Project 2

Maziar Kosarifar

November 2019

Abstract

This paper compares the use and effectiveness of Artificial Neural Network with that of Logistic Regression in solving classification problems. It also compares the use of multi-layer perceptron with methods of Linear Regression, OLS, Lasso, and Ridge in solving the regression problem of simulating Franke's function. The implementation of multi-layer perceptron is done in a way that it can be used both for solving Regression, and Classification problems.

Introduction

The aim of this project is to study the performance and reliability of Logistic Regression and Multi-Layer Perceptron. We see how MLP can be used for studying both regression and classification problems. We introduce two study cases, Franke's function, and dataset on probability of default of credit card clients. We compare Ordinary Least Square with MLP, and study the result and the execution time.

In working on this project I have used the books Machine Learning: An Algorithmic Perspective [Mar15], and Introduction to Evolutionary Computing [EJ15], as main sources for the algorithms implemented for MLP, and LR.

1 Test Cases

In this section we will study the two different data sets used to compare the methods we have implemented.

First we study the simulation of Franke's function, comparing the result and execution time of a multilayer perception, with Ordinary Least Squares. The code for this comparision can be found in franke˙test.py file.

In the second part we will study the data for the classification problem, which deals with probability of default of credit card clients. There we compare the results of Logistic Regression with MLP, and study their execution time.

1.1 Franke's function(Regression)

Franke's function is a widely used function to study methods of regression analysis. Here we are using Franke's function to compare the performance of our multi-layer perceptron with Linear regression methods implemented for the last project.

Here is the definition of Franke's function for real values of (x, y).

$$\begin{aligned} f(x,y) = \frac{3}{4}exp(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}) + \frac{3}{4}exp(-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}) \\ + \frac{1}{2}exp(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}) - \frac{1}{5}exp(-(9x-4)^2 - (9y-7)^2) \end{aligned} \tag{1}$$

The Franke function is defined for $x, y \in [0, 1]$.

Based on the results of the last project, Ordinary Least Square had the best result for Franke's function.

1.2 Probability of default of credit card clients(Classification)

The dataset used for testing the Classification part of the implementation of Artificial Neural Network, and Logistic Regression is provided by UCI [Yeh]
This dataset is a set of 24 attributes, and the final binary variable of 0, or 1. (To default or not to default).
Since this dataset has been filled manually, it is important to run a reprocessing check on the data, to make sure all the entries are in their defined domain. The rows with attributes outside of their domain have been removed. Initially there is 30000 rows available in the dataset, although there will be only 28121 of them left after reprocessing the data.
The considered attributes include the credit value, gender, education, marital status, age, and history of past payments. [Yeh]

2 Methods Implemented

In this section we introduce the methods implemented for this project, Logistic Regression, and Artificial Neural Network. I have also used the Python class that was implemented for the first project for Linear Regression.

2.1 Logistic Regression

Logistic Regression is a method in statistics that is used to study and classify a set of data, given that they have independent properties. In this method we find the probability of one of the each outcome, and the one that is the most probable will be considered the outcome of the algorithm.
Logistic Regression, uses a Logistic function, Sigmoid. This can be represented as:

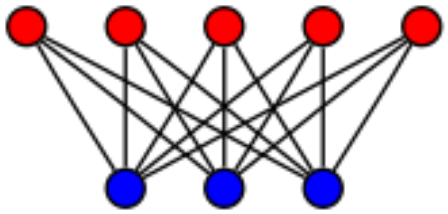
$$P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n)}} \tag{2}$$

Where the value of β_i is the weight given to the i-th feature of the dataset. And P will be a value between 0, and 1. For a binary system with only two defined outcomes, for $P > 0.5$ we can assume the value 1, and for $P < 0.5$ we can assume the outcome 0.
In this implementation the gradient method is the method used to improve the weight of each feature. And there is an early-stopping option, in the code that when the changes between two step is smaller than a pre-defined value, it can terminate the training process earlier. For a better result it is best to lower the value of learning rate, but that will make the learning process more costly. So it is important to find a learning rate that can balance the result with the processing duration.

2.2 Artificial Neural Network

Artificial Neural Network is a method used in Data Science that is inspired by biological Neural Networks, and the connection between neurons. Neurons can have connections to one another, and each connection has a weight, which is used to produce the outcome value of that neuron.
The method implemented for this assignment is a simplified version of that called Multi-layer perceprton. In this method the connection between Neurons are limited. Each set of neurons will use the ones from the previous layer as input values, and their outcome will be used for their next neighboring layer.
The connections are build in a way that each two neighboring layers will structure a complete bipartite graph. And the input will start from one end and through a process called "feed-forward" it will go through each layer, and will produce the output. And then the error is assessed based on the output of the last layer, compared to the expected value.
And another process called the "back-propagation" is used to train the network, and adjust the weighs assigned to each vector, to decrease the error.

Figure 1: Figure downloaded from "Complete bipartite graph" at wikipedia.org



Due to this simplification I have defined a class for each Layer, that will keep track of weights assigned to each node, and biases defined for that layer. And the method forward, and backward, for that

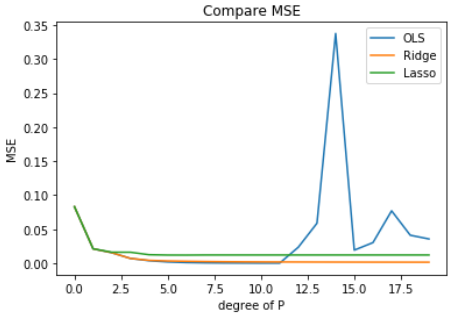
This implementation has been tested against a simpler MLP code written for the second project of the course INF3490 – Biologically inspired computing.

3 Results

3.1 Regression test

As mentioned earlier we are using the Franke's function as the study case for comparing the use of a Neural Network with a linear regression, Ordinary Least Square.
I have decided to use OLS, since it produced the best result for the lower polynomial degrees in the last assignment for the Franke's function.

Figure 2: Comparing MSE score of OLS, LASSO, and Ridge for Franke's function



The design used for the MLP is a simple 3-layer design. The first step has 2 inputs, and 3 outputs. The hidden layer with 3 inputs, and 2 outputs, and the final layer, 2 inputs, and a single output.
I have compared the result of the two different methods (OLS, MLP) based on their final error rate, and process cost, for increasing portion of inputs.

Figure 3: Network used for Franke's test

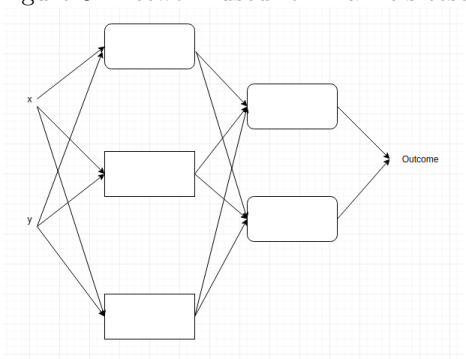
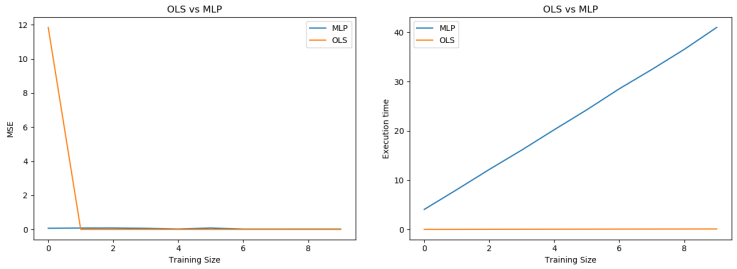


Figure 4: OLS vs MLP



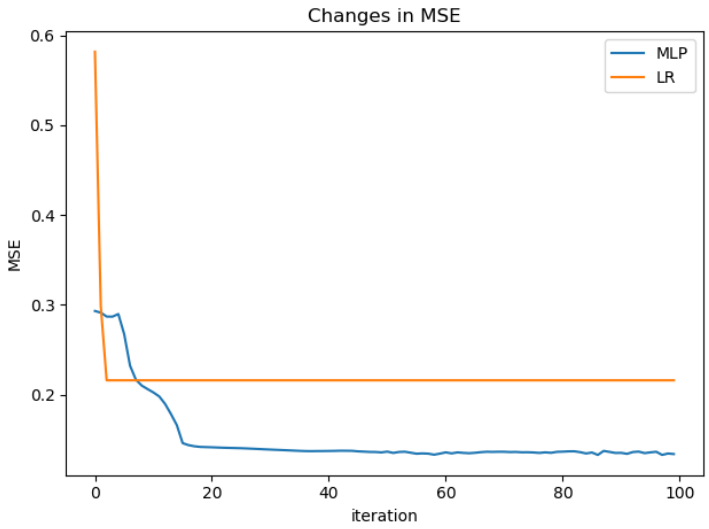
In figure 4, we can see that although MLP can produce a result will low level of errors, but the cost of running MLP increases drastically based on the input size.

3.2 Classification Test

In this section we are using the credit default data, to compare Logistic Regression method with Multilayer Perceptron (MLP)
The network design used for the MLP is:

```
mlp2.new_layer({'input_size': 25, 'number_of_nodes': 20, '
    activation_function': 'tanh'})
mlp2.new_layer({'input_size': 20, 'number_of_nodes': 15, '
    activation_function': 'sigmoid'})
mlp2.new_layer({'input_size': 15, 'number_of_nodes': 5, '
    activation_function': 'sigmoid'})
mlp2.new_layer({'input_size': 5, 'number_of_nodes': 1, '
    activation_function': 'tanh'})
```

Figure 5: MSE score, MLP vs LogisticRegression



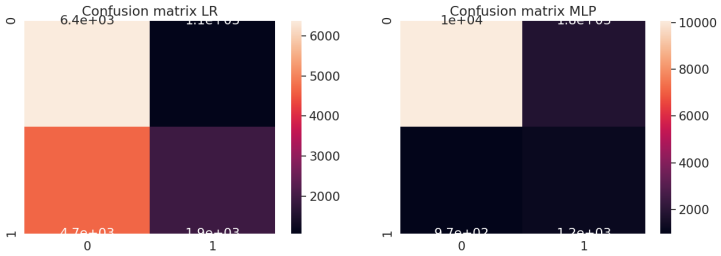
The figure 5, shows the difference between the results of Logistic Regression and MLP over 100 iterations.

To compare the result between the two classification, I have plotted the confusion matrix.
Showing that MLP produces significantly better accuracy than LR.

4 Conclusion

We have introduced a regression, and a classification problem. Compared OLS with MLP using Franke's function, and realized that although MLP can produce similar results with low MSE score, the execution time is significantly higer. Which makes it a costly process, with similar results to OLS.

Figure 6: Confusion matrix, MLP vs LogisticRegression



We have compared LR, with MLP and they ability to solve classification problems, and their accuracy with confusion matrix. Showing that MLP can produce far better results. LR produced 59.17% accuracy while MLP reaches an accuracy of 80.76%. This is considering that a more complicated MLP can produce better results. But it should be considered that a more complicated MLP will be costly both in terms of processing power and memory usage.

Bibliography

[EJ15] A.E. Eiben and Smith J.E. *Introduction to Evolutionary Computing*. 2nd ed. Berlin: Springer, 2015. ISBN: 9783662448748. DOI: http://dx.doi.org/10.1007/978-3-662-44874-8.

[Mar15] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. 2nd ed. International series of monographs on physics. Florida: CRC Press, 2015. ISBN: 9781466583337.

[Yeh] I. Yeh. *default of credit card clients Data Set*. URL: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients?fbclid=IwAR1LghATaMeyDYCFjtRQ4Ovw6bAP83994kU0CJYvPYBZixENY-MrVxsN5fM. (accessed: 13.11.2019).