Linear regression using robust functions

Lluís Garrido – lluis.garrido@ub.edu

October 2018

Abstract

This laboratory is focused on the principles of data regression. We will focus on some of the well known methods and on the importance of the error function used to perform the regression. Notice that this lab is still focused on unconstrained optimization.

1 Introduction

Linear regression is an approach to model the relationship between two continuous, quantitative, variables. One variable, denoted x, is regarded as the predictor or independent variable. The other variable, denoted as y, is regarded as the response or dependent variable.

Take a look at figure 1. We may see a set of data points in blue. In this example the objective is to approximate the data points using a simple linear regression, i.e. we only have one predictor variable.

What is the best fitting line? Let $\mathbf{x}_i = \{x_i, y_i\}$ with $i = 1 \dots m$ be the data points. There are different approaches that one may take to compute the best fitting line, see figure 2. The figure shows two different methods: the standard fitting using vertical offsets, and the fitting using perpendicular offsets. Let us briefly describe both:

1.1 Vertical offsets

We begin with the standard approach. Let $\hat{y}_i$ be the fitted value for element i. The equation for the best fitting line is $\hat{y}_i = w_0 x_i + w_1$, where $w_0 \in R$ and $w_1 \in R$ are the parameters to be estimated. The parameters w_0 and w_1 may be computed in such a way that the prediction error (or residual error), $e_i = |\hat{y}_i - y_i|$, is minimized. A line that fits the data "best" will be one for which the m prediction errors are as small as possible in some overall sense. A classical way to proceed is to use the least squares error function, that is,

$$Q = \frac{1}{2}\sum_{i=1}^{m} e_i^2 = \frac{1}{2}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 \tag{1}$$

One may use the gradient (or Newton) descent method, for instance, to obtain the optimal values for w_0 and w_1. Let us see how to proceed in this case

$$Q = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(w_0 x_i + w_1 - y_i)^2$$

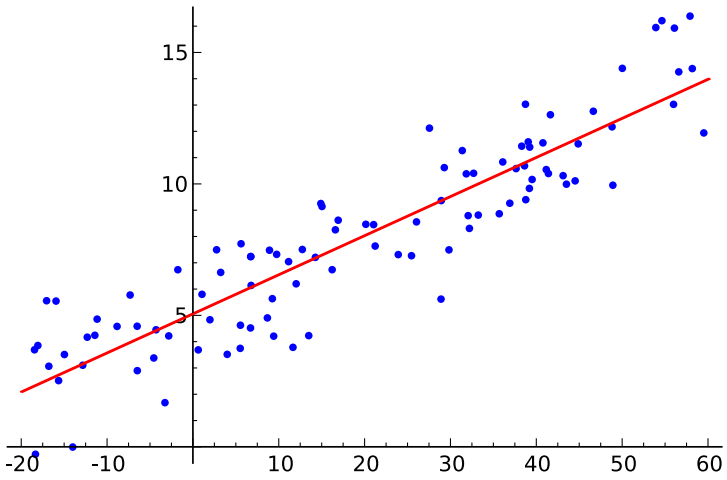

Figure 1: Simple linear regression example. Data points are marked in blue and the objective is to model the points with a linear model. Obtained from https://en.wikipedia.org/wiki/Linear_regression.

We want to obtain the values of w_0 and w_1 that minimize Q. For this purpose let us compute the corresponding gradient

$$\begin{aligned} \frac{\partial Q}{\partial w_0} &= \sum_{i=1}^{m} (w_0 x_i + w_1 - y_i)\, x_i \\ \frac{\partial Q}{\partial w_1} &= \sum_{i=1}^{m} (w_0 x_i + w_1 - y_i) \end{aligned}$$

The parameters to be estimated are $\mathbf{w} = (w_0, w_1)$. The gradient descent is

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha^k \nabla Q(\mathbf{w}) \qquad \nabla Q(\mathbf{w}) = \left(\frac{\partial Q}{\partial w_0}, \frac{\partial Q}{\partial w_1} \right)$$

1.2 Perpendicular offsets

We may also use the perpendicular offsets, see figure 2. The equation $\mathbf{w}^T \hat{\mathbf{x}} + b = 0$ corresponds to the implicit equation of a line, where $\hat{\mathbf{x}}$ represents the points of the fitting line, and $b \in R$ and $\mathbf{w} = (w_0, w_1)$ represent the associated parameters. For a given data point $\mathbf{x}_i = \{x_i, y_i\}$, the value $\left| \mathbf{w}^T \mathbf{x}_i + b \right|$ is the perpendicular distance of $\mathbf{x}_i$ to the line $\mathbf{w}^T \hat{\mathbf{x}} + b = 0$. Using least squares, we may proceed in a similar way as in the previous case to obtain the parameters associated to the fitting line.

$$Q = \frac{1}{2} \sum_{i=1}^{m} e_i^2 = \frac{1}{2} \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}_i + b)^2 \tag{2}$$

The gradient of the previous equation with respect parameters w_0, w_1, b is

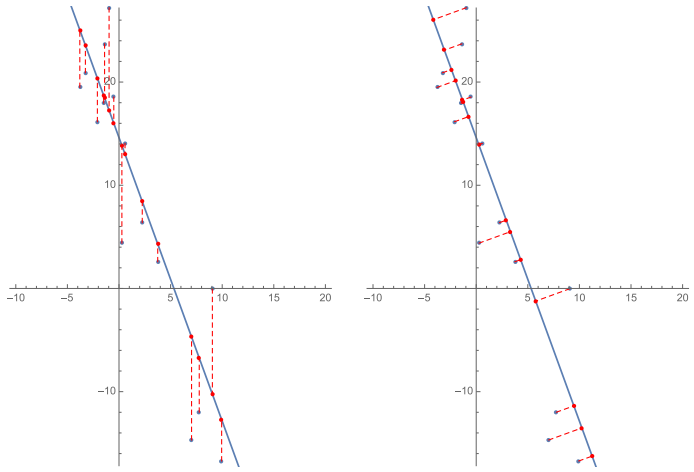

Figure 2: Left: fitting using vertical offsets, Right: fitting using perpendicular offsets. Obtained from http://toddreed.name/articles/line-fitting/.

$$\begin{aligned}
\frac{\partial Q}{\partial w_0} &= \sum_{i=1}^{m} (w_0 x_i + w_1 y_i + b)\, x_i \\
\frac{\partial Q}{\partial w_1} &= \sum_{i=1}^{m} (w_0 x_i + w_1 y_i + b)\, y_i \\
\frac{\partial Q}{\partial b} &= \sum_{i=1}^{m} (w_0 x_i + w_1 y_i + b)
\end{aligned}$$

Let $\mathbf{p} = (w_0, w_1, b)$ be the parameters to be estimated. The gradient descent is

$$\mathbf{p}^{k+1} = \mathbf{p}^k - \alpha^k \nabla Q(\mathbf{p}) \qquad \nabla Q(\mathbf{p}) = \left(\frac{\partial Q}{\partial w_0}, \frac{\partial Q}{\partial w_1}, \frac{\partial Q}{\partial b} \right)$$

Exercises

We propose you to perform the next experiments:

1. Implement both methods, the standard and the perpendicular one using gradient descent with backtracking (or a small constant α value)

2. We begin with an experiment that allows to see the difference between both models we have developed. In particular, we generate a random set of values

```
m = [0.,0.]
angle = 45*math.pi/180
rot = np.array([[math.cos(angle), -math.sin(angle)], [math.sin(angle),
math.cos(angle)]])
lamb = np.array([[100,0],[0,1]])
s = np.matmul(rot, np.matmul(lamb, rot.transpose()))
c = np.random.multivariate_normal(m,s,100)
```

The `angle` value as well as the `lamb` matrix allows us to control the shape of the random values that are generated. You may try with and angle of 0, 45 and 90 degrees.

Compute the parameters associated to both models we have presented and draw the lines that have been obtained. Observe the numerical instability that may be obtained for the perpendicular method, i.e. you'll see that values w_0, w_1 and b get very small. Why do they get so small? Do you think that there is a way to deal with this numerical issue?

You are now proposed to use the Anscombe's dataset (https://en.wikipedia.org/wiki/Anscombe's_quartet) to perform the tests. It is a relatively simple dataset since the number of elements of each dataset is rather low.

1. Graph each of the datasets and see how the samples are distributed. Making a plot of the dataset is commonly an important step to visually analyze the samples. You are recommended to perform it previous to any automatic analysis.

2. For each of the dataset, compute the parameters using both models, the model associated to the vertical offsets and the model of the perpendicular offsets. As before, use the gradient descent using the backtracking algorithm. You may also use a rather low constant step value to avoid that the iterations diverge.

2 Robust functions

The least squares method is known to be sensitive to outliers, i.e. samples that do not fit the model (since they may be noisy samples, for instance). The reason is due to the fact that the least squares method minimizes the squared error which may be very large for an outlier. The outliers may thus have a large influence in the numerical values of the parameters to be estimated.

We need to modify equation (1) so as to make it more robust to outliers. Without entering into exhaustive details, a way to proceed is to minimize

$$\sum_{i=1}^{m} \rho(e_i)$$

where $\rho(u)$ is a robust error function. For the least squared method $\rho(u) = \frac{1}{2}u^2$, but one may take other functions such as the Cauchy function, which is defined to be[1]

$$\rho(u) = \frac{c^2}{2} \log\left[1 + \left(\frac{u}{c}\right)^2\right] \tag{3}$$

where $c \in R$.

Exercises

Assume, for simplicity, that $c = 1$ is taken for the Cauchy function

[1] If you are interested in more information on this issue you may begin with the reference Steward, C. "Robust parameter estimation in computer vision", SIAM Review, Vol. 41, No. 3, pp. 513–537, 1999.

1. Plot the least squares function, $\rho(u) = \frac{1}{2}u^2$, and compare it with the Cauchy function, Eq. (2), in order to see the "importance" that is given to each prediction error u. You may, for instance, plot the function $\rho(u)$ for $|u| \leq 10$.

2. Compute the parameters for both models using the Cauchy function. For that issue you will need to use the a gradient descent method (there is no need to use the Newton method). Plot the results and and compare them with the results you obtained with the least squares method.

Report

You are asked to deliver an *individual* report (PDF, notebook, or whatever else you prefer). Comment each of the steps you have followed as well as the results and plots you obtain. Do not expect the reader (i.e. me) to interpret the results for you. I would like to see if you are able to understand the results you have obtained.

If you want to include some parts of code, please include it within the report. Do not include it as separate files. You may just deliver the Python notebook if you want.