Thomas Whitcomb
thomaswhitcomb@yahoo.com
Feb 17th, 2019

Lesson 5

Code

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

def scale(t):
    tMin = t.min(axis=0)
    tMax = t.max(axis=0)
    return (t-tMin)/(tMax-tMin)

class Problem1Base():
    def __init__(self):
        tf.reset_default_graph()

    def create_dataset(self):
        n_samples = 30
        self.x_point = np.linspace(0,20,n_samples)

        self.y_point = 3.7 * self.x_point + 14 + 4 *
np.random.randn(n_samples)

        plt.plot(self.x_point, self.y_point,'o')
        plt.savefig("lesson5.png")

class Problem1SK(Problem1Base):
    def __init__(self):
        self.RANDOM_SEED = 42
        np.random.seed(self.RANDOM_SEED)

    def compute_regression(self):
        linreg = linear_model.LinearRegression()
        self.x_point = np.reshape(self.x_point,(len(self.x_point),1))
        linreg.fit(self.x_point,self.y_point,self.y_point)
        return linreg.coef_,linreg.intercept_

class Problem1TF(Problem1Base):
    def __init__(self):
        self.graph = tf.Graph()
        self.RANDOM_SEED = 42
        tf.set_random_seed(self.RANDOM_SEED)
        np.random.seed(self.RANDOM_SEED)

    def build(self):
        self.X = tf.placeholder(tf.float32)
        self.Y = tf.placeholder(tf.float32)
        self.LR = tf.placeholder(tf.float32)
        self.slope = tf.Variable(tf.random_uniform([1],-1.0,1.0))
        self.intercept = tf.Variable(tf.zeros([1]))
        self.response = self.slope*self.X + self.intercept
        self.cost = tf.reduce_mean(tf.square(self.response - self.Y))
        self.optimizer =
tf.train.GradientDescentOptimizer(self.LR).minimize(self.cost)

    def compute_regression(self,epochs,learning_rate):
        init = tf.global_variables_initializer()
        with tf.Session() as session:
            session.run(init)
            for epoch in range(epochs):
                session.run(
                        self.optimizer,feed_dict = {
                            self.LR: learning_rate,
                            self.X : self.x_point,
                            self.Y : self.y_point})
                if (epoch % (epochs/20)) == 0:
                    c = session.run(
                            self.cost, feed_dict = {
                                self.LR: learning_rate,
                                self.X : self.x_point,
                                self.Y : self.y_point})
                    print("Epoch:", epoch , "Cost:", c, "Slope:",
                            session.run(self.slope),
                            "Intercept:",
                            session.run(self.intercept))
            c = session.run(self.cost, feed_dict = {
                self.LR: learning_rate,
                self.X : self.x_point,
                self.Y : self.y_point})
            print("Epoch:", epoch , "Cost:", c, "Slope:",
                    session.run(self.slope),
                    "Intercept:",
                    session.run(self.intercept))
            return c,session.run(self.slope),session.run(self.intercept)

class Problem2Base():
    def __init__(self):
        tf.reset_default_graph()

    def create_dataset(self):
        f = open("00 kc_house_data.csv")
        f.readline()
        dataset = np.genfromtxt(fname = f, delimiter = ',',usecols=(2,3,5))
        self.predictors = dataset[:,1:3]
        predictorsMin = self.predictors.min(axis=0)
        predictorsMax = self.predictors.max(axis=0)
        self.predictors_scaled = scale(self.predictors)
        self.response = dataset[:,0:1]
        responseMin = self.response.min(axis=0)
        responseMax = self.response.max(axis=0)
        self.response_scaled = scale(self.response)

class Problem2SK(Problem2Base):
    def __init__(self):
        self.RANDOM_SEED = 42
        np.random.seed(self.RANDOM_SEED)

    def compute_regression(self):
        linreg = linear_model.LinearRegression()
        linreg.fit(self.predictors_scaled,self.response_scaled)
        return linreg.coef_,linreg.intercept_

class Problem2TF(Problem2Base):
    def __init__(self):
        self.RANDOM_SEED = 42
        tf.set_random_seed(self.RANDOM_SEED)
        np.random.seed(self.RANDOM_SEED)

    def build(self):
        self.X1 = tf.placeholder(tf.float32)
        self.X2 = tf.placeholder(tf.float32)
        self.Y = tf.placeholder(tf.float32)

        self.LR = tf.placeholder(tf.float32)

        self.slope1 = tf.Variable([0],dtype=tf.float32,name="weight1")
        self.slope2 = tf.Variable([0],dtype=tf.float32,name="weight2")
        self.intercept = 
tf.Variable(tf.zeros([1]),dtype=tf.float32,name="bias")
        self.response = self.slope1*self.X1 + self.slope2*self.X2 + 
self.intercept
        self.cost = tf.reduce_mean(tf.square(self.response - self.Y))
        self.optimizer = 
tf.train.GradientDescentOptimizer(self.LR).minimize(self.cost)

    def compute_regression(self,epochs,learning_rate):
        init = tf.global_variables_initializer()
        with tf.Session() as session:
            session.run(init)
            for epoch in range(epochs):
                session.run(
                        self.optimizer,feed_dict = {
                            self.LR: learning_rate,
                            self.X1:self.predictors_scaled[:,0:1],
                            self.X2:self.predictors_scaled[:,1:2],
                            self.Y:self.response_scaled})
                if not epoch % 100000:
                    c = session.run(
                            self.cost, feed_dict = {
                                self.LR:learning_rate,
                                self.X1:self.predictors_scaled[:,0:1],
                                self.X2:self.predictors_scaled[:,1:2],
                                self.Y:self.response_scaled})
                    print("Epoch:", epoch , "Cost:", c, "Slope1:",
                            session.run(self.slope1),
                            "Slope2:",
                            session.run(self.slope2),
                            "Intercept:",
                            session.run(self.intercept))
            c = session.run(
                    self.cost, feed_dict = {
                        self.LR: learning_rate,
                        self.X1 : self.predictors_scaled[:,0:1],
                        self.X2 : self.predictors_scaled[:,1:2],
                        self.Y : self.response_scaled} )
            print("Epoch:", epoch , "Cost:", c, "Slope1:",
                    session.run(self.slope1),
                    "Slope2:",
                    session.run(self.slope2),
                    "Intercept:",
                    session.run(self.intercept))

            return 
c,session.run(self.slope1),session.run(self.slope2),session.run(self.intercept
)
```

```
def main():

    print("Problem 1 - SKLearn")
    print("===================")
    problem1 = Problem1SK()
    problem1.create_dataset()
    slope,intercept = problem1.compute_regression()
    print(slope,intercept)
    assert ("%.8f" % slope[0]) == "3.54942311"
    assert ("%.15f" % intercept) == "14.841075232789862"

    print("\nProblem 1, solution 1 - Tensorflow")
    print("==================================")
    print("   Epoch 50000, Learning rate 0.0001")
    problem1 = Problem1TF()
    problem1.create_dataset()
    problem1.build()
    cost,slope,intercept = problem1.compute_regression(50000,0.0001)
    print(cost,slope,intercept)
    assert ("%.5f" % cost) == "11.36642"
    assert ("%.7f" % slope[0]) == "3.5775077"
    assert ("%.6f" % intercept[0]) == "14.184714"

    print("\nProblem 1, solution 2 - Tensorflow")
    print("==================================")
    print("   Epoch 5000, Learning rate 0.001")
    problem1 = Problem1TF()
    problem1.create_dataset()
    problem1.build()
    cost,slope,intercept = problem1.compute_regression(5000,0.001)
    print(cost,slope,intercept)
    assert ("%.6f" % cost) == "11.365601"
    assert ("%.7f" % slope[0]) == "3.5774026"
    assert ("%.6f" % intercept[0]) == "14.186133"

    print("Problem 2 - SKLearn")
    print("===================")
    problem2 = Problem2SK()
    problem2.create_dataset()
    slope,intercept = problem2.compute_regression()
    print(slope,intercept)

    print("Problem 2 - Tensorflow")
    print("=====================")
    print("   Epoch 1000000, Learning rate 0.01")
    problem2 = Problem2TF()
    problem2.create_dataset()
    problem2.build()
    cost,slope1,slope2,intercept = problem2.compute_regression(1000000,0.01)

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

Results

Run > python lesson5.py

```
Problem 1 - SKLearn
====================
[3.54942311] 14.841075232789862

Problem 1, solution 1 - Tensorflow
====================================
   Epoch 50000, Learning rate 0.0001

Epoch: 0 Cost: 1834.8353 Slope: [1.0177796] Intercept: [0.00821498]
Epoch: 2500 Cost: 56.641254 Slope: [4.4693475] Intercept: [2.1118104]
Epoch: 5000 Cost: 46.157352 Slope: [4.3501606] Intercept: [3.7252202]
Epoch: 7500 Cost: 38.084175 Slope: [4.2455807] Intercept: [5.141031]
Epoch: 10000 Cost: 31.867392 Slope: [4.153803] Intercept: [6.3834424]
Epoch: 12500 Cost: 27.080126 Slope: [4.073262] Intercept: [7.4736905]
Epoch: 15000 Cost: 23.39365 Slope: [4.0025835] Intercept: [8.430417]
Epoch: 17500 Cost: 20.554863 Slope: [3.9405656] Intercept: [9.269972]
Epoch: 20000 Cost: 18.368835 Slope: [3.8861418] Intercept: [10.006703]
Epoch: 22500 Cost: 16.68548 Slope: [3.838387] Intercept: [10.653204]
Epoch: 25000 Cost: 15.3892 Slope: [3.7964773] Intercept: [11.220526]
Epoch: 27500 Cost: 14.390979 Slope: [3.7596998] Intercept: [11.718375]
Epoch: 30000 Cost: 13.622302 Slope: [3.7274263] Intercept: [12.155245]
Epoch: 32500 Cost: 13.03038 Slope: [3.699107] Intercept: [12.53861]
Epoch: 35000 Cost: 12.574581 Slope: [3.6742594] Intercept: [12.8750105]
Epoch: 37500 Cost: 12.223575 Slope: [3.652451] Intercept: [13.170223]
Epoch: 40000 Cost: 11.953281 Slope: [3.6333156] Intercept: [13.429284]
Epoch: 42500 Cost: 11.745135 Slope: [3.61652] Intercept: [13.656612]
Epoch: 45000 Cost: 11.584855 Slope: [3.601785] Intercept: [13.856103]
Epoch: 47500 Cost: 11.461434 Slope: [3.5888524] Intercept: [14.031155]
Epoch: 49999 Cost: 11.36642 Slope: [3.5775077] Intercept: [14.184714]
11.36642 [3.5775077] [14.184714]

Problem 1, solution 2 - Tensorflow
====================================
   Epoch 5000, Learning rate 0.001
```

```
Epoch: 0 Cost: 1124.0109 Slope: [1.8315809] Intercept: [0.08464605]
Epoch: 250 Cost: 56.53987 Slope: [4.4682636] Intercept: [2.126466]
Epoch: 500 Cost: 46.077038 Slope: [4.3491845] Intercept: [3.7384503]
Epoch: 750 Cost: 38.02059 Slope: [4.2446923] Intercept: [5.152967]
Epoch: 1000 Cost: 31.817055 Slope: [4.153001] Intercept: [6.3942065]
Epoch: 1250 Cost: 27.040306 Slope: [4.072541] Intercept: [7.483397]
Epoch: 1500 Cost: 23.36218 Slope: [4.001938] Intercept: [8.439158]
Epoch: 1750 Cost: 20.529995 Slope: [3.9399834] Intercept: [9.277843]
Epoch: 2000 Cost: 18.349218 Slope: [3.8856187] Intercept: [10.013782]
Epoch: 2250 Cost: 16.669996 Slope: [3.8379138] Intercept: [10.65957]
Epoch: 2500 Cost: 15.376987 Slope: [3.7960525] Intercept: [11.226247]
Epoch: 2750 Cost: 14.381365 Slope: [3.7593193] Intercept: [11.723507]
Epoch: 3000 Cost: 13.6147175 Slope: [3.7270856] Intercept: [12.15986]
Epoch: 3250 Cost: 13.024407 Slope: [3.6988008] Intercept: [12.542752]
Epoch: 3500 Cost: 12.569859 Slope: [3.673981] Intercept: [12.878743]
Epoch: 3750 Cost: 12.21986 Slope: [3.6522014] Intercept: [13.173572]
Epoch: 4000 Cost: 11.950358 Slope: [3.6330903] Intercept: [13.432283]
Epoch: 4250 Cost: 11.742842 Slope: [3.6163201] Intercept: [13.659301]
Epoch: 4500 Cost: 11.583048 Slope: [3.6016045] Intercept: [13.858509]
Epoch: 4750 Cost: 11.460006 Slope: [3.5886915] Intercept: [14.033316]
Epoch: 4999 Cost: 11.365601 Slope: [3.5774026] Intercept: [14.186133]
11.365601 [3.5774026] [14.186133]

Problem 2 - SKLearn
====================
[[-0.24697745  0.54555018]] [0.01252649]

Problem 2 - Tensorflow
======================
   Epoch 1000000, Learning rate 0.01

Epoch: 0 Cost: 0.005885043 Slope1: [0.00013298] Slope2: [0.00021165]
Intercept: [0.0012199]
Epoch: 100000 Cost: 0.0011529328 Slope1: [-0.1124965] Slope2: [0.5104075]
Intercept: [0.00353261]
Epoch: 200000 Cost: 0.0011446119 Slope1: [-0.19710143] Slope2: [0.5323637]
Intercept: [0.00921143]
Epoch: 300000 Cost: 0.001143469 Slope1: [-0.22829913] Slope2: [0.5405776]
Intercept: [0.0112898]
Epoch: 400000 Cost: 0.0011433085 Slope1: [-0.2400288] Slope2: [0.5436168]
Intercept: [0.01207768]
Epoch: 500000 Cost: 0.0011432866 Slope1: [-0.24415138] Slope2: [0.5445807]
Intercept: [0.01236867]
Epoch: 600000 Cost: 0.0011432837 Slope1: [-0.24561287] Slope2: [0.5449208]
Intercept: [0.0124721]
Epoch: 700000 Cost: 0.0011432837 Slope1: [-0.24561287] Slope2: [0.5449208]
Intercept: [0.0124721]
```

```
Epoch: 800000 Cost: 0.0011432837 Slope1: [-0.24561287] Slope2: [0.5449208]
Intercept: [0.0124721]
Epoch: 900000 Cost: 0.0011432837 Slope1: [-0.24561287] Slope2: [0.5449208]
Intercept: [0.0124721]
Epoch: 999999 Cost: 0.0011432837 Slope1: [-0.24561287] Slope2: [0.5449208]
Intercept: [0.0124721]
```