Gaussians: The Facts

\[ X \sim N(\mu, \sigma^2) \Rightarrow cX \sim N(c\mu, c^2\sigma^2) \]

\[ X+Y \sim N(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2) \text{ if indep} \]

Note that \( X+X \sim N(2\mu, 4\sigma^2) \) but \( X+Y \sim N(2\mu, \sigma_x^2 + \sigma_y^2) \) if \( X, Y \) are independent

\[ \Rightarrow \text{adding independent copies is "decaying" variance, proportionally.} \]

Let \( G \) be a \( D \)-dimensional vector \((g_1, g_2 \ldots g_D)\) each entry is a \( N(0,1) \) r.v. independent.

Let \( v \) be any \( D \)-dim vector.

**Fact 1:**

\[ \langle v, G \rangle = \sum_i v_i g_i \sim \sum_i v_i N(0,1) = N(0, \sum_i v_i^2) = N(0, \|v\|^2) \]

*Euclidean length.

**Prop 2:** \( G \) is "spherically" symmetric. / rotationally invariant.

That is if we pick \( G \) and rotate it, still get a Gaussian vector. By fixed rotation.

Formally if \( R \) is a rotation matrix then \( RG \) has same distribution as \( G \).

**Proof:** the density function of the joint distribution \((g_1, g_2 \ldots g_D)\) is

\[ \frac{1}{(2\pi)^{D/2}} e^{-\frac{1}{2}g^Tg} \]

So for \( RG \) we get

\[ f(x) = \frac{1}{(2\pi)^{D/2}} e^{-\frac{1}{2}gR^{-1}x^T} = \left(\frac{1}{(2\pi)^{D/2}}\right) e^{-\frac{1}{2}x^T R^{-1} x} \]

Note that \( x^T R^{-1} x \) is a quadratic form of \( x \).

**Dimension Reduction II:**

Random Projections (cont'd).

\[ f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

\[ x \sim N(\mu, \sigma^2) \]
Now recall the JL transform (random projection based dimension reduction).

Pick matrix $A \in \mathbb{R}^{k \times d}$ whose entries are all iid Gaussian $N(0,1)$.

Set $f(x) = \frac{1}{\sqrt{k}} Ax \in \mathbb{R}^k$.

Equivalently, pick $k$ vectors $A_1, A_2, \ldots, A_k$ each some distribution $G$ independent

$$
\begin{bmatrix}
\langle A_1, x \rangle \\
\langle A_2, x \rangle \\
\vdots \\
\langle A_k, x \rangle
\end{bmatrix}
= A = 
\begin{bmatrix}
- A_1 \\
- A_2 \\
\vdots \\
- A_k
\end{bmatrix}
$$

Fact 2: each entry of $Ax$ is independent (by construction) and $\sim N(O, \|x\|^2)$. (by Fact 1)

$\Rightarrow Ax \sim (N(O, \|x\|^2) \quad N(O, \|x\|^2) \quad \ldots \quad N(O, \|x\|^2))$

Fact 3: if $W \sim N(O, \sigma^2)$ then $EW^2 = \sigma^2$

PF: $\text{Var}(W) = \sigma^2 = EW^2 - (EW)^2 = EW^2 - O = EW^2$.

So $E\|f(x)\|^2 = E\|\frac{1}{\sqrt{k}} Ax\|^2 = \frac{1}{k} E\|Ax\|^2 = \frac{1}{k} E(\langle A_1, x \rangle^2 + \langle A_2, x \rangle^2 + \ldots + \langle A_k, x \rangle^2)$

$= \frac{1}{k} \sum_{i=1}^{k} E\langle A_i, x \rangle^2$ $= \sum_{i=1}^{k} \frac{\|x\|^2}{k} = \frac{\|x\|^2}{k}.$

Goal: $\|f(x)\|^2$ has expectation $\|x\|^2$.

Which it should have if it's norm preserving.
So $E \tilde{f}(x) \|x\|^2 = \|x\|^2$ correct length in expectation

Now want much more --
Want that $\|\tilde{f}(x)\|^2$ is close to $\|x\|^2$ a lot of the time (with high probability), almost correct length with high prob.

So need a Chernoff bound.

But, we gave Chernoff bounds only for bounded ($[0, 1]$-valued) random variables.

And we need them for sums of squares of Normals.

No worries. Can either derive them yourself (bit of elbow grease) or refer to your favorite text/wikibook.

**Thm:** Let $Y_1, Y_2, \ldots, Y_k$ be independent $N(0, \sigma^2)$ random variables.

Let $S = \sum_{i=1}^k Y_i^2$ (aka $X^2$-random variable)

$ES = k \sigma^2$ (by Fact 2).

For $\epsilon \leq \frac{1}{2}$,

then $\Pr\left[ |S - ES| > \epsilon(ES) \right] \leq e^{-\frac{(k\epsilon^2)}{c}}$ for some universal constant $c$.

Now: $\|\tilde{f}(x)\|^2 \approx \sum_{i=1}^k \frac{N(0, \|x\|^2)}{K} = \frac{S}{K}$ sum of $k$ indep. square of Gaussians.

$\Rightarrow \Pr\left[ \frac{S}{K} - ES > \epsilon \frac{ES}{K} \right] \leq e^{-\frac{(k\epsilon^2)}{c}}$

$\Rightarrow \Pr\left[ \frac{1}{K} \|\tilde{f}(x)\|^2 - \|x\|^2 > \epsilon \|x\|^2 \right] \leq e^{-\frac{(k\epsilon^2)}{c}}$ if $k = \frac{2c \log n}{\epsilon^2}$

want $\leq \frac{1}{n^2}$ (say)

$\Rightarrow \Pr\left[ \|\tilde{f}(x)\|^2 \notin (1 \pm \epsilon) \|x\|^2 \right] \leq \frac{1}{n^2}$.

(*)
Finally: $f(x)$ is a linear map.

\[ f(x+y) = \frac{1}{\sqrt{k}} (A(x+y)) = \frac{1}{\sqrt{k}} (Ax + \frac{1}{\sqrt{k}} Ay) = f(x) + f(y). \]

\[ \Rightarrow \|f(x_i - x_j)\|^2 = \|f(x_i) - f(x_j)\|^2 \]

by (4) above.

\[ \|x_i - x_j\|^2 \]

\[ \Rightarrow \text{get that for a set } X \subseteq \mathbb{R}^D \text{ of points } x_1, x_2, \ldots, x_n, \]

\[ \forall i, j \quad \frac{\|f(x_i) - f(x_j)\|^2}{\|x_i - x_j\|^2} \leq [1 - \epsilon, 1 + \epsilon] \]

with probability \(1 - (\binom{n}{2}) \cdot \frac{1}{n^2}\). Prob of one pair having too much distance change, by (3).

\[ \geq \frac{1}{2}, \quad \text{union bound} \]

So all pairs of points preserved with prob $\geq \frac{1}{2}$.

Can check if map is good, if not throw it away and repeat.

Or increase $k$ by constant factor, then $P[\text{mess up}] \leq \frac{1}{k} \Rightarrow P[\text{mess up overall on single vector}] \leq 1 - (\frac{3}{2})^\frac{1}{k} \leq 1 - \frac{1}{k^\frac{1}{2}}$.

Rubinov: draw $KD = O(D \log n \sqrt{\epsilon^2})$ Normal random vars.

Perform $f(x_i)$ for each of $n$ vectors.

Takes $O(KD)$ time.
Notes (1) Map is Data Independent! (At least until we check)

We build matrix $A$ without considering the pointset $X$ at all.

(1) makes it universal, can do one construction in advance, use it for
any $n$ points in the future, will succeed w.h.p.

(2) does not "adapt" to the data, so if data is almost low dimensional
\[ \text{it does not take advantage of it.} \]

(Compare to PCA).

(2) Can replace $N(0,1)$ entries by random $\pm 13$ uniform entries.

or any "sub gaussian" random variables.

(See notes by Vershynin, or our 850 lecture notes).

(3) Can speed things up a bit (a lot!)

right now matrix $A$ has to be dense (to handle case when input
vector $X$ is sparse).

But using (Fast) Fourier Transform we can make $X$ dense

and then choose sparser $A$, mostly zeros.

intime $O(D \log D)$

("uncertainty principle")

See notes on Fast JL Transform

(4) Use cases:

- $\text{when need distance preservation (near neighbor search, PCA does not preserve distances)}$

- $\text{as preprocessing - e.g. for that matrix multiplication example, see HW.}$
Related topic: (same ideas, extended)

**Compressive Sensing**

Have a signal that we view as a vector \( x \in \mathbb{R}^D \).

(+) Signal is sparse, has only \( S \) non zeros (say).

(\(-\)) Can only do linear measurements on it (and few of them).

So at each measurement, can give a vector\( m^t \in \mathbb{R}^D \) and get back answer \( \langle m^t, x \rangle \in \mathbb{R} \).

Or network sensing, or many other apps.

Or network sensing, or many other apps.

Want to figure out \( x \) with few measurements.

**Strawman solution**: 

Set \( m^t_i \) = \( (0,0, \ldots, 0, 1, 0, 0, \ldots 0) \)

\( \uparrow \)

\( \uparrow \)

All other positions are "shut off".

Then \( \langle m^t_i, x \rangle = x_i \)

\( \Rightarrow \) can learn \( x \) in \( D \) measurements.

does not use sparsity at all!
Theorem [Donoho, Candes - Tao, ...]

\[ \text{3 measurements: } m_1, m_2, \ldots, m_k \text{ for } k = O(s \log \frac{D}{s}) \]
\[ \forall x \in \mathbb{R}^D, x \text{ is } s\text{-sparse} \]
\[ \text{such that we can reconstruct } x \text{ with high probability.} \]
\[ \text{from } \langle m_1, x \rangle, \langle m_2, x \rangle, \ldots, \langle m_k, x \rangle. \]

In fact, let's view
\[ M = \begin{pmatrix} -m_1 \\ -m_2 \\ \vdots \\ -m_k \end{pmatrix} \begin{pmatrix} x \end{pmatrix} = \begin{pmatrix} \langle m_1, x \rangle \\ \langle m_2, x \rangle \\ \vdots \\ \langle m_k, x \rangle \end{pmatrix} \]
\[ \text{vector of } k \text{ observations.} \]

Then some details:

1. \( M \) is a Gaussian matrix, each entry of \( M \sim N(0,1) \).

   Can show that every \( s \)-sparse vectors should have its length preserved approximately.

   "Restricted isometry"

   There are "only" \( \binom{D}{s} \) choices of the \( s \)-sparse coordinates

   \[ \approx \binom{D}{s} \approx \log(\frac{D}{s}) s \log(\frac{D}{s}). \]

2. Algorithm: find \( y \) such that

   \[ My = Mx \]

   know measurement matrix, we built it!

   But many possible solutions; \( x \) is a solution but also many others.

   Which one to choose?
Fact: suppose could choose \( x \) _\text{sparsest} \) such that \( Mx = \text{observations} \) then \( x \) is correct answer with high probability (overchoice \( M \)).

But finding such \( x \) is computationally hard. (NP hard).

Fact: could ask for \( \min \| x \|_2 \leq \text{Euclidean lengths} \)

\[ s.t. \ Ax = \text{obs}. \]

Fast but wrong answer. 😞

Fact: \( \min \| x \|_1 \leq \min_{x} \sum \| x_i \| \]

Such that \( Ax = \text{observations} \).

Is correct! \([\text{Famous algo }\] now called Basis Pursuit\)

and is a linear program, can be solved in polytime (and fast).

See this often:

If we want sparse solutions, then minimize the \( L_1 \) norm of the vector and not the \( L_2 \) norm, or other \( L_p \) norms.

\( \ell_1 \) ball grows fatlest in the coordinate directions.

\( \ell_2 \) ball grows fatlest in even densest.

Sparse, gross fatlest in coordinate dirs.
Some High Level Points:

(4) Problem is akin to:

- signal is sparse in some basis. (standard basis) [but could be other basis too — say the Fourier basis, or wavelet basis]
- but can measure only in some other basis. (that given by rows of M matrix)

- How to do few measurements?
  - justify efficacy of undersampling.

(4) Arises in cameras, (prototypes thus far, AFAIK)

- network sensing (only few routers have useful information, but want to collect these items together by summing them up, then reconstruct which router has what).
- geophysical systems: measure where discontinuities between strata are by sending waves through rocks.
- CAT scans.
  - see CompSci web page for other examples.

(4) Same idea of minimizing L1 norm to get sparsity arisen in

ML applications too — when we want a low rank matrix, instead minimize the L1 norm of the singular values (the nuclear norm)!

- Interesting fact: even though it seems some measurements are sparse (see Valiant/Roughgarden notes about geology) they are sparse in frequency space, hence dense in spatial domain. Hence things are ok.
- Lower bound?

Clearly we need to get \( 2^{O(s \log (n/s))} \) bits for the signal to
tell us which of the \((n/s)\) coordinates are beniquead.

But: each answer can be multiple bits. So don’t get a strong
lower bound for \# measurements this way.

And not surprising: – I a way to do \( 25 \) measurements and
get back signal if it is exactly \( s\)-sparse. (Due to
Baron de Prony (1795), specifically for Fourier transforms.)

OTOH: I a lower bound when we want to handle noisy signals,
and get back (approximately) a signal that has most of its
mass on only \( s \) coordinates. \( \text{LB} = 2(0.38 \log n) \)

And the algorithm (Basis Pursuit – aka LP solving when matrix
is a Gaussian matrix) also works in this setting. Much more
important in practice, because getting an exactly \( s\)-sparse
signal is very unlikely (but almost \( s\)-sparse signals
arise a lot!).

Finally: most ways of speeding up JL (aka the Fast JL transform) also
are applicable in this setting. E.g. use of \( \ell_1 \)-\ell_2 random variables,
using a Fourier transform to convert sparse signals to dense ones, etc.