mlpack 4：快速、仅包含头文件的 C++ 机器学习库

Ryan R. Curtin、MarcusEdel、Omar Shrit、Shubham Agrawal、Suryoday Basak、James J. Balamuta、Ryan Birmingham、Kartik Dutt、Dirk Eddelbuettel、Rishabh Garg、Shikhar Jaiswal、Aakash Kaushik、Sangyeon Kim、Anjishnu Mukherjee、Nanubala Gnana Sai、Nippun Sharma、Yashwant Singh Parihar、Roshan Swain、Conrad Sanderson

arXiv:2302.00820v1 [cs.MS] 2 Feb 2023

摘要

15 年来，mlpack 机器学习库一直是 C++语言机器学习领域的“瑞士军刀”。它对常见及前沿机器学习算法的高效实现，已应用于广泛的科学和工业领域。本文概述 mlpack 4，这是相较于前代版本的一次重大升级。该库经过大幅重构和重新设计，旨在简化从原型开发到部署的流程，其中包括对其他语言（Python、Julia、R、Go 和命令行）的绑定，使原型开发能够无缝地在 C++以外的环境中进行。mlpack 是开源软件，采用宽松的三条款 BSD 许可证发布；可从以下网址获取：https://mlpack.org

背景

机器学习的应用几乎已遍及所有科学学科和无数商业应用 [3, 14]。这些应用几乎都有一个重要的共同点：由于参数数量庞大且训练数据规模巨大，机器学习往往需要大量计算。这正是最初使用 C++ 语言开发 mlpack 的主要动因，因为该语言能够实现高效、贴近底层的实现 [5, 6]。

但速度并非一切：如果整个流程过于困难或繁琐，开发和部署使用机器学习的应用也会受到严重阻碍 [17, 21]。此外，部署环境往往存在计算或工程方面的限制，使得采用完整技术栈的 Python 解决方案难以实现 [12]。因此，提供轻量且易于部署的机器学习解决方案至关重要。这促使我们对 mlpack 4 进行重构和重新设计：将高效实现与简便、轻量的部署相结合，使 mlpack 适用于各种部署环境。有关我们动机的更完整说明，请参阅 mlpack 愿景文档 [19]。

mlpack 是一款通用机器学习库，面向学术和商业用途；例如，需要高效性和易于部署的 数据科学家，或需要灵活性和可扩展性的研究人员。虽然还有其他可通过 C++ 使用的机器学习库，但其中许多库（如 FAISS [13] 和 FLANN [20], ）仅支持少数特定算法，而不像 mlpack 那样提供全面的机器学习算法。另一方面，Dlib-ml [15], 确实提供了丰富的机器学习算法工具集，但其可扩展性较为有限，因为它没有采用基于策略的设计 [1] 来支持任意用户定义的行为，而且其提供的机器学习算法范围也小于 mlpack。

功能

该库包含种类繁多的机器学习算法，其中一些是 mlpack 4 中新增的。算法包括线性回归、逻辑回归、随机森林、最远邻搜索 [8],加速 k 均值变体 [4], 核密度估计 [18], 以及快速最大核搜索 [9]。此外，该库还提供深度神经网络模块，实现了众多层类型、激活函数以及强化学习应用。有关可用功能的详细信息，请参阅在线 mlpack 文档（https://www.mlpack.org/docs.html）。多项研究 [5, 11] 借助 mlpack 的基准测试系统 [10] 展示了这些实现的高效性。

发表于：《开放源代码软件杂志》，第 8 卷，第 82 期，2023 年。**DOI：**10.21105/joss.05026

这些算法可通过自动生成的绑定在 Python、R、Go、Julia 和命令行中使用。所有语言绑定都采用统一的接口；例如，在 Python 中训练的模型可以在 Julia 或 C++ 中使用（也可以在任何提供 mlpack 绑定的其他语言中使用）。这些绑定可通过各语言的包管理器以及 apt 和 *dnf* 等系统级包管理器获取。此外，DockerHub 上提供了已完成环境配置、可直接使用的 Docker 容器；BinderHub 上还提供了通过 *xeus-cling* 项目（https://github.com/QuantStack/xeus-cling）使用的交互式 C++ 笔记本界面。

用户使用自己选择的语言开发机器学习工作流后，部署就很简单了。mlpack 库现在是仅含头文件的库，并且直接依赖的库只有三个：*Armadillo* [24], *ensmallen* [7], 和*cereal* (`https://github.com/USCILab/cereal`)。使用 C++时，唯一的链接要求是使用 BLAS 和 LAPACK 的高效实现（通过 Armadillo 引入）。这大大简化了部署；仅依赖 BLAS/LAPACK 的独立 C++ 应用程序可以轻松部署到多种环境，包括基于 Linux 的标准 Docker 容器、Windows 环境以及资源受限的嵌入式环境。为此，mlpack 的构建系统现在还包含多种支持交叉编译的工具，包括轻松对编译后的程序进行静态链接的功能（这对某些部署环境十分重要）。

重大变更

下面详细介绍 mlpack 4 中的一些重大变更。完整详尽的列表（包括大量错误修复和新技术）可参阅随 mlpack 一同分发的 *HISTORY.md* 文件。

- **移除依赖项**。根据愿景文档 [19], 的规划，大部分重构和重新设计工作都着眼于减少依赖项和编译开销。因此，我们用轻量级替代方案取代了 mlpack 先前依赖的 *Boost C++* 库（https://www.boost.org），其中包括用于序列化的 *cereal* （https://github.com/USCILab/cereal）。整个神经网络模块经过重构，避免使用 *Boost* （几乎相当于彻底重写）。这项工作带来了丰硕成果：在 mlpack 3 中，即使是简单程序，编译时通常也需要数 GB 内存。经过重构并移除依赖项后，编译通常只需数百 MB 内存，而且速度往往提升一个数量级。

- **交互式笔记本环境**。可以通过 *xeus-cling* 项目（https://github.com/QuantStack/xeus-cling）在 Jupyter 笔记本环境中使用 mlpack [16] 。mlpack 主页（https://www.mlpack.org）上提供了交互式演示。mlpack 示例代码库（https://github.com/mlpack/examples）中有 C++ 笔记本示例，这些示例也可以轻松地在 BinderHub 上运行。

- **新增绑定并扩大可用范围**。 通过 mlpack 的自动绑定系统，现已支持 Julia [2], Go [22], 和 R 语言 [23, 25]。安装相应语言的包管理器（*Pkg.jl*、*go get*、*install.packages('mlpack')*）中的 mlpack，即可使用这些绑定。此外，由于 mlpack 精简了依赖项，大幅简化了部署流程，mlpack 的 Python 依赖项现已通过 PyPI 和 conda-forge 支持多种架构。

- **支持交叉编译并改进构建系统**。 mlpack 的构建配置现已支持轻松进行交叉编译，例如使用 *buildroot* (https://buildroot.org) 等工具链。用户只需指定几个标志，即可为各种嵌入式系统构建可用的 mlpack 环境。为此，我们实现了依赖项自动下载器，可下载 *OpenBLAS* (https://github.com/xianyi/OpenBLAS), 并在必要时为目标架构进行编译。自动下载器也可在任何情况下启用和使用，从而简化安装和部署。

致谢

mlpack 的开发由社区主导。它凝聚了 220 多位个人（截至本文撰写时）的辛勤努力。我们也感谢多年来提交错误报告的各位。Google 通过十多年来持续参与 Google Summer of Code（谷歌编程之夏）计划，为开发提供了支持；此外，mlpack 也得到 NumFOCUS 的支持，该组织为其提供财务赞助。

参考文献

[1] A. Alexandrescu.《现代 C++ 设计：泛型编程与设计模式的应用》。Addison-Wesley，2001 年。[2] J. Bezanson、A. Edelman、S. Karpinski 和 V. B. Shah。Julia：数值计算的新方法。《SIAM评论》，59(1):65–98，2017 年。[3] G. Carleo、I. Cirac、K. Cranmer、L. Daudet、M. Schuld、N. Tishby、L. Vogt-Maranto 和 L. Zdeborov´a。机器学习与物理科学。《现代物理评论》，91(4):045002，2019 年。[4] R. R. Curtin。用于大 k 值快速 k 均值聚类的双树算法。载于 SIAM国际数据挖掘会议（SDM ’17），第 300–308 页。SIAM，2017 年。[5] R. R. Curtin、J. R. Cline、N. P. Slagle、W. B. March、P. Ram、N. A. Mehta 和 A. G. Gray。MLPACK：可扩展的 C++机器学习库。《机器学习研究杂志》，14(1):801–805，2013 年。[6] R. R. Curtin、M. Edel、M. Lozhnikov、Y. Mentekidis、S. Ghaisas 和 S. Zhang。mlpack 3：快速、灵活的机器学习库。《开放源代码软件杂志》，3(26):726，2018 年。[7] R. R. Curtin、M. Edel、R. G. Prabhu、S. Basak、Z. Lou 和 C. Sanderson。用于灵活数值优化的 ensmallen库。《机器学习研究杂志》，22:1–6，2021 年。[8] R. R. Curtin 和 A. B. Gardner。利用数据相关候选选择快速近似搜索最远邻居。载于国际相似性搜索与应用会议（SISAP 2016），第 221–235 页。Springer，2016 年。[9] R. R. Curtin 和 P. Ram。双树快速精确最大核搜索。《统计分析与数据挖掘：ASA 数据科学杂志》，7(4):229–253，2014 年。[10] M. Edel、A. Soni 和 R. R. Curtin。自动基准测试系统。载于 NIPS机器学习软件工程研讨会，2014 年。[11] D. Fang 和 D. H. Chau。M3：通过内存映射扩展机器学习规模。载于国际数据管理会议，第 2249–2250 页，2016 年。[12] L. Fischer、L. Ehrlinger、V. Geist、R. Ramler、F. Sobiezky、W. Zellinger、D. Brunner、M. Kumar 和 B. Moser。人工智能系统工程——关键挑战与经验教训。《机器学习与知识提取》，3(1):56–83，2020 年。[13] J. Johnson、M. Douze 和 H. J´egou。利用 GPU 进行十亿规模的相似性搜索。《IEEE大数据汇刊》，7(3):535–547，2019 年。[14] M. I. Jordan 和 T. M. Mitchell。机器学习：趋势、观点与前景。《科学》，349(6245):255–260，2015 年。[15] D. E. King。Dlib-ml：机器学习工具包。《机器学习研究杂志》，10:1755–1758，2009 年。[16] T. Kluyver、B. Ragan-Kelley、F. P´erez、B. E. Granger、M. Bussonnier、J. Frederic、K. Kelley、J. B. Hamrick、J. Grout、S. Corlay 等。Jupyter笔记本——一种用于可复现计算工作流的出版格式。载于第 20 届国际电子出版会议论文集，第 87–90 页，2016 年。[17] A. Lavin、C. M. Gilligan-Lee、A. Visnjic、S. Ganju、D. Newman、S. Ganguly、D. Lange、A. G. Baydin、A. Sharma、A. Gibson 等。机器学习系统的技术成熟度等级。《自然·通讯》，13(1):1–19，2022 年。[18] D. Lee 和 A. G. Gray。利用蒙特卡罗多极法快速计算高维核求和。神经信息处理系统进展，21:929–936，2008 年。[19]mlpack社区。mlpack：打造高效的机器学习库，实现从原型到部署。https://www.mlpack.org/papers/vision.pdf。[20] M. Muja 和 D. G. Lowe。利用自动算法配置快速近似搜索最近邻。载于 2009 年国际计算机视觉理论与应用会议论文集（VISAPP ’09），第 331–340 页，2009 年。[21] A. Paleyes、R.-G. Urma 和 N. D. Lawrence。机器学习部署的挑战：案例研究综述。《ACM计算综述》（CSUR），2020 年。[22] R. Pike。Google 的 Go：服务于软件工程的语言设计。载于第 3 届系统、编程与应用：软件造福人类年度会议论文集，第 5–6 页，2012 年。[23] R核心团队。R：统计计算语言与环境。R统计计算基金会，维也纳，奥地利，2022 年。[24] C. Sanderson 和 R. R. Curtin。Armadillo：基于模板的 C++ 线性代数库。《开放源代码软件杂志》，1(2):26，2016 年。[25] Y. Singh Parihar、R. R. Curtin、D. Eddelbuettel、J. Balamuta 等。mlpack：适用于 mlpack 库的 Rcpp 集成，2022 年。R软件包版本 4.0.0。