

STAT243 Lecture 7.4 Recent Tools and Data Storage Formats

1 高性能数据存储工具 (Recent Tools and Data Formats)

- 近年来，出现了大量新的文件格式与工具，用于高效处理超大规模数据集。
- 相比传统数据库系统，这些工具常以 **速度与并行处理性能** 为优化目标。

1.1 数据湖 (Data Lake)

- 大型数据往往不再存储于单一数据库中，而是分散为多个文件，常使用：
 - Parquet**（面向列的高效存储格式）
 - CSV**（传统文本格式）
 - 当这些文件存储于云端（如 AWS S3、Google Cloud Storage），整体结构被称为 **数据湖**。
-

1.2 Apache Arrow

1.2.1 核心特点

- 提供高效的内存数据结构，用于跨语言、跨系统的数据分析。
- 在 Python 中可通过 **PyArrow** 包使用（R 中为 **arrow** 包）。
- 采用 **列式存储**：同一列的值顺序存储，可实现 **O(1)** 时间复杂度的单值访问。

1.2.2 优势

- 能够从多种文件格式读取数据（如 Parquet、Arrow 原生格式、文本文件）。
- 仅在需要时才从磁盘读取数据，避免将整个数据集加载至内存。
- 通过按需读取与内存映射（memory mapping）显著减少 I/O 开销。

1.2.3 与其他格式的比较

- 原生 Arrow 格式在按列访问与懒加载方面性能最佳。
 - Parquet 在磁盘压缩与云端共享方面更具优势。
-

1.3 Polars

1.3.1 特点

- 一款 **内存中超高速 DataFrame 库**，是 Pandas 的替代方案。
 - 基于 **Apache Arrow** 的列式数据结构。
 - 支持 **惰性执行 (lazy execution)** 模型，类似 Spark 或 Dask，可自动优化查询计划。
 - 常用于需要高效分组、聚合与 join 操作的分析场景。
-

2 稀疏性 (Sparsity)

2.1 概念

- 在统计与机器学习中，许多矩阵包含大量零元素。
- 稀疏矩阵 (Sparse Matrix) 仅存储非零元素，从而减少存储空间与计算时间。

2.2 常见应用

- **图与网络分析**: 邻接矩阵或条件依赖矩阵
- **时间序列与空间模型**: 自回归 (AR) 或空间邻接矩阵
- **高维回归模型**: LASSO 等方法产生稀疏参数估计
- **协方差选择模型**: 稀疏精度矩阵估计

2.3 软件支持

- Python、R、MATLAB 均提供稀疏矩阵的专用数据结构与计算函数。
- Python 中常用模块:
 - `scipy.sparse`
 - `sklearn.utils.sparsefuncs`
- 稀疏矩阵的存储方式包括:
 - COO (坐标形式)
 - CSR (压缩行格式)
 - CSC (压缩列格式)

3 利用统计思想解决计算瓶颈 (Statistical Tools for Computational Bottlenecks)

3.1 采样思想 (Sampling for Computation)

- **传统动机**: 无法收集全体数据。
- **大数据场景**: 无法处理全部数据, 可对数据进行采样分析。
- 使用统计不确定性估计 (e.g., 置信区间) 评估采样误差。
- 若结果精度不够, 可通过增大样本量改进。

3.2 分布式自助法 (Distributed Bootstrap)

- 提出者: Michael Jordan 等人 (Berkeley Statistics/EECS)。
- 方法:
 1. 从大数据中抽取多个较小的自助样本 (bootstrap samples)。
 2. 在每个样本上独立计算估计量。
 3. 综合结果以获得近似全数据估计。
- 优点: 计算可完全并行化, 内存占用显著减少。
- 参考: Kleiner et al., *Journal of the Royal Statistical Society*, 2014.

3.3 随机化算法 (Randomized Algorithms)

- 基于随机选择的计算加速策略。
- 应用示例:
 - **随机梯度下降 (SGD)**: 优化时仅使用部分样本。
 - **随机线性代数**: 在回归中随机选择设计矩阵的部分行。
 - **统计杠杆值 (Leverage Scores)**: 用以加权采样高影响点。
- 优点:
 - 降低计算复杂度。
 - 提高算法可扩展性与鲁棒性。
- 参考文献: 相关算法综述可见于 *arXiv: randomized numerical linear algebra* 系列论文。

3.4 小结

- **Apache Arrow** 与 **Polars** 是现代高性能数据处理的重要基础。
- **稀疏矩阵结构** 能有效减少大规模计算的空间与时间复杂度。
- **统计抽样与随机化算法** 提供应对计算瓶颈的理论框架。
- 结合这些思想，可在不牺牲统计精度的前提下显著提升大数据分析的可行性与效率。