*1. Linear and polynomial regression*

*For this exercise, you will experiment in Matlab with linear and polynomial regression on a given data set. The inputs are in the \_le hw1x.dat and the desired outputs in hw1y.dat.*

*(a) [5 points] Load the data into memory and plot it (using the load and plot functions; use the help function if you do not know how to call them).*

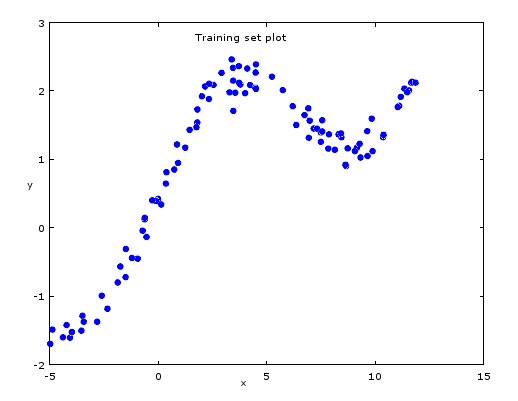


Fig 1. Training set plot

*(b) [5 points] Add a column vector of 1s to the inputs, then use the linear regression*

*formula discussed in class to obtain a weight vector w. Plot both the linear*

*regression line and the data on the same graph. (Note: matrix formulas translate*

*almost verbatim in Matlab)*

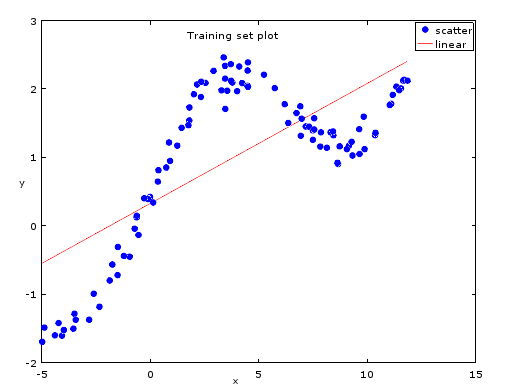
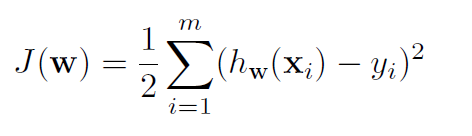


Fig 2. Linear Regression

*(c) [5 points] Write a Matlab function that will evaluate the training error of the*

*resulting fit, and report what this error is.*

the error function is denoted as sum-of-square function:



After obtain the weight vector, and knowing X, and Y

W= 0.17531

0.32768

The training error is = 33.336

*(d) [5 points] Write a Matlab function called PolyRegress(x,y,d) which adds the fea-*

*tures x2; x3; :::xd to the inputs and performs polynomial regression.*

The implementation in in Q1.

*(e) [5 points] Use your function to get a quadratic fit of the data. Plot the data and*

*the fit. Report the training error. Is this a better fit?*

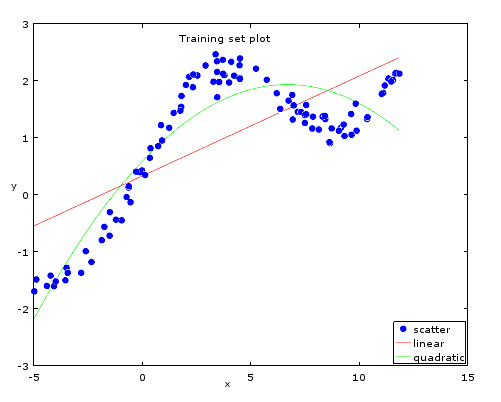


Fig 3. Quadratic regression

After obtain the weight vector, and knowing X, and Y

W= -0.030122

0.402405

0.585094

The training error is = 12.613

It is a better fit since the training error is smaller.

(f) [5 points] Repeat the previous exercise for a cubic fit.

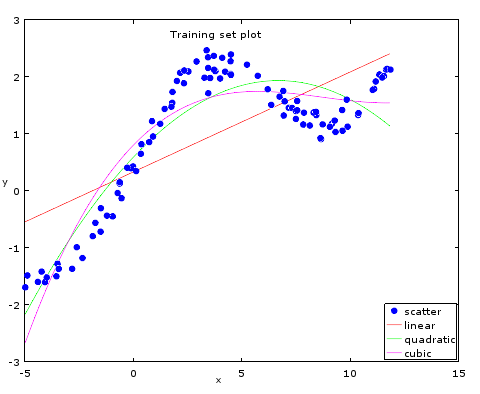


Fig 4. Cubic regression

After obtain the weight vector, and knowing X, and Y

W= 0.0019533

-0.0510024

0.3927950

0.7954747

The training error is = 10.878

It is a better fit since the training error is smaller.

(g) [5 points] Suppose that the data were sorted in increasing value of the target

variable y, and you simply partitioned it by putting the first m=k examples in

the first fold, the next ones in the second fold, etc. Explain what would happen

if you tried to perform cross-validation with these folds.

Perform kth fold cross-validation with these folds could estimate the true error of predicator, and find the best fit model to data set in terms of polynomial order.

By deciding the polynomial order of model, the training error is decreasing as the order increasing, the testing error decreases, then starts increasing again due to over fitting.

If the data sets are organized by increasing order, not equally distributed, the data set will not be independently, and identically distributed.

(h) [10 points]Write a procedure that performs five-fold cross-validation on your data.

Use it to determine the best degree for polynomial regression. Show the data that

supports your conclusion, and explain how you have come to this conclusion. For

the best fit, plot the data and the polynomial obtained.

|  |  |  |
| --- | --- | --- |
| Five-fold cross validation | | |
| degree | Error\_train | Error\_valid |
| 1 | 26.55568 | 6.92412 |
| 2 | 10.03406 | 2.65285 |
| 3 | 8.5881 | 2.47171 |
| 4 | 1.149 | 0.3334 |
| 5 | 1.1382 | 0.33771 |
| 6 | 0.85694 | 0.26229 |
| 7 | 0.85157 | 0.27419 |
| 8 | 14.91192 | 4.17644 |
| 9 | 35.88806 | 9.2989 |
| 10 | 49.34058 | 12.76735 |

Tab. 5 Cross Validation five-fold

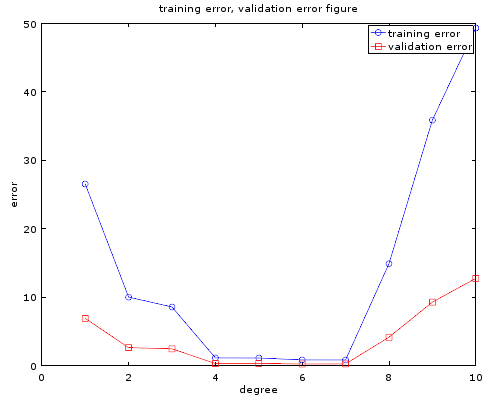


Fig. 5 training error, validation error through all degrees

From the result table we can see degree 6 has the minimum sum of average training error and average validation error. The result is by analyzing polynomial degree from 1 to 10 and by taking the average error of different combination of 5-fold data partition ( 1 out of 5 is validation set, 4 out of 5 is training set).

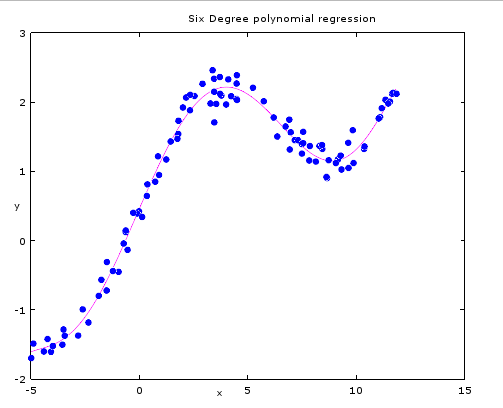


Fig. 6 six degree polynomial regression

The training error is = 1.0936.

Although by additional validation, we found that the 7 degree's polynomial training error of whole data model is = 1.0924

But the difference is too small, and we don't want to have any risk of over fitting. Hence 6 degree polynomial is our best fit.

(i) [10 points] Change the Matlab code such that you normalize the input data in each column by the maximum absolute value in that column. What is the best degree for polynomial regression now? Justify your answer.

|  |  |  |
| --- | --- | --- |
| Normalized Five-fold cross validation | | |
| degree | Error\_train | Error\_valid |
| 1 | 26.55568 | 7.01619 |
| 2 | 10.03406 | 2.63169 |
| 3 | 8.5881 | 2.50249 |
| 4 | 1.149 | 0.42477 |
| 5 | 1.1382 | 0.42697 |
| 6 | 0.85694 | 0.35273 |
| 7 | 0.85157 | 0.3685 |
| 8 | 0.84869 | 0.37166 |
| 9 | 0.83419 | 0.37078 |
| 10 | 0.82882 | 0.36763 |

Table 7. Normalized Five-fold cross validation

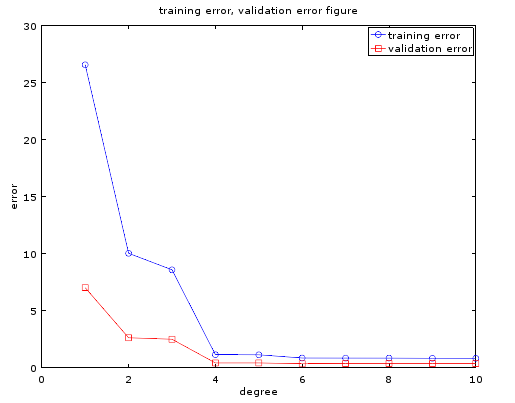


Fig 7. Normalized training error, validation error through degrees

From the result table we can see degree 6 has the minimum sum of average training error and average validation error.

The best fit is still 6 degree with no change, like below figure

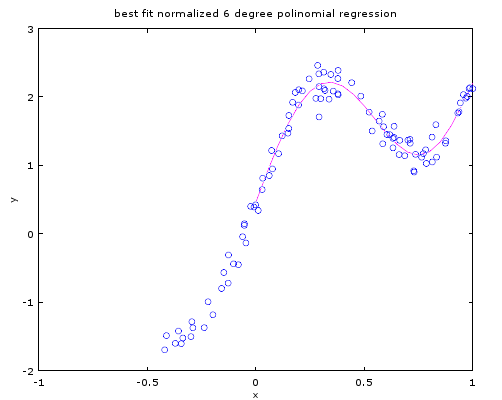


Fig.8 normalized six degree polynomial regression

The training error is 1.0936

(j) [10 points] As you witnessed, polynomial regression often causes the features to

get extreme values, which may cause numerical problems. In such cases, it can

be helpful to normalize the features, e.g. by dividing the value of each feature xj

by maxi jxi;j j; i = 1; : : : ; m; j = 1; : : : ; d + 1, like you did in the example above.

Prove that this change results in a scaling of the output, but has no other e\_ect

on the approximator.

If we normalized the X by times the diagonal matrix , the expression of sum-of-square error function becomes:

its gradient is

gradient is 0 means:

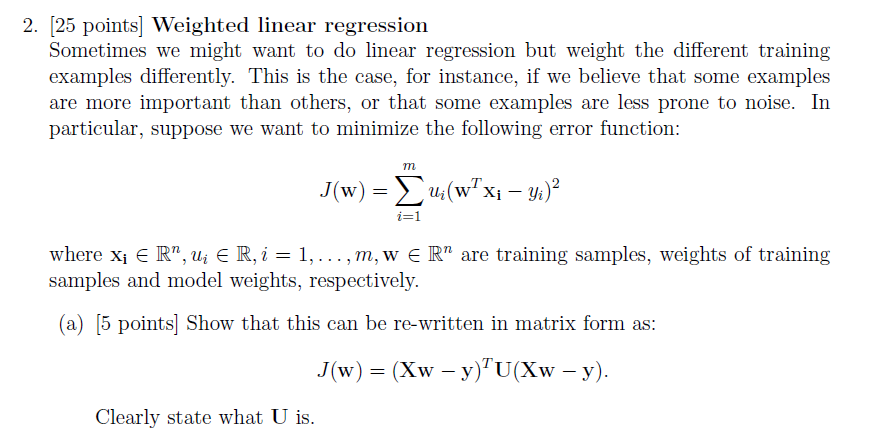
gives:

Diagonal constant matrix can be taken out from transpose. and can be reversed as constant

such normalization will scale up or down the weight factor, but no change.

will become

Hence, error function result will not change，



a) when u is the model weight applies to each of i =1, 2, 3 ....m

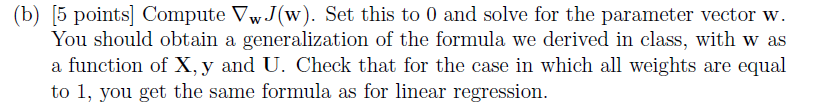
then if

can be rewritten in to

as vector multiplying.

the model weight, can be written as matrix form as

U is the diagonal matrix



expand

take the gradient of w

If all weight are equal to 1

(c) [10 points] Implement weighted linear regression for the data set used in question 1. Weight all points equally, except the point with the largest input value. Gradually increase the weight of this point. Describe what happens, and why.

Linear regression W is 2\*2 matrix, originally started with identity matrix. Then the weight of the largest input value from 1 to 50, by the step of 1.

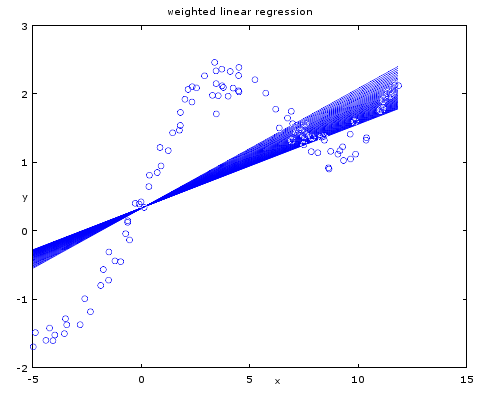


Fig 2.1 weighted linear regression

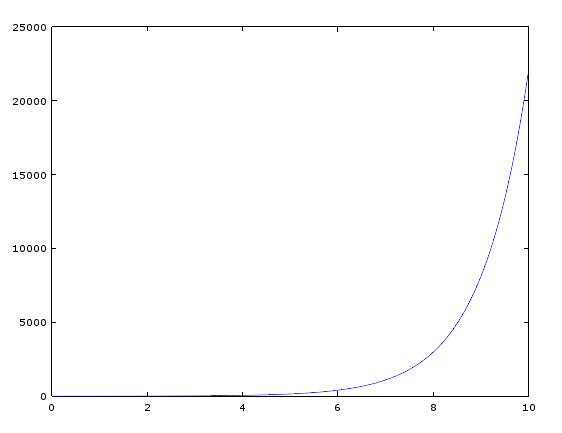
We can see the slope is increasing as the increase of weight. The larger weight of the largest input, the more influence of the term to the cost function.

(d) [5 points] Draw an example of a data set in which you would expect weighted

linear regression to work a lot better than the unweighted version. Explain why

you chose this data set.

if higher order term takes the majority of the weight, the weighted linear regression works better than unweighted.



such plot is the exponential function y = exp(x) , here if we weighted more on the higher order term of linear regression, it could give better description than unweighted.

