Regression

Monday, June 28, 2021 4:32 PM

Residual/Error :
Line which has min. error considered as to be best fitted line.

$$\min \Sigma (y_i - Y_i)^2 \text{ or } \min \Sigma (y_{actual} - Y_{predicted})^2$$

Where yi = actual point
Yi = point on line

m*, c* = $arg_{m,c}$ **min** $\{ \Sigma (y_i - (mx + c))^2 \}$

Where m* = Optimum value of slope
c* = Optimum value of Intercept
This equation is called ordinary least square Equation.

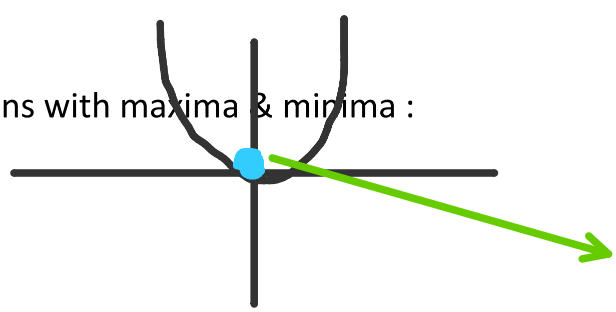
Gradient Descent :
Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost).
It is an Iterative algorithm.

Cost Function :

It is a function that measures the performance of a Machine Learning model for given data. Cost Function quantifies the error between predicted values and expected values and presents it in the form of a single real number.

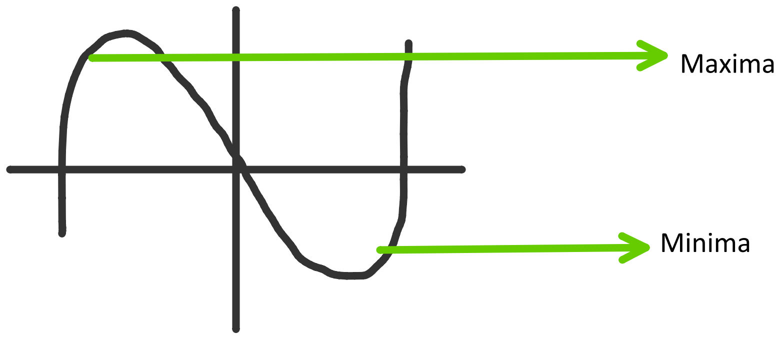
$= 1/n * (y_{actual} - Y_{predicted})^2$

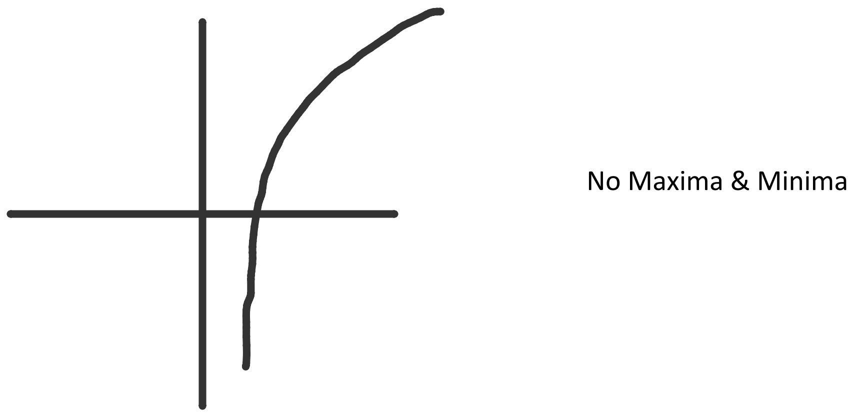
Functions with maxima & minima :

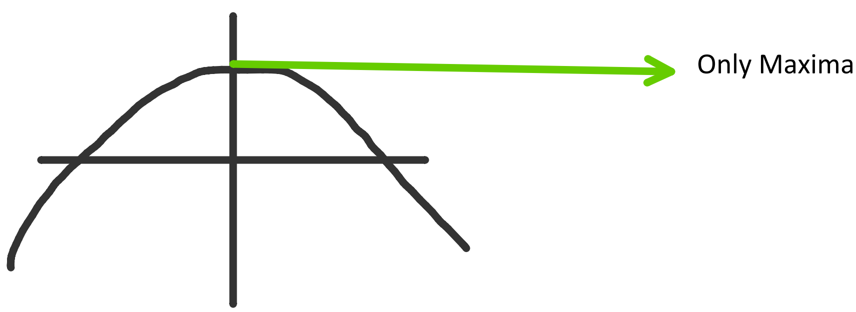


Y = x^2

Minima

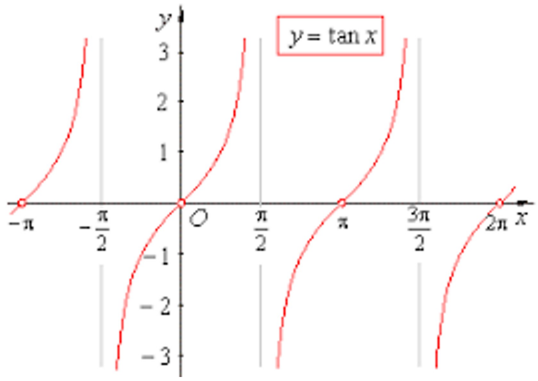






Global Minima : Lowest minima
Local Minima : Other than Global

- Slope is always = 0 at Maxima and Minima because of line parallel with x-axis($\tan\theta = 0$) whether there is local or global minima.



- Slope always be calculate using differentiation because it gives minimized value.
- Slope (Minima / Maxima) = 0

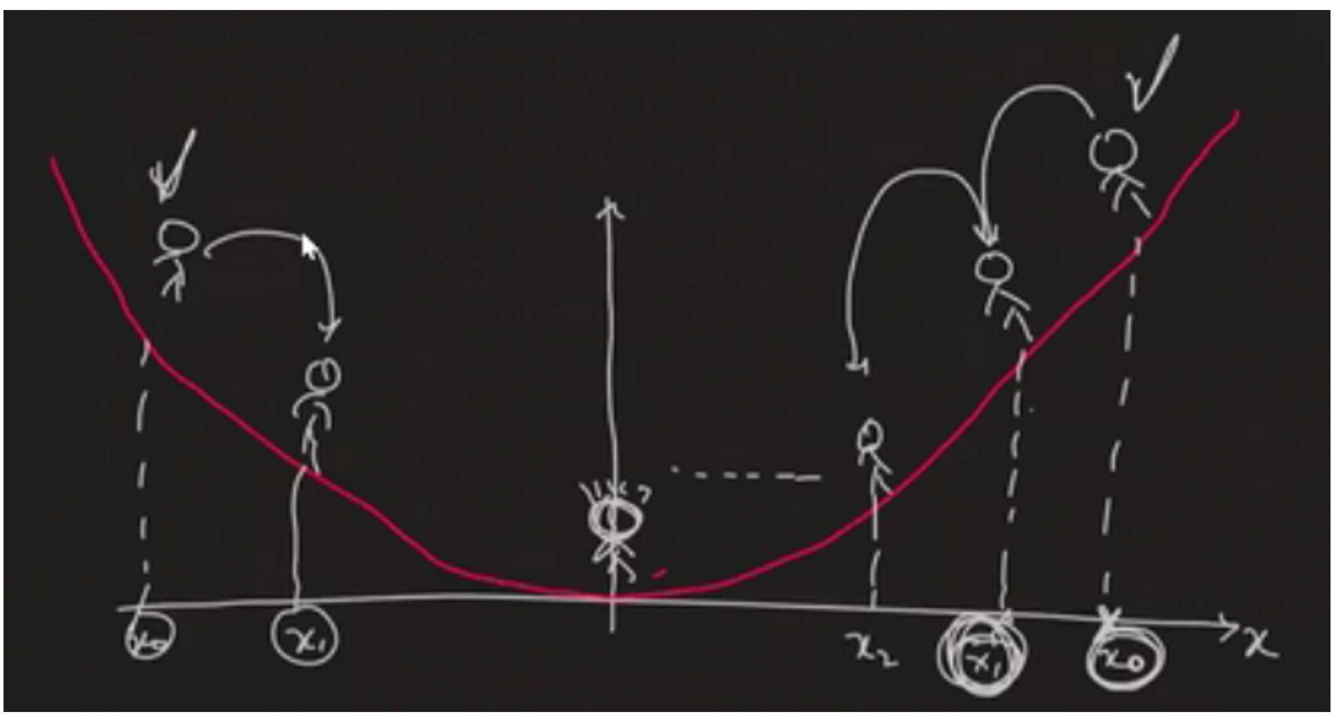
$$\frac{d}{dx}(x^2 - 3x + 2) = 0$$

$$2x - 3 = 0$$

$$x = \frac{3}{2}$$

It means Slope is minima or maxima at 3/2 or 1.5

How to Find Gradient Descent :



1. Randomly Picking a point x_o .
2. Calculate the slope and jump

$$x_1 = x_o - n\left[\frac{d}{dx}f\right]x_o$$

Where n - Learning rate (large means large jump and vice versa)
we have to take optimal value of n otherwise it jumps to another side.

$$\frac{d}{dx} f = slope\ at\ Xo$$

3. Repeat until we get the X* at minima/maxima or get at convergence.

So, In Gradient Descent we need to find the optimum values of m* and c*

m*, c* = $arg_{m,c}$ **min {** $\boldsymbol{\Sigma}$ **(** y_i**- (mx + c))**^{2} **}**

Hence final form of Gradient Descent for Linear Regression :

1. Randomly picking a point:
   m_o = initialize the random slope pt.
   c_o = initialize the intercept pt.

2. Find the values of :
   $m_1 = m_o - \eta \left[\frac{\delta f}{\delta m}\right]_{m_o} \quad = \quad m_1 = m_o - \eta \left[\frac{\triangle f}{\triangle m}\right]_{m_o}$
   $c_1 = c_o - \eta \left[\frac{\delta f}{\delta c}\right]_{c_o}$ (We have use + in case of Maxima)
3. Repeat step 2 until convergence.

<u>Differentiation of function f</u>

$$\frac{\delta f}{\delta m} = \Sigma \frac{\delta}{\delta m}(y_i - (mx_i + c)$$
$$= 2\ \Sigma_{i=1}^{n}\ (y_i - (mx_i + c)\ (-x_i)$$

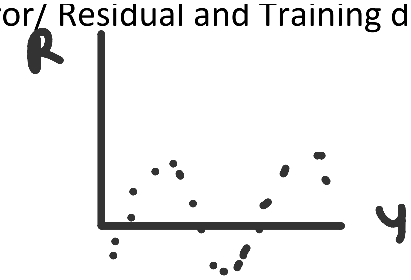
$$\frac{\delta f}{\delta c} = \Sigma \frac{\delta}{\delta c}(y_i - (mx_i + c)$$
$$= 2\ \Sigma_{i=1}^{n}\ (y_i - (mx_i + c)\ (-1)$$

Assumption :

- Linear Relationship between Independent and Dependent variables.
- Error/ Residual on Training data $\sim$ N(0, σ^2)
- Error/ Residual and Training data - Independent of each other.

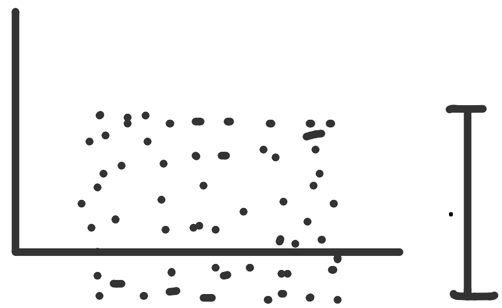
R

- Error/ Residual and Training data - Independent of each other.



In this case, R follows some pattern so it cannot be considered as indep.

- Error/ Residual on Training data expect to be HOMOSCADASTICITY.



KNN

Algorithm :

1. Initialize the List of Distances = [ ].
2. For each of xi in the dataset:
   Compute the distance d(xi, xq) = di
   Store them in list as (xi, yi, di)

3. Sort the list on the basis of distances.

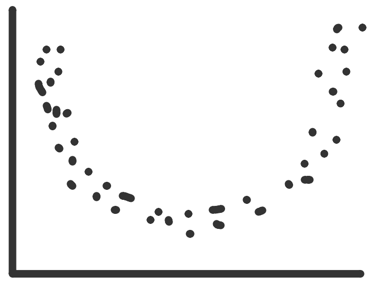
4. Take the smallest 'k' distances and store them in list 'KNN' .

5. For each xi in KNN:
   Take mean of the k no of yi values .

Multiple Linear Regression :

We have more than 1 independent variable in MLR. If there is relationship between independent var. then we call them as multicollinearity which is a big issue in ML, that can cause Interpretable issue.

Polynomial Linear Regression



We cannot use a linear Regression to fit this line. We have to use higher degree of Polynomial to fit the line into data

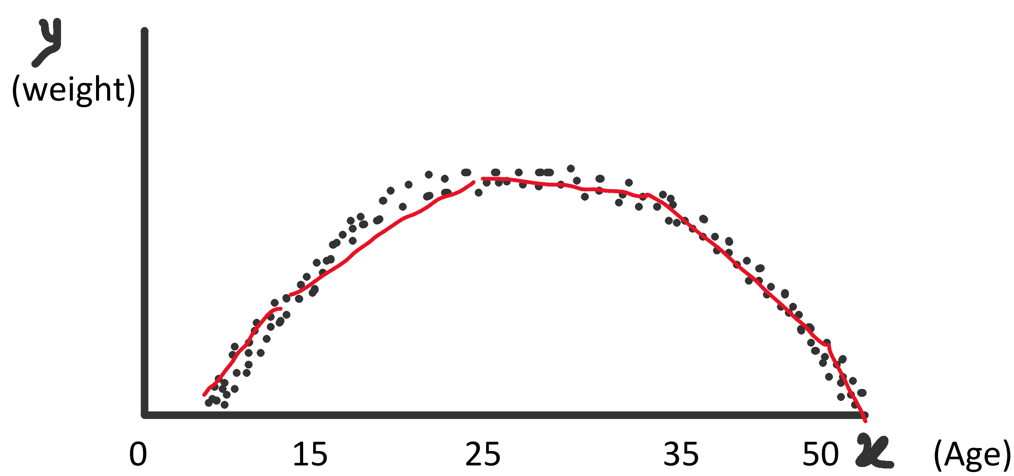
MLR : $w1x1 + w2x2 + w3x3 + w0 = 0$

$$[\ w1\ w2\ w3\ \ w4] \begin{bmatrix} x1 \\ x2 \\ x3 \\ 1 \end{bmatrix} = 0$$

Here, Weightage / Coefficient part(w1, w2..) is always linear while we can change the feature vector i.e. x1, x2, x3 to 2nd degree of polynomial.

Turning Points for No. of Degree of polynomial :
- Linear data / 1st degree : has 0 turning points
- 2nd degree : has 1 turning point
- 3rd degree : has 2 turning point

Decision Tree :
Homogeneity Measure for Classification : Entropy / Gini Impurity
Homogeneity Measure for Regression : MSE / R2_score



{15, 25 , 35, 50} : Thresholds in Decision Tree

MSE - weighted(MSE)

= MSE - ($MSE_1 + MSE_2$)

Whichever Threshold gives the maximum value of (MSE - ($MSE_1 + MSE_2$)), we have to take that threshold value .