STAT 7650: Computational Statistics

2. Numerical Linear Algebra

Peng Zeng

Department of Mathematics and Statistics
Auburn University

Spring 2026

Outline

1. Numerical linear algebra

References: Boyd and Vandenberghe (2004). Appendix C.

Linear Regression

Let $\{(y_i, x_i), y_i \in \mathbb{R}, x_i \in \mathbb{R}^p, i = 1, \ldots, n\}$ be an iid sample from

$$y_i = \beta^T x_i + \varepsilon_i, \quad i = 1, \ldots, n,$$

where $\beta \in \mathbb{R}^p$ is a vector of parameters. The least squares estimate (LSE) of β is

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2.$$

If we assume $\varepsilon_i \sim^{iid} N(0, \sigma^2)$, the LSE of β is exactly the MLE.

Derivatives

- Let $y = f(x) \in \mathbb{R}$ and $x = (x_k) \in \mathbb{R}^p$, then

$$\frac{\partial y}{\partial x} = \left(\frac{\partial y}{\partial x_k}\right)_{p\times 1} \qquad \frac{\partial y}{\partial x^T} = \left(\frac{\partial y}{\partial x_k}\right)_{1\times p}$$

- Let $y = f(x) \in \mathbb{R}^m$ and $x \in \mathbb{R}^p$, then

$$\frac{\partial y}{\partial x^T} = \left(\frac{\partial y_i}{\partial x_k}\right)_{m\times p}$$

- (Chain rule). Let $z = f(y) \in \mathbb{R}$, $y = g(x) \in \mathbb{R}^m$, $x \in \mathbb{R}^p$,

$$\frac{\partial z}{\partial x} = \left(\sum_i \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x_k}\right)_{p\times 1} = \left(\frac{\partial y}{\partial x^T}\right)^T \frac{\partial z}{\partial y}$$

Least Squares Estimate

The objective function for the least squares problem is

$$f(\beta) = (y - X\beta)^T (y - X\beta)$$

where $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n\times p}$, and $\beta \in \mathbb{R}^p$. Then

$$\frac{\partial f(\beta)}{\partial \beta} = -2X^T(y - X\beta)$$

and if $X^T X$ is invertible,

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

Some formulas,

- $\partial(x^T A x)/\partial x = 2Ax$ for a symmetric matrix A.
- $\partial(a^T x)/\partial x = a$ for a vector a.
- $\partial(Bx)/\partial x^T = B$ for a general matrix B

Solving Linear Equations

Consider the problem of solving the set of linear equations,

$$Ax = b,$$

where the coefficient matrix $A \in \mathbb{R}^{n\times n}$ and the righthand side $b \in \mathbb{R}^n$. When A is nonsingular, the solution x is unique and

$$x = A^{-1}b.$$

The standard generic method has a computing complexity $O(n^3)$, which can be reduced greatly for matrices of special structure, such as symmetry, sparseness.

```
solve(A, b)              # better
solve(A) %*% b
```

Flop Count

The cost of an algorithm is measured by the total number of flops (floating-point operations).

- A flop means one addition, subtraction, multiplication, or division of two floating-point numbers.
- Focus on the leading term. For example, the total number of flops of a particular algorithm is

$$m^3 + 3m^2n + mn + 4mn^2 + 5m + 22$$

where m and n are problem dimensions, we simplify it to

$$O(m^3 + 3m^2n + 4mn^2)$$

If $m \ll n$, we may say $O(4mn^2)$ or even $O(mn^2)$.

Basic Matrix-Vector Operations

- vector operations: for $x, y \in \mathbb{R}^n$, $a \in \mathbb{R}$,
  - $x^T y$: $2n-1$ flops (n multiplies and $n-1$ additions)
  - ax: n flops (n multiplies)
  - $x+y$: n flops (n additions)
- matrix-vector multiplication: $A \in \mathbb{R}^{m\times n}$
  - Ax: $m(2n-1)$ flops.
  - If A is diagonal: n flops
- matrix-matrix multiplication: $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$
  - AB: $mp(2n-1)$ flops.
- Two different ways of computing $D = ABC$, where $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{p\times q}$,
  - $D = (AB)C$: $2mnp + 2mpq = 2mp(n+q)$ flops
  - $D = A(BC)$: $2npq + 2mnq = 2nq(m+p)$ flops
  - The first one is better when $mp(n+q) < nq(m+p)$.

BLAS

BLAS (Basic Linear Algebra Subprograms) is a collection of routines for vector and matrix operations. It is the computing engine of modern scientific software such as R, python, and MatLab.

Level 1, $O(n)$	dSCAL	$y = ax$
	dCOPY	$y = x$
	dAXPY	$y = ax + y$
	dDOT	$x^T y$
	dNRM2	$\|x\|_2$
Level 2, $O(n^2)$	dGEMV	$y = aAx + by$
	dSYR	$A = axx^T + A$
	dSYR2	$A = axy^T + ayx^T + A$
Level 3, $O(n^3)$	dGEMM	$C = aAB + bC$
	dSYRK	$C = aAA^T + bC$
	dSYRK2	$C = aAB^T + aBA^T + bC$

Solving Linear Equations for Special Matrices

Compute $x = A^{-1}b$ for special matrices.

- Let A be diagonal. Then $x_i = b_i/a_{ii}$ using n flops.
- Let A be orthogonal. Then $x = A^{-1}b = A^T b$ using $2n^2$ flops.
- Let A be a permutation matrix,

$$A_{ij} = 1, \text{ if } j = \pi_i, \quad A_{ij} = 0, \text{ otherwise.}$$

  where $\pi = (\pi_1, \ldots, \pi_n)$ is a permutation of $(1, 2, \ldots, n)$. Then A is orthogonal. Hence x is obtained by permuting the entries of b by π^{-1}, which is the inverse permutation.

Solving Linear Equations for Special Matrices

- Let A be lower-triangular, $a_{ij} = 0$ for $j > i$. Apply the forward substitution,

$$\begin{aligned} x_1 &= b_1/a_{11} \\ x_2 &= (b_2 - a_{21}x_1)/a_{22} \\ &\dots \\ x_n &= (b_n - a_{n1}x_1 - \dots - a_{n,n-1}x_{n-1})/a_{nn} \end{aligned}$$

  The total number of flops is $1 + 3 + 5 + \cdots + (2n-1) = n^2$.

- Let A be upper-triangular. Apply the backward substitution, which has the same flops.

```
forwardsolve(L, b, transpose = F)      # solve(L) %*% b
backsolve(R, b, transpose = F)         # solve(R) %*% b
```

Factor-Solve Method

Assume A can be expressed as $A = A_1 A_2 \cdots A_k$ Then

$$x = A^{-1}b = A_k^{-1} A_{k-1}^{-1} \cdots A_1^{-1} b$$

We can compute the solution as

$$z_1 = A_1^{-1}b,\ z_2 = A_2^{-1}z_1,\ \ldots,\ z_{k-1} = A_{k-1}^{-1}z_{k-2},\ x = A_k^{-1}z_{k-1}$$

The total flop count is $f + s$, where

- f is the flops of the factorization step
- s is the flops of the solve step.

If we have multiple righthand sides, or compute $X = A^{-1}B$, where $B \in \mathbb{R}^{n\times m}$, the total flop count is $f + ms$.

LU Factorization

Every nonsingular matrix $A \in \mathbb{R}^{n\times n}$ can be factored as

$$A = PLU$$

where $P \in \mathbb{R}^{n\times n}$ is a permutation matrix, $L \in \mathbb{R}^{n\times n}$ is unit lower triangular, and $U \in \mathbb{R}^{n\times n}$ is upper triangular and nonsingular.

The LU factorization can also be represented as

$$P^T A = LU$$

where $P^T A$ is obtained from A by re-ordering its rows.

- The standard algorithm for computing an LU factorization is called Gaussian elimination with partial pivoting or Gaussian elimination with row pivoting.
- The cost is $(2/3)n^3$ flops.

Solving Linear Equations by LU

Solve linear equations $Ax = b$ by LU factorization, where A is nonsingular.

- LU factorization. Factor A as $A = PLU$. $(2/3)n^3$ flops
- Permutation. Solve $Pz_1 = b$. 0 flops.
- Forward substitution. Solve $Lz_2 = z_1$. n^2 flops
- Backward substitution. Solve $Ux = z_2$. n^2 flops

The total cost is $(2/3)n^3 + 2n^2 = (8/3)n^3$.

```
solve(A, b)   # apply LU decomposition with partial pivoting
```

- When b is a matrix with m columns, the cost is $(2/3)n^3 + 2mn^2$.
- Compute A^{-1} by setting $b = I_n$. This cost is $(8/3)n^3$.

Cholesky Factorization

If $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite, then the Cholesky factorization of A is

$$A = LL^T$$

where L is lower triangular and nonsingular with positive diagonal elements.

- The matrix L is uniquely determined by A.
- The cost is $(1/3)n^3$ flops.

```
chol(A)  # compute a upper-triangular matrix, t(R) %*% R = A
```

- If the components of Z are iid $N(0,1)$, then $X = \mu + LZ$ follows multivariate normal $N(\mu, LL^T)$.

Under-determined Linear Equations

Under-determined linear equations mean $A \in \mathbb{R}^{p \times n}$ with $p < n$.

$$Ax = b$$

Assume $\text{rank}(A) = p$, so there is at least one solution for all b. In many applications it is sufficient to find just one particular solution $\hat{x}$. All solutions form a set

$$\{x \mid Ax = b\} = \{Fz + \hat{x} \mid z \in \mathbb{R}^{n-p}\}$$

where F is a matrix whose columns form a basis for the nullspace of A, which implies that $AF = 0$.

Inverting a Nonsingular submatrix

If $A_1 \in \mathbb{R}^{p\times p}$ is a nonsingular submatrix of A,

$$Ax = \begin{pmatrix} A_1 & A_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = A_1 x_1 + A_2 x_2 = b$$

We can express x_1 as

$$x_1 = A_1^{-1}(b - A_2 x_2) = A_1^{-1} b - A_1^{-1} A_2 x_2$$

We may simply take $\hat{x}_2 = 0$ and $\hat{x}_1 = A_1^{-1} b$.

If the first p columns of A is not linearly independent, select a set of p columns of A that is independent, permute them to the front, and then apply the method described above. That is, find a permutation matrix P such that the first p columns of $\tilde{A} = AP$ are independent.

QR Factorization

If $A \in \mathbb{R}^{n\times p}$ with $p \le n$ and $\text{rank}(A) = p$, the QR factorization is

$$A = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix}$$

where $Q_1 \in \mathbb{R}^{n\times p}$ and $Q_2 \in \mathbb{R}^{n\times(n-p)}$ satisfying

$$Q_1^T Q_1 = I, \quad Q_2^T Q_2 = I, \quad Q_1^T Q_2 = 0$$

and $R \in \mathbb{R}^{p\times p}$ is upper triangular with nonzero diagonal elements. It can be calculated in $2p^2(n - p/3)$ flops.

```
Aqr = qr(A)          # QR-factorization
qr.Q(Aqr)            # find matrix Q
qr.R(Aqr)            # find matrix R
```

Under-determined Linear Equations via QR

Suppose the QR factorization of A^T is

$$A^T = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix}$$

we have the complete solution set as

$$\{x = \hat{x} + Q_2 z \mid z \in \mathbb{R}^{n-p}\}$$

where $\hat{x} = Q_1 R^{-T} b$. The QR factorization method is the most common method for solving under-determined equations.

```
qr.Q( qr(t(A)), complete = TRUE)
```

By default, `qr.Q()` only compute Q_1. Use option `complete = TRUE` if you want both Q_1 and Q_2.

More Applications of QR

For a $n \times p$ matrix X, let $X = QR$ be the QR-factorization.

- For over-determined linear equations, the LSE of β is

$$\hat{\beta} = (X^T X)^{-1} X^T y = R^{-1} Q^T y$$

```
qr.coef(qr(X), y)       # least squares estimate
```

```
lm.fit(X, y)            # least squares estimate
lm.wfit(X, y, w)        # weighted least squares
```

- The projection matrix is $P = X(X^T X)^{-1} X^T = QQ^T$.
- For a square matrix X, compute $X^{-1} b$ as $R^{-1} Q^T b$.

```
qr.solve(x, b)
```

Spectral Decomposition

For a symmetric matrix A, its spectral decomposition is

$$A = U\Lambda U^T = \sum_{i=1}^{n} \lambda_i u_i u_i^T,$$

where U is orthogonal and Λ is a diagonal matrix. The columns of A are eigenvectors and the diagonal elements of Λ are eigenvalues.

```
eigen(A)      # output list(values = , vectors = )
```

- Compute $A^{1/2}$, $A^{-1/2}$

Singular Value Decomposition

For a general matrix A, its singular value decomposition is

$$A = U\Lambda V^T$$

where U and V are orthogonal and Λ is a diagonal matrix with all positive elements on diagonal.

```
svd(A)   # output list(d = , u = , v = )
```

The polar decomposition of a matrix $A \in \mathbb{R}^{n\times p}$ is

$$A = HZ, \quad H = A(A^TA)^{-1/2} = UV^T, \quad Z = (A^TA)^{1/2} = V\Lambda V^T$$

where H is a column-orthogonal matrix, representing orientation and Z is positive-definite, representing scaling.

Lapack

Lapack (Linear Algebra Package) is a collection of routines for solving systems of linear equations, linear least squares, eigenvalue problems, and singular value decomposition. It also includes routines for many matrix factorizations.

dTRSM	back solve (level-3 BLAS)
dPOTRF	Cholesky decomposition
dGEQP3	QR decomposition
dSYEVR	eigenvalues and eigenvectors
dGESDD	singular value decomposition