Lab 2: Neural Network and Deep Learning

August 2023

1 In This Lab

1.1 Topics to Cover

- Neural networks and Deep Learning
- Data-sets used in this lab: samples data, MNIST for number classification, and fashion MNIST

1.2 Requirements

- Deep learning concepts and Jupyter Notebook.
- Libraries: PyTorch, Scikit-learn, numpy, matplotlib, seaborn

1.3 Tasks

1. **Import necessary libraries:** Begin by importing the required libraries, including PyTorch, Pandas, NumPy, and Scikit-learn.
2. **Load the dataset:** Load into a Pandas DataFrame and inspect the first few rows to understand the data.
3. **Data Preprocessing:** Check for any missing values in the dataset and handle them appropriately (e.g., by filling with mean or median values). Perform any necessary feature scaling if needed.
4. **Data Visualization:** Create scatter plots or other suitable visualizations to explore the relationships between the features and the target variable (house prices).
5. **Data Splitting:** Split the dataset into training and testing sets. Use 70% of the data for training and 30% for testing.
6. **Initialize Model:** Create a model using PyTorch and train it on the training data.

7. **Model Evaluation:** Evaluate the performance of the trained model on the testing data using appropriate metrics.

8. **Making Predictions:** Use the trained model to make predictions on new data. You can either use sample data or create new data points.

9. **Experiment and Improve:** Experiment with different model parameters and observe how the model's performance changes.

2 Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a type of artificial neural network architecture used in deep learning. It consists of multiple layers of interconnected nodes, each performing transformations on input data to learn and represent complex relationships.

2.1 Key Components

- **Input Layer:** The first layer of the MLP that receives the input data. Each node in the input layer represents a feature or dimension of the input data.
- **Hidden Layers:** Intermediate layers between the input and output layers. These layers process the data using non-linear activation functions to learn and extract features.
- **Output Layer:** The final layer that produces the network's prediction or output. The number of nodes in this layer depends on the problem, such as binary classification, multi-class classification, or regression.
- **Neuron (Node):** Each node in a layer applies a weighted sum of its inputs, followed by an activation function. The activation function introduces non-linearity to the network, allowing it to learn complex patterns.
- **Weights and Biases:** Connections between nodes have associated weights that determine their contribution to the weighted sum. Each node also has a bias term.
- **Activation Functions:** Non-linear activation functions (e.g., ReLU, sigmoid, tanh) introduce non-linearity and enable MLPs to model complex relationships.

2.2 Training

- **Forward Propagation:** During inference, data flows forward through the network from the input layer to the output layer, producing predictions.
- **Backpropagation:** During training, the network's error is computed by comparing the predictions to the ground truth. The error is then propagated backward through the network to adjust the weights using gradient descent optimization.
- **Loss Function:** The loss function quantifies the discrepancy between predictions and ground truth. Common loss functions include mean squared error for regression and cross-entropy for classification.

2.3 Applications

- **Classification:** MLPs are used for tasks like image classification, text categorization, and sentiment analysis.
- **Regression:** MLPs can approximate functions for predicting continuous values, such as predicting housing prices or stock prices.
- **Pattern Recognition:** MLPs excel at learning complex patterns and extracting features from data.
- **Function Approximation:** MLPs can approximate complex functions, making them useful for tasks like modeling non-linear relationships in data.

2.4 Task 1: Iris flower classification

1. Follow *MLPClassification.ipynb* script
2. Plot loss over iterations

2.5 Task 2: house price prediction

1. Follow *MLPRegression.ipynb* script

2.6 Report

1. Create a new jupyter-notebook
2. Use the breast cancer data (Breast_cancer_wisconsin.csv) for data classification
3. Follow similar steps as in the previous example

3 Deep Learning

In this lab exercise, you will work with the famous MNIST dataset and apply the concepts of Neural Networks using PyTorch for image classification. The MNIST dataset consists of images of handwritten digits (0 to 9) and is widely used for testing machine learning algorithms.

3.1 Requirement

1. PyTorch to create a deep neural network model

3.2 Dataset

- MNIST
- Fashion MNIST

3.3 Task 1: MNIST classification

1. Follow *MNIST.ipynb* script
2. Plot loss over iterations

3.4 Report

1. Create a new jupyter-notebook
2. Use the Fashion MNIST data-set.
    - **Hint:** For this use *torchvision.datasets.FashionMNIST* instead of *torchvision.datasets.MNIST*
3. Do the same steps in the previous example