STAT243 Lecture 10.1 Preliminaries

Logic

许多统计学与机器学习方法都依赖于线性代数操作，例如矩阵乘法或矩阵分解（decomposition）。这些运算构成了模型拟合与分析的基础。常见的应用包括:

- 线性回归与广义回归模型
- 深度神经网络（矩阵乘法是反向传播的核心）
- 主成分分析 (PCA) 及其推广形式（如核 PCA、稀疏 PCA 等）
- 各类矩阵分解（SVD、QR、Cholesky 等）

1 Key Principle 关键原则

数学表达式的形式与其在计算机上的实现方式可能完全不同。
良好的计算策略既能加快运算，也能改善数值稳定性。

1.1 Example 1

若 X、Y 为矩阵，z 为向量，计算 $X(Yz)$ 比 $(XY)z$ 更高效。

- $(XY)z$ 需要先计算整个矩阵乘法
- 而 $X(Yz)$ 只需先计算 Yz（得到一个向量），再进行矩阵向量乘

1.2 Example 2

回归估计式 $(X^\top X)^{-1}X^\top Y$

- 实际实现时并不直接计算 $X^\top X$ 或其逆矩阵
- 许多算法（如 QR 或 SVD 分解）完全避免了求逆操作

1.3 Example 3

若要交换矩阵 A 的两行，可引入置换矩阵 P:

- 理论上 PAB 可实现行交换
- 在实际计算中，很多实现仅改变索引，而无需重新存储矩阵元素

---

2 Computational Complexity

2.1 基本定义 (Definition)

计算复杂度通过计数加法/减法与乘法/除法操作量来衡量算法效率。
常使用 **大 O 符号** $O(f(n))$ 表示计算量的增长阶：

$$\#\text{operations} \propto f(n)$$

- $O(n^3)$：多项式时间（常见于矩阵运算）
- $O(b^n)$：指数时间（极慢）
- $O(\log n)$：对数时间（极快）

> ⚠ Remark

加法比乘法稍快, 因此很多算法试图用加法替代乘法

2.2 示例：矩阵乘法

2.2.1 运算速度

若 A 为 $a \times b$，B 为 $b \times c$，则

- 每个输出元素需进行 b 次乘法
- 一共 abc 次乘法，因此复杂度为 $O(abc)$

对称 $n \times n$ 矩阵乘法为 $O(n^3)$
同样地，Cholesky 或 QR 分解也为 $O(n^3)$（若矩阵稀疏则可更快）

2.2.2 存储与内存需求

乘法结果需保存 $ab + bc + ac$ 个元素
对于对称矩阵，近似 $3n^2$ 个元素

例：当 $n = 10{,}000$ 且每个元素为 8 字节浮点数时：

$$3 \times 10{,}000^2 \times 8/10^9 = 2.4 \text{ GB}$$

因此大规模矩阵计算既在时间上 $O(n^3)$，又在空间上 $O(n^2)$。

2.3 多项式、指数与对数时间的意义

- **多项式时间** $O(n^q)$：可接受（如矩阵分解）
- **指数时间** $O(b^n)$：不可扩展（如 NP-complete 问题）
- **对数时间** $O(\log n)$：极高效（如二分查找）

NP-complete 问题是无法在多项式时间内求解的难题族。

> ⚠ Remark

即使复杂度阶相同，常数项也可能决定算法快慢：

- 例：n^2 操作可能比 $1000(n \log n + n)$ 更快
- 因为第二个算法中常数 $c = 1000$ 较大，且存在额外的低阶计算

2.4 FLOPs (Floating Point Operations)

- “flops” 可指：
  1. **浮点运算次数**
  2. **每秒浮点运算数 (floating point operations per second)**
- 常用于衡量算法计算量或计算机性能

---

3 Notation and Dimensions

3.1 Notations

符号	含义
A	矩阵 (matrix)
x	向量 (vector)
x_i	向量第 i 个元素
A_{ij}	矩阵第 i 行、第 j 列元素
$A_{\cdot j}$	第 j 列
$A_{i\cdot}$	第 i 行

默认：

- 向量 x 为列向量
- $x^\top$ 为行向量

3.2 Dimensions

检查矩阵 **是否可相乘（conformable）** 是编程中常见的防错手段：

- 对于 $A+B$，要求维度完全相同
- 对于 AB，要求 A 的列数 = B 的行数

例：

$$\mathrm{Cov}(Ax) = A\mathrm{Cov}(x)A^\top$$

而非 $A^\top\mathrm{Cov}(x)A$，否则维度不匹配。

3.3 Inner and Outer Products

- **内积 (Inner Product)**

$$\langle x, y\rangle = x^\top y = \sum_i x_i y_i$$

- **外积 (Outer Product)**

$$xy^\top$$

结果为矩阵，包含 x 与 y 所有元素的乘积组合。

4 Norms

4.1 向量范数 (Vector Norm)

$$|x|_p = \left(\sum_i |x_i|^p \right)^{1/p}$$

常用的欧几里得范数：

$$|x|_2 = \sqrt{x^\top x}$$

4.2 矩阵范数 (Matrix Norms)

- **Frobenius 范数**

$$||A||_F = \sqrt{\sum_{i,j} a_{ij}^2}$$

- **诱导范数 (Induced Norm)**

$$||A|| = \sup_{x \neq 0} \frac{||Ax||}{||x||}$$

  对于 2-norm：

$$||A||_2 = \sup_{|x|_2 = 1} ||Ax||_2$$

  等价于矩阵 **最大奇异值** (largest singular value)

4.3 范数的几何意义与性质

- $||A||$ 衡量矩阵对向量的最大拉伸倍数
- 基本性质：
  - 次乘法性：$||AB|| \leq ||A||||B||$
  - 三角不等式：$||A + B|| \leq ||A|| + ||B||$
- 向量单位化（normalized）：

$$\tilde{x} = \frac{x}{||x||}$$

- 向量夹角：

$$\theta = \cos^{-1} \left(\frac{\langle x, y \rangle}{\sqrt{\langle x, x \rangle \langle y, y \rangle}} \right)$$

---

5 Orthogonality

5.1 向量正交 (Orthogonal Vectors)

若 $x^\top y = 0$，则称 x 与 y **正交**（orthogonal），记作 $x \perp y$。

5.2 正交矩阵 (Orthogonal Matrix)

一个方阵 A 称为正交矩阵若：

$$A^\top A = I \quad \text{或等价地} \quad A^{-1} = A^\top.$$

正交矩阵的主要性质：

1. **列与行互相正交且单位化**
   $A_{\cdot i}^\top A_{\cdot j} = 0$（当 $i \neq j$）且 $|A_{\cdot i}| = 1$。
2. **满秩（full rank）**
   因为列向量线性无关。
3. **行列式 (determinant)**
   $\det(A) = \pm 1$。
4. **乘积封闭性**
   若 A 与 B 均为正交矩阵，则

   $$(AB)^\top AB = B^\top A^\top AB = B^\top B = I$$

   因此 AB 也是正交的。

5.3 置换矩阵 (Permutation Matrices)

- **定义**：交换矩阵中两行或两列的单位矩阵称为**基本置换矩阵 (elementary permutation matrix)**。
  这类矩阵是正交的，且 $\det(P) = -1$。
- **性质**：
  - **行置换**：左乘 P，即 PA。
  - **列置换**：右乘 P，即 AP。
  - 计算上不必显式乘法，只需调整索引即可（节省计算与内存）。

---

6 Vector and Matrix Properties

6.1 基本代数性质

性质	表达式
非交换性	$AB \neq BA$
可交换性（加法）	$A + B = B + A$
结合律	$A(BC) = (AB)C$

6.2 Python 操作回顾

Python

```python
A + B               # 矩阵加法
np.matmul(A, B)     # 矩阵乘法
A @ B               # 推荐的简写
A * B               # Hadamard（逐元素）乘法
A.dot(B)            # 不推荐（NumPy 已弱化）
```

---

7 Trace and Determinant

7.1 矩阵迹 (Trace)

定义：

$$\mathrm{tr}(A) = \sum_i A_{ii}$$

常见性质：

1. $\mathrm{tr}(A+B) = \mathrm{tr}(A) + \mathrm{tr}(B)$
2. $\mathrm{tr}(A^\top) = \mathrm{tr}(A)$
3. **循环不变性**：

$$\mathrm{tr}(ABC) = \mathrm{tr}(BCA) = \mathrm{tr}(CAB)$$

应用意义：

- 可以通过变换矩阵顺序减少计算量。
- 可将标量二次型转化为矩阵迹形式：

$$x^\top Ax = \mathrm{tr}(x^\top Ax) = \mathrm{tr}(xx^\top A)$$

7.2 矩阵行列式 (Determinant)

性质：

1. $|AB| = |A||B|$
2. $|A^{-1}| = 1/|A|$
3. $|A| = |A^\top|$ （由 QR 分解与三角矩阵性质可知）

---

8 Transposes and Inverses

8.1 转置与逆的关系

若 A 可逆，则：

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

Proof

$$A^\top (A^{-1})^\top = (A^{-1}A)^\top = I$$

8.2 矩阵乘法的逆

对于可逆矩阵 A, B：

$$(AB)^{-1} = B^{-1}A^{-1}$$

验证：

$$B^{-1}A^{-1}AB = I$$

8.3 Other Matrix Products

8.3.1 Hadamard Product（逐元素乘法）

$$(A \circ B)_{ij} = A_{ij}B_{ij}$$

Python 中对应 `A * B`。

Challenge: 如何不计算 `A @ B` 直接求 $\mathrm{tr}(AB)$?

解答：

$$\mathrm{tr}(AB) = \sum_{i,j} A_{ij}B_{ji}$$

在 Python 中：

Python

```python
np.sum(A * B.T)
```

8.3.2 Kronecker Product（克罗内克积）

令矩阵 $\mathbf{A} = (a_{i,j}) \in \mathbb{R}^{m\times n}$, $\mathbf{B} = (b_{i,j}) \in \mathbb{R}^{m\times n}$

则**克罗内积**为:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \cdots & a_{1,n}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \cdots & a_{2,n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1}\mathbf{B} & a_{m,2}\mathbf{B} & \cdots & a_{m,n}\mathbf{B} \end{bmatrix}$$

性质：

1. $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$
2. 计算复杂度显著降低：
   - 直接求逆：$O((nm)^3)$
   - 利用结构：$O(n^3 + m^3)$

---

9 Matrix Decompositions

矩阵分解是将一个复杂矩阵 A 表示为若干更“简单”的矩阵乘积：

$$A = BC \quad 或 \quad A = BCD$$

9.1 常见“简化”形式

- 减少维度：部分行或列
- 特殊结构：对角矩阵、上/下三角矩阵、稀疏矩阵
- 正交矩阵：数值稳定性高

9.2 优势

- 揭示矩阵内部结构（如方向、方差、相关性）
- 简化计算，例如：
  - **线性系统求解**
  - **回归估计**
  - **特征值/奇异值分解**