

Lecture 8: Randomized Linear Algebra Session 1

Instructors: *Alex Andoni, Edo Liberty*Scribes: *Kent Liu, Yuhui Li*

Motivation

Many core computations on high-dimensional data have costs that scale linearly with the dimension d . Two examples:

Definition 1 (Matrix Multiplication). *Given $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{m \times d}$, compute the product AB^T .*

Brute-force multiplication takes $\mathcal{O}(ndm)$ operations. In practice, d can be around 10^4 to 10^7 and m around 10^5 , giving on the order of 10^{16} operations.

Definition 2 (Vector Search). *Given a dataset $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ and a query $q \in \mathbb{R}^d$, compute*

$$\min_{x \in X} \|q - x\|_2$$

A linear scan computes all n distances in $\mathcal{O}(nd)$ time.

In both cases, the cost is driven by d . If we could replace each vector with a much shorter one while approximately preserving distances, every such computation would become cheaper. This is the goal of random projection.

Random Projection & The Johnson-Lindenstrauss Lemma

Core Idea

Definition 3 (Random Projection). *The goal of random projection is to map a set of high-dimensional vectors to a much lower-dimensional space while preserving the pairwise Euclidean distances between them. Formally, we define the mapping:*

$$x_1, x_2, \dots, x_n \longrightarrow y_1, y_2, \dots, y_n$$

where the original vectors $x_i \in \mathbb{R}^d$ are projected to $y_i \in \mathbb{R}^k$ such that the target dimension is significantly smaller than the original ($k \ll d$).

Distance Preservation

To preserve as much information as possible, we require the Euclidean distance in the lower-dimensional space to approximate the original distance within an error margin of $\epsilon \in (0, 1)$:

$$\|y_i - y_j\|_2 \approx \|x_i - x_j\|_2 \quad \forall i, j$$

Theorem 4 (Johnson-Lindenstrauss (JL) Lemma [4]). *There exists a mapping $x_i \mapsto y_i$ satisfying*

$$(1 - \epsilon)\|x_i - x_j\|_2^2 \leq \|y_i - y_j\|_2^2 \leq (1 + \epsilon)\|x_i - x_j\|_2^2 \quad \forall i, j$$

if the target dimension k satisfies:

$$k \geq c \cdot \frac{\log n}{\epsilon^2}$$

where c is a positive constant.

Notably, the required lower bound for k depends entirely on the number of points n and the error tolerance ϵ , remaining completely independent of the original dimension d . The proof is given in a later section.

Almost-Orthogonal Vectors in High Dimensions

Consider the geometric properties of the space:

Fact 5 (Exactly Orthogonal). *In $\mathbb{R}^d$, the maximum number of mutually perpendicular vectors (where $\langle x_i, x_j \rangle = 0$ for $i \neq j$) is exactly d .*

Observation 6 (Almost Orthogonal). *If we allow a small inner-product slackness such that $|\langle x_i, x_j \rangle| \leq \epsilon$, high-dimensional spaces can accommodate an exponentially large number of vectors relative to k .*

Example 7 (One-Hot Encoded Standard Basis). *Let $x_i, x_j \in \mathbb{R}^n$ be distinct standard unit basis vectors (one-hot encodings for coordinates i and j). The squared distance between them is:*

$$\|x_i - x_j\|_2^2 = \|x_i\|_2^2 - 2\langle x_i, x_j \rangle + \|x_j\|_2^2 = 1 - 0 + 1 = 2$$

Thus, $\|x_i - x_j\|_2 = \sqrt{2}$. Applying a stable JL projection implies:

$$\|y_i - y_j\|_2^2 \in [2(1 - \epsilon), 2(1 + \epsilon)] \implies |\langle y_i, y_j \rangle| \leq \mathcal{O}(\epsilon)$$

Corollary 8. *Rearranging the JL bound in terms of n demonstrates this exponential capacity:*

$$n = e^{\frac{k \cdot \epsilon^2}{c}}$$

Distributional Mechanics of Random Projection

To construct this mapping, we use properties of the Normal distribution.

Fact 9 (Stability of Gaussians). *If $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ are independent, then:*

$$X_1 + X_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

Lemma 10. *Let $w = [w_1, w_2, \dots, w_d]^T$ be a random vector where each component is drawn from a standard normal distribution, $w_i \sim \mathcal{N}(0, 1)$. Let $u \in \mathbb{R}^d$ be a fixed unit vector such that $\|u\|_2^2 = 1$. Then $w^T u \sim \mathcal{N}(0, 1)$.*

Proof. We define the scaled components $z_i = w_i u_i$. By scaling rules, $z_i \sim \mathcal{N}(0, u_i^2)$. Projecting u onto our random vector yields:

$$w^T u = \sum_{i=1}^d w_i u_i = \sum_{i=1}^d z_i$$

Using the Gaussian summation property:

$$w^T u \sim \mathcal{N}\left(0, \sum_{i=1}^d u_i^2\right) = \mathcal{N}(0, \|u\|_2^2) = \mathcal{N}(0, 1)$$

□

The Projection Matrix

Definition 11 (Random Projection Matrix). Let $R \in \mathbb{R}^{k \times d}$ be a random matrix whose rows are independent random vectors $w_1, \dots, w_k$ (so each entry $R_{ij} \sim \mathcal{N}(0, 1)$). The projection is R scaled by $1/\sqrt{k}$:

$$\frac{1}{\sqrt{k}} R = \frac{1}{\sqrt{k}} \begin{bmatrix} \text{---} & w_1^T & \text{---} \\ \text{---} & w_2^T & \text{---} \\ & \vdots & \\ \text{---} & w_k^T & \text{---} \end{bmatrix}$$

Claim 12. When we map the vector u into the lower-dimensional space, the resulting projection vector $z \in \mathbb{R}^k$ is:

$$z = \frac{1}{\sqrt{k}} R u$$

and every element of z is distributed as:

$$z_i \sim \mathcal{N}\left(0, \frac{1}{k}\right)$$

Proof. Each row behaves independently as $w_i^T u \sim \mathcal{N}(0, 1)$ by Lemma 10; scaling by $1/\sqrt{k}$ gives variance $1/k$. □

Length Preservation of the Random Projection

Squared Length of the Projected Vector

Let $u \in \mathbb{R}^d$ be a unit vector and let $y = \frac{1}{\sqrt{k}} R u$. By the claim above, each coordinate $g_i := (R u)_i = w_i^T u$ is distributed as $\mathcal{N}(0, 1)$, and the g_i are independent because the rows of R are. Then

$$\|y\|_2^2 = \frac{1}{k} \sum_{i=1}^k g_i^2.$$

Since $\mathbb{E}[g_i] = 0$ and $\text{Var}(g_i) = 1$, we have $\mathbb{E}[g_i^2] = \text{Var}(g_i) + \mathbb{E}[g_i]^2 = 1$. Hence $\mathbb{E}\|y\|_2^2 = 1 = \|u\|_2^2$: in expectation, the projection preserves squared length.

A single term g_i^2 may be smaller or larger than 1, but when we average k independent copies, the average concentrates around its mean as k grows. The sum $\sum_{i=1}^k g_i^2$ is known as a *chi-squared distribution with k degrees of freedom*.

Fact 13 (Concentration of Chi-Squared). *For $\epsilon \in (0, 1)$,*

$$\Pr \left[\left| \frac{1}{k} \sum_{i=1}^k g_i^2 - 1 \right| > \epsilon \right] \leq 2e^{-k\epsilon^2/8}.$$

Since $\frac{1}{k} \sum_i g_i^2 = \|y\|_2^2$, this is the event $|\|y\|_2^2 - \|u\|_2^2| > \epsilon$. The probability of deviating from the mean by more than ϵ decreases *exponentially* in k .

Extending to Arbitrary Vectors and a Single Pair

Define the mapping $x \mapsto y = \frac{1}{\sqrt{k}} Rx$. For any nonzero x , write $x = \|x\|_2 \hat{x}$ with $\hat{x}$ a unit vector. By linearity, $y = \|x\|_2 \cdot \frac{1}{\sqrt{k}} R\hat{x}$, so

$$\|y\|_2^2 = \|x\|_2^2 \cdot \left\| \frac{1}{\sqrt{k}} R\hat{x} \right\|_2^2 \in (1 \pm \epsilon) \|x\|_2^2$$

whenever the event of Fact 13 does not occur for $\hat{x}$.

What we actually care about is the difference $x_i - x_j$. Take $u = \frac{x_i - x_j}{\|x_i - x_j\|_2}$. Then with probability at least $1 - 2e^{-k\epsilon^2/8}$,

$$1 - \epsilon \leq \left\| \frac{1}{\sqrt{k}} R \frac{x_i - x_j}{\|x_i - x_j\|_2} \right\|_2^2 \leq 1 + \epsilon.$$

Let δ denote the allowed failure probability, i.e. we require $2e^{-k\epsilon^2/8} \leq \delta$. Solving for k :

$$k \geq \frac{8}{\epsilon^2} \log \frac{2}{\delta}.$$

Multiplying the inequality above by $\|x_i - x_j\|_2^2$ and using linearity of R , the middle term becomes $\|y_i - y_j\|_2^2$:

$$(1 - \epsilon) \|x_i - x_j\|_2^2 \leq \|y_i - y_j\|_2^2 \leq (1 + \epsilon) \|x_i - x_j\|_2^2.$$

This holds for one fixed pair of points. The question is how small δ must be so that it holds for *all* pairs simultaneously.

The Union Bound and Proof of JL

Lemma 14 (Union Bound). *Let $A_1, \dots, A_n$ be probabilistic events and $p_i = \Pr[A_i]$. Then*

$$\Pr \left[\bigcup_i A_i \right] \leq \sum_i p_i.$$

This is an intuitive but very useful property: it requires no independence between the events.

Let f_{ij} be the event that the random projection fails for the pair x_i, x_j , with $\Pr[f_{ij}] \leq \delta$. There are $\binom{n}{2} < n^2$ pairs, so

$$\Pr \left[\bigcup_{i < j} f_{ij} \right] \leq \sum_{i < j} \delta < n^2 \delta.$$

Suppose we want all pairs to be preserved with probability at least $1/2$, i.e. $\Pr[\bigcup f_{ij}] \leq \frac{1}{2}$. It suffices that $n^2 \delta \leq \frac{1}{2}$, i.e. $\delta = \frac{1}{2n^2}$. Plugging into $k \geq \frac{8}{\epsilon^2} \log \frac{2}{\delta}$, and using $\frac{2}{\delta} = 4n^2 = (2n)^2$:

$$k \geq \frac{16}{\epsilon^2} \log(2n).$$

With this choice of k , the random map $x \mapsto \frac{1}{\sqrt{k}}Rx$ preserves all pairwise squared distances up to a factor $1 \pm \epsilon$ with probability at least $1/2$, so in particular such a map exists. This proves Theorem 4, with an explicit constant (the c in the theorem can be taken to be on the order of 16, absorbing $\log 2n \lesssim \log n$ for $n \geq 2$).

Application: Speeding Up Matrix Multiplication

Return to AB^T with $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{m \times d}$, which costs $\mathcal{O}(ndm)$ by brute force. Let $\Pi = \frac{1}{\sqrt{k}}R \in \mathbb{R}^{k \times d}$ be a random projection matrix and compute

$$(A\Pi^T)(B\Pi^T)^T,$$

where $A\Pi^T \in \mathbb{R}^{n \times k}$ and $(B\Pi^T)^T \in \mathbb{R}^{k \times m}$. The cost is now

$$\mathcal{O}(ndk + mdk) + \mathcal{O}(nmk),$$

where the first term is the cost of projecting A and B , and the second is the cost of the final product. Since $k \ll d$, this is much cheaper than $\mathcal{O}(ndm)$.

Application: Speeding Up Vector Search

Furthermore, we revisit vector search over $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$, with Π as above. As a one-time preprocessing step, we can project the whole dataset, $y_i = \Pi x_i$, in $\mathcal{O}(ndk)$ time. For each query q , compute Πq in $\mathcal{O}(dk)$ time and scan the projected points, returning $\hat{x} = x_{\hat{i}}$ where $\hat{i} = \arg \min_i \|\Pi q - y_i\|_2$, in $\mathcal{O}(nk)$ time. The per-query cost drops from $\mathcal{O}(nd)$ to $\mathcal{O}(dk + nk)$.

The tradeoff is that we only return an *approximate* nearest neighbor. For a fixed query, apply Fact 13 with a union bound to the n difference vectors $q - x_i$: with $k = \mathcal{O}(\log n / \epsilon^2)$, all of their squared lengths are preserved up to a factor $1 \pm \epsilon$ with constant probability. If x^* is the true nearest neighbor, then since $\hat{x}$ minimizes the projected distance,

$$\|q - \hat{x}\|_2^2 \leq \frac{\|\Pi(q - \hat{x})\|_2^2}{1 - \epsilon} \leq \frac{\|\Pi(q - x^*)\|_2^2}{1 - \epsilon} \leq \frac{1 + \epsilon}{1 - \epsilon} \|q - x^*\|_2^2,$$

so $\hat{x}$ is within a factor $\frac{1+\epsilon}{1-\epsilon}$ of the optimal squared distance.

Faster Projections: Fast JL

Multiplying by a dense Gaussian matrix still costs $\mathcal{O}(kd)$ per vector, and storing it takes $\mathcal{O}(kd)$ space. Can we find a matrix that is “random enough” to have the same properties, but can be applied faster? The idea is to use

$$\Pi = PH_d D,$$

where

- D is a random diagonal matrix with independent ± 1 entries,
- H_d is the (normalized) Hadamard matrix, defined recursively for d a power of 2 by

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

- P is a matrix that samples k of the d coordinates (with an appropriate rescaling so that squared length is preserved in expectation).

To compute $\Pi x = PH_d D x$: applying D takes $\mathcal{O}(d)$ time, and applying H_d takes $\mathcal{O}(d \log d)$ time using its recursive structure. Sampling coordinates is cheap. Thus the projection time improves from $\mathcal{O}(kdn)$ to $\mathcal{O}(nd \log d)$, and the space for storing Π improves from $\mathcal{O}(kd)$ to $\mathcal{O}(d)$. This is known as the *Fast JL transform* [2].

Supplementary Material

The following sections were not covered in lecture. They fill in results that were stated without proof above.

Proof of the Chi-Squared Tail Bound

We prove Fact 13 by the Chernoff method. Let $g_1, \dots, g_k \sim \mathcal{N}(0, 1)$ be independent and $S = \sum_i g_i^2$.

Lemma 15 (MGF of a Squared Gaussian). *For $g \sim \mathcal{N}(0, 1)$ and $\lambda < \frac{1}{2}$,*

$$\mathbb{E}[e^{\lambda g^2}] = (1 - 2\lambda)^{-1/2}.$$

Proof.

$$\mathbb{E}[e^{\lambda g^2}] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\lambda x^2} e^{-x^2/2} dx = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-(1-2\lambda)x^2/2} dx = (1 - 2\lambda)^{-1/2},$$

since the last integrand is an unnormalized Gaussian density with variance $\frac{1}{1-2\lambda}$. □

By independence, $\mathbb{E}[e^{\lambda S}] = (1 - 2\lambda)^{-k/2}$.

Upper tail. For $0 < \lambda < \frac{1}{2}$, Markov's inequality applied to $e^{\lambda S}$ gives

$$\Pr[S \geq k(1 + \epsilon)] \leq e^{-\lambda k(1 + \epsilon)} \mathbb{E}[e^{\lambda S}] = e^{-\lambda k(1 + \epsilon)} (1 - 2\lambda)^{-k/2}.$$

Choose $\lambda = \frac{\epsilon}{2(1 + \epsilon)}$, so that $1 - 2\lambda = \frac{1}{1 + \epsilon}$. Then

$$\Pr[S \geq k(1 + \epsilon)] \leq e^{-k\epsilon/2} (1 + \epsilon)^{k/2} = \exp\left(-\frac{k}{2}(\epsilon - \ln(1 + \epsilon))\right).$$

For $\epsilon \in (0, 1)$ we have $\epsilon - \ln(1 + \epsilon) \geq \frac{\epsilon^2}{4}$: the difference $h(\epsilon) = \epsilon - \ln(1 + \epsilon) - \frac{\epsilon^2}{4}$ satisfies $h(0) = 0$ and $h'(\epsilon) = \frac{\epsilon(1 - \epsilon)}{2(1 + \epsilon)} \geq 0$. Hence

$$\Pr[S \geq k(1 + \epsilon)] \leq e^{-k\epsilon^2/8}.$$

Lower tail. For $\lambda > 0$, apply Markov to $e^{-\lambda S}$:

$$\Pr[S \leq k(1 - \epsilon)] \leq e^{\lambda k(1 - \epsilon)} \mathbb{E}[e^{-\lambda S}] = e^{\lambda k(1 - \epsilon)} (1 + 2\lambda)^{-k/2}.$$

Choose $\lambda = \frac{\epsilon}{2(1 - \epsilon)}$, so that $1 + 2\lambda = \frac{1}{1 - \epsilon}$. Then

$$\Pr[S \leq k(1 - \epsilon)] \leq \exp\left(\frac{k}{2}(\epsilon + \ln(1 - \epsilon))\right) \leq \exp\left(-\frac{k\epsilon^2}{4}\right) \leq e^{-k\epsilon^2/8},$$

using $\ln(1 - \epsilon) \leq -\epsilon - \frac{\epsilon^2}{2}$.

Adding the two tails (a union bound) gives $\Pr\left[\left|\frac{1}{k}S - 1\right| > \epsilon\right] \leq 2e^{-k\epsilon^2/8}$, which is Fact 13.

Other Random Projection Matrices

Nothing in the argument above uses the Gaussian distribution beyond two properties of each entry R_{ij} : it has mean 0 and variance 1, and its tail is no heavier than a Gaussian's. This allows cheaper choices of R :

- **Rademacher entries.** Take $R_{ij} = \pm 1$ with probability $\frac{1}{2}$ each. Then $\mathbb{E}[R_{ij}] = 0$ and $\text{Var}(R_{ij}) = 1$, exactly as for $\mathcal{N}(0, 1)$. For a unit vector u , $w^T u = \sum_i \sigma_i u_i$ has mean 0 and variance $\sum_i u_i^2 = 1$, so $\mathbb{E}\|y\|_2^2 = 1$ still holds. It is no longer exactly Gaussian, but its tails are at least as light, so the same exponential concentration holds. The matrix is cheaper to generate and store, and the product needs only additions and subtractions.
- **Sparse entries.** Take $R_{ij} = \sqrt{3}, 0, -\sqrt{3}$ with probabilities $\frac{1}{6}, \frac{2}{3}, \frac{1}{6}$ (Achlioptas [1]). The variance is again $3 \cdot \frac{1}{3} = 1$, but two thirds of the entries are zero, so the projection is about three times cheaper. More aggressive sparse constructions (e.g. Kane–Nelson [5]) keep only a few nonzeros per column.

Additional Chapter: Accuracy of Randomized Matrix Multiplication

In lecture we saw that $(A\Pi^T)(B\Pi^T)^T$ is cheap to compute, but not that it is close to AB^T . We prove that it is, then compare random projection with random sampling, which explains why Fast JL has the form $\Pi = PH_d D$.

Error of Random Projection

Let $a_1, \dots, a_n$ be the rows of A and $b_1, \dots, b_m$ the rows of B , and let $\Pi = \frac{1}{\sqrt{k}}R$ as before. The two matrices we are comparing have entries

$$(AB^T)_{ij} = \langle a_i, b_j \rangle, \quad (A\Pi^T \Pi B^T)_{ij} = \langle \Pi a_i, \Pi b_j \rangle,$$

so the question is whether random projection preserves inner products. We measure total error with the Frobenius norm $\|M\|_F^2 = \sum_{i,j} M_{ij}^2$, which simply adds up the squared errors of all nm inner products (note $\|A\|_F^2 = \sum_i \|a_i\|_2^2$). A small Frobenius error means the inner products are accurate on average. Note, however, that the bound below measures the error against the sizes of A and B , not the size of the product itself, so if the true product AB^T is tiny, the guarantee does not tell us much.

Lemma 16 (A Single Entry). *For fixed $a, b \in \mathbb{R}^d$,*

$$\mathbb{E}[\langle \Pi a, \Pi b \rangle] = \langle a, b \rangle, \quad \text{Var}(\langle \Pi a, \Pi b \rangle) = \frac{1}{k} (\|a\|_2^2 \|b\|_2^2 + \langle a, b \rangle^2).$$

Proof. Write $\langle \Pi a, \Pi b \rangle = \frac{1}{k} \sum_{r=1}^k U_r V_r$ with $U_r = w_r^T a$ and $V_r = w_r^T b$. As in Lemma 10, the pairs (U_r, V_r) are independent across rows, and each pair is jointly Gaussian with mean zero and

$$\mathbb{E}[U^2] = \|a\|_2^2, \quad \mathbb{E}[V^2] = \|b\|_2^2, \quad \mathbb{E}[UV] = \sum_{s,t} a_s b_t \mathbb{E}[w_s w_t] = \langle a, b \rangle.$$

This gives the mean. For the variance, a standard identity for jointly Gaussian variables gives $\mathbb{E}[U^2 V^2] = \mathbb{E}[U^2] \mathbb{E}[V^2] + 2\mathbb{E}[UV]^2$, so $\text{Var}(UV) = \|a\|_2^2 \|b\|_2^2 + \langle a, b \rangle^2$. Averaging k independent copies divides the variance by k . $\square$

Theorem 17 (Approximate Matrix Multiplication [6]).

$$\mathbb{E} [\|A\Pi^T\Pi B^T - AB^T\|_F^2] \leq \frac{2}{k} \|A\|_F^2 \|B\|_F^2.$$

Consequently, if $k \geq \frac{2}{\epsilon^2 \delta}$, then with probability at least $1 - \delta$,

$$\|A\Pi^T\Pi B^T - AB^T\|_F \leq \epsilon \|A\|_F \|B\|_F.$$

Proof. Each entry is unbiased by Lemma 16, so its expected squared error equals its variance. By linearity of expectation (which holds even though all entries share Π), the expected squared Frobenius error is the sum of these per-entry variances. Summing and using Cauchy-Schwarz, $\langle a_i, b_j \rangle^2 \leq \|a_i\|_2^2 \|b_j\|_2^2$:

$$\mathbb{E} [\|A\Pi^T\Pi B^T - AB^T\|_F^2] = \frac{1}{k} \sum_{i,j} (\|a_i\|_2^2 \|b_j\|_2^2 + \langle a_i, b_j \rangle^2) \leq \frac{2}{k} \sum_{i,j} \|a_i\|_2^2 \|b_j\|_2^2 = \frac{2}{k} \|A\|_F^2 \|B\|_F^2.$$

For the second part, Markov's inequality applied to the squared error gives failure probability at most $\frac{2}{k\epsilon^2} \leq \delta$. $\square$

Why not just use JL? We could instead require every inner product to be accurate. By the polarization identity, $\langle \Pi a, \Pi b \rangle = \frac{1}{4} (\|\Pi(a+b)\|_2^2 - \|\Pi(a-b)\|_2^2)$, preserving norms preserves inner products, but a union bound over all nm entries would need $k = \mathcal{O}(\log(nm)/\epsilon^2)$. Theorem 17 only asks for the **total** error to be small, which lets errors in different entries average out. As a result, k depends on neither n , m , nor d . Take the following example: $\epsilon = \delta = 0.1$ gives $k = 2000$ whether $d = 10^4$ or $d = 10^7$.

Random Sampling as an alternative

The product is a sum of d outer products of columns, $AB^T = \sum_{t=1}^d A^{(t)} B^{(t)T}$, where $A^{(t)}$ and $B^{(t)}$ denote the t -th columns. Instead of projecting, we can keep only k of these terms (Drineas, Kannan, and Mahoney [3]). Fix probabilities $p_1, \dots, p_d$, draw indices $t_1, \dots, t_k$ independently with $\Pr[t_s = t] = p_t$, and output

$$M = \frac{1}{k} \sum_{s=1}^k \frac{A^{(t_s)} B^{(t_s)T}}{p_{t_s}}.$$

This skips the projection step entirely and costs only $\mathcal{O}(nmk)$. Forming M amounts to selecting and rescaling k columns of A and B in $\mathcal{O}((n+m)k)$ time, then multiplying the resulting $n \times k$ and $k \times m$ matrices in $\mathcal{O}(nmk)$ time. In contrast, random projection must compute $A\Pi^T$ and $B\Pi^T$ which takes $\mathcal{O}((n+m)kd)$.

Lemma 18 (Error of Sampling). *Let $c_t = \|A^{(t)}\|_2 \|B^{(t)}\|_2$. Then $\mathbb{E}[M] = AB^T$ and*

$$\mathbb{E} [\|M - AB^T\|_F^2] = \frac{1}{k} \left(\sum_{t=1}^d \frac{c_t^2}{p_t} - \|AB^T\|_F^2 \right).$$

Proof. Let $Y = A^{(t)} B^{(t)T} / p_t$ for a single random index t . Then $\mathbb{E}[Y] = \sum_t p_t \cdot A^{(t)} B^{(t)T} / p_t = AB^T$. Since $\|uv^T\|_F = \|u\|_2 \|v\|_2$, we get $\mathbb{E}\|Y\|_F^2 = \sum_t c_t^2 / p_t$, so $\mathbb{E}\|Y - AB^T\|_F^2 = \sum_t c_t^2 / p_t - \|AB^T\|_F^2$. Averaging k independent copies divides this by k . $\square$

Choosing the probabilities. By Cauchy–Schwarz, for any valid probabilities,

$$\left(\sum_t c_t\right)^2 = \left(\sum_t \frac{c_t}{\sqrt{p_t}} \cdot \sqrt{p_t}\right)^2 \leq \sum_t \frac{c_t^2}{p_t},$$

with equality when $p_t \propto c_t$. Applying Cauchy–Schwarz again, $\sum_t c_t \leq \|A\|_F \|B\|_F$. So with $p_t \propto c_t$ the error is at most $\frac{1}{k} \|A\|_F^2 \|B\|_F^2$, matching random projection.

Why not sample uniformly? The good probabilities depend on the data, so computing them requires a pass over A and B . Uniform sampling ($p_t = 1/d$) avoids this but can fail badly. If all of the mass sits in one column, then $c_1 = \|A\|_F \|B\|_F$, $c_t = 0$ for $t > 1$, and $\|AB^T\|_F^2 = c_1^2$, so Lemma 18 gives error $\frac{d-1}{k} \|A\|_F^2 \|B\|_F^2$. We would need k to have a lower bound in terms of d which defeats the purpose.

The connection to Fast JL. This is exactly the problem $H_d D$ solves. Since $Q = H_d D$ is orthogonal ($Q^T Q = I$),

$$AB^T = (AQ^T)(BQ^T)^T,$$

so the rotation does not change the product. It does change the columns: with high probability, it spreads each row’s mass evenly across all d coordinates, so every column of AQ^T carries roughly a $1/d$ fraction of $\|A\|_F^2$ (up to logarithmic factors), and similarly for B . Once the columns are balanced, uniform sampling, which is what P does, is nearly as good as the optimal probabilities. So $\Pi = PH_d D$ can be read as: $H_d D$ spreads out the mass, and P samples uniformly. This also explains the trade-off between the two approaches. Optimal sampling is cheaper, but must see the data first. Random projection is oblivious: Π can be chosen before A and B are known.

References

- [1] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.
- [2] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of STOC*, 2006.
- [3] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.
- [4] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.
- [5] D. M. Kane and J. Nelson. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM*, 61(1):4:1–4:23, 2014.
- [6] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of FOCS*, 2006.