Assignment 6 - Regression model

June 29, 2019

0.1 Assignment 6

0.1.1 Problem

Using Python, develop: * a linear regression model; * a quadratic regression model; * a cubic regression model; * a model of degree 10.

for example 1. Plot and compare the models.

0.1.2 Resolution

We must compute the value of x untill the 10th power. We do this below:

```
In [1]: import numpy as np
        from numpy.linalg import inv
        import matplotlib.pyplot as plt
        import copy

        def normalEquations(X, y):
            theta_ne = np.zeros((X.shape[0], 1))
            pinv = np.linalg.pinv(X.T.dot(X))
            theta_ne = pinv.dot(X.T).dot(y)
            return theta_ne

        x = np.asarray([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [10.11, 15.50, 24.60, 27.91, 42.87, 46.85, 50.56, 50.63, 61.53, 69.52, 86.66, 89.68, 90
        y = np.asarray([1.53, 5.47, 9.43, 10.08, 13.51, 15.12, 13.14, 18.94, 23.65, 22.27, 24.3

        x_1col = np.reshape([x[:,1]], (x.shape[0], 1))
        x_1 = copy.deepcopy(x)
        x_2col = np.power(x_1col,2)
        x_2 = copy.deepcopy(x_1)
        x_2 = np.append(x_2, x_2col, axis = 1)
        x_3col = np.power(x_1col,3)
        x_3 = copy.deepcopy(x_2)
        x_3 = np.append(x_3, x_3col, axis = 1)
        x_4col = np.power(x_1col,4)
        x_4 = copy.deepcopy(x_3)
        x_4 = np.append(x_4, x_4col, axis = 1)
```

```
x_5col = np.power(x_1col,5)
x_5 = copy.deepcopy(x_4)
x_5 = np.append(x_5, x_5col, axis = 1)
x_6col = np.power(x_1col,2)
x_6 = copy.deepcopy(x_5)
x_6 = np.append(x_6, x_6col, axis = 1)
x_7col = np.power(x_1col,2)
x_7 = copy.deepcopy(x_6)
x_7 = np.append(x_7, x_7col, axis = 1)
x_8col = np.power(x_1col,2)
x_8 = copy.deepcopy(x_7)
x_8 = np.append(x_8, x_8col, axis = 1)
x_9col = np.power(x_1col,2)
x_9 = copy.deepcopy(x_8)
x_9 = np.append(x_9, x_9col, axis = 1)
x_10col = np.power(x_1col,2)
x_10 = copy.deepcopy(x_9)
x_10 = np.append(x_10, x_10col, axis = 1)
```

After we can use the Normal equation $ w* = (X^{TX)}{-1}X^Ty $ to solve the problem and plot the results

In [2]:
```
t1 = normalEquations(x_1,y)
t1 = np.asarray(t1).T
t1 = np.reshape([t1], (t1.shape[0], 1))
y_1 = x_1.dot(t1)

t2 = normalEquations(x_2,y)
t2 = np.asarray(t2).T
t2 = np.reshape([t2], (t2.shape[0], 1))
y_2 = x_2.dot(t2)

t3 = normalEquations(x_3,y)
t3 = np.asarray(t3).T
t3 = np.reshape([t3], (t3.shape[0], 1))
y_3 = x_3.dot(t3)

t10 = normalEquations(x_10,y)
t10 = np.asarray(t10).T
t10 = np.reshape([t10], (t10.shape[0], 1))
y_10 = x_10.dot(t10)

plt.figure(figsize=(15, 10))
plt.scatter(x[:,1], y, s=30, c='r', marker='o', linewidths=1)
plt.plot(x_1col,y_1, label='Linear Regression')
plt.plot(x_1col,y_2, label='Polynomial Regression - degree 2')
plt.plot(x_1col,y_3, label='Polynomial Regression - degree 3')
plt.plot(x_1col,y_10, label='Polynomial Regression - degree 10')
```

```
plt.legend()
plt.show()
```