Sparser Johnson-Lindenstrauss Transforms

Daniel M. Kane[*] Jelani Nelson[†]

Abstract

We give two different Johnson-Lindenstrauss distributions, each with column sparsity $s = \Theta(\varepsilon^{-1}\log(1/\delta))$ and embedding into optimal dimension $k = O(\varepsilon^{-2}\log(1/\delta))$ to achieve distortion $1\pm\varepsilon$ with probability $1-\delta$. That is, only an $O(\varepsilon)$-fraction of entries are non-zero in each embedding matrix in the supports of our distributions. These are the first distributions to provide $o(k)$ sparsity for all values of ε, δ. Previously the best known construction obtained $s = \tilde{\Theta}(\varepsilon^{-1}\log^2(1/\delta))$[1] [Dasgupta-Kumar-Sarlós, STOC 2010][2]. In addition, one of our distributions can be sampled from a seed of $O(\log(1/\delta)\log d)$ uniform random bits. Some applications that use Johnson-Lindenstrauss embeddings as a black box, such as those in approximate numerical linear algebra ([Sarlós, FOCS 2006], [Clarkson-Woodruff, STOC 2009]), require exponentially small δ. Our linear dependence on $\log(1/\delta)$ in the sparsity is thus crucial in these applications to obtain speedup.

1 Introduction

The Johnson-Lindenstrauss lemma states:

Lemma 1.1. (JL Lemma [18]) *For any integer $d > 0$, and any $0 < \varepsilon, \delta < 1/2$, there exists a probability distribution on $k \times d$ real matrices for $k = \Theta(\varepsilon^{-2}\log(1/\delta))$ such that for any $x \in \mathbb{R}^d$ with $\|x\|_2 = 1$, $\mathbf{Pr}_S[|\|Sx\|_2^2 - 1| > \varepsilon] < \delta$.*

Proofs of the JL lemma can be found in [1, 5, 6, 11, 12, 16, 18, 20, 24]. The value of k in the JL lemma is optimal [17] (also see a later proof in [19]).

The JL lemma is a key ingredient in the JL flattening theorem, which states that any n points in Euclidean space can be embedded into $O(\varepsilon^{-2}\log n)$ dimensions so that all pairwise Euclidean distances are preserved up to $1 \pm \varepsilon$. The JL lemma is a useful tool for speeding up solutions to several high-dimensional problems: closest pair, nearest neighbor, diameter, minimum spanning tree, etc. It also speeds up some clustering and string processing algorithms, and can further be used to reduce the amount of storage required to store a dataset, e.g. in streaming algorithms. Recently it has also found applications in approximate numerical algebra problems such as linear regression and low-rank approximation [9, 27]. See [15, 29] for further discussions on applications.

Standard proofs of the JL lemma take a distribution over dense matrices (e.g. i.i.d. Gaussian or Bernoulli entries), and thus performing the embedding naïvely takes $O(k \cdot \|x\|_0)$ time where x has $\|x\|_0$ non-zero entries. Several works have devised other distributions which give faster embedding times [2, 3, 4, 14, 23, 31], but all these methods require $\Omega(d)$ embedding time even for sparse vectors (even when $\|x\|_0 = 1$). This feature is particularly unfortunate in streaming applications, where a vector x receives coordinate-wise updates of the form $x \leftarrow x + v \cdot e_i$ in a data stream, so that to maintain some linear embedding Sx of x we should repeatedly calculate Se_i during updates. Since $\|e_i\|_0 = 1$, even the naïve $O(k \cdot \|e_i\|_0)$ embedding time method is faster than these approaches.

Even aside from streaming applications, several practical situations give rise to vectors with $\|x\|_0 \ll d$. For example, a common similarity measure for comparing text documents in data mining and information retrieval is cosine similarity [26], which is approximately preserved under any JL embedding. Here, a document is represented as a bag of words with the dimensionality d being the size of the lexicon, and we usually would not expect any single document to contain anywhere near d distinct words (i.e., we expect sparse vectors). In networking applications, if $x_{i,j}$ counts bytes sent from source i to destination j in some time interval, then d is the total number of IP pairs, whereas we would not expect most pairs of IPs to communicate with each other.

One way to speed up embedding time in the JL lemma for sparse vectors is to devise a distribution over sparse embedding matrices. This was first

[*]dankane@math.stanford.edu

[†]minilek@princeton.edu

[1]We say $g = \tilde{\Omega}(f)$ when $g = \Omega(f/\mathrm{polylog}(f))$, $g = \tilde{O}(f)$ when $g = O(f \cdot \mathrm{polylog}(f))$, and $g = \tilde{\Theta}(f)$ when $g = \tilde{\Omega}(f)$ and $g = \tilde{O}(f)$ simultaneously.

[2]The authors originally showed a bound of $s = \tilde{O}(\varepsilon^{-1}\log^3(1/\delta))$, but tighter analyses of their construction were later given in [Kane-Nelson, CoRR abs/1006.3585] and [Braverman-Ostrovsky-Rabani, CoRR abs/1011.2590].

investigated in [1], which gave a JL distribution where only one third of the entries of each matrix in its support was non-zero, without increasing the number of rows k from dense constructions. Later, the works [8, 28] gave a distribution over matrices with only $O(\log(1/\delta))$ non-zero entries per column, but the algorithm for estimating $\|x\|_2$ given the linear sketch then relied on a median calculation, and thus these schemes did not provide an embedding into ℓ_2. In several applications, such as nearest-neighbor search [16] and approximate numerical linear algebra [9, 27], an embedding into a normed space or even ℓ_2 itself is required, and thus median estimators cannot be used. Recently Dasgupta, Kumar, and Sarlós [10], building upon work in [32], gave a JL distribution over matrices where each column has at most $s = \tilde{O}(\varepsilon^{-1}\log^3(1/\delta))$ non-zero entries, thus speeding up the embedding time to $O(s \cdot \|x\|_0)$. This "DKS construction" requires $O(ds \log k)$ bits of random seed to sample a matrix from their distribution. The work of [10] left open two main directions: (1) understand the sparsity parameter s that can be achieved in a JL distribution, and (2) devise a sparse JL transform distribution which requires few random bits to sample from, for streaming applications where storing a long random seed requires prohibitively large memory.

The previous work [20] of the current authors made progress on both these questions by showing $\tilde{O}(\varepsilon^{-1}\log^2(1/\delta))$ sparsity was achievable by giving an alternative analysis of the scheme of [10] which also only required $O(\log(1/(\varepsilon\delta))\log d)$ seed length. The work of [6] later gave a tighter analysis under the assumption $\varepsilon < 1/\log^2(1/\delta)$, improving the sparsity and seed length further by $\log(1/\varepsilon)$ and $\log\log(1/\delta)$ factors in this case. In the appendix we show that the DKS scheme *requires* $s = \tilde{\Omega}(\varepsilon^{-1}\log^2(1/\delta))$, and thus a departure from their construction is required to obtain better sparsity. For a discussion of other previous work concerning the JL lemma see [20].

Main Contribution: In this work, we give two new constructions which achieve sparsity $s = \Theta(\varepsilon^{-1}\log(1/\delta))$ for ℓ_2 embedding into optimal dimension $k = O(\varepsilon^{-2}\log(1/\delta))$. This is the first sparsity bound which is always $o(k)$, even for small δ. One of our distributions requires $O(\log(1/\delta)\log d)$ uniform random bits to sample a random matrix.

We also describe variations on our constructions which achieve sparsity $\tilde{O}(\varepsilon^{-1}\log(1/\delta))$, but which have much simpler analyses. We describe our simpler constructions in Section 3, and our better constructions in Section 4. We show in the appendix that our analyses are tight up to a constant factor, so any further improvement in sparsity would require a different construction.

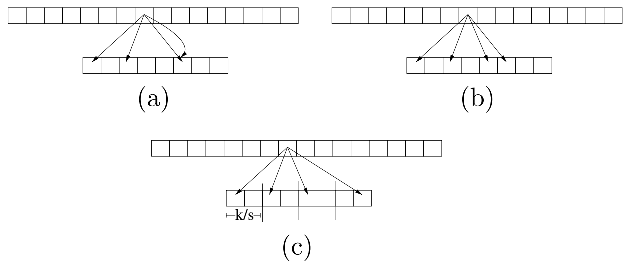

Figure 1: In all three constructions above, a vector in $\mathbb{R}^d$ is projected down to $\mathbb{R}^k$. Figure (a) is the DKS construction in [10], and the two constructions we give in this work are represented in (b) and (c). The out-degree in each case is s, the sparsity.

In Section 5 we discuss how our new schemes speed up the numerical linear algebra algorithms in [9] for approximate linear regression and best rank-k approximation in the streaming model of computation. In Section 5 we show that a wide range of JL distributions automatically provides sketches for approximate matrix product as defined in [27]. While [27] also showed this, it lost a logarithmic factor in the target dimension due to a union bound in its reduction; the work of [9] avoided this loss, but only for the JL distribution of random sign matrices. We show a simple and general reduction which incurs no loss in parameters. Plugging in our sparse JL transform then yields faster linear algebra algorithms using the same space.

1.1 Our Approach Our constructions are depicted in Figure 1. Figure 1(a) represents the DKS construction of [10] in which each item is hashed to s random target coordinates with replacement. Our two schemes achieving $s = \Theta(\varepsilon^{-1}\log(1/\delta))$ are as follows. Construction (b) is much like (a) except that we hash coordinates s times *without* replacement. In (c), the target vector is divided up into s contiguous blocks each of equal size k/s, and a given coordinate in the original vector is hashed to a random location in each block (essentially this is the CountSketch of [8], though we use a higher degree of independence in our hash functions). In all cases (a), (b), and (c), we randomly flip the sign of a coordinate in the original vector and divide by $\sqrt{s}$ before adding it in any location in the target vector.

We give two different analyses for both our constructions (b) and (c). Since we consider linear embeddings, without loss of generality we can assume $\|x\|_2 = 1$. Look at the random variable $Z = \|Sx\|_2^2 - 1$, where S is a random matrix in the JL distribution. Our proofs all use Markov's bound on the ℓth moment Z^ℓ to give $\mathbf{Pr}[|Z| > \varepsilon] < \varepsilon^{-\ell} \cdot \mathbf{E}[Z^\ell]$ for $\ell = \log(1/\delta)$ an even integer. The task is then to bound $\mathbf{E}[Z^\ell]$. In our first approach, we observe that Z is a quadratic form in the random signs, and thus its moments can be bounded via the Hanson-Wright inequality [13]. This analysis turns out to reveal that the hashing to coordinates in the target vector need not be done randomly, but can in fact be specified by any sufficiently good code. Specifically, in (b) it suffices for the columns of the embedding matrix (ignoring the random signs and division by $\sqrt{s}$) to be codewords in a constant-weight binary code of weight s and minimum distance $s - O(s^2/k)$. In (c), if for each $i \in [d]$ we let C_i be a length-s vector with entries in $[k/s]$ specifying where coordinate i is mapped to in each block, it suffices for $\{C_i\}_{i=1}^d$ to be a code of minimum distance $s - O(s^2/k)$. It is fairly easy to see that if one wants a deterministic hash function, it is necessary for the columns of the embedding matrix to be specified by a code: if two coordinates have small Hamming distance in their vectors of hash locations, it means they collide often. Since collision is the source of error, an adversary in this case could ask to embed a vector which has its mass equally spread on the two coordinates whose hash locations have small Hamming distance, causing large error with large probability over the choice of random signs. What our analysis shows is that not only is a good code necessary, but it is also sufficient.

In our second analysis approach, we expand Z^ℓ to obtain a polynomial with roughly $d^{2\ell}$ terms. We view its monomials as being in correspondence with graphs, group monomials whose graphs are isomorphic, then do some combinatorics to make the expectation calculation feasible. In this approach, we assume that the random signs as well as the hashing to coordinates in the target vector are done $2\log(1/\delta)$-wise independently. This graph-based approach played a large role in the analysis in our previous work [20] (which this work subsumes), and was later also used in [6].

We point out here that Figure 1(c) is somewhat simpler to implement, since there are simple constructions of $2\log(1/\delta)$-wise hash families [7]. Figure 1(b) on the other hand requires hashing without replacement, which amounts to using random permutations and can be derandomized using almost $2\log(1/\delta)$-wise independent permutation families [22].

In this conference version we provide the full details of the analyses for Figure 1(c). The analyses for Figure 1(b) are similar and are in the full version. The sparsity bounds we obtain for the code-based and random constructions in Figure 1(b) are the same up to constant factors as what we achieve for Figure 1(c) in Section 3 and Section 4, respectively.

2 Conventions and Notation

Definition 2.1. *For $A \in \mathbb{R}^{n\times n}$, the* Frobenius norm *of A is $\|A\|_F = \sqrt{\sum_{i,j} A_{i,j}^2}$.*

Definition 2.2. *For $A \in \mathbb{R}^{n\times n}$, the* operator norm *of A is $\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2$. In the case A is symmetric, this is also the largest magnitude of an eigenvalue of A.*

Henceforth, all logarithms are base-2 unless explicitly stated otherwise. For a positive integer n we use $[n]$ to denote the set $\{1, \ldots, n\}$. We will always be focused on embedding a vector $x \in \mathbb{R}^d$ into $\mathbb{R}^k$, and we assume $\|x\|_2 = 1$ without loss of generality (since our embeddings are linear). All vectors v are assumed to be column vectors, and v^T denotes its transpose. We often implicitly assume that various quantities are powers of 2 or 4, which is without loss of generality. Space complexity bounds (as in Section 5), are always measured in bits.

Definition 2.3. *The* Hamming distance $\Delta(u, v)$ *of two vectors u, v is $|\{i : u_i \neq v_i\}|$. An $(n, k, d)_q$* code *is a set of q^k vectors in $[q]^n$ with all pairwise Hamming distances at least d.*

3 Code-Based Construction

We provide an analysis of our construction Figure 1(c) when the hash locations are determined by some fixed error-correcting code. The full version gives the analysis for Figure 1(b) as well.

Define $k = C \cdot \varepsilon^{-2} \log(1/\delta)$ for a sufficiently large constant C. Let s be some integer dividing k satisfying $s \geq 2\varepsilon^{-1}\log(1/\delta)$. Let $\mathcal{C} = \{C_1, \ldots, C_d\}$ be any $(s, \log_{k/s} d, s - O(s^2/k))_{k/s}$ code. We specify our JL family by describing the embedded vector y. Define hash functions $\sigma : [d] \times [s] \to \{-1, 1\}$ and $h : [d] \times [s] \to [k/s]$. The former is drawn at random from a $2\log(1/\delta)$-wise independent family, and the latter has $h(i, j)$ being the jth entry of the ith codeword in $\mathcal{C}$. We conceptually view $y \in \mathbb{R}^k$

as being in $\mathbb{R}^{s\times(k/s)}$. Our embedded vector then has $y_{r,j} = \sum_{h(i,r)=j} \sigma(i,r)x_i/\sqrt{s}$. This describes our JL family, which is indexed by σ. Note the sparsity is s.

Remark 3.1. It is important to know whether an $(s, \log_{k/s} d, s - O(s^2/k))_{k/s}$ code exists. By picking h at random from an $O(\log(d/\delta))$-wise independent family and setting $s \geq \Omega(\varepsilon^{-1}\sqrt{\log(d/\delta)\log(1/\delta)})$, it is not too hard to show via the Chernoff bound (or more accurately, Markov's bound applied with the $O(\log(d/\delta))$th moment bound implied by integrating the Chernoff bound) followed by a union bound over all pairs of $\binom{d}{2}$ vectors that h defines a good code with probability $1-\delta$. We do not perform this analysis here since Section 4 obtains better parameters. We also point out that we may assume without loss of generality that $d = O(\varepsilon^{-2}/\delta)$. This is because there exists an embedding into this dimension with sparsity 1 using only 4-wise independence with distortion $(1+\varepsilon)$ and success probability $1-\delta$ [8, 28]. It is worth noting that for the construction in this section, potentially h could be deterministic given an explicit code with our desired parameters.

Analysis of code-based Figure 1(c):

We first note that

$$\|y\|_2^2 = \|x\|_2^2 + \frac{1}{s}\sum_{i\neq j}\sum_{r=1}^{s} \eta_{i,j,r}x_ix_j\sigma(i,r)\sigma(j,r),$$

where $\eta_{i,j,r}$ is 1 if $h(i,r) = h(j,r)$, and $\eta_{i,j,r} = 0$ otherwise. We thus would like that

$$Z = \frac{1}{s}\sum_{i\neq j}\sum_{r=1}^{s} \eta_{i,j,r}x_ix_j\sigma(i,r)\sigma(j,r)$$

is concentrated about 0. Note Z is a quadratic form in σ which can be written as $\sigma^T T\sigma$ for an $sd \times sd$ block-diagonal matrix T. There are s blocks, each $d \times d$, where in the rth block T_r we have $(T_r)_{i,j} = x_ix_j\eta_{i,j,r}/s$ for $i \neq j$ and $(T_r)_{i,i} = 0$ for all i. Now, $\mathbf{Pr}[|Z| > \varepsilon] = \mathbf{Pr}[|\sigma^T T\sigma| > \varepsilon]$. To bound this probability, we use the Hanson-Wright inequality [13] combined with a Markov bound.

Theorem 3.1. (Hanson-Wright inequality) *Let $z = (z_1, \ldots, z_n)$ be a vector of i.i.d. ± 1 random variables. There is a universal constant $C > 0$ so that for any symmetric $B \in \mathbb{R}^{n\times n}$ and $\ell \geq 2$,*

$$\mathbf{E}\left[\left|z^T Bz - \operatorname{trace}(B)\right|^\ell\right] \leq C^\ell \cdot \max\left\{\sqrt{\ell}\cdot\|B\|_F, \ell\cdot\|B\|_2\right\}^\ell$$

We prove our construction satisfies the JL lemma by applying Theorem 3.1 with $z = \sigma, B = T$.

Lemma 3.1. $\|T\|_F^2 = O(1/k)$.

Proof.

$$\begin{aligned}\|T\|_F^2 &= \frac{1}{s^2}\cdot\sum_{i\neq j} x_i^2x_j^2\cdot\left(\sum_{r=1}^{s}\eta_{i,j,r}\right)\\ &= \frac{1}{s^2}\cdot\sum_{i\neq j} x_i^2x_j^2\cdot(s - \Delta(C_i, C_j))\\ &\leq O(1/k)\cdot\|x\|_2^4\\ &= O(1/k)\end{aligned}$$

Lemma 3.2. $\|T\|_2 \leq 1/s$.

Proof. Since T is block-diagonal, its eigenvalues are the eigenvalues of each block. For a block T_r, write $T_r = (1/s)\cdot(S_r - D)$. D is diagonal with $D_{i,i} = x_i^2$, and $(S_r)_{i,j} = x_ix_j\eta_{i,j,r}$, including when $i = j$. Since S_r and D are both positive semidefinite, we have $\|T\|_2 \leq (1/s)\cdot\max\{\|S_r\|_2, \|D\|_2\}$. We have $\|D\|_2 = \|x\|_\infty^2 \leq 1$. For S_r, define u_t for $t \in [k/s]$ by $(u_t)_i = x_i$ if $h(i,r) = t$, and $(u_t)_i = 0$ otherwise. Then $u_1, \ldots, u_{k/s}$ are eigenvectors of S_r each with eigenvalue $\|u_t\|_2^2$, and furthermore they span the image of S_r. Thus $\|S_r\|_2 = \max_t \|u_t\|_2^2 \leq \|x\|_2^2 = 1$.

Theorem 3.2. $\mathbf{Pr}_\sigma[|\|y\|_2^2 - 1| > \varepsilon] < \delta$.

Proof. By a Markov bound applied to Z^ℓ for ℓ an even integer,

$$\mathbf{Pr}_\sigma[|Z| > \varepsilon] < \varepsilon^{-\ell}\cdot\mathbf{E}_\sigma[Z^\ell].$$

Since $Z = \sigma^T T\sigma$, trace$(T) = 0$, applying Theorem 3.1 with $B = T$, $z = \sigma$, $\ell \leq \log(1/\delta)$ gives

(3.1)
$$\mathbf{Pr}_\sigma[|Z| > \varepsilon] < C^\ell\cdot\max\left\{O(\varepsilon^{-1})\cdot\sqrt{\frac{\ell}{k}}, \varepsilon^{-1}\frac{\ell}{s}\right\}^\ell.$$

since the ℓth moment is determined by $2\log(1/\delta)$-wise independence of σ. We conclude the proof by noting that the expression in Eq. (3.1) is at most δ for $\ell = \log(1/\delta)$ and our choices for s, k.

Remark 3.2. Although our proof of Theorem 3.2 only requires $s = \Omega(\varepsilon^{-1}\log(1/\delta))$, by Remark 3.1 we need $s = \Omega(\varepsilon^{-1}\sqrt{\log(1/(1/\varepsilon\delta))\log(1/\delta)})$ to ensure that h defines a good code with high probability. Thus, this code-based construction falls short of our desired target sparsity of $O(\varepsilon^{-1}\log(1/\delta))$ by a $\sqrt{\log_{1/\delta}(1/\varepsilon)}$ factor when $\varepsilon < \delta$.

REMARK 3.3. Only using that $\mathcal{C}$ has sufficiently high minimum distance, it is impossible to improve our sparsity bound further. For example, for any $(s, \log_{k/s} d, s - O(s^2/k))_{k/s}$ code $\mathcal{C}$, create a new code $\mathcal{C}'$ which simply replaces the first letter of each codeword with "1"; $\mathcal{C}'$ then still has roughly the same minimum distance. However, in our construction this corresponds to all indices colliding in the first chunk of k/s coordinates, which creates an error term of $(1/s) \cdot \sum_{i \neq j} x_i x_j \sigma(i,r)\sigma(j,r)$. Now, suppose x consists of $t = (1/2) \cdot \log(1/\delta)$ entries each with value $1/\sqrt{t}$. Then, with probability $\sqrt{\delta} \gg \delta$, all these entries receive the same sign under σ and contribute a total error of $\Omega(t/s)$ in the first chunk alone. We thus need $t/s = O(\varepsilon)$, which implies $s = \Omega(\varepsilon^{-1}\log(1/\delta))$.

4 Random Hashing Construction

Here we show that if the hash function h described in Section 3 is not specified by a fixed code, but rather is chosen at random from some family of sufficiently high independence, then one can achieve sparsity $O(\varepsilon^{-1}\log(1/\delta))$ (in the case of Figure 1(b), we actually need almost k-wise independent *permutations*). Recall our bottleneck in reducing the sparsity in Section 3 was actually obtaining the code, discussed in Remark 3.1. Below we analyze the case of Figure 1(c). We also provide the analysis for Figure 1(b) in the full version.

We let the hash function $h : [d] \times [s] \to [k/s]$ be randomly selected from a $2\log(1/\delta)$-wise independent family. Note that one can sample a random matrix from this family using a $O(\log(1/\delta)\log d)$-length seed. We perform our analysis by bounding the ℓth moment of Z from first principles for $\ell = \log(1/\delta)$ an even integer (for this particular scheme, it seems the Hanson-Wright inequality does not simplify any details of the proof). We then use Markov's inequality to say $\mathbf{Pr}_{h,\sigma}[|Z| > \varepsilon] < \varepsilon^{-\ell} \cdot \mathbf{E}_{h,\sigma}[Z^\ell]$.

Let $Z_r = \sum_{i \neq j} \eta_{i,j,r} x_i x_j \sigma(i,r)\sigma(j,r)$ so that $Z = (1/s) \cdot \sum_{r=1}^{s} Z_r$. We first bound the tth moment of each Z_r for $1 \le t \le \ell$. As in the Frobenius norm moment bound of [20], and also used later in [6], the main idea is to construct a correspondence between the monomials appearing in Z_r^t and certain graphs Notice

$$Z_r^t = \sum_{i_1 \neq j_1, \ldots, i_t \neq j_t} \prod_{u=1}^{t} \eta_{i_u,j_u,r} x_{i_u} x_{j_u} \sigma(i_u,r)\sigma(j_u,r). \tag{4.2}$$

To each monomial above we associate a directed multigraph with labeled edges whose vertices correspond to the distinct i_u and j_u. An $x_{i_u}x_{j_u}$ term corresponds to a directed edge with label u from the vertex corresponding to i_u to the vertex corresponding to j_u. The basic idea we use to bound $E_{h,\sigma}[Z_r^t]$ is to group these monomials based on the isomorphism class of the associated graphs.

LEMMA 4.1. *For $t \le \log(1/\delta)$,*

$$\mathbf{E}_{h,\sigma}[Z_r^t] \le 2^{O(t)} \cdot \begin{cases} s/k & t < \log(k/s) \\ (t/\log(k/s))^t & \text{otherwise} \end{cases}.$$

Proof. Let $\mathcal{G}_t$ be the set of isomorphism classes of directed multigraphs with t labeled edges with distinct labels in $[t]$, where each vertex has positive and even degree (the sum of in- and out-degrees), and the number of vertices is between 2 and t. Let $\mathcal{G}_t'$ be similar, but with labeled vertices and connected components as well, where vertices have distinct labels between 1 and the number of vertices, and components have distinct labels between 1 and the number of components. Let f map the monomials appearing in Eq. (4.2) to the corresponding graph isomorphism class. By $2t$-wise independence of σ, any monomial in Eq. (4.2) whose corresponding graph does not have all even degrees has expectation 0. For a graph G, we let v denote the number of vertices, and m the number of connected components. Let d_u denote the degree of a vertex u. Then,

$$\begin{aligned}
\mathbf{E}_{h,\sigma}[Z_r^t] &= \sum_{\substack{i_1 \neq j_1 \\ \cdots \\ i_t \neq j_t}} \left(\prod_{u=1}^{t} x_{i_u}x_{j_u}\right) \mathbf{E}\left[\prod_{u=1}^{t} \sigma(i_u,r)\sigma(j_u,r)\right] \\
&\qquad \times \mathbf{E}\left[\prod_{u=1}^{t} \eta_{i_u,j_u,r}\right] \\
&= \sum_{G \in \mathcal{G}_t} \sum_{\substack{i_1 \neq j_1, \ldots, i_t \neq j_t \\ f((i_u,j_u)_{u=1}^t) = G}} \left(\prod_{u=1}^{t} x_{i_u}x_{j_u}\right) \\
&\qquad \times \mathbf{E}\left[\prod_{u=1}^{t} \eta_{i_u,j_u,r}\right] \\
&= \sum_{G \in \mathcal{G}_t} \sum_{\substack{i_1 \neq j_1, \ldots, i_t \neq j_t \\ f((i_u,j_u)_{u=1}^t) = G}} \left(\frac{s}{k}\right)^{v-m} \cdot \left(\prod_{u=1}^{t} x_{i_u}x_{j_u}\right) \qquad (4.3) \\
&\le \sum_{G \in \mathcal{G}_t} \left(\frac{s}{k}\right)^{v-m} \cdot v! \cdot \frac{1}{\binom{t}{d_1/2,\ldots,d_v/2}} \qquad (4.4) \\
&= \sum_{G \in \mathcal{G}_t'} \left(\frac{s}{k}\right)^{v-m} \cdot \frac{1}{m!} \cdot \frac{1}{\binom{t}{d_1/2,\ldots,d_v/2}}. \qquad (4.5)
\end{aligned}$$

We now justify these inequalities. The justification of Eq. (4.3) is similar to that in the Frobenius norm bound in [20]. That is, $\prod_{u=1}^{t} \eta_{i_u,j_u,r}$ is determined by $h(i_u,r), h(j_u,r)$ for each $u \in [t]$, and hence its expectation is determined by $2t$-wise independence of h. This product is 1 if i_u and j_u hash to the same element for each u and is 0 otherwise. Every i_u, j_u pair hashes to the same element if and only if for each connected component of G, all elements of $\{i_1, \ldots, i_t, j_1, \ldots, j_t\}$ corresponding to vertices in that component hash to the same value. We can choose one element of $[k/s]$ for each component to be hashed to, thus giving $(k/s)^m$ possibilities. The probability of any particular hashing is $(k/s)^{-v}$, and this gives that the expectation of the product is $(s/k)^{v-m}$.

For Eq. (4.4), note that $(\|x\|_2^2)^t = 1$, and the coefficient of $\prod_{u=1}^v x_{a_u}^{d_u}$ in its expansion for $\sum_u d_u = 2t$ is $\binom{t}{d_1/2,\ldots,d_v/2}$. Meanwhile, the coefficient of this monomial when summing over all $i_1 \neq j_1, \ldots, i_t \neq j_t$ for a particular $G \in \mathcal{G}_\ell$ is at most $v!$. For Eq. (4.5), we move from isomorphism classes in $\mathcal{G}_t$ to those in $\mathcal{G}'_t$. For any $G \in \mathcal{G}_t$, there are $v! \cdot m!$ ways to label vertices and connected components. Note that each of the labelings of vertices and connected components gives us a distinct graph isomorphism class because the edges are already labeled (and thus we have no remaining symmetries).

We now bound the sum of the $1/\binom{t}{d_1/2,\ldots,d_v/2}$ term. Fix $v_1, \ldots, v_m$, $t_1, \ldots, t_m$ (where there are v_i vertices and t_i edges in the ith component C_i), and the assignment of vertex and edge labels to connected components. We upper bound Eq. (4.5) by considering building G edge by edge, starting with 0 edges. We may end up building some graphs which are invalid in the process (e.g. some vertices might not have even degrees), but this is allowed since we are only providing an upper bound. Let the initial graph be G_0, and we form $G = G_t$ by adding edges in increasing label order. We then want to bound the sum of $1/\binom{t}{d_1/2,\ldots,d_v/2}$ over $G \in \mathcal{G}'_\ell$ which satisfy the quantities we have fixed. Note $1/\binom{t}{d_1/2,\ldots,d_v/2}$ equals $2^{O(t)} \cdot t^{-t} \cdot \prod_{u=1}^v \cdot \left(\sqrt{d_u}^{d_u}\right)$. Initially, when $t = 0$, our sum is $S_0 = 1$. When considering all ways to add the next edge to G_u to form G_{u+1}, an edge $i \to j$ contributes a factor of $S_u \cdot \sqrt{d_i d_j}/t$ to S_{u+1}. Since we fixed assignments of edge labels to connected components, this edge must come from some particular component C_w. Summing over vertices $i \neq j$ in C_w and applying Cauchy-Schwarz, $\sum_{i \neq j \in C_w} \sqrt{d_i d_j}/t \leq \frac{1}{t} \cdot \left(\sum_{i \in C_w} \sqrt{d_i}\right)^2 \leq (v_i t_i)/t$. Since there are $\binom{v}{v_1,\ldots,v_m}\binom{t}{t_1,\ldots,t_m}$ ways to assign edge and vertex labels to components, Eq. (4.5) gives

$$\mathbf{E}_{h,\sigma}[Z_r^t] \leq 2^{O(t)} \cdot \sum_{v=2}^{t} \sum_{m=1}^{v/2} \sum_{v_1,\ldots,v_m} \sum_{t_1,\ldots,t_m} \left(\frac{s}{k}\right)^{v-m} \cdot \frac{1}{m^m}$$

$$\times \cdot \binom{v}{v_1,\ldots,v_m} \cdot \binom{t}{t_1,\ldots,t_m} \cdot \frac{\left(\prod_{i=1}^m (v_i t_i)^{t_i}\right)}{t^t} \tag{4.6}$$

$$\leq 2^{O(t)} \cdot \sum_{v=2}^{t} \sum_{m=1}^{v/2} \left(\frac{s}{k}\right)^{v-m} \cdot \left(\frac{v^v}{m^m}\right) \cdot v^{t-v} \tag{4.7}$$

$$\leq 2^{O(t)} \cdot \sum_{v=2}^{t} \sum_{m=1}^{v/2} \left(\frac{s}{k}\right)^{v-m} \cdot (v-m)^t \tag{4.8}$$

$$\leq 2^{O(t)} \cdot \sum_{v=2}^{t} \sum_{q=1}^{v/2} \left(\frac{s}{k}\right)^{q} \cdot q^t.$$

Eq. (4.8) holds since there are at most 2^{v+t} ways to choose the v_i, t_i and $t_i \geq v_i$, and also $v \geq 2m$ and thus $v = O(v-m)$.Setting $q = v - m$ and under the constraint $q \geq 1$, $(s/k)^q \cdot q^t$ is maximized when $q = \max\{1, \Theta(t/\log(k/s))\}$. The lemma follows.

Theorem 4.1. *Our construction in this section gives a JL family with sparsity $s = O(\varepsilon^{-1} \cdot \log(1/\delta))$.*

Proof. We have

$$\mathbf{E}_{h,\sigma}[Z^\ell] = \frac{1}{s^\ell} \cdot \sum_{\substack{r_1 < \ldots < r_q \\ \ell_1,\ldots,\ell_q \\ \forall i\ \ell_i > 1 \\ \sum_i \ell_i = \ell}} \binom{\ell}{\ell_1,\ldots,\ell_q} \cdot \prod_{i=1}^q \mathbf{E}_{h,\sigma}[Z_{r_i}^{\ell_i}]$$

$$\leq \frac{1}{s^\ell} \cdot 2^{O(\ell)} \cdot \sum_{q=1}^{\ell/2} \binom{s}{q} \cdot \frac{\ell^\ell}{\prod_{i=1}^q \ell_i^{\ell_i}} \cdot \left(\frac{s}{k}\right)^q$$

$$\times \prod_{i=1}^q \left\lceil \frac{\ell_i}{\log(k/s)} \right\rceil^{\ell_i} \tag{4.9}$$

$$\leq \frac{1}{s^\ell} \cdot 2^{O(\ell)} \cdot \sum_{q=1}^{\ell/2} \binom{s}{q} \cdot \ell^\ell \cdot \left(\frac{s}{k}\right)^q$$

$$\leq \frac{1}{s^\ell} \cdot 2^{O(\ell)} \cdot \sum_{q=1}^{\ell/2} \ell^\ell \cdot \left(\frac{s^2}{qk}\right)^q$$

Eq. (4.9) follows since there are $\binom{s}{q}$ ways to choose the r_i, and there are at most $2^{\ell-1}$ ways to choose the ℓ_i. Furthermore, even for the $\ell_i > \log(k/s)$, we have $2^{O(\ell_i)} \cdot (\ell_i/\log(k/s))^{\ell_i} = 2^{O(\ell_i)} \cdot (s/k)^2 \cdot (\ell_i/\log(k/s))^{\ell_i}$, so that the $(s/k)^q$ term is valid. Taking derivatives shows that the above is maximized for $q = s^2/(ek) < \ell/2$, which gives a summand of $2^{O(\ell)} \cdot \ell^\ell$. Thus, we have that the above moment is at most $(\varepsilon/2)^\ell$ when $k = C'\ell/\varepsilon^2$ for sufficiently large C'. The claim then follows since $\mathbf{Pr}_{h,\sigma}[|Z| > \varepsilon] < \varepsilon^{-\ell} \cdot \mathbf{E}_{h,\sigma}[Z^\ell]$ by Markov's inequality, and we set $\ell = \log(1/\delta)$.

Remark 4.1. It is worth noting that if one wants distortion $1 \pm \varepsilon_i$ with probability $1-\delta_i$ simultaneously for all i in some set S, our proof of Theorem 4.1 reveals that it suffices to set $s = C \cdot \sup_{i \in S} \varepsilon_i^{-1} \log(1/\delta_i)$ and $k = C \cdot \sup_{i \in S} \varepsilon_i^{-2} \log(1/\delta_i)$.

5 Faster numerical linear algebra streaming algorithms

The works of [9, 27] gave algorithms to solve various approximate numerical linear algebra problems given small memory and a only one or few passes over an input matrix. They considered models where one only sees a row or column at a time of some matrix $A \in \mathbb{R}^{d \times n}$. Another update model considered was the turnstile streaming model. In this model, the matrix A starts off as 0. One then sees a sequence of m updates $(i_1, j_1, v_1), \ldots, (i_m, j_m, v_m)$, where each update (i, j, v) triggers the change $A_{i,j} \leftarrow A_{i,j} + v$. The goal in all these models is to compute some functions of A at the end of seeing all rows, columns, or turnstile updates. The algorithm should use little memory (much less than what is required to store A explicitly). Both works [9, 27] solved problems such as approximate linear regression and best rank-k approximation by reducing to the problem of sketches for approximate matrix products. Before delving further, first we give a definition.

Definition 5.1. *Distribution $\mathcal{D}$ over $\mathbb{R}^{k \times d}$ has* (ε, δ)-JL moments *if for $\ell = \log(1/\delta)$ and for all x with $\|x\|_2 = 1$,*

$$\mathbf{E}_{S \sim \mathcal{D}} \left[\left| \|Sx\|_2^2 - 1 \right|^{\ell} \right] \le (\varepsilon/2)^{\ell}.$$

Now, the following theorem is a generalization of [9, Theorem 2.1]. The theorem states that any distribution with JL moments also provides a sketch for approximate matrix products. A similar statement was made in [27, Lemma 6], but that statement was slightly weaker in its parameters because it resorted to a union bound, which we avoid by using Minkowski's inequality. Our proof can be found in the attached full version.

Theorem 5.1. *Given $0 < \varepsilon, \delta < 1/2$, let $\mathcal{D}$ be any distribution over matrices with d columns with the (ε, δ)-JL moment property. Then for A, B any real matrices with d rows and $\|A\|_F = \|B\|_F = 1$,*

$$\mathbf{Pr}_{S \sim \mathcal{D}} \left[\|A^T S^T S B - A^T B\|_F > 3\varepsilon/2 \right] < \delta.$$

Remark 5.1. Often when one constructs a JL distribution $\mathcal{D}$ over $k \times d$ matrices, it is shown that for all x with $\|x\|_2 = 1$ and for all $\varepsilon > 1/\sqrt{k}$,

$$\mathbf{Pr}_{S \sim \mathcal{D}} \left[\left| \|Sx\|_2^2 - 1 \right| > \varepsilon \right] < e^{-\Theta(\varepsilon^2 k)}.$$

Any such distribution automatically satisfies the $(\varepsilon, e^{-\Theta(\varepsilon^2 k)})$-JL moment property for any $\varepsilon > 1/\sqrt{k}$ by converting the tail bound into a moment bound via integration by parts.

Now we arrive at the main point of this section. Several algorithms for approximate linear regression and best rank-k approximation in [9] simply maintain SA as A is updated, where S comes from the JL distribution with $\Omega(\log(1/\delta))$-wise independent $\pm 1/\sqrt{k}$ entries. In fact though, their analyses of their algorithms only use the fact that this distribution satisfies the approximate matrix product sketch guarantees of Theorem 5.1. Due to Theorem 5.1 though, we know that *any* distribution satisfying the (ε, δ)-JL moment condition gives an approximate matrix product sketch. Thus, random Bernoulli matrices may be replaced with our sparse JL distributions in this work. We now state some of the algorithmic results given in [9] and describe how our constructions provide improvements in the update time (the time to process new columns, rows, or turnstile updates).

As in [9], when stating our results we will ignore the space and time complexities of storing and evaluating the hash functions in our JL distributions. We discuss this issue later in Remark 5.2.

5.1 Linear regression

In this problem we have an $A \in \mathbb{R}^{d \times n}$ and $b \in \mathbb{R}^d$. We would like to compute a vector $\tilde{x}$ such that $\|A\tilde{x} - b\|_F \le (1+\varepsilon) \cdot \min_{x^*} \|Ax^* - b\|_F$ with probability $1 - \delta$. In [9], it is assumed that the entries of A, b require $O(\log(nd))$ bits of precision to store precisely. Both A, b receive turnstile updates.

Theorem 3.2 of [9] proves that such an $\tilde{x}$ can be computed with probability $1 - \delta$ from SA and Sb, where S is drawn from a distribution that simultaneously satisfies both the $(1/2, \eta^{-r}\delta)$ and $(\sqrt{\varepsilon/r}, \delta)$-JL moment properties for some fixed constant $\eta > 1$, and where $\mathrm{rank}(A) \le r \le n$. Thus due to Remark 4.1, we have the following.

Theorem 5.2. *There is a one-pass streaming algorithm for linear regression in the turnstile model where one maintains a sketch of size $O(n^2 \varepsilon^{-1} \log(1/\delta) \log(nd))$. Processing each update requires $O(n + \sqrt{n/\varepsilon} \cdot \log(1/\delta))$ arithmetic operations and hash function evaluations.*

Theorem 5.2 improves the update complexity of [9], which was $O(n\varepsilon^{-1} \log(1/\delta))$.

5.2 Low rank approximation

In this problem, we have an $A \in \mathbb{R}^{d \times n}$ of rank ρ with entries that require precision $O(\log(nd))$ to store. We would like to compute the best rank-r approximation A_r to A. We define $\Delta_r \stackrel{\text{def}}{=} \|A - A_r\|_F$ as the error of A_r. We relax the problem by only requiring that we compute a matrix A'_r such that $\|A - A'_r\|_F \leq (1 + \varepsilon)\Delta_r$ with probability $1 - \delta$ over the randomness of the algorithm.

Two-pass algorithm: Theorem 4.4 of [9] gives a 2-pass algorithm where in the first pass, one maintains SA where S is drawn from a distribution that simultaneously satisfies both the $(1/2, \eta^{-r}\delta)$ and $(\sqrt{\varepsilon/r}, \delta)$-JL moment properties. It is also assumed that $\rho \geq 2r + 1$. The first pass is thus sped up again as in Theorem 5.2.

One-pass algorithm for column/row-wise updates: Theorem 4.5 of [9] gives a one-pass algorithm in the case that A is seen either one whole column or row at a time. The algorithm maintains both SA and SAA^T where S is drawn from a distribution that simultaneously satisfies both the $(1/2, \eta^{-r}\delta)$ and $(\sqrt{\varepsilon/r}, \delta)$-JL moment properties. This implies the following.

Theorem 5.3. *There is a one-pass streaming algorithm for approximate low rank approximation with row/column-wise updates where one maintains a sketch of size $O(r\varepsilon^{-1}(n + d)\log(1/\delta)\log(nd))$. Processing each update requires $O(r + \sqrt{r/\varepsilon} \cdot \log(1/\delta))$ amortized arithmetic operations and hash function evaluations per entry of A.*

Theorem 5.3 improves the amortized update complexity of [9], which was $O(r\varepsilon^{-1}\log(1/\delta))$.

Our construction also improves the complexity for low rank approximation in three passes with row-wise updates, in one pass in the turnstile model with a bi-criteria guarantee, and in one pass in the turnstile model. Details are in the attached full version.

Remark 5.2. In the algorithms above, we counted the number of hash function evaluations that must be performed. We use our construction in Figure 1(c), which uses $2\log(1/\delta)$-wise independent hash functions. Standard constructions of t-wise independent hash functions over universes with elements fitting in a machine word require $O(t)$ time to evaluate [7]. In our case, this would blow up our update time by factors such as n or r, which could be large. Instead, we use fast multipoint evaluation of polynomials. The standard construction [7] of our desired hash functions mapping some domain $[z]$ onto itself for z a power of 2 takes a degree-$(t - 1)$ polynomial p with random coefficients in $\mathbb{F}_z$. The hash function evaluation at some point y is then the evaluation $p(y)$ over $\mathbb{F}_z$. Theorem 5.4 below states that p can be evaluated at t points in total time $\tilde{O}(t)$. We note that in the theorems above, we are always required to evaluate some t-wise independent hash function on many more than t points per stream update. Thus, we can group these evaluation points into groups of size t then perform fast multipoint evaluation for each group. We borrow this idea from [21], which used it to give a fast algorithm for moment estimation in data streams.

Theorem 5.4. ([30, Ch. 10]) *Let $\mathbf{R}$ be a ring, and let $q \in \mathbf{R}[x]$ be a degree-t polynomial. Then, given distinct $x_1, \ldots, x_t \in \mathbf{R}$, all the values $q(x_1), \ldots, q(x_t)$ can be computed using $O(t\log^2 t \log\log t)$ operations over $\mathbf{R}$.*

Acknowledgments

This work was done while the first author was a graduate student at Harvard University supported by an NSF Graduate Research Fellowship. and the second author was a graduate student at MIT supported by a Xerox-MIT Fellowship, and in part by the Center for Massive Data Algorithmics (MADALGO) - a center of the Danish National Research Foundation.

We thank Venkat Chandar, Venkatesan Guruswami, Swastik Kopparty, and Madhu Sudan for useful discussions about error-correcting codes. We thank David Woodruff for answering several questions about [9]. We thank Piotr Indyk and Eric Price for useful comments and discussion, and the anonymous reviewers for useful comments.

References

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

[2] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the 38th ACM Symposium on Theory of Computing (STOC)*, pages 557–563, 2006.

[3] Nir Ailon and Edo Liberty. Fast dimension reduction using Rademacher series on dual BCH codes. *Discrete Comput. Geom.*, 42(4):615–630, 2009.

[4] Nir Ailon and Edo Liberty. Almost optimal unrestricted fast Johnson-Lindenstrauss transform. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 185–191, 2011.

[5] Rosa I. Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 63(2):161–182, 2006.

[6] Vladimir Braverman, Rafail Ostrovsky, and Yuval Rabani. Rademacher chaos, random Eulerian graphs and the sparse Johnson-Lindenstrauss transform. *CoRR*, abs/1011.2590, 2010.

[7] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[8] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[9] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC)*, pages 205–214, 2009.

[10] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse Johnson-Lindenstrauss transform. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 341–350, 2010.

[11] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003.

[12] Peter Frankl and Hiroshi Maehara. The Johnson-Lindenstrauss lemma and the sphericity of some graphs. *J. Comb. Theory. Ser. B*, 44(3):355–362, 1988.

[13] David Lee Hanson and Farroll Tim Wright. A bound on tail probabilities for quadratic forms in independent random variables. *Ann. Math. Statist.*, 42(3):1079–1083, 1971.

[14] Aicke Hinrichs and Jan Vybíral. Johnson-Lindenstrauss lemma for circulant matrices. *arXiv*, abs/1001.4919, 2010.

[15] Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 10–33, 2001.

[16] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th ACM Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

[17] T. S. Jayram and David P. Woodruff. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with low error. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–10, 2011.

[18] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[19] Daniel M. Kane, Raghu Meka, and Jelani Nelson. Almost optimal explicit Johnson-Lindenstrauss transformations. In *Proceedings of the 15th International Workshop on Randomization and Computation (RANDOM), to appear*, 2011.

[20] Daniel M. Kane and Jelani Nelson. A derandomized sparse Johnson-Lindenstrauss transform. *CoRR*, abs/1006.3585, 2010.

[21] Daniel M. Kane, Jelani Nelson, Ely Porat, and David P. Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 745–754, 2011.

[22] Eyal Kaplan, Moni Naor, and Omer Reingold. Derandomized constructions of k-wise (almost) independent permutations. *Algorithmica*, 55(1):113–133, 2009.

[23] Felix Krahmer and Rachel Ward. New and improved Johnson-Lindenstrauss embeddings via the Restricted Isometry Property. *SIAM J. Math. Anal.*, 43(3):1269–1281, 2011.

[24] Jirí Matousek. On variants of the Johnson-Lindenstrauss lemma. *Random Struct. Algorithms*, 33(2):142–156, 2008.

[25] Rajeev Motwani and Prabakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[26] Vipin Kumar Pang-Ning Tan, Michael Steinbach. *Introduction to Data Mining.* Addison-Wesley, 2005.

[27] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 143–152, 2006.

[28] Mikkel Thorup and Yin Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 615–624, 2004.

[29] Santosh Vempala. *The random projection method*, volume 65 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, 2004.

[30] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, 1999.

[31] Jan Vybíral. A variant of the Johnson-Lindenstrauss lemma for circulant matrices. *arXiv*, abs/1002.2847, 2010.

[32] Kilian Q. Weinberger, Anirban Dasgupta, John Langford, Alexander J. Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 1113–1120, 2009.

A On the sparsity required in various schemes

In this section we show that sparsity $\Omega(\varepsilon^{-1}\log(1/\delta))$ is required in Figure 1(b) and Figure 1(c), even if the hash functions used are completely random. We also show that sparsity $\tilde{\Omega}(\varepsilon^{-1}\log^2(1/\delta))$ is required in the DKS construction (Figure 1(a)), nearly matching the upper bounds of [6, 20]. Interestingly, all three of our proofs of (near-)tightness of analyses for these three constructions use the same hard input vectors. In particular, if $s = o(1/\varepsilon)$, then we show that a vector with $t = \lfloor 1/(s\varepsilon)\rfloor$ entries each of value $1/\sqrt{t}$ incurs large distortion with large probability. If $s = \Omega(1/\varepsilon)$ but is still not sufficiently large, we show that the vector $(1/\sqrt{2}, 1/\sqrt{2}, 0, \ldots, 0)$ incurs large distortion with large probability (in fact, for the DKS scheme one can even take the vector $(1, 0, \ldots, 0)$).

A.1 Near-tightness for DKS Construction

The main theorem of this section is the following.

Theorem A.1. *The DKS construction of [10] requires sparsity $s = \Omega(\varepsilon^{-1}\cdot\lceil\log^2(1/\delta)/\log^2(1/\varepsilon)\rceil)$ to achieve distortion $1\pm\varepsilon$ with success probability $1-\delta$.*

Before proving Theorem A.1, we recall the DKS construction (Figure 1(a)). First, we replicate each coordinate s times while preserving the ℓ_2 norm. That is, we produce the vector $\tilde{x} = (x_1, \ldots, x_1, x_2, \ldots, x_2, \ldots, x_d, \ldots, x_d)/\sqrt{s}$, where each x_i is replicated s times. Then, pick a random $k \times ds$ embedding matrix A for $k = C\varepsilon^{-2}\log(1/\delta)$ where each column has exactly one non-zero entry, in a location defined by some random function $h : [ds] \to [k]$, and where this non-zero entry is ± 1, determined by some random function $\sigma : [ds] \to \{-1, 1\}$. The value $C > 0$ is some fixed constant. The final embedding is A applied to $\tilde{x}$. We are now ready to prove Theorem A.1. The proof is similar to that of Theorem A.3.

Our proof will use the following standard fact.

Fact A.1. ([25, Proposition B.3]) *For all $t, n \in \mathbb{R}$ with $n \geq 1$ and $|t| \leq n$,*

$$e^t(1 - t^2/n) \leq (1 + t/n)^n \leq e^t.$$

We now prove Theorem A.1.

Proof. First suppose $s \leq 1/(2\varepsilon)$. Consider a vector with $t = \lfloor 1/(s\varepsilon)\rfloor$ non-zero coordinates each of value $1/\sqrt{t}$. If there is exactly one pair $\{i, j\}$ that collides under h, and furthermore the signs agree under σ, the ℓ_2 norm squared of our embedded vector will be $(st-2)/(st) + 4/(st)$. Since $1/(st) \geq \varepsilon$, this quantity is at least $1 + 2\varepsilon$. The event of exactly one pair $\{i, j\}$ colliding occurs with probability

$$\binom{st}{2}\cdot\frac{1}{k}\cdot\prod_{i=0}^{st-2}(1 - i/k) \geq \Omega\left(\frac{1}{\log(1/\delta)}\right)\cdot(1-\varepsilon/2)^{1/\varepsilon}$$
$$= \Omega(1/\log(1/\delta)),$$

which is much larger than $\delta/2$ for δ smaller than some constant. Now, given a collision, the colliding items have the same sign with probability $1/2$.

We next consider the case $1/(2\varepsilon) < s \leq 4/\varepsilon$. Consider the vector $x = (1, 0, \ldots, 0)$. If there are exactly three pairs $\{i_1, j_1\}, \ldots, \{i_3, j_3\}$ that collide under h in three distinct target coodinates, and furthermore the signs agree under σ, the ℓ_2 norm squared of our embedded vector will be $(s-6)/(s) + 12/(s) > 1 + 3\varepsilon/2$. The event of three pairs colliding occurs with probability

$$\binom{s}{2}\binom{s-2}{2}\binom{s-4}{2}\cdot\frac{1}{3!}\cdot\frac{1}{k^3}\cdot\prod_{i=0}^{s-4}(1 - i/k)$$
$$\geq \Omega\left(\frac{1}{\log^3(1/\delta)}\right)\cdot(1-\varepsilon/8)^{4/\varepsilon}$$
$$= \Omega(1/\log^3(1/\delta)),$$

which is much larger than $\delta/2$ for δ smaller than some constant. Now, given a collision, the colliding items have the same sign with probability $1/8$.

We lastly consider the case $4/\varepsilon < s \leq 2c\varepsilon^{-1}\log^2(1/\delta)/\log^2(1/\varepsilon)$ for some constant $c > 0$ (depending on C) to be determined later. First note this case only exists when $\delta = O(\varepsilon)$. Define $x = (1, 0, \ldots, 0)$. Suppose there exists an integer q so that

1. $q^2/s \geq 4\varepsilon$
2. $q/s < \varepsilon$
3. $(s/(qk))^q(1 - 1/k)^s > \delta^{1/3}$.

First we show it is possible to satisfy the above conditions simultaneously for our range of s. We set $q = 2\sqrt{\varepsilon s}$, satisfying item 1 trivially, and item 2 since $s > 4/\varepsilon$. For item 3, Fact A.1 gives

$$(s/(qk))^q\cdot(1 - 1/k)^s \geq \left(\frac{s}{qk}\right)^q\cdot e^{-s/k}\cdot\left(1 - \frac{s}{k^2}\right).$$

The $e^{-s/k}\cdot(1 - (s/k^2))$ term is at least $\delta^{1/6}$ by the settings of s, k, and the $(s/(qk))^q$ term is also at least $\delta^{1/6}$ for c sufficiently small.

Now, consider the event $\mathcal{E}$ that exactly q of the s copies of x_1 are hashed to 1 by h, and to $+1$

by σ. If $\mathcal{E}$ occurs, then coordinate 1 in the target vector contributes $q^2/s \geq 4\varepsilon$ to ℓ_2^2 in the target vector by item 1 above, whereas these coordinates only contribute $q/s < \varepsilon$ to $\|x\|_2^2$ by item 2 above, thus causing error at least 3ε. Furthermore, the $s-q$ coordinates which do not hash to 1 are being hashed to a vector of length $k-1 = \omega(1/\varepsilon^2)$ with random signs, and thus these coordinates have their ℓ_2^2 contribution preserved up to $1 \pm o(\varepsilon)$ with constant probability by Chebyshev's inequality. It thus just remains to show that $\mathbf{Pr}[\mathcal{E}] \gg \delta$. We have

$$\begin{aligned}\mathbf{Pr}[\mathcal{E}] &= \binom{s}{q} \cdot k^{-q} \cdot \left(1 - \frac{1}{k}\right)^{s-q} \cdot 1/2^q \\ &\geq \left(\frac{s}{qk}\right)^q \cdot \left(1 - \frac{1}{k}\right)^s \cdot \frac{1}{2^q} \\ &> \delta^{1/3} \cdot \frac{1}{2^q}.\end{aligned}$$

The 2^{-q} term is $\omega(\delta^{1/3})$, so $\mathbf{Pr}[\mathcal{E}] = \omega(\delta^{2/3}) \gg \delta$.

Remark A.1. Sparsity $\Omega(\varepsilon^{-1}\log(1/\delta))$ is also a lower bound on the sparsity required in the DKS scheme. This is because the hard inputs that we consider in the next section are also hard for the DKS construction, due to the same analysis that we give in the proof of Theorem A.2.

A.2 Tightness of Figure 1(b) analysis

Theorem A.2. *For δ smaller than a constant depending on C for $k = C\varepsilon^{-2}\log(1/\delta)$, the scheme of Figure 1(b) with a random hash function h without replacement requires $s = \Omega(\varepsilon^{-1}\log(1/\delta))$ to obtain distortion $1 \pm \varepsilon$ with probability $1 - \delta$.*

Proof. First suppose $s \leq 1/(2\varepsilon)$. We consider a vector with $t = \lfloor 1/(s\varepsilon) \rfloor$ non-zero coordinates each of value $1/\sqrt{t}$. If there is exactly one set i, j, r with $i \neq j$ such that $S_{r,i}, S_{r,j}$ are both non-zero for the embedding matrix S (i.e., there is exactly one collision), then the total error is $2/(ts) \geq 2\varepsilon$. It just remains to show that this happens with probability larger than δ. The probability of this occurring is

$$\begin{aligned}&s^2 \cdot \binom{t}{2} \cdot \frac{1}{k} \cdot \frac{k-s}{k-1} \cdots \frac{k-2s+2}{k-s+1} \cdot \left(\frac{(k-2s+1)!}{(k-ts+1)!}\right) \\ &\times \left(\frac{(k-s)!}{k!}\right)^{t-2} \geq \frac{s^2t^2}{2k} \cdot \left(\frac{k-st}{k}\right)^{st} \\ &\qquad \geq \frac{s^2t^2}{2k} \cdot \left(1 - \frac{s^2t^2}{k}\right) \\ &\qquad = \Omega(1/\log(1/\delta)).\end{aligned}$$

Now consider the case $1/(2\varepsilon) < s < c \cdot \varepsilon^{-1}\log(1/\delta)$ for some small constant c. Consider the vector $(1/\sqrt{2}, 1/\sqrt{2}, 0, \ldots, 0)$. Suppose there are exactly $2s\varepsilon$ collisions, i.e. $2s\varepsilon$ distinct values of r such that $S_{r,i}, S_{j,r}$ are both non-zero (to avoid tedium we disregard floors and ceilings and just assume $s\varepsilon$ is an integer). Also, suppose that in each colliding row r we have $\sigma(1,r) = \sigma(2,r)$. Then, the total error would be 2ε. It just remains to show that this happens with probability larger than δ. The probability of signs agreeing in exactly $2\varepsilon s$ chunks is $2^{-2\varepsilon s} > 2^{-2c\log(1/\delta)}$, which is larger than $\sqrt{\delta}$ for $c < 1/4$. The probability of exactly $2\varepsilon s$ collisions is

$$\begin{aligned}&\binom{s}{2\varepsilon s} \cdot \left(\prod_{i=0}^{2\varepsilon s - 1} \frac{s-i}{k-i}\right) \cdot \left(\prod_{i=0}^{s-2\varepsilon s-1} \frac{k-i-s}{k-i-2\varepsilon s}\right) \\ &\geq \left(\frac{1}{2\varepsilon}\right)^{2\varepsilon s} \cdot \left(\frac{(1-2\varepsilon)s}{k}\right)^{2\varepsilon s} \cdot \left(1 - \frac{s}{k-s}\right)^{s-2\varepsilon s} \\ \text{(A.1)} \\ &\geq \left(\frac{s}{4\varepsilon k}\right)^{2\varepsilon s} \cdot \left(1 - \frac{2s}{k}\right)^s.\end{aligned}$$

It suffices for the right hand side to be at least $\sqrt{\delta}$ since h is independent of σ, and thus the total probability of error larger than 2ε would be greater than $\sqrt{\delta}^2 = \delta$. Taking natural logarithms, it suffices to have

$$2\varepsilon s \ln\left(\frac{4\varepsilon k}{s}\right) - s\ln\left(1 - \frac{2s}{k}\right) \leq \ln(1/\delta)/2.$$

Writing $s = q/\varepsilon$ and $a = 4C\log(1/\delta)$, the left hand side is $2q\ln(a/q) + \Theta(s^2/k)$. Taking a derivative shows $2q\ln(a/q)$ is monotonically increasing for $q < a/e$. Thus as long as $q < ca$ for a sufficiently small constant c, $2q\ln(a/q) < \ln(1/\delta)/4$. Also, the $\Theta(s^2/k)$ term is at most $\ln(1/\delta)/4$ for c sufficiently small.

A.3 Tightness of Figure 1(c) analysis

Theorem A.3. *For δ smaller than a constant depending on C for $k = C\varepsilon^{-2}\log(1/\delta)$, the scheme of Section 4 requires $s = \Omega(\varepsilon^{-1}\log(1/\delta))$ to obtain distortion $1 \pm \varepsilon$ with probability $1 - \delta$.*

Proof. First suppose $s \leq 1/(2\varepsilon)$. Consider a vector with $t = \lfloor 1/(s\varepsilon) \rfloor$ non-zero coordinates each of value $1/\sqrt{t}$. If there is exactly one set i, j, r with $i \neq j$ such that $h(i,r) = h(j,r)$ (i.e. exactly one collision), then the total error is $2/(ts) \geq 2\varepsilon$. It just remains to show that this happens with probability larger than δ.

The probability of exactly one collision is

$$\begin{aligned}
s\cdot &\left[\frac{t!\cdot\binom{k/s}{t}}{(k/s)^t}\right]^{s-1}\cdot\binom{t}{2}\cdot\binom{k}{s}\cdot\left[\frac{(t-2)!\cdot\binom{k/s-1}{t-2}}{(k/s)^t}\right]\\
&\geq s\cdot\left(1-\frac{st}{k}\right)^{t(s-1)}\cdot\binom{t}{2}\cdot\left(\frac{s}{k}\right)\left(1-\frac{st}{k}\right)^{t-2}\\
&=\frac{s^2t(t-1)}{2k}\cdot\left(1-\frac{st}{k}\right)^{st-2}\\
&\geq\frac{s^2t(t-1)}{2k}\cdot\left(1-\frac{s^2t^2}{k}\right)\\
&=\Omega(1/\log(1/\delta)),
\end{aligned}$$

which is larger than δ for δ smaller than a universal constant.

Now consider $1/(2\varepsilon) < s < c\cdot\varepsilon^{-1}\log(1/\delta)$ for some small constant c. Consider the vector $x = (1/\sqrt{2}, 1/\sqrt{2}, 0, \ldots, 0)$. Suppose there are exactly $2s\varepsilon$ collisions, i.e. $2s\varepsilon$ distinct values of r such that $h(1,r) = h(2,r)$ (to avoid tedium we disregard floors and ceilings and just assume $s\varepsilon$ is an integer). Also, suppose that in each colliding chunk r we have $\sigma(1,r) = \sigma(2,r)$. Then, the total error would be 2ε. It just remains to show that this happens with probability larger than δ. The probability of signs agreeing in exactly $2\varepsilon s$ chunks is $2^{-2\varepsilon s} > 2^{-2c\log(1/\delta)}$, which is larger than $\sqrt{\delta}$ for $c < 1/4$. The probability of exactly $2\varepsilon s$ collisions is

$$\begin{aligned}
\binom{s}{2\varepsilon s}&\left(\frac{s}{k}\right)^{2\varepsilon s}\left(1-\frac{s}{k}\right)^{(1-2\varepsilon)s}\\
&\geq\left(\frac{s}{2\varepsilon k}\right)^{2\varepsilon s}\left(1-\frac{s}{k}\right)^{(1-2\varepsilon)s}
\end{aligned}$$

The above is at most $\sqrt{\delta}$, by the analysis following Eq. (A.1). Since h is independent of σ, the total probability of having error larger than 2ε is greater than $\sqrt{\delta}^2 = \delta$.