Homework 0

Alex Shah

9/7/16

1 Python Proof 2.1

1.1 Python Installation Proof and Output

```
import sys
import numpy
import scipy
import sklearn
import matplotlib
import pandas
import tensorflow

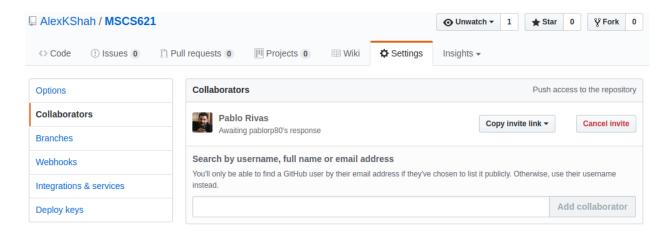
print(sys.version)
print(numpy.__version__)
print(scipy.__version__)
print(sklearn.__version__)
print(matplotlib.__version__)
print(pandas.__version__)
print(tensorflow.__version__)
```

Output:

```
3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609]
1.11.0
0.19.1
0.19.0
1.5.1
0.20.3
1.3.0
```

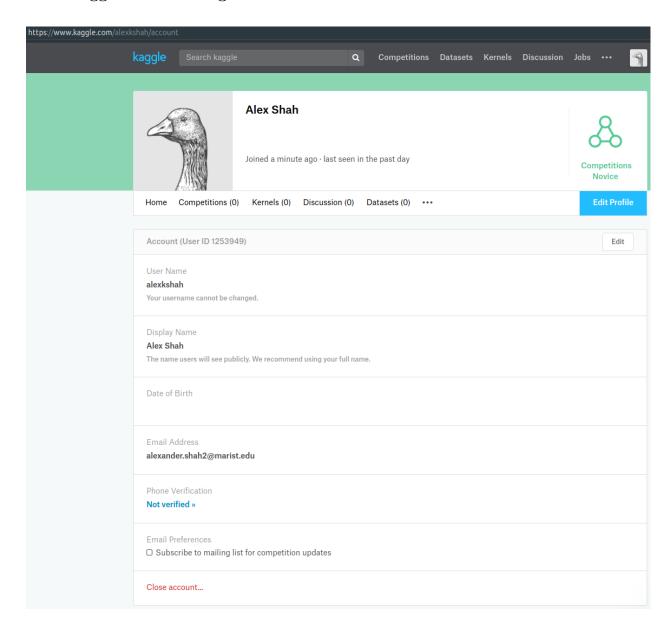
2 Github Proof 2.2

2.1 Github Collaborators



3 Kaggle Proof 2.3

3.1 Kaggle Account Page



4 Problems

4.1

This problem was solved using Scipy's fmin function, inverted to produce a maximum. The answer is 4.

```
# 4-1.py
# Alex Shah
# Homework 0

from scipy.optimize import fmin

def g(x): return -3.0*x**2+24*x-30

maximum = print(fmin(lambda x: -g(x), 0.0))

# Output:
# Optimization terminated successfully.
# Current function value: -18.000000
# Iterations: 29
# Function evaluations: 58
# [ 4.] <-- Answer
```

4.2

Using x and y for readability, the partial derivative of f(x) with respect to x:

$$f(x) = 3x^3 - 3xy^2 + 4y - 8 (1)$$

$$3 * 3 * x^2 - 3 * y^2 + 4 * 0 - 0 \tag{2}$$

$$9x^2 - 3y^2 \tag{3}$$

The partial derivative of f(x) with respect to y:

$$f(x) = 3x^3 - 3xy^2 + 4y - 8 (4)$$

$$3*0 - 3x*2y + 4*1 - 0 \tag{5}$$

$$6xy + 4 \tag{6}$$

```
\# 4-3.py
# Alex Shah
# Homework 0
import tensorflow as tf
import numpy as np
# Matrices
a \, = \, \mathrm{np.array} \, (\, [\, [\, 1.\, 0\,\, , 4\,\, , 3\,] \,\, , \,\, [\, 2\,\, , \, -1\,\, , 3\,] \,]\,)
b = np.array([[-2,0,5], [0,-1,4]])
c = np.array([[1,0], [0,2]])
\# tf const
a = tf.constant(a.astype(float))
b = tf.constant(b.astype(float))
c = tf.constant(c.astype(float))
# matrix cannot be multiplied as is,
# inner dimensions are different,
\# transpose in order to multiply
At = tf.transpose(a)
Bt = tf.transpose(b)
Ci = tf.matrix_inverse(c)
with tf. Session() as sess:
    try: print(tf.matmul(a,b).eval())
    except Exception as err:
         print("\n_A_x_B_fails_as_dimensions_are_different:")
         print(err)
    print("\n_At_x_B")
    res = tf.matmul(At,b).eval()
    print(res)
    print("\n_Rank")
    print(np.linalg.matrix_rank(res))
    print("\n_A_x_Bt")
    res2 = tf.matmul(a,Bt).eval()
    print(res2)
    print("\n_A_x_Bt_+_C_inverse")
    print(tf.add(res2, Ci).eval())
```

Output: