Thomas Whitcomb
thomaswhitcomb@yahoo.com
Feb 22nd, 2019

Lesson 6

Code

```
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn import linear_model
import numpy as np
import sys

def dz_dx(x,y):
    return (2*(x-2)) / ((25-(x-2)**2) - ((y - 3)**2))

def dz_dy(x,y):
    return (2*(y-3)) / ((25-(x-2)**2) - ((y - 3)**2))

def problem1():
    x_start = 0
    y_start = 0
    learning_rate = 0.01
    max_iterations = 10000
    epsilon = 0.000001
    history=[(x_start,y_start)]
    for i in range(max_iterations-1):
        dW = dz_dx(x_start,y_start)
        db = dz_dy(x_start,y_start)
        x_start1 = x_start - (learning_rate *dW)
        y_start1 = y_start - (learning_rate *db)
        if abs(x_start1 - x_start) <= epsilon and abs(y_start1 - y_start) <=
epsilon:
            return i+1,x_start1, y_start1
        x_start = x_start1
        y_start = y_start1
    return -1,None,None
```

```
class Problem2():
    def __init__(self):
      pass
    def create_dataset(self):
        RANDOM_SEED = 42
        tf.set_random_seed(RANDOM_SEED)
        n_samples = 30
        self.train_x = np.linspace(0,20,n_samples)
        self.train_y = 3.7 * self.train_x + 14 + 4 *
np.random.randn(n_samples)
        plt.plot(self.train_x,self.train_y)

    def sk_solution(self):
        x = np.reshape(self.train_x,(len(self.train_x),1))
        linreg = linear_model.LinearRegression()
        linreg.fit(x,self.train_y)
        return linreg.coef_,linreg.intercept_

    def dg_solution(self,m_target,b_target,learning_rate,epsilon):
        def dRSS_dm(m,b):
            return(-2*sum((self.train_y-m*self.train_x-b)*self.train_x))
        def dRSS_db(m,b):
            return(-2*sum((self.train_y-m*self.train_x-b)))
        m_start = 0
        b_start = 0
        iterations = 2500000
        history = [(m_start,b_start)]
        for i in range(iterations):
            dW = dRSS_dm(m_start,b_start)
            db = dRSS_db(m_start,b_start)
            m_start1 = m_start - (learning_rate * dW)
            b_start1 = b_start - (learning_rate * db)
            if abs(m_start1 - m_target) <= epsilon and abs(b_start1 -
b_target) <= epsilon:
                return i,m_start1,b_start1
            m_start = m_start1
            b_start = b_start1
        return -1,None,None


def main():
    print("Problem 1")
    print("=========")
    iterations,slope,intercept = problem1()
    print("  ",iterations,"iterations to reach a slope =",slope,"and intercept
=",intercept)
```

```
    print("Problem 2")
    print("=========")
    problem2 = Problem2()
    problem2.create_dataset()
    m_target,b_target = problem2.sk_solution()
    print("   Scikit-learn slope =",m_target[0],"and intercept =",b_target)
    learning_rate = 0.0001
    iterations,slope,intercept =
problem2.dg_solution(m_target,b_target,learning_rate,0.0000000001)
    print("  ",iterations,"DG iterations with",learning_rate,"learning rate to
reach  m =",slope,"and b =",intercept)
    learning_rate = 0.00001
    iterations,slope,intercept =
problem2.dg_solution(m_target,b_target,learning_rate,0.0000000001)
    print("  ",iterations,"DG iterations with",learning_rate,"learning rate to
reach  m =",slope,"and b =",intercept)
    learning_rate = 0.000001
    iterations,slope,intercept =
problem2.dg_solution(m_target,b_target,learning_rate,0.0000000001)
    print("  ",iterations,"DG iterations with",learning_rate,"learning rate to
reach  m =",slope,"and b =",intercept)


if __name__ == "__main__":
    main()
```

Results

Run > python lesson6.py

```
Problem 1
=========
   9401 iterations to reach a slope = 1.9991674863651752 and intercept =
2.998751229547764
Problem 2
=========
   Scikit-learn slope = 3.639500001678395 and intercept = 15.143902251638679
   16392 DG iterations with 0.000100 learning rate to reach  m =
3.6395000016857826 and b = 15.143902251538693
   164052 DG iterations with 0.000010 learning rate to reach  m =
3.639500001685782 and b = 15.14390225153869
   1645091 DG iterations with 0.000001 learning rate to reach  m =
3.6395000016857963 and b = 15.14390225153868
```