Report

Task 1: Linear Regression

Write a brief about what function the method LinearRegression().fit() performs.

The *LinearRegression().fit()* method trains a linear regression model on a given dataset. In particular, it fits the linear regression model to the training data by estimating the coefficients that best fit the data according to the specified optimisation criterion (e.g., least squares).

The fit() method takes two mandatory arguments: X (a 2-dimensional array or pandas DataFrame representing the independent variables) and y (a 1-dimensional array or pandas Series representing the dependent variable). Once the model is fit, the resulting model object can be used to predict new data.

Overall, the fit() method is a fundamental step in the process of building a linear regression model and is essential for producing accurate and reliable predictions.

Task 2: Gradient Descent

Explain how gradient descent works to find the coefficients. For simplicity, take the case where there is one independent variable and one dependent variable.

Gradient descent is an optimisation algorithm used to find the values of coefficients that minimise the cost function of a linear regression model. In the case of one independent variable and one dependent variable, the linear regression model can be represented as

$$y = \beta_0 + \beta_1 x$$

where y is the dependent variable, x is the independent variable, and β_0 and β_1 are the coefficients that we want to estimate. Our goal is to find the values of β_0 and β_1 that minimise the cost function. The cost function that we will use is the mean squared error (MSE):

$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - (\beta_0 + \beta_1 x_i))^2$$

where m is the number of observations in our dataset, y_i is the actual value of the dependent variable for the ith observation, and x_i is the value of the independent variable for the ith observation.
The gradient descent algorithm works by iteratively adjusting the values of β_0 and β_1 in the direction of the steepest descent of the cost function. The steepest descent is the direction in which the cost function decreases the fastest.
To update the values of β_0 and β_1 at each iteration of the algorithm, we use the following update rules:

$$\beta_0 := \beta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\beta(x_i) - y_i)$$

$$\beta_1 := \beta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\beta(x_i) - y_i) x_i$$

where α is the learning rate, and $h_\beta(x)$ is the predicted value of y for a given value of x and a set of coefficients β_0 and β_1, which is given by:

$$h_\beta(x) = \beta_0 + \beta_1 x$$

The update rules represent the change in the coefficients required to decrease the cost function. The learning rate α controls the step size taken in the direction of the steepest descent.
The algorithm starts with some initial values of β_0 and β_1 and iteratively updates them until the cost function converges to a minimum. Some stopping criteria, such as the change in the cost function between iterations, determine the cost function's convergence.
In summary, gradient descent is an iterative optimisation algorithm that adjusts the coefficients of a model in the direction of the steepest descent of the cost function to minimise the difference between the predicted and actual values of the dependent variable. The update rules for the coefficients are based on the derivative of the cost function with respect to the coefficients, and the learning rate controls the step size taken in the direction of the steepest descent. By repeating this process iteratively, gradient descent reaches the optimal values of the coefficients that minimise the cost function.
Notably, gradient descent can be susceptible to local minima, where the algorithm converges to a suboptimal solution instead of the global minimum. Several variations of gradient descent have been developed to mitigate this issue, such as stochastic gradient descent and mini-batch gradient descent, which update the coefficients using a subset of the dataset instead of the entire dataset at each iteration. These variations can increase the algorithm's efficiency and reduce the risk of converging to a local minimum.

Task 3: Calculating Bias and Variance

Write a detailed report explaining how bias and variance change as you vary your function classes.

Relationship between Bias and Variance

- Bias and variance are inversely proportional to each other. As we decrease one, the other increases and vice versa.
- A model with high bias has low variance, and a model with high variance has low bias.
- Finding the right balance between bias and variance is crucial to building a good machine-learning model.

Function Classes

- Function classes refer to the set of functions we can use to model our data.
- A function class's complexity depends on its number of parameters.
- As we increase the complexity of our function class, we can fit more complex patterns in our data.

Effect of Function Classes on Bias and Variance

- As we increase the complexity of our function class, the bias of our model decreases.
- This is because a more complex model can fit more complex patterns in the data, which reduces the error between the model's predictions and the true values.
- However, as we increase the complexity of our function class, the variance of our model increases.
- This is because a more complex model is more sensitive to small variations in the data, which can lead to overfitting.
- Overfitting occurs when a model fits the noise in the data rather than the underlying pattern.

Bias-Variance Tradeoff

- The bias-variance tradeoff refers to the balance between bias and variance that gives the best performance for our model.
- The ideal function class is the one that minimises both bias and variance.

- In practice, finding the right balance between bias and variance is challenging and requires experimenting with different function classes and model parameters.

Conclusion

- Bias and variance are important concepts that affect the performance of a machine-learning model.
- As we increase the complexity of our function class, the bias of our model decreases, and the variance increases.
- Finding the right balance between bias and variance is crucial to building a good machine-learning model.
- The bias-variance tradeoff helps to explain the relationship between bias and variance and guides us in selecting the right function class for our model.

Task 4: Calculating Irreducible Error

Write a detailed report explaining why or why not the value of irreducible error changes as you vary your class function.

degree	irreducible_error
1	-4.194128e-17
2	-3.728029e-18
3	-1.887867e-18
4	2.969359e-18
5	1.462318e-18
6	9.550330e-18
7	3.554828e-18
8	-2.614282e-18
9	-1.351756e-17
10	1.048966e-18
11	-5.976990e-17
12	7.648287e-17
13	7.094095e-16
14	-6.594009e-17
15	6.298058e-14

Degree vs Irreducible error table

Irreducible error is an error that cannot be reduced by improving the model. It is caused by factors outside the model's control, such as measurement errors or natural variability in the data.

Effect of Function Classes on Irreducible Error

- The value of irreducible error does not change as we vary our function class.
- Irreducible error is a property of the data, not the model, and is independent of the complexity of the model.
- Regardless of the function class used, there will always be a minimum amount of error that cannot be reduced.

Importance of Irreducible Error

- Irreducible error sets an upper bound on the performance of the model.
- No matter how good the model is, it cannot perform better than this upper bound.
- Understanding the value of irreducible error is important for setting realistic expectations and evaluating the model's performance.

Conclusion

- The value of irreducible error does not change as we vary our function class.
- Irreducible error is a property of the data, not the model, and is independent of the complexity of the model.
- In our own data, the order of irreducible error varies between -18 to -17, which is a negligible variation, thereby validating that irreducible error is the property of the data, not the complexity of the function class.

Task 5: Bias^2 - Variance plot

Write your observations in the report with respect to underfitting, overfitting, and also comment on the type of data just by analysing the Bias^2 – Variance plot.

- The Bias^2 - Variance plot visualises the relationship between the bias and variance of the model for different model complexities.
- The x-axis represents the complexity of the model, and the y-axis represents the Bias^2 and Variance values.
- The Bias^2 and Variance values are usually calculated using cross-validation.

Underfitting

- When a model is underfitting, it has high bias and low variance. This means that the model is not complex enough to capture the patterns in the data.
- In the Bias^2 - Variance plot, an underfitting model has high Bias^2 and low Variance values for all model complexities.
- An underfitting model can be improved by increasing the complexity of the model or adding more features.

Overfitting

- When a model is overfitting, it has low bias and high variance.
- This means that the model is too complex and is fitting the noise in the data.
- In the Bias^2 - Variance plot, an overfitting model has low Bias^2 and high Variance values for high model complexities.
- An overfitting model can be improved by reducing the complexity of the model, adding regularisation, or increasing the amount of training data.

Balanced Model

- A balanced model has low Bias^2 and low Variance values for a moderate model complexity.
- This means that the model is complex enough to capture the patterns in the data but not so complex that it overfits the noise.
- In the Bias^2 - Variance plot, a balanced model has a U-shaped curve, where the Bias^2 and Variance values are low at the minimum point of the curve.

In conclusion,

- An underfitting model has high Bias^2 and low Variance values for all model complexities.
- An overfitting model has low Bias^2 and high Variance values for high model complexities.
- A balanced model has low Bias^2 and low Variance values for a moderate model complexity.
- In our model, when taking random samples, the balanced model comes out to be degree 3 or degree 4.