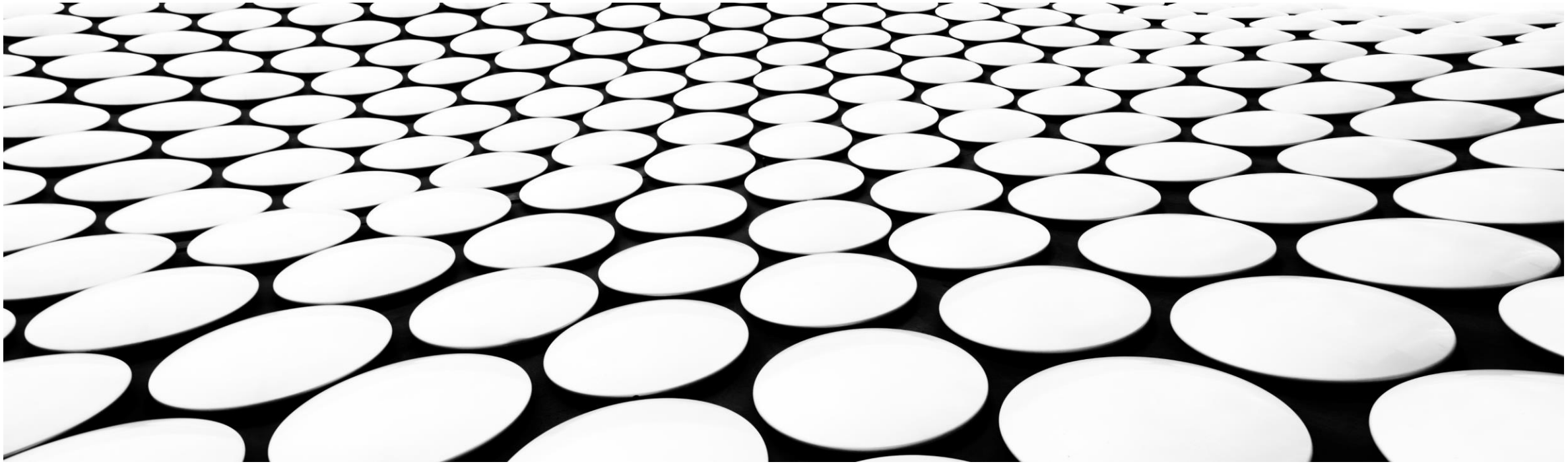
分布式计算

邱怡轩



今天的主题

- PySpark基础
- 数值计算基础

PySpark 基础

数据结构

- 在之前的课程中，我们曾演示用 PySpark 进行简单的数据处理
- 事实上，其操作与基于迭代器的函数式编程非常相近
- PySpark 中支持该类运算的结构叫做 RDD（Resilient Distributed Dataset）

RDD

- RDD 是 Spark/PySpark 中的核心底层数据结构
- 与迭代器类似，RDD 可以进行 Map、Filter、Reduce 和收集等常见操作
- 但 RDD 还有一些独特的性质

RDD

- RDD 具有容错机制（Resilient）
- RDD 是分布式的（Distributed）
- 与迭代器不同，RDD 可以重复进行使用

基本操作

- lec5-pyspark-rdd.ipynb

数据结构

- 后续牵涉到统计模型时主要进行矩阵和向量的操作
- 我们使用 Numpy 模块提供的数据结构
- `lec5-numpy.ipynb`

数值计算基础

计算原则

- 核心准则1：多个矩阵相乘（向量也视为矩阵），优先进行维度较小的操作

计算原则

- 原理：矩阵 $A_{m\times n}$ 与 $B_{n\times p}$ 相乘，计算复杂度为_____

计算原则

- 核心准则2：不到万不得已，不要算矩阵的逆！不要算矩阵的逆！不要算矩阵的逆！

计算原则

- 原理：很多牵涉到矩阵求逆的运算，其实可以归结为解线性方程组 $x = A^{-1}b \Rightarrow Ax = b$
- 解 $Ax = b$ 要比先算 A^{-1} 再乘以 b 高效得多

计算原则

- 核心准则3：尽可能利用矩阵的特殊结构来减少计算量

计算原则

- 例如计算 WA，其中 $A_{n\times p}$，$W_{n\times n}$ 是一个对角矩阵
- WA 相当于将 W 的对角线元素乘到 A 的每一列
- 避免了直接的矩阵乘法

计算原则

- 类似地，如果需要计算 AW，其中 $A_{n\times p}$，$W_{p\times p}$ 是一个对角矩阵
- 只需将 W 的对角线元素乘到 A 的每一行

计算原则

- 核心准则4：尽可能将多个向量运算的循环合并为矩阵运算

- $(Ax_1, Ax_2, \ldots, Ax_p) \Rightarrow AX$，其中 $X = (x_1, x_2, \ldots, x_p)$
- $A \in \mathbf{R}^{m \times n}$，$x_i \in \mathbf{R}^n$，$X \in \mathbf{R}^{n \times p}$

计算原则

- 原理：虽然理论上的计算复杂度相同，但软件层面对矩阵间的运算往往优化更好

效率对比

- lec5-numerical.ipynb

练习

- 回归分析中给出了模型的预测公式

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

- 假设 $X_{n\times p}$，$n > p$，应如何编写程序进行计算?