Theodore Nguyen
704-156-701
Math 156, Spring 2016
Lab 1: Introduction
04/05/15

Intro and Purpose:

The goal of the lab is get acquainted with MATLAB to load and visualize data and perform basic operations to see the effectiveness of different programming practices on big data, as well as provide the outline for doing other labs in this class.

Provided Resources:

We are provided with one resource, a file "**data.mat**" which contains a single matrix **X** that is a 10 row by 1,000,000 column matrix. Essentially, you can look at it as a concatenation of one million 10-dimensional column vectors.

Guide:

We want to observe the differences in efficiency for different algorithms on large datasets. Here, given the matrix **X**, we want to compute the sum of the L2 norms of each of the 1,000,000 column vectors. That is, compute the L2-norm a million times for each of the 10-dimensional column vectors. Then, add up each of those million L2-norms. We will do this using two methods: (1) looping a million iterations and adding the norm of each vector together after computing the norm using MATLAB's **norm** function; (2) using the MATLAB's **sum** function to compute the sum of the million-dimensional row vector whose entries are the L2-norms of each of each column vector in X. We will also time how long it takes to do each of those two methods.

Results:

The attached MATLAB code can be seen on the last page. The code is split up into three segments after the data has been loaded: the computation (1) using the loop, (2) using the sum function step by step, (3) using the sum function in one line of code. (2) and (3) are basically the same thing. Each segment computes the desired sum, times how long it takes, and then outputs the computation's value to verify its accuracy. The specification desires that the value of the sum should be approximately $1.4*10^6$. We observed that the sums from our computations are equivalent with the value $1.380518*10^6$, which is approximately the value we desire.

On one specific run, as shown in the screenshot on the last page following the code (ran on the PIC lab's Linux compute server), we see that the computation time for each of the runs are **0.790199 seconds** for the looping method, and **0.033162 and 0.032006 seconds** for the **sum** methods, respectively to (2) and (3) above. The latter two are approximately the same. As we can see, using MATLAB's built in functions is exceptionally more efficient than computing manually, especially when an individual's manual code is not optimized. The looping method is of **O(n)** complexity, and in this case **n = 1,000,000.** MATLAB's **norm** function might be more optimized than computing the norm of each vector manually, but the overhead generated from using the loop far out scales this benefit. We do not have explicit information on the implementation of the **sum** function, but we can tell it probably has less computational complexity than using a loop. A quick internet search shows that it might be implemented using multithreading, allowing significant more efficiency over the looping method if computed on a multicore machine. Either way, the point is that we can see that using one method over the other can greatly reduce computation time – in this case, the time is over 20 times greater. For a larger dataset, this can make a much larger difference.

Script for the lab:

```
%Theodore Nguyen, 704-156-701, Math 156 Lab 1 Spring 2016. Script.
%load data. Have data.mat in the working directory.
data = load('data.mat');
X = data.X;
%initialize counter sum
s = 0;

%use first looping method to compute sum, and time it
fprintf('We first compute the sum using the loop.\n');
tic
for i = 1:size(X,2)
    s = s + norm(X(:,i));
end
toc
fprintf('The sum computed using the looping method is %e.\n\n', s);

%use sum function to do the same thing
s = 0;
fprintf('We now compute the sum using the sum function with MATLAB functions, step by step.\n');
tic
%compute the power of each entry of the matrix (thus computing the power of
%each element of each vector in each column)
elementwiseSquare = X.^2;
%compute the sum of those squares for each column
sumofsquares = sum(elementwiseSquare, 1);
%compute the square root of those sums to get the sum of squares for each
%column - aka the L2 norm for each column vector
L2norms = sqrt(sumofsquares);
%compute the sum of the L2 norms
s = sum(L2norms);
toc
fprintf('The sum computed using MATLAB functions is %e.\n\n', s);

%try the sum function all in one step
s = 0;
fprintf('We now compute the sum with the same functions all in one step.\n');
tic
s = sum(sqrt(sum(X.^2,1)));
toc
fprintf('The sum computed with MATLAB functions in one step is %e.\n', s);
```

Computation screenshot

```
>> math156lab1
We first compute the sum using the loop.
Elapsed time is 0.790199 seconds.
The sum computed using the looping method is 1.380518e+06.

We now compute the sum using the sum function with MATLAB functions, step by step.
Elapsed time is 0.033162 seconds.
The sum computed using MATLAB functions is 1.380518e+06.

We now compute the sum with the same functions all in one step.
Elapsed time is 0.032006 seconds.
The sum computed with MATLAB functions in one step is 1.380518e+06.
>>
```