Algorithms for Data Mining – Assessment Item 1, Task 1

Polynomial Regression

# Section 1: Description of Polynomial Regression

## Error function used for regression

An error/cost function in regression is essential as it displays the difference between the ground truth and the predicted values, the most commonly used function used is the (Root) Mean Squared Error. The function calculates the error/variance for each point in the dataset from the best fitting line, squares it and then calculates the mean from all those points. The reason for squaring the error value is to ensure that the value is positive (as the variance can be negative), and will highlight outliers. With MSE and RMSE, the lower the value the better, and if you were to have a perfect best fitting line, the value of the MSE/RMSE would be 0. Rooting the MSE enables the values to be more easily read, as it will give you the distance on average from the best fitting line across all the values in the dataset.

The mean squared error cost function can be expressed mathematically, with the RMSE being a simple root of the formula below:

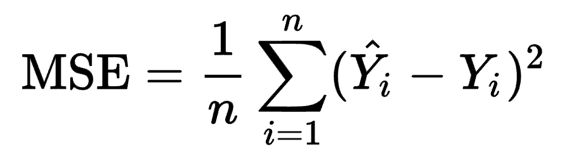


Figure 1 - Mean Squared Error Cost Function

The formula subtracts the predicted individual Y values from the ground truth Y values, before squaring them. From there, the mean is calculated from those values. That would therefore produce the MSE, and to produce the RMSE you could simply square root the MSE.

An easier to read Python solution is below:

Figure 2 - Python solution for Root Mean Squared Error

error = y\_pred – y

squaredError = error \*\* 2

meanSquaredError = squaredError.mean()

rootMeanSquaredError = np.sqrt(meanSquaredError)

Which can be simplified to:

Figure 3 - Simplified Python solution for RMSE

rmse = np.sqrt(((y\_pred - y) \*\* 2).mean())

## Linear Regression Models

## Least Squares Solution

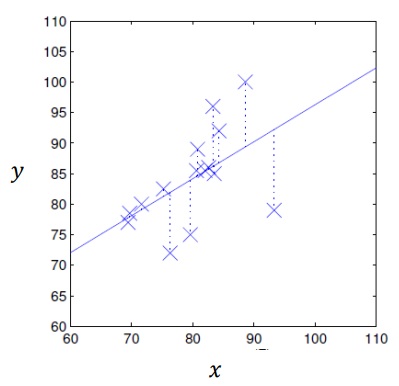
The concept for the ordinary least squares method is to calculate and reduce the sum of the squared errors produced by each point at how far it is away from the line. The error in this case is also sometimes referred to as the ‘residual’. Looking at *Figure 4*, showing the ordinary least squares method in action on a linear regression line, the points (marked with X’s) that are close to the line will have a very low error value, whereas the outliers on the graph (the points far below and above the line) will have a large error. Calculating the lowest sum of the errors produced by the points will result in the ‘best fitting line’.

Figure 4 - Error cost in ordinary least squares

## Polynomial feature expansion

Polynomial feature expansion and its use in polynomial regression is the main step between simple linear regression using least squares and polynomial regression. Often also referred to as a Vandermonde Matrix, The process of expanding the features of values to *n* degree allows for multiple polynomials to create the best fitting line for the degree chosen.

The process to expand the features is as follows:

Taking a degree *n*, and an array of values *X* you need to loop through from 0 to *n* raising variable X to the power of *n* each time. The resulting shape of the array will be *j x n*, this can be expressed visually as:

In practice, the first column of polynomial feature expansion will always be 1, as being multiplied by zero is 1.

## How to use polynomial features in linear regression

## Difference between training set and test set performance

# Section 2: Implementation of Polynomial Regression

# Section 3: Evaluation