**CS 839 Probability and Learning in High Dimension** Lecture 24 - 04/27/2022

# Lecture 24: Randomized Numerical Linear Algebra I

*Lecturer: Yudong Chen* *Scribe: Winfred Li*

In the next few classes, we will discuss randomized numerical linear algebra (RandNLA) and its applications to a number of problems. In this lecture we consider the problem of approximate matrix multiplication, and then begin a discussion on the problem of least squared approximation.[1][2]

## 1 Introduction

In RandNLA, we develop randomized algorithms for several fundamental numerical linear algebra tasks:

1. Approximate matrix multiplication (Today)
2. Least squares approximation (Today and next lecture)
3. Low-rank matrix approximation
4. (Not this semester) Graph sparsification

We will focus on the first three problems. Note that techniques developed in Problem 1 will be used in Problem 2, which in turn will be used in Problem 3.

### 1.1 Efficient large-scale data processing

When processing large-scale data (in particular, streaming data), we desire methods that can be performed with

- a few (e.g., one or two) passes of data
- limited memory (complete data might not fit in memory)
- low time complexity

A general idea is RandNLA is to use randomization to get a rough sketch of the data. There are two main approaches:

- **Random (down)sampling:** randomly select a subset of data
  - Pros: Easy to implement
  - Cons: Quality often depends on property of data
- **Random projection:** rotates / projects data to lower dimensions
  - Pros: Often data-agnostic, more robust
  - Cons: More computationally expensive

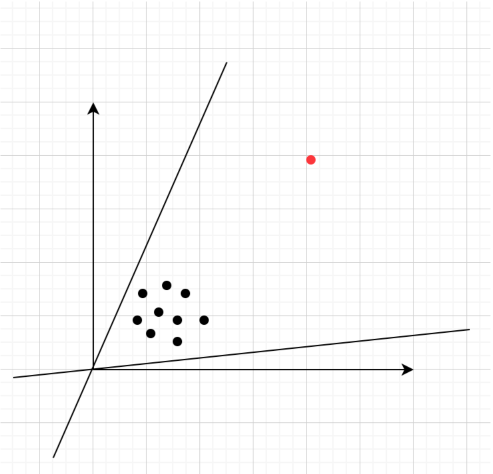
See Figure 1 for an illustrations of the pros and cons of these two approaches.

On a high level, RandNLA involves randomized algorithm + matrix concentration/perturbation theory. Note that Probability used in algorithm *and* analysis (previously only used in analysis). Applications of RandNLA include

[1] Reading: [Mahoney, 2016]: "Lecture notes on randomized linear algebra. Available at `https://arxiv.org/abs/1608.04481`

[2] We thank Yuxin Chen for allowing us to draw from his slides for ELE 520: Mathematics of Data Science, Princeton University, Fall 2020.

**Figure 1:** Consider estimating $\sum \|x_i\|$ for the points below by random sampling. In order for the estimate to be accurate we need to include the red outlier point, for instance by increasing sampling weight for red point. On the other hand, random projections preserve more information from data



- Scientific computing;
- Machine learning: e.g., PCA on ≥terabyte-sized problems.

## 2 Faster Matrix Multiplication

Given $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times p}$, compute or approximate $\boldsymbol{AB}$. A native algorithm for exactly computing $\boldsymbol{AB}$ is given in Algorithm 1.

**Algorithm 1:** Vanilla algorithm for matrix multiplication

```
for $i = 1, \cdots, m$ do
    for $k = 1, \cdots, n$ do
        $M_{i,k} = \boldsymbol{A}_{i,:}\boldsymbol{B}_{:,k}$
return $\boldsymbol{M}$
```

The computational complexity of Algorithm 1 is $O(mnp)$, or $O(n^3)$ if $m = n = p$. For simplicity, we shall assume $m = n = p$ in the sequel unless otherwise noted.

There exists algorithms with sub-cubic time complexity, such as the **Strassen algorithm** for exact matrix multiplication. The asymptotic computational complexity is $\approx O(n^{2.8})$. For various reasons, this algorithm is ot commonly used in practice.

Our general goal is to develop randomized algorithm with time complexity $= \tilde{O}(\frac{n^2}{\epsilon^2})$.

### 2.1 A Simple Randomized Algorithm

We view $\boldsymbol{AB}$ as a sum of rank-one matrices (or outer products)

$$\boldsymbol{AB} = \sum_{i=1}^{n} \boldsymbol{A}_{:,i}\boldsymbol{B}_{i,:}.$$

Our **idea** is to randomly sample $L$ rank-one components. This leads to Algorithm 2.

**Algorithm 2:** Basic randomized algorithm for matrix multiplication

```
for $l = 1, \cdots, L$ do
    Pick $i_l \in \{1, \cdots, n\}$ i.i.d. with probability $\mathbb{P}\{i_l = k\} = p_k$
return
    $M = \frac{1}{L} \sum_{l=1}^{L} \frac{1}{p_{i_l}} A_{:,i_l} B_{i_l,:}$
```

A few remarks for Algorithm 2.

- $\{p_k : k = 1, \ldots, n\}$ are called *importance sampling probabilities.*
- $\boldsymbol{M}$ is an *unbiased* estimate of $\boldsymbol{AB}$, i.e.,

$$\mathbb{E}[\boldsymbol{M}] = \boldsymbol{AB}.$$

- The approximation error (e.g., $\|\boldsymbol{M} - \boldsymbol{AB}\|$) depends on $\{p_k\}$.

## 2.2 Importance Sampling Probabilities

There are two ways for choosing the probabilities $\{p_k : k = 1, \ldots, n\}$. The simplest one is:

- **Uniform sampling**

$$p_k \equiv \frac{1}{n}.$$

  Here one can sample the index set $\{i_1, \ldots, i_L\}$ before looking at data, so it's implementable via one pass over data

Intuitively, one may prefer biasing towards larger rank-1 components. This suggests that we can use

- **Non-uniform sampling**

$$p_k = \frac{\|\boldsymbol{A}_{:,k}\|_2 \|\boldsymbol{B}_{k,:}\|_2}{\sum_{l=1}^{n} \|\boldsymbol{A}_{:,l}\|_2 \|\boldsymbol{B}_{l,:}\|_2}$$

  Note that the probabilities $\{p_k\}$ can be computed using one pass and $O(n)$ memory. Nonuniform sampling favors “larger" rank 1 matrices, with sampling probabilities proportional to

$$\|\boldsymbol{A}_{:,k} \boldsymbol{B}_{k,:}\|_{op} = \|\boldsymbol{A}_{:,k}\|_2 \, \|\boldsymbol{B}_{k,:}\|_2 \, .$$

It turns out the nonuniform sampling probabilities above are optimal with respect to the the mean squared approximation error $\mathbb{E}\left[\|\boldsymbol{M} - \boldsymbol{AB}\|_{\mathrm{F}}^2\right]$.

**Lemma 1.** $\mathbb{E}\left[\|\boldsymbol{M} - \boldsymbol{AB}\|_{\mathrm{F}}^2\right]$ *is minimized by*

$$p_k = \frac{\|\boldsymbol{A}_{:,k}\|_2 \|\boldsymbol{B}_{k,:}\|_2}{\sum_{l=1}^{n} \|\boldsymbol{A}_{:,l}\|_2 \|\boldsymbol{B}_{l,:}\|_2}. \tag{1}$$

- Consequently, we call (1) the optimal sampling probabilities (w.r.t. mean squared error)

**Proof** Since $\mathbb{E}[\boldsymbol{M}] = \boldsymbol{AB}$, one has

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{M}-\boldsymbol{AB}\|_{\mathrm{F}}^2\right] &= \mathbb{E}\left[\sum_{i,j}\left(M_{i,j}-\boldsymbol{A}_{i,:}\boldsymbol{B}_{:,j}\right)^2\right] = \sum_{i,j}\mathsf{Var}[M_{i,j}] \\
&= \frac{1}{L}\sum_k\sum_{i,j}\frac{A_{i,k}^2B_{k,j}^2}{p_k} - \frac{1}{L}\sum_{i,j}\left(\boldsymbol{A}_{i,:}\boldsymbol{B}_{:,j}\right)^2 \quad \text{(check)} \\
&= \frac{1}{L}\sum_k\frac{1}{p_k}\|\boldsymbol{A}_{:,k}\|_2^2\|\boldsymbol{B}_{k,:}\|_2^2 - \frac{1}{L}\|\boldsymbol{AB}\|_{\mathrm{F}}^2 \qquad (2)
\end{aligned}$$

In addition, Cauchy-Schwarz yields $\left(\sum_k p_k\right)\left(\sum_k \frac{\alpha_k}{p_k}\right) \geq \left(\sum_k \sqrt{\alpha_k}\right)^2$, with equality attained if $p_k \propto \sqrt{\alpha_k}$. Setting $\alpha_k = \|\boldsymbol{A}_{:,k}\|_2^2\|\boldsymbol{B}_{k,:}\|_2^2$ gives

$$\mathbb{E}\left[\|\boldsymbol{M}-\boldsymbol{AB}\|_{\mathrm{F}}^2\right] \geq \frac{1}{L}\left(\sum_k\|\boldsymbol{A}_{:,k}\|_2\|\boldsymbol{B}_{k,:}\|_2\right)^2 - \frac{1}{L}\|\boldsymbol{AB}\|_{\mathrm{F}}^2,$$

where the lower bound is achieved when $p_k \propto \|\boldsymbol{A}_{:,k}\|_2\|\boldsymbol{B}_{k,:}\|_2$ □

## 2.3 Error Concentration

The previous analysis concerns $\mathbb{E}\left[\|\boldsymbol{M}-\boldsymbol{AB}\|_{\mathrm{F}}^2\right]$, i.e., the approximation error **in expectation**. In practice, one often hopes, in addition, that $\boldsymbol{M}$ is close to $\boldsymbol{AB}$ most of the time. Thus, we desire an error bound that holds **with high probability.** For matrix multiplication, two error metrics are of particular interest:

- Frobenius norm error: $\|\boldsymbol{M}-\boldsymbol{AB}\|_{\mathrm{F}}$, and
- spectral norm error: $\|\boldsymbol{M}-\boldsymbol{AB}\|$.

Our general idea is to **invoke matrix concentration inequalities to control these metrics.** To this end, recall the Matrix Bernstein inequality.

**Theorem 1** (Matrix Bernstein)**.** *Let $\left\{\boldsymbol{X}_l \in \mathbb{R}^{d_1\times d_2}, l=1,\dots,L\right\}$ be a sequence of independent zero-mean random matrices. Assume*

$$\begin{aligned}
&\text{(range)} && \|\boldsymbol{X}_l\| \leq R, \quad \forall l = 1,\dots,L, \\
&\text{(variance)} && \max\left\{\left\|\mathbb{E}\left[\textstyle\sum_{l=1}^L \boldsymbol{X}_l\boldsymbol{X}_l^\top\right]\right\|, \left\|\mathbb{E}\left[\textstyle\sum_{l=1}^L \boldsymbol{X}_l^\top\boldsymbol{X}_l\right]\right\|\right\} \leq V.
\end{aligned}$$

*Then,*

$$\mathbb{P}\left\{\left\|\textstyle\sum_{l=1}^L\boldsymbol{X}_l\right\| \geq \tau\right\} \leq (d_1+d_2)\exp\left(\frac{-\tau^2/2}{V+R\tau/3}\right)$$

## 2.4 Frobenius Norm Error

We have the following bound on the approximation error in Frobenius norm.

**Theorem 2** (Frobenius Norm Error of Matrix Multiplication)**.** *Suppose $p_k \geq \frac{\beta\|\boldsymbol{A}_{:,k}\|_2\|\boldsymbol{B}_{k,:}\|_2}{\sum_l\|\boldsymbol{A}_{:,l}\|_2\|\boldsymbol{B}_{l,:}\|_2}, \forall k$ for some $\beta\in(0,1]$. If $L \gtrsim \frac{\log n}{\beta}$, then the estimate $\boldsymbol{M}$ returned by Algorithm 2 obeys*

$$\|\boldsymbol{M}-\boldsymbol{AB}\|_{\mathrm{F}} \lesssim \sqrt{\frac{\log n}{\beta L}}\|\boldsymbol{A}\|_{\mathrm{F}}\|\boldsymbol{B}\|_{\mathrm{F}}$$

*with probability at least $1-O(n^{-10})$.*

**Proof** Observe that $\mathsf{vec}(\boldsymbol{M}) = \sum_{l=1}^{L} \boldsymbol{X}_l$, where $\boldsymbol{X}_l = \sum_{k=1}^{n} \frac{1}{Lp_k} \boldsymbol{A}_{:,k} \otimes \boldsymbol{B}_{k,:}^{\top} \mathbb{1}\{i_l = k\}$. These matrices $\{\boldsymbol{X}_l\}$ obey

$$\|\boldsymbol{X}_l\|_2 \leq \max_k \frac{1}{Lp_k} \|\boldsymbol{A}_{:,k}\|_2 \|\boldsymbol{B}_{k,:}\|_2 \asymp \frac{1}{\beta L} \sum_{k=1}^{n} \|\boldsymbol{A}_{:,k}\|_2 \|\boldsymbol{B}_{k,:}\|_2 =: R, \tag{3}$$

$$\mathbb{E}\left[\sum_{l=1}^{L} \|\boldsymbol{X}_l\|_2^2\right] = L \sum_{k=1}^{n} \mathbb{P}\{i_l = k\} \frac{\|\boldsymbol{A}_{:,k}\|_2^2 \|\boldsymbol{B}_{k,:}\|_2^2}{L^2 p_k^2} \leq \underbrace{\frac{\left(\sum_{k=1}^{n} \|\boldsymbol{A}_{k,:}\|_2 \|\boldsymbol{B}_{k,:}\|_2\right)^2}{\beta L}}_{=:V}. \tag{4}$$

We then invoke matrix Bernstein to obtain

$$\begin{aligned}
\|\boldsymbol{M} - \boldsymbol{A}\boldsymbol{B}\|_{\mathrm{F}} &= \left\|\sum_{l=1}^{L} (\boldsymbol{X}_l - \mathbb{E}[\boldsymbol{X}_l])\right\|_2 \\
&\lesssim \sqrt{V \log n} + R \log n \\
&\asymp \sqrt{\frac{\log n}{\beta L}} \left(\sum_{k=1}^{n} \|\boldsymbol{A}_{k,:}\|_2 \|\boldsymbol{B}_{k,:}\|_2\right) \\
&\leq \sqrt{\frac{\log n}{\beta L}} \|\boldsymbol{A}\|_{\mathrm{F}} \|\boldsymbol{B}\|_{\mathrm{F}} \text{ (Cauchy-Schwarz)}
\end{aligned}$$

□

A few remarks on Theorem 2:

- If $L \gtrsim \frac{\log n}{\varepsilon^2 \beta}$, then $\|\boldsymbol{M} - \boldsymbol{A}\boldsymbol{B}\|_{\mathrm{F}} \lesssim \varepsilon \|\boldsymbol{A}\|_{\mathrm{F}} \|\boldsymbol{B}\|_{\mathrm{F}}$.
- Time complexity: $\underbrace{O\left(n^2\right)}_{\text{compute } p_k' s} + \underbrace{O\left(\frac{n^2 \log n}{\epsilon^2}\right)}_{\substack{\text{form } L \text{ rank-1 matrices} \\ \text{and compute their sum}}}$ . Note that this is one of the few cases where the time complexity can be readily estimated. For more complex problems, bounding time complexity requires more careful arguments and in particular is implementation dependent.
- The proof above highlights why nonuniform sampling probabilities is useful. If we instead use the probabilities $p_k = \frac{1}{n}$, then equation (3) would become $\max_k \frac{n}{L} \|\boldsymbol{A}_{:,k}\|_2 \|\boldsymbol{B}_{k,:}\|_2$, from which Matrix Bernstein will result in a worst error bound.

## 2.5 Spectral Norm Error

We also have the following bound on the approximation error in spectral norm. For simplicity, we consider the simpler problem of approximating $\boldsymbol{A}\boldsymbol{A}^{\top}$ (i.e., $\boldsymbol{B} = \boldsymbol{A}^{\top}$).

**Theorem 3.** *Suppose* $p_k \geq \frac{\beta \|\boldsymbol{A}_{:,k}\|_2^2}{\|\boldsymbol{A}\|_{\mathrm{F}}^2}, \forall k$ *for some quantity* $0 < \beta \leq 1$, *and* $L \gtrsim \frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2}{\beta \|\boldsymbol{A}\|^2 \log n}$. *Then the estimate* $\boldsymbol{M}$ *returned by Algorithm 2 obeys*

$$\|\boldsymbol{M} - \boldsymbol{A}\boldsymbol{A}^{\top}\| \lesssim \sqrt{\frac{\log n}{\beta L}} \|\boldsymbol{A}\|_{\mathrm{F}} \|\boldsymbol{A}\|$$

*with prob. at least* $1 - O(n^{-10})$.

**Proof** Write $\boldsymbol{M} = \sum_{l=1}^{L} \boldsymbol{Z}_l$, where $\boldsymbol{Z}_l = \sum_{k=1}^{n} \frac{1}{Lp_k} \boldsymbol{A}_{:,k} \boldsymbol{A}_{:,k}^\top \mathbb{1}\{i_l = k\}$. These matrices satisfy

$$\begin{aligned}
\|\boldsymbol{Z}_l\|_2 &\le \max_k \frac{\|\boldsymbol{A}_{:,k}\|_2^2}{Lp_k} \le \frac{1}{\beta L}\|\boldsymbol{A}\|_{\mathrm{F}}^2 =: R \\
\left\| \mathbb{E}\left[\sum_{l=1}^{L} \boldsymbol{Z}_l \boldsymbol{Z}_l^\top\right] \right\| &= \left\| L \sum_{k=1}^{n} \mathbb{P}\{i_l = k\} \frac{\|\boldsymbol{A}_{:,k}\|_2^2}{L^2 p_k^2} \boldsymbol{A}_{:,k} \boldsymbol{A}_{:,k}^\top \right\| \\
&\le \frac{1}{\beta L} \|\boldsymbol{A}\|_{\mathrm{F}}^2 \left\|\boldsymbol{A}\boldsymbol{A}^\top\right\| \\
&= \frac{1}{\beta L} \|\boldsymbol{A}\|_{\mathrm{F}}^2 \|\boldsymbol{A}\|^2 =: V
\end{aligned}$$

Invoke matrix Bernstein to conclude that with high prob.,

$$\begin{aligned}
\left\|\boldsymbol{M} - \boldsymbol{A}\boldsymbol{A}^\top\right\| = \left\| \sum\nolimits_{l=1}^{L} (\boldsymbol{Z}_l - \mathbb{E}[\boldsymbol{Z}_l]) \right\| &\lesssim \sqrt{V \log n} + B \log n \\
&\asymp \sqrt{\frac{\log n}{\beta L}} \|\boldsymbol{A}\|_{\mathrm{F}} \|\boldsymbol{A}\|
\end{aligned}$$

□

A few remarks on Theorem 3:

- The error bound depends on $\|A\|_F \|A\|$. In comparison, the Frobenius error bound in Theorem 2 involves $\|A\|_F^2$.
- If $L \gtrsim \frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2}{\|\boldsymbol{A}\|^2} \frac{\log n}{\varepsilon^2 \beta}$, then $\|\boldsymbol{M} - \boldsymbol{A}\boldsymbol{A}^\top\| \lesssim \varepsilon \|\boldsymbol{A}\|^2$. Here $\frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2}{\|\boldsymbol{A}\|^2}$ is called the *stable rank* of the matrix $\boldsymbol{A}$. For approximately low rank matrices the stable rank $\approx 1$; for high rank matrices stable rank $\approx n$. Therefore, Theorem 3 is most useful when $A$ is approximately low rank.
- We note that Theorem 3 is a generalization of kernel approximation bound for random features from Lecture 5.
- Can be generalized to approximate $\boldsymbol{AB}$ (Magen, Zouzias '11) [Magen and Zouzias, 2011]
- For many problems, the spectral bound is more useful since

$$\|\boldsymbol{M} - \boldsymbol{AB}\| \le \epsilon \Rightarrow \|\boldsymbol{M}u - \boldsymbol{AB}u\|_2 \le \epsilon \|u\|_2$$

## 2.6 Matrix Multiplication with One Sided Information

What if we only use the information about $\boldsymbol{A}$ (but not $\boldsymbol{B}$)?

- This situation arises when $\boldsymbol{B}$ is defined implicitly, e.g., as the solution to optimization problem.
- We will later see an example of this situation in approximate least squares with nonuniform sampling.

Suppose we choose the sample probabilities as $p_k \ge \frac{\beta \|\boldsymbol{A}_{:,k}\|_2^2}{\|\boldsymbol{A}\|_{\mathrm{F}}^2}$. In this case, matrix Bernstein does NOT give good concentration bounds. In particular, we will get a poor bound in equation (3):

$$R \propto \max_k \underbrace{\frac{\|B_{k,:}\|_2}{\|A_{:,k}\|_2}}_{\text{Can be large}} \|A\|_F^2.$$

Nevertheless, we can still use Markov's inequality to get some useful bound. More precisely, when $p_k \geq \frac{\beta\|\boldsymbol{A}_{:,k}\|_2^2}{\|\boldsymbol{A}\|_{\mathrm{F}}^2}$, it follows from (2) that

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{M}-\boldsymbol{A}\boldsymbol{B}\|_{\mathrm{F}}^2\right] &= \frac{1}{L}\sum_k \frac{1}{p_k}\|\boldsymbol{A}_{:,k}\|_2^2\|\boldsymbol{B}_{k,:}\|_2^2 - \frac{1}{L}\|\boldsymbol{A}\boldsymbol{B}\|_{\mathrm{F}}^2 \\
&\leq \frac{1}{\beta L}\left(\sum_k \|\boldsymbol{B}_{k,:}\|_2^2\right)\|\boldsymbol{A}\|_{\mathrm{F}}^2 \\
&= \frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2\|\boldsymbol{B}\|_{\mathrm{F}}^2}{\beta L}.
\end{aligned}$$

Hence, Markov's inequality yields that with prob. at least $1-\frac{1}{\log n}$,

$$\|\boldsymbol{M}-\boldsymbol{A}\boldsymbol{B}\|_{\mathrm{F}}^2 \leq \frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2\|\boldsymbol{B}\|_{\mathrm{F}}^2\log n}{\beta L}. \tag{5}$$

Note that the probability is worse compared to Theorem 2, which gives $1-\frac{1}{n^{10}}$.

# 3 Least Squares Approximation

We next turn to the least squares (LS) problem.

## 3.1 Least Squares Problem

Given $\boldsymbol{A}\in\mathbb{R}^{n\times d}$ $(n \gg d)$ and $\boldsymbol{b}\in\mathbb{R}^n$, the goal is to find the solution $\boldsymbol{x}_{\mathsf{ls}}$ to the optimization problem

$$\text{minimize}_{\boldsymbol{x}\in\mathbb{R}^d} \quad \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2.$$

See Figure 2 for a geometric interpretation of the least squares problem, which is equivalent to computing the orthogonal projection of $\boldsymbol{b}$ onto range($\boldsymbol{A}$) (the column space of $\boldsymbol{A}$). The least squares solution $\boldsymbol{x}_{\mathsf{ls}}$ satisfies $(\boldsymbol{A}\boldsymbol{x}_{\mathsf{ls}}-\boldsymbol{b}) \perp \mathrm{range}(\boldsymbol{A})$.

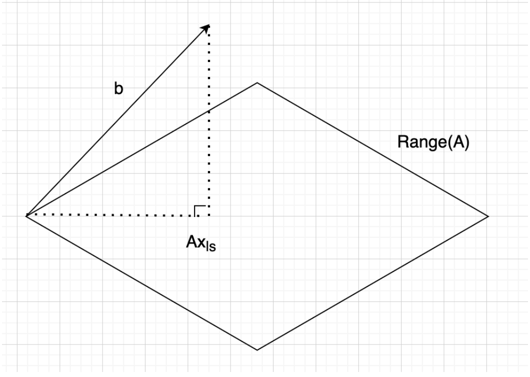


**Figure 2:** Geometric picture of least squares

Algebraically, $\boldsymbol{x}_{\mathsf{ls}}$ satisfies the *normal equation*

$$\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x}_{\mathsf{ls}} = \boldsymbol{A}^\top\boldsymbol{b}, \qquad \text{or equivalently} \qquad \boldsymbol{A}^T(\boldsymbol{A}x_{\mathsf{ls}}-b)=0.$$

If $\boldsymbol{A}$ has full column rank, then $\boldsymbol{x}_{\mathsf{ls}}$ is given in closed form

$$\boldsymbol{x}_{\mathsf{ls}} = (\boldsymbol{A}^\top \boldsymbol{A})^{-1}\boldsymbol{A}^\top \boldsymbol{b} = \underbrace{\boldsymbol{V}_A \boldsymbol{\Sigma}_A^{-1} \boldsymbol{U}_A^\top}_{=:\boldsymbol{A}^\dagger} \boldsymbol{b},$$

where $\boldsymbol{A} = \boldsymbol{U}_A \boldsymbol{\Sigma}_A \boldsymbol{V}_A^\top$ is the SVD of $\boldsymbol{A}$, and $\boldsymbol{A}^\dagger$ is called the pseudo inverse of $\boldsymbol{A}$ (as $\boldsymbol{A}^\dagger \boldsymbol{A} = \boldsymbol{I}_d$).

## 3.2 Methods for Solving LS Problems

In practice, one rarely uses the close-form expression $(\boldsymbol{A}^\top \boldsymbol{A})^{-1}\boldsymbol{A}^\top \boldsymbol{b}$ to compute the least squares solution $\boldsymbol{x}_{\mathsf{ls}}$, as it involves forming and inverting the matrix $\boldsymbol{A}^\top \boldsymbol{A}$, which is costly and unnecessary.

Instead, one typically uses one of the following two main classes of method for computing $\boldsymbol{x}_{\mathsf{ls}}$.

**Direct methods:** these are variants of Gaussian elimination. The computational complexity is roughly $O(nd^2)$. Examples of direct methods include:

- *Cholesky decomposition:* compute upper triangular matrix $\boldsymbol{R}$ s.t. $\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{R}^\top \boldsymbol{R}$, and solve $\boldsymbol{R}^\top \boldsymbol{R}\boldsymbol{x} = \boldsymbol{A}^\top \boldsymbol{b}$
- *QR decomposition:* compute QR decomposition $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$ ($\boldsymbol{Q}$: orthonormal; $\boldsymbol{R}$: upper triangular), and solve $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{Q}^\top \boldsymbol{b}$

Direct methods yield high accuracy solutions. These methods are often used on dense $\boldsymbol{A}$.

**Iterative methods:** Example include *Conjugate gradient* and variants. The computational complexity is roughly $O\big(nd \cdot \frac{\sigma_{\max}(\boldsymbol{A})}{\sigma_{\min}(\boldsymbol{A})} \log \frac{1}{\varepsilon}\big)$, which is linear in $d$ and depends logarithmically on the target accuracy $\epsilon$. Such methods allow one to tradeoff between accuracy and computation cost. They are often used on large sparse $\boldsymbol{A}$, for which matrix-vector product can be computed in time linear in the sparsity of $\boldsymbol{A}$. However, due to the dependence on the condition number $\frac{\sigma_{\max}(\boldsymbol{A})}{\sigma_{\min}(\boldsymbol{A})}$, iterative methods may be slow on ill-conditioned problems.

We will instead look at **randomized methods**. The high-level goal is to achieve sample complexity of the form $\widetilde{O}\big(nd/\epsilon^2\big)$, which is linear in $d$ and (hopefully) independent of the condition number of $\boldsymbol{A}$. That is, randomized algorithms have better dependence on $d$ compared to direct methods, and are still fast on ill-conditioned problem compared to iterative methods. The price we pay is a worst dependence on the target accuracy $\epsilon$.

## 3.3 Randomized LS Approximation

The **Basic idea** is to generate a sketching / sampling matrix $\boldsymbol{\Phi} \in \mathbb{R}^{r\times n}$ (with $r \ll n$, e.g., via random sampling, random projection), and solve a smaller LS problem

$$\widetilde{\boldsymbol{x}}_{\mathsf{ls}} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \quad \|\boldsymbol{\Phi}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\|_2 \tag{6}$$

For example, if $\boldsymbol{\Phi}$ is a subsampling matrix of the form

$$\boldsymbol{\Phi} \in \mathbb{R}^{2\times n} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix},$$

then

$$\boldsymbol{\Phi}\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{1,:} \\ \boldsymbol{A}_{n,:} \end{bmatrix} \in \mathbb{R}^{2\times d}$$

picks out two rows of $\boldsymbol{A}$.

Our **goal** is to find $\boldsymbol{\Phi}$ such that $\widetilde{\boldsymbol{x}}_{\mathsf{ls}}$ approximates $\boldsymbol{x}_{\mathsf{ls}}$ in turns of the estimation error and fitting error, that is,

$$\begin{aligned} \widetilde{\boldsymbol{x}}_{\mathsf{ls}} &\approx \boldsymbol{x}_{\mathsf{ls}}, \\ \|\boldsymbol{A}\widetilde{\boldsymbol{x}}_{\mathsf{ls}} - \boldsymbol{b}\|_2 &\approx \|\boldsymbol{A}\boldsymbol{x}_{\mathsf{ls}} - \boldsymbol{b}\|_2. \end{aligned}$$

Before we proceed, we consider two extreme cases for the choice of $|\boldsymbol{\Phi}$.

- $\boldsymbol{\Phi} = \boldsymbol{I}_{n\times n}$. This $\boldsymbol{\Phi}$ is easy to compute, but it does not change the LS problem and hence has no improvement for computational cost.
- $\boldsymbol{\Phi} = \boldsymbol{U}_A^\top$. In this case, we have
  $$\Phi\boldsymbol{A} = \boldsymbol{U}_A^\top \boldsymbol{U}_A \boldsymbol{\Sigma} \boldsymbol{V}_A^\top = \underbrace{\boldsymbol{\Sigma}\boldsymbol{V}_A^T}_{\boldsymbol{A}'}.$$
  Therefore, the coefficient matrix $\boldsymbol{\Phi}\boldsymbol{A} = \boldsymbol{A}'$ for the subsampled LS problem (6) is product of diagonal and orthonormal matrix. The corresponding normal equation
  $$\underbrace{\boldsymbol{A}'^\top \boldsymbol{A}'}_{\boldsymbol{V}_A \boldsymbol{\Sigma}_A^2 \boldsymbol{V}_A^\top} \widetilde{\boldsymbol{x}}_{\mathsf{ls}} = \boldsymbol{A}'^T \boldsymbol{b}'$$
  is easy to solve using direct methods. Moreover, the solution $\widetilde{\boldsymbol{x}}_{\mathsf{ls}} = \boldsymbol{x}_{\mathsf{ls}}$ is exact. However, using such $\boldsymbol{\Phi}$ requires computing SVD of $\boldsymbol{A}$, which is at least as hard as the original LS problem.

Our goal is to get the best of both worlds above, i.e., choose a $\boldsymbol{\Phi}$ that is easy to compute and leads to a subsampled LS problem that is easier to solve than the original problem.

# References

[Magen and Zouzias, 2011] Magen, A. and Zouzias, A. (2011). Low rank matrix-valued chernoff bounds and approximate matrix multiplication. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1422–1436. SIAM.

[Mahoney, 2016] Mahoney, M. W. (2016). Lecture notes on randomized linear algebra. *arXiv preprint arXiv:1608.04481*.