Stochastic_Gradient_Code

September 20, 2021

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
X = 2 * np.random.rand(100,1)
y = 4 +3 * X+np.random.randn(100,1)
```

[4]:
```
def  cal_cost(theta,X,y):
    '''

    Calculates the cost for given X and Y. The following shows and example of a␣
 ↪single dimensional X
    theta = Vector of thetas
    X     = Row of X's np.zeros((2,j))
    y     = Actual y's np.zeros((2,1))

    where:
        j is the no of features
    '''

    m = len(y)

    predictions = X.dot(theta)
    cost = (1/2*m) * np.sum(np.square(predictions-y))
    return cost
```

[5]:
```
def stocashtic_gradient_descent(X,y,theta,learning_rate=0.01,iterations=10):
    '''
    X    = Matrix of X with added bias units
    y    = Vector of Y
    theta=Vector of thetas np.random.randn(j,1)
    learning_rate
    iterations = no of iterations

    Returns the final theta vector and array of cost history over no of␣
 ↪iterations
    '''
    m = len(y)
```

```
    cost_history = np.zeros(iterations)


    for it in range(iterations):
        cost =0.0
        for i in range(m):
            rand_ind = np.random.randint(0,m)
            X_i = X[rand_ind,:].reshape(1,X.shape[1])
            y_i = y[rand_ind].reshape(1,1)
            prediction = np.dot(X_i,theta)

            theta = theta -(1/m)*learning_rate*( X_i.T.dot((prediction - y_i)))
            cost += cal_cost(theta,X_i,y_i)
        cost_history[it]  = cost

    return theta, cost_history
```