Polynomial regression intro

Thursday, September 26, 2019 9:50 AM

Polynomial regression is nothing but performing linear regression with polynomial Features. We will talk about polynomial features but before that lets look at what is a polynomial.

What is a polynomial ?

It is a collection of terms. Unlike the equation of a straight line, it contains many terms. The general form of a polynomial is written as-

$$y = w_0 + w_1 x + w_2 x^2 + \ldots + w_N x^N$$

where N is the order of the polynomial

Some simple examples

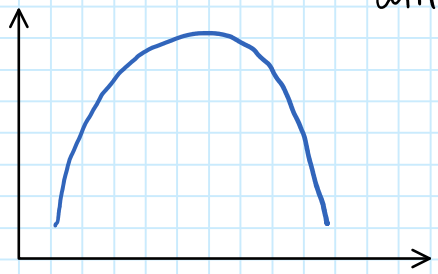
Polynomial of order 1

$y = w_0 + w_1 x$ ← this is our good old straight line

Polynomial of order 2

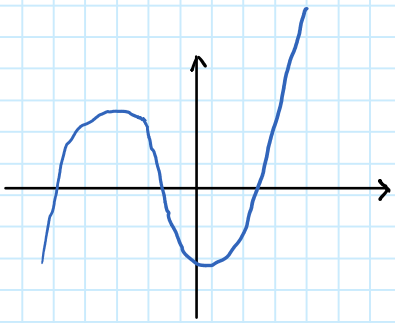
$y = w_0 + w_1 x + w_2 x^2$ ← this is a parabola

will look something like this



Polynomial of order 3

$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$ ← this is called a cubic function



Now that we have a general intro to polynomials let us look at how polynomial regression works.

We can treat polynomial regression just like multi feature linear regression When you have - say 3 features we do linear regression to get -

$$\text{target} = w_0 + w_1(\text{feature 1}) + w_2(\text{feature 2}) + w_3(\text{feature 3})$$

fitting a model with three features gives us the values for the parameters - (w_0, w_1, w_2, w_3). We can do the same thing with the other terms in polynomial regression. For example -

Suppose we want to do 3rd order polynomial regression with a feature.

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

$$\text{target} = w_0 + w_1(\text{feature}) + w_2(\text{feature})^2 + w_3(\text{feature})^3$$

The function PolynomialFeatures() in sklearn does exactly this! You can change the number of polynomial features in the function.

$$\text{PolynomialFeatures}(2) \rightarrow (1, \vec{x}, \vec{x}^2)$$
$$\text{PolynomialFeatures}(3) \rightarrow (1, \vec{x}, \vec{x}^2, \vec{x}^3)$$

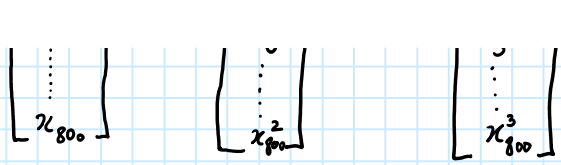
I have put a little $\rightarrow$ on top of x because x is an array of values -

$$\text{feature} \rightarrow \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}$$

In the linear regression example we did, we had 800 points in the training set then N here would be 800.

For example
with N = 800 our 3rd order polynomial would look like

$$\text{target} = w_0 + w_1 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{800} \end{bmatrix} + w_2 \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ \vdots \\ x_{800}^2 \end{bmatrix} + w_3 \begin{bmatrix} x_1^3 \\ x_2^3 \\ x_3^3 \\ \vdots \\ x_{800}^3 \end{bmatrix}$$


x_{800}
x^2_{800}
x^3_{800}

Instructions for Saturday

Thursday, September 26, 2019 10:31 AM

For Saturday-

I want you all to try out the following thing -

1) Train the linear regression model on the training set as we did in class but run the predict function of the test set.

   This means that you will have to import Linear_Regression_test.csv as well. Then reshape it the same way we did with the training data.

   Once you do that run lr.predict on it. It should look something like

   lr.predict(x_test_reshaped)

   Then calculate the mean_squared_error and the r2_score. The code for that is in the notebook I am sending you

2) The second task that I have for you all is to work with the shampoo sales dataset-

- First task is to fit a straight line to It. Just like you did with the Linear regression
- Second is to run polynomial regression on it with. Run it for polynomial features from 1 to 10.

  The way you would do this is to write a for loop to loop over each polynomial feature.

  Your code will have something like-

```
For i in in range(0,10):
    Ploy_features = PolynomialFeatures(i)
    Poly_x = Poly_features.fit_transform(x_train_shampoo)

    Lr = LinearRegression()
    Lr.fit(poly_x, y_shampoo)
    Lr.predict(x_test_shampoo)
```

Once you do this. Plot a scatter plot for the mean squared error and r2 for each polynomial order.