Faster Johnson-Lindenstrauss style reductions

Aditya Menon

August 23, 2007

Distances

- For high-dimensional vector data, it is of interest to have a notion of distance between two vectors
- Recall that the ℓ_p norm of a vector $\mathbf{x}$ is

$$||\mathbf{x}||_p = \left(\sum |x_i|^p\right)^{1/p}$$

- The ℓ_2 norm corresponds to the standard Euclidean norm of a vector
- The ℓ_∞ norm is the maximal absolute value of any component

$$||\mathbf{x}||_\infty = \max_i |x_i|$$

Dimensionality reduction

- Suppose we're given an input vector $\mathbf{x} \in \mathbb{R}^d$
- We want to reduce the dimensionality of $\mathbf{x}$ to some $k < d$, while preserving the ℓ_p norm
  - Can think of this as a metric embedding problem - can we embed ℓ_p^d into ℓ_p^k?
- Formally, we have the following problem

Problem

Suppose we are given an $\mathbf{x} \in \mathbb{R}^d$, and some parameters p, ϵ. Can we find a $\mathbf{y} \in \mathbb{R}^k$ for some $k = f(\epsilon)$ so that

$$(1 - \epsilon)||\mathbf{x}||_p \leq ||\mathbf{y}||_p \leq (1 + \epsilon)||\mathbf{x}||_p$$

The Johnson-Lindenstrauss Lemma

- The Johnson-Lindenstrauss Lemma [5] is the archetypal result for ℓ_2 dimensionality reduction
- Tells us that for n points, there is an ϵ-embedding of $\ell_2^d \to \ell_2^{O(\log n/\epsilon^2)}$

Theorem

Suppose $\{\mathbf{u_i}\}_{i=1\ldots n} \in \mathbb{R}^{n \times d}$*. Then, for* $\epsilon > 0$ *and* $k = O(\log n/\epsilon^2)$*, there is a mapping* $f : \mathbb{R}^d \to \mathbb{R}^k$ *so that*

$$(\forall i,j)(1-\epsilon)||\mathbf{u_i} - \mathbf{u_j}||_2 \leq ||f(\mathbf{u_i}) - f(\mathbf{u_j})||_2 \leq (1+\epsilon)||\mathbf{u_i} - \mathbf{u_j}||_2$$

Johnson-Lindenstrauss in practice

- Proof of Johnson-Lindenstrauss lemma is non-constructive (unfortunately!)
- In practise, we use the probabilistic method to do a Johnson-Lindenstrauss style reduction
- Insert randomness at the cost of an exact guarantee
    - Now the guarantee becomes probabilistic

Johnson-Lindenstrauss in practice

- Standard version:

Theorem

Suppose $\{\mathbf{u_i}\}_{i=1\ldots n} \in \mathbb{R}^{n\times d}$. *Then, for* $\epsilon > 0$ *and* $k = O(\beta \log n/\epsilon^2)$, *the mapping* $f(\mathbf{u_i}) = \frac{1}{\sqrt{k}}\mathbf{u_i}R$, *where* R *is a* $d \times k$ *matrix of i.i.d. Gaussian variables, satisfies with probability at least* $1 - \frac{1}{n^\beta}$,

$$(\forall i, j)(1-\epsilon)||\mathbf{u_i} - \mathbf{u_j}||_2 \leq ||f(\mathbf{u_i}) - f(\mathbf{u_j})||_2 \leq (1+\epsilon)||\mathbf{u_i} - \mathbf{u_j}||_2$$

Achlioptas' improvement

- Achlioptas [1] gave an ever simpler matrix construction:

$$R_{ij} = \sqrt{3} \begin{cases} +1 & \text{probability} = \frac{1}{6} \\ 0 & \text{probability} = \frac{2}{3} \\ -1 & \text{probability} = \frac{1}{6} \end{cases}$$

- $\frac{2}{3}$rds sparse, and simpler to construct than a Gaussian matrix
  - With no loss in accuracy!

A question

- $\frac{2}{3}$rds sparsity is a good speedup in practise
  - But density is still $O(dk)$
  - Computing the mapping is still an $O(dk)$ operation asymptotically
- Let

$$\mathcal{A} = \{A : \forall \text{ unit } \mathbf{x} \in \mathbb{R}^d, \text{ with v.h.p., } (1-\epsilon) \leq ||A\mathbf{x}||_2 \leq (1+\epsilon)\}$$

- **Question**: For which $A \in \mathcal{A}$ can $A\mathbf{x}$ be computed quicker than $O(dk)$?

The answer?

- We look at two approaches that allow for quicker computation
- First is the *Fast Johnson-Lindenstrauss transform*, based on a Fourier transform
- Next is the *Ailon-Liberty Transform*, based on a Fourier transform and error correcting codes!

The Fast Johnson-Lindenstrauss Transform

- Ailon and Chazelle [2] proposed the Fast Johnson-Lindenstrauss transform
- Can speedup ℓ_2 reduction from $O(dk)$ to (roughly) $O(d \log d)$
- How?
  - Make the projection matrix even sparser
  - Need some "tricks" to solve the problems associated with this
- Let's reverse engineer the construction...

Sparser projection matrix

- Use the projection matrix

$$P \sim \begin{cases} \mathcal{N}\left(0, \frac{1}{q}\right) & p = q \\ 0 & p = 1 - q \end{cases}$$

  where

$$q = \min\left\{\Theta\left(\frac{\log^2 n}{d}\right), 1\right\}$$

- Density of the matrix is $O\left(\frac{1}{\epsilon^2}\min\left\{\log^3 n, d \log n\right\}\right)$
  - In practise, this is typically significantly sparser than Achlioptas' matrix

What do we lose?

- Can follow standard concentration-proof methods
- But we end up needing to assume that $||\mathbf{x}||_\infty$ is bounded - namely, that information is *spread out*
  - We fail on vectors like $\mathbf{x} = (1, 0, \ldots, 0)$ i.e. sparse data and a sparse projection don't mix well
- So are we forced to choose between generality or usefulness?

What do we lose?

- Can follow standard concentration-proof methods
- But we end up needing to assume that $||\mathbf{x}||_\infty$ is bounded - namely, that information is *spread out*
  - We fail on vectors like $\mathbf{x} = (1, 0, \ldots, 0)$ i.e. sparse data and a sparse projection don't mix well
- So are we forced to choose between generality or usefulness?
  - Not if we try to insert randomness...

A clever idea

- Can we randomly transform $\mathbf{x}$ so that
  - $||\Phi(\mathbf{x})||_2 = ||\mathbf{x}||_2$
  - $||\Phi(\mathbf{x})||_\infty$ is bounded with v.h.p.?

A clever idea

- Can we randomly transform $\mathbf{x}$ so that
  - $||\Phi(\mathbf{x})||_2 = ||\mathbf{x}||_2$
  - $||\Phi(\mathbf{x})||_\infty$ is bounded with v.h.p.?
- **Answer**: Yes! Use a Fourier transform $\Phi = \mathcal{F}$
  - Distance preserving
  - Has an "uncertainty principle" - a "signal" and its Fourier transform cannot *both* be concentrated
- Use the FFT to give an $O(d \log d)$ random mapping
- Details on the specifics in next section...

Applying a Fourier transform

- Fourier transform will guarantee that

$$||\mathbf{x}||_\infty = \omega(1) \iff ||\widehat{\mathbf{x}}||_\infty = o(1)$$

- But now we will be in trouble if the input is uniformly distributed!
- To deal with this, do a random sign change:

$$\widetilde{\mathbf{x}} = D\mathbf{x}$$

  where D is a random diagonal ± 1 matrix
- Now we get a guarantee of spread with high probability, so the "random" Fourier transform gives us back generality

Random sign change

- The sign change mapping $D\mathbf{x}$ will give us

$$\widetilde{\mathbf{x}} = \begin{bmatrix} d_1 x_1 \\ d_2 x_2 \\ \vdots \\ d_d x_d \end{bmatrix} = \begin{bmatrix} \pm x_1 \\ \pm x_2 \\ \vdots \\ \pm x_d \end{bmatrix}$$

  where the $\pm$ are attained with equal probability
- Clearly norm preserving

Putting it together

- So, we compute the mapping $f : \mathbf{x} \mapsto P\mathcal{F}(D\mathbf{x})$
- Runtime will be

$$O\left(d \log d + \min\left\{\frac{d \log n}{\epsilon^2}, \frac{\log^3 n}{\epsilon^2}\right\}\right)$$

- Under some loose conditions, runtime is

$$O\left(\max\left\{d \log d, k^3\right\}\right)$$

- If $k \in \left[\Omega(\log d), O(\sqrt{d})\right]$, this is quicker than the $O(dk)$ simple mapping
    - In practise, upper bound is reasonable, lower bound might not be though

Summary

- Tried increasing sparsity with disregard for generality
- Used randomization to get back generality (probabilistically)
- Key ingredient was a Fourier transform, with a randomization step first

Ailon and Liberty's improvement

- Ailon and Liberty [3] improved the runtime from $O(d \log d)$ to $O(d \log k)$, for $k = O\left(d^{1/2-\delta}\right)$, $\delta > 0$
- **Idea**: Sparsity isn't the only way to speedup computation time
  - Can also speedup runtime when the projection matrix has a special structure
  - So find a matrix with a convenient structure *and* which will satisfy the JL property

Operator norm

- We need something called the *operator norm* in our analysis
- The operator norm of a transformation matrix A is

$$||A||_{p \to q} = \sup_{||\mathbf{x}||_p = 1} ||A\mathbf{x}||_q$$

  i.e. maximal q norm of the transformation of unit ℓ_p-norm points
- A fact we will need to employ:

$$||A||_{p_1 \to p_2} = ||A^T||_{q_2 \to q_1}$$

  where $\frac{1}{p_1} + \frac{1}{q_1} = 1, \frac{1}{p_2} + \frac{1}{q_2} = 1$

Reverse engineering

- Let's say the mapping is a matrix multiplication
- In particular, say we have a mapping of the form

$$f : \mathbf{x} \mapsto BD\mathbf{x}$$

  where B is some $k \times d$ matrix with unit columns, and D is a diagonal matrix whose entries are randomly ± 1
  - Doing a random sign change again
- Now we just need to see what properties we will need B to satisfy in order for

$$||BD\mathbf{x}||_2 \approx ||\mathbf{x}||_2$$

Bounding the mapping

- Easy to see that

$$BD\mathbf{x} = \begin{bmatrix} B_{11}d_1x_1 + \ldots + B_{1d}d_dx_d \\ \vdots \\ B_{k1}d_1x_1 + \ldots + B_{kd}d_dx_d \end{bmatrix}$$

- Write as $BD\mathbf{x} = M\mathbf{z}$, where

$$M^{(i)} = x_i B^{(i)}$$

$$\mathbf{z} = \begin{bmatrix} d_1 & \ldots & d_d \end{bmatrix}^T$$

- There is a special name for a vector like $M\mathbf{z}$...

Rademacher series

Definition

If M is an arbitrary $k \times d$ real matrix, and $\mathbf{z} \in \mathbb{R}^d$ is so that

$$z_i = \begin{cases} +1 & p = 1/2 \\ -1 & p = 1/2 \end{cases}$$

then $M\mathbf{z}$ is called a *Rademacher random variable*. This is a vector whose entries are arbitrary sums/differences of each of the entires in rows of M.

- Such a variable is interesting because of a powerful theorem...

Talagrand's theorem

Theorem

Suppose $M, \mathbf{z}$ are as above. Let $Z = ||M\mathbf{z}||_p$, and let

$$\sigma = ||M||_{2\to p}$$

$$\mu = median(Z)$$

Then,

$$Pr[|Z - \mu| > t] \leq 4e^{-t^2/8\sigma^2}$$

(see [6])

- σ (the "deviation") is the maximal p-norm of all points on the unit circle
- Theorem says that the norm of a Rademacher variable is sharply concentrated about the median

Implications for us

- Our mapping, $BD\mathbf{x}$, has given us a Rademacher random variable
- We know that we can apply Talagrand's theorem to get a concentration result
- So, all we need to do is find out what the median and deviation are...

Deviation

- Let $Y = ||BD\mathbf{x}||_2 = ||M\mathbf{z}||_2$
- Deviation is

$$\begin{aligned}\sigma &= \sup_{||\mathbf{y}||_2=1} ||y^T M||_2 \\ &= \sup \left(\sum_{i=1}^{d} x_i^2 \left(y^T B^{(i)} \right)^2 \right)^{1/2} \\ &\leq ||\mathbf{x}||_4 \sup \left(\sum_{i=1}^{d} (y^T B^{(i)})^4 \right)^{1/4} \quad \text{by Cauchy-Schwartz} \\ &= ||\mathbf{x}||_4 ||B^T||_{2\to 4}\end{aligned}$$

What do we need?

- So, $\sigma \leq ||\mathbf{x}||_4 ||B^T||_{2\to 4}$
- **Fact**: $|1-\mu| \leq \sqrt{32}\sigma$
- Can combine to get

$$\Pr[|Y-1| > t] \leq c_0 e^{-c_1 t^2/(||\mathbf{x}||_4^2 ||B^T||_{2\to 4}^2)}$$

What do we need?

- So, $\sigma \leq ||\mathbf{x}||_4 ||B^T||_{2\to 4}$
- **Fact**: $|1-\mu| \leq \sqrt{32}\sigma$
- Can combine to get

$$\Pr\left[|Y-1| > t\right] \leq c_0 e^{-c_1 t^2 / (||\mathbf{x}||_4^2 ||B^T||_{2\to 4}^2)}$$

- **Result**: We need to control both $||\mathbf{x}||_4$ and $||B^T||_{2\to 4}$
  - i.e. we want them both to be small
  - If we manage this, we've got our concentration bound

The two ingredients

- To get the concentration bound, we need to ensure that $||\mathbf{x}||_4, ||B^T||_{2\to 4}$ are sufficiently small
- How to control $||\mathbf{x}||_4$?
- How to control $||B^T||_{2\to 4}$?

The two ingredients

- To get the concentration bound, we need to ensure that $||\mathbf{x}||_4, ||B^T||_{2\to4}$ are sufficiently small
- How to control $||\mathbf{x}||_4$?
  - Use repeated Fourier/Walsh-Hadamard transforms
- How to control $||B^T||_{2\to4}$?

The two ingredients

- To get the concentration bound, we need to ensure that $||\mathbf{x}||_4, ||B^T||_{2\to 4}$ are sufficiently small
- How to control $||\mathbf{x}||_4$?
    - Use repeated Fourier/Walsh-Hadamard transforms
- How to control $||B^T||_{2\to 4}$?
    - Use error correcting codes

Controlling $||\mathbf{x}||_4$

- **Problem**: Input $\mathbf{x}$ is "adversarial" - so how to make $||\mathbf{x}||_4$ small?

Controlling $||\mathbf{x}||_4$

- **Problem**: Input $\mathbf{x}$ is "adversarial" - so how to make $||\mathbf{x}||_4$ small?
- **Solution**: Use an isometric mapping Φ, with a guarantee that $||\Phi\mathbf{x}||_4$ is small with very high probability

Controlling $||\mathbf{x}||_4$

- **Problem**: Input $\mathbf{x}$ is "adversarial" - so how to make $||\mathbf{x}||_4$ small?
- **Solution**: Use an isometric mapping Φ, with a guarantee that $||\Phi\mathbf{x}||_4$ is small with very high probability
- **Problem**: What is such a Φ?

Controlling $||\mathbf{x}||_4$

- **Problem**: Input **x** is "adversarial" - so how to make $||\mathbf{x}||_4$ small?
- **Solution**: Use an isometric mapping Φ, with a guarantee that $||\Phi\mathbf{x}||_4$ is small with very high probability
- **Problem**: What is such a Φ?
- **(Final!) Solution**: Back to the Fourier transform!

The Discrete Fourier transform

- Discrete Fourier transform on $\{a_0, a_1, \ldots, a_{N-1}\}$ is

$$\begin{aligned} a_k &\mapsto \sum_{n=0}^{N-1} a_n e^{-2\pi ikn/N} \\ &= \sum_{n=0}^{N-1} a_n \left(e^{-2\pi ik/N}\right)^n \end{aligned}$$

- Can think of it as a polynomial evaluation - if

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{N-1} x^{N-1}$$

then we have

$$a_k \mapsto P\left(e^{-2\pi ik/N}\right)$$

The finite-field Fourier transform

- Notice that $\omega^k = e^{-2\pi ik/N} \neq 1$ satisfies $(\omega^k)^N = 1$
- ω^k is a *primitive root* of 1
- Transform is
$$a_k \mapsto P\left(\omega^k\right)$$
for any primitive root ω

The multi-dimensional Fourier transform

- We can also consider the transform of multi-dimensional data
- 1-D case:

$$a_k \mapsto \sum_{n=0}^{N-1} a_n \omega^{kn}$$

- υ-D case: If $\mathbf{n} = (n_1, \ldots, n_\upsilon)$,

$$a_{\mathbf{k}} \mapsto \sum_{n_1, \ldots, n_\upsilon = 0}^{N-1} a_{\mathbf{n}} \omega^{\mathbf{k}.\mathbf{n}}$$

The Walsh-Hadamard transform

- Consider the case $N = 2$, $\omega = -1$ [7]:

$$a_{k_1,k_2} \mapsto \sum_{n_1,n_2=0}^{1} a_{n_1,n_2}(-1)^{k_1 n_1 + k_2 n_2}$$

- This is called the *Walsh-Hadamard* transform
- **Intuition**: Instead of using sinusoidal basis functions, use square-wave functions
  - The square waves are called Walsh-functions
- Why not the standard discrete FT?
  - We use a technical property about the Walsh-Hadamard transform matrix...

Fourier transform on the binary hyper-cube

- Suppose we work with $\mathbb{F}_2 = \{0, 1\}$
- We can encode the Fourier transform with the *Walsh-Hadamard matrix* H_d,

$$H_d(i, j) = \frac{1}{2^{d/2}}(-1)^{<i-1, j-1>}$$

  where $< i, j >$ is the dot-product of i, j as expressed in binary
- **Fact**:

$$H_d = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{d/2} & H_{d/2} \\ H_{d/2} & -H_{d/2} \end{bmatrix}$$

- **Corollary**: We can compute H_d in $O(d \log d)$ time

Example of Hadamard matrix

- When $d = 4$, we get

$$H_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

- Note entries are always ± 1

Fourier again?

- Let $\Phi : \mathbf{x} \mapsto H_d D_0 \mathbf{x}$
    - D_0 as before a random diagonal ± 1 matrix
- Already know that it will preserve the ℓ_2 norm
- But is $||\Phi(\mathbf{x})||_4$ small?

Fourier again?

- Let $\Phi : \mathbf{x} \mapsto H_d D_0 \mathbf{x}$
  - D_0 as before a random diagonal ± 1 matrix
- Already know that it will preserve the ℓ_2 norm
- But is $||\Phi(\mathbf{x})||_4$ small?
- **Answer**: Yes - by another application of Talagrand's theorem!

Towards Talagrand

- Need σ, μ for Talagrand's theorem
- Write $\Phi(\mathbf{x}) = M\mathbf{z}$ as before, where $M^{(i)} = x_i H^{(i)}$
- Estimate deviation:

$$\begin{aligned}\sigma &= ||M||_{2\to 4} \\ &= ||M^T||_{\frac{4}{3}\to 2} \text{ (from earlier fact)} \\ &\leq \left(\sum x_i^4\right)^{1/4} \sup_{||\mathbf{y}||_{4/3}=1} \left(\sum \left(y^T H^{(i)}\right)^4\right)^{1/4} \\ &= ||\mathbf{x}||_4 ||H||_{\frac{4}{3}\to 4}\end{aligned}$$

Some magic

- We now employ the following theorem [4]

Theorem

Haussdorf-Young theorem. *For any $p \in [1,2]$, if H is the Hadamard matrix, and $\frac{1}{p} + \frac{1}{q} = 1$, then*

$$||H||_{p \to q} \leq \sqrt{d}.d^{-\frac{1}{p}}$$

- As a result, for $p = \frac{4}{3}$,

$$\sigma \leq ||\mathbf{x}||_4 d^{-1/4}$$

- Further, we have the following fact (see [3] for proof!)...
- **Fact**: $\mu = O\left(\frac{1}{d^{1/4}}\right)$

Getting the desired result

- With the above σ, μ, an application of Talagrand, along with the assumption $k = O(d^{1/2-\delta})$, reveals

$$||HD_0\mathbf{x}||_4 \leq c_0 d^{-1/4} + c_1 d^{-\delta/2} ||\mathbf{x}||_4$$

- If we compose the mapping,

$$||HD_1(HD_0\mathbf{x})|| \leq c_0 d^{-1/4} + c_0 c_1 d^{-1/4-\delta/2} + c_1^2 d^{-\delta} ||\mathbf{x}||_4$$

- If we repeat this $r = \frac{1}{2\delta}$ times,

$$||HD_{r-1}HD_{r-2}\dots HD_0\mathbf{x}||_4 = O\left(d^{-1/4}\right)$$

Our resultant transform

- To control $||\mathbf{x}||_4$, use the composed transform

$$\Phi^{(r)} : \mathbf{x} \mapsto HD_{r-1}HD_{r-2}\ldots HD_0\mathbf{x}$$

- We manage to preserve $||\mathbf{x}||_2$, and contract

$$||\Phi^r\mathbf{x}||_4$$

- Runtime is $O\left(\frac{d \log d}{\delta}\right)$

Error-correcting codes

- The Hadamard matrix also has a connection to *error-correcting codes*
- Such codes look to represent one's message in such a way that it can be decoded correctly even if there are some errors during transmission
- Suppose we want to send out a message to a decoder which allows for at most d errors
  - i.e. we can recover from d or less errors in the transmission
- **Fact**: By choosing our "code-words" from the matrix $\begin{bmatrix} H_{2d} \\ -H_{2d} \end{bmatrix}$, where $-1 \mapsto 0$, we can correct up to d errors

Code matrix

- An $m \times d$ matrix A is called a *code matrix* if

$$A = \sqrt{\frac{d}{m}} \begin{bmatrix} H_d(i_1, :) \\ H_d(i_2, :) \\ \vdots \\ H_d(i_m, :) \end{bmatrix}$$

- Picking out only m out of d rows of the Hadamard matrix

Independence in codes

- A code matrix is called *a*-wise independent if exactly $\frac{d}{2^a}$ columns agree in a places
- Independence is very useful for us:

Theorem

Suppose B is a $k \times d$, 4-wise independent code matrix. Then,

$$||B^T||_{2 \to 4} = O\left(\frac{d^{1/4}}{\sqrt{k}}\right)$$

Proof of theorem

- Recall that we need to bound

$$||B^T||_{2\to 4} = \sup_{||\mathbf{y}||_2=1} ||y^T B||_4$$

- Consider:

$$\begin{aligned} ||y^T B||_4^4 &= dE\left[(y^T B(j))^4\right] \\ &= \frac{d}{k^2}\sum_{i_1}\sum_{i_2}\sum_{i_3}\sum_{i_4} E\left[y_{i_1}y_{i_2}y_{i_3}y_{i_4}b_1b_2b_3b_4\right] \\ &= \frac{d}{k^2}(3||\mathbf{y}||_2^4 - 2||\mathbf{y}||_4^4) \\ &\le \frac{3d}{k^2} \end{aligned}$$

- Consequently,

$$||B^T||_{2\to 4} \le \frac{(3d)^{1/4}}{\sqrt{k}}$$

Making our matrix

- We're set if we get a $k \times d$, 4-wise independent code matrix
- **Problem**: How do we make such a matrix?

Making our matrix

- We're set if we get a $k \times d$, 4-wise independent code matrix
- **Problem**: How do we make such a matrix?
- **Fact**: There exists a 4-wise independent code matrix of size $k \times BCH(k) = \Theta(k^2)$
    - Called the *BCH* code matrix
- Which is good, because...

Making our matrix

- We're set if we get a $k \times d$, 4-wise independent code matrix
- **Problem**: How do we make such a matrix?
- **Fact**: There exists a 4-wise independent code matrix of size $k \times BCH(k) = \Theta(k^2)$
  - Called the *BCH* code matrix
- Which is good, because...
- **Fact**: By padding and "copy-pasting", we retain independence. In particular, we can construct a $k \times d$ matrix from a $k \times BCH(k)$ matrix:

$$B = \underbrace{\begin{bmatrix} B_{BCH} & B_{BCH} & \dots & B_{BCH} \end{bmatrix}}_{\frac{d}{BCH(k)} \text{ copies}}$$

Time to make matrix

- Time to compute the mapping $\mathbf{x} \mapsto B\mathbf{x}$?
- We have to do $\frac{d}{BCH(k)}$ mappings $B_{BCH}\mathbf{x}_{BCH}$
- Each such mapping can be done via a Walsh-Hadamard transform, by construction of BCH codes
    - Takes time $O(BCH(k). \log BCH(k))$
- Total runtime is therefore $O(d \log k)$

Merging results

- Use the randomized Fourier transform to keep $||\mathbf{x}||_4$ small
  - $O(d \log d)$ time
- Use the error-correcting code matrix to keep $||B||_{2\to4}$ small
  - $O(d \log k)$ time
- **Result**: We get the concentration bound!

Runtime

- Runtime is still going to be $O(d \log d)$
- **Question**: Can we speed up the computation of $\Phi^{(r)}$?

Runtime

- Runtime is still going to be $O(d \log d)$
- **Question**: Can we speed up the computation of $\Phi^{(r)}$?
- **Answer**: Yes - use the same "block" idea as with the error-correcting codes
  - Some rather technical calculation reveals this will still work

Blocked transform

- Choose $\beta = BCH(k).k^{\delta} = \Theta(k^{2+\delta})$
- Let
$$H = \begin{bmatrix} H_1 & & & \\ & H_2 & & \\ & & \ddots & \\ & & & H_{d/\beta} \end{bmatrix}$$
where each H_i is of size $\beta \times \beta$
- **Fact**: The above mapping can replace Φ^r
- The mapping $HD'\mathbf{x}$ can be computed in time $O(d \log k)$, so our total runtime is $O(d \log k)$

A tabular comparison

- Runtimes of the three approaches (standard JL, Fast JLT, and Ailon-Liberty) (from [3]):

$k = o(\log d)$	$k \in [\omega(\log d), o(\text{poly(d)})]$	$k \in [\Omega(\text{poly(d)}), o((d \log d)^{1/3})]$	$k \in [\omega((d \log d)^{1/3}), O(d^{1/2-\delta})]$
AL	AL	AL, FJLT	AL
JL	FJLT		FJLT
FJLT	JL	JL	JL

Conclusion

- ℓ_2 dimensionality reduction is based on the Johnson-Lindenstrauss lemma
- The standard approach takes $O(dk)$ time to perform the reduction
- By sparsifying, and compensating with a randomized Fourier transform, we can reduce the runtime to roughly $O(d \log d)$ via the Fast Johnson-Lindenstrauss transform [2]
- By using error-correcting codes and a randomized Fourier transform, we can reduce the runtime to roughly $O(d \log k)$ via Ailon and Liberty's transform [3]
- **Open questions**: Can one extend this to $k = O(d^{1-\delta})$? $k = \Omega(d)$?

Achlioptas, D.
Database-friendly random projections.
In *PODS '01: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (New York, NY, USA, 2001), ACM Press, pp. 274–281.

Ailon, N., and Chazelle, B.
Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform.
In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing* (New York, NY, USA, 2006), ACM Press, pp. 557–563.

Ailon, N., and Liberty, E.
Fast dimension reduction using Rademacher series on dual BCH codes.
Tech. Rep. TR07-070, Electronic Colloquium on Computational Complexity, 2007.

Bergh, J., and Lofstrom, J.
Interpolation Spaces.
Springer-Verlag, 1976.

JOHNSON, W., AND LINDENSTRAUSS, J.
Extensions of Lipschitz mappings into a Hilbert space.
In *Conference in Modern Analysis and Probability* (Providence, RI, USA, 1984), American Mathematical Society, pp. 189–206.

LEDOUX, M., AND TALAGRAND, M.
Probability in Banach Spaces: Isoperimetry and Processes.
Springer, 2006.

MASSEY, J. L.
Design and analysis of block ciphers.
`http://www.win.tue.nl/math/eidma/courses/minicourses/massey/dabcmay2000f3.pdf`, May 2000.
Presentation.