Report

Aryan Chandramania

1 `LinearRegression().fit()`

The method `LinearRegression().fit()` is a function in Python's `scikit-learn` library that fits a linear regression model to a given set of training data.

More specifically, the `fit()` function calculates the optimal values for the coefficients of the linear regression model, which minimize the sum of squared errors between the predicted values and the actual values of the dependent variable in the training data.

Once the model has been fitted using the `fit()` function, it can then be used to make predictions on new data using the `predict()` function.

Overall, the `fit()` function is a critical step in the machine learning process and is essential for building accurate and reliable predictive models using linear regression.

2 Gradient descent with an example

Gradient descent is an iterative optimization algorithm used to find the optimal values of the coefficients in a linear regression model. The algorithm works by minimizing the cost function, which is a measure of the difference between the predicted values and the actual values of the dependent variable in the training data.

In the case where there is one independent variable and one dependent variable, the linear regression model can be represented by the equation:

$$y = mx + c$$

where y is the dependent variable, x is the independent variable, c is the intercept term, and m is the coefficient of the independent variable.

To find the optimal values of c and m using gradient descent, we start with some initial values for these coefficients and then update them iteratively until the cost function is minimized. The algorithm works as follows:

1. Initialize the values of c and m to some arbitrary values.
2. Calculate the predicted values of y using the current values of c and m.
3. Calculate the cost function, which is the mean squared error between the predicted values and the actual values of y in the training data.
4. Calculate the partial derivatives of the cost function with respect to c and m.
5. Update the values of c and m using the following formulae:

$$c = c - lr * \frac{\partial C}{\partial c}$$
$$m = m - lr * \frac{\partial C}{\partial c}$$

   where lr is the learning rate and C is the cost function.

6. Repeat steps $2 - 5$ until the cost function is minimized, or until a maximum number of iterations is reached

By iteratively updating the values of c and m using the gradient descent algorithm, we can find the optimal values of these coefficients that minimize the cost function and provide the best fit to the training data.

3 Tasks $3-5$

bias	bias^2	variance	mean squared error	irreducible error	bias(without abs)
0.26927	0.114445	0.00785544	0.1223	2.10942e-17	-0.0157819
0.0863699	0.0121412	0.00109592	0.0132371	-1.80411e-18	-0.0164617
0.03324	0.00470189	0.000379057	0.00508094	-4.44089e-18	-0.0041594
0.0245285	0.00425852	0.000461525	0.00472005	-1.94289e-18	-0.00455756
0.0238292	0.00420495	0.000545061	0.00475001	5.20417e-19	-0.00444024
0.0243632	0.00418425	0.000776679	0.00496093	-1.04083e-18	-0.00363768
0.0256472	0.00429102	0.00105058	0.0053416	-6.52256e-18	-0.00515833
0.0265547	0.0044004	0.00226527	0.00666566	1.38778e-18	-0.00670077
0.0271765	0.0045232	0.00473515	0.00925835	-2.94903e-18	-0.00498439
0.0383991	0.0093257	0.0981404	0.107466	1.85962e-17	0.0139454
0.0518602	0.0153852	0.210465	0.22585	1.4988e-17	-0.0123973
0.06429	0.0260515	0.298884	0.324936	5.55112e-18	-0.000998878
0.0702471	0.0375608	1.09599	1.13355	2.44249e-17	0.0335297
0.629263	5.00879	72.4717	77.4804	-9.66338e-15	0.107484
0.65843	5.49873	95.6927	101.191	-1.46372e-14	-0.636643

Figure 1: Tabulation of all possible statistics

3.1 Calculating bias and variance

As can be seen in the last column, the absolute bias decreases until around degree 14 or so, when it starts to increase. This is a trend that has been seen for all random shuffling of data points. The initial decrease in bias can be explained by discussing the phenomena of *underfitting* and *overfitting*. As the complexity of a model increases, its **bias** tends to decrease, as the model can better fit the training data. However, if the complexity of the model is too high, the bias may increase again due to overfitting.

Increasing the degree of the polynomial increases the complexity of the model. At low polynomial degrees, the model may have high bias, as it is not able to capture the non-linear relationships between the independent and dependent variables in the training data. However, as the degree of the polynomial increases, the model can better fit the training data, and the bias decreases.

However, as the degree of the polynomial continues to increase, the model may start to overfit the training data, leading to a high variance and increased bias. This is because the higher degree polynomial may start to fit the noise

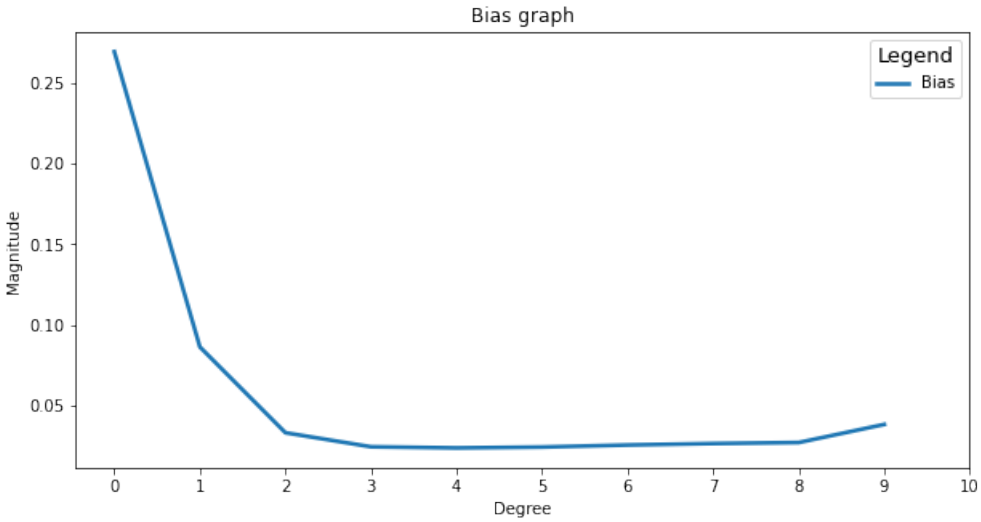

Figure 2: Plotting change in bias with increasing complexity

in the training data instead of the underlying pattern, resulting in poor performance on new data.

Therefore, the bias in a polynomial regression model can decrease and then increase as the degree of the polynomial increases. **Variance** is the amount by which the model's predictions vary for different training sets. We notice that as the complexity of the model increases, the variance of the model tends to increase as well.

As the complexity of a model increases, it becomes better at fitting the training data, and the model's predictions become more sensitive to small fluctuations in the training data. This can cause the model to overfit the training data, meaning that it captures the noise or random fluctuations in the training data, rather than the underlying pattern. As a result, the model may perform poorly on new or unseen data.

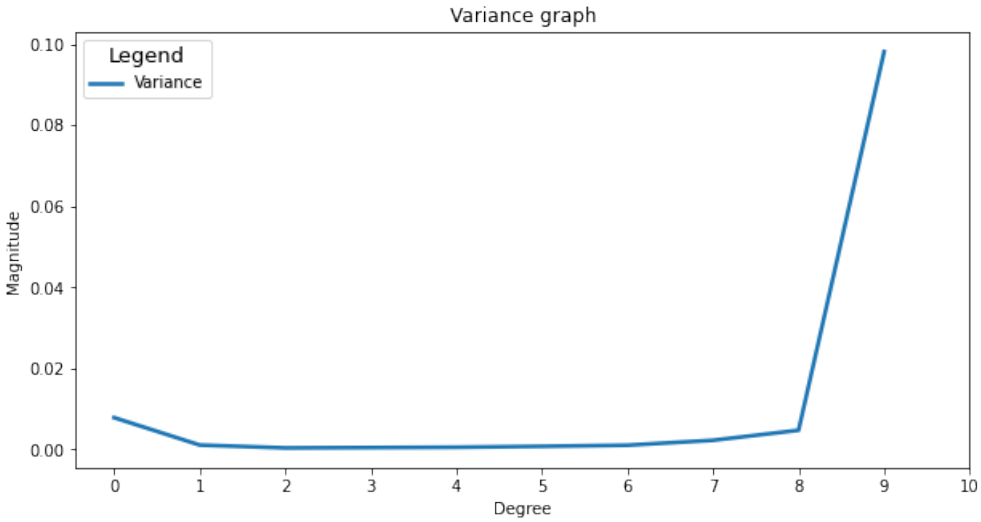

Figure 3: Plotting change in variance with increasing complexity

3.2 Calculating irreducible error

As the complexity of a machine learning model increases, the irreducible error remains constant. This is because it is not affected by changes in the model's complexity. The reason for this is that the irreducible error represents the intrinsic noise or variability in the data, which cannot be reduced by building a more complex model or by improving the quality of the training data.

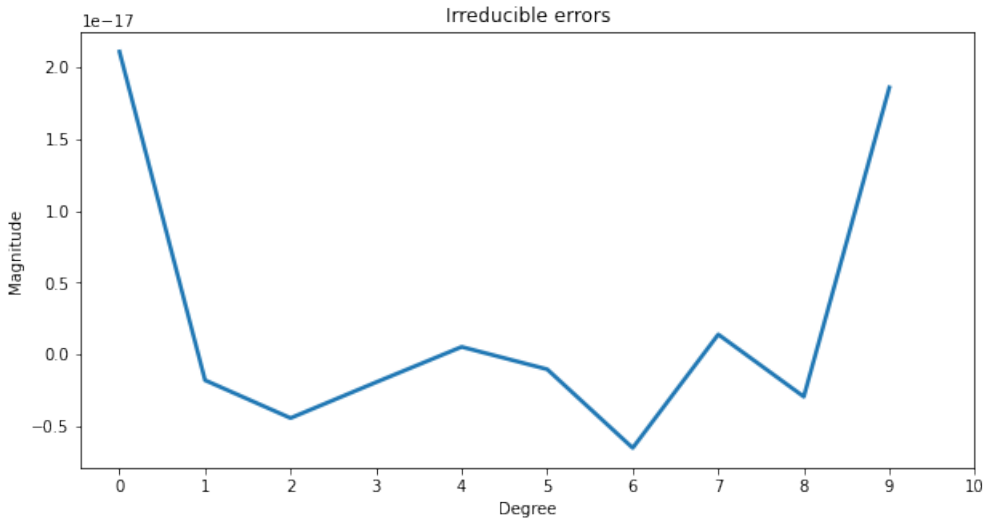

Figure 4: Plotting the (lack of) change in irreducible error. Note the extremely small order of magnitude

3.3 Plotting $Bias^2 - Variance$ graph

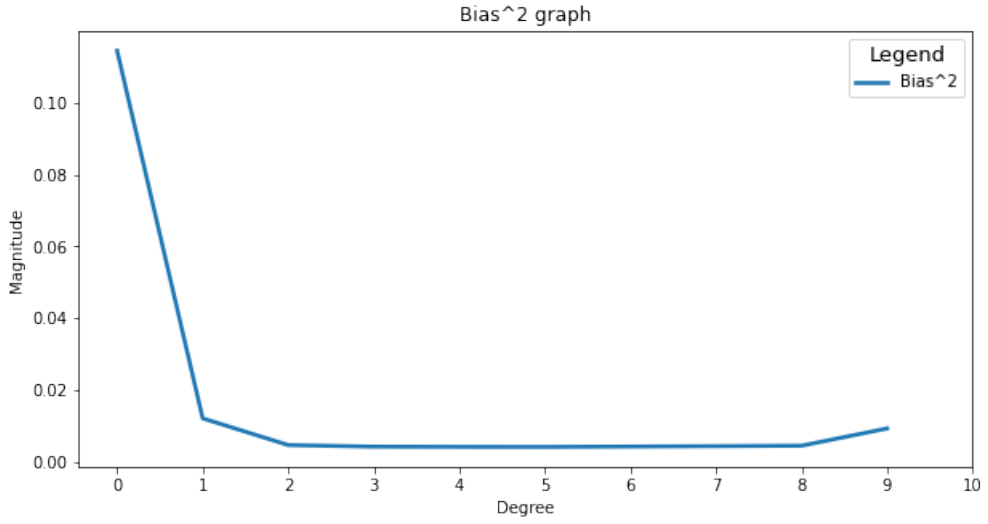

Figure 5: Plotting change in bias^2 with increasing complexity

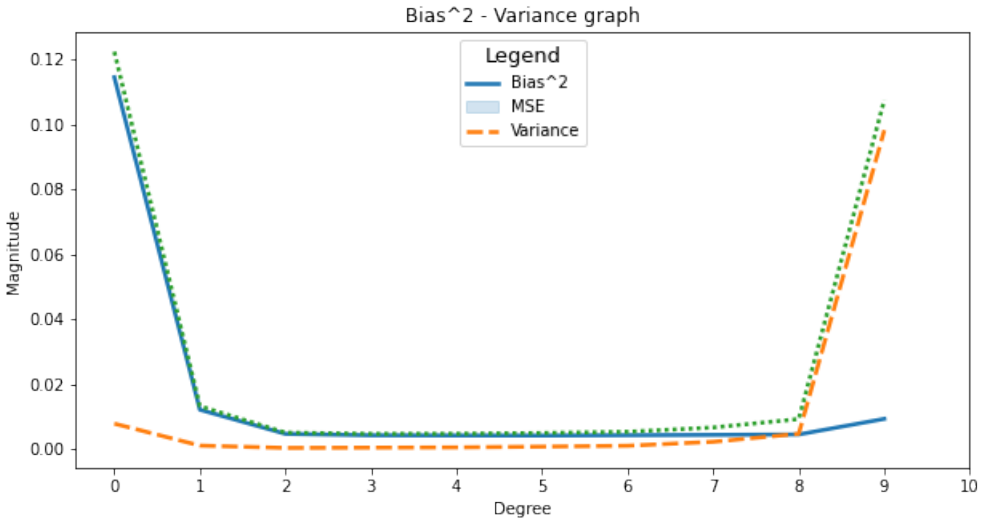

Figure 6: The bias^2-variance trade-off

Bonus

Kindly refer to the `bonus.ipynb` notebook.