Nasri Chiheb

UNIVERSITAT ROVIRA I VIRGILI

Neural and Evolutionary Computation

A1: Prediction with Back-Propagation and Linear Regression

Introduction:

The goal of the first assignment is to implement a Back-propagation algorithm in order to make predictions on a dataset. In this report, we will describe the implementation and discuss about the decisions made. Then how to run part . We will finish by the analyze of BP and MLR results, conclusion, and the improvements. Please find attached to this report the commented code for more details.

Description of the implementation:

The Back-propagation algorithm is implemented in Python. I tried to implement with Julia but discovering a new programming language takes time so after multiple tries, I decided to go with Python which will be easier for me. Backpropagation algorithm do not need any external package except Matplotlib for final plots. MLR, implementation was made with MATLAB it uses premade functions.

Let us now discuss about implementation decisions. First training data and test data is scales in a [0.1;0.9] range as described in the instructions. Then it is unscaled after the predictions. Network creation can accept any configuration that follows this schema “Input-Hidden Layer-Output”. The algorithm creates a fully connected network. At the beginning weight are initialized randomly . The transfer function uses sigmoid function as it is the most used. The algorithm follows online learning , weight are updates for each training pattern but in next version it will be a partial batched back propagation to get the speed of online BP and the consistency of batched BP as described in the implementation instructions.

MLR function is quite easy, it uses “fitlm” function which automatically creates the model and “predict” function to predict.

How to run:

For BP

Prerequisite :

- Python 3.7
- Matplotlib package
- Open BackPropagation.py
- A1-turbine.txt file in the same root that BackPropagation.py

Run :

- Open BackPropagation.py
- Select parameters in line 215
- Run the algorithm

For MLR:

- Matlab
- MLR.m and importfile.m
- A1-turbine.txt file in the same root that BackPropagation.py

Run:

- Open and MLR.m

Results:

In this section we will discuss about the results. Let us first of all see the results through some plots.

```
Final relative absolut error: 1.4238317319693352%
Init parameters:
 Inputs = 4  layers = 10, Output = 1
 learning Rate = 0.2  Epoch = 1000

--- 13.943222284317017 Execution time ---
```

```
>> MLR
Error rate MLR : 4.8646%
```

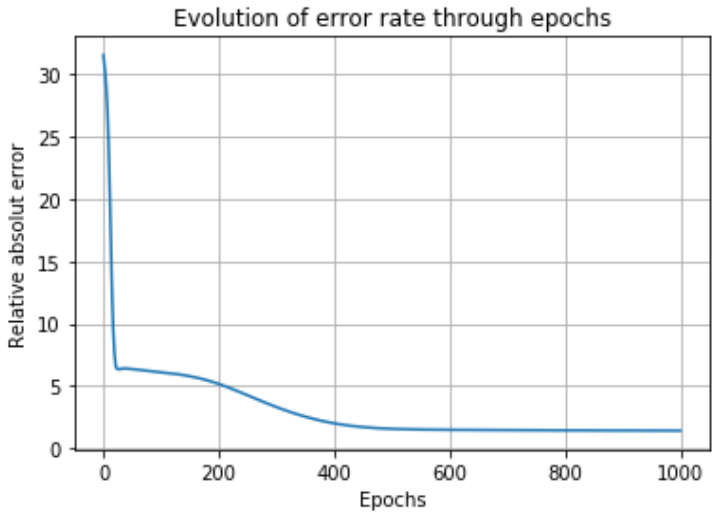

Fig1

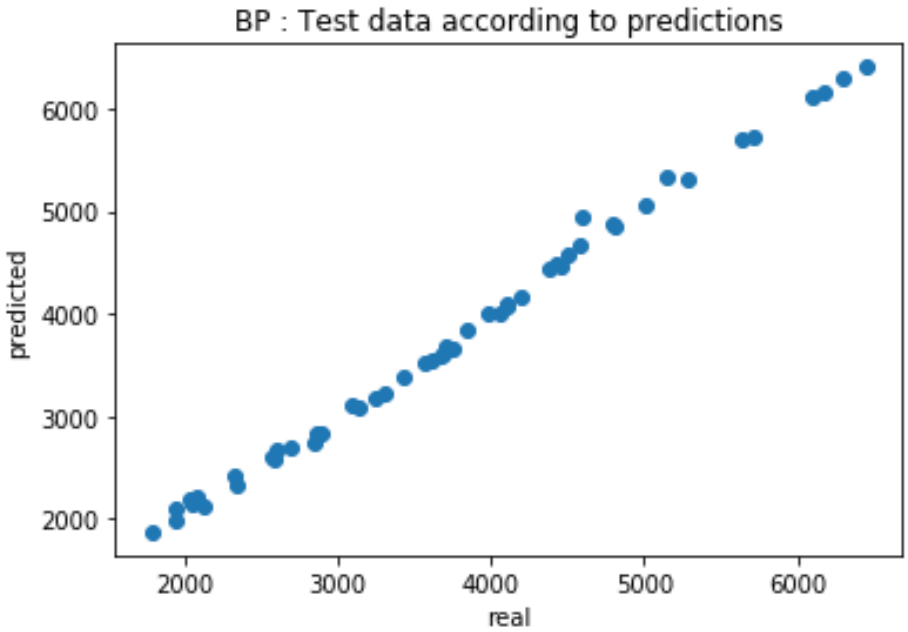

Fig2

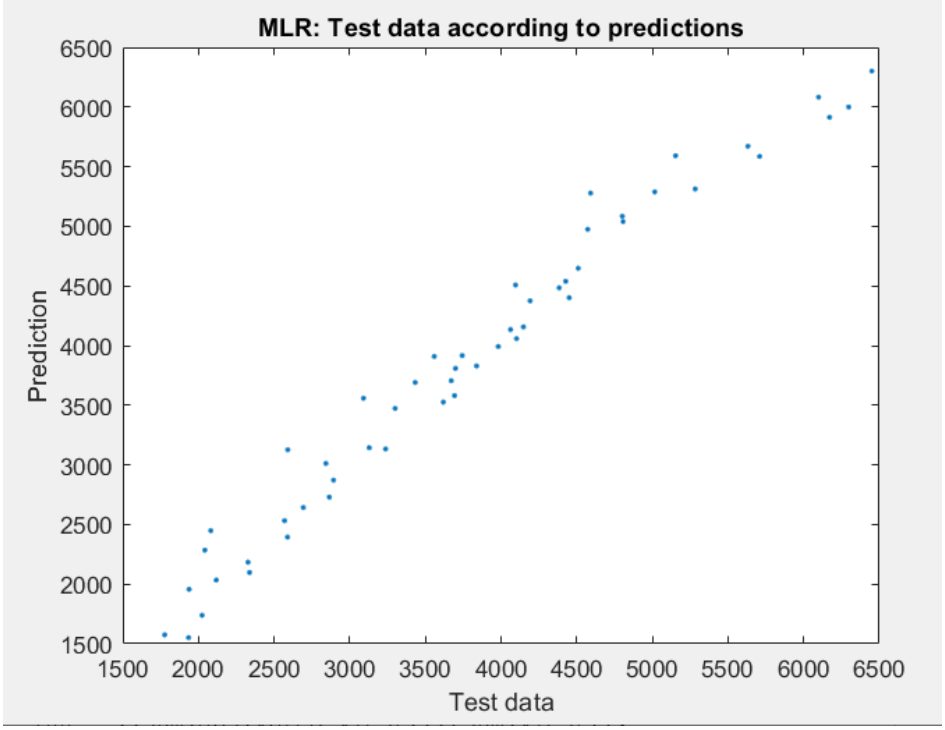

Fig3

Predictions are evaluated with the relative Absolut error. For BP we use it two times. First, in every epoch to get evolution of the error rate and plot it. It will be useful to automatically get the best number of epochs and avoid overfitting as we can see if fig 1. Fig1 shows that the algorithm starts overfitting between 500 and 600 epochs. Second error measure is at the end with trained network to get final result(this will be useful to evaluate the algorithm later with cross-validation).

Fig 2 and 3 shows that predictions are overall accurate as the plot is “almost” linear function. Error rate is below 5% so it is acceptable for both algorithms.

Finally, BP results are more accurate than MLR. We have to consider that MLR algorithm is here more basic, data is not preprocessed no cross validation.

Conclusion:

To conclude, results are quite good for both algorithms. This assignment took a little time to start for me as I wanted to try Julia at the beginning, but it was time consuming, so I chose python instead. This is only V1 of the project more improvements are coming. Finally, implemented BP algorithm was a true help to understand how it works, even if for me ideas were not totally clear at the beginning but know it seams better. This method of programming from the description of an algorithm is quite new (as described in BI) it takes more time but helps to understand.

Improvements:

This part is dedicated to upcoming improvements for this assignment:

- Implement cross validation
- Implement function to decide best input parameters(weights, epochs, learning rate)
- Go from online BP to partial batched BP
- More user-friendly interface for BP