IE531: Algorithms for Data Analytics
Spring, 2018
Programming Assignment 2: Experimental Verification of the Johnson-Lindenstrauss Lemma
Due Date: March 2, 2018

1 Introduction

We have n-many, d-dimensional vectors, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\forall i, \mathbf{x}_i \in \mathcal{R}^d$, and d is very large. We wish to find n-many, k-dimensional vectors, $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$ where $\forall i, \mathbf{y}_i \in \mathcal{R}^k$, and $k \ll d$ is comparatively smaller than d. Theorem 2.11 of the text as the following form –

Theorem 1.1 *(Johnson-Lindenstrauss Lemma) For any $\epsilon \in (0, 1)$, and any integer n, let $k \geq \frac{3}{c\epsilon^2} \ln n$, with c as defined in Theorem 2.9. For any set of n-many vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ where $\mathbf{x}_i \in \mathcal{R}^d$, the random projection $f : \mathcal{R}^d \to \mathcal{R}^k$ defined in section 2.7 of the text has the property that for all pairs $\mathbf{x}_i$ and $\mathbf{x}_j$, with probability at least $1 - \frac{3}{2n}$,*

$$(1-\epsilon)\sqrt{k}\|\mathbf{x}_i - \mathbf{x}_j\| \leq \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\| \leq (1+\epsilon)\sqrt{k}\|\mathbf{x}_i - \mathbf{x}_j\|$$

(Note, I have replaced the text's $| \bullet |$ with my $\| \bullet \|$ to denote length of the vector-argument).

2 Discussion

For this programming assignment we will take a slightly different form/version of the JL-Lemma. As before, we would like a mapping $f : \mathcal{R}^d \to \mathcal{R}^k$ (where $k \ll d$) such that the distance between any two d-dimensional points $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}^d$ is more-or-less preserved when we look at $f(\mathbf{x}_1), f(\mathbf{x}_2) \in \mathcal{R}^k$. We are going to rewrite the requirement of theorem 1.1 as equation 1 shown below, where for some (small) $\epsilon \in \mathcal{R}$, and $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}^d$, we want

$$(1-\epsilon)\|\mathbf{x}_1 - \mathbf{x}_2\| \leq \|f(\mathbf{x}_1) - f(\mathbf{x}_2)\| \leq (1+\epsilon)\|\mathbf{x}_1 - \mathbf{x}_2\| \tag{1}$$

As suggested by the text, you make k-many calls to a routine that generates d-dimensional Gaussian RVs with zero-mean and an identity-covariance matrix. Let us suppose this process gets us the vectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$, where each $\mathbf{u}_i \in \Re^d$. You re-scale each $\mathbf{u}_i \in \Re^d$ according to

$$\mathbf{u}_i \leftarrow \sqrt{\frac{d}{k}} \times \frac{\mathbf{u}_i}{\|\mathbf{u}_1\|}$$

This is needed to preserve the expected-norm (Why?[1]). You stack the k-many, members of $\{\mathbf{u}_i^T\}_{i=1}^k$ on top of each other to get the $(k \times d)$ matrix $\mathbf{A}$. For each $\mathbf{x}_i \in \Re^n$, you define $f(\mathbf{x}_i) = \mathbf{A}\mathbf{x}_i$ $(\in \mathcal{R}^k)$.

There is a version of *Johnson-Lindenstrauss Lemma* that says that if

$$k = O\left(\frac{1}{\epsilon^2}\ln(\frac{n}{\delta})\right),$$

then $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}^n$ equation 1 will be satisfied with probability at least $(1 - \delta)$. I leave it as an exercise to you to show that theorem 1.1 and the above version are theoretically similar. As noted in class, the neat thing about the JL-Lemma is that this bound on k does not depend on d (the dimension we want to reduce/scale-down). There are ways of gaining small efficiencies when it comes to producing the random map $f(\bullet)$ alluded to above. But that is for you to discover after you have done some experimentation.

If you spend enough time experimenting with this, you will see that while the $(\ln n)$-term is nice, the $\frac{1}{\epsilon^2}$-term can be a killer. For example, if $\epsilon = 0.01$ (i.e. 1% error), we will need $k \approx \ln n \times 10^4$. If this is to have any practical-value/use we should have a $d \gg \ln n \times 10^4$. If $\epsilon = 0.1$ (i.e. 10% error), we will need $k \approx \ln n \times 100$. On the flip-side, keep in mind that the error-bounds in the formal theorem is an upper bound (i.e. it is the worst-case) – you may find that the JL-Lemma in practice might not be as bad as the worst-case. It is for you to experiment and find out – as a part of this programming exercise.

3 Data for Verification

The Compass site should contain a data file JL_Data1 that contains data for a (10000×1000) matrix that represents the set $\{\mathbf{x}_i\}_{i=1}^{1000}$, where each $\mathbf{x}_i \in \mathcal{R}^{10,000}$. We want to represent each $\mathbf{x}_i \in \mathcal{R}^{10,000}$ with a $\mathbf{y}_i \in \mathcal{R}^k$, where $k < 10,000$. As you can gather, $n = 1,000$ for this data set. The plan is to have you try a bunch of different values for k (say, $k \in \{200, 300, 400, 500\}$) and see if the bounds in the statement of the JL-Lemma holds for this data-set.

Thrombin Data

This is not necessary for the programming assignment. I am just providing this to you to work with a bigger dataset. I know practically nothing about drug-discovery. All I know is that there is a small role for the *JL-Lemma* in reducing the dimension for a standard search that is part of the process. My superficial understanding is that every time a new drug (i.e. chemical compound) is to be considered, you have to check if there are others in your list that have a similar "performance" – this involves checking the nearest neighbor of the new compound in the list. I am not even sure about all this. That said, we have a "real-world" data set that we can for testing the JL-Lemma.

[1] A very careful reading of my notes, the lectures, and the text, should tell you why!

You can try the training-data for *Prediction of Molecular Bioactivity for Drug Design* at this link. Each row of this CSV file represents a chemical-compound (i.e. a component of a drug). The first-item in each row is either an "A" or an "I" (representing *Active*/*Inactive*). This is followed by 139,531 many comma-seperated 0/1 values representing location on the Thrombin molecule (i.e. $d = 139,531$). There are 1909-many chemical compounds (i.e. $n = 1909$). The Compass website contains some Python-Code I wrote to process this data set to output a file that can be read the code you would have written for this programming assignment. You can explore if each 139,531-dimensional binary vector can be represented by a shorter k-dimensional vectors that meet all the requirements of the JL-Lemma. Do not be surprised if you see some weird/interesting results ☺☺.

4 Requirements

What I need from you:

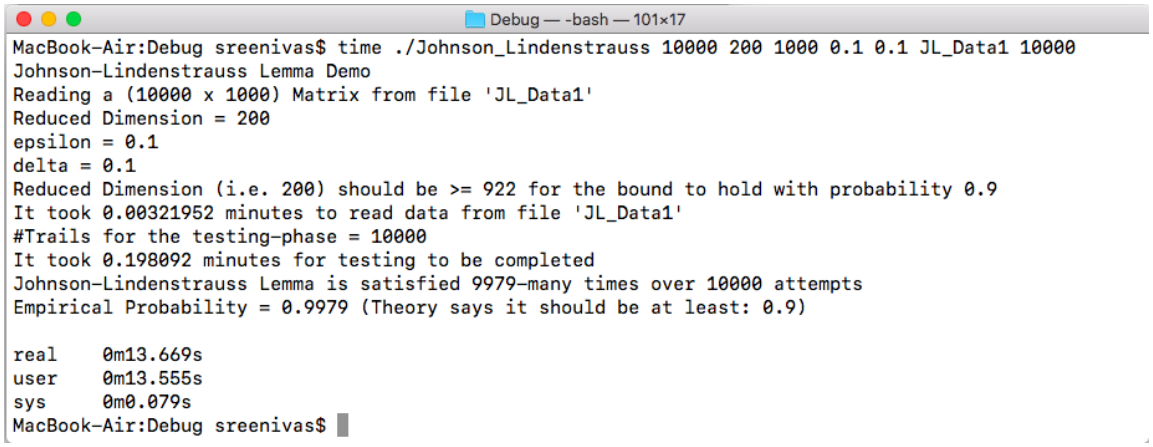
1. *.cpp and *.h that verifies the JL-Lemma by taking a bunch of values at the command-line.
   (a) First Command Line Variable - d,
   (b) Second Command Line Variable - k,
   (c) Third Command Line Variable - n,
   (d) Fourth Command Line Variable - ϵ,
   (e) Fifth Command Line Variable - δ,
   (f) Sixth Command Line Variable - input-filename (that contains the relevant data), and
   (g) Seventh Command Line Variable - number_of_trials, which is the number of statistical-trials where the JL-Lemma is experimentally/statistically verified.

I have provided a hint on Compass for anyone that needs it. To verify the JL-Lemma, you will implement what is shown below in pseudo-code.

```
no_of_hits = 0;
for int i = 1; i ≤ no_of_trials; i++ do
    pick a pair of vectors x_1, x_2 ∈ R^d;
    compute y_1 = A x_1 and y_2 = A x_2;
    if (1 − ε)‖x_1 − x_2‖ ≤ ‖y_1 − y_2‖ ≤ (1 + ε)‖x_1 − x_2‖ then
        no_of_hits++;
    end if
end for
Check if no_of_hits / no_of_trials ≥ (1 − δ)
```

I am looking for an output that is along the lines of what is shown in figures 1 and 2. I want you to try a bunch of values for k, and see if the specifications

of the JL-Lemma is verified for all of them. A single page statement of your experiments will suffice.



```
Debug — -bash — 101×17
MacBook-Air:Debug sreenivas$ time ./Johnson_Lindenstrauss 10000 200 1000 0.1 0.1 JL_Data1 10000
Johnson-Lindenstrauss Lemma Demo
Reading a (10000 x 1000) Matrix from file 'JL_Data1'
Reduced Dimension = 200
epsilon = 0.1
delta = 0.1
Reduced Dimension (i.e. 200) should be >= 922 for the bound to hold with probability 0.9
It took 0.00321952 minutes to read data from file 'JL_Data1'
#Trails for the testing-phase = 10000
It took 0.198092 minutes for testing to be completed
Johnson-Lindenstrauss Lemma is satisfied 9979-many times over 10000 attempts
Empirical Probability = 0.9979 (Theory says it should be at least: 0.9)

real    0m13.669s
user    0m13.555s
sys     0m0.079s
MacBook-Air:Debug sreenivas$
```

Figure 1: A sample output.

```
Debug — -bash — 103×65
csl-155-01:Debug rsree$ time ./Johnson_Lindenstrauss 139351 400 1908 0.01 0.01 thrombin.data 1000
Johnson-Lindenstrauss Lemma Demo
Reading a (139351 x 1908) Matrix from file 'thrombin.data'
Reduced Dimension = 400
epsilon = 0.01
delta = 0.01
Reduced Dimension (i.e. 400) should be >= 121590 for the bound to hold with probability 0.99
It took 1.00842 minutes to read data from file 'thrombin.data'
#Trials for the testing-phase = 1000
It took 0.420848 minutes for testing to be completed
Johnson-Lindenstrauss Lemma is satisfied 228-many times over 1000 attempts
Empirical Probability = 0.228

real    2m57.862s
user    2m55.855s
sys     0m1.739s
csl-155-01:Debug rsree$ time ./Johnson_Lindenstrauss 139351 500 1908 0.01 0.01 thrombin.data 1000
Johnson-Lindenstrauss Lemma Demo
Reading a (139351 x 1908) Matrix from file 'thrombin.data'
Reduced Dimension = 500
epsilon = 0.01
delta = 0.01
Reduced Dimension (i.e. 500) should be >= 121590 for the bound to hold with probability 0.99
It took 0.994209 minutes to read data from file 'thrombin.data'
#Trials for the testing-phase = 1000
It took 0.421682 minutes for testing to be completed
Johnson-Lindenstrauss Lemma is satisfied 245-many times over 1000 attempts
Empirical Probability = 0.245

real    3m19.575s
user    3m17.950s
sys     0m1.542s
csl-155-01:Debug rsree$ time ./Johnson_Lindenstrauss 139351 1000 1908 0.01 0.01 thrombin.data 1000
Johnson-Lindenstrauss Lemma Demo
Reading a (139351 x 1908) Matrix from file 'thrombin.data'
Reduced Dimension = 1000
epsilon = 0.01
delta = 0.01
Reduced Dimension (i.e. 1000) should be >= 121590 for the bound to hold with probability 0.99
It took 0.977198 minutes to read data from file 'thrombin.data'
#Trials for the testing-phase = 1000
It took 0.422498 minutes for testing to be completed
Johnson-Lindenstrauss Lemma is satisfied 363-many times over 1000 attempts
Empirical Probability = 0.363

real    5m11.638s
user    5m9.670s
sys     0m1.853s
csl-155-01:Debug rsree$ time ./Johnson_Lindenstrauss 139351 1000 1908 0.05 0.05 thrombin.data 1000
Johnson-Lindenstrauss Lemma Demo
Reading a (139351 x 1908) Matrix from file 'thrombin.data'
Reduced Dimension = 1000
epsilon = 0.05
delta = 0.05
Reduced Dimension (i.e. 1000) should be >= 4220 for the bound to hold with probability 0.95
It took 0.981462 minutes to read data from file 'thrombin.data'
#Trials for the testing-phase = 1000
It took 0.423059 minutes for testing to be completed
Johnson-Lindenstrauss Lemma is satisfied 968-many times over 1000 attempts
Empirical Probability = 0.968

real    5m12.667s
user    5m10.552s
sys     0m1.916s
csl-155-01:Debug rsree$
```

Figure 2: A sample output on Thrombin-Data (run on my iMac).