Lab 1 - Wyatt Madden & Dan Crowley

January 29, 2020

1

Lets input the mat file and check the dimensions of the array.

```
In [2]: import scipy.io as scipy
        import numpy as np
        from pytictoc import TicToc as tictoc
        mat = scipy.loadmat('/Users/wyattmadden/Documents/school/' +
                            'MSU/2020/m508/labs/lab_1/data.mat')
        X = mat['X']
        X.shape
```

```
Out[2]: (10, 1000000)
```

First we calculate the L2 of X using a for-loop to sum the euclidean distances. We see this gives us the expected output.

```
In [3]: s = 0
        for i in range(0, X.shape[1]):
            s = s + np.linalg.norm(X[0:X.shape[0], i])
```

Next we calculate the L2 of X using vectorized operations in NumPy, first calulating the sums of squares within the first dimension of X, resulting in a 1,000,000 length vector with values that are each the euclidean distance of 10 numbers, and then summing this vector. We see this results in the same value, as expected.

```
In [4]: vector_of_L2_norms = np.linalg.norm(X, axis = 0)
        sum_of_squares = np.sum(vector_of_L2_norms)
        sum_of_squares
```

```
Out[4]: 1380518.375334337
```

2

Now lets run both calculations again, timing each.

```
In [6]: non_vectorized_time = tictoc()

        non_vectorized_time.tic()

        s = 0
        for i in range(0, X.shape[1]):
            s = s + np.linalg.norm(X[0:X.shape[0], i])

        non_vectorized_time.toc()

Elapsed time is 4.679219 seconds.
```

```
In [7]: vectorized_time = tictoc()

        vectorized_time.tic()

        vector_of_L2_norms = np.linalg.norm(X, axis = 0)
        sum_of_squares = np.sum(vector_of_L2_norms)

        vectorized_time.toc()

Elapsed time is 0.112750 seconds.
```

We see that the non-vectorized calculation took 3.58 seconds, while the non-vectorized calculation took only 0.05 seconds. The vectorized calculation was over 70 times faster than the non-vectorized calculation!