Linear Regression Notes

Linear Regression

Below, types of Linear Regression.

1. Simple Linear Regression
2. Multiple Linear Regression
3. Polynomial Linear Regression
4. Regularisation Linear Regression

* Regression Metrics
* Gradient Descent
* Bias Variance Trade-Off

1. Simple Linear Regression

Simple linear regression is a statistical method that models the linear relationship between two variables using a straight line equation.

Intuition

$y = mx + b$ (m is slop and b is intercept)

Now, for finding a value of m and b we have two methods.

1. Closed form (Ordinary List Squared)

For info in scikitlearn LinearRegression() class.

2. Non closed form (Gradient Descent)

For info in scikitlearn SGDRegressor() class.

1. Closed form (Ordinary List Squared)

Formula $y = m\,x + b$

Now, we will see how we can get m and b.

$b = \bar{y} - m\,\bar{x}$ ($\bar{x}$ is mean of x-axis values, $\bar{y}$ is mean of y-axis values and m is below)

$$m = \frac{\sum_{i=1}^{n} (x\,i - \bar{x})(y\,i - \bar{y})}{\sum_{i=1}^{n} (x\,i - \bar{x})^2}$$

2. Non closed form (Gradient Descent)

** It has been discussed in Gradient Descent discussion which is below the Simple and Multiple Linear Regression's discussion.*

Regression Metrics

They provide a quantitative assessment of how well the regression model is performing in terms of predicting the dependent variable.

1. *Mean Squared Error (MSE)*
2. *Root Mean Squared Error (RMSE)*
3. *Mean Absolute Error (MAE)*
4. *R-squared (Coefficient of Determination)*
5. *Adjusted R-squared*

1. Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is a metric used to measure the average absolute difference between the observed values and the values predicted by a model.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|yi - \hat{y}i\right|$$

Here, y is actual data point, $\hat{y}$ is predicted.

Advantages

Interpretability and Robustness to Outliers.

Disadvantages

Sensitivity to Outliers and Equal weight to all errors.

2. Mean Squared Error (MSE)

The Mean Absolute Error (MAE) is a common metric used to measure the average squared difference between predicted and actual values in regression analysis.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(yi - \hat{y}i\right)^2$$

Here, y is actual data point, $\hat{y}$ is predicted.

Advantages

Sensitivity Error and Mathematically Simplicity.

Disadvantages

Sensitive to outliers and Units of measurement.

3. Root Mean Squared Error (RMSE)

It is derived from the Mean Squared Error (MSE) but provides a measure of error.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(yi - \hat{y}i\right)^2}$$

Here, y is actual data point, $\hat{y}$ is predicted.

4. R-squared (Coefficient of Determination)

The R-squared score, also known as the coefficient of determination, is a metric used to evaluate the goodness of fit of a regression model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(yi - \hat{y}i \right)^2}{\sum_{i=1}^{n} \left(yi - \bar{y}\, i \right)^2}$$

Here, y *is actual data point,* $\hat{y}$ *is predicted.* $\bar{y}$ *is the mean of the observed values.*

5. Adjusted R-squared

The Adjusted R-squared (Adjusted R-squared) is a modified version of the R-squared score that adjusts for the number of predictors in a regression model. It penalises the inclusion of irrelevant predictors that do not improve the model's explanatory power.

$$R^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - k - 1}$$

Here, n is numbers of data points, k is the number of independent variables in data point and R^2 is for coefficient for determination.

Advantages

Penalty for Additional Predictors and Range of values.

2. Multiple Linear Regression

Multiple linear regression is an extension of simple linear regression, allowing for the modelling of relationships between a dependent variable and multiple independent variables.

Equation

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n + \varepsilon$$

Formula

$$y = Xb + \varepsilon$$

Here, y is the dependent variable, X is the matrix of independent variables, b is the vector of coefficients (including the intercept), ε is the error term.

Let's see, how we can find ε and b by formulas.

$$\varepsilon = \left(y - \hat{y}\right)$$

Here y is actual or observed values and $\hat{y}$ is predicted values of the target variable. Even $\hat{y} = Xb$ where X is the matrix of independent variables, b is the vector of coefficients (including the intercept).

$$b = \left(X^{\mathrm{T}} X\right)^{-1} X^{\mathrm{T}} y$$

Here X is input matrix, b is vector of coefficients and intercept and y is observed target value. It a differentiation of error function result.

Gradient Descent

Gradient descent is an iterative optimisation algorithm used for finding the minimum of a function. The primary goal of gradient descent is to minimise a cost function or loss function. This function measures the difference between the model's predictions and the actual target values.

How it works ?

Step 1: Start with initial values for the model parameters.

Step 2: In each iteration, the algorithm updates the parameters in the direction opposite to the gradient of the cost function.

Step 3: The process continues until the algorithm converges to a minimum or a predetermined number of iterations is reached.

What is Gradient ?

The gradient represents the direction of the steepest ascent in the cost function.

How we will reach to minimum ? For that Learning Rate.

The learning rate determines the size of the steps taken during each iteration. Choose learning rate according to condition for less iteration.

When Iterations will stop ?

Here, we have two methods to do this thing.

1. *Epochs*

2. *Until O is reached.*

Below, the types of Gradient Descent.

1. *Batch: Updates are made in each iteration.*

2. *Stochastic: Updates are made choosing any data point at a time.*

3. *Mini-Batch: Updates are made using a small batch of data points.*

1. Batch Gradient Discent

Updates are made in each iteration. For that we have to find b *and* m *every iteration. One thing that iteration is based on epoch or until we reach at* 0*. So we will find* b *and* m *like this. This is useful when we have convex function or small dataset.*

For Simple Linear Regression.

$$b_{\text{new}} = b_{\text{old}} - \left(\alpha \frac{\partial J}{\partial b}\right)$$

Here b_{new} *is next iteration,* b_{old} *is last iteration and* α *is Learning Rate and* $\frac{\partial J}{\partial b}$ *is the partial derivative of the cost function with respect to the intercept.* $\frac{\partial J}{\partial b}$ *would be* $-2\sum_{i=1}^{n}(y - \hat{y}i)$ *here* $\hat{y}i$ *would be* $mX + b$

$$m_{\text{new}} = m_{\text{old}} - \left(\alpha \frac{\partial J}{\partial m}\right)$$

Here m_{new} *is next iteration,* m_{old} *is last iteration and* α *is Learning Rate and* $\frac{\partial J}{\partial m}$ *is the partial derivative of the cost function with respect to the intercept.* $\frac{\partial J}{\partial m}$ *would be* $-2\sum_{i=1}^{n}(y - \hat{y}i) \times X$ *here* $\hat{y}i$ *would be* $mX + b$

y *is the dependent variable which the output we want to predict.*

x *is the independent variable which is the input feature.*

m *is the slope of the line.*

b *is the intercept.*

For Multiple Linear Regression.

$$b_{\text{new}} = b_{\text{old}} - \left(\alpha \frac{\partial J}{\partial b}\right)$$

Here b_{new} is next iteration, b_{old} is last iteration and α is Learning Rate and $\frac{\partial J}{\partial b}$ is the partial derivative of the cost function with respect to the intercept. $\frac{\partial J}{\partial b}$ would be

$$-2\sum_{i=1}^{m}\left(yi - \hat{y}i\right)$$

Here, $\hat{y}i = Xi \cdot \beta + \beta 0$ where Xi represents the feature vector for the i-th data point, β is the vector of coefficients, and $\beta 0$ is the intercept.

$$m_{\text{new}} = m_{\text{old}} - \left(\alpha \frac{\partial J}{\partial m}\right)$$

Here m_{new} is next iteration, m_{old} is last iteration and α is Learning Rate and $\frac{\partial J}{\partial m}$ is the partial derivative of the cost function with respect to the intercept. $\frac{\partial J}{\partial m}$ would be

$$\frac{-2}{m}\sum_{i=1}^{m}\left(yi - \hat{y}i\right) \times X$$

Here, $\hat{y}i = Xi \cdot \beta + \beta 0$ where Xi represents the feature vector for the i-th data point, β is the vector of coefficients, and $\beta 0$ is the intercept.

2. Stochastic Gradient Discent

In SGD, instead of using the entire training dataset to compute the gradient of the cost function, only one randomly chosen training example is used at a time.

The parameters (weights and biases) are updated after each individual data point.

SGD introduces more randomness into the optimisation process, which can sometimes help escape local minima but may also introduce more noise.

Formula

It is same as Batch Gradient Descent but the small changes is we would take one random row of independent variable X instead of taking whole independent variable X

3. Mini-Batch Gradient Discent

Mini-Batch GD strikes a balance between the efficiency of batch GD and the stochastic nature of SGD.

It updates the parameters using a small random subset or "mini-batch" of the training data.

The mini-batch size is a hyper-parameter that is typically chosen based on the available computational resources.

Formula

As you know it a combination of Batch and Stochastic so we will use both formula in combine. We will find the random value times the value of the epochs.

3. Polynomial Linear Regression

Instead of fitting a straight line (as in linear regression), polynomial regression uses a polynomial equation to capture the non-linear patterns in the data.

Equation

$$y = b0 + b1 \cdot x + b2 \cdot x^2 + \cdots + \varepsilon$$

y is the dependent variable (output). x is the independent variable (input). $b0, b1, b2, \ldots, bn$ are the coefficients of the polynomial terms. n is the degree of the polynomial. ϵ represents the error term.

The choice of the degree n determines the flexibility of the model. Higher degrees allow the model to capture more complex patterns but also increase the risk of overfitting, especially when the dataset is small.

In scikitlearn PolynomialFeatures() class. We use training for fit_transform() method and for test transform() method.

Bias Variance Trade-Off

It refers to the balance between bias and variance in the predictive performance of a model.

Bias

Bias refers to the error introduced by problem, which may be extremely complex, by a much simpler model. High bias can lead the model to under-fit the data, meaning it fails to capture the underlying patterns in the training data.

Variance

Variance is the model's sensitivity to small fluctuations or noise in the training data. A model with high variance is very flexible and can fit the training data very closely, but it might fail to generalise well to new, unseen data.

According to it we should keep low bias and low variance for batter result. Few techniques such as cross validation, and proper model selection and finding a optimal trade-off which can aim for right strike.

4. Regularisation Linear Regression

It helps prevent overfitting by discouraging overly complex models, usually by penalising large coefficients.

Below, the types of Regularisation Techniques.

1. *Ridge Regularisation (L2)*
2. *Lasso Regularisation (L1)*
3. *Elastic Net*

1. Ridge Regularisation (L2)

Ridge Regression is a regularisation technique used in linear regression to prevent overfitting and improve the stability of the model.

Formula

$$\frac{1}{n}\sum_{i=1}^{n}\left(yi-\hat{y}i\right)^2+\lambda\,||\,w\,||^2$$

Ridge Regularisation with Simple Linear Regression.

$$y=mx+b$$

Now, we will see how we can get m and b.

$b=\bar{y}-m\,\bar{x}$ ($\bar{x}$ is mean of x-axis values, $\bar{y}$ is mean of y-axis values and m is below)

$$m=\frac{\sum_{i=1}^{n}(xi-\bar{x})(yi-\bar{y})}{\sum_{i=1}^{n}(xi-\bar{x})^2+\lambda}$$

λ is essential to strike a balance between fitting the data well and preventing overfitting by adjusting it.

Ridge Regularisation with Multiple Linear Regression.

Formula

$$w=\left(X^{\mathrm{T}}X+\alpha I\right)^{-1}X^{\mathrm{T}}y$$

Here X is input matrix, w is the weight vector (including the intercept) and y is observed target value. I is the identity matrix. α is the regularisation parameter theta. you will obtain the values for β which include the

intercept $\beta 0$ and coefficients for each independent variable.

Ridge Regularisation with Batch Gradient Descent.

Formula

$$w = w - \lambda \cdot \left(X^{\mathrm{T}} X \cdot w - X^{\mathrm{T}} y + \alpha w\right)$$

Here X is input matrix, w is the weight vector (including the intercept) and y is observed target value. λ is the learning rate. αw is the alpha (regularisation parameter) multiply with theta w. After applying this formula we can abstract b and m from it answer.

Also we can use regularisation parameter (Ridge) with stochastic and mini-batch gradient descent.

2. Lasso Regularisation (L1)

Lasso regression, or L1 regularisation, is a linear regression technique that incorporates a regularisation term in the objective function to prevent overfitting and promote feature selection.

The regularisation term is the absolute value of the coefficients multiplied by a regularisation parameter (α).

The addition of the regularisation term encourages the model to shrink some of the coefficients to exactly zero, effectively performing feature selection. This can be beneficial in situations where only a subset of features is truly relevant, leading to a simpler and more interpretable model.

Formula

$$\frac{1}{n}\sum_{i=1}^{n}\left(yi-\hat{y}i\right)^2+\lambda||w||$$

3. Elastic Net

Elastic Net regression is a regularisation technique that combines both L1 (Lasso) and L2 (Ridge) regularisation penalties.

It is useful when dealing with datasets that have highly correlated features, as it helps to select a group of correlated features rather than selecting only one from the group (as Lasso might do). When any input data has more multicollinearity that time elastic net is useful.

Formula

$$\frac{1}{n}\sum_{i=1}^{n}\left(yi-\hat{y}i\right)^{2}+a\,||w||^{2}+\mathrm{b}\,||w||$$