Johnson-Lindenstrauss:

the damp squib of dimension reduction

0. Wide Data

Many BIG datasets are wide, not deep. If you think of data as a 2D matrix where the rows are **samples** and the columns are **features** (or variables, or covariates, or …) then a typical big data set has a very large number of samples, so it is *deep* in the sense of being a tall (deep) matrix. But other datasets are *wide* in the sense that they have many columns. Of course, they may be deep and wide, but that's a story for another day.

In mathematical terms, we write our data as a matrix X such that each row $\mathbf{x}_i \in \mathbb{R}^N$ of this matrix corresponds to a sample with N features (a feature vector doesn't have to lie in the space $\mathbb{R}^N$ but this will do for now). If there are m rows then the matrix is $m \times N$ in size.

Now for a wide dataset, N is large. For instance:

1. Images: an image might have 20 million pixels (20 megapixels) each of which has three channels, e.g., red-green-blue, so it has $N = 60$ million.
2. Video: 4K video can have different sizes, but each frame might be 3840 × 2160 pixels (over 8 million pixels), each with three channels, and a frame rate of 30 fps, so even 30 seconds of footage has $N = 22$ billion.
3. Hyperspectral cameras: an image might have a smaller number of pixels, but breaks 400 - 1100 nm into steps of 1 nm, so it has hundreds of channels. Again N would be measured in multi-millions.
4. All your online social media stuff (every post, …).
5. Natural language (e.g., a corpus of a few hundred documents and several million words).
6. The large hadron collider has 150 million sensors. Each is producing information for each experiment so we might take $N = 150$ million.
7. Micro-array genomics can produce information on tens of thousands of genes at once.
8. High-resolution vectoral time series (e.g., temperature at many points on the earth measured every minute over several years).
9. And much, much more.

1. The Curse of Dimensionality

Deep data is computationally challenging, but actually kind of easy once you know how to use computing power well.

Wide data is intrinsically hard. That is the "curse of dimensionality." The term – coined by Bellman with a specific meaning – now has several aspects:

- We can't visualise high-D data.
- Many algorithms (even polynomial time algorithms) are computationally infeasible in high-D. Wide data doesn't parallelise as easily.
- The volume of a high-D space is exponential in the number of dimensions, so it becomes impossible to obtain a dense sample of the space (a sample that hits a reasonable proportion of the volume). Any sample will just be too *thin*. A too thin sample can result in overfitting amongst other problems.
- Noisy features can swamp informative features.

- Distance is high-D spaces are weird (more on this later).

https://www.cs.princeton.edu/courses/archive/fall16/cos521/Lectures/lec9.pdf

2. Dimension Reduction

The primary approach to deal with high-D data is to find an approximation in a lower dimension space, i.e., to perform *dimension reduction*. There are many famous techniques to do so:

- SVD/PCA
- MDS (MultiDimensional Scaling)
- Feature selection

https://towardsdatascience.com/11-dimensionality-reduction-techniques-you-should-know-in-2021-dcb9500d388b

However, these each seek to optimise some aspect of the data, and in doing so they have to perform some work. For very high-D data, that may be impractical.

These approaches also require analysis of the data, so it can't be done in advance of data collection. In some applications we don't start with data. In fact, the data may be so side that collecting the raw data may be impractical in itself.

There is another approach to dimension reduction — random projection. The idea is that we perform a random dimension reduction. That seems crazy – how can a random project work. One of the key insights comes from the Johnson-Lindenstrauss lemma.

3. Johnson-Lindenstrauss (JL) lemma

It is worth keeping in mind that in the data science space we are often performing clustering or some other classification that involves breaking the volume of the space into chunks. Those inherently involve volumes or distances in the space. Hence a natural objective in dimension reduction is to preserve distances as much as is possible. The statement of the lemma follows:

Lemma (JL): Given any $0 < \varepsilon < 1$ and a set of m points $\mathbf{x}_i \in \mathbb{R}^N$ we can chose a number $n > 8 \ln(m)/\varepsilon^2$ and a linear map $f : \mathbb{R}^N \to \mathbb{R}^n$ such that for all $i, j = 1, \ldots, m$ we have

$$(1-\varepsilon)||\mathbf{x}_i - \mathbf{x}_j||^2 \leq ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||^2 \leq (1+\varepsilon)||\mathbf{x}_i - \mathbf{x}_j||^2. \tag{1}$$

What does this mean? Well, we are mapping from an N-D space to an n-D space where hopefully $n \ll N$.

The parameter ε provides a tolerance or a relative error bound for distances in the new space.

The first crucial factor in the lemma is that the dimension of the new space depends on the required tolerance and the number of points we want to embed m. There is **no** dependence on N the original dimension. That is astounding!

The second crucial factor here is not immediately visible because it isn't in the statement. It is in the proof (which I am omitting). The proof provides a method to find the mapping $f(\cdot)$ and it is simply a *random projection*.

One way to do this (at least approximately) is to just chose a random $n \times N$ matrix A (there are several random models that work, but lets just take the elements to be IID Gaussian) and then take

$$f(\mathbf{x}) = A\mathbf{x}. \tag{2}$$

It's that simple!

There are variations on the lemma itself, but those two insights are the ideas of key importance here.

https://www.johnmyleswhite.com/notebook/2014/03/24/a-note-on-the-johnson-lindenstrauss-lemma/

https://cseweb.ucsd.edu//~dasgupta/papers/jl.pdf

4. An Example

Let's look at a quick example (written in Julia). John Myles White gives an example in Julia blog, but I wanted to do my own. The following takes about 80 seconds on my computer and uses a lot of memory so there are better ways to do it but I wanted it to be simple. The main issue is storing the roughly $3,000 \times 1,000,000$ million sized projection matrix. If you run it on a smaller machine you may need to reduce some of the parameters accordingly. Or have a look into faster versions.

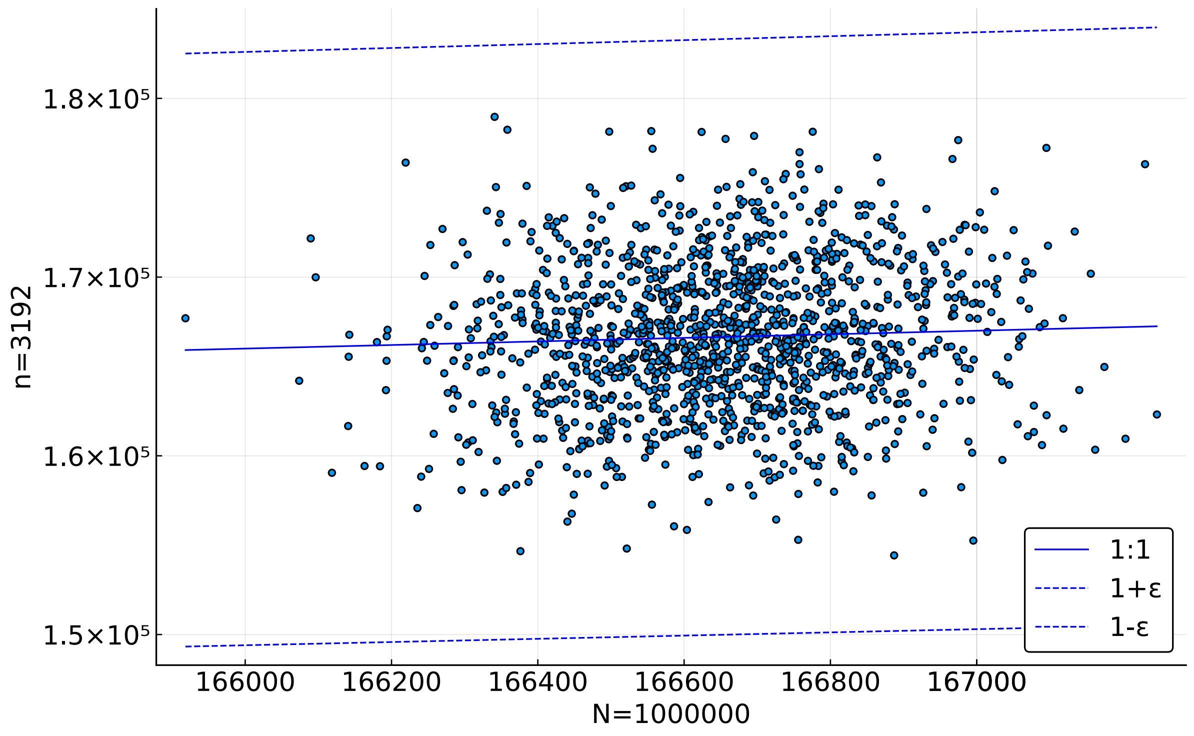
```julia
using StableRNGs
using Distributions
using LinearAlgebra
using StatsBase

seed = 1
rng =  StableRNG(seed)

iceil(x)  = Int( ceil( x ) )
ifloor(x) = Int( floor( x ) )

# parameters
N = 1_000_000
m = ifloor( exp(4) )
ε = 1.0 / 10.0
n = iceil( 8*log(m)/ ε^2 )

# generatge random data
X = rand( N, m )

# generate random projection matrix
normal = Normal(0.0, 1 / sqrt(n))
A = rand(rng, normal, n, N)

# projection
Y = A*X

# calculate distances
D = zeros( Int(m*(m-1)/2) )
d = zeros( Int(m*(m-1)/2) )
k = 0
for i=1:m
    for j=1:i-1
        D[k + j] = sum( ( X[:,i] .- X[:,j] ).^2 )
        d[k + j] = sum( ( Y[:,i] .- Y[:,j] ).^2 )
    end
    global k += i-1
end
mD = mean(D)
md = mean(d)
d = d .* (mD/md) # rescale so that they have the same mean

# now do some plots
```

You can see the parameters I chose (or calculated):

Parameter	Value
N (original dimension)	1 million
m (number of data points)	54
ε	0.1
n	3192

So we are projecting into a space with $\sim 1/300$ th as many dimensions (though it is still quite large).

The results are shown in the following figure:

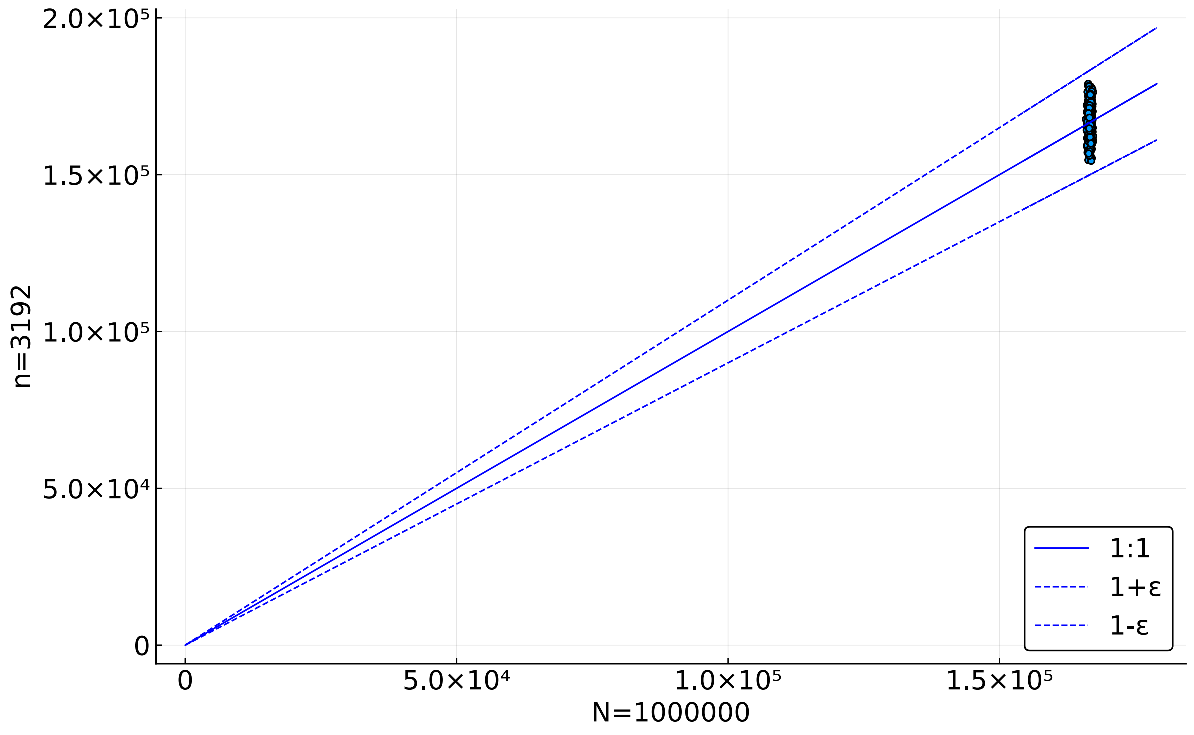


You can see that it works. The points in the new space lie within the bounds as compared to the old space (after I rescale them).

Yay for random projections!

You didn't need a plot to know that it would work because the lemma said so. But the plot is interesting because it also shows the inherent weakness of this approach. You can start to see it already in the plot above. We see a big blob in the scatter plot. It isn't obvious that there is a correlation between the two sets of distances.

When we look at it on a full scale we see the real problem:



Now we see that all the distances in the original space were close together. Much closer than in the new space. What the heck?

5. Why is JL a "Damp Squib"? [1]

High-D spaces are weird. Let's look some simple thought experiments to get started.

Volumes

Imagine a set of points being generated uniformly at random in the N-D hyper-cube. Now imagine inscribing a N-D hyper-sphere inside the hyper-cube. That is, a sphere that *just* fits inside the cube. What is the probability that a random point in the hyper-cube also lies inside the hyper-sphere?

Our intuition, from 3- or 3-D says the probability shouldn't be 1, but isn't that small. For instance, take the side of the hyper-cube to have length ℓ then the cube/square has volume/area of ℓ^2 and the sphere/disk has volume/area of $\pi(\ell/2)^2$ so the probability of being in the sphere/disk is $\pi/4 \simeq 0.79$. That fits my intuition.

But what happens as $N \to \infty$? The N-D hypercube has volume $V_{cube} = \ell^N$ and the N-D hypersphere has volume

$V_{sphere} = \frac{\pi^{N/2}\ell^N}{\Gamma(N/2+1)}$,

where $\Gamma(\cdot)$ is the gamma function. The limit of their ratio is dominated by the gamma function as $N \to \infty$ (as $\Gamma(n+1) = n!$) and hence the ratio goes to zero. That is, the probability of a random point occurring in the hypersphere goes to 0 high-D.

This seems weird to me. Essentially it is saying that almost all points in high-D hypercubes are "in the corners." That means, in essence, that most points are closer to a face or corner of the cube than to each other.

Another way to think about it is the idea from cartography that most of the area in a map lies near the edges. That becomes more and more true in higher-D.

https://stats.stackexchange.com/questions/451027/mathematical-demonstration-of-the-distance-concentration-in-high-dimensions

Distances

Distances are even weirder.

Let us once again take a simple random case. Consider the distances between a set of points chosen uniformly at random in an N-D space. The distances will come from a distribution described by the "line picking problem," i.e., the problem of calculating the distribution of the lengths of randomly chosen lines in some space. The randomly chosen lines are selected by randomly choosing their end-points.

For the sake of simplicity let us this time choose a space which is a hypersphere (technically a ball) and we use ℓ_2 distances (Euclidean distances). We can actually calculate the full distribution for this case, but the first two moments are enough to understand limiting behaviour of the moments, and in particular the relative variance, which goes to zero. That is, the distance between points are (relatively) closer together in higher dimensions.

A simple way to understand it is through the central limit theorem. We can write the distance calculation as

$$d^2 = \sum_{i=1}^{N} (x_i - y_i)^2. \tag{3}$$

If the coordinates are independent (which they are in our example), then the central limit theorem says that the distribution of d^2/N will approach a normal distribution $N(\mu, \sigma^2/N)$, where $\mu = \ell/3$ and $\sigma^2 = \ell^2/18$ are the mean and variance for a single coordinate. Thus the relative variance decreases as $1/N$. Thus points in the space (while distributed wildly through a huge volume) are almost the same distance apart.

What that means is that the very notion of "distance" in high-D spaces is at question.

There are some arguments that other distance metrics might be better, but ultimately I suspect the same results hold (see issues with volumes).

The result is plainly clear in the data I used in my example. The spread of the distances shown in the original space is tiny, and much smaller than that in the lower-D space.

Thus JL isn't doing anything useful here. We have a bunch of randomish distances, all roughly the same, and they become another set of randomish distances, now somewhat further (relatively) apart.

So do any of the insights from the lemma provide actual useful insights anywhere else? That's a topic for next week.

https://www.jstor.org/stable/3647669?seq=1

https://dl.acm.org/doi/10.5555/645504.656414

https://stats.stackexchange.com/questions/99171/why-is-euclidean-distance-not-a-good-metric-in-high-dimensions

https://homes.cs.washington.edu/%7Epedrod/papers/cacm12.pdf

https://stats.stackexchange.com/questions/451027/mathematical-demonstration-of-the-distance-concentration-in-high-dimensions

[1] Damp squib = https://www.phrases.org.uk/meanings/damp-squib.html