Linear Regression in Machine learning

Last Updated : 02 Sep, 2024

Machine Learning is a branch of Artificial intelligence that focuses on the development of algorithms and statistical models that can learn from and make predictions on data. **Linear regression** is also a type of machine-learning algorithm more specifically a **supervised machine-learning algorithm** that learns from the labelled datasets and maps the data points to the most optimized linear functions. which can be used for prediction on new datasets.

First of we should know what supervised machine learning algorithms is. It is a type of machine learning where the algorithm learns from labelled data. Labeled data means the dataset whose respective target value is already known. Supervised learning has two types:

- **Classification**: It predicts the class of the dataset based on the independent input variable. Class is the categorical or discrete values. like the image of an animal is a cat or dog?
- **Regression**: It predicts the continuous output variables based on the independent input variable. like the prediction of house prices based on different parameters like house age, distance from the main road, location, area, etc.

Here, we will discuss one of the simplest types of regression i.e. **Linear Regression.**

Table of Content

What is Linear Regression?

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data.

When there is only one independent feature, it is known as Simple Linear Regression, and when there are more than one feature, it is known as Multiple Linear Regression.

Similarly, when there is only one dependent variable, it is considered Univariate Linear Regression, while when there are more than one dependent variables, it is known as Multivariate Regression.

Why Linear Regression is Important?

The interpretability of linear regression is a notable strength. The model's equation provides clear coefficients that elucidate the impact of each independent variable on the dependent variable, facilitating a deeper understanding of the underlying dynamics. Its simplicity is a virtue, as linear regression is transparent, easy to implement, and serves as a foundational concept for more complex algorithms.

Linear regression is not merely a predictive tool; it forms the basis for various advanced models. Techniques like regularization and support vector machines draw inspiration from linear regression, expanding its utility. Additionally, linear regression is a cornerstone in assumption testing, enabling researchers to validate key assumptions about the data.

Types of Linear Regression

There are two main types of linear regression:

Simple Linear Regression

This is the simplest form of linear regression, and it involves only one independent variable and one dependent variable. The equation for simple linear regression is:

$y = \beta_0 + \beta_1 X$

where:

- Y is the dependent variable
- X is the independent variable
- β0 is the intercept
- β1 is the slope

Multiple Linear Regression

This involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:

$y = \beta_0 + \beta_1 X1 + \beta_2 X2 + \ldots\ldots\ldots \beta_n Xn$

where:

- Y is the dependent variable
- X1, X2, ..., Xn are the independent variables
- β0 is the intercept
- β1, β2, ..., βn are the slopes

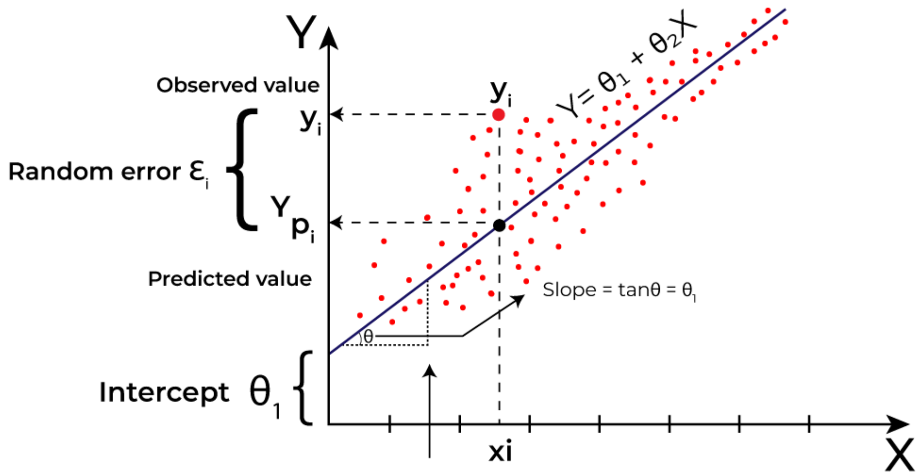
The goal of the algorithm is to find the best Fit Line equation that can predict the values based on the independent variables.

In regression set of records are present with X and Y values and these values are used to learn a function so if you want to predict Y from an unknown X this learned function can be used. In regression we have to find the value of Y, So, a function is required that predicts continuous Y in the case of regression given X as independent features.

What is the best Fit Line?

Our primary objective while using linear regression is to locate the best-fit line, which implies that the error between the predicted and actual values should be kept to a minimum. There will be the least error in the best-fit line.

The best Fit Line equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable(s).


Y
Observed value
y_i
Random error ε_i
Y_{p_i}
Predicted value
Intercept θ_1
y_i
$Y = \theta_1 + \theta_2 X$
Slope = tanθ = θ_1
θ
xi
X

Linear Regression

Here Y is called a dependent or target variable and X is called an independent variable also known as the predictor of Y. There are many types of functions or modules that can be used for regression. A linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x)). Hence, the name is Linear Regression. In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best-fit line for our model.

We utilize the cost function to compute the best values in order to get the best fit line since different values for weights or the coefficient of lines result in different regression lines.

Hypothesis function in Linear Regression

As we have assumed earlier that our independent feature is the experience i.e X and the respective salary Y is the dependent variable. Let's assume there is a linear relationship between X and Y then the salary can be predicted using:

$\hat{Y} = \theta_1 + \theta_2 X$

OR

$\hat{y}_i = \theta_1 + \theta_2 x_i$

Here,

- $y_i \epsilon Y \ \ (i = 1, 2, \cdots, n)$ are labels to data (Supervised learning)
- $x_i \epsilon X \ \ (i = 1, 2, \cdots, n)$ are the input independent training data (univariate – one input variable(parameter))
- $\hat{y}_i \epsilon \hat{Y} \ \ (i = 1, 2, \cdots, n)$ are the predicted values.

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.

How to update θ_1 and θ_2 values to get the best-fit line?

To achieve the best-fit regression line, the model aims to predict the target value $\hat{Y}$ such that the error difference between the predicted value $\hat{Y}$ and the true value Y is minimum. So, it is very important to update the θ_1 and θ_2 values, to reach the best value that minimizes the error between the predicted y value (pred) and the true y value (y).

$minimize\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$

Cost function for Linear Regression

The cost function or the loss function is nothing but the error or difference between the predicted value $\hat{Y}$ and the true value Y.

In Linear Regression, the **Mean Squared Error (MSE)** cost function is employed, which calculates the average of the squared errors between the predicted values $\hat{y}_i$ and the actual values y_i. The purpose is to determine the optimal values for the intercept θ_1 and the coefficient of the input feature θ_2 providing the best-fit line for the given data points. The linear equation expressing this relationship is $\hat{y}_i = \theta_1 + \theta_2 x_i$.

MSE function can be calculated as:

$\text{Cost function}(J) = \frac{1}{n}\sum_{n}^{i}(\hat{y}_i - y_i)^2$

Utilizing the MSE function, the iterative process of gradient descent is applied to update the values of \$\theta_1 \& \theta_2$. This ensures that the MSE value converges to the global minima, signifying the most accurate fit of the linear regression line to the dataset.

This process involves continuously adjusting the parameters \(\theta_1\) and \(\theta_2\) based on the gradients calculated from the MSE. The final result is a linear regression line that minimizes the overall squared differences between the predicted and actual values, providing an optimal representation of the underlying relationship in the data.

Gradient Descent for Linear Regression

A linear regression model can be trained using the optimization algorithm gradient descent by iteratively modifying the model's parameters to reduce the mean squared error (MSE) of the model on a training dataset. To update θ_1 and θ_2 values in order to reduce the Cost function (minimizing RMSE value) and achieve 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 update the values, reaching minimum cost.

A gradient is nothing but a derivative that defines the effects on outputs of the function with a little bit of variation in inputs.

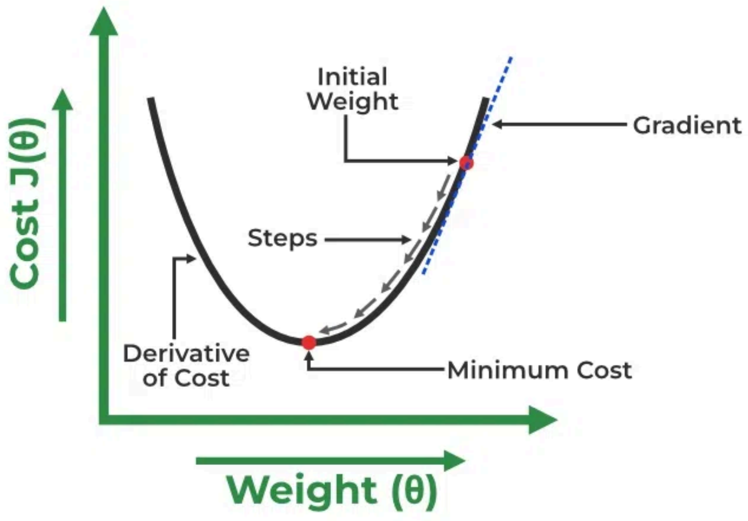
Let's differentiate the cost function(J) with respect to θ_1

$$\begin{aligned}
J'_{\theta_1} &= \frac{\partial J(\theta_1, \theta_2)}{\partial \theta_1} \\
&= \frac{\partial}{\partial \theta_1}\left[\frac{1}{n}\left(\sum_{i=1}^{n}(\hat{y}_i - y_i)^2\right)\right] \\
&= \frac{1}{n}\left[\sum_{i=1}^{n}2(\hat{y}_i - y_i)\left(\frac{\partial}{\partial \theta_1}(\hat{y}_i - y_i)\right)\right] \\
&= \frac{1}{n}\left[\sum_{i=1}^{n}2(\hat{y}_i - y_i)\left(\frac{\partial}{\partial \theta_1}(\theta_1 + \theta_2 x_i - y_i)\right)\right] \\
&= \frac{1}{n}\left[\sum_{i=1}^{n}2(\hat{y}_i - y_i)(1 + 0 - 0)\right] \\
&= \frac{1}{n}\left[\sum_{i=1}^{n}(\hat{y}_i - y_i)(2)\right] \\
&= \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)
\end{aligned}$$

Let's differentiate the cost function(J) with respect to θ_2

$$\begin{aligned} J'_{\theta_2} &= \frac{\partial J(\theta_1,\theta_2)}{\partial\theta_2} \\ &= \frac{\partial}{\partial\theta_2}\left[\frac{1}{n}\left(\sum_{i=1}^{n}(\hat{y}_i - y_i)^2\right)\right] \\ &= \frac{1}{n}\left[\sum_{i=1}^{n}2(\hat{y}_i - y_i)\left(\frac{\partial}{\partial\theta_2}(\hat{y}_i - y_i)\right)\right] \\ &= \frac{1}{n}\left[\sum_{i=1}^{n}2(\hat{y}_i - y_i)\left(\frac{\partial}{\partial\theta_2}(\theta_1 + \theta_2 x_i - y_i)\right)\right] \\ &= \frac{1}{n}\left[\sum_{i=1}^{n}2(\hat{y}_i - y_i)(0 + x_i - 0)\right] \\ &= \frac{1}{n}\left[\sum_{i=1}^{n}(\hat{y}_i - y_i)(2x_i)\right] \\ &= \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\cdot x_i \end{aligned}$$

Finding the coefficients of a linear equation that best fits the training data is the objective of linear regression. By moving in the direction of the Mean Squared Error negative gradient with respect to the coefficients, the coefficients can be changed. And the respective intercept and coefficient of X will be if α is the learning rate.



Gradient Descent

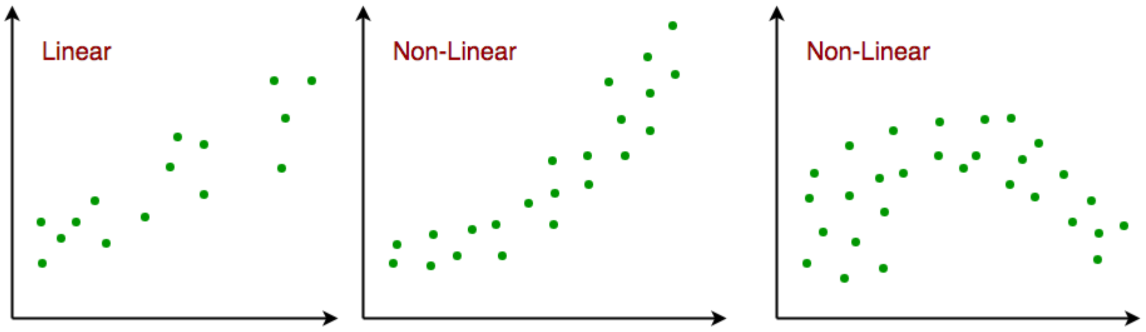
$$\begin{aligned} \theta_1 &= \theta_1 - \alpha\left(J'_{\theta_1}\right) \\ &= \theta_1 - \alpha\left(\frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\right) \end{aligned}$$

$$\theta_2 = \theta_2 - \alpha\left(J'_{\theta_2}\right)$$
$$= \theta_2 - \alpha\left(\frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\cdot x_i\right)$$

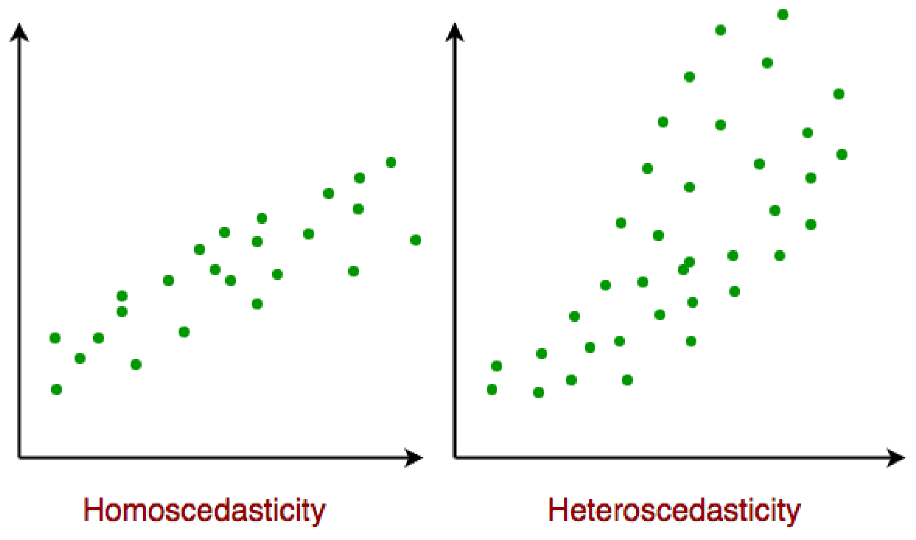
Assumptions of Simple Linear Regression

Linear regression is a powerful tool for understanding and predicting the behavior of a variable, however, it needs to meet a few conditions in order to be accurate and dependable solutions.

1. **Linearity**: The independent and dependent variables have a linear relationship with one another. This implies that changes in the dependent variable follow those in the independent variable(s) in a linear fashion. This means that there should be a straight line that can be drawn through the data points. If the relationship is not linear, then linear regression will not be an accurate model.



2. **Independence**: The observations in the dataset are independent of each other. This means that the value of the dependent variable for one observation does not depend on the value of the dependent variable for another observation. If the observations are not independent, then linear regression will not be an accurate model.
3. **Homoscedasticity**: Across all levels of the independent variable(s), the variance of the errors is constant. This indicates that the amount of the independent variable(s) has no impact on the variance of the errors. If the variance of the residuals is not constant, then linear regression will not be an accurate model.



Homoscedasticity in Linear Regression

4. **Normality**: The residuals should be normally distributed. This means that the residuals should follow a bell-shaped curve. If the residuals are not normally distributed, then linear regression will not be an accurate model.

Assumptions of Multiple Linear Regression

For Multiple Linear Regression, all four of the assumptions from Simple Linear Regression apply. In addition to this, below are few more:

1. **No multicollinearity**: There is no high correlation between the independent variables. This indicates that there is little or no correlation between the independent variables. Multicollinearity occurs when two or more independent variables are highly correlated with each other, which can make it difficult to determine the individual effect of each variable on the dependent variable. If there is multicollinearity, then multiple linear regression will not be an accurate model.
2. **Additivity:** The model assumes that the effect of changes in a predictor variable on the response variable is consistent regardless of the values of the other variables. This assumption implies that there is no interaction between variables in their effects on the dependent variable.
3. **Feature Selection:** In multiple linear regression, it is essential to carefully select the independent variables that will be included in the

model. Including irrelevant or redundant variables may lead to overfitting and complicate the interpretation of the model.

4. **Overfitting:** Overfitting occurs when the model fits the training data too closely, capturing noise or random fluctuations that do not represent the true underlying relationship between variables. This can lead to poor generalization performance on new, unseen data.

Multicollinearity

Multicollinearity is a statistical phenomenon that occurs when two or more independent variables in a multiple regression model are highly correlated, making it difficult to assess the individual effects of each variable on the dependent variable.

Detecting Multicollinearity includes two techniques:

- **Correlation Matrix:** Examining the correlation matrix among the independent variables is a common way to detect multicollinearity. High correlations (close to 1 or -1) indicate potential multicollinearity.
- **VIF (Variance Inflation Factor):** VIF is a measure that quantifies how much the variance of an estimated regression coefficient increases if your predictors are correlated. A high VIF (typically above 10) suggests multicollinearity.

Evaluation Metrics for Linear Regression

A variety of evaluation measures can be used to determine the strength of any linear regression model. These assessment metrics often give an indication of how well the model is producing the observed outputs.

The most common measurements are:

Mean Square Error (MSE)

Mean Squared Error (MSE) is an evaluation metric that calculates the average of the squared differences between the actual and predicted values for all the data points. The difference is squared to ensure that negative and positive differences don't cancel each other out.

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

Here,

- n is the number of data points.
- y_i is the actual or observed value for the ith data point.
- y_i is the predicted value for the ith data point.

MSE is a way to quantify the accuracy of a model's predictions. MSE is sensitive to outliers as large errors contribute significantly to the overall score.

Mean Absolute Error (MAE)

Mean Absolute Error is an evaluation metric used to calculate the accuracy of a regression model. MAE measures the average absolute difference between the predicted values and actual values.

Mathematically, MAE is expressed as:

$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - Y_i|$

Here,

- n is the number of observations
- Y_i represents the actual values.
- Y_i represents the predicted values

Lower MAE value indicates better model performance. It is not sensitive to the outliers as we consider absolute differences.

Root Mean Squared Error (RMSE)

The square root of the residuals' variance is the Root Mean Squared Error. It describes how well the observed data points match the expected values, or the model's absolute fit to the data.

In mathematical notation, it can be expressed as:

$$RMSE = \frac{RSS}{n} = \frac{\sum_{i=2}^{n}(y_i^{actual} - y_i^{predicted})^2}{n}$$

Rather than dividing the entire number of data points in the model by the number of degrees of freedom, one must divide the sum of the squared residuals to obtain an unbiased estimate. Then, this figure is referred to as the Residual Standard Error (RSE).

In mathematical notation, it can be expressed as:

$$RMSE = \frac{RSS}{n} = \frac{\sum_{i=2}^{n}(y_i^{actual} - y_i^{predicted})^2}{(n-2)}$$

RSME is not as good of a metric as R-squared. Root Mean Squared Error can fluctuate when the units of the variables vary since its value is dependent on the variables' units (it is not a normalized measure).

Coefficient of Determination **(R-squared)**

R-Squared is a statistic that indicates how much variation the developed model can explain or capture. It is always in the range of 0 to 1. In general, the better the model matches the data, the greater the R-squared number. In mathematical notation, it can be expressed as:

$$R^2 = 1 - \left(\frac{RSS}{TSS}\right)$$

- **Residual sum of Squares (RSS): The** sum of squares of the residual for each data point in the plot or data is known as the residual sum of squares, or RSS. It is a measurement of the difference between the output that was observed and what was anticipated.
  $$RSS = \sum_{i=2}^{n}(y_i - b_0 - b_1x_i)^2$$
- **Total Sum of Squares (TSS):** The sum of the data points' errors from the answer variable's mean is known as the total sum of squares, or TSS.
  $$TSS = \sum(y - y_i)^2$$

R squared metric is a measure of the proportion of variance in the dependent variable that is explained the independent variables in the model.

Adjusted R-Squared Error

Adjusted R^2 measures the proportion of variance in the dependent variable that is explained by independent variables in a regression model. Adjusted R-square accounts the number of predictors in the model and penalizes the model for including irrelevant predictors that don't contribute significantly to explain the variance in the dependent variables.

Mathematically, adjusted R^2 is expressed as:

$$Adjusted\,R^2 = 1 – (\frac{(1-R^2).(n-1)}{n-k-1})$$

Here,

- n is the number of observations
- k is the number of predictors in the model
- R^2 is coeeficient of determination

Adjusted R-square helps to prevent overfitting. It penalizes the model with additional predictors that do not contribute significantly to explain the variance in the dependent variable.

Python Implementation of Linear Regression

Import the necessary libraries:

Python

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.axes as ax
from matplotlib.animation import FuncAnimation
```

Load the dataset and separate input and Target variables

Here is the link for dataset: Dataset Link

Python

```
url = 'https://media.geeksforgeeks.org/wp-
content/uploads/20240320114716/data_for_lr.csv'
data = pd.read_csv(url)
data

# Drop the missing values
data = data.dropna()

# training dataset and labels
train_input = np.array(data.x[0:500]).reshape(500, 1)
train_output = np.array(data.y[0:500]).reshape(500, 1)

# valid dataset and labels
test_input = np.array(data.x[500:700]).reshape(199, 1)
test_output = np.array(data.y[500:700]).reshape(199, 1)
```

Build the Linear Regression Model and Plot the regression line

Steps:

- In forward propagation, Linear regression function Y=mx+c is applied by initially assigning random value of parameter (m & c).
- The we have written the function to finding the cost function i.e the mean

Python

```
class LinearRegression:
    def __init__(self):
        self.parameters = {}

    def forward_propagation(self, train_input):
        m = self.parameters['m']
        c = self.parameters['c']
        predictions = np.multiply(m, train_input) + c
        return predictions

    def cost_function(self, predictions, train_output):
        cost = np.mean((train_output - predictions) ** 2)
        return cost

    def backward_propagation(self, train_input, train_output,
predictions):
        derivatives = {}
        df = (predictions-train_output)
        # dm= 2/n * mean of (predictions-actual) * input
        dm = 2 * np.mean(np.multiply(train_input, df))
        # dc = 2/n * mean of (predictions-actual)
        dc = 2 * np.mean(df)
        derivatives['dm'] = dm
        derivatives['dc'] = dc
        return derivatives

    def update_parameters(self, derivatives, learning_rate):
        self.parameters['m'] = self.parameters['m'] - learning_rate *
derivatives['dm']
        self.parameters['c'] = self.parameters['c'] - learning_rate *
derivatives['dc']

    def train(self, train_input, train_output, learning_rate, iters):
        # Initialize random parameters
        self.parameters['m'] = np.random.uniform(0, 1) * -1
        self.parameters['c'] = np.random.uniform(0, 1) * -1

        # Initialize loss
        self.loss = []

        # Initialize figure and axis for animation
        fig, ax = plt.subplots()
        x_vals = np.linspace(min(train_input), max(train_input), 100)
        line, = ax.plot(x_vals, self.parameters['m'] * x_vals +
                        self.parameters['c'], color='red',
label='Regression Line')
        ax.scatter(train_input, train_output, marker='o',
                color='green', label='Training Data')

        # Set y-axis limits to exclude negative values
        ax.set_ylim(0, max(train_output) + 1)

        def update(frame):
            # Forward propagation
            predictions = self.forward_propagation(train_input)

            # Cost function
            cost = self.cost_function(predictions, train_output)

            # Back propagation
            derivatives = self.backward_propagation(
                train_input, train_output, predictions)

            # Update parameters
            self.update_parameters(derivatives, learning_rate)

            # Update the regression line
            line.set_ydata(self.parameters['m']
                        * x_vals + self.parameters['c'])

            # Append loss and print
            self.loss.append(cost)
            print("Iteration = {}, Loss = {}".format(frame + 1,
cost))

            return line,
        # Create animation
        ani = FuncAnimation(fig, update, frames=iters, interval=200,
blit=True)

        # Save the animation as a video file (e.g., MP4)
        ani.save('linear_regression_A.gif', writer='ffmpeg')

        plt.xlabel('Input')
        plt.ylabel('Output')
        plt.title('Linear Regression')
        plt.legend()
        plt.show()

        return self.parameters, self.loss
```

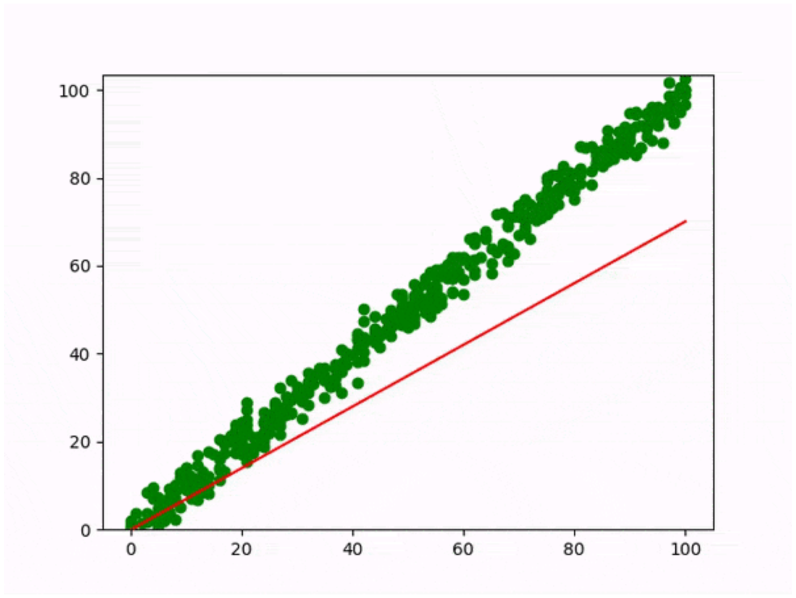
Trained the model and Final Prediction

Python

```
#Example usage
linear_reg = LinearRegression()
parameters, loss = linear_reg.train(train_input, train_output,
0.0001, 20)
```

Output:

```
Iteration = 1, Loss = 9130.407560462196
Iteration = 1, Loss = 1107.1996742908998
Iteration = 1, Loss = 140.31580932842422
Iteration = 1, Loss = 23.795780526084116
Iteration = 2, Loss = 9.753848205147605
Iteration = 3, Loss = 8.061641745006835
Iteration = 4, Loss = 7.8577116490914864
Iteration = 5, Loss = 7.8331350515579015
Iteration = 6, Loss = 7.830172502503967
Iteration = 7, Loss = 7.829814681591015
Iteration = 8, Loss = 7.829770758846183
Iteration = 9, Loss = 7.829764664327399
Iteration = 10, Loss = 7.829763128602258
Iteration = 11, Loss = 7.829762142342088
Iteration = 12, Loss = 7.829761222379141
Iteration = 13, Loss = 7.829760310486438
Iteration = 14, Loss = 7.829759399646989
Iteration = 15, Loss = 7.829758489015161
Iteration = 16, Loss = 7.829757578489033
Iteration = 17, Loss = 7.829756668056319
Iteration = 18, Loss = 7.829755757715535
Iteration = 19, Loss = 7.829754847466484
Iteration = 20, Loss = 7.829753937309139
```



Linear Regression Line

The linear regression line provides valuable insights into the relationship between the two variables. It represents the best-fitting line that captures the overall trend of how a dependent variable (Y) changes in response to variations in an independent variable (X).

- **Positive Linear Regression Line**: A positive linear regression line indicates a direct relationship between the independent variable (X) and the dependent variable (Y). This means that as the value of X increases, the value of Y also increases. The slope of a positive linear regression line is positive, meaning that the line slants upward from left to right.
- **Negative Linear Regression Line**: A negative linear regression line indicates an inverse relationship between the independent variable (X) and the dependent variable (Y). This means that as the value of X increases, the value of Y decreases. The slope of a negative linear

regression line is negative, meaning that the line slants downward from left to right.

Regularization Techniques for Linear Models

Lasso Regression (L1 Regularization)

Lasso Regression is a technique used for regularizing a linear regression model, it adds a penalty term to the linear regression objective function to prevent overfitting.

The objective function after applying lasso regression is:

$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - y_i) + \lambda \sum_{j=1}^{n} |\theta_j|$

- the first term is the least squares loss, representing the squared difference between predicted and actual values.
- the second term is the L1 regularization term, it penalizes the sum of absolute values of the regression coefficient θ_j.

Ridge Regression (L2 Regularization)

Ridge regression is a linear regression technique that adds a regularization term to the standard linear objective. Again, the goal is to prevent overfitting by penalizing large coefficient in linear regression equation. It useful when the dataset has multicollinearity where predictor variables are highly correlated.

The objective function after applying ridge regression is:

$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - y_i) + \lambda \sum_{j=1}^{n} \theta_j^2$

- the first term is the least squares loss, representing the squared difference between predicted and actual values.
- the second term is the L1 regularization term, it penalizes the sum of square of values of the regression coefficient θ_j.

Elastic Net Regression

Elastic Net Regression is a hybrid regularization technique that combines the power of both L1 and L2 regularization in linear regression objective.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - y_i) + \alpha\lambda \sum_{j=1}^{n} |\theta_j| + \frac{1}{2}(1-\alpha)\lambda \sum_{j=1} n\theta_j^2$$

- the first term is least square loss.
- the second term is L1 regularization and third is ridge regression.
- λ is the overall regularization strength.
- α controls the mix between L1 and L2 regularization.

Applications of Linear Regression

Linear regression is used in many different fields, including finance, economics, and psychology, to understand and predict the behavior of a particular variable. For example, in finance, linear regression might be used to understand the relationship between a company's stock price and its earnings or to predict the future value of a currency based on its past performance.

Advantages & Disadvantages of Linear Regression

Advantages of Linear Regression

- Linear regression is a relatively simple algorithm, making it easy to understand and implement. The coefficients of the linear regression model can be interpreted as the change in the dependent variable for a one-unit change in the independent variable, providing insights into the relationships between variables.
- Linear regression is computationally efficient and can handle large datasets effectively. It can be trained quickly on large datasets, making it suitable for real-time applications.
- Linear regression is relatively robust to outliers compared to other machine learning algorithms. Outliers may have a smaller impact on the overall model performance.
- Linear regression often serves as a good baseline model for comparison with more complex machine learning algorithms.

- Linear regression is a well-established algorithm with a rich history and is widely available in various machine learning libraries and software packages.

Disadvantages of Linear Regression

- Linear regression assumes a linear relationship between the dependent and independent variables. If the relationship is not linear, the model may not perform well.
- Linear regression is sensitive to multicollinearity, which occurs when there is a high correlation between independent variables. Multicollinearity can inflate the variance of the coefficients and lead to unstable model predictions.
- Linear regression assumes that the features are already in a suitable form for the model. Feature engineering may be required to transform features into a format that can be effectively used by the model.
- Linear regression is susceptible to both overfitting and underfitting. Overfitting occurs when the model learns the training data too well and fails to generalize to unseen data. Underfitting occurs when the model is too simple to capture the underlying relationships in the data.
- Linear regression provides limited explanatory power for complex relationships between variables. More advanced machine learning techniques may be necessary for deeper insights.

Conclusion

Linear regression is a fundamental machine learning algorithm that has been widely used for many years due to its simplicity, interpretability, and efficiency. It is a valuable tool for understanding relationships between variables and making predictions in a variety of applications.

However, it is important to be aware of its limitations, such as its assumption of linearity and sensitivity to multicollinearity. When these limitations are carefully considered, linear regression can be a powerful tool for data analysis and prediction.