Introduction to Reinforcement Learning & Deep Learning

CSI 5340

Polynomial Regression & Stochastic Gradient Descent

Assignment 1

Authors
Alexandra Sklokin (300010511)

Professor
Runzhi Tian

September 22, 2022

Contents

1 Introduction

In the following report I will discuss the results obtained from fitting a Polynomial Regression model to generated datasets. The Polynomial Regression model is trained using Stochastic Gradient Descent (SGD), with and without Weight Decay. We would like to evaluate the fitting and generalization capability of these models for different values of training data size (N), polynomial model degree (d), and dataset variance (σ^2).

2 Model Training

2.1 Polynomial Regression

The Polynomial Regression model is formulated as follows:

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + ... + \theta^d x^d \tag{1}$$

This model is in the class of Linear Regression Models, evident as y spans $\{1, x, x^2, ..., x^d\}$. In the training phase, to learn the model, we simply need to learn $\{\theta_0, \theta_1, ..., \theta_d\}$. This is done using some variation of the Gradient Descent (GD) algorithm.

2.2 Stochastic Gradient Descent

For GD, each θ_i is set to a random seed. Then they are incrementally updated until some stopping criteria is met (eg. have reached a certain number of repetitions/epochs). Each parameter is updated by subtracting the learning rate times the gradient (ie. derivative of loss function).

For SGD, the gradient is calculated, in each epoch, for one point in the training dataset. By contrast, in GD, the gradient is calculated using all points in the dataset, which increases run time significantly.

To save time without compromising model performance, I chose to use SGD with a small training rate and many epochs. From some experimentation, I determined that SGD with many epochs converged to similar values as GD with less epochs, but in significantly less time.

2.3 Weight Decay

Weight Decay is a regularization technique to reduce model complexity, thus trying to avoid the issue of overfitting and improving generalization performance. We add a weight decay term which works to penalize overly complex models. Theoretically, even polynomials models with a large d should be less prone to overfitting when using weight decay.

2.4 Parameters N, d, and σ^2

In this assignment, we are to discuss the fitting and generalization of models trained using the following parameters and values:

- $\mathbf{N}$ = {2,5,10,20,50,100,200} : the size of the training dataset while the Polynomial Regression Model is being learned.
- $\mathbf{d}$ = {0,1,2,...,20} : the degree of the Polynomial Regression Model. As d increases, model complexity increases.
- $\boldsymbol{\sigma^2}$ = {0.01,0.1,1} : the level of noise added to the generated datasets.

3 Model Evaluation

3.1 Performance Metrics

The following performance metrics were obtained at the end of parts (E) and (F), and they will be used to evaluate the fitting, generalization, and relationships between parameters N, d, and σ^2:

- $\overline{\mathbf{E_{in}}}$: training MSE. Gradient Descent algorithm is attempting to minimize this value, so we expect it to be small.
- $\overline{\mathbf{E_{out}}}$: testing MSE (over testing dataset with N = 20,000). Small $\overline{E_{out}}$ indicates good fitting. Large $\overline{E_{out}}$ might indicate overfitting.
- $\mathbf{E_{bias}}$ = : systematic error between average model (over M=50 trials) and true value of target feature (y). If this is large, it could indicate underfitting.

3.2 Bias-Variance Trade-Off

A well known phenomenon in Machine Learning is the Bias-Varance tradeoff, as illustrated in Figure 1. This can be used to explain overfitting and underfitting.

Underfitting can occur when a model is too simple. In this case, the model is unable to fit data in the training and testing sets.

Overfitting can occur when a model is trained to fit the training data perfectly (reducing bias). As model complexity increases, the model is trained to accurately make predictions on the training dataset, but is not generalizable to other datasets. Thus, the performance of the model on the testing set is not good.

Thus, it is obvious that testing error is estimated by adding the squared bias and variance. The best model complexity is one which yields the smallest testing error by compromising between the decrease in bias and increase in variance.

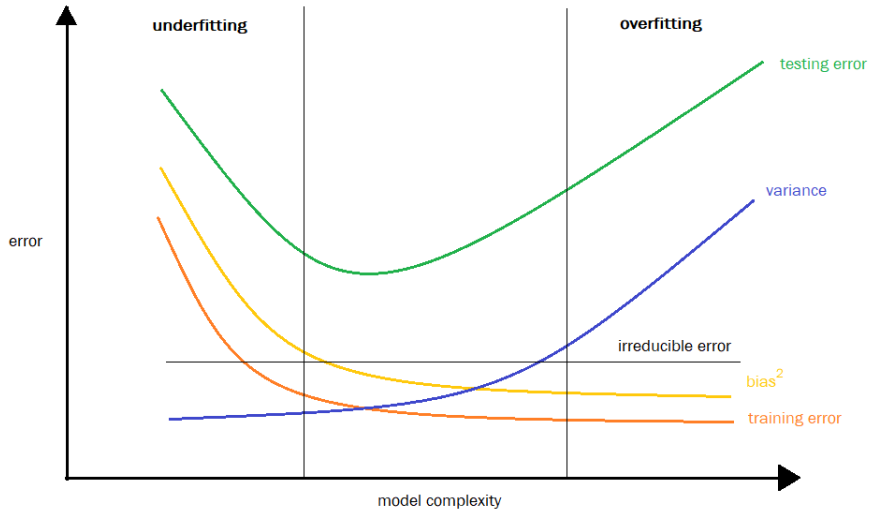

Figure 1: Graph of Bias-Variance Tradeoff

4 Methodology

For this assignment I used *jupyter notebooks* and Excel spreadsheets. Please find all of the code for this assignment in my Git repository[1]. *StochasticGradientDescent.ipynb* contains all of the code as required by parts (A) to (F).

5 Experimental Results

For the purpose of preserving space, the following tables contain only samples of interesting rows from the experiment. Full tables are in my Git repository: *output.csv* and *output_weight_decay.csv*.

Graphs are produced by collecting average $\overline{E_{in}}$'s, $\overline{E_{out}}$'s, and E_{bias}'s.

[1] https://github.com/alexandrasklokin/CSI5340/tree/main/CSI5340_A1

5.1 Stochastic Gradient Descent (E)

N	σ	d	$\overline{E_{in}}$	$\overline{E_{out}}$	E_{bias}
2	0	0.01	0.0706	0.6138	0.6084
2	0	0.1	0.9775	0.6079	0.5948
2	0	1.0	0.4262	1.7516	1.7497
...					
2	20	0.01	0.3577	45.5592	4.7603
2	20	0.1	0.0000	12.0120	1.3447
2	20	1.0	0.0001	12.7628	3.5185
5	0	0.01	0.2685	0.5321	0.5296
5	0	0.1	0.6511	0.6083	0.6151
5	0	1.0	1.5643	2.4050	2.3245
...					
5	20	0.01	0.1492	15.5631	2.1301
5	20	0.1	0.4126	3.2615	0.5169
5	20	1.0	1.4874	9.0866	1.9157
...					
100	0	0.01	0.4758	0.5159	0.5211
100	0	0.1	0.5635	0.6155	0.6194
100	0	1.0	1.5727	1.5360	1.5552
...					
100	20	0.01	0.1355	0.1412	0.0700
100	20	0.1	0.1781	0.1934	0.1342
100	20	1.0	1.0515	1.3046	1.1282
200	0	0.01	0.5161	0.5223	0.5260
200	0	0.1	0.6282	0.6019	0.6145
200	0	1.0	1.8087	1.5186	1.5637
...					
200	20	0.01	0.1112	0.1065	0.0631
200	20	0.1	0.1909	0.2118	0.1450
200	20	1.0	1.2662	1.1843	1.0929

Table 1: Sample of Experimental Results for Polynomial Regression with Stochastic Gradient Descent

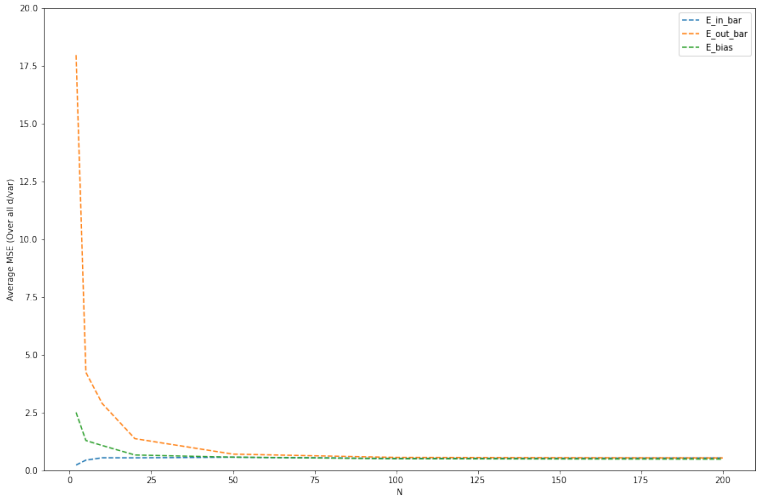

Figure 2: Graph of Model Performance at Different Training Sample Size Levels

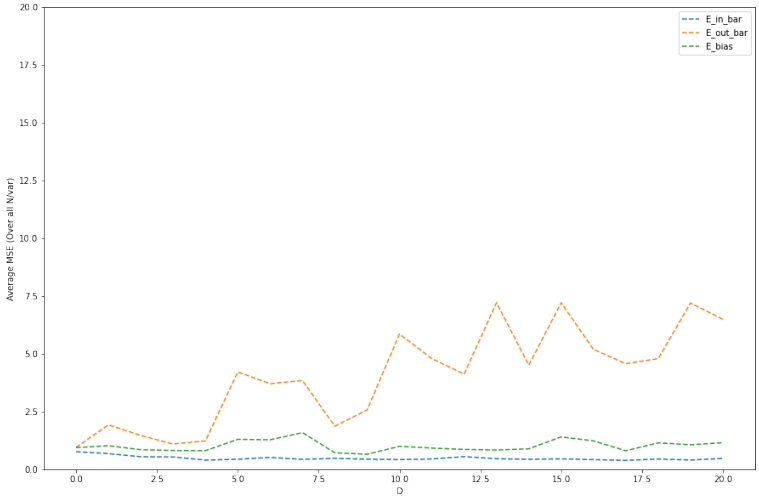

Figure 3: Graph of Model Performance at Different Polynomial Degree Levels

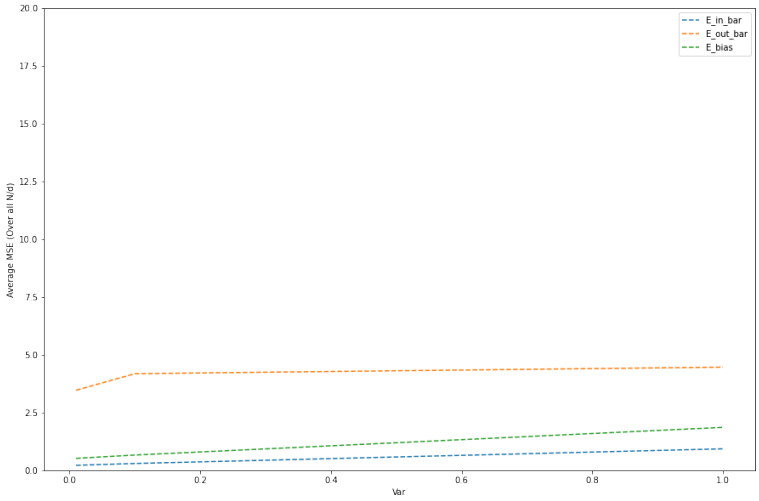

Figure 4: Graph of Model Performance at Different Variance Levels

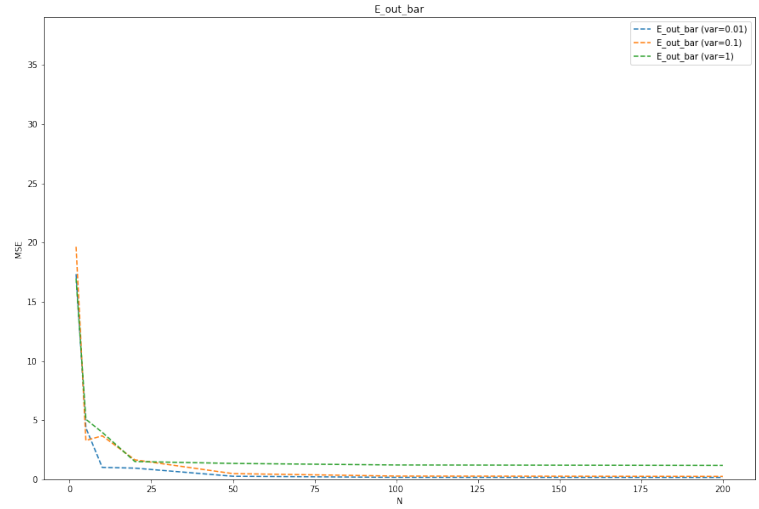

Figure 5: Graph to Compare Performance at various levels of N and σ^2.

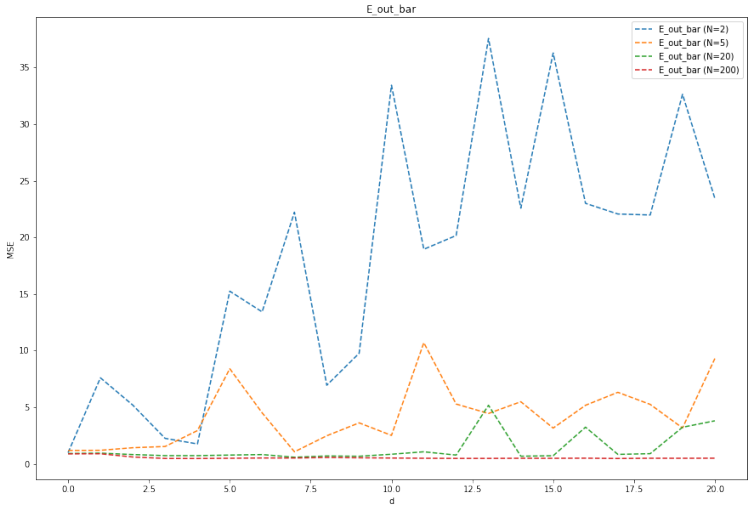

Figure 6: Graph to Compare Performance at various levels of d and N.

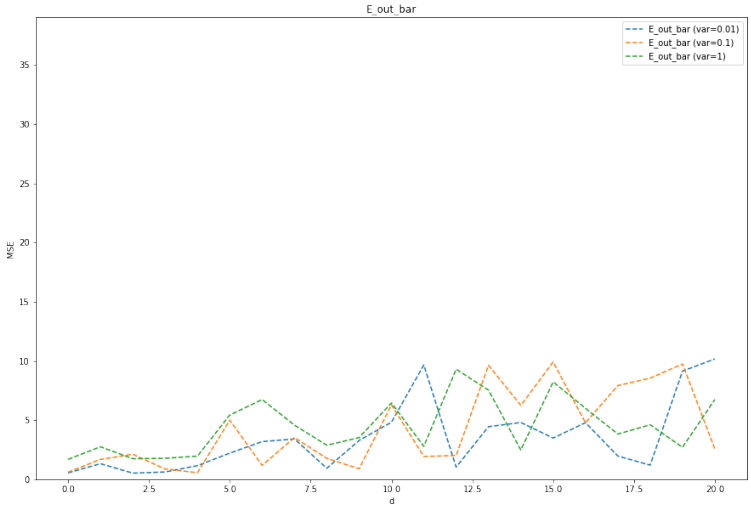

Figure 7: Graph to Compare Performance at various levels of d and σ^2.

5.2 SGD with Weight Decay (F)

N	σ	d	$\overline{E_{in}}$	$\overline{E_{out}}$	E_{bias}
2	0	0.01	0.2603	0.6114	0.6309
2	0	0.1	0.4539	0.6226	0.6129
2	0	1.0	1.6329	1.5616	1.5027
...					
2	20	0.01	0.0470	22.8117	0.9681
2	20	0.1	0.2955	34.2034	0.9392
2	20	1.0	0.0113	20.6353	4.0947
5	0	0.01	0.7788	0.5267	0.5147
5	0	0.1	0.5275	0.6169	0.5960
5	0	1.0	1.1483	1.6075	1.5926
...					
5	20	0.01	0.6915	7.3993	1.3948
5	20	0.1	0.3490	10.5306	0.4531
5	20	1.0	0.3671	6.1139	2.3090
...					
100	0	0.01	0.5604	0.5156	0.5165
100	0	0.1	0.5757	0.6073	0.5738
100	0	1.0	1.3618	1.5034	1.4975
...					
100	20	0.01	0.8550	0.8713	0.3236
100	20	0.1	1.0034	1.0816	0.3419
100	20	1.0	1.5903	1.8481	1.1941
200	0	0.01	0.5064	0.5184	0.5068
200	0	0.1	0.6325	0.6078	0.6033
200	0	1.0	1.5026	1.5254	1.4313
...					
200	20	0.01	0.8104	0.8690	0.2806
200	20	0.1	0.9173	0.9692	0.3944
200	20	1.0	1.8373	1.9460	1.3959

Table 2: Sample of Experimental Results for Polynomial Regression with Stochastic Gradient Descent and Weight Decay

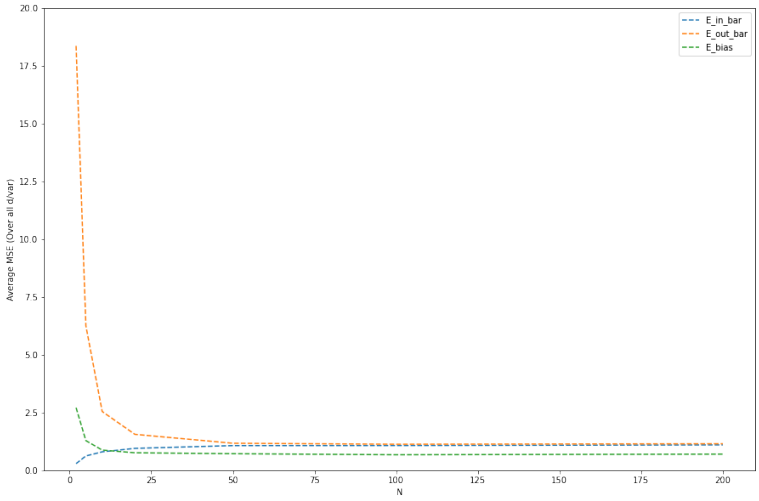

Figure 8: Graph of Model Performance at Different Training Sample Size Levels

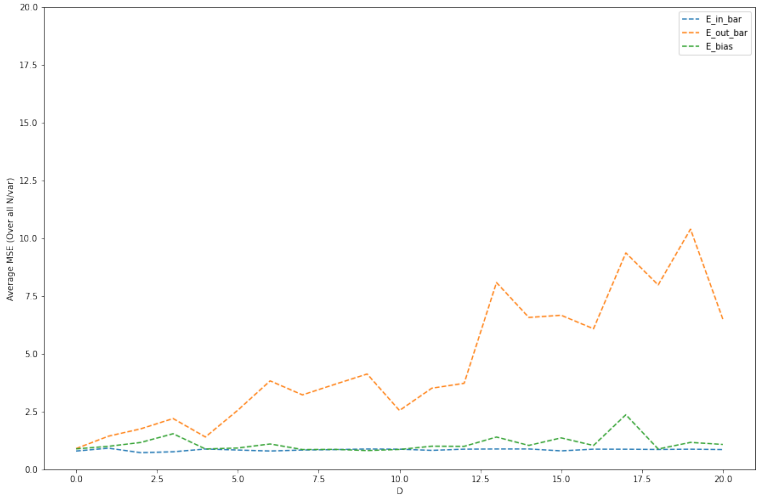

Figure 9: Graph of Model Performance at Different Polynomial Degree Levels

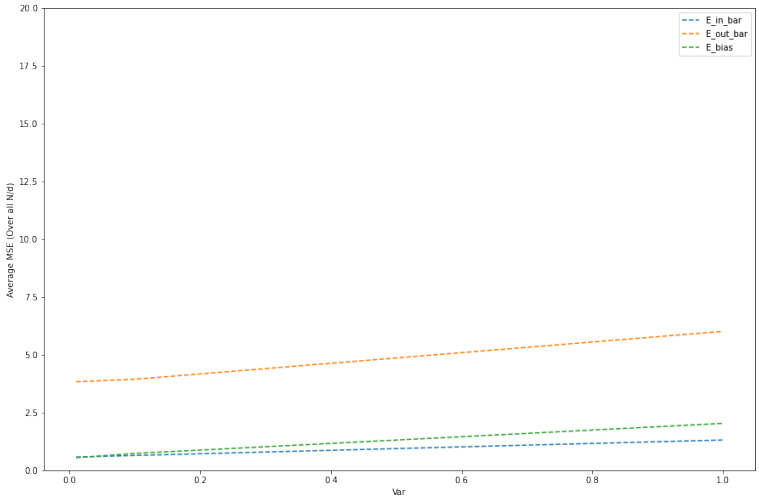

Figure 10: Graph of Model Performance at Different Variance Levels

6 Discussion

Using SGD, with and without weight decay, I was able to train many Polynomial Regression Models, using different values of N, d, and σ^2. Figure 11 shows an example of how SGD converges to the 'best' polynomial coefficients over 1500 epochs. The displayed MSE corresponds to the E_{in} for that epoch. Is it evident that the learning is negative exponential (ie. it learns fastest at the beginning, and then slows down).

```
Random Theta:  [10, -7, -10] MSE: 29.91258
Epoch: 0, Coefs:  [9.860099939020031, -6.994899294857371, -9.97234342577885] MSE: 28.79595
Epoch: 1, Coefs:  [9.682244829604253, -7.094817390271859, -10.030728313276764] MSE: 27.05715
Epoch: 2, Coefs:  [9.594767643336526, -7.068277922042244, -9.994234676823778] MSE: 26.49035
Epoch: 3, Coefs:  [9.420745125479531, -7.147982483211167, -10.03616042429579] MSE: 24.99009
                                    ...
Epoch: 190, Coefs:  [4.371240464585435, -5.066316458968005, -6.937230260480147] MSE: 6.64143
Epoch: 191, Coefs:  [4.315488803655624, -5.101291350939528, -6.959695002025792] MSE: 6.61741
Epoch: 192, Coefs:  [4.325077391005596, -5.097952793378793, -6.959092247264344] MSE: 6.6239
                                    ...
Epoch: 1496, Coefs:  [-0.1083321865876803, -1.1884290031881268, -1.5321298180434497] MSE: 2.18254
Epoch: 1497, Coefs:  [-0.12041735963700428, -1.1951624878907998, -1.5359874719504407] MSE: 2.19449
Epoch: 1498, Coefs:  [-0.051593758886744875, -1.167180781475 75, -1.521558360878294] MSE: 2.13838
Epoch: 1499, Coefs:  [-0.05079589511922756, -1.1667819751875135, -1.521352966350741] MSE: 2.13779
```

Figure 11: Example of Polynomial Regression using SGD

For each combination of parameters, this experiment was run $M = 50$ times. Often each of the trial polynomial's coefficients converged to similar values. However, since we always start random θ_i's, there were instances of dissimilar coefficients. Nevertheless, as shown in Figure 12, these polynomials estimators achieved comparable $E_{in,i}$'s and $E_{out,i}$'s. The average model (over M) was computed and used to estimate the E_{bias} of the models.

```
_____ M=1
FINAL MODEL: Coefs:  [0.1070964897195163, -1.719695514150569, 2.5163724414593474]  E_in: 0.3187165157990935 E_out: 0.3636258049
5265046
_____ M=2
FINAL MODEL: Coefs:  [-0.23472716533981497, -1.2317566964835565, 2.296460506705458 6]  E_in: 0.38714105411413974 E_out: 0.408982
31971947074
_____ M=3
FINAL MODEL: Coefs:  [-1.030943731364606, 5.41992264329646, -5.158993787150973]  E_in: 1.1782542169952528 E_out: 1.178836152005
213
_____ M=4
FINAL MODEL: Coefs:  [-0.8268811987571344, 2.90254251688469, -1.4980637271164101]  E_in: 0.8140873050450171 E_out: 0.8360413016
331399
_____ M=5
FINAL MODEL: Coefs:  [0.628508468749726 8, -3.7187105581433553, 3.8669158240713237]  E_in: 0.18388923926730066 E_out: 0.20893494
499656162
```

Figure 12: Example of Running Polynomial Regression M=5 times

I will now discuss the relationships between our performance metrics and parameters. As well, I will compare performance of GD with and without weight decay.

6.1 Stochastic Gradient Descent (E)

6.1.1 N

As N increases, $\overline{E_{out}}$ and E_{bias} decreases and $\overline{E_{in}}$ increases. Intuitively, when $N = 2$, the best model connects the two points with a single line, however when the testing set is constructed, that line is not likely to pass through the new points. As N increases, the training error increases slightly, since we are unable to reach all points (particularly outliers), however our testing error decreases significantly since our model is more generalizable. Thus, we have learnt that a large training and testing set is imperative to Machine Learning. [Figure 2]

6.1.2 d

As d increases, $\overline{E_{out}}$ increases slightly. This is a product of overfitting, since the complex model is trained to fit the training data (as seen by the low $\overline{E_{in}}$). It seems d has little effect on the E_{bias}, but rather significantly increases the model variance. As d increases, so does the model complexity, thus placing us in the 'overfitting' section of the Bias-Variance Trace-Off. Our data does not favour a particular value of d, but best performance is achieved with less model complexity. [Figure 3]

6.1.3 σ^2

As σ^2 increases, all three performance metrics increase. This is because we are introducing more noise into our data. [Figure 4]

Based on the above mentioned figures, it is clear that N is the most significant parameter for good fitting and generalization. It had the more drastic impact on the decrease/increase in $\overline{E_{out}}$.

6.1.4 N vs. σ^2

When N is small and σ^2 is large, our $\overline{E_{out}}$ is large. The effect of the noise is magnified by the lack of data. When N is small, we would like our data to be 'clean' so that the model has a better chance of making accurate predictions. [Figure 5]

6.1.5 d vs. N

When N is small, d can be small without compromising on performance and testing error. A small dataset does not require a complex model, or else we witness overfitting. As N increases, the d value may increase to fit the data better. Again, it is obvious that the more data we have, the better our generalization

capabilities, regardless of the d value. If we would like to do Machine Learning with complex models, we will need large datasets. [Figure 6]

6.1.6 d vs. σ^2

Here we see that that higher d polynomials may be able to help to negate the effects of a higher σ^2. We see, in general, that as d increases, for any variance, the error rate increases. However, as σ^2 increases, for any d, there is not a clear growth in error rate. [Figure 7]

6.2 SGD with Weight Decay (F)

After repeating the experiment with Weight Decay, the same observations with regards to parameters N, d, and σ^2 were made. Again, a large N had the most significant impact on improving fit and generalization. Increase in d usually witnessed overfitting. Increase in σ^2 always introduced more noise, thus increases the error rates. [Figures 8, 9, 10]

When comparing between SGD with and without Weight Decay:

- Weight decay has lower/equal $\overline{E_{in}}$ 65 / 441 times;
- Weight decay has lower/equal $\overline{E_{out}}$ 103 / 441 times;
- Weight decay has lower/equal E_{bias} 119 / 441 times.

Although Weight Decay is used to reduce overfitting, we see that it did not outperform the regular SGD in any metric. Potentially, we need to increase the number of epochs (=1500) for the Weight Decay algorithm to converge to the same MSE. Nevertheless, more than 90% of the time the Weight Decay had a higher error rate by less than 1.0 units. With some additional experimentation, I found that increasing the number of epochs of the GD algorithm usually brought the Weight Decay MSE's down this 1.0 unit.

7 Conclusion

In conclusion, I was able to successfully train a Polynomial Regression model to a generated dataset by implementing Stochastic Gradient Descent. I discovered that increasing N has the most significant positive impact on model fitting and generalization. Increases in d and σ^2 usually had a negative impact on fitting due to overfitting and noise, respectively. I also implemented Weight Decay to avoid overfitting and discovered that the Gradient Descent algorithm requires more epochs for better model performance. Overall, the Polynomial Regression models obtained in this assignment had excellent performance, and clearly demonstrated the relationships between parameters N, d, σ^2, and training/error rate.