pip install notebook

**# 1. Gradient Descent for cost function**

import numpy as npy

def grad\_desc(x,y):

m\_c = 1

b\_c = 1

iterations = 50

n = 2

l\_r = 0.001

for i in range(iterations):

y\_p = m\_c\*x + b\_c

cost = 1/n\*sum([val\*\*2 for val in (y-y\_p)])

mder = -(2/n)\*sum(x\*(y-y\_p))

bder = -(2/n)\*sum(y-y\_p)

m\_c = m\_c - l\_r\*mder

b\_c = b\_c - l\_r\*bder

print ( m\_c, b\_c, i, cost)

x = npy.array([1,2,3,4,5])

y = npy.array([10,15,20,25,30])

grad\_desc(x,y)

**2.#Gradient Descent 2\*x\*\*3 + 3\*y\*\*3**

2#Gradient Descent 2\*x\*\*3 + 3\*y\*\*3

import numpy as npy

def grad\_desc(x,y):

iterations = 10

l\_r = 0.01

a = 100

b = 100

for i in range(iterations):

xdr = 6\*x\*2

ydr = 2\*x\*3

a = a - l\_r\*xdr

b = b - l\_r\*ydr

z = npy.array([a + b])

print (z,i)

x = npy.array([npy.random.randint(100, size=(5))])

y = npy.array([npy.random.randint(100, size=(5))])

grad\_desc(x,y)

**3.#Gradient Descent Map**

import numpy as np

import matplotlib.pyplot as plt

a = np.linspace(-20.0, 5.0, 50)

b = np.linspace(-20.0, 20.0, 50)

a, b = np.meshgrid(a, b)

z = np.sqrt(np.square(a) + np.square(b))

cp = plt.contour(a, b, z, colors='black', linestyles='solid', linewidths=1)

plt.clabel(cp, inline=1, fontsize=10)

cp = plt.contourf(a, b, z)

plt.xlabel('a')

plt.ylabel('b')

plt.show()

**4.# Gradient Descent of Sphere Function 2\*x\*\*3 + 3\*y\*\*3**

import numpy as npy

def sph\_func(x,y):

iterations = 10

l\_r = 0.01

a = 10

b = 10

for i in range(iterations):

xdr = 6\*x\*2

ydr = 2\*x\*3

a = a - l\_r\*xdr

b = b - l\_r\*ydr

z = npy.array([a\*\*2 + b\*\*2])

print (z,i)

x = npy.array([1,2,3,4,5])

y = npy.array([10,15,20,25,30])

sph\_func(x,y)

**5.#Plotting Sphere Function**

import numpy as npy

import matplotlib.pyplot as plt

from matplotlib.pyplot import cm

from matplotlib.ticker import LinearLocator, FormatStrFormatter

from mpl\_toolkits.mplot3d import Axes3D

%matplotlib inline

#z = a\*\*2 + b\*\*2

fig = plt.figure()

ax = fig.gca(projection='3d')

ax.set\_title('z = a\*\*2 + b\*\*2')

ax.set\_xlabel('x-axis')

ax.set\_ylabel('y-axis')

ax.set\_zlabel('z-axis')

a = np.arange(-1, 1, 0.15)

b = np.arange(-1, 1, 0.15)

a,b = npy.meshgrid(a,b)

z = a\*\*2 + b\*\*2

surface = ax.plot\_surface(a, b, z, cmap = cm.summer, linewidth=0)

fig.colorbar(surface, shrink=0.5)

plt.show()

**6.#Gradient Descent of Rosenbrock Function 2\*x\*\*3 + 3\*y\*\*3**

import numpy as npy

def rosbn\_func(x,y):

iterations = 10

l\_r = 0.01

a = 0.5

b = 0.5

for i in range(iterations):

xdr = 6\*x\*2

ydr = 2\*x\*3

a = a - l\_r\*xdr

b = b - l\_r\*ydr

z = npy.array([100\*((b - a\*\*2)\*\*2) + (1-a)\*\*2])

print (z,i)

x = npy.array([1,2,3,4,5])

y = npy.array([10,15,20,25,30])

rosbn\_func(x,y)

**7.# Plotting of Rosenbrock function**

import numpy as npy

import matplotlib.pyplot as plt

from matplotlib.pyplot import cm

from matplotlib.ticker import LinearLocator, FormatStrFormatter

from mpl\_toolkits.mplot3d import Axes3D

%matplotlib inline

#'z = 100\*((b - a\*\*2)\*\*2) + (1-a)\*\*2'

fig = plt.figure()

ax = fig.gca(projection='3d')

ax.set\_title('z = 100\*((b - a\*\*2)\*\*2) + (1-a)\*\*2')

ax.set\_xlabel('x-axis')

ax.set\_ylabel('y-axis')

ax.set\_zlabel('z-axis')

a = np.arange(-1, 1, 0.15)

b = np.arange(-1, 1, 0.15)

a,b = npy.meshgrid(a,b)

z = 100\*((b - a\*\*2)\*\*2) + (1-a)\*\*2

surface = ax.plot\_surface(a, b, z, cmap = cm.spring, linewidth=0)

fig.colorbar(surface, shrink=0.5)

plt.show()

**8.#Gradient Descent of Six Hump Function 2\*x\*\*3 + 3\*y\*\*3**

import numpy as npy

def shump\_func(x,y):

iterations = 50

l\_r = 0.01

a = 1

b = 1

for i in range(iterations):

xdr = 6\*x\*2

ydr = 2\*x\*3

a = a - l\_r\*xdr

b = b - l\_r\*ydr

z = npy.array([(4 - 2.1\*a\*\*2 + a\*\*4/3)\*a\*\*2 + a\*b + (-4 + 4\*b\*\*2)\*b\*\*2])

print (z,i)

x = npy.arange(-5, 5, 0.25)

y = npy.arange(-5, 5, 0.25)

shump\_func(x,y)

**9.#Plotting Six Hump Function**

import numpy as npy

import matplotlib.pyplot as plt

from matplotlib.pyplot import cm

from matplotlib.ticker import LinearLocator, FormatStrFormatter

from mpl\_toolkits.mplot3d import Axes3D

%matplotlib inline

#'z = (4 - 2.1\*a\*\*2 + a\*\*4/3)\*a\*\*2 + a\*b + (-4 + 4\*b\*\*2)\*b\*\*2'

fig = plt.figure()

ax = fig.gca(projection='3d')

ax.set\_title('z = (4 - 2.1\*a\*\*2 + a\*\*4/3)\*a\*\*2 + a\*b + (-4 + 4\*b\*\*2)\*b\*\*2')

ax.set\_xlabel('x-axis')

ax.set\_ylabel('y-axis')

ax.set\_zlabel('z-axis')

a = np.arange(-1, 1, 0.15)

b = np.arange(-1, 1, 0.15)

a,b = npy.meshgrid(a,b)

z = (4 - 2.1\*a\*\*2 + a\*\*4/3)\*a\*\*2 + a\*b + (-4 + 4\*b\*\*2)\*b\*\*2

surface = ax.plot\_surface(a, b, z, cmap = cm.winter, linewidth=0)

fig.colorbar(surface, shrink=0.5)

plt.show()

**10.#Gradient Descent of Beale Function 2\*x\*\*3 + 3\*y\*\*3**

import numpy as npy

def be\_func(x,y):

iterations = 50

l\_r = 0.01

a = 1

b = 1

for i in range(iterations):

xdr = 6\*x\*2

ydr = 2\*x\*3

a = a - l\_r\*xdr

b = b - l\_r\*ydr

z = npy.array([(1.5 - a + a\*b)\*\*2 + (2.25 - a + a\*b\*\*2)\*\*2 + (2.65 - a + a\*b\*\*3)\*\*2])

print (z,i)

x = npy.array([npy.random.randint(100, size=(5))])

y = npy.array([npy.random.randint(100, size=(5))])

be\_func(x,y)

**11. #Plotting Beale Function**

import numpy as npy

import matplotlib.pyplot as plt

from matplotlib.pyplot import cm

from matplotlib.ticker import LinearLocator, FormatStrFormatter

from mpl\_toolkits.mplot3d import Axes3D

%matplotlib inline

#'z =(1.5 - a + a\*b)\*\*2 + (2.25 - a + a\*b\*\*2)\*\*2 + (2.65 - a + a\*b\*\*3)\*\*2'

fig = plt.figure()

ax = fig.gca(projection='3d')

ax.set\_title('z = (1.5 - a + a\*b)\*\*2 + (2.25 - a + a\*b\*\*2)\*\*2 + (2.65 - a + a\*b\*\*3)\*\*2')

ax.set\_xlabel('x-axis')

ax.set\_ylabel('y-axis')

ax.set\_zlabel('z-axis')

a = np.arange(-1, 1, 0.15)

b = np.arange(-1, 1, 0.15)

a,b = npy.meshgrid(a,b)

z = (1.5 - a + a\*b)\*\*2 + (2.25 - a + a\*b\*\*2)\*\*2 + (2.65 - a + a\*b\*\*3)\*\*2

surface = ax.plot\_surface(a, b, z, cmap = cm.autumn, linewidth=0)

fig.colorbar(surface, shrink=0.5)

plt.show()