Machine Learning Assignment n.2

LINEAR REGRESSION

November 5, 2020

Chiara Terrile ID: 4337786
Cristina Naso Rappis ID: 4378585
University of Genoa

Contents

Chapter 1

Introduction

1.1 Defining the objective

The aim of this assignment is to compute the linear regression using two different data set. We manipulated the data in order to be able to apply both the one-dimensional and the multidimensional problem.
We calculated the corresponding models and plotted the results.
In the second part of the Lab we computed the objective(mean square error) under some conditions and reported the results in a table.

1.2 Linear regression

The term regression means approximating a functional dependency based on measured data. Our dataset is divided into observations and a target.
We want to find a linear model $y(x)$ that predicts t given x, where $y(x) = wx$. In particular we want $y(x)$ to be similar to $t(x)$ for each possible x. Therefore, our goal is to calculate the parameter w that makes our assumption true with a low average error.
To do this we consider the linear regression as an optimization problem. We choose as loss function the square error :

$$\lambda_{SE}(t, y) = (t - y)^2 \tag{1.1}$$

The generic goal is to minimize the mean value of the loss over the whole data set. In case of the square error loss, our objective function is:

$$J_{MSE} = \frac{1}{N} \sum_{l=1}^{N} (t_l - y_l)^2 \tag{1.2}$$

When building a model, the objecting is function of the parameters of the model (w) while the data are fixed. When using a model instead, the objective becomes function of the data since the parameters have already been fixed.
In this assignment we considered three different linear regression problems. In each of

them the goal was to minimize the objective function. To do this we considered its derivative and computed the parameter w such that:

$$\frac{d}{dw} J_{MSE} = 0 \tag{1.3}$$

which correspond to the value where the function has a minimum.
The first problem was the one-dimensional linear regression problem **without interception**. In this case the least squares solution is:

$$w = \frac{\sum_{l=1}^{N} x_l t_l}{\sum_{l=1}^{N} x_l^2} \tag{1.4}$$

The corresponding model is $y = wx$.
The second was the one-dimensional linear regression **with interception**, which is a more flexible model with respect to the previous one. In this case we computed 2 different parameters:

$$w_1 = \frac{\sum_{l=1}^{N} (x_l - \bar{x})(t_l - \bar{x})}{\sum_{l=1}^{N} (x_l - \bar{x})^2} \tag{1.5}$$

and

$$w_0 = \bar{t} - w_1 \bar{x} \tag{1.6}$$

with:

$\bar{x} = \frac{1}{N} \sum_{l=1}^{N} x_l$
and

$\bar{t} = \frac{1}{N} \sum_{l=1}^{N} t_l$
The corresponding model is $y = w_0 + x w_1$.

The last case was the **multi-dimensional** linear regression problem without interception. The term multi-dimensional comes from the fact that in the observations set we have **d** vectors instead of one. Our x will become a matrix X of dimensions Nxd:

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_N \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & ... & x_{1,d} \\ x_{2,1} & x_{2,2} & x_{2,3} & ... & x_{2,d} \\ x_{3,1} & x_{3,2} & x_{3,3} & ... & x_{3,d} \\ & & & . & \\ & & & . & \\ & & & . & \\ x_{N,1} & x_{N,2} & x_{N,3} & ... & x_{N,d} \end{pmatrix}$$

The vector **w** is now defined as :

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ . \\ . \\ . \\ w_N \end{pmatrix}$$

And similarly we define the target vector **t**.

$$\mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ . \\ . \\ . \\ t_N \end{pmatrix}$$

Given X and **t**, we are able to define **w** as :

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{t} = X^{\dagger} \mathbf{t} \tag{1.7}$$

Where $X^{\dagger}$ is the **Moore-Penrose pseudoinverse** of X.

The corresponding model is $y = wX$.

Chapter 2

Implementation

The **code** is divided in three parts:

- the *main* which contains the Task1 and calls the two functions for the Task2 and Task3
- *task2* which uses other functions
- *task3* which uses other functions

The functions used in ***task2*** are:

- *oneDimRegression* which simply comute the linear regression in the one dimensional case with, or without interception
- *randomGenerator* which given an interval generates a random subset of non-consequent data and computes its linear regression with the prevoius function
- *randomPlot* which, using the regression obtained from the randomGenerator, plots the result for 10 different random subset
- *multivarRegression* which performs a linear regression in case of multimensional set.

The functions used in ***task3*** are:

- *subsetCreator* which generates two subsets, one with the 5% of consequent data of a set, and one with the remaining 95%.
- *objective_5* which computes the mean square error for a subset in three cases: one dimensional regression with and without interception and multidimensional regression.
- *objective_95* which is similar to the previous one, but it computes the regression values in a different way

In the main file the *task3* is performed 10 times in order to collect more interesting data into a table (see Chapter 3).

2.1 Task1

Before using the two data set (*turkish-se* and *mtcarsdata-4features*) we have prepared them in order to be used on MATLAB.
For *turkish-se* we have simply removed the comma between the data in the first column and in the second one, obtaining a matrix 536x2.
For *mtcarsdata-4features* we have removed the first row, the first column and again the commas between each column, obtaining a matrix 32x4.
Both matrix are then loaded in the main file withe the MATLAB function *load*.

2.2 Task2

Task 2 is divided into 4 subsets. In each of them we used the 2 sets of data(*turkish* and *mtcars*) to solve linear regression problems.
The first subset is about solving the one-dimensional problem without intercept on the Turkish data, using the second column as target. To do this we used the function *oneDimRegression* which takes as input the Turkish set and a flag whose value determine if the regression will be with or without interception.
Since we want to solve the problem without interception, inside the function we only calculated the parameter w accordingly to the formula (1.4). After that we computed and plotted our model $y = wx$ (Figure 3.1).
In the second subset, we extracted different random subsets (10%) and repeated the steps in subset 1 for each of them. At the end we plotted the solutions obtained.
We used the function *randomGenerator* that generetes 10 different 10% subsets from the whole data and for each of them it calculates the solution of the linear regression problem (w) and the corresponding model using the function *oneDimRegression*.
The function *RandomPlot* will now plot the 10 different models in the same figure (Figure 3.2).
In order to have a more visible difference, we repeated the same steps extracting just the first and last 10% from the data set and plotting the two results. This is reasonable since the data are collected across time, therefore we are sure that data collected from the beginning and the end of the period will be as different as possible.
Here we can see the figures with the 2 results obtained (Figure 3.3 and Figure 3.4).
The third subset is about solving the one-dimensional problem with intercept on the *Motor Trends car* data.
We started cleaning the data, because we had to use only the first and fourth column(the first as target). We used the function *oneDimRegression* again, giving as input the *mtcars* data set cleaned.
As said before, the function calculates the linear regression, this time with interception. It will therefore returns the 2 parameters *w0* and *w1* calculated as above(see formulas 1.5 and 1.6).
Then we calculated and plotted the corresponding model $y = w_1x + w_0$ (see Figure 3.6).
In the last subset we solved the multi-dimensional problem on the complete *mtcars* data,

using the first column as target.
We used the function *multivarRegression* that takes as input the data set and, using the pseudoinverse, calculates and returns w according to the formula 1.7. We computed and plotted the corresponding model $y = wx$ (see Figure 3.7).

2.3 Task3

The *task3* function takes in input the two data set (*turkish* and *mtcars*) and i (number of iteration).
The aim of this part is to compute the objective (mean square error) in different situations.
First of all we used the function *subsetCreator* to generate the two subsets that will be used in this task. Everytime this function is called it generates a new random subset, so it's useful in case of multiple repetitions.
We applied this function to both sets (*turkish* and *mtcars*) and we get *turkish_subset_5*, *turkish_subset_95*, *mtcars_subset_5* and *mtcars_subset_95* which are the subsets of the 5% and 95% of the two sets.
Using the subsets of the 5% we have called the function *objective_5* which takes in input several values among which there are the two subsets.
This function as output gives the three J_{MSE} for the three main cases (1D regression with or without interception and multidimensional regression) and also returns the three values of the regression (w1, w2 and w3).
To compute J_{MSE} we have used the formula 1.2.
For the subsets of the 95% we have used another function, which is *objective_95.*
This function takes in input the two subsets and the regression values obtained from the prevoius funcion (w1, w2 and w3) because our aim is to compute the objetive of the same models on another subset.
So in this function we do not recompute the regression, but to obtain y we simply multiply our w (which is the input of the function) with the test set x (obtained from the subset of the 95%).
With these values we are able to compute again the mean square error in the three cases of interest using the formula 1.2.
Once we have the values of the objective in the three cases for both subsets, we put them into a vector J which is the output of our function *task3.*
In the *main* we repeat this task 10 times with random splits in order to get results as different as possible . Then we put them into a table to compare the error values.

Chapter 3

Result and Conclusions

When we run the code,for the **Task2**, we obtain the images below.

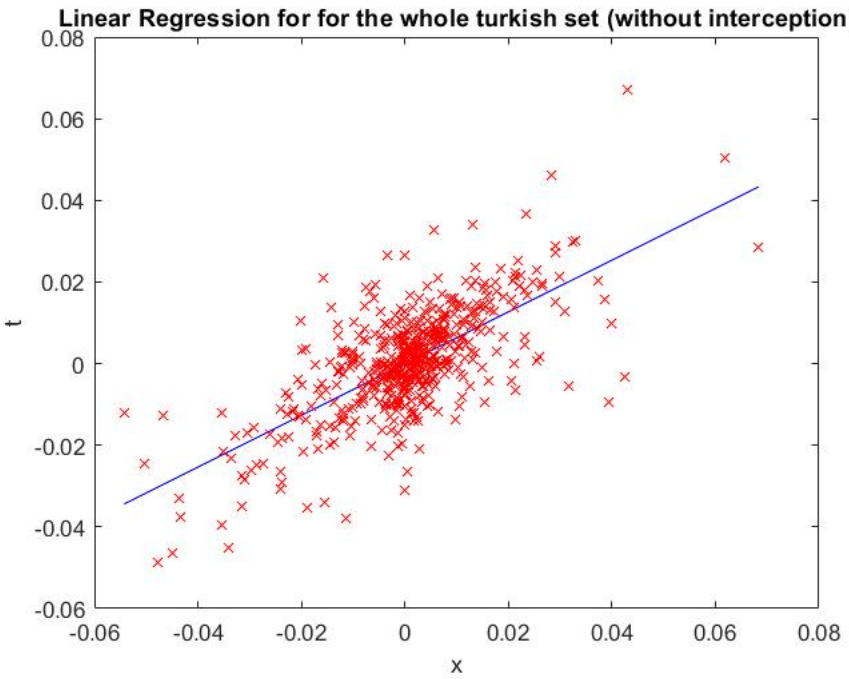

Figure 3.1: We plotted the result of the regression calculated on the whole data set (*turkish*). We can see that the blue line passes through 0 since we're not considering the interception. The blue line also approximates the distribution of the red crosses; where the blue line represents the predicted values *y(x)* and the red crosses are the target values *t(x)*.

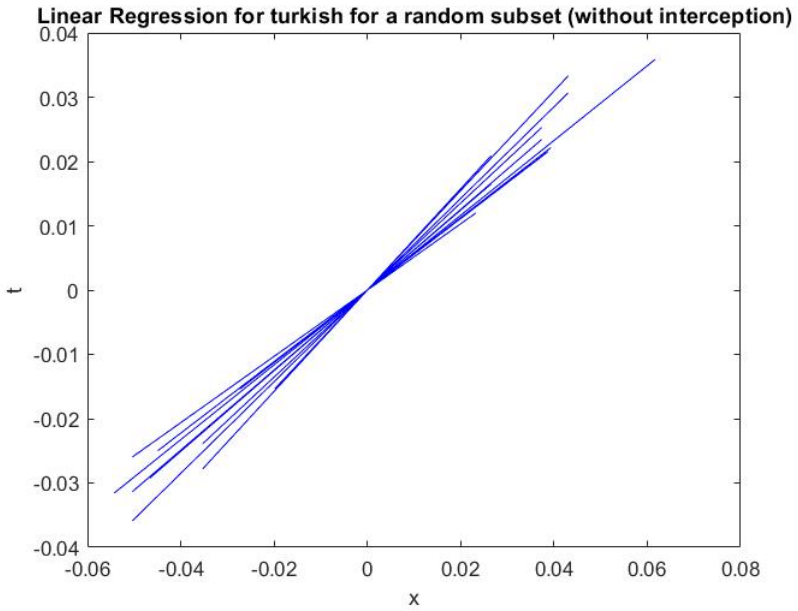

Figure 3.2: We have considered 10 random subset and plotted the corresponding predicted values y(always considering the case without interception).
We can see that some lines are almost overlapped. This is expected since we're considering 10 random subsets which may be similar among each other.

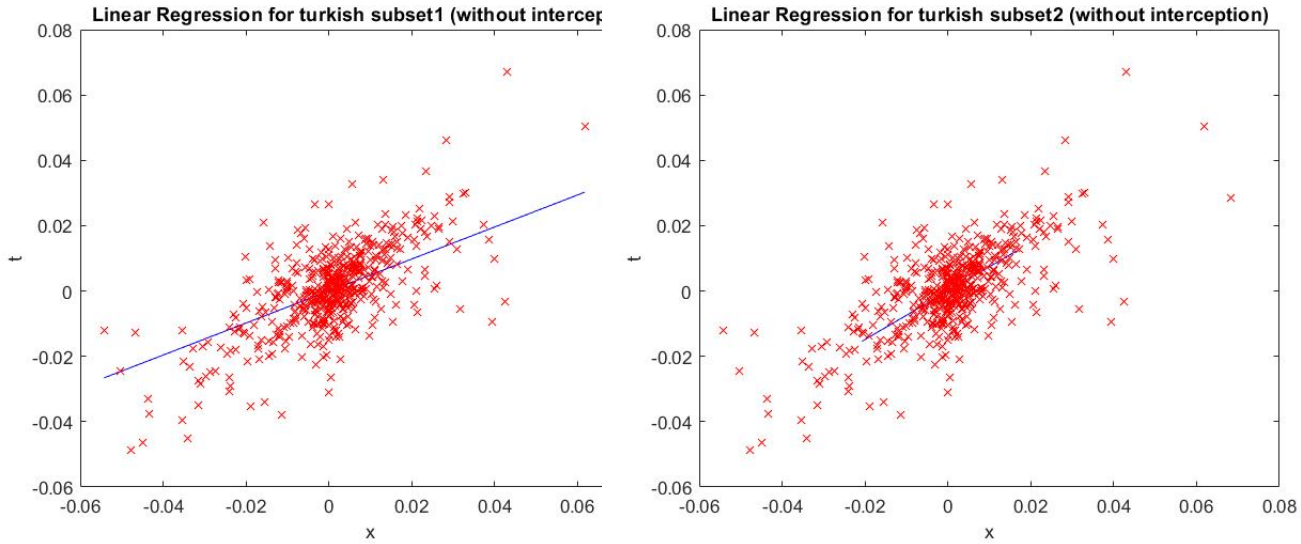

Figure 3.3: We extracted 2 subset which are not random, since we chose the first and the last subset of the dataset.

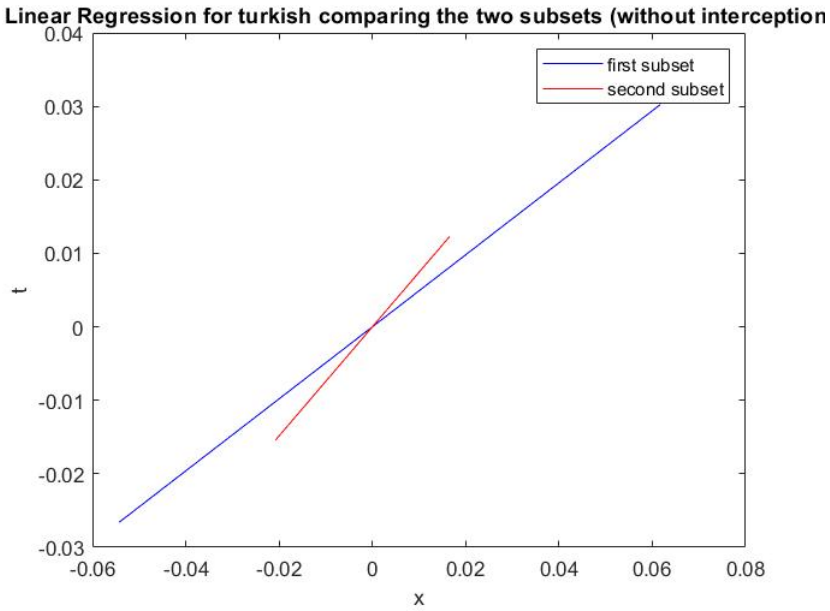

Figure 3.4: Here we can see the comparison of the two lines already plotted in figure 3.3. We can notice the difference between the two lines due to the choice of the two subsets.

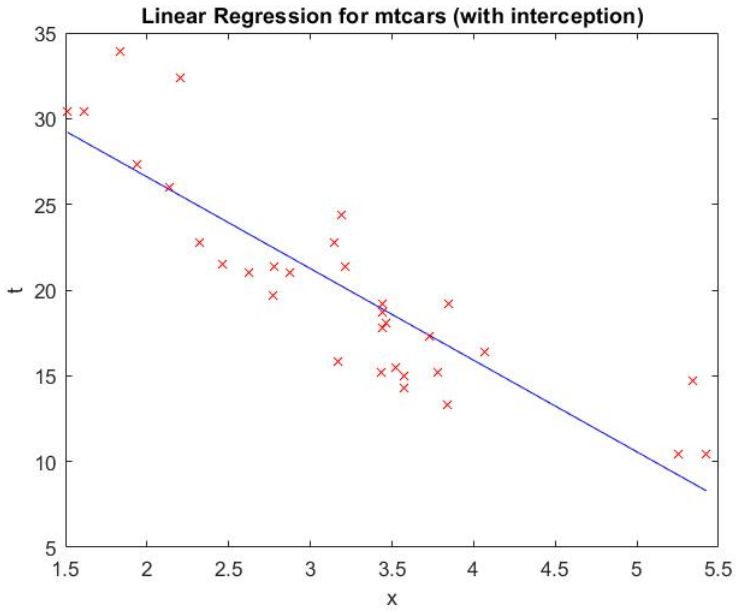

Figure 3.5: We plotted the result of the regression calculated on the data set (*mtcars*) considering just two columns (x and t) and so considering a one dimensional case. The blue line does not pass through the origin because of the offset w_0 used in the problem with interception.

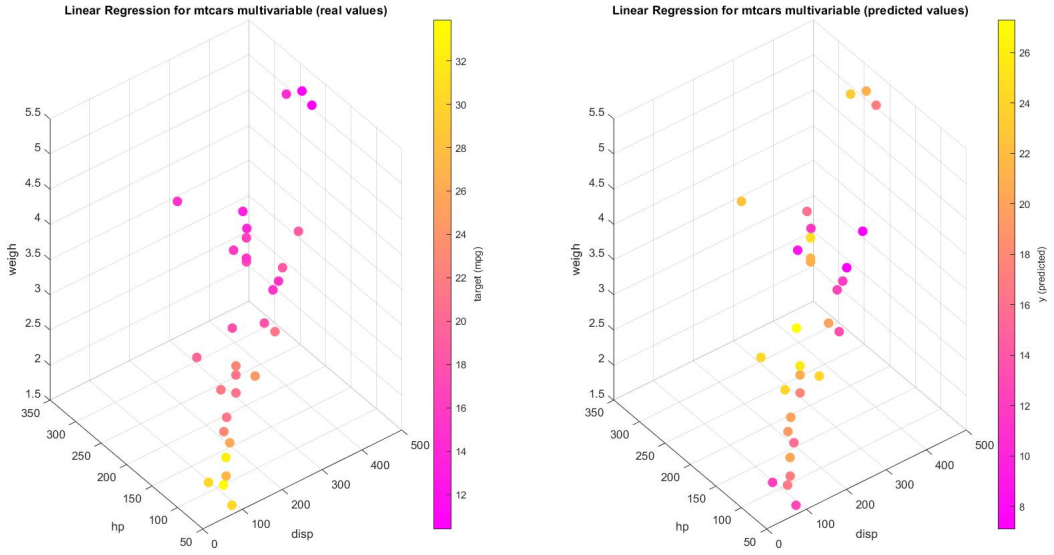

Figure 3.6: We plotted the result of the linear regression applied on the whole data set *mtcars*, so in the multidimensional case.
On the left it is shown the target value (mpg) in function of X which is composed of three features (disp,hp,weigh).
While on the right we plotted the Y value, which is the one predicted via regression, always in function of X.

Regarding the **Task3** what we obtain is a table containing the objective in the three significant cases and in the two subsets (5% and 95%) which is reported below.

J_{MSE}	rep. 1	rep. 2	rep. 3	rep. 4	rep. 5	rep. 6	rep. 7	rep. 8	rep. 9	rep. 10
J_{MSE} (at 5% 1D without interception)	5.8774 e-05	3.0393 e-05	2.2083 e-04	2.1496 e-05	1.9147 e-05	4.1185 e-05	1.9099 e-05	3.4856 e-05	1.7988 e-04	3.4913 e-05
J_{MSE} (at 5% 1D with interception)	0	1.2622 e-29	0	0	8.2042 e-29	3.1554 e-29	1.2622 e-29	0	8.6001 e-27	1.8302 e-28
J_{MSE} (at 5% multidimensional)	1.2622 e-29	1.8302 e-28	6.3109 e-30	5.3643 e-28	1.2369 e-27	1.1360 e-28	1.2622 e-29	2.9409 e-27	3.9443 e-29	7.8886 e -30
J_{MSE} (at 95% 1D without interception)	9.5100 e-05	1.1934 e-04	9.0260 e-05	9.2391 e-05	9.4895 e-05	9.2414 e-05	9.2889 e-05	1.1328 e-04	1.1336 e-04	1.0395 e-04
J_{MSE} (at 95% 1D with interception)	25.233	18.172	38.417	137.70	469.06	2.0745 e+03	52.558	44.799	1.9767 e+06	896.47
J_{MSE} (at 95% multidimensional)	641.17	448.18	510.40	291.35	1.0709 e+03	1.0149 e+03	351.70	1.7415 e+03	259.56	401.14

Table 3.1: In the first 3 lines of the table we reported the mean square error of the 5% of the whole data set. We can notice that the error is pretty small since we used the 5% of the set for the training and we tested the model using the same 5%.
On the other hand on the last 3 lines the error increases. This is expected since we used the 5% of the whole set for the training and we tested the model using the remaining 95%. The error increases also because our training set is very small(especially considering the mtcars, whose 5% only contains 2 observations).
We wouldn't get a high difference if our training set was bigger. In fact if we look at the results obtained using the turkish set (first and fourth row) we can notice that the error is not so different since the 5% of the data set contains 26 observations.