Linear Regression

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

Linear Regression Model Representation

The model for a simple regression problem (a single x and a single y) would be:

$$y = B0 + B1*x$$

When we have more than one input (x) in higher dimensions, the line is called a plane or a hyper-plane. The form of the equation and the precise numbers utilised for the coefficients are hence the representation (e.g. B0 and B1 in the above example).

It is usual to discuss the complexity of a regression model, such as linear regression. The number of coefficients employed in the model is denoted by this.

When a coefficient reaches zero, it effectively removes the influence of the input variable on the model and, as a result, the model's prediction (0 * x = 0). This is important when considering regularisation strategies, which alter the learning algorithm to minimise the complexity of regression models by exerting pressure on the absolute magnitude of the coefficients, driving some to zero.

Now that we've covered the representation used for a linear regression model, let's look at several methods for learning this representation from data.

Linear Regression: Learning the Model

Learning a linear regression model entails estimating the values of the coefficients used in the representation using the given data.
In this section, we'll take a quick look at four methods for creating a linear regression model. This isn't enough knowledge to build them from scratch, but it'll give you a sense of the computation and trade-offs required:

1. Simple Linear Regression

When we have a single input for basic linear regression, we can utilise statistics to estimate the coefficients.

This necessitates calculating statistical features such as means, standard deviations, correlations, and covariance from the data. To traverse and calculate statistics, all of the data must be available.

2. Ordinary Least Squares

When there are multiple inputs, we can utilise Ordinary Least Squares to estimate the coefficient values.

The OLS approach aims to reduce the sum of squared residuals. This means that, given a regression line across the data, we calculate the distance between each data point and the regression line, square it, and add all of the squared errors together. Ordinary least squares aims to reduce this amount.

This method analyses the data as a matrix and employs linear algebra operations to estimate the optimal coefficient values. It implies that all of the data must be available, as well as sufficient memory to fit the data and perform matrix operations.

It is unusual to implement the Ordinary Least Squares algorithm yourself, unless as a linear algebra homework. You are more likely to use a procedure from a linear algebra library. This process is extremely fast to compute.

3. Gradient Descent

When there are one or more inputs, you can employ a procedure of optimising the coefficient values by iteratively minimising the model's error on the training data.

Gradient Descent is the name given to this operation, which begins with random values for each coefficient. For each pair of input and output values, the sum of squared errors is computed. As a scale factor, a learning rate is used, and the coefficients are updated in the direction of error minimization. The technique is repeated until either a minimum sum squared error is obtained or no further improvement is achievable.

When employing this method, you must choose a learning rate (alpha) parameter that sets the magnitude of the improvement step to take on each procedure repetition.

Gradient descent is frequently taught using a linear regression model since it is simple to understand. In practise, it is handy when you have a large dataset, either in terms of the number of rows or columns, that may not fit into memory.

Making Predictions with Linear Regression

Making predictions is as simple as solving the equation for a certain set of inputs, given that the representation is a linear equation.

Let me illustrate this with an example. Assume we're predicting weight (y) based on height (x). For this situation, our linear regression model would be:

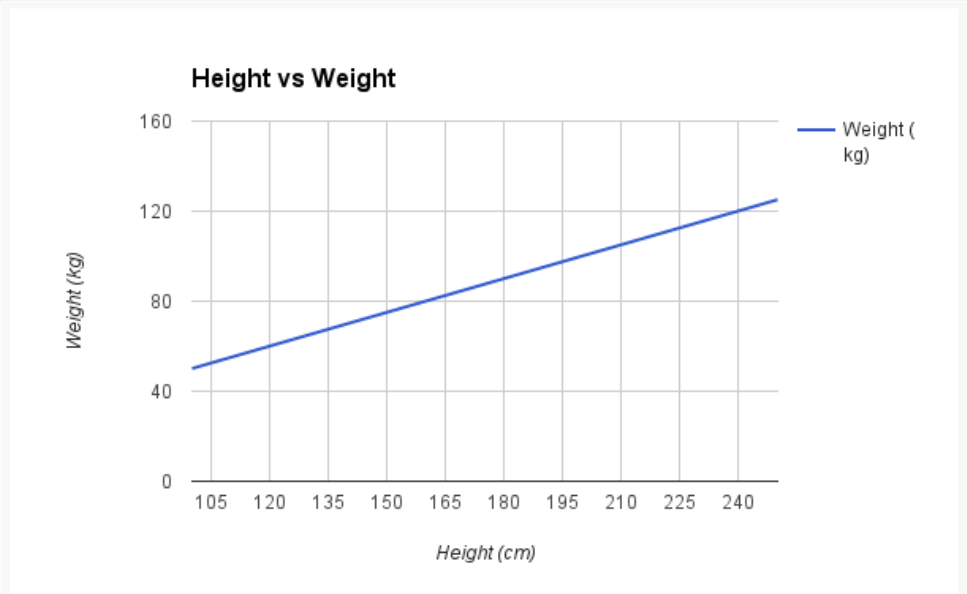
y = B0 + B1 * x1.
or
weight =B0 +B1 * height

Where B0 is the bias coefficient and B1 is the height column coefficient. To obtain a decent set of coefficient values, we employ a learning technique. Once discovered, we can anticipate the weight by plugging in different height values.

As an example, consider B0 = 0.1 and B1 = 0.5. Let's enter them in and get the weight (in kilogrammes) of a person who is 182 centimetres tall.

weight = 0.1 + 0.5 * 182
weight = 91.1

As you can see, the above equation can be displayed as a two-dimensional line. Regardless of our height, the B0 is our starting point. We can go through a series of heights ranging from 100 to 250 cm, plug them into the equation, and get weight values to create our line.



Sample Height vs Weight Linear Regression

Difference between R-Square and Adjusted R Square?

Short Answer: In case of adjusted R2 we include only the important and useful variables, whereas in R2 all the variables are included.

Mathematically: Mathematically, R-squared is calculated by dividing the sum of squares of residuals (**SSres**) by the total sum of squares ($\mathbf{SS_{tot}}$) and then subtracting it from 1. In this case, $\mathbf{SS_{tot}}$ measures total variation. $\mathbf{SS_{res}}$ measures explained variation and $\mathbf{SS_{reg}}$ measures unexplained variation.

As,

$\mathbf{SS_{res} + SS_{reg} = SS_{tot}}$**, $\mathbf{R^2}$ = Explained variation / Total Variation**

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \Rightarrow 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

$$R^2 = \frac{SS_{\text{reg}}}{SS_{\text{tot}}}$$

R-Squared is also called coefficient of determination. It lies between 0% and 100%. A r-squared value of 100% means the model explains all the variation of the target variable. And a value of 0% measures zero predictive power of the model. Higher R-squared value, better the model.

Adjusted R-Squared

It measures the proportion of variation explained by only those independent variables that really help in explaining the dependent variable. It penalizes you for adding independent variables that do not help in predicting the dependent variable.

Adjusted R-Squared can be calculated mathematically in terms of sum of squares. The only difference between the R-squared and Adjusted R-square equation is degree of freedom.

$$\bar{R}^2 = 1 - \frac{SS_{\text{res}}/\text{df}_e}{SS_{\text{tot}}/\text{df}_t}$$

Adjusted R-Squared Equation

In the above equation, df_t is the degrees of freedom n– 1 of the estimate of the population variance of the dependent variable, and df_e is the degrees of freedom n – p – 1 of the estimate of the underlying population error variance.

Adjusted R-squared value can be calculated based on value of r-squared, number of independent variables (predictors), total sample size.

$$R^2_{adjusted} = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

where

R^2 = sample R-square

p = Number of predictors

N = Total sample size.

Adjusted R-Squared Equation 2

1. Every time you add an independent variable to a model, the R-squared increases, even if the independent variable is insignificant. It never declines. Whereas Adjusted R-squared increases only when independent variable is significant and affects dependent variable.

   In the table below, adjusted r-squared is maximum when we include two variables. It declines when a third variable is added. Whereas r-squared increases when we include the third variable. It means the third variable is insignificant to the model.

Variables	R-Squared	Adjusted R- Squared
1	67.5	67.1
2	85.9	84.2
3	88.9	81.7

R-Squared vs. Adjusted R-Squared

2. 2.Adjusted r-squared can be negative when r-squared is close to zero.
3. 3.Adjusted r-squared value always be less than or equal to r-squared value.

Which is Better?

Adjusted R-square should be used to compare models with different numbers of independent variables. Adjusted R-square should be used while selecting important predictors (independent variables) for the regression model.

Code:

```
import numpy as np
y = np.array([21, 21, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2])
yhat = np.array([21.5, 21.14, 26.1, 20.2, 17.5, 19.7, 14.9, 22.5, 25.1, 18])
R2 = 1 - np.sum((yhat - y)**2) / np.sum((y - np.mean(y))**2)
R2
n=y.shape[0]
p=3
adj_rsquared = 1 - (1 - R2) * ((n - 1)/(n-p-1))
adj_rsquared
```

What metrics can we use for regression problems?

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

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

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

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

Some Additional Things to keep in mind:

Assumptions of Linear Regression?

Short Trick: Assumptions can be abbreviated as LINE in order to remember.

L : Linearity (Relationship between x and y is linear)
I : Independence (Observations are independent of each other)
N : Normality (for any fix value of x, y is normally distributed)
E : Equal Variance (homoscedasticity)

What is Multicollinearity? How to Avoid?

Multicollinearity occurs when your model includes multiple factors that are correlated not just to your response variable, but also to each other. In other words, it results when you have factors that are a bit redundant.

You can think about it in terms of a cricket game: If one player catches the ball, it's easy to give credit. But if three players are going for the catch simultaneously, it's much more difficult to determine which of the three makes the biggest contribution to the catch.
What does it mean: Multicollinearity increases the standard errors of the coefficients. Increased standard errors in turn means that coefficients for some independent variables may be found not to be significantly different from 0. In other words, by overinflating the standard errors, multicollinearity makes some variables statistically insignificant when they should be significant. Without multicollinearity (and thus, with lower standard errors), those coefficients might be significant.

If multicollinearity is a problem in your model -- if the VIF for a factor is near or above 5 -- the solution may be relatively simple. Try one of these:

- Remove highly correlated predictors from the model. If you have two or more factors with a high VIF, remove one from the model. Because they supply redundant information, removing one of the correlated factors usually doesn't drastically reduce the R-squared. Consider using stepwise regression, or specialized knowledge of the data set to remove these variables. Select the model that has the highest R-squared value. Also check for Adj. R-Squared.

- Use Principal Component Analysis that cut the number of predictors to a smaller set of uncorrelated components.

Code:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):

    # Calculating VIF
    vif = pd.DataFrame()
```

```
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

    return(vif)

…

X = df.iloc[:,:-1]
calc_vif(X)
```