HOMEWORK 1

INDEX

Abbreviation List

N	Sample size/ Dataset Size
σ^2	Variance
MSE or E	Mean Square Error
d or D	Degree of the polynomial
x	Input of the polynomial
y	Output of the polynomial
E_{in}	MSE of the model during training
E_{out}	MSE of the model during testing

Introduction

Regression is a machine learning method which is used to understand the relation between independent variables i.e the input and dependant variable which is the output. Polynomial Regression is a part of regression analysis where the relation between input and output is expressed in the form of a n-degree polynomial. Polynomial Regression is used when the relation between variables cannot be or is hard to be shown with simple linear regression.

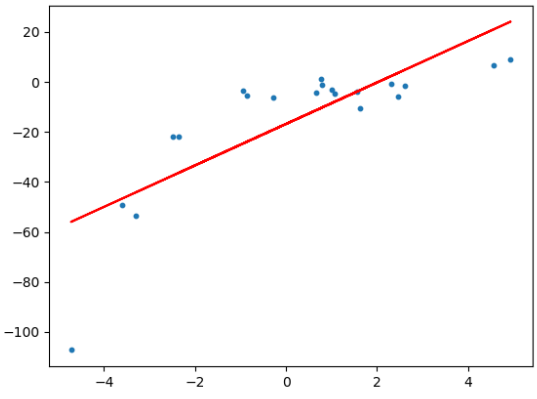


Figure 1 Linear Regression

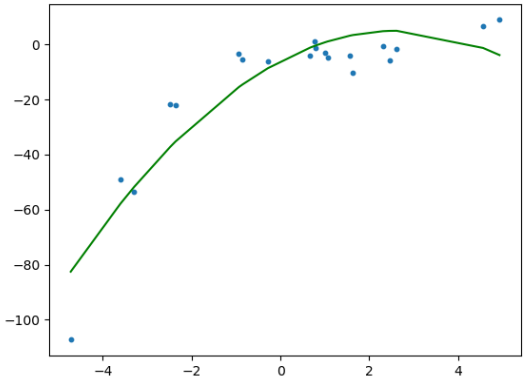


Figure 2 Polynomial Regression with degree 2

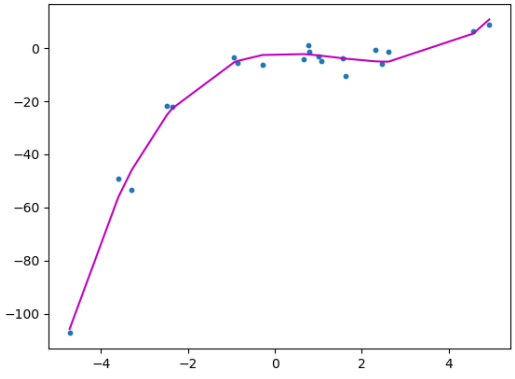


Figure 3 Polynomial Regression with degree 3

Problem definition and Algorithm

Task Definition

The main idea behind the assignment is to explore the fitting and generalization of regression models via simulation and have a better understanding on its importance with relation to the model accuracy.

We are provided with a dataset having variance σ^2, and we need to make a polynomial regression model along with examining the capability of the model in relation with model complexity and sample size. We also need to study the effect of weight-decay regularization on the model.

Algorithm Definition

1. Polynomial Equation of *d* degree

$$Y = a_0 + a_1X + a_2X^2 + \ldots + a_dX^d$$

where *d* is the polynomial degree and a_i's are coefficients to be estimated. This equation is used to calculate the output (value of *y*) when provided with a value for *x*.

2. Mean Squared Error (MSE)

$$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)^2$$

It calculates the mean value of the sum of squared difference between the predicted and actual output. The difference between the predicted and actual output depicts the error.

3. Partial Derivative or more commonly used term, Gradient, of the polynomial w.r.t each coefficient is

- Partial derivative with respect to a:

$$\frac{dE}{da} = \frac{-2}{n}\sum_{i=0}^{n} x_i^2(y_i - (ax_i^2 + bx_i + c))$$

- Partial derivative with respect to b:

$$\frac{dE}{db} = \frac{-2}{n}\sum_{i=0}^{n} x_i(yi - (ax_i^2 + bx_i + c))$$

- Partial derivative with respect to c:

$$\frac{dE}{dc} = \frac{-2}{n}\sum_{i=0}^{n} (y_i - (ax_i^2 + bx_i + c))$$

when a, b, c are the coefficient of a 2nd degree polynomial, $\bar{y}_i$

$$\bar{y}_i = ax_i^2 + bx_i + c$$

4. To decrease the loss i.e. the MSE of a polynomial, we decrease the value of each coefficient by some learning rate (lr) towards the global minimum using gradient descent. This can be expressed using the formula,

$$a_{new} = a - lr \times g_a$$
$$b_{new} = b - lr \times g_b$$
$$c_{new} = c - lr \times g_c$$

where a, b and c are the coefficients, lr is the learning rate and g_a , g_b and g_c are the gradients.

5. Lasso Regularization/weight decay can be applied using the below equation on the update step,

$$c_i = c_i - learning\_rate * gradient - learning\_rate * lamdba * c_i$$

where c_i is the coefficient, and learning rate, and lambda are any positive value.

6. The MSE of a polynomial, we decrease the value of each coefficient by some learning rate (lr) towards the global minimum using gradient descent. This can be expressed using the formula,

$$a_{new} = a - lr \times g_a$$
$$b_{new} = b - lr \times g_b$$
$$c_{new} = c - lr \times g_c$$

where a, b and c are the coefficients, lr is the learning rate and g_a , g_b and g_c are the gradients.

Experimental Evaluation

Methodology

1. Import packages

   The first step of this assignment is selecting the required packages and importing them. I have selected libraries for basic computation, so it just helps to make the process smoother but not easier. The libraries are as follows:

   - **Numpy**: Numpy supports large, multi-dimensional arrays and matrices, and provides a large collection of high-level mathematical functions to operate on these arrays. I have used numpy for the following needs:
     - Getting random N values for input.
     - To calculate noise
     - To generate a polynomial of n degree
   - **Math**: This module provides access to the mathematical functions defined by the C standard. This library provided an easy way to access values and functions like *pi* value, the *cos* function and the *square root* function.
   - **Matplotlib Pyplot**: Matplotlib is basically used for plotting various graphs which provided insights on relation between variables.

2. Create Dummy Dataset

   Here, we are creating a dummy dataset of size N. We have also added noise with variance σ^2 so that it resembles the real world dataset.

3. Define d-degree polynomial function

   This part defines our polynomial function of d degree which is created when we pass a list of coefficient. It returns the value of the function when input = x.

4. Mean Square Error (MSE)

   This step involves the calculation of the Mean square error, more commonly abbreviated as the MSE, of the model when predicted output and actual output is provided. While training the model on the dataset we need to decrease this value to the lowest so as to obtain an accurate predictor model.

5. Calculate New coefficient using gradient descent

   The use of gradient descent is shown in this step which helps us to calculate new coefficient which brings our model much nearer to the ideal model. First we are getting the partial derivative/gradient of the function w.r.t. each coefficient and then updating the coefficient using learning rate and the calculated gradient.

In Lasso Regularization/Weight decay, in addition to the gradient descent method we are also applying the weight decay formula to get a more generalized model.

6. Get the best fit for data by calculating new coefficients over 320 epochs

   We are doing 2 things here; first getting the new coefficients which are updated using gradient descent and then testing the model with a much larger dataset. The MSE calculated with training dataset, E_{in} and the MSE calculated using the test dataset, E_{out} is returned along with the new coefficients.

7. Get the Ein and Eout for each value of N, d, and variance

   This is the start of the assignment where we initialize all the variables and start with the training of the model for M trails with different values of N (dataset size), d (degree) and variance.

Observation and Discussion

1. **The value for epoch**

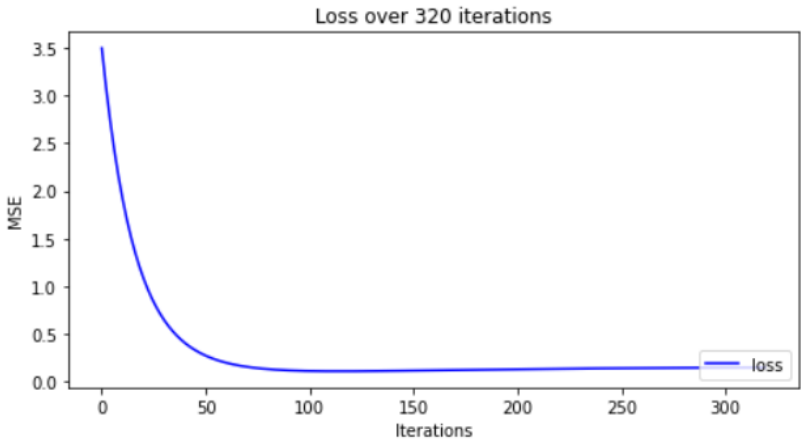


Figure 4 Decrease is seen in the value of MSE as the iteration increases.

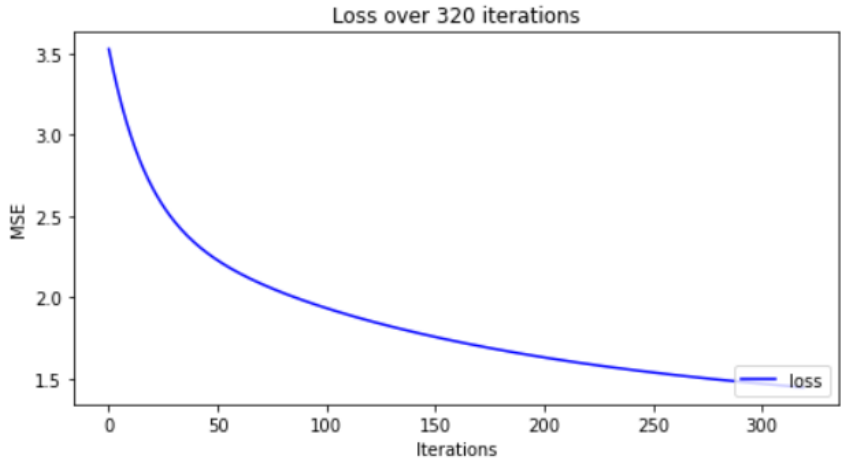


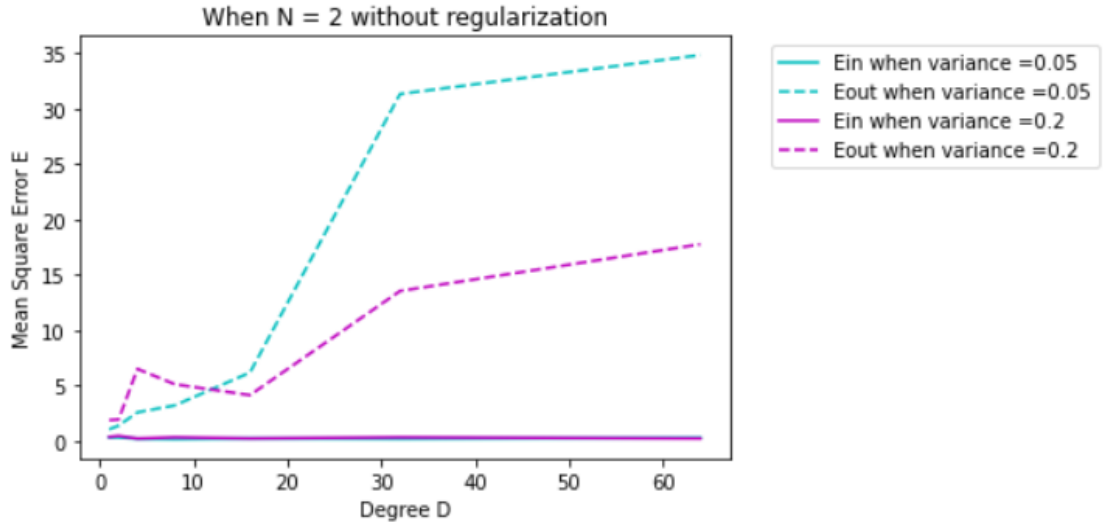
Figure 5 Here the random coefficient generated is too far from the final calculated coefficient, so more number of epoch/iterations is needed to decrease the MSE value to 0.

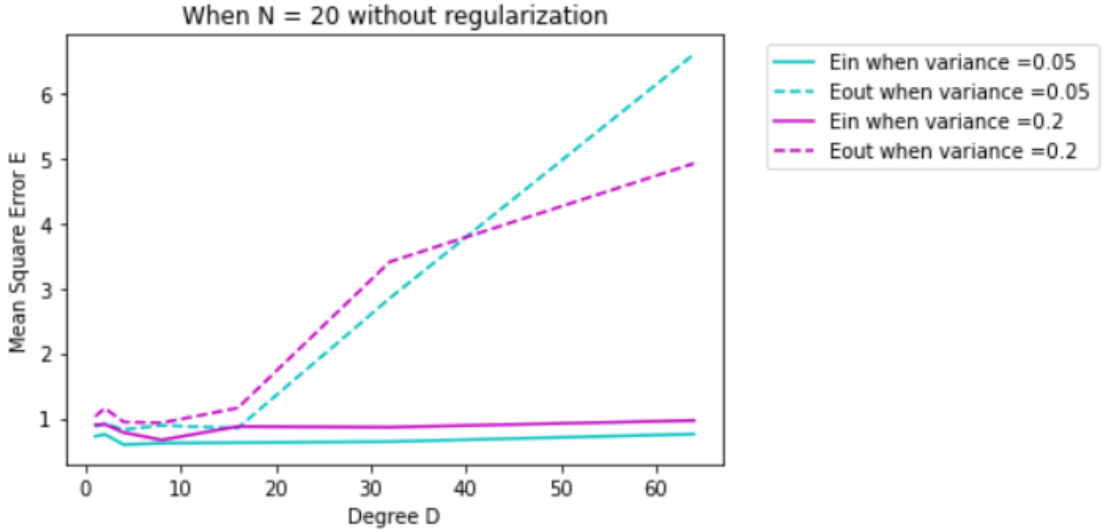
Deciding upon the value for epoch mainly depends on the basis of how far the random coefficient is generated from the new coefficient (the final calculated coefficient) and what the learning rate is. If the learning rate is too small then we need to have more number of epoch to iterate over the coefficient. But if the learning rate is too large then we cannot get a good value for coefficient.

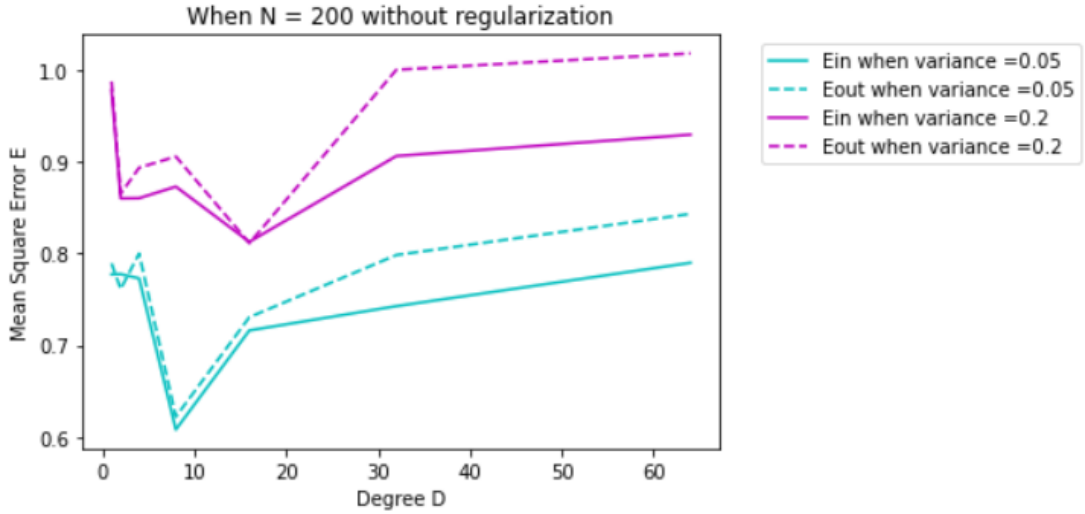
2. Effect of Sample size on the model

In the images below, for the 3 values of N, the difference between error E_{in} and E_{out} decreases as the Sample size / the dataset size increases. This means that if provided with enough data then the chances of model accurately predicting during testing increases. We can also see that the when N=200 then the error E_{in} and E_{out} of the model decreases as compared to when N was 2 or 20.

Here as a side note, we can also observe that as the degree of the polynomial increases, the difference between error E_{in} and E_{out} also increase, meaning that the model tends to become overfit with large value of degree. Also we can see that when N=200 the lowest MSE is at the point when degree is in the range of 2-12. Hence, the best model is when degree is in the range of 2-12.


When N = 2 without regularization
Mean Square Error E
Degree D
Ein when variance =0.05
Eout when variance =0.05
Ein when variance =0.2
Eout when variance =0.2

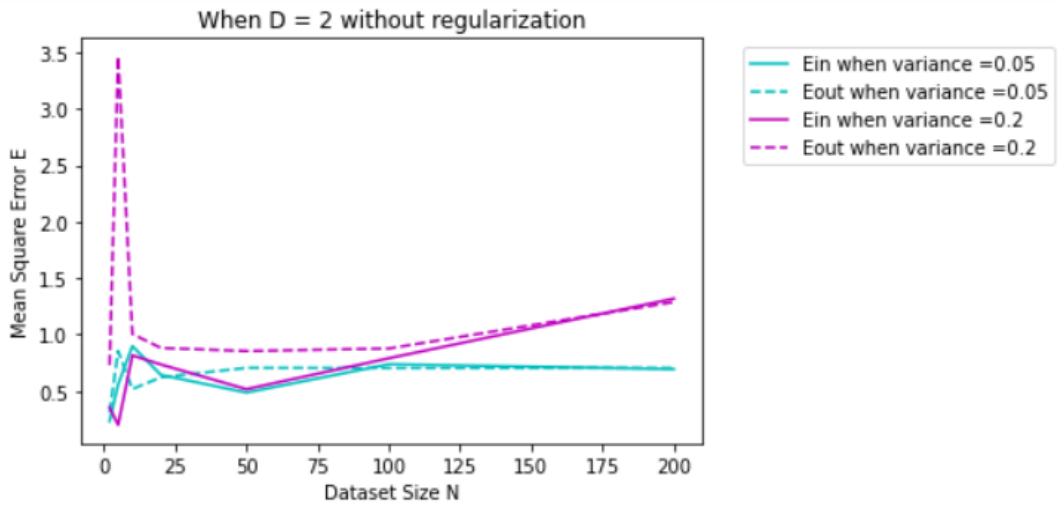

When N = 20 without regularization
Mean Square Error E
Degree D
Ein when variance =0.05
Eout when variance =0.05
Ein when variance =0.2
Eout when variance =0.2

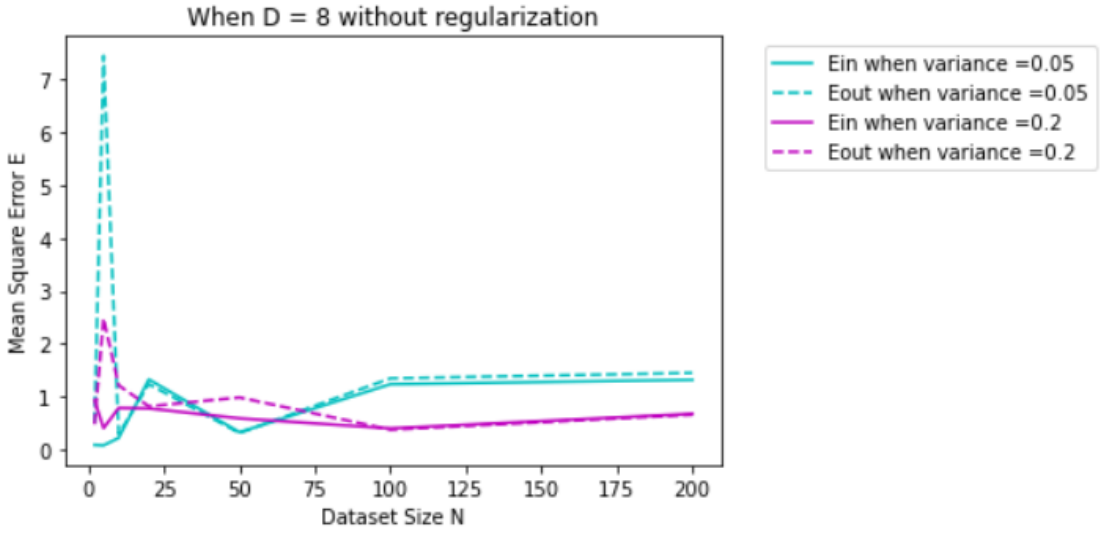

When N = 200 without regularization
Mean Square Error E
Degree D
Ein when variance =0.05
Eout when variance =0.05
Ein when variance =0.2
Eout when variance =0.2

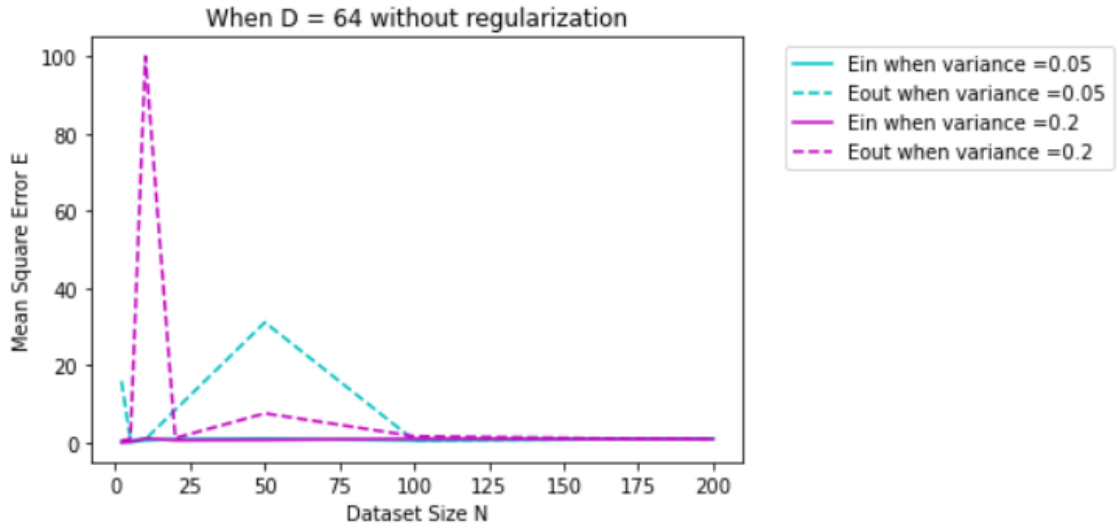
3. Effect of Polynomial degree on the model

The graphs show us that as the degree of the polynomial increases, the MSE of the model increases. Among the image shown below D = 2 has the least MSE making the model better as compared to models having larger value of polynomial degree.

Here also, as a side note we can see that when the sample size is very less, the E_{out} value first decreases and then remains stable along with the increase of N while the E_{in} first increases then remains stable. And in all the values of D, we can see that the difference between E_{in} and E_{out} decreases as N increases but only till a certain value of N i.e. when N≈125 to 150, after which there is not much decrease seen in the difference between E_{in} and E_{out} .

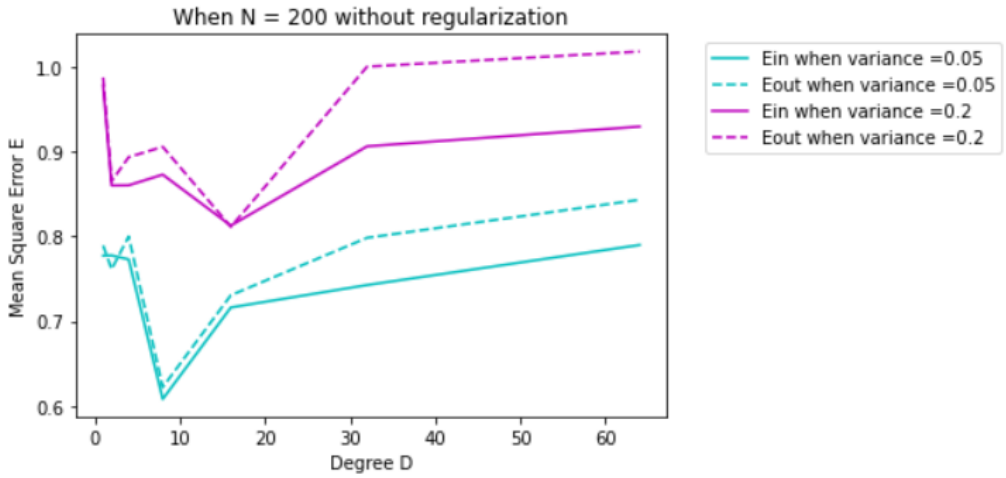


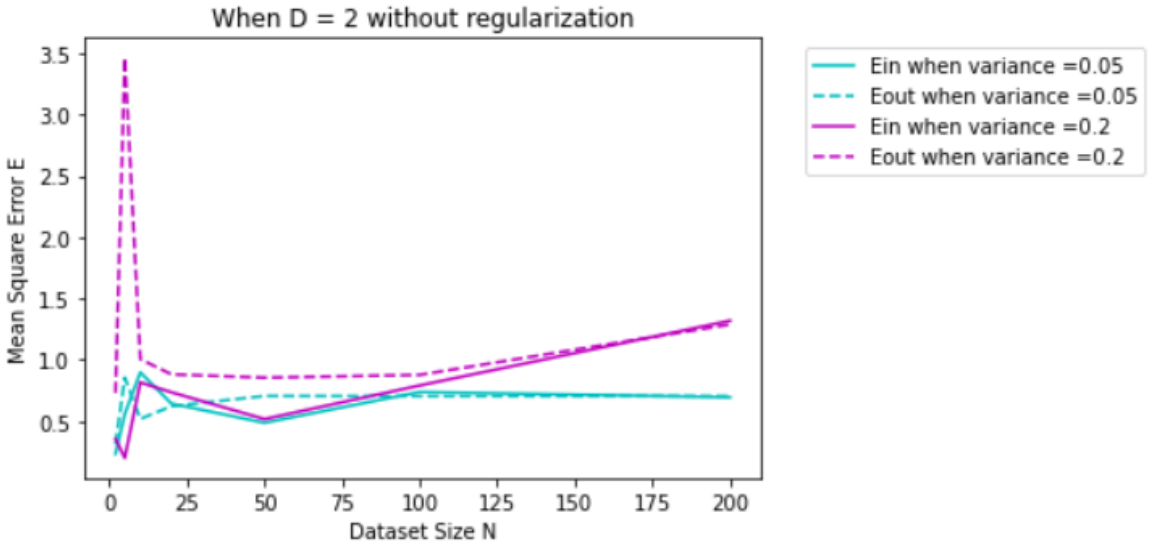




4. Effect of Noise on the model

In both the graph, we can see that as the variance i.e. the noise in the data increase the accuracy of the model decrease. The graph with variance 0.2 has more error than the graph with variance 0.05.

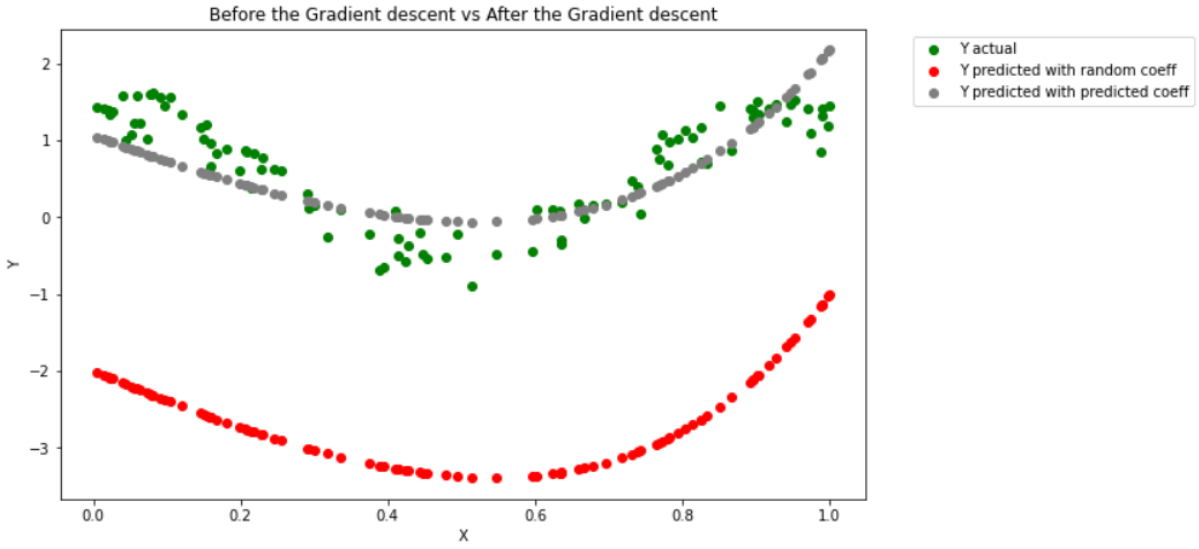


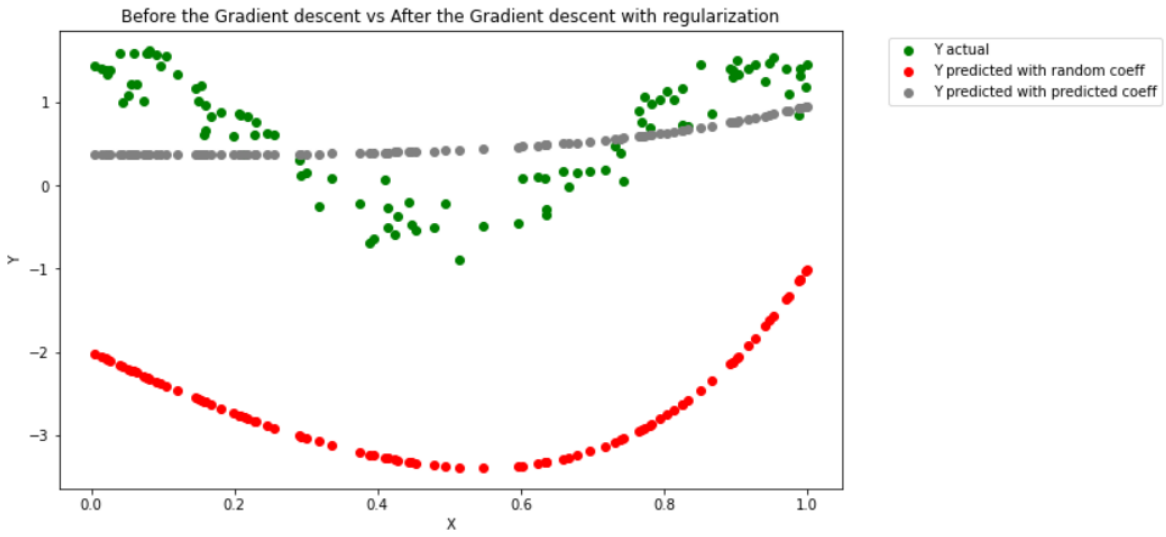


5. The value of Lambda for lasso regularization

The value of lambda for lasso should be greater than 0, but when I kept the value of lambda at 0.01, the MSE of the model was quite high for both the train samples and the testing sample (greater than 10). Later on the value was set to 1, only then did the value of MSE subside, improving the accuracy of the model. So I assume that we also need to choose the lambda value very carefully as it can also have an effect on the model accuracy.

6. Effect of weight decay with L1 regularization (Lasso Regularization)

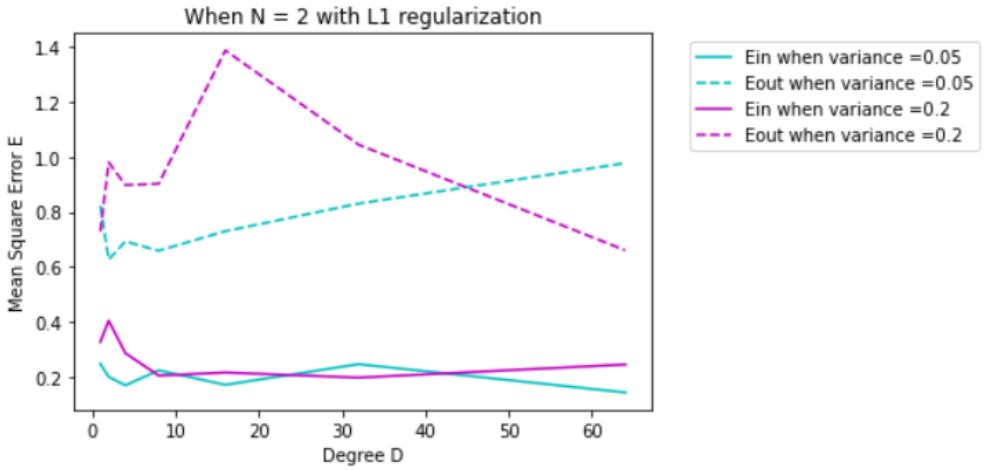


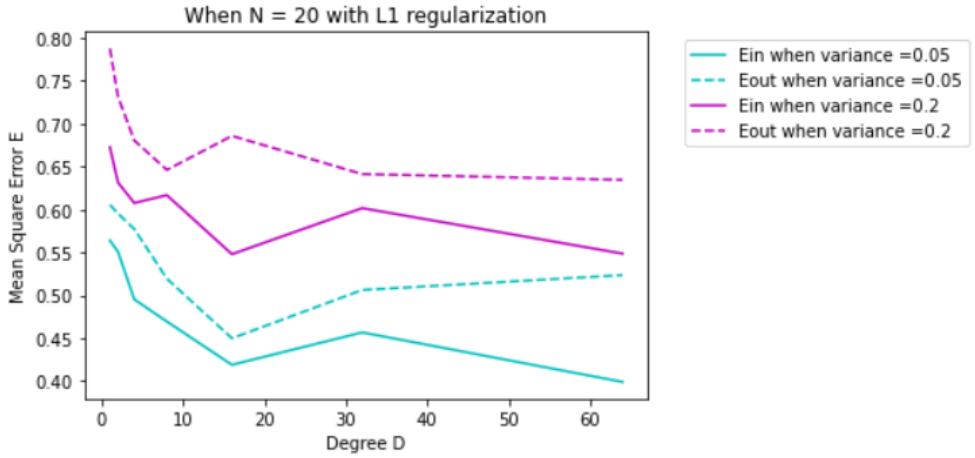


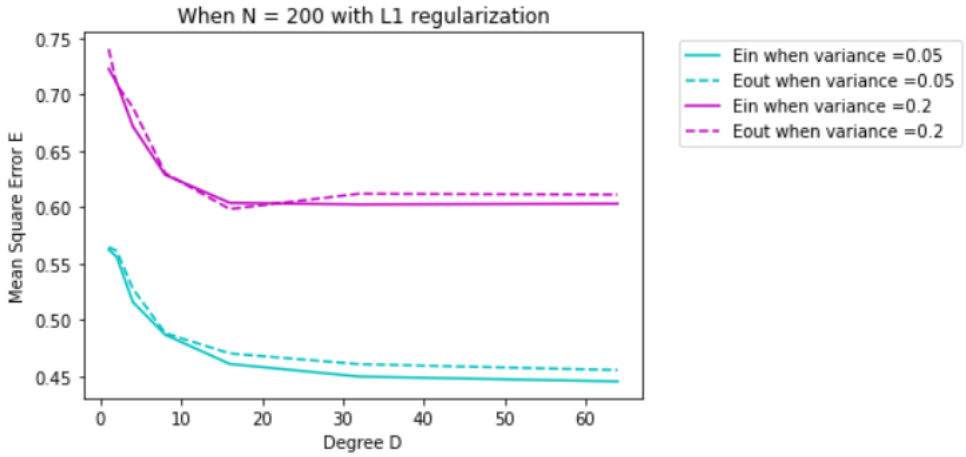
These graphs show us the effect of weight decay along with gradient decent. The other parameters such as the polynomial degree(d=4), the variance(variance=0.05), random coefficient generated at the start and the dataset are kept constant. In the first picture, we are able to see that without regularization, Y predicted with predicted coefficient line has aligned as per the given dataset. So we can say that if there is a lot of noise in the dataset then the line will also align with the noise. But in the second graph, the final line is more generalized as compared to earlier one. This makes the model more accurate as compared to a non regularized model.

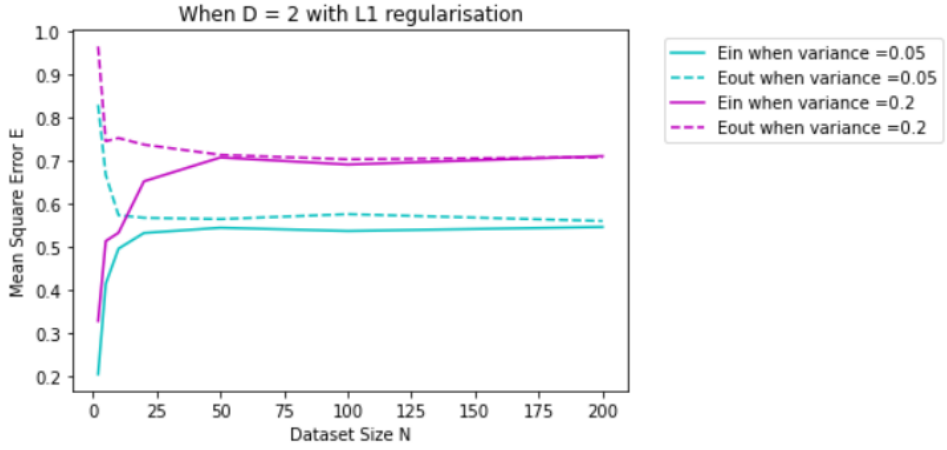
Thus, we can come to the conclusion that regularization is a crucial step for the training of the model as it makes the model more generalized and less dependent on the training dataset.

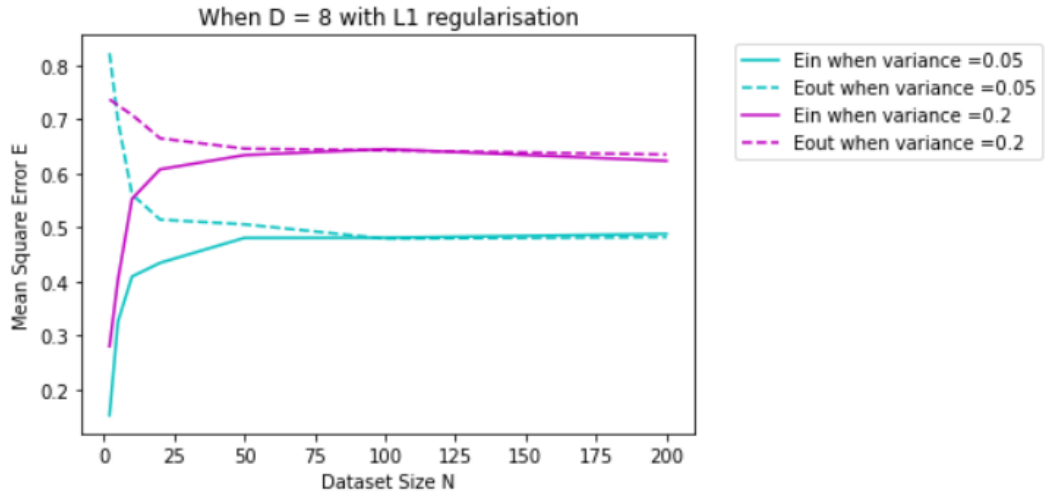
Below are some MSE vs Degree and MSE vs Sample Size graphs with L1 regularization.

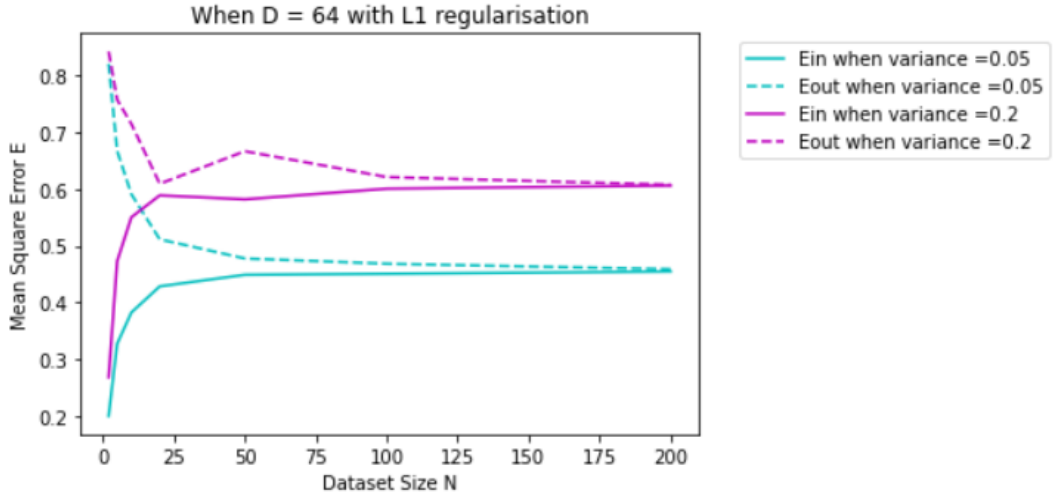



When N = 20 with L1 regularization
Mean Square Error E
Degree D
Ein when variance =0.05
Eout when variance =0.05
Ein when variance =0.2
Eout when variance =0.2


When N = 200 with L1 regularization
Mean Square Error E
Degree D
Ein when variance =0.05
Eout when variance =0.05
Ein when variance =0.2
Eout when variance =0.2


When D = 2 with L1 regularisation
Mean Square Error E
Dataset Size N
Ein when variance =0.05
Eout when variance =0.05
Ein when variance =0.2
Eout when variance =0.2


When D = 8 with L1 regularisation
Mean Square Error E
Dataset Size N
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
25
50
75
100
125
150
175
200
Ein when variance =0.05
Eout when variance =0.05
Ein when variance =0.2
Eout when variance =0.2


When D = 64 with L1 regularisation
Mean Square Error E
Dataset Size N
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
25
50
75
100
125
150
175
200
Ein when variance =0.05
Eout when variance =0.05
Ein when variance =0.2
Eout when variance =0.2

Conclusion

I see this assignment as an opportunity to learn and practice the fitting and generalization of regression models….Machine Learning emphasizes more on the model fitting. If the fitting of the model is amiss, then the model will lack any practical value due to its inability to predict accurate values.

Also the accuracy depends on a lot of factors. Some factors can be controlled/handled, for instance, the value of parameters like lambda, learning rate, the degree of polynomial can be adjusted, while some factors are out of our control, like the noise in the data(noise can be reduced but not removed), the sample size (if for a particular problem there are not enough samples). It is very difficult to get an ideal model, because a lot of factors affect it. And even if we get a model which is close to the ideal model, it is very difficult to generalize it. So in the end, it mainly comes down to bias variance trade off. The model can't be too simple (with degree 1) which may result in high bias and low variance. Nor the model can be too complex with low bias but high variance. So we have to select a model which has a balance between both the bias and variance.

Reference

- https://www.youtube.com/watch?v=H8kocPOT5v0&t=66s – Polynomial Regression in Python
- https://www.youtube.com/watch?v=Okwa-URH1cs - Automatic Differentiation with PyTorch — Topic 63 of Machine Learning Foundations
- https://www.youtube.com/watch?v=gsfbWn4Gy5Q - Gradient Descent From Scratch in Python - Visual Explanation
- https://stackoverflow.com/questions/8815706/random-number-with-specific-variance-in-python
- https://towardsdatascience.com/polynomial-regression-gradient-descent-from-scratch-279db2936fe9
- https://medium.com/@ms_somanna/guide-to-adding-noise-to-your-data-using-python-and-numpy-c8be815df524
- https://medium.com/@iamsamng/learning-derivative-function-for-polynomials-with-pytorch-818d85e9c9ae
- https://machinelearningmastery.com/using-autograd-in-pytorch-to-solve-a-regression-problem/
- https://www.youtube.com/watch?v=9lRv01HDU0s - Tutorial 27- Ridge and Lasso Regression Indepth Intuition- Data Science
- https://benihime91.github.io/blog/machinelearning/deeplearning/python3.x/tensorflow2.x/2020/10/08/adamW.html
- https://d2l.ai/chapter_linear-regression/weight-decay.html
- https://www.statease.com/blog/understanding-lack-of-fit-when-to-worry/