

INTRODUCTION TO MACHINE LEARNING

Assignment 1-2021AMA2090(Rahman Alam)

Contents

Non-AIP701-PART 1A	2
1. Scikit Learn Library.....	2
2. Design Matrix.....	2
3. Curve Fitting (Moore-Penrose).....	2
4. Plots.....	4
5. Coefficient for Best and worst model and Noise variance	6
6. Conclusion-1A.....	7
7. Gradient Descent Algorithm	8
Non-AIP701-PART 1B	10
8. Design Matrix	10
9. Curve Fitting (Moore-Penrose).....	11
10. Plots.....	12
11. Coefficient for best and worst model and Noise Variance.....	15
12. Conclusion	16
13. Gradient Descent Algorithm 1B.....	16

NON-AIP701-PART 1A

1. Scikit Learn Library

i 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*.

2. Design Matrix

i 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 = 1)
X_poly = polynomial_features.fit_transform(x_train)
design_matrix_20.append(X_poly)
```

Design Matrix for M=1 is shown here for 20 data points

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 for 20 data points

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			

3. Curve Fitting (Moore-Penrose)

i 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=0:247.0541643191767

Root Mean Square Error for M=1:211.12447248939546

Root Mean Square Error for M=2:154.8615544456756

Root Mean Square Error for M=3:96.24902022347398

Root Mean Square Error for M=4:46.02383183717053

Root Mean Square Error for M=5:22.935950053185138

Root Mean Square Error for M=6:18.149825822102095

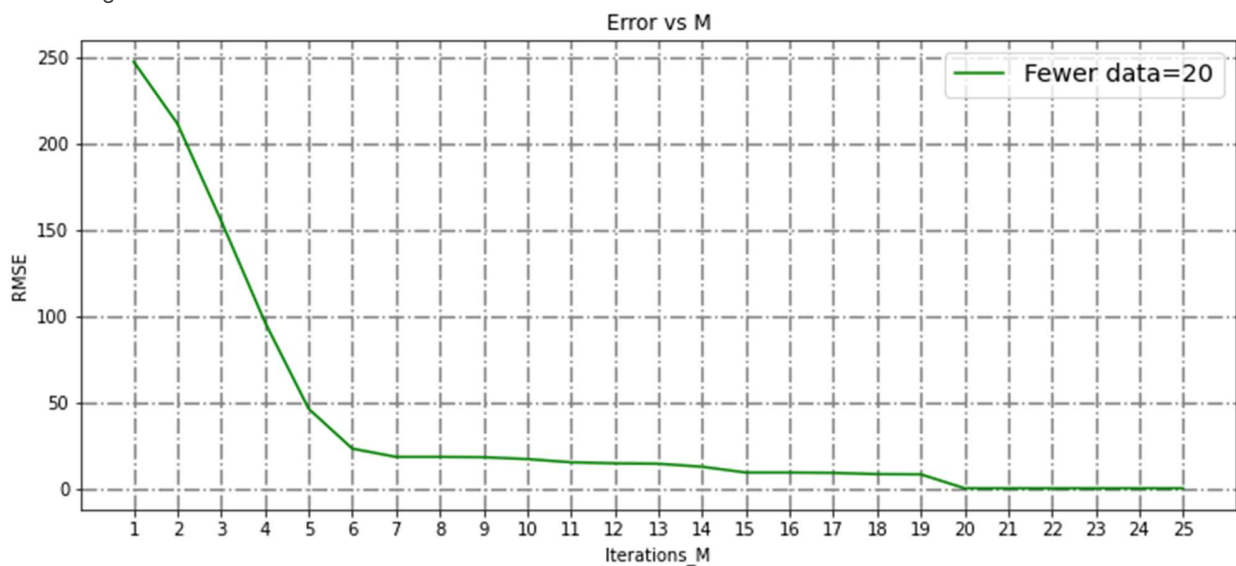
Root Mean Square Error for M=7:18.145050251430355

Root Mean Square Error for M=8:17.97025665371971

Root Mean Square Error for M=9:16.88613992844558

Root Mean Square Error for M=10:15.031923356337714

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

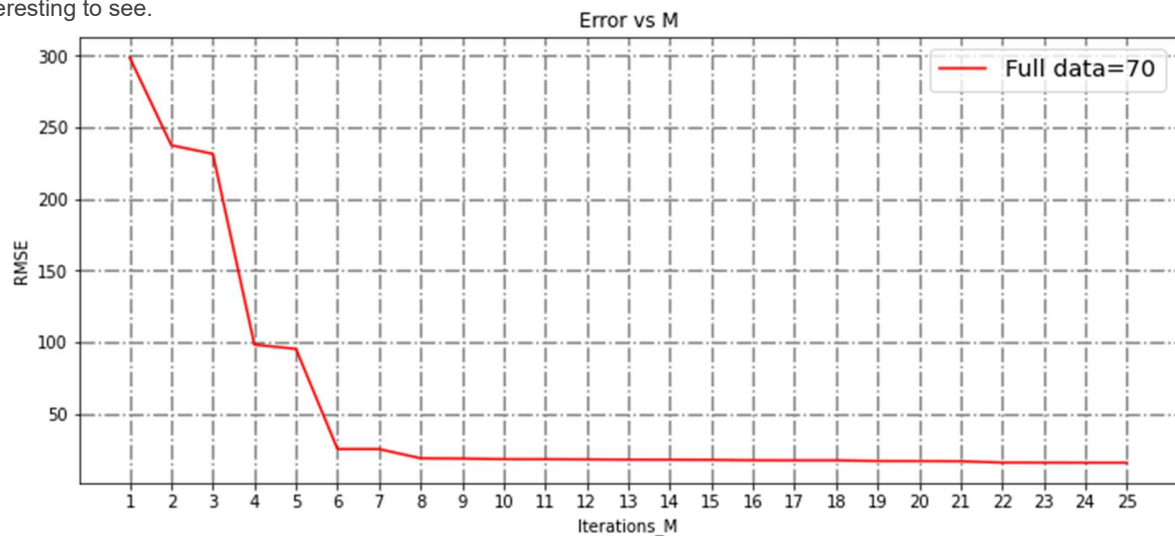
Root Mean Square Error for M=7:19.080837936561306

Root Mean Square Error for M=8:18.896599506145428

Root Mean Square Error for M=9:18.44095661320482


Root Mean Square Error for M=10:18.426066987822978

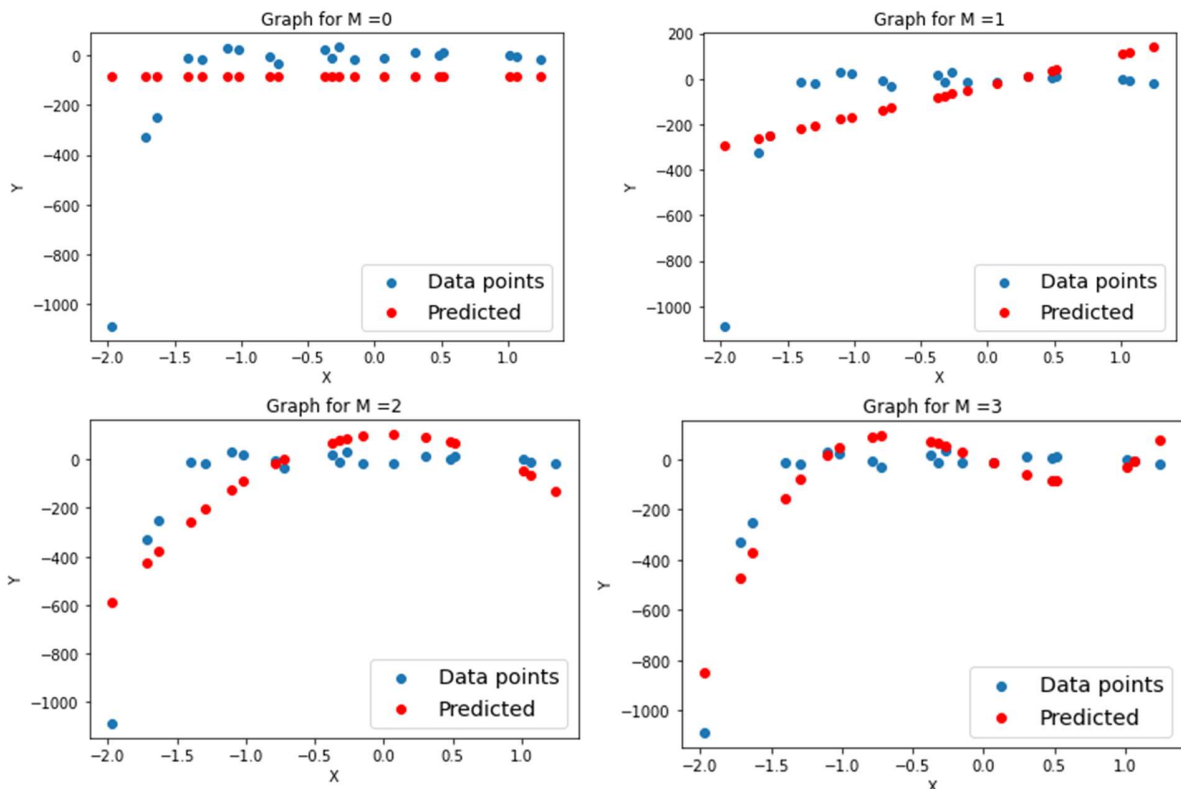
The Plot for 25-degree polynomials is shown here for the 70 data points taken for the training. Conclusions are very interesting to see.

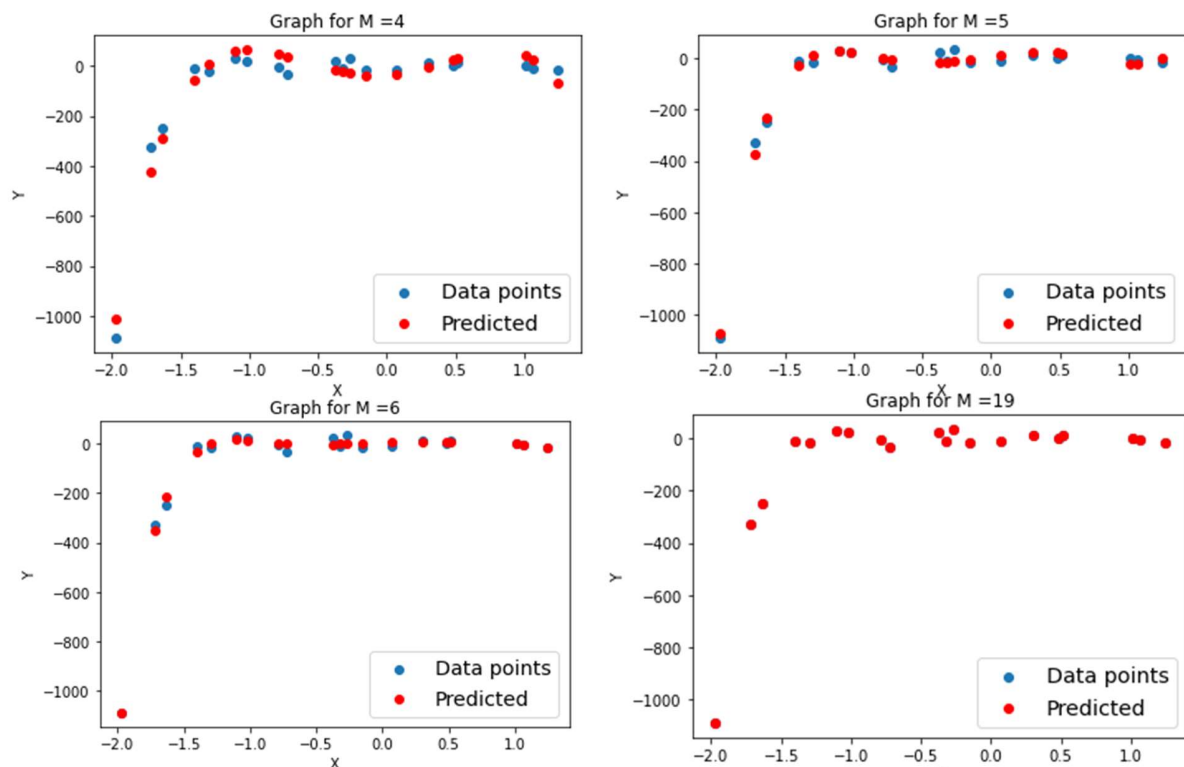


Error vs M for 70 data points

4. Plots

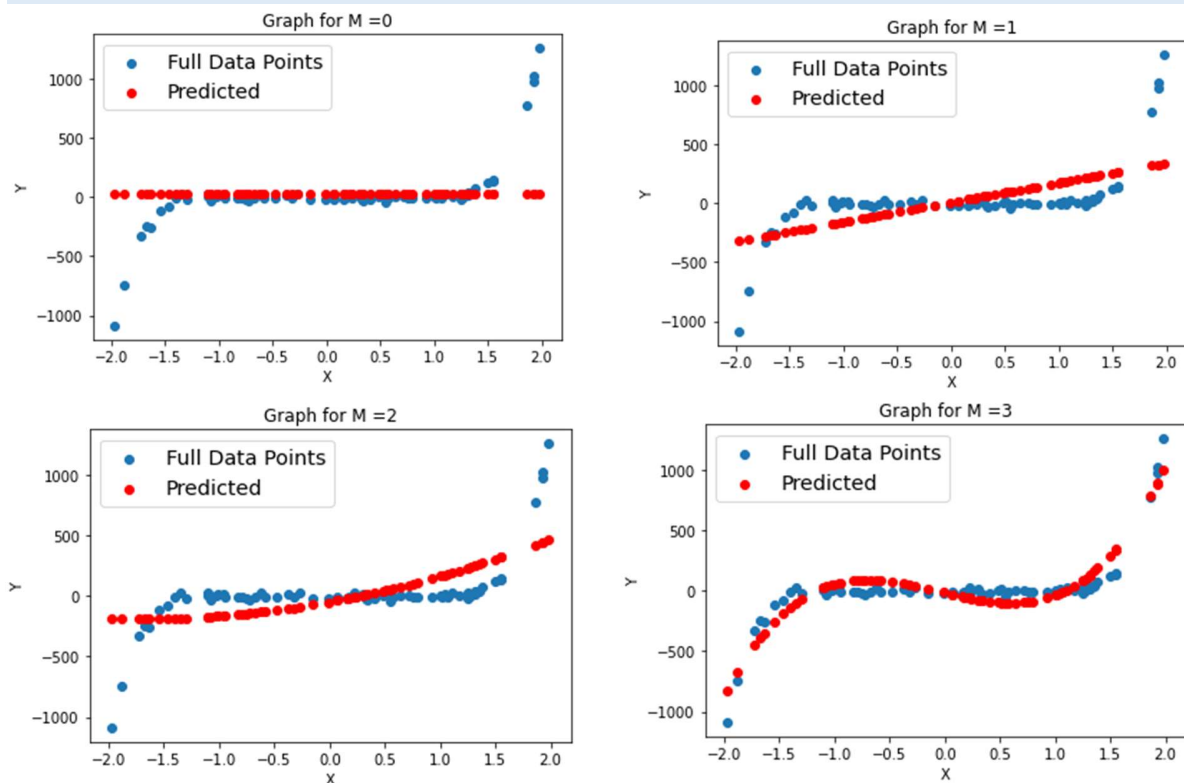
 Plots for 20 data points iterations

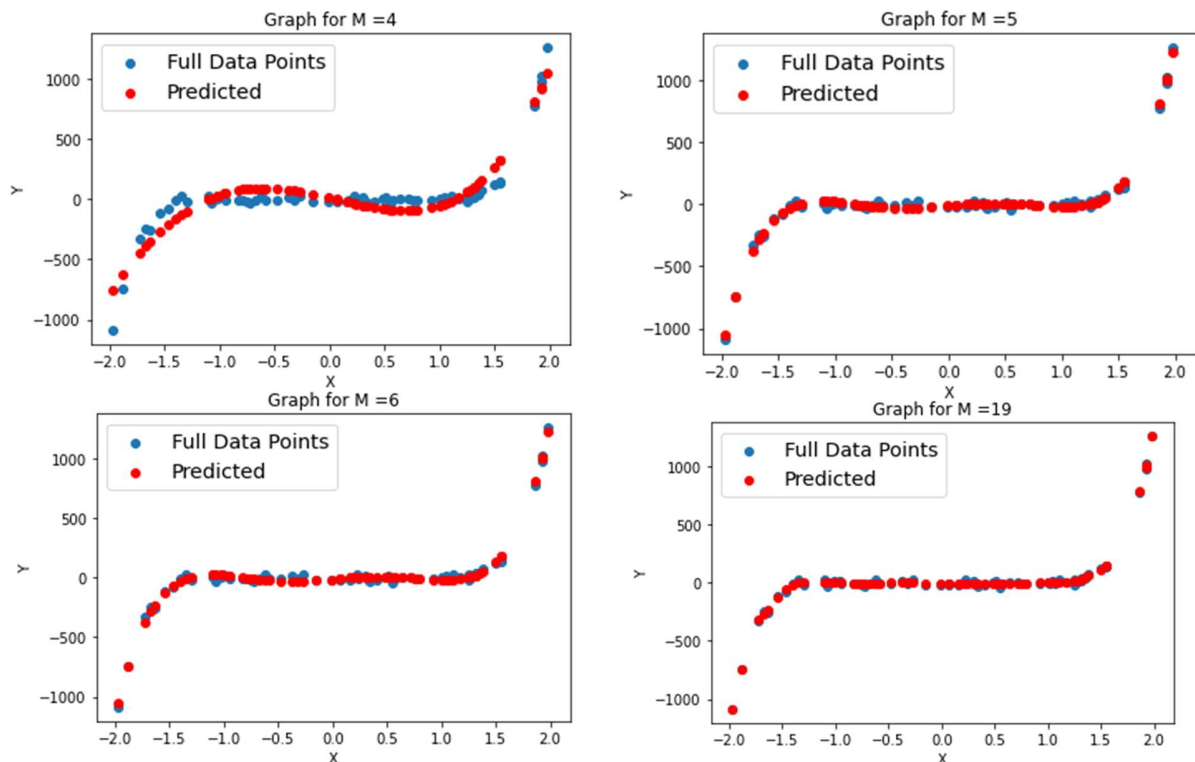




We can see till $M=6$ model is giving a good fit. Higher M may also give better result, but the computational power needed will be more. So, choosing $M=6$ can be good choice. We can see for $M=19$ model for 20 data points is getting totally overfit. For $M=0,1,2,3$ we can clearly see the model is showing 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.

5. Coefficient for Best and worst model and Noise variance

i 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, \dots, 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 70 data point case. Corresponding to X^6, X^5, \dots, X^0 , constant is given below

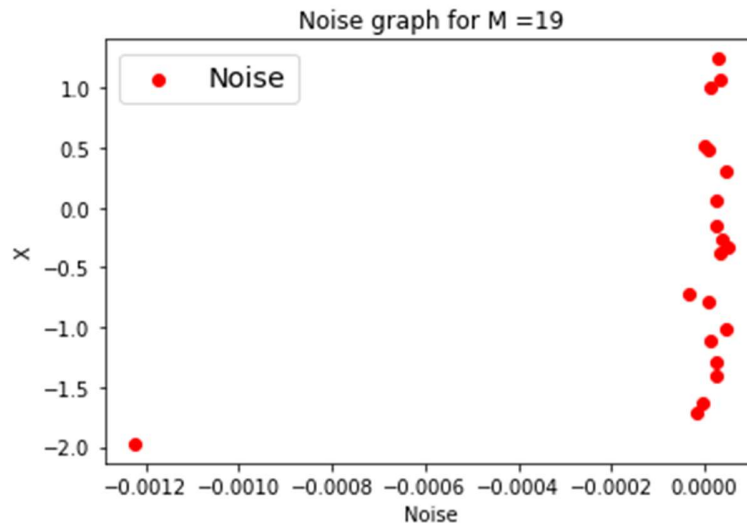
[1.14116218 , 10.62512673, 0.25552779, -2.87813669, -21.87741547, 1.21642023, 15.54965251]

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, \dots, X^0 , constant is given below. And the noise variance for this case is [7.5767451e-08].

```
1.89838006e+03 1.89860233e+03 5.29050852e+04 8.23409178e+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
```

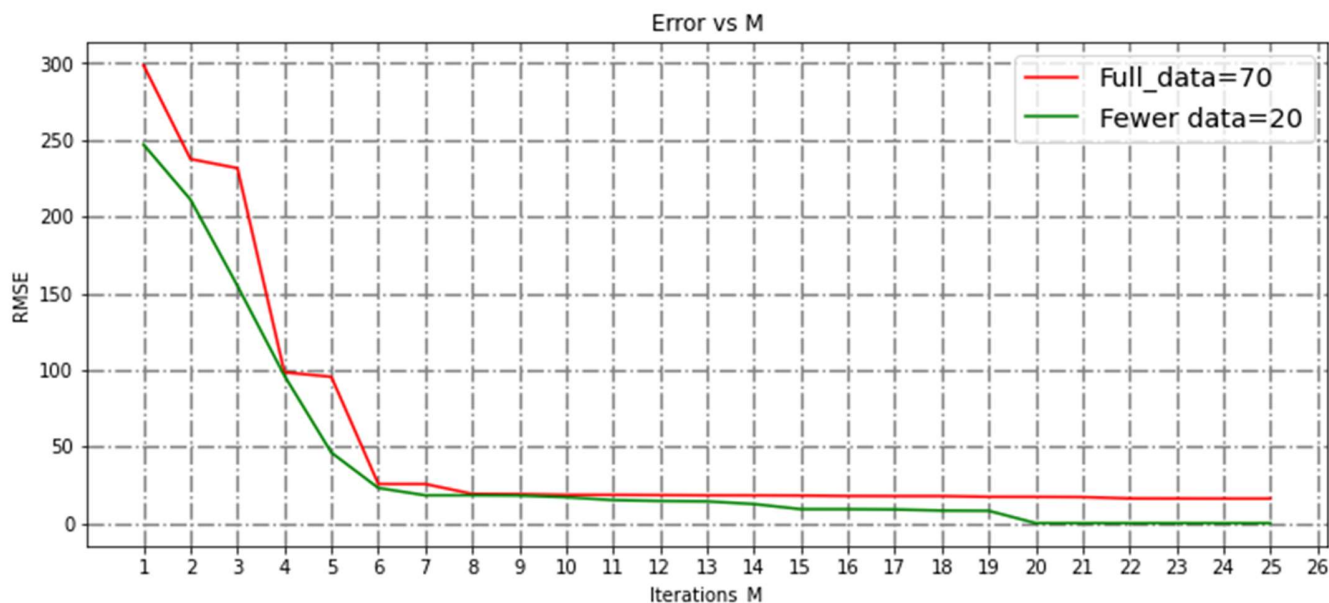
So, we can again see that less data points in the overfitting case have overshooted constants.

6. Conclusion-1A

Plotting the Graph together gives a lot of insights

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 $M=25$.

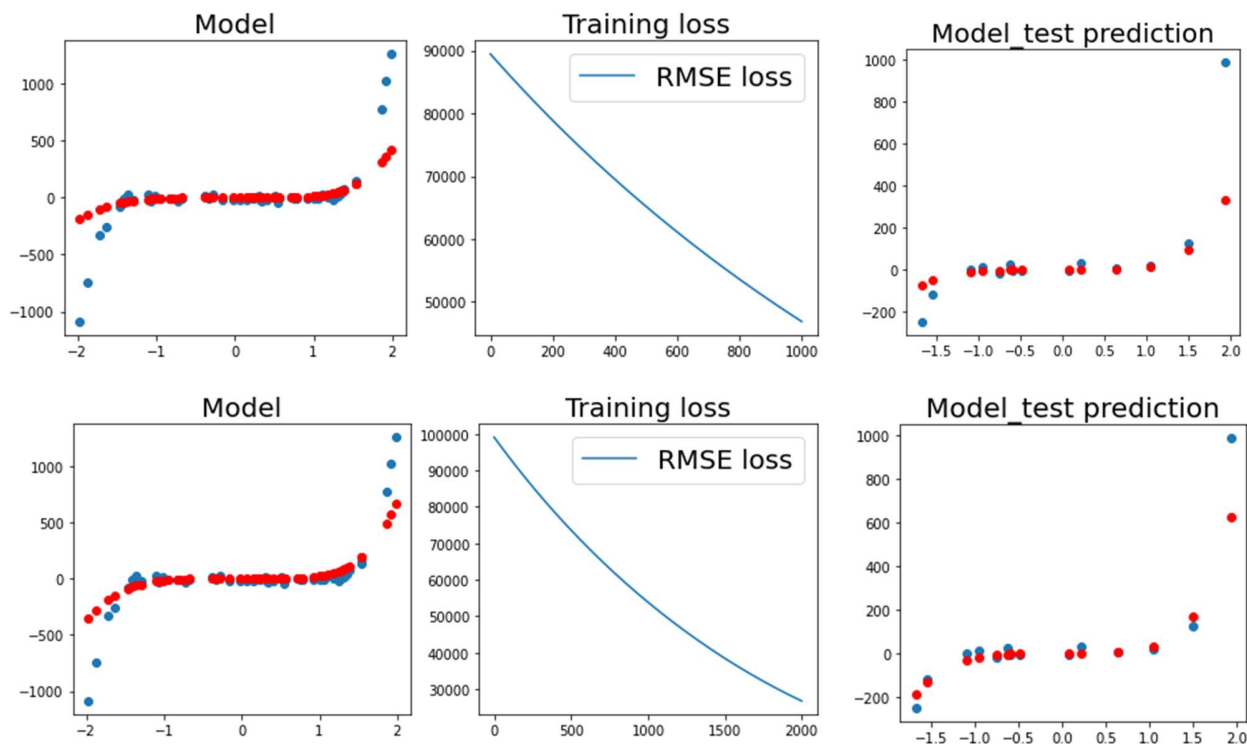
So, we can say that more data can save us from overfitting a model.

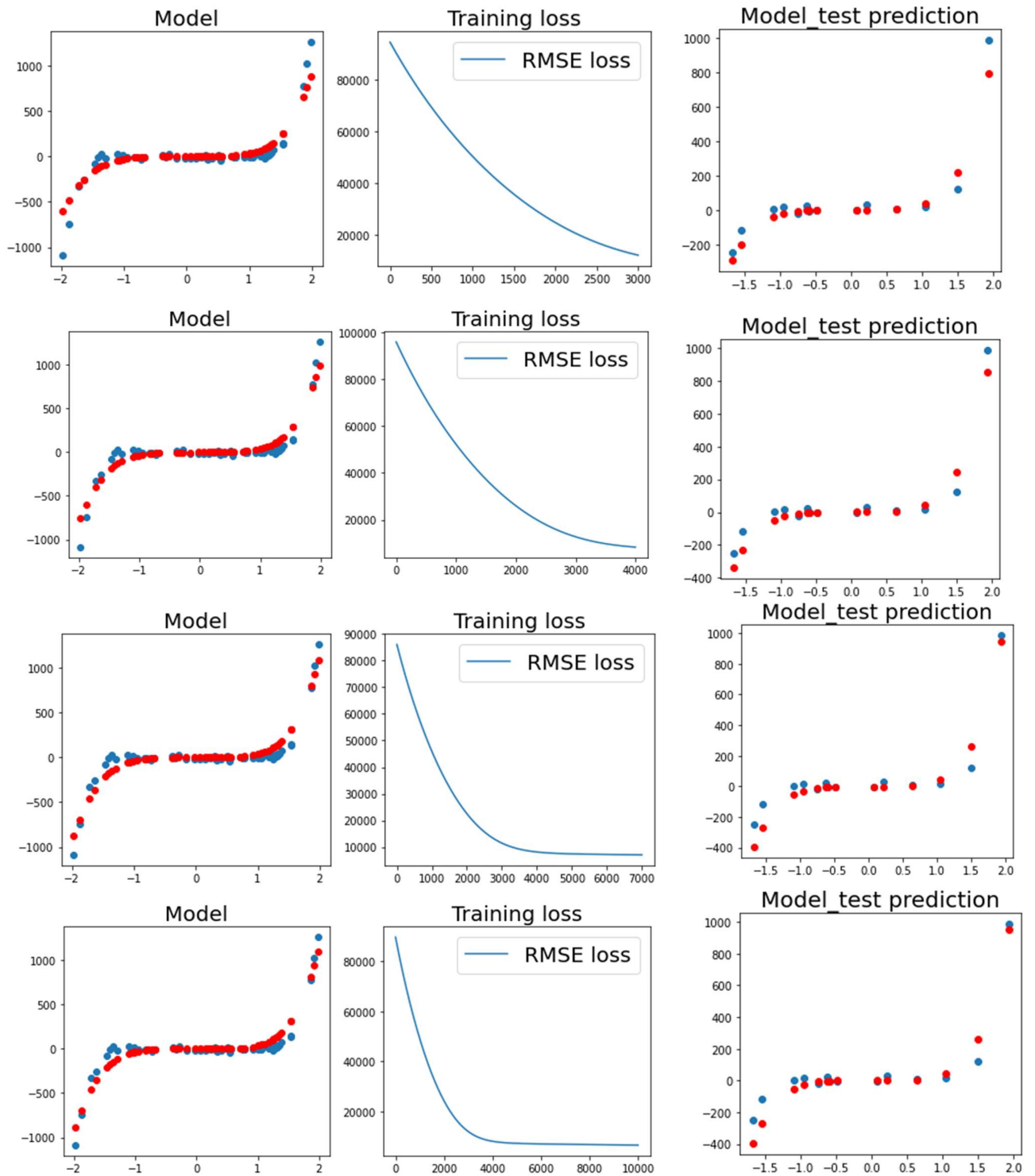


7. Gradient Descent Algorithm

- Stochastic gradient descent

Here we are updating the weights for every sample, so the convergence is noisy but computationally effective. We are using MSE as our loss function and a function defined to give a Mth order polynomial. No regularization is used. I have shown plots for 1000, 2000,...10000 epochs and we can see that training loss is almost **constant after 4000** epochs so we can just **stop till 4000 epochs**. Learning rate I have taken as 0.001. test data with 30 points have been used to test the models after every 1000 epochs. M=6 has chosen here for SGD. So more epoch are helping reducing error but upto 4000 epochs. Plots are shown below.

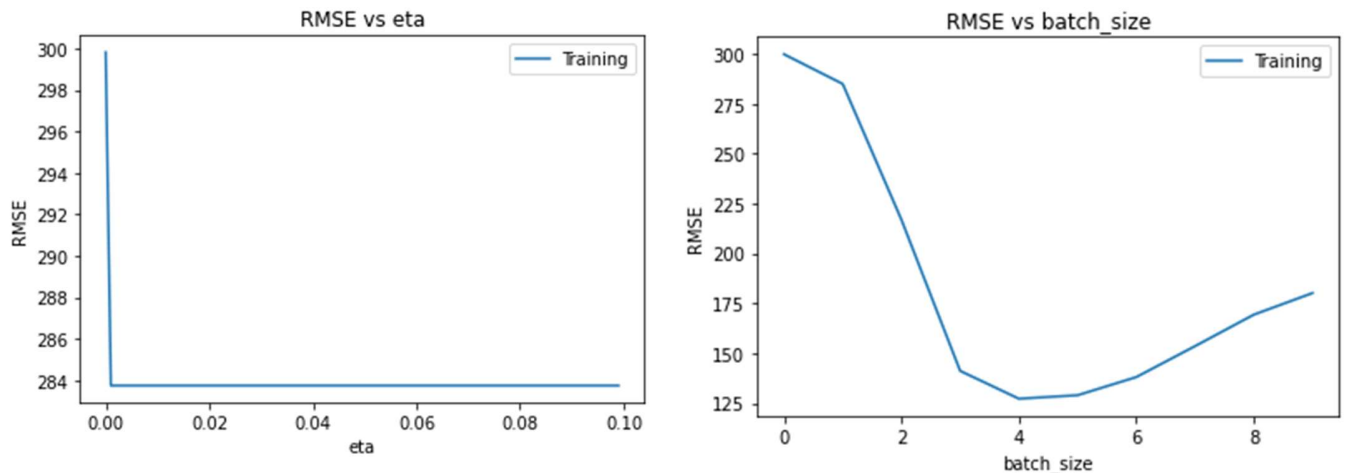




- Mini Batch Gradient Descent

Here we will be updating the weights after few samples are passed or you can say after a handful of samples called mini batches is passed. It will help us smoothly enter towards the optimal points/weights.

Model is trained for 6th order polynomial



So as above graph suggest to pick the learning rate at 0.001 and a optimal batch size to be 4 as our data set is onjly having 70 data points for training.

Weights for the 6th order polynomial for 0.001 learning rate,4 batch size and 4000 epoch with 70 data points are as below.

2.25212651 , -10.81834511 , 10.53234425 , -4.54626229 , 7.38617364 , 5.95717802 , -1.94808232

NON-AIP701-PART 1B

8. Design Matrix

Design Matrix for M=1 is shown here for 20 data points

[0.246119	0.06057456],	[0.87242	0.76111666],	[0.2192	0.04804864],	[1.809242	3.27335661],	[
-0.923474	0.85280423],	[1.006063	1.01216276],	[-1.41848	2.01208551],	[-		
1.24184	1.54216659],	[-1.278768	1.6352476],	[-					
0.32844	0.10787283],	[0.599934	0.3599208],	[1.248036	1.55759386],	[1.645741	2.70846344],	[
0.9818293],	[1.	1.681223	2.82651078],	[1.	0.242633	0.05887077],	[1.38482	1.91772643],	[
-1.858806	3.45515975],	[-0.559941	0.31353392],	[-1.318246	1.73777252]				

Design Matrix for M=2 is shown here for 20 data points

[0.246119	0.06057456	0.01490855],	[0.87242	0.76111666	0.66401339],	[0.2192	0.04804864	
0.01053226],	[1.809242	3.27335661	5.92229427],	[-0.923474	0.85280423	-			
0.78754253],	[1.006063	1.01216276	1.0182995],	[-1.41848	2.01208551	-2.85410305],	[-
1.24184	1.54216659	-1.91512415],	[-1.278768	1.6352476	-2.0911023],	[-0.32844	0.10787283	-	
0.03542975],	[0.599934	0.3599208	0.21592873],	[1.248036	1.55759386				
1.94393321],	[1.645741	2.70846344								
4.45742933],	[0.990873	0.9818293	0.97286815],	[1.681223	2.82651078				
4.75199493],	[0.242633	0.05887077	0.01428399],	[1.38482	1.91772643	2.65570592],	[-	
1.858806	3.45515975	-6.42247167],	[-0.559941	0.31353392	-0.1755605],	[-1.318246	1.73777252	-
2.29081167]											

9. Curve Fitting (Moore-Penrose)

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

Root Mean Square Error for M=0:55.62287520856701

Root Mean Square Error for M=1:53.77283866902993

Root Mean Square Error for M=2:28.491239299489813

Root Mean Square Error for M=3:20.993855995451245

Root Mean Square Error for M=4:6.025820376031074

Root Mean Square Error for M=5:5.2297372513875064

Root Mean Square Error for M=6:5.089281088633635

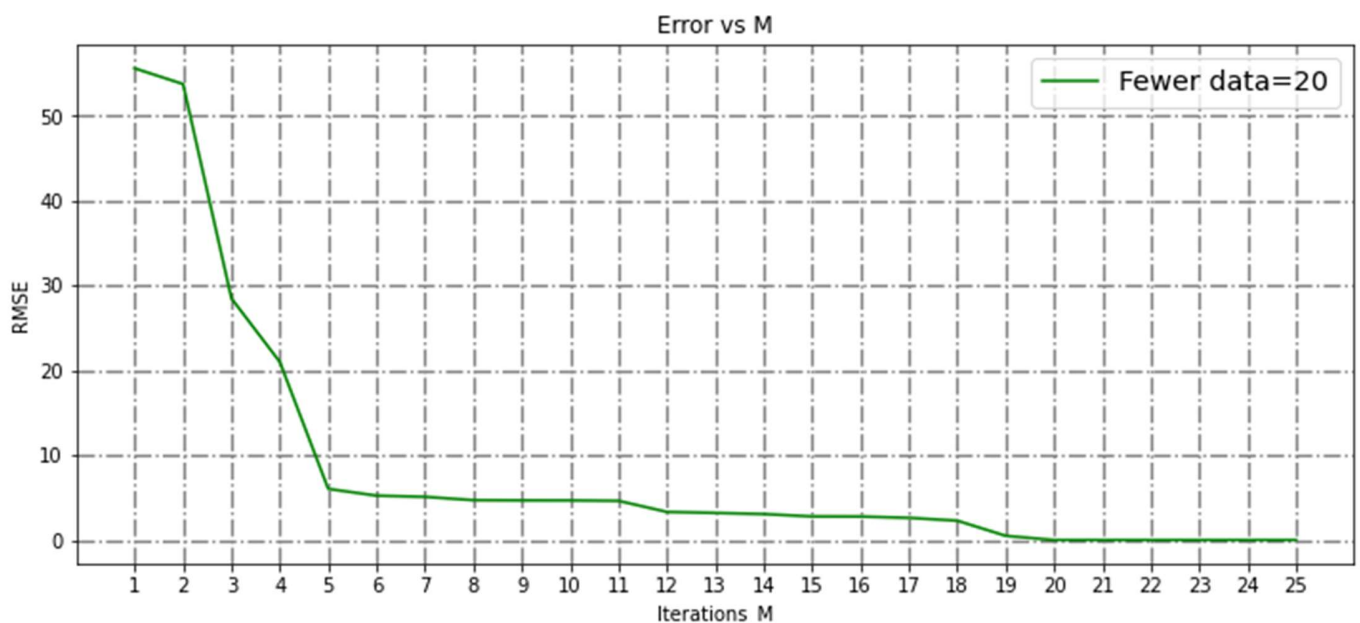
Root Mean Square Error for M=7:4.698322039383037

Root Mean Square Error for M=8:4.676583516582949

Root Mean Square Error for M=9:4.669799870651431

Root Mean Square Error for M=10:4.616500904873091

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=0:70.09721538978584

Root Mean Square Error for M=1:67.22070491028734

Root Mean Square Error for M=2:33.696388497087085

Root Mean Square Error for M=3:25.181823338879145

Root Mean Square Error for M=4:8.752391253783841

Root Mean Square Error for M=5:8.455902961555978

Root Mean Square Error for M=6:7.842903432512381

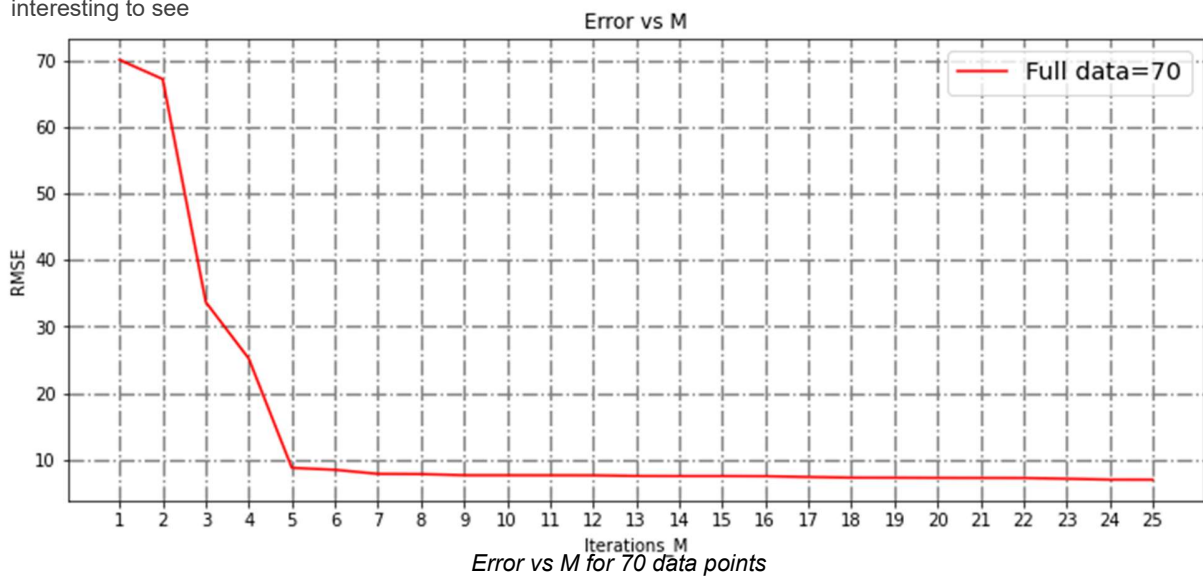
Root Mean Square Error for M=7:7.812639922861118

Root Mean Square Error for M=8:7.638369766057252

Root Mean Square Error for M=9:7.636777019339523

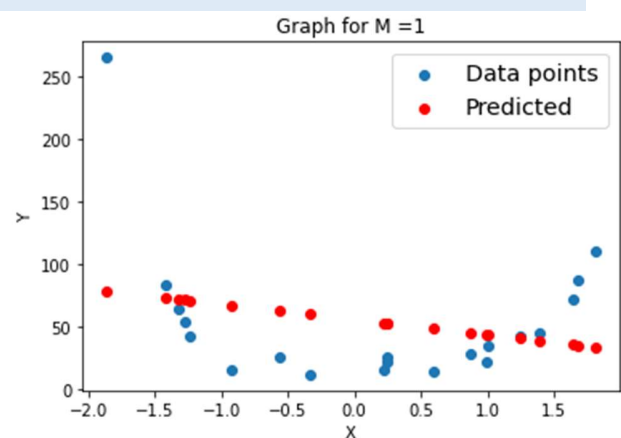
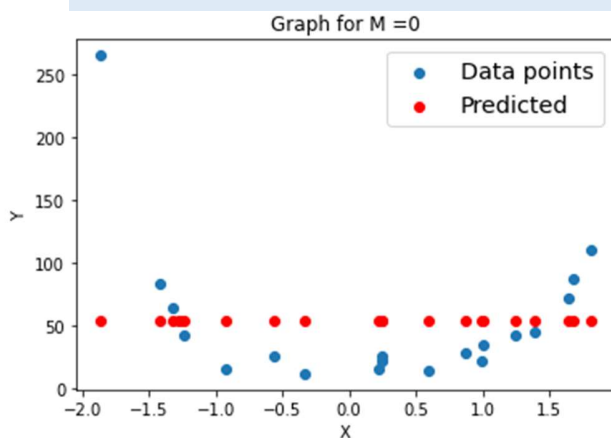
Root Mean Square Error for M=10:7.62929718421274

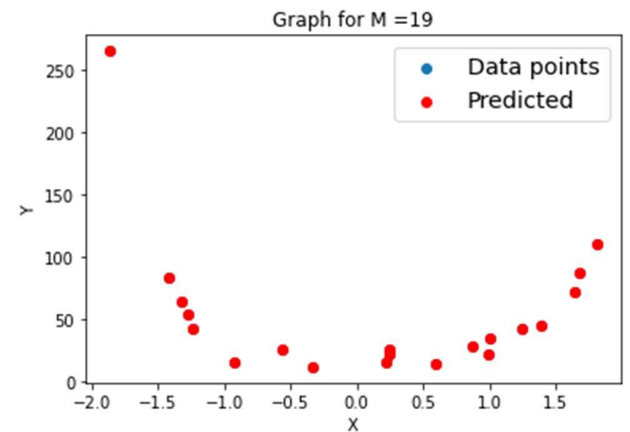
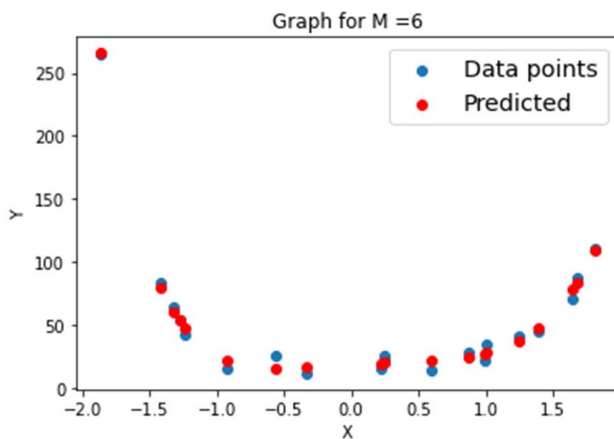
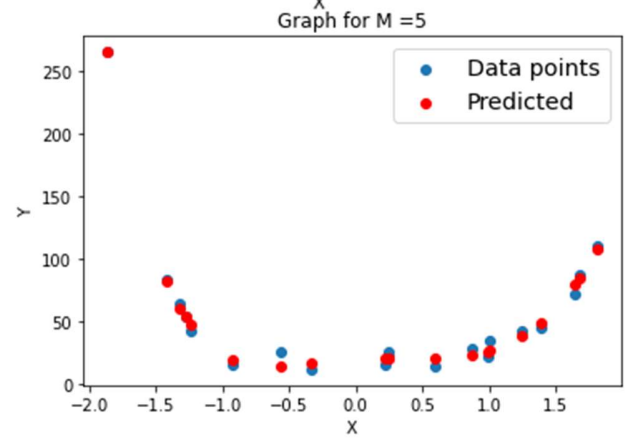
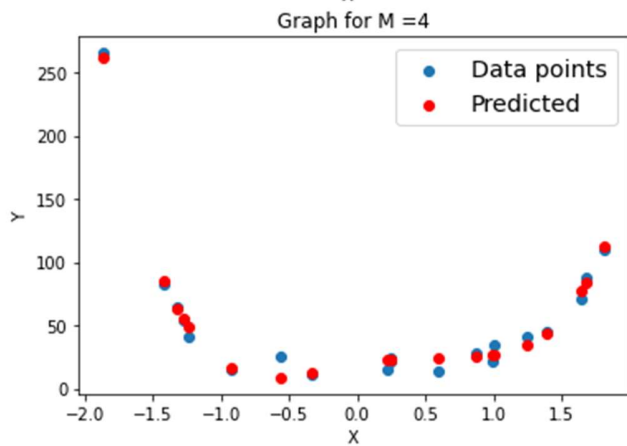
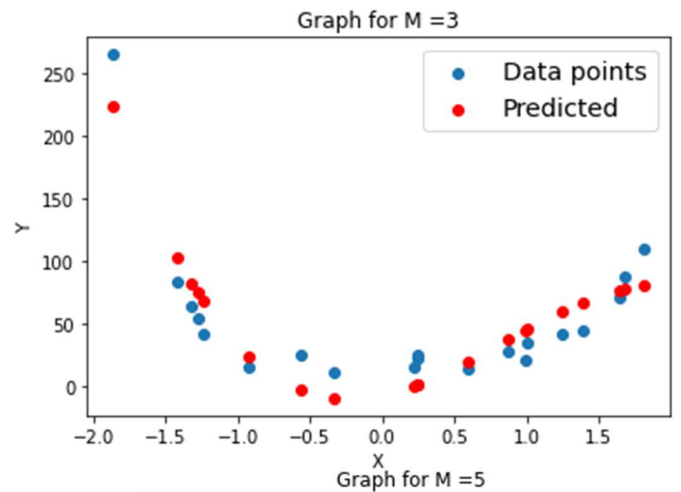
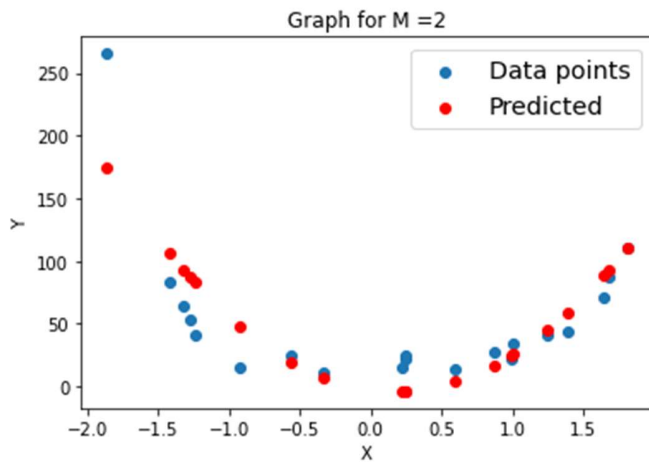
The Plot for 25-degree polynomials is shown here for the 70 data points taken for the training. Conclusions are very interesting to see



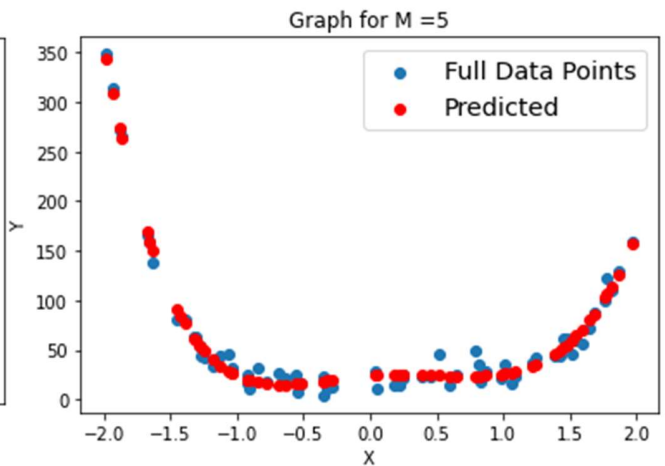
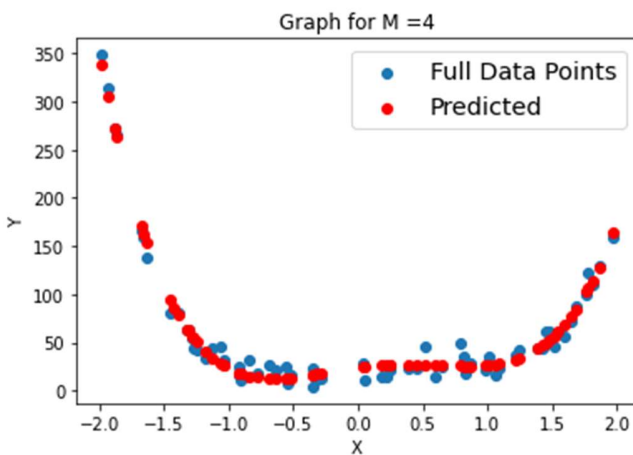
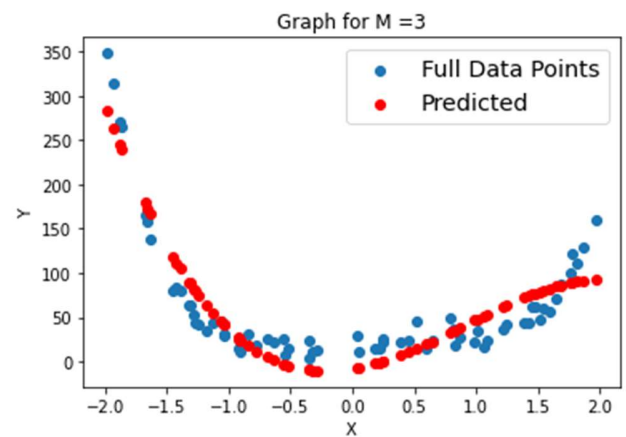
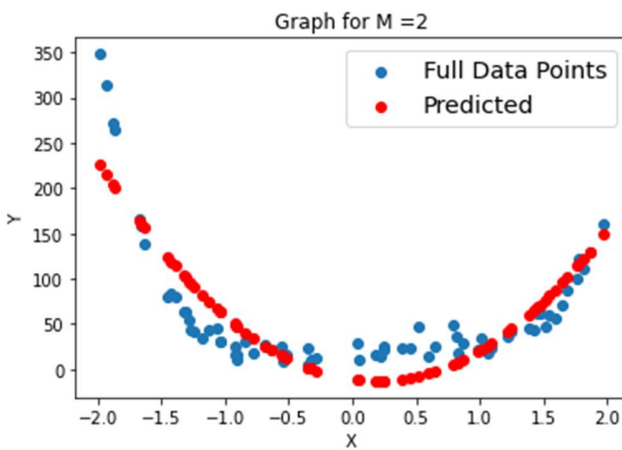
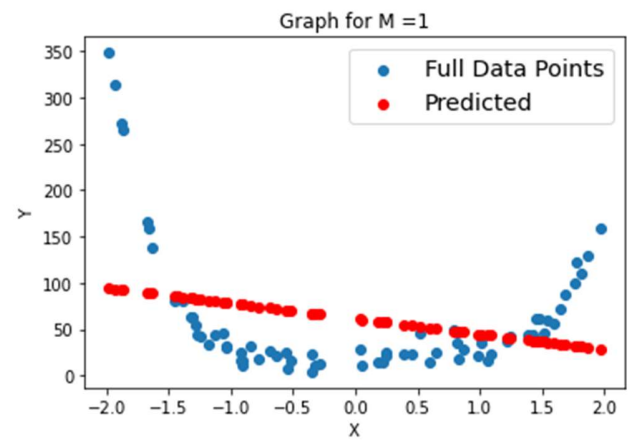
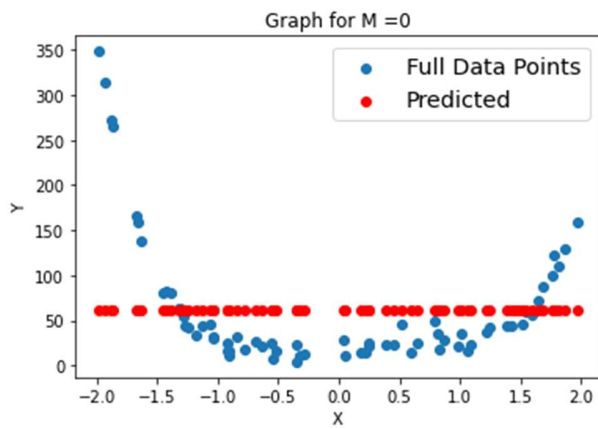
10. Plots

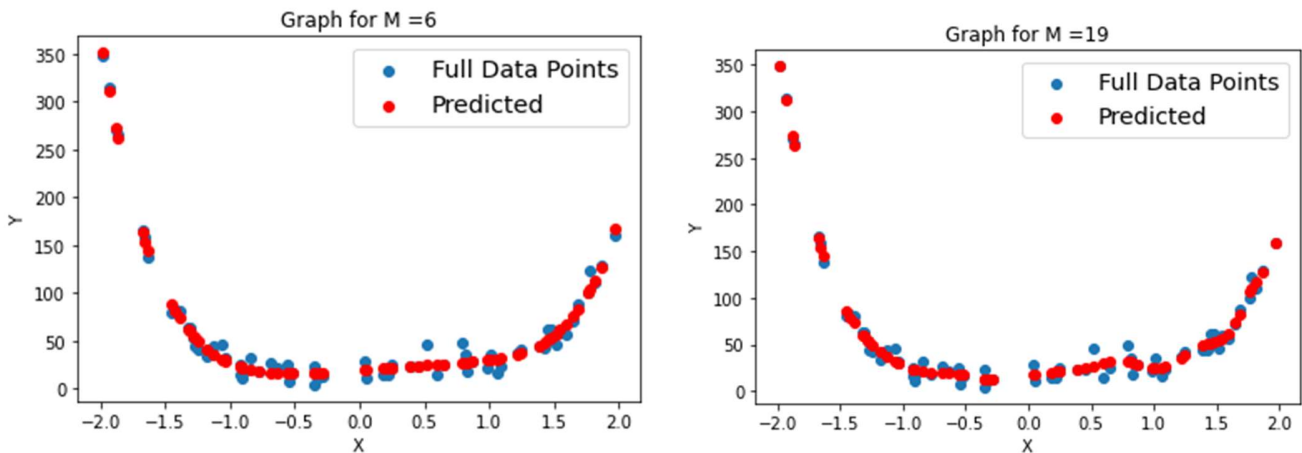
i Plots for 20 data points iterations





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. so for the in 1A and 1B model is getting overfit at M=19 for fewer (20) data points.





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.

11. Coefficient for best and worst model and Noise Variance

i Both the case (20 data points and 70 data points) is 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

-4.45736087 -13.12884113 26.66689349 17.32522203 -21.60795539 0.36253482 3.23142133

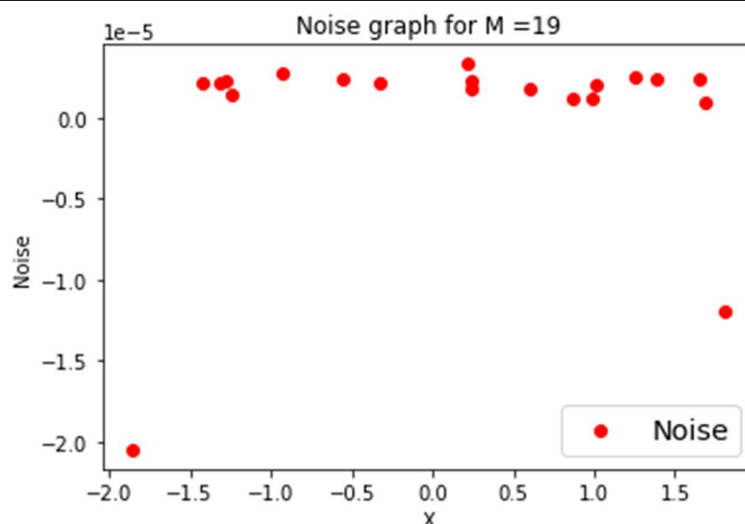
The Coefficients for the 6th degree polynomial are given here for 70 data point case. Corresponding to X^6 , X^5 , ..., X^0 , constant is given below

13.53666303 1.38512963 -14.09892717 4.92232874 1.92625672 2.70603127 -0.58152941

So, we can see and compare the coefficients for less data are high. Resulting early overfit for fewer data.

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. And the noise variance for this case is $[3.21873423e-11]$.

0	23445.74407486	21340.3024207	-378379.81275263
241346.22046543	1817390.01978246	-1969987.81349591	-3693079.60571943
5147336.39235743	3590483.98839817	-6643345.86394168	-1478845.35574118
4758062.17249966	-115855.53950881	-1922688.52994446	330160.90240019
408914.49765016	-109282.22189523	-35369.66056063	11746.49232729



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

```
0.0000000e+00 2.82510067e+01 -3.06776908e+01 -2.13878112e+02
4.13728110e+02 9.13341806e+02 -1.16099300e+03 -1.90363058e+03
1.49306462e+03 2.12229516e+03 -1.03810150e+03 -1.36974022e+03
4.17399713e+02 5.27792990e+02 -9.70140376e+01 -1.19701269e+0
1.20982340e+01 1.47340123e+01 -6.26388966e-01 -7.58955625e-01
```

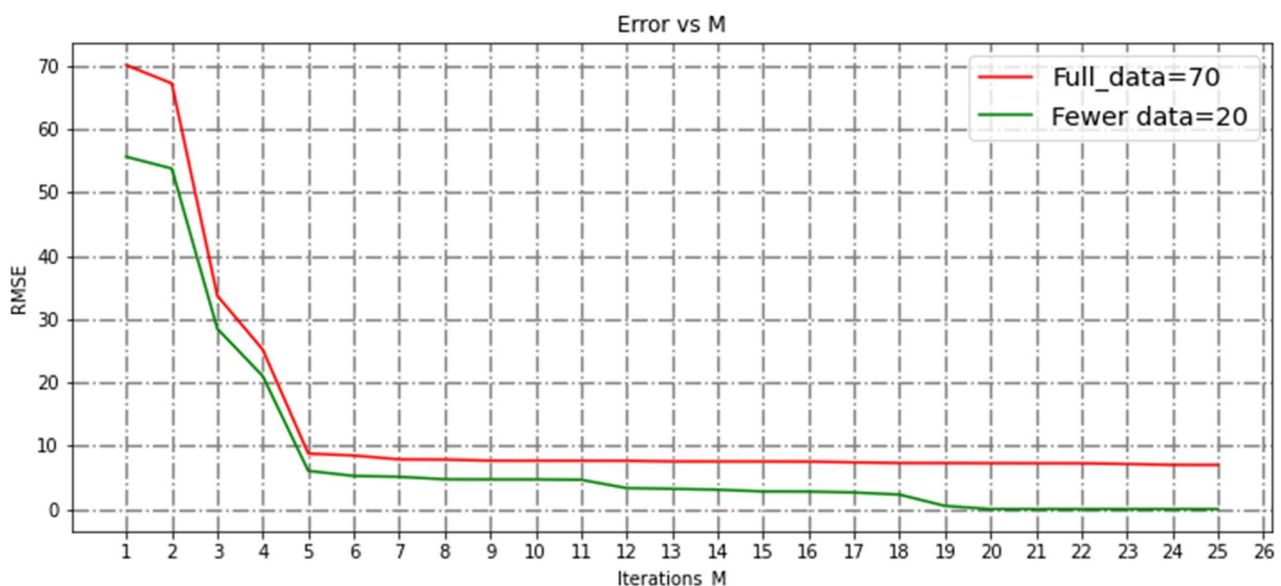
So, we can again see that less data points in the overfitting case have overshooted constants. Resulting early overfitting.

12. Conclusion

Plotting the Graph together gives a lot of insights

1. We can see for both fewer data (20) and Full data (70) the optimal M value is 6. Observed in both 1A and 1B.
 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. Observed in both 1A and 1B
 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. Observed in both 1A and 1B.
- 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.



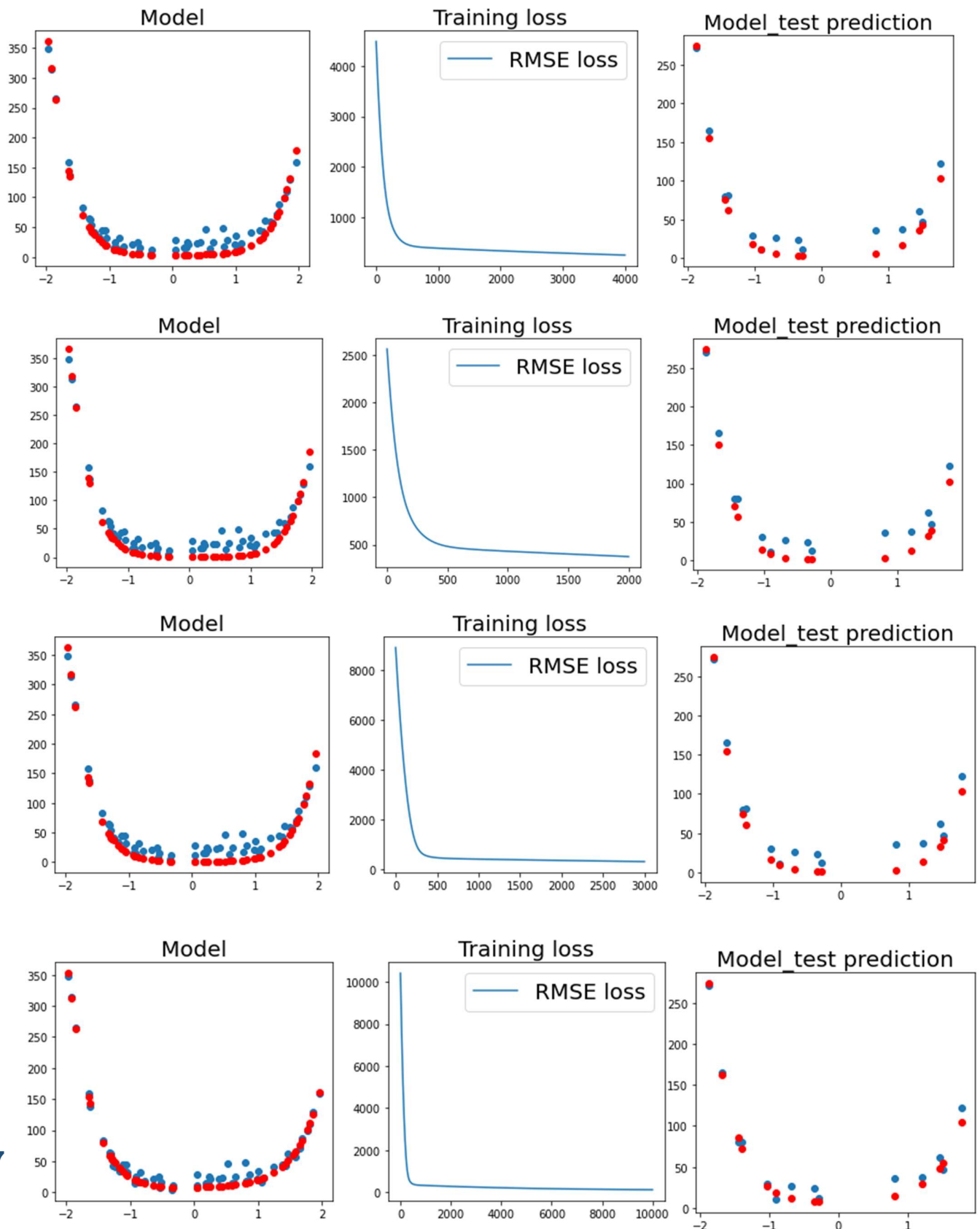
13. Gradient Descent Algorithm 1B

- Stochastic gradient descent

Here we are updating the weights for every sample, so the convergence is noisy but computationally effective. We are using MSE as our loss function and a function defined to give a M th order polynomial. No regularization is used. I have shown plots for 1000, 2000, ..., 10000 epochs and we can see that training loss is almost **constant after 400 epochs** so we can just **stop till 400 epochs**. Learning rate I have taken as 0.001. Train data with 56 points are used to train the model. Test data



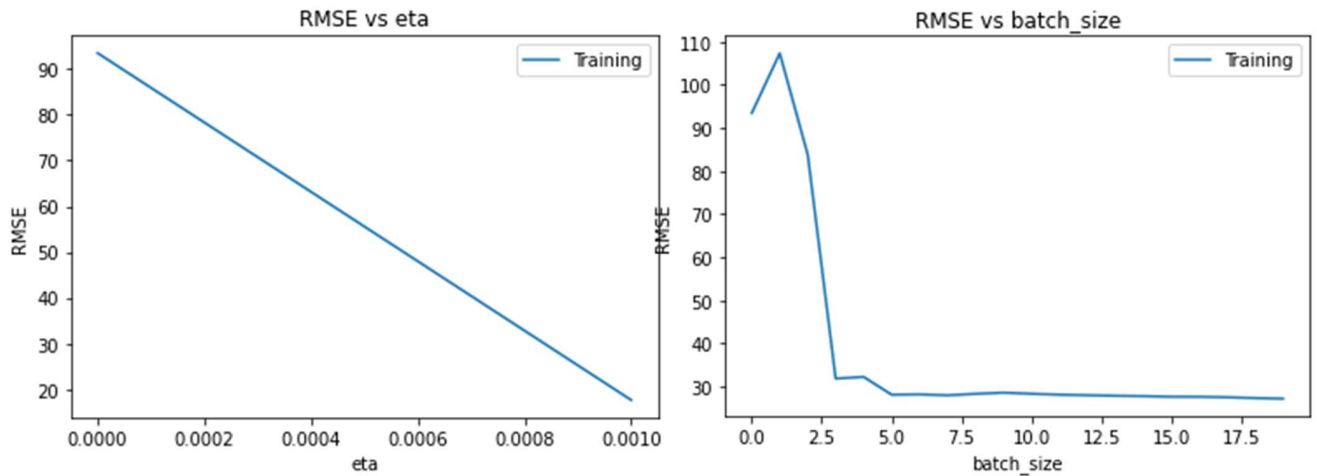
with 14 points have been used to test the models after every 1000 epochs. M=6 has chosen here for SGD. So more epoch are helping reducing error but up to 400epochs we will be getting the maximum error reduction so we can choose **400 as stopping criteria**. Data points are in blue and predicted points are in red. The plots for various epochs till 10000 are shown below. In problem 1A model converged at around 4000 epochs comparably 10 times taken more epochs to converge t



- Mini Batch Gradient Descent

Here we will be updating the weights after few samples are passed or you can say after a handful of samples called mini batches is passed. It will help us smoothly enter towards the optimal points/weights.

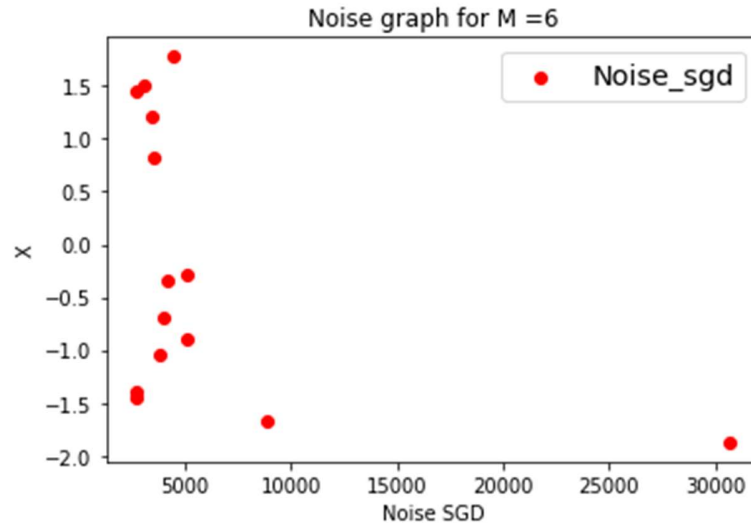
Model is trained for 5th order polynomial



So as above graph suggest to pick the learning rate at 0.001 and a optimal batch size to be 4 as our data set is only having 70 data points for training.

Weights for the 5th order polynomial for 0.001 learning rate,4 batch size and 400 epoch with 70 data points are as below.

[[10.55692608], [2.6134792], [9.2856239], [0.94688129], [11.65410628], [-2.47706711]]



Noise is distributed about 0 with very low variance as 71.35 for 6th degree polynomial.

