Neural Networks & LLMs

Unit 4: Session 03

By Md. Meftaul Haque Mishu

Agenda

- Understand **Neural Networks**
- What is **Deep Learning**
- Understand **Large Language Models (LLM)**
- Basics of Prompt Engineering

Neural Networks

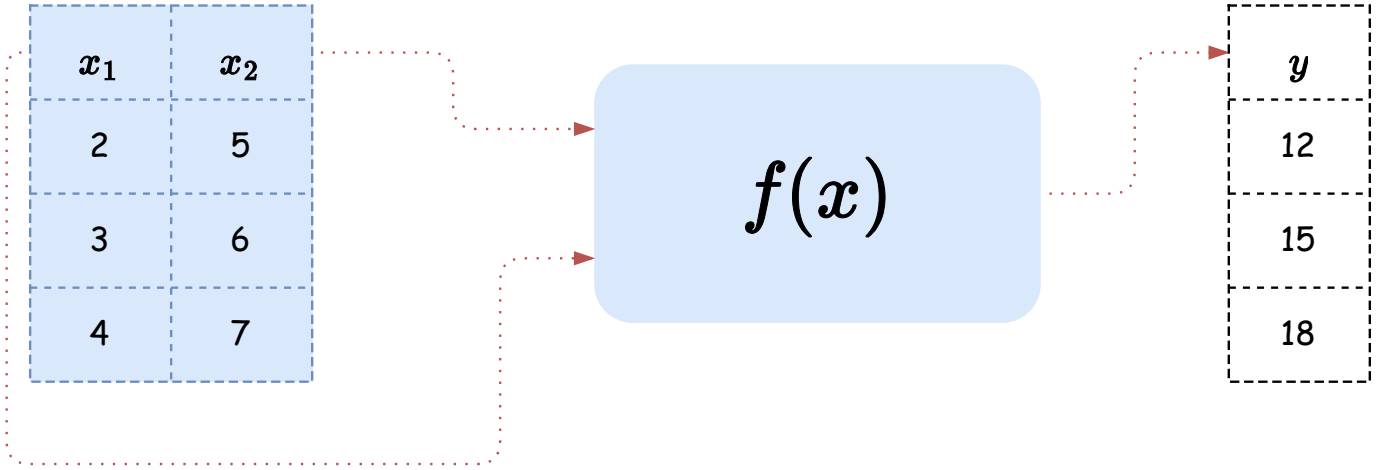
Topics to cover:

- Intuition
- Perceptron
- Neuron
- Forward Propagation
- Back Propagation
- Deep Neural Network
- Implement a NN Using Tensorflow

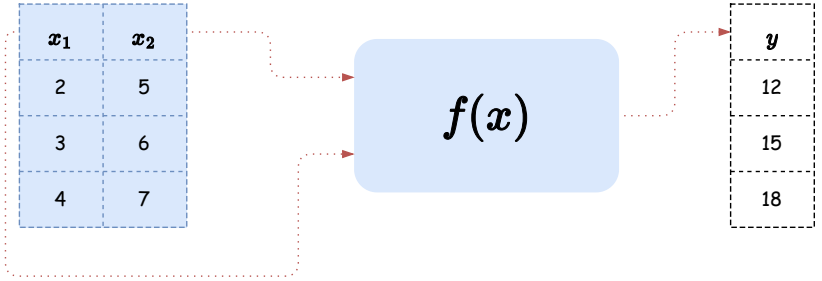
Intuition

Guess the Equation $f(x)$

- $f(x) = x_1 + 2x_2$

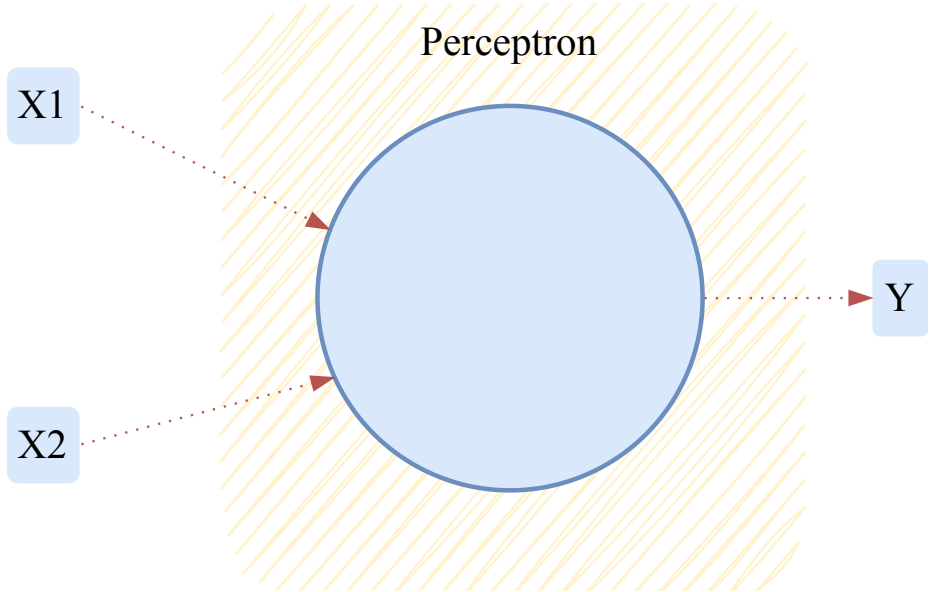


- **Observations:**
  - Inputs and Output are known
  - We need to find the equation and **parameters**
  - We can multiply the inputs with some **weights** and add a **bias** to get the output
- **Question:**
  - How to find the **weights** and **bias**?



Perceptron

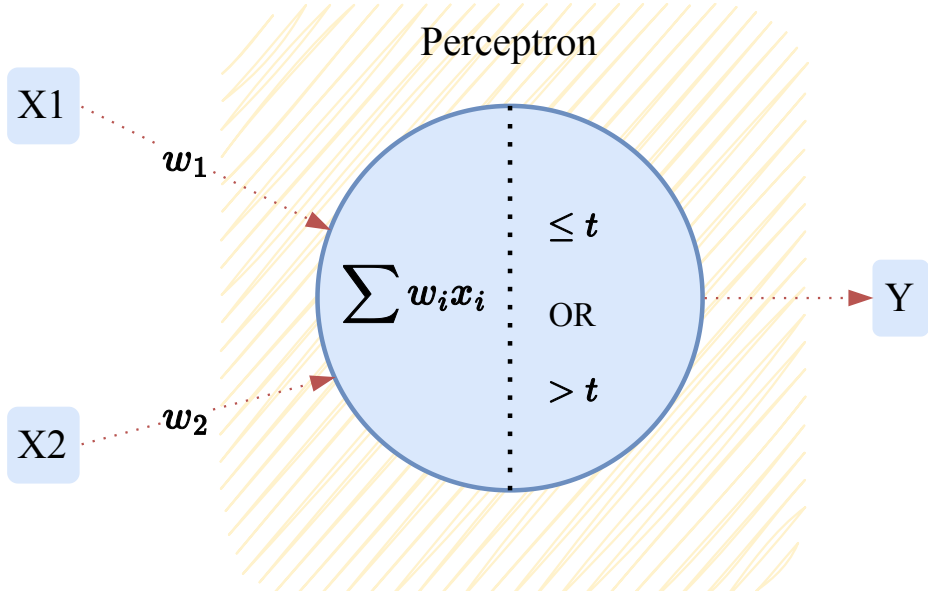
- Takes several **binary inputs**, $x_1, x_2, \ldots, x_n$
- Produces a **single binary output**



How the output of a Perceptron is calculated?

- Each input is associated with a **weight**, $w_1, w_2, \ldots, w_n$
- The **weighted sum** of the inputs is calculated, $w_1x_1 + w_2x_2 + \ldots + w_nx_n$
- If the **weighted sum** is above a **threshold** value, the perceptron **fires** and outputs 1, otherwise it outputs 0

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$



- The **threshold** value is a **bias** value, b
- Weights express the **importance** of the respective inputs to the output
- Weights and bias are **parameters** of the neuron

Let's build a **perceptron** that can decide whether to go to a concert or not

Input:

- Weather is Good or Bad (Lets call it x_1)
- Friend is going or not (x_2)
- Public transport is available or not (x_3)

Output:

- Go to the concert or not (y)

I am going to the concert if the weighted sum is greater than 60

Let's assume weight for **Weather is** w_1, weight for **Friend is** w_2 and weight for **Public Transport is** w_3

Concert decision table

Weather	Friend	Public Transport
Good	Yes	Yes
Bad	No	Yes
Bad	No	No

- $w_1 = 20, w_2 = 50$ and $w_3 = 30$
- For **Good Weather**, **Friend is going** and **Public Transport is available**, the weighted sum is $20 + 50 + 30 = 100$
- For **Bad Weather**, **Friend is not going** and **Public Transport is available**, the weighted sum is $0 + 0 + 30 = 30$
  - Add Bias $b = 30$ to get 60
  - Total formula is $20x_1 + 50x_2 + 30x_3 + 30$
- For **Bad Weather**, **Friend is not going** and **Public Transport is not available**, the weighted sum is $0 + 0 + 0 = 0$
  - Add Bias $b = 60$ to get 00
  - Total formula is $20x_1 + 50x_2 + 30x_3 + 60$

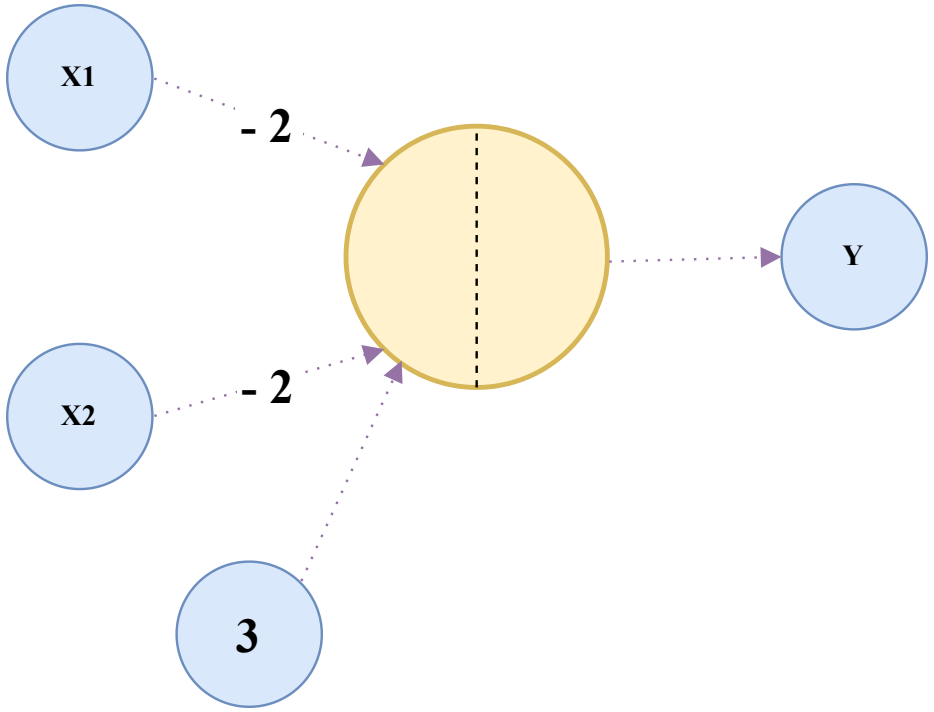
Key Takeaways from Perceptron

- Weights determine the **importance** of the input
- Bias shifts the **decision boundary**

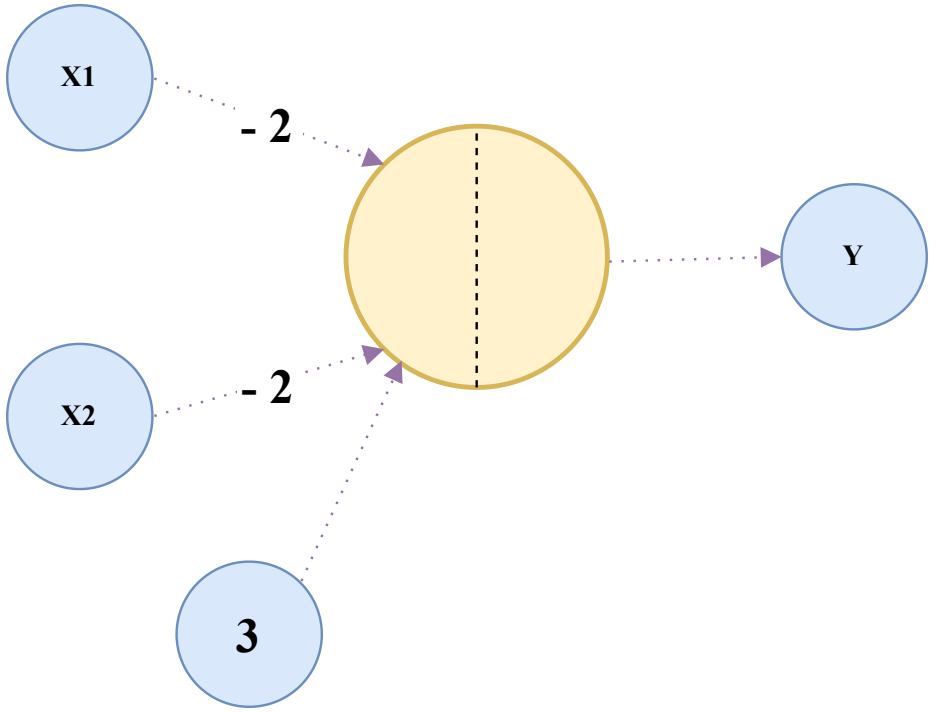
Example 02

- Here $w_1 = -2, w_2 = -2$ and $b = 3$
- $result = -2x_1 - 2x_2$

$$\mathrm{y} = \begin{cases} 0 & result + 3 \leq 0 \\ 1 & result + 3 > 0 \end{cases}$$

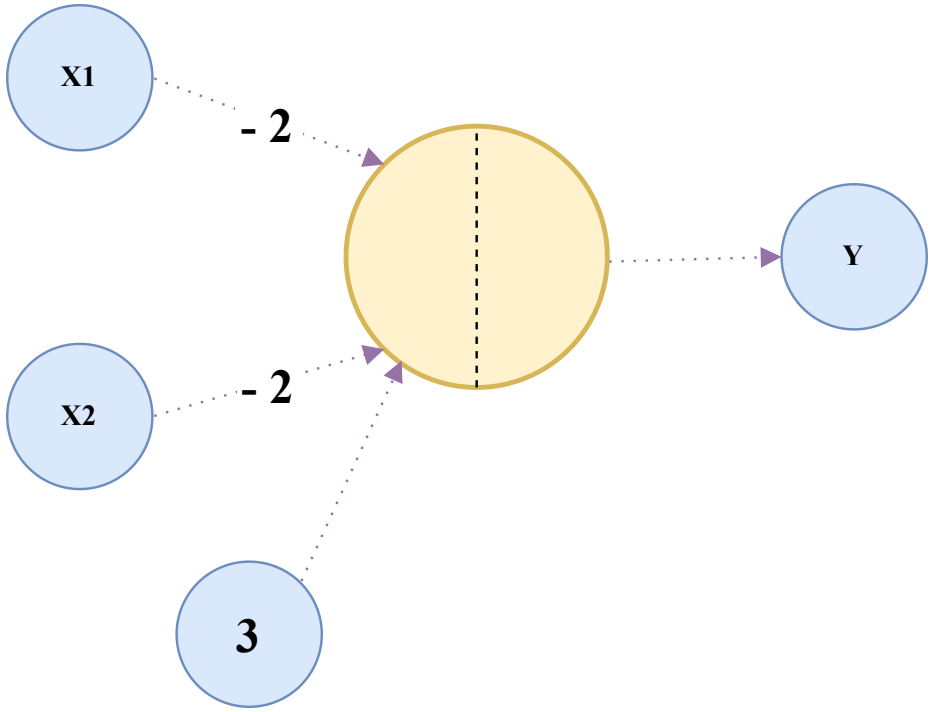


- Compute the output for input
  - $x_1 = 0$ and $x_2 = 0$
  - $x_1 = 0$ and $x_2 = 1$
  - $x_1 = 1$ and $x_2 = 0$
  - $x_1 = 1$ and $x_2 = 1$



x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

- It's a **NAND** gate, right !!



Historical Note:

- Perceptrons were developed in the 1950s and 1960s by the scientist Frank Rosenblatt
- Inspired by earlier work by Warren McCulloch and Walter Pitts.

Wiki: https://en.wikipedia.org/wiki/Perceptron

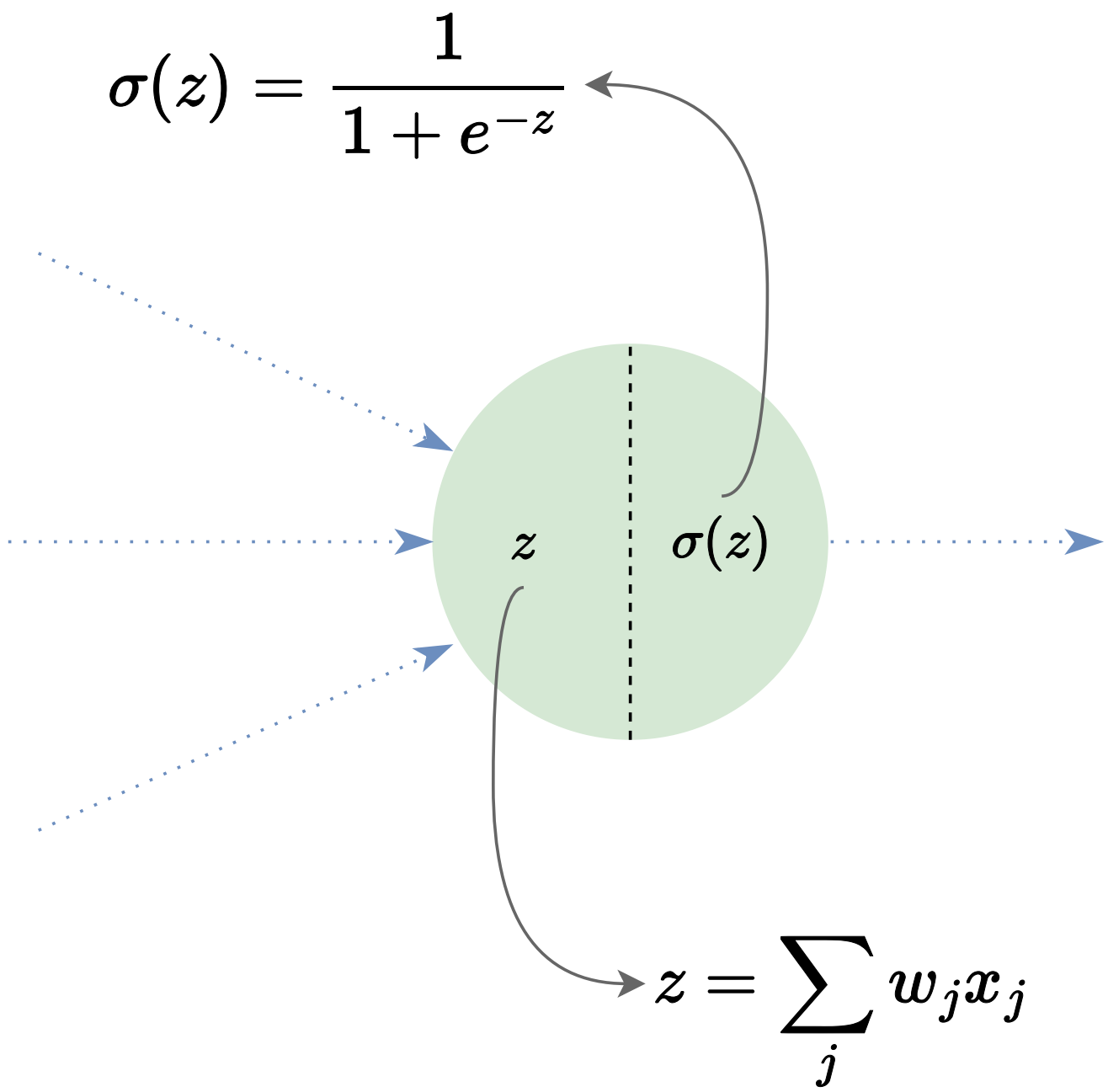
Limitations of Perceptron:

- Takes only **binary inputs** and produces a **single binary output**
- Perceptron can only solve **linearly separable** problems
- Perceptron can't solve **XOR** problem

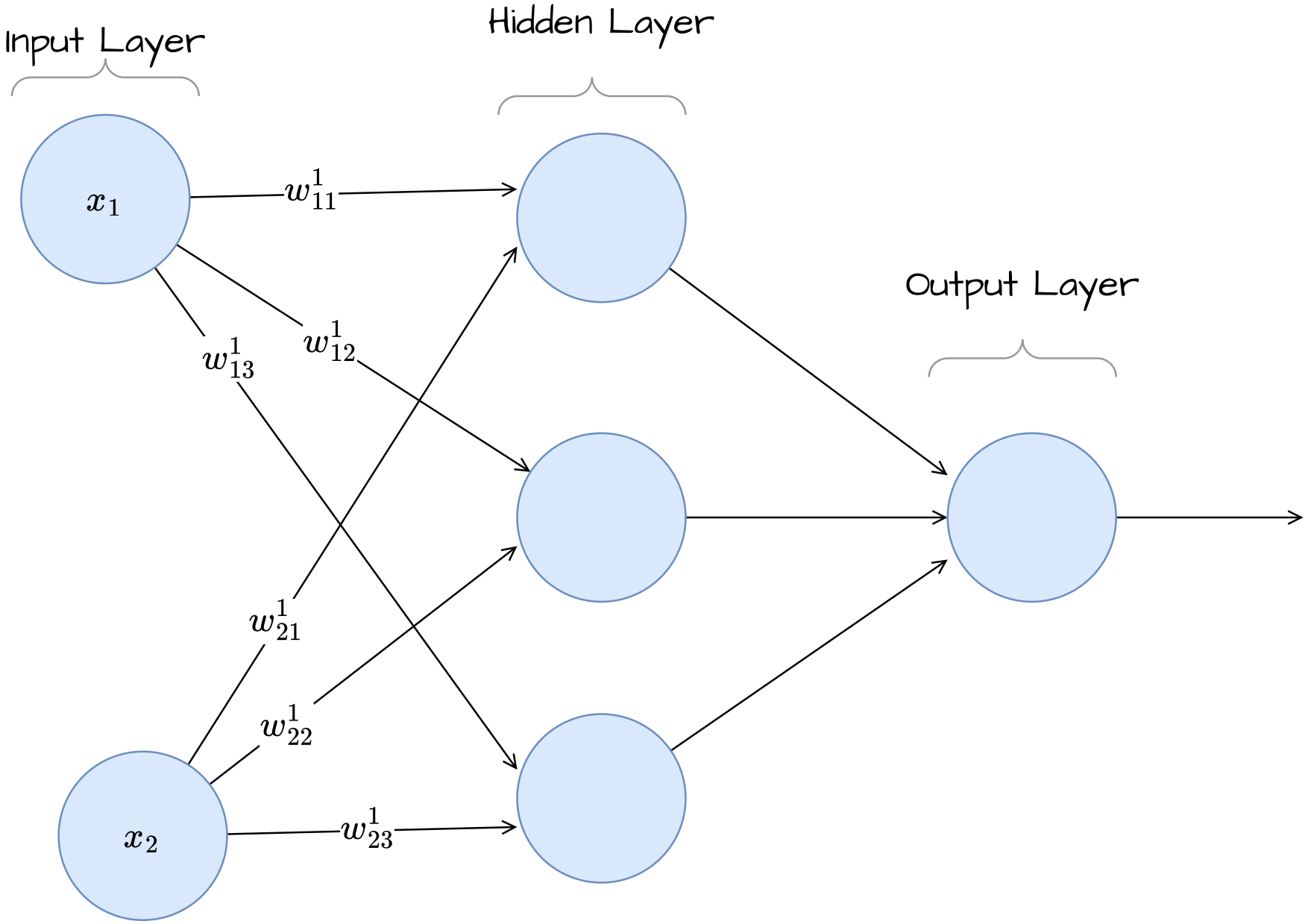
Sigmoid Neuron

- A sigmoid neuron takes several **inputs**, $x_1, x_2, \ldots, x_n$
- Input includes 0, 1 and any values in between 0 and 1
- Just like a perceptron, sigmoid neuron has **weights**, $w_1, w_2, \ldots, w_n$ for each input and a **bias** value, b
- Output is $\sigma(w \cdot x + b)$ where σ is called **sigmoid function**

Sigmoid Neuron



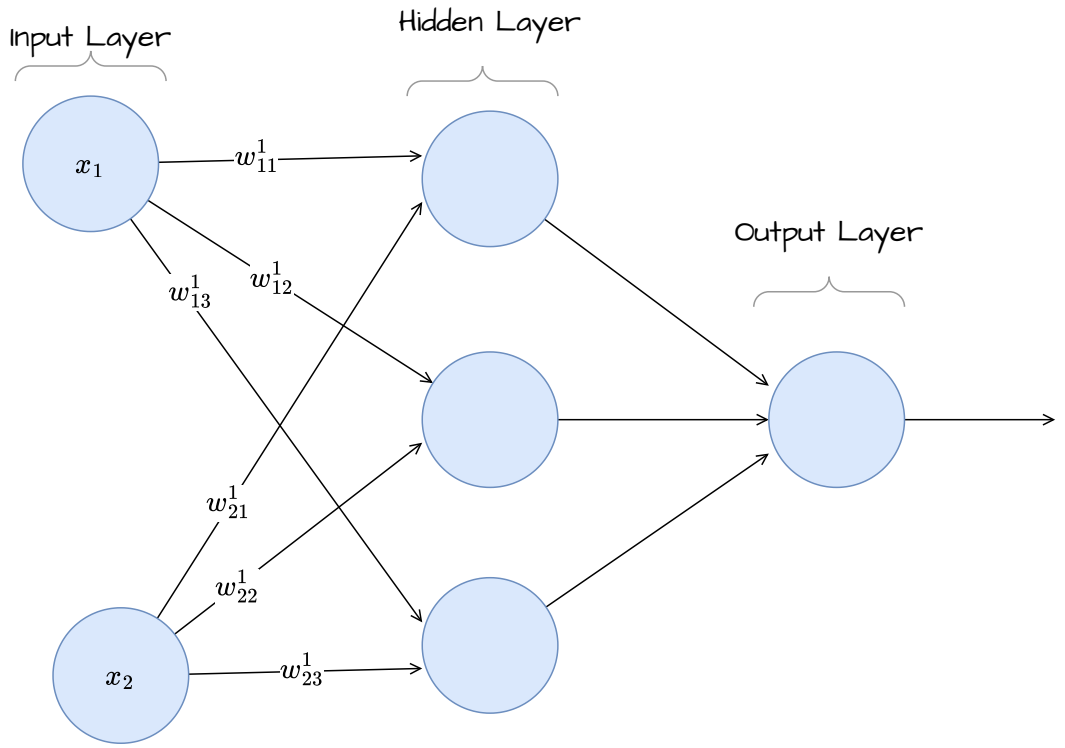
Architecture of Neural Network


Input Layer
Hidden Layer
Output Layer
x_1
x_2
w_{11}^1
w_{12}^1
w_{13}^1
w_{21}^1
w_{22}^1
w_{23}^1

x_1	x_2	$\hat{y}$
5	2	12
3	1	7
2	2	6

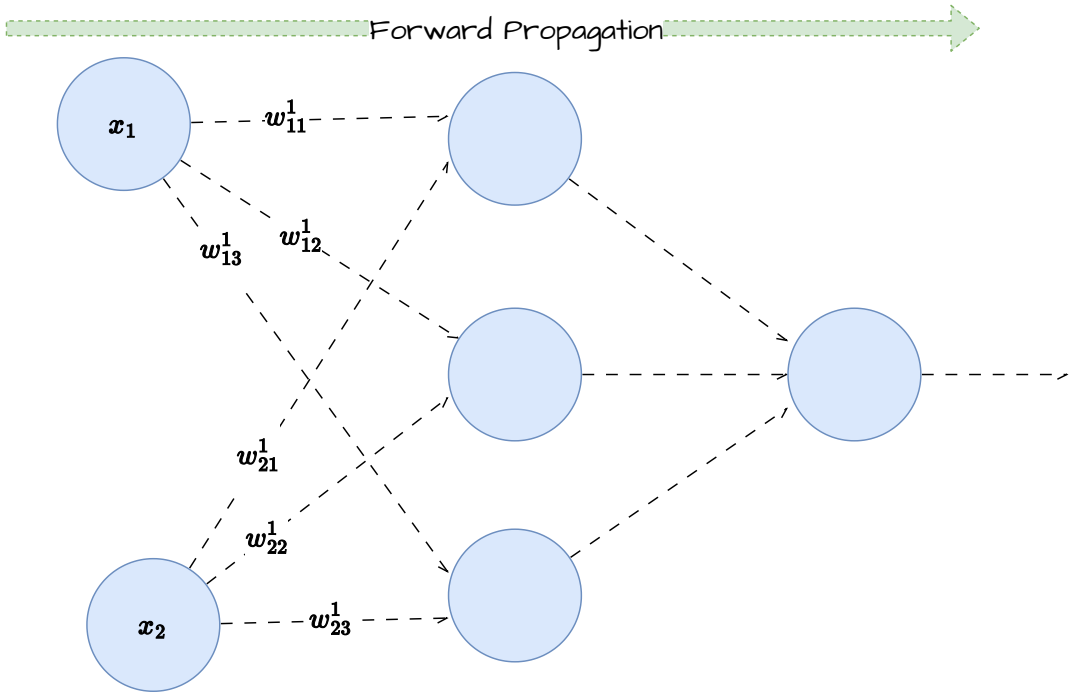
$$\begin{bmatrix} 5w_{11}+2w_{21} & 5w_{12}+2w_{22} & 5w_{13}+2w_{23} \\ 3w_{11}+1w_{21} & 3w_{12}+1w_{22} & 3w_{13}+1w_{23} \\ 2w_{11}+2w_{21} & 2w_{12}+2w_{22} & 2w_{13}+2w_{23} \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 2 \\ 3 & 1 \\ 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} = X \cdot W$$



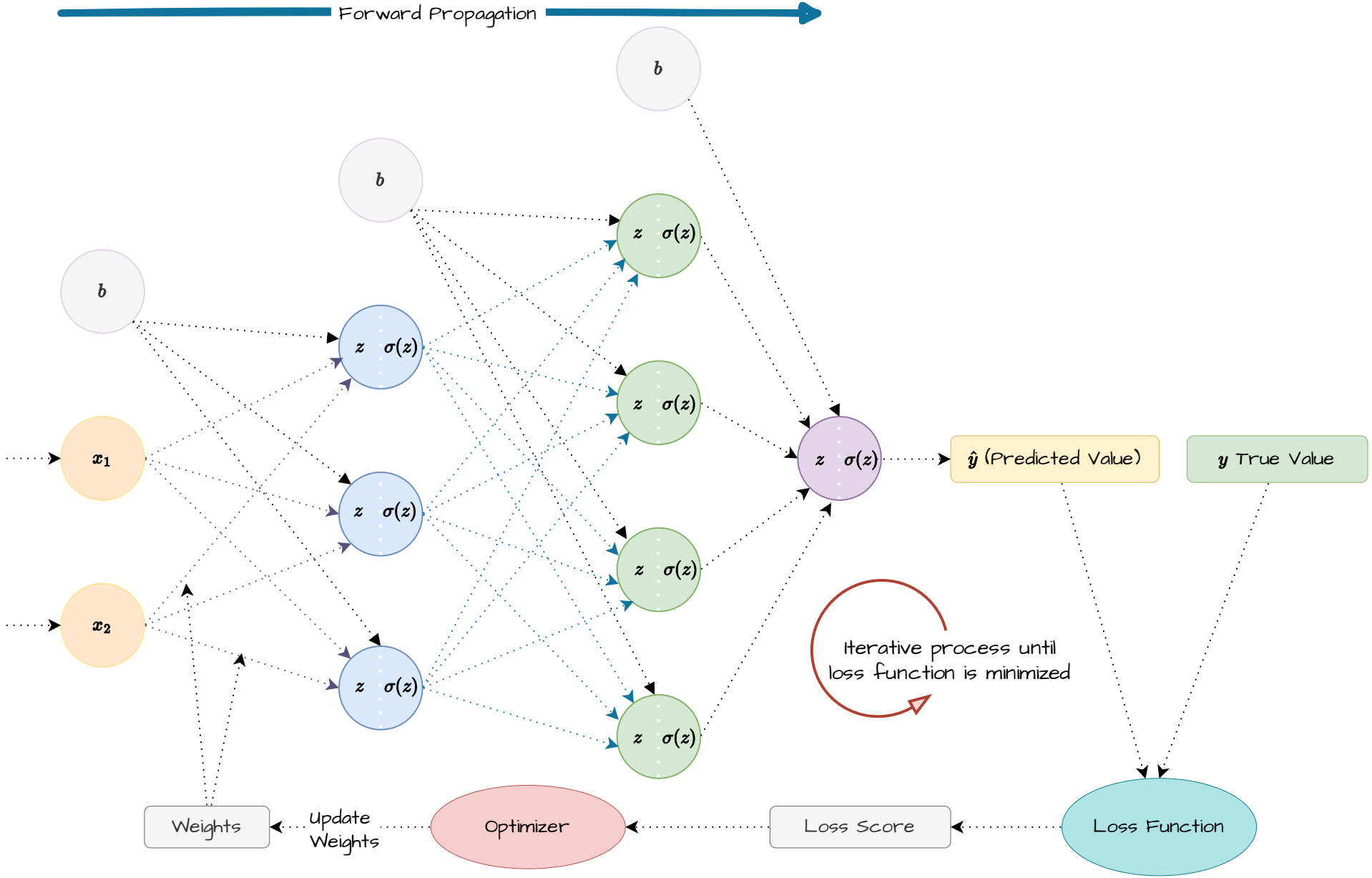
Forward Propagation

- Forward propagation is the process of moving from left to right through the neural network



Backward Propagation

- Backward propagation is the process of moving from right to left through the neural network
- **Step 01:** Calculate the **loss** of the network
- **Step 02:** Calculate the **gradient** of the loss with respect to the parameters of the network
- **Step 03:** Update the parameters using **gradient descent**


Forward Propagation
b
b
b
x_1
x_2
z $\sigma(z)$
$\hat{y}$ (Predicted Value)
y True Value
Iterative process until loss function is minimized
Weights
Update Weights
Optimizer
Loss Score
Loss Function
Backward Propagation

Neural Network

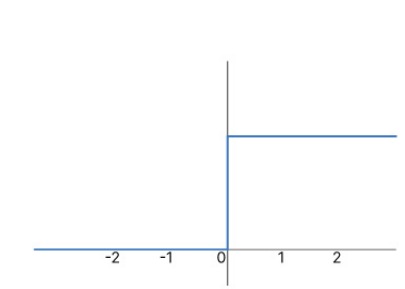
Core Concepts

Neuron

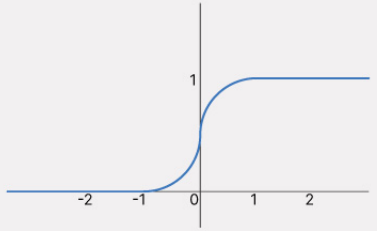
- Basic building block of a neural network.
- Receives one or more inputs, perform computation and produce single output.

Activation Function

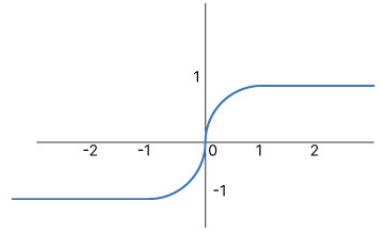
- It is the main component that enable neural networks to learn complex non-linear relationships.
- Types of Activation Functions:
  - Sigmoid
  - Tanh
  - ReLU
  - Softmax (Typically this is used in the output layer)



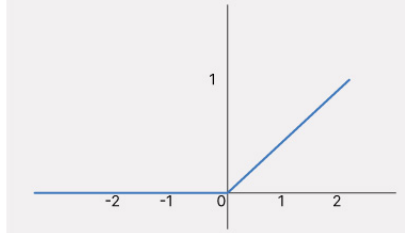
Step Function



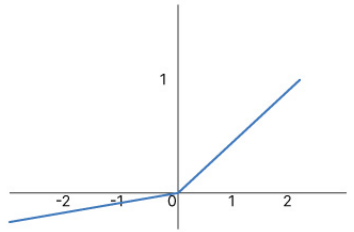
Sigmoid



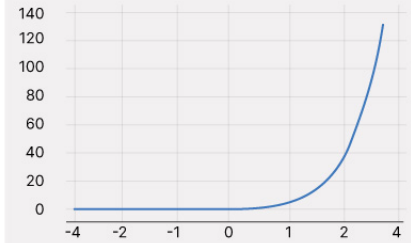
tanh



ReLU



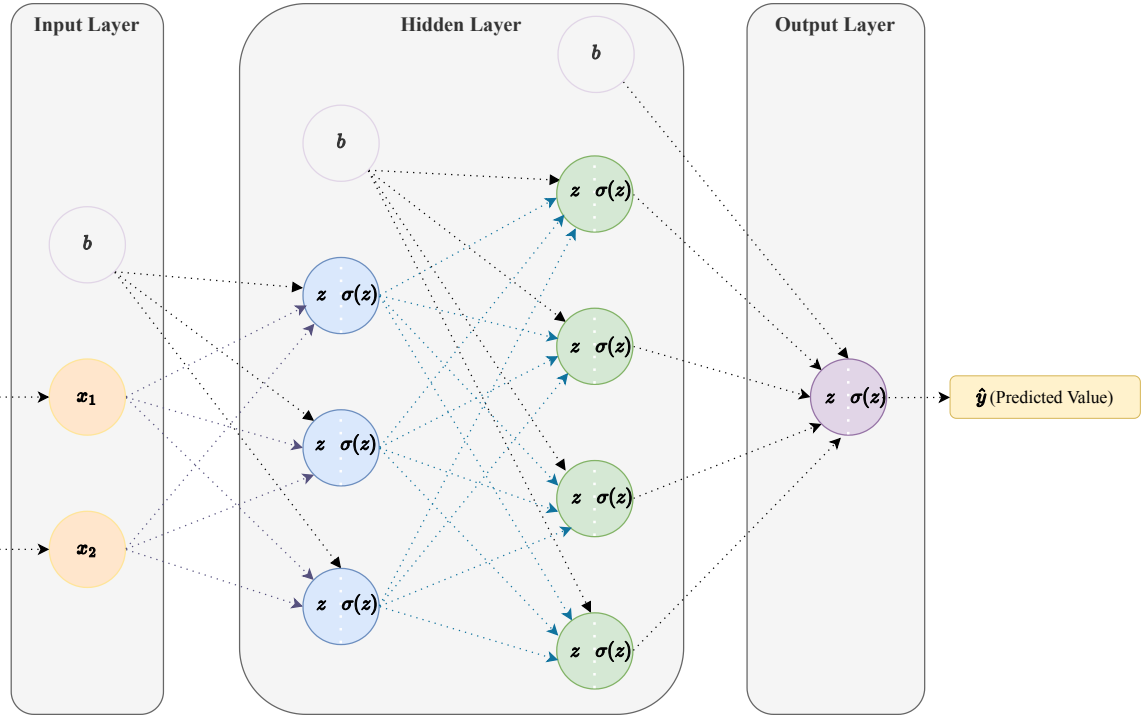
Leaky ReLU



Softmax Function

Layers

- Neurons are organized into layers
- Input layer is the first layer takes the input
- Hidden layers are the layers between input and output layer
- Output layer is the last layer that produces the output



Weights and Biases

- Connections between neurons have associated weights and biases.
- Weights represent the strength of the connection between units.
- Biases allow neurons to have an influence even if the input is zero.
- At the start we initialize the weights and biases randomly.
- During the training process, the network learns the optimal values for weights and biases.

Loss Function

- Quantifies how well the network is performing on the dataset.
- For regression problems, we can use **MSE**, **MAE**, etc.
- For classification problems, we can use **Cross Entropy**, **Binary Cross Entropy**, etc.

Optimization Algorithm

- Stochastic Gradient Descent (SGD)
- RMSProp
- Adam

Epoch

- One epoch is when an entire dataset is passed forward and backward through the neural network only once.

Implement Neural Network Using Tensorflow

Step 01: Import Libraries

```
import tensorflow as tf
```

Step 02: Define the Neural Network

```
# Set random seed for reproducibility
tf.random.set_seed(42)

# Define the model
model = tf.keras.Sequential([
    # Hidden layer with 4 neurons and ReLU activation
    tf.keras.layers.Dense(units=4, input_shape=(2,), activation='relu'),
    # Output layer with 1 neuron and sigmoid activation for binary classification
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
```

Step 03: Compile Model

```
# Compile the model
model.compile(optimizer='adam',    # Adam optimizer is commonly used
              loss='binary_crossentropy',    # Binary crossentropy for binary classification
              metrics=['accuracy'])    # Monitor accuracy during training
```

Step 04: Train the Model

```
# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

Step 05: Evaluate the Model

```
# Evaluate the model
model.evaluate(X_test, y_test)
```

Step 06: Make Predictions

```
# Make predictions
y_pred = model.predict(X_test)
```

Types of Neural Network

Convolutional Neural Network

- Image / Video analysis
- Object detection

RNN (Recurrent Neural Network)

- Time series analysis
- Machine translation
- Speech recognition
- Text Generation
- **Note**: Long Short-Term Memory **LSTM** is a special type of RNN

GANs (Generative Adversarial Networks)

- Generate image
- Create art

LLM (Large Language Models)

LLM Topics to cover:

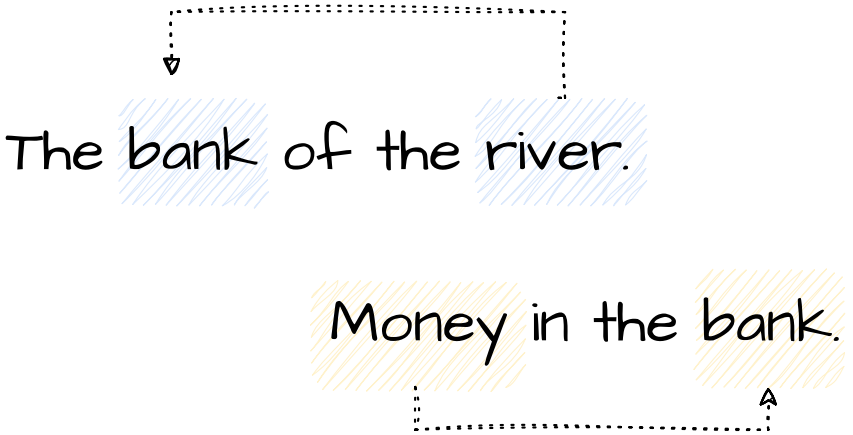
- What is LLM
- Popular LLM Models
- Architecture of LLM
- What is **Transformer**
- What is **Attention**

What is LLM

- Machine learning model that can predict and generate plausible text is called **Language Model**
- LLMs are **deep learning** models that are trained on **large datasets** of text
- Trained to generate human-like text

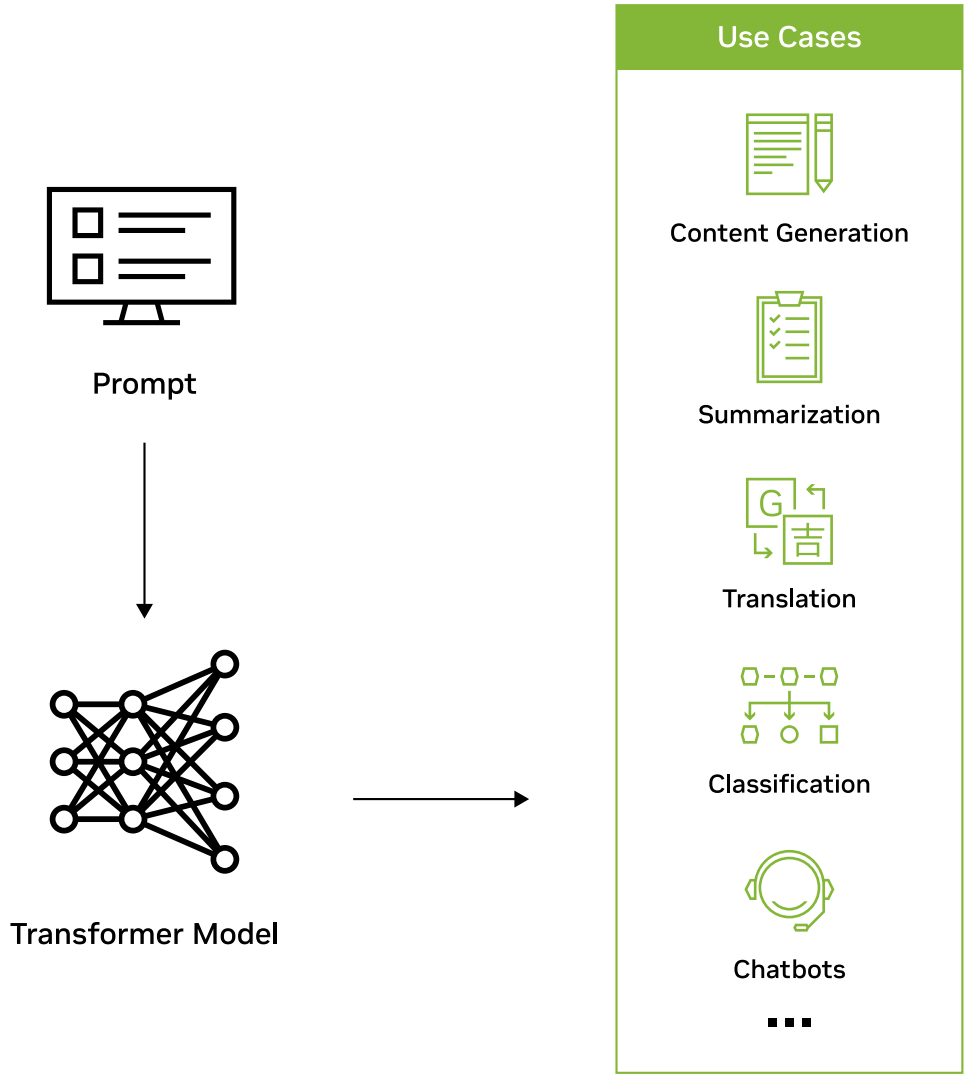
Attention Network

- The model learns to pay attention to relevant words in the input sequence.

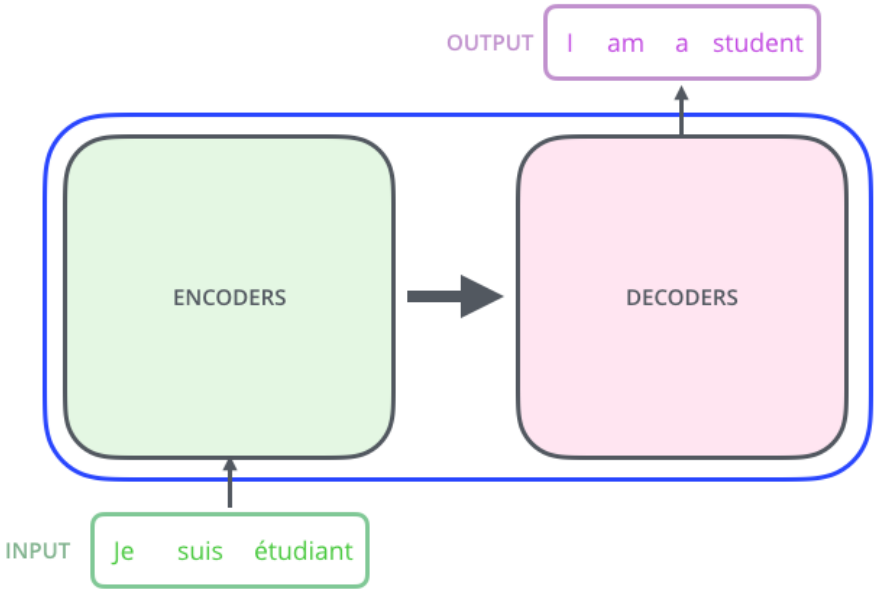


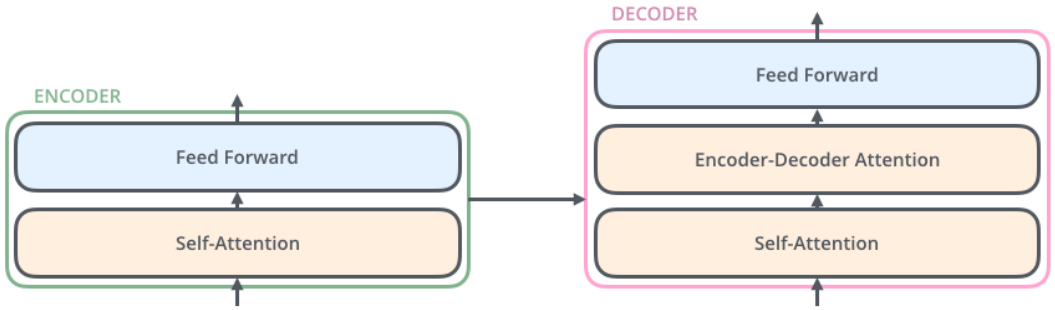
Transformer

- Special type of neural network architecture that is used in LLMs
- Transformer is based on the **Attention** mechanism
- Works well on **sequence-to-sequence** tasks such as machine translation and text summarization



Transformer





Popular LLM Models

- GPT-4 Developed by OpenAI
- PaLM 2 Developed by Facebook
- Llama 2 Developed by Google

https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

Prompt Engineering

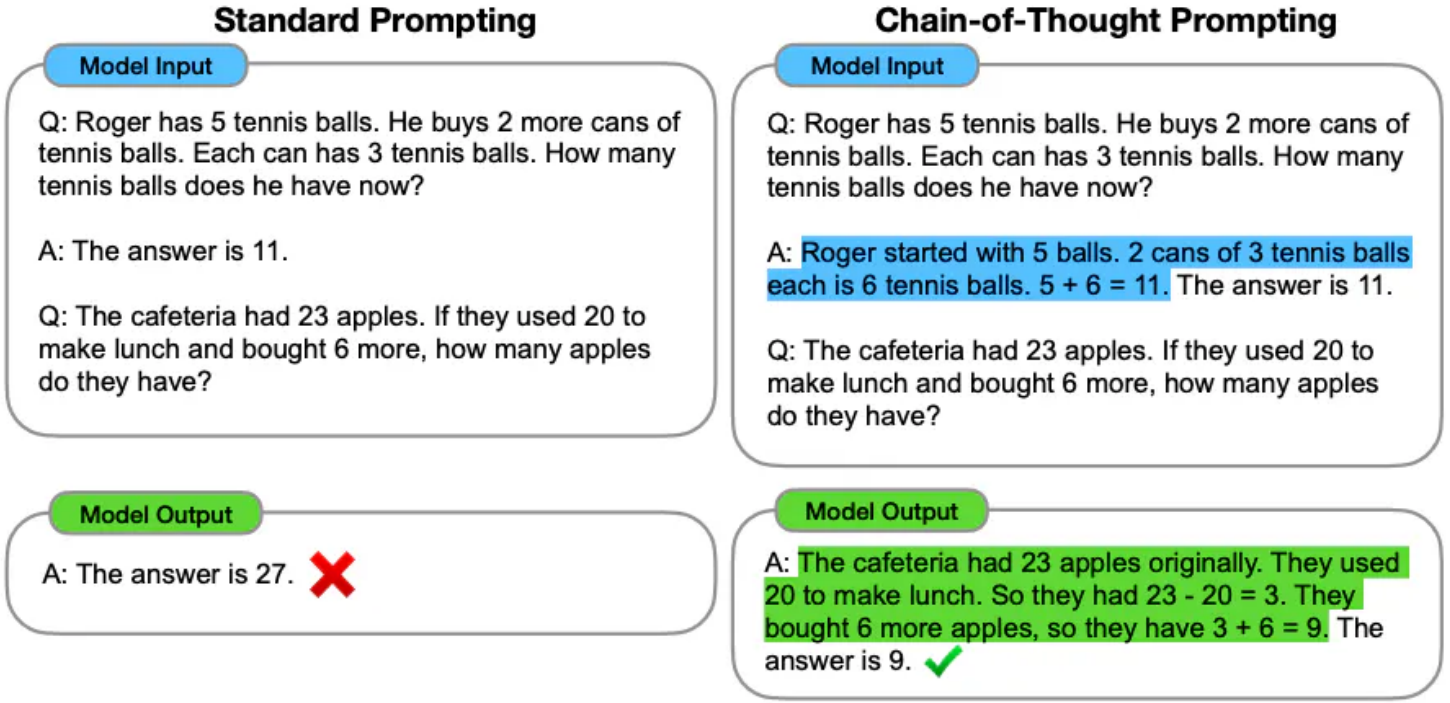
What is Prompt Engineering

- Art of asking the right question to the language model
- Prompt is a **text snippet** that is used to **generate** text from the language model

Types of Prompt

- Describe a topic
- Ask a question
- Opinion based
- Research
- Translation
- Summarization

- **Chain of Thought (COT)**
  - It enables complex reasoning capabilities through intermediate reasoning steps
- More: https://www.promptingguide.ai



Best practices of Prompt Engineering

- Clearly communicate what content or information is important
- Use **keywords** to guide the model
- Use specific and varied **prompts** to get the best results
- Use **multiple prompts** to get the best results

NN Resources:

http://neuralnetworksanddeeplearning.com (Book)

https://playground.tensorflow.org/

LLM Resources:

https://jalammar.github.io/illustrated-transformer/

https://www.promptingguide.ai

https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

Thank You

Happy Learning 🚀 !!

meftaulhaque@gmail.com