

11 Lecture 11, Feb 9

Announcements

- HW3 due Feb 16 @ 11:59PM.
- Quiz 2 today. In class, closed book.

Last time

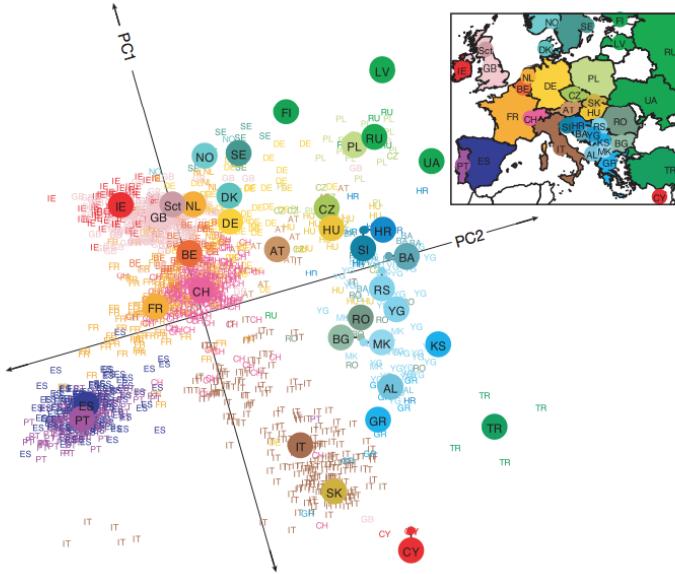
- Iterative methods for solving linear equations: Jacobi, Gauss-Seidel, successive over-relaxation, conjugate gradient (to do later).
- A catalog of easy linear systems.
- Linear algebra review: eigen-decomposition and SVD.

Today

- Applications of eigen-decomposition and SVD.
- Algorithms for eigen-decomposition and SVD.
Reading: *The QR algorithm* by Beresford N. Parlett. <http://hua-zhou.github.io/teaching/biostatm280-2016winter/readings/qr.pdf>
- Concluding remarks of numerical linear algebra.

Applications of eigen-decomposition and SVD

- Principal components analysis (PCA). $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a centered data matrix. Perform SVD $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$ or equivalently $\mathbf{X}^\top\mathbf{X} = \mathbf{V}\Sigma^2\mathbf{V}^\top$. The linear combinations $\tilde{\mathbf{x}}_i = \mathbf{X}\mathbf{v}_i$ are the principal components (PC) and have variance σ_i^2 . Usages:
 1. Dimension reduction: reduce dimensionality p to $q \ll p$. Use top PCs $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_q$ in downstream analysis.
 2. Use PCs to adjust for confounding – a serious issue in association studies in large data sets.



- Low rank approximation, e.g., image/data compression.

Eckart-Young theorem:

$$\min_{\text{rank}(\mathbf{Y})=r} \|\mathbf{X} - \mathbf{Y}\|_F^2$$

is achieved by $\mathbf{Y} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$ with optimal value $\sum_{i=1}^{r-1} \sigma_i^2$, where $(\sigma_i, \mathbf{u}_i, \mathbf{v}_i)$ are singular values and vectors of \mathbf{X} .

Gene Golub's 2691×598 picture requires $2691 \times 598 \times 6 = 9,655,308$ bytes (RGB 16 bit per channel). Rank 120 approximation requires $120 \times (2691 + 598) \times 6 = 2,368,080$ bytes. Rank 50 approximation requires $50 \times (2691 + 598) \times 6 = 986,700$ bytes. Rank 12 approximation requires $12 \times (2691 + 598) \times 8 = 236,808$ bytes.

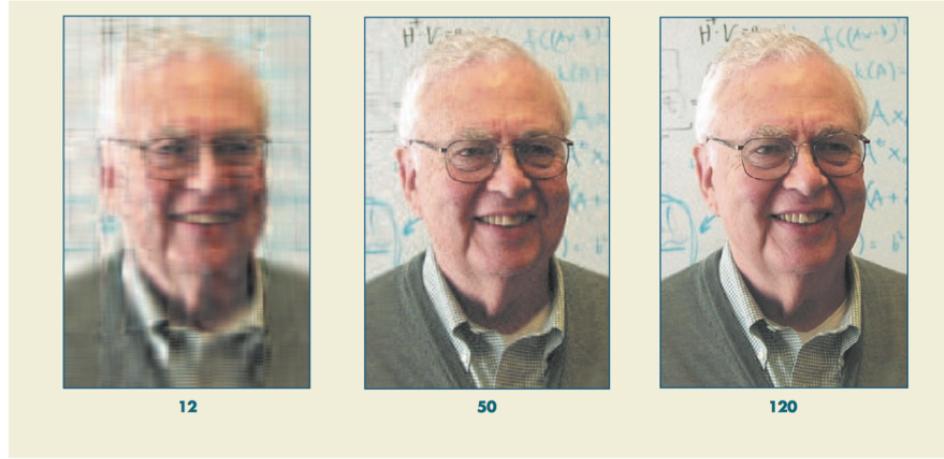


Figure 2. Rank 12, 50, and 120 approximations to a rank 598 color photo of Gene Golub.

- Least squares, ridge regression (HW3,4), least squares over a sphere, ...
- Moore-Penrose (MP) inverse: Using thin SVD,

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

where $\Sigma^+ = \text{diag}(\sigma_1^{-1}, \dots, \sigma_r^{-1}, 0, \dots, 0)$, $r = \text{rank}(\mathbf{A})$. This is how the `ginv()` function is implemented in `MASS` package.

- Read KL Chapters 8 and 9 and do HW4 for some more applications.

Algorithm: One eigen-pair - power method

Assume $\mathbf{A} \in \mathbb{R}^{n \times n}$ symmetric.

- *Power method* iterates according to

$$\begin{aligned}\mathbf{x}^{(t)} &\leftarrow \frac{1}{\|\mathbf{A}\mathbf{x}^{(t-1)}\|_2} \mathbf{A}\mathbf{x}^{(t-1)} \\ \lambda_1^{(t)} &\leftarrow \mathbf{x}^{(t)\top} \mathbf{A} \mathbf{x}^{(t)}\end{aligned}$$

from an initial guess $\mathbf{x}^{(0)}$ of unit norm.

- Suppose we arrange $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$ (the first inequality strict) with

corresponding eigenvectors \mathbf{u}_i , and expand $\mathbf{x}^{(0)} = c_1\mathbf{u}_1 + \cdots + c_n\mathbf{u}_n$, then

$$\begin{aligned}\mathbf{x}^{(t)} &= \frac{(\sum_i \lambda_i^t \mathbf{u}_i \mathbf{u}_i^\top) (\sum_i c_i \mathbf{u}_i)}{\|(\sum_i \lambda_i^t \mathbf{u}_i \mathbf{u}_i^\top) (\sum_i c_i \mathbf{u}_i)\|_2} \\ &= \frac{\sum_i c_i \lambda_i^t \mathbf{u}_i}{\|\sum_i c_i \lambda_i^t \mathbf{u}_i\|_2} \\ &= \frac{c_1 \mathbf{u}_1 + c_2 (\lambda_2/\lambda_1)^t \mathbf{u}_2 + \cdots + c_n (\lambda_n/\lambda_1)^t \mathbf{u}_n}{\|c_1 \mathbf{u}_1 + c_2 (\lambda_2/\lambda_1)^t \mathbf{u}_2 + \cdots + c_n (\lambda_n/\lambda_1)^t \mathbf{u}_n\|_2} \left(\frac{\lambda_1}{|\lambda_1|}\right)^t.\end{aligned}$$

Thus $\mathbf{x}^{(t)} - \frac{c_1 \mathbf{u}_1}{\|c_1 \mathbf{u}_1\|_2} \left(\frac{\lambda_1}{|\lambda_1|}\right)^t \rightarrow 0$ as $t \rightarrow \infty$. The convergence rate is $|\lambda_2|/|\lambda_1|$.

- $\mathbf{x}^{(t)^\top} \mathbf{A} \mathbf{x}^{(t)}$ converges to λ_1 .
- *Inverse power method* for finding the eigenvalue of smallest absolute value: Substitute \mathbf{A} by \mathbf{A}^{-1} in the power method. (E.g., pre-compute LU or Cholesky of \mathbf{A}).
- *Shifted inverse power*: Substitute $(\mathbf{A} - \mu \mathbf{I})^{-1}$ in the power method. It converges to an eigenvalue close to the given μ .
- Initial guess of the desired eigenvalue can be obtain by Gerschgorin's circle theorem and so on.
- Power method also applies to asymmetric \mathbf{A} , e.g., PageRank problem costs $O(n)$ per iteration.

Algorithm: Top r eigen-pairs - orthogonal iteration

Generalization of power method to higher dimensional invariant subspace.

- *Orthogonal iteration*: Initialize $\mathbf{Q}^{(0)} \in \mathbb{R}^{n \times r}$ with orthonormal columns. For $t = 1, 2, \dots$,

$$\begin{aligned}\mathbf{Z}^{(t)} &\leftarrow \mathbf{A} \mathbf{Q}^{(t-1)} \quad (2n^2r \text{ flops}) \\ \mathbf{Q}^{(t)} \mathbf{R}^{(t)} &\leftarrow \mathbf{Z}^{(t)} \quad (\text{QR factorization})\end{aligned}$$

- $\mathbf{Z}^{(t)}$ converges to the eigenspace of the largest r eigenvalues if they are real and separated from remaining spectrum. The convergence rate is $|\lambda_{r+1}|/|\lambda_r|$.

Algorithm: (Impractical) full eigen-decomposition - QR iteration

Assume $\mathbf{A} \in \mathbb{R}^{n \times n}$ symmetric.

- take $r = n$ in the orthogonal iteration. Then $\mathbf{Q}^{(t)}$ converges to the eigenspace \mathbf{U} of \mathbf{A} . This implies that

$$\mathbf{T}^{(t)} := \mathbf{Q}^{(t) T} \mathbf{A} \mathbf{Q}^{(t)}$$

converges to a diagonal form $\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_n)$.

- Note how to compute $\mathbf{T}^{(t)}$ from $\mathbf{T}^{(t-1)}$

$$\begin{aligned}\mathbf{T}^{(t-1)} &= \mathbf{Q}^{(t-1) T} \mathbf{A} \mathbf{Q}^{(t-1)} = \mathbf{Q}^{(t-1) T} (\mathbf{A} \mathbf{Q}^{(t-1)}) = (\mathbf{Q}^{(t-1) T} \mathbf{Q}^{(t)}) \mathbf{R}^{(t)} \\ \mathbf{T}^{(t)} &= \mathbf{Q}^{(t) T} \mathbf{A} \mathbf{Q}^{(t)} = \mathbf{Q}^{(t) T} \mathbf{A} \mathbf{Q}^{(t-1)} \mathbf{Q}^{(t-1) T} \mathbf{Q}^{(t)} = \mathbf{R}^{(t)} (\mathbf{Q}^{(t-1) T} \mathbf{Q}^{(t)}).\end{aligned}$$

- *QR iteration:* Initialize $\mathbf{U}^{(0)} \in \mathbb{R}^{n \times n}$ orthogonal and set $\mathbf{T}^{(0)} = \mathbf{U}^{(0) T} \mathbf{A} \mathbf{U}^{(0)}$.
For $t = 1, 2, \dots$

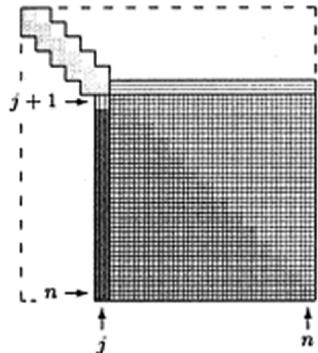
$$\begin{aligned}\mathbf{U}^{(t)} \mathbf{R}^{(t)} &\leftarrow \mathbf{T}^{(t-1)} \quad (\text{QR factorization}) \\ \mathbf{T}^{(t)} &\leftarrow \mathbf{R}^{(t)} \mathbf{U}^{(t)}\end{aligned}$$

- QR iteration is expensive: $O(n^3)$ per iteration and linear convergence rate.

QR algorithm for symmetric eigen-decomposition

Assume $\mathbf{A} \in \mathbb{R}^{n \times n}$ symmetric.

- As done in LAPACK: `eigen()` in R, `eig()` in Matlab and Julia.
- Idea: Tri-diagonalization (by Householder) + QR iteration on the tri-diagonal system with implicit shift
 - Step 1: Householder tri-diagonalization: $4n^3/3$ for eigenvalues only, $8n^3/3$ for both eigenvalues and eigenvectors. (Why can't we apply Householder to make it diagonal directly?)

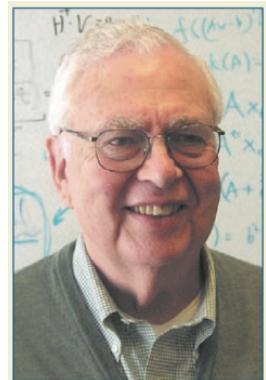


- Step 2: QR iteration on the tridiagonal matrix. Implicit shift accelerates convergence rate. On average 1.3-1.6 QR iteration per eigenvalue, $\sim 20n$ flops per QR iteration. So total operation count is about $30n^2$. Eigenvectors need an extra of about $6n^3$ flops.

	Eigenvalue	Eigenvector
Householder reduction	$4n^3/3$	$4n^3/3$
QR with implicit shift	$\sim 30n^2$	$\sim 6n^3$

- Message: Don't request eigenvectors unless necessary: set `only.values = TRUE` when calling `eigen()` in R.
- The *unsymmetric QR algorithm* obtains the real Schur decomposition of an asymmetric matrix \mathbf{A} .

Algorithm for singular value decomposition (SVD)



Gene Golub's license plate, photographed by Professor P.M. Kroonenberg of Leiden University.

Assume $\mathbf{A} \in \mathbb{R}^{m \times n}$ and we seek the SVD $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$.

- “Golub-Kahan-Reinsch” algorithm:

- Stage 1: Transform \mathbf{A} to an upper bidiagonal form \mathbf{B} (by Householder).

$$\mathbf{U}_B^T \mathbf{A} \mathbf{V}_B = \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} d_1 & f_1 & & \cdots & 0 \\ 0 & d_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & f_{n-1} \\ 0 & \cdots & 0 & d_n & \end{bmatrix} \in \mathbb{R}^{n \times n}$$

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \xrightarrow{U_1} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{V_1}$$

$$\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{U_2} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{V_2}$$

$$\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{U_3} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \xrightarrow{U_4} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

– Stage 2: Apply implicit-shift QR step to the tridiagonal matrix $\mathbf{B}^\top \mathbf{B}$ implicitly.

- See Golub and Van Loan (1996, Section 8.6) for more details.
- $4m^2n + 8mn^2 + 9n^3$ flops for a tall ($m \geq n$) matrix.
- `svd()` in R and Matlab: wrapper of the `_GESVD` subroutine in LAPACK.

Lanczos/Arnoldi iterative method for top eigen-pairs

- Motivation
 - Consider the Google PageRank problem. We want to find the top left eigenvector of the transition matrix

$$\mathbf{P} = p\mathbf{R}^+ \mathbf{A} + z\mathbf{1}_n^\top,$$

where $\mathbf{R} = \text{diag}(r_1, \dots, r_n)$ and $z_j = (1 - p)/n$ if $r_i > 0$ and $1/n$ if $r_i = 0$. Suppose there are $n \approx$ billion web pages.

The (unsymmetric) QR algorithm will take order

$$\frac{(1 \times 10^9)^3}{10^{12}} \approx 3.33 \times 10^{14} \text{ seconds} \approx 1 \times 10^7 \text{ years}$$

on a tera-flop supercomputer!

- Consider adjusting for confounding by PCA in modern GWAS (genome-wide association studies). We want to find the top singular values/vectors of a genotype matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, where $n \sim 10^3$ and $p \sim 10^6$.
- *Krylov subspace methods* are the state-of-art iterative method for obtaining the top eigen-values/vectors or singular values/vectors of large sparse or structured matrices.
- Lanczos method: top eigen-pairs of a large *symmetric* matrix.
- Arnoldi method: top eigen-pairs of a large *asymmetric* matrix.
- Both methods are also adapted to obtain top singular values/vectors of large sparse or structured matrices.
- We will give an overview of these methods together with the conjugate gradient method for solving large linear system.
- `eigs()` and `svds()` in Matlab and Julia are wrappers of the ARPACK package.
No native functions in R (?).

Algorithm: Jacobi method for symmetric eigen-decomposition (KL 8.2)

Assume $\mathbf{A} \in \mathbb{R}^{n \times n}$ is symmetric and we seek the eigen-decomposition $\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^T$.

- Idea: Systematically reduce off-diagonal entries

$$\text{off}(\mathbf{A}) = \sum_i \sum_{j \neq i} a_{ij}^2$$

by Jacobi rotations.

- Jacobi/Givens rotations:

$$\mathbf{J}(p, q, \theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ \vdots & \vdots & \vdots & \ddots \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$\mathbf{J}(p, q, \theta)$ is orthogonal.

- Consider $\mathbf{B} = \mathbf{J}^\top \mathbf{A} \mathbf{J}$. \mathbf{B} preserves the symmetry and eigenvalues of \mathbf{A} .

Taking

$$\begin{cases} \tan(2\theta) = 2a_{pq}/(a_{qq} - a_{pp}) & \text{if } a_{pp} \neq a_{qq} \\ \theta = \pi/4 & \text{if } a_{pp} = a_{qq} \end{cases}$$

forces $b_{pq} = 0$.

- Since orthogonal transform preserves Frobenius norm, we have

$$b_{pp}^2 + b_{qq}^2 = a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2.$$

(Just check the 2-by-2 block)

- Since $\|\mathbf{A}\|_F = \|\mathbf{B}\|_F$, this implies that the off-diagonal part

$$\text{off}(\mathbf{B}) = \text{off}(\mathbf{A}) - 2a_{pq}^2$$

is decreased whenever $a_{pq} \neq 0$.

- One Jacobi rotation costs $O(n)$ flops.
- *Classical Jacobi*: search for the largest $|a_{ij}|$ at each iteration.
- $\text{off}(\mathbf{A}) \leq n(n-1)a_{ij}^2$ and $\text{off}(\mathbf{B}) = \text{off}(\mathbf{A}) - 2a_{ij}^2$ together implies

$$\text{off}(\mathbf{B}) \leq \left(1 - \frac{2}{n(n-1)}\right) \text{off}(\mathbf{A}).$$

- In practice, cyclic-by-row implementation, to avoid the costly $O(n^2)$ search in the classical Jacobi.
- Jacobi method attracts a lot recent attention because of its rich inherent parallelism.
- *Parallel Jacobi*: “merry-go-round” to generate parallel ordering.

432 CHAPTER 8. THE SYMMETRIC EIGENVALUE PROBLEM

lism of the latter algorithm. To illustrate this, suppose $n = 4$ and group the six subproblems into three *rotation sets* as follows:

$$\begin{aligned} \text{rot.set}(1) &= \{(1,2), (3,4)\} \\ \text{rot.set}(2) &= \{(1,3), (2,4)\} \\ \text{rot.set}(3) &= \{(1,4), (2,3)\} \end{aligned}$$

Note that all the rotations within each of the three rotation sets are “nonconflicting.” That is, subproblems (1,2) and (3,4) can be carried out in parallel. Likewise the (1,3) and (2,4) subproblems can be executed in parallel as can subproblems (1,4) and (2,3). In general, we say that

$$(i_1, j_1), (i_2, j_2), \dots, (i_N, j_N) \quad N = (n-1)n/2$$

is a *parallel ordering* of the set $\{(i,j) \mid 1 \leq i < j \leq n\}$ if for $s = 1:n-1$ the rotation set $\text{rot.set}(s) = \{(i_r, j_r) : r = 1 + n(s-1)/2:ns/2\}$ consists of nonconflicting rotations. This requires n to be even, which we assume of throughout this section. (The odd n case can be handled by bordering A with a row and column of zeros and being careful when solving the subproblems that involve these augmented zeros.)

A good way to generate a parallel ordering is to visualize a chess tournament with n players in which everybody must play everybody else exactly once. In the $n = 8$ case this entails 7 “rounds.” During round one we have the following four games:

1	3	5	7
2	4	6	8

$$\text{rot.set}(1) = \{(1,2), (3,4), (5,6), (7,8)\}$$

i.e., 1 plays 2, 3 plays 4, etc. To set up rounds 2 through 7, player 1 stays put and players 2 through 8 embark on a merry-go-round:

1	2	3	5
4	6	8	7

$$\text{rot.set}(2) = \{(1,4), (2,6), (3,8), (5,7)\}$$

1	4	2	3
6	8	7	5

$$\text{rot.set}(3) = \{(1,6), (4,8), (2,7), (3,5)\}$$

1	6	4	2
8	7	5	3

$$\text{rot.set}(4) = \{(1,8), (6,7), (4,5), (2,3)\}$$

1	8	6	4
7	5	3	2

$$\text{rot.set}(5) = \{(1,7), (5,8), (3,6), (2,4)\}$$

1	7	8	6
5	3	2	4

$$\text{rot.set}(6) = \{(1,5), (3,7), (2,8), (4,6)\}$$

8.4.4 JACOBI METHODS

$\boxed{\begin{array}{|c|c|c|c|}\hline 1 & 5 & 7 & 8 \\ \hline 3 & 2 & 4 & 6 \\ \hline \end{array}}$ $\text{rot.set}(7) = \{(1,3), (2,5), (4,7), (6,8)\}$

We can encode these operations in a pair of integer vectors $\text{top}(1:n/2)$ and $\text{bot}(1:n/2)$. During a given round $\text{top}(k)$ plays $\text{bot}(k)$, $k = 1:n/2$. The pairings for the next round is obtained by updating top and bot as follows:

```
function: [new.top,new.bot] = music(top,bot,n)
    m = n/2
    for k = 1:m
        if k = 1
            new.top(1) = 1
        else if k = 2
            new.top(k) = bot(1)
        elseif k > 2
            new.top(k) = top(k-1)
        end
        if k = m
            new.bot(k) = top(k)
        else
            new.bot(k) = bot(k+1)
        end
    end
```

Using `music` we obtain the following parallel order Jacobi procedure.

Algorithm 8.4.4 (Parallel Order Jacobi) Given a symmetric $A \in \mathbb{R}^{n \times n}$ and a tolerance $\text{tol} > 0$, this algorithm overwrites A with $V^T A V$ where V is orthogonal and $\text{off}(V^T A V) \leq \text{tol} \|A\|_F$. It is assumed that n is even.

```
V = I_n
eps = tol \|A\|_F
top = 1:2:n; bot = 2:2:n
while off(A) > eps
    for set = 1:n-1
        for k = 1:n/2
            p = min(top(k),bot(k))
            q = max(top(k),bot(k))
            (c, s) = sym.schur2(A,p,q)
            A = J(p,q,theta)^T A J(p,q,theta)
            V = V J(p,q,theta)
        end
        [top,bot] = music(top,bot,n)
    end
end
```

Generalized eigen-problem

- Generalized eigen-problem: $\mathbf{Ax} = \lambda \mathbf{Bx}$, where \mathbf{A} psd and \mathbf{B} pd.
- Applications: partial least squares (PLS), sliced inverse regression (SIR), canonical correlation analysis (CCA).

- Method 1: $\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$. Non-symmetric eigen-problem \circledcirc .
- Method 2: Cholesky $\mathbf{B} = \mathbf{L}\mathbf{L}^\top$. Then $\mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}\mathbf{y} = \lambda\mathbf{y}$ where $\mathbf{y} = \mathbf{L}^\top\mathbf{x}$.
- Method 3 (most numerically stable, \mathbf{B} can be rank deficient): QZ algorithm.
- `eig()` and `qz()` in Matlab and Julia implement QZ. No native function in R?

Generalized singular value decomposition

- $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times n}$. Then there exists orthogonal $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{p \times p}$ and an invertible $\mathbf{X} \in \mathbb{R}^{n \times n}$ such that

$$\begin{aligned}\mathbf{U}^T \mathbf{A} \mathbf{X} &= \mathbf{C} = \text{diag}(c_1, \dots, c_n), \quad c_i \geq 0 \\ \mathbf{V}^T \mathbf{B} \mathbf{X} &= \mathbf{S} = \text{diag}(s_1, \dots, s_q), \quad s_i \geq 0,\end{aligned}$$

where $q = \min\{p, n\}$.

- Applications: quadratically inequality-constrained least squares problem (LSQI).
- `gsvd()` in Matlab implements generalized SVD. No native function in R?

In the zoo of least squares (self-study)

Weighted least squares

- In weighted least squares, we minimize $\sum_{i=1}^n w_i(y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2$, where $w_i > 0$ are observation weights.
- Let $\mathbf{W} = \text{diag}(w_1, \dots, w_n)$. Then the criterion is $\|\mathbf{W}^{1/2}\mathbf{y} - \mathbf{W}^{1/2}\mathbf{X}\boldsymbol{\beta}\|_2^2$, which can be solved by standard methods for least squares with $\tilde{\mathbf{y}} = \mathbf{W}^{1/2}\mathbf{y}$ and $\tilde{\mathbf{X}} = \mathbf{W}^{1/2}\mathbf{X}$.

General least squares

- In Aitken model: $E(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta}$, $\text{Cov}(\mathbf{y}) = \sigma^2\mathbf{V}$, where \mathbf{V} is a positive semidefinite matrix. We minimize the generalized least squares criterion

$$(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T \mathbf{M}^{-1} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}),$$

where $\mathbf{M} \in \mathbb{R}^{n \times n}$ is some positive semidefinite matrix, e.g., $\mathbf{M} = \mathbf{V}$ for non-singular \mathbf{V} or $\mathbf{M} = \mathbf{V} + \mathbf{X}\mathbf{X}^T$ for singular \mathbf{V} .

- Let $\mathbf{M} = \mathbf{B}\mathbf{B}^T$ for some $\mathbf{B} \in \mathbb{R}^{n \times n}$ (e.g., the Cholesky factor). One approach is to minimize

$$\|\mathbf{B}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\|_2^2.$$

Unfortunately, when \mathbf{B} is poorly conditioned (or even not invertible), the procedure produces a poor solution.

- Paige's method. The generalized least squares problem is equivalent to

$$\begin{aligned} & \text{minimize} && \mathbf{v}^T \mathbf{v} \\ & \text{subject to} && \mathbf{X}\boldsymbol{\beta} + \mathbf{B}\mathbf{v} = \mathbf{y}. \end{aligned}$$

To solve this problem, first compute the QR of \mathbf{X}

$$\mathbf{X} = (\mathbf{Q}_1, \mathbf{Q}_2) \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{pmatrix}.$$

Compute another QR for the (flat) matrix $\mathbf{Q}_2^T \mathbf{B}$ such that

$$\mathbf{Q}_2^T \mathbf{B} = (\mathbf{0}, \mathbf{S}) \begin{pmatrix} \mathbf{Z}_1^T \\ \mathbf{Z}_2^T \end{pmatrix},$$

where \mathbf{S} is upper triangular and $(\mathbf{Z}_1, \mathbf{Z}_2) \in \mathbb{R}^{n \times n}$ is orthogonal. Then the constraint becomes

$$\begin{pmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{pmatrix} \boldsymbol{\beta} + \begin{pmatrix} \mathbf{Q}_1^T \mathbf{B} \mathbf{Z}_1 & \mathbf{Q}_1^T \mathbf{B} \mathbf{Z}_2 \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \begin{pmatrix} \mathbf{Z}_1^T \mathbf{v} \\ \mathbf{Z}_2^T \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_1^T \mathbf{y} \\ \mathbf{Q}_2^T \mathbf{y} \end{pmatrix}.$$

From the bottom half we can solve for \mathbf{v} from the equation (how?)

$$\mathbf{S} \mathbf{Z}_2^T \mathbf{v} = \mathbf{Q}_2^T \mathbf{y}.$$

Then we solve for $\boldsymbol{\beta}$ from the equation

$$\mathbf{R}_1 \boldsymbol{\beta} = \mathbf{Q}_1^T \mathbf{y} - (\mathbf{Q}_1^T \mathbf{B} \mathbf{Z}_1 \mathbf{Z}_1^T + \mathbf{Q}_1^T \mathbf{B} \mathbf{Z}_2 \mathbf{Z}_2^T) \mathbf{v} = \mathbf{Q}_1^T \mathbf{y} - \mathbf{Q}_1^T \mathbf{B} \mathbf{Z}_2 (\mathbf{Z}_2^T \mathbf{v}).$$

- Paige's method also works for singular \mathbf{X} and \mathbf{B} (using QR with column pivoting).
- MATLAB's `lscov()` function implements Paige's method for singular covariance \mathbf{V} . No R implementation (?)

Ridge regression

- In ridge regression, we minimize

$$\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_2^2,$$

where λ is a tuning parameter.

- Ridge regression by augmented linear regression. Ridge regression problem is equivalent to

$$\left\| \begin{pmatrix} \mathbf{y} \\ \mathbf{0}_p \end{pmatrix} - \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda} \mathbf{I}_p \end{pmatrix} \beta \right\|_2^2.$$

Therefore any methods for linear regression can be applied.

- Ridge regression by method of normal equation. The normal equation for the ridge problem is

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_p) \beta = \mathbf{X}^T \mathbf{y}.$$

Therefore Cholesky or sweep can be used.

- Ridge regression by SVD. If we obtain the (thin) SVD of \mathbf{X}

$$\mathbf{X} = \mathbf{U} \Sigma_{p \times p} \mathbf{V}^T.$$

Then the normal equation reads

$$(\Sigma^2 + \lambda \mathbf{I}_p) \mathbf{V}^T \beta = \Sigma \mathbf{U}^T \mathbf{y}$$

and we get

$$\hat{\beta}(\lambda) = \sum_{i=1}^p \frac{\sigma_i \mathbf{u}_i^T \mathbf{y}}{\sigma_i^2 + \lambda} \mathbf{v}_i = \sum_{i=1}^r \frac{\sigma_i \mathbf{u}_i^T \mathbf{y}}{\sigma_i^2 + \lambda} \mathbf{v}_i, \quad r = \text{rank}(\mathbf{X}).$$

It is clear that

$$\lim_{\lambda \rightarrow 0} \hat{\beta}(\lambda) = \hat{\beta}_{\text{OLS}}$$

and $\|\hat{\beta}(\lambda)\|_2$ is monotone decreasing as λ increases.

- Only one SVD is needed for all λ (!), in contrast to the method of augmented linear regression, Cholesky, or sweep.

Least squares over a sphere

- Ridge regression “shrinks” the solution via penalty. Alternatively we can simply fit a least squares problem subject to the constraint that the solution lives in a sphere

$$\begin{aligned} & \text{minimize} && \|\mathbf{y} - \mathbf{X}\beta\|_2^2 \\ & \text{subject to} && \|\beta\|_2 \leq \alpha. \end{aligned}$$

- Suppose we obtain the (thin) SVD $\mathbf{X} = \mathbf{U}\Sigma_{p \times p}\mathbf{V}^T$. If the ordinary least squares solution

$$\hat{\beta}_{\text{OLS}} = \sum_{i=1}^r \frac{\mathbf{u}_i^T \mathbf{y}}{\sigma_i} \mathbf{v}_i$$

has ℓ_2 norm less than α , then we are done. If not, we use the method of Lagrangian multipliers

$$\psi(\beta, \lambda) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \frac{\lambda}{2}(\|\beta\|_2^2 - \alpha^2).$$

Setting the gradient to 0, we have the shifted normal equation

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_p) \beta = \mathbf{X}^T \mathbf{y},$$

which has solution

$$\hat{\beta}(\lambda) = \sum_{i=1}^r \frac{\sigma_i \mathbf{u}_i^T \mathbf{y}}{\sigma_i^2 + \lambda} \mathbf{v}_i.$$

We need to choose the λ such that $\|\hat{\beta}(\lambda)\|_2 = \alpha$. That is we need to find the (unique) zero of the function

$$f(\lambda) = \|\hat{\beta}(\lambda)\|_2^2 - \alpha^2 = \sum_{i=1}^r \left(\frac{\sigma_i \mathbf{u}_i^T \mathbf{y}}{\sigma_i^2 + \lambda} \right)^2 - \alpha^2.$$

This is easily achieved by Newton's or other methods.

Least squares with equality constraints

- In many applications, there are *a priori* constraints on the regression parameters. Let's consider how to solve linear regression with equality constraints (LSE)

$$\begin{aligned} & \text{minimize} && \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \\ & \text{subject to} && \mathbf{B}\boldsymbol{\beta} = \mathbf{d}. \end{aligned}$$

- LSE by QR. First compute QR of $\mathbf{B}^T \in \mathbb{R}^{p \times m}$

$$\mathbf{B}^T = \mathbf{Q} \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix}$$

and set

$$\mathbf{X}\mathbf{Q} = (\mathbf{X}_1, \mathbf{X}_2) \quad \text{and} \quad \mathbf{Q}^T \boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix}.$$

Then the original minimization problem becomes

$$\begin{aligned} & \text{minimize} && \|\mathbf{y} - \mathbf{X}_1\boldsymbol{\beta}_1 - \mathbf{X}_2\boldsymbol{\beta}_2\|_2^2 \\ & \text{subject to} && \mathbf{R}^T \boldsymbol{\beta}_1 = \mathbf{d}. \end{aligned}$$

Now $\boldsymbol{\beta}_1$ is determined from the constraint $\mathbf{R}^T \boldsymbol{\beta}_1 = \mathbf{d}$ and $\boldsymbol{\beta}_2$ is solved from the unconstrained least squares problem

$$\text{minimize } \|(\mathbf{y} - \mathbf{X}_1\boldsymbol{\beta}_1) - \mathbf{X}_2\boldsymbol{\beta}_2\|_2^2.$$

Finally we recover the solution from

$$\boldsymbol{\beta} = \mathbf{Q} \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix}.$$

- LSE by augmented system. Define the Lagrangian function

$$\phi(\boldsymbol{\beta}, \boldsymbol{\lambda}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 - \boldsymbol{\lambda}^T (\mathbf{B}\boldsymbol{\beta} - \mathbf{d}).$$

Setting gradient to zero yields

$$\begin{aligned} \mathbf{X}^T \mathbf{X} \boldsymbol{\beta} - \mathbf{B}^T \boldsymbol{\lambda} &= \mathbf{X}^T \mathbf{y} \\ \mathbf{B} \boldsymbol{\beta} &= \mathbf{d}, \end{aligned}$$

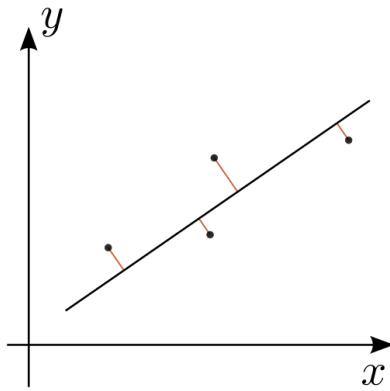
suggesting the augmented system

$$\begin{pmatrix} \mathbf{X}^T \mathbf{X} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta} \\ -\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{X}^T \mathbf{y} \\ \mathbf{d} \end{pmatrix}.$$

This linear system is non-singular when \mathbf{X} and \mathbf{B} have full rank and can be solved by Cholesky, sweep, and so on.

- LSE by generalized SVD.

Total least squares (TLS)



TLS considers the case both predictors and observations are subject to errors. It is solved by SVD. Read KL 9.3.6 if interested.

Tikhonov regularization

Tikhonov regularization is an extension of the ridge regression

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\mathbf{B}\boldsymbol{\beta}\|_2^2,$$

where $\mathbf{B} \in \mathbb{R}^{m \times p}$ is a fixed regularization matrix and λ is a tuning parameter. It is solved by the generalized singular value decomposition (GSVD).

Least squares with quadratic inequality constraint (LSQI)

Least squares with quadratic inequality constraint (LSQI) minimizes the least squares criterion over a hyper-ellipsoid:

$$\begin{aligned} & \text{minimize} && \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \\ & \text{subject to} && \|\mathbf{B}\boldsymbol{\beta}\|_2 \leq \alpha, \end{aligned}$$

where $\mathbf{B} \in \mathbb{R}^{m \times p}$ is a fixed regularization matrix. It is solved by the generalized singular value decomposition (GSVD). See Golub and Van Loan (1996, Section 2.1.1).

Concluding remarks on numerical linear algebra

- Numerical linear algebra forms the building blocks of most computation we do.
- Be flop and memory aware.

The form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different.

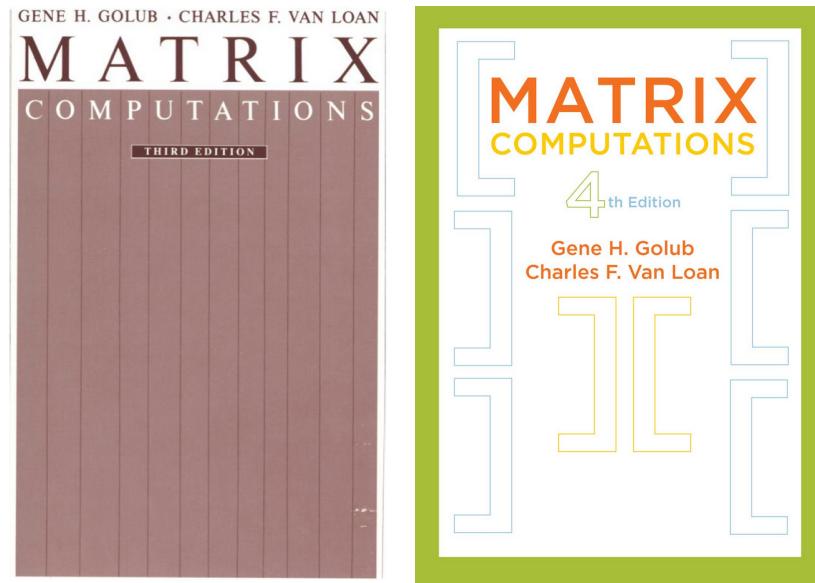
- Be alert to problem structure and make educated choice of software/algorithm.

The structure should be exploited whenever solving a problem.

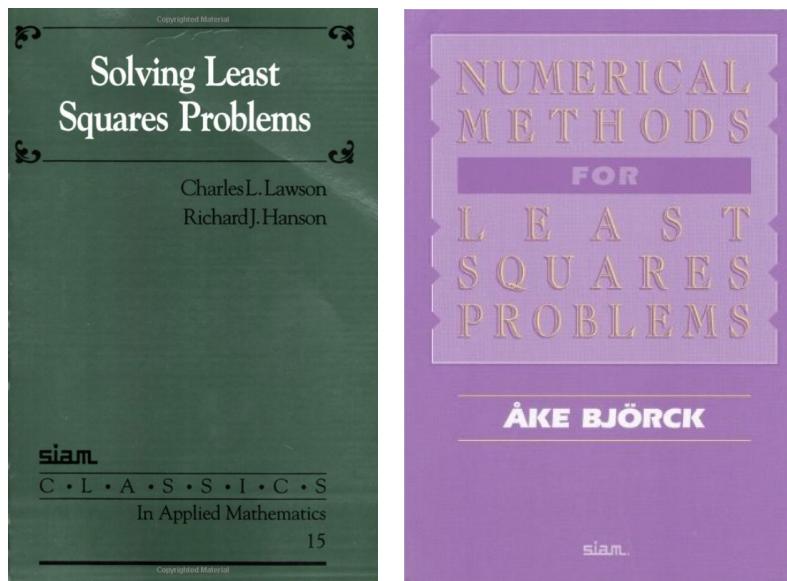
- Do not write your own matrix computation routines unless for good reason.
Utilize BLAS and LAPACK as much as possible!
- In contrast, for optimization, often we need to devise problem specific optimization routines, or even “mix and match” them.

Reference books on numerical linear algebra

- Golub and Van Loan (1996): “Bible” in numerical linear algebra. Good for reference.



- Lawson and Hanson (1987) and Björck (1996): classical monographs on solving least squares problems.



- Saad (2003): standard reference for iterative methods

