Machine Learning(IDC410) Report: Programming Exercise 1

Chhavi Chahar
Roll No: MS18136
email: chhavichahar01@gmail.com, ms18136@iisermohali.ac.in

Contents

---

1 Code

Question 1: Generating the data set

```python
import numpy as np

def gen_data(sigma,n,m):
    X = np.random.rand(n,m+1)
    X[:,0]=1      # first column set to 1
    e = sigma*np.random.randn(n,1)   # Gaussian distribution of noise 'e' with mean = 0 an standard deviation sigma
    beta = np.random.rand(m+1,1)     # random coefficients
    Y= np.add(np.matmul(X,beta),e)   # Y = bX+e

    return X,beta,Y


gen_data(0.05,3,4)
```

Question 2: Linear regression

$$Costfunction = \frac{1}{n}(Y - Y_{predicted})^T(Y - Y_{predicted})$$

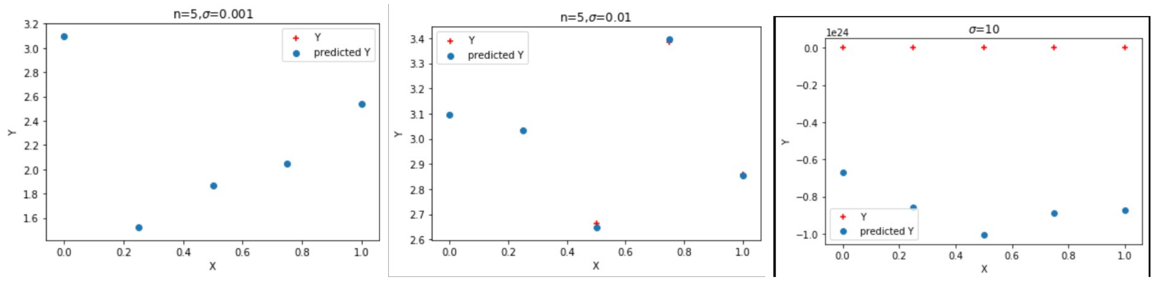
$$Gradient = -X^T(Y - Y_{predicted})$$

where $Y_{predicted} = Xb$

```python
def lin_reg(gen_data,k,t,delta):

    X,beta,Y = gen_data(sigma,n,m)
    z = float(len(X))

    b = np.random.rand(m+1,1)  #random initialisation of the parameters
    previous_cost = 0   #initial cost set to 0
    cost = []

    for i in range(k): # k is no. of iterations
        cost.append(previous_cost)
        y_pred = np.matmul(X,b)  # predicted y
        temp = np.subtract(Y,y_pred)  # Y-Xb

        current_cost  = (1/z)*np.matmul(np.transpose(temp),temp)  #cost function
        if abs(current_cost-previous_cost)<=t:  # t - threshold condition on change in cost function
            break
        previous_cost = current_cost

        grad = - np.matmul(np.transpose(X),temp)  # gradient calculation
        b = b - delta*grad  #improved b, delta is step size/learning parameter

    print(' actual beta values :\n {}'.format(beta))
    print("coefficients learnt using gradient descent:\n {} \ncost: {} ".format(b,cost[-1]))



sigma = 0.07
n = 3
m = 4
lin_reg(gen_data,1000000,1e-10,0.105)
```

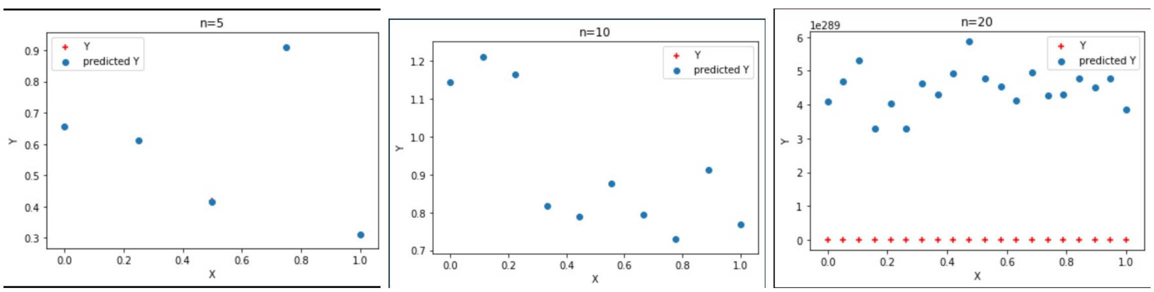
2 Impact of variation of n and σ on linear regression

Impact of σ: σ represents random noise in the sample. When we apply linear regression to learn the parameters, smaller the noise,faster the convergence. Larger the noise, higher the variation in the sample. As the data becomes more and more dispersed with increasing noise, the regression curve will flatten and



(a) (b) (c)

Figure 1: Y and predicted Y with changing noise



(a) (b) (c)

Figure 2: Y and predicted Y with changing n

will not be a 'good' fit.
For $\sigma = 0.001$ and $\sigma = 0.01$, the predicted values have completely masked the true values whereas for $\sigma = 10$, the values are far apart.

Impact of n: For a fixed m (=5) and zero noise, data points for n=5 and n=10 are completely masked by the predicted data points, where as for n=20 gives a very bad fit.
This suggests that more number of data points alone do not guarantee a better fit and faster convergence. If enough important features can be identified with a smaller amount of data, the fit will be better. Large/small dataset does not imply better/worse fit. Large data with a high number of outliers is not helpful.

.

3 List of files submitted

1. .ipynb notebook containing code
2. pdf report