STAT 3690 Lecture Note

Part I: R and matrix basics

Zhiyang Zhou (zhiyang.zhou@umanitoba.ca, zhiyanggeezhou.github.io)

2023/Feb/27 08:27:57

Matrix basics

Matrix decomposition

- Eigen-decomposition (for square matrix $\mathbf{A}_{n\times n}$): $\mathbf{A}=\mathbf{V}\Lambda\mathbf{V}^{-1}$
  - $\Lambda=\operatorname{diag}(\lambda_1,\ldots,\lambda_n)$
    - $\lambda_1\ge\cdots\ge\lambda_n$ are the eigenvalues of $\mathbf{A}$, i.e., n roots of characteristic equation $\det(\lambda\mathbf{I}_n-\mathbf{A})=0$
  - $\mathbf{V}=[\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n]_{n\times n}$
    - $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n$ are (right) eigenvectors of $\mathbf{A}$, i.e., $\mathbf{A}\boldsymbol{v}_i=\lambda_i\boldsymbol{v}_i$
  - Implementation in R: eigen()

- Spectral decomposition (for symmetric $\mathbf{A}$): $\mathbf{A}=\mathbf{V}\Lambda\mathbf{V}^\top$
  - $\mathbf{V}$ is orthogonal, i.e., $\mathbf{V}^\top=\mathbf{V}^{-1}$

- Singular value decomposition (SVD) for $n\times p$ matrix $\mathbf{B}$: $\mathbf{B}=\mathbf{U}\mathbf{S}\mathbf{W}^\top$
  - $\mathbf{U}=[\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n]_{n\times n}$ with $\boldsymbol{u}_i$ the ith eigenvector of $\mathbf{B}\mathbf{B}^\top$
    - $\mathbf{U}$ is orthogonal
  - $\mathbf{W}=[\boldsymbol{w}_1,\ldots,\boldsymbol{w}_p]_{p\times p}$ with $\boldsymbol{w}_i$ the ith eigenvector of $\mathbf{B}^\top\mathbf{B}$
    - $\mathbf{W}$ is orthogonal
  - $$\mathbf{S}=\left[\begin{array}{c:c}\mathbf{S}_1 & \mathbf{0}_{n\times(p-n)}\end{array}\right]_{n\times p}\text{ if } n\le p \text{ AND } \left[\begin{array}{c}\mathbf{S}_1\\ \hdashline \mathbf{0}_{(n-p)\times p}\end{array}\right]_{n\times p}\text{ if } n>p$$
    - $\mathbf{S}_1=\operatorname{diag}(s_1,\ldots,s_n)$ if $n\le p$ and $\operatorname{diag}(s_1,\ldots,s_p)$ if $n>p$
    - $s_1\ge\cdots\ge s_n$ are squre roots of eigenvalues of $\mathbf{B}\mathbf{B}^\top$
    - $s_1\ge\cdots\ge s_p$ are squre roots of eigenvalues of $\mathbf{B}^\top\mathbf{B}$
  - Thin/compact SVD for $n\times p$ matrix $\mathbf{B}$:
    $$\mathbf{B}=[\boldsymbol{u}_1,\ldots,\boldsymbol{u}_r]\operatorname{diag}(s_1,\ldots,s_r)[\boldsymbol{w}_1,\ldots,\boldsymbol{w}_r]^\top=s_1\boldsymbol{u}_1\boldsymbol{w}_1^\top+\cdots+s_r\boldsymbol{u}_r\boldsymbol{w}_r^\top$$
    - $r=\operatorname{rank}(\mathbf{B})\le\min\{n,p\}$
    - $s_1\ge\cdots\ge s_r>0$ are square roots of non-zero eigenvalues of $\mathbf{B}^\top\mathbf{B}$ or $\mathbf{B}\mathbf{B}^\top$
    - Implementation via R: svd()

- The connection of decompositions

$$\begin{array}{ccc} \text{Spectral decomposition} & \Leftarrow & \text{Eigen-decomposition} \\ \text{(for symmetric matrices)} & & \text{(for square matrices)} \\ \Uparrow & & \\ \text{(Thin) SVD} & & \\ \text{(for any matrices)} & & \end{array}$$

```
options(digits = 4) # control the number of significant digits
set.seed(1)
# Generate a symmetric matrix
A = matrix(runif(12), nrow = 2, ncol = 6)
B = t(A) %*% A # guaranteed to be symmetric
isSymmetric(B) # check symmetry
# Eigen-decomposition
(res_eigen = eigen(B))
res_eigen$vectors %*% diag(res_eigen$values) %*% t(res_eigen$vectors) - B # diff between B and decompos
# SVD
(res_svd = svd(B))
res_svd$u %*% diag(res_svd$d) %*% t(res_svd$v) - B # diff between B and decomposed B
# Thin SVD
r = qr(B)$rank # rank
res_svd$u[,1:r] %*% diag(res_svd$d[1:r]) %*% t(res_svd$v[,1:r]) - B # diff between B and decomposed B
# Comparing eigen-decomposition and SVD
res_eigen$values - res_svd$d
res_eigen$vectors - res_svd$u
res_eigen$vectors - res_svd$v
```

Square root and inverse of positive (semi-)definite matrix

- $\mathbf{A}$ is positive semi-definite (say $\mathbf{A} \geq 0$) iff $\mathbf{A}$ is symmetric and its eigenvalues are all non-negative
  - Equiv., $\boldsymbol{u}^\top \mathbf{A} \boldsymbol{u} \geq 0$ for any non-zero real n-vector $\boldsymbol{u}$ (i.e., $n \times 1$ real matrix, say $\boldsymbol{u} \in \mathbb{R}^{n\times 1}$ OR $\boldsymbol{u} \in \mathbb{R}^n$)
- If $\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^\top$ is the spectral decomposition of positive semi-definite $\mathbf{A}$, then $\mathbf{A}^{1/2} = \mathbf{V}\Lambda^{1/2}\mathbf{V}^\top$, where
  - $\Lambda^{1/2} = \mathrm{diag}(\lambda_1^{1/2}, \ldots, \lambda_n^{1/2})$
  - $\mathbf{A}^{1/2}\mathbf{A}^{1/2} = \mathbf{A}$
- $\mathbf{A}$ is positive definite (say $\mathbf{A} > 0$) iff $\mathbf{A}$ is symmetric and its eigenvalues are all positive
  - Equiv., $\boldsymbol{u}^\top \mathbf{A} \boldsymbol{u} > 0$ for all non-zero $\boldsymbol{u} \in \mathbb{R}^n$
- If $\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^\top$ is the spectral decomposition of positive definite $\mathbf{A}$, then
  - $\mathbf{A}^{-1} = \mathbf{V}\Lambda^{-1}\mathbf{V}^\top$, where $\Lambda^{-1} = \mathrm{diag}(\lambda_1^{-1}, \ldots, \lambda_n^{-1})$
  - $\mathbf{A}^{-1/2} = \mathbf{V}\Lambda^{-1/2}\mathbf{V}^\top$ is the inverse of $\mathbf{A}^{1/2}$ and also the root of $\mathbf{A}^{-1}$, where $\Lambda^{-1/2} = \mathrm{diag}(\lambda_1^{-1/2}, \ldots, \lambda_n^{-1/2})$

```
options(digits = 4) # control the number of significant digits
set.seed(1)
## Generate a demo of positive semi-definite matrices
A = matrix(runif(12), nrow = 2, ncol = 6)
B = t(A) %*% A # guaranteed to be positive semi-definite
# Get the root of B via the eigen-decomposition of B
res_eigen_B = eigen(B)
B_root1 = res_eigen_B$vectors %*%
```

```
  diag((res_eigen_B$values*(res_eigen_B$values>1e-6))^.5) %*%
  t(res_eigen_B$vectors)
# Get the root of B via an existing function
B_root2 = expm::sqrtm(B)
# Comparing
B_root1 - B_root2

## Generate a demo of positive definite matrices
C = A %*% t(A) # (almost surely) guaranteed to be positive definite
# Get the inverse of C via the eigen-decomposition of B
res_eigen_C = eigen(C)
C_inv1 = res_eigen_C$vectors %*%
  diag(res_eigen_C$values^-1) %*%
  t(res_eigen_C$vectors)
# Get the inverse of C via an existing function
C_inv2 = solve(C)
# Comparing
C_inv1 - C_inv2
```

Determinant and trace

- Merely applicable to square matrices
- Properties for determinant
  - $\det(\mathbf{A}) = \prod_i \lambda_i$
  - $\det(\mathbf{A}^\top) = \det(\mathbf{A})$
  - $\det(\mathbf{A}^{-1}) = 1/\det(\mathbf{A})$
  - $\det(c \cdot \mathbf{A}) = c^n \det(\mathbf{A})$ for $n \times n$ matrix $\mathbf{A}$ and scalar c
  - $\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$ if $\mathbf{A}$ and $\mathbf{B}$ are square matrices of the identical dimension
- Properties for trace
  - $\mathrm{tr}(\mathbf{A}) = \sum_i \lambda_i$
  - $\mathrm{tr}(c \cdot \mathbf{A}) = c \cdot \mathrm{tr}(\mathbf{A})$ for scalar c
  - $\mathrm{tr}(\mathbf{A} + \mathbf{B}) = \mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B})$ if $\mathbf{A}$ and $\mathbf{B}$ are square matrices of the identical dimension
  - (Trace trick) $\mathrm{tr}(\mathbf{A}_1 \cdots \mathbf{A}_k) = \mathrm{tr}(\mathbf{A}_{k'+1} \cdots \mathbf{A}_k \mathbf{A}_1 \cdots \mathbf{A}_{k'})$ for $1 < k' < k$.
    - Specifically, $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$
- Remark: $\det(\mathbf{A})$ and $\mathrm{tr}(\mathbf{A})$ can be taken as measures of the size of $\mathbf{A}$ when $\mathbf{A} > 0$

Block/partitioned matrix

- A partition of matrix: Suppose $\mathbf{A}_{11}$ is of $p \times r$, $\mathbf{A}_{12}$ is of $p \times s$, $\mathbf{A}_{21}$ is of $q \times r$ and $\mathbf{A}_{22}$ is of $q \times s$. Make a new $(p+q) \times (r+s)$-matrix by organizing $\mathbf{A}_{ij}$'s in a 2 by 2 way:

$$\mathbf{A} = \left[\begin{array}{c:c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hdashline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right]$$

  e.g.,

$$\mathbf{A} = \left[\begin{array}{cc:c} 1 & 0 & 2 \\ 0 & 1 & 3 \\ \hdashline 4 & 5 & 6 \end{array}\right]$$

if

$$\mathbf{A}_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{A}_{12} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \mathbf{A}_{21} = \begin{bmatrix} 4 & 5 \end{bmatrix}, \quad \text{and} \quad \mathbf{A}_{22} = \begin{bmatrix} 6 \end{bmatrix}.$$

- Operations with block matrices
  - Working with partitioned matrices just like ordinary matrices
  - Matrix addition: if dimensions of $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ are quite the same, then

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} + \mathbf{B}_{11} & \mathbf{A}_{12} + \mathbf{B}_{12} \\ \mathbf{A}_{21} + \mathbf{B}_{21} & \mathbf{A}_{22} + \mathbf{B}_{22} \end{bmatrix}$$

  - Matrix multiplication: if $\mathbf{A}_{ij}\mathbf{B}_{jk}$ makes sense for each i, j, k, then

$$\mathbf{A}\mathbf{B} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} \end{bmatrix}$$

  - Inverse: if $\mathbf{A}$, $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ are all invertible, then

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_{11.2}^{-1} & -\mathbf{A}_{11.2}^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21}\mathbf{A}_{11.2}^{-1} & \mathbf{A}_{22.1}^{-1} \end{bmatrix}$$

    * $\mathbf{A}_{11.2} = \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}$
    * $\mathbf{A}_{22.1} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$

```r
options(digits = 4)
set.seed(1)
## Generate an (almost surely) invertible matrix
(A = matrix(runif(9), nrow = 3, ncol = 3)) #

# Verify the inverse of partition matrix
## Method 1: following the above formula
(A11 = A[1:2, 1:2])
(A12 = matrix(A[1:2, 3], nrow = 2, ncol = 1))
(A21 = matrix(A[3, 1:2], nrow = 1, ncol = 2))
(A22 = matrix(A[3, 3], nrow = 1, ncol = 1))
(A11.2 = A11 - A12 %*% solve(A22) %*% A21)
(A22.1 = A22 - A21 %*% solve(A11) %*% A12)

(Ainv1 = rbind(
  cbind(solve(A11.2), -solve(A11.2) %*% A12 %*% solve(A22)),
  cbind(-solve(A22) %*% A21 %*% solve(A11.2), solve(A22.1))
))

## Method 2: solve()
Ainv2 = solve(A)

## Comparison
Ainv2 - Ainv1
```

An example utilizing matrix basics: rephrasing the ridge estimator