Machine, Data and Learning - Assignment 2

Chinmay Sharma
Roll No: 2022113005

February 2024

1 Task 1

1.1 Task 1(a)

From multivariable calculus, we know that the gradient of a multivariable function represents the direction of steepest ascent. Hence, the negative of the gradient represents the direction of steepest descent.

Mathematically it is represented as a matrix of the partial derivatives of the function:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

We use this property for the optimization algorithm known as gradient descent. Here the objective is to minimize a cost function that may change depending on the context. The cost function is chosen such that minimizing it brings us the closest to the objective value.

For example, for Linear regression the cost function may be the MSE (Mean Squared Error). By minimizing the MSE, we are in a sense obtaining the line that best fits the dataset. The MSE is a function of the parameters of the linear model, m and b, and can be represented as:

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

where y_i is the actual target value, (x_i) is the corresponding feature value, and (n) is the number of data points.

To minimize this cost function, we ue gradient descent. Using the property described earlier, we move in the opposite direction of the gradient to descend towards the minimum of the function.

The gradient of the MSE with respect to the parameters m and b can be calculated as follows:

$$\nabla MSE(m,b) = \begin{bmatrix} \frac{\partial MSE}{\partial m} \\ \frac{\partial MSE}{\partial b} \end{bmatrix}$$

Where

$$\frac{\partial MSE}{\partial m} = -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - (mx_i + b))$$

and

$$\frac{\partial MSE}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - (mx_i + b))$$

Once we have the gradient, we update the parameters m and b iteratively using the following update rule:

$$m := m - \alpha\frac{\partial(MSE)}{\partial m}$$

$$b := b - \alpha\frac{\partial(MSE)}{\partial b}$$

Here, α is the learning rate, which determines the size of the steps we take in the direction of the negative gradient. By repeating this process iteratively, we gradually approach the values of m and b that minimize the MSE, thereby obtaining the best-fitting line for the given dataset. We know this minima is reached if further iterations do not reduce the value of our cost function.

1.2 Task 1(b)

In the case of multiple linear regression, we have m independent variables $(x_1, x_2, \ldots, x_m)$, and the linear model can be represented as:

$$y = b + m_1x_1 + m_2x_2 + \ldots + m_mx_m$$

where y is the dependent variable (the target we want to predict), m_i is the coefficient for the i-th independent variable x_i, and b is the y-intercept.

The MSE for multiple linear regression can be expressed as:

$$MSE(m_1, m_2, \ldots, m_m, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (m_1x_{i1} + m_2x_{i2} + \ldots + m_mx_{im} + b))^2$$

where $x_{i1}, x_{i2}, \ldots, x_{im}$ are the feature values of the i-th data point.

To minimize this cost function using gradient descent, the process reamains exactly the same. We need to compute the gradient of the MSE with respect to each parameter m_i and b which can be expressed as:

$$\nabla MSE(m_1, m_2, \ldots, m_m, b) = \begin{bmatrix} \frac{\partial MSE}{\partial m_1} \\ \frac{\partial MSE}{\partial m_2} \\ \vdots \\ \frac{\partial MSE}{\partial m_m} \\ \frac{\partial MSE}{\partial b} \end{bmatrix}$$

The partial derivatives of the MSE with respect to each parameter can be calculated as follows:

$$\frac{\partial MSE}{\partial m_i} = -\frac{2}{n}\sum_{i=1}^{n} x_{ij}(y_i - (m_1 x_{i1} + m_2 x_{i2} + \ldots + m_m x_{im} + b))$$

$$\frac{\partial MSE}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - (m_1 x_{i1} + m_2 x_{i2} + \ldots + m_m x_{im} + b))$$

Once we have the gradient, we update each parameter m_i and b iteratively using the following update rules:

$$m_i := m_i - \alpha \frac{\partial(MSE)}{\partial m_i}$$

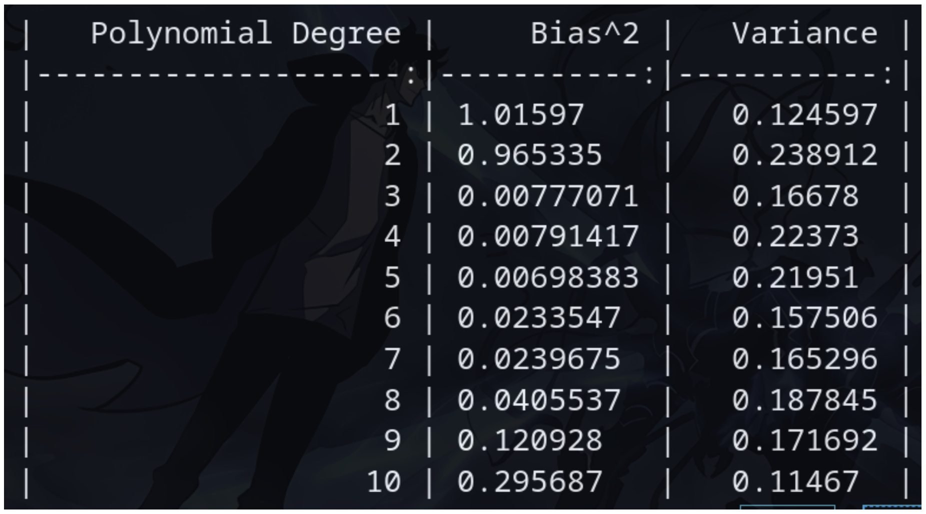
$$b := b - \alpha \frac{\partial(MSE)}{\partial b}$$

Here, α is the learning rate, which determines the size of the steps we take in the direction of the negative gradient. By repeating this process iteratively, we gradually approach the values of m_i and b that minimize the MSE, thereby obtaining the best-fitting line for the given dataset with multiple independent variables.

2 Task 2

Calculated value of Bias $^2 = 2.\overline{629}$
$Calculated value of Variance = 5.\overline{481}$
$Calculated value of MSE = 8.\overline{1}$
$We can easily verify from these values that : Bias^2 + Variance = MSE$

3 Task 3

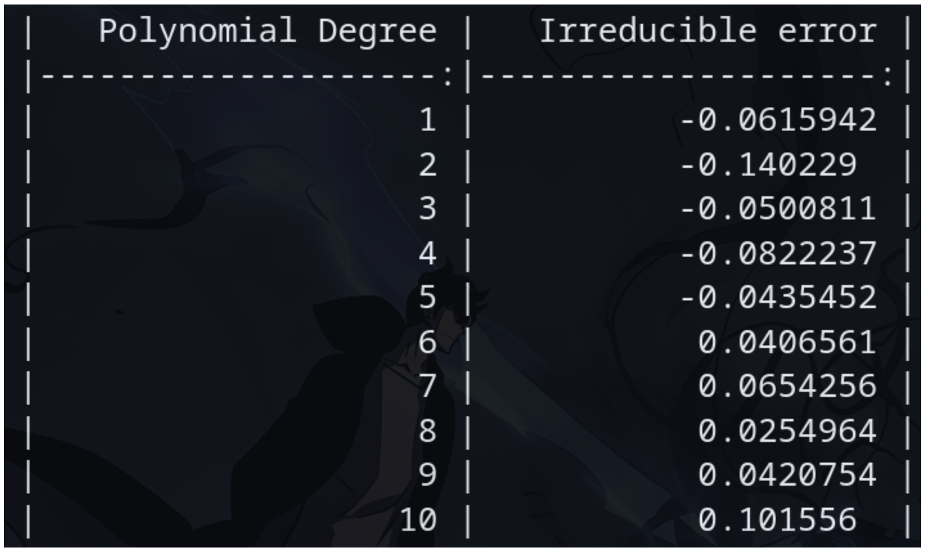


```
|   Polynomial Degree |     Bias^2 |   Variance |
|--------------------:|-----------:|-----------:|
|                   1 | 1.01597    |   0.124597 |
|                   2 | 0.965335   |   0.238912 |
|                   3 | 0.00777071 |   0.16678  |
|                   4 | 0.00791417 |   0.22373  |
|                   5 | 0.00698383 |   0.21951  |
|                   6 | 0.0233547  |   0.157506 |
|                   7 | 0.0239675  |   0.165296 |
|                   8 | 0.0405537  |   0.187845 |
|                   9 | 0.120928   |   0.171692 |
|                  10 | 0.295687   |   0.11467  |
```

We see that as we change our function classes and move towards more complex models, the magnitude of our bias, which the bias squared term represents, keeps decreasing. A model with high bias closely resembles the initial data set but it doesn't mimic the curvature etc. higher order terms of the data. As we reach a suitably complex model, the magnitude of bias drops by a relatively large amount, representing the fact that we have enough complexity to model the features of the data. However as we keep increasing the complexity, it does not decrease the magnitude of the bias and sometimes even increases it. This is the general pattern.

The variance usually rises as the complexity of the function class rises, however it is due to the nature of this data set that it increases then comes close to the original value. If we increase variance too much, we end up with an Overfit model. This performs well with the training data however, it does not perform well for testing data.

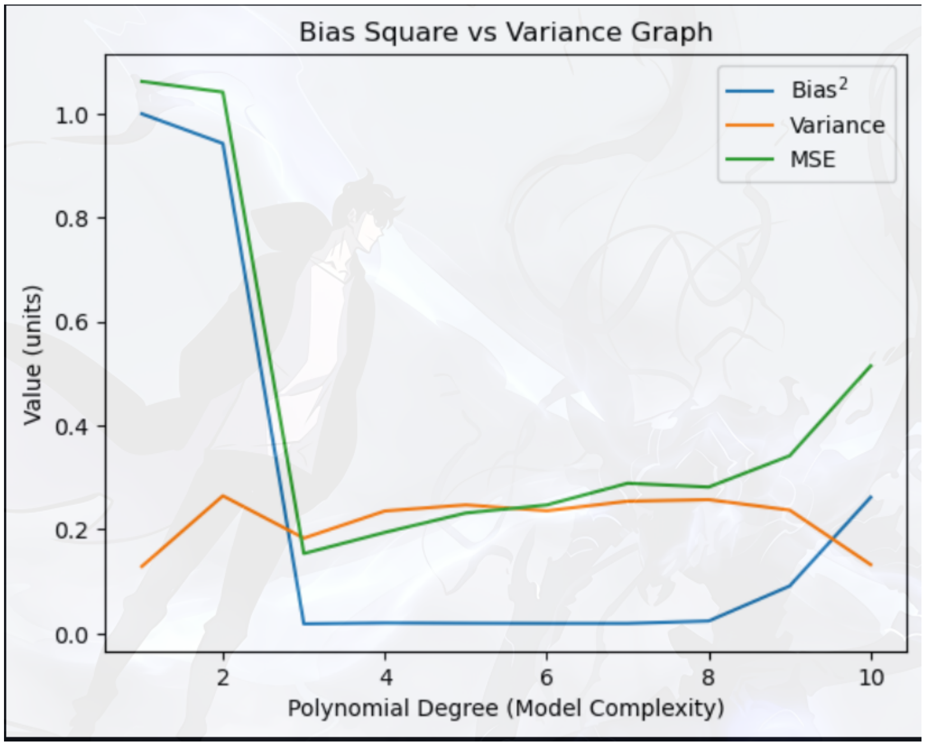
4 Task 4



```
|   Polynomial Degree |   Irreducible error |
|--------------------:|--------------------:|
|                   1 |          -0.0615942 |
|                   2 |          -0.140229  |
|                   3 |          -0.0500811 |
|                   4 |          -0.0822237 |
|                   5 |          -0.0435452 |
|                   6 |           0.0406561 |
|                   7 |           0.0654256 |
|                   8 |           0.0254964 |
|                   9 |           0.0420754 |
|                  10 |           0.101556  |
```

We see that the irreducible error remains roughly constant wrt polynomial degree. This represents the fact that the irreducible error is inherent to the data itself and hence cannot be reduced by increasing the number of parameters of our model etc.
The irreducible error is characteristic to the noise of the data and hence we cannot circumvent this by any type of model. If we use a very complex model that has low bias the variance increases. Similarly for a high bias model, the variance is low. Because of all this the irreducible error represented by the difference in the MSE and the squared bias + variance terms remains almost constant even as we vary the complexity of the model.

5 Task 5



We have high bias and low variance initially, but as model complexity keeps increasing, the bias decreases whereas the variance is seen to increase.

We go from a **simple, high bias, low variance** model, that is an **underfit** model to a **complex, low bias, high variance** model which is an overfit model.

We observe that the minima for the MSE occurs at polynomial Degree = 3.

Therefore the data complexity is best approximated by a univariate polynomial of degree 3. i.e a polynomial of the form: $ax^3 + bx^2 + cx + d$