LINEAR REGRESSION

Machine Learning 2018/19: Assignment 1

Torielli Davide Fusaro Fabio
EMARO - European Master on Advanced Robotics
DIBRIS - Dip. Informatica, Bioingegneria, Robotica, Ing. Dei Sistemi
Università Degli Studi Di Genova, October 17, 2018

I. INTRODUCTION

LINEAR REGRESSION means approximating a functional dependency based on measured data. With terms "linear" we indicates that the approximation will be a linear one (a line in the one-dimensional case).
In particular, given a certain dataset, we try to understand how the *target* **t** is correlated with the *observations* **x**. With this prevision, we can estimate the target given a certain observations. The problem we have faced was to implement the algorithm on Matlab and then analyze the results.

II. THE THEORY

A. One-dimensional input without intercept

We want to build a linear model $y(x)$ that predicts t given x, so:

$$t \approx y \quad \textit{where} \quad y = wx$$

Where t is the *target*, y the prediction, x the observations that it is a vector because we are in the one-dimensional case.
We want $y(x)$ to be *similar* to $t(x)$ for any x.
For example, given a certain observation x_1 and a certain related target t_1 we found the parameter w as:

$$w = \frac{t_1}{x_1}$$

But given another couple x_2, y_2 the resulting w will be different: it is impossible to find a perfect value for w.
Thus, the goal is to find a correct parameter w that has the most suitable value based on a certain optimization.
We need to quantify how wrong is each estimate based on certain *measure* and make this measure as small as possible *on average*.
The square error $(t-y)^2$ is a good choice because is differentiable and even. It also give more weight to large error respect to smaller ones. Having N couples t_i, y_i, we have to minimize the mean value of the loss over the whole dataset:

$$\frac{d}{dw} J_{MSE} = 0 \quad \textit{where} \quad J_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2$$

After some passages, we arrive to this final solution:

$$w = \frac{\sum_{i=1}^{N} x_i t_i}{\sum_{i=1}^{N} x_i^2}$$

So, with a new given observation x, we will be able to predict the target as:

$$y = wx$$

B. One-dimensional input with intercept

With the previous model $y = wx$ we build a line that intercepts the origin.
We can improve the model adding one parameter w_0 to make the line intercepting another (more suitable) point on y axis:

$$y = w_1 x + w_0$$

The solution in this case can be found by *centering* around the means $\bar{x}$ and $\bar{t}$ of x and t:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_1 \quad \bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_1$$

The wanted parameters w_0 (*intercept*) and w_1 (*slope*) are:

$$w_1 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(t_i - \bar{t})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \qquad w_0 = \bar{t} - w_1 \bar{x}$$

C. Multi-dimensional inputs

The formulas can be generalized for the multi-dimensional case. In this case each observation $\boldsymbol{x}_i$ is now a vector of *d* elements; thus, we have *d* parameters w_j with $j = 1 \dots d$ plus a parameter w_0 for the intercept :

$$X = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_N \end{pmatrix} \qquad \boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{pmatrix}$$

The output of the model for a single observation will be:

$$y_i = w_0 + x_{i,1} w_1 + x_{i,2} w_2 + \dots x_{i,d} w_d$$

Combining all y_i we obtain:

$$\boldsymbol{y} = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N,1} & \dots & x_{N,d} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{pmatrix} = \mathcal{X}\boldsymbol{w}$$

Note the difference between the $N \times d$ matrix X and the $N \times (d+1)$ matrix $\mathcal{X}$.
As in the one-dimensional case, we have to minimize the mean square error objective J_{MSE}; but now we have $d+1$ parameters so we must set the *gradient* of J_{MSE} equal to zero. After some passages we obtain:

$$\nabla J_{MSE} \triangleq \frac{\partial}{\partial \boldsymbol{w}} J_{MSE} = \mathcal{X}^T \mathcal{X} \boldsymbol{w} - \mathcal{X}^T \boldsymbol{t} = \boldsymbol{0}$$

and so we obtain the value for $\boldsymbol{w}$:

$$\boldsymbol{w} = (\mathcal{X}^T \mathcal{X})^{-1} \mathcal{X}^T \boldsymbol{t} = \mathcal{X}^{\dagger} \boldsymbol{t}$$

where $\mathcal{X}^{\dagger}$ is the *Moore-Penrose pseudoinverse* of $\mathcal{X}$.
The computation of the pseudoinverse is not always numerical reliable. It is almost impossible that $\mathcal{X}^T\mathcal{X}$ is not invertible because it would mean that we have two *identical* observations on all columns (noise is always present).
But we can also have problems if the variables are *correlated* (similar to each one): this would mean that $\mathcal{X}^T\mathcal{X}$ has high *condition number* and the pseudoinverse will be numerical unstable. In this case, there are solutions based on iterative computation by successive approximations.

III. Experimental results

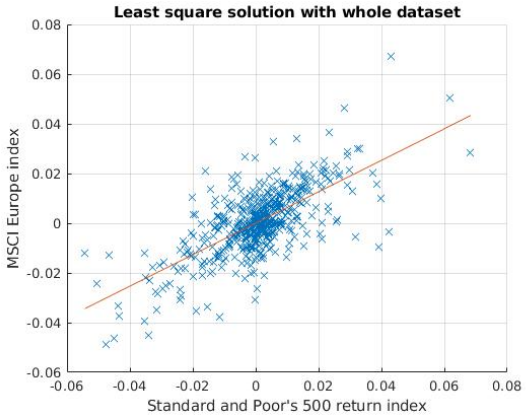

Fig. 1: *One-dimensional problem without intercept on the Turkish stock exchange data*

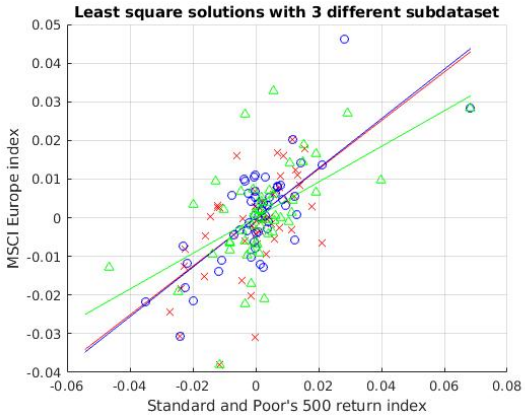

Fig. 2: *Three least square solutions obtained on different random subsets (10%) of the data set*

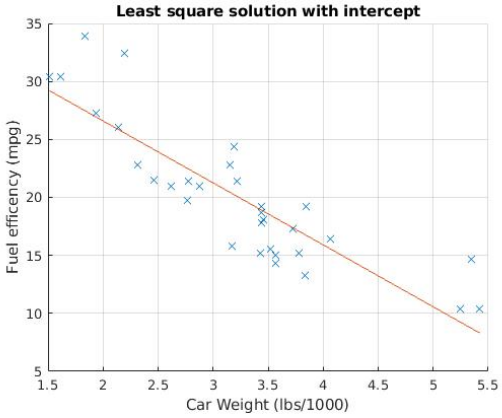

Fig. 3: *One-dimensional problem with intercept on the Motor Trends car data (using mpg and weight)*

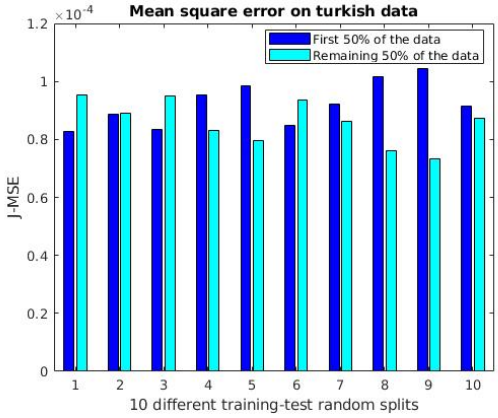

Fig. 4: *Mean square error on the Turkish data for 10 different training-test random splits*

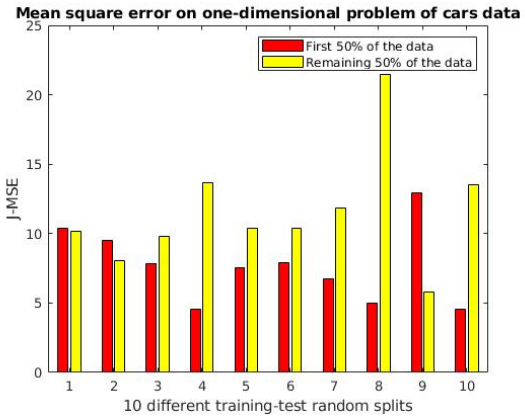

Fig. 5: *Mean square error on the Motor Trends car data (using mpg and weight) for 10 different training-test random splits*

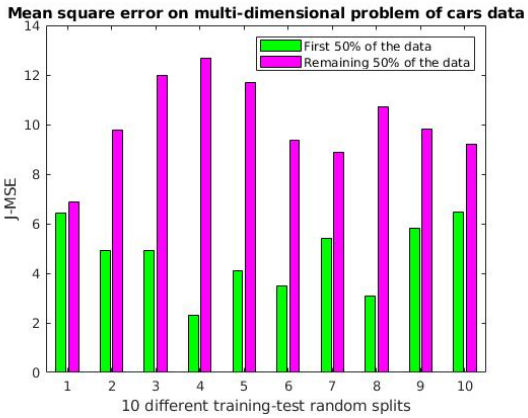

Fig. 6: *Mean square error on the Motor Trends car data for 10 different training-test random splits*

The experiments consist in the analysis of the linear regression on two different dataset: *Turkish stock exchange data* and *1974 Motor Trends car data*.

This was implemented with three tasks:

- **Task 1**: Get data.
- **Task 2**: Fit a linear regression model.
- **Task 3**: Test regression model.

A. *Task 2*

The Fig. 1 shows the result (the red line $y = wx$) of the one-dimensional problem without intercept on the whole Turkish dataset as training set.

In Fig. 2 instead, the results shown are the ones consider only 10% of the whole dataset. The same color refers to same experiment. Seen that each one of the three experiments takes different random subset, the results vary and are, in general, worse than in the previous figure, due to the fact that we have less information. However, it can be note that the three lines pass all through the origin because there is no intercept parameter.

The Fig. 3 shows the one-dimensional problem with intercept on the cars data. In this case we only consider as known the weight that it is used to predict the the full efficiency (mpg). It is noticeable that the line now does not pass anymore on the origin.

The fourth result is not shown in the plots because it is a multi-dimensional problem on the complete MTcars data. In this one, we have to predict the full efficiency (mpg) knowing displacement (disp), horse-power (hp) and weight. So the input has 3 dimensions; thus we have to find 3+1 parameters (+1 for the intercept parameter w_0).
The obtained $\boldsymbol{w}$ is:

$$\begin{aligned} w_0 &= 37.1055 \\ w_1 &= -0.0009 \\ w_2 &= -0.312 \\ w_3 &= -3.8009 \end{aligned}$$

B. *Task 3*

This task consists to take the 50% of the data as training set, to make a model (compute w) and then compute the mean square error on the remaining 50% of the data. AT the end we compare the mean square errors between the two halves. This was done for ten different training-test random splits. The last three plots show the comparison of the mean square errors J_{MSE}.

The Fig. 4 shows the mean square error on the Turkish data; as we can see the J_{MSE} value for the two halves of the dataset is similar in all the ten experiments. Moreover the error of the first half is not always lower than one in the second half. This is due to the fact that there are a lot of observations (approximately 500).

In Fig. 5 and in Fig. 6 the Motor Trend car data is used. As we can notice in the Fig. 5 the mean square error is less similar between the two halves with respect to the Fig. 4 because in the MTcars data there are much less observations (approximately 30) than in the Turkish data.
In the multi-dimensional problem (Fig. 6) J_{MSE} of the training dataset (first 50%) is always lower than the one of the other subset; this because the inputs are more than in the one-dimensional problem and so the observations are more spread (it is more likely that, having more dimensions, the values of the second half are farther from the estimated model).