Linear Regression
Assignment 2 of the Machine Learning 1 course 2022/2023

Nicchiarelli Gabriele

DIBRIS - Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi
Università Degli Studi di Genova

1 Introduction

Linear regression is an algorithm used to visualize a relationship between two variables, a dependent variable, and an independent variable.
The independent variable stands by itself and isn't impacted by the other variable. While the value of the dependent variable will fluctuate according to the independent variable. So in a regression problem, we try to find the relationship between the independent and the dependent variable, in order to be able to approximate the target variable.
It is a *linear* problem so we use a first-order polynomial function to approximate the data. Indeed, for a one-dimensional problem, the approximation will be a line, for a two-dimensional problem, a plane, and so on.

2 Theory of Linear Regression

2.1 One-dimensional problem

The linear regression problem can be formalized in the following way:
define a linear function such that, given an observation (*x*), computes the target (*t*) that best approximates the real value (*y*).

$$t = wx, \quad t \approx y \tag{1}$$

Linear regression aims to compute the best value of the parameter *w* that better approximates each observation, to do so it must minimize the *mean error*.
We use a *square error* for *w* calculations because it grows more than linearly, providing heavier weights to larger errors, it is even and differentiable:

$$\frac{d}{dx}\lambda_{SE}\left(y,t\right) = 2\left(y-t\right) \tag{2}$$

The Mean Square error (MSE) can be computed as:

$$J_{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(t_i - y_i\right)^2 \tag{3}$$

Omitting the long computation, it is proven that the best value of *w* used for approximation is:

$$w = \frac{\sum_{l=1}^{N} x_l t_l}{\sum_{l=1}^{N} {x_l}^2} \tag{4}$$

2.2 Adding the intercept

It is possible to improve our regression model by adding the *intercept* factor, w_0, to equation 1:

$$t = w_1 x + w_0 \tag{5}$$

The two parameters, w_0 and w_1 can then be computed as follows:

$$w_1 = \frac{\sum_{l=1}^{N}\left(x_l - \bar{x}\right)\left(t_l - \bar{t}\right)}{\sum_{l=1}^{N}\left(x_l - \bar{x}\right)^2} \tag{6}$$

$$w_0 = \bar{t} - w_1\bar{x} \tag{7}$$

2.3 Multi-dimensional linear regression

Now that we have the basics, we can generalize to the multi-dimensional problem.
$\mathbf{X}$ now is a matrix of observations, and $\mathbf{w}$ is a vector of the approximation parameters:

$$\mathbf{X} = \begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \dots\\ \mathbf{x}_n\end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix}w_0\\ w_1\\ \dots\\ w_n\end{bmatrix} \tag{8}$$

We obtain the final output ($\mathbf{y}$) of the linear regression as a matrix product:

$$\mathbf{y} = \begin{bmatrix}1 & x_{1,1} & \dots & x_{1,n}\\ 1 & x_{2,1} & \dots & x_{2,n}\\ \dots & \dots & \dots & \dots\\ 1 & x_{n,1} & \dots & x_{n,n}\end{bmatrix}\begin{bmatrix}w_0\\ w_1\\ \dots\\ w_n\end{bmatrix} = \mathbf{X}\mathbf{w} \tag{9}$$

Note that in the $\mathbf{X}$ matrix we have the first column of ones because we incorporated the term w_0 in it.
As for the one-dimensional case, also here we have to minimize the MSE. However, since we are dealing with matrices, now we must set $\nabla J_{MSE} = 0$ (i.e. all the partial derivatives to 0).
Finally we can compute $\mathbf{w}$:

$$\mathbf{w} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{t} = \mathbf{X}^{\dagger}\mathbf{t} \tag{10}$$

Where $\mathbf{X}^{\dagger}$ is the *Moore-Penrose pseudoinverse* of $\mathbf{X}$. Note also that when computing the pseudoinverse a small but notable error can be introduced.

3 Assignment

The assignment is composed of three tasks:

1. Get data;
2. Fit a linear regression model;
3. Test regression model.

3.1 Get data

For this assignment we will use two data sets, provided in a *csv* format:

- the Turkish stock exchange data (*turkish-se-SP500vsMSCI*);
- the 1974 Motor Trend Car Road Tests (*mtcarsdata-4features*).

The first one has 536 observations and we will use it for one-dimensional problems, while the second has 32 observations and has 4 variables, so we can solve multi-variable problems.
First thing first, we import the data using the *read_csv()* Python function from the *Pandas library*, in order convert them into dataframes.
We don't have to do particular data pre-processing since the data sets are already ready to use.

3.2 Fit a linear regression model

Once we imported our data we have to compute the linear regression parameters in four different cases:

1. One-dimensional problem without intercept on the Turkish stock exchange data;
2. Compare graphically the solution obtained on different random subsets (10%) of the whole data set;
3. One-dimensional problem with intercept on the Motor Trends car data, using columns mpg and weight;
4. Multi-dimensional problem on the Motor Trends car data, using all four columns (predict mpg with the other three columns).

Before starting we prepare in advance two functions to print the linear regression graphs, one for the one-dimensional problem and one for the multi-dimensional problem. The functions use the *matplotlib* Python's library.
Take notice that in the code *y* is used to indicate the target *t*.

3.2.1 1

In order to calculate the linear regression (without intercept) we create a function, *linear_regression_one_dim_no_intercept()*, which takes as arguments a dataframe, the column of the observation variable, and the column of the target variable. The function calculates the parameter *w*, which is called *slope* in the code, then returns a lambda function that corresponds to the linear regression function. The result is shown in figure 1.

3.2.2 2

Now we create 4 random subsets of 10% of the data set. Then we pass directly the function *linear_regression_one_dim_no_intercept()* as an argument in a function that creates a plot with all the 4 linear regression lines and their relative subsets used for computing the lines.
The result can be analyzed in figure 2.

3.2.3 3

For this task, we create another function, called *linear_regression_one_dim()*, that calculates the linear regression line taking into account the intercept. We pass the *weight* column as an independent variable and the *mpg* as a target vector. The calculations are the ones shown in subsection 2.2. The plot of this task is in figure 3.

3.2.4 4

In this last step, we create a function, called *linear_regression_multidim()*, in order to predict *mpg* using the other three columns on the Motor Trends car data set. This time we pass a matrix of observations as an argument instead of a vector.
According to the theory stated in subsection 2.3, we compute the Moore-Penrose pseudoinverse, then use it to compute the *w* factor, and finally return a lambda for the linear regression function, similarly to the previous functions. Note that in *Python* we use the symbol @ to perform matrix multiplications.
Since it's a multi-dimensional problem we don't plot anything but instead, we print the predicted output on the console (figure 4).

3.3 Test regression model

The last task requires running again points 1, 3, and 4 from the previous task using 5% of the data, computing the MSE on the training data, and repeating the process for the remaining 95% of the data. Then repeat for different training-test random splits.
Here the only new thing is that I introduced two functions, one to compute the MSE in a one-dimensional problem and one to calculate the MSE for a multi-dimensional problem.
Lastly, to compare the performances for different training-test sets, I plotted a histogram for the MSE using the function *mse_histograms()*.
Figures 5, 6, 7 show the linear regression problem for the different cases with the 5%-95% split. While 8 show the respective Mean Square Errors.
Note that for the second data set, if we take only 5% of the data, it means we have only 2 observation points, so the regression line will fit the data perfectly. We can also prove it by observing that the MSE (third line of figure 8) is really small.
Finally, we compute the MSE for 100 random splits and plot the histograms (figure 9).

4 Results

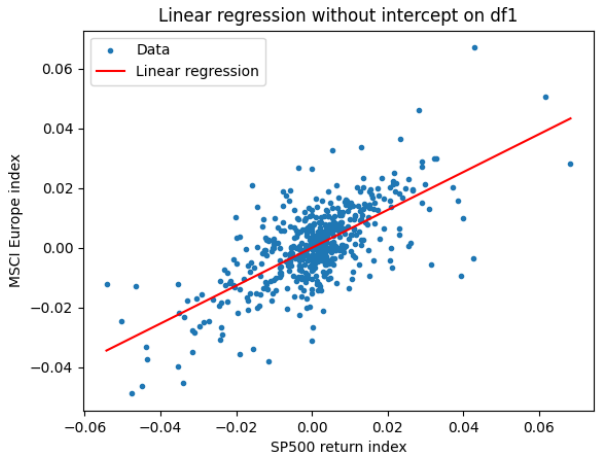

Figure 1. One-dim linear regression without intercept.

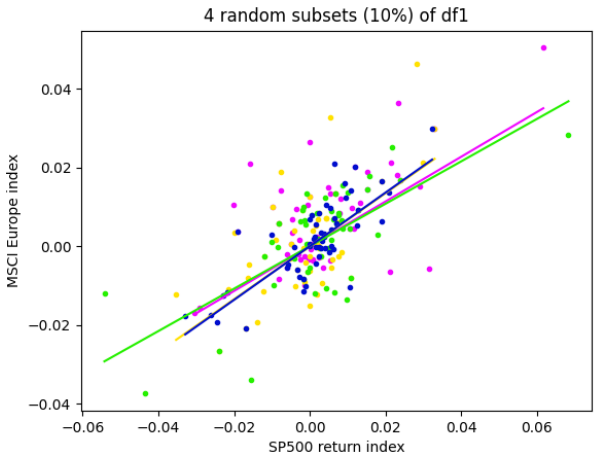

Figure 2. One-dim linear regression without intercept on 4 random subsets (10% of data).

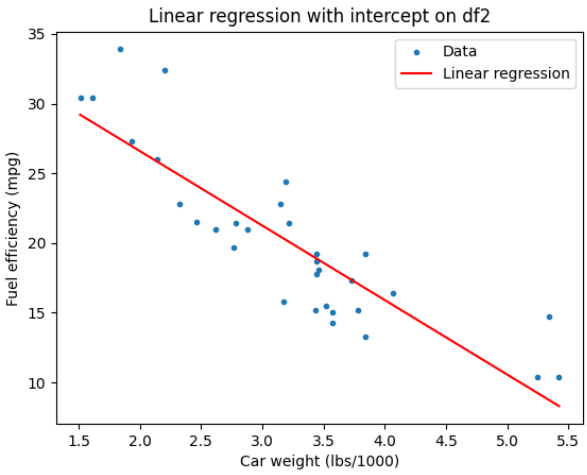

Figure 3. One-dimensional problem with intercept.

0	17.730038
1	20.806333
2	19.442687
3	13.455631
4	7.098581
5	20.048607
6	11.734448
7	24.057253
8	25.710315
9	27.303982
10	27.303982
11	24.757694
12	20.655966
13	21.259162
14	17.160427
15	21.100108
16	23.141624
17	20.235860
18	12.915548
19	16.676860
20	19.953218
21	11.876321
22	12.486951
23	16.160318
24	7.310002
25	17.003866
26	15.746142
27	12.091074
28	8.793263
29	24.140994
30	22.573283
31	24.174040

Figure 4. mpg prediction: multi-dim linear regression.

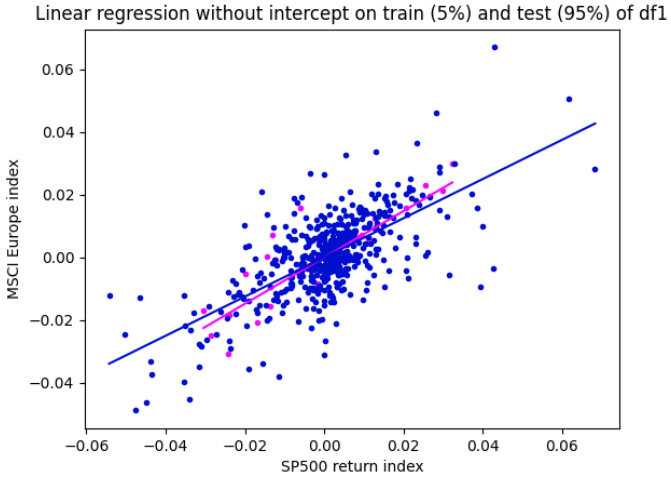

Figure 5. One-dim linear regression without intercept 5%-95% split (first data set).

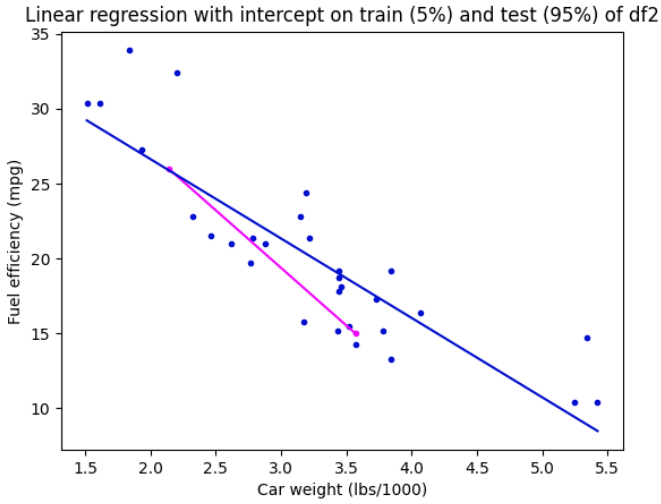

Figure 6. One-dim linear regression without intercept 5%-95% split (second data set).

```
mpg predicted train:
30     2.957536
26    32.575158
dtype: float64
mpg predicted test:
0     17.844098
1     20.682156
2     19.620462
3     12.369320
4      7.266853
5     18.786144
6     14.081122
7     22.149202
8     24.962653
9     27.029079
10    27.029079
11    25.055381
12    21.271305
13    21.827787
14    15.886775
15    20.071360
16    22.811037
17    19.831346
18    12.617339
19    16.630480
20    20.047370
21    11.424681
22    12.206792
23    18.320503
24     6.836841
25    16.844961
27    13.764741
28    12.197078
29    26.349141
31    24.362240
dtype: float64
```

Figure 7. mpg prediction: multi-dim linear regression (5%-95% split).

```
MSE for the train set of df1: 5.3671818457983646e-05
MSE for the test set of df1: 9.047280559879439e-05
MSE for the train set of df2: 1.262177448353619e-29
MSE for the test set of df2: 8.92203986865968
MSE for the train set of df2 (multidimensional): 94.12682047794132
MSE for the test set of df2 (multidimensional): 1071.6917903646338
```

Figure 8. mpg prediction: multi-dim linear regression (5%-95% split).

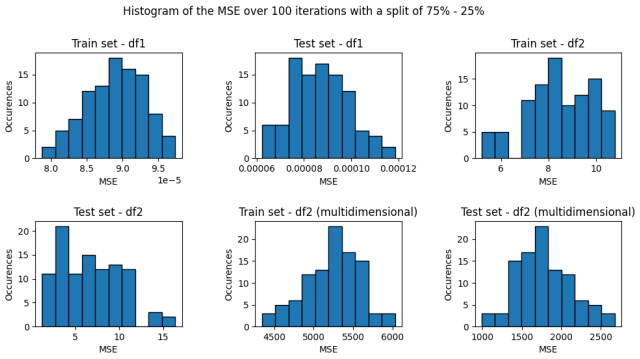

Figure 9. Histogram of MSEs (100 tests).