Programming Exercise 1

Matheus M. dos Santos
Computer Engineering
CEFET-MG
matheusmstos@yahoo.com.br

Abstract

The goal of this report is to show how Linear Regression and the Gradient Descent works in machine learning. It tries to explain with it's own words, how the author understood this concepts.

1 Linear Regression

1.1 Theory

The Linear Regression it's a model that tries to relate a dependent variable to an independent variable. So we can have two types of models: with one independent variable (one feature) which is called Simple Linear Regression and with multiple independent variables, which is called Multivariate Linear Regression (multiple features).

This model uses pairs of data (x,y), and comes up if a function. It can be any type of function, but the goal is to find a function that best describes the problem. So, when the problem is described by the function, it can be used to predict y values.

So, Linear Regression it's a model that tries to analyze data to describe and predict a real life problem.

1.2 Examples

$$h_\beta(x) = \beta^T x = \beta_0 + \beta_1 x_1$$

Figure 1: Linear Regression

In these examples, it can be seen that we have pairs of data in thw form of (x,y), and we try to describe their relation by a function.

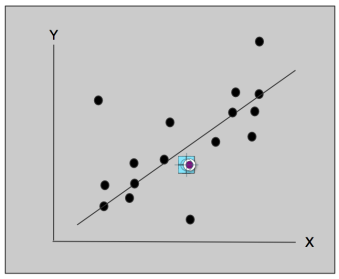

Figure 2: Linear Regression: Line

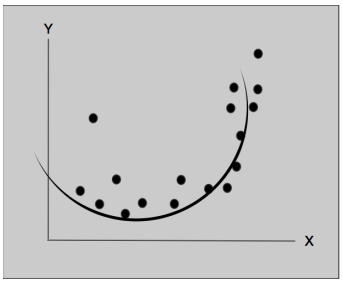

Figure 3: Linear Regression: Quadratic

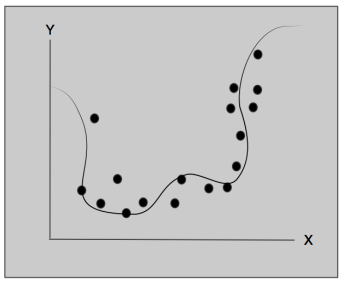

Figure 4: Linear Regression: Xn

1.3 Code

```
def compute_cost(X, y, beta):

    #X is the population X = x1, x2, x3... xn
    #X has the shape [beta, pop]
    #beta has this shape [b1, b2]
    #hb(pop) = profit predicted

    n    = X.shape[0]
    sumt = 0

    for i in range(0,n):
        hb    = beta.item(0,1) + X.item(i,1)*beta.item(0,0)
        sub   = (hb - y.item(i)) * (hb - y.item(i))
        sumt += sub

    Jb = sumt/(2*n)
    return Jb
```

2 Task 0: Gradient Descent

2.1 Theory

Gradient descent makes an analysis of the error that Linear Regression model gives compared to the actual desired answer. Because Linear Regression parameters (betha) might not be precise and accurate. So Gradient Descent comes to minimize the error (the difference between predicted answer and actual answer).

The algorithm does that by learning with itself. It does lots of interactions to learn with its own answers and gives the output of new and more precise parameters (betha).

So, this means that Gradient Descent is an optimization of the Linear Regression modeling. But if our problem is a cost function, the optimization is to find the minimum of our problem.

$$\beta_j = \beta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_\beta(x_i) - y_i) x_{ij}$$

Figure 5: Gradient Descent

2.2 Examples

In the examples below, it ca be seen that the Gradient Descent change the linear regression and optimize the function itself.

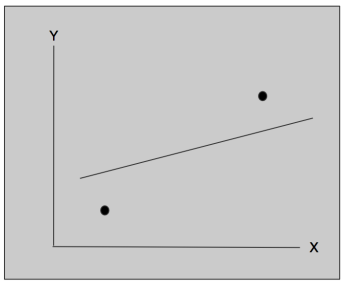

Figure 6: Iteration x

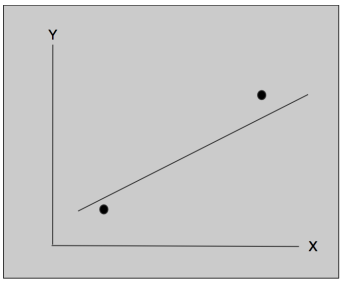

Figure 7: Iteration x+1

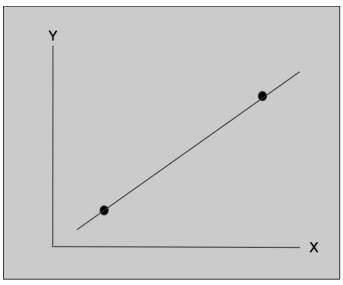

Figure 8: Iteration x+2

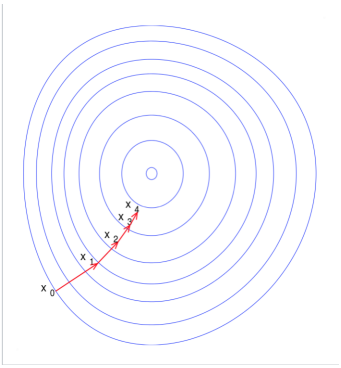

Figure 9: This shows that in each interaction, we optimize the function of cost and move to find the minimum.

2.3 Code

```
def gradient_descent(X, y, theta, alpha, iters):
    '''
    alpha: learning rate
    iters: number of iterations
    OUTPUT:
    theta: learned parameters
    cost:  a vector with the cost at each training iteration
    '''
    temp       = np.matrix(np.zeros(theta.shape))
    parameters = int(theta.ravel().shape[1])
    cost       = np.zeros(iters)
    n          = X.shape[0]
    sumt       = 0


    for i in range(iters):
        for j in range(parameters):
            #hb   = theta.item(0,1) + X.item(i,1)*theta.item(0,0)
            dif  = (X * theta.T) - y
            mult = np.multiply(dif, X[:,j])
            sumt = np.sum(mult)

            temp[0,j] = theta[0,j] - ((alpha/n) * sumt)

        theta = temp
        cost[i] = compute_cost(X, y, theta)

    return theta, cost
```

References

[1] https://www.statisticssolutions.com/what-is-linear-regression/

[2] https://onlinecourses.science.psu.edu/stat501/node/252

[3] https://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html

[4] https://en.wikipedia.org/wiki/Gradient$_d escent$

[5] https://www.youtube.com/watch?v=L-Lsfu4ab74