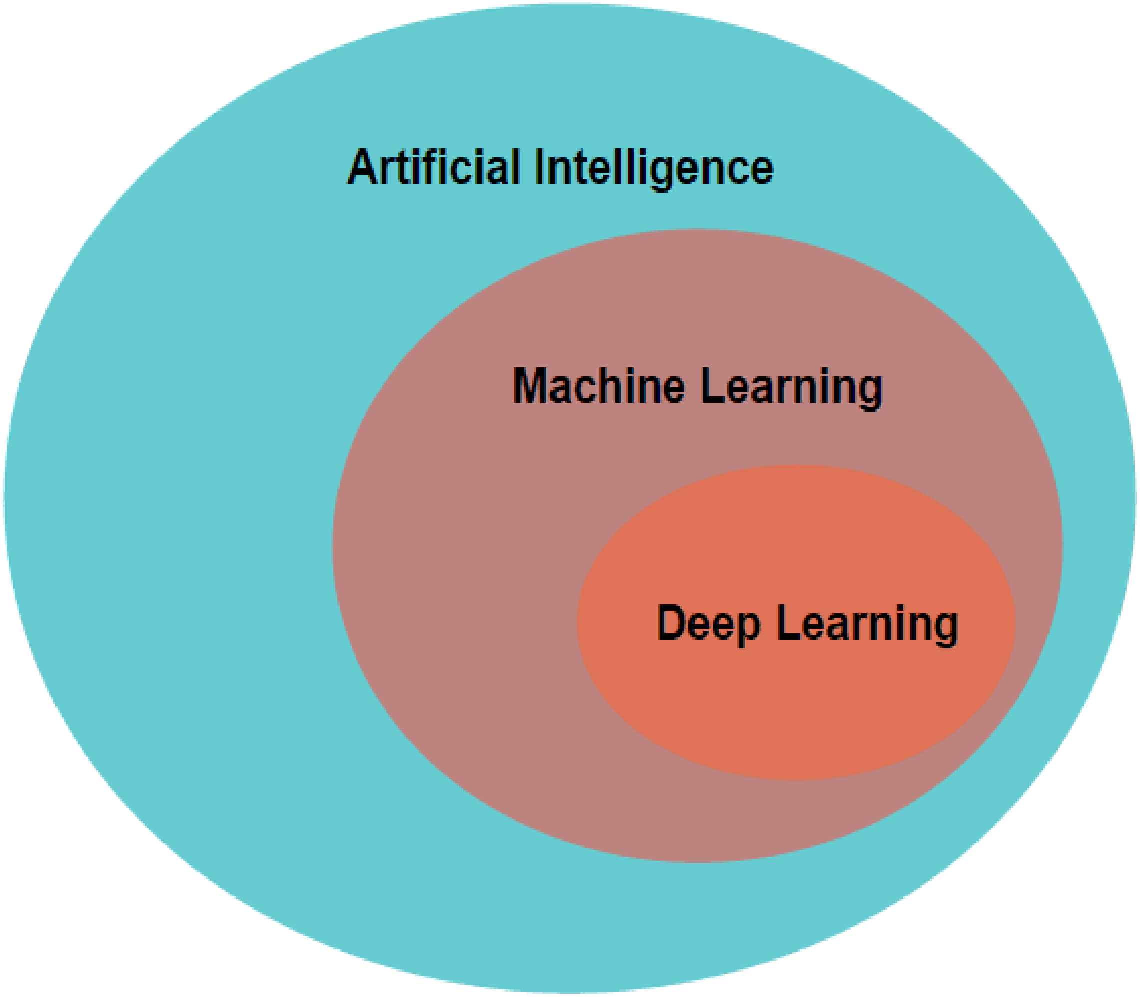
Neural Networks

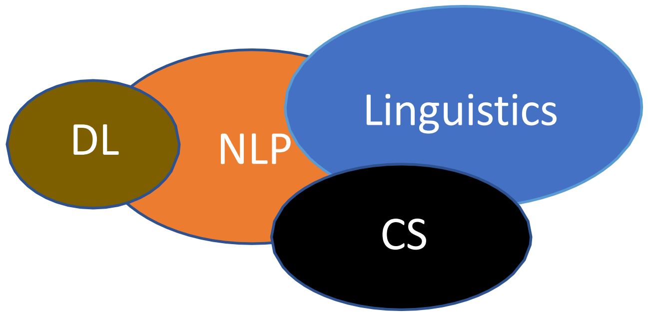
- **Artificial intelligence** is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition, and machine vision.

- **Machine Learning** is the study that uses statistical methods to enable machines to improve with experience.

- **Deep learning** is a subset of machine learning, which is a subset of AI. Artificial intelligence is any computer program that does something smart. Deep Learning is the study that makes use of Neural Networks (similar to neurons present in the human brain) to imitate functionality just like a human brain.



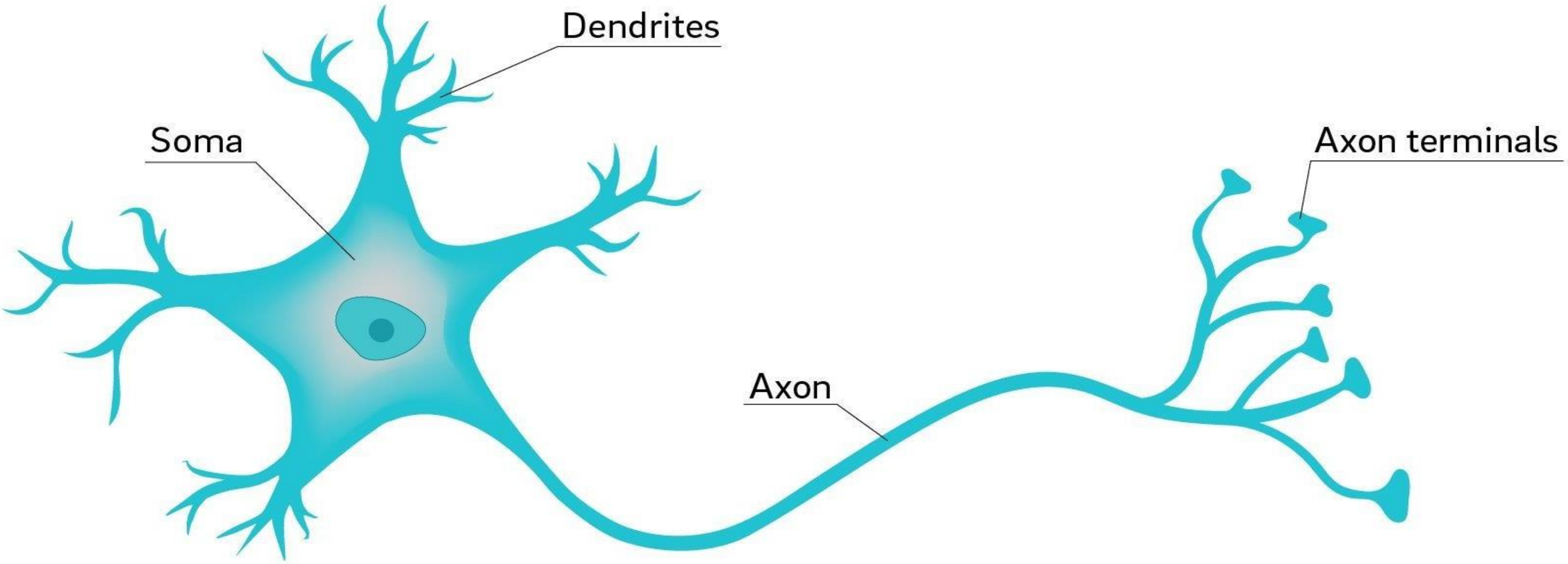
Neural Networks



NLP-powered machine translation helps us to access accurate and reliable translations of foreign texts. Natural language processing is also helping to optimize the process of sentiment analysis. Natural language processing-powered algorithms can understand the meaning behind a text. Voice assistant, Alexa using Natural Language Processing provides a variety of services using artificial intelligence systems equipped through the user's voice commands. NLP Top projects-

- Email filters
- Voice Assistant
- Amazon Alexa
- Google Translator
- Voice Translator
- Text Analysis

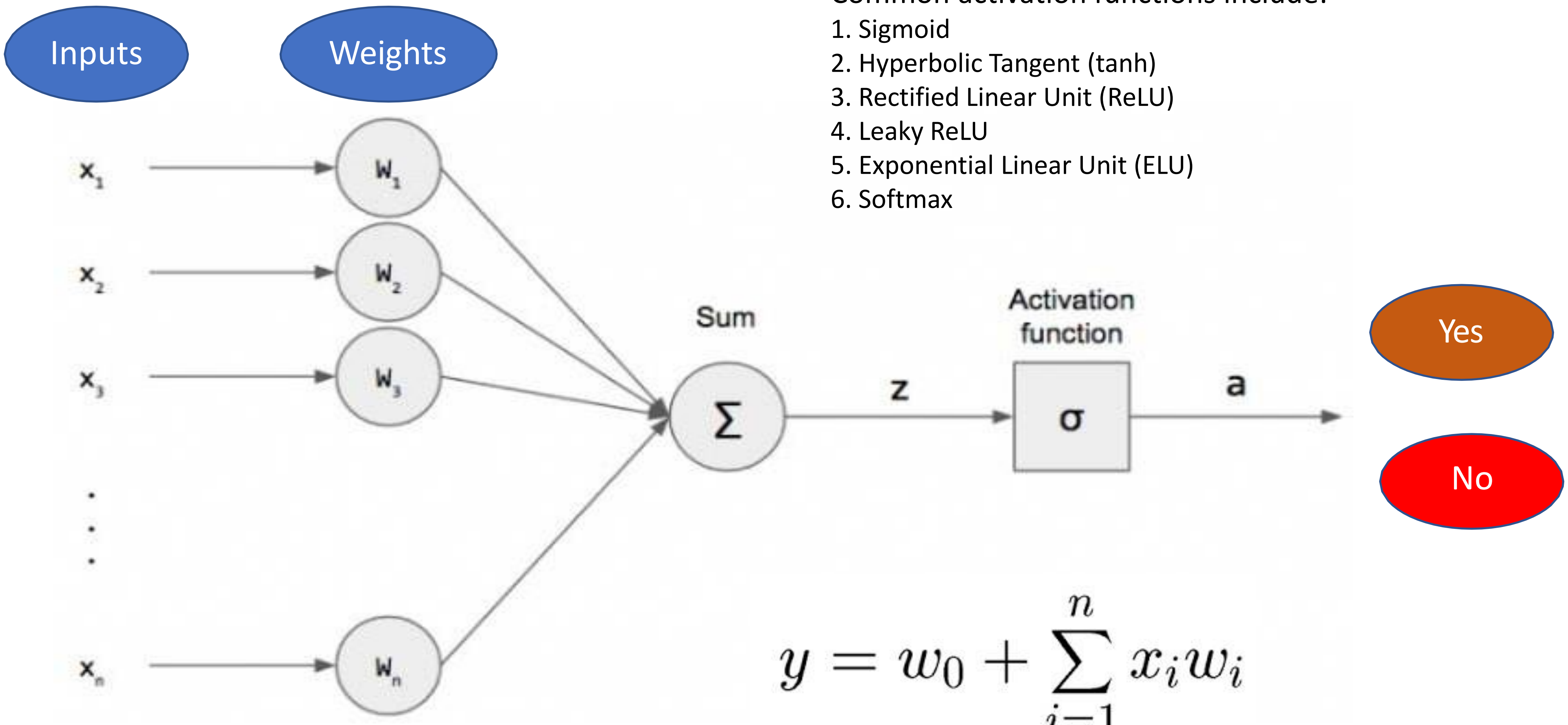
Neural Networks



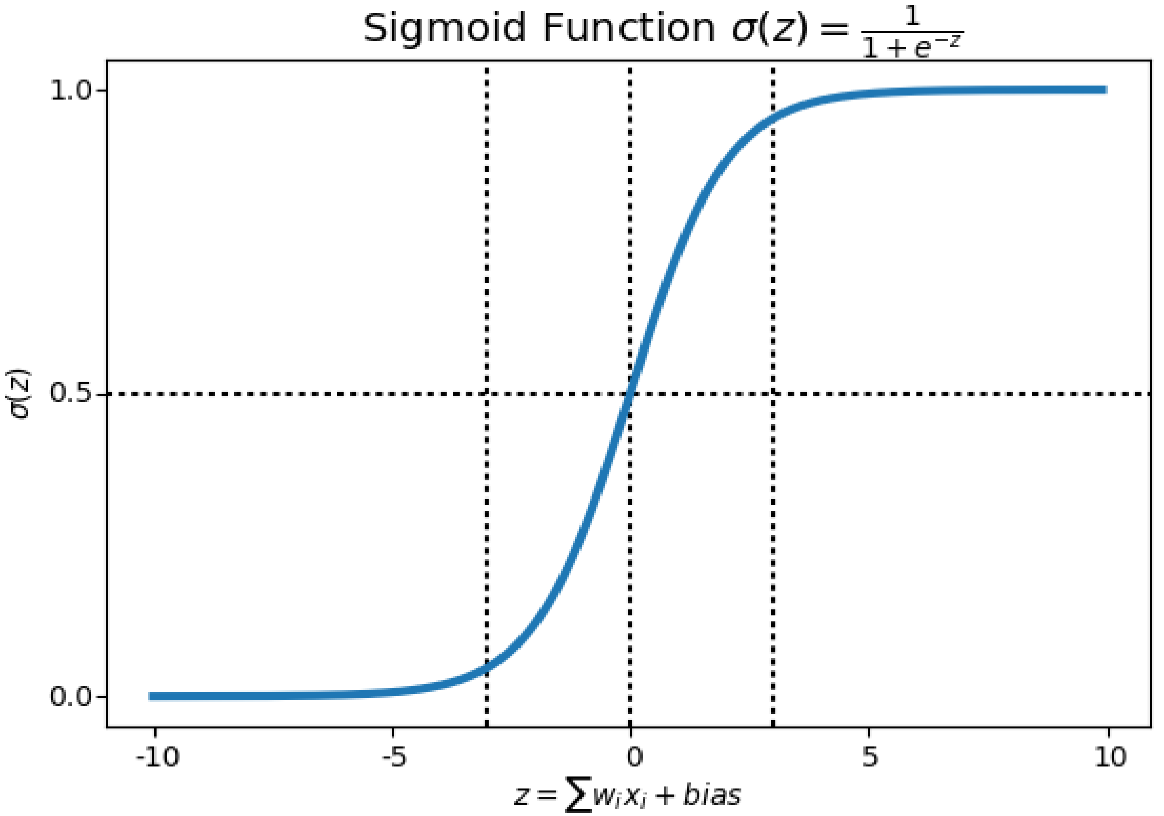
Neural Networks

Common activation functions include:

1. Sigmoid
2. Hyperbolic Tangent (tanh)
3. Rectified Linear Unit (ReLU)
4. Leaky ReLU
5. Exponential Linear Unit (ELU)
6. Softmax

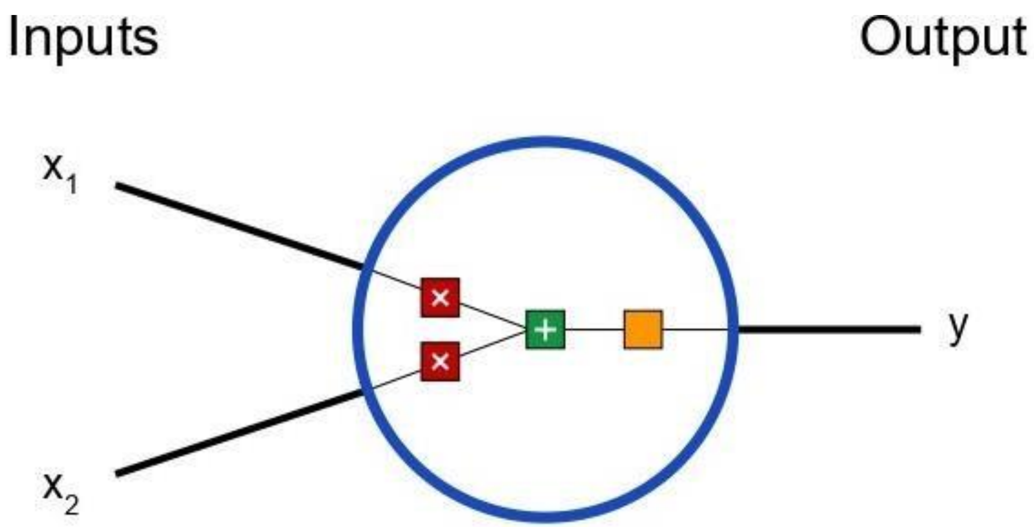


$$y = w_0 + \sum_{i=1}^{n} x_i w_i$$



Sigmoid Function $\sigma(z) = \frac{1}{1+e^{-z}}$
1.0
0.5
0.0
$\sigma(z)$
-10
-5
0
5
10
$z = \sum w_i x_i + bias$

First, we must talk about neurons, the basic unit of a neural network. A neuron takes inputs, does some math with them and produces one output. Here's what a 2-input neuron looks like:



In neural network training, the weights are typically initialized randomly and then adjusted during the training process through backpropagation. Backpropagation is an iterative algorithm that updates the weights in the network based on the error between the predicted output and the actual output. By minimizing this error, the network can learn to make more accurate predictions.

- Assume we have a 2-input neuron that uses the sigmoid activation function and has the following parameters:
    - w = [0, 1]
    - b = 0.5
- Now, let's give the neuron an input of x = [2, 3]. We'll use the dot product to write things more concisely:
    - (w.x) + b = ((w1*x1) + (w2*x2)) + b
        - = (0*2 + 1*3 + 0.5)
        - = 3.5

So, y = f ((x1*w1)+(x2*w2)+b) #sigmoid

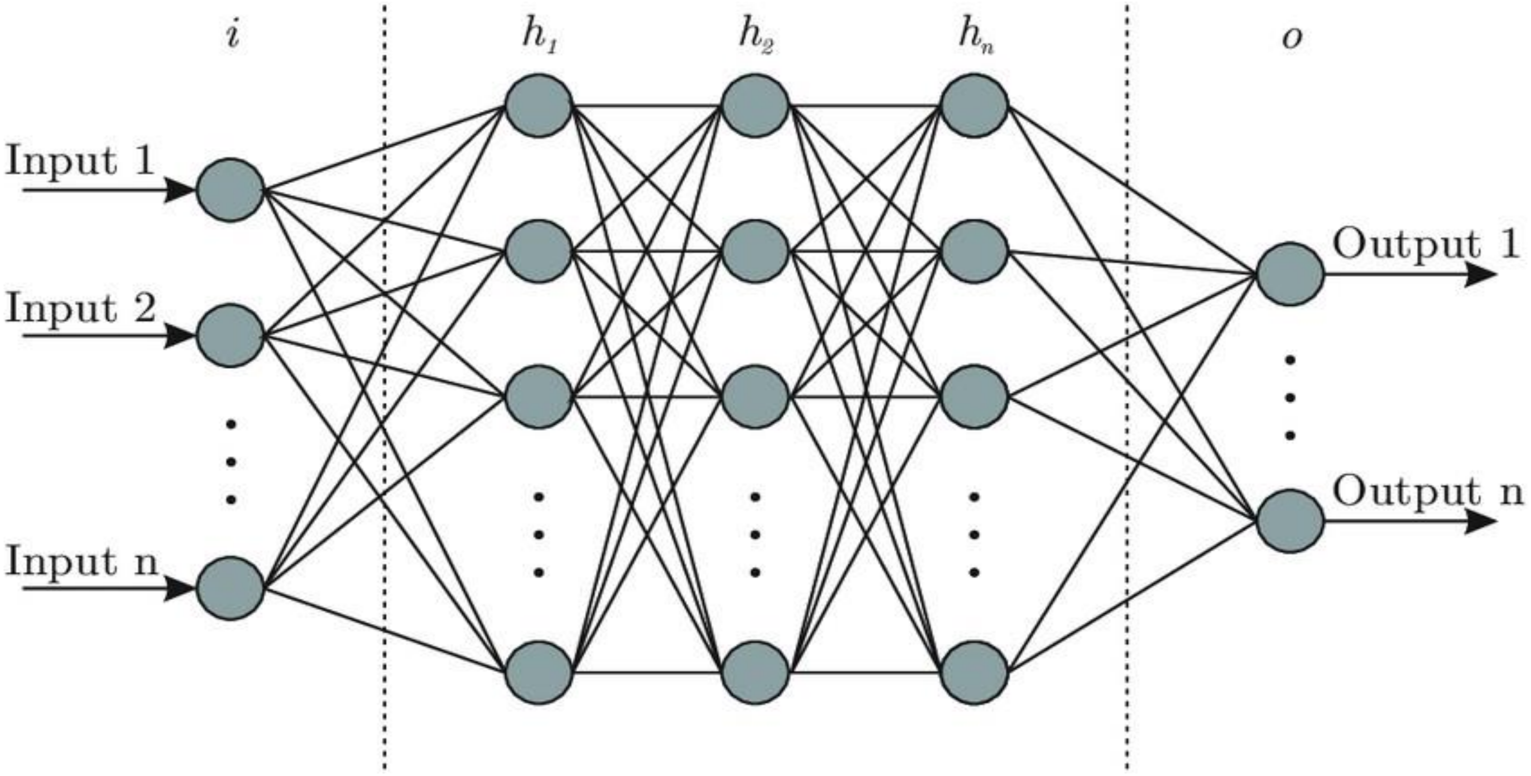
=f(3.5)

~0.97

~1

The neuron outputs given the inputs x=[2,3]. This process of passing inputs forward to get output is known as feedforward. That's it!

Neural Networks



1. Forward Propagation:

- Input data is passed through the network's layers from the input layer to the output layer.
- At each layer, the input is linearly combined with the weights and biases, and then passed through an activation function to produce the output for that layer.
- The output of one layer becomes the input of the next layer.
- The final output of the network is compared to the true target values to compute a loss (a measure of the error in the predictions).

2. Loss Calculation:

- The loss function quantifies how far off the predictions are from the actual targets. It's a measure of the error in the model's predictions.

3. Backpropagation:

- Gradients of the loss function concerning the parameters (weights and biases) of the network are computed. This is done by applying the chain rule of calculus.
- These gradients are used to update the parameters during the optimization process (e.g., using gradient descent).

4. Parameter Update:

- The weights and biases are adjusted using an optimization algorithm (like gradient descent) based on the computed gradients. The goal is to minimize the loss function.

5. Repeat:

- Steps 1 to 4 are repeated for several iterations or epochs to improve the model's performance.

The reason forward propagation is necessary before backpropagation is because the forward pass generates the predicted outputs, which are compared to the true targets to compute a loss. The gradients used in backpropagation are derived from this loss. In other words, backpropagation relies on the results of forward propagation to compute the gradients needed for parameter updates.

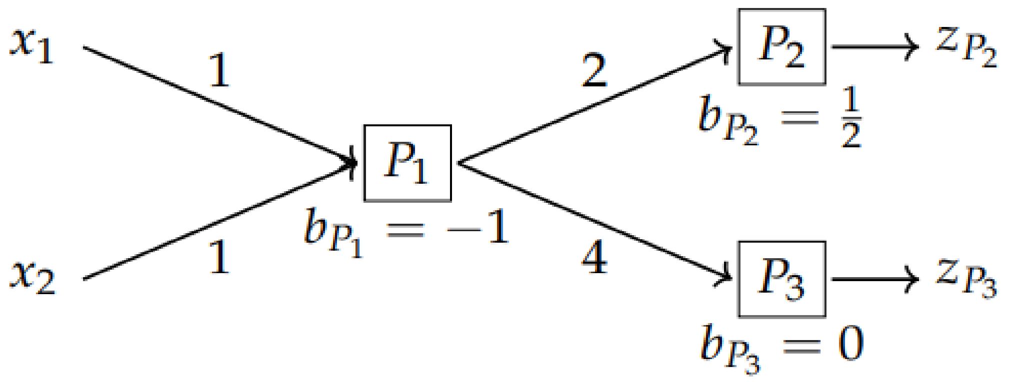
Specifying the epochs=10 means that the neural network will train on the entire training dataset for 10 iterations. During each epoch, the neural network will update its weights multiple times using backpropagation and stochastic gradient descent (or other optimization algorithms) until it has seen all the training examples. The number of weight updates during an epoch depends on the **batch_size**, which is another hyperparameter that determines how many samples are used to update the weights in each iteration.

For example, if we have a training dataset of 1000 samples and set the batch size to 100, the neural network will update its weights 10 times during an epoch (since 1000/100 = 10). During each weight update, the neural network will calculate the gradient of the loss function with respect to the weights and use this gradient to adjust the weights in the direction that reduces the loss.

After 10 epochs, the neural network will have updated its weights 10 times on the entire training dataset and hopefully learned to make accurate predictions on new data.

What is the batch size in neural network training? The batch size is a hyperparameter that specifies the number of training examples used in one iteration of the optimization algorithm. The training examples are divided into small groups or batches, and the optimization algorithm updates the

Consider the following network with 3 neurons P_1, P_2, P_3



with initial weights (as denoted in the graph)

$$w_{P_1 x_1} = 1,\ w_{P_1 x_2} = 1,\ w_{P_2 P_1} = 2,\ w_{P_3 P_1} = 4$$

initial biases (as denoted in the graph) $b_{P_1} = -1$, $b_{P_2} = \frac{1}{2}$, $b_{P_3} = 0$, and activation functions

$$\psi_{P_1}(t) = \frac{1}{1+3^{-t}}, \ \ \psi_{P_2}(t) = t^2, \ \ \psi_{P_3}(t) = t^2.$$

You may use without proof that the derivative of ψ_{P_1} is given by

$$\psi'_{P_1}(t) \approx \psi_{P_1}(t)(1 - \psi_{P_1}(t)).$$

Let

$$\theta = (w_{P_1x_1}, w_{P_1x_2}, w_{P_2P_1}, w_{P_3P_1}, b_{P_1}, b_{P_2}, b_{P_3})^T$$

and let $f_\theta(x) = (z_{P_2}, z_{P_3})^T \in \mathbb{R}^2$ denote the output of the network using parameters θ and input $x = (x_1, x_2)^T \in \mathbb{R}^2$. Consider the loss function $C(\theta; x, y) = \frac{1}{2}||f_\theta(x) - y||^2$ for a given training pair (x, y).

a) Perform one training iteration using the input data $x^1 = \begin{pmatrix} -1 \\ 3 \end{pmatrix}$, $y^1 = \begin{pmatrix} 3 \\ 8 \end{pmatrix}$, and step size $\eta = 0.1$. State the updated weights and biases.

b) Assume that you are given a second point (x^2, y^2) with

$$\nabla C(\theta; x^2, y^2) = (6, 2, 1, 5.5, 10, -4, 2)^T.$$

What are the updated weights and biases if you use both points and the mean squared loss function in the first training iteration instead of only using x^1 as in a) (again with stepsize $\eta = 0.1$)?

a) We start computing the layers' outputs using a forward pass

$$
\begin{aligned}
a_{P_1} &= w_{P_1 x_1} \cdot x_1 + w_{P_1 x_2} \cdot x_2 + b_{P_1} = 1 \cdot (-1) + 1 \cdot 3 - 1 = 1 \\
z_{P_1} &= \psi_{P_1}(a_{P_1}) = \frac{1}{1 + 3^{-1}} = \frac{1}{1 + \frac{1}{3}} = \frac{3}{4} \\
a_{P_2} &= w_{P_2 P_1} \cdot z_{P_1} + b_{P_2} = 2 \cdot \frac{3}{4} + \frac{1}{2} = 2 \\
z_{P_2} &= \psi_{P_2}(a_{P_2}) = 2^2 = 4 \\
a_{P_3} &= w_{P_3 P_1} \cdot z_{P_1} + b_{P_3} = 4 \cdot \frac{3}{4} + 0 = 3 \\
z_{P_3} &= \psi_{P_3}(a_{P_3}) = 3^2 = 9.
\end{aligned}
$$

We need the derivatives of the activation functions:

$$
\begin{aligned}
\psi'_{P_1}(t) &\approx \psi_{P_1}(t)(1 - \psi_{P_1}(t)) \\
\psi'_{P_2}(t) &= \psi'_{P_3}(t) = 2t.
\end{aligned}
$$

Now, we compute the partial derivatives of the loss function C with respect to all elements in θ using backpropagation

$$\frac{\partial C}{\partial b_{P_2}} = (z_{P_2} - y_1) \cdot \psi'_{P_2}(a_{P_2}) = (4-3) \cdot 2 \cdot 2 = 4$$
$$\frac{\partial C}{\partial b_{P_3}} = (z_{P_3} - y_2) \cdot \psi'_{P_3}(a_{P_3}) = (9-8) \cdot 2 \cdot 3 = 6$$
$$\begin{aligned}\frac{\partial C}{\partial b_{P_1}} &= \left(\frac{\partial C}{\partial b_{P_2}} \cdot w_{P_2P_1} + \frac{\partial C}{\partial b_{P_3}} \cdot w_{P_3P_1}\right) \cdot \psi'_{P_1}(a_{P_1}) \\ &= (4 \cdot 2 + 6 \cdot 4) \cdot z_{P_1} \cdot (1 - z_{P_1}) \\ &= (8+24) \cdot \frac{3}{4} \cdot \left(1 - \frac{3}{4}\right) = 32 \cdot \frac{3}{4} \cdot \frac{1}{4} = 6\end{aligned}$$
$$\frac{\partial C}{\partial w_{P_2P_1}} = \frac{\partial C}{\partial b_{P_2}} \cdot z_{P_1} = 4 \cdot \frac{3}{4} = 3$$
$$\frac{\partial C}{\partial w_{P_3P_1}} = \frac{\partial C}{\partial b_{P_3}} \cdot z_{P_1} = 6 \cdot \frac{3}{4} = 4,5$$
$$\frac{\partial C}{\partial w_{P_1x_1}} = \frac{\partial C}{\partial b_{P_1}} \cdot x_1 = 6 \cdot (-1) = -6$$
$$\frac{\partial C}{\partial w_{P_1x_2}} = \frac{\partial C}{\partial b_{P_1}} \cdot x_2 = 6 \cdot 3 = 18.$$

Therefore, the gradient reads:

$$\nabla C(\theta) = \begin{pmatrix} -6 \\ 18 \\ 3 \\ 4,5 \\ 6 \\ 4 \\ 6 \end{pmatrix}.$$

We update the parameters with a gradient step

$$\theta^{new} = \theta - \eta \nabla C(\theta) = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 4 \\ -1 \\ 0.5 \\ 0 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} -6 \\ 18 \\ 3 \\ 4,5 \\ 6 \\ 4 \\ 6 \end{pmatrix} = \begin{pmatrix} 1,6 \\ -0,8 \\ 1,7 \\ 3,55 \\ -1,6 \\ 0,1 \\ -0,6 \end{pmatrix}.$$

The updated parameters are therefore given by

$$w_{P_1 x_1}^{new} = 1,6 \quad w_{P_1 x_2}^{new} = -0,8 \quad w_{P_2 P_1}^{new} = 1,7 \quad w_{P_3 P_1}^{new} = 3,55$$

$$b_{P_1}^{new} = -1,6 \quad b_{P_2}^{new} = 0,1 \quad b_{P_3}^{new} = -0,6.$$

b) Using a second data point we have to compute the averaged gradient $\overline{\nabla}C(\theta)$ and use it for the update step. The averaged gradient is given by

$$\overline{\nabla}C(\theta) = \frac{1}{2}(\nabla(\theta, x^1, y^1) + \nabla(\theta, x^2, y^2)) = \frac{1}{2}\begin{pmatrix} 0 \\ 20 \\ 4 \\ 10 \\ 16 \\ 0 \\ 8 \end{pmatrix} = \begin{pmatrix} 0 \\ 10 \\ 2 \\ 5 \\ 8 \\ 0 \\ 4 \end{pmatrix}.$$

The update is given by

$$\theta^{new} = \theta - \eta\overline{\nabla}C(\theta) = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 4 \\ -1 \\ 0.5 \\ 0 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 0 \\ 10 \\ 2 \\ 5 \\ 8 \\ 0 \\ 4 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1,8 \\ 3,5 \\ -1,8 \\ 0,5 \\ -0,4 \end{pmatrix}.$$

The updated parameters are therefore given by

$$w^{new}_{P_1 x_1} = 1 \quad w^{new}_{P_1 x_2} = 0 \quad w^{new}_{P_2 P_1} = 1,8 \quad w^{new}_{P_3 P_1} = 3,5$$

$$b^{new}_{P_1} = -1,8 \quad b^{new}_{P_2} = 0,5 \quad b^{new}_{P_3} = -0,4.$$

Resources:

- Deep Learning Full course
- FAU Deep Learning
- DL & AI Specialization - by aiQuest Intelligence (Bangla)