Lecture 12: Applications of JL & Subspace Embedding

10-02-2025 Lecturer: Ali Vakilian | Scribe: Asha Barua | Editor: Ali Vakilian

Motivation: The subspace embedding and sketching approach provides a framework for developing faster algorithms that offer approximate solutions to various matrix-related problems such as Matrix Multiplication, Linear Regression, and Low-Rank approximation.

Basic Idea: We consider the setting where we want to perform computations on a large matrix A that has n data columns in a high-dimensional space $\mathbb{R}^h$, but whose actual rank d is much smaller. The main objective is to reduce the computation to an equivalent problem involving a matrix of size roughly $\mathbb{R}^{d\times d}$ by spending time proportional to the number of non-zero entries in A.

In particular, to achieve this, we need to find a suitable projection or sketch of the data points into a lower-dimensional space such that the important geometric properties of the original high-dimensional space hold. Once such a projection is applied, the resulting lower-dimensional representation contains enough information to allow us to solve the underlying problem efficiently without significant loss of accuracy.

Next, we will discuss regression, one of the classical problems in data analysis and machine learning.

1 Regression: Linear Model Fitting

Given a collection of data points $a_1, \ldots, a_n \in \mathbb{R}^d$ together with their corresponding responses $b_1, \ldots, b_n \in \mathbb{R}$, our goal is to find a linear model that best explains the observed data. Formally, we want to find parameters $w_0, w_1, \ldots, w_d$ such that

$$b_i = w_0 + \sum_{j=1}^{d} w_j a_{i,j}, \qquad \text{for all } i = 1, \ldots, n.$$

The vector $w = (w_1, \ldots, w_d)$ represents the weights of the model, and w_0 is the intercept term. In practice, perfect equality is rarely possible due to noise or measurement error in the data.

Without loss of generality, the intercept can be absorbed into the feature representation by appending an additional coordinate of 1 to each data vector, allowing us to set $w_0 = 0$ in the augmented space of dimension $d+1$.

Let $A \in \mathbb{R}^{n\times d}$ be the matrix whose i-th row is $a_i^\top$, and let $b \in \mathbb{R}^n$ be the vector of responses. The regression problem can then be written compactly as

$$\min_{x\in\mathbb{R}^d} \|Ax - b\|,$$

where the choice of norm (e.g., ℓ_1, ℓ_2, or ℓ_∞) specifies the loss function used to measure the quality of fit. In most applications, the ℓ_2 norm—leading to the least-squares regression problem which is the most common and mathematically convenient choice.

1.1 Least-Squares Regression

In the least-squares formulation, we adopt the ℓ_2 norm as our measure of error. Given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, the objective is to find

$$\min_{x\in\mathbb{R}^d} \|Ax - b\|_2.$$

When the number of data points n is much larger than the number of features d, that is $n >> d$, the linear system $Ax = b$ generally has no exact solution. In this case, we seek the best fit such as a vector Ax lying in the column space of A that is as close as possible to b in the Euclidean space (ℓ_2-norm) .

Since every vector Ax is a linear combination of the columns of A, the goal is to find a vector $z \in \text{colspace}(A)$ that minimizes $\|z - b\|_2$. Geometrically, this vector z is the projection of b onto the column space of A.

Normal Equations, Singular Value Decomposition, and the Pseudoinverse: As discussed earlier, the closest vector to b within the column space of A is its projection onto that subspace. Let $\{z_1, z_2, \ldots, z_r\}$ be an orthonormal basis for the columns of A. Then the projection c of b to colspace(A) is given by

$$c = \sum_{1 \le j \le r} \langle b, z_j \rangle z_j.$$

The optimal vector x^* must satisfy $Ax^* = c$, meaning that x^* is the solution to the linear system whose right-hand side is this projected vector.

To compute x^* directly, we use the normal equation:

$$A^\top Ax = A^\top b.$$

If the columns of A are linearly independent, then $A^\top A$ is full rank and invertible, and hence

$$x^* = (A^\top A)^{-1} A^\top b.$$

When the columns of A are not linearly independent, the system may have infinitely many solutions. In that case, the unique solution with the smallest Euclidean norm is given by the Moore-Penrose pseudoinverse:

$$x^* = A^+ b,$$

where $A^+ = (A^\top A)^{-1} A^\top$ when A has full column rank, and more generally A^+ is defined via the singular value decomposition of A.

To further understand the structure of the least-squares solution and the role of the pseudoinverse, it is helpful to explore the singular value decomposition (SVD) of a matrix. For every matrix $A \in \mathbb{R}^{m \times d}$, there exists matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{d \times d}$ with orthonormal columns such that

$$A = U \Sigma V^\top,$$

where $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$ is a diagonal matrix containing the singular values $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_r > 0$. The Moore-Penrose pseudoinverse of A is defined using this decomposition as

$$A^+ = V \Sigma^{-1} U^\top,$$

where

$$\Sigma^{-1} = \text{diag}\left(\tfrac{1}{\sigma_1}, \ldots, \tfrac{1}{\sigma_r}, 0, \ldots, 0\right).$$

The pseudoinverse satisfies the following fundamental algebraic properties:

$$AA^+A = A, \qquad (AA^+)^\top = AA^+.$$

When A has linearly independent columns (i.e., full column rank), the pseudoinverse reduces to

$$A^+ = (A^\top A)^{-1} A^\top.$$

The product AA^+ is the projection matrix in the column space of A.

Let $P = AA^+$. Then $Pv = A(A^+)v$, which is a linear combination of columns in A. Moreover, for any vector y, assume that $y \in \text{colspan}(A)$ (i.e., $y = Ax$ for some x), then we have

$$Py = AA^+(Ax) = Ax = y,$$

showing that P leaves all vectors in $\text{col}(A)$ unchanged while projecting any general vector in $\mathbb{R}^m$ onto that subspace.

However, computing x^* exactly following these methods typically requires $\mathcal{O}(nd^2)$ time, which becomes expensive when n is large. Perhaps, can we obtain an approximate solution much faster by using a lower-dimensional representation that still preserves the essential structure of the problem?

To answer this question, we now introduce the subspace embeddings which allow us to solve such regression problems efficiently while maintaining strong approximation guarantees.

1.2 Least Squares Regression via Subspace Embedding

We now explore how least squares regression can be made more efficient through the use of subspace embeddings. Let E denote the subspace spanned by the columns of A together with the vector b. This subspace has dimension at most $d + 1$. The goal is to reduce the regression problem to a much smaller equivalent problem that can be solved faster while preserving its geometric structure.

We apply Subspace Embedding Π on E with $k = \mathcal{O}(d/\varepsilon^2)$ rows to reduce

$$\{A^{(1)}, A^{(2)}, \ldots, A^{(d)}, b\} \quad \text{to} \quad \{A'^{(1)}, A'^{(2)}, \ldots, A'^{(d)}, b'\},$$

which are in $\mathbb{R}^k$.

After projection, we solve the reduced least squares problem:

$$\min_{x' \in \mathbb{R}^d} \|A'x' - b'\|_2.$$

Intuitively, (A', b') are lower-dimensional representations of (A, b) that still capture the essential informations required for accurate regression.

Lemma 1.1. *(Subspace Embedding Guarantee) With probability at least $1 - \delta$, the embedding Π preserves the regression cost up to a $(1 \pm \varepsilon)$ factor:*

$$(1 - \varepsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2 \;\le\; \min_{x' \in \mathbb{R}^d} \|A'x' - b'\|_2 \;\le\; (1 + \varepsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2.$$

Proof of Lemma 1.1. With probability at least $(1 - \delta)$, the subspace embedding guarantee ensures that for all vectors $z \in E$,

$$(1 - \varepsilon)\|z\|_2 \le \|\Pi z\|_2 \le (1 + \varepsilon)\|z\|_2.$$

Let x^* and y^* denote the optimal solutions to

$$x^* = \arg\min_{x \in \mathbb{R}^d} \|Ax - b\|_2, \qquad y^* = \arg\min_{x' \in \mathbb{R}^d} \|A'x' - b'\|_2.$$

Let $z = Ax^* - b$. Since $z \in E$, it follows that

$$\|A'x^* - b'\|_2 \;=\; \|\Pi z\|_2 \;\le\; (1 + \varepsilon)\|z\|_2 \;=\; (1 + \varepsilon)\|Ax^* - b\|_2. \tag{1}$$

By optimality of y^*,

$$\|A'y^* - b'\|_2 \;\le\; \|A'x^* - b'\|_2. \tag{2}$$

Combining (1) and (2) yields,

$$\min_{x' \in \mathbb{R}^d} \|A'x' - b'\|_2 \;=\; \|A'y^* - b'\|_2 \;\leq\; (1+\varepsilon)\|Ax^* - b\|_2 \;=\; (1+\varepsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2.$$

Similarly, for any $y \in \mathbb{R}^d$,

$$\|Ay - b\|_2 \leq (1+\varepsilon)\|A'y - b'\|_2 \leq (1+\varepsilon)\|A'y^* - b'\|_2 \leq (1+3\varepsilon)\|Ax^* - b\|_2.$$

Combining the two inequalities yields the desired approximation guarantee:

$$(1-\varepsilon)\|Ax^* - b\|_2 \;\leq\; \|A'y^* - b'\|_2 \;\leq\; (1+\varepsilon)\|Ax^* - b\|_2.$$

This completes the proof. □

Running Time: The above approach achieves the following results:

- The problem over d vectors in $\mathbb{R}^n$ is reduced to d vectors in $\mathbb{R}^k$, where $k = O(d/\varepsilon^2)$.
- Computing ΠA and Πb can be done in time proportional to $\mathrm{nnz}(A)$ using sparse or fast JL transforms.
- The reduced problem can be solved in $O(d^3/\varepsilon^2)$ time.
- This method is particularly effective when $n \gg d/\varepsilon^2$.

2 Approximate Matrix Multiplication

Now we present a fundamental subroutine in countless computational tasks, including scientific computing, numerical linear algebra, optimization, and machine learning.

2.1 Matrix Multiplication

Given matrices $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{n \times p}$, we need to compute $A^\top B$. The standard naïve algorithm approach requires $O(ndp)$, which is expensive for large n.

For square matrices of size $n \times n$, decades of research have produced significantly faster algorithms. The running time of such algorithms can be expressed as $O(n^\omega)$, where $\omega > 2$ is known as the **exponent of matrix multiplication**.

Optional Reading

Historically, the exponent has been progressively reduced through a series of breakthroughs such as $\omega \leq \log_2 7$ [Strassen, 1969], $\omega \leq 2.376$ [Coppersmith and Winograd, 1987], and $\omega \leq 2.371339$ [Alman et al., 2023]. Although these bounds represent remarkable theoretical progress, the resulting algorithms are not yet practical for large-scale numerical computations. In practice, approximate or randomized methods (e.g., sketching) are often preferred since they can compute approximate matrix products much more efficiently while maintaining small, quantifiable error bounds.
For reference, we include a table showing how researchers have gradually improved the matrix multiplication exponent ω over the five decades (from Wikipedia).

Year	Best known ω	Authors / Contribution
1969	2.8074	Strassen
1978	2.796	Pan
1979	2.780	Bini, Capovani, Romani
1981	2.522	Schönhage
1981	2.517	Romani
1981	2.496	Coppersmith, Winograd
1986	2.479	Strassen (laser method)
1990	2.3755	Coppersmith, Winograd
2010	2.3737	Stothers
2012	2.3729	Vassilevska Williams
2014	2.3728639	Le Gall
2020	2.3728596	Alman, Vassilevska Williams (SODA'21)
2022	2.371866	Duan, Wu, Zhou (FOCS'23)
2024	2.371552	Vassilevska Williams, Y. Xu, Z. Xu, Zhou (SODA'24)
2024	**2.371339**	**Alman, Duan, Vassilevska Williams, Y. Xu, Z. Xu, Zhou (SODA'25)**

2.2 Approximate Matrix Multiplication

Although exact matrix multiplication algorithms have made remarkable theoretical progress, they are often complex and impractical for large-scale data computations. In many modern applications such as machine learning, data analysis, and large-scale optimization, high-quality approximation is often sufficient because we don't always need the perfect answer. This motivates the concept of Approximate Matrix Multiplication.

Overall, the key idea is to trade a small, controlled loss in precision for a substantial gain in computational efficiency. Rather than computing the exact product, we seek an approximation that is close enough for practical use.

Main Goal: Given matrices $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times p}$, we aim to find a matrix $C \in \mathbb{R}^{d\times p}$ such that

$$\|A^\top B - C\|_F \le \varepsilon,$$

with probability at least $1-\delta$.

2.2.1 JL-Based Approach for Fast Matrix Multiplication

We now describe how the Johnson-Lindenstrauss (JL) theorem can be used to design efficient randomized algorithms for approximate matrix multiplication.

Theorem 2.1. *[JL-based Approximation Guarantee] Let $\mathcal{D}$ be a distribution over random matrices $\Pi \in \mathbb{R}^{k\times n}$ satisfying the* Distributional JL Property *with $k = O(\varepsilon^{-2}\log(1/\delta))$. Then for any $A, B \in \mathbb{R}^{n\times d}$,*

$$\Pr_{\Pi\sim\mathcal{D}}\Big[\|A^\top B - (\Pi A)^\top(\Pi B)\|_F \ge \varepsilon \|A\|_F \|B\|_F\Big] \le \delta.$$

Proof sketch of 2.1. We first define the error matrix as follows,

$$M = A^\top B - (\Pi A)^\top(\Pi B).$$

Our goal is to show that $\|M\|_F$ is small with high probability.

Then we analyze a single entry of the error matrix, For column vectors a_i of A and b_j of B, we have

$$M_{ij} = \langle a_i, b_j \rangle - \langle \Pi a_i, \Pi b_j \rangle.$$

The contribution of this entry to $\|M\|_F^2$ is $M_{ij}^2 = (\langle a_i, b_j \rangle - \langle \Pi a_i, \Pi b_j \rangle)^2$.

From the Distributional JL Property, for any unit vectors $x, y \in \mathbb{R}^n$, and Π with k-rows,

$$\mathbb{E}_\Pi \left[(\langle \Pi x, \Pi y \rangle - \langle x, y \rangle)^2 \right] \leq \frac{1}{k}.$$

Now, for any vectors $a_i, b_j \in \mathbb{R}^n$, the following property can be written as follows,

$$\mathbb{E}\Big[\big(\langle \Pi a_i, \Pi b_j \rangle - \langle a_i, b_j \rangle \big)^2 \Big] \;\leq\; \frac{1}{k} \|a_i\|_2^2 \, \|b_j\|_2^2.$$

We apply the Distributional JL property to bound the expected error to get,

$$\begin{aligned}
\mathbb{E}[\|M\|_F^2] = \mathbb{E}[\sum_{i,j} M_{ij}^2] &= \sum_{i,j} \mathbb{E}[(\langle a_i, b_j \rangle - \langle \Pi a_i, \Pi b_j \rangle)^2] \\
&\leq \sum_{i,j} \frac{1}{k} \|a_i\|_2^2 \, \|b_j\|_2^2 \\
&= \frac{1}{k} \|A\|_F^2 \, \|B\|_F^2. && \text{(Summing over } i, j\text{)}
\end{aligned}$$

Hence, for any $\varepsilon > 0$, we apply Markov's inequality,

$$\Pr \left[\|M\|_F^2 \geq \varepsilon^2 \|A\|_F^2 \|B\|_F^2 \right] \leq \frac{\mathbb{E}\left[\|M\|_F^2 \right]}{\varepsilon^2 \|A\|_F^2 \|B\|_F^2} \leq \frac{1}{k\varepsilon^2}$$

Finally, setting $k = O(1/(\varepsilon^2 \delta))$; completes the proof. □

Runtime Analysis: The proposed Approximate Matrix Multiplication achieves substantial speedup compared to the standard matrix multiplication $A^\top B$, while preserving strong accuracy guarantees. Therefore,

- It calculates the projections ΠA and ΠB, where $\Pi \in \mathbb{R}^{k \times n}$ is the JL matrix. This step requires $O(knd + knp)$ time.
- Then, it multiplies the reduced matrices $\Pi A \in \mathbb{R}^{k \times d}$ and $\Pi B \in \mathbb{R}^{k \times p}$, which takes $O(kdp)$ time.
- Overall, the total runtime is $O(k(nd + np + dp)$. Since $k = O(1/(\varepsilon^2 \delta))$, this is much smaller than the naive $O(ndp)$ runtime whenever $n \gg d/\varepsilon^2$.

The Johnson–Lindenstrauss (JL) random projection approximately preserves the dot products between vectors. This property serves as the basis for the Distributional JL guarantee used in Approximate Matrix Multiplication. Now, we establish this observation through the following lemma.

Lemma 2.2. *Let $\Pi \in \mathbb{R}^{k \times d}$ be a random projection matrix whose entries $\Pi_{r,c}$ are independent random variables satisfying*

$$\mathbb{E}[\Pi_{r,c}] = 0, \qquad \mathbb{E}[\Pi_{r,c}^2] = \frac{1}{k}.$$

Then, for any $a, b \in \mathbb{R}^d$,

$$\mathbb{E}[\langle \Pi a, \Pi b \rangle] = \langle a, b \rangle, \qquad \mathbb{E}\big[(\langle \Pi a, \Pi b \rangle - \langle a, b \rangle)^2 \big] \leq \frac{1}{k} \|a\|_2^2 \|b\|_2^2.$$

Proof of Lemma 2.2. Define

$$Z = \langle \Pi a, \Pi b \rangle = \sum_{r=1}^{k} \langle \Pi_r, a \rangle \langle \Pi_r, b \rangle.$$

By taking expectation on both sides, we have

$$\begin{aligned}
\mathbb{E}[Z] &= \sum_{r=1}^{k} \mathbb{E}[\langle \Pi_r, a \rangle \langle \Pi_r, b \rangle] \\
&= k \cdot \mathbb{E}[\langle \Pi_1, a \rangle \langle \Pi_1, b \rangle] \\
&= k \cdot \mathbb{E}\left[\Big(\sum_{s=1}^{d} a_s \pi_{1,s} \Big) \Big(\sum_{s=1}^{d} b_s \pi_{1,s} \Big) \right] \\
&= k \cdot \sum_{s} a_s b_s \mathbb{E}[\pi_{1,s}^2] = k \cdot \frac{1}{k} \sum_{s} a_s b_s = \langle a, b \rangle.
\end{aligned}$$

Thus, Π preserves the dot product in expectation.

Similarly, using independence and the bounded variance of entries,

$$\mathrm{Var}(Z) = \mathrm{Var}(\langle \Pi a, \Pi b \rangle) \leq \frac{1}{k} \|a\|_2^2 \|b\|_2^2.$$

So,

$$\mathbb{E}\big[(\langle \Pi a, \Pi b \rangle - \langle a, b \rangle)^2 \big] = \mathbb{E}\big[(\langle \Pi a, \Pi b \rangle - \mathbb{E}[(\langle \Pi a, \Pi b \rangle])^2 \big] \leq \frac{1}{k} \|a\|_2^2 \|b\|_2^2.$$

Therefore, the random projection approximately preserves dot products in expectation, with variance decaying as $1/k$. □