**CS 388R: Randomized Algorithms, Fall 2023** November 16th, 2023

# Lecture 24: Randomized Numerical Linear Algebra-1

*Prof. Eric Price* *Scribe: Liangchen Liu, Ziheng Chen*

**NOTE:** THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

## 1 Overview

In this lecture and the following lecture, we will delve into more efficient (randomized) methods for solving the linear equation $Ax = b$when $A \in \mathbb{R}^{n\times d}$ for $n \gg d$. The primary goal of this initial lecture is to establish the necessary background, introduce theoretical concepts and tools that will set the stage for the algorithm to be presented in the subsequent lecture.

## 2 Problem Setup

### 2.1 Background

We are interested in solving the linear equation $Ax = b$ given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, specifically when $n \gg d$. This configuration is prevalent in various applications. For instance, in a machine learning problem with $n$ data points in a $d$-dimensional feature space, if the objective is to fit a linear function to these data to approximate a real value function, the values of these data points can be stored in $A$ while the value of the target function can be stored in $b$. Solving for $Ax = b$ provides the optimal coefficients for the linear approximation

**Example: (Polynomial Regression)** data: $(x, y) \in \mathbb{R}^2$, $y = p(x)$ a degree $d$ polynomial.

$$y = p(x) = \sum_{i=0}^{d} a_i x^i$$

One can obtain $p(x)$ by finding $a_i$'s, which can be formulated as $X\mathbf{a} = y$ where each row of $X$ represents the values of $\{1, x, x^2, \ldots, x^d\}$ for a specific values of $x$. And $\mathbf{a} = [a_0, \ldots, a_d]^T$.

### 2.2 Alternative formulation

However, when $n \gg d$, the system is over-determined, making it challenging to obtain an exact solution, or in some cases, an exact solution may not even exist. The following minimization problem offers an appealing alternative:

$$x^* = \arg\min_x \|Ax - b\|_2, \tag{1}$$

where we pick the $\|\cdot\|_2$ for ease of computation. Solution to Equation (1) is given by $x^* = A^\dagger b$ where $A^\dagger$ is the Moore-Penrose inverse or the pseudo inverse, which leads to:

$$x^* = A^\dagger b = (A^T A)^{-1} A^T b. \tag{2}$$

Another perspective to derive Equation (2) is to consider minimizing the loss function:

$$f(x) = \|Ax - b\|_2^2 = (Ax - b)^T (Ax - b).$$

By taking the gradient to be 0, we arrive at:

$$\nabla f(x^*) = 2A^T A x^* - 2A^T b = 0 \implies x^* = (A^T A)^{-1} A^T b. \tag{3}$$

This is called the normal equation for the least square problem (1).

## 2.3 Computation cost

**Direct solve:** For $A \in \mathbb{R}^{n \times d}$ where $n \gg d$, solving Equation (2) directly can be expensive:

1. compute $A^T A$ needs $\mathcal{O}(nd^2)$, or $\mathcal{O}(nd^{1.38})$ with fancy algorithms
2. finding the inverse $(A^T A)^{-1}$ by Gaussian elimination requires $\mathcal{O}(d^3)$, or $\mathcal{O}(d^{2.38})$ with fancy.
3. matrix-vector multiplication $A^T b$ needs $\mathcal{O}(nd)$ and $(A^T A)^{-1}(A^T b)$ needs $\mathcal{O}(d^2)$.

Therefore the total cost is dominated by $\mathcal{O}(nd^2)$ or $\mathcal{O}(nd^{1.38})$ which is huge for $n \gg d$.

**Iterative method:** A better approach is to consider solving iteratively. Given the current approximate minimizer $x_{k-1}$, we want to update to $x_k$ such that $x_k$ is a better minimizer. The most common and well-known approach is the steepest descent, or gradient descend. From Equation (3), we learn the gradient direction of the minimization function is $\nabla f(x) = b - Ax$, therefore, gradient descent results in the iteration:

$$x_k = x_{k-1} + \eta_k (b - Ax_{k-1}).$$

$\eta_k$ is the step size, or the learning rate in deep learning context. $\eta_k$ can be either prescribed or obtained through line search:

$$\eta_k = \frac{(b - Ax)^T (b - Ax)}{(b - Ax)^T A (b - Ax)}.$$

For $A \in \mathbb{R}^{n \times d}$, the number of steps required for an $(1 + \varepsilon)$-accuracy, that is,

$$\|A\hat{x} - b\|_2 \leq (1 + \varepsilon) \min_x \|Ax - b\|_2$$

is given by

$$\mathcal{O}\left( nd \log(\frac{n}{\varepsilon}) \kappa(A^T A) \right),$$

where $\kappa$ denotes the condition number of a matrix:

$$\kappa(B) := \frac{\sigma_{\max}(B)}{\sigma_{\min}(B)} \implies \kappa(A^T A) = \frac{\sigma_{\max}(A^T A)}{\sigma_{\min}(A^T A)} = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)},$$

and $\sigma$ denotes the singular value and $\lambda$ the eigenvalues.

Nonetheless, when some features are highly correlated, the feature distributions will become elongated, leading to $\lambda_{\max} \gg \lambda_{\min}$, an ill-conditioned system. For an example, consider for $\varepsilon \ll 1$,

$$A = \begin{bmatrix} 1, & \mathcal{O}(\varepsilon) \\ \mathcal{O}(\varepsilon), & \mathcal{O}(\varepsilon^2) \end{bmatrix} \implies \lambda_{\max} = 1,\ \lambda_{\min} \sim \mathcal{O}(\varepsilon^2) \implies \kappa(A^T A) \sim \mathcal{O}(\varepsilon^{-2}) \gg 1.$$

Under this circumstance, gradient descent requires a a long time to converge. One possible improvement is to adopt the celebrated conjugate [HS52][1], which brings down the required number of steps to

$$\mathcal{O}\left(nd \log(n/\varepsilon)\sqrt{\kappa(A^T A)}\right),$$

but in general for iterative methods, there is always a dependency on the condition number $\kappa$.

## 3 Randomized Technique

The goal of our lectures is to get rid of the $\kappa$ dependency to achieve a computation cost of $\tilde{\mathcal{O}}\left(nd + \frac{d^3}{\varepsilon^2}\right)$. The framework to be introduce is called *"Sketch&Solve"*. In the final part of the next lecture, we describe methods of preconditioning to get rid of the *poly*($\varepsilon$)-dependency as well.

### 3.1 Sketch&Solve

Recall we want to find $\arg\min_x \|Ax - b\|_2$ for $A \in \mathbb{R}^{n\times d}$ with $n \gg d$. The main idea is *to pick a "sketch" matrix* $S \in \mathbb{R}^{m\times n}$ and solve the following instead in $\mathcal{O}(md^2)$:

$$\arg\min_x \|SAx - Sb\|_2. \tag{4}$$

Note if there's no noise in $b$, then for $m > d$ a full rank system is guaranteed in general.

**Intuition:** the idea behind "Sketch&Solve" is that by selecting a "good" $S$ to sketch the important information from $A$ into a smaller system $SA$, the solution obtained from solving Equation (4), which is a fast process, will be sufficiently accurate.

The concept of *subspace embedding* provides a criteria to describe what is to be a "good" $S$.

**Definition 1** (Subspace Embedding). *$S$ is a subspace embedding for a space $X$ if*

$$\|Sx\|_2^2 = (1 \pm \varepsilon)\|x\|_2^2 \quad \forall x \in X$$

*$S$ is a d-dimensional oblivious subspace embedding (OSE) if for any $A \in \mathbb{R}^{n\times d}$*

$$\|SAx\|_2^2 = (1 \pm \varepsilon)\|Ax\|_2^2 \quad \forall x \in \mathbb{R}^d \tag{5}$$

[1]There are various improvements available such as classical and Nestrov momentum to achieve the $\sqrt{\kappa}$ rate

**Remark:** We effectively apply $S$ on $col(A)$ above. More generally, OSE is defined as for any $d$-dimensional U:

$$\|Sx\|_2^2 = (1 \pm \varepsilon)\|x\|_2^2 \quad \forall x \in U$$

The main reason to consider OSE is that it is not practical to come up with a subspace embedding $S$ for a specific $A$ (or $col(A)$), obtaining an OSE is more versatile.

**Claim 2.** *A sufficient condition for "Sketch&Solve" to be $(1 + \mathcal{O}(\varepsilon))$-accurate is that $S$ is a subspace embedding for $\bar{A} := \begin{bmatrix} A \,|b \end{bmatrix}$. That is:*

$$S \textit{ being a subspace embedding of } \bar{A} \implies \|A\hat{x} - b\|_2 \le (1+\varepsilon)\min_x \|Ax - b\|_2,$$

*where $\hat{x}$ comes from solving Equation* (4)*.*

*Proof.* Note that $Ax - b = \bar{A}\begin{bmatrix} x \\ -1 \end{bmatrix}$ implies $Ax - b$ lies in $col(\bar{A})$. Let $\hat{x}$ denote the solution returned by "Sketch&Solve" and $x^*$ the true solution, then since $S$ is an OSE for $\bar{A}$, by definition:

$$\|S(Ax - b)\|_2^2 = (1 \pm \varepsilon)\|Ax - b\|_2^2 \quad \forall x.$$

$$\implies \|A\hat{x} - b\|_2^2 \le \frac{1}{1-\varepsilon}\|S(A\hat{x} - b)\|_2^2 \le \frac{1}{1-\varepsilon}\|S(Ax^* - b)\|_2^2 \le \frac{1+\varepsilon}{1-\varepsilon}\|Ax^* - b\|_2^2,$$

where the middle inequality follows from the fact that $\hat{x}$ is the optimal solution for $SAx = b$. Therefore we can conclude:

$$\implies \|A\hat{x} - b\|_2 \le (1 + \varepsilon')\min_x \|Ax - b\|_2$$

□

Now that we have an (oblivious) subspace embedding, we can obtain an approximate solution to $Ax = b$ with $\big(1 + \mathcal{O}(\varepsilon)\big)$-accuracy. The remaining questions to investigate are:

1. How to find such an embedding $S$?
2. Is $SA$ fast to compute so that embedding does not dominate the computation cost?

These questions will be addressed in the next lecture, as a few more concepts need to be introduced before we are able to arrive at the answer. We first introduce the notion of distributional Johnson-Lindenstrauss, and later on demonstrate how one can obtain an OSE from it.

## 3.2 Distributional Johnson–Lindenstrauss

**Definition 3.** *A random matrix $S$ is $(\varepsilon, \delta)$-distributional Johnson-Lindenstrauss (JL) if for any $x \in \mathbb{R}^d$:*

$$\|Sx\|_2^2 = (1 \pm \varepsilon)\|x\|_2^2 \tag{6}$$

*with probability $1 - \delta$.*

**Remark:** Distributional JL is the subspace embedding applied on the whole space $\mathbb{R}^d$. It also describes that the random matrix $S$ has a high probability to be orthogonal. Later on we will see a connection from distributional JL to OSE for a certain subspace.

**Example:** $S$ with *i.i.d.* Gaussian entries $\sim \mathcal{N}(0, \frac{1}{m})$ and $m = \mathcal{O}\left(\frac{1}{\varepsilon^2}\log(\frac{1}{\delta})\right)$ rows is distributional JL. A rough intuition is that $\|Sx\|_2^2$ can be thought as a sum of $[\mathcal{N}(0,1)]^2 \sim \chi^2$, where the $\chi^2$ distribution is sub-exponential, therefore $\|Sx\|_2^2$ is sub-Gamma so it concentrates.

**Corollary 4.** *If $S$ is $(\varepsilon,\delta)$-distributional JL, then $\forall x, y \in \mathbb{R}^d$, with probability $1-2\delta$:*

$$\langle Sx, Sy\rangle = \langle x, y\rangle \pm \varepsilon\|x\|_2\|y\|_2.$$

*Proof.* By expanding the inner product:

$$\langle S(x+y), S(x+y)\rangle - \langle S(x-y), S(x-y)\rangle = 4\langle Sx, Sy\rangle.$$

On the other hand, by definition of $(\varepsilon,\delta)$-distributional JL:

$$\begin{aligned}&\langle S(x+y), S(x+y)\rangle - \langle S(x-y), S(x-y)\rangle \\ =&\langle x+y, x+y\rangle - \langle x-y, x-y\rangle \pm \varepsilon\|x+y\|^2 \pm \varepsilon\|x-y\|^2 \\ =&4\langle x, y\rangle \pm \varepsilon(2\|x\|_2^2 + 2\|y\|_2^2).\end{aligned}$$

Equating the two we have

$$\langle Sx, Sy\rangle = \langle x, y\rangle \pm \varepsilon(\|x\|_2^2/2 + \|y\|_2^2/2).$$

By scaling $x, y$ to have norm 1:

$$\langle Sx, Sy\rangle = \langle x, y\rangle \pm \varepsilon \implies \left\langle S\frac{\tilde{x}}{\|\tilde{x}\|_2}, S\frac{\tilde{y}}{\|\tilde{y}\|_2}\right\rangle = \left\langle \frac{\tilde{x}}{\|\tilde{x}\|_2}, \frac{\tilde{y}}{\|\tilde{y}\|_2}\right\rangle \pm \varepsilon \implies \langle S\tilde{x}, S\tilde{y}\rangle = \langle \tilde{x}, \tilde{y}\rangle \pm \varepsilon\|\tilde{x}\|_2\|\tilde{y}\|_2,$$

holds for any general $\tilde{x}, \tilde{y} \in \mathbb{R}^d$. □

**Remark:** The distributional JL works effectively only on a countable number of points. To extend its applicability to the entire subspace, it is essential to demonstrate that a finite number of points can provide good coverage of the underlying subspace. This motivates the introduction of the concept of $\varepsilon$-Net.

## 3.3 Net (cover)

**Definition 5** ($\varepsilon$-net)**.** *Let $X$ be a set (e.g. $\mathcal{S}^{d-1} = \{x \in \mathbb{R}^d \mid \|x\|_2 = 1\}$), $N$ is an $\varepsilon$-net for $X$ if:*

$$\forall x \in X : \exists y \in N,\ \|x-y\| \le \varepsilon.$$

**Lemma 6.** *$\exists \varepsilon$-net $N$ for $\mathcal{S}^{d-1}$ of size $\le \left(1+\frac{2}{\varepsilon}\right)^d$.*

*Proof.* Consider a greedy approach by constantly adding points to $N$ (that is, go through all the points $x \in \mathcal{S}$, if $x$ is not $\varepsilon$-close to any $y \in N$, add $x$ to $N$). Let there be $n$ points in $N$: $N = \{x_1, x_2, \ldots, x_n\}$; by definition:

$$\|x_i - x_j\| \geq \varepsilon \,\forall\, i, j \implies \mathcal{B}\left(x_i, \frac{\varepsilon}{2}\right) \text{ are disjoint,}$$

where $\mathcal{B}\left(x_i, \frac{\varepsilon}{2}\right)$ are balls centered at $x_i$ with radius $\frac{\varepsilon}{2}$. Then, since $x_i$ are all from $\mathcal{S}^{d-1}$:

$$N \times Vol\left(\mathcal{B}\left(0, \frac{\varepsilon}{2}\right)\right) = Vol\left(\bigcup_i \mathcal{B}\left(x_i, \frac{\varepsilon}{2}\right)\right) \leq Vol\left(\mathcal{B}\left(0, 1 + \frac{\varepsilon}{2}\right)\right).$$

Since $Vol(\mathcal{B}_r) \sim r^d Vol(\mathcal{B}_1)$, we can conclude that:

$$N \leq \frac{(1 + \varepsilon/2)^d}{(\varepsilon/2)^d} = \left(1 + \frac{2}{\varepsilon}\right)^d$$

$\square$

**Remark:** By choosing $\varepsilon$ not so small above, the size of the net is roughly $2^{\mathcal{O}(d)}$. In fact, morally one can show any $d$-dimensional subspace "has $2^{\mathcal{O}(d)}$ points". Therefore, if $S$ is $\left(\varepsilon, \delta 2^{-\mathcal{O}(d)}\right)$-distributional JL, using a union bound, $S$ will be an OSE with probability $1 - \delta$. This is how one can connect distributional JL to OSE, hence obtain a reasonably good approximate solution. We will show this in a more precised manner in the next lecture (Theorem 7).

## References

[HS52] Magnus R Hestenes, Eduard Stiefel. Methods of conjugate gradients for solving linear systems *Journal of Research of the National Bureau of Standards*, 49(6):409-436, 1952.