ASSIGNMENT - 1

Prabhav Bansal & Dhruv Sharma
Team number : 79
Prof.Praveen Paruchuri
Machine and Data Learning
14/02/2020

Question - 1

1.1 **INTRODUCTION**

We are given a problem statement where we have to tabulate the biases and variances of the predicted functions to given training and test data set. We have been provided a dataset consisting of pairs (x_i , y_i). Then we were supposed to split the dataset into training and testing (90:10 split) sets .After that we were asked to divide the train set into 10 equal parts randomly, so that we get 10 different dataset to train our model. Further we have to train a linear classifier on each of the 10 train sets separately, so that we have 10 different classifiers or models. So now we have 10 different models or classifiers trained separately on 10 different training set, and then at last we should calculate the bias and variance. We need to repeat the above process for the following class of functions.

$$y = mx$$

$$y = ax^2 + bx + c$$

$$y = ax^4 + bx^3 + cx^2 + dx + e$$

And so on up till polynomial of degree 9. We were only supposed to use **sklearn's linear_model.LinearRegression().fit()** for finding the appropriate coefficients with the default parameters.

1.2 ALGORITHM DEFINITION

Linear Regression : Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output).

Hypothesis Function for Linear Regression:

$$Y = \theta_1 + \theta_2 * X$$

While training the model we are given :

X : Input Training data (univariate – one input variable(parameter))

Y : Labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

θ_1 : intercept

θ_2 : coefficient of x

Once we find the best θ_1 and θ_2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

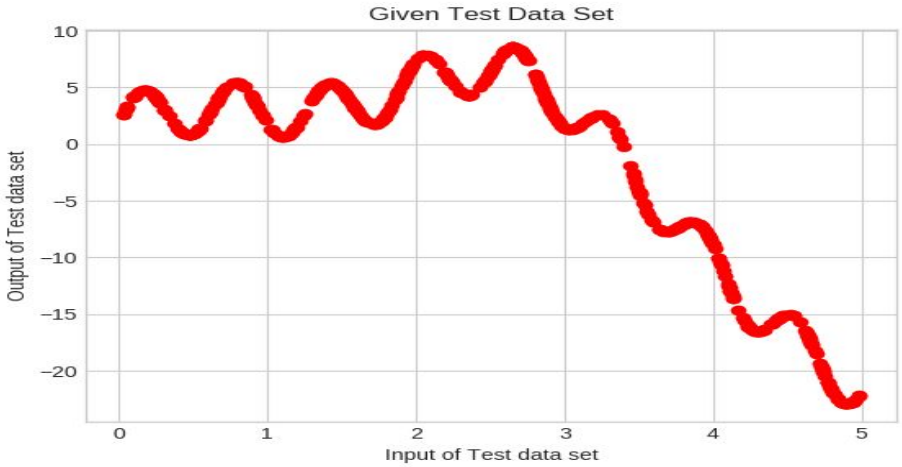
Cost Function (J) : By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ_1 and θ_2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y). Cost function(J) of Linear Regression is the Root Mean Squared Error (RMSE) between predicted y value (pred) and true y value (y).

Gradient Descent: To update θ_1 and θ_2 values in order to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ_1 and θ_2 values and then iteratively updating the values, reaching minimum cost.

1.3 **EXPERIMENTAL EVALUATION**

Methodology: Following are steps to proceed to given problem statement and to get answer.

- ***Loading the Data :*** We start by loading the data into Numpy Arrays and Pandas DataFrames, and plotting the basic structure of it all. Some sampling, shuffling and splitting is done as we were tasked in the problem statement.



- ***Building the Models*** **:** We start off writing the models and getting their bias and variance in our DataFrames and Matrices.

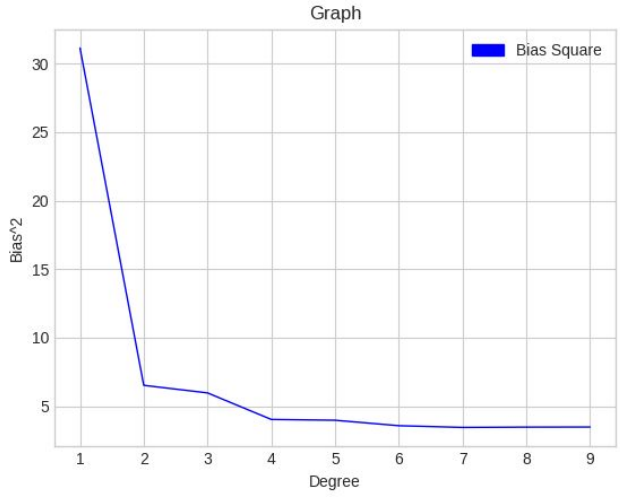
  For getting **bias** the following function can be used :

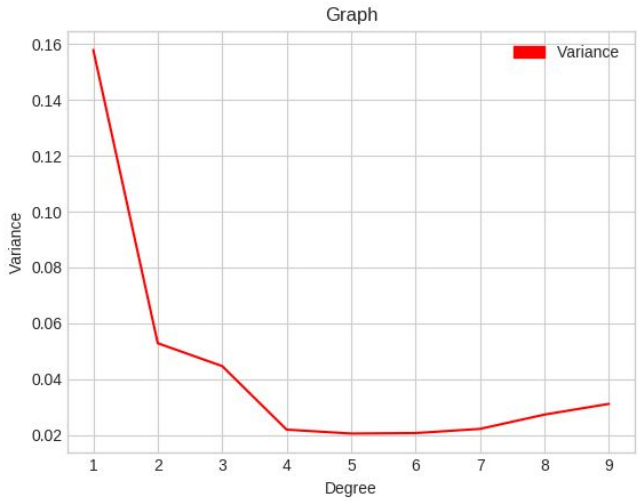
```
def get_bias(degree):
    res = 0

    for i in range(len(y_test)):
        res += ((np.mean(np.array(predictions[degree - 1])[:, i:i+1]) - y_test[i]) ** 2)

    return res/len(y_test)
```

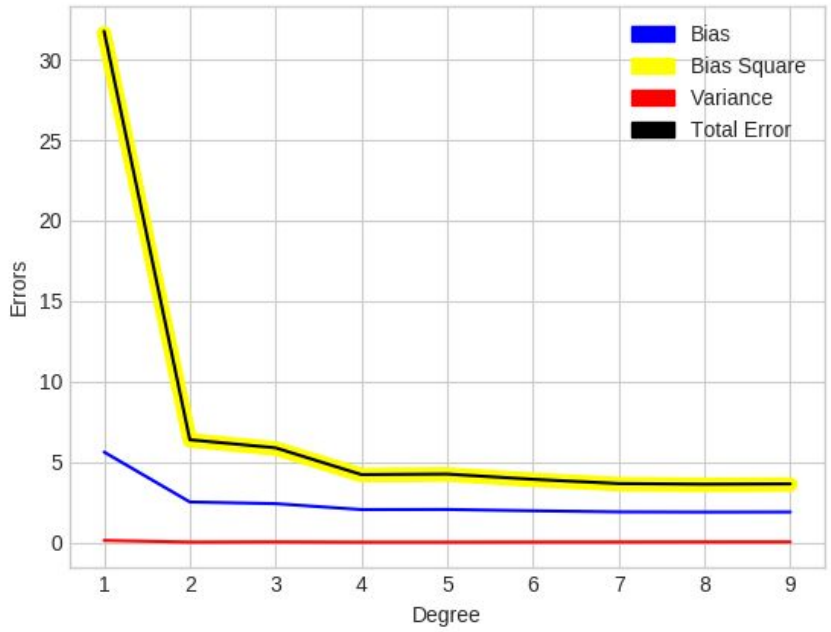
  For getting **variance** the following function can be used :

```
def get_variance(degree):
    a = (np.mean(np.var(predictions[degree - 1],axix = 0)))
    return a
```

- ***Analyzing Bias and Variance:*** We are plotting Bias and Variance for each model and against model size to see if the trends are satisfied. We expect that * *The Smaller Models will be High Bias and Low Variance since it has very little space to produce varying outputs, i.e. The function it learns will be simple. But bias is huge cause it didn't really learn a lot.* * The Bigger Models will be High Variance and Low Bias since it can overfit the data getting the mean almost perfectly equal, but have huge deviations due to learning too complex a function on different inputs.

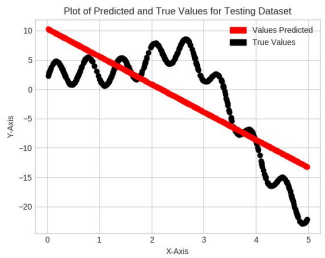


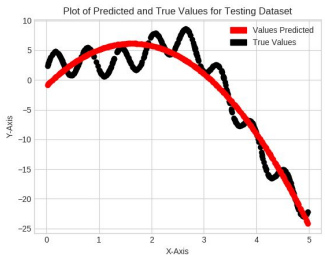


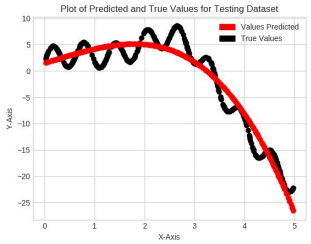


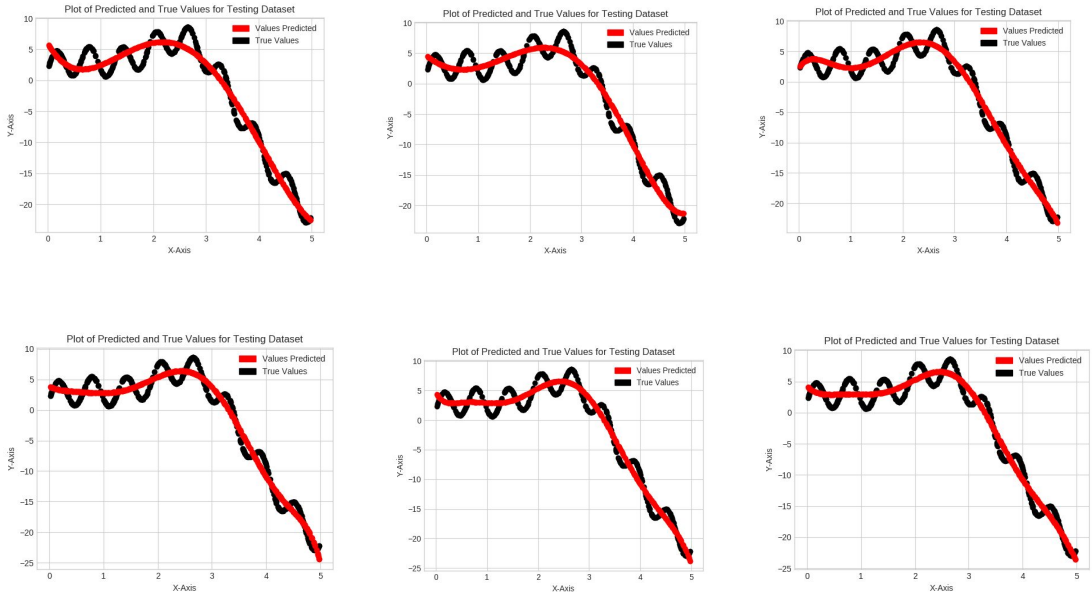
Since here in the given test data set, the value of variance is negligible with respect to Bias^2. Hence, We get almost coincident lines representing Total Error and Bias^2. \

- ***Check the Model's Predictions:*** One final run to see how the models, on average are fitting the data. Here we have checked the models trained on 90 parts of data set i.e. whole training set rather than splitting training set into 10 sets. This is because if we check for all the parts of training sets then we will get 90 different models and that's very tedious to present on a report. The basic idea of training a model is the same.. The difference lies in the fact of splitting the training data set into 10 parts.







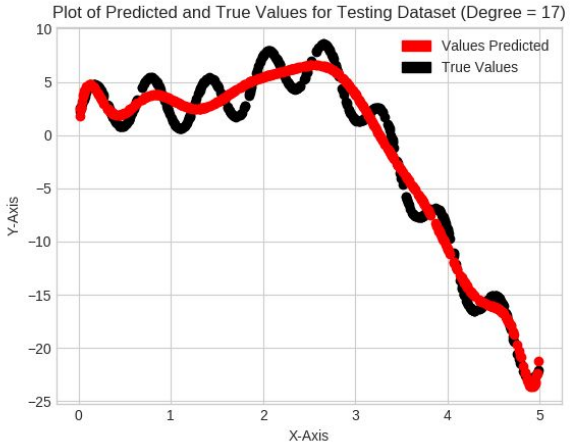


1.4 CONCLUSION

The **Variance is bizarrely decreasing with the increase in complexity** of the model. This is because of the way the dataset is structured. There is little to **no noise in the data**, all the data follows a very regular distribution, which is approximated by all the models almost equally well and in the same fashion. This would be very different if there was a lot of noise in the data.

The **Smallest Agent are both High Bias and High Variance** as they can neither approximate well nor correctly, the approximations keep changing. The Wavy nature of the curve allows for multiple lines of best fit.

We also see that **after decreasing the variance increases a little bit**. This is where the models are actually becoming powerful, and are overfitting a little. However, until we take a 17 degree model, we will not overfit too much.



Question 2

2.1 INTRODUCTION:

We are given a problem statement where we have to tabulate the biases and variances of the predicted functions to given training and test data set. We have been provided a training dataset and test data set. We will have a model trained on training data set and calculate the bias and variance on test data. We need to repeat the above process for the following class of functions.

$$y = mx$$

$$y = ax^2 + bx + c$$

$$y = ax^4 + bx^3 + cx^2 + dx + e$$

And so on up till polynomial of degree 9. We were only supposed to use **sklearn's linear_model.LinearRegression().fit()** for finding the appropriate coefficients with the default parameters.

We have been given 20 subsets of training data containing 400 samples each. For each polynomial, we created 20 models trained on the 20 different subsets and found the variance of the predictions on the testing data. Also, we found the bias of trained models on the testing data. Finally plotted the bias-variance trade-Off graph. We applied the formula for bias and variance for a single input but since the testing data contains more than one input, we took the mean wherever required.

2.2 ALGORITHM DEFINITION

Linear Regression : Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output).

Hypothesis Function for Linear Regression:

$$Y = \theta_1 + \theta_2 * X$$

While training the model we are given :

X : Input Training data (univariate – one input variable(parameter))

Y : Labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

θ_1 : intercept

θ_2 : coefficient of x

Once we find the best θ_1 and θ_2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

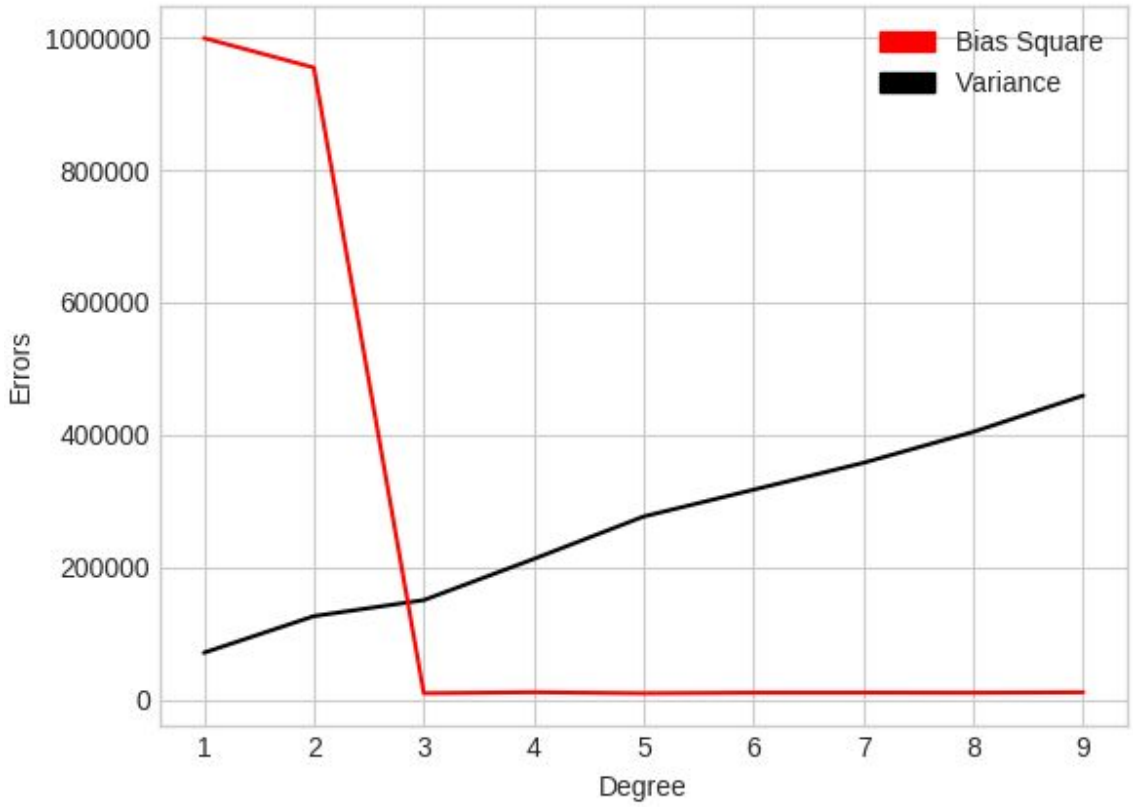
Cost Function (J) : By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ_1 and θ_2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y). Cost function(J) of Linear Regression is the Root Mean Squared Error (RMSE) between predicted y value (pred) and true y value (y).

Gradient Descent: To update θ_1 and θ_2 values in order to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ_1 and θ_2 values and then iteratively updating the values, reaching minimum cost.

2.3 **TABLE:**

Degree	Variance	Bias^2
1	70545.489146	999228.396872
2	125870.855549	954619.273794
3	150073.739546	9389.730117
4	212235.708325	10907.348134
5	276388.480255	9339.194291
6	316863.498437	10248.585941
7	357510.984757	10335.275862
8	404286.670686	10149.419244
9	459132.378372	10815.487037

2.4 **GRAPH :**



2.5 **EXPLANATION:**

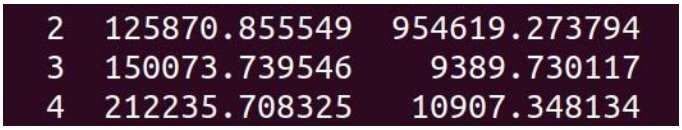
Bias : Bias is the simplifying assumptions made by the model to make the target function easier to approximate.
Variance : Variance is the amount that the estimate of the target function will change given different training data.

The following points explain what we can infer from the graphs that our model thrown:

- Initially when the **degree of polynomial is very less**, at that time ,our function is too simple and such a simple model can not fit our training dataset. So it **will cause under-fitting.** This will give very bad accuracy on training dataset therefore initially value is *bias is too high*. And at this time as model is not good therefore accuracy on test dataset is also very low i.e with simpler models accuracy is too low with both training and test dataset so *variance is too low* which in above graph is for degree 1 and 2.

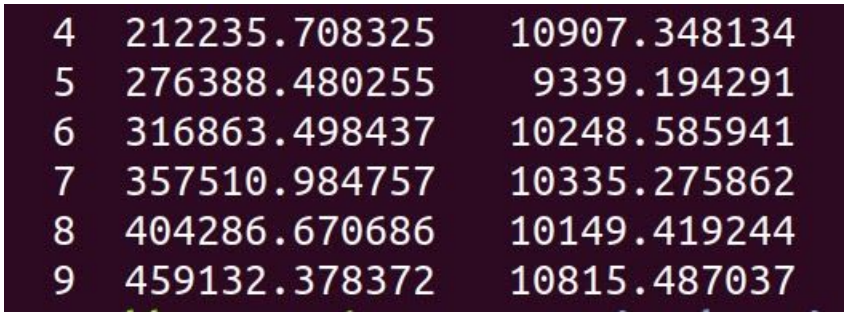
Degree	Variance	Bias^2
1	70545.489146	999228.396872
2	125870.855549	954619.273794

- As we go on **increasing the complexity of models** , **training accuracy goes on increasing till some limit** as models fit more accurately to training dataset. So at that time training accuracy increases therefore *bias decreases* and also test accuracy also goes on increasing as model is good. So both training and test accuracy are good at this time therefore the value of *variance is low* which in the above graph is near 3.



```
2  125870.855549  954619.273794
3  150073.739546    9389.730117
4  212235.708325   10907.348134
```

- As we go on further **complexity of model increases too much** which gives **very high accuracy** on training dataset **but this model is overfitted model** because for this model training accuracy is too high but *test accuracy is very bad.* This is because as we increase the complexity of the model it tries to best fit the training data points. Therefore due to very large complexity it tries to exactly fit the training data points which results in very bad over-fitted model which give very bad test accuracy. Therefore at this point *bias is too low* but *variance is too high* because of the large difference between training and test accuracy which in above graph is 4 onwards.



```
4  212235.708325   10907.348134
5  276388.480255    9339.194291
6  316863.498437   10248.585941
7  357510.984757   10335.275862
8  404286.670686   10149.419244
9  459132.378372   10815.487037
```

Thanks

****