Neural and Evolutionary Computation

Activity 1: Prediction with Back-Propagation and Linear Regression

Mark Safronov
MESIIA 2024-2025

The assignment has been done as a public GitHub repository: https://github.com/hijarian/urv-nec-assignment-1

The dataset analyzed in the work is Restaurant Revenue Prediction Dataset from Kaggle. (https://www.kaggle.com/datasets/anthonytherrien/restaurant-revenue-prediction-dataset)

The idea of the dataset is given the set of characteristics of the restaurant, predict its revenue. This is a regression problem.

Summary of the work is as follows.

1. The dataset has been analyzed for particularities. Two overly correlated input variables has been combined into new one using Principal Component Analysis. Several input variables has been determined to not correlate with the output variable at all. After cleaning up the completely unnecessary data, two base datasets has been created as the result. One contained all the input variables (the “full dataset”) and another one didn’t contain the variables deemed useless (the “reduced dataset”).

All the data handling happened in the Jupyter Notebook https://github.com/hijarian/urv-nec-assignment-1/blob/main/01-prepare-data.ipynb . Repository already contains both pre-processed datasets.

2. A custom homemade NeuralNet class in Python has been implemented. It uses the online backpropagation of errors without cross-validation.

A test run of this class revealed that it behaves inadequately, with the performance more closely resembling the classification model instead of a regression one. The mean absolute percentage error was never below 25%

A little of exploration of the parameters hyperspace has been performed. 36 variants of the settings for the neural network was checked for performance.

All the testing is extensively covered in the Jupyter Notebook https://github.com/hijarian/urv-nec-assignment-1/blob/main/02-use-neural-network.ipynb

3a. A by-the-book multilinear regressor from Scikit has been trained on the reduced dataset. All parameters are internal defaults of the library. It trained incomparably faster than the homemade NeuralNet class and successfully approximated the output variable with the MAPE ~8%.

Then, a by-the-book neural network setup using PyTorch has been used to perform the same regression task. It also trained with the same high speed and provided the MAPE of ~6%.

This has been covered in the Jupyter notebook
https://github.com/hijarian/urv-nec-assignment-1/blob/main/03-compare-with.ipynb

3b. The same Scikit and Pytorch setup has been applied to the full dataset.

It was shown that the performance of both models is worse by a minuscule degree using the full dataset instead of the reduced one.

The Jupyter notebook was almost the copy of the `03-compare-with.ipynb` with the only difference being the source data file and the settings for the values scaler. It is placed in the repository as a file https://github.com/hijarian/urv-nec-assignment-1/blob/main/03-compare-with.fulldata.ipynb