Accelerated Dense Random Projections

A Dissertation

Presented to the faculty of the Graduate School

of

Yale University

in Candidacy for the degree of

Doctor of Philosophy

by

Edo Liberty

Dissertation Director: Steven Zucker

May 2009

Accelerated Dense Random Projections

Edo Liberty

May 2009

ABSTRACT

In dimensionality reduction, a set of points in $\mathbb{R}^d$ is mapped into $\mathbb{R}^k$, with the target dimension k smaller than the original dimension d, while distances between all pairs of points are approximately preserved. Currently popular methods for achieving this involve random projection, or choosing a linear mapping (a $k \times d$ matrix) from a distribution that is independent of the input points. Applying the mapping (chosen according to this distribution) is shown to give the desired property with at least constant probability. The contributions in this thesis are twofold. First, we provide a framework for designing such distributions. Second, we derive efficient random projection algorithms using this framework. Our results achieve performance exceeding other existing approaches. When the target dimension is significantly smaller than the original dimension we gain significant improvement by designing efficient algorithms for applying certain linear algebraic transforms. To demonstrate practicality, we supplement the theoretical derivations with experimental results.

Acknowledgments: Special thanks to Steven Zucker, Nir Ailon, Daniel Spielman, Ronald Coifman, Amit Singer and Mark Tygert. Their work, ideas and guidance can be seen throughout this manuscript.

1. LIST OF COMMON NOTATIONS

d ..... input vectors' original dimension

$\mathbb{R}^d$ ..... real d dimensional space

x or x_i ..... input vector(s) in $\mathbb{R}^d$

n ..... number of input vectors or a constant polynomial in that number

k ..... target dimension

ε ..... constant required precision, $0 < \varepsilon < 1/2$

$\mathbb{S}^{d-1}$ ..... the set $\{x \in \mathbb{R}^d \mid \|x\|_2 = 1\}$

$\|\cdot\|$ ..... the ℓ_2 norm for vectors and the Spectral norm for matrices

$\|\cdot\|_p$ ..... ℓ_p norm (for vectors)

$\|\cdot\|_{p\to q}$ ..... operator norm from ℓ_p to ℓ_q (for matrices)

H, H_d ..... a $d \times d$ Walsh Hadamard transform

± 1 ..... $\{+1, -1\}$

χ ..... a subset of $\mathbb{R}^d$

FJLT ..... the fast JL transform by Ailon Chazelle [1]

FJLTr ..... a revised version of the FJLT algorithm, chapter 3 and [2]

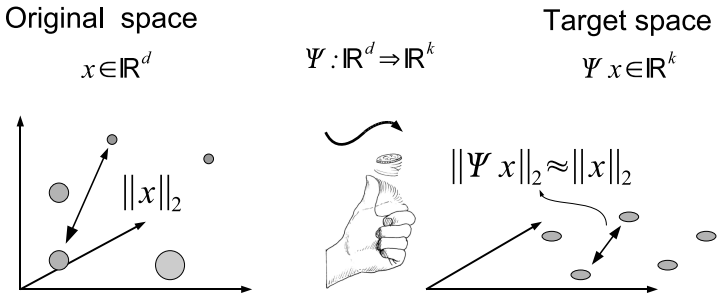
FWI ..... a fast projection algorithm described in chapter 5 and [2]

CONTENTS

2. INTRODUCTION TO RANDOM PROJECTIONS

2.1 Linear embedding and the JL property

In many applications one is given a set of n points in high dimension, say d, and is interested in embedding these points into a space of lower dimension, k, such that all distances are preserved almost exactly.



More precisely, given n vectors $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$ and a constant $0 \leq \varepsilon \leq 1/2$, we are interested in a mapping $\Psi : \mathbb{R}^d \to \mathbb{R}^k$ such that $k \ll d$ and

$$\forall i, j \;\; (1-\varepsilon)\|x_i - x_j\| \leq \|\Psi(x_i) - \Psi(x_j)\| \leq (1+\varepsilon)\|x_i - x_j\|. \tag{2.1}$$

Naturally, one might approach this task in a deterministic fashion by evaluating the incoming vectors. However, it is well known that a randomized approach to this problem is significantly easier. Johnson and Lindenstrauss in [3] were the first to give such a randomized construction. The final step in their proof is still common to all random projection schemes; Let Ψ be a linear mapping (a $k \times d$ matrix) chosen from a probability distribution $\mathbb{D}_{k,d}$ such that for any vector $x \in \mathbb{R}^n$, $\|\Psi x\|_2$ is approximately $\|x\|_2$ with high probability. Let the vector x be the difference $x_i - x_j$. Since Ψ is a linear operator we have that $\Psi(x_i) - \Psi(x_j) = \Psi(x_i - x_j) = \Psi(x)$. Moreover, since there are only $\binom{n}{2}$ pairwise distances, if the failure

probability (distortion larger than ε) is smaller than $1/n^2$, by the union bound, the entire metric is ε preserved with probability at least 1/2. Without loss of generality, we can consider only unit vectors, $\|x\|_2 = 1$. The notation $\mathbb{S}^{d-1}$ stands for the $d-1$ dimensional ℓ_2 sphere, i.e. the set $\{x \in \mathbb{R}^d \mid \|x\|_2 = 1\}$. We have that if

$$\forall x \in \mathbb{S}^{d-1} \quad \Pr_{\Psi \sim \mathbb{D}_{k,d}}[|\|\Psi x\| - 1| > \varepsilon] \leq \frac{1}{n^2} \tag{2.2}$$

then for every i and j simultaneously

$$\forall i, j \;\; (1-\varepsilon)\|x_i - x_j\| \leq \|\Psi(x_i) - \Psi(x_j)\| \leq (1+\varepsilon)\|x_i - x_j\| \tag{2.3}$$

with probability at least 1/2. Thus, a matrix, Ψ, chosen according to a distribution $\mathbb{D}_{k,d}$ which satisfies Equation 2.2 exhibits Equation 2.3 with at least constant probability.

Definition 2.1.1. *A distribution $\mathbb{D}_{k,d}$ over $k \times d$ matrices is said to exhibit the JL (Johnson Lindenstrauss) property if*

$$\forall x \in \mathbb{S}^{d-1} \quad \Pr_{\Psi \sim \mathbb{D}_{k,d}}[|\|\Psi x\| - 1| > \varepsilon] \leq c_1 e^{-c_2 k \varepsilon^2} \tag{2.4}$$

for some constants c_1 and c_2.

Lemma 2.1.1 (Johnson, Lindenstrauss [3])**.** *Let $\mathbb{D}_{k,d}$ denote the uniform distribution over all $k \times d$ projection matrices. The distribution $\mathbb{D}_{k,d}$ exhibits the JL property.*[1]

The surprising fact about the JLlemma (Lemma 2.1.1) is that Equation (2.2) is satisfied for $k = \Omega(\log(n)/\varepsilon^2)$. This means that *any* set of n points in $\mathbb{R}^d$ (equipped with the ℓ_2 metric) can be embedded into dimension $k = \Theta(\log(n)/\varepsilon^2)$ with distortion at most ε using a randomly generated linear mapping Ψ. Moreover Noga Alon [4] showed that this result is essentially tight (in its dependence on n).

It is remarkable to notice that the target dimension k is not only logarithmic in the input size (n) but also independent of the original dimension d. Further, the randomized algorithm which achieves this is independent of the input vectors $\{x_1, \ldots, x_n\}$, (depends only on their number). These properties make random projections a critical ingredient in many algorithms. Examples for such application can by found in:

[1] In Linear Algebra, a projection matrix P is a *square* matrix such that $P = P^2$. Here, a rectangular matrix Ψ is a said to be a projection if $\Psi^T\Psi = (\Psi^T\Psi)^2$.

approximate nearest neighbor searching [5, 6, 7, 8, 9], learning [10, 11, 12], matrix low rank approximation [13, 14, 15, 16, 17, 18, 19, 20], other linear algebraic operations [21, 22, 23, 24], and many other algorithms and applications, e.g, [25, 26, 27, 28, 29, 30].

2.2 Classic results, review of known JL distributions

The construction of Johnson and Lindenstrauss is surprisingly simple. They proposed choosing Ψ uniformly at random from the space of projection matrices. In other words, Ψ contains a random k dimensional subspace of $\mathbb{R}^d$. One technique to sample from this distribution is to start with a $k \times d$ matrix whose entries are chosen i.i.d. according to the Gaussian distribution, then orthogonalize and normalize its rows (using the Grahm-Schmidt procedure, for example). The idea of their proof is as follows; the success probability of accurately projecting any *fixed* vector x onto a random subspace is equal to the success probability of projecting a *random* vector in $\mathbb{R}^d$ onto the first k vectors of the canonical basis (its first k coordinates). Johnson and Lindenstrauss proceed to give the relevant concentration over $\mathbb{S}^{d-1}$ which proves the lemma.

Their proof, however, can be made significantly simpler by considering a slightly modified construction. Gupta and Dasgupta [31] as well as Frankl and Maehara [32] suggested that each entry in Ψ be chosen uniformly at random from a Gaussian distribution (without orthogonalization). These proofs still relay on the rotational invariance of the distribution but are significantly easier. A sketch of a possible proof is given below.[2]

Since the distribution of Ψ is the same as the that of $\Psi' = \Psi U$ for any complete orthogonal transformation U. We look at $\Psi U U^T x$ for U such that $Ux = e_1 = (1, 0, \ldots, 0)^T$ (w.l.o.g $\|x\|_2 = 1$). Since $\|\Psi' e_1\|_2$ is the ℓ_2 norm of $\Psi'^{(1)}$, the first column of Ψ', and Ψ' distributes like Ψ, we need only show that $\Pr[\big|\|\Psi^{(1)}\| - 1\big| > \varepsilon] \leq c_1 e^{-c_2 k \varepsilon^2}$. Note that the term $\|\Psi^{(1)}\|$ is simply sum of squared values of k i.i.d. Gaussian distributed variables. The statistical concentration is achieved using well studied properties of the χ^2 distribution.

More difficult to prove are cases where the distribution is not rotationally invariant. The first such

[2] The given proof is not the one supplied in [31] or in [32]. The authors encountered this idea in several informal discussions but are not aware of its specific origin.

construction was given by Dimitris Achlioptas [33] who proposed a distribution over matrices Ψ such that $\Psi(i,j) \in \{-1,0,1\}$ with constant probabilities. Matousek [34] extended this result to any i.i.d. sub-Gaussian symmetric distributed entries. These proofs relay on a slightly weaker condition which is the independence of the rows of Ψ. Denote by $\Psi_{(i)}$ the i'th row of Ψ, $\|\Psi x\|^2 = \sum_{i=1}^{k} \langle \Psi_{(i)}, x \rangle^2$. We notice that if the rows of $\Psi_{(i)}$ are i.i.d. then $\langle \Psi_{(i)}, x \rangle^2$ are also i.i.d. By characterizing the distribution of the variables $\langle \Psi_{(i)}, x \rangle^2$ one can derive a concentration result using a quantitative version of the central limit theorem. We review this in more detail further into the introduction.

2.3 Motivation for accelerating random projections

In many of the applications mentioned in section 2.1 the original dimension d is very large whereas the target dimension in manageable in size. An implementation of any of the constructions described in the last section requires $O(kd)$ bits to store and $O(kd)$ operations to apply to each input vector. This is, in many cases, impractical. For example, if the input vectors are grey scale values of 5 megapixels images, and the target dimension, k, is 1000, the matrix Ψ will occupy 20 gigabytes of space/memory (in 4 byte float precision).[3] This amount of memory use is (as of today) extremely inconvenient. On most modern machines it will invoke intensive paging and thus perform very poorly.

In some situations, one might be able to avoid storing the matrix Ψ altogether; e.g., when the task is to embed a fixed given set of points. In this case, one can successively generate each row $\Psi_{(i)}$, compute its dot product with all input vectors, store the results and discard the row $\Psi_{(i)}$. This way only one row of Ψ is stored in any point in time. However, this 'generate-and-forget' method can not be used in the Nearest Neighbor setup, for example, becauset query points must be projected using the same matrix as the data points and are not known in advance. Nevertheless, this idea is used in practice quite intensively mainly due to its simplicity.

Regardless of space usage and handling, applying these matrices is prohibitive from the time stand point.

[3] It is worth mentioning that Achlioptas's matrix will only occupy 1/32 of this amount since each entry is representable by 1 bit instead of a 32 bit float.

For the described modest application, even when the matrix fits in memory, the running time on a $3Ghz$ processor will be in the minutes, for *each* projected vector. This prevents this method from being used in any real time system or on any moderately large data set. If random projection is to be used in practice one must come up with faster, more efficient, ways of accomplishing it.

2.4 Sparse Projective matrices

One possible approach to accelerate the projection process is to seek JLdistributions over sparse matrices. Applying a matrix to any vector requires as many operations as the number of non-zeros it contains. If the i.i.d. entries in Ψ are very likely to be zero, the number of non-zeros in Ψ should be much less than kd. However intuitive this idea is, it can be shown that, unless the number of non-zeros in Ψ is $\Omega(kd)$, the same success probability cannot be achieved. One way to see this is to consider projecting the vector $[1, 0, \ldots, 0]^T$. The resulting quantity $\|\Psi x\|_2$ is exactly the norm of the first column of Ψ. Hence, the norm of the first column of Ψ must statistically concentrate around 1, i.e., $|\sum_{i=1}^{k} \Psi(i,1)^2 - 1| \leq \varepsilon$ with probability at least $1 - 1/n$. Since any event with probability $1 - 1/n \leq \Pr < 1$ must rely on $\Omega(\log(n))$ random bits we get that any column of Ψ contains $\Omega(\log(n))$ non-zeros.[4] Since ε is a constant, each column contains $\Omega(\log(n)) = \Omega(k)$ non-zeros. Therefore, the entire matrix contains $O(kd)$ non-zeros and our hope for sparse projection matrices is shattered.

As we saw, sparse matrices cannot exhibit the JL property. We showed this by considering success probability of accurately projecting sparse vectors like $x = [1, 0, \ldots, 0]^T$. Notice however, that such sparse vectors are relatively "few". By "few" we mean that a very small portion of $\mathbb{S}^{d-1}$ is occupied by them. We can therefore ask: can sparse projective matrices preserve lengths of non-sparse vectors? This question was posed and answered by Ailon and Chazelle in [1], and resulted in the FJLT algorithm. It is worth mentioning that this result was the first to give an asymptotical acceleration of a JL transform. Their work gave theoretical insights and motivations that can be seen throughout this document. They showed that, if the ℓ_∞ norm of the input vector $x \in \mathbb{S}^{d-1}$ is bounded by $O(\sqrt{k/d})$, then Ψ can be chosen from a

[4] Under the assumption that producing each $\Psi(i,j)$ requires a constant number of random bits.

distribution over sparse matrices which contain only $O(k^3)$ non-zeros in expectation. This result was then generalized by Matousek [34] who showed that any vector x such that $\|x\|_\infty \leq \eta$ can be projected with at most ε distortion and high probability by a matrix containing only $O(k^2\eta^2 d)$ non-zeros in expectation. Ailon and Chazelle further showed that any vector $x \in \mathbb{R}^d$ can be isometrically, randomly, rotated such that $\|\Phi x\|_\infty \leq \sqrt{k/d}$ w.h.p. Here Φ is a fast randomized isometry which requires only $O(d\log(d))$ operations to apply. For completeness we show a sketch of the proof by Matousek [34].

Consider the term $\|\Psi x\|_2^2 = \sum y^2(i)$ where $y_i = \langle \Psi_{(i)}, x\rangle$ and $\Psi_{(i)}$ denotes the i'th row of Ψ. Since the entries of Ψ are i.i.d., so are the random variables y_i. We need only show that the sum of k such variables concentrates in the right way. It turns out that it is sufficient for y_i to be distributed s.t. $E[y_i] = 0$, $Var[y_i] = k^{-1/2}$ and y_i have a uniform sub-Gaussian tail. The definitions are given below.

Definition 2.4.1 (Matousek [34])**.** *A real random variable Y is said to have a sub-Gaussian upper tail if for some constant α:*

$$\Pr(Y > t) \leq E[e^{-\alpha t^2}] \tag{2.5}$$

for all $t > 0$. If this condition holds only up to some $t \leq t_0$, the distribution of Y is said to have a sub-Gaussian upper tail up to t_0. A collection of variables Y_i is said to have uniform *sub-Gaussian upper tail if they are all sub-Gaussian with the same constant α.*

Lemma 2.4.1 (Matousek [34])**.** *Let Y_i be i.i.d. random variables with $E[Y_i] = 0$, $Var[Y_i] = 1$ and Y_i have a uniform sub-Gaussian tail up to at least $\sqrt{k}$. Define the random variable $Z = \frac{1}{\sqrt{k}}(\sum_{i=1}^k Y_i^2 - 1)$. The variable Z has a sub-Gaussian tail up to at least $\sqrt{k}$.*

Proving lemma 2.4.1 turns out to be rather technical and it is given in full detail in [34]. However, given its correctness, the JL property follows almost directly. Let $Y_i = \sqrt{k}y_i$ where $y_i = \langle \Psi_i, x\rangle$. First notice that:

$$\begin{aligned} \|\Psi x\|_2^2 - 1 &= \sum_{i=1}^k y_i^2 - 1 && (2.6)\\ &= \frac{1}{k}\sum_{i=1}^k Y_i^2 - k && (2.7)\\ &= \frac{1}{\sqrt{k}} Z && (2.8) \end{aligned}$$

where $Z = \frac{1}{\sqrt{k}}(\sum_{i=1}^{k} Y_i^2 - 1)$. Assume that $E[Y_i] = 0$, $Var[Y_i] = 1$ and Y_i exhibit a uniform sub-Gaussian upper tail up to $\sqrt{k}$. According to lemma 2.4.1, Z is sub-Gaussian up to $\sqrt{k}$, which is used as follows:

$$\begin{aligned} \Pr[|\|\Psi x\|_2 - 1| > \epsilon] &\leq 2\Pr[\|\Psi x\|_2^2 - 1 > 2\epsilon] && (2.9) \\ &= 2\Pr[Z > 2\sqrt{k}\epsilon] && (2.10) \\ &\leq 2e^{-C(2\sqrt{k}\epsilon)^2} = c_1 e^{-c_2 k\epsilon^2} && (2.11) \end{aligned}$$

The last equation follows from the fact that Z is sub-Gaussian up to $\sqrt{k}$ and $\varepsilon \leq 1/2$. This matches the success probability required by the JL property (definition 2.1.1).

A simple example can be random Gaussian entries for $\Psi(i,j)$ (normalized be $1/\sqrt{k}$). Due to the rotational invariance of the Gaussian distribution Y_i is, itself, distributed like a Gaussian with mean zero and variance one. Clearly a Gaussian has a sub-Gaussian upper tail. This proves the JL lemma in yet another way.

Matousek further gives the connection between the ℓ_∞ norm of x and the required (expected) density of Ψ for Y_i to exhibit the appropriate sub-Gaussian tails. Let $s(i)$ be i.i.d. copies of s such that

$$s = \begin{cases} +\frac{1}{\sqrt{q}} & \text{with probability } q/2 \\ -\frac{1}{\sqrt{q}} & \text{with probability } q/2 \\ 0 & \text{with probability} 1-q \end{cases} \tag{2.12}$$

Lemma 2.4.2 (Matousek [34]). *Let $\eta^2 \leq q$, let $x \in \mathbb{S}^{d-1}$ such that $\|x\|_\infty \leq \eta$, and let $Y = \sum_{i=1}^{d} s(i)x(i)$, where the $s(i)$ are as described above. Then Y has a sub-Gaussian tail up to $\sqrt{2q}/\eta$.*

Combining lemmas 2.4.1 and 2.4.2 gives the minimal expected sparsity for Ψ required to successfully project a vector $x \in \mathbb{S}^{d-1}$ whose ℓ_∞ norm is bounded by η. Lemma 2.4.1 requires $\sqrt{2q}/\eta \geq \sqrt{k}$ and thus $q = \Theta(\eta^2 k)$. The expected number of non-zeros in Ψ is therefore $kdq = \Theta(k^2\eta^2 d)$. Putting these ideas together proves the following Lemma 2.4.3.

Lemma 2.4.3 (Matousek [34]). *Let $\eta \in [1/\sqrt{d}, 1]$ be a constant. Set q to be*

$$q = C_0 \eta^2 k \tag{2.13}$$

for a sufficiently large constant C_0. *Let* $\Psi(i,j)$ *be i.i.d.*

$$\Psi(i,j) = \begin{cases} +\frac{1}{\sqrt{qk}} & \textit{with probability } q/2 \\ -\frac{1}{\sqrt{qk}} & \textit{with probability } q/2 \\ 0 & \textit{with probability } 1-q \end{cases} \tag{2.14}$$

The lemma claims that:

$$\Pr\left[|\|\Psi x\| - 1|\right] \leq c_1 e^{-c_2 k\varepsilon^2} \tag{2.15}$$

for all $x \in \mathbb{S}^{d-1}$ *such that* $\|x\|_\infty \leq \eta$.

Furthermore, Matousek shows that the above lower bound for q is, up to a constant multiplicative factor, tight.

We now discuss the FJLT algorithm. It begins by randomly and *isometrically* rotating the input vector x such that the rotated vector's ℓ_∞ norm is bounded by $O(\sqrt{\log(n)/d})$ (and in that sense behaves like a random vector). According to Lemma 2.4.3 such vectors can be accurately projected into $\mathbb{R}^k$ using a matrix containing only $O(k^3)$ entries in expectation. This is, of course, better than $O(kd)$ for any $k \in o(d^{1/2})$. The result is recaped in Lemma 2.4.4.

Lemma 2.4.4 (Ailon, Chazelle [1])**.** *Let* $\Psi = PHD$ *be chosen according to the following distribution:*

- H*: The Walsh Hadamard* $d \times d$ *Matrix (deterministic).*
- D*: A diagonal matrix whose entries are distributed i.i.d. uniformly over* $\{-1, +1\}$.
- P*: A* $k \times d$ *matrix whose i.i.d. entries are either zero with probability* $1-q$ *or normally distributed according to* $N(0, q^{-1})$ *with probability* q.

where $q = \Theta(\varepsilon^3 \log^2(n) d^{-1})$.[5] *The distribution for* Ψ *exhibits the JL property. Moreover, applying* Ψ *to any vector* $x \in \mathbb{R}^d$ *requires* $O(d\log(d) + \min\{kd, \log^3(n)\varepsilon^{-2}\})$ *operations in expectation.*

Given the formalization above, to prove this lemma we need only show that

$$\forall\, x \in \mathbb{S}^{d-1} \quad \Pr[\|HDx\|_\infty > \sqrt{log(n)/d}] \leq 1/n. \tag{2.16}$$

[5] We assume that $q < 1$. if $q \geq 1$ we set $q = 1$ and the claim is trivial due to [32]

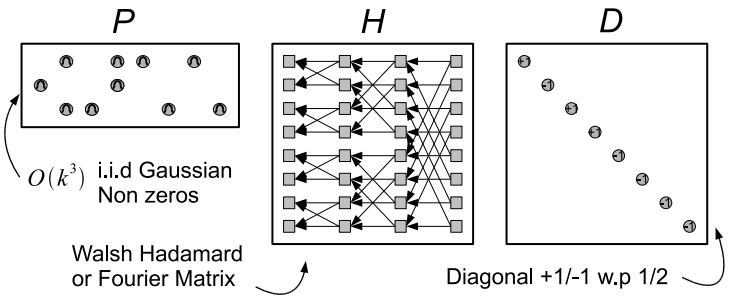


Fig. 2.1: A sketch of the FJLT construction. The FJLT algorithm was given in [1] and is recaped in lemma 2.4.4.

This can be easily seen by noticing that each $H(i,j) = \pm 1/\sqrt{d}$ and so $(HDx)(i) = \frac{1}{\sqrt{d}} \sum_{j=1}^{d} b(j)x(j)$ where $b(j)$ are ± 1 variables w.p. $1/2$ each. Using the Hoeffding bound and then a union bound for all coordinates and and all vectors yields the desired result.

Finally, the running time can be thought of as $O(d \log(d) + k^3)$, where $d \log(d)$ operations are required to perform the Hadamard transform and $O(k^3)$ to apply the sparse projective matrix P.[6]

2.5 Our contributions

The contributions in this thesis can be organized into three groups. The first gives efficient algorithms for computing several linear transforms. This cleans up a number of inefficiencies in the FJLT algorithm and the way it is applied. The second gives, relatively simple, sufficient conditions for a matrix distribution to exhibit the JLproperty, thus a framework for finding such distributions emerges. Finally, working within this framework, we develop a number of new algorithms which reduce the required randomness and running time of current random projection algorithms. We now discuss each of these in more detail.

2.5.1 Fast linear transforms

The first improvement revises the FJLT algorithm, to remove inefficiencies in the way it was originally described. When k is significantly smaller than d, the FJLT algorithm computes a complete d dimensional Fourier or Hadamard transform, but this is inefficient since at most $O(k^3)$ coefficients are required (potentially

[6] It is worth mentioning that this result holds also for mapping into $\mathbb{R}^k$ equipped with the ℓ_1 matric.

much less than d). We give simple algorithms which compute any k' coefficients out of a $d \times d$ Hadamard or Fourier transform in $O(d \log(k'))$ operations instead of $O(d \log(d))$. This simple modification reduces the running time of the $FJLT$ algorithm to $O(d \log(k) + k^3)$ which gives an improvement over the inefficient construction whenever $k \in o(\text{poly}(d))$. The revised FJLT algorithm is denoted by FJLTr and is discussed in Chapter 3.

We also consider algorithms for applying unstructured matrices to real vectors. We show that given any $O(\log(d)) \times d$ matrix over a finite alphabet, A, one can apply A to any vector in $\mathbb{R}^d$ in $O(d)$ operations. Although similar results have been known for over fifty years our construction is new and slightly more general. We nickname our algorithm the mailman algorithm and we describe it in Chapter 7. Applying a matrix chosen according to Achlioptas' ± 1 JL distribution using the Mailman algorithm, gives an $O(d)$ running time random projection algorithm, for the case where $k \in O(\log(d))$. This matches the running time lower bound for random projections.

2.5.2 A two stage framework

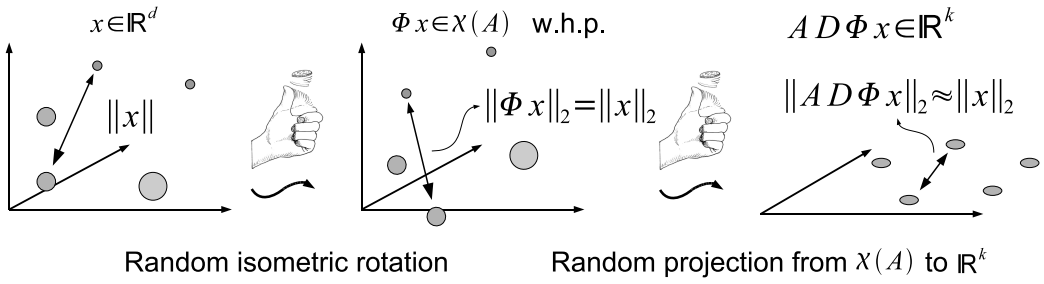


Fig. 2.2: A random projection algorithm consisting of two stages. First, vectors are rotated isometrically in $\mathbb{R}^d$ so that they lay in a set $\chi(A) \subset \mathbb{R}^d$ w.h.p. Then they are projected into dimension k using the $k \times d$ matrix A, composed with a random diagonal ± 1 matrix D.

Inspired by the ideas in [1] and later [34], we consider linear dimensionality reduction as a two stage process. In the first stage, each vector $x \in \mathbb{S}^{d-1}$ is rotated isometrically in $\mathbb{R}^d$, using a random isometry Φ, such that, with high probability, Φx lies in a subset of $\mathbb{S}^{d-1}$ which we denote by χ. A useful example

discussed above for χ is the set of unit vectors such that $\|x\|_\infty \leq \eta$ for some constant η. In the second stage, the vectors Φx are projected into dimension k using a $k \times d$ matrix A. Our requirement for A is that

$$\forall x \in \chi \ \Pr\left[\left|\|ADx\|_2 - 1\right| \geq \varepsilon)\right] \leq 1/n. \tag{2.17}$$

The matrix D is a diagonal random ± 1 matrix. It is added so that A itself can potentially be deterministic. In words, the requirement from A is that it (when composed with D) projects vectors from χ into $\mathbb{R}^k$ with high probability and accuracy. There is no such guaranty for vectors not in χ. We say that χ is the probabilistic domain of A. Intuitively, this requirement is easier to fulfill for a smaller set χ. However, mapping all vectors into a smaller set χ (applying Φ) might require more randomness and computational effort. Chapter 4 is dedicated to characterizing the relation between any matrix A and its corresponding probabilistic domain χ.

2.5.3 Dense fast projective matrices

The accelerations described thus far all rely on the spareness of the projection matrix A and require the independence of its entries. Using the aforementioned connection between matrices and their probabilistic domain, we consider *dense* and *deterministic* choices for A and show their usefulness.

In Chapter 5 we take A to be a ± 1 four-wise independent matrix (definition 5.1.2). We show that one can apply both A and its appropriate Φ in time $O(d \log(k))$ for $k \in O(d^{1/2-\delta})$ for any positive constant δ. This matches the FJLTr algorithm when $k \in O((d \log(d))^{1/3})$ but outperforms it when $k \in O(d^{1/2-\delta})$ and $k \in \omega((d \log(d))^{1/3})$.

The second construction, described in Chapter 6, is a dense orthogonal ± 1 matrix which can be applied to any vector in $\mathbb{R}^d$ in linear time, i.e. $O(d)$. We term these matrices *lean Walsh matrices*. Since the trivial lower bound on the running time of dimension reduction is $O(d)$, searching for projections which require this amount of computational effort is worthwhile. We show that lean Walsh matrices are strictly better suited for the task of random projections than sparse matrices, i.e. exhibit a strictly larger probabilistic domain χ. It is, however, not clear if there exists a corresponding random rotation which is also applicable in $O(d)$) operations.

2.5.4 Results summary

	Naïve or Slower	Faster than naïve	$O(d\log(k))$	Optimal, $O(d)$
k in $o(\log d)$	JL, FJLT, FWI		FJLTr	JL + Mailman
k in $\omega(\log d)$ and $o(\text{poly}(d))$	JL	FJLT, FWI	FJLTr	
k in $\Omega(\text{poly}(d))$ and $o((d\log(d)^{1/3})$	JL		FJLT, FJLTr, FWI	
k in $\omega((d\log d)^{1/3})$ and $O(d^{1/2-\delta})$	JL	FJLT, FJLTr	FWI	
k in $O(d^{1/2-\delta})$ and $k < d$	JL, FJLT, FJLTr	JL concatenation		

Tab. 2.1: Results summary. Schematic comparison of asymptotic running time of six projection algorithms for different values of k and d. The projection algorithms denoted by, FJLTr, FWI, JL concatenation and JL + Mailman are developed in this thesis. (1) JL: a naïve implementation of the Johnson-Lindenstrauss lemma. (2) FJLT: the fast JL transform by Ailon Chazelle [1]. (3) FJLTr: a revised version of the FJLT algorithm, Chapter 3 and [2]. (4) FWI: a projection algorithm that uses the properties of four-wise independent matrices. Chapter 5 and [2]. (5) JL concatenation: a concatenation of several independent projections. Section 5.5 (6) JL + Mailman: implementation of the Mailman algorithm [35] to Achlioptas's result, Chapter 7. Except for the case where k is $\Omega(\text{poly}(d))$ and $o((d\log(d))^{1/3})$ our results strictly outperform previous algorithms.

	$k \times d$ projection matrix	Application complexity	$x \in \chi$ if $\|x\|_2 = 1$ and:
Johnson, Lindenstrauss [3]	Random k dimensional subspace	$O(kd)$	
Various Authors [32, 31, 33]	i.i.d. random entries	$O(kd)$	
Ailon, Chazelle [1]	Sparse i.i.d. Gaussian entries	$O(k^3)$	$\|x\|_\infty = O((d/k)^{-1/2})$
Matousek [34]	Sparse ± 1 entries	$O(k^2 d\eta^2)$	$\|x\|_\infty \leq \eta$
General rule	**Any matrix**		$\|x\|_A = O(k^{-1/2})$
This thesis [2]	4-wise independent matrix	$O(d \log k)$	$\|x\|_4 = O(d^{-1/4})$
This thesis [36]	Lean Walsh Transform	$O(d)$	$\|x\|_\infty = O(k^{-1/2} d^{-\delta})$
This thesis	Identity copies	$O(d)$	$\|x\|_\infty = O((k \log k)^{-1/2})$

Tab. 2.2: Different distributions for $k \times d$ matrices and the set $\chi \subset \mathbb{S}^{d-1}$ for which they constitute a good random projection. Fixed matrices (last four rows) are composed with a random ± 1 diagonal matrix, see Section 4.2. The meaning of $\|\cdot\|_A$ is given in Definition 4.2.1.

3. REVISED FAST JOHNSON LINDENSTRAUSS TRANSFORM

In this chapter we review the Fast Johnson Lindenstrauss transform result by Ailon and Chazelle [1] and improve its running time by revising it slightly. The FJLT algorithm (recaped in subsection 3.1) performs dimensionality reduction from dimension d to dimension k in expected $O(d\log(d) + k^3)$ operations (per vector). This running time cannot be optimal because when $k \in o(\log(d))$ we have $dk \in o(d\log(d) + k^3)$ implying that the FJLT algorithm is slower than a naïve implementation of the JL lemma. This chapter is dedicated to revising the FJLT algorithm to use efficient versions of either Hadamard or Discrete Fourier transforms to improve its running time to $O(d\log(k) + k^3)$. Other than using more efficient transforms this revised FJLT algorithm (FJLTr) is identical to the FJLT algorithm and its correctness does not need to be reproved.

3.1 Review of the FJLT algorithm

The FJLT result claims that a composition of three matrices PHD exhibits the JLP for P, H, and D being:

- D: A diagonal matrix whose diagonal entries are i.i.d. over $\{+1, -1\}$ uniformly.
- H: Either a Walsh Hadamard $d \times d$ Matrix or a $d \times d$ Discrete Fourier matrix.
- P: A k by d matrix whose i.i.d. entries are either zero with probability $1-q$ or independently normally distributed according to $N(0, q^{-1})$ with probability q.

Here $q = \Theta(\varepsilon^3 \log^2(n) d^{-1})$ and we assume that $q < 1$. (If $q \geq 1$ we set $q = 1$ and the claim is trivial due to prior constructions).

Lemma 3.1.1 (Ailon, Chazelle [1]). *Let $\Psi = PHD$ be chosen according to the distribution described*

above, then Ψ exhibits the JL property. Moreover, applying Ψ to any vector $x \in \mathbb{R}^d$ requires $O(d\log(d) + \min\{kd, \log^3(n)\varepsilon^{-2})\}$ operations in expectation.

The running time required for applying Ψ depends on the number of nonzeroes, n_{nnz}, in P, $E(n_{nnz}) = kdq = k\varepsilon^3\log^2(n)$. By recalling that $k = \Theta(\log(n)/\varepsilon^2)$ and viewing ε as a constant we get that $E(n_{nnz}) = O(k^3)$. Using Markov's inequality for the random variable n_{nnz}, $\Pr(n_{nnz} > \frac{1}{\delta}E(n_{nnz})) \leq \delta$. Thus, the case where $n_{nnz} > \frac{1}{\delta}E(n_{nnz})$ can be added to the overall failure probability and the number of non-zeros in P can be thought of as $O(k^3)$.[1]

Observe that computing $\Psi x = PHDx$ as proposed by the FJLT algorithm, i.e., $P(H(Dx))$, is wasteful. Clearly, computing only the relevant coefficients, at most as many as the number of non-zeros in P, would be better. The next two sections describe how to efficiently achieve this for both the Walsh Hadamard and the Distraite Fourier transforms.

3.2 Trimmed Walsh Hadamard transform

The Walsh Hadamard transform of a vector $x \in \mathbb{R}^d$ is the result of the matrix-vector multiplication Hx where H is a $d \times d$ matrix whose entries are $H(i,j) = (-1)^{\langle i,j\rangle}$. Here $\langle i,j\rangle$ means the dot product over $\mathbb{F}_2$ of the bit representation of i and j as binary vectors of length $\log(d)$.

The number of operations to compute a single coefficient $(Hx)(i) = \sum_{i=0}^{d-1}(-1)^{\langle i,j\rangle}x(j)$, denoted by $T(d,1)$, is $O(d)$. Moreover, all d coefficients can be computed in $O(d\log(d))$. We remind the reader that the Walsh-Hadamard matrix (up to normalization) can be constructed recursively.

$$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H_q = \left(\begin{array}{c:c} H_{q/2} & H_{q/2} \\ \hdashline H_{q/2} & -H_{q/2} \end{array}\right)$$

Claim 3.2.1. *Any k' coefficients of a $d \times d$ Hadamard transform can be computed in $O(d\log(k'))$ operations.*

Proof. Computing k' coefficients out of a $d \times d$ Hadamard transform H can be viewed as computing the outcome of PHx where P is $k' \times d$ matrix containing k' non-zeros, one per row. Define x_1 and x_2 to be the

[1] The random variable n_{nnz} actually concentrates much more sharply than the above bound, but here, using a Markov inequality suffices.

first and second halves of x. Similarly, define P_1 and P_2 as the left and right halves of P respectively.

$$PH_q x = \left(\begin{array}{c:c} P_1 & P_2 \end{array} \right) \left(\begin{array}{c:c} H_{q/2} & H_{q/2} \\ \hdashline H_{q/2} & -H_{q/2} \end{array} \right) \left(\begin{array}{c} x_1 \\ \hdashline x_2 \end{array} \right) \tag{3.1}$$

$$= P_1 H_{q/2}(x_1 + x_2) + P_2 H_{q/2}(x_1 - x_2)$$

Let P_1 and P_2 contain k_1' and k_2' non-zeros respectively, $k_1' + k_2' = k'$. Let $T(d, k')$ denote the number of operations required to compute k' coefficients out of a $d \times d$ Hadamard transform. Equation 3.1 yields the following recurrence relation

$$T(d, k') = T(d/2, k_1') + T(d/2, k_2') + d. \tag{3.2}$$

The base cases are $T(d, 0) = 0$ and $T(d, 1) = d$. We use induction to show that $T(d, k') \leq 2d \log_2(k' + 1)$.

$$\begin{aligned} T(d,k) &= T(d/2, k_1') + T(d/2, k_2') + d \quad \text{by the recurrence relation} \\ &\leq 2\frac{d}{2}\log_2(k_1' + 1) + 2\frac{d}{2}\log_2(k_2' + 1) + d \quad \text{by the induction assumption} \\ &\leq 2d\log_2(\sqrt{2(k_1' + 1)(k_2' + 1)}) \\ &\leq 2d\log_2(k_1' + k_2' + 1) \ \text{ for all } \ k_1' + k_2' = k' \geq 1 \\ &\leq 2d\log_2(k' + 1) \end{aligned}$$

The last sequence of inequalities together with the base cases also give a simple and efficient Divide and Conquer algorithm; see also Figure 3.2. □

3.3 Trimmed Discrete Fourier Transform

Since in [1] both Hadamard and Fourier transforms were considered, for completeness we also describe a simple trimmed version of the Coley Tukey fast Discrete Fourier Transform (DFT) algorithm (Figure 3.3).

The Discrete Fourier Transform $f_x = DFT(x)$ for a vector $x \in \mathbb{R}^d$ is given by $f_x(i) = \sum_{j=0}^{d-1} e^{2\pi\sqrt{-1}ij/d} x(j)$. Thus, any single coefficient $f_x(i)$ is easily computed in $O(d)$ operations. Fast DFT algorithms compute all

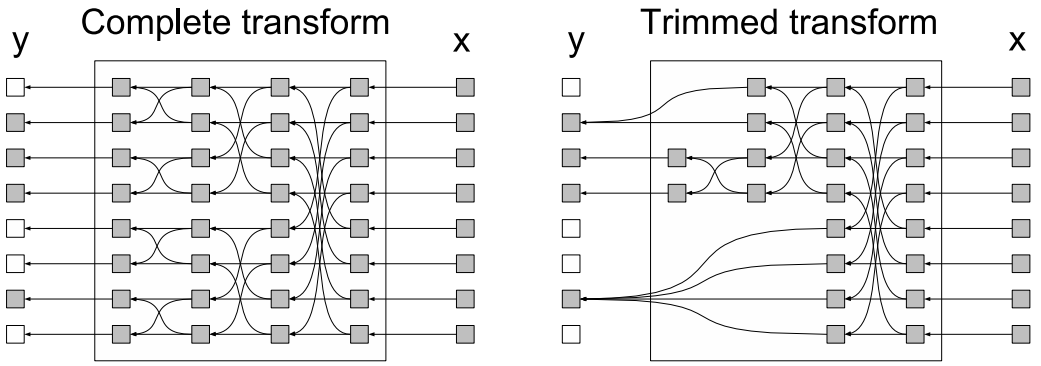


Fig. 3.1: A diagram describing the complete and trimmed Walsh Hadamard transforms.

d coefficients in $O(d\log(d))$ operations instead of $O(d^2)$. The Coley Tukey algorithm is a generalization of the more well known radix-2 algorithm. A sketch of the algorithm is shown in Figure 3.3.

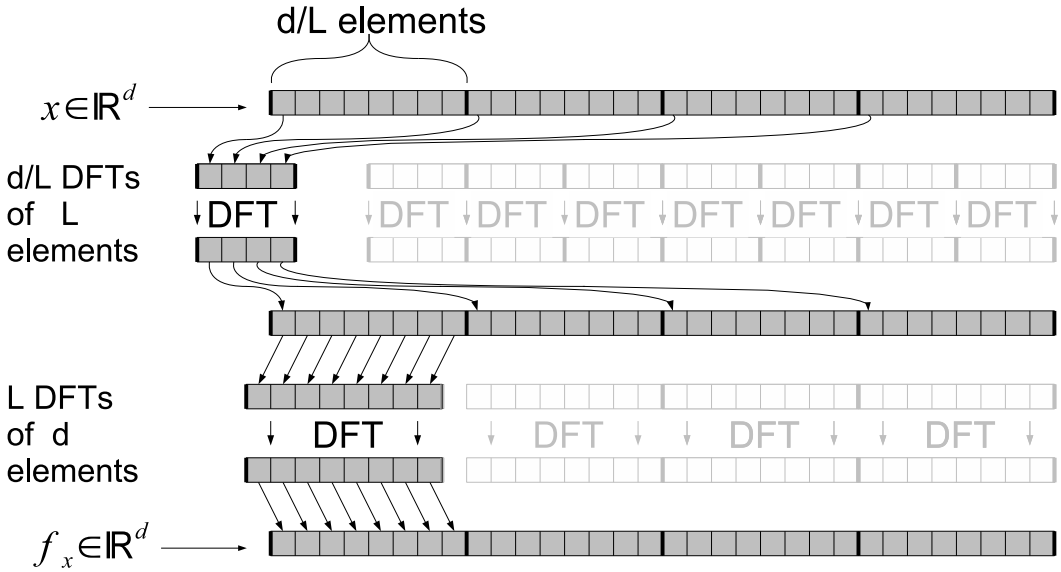


Fig. 3.2: A sketch describing the famous Fast DFT Coley Tukey Algorithm. x is sectioned into L blocks of length L/d each. First one performs d/L DFTs (recursively) to corresponding elements between blocks. The results are multiplied by "twiddle factors" and stored in a temporary arrays. Then, L DFTs of size d/L are performed within each block. This gives a total running time of $O(d\log(d))$ assuming L divides d. The sketch does not show the multiplication by twiddle factors of the temporary array.

In order to compute only k' coefficients of $DFT(x)$ we divide x into L blocks of size d/L and begin with the first step of the Cooley Tukey algorithm, which performs d/L DFTs of size L between the blocks (and multiplies them by twiddle factors). In the second step, instead of computing DFTs inside each block, each coefficient is computed naïvely, by summation, inside its block. These two steps require $O((d/L) \cdot L\log(L))$ and $O(k'd/L)$ operations respectively. By choosing $k'/\log(k') \leq L \leq k'$ we achieve a running time of $O(d\log(k'))$.

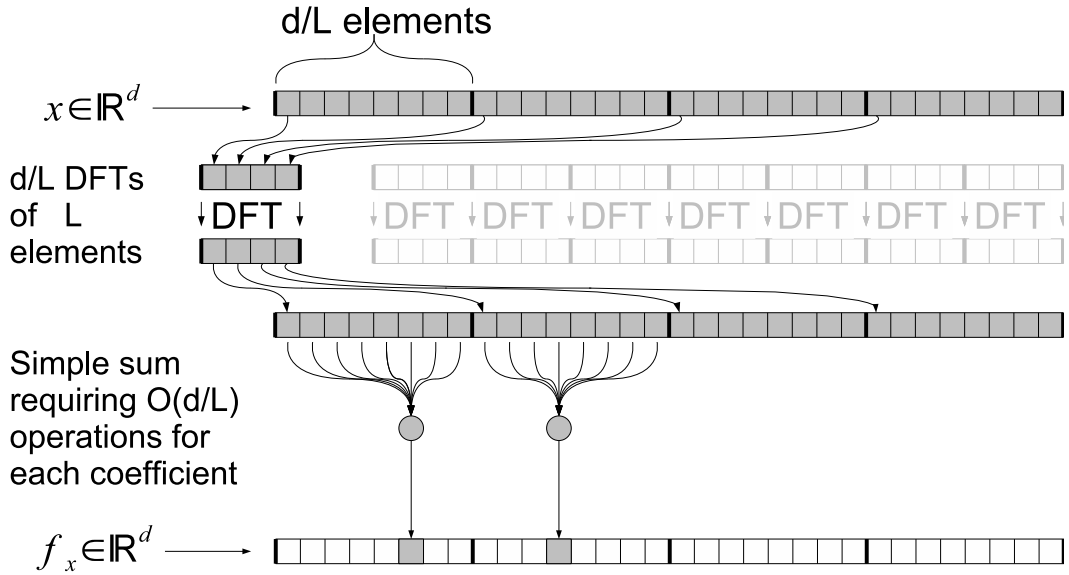


Fig. 3.3: A sketch describing the trimmed DFT Coley Tukey Algorithm. If $k'/\log(k') \leq L \leq k'^{O(1)}$ computing k' coefficient requires only $O(d\log(k'))$ operations.

3.4 FJLT revised

The proposed revised FJLT algorithm (FJLTr) is thus identical to the original FJLT algorithm except for replacing the application of Ψ as $P(H(Dx))$ by $(PH)(Dx)$ using the fast trimmed transforms above. Taking $k' = k^3$ yields the improved running time of $O(d\log(k) + k^3)$. Thus, for any $k \in o(\text{poly}(d))$ the FJLTr algorithm outperforms the FJLT algorithm. Table 3.4 summarizes this chapter's result.

	Naïve or Slower	Faster than naïve	$O(d\log(k))$
k in $\omega(\log d)$ and $o(\text{poly}(d))$	JL	FJLT	FJLTr
k in $\Omega(\text{poly}(d))$ and $o(d\log(d)^{1/3})$	JL		FJLT, FJLTr
k in $\omega((d\log d)^{1/3})$ and $o(d^{1/2})$	JL	FJLT, FJLTr	
k in $\Omega(d^{1/2})$ and $O(d)$	JL, FJLT, FJLTr		

Tab. 3.1: Results summary. Schematic comparison of asymptotic running time of three projection algorithms. (1) JL: a naïve implementation of the Johnson-Lindenstrauss lemma.(2) FJLT: the fast JL transform by Ailon Chazelle [1]. (3) FJLTr: the revised version of the FJLT algorithm described in this chapter.

4. TWO STAGE PROJECTION PROCESS

In this chapter we develop a new framework for generating random projection distributions. We view random projection as a concentration phenomenon of high dimensional random walks. This approach allows us to analyze new constructions for random projection distributions which could not be analyzed using existing methods. The distance from the origin, Y, achieved by d steps of a k dimensional random walk is $Y = \| \sum_{i=1}^{d} v_i s(i) \|_2$ where $s(i)$ are Rademacher variables (i.i.d. ± 1 w.p. $1/2$ each) and $v_i \in \mathbb{R}^k$. In one dimension ($k = 1$, and v_i are scalar) the concentration of Y is given by the Hoeffding bound and depends on the sum of squares of v_i (the ℓ_2 norm of a vector holding their values). In higher dimensions the concentration of Y depends on the spectral norm of a matrix M whose columns consist of the vectors v_i. To express random projections as random walks, consider the term $\Psi x = \sum_{i=1}^{d} \Psi^{(i)} x(i) = \sum_{i=1}^{d} A^{(i)} s(i) x(i)$ where $\Psi^{(i)} = A^{(i)} s(i)$ and $s(i)$ are Rademacher variables. Clearly, the projection norm $Y = \|\Psi x\|_2$ is equal to the final distance from the origin of a k dimensional random walk whose i'th step is equal to $A^{(i)} x(i)$. We study the concentration behavior of the random variable Y using a result by Ledoux and Talagrand [37]. Denoting by D_s a diagonal matrix such that $D_s(i,i) = s(i)$, if Y concentrates sufficiently well around the value $\|x\|_2$ we get that AD_s behaves like a good random projection for x. This concentration very much depends on both A and x. This chapter is dedicated to exploring the relation between the properties of A, x, and the concentration of Y. Namely, for a given *fixed* matrix A, we seek a set $\chi(A) \subset \mathbb{S}^d$ such that $x \in \chi(A)$ guaranties that AD_s is a good random projection for x.

4.1 Concentration result

Let A be a *fixed* $k \times d$ matrix and $x \in \mathbb{S}^d$ be a *fixed* vector. We say that A is a good projection for x if:

$$\Pr[|\|AD_s x\|_2 - 1| \geq \varepsilon] \leq 1/n \tag{4.1}$$

for given constants n and $0 < \varepsilon < 1/2$. D_s denotes a diagonal matrix whose entries are i.i.d. ± 1 with probability $1/2$ each. Note that the randomness is only in the d random bits required to generate D_s. If also

$$\Pr[|\|AD_s x\|_2 - 1| \geq \varepsilon] \leq c_1 e^{-c_2 k \varepsilon^2} \tag{4.2}$$

We say that AD_s is JL with respect to x.

Theorem 4.1.1 (Ledoux, Talagrand [37] page 19)**.** *Let f be a convex function $[0,1]^d \to \mathbb{R}$. Let $\|f\|_{Lip}$ denote its Lipschitz constant $\|f\|_{Lip} = \sup_{z_1, z_2 \in [0,1]^d} \{|f(z_1) - f(z_2)|/\|z_1 - z_2\|\}$. Let μ denote a median of f for a product space distribution $\mathbb{D}$, i.e. $\Pr_{z \sim \mathbb{D}}(f(z) \geq \mu) \geq 1/2$ and $\Pr_{z \sim \mathbb{D}}(f(z) \leq \mu) \geq 1/2$.*

$$\Pr_{x \sim \mathbb{D}}(|f(x) - \mu| > t) \leq 4e^{-t^2/8\|f\|_{Lip}^2} \tag{4.3}$$

Express Y by $AD_x s$ where D_x is a diagonal matrix holding on its diagonal the values of x, i.e. $D_x(i,i) = x(i)$ and s is a vector of random i.i.d. ± 1. The convex function over the solid cube in $\mathbb{R}^d$ used is $f(s) = \|Ms\| = Y$ where $M = AD_x$. The function f is convex on the relax domain $s \in [-1,1]^d$. Moreover, $\|f\|_{Lip} = \|M\|_{2 \to 2}$ by definition of the spectral norm.

Lemma 4.1.1 (Variation on Ledoux-Talagrand)**.** *For a matrix M and a random ± 1 vector s. Define the random variable $Y = \|Ms\|_2$. Denote by μ the median of Y and $\sigma = \|M\|_{2 \to 2}$ the spectral norm of M, then:*

$$\Pr[|Y - \mu| > t] < 4e^{-t^2/8\sigma^2} \tag{4.4}$$

Remark 4.1.1. *Notice that the lemma also holds for $Y = \|Ms\|_p$ for any norm index $p \geq 1$ if one defines $\sigma = \|M\|_{2 \to p} = \sup_{x \in \mathbb{R}^d, \|x\|_2 = 1} \|Mx\|_p$.*

The lemma asserts that $\|AD_s x\|$ distributes like a (sub) Gaussian around its median, with standard

deviation 2σ. Let us first compute the expected value of $Y^2 = \|AD_s x\|^2$ by expanding the square term.

$$E(Y^2) = E(\|AD_s x\|_2^2) = E\left(\sum_{i,j=1}^{d} \langle A^{(i)}, A^{(j)} \rangle x(i)x(j)s(i)s(j)\right) \tag{4.5}$$

$$= \sum_{i=1}^{d} \|A^{(i)}\|^2 x^2(i) = \|x\|^2 = 1 \tag{4.6}$$

The last equation holds if the columns of A are normalized such that $E[\|A^{(i)}\|_2^2] = 1$. From this point on we shall assume that this is the case. To estimate the median, μ, we substitute $t^2 \to t'$ and compute:

$$E[(Y-\mu)^2] = \int_0^\infty \Pr[(Y-\mu)^2] > t']dt'$$
$$\leq \int_0^\infty 4e^{-t'/(8\sigma^2)}dt' = 32\sigma^2$$

Furthermore, $(E[Y])^2 \leq E[Y^2] = 1$, and so $E[(Y-\mu)^2] = E[Y^2] - 2\mu E[Y] + \mu^2 \geq 1 - 2\mu + \mu^2 = (1-\mu)^2$. Combining, $|1-\mu| \leq \sqrt{32}\sigma$. We set $\varepsilon = t + |1-\mu|$:

$$\Pr[|Y-1| > \varepsilon] \leq 4e^{-\varepsilon^2/32\sigma^2} \text{ , for } \varepsilon > 2|1-\mu| \tag{4.7}$$

If we set $k = 33\log(n)/\varepsilon^2$ (for $\log(n)$ larger than a sufficient constant) and set $\sigma \leq k^{-1/2}$, Equation (4.7) meets the requirements of the JL property defined in (4.2). Moreover, since $|1-\mu| \leq \sqrt{32}\sigma$ the condition $\varepsilon > 2|1-\mu|$ is met for any constant ε. We see that $\sigma = \|AD_x\|_{2\to 2} \leq k^{-1/2}$ is a sufficient condition for the JL property to hold.

4.2 $\chi(A)$ the probabilistic Image

As discussed above the value of the term $\sigma = \|AD_x\|$ plays a crucial role in the random projection concentration phenomenon. We formally define $\|AD_x\|_{2\to 2}$ as $\|x\|_A$ below.

Definition 4.2.1. *For a given matrix $A \in \mathbb{R}^{k\times d}$ we define the vector seminorm of $x \in \mathbb{R}^d$ with respect to A as $\|x\|_A \equiv \|AD_x\|_{2\to 2}$ where D_x is a diagonal matrix such that $D_x(i,i) = x(i)$.*

Claim 4.2.1. *Let A be a constant matrix such that no column of A has zero norm. $\|x\|_A$ induces a proper norm on $\mathbb{R}^d$.*

Proof. • Scalability: For any x and a constant c, $\|cx\|_A = |c| \cdot \|x\|_A$.

- Positive definiteness: For any $x > 0$ and a matrix A with no zero columns, $\|x\|_A > 0$. To show this we multiply AD_x from the left by a test vector

$$\|AD_x\|_{2\to 2}^2 \geq \|y^T AD_x\|_2^2 = \sum_{i=1}^{d} \langle y^T, A^{(i)} \rangle^2 x^2(i) \tag{4.8}$$

  The last sum is larger than zero if y is set to $A^{(i)}$ for i such that $|x(i)| > 0$.

- Triangle inequality: For any x_1 and x_2 we have:

$$\|x_1 + x_2\|_A = \|AD_{x_1} + AD_{x_2}\|_{2\to 2} \leq \|AD_{x_1}\|_{2\to 2} + \|AD_{x_2}\|_{2\to 2} = \|x_1\|_A + \|x_2\|_A \tag{4.9}$$

□

In the case of dense random projections, we consider only matrices A for which all column norms are equal to 1, and thus $\|x\|_A$ induces a proper norm on $\mathbb{R}^d$. Also, recall from the previous section that a sufficient condition on x is that $\|AD_x\| = \|x\|_A \leq O(k^{-1/2})$. This gives a concise geometric description of the set of vectors for which A is a good random projection. This set, denoted by χ, is the intersection of the Euclidian unit sphere and a ball of radius $k^{-1/2}$ in A-norm.

Definition 4.2.2. *Let A be a column-normalized matrix. Let n and $0 < \varepsilon < 1/2$ be constants and let $k = 33\log(n)/\varepsilon^2$.*

$$\chi(A, \varepsilon, n) \equiv \mathbb{S}^{d-1} \bigcap \left\{ x \mid \|x\|_A \leq k^{-1/2} \right\}. \tag{4.10}$$

Lemma 4.2.1. *For any column-normalized matrix A and χ as in definition 4.2.2 the following holds:*

$$\forall x \in \chi(A, \varepsilon, n) \ \Pr\left[\left| \|AD_s x\|^2 - 1 \right| \geq \varepsilon) \right] \leq 1/n. \tag{4.11}$$

Proof. To see this we substitute $\|x\|_A^2 = \sigma^2 \leq 1/k$ into Equation (4.7). □

4.3 ℓ_p bounds on A-norms

We turn to bound $\|x\|_A$ for a given A and x using more manageable terms. We use $\|x\|_p$ to denote the p-norm of x, $\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{1/p}$ where $1 \leq p < \infty$ and $\|x\|_\infty = \max_{i=1}^{d} |x_i|$. The dual norm index q

is defined by the solution to $1/q + 1/p = 1$. We remind the reader that $\|x\|_p = \sup_{y,\|y\|_q=1} x^T y$. For a real $k \times d$ matrix A, the matrix norm $\|A\|_{p_1 \to p_2}$ is defined as the operator norm of $A : (\mathbb{R}^d, \ell_{p_1}) \to (\mathbb{R}^k, \ell_{p_2})$

$$\|A\|_{p_1 \to p_2} = \sup_{x \in \mathbb{R}^d, \|x\|_{p_1}=1} \|Ax\|_{p_2} \tag{4.12}$$

Lemma 4.3.1. *For any dual norm indices p and q*

$$\|x\|_A \leq \|x\|_{2p} \|A^T\|_{2 \to 2q} \tag{4.13}$$

Proof. We multiply the matrix AD_x from the left by a test vector $y \in \mathbb{R}^k$.

$$\begin{aligned}
\|x\|_A^2 &= \|AD_x\|_{2\to 2}^2 = \max_{y,\|y\|_2=1} \|y^T AD_x\|_2^2 && (4.14)\\
&= \max_{y,\|y\|_2=1} \sum_{i=1}^{d} x^2(i)(y^T A^{(i)})^2 && (4.15)\\
&\leq \left(\sum_{i=1}^{d} x^{2p}(i) \right)^{1/p} \left(\max_{y,\|y\|_2=1} \sum_{i=1}^{d} (y^T A^{(i)})^{2q} \right)^{1/q} && (4.16)\\
&= \|x\|_{2p}^2 \|A^T\|_{2\to 2q}^2 && (4.17)
\end{aligned}$$

The transition from the second to the third line follows from Hölder's inequality which states that for vectors $z_1, z_2 \in \mathbb{R}^d$, $\sum_{i=1}^{d} z_1(i) z_2(i) \leq \|z_1\|_p \|z_2\|_q$ for dual norms p and q. □

4.4 Conclusion

This chapter gave two results which together provide the skeleton on which we build from this point on. First, a matrix A can be used to project any vector x provided $\|x\|_A \leq k^{-1/2}$. Second, $\|x\|_A \leq \|x\|_{2p} \|A^T\|_{2\to 2q}$ for any dual norms p and q. This gives a convenient relation between A and $\chi(A)$.

Theorem 4.4.1. *For any column-normalized matrix A, $p > 1$ and $1/p + 1/q = 1$,*

$$\{x \in \mathbb{S}^{d-1} | \|x\|_{2p} \|A^T\|_{2\to 2q} \leq k^{-1/2}\} \subset \chi(A). \tag{4.18}$$

Our framework is thus as follows: Choose a column-normalized matrix $A \in \mathbb{R}^{k \times d}$ and a norm index q. Compute $\|A^T\|_{2\to 2q}$ and set $\eta = k^{-1/2}/\|A^T\|_{2\to 2q}$. Devise a randomized isometry Φ such that $\|\Phi x\|_{2p} \leq \eta$

w.h.p. Given the results of this chapter, such a construction guaranties that $\Psi = AD_s\Phi$ exhibits the JL property.

Note that A might consist of more than k rows which might help in reducing its operator norm. In this case $AD_s\Phi$ might still be a good random projection matrix but it does not necessarily exhibit the JL property. This is the case, for example, in Chapter 6.

5. FOUR-WISE INDEPENDENCE AND RANDOM PROJECTIONS

In Chapter 3 we showed that the running time of the FJLT algorithm can be reduced to $O(d\log(k)+k^3)$ by using efficient partial fast transforms. This, however, is dominated by $O(k^3)$ when $k \in \Omega((d\log(d))^{1/3})$. As claimed in the introductory chapter and in [34], the k^3 term is unavoidable if one uses sparse random i.i.d. projection matrices. The goal of this chapter is to show that fixed dense projections can be used to improve the running time to $O(d\log k)$ up to $k \in O(d^{1/2-\delta})$ for any positive constant δ.

We use the two stage projection scheme developed in Chapter 4, first reminding the reader of Lemmas 4.2.1 and 4.3.1. Let B be a $k \times d$ column-normalized matrix. If $\|x\|_B = O(k^{-1/2})$ then the matrix BD_s exhibits the JL property with respect to x, where D_s is a diagonal random i.i.d. ± 1 matrix. Second, $\|x\|_B \leq \|B^T\|_{2\to 4}\|x\|_4$.

The organization is as follows. First we show that a 4-wise independent, $k \times d$, matrix B exhibits $\|B^T\|_{2\to 4} \in O(k^{-1/2}d^{1/4})$ and can be applied in $O(d\log(k))$ operations. We further devise a mapping Φ which rotates any $x \in \mathbb{S}^{d-1}$ isometrically such that $\|\Phi x\|_4 \in O(d^{-1/4})$ with high probability. Applying Φ requires $O(d\log d)$ operation which is also $O(d\log(k))$ for all d polynomial in k. Thus, the concatenation $B\Phi$ exhibits the JLproperty and is applicable in $O(d\log(d))$ operations to any vector. This construction is referred to as FWI due to the *four-wise interdependence* of B.

Unfortunately, our construction for B is only possible for $k \in O(d^{1/2})$. Moreover, the mapping Φ only succeeds with high probability when $k \in O(d^{1/2-\delta})$ for a constant positive δ. We thus improve on the FJLT algorithm for values of k such that $k \in O(d^{1/2-\delta})$ and $k \in \omega((d\log(d))^{1/3})$. This chapter's main contribution is given formally in Theorem 5.0.2.

Theorem 5.0.2. *Let $\delta > 0$ be some arbitrarily small constant. For any d, k satisfying $k \leq d^{1/2-\delta}$ there*

exists an algorithm constructing a random matrix Ψ of size $k \times d$ satisfying JLP, such that the time to compute $x \mapsto \Psi x$ for any $x \in \mathbb{R}^d$ is $O(d \log k)$. The construction uses $O(d)$ random bits and applies to both the Euclidean and the Manhattan cases.

5.1 Preliminaries

Definition 5.1.1. *A matrix $\widetilde{B} \in \mathbb{R}^{k \times m}$ is a* code *matrix if every row of $\widetilde{B}$ is equal to some row of H_m multiplied by $\sqrt{m/k}$. The normalization is chosen so that columns have Euclidean norm 1.*

Definition 5.1.2. *A code matrix $\widetilde{B}$ of size $k \times m$ is* a-wise independent *if, for each $1 \leq i_1 < i_2 < \ldots < i_a \leq m$ and $(b_1, b_2, \ldots, b_a) \in \{+1, -1\}^a$, the number of columns $\widetilde{B}^{(j)}$ for which $(\widetilde{B}^{(j)}_{i_1}, \widetilde{B}^{(j)}_{i_2}, \ldots, \widetilde{B}^{(j)}_{i_a}) = k^{-1/2}(b_1, b_2, \ldots, b_a)$ is exactly $m/2^a$.*

Let $\widetilde{B}$ be a code matrix, as defined above. The columns of $\widetilde{B}$ can be viewed as vectors over $\mathbb{F}_2$ under the transformation $((+) \to 0, (-) \to 1)$. Clearly, the set of vectors thus obtained are closed under addition, and hence constitute a linear subspace of $\mathbb{F}_2^k$. Conversely, any linear subspace V of $\mathbb{F}_2^k$ of dimension ν can be encoded as a $k \times 2^\nu$ code matrix (by choosing some ordered basis of V). We will borrow well known constructions of subspaces from coding theory for constructing a four-wise independent code matrix.

Lemma 5.1.1. *There exists a 4-wise independent code matrix of size $k \times m$, where $m = \Theta(k^2)$.*

One such family of matrices is known as binary dual BCH codes of designed distance 5. Details of the construction can be found in [38].

The following is known as an interpolation theorem in the theory of Banach spaces. For a proof, the reader is referred to [39].

Theorem 5.1.1. [Riesz-Thorin] *Let A be a matrix such that $\|A\|_{p_1 \to r_1} \leq C_1$ and $\|A\|_{p_2 \to r_2} \leq C_2$ for some norm indices p_1, r_1, p_2, r_2. Let λ be a real number in the interval $[0, 1]$, and let p, r be such that $1/p = \lambda(1/p_1) + (1 - \lambda)(1/p_2)$ and $1/r = \lambda(1/r_1) + (1 - \lambda)(1/r_2)$. Then $\|A\|_{p \to r} \leq C_1^\lambda C_2^{1-\lambda}$.*

Theorem 5.1.2. **[Hausdorff-Young]** *Let H denote either a Hadamard or a Fourier operator. For norm index $1 \leq p \leq 2$, $\|H\|_{p\to q} \leq d^{-1/p+1/2}$, where q is the dual norm index of p.*[1]

5.2 Bounding $\|B^T\|_{2\to 4}$ using four-wise independence

Since $\|B^T\|_{2\to 4}^4$ is a degree 4 polynomial in the entries of B, intuitively, it values for a 4-wise independent matrix B should be similar to that expected for a random ± 1 matrix. Lemma 5.2.1 makes this intuition exact.

Lemma 5.2.1. *Assume B is a $k \times d$ 4-wise independent matrix. Then $\|B^T\|_{2\to 4} \leq (3d)^{1/4}k^{-1/2}$.*

Proof. For $y \in \mathbb{R}^k$, $\|y\|_2 = 1$,

$$\begin{aligned}
\|y^T B\|_4^4 &= dE_{j\in[d]}[(y^T B^{(j)})^4] \\
&= dk^{-2} \sum_{i_1,i_2,i_3,i_4=1}^{k} E_{b_{i_1},b_{i_2},b_{i_3},b_{i_4}}[y_{i_1}y_{i_2}y_{i_3}y_{i_4}b_{i_1}b_{i_2}b_{i_3}b_{i_4}] \qquad (5.1) \\
&= dk^{-2}(3\|y\|_2^4 - 2\|y\|_4^4) \leq 3dk^{-2} \ ,
\end{aligned}$$

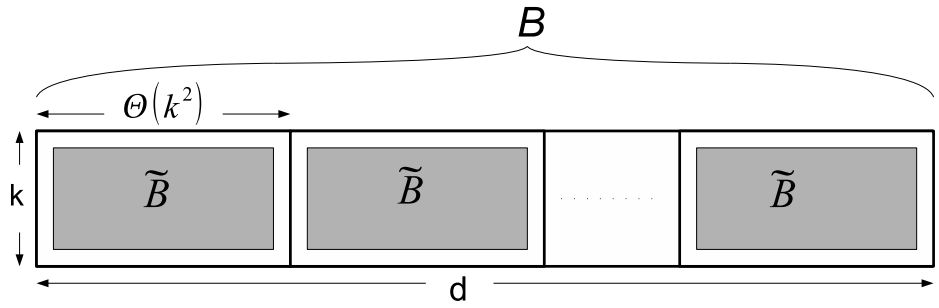
□

where b_{i_1} through b_{i_k} are independent random $\{+1, -1\}$ variables. Let $\widetilde{B}$ denote the $k \times m$ matrix from Lemma 5.1.1 (we assume here that $k = 2^a - 1$ for some integer a; This is harmless because otherwise we can reduce onto some $k' = 2^a - 1$ such that $k/2 \leq k' \leq k$ and pad the output with $k - k'$ zeros). In order to construct a matrix B of size $k \times d$ for $k \leq d^{1/2-\delta}$, we first make sure that d is divisible by m (by at most multiplying d by a constant factor and padding with zeros), and then take B to be d/m copies of $\widetilde{B}$ side by side. Clearly B remains 4-wise independent. Note that B may no longer be a code matrix, but $x \mapsto Bx$ is still computable in time $O(d \log k)$ by performing d/m Walsh transforms on blocks of size m.

5.3 Controlling $\|x\|_4$ for $k < d^{1/2-\delta}$

We define a randomized orthogonal transformation Φ such that for any vector $x \in \mathbb{S}^{d-1}$, $\|\Phi x\|_4 = O(d^{-1/4})$, with probability at least $1 - O(e^{-k})$ for all $k < d^{1/2-\delta}$. (Note: Both big-O's hide factors exponential in

[1] This can be easily obtained by using Riesz-Thorin and the fact that $\|H\|_{2\to 2} = 1$ and $\|H\|_{1\to\infty} = d^{-1/2}$.



δ). The rotation Φ is applicable to any $x \in \mathbb{S}^{d-1}$ in $O(d \log d)$ operations. The basic building block is the product HD', where $H = H_d$ is the Walsh-Hadamard matrix and D' is a diagonal matrix with random i.i.d. uniform ± 1 on the diagonal. Note that this random transformation was the main ingredient in [1]. Let $H^{(i)}$ denote the i'th column of H.

We are interested in the random variable $X = \|HD'x\|_4$. We define M as the $d \times d$ matrix with the i'th column $M^{(i)}$ being $x_i H^{(i)}$, we let $p = 4$ ($q = 4/3$), and notice that X is the norm of the Rademacher random variable in $(\mathbb{R}^d, \ell_4)$ corresponding to M (using the notation of Chapter 4 and specifically Lemma 4.1.1). We compute the deviation σ,

$$\begin{aligned}
\sigma &= \|M\|_{2\to 4} = \|M^T\|_{4/3\to 2} \\
&= \sup_{\substack{y\in \ell_{4/3}^k \\ \|y\|_{4/3}=1}} \left(\sum_i x_i^2 (y^T H^{(i)})^2 \right)^{1/2} \\
&\le \left(\sum x_i^4 \right)^{1/4} \sup \left(\sum_i (y^T H^{(i)})^4 \right)^{1/4} \\
&= \|x\|_4 \|H^T\|_{\frac{4}{3}\to 4} \ .
\end{aligned} \tag{5.2}$$

(Note that $H^T = H$.) By the Hausdorff-Young theorem, $\|H\|_{\frac{4}{3}\to 4} \le d^{-1/4}$. Hence, $\sigma \le \|x\|_4 d^{-1/4}$. We now get by Theorem 4.1.1 that for all $t \ge 0$,

$$\Pr[|\|HD'x\|_4 - \mu| > t] \le 4e^{-t^2/(8\|x\|_4^2 d^{-1/2})} \ , \tag{5.3}$$

where μ is the median of X.

Claim 5.3.1. $\mu = O(d^{-1/4})$.

Proof. To see the claim, notice that for each separate coordinate, $E[(HD'x)_i^4] = O(d^{-2})$ and then use linearity of expectation to get $E[\|HD'x\|_4^4] = O(d^{-1})$. By Jensen's inequality, $E[\|HD'x\|_4^b] \leq E[\|HD'x\|_4^4]^{b/4} = O(d^{-b/4})$ for $b = 1, 2, 3$. Now

$$\begin{aligned} E[(\|HD'x\|_4 - \mu)^4] &= \int_0^\infty \Pr[(\|HD'x\|_4 - \mu)^4 > s]ds \\ &\leq \int_0^\infty 4e^{-s^{1/2}/(8\|x\|_4^2 d^{-1/2})}ds \\ &= O(d^{-1}) \ . \end{aligned}$$

This implies by opening the left hand side expression that $-\gamma_1 d^{-3/4}\mu - \gamma_2 d^{-1/4}\mu^3 + \mu^4 \leq \gamma_3 d^{-1}$, where $\gamma_i > 0$ are global constants for $i = 1, 2, 3$. The statement of the claim immediately follows. □

Let c be such that $\mu \leq cd^{-1/4}$. We weaken inequality (5.3) using the last claim to obtain the following convenient form:

$$\Pr[\|HD'x\|_4 > cd^{-1/4} + t] \leq 4e^{-t^2/(8\|x\|_4^2 d^{-1/2})} \ . \tag{5.4}$$

In order to get a desired failure probability of $O(e^{-k})$ set $t = c'k^{1/2}\|x\|_4 d^{-1/4}$. For $k < d^{1/2-\delta}$ this gives $t < c'd^{-\delta/2}\|x\|_4$. In other words, with probability $1 - O(e^{-k})$ we get

$$\|HD'x\|_4 \leq cd^{-1/4} + c'd^{-\delta/2}\|x\|_4 \tag{5.5}$$

Now compose this r times: Take independent random diagonal $\{\pm 1\}$ matrices $D_1, \ldots, D_r$ and define $\Phi^r = HD_rHD_{r-1}\cdots HD_1$. Using a union bound on the conditional failure probabilities, we easily get:

Lemma 5.3.1. [ℓ_4 **reduction for** $k < d^{1/2-\delta}$] *With probability* $1 - O(e^{-k})$

$$\|\Phi^r x\|_4 = O(d^{-1/4}) \tag{5.6}$$

for $r = \lceil 1/2\delta \rceil$.

Note that the constant hiding in the bound (5.6) is exponential in $1/\delta$.

Combining the above, the random transformation $\Psi = BD\Phi^{(r)}$ exhibits the JL property for $k < d^{1/2-\delta}$, and can be applied to any vector in time $O(d \log d)$. This proves the Euclidean case of Theorem 5.0.2.

5.4 Reducing to Manhattan Space for $k < d^{1/2-\delta}$

As we did for the Euclidean case, we start by studying the random variable W defined as $W = \|k^{1/2}BDx\|_1$ for B as described in Section 5.2 and D a random ± 1 diagonal matrix. In order to characterize the concentration of W we compute the deviation σ, and estimate the median μ. According to Theorem 4.1.1, M is set to be the $k \times d$ matrix $k^{1/2}BD_x$.

$$\begin{aligned}
\sigma &= \|M\|_{2\to 1} = \|M^T\|_{\infty\to 2} \\
&= \sup_{y\in\mathbb{R}^k, \|y\|_\infty = 1} \|y^T M\|_2 = \sup\left(k\sum_{i=1}^{d} x_i^2 (y^T B^{(i)})^2\right)^{1/2} \\
&\leq \sup_{y\in\mathbb{R}^k, \|y\|_\infty = 1} k^{1/2}\|x\|_4 \|y^T B^{(i)}\|_4 = k^{1/2}\|x\|_4 \|B^T\|_{\infty\to 4}
\end{aligned} \tag{5.7}$$

Using the tools developed in the Euclidean case, we can reduce $\|x\|_4$ to $O(d^{-1/4})$ with probability $1 - O(e^{-k})$ using Φ^r, in time $O(d \log d)$. Also we already know from Section 5.2 that $\|B^T\|_{2\to 4} = O(d^{1/4}k^{-1/2})$. Since $\|y\|_\infty \geq k^{-1/2}\|y\|_2$ for any $y \in \mathbb{R}^k$, we conclude that $\|B^T\|_{\infty\to 4} = O(d^{1/4})$. Combining, we get $\sigma = O(k^{1/2})$. We now estimate the median μ of W.

In order to calculate μ we first calculate $E(W) = kE[|P|]$ where P is any single coordinate of $k^{1/2}BDx$. We follow (almost exactly) a proof by Matousek in [34] where he uses a quantitative version of the Central Limit Theorem by König, Schütt, and Tomczak [40].

Lemma 5.4.1. [König-Schütt-Tomczak] *Let $z_1 \dots z_d$ be independent symmetric random variables with $\sum_{i=1}^d E[z_i^2] = 1$, let $F(t) = \Pr[\sum_{i=1}^d z_i < t]$, and let $\overline{\varphi}(t) = \frac{1}{2\pi}\int_{-\infty}^{t} e^{-x^2/2}dx$. Then*

$$|F(t) - \overline{\varphi}(t)| \leq \frac{C}{1+|t|^3}\sum_{i=1}^{d} E[|z_i|^3]$$

for all $t \in \mathbb{R}$ and some constant C.

Clearly we can write $P = \sum_{i=1}^d z_i$ where $z_i = D_i' x_i$ and each D_i' is a random ± 1. Note that $\sum_{i=1}^d E[|z_i|^3] = \|x\|_3^3$.

Let β be the constant $\int_{-\infty}^{\infty} |t| d\overline{\varphi}(t)$ (the expectation of the absolute value of a Gaussian).

$$\begin{aligned} |E[|P|] - \beta| &= \left| \int_{-\infty}^{\infty} |t| dF(t) - \int_{-\infty}^{\infty} |t| d\overline{\varphi}(t) \right| \\ &\leq \int_{-\infty}^{\infty} |F(t) - \overline{\varphi}(t)| \, dt \\ &\leq \|x\|_3^3 \int_{-\infty}^{\infty} \frac{C}{1+|t|^3} dt \ . \end{aligned}$$

We claim that $\|x\|_3^3 = O(k^{-1})$. To see this, recall that $\|x\|_2 = 1$, $\|x\|_4 = O(d^{-1/4})$. Equivalently, $\|x^T\|_{2\to 2} = 1$ and $\|x^T\|_{4/3\to 2} = O(d^{-1/4})$. By applying Riesz-Thorin, we get that $\|x\|_3 = \|x^T\|_{3/2\to 2} = O(d^{-1/6})$, hence $\|x\|_3^3 = O(d^{-1/2})$. Since $k = O(d^{1/2})$ the claim is proved.

By linearity of expectation we get $E(W) = k\beta(1 \pm O(k^{-1}))$. We now bound the distance of the median from the expected value.

$$\begin{aligned} |E(W) - \mu| &\leq E[|W - \mu|] \\ &= \int_0^{\infty} \Pr[|W - \mu| > t] dt \\ &\leq \int_0^{\infty} 4e^{-t^2/(8\sigma^2)} dt = O(k^{1/2}) \end{aligned}$$

(we used our estimate $\sigma = O(k^{1/2})$ above.) We conclude that $\mu = k\beta(1 + O(k^{-1/2}))$. This clearly shows that (up to normalization) the random transformation $BD\Phi^{(r)}$ (where $r = \lceil 1/\delta \rceil$) has the JL property with respect to embedding into Manhattan space. The running time is $O(d \log d)$.

5.5 JL concatenation

The previous section produced a fast random projection from dimension d to k in $O(d \log(k))$ as long as $k \leq d^{1/2-\delta}$ for any constant positive δ. It, however, fails to exhibit the JL property for larger values of k. The only construction described thus far that exhibits the JL property for these values of k is the trivial one which requires $O(kd)$ operations to apply. This sudden increase in running time for a small increase in the target dimension k is unnatural. This section resolves this problem and allows to smoothly increase the running time as k grows. The idea described is a natural one. We claim that a concatenation of projections which independently exhibit the JL property also exhibits the JLP. This improves our performance by allowing us

to break the target dimension k into m sections of length k' each and apply m fast transforms from dimension d to dimension k'. The best results possible using the FWI construction gives a total projection running time of $O(kd^{1/2+\delta}\log(d))$ which asymptotically outperforms the naïve implementation for any k and d.

Lemma 5.5.1. *Let $\mathbb{D}_{k',d}$ be a distribution over $k' \times d$ matrices which exhibits the JL property. Define the distribution $\mathbb{D}^m_{k',d}$ to be a vertical concatenation of m matrices chosen independently from $\mathbb{D}_{k',d}$ normalized by $\frac{1}{\sqrt{m}}$. The distribution $\mathbb{D}^m_{k',d}$ over $k'm \times d$ matrices exhibits the JL property as well.*

Proof. Let A be a vertical concatenation of m, $k' \times d$, matrices $A_1, A_2, \ldots, A_m$ such that $A_1, A_2, \ldots, A_m$ are chosen i.i.d. from a distribution $\mathbb{D}_{k',d}$ which exhibits the JL property. Let $y_i = A_i x$, let $Y_i = \|y_i\|_2^2$ and let $Z_i = \sqrt{k'}(Y_i - 1)$. Since $\mathbb{D}_{k',d}$ is JL we have that

$$\begin{aligned} \Pr(Y_i > 1 + 3\varepsilon) &\leq \Pr(\|y_i\| > 1 + \varepsilon) && (5.8)\\ &< c_1 e^{c_2 k' \varepsilon^2} && (5.9)\\ \Pr(Z_i > u) &\leq c_1 e^{-c_2 u^2} \text{ for } u \leq 3\varepsilon\sqrt{k'}/2. && (5.10) \end{aligned}$$

Moreover we have that $E(Z_i) = 0$. Let us define the random variable $Y = \|Ax\|_2^2$. From the concatenation structure we have that $Y = \sum_{i=1}^m \frac{1}{m} Y_i$.

$$\begin{aligned} Y - 1 &= \sum_{i=1}^m \frac{1}{m}(Y_i - 1) && (5.11)\\ &= \frac{1}{\sqrt{k'm}} \sum_{i=1}^m \frac{1}{\sqrt{m}} \sqrt{k'}(Y_i - 1) && (5.12)\\ &= \frac{1}{\sqrt{k}} \sum_{i=1}^m \frac{1}{\sqrt{m}} Z_i && (5.13)\\ & && (5.14) \end{aligned}$$

Lemma 5.5.2. *(Matousek [34] (Lemma 2.2)) Let $Z_1, \ldots, Z_m$ be independent random variables, satisfying $E[Z_i] = 0$, $Var[Z_1] = 1$ and all having a uniform sub-Gaussian tail. Let $\alpha_1, \ldots, \alpha_m$ be such that $\sum_{i=1}^m \alpha_i^2 = 1$, the variable $Z = \sum_{i=1}^m \alpha_i Z_i$ has $E[Z] = 0$, $Var[Z] = 1$ and a sub-Gaussian tail.*

The following lemma holds also if $Var[Z_1]$ is equal to a constant other than 1. From Equation (5.8) we have that Z_i are sub-Gaussian, mean zero and constant variance. Therefore, the variable $Z = \sum_{i=1}^m \frac{1}{\sqrt{m}} Z_i$

also distributed like a sub-Gaussian. Finally we have that $\sqrt{k}(Y-1) = Z$ and thus $\Pr(\sqrt{k}(Y-1) > u) \leq c_1 e^{-c_2 u^2}$. Replacing $u = \sqrt{k}\varepsilon$ we get $\Pr(Y > 1+\varepsilon) \leq c_1 e^{-c_2 k\varepsilon^2}$ which is the JL property. □

Composing the result of Lemma 5.5.1 and the efficient FJLT construction gives us the best result possible for the cases where $k = \Omega(d^{1/2-\delta})$. We take k' to be $\Theta(d^{1/2-\delta})$ and perform $m = k/k' = \Theta(kd^{-1/2+\delta})$ independent transforms. Since each transform can be computed in $O(d\log(d))$ operations, the entire transform is computable in $O(kd^{1/2+\delta}\log(d))$. This outperforms the naive algorithm up to $k = \Theta(d)$ achieving a running time of $O(d^{3/2+\delta}\log(d))$.

5.6 Results summary

The current chapter described another approach in which a random projection can be obtained faster than the FJLT algorithm using four-wise independent matrices. This construction (FWI) permitted the acceleration to be useful for larger values of k than the FJLT algorithm permitted. The asymptotic running times of the improvements we achieved thus far are give in Table 5.6.

	Naïve or Slower	Faster than naïve	$O(d\log(k))$
k in $o(\log d)$	JL,FJLT,FWI		FJLTr
k in $\omega(\log d)$ and $o(\text{poly}(d))$	JL	FJLT, FWI	FJLTr
k in $\Omega(\text{poly}(d))$ and $o((d\log(d)^{1/3})$	JL		FJLT, FJLTr, FWI
k in $\omega((d\log d)^{1/3})$ and $O(d^{1/2-\delta})$	JL	FJLT, FJLTr	FWI
k in $O(d^{1/2-\delta})$ and $k < d$	JL,FJLT,FJLTr	JL concatenation	

Tab. 5.1: Result summary. Schematic comparison of asymptotic running time of five projection algorithms. (1) JL: a naïve implementation of the Johnson-Lindenstrauss lemma. (2) FJLT: the fast JL transform by Ailon Chazelle [1]. (3) FJLTr: a revised version of the FJLT algorithm, Chapter 3. (4) FWI: four-wise independent projection matrix. This chapter. (5) JL concatenation: a concatenation of several independent projections. This chapter.

6. TOWARDS LINEAR TIME DIMENSIONALITY REDUCTION

The trivial running time lower bound for randomly projecting vectors from dimension d to dimension k is $O(d)$. This is because, at the very least, each entry of the vector $x \in \mathbb{R}^d$ must be accessed. We ask ourselves whether achieving $O(d)$ running time is possible? Proving the contrary seems to be a hard problem. The application bottleneck for fast JL transforms is a Fourier transform. Thus, any running time lower bound greater than $O(d)$ will give the same lower bound for applying Fourier transforms. This, of course, is a longstanding open problem. Assuming that the lower bound of $O(d)$ can be achieved, we seek a mapping A which is applicable to any vector in $O(d)$ operations and constitutes a good random projection to the largest possible portion of $\mathbb{S}^{d-1}$, i.e. exhibits the largest possible set $\chi(A)$. Recall that $\chi(A)$ denoted the set of vectors for which A is a good random projection.

In section 6.1 we introduce a tensor product construction for dense orthogonal ± 1 matrices which we term *lean Walsh*. We show that they are applicable in $O(d)$ operations to any vector $x \in \mathbb{R}^d$. Section 6.2 gives an upper bound for operator norms of lean Walsh matrices. Using these bounds and Theorem 4.4.1 shows that they exhibit a strictly larger probabilistic domain $\chi(A)$ than sparse projections under comparable settings.

Due to their construction, lean Walsh matrices are of size $\widetilde{d} \times d$ where $\widetilde{d} = d^{\alpha}$ for some $0 < \alpha < 1$. In order to reduce the dimension to $k \leq \widetilde{d}$, $k = O(\log(n)/\varepsilon^2))$, we must compose our construction with another fast dimension reduction. This requires $O(\widetilde{d}\log(k)) \in O(d)$ operations and is explained in more detail in section 6.2.1.

6.1 Lean Walsh transforms

The *lean* Walsh Transform, similar to the Walsh Transform, is a recursive tensor product matrix. It is initialized by a constant seed matrix, A_1, and constructed recursively by using Kronecker products $A_{\ell'} = A_1 \otimes A_{\ell'-1}$. The main difference is that the lean Walsh seeds have fewer rows than columns. We formally define them as follows:

Definition 6.1.1. *A_1 is a lean Walsh seed (or simply 'seed') if i) A_1 is a rectangular matrix $A_1 \in \mathbb{C}^{r\times c}$, such that $r < c$; ii) A_1 is absolute valued $1/\sqrt{r}$ entry-wise, i.e., $|A_1(i,j)| = r^{-1/2}$; iii) the rows of A_1 are orthogonal; and iv) all inner products between its different columns are equal in absolute value to a constant $\rho \leq 1/\sqrt{(c-1)}$. ρ is called the Coherence of A_1.*

Definition 6.1.2. *A_ℓ is a lean Walsh transform, of order ℓ, if for all $\ell' \leq \ell$ we have $A'_\ell = A_1 \otimes A_{\ell'-1}$, where $\otimes$ stands for the Kronecker product and A_1 is a seed according to definition 6.1.1.*

The following are examples of seed matrices:

$$A'_1 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \qquad A''_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 \\ 1 & e^{2\pi i/3} & e^{4\pi i/3} \end{pmatrix} \tag{6.1}$$

$$r' = 3,\ c' = 4,\ \rho' = 1/3 \qquad\qquad r'' = 2,\ c'' = 3,\ \rho'' = 1/2$$

These examples are a part of a large family of possible seeds. This family includes, amongst other constructions, sub-Hadamard matrices (like A'_1) or sub-Fourier matrices (like A''_1). A simple construction is given for possible larger seeds.

Fact 6.1.1. *Let F be the $c \times c$ Discrete Fourier matrix such that $F(i,j) = e^{2\pi\sqrt{-1}ij/c}$. Define A_1 to be the matrix consisting of the first $r = c-1$ rows of F normalized by $1/\sqrt{r}$. A_1 is a lean Walsh seed with coherence $1/r$.*

Proof. The facts that $|A_1(i,j)| = 1/\sqrt{r}$ and that the rows of A_1 are orthogonal are trivial. Moreover, because $\sum_{i=1}^{c} \bar{F}(i,j_1)F(i,j_2) = 0$ (due to the orthogonality of the columns of F) we have that the inner product of two different columns of A_1 must equal $\rho = 1/r$ in absolute value.

$$\left|\langle A_1^{(j_1)}, A_1^{(j_2)}\rangle\right| = \frac{1}{r}\left|\sum_{i=1}^{r}\bar{F}(i,j_1)F(i,j_2)\right| = \frac{1}{r}\left|-\bar{F}(c,j_1)F(c,j_2)\right| = \frac{1}{r} \tag{6.2}$$

where $\bar{F}(\cdot,\cdot)$ stands for the complex conjugate of $F(\cdot,\cdot)$. □

We use elementary properties of Kronecker products to characterize A_ℓ in terms of the number of rows, r, the number of columns, c, and the coherence, ρ, of A_1. The following facts hold true for A_ℓ:

Fact 6.1.2. *i) A_ℓ is of size[1] $\widetilde{d}\times d$, where $\widetilde{d} = d^\alpha$ and $\alpha = \log(r)/\log(c) < 1$ is the skewness of A_1; ii) for all i and j, $A_\ell(i,j) \in \pm\widetilde{d}^{-1/2}$, which means that A_ℓ is column-normalized; and iii) the rows of A_ℓ are orthogonal.*

Fact 6.1.3. *The time complexity of applying A_ℓ to any vector $z \in \mathbb{R}^d$ is $O(d)$.*

Proof. Let $z = [z_1;\ldots;z_c]$ where z_i are sections of length d/c of the vector z. Using the recursive decomposition for A_ℓ we compute $A_\ell z$ by first summing over the different z_i according to the values of A_1 and applying to each sum the matrix $A_{\ell-1}$. Denoting by $T(d)$ the time to apply A_ℓ to $z \in \mathbb{R}^d$ we get that $T(d) = rT(d/c) + rd$. Due to the Master Theorem, and the fact that $r < c$ we have that $T(d) = O(d)$. More precisely, $T(d) \le dcr/(c-r)$. □

For clarity, we demonstrate Fact 6.1.3 for A_1' (Equation 6.1):

$$A_\ell' z = A_\ell' \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \frac{1}{\sqrt{3}} \begin{pmatrix} A_{\ell-1}'(z_1+z_2-z_3-z_4) \\ A_{\ell-1}'(z_1-z_2+z_3-z_4) \\ A_{\ell-1}'(z_1-z_2-z_3+z_4) \end{pmatrix} \tag{6.3}$$

In what follows we characterize $\chi(A,\varepsilon,n)$ for a general lean Walsh transform by the parameters of its seed, r and c. The omitted notation, A, stands for A_ℓ of the right size to be applied to x, i.e., $\ell = \log(d)/\log(c)$. Moreover, we freely use α to denote the skewness $\log(r)/\log(c)$ of the seed at hand.

[1] The size of A_ℓ is $r^\ell \times c^\ell$. Since the running time is linearly dependent on d, we can always pad vectors to be of length c^ℓ without effecting the asymptotic running time. From this point on we assume w.l.o.g. that $d = c^\ell$ for some integer ℓ.

6.2 Lean Walsh operator norms

After describing lean Walsh transforms we turn our attention to exploring their probabilistic domain χ .We remind the reader of Theorem 4.4.1 which claims that for any column-normalized matrix A and dual norms $p \geq 1$ and q

$$\{x \in \mathbb{S}^{d-1} | \|x\|_{2p} \|A^T\|_{2\to 2q} \leq k^{-1/2}\} \subset \chi(A) \tag{6.4}$$

To evaluate $\|A_\ell^T\|_{2\to 2q}$ we first bound the value of $\|A_1^T\|_{2\to 2q}$. Denote by $\overline{A}_1$ a matrix of size $c-r \times c$ which completes A to an orthogonal matrix. If, for example, A_1 is constructed by picking a set of r rows from a Hadamard transform H_c (normalized by $\sqrt{c/r}$), $\overline{A}_1$ will consist of all the rows in H_c not used by A_1 (also normalized).

$$\begin{aligned}
\|A_1^T\|_{2\to 2q} &= \max_{x\in\mathbb{R}^r,\, \|x\|_2=1} \|A_1^T x\|_{2q} && (6.5)\\
&\leq \max_{x\in\mathbb{R}^c,\, \|x\|_2=1} \| \begin{pmatrix} A_1^T & \overline{A}_1^T \end{pmatrix} x\|_{2q} && (6.6)\\
&= \max_{x\in\mathbb{R}^c,\, \|x\|_2=1} \sqrt{\frac{c}{r}} \|H_c x\|_{2q} && (6.7)\\
&= \sqrt{\frac{c}{r}} \|H\|_{2\to 2q} = \sqrt{\frac{c}{r}} && (6.8)
\end{aligned}$$

The notation $\begin{pmatrix} A_1^T & \overline{A}_1^T \end{pmatrix}$ stands for a horizontal concatenation of A_1^T and $\overline{A}_1^T$. The claim that $\|H\|_{2\to 2q} = 1$ for all $q \geq 1$ stems from the Riesz-Thorin theorem and the fact that $\|H\|_{2\to 2} = \|H\|_{2\to\infty} = 1$.

A useful property of Kronecker products is that for $q > 1$, $\|B \otimes C\|_{2\to 2q} = \|B\|_{2\to 2q} \|C\|_{2\to 2q}$ for any pair of matrices B and C. Therefore:

$$\|A_\ell^T\|_{2\to 2q} = \|A_1^T\|_{2\to 2q}^\ell < \left(\frac{c}{r}\right)^{\frac{1}{2}\log_c(d)} = d^{\frac{1}{2}(1-\alpha)} \tag{6.9}$$

where $\alpha = \log(r)/\log(c)$ is the skewness of the seed A_1.

Lemma 6.2.1. *For a lean Walsh transform, A, we have that for any $p > 1$ the following holds:*

$$\{x \in \mathbb{S}^d \mid \|x\|_{2p} \leq k^{-1/2} d^{-\frac{1-\alpha}{2}}\} \subset \chi(A, \varepsilon, n) \tag{6.10}$$

where $k = O(\log(n)/\varepsilon^2)$, $\alpha = \log(r)/\log(c)$, r is the number of rows, and c is the number of columns in the seed of A.

6.2.1 Controlling α and projecting to dimension k

We see that increasing α is beneficial from the theoretical stand point since it weakens the constraint on $\|x\|_{2p}$. However, the application oriented reader should keep in mind that this requires the use of a larger seed, which subsequently increases the constant hiding in the big O notation of the running time.

Consider the seed constructions described in Fact 6.1.1 for which $r = c - 1$. Their skewness $\alpha = \log(r)/\log(c)$ approaches 1 as their size increases. Namely, for any positive constant δ there exists a constant size seed such that $1 - 2\delta \leq \alpha < 1$.

Lemma 6.2.2. *For any positive constant $\delta > 0$ there exists a lean Walsh matrix, A, such that:*

$$\{x \in \mathbb{S}^d \mid \|x\|_\infty \leq k^{-1/2} d^{-\delta}\} \subset \chi(A, \varepsilon, n) \tag{6.11}$$

Proof. Generate A from a seed such that its skewness $\alpha = \log(r)/\log(c) \geq 1 - 2\delta$ and substitute $p = \infty$ into the statement of Lemma 6.2.1. □

The constant α also determines the minimal dimension d (relative to k) for which the projection can be completed in $O(d)$ operations, the reason being that the vectors $z = AD_s x$ must be mapped from dimension $\widetilde{d}$ ($\widetilde{d} = d^\alpha$) to dimension k in $O(d)$ operations. This can be done using the FWI construction from Chapter 5 ([2]) serving as a random projection from dimension $\widetilde{d}$ to dimension k. The FWI construction can be applied in $O(\widetilde{d}\log(k))$ operations as long as $\widetilde{d} = d^\alpha \geq k^{2+\delta''}$ for an arbitrary small δ''. For the same choice of a seed as in lemma 6.2.2, the condition becomes $d \geq k^{2+\delta''+2\delta}$ which can be achieved for $d \geq k^{2+\delta'}$ for arbitrary small δ'. Therefore, for $d\omega(k^{2+\delta'})$ one can complete the projection onto dimension k in $O(d^\alpha \log(k)) = O(d)$ operations.

6.3 Comparison to sparse projections

Sparse random ± 1 projection matrices where analyzed by Matousek in [34]. For completeness we restate his result. Theorem 4.1 in [34] (slightly rephrased) claims the following:

Theorem 6.3.1 (Matousek 2006 [34])**.** *let $\varepsilon \in (0, 1/2)$ and $\eta \in [1/\sqrt{d}, 1]$ be constant parameters. Set*

$q = C_0\eta^2 \log(n)$ for a sufficiently large constant C_0. Let S be a random variable such that

$$S = \begin{cases} +\frac{1}{\sqrt{q}} & \text{with probability } q/2 \\ -\frac{1}{\sqrt{q}} & \text{with probability } q/2 \\ 0 & \text{with probability } 1-q \end{cases} \tag{6.12}$$

Let k be $C_1 \log(n)/\varepsilon^2$ for a sufficiently large C_1. Let the matrix $A \in \{-\frac{1}{\sqrt{q}}, 0, +\frac{1}{\sqrt{q}}\}^{k\times d}$ contain i.i.d. copies of S then

$$\Pr[|\|Ax\|_2^2 - 1| > \varepsilon] \leq 1/n \tag{6.13}$$

for any $x \in \mathbb{S}^{d-1}$ such that $\|x\|_\infty \leq \eta$.

In the terminology of this thesis we say that for such distributions $\{x \in \mathbb{S}^{d-1} | \|x\|_\infty \leq \eta\} \subset \chi(A, \varepsilon, n)$.

With high probability, a matrix A chosen according the above distribution contains $O(kdq) = O(k^2 d\eta^2)$ non-zeros (since ε is a constant $\log(n) = O(k)$). Notice that for a linear application time we must have $O(k^2 d\eta^2) = O(d)$ which requires $\|x\|_\infty \leq \eta = k^{-1}$. For any d polynomial in k this is a stronger requirement than that posed by using lean Walsh matrices, i.e. $\|x\|_\infty \leq k^{-1/2} d^{-\delta}$.

6.4 Dimensionality reduction by sums of coordinates

It is interesting to note that another simple set of matrices can be analyzed using our framework. Consider the $\widetilde{d} \times d$ matrix A such that $A(i,j) = 1$ if $i \cong j \mod \widetilde{d}$. This matrix is simply $d/\widetilde{d}$ copies of the identity matrix concatenated horizontally. A is clearly column-normalized, and applicable in $O(d)$ operations. It is trivial to verify that $\|A\|_{2\to 2} = (d/\widetilde{d})^{1/2}$. Using the same argument as above

$$\{x \in \mathbb{S}^{d-1} | \|x\|_\infty \leq k^{-1/2}(d/\widetilde{d})^{-1/2}\} \subset \chi(A, \varepsilon, n) \tag{6.14}$$

By choosing $\widetilde{d} = d/\log(k)$ we get the constraint $\|x\|_\infty \leq (k\log(k))^{-1/2}$. As above, one can project the resulting vector from dimension $\widetilde{d}$ to dimension k using the FWI construction in $O(\widetilde{d}\log(k)) = O(d)$ operation.

6.5 Conclusions

Theoretically, the ℓ_∞ requirement achieved by identity copies is strictly better (larger) than the one achieved by lean Walsh transforms. However, the experimental chapter shows that using lean Walsh transforms should be preferred in practice to using identity copies. I.e., one achieves more accurate projections for a similar ℓ_∞ bound. This hints at the fact that we should better understand the term $\|x\|_A$. Here, we only bound its value by $\|x\|_\infty \|A\|_{2\to 2}$. This is problematic, for example, because $\|x\|_A$ is not invariant to permuting the vector x whereas $\|x\|_\infty \|A\|_{2\to 2}$ is. In other words, it would seem that the set $\{x \in \mathbb{S}^d \mid \|x\|_A \leq \eta\}$ is, in fact, much larger than the set $\{x \in \mathbb{S}^d \mid \|x\|_\infty \|A\|_{2\to 2} \leq \eta\}$

In order to be able to perform general random projections in $O(d)$ operations, one must design a linear time preprocessing matrix Ψ which maps all vectors in $\mathbb{R}^d$ into χ (w.h.p). It is not known whether such transformations exist. Achieving such Ψ would be extremely interesting from both the theoretical and the practical standpoints. Possible choices for Ψ may include random permutations, various wavelet/wavelet-like transforms, or any other sparse orthogonal transformation.

	$k \times d$ projection matrix	Application complexity	$x \in \chi$ if $\|x\|_2 = 1$ and:
Johnson, Lindenstrauss [3]	Random k dimensional subspace	$O(kd)$	
Various Authors [32, 31, 33]	i.i.d. random entries	$O(kd)$	
Ailon, Chazelle [1]	Sparse i.i.d. Gaussian entries	$O(k^3)$	$\|x\|_\infty = O((d/k)^{-1/2})$
Matousek [34]	Sparse ± 1 entries	$O(k^2 d\eta^2)$	$\|x\|_\infty \leq \eta$
General rule	**Any matrix**		$\|x\|_A = O(k^{-1/2})$
This thesis [2]	4-wise independent matrix	$O(d \log k)$	$\|x\|_4 = O(d^{-1/4})$
This thesis [36]	Lean Walsh Transform	$O(d)$	$\|x\|_\infty = O(k^{-1/2} d^{-\delta})$
This thesis	Identity copies	$O(d)$	$\|x\|_\infty = O((k \log k)^{-1/2})$

Tab. 6.1: Different distributions for $k \times d$ matrices and the set $\chi \subset \mathbb{S}^{d-1}$ for which they constitute a good random projection. Fixed matrices (last four rows) are composed with a random ± 1 diagonal matrix, see Section 4.2. The meaning of $\|\cdot\|_A$ is given in Definition 4.2.1.

7. THE MAILMAN ALGORITHM: A NOTE ON MATRIX VECTOR MULTIPLICATION

A claim that has repeated throughout this thesis is that $O(mn)$ operations are required to apply a general $m \times n$ matrix to a vector $x \in \mathbb{R}^n$. This fact was proved by Winograd in [41]. However, we claim that if the matrix contains only a constant number of distinct values, then reading the matrix once in $O(mn)$ steps is sufficient to preprocess it such that any subsequent application to vectors requires only $O(mn/\log(max\{m,n\}))$ operations. Using this fact in conjunction with Achlioptas's $\{+1,-1\}$ distribution for random projections allows us to apply general random projections in $O(d)$ operations for the case where $k \in O(\log(d))$. Algorithms for matrix-vector multiplication over finite fields, which save a log factor, have been known for many years. Our contribution is unique in its simplicity and in the fact that it applies also to real valued vectors. The mailman algorithm is also shown to be useful (faster than naïve) even for small matrices.

7.1 Matrix-vector multiplication background

A classical result of Winograd ([41]) shows that general matrix-vector multiplication requires $\Omega(mn)$ operations. This matches the running time of the naïve algorithm. However, since matrix-vector multiplication is such a common, basic operation, an enormous amount of effort has been put into exploiting the special structure of matrices to accelerate it. For example, Fourier, Hadamard, Toeplitz, Vandermonde and others can be applied to vectors in $O(npolylog(n))$ operations.

Others have focussed on matrix-matrix multiplication. For two $n \times n$ binary matrices the historical *Four Russians Algorithm* [42] (modified in [43]) gives a log factor improvement over the naïve algorithm, i.e, running time of $n^3/log(n)$. Techniques for saving log factors in real valued matrix-matrix multiplications

were also found [44]. These methods are reported to be practically more efficient than naïve implementations. Classic positive results, achieving a polynomial speedup, by Strassen [45] and Coppersmith and Winograd [46] as well as lower bounds [47] for general matrix-matrix multiplications are known. These, however, do not extend to matrix-vector multiplication.

We return to matrix-vector operations. Since every entry of the matrix, A, must be accessed at least once, we consider a preprocessing stage. After the preprocessing stage x is given and we seek an algorithm to produce the product Ax as fast as possible. Within this framework Williams [48] showed that an $n \times n$ binary matrix can be preprocessed in time $O(n^{2+\epsilon})$ and subsequently applied to *binary vectors* in $O(n^2/\epsilon \log^2 n)$. Williams also extends his result to matrix operations over finite semirings. However, to the best of the authors' understanding, his technique cannot be extended to real vectors.

7.2 The Mailman algorithm

Intuitively, our algorithm multiplies A by x in a manner that brings to mind the way a mailman distributes letters, first sorting the letters by address and then delivering them. Recalling the identity $Ax = \sum_{i=1}^{n} A^{(i)}x(i)$, metaphorically one can think of each column $A^{(i)}$ as indicating one "address", and each entry $x(i)$ as a letter addressed to it. To continue the metaphor, imagine that computing and adding the term $A^{(i)}x(i)$ to the sum is equivalent to the effort of walking to house $A^{(i)}$ and delivering $x(i)$. From this perspective, the naive algorithm functions by delivering each letter individually to its address, regardless of how far the walk is or if other letters are going to the same address. Actual mailmen, of course, know much better. First, they arrange their letters according to the shortest route (which includes all houses) without moving; then they walk the route, visiting each house regardless of how many letters should be delivered to it (possibly none).

To extend this idea to matrix-vector multiplication, our algorithm decomposes A into two matrices, U and P, such that $A = UP$. P is the "address-letter" correspondence matrix. Applying P to x is analogous to arranging the letters. U is the matrix containing all possible columns in A. Applying U to (Px) is analogous to walking the route. Hence, we name our algorithm after the age-old wisdom of the men and women of the postal service.

For simplicity, we describe the algorithm for an $m \times n$ matrix A, where $m = \log_2(n)$ (w.l.o.g. assume $\log_2(n)$ is an integer) and $A(i,j) \in \{0,1\}$. Later we shall generalize it to include other matrix dimensions and possible entry values. There are precisely $2^m = n$ possible columns in the matrix A, by construction, since each of the m column entries can be only 0 or 1. Define the universal columns matrix, U_n, as the matrix containing *each* possible column $\{0,1\}^m$ once. By definition, for any column $A^{(j)}$ there exists exactly one index $1 \leq i \leq n$ such that $A^{(j)} = U_n^{(i)}$. Define the $n \times n$ correspondence matrix P as $P(i,j) = \delta(U^{(i)}, A^{(j)})$. Here δ stands for the Kronecker delta function. $\delta(U_n^{(i)}, A^{(j)}) = 1$ if $U_n^{(i)} = A^{(j)}$ and zero otherwise.

$$\begin{aligned}(U_nP)(i,j) &= \sum_{k=1}^{n} U_n(i,k)P(k,j) \\ &= \sum_{k=1}^{n} U_n^{(k)}(i)\delta(U_n^{(k)}, A^{(j)}) \\ &= A^{(j)}(i) = A(i,j)\end{aligned}$$

The Mailman algorithm simply uses the associativity of matrix multiplication to compute $Ax = (UP)x = U(Px)$. The fact that computing Px requires $O(n)$ operations is clear since P is a sparse matrix containing only n non-zeros. We later show that applying U_n to any vector also requires $O(n)$ operations. Thus, although A is of size $\log(n) \times n$, the product Ax can be computed via $U(Px)$ while performing only $O(n)$ operations. This gains a $\log(n)$ factor over the naïve application, requiring $O(n\log(n))$ operations. If the number of rows, m, in A is more than $\log(n)$ we partition A into at most $\lceil m/\log(n) \rceil$ submatrices each of size at most $\log(n) \times n$. Since each of the submatrices can be applied in $O(n)$ operations, the entire matrix can be applied in $O(mn/\log(n))$ operations.

We now describe how to construct the correspondence matrix P and how to apply the universal column matrix U_n efficiently.

7.2.1 Preprocessing: constructing the correspondence matrix

An entry in the correspondence matrix $P(i,j)$ is set to 1 if $A^{(j)} = U^{(i)}$ otherwise $P(i,j) = 0$. In the next section we construct U such that each column, i, encodes in binary the value $i-1$. Thus, if a column of $A^{(j)}$ contains the binary representation of the value $i-1$, it is equal to $U^{(i)}$. We therefore construct P by reading

A and setting $P(i,j)$ to 1 if $A^{(j)}$ represents the value $i-1$. This concludes the preprocessing stage and can clearly be achieved in $O(mn)$ steps. Notice that since U is fixed, P encodes all the infirmation about A.

7.2.2 Application: universal column matrices

Denote by U_n the $\log_2(n) \times n$ matrix, which contains all strings $\{0,1\}^{\log(n)}$ as columns. Also denote by $\mathbf{0}_n$ and $\mathbf{1}_n$ the all zeros and all ones vectors of length n. Notice immediately that U_n can be constructed recursively:

$$U_1 = \begin{pmatrix} 0 & 1 \end{pmatrix}, \quad U_n = \left(\begin{array}{c|c} \mathbf{0}_{n/2}^T & \mathbf{1}_{n/2}^T \\ \hdashline U_{n/2} & U_{n/2} \end{array}\right).$$

Applying U_n to any vector z requires less than $4n$ operations, which can be shown by dividing z into its first and second halves, z_1 and z_2, and computing the product $U_n z$ recursively.

$$U_n z = \left(\begin{array}{c|c} \mathbf{0}_{n/2}^T & \mathbf{1}_{n/2}^T \\ \hdashline U_{n/2} & U_{n/2} \end{array}\right) \left(\begin{array}{c} z_1 \\ \hdashline z_2 \end{array}\right) = \left(\begin{array}{c} \mathbf{0}_{n/2}^T z_1 + \mathbf{1}_{n/2}^T z_2 \\ \hdashline U_{n/2}(z_1 + z_2) \end{array}\right)$$

Computing the vector $z_1 + z_2$, and the sums $\mathbf{0}_{n/2}^T z_1$ and $\mathbf{1}_{n/2}^T z_2$ requires (at most) $2n$ operations. Denoting by $T(n)$ the number of operations required for applying U_n to a vector $x \in \mathbb{R}^n$, we get the recurrence relation:

$$T(2) = 2, \qquad T(n) = T(n/2) + 2n \qquad \Rightarrow \qquad T(n) \leq 4n.$$

Computing $\mathbf{0}_{n/2}^T z_1$ can of course be removed from the runtime analysis, but we keep it in anticipation of alphabets other than $\{0,1\}$.

7.2.3 Constant sized alphabets

A matrix A is said to be over an alphabet Σ if $\forall i,j \;\; A(i,j) \in \Sigma$. Such matrices can be decomposed similarly and applied to any vector in $O(mn \log(|\Sigma|)/\log(n))$ time. The definition of U_n is changed such that it encodes all possible strings over $\Sigma = \{\sigma_1, \ldots, \sigma_S\}$, $|\Sigma| = S$.

$$U_1 = (\sigma_1, \ldots, \sigma_S) \qquad U_n = \left(\begin{array}{c|c|c} \sigma_1 \mathbf{1}_{n/S}^T & \ldots & \sigma_S \mathbf{1}_{n/S}^T \\ \hdashline U_{n/S} & \ldots & U_{n/S} \end{array}\right)$$

The matrix U_n is of size $\log_S(n) \times n$ and it can be applied to $x \in \mathbb{R}^n$ in $O(n)$ time. The construction of P also changes. $P(i,j)$ is set to 1 iff $A^{(j)}$ represents the number $i-1$ in base S under the transformation $\sigma_1 \to 0, \ldots, \sigma_S \to S-1$. If the number of rows, m in A is larger than $\log_S(n)$, we divide A it into $\lceil m/\log_S(n) \rceil$ sections of size at most $\log_S(n) \times n$. The total running time therefore becomes $O(mn\log(S)/\log(n))$.

7.2.4 *Saving* a $\log(m)$ *factor*

If the number of rows, m, in A is larger than the number of columns, n, we can achieve an application running time of $O(mn/\log(m))$. Assume that A is of size $m \times n = \log(m)$. We have that $A = P^T U_m^T$, where P is the correspondence matrix for A^T as explained above. It is left to show that computing $U^T x$ for $x \in \mathbb{R}^{\log(m)}$ requires $O(m)$ operations. Denote by x_1 the first element of x and by $x_{2:n}$ the remaining $n-1$ elements.

$$U_m^T x = \left(\begin{array}{c|c} \mathbf{0}_{m/2} & U_{m/2}^T \\ \hdashline \mathbf{1}_{m/2} & U_{m/2}^T \end{array}\right) \left(\begin{array}{c} x_1 \\ \hdashline x_{2:n} \end{array}\right) = \left(\begin{array}{c} \mathbf{0}_{m/2} x_1 + U_{m/2}^T x_{2:n} \\ \\ \mathbf{1}_{m/2} x_1 + U_{m/2}^T x_{2:n} \end{array}\right)$$

Clearly, computing $U_m^T x$ requires computing $U_{m/2}^T x_{2:n}$ and an additional $O(m)$ operations which gives a total running time of $O(m)$. If the number of columns n of A is more than $\log(m)$, we partition A vertically to at most $\lceil n/\log m \rceil$ submatrices of size $m \times \log m$. Since each submatrix is applicable in $O(m)$ operations, the total running time of applying A to a vector $x \in \mathbb{R}^n$ is $O(mn/\log(m))$.

Remark 7.2.1 (Matrix operations over semirings)**.** *Suppose we supply* Σ *with an associative and commutative addition operation* $(+)$*, and a multiplication operation* $(\cdot)$ *that distributes over* $(+)$*. If both* A *and* x *are chosen from* Σ *the matrix vector multiplication over the semiring* $\{\Sigma, +, \cdot\}$ *can be performed using the same algorithm. This is, of course, not a new result. However, it includes all other* log*-factor-saving results.*

7.3 *Dimensionality reduction using the Mailman algorithm*

In the dimensionality reduction setup one is given n points $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$ and is asked to embed them into $\mathbb{R}^k$ such that all distances between points are preserved up to distortion ε and $k \ll d$. As seen throughout this manuscript, this can be achieved by multiplying each vector x_i by a $k \times d$ real matrix for

$k = \Theta(\log(n)/\varepsilon^2)$. The two previous chapters were dedicated to crafting different JL distributions for A such that it still admits a fast transform. We recall the result of Achlioptas.

Lemma 7.3.1 (Achlioptas [33]). *A $k \times d$ matrix A such that $A(i,j)$ are i.i.d and*

$$A(i,j) = \begin{cases} \frac{1}{\sqrt{k}} & w.p \quad 1/2 \\ -\frac{1}{\sqrt{k}} & w.p \quad 1/2 \end{cases} \tag{7.1}$$

exhibits the JL property.

Clearly a naive application of A to each vector requires $O(mn)$ operations. However, using the mailman algorithm to apply Achlioptas's matrix, one can apply A to each vector in $O(dk/log(d))$ operations. Moreover, $k = O(\log(n)/\varepsilon^2)$. For a constant ε and n polynomial in d, $k = O(\log(n)) = O(\log(d))$ and applying A to x_i requires only $O(n/\varepsilon^2) = O(n)$ operations. This result matches the lower bound for general dimensionality reduction. Notice that the running time dependence on ε is $1/\varepsilon^2$ instead of $\log(1/\varepsilon)$, as in [2], which makes this latter result actually much slower for most practical purposes.

Experiments

Here we compare the running time of applying a $log(n) \times n$, $\{0,1\}$ matrix A to a vector of floating point variables $x \in \mathbb{R}^n$ using three methods. The first is a *naïve* implementation in C: this simply consists of two nested loops ordered with respect to memory allocation to minimize cache faults. The second is an *optimized* matrix vector code: we test ourselves against LAPACK which uses BLAS subroutines [49, 50]. The third is, of course, the *mailman* algorithm, not including the $O(mn)$ preprocessing stage.

Although the complexity of the first two methods is $O(n\log(n))$ and that of the mailman algorithm is $O(n)$, we do not seem to get a $\log(n)$ speedup. This is because the memory access pattern in applying P is problematic with respect cache usage. The reader should bare in mind that these results might depend on memory vs. CPU specifications. The experiments were conducted on an Xeon Quad core 2.33GHz machine running Linux Ubuntu with 8G of RAM and a Bus speed of 1333MHz. Similar results were obtained using other, less powerful, machines as well. As seen in figure 7.1, the mailman algorithm operates faster than the

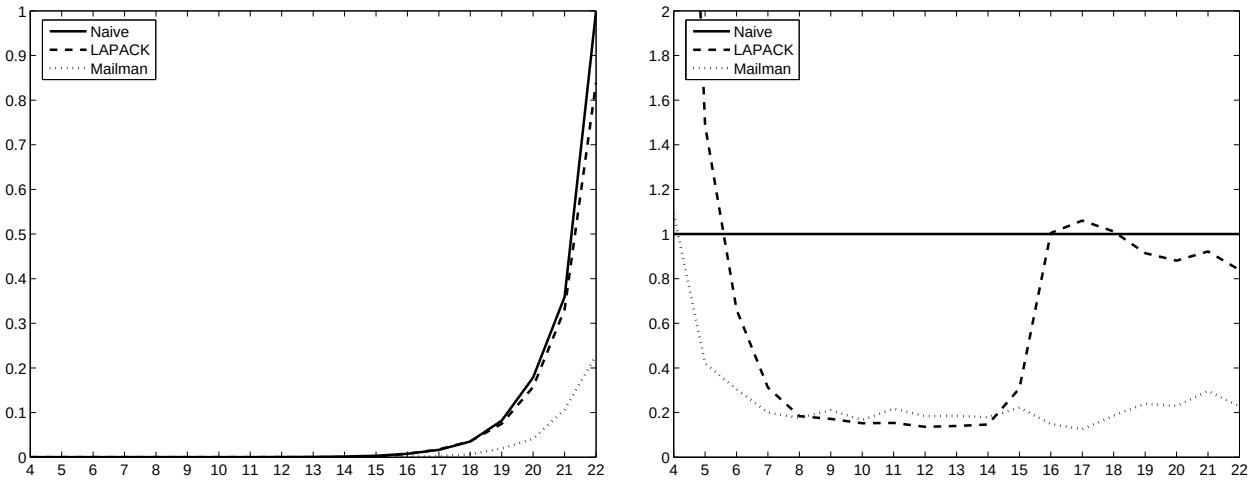


Fig. 7.1: Running time for multiplying an $m \times 2^m$ ± 1 matrix to a double precision vector. m is given on the x-axis. The y-axis gives the running time for achieving the multiplication using three different algorithms. (1) Naive: matrix multiplication implemented as two nested for-loops written in C and ordered correctly with respect to memory access. (2) LAPACK: a general purpose matrix multiplication implementation which is machine optimized. (3) Mailman: the mailman algorithm as described above. The left figure is a plot of the 3 running times on an absolute scale, the time scale is such that applying the maximal sized matrix naively requires 1 unit of time. The right figure plots the application time relative to the naive algorithm for each matrix size.

naïve algorithm even for 5×32 matrices. It also outperforms the LAPACK procedure for most matrix sizes. The machine specific optimized code (LAPACK) is superior when the matrix row allocation size approaches the memory page size. A machine specific optimized mailman algorithm might take advantage of the same phenomenon as well.

7.4 Concluding remark

It has been known for a long time that a log factor can be saved in matrix-vector multiplication when the matrix and the vector are over constant size alphabets. In this chapter we described an algorithm that achieves this while also dealing with real-valued vectors. As such, the idea by itself is neither revolutionary nor complicated. It is interesting to note though, that this simple idea gives the first random projection

algorithm which matches the lower bound for running time. Even if only in the case where $k \in O(\log(d))$. Table 7.4 summarizes the best known asymptotic rannind times for applying random projections updated with the results of this chapter.

	Naïve or Slower	Faster then naïve	$O(d \log(k))$	Optimal, $O(d)$
k in $o(\log d)$	JL, FJLT		FJLTr, FWI	JL + Mailman
k in $\omega(\log d)$ and $o(\text{poly}(d))$	JL	FJLT	FJLTr, FWI	
k in $\Omega(\text{poly}(d))$ and $o((d \log(d)^{1/3})$	JL		FJLT, FJLTr, FWI	
k in $\omega((d \log d)^{1/3})$ and $O(d^{1/2-\delta})$	JL	FJLT, FJLTr	FWI	
k in $O(d^{1/2-\delta})$ and $k < d$	JL, FJLT, FJLTr	JL concatenation		

Tab. 7.1: Result summary. Schematic comparison of asymptotic running time of six projection algorithms. 1) JL: a naïve implementation of Johnson-Lindenstrauss. 2) FJLT: the fast JL transform by Ailon Chazelle [1]. 3) FJLTr: a revised version of the FJLT algorithm, Chapter 3 and [2]. 4) FWI: A new two stage projection process, chapter 5 and [2]. 5) JL concatenation: a concatenation of several independent projections. 6) JL + Mailman: implementation of the Mailman algorithm [35] to Achlioptas's result. Note that $k \in O(\log d)$ when the number of projected vectors n is polynomial in their dimension d.

8. APPLICATIONS AND EXPERIMENTS

Random linear dimensionality reduction has supplied us not only with interesting theoretical results but also with useful practical tools. Experimental papers which use random projections deal with: information retrieval for text documents and images [51], learning Gaussian mixture models [52], data mining and PCA [53] and [27]to name just a few. Although many consider the usage of random projections, they do not consider the differences between different projection algorithms: the quality of length preservation and the speed of computation. Since this manuscript studies the accuracy and efficiency of different projection methods, we now compare between them experimentally. We also experimentally show that random projections are beneficial for fast matrix rank-k approximation.

8.1 Accuracy of projection

In this chapter we will refer to the different constructions by the following names:

- Random-projection: a random k dimensional subspace of $\mathbb{R}^d$ chosen uniformly at random.
- Random-Gaussian: each entry is drawn i.i.d. from the Gaussian distribution with standard deviation $k^{-1/2}$, $\Psi(i,j) \sim N(0, k^{-1/2})$.
- Plus-minus-one: a matrix whose entries are drawn i.i.d. such that:

$$\Psi(i,j) = \begin{cases} \frac{1}{\sqrt{k}} & \text{w.p} \quad 1/2 \\ -\frac{1}{\sqrt{k}} & \text{w.p} \quad 1/2 \end{cases} \tag{8.1}$$

- Sub-Hadamard: a matrix containing k rows (not necessarily the first) out of a $d \times d$ complete Hadamard transform. This family of matrices contains also four-wise independent matrices as in Chapter 5.

$$A(i,j) = H_d(g(i), j). \tag{8.2}$$

For H_d a $d \times d$ Hadamard transform, $i \leq k$ and $g : \{1, \ldots, k\} \to \{1, \ldots, d\}$. The function g can be either random or deterministic.

- Lean-Walsh: a lean Walsh matrix with a seed of size 3×4 as shown in Chapter 6.

$$A_1 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}, \quad A_\ell = A_1 \otimes A_{\ell-1}$$

- Identity-copies: $\Psi(i, j) = 1$ if $i = j \bmod k$ and zero otherwise.

Notice that the latter three constructions are potentially deterministic. Therefore, constructing vectors for which they fail as length preserving projections is trivial using, for example, their eigenvectors. We therefore compose each of the above constrictions with a random permutation and a random ± 1 diagonal matrix. These are both isometric transformations which can be applied to any vector $x \in \mathbb{R}^d$ in linear ($O(d)$) time.

In this scenario, it is unclear which vectors we should compare these constructions on. Randomly chosen vectors from $\mathbb{S}^{d-1}$, for example, are a poor choice since all the methods would behave essentially identical. Indeed, a random vector from $\mathbb{S}^{d-1}$ can be viewed as a unit vector which has been randomly rotated in $\mathbb{R}^d$. Therefor, any projection matrix, fixed or not, will behave similarly and according to the original result by Johnson and Lindenstrauss.

To measure our performance against a challenging set of input vectors, we first look at the error term of the projection. Assume the projection matrix A consists of columns $A^{(i)}$ such that $E[\|A^{(i)}\|_2^2 = 1]$. This is the case for all of the above. Let us compute the expected value of $\zeta = (\|AD_s x\|_2^2 - \|x\|_2^2)^2$, which measures the squared error of the projection.

$$\begin{aligned} E[\zeta] &= E[(\|AD_s x\|_2^2 - \|x\|_2^2)^2] && (8.3) \\ &= E[(x^T D_s A^T A D_s x - x^T x)^2] && (8.4) \\ &= E[(x^T D_s (A^T A - I) D_s x)^2] && (8.5) \\ &= E[(\sum_{i \neq j} \langle A^{(i)}, A^{(j)} \rangle s(i) s(j) x(i) x(j))^2] && (8.6) \\ &= E[\sum_{i_1 \neq j_1} \sum_{i_2 \neq j_2} \langle A^{(i_1)}, A^{(j_1)} \rangle s(i_1) s(j_1) x(i_1) x(j_1) \langle A^{(i_2)}, A^{(j_2)} \rangle s(i_2) s(j_2) x(i_2) x(j_2)] && (8.7) \end{aligned}$$

The term $\langle A^{(i)}, A^{(j)} \rangle$ stands for the inner product between columns i and j. If we open the sum and take the expectation over the $s(i)$ we get that the only non-zero contributions occur when $i_1 = i_2$ and $j_1 = j_2$ or when $i_1 = j_2$ and $j_1 = i_2$ and so:

$$E[\zeta] \quad = \quad 2 \sum_{i \neq j} \langle A^{(i)}, A^{(j)} \rangle^2 x^2(i) x^2(j) \tag{8.8}$$

Notice now that $E[\zeta]$ is a convex function in the variables $y(i) = x^2(i)$. Therefore, $E[\zeta]$ assumes its maximal value on an extremal point of the polytope defined by $y(i) \geq 0$, $\sum_{i=1}^{d} y(i) = 1$. The maximal value for ζ is achieved by $y(i_1) = y(i_2) = 1/2$ where $\langle A^{(i_1)}, A^{(i_2)} \rangle$ assumes the maximal inner product between any pair of columns in A. Assume now that for all i, $y(i) \leq 1/m$ for some integer m ($\|x\|_\infty \leq 1/\sqrt{m}$). The expected distortion is again maximized on extremal points of the polytope by vectors y which contain m entries of value $1/m$ and the rest are all zeros. In order for the input to be as challenging as possible, we consider only such sparse vectors. In what follows m denotes the sparsity of the input vector x, i.e. the number of non-zeros of value $1/\sqrt{m}$. Figures 8.1, 8.2, 8.3, 8.4 show the results of the first experiment. Each of the projection methods was used to project $n = 1000$ sparse unit vectors from dimension $d = 5000$ to dimension $k = 500$. To demonstrate the distribution of projection norms, for each method, the results were sorted such that the hight of the plot line in location i gives the i'th smallest value of $\|\Psi x\|$.

As expected, increasing the value of m, and thereby forcing $\|x\|_\infty \leq 1/\sqrt{m}$, reduces the expected distortion. It can be seen from the plots that even for m being rather small, for example 32, the distribution of projection norms behave very similarly for all the above methods, not including Identity-copies. Remarkably though, even Identity-copies perform rather well when m grows larger.

8.2 Application speed

In what follows we evaluate and compare the running time of applying different transforms. The application running time is measured as a function of the original dimension d and the target dimension k. These are the only important parameters since the the considered algorithms' running time is independent of the vectors' or matrices' entry values.

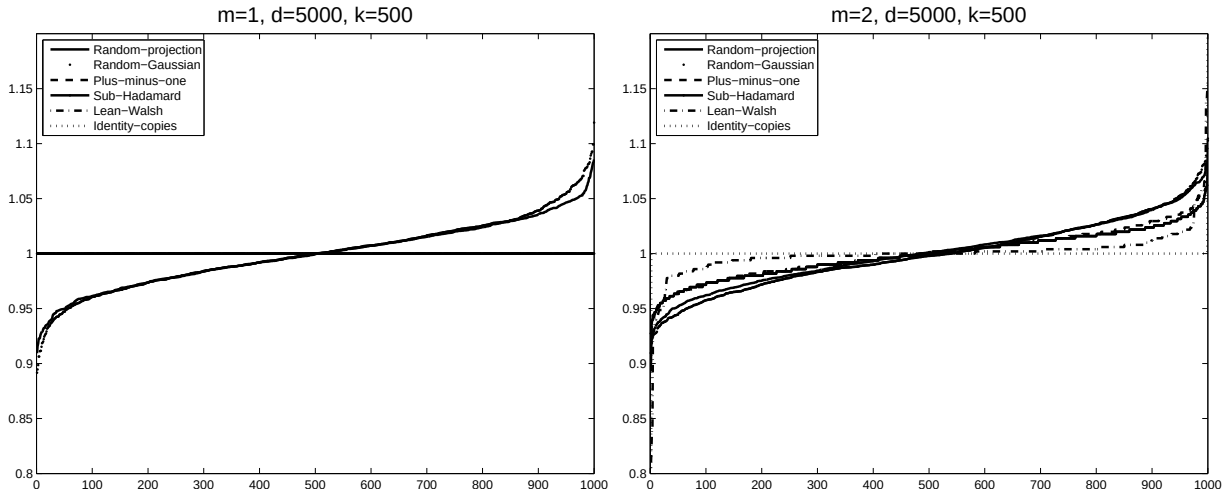


Fig. 8.1: Accuracy of projection for six projection methods as a function of m, the number of non-zeros of value $1/\sqrt{m}$ in the input vectors. When $m = 1$ (left) all deterministic matrices exhibit zero distortion since their column norms are equal to 1. When $m = 2$ (right) all constructions might exhibit a distortion equal to their coherence.

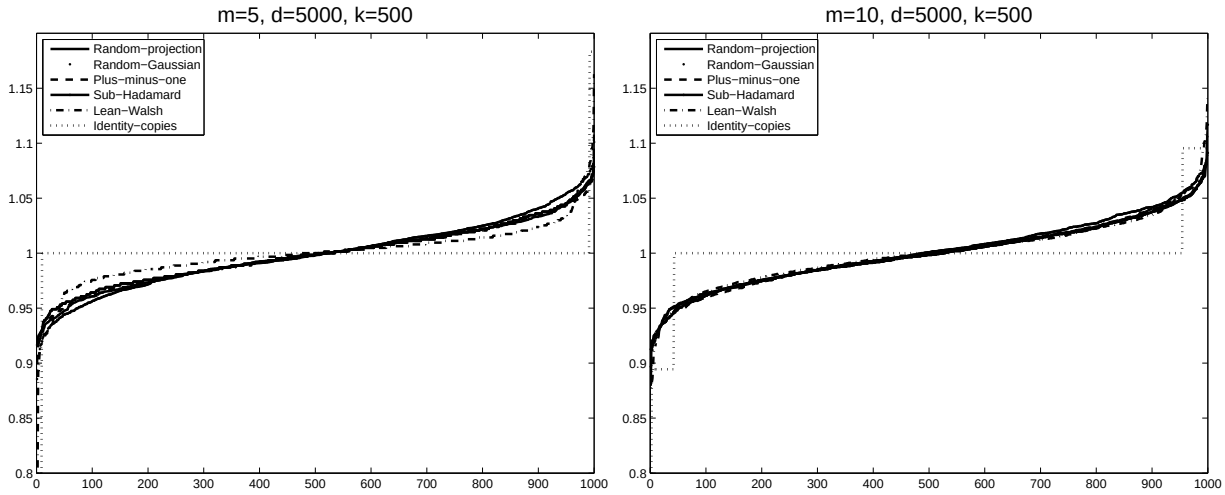


Fig. 8.2: Small values of m give rise to better average behavior by deterministic matrices, but worse worst-case behavior. This stems from the fact that their average coherence is smaller but their maximum coherence is larger.

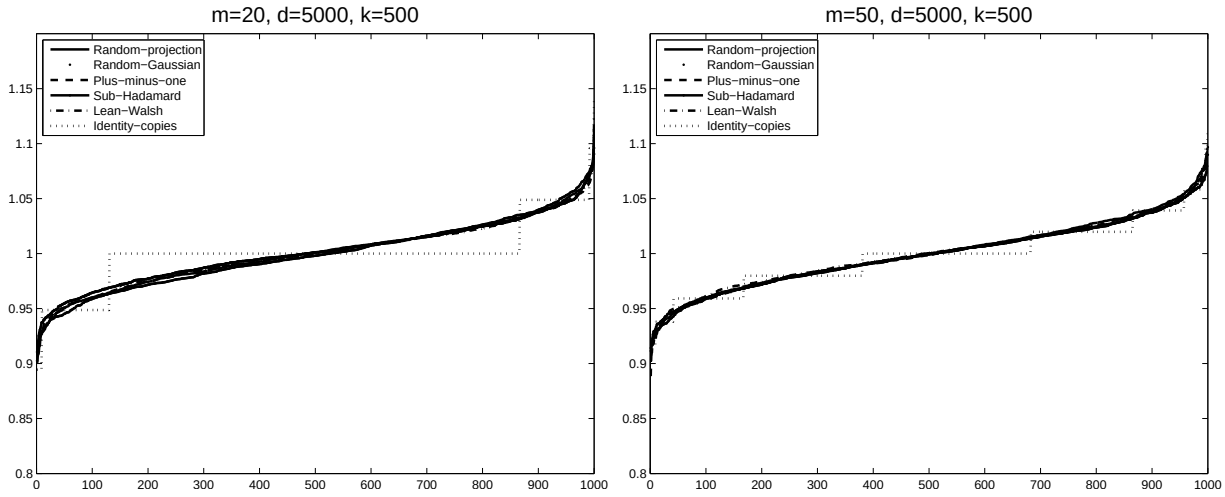


Fig. 8.3: When m grows the behavior of deterministic matrices and dense random ones becomes indistinguishable, with the exception of Identity-copies.

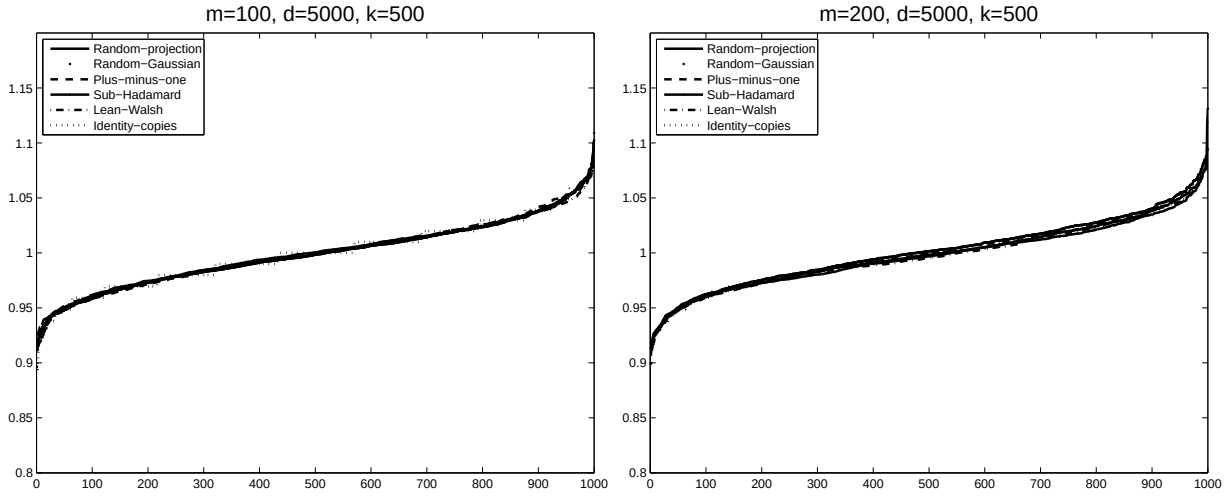


Fig. 8.4: Large values of m allow all methods including Identity-copies to be used equally reliably.

8.2.1 Comparison of dense matrix multiplication and fast transforms

We start with comparing the running time of applying a complete Hadamard transform to a vector of length d as apposed to multiplying it by a dense k by d matrix. Applying a dense $k \times d$ stored matrix to a vector in $\mathbb{R}^d$ requires $O(kd)$ operations whereas computing the Hadamard transform requires $O(d \log(d))$ operations.

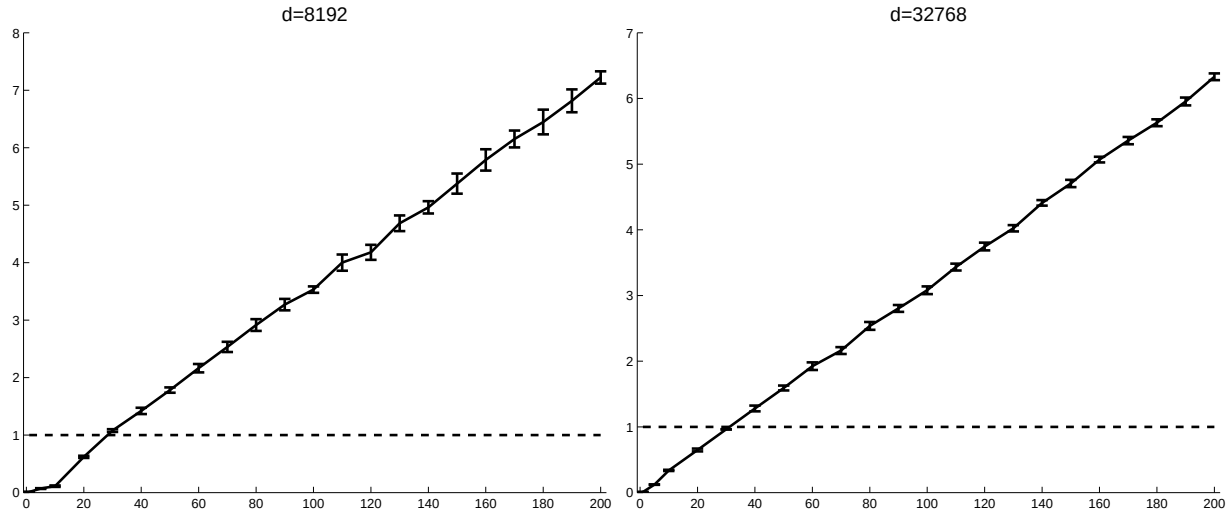


Fig. 8.5: Running time of applying dense stored matrices compared to applying a complete $d \times d$ Hadamard transform. The *x-axis* gives the number of rows in the dense matrix whose application time is given by the solid line. On the *y-axis* running time values are normalized such that applying a complete Hadamard Transform requires 1 unit of time, indicated by the dashed line.

Theoretically, if k is logarithmic in d the two methods should be comparable. Figure 8.5 shows the running time of applying a $k \times d$ matrix as a function of k. The horizontal line gives the running time of applying a complete Hadamard transform. We chose to compare ourselves to Hadamard transforms instead of Fourier transforms since they are real valued and exhibit better memory access patterns. We see in Figure 8.5 that the computational cost of applying a complete Hadamard transform is significantly lower than that of applying a $k \times d$ matrix even for small values of k. Note that the Hadamard transform used is a simple recursive function which is not optimized in any way and includes allocating an array of length d in memory. The matrix multiplication uses the optimized LAPACK toolbox used by Matlab. The values of k and d used in this experiment are limited by the amount of space needed to store dense matrices.

8.2.2 Dependance on the target dimension

After establishing that the use of dense matrices is significantly slower than using fast transforms, we turn to compare those. We start by examining the application running time dependance of sub-Hadamard, Lean-

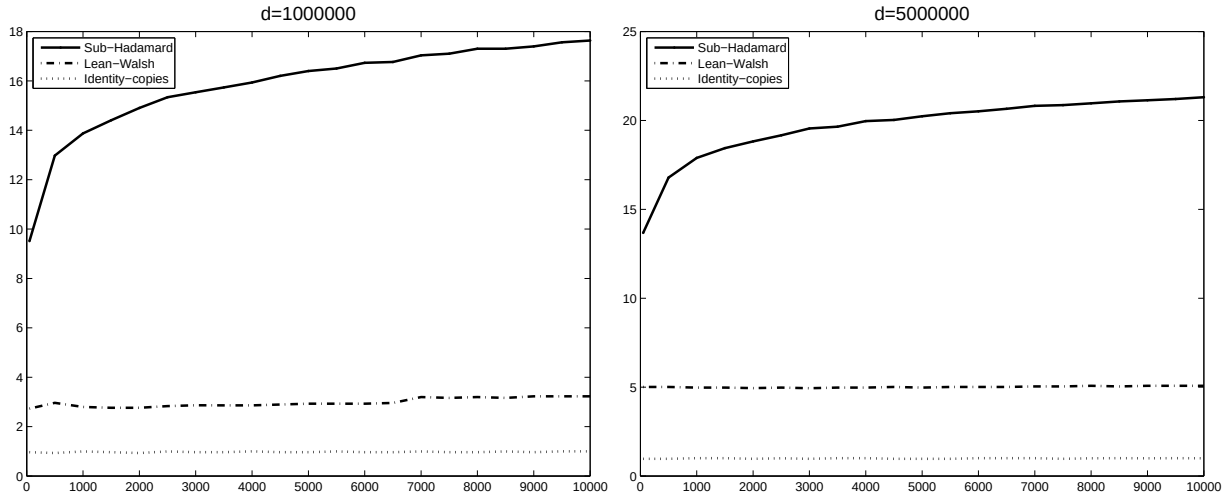


Fig. 8.6: Running time of applying Sub-Hadamard, Lean-Walsh and Identity-copies $k \times d$ matrices. k ranges from 1 to $10,000$ and $d = 10^5$ (left) $d = 5 \cdot 10^6$ (right).

Walsh and Identity-Copies matrices on their target dimension k. Applying sub-Hadamard matrices requires $O(d \log k)$ operations, the logarithmic dependance in k can be seen in Figure 8.6. The application time of Lean-Walsh and Identity-Copies should, however, be independent of k. The slight dependence on k which is seen in Figure 8.6 is due to two facts. One, small values of k allow for partial applications of recursive calls during the computation of lean Walsh transforms. Two, memory allocation and management issues still depend on the length of the output, for example, the time required to allocate (and initialize) the result vector clearly depends on its length.

8.2.3 Dependance on the original dimension

Here we examine the running time for applying Sub-Hadamard, Lean-Walsh, and Identity-copies for a fixed target dimension k and varying original dimension d. The results in Figure 8.7 do not seem linear because the application time required to perform sub-Hadamard transforms and lean Walsh transform depends on the proximity of the original dimension to a round power of 2 or 4 respectively. The running time is, nonetheless, linearly dependent on d. The results are normalized such that applying Identity-Copies to the maximal value of k and d requires 1 unit of time.

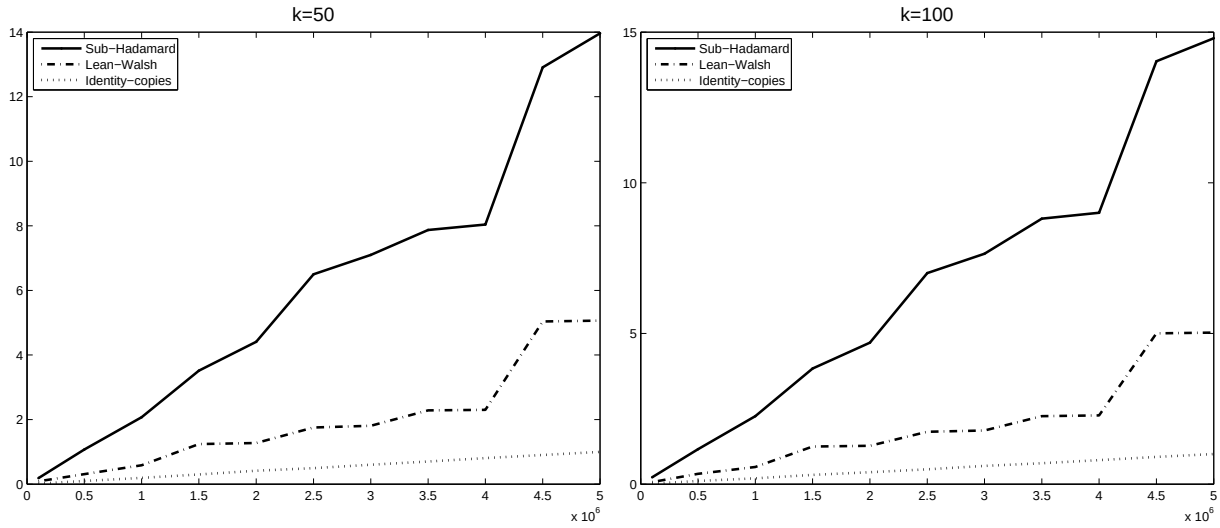


Fig. 8.7: Running time of applying three $k \times d$ matrices: Sub-Hadamard, lean Walsh and Identity-copies. Here, the original dimension ranges between $d = 10^5$ to $d = 5 \cdot 10^6$. The target dimension is fixed to $k = 100$ (left) and $k = 1000$ (right). All values are normalized such that applying Identity-copies to $d = 5 \cdot 10^6$ requires 1 unit of time.

We see that, across the board, there is a clear order of application efficiency. Most efficient are, of course, Identity copies. Lean-Walsh are slower to apply and least efficient are Sub-Hadamard matrices. The reader should bare in mind that computing complete Hadamard or Fourier transforms is a much heavier operation than all of the above by a large factor. Needless to say, application of dense random matrices is prohibitively heavy for most of the matrix dimensions tested above.

8.3 Rank-k approximations

In many algorithms in data mining, signal processing, physical simulations (and many others) a key step is computing a rank deficient approximation of a large matrix M. The matrix M is usually assumed to be the addition of a rank deficient matrix and a matrix of entry-wise i.i.d. noise. More formally we seek a matrix M_r that minimizes $\|M - M_r\|_2$ under the constraint that M_r is rank r. Finding M_r exactly is possible using the Singular Value Decomposition (SVD) of M. An SVD of a matrix M is a decomposition of it into three

matrices U, S, and V such that $M = USV^T$, U and V are orthonormal and S is a diagonal matrix holding on the diagonal the singular values of M, $S(i,i) = \sigma_i$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d \geq 0$. It is a well known fact that the best low rank approximation of M is $M_r = U_r S_r V_r^T$, here U_r and V_r denote the first r columns of U and V and S_r contains $\sigma_1 \ldots \sigma_r$. Directly computing the SVD of a matrix requires iteratively applying M to candidate singular vectors. These methods include the Power-Method and iterative methods such as Lanczos. These are time-wise prohibitive if the size of M is large.

Since finding M_r exactly and deterministically is prohibitive, randomized approximation algorithms were explored for this problem. The idea in all of the following algorithms is similar. Randomly generate a matrix M_k by multiplying M from the left by a random matrix R such that the matrix $M_k = RM$ is much smaller than M. Decompose M_k exactly and use its SVD to approximate that of M. The crux is to show that a proper choice of distribution for R guaranties (w.h.p.) that the singular values and right singular vectors of RM are close to those of M. The definition of "closeness" and the distributions over R vary. It is clear that k must be at least as large as r, the desired rank. However, it has been the focus of much research as to what values of k relative to r guarantee what results.

One possible distribution for R is a matrix that encodes a sub-sampling (and possibly scaling) of k rows from M. This method, pioneered by Frieze Kannan and Vempala in [13] was later refined and improved by Kannan, Vempala, Mahoney, Muthukrishnan, and Drineas in a series of papers [14, 15, 16]. The strongest result in this line of work was given recently by Rudelson and Vershinin [54]. From this point on, we refer to the method described in [54] as *random sampling*. For completeness we recall the relevant theorem.

Theorem 8.3.1 (Rudelson, Vershinin [54])**.** *Let M be a $d \times n$ matrix with numerical rank $r = \|M\|_{fro}^2 / \|M\|_2^2$. Let $\varepsilon, \delta \in (0,1)$ and let $k \leq d$ be an integer such that:*

$$k \geq C \frac{r}{\varepsilon^4 \delta} \log\left(\frac{r}{\varepsilon^4 \delta}\right) \tag{8.9}$$

Consider the $k \times m$ matrix M_k which consists of k normalized rows of M, picked independently with replacement with probabilities proportional to their squared euclidian norms. Then, with probability at least $1 - 2e^{-c/\delta}$ the following holds; For a positive integer ℓ, let P_ℓ be the orthogonal projection onto the top ℓ left

singular vectors of M_k then:

$$\|M - MP_\ell\|_2 \leq \sigma_{\ell+1}(M) + \varepsilon\|M\|_2 \tag{8.10}$$

Remark 8.3.1 (Rudelson, Vershinin [54]). *The numerical rank $r = r(M) = \|M\|_{fro}^2/\|M\|_2^2$ in Theorem 8.3.1 is a relaxation of the exact notion of rank. Indeed, one always has $r(M) \leq rank(M)$. But as opposed to the exact rank, the numerical rank is stable under small perturbations of the matrix M. In particular, the numerical rank of M tends to be low when M is close to a low rank matrix, or when M is sufficiently sparse.*

Another idea, natural to this manuscript, is to use random projections for this task. Such algorithms are given by Sarlos [17], Drineas, Mahoney, Muthukrishnan [23] and Tygert, Martinsson, Rokhlin, Woolfe, and Liberty [20]. In a sense, a random projection of a matrix M can be viewed as simply sampling an orthogonally rotated version ΦM of the original matrix M. This reduces random projections to random sampling. In what follows we claim that although a random projection of a matrix is more computationally expensive than random sampling, it is not redundant. It serves as an important preconditioning step to the sampling algorithm which improves its accuracy. Thus, when computationally possible, one should prefer random projection to random sampling.

Our measure for accuracy is motivated by learning algorithms such as kernel methods and Latent Semantic Indexing (LSI). In these applications the data matrix M is assumed to be low rank. For kernel methods M is the $d \times d$ *Kernel* matrix and for LSI M is the $d \times n$ document-word frequency matrix. The r right singular vectors of M are thus assumed (by the model) to span the part of $\mathbb{R}^d$ which includes the information about the data and does not include artifacts arising from noise. The definition of noise is specific to the data model but a common assumption, especially in signal processing, is that the noise is additive i.i.d. and Gaussian distributed. Moreover, it is assumed that all singular values of the noise matrix are lower than the smallest non-zero singular value of M.

In the experiment described below we set the data matrix to $D = UU^T$ where U is a $d \times r$ orthogonal matrix. D, therefore is a symmetric matrix with r singular values equal to 1. We also define the noise matrix N to be a $d \times d$ random Gaussian matrix, $N(i,j) \sim N(0, 1/\sqrt{d})$. The entry-wise standard deviation of $1/\sqrt{d}$ is required to have $E(\|N\|_2) = O(1)$ (for noise distribution other than Gaussian the reader is referred to

[55]). Finally, we denote by M the matrix $M = D + \sigma N$.

Using the matrix M we are interested in producing a $d \times k$ orthogonal matrix V which numerically spans the right singular space of D i.e. the columns of the orthogonal $d \times r$ matrix U. We measure the accuracy of the two algorithms by the portion of U that is captured by V. A vector $u \in \mathbb{R}^d$ is perfectly spanned by a subspace V if $\|u^T V\| = \|u\|$ (for any vector $0 \leq \|u^T V\| \leq \|u\|$). The quality of spanning all r columns in U can be measured by $q = \|U^T V\|_{Fro}/\sqrt{r}$. The coefficient $q \in [0, 1]$ measures the quality of the algorithms' output.

Remark 8.3.2. *The experiment is set up in a way which allows us not to store the entire matrix M in memory in any point. Rather, we generate RD directly from $(RU)U^T$ and generate N on the fly to calculate RN. This is crucial since working directly with M stored on disk would have generated a large number of memory page faults, requiring an excessive amount of time. This, however, is irrelevant for the accuracy results stated below.*

Remark 8.3.3. *Although the running time of randomly projecting M is much longer than that of randomly sampling it, the application time of both methods is not compared for two reasons. First, as stated above, we do not work with the matrix M directly and so the running time measured would have been heavily skewed. Second, usually the computational bottleneck of these method is that of computing the exact decomposition of the projected matrix, RM. Thus, the accuracy of the output is usually a more important factor.*

Remark 8.3.4. *Due to the size of M, one cannot calculate its true rank-k approximation. We therefore measure our success by the accuracy of spanning U. This is justified by the assumption of the model that U and the singular space of M are close.*

Remark 8.3.5. *The choice of the Gaussian noise matrix N, $N(i, j) \sim N(0, 1/\sqrt{d})$, only guaranties that $E(\|N\|_2) \leq c$ for some constant c. It is important to notice that it does not imply $\|N\|_2 \leq \lambda_r = 1$ as would be required by the model. Moreover, for Gaussian i.i.d. entries $E(\|N\|_2)$ is larger than 1 and experimentally takes a value close to 2. Since $M = D + \sigma N$, the coefficient c will be implicitly included in the value for σ.*

Figure 8.8 compares the two algorithms' results for a random orthogonal matrix U. Since the distribution

for U is rotationally invariant, the results of sampling M directly should be comparable to those of sampling a rotated version of it. The random projection matrix used was the Lean-Walsh matrix as above. Figure 8.8 shows the results obtained for $r = 10$, $d = 10,000$ and $d = 20,000$ and $\sigma = 0.05$ and $\sigma = 0.1$. The results of the two approaches are comparable as expected.

The difference between the methods becomes apparent when the distribution of U is not isotropic. In the next experiment we devise a different matrix U. The second version of U is such that most of its row norms are small and a few are large.[1] This can be achieved in many ways. In the following experiment, U was chosen as follows. First, all $d \times r$ entrees were assigned i.i.d random Gaussian entrees. Second, each row was multiplied by the square of a random Gaussian variable. Finally, U is orthogonalized using a Graham-Schmidt procedure. The results are shown in Figure 8.9.

In all of the above experiments r, the rank of U, was fixed. Figure 8.10 shows the effect of increasing the value of r. As expected, the number of sampled rows or the projection dimension required to achieve a certain accuracy increases as well. Note that the plot look similar since our measure of quality is normalized with respect to the rank. The advantage to random projection is still evident.

8.4 Conclusion

This chapter dealt with practical issues regarding random projections. First we saw that the projection accuracy of different projection distributions is essentially identical unless the projected vectors are extremely sparse. This led us to prefer one projection over another according to its application speed which was reviewed in the second section. The third section was dedicated to affirming the usefulness of random projections for performing matrix Singular Value Decomposition.

[1] The definition of "most" and "few" is somewhat loose.

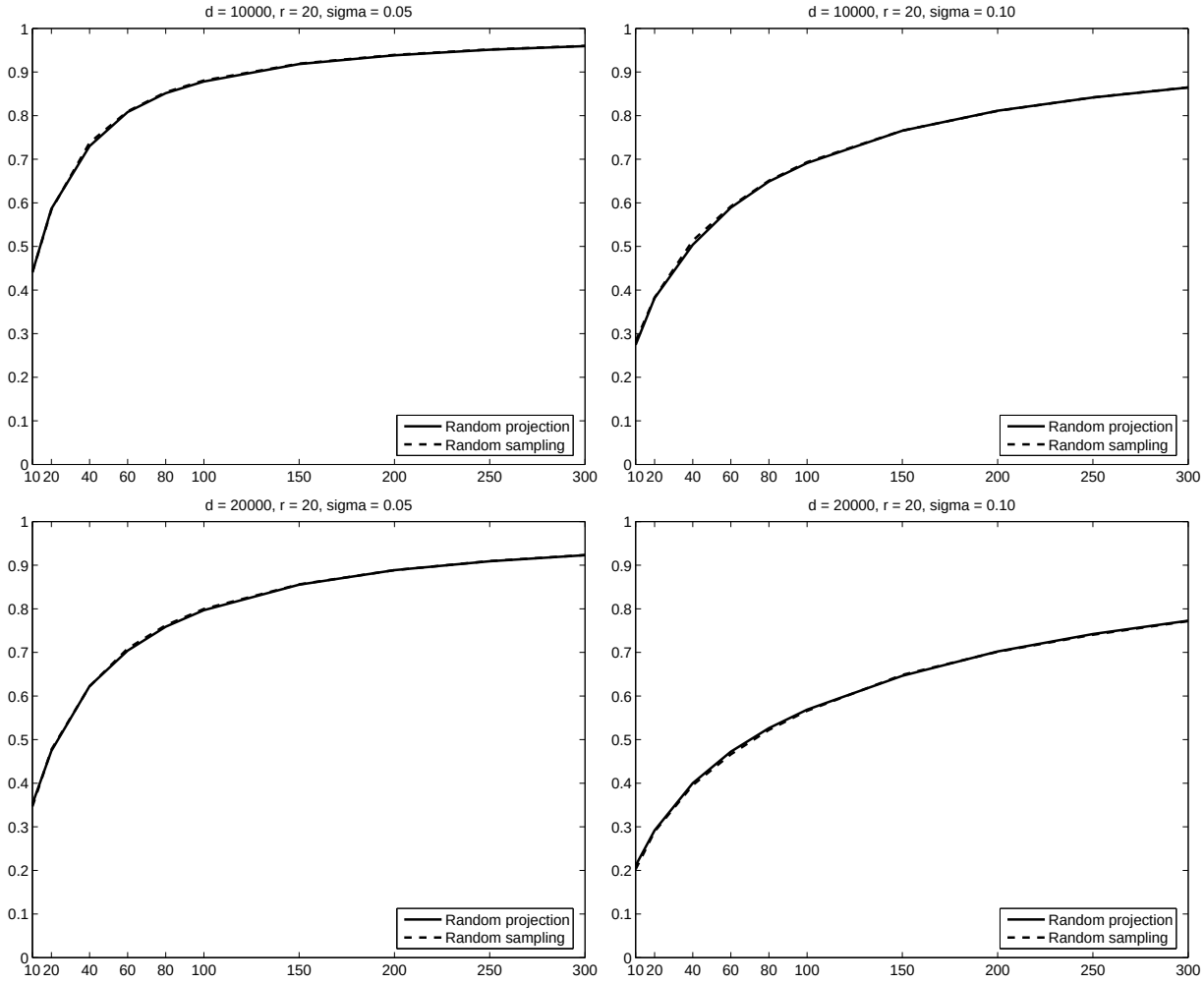


Fig. 8.8: Four plots describing the rank-k approximation of $d \times d$ matrices, $d = 10,000$ or $d = 20,000$ as indicated. On the x axis the reduced dimension of the matrix: either the sample size in the case of random sampling or the target dimension in the case of random projections. The y axis indicates the quality of approximation q. The approximated matrix is $M = UU^T + \sigma N$. Where U is a random $d \times r$ partial basis for $\mathbb{R}^d$ and $N(i,j) \sim N(0, 1/\sqrt{d})$ i.i.d. Since U is chosen according to an isotropic distribution the two approaches perform comparably.

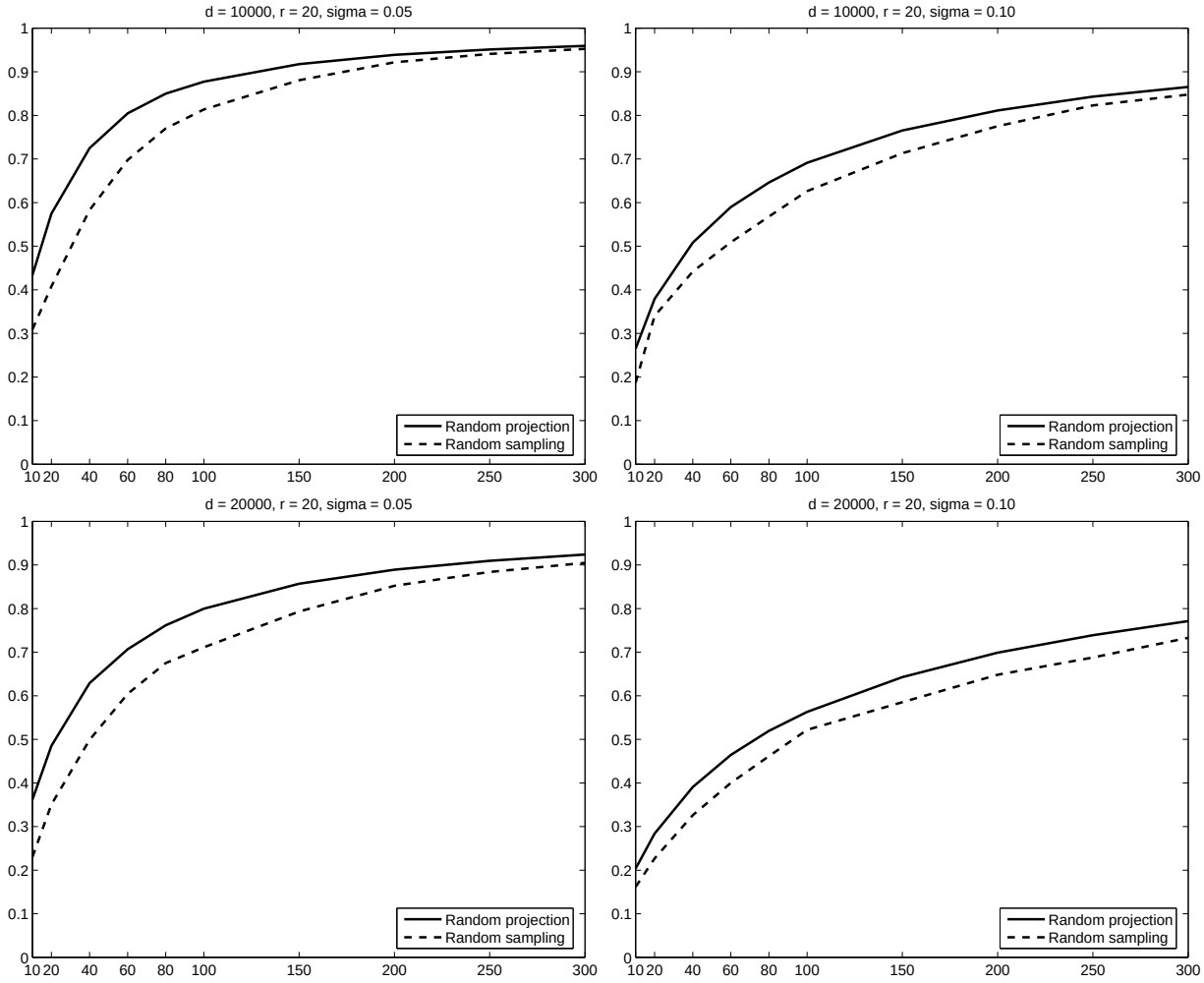


Fig. 8.9: Four plots describing the rank-k approximation of $d \times d$ matrices, $d = 10,000$ or $d = 20,000$ as indicated. On the x axis the reduced dimension of the matrix: either the sample size in the case of random sampling or the target dimension in the case of random projections. The y axis indicates the quality of approximation q. The approximated matrix is $M = UU^T + \sigma N$, where U is a column orthogonal $d \times r$ matrix and $N(i,j) \sim N(0, 1/\sqrt{d})$ i.i.d. The choice of U here is such that its row norms vary like square gaussians. This is not an isotropic phenomenon in $\mathbb{R}^d$ and thus the difference in accuracy between the two algorithms' performance.

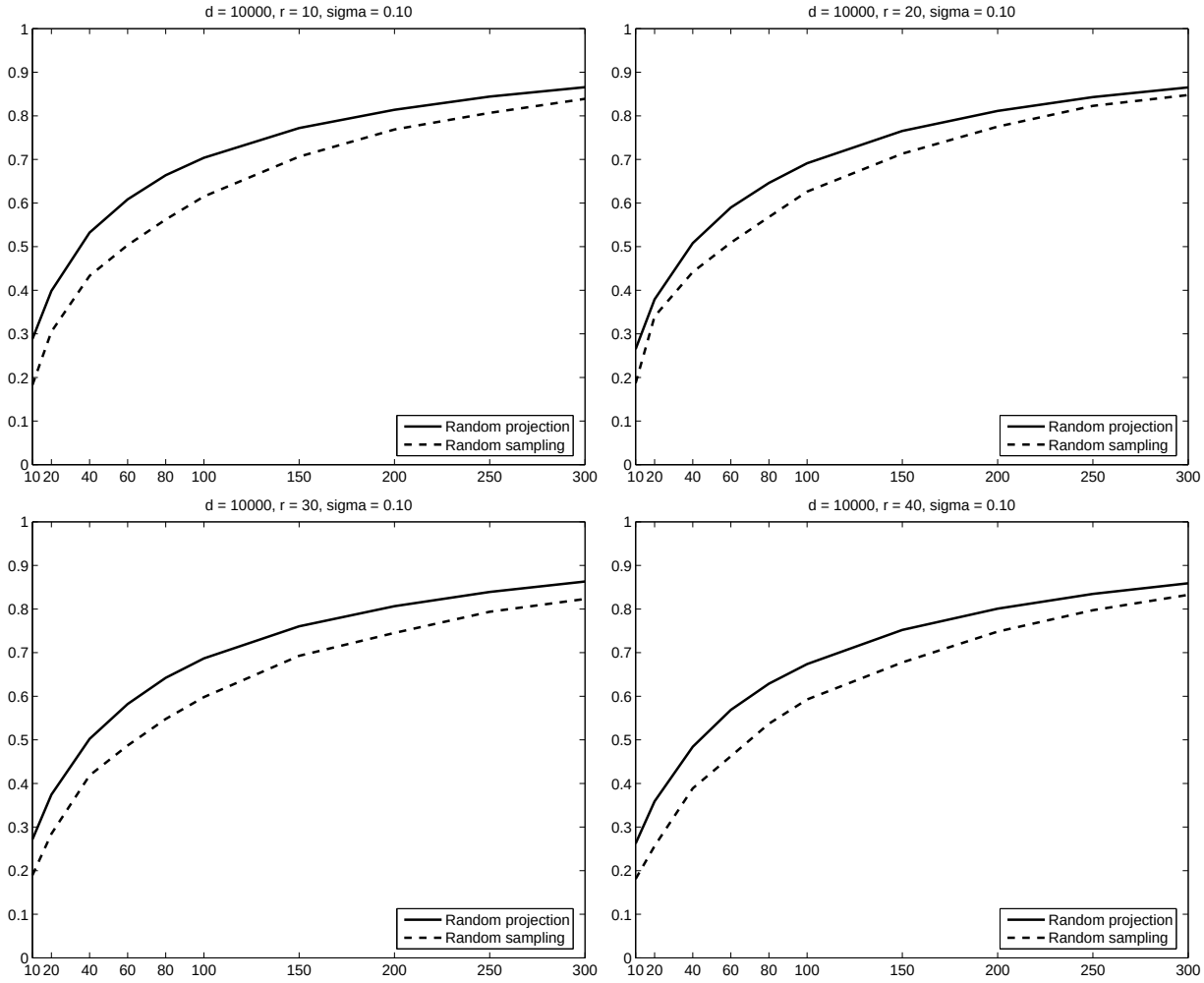


Fig. 8.10: Rank-k approximation of $d \times d$ matrices, $d = 5000$ as indicated. On the x axis the reduced dimension of the matrix: either the sample size in the case of random sampling or the target dimension in the case of random projections. The y axis indicates the quality of approximation q. The approximated matrix is $M = UU^T + \sigma N$. Where U is a column orthogonal $d \times r$ matrix and $N(i,j) \sim N(0, 1/\sqrt{d})$ i.i.d. The value of r is increases from left to right and top down from 10 to 20 to 30 to 40.

BIBLIOGRAPHY

[1] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563, New York, NY, USA, 2006. ACM Press.

[2] Nir Ailon and Edo Liberty. Fast dimension reduction using rademacher series on dual bch codes. In *SODA*, pages 1–9, 2008.

[3] W.B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[4] Noga Alon. Problems and results in extremal combinatorics–I. *Discrete Mathematics*, 273(1-3):31–53, 2003.

[5] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

[6] Sunil Arya and David M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 271–280, Austin, Texas, United States, 1993.

[7] Piotr Indyk. On approximate nearest neighbors in non-Euclidean spaces. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 148–155, 1998.

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

[9] Piotr Indyk. Nearest neighbors in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry*. CRC Press, 2004.

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

[11] Leslie G. Valiant. A neuroidal architecture for cognitive computation. *Lecture Notes in Computer Science*, 1443:642, 1998.

[12] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[13] Alan M. Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. In *IEEE Symposium on Foundations of Computer Science*, pages 370–378, 1998.

[14] Petros Drineas and Ravi Kannan. Fast monte-carlo algorithms for approximate matrix multiplication. In *IEEE Symposium on Foundations of Computer Science*, pages 452–459, 2001.

[15] P. Drineas, R. Kannan, and M. Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix, 2004.

[16] P. Drineas, R. Kannan, and M. Mahoney. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition, 2004.

[17] 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)*, Berkeley, CA, 2006.

[18] Achlioptas and McSherry. Fast computation of low rank matrix approximations. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2001.

[19] P.G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the approximation of matrices.

[20] Liberty Edo, Woolfe Franco, Martinsson Per-Gunnar, Rokhlin Vladimir, and Tygert Mark. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, December 2007.

[21] P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Relative-error cur matrix decompositions. *TR arXiv:0708.3696*, submitted for publication, 2007.

[22] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlos. Faster least squares approximation. *TR arXiv:0710.1435*, submitted for publication, 2007.

[23] P. Drineas, M. W. Mahoney, and S.M. Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Miami, Florida, United States, 2006.

[24] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. *Proc. of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008.

[25] Santosh Vempala. *The Random Projection Method.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 2004.

[26] Sariel Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, Las Vegas, Nevada, USA, 2001.

[27] P. Paschou, E. Ziv, E. Burchard, S. Choudhry, W. Rodriguez-Cintron, M. W. Mahoney, and P. Drineas. Pca-correlated snps for structure identification in worldwide human populations. *PLOS Genetics, 3, pp. 1672-1686*, 2007.

[28] P. Paschou, M. W. Mahoney, J. Kidd, A. Pakstis, K. Kidd S. Gu, and P. Drineas. Intra- and inter-population genotype reconstruction from tagging snps. *Genome Research, 17(1), pp. 96-107*, 2007.

[29] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Knowledge Discovery and Data Mining*, pages 245–250, 2001.

[30] C.H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the 17th Annual Symposium of Database Systems*, pages 159–168, 1998.

[31] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical Report TR-99-006, Berkeley, CA, 1999.

[32] P. Frankl and H. Maehara. The johnson-lindenstrauss lemma and the sphericity of some graphs. *J. Comb. Theory Ser. A*, 44(3):355–362, 1987.

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

[34] J. Matousek. On variants of the Johnson-Lindenstrauss lemma. *Private communication*, 2006.

[35] Edo Liberty and Steven Zucker. The mailman algorithm: a note on matrix vector multiplication. In *Yale university technical report #1402*, 2008.

[36] Edo Liberty, Nir Ailon, and Amit Singer. Dense fast random projections and lean walsh transforms. In *APPROX-RANDOM*, pages 512–522, 2008.

[37] M. Ledoux and M. Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer-Verlag, 1991.

[38] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error Correcting Codes*. North-Holland, 1983.

[39] J. Bergh and J. Lofstrom. *Interpolation Spaces*. Springer-Verlag, 1976.

[40] Carsten Schütt Hermann König and Nicole Tomczak Jaegermann. Projection constants of symmetric spaces and variants of khintchine's inequality. *J. Reine Angew. Math*, 511:1–42, 1999.

[41] Shmuel Winograd. On the number of multiplications necessary to compute certain functions. *Communications on Pure and Applied Mathematics*, (2):165–179, 1970.

[42] M.A. Kronrod V.L. Arlazarov, E.A. Dinic and I.A. Faradzev. On economic construction of the transitive closure of a direct graph. *Soviet Mathematics, Doklady*, (11):1209–1210, 1970.

[43] N Santoro and J Urrutia. An improved algorithm for boolean matrix multiplication. *Computing*, 36(4):375–382, 1986.

[44] Nicola Santoro. Extending the four russians' bound to general matrix multiplication. *Inf. Process. Lett.*, 10(2):87–88, 1980.

[45] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, (4):354–356, 08 1969.

[46] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 1–6, New York, NY, USA, 1987. ACM.

[47] Roger W. Brockett and David P. Dobkin. On the number of multiplications required for matrix multiplication. *SIAM J. Comput.*, 5(4):624–628, 1976.

[48] Ryan Williams. Matrix-vector multiplication in sub-quadratic time: (some preprocessing required). In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 995–1001, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[49] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for fortran usage. *ACM Trans. Math. Softw.*, 5(3):308–323, 1979.

[50] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Richard J. Hanson. An extended set of fortran basic linear algebra subprograms. *ACM Trans. Math. Softw.*, 14(1):1–17, 1988.

[51] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, New York, NY, USA, 2001. ACM.

[52] Sanjoy Dasgupta. Experiments with random projection. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 143–151, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[53] Dmitriy Fradkin and David Madigan. Experiments with random projections for machine learning. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–522, New York, NY, USA, 2003. ACM.

[54] Mark Rudelson and Roman Vershynin. Sampling from large matrices: an approach through geometric functional analysis, 2005.

[55] Rafal Latala. Some estimates of norms of random matrices. *Proc. Amer. Math. Soc.*, 2005.