MACHINE LEARNING

DATA ANALYSIS

IIT PALAKKAD

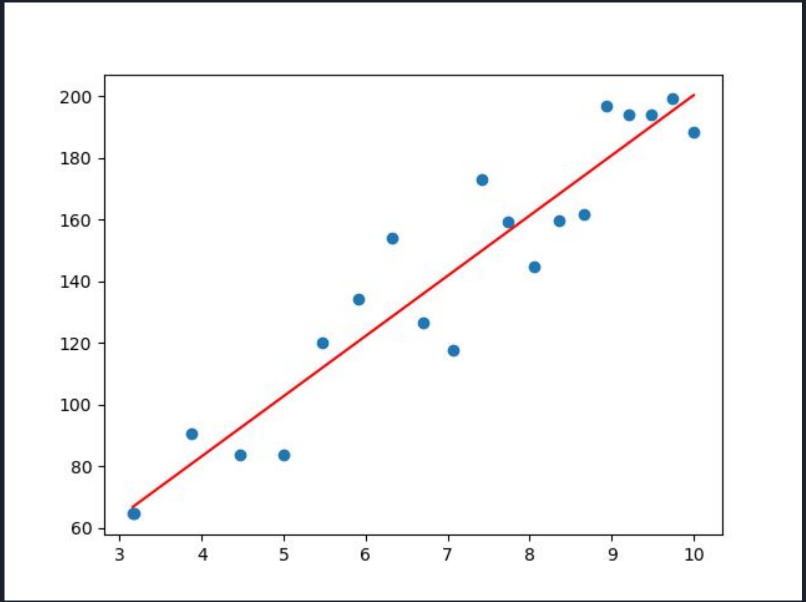
TOPICS

- Quick Recap of Linear Regression
- Notations
- Gradient Descent
- Polynomial Regression
- Implementation of Polynomial Regression (Python3)
- Gradient Descent Implementation (Python3)

LINEAR REGRESSION

Linear Regression is a type of regression in which output(Y) can be expressed as linear function of input(X).

$Y = \Theta.X + C;$



NOTATIONS:-

$x^{(i)}$ = input variable (feature)

$y^{(i)}$ = output or target variable (label)

$(x^{(i)}, y^{(i)})$ = a training example

Note that the ‘i’ is not the power of the variable. It's just a representation to denote a particular example.

m = number of training examples

$\therefore i = \{1, 2, \dots, m\}$

n= number of features

NOTATIONS:-

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \ldots\ldots + \theta_n x_n$ = hypothesis function

Where ,

θ_j = weights or parameters

For a line type of hypothesis:-

$h_\theta(x) = \theta_0 + \theta_1 x_1$ (slope-intercept form)

For simplicity let's assume $x_0 = 1$ then,

$h_\theta(x) = \sum \theta_j x_j$ where $j = \{0, 1, 2, \ldots.., n\}$

$\Rightarrow$ $h_\theta(x^{(i)}) = \sum_j \theta_j x^{(i)}_j$ (for a particular training example 'i')

NOTATIONS:-

Representing θ_j and x_j in the form of column vectors.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$$

$\therefore h_\theta(x^{(i)}) = \sum_j \theta_j x^{(i)}_j = \theta^T x^{(i)}$ (θ^T = transpose of parameters matrix)

NOTATIONS:-

“Cost Function” OR “mean squared error” OR “squared error function” : -

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

Sometimes, abbreviated as MSE.

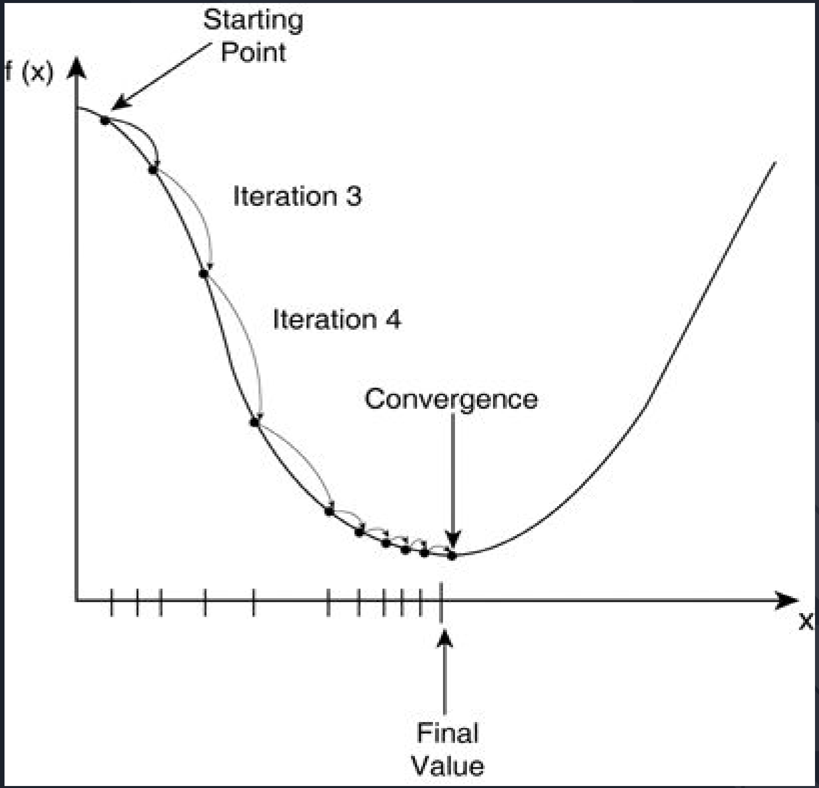
Gradient Descent(2D)

Let f(x) is a convex function.
A convex function has only global minimum.

Aim: To find X at which value of function f(x) is minimum.

Algorithm:-

Repeat till you reach minimum {
  X := X - α*(slope)
}
α is a very small constant and it is called as learning rate.



GRADIENT DESCENT FOR LINEAR REGRESSION (1-FEATURE):

$$Repeat - Until - Convergence\{$$

$$\theta_j := \theta_j - \alpha \left(\frac{\partial J}{\partial \theta_j} \right)$$

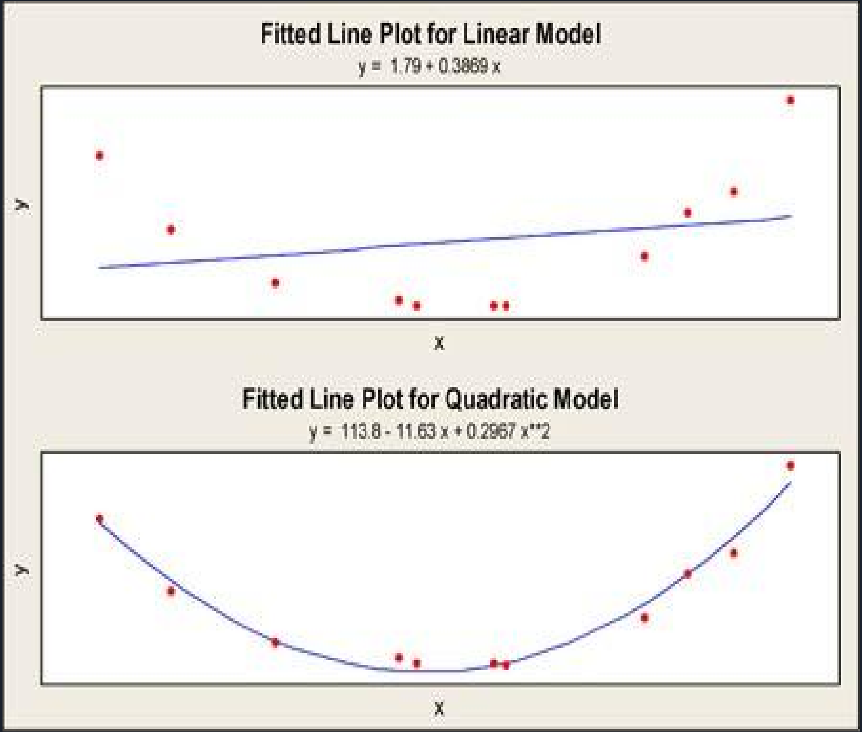
$$\}$$

POLYNOMIAL REGRESSION

Linear Regression enables you to find a best line which fits the data. I.e, your mapping will be in form of :

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \ldots\ldots + \theta_n x_n$

But there are times this type of regression model fail to fit a data shown in the adjoining image, while using polynomial features we can fit the model perfectly.



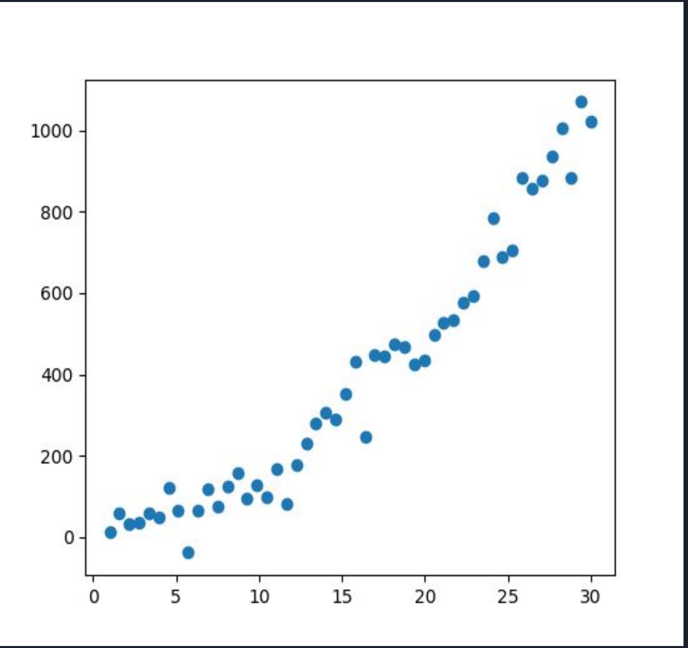
POLYNOMIAL REGRESSION

For the adjoining figure let's consider the hypothesis function as follows:-

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

On applying, gradient descent on this data, we can get a better cure than a line to this data.

The next slide deals on implementing gradient descent in python.



POLYNOMIAL REGRESSION

```
# Import the required modules

import numpy as np
import matplotlib.pyplot as plt

# Cost Function or Mean Squared Error

def costFunction(theta , X, y, m):
    Z = X.dot(theta) - y
    return (0.5/m)*(np.dot(Z.T, Z))


# Normal Equation Directly gives us the result , returning an array

#  containing the parameters : [θ0 , θ1 , θ2 ]

def normalEqn(X, y):
    return np.linalg.pinv(np.dot(X.T,X)).dot(X.T).dot(y)
```

POLYNOMIAL REGRESSION

Load the data file. Download Here: http://goo.gl/ZWKPbD

```
data = np.loadtxt('dat.txt')

X = data[:,0].reshape(-1,1)      # X is a column-matrix
y = data[:,1].reshape(-1,1)      # y is also column-matrix
m = X.shape[0]
```

Generate a x^2 feature and stack to the X matrix to get a new matrix with

two columns one for original X and other for X^2

```
X = np.column_stack((X , np.square(X)))
```

POLYNOMIAL REGRESSION

Accounting for x_0=1 feature

```
X = np.column_stack((np.ones((m,1)) , X))
```

After doing all this pre-processing one can use either Normal Equation Method or Gradient Descent to Find Optimal Theta

To keep the discussion simple let's keep implementation of gradient descent out of the current discussion. The code is given at the end of this ppt, one can go through it.

Now one can use normal equation by calling the function normalEqn(X,y) to get the optimal parameters containing the parameters : $[\theta_0 , \theta_1 , \theta_2]$

```
bestTheta = normalEqn(X, y)
```

POLYNOMIAL REGRESSION

Plotting the data

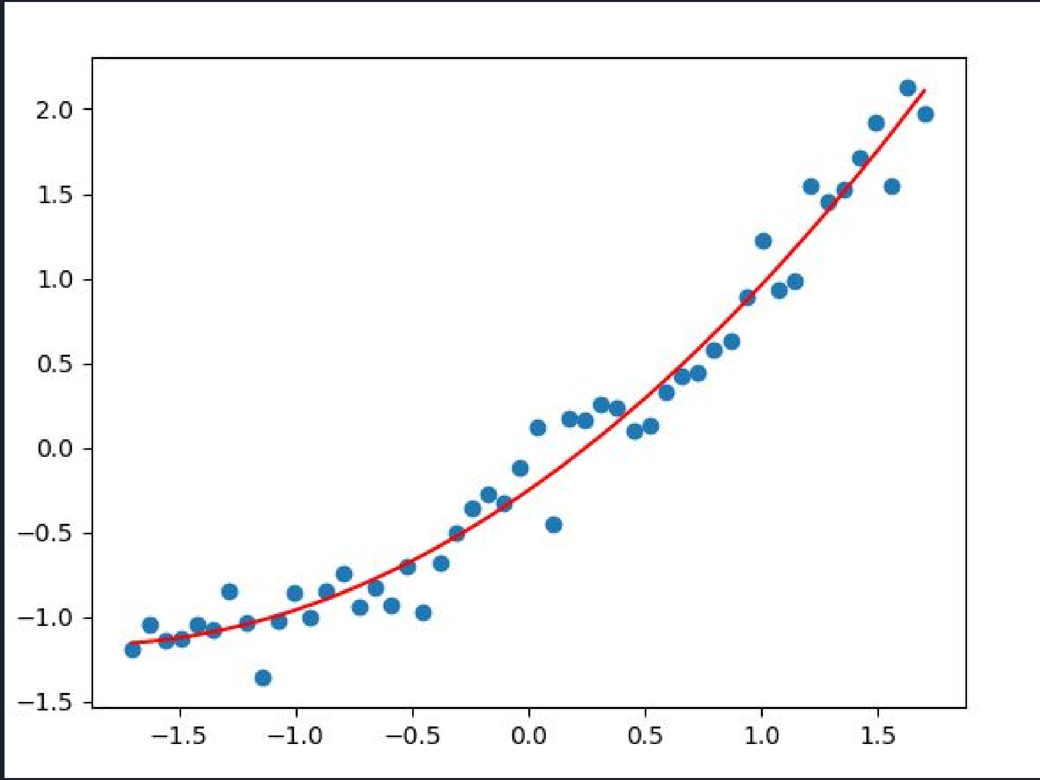
Note that X[:,0] contains only ‘1’ corresponding to the x_0(=1) feature. I am not cheating on you, initially X[:,0] was our x-coordinate but after some of the column stack operation the order change. Please look the previous commands.

Now, X[:,1] is our original x-coordinate.

```
x_cord = np.linspace(np.min(X[:,1]) , np.max(X[:,1]))
y_cord = np.dot(bestTheta.T , X.T).reshape(m,1)
plt.figure()
plt.scatter(X[:,1] , y[:,0])
plt.plot(x_cord , y_cord , 'r-')
plt.show()
```

POLYNOMIAL REGRESSION

If everything went well then you would see a plot like this:-



GRADIENT DESCENT FUNCTION

```
def gradientDescent(theta , X , y ):
    # m = no. of training examples
     m = y.shape[0]
     alpha = 0.037                  # learning rate parameter
     eps = 1e-8                     # maximum change in MSE
     maxIter=100000                 # maximum iteration

     theta = np.array(theta).astype(float)
     X = np.array(X).astype(float)
     y = np.array(y).astype(float)
     iter =0                        # initializing iteration no.
     prev = costFunction(theta, X, y, m)
```

GRADIENT DESCENT FUNCTION

```
while iter<maxIter:
        iter += 1
        Z = (X.dot(theta) - y)
        theta = theta - (alpha/m)*(np.dot(X.T , Z))
        J=costFunction(theta, X, y, m)
        print(iter , J)
        if(abs(prev-J)<eps):
            break
        prev=J
 return theta
```

You can call the function by typing this:-

bestTheta = `gradientDescent(theta , X , y )`

You can try polynomial regression by using gradient descent algorithm. Note that changing the parameters like alpha, maxiter , etc. can affect the program. I will suggest you to change the parameters to see the effect.

Thanks!

Kaushal Kishore (111601008)
Amit Vikram Singh (111601001)
Sai Suchith Mahajan (121601016)