Understand Power of Polynomials with Polynomial Regression

Polynomial regression is a special case of linear regression. With the main idea of how do you select your features. Looking at the multivariate regression with 2 variables: x1 and x2. Linear regression will look like this:

y = a1 \* x1 + a2 \* x2.

Now you want to have a polynomial regression (let's make 2 degree polynomial). We will create a few additional features: x1\*x2, x1^2 and x2^2. So we will get your 'linear regression':

y = a1 \* x1 + a2 \* x2 + a3 \* x1\*x2 + a4 \* x1^2 + a5 \* x2^2

A polynomial term : a quadratic (squared) or cubic (cubed) term turns a linear regression model into a curve. But because it is the data X that is squared or cubed, not the Beta coefficient, it still qualifies as a linear model.

This makes it a nice, straightforward way to model curves without having to model complicated nonlinear models.

One common pattern within machine learning is to use linear models trained on nonlinear functions of the data. This approach maintains the generally fast performance of linear methods, while allowing them to fit a much wider range of data.

For example, a simple linear regression can be extended by constructing polynomial features from the coefficients. In the standard linear regression case, you might have a model that looks like this for two-dimensional data:



If we want to fit a paraboloid to the data instead of a plane, we can combine the features in second-order polynomials, so that the model looks like this:



The (sometimes surprising) observation is that this is still a linear model: to see this, imagine creating a new variable



With this re-labeling of the data, our problem can be written



We see that the resulting polynomial regression is in the same class of linear models we’d considered above (i.e. the model is linear in w) and can be solved by the same techniques.

By considering, linear fits within a higher-dimensional space built with these basis functions, the model has the flexibility to fit a much broader range of data.

Here is an example of applying this idea to one-dimensional data, using polynomial features of varying degrees:

|  |
| --- |
| from sklearn.preprocessing import PolynomialFeatures  import numpy as np  X = np.arange(6).reshape(3, 2)  X  poly = PolynomialFeatures(degree=2)  poly.fit\_transform(X) |

In some cases it’s not necessary to include higher powers of any single feature, but only the so-called interaction features that multiply together at most d distinct features. These can be gotten from PolynomialFeatures with the setting interaction\_only=True.

For example, when dealing with boolean features, x\_i^n = x\_i for all n and is therefore useless; but x\_i x\_j represents the conjunction of two booleans. This way, we can solve the XOR problem with a linear classifier:

|  |
| --- |
| from sklearn.linear\_model import Perceptron  from sklearn.preprocessing import PolynomialFeatures  import numpy as np  X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])  y = X[:, 0] ^ X[:, 1]  y  X = PolynomialFeatures(interaction\_only=True).fit\_transform(X).astype(int)  X |

Enough of theory , let’s start with implementation.

**Problem Statement :**

To understand the relationship between the degree of the independent variable and the dpenedent variable.

**Data details**

Boston House Prices dataset  
===========================  
  
Notes  
------  
Data Set Characteristics:   
  
 :Number of Instances: 506   
  
 :Number of Attributes: 13 numeric/categorical predictive  
   
 :Median Value (attribute 14) is usually the target  
  
 :Attribute Information (in order):  
 - CRIM per capita crime rate by town  
 - ZN proportion of residential land zoned for lots over 25,000 sq.ft.  
 - INDUS proportion of non-retail business acres per town  
 - CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)  
 - NOX nitric oxides concentration (parts per 10 million)  
 - RM average number of rooms per dwelling  
 - AGE proportion of owner-occupied units built prior to 1940  
 - DIS weighted distances to five Boston employment centres  
 - RAD index of accessibility to radial highways  
 - TAX full-value property-tax rate per $10,000  
 - PTRATIO pupil-teacher ratio by town  
 - B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town  
 - LSTAT - MEDV Median value of owner-occupied homes in $1000's  
  
 :Missing Attribute Values: None  
% lower status of the population  
  
 :Creator: Harrison, D. and Rubinfeld, D.L.  
  
This is a copy of UCI ML housing dataset.  
<http://archive.ics.uci.edu/ml/datasets/Housing>  
  
This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.  
  
The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic  
prices and the demand for clean air', J. Environ. Economics & Management,  
vol.5, 81-102, 1978. Used in Belsley, Kuh & Welsch, 'Regression diagnostics  
...', Wiley, 1980. N.B. Various transformations are used in the table on  
pages 244-261 of the latter.  
  
The Boston house-price data has been used in many machine learning papers that address regression  
problems.

**Tools used** :

Pandas , Numpy , Matplotlib , scikit-learn

**Python Implementation with code :**

**Import necessary libraries**

Import the necessary modules from specific libraries.

|  |
| --- |
| import numpy as np  import pandas as pd  %matplotlib inline  import matplotlib.pyplot as plt  from sklearn.model\_selection import train\_test\_split  from sklearn import datasets  from sklearn.metrics import mean\_squared\_error  from sklearn import ensemble |

**Load the data set**

Use pandas module to read the taxi data from the file system. Check few records of the dataset.

|  |
| --- |
| # #############################################################################  # Load data  boston = datasets.load\_boston()  print(boston.data.shape, boston.target.shape)  print(boston.feature\_names)  (506, 13) (506,) ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO' 'B' 'LSTAT'] |

|  |
| --- |
| data = pd.DataFrame(boston.data,columns=boston.feature\_names)  data = pd.concat([data,pd.Series(boston.target,name='MEDV')],axis=1)  data.head()  CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX PTRATIO B LSTAT MEDV  0 0.00632 18.0 2.31 0.0 0.538 6.575 65.2 4.0900 1.0 296.0 15.3 396.90 4.98 24.0  1 0.02731 0.0 7.07 0.0 0.469 6.421 78.9 4.9671 2.0 242.0 17.8 396.90 9.14 21.6  2 0.02729 0.0 7.07 0.0 0.469 7.185 61.1 4.9671 2.0 242.0 17.8 392.83 4.03 34.7  3 0.03237 0.0 2.18 0.0 0.458 6.998 45.8 6.0622 3.0 222.0 18.7 394.63 2.94 33.4  4 0.06905 0.0 2.18 0.0 0.458 7.147 54.2 6.0622 3.0 222.0 18.7 396.90 5.33 36.2 |

**Feature Selection :**

Let’s select one of the numerical fields for model fitting here “LSTAT” i.e. % of the population in the neighborhood which comes under lower economic status.

|  |
| --- |
| X = data[['LSTAT']]  y = data['MEDV'] |

**Train test split :**

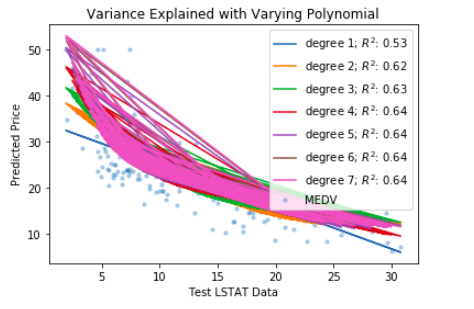
|  |
| --- |
| x\_training\_set, x\_test\_set, y\_training\_set, y\_test\_set = train\_test\_split(X,y,test\_size=0.10,  random\_state=42,  shuffle=True) |

**Training / model fitting with various degrees :**

Fit the model to selected supervised data. We will use the API called PolynomialFeatures which takes the parameter as the degree of the polynomial . With the given polynomial degree we will fit the data with linear regression model.

The motive of this fitting is to see if there is a better explanation of the variance with increase in the degree of polynomial of the selected feature.

|  |
| --- |
| ## Ploynomial Regression-nth order  # Ploynomial Regression-nth order  plt.scatter(x\_test\_set, y\_test\_set, s=10, alpha=0.3)  for degree in [1,2,3,4,5,6,7]:  model = make\_pipeline(PolynomialFeatures(degree), LinearRegression())  model.fit(x\_training\_set,y\_training\_set)  y\_plot = model.predict(x\_test\_set)  plt.plot(x\_test\_set, y\_plot, label="degree %d" % degree  +'; $R^2$: %.2f' % model.score(x\_test\_set, y\_test\_set))  plt.legend(loc='upper right')  plt.xlabel("Test LSTAT Data")  plt.ylabel("Predicted Price")  plt.title("Variance Explained with Varying Polynomial")  plt.show() |



We can notice that R^2 has increased from 0.53 - > 0.62 -> 0.63 -> 0.64 with rise in the degree of the polynomial initially and then it has saturated at 0.64. This kind of validate the belief that higher order polynomials are a good candidate to model non-linear pattern of tha data. However this doesn't give you a thumb rule to use larger degrees of polynomial because it has a limitation and it always depends upon the data .