ELL409 Assignment 1

NIRJHAR DAS
2019EE30585

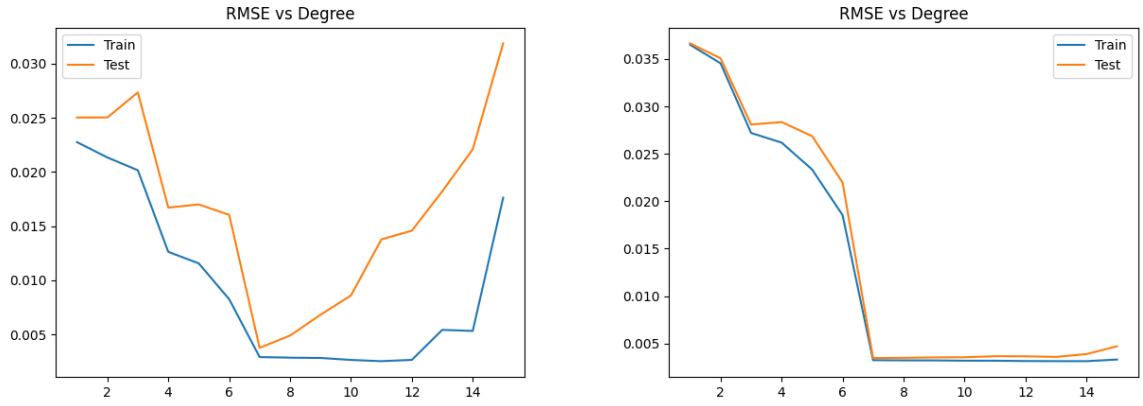
28 September 2021

1 Answer 1

1.1 Part A

For Part A, the python files *gaussian_data_fitting.py* and *gaussian_poly.py* are implemented.

The degree of the polynomial to be used as hypothesis was selected to be 7 using the following plot showing the degree of the polynomial vs. the training and testing RMSE. It can be seen that the region for degree < 7, it is underfitting as both train and test RMSEs are high, while for degree > 7, it is overfiiting as train RMSE decreases but test RMSE increases.



(a) With first 20 data points

(b) With all data points

Figure 1: Degree of Polynomial vs. RMSE

Various error functions have been tested which fall under the general formulation

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} |y(x_n, \underline{w}) - t_n|^p + \frac{\lambda}{2}\sum_{j=1}^{M} |w_j|^q$$

The values of p and q experimented with and the results are listed in Table 1. Note that $p = 0$ correspond to Huber Loss function.

The entries in the table correspond to training using Gradient Descent algorithm. While training with Gradient Descent, an issue was observed that despite lowering learning rate and increasing the number of iterations (epochs), the RMSE values were getting stuck at the values shown above. Subsequently, ADAM optimizer was implemented to adapt the learning rate so as to reach better convergence point. However, no significant changes were seen, except that the loss (and RMSE) decreased at a faster rate initially, after which it oscillated around the values shown in the table. Note that these values are the averages of 10-fold cross-validation set.

The effect of Gradient Descent vs. Number of Epochs can be seen below. Note that these plots were obtained for the specific configuration of $p = 2$ and $q = 2$ with, batch size $= 100$.

p	q	Train RMSE	Test RMSE
0	1	0.028551833568711882	0.026678103250304043
0	2	0.02854556298148307	0.02625563522710479
1	1	0.17235662016598446	0.12939689108306182
1	2	0.17235662016598446	0.12939689108306182
2	1	0.02991192182113585	0.026860959335720986
2	2	0.02991192182113585	0.026860959335720986

(a) First 20 Data Points, $\lambda = 0$

p	q	Train RMSE	Test RMSE
0	1	0.028494077768444297	0.02652183576853895
0	2	0.028481119733730194	0.02608676619552223
1	1	0.17316510980322322	0.17483775073926427
1	2	0.15996410089631163	0.11426701160001222
2	1	0.029895693500994525	0.026846155961183293
2	2	0.029909161470596503	0.026857206050257952

(b) First 20 Data Points, $\lambda \neq 0$

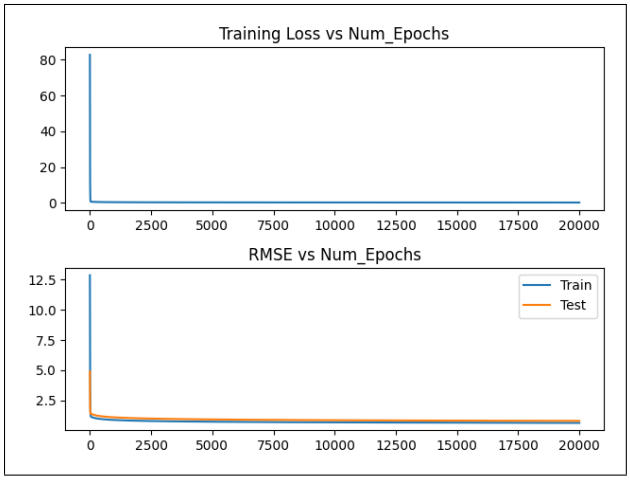
p	q	Train RMSE	Test RMSE
0	1	0.05694407887363011	0.05599654274654823
0	2	0.05224493074743018	0.05047932284470393
1	1	0.111009629232616	0.09554235770673904
1	2	0.111009629232616	0.09554235770673904
2	1	0.05478546864878288	0.05298971876601448
2	2	0.05478546864878288	0.05298971876601448

(c) All Data Points, $\lambda = 0$

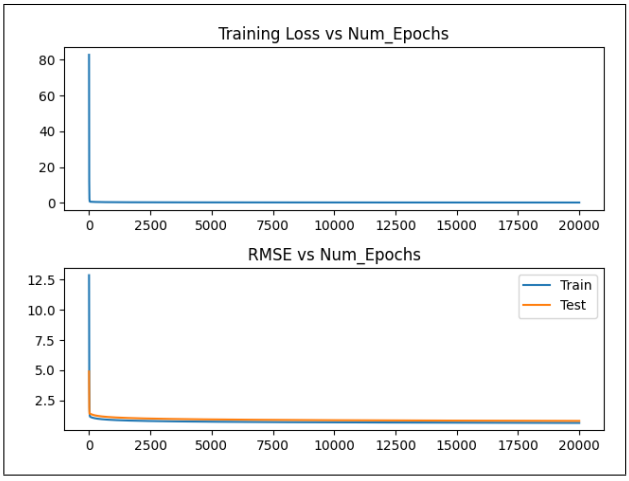
p	q	Train RMSE	Test RMSE
0	1	0.05680293334535684	0.055792629441605714
0	2	0.05221389786101342	0.050448116835662624
1	1	0.13568304995384894	0.11559264273303645
1	2	0.15959498482222023	0.14595493172037474
2	1	0.054785965238200854	0.05299661639471871
2	2	0.05478499686920004	0.05298542578246703

(d) All Data Points, $\lambda \neq 0$

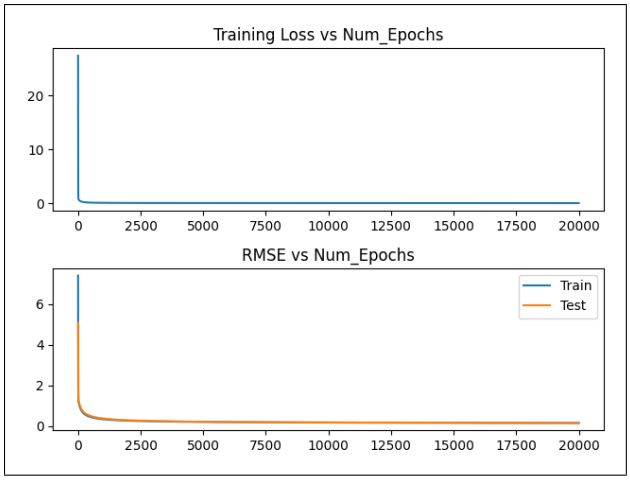
Table 1: Results of various error functions



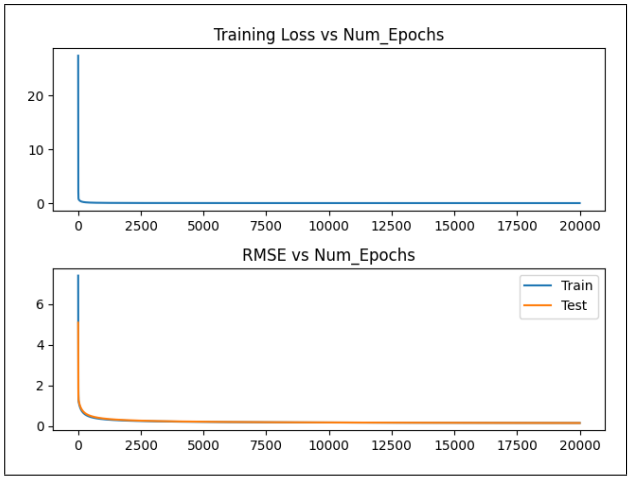
(a) First 20 Data Points, $\lambda = 0$



(b) First 20 Data Points, $\lambda \neq 0$



(c) All Data Points, $\lambda = 0$



(d) All Data Points, $\lambda \neq 0$

Figure 2: Effect of Number of epochs on Training

However, a lower value of RMSE can be obtained using the Pseudo-Inverse method, although it is restricted to only the loss function corresponding to $p = 2$ and $q = 2$. The values are:

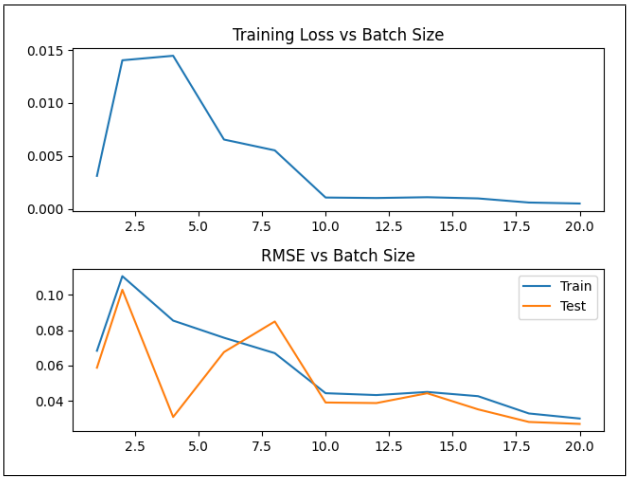
Dataset	λ	Train RMSE	Test RMSE
All	0	0.0032487174886034072	0.003471203510339035
All	$\neq 0$	0.003250829069465478	0.0034582729985219497
First 20	0	0.002976037598891525	0.0046805808840869444
First 20	$\neq 0$	0.0034585263879205715	0.005765323651300282

Table 2: Results using Pseudo-Inverse

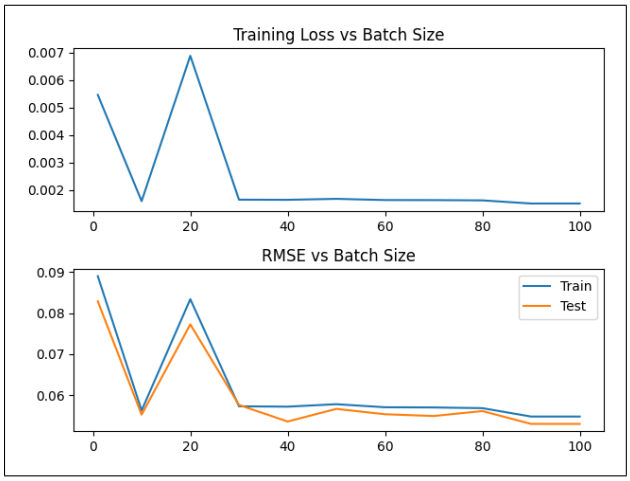
Thus, we see that Pseudo-inverse converges to a better solution than Gradient Descent.This is probably because due to the stochastic nature of Gradient Descent, the weights oscillate near the minimum, and even a slight increase in learning rate can cause the the algorithm to never converge to a good enough fit. However, Gradient Descent is a faster optimization method for high dimensional data since calculating the pseudo-inverse can be prohibitive for large M.

The effect of batch size on gradient descent can be seen from the following graphs.

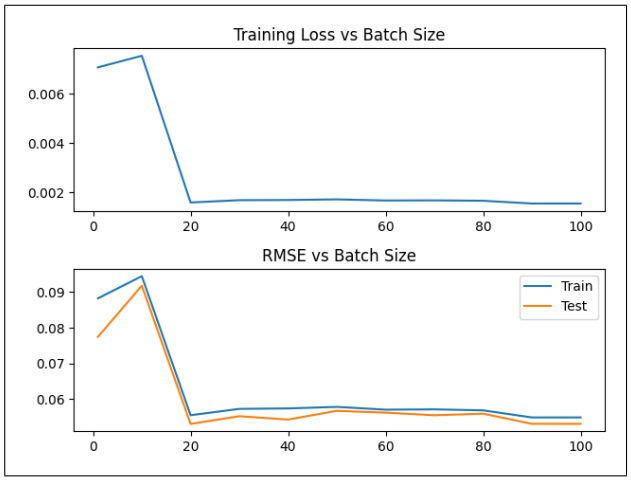
(a) First 20 Data Points, $\lambda = 0$



(b) First 20 Data Points, $\lambda \neq 0$



(c) All Data Points, $\lambda = 0$

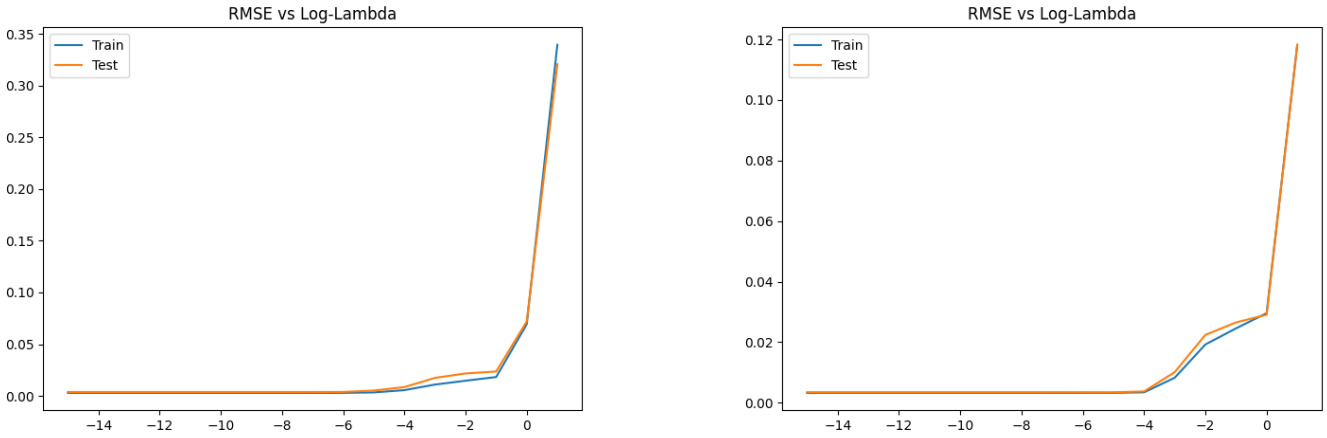


(d) All Data Points, $\lambda \neq 0$

Figure 3: Effect of Batch Size on Gradient Descent

From the graph, it can be observed that increasing the batch size makes the model generalize well as the test and train RMSEs become more in sync. Moreover, increasing the batch size not only speeds up training, but also causes smaller oscillations and better convergence.

Now, to pick the value of the regularization parameter λ, a sweep was performed. The results can be seen below.



(a) First 20 Data Points

(b) All Data Points

Figure 4: Effect of Regularization on Model

From the graph, it can be seen that for $\lambda \geq 10^{-3}$, the train and test RMSE start to increase. Hence, *lambda* was chosen to be 10^{-4} for all the experiments on this dataset wherever applicable. (In all the plots and tables above, $\lambda = 10^{-4}$).

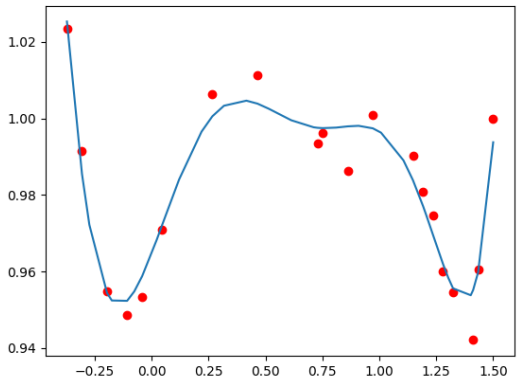
The distinction between overfitting and underfitting is not ver clear from these graphs, except for the plot of degree vs. RMSE.

Finally, the polynomials estimated are:

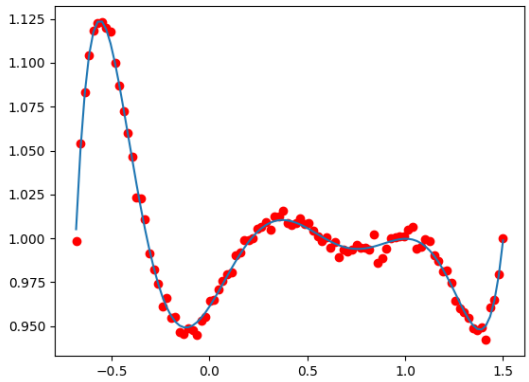
Weights	20 Data Points	All Data Points
w_1	0.9647354143234903	0.9610265584247616
w_2	0.15423133195965555	0.17146904323555356
w_3	0.19501293245716123	0.38084451163779687
w_4	-1.4020394040211634	-1.8266928525840287
w_5	1.400000616160682	0.7920145548434916
w_6	0.5611920485778779	2.679200441784392
w_7	-1.355181240320075	-3.0913948783140413
w_8	0.4786208744169125	0.9333091356155307

Table 3: Weights of the Polynomial $w_1 + \sum_{i=2}^{8} w_i x^{i-1}$

The polynomials can be visualized in the following plots in Figure 5.



(a) First 20 Data Points



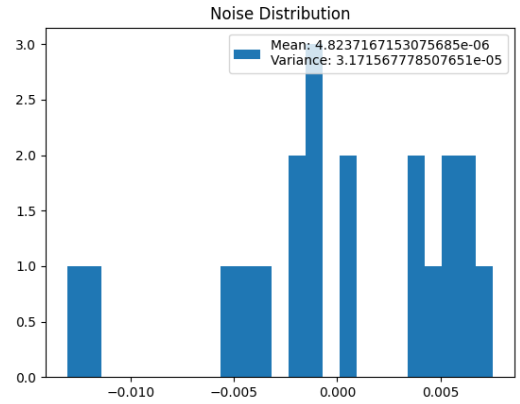
(b) All Data Points

Figure 5: Visualization of the polynomials obtained using Pseudo-Inverse

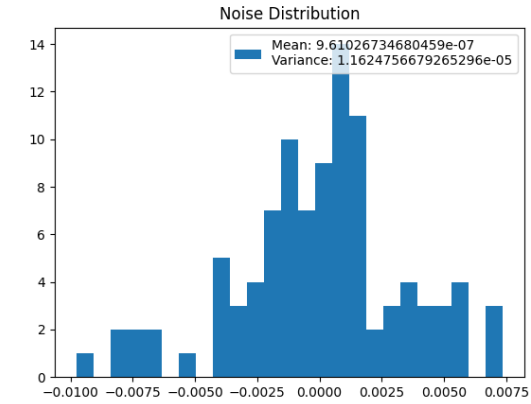
We can thus observe that adding more data increases the generalization power of the model as can be seen in Table 2, since the Train and Test RMSEs are lower for the model trained on 100 data points than on 20. Also, the train and test RMSEs are more in sync for the case of 100 data points. Also from Figure 2, we can see that, the Gradient Descent converges to a lower RMSE for 100 data points than 20 data points. Hence, introducing more data makes the model less susceptible to noise and leads to a better solution point.

From Figure 5b, we see that the obtained polynomial fits the data well, while the model weights have reasonable values, thus avoiding overfitting, while attaining a low value of RMSE to avoid underfitting.

The noise estimates are shown below:



(a) First 20 Data Points



(b) All Data Points

Figure 6: Noise Estimate of the data. Clearly, the noise distribution looks Gaussian.

1.2 Part B

For Part B, the code files *non_gaussian_data_fitting.py* and *non_gaussian_poly.py* have been used.

The order of the polynomial is estiamted using 10-fold cross validation search on the degree of polynomial. The result can be seen in the plot below.

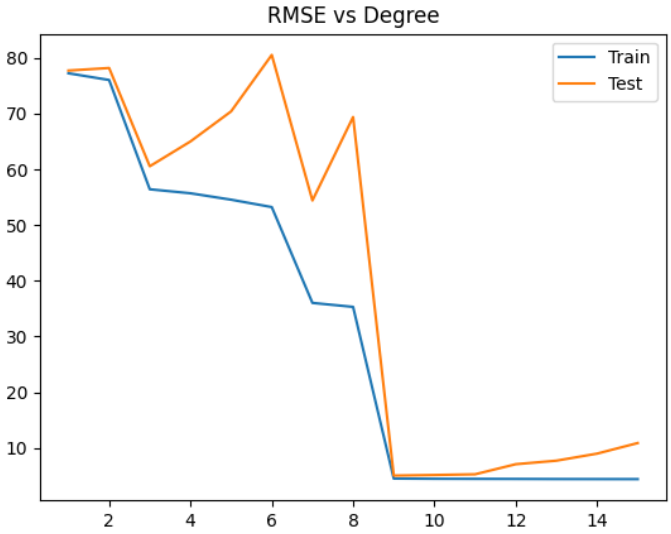


Figure 7: Degree vs. RMSE

From the plot, it is clear that the degree of polynomial should be 9, as the curve is underfitting (both train and test RMSE is high) for degree < 9 and overfitting (train RMSE is low while test RMSE increases) for degree > 9.

After this, several models were trained using 10-fold cross-validation on multiple loss functions. The results are shown in the following table.

p	q	Train RMSE	Test RMSE
0	1	320.25500323810405	284.61514764210534
0	2	388.5723653094422	645.0501829971832
1	1	346.7782922509427	384.05743745743604
1	2	350.0234367992331	364.94207415994094
2	1	138.7154083869904	158.04189158253342
2	2	138.71540835380927	158.04189153127112

Table 4: RMSE values for various Loss functions

Thus we can see that the loss functions corresponding to $p = 2$ and $q = 1,\ 2$ gives better fit among these losses. The final model is therefore trained using $p = 2$ and $q = 2$ since this can be effective done using Pseudo-Inverse, as Gradient Descent does not seem to converge to a good enough solution.

Now, to select the regularization parameter λ, an extensive search was performed. The plot obtained is as follows.

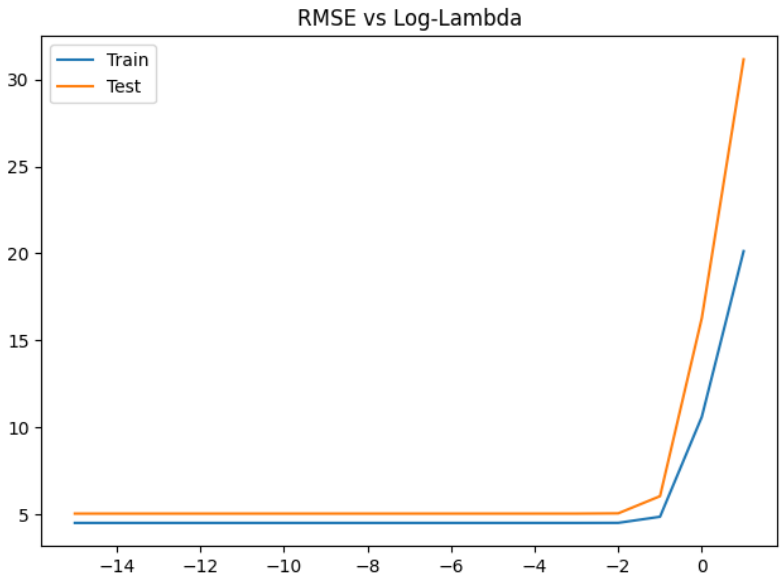


Figure 8: Regularization effect on RMSE

From the plot, it is observed that the value of $\lambda = 10^{-2}$ is well-suited for the model, as the model underfits for higher values of λ.

Thus, final model is solved using Pseudo-Inverse with a polynomial of degree 9 and $\lambda = 10^{-2}$. With this, the average values of train and test RMSEs over 10-fold cross-validation obtained are 4.51210955947599 and 5.058393154289743 respectively.

The corresponding weights are:

Weights	Values
w_1	21.337202781373172
w_2	-43.54948313923166
w_3	87.12011898172165
w_4	95.71627521148355
w_5	-117.55905787875317
w_6	-34.624573406345924
w_7	48.57070746213878
w_8	-1.7670716939127658
w_9	-5.635526365375034
w_{10}	1.0125616922722465

Table 5: Weights of the polynomial $\sum_{i=1}^{10} w_i x^{i-1}$

Note that although the values of weights corresponding to the lower powers of x are somewhat large, this should not be seen as a sign of overfitting as the value of the targets t (between 300 and -300) is quite high compared to the values of x (between -2 and 4).

The polynomial obtained can be visualized below.

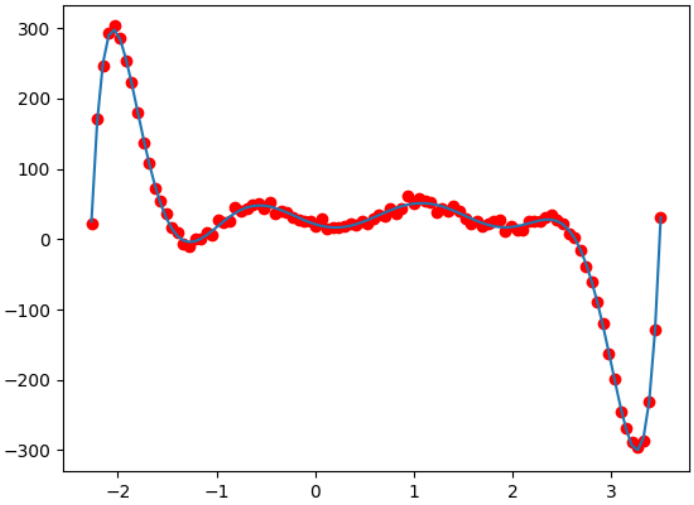


Figure 9: Visualization of the obtained polynomial

The noise estimate obtained from this polynomial can be seen in the following histogram plot.

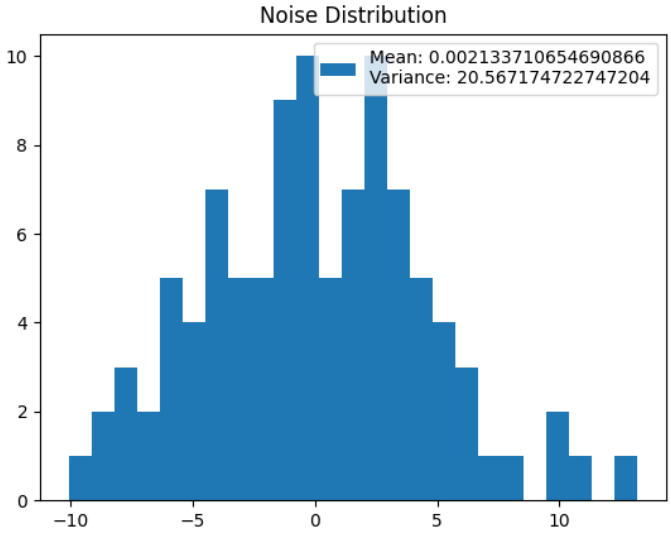


Figure 10: Noise Distribution

The distribution seems to have two modes and can be approximated as a weighted mixture of two Gaussian Distribution with different means. This can also explain the model's relatively better performance with Pseudo-Inverse as this method works the best for a Gaussian noise.

2 Answer 2

For this part, the linear regression is built on using the Fourier Transform as a basis function. This is motivated by the fact that the data seems to periodic with a fundamental period of 12, which corresponds to the 12 months of the year. Thus, the polynomial is defined as follows:

$$y(x) = w_0 + \sum_{i=1}^{M} w_{2i-1} \cos(\frac{2\pi i x}{N}) + w_{2i} \sin(\frac{2\pi i x}{N}) \ , \qquad M < N$$

The values of M and N were chosen to be 20 and 120 as this gave the lowest average test RMSE. Note that value of N was chosen to be 120 according to rules of Fourier Transform (total span of data = 120).

With this transformation, the model is fit using Pseudo-Inverse method with a value of $\lambda = 10^{-2}$. This was obtained using a 10-fold cross validation method and this value of λ gave the lowest average RMSE on the test sets. The obtained plot can be visualized as below.

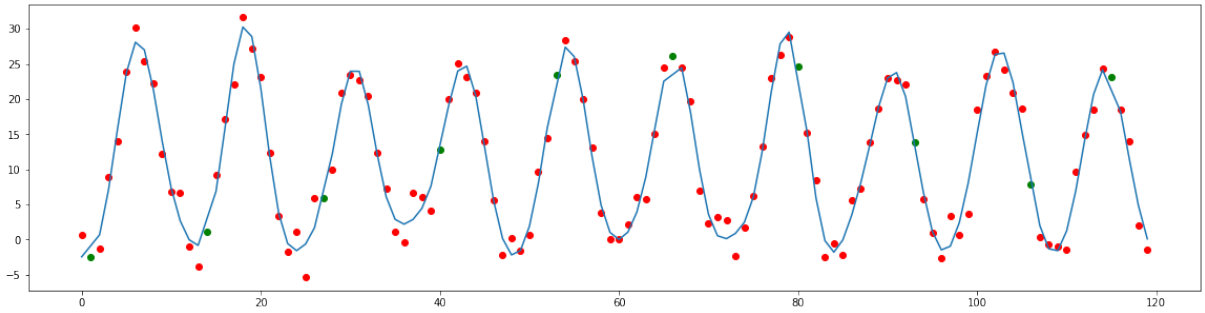


Figure 11: Plot of the polynomial. The green points correspond to test set.