Documentation Experiment : 2

Name - Aditya Luthra
Roll no - 2K22/CO/30

1. Introduction

This experiment demonstrates the implementation of a simple neural network to classify both linearly separable and non-linearly separable datasets, such as the moons dataset, using **PyTorch**. The experiment includes training with different neural network architectures, one with no hidden layers and another with a hidden layer.

2. Libraries Used

The following Python libraries are used in this project:

- **`numpy`** – For numerical operations.
- **`matplotlib.pyplot`** – For visualizations.
- **`sklearn.datasets`** – To generate synthetic datasets (`make_classification` for linearly separable data, `make_moons` for non-linearly separable data).
- **`torch`** – For defining and training the neural network using PyTorch.
- **`torch.nn`** – For the neural network modules and layers.
- **`torch.optim`** – For optimization algorithms (Adam optimizer).

3. Dataset Preparation

Two types of datasets are generated:

- **Linearly separable dataset** using `make_classification`.
- **Non-linearly separable dataset** using `make_moons`.

Both datasets are plotted to visually show the separability between the classes.

4. Neural Network Model

A simple feedforward neural network is implemented using PyTorch's `nn.Module`. Two architectures are tested:

- **No hidden layers**: The network directly maps the input to the output with one output layer.
- **One hidden layer with 10 neurons**: A hidden layer is added to show how more complex models can better handle non-linearly separable data.

The architecture consists of:

- **Linear layers** for both the input-output and hidden layers.
- **ReLU** activation function for the hidden layer.
- **Sigmoid** activation function for the output layer (for binary classification).
- **Binary cross-entropy** loss function for classification.
- **Adam optimizer** with a learning rate of 0.005.

5. Model Training and Evaluation

The model is trained for 50 epochs . The performance is evaluated based on accuracy and loss metrics.

- **For the linearly separable dataset**: The model with no hidden layers should perform well.
- **For the non-linearly separable dataset**: The model with a hidden layer should perform better, showing the importance of network complexity for non-linear decision boundaries.

6. Results

- **For the linearly separable dataset**: The model with no hidden layers performs well.
- **For the non-linearly separable moons dataset**: Adding a hidden layer improves the model's performance, illustrating the benefit of more complex architectures for non-linear problems.

7. Conclusion

This experiment shows how the architecture of a neural network can influence its ability to classify different types of datasets. A simple architecture suffices for linearly separable data, but for non-linearly separable data, deeper models with hidden layers are required for better performance.

GitHub Link : https://github.com/Adiiii02/DL-LAB/tree/main/EXP-2