Maaz Siddiqui
Assignment 1
Machine Learning Image Analysis

Project Report

The problem:

Given NxN matrix, calculate pairwise Euclidean distance using “for” loops and “numpy” library in Python. Generate random matrices and profile the performance. The most important reason to do this exercise is to measure huge performance difference between “for” loops and “numpy” library.

Solution:

For Loop:

“For” loop implementation was simple using basic Euclidean formula.

```
def EDistance_loop(X):
    result_loop = np.zeros_like(X)
    distance = 0
    for i in range(len(X)): # no. of rows
        ci = 0
        while(ci < len(X)):
            for j in range(len(X[0])): # no. of cols
                distance += ((X[i][j]-X[ci][j])**2)
            result_loop[i][ci] = np.float64(np.sqrt(distance))
            ci += 1
            distance = 0
    return result_loop
```

Numpy:

Numpy implementation was too complex because I have to learn all concepts from scratch and then come up with the following function. A lot of online research/help was used to impltement this function.

```
def EDistance_cool(X):
    result_cool = np.zeros_like(X)
    result_cool = np.sqrt((np.square(X[:,np.newaxis]-X).sum(axis=2)))
    return result_cool
```

Experiments:

Both functions were tested manually first and then using random numbers. Results were plotted on a graph using Pythons’s “matplotlib” library. Run time was measured between both functions.

Result and discussion:

Results came out with huge time difference. Seems like “for” loop result grows exponentially and on the other hand “numpy” seems like a constant. I will now spend more time to learn “numpy” and Python.