STAT243 Lecture 10.6 Computation

1 Linear algebra in Python

> **Logic**
>
> 在许多矩阵分解中，你可以选择只返回部分结果而非全部结果，从而显著提升计算速度。

> **Logic**
>
> 在线性代数是统计与机器学习算法核心的背景下，**大型问题的效率在很大程度上依赖于是否调用了快速的线性代数后端**。这些后端可能同时利用并行 CPU 或 GPU，从而带来数量级的加速。

1.1 BLAS and LAPACK

Python 和 R 的矩阵运算大多直接调用底层 C 或 Fortran 实现，并不会经过大量解释层的开销。因此速度可以非常快。这些核心线性代数库主要包括：

- **BLAS** (Basic Linear Algebra Subprograms)
- **LAPACK** (Linear Algebra PACKage)

不同系统可选择不同 BLAS 后端，例如：

- OpenBLAS
- Intel MKL
- AMD ACML
- Apple vecLib（在 Apple Silicon 上利用 AMX 协处理器）

在 Conda 环境中，numpy 通常会被构建为使用 MKL 或 OpenBLAS；pip 安装通常使用 OpenBLAS。

BLAS 例程分为三个层级：

- **Level 1**：向量操作
- **Level 2**：矩阵–向量操作
- **Level 3**：矩阵–矩阵操作（最高效、并行化最强）

LAPACK 在 BLAS 之上构建，提供：

- 特征分解
- 线性系统求解
- 多种矩阵分解（QR、Cholesky、SVD 等）

1.2 JAX and PyTorch (and GPUs)

JAX 与 PyTorch 提供了替代性的线性代数后端，能够自动检测 CPU 或 GPU。

- JAX 支持 **just-in-time compilation (JIT)**
- PyTorch 和 JAX 均在 GPU 上具备高度并行的矩阵运算能力

对于连续大规模矩阵运算，例如矩阵乘法，它们在 GPU 上的速度可能是 NumPy 的数十倍。

相关示例可参考 SCF parallelization tutorial 中的 CPU/GPU 对比演示

---

2 Exploiting known structure in matrices

如果矩阵具有特定结构，则可以通过使用特定格式获得显著速度与存储收益。例子包括：

- 对称矩阵（仅存储半个三角）
- Banded matrices（只有若干条带非零）
- Block diagonal matrices（分块对角）
- 稀疏矩阵（大部分为零）

在 Python 中，可使用 `scipy.sparse` 模块，它支持：

- 结构化稀疏矩阵（如 diagonal、triangular）
- 非结构化稀疏矩阵（CSR、CSC 等格式）

在 R 中对应的工具包括 `Matrix`、`spam` 和 `bdsmatrix`。

2.1 Sparse matrix formats

以下示例展示如何通过 CSR (compressed sparse row) 格式高效存储稀疏矩阵：

Python

```python
import scipy.sparse as sparse
mat = np.array([[0,0,1,0,10],
                [0,0,0,100,0],
                [0,0,0,0,0],
                [1000,0,0,0,0]])
mat = sparse.csr_array(mat)

mat.data     # non-zero entries
# array([ 1, 10, 100, 1000])

mat.indices  # column indices
# array([2, 4, 3, 0], dtype=int32)

mat.indptr   # row pointers
# array([0, 2, 3, 3, 4], dtype=int32)
```

CSR 格式包含三个数组：

- `data`：按行排列的非零元素
- `indices`：每个非零元素的列索引
- `indptr`：每一行起始元素的位置指针

> ⚠ **Remark**
>
> `indptr[i]` = 第 i 行的非零元素在 `data` 里的开始位置
> `indptr[i+1]` = 第 i 行的非零元素在 `data` 里的结束位置

可根据以上三部分直接恢复矩阵：

Python

```
mat2 = sparse.csr_array((mat.data, mat.indices, mat.indptr))
mat2.toarray()
# array([[ 0,   0,  1,  0,      10],
#        [ 0,   0,  0,  100,    0],
#        [ 0,   0,  0,  0,      0],
#        [1000, 0,  0,  0,      0]])
```

2.2 Sparse matrix multiplication

CSR 结构可以将矩阵乘法 $x = Ab$ 的计算量降至 $O(k)$，其中 k 为非零元素个数：

```
for(i in 1:nrows(A)){
    x[i] = 0
    for(j in rowpointers[i] : rowpointers[i+1]-1){
        x[i] = x[i] + entries[j] * b[colindices[j]]
    }
}
```

对比 dense 乘法的 $O(n^2)$ 耗时，提升显著。

稀疏矩阵 Cholesky 分解时，如果稀疏结构固定，可以预先做 fill–reducing reordering（如 AMD, nested dissection），保持稀疏性。

---

3 Banded matrices

若矩阵具有带状结构：

- 下带宽 p：$A_{ij} = 0$ 当 $i > j + p$
- 上带宽 q：$A_{ij} = 0$ 当 $j > i + q$

则 LU 分解的成本为 $O(npq)$。相比 dense LU 的 $O(n^3)$，速度提升巨大。

时间序列模型中（如 MA 模型），协方差矩阵常呈带状结构，因此非常适用。

---

4 Low rank updates (optional)

对于 rank-one 更新：

$$\tilde{A} = A - uv^\top$$

更一般的低秩更新：

$$\tilde{A} = A - UV^\top$$

其中 $U, V \in \mathbb{R}^{n \times m}$。

著名结果是 **Sherman–Morrison–Woodbury formula**：

$$\tilde{A}^{-1} = A^{-1} - A^{-1}U(I_m - V^\top A^{-1}U)^{-1}V^\top A^{-1}$$

或者等价形式：

$$\tilde{A} = A + UCV^\top$$

则：

$$\tilde{A}^{-1} = A^{-1} - A^{-1}U(C^{-1} + V^\top A^{-1}U)^{-1}V^\top A^{-1}$$

意义：

- 若 m 很小（低秩更新），则可避免重新求整个 A^{-1}
- 在贝叶斯计算、Gaussian Markov random fields、精度矩阵更新等领域非常常见