# Vanilla GD in python

February 24, 2019

```
In [1]: import numpy as np

In [5]: my_array = np.random.random((6,3))

In [7]: my_array

Out[7]: array([[0.29830317, 0.19431556, 0.98847037],
               [0.67910816, 0.98361318, 0.77716166],
               [0.78833642, 0.66386854, 0.56767822],
               [0.59500352, 0.54201272, 0.21982629],
               [0.7673696 , 0.29115337, 0.25746824],
               [0.0691452 , 0.07161926, 0.22672101]])

In [8]: y = my_array[:, -1] # for last column
        X = my_array[:, :-1] #for all except last column

In [12]: alpha = 0.01
         theta = np.random.random((2,))
         num_iters = 1

In [14]: theta

Out[14]: array([0.25519809, 0.34139396])

In [82]: weights = []

In [83]: def gradientDescent(X, y, theta, alpha, num_iters):
             """
                Performs gradient descent to learn theta
             """
             m = y.size  # number of training examples
             for i in range(num_iters):
                 y_hat = np.dot(X, theta)
                 theta = theta - alpha * np.dot(X.T, y_hat-y)
                 weights.append(theta)
             return weights

In [84]: num_iters = 10
```

```
In [85]: z = gradientDescent(X, y, theta, alpha, num_iters)

In [86]: z

Out[86]: [array([0.25978606, 0.34599078]),
          array([0.26419405, 0.3504209 ]),
          array([0.26842888, 0.35469062]),
          array([0.27249709, 0.35880598]),
          array([0.276405  , 0.36277281]),
          array([0.28015867, 0.36659671]),
          array([0.28376396, 0.37028307]),
          array([0.28722649, 0.37383709]),
          array([0.29055165, 0.37726375]),
          array([0.29374465, 0.38056785])]

In [ ]:
```