Prelude

Part of my research focuses on random projections, which involves running simulations and evaluating the performance of algorithms using random projections (and other random algorithms) on specific datasets.

I intend this Github folder to serve several purposes:

1. Make code I use in my research public, to encourage reproducible research (or help spot annoying tpyos)
2. Give some motivation to why I do this research
3. Encourage potential students (undergraduates, PhDs) to do similar research

Contents

1 Data and Distances

Most data can be represented as a vector in $\mathbb{R}^D$, where D is some positive integer. Here are some examples.

1. **Book data**

   We could record the width, thickness, and height of a book as a vector in $\mathbb{R}^3$.

   For example, the hard copy of `Simulation, 4th Edition` by [Ross, 2006] in my book collection has a width of 15.6cm, 2.2cm, and height of 23.5cm. This could be recorded as the vector $(15.6, 2.2, 23.5) \in \mathbb{R}^3$.

2. **Pictures of objects**

   This is a picture of a cat, which has dimensions 1000×700 pixels. Each pixel can be represented as a vector in $\mathbb{R}^3$ (storing colour intensity in red, green or blue). Hence this picture can be represented as a vector in $\mathbb{R}^{2100000}$, where $2100000 = 1000 \times 700 \times 3$.

Figure 1: Cat picture from AP/Nestle Purina PetCare

3. **Exam data**

   For a mathematics exam consisting of N questions, we could store a student's score for the exam as a vector of $\mathbb{R}^N$, where the i^{th} element of the vector, $i = 1, \ldots, N$ consists of the student's score for the i^{th} question.

The types of data above are all examples of continuous data, and this is the type of data I currently look at.

In most cases, we can use the notion of distance to do some kind of learning on the data.

1.1 k-nearest neighbours

Example 1.1. *Finding which group an unknown vector belongs to*
Suppose we have vectors (data) in $\mathbb{R}^2$, and we know beforehand whether they belong to Group 1, or Group 2. Suppose further we have an unknown vector in $\mathbb{R}^2$, and we want to figure out whether it belongs to Group 1 or Group 2.

Since the vectors are in $\mathbb{R}^2$, we can plot them as follows in Figure 2. Which group do you think the unknown vector (represented by ∇) belongs to - Group 1 or Group 2?

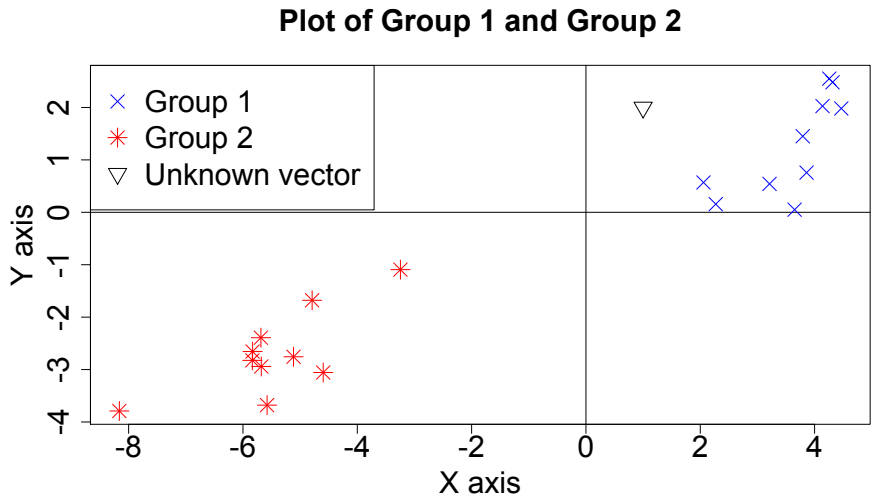

Figure 2: Toy example of two groups and one unknown vector.

Based on the plot, we might reason thus: "*Well, the black triangle is closer to all the* $\times$ *than all the* $*$*, so we say that the unknown vector belongs to Group 1.*"

However, when we reasoned "closer", what we really meant was that the Euclidean distances[1] of our unknown point to the $\times$ points were much shorter than the Euclidean distances of our unknown point to the $*$.

We might be interested in extending this example to higher dimensions $\mathbb{R}^D$ where $D > 2$, but as we[2] cannot visualize plots in 4 dimensions or more, it is useful to look at our 2D example

[1] It is also possible that you might have looked at the angular distance, i.e. the angle between the vector representing our unknown point and the vector of a point in Group 1 / Group 2 rather than the Euclidean distance- that is okay as well.

[2] most humans

in Example 1.1 to come up with some heuristic to classify vectors in higher dimensions.

This gives rise to the k-nearest neighbours algorithm which works like this.

Require vectors $\vec{x}_1^{(1)}, \ldots, \vec{x}_{n_1}^{(1)}$ belonging to Group 1
Require vectors $\vec{x}_1^{(2)}, \ldots, \vec{x}_{n_2}^{(2)}$ belonging to Group 2
Choose a value of k
Have vectors $\vec{x}_1, \ldots, \vec{x}_m$ to classify
for each $\vec{x}_i, 1 = 1, 2, \ldots, m$ **do**
  Compute and store the $n_1 + n_2$ `distances` between $\vec{x}_i$ and the $n_1 + n_2$ vectors $\vec{x}_1^{(1)}, \ldots, \vec{x}_{n_1}^{(1)}, \vec{x}_1^{(2)}, \ldots, \vec{x}_{n_2}^{(2)}$
  Pick the k shortest distances
  Out of the k shortest distances, tabulate the number of vectors belonging to Group 1 which $\vec{x}_i$ is closest to, and the number of vectors belonging to Group 2 which $\vec{x}_i$ is closet to
  Classify $\vec{x}_i$ based on majority vote
end

Algorithm 1: k nearest neighbours algorithm

1.2 Types of Distances ...

There are many other types of distances that we can use apart from the Euclidean distance or angular distance. For example, here is a (non-exhaustive) list of certain types of distances which we can compute

1. Angular distance
2. Dot product
3. Euclidean distance
4. Hamming distance
5. Jaccard similarity
6. l_1 distance
7. l_p distance
8. l_∞ (Chebyshev) distance
9. Resemblance

There is Matlab code in the folder `actual_distances` that computes these distances.

More precisely, suppose we have $N := n_1 + n_2$ vectors in $\mathbb{R}^p$, where n_1 vectors belong to one group, and n_2 vectors belong to another group. Then we can create matrices X_1 of size $n_1 \times p$, and X_2 of size $n_2 \times p$, where the row vectors of X_1 correspond to vectors in the first group, and the row vectors of X_2 correspond to vectors in the second group.

This code in the folder takes in as input the matrices X_1, X_2, a certain distance type, optional parameters corresponding to the distance type, and outputs a distance structure consisting of a distance matrix D of size $n_1 \times n_2$.

The $(i, j)^{\text{th}}$ element in this distance matrix D corresponds to the distance between the i^{th} row of X_1 and the j^{th} row of X_2.

Here are some example inputs and outputs:

```
>X1 = binornd(1,0.5,10,50);  % randomly generate X1
>X2 = binornd(1,0.5,30,50);  % randomly generate X2

% computes l_4 distance between pairs
>get_pairwise_distances(X1, X2, 'lp_distance', 4)

ans =

  struct with fields:

     dist_mat: [10 x 30 double]
    dist_type: 'lp_distance'
       dist_p: 4


>get_pairwise_distances(X1,X2,'hamming_distance')

ans =

  struct with fields:

     dist_mat: [10 x 30 double]
    dist_type: 'hamming_distance'
```

You can see from the code that we output a distance structure rather than just a distance

matrix. This makes it easier for debugging (for example, which distance did we choose to compute?)

1.3 ... and techniques requiring them

Surprisingly, a lot of statistical techniques use the concept of distances. Some brief examples

1. Least squares / projection (Euclidean distance)
2. Linear / Quadratic discriminant analysis (Euclidean distance, Mahalanobis distance)
3. Support vector machines (inner products)
4. Similarity search (Hamming distance, Jaccard similarity, resemblance)

We even use the concept of distance in hypothesis testing as well.

1.4 Where does the research come in?

We could imagine new machine learning techniques to work as such:

1. We throw in data into a black box machine learning algorithm
2. Distances between data are computed
3. Magic happens
4. Machine learning algorithm makes some prediction

and hence there is nothing new to be done (apart from coming up with new algorithms).

However, there are some points to ponder about.

1. *Speed of computation of these distances*

   One can imagine that as we store more and more data, i.e, we store vectors of size $\mathbb{R}^D$ when D is large, computing distances between two vectors will take time proportional to D.

   For example, computing the angular distance between two vectors in $\mathbb{R}^2$ would be much faster than computing the angular distance between two vectors in $\mathbb{R}^{1000000000}$.

   A machine learning algorithm might take drastically more time to run with bigger vectors.

2. **'Storage' of vectors**

   In the above example, we might have had vectors in $\mathbb{R}^{1000000000}$. But suppose we also had lots of these vectors (eg, imagine the number of all cat pictures N on the internet). N and D might be large, and we may not have enough space to store them in memory.

   However, if we aim to do some prediction, and a machine learning algorithm only uses the computed distances for prediction, then we don't really need these original vectors, just the distances between vectors. Maybe there might be a better way to store these vectors?

3. **Privacy**

   People might be concerned about data breaches. Similar to the 'storage' of vectors point above, we might be interested in a way to store these vectors such that we store the "distances" between vectors for any kind of analysis, yet do not retain the original vectors which might store sensitive information.

Random projections (and locality-sensitive hashing) types of algorithms are dimension reduction algorithms, reducing the size of data from dimension D to a smaller dimension k. Distances are preserved under such algorithms in expectation, and the relative error of these distances usually depend on the smaller dimension k, rather than the initial dimension D.

This means - if today we work with vectors in $\mathbb{R}^D$ where $D = 1,000,000$, and we reduce the size of the vectors to $k = 100$ dimensions, we may have a certain relative error E when computing pairwise distances. Tomorrow, if we work with vectors in $\mathbb{R}^D$ where $D = 1,000,000,000$ and we reduce the size of the vectors to $k = 100$ dimensions as well, our relative error is still about the same E.

So the initial large dimension D does not matter. Only the smaller dimension k matters!

My research focuses on **further reducing the relative error of the computation of distances** for such algorithms, and looking at the effects of this reduction with several machine learning algorithms.

2 Construction of Random Projections with i.i.d. $N(0,1)$ for Euclidean distance and Dot product

Suppose we had vectors $\vec{x}_1, \vec{x}_2 \in \mathbb{R}^m$, and we are interested in some distance D between the two vectors, which we can compute by some function[3] $D = d(\vec{x}_1, \vec{x}_2)$. Some distances we may be interested in include

1. Euclidean distance, given by

$$\begin{aligned} d_{\text{Euclidean distance}}(\vec{x}_1, \vec{x}_2) &:= \sqrt{(x_{11} - x_{21})^2 + \ldots + (x_{1m} - x_{2m})^2} \\ &= \sqrt{\|\vec{x}_1\|^2 - 2\vec{x}_1^T\vec{x}_2 + \|\vec{x}_2\|^2} \end{aligned}$$

2. Dot product, given by

$$\begin{aligned} d_{\text{Dot product}}(\vec{x}_1, \vec{x}_2) &:= x_{11}x_{21} + \ldots + x_{1m}x_{2m} \\ &= \vec{x}_1^T\vec{x}_2 \end{aligned}$$

3. Angular distance, given by

$$d_{\text{Angular distance}}(\vec{x}_1, \vec{x}_2) := \arccos\left(\frac{\vec{x}_1^T\vec{x}_2}{\|\vec{x}_1\|\|\vec{y}_2\|}\right)$$

4. l_p distance, given by

$$d_{l_p \text{ distance}}(\vec{x}_1, \vec{x}_2) := ((x_{11} - x_{21})^p + \ldots + (x_{1m} - x_{2m})^p)^{\frac{1}{p}}$$

where we denote

$$\|\vec{x}\| := x_1^2 + \ldots + x_m^2$$

Consider the number of operations taken to compute any of these above distances. For example, to compute the Euclidean distance, we have to do

- One "subtract" operation, and one "square" operation for each $(x_{1i} - x_{2i})^2$.
- $m - 1$ "adding" operations to sum up all m terms
- One "square root" operation

[3]This is not necessarily a metric. For example, angular distance.

which gives us $2m + m - 1 + 1 = 3m$ operations. So the number of operations we need to do is proportional to m, which is the dimension of the vectors $\vec{x}_1, \vec{x}_2$. One can expect that as m gets large, we need to spend about $O(m)$ of time computing such distances.

That's just for one pair of vectors though. Suppose we have vectors $\vec{x}_1, \ldots, \vec{x}_n$, and we might want to compute all pairwise distances $d(.\ ,\ ,.)$ between them. There's $\binom{n}{2}$ such distances, so in total we might spend about $\frac{n!}{(n-2)!2!}(3m)$ computations, which works out to about $\frac{3n(n-1)m}{2}$ operations, or something which takes about $O(n^2 m)$ of time.

The goal here is to speed up the time taken in computing such distances.

We consider the following steps

1. "Hope" that we can find some map given by $T : \mathbb{R}^m \to \mathbb{R}^k$, where we map vectors from a high dimensional space to a low dimensional space, with $k \ll m$.
2. Instead of computing distances $d(\vec{x}_i, \vec{x}_j)$ which take $O(m)$ of time, we find some (other) function $\widetilde{d}(T\vec{x}_i, T\vec{x}_j)$ which approximates the distance $d(\vec{x}_i, \vec{x}_j)$, i.e

   $$\widetilde{d}(T\vec{x}_i, T\vec{x}_j) \approx d(\vec{x}_i, \vec{x}_j)$$

   which takes only $O(k)$ time to compute.

We can represent such maps T by a random matrix R, where each entry of R is generated from some probability distribution. Figure 3 shows how this works.

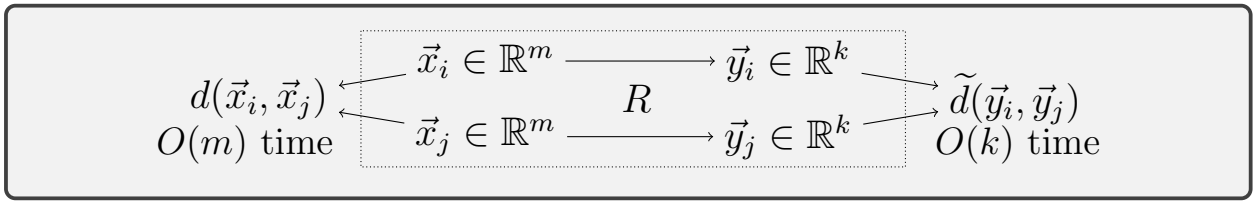

Figure 3: Estimation process for $\widetilde{d}(\vec{y}_i, \vec{y}_j) \approx d(\vec{x}_i, \vec{x}_j)$ where $T\vec{x} = \vec{y}$. The "dotted box" shows what happens for " $\approx$ ".

2.1 Finding estimates of the Euclidean distance and dot product between vector pairs

Suppose we look at two measures of distances, the Euclidean distance given by

$$d_1(\vec{x}_1, \vec{x}_2) := \sqrt{\|\vec{x}_1\|^2 - 2\vec{x}_1^T\vec{x}_2 + \|\vec{x}_2\|^2}$$

and the dot product given by

$$d_2(\vec{x}_1, \vec{x}_2) := \vec{x}_1^T\vec{x}_2$$

Consider the map $T : \mathbb{R}^m \to \mathbb{R}^k$ represented by a random matrix $R_{k\times m}$, where each r_{ij} entry in R is i.i.d. $N(0, 1)$.

The goal is to find some function $\widetilde{d}_i(R\vec{x}_1, R\vec{x}_2)$ that approximates $d_i(\vec{x}_1, \vec{x}_2)$, for $i = 1, 2$.

Let's consider the matrix multiplication of $R\vec{x}_1, R\vec{x}_2$. We have that

$$\begin{pmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \dots & \dots & \ddots & \dots \\ r_{k1} & r_{k2} & \dots & r_{km} \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1m} \end{pmatrix} = \begin{pmatrix} r_{11}x_{11} + \ldots + r_{1m}x_{1m} \\ r_{21}x_{11} + \ldots + r_{2m}x_{1m} \\ \vdots \\ r_{k1}x_{11} + \ldots + r_{km}x_{1m} \end{pmatrix} = \begin{pmatrix} \sum_{s=1}^m r_{1s}x_{1s} \\ \sum_{s=1}^m r_{2s}x_{1s} \\ \vdots \\ \sum_{s=1}^m r_{ks}x_{1s} \end{pmatrix} = \begin{pmatrix} y_{11} \\ y_{12} \\ \vdots \\ y_{1k} \end{pmatrix}$$

and similarly, we have that

$$\begin{pmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \dots & \dots & \ddots & \dots \\ r_{k1} & r_{k2} & \dots & r_{km} \end{pmatrix} \begin{pmatrix} x_{21} \\ x_{22} \\ \vdots \\ x_{2m} \end{pmatrix} = \begin{pmatrix} r_{11}x_{21} + \ldots + r_{1m}x_{2m} \\ r_{21}x_{21} + \ldots + r_{2m}x_{2m} \\ \vdots \\ r_{k1}x_{21} + \ldots + r_{km}x_{2m} \end{pmatrix} = \begin{pmatrix} \sum_{s=1}^m r_{1s}x_{2s} \\ \sum_{s=1}^m r_{2s}x_{2s} \\ \vdots \\ \sum_{s=1}^m r_{ks}x_{2s} \end{pmatrix} = \begin{pmatrix} y_{21} \\ y_{22} \\ \vdots \\ y_{2k} \end{pmatrix}$$

For our vector $\vec{y}_i = (y_{i1}, y_{i2}, \ldots, y_{ik}), i = 1, 2$, we consider each entry $y_{is}, s = 1, \ldots, k$ as an i.i.d. observations coming from some probability distribution. In fact, each y_{is} is a weighted sum of m random $N(0, 1)$ random variables, with the weights given by $x_{i1}, x_{i2}, \ldots, x_{im}$.

Consider $\mathbb{E}[y_{i1}]$. We must have

$$\begin{aligned} \mathbb{E}\left[\sum_{s=1}^m r_{1s}x_{is}\right] &= \mathbb{E}\left[r_{11}x_{i1} + r_{12}x_{i2} + \ldots + r_{1m}x_{im}\right] \\ &= x_{i1}\mathbb{E}[r_{11}] + x_{i2}\mathbb{E}[r_{12}] + \ldots + x_{im}\mathbb{E}[r_{1m}] \\ &= x_{i1} \times 0 + x_{i2} \times 0 + \ldots + x_{im} \times 0 \\ &= 0 \end{aligned}$$

and by independence, each $y_{i1}, y_{i2}, \ldots, y_{ik}$ have a mean of zero for $i = 1, 2$.

Suppose we square our observations, and look at $y_{i1}^2, y_{i2}^2, \ldots, y_{ik}^2$. Each squared observations are also i.i.d. from some distribution. We consider $\mathbb{E}[(y_{i1})^2]$, and must have

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{s=1}^{m} r_{1s}x_{is}\right)^2\right] &= \mathbb{E}\left[\sum_{s=1}^{m} r_{1s}^2 x_{is}^2 + 2\sum_{0<s<t}^{m} r_{1s}x_{is}r_{1t}x_{it}\right] \\
&= \sum_{s=1}^{m} \mathbb{E}[r_{1s}^2]x_{is}^2 + 2\sum_{0<s<t}^{m} \mathbb{E}[r_{1s}]\mathbb{E}[r_{1t}]x_{is}x_{it} \\
&= \sum_{s=1}^{m} 1 \cdot x_{is}^2 + 2\sum_{0<s<t}^{m} (0)(0)x_{is}x_{it} \\
&= \sum_{s=1}^{m} x_{is}^2 \\
&= \|\vec{x}_i\|^2 \qquad (2.1)
\end{aligned}$$

Since the mean of $(y_{i1})^2$ is equal to $\|\vec{x}_i\|^2$, then as each $(y_{i1})^2, (y_{i2})^2, \ldots, (y_{ik})^2$ are i.i.d., we have that

$$\begin{aligned}
\mathbb{E}\left[\|\vec{y}_i\|^2\right] &= \mathbb{E}\left[y_{i1}^2 + y_{i2}^2 + \ldots + y_{ik}^2\right] \\
&= \mathbb{E}\left[\sum_{s=1}^{k} y_{is}^2\right] \\
&= k\|\vec{x}_i\|^2
\end{aligned}$$

This result implies that

$$\begin{aligned}
\mathbb{E}\left[\|\vec{y}_1 - \vec{y}_2\|^2\right] &= \mathbb{E}\left[(y_{11} - y_{21})^2 + (y_{12} - y_{22})^2 + \ldots + (y_{1k} - y_{2k})^2\right] \\
&= \mathbb{E}\left[\sum_{s=1}^{k}(y_{1s} - y_{2s})^2\right] \\
&= k\|\vec{x}_1 - \vec{x}_2\|^2 \qquad (2.2)
\end{aligned}$$

Hence, if we are interested in some $\widetilde{d}_1(\vec{y}_1, \vec{y}_2) \approx d_1(\vec{x}_1, \vec{x}_2)$, we can set

$$\widetilde{d}_1(\vec{y}_1, \vec{y}_2) \approx \sqrt{\frac{d_1(\vec{y}_1, \vec{y}_2)}{k}} \qquad (2.3)$$

By considering the mean of $y_{is}y_{js}, s = 1, \ldots, k$, and repeating the process above, we have

$$\begin{aligned}
\mathbb{E}\left[y_{is}y_{js}\right] &= \mathbb{E}\left[\left(\sum_{s=1}^{m} r_{1s}x_{is}\right)\left(\sum_{s=1}^{m} r_{1s}x_{js}\right)\right] \\
&= \mathbb{E}\left[\sum_{s=1}^{m} r_{1s}^2 x_{is}x_{js} + 2\sum_{0<s<t}^{m} r_{1s}r_{1t}x_{is}x_{js}\right] \\
&= \sum_{s=1}^{m} x_{is}x_{js}
\end{aligned}$$

and we get

$$\widetilde{d}_2(\vec{y}_1, \vec{y}_2) \approx \frac{d_2(\vec{y}_1, \vec{y}_2)}{k} \tag{2.4}$$

In practice, one can form a matrix $X = [\vec{x}_1|\vec{x}_2|\ldots|\vec{x}_n]$ by concatenating columns of vectors in $\mathbb{R}^m$. The matrix product $V = \frac{1}{\sqrt{k}}RX$ is computed, where R is of size $k \times m$ with entries i.i.d. $N(0, 1)$.

Computing the Euclidean distance (or dot product) between any two columns i, j of V gives an estimate of the Euclidean distance (or dot product) between $\vec{x}_i, \vec{x}_j$.

This is the original random projection algorithm [Indyk and Motwani, 1998] which has been used since the late 1990s.

While we can see that $(y_{1s} - y_{2s})^2$ is an unbiased estimator for the squared Euclidean distance between $\vec{x}_1, \vec{x}_2$, and that $y_{1s}y_{2s}$ is an unbiased estimator for the dot product $\vec{x}_1^T\vec{x}_2$, we might be concerned about the variance of our estimator, or interested in probability bounds on the the relative error of our estimated distance.

Before we look at the variances and probability bounds of these estimates, we should plot some graphs to get some intuition on how well the random projection estimate does.

The Matlab file `script_to_try.m` has the code for the following graphics.

We first load the `Colon` dataset [Alon et al., 1999], center the data, and normalize the vectors to have a length of 1. Each vector in the colon dataset is of $\mathbb{R}^{2000}$.

We then compute the estimate of the Euclidean distance as well as the dot product for the first two vectors in the `Colon` dataset when we project the data down to $\mathbb{R}^k$ for $k \in \{10, 20, \ldots, 1000\}$. We find the relative RMSE of these estimates.

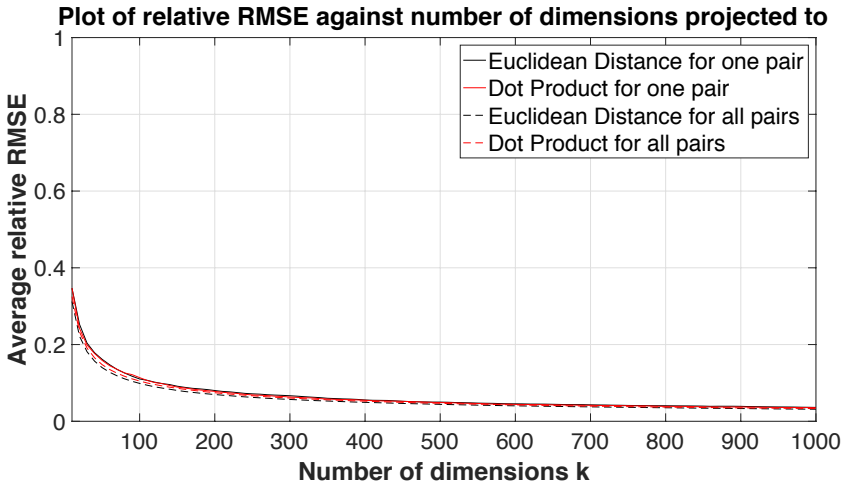

Figure 4: Plot of relative RMSE for all pairwise Euclidean distances and dot products for the Colon Dataset against dimension projected to for 1000 iterations for $k \in \{10, 20, \ldots, 1000\}$

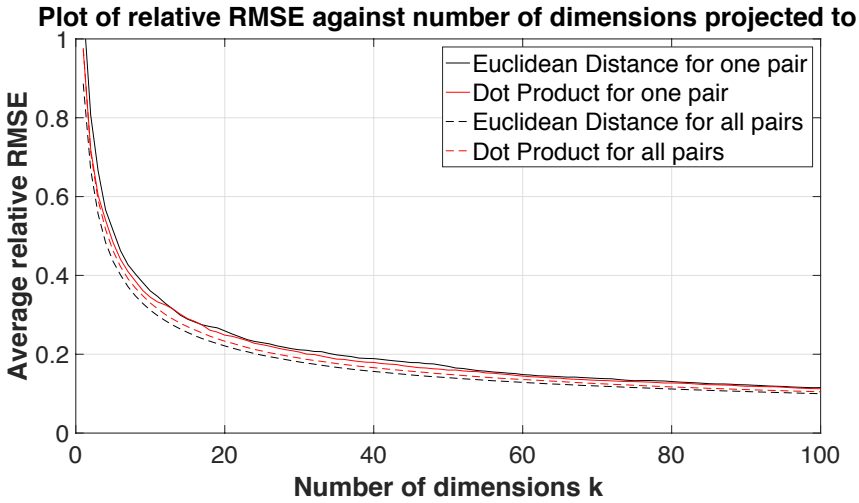

Figure 5: Plot of relative RMSE for all pairwise Euclidean distances and dot products for the Colon Dataset against dimension projected to for 1000 iterations for $k \in \{1, 2, \ldots, 100\}$.

We next compute the estimates of all pairwise Euclidean distances and the dot product for the Colon dataset (1891 pairwise distances), and compute their relative RMSE of these estimates as well.

The motivation for doing all pairwise distances is that our first two vectors of the Colon dataset may have been "good" vector pairs or "bad" vector pairs, and we want to avoid cherry picking. Computing the estimates of all pairwise distances gives some form of overall "average" on how well the random projection algorithm does.

Finally, we plot our RMSEs in Figure 4.

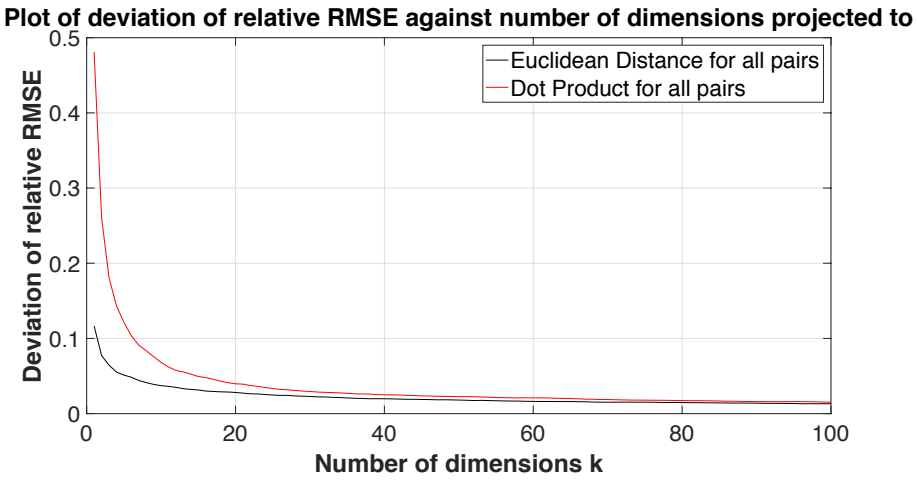

Figure 6: Plot of deviation of relative RMSE for all pairwise Euclidean distances and dot products for the Colon Dataset against dimension projected to for 1000 iterations for $k \in \{1, 2, \ldots, 100\}$

We can see that the relative RMSE quickly reduces to about 0.1 as k increases between $k \in (0, 100)$. Afterwards, the rate of decrease in the relative RMSE slows down a lot. We can't really tell much, so let's zoom in.

We hence repeat our above experiment but for values of $k = \{1, 2, \ldots, 100\}$. The results are shown in Figure 5.

It seems that the relative RMSE of the dot product is slightly higher than the relative RMSE of the Euclidean distance from Figure 5. We will plot the deviation of the relative RMSE

instead.

Figure 6 shows that the estimates of the Euclidean distance has a smaller deviation of the RMSE than the estimates of the dot product. This is in fact true in theory, as we will see in the next subsection.

2.2 Finding variance and bounds of estimates of the squared Euclidean distance and dot product between vector pairs

While our experiments show that the random projection estimates of Euclidean distance and dot products between vectors have lower relative error as the dimensions we project the vectors down increase, it is always good to have some theory about the error of the estimates.

There are actually several methods to compute such errors, and usually the method chosen depends on the audience[4].

2.2.1 Method 1: Computation of variance using Junior College / high school methods

Here, we use the generic variance formula we all know since pre-university days, where

$$\mathrm{Var}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

So for example, if we want to compute the variance of our estimate of the squared Euclidean distance between two vectors $\vec{x_i}, \vec{x_j}$, say $\mathrm{Var}[(y_{i1} - y_{j1})]$ then we need to compute

$$\mathrm{Var}\left[\left\{\sum_{s=1}^{m} r_{1s}\left(x_{is} - x_{js}\right)\right\}^2\right] = \mathbb{E}\left[\left\{\sum_{s=1}^{m} r_{1s}\left(x_{is} - x_{js}\right)\right\}^4\right] - \left(\mathbb{E}\left[\left\{\sum_{s=1}^{m} r_{1s}\left(x_{is} - x_{js}\right)\right\}^2\right]\right)^2 \tag{2.5}$$

Computing this variance can be tedious. However, there is some value in doing so because there are some neat tricks to simplify this computation. One trick is to rewrite the term $r_{1s}(x_{is} - x_{js})$ as c_s, since the indices $1, i, j$ do not change throughout the computation.

Then Equation 2.5 can be written as

$$\mathrm{Var}\left[\left(\sum_{s=1}^{m} c_s\right)^2\right] = \mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^4\right] - \left(\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^2\right]\right)^2 \tag{2.6}$$

[4]More cynically, on what venue and discipline a paper is published in.

This simplifies the expression (and helps to prevent careless mistakes). The second trick appears when we expand out the term $\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^4\right]$, and we make use of the property of the mean and variance of the Normal distribution. We do the expansion now:

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^4\right] &= \mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^2\left(\sum_{s=1}^{m} c_s\right)^2\right] \\
&= \mathbb{E}\left[\left(\sum_{s=1}^{m} c_s^2 + 2\sum_{0<s<t}^{m} c_s c_t\right)^2\right] \\
&= \mathbb{E}\left[(A+B)^2\right] \\
&= \mathbb{E}\left[A^2 + 2AB + B^2\right] \\
&= \mathbb{E}[A^2] + 2\mathbb{E}[AB] + \mathbb{E}[B^2]
\end{aligned}$$

where

$$A = \sum_{s=1}^{m} c_s^2 \qquad\qquad B = 2\sum_{0<s<t}^{m} c_s c_t$$

It suffices to find the expressions of the three expectations.

We first consider

$$\mathbb{E}[AB] = 2\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s^2\right)\left(\sum_{0<s<t}^{m} c_s c_t\right)\right]$$

and consider an arbitrary term inside this product of $\left(\sum_{s=1}^{m} c_s^2\right)\left(\sum_{0<s<t}^{m} c_s c_t\right)$.

Such a term will either be of form $c_i^2 c_j c_k$, or $c_i^3 c_j$ or $c_i^3 c_k$, since we are given that $s \neq t$ in our expression of $\left(\sum_{0<s<t}^{m} c_s c_t\right)$. But we have

$$\mathbb{E}\left[c_i^2 c_j c_k\right] = \mathbb{E}\left[c_i^2\right]\mathbb{E}\left[c_j\right]\mathbb{E}\left[c_k\right]$$

which is equal to zero since our original r_{1s} were i.i.d. for $s = 1, \ldots, p$, so $\mathbb{E}[c_k] = \mathbb{E}[r_{1k}(x_{ik} - x_{jk})] = (x_{ik} - x_{jk})\mathbb{E}[r_{1k}] = 0$.

Similarly, we have that

$$\mathbb{E}[c_i^3 c_j] = \mathbb{E}[c_i^3 c_k] = 0$$

since $i \neq j$ and $i \neq k$. Hence $\mathbb{E}[AB] = 0$.

We just need to consider $\mathbb{E}[A^2]$ and $\mathbb{E}[B^2]$.

We have

$$\begin{aligned}\mathbb{E}[A^2] &= \mathbb{E}\left[\left(\sum_{s=1}^{m} c_s^2\right)^2\right] \\ &= \mathbb{E}\left[\sum_{s=1}^{m} c_s^4 + 2\sum_{0<s<t}^{m} c_s^2 c_t^2\right]\end{aligned}$$

and

$$\mathbb{E}\left[B^2\right] = 4\mathbb{E}\left[\left(\sum_{0<s<t}^{m} c_s c_t\right)\left(\sum_{0<s<t}^{m} c_s c_t\right)\right]$$

Similar to how we analyzed $\mathbb{E}[AB]$, let us look at an arbitrary term in the expansion of $\left(\sum_{0<s<t}^{m} c_s c_t\right)\left(\sum_{0<s<t}^{m} c_s c_t\right)$. We argue that an arbitrary term will either be of the form $c_s^2 c_t^2$, $c_s^2 c_t c_{t'}$, or $c_s c_t^2 c_{s'}$.

The terms which give a non zero expectation must be of the form $c_s^2 c_t^2$, and hence we have

$$\mathbb{E}[B^2] = 4\mathbb{E}\left[4\left(\sum_{0<s<t}^{m} c_s^2 c_t^2\right)\right]$$

Putting everything together, we have

$$\begin{aligned}\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^4\right] &= \mathbb{E}\left[\sum_{s=1}^{m} c_s^4 + 6\sum_{0<s<t}^{m} c_s^2 c_t^2\right] \\ &= \mathbb{E}\left[\sum_{s=1}^{m} r_{1s}^4 (x_{is} - x_{js})^4 + 6\sum_{0<s<t}^{m} r_{1s}^2 (x_{is} - x_{js})^2 r_{1t}^2 (x_{it} - x_{jt})^2\right] \\ &= \mu_4 \sum_{s=1}^{m} (x_{is} - x_{js})^4 + 6\sum_{0<s<t}^{m} (x_{is} - x_{js})^2 (x_{it} - x_{jt})^2 \qquad (2.7)\end{aligned}$$

where we will let μ_4 denote $\mathbb{E}[r^4]$, where $r \sim N(0,1)$. For the Normal distribution with $N(0,1)$, we have $\mu_4 = 3$.

Thankfully, the second term in the expression of the variance is easier to calculate, since we

have

$$\left(\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^2\right]\right)^2 = \left(\sum_{s=1}^{m}(x_{is}-x_{js})^2\right)$$
$$= \sum_{s=1}^{m}(x_{is}-x_{js})^4 + 2\sum_{0<s<t}^{m}(x_{is}-x_{js})^2(x_{it}-x_{jt})^2$$

We finally get our expression of our variance to be

$$\text{Var}[(y_{i1}-y_{j1})^2] = 2\sum_{s=1}^{m}(x_{is}-x_{js})^4 + 4\sum_{0<s<t}^{m}(x_{is}-x_{js})^2(x_{it}-x_{jt})^2 \tag{2.8}$$

With k such estimates, then we have that

$$\text{Var}\left[\frac{(y_{i1}-y_{j1})^2+\ldots+(y_{ik}-y_{jk})^2}{k}\right] = \frac{1}{k^2}\left(\text{Var}\left[(y_{i1}-y_{j1})^2\right]+\ldots+\text{Var}\left[(y_{ik}-y_{jk})^2\right]\right)$$
$$= \frac{1}{k^2}\left(k\text{Var}\left[(y_{i1}-y_{j1})^2\right]\right)$$
$$= \frac{\text{Var}\left[(y_{i1}-y_{j1})^2\right]}{k}$$
$$= \frac{2\sum_{s=1}^{m}(x_{is}-x_{js})^4 + 4\sum_{0<s<t}^{m}(x_{is}-x_{js})^2(x_{it}-x_{jt})^2}{k}$$

We can do something similar to compute the variance of the dot product. We have

$$\text{Var}\left[y_{is}y_{js}\right] = \mathbb{E}\left[\left(\sum_{s=1}^{m} r_{1s}x_{is}\right)^2\left(\sum_{s=1}^{m} r_{1s}x_{js}\right)^2\right] - \left(\mathbb{E}\left[\left(\sum_{s=1}^{m} r_{1s}x_{is}\right)\left(\sum_{s=1}^{m} r_{1s}x_{js}\right)\right]\right)^2$$

Here, let us use the substitution $c_s = r_{1s}x_{is}$ and $d_s = r_{1s}x_{is}$. Then we have

$$\text{Var}\left[y_{is}y_{js}\right] = \mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^2\left(\sum_{s=1}^{m} d_s\right)^2\right] - \left(\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)\left(\sum_{s=1}^{m} d_s\right)\right]\right)^2$$

We can always expand

$$\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^2\left(\sum_{s=1}^{m} d_s\right)^2\right] = \mathbb{E}\left[\left(\sum_{s=1}^{m} c_s^2 + 2\sum_{0<s<t}^{m} c_s c_t\right)\left(\sum_{s=1}^{m} d_s^2 + 2\sum_{0<s<t}^{m} d_s d_t\right)\right]$$
$$= \mathbb{E}\left[(A+B)(C+D)\right]$$

and find expressions for $\mathbb{E}[AC], \mathbb{E}[AD], \mathbb{E}[BC], \mathbb{E}[BD]$, where

$$A = \sum_{s=1}^{m} c_s^2 \qquad B = 2 \sum_{0<s<t}^{m} c_s c_t \qquad C = \sum_{s=1}^{m} d_s^2 \qquad D = 2 \sum_{0<s<t}^{m} d_s d_t$$

We can see straightaway that $\mathbb{E}[AD] = \mathbb{E}[BC] = 0$, due to the fact that we will either get some term that is $r_{1s} r_{1j}^2$ or $r_{1s}^2 r_{1j}$, and hence the expectation of a single $\mathbb{E}[r_{1s}] = 0$ (same for $\mathbb{E}[r_{1j}]$).

Consider $\mathbb{E}[AC]$, and think about matching indices. Since we have that $\mathbb{E}[r_s^2] = 1$ for the normal $N(0,1)$, and used μ_4 to denote $\mathbb{E}[r_s^4]$, then we have

$$\begin{aligned}
\mathbb{E}[AC] &= \mathbb{E}\left[\sum_{s=1}^{m} c_s^2 d_s^2 + \sum_{0<s,t}^{m} c_s^2 d_t^2\right] \\
&= \mathbb{E}\left[\sum_{s=1}^{m} r_{1s}^4 x_{is}^2 x_{js}^2 + \sum_{0<s,t}^{m} r_{1s}^2 x_{is}^2 x_{jt}^2\right] \\
&= \mu_4 \sum_{s=1}^{m} x_{is}^2 x_{js}^2 + \sum_{0<s,t}^{m} x_{is}^2 x_{jt}^2
\end{aligned}$$

We finally consider $\mathbb{E}[BD]$. Suppose we write all possible indices in the sum of B as (s,t), and all indices in the sum of D as (s',t'). Note that since $s \neq t$ and $s' \neq t'$, consider what happens when we pick an arbitrary term in the product of BD. We have three small cases:

1. Case 1: s, t, s', t' are all different. In this case, our expected value of this term is of some form $\mathbb{E}[r_{1s}]\mathbb{E}[r_{1s'}]\mathbb{E}[r_{1t}]\mathbb{E}[r_{1t'}] = 0$
2. Case 2: There is only one match, i.e. s matches with s' or t', or t matches with s' or t'. In this case, we will have something of the form $\mathbb{E}[r_{1s}^2]\mathbb{E}[r_{1t}]\mathbb{E}[r_{1t'}] = 0$ as well
3. Case 3: $s = s'$ and $t = t'$. We therefore have an expression of the form $\mathbb{E}[r_{1s}^2]\mathbb{E}[r_{1t}^2] = 1$. These are the terms that are left.

We can write the terms in Case 3 as

$$\begin{aligned}
\mathbb{E}[BD] &= 4\mathbb{E}\left[\sum_{0<s<t}^{m} c_s d_s c_t d_t\right] \\
&= 4 \sum_{0<s<t}^{m} x_{is} x_{it} x_{js} x_{jt}
\end{aligned}$$

Putting everything together, we have

$$\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)^2 \left(\sum_{s=1}^{m} d_s\right)^2\right] = \mu_4 \sum_{s=1}^{m} x_{is}^2 x_{js}^2 + \sum_{0<s,t}^{m} x_{is}^2 x_{jt}^2 + 4 \sum_{0<s<t}^{m} x_{is} x_{it} x_{js} x_{jt} \tag{2.9}$$

Now, we have

$$\begin{aligned}\left(\mathbb{E}\left[\left(\sum_{s=1}^{m} c_s\right)\left(\sum_{s=1}^{m} d_s\right)\right]\right)^2 &= \left(\sum_{s=1}^{m} x_{is} x_{js}\right)^2 \\ &= \sum_{s=1}^{m} x_{is}^2 x_{js}^2 + 2 \sum_{0<s<t}^{m} x_{is} x_{js} x_{it} x_{jt}\end{aligned}$$

Finally, we have

$$\operatorname{Var}\left[y_{is} y_{js}\right] = \mathbb{E}\left[\left(\sum_{s=1}^{m} r_{1s} x_{is}\right)^2 \left(\sum_{s=1}^{m} r_{1s} x_{js}\right)^2\right] - \left(\mathbb{E}\left[\left(\sum_{s=1}^{m} r_{1s} x_{is}\right)\left(\sum_{s=1}^{m} r_{1s} x_{js}\right)\right]\right)^2 \tag{2.10}$$

$$= 3 \sum_{s=1}^{m} x_{is}^2 x_{js}^2 + \sum_{0<s,t}^{m} x_{is}^2 x_{jt}^2 + 4 \sum_{0<s<t}^{m} x_{is} x_{it} x_{js} x_{jt} - \sum_{s=1}^{m} x_{is}^2 x_{js}^2 - 2 \sum_{0<s<t}^{m} x_{is} x_{js} x_{it} x_{jt} \tag{2.11}$$

$$= 2 \sum_{s=1}^{m} x_{is}^2 x_{js}^2 + \sum_{0<s,t}^{m} x_{is}^2 x_{jt}^2 + 2 \sum_{0<s<t}^{m} x_{is} x_{js} x_{it} x_{jt} \tag{2.12}$$

With k such estimates, we have

$$\operatorname{Var}\left[\frac{y_{i1} y_{j1} + \ldots + y_{ik} y_{jk}}{k}\right] = \frac{2 \sum_{s=1}^{m} x_{is}^2 x_{js}^2 + \sum_{0<s,t}^{m} x_{is}^2 x_{jt}^2 + 2 \sum_{0<s<t}^{m} x_{is} x_{js} x_{it} x_{jt}}{k}$$

In general, the variance of the estimate of the dot product is higher than the variance of the estimate of the squared Euclidean distance.

These expressions on its own are not very meaningful at first glance. However, viewing them "pictorially" helps to give some intuition.

To view such expressions pictorially, we first explain some insight behind squares. Since secondary school, most students may have remembered this formula

$$(x_1 + x_2)^2 = x_1^2 + 2x_1 x_2 + x_2^2$$

and in general

$$(x_1 + \ldots + x_n)^2 = \sum_{s=1}^{n} x_s^2 + 2 \sum_{0<s<t}^{n} x_s x_t$$

Pictorially for $n = 4$, we can visualize this to be the area of the following square.

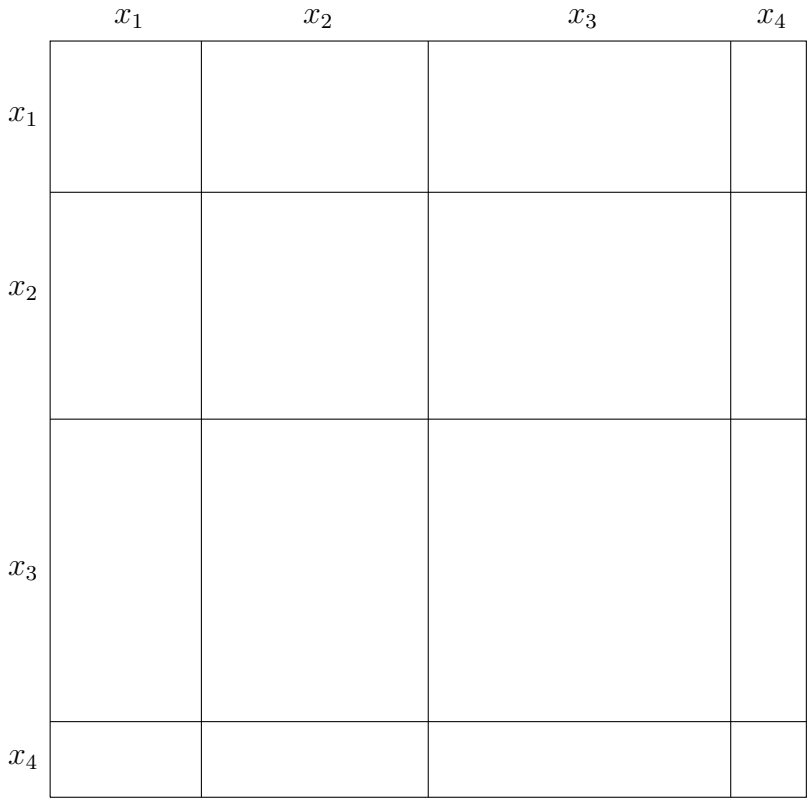

Figure 7: Area of Square for $n = 4$

From Figure 7, we can see that the area of the square is given by the sum of the diagonal squares $(x_1)^2 + (x_2)^2 + (x_3)^2 + (x_4)^2$, as well as the "cross-terms" $2\sum_{0<s<t}^{4} x_s x_t$.

Let us look at the variance of the estimate of the squared Euclidean distance in this context, where our variance is given by

$$\mathrm{Var}[(y_{i1} - y_{j1})^2] = 2\sum_{s=1}^{m}(x_{is} - x_{js})^4 + 4 \sum_{0<s<t}^{m} (x_{is} - x_{js})^2(x_{it} - x_{jt})^2 \tag{2.13}$$

Note that if we set $z_s := x_{is} - x_{js}$, then we have

$$\mathrm{Var}[(y_{i1} - y_{j1})^2] = 2\sum_{s=1}^{m} z_s^4 + 4 \sum_{0<s<t}^{m} z_s^2 z_t^2 \tag{2.14}$$

There's a very nice explanation for this. Suppose $\vec{z} = \vec{x}_i - \vec{x}_j$. We consider the **square** of each term of $\vec{z}$, i.e. $z_1^2, z_2^2, \ldots, z_k^2$. Then the picture we should have in mind is Figure 8.

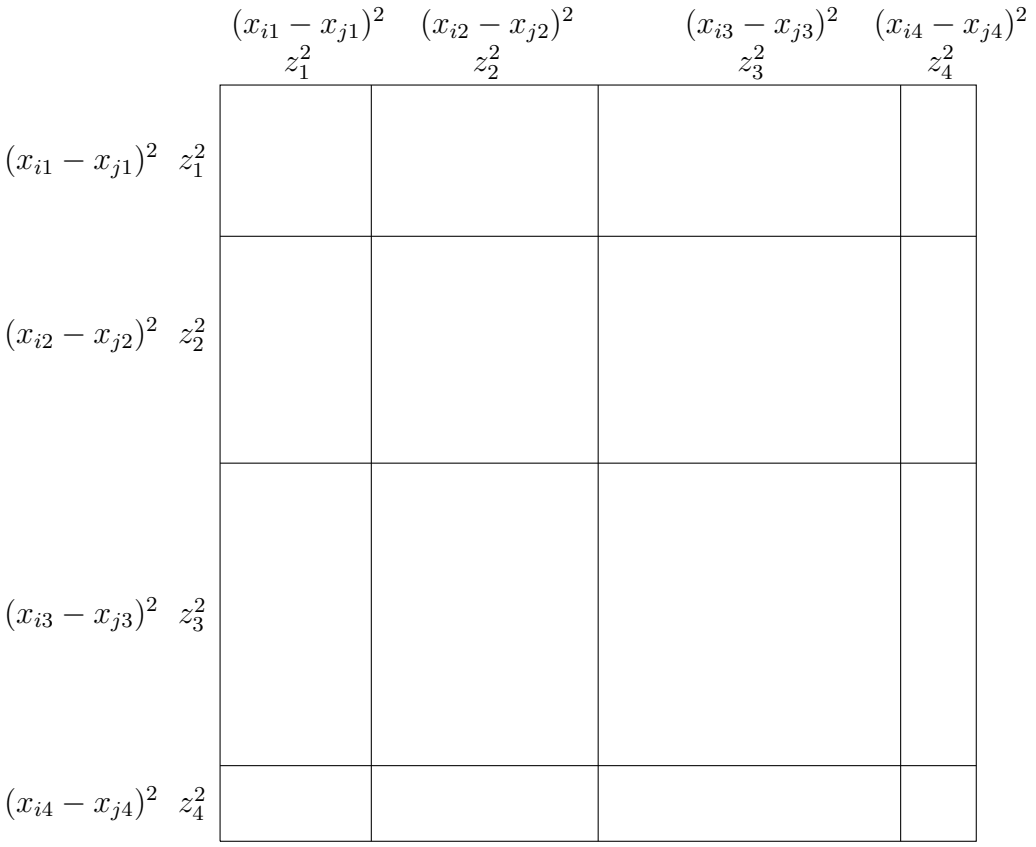

Figure 8: Area of "Euclidean distance" square for $m = 4$

The variance of our squared Euclidean distance estimate is thus twice of the area of the square of sides z_s! In fact, we could even write our squared Euclidean distance variance to be

$$\text{Var}[(y_{i1} - y_{j1})^2] = 2\left(\sum_{s=1}^{m}(x_{is} - x_{js})^2\right)^2 \tag{2.15}$$

We can again do a similar picture for the variance of the estimate for the dot product, and in fact simplify our expression.

From Equation 2.10, we had

$$\text{Var}\left[y_{is}y_{js}\right] = 2\sum_{s=1}^{m} x_{is}^2 x_{js}^2 + \sum_{0<s,t}^{m} x_{is}^2 x_{jt}^2 + 2\sum_{0<s<t}^{m} x_{is}x_{js}x_{it}x_{jt}$$

The trick here is to split the first term into two parts to get

$$\text{Var}\left[y_{is}y_{js}\right] = \sum_{s=1}^{m} x_{is}^2 x_{js}^2 + \sum_{0<s,t}^{m} x_{is}^2 x_{jt}^2 + \sum_{s=1}^{m} x_{is}^2 x_{js}^2 + 2\sum_{0<s<t}^{m} x_{is}x_{js}x_{it}x_{jt}$$

The last two terms in red can be seen as the square with sides $x_{is}x_{js}$, and hence we can write

$$\sum_{s=1}^{m} x_{is}^2 x_{js}^2 + 2\sum_{0<s<t}^{m} x_{is}x_{js}x_{it}x_{jt} = \left(\sum_{s=1}^{m} x_{is}x_{js}\right)^2$$

The first two terms in blue can be seen as the rectangle with one side given by x_{is}^2 and the other side given by x_{js}^2, and hence we can write

$$\sum_{s=1}^{m} x_{is}^2 x_{js}^2 + \sum_{0<s,t}^{m} x_{is}^2 x_{jt}^2 = \left(\sum_{s=1}^{m} x_{is}^2\right)\left(\sum_{s=1}^{m} x_{js}^2\right)$$

and therefore

$$\text{Var}[y_{is}y_{js}] = \left(\sum_{s=1}^{m} x_{is}x_{js}\right)^2 + \left(\sum_{s=1}^{m} x_{is}^2\right)\left(\sum_{s=1}^{m} x_{js}^2\right) \tag{2.16}$$

So we can think of the variance of the estimate given by the dot product as the "sum" of the areas of one square and one rectangle, where the length of the square is the actual dot product, and the lengths of the rectangle are the squared norms of $\vec{x}_i, \vec{x}_j$ respectively.

The motivation for thinking about squares when looking at the variance of the estimates is that it gives some intuition to how the variance is distributed based on given vectors $\vec{x}_i, \vec{x}_j$, and how best to "tinker" with the random projection algorithm to reduce the variance.

2.2.2 Method 2: Computation of variance using Statistics 101 methods

The idea here is to treat the variables $\{y_{is}, y_{js}\}_{s=1}^{k}$ as k draws from some joint distribution.

Recall that each

$$y_{is} = \sum_{t=1}^{m} r_{st}x_{it} \qquad y_{js} = \sum_{t=1}^{m} r_{st}x_{jt}$$

is the sum of m weighted random normal variables, where each r_{st} is i.i.d. $N(0,1)$.

Our first goal is to find the distribution for which each y_{is}, y_{js} come from.

Using moment generating functions (see Appendix B for more details), we know that the MGF of a normal $N(0, \sigma^2)$ is given by

$$\begin{aligned}
M_R(t) &= \mathbb{E}\left[\exp\{tR\}\right] \\
&= \int_{-\infty}^{\infty} \exp\{tr\} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-r^2/2\sigma^2\}\ \mathrm{d}r \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{tr - r^2/2\sigma^2\}\ \mathrm{d}r \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}\left(r^2 - 2t\sigma^2 r\right)\right\}\ \mathrm{d}r \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}\left((r - t\sigma^2)^2 - t^2\sigma^4\right)\right\}\ \mathrm{d}r \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(r - t\sigma^2)^2}{2\sigma^2}\right\} \exp\left\{\frac{t^2\sigma^2}{2}\right\}\ \mathrm{d}r \\
&= \exp\left\{\frac{t^2\sigma^2}{2}\right\} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{(r - t\sigma^2)^2}{2\sigma^2}\right\}\ \mathrm{d}r \\
&= \exp\left\{\frac{t^2\sigma^2}{2}\right\}
\end{aligned} \tag{2.17}$$

where we completed the square inside the exponential, and used the fact that we had the pdf of $N(t\sigma^2, \sigma^2)$ in our resultant expression.

Since $r_{st} \sim N(0, 1)$, we have $M_R = \exp\left\{\frac{t^2}{2}\right\}$.

Suppose we change the indices of the summation

$$y_{is} = \sum_{n=1}^{m} r_{sn} x_{in}$$

since we have a "t" in our MGF.

Then for any term in the summation $r_{sn}x_{in}$, we have that

$$\begin{aligned}
M_{x_{in}R} &= \mathbb{E}\left[\exp\left\{x_{in}Rt\right\}\right] \\
&= \mathbb{E}\left[\exp\left\{uR\right\}\right] \quad \text{set } u = x_{in}t \\
&= \exp\left\{\frac{u^2}{2}\right\} \qquad \text{from 2.17} \\
&= \exp\left\{\frac{x_{in}^2 t^2}{2}\right\}
\end{aligned}$$

which implies that each weighted term is distributed i.i.d. $N(0, x_{in}^2)$.

We can hence write down the MGF of our weighted sum to be

$$\begin{aligned} M_Y(t) &= M_{x_{i1}R+\ldots+x_{im}R} \\ &= \mathbb{E}\left[\exp\left\{t\left(x_{i1}R + \ldots + x_{im}R\right)\right\}\right] \\ &= \mathbb{E}\left[\exp\left\{x_{i1}Rt\right\} \cdot \cdots \cdot \exp\left\{x_{im}Rt\right\}\right] \\ &= M_{x_{i1}R}(t) M_{x_{i2}R}(t) \ldots M_{x_{im}R}(t) \\ &= \exp\left\{\frac{x_{i1}^2 t^2}{2}\right\} \cdot \cdots \cdot \exp\left\{\frac{x_{im}^2 t^2}{2}\right\} \\ &= \exp\left\{\frac{\|\vec{x}_i\|_2^2 t^2}{2}\right\} \end{aligned}$$

which implies that each y_{is} is distributed $N(0, \|\vec{x}_i\|_2^2)$, for $1 \leq s \leq k$.

Therefore, we have that

$$\begin{aligned} y_{is} &\sim N(0, \|\vec{x}_i\|_2^2) \\ y_{js} &\sim N(0, \|\vec{x}_j\|_2^2) \\ y_{is} - y_{js} &\sim N(0, \|\vec{x}_i - \vec{x}_j\|_2^2) \end{aligned}$$

With these distributions, we are ready to find the variance of the squared Euclidean distance

$$\begin{aligned} \operatorname{Var}\left[\left\{\sum_{t=1}^{m} r_{1t}(x_{it} - x_{jt})\right\}^2\right] &= \operatorname{Var}\left[(y_{i1} - y_{j1})^2\right] \\ &= \mathbb{E}\left[(y_{i1} - y_{j1})^4\right] - \left(\mathbb{E}\left[(y_{i1} - y_{j1})^2\right]\right)^2 \end{aligned}$$

which necessitates us finding the second and fourth moment of $y_{is} - y_{js}$. To do so, we find the second and fourth derivative of our MGF for $y_{is} - y_{js}$, and evaluate it at $t = 0$.

Setting $M_X(t) = \exp\left\{\frac{kt^2}{2}\right\}$ where $k = \|\vec{x}_i - \vec{x}_j\|_2^2$, we have

$$\begin{aligned} M_X^{(1)}(t) &= kt \exp\left\{\frac{kt^2}{2}\right\} \\ M_X^{(2)}(t) &= k\left(1 + kt^2\right) \exp\left\{\frac{kt^2}{2}\right\} \\ M_X^{(3)}(t) &= k^2 t\left(3 + kt^2\right) \exp\left\{\frac{kt^2}{2}\right\} \\ M_X^{(4)}(t) &= k^2\left(3 + 6kt^2 + k^2 t^4\right) \exp\left\{\frac{kt^2}{2}\right\} \end{aligned}$$

Hence, we can write

$$\begin{aligned}
\text{Var}\left[\left\{\sum_{t=1}^{m} r_{1t}(x_{it} - x_{jt})\right\}^2\right] &= \text{Var}\left[(y_{i1} - y_{j1})^2\right] \\
&= \mathbb{E}\left[(y_{i1} - y_{j1})^4\right] - \left(\mathbb{E}\left[(y_{i1} - y_{j1})^2\right]\right)^2 \\
&= M_X^{(4)}(0) - M_X^{(2)}(0)^2 \\
&= 3k^2 - k^2 \\
&= 2k^2 \\
&= 2\left(\|\vec{x}_i - \vec{x}_j\|_2^2\right)^2
\end{aligned}$$

which is exactly the same as Equation 2.15.

Let's try to get an expression for the variance of the inner product estimate. We note that each y_{is} is correlated to y_{js}, since we used the same random variables $r_{s.}$ to generate them, and hence we cannot use MGFs directly. [Li et al., 2006a] has a nice derivation of the variance, and we will prove this along similar lines.

We will use the following two theorems from [Mardia et al., 1979] to help us. The following two theorems are Theorem 3.2.3 and Theorem 3.2.4 from [Mardia et al., 1979] respectively which we give without proof.

Suppose we are given a vector $\vec{x} \in \mathbb{R}^p$, and we partition $\vec{x}$ into two subvectors with r and s elements respectively.

Theorem 2.1. If $\vec{x} = (\vec{x}_1, \vec{x}_2)' \sim N_p(\vec{\mu}, \Sigma)$, then $\vec{x}_1$ and $\vec{x}_{2.1} = \vec{x}_2 - \Sigma_{21}\Sigma_{11}^{-1}\vec{x}_1$ have the following distributions and are statistically independent

$$\vec{x}_1 \sim N_r(\vec{\mu_1}, \Sigma_{11}) \qquad \vec{x}_{2.1} \sim N_s(\vec{\mu}_{2.1}, \Sigma_{22.1})$$

where

$$\begin{aligned}
\vec{\mu}_{2.1} &= \vec{\mu}_2 - \Sigma_{21}\Sigma_{11}^{-1}\vec{\mu}_1 \\
\Sigma_{22.1} &= \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}
\end{aligned}$$

Theorem 2.2. Using the assumptions and notations of the previous theorem, the conditional distribution of $\vec{x}_2$ for a given value of $\vec{x}_1$ is

$$\vec{x}_2 \mid \vec{x}_1 \sim N_s(\vec{\mu}_2 + \Sigma_{21}\Sigma_{11}^{-1}(\vec{x}_1 - \vec{\mu}_1), \Sigma_{22.1})$$

Hence we can think of $\{y_{is}, y_{js}\}_{s=1}^{k}$ as draws from a bivariate normal distribution, with

$$(y_{i.}, y_{j.}) \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma\right) \tag{2.18}$$

The goal is to find the elements of Σ. We know that $\Sigma_{11} = \|\vec{x}_i\|_2^2$, and $\Sigma_{22} = \|\vec{x}_j\|_2^2$.

$\Sigma_{12} = \Sigma_{21}$ is given by $\text{Cov}[y_{i\cdot}, y_{j\cdot}]$, and we can write this as

$$\begin{aligned}\text{Cov}[y_{i\cdot}, y_{j\cdot}] &= \mathbb{E}[y_{i\cdot}y_{j\cdot}] - \mathbb{E}[y_{i\cdot}, y_{j\cdot}] \\ &= \vec{x}_i^T\vec{x}_j\end{aligned}$$

Therefore we have

$$(y_{i\cdot}, y_{j\cdot}) \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \|\vec{x}_i\|_2^2 & \vec{x}_i^T\vec{x}_j \\ \vec{x}_i^T\vec{x}_j & \|\vec{x}_j\|_2^2 \end{pmatrix}\right) \tag{2.19}$$

We can now write

$$y_{j\cdot} \mid y_{i\cdot} \sim N\left(\frac{\vec{x}_i^T\vec{x}_j}{\|\vec{x}_i\|_2^2}y_{i\cdot}, \|\vec{x}_j\|_2^2 - \frac{(\vec{x}_i^T\vec{x}_j)^2}{\|\vec{x}_i\|_2^2}\right)$$

The motivation for finding the conditional distribution of $y_{j\cdot} \mid y_{i\cdot}$ is to allow us to compute the expectation

$$\mathbb{E}[\mathbb{E}[y_{i\cdot}^2 y_{j\cdot}^2 \mid y_{i\cdot}]]$$

We can do this again with moment generating functions. We can derive the MGF of a normally distributed variable $N(\mu, \sigma^2)$ to be

$$M_X(t) = \exp\{\mu t + \sigma^2 t^2/2\}$$

which is a simple exercise from 2.17, and its second derivative as well as fourth derivative evaluated at $t = 0$ given by

$$\begin{aligned}M_X^{(2)}(0) &= \mu^2 + \sigma^2 \\ M_X^{(4)}(0) &= \mu^4 + 6\mu^2\sigma^2 + 3\sigma^4\end{aligned}$$

Therefore, we have

$$\begin{aligned}\mathbb{E}[\mathbb{E}[y_{i\cdot}^2 y_{j\cdot}^2 \mid y_{i\cdot}]] &= \mathbb{E}\left[y_{i\cdot}^2\left(\left(\frac{\vec{x}_i^T\vec{x}_j}{\|\vec{x}_i\|_2^2}y_{i\cdot}\right)^2 + \|\vec{x}_j\|_2^2 - \frac{(\vec{x}_i^T\vec{x}_j)^2}{\|\vec{x}_i\|_2^2}\right)\right] \\ &= \mathbb{E}\left[\left(\left(\frac{(\vec{x}_i^T\vec{x}_j)^2}{\|\vec{x}_i\|_2^4}\right)y_{i\cdot}^4 + y_{i\cdot}^2\|\vec{x}_j\|_2^2 - y_{i\cdot}^2\frac{(\vec{x}_i^T\vec{x}_j)^2}{\|\vec{x}_i\|_2^2}\right)\right] \\ &= 3\|\vec{x}_i\|_2^4\left(\frac{(\vec{x}_i^T\vec{x}_j)^2}{\|\vec{x}_i\|_2^4}\right) + \|\vec{x}_i\|_2^2\|\vec{x}_j\|_2^2 - \|\vec{x}_i\|_2^2\frac{(\vec{x}_i^T\vec{x}_j)^2}{\|\vec{x}_i\|_2^2} \\ &= 3(\vec{x}_i^T\vec{x}_j)^2 + \|\vec{x}_i\|_2^2\|\vec{x}_j\|_2^2 - (\vec{x}_i^T\vec{x}_j)^2 \\ &= 2(\vec{x}_i^T\vec{x}_j)^2 + \|\vec{x}_i\|_2^2\|\vec{x}_j\|_2^2\end{aligned}$$

Now it suffices to check that

$$\begin{aligned}\text{Var}[y_{i\cdot}y_{j\cdot}] &= \mathbb{E}\left[y_{i\cdot}^2 y_{j\cdot}^2\right] - \left(\mathbb{E}[y_{i\cdot}y_{j\cdot}]\right)^2 \\ &= 2(\vec{x}_i^T\vec{x}_j)^2 + \|\vec{x}_i\|_2^2\|\vec{x}_j\|_2^2 - (\vec{x}_i^T\vec{x}_j)^2 \\ &= (\vec{x}_i^T\vec{x}_j)^2 + \|\vec{x}_i\|_2^2\|\vec{x}_j\|_2^2\end{aligned}$$

which matches Equation 2.16 exactly.

While the second method (using MGFs) require a bit more conceptual understanding of what MGFs are, they simplified the calculation of variances by quite a bit, which would prevent any numerical mistakes.

However, the "geometric" (square) intuition as well as the weights put on elements of the square is not captured with the MGF method.

2.2.3 Interlude: Verifying Theory

Computing the variances of the estimates of the Euclidean distance and the dot product can be exciting, but we also want to know if our theoretical variances are correct.

They should be[5], but it always does no harm to run experiments and check that the empirical variance matches the theoretical variance.

Here, we similarly use the `Colon` dataset [Alon et al., 1999], and pick two vectors from this dataset using a fixed random seed of `rng(0)`. We then compute the estimate of the squared Euclidean distance as well as the estimate of the dot product for these vectors when we project the data down to $\mathbb{R}^k$ for $k \in \{1, 2, \ldots, 100\}$. We repeat this 1000 times, and take the variance of the estimates. The code for doing so is in `script_to_try.m`.

Figures 9 and 10 show our plots of the theoretical variance and empirical variance of both the squared Euclidean distance and the dot product.

We make two observations.

1. The empirical variance matches the theoretical variance quite well, since the two lines hardly deviate from each other (except at low k).
2. The scale of the variances is huge at about 3×10^{18}. This doesn't mean that our estimates are bad, but rather we should look at the **relative error** (or center the data to have length of 1 and then compute the respective variances).

[5] and in general can be checked these days with Mathematica.

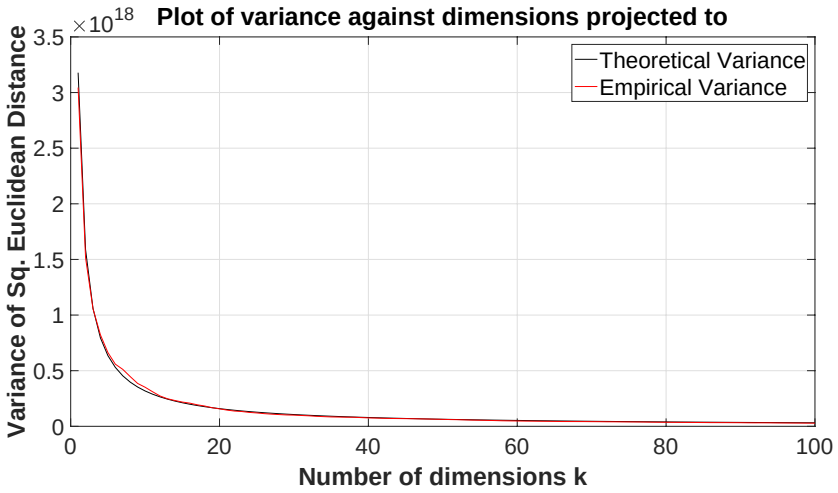

Figure 9: Plot of theoretical variance and empirical variance for squared Euclidean distance product estimate of two vectors in the Colon Dataset against dimension projected to for 1000 iterations for $k \in \{1, 2, \ldots, 100\}$

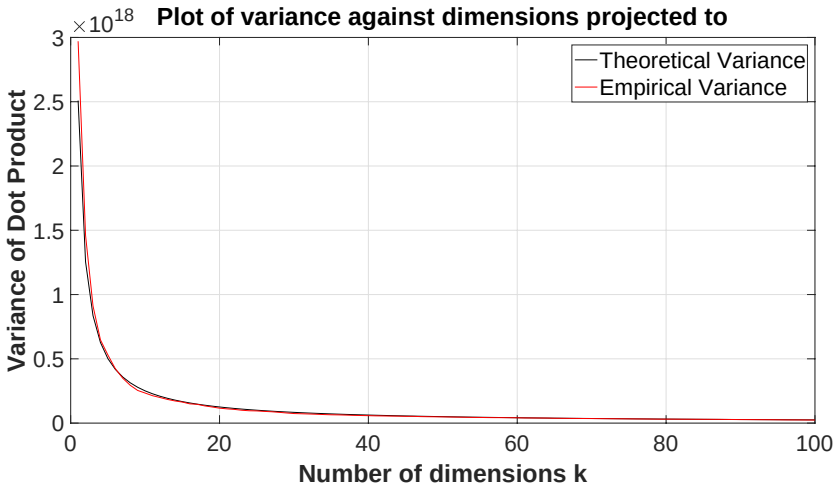

Figure 10: Plot of theoretical variance and empirical variance for dot product estimate of two vectors in the Colon Dataset against dimension projected to for 1000 iterations for $k \in \{1, 2, \ldots, 100\}$.

The motivation is that if we are measuring something that is relatively small, say an ant, an error of 1 metre would be catastrophic. But if we are measuring something huge, say the Death Star[6], then an error of 1 metre would be extremely small. Relative size matters!

2.2.4 Method 3: Computation of probability bounds (and the JL Lemma)

Finally, we look at a third method that bounds the errors of our squared Euclidean distance and dot product estimates. This is also where the Johnson-Lindenstrauss (JL) Lemma of [Johnson et al., 1986] fame is usually mentioned.

Here is a version of the JL Lemma from [Vempala, 2004].

Lemma 2.1. For any ϵ such that $\frac{1}{2} > \epsilon > 0$, and any set of points S in $\mathbb{R}^n$ with $|S| = m$, upon projection to a uniform random k- dimensional subspace where $k \geq \frac{9 \log m}{\epsilon^2 - \frac{2}{3}\epsilon^3} + 1$ the following property holds: with probability at least $\frac{1}{2}$ for every pair $\vec{u}, \vec{u'} \in S$

$$(1-\epsilon)\|\vec{u} - \vec{u'}\|_2^2 \leq \|f(\vec{u}) - f(\vec{u'})\|_2^2 \leq (1+\epsilon)\|\vec{u} - \vec{u'}\|_2^2$$

where $f(\vec{u}), f(\vec{u'})$ are the projections of $\vec{u}, \vec{u'}$.

There are several variants of the proof [Johnson et al., 1986, Frankl and Maehara, 1988, Indyk and Motwani, 1998, Achlioptas, 2003]of the JL Lemma which will be omitted in this document.

The JL Lemma tells us that points in a high dimensional space can be projected down (and represented in some) lower dimensional space, where distances between these points are preserved with some probability. This hints at the fact that given some set of high dimensional data, we can probably find a low dimensional space (or manifold) to represent the data.

Moreover, the term $\frac{9 \log m}{\epsilon^2 - \frac{2}{3}\epsilon^3}$ is independent of n, the initial dimension of these high dimensional points. In fact, the number of dimensions k to project to is only related to m, which is the number of points in the high dimensional space.

Figure 11 gives some intuition on the minimal value of k needed when we want to control the error ϵ. For example, if we want to be sure that our projected squared Euclidean distance is within 1 ± 0.1 of the true squared Euclidean distance, then if we had 100,000 observations, we can find a low dimensional subspace of $k = 1000$ to project our data onto. This is regardless of whether we had 100,000 observations in $\mathbb{R}^{1000}, \mathbb{R}^{100000}$, or even $\mathbb{R}^{10^{10^{10}}}$.

[6] This is about 160 km in width and 120 km in radius.

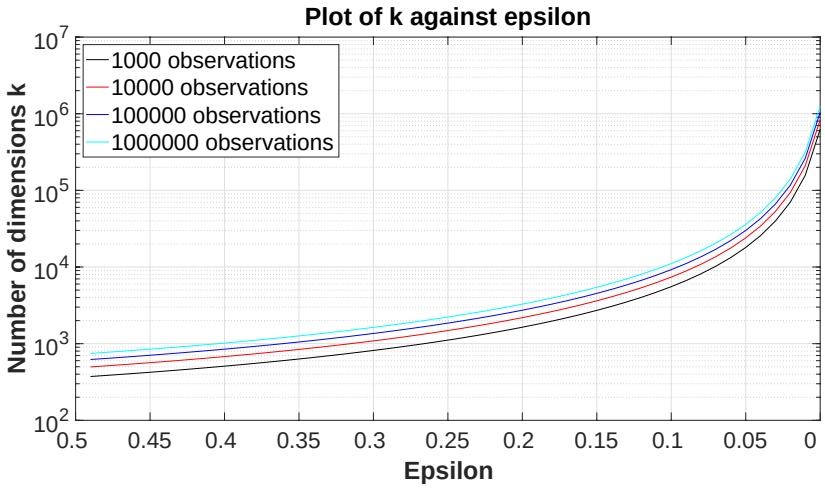

Figure 11: Plot of minimal number of dimensions k required against ϵ in the JL Lemma.

Here is a lemma from [Vempala, 2004] which looks the probability bounds on the error of the estimated norm of a vector after random projection.

Lemma 2.2. Let each entry of an $n \times k$ matrix R be chosen independently from $N(0,1)$. Let $\vec{v} = \frac{1}{\sqrt{k}} R^T \vec{u}$ for $\vec{u} \in \mathbb{R}^n$. Then for any $\epsilon > 0$

- $\mathbb{E}\left[\|\vec{v}\|_2^2\right] = \|\vec{u}\|_2^2$
- $\mathbb{P}\left[\|\vec{v}\|_2^2 - \|\vec{u}\|_2^2 \geq \epsilon \|\vec{u}\|_2^2\right] < 2\exp\left\{-\frac{k(\epsilon^2-\epsilon^3)}{4}\right\}$

The proof of this lemma is given in [Vempala, 2004], hence we will omit this. However, we note the following items.

1. The notation here is slightly different - as the matrix-vector product $R^T\vec{u}$ is scaled by the scaling factor of $\frac{1}{\sqrt{k}}$, which is equivalent to dividing our observations in 2.3 and 2.4 by k. This is numerically simpler too, since we do not need to compute an additional $\sqrt{k}$ for the scaling factor.

2. The proof relies on finding the bounds

$$\mathbb{P}\left[\|\vec{v}\|_2^2 \geq (1+\epsilon)\|\vec{u}\|_2^2\right] < \exp\left\{-\frac{k(\epsilon^2-\epsilon^3)}{4}\right\}$$

and

$$\mathbb{P}\left[\|\vec{v}\|_2^2 \leq (1-\epsilon)\|\vec{u}\|_2^2\right] < \exp\left\{-\frac{k(\epsilon^2-\epsilon^3)}{4}\right\}$$

which we state here.

While Lemma 2.2 from [Vempala, 2004] looks at bounds on the norms, this also gives us the bounds for the estimated squared Euclidean distance and the estimated inner product between two vectors $\vec{x}_i, \vec{x}_j$.

It is easy to see that we have the bounds of the estimated square Euclidean distance to be

$$\mathbb{P}\left[\|\vec{v}_i - \vec{v}_j\|_2^2 - \|\vec{u}_i - \vec{u}_j\|_2^2 \geq \epsilon\|\vec{u}_i - \vec{u}_j\|_2^2\right] < 2\exp\left\{-\frac{k(\epsilon^2 - \epsilon^3)}{4}\right\} \tag{2.20}$$

by replacing $\vec{v}, \vec{u}$ by $\vec{v}_i - \vec{v}_j$ and $\vec{u}_i - \vec{u}_j$ respectively.

To get the bounds of the estimated inner product, we have to do a bit more work. We want to get some expression

$$\mathbb{P}\left[|\vec{v}_i^T\vec{v}_j| \geq \epsilon\vec{x}_i^T\vec{x}_j\right] < f(\epsilon, k)$$

We can use the same idea in 2.20 to get

$$\begin{aligned}
\mathbb{P}\left[\|\vec{v}_i - \vec{v}_j\|_2^2 \geq (1+\epsilon)\|\vec{u}_i - \vec{u}_j\|_2^2\right] &< \exp\left\{-\frac{k(\epsilon^2 - \epsilon^3)}{4}\right\} \\
\mathbb{P}\left[\|\vec{v}_i + \vec{v}_j\|_2^2 \geq (1+\epsilon)\|\vec{u}_i + \vec{u}_j\|_2^2\right] &< \exp\left\{-\frac{k(\epsilon^2 - \epsilon^3)}{4}\right\} \\
\mathbb{P}\left[\|\vec{v}_i - \vec{v}_j\|_2^2 \leq (1-\epsilon)\|\vec{u}_i - \vec{u}_j\|_2^2\right] &< \exp\left\{-\frac{k(\epsilon^2 - \epsilon^3)}{4}\right\} \\
\mathbb{P}\left[\|\vec{v}_i + \vec{v}_j\|_2^2 \leq (1-\epsilon)\|\vec{u}_i + \vec{u}_j\|_2^2\right] &< \exp\left\{-\frac{k(\epsilon^2 - \epsilon^3)}{4}\right\}
\end{aligned}$$

Noting the fact that

$$\|\vec{v}_i + \vec{v}_j\|_2^2 - \|\vec{v}_i - \vec{v}_j\|_2^2 = 4\vec{v}_i^T\vec{v}_j$$

then we can write

$$\mathbb{P}\left[\frac{1}{4}\left(\|\vec{v}_i + \vec{v}_j\|_2^2 - \|\vec{v}_i - \vec{v}_j\|_2^2\right) \geq \frac{(1+\epsilon)}{4}\left(\|\vec{u}_i + \vec{u}_j\|_2^2 - \|\vec{u}_i - \vec{u}_j\|_2^2\right)\right] < 2\exp\left\{-\frac{k(\epsilon^2 - \epsilon^3)}{4}\right\}$$

and

$$\mathbb{P}\left[\frac{1}{4}\left(\|\vec{v}_i + \vec{v}_j\|_2^2 - \|\vec{v}_i - \vec{v}_j\|_2^2\right) \leq \frac{(1-\epsilon)}{4}\left(\|\vec{u}_i + \vec{u}_j\|_2^2 - \|\vec{u}_i - \vec{u}_j\|_2^2\right)\right] < 2\exp\left\{-\frac{k(\epsilon^2 - \epsilon^3)}{4}\right\}$$

and combine both to get

$$\mathbb{P}\left[|\vec{v}_i^T\vec{v}_j| \geq \epsilon\vec{x}_i^T\vec{x}_j\right] < 4\exp\left\{-\frac{k(\epsilon^2 - \epsilon^3)}{4}\right\}$$

With these bounds for the estimated squared Euclidean distance, and estimated dot product between any two vectors, we get some idea of the probability that our estimates are within some error ϵ with the value k chosen for our random projections.

It is useful to look at some plots of this probability that our estimates are within $(1 \pm \epsilon)$ versus ϵ for varying values of k.

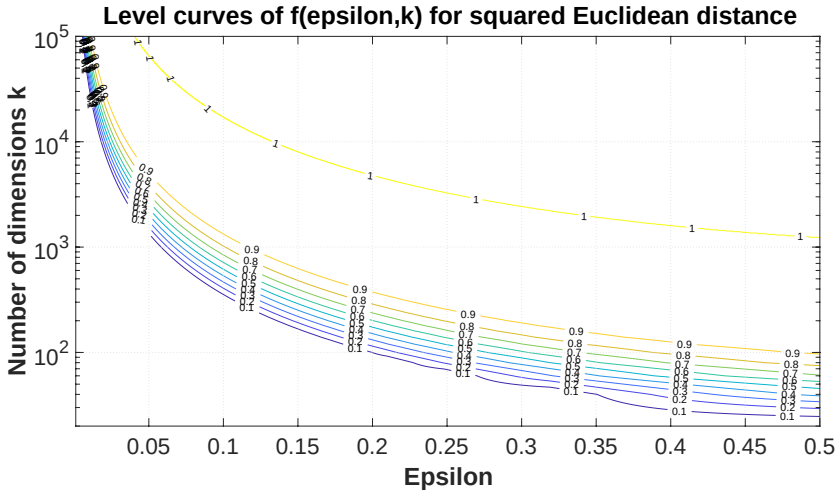

Figure 12: Plot of level curves for probability bounds in the estimation of the squared Euclidean distance under random projections where $f(\epsilon, k) = 2\exp\left\{-\frac{k(\epsilon^2-\epsilon^3)}{4}\right\}$.

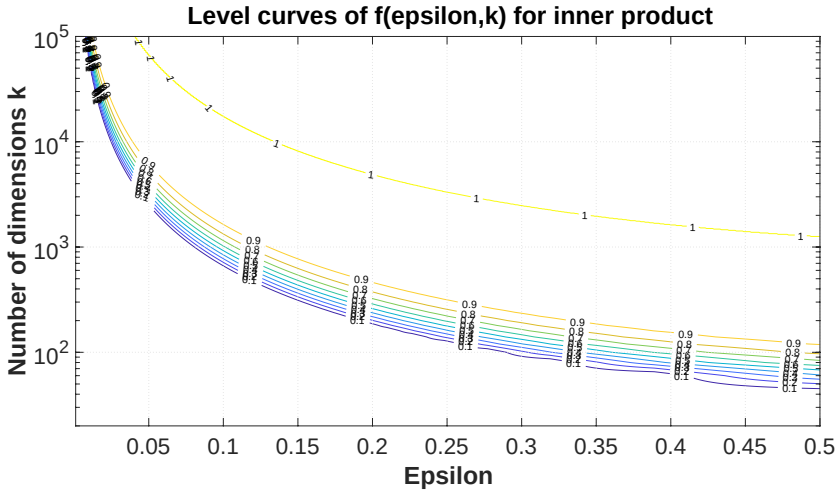

Figure 13: Plot of level curves for probability bounds in the estimation of the inner product under random projections where $f(\epsilon, k) = 4\exp\left\{-\frac{k(\epsilon^2-\epsilon^3)}{4}\right\}$.

In general, we see that in order to have our estimates be within (low) ϵ with some high probability, we necessarily need to increase the number of dimensions k.

It is also useful to see how well our bounds work in practice.

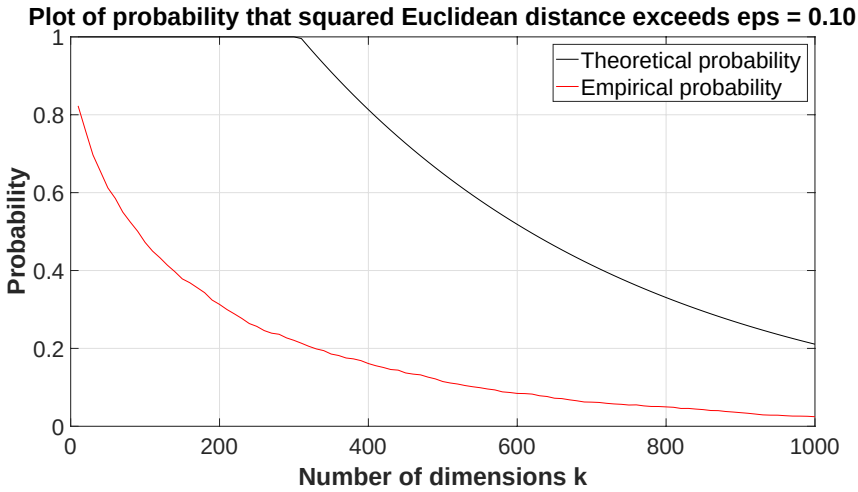

Figure 14: Plot of empirical proportion of estimate of squared Euclidean distance that is away from $(0.9, 1.1)$ of true Euclidean distance between observations 51 and 57 of Colon data compared with theoretical proportion for 10000 iterations.

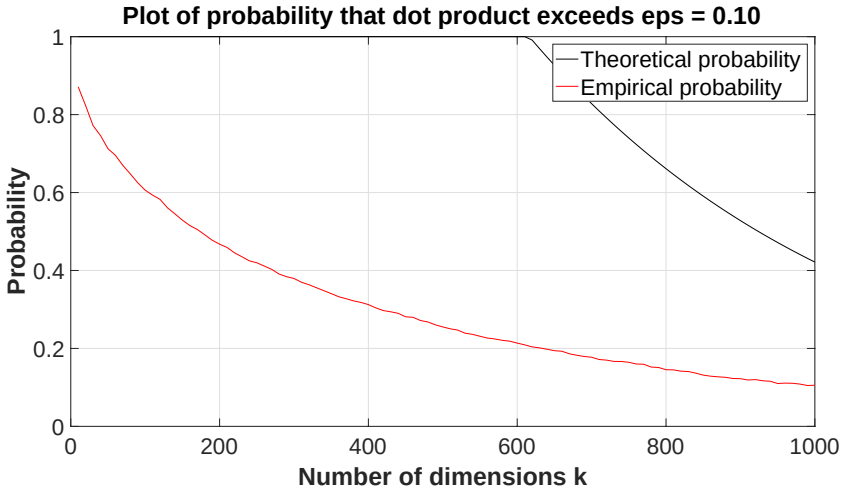

Figure 15: Plot of empirical proportion of estimate of dot product that is away from $(0.9, 1.1)$ of true dot product between observations 51 and 57 of Colon data compared with theoretical proportion for 10000 iterations.

From Figure 14 and 15 we see that the bounds given by Lemma 2.2 are not as tight for low values of k, as our empirical observations have fewer observations outside of our $(1-\epsilon, 1+\epsilon)$ range of our true values.

2.3 Summary and Conclusion

In this section, we looked at how random projections with entries i.i.d. $N(0, 1)$ can be used to compute an estimate of the squared Euclidean distance and dot product between vector

pairs. Moreover, we looked at three methods to quantify some "accuracy" of these estimates, and plotted some graphs to verify our theory.

Usually, we pick the most elegant method to solve a problem, and indeed, Method 2 and Method 3 seem to be more elegant compared to Method 1.

However, when I come up with new ideas, I try to look at different methods even though there might be a more efficient one, since different methods have different ideas behind them. Some ideas lead nowhere, but no one knows that in advance.

The analogy I like to think of regarding this stems from when I was working in Boston. My workplace was near Downtown Crossing, and I lived at Sanger Street in Somerville. The most efficient way for me to go back after work would be to take the orange line from Downtown Crossing to Sullivan Square, before taking a bus to Magoun Square. Of course I didn't always do that, because it would be boring. In fact, sometimes I:

1. Decided to walk the way back from Sullivan Square instead of taking the bus - there's a nice ice-cream shop along the way.
2. Took the Red Line to Davis Square - sometimes I caught shows at the Somerville theatre, otherwise I would check out the restaurants when walking back - there were pretty decent Mexican and Tibetan restaurants on the way back.
3. Took the Green Line to Lechmere - there were usually free drinks at the Museum of Science events - and then a bus all the way up to Magoun Square.

While taking the Red and Green line added to my travel time, I discovered more of Boston and grew to appreciate the city more :-)

The same holds for research.

In **Method 1**, we computed the variance of our estimated squared Euclidean distance and dot product by examining individual elements of vectors. While we may have had some complicated expressions, expressing these expressions in terms of the fourth moment μ_4 as in Equations 2.7 and 2.9 can given some insight on how to reduce the variance further.

For example, the normal distribution $N(0, 1)$ has fourth moment $\mu_4 = 3$. If another probability distribution can be picked with $\mu_2 = 1$, but with lower fourth moment μ_4, then the variance can be reduced further. In fact, [Achlioptas, 2003] used the Rademacher distribution with $\mu_4 = 1$ to further improve random projections.

Thinking about the variances in terms of putting weight on "squares" as on Page 24 can also lead to construction of certain type of random projection matrices based on the type of

data given, or more specifically, the range of each parameter. [Kang and Hooker, 2016] was the first paper I wrote which explored this.

Method 2 on the other hand, looked at statistical techniques like moment generating functions to compute the variance of the estimates. To do so, the entries of the matrix V were seen as entries from a bivariate normal distribution. [Li et al., 2006a] built on this and used a maximum likelihood estimator to further improve the estimates of the dot product. In fact, thinking about statistical techniques and the bivariate normal opens up a lot of doors to improving random projections, leading to papers such as [Kang and Hooker, 2017b, Kang and Hooker, 2017a, Kang, 2017b, Kang, 2017a].

Finally, **Method 3** looks at computing probability bounds. While we omitted the proof of Lemma 2.2 in [Vempala, 2004], the techniques behind the proof are useful in works such as [Li et al., 2006b] and more recently [Kaban, 2015].

You can see that while any of the three methods can be used to quantify the error of random projections, each of these methods spur further research in different areas, by dint of the techniques and ideas in these methods.

We explore some of these research in subsequent sections.

3 Random Projections with the Bivariate Normal

Recall that when we do the matrix multiplication $R\vec{x}_i$ and $R\vec{x}_j$, we get

$$\begin{pmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \dots & \dots & \ddots & \dots \\ r_{k1} & r_{k2} & \dots & r_{km} \end{pmatrix} \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{im} \end{pmatrix} = \begin{pmatrix} r_{11}x_{i1} + \ldots + r_{1m}x_{im} \\ r_{21}x_{i1} + \ldots + r_{2m}x_{im} \\ \vdots \\ r_{k1}x_{i1} + \ldots + r_{km}x_{im} \end{pmatrix} = \begin{pmatrix} \sum_{s=1}^{m} r_{1s}x_{is} \\ \sum_{s=1}^{m} r_{2s}x_{is} \\ \vdots \\ \sum_{s=1}^{m} r_{ks}x_{is} \end{pmatrix} = \begin{pmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{ik} \end{pmatrix}$$

and

$$\begin{pmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \dots & \dots & \ddots & \dots \\ r_{k1} & r_{k2} & \dots & r_{km} \end{pmatrix} \begin{pmatrix} x_{j1} \\ x_{j2} \\ \vdots \\ x_{jm} \end{pmatrix} = \begin{pmatrix} r_{11}x_{j1} + \ldots + r_{1m}x_{jm} \\ r_{21}x_{j1} + \ldots + r_{2m}x_{jm} \\ \vdots \\ r_{k1}x_{j1} + \ldots + r_{km}x_{jm} \end{pmatrix} = \begin{pmatrix} \sum_{s=1}^{m} r_{1s}x_{js} \\ \sum_{s=1}^{m} r_{2s}x_{js} \\ \vdots \\ \sum_{s=1}^{m} r_{ks}x_{js} \end{pmatrix} = \begin{pmatrix} y_{j1} \\ y_{j2} \\ \vdots \\ y_{jk} \end{pmatrix}$$

Moreover, from Equation 2.19, we have that each tuple $\{(y_{is}, y_{js})\}_{s=1}^{k}$ is distributed as a bivariate normal

$$(y_{i.}, y_{j.}) \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma\right) \tag{3.1}$$

where

$$\Sigma = \begin{pmatrix} \|\vec{x}_i\|_2^2 & \vec{x}_i^T\vec{x}_j \\ \vec{x}_i^T\vec{x}_j & \|\vec{x}_j\|_2^2 \end{pmatrix}$$

The goal here is to estimate the value of Σ_{12}, i.e. the dot product $\vec{x}_i^T\vec{x}_j$. We can appeal to several techniques in computational and mathematical statistics to do so, such as

1. using a maximum likelihood estimator [Li et al., 2006a]
2. using control variates [Kang and Hooker, 2017b, Kang and Hooker, 2017a, Kang, 2017b]
3. using Bayesian statistics with a prior [Kang, 2017a]

The idea I got for my research was inspired by [Li et al., 2006a], together with my time as a TA for BTRY 6520 and BTRY 3520 in Cornell University, where I juxtaposed computational statistics with estimating Σ_{12} of the bivariate normal.

This is not necessarily new. For example, [Fosdick and Raftery, 2012] looked at estimating the correlation coefficient of small sample sizes, which is in essence the same problem.

However, the difference is this: With random projections, we are dealing with many observations. For example, we might have 1,000,000 webpages, and might be interested in

the pairwise similarities of all of them, which would be about 499999500000 pairs. In our case, we would be estimating 499999500000 pairwise similarities (or correlation coefficients). On the other hand, [Fosdick and Raftery, 2012] looks at data which is extremely difficult to sample (hence small sample size), and only has to estimate **one** correlation coefficient.

Hence, techniques which sacrifice speed for accuracy in [Fosdick and Raftery, 2012] cannot be used in our case, where speed is still essential.

3.1 Improving Random Projections with Marginal Information

We will review the key idea in [Li et al., 2006a], which makes use of a maximum likelihood estimator. More information about maximum likelihood estimators can be found in the appendix on Page 43.

To do so, let us take a look at the pdf of a bivariate normal distribution $N(\vec{\mu}, \Sigma)$, which is given by

$$f(\vec{y}) = \frac{1}{2\pi \left(\det \Sigma\right)^{\frac{1}{2}}} \exp\left\{-\frac{(\vec{y}-\vec{\mu})^T \Sigma^{-1} (\vec{y}-\vec{\mu})}{2}\right\}$$

Our tuples $\{(y_{is}, y_{js})\}_{s=1}^{k}$ after random projection are distributed as i.i.d. bivariate normals

$$(y_{i.}, y_{j.}) \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma\right) \tag{3.2}$$

where

$$\Sigma = \begin{pmatrix} \|\vec{x}_i\|_2^2 & \vec{x}_i^T \vec{x}_j \\ \vec{x}_i^T \vec{x}_j & \|\vec{x}_j\|_2^2 \end{pmatrix}$$

and we use Li's notation to write

$$\Sigma = \begin{pmatrix} m_1 & a \\ a & m_2 \end{pmatrix}$$

Usually, when we want to estimate the parameters of a bivariate normal, we intend to estimate the values of $\vec{\mu}, \Sigma$. However, since we know that $\mu_1 = \mu_2 = 0$, at first glance we want to estimate Σ, or rather, the entries of Σ. Hence, after observing our k tuples $\{(y_{is}, y_{js})\}_{s=1}^{k}$, we can write our likelihood function as

$$\begin{aligned} L(m_1, m_2, a) &= \prod_{s=1}^{k} \frac{1}{2\pi \left(\det \Sigma\right)^{\frac{1}{2}}} \exp\left\{-\frac{(\vec{y}_s)^T \Sigma^{-1} (\vec{y}_s)}{2}\right\} \\ &= \left(\frac{1}{2\pi \left(\det \Sigma\right)^{\frac{1}{2}}}\right)^k \prod_{s=1}^{k} \exp\left\{-\frac{(\vec{y}_s)^T \Sigma^{-1} (\vec{y}_s)}{2}\right\} \end{aligned}$$

and log-likelihood function as

$$\begin{aligned} l(m_1, m_2, a) &= -k\log(2\pi) - \frac{k}{2}\log\left(\det(\Sigma)\right) - \frac{1}{2}\sum_{s=1}^{k}(\vec{y}_s)^T\Sigma^{-1}(\vec{y}_s) \\ &= -k\log(2\pi) - \frac{k}{2}\log\left(m_1m_2 - a^2\right) - \frac{1}{2(m_1m_2 - a^2)}\sum_{s=1}^{k}(y_{1s}\ y_{2s})\begin{pmatrix} m_2 & -a \\ -a & m_1 \end{pmatrix}\begin{pmatrix} y_{1s} \\ y_{2s} \end{pmatrix} \\ &= -k\log(2\pi) - \frac{k}{2}\log\left(m_1m_2 - a^2\right) - \frac{1}{2(m_1m_2 - a^2)}\sum_{s=1}^{k}\left(y_{1s}^2m_2 + y_{2s}^2m_1 - 2ay_{1s}y_{2s}\right) \end{aligned}$$

and in theory, we could find the optimal value of $\hat{a}$ from this by jointly optimizing m_1, m_2, a, which is our inner product estimate.

Li's key observation was that if the values of m_1, m_2 were known beforehand (this is the norm of the vectors $\|\vec{x}_i\|_2^2$ and $\|\vec{x}_j\|_2^2$), then the above log likelihood equation simplifies to

$$l(a) = -k\log(2\pi) - \frac{k}{2}\log\left(m_1m_2 - a^2\right) - \frac{1}{2(m_1m_2 - a^2)}\sum_{s=1}^{k}\left(y_{1s}^2m_2 + y_{2s}^2m_1 - 2ay_{1s}y_{2s}\right)$$

an equation just in terms of a. Moreover, if the values of m_1, m_2 are known, then the value of a would be "more accurate".

In fact, by taking the first derivative of $l(a)$, the optimal value of a would be the solution to the cubic

$$a^3 - (\vec{y_1}^T\vec{y_2})a^2 + \left(-m_1m_2 + m_1\|\vec{y_1}\|_2^2 + m_2\|\vec{y_2}\|_2^2\right)a - m_1m_2\vec{y_1}^T\vec{y_2} = 0$$

The variance of Li's estimate is given by

$$\mathrm{Var}[\hat{a}_{\mathrm{Li}}] = \frac{(m_1m_2 - a^2)^2}{m_1m_2 + a^2}$$

which is lower than the variance of the original dot product estimate, given by

$$\mathrm{Var}[\hat{a}_{\mathrm{original}}] = a^2 + m_1m_2$$

Li also noted that if there were n observations, then an extra $O(np)$ of time was needed to compute all the norms $\|\vec{x}_i\|_2^2, i = 1, \ldots, n$ but the tradeoff in error reduction was worth it.

It is useful to see a comparison of the variances of the original dot product estimate and Li's estimate. While there are three varying parameters m_1, m_2, a, we simplify the equation for variances by assuming vectors are standardized to have length $m_1 = m_2 = 1$.

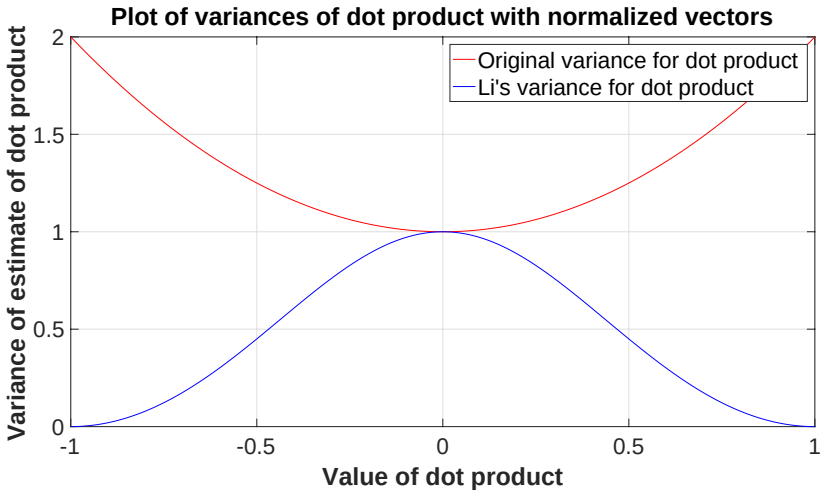

Figure 16: Comparison of variance of original dot product estimate with Li's variance.

Figure 16 shows how the variances differ. When the vectors are pointing in the same direction or opposite directions, Li's MLE method gives the most variance reduction. However, as the vectors get more orthogonal, the variance reduction is negligible until there is no variance reduction at all when the vectors are almost orthogonal.

When I read Li's paper, I was impressed by idea where he made use of information already know, or rather, information that was cheap to compute and store, in order to get a better estimate of the dot product.

This led me to think about whether it was possible to decrease the variance further, especially when the vectors were becoming orthogonal. Hence I was playing about with other computational statistics techniques, as well as think about what ways on how "extra" information could be stored cheaply.

This eventually led to random projections with control variates, which will be described in the next section.

A Maximum Likelihood Estimation

In secondary school or JC, such questions below are common

1. Given a fair dice which is rolled three times, what is the probability of first observing a 5, then an even number, and finally a 6?
2. Given a fair coin which is tossed five times, what is the probability of observing HHTTH?

Such observations can be seen as sequences $x_1, \ldots, x_n$. So in the first case, you get

$$x_1 = \text{“rolled a 5”} \tag{A.1}$$
$$x_2 = \text{“rolled an even number”} \tag{A.2}$$
$$x_3 = \text{“rolled a 6”} \tag{A.3}$$

and you multiply the probability of seeing these sequences. In the first case, this is given by

$$p_1 = \frac{1}{6} \tag{A.4}$$
$$p_2 = \frac{1}{2} \tag{A.5}$$
$$p_3 = \frac{1}{6} \tag{A.6}$$

and we compute $\prod_{i=1}^{3} p_i$.

How does this relate to maximum likelihood estimates? We give an example here.

Example A.1. Suppose we toss a biased coin ten times, and record the sequence of tosses. Suppose we see HHTHHHHTHH. What do you think the probability of tossing a head is?

What we can do is to think of this as a Bernoulli distribution (success / failure). Suppose we let p denote the probability of getting heads, and $1 - p$ to be the probability of getting tails. We denote the likelihood to be a function of our parameter p, i.e.

$$L(p) = \prod_{i=1}^{10} p_i = p^2(1-p)p^4(1-p)p^2 = p^8(1-p)^2 \tag{A.7}$$

Let's think about what $L(p)$ really means. Suppose we chose $p = 0.01$, then $L(0.01) = (0.01)^8(0.99)^2 \approx 1 \times 10^{-16}$ which is rather small.

On the other hand, if we chose $p = 0.5$, then we have $L(0.5) = (0.5)^8(0.5)^2 \approx 1 \times 10^{-3}$.

We can think of a higher value of $L(p)$ to be *more likely* for the observed events to happen. This intuitively makes sense, since $p = 0.01$ means the probability of the coin landing H is 0.01, and seeing a sequence of HHTHHHHTHH is extremely, extremely, unlikely (given by the extremely small probability of 1×10^{-16}.

We might then be interested in the value of p that maximizes $L(p)$, and we can do this by calculus, getting $p = \frac{4}{5}$.

We now formalize this concept.

Usually, we do not know the (parameters of the) probability distribution which data comes from.

We define a **statistical model** to be a family $\{\mathbb{P}_\theta \mid \theta \in \Theta\}$ of different probability distributions parameterized by some θ belonging to a set of possible values Θ.

We usually choose P_θ which is the "most probable".

The above can be seen as a "recipe" on how to find maximum likelihood estimates.

1. Given data, find a statistical model that could represent it. For coin tossing example, pick "Bernoulli".
2. Write down the parameters of this statistical model. For coin tossing example, θ (probability of tossing H).
3. Write down the likelihood function *based on the parameters and what you observe*. For coin tossing example, we observed HHTHHHHTHH, so likelihood function is $\theta^8(1-\theta)^2$.
4. Find the maximum of this likelihood function *over plausible range of parameters*. For coin tossing example, $\theta \in [0, 1]$, so only look at values in this range.
5. This value of θ which gives the maximum is your maximum likelihood estimate.

Sometimes, we look at the log-likelihood function instead (denoted by $l(\theta)$), and find the parameter/s θ which maximizes this function.

We do this because of three reasons.

1. log is a monotonic function, which implies that
$$\arg\max l(\theta) \qquad = \qquad \arg\max L(\theta)$$
2. Instead of finding the maximum of products, we can find the maximum of sums (which are easier to differentiate)

3. Usually, we find optimal parameters by numerical optimization (Newton Raphson, etc). Computing sums (as opposed to computing products) can lead to less numerical error and/or numerical overflow.

B Moment Generating Functions

Moment generating functions (MGFs) can be found in almost any undergraduate textbook on mathematical statistics, such as [Casella and Berger, 2001].

MGFs follow one of the common overarching theme in mathematics, which is illustrated by the following diagram. This theme is actually first taught in high school mathematics (or secondary school mathematics).

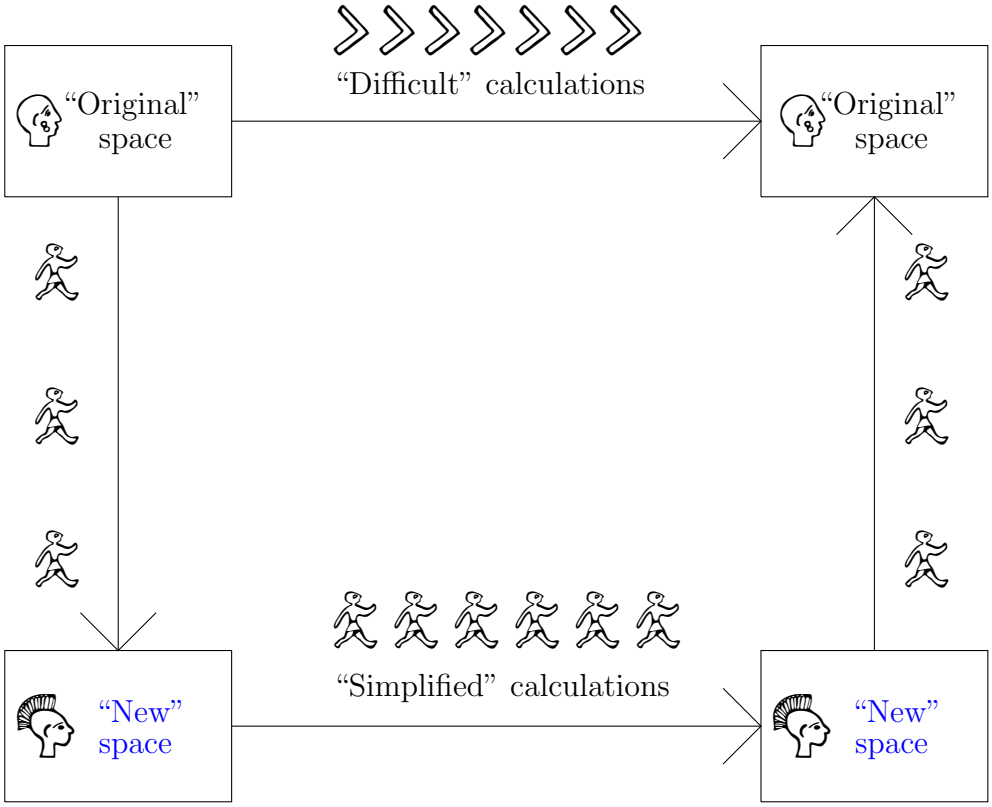

Figure 17: Common theme

Let's suppose we want to do some difficult calculations with some data, and we start in the top left hand side. We can always do the calculations and "go across" to the top right hand side.

Being mathematicians, we are kind of lazy, so we would like to see if we can transform our original data to some "new" space to make our calculations easier. Then we could take a leisurely stroll, do the simplified calculations in the "new" space, and then transform back to the "original space".

The first example most people might have seen would have been logarithms, with the familiar

$\log(ab) = \log(a) + \log(b)$. This simplified multiplication and division by changing them into addition and subtraction. Very useful as a ship's navigator if you had a book of logarithms, and didn't have a slide rule (or modern day calculators). Several other examples could be the Laplace transformation in Calculus, or change of basis / diagonalization in Linear Algebra.

But let's go back to MGFs.

Suppose X is some random variable with some cumulative distribution function F_X, and we are interested in computing successive moments of X, e.g. $\mathbb{E}[X], \mathbb{E}[X^2], \ldots$. Computing such moments can sometimes be tedious. We use MGFs to simplify this calculation.

We use the following definition for MGFs, taken from Definition 2.3.6 in [Casella and Berger, 2001].

Definition B.1. *Moment generating function*
Let X be a random variable with cumulative distribution F_X. The moment generating function of X, denoted by $M_X(t)$ is

$$M_X(t) = \mathbb{E}\left[\exp\{tX\}\right]$$

provided that the expectation exists for t in some neighborhood of 0. That is, there is an $h > 0$ such that for all t in $-h < t < h$, $\mathbb{E}\left[\exp\{tX\}\right]$ exists. If the expectation does not exist in a neighborhood of 0, we say that the moment generating function does not exist.

The MGF can be written as

$$M_X(t) = \int_{-\infty}^{\infty} \exp\{tx\} f_X(x)\, \mathrm{d}x \qquad \text{if } X \text{ is continuous}$$
$$M_X(t) = \sum_x \exp\{tx\} \mathbb{P}[X = x] \qquad \text{if } X \text{ is discrete}$$

Given this definition, once we have the MGF of a distribution, we just need to evaluate the n^{th} derivative of $M_X(t)$ at $t = 0$ to recover $\mathbb{E}[X^n]$.

This can be easily proved by writing down the definition of the MGF and differentiating under the integral sign.

Moreover, for any constants a and b, the MGF of the random variable $aX + b$ is given by

$$M_{aX+b}(t) = \exp\{bt\} M_X(at)$$

where $M_X(t)$ is the MGF of X.

Hence MGFs make it easy to find the moments of random variables coming from a certain distribution.

Moreover, MGFs can be used to characterize a distribution. In some sense, if we know that a distribution has an MGF of the form $\exp\left\{\frac{kt^2}{2}\right\}$, then we know that this is a Normal distribution. See Page 26 for more info.

References

[Achlioptas, 2003] Achlioptas, D. (2003). Database-friendly Random Projections: Johnson-Lindenstrauss with Binary Coins. J. Comput. Syst. Sci., 66(4):671–687.

[Alon et al., 1999] Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., and Levine, A. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proceedings of the National Academy of Sciences, 96(12):6745–6750.

[Casella and Berger, 2001] Casella, G. and Berger, R. (2001). Statistical Inference. Duxbury Resource Center.

[Fosdick and Raftery, 2012] Fosdick, B. K. and Raftery, A. E. (2012). Estimating the Correlation in Bivariate Normal Data With Known Variances and Small Sample Sizes. The American Statistician, 66(1):34–41.

[Frankl and Maehara, 1988] Frankl, P. and Maehara, H. (1988). The johnson-lindenstrauss lemma and the sphericity of some graphs. Journal of Combinatorial Theory, Series B, 44(3):355–362.

[Indyk and Motwani, 1998] Indyk, P. and Motwani, R. (1998). Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC '98, pages 604–613, New York, NY, USA. ACM.

[Johnson et al., 1986] Johnson, W. B., Lindenstrauss, J., and Schechtman, G. (1986). Extensions of lipschitz maps into banach spaces. Israel Journal of Mathematics, 54(2):129–138.

[Kaban, 2015] Kaban, A. (2015). Improved bounds on the dot product under random projection and random sign projection. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 487–496. ACM.

[Kang, 2017a] Kang, K. (2017a). Random Projections with Bayesian Priors. In Natural Language Processing and Chinese Computing - 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings, pages 170–182.

[Kang, 2017b] Kang, K. (2017b). Using the Multivariate Normal to Improve Random Projections. In Intelligent Data Engineering and Automated Learning – IDEAL 2017: 18th International Conference, Guilin, China, October 30 – November 1, 2017, Proceedings, pages 397–405, Cham. Springer International Publishing.

[Kang and Hooker, 2016] Kang, K. and Hooker, G. (2016). Block Correlated Deterministic Random Projections. In Proceedings of the 5th Annual International Conference on Computational Mathematics, Computational Geometry and Statistics.

[Kang and Hooker, 2017a] Kang, K. and Hooker, G. (2017a). Control Variates as a Variance Reduction Technique for Random Projections. In Pattern Recognition Applications and Methods - 6th International Conference, ICPRAM 2017, Porto, Portugal, February 24-26, 2017, Revised Selected Papers, pages 1–20.

[Kang and Hooker, 2017b] Kang, K. and Hooker, G. (2017b). Random Projections with Control Variates. In Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,, pages 138–147. INSTICC, ScitePress.

[Li et al., 2006a] Li, P., Hastie, T., and Church, K. W. (2006a). Improving Random Projections Using Marginal Information. In Lugosi, G. and Simon, H.-U., editors, COLT, volume 4005 of Lecture Notes in Computer Science, pages 635–649. Springer.

[Li et al., 2006b] Li, P., Hastie, T. J., and Church, K. W. (2006b). Very Sparse Random Projections. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pages 287–296, New York, NY, USA. ACM.

[Mardia et al., 1979] Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). Multivariate Analysis. Academic Press.

[Ross, 2006] Ross, S. M. (2006). Simulation, Fourth Edition. Academic Press, Inc., Orlando, FL, USA.

[Vempala, 2004] Vempala, S. S. (2004). The Random Projection Method, volume 65 of DIMACS series in discrete mathematics and theoretical computer science. Providence, R.I. American Mathematical Society. Appendice p.101-105.