# Machine And Data Learning


## Nipun Tulsian (2021101055)


## Assignment-2

## 2.1 Task 1: Linear Regression

The method LinearRegression().fit() is a function used in machine learning for linear regression, a method used to model the relationship between a dependent variable and one or more independent variables.

When called on an instance of the LinearRegression class, the fit() function trains the model by fitting a linear equation to the training data, which involves determining the coefficients for the equation that best fits the data. These coefficients are determined using a method called ordinary least squares (OLS) regression, which minimizes the sum of the squared differences between the predicted and actual values of the dependent variable.

After the model is trained using the fit() function, it can be used to make predictions on new data, by calling the predict() function on the model instance.

- ***LinearRegression()*** creates an instance of the class LinearRegression which is used as our Regression model.
- ***fit()*** is used to fit the model to the given dataset

## 2.2 Task 2 : Gradient Descent

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. In machine learning, we use gradient descent to update the parameters of our model.

**Learning Rate:**

The size of steps taken by gradient descent in the direction specified by the negative gradient is called learning rate. With a high learning rate we can cover more space each step, but we risk overshooting the lowest point. With a very low learning rate, we can

confidently move in the direction of the negative gradient since we are recalculating it so frequently. A low learning rate is more precise, but calculating the gradient is time-consuming.

**Cost Function:**

A Loss Functions tells us “how good” our model is at making predictions for a given set of parameters. The slope of this curve tells us how to update our parameters to make the model more accurate. It is typically represented as the mean squared error (MSE) between the predicted and actual values.

**Working:**

In the case of a single independent variable and a single dependent variable, the linear regression model is represented by the equation: y = mx + b ,

y is the dependent variable, x is the independent variable, b is the y-intercept, and m is the slope of the line. We need to consider the impact each one has on the final prediction, we need to use partial derivatives. We calculate the partial derivatives of the cost function with respect to each parameter and store the results in a gradient.

Given the cost function:

$$f(m,b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

The gradient can be calculated as:

$$f'(m,b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N}\sum -2(y_i - (mx_i + b)) \end{bmatrix}$$

The algorithm starts with an initial estimate for the coefficients, typically set to zero or a random value. Then, the gradient of the cost function is calculated with respect to each coefficient. The gradient gives the direction of the steepest ascent, and the negative of the gradient gives the direction of the steepest descent. The coefficients are then updated by subtracting a small step size multiplied by the gradient from the current

estimate. This process is repeated multiple times until the cost function reaches a minimum or until a maximum number of iterations is reached.

In each iteration, the gradient of the cost function is calculated using the partial derivatives of the cost function with respect to each coefficient. The partial derivatives give the rate of change of the cost function with respect to each coefficient, and the coefficients are updated in the direction of the steepest descent.

## 2.3.2 Task 3

| Degree | Bias | Variance | Irreducible Error | MSE | Bias Square |
|---|---|---|---|---|---|
| 1 | 0.26911 | 0.00689187 | -6.81122e-18 | 0.121405 | 0.114513 |
| 2 | 0.0863558 | 0.00107127 | -8.62921e-18 | 0.0132581 | 0.0121868 |
| 3 | 0.0331786 | 0.000378802 | -3.81275e-18 | 0.00508781 | 0.00470901 |
| 4 | 0.0250037 | 0.000414983 | 2.52541e-18 | 0.0046737 | 0.00425872 |
| 5 | 0.0244308 | 0.000447616 | -2.04003e-19 | 0.00468021 | 0.0042326 |
| 6 | 0.0244614 | 0.000698334 | 4.12656e-18 | 0.00493211 | 0.00423378 |
| 7 | 0.024705 | 0.000819595 | -2.57711e-18 | 0.00505721 | 0.00423761 |
| 8 | 0.024809 | 0.00128633 | -4.75175e-18 | 0.00555046 | 0.00426413 |
| 9 | 0.0255521 | 0.00178906 | 7.53841e-18 | 0.00604839 | 0.00425933 |
| 10 | 0.0273942 | 0.00350139 | 1.02141e-18 | 0.00778658 | 0.0042852 |
| 11 | 0.0314847 | 0.0365296 | 7.10543e-18 | 0.0416669 | 0.0051373 |
| 12 | 0.0351281 | 0.0627648 | -1.13687e-17 | 0.0693178 | 0.00655298 |
| 13 | 0.0355162 | 0.0194935 | -4.9738e-18 | 0.0256238 | 0.00613034 |
| 14 | 0.0379266 | 0.204352 | 0 | 0.211604 | 0.00725245 |
| 15 | 0.0506351 | 0.901883 | -5.91172e-16 | 0.914922 | 0.0130387 |

As functional classes change, with increasing degree we observe:

### Bias:

- After degree 1 bias takes a deep dive in values and then gradually decreases till degree 8 or 9.
- After that the bias increases unevenly till degree 14 and then it increases from there.
- For initial degree like 1 the bias is high so its a case of underfitting as it not able to capture underlying trend of data i.e. model is not able to learn enough from training data and so reduces accuracy.
- For higher degree bias decreases as it shifts towards overfitting as model tries to cover all data points so model starts catching noise and inaccurate values.

### Variance:

- How sensitive our model is for prediction with variation in test dataset.
- It keeps increasing with some troughs till degree *11*.
- Then it oscillates till degree *15 and then increases monotonically.*
- For initial degrees the variance is small as its case of underfitting and then increases as its slowly converts to overfitting model.

## 2.3.3 Task 4:

| Irreducible Error |
|---|
| -6.81122e-18 |
| -8.62921e-18 |
| -3.81275e-18 |
| 2.52541e-18 |
| -2.04003e-19 |
| 4.12656e-18 |
| -2.57711e-18 |
| -4.75175e-18 |
| 7.53841e-18 |
| 1.02141e-18 |
| 7.10543e-18 |
| -1.13687e-17 |
| -4.9738e-18 |
| 0 |
| -5.91172e-16 |

**Observation:**

- Running the program several times shows that no general trend is followed by the irreducible error
- All the values for the irreducible error are extremly close to each other
- Changes in the irreducible error of a particular model after every run is extremly small

**Conclusions:**

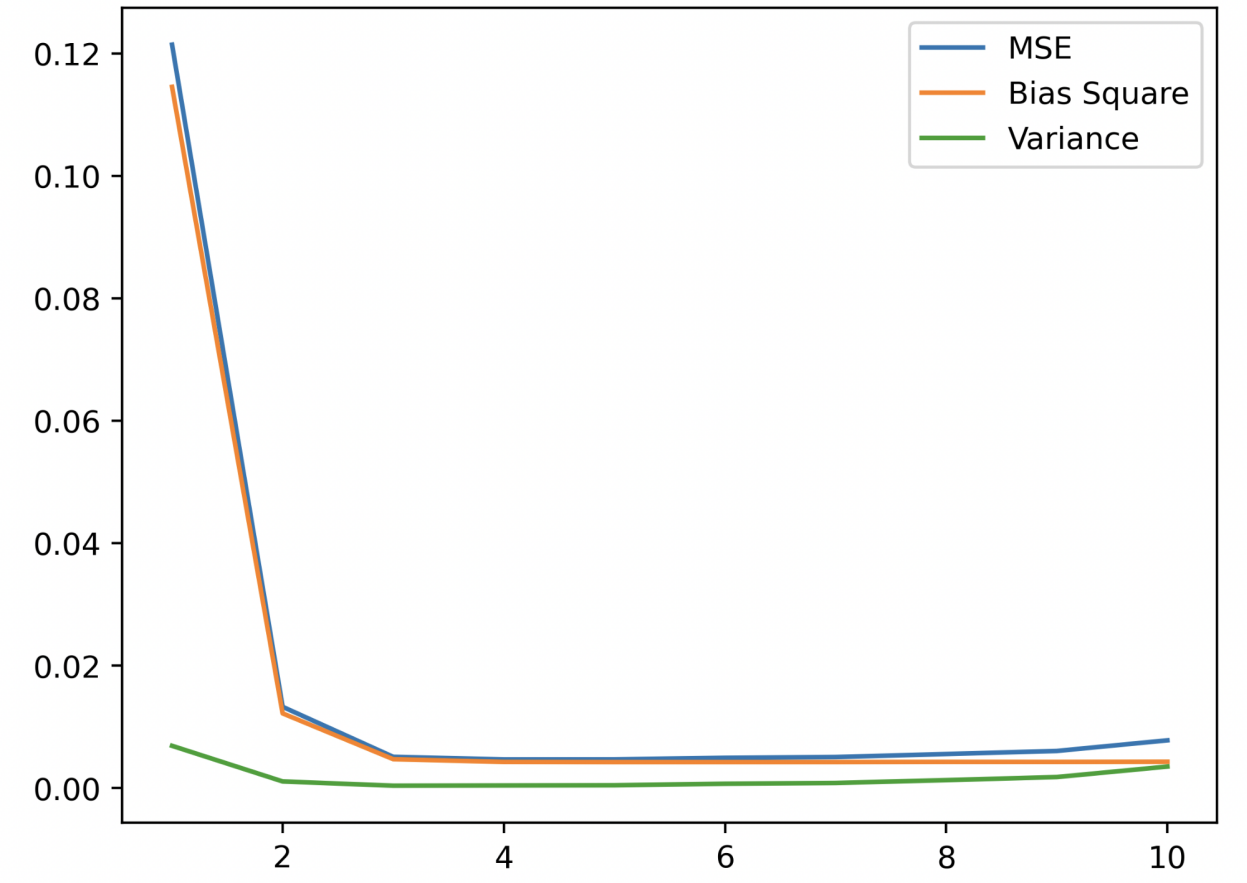
- The small changes in the irreducible error values are caused by rounding errors

- The irreducible error is equal to some value which is the same for all models

**Reasons:**

- Irreducible errors are errors which can't be removed by any model used for prediction. These errors are caused by unknown variables that are affecting the independent/output variable, i.e. the noise in the data, but are not one of the dependent/input variable while designing the model.
- Therefore, the irreducible error represents the inherent noise in the data, and hence is irrelevant and independant of the model.

## 2.3.5 Task 5 :




Due to high bias and low variance values, as shown by the study above, a fairly straightforward model, like the linear model, was unable to match the data. This is known as underfitting. Obviously, underfitting causes a lot of forecast mistakes. To

obtain the best model, we must prevent underfitting. To do this, we must ensure that the model is not very straightforward.

Similar to the above, we must ensure that the model's variance is minimal because we depend on it to forecast identical values across several realisations (with different datasets). Fundamentally, we must ensure that a model does not rely on an incorrect characteristic. Overfitting is the term used to describe a model that performs well with the train dataset but generates a lot of mistakes when applied to real-world data i.e. low bias and high variance
So, it is crucial to keep the variance low in order to keep the model as straightforward as feasible.

So based on observation:

- Bias^2 has a local minima around degree 4 and variance gradually increases, and since irreducible error is almost same, we can see that the the total error, or mean squared error, is the least around degree 4.
- Therefore, the optimal model complexity is a polynomial of degree 4.
- This would mean that for degree *1*,*2,3* the model underfits and from degree 4 onwards, the predicted values start to align themselves with the actual values and after degree 10 it overfits.
- Hence, we can say the the given data approximately forms a 4th degree curve in the 2D Plane.
- Also since irreducible error in model is coming very close to zero, the data has very less noice or randomness that the model cannot account for and so data is well structured.
- In terms of the type of data, this could apply to any type of data, whether it is numerical, categorical, text, or image data. The key factor is the quality and structure of the data, and the power and flexibility of the model used to analyze it.

## Bonus:

Value of **Capacitance**: 4.99 * 10^-5

Value of **Resistance**: 10^5