Least square regression method for m degree polynomial

Abduallah Mohamed
Email: abduallah.mohamed@utexas.edu

September 20, 2017

Abstract

This paper is a solution for the first home work, for course Advanced traffic theory, Fall 2017 , taught by Dr. Christian Claudel at University of Texas at Austin .

1 Introduction

The problem is mainly related to the method of least square, targeting concepts like over fitting, under fitting and exploring different the effect of choosing the degree of a polynomial on the fitting behavior of a specific dataset. The first section will present the problem itself, followed by the second section which show how to do a least square fitting for m degree polynomial, then the third section contains answers for the questions. The code is open source: https://github.com/abduallahadel/leastsquaremethods

2 Problem statement

A dataset relating two given variables x and y has been collected through experiments, and is detailed in the following table: Training:

x	-12	-10	-8	-6	-4	-2	0	2	4	6	8	10	12
y	-60	-34	-3	11	6	-1	5	11	-3	2	23	35	73

Validation

x	-11	-9	-7	-5	-3	-1	1	3	5	7	9	11
y	-37	-10	6	9	4	-2	6	11	17	22	37	44

Our objective is to build a model between x_i and x_i , such that $f(x_i) = y_i$. For this, we split the dataset in two: a training dataset and a validation dataset,

1. Plot the training dataset (do not plot the validation dataset yet).
2. Use least squares to fit polynomial models of increasing degree, from degree zero (constant) to degree 7.
3. Plot the root mean squared error of the model, i.e. the square root of the average of the squared errors between the model and the actual data, as a function of the order of the polynomial, using the data from the training dataset.
4. Now use the same polynomial model fits (that you computed for question 2) to determine the root mean square error on the model, using the validation dataset. Which polynomial order is best?
5. Plot, on the same graph an example of polynomial model that clearly under-fits the data, and an example of model that clearly over-fits the data.

3 Deriving least squares for m degree polynomial

As in [1], it been shown how to derive the least square methods for a second degree polynomial, by defining an error function E of the polynomial coefficients, in the next section we are expanding it into m degree polynomial.

let $\mathrm{E}[a_0, \cdots, a_m] = \sum_{n=1}^{N}(y_n - (a_m x_n^m + a_{m-1}x_n^{m-1} + \cdots + a_0))^2$ which is the error function between m degree polynomial and our N dataset points.

In order to minimize the error function , we need these conditions

$$\begin{aligned} \frac{\partial E}{\partial a_0} &= 0 \\ &\cdots \\ \frac{\partial E}{\partial a_{m-1}} &= 0 \\ \frac{\partial E}{\partial a_m} &= 0 \end{aligned} \tag{1}$$

Solving the previous equations, yields

$$\begin{aligned} \frac{\partial E}{\partial a_0} &= -2\sum_{n=1}^{N}(y_n - (a_m x_n^m + a_{m-1}x_n^{m-1} + \cdots + a_0))(1) \\ &\cdots \\ \frac{\partial E}{\partial a_{m-1}} &= -2\sum_{n=1}^{N}(y_n - (a_m x_n^m + a_{m-1}x_n^{m-1} + \cdots + a_0))(x_n^{m-1}) \\ \frac{\partial E}{\partial a_m} &= -2\sum_{n=1}^{N}(y_n - (a_m x_n^m + a_{m-1}x_n^{m-1} + \cdots + a_0))(x_n^m) \end{aligned} \tag{2}$$

Setting

$$\frac{\partial E}{\partial a_0} = \cdots = \frac{\partial E}{\partial a_{m-1}} = \frac{\partial E}{\partial a_m} = 0 \tag{3}$$

and some reformatting

$$\begin{aligned} \sum_{n=1}^{N} y_n &= a_m(\sum_{n=1}^{N} x_n^m) + a_{m-1}(\sum_{n=1}^{N} x_n^{m-1}) + \cdots + a_0(\sum_{n=1}^{N} 1) \\ &\cdots \\ \sum_{n=1}^{N} y_n x_n^{m-1} &= a_m(\sum_{n=1}^{N} x_n^{2m-1}) + a_{m-1}(\sum_{n=1}^{N} x_n^{2m-2}) + \cdots + a_0(\sum_{n=1}^{N} x_n^{m-1}) \\ \sum_{n=1}^{N} y_n x_n^m &= a_m(\sum_{n=1}^{N} x_n^{2m}) + a_{m-1}(\sum_{n=1}^{N} x_n^{2m-1}) + \cdots + a_0(\sum_{n=1}^{N} x_n^m) \end{aligned} \tag{4}$$

We may rewrite these equations as

$$\begin{bmatrix} \sum_{n=1}^{N} x_n^{2m} & \sum_{n=1}^{N} x_n^{2m-1} & \cdots & \sum_{n=1}^{N} x_n^m \\ \sum_{n=1}^{N} x_n^{2m-1} & \sum_{n=1}^{N} x_n^{2m-2} & \cdots & \sum_{n=1}^{N} x_n^{m-1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{n=1}^{N} x_n^m & \sum_{n=1}^{N} x_n^{m-1} & \cdots & \sum_{n=1}^{N} 1 \end{bmatrix} \begin{bmatrix} a_m \\ a_{m-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{N} y_n x_n^m \\ \sum_{n=1}^{N} y_n x_n^{m-1} \\ \vdots \\ \sum_{n=1}^{N} y_n \end{bmatrix} \tag{5}$$

Solving it with respect to the polynomial coefficients :

$$\begin{bmatrix} a_m \\ a_{m-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{N} x_n^{2m} & \sum_{n=1}^{N} x_n^{2m-1} & \cdots & \sum_{n=1}^{N} x_n^m \\ \sum_{n=1}^{N} x_n^{2m-1} & \sum_{n=1}^{N} x_n^{2m-2} & \cdots & \sum_{n=1}^{N} x_n^{m-1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{n=1}^{N} x_n^m & \sum_{n=1}^{N} x_n^{m-1} & \cdots & \sum_{n=1}^{N} 1 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{n=1}^{N} y_n x_n^m \\ \sum_{n=1}^{N} y_n x_n^{m-1} \\ \vdots \\ \sum_{n=1}^{N} y_n \end{bmatrix} \tag{6}$$

let A equals this matrix:

$$\begin{bmatrix} \sum_{n=1}^{N} x_n^{2m} & \sum_{n=1}^{N} x_n^{2m-1} & \cdots & \sum_{n=1}^{N} x_n^{m} \\ \sum_{n=1}^{N} x_n^{2m-1} & \sum_{n=1}^{N} x_n^{2m-2} & \cdots & \sum_{n=1}^{N} x_n^{m-1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{n=1}^{N} x_n^{m} & \sum_{n=1}^{N} x_n^{m-1} & \cdots & \sum_{n=1}^{N} 1 \end{bmatrix}$$

Equation 6 is valid if and only if A^{-1} exists which implies that $det(\mathrm{A}^{-1}) \neq 0$

4 Solutions

4.1 Plot the training dataset (do not plot the validation dataset yet).

1.

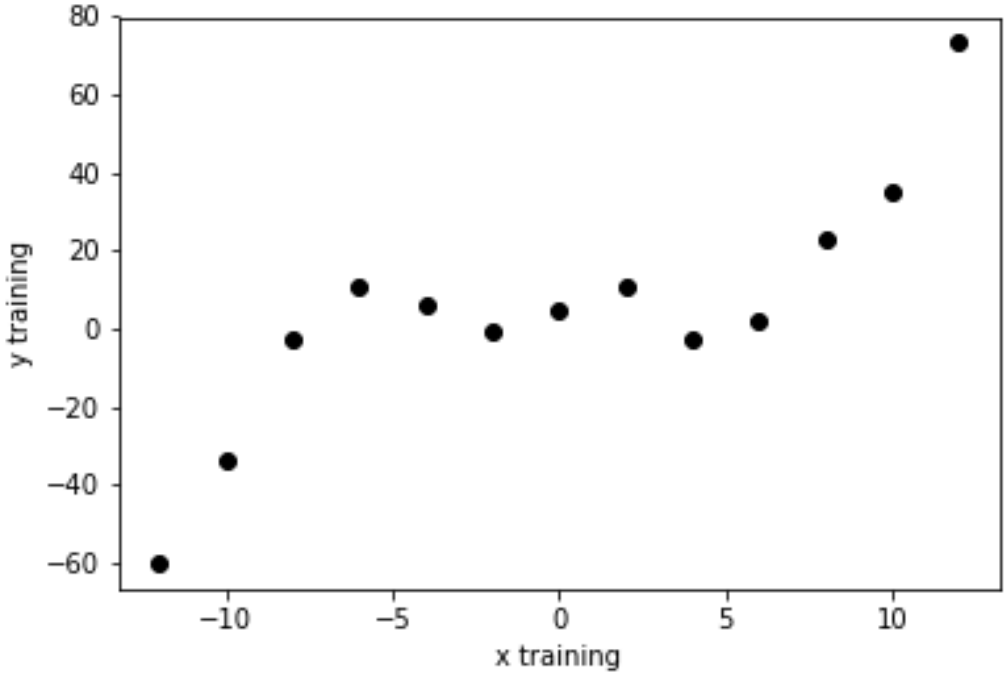

Figure 1: x training vs y training dataset

4.2 Use least squares to fit polynomial models of increasing degree, from degree zero (constant) to degree 7.

Figure 2 shows different choices of the degree of polynomial and the resulted model versus actual training data.

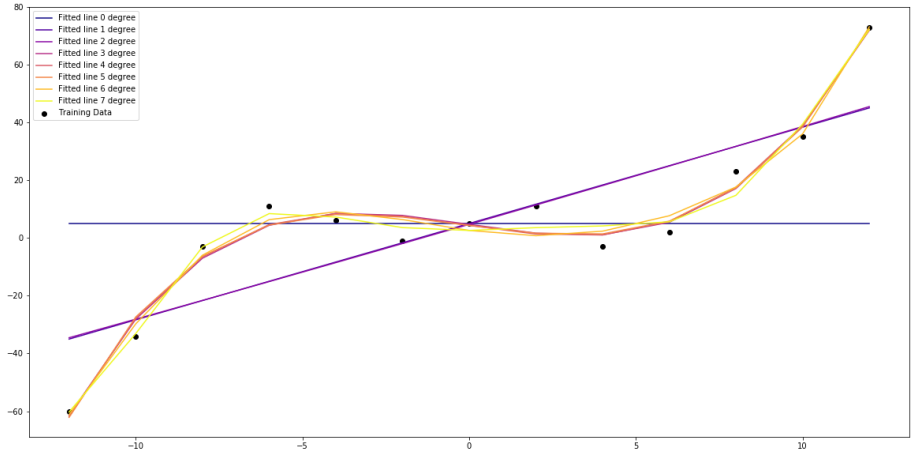

Figure 2: Visualization of multiple fitting lines vs the training dataset

4.3 Plot the root mean squared error of the model, i.e. the square root of the average of the squared errors between the model and the actual data, as a function of the order of the polynomial, using the data from the training dataset.

From figure 3, we can see an improvement of MSE versus increasing the degree of polynomial, nevertheless this doesn't mean we have a good model for our data, without checking the behavior of a testing set, we can't say that increasing the degree of polynomial is in our favor.

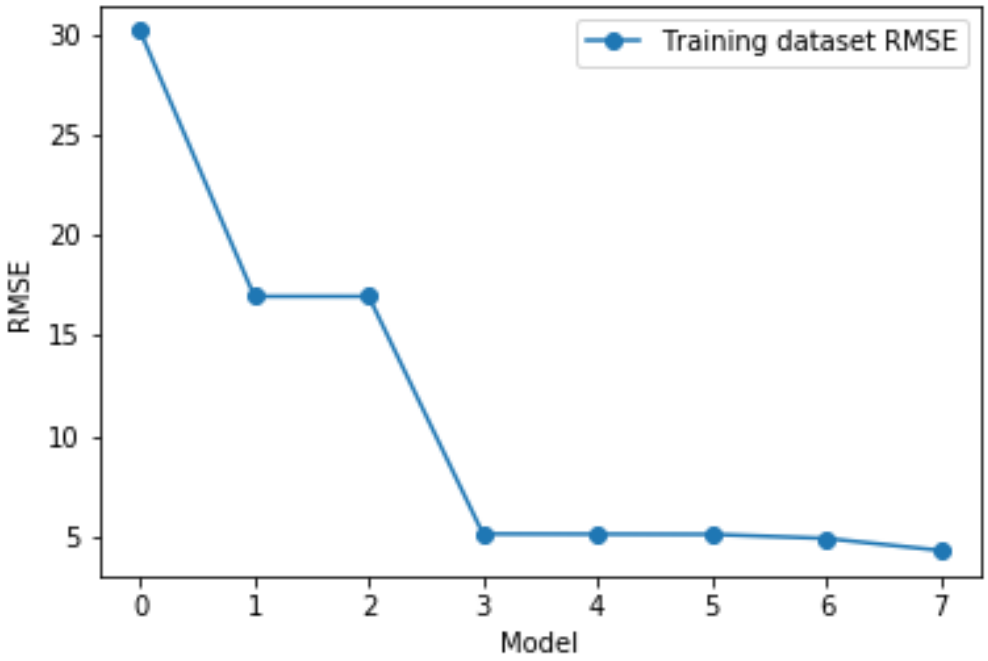

Figure 3: RMSE per model for testing dataset

4.4 Now use the same polynomial model fits (that you computed for question 2) to determine the root mean square error on the model, using the validation dataset. Which polynomial order is best?

From figure 4 we can see the the model with 6 degree polynomial is the best, because it's exactly between the point , where the models starts to over-fit, but this doesn't exclude that models from 3 to 5 degree polynomial are also a good option.

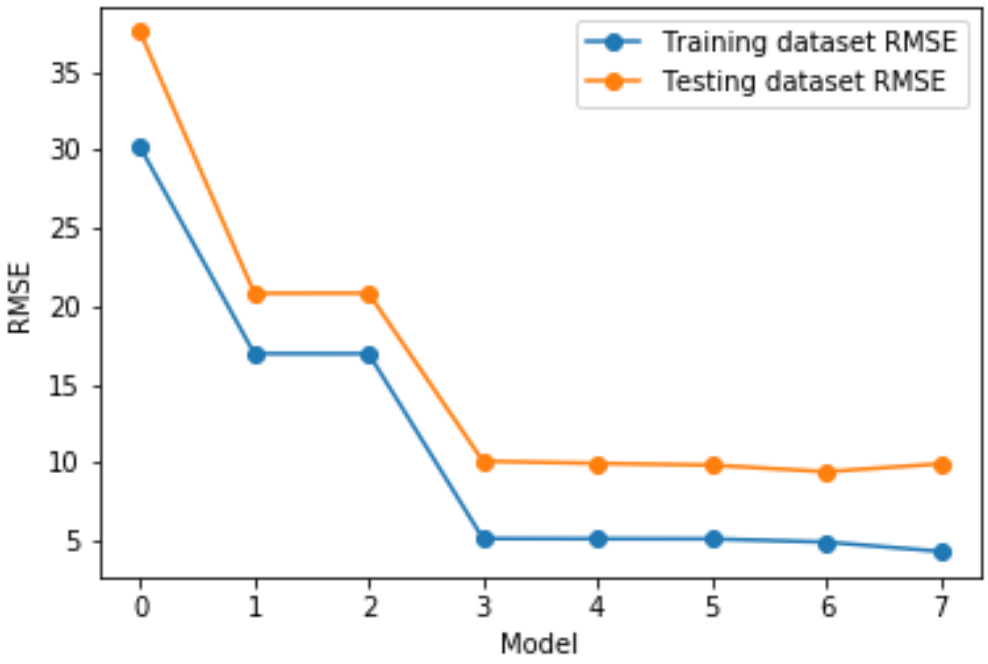

Figure 4: RMSE per model for testing dataset

4.5 Plot, on the same graph an example of polynomial model that clearly under-fits the data, and an example of model that clearly over-fits the data.

We can see that the model with 0 degree polynomial in figure 5 is a clear example of under fitting, also we can see that model with 7 degree polynomial in figure 6 is a clear example of over-fitting, mainly because the RMSE of testing error started to increase at this point .

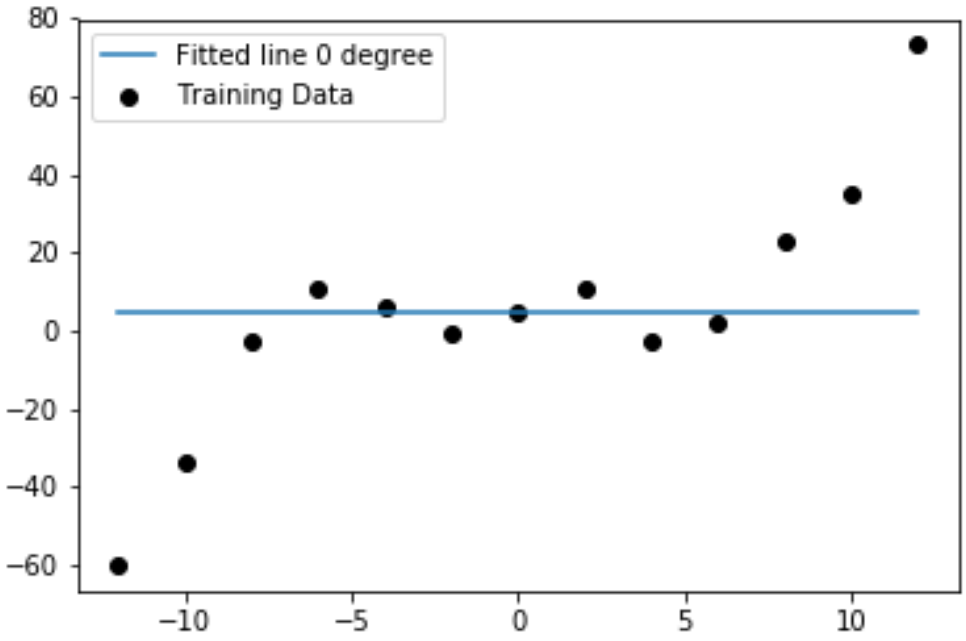

Figure 5: A model that under-fits

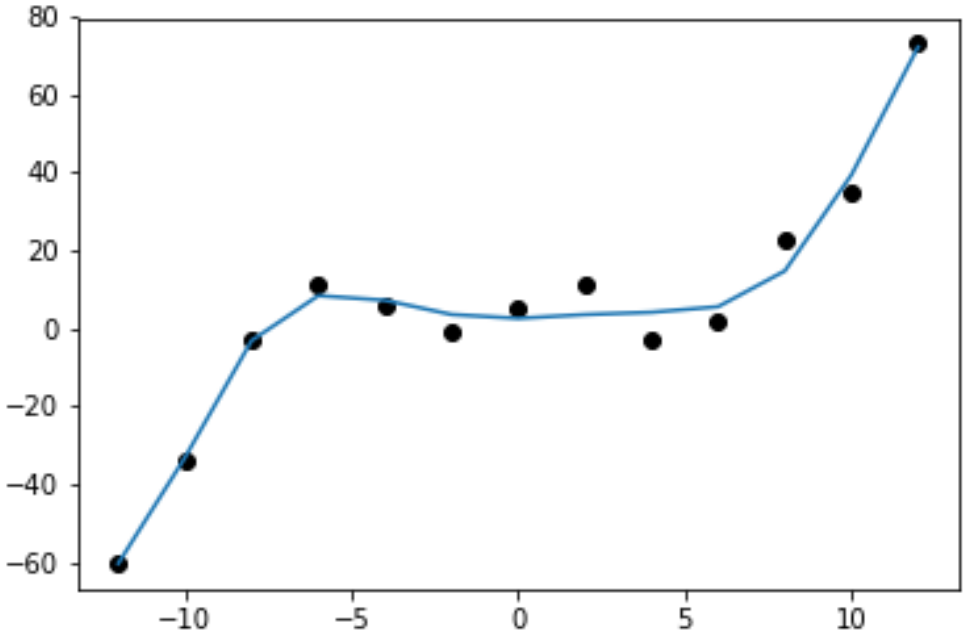

Figure 6: A model that over-fits

References

[1] S.J. Miller. The Method of Least Squares and Signal Analysis. pages 1–7, 1992.