Forward Propagation, Backward Propagation, Perceptrons, Neural Networks

- 💡 **Different real-world examples**
- 💉 **Code snippets**
- 🏋️ **Exercises for practice**
- 🔁 Covers: **Forward Propagation, Backward Propagation, Perceptrons, Neural Networks**

🧠 **Deep Learning Essentials – Notes, Examples & Exercises**

🔁 **1. Forward Propagation**

💡 **What is it?**

It's the process of pushing inputs through a model to get an output.

📌 **Example: Loan Approval Prediction**

You want to predict whether a loan will be approved based on income and credit score.

```
import numpy as np

X = np.array([[50_000, 700]])  # income, credit score
W = np.array([[0.0001], [0.01]])  # trained weights
b = -3  # bias

def sigmoid(x):
  return 1 / (1 + np.exp(-x))

Z = np.dot(X, W) + b
output = sigmoid(Z)

print("Approval probability:", output[0][0])
```

🏋️ **Exercise 1:**

Modify the code above to:

- Add a third feature: number of previous loans

Mentor Sowmya

- Try different weights and biases
- Change the activation to ReLU and observe

🔁 2. Backward Propagation

💡 What is it?

Backpropagation adjusts weights by calculating how much each one contributed to the error.

📌 Example: House Price Estimation

```
import numpy as np

X = np.array([[1200, 2]])  # 1200 sq.ft, 2 bedrooms
y_true = np.array([[250000]])  # actual price

W = np.array([[150], [5000]])  # $150/sq.ft, $5000 per bedroom
b = np.array([[20000]])

# Forward pass
y_pred = np.dot(X, W) + b
loss = 0.5 * (y_true - y_pred)**2

# Backward pass
error = y_pred - y_true
dW = np.dot(X.T, error)
db = error

# Update
lr = 0.00001
W -= lr * dW
b -= lr * db

print("Updated weights:", W.flatten())
```

Mentor Sowmya

```
print("Updated bias:", b)
```

🏋️ **Exercise 2:**

- Try a batch of 3 different houses.
- Plot loss before and after update.
- Use different learning rates and see what happens.

🧍 **3. Perceptron**

💡 **What is it?**

A single-layer neural model — foundation of all neural networks.

📌 **Example: AND Gate Simulation**

```
import numpy as np

def perceptron(x, w, b):
    result = np.dot(x, w) + b
    return 1 if result >= 0 else 0

# AND gate
inputs = np.array([[0,0], [0,1], [1,0], [1,1]])
weights = np.array([1, 1])
bias = -1.5

for x in inputs:
    print(f"{x} -> {perceptron(x, weights, bias)}")
```

🏋️ **Exercise 3:**

- Implement OR gate and NAND gate.
- Try using 3-input logic gates.
- Change bias and observe behavior.

Mentor Sowmya

🧱 4. Simple Neural Network

💡 What is it?

A neural net with a **hidden layer**, capable of learning non-linear patterns.

📌 Example: XOR Logic Gate

```
import numpy as np

def sigmoid(x):
  return 1 / (1 + np.exp(-x))
def relu(x):
  return np.maximum(0, x)

# Inputs & targets
X = np.array([[0,0], [0,1], [1,0], [1,1]])
y = np.array([[0], [1], [1], [0]])

# Random weights
np.random.seed(0)
W1 = np.random.randn(2, 2)
b1 = np.zeros((1, 2))
W2 = np.random.randn(2, 1)
b2 = np.zeros((1, 1))

# Forward pass
hidden = relu(np.dot(X, W1) + b1)
output = sigmoid(np.dot(hidden, W2) + b2)

print("Predicted XOR outputs:")
print(output)
```

🏋️ Exercise 4:

Mentor Sowmya

- Replace ReLU with tanh
- Add more neurons to hidden layer
- Train the network using backpropagation (challenge!)

🔁 **Extra Practice Ideas:**

🧠 **Conceptual**

- Describe in words how backpropagation works.
- When should you use sigmoid vs ReLU?

👷 **Coding**

- Build a perceptron that learns using a training loop.
- Visualize decision boundary of a perceptron in 2D.
- Build a simple neural net that learns to mimic a sine wave.
