Introduction to Machine Learning (ELL784)

Assignment-1 Report

Rahman Alam
2021AMA2090
Applied Mechanics Department

Submitted to

Professor
Sumeet Agarwal

21-sep-2022

Content

Scikit Learn Library

This is a scientific library which contains various model class within itself. Least Square Regression is one part of the various class containing this library. This library helps us apply various model on classification, regression, clustering, dimensionality reduction,

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
```

These three subclasses can be use for simple polynomial regression fit and can be used to calculate mean squared error for the train-test data.
More information about the library can be access here at its documentation.

Design Matrix

Design matrix are the set of linear equations whose solution will give the polynomial coefficient being fit.

I will be using inbuilt class presented in the scikit learn library for the design matrix.

Ex-

```
polynomial_features = PolynomialFeatures(degree = i)
X_poly = polynomial_features.fit_transform(x_train)
design_matrix_20.append(X_poly)
```

Degree=i is used to iterate for M so that we can get various model having degrees M.

Design matrix one can find here for various M from 1 up to 24 degree of polynomials here for 20 data points taken in consideration on the same link for full data also design matrix is available.

Design Matrix for M=1 is shown here

```
[1.        1.24561],[ 1.         -
1.016511],[1.        0.305252],[ 1.         -
1.104524],[ 1.         -
0.785528],[1.        0.486322],[ 1.         -
1.720929],[ 1.         -
0.150237],[1.        1.065645],[ 1.         -
0.272616],[ 1.        -1.405794],[ 1.         -
1.972946],[ 1.         -
0.326951],[1.        1.006105],[1.        0.067061],[
1.        -1.298048],[ 1.        -1.634338],[
1.        -0.727636],[1.        0.514267],[
1.        -0.379221]
```

Design Matrix for M=2 is shown here

```
[1.          1.24561     1.55154427],[ 1.          -
1.016511     1.03329461],[1.          0.305252    0.09
317878],[ 1.          -1.104524     1.21997327],[
1.          -
0.785528     0.61705424],[1.          0.486322    0.23
650909],[ 1.          -1.720929     2.96159662],[
1.          -
0.150237     0.02257116],[1.          1.065645    1.13
559927],[ 1.          -0.272616     0.07431948],[
1.          -1.405794     1.97625677],[ 1.          -
1.972946     3.89251592],[ 1.          -
0.326951     0.10689696],[1.          1.006105    1.01
224727],[1.          0.067061    0.00449718],[
1.          -1.298048     1.68492861],[ 1.          -
1.634338     2.6710607],[ 1.          -
0.727636     0.52945415],[1.          0.514267    0.26
447055],[ 1.          -0.379221     0.14380857]
```

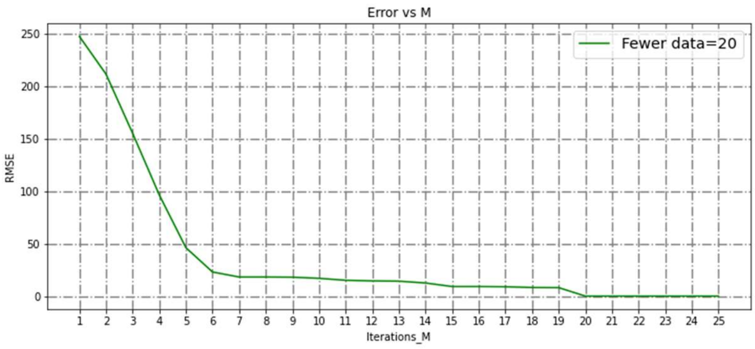
Curve Fitting

After Creating the Design matrix, we can solve the equation and get the coefficients to build the polynomial regression model. Started from 0/1 polynomial degree to M degree.

The First 10 iterations RMSE are shown here for 20 data points taken for training.

```
Root Mean Square Error for M=0:298.8398296781834
Root Mean Square Error for M=1:237.58781279272927
Root Mean Square Error for M=2:231.6178705136389
Root Mean Square Error for M=3:98.3836525965776 Root
Mean Square Error for M=4:95.48674442022723 Root
Mean Square Error for M=5:25.514474057246286 Root
Mean Square Error for M=6:25.51339401238804 Root
Mean Square Error for M=7:19.080837936561306 Root
Mean Square Error for M=8:18.896599506145428 Root
Mean Square Error for M=9:18.44095661320482
```

The Plot for 25-degree polynomials is shown here for the 20 data points taken for the training. Conclusions are very interesting to see.

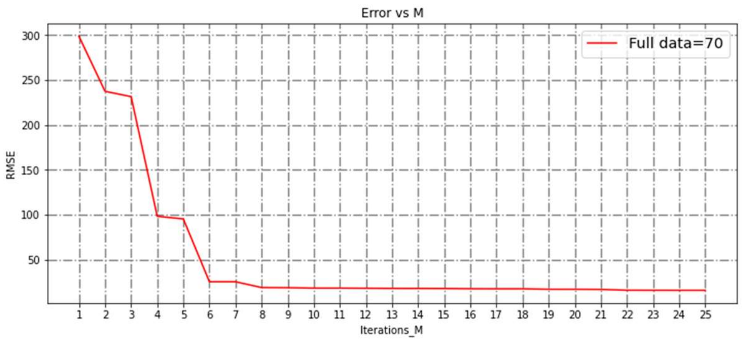


Error vs M for 20 Data Points

The First 10 iterations RMSE are shown here for 70 data points taken for training.

```
Root Mean Square Error for M=0:298.8398296781834
Root Mean Square Error for M=1:237.58781279272927
Root Mean Square Error for M=2:231.6178705136389
Root Mean Square Error for M=3:98.3836525965776 Root
Mean Square Error for M=4:95.48674442022723 Root
Mean Square Error for M=5:25.514474057246286 Root
Mean Square Error for M=6:25.51339401238804 Root
Mean Square Error for M=7:19.080837936561306 Root
Mean Square Error for M=8:18.896599506145428 Root
Mean Square Error for M=9:18.44095661320482
```

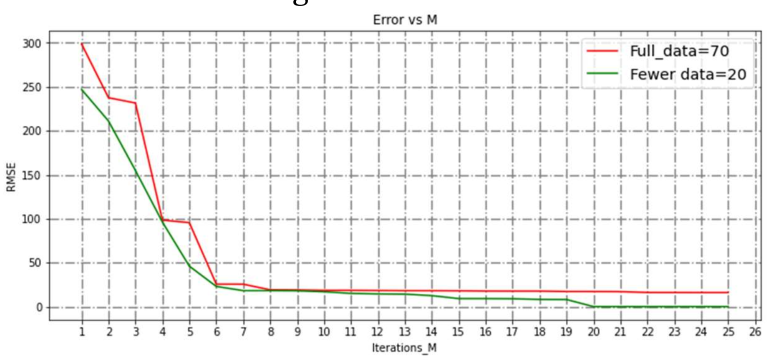
The Plot for 25-degree polynomials is shown here for the 70 data points taken for the training. Conclusions are very interesting to see.



Error vs M for 70 data points

Conclusion

Plotting the Graph together gives a lot of insights.



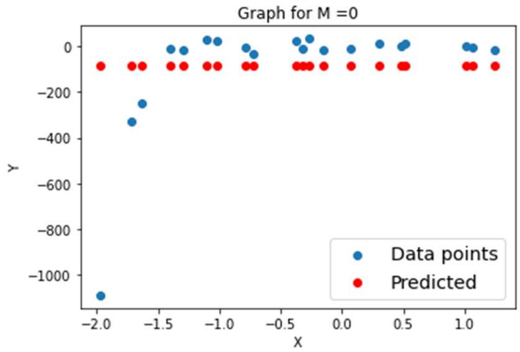
Error vs M for Fewer data and full data

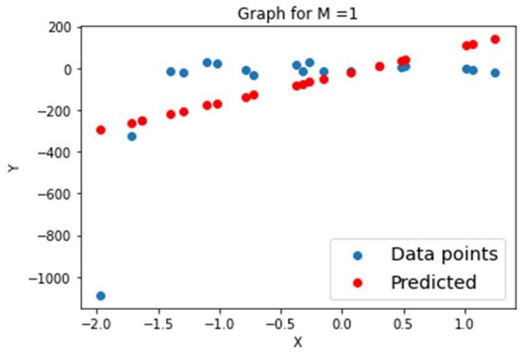
1. We can see for both fewer data (20) and Full data (70) the optimal M value is 6.
2. The Maximum error for more data is more which supposed to be high for more data as then the data will have more chances of being error at early stages.
3. We can observe that fewer data got over fitted at just for M=19 but the more data has saved us at that point.
   Even more data is not overfitted (considering error close to 0) at M=25.

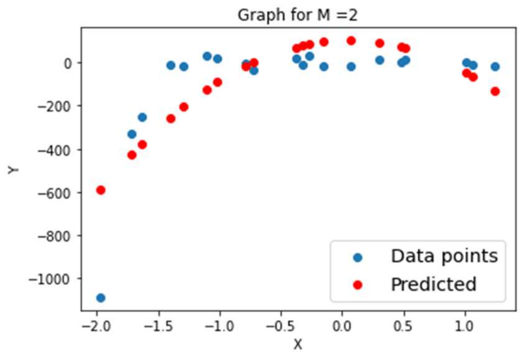
So, we can say that more data can save us from overfitting a model.

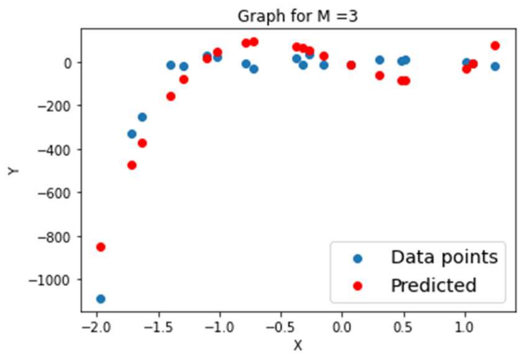
Plots

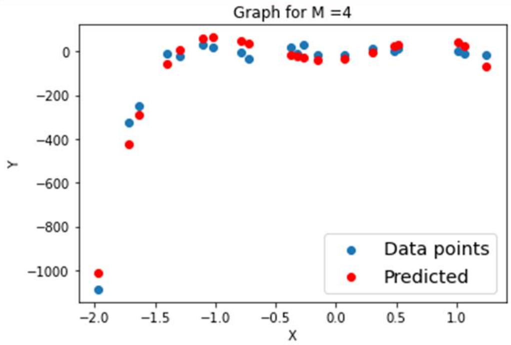
Plots for 20 data points iterations

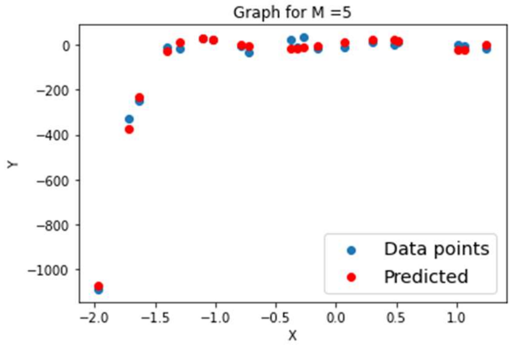


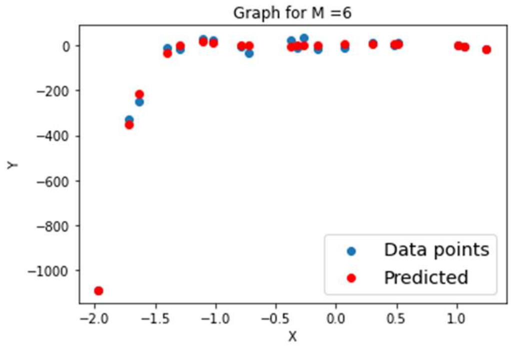


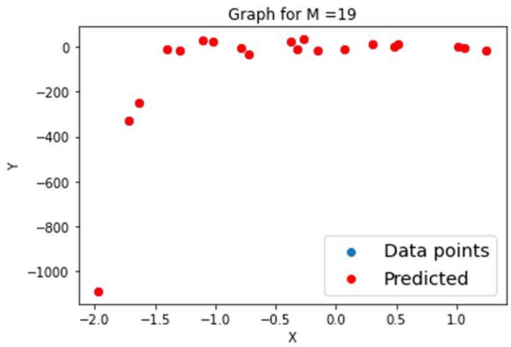








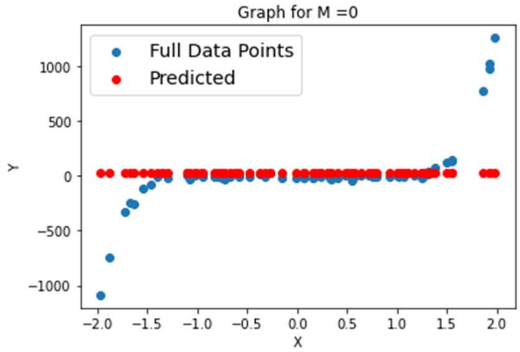


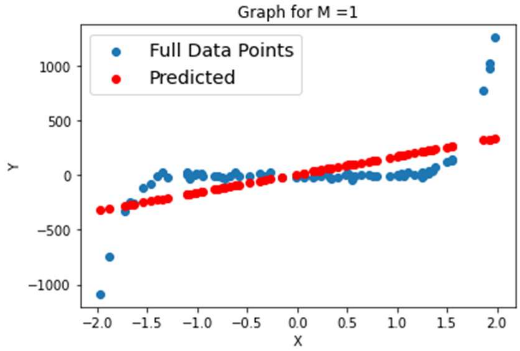


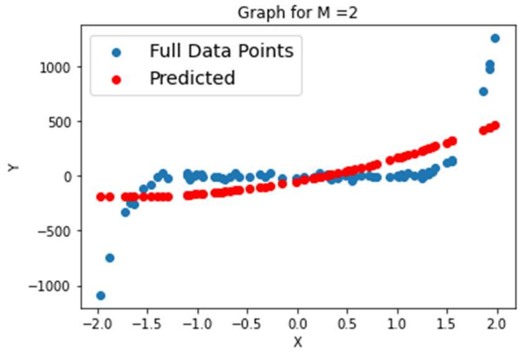
We can see till M=6 model is giving a good fit. Higher M may also give better result, but the computational power needed will be more.
So, choosing M=6 can be good choice.
We can see for M=19 model for 20 data points is getting totally overfit.

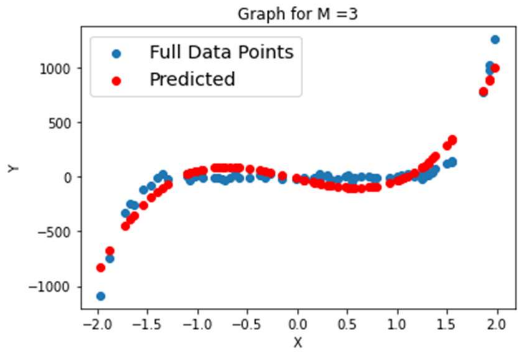
For M=0,1,2,3 we can clearly see the model is showing the underfitting.

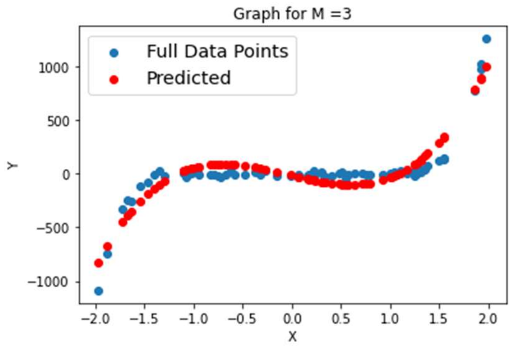
Plots for 70 data points iterations

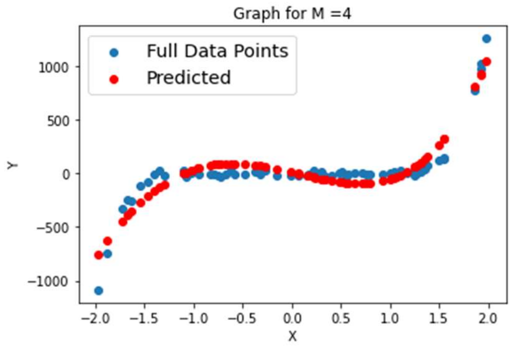


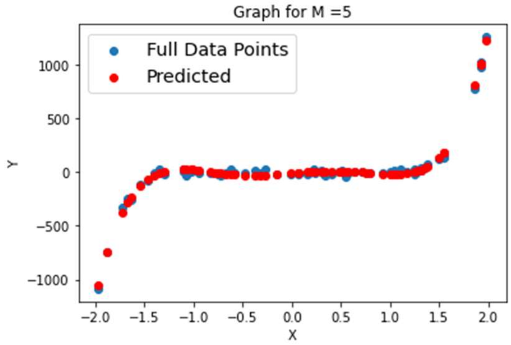


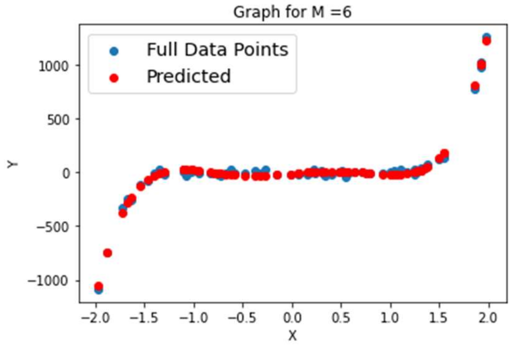


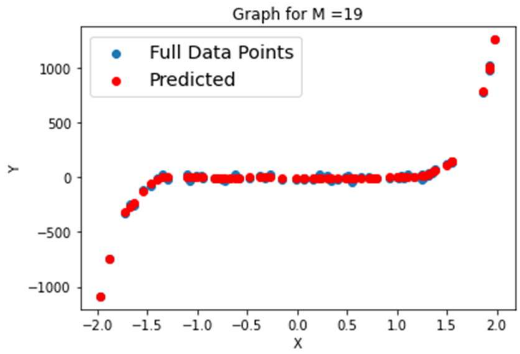












We can see till M=6 model is giving a good fit. Higher M may also give better result, but the computational power needed will be more.
So, choosing M=6 can be good choice.
We can see for M=19 model for 70 data points is not getting overfit.

Coding link is attached [here](#).

Coefficient for Best and worst model

Both the case (20 data points and 70 data points) are giving good results for the polynomial degree 6 however less data points are getting overfit early compared to more data points.

The Coefficients for the 6th degree polynomial are given here for 20 data point case. Corresponding to X^6, X^5....X^0, constant is given below

```
19.48598671
-32.4384702
-36.82352355
60.83358937
9.32876641
-28.66575686
1.3944773 ]
```

The Coefficients for the 6th degree polynomial are given here for 0 data point case. Corresponding to X^6, X^5....X^0, constant is given below

```
1.14116218
10.62512673
0.25552779
-2.87813669
-21.87741547
1.21642023
15.54965251]
```

So, we can see and compare the coefficients for less data are high.

The Coefficients for the 19th degree polynomial are given here for 20 data point case. Corresponding to X^6, X^5....X^0, constant is given below

```
1.89838006e+03  1.89860233e+03  5.29050852e+04  8.23409
178e+04
 -9.07753437e+05 -
2.28226954e+06  4.74268294e+06  1.69995519e+07
 -4.82488169e+06 -5.23357147e+07 -
2.42826626e+07  6.84456009e+07
  6.48568625e+07 -3.21481783e+07 -5.91758988e+07 -
5.86983714e+06
  2.17355273e+07  8.89447424e+06 -1.90631899e+06 -
1.79301770e+06
 -2.95161580e+05
```

The Coefficients for the 19th degree polynomial are given here for 70 data point case. Corresponding to X^6, X^5....X^0, constant is given below

```
 0.00000000e+00 -1.46777661e+01  1.42125328e+02 -
1.37632939e+02
 -5.24993917e+02  9.71663726e+02  4.24941255e+00 -
2.16817196e+03
  2.47790028e+03  2.52024805e+03 -4.61082676e+03 -
1.78189342e+03
  3.98741644e+03  8.00475188e+02 -1.91655946e+03 -
2.21190935e+02
  5.24313345e+02  3.39278727e+01 -7.63274641e+01 -
2.19158723e+00
  4.58815342e+00
```

So, we can again see that less data points in the overfitting case have overshooted constants.

Code Link-
https://github.com/KAJURAMBO/Assignment_intro-to-ml_ell784