Machine Learning

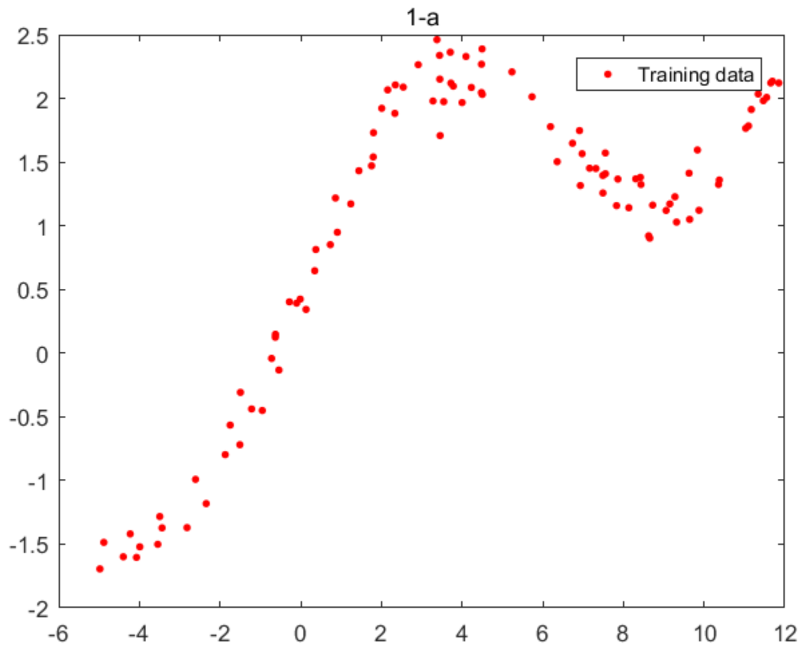
Solutions to Assignment 1

Student: Qingbo Kang **Student ID: 40058122**

1. Linear and polynomial regression

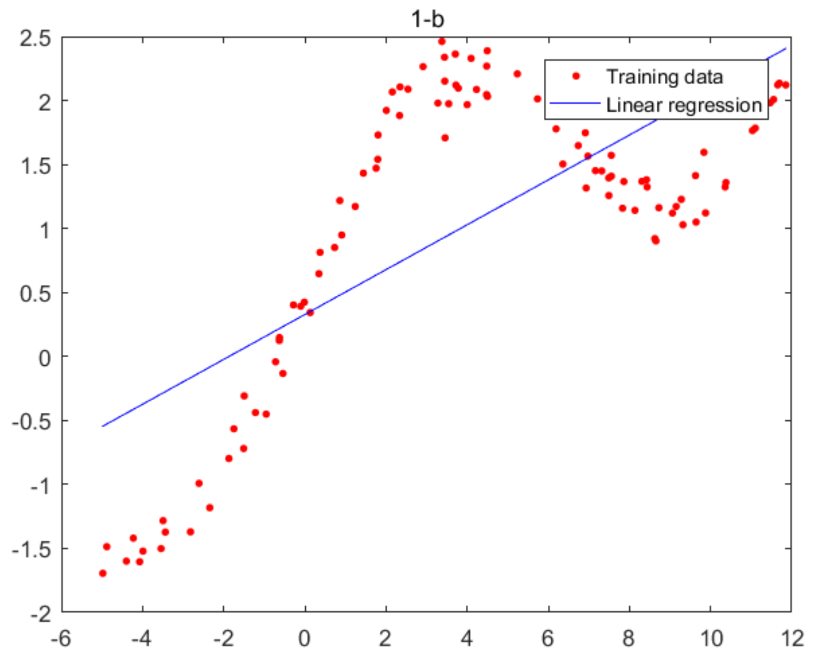
(a) See the code file Solution_1.m

Below shows the figure of the running result:

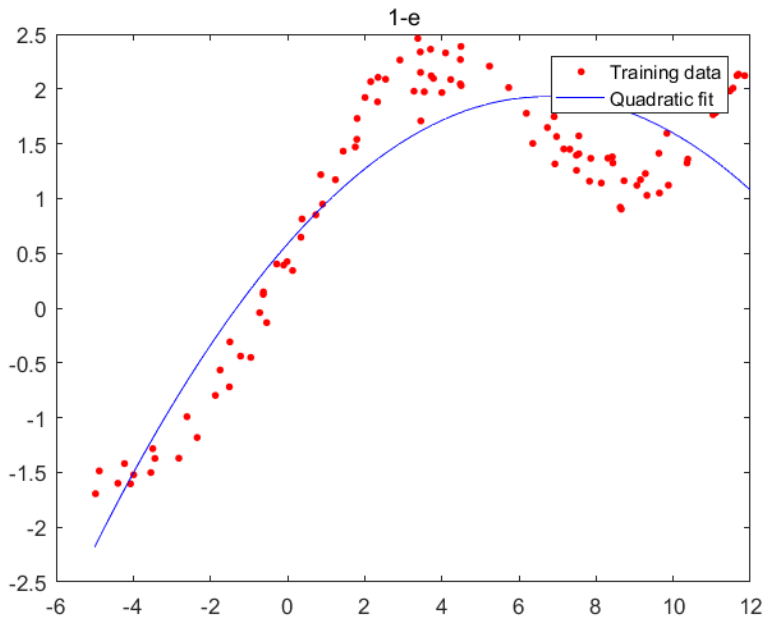


(b) See the code file Solution_1.m

Below shows the figure of the running result:

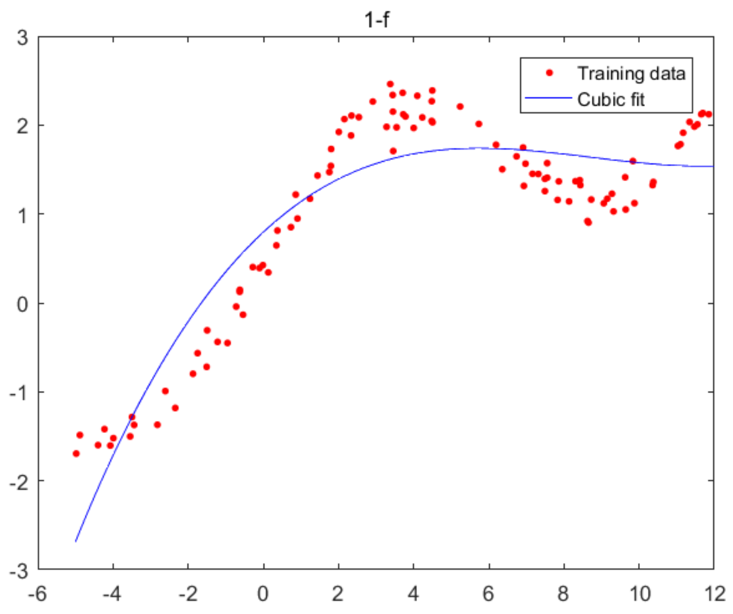


(c) See the code file evaluteTrainError.m
    The training error of the resulting fit is: 33.336445
(d) See the code file PolyRegress.m.
(e) See the code file Solution_1.m
    Below shows the figure of the running result:



The training error of the quadratic fit is: 12.612585, since the training error is smaller, this is a better fit than the linear regression fit.

(f) See the code file Solution_1.m
    Below shows the figure of the running result:



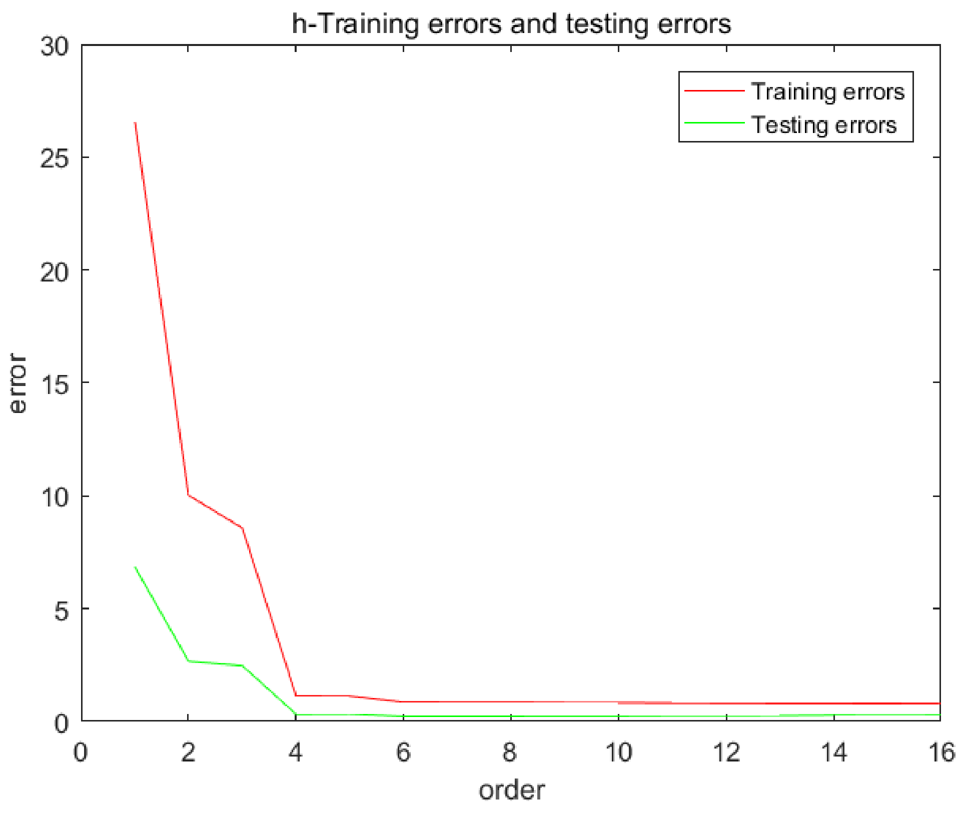
The training error of the cubic fit is: 10.877777, since the training error is smaller, this is a better fit than the quadratic regression fit.

(g) In this case, the ordered data will result in choosing the optimal model which has less complexity which means has a small order. The randomness in input data is eliminated means a small certain range of data will have a high compact to the training result model, hence let the result model to be biased.

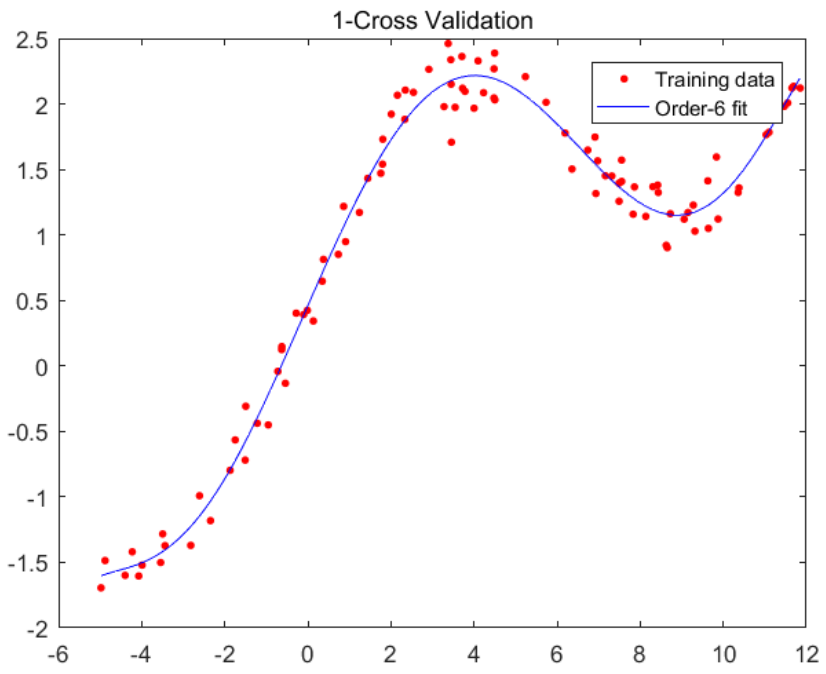
(h) See the code file FiveFoldCrossValidation.m
The best degree for polynomial regression is 6.
The data that supports my conclusion is the error on the testing set, that is:

order	The error on the testing set
1	6.7797
2	2.5691
3	2.3698
4	0.3306
5	0.3361
6	0.2633
7	0.2643
8	0.2653
9	0.2664
10	0.2649
11	0.2753
12	0.2801

And the figure of training errors vs testing errors:



Because of the property of cross-validation, from the testing error we can see the lowest is 0.2633 for order-6 fit, so the order-6 fit is the best fit.
For the order-6 fit, the figure shows below:



(i) See the code file Solution_1.m
The best degree for polynomial regression is still 6.

The data that supports my conclusion is the error on the testing set, that is:

order	The error on the testing set
1	1.2226
2	0.4313
3	0.4421
4	0.0558
5	0.0773
6	0.0414
7	0.0421
8	0.0427
9	0.0766
10	0.2151
11	0.3549
12	0.0508

(j) After feature normalization, we have:

$$h_w(x) = \sum_{j=1}^{d+1} w_j \frac{x_j}{\max_i |x_{i,j}|} = w^T XM \quad \text{for } i = 1,\dots,m \text{ and } j = 1,\dots,d+1 .$$

where M is a diagonal matrix:

$$M = \begin{bmatrix} \frac{1}{\max_i |x_{i,d+1}|} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{\max_i |x_{i,d}|} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\max_i |x_{i,1}|} & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix},$$

So the solution to w is:

$$\begin{aligned} w^* &= ((XM)^T(XM))^{-1}(XM)^T Y = (M^T(X^TX)M)^{-1}M^TX^TY \\ &= M^{-1}(X^TX)^{-1}(M^T)^{-1}M^TX^TY \end{aligned},$$

Since

$$(M^T)^{-1}M^T = I_{d+1},$$

So we get:

$$w^* = M^{-1}(X^TX)^{-1}X^TY ,$$

And M^{-1} is a constant diagonal matrix, which is:

$$M^{-1} = \begin{bmatrix} \max_i |x_{i,d+1}| & 0 & \cdots & 0 & 0 \\ 0 & \max_i |x_{i,d}| & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \max_i |x_{i,1}| & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

So it prove that this change results in a scaling of the output, but has no other effect on the approximator.

2. Weighted linear regression

(a) U is the diagonal matrix of dimension[$m*m$], more specifically,

$$U = \begin{bmatrix} u_1 & 0 & \cdots & 0 & 0 \\ 0 & u_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & u_{m-1} & 0 \\ 0 & 0 & \cdots & 0 & u_m \end{bmatrix}$$

(b)

$$\begin{aligned} \nabla_W J(W) &= \nabla_W[(XW-y)^T U(XW-y)] \\ &= \nabla_W(W^T X^T UXW - W^T X^T Uy - y^T UXW + y^T Uy) \\ &= 2X^T UXW - 2X^T Uy \end{aligned}$$

Setting this to 0:

$$\begin{aligned} &2X^T UXW - 2X^T Uy = 0 \\ &\Rightarrow X^T UXW = X^T Uy \\ &\Rightarrow W = (X^T UX)^{-1} X^T Uy \end{aligned}$$

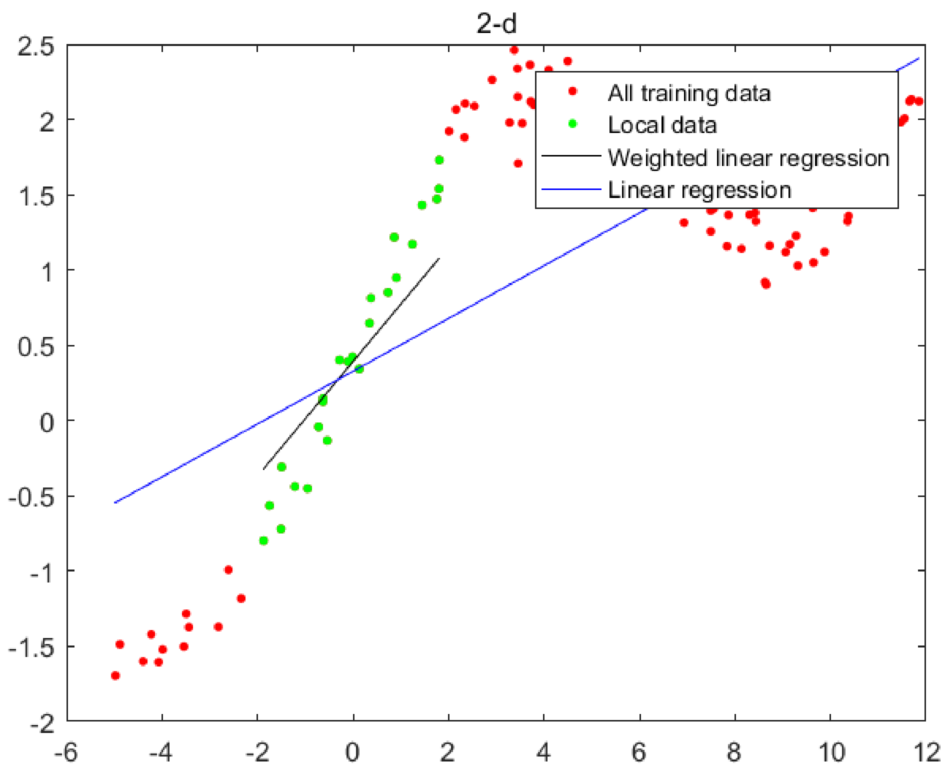
(c) The code can be seen in Solution_2.m

I increase the weight of the largest input value from 0 to 10, and I observed that as the weight increase, the result fitting line are more and more close to the point with the largest input value. The implication is that the point with the largest value become more important in the result fitting model.

(d) The code can be seen in Solution_2.m

When I want to predict the output value which input value is just lies in range [-2, 2], instead of giving all the data samples the same weight, it’s better to give the data points which lies in range [-2, 2] a much higher weight, so under the circumstances, the weighted linear regression works better than the unweighted version.

The result figure shows below:



In this graph, the black line is the weighted linear regression fitting line, whereas the blue line is the unweighted linear regression fitting line. It can be seen that in the range [-2, 2], the weighted linear regression line works much better than the unweighted linear regression line.

3. Error criterion for exponential noise

Adopt the assumption that:

$$y_i = h_w(x_i) + \varepsilon_i,$$

where $\varepsilon_i \sim E(\lambda)$.

the best hypothesis maximizes the likelihood of $y_i - h_w(x_i) = \varepsilon_i$.

And because the noise variables ε_i are from an exponential distribution:

$$L(w) = \prod_{i=1}^{m} \lambda e^{-\lambda(y_i - h_w(x_i))}$$

$$\log L(w) = \sum_{i=1}^{m} \log(\lambda e^{-\lambda(y_i - h_w(x_i))})$$

$$= \sum_{i=1}^{m} \log(\lambda) - \sum_{i=1}^{m} \lambda(y_i - h_w(x_i))$$

So maximizing the likelihood parameters w is the same as minimizing:

$$w^* = \arg\min_w \sum_{i=1}^{m} \lambda(y_i - h_w(x_i)).$$

In conclusion, the error criterion minimized in the exponential noise case is:

$$\sum_{i=1}^{m} \lambda(y_i - h_w(x_i)).$$