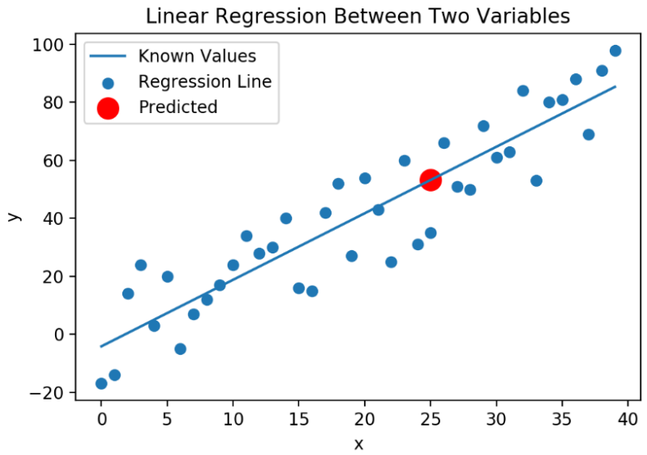
Linear & Polynomial Regression

Introduction

Linear and polynomial regression are **supervised learning** techniques, which use correlation between variables to establish a mathematical relationship between them.

This relationship can be used to estimate values for new data:



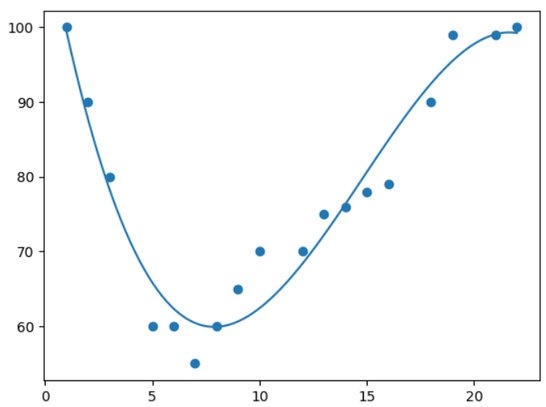
Based on known values, we can find the optimal $y = mx + c$ relationship between the two. This is done by finding the m and c values which correspond to the minimum **cost** (sum of the differences between the y actual values in the training data and our estimate for the same x). We can then estimate values of y where it is unknown for a given x.

Polynomial Regression

The relationship between variables may not be linear, in which cases adding powers or applying functions to your variables before performing regression may be appropriate. An example polynomial equation would be:

$$y = \theta_2 x^2 + \theta_1 x + \theta_0$$

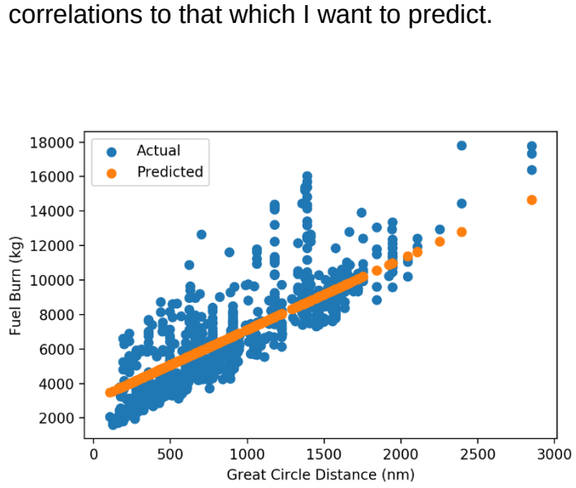
Where θ_2, θ_1 and θ_0 are tweaked by regression to fit the data.

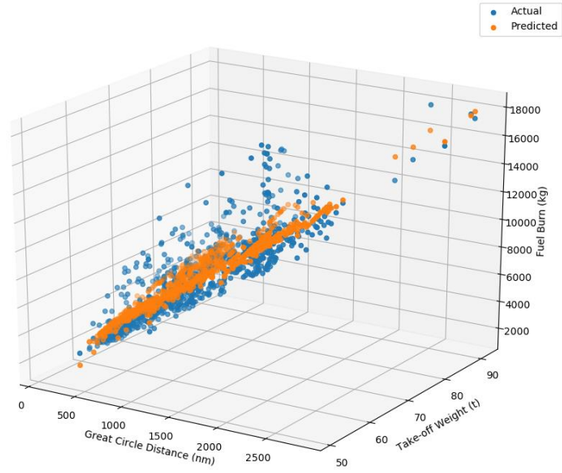


Regression in Higher Dimensions

Regression can be performed using an almost infinite number of input variables. As long as there is some relationship between it and what you're trying to predict, the model will get more accurate when you add another variable.

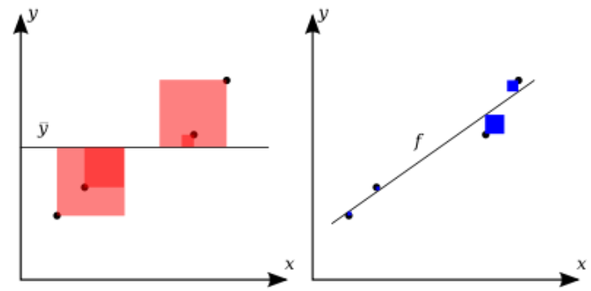
Say for example I wanted to predict the fuel burn for an A320 from London Gatwick to Marrakech and I had data from other routes. I could use the route lengths as my input data and guess based on that, but I could also use take-off weight as another dimension for more accurate results. When I add this new dimension of take-off weight, the predicted values were only half as far from their real values in the test data. This is also a good example of **selecting relevant variables**, the model will be more accurate if I select parameters with strong correlations to that which I want to predict.





Evaluating the Model

The quality of the model can be evaluated by the R-squared coefficient, a value between 0 and 1. Using the difference between the predicted and real values the model produces, and the same metric if there was no correlation in any of the variables, the R-squared value represents what percentage of the variation is explained by the features used.



The square's area represents the squared difference between the predicted and real values, for no correlation (left) and the model (right) [1]

[1] - https://en.wikipedia.org/wiki/Coefficient_of_determination

Implementing Regression

Source code can be found at P:\ENGINEERING\FPO\Digitalisation\1 FPO Projects\IIX Data Analytics\02-General presentations\Knowledge folder - preparation\Data Science Learning Framework\Machine Learning Posters\2 - Linear Regression

Introduction

We're going to use the above mentioned example, predicting fuel burn using take-off weight and mission range to give an example of how to implement regression. Using training data, we'll tune θ_0, θ_1, θ_2 such that the equation

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

y – Fuel Burn
x_1- Mission Range
x_2- Take-Off Weight

is representative. This can then be used to predict fuel burn for untested routes!

Formatting your data

If you believe your variables have a non-linear relationship, this is the time to perform the desired function on your feature to get the format you want. We'll just use linear regression in this example, but if I believed fuel burn was proportional to take-off weight squared, I'd add a new column with take-off weight squared, and remove the old column. From here I can continue analysis as usual.

A column of ones should be added to calibrate θ_0, then, to get data into the correct format to train the model, variables must be separated into features (the values you're using) and labels (the value you're trying to predict). The features will be separated into table X, and the labels into table y.

constant	total_mission_great_circle_nm	take_off_weight
1	662.158601	73.173203
1	356.277530	76.674927
1	1066.726470	67.857124
1	842.082199	66.460068
1	309.327138	61.434284

Table X

fuel_burn_total2
5751.545898
3864.602539
6549.867188
6041.843994
3465.442139

Table y

To allow faster convergence of the model, all feature columns except the constant should be scaled such that they're of the same magnitude. The equation

$$\mathrm{X}_{i,j} = \frac{\mathrm{X}_{i,j} - \overline{\mathrm{X}_j}}{\max \mathrm{X}_j - \min \mathrm{X}_j}$$

will return all values scaled to $-0.5 < x < 0.5$.

Finally, data should be separated (randomly to avoid bias) into training and testing data, essentially taking a number of rows that you won't show the model while it's training, such that it can be tested against them without having "seen the answers". The % split of training and testing data is arbitrary, but 70/30 is a good ballpark figure.

Cost and Gradient Descent

Cost (J) is a measure of the accuracy of the model. It assesses the mean squared difference between the predicted values in the training data and the real values, by taking the sum of the squared differences, and dividing it by the number of training examples. Minimizing this value is key to machine learning, it assesses the success of the model.

$$J(\theta_0, \theta_1, \theta_2) = \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} - y^{(i)})^2$$

m – Number of rows in training data
$y^{(i)}$ – Label for the i^{th} row
$x_j^{(i)}$- Value for the j^{th} feature of the i^{th} row

They key to minimizing J is iteratively modifying $\theta_0, \theta_1, \theta_2$ by means of gradient descent. The equation for gradient descent is

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \theta_2)$$

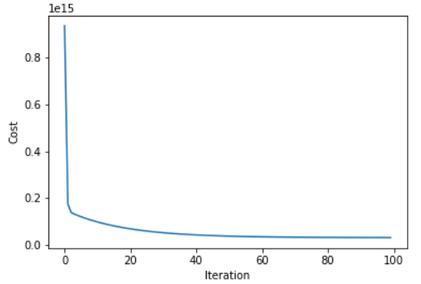
or

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} - y^{(i)}) x_j^{(i)}$$

New value of θ_j — Old value of θ_j

α – Learning rate
θ_j – θ corresponding to the j^{th} feature
$x_j^{(i)}$- Value for the j^{th} feature of the i^{th} row

This equation is performed iteratively, updating the values of $\theta_0, \theta_1, \theta_2$ before using these updated values to repeat itself. The derivative term is subtracted from the old θ values, reducing the cost by bringing θ closer to the desired values. The learning rate is a coefficient which dictates how much the model will change its θ values. If it's too small excessive iterations will be needed, too large and the model will diverge as it hops over the correct value with each iteration.



Results

Having used gradient descent on our scaled data with an α of 0.8 (found through trial and error) and 100 iterations, θ values have converged:

$$\theta_0 = 6609, \theta_1 = 11388, \theta_2 = 6062$$

So we can estimate fuel burn for new routes, given the mission length and take-off weight using (as long as we've scaled our data first!):

$$y = 6609 + 11388 x_1 + 6062 x_2$$

The model can be applied to our test data to for insight into its accuracy.