[Project Code: TFNN]
Taxi Fare Prediction using Neural Networks
Project Duration: 25-Feb-2024 – 16-Mar-2024
Submission Information: (via) CSE-Moodle

Objective:
This project focuses on predicting the fare prices for taxi rides using neural networks. The dataset contains key information such as unique trip identifiers, fare amounts, pickup and dropoff locations (longitude and latitude), pickup date and time, and the number of passengers. The objective is to understand the dataset, and then build the model that can accurately predict the fare prices of future taxi rides.

DataSet: Filename: ***taxi.csv***
Consider 80% of the dataset (randomly selected) as training data, and the rest 20% as test data. Save this training and test data. In the rest of this assignment, you will be asked to develop various NN classifiers, and each classifier should be trained using this training data, and evaluated using this test data. In other words, each NN should be trained and tested using the same datasets.

Data Description: The **attribute** Information is given as follows.

- key - a unique identifier for each trip
- fare_amount - the cost of each trip in usd
- pickup_datetime - date and time when the meter was engaged
- passenger_count - the number of passengers in the vehicle (driver entered value)
- pickup_longitude - the longitude where the meter was engaged
- pickup_latitude - the latitude where the meter was engaged
- dropoff_longitude - the longitude where the meter was disengaged
- dropoff_latitude - the latitude where the meter was disengaged

Tasks to be done:

1. Building your own Neural Network

Build a neural network and perform the classification task with the specifications mentioned in Parts 2 and 3. ***DO NOT use scikit-learn for this part***.Your implementation should have the following modules:

I. *Preprocess:* Use this module to preprocess the data and divide into train and test.
II. *Data loader*: Use this module to load all datasets and create mini batches, with each minibatch having 32 training examples.
III. *Weight initialiser*: This module should initialize all weights randomly between -1 and 1.
IV. *Forward pass:* Define the forward() function where you do a forward pass of the neural network.
V. *Backpropagation*: Define a backward() function where you compute the loss and do

a backward pass (backpropagation) of the neural network and update all weights.

VI. *Training*: Implement a simple mini batch SGD loop and train your neural network, using forward and backward passes.

VII. *Predict*: To test the learned model weights to predict the classes of the test set.

2. *ANN Specification 1*

- No of hidden layers: 1
- No. of neurons in hidden layer: 32
- Activation function in the hidden layer: Sigmoid
- 1 neuron in the output layer.
- Activation function in the output layer: Linear
- Optimisation algorithm: Mini Batch Stochastic Gradient Descent (SGD)
- Loss function: Mean Squared Error (MSE)
- Learning rate: 0.01
- No. of epochs = 200
- ***DO NOT use scikit-learn for this part.***

3. *ANN Specification 2*

- No of hidden layers: 2
- No. of neurons in the 1st hidden layer: 64
- No. of neurons in the 2nd hidden layer: 32
- Activation function in both the hidden layers: ReLU
- 1 neuron in the output layer.
- Activation function in the output layer: Linear
- Optimisation algorithm: Mini Batch Stochastic Gradient Descent (SGD)
- Loss function: Mean Squared Error (MSE)
- Learning rate: 0.01
- No. of epochs = 200
- ***DO NOT use scikit-learn for this part.***

4. *Implementation with scikit learn*

- Use the MLP implementation of scikit-learn.
- Use the specifications from Part 2 and Part 3, and use the same training and test data.

Note: *The program can be written in C / C++ / Java / Python programming language from scratch. No machine learning /data science /statistics package / library should be used.*

Your code must output the following for each specification:

1. Plot the training accuracy and test accuracy after every 10 epochs (in a single plot)
2. Print the final training accuracy
3. Print the final test accuracy

Submission Details: (to be submitted under the specified entry in CSE-Moodle)

1. ZIPPED Code Distribution in CSE-Moodle
2. A brief (2-3 page) report/manual of your work

Submission Guidelines:

1. You may use one of the following languages: C/C++/Java/Python.
2. Your Programs should run on a Linux Environment.

3. Your program should be standalone and should **not** use any *special purpose* library for Machine Learning for the NN classifier algorithm. Numpy and Pandas may be used. And, you can use libraries for other purposes, such as generation and formatting of data. 4. You should submit the program file and README file and **not** the output/input file.
5. You should name your file as <GroupNo_ProjectCode.extension>.
   (e.g., *Group99_TFNN.zip* for code-distribution and *Group99_TFNN.pdf* for report)
6. The submitted program file *should* have the following header comments: # Group Number
   # Roll Numbers : Names of members (listed line wise)
   # Project Number
   # Project Title
7. Submit through CSE-MOODLE only. Link to our Course page:
   https://moodlecse.iitkgp.ac.in/moodle/course/view.php?id=561

You should not use any code available on the Web. Submissions found to be plagiarized or having used ML libraries (except for parts where specifically allowed) will be awarded zero marks.

For any questions about the assignment, contact the following TA:

Kajori Ghosh (Email: kajorighosh4@gmail.com)