LINEAR ALGEBRA

MARIA CAMERON

Contents

1. Linear Algebra I: theory and conditioning

References:

- D. Bindel's and J. Goodman's book "Principles of Scientific Computing", Chapter 4.
- J. Demmel, "Applied Numerical Linear Algebra", Section 1.7 (vector and matrix norms) and Chapter 3.

Linear algebra is one of the most important tools of modern computational science. In recent years, the importance of numerical linear algebra has increased due to the need to solve large-scale problems arising in data science. For example, numerous personal recommendations that you encounter in such services as Netflix, Amazon, etc., are obtained for you by solving certain large-scale optimization problems with algorithms heavy on the use of linear algebra. New methods for solving large-scale linear algebra problems have been developed in recent years. These include, e.g., the butterfly algorithm for fast Fourier transform, fast direct algorithms for solving structured linear systems, etc.

The operations of linear algebra include but are not limited to:

- solving linear systems of algebraic equations;
- finding subspaces;
- matrix factorization (PLU, QR, SVD, CUR, etc);
- solving least squares problems;
- computing eigenvectors and eigenvalues.

In this section, we will go over the aspect of linear algebra that you should know as a user of linear algebra software: basic concepts, basic theory, and conditioning. The last item is extremely important as you should be aware of what can go wrong when you are using some standard linear algebra operations.

There are publicly available linear algebra libraries on low-level languages: `clapack` (C/C++), `lapack` (Fortran). Matlab contains excellent linear algebra commands for both dense and sparse matrices.

Standard linear algebra algorithms are *backward stable*. This means that the output of any standard linear algebra algorithm is as accurate as the *condition number* for the problem allows. Recall that

- an algorithm is backward stable if its output is the exact answer for a slightly perturbed input;
- the condition number for the problem is the strict upper bound for the ratio of the relative error in the output to the relative error in the input that caused it.

This means, in particular, that the error produced by a backward stable algorithm can be large if the condition number of the problem being solved is large.

We start with reviewing the basic concepts of linear algebra.

1.1. **Vector spaces.** Typically we are happy with the results of any numerical algorithm if the produced error is small. If the error is multi- or infinite-dimensional, in order to say that it is small, we need some reasonable way to convert it to a single nonnegative number and compare it with some threshold. An appropriate vector norm often serves this purpose.

Definition 1. *A vector space V is a set closed with respect to the operations of addition "$+$": $V \times V \to V$, and scalar multiplication "$\cdot\alpha$": $V \to V$. These operations satisfy the following properties.*

(1) $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$,
(2) $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$,
(3) $\alpha(\mathbf{a} + \mathbf{b}) = \alpha\mathbf{a} + \alpha\mathbf{b}$,
(4) $(\alpha + \beta)\mathbf{a} = \alpha\mathbf{a} + \beta\mathbf{a}$,
(5) there is $\mathbf{0} \in V$ s.t. $\mathbf{a} + \mathbf{0} = \mathbf{a}$ for any $\mathbf{a} \in V$,
(6) for any $\mathbf{a} \in V$ there is $(-\mathbf{a}) \in V$ s.t. $\mathbf{a} + (-\mathbf{a}) = 0$,
(7) $\alpha(\beta\mathbf{a}) = (\alpha\beta)\mathbf{a}$,
(8) $1\mathbf{a} = \mathbf{a}$ for any $\mathbf{a} \in V$.

Exersise 1. *Prove that $0\mathbf{a} = \mathbf{0}$ for any $\mathbf{a} \in V$.*

Below we remind some basic concepts. Please read Sections 4.2.1 and 4.2.2 in Bindel and Goodman for more details.

- A *subspace* W of a vector space V is a subset of V that is a vector space itself with respect to the same operations as in V, i.e., W is closed under addition and scalar multiplication: for any $w_1, w_2 \in W$ and $\alpha \in \mathbb{R}$ or $\mathbb{C}$, $w_1 + w_2 \in W$ and $\alpha w_1 \in W$. Therefore, to check if W is a subspace, it suffices to check if it is closed under addition and scalar multiplication. The properties of the operations are inherited for those in V.
- The span of vectors v_1, ..., v_n in V is the set of their all possible linear combinations.
- We say that vectors v_1, ..., v_n are *linearly independent* if any their zero linear combination implies that all of its coefficients are zero.
- A *basis* of V is a subset of vectors $\{b_i\}_{i\in\mathcal{I}}$ such that:
  (1) any $v \in V$ can be represented as

$$v = \sum_{i\in\mathcal{I}} \alpha_i b_i,$$

(2) and the $\{b_i\}_{i\in\mathcal{I}}$ is minimal in the sense such that for any $m \in \mathcal{I}$ one can find $v \in V$ such that

$$v - \sum_{i\in\mathcal{I}\setminus m} \alpha_i b_i \neq 0$$

for any set of values of α_i, $i \in \mathcal{I}\setminus\{m\}$.

Recall a theorem in linear algebra saying that if $\{b_i\}_{i=1}^n$ is a basis in V, then any other basis in V also has n vectors.

- If the number of vectors in a basis of V is finite, this number is called the *dimension* of V. Otherwise, the vector space is *infinitely dimensional*.
- A *linear transformation* or a linear map for a vector space V to a vector space W is a map $L : V \to W$ such that for any $v_1, v_2 \in V$ and any $\alpha \in \mathbb{R}$ or $\mathbb{C}$

$$L(v_1 + v_2) = L(v_1) + L(v_2) \quad \text{and} \quad L(\alpha v_1) = \alpha L(v_1).$$

Let $\mathcal{B} = \{b_i\}$ be a basis in V and $\mathcal{E} = \{e_i\}$ be a basis in W. Then by linearity, we have:

$$L(v) = L\left(\sum_j v_j b_j\right) = \sum_j v_j L(b_j) = \sum_j v_j \sum_i a_{ij} e_i \quad \text{where} \quad L(b_j) = \sum_i a_{ij} e_i.$$

Therefore, we can define the matrix of the linear transformation

$$A =_{\mathcal{E}} [L]_{\mathcal{B}} = (a_{ij}).$$

Hence, the columns of the matrix of linear transformation from V to W are the images of the basis vectors in V written in the basis in W.

- A matrix product AB is defined if and only if the number of columns in A is equal the number of rows in B. The matrix product AB corresponds to a composition of linear transformations with matrices A and B. Matrix multiplication is associative but not commutative.
- For a matrix $A = (a_{ij})$ the *transpose* is defined by $A^\top := (a_{ji})$. If A has complex entries, than its *adjoint* is defined as its transpose with complex conjugation: $A^* := (\bar{a}_{ji})$.

Now let us list some examples illustrating these concepts.

Example (1) $\mathbb{R}^n$ is an n-dimensional vector space. Its standard basis is

$$\mathcal{E} = \{e_1, e_2, \ldots, e_n\} = \left\{ \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \right\}.$$

The subset V_0 defined as

$$V_0 := \left\{ v \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^n v_i = 0 \right\}$$

is an $(n-1)$-dimensional subspace of $\mathbb{R}^n$, while the subset

$$V_1 := \left\{ v \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} v_i = 1 \right\}$$

is not a subspace as it is not closed under addition and scalar multiplication.

(2) The set of polynomials of degree $\leq n$ denoted by $\mathcal{P}_n$ is an $(n+1)$-dimensional vector space. One basis for $\mathcal{P}_n$ is the set

$$\mathcal{X} := \{1, x, \ldots, x^n\}.$$

(3) An example of linear transformation from $\mathcal{P}_n$ to $\mathcal{P}_{n-1}$ is the differentiation:

$$\frac{d}{dx} : \mathcal{P}_n \to \mathcal{P}_{n-1}.$$

Its matrix in the basis $\mathcal{X}$ is

$$D_{\mathcal{X}} := \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 2 & \ldots & \\ & & & \ddots & \\ 0 & \ldots & & 0 & n \end{bmatrix}.$$

If we pick another basis, for example, Chebyshev's basis, the differentiation matrix will be different.

(4) Examples of infinite-dimensional spaces are
- $\mathcal{P}$, the space of all polynomials,
- $C(a,b)$, the space of all continuous functions on an interval (a,b),
- the space of all continuous functions on $[a,b]$ satisfying the homogeneous boundary conditions $f(a) = f(b) = 0$.

1.2. **Vector norms.**

Definition 2. *Norm is a function defined on a vector space V:*

$$\mathcal{N} : V \longrightarrow \overline{\mathbb{R}}_+ \equiv [0, +\infty]$$

such that

(1) $\|\mathbf{a}\| \geq 0,\ \|\mathbf{a}\| = 0$ iff $\mathbf{a} = \mathbf{0}$,
(2) $\|\alpha \mathbf{a}\| = |\alpha| \|\mathbf{a}\|$,
(3) $\|\mathbf{a} + \mathbf{b}\| \leq \|\mathbf{a}\| + \|\mathbf{b}\|$.

Example The space of continuous functions on the interval $[a,b]$ with the maximum norm

$$V = C([a,b]),\ \|f\| = \sup_{[a,b]} |f(x)|.$$

If the interval is finite, $\|f\| = \max_{[a,b]} |f(x)|$.

Example The space of continuous functions on the interval $[a,b]$ with the maximum norm

$$V = L_p([a,b]),\ \|f\| = \left(\int_a^b |f(x)|^p dx\right)^{1/p}.$$

Example The space $V = l_p$ of all sequences $\{a_k\}_{k=1}^{\infty}$ such that

$$\|\{a\}\|_p := \left(\sum_{k=1}^{\infty} |a_k|^p\right)^{1/p} < \infty.$$

In particular, l_1 is the space of all absolutely convergent sequences as

$$\|\{a\}\|_1 := \sum_{k=1}^{\infty} |a_k| < \infty.$$

Example The space $V = l_\infty$ of all sequences $\{a_k\}_{k=1}^{\infty}$ such that

$$\|\{a\}\|_\infty := \sup_k |a_k| < \infty.$$

In other words, l_∞ is the space of all bounded sequences.

The concept of orthogonality is generalized to vector spaces via the notion of the inner product.

Definition 3. *An inner product is a function* $(\cdot,\cdot) : V \times V \longrightarrow \mathbb{R}$ *or* $\mathbb{C}$ *satisfying*

(1) $(\mathbf{a},\mathbf{a}) \geq 0,\ (\mathbf{a},\mathbf{a}) = 0$ iff $\mathbf{a} = \mathbf{0}$,
(2) $(\mathbf{a},\mathbf{b}) = \overline{(\mathbf{b},\mathbf{a})}$,
(3) $(\mathbf{a},\mathbf{b}+\mathbf{c}) = (\mathbf{a},\mathbf{b}) + (\mathbf{a},\mathbf{c})$,
(4) $(\alpha\mathbf{a},\mathbf{b}) = \alpha(\mathbf{a},\mathbf{b})$.

The norm induced by an inner product is given by $\|f\| = \sqrt{(f,f)}$. The norms that are associated with inner products are especially important.

Example
(1) Let $V = \mathbb{R}^d$. The dot product $v^\top w$ is an inner product.
(2) Let $V = \mathbb{R}^d$ and A be a symmetric positive definite $d \times d$ matrix, i.e., for all $v \in \mathbb{R}^d \backslash \{\mathbf{0}\}$, $v^\top A v > 0$. Then

$$(v,w)_A := w^\top A v$$

is an inner product.
(3) The Legendre inner product:

$$f,g \in L_2([a,b]),\ (f,g) = \int_a^b f(x)\overline{g(x)}dx.$$

(4) the Chebyshev inner product:

$$f,g \in C([-1,1]),\ (f,g) = \int_a^b \frac{f(x)g(x)}{\sqrt{1-x^2}}dx.$$

Suppose we are looking at the error $e(x) = f(x) - p(x)$ where f is a given function and p is its approximation. The function $w := (1-x^2)^{-1/2}$ is the Chebyshev weight function. It puts more weight on points near the ends of the interval, i.e., the error near the ends of the interval contributes more to the norm than the error near its midpoint. This choice of the weight function eliminated the Runge phenomenon of interpolation error being large near the ends of the interpolation interval.

(5) Hermite inner product:

$$f, g \in C([-\infty, \infty]),\ (f, g) = \int_{-\infty}^{\infty} f(x)g(x)e^{-x^2}dx.$$

Suppose we are looking at the error $e(x) = f(x) - p(x)$ where f is a given function and p is its approximation. Only the error around the origin will contribute significantly to the norm.

1.3. **Matrix norm.**

Definition 4. *The norm of a matrix associated with the vector norm $\|\cdot\|$ is defined as*

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}. \tag{1}$$

The geometric sense of the matrix norm is the maximal elongation of a unit vector as a result of the corresponding linear transformation.

Exersise 2. *Let $A = (a_{ij})$ be an $m \times n$ matrix. Show that then:*

(1) *For the l_1-norm,*

$$\|A\|_1 = \max_j \sum_i |a_{ij}|,$$

i.e., the maximal column sum of absolute values. Find the maximizing vector.

(2) *For the max-norm or l_∞-norm*

$$\|A\|_{\max} = \max_i \sum_j |a_{ij}|,$$

i.e., the maximal row sum of absolute values. Find the maximizing vector.

To solve this exercise, first find an upper bound for $\|A\|$, then find a maximizing vector and show that this bound is achieved on this vector.

1.4. **Eigenvalues and eigenvectors.** Finding eigenvalues and eigenvectors is very useful in many different contexts. For example, the general analytic solution to a linear system of ODEs $\dot{x} = Ax$ is often written in terms of eigenvalues and eigenvectors of A. The 2-norm of A is expressed in terms of eigenvalues of $A^\top A$. The eigenvectors corresponding to maximal eigenvalues serve as principle components in principal component analysis and diffusion maps.

1.4.1. *Diagonalizable matrices.* Recall that an $n \times n$ matrix A is called *diagonalizable* if it has n linearly independent eigenvectors. In this case, A can we written as

$$A = R\Lambda R^{-1} \equiv R\Lambda L = \begin{bmatrix} r_1 & r_2 & \cdots & r_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \begin{bmatrix} l_1 & \rightarrow \\ l_2 & \rightarrow \\ \vdots & \\ l_n & \rightarrow \end{bmatrix}. \tag{2}$$

The columns of R are the *right eigenvectors* of A. They satisfy:

$$Ar_j = \lambda_j r_j.$$

The rows of $L := R^{-1}$ are the *left eigenvectors* of A satisfying

$$l_j A = \lambda_j l_j.$$

Even if A is real, eigenvectors and eigenvalues do not need to be real. They are complex in the general case.

1.4.2. *Symmetric matrices.*

Proposition 1. *If A is real and symmetric, there exists an orthonormal basis of real eigenvectors. Moreover, the eigenvalues are real, and the eigenvectors corresponding to distinct eigenvalues are orthogonal.*

Proof. We will only prove that the eigenvalues are real and the eigenvectors corresponding to distinct eigenvalues are orthogonal. The Existence of an orthonormal basis follows from the Jordan Decomposition theorem proved in MATH405.

Let λ be an eigenvalue, and r be the corresponding unit right eigenvector. Then $r^* := \bar{r}^\top$ is the left eigenvector for $\bar{\lambda}$. Indeed, since A is real and symmetric, we have:

$$Ar = \lambda r, \text{ hence } (Ar)^* = (\lambda r)^*, \text{ i.e. } r^* A = \bar{\lambda} r^*,$$

which shows that r^* is the left eigenvector for $\bar{\lambda}$. Then, sandwiching A between r^* and r we get

$$r^* A r = r^*(Ar) = \lambda r^* r = \lambda \|r\|^2 = \lambda.$$

On the other hand, applying A to r^*, we get

$$r^* A r = (Ar)^* r = \bar{\lambda} r^* r = \bar{\lambda} \|r\|^2 = \bar{\lambda}.$$

Therefore, $\lambda = \bar{\lambda}$, i.e., λ is real.

Now we show that eigenvectors corresponding to distinct eigenvalues are orthogonal. Let $Ar_1 = \lambda_1 r_1$ and $Ar_2 = \lambda_2 r_2$ with $\lambda_1 \neq \lambda_2$. Then

$$r_1^* A r_2 = \lambda_1 r_1^* r_2 = \lambda_2 r_1^* r_2.$$

Since $\lambda_1 \neq \lambda_2$, $r_1^* r_2$ must be zero. Hence r_1 and r_2 are orthogonal.

□

Note that we can always pick real eigenvectors for real eigenvalues of a real symmetric matrix.

Exersise 3. *Let $A = (a_{ij})$ be an $m \times n$ matrix. Show that then for the vector l_2-norm,*

$$\|A\|_2 = \sqrt{\rho(A^\top A)}.$$

Solution. Recall that the vector 2-norm is given by $\|x\|_2 = \sqrt{x^\top x}$. Using this we get

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2 = \max_{x^\top x=1} \sqrt{x^\top A^\top A x}.$$

Since $A^\top A$ is symmetric, its eigendecomposition is given by

$$A^\top A = U \Lambda U^\top,$$

where U is an orthogonal matrix (i.e., $U^\top U = U U^\top = I$, or $U^\top = U^{-1}$) whose columns are the eigenvectors of A, and Λ is a diagonal matrix whose diagonal entries are the corresponding eigenvalues. Using this we continue:

$$\|A\|_2 = \max_{x^\top x=1} \sqrt{x^\top U \Lambda U^\top x} = \max_{x^\top x=1} \sqrt{(U^\top x)^\top \Lambda (U^\top x)}.$$

Now we note that

$$\|x\|_2 = \|U^\top x\|_2$$

because

$$\|x\|_2^2 = x^\top x = x^\top U U^\top x = (U^\top x)^\top (U^\top x) = \|U^\top x\|^2.$$

Let us denote $U^\top x$ by y. Then

$$\|A\|_2 = \max_{y^\top y=1} \sqrt{y^\top \Lambda y} = \max_{y^\top y=1} \sqrt{y_1^2 \lambda_1 + y_2^2 \lambda_2 + \ldots + y_n^2 \lambda_n} = \max_{j=1,\ldots,n} \sqrt{|\lambda_n|} \equiv \sqrt{\rho(A^\top A)}.$$

Remark If A is a square real symmetric matrix, then the eigenvalues of $A^\top A$ are squares of the eigenvalues of A. Hence the 2-norm of A is the spectral radius of A:

$$\|A\|_2 = \max_i |\lambda_i| = \rho(A).$$

1.4.3. *Defective matrices and the Jordan form.* If matrix is not diagonalizable, it is called *defective*. An example of such a matrix is

$$A = \begin{bmatrix} 1 & 10 \\ 0 & 1 \end{bmatrix}. \tag{3}$$

This matrix has eigenvalue 1 of algebraic multiplicity 1 and just one eigenvector $[1, 0]^\top$. In theoretical linear algebra, the Jordan canonical form J is often considered for such matrices. The Jordan decomposition of A is

$$A = V J V^{-1} \tag{4}$$

where columns of V form the *Jordan basis* and J is a block-diagonal matrix with blocks of the form

$$J_j := \begin{bmatrix} \lambda_j & 1 & & & \\ & \lambda_j & 1 & & \\ & & \ddots & \ddots & \\ & & & \lambda_j & 1 \\ & & & & \lambda_j \end{bmatrix}.$$

There is a unique eigenvector v_j corresponding to each block.

Exersise 4. *Find the Jordan form and the Jordan basis for the matrix in* (3).

In numerical linear algebra, the Jordan form is rarely computed. The reason is that it is unstable with respect to small perturbations of A. For example, consider a 16×16 matrix A

$$A := \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ & & & & 0 \end{bmatrix}. \tag{5}$$

It is already in the Jordan form consisting of a single block, and its unique eigenvalue of algebraic multiplicity 16 is zero. Indeed,

$$\det(\lambda I - A) = \lambda^{16} = 0.$$

Now consider a perturbation of A such that the zero at its bottom left corner is replaced with 10^{-16}:

$$A + \delta A := \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ 10^{-16} & & & & 0 \end{bmatrix}. \tag{6}$$

The eigenvalues of $A + \delta A$ are the roots of

$$\det(\lambda I - A) = \lambda^{16} - 10^{-16} = 0.$$

There are 16 distinct complex eigenvalues located at the corners of the 16-gon in the complex plane:

$$\lambda_k = 0.1 e^{i2\pi k/16}, \quad k = 0, 1, \dots 15.$$

Hence, the Jordan form of A will be $\mathrm{diag}\{\lambda_0, \dots, \lambda_{15}\}$ which is not close to (6). Thus, we see that a perturbation of the size of the machine epsilon has a dramatic effect on the Jordan form and on the magnitudes of the eigenvalues of A.

1.4.4. *The Schur form.* For reasons indicated in Section 1.4.3 the Jordan form of a matrix is rarely computed. Another eigenvalue revealing form is much more preferable. This is the Schur form defined by:

$$A = QTQ^\top$$

where T is upper-triangular,

$$T = \begin{bmatrix} \lambda_1 & t_{12} & t_{13} & \dots & t_{1n} \\ & \lambda_2 & t_{23} & \dots & t_{2n} \\ & & \ddots & \ddots & \\ & & & \lambda_{n-1} & t_{n-1,n} \\ & & & & \lambda_n \end{bmatrix}.$$

and Q is orthogonal (or unitary if it is complex), i.e., its columns form an orthonormal basis, or $Q^*Q = I$. Often it is more preferable to deal with the so-called real Schur form in which complex pairs of eigenvalues form 2×2 blocks along the diagonal of T. Then both Q and T are real. The Matlab command to compute the Schur form is

```
A = rand(10);
[Q,T] = schur(A);
```

If A is real, this command computes the real Schur form. If you would like the complex Schur form, type

```
[Q,T] = schur(A,'complex');
```

Exersise 5. *Let $u + iv$ be a complex eigenvector of a real matrix A, and $\mu + i\nu$ be the corresponding eigenvalue. Show that*

$$A[u, v] = [u, v] \begin{bmatrix} \mu & \nu \\ -\nu & \mu \end{bmatrix}, \tag{7}$$

i.e., the vectors u and v span a 2-dimensional invariant subspace of A.

1.5. **Normal equations.** Consider an overdetermined system of linear equations

$$Ax = b, \quad A_{m\times n}, \quad m \geq n.$$

Such problems arise, for example, when we want to find a line that best fits measured data points (x_i, y_i), $i = 1, \dots, m$, that ideally lie on a straight line $ax + b = y$. Thus, we set up the following system:

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}. \tag{8}$$

The system (8) typically does not have a solution unless the points happen to lie on the same line. However, we always can find a line $ax + b$ that fits the data best in the least squares sense, the so-called least squares solution.

Definition 5. *We say that x^* is the least squares solution of $Ax = b$, A is $m \times n$, $m \geq n$, if*

$$x^* = \arg\min_{x \in \mathbb{R}^n} \|Ax - b\|.$$

Proposition 2. *Let A be $m \times n$, $m \geq n$, $\mathsf{rank}(A) = n$. Then the least squares solution x^* to $Ax = b$ is given by*

$$x^* = (A^\top A)^{-1} A^\top b. \tag{9}$$

Proof. Note that x^* is the solution of the so-called normal equation that is obtained from $Ax = b$ by multiplication by $A^\top$ from the left. Since the matrix A has full rank, i.e., $\mathrm{rank}(A) = n$, the matrix $A^\top A$ is symmetric positive definite. Write $x = x^* + e$ and consider $\|Ax - b\|^2$. We want to show that it is minimal if and only if $e = 0$, i.e., $x = x^*$.

$$\begin{aligned}
&\|Ax - b\|^2 = (Ax - b)^\top (Ax - b) = (Ax^* + Ae - b)^\top (Ax^* + Ae - b) = \\
&\|Ae\|^2 + \|Ax^* - b\|^2 + 2(Ae)^\top (Ax^* - b) = \\
&\|Ae\|^2 + \|Ax^* - b\|^2 + 2e^\top (A^\top A x^* - A^\top b) = \\
&\|Ae\|^2 + \|Ax^* - b\|^2 \geq \|Ax^* - b\|^2
\end{aligned}$$

The equality occurs if and only if $e = 0$, i.e., the norm $\|Ax - b\|$ is minimal if and only if $x = x^*$ given by Eq. (9). □

1.6. The QR decomposition and Gram-Schmidt Algorithm.

Theorem 1. *Let A be $m \times n$, $m \geq n$. Suppose that A has full column rank. Then there exist a unique $m \times n$ orthogonal matrix Q, i.e., $Q^\top Q = I_{n \times n}$, and a unique $n \times n$ upper-triangular matrix R with positive diagonals $r_{ii} > 0$ such that $A = QR$.*

Proof. The proof of this theorem is given by the Gram-Schmidt orthogonalization process.

Algorithm 1: Gram-Schmidt orthogonalization

Input : matrix $A = [a_1\ a_2\ \ldots\ a_n]$, $m \times n$, $\mathrm{rank}(A) = n$.
Output: orthogonal matrix Q $m \times n$, $Q^\top Q = I_{n \times n}$, and upper-triangular $n \times n$ matrix R with $r_{ii} > 0$.

for $i = 1, \ldots, n$ **do**
  $q_i = a_i$;
  for $j = 1, \ldots, i-1$ **do**
    $\begin{cases} r_{ji} = q_j^\top a_i & \text{CGS} \\ r_{ji} = q_j^\top q_i & \text{MGS} \end{cases}$;
    $q_i = q_i - r_{ji} q_j$;
  end
  $r_{ii} = \|q_i\|$;
  $q_i = q_i / r_{ii}$;
end

Here CGS and MGS stand for the Classic Gram-Schmidt and the Modified Gram-Schmidt respectively. □

Unfortunately the classic Gram-Schmidt algorithm is numerically unstable when the columns of A are nearly linearly dependent. The modified Gram-Schmidt is better but still can result in Q that is far from orthogonal (i.e., $\|Q^\top Q - I\|$ is much larger than the machine ϵ) when A is ill-conditioned. There are numerically stable ways to compute the QR-decomposition, i.e., by using the Householder reflections or Givens' rotations. We will consider the Householder reflections in homework exercises.

Exersise 6. *Show that the least squares solution of $Ax = b$ is given by*

$$x^* = R^{-1}Q^\top b,$$

where $A = QR$ is the QR decomposition of A.

In Matlab, the least squares solution of $Ax = b$ is found by $A\backslash b$. The QR decomposition per se can be obtained by [Q,R]=`qr`(A).

1.7. **Singular Value Decomposition (SVD).** The Singular Value Decomposition is a very useful decomposition. It has numerous practical applications. Examples are image compression and determination of effective dimensionality of a data set.

Theorem 2. *Let A be an arbitrary $m \times n$ matrix with $m \geq n$. Then we can write*

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

where

$$U \text{ is } m \times n \text{ and } U^\top U = I_{n\times n},$$

$$\Sigma = \text{diag}\{\sigma_1, \ldots, \sigma_n\},\ \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0,$$

$$\text{and } V \text{ is } n \times n \text{ and } V^\top V = I_{n\times n}.$$

The columns of U, u_1, ..., u_n, are called left singular vectors. The columns of V, v_1, ..., v_n are called right singular vectors. The numbers σ_1, ..., σ_n are called singular values. If $m < n$, the SVD is defined for $A^\top$.

The geometric sense of this theorem is the following. Let us view the matrix A as a map from $\mathbb{R}^n$ into $\mathbb{R}^m$:

$$A : \mathbb{R}^n \to \mathbb{R}^m,\ x \mapsto Ax.$$

Then one can find orthogonal bases in $\mathbb{R}^n$, v_1, ... ,v_n, and in $\mathbb{R}^m$, u_1, ..., u_m and numbers σ_1, ..., σ_n, such that

$$v_j \mapsto \sigma_j u_j,\ j = 1, \ldots, n.$$

Then for any $x \in \mathbb{R}^n$ we have:

$$\text{if } x = \sum_{j=1}^{n} x_j v_j \text{ then } Ax = \sum_{j=1}^{n} x_j \sigma_j u_j.$$

Proof. We use induction in m and n. We assume that the SVD exists for $(m-1)\times(n-1)$ matrices and prove it for $m\times n$. We assume $A\neq 0$; otherwise we take $\Sigma=0$ and U and V are arbitrary orthogonal matrices.

The basic step occurs when $n=1$ (since $m>n$). We write

$$A = U\Sigma V^\top \text{ with } U=\frac{A}{\|A\|},\quad \Sigma=\|A\|,\ V=1,$$

where $\|\cdot\|$ is the 2-norm.

For the induction step, choose v so that

$$\|v\|=1 \text{ and } \|A\|=\|Av\|>0.$$

Let

$$u=\frac{Av}{\|Av\|},$$

which is a unit vector. Choose $\tilde{U}$ and $\tilde{V}$ so that $U=[u,\tilde{U}]$ and $V=[v,\tilde{V}]$ are $m\times m$ and $n\times n$ orthogonal matrices respectively. Now write

$$U^\top AV = \begin{bmatrix} u^\top \\ \tilde{U}^\top \end{bmatrix}\cdot A\cdot [v\ \tilde{V}] = \begin{bmatrix} u^\top Av & u^\top A\tilde{V} \\ \tilde{U}^\top Av & \tilde{U}^\top A\tilde{V}\end{bmatrix}.$$

Then

$$u^\top Av = \frac{(Av)^\top(Av)}{\|Av\|} = \|Av\| := \sigma$$

and

$$\tilde{U}^\top Av = \tilde{U}^\top u\|Av\| = 0.$$

We claim that $u^\top A\tilde{V}=0$ too because otherwise

$$\sigma=\|A\|=\|U^\top AV\|\geq\|[1,0,\dots,0]U^\top AV\| = \|[\sigma,\ u^\top A\tilde{V}]\|>\sigma,$$

a contradiction. Therefore,

$$U^\top AV = \begin{bmatrix}\sigma & 0\\ 0 & \tilde{U}^\top AV\end{bmatrix} = \begin{bmatrix} u^\top Av & 0\\ 0 & \tilde{A}\end{bmatrix}.$$

Now we apply the induction hypothesis that

$$\tilde{A}=U_1\Sigma_1 V_1^\top.$$

Hence,

$$U^\top AV = \begin{bmatrix}\sigma & 0\\ 0 & U_1\Sigma_1V_1^\top\end{bmatrix} = \begin{bmatrix}1&0\\0&U_1\end{bmatrix}\begin{bmatrix}\sigma&0\\0&\Sigma_1\end{bmatrix}\begin{bmatrix}1&0\\0&V_1\end{bmatrix}^\top$$

or

$$A=\left(U\begin{bmatrix}1&0\\0&U_1\end{bmatrix}\right)\begin{bmatrix}\sigma&0\\0&\Sigma_1\end{bmatrix}\left(V\begin{bmatrix}1&0\\0&V_1\end{bmatrix}\right)^\top,$$

which is our desired decomposition.

□

The SVD has a large number of important algebraic and geometric properties, the most important of which are summarized in the following theorem.

Theorem 3. *Let $A = U\Sigma V^\top$ be the SVD of the $m \times n$ matrix A, $m \geq n$.*

(1) *Suppose A is symmetric and $A = U\Lambda U^\top$ be an eigendecomposition of A Then the SVD of A is $U\Sigma V^\top$ where $\sigma_i = |\lambda_i|$ and $v_i = u_i \text{sign}(\lambda_i)$, where $\text{sign}(0) = 1$.*

(2) *The eigenvalues of the symmetric matrix $A^\top A$ are σ_i^2. The right singular vectors v_i are the corresponding orthonormal eigenvectors.*

(3) *The eigenvectors of the symmetric matrix $AA^\top$ are σ_i^2 and $m - n$ zeroes. The left singular vectors u_i are the corresponding orthonormal eigenvectors for the eigenvalues σ_i^2. One can take any $m - n$ orthogonal vectors as eigenvectors for the eigenvalue 0.*

(4) *If A has full rank, the solution of*

$$\min_x \|Ax - b\| \quad \text{is} \quad x = V\Sigma^{-1}U^\top b.$$

(5)

$$\|A\|_2 = \sigma_1.$$

If A is square and nonsingular , then

$$\|A^{-1}\|_2 = \frac{1}{\sigma_n}.$$

(6) *Suppose*

$$\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0.$$

Then

$$\text{rank}(A) = r,$$
$$\text{null}(A) = \{x \in \mathbb{R}^n \ : \ Ax = 0 \in \mathbb{R}^m\} = \text{span}(v_{r+1}, \ldots, v_n),$$
$$\text{range}(A) = \text{span}(u_1, \ldots, u_r).$$

(7)

$$A = U\Sigma V^\top = \sum_{i=1}^{n} \sigma_i u_i v_i^\top,$$

i.e., A is a sum of rank 1 matrices. Then a matrix of rank $\leq k < n$ closest to A in the sense of the 2-norm is

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^\top, \quad \text{and} \quad \|A - A_k\| = \sigma_{k+1}.$$

Proof. **Proof of item (7) for the matrix 2-norm.** We will use the following notation:

- $V_k = [v_1, \ldots, v_k]$,
- $U_k = [u_1, \ldots, u_k]$,
- $\Sigma_k = \mathsf{diag}\{\sigma_1, \ldots, \sigma_k\}$.

If $M = U_k \Sigma_k V_k^\top$ then

$$\|A - U_k \Sigma_k V_k^\top\|_2 = \left\| \sum_{j=k+1}^{d} \sigma_j u_j v_j^\top \right\|_2 = \sigma_{k+1}.$$

Now let M be an arbitrary rank $\leq k$ matrix. If rank of A is $\leq k$, then taking $M = A$ will zero out any norm of the difference $A - M$. So, assume $\mathsf{rank}(A) = r > k$. The null-space of M has dimension $\geq n - k$. The space spanned by $\{v_1, \ldots, v_{k+1}\}$ has dimension $k + 1$. Hence

$$\mathsf{dim\,null}(M) + \mathsf{dim\,span}(V_{k+1}) \geq n + 1 > n,$$

which means that they have an intersection of dimension ≥ 1. Let

$$\mathbf{x} \in \mathsf{null}(M) \cap \mathsf{span}(V_{k+1}), \quad \|\mathbf{x}\|_2 = 1.$$

Then

$$\|A - M\|_2^2 \geq \|(A - M)\mathbf{x}\|_2^2 = \|U\Sigma V^\top \mathbf{x}\|_2^2 = \|\Sigma V^\top \mathbf{x}\|_2^2 \geq \sigma_{k+1}^2 \|V^\top \mathbf{x}\|_2^2 = \sigma_{k+1}^2.$$

This completes the proof. □

Example This example illustrates the low-rank approximation of a large matrix. The original image is shown in Fig. 1(a). The rank 3, 10, and 20 approximations are shown in Figs. 1 (b), (c), and (d) respectively. The sequence of Matlab commands to create an approximation of rank m for a given image is the following.

```
>> clear all
>> im=imread('IMG_1413.jpg');
>> [m n k]=size(im)
m =        1600
n =        1200
k =     3
>> mimi=zeros(m,n);
>> mimi=sum(im,3);
>> fig=figure;
>> imagesc(mimi)
>> colormap gray
>> set(gca,'DataAspectRatio',[1 1 1])
>> [U S V]=svd(mimi);
>> size(U)
ans =        1600        1600
>> size(V)
ans =        1200        1200
>> size(S)
ans =        1600        1200
>> fig=figure;
>> m=10;
```

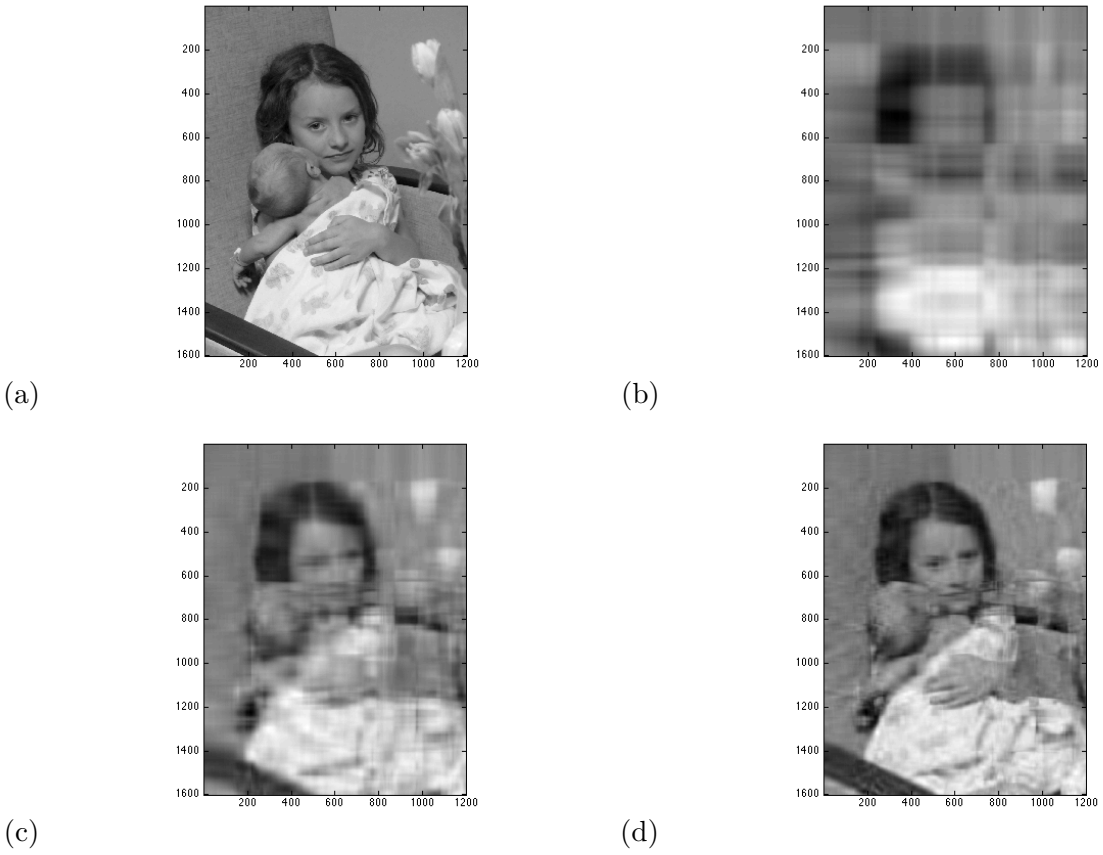

FIGURE 1. Low rank approximations of image. (a): original; (b): rank 3; (c) rank 10; (d) rank 20.

```
>> rm=U(:,1:m)*S(1:m,1:m)*V(:,1:m)';
>> colormap gray
>> set(gca,'DataAspectRatio',[1 1 1])
```

2. Condition number

We start by making the definition of the condition number more precise. Let $f(x)$ be a generally vector-valued function that we need to evaluate. The condition number $\kappa(f;x)$ is the ratio of the relative error in f caused by the relative error in x provided that the change in x is small. Hence, we define κ as

$$\kappa(f;x) := \lim_{\epsilon\to 0} \max_{\|\Delta x\|=\epsilon} \frac{\|f(x+\Delta x)-f(x)\|/\|f(x)\|}{\|\Delta x\|/\|x\|}. \tag{10}$$

2.1. **Condition numbers for differentiable functions.** Let us calculate the condition numbers for differentiable functions. Let $f(x)$ be a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$. Then

$$f(x + \Delta x) = f(x) + \nabla f(x)^\top \Delta x + O(\|\Delta x\|^2).$$

Therefore, the expression for the condition number can be rewritten as follows:

$$\kappa(f; x) = \lim_{\epsilon \to 0} \max_{\|\Delta x\| = \epsilon} \frac{\|x\| |\nabla f(x)^\top \Delta x + O(\|\Delta x\|^2)|}{|f(x)| \|\Delta x\|} = \lim_{\epsilon \to 0} \max_{\|\Delta x\| = \epsilon} \frac{\|x\| |\nabla f(x)^\top \Delta x|}{|f(x)| \|\Delta x\|}$$

The maximum over $\Delta x \in \mathbb{R}^n$ such that $\|\Delta x\| = \epsilon$ is achieved at

$$\Delta x = \frac{\nabla f(x)}{\|\nabla f(x)\|} \epsilon.$$

Therefore,

$$\kappa(f; x) = \frac{\|\nabla f(x)\| \|x\|}{|f(x)|}.$$

Now let $f(x)$ be a differentiable vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$. Then

$$f(x + \Delta x) = f(x) + J(x)\Delta x + O(\|\Delta x\|^2),$$

and J is the jacobian matrix of f with entries:

$$J_{ij}(x) := \frac{\partial f_i}{\partial x_j}.$$

Then

$$\kappa(f; x) = \lim_{\epsilon \to 0} \max_{\|\Delta x\| = \epsilon} \frac{\|x\| \|J(x)\Delta x\|}{\|f(x)\| \|\Delta x\|}.$$

The maximum over $\Delta x \in \mathbb{R}^n$ such that $\|\Delta x\| = \epsilon$ is achieved if Δx is parallel to the first right singular vector v_1 of $J(x) = U\Sigma V^\top$. Therefore,

$$\kappa(f; x) = \frac{\|J\| \|x\|}{\|f(x)\|}.$$

2.2. **Condition number for matrix-vector multiplication.** A particular case is when $f(x)$ is a linear function, i.e., $f(x) = Ax$ where A is an $m \times n$ matrix. Then the Jacobian matrix of f is constant and is equal to A. Hence, the condition number for matrix-vector multiplication is

$$\kappa(A; x) = \frac{\|A\| \|x\|}{\|Ax\|} = \|A\| \frac{\|x\|}{\|Ax\|}. \tag{11}$$

Identity (11) shows that the condition number will be large if

$$\frac{\|Ax\|}{\|x\|} \ll \|A\|,$$

i.e., if there is a vector y that is elongated by A by a much larger factor than x.

Let $A = U\Sigma V^\top$ be an SVD of A. Recall that $\|A\| = \sigma_1$. The best-case scenario: x is parallel to v_1. Then $\frac{\|Ax\|}{\|x\|} = \sigma_1$ and $\kappa(A;x) = 1$. The worst-case scenario: x is parallel to v_n. Then $\frac{\|Ax\|}{\|x\|} = \sigma_n$ and $\kappa(A;x) = \frac{\sigma_1}{\sigma_n}$.

Let us illustrate this phenomenon on a simple example from D. Bindel's and J. Goodman's book "Principles of Scientific Computing", Chapter 4, page 89. Let

$$A = \begin{bmatrix} 1000 & 0 \\ 0 & 10 \end{bmatrix}, \quad \text{and} \quad x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Then

$$Ax = \begin{bmatrix} 0 \\ 10 \end{bmatrix}.$$

Suppose x is perturbed by

$$\Delta x = \begin{bmatrix} \epsilon \\ 0 \end{bmatrix}. \quad \text{Then} \quad A(x + \Delta x) - Ax = A\Delta x = \begin{bmatrix} 1000\epsilon \\ 0 \end{bmatrix}.$$

The error in x is amplified by the factor of 1000 which is 100 times larger than the elongation of x. It is easy to check that for this example, $\kappa(A;x) = 100$.

2.3. **Condition number for solving linear systems.** On the other hand, let us consider the problem of solving a linear system $Ax = b$, i.e., $f(b) = A^{-1}b$. We find:

$$\kappa(A^{-1};b) = \|A^{-1}\|\frac{\|b\|}{\|A^{-1}b\|} = \|A^{-1}\|\frac{\|Ax\|}{\|x\|}. \tag{12}$$

Recall that $\|A^{-1}\| = 1/\sigma_n$, where σ_n is the smallest singular value of A.The condition number for the linear system $Ax = b$ is large if some vector is stretched by A much less than the solution x.

Therefore, the best-case scenario: x is parallel to v_n. Then $\frac{\|Ax\|}{\|x\|} = \sigma_n$ and $\kappa(A;x) = 1$. The worst-case scenario: x is parallel to v_1. Then $\frac{\|Ax\|}{\|x\|} = \sigma_1$ and $\kappa(A;x) = \frac{\sigma_1}{\sigma_n}$.

2.4. **The condition number of a matrix.** The condition number of a matrix A is often defined as

$$\kappa(A) = \|A\|\|A^{-1}\|.$$

This is the worst-case scenario condition number for both matrix-vector multiplication and solving linear systems.

2.5. **Condition numbers for eigenvalue problem.** Ref.: D. Bindel's and J. Goodman's book "Principles of Scientific Computing", Sections 4.2.6 and 4.3.3.

Let A be a square $n \times n$ matrix and (λ_j, r_j) be its jth eigenpair. We will split the problem of finding the condition numbers into four subproblems:

- Condition number for eigenvalue λ_j if A is symmetric;
- Condition number for eigenvalue λ_j if A is not symmetric;
- Condition number for eigenvector r_j if A is symmetric;
- Condition number for eigenvector r_j if A is not symmetric.

Thus, we have: $Ar_j = \lambda_j r_j$. First, we want to evaluate the change in λ_j as A changes to $A + \Delta A$. A useful technique for doing this is the *method of virtual perturbations*. Let ΔA be an arbitrary small perturbation of A. We can connect A and $A + \Delta A$ by a path in the matrix space $\mathbb{R}^{n\times n}$ and parametrize it by its archlength. Then

$$A + \Delta A = A + \dot{A}\Delta t + O((\Delta t)^2) = A + \dot{A}\Delta t + O(\|\Delta A\|^2), \tag{13}$$

where $\dot{A}$ denotes the derivative of A with respect to t. As we move along this path, λ_j and r_j change in a way so that the equality $Ar_j = \lambda_j r_j$ is preserved. For their perturbations, we have:

$$\lambda_j + \Delta\lambda_j = \lambda_j + \dot{\lambda}_j dt + O(\|\Delta A\|^2), \tag{14}$$

$$r_j + \Delta r_j = r_j + \dot{r}_j dt + O(\|\Delta A\|^2). \tag{15}$$

The appeal of this approach is that it allows us to use differentiation rules for finding perturbations. We consider the following chain of identities:

$$Ar_j = \lambda_j r_j; \tag{16}$$

$$\frac{d}{dt}\left(Ar_j\right) = \frac{d}{dt}\left(\lambda_j r_j\right);$$

$$\dot{A}r_j + A\dot{r}_j = \dot{\lambda}r_j + \lambda_j\dot{r}_j. \tag{17}$$

Equation (17) will be a key equation for calculating the desired condition numbers.

2.5.1. *Condition numbers for eigenvalues of symmetric eigenvalue problem.* Since A is symmetric, r_j and λ_j are real. We choose r_j so that $\|r_j\| = 1$ where the norm is the 2-norm. Furthermore, $r_j^\top$ is the left jth eigenvector of A corresponding to eigenvalue λ_j. Multiplication on (17) by $r_j^\top$ on the left gives

$$r_j^\top\dot{A}r_j + r_j^\top A\dot{r}_j = r_j^\top\dot{\lambda}_j r_j + r_j^\top\lambda_j\dot{r}_j;$$

$$r_j^\top\dot{A}r_j + \cancel{\lambda_j r_j^\top\dot{r}_j} = \dot{\lambda}_j r_j^\top r_j + \cancel{\lambda_j r_j^\top\dot{r}_j};$$

$$r_j^\top\dot{A}r_j = \dot{\lambda}_j.$$

Therefore, using (13) and (14), we get

$$\Delta\lambda_j = r_j^\top\Delta A r_j + O(\|\Delta A\|^2) \tag{18}$$

Next, we observe that

$$r_j^\top\Delta A r_j \le \|\Delta A\|\|r_j^\top\|\|r_j\| = \|\Delta A\|.$$

The equality is achieved if $\Delta A = \epsilon r_j r_j^\top$. Therefore, using this bound and (21) we calculate:

$$\left|\frac{\delta\lambda_j}{\lambda_j}\right| \le \frac{\|\Delta A\|}{|\lambda_j|} = \frac{\|\Delta A\|}{\|A\|}\frac{\|A\|}{|\lambda_j|}. \tag{19}$$

Finally, we compute the condition number $\kappa(\lambda_j; A)$ for the jth eigenvalue of A using (22):

$$\boxed{\kappa(\lambda_j; A) = \lim_{\epsilon\to 0} \max_{\|\Delta A\|=\epsilon} \frac{\left|\frac{\delta\lambda_j}{\lambda_j}\right|}{\frac{\|\Delta A\|}{\|A\|}} = \frac{\|A\|}{|\lambda_j|} = \frac{|\lambda_{\max}(A)|}{|\lambda_j|}.} \tag{20}$$

Equation (20) shows that if λ_j is the eigenvalue with the absolute value has condition number 1 and the only reason why $\kappa_j(A)$ can be large is that $|\lambda_{\max}(A)|$ might be much larger than $|\lambda_j|$.

2.5.2. *Condition numbers for eigenvalues of nonsymmetric eigenvalue problem.* If A is not symmetric, its eigenvalues and eigenvectors do not need to be real. Let l_j be the left eigenvector corresponding to λ_j. Multiplying (17) on l_j on the left we get:

$$\begin{aligned} l_j\dot{A}r_j + l_jA\dot{r_j} &= l_j\dot{\lambda_j}r_j + l_j\lambda_j\dot{r_j};\\ l_j\dot{A}r_j + \cancel{\lambda_j l_j\dot{r_j}} &= \dot{\lambda_j}l_jr_j + \cancel{\lambda_j l_j\dot{r_j}};\\ l_j\dot{A}r_j &= \dot{\lambda_j}. \end{aligned}$$

Here we took into account that $l_jr_j = 1$. Using (13) and (14), we get

$$\Delta\lambda_j = l_j\Delta A r_j + O(\|\Delta A\|^2) \tag{21}$$

Now we would like to bound $|l_j\Delta A r_j|$:

$$|l_j\Delta A r_j| \le \|l_j\|\|\Delta A\|\|r_j\|.$$

The equality is achieved if $\Delta A = r_j l_j$. However, contrary to the case where A is symmetric, the norms of r_j and l_j no longer need to be one. The matrices of eigenvectors may be ill-conditioned. Proceeding with the calculation of $\kappa(\lambda_j; A)$ we obtain:

$$\left|\frac{\delta\lambda_j}{\lambda_j}\right| \le \frac{\|\Delta A\|\|l_j\|\|r_j\|}{|\lambda_j|}\|l_j\|\|r_j\| = \frac{\|\Delta A\|}{\|A\|}\frac{\|A\|}{|\lambda_j|}\|l_j\|\|r_j\|; \tag{22}$$

$$\boxed{\kappa(\lambda_j; A) = \lim_{\epsilon\to 0} \max_{\|\Delta A\|=\epsilon} \frac{\left|\frac{\delta\lambda_j}{\lambda_j}\right|}{\frac{\|\Delta A\|}{\|A\|}}\|l_j\|\|r_j\| = \frac{\|A\|}{|\lambda_j|}\|l_j\|\|r_j\|.} \tag{23}$$

Equation (23) shows that there are two reasons for the problem of finding the jth eigenvalue of a nonsymmetric matrix A can be ill-conditioned:

- $|\lambda_j|$ might be much smaller than $\|A\| = \sqrt{\lambda_{\max}(A^\top A)}$;
- the matrix of eigenvectors may be ill-conditioned.

A useful bound for the product $\|l_j\|\|r_j\|$ is given by:

$$\|l_j\|\|r_j\| \le \|R^{-1}\|\|R\| = \frac{\sigma_1(R)}{\sigma_n(R)},$$

where $\sigma_1(R)$ and $\sigma_n(R)$ are the largest and the smallest singular values of R. It is possible to show that if a family of matrices approaches a matrix with a Jordan block, the condition

number of R approaches infinity. Indeed, recall the example in Section 1.4.3. A tiny but specially crafted perturbation changes the set of eigenvalues in a nondifferentiable manner.

2.5.3. *Condition numbers for eigenvectors of symmetric eigenvalue problem.* We start with (17) and expand the perturbation $\dot{r}_j$ in terms of the eigenvectors $\{r_1, \dots, r_n\}$:

$$\dot{r}_j = \sum_{l=1}^{n} m_{jl} r_l.$$

We assume that all eigenvectors are distinct so that A is diagonalizable. Multiplying (17) on $r_k^\top$ on the left we obtain

$$r_k^\top \dot{A} r_j + r_k^\top A \sum_{l=1}^{n} m_{jl} r_l = r_k^\top \dot{\lambda}_j r_j + r_k^\top \lambda_j \sum_{l=1}^{n} m_{jl} r_l.$$

Using the orthonormality of the eigenvectors we get

$$r_k^\top \dot{A} r_j + m_{jk}\lambda_k = \dot{\lambda}_j r_k^\top r_j + m_{jk}\lambda_j.$$

We can assume that $m_{jj} = 0$. If $k \neq j$, we have $r_k^\top r_j = 0$. Hence,

$$r_k^\top \dot{A} r_j = m_{jk}(\lambda_j - \lambda_k).$$

Therefore,

$$m_{jk} = \frac{r_k^\top \dot{A} r_j}{\lambda_j - \lambda_k}.$$

Then Δr_j can be calculated as

$$\Delta r_j = \sum_{k\neq j} m_{jk} r_k = \sum_{k\neq j} \frac{r_k^\top \Delta A r_j}{\lambda_j - \lambda_k} r_k + O(\|\Delta A\|^2).$$

Then

$$\|\Delta r_j\| \leq \|\Delta A\| \sqrt{\sum_{k\neq j} (\lambda_j - \lambda_k)^{-2}}.$$

Hence, the condition number for r_j is

$$\kappa(r_j; A) = \lim_{\epsilon\to 0} \max_{\|\Delta A\| = \epsilon} \frac{\frac{\|\Delta r_j\|}{\|r_j\|}}{\frac{\|\Delta A\|}{\|A\|}} \leq \|A\| \sqrt{\sum_{k\neq j} (\lambda_j - \lambda_k)^{-2}}. \tag{24}$$

This equation shows that the condition number $\kappa(r_j; A)$ is large if there is an eigenvalue close to λ_j.

Exersise 7. *Find an upper bound for the condition number for eigenvector r_j of a non-symmetric matrix A assuming that all its eigenvalues are distinct. In what case will this condition number be large?*

3. Linear Algebra II: algorithms

References:

- D. Bindel's and J. Goodman's book "Principles of Scientific Computing", Chapter 5.
- J. Demmel, "Applied Numerical Linear Algebra", Chapter 2.

In this section, we will get familiar with two very important linear algebra algorithms: solving systems of linear algebraic equations $Ax = b$ via LU factorization with pivoting and computing the Cholesky decomposition for symmetric positive definite matrices $A = LL^\top$ where L is lower triangular. On the example of these two algorithms, we will learn how to estimate the computational cost of linear algebra algorithms via counting *flops* (floating-point operations). Furthermore, we will prove the existence of the Cholesky decomposition and a number of useful properties of symmetric positive definite matrices.

3.1. Solving $Ax = b$ via LU factorization with pivoting.

Throughout this section, we will assume that A is an invertible $n \times n$ matrix. When we solve linear systems $Ax = b$ using paper and pencil, we start with converting the matrix A into a *row echelon form*. This operation often required pivoting. Then we solve the resulting equivalent equation $Ux = c$ with an upper-triangular matrix U using back substitution, i.e., first, we find x_n from $u_{nn}x_n = c_n$, then we find x_{n-1} from $u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n = c_{n-1}$, and so on. Now we would like to convert this approach into a reliable algorithm that finds the solution to $Ax = b$ whenever A is invertible.

3.1.1. *LU without pivoting.*

Converting A into a row echelon form without pivoting is equivalent to computing its LU decomposition, i.e., $A = LU$ where L is unit lower triangular, and U is upper triangular:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & l_{n2} & \dots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ & u_{22} & \dots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix}. \tag{25}$$

It is easy to check by multiplying L and U that the first row of U coincides with the first row of A. Next, l_{21} is found from $l_{21}u_{11} = a_{21}$, i.e., $l_{21} = a_{21}/u_{11}$. Then the second row of U is found by multiplying the second row of L by columns 2 through n of A:

$$l_{21}u_{1k} + u_{2k} = a_{2k}, \quad \text{hence} \quad u_{2k} = a_{2k} - l_{21}u_{1k}, \quad 2 \le k \le n.$$

Then l_{31} and l_{32} are found from the equations obtained by multiplying the third row of L by the first and the second columns of A:

$$l_{31} = a_{31}/u_{11}, \quad l_{32} = \frac{a_{32} - l_{31}u_{12}}{u_{22}}.$$

Then the third row of U is found by multiplying the third row of L by columns 3 through n of A:

$$l_{31}u_{1k} + l_{32}u_{2k} + u_{3k} = a_{3k}, \quad \text{hence} \quad u_{3k} = a_{3k} - l_{31}u_{1k} - l_{32}u_{2k}, \quad 2 \le k \le n.$$

And so on.

Looking at Eq. (25) we observe that both matrices, L and U, can be overwritten on A. Indeed, we do not need to store the diagonal entries of L as we know that they are all 1. We also notice that in calculating LU factorization we can proceed exactly like we did when we were converting A into an echelon form. We compute the first column of L by

$$l_{j1} = \frac{a_{j1}}{a_{11}}, \quad j = 2, \dots, n.$$

Then we modify all entries of A in the submatrix $A_{2:n,2:n}$ according to

$$a_{jk} = a_{jk} - l_{k1}a_{1k}, \quad 2 \le j \le n, 2 \le k \le n.$$

The second row of the modified A starting from the second entry will be the row of U. Then we apply the same procedure to the submatrix $A_{2:n,2:n}$ of the modified A. The resulting LU factorization algorithm implemented in Matlab is displayed below.

```
function [L,U] = simpleLU(A)
n = length(A); % get the dimension of A
for i = 1 : n - 1
    for j = i + 1 : n
        % L(j,i) = new A(j,i)
        A(j,i) = A(j,i)/A(i,i); % one flop: "/"
    end
    for j = i + 1 : n
        for k = i + 1 : n
            % U(j,k) = new A(j,k);
            A(j,k) = A(j,k) - A(j,i)*A(i,k); % two flops: "-" and "*"
        end
    end
end
end
```

3.1.2. *Counting flops for LU.* This code allows us readily calculate the number of flops $W(n)$:

$$W(n) = \sum_{i=1}^{n-1} \left[\sum_{j=i+1}^{n} 1 + \sum_{j=i+1}^{n} \sum_{k=i+1}^{n} 2 \right] = \sum_{i=1}^{n-1} \left[n - i + 2(n-i)^2 \right]. \tag{26}$$

In order to estimate the computational cost, we do not need to compute the number of flops exactly. It suffices to find the term that grows fastest with n. This accuracy is good enough for all practical purposes as different arithmetic operations take different CPU times, and, in our count of flops, we do not take into account logical operations. Hence, even if we calculate the number of flops exactly, it will not give us an exact prediction for the required CPU time. Therefore, we proceed with calculating the number of flops in the simplest possible manner: we approximate sums that are hard to compute with integrals.

We continue the calculation in Eq. (26) as follows:

$$W(n) = \sum_{i=1}^{n-1} \left[n - i + 2(n-i)^2\right] \approx \int_0^n (n-x) + 2(n-x)^2 dx$$

$$= \int_0^n t + 2t^2 dt \approx \frac{2}{3}n^3 + O(n^2). \tag{27}$$

We intentionally wrote the term $n^2/2$ in (27) as $O(n^2)$ because we already introduced an error by replacing the sum with an integral and by replacing $n-1$ with n in the upper limit of the integral for the ease of calculation.

Thus, we conclude that the computational cost of LU factorization is $\frac{2}{3}n^3 + O(n^2)$.

3.1.3. *Need for pivoting.* It is easy to see that the LU algorithm in Section 3.1.1 can encounter division by zero even if the matrix A is invertible. For example, consider the matrix

$$A = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}.$$

To eliminate this problem, we will do *row pivoting*, i.e., at each value of i in the outer `for`-loop, we will find the largest in absolute value entry among $a_{ii}, a_{i+1,i}, \ldots, a_{n,i}$ and swap the row containing this entry with row i. We will also need to swap the corresponding entries of the right-hand side b. The LU algorithm with row pivoting implemented as a Matlab script is shown below. It computes the LU factorization of PA where P is a permutation matrix with 1 in row i located in column $p(i)$. The same permutation is applied to the vector b.

```
% LU factorization with pivoting, overwriting L and U on A
for i = 1 : n - 1
    % find pivot
    v = abs(A(i,i));
    for k = i+1 : n
        vtemp = abs(A(k,i));
        if vtemp > v
            v = vtemp;
            p(i) = k;
        end
    end
    % p(i) is the index of the row with max abs value A(k,i)
    % swap rows p(i) and (i)
    for k = 1 : n
        temp(k) = A(p(i),k);
        A(p(i),k) = A(i,k);
        A(i,k) = temp(k);
    end
    btemp = b(p(i));
```

```
    b(p(i)) = b(i);
    b(i) = btemp;
    % LU algorithm
    for j = i + 1 : n
        A(j,i) = A(j,i)/A(i,i); % L(j,i) = new A(j,i)
    end
    for j = i + 1 : n
        for k = i + 1 : n
            A(j,k) = A(j,k) - A(j,i)*A(i,k); % U(j,k) = new A(j,k)
        end
    end
end
```

3.1.4. *Solving LUx=b.* Once the LU factorization of A (or PA) is computed, we solve $LUx = b$ (or $LU = Pb$) in two stages. We introduce a new variable $y := Ux$ and solve $Ly = b$ (or Pb) taking an advantage of the fact that L is lower triangular by computing one entry of y in a time starting from y_1. Note that in the code below `A(i,j) = L(i,j)` as `L` is overwritten on the subdiagonal part of `A`.

```
% Solve Ly = b
y = zeros(n,1);
for i = 1 : n
    rhs = b(i);
    for j = 1 : i - 1
        rhs = rhs - A(i,j)*y(j); % 2 flops
    end
    y(i) = rhs;
end
```

After that, x is found by solving $Ux = y$ also by finding one entry of x in a time starting from x_n:

```
x = zeros(n,1);
% back substitution Ux = y
for i = n : -1 : 1
    rhs = y(i);
    for j = i + 1 : n
        rhs = rhs - A(i,j)*x(j); % two flops
    end
    x(i) = rhs/A(i,i); % one flop
end
```

Let us count the number of flops. For solving $Ly = b$ we get:

$$\texttt{Ly} = \texttt{b}: \quad \sum_{i=1}^{n}\sum_{j=1}^{i-1} 2 = \sum_{i=1}^{n} 2(i-1) = 2\frac{n(n-1)}{2} = n^2 - n \tag{28}$$

$$\texttt{Ux} = \texttt{y}: \quad \sum_{i=1}^{n}\left[1 + \sum_{j=1}^{i-1} 2\right] = \sum_{i=1}^{n}(2i-1) = 2\frac{n(n+1)}{2} - n = n^2. \tag{29}$$

Therefore, the overall cost of solving $LUx = b$ is $2n^2 - n$ which is much lower than $2n^3/3 + O(n^2)$, the cost of LU factorization. Hence, the total cost of solving $Ax = b$ via LU factorization is $2n^3/3 + O(n^2)$.

3.2. **Cholesky decomposition.**

Definition 6. *An $n \times n$ matrix A is symmetric positive definite (abbreviation* SPD*) if*

$$A^\top = A \quad \text{and} \quad x^\top Ax > 0 \quad \forall x \in \mathbb{R}^n \backslash \{0\}. \tag{30}$$

Symmetric positive definite matrices form a special class of matrices arising in many different applications. One of them you have encountered. Recall the normal equation for an overdetermined system of linear algebraic equations $Ax = b$, where A is $m \times n$, $m > n$, and columns of A are linearly independent. The normal equation is obtained by premultiplying $Ax = b$ by $A^\top$: $A^\top Ax = A^\top b$. The matrix $A^\top A$ is symmetric positive definite.

Symmetric positive definite matrices allow us to design more efficient computational algorithms, i.e., algorithms of lower computational cost and superior accuracy. In particular, instead of the LU factorization, it is beneficial to compute the *Cholesky decomposition* $A = LL^\top$ where L is *lower triangular with positive diagonal elements*. The matrix L is called the *Cholesky factor* of A:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & & \ddots & \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \dots & u_{1n} \\ & l_{22} & \dots & u_{n2} \\ & & \ddots & \vdots \\ & & & l_{nn} \end{bmatrix}. \tag{31}$$

As you will see, the computational cost of Cholesky factorization is $n^3/3 + O(n^2)$, approximately twice as low as that of the LU decomposition.

To understand the Cholesky algorithm, let us start calculating the entries of L column-by-column. Multiplying each row of L by the first column of $L^\top$ we find:

$$\begin{aligned} l_{11}^2 = a_{11}, &\quad \text{hence} \quad l_{11} = \sqrt{a_{11}}, \\ l_{21}l_{11} = a_{21}, &\quad \text{hence} \quad l_{21} = a_{21}/l_{21}, \\ &\vdots \\ l_{n1}l_{11} = a_{n1}, &\quad \text{hence} \quad l_{21} = a_{n1}/l_{21}, \end{aligned}$$

Then we multiply the 2nd, 3rd, ..., and nth row of L by the second column of $L^\top$ and find:

$$\begin{aligned} l_{21}^2 + l_{22}^2 = a_{22}, \quad \text{hence} \quad l_{22} &= \sqrt{a_{22} - l_{21}^2}, \\ l_{31}l_{21} + l_{32}l_{22} = a_{32}, \quad \text{hence} \quad l_{32} &= \frac{a_{32} - l_{31}l_{21}}{l_{22}}, \\ &\vdots \\ l_{n1}l_{21} + l_{n2}l_{22} = a_{n2}, \quad \text{hence} \quad l_{n2} &= \frac{a_{n2} - l_{n1}l_{21}}{l_{22}}. \end{aligned}$$

And so on. The Cholesky factorization is summarized in the following code:

```
for j = 1 : n
    L(j,j) = (A(j,j) - sum(L(j,1:j-1).^2))^(1/2);  % 2j-1 flops
    for i = j + 1 : n
        L(i,j) = (A(i,j) - sum(L(i,1:j-1).*L(j,1:j-1)))/L(j,j); % 2j-1 flops
    end
end
```

The total number of flops is:

$$\begin{aligned} &\sum_{j=1}^{n} \left[2j - 1 + \sum_{i=j+1}^{n} (2j-1) \right] = \sum_{j=1}^{n} (n-j+1)(2j-1) \\ &\approx \int_0^n (n-x+1)(2x-1)dx = n^3 - \frac{2}{3}n^3 + O(n^2) = \frac{1}{3}n^3 + O(n^2). \end{aligned} \tag{32}$$

The cheapest way to check if a symmetric matrix is positive definite is by computing its Cholesky factorization. If at any stage of the algorithm the number under the square root is less or equal to zero, the matrix is not symmetric positive definite. Otherwise, it is symmetric positive definite. In the next section, we will prove a number of useful facts about SPD matrices, in particular, that a matrix A is SPD iff there exists a unique Cholesky decomposition $A = LL^\top$ with L having positive diagonal entries.

3.3. **Properties of symmetric positive definite matrices.**

Theorem 4. *If X is nonsingular then A is SPD if and only if $X^\top AX$ is SPD.*

Proof. $\Leftrightarrow$ Let X be nonsingular, and A be SPD. We need to prove that then $X^\top AX$ is SPD. First check that $X^\top AX$ is symmetric. Indeed, since A is symmetric, we have:

$$(X^\top AX)^\top = X^\top A^\top X = X^\top AX.$$

Let $x \in \mathbb{R}^n \backslash 0$. Then $Xx = y \neq 0$ as X is nonsingular. Therefore, since A is SPD, we have:

$$x^\top X^\top AXx = (Xx)^\top A(Xx) = y^\top Ay > 0,$$

Hence $X^\top AX$ is SPD.

$\Leftarrow$ Let $X^\top AX$ be SPD. We want to show that then A is SPD. First check that A is symmetric. We multiply $X^\top AX$ by $X^{-\top}$ and X^{-1} on the left and on the right and get $A = X^{-\top}(X^\top AX)X^{-1}$. Hence, since $X^\top AX$ is symmetric, we get:

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

Therefore, A is symmetric. Now we show that A is positive definite. Let $y \in \mathbb{R}^n \backslash 0$. We define $x = X^{-1}y$ as X is nonsingular and calculate:

$$y^\top Ay = y^\top (X^{-\top}X^\top)A(XX^{-1})y = (X^{-1}y)^\top (X^\top AX)X^{-1}y = x^\top X^\top AXx > 0.$$

Hence A is SPD. □

Theorem 5. *If A is SPD and H is any principal submatrix of A then H is SPD.*

Let us first understand what is a principal submatrix. Let $J \subseteq \{1, \ldots, n\}$ be any nonempty subset of indices. Then a principal submatrix of A corresponding to J is obtained by taking the intersection of columns of A with indices in J and rows of A with indices in J. For example, if $A = (a_{ij})$ is 4×4 SPD and $J = \{1, 3\}$ then the corresponding principal submatrix H is:

$$H = \begin{bmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{bmatrix}.$$

Proof. First, we observe that the principal submatrix H corresponding to a subset of indices $J = \{j_1, \ldots, j_{|J|}\}$ can be obtained from A by sandwiching it with the $|J| \times n$ matrix P whose row i has a single nonzero entry 1 at the position j_i, $1 \le i \le |J|$:

$$H = PAP^\top.$$

Let $x \in \mathbb{R}^{|J|} \backslash \{0\}$. Then $P^\top x = y \in \mathbb{R}^n \backslash \{0\}$ and $y_{j_i} = x_i$, $1 \le i \le |J|$, and the rest of entries of y are zeros. Let us show that $x^\top Hx > 0$:

$$x^\top Hx = x^\top PAP^\top x = (P^\top x)A(P^\top x) = y^\top Ay > 0.$$

□

Theorem 6. *A is SPD if and only if $A = A^\top$ and all its eigenvalues are positive.*

Exersise 8. *Prove this theorem.*

Theorem 7. *If A is SPD then all*

$$a_{ii} > 0 \quad \text{and} \quad \max_{ij} |a_{ij}| = \max_i a_{ii} > 0.$$

Proof. First, we observe that all diagonal entries of A are its 1×1 principal submatrices. Therefore, they must be positive. The second part of the claim of this theorem will be proven from the converse. Assume that

$$\max_{ij} |a_{ij}| = |a_{kl}| \quad \text{with} \quad k \neq l.$$

Let us sandwich A with a specially crafted vector $x := e_k - \mathsf{sign}(a_{kl})e_l$ where e_k and e_l are the standard unit vectors with ones at positions k and l, respectively, and zeros everywhere

else. Denoting the kth and lth columns of A by $A_{:,}k$ and $A_{:,l}$, respectively, using the fact that $a_{kl} = a_{lk}$, and noting that $\mathsf{sign}(a_{lk})a_{lk} = |a_{lk}|$, we calculate:

$$\begin{aligned} x^\top Ax &= [e_k - \mathsf{sign}(a_{kl})e_l]^\top [A_{:,k} - \mathsf{sign}(a_{kl})A_{:,l}] \\ &= a_{kk} - \mathsf{sign}(a_{kl})a_{lk} - \mathsf{sign}(a_{kl})a_{kl} + a_{ll} = a_{kk} + a_{ll} - 2|a_{kl}| < 0. \end{aligned}$$

The last inequality follows from the assumption that $|a_{kl}| > a_{kk}$ and $|a_{kl}| > a_{ll}$. This contradicts to the fact that A is SPD, i.e., $x^\top Ax$ must be positive for any nonzero x. □

Theorem 8. *A is SPD if and only if there is a unique lower triangular matrix L with positive diagonal entries such that $A = LL^\top$.*

Proof. ⟸ Let $A = LL^\top$ with L having positive diagonal elements. Then, A is symmetric as $LL^\top = (LL^\top)^\top$. Since L is nonsingular,

$$x^\top Ax = (x^\top L)(L^\top x) = \|Lx\|_2^2 > 0 \quad \forall x \neq 0.$$

Hence A is SPD.

⟹ Let A be SPD. To show that is admits a Cholesky decomposition, we proceed by induction in n.

Basis. If $n = 1$, $A = a_{11} > 0$, and its Cholesky decomposition exists and is given by $\sqrt{a_{11}}\sqrt{a_{11}}$.

Induction hypothesis: A Cholesky decomposition exists for every SPD matrix A $k \times k$.

Induction step. Let us construct a Cholesky decomposition for an SPD A of size $(k+1) \times (k+1)$. We will write A in a block form

$$A = \begin{bmatrix} a_{11} & A_{12} \\ A_{12}^\top & A_{22} \end{bmatrix}$$

where a_{11} is 1×1, A_{12} is $1 \times k$, and A_{22} is $k \times k$. We wish to decompose A into a product of three matrices with a block-diagonal matrix in the middle of it and a lower-triangular matrix and its transpose being sandwiched around it. Then we can apply the induction hypothesis to the matrix in the middle of this sandwich. We write:

$$A = \begin{bmatrix} a_{11} & A_{12} \\ A_{12}^\top & A_{22} \end{bmatrix} = \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}}A_{12}^\top & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix} \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}}A_{12}^\top & I \end{bmatrix}^\top, \tag{33}$$

where $\tilde{A}_{22}$ needs to be determined. Multiplying these three matrices back together we get:

$$A = \begin{bmatrix} a_{11} & A_{12} \\ A_{12}^\top & A_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & A_{12} \\ A_{12}^\top & \tilde{A}_{22} + \frac{1}{a_{11}}A_{12}^\top A_{12} \end{bmatrix}.$$

Hence

$$\tilde{A}_{22} = A_{22} - \frac{1}{a_{11}}A_{12}^\top A_{12}.$$

We need to show that $\tilde{A}_{22}$ is SPD. Indeed, by Theorem 4, the matrix

$$\begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix} \quad \text{is SPD.}$$

Then, by Theorem 5, its principal submatrix $\tilde{A}_{22}$ is SPD. Therefore, there exists a Cholesky decomposition of $\tilde{A}_{22}$ by induction assumption: $\tilde{A}_{22} = \tilde{L}\tilde{L}^\top$. Plugging in this decomposition for $\tilde{A}_{22}$ in (33) we get:

$$A = \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}} A_{12}^\top & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L} \end{bmatrix}^\top \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}} A_{12}^\top & I \end{bmatrix}^\top$$

$$= \left(\begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}} A_{12}^\top & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L} \end{bmatrix} \right) \left(\begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}} A_{12}^\top & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L} \end{bmatrix} \right)^\top. \tag{34}$$

Noting that the matrices in the parentheses are the products of two lower-triangular matrices with positive diagonal entries, we conclude that these matrices are lower-triangular with positive diagonal entries. I will leave it as an exercise to prove the fact that lower-triangular matrices with positive diagonal entries form a group with respect to matrix multiplication. Hence, the decomposition (34) is a Cholesky decomposition for A of size $(k+1) \times (k+1)$. This proves the induction step.

□

Exersise 9. *Prove that lower triangular matrices form a group with respect to matrix multiplication.* Hint: to prove that the inverse of a lower triangular matrix is lower triangular, you can use induction in n.

Exersise 10. *Prove that the Cholesky decomposition is unique.*

4. Linear Algebra III: Matrix factorization for low-rank approximation

4.1. The full SVD and the truncated SVD.

We have discussed an 'economy' SVD for the case where A was $n \times d$ with $n \geq d$. In the case of $d > n$, 1economy' SVD of A is obtained from the one for $A^\top$. For theoretical purposes, it is often useful to think of the full SVD of A defined as $A = U\Sigma V = A$, where

- U is $n \times n$, $UU^\top = U^\top U = I$,
- Σ is $n \times d$ with the top left submatrix being $\mathsf{diag}\{\sigma_1, \ldots, \sigma_d\}$ and all entries in the rest of Σ are zeros, and
- V is $d \times d$, $VV^\top = V^\top V = I$.

4.2. Ky-Fan norms.

Definition 7. *We say that a function $f : \mathbb{R}^{n \times d} \to \mathbb{R}$ is orthogonally (or unitarily in the complex case) invariant if $f(Q_1 A Q_2) = f(A)$ for any orthogonal (unitary) matrices Q_1 and Q_2.*

Any unitarily invariant function f can be written in terms of singular values of A. Indeed, take $Q_1 = U^\top$ and $Q_2 = V$. Then, it follows from the unitary invariance of f that

$$f(A) = f(\Sigma) = \tilde{f}(\sigma_1, \ldots, \sigma_d).$$

Among the most important unitarily invariant functions are the *Ky-Fan norms*, which are l^p norms of the vectors of singular values. The Ky-Fan norms we care about are:

- The l^∞ Ky-Fan norm is the 2-norm of A (the *operator 2-norm* or the *spectral norm*):

$$\|A\|_2 = \sigma_1 = \max_{1\le i\le d} \sigma_i. \tag{35}$$

- The l^2 Ky-Fan norm is the *Frobenius norm* of A:

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{d} |a_{ij}|^2}. \tag{36}$$

Exersise 11. *Prove that*

$$\|A\|_F^2 = \sum_{i=1}^{d} \sigma_i^2. \tag{37}$$

Hint: *use the full SVD of A and* the cyclic property of trace *(you can prove this property by direct checking):*

$$\mathsf{trace}(ABC) = \mathsf{trace}(BCA) = \mathsf{trace}(CAB) \tag{38}$$

for all A, B , C such that their product is defined and is a square matrix.

- The *nuclear norm*

$$\|A\|_* = \sum_{i=1}^{d} \sigma_j \tag{39}$$

is the Ky-Fan l^1 norm.

4.3. **Eckart-Young-Mirsky theorem.** Ref.: A. Dax "From Eigenvalues to Singular Values: A Review".

The Eckart-Young-Mirsky theorem says that the truncated SVD of A is the best rank k approximation to A in the 2-norm and in the Frobenius norm. It also holds for any Ky-Fan norm, so, for the nuclear norm as well.

Theorem 9. (Eckart-Young-Mirsky) *Let $A = U\Sigma V^\top$ be the SVD of A. Then for any matrix M of rank $\le k$ we have*

$$\|A - M\|_2 \ge \left\|A - U_k\Sigma_k V_k^\top\right\|_2 = \sigma_{k+1}, \tag{40}$$

$$\|A - M\|_F \ge \left\|A - U_k\Sigma_k V_k^\top\right\|_F = \sqrt{\sum_{j=k+1}^{n} \sigma_j^2} \tag{41}$$

We have already proven this theorem for the 2-norm in the beginning of this chapter. The proof for the Frobenius norm makes use of the following lemma.

Lemma 1. *Let a matrix $A \in \mathbb{R}^{n\times d}$ be decomposed into a sum $A = B + C$. Let $\sigma_j(M)$ denote the jth singular value of the matrix M, $M = A, B, C$. We also set $\sigma_j(M) = 0$ for all integer $j > \mathsf{rank}(M)$. Then*

$$\sigma_{i+j-1}(A) \le \sigma_i(B) + \sigma_j(C) \quad \forall i, j \ge 1, \tag{42}$$

Proof. Let i and j be arbitrary integers $1 \le i, j \le d$. Then, by the Eckart-Young-Mirsky theorem for the 2-norm

$$\sigma_i(B) + \sigma_j(C) \equiv \|B - B_{i-1}\|_2 + \|C - C_{j-1}\|_2 = \sigma_1(B - B_{i-1}) + \sigma_1(C - C_{j-1}).$$

By the triangle inequality for the 2-norm we have:

$$\|B - B_{i-1}\|_2 + \|C - C_{j-1}\|_2 \ge \|B - B_{i-1} + C - C_{j-1}\|_2 \equiv \|A - B_{i-1} - C_{j-1}\|_2.$$

Now we observe that the rank of a sum of two matrices cannot exceed the sum of their ranks. Therefore,

$$\mathsf{rank}(B_{i-1} + C_{j-1}) \le i + j - 2.$$

Then, by the Eckart-Young-Mirsky theorem for the 2-norm we have:

$$\|A - B_{i-1} - C_{j-1}\|_2 \ge \|A - A_{i+j-2}\|_2 = \sigma_{i+j-1}(A)$$

as desired. □

Now we prove the Eckart-Young-Mirsky theorem for the Frobenius norm.

Proof. **Proof for the Frobenius norm.** Lemma 1 implies that for any matrix M for rank $\le k$ and for all $i = 1, 2, \dots$ we have:

$$\sigma_{k+i}(A) \le \sigma_i(A - M) + \sigma_{k+1}(M) \equiv \sigma_i(A - M), \tag{43}$$

as $\sigma_{k+1}(M) = 0$ as $k + 1$ exceeds the rank of M. By (37):

$$\|A - M\|_F^2 = \sum_{i=1}^{d} \sigma_i^2(A - M) \ge \sum_{i=1}^{d-k} \sigma_i^2(A - M).$$

Then we apply (43) and obtain

$$\|A - M\|_F^2 \ge \sum_{i=1}^{d-k} \sigma_i^2(A - M) \ge \sum_{i=1}^{d-k} \sigma_{k+i}^2(A) = \sum_{j=k+1}^{d} \sigma_j^2(A)$$

as desired. □

Exersise 12. *Prove Eckart-Young-Mirsky theorem for an arbitrary Ky-Fan norm.*

4.4. **Gradient descent for SPD quadratic functions.** To proceed to discuss methods for computing low-rank we need to get familiar with a fundamental optimization algorithm called the *gradient descent*. The task of optimization is to find the minimum of a given function. The function to be optimized is called the *objective function*.

We consider the case where the objective function $\phi : \mathbb{R}^n \to \mathbb{R}$ is quadratic with an SPD matrix A:

$$\phi(x) := \frac{1}{2} x^\top A x + b^\top x + c. \tag{44}$$

Its gradient is given by

$$\nabla \phi(x) = Ax + b. \tag{45}$$

The point $x^* = -A^{-1}b$, the zero of $\nabla\phi$, is the minimizer of $\phi(x)$. Indeed, any $y \in \mathbb{R}^n$ can be written as $y = x^* + e$. Then

$$\begin{aligned}\phi(y) &= \phi(x^* + e)\\ &= \frac{1}{2}(x^* + e)^\top A(x^* + e) + b^\top(x^* + e) + c\\ &= \phi(x^*) + e^\top Ax^* + b^\top e + \frac{1}{2}e^\top Ae\\ &= \phi(x^*) - e^\top AA^{-1}b + b^\top e + \frac{1}{2}e^\top Ae\\ &= \phi(x^*) + \frac{1}{2}e^\top Ae \geq \phi(x^*).\end{aligned}$$

The equality takes place if and only if $e = 0$ as A is SPD.

The gradient descent iteration is

$$x_{k+1} = x_k - \alpha\nabla\phi(x_k). \tag{46}$$

The parameter α is called *stepsize* in classical optimization textbooks or *learning rate* in machine learning contexts. In the case of the objective function (44) it becomes

$$x_{k+1} = x_k - \alpha(Ax_k + b) = (I - \alpha A)x_k - b. \tag{47}$$

Let us estimate the error $e_k = x_k - x^*$ at iteration k. Note that for the solution $x^* = -A^{-1}b$ we have:

$$x^* = x^* - \alpha(A^* + b). \tag{48}$$

Subtracting (48) from (47) we obtain

$$e_{k+1} = (I - \alpha A)e_k. \tag{49}$$

Therefore, we can predict error at iteration k given the initial error e_0:

$$e_k = (I - \alpha A)^k e_0. \tag{50}$$

The necessary and sufficient condition for $\|e_k\| \to 0$ for any initial error e_0 (i.e., for any initial guess x_0) is that $\|I - \alpha A\| < 1$. The norm considered in this section is the 2-norm unless noted otherwise. Recall that $\|I - \alpha A\| = |\lambda_{\max}(I - \alpha A)|$ since $I - \alpha A)$ is symmetric. The eigenvalues of $I - \alpha A$ are $1 - \alpha\lambda_j(A)$, $1 \leq j \leq n$. Let us order the eigenvalues of A in decreasing order:

$$\lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A) > 0$$

Then

$$|\lambda_{\max}(I - \alpha A)| = \max\{|1 - \alpha\lambda_n(A)|, |1 - \alpha\lambda_1(A)|\}.$$

Note that

$$1 > 1 - \alpha\lambda_n(A) > 1 - \alpha\lambda_1(A)$$

Hence the only way $|\lambda_{\max}(I - \alpha A)|$ can reach 1 is when

$$1 - \alpha\lambda_1(A) \leq -1.$$

Hence, in order for the iteration to converge, α must be such that

$$1 - \alpha\lambda_1(A) > -1, \quad \text{i.e.} \quad \alpha < \frac{2}{\lambda_1(A)}. \tag{51}$$

The optimal convergence rate is achieved when $|\lambda_{\max}(I - \alpha A)|$ is minimal which happens at λ^* such that

$$1 - \alpha^*\lambda_n(A) = \alpha^*\lambda_1(A) - 1, \quad \text{i.e.} \quad \alpha^* = \frac{2}{\lambda_n(A) + \lambda_1(A)}. \tag{52}$$

At α^*, the norm of $I - \alpha^* A$ is

$$\|I - \alpha^* A\| = 1 - \frac{2\lambda_1(A)}{\lambda_n(A) + \lambda_1(A)} = 1 - \frac{2}{1 + \kappa(A)}, \tag{53}$$

where $\kappa(A) = \lambda_1/\lambda_n \equiv \sigma_1/\sigma_n$ is the condition number of A. Equation (53) shows that if A is ill-conditioned, then the norm of the error at every iteration might reduce by a factor very close to 1.

5. Nonnegative matrix factorization (NMF)

Reference: D. Bindel's Lecture 7 "NMF". A common problem with low-rank factorizations is that they are hard-to-interpret. In this section, we switch to interpretable matrix factorizations.

Let A be an $n \times d$ matrix with nonnegative entries. We seek matrices $W \in \mathbb{R}_+^{n\times k}$ and $H \in \mathbb{R}_+^{k\times d}$ where the subscripts $+$ means that their entries must be nonnegative, such that

$$A \approx WH. \tag{54}$$

5.1. Projected gradient descent.

Perhaps the simplest method to compute an NMF is using *projected gradient descent* (PGD). The projection used here is a simple nonnegativity constraint:

$$\mathcal{P}(\mathbf{x}) = [\mathbf{x}]_+, \quad \text{elementwise maximum of } \mathbf{x} \text{ and } 0.$$

Let ϕ be the objective function. The iteration is defined by

$$\mathbf{x}_{k+1} = \mathcal{P}(\mathbf{x}_k - \alpha_k \nabla\phi(\mathbf{x}_k)). \tag{55}$$

Its convergence properties are similar to those of the unprojected version. A convergence for convex functions and sufficiently small stepsizes can be proven. Ill-conditioning may make the convergence slow.

To work out the PGD iteration, it is handy to introduce the Frobenius inner product

$$\langle X, Y\rangle_F := \sum_{i,j} x_{ij}y_{ij} = \mathsf{trace}(X^\top Y) = \mathsf{trace}(Y^\top X). \tag{56}$$

For the problem (54), the objective function is

$$\phi(W, H) = \frac{1}{2}\|A - WH\|_F^2 = \frac{1}{2}\langle A - WH, A - WH\rangle_F. \tag{57}$$

Furthermore, to reduce the amount of writing, it is useful to use the notation $\delta\phi$, δW and δH for the variations of ϕ, W, and H, respectively. This is an analog of the differential of

a function of several variables. Regular differentiation rules apply. Let $R := A - WH$. We have:

$$\begin{aligned}\delta\phi &= \frac{1}{2}\delta\langle R, R\rangle_F = \langle \delta R, R\rangle_F \\ &= -\langle (\delta W)H, R\rangle_F - \langle W(\delta H), R\rangle_F. \end{aligned} \tag{58}$$

In the calculation below, we will use the cyclic property of the trace (38) to isolate the variations of W and H in the Frobenius inner products in (58):

$$\langle (\delta W)H, R\rangle_F = \mathsf{trace}\left(H^\top (\delta W)^\top R\right) = \mathsf{trace}\left((\delta W)^\top R H^\top\right) = \langle (\delta W), RH^\top\rangle_F, \tag{59}$$

$$\langle W(\delta H), R\rangle_F = \mathsf{trace}\left((\delta H)^\top W^\top R\right) = \langle (\delta H), W^\top R\rangle_F. \tag{60}$$

These equations mean that

$$\frac{\partial \phi}{\partial W_{ij}} = -\left(RH^\top\right)_{ij}, \quad \frac{\partial \phi}{\partial H_{ij}} = -\left(W^\top R\right)_{ij}, \quad 1 \le i \le n,\ 1 \le j \le d. \tag{61}$$

Therefore, the PGD iteration for minimizing (57) among $W \in \mathbb{R}_+^{n\times k}$ and $H \in \mathbb{R}_+^{k\times d}$ is:

$$W_{new} = \left[W + \alpha R H^\top\right]_+, \quad H_{new} = \left[H + \alpha W^\top R\right]_+. \tag{62}$$

5.2. **Multiplicative update scheme by Lee and Seung.** One of the earliest and most popular algorithms for NMF is the multiplicative update by D. D. Lee and H. S. Seung (2001) [1]. A derivation of their iteration can also be found here. In this algorithm, the entries of the matrices W and H are all updated with individually selected stepsizes. Let S_W be the matrix of stepsizes for the entries of W, and S_H be the same for H. The iteration is the projected gradient descend modified accordingly:

$$W_{new} = \left[W + S_W \odot R H^\top\right]_+, \quad H_{new} = \left[H + S_H \odot W^\top R\right]_+. \tag{63}$$

The projection in (63) zeroes out negative values. They may appear due to the subtractions hidden in $R = A - WH$ provided that all current entries in all matrices involved are nonnegative. The trick proposed by Lee and Seung allows us to avoid the need for the projection: the stepsizes are chosen so that the subtraction is eliminated! Let us rewrite (63) decoding R and removing the projection:

$$W_{new} = W + S_W \odot \left[AH^\top - WHH^\top\right], \quad H_{new} = H + S_H \odot \left[W^\top A - W^\top W H\right]. \tag{64}$$

The Lee and Seung stepsizes are

$$S_W = W \oslash \left[WHH^\top\right], \quad S_H = H \oslash \left[W^\top W H\right], \tag{65}$$

where $\oslash$ denotes entrywise division. It is easy to see that with this choice of stepsizes, the subtractions in (64) are completely eliminated as the subtracted term is canceled with the W and H, respectively. Therefore, the Lee-Seung iteration is:

$$W_{new} = \left[W \odot AH^\top\right] \oslash \left[WHH^\top\right], \quad H_{new} = \left[H \odot W^\top A\right] \oslash \left[W^\top W H\right]. \tag{66}$$

Monotone convergence of this algorithm is proven in [1]. A shortcoming of this algorithm is that it takes very conservative stepsizes, and it may take very many steps to achieve the desired convergence.

5.3. **Coordinate descent (CD).** Coordinate descent embraces the class of methods where the update directions are chosen along particular coordinates or their blocks.

5.3.1. *One entry at-a-time.* The simplest version of CD updates one entry of W or H at-a-time. Let s be step length and $\mathbf{e}_i$ be a column vector with entry 1 at position i and the rest of its entries being zeros. To determine s for updating entry (i,j) of W, we solve the following constrained least squares problem:

$$\frac{1}{2}\left\|A-(W+s\mathbf{e}_i\mathbf{e}_j^\top)H\right\|_F^2=\frac{1}{2}\left\|R-s\mathbf{e}_i\mathbf{e}_j^\top H\right\|_F^2$$

$$=\frac{1}{2}\|R\|_F^2-s\langle(\mathbf{e}_i\mathbf{e}_j^\top),RH^\top\rangle_F+\frac{s^2}{2}\|\mathbf{e}_i\mathbf{e}_j^\top H\|_F^2 \tag{67}$$

$$\text{subject to}\quad s\geq -w_{ij}. \tag{68}$$

Here, we used the cyclic property of the trace (38) and the rule:

$$\|A+B\|_F^2=\|A\|_F^2+\|B\|_F^2+2\langle A,B\rangle_F. \tag{69}$$

Exersise 13. *Prove* (69).

The matrix $\mathbf{e}_i\mathbf{e}_j^\top$ has only one nonzero entry at the position (i,j) equal to one, and the matrix $\mathbf{e}_i\mathbf{e}_j^\top H$ has a single nonzero row i equal to the row j of H. Therefore,

$$\langle(\mathbf{e}_i\mathbf{e}_j^\top),RH^\top\rangle_F=(RH^\top)_{ij},\quad \|\mathbf{e}_i\mathbf{e}_j^\top H\|_F^2=(HH^\top)_{jj}.$$

Hence s minimizing the quadratic function

$$\frac{s^2}{2}(HH^\top)_{jj}-s(RH^\top)_{ij}+\frac{1}{2}\|R\|_F^2\quad\text{is}\quad s=\frac{(RH^\top)_{ij}}{(HH^\top)_{jj}}.$$

Applying the constraint (68) we get the stepsize for the entry (i,j):

$$s=\max\left\{-w_{ij},\frac{(RH^\top)_{ij}}{(HH^\top)_{jj}}\right\}. \tag{70}$$

This leads to the update for w_{ij} and for row i of R:

$$w_{ij}=w_{ij}+s,\quad R_{i,:}=R_{i,:}-sH_{j,:}. \tag{71}$$

To update entry (i,j) of H, we are solving

$$\frac{1}{2}\left\|A-W(H+s\mathbf{e}_i\mathbf{e}_j^\top)\right\|_F^2=\frac{1}{2}\left\|R-sW\mathbf{e}_i\mathbf{e}_j^\top\right\|_F^2$$

$$=\frac{1}{2}\|R\|_F^2-s\langle(\mathbf{e}_i\mathbf{e}_j^\top),W^\top R\rangle_F+\frac{s^2}{2}\|W\mathbf{e}_i\mathbf{e}_j^\top\|_F^2 \tag{72}$$

$$\text{subject to}\quad s\geq -h_{ij}. \tag{73}$$

The stepsize and the update is obtained by a similar calculation:

$$s = \max\left\{-h_{ij}, \frac{(W^\top R)_{ij}}{(W^\top W)_{ii}}\right\}, \quad h_{ij} = h_{ij} + s, \quad R_{:,j} = R_{:,j} - sW_{:,i}. \tag{74}$$

5.3.2. *Hierarchical alternating least squares (HALS) or rank-one residual iteration (RRI).* The formulas developed in Section 5.3.1 are readily adapted for updating one column of W at-a-time and one row of H at-a-time. The corresponding constrained least squares problems

$$\frac{1}{2}\|R - uH_{j,:}\|_F^2 \to \min \quad \text{subject to} \quad u \geq -W_{:,j}, \tag{75}$$

$$\frac{1}{2}\|R - W_{:,i}v\|_F^2 \to \min \quad \text{subject to} \quad v \geq -H_{i,:}. \tag{76}$$

These problems are equivalent to

$$\frac{1}{2}\|R - u_iH_{j,:}\|_F^2 \to \min \quad \text{subject to} \quad u_i \geq -w_{ij}, \quad 1 \leq i \leq n, \tag{77}$$

$$\frac{1}{2}\|R - W_{:,i}v_j\|_F^2 \to \min \quad \text{subject to} \quad v_j \geq -h_{ij}, \quad 1 \leq j \leq d. \tag{78}$$

Therefore, the formulas for stepsizes (70) and (74) are suitable for computing u and v. The update formulas for column j of W and the matrix R, and then row i or H and R are:

$$W_{:,j} = W_{:,j} + u, \quad R = R - uH_{j,:}; \quad H_{i,:} = H_{i,:} + v, \quad R = R - W_{:,i}v.$$

5.3.3. *Alternating non-negative least squares (ANLS).* The ANLS updates all entries of W then all entries of H by solving the following constrained convex optimization problems:

$$\phi_1(W) = \frac{1}{2}\|A - WH\|_F^2 \to \min \quad \text{subject to} \quad W \geq 0, \tag{79}$$

$$\phi_2(H) = \frac{1}{2}\|A - WH\|_F^2 \to \min \quad \text{subject to} \quad H \geq 0. \tag{80}$$

Unfortunately, contrary to HALS, these problems cannot be solved in simple closed form. One approach is to solve them using some modern versions of the active set method that we will study in the chapter on optimization.

6. Collaborative filtering and matrix completion

Reference: D. Bindel's Lecture 8 "Matrix Completion". Imagine a spreadsheet with columns corresponding to movies and rows corresponding to users. Each user has watched some subset of movies. The problem posed by the Netflix company is to make intelligent guesses based on the available data how much each user would like the movies that she/he hasn't watched yet and make appropriate recommendations. Similar problems are also important for other online sellers. When you are shopping for some product, you often see that the website recommends you to look at some other products by saying something like: "The customers who looked at this products also looked at these products". I see this often and find that the system predicts my tastes and my needs quite well.

In this section, we will explore some methods for making such intelligent predictions. Let A be an incomplete $n \times d$ matrix in which only a_{ij} for

$$(i,j) \in \Omega \subset \{(l_1, l_2) \mid 1 \leq l_1 \leq n,\ 1 \leq l_2 \leq d\}$$

are known. We will denote by $P_\Omega(A)$ the projection of A onto Ω:

$$P_\Omega(A) = \begin{cases} a_{ij}, & (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases}$$

The idea is to pick a model M for some parametric family to minimize some loss function. For now, we pick the squared errors loss:

$$\phi(M) = \frac{1}{2}\|P_\Omega(A - M)\|_F^2 = \frac{1}{2}\sum_{(i,j)\in\Omega}(a_{ij} - m_{ij})^2. \tag{81}$$

Let us consider some models for M to build up some intuition.

6.1. **Two simple trial models.**

6.1.1. *Baseline model.* Let $M = \mu 1_{n\times d}$ where μ is to be determined. Plugging M into (81) and finding the minimizer of the resulting 1D quadratic function, we find

$$\mu = \frac{1}{|\Omega|}\sum_{(i,j)\in\Omega} a_{ij}. \tag{82}$$

While this model is exactly solvable, it is useless for making predictions. It gives the same rating for all movies and all users.

6.1.2. *Baseline plus uniform adjustments for each user and each movie.* This model is given by

$$M = \mu 1_{n\times d} + \mathbf{b}1_{1\times d} + 1_{n\times 1}\mathbf{c}^\top, \quad \mathbf{b} \in \mathbb{R}^n, \quad \mathbf{c} \in \mathbb{R}^d. \tag{83}$$

The first term in (106) gives some uniform base rating to all movies for each user. The second term uniformly adjusts ratings for all movies, but these adjustments are chosen individually for each user. The third term adjusts movie ratings uniformly for all users, but these adjustments are individual for each movie.

The solution to this problem is the solution to the linear system obtained by taking the gradient of

$$\phi(\mu, \mathbf{b}, \mathbf{c}) = \frac{1}{2}\sum_{(i,j)\in\Omega}(a_{ij} - \mu - b_i - c_j)^2 : \tag{84}$$

$$\frac{\partial \phi}{\partial \mu} = -\sum_{(i,j)\in\Omega} (a_{ij} - \mu - b_i - c_j) = 0, \tag{85}$$

$$\frac{\partial \phi}{\partial b_i} = -\sum_{j\in\Omega_i} (a_{ij} - \mu - b_i - c_j) = 0, \quad \Omega_i := \{j \mid (i,j)\in\Omega\}, \tag{86}$$

$$\frac{\partial \phi}{\partial c_j} = -\sum_{i\in\Omega_j} (a_{ij} - \mu - b_i - c_j) = 0, \quad \Omega_j := \{i \mid (i,j)\in\Omega\}. \tag{87}$$

- Note that the set of solutions to (85)–(87) is at least two-dimensional:

if $(\mu, \mathbf{b}, \mathbf{c})$ is a solution, then so is $(\mu+\alpha, \mathbf{b}+\beta 1_{n\times 1}, \mathbf{c}-(\alpha+\beta)1_{d\times 1}), \quad \forall \alpha, \beta \in \mathbb{R}$.

  However, this is not a problem since each of these solutions gives the same matrix M, hence the same recommendations. One way to get rid of this nonuniqueness is to impose the condition that $\mathbf{b}$ and $\mathbf{c}$ both sum up to zero. Them μ is given by (82).
- Also note that the resulting linear system is large: $|\Omega| \times (1+n+d)$. This only will be a problem if we try to use factorization-based direct methods. But certain iterative methods will work just fine.
- Finally, and most importantly, this model is still not useful as it predicts the same relative rankings for each user.

6.2. **Low-rank factorization.** A useful model is the low-rank model:

$$M = XY^\top, \quad X \in \mathbb{R}^{n\times k}, \quad Y \in \mathbb{R}^{d\times k}. \tag{88}$$

As the second model considered in the previous section, the factorization $M = XY^\top$ is not unique. To get rid of this nonuniqueness and to penalize crazy choices of X and Y, we regularize the problem by introducing a penalty for large Frobenius norms of X and Y:

$$F(X,Y) = \frac{1}{2}\|P_\Omega(A - XY^\top)\|_F^2 + \frac{\lambda}{2}\left(\|X\|_F^2 + \|Y\|_F^2\right). \tag{89}$$

Let $R = P_\Omega(A - XY^\top)$. Then $F(X,Y) = \frac{1}{2}\langle R, R\rangle_F + \frac{\lambda}{2}\left(\|X\|_F^2 + \|Y\|_F^2\right)$.

To take the variation of F, we first calculate the variation of its first term:

$$\begin{aligned}
\langle \delta R, R\rangle_F &= \langle (-\delta X)Y^\top - X(\delta Y)^\top, R\rangle_F \\
&= -\langle (\delta X)Y^\top, R\rangle_F - \langle X(\delta Y)^\top, R\rangle_F \\
&= -\mathsf{trace}(Y(\delta X)^\top R) - \mathsf{trace}((\delta Y)X^\top R) \\
&= -\mathsf{trace}((\delta X)^\top RY) - \mathsf{trace}((R^\top X)^\top(\delta Y)) \\
&= -\langle \delta X, RY\rangle_F - \langle \delta Y, R^\top X\rangle_F. \qquad (90)
\end{aligned}$$

Note that in the calculation above we did not need to project the first argument of the Frobenius inner product because the terms of $(-\delta X)Y^\top - X(\delta Y)^\top)$ corresponding to unknown entries of A will be zeroes out due to the second term R.

The variation of the second term of F is

$$\langle \delta X, \lambda X \rangle_F + \langle \delta Y, \lambda Y \rangle_F.$$

Hence

$$\delta F = \langle \delta X, \lambda X - RY \rangle_F + \langle \delta Y, \lambda Y - R^\top X \rangle_F. \tag{91}$$

First, consider the case where all entries of A are available. Let us show that in this case, the solution to $F \to \min$ would be

$$X = U_k \sqrt{s_\lambda(\Sigma_k)}, \quad Y = V_k \sqrt{s_\lambda(\Sigma_k)}, \quad \text{where} \quad s_\lambda(\sigma) = [\sigma - \lambda]_+, \tag{92}$$

and $U_k \Sigma_k V_k^\top$ is the truncated SVD of A. Note that if $\lambda = 0$ then $XY^\top = U_k \Sigma_k V_k^\top$ would be the optimal rank k approximation of k as we know from the Eckart-Young-Mirsky theorem.

We will show that, if X and Y are given by (92), then the variation of F is zero, i.e.

$$RY = \lambda X, \quad R^\top X = \lambda Y. \tag{93}$$

Indeed,

$$\begin{aligned}
RY &= U\Sigma V^\top V_k \sqrt{s_\lambda(\Sigma_k)} - U_k s_\lambda(\Sigma_k) V_k^\top V_k \sqrt{s_\lambda(\Sigma_k)} \\
&= U_k \Sigma_k \sqrt{s_\lambda(\Sigma_k)} - U_k s_\lambda(\Sigma_k) \sqrt{s_\lambda(\Sigma_k)} \\
&= U_k \left[\Sigma_k - s_\lambda(\Sigma_k)\right] \sqrt{s_\lambda(\Sigma_k)} = \lambda U_k \sqrt{s_\lambda(\Sigma_k)} = \lambda X, \\
R^\top X &= V\Sigma U^\top U_k \sqrt{s_\lambda(\Sigma_k)} - V_k s_\lambda(\Sigma_k) U_k^\top U_k \sqrt{s_\lambda(\Sigma_k)} \\
&= V_k \Sigma_k \sqrt{s_\lambda(\Sigma_k)} - V_k s_\lambda(\Sigma_k) \sqrt{s_\lambda(\Sigma_k)} \\
&= V_k \left[\Sigma_k - s_\lambda(\Sigma_k)\right] \sqrt{s_\lambda(\Sigma_k)} = \lambda V_k \sqrt{s_\lambda(\Sigma_k)} = \lambda Y.
\end{aligned}$$

Here we have used the fact that for $1 \le j \le k$,

$$\sigma_j - [\sigma_j - \lambda]_+ = \begin{cases} \lambda, & \lambda \le \sigma_j, \\ \sigma_j, & \lambda > \sigma_j, \end{cases}$$

and, if $\lambda > \sigma_j$, then $s_\lambda(\sigma_j) = 0$.

The situation is trickier if only partial data are available. In this case, we need to minimize F numerically. For example, we can use stochastic gradient descent. Another option is to do the alternating iteration:

$$X^{k+1} = \arg\min_X F(X, Y^k), \tag{94}$$

$$Y^{k+1} = \arg\min_Y F(X^{k+1}, Y). \tag{95}$$

Each of these steps can be further decomposed into a collection of small linear least squares problems. For example, at each substep of (94), we solve the linear least squares problem to compute the row i of X:

$$\mathbf{x}_i^\top = \arg\min_{\mathbf{x}} \frac{1}{2} \left\| \mathbf{x}^\top Y_{\Omega_i}^\top - a_{\Omega_i} \right\|_2^2 + \frac{\lambda}{2} \|\mathbf{x}\|^2, \tag{96}$$

where $\Omega_i := \{j \mid (i,j) \in \Omega\}$, $Y^\top_{\Omega_i}$ is the set of columns of $Y^\top$ with indices in Ω_i, and a_{Ω_i} is the set of known entries of A in its row i.

6.3. **Penalizing nuclear norm.** Reference: Bindel's lecture 8, Section 3 "Nuclear norm trick".

An alternative approach to optimizing factors of the model matrix M is to optimize the matrix M itself. Note that rank is not a continuous function of matrix entries, hence, imposing rank constraints is not a promising approach. Instead, we are going to penalize the nuclear norm of M, i.e.,

$$\phi(M) = \frac{1}{2}\|P_\Omega(A) - P_\Omega(M)\|_F^2 + \lambda\|M\|_*, \quad \|M\|_* = \sum_i \sigma_i(M). \tag{97}$$

The nuclear norm constraint is low-rank promoting for the same reason as the lasso regularizer is sparsity promoting. Below I offer an explanation for it.

Observe that $\phi(M)$ can be viewed as a Lagrangian function minus $t\lambda$ for the following constrained optimization problem

$$f(M) := \frac{1}{2}\|P_\Omega(A) - P_\Omega(M)\|_F^2 \ \to \ \min \text{ subject to } t - \|M\|_* \geq 0 \tag{98}$$

for some positive constant t. If there exists a matrix M with $\|M\|_* < t$ such that $P_\Omega(A - M) = 0$, then the KKT optimality conditions require that $\lambda = 0$ as $\lambda(t - \|M\|_*) = 0$ while $t - \|M\|_* > 0$. Since we set $\lambda > 0$ in (97), this is not the case. This means that $\|M\|_* = t$.

To get a sense of what is the set $\|M\|_* = t$, let us consider a very simple example that we can visualize. Consider the set of 2×2 matrices

$$M(w,x,y,z) := \begin{bmatrix} w & x \\ y & z \end{bmatrix}.$$

Let us find a subset S_1 of $(w,x,y,z) \in \mathbb{R}^4$ such that the nuclear norm of the matrix is 1, i.e.

$$S_1 := \left\{(w,x,y,z) \in \mathbb{R}^4 \mid \sigma_1(M(w,x,y,z)) + \sigma_2(M(w,x,y,z)) = 1\right\}. \tag{99}$$

The set S_1 is a 3D surface in a 4D space. We are particularly interested in two aspects of S_1:

- Does the surface have singularities (2D edges)? This is because the level sets of $f(M)$ in (98) are smooth ellipsoids, and the minimal ellipsoid having a nonempty intersection with the surface $\|M\|_*$ tends to have this intersection on a singular edge.
- Do these singularities of the surface correspond to a low rank of $M(w,x,y,z)$?

We cannot visualize a 3D surface in 4D, but we can visualize a family of its 2D slices each of which corresponds to a fixed value of w. Three of these slices are displayed in Fig. 2. We color the slices according to the value of the determinant of $M(w,x,y,z)$. If $\det(M(w,x,y,z)) = 0$ then $\mathsf{rank}(M(w,x,y,z)) < 2$. We see that each slice has singular a singular edge, and the surface color near the edge is green which corresponds to $\det(M(w,x,y,z)) = 0$. The full set of slices is shown in the Youtube video.

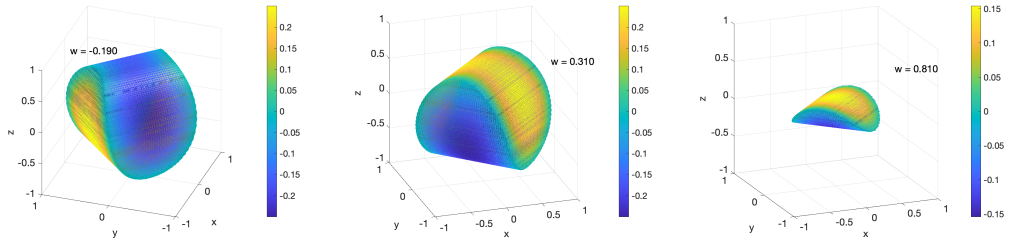

FIGURE 2. Slices of the set S_1 (see (99)) corresponding to $w = -0.19$ (left), $w = 0.31$ (middle), and $w = 0.81$ (right). The coloring of the surfaces corresponds to the values of $\det(M(w, x, y, z))$. Note that the edge of these surfaces is green which corresponds to $\det(M(w, x, y, z)) = 0$.

Now let us discuss methods for solving the minimization problem (97). It is helpful to see what is the solution to it in the case if all entries of A are known, the so-called *proximal problem*. Then (97) becomes

$$\phi(M) = \frac{1}{2}\|A - M\|_F^2 + \lambda\|M\|_*. \tag{100}$$

Its minimizer is given by

$$S_\lambda(A) := U s_\lambda(\Sigma) V^\top, \tag{101}$$

where $A = U\Sigma V^\top$ is the SVD of A and $s_\lambda(\sigma_j) = \max\{\sigma_j - \lambda, 0\}$ as before. Note that if there are exactly k singular values of A greater than λ then (101) is also the minimizer for the problem considered in Section (6.2):

$$F(X, Y) = \frac{1}{2}\left\|A - XY^\top\right\|_F^2 + \frac{\lambda}{2}\left(\|X\|_F^2 + \|Y\|_F^2\right) \to \min, \quad X \in \mathbb{R}^{n\times k},\ Y \in \mathbb{R}^{d\times k}. \tag{102}$$

A similar result holds when only part of the data matrix A is available: the nuclear norm regularization and the optimization of the factored form with Frobenius norm regularization on the factors yield the same model predictions when the factor size k in the latter problem is at least as large as the rank observed in the nuclear norm problem.

The SVD solution (101) to the proximal problem suggests the following iteration:

$$M^{j+1} = S_\lambda\left(M^j + P_\Omega(A - M^j)\right). \tag{103}$$

Since there are only a few singular values greater than λ at each step, the necessary components of the SVD at each step can be computed very efficiently using a Lanczos-type algorithm (see e.g. Trefenthen and Bau "Numerical Linear Algebra").

7. CUR MATRIX DECOMPOSITION

Please read the PNAS article by M. Mahoney and P. Drineas of 2009 [2]. The CUR algorithm in it is the one I would like you to implement. The preceding article by the same authors plus S. Muthukrishnan [3] contains detailed proofs and more complicated

algorithms with better worst-case scenario guarantees. Background material on leverage scores can be found e.g. here. In addition, here is a nice lecture on CUR by Jeff M. Philips, University of Utah.

8. Conjugate gradient Methods

Read Section 5.1 in J. Nocedal and S. Wright "Numerical Optimization", Second Edition, Springer, 2006 (available online)

9. Direct methods for solving linear systems with sparse and structured matrices

Ref.: Ten lectures by G. Martinsson (2014). Read Lecture 1 for the general introduction to direct methods and Lecture 6 for the *Nested Dissection* algorithm (George, 1973).

9.1. **A model problem.** We start with a simple model problem, the Poisson equation in 2D in a unit square $\Omega = \{0 \le x \le 1, 0 \le y \le 1\}$ with Dirichlet boundary conditions:

$$u_{xx} + u_{yy} - f(x,y) = 0, \quad (x,y) \in \Omega, \tag{104}$$

$$u = 0, \quad (x,y) \in \partial\Omega. \tag{105}$$

Let us introduce a mesh with step $h = 1/J$ and approximate Eq. (104) with the central difference scheme with a 5-point stencil (Fig. 3):

$$\frac{1}{h^2}\left(U_N + U_S + U_W + U_E - 4U_P\right) = f_P. \tag{106}$$

Each mesh point is naturally indexed with two indices (i,j) where $i,j \in \{0,1,\dots,J\}$,

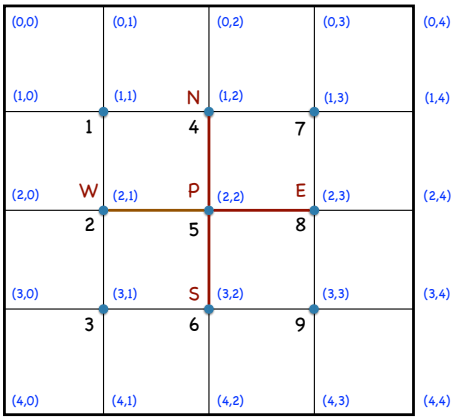

Figure 3. An illustration for the 5-point stencil and indexing of mesh points.

i.e., $J+1$ points in each direction in total. I prefer to index them in the same order as the matrix entries are indexed (Fig. 3). The function u is known to be 0 at all boundary

points of the mesh: $u_{0j} = u_{i0} = u_{Jj} = u_{iJ} = 0$, $i, j \in \{0, 1, \ldots, J\}$. Therefore, we need to find u_{ij}, $1 \le i, j \le J-1$. To do it, we set up a linear system for u_{ij} of the form $Au = f$. The matrix A will be $(J-1)^2 \times (J-1)^2$. We start with casting u_{ij}, $1 \le i, j \le J-1$ into a vector column-wise as shown in Fig. 3. This indexing is consistent with Matlab's operator (:) and its inverse operator `reshape` illustrated in the following example.

```
>> a = [1 4 7; 2 5 8; 3 6 9]

a =

     1     4     7
     2     5     8
     3     6     9

>> a1 = a(:)

a1 =

     1
     2
     3
     4
     5
     6
     7
     8
     9

>> a2 = reshape(a1,3,3)

a2 =

     1     4     7
     2     5     8
     3     6     9
```

For the example in Fig. 3 we obtain the following linear system.

$Au = \frac{1}{h^2}$

	1	2	3	4	5	6	7	8	9		
1	-4	1		1						u_1	f_1
2	1	-4	1		1					u_2	f_2
3		1	-4			1				u_3	f_3
4	1			-4	1		1			u_4	f_4
5		1		1	-4	1		1		u_5	f_5
6			1		1	-4			1	u_6	f_6
7				1			-4	1		u_7	f_7
8					1		1	-4	1	u_8	f_8
9						1		1	-4	u_9	f_9

From this example, it is easy to catch the block structure of the matrix A, where each block is $(J-1)\times(J-1)$:

$$A = \frac{1}{h^2}\begin{bmatrix} T & I & & \\ I & T & I & \\ & \ddots & \ddots & \ddots \\ & & I & T \end{bmatrix}, \quad \text{where} \quad T = \begin{bmatrix} -4 & 1 & & \\ 1 & -4 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -4 \end{bmatrix}, \tag{107}$$

and I is the $(J-1)\times(J-1)$ identity matrix.

Matlab has two types of matrices: full, where all entries are kept in memory, and sparse, where only nonzero entries and their indices are kept in memory. Each row of A in Eq. (107) has at most 5 nonzero entries, hence it is worthwhile to set it up as sparse. Note that if you set it up as full, Matlab will be able to solve your system for at most $J = 150$ or so depending on your computer. It can handle much larger values of J if you set up A as sparse. Look how A is set up in the code below. The command `kron(A,B)` gives the Kronecker product of matrices A and B. The code below solves Eq. (104) with $f(x,y) = \sin(2\pi x)\sin(2\pi y)$ for $J = 2^k$, $k = 8$. The exact solution is $u_{exact} = -f(x,y)/(8\pi^2)$.

```
k = 8;
n = 2^(k + 2) + 1;
n2 = n - 2;
t = linspace(0,1,n);
[x,y] = meshgrid(t,t);
f = sin(2*pi*x).*sin(2*pi*y);
f1 = f(2 : n - 1,2 : n - 1);
f_aux = f1(:);
h = 1/(n - 1);
u = zeros(n);

% Set up the matrix A
I = speye(n2); % n2-by-n2 sparse identity matrix
e = ones(n2,1);
T = spdiags([e -4*e e],[-1:1],n2,n2); % set up a sparse matrix T
S = spdiags([e e],[-1 1],n2,n2); % set up a sparse matrix S
```

```
A = (kron(I,T) + kron(S,I))/h(k)^2; % kron is the Kronecker product

% Solve the linear system
u_aux = A\f_aux;
u(2:n-1,2:n-1) = reshape(u_aux,n2,n2);
```

Nested Dissection

References

[1] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Proc. Neural Information Processing Systems*, 2001.

[2] M. W. Mahoney and P. Drineas, "Cur matrix decompositions for improved data analysis," *PNAS*, vol. 106, no. 3, pp. 697–702, 2009.

[3] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Relative-error cur matrix decompositions," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 844–881, 2008.