上海交通大学 2023 ~ 2024 学年度 应用统计公众号文章项目

软件运行效率对比

2024 - 01 - 17 编制

编者院系: **数学科学学院**　　指导教师: **王 成**　　共同完成者: **李昌骏　刘若禹　邵毅诚**

(共同完成者依照姓氏拼音字母顺序排序)

前　言

本项目受到 2023 学年秋季开设的《统计软件与算法》课程的启发, 对于 R 语言当中可采用的加速库 BLAS, OpenBLAS, 以及 Python, Matlab 语言, 对于多个数学, 统计领域相关运算任务进行软件运行效率的对比.

第一部分　引　例

在当今信息时代, 数据分析已经成为各行各业中不可或缺的一部分. 在金融, 互联网, 生物医学, 市场营销等众多研究领域, 数据分析为决策提供了可视化的数据支撑材料. 然而, 在庞大的数据海洋中航行, 并从中提取有用的信息涉及到诸多复杂的统计主题.

在本文中, 我们将基于三大类统计任务: 基础矩阵运算和数据处理, 高级矩阵运算和分解, 复杂数学计算和特征提取. 这几类任务代表了实际应用中常见的数据分析场景, 涉及到不同层次和类型的统计操作, 例如: 大数据生成, 回归分析, 主成分分析等. 生成大数据考察了处理大规模数据的能力, 回归分析关注了建模和预测的技术, 而高维数据主成分分析则挑战了对高维数据的降维和特征提取.

为了对比不同编程语言在这些任务上的表现, 我们选择了 R 语言, Python 和 Matlab. 这三种语言在数据科学和统计分析领域具有广泛的应用, 并代表了不同的编程范式和生态系统. 通过对它们在各个任务上所花费的时间进行比较, 我们将能够评估它们在不同统计主题下的性能表现.

此外, 我们在比较过程中将适当地采用切换底层库实现运算的加速方法. 进而, 有助于深入地了解每种语言的性能瓶颈和优化空间, 以更全面地评估它们在实际任务中的可行性.

最终的实验结果表明: Matlab 在底层逻辑库方面呈现出的相对明显的优势. 特别是在矩阵运算和并行计算方面, Matlab 展现出卓越的性能, 使得其在处理大规模数据时表现出色, 运算更加高效.

第二部分　BLAS 简介

BLAS(Basic Linear Algebra Subprograms)是一组用于执行基本线性代数运算的软件库. BLAS 不仅包括矩阵乘法和向量运算, 还涵盖了一系列其他基本的线性代数操作, 如矩阵-矩阵加法, 矩阵-矩阵乘法, 向量内积等. 这些操作的高效实现在科学计算, 工程应用等领域起到了至关重要的作用.

BLAS 的起源, 可以追溯到 1970 年代. 当时科学计算社区对于高性能线性代数运算的需求日益增加. 为了满足这一需求, BLAS 标准应运而生, 它定义了一套常用的线性代数子程序接口, 使得科学家和工程师能够以一致的方式调用这些功能.

BLAS 的发展历程中, 有些关键的实现库, 其中包括 Maltba(Mathematical Software for Transcendental Bundle Adjustment)以及两个主要的 BLAS 实现库, 分别是 Intel Math Kernel Library(MKL)和 OpenBLAS.

BLAS 库通常会针对特定硬件进行优化, 以获得最佳性能. 除了以上提及的两个主要实现库之外, 还有一些其他针对不同硬件架构的 BLAS 实现, 如 cuBLAS(CUDA Basic Linear Algebra Subprograms)用于 NVIDIA GPU 加速.

BLAS 的演进过程中, MKL 成为一个引人瞩目的商业库. MKL 由英特尔公司开发, 是一款高度优化的数学库, 专为英特尔架构的处理器而设计. MKL 不仅实现了 BLAS 标准, 还提供了针对英特尔处理器的优化, 使得线性代数运算能够在这些处理器上获得最佳性能. 这使得科学家和工程师在进行大规模计算时能够更有效地利用硬件资源. BLAS 的最新标准是 BLAS3, 包括了一些针对高性能计算的新操作和规范.

此外, 我们需要介绍 OpenBLAS: 一款由中国开发者共同努力创建的开源 BLAS 库. 它致力于提供高性能, 跨平台的线性代数运算支持. OpenBLAS 的发展反映了全球开源社区在科学计算领域的贡献, 为用户提供了在不同硬件平台上进行高效线性代数计算的选择. 在 Maltba 中, BLAS 的应用体现在其对于大规模捆绑调整问题的解决上. Maltba 利用了 BLAS 库的高性能特性, 通过优化的线性代数运算实现对于复杂问题的求解, 从而在计算机视觉和三维重建等领域取得了显著的成果.

BLAS 的应用也十分广泛. 作为科学计算领域的基础工具, 包括数值模拟, 信号处理, 数据分析等. 在不同实现库的支持下, 为研究人员和工程师提供了强大的数学计算能力. MKL 和 OpenBLAS 作为两个代表性的实现, 分别展现了商业和开源领域在这一领域的贡献, 推动了科学计算的发展. 深度学习框架(如 TensorFlow, PyTorch)也广泛使用了 BLAS 库, 尤其是在进行矩阵计算和神经网络训练时. 这进一步突显了 BLAS 在现代计算领域的重要性.

第三部分　如何更新 R 自带的 BLAS

(一) 在 Windows 中使用 OpenBLAS 的 R 教程

这是一篇简短的文章, 演示了如何轻松让 Windows 上的 R 使用 OpenBLAS 作为线性代数操作的后端.

大多数教程都是关于从源代码构建 R 本身与 OpenBLAS, 这非常复杂, 但只要愿意使安装变得更重约 120MB, 对于当今的硬盘来说应该不是问题.

简而言之: R 自带两个 DLL 文件 Rblas.dll 和 Rlapack.dll, 分别包含来自 BLAS 和 LAPACK 的功能. 这些可以被替换为实现相同接口的其他 DLL 文件, 如 libopenblas.dll 或 libmkl_rt.dll. 本教程说明了如何用 OpenBLAS 更优化的版本替换这些文件.

什么是 OpenBLAS

OpenBLAS 是 BLAS(基本线性代数子程序)和 LAPACK(线性代数包)的后端, 用于执行快速的线性代数运算, 例如矩阵乘法, 解线性系统, 计算特征值等. 这些听起来像是微不足道的操作, 但它们在不同的软件中执行速度可能差异很大, 因此用于它们的特定后端非常重要.

默认情况下, R for Windows 使用其自己未优化的 BLAS 和 LAPACK 替代品, 它们在功能上做得很好(对于 R 来说提供了所有必要的功能并计算结果正确), 但由于它们没有利用现代 CPU 的所有特性, 因此比优化的 BLAS 要慢得多.

要按照以下步骤进行, 需要以下内容:

1. 对 R 安装文件夹具有写入权限. 如果 R 是通过安装程序中的默认选项(全部选择‘Yes’)安装的, 这意味着需要管理员权限.

2. 一些可以提取压缩 zip 文件的压缩/解压软件, 如 7-zip.

详细说明

1. 关闭当前正在运行的任何 R/RStudio 会话, 如果有的话.

2. 从 OpenBLAS 的 GitHub 发布页面下载 Windows 的最新版本. 确保下载适用于您平台的正确版本.

3. 解压文件. 如果使用 7-zip, 右键单击文件, 选择“7-zip”, 然后选择“提取到...”或类似的选项.

4. 配置 Windows 资源管理器以显示文件扩展名: 点击顶部栏的“查看”, 勾选“文件扩展名”.

5. 在解压的文件夹内, 找到一个名为 libopenblas.dll 的子文件夹, 不应该在同一个文件夹下有其他以.a 结尾的文件(如果有, 请确保该文件至少几十兆字节). 这很可能在解压文件的路径中的一个名为 lib 的文件夹下.

6. 找到 R 本身安装的文件夹. 通常, 这应该是类似于: C:\Program Files\R\R-4.3.0(或者根据您安装的版本而有所不同).

7. 在 R 文件夹内, 找到子文件夹 bin\x64(例如 C:\Program Files\R\R-4.3.0\bin\x64).

8. 在这个文件夹里应该有两个关键文件: Rblas.dll 和 Rlapack.dll. 将它们复制到其他地方作为备份, 以防出现问题.

9. 从 bin\x64 中删除这两个文件(Rblas.dll 和 Rlapack.dll).

10. 将从 zip 文件中提取的 openblas dll 文件复制到这个相同的文件夹两次(参见下面的 GIF 视频).

11. 将其中一个副本重命名为 Rblas.dll, 另一个重命名为 Rlapack.dll(参见下面的 GIF 视频).

12. 可选地, 如果重新安装软件包时出现有关缺少 DLL 的错误, 将来自 zip 文件的 DLL 的第三个副本保留在相同的文件夹中, 但使用其原始名称(例如 libopenblas.dll - 也就是说, 应该有 3 个相同文件的副本, 名称分别为 Rblas.dll, Rlapack.dll 和 libopenblas.dll).

在这一点上, 您已经完成了, 下次启动 R 时, 它将使用 OpenBLAS 进行加速的线性代数操作.

控制线程数量

OpenBLAS 支持多线程, 可以通过 RhpcBLASctl 软件包动态控制, 也可以通过环境变量 OPENBLAS_NUM_THREADS 控制. 重要的是, 此环境变量需要在启动 R 之前设置, 这可以通过通过 Windows 控制面板进行设置来完成. 本教程说明了如何设置该变量.

首先, 您需要找出您的 CPU 支持的最大线程数. 这可以通过运行 R 函数 parallel::detectCores()来得知.

知道了这个数字后, 现在您可以将其配置为 OpenBLAS 的默认值. 首先, 转到控制面板(单击 Windows 开始按钮并选择“设置”):

在控制面板中, 搜索“环境变量”并选择编辑它们的选项:

在弹出的菜单中, 选择为您的用户添加一个新变量:

将变量命名为 OPENBLAS_NUM_THREADS, 并为其设置来自 parallel::detectCores() 的值(这里设置为 16):

重新启动计算机. 下次启动 R 时, OpenBLAS 将配置为利用 CPU 的所有可用线程以更快地运行.

第四部分　基于更新之后的 OpenBLAS 的比较

在前文中, 我们对 R-BLAS 与 R-OpenBLAS 进行了一定的介绍, 探讨了更新底层库对 R 语言在数据分析任务中的性能提升. 接下来, 我们将更加详细地关注任务主题, 通过具体的性能测量数据进一步分析 R-BLAS 与 R-OpenBLAS 的表现.

1-5 任务主题: 基础矩阵运算和数据处理

在这一类任务中, 我们观察到 R-OpenBLAS 在任务 1 至任务 5 的表现相较于 R-BLAS 有轻微的性能提升. 具体而言, 在创建, 转置, 矩阵乘法等基础矩阵运算中, 更新后的 OpenBLAS 库使得 R 语言在执行某些基础任务上的效率有所改善, 更新底层库对于基础矩阵运算和数据处理任务带来了一定的优化效果. 但限于数据体量较小, 这种优势并不明显, 亦或是受到随机干扰的影响更大一些.

表格: 基于 R-BLAS, R-OpenBLAS 进行基础矩阵运算和数据处理的效率对比

任务编号	R-BLAS	R-OpenBLAS
1	0.115	0.125
2	0.447	0.447
3	0.560	0.552
4	0.015	0.019
5	0.007	0.003
总和	**1.144**	**1.146**

6-10 任务主题: 高维矩阵运算和分解

在这一类任务中, R-OpenBLAS 在任务 6 至任务 10 的表现明显优于 R-BLAS. 特别是在涉及到高级矩阵运算(如: 高维 Cholesky 分解), 回归分析和矩阵分解等任务上, 更新后的 OpenBLAS 库展现了更高的执行效率. 这说明更新底层库不仅对基础操作有影响, 而且在高级矩阵运算和复杂分析中也发挥了积极作用.

表格: 基于 R-BLAS, R-OpenBLAS 进行高维矩阵运算和分解的效率对比

任务编号	R-BLAS	R-OpenBLAS
6	0.120	0.111
7	0.744	0.355
8	1.506	0.266
9	13.152	0.581
10	1.782	0.158
总和	**17.304**	**1.471**

11-15 **任务主题: 复杂数学计算和特征提取**

在这一类任务中, 我们发现 R-OpenBLAS 在任务 11 至任务 15 中相较于 R-BLAS 有显著的性能提升. 尤其是在解递推数列, 取特征值, 主成分分析等涉及复杂数学计算和特征提取的任务上, 更新后的 OpenBLAS 库为 R 语言带来了更高的运行效率. 这表明更新底层库对于处理复杂数学问题和高级统计任务的性能提升具有一定的促进作用.

表格: 基于 R-BLAS, R-OpenBLAS 进行复杂数学计算和特征提取的效率对比

任务编号	R-BLAS	R-OpenBLAS
11	0.210	0.218
12	0.080	0.079
13	1.197	1.189
14	0.025	0.026
15	0.002	0.001
总和	**1.514**	**1.513**

通过对比 R-BLAS 与 R-OpenBLAS 在不同任务主题下的性能, 我们可以得出结论: 更新后的 OpenBLAS 库在完成任务, 特别是高维矩阵运算上取得了较为显著的性能提升, 使得 R 语言在大数据分析领域中更具竞争力. 此优化对于从基础矩阵运算到复杂数学计算的各类任务都带来了实质性的改进.

第五部分　更加详细的比较

在本部分当中, 我们进行了 R-BLAS, R-OpenBLAS, Python 和 Matlab 在 15 个不同的统计相关任务上的效率对比. 这一系列任务涵盖了广泛的统计学相关问题, 包括但不限于矩阵分解, 高维数据分析等. 通过详细的性能测量, 我们能够深入了解每个平台在处理这些任务时的表现.

这一对比不仅有助于科研人员和数据科学家选择最适合其工作需求的工具, 也为开发者提供了有关优化和性能改进的重要见解. 通过深入研究每个工具在不同任务上的表现, 我们可以更好地理解它们的优势和局限性, 从而为统计计算领域的实际应用提供更可靠的指导.

表格: 任务清单

任务编号	任务内容
1	创建高维矩阵, 进行简单转置运算或矩阵乘法运算.
2	生成高维矩阵, 并进行高次矩阵乘法运算.
3	生成 7 百万个随机值, 并对随机值进行排序.
4	生成 2800 维向量做交叉乘积.
5	基于 3000 维矩阵的回归分析.
6	对二百万个随机值做快速 Fourier 变换.
7	对于高维矩阵取特征值.
8	对于高维矩阵取行列式.
9	对于高维矩阵进行 Cholesky 分解.
10	对于高维矩阵进行求逆运算.
11	求解 Fibonacci 递推数列.
12	创建高维 Hilbert 矩阵.
13	计算多对数组的最大公约数.
14	创建高维 Toeplitz 矩阵.
15	对高维矩阵做主成分分析.

在实验结果(见附表)当中, Matlab 以自身高效的运作速度, 在多项任务运行对比当中, 拔得头筹. 同时, 我们不难发现, R-OpenBLAS 相较于 R-BLAS, 运行效率上得到了显著的提升, 更新加速库的方案卓有成效!

附表.

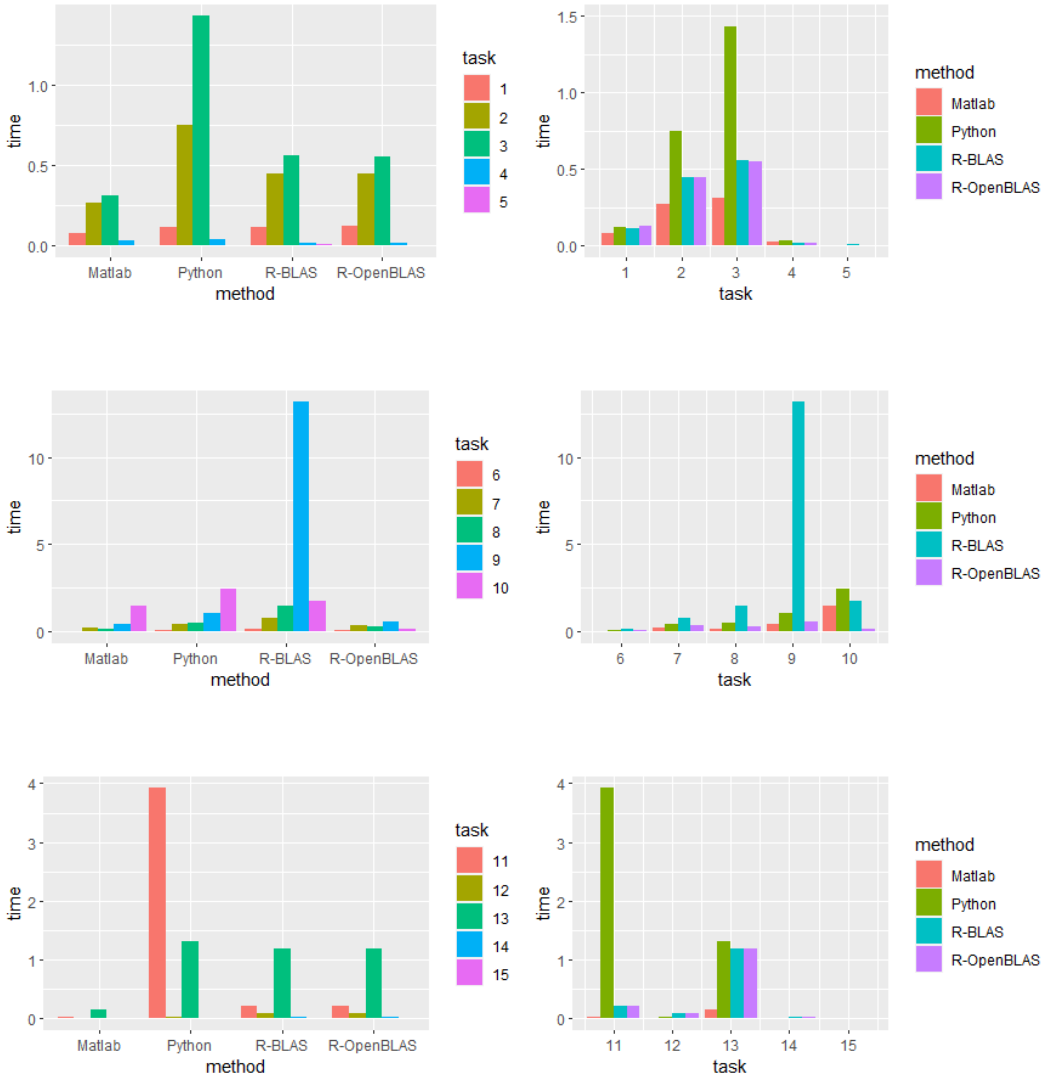
表格: 基于不同软件进行复杂运算的效率对比

任务编号	R-BLAS	R-OpenBLAS	Python	Matlab
1	0.115	0.125	0.117	0.079
2	0.447	0.447	0.750	0.269
3	0.560	0.552	1.427	0.310
4	0.015	0.019	0.036	0.028
5	0.007	0.003	0.002	0.001
6	0.120	0.111	0.102	0.035
7	0.744	0.355	0.406	0.193
8	1.506	0.266	0.512	0.181
9	13.152	0.581	1.073	0.426
10	1.782	0.158	2.475	1.468
11	0.210	0.218	3.922	0.020
12	0.080	0.079	0.025	0.014
13	1.197	1.189	1.316	0.151
14	0.025	0.026	0.000	0.003
15	0.002	0.001	0.001	0.014
总和	**19.962**	**4.130**	**12.164**	**3.192**

注. 在上述表格中, 蓝色字代表在同一次试验中, 运行程序最为缓慢的情形; 红色字代表在同一次试验中, 运行程序最为迅速的情形. 试验具有随机性, 不代表总体性能比较的必然性.

附图.

运行效率可视化


time
method
Matlab
Python
R-BLAS
R-OpenBLAS
task
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

致　谢

本项目由上海交通大学数学科学学院 2023 级应用统计专业组织编写. 在本项目编写的过程当中:

感谢王成老师对于本项目的倾力指导!

感谢刘若禺同学对于本项目的合作与贡献!

感谢李昌骏同学对于本项目提供的诸多宝贵建议!

项目仍然存在诸多提升空间, 殷切读者们对本项目工作提供宝贵的意见. 在此, 编者表达衷心的感谢!

编 写 组

2024 年 1 月

参 考 文 献 或 链 接

文　献

[1] R for Data Science, Hadley Wickham and Garrett Grolemund, 2017.

[2] Advanced R, Hadley Wickham, 2014.

链　接

[1] https://csantill.github.io/RPerformanceWBLAS.

[2] https://github.com/david-cortes/R-openblas-in-windows.

[3] https://www.microsoft.com/en-us/download/details.aspx?id=51205.