Decoding noisy messages: a method that just shouldn't work

Leo Liberti

Decoding noisy messages: a method that just shouldn't work

Leo Liberti[1]

[1] *LIX CNRS, École Polytechnique, Institut Polytechnique de Paris, F-91128 Palaiseau, France*
Email:liberti@lix.polytechnique.fr

May 27, 2022

Abstract

This paper is about receiving text messages through a noisy and costly line. Because the line is noisy we need redundancy, but because it is costly we can afford very little of it. I start by using well-known machinery for decoding noisy messages (compressed sensing), then I attempt to reduce the redundancy (using random projections), until I get to a point where I use more orthogonal vectors than the space dimension allows. Instead of grinding to a halt or spurting out noise, this method is still able to decode messages correctly or almost correctly. I have no idea why the method works: this is my first reason for writing this paper using a narrative instead of formal scientific style (the second one is that I am tired of writing semi-formal prose, and long for a change).

1 The risks of office gossip

At a time when I was mainly working on optimization and Euclidean distance geometry, I happened to walk past an open office door, where I heard someone mumble "random projection"; my eye glanced upon a whiteboard bearing the formula

$$\|Tx - Ty\|_2 \approx \|x - y\|_2. \tag{1}$$

Intrigued by the Euclidean distances, I stopped, entered the office, and interrupted the ongoing discussion. "What's that T?", I asked. I was told that it was a $k \times n$ random matrix, and that the vectors x, y were any pair of n-dimensional vectors out of a finite set. I immediately objected that no such T could exist, unless the "approximation" relationship between the two distances was taken very liberally. "Don't you know the Johnson-Lindenstrauss lemma?", was the reply. I did not. After being given a summary explanation I left that office, my head in turmoil, and headed towards the library in search of more information.

Upon coming back from the library, I met a friendly colleague and asked him what he knew about random matrices. He immediately started talking about things unrelated to Eq. (1). He had a disposition to stray far from the subject of conversation, which was one of the reasons I liked him. This attitude resonated with my habit of making connections between different topics, the more disconnected the better. He was talking about tiny cameras with a ridiculously small number of pixels, but able to take fantastically detailed pictures nonetheless. He asked me "do you know how many pixels I really need for those pictures? guess!". I guessed in the thousands, then in the hundreds, then gave up. "One!", he said, smiling broadly. For the second time that day, I was nonplussed, and couldn't believe what I was hearing. A whole parallel world existed in my colleagues' minds, something that appeared to go against the expected tenets of normal mathematics, and I was cut off. This had to stop: I would measure the extent of their lies, and ascertain what truth might be hidden in their unlikely stories.

Thus started a long period of learning, when I became aware of a large part of mathematics I had ignored so far. I initially focused on the office gossip about Euclidean distances: I read the lemma by Johnson and Lindenstrauss [17] (whose proof remained unintelligible to me for a long time), learned some unexpected properties of random matrices having components sampled from Gaussian distributions [4], and became aware of their relation with data science [14]. I heard a reference to the one-pixel nonsense

in a talk by Emmanuel Candès at the International College of Mathematicians in 2014 [6], where, again, random matrices were mentioned. A few web searches later, I had uncovered a name, that of Matoušek, connected to these concepts.

Connections notwithstanding, I was by then fully convinced that my ignorance was abysmal, all-encompassing, and was procuring me a scientific damage beyond repair. I had been living a quiet mathematical life, sheltered from the stormy developments of the youngest greatgrandchildren of functional analysis, linear algebra, and geometry. Contemporary mathematics was passing me by. The Mathematical Programming (MP) niche that looked so rich and endless appeared like a prehistorical cave with a few ancient rat bones scattered about.

Luckily, as I walked into one of my favorite maths bookshops (now unfortunately closed forever), I stumbled upon a copy of a book about Linear Programming (LP) written by Jiří Matoušek and Bernd Gärtner [25]. It attracted my attention because it had an in-depth discussion of the Delsarte bound for codes [12]: my interest was in understanding some difficult papers about the Delsarte bound for *spherical* codes [13], which arise in the study of the Kissing Number Problem [22]. Unfortunately, Matoušek and Gärtner only discussed the application to discrete codes. The next section [25, §8.5], however, was titled "Sparse solutions of linear systems", which resonated with the title of Candès' talk at ICM14, "The mathematics of sparsity". So, charmed by serendipity, I bought the book and kept on reading.

2 Noisy communication channels

At this point I will hand over the story to the authors of [25].

> We begin by discussing error-correcting codes again, but this time we want to send a sequence $w \in \mathbb{R}^d$ of d real numbers. Or rather not we, but a deep-space probe which needs to transmit its priceless measurements represented by w back to Earth. We want to make sure that all components of w can be recovered correctly even if some fraction, say 8%, of the transmitted numbers are corrupted, due to random errors or even maliciously (imagine that the secret *Brotherhood for Promoting the Only Truth* can somehow tamper with the signal slightly in order to document the presence of supernatural phenomena in outer space). We admit *gross errors*; that is, if the number 3.1415 is sent and it gets corrupted, it can be received as 2152.66, or 3.1425, or -1011, or any other real number.
>
> Here is a way of encoding w: We choose a suitable number $n > d$ and a suitable $n \times d$ encoding matrix Q of rank d, and we send the vector $z = Qw \in \mathbb{R}^n$. Because of the errors, the received vector is not z but $\bar{z} = z + x$, where $x \in \mathbb{R}^n$ is a vector with at most $r = \lfloor 0.08n \rfloor$ nonzero components. We ask, under what conditions on Q can z be recovered from $\bar{z}$?
>
> Somewhat counterintuitively, we will concentrate on the task of finding the "error vector" x. Indeed, once we know x, we can compute w by solving the system of linear equations $Qw = z = \bar{z} - x$. The solution, if one exists, is unique, since we assume that Q has rank d and hence the mapping $w \mapsto Qw$ is injective.

Let me summarize: we need to send a vector $w \in \mathbb{R}^d$ on a noisy channel. We would like to find an encoding $n \times d$ matrix Q, with $n > d$, and send $z = Qw \in \mathbb{R}^n$ to the recipient. We assume that both parties know Q. There is a low probability Δ of communication error (assumed to be 8% in [25]), which implies that, on average, on sufficiently long vectors, Δn components of z will be wrong at reception. Let $x \in \mathbb{R}^n$ be the error vector, so that the received message is $\bar{z} = z + x$. Evidently, x will have on average only Δn nonzero components. What conditions on Q should we impose so that the recipient can retrieve w from $\bar{z}$?

The book [25] goes on to explain that we (stepping in the recipient's shoes) should choose an $m \times n$ matrix A with $m = n - d$ such that $AQ = 0$, and let $b = A\bar{z}$. We should then find the sparsest solution

x' to the linear system $Ax = b$. Armed with x', we find $z' = \bar{z} - x'$, and finally $w' = (Q^\top Q)^{-1} Q^\top z'$. In general, we should find that $w = w'$, or at least that w, w' are not very different. Note that the sparsity of a vector $x' \in \mathbb{R}^n$ is related to its support, i.e. the set of indices of the nonzero components $\mathsf{supp}(x') = \{j \leq n \mid x'_j \neq 0\}$: x' is sparse when its support is small w.r.t. n.

There are three technical points worth making about the preceding discussion.

1. We compute b as $A\bar{z}$ and then want to solve the system $Ax = b$. An encoded noisy message $\bar{z}$ and an error vector x (two vectors with apparently very different meanings) are both being cast in the same role, in the sense that they are being multiplied by A and give b as a result: why? The reason is that we have
$$b = A\bar{z} = A(z + x) = A(Qw + x) = (AQ)w + Ax = Ax, \tag{2}$$
since $AQ = 0$ by definition.

2. The system $Ax = b$ is underdetermined, since A is an $(n - d) \times n$ matrix. It has a solution as long as the rank of A is equal to the rank of the extended matrix (A, b), a sufficient condition being that A has full rank $n - d$.

3. We compute an approximation w' of the original message as $(Q^\top Q)^{-1} Q^\top z'$: this is the well-known Moore-Penrose pseudoinverse, which is being used here to "invert" the (overdetermined, hence in general uninvertible) linear system $Qw = z$ used during the encoding.

Note that, in the aforementioned discussion, x is being used both as a numerical error vector in $\mathbb{R}^n$ and as symbol for a vector of variables in the system $Ax = b$ to be solved. To avoid ambiguities, I shall henceforth reserve x for the variable vector (as well as to make occasional appearances as a vector in theorem statements), and use $\bar{x}$ for the error vector. As long as I am discussing notation, let me clarify that $|\cdot|$ denotes both absolute value and set cardinality: I accept this ambiguity because both notations are standard and established, and because I made it clear what sign means what in the only possible case of ambiguity, Eq. (4).

And then, of course, there is the metaphorical elephant in the proverbial room: why are we finding the sparsest solution x' to $Ax = b$, and why, having found it, would it be equal to the error vector $\bar{x}$, which does not seem likely that we — the receiving party, now — could ever know, since it depends on the whims of the noisy line?

3 The one-pixel camera

Eventually, I found some answers to the last questions of the foregoing section. But they are not the type of answers I had come to expect from mathematics. Rather than the short, clear-cut first-order logical propositions with the usual, well-known, and trusted sentences of the Zermelo-Fraenkel-Choice (ZFC) axiom system [20], which are either provable or unprovable (and those that are provable end up being true in all sensible models), I found myself entangled in vague probabilistic statements. Their associated probabilities exceed certain threshold values that can be made arbitrarily close to one in function of a monotonically varying parameter, the actual value of which is at best hard to compute, and at worst unknowable, in all practical settings. I found something similar in the study of random projections I had pursued a couple of years earlier — something else that Johnson, Lindenstrauss and the one-pixel camera had in common.

Let us start from one of the basic questions: how does one find the sparsest solution to an underdetermined linear system $Ax = b$? A MP formulation for this problem is $\min\{\|x\|_0 \mid Ax = b\}$, where $\|\cdot\|_0$ is the "zero-norm", which is not really a norm, but simply counts the number of nonzero components of x. For later reference, I shall call this formulation $P^0(A, b)$. It turns out that $P^0(A, b)$ is **NP**-hard by reduction from Exact Cover by 3-Sets [15, A6[MP5]].

The signal processing community had been trying to solve these noisy decoding problems for decades: for example in order to build telephones, radios, TVs. In attempting to solve $P^0(A,b)$ efficiently they tried everything, including the replacement of the zero-norm with the much friendlier ℓ_1 norm. The corresponding problem $\min\{\|x\|_1 \mid Ax = b\}$, which I will refer to as $P^1(A,b)$, can be reformulated exactly to the following LP:

$$\left.\begin{array}{rrcl} \min & \sum_{j\le n} s_j & & \\ \forall j \le n & -s_j \le & x_j & \le s_j \\ & Ax & = & b. \end{array}\right\} \tag{3}$$

With much amazement, signal processing researchers noticed that, if they chose the dimensions d and n carefully, the solution x' of $P^1(A,b)$ was the same as the solution of $P^0(A,b)$. This happened too often for them to accept it as a "coincidence". The theoretical justification came from the work of many people, but principally David Donoho [27], Terence Tao, and Emmanuel Candès [7, 8]. I do not really want to provide a full proof here: I will just sketch a proof strategy which I borrowed from many sources, but mainly from [10].

Part of the vagueness I referred to above comes from a practical need: that of computing with floating point numbers. If you find that a component of x' has order of magnitude 10^{-9}, do you say that it is zero or nonzero? Let us provide ourselves with a small $\epsilon > 0$. We say that a scalar σ is "almost zero" or "near-zero" if $|\sigma| \le \epsilon$. Moreover, given an integer $s \le n$, for a vector $\hat{x} \in \mathbb{R}^n$ having support size $\ge s$, we say that $\hat{x}$ "almost" has support size s if

$$|\mathsf{supp}(\max(\mathbf{0}, |\hat{x}| - \epsilon\mathbf{1}))| = s. \tag{4}$$

Notationwise, the inner $|\cdot|$ operator denotes absolute value, while the outer denotes set cardinality. Moreover, operators are applied vectorially: $\mathbf{0}$ and $\mathbf{1}$ are the all-zero and all-one vectors, and we extend the absolute value and maximum operators $|\cdot|, \max$ to apply to vectors componentwise. So a vector almost has support size s if there are exactly[1] s components with absolute value larger than ϵ.

3.1 Theorem
Given $\hat{x} \in \mathbb{R}^n$, a scalar $\delta \in (0, 1/(2+2\sqrt{2}))$, and a $m \times n$ matrix A sampled componentwise from a normal distribution with mean 0 and standard deviation 1 (denoted $A \sim \mathsf{N}(0,1)^{mn}$), there exist two constants $c_1, c_2 > 0$ depending on δ such that, under the following assumptions:

1. *$\hat{x}$ almost has support size s;*
2. *$m \ge \frac{1}{c_1} s \ln(\frac{n}{s})$;*
3. *$P^1(A, A\hat{x})$ has a minimum x^* with $|\mathsf{supp}(x^*)| = s$,*

we have

$$\mathsf{Prob}\left[\|x^* - \hat{x}\|_1 \le 2\frac{1+2\delta(\sqrt{2}-1)}{1+2\delta(1-\sqrt{s})}\|\tilde{x} - \hat{x}\|_1 \le (n-s)\epsilon\right] \ge 1 - e^{-c_2 m}. \tag{5}$$

where $\tilde{x}$ is obtained by zeroing the $n - s$ smallest (in absolute value) components of $\hat{x}$.

This theorem is a convincing example of my accusations of vagueness. (a) It is based on some "constants c_1, c_2 depending on δ" (but we are not told how to compute them). (b) It proves a probabilistic statement which approaches 1 rapidly as m increases, but we may not be able to control m in practice: after all if $m = n - d$ is large, it means that the transmitted message is much larger than the original one: what if the communication channel is both noisy and costly? (c) It expects that a human reader would interpret the almost unparseable statement in Eq. (5) as "x^* is a good approximation of $\hat{x}$": what if it is not quite as good an approximation as we need? We can certainly fine-tune δ, but probably at the expense of the constants c_1, c_2.

[1] Some results in [10] relax *exactly* to *at most*. This relaxation is, however, motivated and formally explained.

In fact, my understanding of these results is that they explain phenomena that were already known computationally within a broader community. They provide a qualitative account, more than a workable recipe. These theories work inconclusively often, with unforeseeable exceptions that are possibly curable at the expense of increasing sizes to unwieldy millions, billions or more. Nonetheless, they have the merit of addressing foundational doubts that other communities must have had. "Goodness gracious, I'm designing a communication protocol for a new aircraft, and I'm using a technique which has always worked, but no-one knows why. I'm going to hell, aren't I?" Such researchers will sleep soundly, from now on — at least with arbitrarily high probability depending on an unknowable constant.

Proving Thm. 3.1 requires quite a bit of time and patience, at least in the treatment of [10]. The strategy is as follows:

1. If A has a certain complicated property called "null space property" (NSP), Eq. (5) follows.
2. If A has a certain other somewhat less complicated property called "restricted isometry property" (RIP), then A also has the NSP.
3. If $A \sim \mathsf{N}(0,1)^{mn}$, then A has the RIP.

Note that this proof is a chain of implications. We end up proving that matrices sampled componentwise from a normal distribution are good, but this does not mean they are the only ones. Moreover, the lower bound for m given in Assumption 2 in Thm. 3.1 is valid, but it may not be tight. In proving Thm. 3.1, by the way, we also learn that the NSP guarantees that the minimum x^* of $P^1(A, A\hat{x})$ having support size s is unique.

Again, all this comes mostly from [10]. The theoretical treatment of Thm. 3.1 in [25] is more compact than that of [10], and has no probabilistic statements of the likes of Eq. (5), but it actually proves a much weaker and less general result than Thm. 3.1 — only enough to justify the LP with the application at hand. I find that the treatment in [26, §5.5] strikes a better balance between vagueness, clarity, and generality.

To go back to our original issue, we now know that solving $P^1(A, b)$ gives a good approximation to the sparsest solution of $Ax = b$ with high probability, as long as the number of rows m of A is large enough: we have a lower bound for m, but it is not tight.

In case you are still wondering about the one-pixel camera, let me provide some closure. I did find a few mentions of a "one-pixel camera" connected with the names usually attributed to Thm. 3.1 (namely: "compressed sensing", "compressive sampling", "mathematics of sparsity"). I would not be surprised if the physical existence of this object were just as vague as many of the theoretical statements that no doubt prompted my colleague to mention it to me.

4 Finding hay in a haystack

I want to go back to Eq. (1), and replace that nasty "≈" sign by something more precise. Johnson and Lindenstrauss worked in a period where statements such as Eq. (5) were less common. Their lemma is existential rather than probabilistic (the method of proof is probabilistic [2], however). Again with minor notation changes, we have:

4.1 Lemma
For each $\varepsilon \in (0,1)$ there is a constant $c = c(\varepsilon) > 0$ (not depending on n) so that if $X \subset \mathbb{R}^n$ with $|X| = n$, then there is a mapping $f : X \to \mathbb{R}^k$ (where $k = \lceil c(\varepsilon) \ln n \rceil$) which satisfies

$$\forall x, y \in X \quad (1-\varepsilon)\|x-y\|_2 \leq \|f(x)-f(y)\|_2 \leq (1+\varepsilon)\|x-y\|_2. \tag{6}$$

Any probabilistic proof argues a positive probability of the existence of a certain entity, whence it infers that the entity must exist. The entity referred to in Lemma 4.1 is the function f. The proof shows that

f is a random orthogonal projection to a lower-dimensional subspace — of course it does not exclude that f could take other forms.

Other than that, the proof is barely understandable. It contains sentences an old-school mathematician would never want to read in a proof, e.g. "We plugged in the exact constant of $\sqrt{2}$ in Khintchine's inequality [19], but of course any constant would serve as as well", which the part of me who was educated on Kenneth Kunen's *Set Theory* [20] considers as inacceptably vague[2]. If the constants do not matter, how can one ever hope to use the result in practice, where computation requires that all constant symbols must take a definite value? Or are we to understand that we are facing a computation which will give the same result no matter what the value of the constant is? But that would simply imply that the statement is independent of such a constant, which should therefore be ignored: so why isn't it? And so on with the nagging doubts.

At the end of the proof we learn that

$$c(\varepsilon) \geq \frac{100}{\varepsilon^2}. \tag{7}$$

Now suppose you had a set X of 1000 vectors in $\mathbb{R}^{1000}$, and you wanted to lose some dimensions, obtaining a corresponding set $f(X)$ such that all pairwise distances are preserved up to 1% accuracy. Then $\varepsilon = 0.01$, which means that the projected dimension $k = \lceil 100/(0.01)^2 \ln(1000) \rceil$, which has value $6,907,756$: a remarkable increase from $n = 1000$. Of course, if you had a set of one billion vectors in one billion dimensions, k would be just over one fiftieth of that, at $20,723,266$. From one trillion, the figure for k would be only 28 millions. When the logarithm in the growth order starts kicking in, we are obviously doing fine. The fact that the minimum value for k starts at six millions (in the above example), however, makes one wonder about the practical usefulness. The issue, here, resides in the "inacceptably vague" statement in the proof of Lemma 4.1 to the effect that Johnson and Lindenstrauss chose a certain constant (namely $\sqrt{2}$) resulting in the number 100 in Eq. (7). Choosing something other than $\sqrt{2}$ would have perhaps yielded a value smaller than 100 in Eq. (7).

Later proofs (see e.g. [11]) contributed a few other fundamental concepts to the Johnson-Lindenstrauss Lemma (JLL): (i) the notation $k = \lceil C\varepsilon^{-2} \ln(n) \rceil$, which makes the existence of coefficient C explicit, without providing any hint about its value; (ii) the fact that the dimension m of the vectors in X need not be equal to the number n of such vectors; (iii) an estimation of the probability of f to do what it's supposed to (a probability which tends to one with exponential speed, but where the exponential term involves a "universal constant" c which we cannot really compute); (iv) a variety of simpler constructions for f. Nowadays, we take f to be $k \times n$ matrices sampled from sub-Gaussian distributions, of which the Gaussian distribution is one. But there are also nicer sub-Gaussian distributions, for example the distribution which samples 1 or -1 with probability 1/6 and 0 with probability 2/3 [1], its subsequent variant where $\mathsf{Prob}(1) = \mathsf{Prob}(-1) = \alpha/2$ and $\mathsf{Prob}(0) = 1 - \alpha$ for some given $\alpha < 1$ [18], or the more comfortable sparse normal matrices with given sparsity described in [9, §5.1].

The literature on the JLL is by now vast. In modern treatments (e.g. [29]), the JLL is presented as a statement that has clear analogies with Thm. 3.1.

4.2 Theorem
Let X be a set of n points in $\mathbb{R}^m$, $\varepsilon > 0$, and C, c be two universal constants. Then there exist an integer $k \geq \frac{C}{\varepsilon^2} \ln(n)$, and a $k \times m$ matrix T sampled componentwise from a sub-Gaussian distribution with zero mean and standard deviation $\frac{1}{\sqrt{k}}$, such that:

$$\mathsf{Prob}\left[\forall x, y \in X \ (1-\varepsilon)\|x-y\|_2 \leq \|Tx - Ty\|_2 \leq (1+\varepsilon)\|x-y\|_2\right] \geq 1 - 2n(n+1)e^{-c\varepsilon^2 k}. \tag{8}$$

In the form of Thm. 4.2, the result says that any random sub-Gaussian matrix T is a good choice for the f in the JLL (Lemma 4.1). The issue of determining the constant has been made a non-issue: the JLL today is viewed as a principle more than a recipe for computation. It is up to the computer programmer

[2]This should not be misconstrued into a feeling that I lack respect for this result or either of its authors! I am only lamenting my lack of imagination when reading creative proofs.

to find values for C, c that verify Eq. (8). Some attempts have been made in this sense, see e.g. [28]. Once you try sampling T you soon realize that values in the range $[0.5, 2]$ seem to be fine, which lowers the usefulness bar from "over one million" to "between $5,000$ and $20,000$", a much more acceptable threshold. If you want smaller thresholds, you can allow for higher errors, and raise that 1% to 10% or even 20%. There are many more caveats in order to successfully use the JLL in practice (see e.g. [23, §7.3.1] and [24]), but there is also considerable flexibility.

The vagueness of these probabilistic results turns out to be a blessing, albeit in disguise. It only captures a wealth of gauges which you can calibrate for your own uses. The trade-off of this flexibility is that making theoretically valid choices for the universal constants is only possible if the specific mathematical properties of the problem at hand allow it (moreover, you have to construct the theoretical arguments, which takes time and skill). Otherwise, if you lack skill and time (like me), you can resort to informed guessing aided by empirical tests.

There is one further safeguard against poor choices for C, c, k, and T in Thm. 4.2. The JLL is based on the phenomenon of "concentration of measure" [21, 5], which ensures that a certain sampled function has high probability of being close to its median value. As a consequence, the event of sampling a random projector T that *completely fails* (i.e. it fails for most of all pairs x, y) to satisfy its expected property Eq. (6) (where $f(x) = Tx$) is going to be extremely rare. It may be the case that Eq. (6) fails to hold for a few pairs x, y, but it will hold for most pairs. This may or may not be an issue, depending on the problem at hand. But in most large-dimensional datasets, where the data are prone to be partly wrong and noisy, small failures of Eq. (6) should not impair the usefulness of the dimensionally reduced vector set TX.

In summary, acceptable random projectors look more like hay (than needles) in a haystack.

5 Epiphany and communion

I wrote in Sect. 3 that the bound in Assumption 2 in Thm. 3.1 is not tight. In choosing the $n \times d$ encoding matrix Q, it would be great to decrease n, so that the resulting encoded vector $z = Qw$ is not excessively costly in terms of storage and bandwidth: after all, the noisy channel is likely to be costly too, especially if it is a two-way communication link between Earth and a deep-space probe, as in Matoušek and Gärtner's example. But what sort of result can I invoke in order to reassure myself that I am following, if not a theoretically proven recipe, at least a theoretically solid principle?

Let us go back to the formulation of $P^1(A, b)$ in Eq. (3): the equation constraints $Ax = b$ can be equivalently written as $\sum_j A_j x_j = b$, where $j \leq n$ indexes the column vectors A_j of A, and x_j is the j-th component of the decision variable vector x. We know that $X = \{A_1, \ldots, A_n, b\} \subset \mathbb{R}^m$. If we apply the JLL to X, we should obtain lower dimensional vectors $TX = \{TA_1, \ldots, TA_n, Tb\} \subset \mathbb{R}^k$, with $k = O(\varepsilon^{-2} \ln(n+1))$. If m is large enough, we can hopefully concoct some universal constants, and choose an appropriate ε, such that $k \ll m$ (or at least $k < m$), and that the probability of Eq. (6) is still high. But one question remains: will the corresponding LP $P^1(TA, Tb)$ yield approximately the same optimal value and optimal solutions as $P^1(A, b)$, simply because the pairwise distances between the vectors in TX are a good approximation of the corresponding distances in X?

This question was answered in the affirmative (vaguely, as usual in these types of results) in [30, 24]. This justifies, at least in principle, the application of the JLL to $P^1(A, b)$, and its replacement with the smaller-sized $P^1(TA, Tb)$. Forward-thinking readers might make the following objection. Suppose you are able to compute a tight lower bound for the number of rows m of A, for which Thm. 3.1 still held; again, you would be confronted with an LP, and again you could apply the JLL to its columns in order to make them shorter. Would this not negate the tightness of your bound for m and create a paradox?

The answer to this objection may reside in the observation that varying m in $P^1(A, b)$ induces a phase transition w.r.t. the truth value of Eq. (5) [3]. When m is at its empirical lower bound, decreasing m by a

few units will cause $P^1(A,b)$ to produce completely dense solutions arbitrarily far from the original sparse error vector, with sharply rising probability [3, Fig. 1]. I will venture to say that this is compatible with the vague nature of the JLL, full of unknown constants and guessed ε's. The JLL works in "areas" with comfortably large neighbourhoods of variation tolerance. If you are on the edge of a phase transition, it may well fail to apply. Supposing I were able to compute the universal constants exactly in the case of compressed sensing (which I am not), and that I wanted to work with a tiny ε in order to carefully control the risk of overstepping the phase transition threshold, the JLL would perhaps tell me that k would need to be as large as m or even higher (due to the huge constant C/ε^2, since the $\ln(n)$ term increases too slowly to wreak such havoc).

But I have other weapons to deploy: for example, the fact that messages are strings of characters rather than real vectors.

6 Pardoning more errors

We assume that all messages have been segmented into textual pieces of d characters (including the spaces), that a redundancy factor $R > 1$ is given, and that a certain $n \times d$ matrix Q (with $n = Rd$) is known to both sender and recipient. Every party also knows that the communication line has an error rate Δ. Let us now look at the encoding and decoding algorithm.

Given a string μ such as "`I am a string`", we have a function called $\mathsf{string2bitlist}$ which takes μ as input and returns a binary vector $w \in \{0,1\}^d$ as output, with $d = 8|\mu|$ This function works by expressing the ASCII codes of the string characters into a base-2 representation. We assume that each ASCII code is in the range $\{0,\ldots,255\}$, which means that each character takes 8 bits of binary storage to be encoded. The vector w is then considered a real vector in $\mathbb{R}^d$. The sender computes $z = Qw \in \mathbb{R}^n$, which will be sent over the Δ-noisy (and somewhat costly) communication line.

Note that $\mathsf{string2bitlist}$ has an inverse function $\mathsf{bitlist2string}$ which takes a vector $w \in \{0,1\}^d$ (where d is an integer multiple of 8), splits w into $d/8$ contiguous sub-vectors in $\{0,1\}^8$, which are then expressed in base 10 and interpreted as ASCII codes. The output of $\mathsf{bitlist2string}$ is the corresponding string μ.

We make a few assumptions specific to the receiving end: the recipient has computed an $m \times n$ matrix A where $m = n - d$ and $AQ = 0$. Moreover, the recipient has implemented two functions: $\lfloor\sigma\rceil$, which rounds the scalar σ to the nearest integer, and $\mathsf{cap}(\sigma,[L,U]) = \min(\max(\sigma,L),U)$, which restricts σ to lie in $[L,U]$. The decoding algorithm works as follows:

1. receive $\bar{z}$ on the noisy communication line;
2. compute $b = A\bar{z}$;
3. solve $P^1(A,b)$ and obtain the optimal solution x';
4. compute $z' = \bar{z} - x'$;
5. compute $w' = \mathsf{cap}(\lfloor((Q^\top Q)^{-1}Q^\top z')\rceil,[0,1])$;
6. return $\mu' = \mathsf{bitlist2string}(w')$.

We also want to evaluate the difference between the decoded message μ' and the original message μ. We therefore also compute $\mu_{\mathsf{err}} = \mathsf{dist}(\mu,\mu')$ for some "distance function" dist (for example the number of different characters). Note that, at Step 3, we know that if the assumptions of Thm. 3.1 hold, then the solution of $P^1(A,b)$ will be unique and a good approximation of the communication error vector $\bar{x} = z - z'$ (which has density Δ, and support size $\lfloor\Delta n\rceil$).

The rounding and capping operation at Step 5 is extremely forgiving to errors, in the sense that even if the number m of rows in A is smaller than required by Eq. (2), resulting in x' being dense, after the rounding and capping operation in Step 5 every component of x' in $(-\infty,0.5]$ will have been set to zero (with every other component being set to one). Since we started from binary vectors, no undue trick

is being played. We exposed our sent vector z to a lot more noise than was being logically necessary by embedding it in a real vector space; and now we are reducing that noise in the most common-sense possible way. This might allow us to guess the universal constants C, c of Thm. 4.2 carelessly, and set ε to a very tolerant 20%, and still be able to decode messages correctly. Moreover, if we are sending plain text in natural language, which comes with its own redundancy, we might even be able to tolerate some errors and still understand the message.

7 The outcome

I set up my code with $R = 4$, $\Delta = 0.08$ (in accordance with [25]), so it could solve $P^1(A, b)$ as well as $P^1(TA, Tb)$ where T is a sparse $(1, 0, -1)$ sub-Gaussian random projector with density $\alpha = 0.02$ (see page 6) and tolerance $\varepsilon = 0.2$. I used the following algorithm in order to generate appropriately sized matrices A, Q such that $AQ = 0$:

1. sample an $n \times n$ matrix M componentwise from a uniform distribution (e.g. on $[-1, 1]$): M will have full rank with probability 1 (since the probability that there should be fortuitous linear relations is proportional to the volume occupied by a subspace in the full space: zero);
2. find an eigenvector matrix of $M^\top M$: this provides an orthonormal basis of $\mathbb{R}^n$;
3. concatenate d (column) eigenvectors to make Q;
4. stack $m = n - d$ (row) eigenvectors to make A.

Clearly, $AQ = 0$ by construction, since $(Q, A^\top)$ is an orthonormal $n \times n$ matrix. The corresponding implementation is fast, since there are robust and efficient implementations of pseudorandom sampling and spectral decomposition.

I then simulated the noisy message encoding and decoding for various initial segments of the beginning of Virgil's *Æneid*'s second canto:

> *Conticuere omnes intentique ora tenebant*
> *inde toro pater Æneas sic orsus ab alto:*
> *"Infandum, regina, iubes renovare dolorem:*
> *Troianas ut opes et lamentabile regnum*
> *eruerint Danai, quæquæ ipse miserrima vidi,*
> *et quorum pars magna fui. Quis talia fando,*
> *Myrmidonum Dolopumve, aut duri miles Ulixi*
> *temperet a lacrymis? Et iam nox humida cælo*
> *præcipitat, suadentque cadentia sidera somnos.*
> *Sed si tantus amor casus cognoscere nostros,*
> *et breviter Troiæ supremum audire laborem,*
> *quamquam animus meminisse horret luctuque refugit,*
> *incipiam."*

The computational results of these tests have been reported in Table 1, where I reported the size (d and n) of each tested instance, the error between original and decoded messages for the original LP $P^1(A, b)$ ($\mu_{\mathsf{err}}^{\mathsf{org}}$) and the projected LP $P^1(TA, Tb)$ ($\mu_{\mathsf{err}}^{\mathsf{prj}}$), with the corresponding CPU times. Evidently, the redundancy ratio $R = 4$ for an error rate $\Delta = 0.08$ (used in [25]) is overkill for computation, which is always successful even with the projected LP. Since $P^1(TA, Tb)$ has fewer constraints than $P^1(A, b)$, it takes much less time to solve (see Fig. 1).

d	n	$\mu_{\text{err}}^{\text{org}}$	$\mu_{\text{err}}^{\text{prj}}$	CPU^{org}	CPU^{prj}
80	320	0	0	1.05	**0.56**
128	512	0	0	2.72	**1.10**
216	864	0	0	8.83	**2.12**
248	992	0	0	12.53	**2.53**
320	1280	0	0	23.70	**3.35**
408	1632	0	0	43.80	**4.75**

Table 1: Comparison of accuracy and CPU times taken to solve $P^1(A, b)$ and $P^1(TA, Tb)$. Each line refers to a single run over the corresponding instance

.

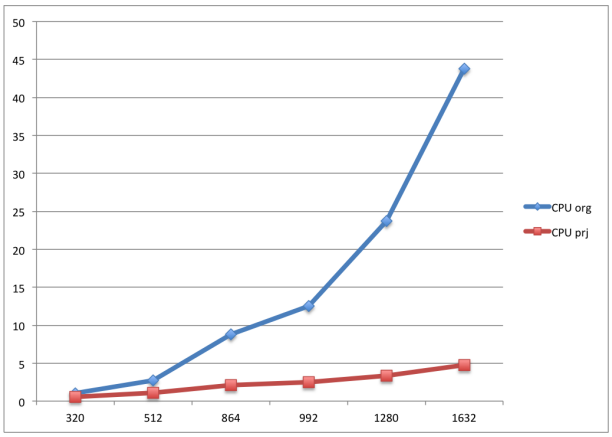

Figure 1: CPU time for original and projected LP.

8 Aiming for the impossible

If you think about it, a redundancy $R = 4$ for an error rate $\Delta = 0.08$ appears excessive. I told myself that a good communication line should work with a redundancy R that should be more or less $1 + \Delta$. So then $n = (1 + \Delta)d$. In particular, n and d would be very close (say $d = O(n)$), which would in turn make A very short and fat: specifically, the number m of rows of A would be $n - d$, which would be very close to zero. On the other hand, Thm. 3.1 requires m to be "large enough". As we have seen before, "almost zero" can be "large enough" if all sizes involved are enormous. But I wanted this protocol to work with small sizes too, which means that, in practice, m would need to be $O(n)$. So m needs to be both "close to zero" and "almost n", clearly an impossible feat.

I decided to drop the first condition: let us make m "sizable" with respect to n, say $O(n)$. Now the orthogonality condition $AQ = 0$ implies that every one of the $O(n)$ rows of A should be orthogonal to every one of the $O(n)$ columns of Q. This requires finding around $2n$ orthogonal vectors in $\mathbb{R}^n$, which is, again, impossible. Given the vagueness of the results leading to my idea, I thought it worthwhile to ask what one can obtain by relaxing "orthogonality" to "near-orthogonality".

As it happens, one can obtain quite a lot. Another way of seeing the JLL is that $\mathbb{R}^n$ can contain exponentially many almost orthogonal vectors. This follows from two corollaries of the JLL.

8.1 Corollary

Let X be a set of n points in $\mathbb{R}^m$, $\varepsilon > 0$, and C, c be two universal constants. Then there exist $k \geq \frac{C}{\varepsilon^2} \ln(n)$ and a $k \times m$ matrix T sampled componentwise from a sub-Gaussian distribution with zero mean and

standard deviation $\frac{1}{\sqrt{k}}$*, such that:*

$$\mathsf{Prob}\left[\forall x, y \in X \ |\langle Tx, Ty\rangle - \langle x, y\rangle| \leq \frac{\varepsilon}{2}(\|x\|_2^2 + \|y\|_2^2)\right] \geq 1 - 4e^{-c(\varepsilon^2-\varepsilon^3)k}.$$

8.2 Corollary
Let $k \in \mathbb{N}$, $\varepsilon > 0$, C', c *be two universal constants, and* $n = \lceil e^{\frac{\varepsilon^2}{C'}k}\rceil$*. Then*

$$\mathsf{Prob}\left[|\langle Te_i, Te_j\rangle| \leq \varepsilon\right] \geq 1 - 4e^{-c(\varepsilon^2-\varepsilon^3)k}.$$

The first corollary says that if Euclidean distances are almost preserved, then vector angles are also almost preserved (it makes sense, as preserving distances leads to a congruence, which also preserves angles). The second corollary follows from the first corollary when it is applied to $X = \{e_1, \ldots, e_n\} = B_n$, the standard basis of $\mathbb{R}^n$, such that the constant $C' \leq C$ is chosen in a way that yields $k = \lceil \frac{C'}{\varepsilon^2}\ln(n)\rceil$ in the first corollary: then the second corollary says that there is sufficient space in $\mathbb{R}^k$ to host $O(e^k)$ almost orthogonal vectors. Then the same will hold for n by symbolic replacement.

The issue is now to see whether "almost orthogonality" is as good as orthogonality for the decoding process. After all, the last step of Eq. (2) states $(AQ)w + Ax = Ax$ because $AQ = 0$. What if $\|AQ\| \leq \varepsilon$ for some small ε instead? A few tests with the almost orthogonal vectors that can be produced using TB_{2n} with an $n \times 2n$ random projector T led to disappointing results.

There is one further murky corner where I could try and scrape some more advantage for my impossible method: all I need, in order to decode $\bar{z}$ at least approximately, is that:

- A is an $m \times n$ matrix sampled componentwise from $\mathsf{N}(0, 1)$ (A will be used to solve $P^1(TA, Tb)$, where $n = \lfloor(1 + \Delta)m\rceil$ and T is a $k \times m$ random projector);
- Q is an $n \times m$ matrix with $AQ \approx 0$: in other words, requiring that all the columns of Q should be orthogonal is overkill.

I therefore formulated a LP in order to decide the components of Q. For every $\ell \leq m$ let $q_\ell \in \mathbb{R}^n$ be the ℓ-th column of Q. Then for each $\ell \leq m$ I need to decide q_ℓ so that: (a) $Q = (q_\ell \mid \ell \leq m)$ has full rank, and (b) $Aq_\ell \approx 0$. I attempted to enforce (a) by optimizing a random objective function, and (b) by imposing the linear constraints $Aq_\ell = 0$. The following LP (Eq. 9) must be solved m times (each time with different random cost coefficients): the ℓ-th solution q_ℓ must be stored in the corresponding column of Q (for $\ell \leq m$).

$$\left.\begin{array}{rl} \max\limits_{q_\ell \in [-1,1]^n} & \sum\limits_{j \leq n} \mathsf{Uniform}(-1, 1)q_{\ell j} \\ \forall i \leq m & \sum\limits_{j \leq n} A_{ij}q_{\ell j} = 0. \end{array}\right\} \tag{9}$$

Eq. (9) can be solved with any LP algorithm. In my experiments, I used the interior point method without crossing over to the simplex algorithm at the end, since I only needed feasible solutions.

Let me add that solving these LPs in order to create Q can take a considerable CPU time: definitely more than solving $P^1(A, b)$. But these matrices can be pre-computed for given sizes, and shared before communications occur. So I have not taken these CPU times into account in my tests.

I was expecting Eq. (9) to lead to the inevitable trivial solution $Q = 0$ (which negates the requirement of Q having full rank), and was prepared to increase the feasibility tolerances of the CPLEX solver I was using [16]. Instead, CPLEX provided me with a magnificent (and impossible) matrix Q with full rank, and such that $AQ = 0$ with precision 10^{-10}. When I fine-tuned my rank evaluation code with sufficiently small numerical tolerances I discovered the expected rank deficiency, which was not acceptable since I would need $Q^\top Q$ to be truly invertible for pseudoinverse purposes. So I switched Q with A in my process: I sampled Q randomly, obtaining a full rank Q with probability 1, and then applied Eq. (9) to the rows

of A instead. A would not be a normally sampled matrix satisfying the assumption of Thm. (3.1), but I hoped that the non-necessary nature of that theorem (which only lists sufficient conditions to its conclusion) would allow my A, with full rank up to 10^{-10} precision and surely rank-deficient beyond, but almost orthogonal to a random matrix, to do the trick nonetheless.

The results of my experiments are in Table 2: on top of reporting the error measures for the accuracy of the two methods (original and projected LP) and the CPU times, I have also tried different values of Δ.

m	n	Δ'	$\mu_{\text{err}}^{\text{org}}$	$\mu_{\text{err}}^{\text{prj}}$	CPU^{org}	CPU^{prj}
328	426	0.3	182	**15**	2.45	**1.87**
328	426	0.3	154	**0**	2.20	**1.49**
328	459	0.4	**0**	1	4.47	**2.45**
328	459	0.4	**5**	17	2.86	**1.46**
328	492	0.5	60	**0**	4.53	**1.18**
328	492	0.5	34	**0**	5.38	**1.18**
328	590	0.8	14	**0**	8.30	**1.41**
328	590	0.8	107	**4**	6.76	**1.43**
1896	2465	0.3	**0**	2	29.67	**17.13**

Table 2: Comparison of accuracy and CPU time taken to solve $P^1(A, b)$ and $P^1(TA, Tb)$ with the impossible choice of A. Each instance was solved twice, and the results are reported for each of the two runs (black and grey entries).

Table 2 tells an exhilarating and disturbing story. In most cases, solving the original LP $P^1(A, b)$ leads to Thm. 3.1 failing, as can be ascertained from the high values of the error $\mu_{\text{err}}^{\text{org}}$. Solving the projected LP $P^1(TA, Tb)$, however, sometimes just does the trick. It is exhilarating to think that using one theorem wrongly is bad, but that using two of them wrongly is fine. It is even more satisfying to think that I am exploiting a redundancy ratio as small as the error ratio, and managing to reconstruct messages "reasonably well". On the other hand, it is disturbing to think that, in order to achieve these results, I have misused Thm. 3.1, Thm. 4.2, basic linear algebra, and the LP machinery in ways that I really should not have done: *eppur si muove*.

9 Conclusion

The lack of a formal (if vague) theoretical justification for the apparent success of my impossible method for decoding messages is part of the reason why this paper is written with an unusual style. Physicists and engineers working in signal processing, however, had used $P^1(A, b)$ instead of $P^0(A, b)$ in applications for years, before a theory was constructed. So I am not despairing that one day someone will find a reason why this impossible method works.

References

[1] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66:671–687, 2003.

[2] N. Alon and J. Spencer. *The probabilistic method*. Wiley, Hoboken, NJ, 2008.

[3] D. Amelunxen, M. Lotz, M. McCoy, and J. Tropp. Living on the edge: phase transitions in convex programs with random data. *Information and Inference: A Journal of the IMA*, 3:224–294, 2014.

[4] R. Arriaga and S. Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 63:161–182, 2006.

[5] A. Barvinok. Math 710: Measure Concentration, 2005. Class notes.

[6] E. Candès. The mathematics of sparsity. In S.Y. Jang, Y.R. Kim, D.-W. Lee, and I. Yie, editors, *Proceedings of the International Congress of Mathematicians*, volume I. Kyung Moon SA, Seoul, 2014.

[7] E. Candès and T. Tao. Decoding by Linear Programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[8] E. Candès and T. Tao. Reflections on compressed sensing. *IEEE Information Theory Society Newsletter*, 58(4):14–17, 2008.

[9] C. D'Ambrosio, L. Liberti, P.-L. Poirion, and K. Vu. Random projections for quadratic programs. *Mathematical Programming B*, 183:619–647, 2020.

[10] S. Damelin and W. Miller. *The mathematics of signal processing*. CUP, Cambridge, 2012.

[11] S. Dasgupta and A. Gupta. An elementary proof of a theorem by Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22:60–65, 2002.

[12] P. Delsarte. Bounds for unrestricted codes by linear programming. *Philips Research Reports*, 27:272–289, 1972.

[13] P. Delsarte, J.M. Goethals, and J.J. Seidel. Spherical codes and designs. *Geometriæ Dedicata*, 6:363–388, 1977.

[14] S. Dirksen. Dimensionality reduction with subgaussian matrices: A unified theory. *Foundations of Computational Mathematics*, 16:1367–1396, 2016.

[15] M. Garey and D. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman and Company, New York, 1979.

[16] IBM. *ILOG CPLEX 12.8 User's Manual*. IBM, 2017.

[17] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In G. Hedlund, editor, *Conference in Modern Analysis and Probability*, volume 26 of *Contemporary Mathematics*, pages 189–206, Providence, RI, 1984. AMS.

[18] D. Kane and J. Nelson. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM*, 61(1):4, 2014.

[19] A. Khintchine. Über dyadische brüche. *Mathematische Zeitschrift*, 18(1):109–116, 1923.

[20] K. Kunen. *Set Theory. An Introduction to Independence Proofs*. North Holland, Amsterdam, 1980.

[21] M. Ledoux. *The concentration of measure phenomenon*. Number 89 in Mathematical Surveys and Monographs. AMS, Providence, RI, 2005.

[22] L. Liberti. Mathematical programming bounds for kissing numbers. In A. Sforza and C. Sterle, editors, *Optimization and Decision Science: Methodologies and Applications (AIRO-ODS17)*, volume 217 of *Proceedings in Mathematics and Statistics*, pages 213–222, New York, 2017. Springer.

[23] L. Liberti. Distance geometry and data science. *TOP*, 28:271–339, 220.

[24] L. Liberti, B. Manca, and P.-L. Poirion. Practical performance of random projections in linear programming. In C. Schulz and B. Uçar, editors, *Proceedings of the 20th International Symposium on Experimental Algorithms (SEA22)*, volume 233 of *LIPIcs*, Schloß Dagstuhl, 2022. Dagstuhl Publishing.

[25] J. Matoušek and B. Gärtner. *Understanding and using Linear Programming*. Springer, Berlin, 2007.

[26] A. Moitra. *Algorithmic aspects of Machine Learning*. CUP, Cambridge, 2018.

[27] J.-L. Starck, M. Elad, and D. Donoho. Image decomposition via the combination of sparse representations and a variational approach. *IEEE Transactions on Image Processing*, 14(10):1570–1582, 2005.

[28] S. Venkatasubramanian and Q. Wang. The Johnson-Lindenstrauss transform: An empirical study. In *Algorithm Engineering and Experiments*, volume 13 of *ALENEX*, pages 164–173, Providence, RI, 2011. SIAM.

[29] R. Vershynin. *High-dimensional probability*. CUP, Cambridge, 2018.

[30] K. Vu, P.-L. Poirion, and L. Liberti. Random projections for linear programming. *Mathematics of Operations Research*, 43(4):1051–1071, 2018.