# **Gradient Descent - Optimization**

#**Minimize the function f(x) = 1.2 \* (x-2)^2 + 3.2**

#Basic calculus requires that we find the 1st derivative, and

#solve for the value of x such that f'(x) = 0.

#This is easy enough to do, f'(x) = 2\*1.2\*(x-2)

# step factor/learning rate = 0.05, initial x value= 0.1

#finds the values of x that minimize the function above,

#and plots the progress of the algorithm with each iteration.

**rm(list=ls())**

# create a sequence of elements in a Vector

#to generate sequences when plotting the axes of figures or simulating data.

**xs <- seq(0,4,len=20)**

**xs**

# define the function we want to optimize

**f <- function(x) {1.2 \* (x-2)^2 + 3.2}**

# plot the function

**plot(xs , f (xs), type="l",xlab="x",ylab=expression(1.2(x-2)^2 +3.2))**

# calculate the gradient df/dx

**grad <- function(x){**

**1.2\*2\*(x-2)**

**}**

# df/dx = 2.4(x-2), if x = 2 then 2.4(2-2) = 0

# The actual solution we will approximate with gradient descent

# is x = 2 as depicted in the plot below

# gradient descent implementation

**x <- 0.1** # initialize the first guess for x-value

**xtrace <- x** # store x -values for graphing purposes (initial)

**ftrace <- f(x)** # store y-values (function evaluated at x) for graphing purposes (initial)

**stepFactor <- 0.01** # learning rate 'alpha'

**for (step in 1:5000) {**

**x <- x - stepFactor\*grad(x)** # gradient descent update

**xtrace <- c(xtrace,x)** # update for graph

**ftrace <- c(ftrace,f(x))** # update for graph

**}**

**lines ( xtrace , ftrace , type="b",col="blue") # type=b (both points & lines)**

**text (0.5,6, "Gradient Descent",col="red",pos= 4)**

# print final value of x

**print(x)** # x converges to 2.0

**text(2,4,"x=2",col="red",pos=1)**

**text(2,4,"(Global minimum)",col="red",pos=3)**