Deep Learning Using TensorFlow

Dr. Ash Pahwa

Lesson 6:

Gradient Descent & Backpropagation Algorithms

Lesson 6.1: Gradient Descent Algorithm

Outline

- What is a Gradient?
- What is Gradient Descent Algorithm?
- Gradient Descent Algorithm
  - Minimum of a 2-Variable Function
  - Minimum & Maximum of a 2-Variable Function
  - Minimum of 3 Variable Function
  - Solving Regression problem
  - Solving Regression problem – Iris Dataset

Lesson#4.1/Slide#11
Example#1: Neural Network: Data = XOR
5. Define Cost & Optimization Functions

- Cost Function
  - Cross Entropy Cost Function
- Gradient Descent
  - Feed “Learning Rate” as a parameter to the optimization function

```
#######################################################
# 5. Define Cost and Optimization Function
# Cost = Cross Entropy Cost Function
# Optimization = Gradient Descent
#
cost = tf.reduce_mean(-Y*tf.log(compOutput) - (1-Y)*tf.log(1-compOutput))

optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

Lesson#4.1/Slide#29
Example#2: Neural Network: Data = Iris
Define Cost and Optimization Function

- Cost Function
  - Cross Entropy Cost Function
- Gradient Descent
  - Feed "Learning Rate" as a parameter to the optimization function

```
cost    = tf.reduce_mean
(tf.nn.softmax_cross_entropy_with_logits_v2(labels=outputs, logits=pred_tensor))

updates = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
```

Lesson#5.1/Slide#19
Example#1: Linear Regression: 2 Vars
Build the TensorFlow Graph

```
graph = tf.Graph()
with graph.as_default():
    slope = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
    intercept = tf.Variable(tf.zeros([1]))
    response = slope*x_point + intercept

    #############################################
    # Define Cost + Optimization Functions
    cost = tf.reduce_mean(tf.square(response - y_point))
    optimizer = tf.train.GradientDescentOptimizer(0.5).minimize(cost)
```

- $Residual = \ y_i - (mx_i + b)$
- $Residual^2 = \ (y_i - (mx_i + b))^2$
- $Residuals\ Sum\ of\ Squares = (RSS) = \ \sum_{i=1}^{N}(y_i - (mx_i + b))^2$

Lesson#5.2/Slide#31
Example#2: Linear Regression: Multi variables
Define the ‘cost’ and ‘optimization’ functions

```
#############################################
# 6. Define the 'cost' and 'optimization' Functions
# Initialize the variables
#

cost = tf.reduce_sum(tf.square(computed_y - y))

optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

init = tf.global_variables_initializer()
```

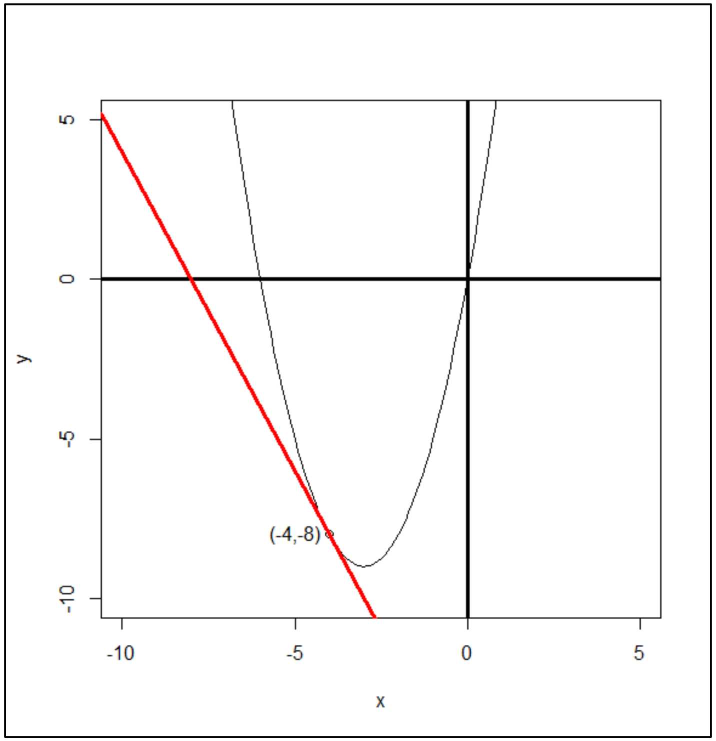
What is a Gradient?

Gradient Vector: 2 Variables

- Definition: 2 variables x, y
  - The gradient vector of a function y=f(x)
  - $\nabla y = \nabla f(x) = \frac{\partial f}{\partial x} = \frac{df}{dx}$
- Gradient property
  - Gradient vector gives the direction of fastest increase (or decrease) of function f(x)

Example: Gradient

- $y = x^2 + 6x$
  - $\frac{dy}{dx} = 2x + 6$
  - $x = -4$
  - $y = (-4)^2 + 6(-4) = 16 - 24 = -8$
  - $\frac{dy}{dx} = 2x + 6 = 2 * (-4) + 6 = -2$
  - Gradient at point(-4,-8) = -2



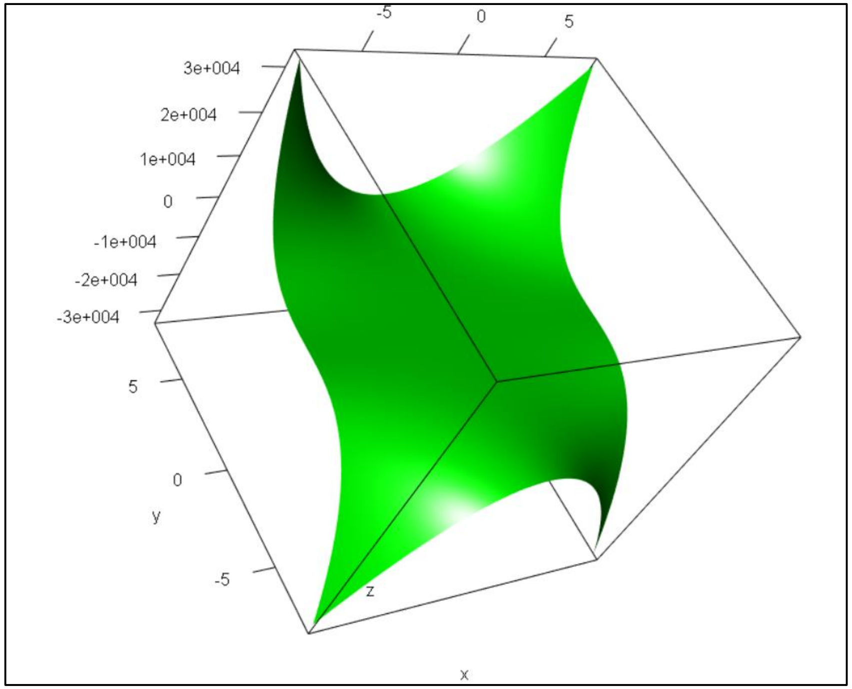
Gradient Vector: 3 Variables

- Definition: 3 variables x, y, z
  - The gradient vector of a function z=f(x, y)
  - $\nabla z = \nabla f(x, y) = \frac{\partial f}{\partial x}\boldsymbol{i} + \frac{\partial f}{\partial y}\boldsymbol{j}$
- Gradient vector has 2 properties
  - Gradient vector gives the direction of fastest increase (or decrease) of function f(x, y)
  - Gradient vector is orthogonal (perpendicular) to the level surface (contour)

Example#1: Gradient

- $z = f(x, y) = x^2y^3 - 4y$
- $\frac{\partial f}{\partial x} = 2xy^3$
- $\frac{\partial f}{\partial y} = 3x^2y^2 - 4$
- $\nabla f(x, y) = 2xy^3\mathbf{i} + (3x^2y^2 - 4)\boldsymbol{j}$
- $\nabla f(2, -1) = -4\mathbf{i} + 8\boldsymbol{j}$

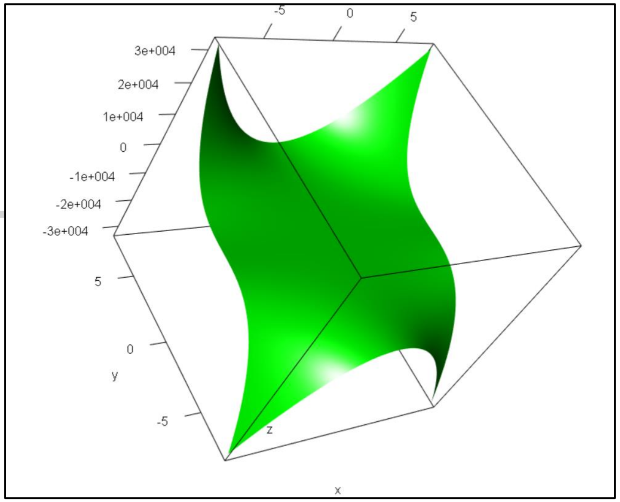
$f(x, y) = x^2y^3 - 4y$

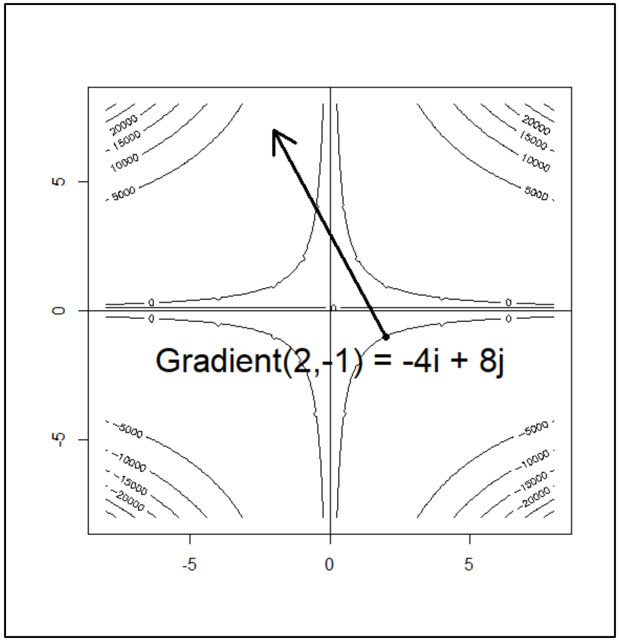


Example#1 : Gradient

- $z = f(x,y) = x^2y^3 - 4y$
- $\frac{\partial f}{\partial x} = 2xy^3$
- $\frac{\partial f}{\partial y} = 3x^2y^2 - 4$
- $\nabla f(x,y) = 2xy^3\mathbf{i} + (3x^2y^2 - 4)\boldsymbol{j}$
- $\nabla f(2,-1) = -4\mathbf{i} + 8\boldsymbol{j}$

$f(x,y) = x^2y^3 - 4y$





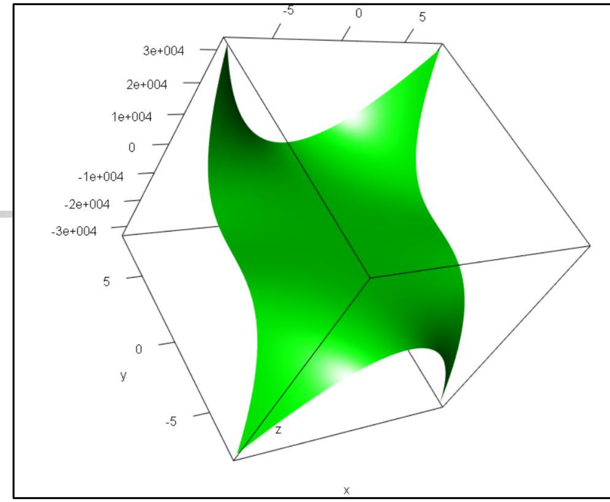
- Gradient vector has 2 properties
  - Gradient vector gives the direction of fastest increase (or decrease) of function f(x,y)
  - Gradient vector is orthogonal (perpendicular) to the level surface (contour)

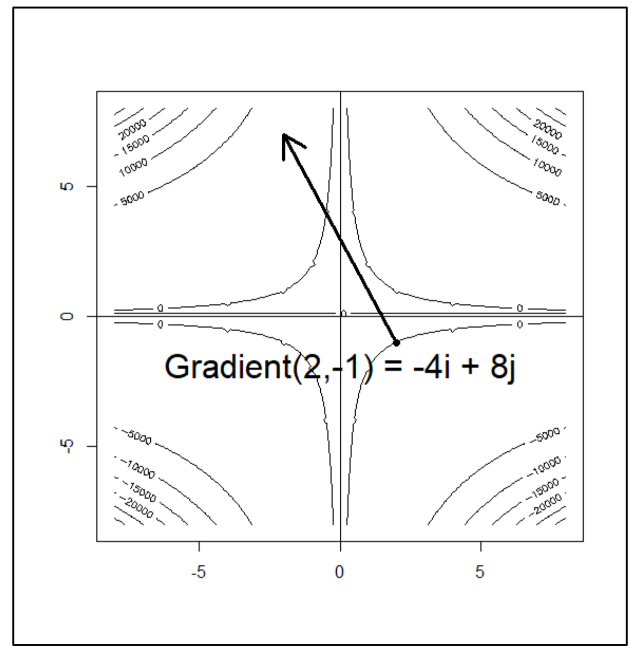
R Code to Generate Plot

- $z = f(x,y) = x^2y^3 - 4y$
- $\dfrac{\partial f}{\partial x} = 2xy^3$
- $\dfrac{\partial f}{\partial y} = 3x^2y^2 - 4$
- $\nabla f(x,y) = 2xy^3\mathbf{i} + (3x^2y^2 - 4)\boldsymbol{j}$
- $\nabla f(2,-1) = -4\mathbf{i} + 8\boldsymbol{j}$

```
x = seq(-8,8,0.1)
y = seq(-8,8,0.1)
z1 = function (x,y) { (x^2)*(y^3) - 4*y }
z = outer(x,y,z1)

library(rgl)
persp3d(x,y,z,col="green")

contour(x,y,z)
text(0,-2,"Gradient(2,-1) = -4i + 8j",cex=2)
points(2,-1,pch=19)

# Since gradient = -4i + 8j
# End point of the gradient
# -4 + 2 = -2
# 8 - 1 = 7
arrows(2,-1, -2,7,lwd=3 )
abline(v=0)
abline(h=0)
```

$f(x,y) = x^2y^3 - 4y$





What is Gradient Descent Algorithm?

What is Gradient Descent Algorithm?

- Suppose a function $y = f(x)$ is given
  - Gradient Descent algorithm allows us to find the values of 'x' where the 'y' value becomes minimum or maximum values
- The Gradient Descent algorithm can be extended to any function with 2 or more variables $'z = f(x, y)'$

- Function y=f(x)
- Gradient Descent Algorithm: Minimum
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial y}{\partial x} \|_{x^t}$

- Function z=f(x,y)
- Gradient Descent Algorithm: Minimum
  - Initialize the value of x and y
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial z}{\partial x} \|_{x^t, y^t}$
    - $y^{t+1} \leftarrow y^t - \eta \frac{\partial z}{\partial y} \|_{x^t, y^t}$

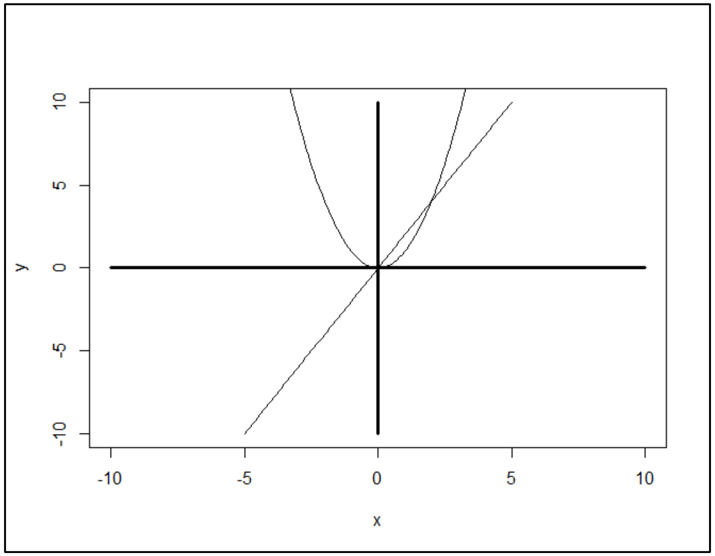
Finding Minimum of a Function Using Gradient Descent Algorithm

2 Variables (x, y) Function

Example 1
Find the value of ‘x’ where the value of ‘y’ is minimum

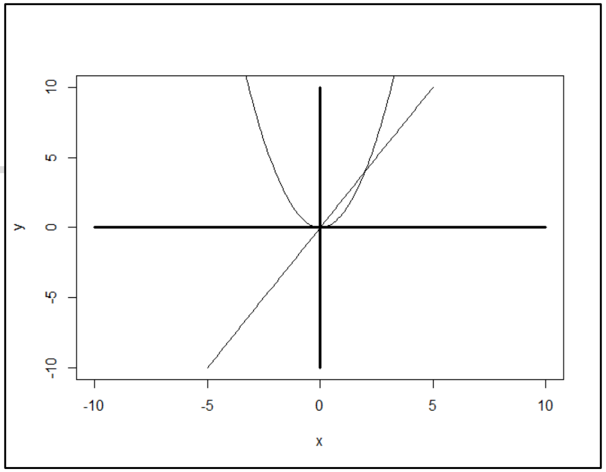
- $y = x^2$
- To find minimum point, equate the first derivative = 0
  - $\frac{dy}{dx} = 2x = 0$
  - $x = 0$
- To find the value of ‘x’
  - Where the value of ‘y’ is minimum
  - Correct answer: x = 0

```
> x = seq(-5,5,0.1)
> y = x^2
> dy_dx = function (w1) { 2*w1 }
> plot(x,y,type='l',xlim=c(-10,10),ylim=c(-10,10))
> lines(x,dy_dx(x))
> lines(c(0,0),c(-10,10),lwd=3)
> lines(c(-10,10),c(0,0),lwd=3)
```

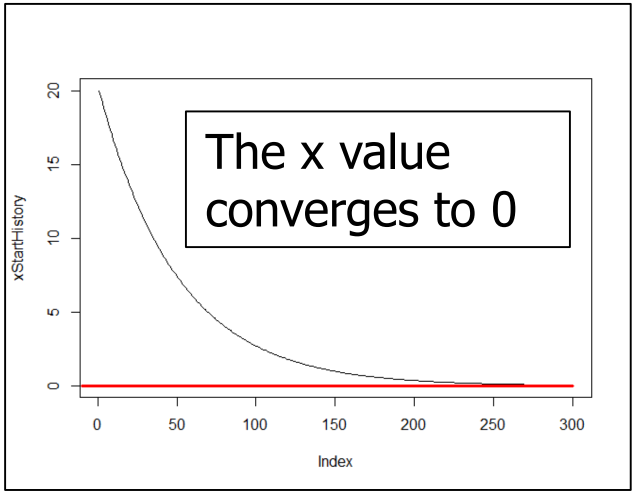


Example 1: R Code

- $y = x^2$
- To find minimum point, equate the first derivative = 0
  - $\frac{dy}{dx} = 2x = 0$
  - $x = 0$
- Gradient Descent Algorithm
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial y}{\partial x} \|_{x^t}$



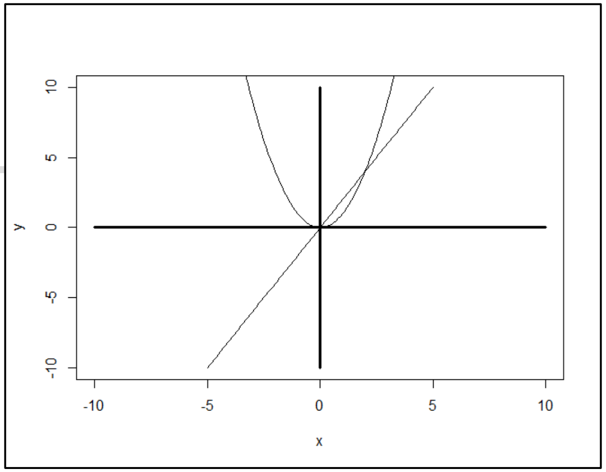

```
> dy_dx = function (w1) { 2*w1 }
> xStart = 20
> learningRate = 0.01
> maxLimit = 300
> xStartHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit )
+ {
+   xStartHistory[i] = xStart
+   xStart = xStart - learningRate * dy_dx(xStart)
+ }
> plot(xStartHistory,type='l')
> lines(c(-10,10),c(0,0),lwd=3,col='red')
> lines(c(maxLimit,maxLimit),c(0,0),lwd=3,col='red')
> lines(c(0,maxLimit),c(0,0),lwd=3,col='red')
```

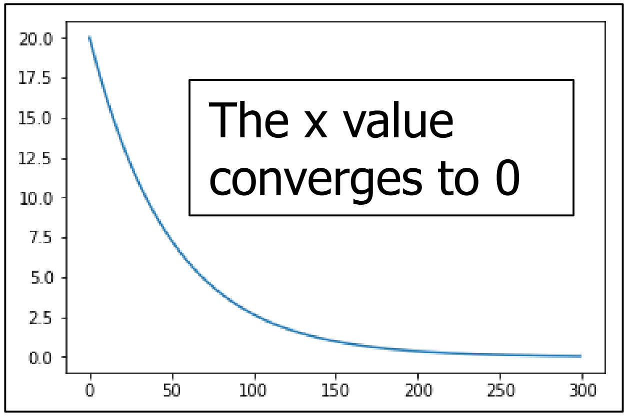


Example 1: Python Code

- $y = x^2$
- To find minimum point, equate the first derivative = 0
  - $\frac{dy}{dx} = 2x = 0$
  - $x = 0$
- Gradient Descent Algorithm
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial y}{\partial x} \|_{x^t}$

```
import numpy as np
import matplotlib.pyplot as plt

def dy_dx(w1):
    return(2*w1)

xStart = 20
learning_rate = 0.01
maxLimit = 300
xStartHistory = np.zeros(maxLimit)

for i in range(maxLimit):
    xStartHistory[i] = xStart
    xStart = xStart - learning_rate * dy_dx(xStart)

plt.plot(xStartHistory)
```

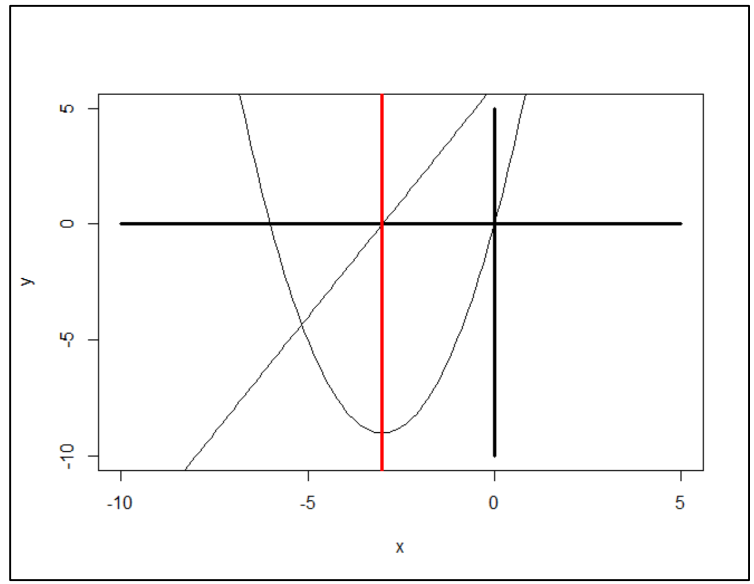




Example 2
Find the value of 'x' where the value of 'y' is minimum

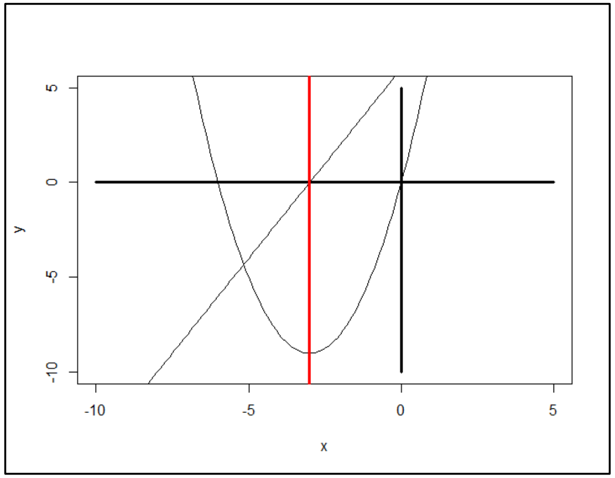
- $y = x^2 + 6x$
- To find minimum point, equate the first derivative = 0
  - $\frac{dy}{dx} = 2x + 6 = 0$
  - $x = -3$
- To find the value of 'x'
  - Where the value of 'y' is minimum
  - Correct answer: x = -3

```
> x = seq(-10,5,0.1)
> y = x^2 + 6*x
> dy_dx = function (w1) { 2*w1 + 6 }
> plot(x,y,type='l',xlim=c(-10,5),ylim=c(-10,5))
> lines(x,dy_dx(x))
> lines(c(0,0),c(-10,5),lwd=3)
> lines(c(-10,5),c(0,0),lwd=3)
> lines(c(-3,-3),c(-20,20),lwd=3,col='red')
```

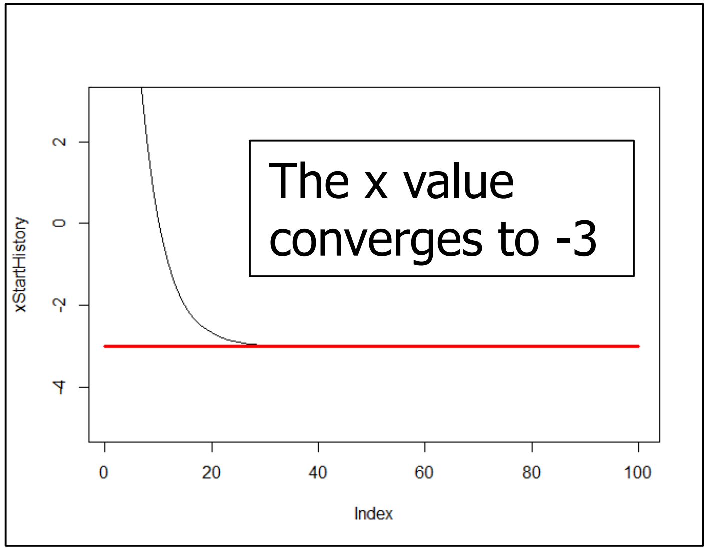


Example 2

- $y = x^2 + 6x$
- To find minimum point, equate the first derivative = 0
  - $\frac{dy}{dx} = 2x + 6 = 0$
  - $x = -3$
- Gradient Descent Algorithm
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial y}{\partial x} \|_{x^t}$




```
> dy_dx = function (w1) { 2*w1 + 6 }
> xStart = 20
> learningRate = 0.1
> maxLimit = 100
> xStartHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit )
+ {
+   xStartHistory[i] = xStart
+   xStart = xStart - learningRate * dy_dx(xStart)
+ }
> plot(xStartHistory,type='l',ylim=c(-5,3))
> lines(c(0,maxLimit),c(-3,-3),lwd=3,col='red')
```



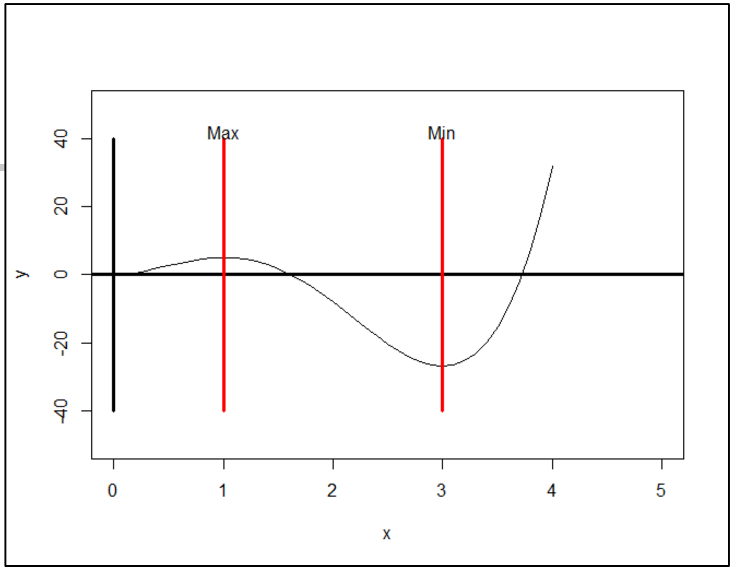
Gradient Descent Algorithm Parameters

- Parameters
  - Initial value of 'x'
  - Learning Rate
- If the choice of initial value of 'x' and learning rate is different
  - The Gradient Descent algorithm may not converge

Finding Minimum + Maximum of a Function Using Gradient Descent Algorithm

2 Variables (x,y) Function

Example 3
Find the value of 'x' where the value of 'y' is
* Minimum
* Maximum



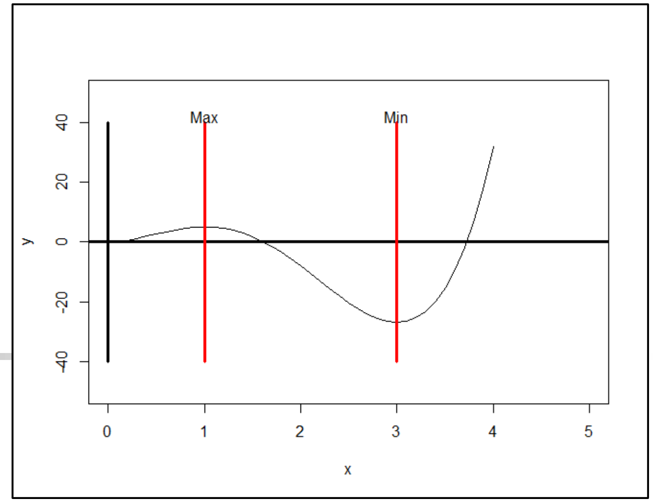
- $y = 3x^4 + 16x^3 + 18x^2$
- $\frac{dy}{dx} = 12x^3 + 48x^2 + 36x$
- To find the value of 'x'
  - Where the value of 'y' is local minimum
  - Correct answer: x = 3, y = -27

- To find the value of 'x'
  - Where the value of 'y' is local maximum
  - Correct answer: x = 1, y = 5

```
> x = seq(0,4,0.1)
> y = 3*x^4 - 16*x^3 + 18*x^2
> dy_dx = function (w1) { 12*w1^3 - 48*w1^2 + 36*w1 }
> plot(x,y,type='l',xlim=c(0,5),ylim=c(-50,50))
> #lines(x,dy_dx(x))
> lines(c(0,0),c(-40,40),lwd=3)
> lines(c(-10,10),c(0,0),lwd=3)
> lines(c(3,3),c(-40,40),lwd=3,col='red')
> lines(c(1,1),c(-40,40),lwd=3,col='red')
> text(1,42,"Max")
> text(3,42,"Min")
```

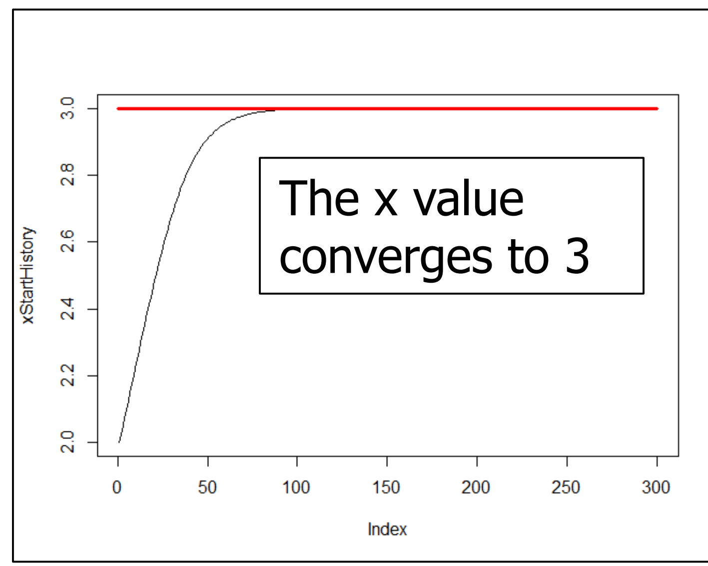
Example 3: Minimum

- $y = 3x^4 + 16x^3 + 18x^2$
- $\frac{dy}{dx} = 12x^3 + 48x^2 + 36x$
- Gradient Descent Algorithm: Minimum
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial y}{\partial x} \|_{x^t}$



- To find the value of 'x'
  - Where the value of 'y' is local minimum
  - Correct answer: x = 3, y = -27

```
> dy_dx = function (w1) { 12*w1^3 - 48*w1^2 + 36*w1 }
> xStart = 2
> learningRate = 0.001
> maxLimit = 300
> xStartHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit )
+ {
+   xStartHistory[i] = xStart
+   xStart = xStart - learningRate * dy_dx(xStart) #
For minimum 'subtract'
+   ## xStart = xStart + learningRate * dy_dx(xStart)
# maximum 'add'
+
+ }
> plot(xStartHistory,type='l')
> lines(c(0,maxLimit),c(3,3),lwd=3,col='red')
```



Minimum and Maximum

Minimum

- Algorithm: Minimum
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial y}{\partial x} \|_{x^t}$

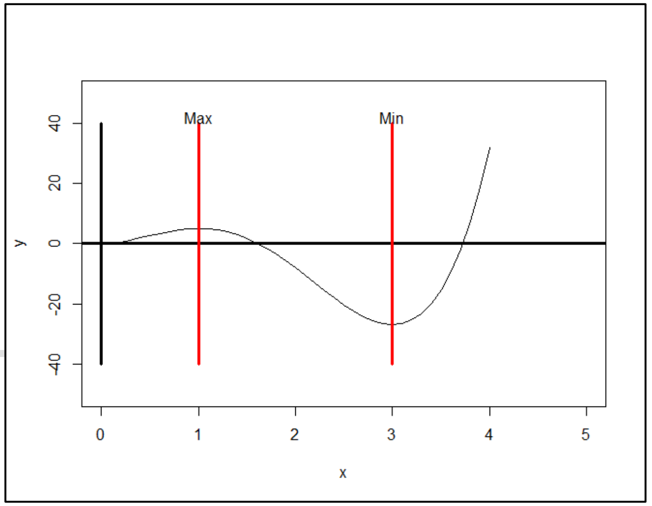
Maximum

- Algorithm: Maximum
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t + \eta \frac{\partial y}{\partial x} \|_{x^t}$

- Change value = $\eta \frac{\partial y}{\partial x} \|_{x^t}$
- The only difference between the algorithm of minimum and maximum is that instead of subtracting we add the 'change' value to the 'x' value in the loop.

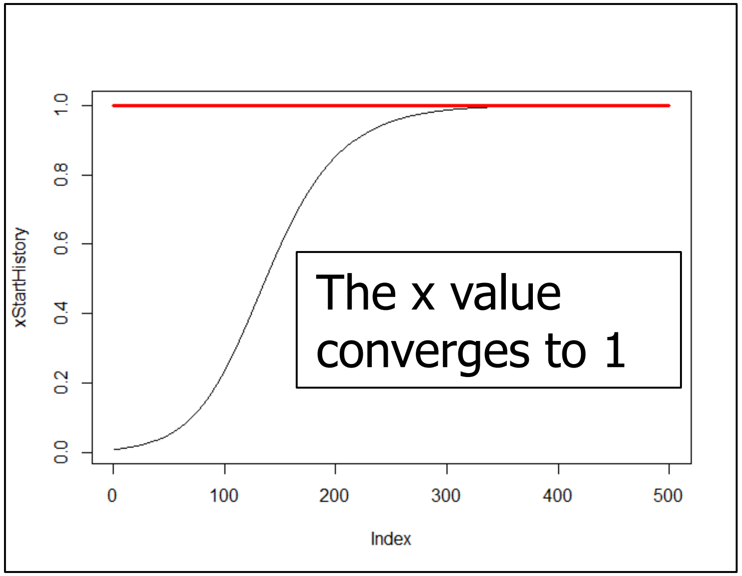
Example 3: Maximum

- $y = 3x^4 + 16x^3 + 18x^2$
- $\frac{dy}{dx} = 12x^3 + 48x^2 + 36x$
- Gradient Descent Algorithm: Maximum
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t + \eta \frac{\partial y}{\partial x} \|_{x^t}$



- To find the value of 'x'
  - Where the value of 'y' is local maximum
  - Correct answer: x = 1, y = 5

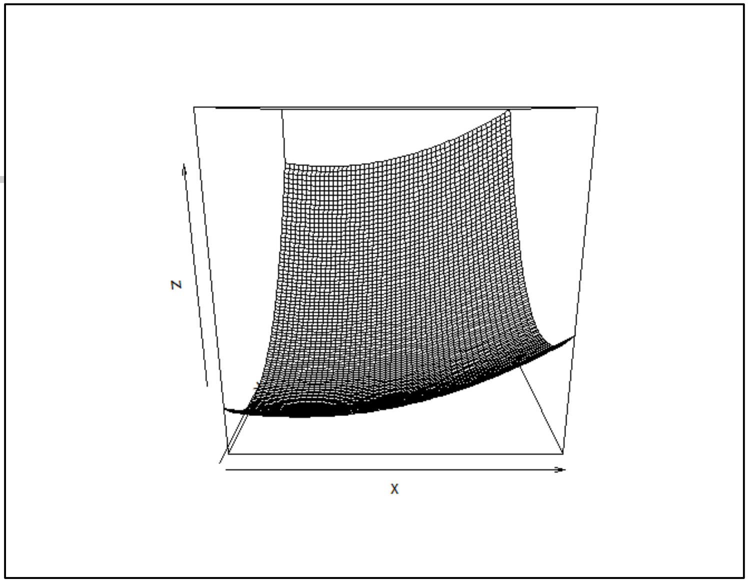
```
> dy_dx = function (w1) { 12*w1^3 - 48*w1^2 + 36*w1 }
> xStart = 0.01
> learningRate = 0.001
> maxLimit = 500
> xStartHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit )
+ {
+   xStartHistory[i] = xStart
+   ## xStart = xStart - learningRate * dy_dx(xStart)
# For minimum 'subtract'
+   xStart = xStart + learningRate * dy_dx(xStart)
# maximum 'add'
+
+ }
> plot(xStartHistory,type='l')
> lines(c(0,maxLimit),c(1,1),lwd=3,col='red')
```



Finding Minimum of a Function Using Gradient Descent Algorithm

3 Variables (x,y,z) Function

Example 4
Find the value of ‘x’ where the value of ‘y’ is
* Minimum

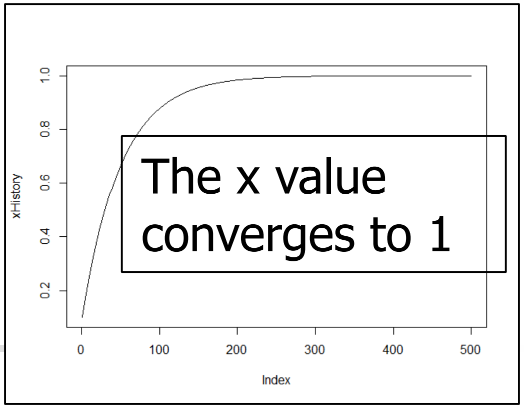


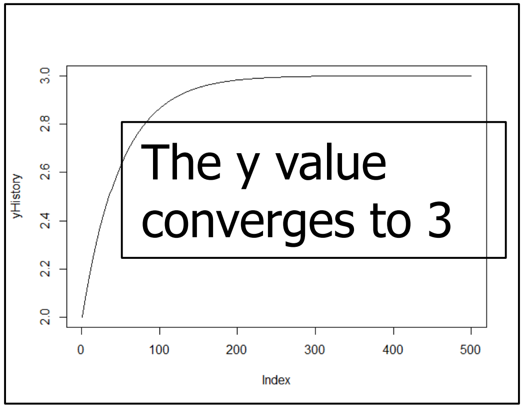
- $z = f(x,y) = x^2 + y^2 - 2x - 6y + 14$
- $\frac{\partial z}{\partial x} = 2x - 2$
- $\frac{\partial z}{\partial y} = 2y - 6$
- To find the value of ‘x’ and ‘y’
  - Where the value of ‘z’ is local minimum
  - Correct answer: x = 1, y = 3
  - $\frac{\partial z}{\partial x} = 2x - 2 = 0; x = 1$
  - $\frac{\partial z}{\partial y} = 2y - 6 = 0; y = 3$

```
> x = seq(0,5,0.1)
> y = seq(0,10,0.1)
> z = function (x,y) { x^2 + y^2 - 2*x - 6*y + 14 }
> dz_dx = function (x1,y1) { 2*x1 - 2 }
> dz_dy = function (x1,y1) { 2*y1 - 6 }
> z<-outer(x,y,z)
> persp(x, y, z)
> contour(z)
```

Example 4: Minimum

- $z = f(x,y) = x^2 + y^2 - 2x - 6y + 14$
- $\frac{\partial z}{\partial x} = 2x - 2; \quad \frac{\partial z}{\partial y} = 2y - 6$
- To find the value of 'x' and 'y'
  - Where the value of 'z' is local minimum
  - Correct answer: x = 1, y = 3
  - $\frac{\partial z}{\partial x} = 2x - 2 = 0; x = 1$
  - $\frac{\partial z}{\partial y} = 2y - 6 = 0; y = 3$






```
> dz_dx = function (x1,y1) { 2*x1 - 2 }
> dz_dy = function (x1,y1) { 2*y1 - 6 }
> xStart = 0.1;  yStart = 2
> learningRate = 0.01;  maxLimit = 500
> xHistory = yHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit)
+ {
+   xHistory[i] = xStart
+   yHistory[i] = yStart
+   dW = dz_dx(xStart,yStart)
+   db = dz_dy(xStart,yStart)
+
+   xStart = xStart - learningRate * dW
+   yStart = yStart - learningRate * db
+
+ }
> plot(xHistory,type='l')
> plot(yHistory,type='l')
```

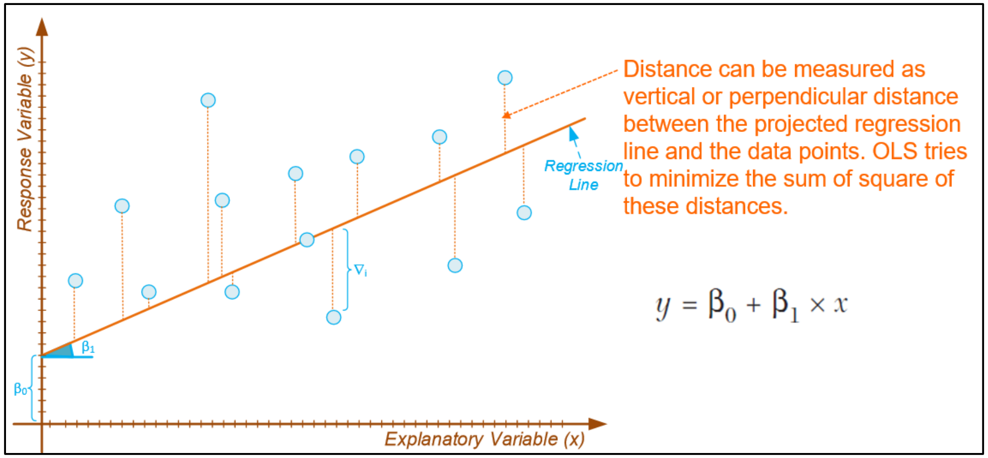
- Function z=f(x,y)
- Gradient Descent Algorithm: Minimum
  - Initialize the value of x and y
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial z}{\partial x} \|_{x^t, y^t}$
    - $y^{t+1} \leftarrow y^t - \eta \frac{\partial z}{\partial y} \|_{x^t, y^t}$

Solving Regression Problem Using Gradient Descent Algorithm

3 Variables (x,y,z) Function

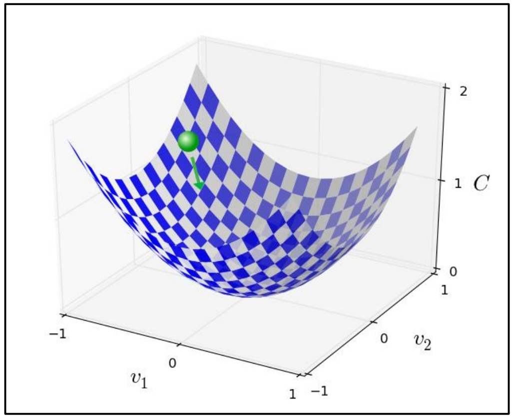
Computing the Regression Line Compute: Intercept and Slope

- Residual = Observed value – Computed Value
- Suppose regression equation is
  - $y = mx + b$
  - $y \text{ is the explanatory variable}$
  - $x \text{ is the pedictor variable}$
  - $m \text{ is the slope of the line}$
  - $b \text{ is the intercept}$
- $Residual = \; y_i - (mx_i + b)$
- $Residual^2 = \; (y_i - (mx_i + b))^2$
- $Residuals\ Sum\ of\ Squares = (RSS) = \sum_{i=1}^{N} (y_i - (mx_i + b))^2$



Residual Sum of Squares

- $Residuals\ Sum\ of\ Squares = (RSS) = \sum_{i=1}^{N}(y_i - (mx_i + b))^2$
- To find the minimum point of this function,
  - we will take the partial derivative of RSS with respect to 'm' and 'b' and set that to zero.
- The RSS is a convex function and it has a minimum point



Partial Derivatives of the RSS w.r.t. Intercept and Slope

- $Residuals\ Sum\ of\ Squares = (RSS) = \sum_{i=1}^{N}(y_i - (mx_i + b))^2$
- To find the minimum point of this function,
  - we will take the partial derivative of RSS with respect to 'm' and 'b' and set that to zero.

- $RSS = \sum_{i=1}^{N}(y_i - (mx_i + b))^2$
- $\frac{\partial RSS(m,b)}{\partial b} = \sum_{i=1}^{N} \frac{\partial}{\partial b}(y_i - (mx_i + b))^2$
- $\frac{\partial RSS(m,b)}{\partial b} = -2 \sum_{i=1}^{N}(y_i - (mx_i + b))$

- $RSS = \sum_{i=1}^{N}(y_i - (mx_i + b))^2$
- $\frac{\partial RSS(m,b)}{\partial m} = \sum_{i=1}^{N} \frac{\partial}{\partial m}(y_i - (mx_i + b))^2$
- $\frac{\partial RSS(m,b)}{\partial m} = -2 \sum_{i=1}^{N}(y_i - (mx_i + b))x_i$

$$\nabla RSS(b,m) = \begin{vmatrix} \frac{\partial RSS(m,b)}{\partial b} \\ \frac{\partial RSS(m,b)}{\partial m} \end{vmatrix} = \begin{vmatrix} -2\sum_{i=1}^{N}(y_i - (mx_i + b)) \\ -2\sum_{i=1}^{N}(y_i - (mx_i + b))x_i \end{vmatrix} = 0$$

Regression
Closed Form Solution

- To Compute 'm' and 'b'
  - SET GRADIENT = 0
- $\nabla \, RSS(b,m) = \begin{vmatrix} \frac{\partial RSS(m,b)}{\partial b} \\ \frac{\partial RSS(m,b)}{\partial m} \end{vmatrix} = \begin{vmatrix} -2\sum_{i=1}^{N}(y_i - (mx_i + b)) \\ -2\sum_{i=1}^{N}(y_i - (mx_i + b))x_i \end{vmatrix} = 0$
- ----------------------------------
- Top term
- $b = \left(\frac{\sum y_i}{N} - m\frac{\sum x_i}{N}\right) = \mu_y - m\mu_x$
- -------------------------------------
- Bottom term
- $m = \frac{\sum y_i x_i - \frac{\sum y_i \sum x_i}{N}}{\sum x_i^2 - \frac{\sum x_i \sum x_i}{N}} = r\frac{\sigma_y}{\sigma_x} = Correlation\,\frac{Std\ Dev\ of\ y}{Std\ Dev\ of\ x}$
- ---

Closed Form Solution

$$m = \frac{\sum y_i x_i - \frac{\sum y_i \sum x_i}{N}}{\sum x_i^2 - \frac{\sum x_i \sum x_i}{N}}$$

$$b = \left(\frac{\sum y_i}{N} - m\frac{\sum x_i}{N}\right)$$

	A	B	C	D	E	F	G
1							
2							
3		X	Y		X*Y		X^2
4		0	1		0		0
5		1	3		3		1
6		2	7		14		4
7		3	13		39		9
8		4	21		84		16
9							
10	SUM	10	45		140		30
11	AVERAGE	2	9		28		6
12	StdDev	1.58113883	8.124038405				
13	Correlation	0.97312368					
14							

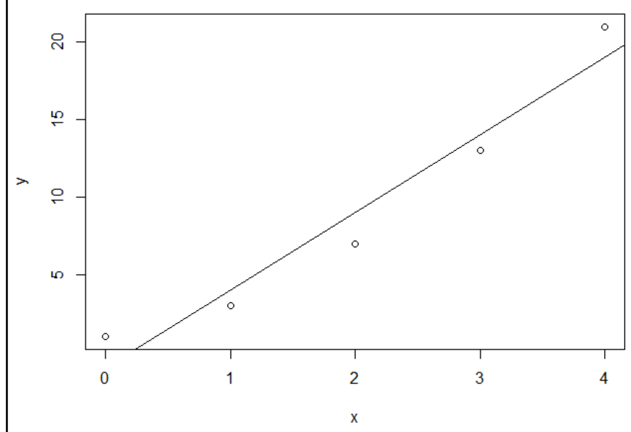
Regression Equation
$y = 5x - 1$

C16 =E10-(B10*C10/5)

	A	B	C	D	E	F	G
14							
15	Closed Form	Slope : Using SUM					
16		Numerator	50		(Sum of X*Y) - (1/N)*((Sum of X) * (Sum of Y))		
17		Denominator	10		(Sum of X^2) - (1/N)*((Sum of X * Sum of X))		
18		Slope	5				
19							
20		Intercept	-1		(Mean of Y) - slope * (Mean of X)		
21							

Regression
Using R 'lm' command

```
> x = c(0,1,2,3,4)
> y = c(1,3,7,13,21)
> plot(x,y)
> model = lm(y~x)
> summary(model)

Call:
lm(formula = y ~ x)

Residuals:
 1  2  3  4  5
 2 -1 -2 -1  2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.0000     1.6733  -0.598  0.59220
x             5.0000     0.6831   7.319  0.00527 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.16 on 3 degrees of freedom
Multiple R-squared:  0.947,  Adjusted R-squared:  0.9293
F-statistic: 53.57 on 1 and 3 DF,  p-value: 0.005268

> abline(model)
>
```



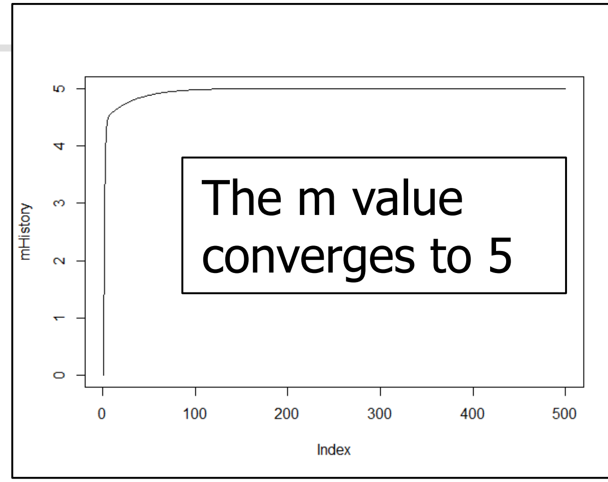
Regression Equation

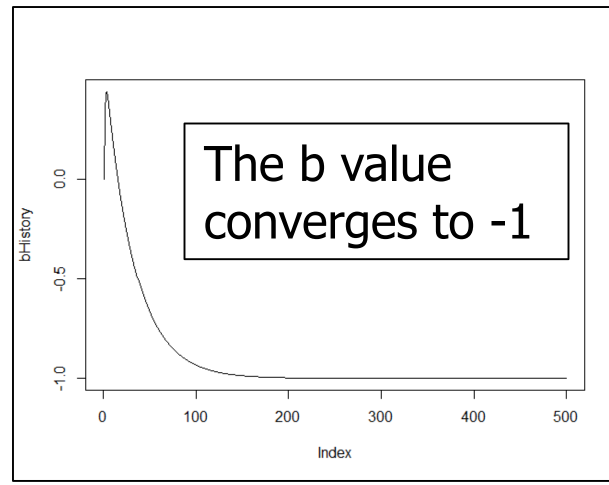
$y = 5x - 1$

Regression
Gradient Descent Algorithm Approach

Regression Equation
$y = 5x - 1$

```
> dRSS_dm = function (m,b) {-2*sum((y-m*x-b)*x)   }
> dRSS_db = function (m,b) { -2*sum(y-m*x-b) }
> mStart = bStart = 0
> learningRate = 0.01;  maxLimit = 500
> mHistory = bHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit )
+ {
+   mHistory[i] = mStart
+   bHistory[i] = bStart
+
+   dW = dRSS_dm(mStart,bStart)
+   db = dRSS_db(mStart,bStart)
+
+   mStart = mStart - learningRate * dW
+   bStart = bStart - learningRate * db
+ }
> plot(mHistory,type='l')
> plot(bHistory,type='l')
```



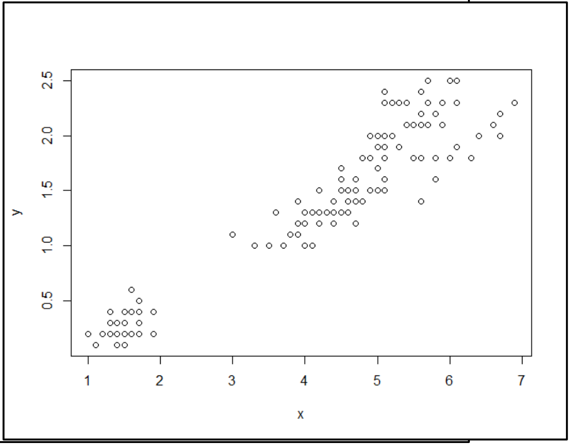


Solving Regression Problem Using Gradient Descent Algorithm

Iris Dataset

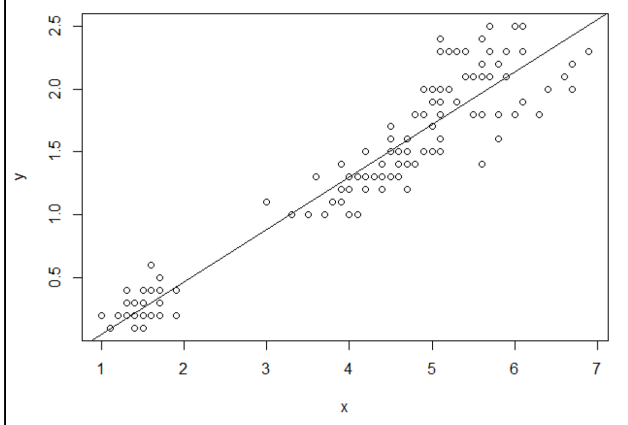
Read the Iris Dataset R Code

```
> data(iris)
> ########################
> dim(iris)
[1] 150   5
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> x = iris$Petal.Length
> y = iris$Petal.Width
> plot(x,y)
```



Regression: R
Using R ‘lm’ command

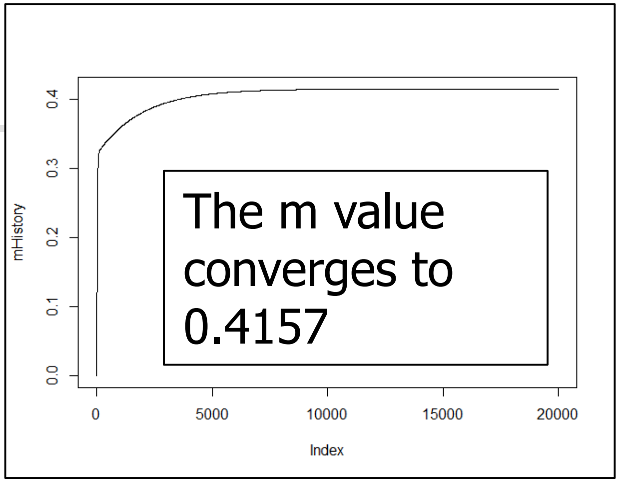
```
> model = lm(y~x)
> summary(model)

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.56515 -0.12358 -0.01898  0.13288  0.64272 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.363076   0.039762  -9.131  4.7e-16 ***
x            0.415755   0.009582  43.387  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2065 on 148 degrees of freedom
Multiple R-squared:  0.9271,  Adjusted R-squared:  0.9266 
F-statistic:  1882 on 1 and 148 DF,  p-value: < 2.2e-16

> abline(model)
```

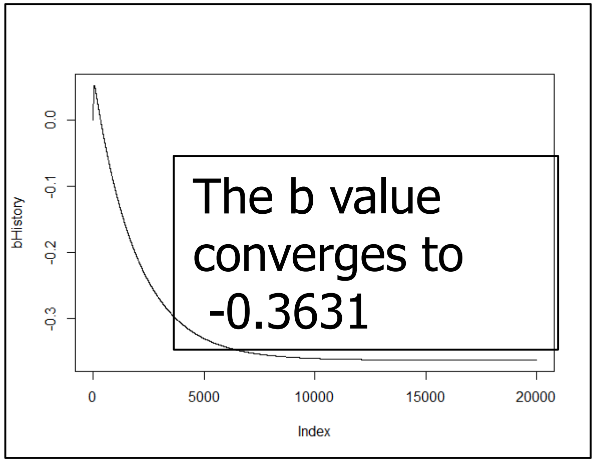


$$Regression\ Equation\colon R\colon\ Petal.Width = 0.4157 * Petal.Length - 0.3631$$

Regression: R
Gradient Descent Algorithm Approach

```
> dRSS_dm = function (m,b) {-2*sum((y-m*x-b)*x)   }
> dRSS_db = function (m,b) { -2*sum(y-m*x-b) }
> mStart = bStart = 0
> learningRate = 0.00001;   maxLimit = 20000
> mHistory = bHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit )
+ {
+   mHistory[i] = mStart
+   bHistory[i] = bStart
+
+   dW = dRSS_dm(mStart,bStart)
+   db = dRSS_db(mStart,bStart)
+
+   mStart = mStart - learningRate * dW
+   bStart = bStart - learningRate * db
+
+ }
> plot(mHistory,type='l')
> plot(bHistory,type='l')
> mHistory[maxLimit]
[1] 0.4157522
> bHistory[maxLimit]
[1] -0.3630608
```

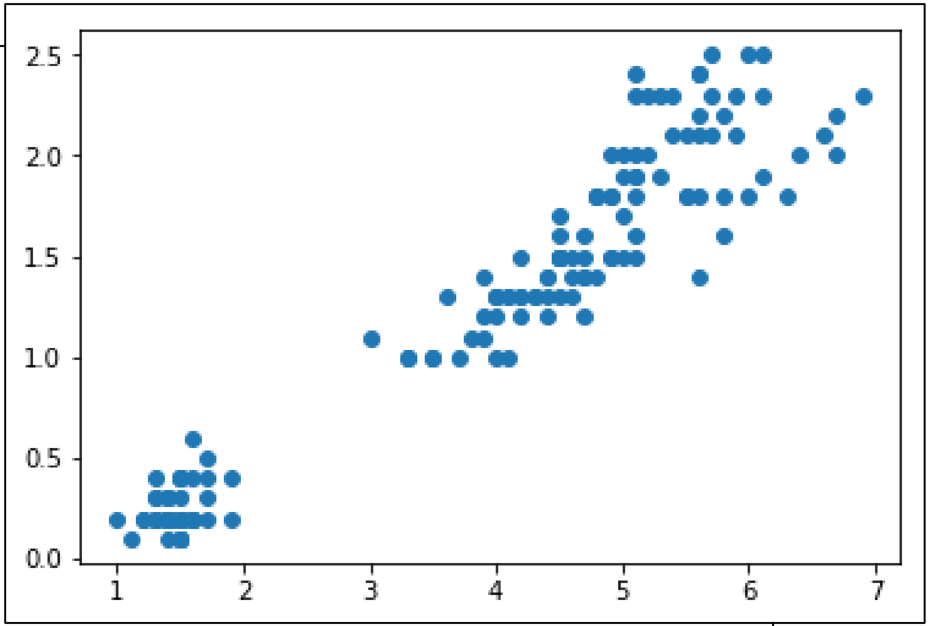




$$Regression\ Equation: R:\ Petal.Width = 0.4157 * Petal.Length - 0.3631$$
$$Regression\ Eq: R\ Grad\ Desc:\ Petal.Width = 0.4157 * Petal.Length - 0.3631$$

Read the Iris Dataset Python Code

```
###################################
# Load Libraries
#
from sklearn import linear_model
from sklearn import datasets
import matplotlib.pyplot as plt
####################################
# 2. Read the Dataset
#
iris   = datasets.load_iris()

features = iris["data"]

petalLength = features[:,2]
petalLength[0:5]
Out[13]: array([ 1.4,  1.4,  1.3,  1.5,  1.4])

petalWidth = features[:,3]
petalWidth[0:5]
Out[15]: array([ 0.2,  0.2,  0.2,  0.2,  0.2])

plt.plot(petalLength,petalWidth,'o')
Out[16]: [<matplotlib.lines.Line2D at 0x16b604b01d0>]
```



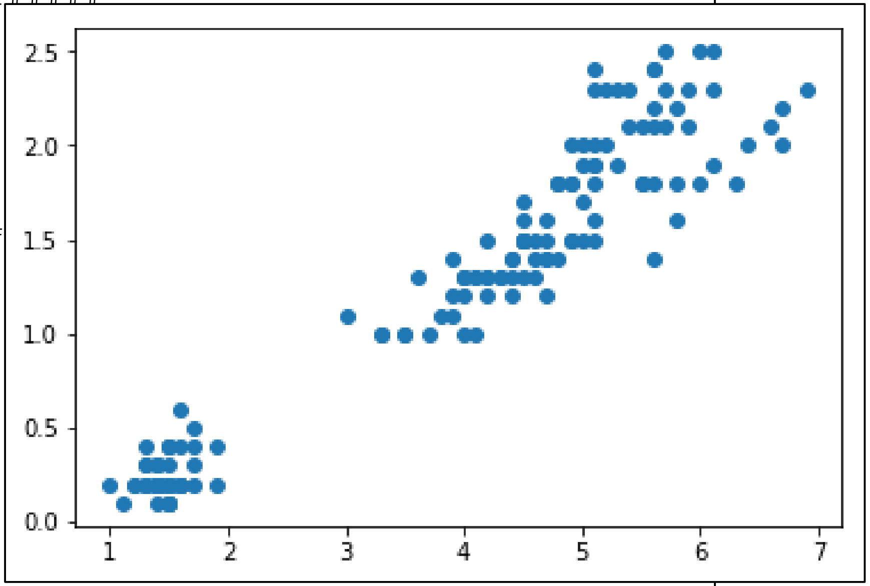
Regression
Using Python 'Scikit-Learn' Library

```
##############################################
# 3. Compute the regression equation using Scikit-Learn
#
petalLength = petalLength.reshape(-1,1)

petalWidth = petalWidth.reshape(-1,1)

linreg = linear_model.LinearRegression()

linreg.fit(petalLength, petalWidth)
Out[23]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1,
normalize=False)

print (linreg.intercept_)
[-0.36651405]

print (linreg.coef_)
[[ 0.41641913]]
```

$$Regression\ Equation: R:\ Petal.Width = 0.4157 * Petal.Length - 0.3631$$
$$Regression\ Eq: R\ Grad\ Desc:\ Petal.Width = 0.4157 * Petal.Length - 0.3631$$
$$Regression\ Equation: Scikit:\ Petal.Width = 0.4164 * Petal.Length - 0.3665$$

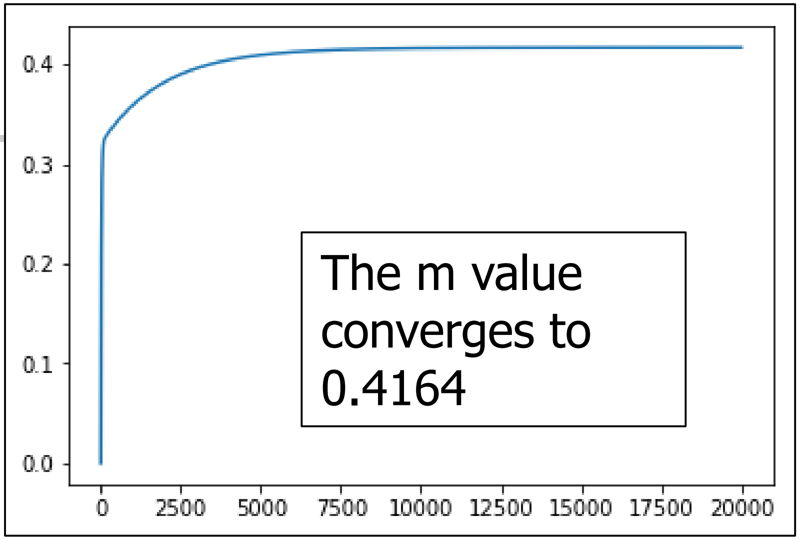
Regression: Python Gradient Descent Algorithm Approach

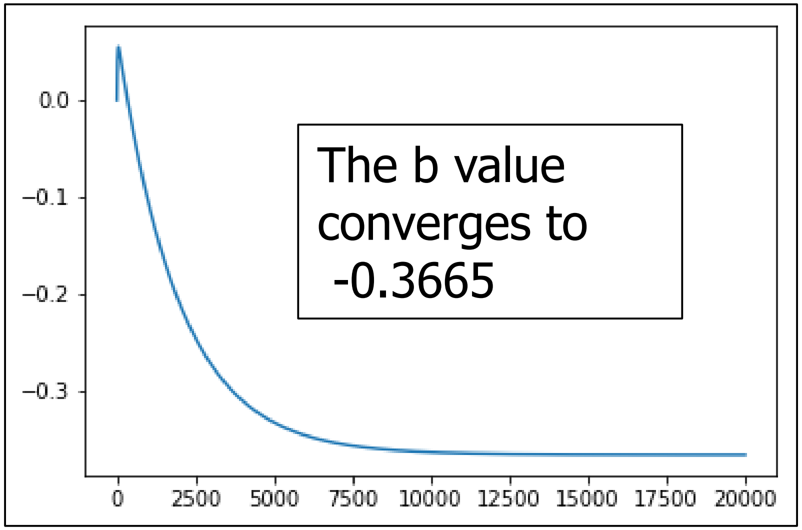
```
#####################################################
# 1. Load the libraries
#
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
#####################################################
# 2. Read the Dataset
#
iris    = datasets.load_iris()
features = iris["data"]

x = petalLength = features[:,2]
x[0:5]
Out[13]: array([ 1.4,  1.4,  1.3,  1.5,  1.4])

y = petalWidth = features[:,3]
y[0:5]
Out[15]: array([ 0.2,  0.2,  0.2,  0.2,  0.2])

plt.plot(x,y,'o')
```



Regression: Python
Gradient Descent Algorithm Approach

```
def dRSS_dm(m,b):
    return(-2*sum((y-m*x-b)*x))

def dRSS_db(m,b):
    return(-2*sum((y-m*x-b)))

mStart = 0
bStart = 0
learning_rate = 0.00001
maxLimit = 20000
mHistory = np.zeros(maxLimit)
bHistory = np.zeros(maxLimit)

for i in range(maxLimit):
    mHistory[i] = mStart
    bHistory[i] = bStart
    #print(mHistory[i], bHistory[i])

    dW = dRSS_dm(mStart,bStart)
    db = dRSS_db(mStart,bStart)

    mStart = mStart - learning_rate * dW
    bStart = bStart - learning_rate * db

print("mHistory=",mHistory[maxLimit-1])
mHistory= 0.416415833891

print("bHistory=",bHistory[maxLimit-1])
bHistory= -0.366499084269
```





Final Result

$$Regression\ Eq: R:\ Petal.Width = 0.4157 * Petal.Length - 0.3631$$
$$Regression\ Eq: R\ Grad\ Desc:\ Petal.Width = 0.4157 * Petal.Length - 0.3631$$

$$Regression\ Eq: Scikit:\ Petal.Width = 0.4164 * Petal.Length - 0.3665$$
$$Regression\ Eq: Python\ Grad\ Desc:\ Petal.Width = 0.4164 * Petal.Length - 0.3665$$

Lesson#5.2/Slide#31
Example#2: Linear Regression: Multi variables
Define the ‘cost’ and ‘optimization’ functions

```
#############################################
# 6. Define the 'cost' and 'optimization' Functions
# Initialize the variables
#

cost = tf.reduce_sum(tf.square(computed_y - y))

optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

init = tf.global_variables_initializer()
```

Other Optimization Algorithms

- Stochastic Gradient Descent
- Momentum
- Nesterov Momentum
- AdaGrad
- RMSProp
- Adam: Adaptive Moments

Other Optimization Algorithms

Neural Network Optimization Algorithms
A comparison study based on TensorFlow
Vadim Smolyakov
https://towardsdatascience.com/neural-network-optimization-algorithms-1a44c282f61dBy

====================================
Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent
Anish Singh Walia
https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f

Summary

- What is a Gradient?
- What is Gradient Descent Algorithm?
- Gradient Descent Algorithm
  - Minimum of a 2-Variable Function
  - Minimum & Maximum of a 2-Variable Function
  - Minimum of 3 Variable Function
  - Solving Regression problem
  - Solving Regression problem – Iris Dataset
    - R + Python