LINEAR REGRESSION

November 24, 2019

Alberto Grillo ID: 4229492
Alberto Ghiotto ID: 4225586

***Abstract*— In Statistics, Linear regression refers to a model that can show relationship between two variables and how one can impact the other, this process aims at finding a linear relationship between a target and one or more predictors based on measured data.**
Linear regression plays an important role in the field of artificial intelligence and machine learning where it is considered one of the fundamental supervised algorithms. It consists in showing how the variation in the target, called "dependent variable", can be captured by changes in the observed and measured data, called "independent variables".
Linear Regression is a very powerful statistical technique and can be used to generate insights on consumer behaviour, analyze the marketing effectiveness, assess risk in financial services, understanding business and evaluate trends to make estimates or forecasts and much more.
In this laboratory the objective is to implement different example of linear regression, in both one and multi-dimensional cases and to asses their performances by graphic comparison and by computing the mean square error on the data.

I. INTRODUCTION

In this assignment some guidelines on how to develop the project were given, along with two different datasets on which to fit the linear regression model and on which to perform the testing.
The first dataset is the evolution of two Turkish stock market indexes: S&P500 and MSCI. The second is a collection of cars with 4 relative features, namely: miles per gallon (mpg), displacement (dsp), horsepower (hp) and weight.
The first task consisted in manipulating the given datasets in order to make them readable by the MATLAB pre-processing script that was gonna be used to fit the different linear regression model on the given data.
The next task consisted in implementing a linear regression model for three different istances of the problem plus a graphic comparation task for the first case:

1) One-dimensional problem without intercept on the Turkish stock exchange dataset
2) Graphic comparison of results from point 1 implemented on different random subsets of the whole dataset
3) One-dimensional problem with intercept on the Motor Trends Car data
4) Multi-dimensional problem on the Motor Trends Car data

The last task consisted in testing the implemented models. In particular it was asked to run points 1,3 and 4 from step 2 and compute the objective function using the square error loss (MSE) on different sized subset of the dataset.

II. BACKGROUND

In this section, a brief explanation of the implemented principles and equations will be given. The detailed steps can be found in the class slides.
Linear regression can be seen as the optimization problem of approximating a functional dependency based on measured data, however we cannot hope to find a value of w, where w is the parameter which approximates the model predicting the target t given the observation x (i.e $y = wx$), which is good for every point.
The generic goal is to minimize the mean value of the loss over the whole dataset, the objective function does exactly that:

$$J = \sum_{l=0}^{N} \lambda(t_l, y_l) \qquad (2.1)$$

Where J is the objective function or cost function, λ is the loss function, N is the number of observations, t_l is the measured target value and y_l is the inferred target value.

For this experiment the square error loss function was used, yielding to a formula for the mean square error such as:

$$J_{MSE} = \sum_{l=0}^{N} (t_l, y_l)^2 \qquad (2.2)$$

In all the below cases, the goal is to minimize the J_{MSE} with respect to the parameters with fixed data, which means finding a value of w such that:

$$\frac{d}{dw} J_{MSE} = 0 \qquad (2.3)$$

A. One-dimensional problem

In the one-dimensional problem the obtained value of w is:

$$w = \frac{\sum_{l=1}^{N} x_l t_l}{\sum_{l=1}^{N} x_l^2} \qquad (2.4)$$

Once having computed the value of w, the model is built as:

$$y = wx \qquad (2.5)$$

B. One-dimensional problem with intercept

In this case the solution can be found by centering around the mean $\bar{x}$ of x and $\bar{t}$ of t:

$$\bar{x} = \sum_{l=1}^{N} x_l \qquad \bar{t} = \sum_{l=1}^{N} t_l \qquad (2.6)$$

The current model is built as

$$y = w_1 x + w_0 \qquad (2.7)$$

Where the gain is:

$$w_1 = \frac{\sum_{l=1}^{N} (x_l - \bar{x})(t_l - \bar{t})}{\sum_{l=1}^{N} (x_l - \bar{x})^2} \qquad (2.8)$$

and the intercept is:

$$w_0 = \bar{t} - w_1 \bar{x} \qquad (2.9)$$

C. Multi-dimensional problem

The last implemented case required to organize the data in Nxd vectors, where N is the number of observation and d is the number of features observed. Since the data is now d-dimensional, w turns into a vector of dimension $1xd$ and t becomes a vector of dimension $1xd$.
The linear model is built as:

$$y = Xw \qquad (2.10)$$

Where this time:

$$w = (X^T X)^{-1} X^T t \qquad (2.11)$$

Once having fitted a linear regression model thanks to the above methods, it could be necessary to evaluate the performance of the framework, to do so it was required to compute J_{MSE} of the models.

III. Implementation and Results

A. Task 1 - Get the data

As anticipated in the introduction section, getting the data was just a matter of loading the .csv files in MATLAB. For this particular implementation, the *"readtable"* and *"load"* MATLAB function were exploited.

B. Task 2 - Fit a linear regression model

As per instructions, all the requested instances of the problem were solved by simply implementing the procedure highlighted in the slides, which is briefly shown in chapter 2.

1) One-dimensional problem without intercept: In this instance it was requested to predict the MSCI index by observing the SP's one from the stock market dataset. The resolution is quite straightforward since equation 2.4 lends itself well to being implemented in a MATLAB function: it is sufficient to compute the value of w and then plot the line identifying the linear relation as shown in equation 2.5. The computed model is then plotted by the function as shown in figure 1 below.

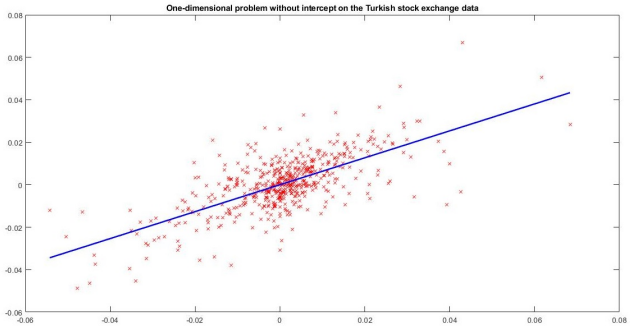

Fig. 1. 1-Dimensional linear regression without intercept

2) Graphic comparison of result from point 1: This task required to put into comparison different subset of the stock market dataset. In order to do so it was created a function which randomly splits the dataset in 9 equally sized subset and fits a linear model to each set, as done in the previous point. The function proceeds to then plot each model with the relative data as shown in figure 2.

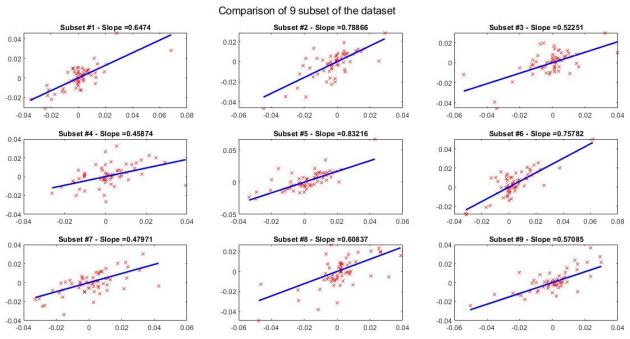

Fig. 2. Comparison of result from point 1 of different subset of stock exchange dataset

As it can be seen from figure 2, since the subset are quite small, the obtained slope are very different from each other.

3) One-dimensional problem with intercept: In this case it was required to predict the feature mpg with the weight's one from the motor car trends dataset. To fit the linear model it is sufficient to implement equations 2.6, 2.8 and 2.9 in a MATLAB function and then plot the resulting line as shown in equation 2.7. As in point 1, the function plots the linear model as shown in figure 3 below.

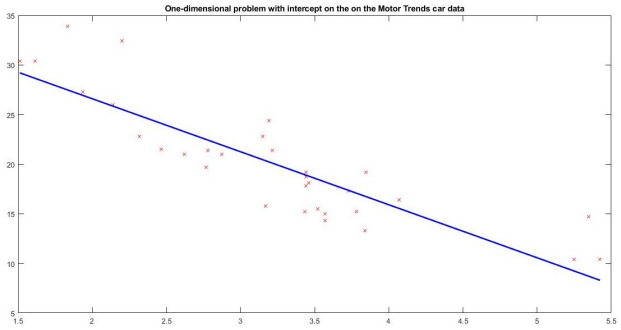

Fig. 3. 1-Dimensional linear regression with intercept

4) Multi-dimensional problem: In the last case the goal was to predict the mpg feature with all the remaining ones. As for the other instances it was necessary to implement the equation relative to the regression problem at hand. The MATLAB function implements equation 2.11 and then computes the estimated target value using equation 2.10. It then displays a table with the dataset values and the inferred ones, as shown in figure 4 below.

	disp	hp	weight	dataset mpg	Predicted mpg
1	160	110	2.6200	21	17.7300
2	160	110	2.8750	21	20.8063
3	108	93	2.3200	22.8000	19.4427
4	258	110	3.2150	21.4000	13.4556
5	360	175	3.4400	18.7000	7.0986
6	225	105	3.4600	18.1000	20.0486
7	360	245	3.5700	14.3000	11.7344
8	146.7000	62	3.1900	24.4000	24.0573
9	140.8000	95	3.1500	22.8000	25.7103
10	167.6000	123	3.4400	19.2000	27.3040
11	167.6000	123	3.4400	17.8000	27.3040
12	275.8000	180	4.0700	16.4000	24.7577
13	275.8000	180	3.7300	17.3000	20.6560
14	275.8000	180	3.7800	15.2000	21.2592
15	472	205	5.2500	10.4000	17.1604
16	460	215	5.4240	10.4000	21.1001
17	440	230	5.3450	14.7000	23.1416
18	78.7000	66	2.2000	32.4000	20.2359
19	75.7000	52	1.6150	30.4000	12.9155
20	71.1000	65	1.8350	33.9000	16.6769
21	120.1000	97	2.4650	21.5000	19.9532
22	318	150	3.5200	15.5000	11.8763

Fig. 4. Multi-dimensional linear regression

C. Task 3 - Test the implemented regression model

As anticipated in the introduction chapter, this task consisted in computing the objective function on different dataset split. In particular it was required to run task 1,3 and 4 with as training set the 10% of the dataset, compute the MSE on that data and then apply the obtained model on the remaining 90% of the dataset and compute the relative MSE. It was then decided to add a test with a different train-test split such as 40% training 60% testing to compare the results. In both the above cases the MSE was computed by averaging over one hundred thousand iterations to obtain a reliable value.
Below it can be found the two set of tables representing the testing done with the 10/90 train-test split (figure 5) and the 40/60 train-test split (figure 6).

1-Dimensional case without offset

	Dataset	Percentage	MSE
1	Train set	9.8881	8.5612e-05
2	Test set	89.9254	9.2872e-05

1-Dimensional case with offset

	Dataset	Percentage	MSE
1	Train set	12.5000	4.0376
2	Test set	87.5000	40.0476

Multi-dimensional case

	Dataset	Percentage	MSE
1	Train set	12.5000	21.6382
2	Test set	87.5000	5.5824e+05

Fig. 5. MSE of 10/90 train-test split

1-Dimensional case without offset

	Dataset	Percentage	MSE
1	Train set	39.9254	8.8238e-05
2	Test set	59.8881	9.0115e-05

1-Dimensional case with offset

	Dataset	Percentage	MSE
1	Train set	37.5000	7.4574
2	Test set	59.3750	12.1785

Multi-dimensional case

	Dataset	Percentage	MSE
1	Train set	37.5000	61.6721
2	Test set	59.3750	112.1819

Fig. 6. MSE of 40/60 train-test split

To fulfill all the above it was found necessary to develop a dedicated function implementing equation 2.2, to compute the MSE, for each different type of Linear Regression to test.
For each case to be tested, the relative regression and MSE-computing function were passed as parameters to a general function, along with the number of iteration and the percentage with which to split the dataset in train and test set. This function would simply compute the linear model by exploiting the relative regression function and compute the MSE with the corresponding MSE-computing function on both the train and test set, before displaying a table with the obtained results.

IV. CONCLUSIONS

Linear regression can be seen as an example of a simple learning problem. Thanks to its properties it be forecast one continuous variable by using one or more observations from related variables. Its limits are as clear as its potentials: the simplicity of its algorithm and implementation trade off with a poor performance in the case of a non linear relationship between the examinated variables.
The results clearly show that when the train set is made of only about the 10% of the whole dataset, the model overfits on those data and the MSE obtained on the rest of the dataset is rather higher with respect to the one obtained on the train set. Accordingly to this statement, when the train set is made of around of the 40% of the dataset, the gap between the two MSE reduces of an interesting amount because the model, thanks to the bigger quantity of observation available, manages to generalize more.