CS F320: Foundations of Data Science
Assignment 3 Report

Siddarth Gopalakrishnan - 2017B3A71379H
Rohan Maheshwari - 2017B4A70965H
Sai Satvik Vuppala - 2017B4A71449H

Contents

Data Description

The dataset which was provided had three features which are age, BMI, and the number of children of an individual. We were asked to drop the 'number of children' feature before preprocessing the data. Now, the features that are remaining i.e., age and BMI we are to predict the insurance amount for that person by constructing matured polynomial features for polynomials of degrees varying from 1 to 10, and by using the Gradient Descent/ Stochastic Gradient Descent we are to optimize the weights.

```
df.describe()
```

	age	bmi	charges
count	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	13270.422265
std	14.049960	6.098187	12110.011237
min	18.000000	15.960000	1121.873900
25%	27.000000	26.296250	4740.287150
50%	39.000000	30.400000	9382.033000
75%	51.000000	34.693750	16639.912515
max	64.000000	53.130000	63770.428010

where age and BMI are the feature variables and charges is the target attribute which we have to predict.
There are a total of 1338 data points.

Pre Processing

Variables that are measured at different scales do not contribute equally to the model fitting & learned function and might end up creating a bias. We can clearly see from the above data description that min and max values of the features differ widely. Hence to solve this problem we standardized all the features of the dataset to have mean 0 and unit variance (μ=0, σ=1).

Standardization:

$$z = \frac{x - \mu}{\sigma}$$

with mean:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

and standard deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Models

We used 2 models for finding the best fit line for the given data:

1. Gradient Descent Algorithm

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.

We have to do polynomial regression which can be mathematically formulated as:

Simple Linear Regression

$$y = b_0 + b_1 x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ 1 & x_3 & x_3^2 & \cdots & x_3^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

We will use RMSE error function which is:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

(For simplicity sake, we will present the model for univariate polynomial regression, we can easily extend it to multivariate regression)

Representing in vector format we get:

$$h(x) = [theta_0\ theta_1\ theta_2\ \dots theta_n] * \begin{bmatrix} 1 \\ x \\ x^2 \\ . \\ . \\ . \\ x^n \end{bmatrix}$$

Error function:

$$J(theta_0, theta_1, theta_2, \dots, theta_n) = \frac{1}{2m} * \sum_{i=1}^{m} \left(h(x^i) - y^i\right)^2$$

2. Stochastic Gradient Descent

Similar to gradient descent but instead of going through all the training data in each iteration, we select a random data point in each iteration and use that to update our parameters.
We shuffle the data points before applying SGD every time to obtain a less biased estimation of the true gradient.

REGULARIZATION

We implemented two types of regularizations namely L1(Lasso) and L2(Ridge).

Lasso Regression : The cost function for Lasso regression can be written as:

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}|w_j|$$

Ridge Regression : The cost function for Ridge regression can be written as:

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}w_j^2$$

OBSERVATIONS

RMSE VALUES :

WITHOUT REGULARIZATION

	GRADIENT DESCENT		STOCHASTIC GD	
Degree\Model	Min Train	Min Test	Min Train	Min Test
1	0.9456180639520197	0.9499672876438658	0.9465482388663492	0.8829339395706615
2	0.9432960061153157	0.9567914879057 29	0.9384480366192 91	0.8957279743947091
3	0.9425972949844833	0.9615181376091 25	0.9392482950902721	0.8969149329005719
4	0.9419702786280362	0.9646137884049664	0.9402916060511411	0.8951110432799042
5	0.9412376140989336	0.9696631991229393	0.9408892059207671	0.8932584529845138
6	0.9406793268649535	0.9750285330594713	0.9407310062066637	0.8916106330840712
7	0.9402243809059487	0.9783280681763415	0.9398779923891036	0.8903683945836797
8	0.9396769270699264	0.9799918766130 31	0.9386732759254579	0.8898998431235676
9	0.9389332054970614	0.9810370558526068	0.9375006492783199	0.8905520350555932
10	0.9380033777142605	0.9822490661567592	0.9365990149602285	0.8924434265924285

LASSO REGULARIZATION (using optimal lambda values)

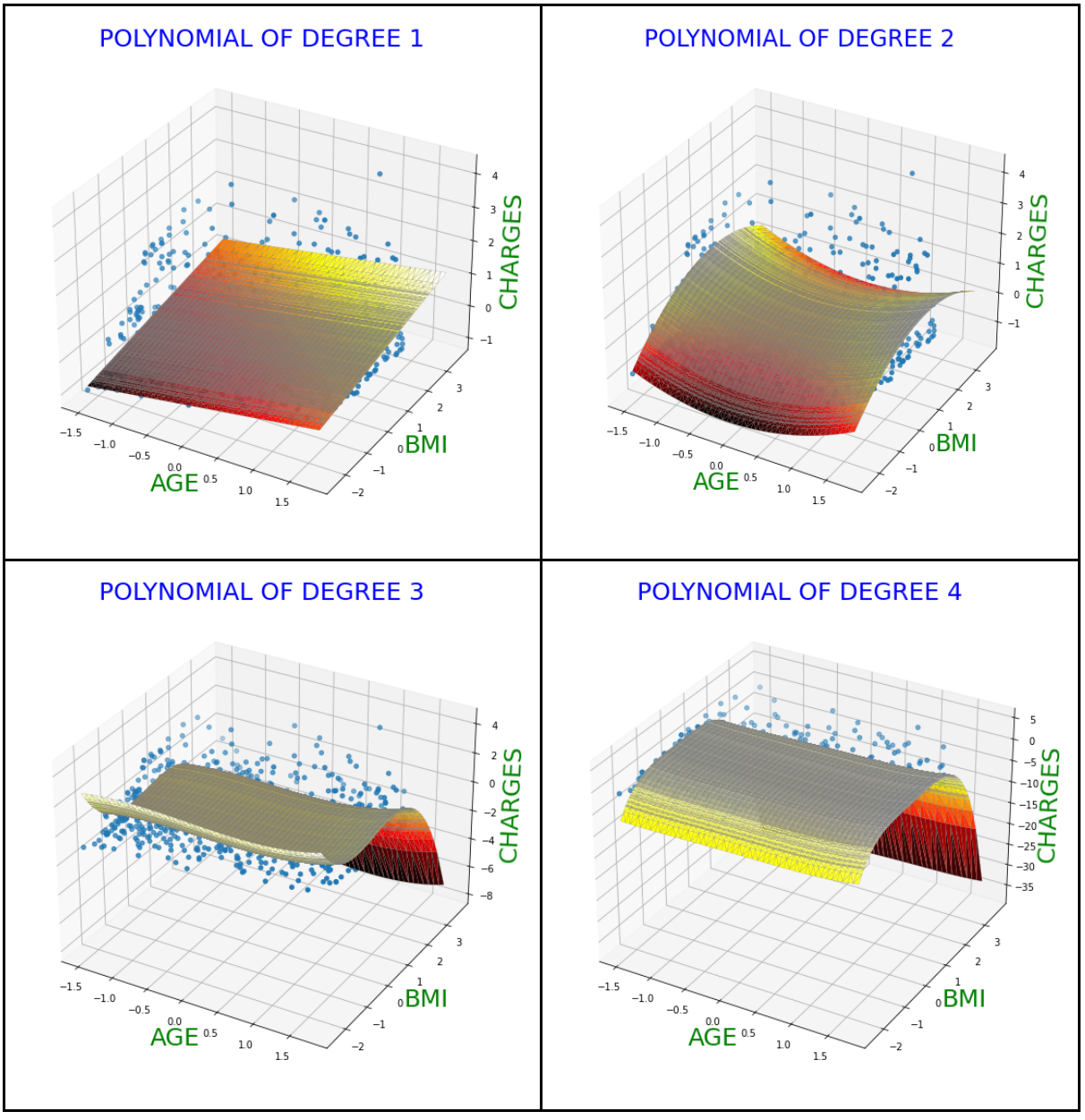
	GRADIENT DESCENT			STOCHASTIC GD		
Degree \Model	Min Train Error	Min Validation Error	Min Test Error	Min Train Error	Min Validation Error	Min Test Error
1	0.9456180817973235	0.9112137010676232	0.9499625318764741	0.9465640800542509	0.9706535184640586	0.882925386395018
2	0.9435593367441211	0.9134240376300856	0.9554838577569136	0.9384734603674185	0.9650698351895126	0.8955122055848045
3	0.9426713727303841	0.9163617388716591	0.9606543526027445	0.9392572914629879	0.9663098166319051	0.8966756765518956
4	0.9419937514251548	0.9169314559668729	0.9645563099188326	0.9403043724182436	0.9677242537516378	0.8948764249744695
5	0.9412739337039863	0.9166197837367736	0.9693168135561216	0.940906321319547	0.9688683613404423	0.8930305881083777
6	0.9407684767996691	0.9162631249475734	0.973605766189058	0.9407682198805576	0.9694538167308391	0.8914567161865676
7	0.9404021821479133	0.9160564603155067	0.9769116613495424	0.9399698027751173	0.9695074662292076	0.8903118976920636
8	0.9402158626605572	0.9162018035659655	0.9775047934671881	0.9388074618138047	0.9692612511456844	0.8898378466661581
9	0.9398060249412411	0.9168674930084791	0.9791847426516187	0.9376481683227088	0.9690096603241324	0.8903742627823131
10	0.9392769866555492	0.9178782330268501	0.980443406107729	0.9367311647433226	0.9689281787738755	0.8920522822291146

RIDGE REGULARIZATION (using optimal lambda values)

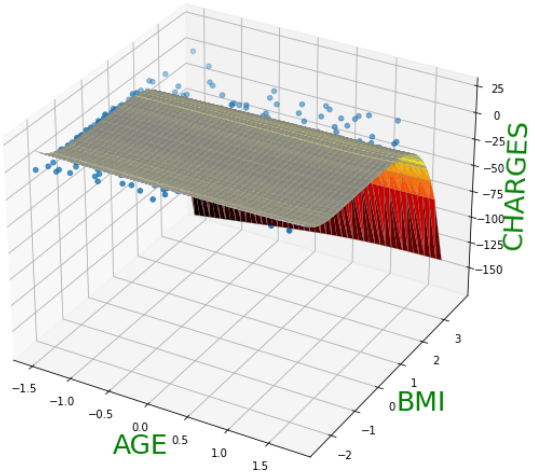
	GRADIENT DESCENT			STOCHASTIC GD		
Degree \Model	Min Train Error	Min Validation Error	Min Test Error	Min Train Error	Min Validation Error	Min Test Error
1	0.9456180 6775733	0.91121177 82457758	0.9499681 0153131	0.9465517 425625369	0.970644883 6098812	0.88293201 24644735
2	0.9433813 3371136	0.91371973 21247346	0.9562508 2623457	0.9384485 568431219	0.965079580 4522024	0.89572322 92376156
3	0.9426174 2186590	0.91635974 36194302	0.9609556 3587161	0.9392489 581859781	0.966340340 339528	0.89689116 67933934
4	0.9420061 715135636	0.91677326 4668574	0.9641279 31884823	0.9402926 488042009	0.967761224 4079768	0.89509636 12421129
5	0.9412892 259180122	0.91658866 77768263	0.9689641 157545286	0.9408894 942053714	0.968913678 4835916	0.89325716 11925831
6	0.9406853 992860448	0.91627154 37431326	0.9749081 742194454	0.9407314 991754817	0.969500355 0894884	0.89160974 84179609
7	0.9402716 146031666	0.91629764 05314684	0.9776332 122224958	0.9398786 79542826	0.969539036 3991196	0.89036763 21963963
8	0.9397466 874143796	0.91703568 01688861	0.9795061 5037248	0.9386740 691628233	0.969298582 0499731	0.88989885 64002482
9	0.9390426 499803493	0.91847035 20889854	0.9806744 278441218	0.9375014 269345492	0.969077489 6358725	0.89055045 10173409
10	0.9381617 130650148	0.92029113 44100865	0.9818736 47224909	0.9365996 793204203	0.969044101 6890855	0.89244094 32685153

SURFACE PLOTS OF THE PREDICTIONS

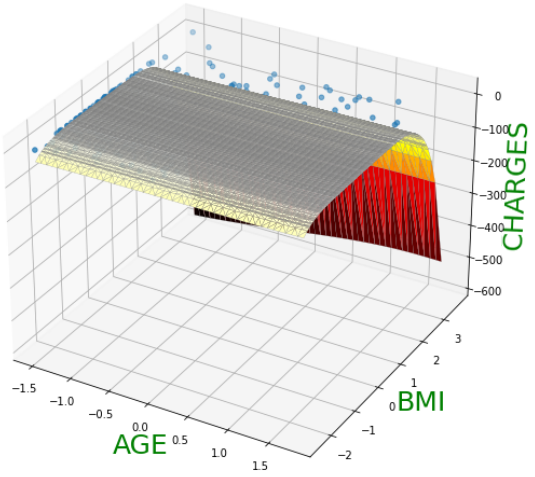
- Gradient Descent without regularisation



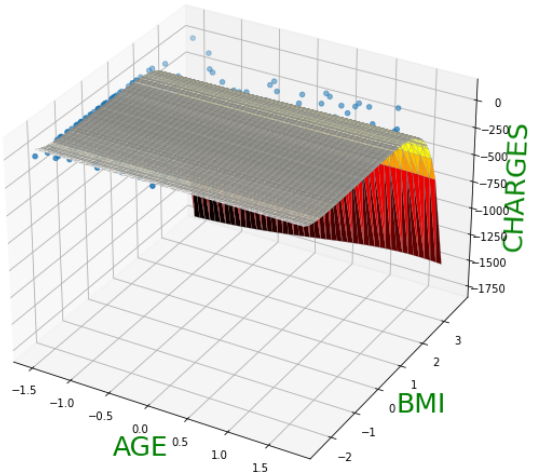
POLYNOMIAL OF DEGREE 5



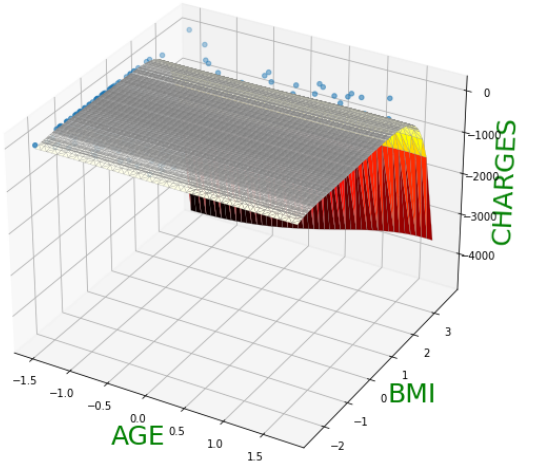
POLYNOMIAL OF DEGREE 6



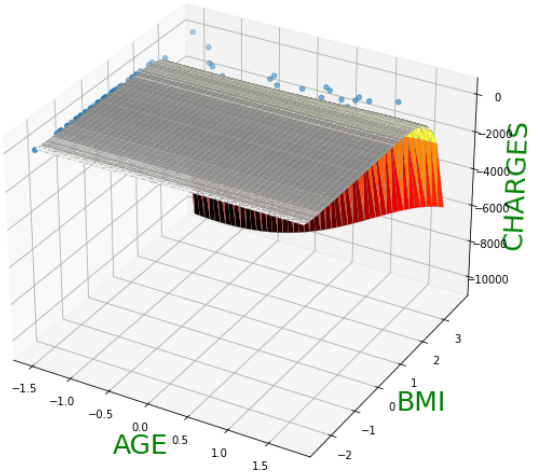
POLYNOMIAL OF DEGREE 7



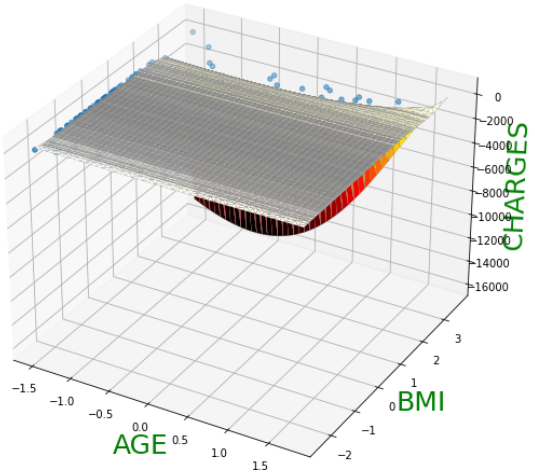
POLYNOMIAL OF DEGREE 8



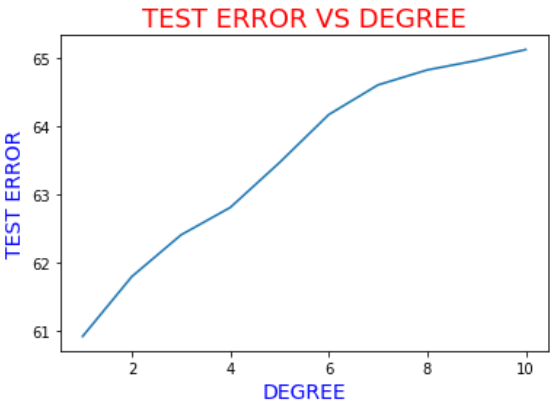
POLYNOMIAL OF DEGREE 9



POLYNOMIAL OF DEGREE 10



As we can clearly see from the above plots, **as the degree of the polynomial is increasing** our model is fitting the training data points more accurately in effect fitting the noise in the dataset and not generalising well i.e **overfitting**. This can be clearly seen from the following plots:



RESULTS

- *On using polynomials of higher degree*

  First, let us consider our problem is to fit the train data. The general trend is that on increasing the order of the polynomial the fitted curve gets closer to the train data points, but increase it beyond an extent will lead to overfitting of train data points, where the curve may pass through almost all the train data points but may fluctuate drastically in between them. This can be observed when we use test on the test data (which is not part of the train data), then the general trend is that the test error goes down initially up to an extent and the test error starts increasing as the degree of the polynomial increases as it overfits the train data and is unable to estimate the test data.

- *Does a single global minima exist?*

  Yes. This is because we have taken our error function to be the sum of squares of error values, due to which, it is always the case that the eigenvalues of the Hessian matrix are positive, due to this, our error function stays convex. Hence we can say that there is always a single global minima.

- *Preventing Overfitting*

  For our case Ridge regression works better as there are a large number of parameters about the same value i.e. almost all the predictors influence the target variables.

  In this report we used the two most used regularisation techniques i.e., L1 or Lasso regularisation and L2 or ridge regularisation. As a penalty term (λ), the L1 regularization adds the sum of the absolute values of the model parameters to the objective function whereas the L2 regularization adds the sum of the squares of the model parameters. It can also be noticed that as we increase the value of λ, the weights take values that are approaching zero, but if you see in the case of lasso regularisation, even at smaller λ's, our coefficients are reduced to absolute zeroes. Therefore, lasso selects only some features while reducing the weights of others to zero. This property is absent in the case of ridge regularisation. So from above and generally we can see that L2 or ridge is preferable for models with lower dimensions and L1 or lasso is better suited for the models with higher dimensions where it leads to sparse models (models with less non-zero parameters).

  Lasso tends to do well if there are a small number of significant parameters and the others are close to zero (ergo: when only a few predictors actually influence the response).

- *Regularisation Parameter*

  Regularization is applied as it keeps the weights small making the model simpler and thereby avoiding overfitting. λ is the penalty term or regularization parameter that determines to what extent the weights are to be penalized. When λ is zero then the regularization term becomes zero. As the magnitudes of the fitting parameters increase, there will be an increasing penalty on the cost function. This penalty is dependent on the squares of the parameters as well as the magnitude of λ. As we increase the value of the λ (> 2), the magnitude of the weights decreases i.e., the optimization function will have to choose smaller weights (where the values reach zero but not absolute zero) in order to minimize the total cost.

- *Is regularisation for a large number of features and limited training instances?*

  Yes, we agree that regularization is necessary when you have a large number of features but limited training instances. As given if there are low numbers of training samples, our polynomial regression model tries to fit perfectly to these training data points and thus the coefficients of the features also become large. When the same weights are used to make predictions on testing data, the variance is very high, this occurs due to overfitting of data where the model tries to fit perfectly for the training data but fluctuates drastically between them, so when an unknown test set is tested it thereby produces high variance. Therefore, a regularization term is introduced to penalize the large parameters that lead to overfitting.

- *Parameters in polynomial having degree N with D number of features*

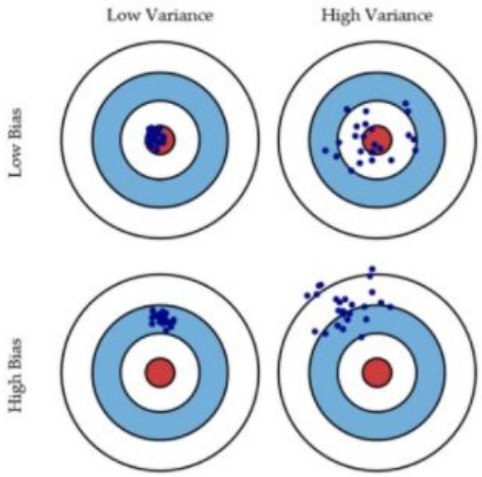
  If we are provided with D number of original features and we are asked to generate new matured features of degree N, the number of matured features we will be able to generate are:

  $$(D,N)= (N+D)\ C\ (N)$$

- *Bias-variance tradeoff*

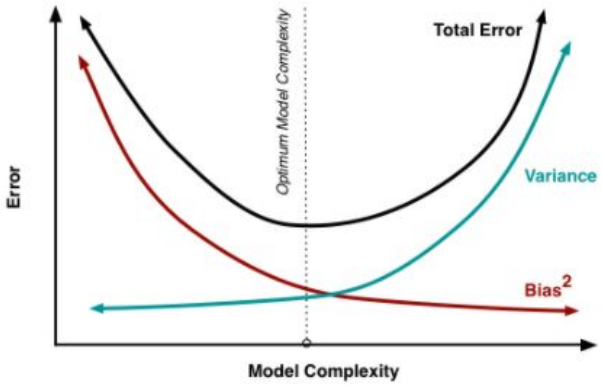
  - The *bias error* is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
  - The *variance* is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

Let us understand bias and variance with the help of a diagram, so first we assume that our model is very accurate, in terms of fitting the train and test set well, implying that the error that would be incurred would be low, meaning a low bias and low variance model as seen in the first figure. All the data points accurately fit inside the bulls-eye. Similarly, we can see in the figure that if the variance increases, the spread of our data point increases which in turn reduces the accuracy of the prediction. And if we increase the bias, the error between our predicted value and the observed values increases.



Assuming the case of a polynomial regression model, if the model is unbiased, it passes through each and every sample point and hence will have unbounded variation. On the other hand, if we consider a model that underfits our sample data points, the sum of squared errors will increase, but the variance of the model itself, would be low as there is a uniformity in our model. **This implies that bias and variance are inversely proportional.**

Now as we are considering a polynomial model, as we add more and more parameters to our model, the complexity increases, resulting in the increase of variance and decrease of bias, as the model passes through all the data points, thereby leading to overfitting of the data. So as we can see in the below graph we need to find out one optimum point in our model where the decrease in bias is proportional to increase in variance. Regularization helps us find this central point.



In the case of overfitting, our parameters could take extreme values, which doesn't accurately represent the impact of each feature on the target. Due to this, the model has many peaks and valleys while passing through all the data points. In order to curtail the freedom these parameters have in the case of overfitting, we introduce a regularization term which puts a limit/controls the values which these parameters could take.

.