STAT 309: MATHEMATICAL COMPUTATIONS I
FALL 2019
LECTURE 6

1. PROJECTIONS

- the solution $\mathbf{x}$ of the least-squares problem minimizes $\|A\mathbf{x}-\mathbf{b}\|_2$, and therefore is the vector that solves the system $A\mathbf{x} = \mathbf{b}$ as closely as possible
- we can use the SVD to show that $\mathbf{x}$ is the exact solution to a related system of equations
- write $\mathbf{b} = \mathbf{b}_0 + \mathbf{b}_1$, where
$$\mathbf{b}_1 = AA^\dagger \mathbf{b}, \quad \mathbf{b}_0 = (I - AA^\dagger)\mathbf{b}$$
- the matrix $AA^\dagger$ has the form
$$AA^\dagger = U\Sigma V^* V \Sigma^\dagger U^* = U\Sigma\Sigma^\dagger U^* = U \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} U^*$$
- it follows that $\mathbf{b}_1$ is a linear combination of $\mathbf{u}_1, \dots, \mathbf{u}_r$, the columns of U that form an orthogonal basis for the range of A
- from $\mathbf{x} = A^\dagger \mathbf{b}$ we obtain
$$A\mathbf{x} = AA^\dagger \mathbf{b} = P_1 \mathbf{b} = \mathbf{b}_1$$
where $P_1 = AA^\dagger \in \mathbb{C}^{m\times m}$
- therefore, the solution to the least squares problem, is also the exact solution to the system
$$A\mathbf{x} = P_1 \mathbf{b}$$
- it can be shown that the matrix P_1 is an *orthogonal projection*
- in general a matrix $P \in \mathbb{C}^{m\times m}$ is called a *projection* if $P^2 = P$ (this condition is also called idempotent in ring theory)
- a projection is called an orthogonal projection if it is also Hermitian, i.e. an orthogonal projection is a matrix $P \in \mathbb{C}^{m\times m}$ satisfying
  (i) $P = P^*$
  (ii) $P^2 = P$
- caveat: an orthogonal projection is in general *not* an orthogonal/unitary matrix (i.e., $P^* \neq P^{-1}$) in fact, projections are usually non-invertible
- example: $\left[\begin{smallmatrix} 1 & \alpha \\ 0 & 0 \end{smallmatrix}\right]$ is a projection for any $\alpha \in \mathbb{C}$, it is an orthogonal projection if and only if $\alpha = 0$
- if $P \in \mathbb{C}^{m\times m}$ is a projection and $\operatorname{im}(P) = W$, we say that P is a projection onto the subspace W
- if $P \in \mathbb{C}^{m\times m}$ is a projection matrix, then $I - P$ is also a projection
- furthermore if $\operatorname{im}(P) = W$ and $\operatorname{im}(I-P) = W'$, then
$$\mathbb{C}^m = W \oplus W'$$
- if P is an orthogonal projection and $\operatorname{im}(P) = W$, then $\operatorname{im}(I-P) = W^\perp$
- we sometimes write P_W if we know the subspace P that projects onto
- in particular, $P_1 = AA^\dagger$ is a projection onto the space spanned by the columns of A, i.e., $\operatorname{im}(A)$, so $P_1 = P_{\operatorname{im}(A)}$

Date: October 15, 2019, version 1.0. Comments, bug reports: `lekheng@galton.uchicago.edu`.

2. COMPUTING PROJECTIONS ONTO FUNDAMENTAL SUBSPACES

- we can write down the orthogonal projections onto all four fundamental subspaces in terms of the pseudoinverse

$$P_{\mathrm{im}(A)} = AA^\dagger, \quad P_{\ker(A^*)} = I - AA^\dagger, \quad P_{\mathrm{im}(A^*)} = A^\dagger A, \quad P_{\ker(A)} = I - A^\dagger A$$

- note that $P_{\mathrm{im}(A)}, P_{\ker(A^*)} \in \mathbb{C}^{m\times m}$ and $P_{\mathrm{im}(A^*)}, P_{\ker(A)} \in \mathbb{C}^{n\times n}$
- with the SVD, we can write down the projections in terms of unitary matrices

$$P_{\mathrm{im}(A)} = U \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} U^* = U_r U_r^*, \quad P_{\ker(A^*)} = U \begin{bmatrix} 0 & 0 \\ 0 & I_{m-r} \end{bmatrix} U^* = U_{m-r} U_{m-r}^*,$$
$$P_{\mathrm{im}(A^*)} = V \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} V^* = V_r V_r^*, \quad P_{\ker(A)} = V \begin{bmatrix} 0 & 0 \\ 0 & I_{n-r} \end{bmatrix} V^* = V_{n-r} V_{n-r}^*$$

  where $U = [U_r, U_{m-r}]$ and $V = [V_r, V_{n-r}]$
- we will often have to project vectors onto subspaces spanned by singular vectors, it is important to note that we will *not* actually compute the projection matrix and then multiply them to the vectors to achieve this
- we will see in Homework 2 how one can compute $P_W \mathbf{v}$ without forming P_W for a subspace W spanned by singular vectors
- in general, one *never* explicitly forms $P = AA^\dagger$ nor even $A^\dagger$ — doing so is a waste of computing time and gives inaccurate results

3. LEAST SQUARES WITH QUADRATIC CONSTRAINTS

- let $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and α be some given positive number
- we wish to solve the problem

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{b} - A\mathbf{x}\|_2 \\ \text{subject to} \quad & \|\mathbf{x}\|_2 \le \alpha \end{aligned} \tag{3.1}$$

- this problem is known as *least squares with quadratic constraints*
- arises in many situations:
  - ridge regression
  - Tychonov regularization
  - generalized cross-validation (GCV)
- note that if $\alpha \ge \|A^\dagger \mathbf{b}\|_2$, the unconstrained minimum norm solution $A^\dagger \mathbf{b}$ would already be a solution
- so for a non-trivial solution, we assume that $\alpha < \|A^\dagger \mathbf{b}\|_2$ and in which case the solution $\mathbf{x}$ to (3.1) must sit on the boundary of the ball of radius α, i.e., $\|\mathbf{x}\|_2 = \alpha$
- to solve this problem, we define the *Lagrangian*

$$L(\mathbf{x}, \mu) = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \mu(\|\mathbf{x}\|^2 - \alpha^2)$$

  where μ is called the *Lagrange multiplier*
- first-order condition for minimality: set derivative to zero

$$\mathbf{0} = \nabla_{\mathbf{x}} L(\mathbf{x}, \mu) = -2A^\mathsf{T}\mathbf{b} + 2A^\mathsf{T}A\mathbf{x} + 2\mu\mathbf{x}$$

- we obtain

$$(A^\mathsf{T}A + \mu I)\mathbf{x} = A^\mathsf{T}\mathbf{b} \tag{3.2}$$

- if we denote the eigenvalues of $A^\mathsf{T}A$ by

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \ge 0$$

- then eigenvalues of $A^\mathsf{T}A + \mu I$ are

$$\lambda_1 + \mu, \cdots, \lambda_n + \mu$$

- if $\mu \geq 0$, then $\kappa_2(A^\intercal A + \mu I) \leq \kappa_2(A^\intercal A)$, because
$$\frac{\lambda_1 + \mu}{\lambda_n + \mu} \leq \frac{\lambda_1}{\lambda_n}$$
- so $A^\intercal A + \mu I$ is better conditioned
- to solve (3.2), we see that we need to compute
$$\mathbf{x} = (A^\intercal A + \mu I)^{-1} A^\intercal \mathbf{b} \tag{3.3}$$
where
$$\mathbf{x}^\intercal \mathbf{x} = \mathbf{b}^\intercal A (A^\intercal A + \mu I)^{-2} A^\intercal \mathbf{b} = \alpha^2$$
- if $A = U\Sigma V^\intercal$ is the full SVD of A, we let $\mathbf{c} = U^\intercal \mathbf{b}$, then we have
$$\begin{aligned}
\alpha^2 &= \mathbf{b}^\intercal U\Sigma V^\intercal (V\Sigma^\intercal \Sigma V^\intercal + \mu I)^{-2} V\Sigma^\intercal U^\intercal \mathbf{b} \\
&= \mathbf{c}^\intercal \Sigma [(V\Sigma^\intercal \Sigma V^\intercal + \mu I)V]^{-1} [V^\intercal (V\Sigma^\intercal \Sigma V^\intercal + \mu I)]^{-1} \Sigma^\intercal \mathbf{c} \\
&= \mathbf{c}^\intercal \Sigma (V\Sigma^\intercal \Sigma + \mu V)^{-1} (\Sigma^\intercal \Sigma V^\intercal + \mu V^\intercal)^{-1} \Sigma^\intercal \mathbf{c} \\
&= \mathbf{c}^\intercal \Sigma [(\Sigma^\intercal \Sigma V^\intercal + \mu V^\intercal)(V\Sigma^\intercal \Sigma + \mu V)]^{-1} \Sigma^\intercal \mathbf{c} \\
&= \mathbf{c}^\intercal \Sigma (\Sigma^\intercal \Sigma + \mu I)^{-2} \Sigma^\intercal \mathbf{c} \\
&= \sum_{i=1}^{r} \frac{c_i^2 \sigma_i^2}{(\sigma_i^2 + \mu)^2} \\
&=: f(\mu)
\end{aligned}$$
where $\mathbf{c} = (c_1, \dots, c_m)^\intercal$
- the function $f(\mu)$ has poles at $-\sigma_i^2$ for $i = 1, \dots, r$
- furthermore, $\lim_{\mu \to \infty} f(\mu) = 0$
- algorithm for solving this problem, given A, $\mathbf{b}$, and α^2:
  - step 1: compute SVD of A to obtain $A = U\Sigma V^\intercal$
  - step 2: compute $\mathbf{c} = U^\intercal \mathbf{b}$
  - step 3: solve $f(\mu_*) = \alpha^2$ with Newton–Raphson method
  - step 4: use the SVD to compute
$$\mathbf{x} = (A^\intercal A + \mu I)^{-1} A^\intercal \mathbf{b} = V(\Sigma^\intercal \Sigma + \mu I)^{-1} \Sigma^\intercal U^\intercal \mathbf{b}$$
- don't use Newton–Raphson method on this equation directly; solving $1/f(\mu) = 1/\alpha^2$ is much better
- this is an example of an 'almost closed form' solution: we have an analytic expression for $\mathbf{x}$ that depends on just one unknown parameter μ_*, which is the root of a univariate nonlinear equation

4. Solving total least squares problems

- assume $A \in \mathbb{C}^{m \times n}$ has full column rank, i.e., $\operatorname{rank}(A) = n \leq m$
- in ordinary least squares problem, we solve
$$A\mathbf{x} = \mathbf{b} + \mathbf{r}, \quad \|\mathbf{r}\|_2 = \min$$
- in *total least squares* problem, we wish to solve
$$(A + E)\mathbf{x} = \mathbf{b} + \mathbf{r}, \quad \|E\|_F^2 + \lambda^2 \|\mathbf{r}\|_2^2 = \min$$
- note that if $\mathbf{b} \in \operatorname{im}(A)$, then the solution is given by setting $E = O$, $\mathbf{r} = \mathbf{0}$ and choosing $\mathbf{x}$ to be any solution of $A\mathbf{x} = \mathbf{b}$
- so assume $\mathbf{b} \notin \operatorname{im}(A)$ and therefore
$$\operatorname{rank}([A, \mathbf{b}]) = n + 1$$

- from $A\mathbf{x} - \mathbf{b} + E\mathbf{x} - \mathbf{r} = \mathbf{0}$ we obtain the system
$$\begin{bmatrix} A & \mathbf{b} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} + \begin{bmatrix} E & \mathbf{r} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} = \mathbf{0}$$
or
$$(C + F)\mathbf{z} = \mathbf{0} \tag{4.1}$$
- since
$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} \neq \mathbf{0} \tag{4.2}$$
we must have $\operatorname{nullity}(C + F) \geq 1$
- and since $\operatorname{rank}(C) = n + 1$, we must have
$$\operatorname{rank}(C + F) \leq n$$
- we need the matrix $C + F$ to have $\operatorname{rank}(C + F) \leq n$, and we want to minimize $\|F\|_F$
- to solve this problem, we compute the SVD of $C \in \mathbb{C}^{m \times (n+1)}$
$$C = \begin{bmatrix} A & \mathbf{b} \end{bmatrix} = U\Sigma V^* = U \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_n & \\ & & & \sigma_{n+1} \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} V^*$$
- note that $\sigma_{n+1} > 0$ since $\operatorname{rank}(C) = n + 1$
- we want F so that $\operatorname{rank}(C + F) \leq n$ so need to zero out σ_{n+1}, i.e., we want
$$C + F = U \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_n & \\ & & & 0 \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} V^* \tag{4.3}$$
or to be more precise, we want
$$\min_{\operatorname{rank}(C+F) \leq n} \|F\|_F = \min_{\operatorname{rank}(C+F) \leq n} \|C - (C + F)\|_F = \min_{\operatorname{rank}(X) \leq n} \|C - X\|_F$$
and Eckhart–Young theorem tells us that $X := C + F$ must take the form in (4.3)
- so pick
$$F = U \begin{bmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & -\sigma_{n+1} \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} V^*$$
and note that this F would produce the effect needed for (4.3)
- let $V = [\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}] \in \mathbb{C}^{(n+1)\times(n+1)}$ where $\mathbf{v}_i \in \mathbb{C}^{n+1}$ is the ith column of V note that $\mathbf{v}_i^* \mathbf{v}_{n+1} = 0$ for all $i = 1, \ldots, n$

- we have

$$(C+F)\mathbf{v}_{n+1} = U \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_n & \\ & & & 0 \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} V^* \mathbf{v}_{n+1} = U \begin{bmatrix} \sigma_1 \mathbf{v}_1^* \\ \vdots \\ \sigma_n \mathbf{v}_n^* \\ \mathbf{0}^\mathsf{T} \\ \mathbf{0}^\mathsf{T} \\ \vdots \\ \mathbf{0}^\mathsf{T} \end{bmatrix} \mathbf{v}_{n+1} = U \begin{bmatrix} \sigma_1 \mathbf{v}_1^* \mathbf{v}_{n+1} \\ \vdots \\ \sigma_n \mathbf{v}_n^* \mathbf{v}_{n+1} \\ \mathbf{0}^\mathsf{T} \mathbf{v}_{n+1} \\ \mathbf{0}^\mathsf{T} \mathbf{v}_{n+1} \\ \vdots \\ \mathbf{0}^\mathsf{T} \mathbf{v}_{n+1} \end{bmatrix} = U \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{bmatrix} = \mathbf{0}$$

- so the vector $\mathbf{v}_{n+1}$ ought to be a candidate for the solution $\mathbf{z}$ in (4.1) but there is one caveat — the last coordinate of $\mathbf{z}$ must be -1 by (4.2)
- how do we achieve that? we divide $\mathbf{v}_{n+1}$ by the negative of its last coordinate, so

$$\begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} = \mathbf{z} = -\frac{1}{v_{n+1,n+1}} \mathbf{v}_{n+1}$$

provided that $v_{n+1,n+1} \neq 0$
- this gives the solution

$$\mathbf{x} = -\begin{bmatrix} v_{1,n+1}/v_{n+1,n+1} \\ \vdots \\ v_{n,n+1}/v_{n+1,n+1} \end{bmatrix}$$

where the v_{ij} refers to the entries of $V = [v_{ij}]_{i,j=1}^{n+1}$

5. Finding closest unitary/orthogonal matrix

- let $U(n)$ be the set of all $n \times n$ unitary matrices
- given $A \in \mathbb{C}^{n\times n}$, we wish to find the matrix $X \in U(n)$ that satisfies

$$\min_{X\in U(n)} \|A - X\|_F$$

- let $A = U\Sigma V^*$ be the SVD of A
- if we set

$$X = UV^*,$$

then

$$\|A - X\|_F^2 = \|U(\Sigma - I)V^*\|_F^2 = \|\Sigma - I\|_F^2 = (\sigma_1 - 1)^2 + \cdots + (\sigma_n - 1)^2$$

- it can be shown that this is in fact the minimum (see Homework 2)
- for real matrices A, one could also ask for

$$\min_{X\in O(n)} \|A - X\|_F$$

which is just a special case

6. Procrustes problem

- a more general problem is to find $X \in U(n)$ such that

$$\min_{X\in U(n)} \|A - BX\|_F$$

for given matrices $A, B \in \mathbb{C}^{m\times n}$
- let $B^*A = U\Sigma V^*$ be the SVD of B^*A
- the solution is given by

$$X = UV^*$$

- you will be asked to prove this in Homework 2

7. aside: closest Hermitian/symmetric matrix

- this one doesn't require SVD but is interesting nonetheless
- given $A \in \mathbb{C}^{n\times n}$, find its closest Hermitian matrix
$$\min_{X^*=X} \|A - X\|_F \tag{7.1}$$
or its closest skew-Hermitian matrix
$$\min_{X^*=-X} \|A - X\|_F \tag{7.2}$$
- note that any square matrix can be written as a sum of a Hermitian matrix and a skew-Hermitian matrix
$$A = \frac{1}{2}(A + A^*) + \frac{1}{2}(A - A^*)$$
- the solutions to (7.1) and (7.2) are given by $X = \frac{1}{2}(A+A^*)$ and $X = \frac{1}{2}(A-A^*)$ respectively (why?)
- for $A \in \mathbb{R}^{n\times n}$ these yield the closest symmetric and skew-symmetric matrices to A

8. other applications

- in the homework you see yet other uses of SVD
- here are some other uses of SVD that we didn't have time to consider:
  - least squares with linear constraints (we will discuss this under QR though)
  - least squares with quadratic constraints
  - finding angles between subspaces
  - orthonormal basis for intersection of subspaces
  - subset selection
- all these should convince you that SVD truly is a swiss army knife of matrix computations