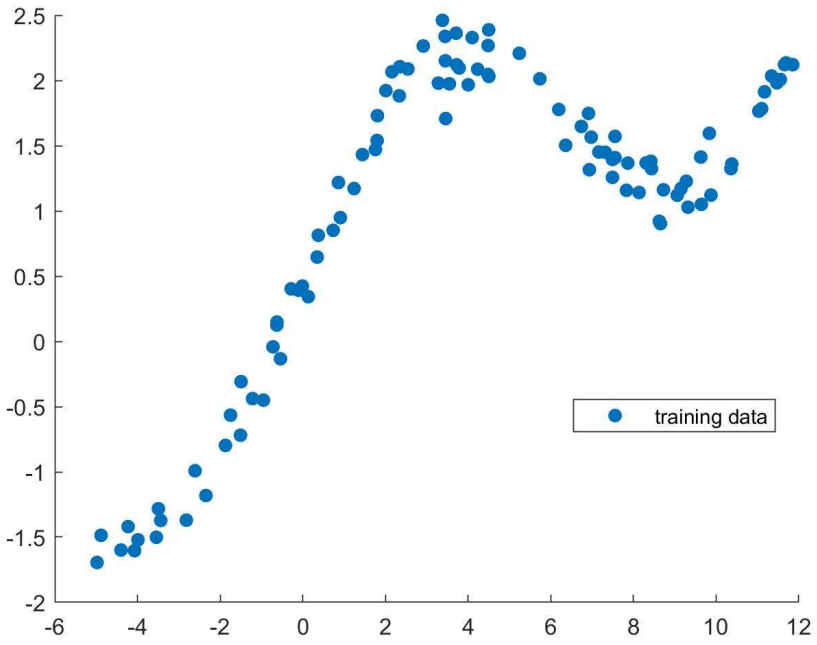
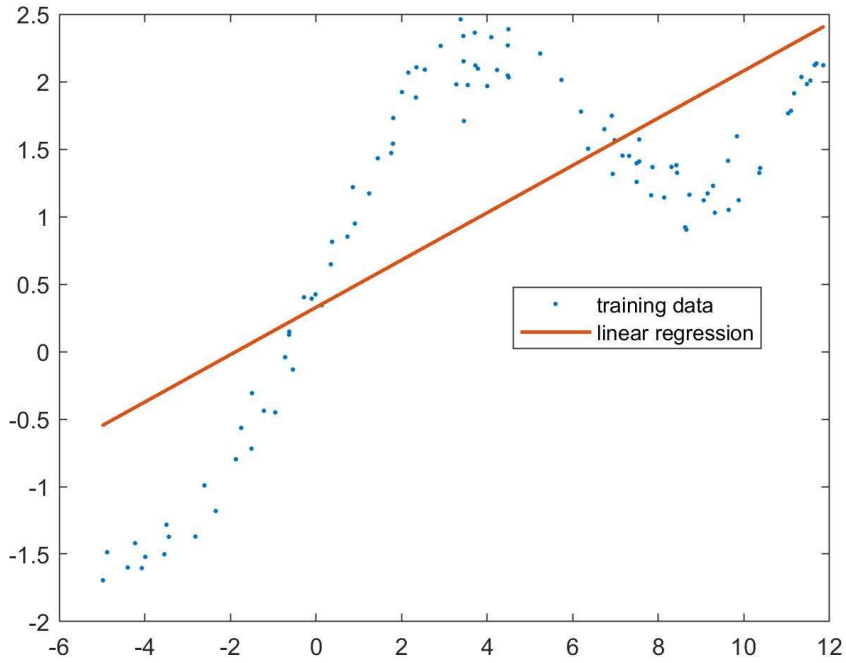
Question 1

(a)



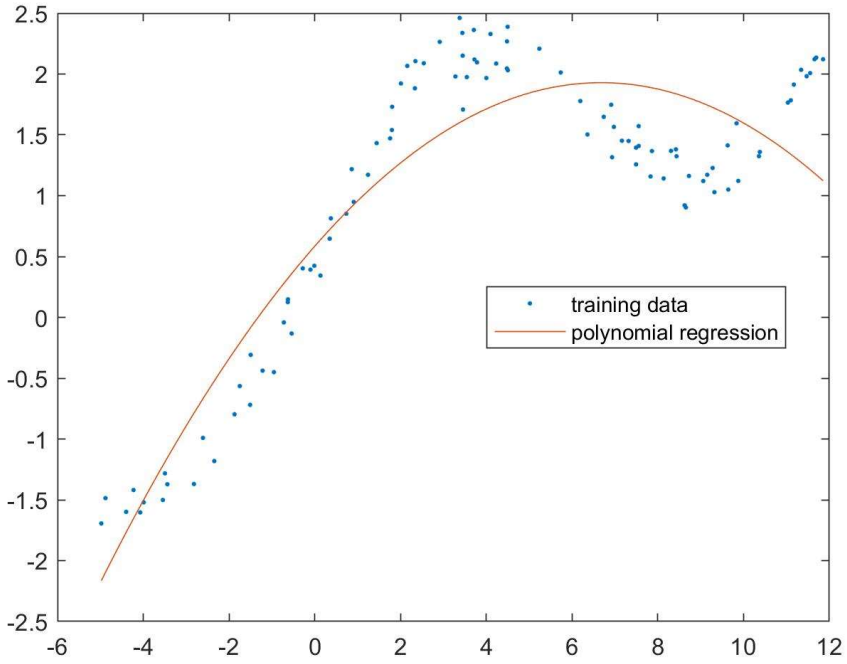
(b)



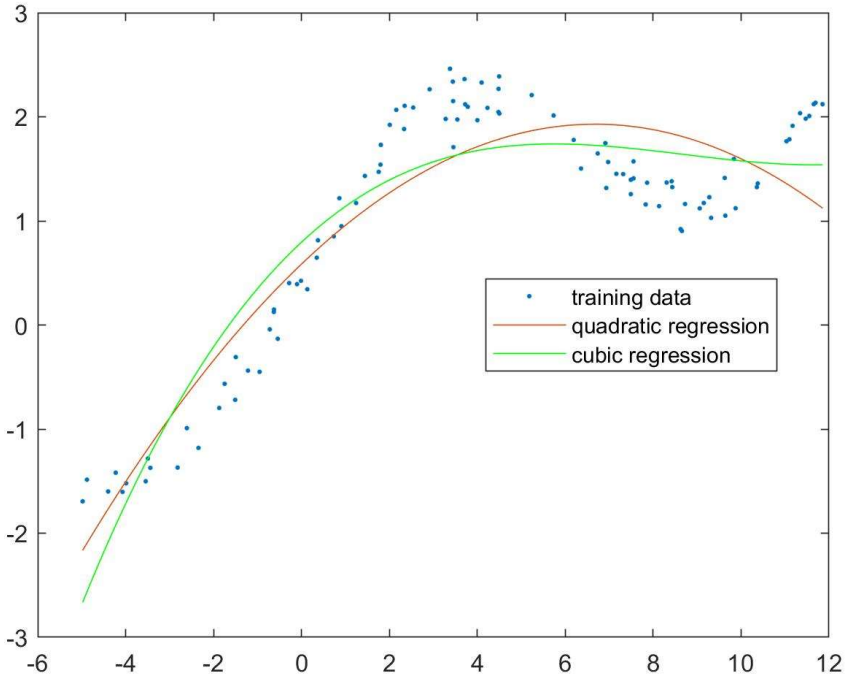
(c) The training error of the resulting fit is: 33.336445

(d) the function code in the q1.m

(e) The training error of the quadratic fitting line is: 12.612585



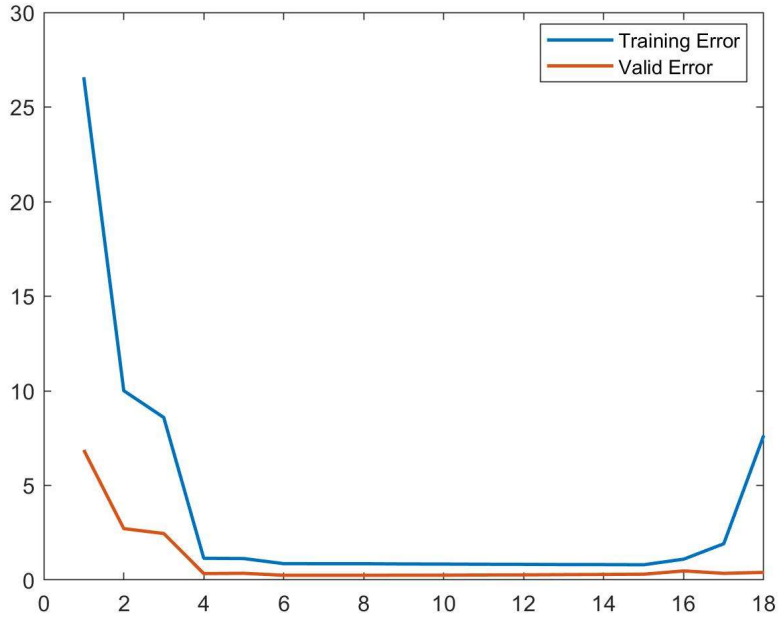
(f) The training error of the cubic fitting line is: 10.877777

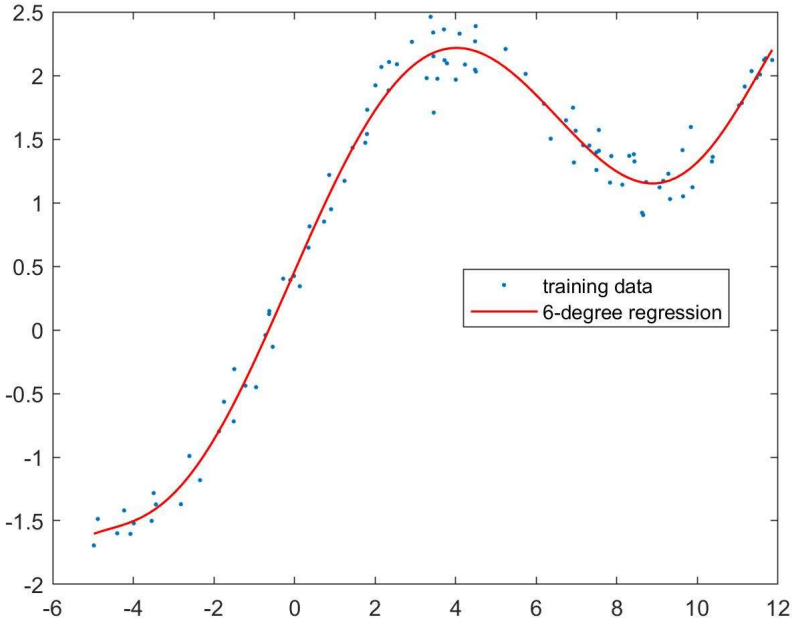


(g) Using K-fold cross validation would be able to determine the best degree for polynomial regression. Increasing the degree for polynomial regression may fix under-fitting problem. but may also cause the over-fitting. So, we can see the valid error and training error go down to the minimum then increasing. The minimum point means the best degree that would fit the training data.

(h)From table we can see that degree 6 has the minimum valid error.

Degree	Training Error	Valid Error
1	26.57760429	6.873577407
2	10.00735617	2.716180645
3	8.587672799	2.456318874
4	1.147246616	0.338753444
5	1.132806528	0.354377515
6	0.86244051	0.248961634
7	0.861429558	0.249026714
8	0.859978265	0.250890687
9	0.846216569	0.253011496
10	0.841432106	0.25453543
11	0.83198333	0.264772254
12	0.826773998	0.269180843
13	0.815992724	0.282824124
14	0.813854217	0.293148101
15	0.805716895	0.305900643
16	1.101950753	0.479639037
17	1.914800163	0.351004104
18	7.643500852	0.402783165





(i)

From the table we can see, the best degree is still 6.

Degree	Training Error	Valid Error
1	26.56495135	6.897089676
2	9.942238977	2.866452176
3	8.546103544	2.551706174
4	1.137981798	0.360118987
5	1.128097579	0.362331472
6	0.850735287	0.277398371
7	0.849388164	0.278211055
8	0.84784775	0.27936917
9	0.833854827	0.281058399
10	0.830707494	0.280472826
11	0.824114563	0.281241805
12	0.819420747	0.285793932
13	0.810135888	0.298391664
14	0.80455237	0.319694509
15	0.799393536	0.313452406
16	0.81740234	0.333696816
17	4.635672434	2.400114213
18	2.093174735	0.493901471

(j)

Suppose the normalization matrix M:

$$h_w(x) = w_n XM$$

So, the normalized parameter w_n is:

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

Since the M^{-1} is a constant matrix, so this change results in a scaling of the output. but has no other effect on the approximator.

Question 2

(a) U is a [$m*m$]diagonal matrix

$$U = \begin{vmatrix} u_1 & 0 & 0 & \cdots & 0 \\ 0 & u_2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \cdots & u_m \end{vmatrix}$$

(b) Similar to the multivariate calculus in the class

$$J(w) = (Xw - y)^T U(Xw - y)$$

$$\nabla_w J(w) = \nabla_w(w^T X^T UXw - w^T X^T Uy - y^T UXw + y^T Uy)$$

$$\nabla_w J(w) = \nabla_t race(w^T X^T UXw - w^T X^T Uy - y^T UXw + y^T Uy)$$

$$\nabla_w J(w) = 2X^T UXw - 2X^T Uy$$

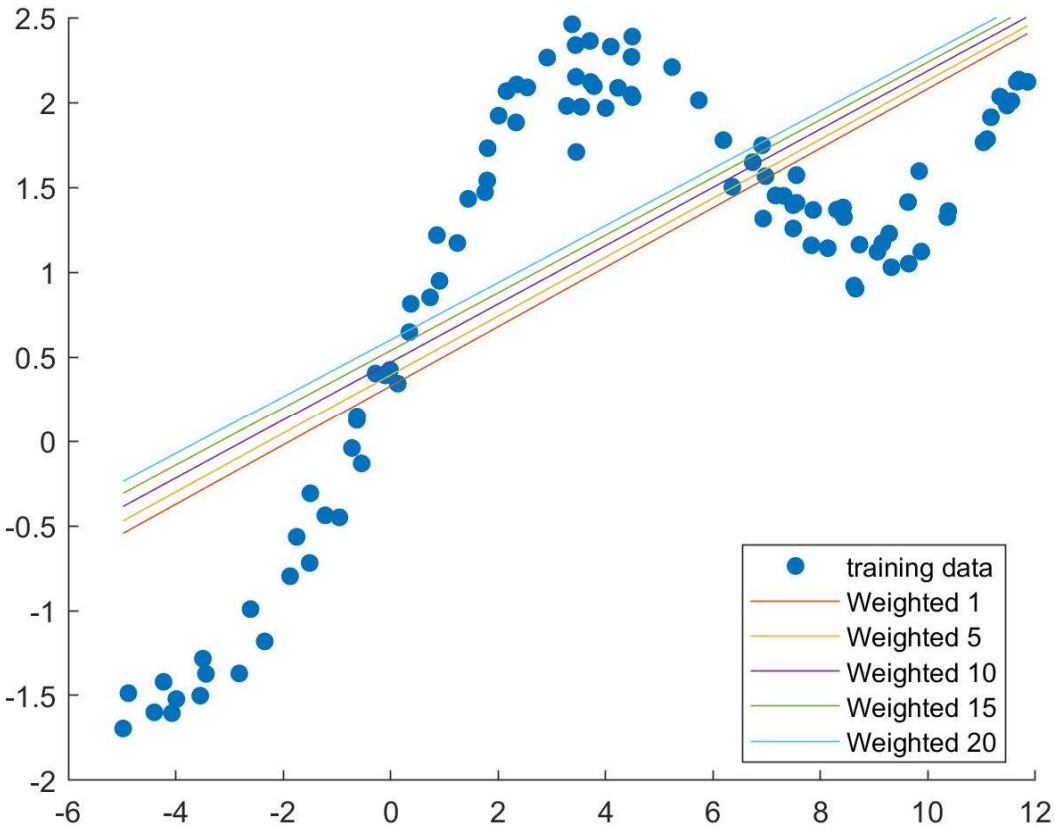
Set gradient to 0:

$$\nabla_w J(w) = 2X^T UXw - 2X^T Uy = 0$$

$$w = (X^T UX)^{-1} X^T Uy$$

(c)

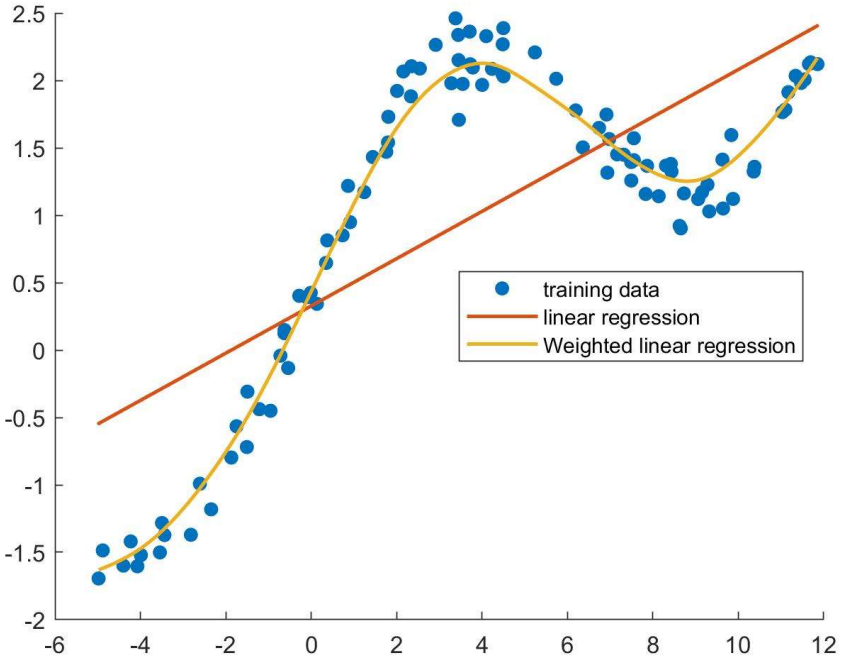
I set all the other point weight to 1, and increase the largest input value from 1 to X, it's obviously to see, with the weight growing, the linear regression line becoming closer to the largest output point. Because the weighted linear regression not treat all the input value equally, the higher weighted input is more important to the results than lower ones, so it can help to avoid the over fitting in some cases.



(d)

I implemented a weighted linear regression by using Gaussian distribution to decide the weight of each training point, it can focus on the big picture and ignore the noise which may cause over-fitting. For each input point of fitting line, defining the weight according to the distance to every training data point, the closer one gets the higher weight. So, in the end, every point of the fitting line closes to the training point which is closest to the current point, we can get an exactly fitting line with using a weighted linear regression instead of the polynomial regression.

In this graph, we can see the weighted linear regression line works much better than the unweighted linear regression line.



Question 3

Adopt the assumption that:

$y_i = h_w(x_i) + \epsilon_i$ where $\varepsilon_i \sim E(\lambda)$.

$$p_\lambda(t) = \begin{cases} \lambda e^{-\lambda t} & if t \geq 0 \\ 0 & otherwise \end{cases}$$

the best hypothesis maximizes the likelihood of $y_i = h_w(x_i) + \epsilon_i$.

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

Apply the log trick:

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

So maximizing the likelihood need to minimizing:

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

The error criterion minimized is:

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