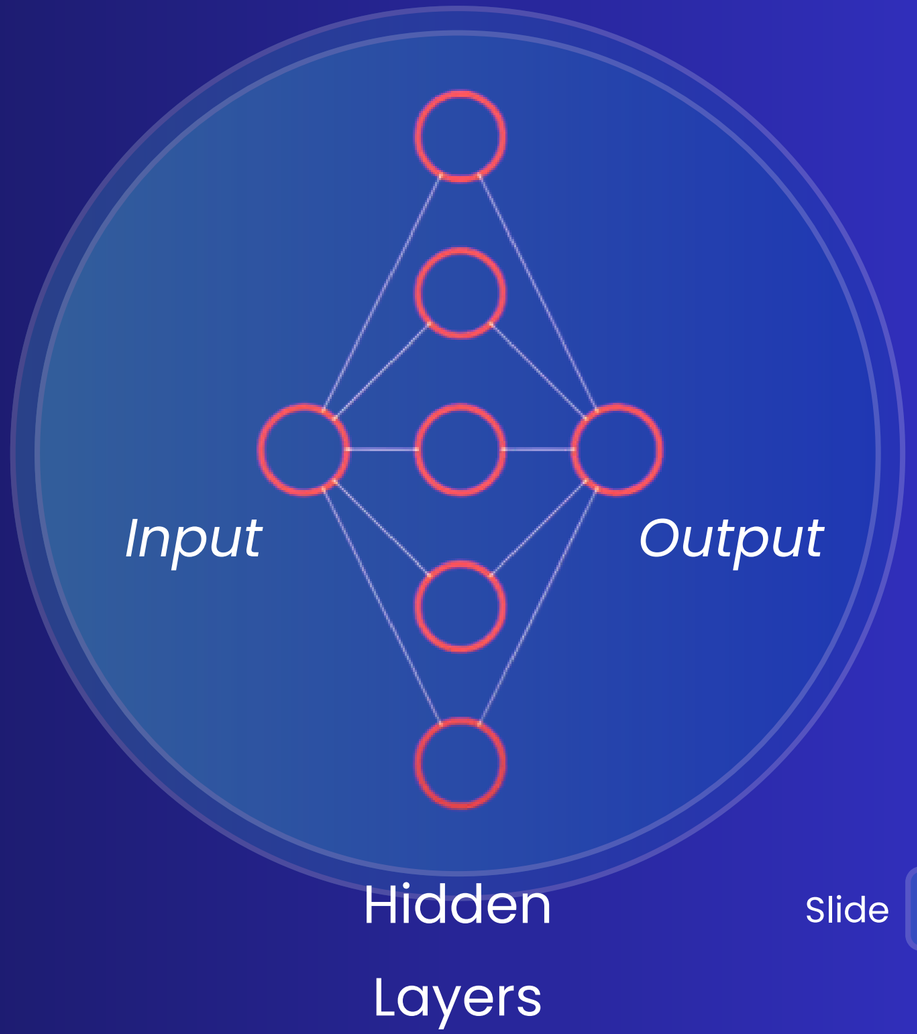
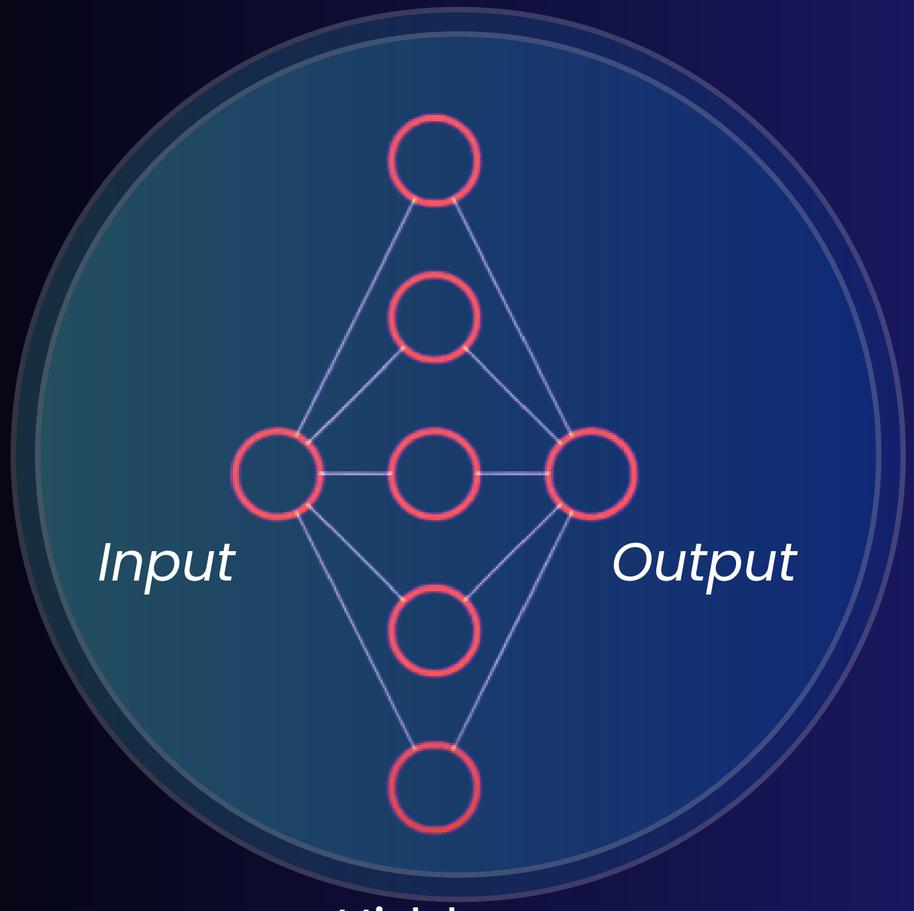
ACTIVATION FUNCTIONS

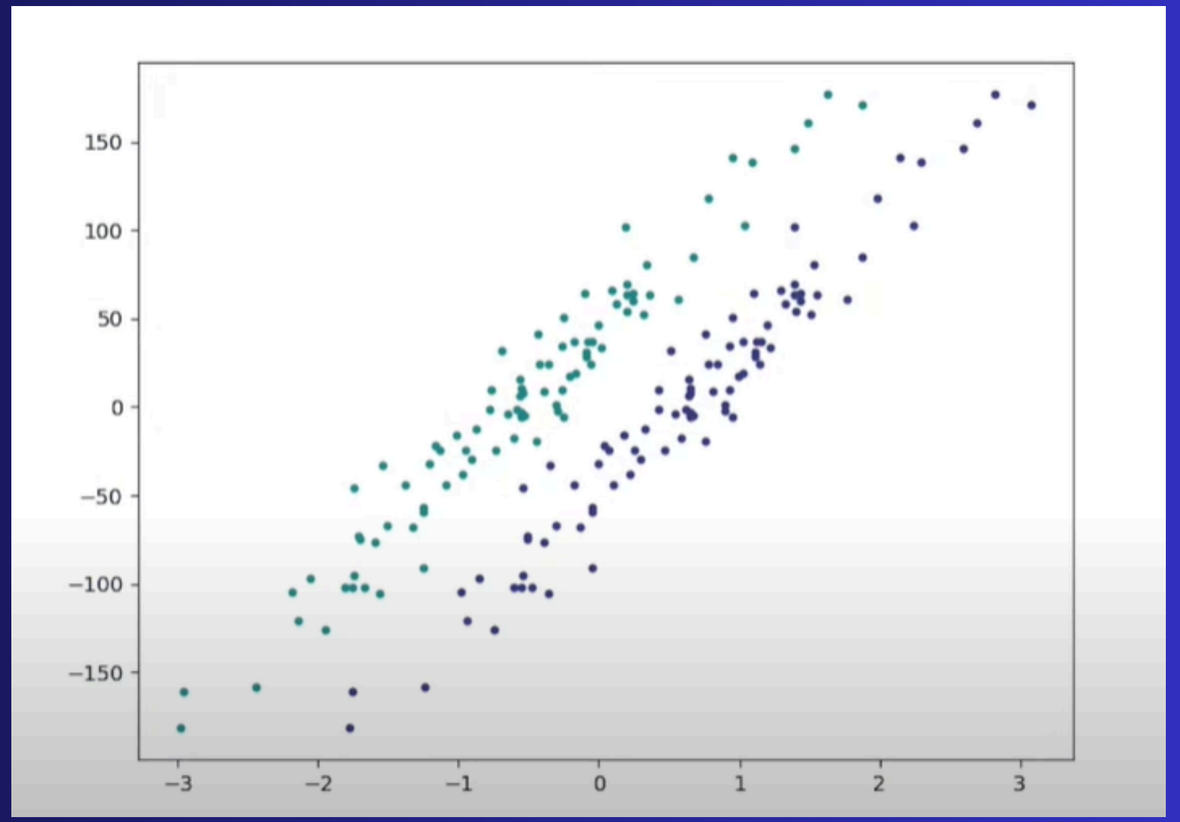
IN DEEP LEARNIG

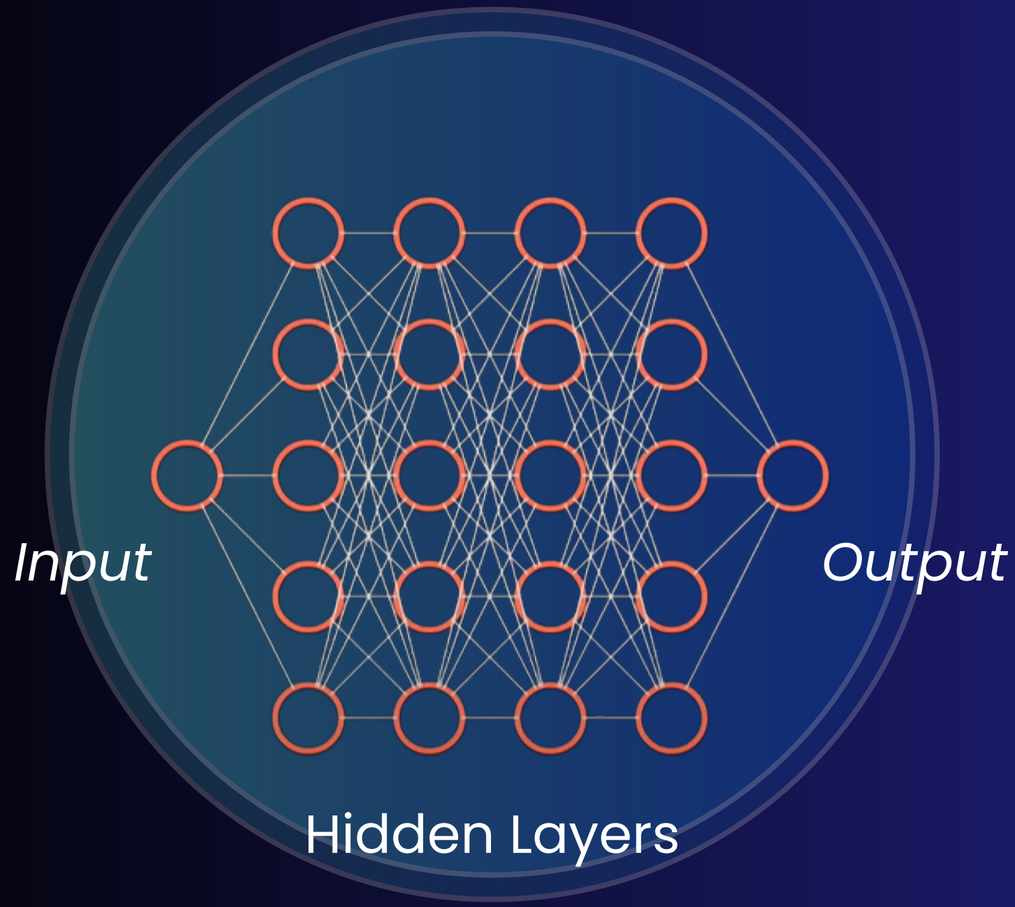
GET STARTED

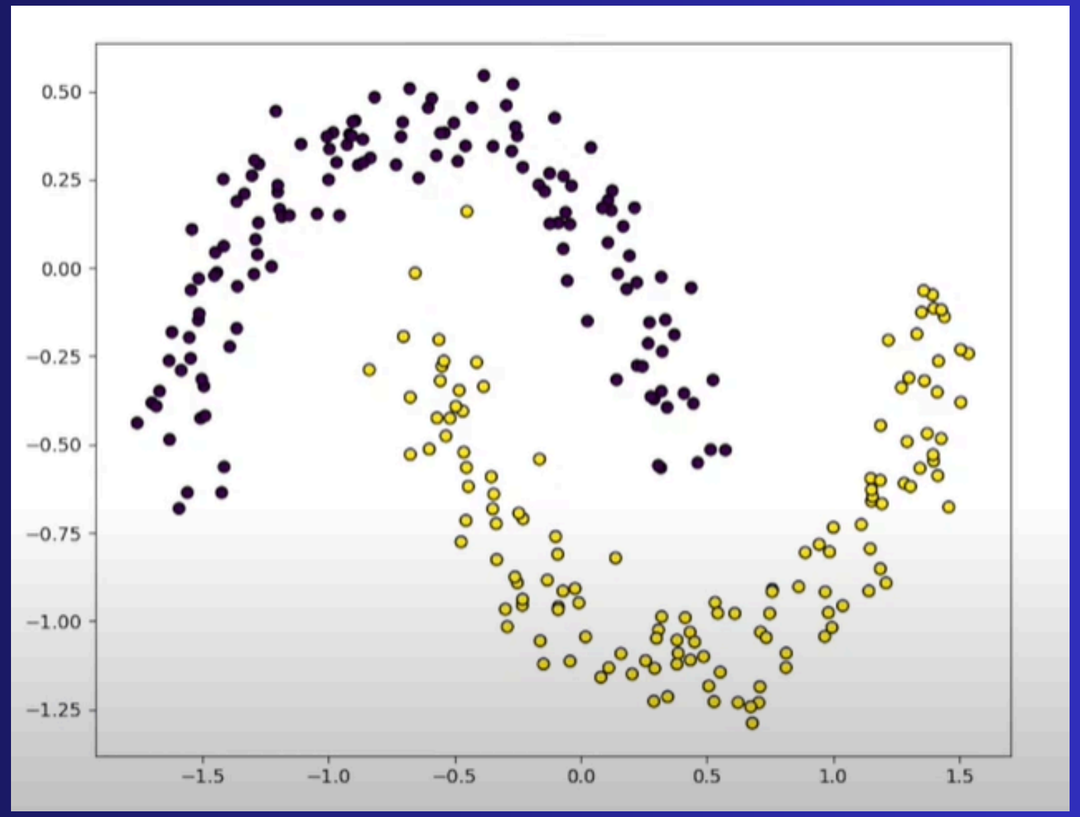
- Apply non-linear transformations to neuron outputs.
- Decide whether a neuron should be activated or not.
- Essential for learning complex patterns in neural networks.



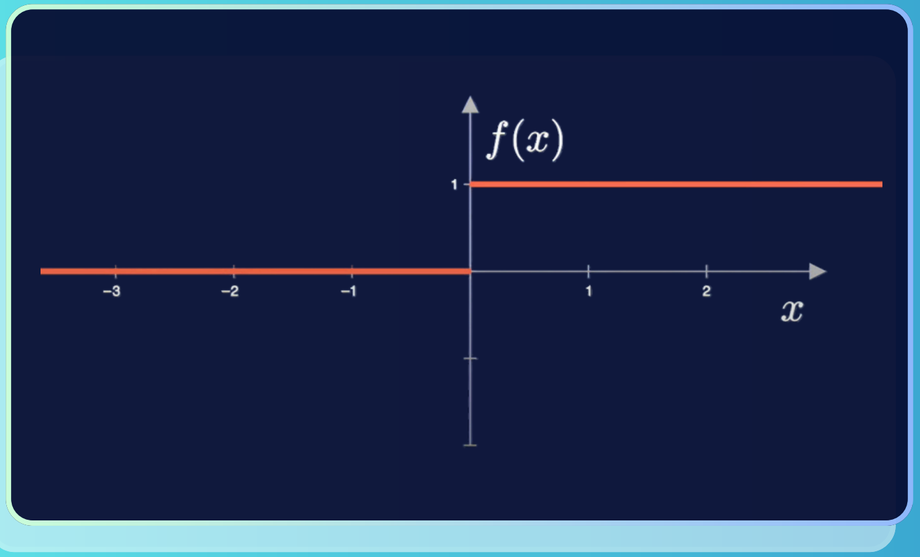

Input
Output
Hidden
Layers


150
100
50
0
−50
−100
−150
−3
−2
−1
0
1
2
3


Input
Output
Hidden Layers


0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

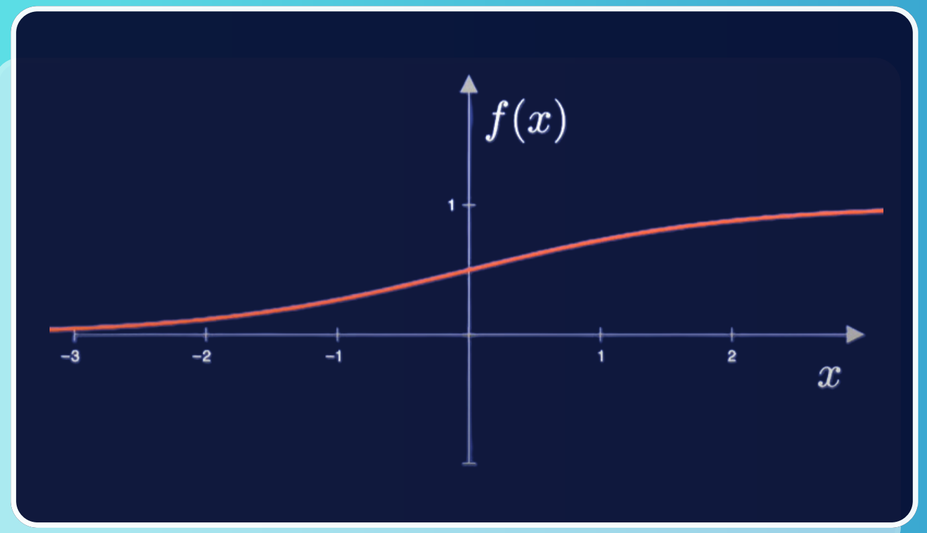
Step Function



FORMULA $$f(x) = \begin{cases} 1 \quad if \quad x \geq \theta \\ 0 \quad otherwise \end{cases}$$

- Output 1 if input > threshold, otherwise 0.
- Too simple and rarely used in practice.

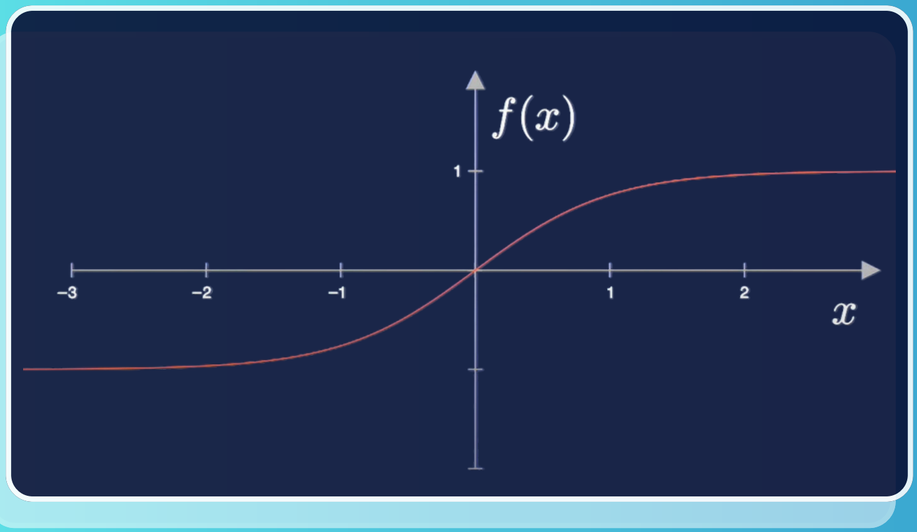
Sigmoid Function



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

- Outputs values between 0 and 1 (like a probability).
- Often used in the last layer for binary classification.

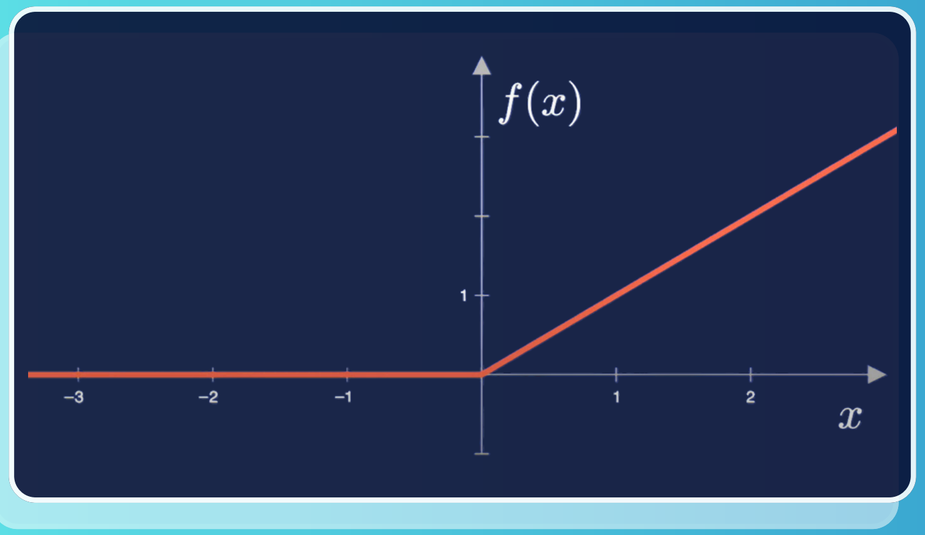
Tanh



FORMULA $f(x) = \frac{2}{1+e^{-2x}} - 1$

- Outputs values between -1 and 1.
- A scaled version of sigmoid, often used in hidden layers.

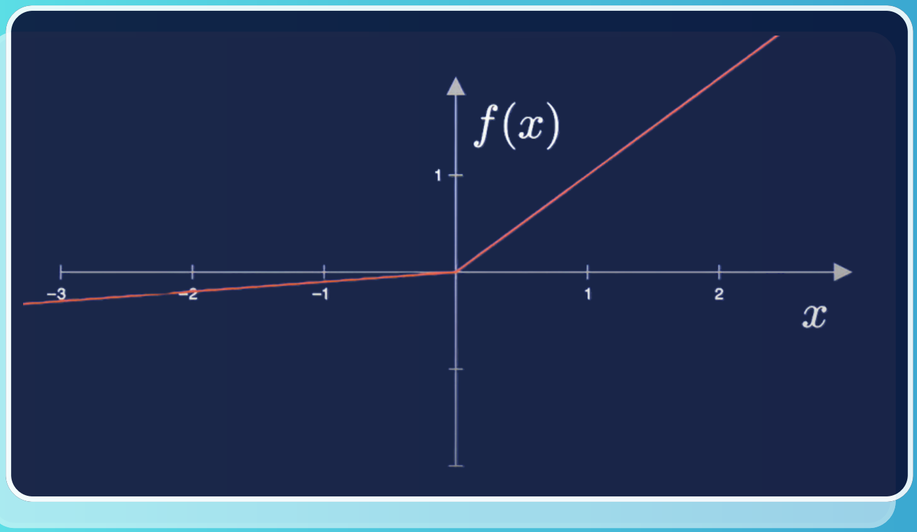
ReLU



FORMULA $f(x) = max(0, x)$

- Outputs 0 for negative inputs and the input value for positive ones.
- Simple, efficient, and widely used.

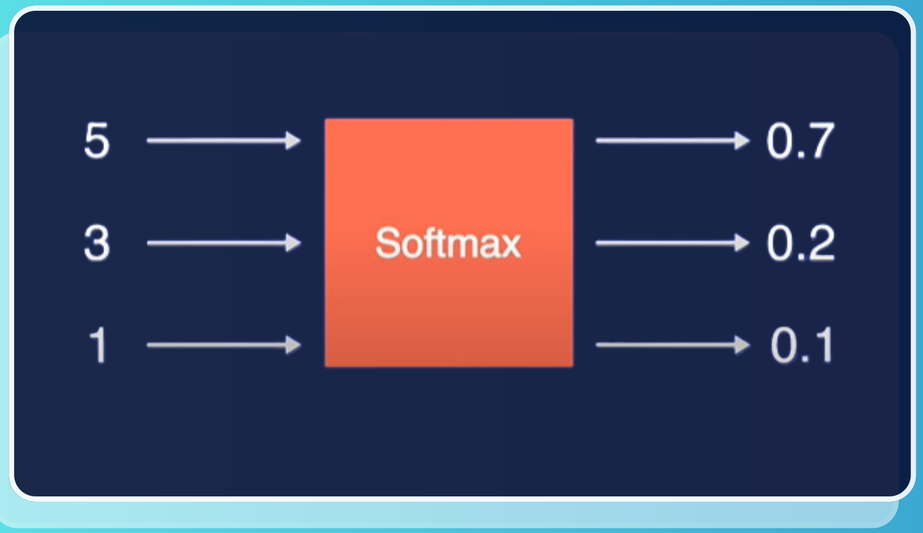
Leaky ReLU



FORMULA $$f(x) = \begin{cases} x & if \quad x \geq 0 \\ a \cdot x & otherwise \end{cases}$$

- Similar to ReLU but allows a small, non-zero slope for negative inputs.
- Prevents "dying ReLU" problem (neurons stuck at 0).

SoftMax



FORMULA $$S(y_i) = \frac{e^{y_i}}{\sum e^{y_i}}$$

- Outputs probabilities between 0 and 1, summing to 1.
- Used in the last layer for multi-class classification.

```python
import torch
import torch.nn as nn



class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(NeuralNet, self).__init__()
        self.linear1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.linear2 = nn.Linear(hidden_size, 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        out = self.linear1(x)
        out = self.relu(out)
        out = self.linear2(out)
        out = self.sigmoid(out)
        return out
```

Implementation

This network is a binary classifier. It takes numerical input, processes it through hidden layers, and outputs a probability.

Example Problem: Spam Detection

- Input: Features like "word count," "number of links," etc.
- Output: A probability value (e.g., 0.9 means 90% chance of being spam).

(Double Click to Execute Code)

REFERENCES

- GeeksforGeeks. (2024, November 19). Activation functions in Neural Networks. GeeksforGeeks. https://www.geeksforgeeks.org/activation-functions-neural-networks/
- AssemblyAI. (2021, December 6). Activation functions in neural networks explained | Deep learning tutorial [Video]. YouTube. https://www.youtube.com/watch?v=Fu273ovPBmQ
- Monteiro, T. C. (2024, April 10). How AI Models Think: The Key Role of Activation Functions with Code Examples. freeCodeCamp.org. https://www.freecodecamp.org/news/activation-functions-in-neural-networks/
- Activation Functions in Neural Networks [12 Types & Use cases]. (n.d.). https://www.v7labs.com/blog/neural-networks-activation-functions

REPORTED BY: QUINTO, HONEY PEARL G. & SONQUIPAL, EMMANUEL Y.