Fast Johnson-Lindenstrauss

Preetum Nakkiran
May 19, 2016

The Johnson-Lindenstrauss (JL) Transform says that, informally, we can embed high-dimensional points into a much lower dimension, while still preserving their pairwise distances. In this post we will start with the classical JL transform, then focus on the Fast JL Transform (FJLT) by Ailon and Chazelle [AC09], which achieves the JL embedding more efficiently (w.r.t. runtime and randomness). We will develop the FJLT from the perspective of "preconditioning" a sparse estimator, which comes with nice intuition from Fourier duality. We conclude by mentioning more recent developments in this area (faster/sparser/derandomizeder).

Motivation. Aside from being an interesting structural result, the JL transform also has algorithmic and machine-learning applications. Eg, any application that only depends on approximate pairwise distances (such as nearest-neighbors), or even on pairwise inner-products[1] (such as kernel methods) may equivalently work with the dimensionality-reduced version of their input. This also has applications in sketching and scarification.

1 Classical JL

The JL embedding theorem is:

Theorem 1. *Given points $x_1, \dots, x_n \in \mathbb{R}^d$, and $\varepsilon > 0$, there exists an embedding $f : \mathbb{R}^d \to \mathbb{R}^k$ such that*

$$\forall i, j : \quad (1-\varepsilon)||x_i - x_j||_2 \leq ||f(x_i) - f(x_j)||_2 \leq (1+\varepsilon)||x_i - x_j||_2$$

and $k = O(\varepsilon^{-2} \log n)$.

That is, the embedding roughly preserves pairwise distances in ℓ_2. Think of the regime $n \gg d \gg k$. Note that the target dimension $k = O(\varepsilon^{-2} \log n)$ is (perhaps surprisingly) independent of the source dimension d, and only logarithmic in the number of points n.

In fact, a random linear map works as an embedding (w.h.p.). This is established by the following lemma.

Lemma 1. *For any $\delta > 0$, set $k = O(\varepsilon^{-2} \log(1/\delta))$, and let $A \in \mathbb{R}^{k \times d}$ be a random matrix with iid normal $\mathcal{N}(0, 1/k)$ entries. Then*

$$\forall x \in \mathbb{R}^d : \quad \Pr_A \left[\; ||Ax||_2^2 \in (1 \pm \varepsilon)||x||_2^2 \; \right] \geq 1 - \delta$$

That is, a random matrix preserves the norm of vectors with good probability. To see that this implies the JL Theorem, consider applying the matrix A on the $O(n^2)$ vectors of pairwise differences $(x_i - x_j)$. For fixed i, j, the lemma implies that $||A(x_i - x_j)|| \approx ||x_i - x_j||$ except w.p. δ. Thus, setting $\delta = 1/n^3$ and union bounding, we have that A preserves the norm of *all* differences $(x_i - x_j)$ with high probability. Letting the embedding map $f = A$, we have

$$\forall i, j : \quad ||f(x_i) - f(x_j)|| = ||Ax_i - Ax_j|| = ||A(x_i - x_j)|| \approx ||x_i - x_j||$$

as desired. [2]

Runtime. The runtime of embedding a single vector with this construction (setting $\delta = 1/n^3$ and $\varepsilon = O(1)$) is $O(dk) = O(d \log n)$. In the Fast JL below, we will show how to do this in time almost $O(d \log d)$.

[1] Because, if the norms $||x_i - x_j||$, $||x_i||$, and $||x_j||$ are approximately preserved, then so is the inner-product $\langle x_i, x_j \rangle$.

[2] Note that it was important that f be linear for us to reduce preserving pairwise-distances to preserving norms.

We will now prove Lemma 1. First, note that our setting is scale-invariant, so it suffices to prove the lemma for all unit vectors x.

As a warm-up, let $g \in \mathbb{R}^d$ be a random vector with entires iid $\mathcal{N}(0,1)$, and consider the inner-product

$$Y := \langle g, x \rangle$$

(for a fixed unit vector $x \in \mathbb{R}^d$). Notice the random variable Y has expectation $\mathbb{E}[Y] = 0$, and variance

$$\mathbb{E}[Y^2] = \mathbb{E}[\langle g, x \rangle^2] = ||x||^2$$

Thus, Y^2 is an unbiased estimator for $||x||^2$. Further, it concentrates well: assuming (wlog) that $||x|| = 1$, we have $Y \sim \mathcal{N}(0,1)$, so Y^2 has constant variance (and more generally, is subguassian[3] with a constant parameter).

This would correspond to a random linear projection into 1 dimension, where the matrix A in Lemma 1 is just $A = g^T$. Then $||Ax||^2 = \langle g, x \rangle^2 = Y^2$. However, this estimator does not concentrate well enough (we want tail probability δ to eventually be inverse-poly, not a constant).

We can get a better estimator for $||x||^2$ by averaging many iid copies. In particular, for any iid subgaussian random variables Z_i, with expectation 1 and subguassian parameter σ, the Hoeffding bound gives

$$\Pr\left[\left|\left(\frac{1}{k}\sum_{i=1}^{k} Z_i\right) - 1\right| > \varepsilon\right] \leq e^{-\Omega(k\varepsilon^2/\sigma^2)}$$

Applying this for $Z_i = Y_i^2$ and $\sigma = O(1)$, we can set $k = O(\varepsilon^{-2}\log(1/\delta))$ so the above tail probability is bounded by δ.

This is exactly the construction of Lemma 1. Each row of A is $\frac{1}{\sqrt{k}} g_i^T$, where $g_i \in \mathbb{R}^d$ is iid $\mathcal{N}(0,1)$. Then

$$||Ax||^2 = \sum_{i=1}^{k} \langle \frac{1}{\sqrt{k}} g_i, x \rangle^2 = \frac{1}{k}\sum_{i=1}^{k} \langle g_i, x \rangle^2 = \frac{1}{k}\sum_{i=1}^{k} Y_i^2$$

And thus, for $||x|| = 1$,

$$\Pr\left[\left|||Ax||^2 - 1\right| > \varepsilon\right] \leq \delta$$

as desired in Lemma 1. ■

To recap, the key observation was that if we draw $g \sim N(0, I_d)$, then $\langle g, x \rangle^2$ is a "good" estimator of $||x||^2$, so we can average $O(\varepsilon^{-2}\log(1/\delta))$ iid copies and get an estimator within a multiplicative factor $(1 \pm \varepsilon)$ with probability $\geq 1 - \delta$. Filling the transform matrix A with guassians is clearly not necessary; any distribution with iid entries that are subguassian would work, with the same proof as above. For example, picking each entry in A as iid $\pm\frac{1}{\sqrt{k}}$ would work.

We can now think of different JL transforms as constructing different estimators of $||x||^2$. For example, can we draw from a distribution such that g is sparse? (Not quite, but with some "preconditioning" this will work, as we see below).

[3] Subguassian with parameter σ basically means tail probabilities behave as a guassian with variance σ^2 would behave. Formally, a zero-mean random variable X is "subgaussian with parameter σ" if: $\mathbb{E}[e^{\lambda X}] \leq e^{\sigma^2\lambda^2/2}$ for all $\lambda \in \mathbb{R}$.

2 Fast JL

2.1 First Try: Coordinate Sampling

As a (bad) first try, consider the estimator that randomly samples a coordinate of the given vector x, scaled appropriately. That is,

$$Y := \sqrt{d}\, x_j \quad \text{for uniformly random coordinate } j \in [d]$$

Equivalently, draw a random standard basis vector e_j, and let $Y := \sqrt{d}\langle e_j, x\rangle$.

Notice that Y^2 has the right expectation:

$$\mathbb{E}[Y^2] = \mathop{\mathbb{E}}_{j}[(\sqrt{d}x_j)^2] = d \mathop{\mathbb{E}}_{j}[x_j^2] = ||x||^2$$

However, it does not concentrate well. The variance is $Var[Y^2] = Var[dx_j^2] = d^2 Var[x_j]$. If x is a standard basis vector (say $x = e_1$), then this could be as bad as $Var[Y^2] \approx d$. This is bad, because it means we would need to average $\Omega(d)$ iid samples to get a sufficiently good estimator, which does not help us in reducing the dimension of $x \in \mathbb{R}^d$.

The bad case in the above analysis is when x is very concentrated/sparse, so sampling a random coordinate of x is a poor estimator of its magnitude. However, if x is very "spread out", then sampling a random coordinate would work well. For example, if all entries of x are bounded by $\pm O(\sqrt{\frac{1}{d}})$, then $Var[Y^2] = O(1)$, and taking iid copies of this estimator would work. This would be nice for runtime, since randomly sampling a coordinate can be done quickly (it is not a dense inner-product).

Thus, if we can (quickly) "precondition" our vector x to have $||x||_\infty \le O(\sqrt{\frac{1}{d}})$, we could then use coordinate sampling to achieve a fast JL embedding. We won't quite achieve this, but we will be able to precondition such that $||x||_\infty \le O(\sqrt{\frac{\log(d/\delta)}{d}})$, as described in the next section. With this in mind, we will need the following easy claim (that with the weaker bound on ℓ_∞, coordinate sampling works to reduce the dimension to almost our target dimension).

Lemma 2. *Let $t = \Theta(\varepsilon^{-2}\log(1/\delta)\log(d/\delta))$. Let $S \in \mathbb{R}^{t\times d}$ be a matrix with rows $s_i := \sqrt{\frac{d}{t}} e_{j_i}^{(i)}$, where each $j_i \in [d]$ is an iid uniform index. (That is, each row of S randomly samples a coordinate, scaled appropriately). Then, for all x s.t $||x||_2 = 1$ and $||x||_\infty \le O(\sqrt{\frac{\log(d/\delta)}{d}})$, we have*

$$\Pr_S[|||Sx||^2 \in 1 \pm \varepsilon] \ge 1 - \delta$$

Proof.

$$||Sx||^2 = \sum_{i=1}^{t} \langle \sqrt{\frac{d}{t}} e_{j_i}, x\rangle^2 = \frac{1}{t}\sum_{i=1}^{t} d(x_{j_i})^2$$

Then, the r.vs $\{d(x_{j_i})\}$ are iid and absolutely bounded by $O(\sqrt{\log(d/\delta)})$, so by Chernoff-Hoeffding and our choice of t,

$$\Pr[|\frac{1}{t}\sum_{i=1}^{t} d(x_{j_i})^2 - 1| > \varepsilon] \le e^{-\Omega(\varepsilon^2 t/\log(d/\delta))} \le \delta$$

■

2.2 FJLT: Preconditioning with random Hadamard

The main idea of FJLT is that we can quickly precondition vectors to be "smooth", by using the Fast Hadamard Transform.

Recall the $d \times d$ Hadamard transform H_d (for d a power of 2) is defined recursively as

$$H_1 := 1, \quad H_{2d} := \frac{1}{\sqrt{2}} \begin{bmatrix} H_d & H_d \\ H_d & -H_d \end{bmatrix}$$

More explicitly, $H_d[i,j] = \frac{1}{\sqrt{d}}(-1)^{\langle i,j \rangle}$ where indices $i, j \in \{0,1\}^{\log d}$, and the inner-product is mod 2. The Hadamard transform is just like the discrete Fourier transform[4] : it can be computed in time $O(d \log d)$ by recursion, and it is an orthonormal transform.

Intuitively, the Hadamard transform may be useful to "spread out" vectors, since Fourier transforms take things that are sparse/concentrated in time-domain to things that are spread out in frequency domain (by time-frequency duality/Uncertainty principle). Unfortunately this won't quite work, since duality goes both ways: It will also take vectors that are already spread out and make them sparse.

To fix this, it turns out we can first randomize the signs, then apply the Hadamard transform.

Lemma 3. *Let H_d be the $d \times d$ Hadamard transform, and let D be a random diagonal matrix with iid ± 1 entries on the diagonal. Then,*

$$\forall x \in \mathbb{R}^d, ||x|| = 1 : \quad \Pr_D[||H_d D x||_\infty > \Omega(\sqrt{\frac{\log(d/\delta)}{d}})] \leq \delta$$

We may expect something like this to hold: randomizing the signs of x corresponds to pointwise-multiplying by random white noise. White noise is spectrally flat, and multiplying by it in time-domain corresponds to convolving by its (flat) spectrum in frequency domain. Thus, multiplying by D should "spread out" the spectrum of x. Applying H_d computes this spectrum, so should yield a spread-out vector.

The above intuition seems messy to formalize but the proof is surprisingly simple. [5]

Proof of Lemma 3. Consider the first entry of $H_d D x$. Let $D = diag(a_1, a_2, \ldots, a_d)$ where a_i are iid ± 1. The first row of H_d is $\frac{1}{\sqrt{d}} \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix}$, so

$$(H_d D x)[1] = \frac{1}{\sqrt{d}} \sum_i a_i x_i$$

Here, the x_i are fixed s.t. $||x||_2 = 1$, and the $a_i = \pm 1$ iid. Thus we can again bound this by Hoeffding [6] (surprise),

$$\Pr[|\sum_{i=1}^{d} a_i \frac{x_i}{\sqrt{d}}| > \eta] \leq e^{-\Omega(\eta^2 d / ||x||_2^2)}$$

For $\eta = \Omega(\sqrt{\frac{\log(d/\delta)}{d}})$, this probability is bounded by $(\frac{\delta}{d})$. Moreover, the same bound applies for all coordinates of $H_d D x$, since all rows of H_d have the form $\frac{1}{\sqrt{d}} \begin{bmatrix} \pm 1 & \pm 1 & \ldots & \pm 1 \end{bmatrix}$. Thus, union bound over d coordinates establishes the lemma. ■

[4] Indeed, it is exactly the Fourier transform over the group $(\mathbb{Z}_2)^n$. For more on Fourier transforms over abelian groups, see for example Luca's notes.

[5] This proof presented slightly differently from the one in Alon-Chazelle, but the idea is the same.

[6] The following form of Hoeffding bound is useful (it follows directly from Hoeffding for subgaussian variables, but is also a corollary of Azuma-Hoeffding): For iid zero-mean random variables Z_i, absolutely bounded by 1, $\Pr[|\sum c_i Z_i| > \varepsilon] \leq 2exp(-\frac{\varepsilon^2}{2\sum_i c_i^2})$.

2.3 The Full Fast JL Transform

This presentation of FJLT is due to Jelani Nelson; see the notes [Nel10].

Putting all the pieces together, the FJLT is defined as:

$$A = JSH_dD$$

or,

$$A: \quad \mathbb{R}^d \xrightarrow{D} \mathbb{R}^d \xrightarrow{H_d} \mathbb{R}^d \xrightarrow{S} \mathbb{R}^t \xrightarrow{J} \mathbb{R}^k$$

where

- S: the sparse coordinate-sampling matrix of Lemma 2
- H_d: the $d \times d$ Hadamard transform.
- D: diagonal iid ± 1.
- J: a dense "normal" JL matrix (iid Gaussian entries).

For parameters

- $t = \Theta(\varepsilon^{-2} \log(1/\delta) \log(d/\delta))$
- $k = \Theta(\varepsilon^{-2} \log(1/\delta))$

That is, we first precondition with the randomized Hadamard transform, then sample random coordinates (which does most of the dimensionality reduction), then finally apply a normal JL transform to get rid of the last $\log(d/\delta)$ factor in the dimension.

Correctness. Since the matrix D and the Hadamard transform are isometric, they do not affect the norms of vectors. Then, after the preconditioning, Lemma 2 guarantees that S only affects norms by $(1 \pm \varepsilon)$, and Lemma 1 guarantees that the final step is also roughly isometric. These steps fail w.p. δ, so the final transform affects norms by at most say $(1 \pm 3\varepsilon)$ except w.p. 3δ. This is sufficient to establish the JL embedding.

Runtime. For computing a JL embedding (ie, setting $\delta = 1/n^3, \varepsilon = O(1)$), the time to embed a single vector is $O(d \log d + \log^3 n)$.

3 Closing Remarks

Optimality. The target dimension given by the JL construction is known to be optimal. That is, one cannot embed n points into dimension less than $k = \Omega(\varepsilon^{-2} \log n)$ with distortion ε. The first near-optimal lower-bound, in [Alo03, Section 9] works by showing upper-bounds on the number of nearly-orthogonal vectors in a given dimension (so a too-good embedding of orthogonal vectors would violate this bound). A more recent, optimal bound, is in [KMN11, Section 6]. They actually show optimality of the JL Lemma (that is, restricting to linear embeddings), which works (roughly) by arguing that if the target dimension is too small, then the kernel is too big, so a random vector is likely to be very distorted.

Recent Advances. Note that the FJLT is fast, but is not *sparse*. We may hope that embedding a sparse vector x will take time proportional to the sparsity of x. A major result in this area was the sparse JL construction of [KN14]; see also the notes [Nel10]. There is also work in derandomized JL, see for example [KMN11].

I'll stop here, since I haven't read these works yet, but perhaps we will revisit this another time.
This post was derived from my talk at Berkeley theory retreat, on the theme of "theoretical guarantees for machine learning."

References

[AC09] Nir Ailon and Bernard Chazelle. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009. URL: `https://www.cs.princeton.edu/~chazelle/pubs/FJLT-sicomp09.pdf`.

[Alo03] Noga Alon. Problems and results in extremal combinatorics, part i. *Discrete Math*, 273:31–53, 2003. URL: `http://www.tau.ac.il/~nogaa/PDFS/extremal1.pdf`.

[KMN11] Daniel Kane, Raghu Meka, and Jelani Nelson. Almost optimal explicit johnson-lindenstrauss families. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 628–639. Springer, 2011. URL: `http://people.seas.harvard.edu/~minilek/papers/derand_jl.pdf`.

[KN14] Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):4, 2014. URL: `https://arxiv.org/pdf/1012.1577v6.pdf`.

[Nel10] Jelani Nelson. Johnson-lindenstrauss notes. Technical report, Technical report, MIT-CSAIL, 2010. URL: `http://web.mit.edu/minilek/www/jl_notes.pdf`.