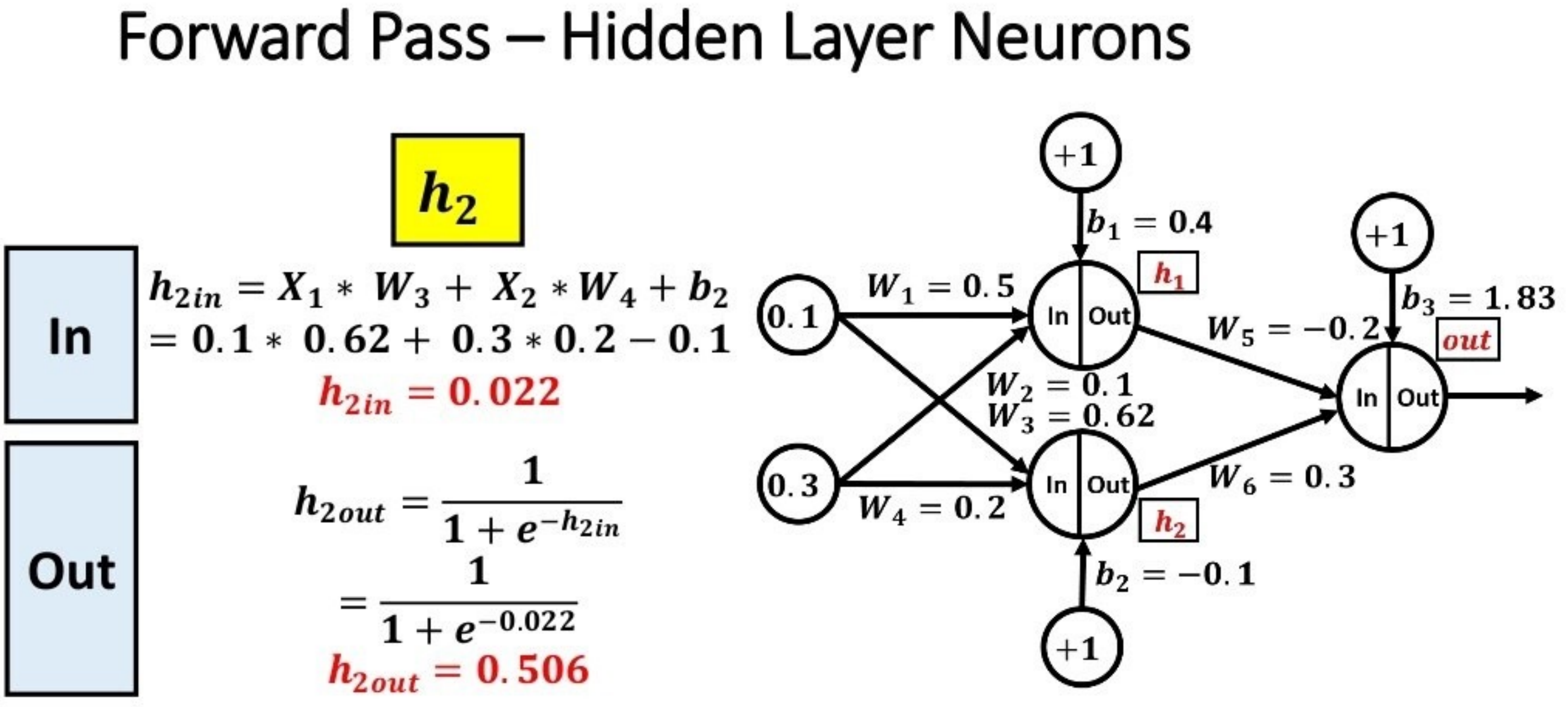
$$h_{1in} = X_1 * W_1 + X_2 * W_2 + b_1$$
$$= 0.1 * 0.5 + 0.3 * 0.1 + 0.4$$
$$h_{1in} = 0.48$$

Out

$$h_{1out} = \frac{1}{1 + e^{-h_{1in}}}$$
$$= \frac{1}{1 + e^{-0.48}}$$
$$h_{1out} = 0.618$$



Forward Pass



Forward Pass – Hidden Layer Neurons

h_2

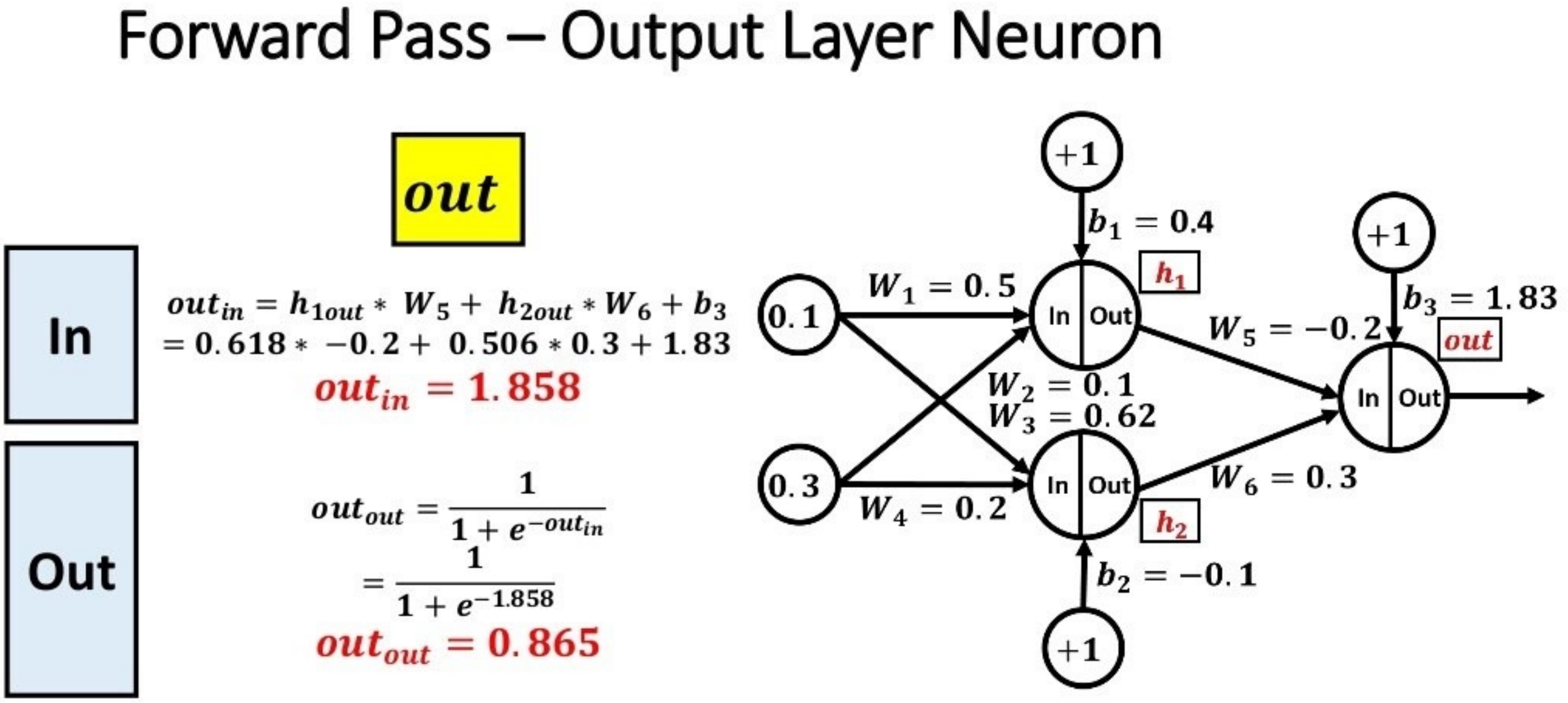
In

$$h_{2in} = X_1 * \ W_3 + \ X_2 * W_4 + b_2$$
$$= 0.1 * \ 0.62 + \ 0.3 * 0.2 - 0.1$$
$$h_{2in} = 0.022$$

Out

$$h_{2out} = \frac{1}{1 + e^{-h_{2in}}}$$
$$= \frac{1}{1 + e^{-0.022}}$$
$$h_{2out} = 0.506$$

Forward Pass



Forward Pass – Output Layer Neuron

out

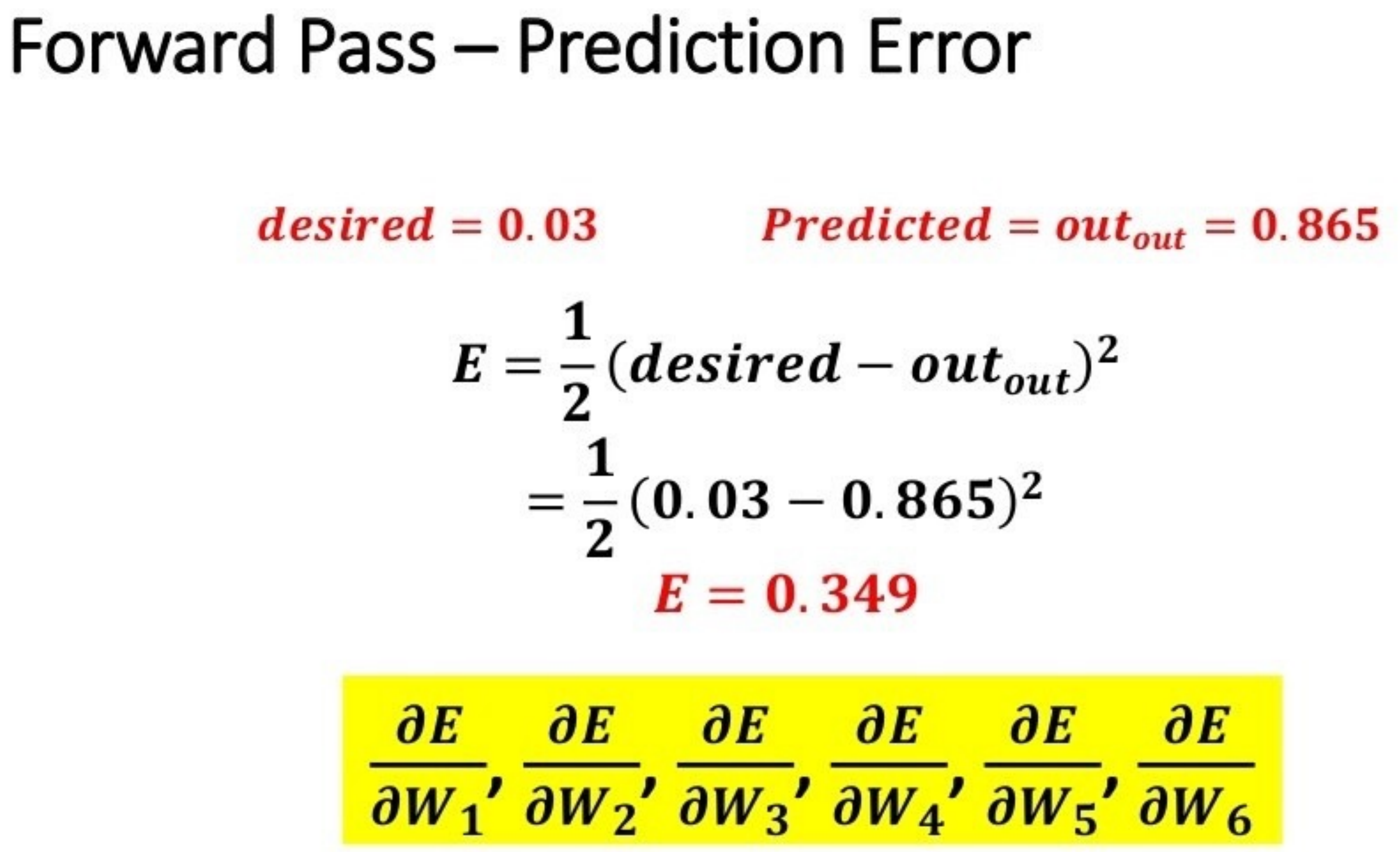
In

$$out_{in} = h_{1out} * W_5 + h_{2out} * W_6 + b_3$$
$$= 0.618 * -0.2 + 0.506 * 0.3 + 1.83$$
$$out_{in} = 1.858$$

Out

$$out_{out} = \frac{1}{1 + e^{-out_{in}}}$$
$$= \frac{1}{1 + e^{-1.858}}$$
$$out_{out} = 0.865$$

Forward Pass



Forward Pass – Prediction Error

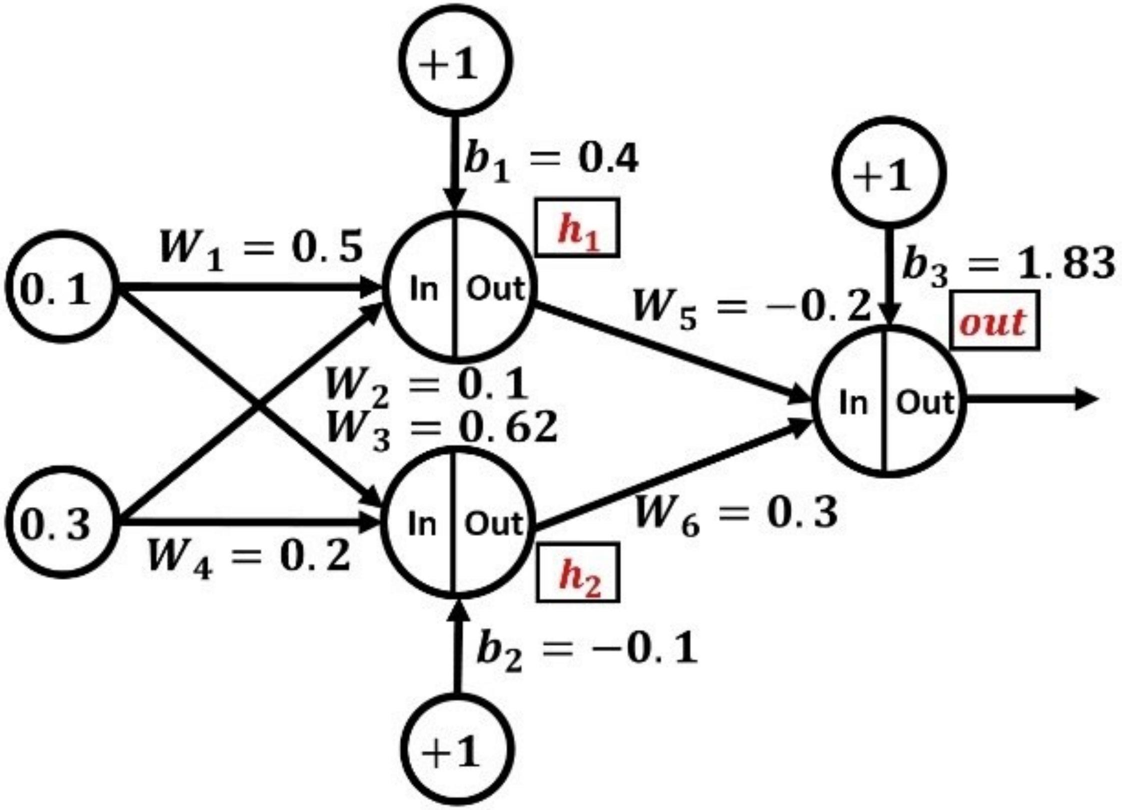
$\boldsymbol{desired = 0.03}$ $\qquad$ $\boldsymbol{Predicted = out_{out} = 0.865}$

$$E = \frac{1}{2}(desired - out_{out})^2$$
$$= \frac{1}{2}(0.03 - 0.865)^2$$
$$E = 0.349$$

$$\boxed{\frac{\partial E}{\partial W_1}, \frac{\partial E}{\partial W_2}, \frac{\partial E}{\partial W_3}, \frac{\partial E}{\partial W_4}, \frac{\partial E}{\partial W_5}, \frac{\partial E}{\partial W_6}}$$

Backward Pass

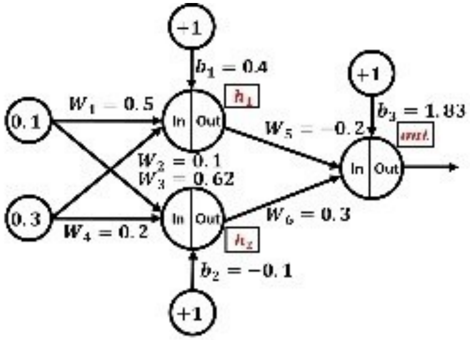
Partial Derivatives Calculation



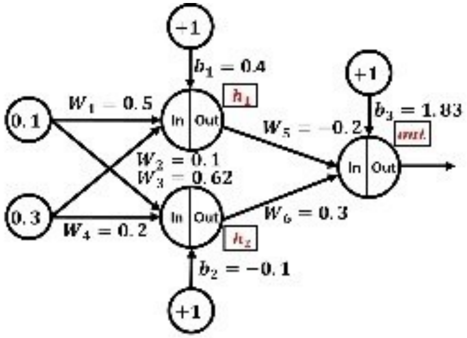
Backward Pass

E$-W_5$ $(\frac{\partial E}{\partial W_5})$ Parial Derivative

$$\frac{\partial E}{\partial W_5} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial W_5}$$



Backward Pass



E$-W_5$ ($\frac{\partial E}{\partial W_5}$) Parial Derivative

$$\frac{\partial E}{\partial W_5} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial W_5}$$

Partial Derivative

$$\frac{\partial E}{\partial out_{out}} = \frac{\partial}{\partial out_{out}}(\frac{1}{2}(desired - out_{out})^2)$$

$$= 2 * \frac{1}{2}(desired - out_{out})^{2-1} * (0 - 1)$$

$$= desired - out_{out} * (-1)$$

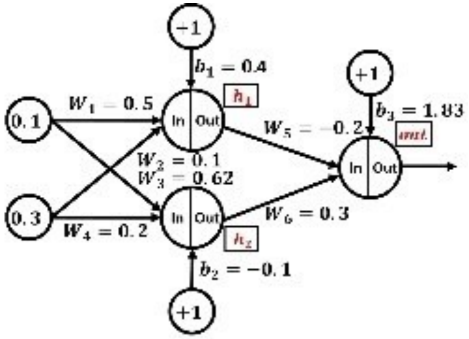
$$\frac{\partial E}{\partial out_{out}} = out_{out} - desired$$

Substitution

$$\frac{\partial E}{\partial out_{out}} = out_{out} - desired = 0.865 - 0.03$$

$$\frac{\partial E}{\partial out_{out}} = 0.835$$

Backward Pass



$$\text{E}-W_5\ (\frac{\partial E}{\partial W_5})\ \text{Parial Derivative}$$

$$\frac{\partial E}{\partial W_5} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial W_5}$$

Partial Derivative

$$\frac{\partial out_{out}}{\partial out_{in}} = \frac{\partial}{\partial out_{in}}(\frac{1}{1+e^{-out_{in}}})$$

$$\frac{\partial out_{out}}{\partial out_{in}} = (\frac{1}{1+e^{-out_{in}}})(1-\frac{1}{1+e^{-out_{in}}})$$

Substitution

$$\frac{\partial out_{out}}{\partial out_{in}} = (\frac{1}{1+e^{-1.858}})(1-\frac{1}{1+e^{-1.858}})$$

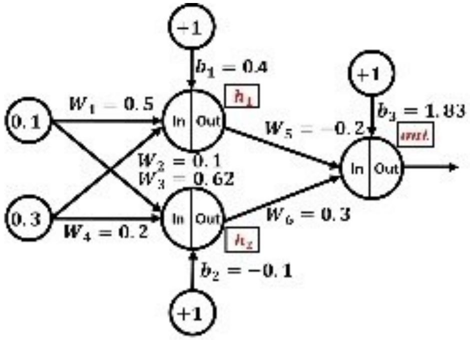
$$= (\frac{1}{1.56})(1-\frac{1}{1.56})$$

$$= (0.641)(1-0.641) = (0.641)(0.359)$$

$$\frac{\partial out_{out}}{\partial out_{in}} = 0.23$$

Backward Pass

E$-W_5$ $(\frac{\partial E}{\partial W_5})$ Parial Derivative



$$\frac{\partial E}{\partial W_5} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial W_5}$$

Partial Derivative

$$\frac{\partial out_{in}}{\partial W_5} = \frac{\partial}{\partial W_5}(h_{1out} * W_5 + h_{2out} * W_6 + b_3)$$

$$= 1 * h_{1out} * (W_5)^{1-1} + 0 + 0$$

$$\frac{\partial out_{in}}{\partial W_5} = h_{1out}$$

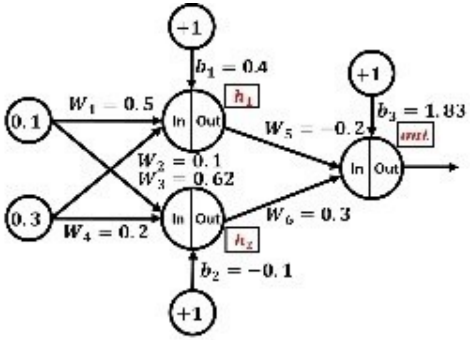
Substitution

$$\frac{\partial out_{in}}{\partial W_5} = h_{1out}$$

$$\frac{\partial out_{in}}{\partial W_5} = 0.618$$

Backward Pass

$$E-W_5\ \left(\frac{\partial E}{\partial W_5}\right)$$ Parial Derivative



$$\frac{\partial E}{\partial W_5} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial W_5}$$

$$\frac{\partial E}{\partial out_{out}} = 0.835 \qquad \frac{\partial out_{out}}{\partial out_{in}} = 0.23 \qquad \frac{\partial out_{in}}{\partial W_5} = 0.618$$

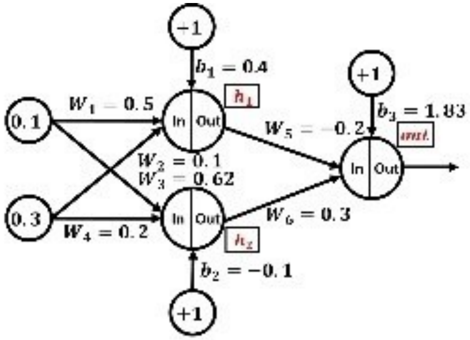
$$\frac{\partial E}{\partial W_5} = 0.835 * 0.23 * 0.618$$

$$\frac{\partial E}{\partial W_5} = 0.119$$

Backward Pass

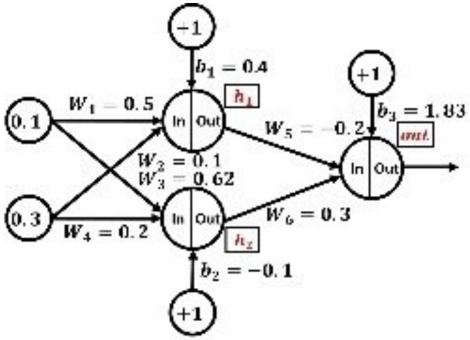
E$-W_6$ ($\frac{\partial E}{\partial W_6}$) Parial Derivative

$$\frac{\partial E}{\partial W_6} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial W_6}$$



Backward Pass

E$-W_6$ $(\frac{\partial E}{\partial W_6})$ Parial Derivative

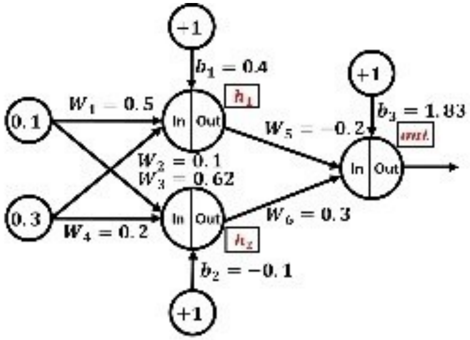


$$\frac{\partial E}{\partial W_6} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial W_6}$$

$$\frac{\partial E}{\partial out_{out}} = 0.835 \qquad \frac{\partial out_{out}}{\partial out_{in}} = 0.23$$

Backward Pass

E$-W_6$ $\left(\frac{\partial E}{\partial W_6}\right)$ Parial Derivative



$$\frac{\partial E}{\partial W_5} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial W_6}$$

Partial Derivative

$$\frac{\partial out_{in}}{\partial W_6} = \frac{\partial}{\partial W_6}(h_{1out} * W_5 + h_{2out} * W_6 + b_3)$$

$$= 0 + 1 * h_{2out} * (W_6)^{1-1} + 0$$

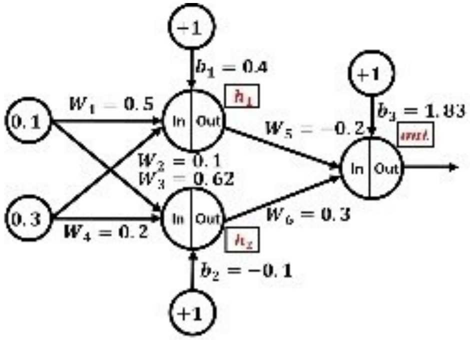
$$\frac{\partial out_{in}}{\partial W_6} = h_{2out}$$

Substitution

$$\frac{\partial out_{in}}{\partial W_6} = h_{2out}$$

$$\frac{\partial out_{in}}{\partial W_6} = 0.506$$

Backward Pass



E$-W_6$ $(\frac{\partial E}{\partial W_6})$ Parial Derivative

$$\frac{\partial E}{\partial W_6} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial W_6}$$

$$\frac{\partial E}{\partial out_{out}} = 0.835 \qquad \frac{\partial out_{out}}{\partial out_{in}} = 0.23 \qquad \frac{\partial out_{in}}{\partial W_6} = 0.506$$

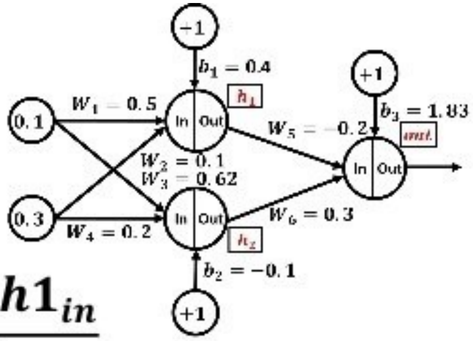
$$\frac{\partial E}{\partial W_6} = 0.835 * 0.23 * \ 0.506$$

$$\frac{\partial E}{\partial W_6} = 0.097$$

Backward Pass

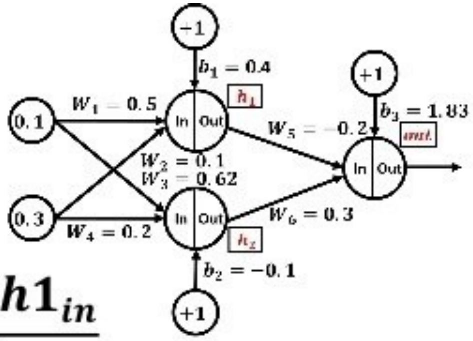
$\mathrm{E}{-}W_1$ $(\frac{\partial E}{\partial W_1})$ Parial Derivative

$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_1}$$



Backward Pass

$$\text{E}-W_1\ \left(\frac{\partial E}{\partial W_1}\right)\ \text{Parial Derivative}$$

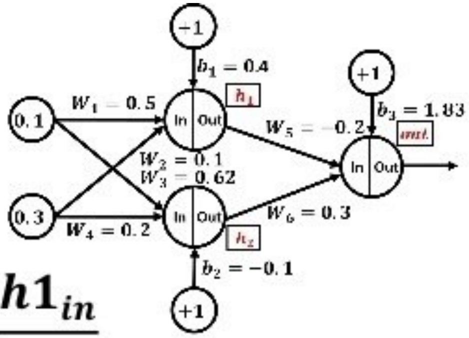


$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_1}$$

$$\frac{\partial E}{\partial out_{out}} = 0.835 \qquad \frac{\partial out_{out}}{\partial out_{in}} = 0.23$$

Backward Pass

E$-W_1$ $\left(\frac{\partial E}{\partial W_1}\right)$ Parial Derivative



$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_1}$$

Partial Derivative

$$\frac{\partial out_{in}}{\partial h1_{out}} = \frac{\partial}{\partial h1_{out}}(h_{1out} * \ W_5 + \ h_{2out} * W_6 + b_3)$$

$$= (h_{1out})^{1-1} * \ W_5 + 0 + 0$$

$$\frac{\partial out_{in}}{\partial h1_{out}} = W_5$$

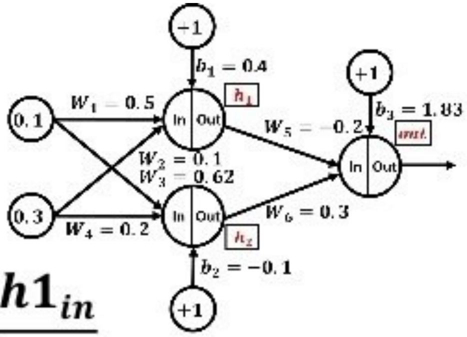
Substitution

$$\frac{\partial out_{in}}{\partial h1_{out}} = W_5$$

$$\frac{\partial out_{in}}{\partial h1_{out}} = -0.2$$

Backward Pass

E$-W_1$ ($\frac{\partial E}{\partial W_1}$) Parial Derivative



$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_1}$$

Partial Derivative

$$\frac{\partial h1_{out}}{\partial h1_{in}} = \frac{\partial}{\partial h_{1in}}\left(\frac{1}{1+e^{-h_{1in}}}\right)$$

$$\frac{\partial h1_{out}}{\partial h1_{in}} = \left(\frac{1}{1+e^{-h_{1in}}}\right)\left(1-\frac{1}{1+e^{-h_{1in}}}\right)$$

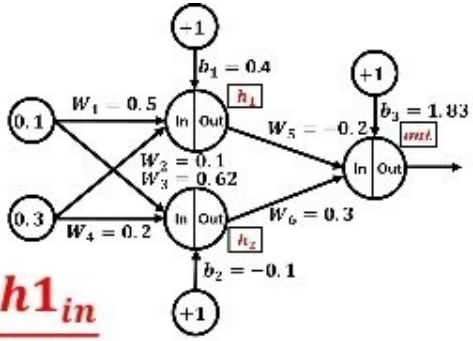
Substitution

$$\frac{\partial h1_{out}}{\partial h1_{in}} = \left(\frac{1}{1+e^{-h_{1in}}}\right)\left(1-\frac{1}{1+e^{-h_{1in}}}\right)$$

$$= \left(\frac{1}{1+e^{-0.48}}\right)\left(1-\frac{1}{1+e^{-0.48}}\right)$$

$$\frac{\partial h_{2out}}{\partial h_{2in}} = 0.236$$

Backward Pass



E$-W_1$ $\left(\frac{\partial E}{\partial W_1}\right)$ Parial Derivative

$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_1}$$

Partial Derivative

$$\frac{\partial h1_{in}}{\partial W_1} = \frac{\partial}{\partial W_1}(X_1 * W_1 + X_2 * W_2 + b_1)$$

$$= X_1 * (W_1)^{1-1} + 0 + 0$$

$$\frac{\partial h1_{in}}{\partial W_1} = X_1$$

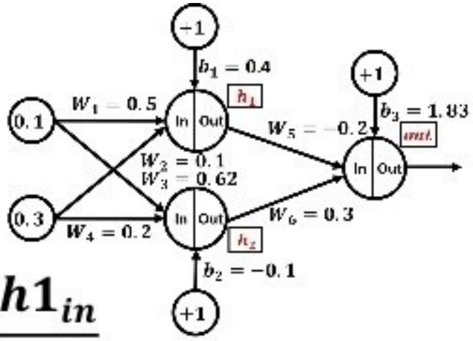
Substitution

$$\frac{\partial h1_{in}}{\partial W_1} = X_1$$

$$\frac{\partial h1_{in}}{\partial W_1} = 0.1$$

Backward Pass

$$\mathrm{E}-W_1\ \left(\frac{\partial E}{\partial W_1}\right)$$ Parial Derivative



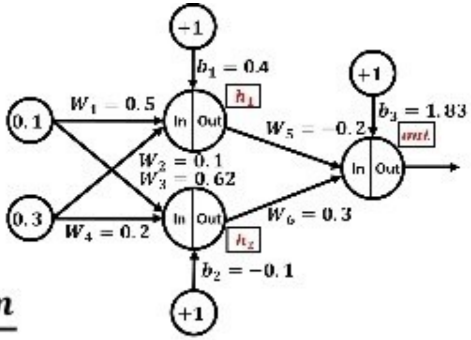
$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_1}$$

$$\frac{\partial E}{\partial out_{out}} = 0.835 \qquad \frac{\partial out_{out}}{\partial out_{in}} = 0.23 \qquad \frac{\partial out_{in}}{\partial h1_{out}} = -0.2 \qquad \frac{\partial h_{2out}}{\partial h_{2in}} = 0.236 \qquad \frac{\partial h1_{in}}{\partial W_1} = 0.1$$

$$\frac{\partial E}{\partial W_1} = 0.835 * 0.23 * -0.2 * 0.236 * 0.1$$

$$\frac{\partial E}{\partial W_1} = -0.001$$

Backward Pass

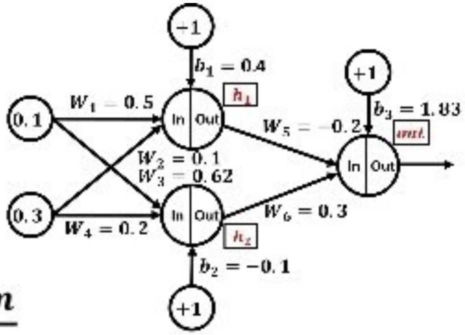


E−W_2 ($\frac{\partial E}{\partial W_2}$) Parial Derivative:

$$\frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_2}$$

Backward Pass

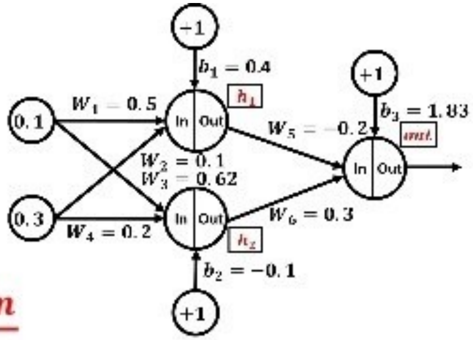
$\text{E}-W_2$ $(\frac{\partial E}{\partial W_2})$ Parial Derivative:



$$\frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_2}$$

$\frac{\partial E}{\partial out_{out}} = 0.835$ $\frac{\partial out_{out}}{\partial out_{in}} = 0.23$ $\frac{\partial out_{in}}{\partial h1_{out}} = -0.2$ $\frac{\partial h_{2out}}{\partial h_{2in}} = 0.236$

Backward Pass



E$-W_2$ ($\frac{\partial E}{\partial W_2}$) Parial Derivative:

$$\frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_2}$$

Partial Derivative

$$\frac{\partial h1_{in}}{\partial W_2} = \frac{\partial}{\partial W_2}(X_1 * W_1 + X_2 * W_2 + b_1)$$

$$= 0 + X_2 * (W_2)^{1-1} + 0$$

$$\frac{\partial h1_{in}}{\partial W_2} = X_2$$

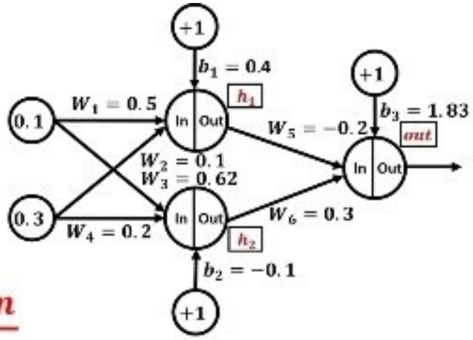
Substitution

$$\frac{\partial h1_{in}}{\partial W_2} = X_2$$

$$\frac{\partial h1_{in}}{\partial W_2} = 0.3$$

Backward Pass

E$-W_2$ $(\frac{\partial E}{\partial W_2})$ Parial Derivative:



$$\frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h1_{out}} * \frac{\partial h1_{out}}{\partial h1_{in}} * \frac{\partial h1_{in}}{\partial W_2}$$

$$\frac{\partial E}{\partial out_{out}} = 0.835 \qquad \frac{\partial out_{out}}{\partial out_{in}} = 0.23 \qquad \frac{\partial out_{in}}{\partial h1_{out}} = -0.2 \qquad \frac{\partial h_{2out}}{\partial h_{2in}} = 0.236 \qquad \frac{\partial h1_{in}}{\partial W_2} = 0.3$$

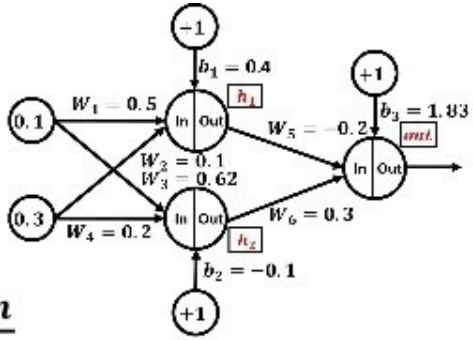
$$\frac{\partial E}{\partial W_2} = 0.835 * 0.23 * -0.2 * 0.236 * 0.3$$

$$\frac{\partial E}{\partial W_2} = -.003$$

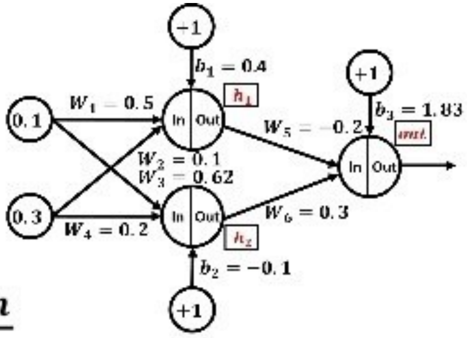
Backward Pass

E$-W_3$ ($\frac{\partial E}{\partial W_3}$) Parial Derivative:

$$\frac{\partial E}{\partial W_3} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_3}$$



Backward Pass



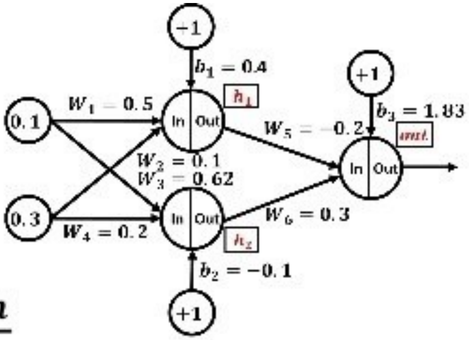
E$-W_3$ ($\frac{\partial E}{\partial W_3}$) Parial Derivative:

$$\frac{\partial E}{\partial W_3} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_3}$$

$$\frac{\partial E}{\partial out_{out}} = 0.835 \qquad \frac{\partial out_{out}}{\partial out_{in}} = 0.23$$

Backward Pass

E$-W_3$ ($\frac{\partial E}{\partial W_3}$) Parial Derivative:



$$\frac{\partial E}{\partial W_3} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_3}$$

Partial Derivative

$$\frac{\partial out_{in}}{\partial h2_{out}} = \frac{\partial}{\partial h2_{out}}(h_{1out} * W_5 + h_{2out} * W_6 + b_3)$$

$$= 0 + (h_{2out})^{1-1} * W_6 + 0$$

$$\frac{\partial out_{in}}{\partial h2_{out}} = W_6$$

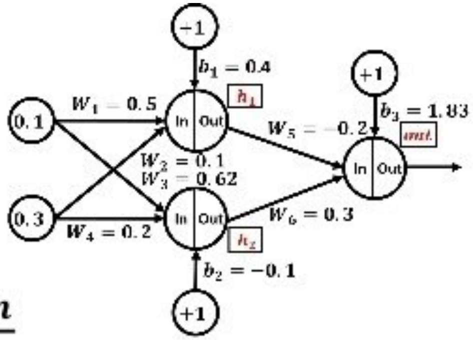
Substitution

$$\frac{\partial out_{in}}{\partial h2_{out}} = W_6$$

$$\frac{\partial out_{in}}{\partial h2_{out}} = 0.3$$

Backward Pass

E$-W_3$ $(\frac{\partial E}{\partial W_3})$ Parial Derivative:



$$\frac{\partial E}{\partial W_3} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_3}$$

Partial Derivative

$$\frac{\partial h2_{out}}{\partial h2_{in}} = \frac{\partial}{\partial h_{2in}} (\frac{1}{1 + e^{-h_{2in}}})$$

$$\frac{\partial h2_{out}}{\partial h2_{in}} = (\frac{1}{1 + e^{-h_{2in}}})(1 - \frac{1}{1 + e^{-h_{2in}}})$$

Substitution

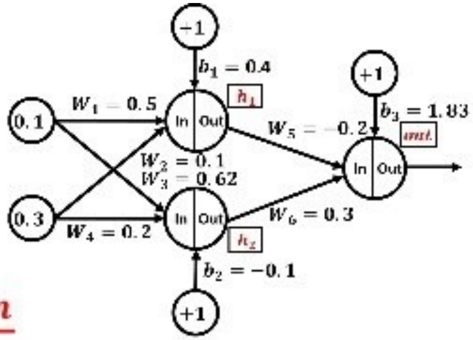
$$\frac{\partial h2_{out}}{\partial h2_{in}} = (\frac{1}{1 + e^{-h_{2in}}})(1 - \frac{1}{1 + e^{-h_{2in}}})$$

$$= (\frac{1}{1 + e^{-0.022}})(1 - \frac{1}{1 + e^{-0.022}})$$

$$\frac{\partial h_{2out}}{\partial h_{2in}} = 0.25$$

Backward Pass

E$-W_3$ $(\frac{\partial E}{\partial W_3})$ Parial Derivative:



$$\frac{\partial E}{\partial W_3} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_3}$$

Partial Derivative

$$\frac{\partial h2_{in}}{\partial W_3} = \frac{\partial}{\partial W_3}(X_1 * \ W_3 + \ X_2 * W_4 + b_2)$$

$$= X_1 * \ W_3 + \ X_2 * W_4 + b_2$$

$$= (X_1)^{1-1} * \ W_3 + 0 + 0$$

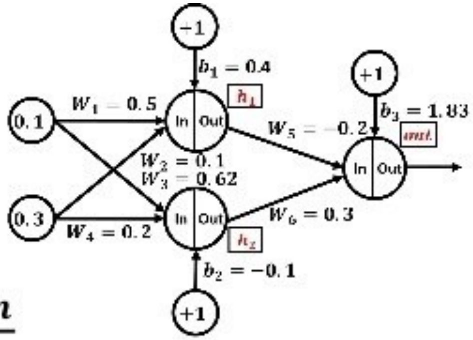
$$\frac{\partial h2_{in}}{\partial W_3} = W_3$$

Substitution

$$\frac{\partial h2_{in}}{\partial W_3} = W_3$$

$$\frac{\partial h2_{in}}{\partial W_3} = 0.62$$

Backward Pass



E$-W_3$ ($\frac{\partial E}{\partial W_3}$) Parial Derivative:

$$\frac{\partial E}{\partial W_3} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_3}$$

$\frac{\partial E}{\partial out_{out}} = 0.835$ $\frac{\partial out_{out}}{\partial out_{in}} = 0.23$ $\frac{\partial out_{in}}{\partial h2_{out}} = 0.3$ $\frac{\partial h_{2out}}{\partial h_{2in}} = 0.25$ $\frac{\partial h2_{in}}{\partial W_3} = 0.62$

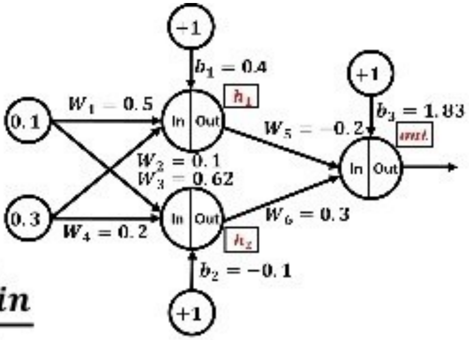
$$\frac{\partial E}{\partial W_3} = 0.835 * 0.23 * 0.3 * 0.25 * 0.62$$

$$\frac{\partial E}{\partial W_3} = 0.009$$

Backward Pass

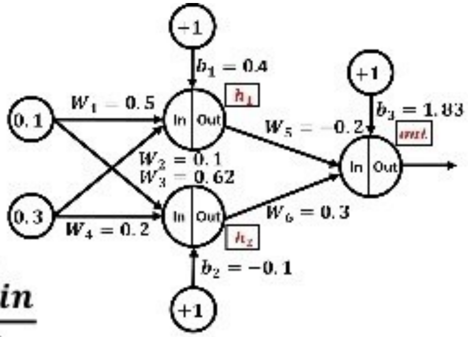
E$-W_4$ $(\frac{\partial E}{\partial W_4})$ Parial Derivative:

$$\frac{\partial E}{\partial W_4} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_4}$$



Backward Pass

$\text{E}-W_4$ $(\frac{\partial E}{\partial W_4})$ Parial Derivative:

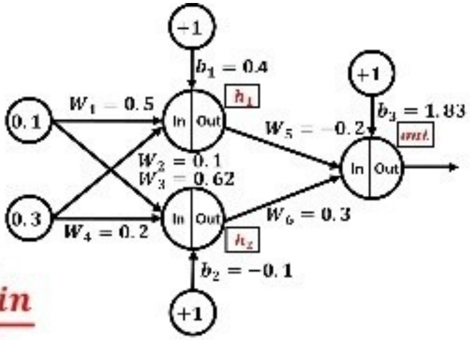


$$\frac{\partial E}{\partial W_4} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_4}$$

$$\frac{\partial E}{\partial out_{out}} = 0.835 \qquad \frac{\partial out_{out}}{\partial out_{in}} = 0.23 \qquad \frac{\partial out_{in}}{\partial h2_{out}} = 0.3 \qquad \frac{\partial h_{2out}}{\partial h_{2in}} = 0.25$$

Backward Pass

E$-W_4$ ($\frac{\partial E}{\partial W_4}$) Parial Derivative:



$$\frac{\partial E}{\partial W_4} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_4}$$

Partial Derivative

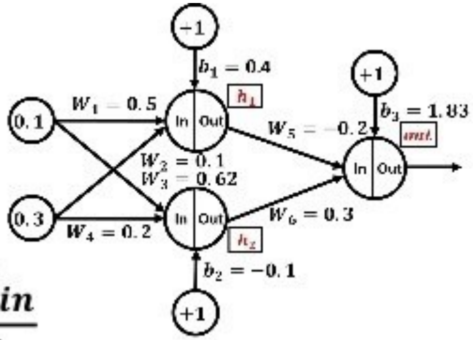
$$\frac{\partial h2_{in}}{\partial W_4} = \frac{\partial}{\partial W_4}(X_1 * W_3 + X_2 * W_4 + b_2)$$
$$= X_1 * W_3 + X_2 * W_4 + b_2$$
$$= 0 + (X_2)^{1-1} * W_4 + 0$$
$$\frac{\partial h2_{in}}{\partial W_4} = W_4$$

Substitution

$$\frac{\partial h2_{in}}{\partial W_4} = W_4$$
$$\frac{\partial h2_{in}}{\partial W_4} = 0.2$$

Backward Pass

E$-W_4$ ($\frac{\partial E}{\partial W_4}$) Parial Derivative:



$$\frac{\partial E}{\partial W_4} = \frac{\partial E}{\partial out_{out}} * \frac{\partial out_{out}}{\partial out_{in}} * \frac{\partial out_{in}}{\partial h2_{out}} * \frac{\partial h2_{out}}{\partial h2_{in}} * \frac{\partial h2_{in}}{\partial W_4}$$

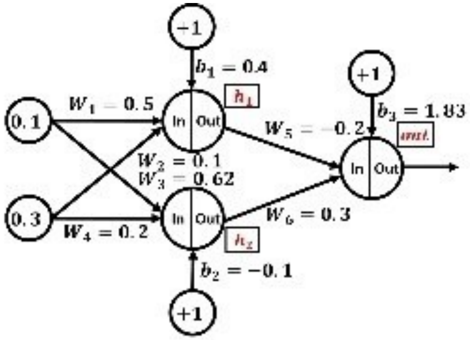
$\frac{\partial E}{\partial out_{out}} = 0.835$ $\frac{\partial out_{out}}{\partial out_{in}} = 0.23$ $\frac{\partial out_{in}}{\partial h2_{out}} = 0.3$ $\frac{\partial h_{2out}}{\partial h_{2in}} = 0.25$ $\frac{\partial h2_{in}}{\partial W_4} = 0.2$

$$\frac{\partial E}{\partial W_4} = 0.835 * 0.23 * 0.3 * 0.25 * 0.2$$

$$\frac{\partial E}{\partial W_4} = 0.003$$

Backward Pass

All Error-Weights Partial Derivatives

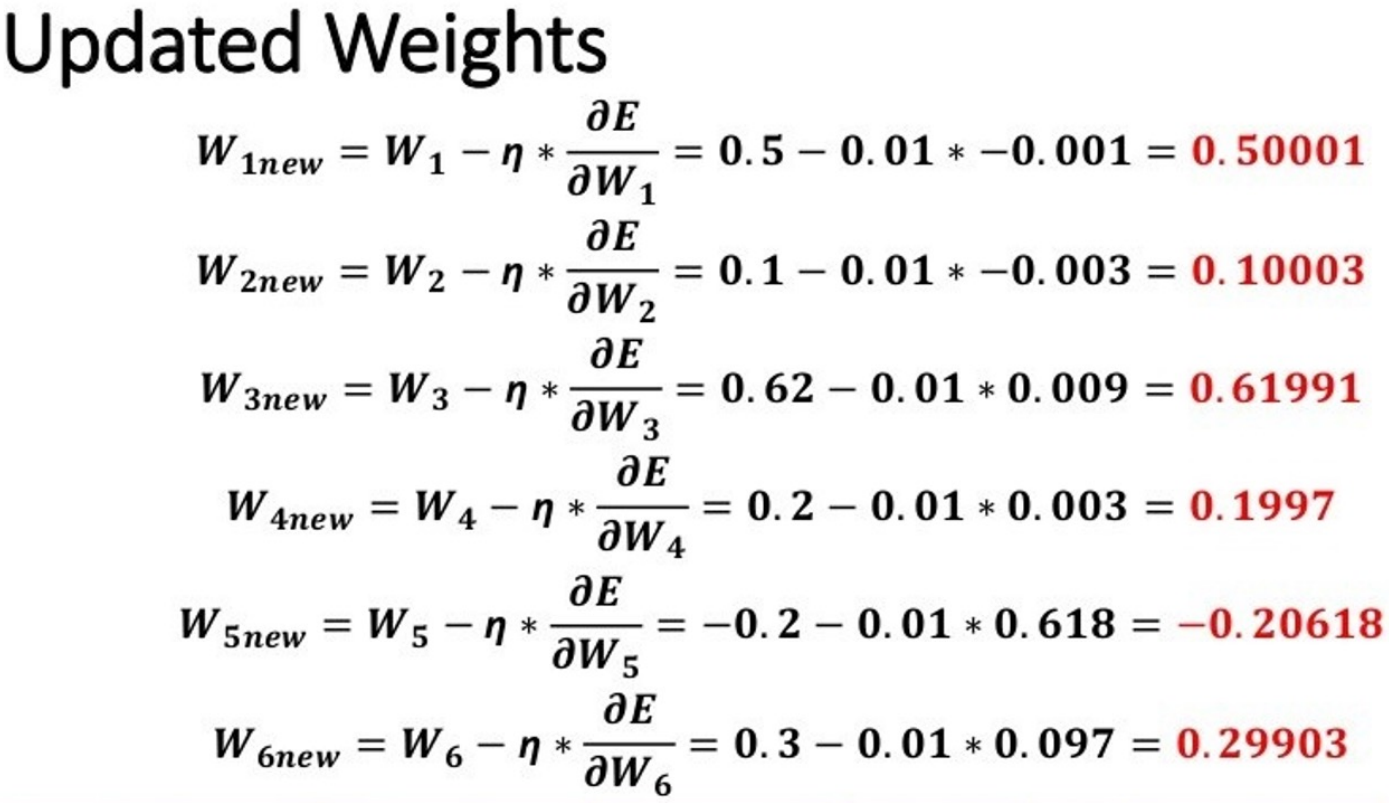


$$\frac{\partial E}{\partial W_1} = -0.001 \qquad \frac{\partial E}{\partial W_2} = -.003$$

$$\frac{\partial E}{\partial W_3} = 0.009 \qquad \frac{\partial E}{\partial W_4} = 0.003$$

$$\frac{\partial E}{\partial W_5} = 0.119 \qquad \frac{\partial E}{\partial W_6} = 0.097$$

Weights Update



Updated Weights

$$W_{1new} = W_1 - \eta * \frac{\partial E}{\partial W_1} = 0.5 - 0.01 * -0.001 = 0.50001$$

$$W_{2new} = W_2 - \eta * \frac{\partial E}{\partial W_2} = 0.1 - 0.01 * -0.003 = 0.10003$$

$$W_{3new} = W_3 - \eta * \frac{\partial E}{\partial W_3} = 0.62 - 0.01 * 0.009 = 0.61991$$

$$W_{4new} = W_4 - \eta * \frac{\partial E}{\partial W_4} = 0.2 - 0.01 * 0.003 = 0.1997$$

$$W_{5new} = W_5 - \eta * \frac{\partial E}{\partial W_5} = -0.2 - 0.01 * 0.618 = -0.20618$$

$$W_{6new} = W_6 - \eta * \frac{\partial E}{\partial W_6} = 0.3 - 0.01 * 0.097 = 0.29903$$

Continue updating weights according to derivatives and re-train the network until reaching an acceptable error.

Outline

- Introduction and Motivation

- Neural Network Architecture
  - The Perceptron
  - MLPs
  - Multi-class Perceptron

- Training MLPs

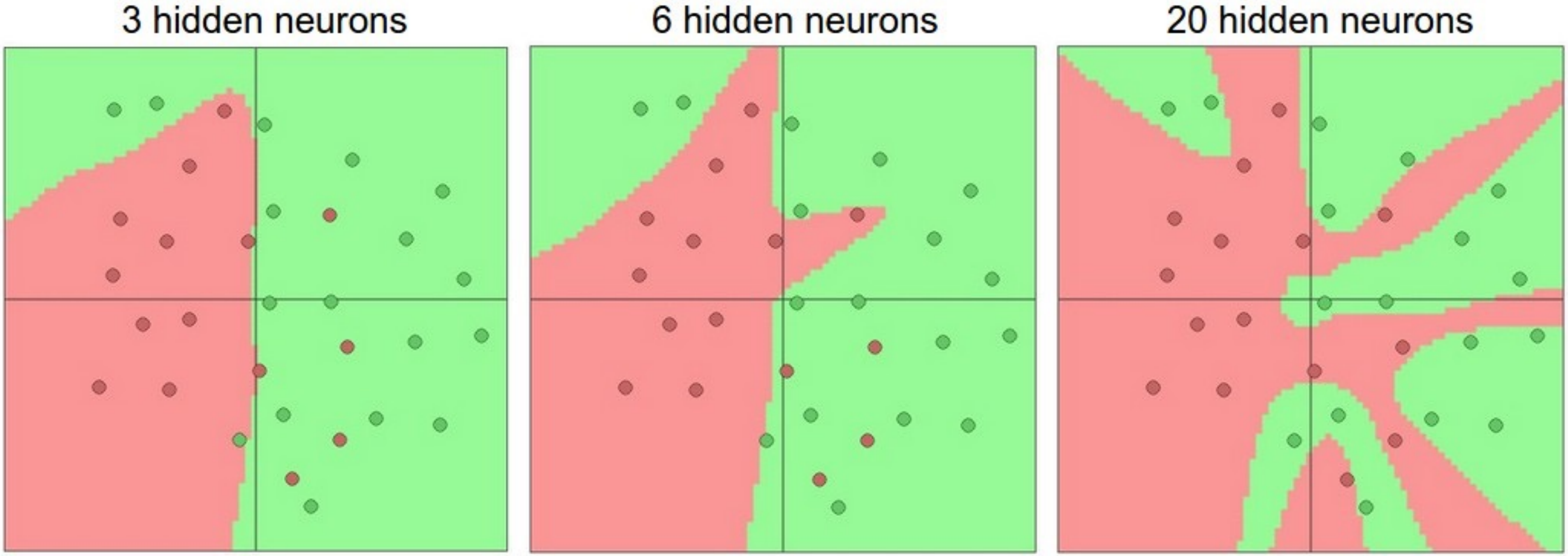
- Choosing Network Structure
  - Depth vs Width
  - Expressive Power of MLPs

MLPs Network Structure

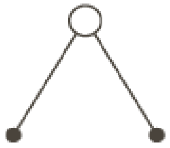
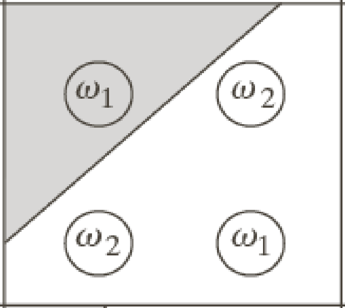
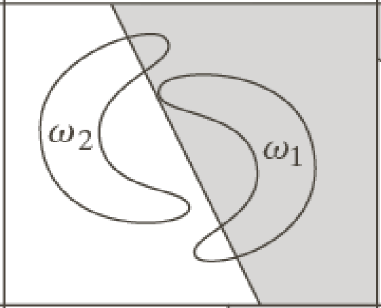
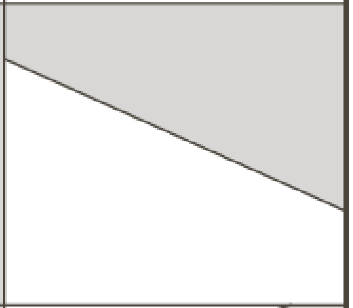
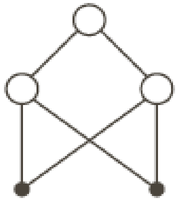
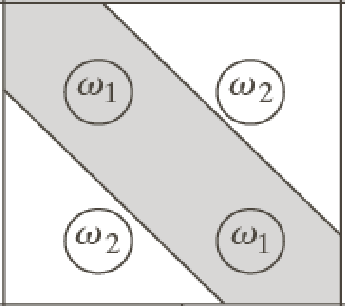
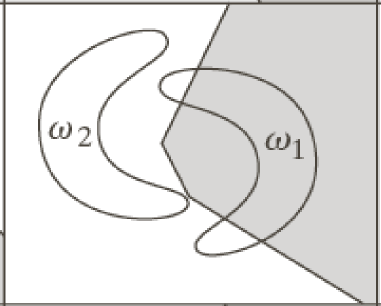
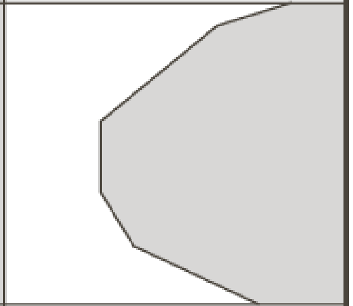
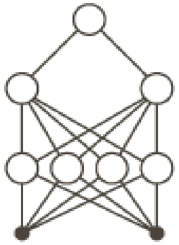
- The number of hidden neurons and the number of layers in MLP have a significant impact on the network's ability to handle different levels of problem complexity, as well as its susceptibility to overfitting and underfitting.
- In complex problems, determining whether to increase the number of hidden neurons (width) or the number of layers (depth) in a neural network depends on various factors.
- **Width vs. Depth**
  - Increasing the number of hidden neurons in a layer allows the network to capture or learn more complex representations or patterns in the data.
    - Learning more complex representations means capturing non-linear relationships, fine-grained patterns, and subtle variations in the input data.
  - Increasing the number of layers enables the network to capture hierarchical features and abstractions.
    - This means that the network's ability to learn and represent information at multiple levels of abstraction. In many real-world problems, data can be organized in a hierarchical manner, where high-level features are built upon lower-level features.
  - There is no one-size-fits-all answer, and often a combination of both approaches may yield the best results.

Number of Neurons in the Hidden Layer

- Larger Neural Networks can represent more complicated functions. The data are shown as circles colored by their class, and the decision regions by a trained neural network are shown underneath.



Number of Hidden Layer

Network structure	Type of decision region	Solution to exclusive-OR problem	Classes with meshed regions	Most general decision surface shapes
Single layer 	Single hyperplane			
Two layers 	Open or closed convex regions			
Three layers 	Arbitrary (complexity limited by the number of nodes)