# 首个针对同质和异构数据、经证明达到最优的异步随机梯度下降方法

学位论文，作者：

Artavazd Maranjyan

arXiv:2601.02523v1 [math.OC] 5 Jan 2026

部分满足以下要求

以获得

哲学博士

沙特阿拉伯王国图瓦勒阿卜杜拉国王科技大学

©2025年12月4日
Artavazd Maranjyan 版权所有

htt
ps://artomaranjyan.github.io

# 摘要

## 首个针对同质和异构数据的可证明最优异步随机梯度下降算法 Artavazd Maranjyan

人工智能已在语言建模、计算机视觉和自主系统等领域取得了显著成功。这些突破得益于规模不断扩大的神经网络，以及利用数千个图形处理器或张量处理器在海量数据集上进行的训练。此类训练运行可能占用整个数据中心数周至数月，并消耗大量计算和能源资源。

虽然硬件和数据可用性的进步使这种扩展成为可能，但用于训练的优化算法并未以同样的速度发展。大多数大规模训练仍依赖于同步方法，在这种方法中，所有工作节点都必须完成各自的任务，下一次迭代才能开始。随着设备数量增加，同步造成的低效率也可能随之加剧：较快的工作节点在等待较慢的工作节点时处于空闲状态，计算资源和能源都因此被浪费。在实践中，几乎不可能让所有工作节点以完全相同的速度运行——硬件故障和网络延迟不可避免地会造成计算异质性。

乍看之下，似乎有一个简单的解决办法——移除同步机制——从而让所有工作节点持续运行。然而，异步性会引入陈旧性：某些结果是使用模型的过时版本生成的，这使算法更难分析，尤其是在延迟源于系统级变异而非算法本身的情况下。尽管进行了大量研究，异步方法的时间复杂度仍未得到充分理解。

本学位论文致力于弥补这一空白。我们为异步一阶随机优化建立了一个严谨的框架，聚焦于这类方法旨在应对的核心挑战：工作节点速度异质性。在此框架下，我们研究随机梯度下降（随机梯度下降），并证明，通过恰当的设计，异步随机梯度下降 在时间复杂度方面可以达到可证明的最优性，与近期仅针对同步版随机梯度下降取得的最优性结果相媲美。

第一项贡献RingmasterASGD通过有选择地丢弃陈旧更新，在同质数据 场景下实现了最优时间复杂度。第二项贡献RingleaderASGD利用一种用于协调模型更新的结构化梯度表机制，将这一结果扩展到异构数据 场景——这也是联邦学习的典型特征。最后，ATA 通过学习工作节点的计算时间分布并自适应地分配任务，提高资源效率，以少得多的计算量实现近乎最优的实际运行时间。

总之，这些结果确立了异步优化作为并行与分布式学习在理论上可靠、在实践中高效的基础——它们表明，无需同步的协调不仅可行，而且这种策略在理论上具有最优时间复杂度，在实践中也优于其他同步方法。

# 致谢

首先，我想向我的导师 Peter Richt´arik 教授表达最诚挚的谢意。他一直以来的支持、鼓励和深刻指导，使这篇学位论文得以完成。与他的每一次讨论都让我深受启发、思路更加清晰，推动我走过这段充满挑战的旅程。我从他身上学会了如何从基本原理出发思考复杂问题——在研究中追求严谨、简洁与数学之美。他让我明白，有影响力的研究往往始于提出简单却根本的问题。他的创造力与思想深度，深刻影响了我处理问题的方式。

我也由衷感谢我的合著者——Abdurakhmon Sadiev、Alexander Tyurin、ElMehdiSaad、FrancescoOrabona、LaurentCondat、MherSafaryan 和 Omar Shaikh Omar。在攻读博士学位期间，我有幸与他们密切合作。特别感谢 Francesco Orabona 教授——他在阿卜杜拉国王科技大学开设的在线学习课程是我最喜欢的课程之一，我也非常享受与他进行的许多富有启发性的讨论。

除了与合著者的合作，我也有幸与阿卜杜拉国王科技大学研究团队中的许多杰出同事共事，并与他们进行了无数次技术及非技术方面的交流。这些交流塑造了我的研究兴趣，也让我的这段时光真正充满乐趣。他们是：Ahmad Rammal、Alina Abdikarimova、Alexander Gaponov、Avetik Karagulyan、Egor Shulgin、Elnur Gasanov、Grigory Malinovsky、Hanmin Li、Hussein Rammal、Igor Klimczak、Igor Sokolov、Ivan Ilin、Kai Yi、Kaja Gruntkowska、Konstantin Burlachenko、Majied Ammar Mahran、Samuel Horv´ath、Sarit Khirirat、Slavom´ır Hanzely、Yassine Maziane、Yury Demi- dovich 和 Zhirayr Tovmasyan。

我还要感谢 Yi-Shuai Niu 教授在我赴中国北京进行研究访问期间给予我的接待。这次访问让我有幸结识了令人敬佩的教授，并参观了北京的几所知名大学。

我尤其感谢 Hrant Khachatrian 和 Mher Safaryan。若没有他们的帮助与指导，我可能不会了解到阿卜杜拉国王科技大学，也或许不会在这里开启博士生涯。我也衷心感谢阿卜杜拉国王科技大学为我提供卓越的研究环境和持续的机构支持。

最后但同样重要的是，我由衷感谢家人和朋友在这段旅程中始终如一的关爱与鼓励。特别感谢我的母亲，她无尽的支持和无私的付出让这一切成为可能。

# 目录

**摘要** 2

**致谢** 3

**图目录** 8

**表目录** 11

**1 引言** 14

1.1 同步训练：元算法 .151.1.1 实践中的同步挑战.151.2 异步训练：元算法 .161.3 论文重点 .171.3.1 范围：数据并行 .181.4 问题公式化 .18

1.4.1 优化问题 .181.4.2 假设.201.4.3 工作者异质性模型 .21

1.5 工作者与服务器动态 .23

1.5.1 工作者操作 .231.5.2 服务器操作.23

1.6 同步与异步算法：形式化定义 .241.6.1 同步算法 .241.6.2 异步算法 .251.7 固定计算下同质环境的算法

时间 .25

1.7.1 HeroSGD .261.7.2 朴素小批量随机梯度下降 .271.7.3 朴素异步随机梯度下降 .281.7.4 RennalaSGD .29

1.8 本工作的开端 .311.9 贡献与结构 .31

1.9.1 第 2 章 – Ringmaster ASGD：首个具有最优时间复杂度的异步随机梯度下降 .321.9.2 第 3 章 – Ringleader ASGD：首个在数据异质性条件下具有最优时间复杂度的异步随机梯度下降 .341.9.3 第4章 – ATA：分布式机器学习中的高效资源管理自适应任务分配 .361.10 本学位论文未收录的出版物 .391.11 常用符号 .40

**2 RingmasterASGD：首个具有最优时间复杂度的异步随机梯度下降算法42**

2.1 引言.42
2.1.1 假设.442.1.2 相关工作 .442.1.3 贡献 .46

2.2 预备知识与朴素方法 .46
2.2.1 一种朴素最优异步随机梯度下降算法 .472.2.2 为什么算法 2.2.1 被称为“朴素”? . . . . . 48
2.3 RingmasterASGD .49
2.3.1 描述 .49
2.3.2 延迟阈值 .502.3.3 为什么忽略旧的随机梯度? . . . . . 502.3.4 与 Rennala SGD 的比较 .502.3.5 与先前异步随机梯度下降变体的比较 . . . 512.3.6 停止无关计算 .51

2.4 理论分析 .522.4.1 阈值选择 .542.4.2 证明技术 .542.5 任意计算动态下的最优性 . . . . . 552.6 实验.572.6.1 神经网络实验 .582.7 其他相关工作 .582.8 结论与未来工作 .58

**3 Ringleader ASGD：首个针对数据异质性的、具有最优时间复杂度的异步随机梯度下降60**

3.1 引言.603.1.1 贡献 .63

3.2 相关工作 .643.3 问题设置 .653.3.1 工作者异质性模型 .653.3.2 符号说明 .66

3.3.3 假设.66
3.4 背景与动机 .67
3.4.1 朴素小批量随机梯度下降 .673.4.2 MaleniaSGD .683.4.3 迈向异步方法 .703.4.4 IA$^2$随机梯度下降 .71

3.4.5 动机.72
3.5 RingleaderASGD .72
3.5.1 详细描述 .743.5.2 延迟控制分析 .763.5.3 与 IA 的比较$^2$随机梯度下降 .773.5.4 与 Malenia SGD 的比较.77

3.6 理论结果 .783.6.1 迭代复杂度 .783.6.2 时间复杂度 .813.7 实验.823.8 结论.83

**4 ATA: Adapti ve Tas k All ocation for Effici ent 分布式机器学习中的资源管理 85**

4.1 引言.854.1.1 贪心任务分配 .854.1.2 GTA 的浪费性 .864.1.3 本工作的目标 .864.1.4 一个激励性示例：最优并行 SGD . . . . . 864.1.5 贡献 .874.2 相关工作 .874.3 问题设置 .884.3.1 任务分配协议 .884.3.2 建模假设 .884.3.3 分配算法的目标 .894.4 自适应任务分配 .904.4.1 化归为多臂老虎机与代理损失 . . . . . 904.4.2 与组合老虎机设定的比较 . . . 914.4.3 自适应任务分配算法 .91

4.4.4 总计算时间上界 . . . . . 934.5 经验自适应任务分配 .944.6 理论结果 .944.6.1 ATA 的保证 .954.6.2 ATA-Empirical 的保证 .964.7 实验.97

**5 总结101**

5.1 贡献总结.1015.1.1 Ringmaster ASGD（第 2 章）.1015.1.2 Ringleader ASGD（第 3 章）.1025.1.3 自适应任务分配（第4章）.102

5.2 未来方向.103

5.2.1 特定算法扩展.1035.2.2 更广泛的研究方向.1045.3 总结.105

**参考文献** **107**

**附录** **122**

**A 第 2 章附录 123**

A.1 引理 2.4.2 的证明 .123A.2 引理 2.5.1 的证明 .124A.3 定理 2.4.1 的证明 .125A.3.1 引理 A.3.1 的证明 .127A.3.2 引理 A.3.2 的证明 .128A.4 定理 2.5.2 的证明 .129A.5 第 2.2 节示例的推导.131A.6 当初始点是 $\varepsilon$–驻点 . 132

**B 第3章附录 13** **3**

B.1 任意变化的计算时间 .133B.1.1 通用计算模型 .133B.1.2 迈向最优方法 . 134B.1.3 一种最优方法 .134

B.2 辅助引理 .136

B.2.1 引理 3.3.4 的证明.136B.2.2 方差项 .137B.2.3 引理 3.6.1 的证明. 138B.2.4 引理 3.6.2 的证明.142

B.3 IA 的时间复杂度²随机梯度下降.143改进的 Malenia SGD.144

**C 第4章附录 14** **6**

C.1 补充实验 .146C.1.1 线性噪声 .146C.1.2 异质时间分布 .146C.1.3 遗憾 .147

C.1.4 真实数据集 .148C.1.5 时间分布中先验知识的影响 . . . . . 148

C.2 具体优化方法 .149C.2.1 随机梯度下降 .149C.2.2 异步 SGD .152

C.3 递归分配选择算法 .153

C.3.1 最优性.155C.3.2 最小基数 .156

C.4 定理 4.6.1、定理 4.6.3 和定理 4.4.2 的证明 . . . 157

C.4.1 定理 4.6.1 的证明 .159C.4.2 定理 4.6.3 的证明 .162C.4.3 定理 4.4.2 的证明 .166

C.5 技术引理 .166

# 图目录

1.7.1 Hero SGD （算法 1.7.1）示意图：包含 3 个工作节点。只有速度最快的工作节点（底部）执行更新，其他工作节点则处于空闲状态。从理论上说，可以利用它们空闲的计算能力来计算额外的梯度。 .261.7.2 朴素小批量随机梯度下降 （算法 1.7.2）示意图：包含 3 个工作节点。阴影区域表示空闲时段，此时较快的设备必须等待最慢的设备——计算资源因此被浪费，且无法进行任何有用的计算。 .27

1.7.3 朴素异步随机梯度下降 示意图（算法 1.7.3），包含 3 个工作节点。请注意没有阴影区域：所有工作节点始终处于活动状态，每个计算得到的梯度都会用于更新模型。 .28

1.7.4 Rennala SGD（算法 1.7.4）的示意图，包含 3 个工作节点。阴影区域表示空闲时段，或在同步过程中最终被丢弃的梯度计算。. . . . . 30 2.6.1 使用 $n = 6174$ 和 $d = 1729$ 进行实验，展示 Ringleader ASGD、延迟自适应异步随机梯度下降和 Rennala SGD 的收敛情况。. . . . . 58 2.6.2 我们在一个小型两层神经网络上进行实验，该网络使用 ReLU 激活，并在 MNIST 数据集上训练。实验表明，我们的方法 Ringmaster ASGD 更具鲁棒性，且优于延迟自适应异步随机梯度下降和 Rennala SGD.59 3.7.1 收敛比较：绘制梯度范数平方的中位数$\|\nabla f(x^k)\|^2$ （实线）及四分位距（阴影区域）随实际运行时间的变化，并对 30 个随机种子取平均。设置：使用架构为 Linear($d$, 128) $\to$ ReLU$\to$ Linear(128, 10) 的两层多层感知机，在 (a) MNIST 和 (b) Fashion-MNIST 数据集上训练。客户端延迟：异构延迟模拟为$\tau_i = i + |\eta_i|$，其中每个客户端 $i \in [n]$ 的 $\eta_i \sim \mathcal{N}(0, i)$，且 $n = 100$。结果：在步长经过最优调优后，Ringleader ASGD 的收敛速度快于 Malenia SGD 和 IA$^2$SGD，尽管 Ringleader ASGD 和 Malenia SGD 具有相同的理论时间复杂度保证.84

4.7.1 每一行的工作节点数量都增加为原来的 3 倍，从 17 开始，即 $n = 17$、51、153、从上到下依次为 459。第一列展示运行时间与次优性之间的关系。第二列也绘制次优性，但以工作者总时间为横轴，即 $\sum_{i=1}^{n} T_i^k$算法 4.4.1 中的内容。第三列展示平均迭代时间，其计算方式为 $c_k/k$ 对所有迭代 $k$取平均值。最后一列展示平均累积遗憾，定义见式 (4.9)。. . . . . .

98

C.1.1每一行的工作节点数量从 17 开始，每次增加为原来的 3 倍，即 $n = 17$，51，153，从上到下依次为 459。第一列展示运行时间与次优性之间的关系。第二列也绘制次优性，但横轴为工作者总时间，即算法 4.4.1 中的 $\sum_{i=1}^{n} T_i^k$。第三列展示平均迭代时间，其计算方式为在所有迭代 $c_k/k$ 中取平均 $k$。最后一列展示平均累积遗憾，如式（4.9）所定义。. . . . . .

147

C.1.2每一行对应工作节点数量逐渐增加的情况，从上到下依次为 $n$ =15、45、150。我们考察五种分布——指数分布、均匀分布、半正态分布、对数正态分布和伽马分布——将它们分组，使每组的均值相同，再改变不同组之间的均值。结果表明，这些算法在不同分布下仍然稳健。各列所示内容与图 C.1.1 相同。 .1

49C.1.3迭代中的遗憾增长。.150

C.1.4我们使用 CIFAR-100 数据集（Krizhevsky 等人，2009年）。该模型是一个卷积神经网络，包含三层卷积层和两层全连接层，共有 160k 个参数。使用 Adam 优化器（Kingma 和 Ba，2014年），步长为恒定值 $8 \cdot 10^{-5}$。工作节点的计算时间设置与图 C.1.1 相同，其中均值随时间线性增长。批量大小保持不变，为 $B = 23$。每一行对应不同数量的工作节点，从上到下分别为 $n = 17$，51，153。

.151C.1.5我们使用与图 C.1.4 相同的优化设置，其中 $B$ =23 和 $n = 51$。先前迭代的数量 $P$在各行之间有所不同，从顶部开始依次为 $P = 0$、$5 \cdot 10^5$、$5 \cdot 10^6$。随着先前迭代数量的增加，我们观察到ATA 和 ATA-Empirical 的训练速度加快，其性能逐渐接近 OFTA 的最优性能。 .152

# 表目录

1.1 本学位论文的每个章节都介绍一项主要的算法贡献，并在第二列中连同其参考文献一并列出。第三列表示该方法是否实现了最优时间复杂度，即是否达到了收敛时间的理论下界。第四列说明每种方法所适用的数据体制——同质或异质。第五列描述分析中所采用的工作节点计算时间模型，而第六列表示该方法是否具有“资源高效”特性。前两种方法仅专注于最小化实际运行时间，因此利用所有可用的计算资源；而第三种方法ATA则在实际运行时间仅损失常数因子的情况下，提高了计算效率。 .1.2 本学位论文未收录的出版物。 .

32

39

2.1 在非凸情形下，异步随机梯度方法执行步骤 $x^{k+1} = x^k - \gamma^k \nabla f(x^{k-\delta^k}; \xi_{i^k}^{k-\delta^k})$，以获得一个 $\varepsilon$–平稳点的时间复杂度。本表考虑第 2.2 节中的固定计算模型。缩写：$\sigma^2$ 定义为 $\mathbb{E}_\xi[\|\nabla f(x; \xi) - \nabla f(x)\|^2] \le \sigma^2$，对所有 $x \in \mathbb{R}^d$， $L$ 是 $f$ 的光滑常数， $\Delta := f(x^0) - f^*$， $\tau_i \in [0, \infty]$ 是工作节点 $i^k$ 计算单个随机梯度所需的时间上界。

43

3.1 固定计算时间模型下并行一阶方法的时间复杂度比较，其中每个工作节点 $i$ 计算一个随机梯度需要固定时间 $\tau_i$ ，且各时间按 $\tau_n$ 为最大值的顺序排列（3.2）。我们用 $\tau_{\text{avg}} := \frac{1}{n}\sum_{i=1}^{n}\tau_i$ 表示所有工作节点的平均计算时间。该表展示了不同算法的时间复杂度如何依赖于若干关键问题参数：初始函数次优性 $\Delta := f(x^0) - f^*$（假设 3.3.5）、目标平稳性 $\varepsilon$、随机梯度的方差上界 $\sigma^2$ （假设 3.3.1）以及光滑常数。具体而言， $L_f$ 是 $f$ 的光滑常数（定义 3.3.3）； $L_{\max} := \max_{i\in[n]} L_{f_i}$ ，其中 $L_{f_i}$ 是 $f_i$ 的光滑常数； $L$ 是与我们新提出的光滑性类假设（假设 3.3.2）相关的常数。它们满足 $L_f \le L \le L_{\max}$ （引理 3.3.4）。所有给出的时间复杂度均隐去通用常数因子。每一列都标明一种方法是否满足以下理想性质：最优：达到 Tyurin 和 Richt´arik（2024年）针对异构数据环境下并行一阶随机方法推导的理论下界。无需同步：不需要同步，因此属于异步方法。无空闲工作节点：所有工作节点始终忙于工作，无需等待，从而充分利用计算资源。无丢弃工作：不浪费任何计算，也不会让工作节点在计算过程中途停止。我们提出的新方法 Ringleader ASGD 是首个实现最优时间复杂度的异步方法，同时确保充分利用资源（没有空闲工作节点），且不丢弃任何计算/工作.

62

4.1 达到$f(x)-f^* < 10^{-5}$所需的工作者总时间与运行时间之比。对于工作者总时间，我们将 GTA 的工作者总时间除以所列其他算法对应的工作者总时间。对于运行时间，则反过来，将其他算法的运行时间除以 GTA 的运行时间，因为 GTA 速度最快。为简化名称，我们将 ATA-Empirical 称为ATA-E。 .

99

C.1 达到$f(x) - f^* < 10^{-5}$所需的工作者总时间与运行时间之比。对于工作者总时间，我们将 GTA 的工作者总时间除以所列其他算法对应的工作者总时间。对于运行时间，则反过来，将其他算法的运行时间除以 GTA 的运行时间，因为GTA 速度最快。 .148

# 第1章

# 引言

人工智能（AI）曾主要是一个研究课题，如今却塑造着数十亿人日常使用的应用。如今最引人注目的例子是 ChatGPT 等对话式人工智能系统及其他聊天机器人（Brown et al., 2020; Achiam et al., 2023; Team et al., 2023）。除了聊天之外，人工智能还为推荐系统（Covington et al., 2016; Zhang et al., 2019）、自动驾驶（Bo- jarski et al., 2016; Chib & Singh, 2023）、图像识别（Krizhevsky 等人, 2012; Maur´ıcio et al., 2023）和医学诊断（Esteva et al., 2017; Ra- jpoot et al., 2024）提供动力。这些应用的共同点是都依赖于使用海量数据训练的大规模深度学习模型（Kaplan et al., 2020）。

这些模型对计算资源的需求极高，以至于无法在单台机器上高效训练，甚至无法高效部署（Dean 等人, 2012年; Li 等人,2020a）。它们需要在由专用硬件构成的集群上进行分布式训练，通常涉及数千个互连的图形处理器或张量处理器（Jouppi et al., 2017年;Narayanan 等人, 2021年）。因此，大型数据中心——用于互连这些硬件的庞大设施——已成为必不可少的基础设施，由 Google、Meta 和亚马逊等公司建造。如今，最先进的人工智能训练任务往往需要在这些数据中心进行数周或数月的计算（Narayanan 等人,2021年; Grattafiori 等人, 2024年）。

随着模型和数据集不断增长，人工智能训练对计算和能源的需求也在不断增加。数据中心已经属于能耗最高的建筑类型之一，其需求增长速度快于大多数其他行业，对可持续能源系统构成了重大挑战（Strubell 等，2020年；Agency，2025年）。近期估计表明，到2030年，数据中心的能源消耗量可能达到与日本目前年度用电量相当的水平（Agency，2025年）。

随着人工智能训练对计算资源和能源的需求不断增长，提高大型模型训练效率已成为一个关键且备受关注的研究领域（Shoeybi等人，2019年；Rajbhandari等人，2020年；Narayanan等人，2021年；Hu等人，2022年；Wan等人，2024年）。尽管硬件和软件迅速发展，目前的大规模训练流程仍有很大的改进空间。例如，LLaMA-3 的训练中，模型浮点运算利用率仅约为 38–43%（这是衡量训练效率的一项标准指标（Chowdhery等人，2023年））（Grattafiori 等人，2024年）。这意味着，在大部分时间里，昂贵的图形处理器都处于闲置状态，却仍在消耗电力，而没有执行有用的计算。这种利用不足就像预订整支机队一周，却每天只让其中少数飞机起飞，其余飞机则停在跑道上空转，发动机仍在运行。在这种情况下，资源浪费造成的成本极其高昂；而对于大型人工智能训练任务，这种效率低下会转化为数十亿美元的基础设施开支和能源消耗（De Vries，2023年；Cottier 等人，2024年；O’Donnell & Crownhart，2025年；

Elsworth 等人，2025年）。

大规模人工智能训练中的资源利用不足来自多个方面，包括当前硬件的限制，以及在机器之间传输数据所产生的通信开销。然而，一个尤为重要的低效来源在于算法层面。大多数现有训练算法最初并非针对当今大规模计算集群而设计，因而未能对这类大规模系统进行最优的利用。具体而言，它们依赖跨机器的同步，而随着系统规模扩大，这种方法的成本会越来越高。

## 1.1 同步训练：元算法

历史上，大规模人工智能训练一直依赖同步迭代算法，即所有工作节点必须完成各自分配的任务后，系统才能进入下一次迭代。这一策略一直是训练大型人工智能模型的主流方法（Goyal 等人, 2017年; Li 等人, 2020a;Douillard 等人, 2024年).

为说明这些方法背后的原理，我们抽离底层优化问题的具体细节，并提出一个通用的元算法（算法1.1.1），它刻画了许多同步方法所共有的结构。“同步算法”的精确定义留待第1.6节；在此，该元算法有意保持非正式，用以突出共同行为及其瓶颈。

**算法 1.1.1** 同步元算法（服务器视角）

1: **对于**迭代 $k = 0$，1，2，.....**执行**2: 向所有工作节点广播当前模型 $x^k$ 3: 指示所有工作节点使用 $x^k$4: 等待所有工作节点完成，并接收它们的结果5: 汇总收集到的结果6: 更新模型，得到 $x^{k+1}$7: **结束循环**

这些方法的定义性特征是同步屏障：服务器必须等待所有工作节点完成（第4行）后，才能汇总它们的结果（第5行）、计算新模型（第6行），并再次将其广播给工作节点（第2行）。我们特别指出第4行，因为它是瓶颈：速度较快的工作节点无法开始新的计算，只能闲置，直到最慢的工作节点完成。这种闲置是同步方法的主要缺点，也是大型数据中心资源利用率不足的重要原因。

### 1.1.1 实践中的同步挑战

人们或许希望通过构建采用相同硬件的数据中心，确保所有工作节点以相同速度运行，从而解决同步问题。然而在实践中，这一策略面临着若干根本性障碍。

### 即使硬件相同，掉队节点仍然存在

即使集群中的所有机器都完全相同，掉队节点——即因计算或通信延迟而运行缓慢的工作节点——仍会出现，因为存在硬件故障（例如过热）和网络问题等不可预测的问题（Dean 与 Barroso，2013年；Ananthanarayanan 等，2013年；Maranjyan 等人，2025a）。单个组件发生故障，就可能导致一个工作节点明显落后于其他节点，迫使整个集群等待。换言之，相同的硬件规格并不能保证实践中的性能完全一致。

### 硬件升级的成本高得令人望而却步

强制保持硬件同质性会带来第二个问题：这会使数据中心的升级变得极其困难。出现更新、更强大的处理器时，将它们加入现有集群会引入异质性。由于在同步训练中，速度较快的机器仍必须等待较慢的机器，升级带来的大部分性能优势都会丧失。这使运营者陷入代价高昂的两难境地：要么同时更换所有机器（成本极高），要么等到能够整体更换时再进行升级。此类限制使得逐步扩充容量或在新技术推出后及时采用变得困难。

### 联邦与边缘环境

最后，还有一些重要场景，根本无法确保硬件完全相同。在联邦学习中，训练会在智能手机、传感器或其他个人电子设备等日常设备上进行（Koneˇcnýet 等人，2016年，a；McMahan 等人，2017年）。这些设备在计算能力、电池续航时间、互联网连接和可用性方面差异巨大。参与方可能会不可预测地加入或退出训练过程，连接质量也可能大幅波动。在此类环境中，等待速度最慢的设备完成每一轮不仅效率低下，而且可能根本无法实现，因为有些设备可能会断开连接，或无法在合理时间内完成任务。

## 1.2 异步训练：元算法

鉴于上述讨论的同步固有限制，研究人员提出了一种算法解决方案：干脆彻底取消同步要求（Tsitsiklis 等人，1986年；Recht 等人，2011年；Agarwal 和 Duchi，2011 年）。如果同步屏障是效率低下的根源，那么消除它应该能让工作节点持续运行，不再有空闲时间。

为了说明这种方法，我们再次给出一个通用元算法（算法1.2.1），它概括了许多异步方法共有的结构。与之前一样，“异步算法”的精确定义将在第1.6节中给出。

**算法 1.2.1** 异步元算法（服务器视角）

1: 初始化模型并发送给所有工作节点
2: **当**尚未终止时**执行**
3: 接收来自某个工作节点的结果 $i$
4: 无需等待——立即继续
5: 可选地更新模型（依据某个判据）
6: 将当前模型发送给工作节点 $i$
7: 工作节点 $i$ 重新开始计算
8: **结束 while 循环**

同步方法与异步方法的关键区别体现在第4行：服务器一收到任一工作节点的结果便立即继续运行，无需等待其他工作节点。同样，在第7行，每个工作节点都会立即继续计算，无需等待其他工作节点。这种设计消除了同步方法中令人头疼的空闲时间，使所有工作节点都能独立、持续地使用各自当前的模型版本运行。

然而，这种独立性是有代价的。当某个工作节点开始计算并随后返回结果时，其他工作节点可能已经在此期间多次更新了模型。这意味着该工作节点的结果是基于模型的过时版本，即陈旧版本。此类陈旧性可能会阻碍优化进展，甚至减缓收敛或导致次优解。

此外，第 5行表明，异步方法还提供了额外的灵活性——服务器无需在收到每个结果后都更新模型。这些观察引出了根本性的设计问题：应如何设计异步方法，在最大限度发挥其计算优势的同时减轻陈旧性的影响?

## 1.3 论文重点

虽然异步方法消除了同步方法的关键效率问题——工作节点在等待慢速工作节点时处于闲置状态——但也带来了新的挑战：模型更新的陈旧性。这造成了一个根本性的悖论。陈旧性并非只是异步性的副作用，而正是能够持续利用工作节点的机制。工作节点之所以能够继续计算，恰恰是因为它们不必等待模型完全更新到最新状态。然而，同样的陈旧性意味着更新是在过时信息的基础上计算得出的，这可能会妨碍优化进展，甚至损害收敛。如果工作节点基于严重过时的模型进行计算，其更新可能不再有益，甚至会主动损害训练过程。

预测和管理陈旧性面临多方面的挑战。我们难以预判给定系统会经历多大程度的陈旧性、陈旧性会如何影响收敛，以及哪些设计选择能够使算法在实践中表现良好，同时维持理论保证。此外，陈旧性也使分析变得困难得多：同步方法的经典收敛性分析不再适用，因此必须开发新的理论工具，以理解异步行为。

本学位论文的核心目标是深入理解异步优化，并为大规模机器学习设计可证明高效的异步算法。

### 1.3.1 范围：数据并行

现代分布式训练系统采用多种并行方式来处理大型模型和数据集，包括数据并行，即工作节点处理不同的数据子集；模型并行，即将模型的不同部分分布到各个工作节点；以及流水线并行，即将模型的不同层分配到流水线的不同阶段（Narayanan 等人，2021年；Grattafiori 等人，2024年）。实践中，最先进的训练系统通常会同时结合其中若干种甚至全部策略。

本学位论文专门研究数据并行，这是最基础且适用范围最广的并行方式。在数据并行中，所有工作节点都保存完整模型的副本，但处理不同批次的训练数据。我们出于两个原因将研究范围限定于这种设置。首先，数据并行是最简单的并行方式，也是开展理论研究的自然起点。其次，尽管异步数据并行看似简单，但我们对它的理解仍然有限——关于收敛、最优算法设计以及陈旧性影响的基本问题仍未解决。

## 1.4 问题公式化

在概述了本学位论文的总体重点后，下面我们对所研究的问题进行具体公式化。我们首先介绍分布式机器学习的核心优化问题。然后，由于我们主要关注算法的收敛时间，因此我们将形式化定义哪些操作需要时间，以及在分析中如何对这些量进行建模。

### 1.4.1 优化问题

我们考虑一个包含 $n$ 个客户端的分布式学习设置，其中每个客户端 $i$ 都可以访问自己的本地数据分布 $\mathcal{D}_i$。目标是通过求解优化问题，学习一个由向量 $x \in \mathbb{R}^d$参数化的全局模型，其中 $d$表示模型参数的维数，

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \left\{ f(x) \coloneqq \frac{1}{n}\sum_{i=1}^{n} f_i(x) \right\}, \tag{1.1}$$

其中，$f_i\colon \mathbb{R}^d \to \mathbb{R}$ 表示客户端 $i$的局部目标函数，全局目标函数 $f$ 定义为它们的平均值。

局部目标函数通常可以写成

$$f_i(x) \coloneqq \mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left[f_i(x;\xi_i)\right] , \tag{1.2}$$

其中，$f_i(x;\xi_i)$ 是模型在数据样本 $\xi_i$ 上的损失函数，该模型由 $x$ 参数化。因此，局部目标函数是该损失在

整个本地数据分布 $\mathcal{D}_i$ 上的期望。这种表述对应于最小化总体风险。

当本地数据分布是有限的，即 $|\mathcal{D}_i| < \infty$时，本地目标函数（1.2）可化简为本地训练数据点上的平均损失，

$$f_i(x) = \frac{1}{|\mathcal{D}_i|} \sum_{j=1}^{|\mathcal{D}_i|} f_i(x; \xi_{i,j}) \ .$$

## 同质与异质数据

通常，根据各客户端的本地数据分布 $\mathcal{D}_i$，其本地目标函数 $f_i$ 可能存在显著差异。这种异质性给求解分布式优化问题带来了根本性挑战，也是区分不同实际场景的一个关键特征。

**异构环境**。 在联邦学习场景中，数据异质性不可避免。例如，当客户端是智能手机等个人设备时，每台设备都包含反映特定个人习惯、偏好和环境的用户特定数据。若重新分发或集中存储此类数据，就会违反隐私约束。因此，各客户端的本地数据分布 $\mathcal{D}_i$ 可能存在巨大差异——从不同语言输入的文本到在不同条件下拍摄的照片皆是如此，这使得优化问题本身更具挑战性。我们将这种情况称为异构数据环境。

**同质环境**。 相比之下，数据中心环境在组织和分发数据方面具有更大的灵活性。在这类环境中，无需担忧隐私问题，数据可以自由地进行混洗、复制或集中存储。因此，通常会确保所有工作节点都能访问来自同一底层分布的数据，方法包括共享一个公共数据集，或对数据进行分区，使各工作节点得到的子集在统计上近似相同。形式化地说，这相当于假设 $\mathcal{D}_i = \mathcal{D}$ 对所有客户端 $i$均成立。

此外，我们通常在所有客户端上采用相同的损失函数，即$f_i(x;\ \xi_i) \equiv f(x;\ \xi)$。在这些假设下，优化问题可简化为 (1.1)

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \left\{ f(x) \coloneqq \mathbb{E}_{\xi \sim \mathcal{D}} \left[ f(x; \xi) \right] \right\} . \tag{1.3}$$

我们将此称为同质数据设置。

这两种环境都具有实际意义，也带来了各自不同的算法挑战。同质环境虽然结构更明确，但仍需设计高效的分布式算法。异构环境更具一般性，因此需要能够适应客户端之间统计多样性的稳健方法。重要的是，这两种环境需要分别研究：仅分析一般的异构情形，并不一定能为同构情形带来最优的解决方案。在同质环境中，我们可以利用所有客户端共享相同数据分布这一事实，从而获得更紧确的保证，并

且通常能够采用更简单的算法。在本学位论文中，我们针对这两种环境分别开发并分析了相应的算法。

### 1.4.2 假设

为了明确问题设置，我们针对所研究的函数类引入若干标准假设。

**假设 1.4.1**（光滑性）。函数 $f$ 可微，且其梯度是 Lipschitz连续的。也就是说，存在一个常数 $L > 0$，使得对所有 $x$，$y \in \mathbb{R}^d$，

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\| .$$

这一假设通常称为 $L$-光滑性，它限制了梯度变化的速度。

**假设 1.4.2**（下有界性）。目标函数有下界：存在 $f^* > -\infty$，使得对所有 $x \in \mathbb{R}^d$，$f(x) \geq f^*$。

这一假设确保优化问题定义良好：若没有这一假设，函数值可能无限下降，从而使收敛保证失去意义。

我们定义 $\Delta := f(x^0) - f^*$，其中 $x^0$ 是优化算法的初始点。量 $\Delta$ 表示初始次优性，可用作衡量问题固有难度的指标。

### 收敛准则

在这些假设下，非凸优化中的标准目标是找到一个 $\varepsilon$-驻点，即满足以下条件的（可能是随机的）向量 $x$

$$\mathbb{E}\left[\|\nabla f(x)\|^2\right] \leq \varepsilon . \tag{1.4}$$

这一判据在非凸情形下很自然：尽管通常难以找到全局极小值，但一阶驻点是可以达到的，而且往往对应着实际有用的解。

### 随机一阶方法

我们重点研究随机一阶方法，这类方法仅依赖梯度信息。最早且最基础的示例是随机梯度下降（随机梯度下降），由 Robbins& Monro （1951年）提出，后来由 Bottou 等人 （2018年）在机器学习领域推广。在此基础上，Adam （Kingma 和 Ba，2014年）等自适应方法及其变体 AdamW （Loshchilov & Hutter，2019年）如今已成为训练大规模深度学习模型的标准方法。在我们的分析中，我们从 随机梯度下降 开始，这既是因为它简单，也因为它构成了许多现代变体的理论基础。为了研究其在分布式设置下的行为，我们对工作节点计算的随机梯度施加额外的假设。

**假设 1.4.3。** 对于每个 $i \in \{1, \ldots, n\}$ 以及每个 $\xi_i$，函数 $f_i(x; \xi_i)$ 关于其第一个参数 $x$ 可微。此外，随机梯度是无偏的，并且具有有界方差 $\sigma^2 \geq 0$，即：

$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i}\left[\nabla f_i(x; \xi_i)\right] = \nabla f_i(x), \quad \forall x \in \mathbb{R}^d, \ \forall i \in \{1, \ldots, n\}\ ,$$
$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i}\left[\|\nabla f_i(x; \xi_i) - \nabla f_i(x)\|^2\right] \leq \sigma^2, \quad \forall x \in \mathbb{R}^d, \ \forall i \in \{1, \ldots, n\}\ .$$

第一个条件确保随机梯度是真实梯度的无偏估计量，而第二个条件则限定其方差。参数$\sigma$ 量化了随机噪声的大小：当 $\sigma = 0$时，梯度是精确的。

### 1.4.3 工作者异质性模型

我们关注的是求解问题（*1.1*）的时间复杂度，即达到所需平稳性水平（1.4）所需的实际运行时间。为了分析分布式系统中的时间复杂度，我们对各个工作节点的计算特征进行建模。本节将通过工作节点的计算时间，正式定义工作者计算异质性的概念。

#### 固定计算时间

我们首先讨论工作节点以恒定计算速度运行的情形。这一设定为理解异步分布式优化的动态过程提供了直观基础，并将异质性的影响与其他变异来源区分开来。

遵循*Mishchenko* 等人（2022a）提出的固定计算模型，我们作如下假设：

$$\begin{aligned}&\text{Each worker } i \text{ requires } \tau_i \text{ seconds}^1 \text{ to compute one stochastic gradient } \nabla f_i(x; \xi_i).\\ &\text{Without loss of generality, we assume}\\ &\qquad 0 < \tau_1 \leq \tau_2 \leq \cdots \leq \tau_n\ .\end{aligned} \tag{1.5}$$

计算时间的差异 $\{\tau_i\}_{i=1}^n$ 可能源于多种因素：异构硬件（不同的 GPU 代际）、本地数据集大小不同，或对共享计算资源的竞争。该模型抓住了异步优化的核心挑战：设计能够高效利用进度各异的工作节点所提交更新的算法。

#### 时变计算时间

在实践中，计算时间很少是常数。即使在数据中心，工作节点也可能因共享资源使用、后台进程、过热或偶发的硬件故障而变慢。在联邦学习中，这种变异性更加明显，因为客户端可能面临不稳定的网络连接、电池续航时间有限，或受到用户活动的干扰。

为捕捉这些影响，我们还考虑计算时间随时间变化的模型（Tyurin，2024年）。对这一更一般情形的形式化处理

[1]在本学位论文中，我们使用“秒”作为通用时间单位。该分析适用于任何一致的时间单位。

将留待后续章节讨论（例如，参见第B.1.1节）。关键是，我们假设这些计算时间的变化并非由对手操控：它们可能取决于系统状况或外部因素，但不会取决于优化算法本身。这一假设反映了现实情形：延迟是由环境和硬件局限性造成的，而非蓄意干扰。

### 随机计算时间

虽然上述引入的时变模型几乎可以描述任何计算行为，但它并未提供有意义的理论保证——其性能完全取决于计算时间随时间的变化方式。为了获得更易于解释的结果，我们还考虑一种随机设定，其中每个工作节点的计算时间都被建模为从某个底层概率分布中抽取的随机变量。这种表述使我们能够推导期望意义下的保证，并明确刻画分布参数（例如均值、方差）如何影响算法的行为。本学位论文的部分内容采用了这一随机假设；其精确表述见第 4.3.2 节。

### 零通信与辅助开销

为便于阐述，我们假设唯一耗时的操作是计算随机梯度。所有其他操作——包括工作节点到服务器和服务器到工作节点的通信、服务器端模型更新，以及内存读写——均假定耗时为零秒。当然，这一假设并不现实，因为在实际系统中，通信延迟和服务器开销都不可忽视。

### 内存锁定与不一致读取

由 Recht 等人（2011年）发起的一项研究无锁 共享参数向量上的更新，其中多个工作节点同时进行读取和写入。在这种情况下，当一个工作节点正在读取服务器的参数向量时，其他工作节点可能会更改尚未读取的坐标；结果可能是，模型的本地副本混合了这些更新之前和之后取得的值。

我们在分析中不考虑此类影响，因为我们假设通信、读取和写入都是即时的（耗时为零）且具有原子性。因此，每次读取都会返回一个单一且明确定义的模型向量，而每次写入都会不可分割地完成。在这些假设下，此类不一致不会出现，因此无需加锁。

### 这些简化的理由

上述引入的假设有意排除了除工作节点计算时间之外的所有时间来源。我们采用这一简化设定，是因为异步方法旨在解决的主要挑战是工作节点速度的异质性。即使在这种理想化条件下，异步算法的行为也尚未得到充分理解：在许多情况下，它们并不比同步对应方法更具优势。因此，在引入额外的系统级复杂性之前，建立严格的

理解是必要的第一步。

## 1.5 工作者与服务器动态

在介绍具体算法之前，我们先概述工作节点和服务器可执行的基本操作。之后构建的所有算法都将基于这些基本操作。

### 1.5.1 工作者操作

每个工作节点在任何时刻只能执行以下两种动作：

- **空闲：** 工作节点什么也不做，等待服务器的指令。

- **计算并发送梯度：** 工作节点从独立同分布的数据中采样一个新数据点，在分配给它的模型上计算随机梯度，并立即将该梯度发送给服务器。这些随机梯度是无偏的，且其方差有界，正如假设 1.4.3 中所假定的。

这些是我们考虑的全部工作者操作。可以设想允许工作节点在发送梯度前将其存储在本地，但在我们的设定中，这不会造成任何差异，因为我们假设通信是瞬时的。

一种相关的方法是执行本地模型更新，这在联邦学习中很常见（McMahan 等人，2017年）。虽然本学位论文不考虑本地步数，但可以进行此类扩展（Tyurin 和 Sivtsov，2025年；Fradin 等人，2025年）。

### 1.5.2 服务器操作

服务器有更多可选动作，因为它必须协调整个训练过程。当工作节点传来梯度时，服务器可以：

- **存储梯度：**将其保留在内存中，以便将来更新时使用。

- **丢弃梯度：**例如，如果服务器判定该梯度已过时或质量不佳。

除了响应传入梯度外，服务器还可以随时独立执行以下动作：

- **更新模型：**使用已存储梯度更新模型。这包括梯度到达时立即更新，以及使用多个已存储梯度进行延迟更新。

- **丢弃已存储梯度：** 移除之前存储的部分梯度，例如这些梯度已经被充分使用或不再需要。

- **请求梯度：** 将当前模型发送给一个或多个工作节点，并指示它们计算梯度。实际上，服务器有时可能会要求某个工作节点在其已经使用的同一模型上重新计算梯度，在这种情况下无需再次发送模型。然而，由于我们假设通信是即时的，因此我们将请求简单地建模为始终发送模型。

- **终止一个或多个工作节点的计算：** 如果服务器预计某些正在进行的计算所产生的结果将被丢弃，则可以提前取消这些计算，以避免无谓的工作。受影响的工作节点随后只需返回空闲状态，直到收到新的指令。

尽管这些动作原则上可以在任何时间发生，但实际上，算法会明确规定它们发生的确切时机（例如，在收到梯度时、更新之后等）。因此，服务器的行为是有结构的，而非任意的。

在此框架内运行的方法将被称为分布式或并行方法。同步和异步算法都可以用这些动作来表示。下一节将形式化说明，在此框架内，同步方法与异步方法之间的区别是如何产生的。

## 1.6 同步算法与异步算法：形式化定义

现在，我们正式定义同步算法和异步算法的含义。

### 1.6.1 同步算法

如果服务器使用已存储的随机梯度执行模型更新时，仅使用基于当前服务器模型计算出的梯度，我们就称该算法为同步算法。等价地说，一次更新中的所有梯度都对应于同一模型状态。因此，这种更新与采用无偏梯度估计量的标准随机梯度下降步骤相一致。

形式上，同步更新可写为

$$x^{k+1} = x^k - \gamma^k \sum_{i=1}^{n} w_i^k \sum_{j=1}^{b_i^k} \nabla f_i\left(x^k; \xi_i^{k,j}\right), \tag{1.6}$$

其中 $\gamma^k > 0$ 为步长，$w_i^k \geq 0$ 为聚合权重，$b_i^k \in \{0, 1, 2, \ldots\}$为工作节点 $i$ 在第 $k$ 轮中贡献的梯度数量（按照惯例，如果$b_i^k = 0$，则总和取为 0）。本学位论文中讨论的所有同步算法都可以通过对（$\gamma^k$、$w_i^k$、$b_i^k$）作不同选择来套用此模板，包括HeroSGD、朴素小批量随机梯度下降和RennalaSGD （参见第1.7节）。

由于更新只能使用基于相同 模型状态计算出的梯度，同步方法存在固有的局限性。当工作节点速度不同时，它们要么（i）迫使较快的工作节点等待，直到最慢的工作节点完成；要么（ii）丢弃错过同步点的延迟计算。当然，如果

工作节点速度大致相同，这就不是问题，但正如第1.1.1节中所讨论的，这一点很难保证。

### 1.6.2 异步算法

相比之下，如果一种方法的更新可能使用基于过时模型副本计算出的梯度——也就是说，使用的模型版本早于服务器当前持有的版本——我们就称其为异步方法。虽然这种情况不一定会在每次迭代中发生，但异步方法通常必须考虑此类延迟。

为简单起见，我们假设在每一轮中，给定工作节点贡献的所有梯度都使用同一个模型迭代点计算。本文分析的所有异步方法都满足这一假设；将其放宽为允许每个工作节点在每轮中混合使用过时点也很容易做到，但对我们的目的而言没有必要。

令 $\delta_i^k \in \{0,1,2,\ldots\}$ 表示迭代 $k$时的延迟：即自工作节点 $i$收到其模型迭代点以来发生的服务器更新次数。用这一记号，异步更新的形式为

$$x^{k+1} = x^k - \gamma^k \sum_{i=1}^{n} w_i^k \sum_{j=1}^{b_i^k} \nabla f_i\left(x^{k-\delta_i^k}; \xi_i^{k-\delta_i^k,j}\right),$$

其中，$\gamma^k > 0$ 为步长，$w_i^k \geq 0$ 为聚合权重，$b_i^k \in \{0,1,2,\ldots\}$为工作节点 $i$ 在第 $k$ 轮中贡献的梯度数量。

注意，当所有 $\delta_i^k = 0$时，我们就恢复了同步更新（1.6）；有些算法偶尔会执行这样的同步步骤。不过，我们的目标是针对任何允许的延迟模式提供保证。异步方法的一个示例是朴素异步随机梯度下降，见第1.7节。

## 1.7 固定计算时间下同质设置的算法

现在我们已对问题设置进行了形式化描述，可以开始探讨算法解法。回顾一下，我们有 $n$ 台机器，每台都能够计算具有有界方差的无偏随机梯度。我们假设每台机器的计算时间固定，如（1.5）中的固定计算模型所规定，且服务器知道这些时间。

我们的目标是设计能够在此分布式设置中高效求解优化问题的算法。我们首先聚焦于齐次数据设置，目标是求解问题（1.3）。

如果不考虑时间复杂度，我们可以直接应用标准随机梯度下降，从而实现最优的迭代复杂度（Ghadimi 与 Lan，2013年；Ar- jevani 等，2022年）。因此，在单机设置中，朴素的随机梯度下降 是最优的。这一观察促使我们从一个自然的起点出发：从随机梯度下降开始，探究能否对其进行调整，使其在分布式设置中实现最优时间复杂度。

### 1.7.1 Hero SGD

在我们的设置中应用 随机梯度下降 的最简单方法，是忽略我们拥有 $n$ 台机器这一事实，只使用其中一台。在这种情况下，自然的选择是选取最快的机器——即计算时间为 $\tau_1$ 的机器——并在其上运行标准的随机梯度下降 。我们将这一基线称为 HeroSGD，[2] 相应的算法见算法 1.7.1。我们还在图1.7.1中使用一个包含三个工作节点的简单示例说明其行为。

**算法 1.7.1** HeroSGD

**输入：** 初始点 $x^0 \in \mathbb{R}^d$，步长 $\gamma^k > 0$ **对于** $k = 0, \ldots, K-1$ **执行** 请求最快的工作节点计算一个随机梯度 $\nabla f\left(x^k;\ \xi^k\right)$ 等待梯度到达 更新模型：

$$x^{k+1} = x^k - \gamma^k \nabla f\left(x^k; \xi^k\right)$$

**结束循环**

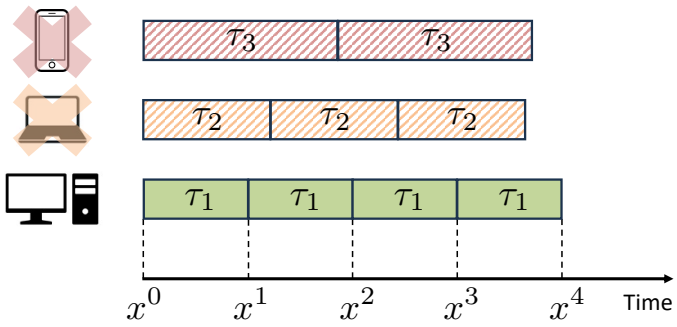



图 1.7.1： HeroSGD （算法 1.7.1）使用 3 个工作节点的示意图。只有最快的工作节点（底部）执行更新，其他工作节点保持空闲。原则上，可以利用它们空闲的计算能力来计算额外的梯度。

该算法需要知道哪个工作节点最快。如果无法获得此信息，可以采用 贪心策略：要求所有工作节点计算一个梯度，并使用最先到达的结果，丢弃其余结果。或者，也可以尝试学习各工作节点的计算时间分布，并据此分配任务；我们将在第 1.9.3 节稍后讨论此类自适应策略。不过，在本节中，我们假设计算时间固定且已知，因此无需考虑这些问题。

由于该方法每次迭代需要 $\tau_1$ 秒，而 随机梯度下降 的迭代复杂度已知为（Ghadimi 与 Lan，2013年）

$$K = \mathcal{O}\left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{\varepsilon^2}\right),$$

[2]这个名称取自Fradin 等人（2025年）。

总时间复杂度为

$$T = \tau_1 K \ .$$

虽然这种方法很简单，但它完全未能利用其余 $n-1$ 台机器的计算能力，这些机器在整个训练过程中都处于闲置状态。

### 1.7.2 朴素小批量随机梯度下降

一个自然的下一步是利用所有 $n$ 机器。最直接的做法是让每台机器在每次迭代中计算一个随机梯度，然后用这些梯度的平均值更新模型。这便得到了朴素小批量随机梯度下降算法，其形式化定义见算法1.7.2。与前一种情况类似，我们通过图1.7.2中的三工作节点示例来说明朴素小批量随机梯度下降的行为。

**算法 1.7.2** 朴素小批量随机梯度下降

**输入：** 初始点 $x^0 \in \mathbb{R}^d$，步长 $\gamma^k > 0$ **对于** $k = 0, \ldots\ldots, K-1$ **执行** 将当前模型 $x^k$ 广播给所有工作节点 每个工作节点 $i$ 计算一个随机梯度 $\nabla f\left(x^k;\ \xi_i^k\right)$ 等待接收所有 $n$ 个梯度 聚合梯度并更新模型：

$$x^{k+1} = x^k - \gamma^k \frac{1}{n} \sum_{i=1}^{n} \nabla f\left(x^k; \xi_i^k\right)$$

**结束循环**

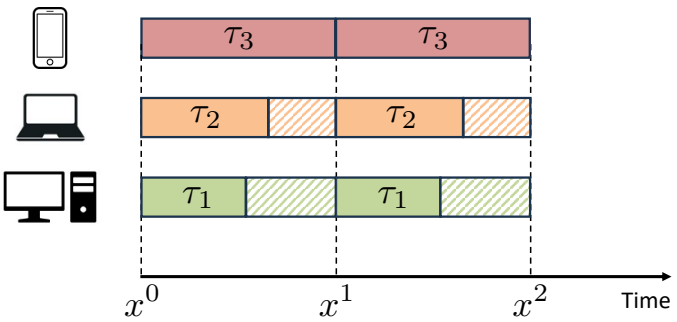



图 1.7.2： 朴素小批量随机梯度下降 （算法 1.7.2），使用 3 个工作节点。阴影区域表示空闲时段：较快的设备必须等待最慢的设备，造成计算资源浪费，无法完成任何有用的工作。

该方法的迭代复杂度与HeroSGD类似。令$K(n$表示朴素小批量随机梯度下降达到 $\varepsilon$–驻点所需的迭代次数；根据Ghadimi 与 Lan (2013年，我们有

$$K(n) = \mathcal{O}\left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right). \tag{1.7}$$

关键区别在于，现在方差项除以 $n$，这反映了对 $n$ 个独立梯度估计取平均所实现的方差缩减。

然而，每次迭代都需要 $\tau_n$ 秒——这段时间是最慢的工作节点完成计算所需的时间。因此，在最慢的机器完成任务、下一次迭代开始之前，所有速度更快的工作节点都只能闲置。因此，总时间复杂度为

$$\tau_n K(n) \ .$$

在异构环境中，这种同步开销可能相当显著：当工作节点速度差异很大时，训练的整体速度就由最慢的设备决定。在极端情况下，如果 $\tau_n$ 远慢于其他设备，使用多个工作节点几乎完全失去优势。

### 1.7.3 朴素异步随机梯度下降

为了解决同步方法中工作节点闲置导致的低效率问题，研究人员提出了异步随机梯度下降(Recht 等人，2011年；Agarwal 和 Duchi，2011年）这一理念。其核心思想很简单：允许所有工作节点持续且独立地运行，无需等待其他节点。每个工作节点都在自己的模型副本上计算随机梯度，而服务器会在任一工作节点完成计算时立即执行更新。我们将这种简单而朴素的算法称为朴素异步随机梯度下降：一旦收到任何梯度，就立即更新模型。其伪代码见算法1.7.3。与之前一样，我们在图1.7.3中展示了朴素异步随机梯度下降的三工作节点示意图。

**算法 1.7.3** 朴素异步随机梯度下降

**输入：** 初始点 $x^0 \in \mathbb{R}^d$，步长 $\gamma^k > 0$
工作节点开始计算随机梯度，计算点为 $x^0$
**对于** $k = 0, \ldots\ldots, K-1$ **执行**
梯度 $\nabla f\left(x^{k-\delta^k}; \xi_{i^k}^{k-\delta^k}\right)$ 从某个工作节点到达 $i^k$
更新模型：$x^{k+1} = x^k - \gamma^k \nabla f\left(x^{k-\delta^k}; \xi_{i^k}^{k-\delta^k}\right)$
工作节点 $i^k$ 开始计算 $\nabla f\left(x^{k+1}; \xi_{i^k}^{k+1}\right)$
**结束循环**

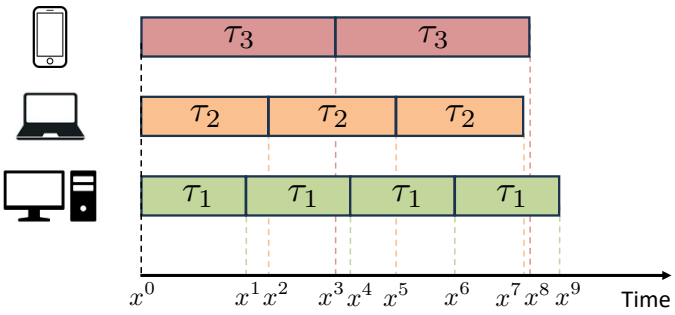



图 1.7.3：朴素异步随机梯度下降的示意图（算法 1.7.3），包含 3 个工作节点。请注意图中没有阴影区域：所有工作节点始终处于活动状态，并且每个计算得到的梯度都会促成一次模型更新。

在算法 1.7.3中，术语 $\delta^k$ 表示在迭代 $k$收到的梯度的 $\delta^k$ 延迟，计数的是

自工作节点 $i^k$ 开始计算其梯度以来，服务器上发生的模型更新次数。由于工作节点以异步方式运行，延迟自然会产生——当一个工作节点正在计算时，其他工作节点可能已经完成梯度计算，并在服务器上执行模型更新。

虽然该算法确保所有工作节点始终处于忙碌状态，避免了计算空闲，但也带来了一个新的挑战：陈旧性。基于过时模型计算出的梯度（ $\delta^k$ 较大）可能会误导更新，导致优化朝错误方向推进，或使收敛速度变慢。

过去十年间，人们提出了许多异步 随机梯度下降 的变体，以减轻这一影响，主要通过改善收敛界对最大延迟的依赖关系（Cohen 等人，2021年；Koloskova 等人，2022年；Mishchenko 等人，2022a）。尽管取得了这些进展，但当延迟较大时，理论保证往往仍然较为悲观。

另一条研究路线由Tyurin & Richt´arik （2024年）开创，采取了不同的路径。该研究没有试图缓解异步方法固有的延迟问题，而是将注意力重新转向同步场景，力求尽可能提高其性能。他们提出了一种名为 RennalaSGD 的方法。

## 1.7.4 Rennala SGD

Rennala SGD 算法通过一个关键洞见解决了朴素小批量随机梯度下降的低效问题：由于由哪个工作节点计算特定的随机梯度并不重要，因此无需从每个工作节点恰好收集一个梯度。相反，服务器会在梯度到达时持续收集，并在收集到“足够多”梯度后立即执行更新。该算法总结于算法1.7.4中。沿用相同的设置，我们在图1.7.4中展示了三个工作节点下Rennala SGD 的运行情况。

**算法 1.7.4** RennalaSGD (Tyurin 与 Richt´arik，2024年)

1：输入：初始点 $x^0 \in \mathbb{R}^d$，步长 $\gamma^k > 0$，批量大小 $B \in \{1, 2, \ldots\}$
2：对于 $k = 0, \ldots, K-1$，执行 3：将 $x^k$ 广播给所有工作节点以计算随机梯度 4：初始化 $g^k = 0$ 和 $b = 0$5：当 $b < B$ 时 6：来自工作节点 $i^k$ 的梯度 $\nabla f\left(x^k; \xi^k, {}^{b}_{i^k}, {}_{b}\right)$ 到达 $_{b}7$ $g^k = g^k + \nabla f\left(x^k \; \xi^{k\,b}_{i^{k}\,b}\right)$ : ; ,
, : 工作节点立即开始在 $x^k$ 处计算新的梯度 9：$b = b + 1$10：结束循环
11：更新模型：

$$x^{k+1} = x^k - \gamma^k \frac{g^k}{B}$$

12: **结束循环**

参数 $B$ 表示服务器在执行更新前计划收集的梯度数量。这些梯度会异步收集——一收到就收集——直到总共收到 $B$ 个。然而，由于RennalaSGD 归根结底是一种同步 方法，工作节点必须在一次

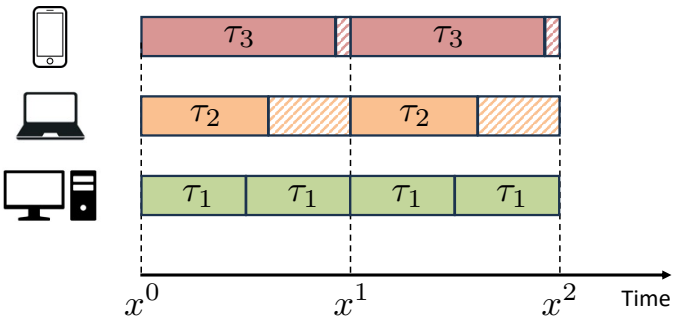



图 1.7.4：Rennala SGD 的示意图（算法 1.7.4），包含 3 个工作节点。阴影区域表示空闲时段，或在同步过程中最终被舍弃的梯度计算。

服务器更新模型时进行同步。具体而言，如第 3 行所示，当服务器广播新模型时，所有工作节点都必须丢弃 正在进行的梯度计算，并从更新后的点重新开始。或者，如果服务器知道所有工作节点的计算时间，就可以安排任务，以尽可能短的时间收集所需的 $B$个梯度；在这种情况下，一些工作节点完成分配给它们的梯度计算后，可能会闲置 下来，等待下一次同步。

迭代复杂度与朴素小批量随机梯度下降情形（1.7）类似，因为从根本上说，这就是采用异步批次收集的随机梯度下降：

$$K(B) = \mathcal{O}\left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{B\varepsilon^2}\right).$$

由于 $B$ 梯度取了平均，方差项现在随 $B$ 而非 $n$缩放。

然而，每次迭代所需的时间还需要进一步分析。正如 Tyurin& Richt´arik （2024年）所示（另见第 A.1节中的另一种证明），收集 $B$ 个梯度所需时间的一个上界为

$$t(B) := 2 \min_{m \in \{1,\ldots,n\}} \left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(1+\frac{B}{m}\right).$$

因此，总时间复杂度为

$$t(B)K(B)\ .$$

值得注意的是，Tyurin& Richt´arik （2024年）证明，通过选择

$$B = \max\left\{1, \left\lceil\frac{\sigma^2}{\varepsilon}\right\rceil\right\}\ ,$$

RennalaSGD 达到了最优时间复杂度。他们通过推导所有应用于光滑非凸目标函数的一阶随机并行方法的下界——这正是本论文所研究的设置——并证明 RennalaSGD的复杂度与这些下界相匹配，从而确立了这一结果。

一个重要的观察是，RennalaSGD 表现出的行为类似于一种

异步算法，因为它允许工作节点不必等待掉队节点即可继续运行，从而减少空闲时间。它几乎符合异步元算法框架中的算法 1.2.1。由于算法1.2.1 中的更新是可选的，服务器可以像RennalaSGD 那样累积梯度，并在收集到 $B$ 个梯度后执行一次更新。唯一的区别是，算法 1.2.1 无法同时向所有工作节点广播更新后的模型。不过，服务器无需一次性向所有工作节点广播，而是可以在每个工作节点提交下一个梯度时，单独向其发送更新后的模型。事实上，RennalaSGD 最初正是由Tyurin &Richt´arik （2024年）提出的。

然而，我们并未将 RennalaSGD 归类为异步算法，因为其更新规则（详见第1.6节）并非如此。

## 1.8 本研究的开端

除了证明 RennalaSGD 达到了最优时间复杂度之外，Tyurin& Richt´arik （2024年）还发现——或许令人惊讶的是——现有的异步随机梯度下降算法均未达到这一最优性。这些结果引出了一个根本性问题：

异步算法从根本上存在缺陷吗？

如果最优解属于同步算法这一类，学界是否应该放弃异步随机梯度下降，转而专注于同步方法？长期以来对异步性的热情是否 misplaced?

这个问题构成了本学位论文的核心动机。正如我们将在后续章节中展示的，答案是否。设计出能够达到最优时间复杂度的异步随机梯度下降算法确实是可能的。

此外，这些异步方法不仅实现了理论上的最优时间复杂度，在实践中也优于同步算法。如第 1.6 节所述，同步算法——即便是最优算法，例如 RennalaSGD——也存在一定的计算浪费：有些设备处于闲置状态，或者其工作成果会在同步过程中被丢弃。异步方法有望消除这种低效，从而在实际应用场景中带来额外的性能提升。

在本学位论文中，我们通过引入新的异步算法，重新彰显异步性的理论与实践意义。这些算法不仅经证明达到了最优水平，而且在经验上也具有高效性。

## 1.9 贡献与结构

本学位论文重点研究异步随机梯度下降 算法的设计，以实现理论上最优的时间复杂度。我们针对同质和异质数据环境开发了最优算法。对于同构情形，我们提出了 RingmasterASGD （章节2）；对于异质情形，我们引入了RingleaderASGD （章节3）。据我们所知，Maranjyan 等人 （2025d）；Maranjyan& Richt´arik （2025年）的研究提出了这些算法，它们是首批理论上最优的异步 随机梯度下降 方法。

表 1.1：本学位论文的每一章都介绍一项主要的算法贡献，列于第二列，并附有相应的参考文献。第三列说明该方法是否实现了最优时间复杂度，即是否达到收敛时间的理论下界。第四列指出各方法适用的数据体制——同质或异质。第五列描述分析中采用的工作节点计算时间模型，第六列则说明该方法是否“资源高效”。前两种方法只专注于缩短实际运行时间，因此会使用所有可用的计算资源；而第三种方法ATA则以实际运行时间仅增加常数因子为代价，提高了计算效率。

| 章节 | 算法<br>与参考文献 | 最优的<br>time<br>复杂度 | Data<br>异质性 | 工作节点<br>计算<br>动态 | 资源<br>高效 |
|---|---|---|---|---|---|
| 2 | Ringmaster ASGD<br>(Maranjyan 等人，2025d) | ✔ | ✘ | 固定且<br>任意变化的 | ✘ |
| 3 | Ringleader ASGD<br>(Maranjyan &Richt´arik, 2025年) | ✔ | ✔ | 固定且<br>任意变化的 | ✘ |
| 4 | ATA [a]<br>（Maranjyan 等人，2025b） | ✔[b] | ✘ | 随机 | ✔ |

[a] ATA（自适应任务分配）是一种元算法，可以应用于其他方法之上——例如 Ringmaster ASGD、Ren-nala SGD 或 Hero SGD——以提高资源效率。 [b] 这里的最优性定义有所不同：我们将时间复杂度与一个了解工作节点耗时分布的固定竞争基准进行比较，并证明 ATA 的期望性能在最优固定分配的常数倍范围内。

这些章节的主要贡献将在第 1.9.1节和第 1.9.2节中分别总结并作更详细的讨论。

在两种设定下都确立了最优方法之后，我们接下来将重点放在提升其资源效率上。虽然最优方法能够最小化总收敛时间，但这是通过利用所有可用计算资源实现的。然而，我们证明，在实现资源使用大幅节省的同时，稍微放宽时间最优性是可行的。为应对这一权衡，我们开发了自适应任务分配 (ATA算法，见第4章。该算法引入了一系列策略，使RingmasterASGD 及相关方法在同质数据体制下更具资源效率。第1.9.3节将进一步讨论这些想法。

最后，表 1.1 总结了各章节的主要贡献和区别。如表 1.1所示，本学位论文从最优异步方法（RingmasterASGD、RingleaderASGD）逐步发展到资源感知型扩展（ATA）。各列突出展示了每种方法的不同方面——最优性、异质性、计算时间模型和效率。在接下来的小节中，我们将详细概述各章节，阐明表格各列的含义，并进一步说明具体贡献。

### 1.9.1 章节2 – Ringmaster ASGD：首个具有最优时间复杂度的异步随机梯度下降

本章节中，我们聚焦于较为简单的情形——齐次数据设置。正如前文第 1.4.1 节所讨论的，尽管这种设置看起来可能更简单，但它绝不

意味着其重要性较低。这不仅仅是一种简化问题的理论假设；相反，它代表了一种独特且具有实际意义的情形，值得单独研究。事实上，这种设置与当今大多数大型数据中心的运行方式十分接近。

本章的目标是设计一种异步随机梯度下降方法，使其达到最优时间复杂度——也就是说，其上界与 Tyurin & Richt´arik (2024年提出的理论下界相匹配。换言之，我们旨在构建一种理论上尽可能快的算法。此前，只有一种同步方法达到了这种最优性——RennalaSGD (Tyurin & Richt´arik，2024年）。

我们发现，只需对朴素异步随机梯度下降算法进行一项简单却至关重要的修改，就能在异步环境中达到这种最优性。近期有多项工作尝试改进朴素异步随机梯度下降，例如引入延迟自适应步长（Mishchenko 等人，2022a；Koloskova 等人，2022年）。尽管这些方法优于同步小批量随机梯度下降基线，但总体而言仍非最优。我们的重新设计采取了不同的方法：它利用了第1.5节介绍的一种可选服务器操作——即丢弃某些传入的随机梯度。

核心思路如下。异步方法的主要问题在于陈旧性——一些随机梯度是使用过时的模型参数计算得出的，如果使用这些梯度更新模型，可能会损害收敛。朴素的朴素异步随机梯度下降算法（算法1.7.3）会贪心地应用所有收到的梯度，不论其延迟多久。相反，我们提出丢弃过度陈旧的梯度，即延迟超过某个阈值的梯度。通过谨慎选择该阈值，我们可以证明达到最优时间复杂度。

我们将由此得到的方法称为RingmasterASGD，其名称灵感来自马戏团领班这一比喻：领班能让混乱井然有序——驯服异步工作节点和延迟梯度的“狂野”行为，而这正是HOGWILD! （Recht 等人，2011年）的特点。该算法正式表述为算法1.9.1。

**算法 1.9.1** Ringmaster ASGD

**1: 输入：初始点** $x^0 \in \mathbb{R}^d$**，步长** $\gamma^k > 0$**，延迟阈值** $R \geq 1$**2: 设置** $k = 0$**3: 工作节点开始在** $x^0$ **处计算随机梯度4: 当尚未终止时执行**

5 $i$**: 梯度** $\nabla f\left(x^{k-\delta^k}\right.$**;**

6 $\left.\xi_i^{k-\delta^k}\right)$ **从工作节点到达：如果则 7: 更新模型：** $x^{k+1} = x^k - \gamma^k \nabla f\left(x^{k-\delta^k}\right.$**;** $\left.\xi_i^{k-\delta^k}\right)$**8: 工作节点** $i$ **立即开始计算** $\nabla f\left(x^{k+1}\right.$**;** $\left.\xi_i^{k+1}\right)$**9: 递增迭代计数器：** $k = k + 1$**10: 否则**

11 $\nabla f\left(x^{k-\delta^k}\right.$ $\left.\xi_i^{k-\delta^k}\right)$**: 丢弃过时梯度；：**

12 $i$

**工作节点开始计算** $\nabla f\left(x^k\right.$**;** $\left.\xi_i^k\right)$**13: 结束 if 14: 结束 while**

请注意，可以更高效地实现这一点——无需等待

等待过时梯度到达并将其丢弃后，服务器可以终止（第1.5节介绍的另一种可能动作）正在使用过时模型的工作节点的计算。

尽管 RingmasterASGD 达到了与同步算法 RennalaSGD 相同的理论时间复杂度（至多相差常数因子），但在实践中通常更快，主要有两个原因：

- **避免同步浪费**。 在 RennalaSGD 中，每次模型更新后，所有设备都必须与服务器同步，并丢弃仍在进行的部分计算。RingmasterASGD 通过只同步梯度已过时的工作节点，避免了这种浪费。

- **更频繁地更新**。 RennalaSGD 在收集到 $B$ 个梯度后才执行一次更新，而 RingmasterASGD 则在收到新梯度时立即更新模型。这使其进展更快，尤其是在稀疏模型中，因为每个工作节点的梯度只影响一小部分参数，且各工作节点之间的重叠有限。

我们通过第2.6节中呈现的玩具模拟验证了这一改进。本章节基于Maranjyan 等人 (2025d) 的工作：

> Artavazd Maranjyan、Alexander Tyurin 和 Peter Richt´arik。
> Ringmaster ASGD：首个具有最优时间复杂度的异步随机梯度下降算法。
> 发表于国际机器学习会议，2025d

### 1.9.2 章节3—— Ringleader ASGD：首个在数据异质性下具有最优时间复杂度的异步随机梯度下降方法

本章将从同质环境转向更一般、也更具挑战性的异构数据情形。这种环境在联邦学习等应用中尤为重要，因为数据分布在许多客户端（例如移动设备或组织）上，而且每个本地数据集都可能服从不同的分布。此外，这些数据集通常是私有的，由于隐私约束，无法共享或集中存储。因此，在联邦学习中，异质性是常态，而非例外。

现在我们来讨论为什么这种环境比同质环境更困难。回想一下，我们的目标是最小化 $n$ 局部函数的平均值：

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \left\{ f(x) \coloneqq \frac{1}{n}\sum_{i=1}^{n} f_i(x) \right\}.$$

假设我们在这种环境下应用朴素异步随机梯度下降算法（算法1.7.3）。每当收到来自工作节点的梯度，服务器就会更新模型。然而，在异质情形下，这些梯度对应于不同的局部函数 $f_i$，因此可能无法准确代表全局目标函数 $f$。因此，朴素异步更新可能会引入偏差，导致训练不稳定。

为解决这一问题，针对数据异质性下异步优化的先前研究通常会对局部目标函数之间的相似性假设 $f_i$

以及全局目标函数 $f$（例如，参见（Mishchenko 等人，2022a；Koloskova 等人，2022年；Nguyen 等人，2022年；Islamov 等人，2024年））。然而在实践中，这类假设很少成立——尤其是在联邦学习中，不同用户或设备的数据可能存在显著差异（例如个性化文本或图像数据）。

在这种异构数据环境下，唯一已知能够在不作任何相似性假设的情况下达到最优时间复杂度的方法，是同步方法MaleniaSGD （Tyurin & Richt´arik，2024年），这与同构情形颇为相似。因此，本章的目标是设计一种异步方法，在无需任何相似性假设的情况下达到最优时间复杂度。我们希望这种异步 方法在达到与同步方法MaleniaSGD相同的最优时间复杂度的同时，还能通过避免同步造成的计算浪费，在实践中表现得更好。

我们通过利用第1.5节所述的服务器另一种可能的动作来实现这一点——服务器能够存储收到的梯度，而不是立即应用它们。具体而言，我们维护一个梯度表，其条目对应于每个工作节点最近计算出的梯度。每次模型更新时，服务器都会对所有已存储梯度求平均，从而同时纳入所有工作节点的信息。这一策略消除了对相似性假设的需求，因为更新采用了对整个系统的均衡视角。

这一思路基于维护并聚合已存储梯度的概念，类似于(增量聚合梯度) (Blatt 等人，2007年；Gur- buzbalaban 等人，2017年；Vanli 等人，2018年) 方法，以及随机平均梯度(SAG) (Roux 等人，2012年；Schmidt 等人，2017年) 方法。近期，IAG的一个异步扩展版本被提出，称为IA$^2$随机梯度下降，提出者为Wang 等人(2025年)，但其算法未能达到最优时间复杂度。

虽然梯度表的思路消除了对相似性假设的需求，但要实现最优时间复杂度仍需格外谨慎。与同质情形一样，主要挑战在于控制陈旧性。然而在这里，陈旧性是由于某些工作节点长时间未发送随机梯度，导致梯度表中的部分条目可能一直未得到更新。如果多次迭代都没有刷新这些条目，相应的梯度就会变得陈旧，从而降低性能。

为解决这一问题，我们修改了更新机制，避免每接收到一个随机梯度就执行一次模型更新。相反，我们将更新组织成轮次。在每一轮中，服务器执行 $n$ 次更新 —— 每个工作节点一次 —— 并将相应的新模型发送给每个工作节点。这确保在下一轮中，工作节点计算的梯度都基于最新模型。在下一轮开始前，服务器清空梯度表，并开始收集基于这些更新后模型计算的新梯度。这种设计确保延迟保持有界，从而得到一种最优方法。我们将由此产生的算法称为RingleaderASGD——这是RingmasterASGD的自然继承者。

我们的算法有一个很好的特性：它从不丢弃任何梯度，所有接收到的梯度都会存储在表中，并最终用于模型更新。相比之下，最优的同步方法MaleniaSGD （Tyurin & Richt´arik，2024年）由于同步机制会丢弃一些工作节点的计算结果。因此，尽管 RingleaderASGD 实现了相同的最优时间复杂度（忽略常数因素），但在实践中可能更快——我们在实验部分的

章节部分（第3.7节）中展示了这一结果。本章节基于Maranjyan& Richt´arik （2025年）的研究：

> Artavazd Maranjyan 和 Peter Richt´arik。《Ringleader ASGD：首个在数据异质性条件下具有最优时间复杂度的异步随机梯度下降》。*arXiv* 预印本 *arXiv:2509.22860*，2025年

### 1.9.3 第4 章节 – ATA：用于分布式机器学习的高效资源管理的自适应任务分配

本章首先指出，在同质数据设置下，最优的并行方法——RennalaSGD （Tyurin 和 Richt´arik，2024年）和RingmasterASGD——可能会造成大量资源浪费。这些方法的目标纯粹是缩短训练时间，因此会始终利用所有可用资源。然而，这往往会导致冗余计算：较慢机器完成的大量工作可能永远不会被用到。我们将说明，只要稍微放宽最小化实际训练时间这一目标，就能大幅减少资源浪费。节省下来的资源随后可以分配给其他作业或任务。

#### 浪费从何而来

为说明这一点，我们来看RennalaSGD，它是一个很合适的示例。在每次迭代中，服务器必须恰好收集 $B$ 个梯度，才能执行模型更新（见算法1.7.4）。所有 $n$ 个工作节点都会持续计算梯度，直到收到 $B$ 个梯度。一旦 $B$ 个梯度到达，服务器就会更新模型，并丢弃所有正在进行的计算。

例如，假设有 $n = 1,\,010$ 个工作节点和 $B = 10$。在这种情况下，至少有 1，000 个工作节点的计算会被丢弃——因为服务器会在收到最先到达的 10 个梯度后进行同步。理想情况下，我们会避免向那些速度较慢、其结果最终会被丢弃的工作节点分配任务。然而，这就需要预先知道每个工作节点计算其梯度所需的时间。如果这些计算时间是固定且已知的，我们只需选择最优的工作节点子集，使总计算时间最短。但我们考虑的是一个更具挑战性的设定：计算时间是随机变量。现在让我们将这一想法形式化。

#### 问题公式化

我们考虑一个由 $n$ 个工作节点组成的系统，每个工作节点都能够重复执行相同的任务（例如，计算随机梯度）。每一轮中，算法有 $B$ 个任务的预算，可分配给各个工作节点。向一个工作节点分配一个单位的预算，相当于给它分配一个任务。

我们将所有可行分配的集合定义为

$$\mathcal{A} := \left\{ a \in \mathbb{Z}_+^n : \sum_{i=1}^n a_i = B \right\},$$

其中， $a = (a_1, \ldots, a_n)$ 表示如何将 $B$ 个任务分配给 $n$ 个工作节点——也就是说，第 $i$ 个工作节点接收 $a_i$ 个任务； $\mathbb{Z}_+$ 表示非负整数集。

## 固定计算时间

在（1.5）中描述的固定时间模型下，每个工作节点 $i$每项任务需要 $\tau_i$ 秒，接收 $B$ 个已完成梯度（分配方案为$a \in \mathcal{A}$ ）所需的时间为

$$C(a) := \max_{i \in \{1,\ldots,n\}} a_i \tau_i \ .$$

因此，可以通过求解组合问题来找到最优分配。 m

$$a^* \in \arg\min_{a \in \mathcal{A}} C(a) \ .$$

此分配方案达到的实际运行时间与算法1.7.4中使用的贪心异步收集策略相同，但总计算量最小，从而避免了任何无用功。

## 随机计算时间

然而，在实践中，工作器计算时间并非固定不变，可能会在训练过程中发生变化。我们通过假设工作节点 $i$ 的计算时间服从未知分布 $\nu_i$来对此建模。在轮次 $k$中，工作节点 $i$ 完成第 $u$ 项任务所需的时间记为 $X^{k,u}_i$ ，其中 $X^{k,u}_i \sim \nu_i$ 在 $u$ 上独立同分布。如果工作节点 $i$ 在轮次 $k$ 中执行 $a_i^k$ 项任务，则该工作节点的总计算时间为

$$\sum_{u=1}^{a_i^k} X_i^{k,u}$$

约定当 $a_i^k = 0$ 时，此和等于 0。

因此，在分配 $a^k$ 下收集 $B$ 个已完成的梯度所需的时间为

$$C(a^k) := \max_{i \in \{1,\ldots,n\}} \sum_{u=1}^{a_i^k} X_i^{k,u} \ .$$

经过 $K$ 轮次后，总期望计算时间为

$$\mathcal{C}_K := \mathbb{E}\left[\sum_{k=1}^{K} C(a^k)\right] = \sum_{k=1}^{K} \mathbb{E}\left[C(a^k)\right] \ .$$

如果已知分布 $\nu_i$ ，则可通过求解以下问题得到最优分配

$$a^* \in \arg\min_{a \in \mathcal{A}} \ \mathbb{E}\left[C(a)\right] \ ,$$

随后在每一轮中都使用相同的分配。我们将这一策略称为最优固定任务分配（OFTA）。

## 目标：无需了解分布的自适应策略

在实际应用中，分布 $\nu_i$ 是未知的。我们的目标是设计一种自适应分配策略，使其总计算时间 $\mathcal{C}_K$ 满足

$$\mathcal{C}_K \leq c \cdot \mathcal{C}_K^* + \mathcal{E}_K \ , \tag{1.8}$$

其中， $c \geq 1$ 是一个接近 1 的常数，而当 $K \to \infty$ 时，与$\mathcal{C}_K^*$ 相比， $\mathcal{E}_K$ 是一个可忽略项。这一保证表明，从渐近角度看，我们的自适应方法表现几乎与了解所有 $\nu_i$的最优策略一样好。

## ATA算法

我们通过算法自适应任务分配（ATA）实现这一点。该算法将问题表述为一个非线性随机多臂老虎机（MAB）问题（Lattimore & Szepesv´ari，2020年）。在标准的多臂老虎机设定中，智能体会反复从若干个“臂”（此处指工作节点）中进行选择，每个臂都对应一个未知的随机结果。智能体必须在两个目标之间取得平衡：

- **探索：**学习每个臂的分布。
- **利用：**优先选择当前被认为能带来更好结果（在我们的场景中即速度更快的工作节点）的臂。

在多臂老虎机文献中，这种平衡通常通过使用置信界来估计每个臂的均值奖励（或成本）实现。具体而言，基于下置信界（LCB）的算法使用悲观估计来选择臂，在确保充分探索的同时最小化遗憾（Auer，2002年）。

遵循这一原则，ATA 为每个工作节点的预期计算时间维护下置信界，并根据观测到的样本对其进行更新。在每次迭代中，给定下置信界向量 $s^k = (s_1^k, \ldots\ldots, s_n^k)$ 和总预算 $B$，该算法通过求解以下问题来选择下一次分配

$$a^k \in \arg\min_{a \in \mathcal{A}} \max_{i \in \{1,\ldots,n\}} a_i^k s_i^k \ .$$

该规则以一种有原则的方式平衡探索与利用，并给出了（1.8）中所述的性能保证。

## 异步情形

由于RingmasterASGD 是Rennala SGD的异步版本，因此ATA 算法也可以应用于其之上。在此设置下，ATA 计算出的分配决定每个工作节点贡献多少个梯度；但服务器不会等待收集完所有梯度后再更新模型，而是在接收到每个梯度后立即执行模型更新。同时，服务器通过向本地副本在此过程中变得过时的工作节点发送最新模型，确保延迟保持在有界范围内。

本章节基于Maranjyan 等人（2025b）：

Artavazd Maranjyan、El Mehdi Saad、Peter Richt´arik 和 Francesco Orabona。ATA：用于分布式机器学习中高效资源管理的自适应任务分配。发表于国际机器学习会议，2025b

## 1.10 未纳入本学位论文的出版物

为保持叙述连贯、重点突出，我没有将博士期间撰写的几篇论文纳入本学位论文。为求完整，我也列出了博士阶段之前发表的出版物。所有这些作品均汇总于表 1.2。

表 1.2：未纳入本学位论文的出版物。

| 参考文献与标题 | 博士期间 |
|---|---|
| Maranjyan 等人 （2025a）<br>MindFlayer SGD：在存在<br>异质且随机的工作节点计算时间时实现高效并行随机梯度下降 | ✔ |
| Condat 等人(2025年)<br>LoCoDL：通信高效分布式学习<br>与本地训练和压缩 | ✔ |
| Maranjyan 等人 (2024年)<br>差分隐私随机块坐标下降 | ✔ |
| Maranjyan 等人 （2025c）<br>GradSkip：通信加速的局部梯度方法<br>具有更低计算复杂度 | ✘ |
| Grigoryan 等人 （2023年）<br>发散集的 Menshov 型定理<br>局部化算子序列 | ✘ |
| Grigoryan 和 Maranjyan (2021a)<br>一般 Haar 系统中傅里叶级数的发散 | ✘ |
| Grigoryan 和 Maranjyan (2021年b)<br>$L^1$ 中 Faber-Schauder 级数的无条件收敛 | ✘ |

在这些研究中，我想简要介绍一项相关工作。在Maranjyan 等人（2025a）中，我们研究了RennalaSGD （Tyurin & Richt´arik，2024年）——齐次数据设置下的最优同步随机梯度下降算法——并假设每个工作节点的计算时间是一个随机变量。我们证明，当工作节点计算时间的分布具有重尾时，Ren-nalaSGD 可能会显著变慢。为了解决这一问题，我们提出了一种新算法 MindFlayerSGD，通过对每个工作节点计算其梯度所允许的最长时间设置阈值，减轻极端延迟的影响。我们的分析表明，MindFlayerSGD 始终优于RennalaSGD，并且随着这些分布的偏度增大，可以实现任意大的加速比。

## 1.11 常用符号

本节总结了本学位论文中通用的数学符号。

### 基本集合

我们用 $\mathbb{R}$ 表示实数集，用 $\mathbb{R}_+ := [0, \infty)$ 表示非负实数集。自然数集记为 $\mathbb{N} := \{1, 2, \dots\}$，非负整数集记为 $\mathbb{Z}_+ := \{0, 1, 2, \dots\}$。对于正整数 $n$，我们使用简写记号 $[n] := \{1, 2, \dots, n\}$。

### 向量与范数

设 $d \in \mathbb{N}$ 表示向量的维数。对于向量 $a,\ b \in \mathbb{R}^d$，标准内积定义为

$$\langle a, b \rangle := \sum_{i=1}^{d} a_i b_i \ ,$$

其中 $a_i$ 和 $b_i$ 是坐标，相应的欧几里得范数为

$$\|a\| := \sqrt{\langle a, a \rangle} \ .$$

我们还分别使用 $\|\cdot\|_1$ 和 $\|\cdot\|_\infty$ 表示 $\ell_1$ 和 $\ell_\infty$ 范数：

$$\|a\|_1 := \sum_{i=1}^{d} |a_i| \ , \qquad \|a\|_\infty := \max_{1 \le i \le d} |a_i| \ .$$

符号 $\odot$ 表示两个向量的逐元素（Hadamard）乘积。对于任意标量 $x \in \mathbb{R}$，我们将 $(x)_+ := \max\{x, 0\}$记作：

对于整数 $a, b \in \mathbb{Z}$，我们使用 $a \bmod b$ 表示 $a$ 除以 $b$所得的余数，即

$$a \bmod b := a - b \left\lfloor \frac{a}{b} \right\rfloor \ .$$

### 概率与期望

关于随机变量的期望记作 $\mathbb{E}[\cdot]$。对于事件$\mathcal{E}$，我们用 $\neg\mathcal{E}$ 表示其补集（即 $\mathcal{E}$ 不发生的事件）。我们用 $\mathbf{1}(\cdot)$表示示性函数：当括号内的条件为真时取值为 1，否则取值为 0。

### 渐近记号

对于函数 $\phi,\ \psi : \mathcal{X} \to \mathbb{R}$，我们使用标准渐近符号：

- $\phi = \mathcal{O}(\psi)$ 表示存在一个常数 $C > 0$ 使得

$$\phi(x) \le C\, \psi(x), \quad \forall x \in \mathcal{X}.$$

- $\phi = \Omega(\psi)$ 表示存在一个常数 $C > 0$ 使得

$$\phi(x) \geq C\,\psi(x), \quad \forall x \in \mathcal{X}.$$

- $\phi = \Theta(\psi)$表示 $\phi = \mathcal{O}(\psi)$和 $\phi = \Omega(\psi)$均成立，即 $\phi$ 和 $\psi$ 的阶相同，至多相差一个常数因子。

# 第 2 章

## Ringmaster ASGD：首个具有最优时间复杂度的异步随机梯度下降

本章节基于Maranjyan 等人（2025d）的研究：

> Artavazd Maranjyan、Alexander Tyurin 和 Peter Richt´arik。《Ringmaster ASGD：首个具有最优时间复杂度的异步随机梯度下降》。发表于国际机器学习会议，2025d

## 2.1 引言

我们考虑如下形式的随机非凸优化问题

$$\underset{x\in\mathbb{R}^d}{\text{minimize}}\left\{f(x) \coloneqq \mathbb{E}_{\xi\sim\mathcal{D}}\left[f(x;\xi)\right]\right\},$$

其中 $f:\ \mathbb{R}^d \times \mathbb{S}_\xi \to \mathbb{R}$，$\mathbb{R}^d$ 是一个线性空间，而 $\mathbb{S}_\xi$ 是一个样本空间。在机器学习中，$f(x;\xi)$ 表示由参数 $x$ 确定的模型在数据样本 $\xi$上的损失，而 $\mathcal{D}$ 表示训练数据集的分布。在非凸优化中，我们的目标是找到一个 $\varepsilon$–驻点，即一个（随机）向量$x \in \mathbb{R}^d$，使得 $\mathbb{E}[\|\nabla f(x)\|^2] \leq \varepsilon$。

我们考虑一种涉及 $n$个工作节点（例如 CPU、图形处理器、服务器）的设置，每个工作节点都可以访问相同的分布 $\mathcal{D}$。每个工作节点都能够计算相互独立、无偏且方差有界的随机梯度（假设2.1.3）。我们考虑计算速度异步、异质且不断变化的设置，并力求涵盖所有可能的情形，例如随机故障、计算性能随时间变化，以及存在速度较慢或拖后的工作节点（Dean 与 Barroso，2013年）。

这种设置常见于数据中心环境（Dean 等人，2012年）和用于分布式训练的联邦学习中（Kone˘cn´y 等人，2016b；McMahan 等人，2016年；Kairouz 等人，2021年）。尽管并行有助于快速收敛，但工作节点速度的差异使有效协调更具挑战性。

异步随机梯度下降是这类分布式环境中一种常用的并行化方法。我们从一个简单的变体开始，该变体让所有工作节点持续处于活动状态，并在梯度到达时执行更新；我们将此方法称为朴素异步随机梯度下降。该方法概述于算法2.1.1。

表 2.1：异步随机梯度方法的时间复杂度。该方法执行步骤 $x^{k+1} = x^k - \gamma^k \nabla f(x^{k-\delta^k}; \xi_{i^k}^{k-\delta^k})$，以在非凸情形下获得一个 $\varepsilon$–驻点。在此表中，我们考虑第 2.2 节中的固定计算模型。缩写：对于所有 $x \in \mathbb{R}^d$，$\sigma^2$ 定义为 $\mathbb{E}_\xi[\|\nabla f(x; \xi) - \nabla f(x)\|^2] \leq \sigma^2$；$L$ 是 $f$ 的光滑常数；$\Delta := f(x^0) - f^*$；$\tau_i \in [0, \infty]$ 是工作节点 $i^k$ 计算单个随机梯度所需的时间上界。

| 算法 | 最坏情况时间复杂度 | 最优的 | 适应于不同的速度 |
|---|---|---|---|
| 延迟自适应异步随机梯度下降 (Koloskova 等人，2022年) (Mishchenko 等人，2022a) | $\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{n\varepsilon^2}\right)$ | ✘ | ✔ |
| 朴素最优异步随机梯度下降 **（新）** （算法 2.2.1；定理 2.2.1） | $\min_{m\in[n]}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{m\varepsilon^2}\right)$ | ✔ | ✘ |
| RingmasterASGD **（新）** （算法2.3.1 或 2.3.2；定理 2.4.3） | $\min_{m\in[n]}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{m\varepsilon^2}\right)$ | ✔ | ✔ |
| 下界 (Tyurin &Richt´arik, 2024年) | 最小值 $\min_{m\in[n]}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{m\varepsilon^2}\right)$ | — | — |

**算法 2.1.1** 朴素异步随机梯度下降

1: **输入：** 点 $x^0 \in \mathbb{R}^d$，步长 $\gamma^k \geq 0$2: 工作节点开始计算随机梯度，时间为 $x^0$3: **对于** $k = 0, \ldots\ldots, K-1$ **执行**4: 来自工作节点 $\nabla f\left(x^{k-\delta^k}\right.$ 的梯度到达；$\left.\xi_{i^k}^{k-\delta^k}\right)$ $i^k$5: 更新：$x^{k+1} = x^k - \gamma^k \nabla f\left(x^{k-\delta^k}; \xi_{i^k}^{k-\delta^k}\right)$6: 工作节点 $i^k$ 开始计算 $\nabla f\left(x^{k+1}; \xi_{i^k}^{k+1}\right)$7: **结束循环**

这是一种贪心的异步方法。工作节点完成随机梯度的计算后，会立即将梯度发送给服务器，服务器无需等待其他工作节点，便会更新当前迭代点。请注意，与标准的随机梯度下降不同，更新使用的是在点 $x^{k-\delta^k}$ 处计算得到的随机梯度，其中索引 $k - \delta^k$ 对应于工作节点开始计算梯度时的迭代，这可能已经明显过时。序列 $\{\delta^k\}$ 是异步随机梯度下降的延迟序列，其中 $\delta^k \geq 0$ 定义为工作节点 $i^k$ 开始计算梯度时的迭代与应用该梯度时的迭代 $k$ 之差。

异步随机梯度下降方法历史悠久，始于1986年（Tsitsiklis 等人，1986年），并随着 Recht 等人（2011年）的开创性工作而重新受到关注。异步随机梯度下降的核心思想很简单：为实现快速收敛，充分利用所有可用资源，让所有工作节点始终保持忙碌。大量研究验证了这一原则，表明异步随机梯度下降能够优于简单的同步随机梯度下降方法（Feyzmahdavian 等人，2016年；Dutta 等人，2018年；Nguyen 等，2018；Arjevani 等人，2020年；Cohen 等人，2021年；Mishchenko 等人，2022a；Koloskova 等人，2022年；Islamov 等人，2024年；Feyzmahdavian 和 J ohansson，2023年）。

### 2.1.1 假设

I在本文中，我们考虑非凸世界中的标准假设 d.

**假设 2.1.1**（光滑性）。函数 $f$ 可微，且其梯度是 Lipschitz连续的。也就是说，存在一个常数 $L > 0$，使得对于所有 $x$，$y \in \mathbb{R}^d$，

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\| .$$

该假设通常称为 $L$-光滑性，它控制了梯度变化的快慢。

**假设 2.1.2**（下有界性）。目标函数下有界：存在 $f^* > -\infty$，使得对所有 $x \in \mathbb{R}^d$，$f(x) \geq f^*$。

该假设确保优化问题是适定的：如果没有这一假设，函数值可能无限下降，从而使收敛保证失去意义。

我们定义 $\Delta := f(x^0) - f^*$，其中 $x^0$ 是优化算法的初始点。量 $\Delta$ 表示初始次优性，可作为衡量问题内在难度的指标。

**假设 2.1.3.** 对于每个 $\xi$，函数 $f(x; \xi)$ 关于其第一个参数 $x$ 可微。此外，随机梯度是无偏的，且具有有界方差 $\sigma^2 \geq 0$，即，

$$\mathbb{E}_\xi \left[\nabla f(x; \xi)\right] = \nabla f(x), \quad \forall x \in \mathbb{R}^d ,$$
$$\mathbb{E}_\xi \left[\|\nabla f(x; \xi) - \nabla f(x)\|^2\right] \leq \sigma^2, \quad \forall x \in \mathbb{R}^d .$$

### 2.1.2 相关工作

尽管多年来人们提出了各种异步 随机梯度下降算法，但一个根本问题仍未解决：在这种情形下，并行化的最优策略是什么？

当只有一个工作节点时，找到一个 $\varepsilon$–驻点所需的随机梯度数量的最优值为

$$\Theta\left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{\varepsilon^2}\right),$$

由标准的随机梯度下降方法实现（Ghadimi 与 Lan，2013年；Arjevani 等人，2022年）。在拥有多个工作节点的并行设置中，已有多种方法被提出以获得预言机下界（Scaman 等人，2017年；Woodworth 等人，2018年；Arjevani 等人，2020年；Lu 与 De Sa，2021年）。Tyurin 与 Richt´arik（2024年）；Tyurin（2024年）的近期研究通过在固定计算模型和通用计算模型下建立异步方法的时间复杂度下界，探讨了这一问题。令人惊讶的是，他们证明现有的异步随机梯度下降方法无一是最优的。此外，他们还提出了

**算法 2.1.2** RennalaSGD （Tyurin & Richt´arik，2024年）

1：输入：点 $x^0 \in \mathbb{R}^d$，步长 $\gamma > 0$，批量大小 $B \in \mathbb{N}$2：工作节点开始计算随机梯度，计算位置为 $x^0$3：对于 $k = 0, \ldots, K-1$，执行 4：初始化 $g^k = 0$ 和 $b = 0$5：当 $b < B$ 时执行

6 $\nabla f\left(x^{k-\delta^{k\,b}}\ \xi^{k\,b}\right)$ $i^{k\,b}$：来自工作

7 $\delta^{k\,b} = 0$

节点 的梯度 , ; , 到达：若 , 则 8：$g^k = g^k + \nabla f\left(x^{k-\delta^{k\,b}},;\ \xi^{k,b}\right)$9：结束 if 10：工作节点 $i^{k\,b}$ 立即开始在 $x^k$处计算新的梯度 11：$b = b + 1$12：结束 while 13：更新模型：

$$x^{k+1} = x^k - \gamma \frac{g^k}{B}$$

14：**结束循环**

一种极小极大最优方法——RennalaSGD，其时间复杂度达到了理论下界。

RennalaSGD 是 半异步的 ， 可以视为 小批量随机梯度下降 （采用同步 迭代/模型更新）与异步 小批量收集机制的结合。现在，我们使用算法 RennalaSGD 的记号来解释其工作原理。算法 2.1.2，这有助于将其与本文进一步介绍的方法进行比较。由于条件 $\delta^{k,\,b} = 0$忽略了在先前点计算的所有随机梯度，RennalaSGD 执行以下步骤

$$x^{k+1} = x^k - \gamma \frac{1}{B} \sum_{j=1}^{B} \nabla f\left(x^k; \xi^{k,j}\right) \ ,$$

其中， $\xi^{k,\,1}$，……， $\xi^{k,\,b}$是从 $\mathcal{D}$中收集的独立样本，由所有工作节点异步地收集。注意，工作节点在相同的 点 $x^k$处计算随机梯度，其中工作节点 $i$ 计算 $B_i \geq 0$ 个梯度，使得 $\sum_{i=1}^{n} B_i = B$。

这种方法至少有两个根本缺陷：

- 最快的工作节点完成第一个随机梯度的计算后， $\nabla f\left(x^k; \xi^{k,\,1}\right)$，它会在同一点 $x^k$开始计算另一个随机梯度，尽管它已从 $\nabla f\left(x^k; \xi^{k,\,1}\right)$获得了额外信息。Rennala SGD 不会立即更新迭代点 $x^k$。

- RennalaSGD 完成内部 while 循环后，会忽略循环结束前正在计算的所有随机梯度，即使某个工作节点只是在片刻之前才开始计算。相比之下，朴素异步随机梯度下降 在要求工作节点计算下一个随机梯度时，会充分利用当前所有可用信息，并且不会忽略任何随机梯度，从而避免这些问题。

这些发现引出了一个耐人寻味的问题：异步并行化从根本上来说存在缺陷吗？如果最优的解决方案在于同步方法，

那么，社区是否应该放弃异步随机梯度下降，转而专注于开发同步方法？也许，人们对异步随机梯度下降 方法的广泛热情本就不合时宜。

或者，是否存在尚未发现的异步随机梯度下降 变体，能够实现最优时间复杂度？在本研究中，我们对这个问题给出了肯定的回答。我们提出了一种新颖的异步优化方法，实现了最优时间复杂度，从而重新确立了异步随机梯度下降 的重要地位。

### 2.1.3 贡献

我们的贡献总结如下：

- 我们提出了一种新颖的异步随机梯度下降方法，RingmasterASGD，详见算法 2.3.1 和算法 2.3.2。这是首个在工作节点计算时间任意异构的情况下实现最优时间复杂度的异步方法（见表 2.1）。具体而言，在定理 2.4.3 和 2.5.2中，我们证明了其时间复杂度达到 Tyurin & Richt´arik (2024年)；Tyurin (2024年)提出的下界。

- 我们的研究从另一种新的最优方法 朴素最优异步随机梯度下降（算法 2.2.1）开始。我们证明，朴素最优异步随机梯度下降 在固定计算模型下能够达到最优性。然而，我们发现该方法过于简单，在工作节点计算时间混乱且动态变化的场景中缺乏稳健性。为解决这一局限，我们设计了 Ringmaster ASGD，结合了 朴素最优异步随机梯度下降、先前非最优版本的 朴素异步随机梯度下降方法（Cohen 等人，2021年；Koloskova 等人，2022年；Mishchenko 等人，2022a），以及半同步的 RennalaSGD （Tyurin & Richt´arik，2024年）。

- 我们的所有主张均得到严谨理论分析的支持，该分析表明，本方法在几乎任何计算场景下都具有最优性，包括不可预测的停机时间、计算性能随时间波动、缓慢或拖后工作节点造成的延迟，以及在分布式系统中维持同步所面临的挑战（见第2.4 节和2.5节）。通过数值实验，我们证明 RingmasterASGD优于现有方法（见第2.6节）。

## 2.2 预备知识与朴素方法

为了比较各种方法，我们考虑固定计算模型（Mishchenko 等人，2022a）。在该模型中，假设

$$\begin{aligned}&\text{Each worker } i \text{ requires } \tau_i \text{ seconds to compute one stochastic gradient } \nabla f(x;\xi).\\&\text{Without loss of generality, we assume}\\&\qquad 0 < \tau_1 \leq \tau_2 \leq \cdots \leq \tau_n\ .\end{aligned} \tag{2.1}$$

然而，在第 2.5节中，我们将讨论如何轻松地将我们的结果推广到任意计算动态，例如，当计算时间不受固定值 $\{\tau_i\}$的限制，并且可以以任意/混沌的方式变化时

在时间上。在固定计算模型下，Tyurin & Richt´arik (2024年) 证明了最优时间复杂度下界为

$$T_{\mathrm{R}} := \Theta\left(\min_{m\in[n]}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{m\varepsilon^2}\right)\right) \tag{2.2}$$

秒，由 RennalaSGD （算法 2.1.2）。然而，对 朴素异步随机梯度下降 （Koloskova 等人，2022年；Mishchenko 等人，2022a）采用适当步长的分析最为出色，可达到相应的时间复杂度（见 Tyurin &Richt´arik （2024年）中的 L 节）

$$T_{\mathrm{A}} := \Theta\left(\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{n\varepsilon^2}\right)\right). \tag{2.3}$$

注意， $T_{\mathrm{R}} \leq T_{\mathrm{A}}$；这是因为对于任意函数$g$： $\mathbb{N}\to\mathbb{R}$，$\min^{m\in[n]} g(m) \leq g(n)$。此外，$T_{\mathrm{R}}$ 可以任意更小。为说明这种差异，考虑一个示例，其中对所有 $i\in[n]$，$\tau_i=\sqrt{i}$ 。那么，

$$T_{\mathrm{R}} = \Theta\left(\max\left\{\frac{\sigma L\Delta}{\varepsilon^{3/2}}, \frac{L\Delta\sigma^2}{\sqrt{n}\varepsilon^2}\right\}\right)$$

and

$$T_{\mathrm{A}} = \Theta\left(\max\left\{\frac{\sqrt{n}L\Delta}{\varepsilon}, \frac{L\Delta\sigma^2}{\sqrt{n}\varepsilon^2}\right\}\right),$$

推导见 A.5 节。如果 $n$ 很大（这在现代大规模训练场景中很常见）， $T_{\mathrm{A}}$ 可能任意大于 $T_{\mathrm{R}}$。因此，已知的最佳异步随机梯度下降变体无法适应工作节点数量庞大且计算时间存在异质性／混乱性的情形。

### 2.2.1 一种朴素的最优异步随机梯度下降

现在，我们介绍第一个简单有效的策略，以改进时间复杂度 $T_{\mathrm{A}}$。具体而言，我们假设，在优化过程开始时选择一部分工作节点，而非使用所有可用的工作节点，可以带来更高效、更稳定的方法。正如我们将要展示的，这一调整不仅简化了计算动态，而且足以实现最优时间复杂度。

其理念是选择速度最快的 $[m] := \{1,2,\ldots,m\}$ 个工作节点，从而忽略较慢的工作节点并消除延迟的梯度更新。我们证明，最优的算法包括找出理想的工作节点数量 $m$ ，并在这些工作节点上运行 朴素异步随机梯度下降 （Koloskova 等人，2022年；Mishchenko 等人，2022a）的延迟自适应版本。该方法在算法 2.2.1中进行了形式化描述。

**算法 2.2.1** 朴素最优异步随机梯度下降

1: 查找

$$m_\star \in \arg\min_{m\in[n]} \left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(1+\frac{\sigma^2}{m\varepsilon}\right)$$

2: 在 个工作节点上运行朴素异步随机梯度下降 (算法 2.1.1) $[m_\star]$ 个工作节点

在算法 $m_\star$ 中选择 2.2.1实际上只选择速度最快的 $m_\star$ 个工作节点。注意， $m_\star$ 可能等于 $n$，这意味着所有工作节点都参与其中；当所有工作节点的速度几乎相同时，就会出现这种情况。在这种情况下，如果所有 $\tau_i$都相等，算法 2.2.1 第 1 行中的调和平均数保持不变，但右侧会减小。然而，如果某些工作节点遭遇较大延迟，随着 $m$ 增大，调和平均数也会增大，从而在这两个因素之间形成权衡。反过来，如果大多数工作节点都很慢，那么最优选择可能是只让一个工作节点参与。

接下来，我们可以轻松证明，我们的算法 朴素最优异步随机梯度下降 (算- 法 2.2.1)，在时间复杂度方面是最优的。

**定理 2.2.1**。 考虑固定计算模型 (2.1)。假设2.1.1、2.1.2和2.1.3 均成立。则 朴素最优异步随机梯度下降 (算法 2.2.1)，使用$m_\star$ 个工作节点时，可达到最优时间复杂度（2.2）。

证明。 证明很直接。事实上，算法2.3）的时间复杂度2.1.1 使用 $m_*$ 个工作节点时为

$$\Theta\left(\left(\frac{1}{m_*}\sum_{i=1}^{m_*}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{m_*\varepsilon^2}\right)\right),$$

由于算法2.2中 $m_*$ 的定义，这与（2.2.1）相等。 □

据我们所知，这是首个能够保证达到最优时间复杂度的异步随机梯度下降 变体。

## 2.2.2 为什么算法 2.2.1被称为“朴素”?

请注意，确定最优的 $m_\star$ 需要知道工作节点的计算时间 $\tau_1$、……、$\tau_n$。如果工作节点的计算时间确实始终固定，这就不是问题，因为算法运行前，可以向每个工作节点查询一个梯度来获取这些时间。然而，在实际系统中，计算时间很少保持不变，而且可能随着迭代而变化（Dean 与 Barroso，2013年；Chen 等人，2016年；Dutta 等人，2018年；Maranjyan 等人，2025a），甚至有时会变为无穷大，这表示系统处于停机状态。

因此，在方法开始时简单地选择速度最快的 $m_*$ 工作节点，并始终保持这一选择不变，可能会在实践中导致严重问题。计算环境可能表现出对抗性行为，即工作节点速度会随时间发生变化。例如，起初第一个工作节点可能是速度最快的，而最后一个工作节点则是最慢的。在这种情况下，朴素最优异步随机梯度下降会排除最慢的工作节点。然而，随着时间推移，它们的性能

可能会发生反转，导致最初选中的 $m_*$ 工作节点（包括第一个工作节点）变成速度最慢的工作节点。这暴露了该策略的一个关键局限性：它无法适应随时间变化的工作节点速度。

## 2.3 Ringmaster ASGD

现在，我们可以介绍异步随机梯度下降的新版本，称为Ringmaster ASGD（算法2.3.1 和算法2.3.2），它们在无需预先知道计算时间的情况下，保证最优时间复杂度。这两种方法除了我们稍后将讨论的一个细微差异外是等价的。我们先关注算法2.3.1。

**算法 2.3.1** RingmasterASGD （不停止计算）

**1：输入：点** $x^0 \in \mathbb{R}^d$**、步长** $\gamma > 0$**、延迟阈值** $R \in \mathbb{N}$**2：设置** $k = 0$**3：工作节点从** $x^0$ **开始计算随机梯度4：当未终止时执行**5 $i$**：梯度** $\nabla f\left(x^{k-\delta^k};\ \xi_i^{k-\delta^k}\right)$ **从工作节点到达：如果则** $\delta^k < R$ **7：更新模型：** $x^{k+1} = x^k - \gamma \nabla f\left(x^{k-\delta^k};\ \xi_i^{k-\delta^k}\right)$**8：工作节点** $i$ **开始计算** $\nabla f\left(x^{k+1};\ \xi_i^{k+1}\right)$**9：更新迭代次数** $k = k + 1$**10：否则**11 $\nabla f\left(x^{k-\delta^k}\ \xi_i^{k-\delta^k}\right)$**：忽略过**12 $i$**时梯度；：工作节点开始计算** $\nabla f\left(x^k;\ \xi_i^k\right)$**13：结束条件 14：结束循环**

**算法 2.3.2** RingmasterASGD （带计算停止机制）

1: 输入：点 $x^0 \in \mathbb{R}^d$，步长 $\gamma > 0$，延迟阈值 $R \in \mathbb{N}$2: 设置 $k = 0$3: 工作节点从 $x^0$ 开始计算随机梯度4: 当尚未终止时执行 5: 停止计算延迟为 $\geq R$ 的随机梯度，并改为从 $x^k$ 开始计算新的随机梯度 6: 梯度 $\nabla f\left(x^{k-\delta^k};\right.$ 来自工作节点 $i$ 的 $\left.\xi_i^{k-\delta^k}\right)$ 到达 7: 更新模型：$x^{k+1} = x^k - \gamma \nabla f\left(x^{k-\delta^k};\ \xi_i^{k-\delta^k}\right)$8: 工作节点 $i$ 开始计算 $\nabla f\left(x^{k+1};\ \xi_i^{k+1}\right)$9: 更新迭代次数 $k = k + 1$10: 结束循环（对算法 2.1.1 的核心和必要修改已突出显示。或者，算法 2.3.1 是采用式 (2.4) 定义的特定自适应步长的算法 2.1.1）

### 2.3.1 描述

让我们比较算法 2.3.1和算法 2.1.1：二者唯一的区别（已突出显示）在于，算法 2.3.1 会忽略“非常陈旧的随机梯

度”，即在存在显著延迟的点上计算出的梯度。具体而言，算法 2.3.1 接收 $\nabla f\left(x^{k-\delta^k};\ \xi_i^{k-\delta^k}\right)$，将 $\delta^k$ 与我们的 延迟阈值 超参数 $R$进行比较。如果 $\delta^k \geq R$，那么算法 2.3.1 会完全忽略这一非常过时的随机梯度，并请求工作节点在最相关的点 $x^k$处计算新的随机梯度。

注意，RingmasterASGD （算法2.3.2）是算法2.1.1 采用以下自适应步长规则的版本：

$$\gamma^k = \begin{cases} \gamma, & \text{if} \quad \bar{\delta}_i^k < R, \\ 0, & \text{if} \quad \bar{\delta}_i^k \geq R, \end{cases}$$
$$\bar{\delta}_j^{k+1} = \begin{cases} 0, & \text{if} \quad j = i, \\ \bar{\delta}_j^k + 1, & \text{if} \quad j \neq i \quad \text{and} \quad \bar{\delta}_i^k < R, \\ \bar{\delta}_j^k, & \text{if} \quad j \neq i \quad \text{and} \quad \bar{\delta}_i^k \geq R, \end{cases} \tag{2.4}$$

其中， $i$ 是其随机梯度在迭代$k$中被应用的工作节点的索引，并且我们通过设定 $\bar{\delta}_j^0 = 0$（对所有 $j \in [n]$），来初始化虚拟延迟序列$\{\bar{\delta}_j^k\}_{k,\ j}$。

## 2.3.2 延迟阈值

回到算法2.3.1，注意我们有一个参数 $R$，称为延迟阈值。当 $R = 1$时，该算法退化为经典的随机梯度下降方法，即：

$$x^{k+1} = x^k - \gamma \nabla f\left(x^k; \xi_i^k\right) \ ,$$

因为 $\delta^k = 0$ 对所有 $k \geq 0$均成立。在这种情况下，算法会变得非常保守，忽略在较早时刻计算出的所有随机梯度 $x^{k-1}$、.....、 $x^0$。相反，如果 $R = \infty$，该方法会纳入延迟任意大的随机梯度，并成为经典的朴素异步随机梯度下降。直观而言，两者之间应当有所平衡——存在一个恰当的 $R$ 取值，可以（i）避免该方法过于保守，同时（ii）确保稳定性，确保只有信息丰富的随机梯度才用于更新模型。我们将在第2.4节提出一个最优的 $R$，以形式化地阐述这些直觉。有趣的是， $R$ 的取值并不取决于计算时间。

## 2.3.3 为什么我们忽略旧的随机梯度?

主要原因是，这样做使我们能够开发出首个达到下界的最优异步随机梯度下降方法（见2.4 和2.5节）。忽略旧梯度使我们能够建立更紧的收敛保证。直观地说，旧梯度不仅无法提供关于函数 $f$ 的额外有用信息，还可能对算法的性能产生负面影响。因此，在优化过程中忽略这些梯度，对于实现我们开发最优异步随机梯度下降 方法的目标至关重要。

## 2.3.4 与 Rennala SGD 的比较

与RennalaSGD不同，后者将同步的小批量随机梯度下降 更新与异步小批量收集策略相结合，RingmasterASGD 则完全异步，

正如我们在实验中所展示的，这带来了进一步的实际优势：

- RingmasterASGD 在收到新的且相关（即未过时太久）的随机梯度后，会立即更新模型。这种即时更新策略在实际应用中对于稀疏模型尤为有利，因为不同的梯度可能只更新模型中互不重叠的部分，从而加快处理速度。这一概念与Recht 等人（2011年）的思想一致；他们表明，无锁异步更新能够有效利用稀疏性来提升性能。

- RingmasterASGD 确保快速工作节点计算出的随机梯度绝不会被忽略。相比之下，RennalaSGD 可能会丢弃随机梯度，即使这些梯度是最近才开始计算、且能很快算出的（参见第2.1节的讨论）。

与此同时，RingmasterASGD 遵循了使Rennala SGD 达到最优的相同原则。Rennala SGD 的核心理念是通过采用异步批次收集来优先考虑快速工作节点，从而有效忽略慢速工作节点。这通过在算法2.1.2中谨慎选择批量大小 $B$ 来实现：批量大小要足够大，使快速工作节点能够完成计算；同时又要足够小，以便忽略慢速工作节点和延迟过久的随机梯度。类似地，RingmasterASGD 使用延迟阈值 $R$来实现这一理念。

总而言之，尽管从理论角度看，RennalaSGD 和RingmasterASGD 都达到了最优，但后者完全不需要同步，这直观上使其在实践中更具吸引力，也使其能够获得理论分析未能体现的额外收益。正如我们将看到的，我们的实验支持这一判断。

### 2.3.5 与以往异步随机梯度下降变体的比较

Koloskova 等人 (2022年) 和 Mishchenko 等人 (2022a) 提供了此前关于使用延迟自适应步长的 朴素异步随机梯度下降 的最先进分析。然而，正如第 2.2 节所示，他们的分析无法保证最优性。这至少有两个原因：

- 他们的方法不会丢弃过时的随机梯度，而是试图利用所有梯度，即使其延迟很大；
- 尽管他们选择的步长 $\gamma^k$ 会随延迟增加而减小，但这种调整可能不足以确保最优性能，而且与（2.4）相比，他们选择的步长可能不是最优的。

### 2.3.6 停止无关计算

如果能够停止计算，我们可以通过引入算法2.3.1 进一步改进算法2.3.2。我们不再等待工作节点完成延迟大于 $R$ 的过时随机梯度的计算（这些梯度无论如何也不会被使用），而是提出立即停止/终止这些计算，并将工作节点重新分配到最相关的点 $x^k$。这一调整

让工作节点有机会赶上进度，因为它们可能会变得更快，并在更新后的点上更高效地执行计算。

## 2.4 理论分析

我们现在可以给出RingmasterASGD 的理论分析。首先给出迭代复杂度 界：

**定理 2.4.1** （证明见第A.3节）。在假设2.1.1、2.1.2和2.1.3成立时，设 RingmasterASGD （算法 2.3.1 或算法 2.3.2）中的步长为

$$\gamma = \min\left\{\frac{1}{2RL}, \frac{\varepsilon}{4L\sigma^2}\right\}.$$

Then

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right] \leq \varepsilon\ ,$$

只要

$$K \geq \frac{8RL\Delta}{\epsilon} + \frac{16\sigma^2 L\Delta}{\epsilon^2}\ , \tag{2.5}$$

其中， $R \in \{1, 2, \ldots, \}$ 是任意延迟阈值。

朴素异步随机梯度下降的经典分析可达到相同的收敛速率，其中 $R$ 定义为 $R \equiv \max_{k\in[K]} \delta^k$ （Stich & Karimireddy，2020年；Arjevani 等人，2020年）。这一结果并不意外，因为在 Ringmaster ASGD 中设定 $R = \max_{k\in[K]} \delta^k$ 后，该算法等价于经典的朴素小批量随机梯度下降，因为没有梯度会被忽略。此外，Cohen 等（2021年）、Koloskova 等人（2022年）和 Mishchenko 等人（2022a）的分析也得到了相同的收敛速率，其中 $R = n$。然而，值得注意的是， $R$ 是 Ringmaster ASGD 中的一个自由参数，可以任意选择。虽然设定 $R = \max_{k\in[K]} \delta^k$ 或 $R = n$ 可有效恢复先前的迭代复杂度，但我们证明，存在另一种 $R$ 的选择，可实现最优的时间复杂度（见定理 2.4.3 和 2.5.2）。

必须认识到，迭代复杂度(2.5) 并不能反映算法实际的“运行时间”表现。为了选取$R$的最优值，我们必须将关注点从迭代复杂度转向时间复杂度，它衡量算法的运行时间。通过优化时间复杂度在 $R$上的表现，我们就能获得实际应用中最佳且有效的选择。为了求得时间复杂度，我们需要以下引理。

**引理 2.4.2** (证明见附录 A.1)。 设工作节点的计算时间满足固定计算模型 (2.1)。设 $R$ 为算法 2.3.1 或算法 2.3.2的延迟阈值。完成任意 $R$次连续的算法迭代点更新2.3.1 或算法2.3.2所需的时间至多为

$$t(R) := 2\min_{m\in[n]}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(1+\frac{R}{m}\right). \tag{2.6}$$

结合定理 2.4.1 和引理 2.4.2，我们给出主要结果。

**定理 2.4.3** （RingmasterASGD 的最优性）。 设假设 2.1.1、2.1.2 和 2.1.3 成立。令 RingmasterASGD （算法 2.3.1 或算法2.3.2）中的步长为

$$\gamma = \min\left\{\frac{1}{2RL}, \frac{\varepsilon}{4L\sigma^2}\right\}.$$

因此，在固定计算模型 （2.1）下，RingmasterASGD 实现了最优的时间复杂度

$$\mathcal{O}\left(\min_{m\in[n]}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{m\varepsilon^2}\right)\right) \tag{2.7}$$

其延迟阈值为

$$R = \max\left\{1, \left\lceil\frac{\sigma^2}{\varepsilon}\right\rceil\right\}. \tag{2.8}$$

注意， $R$ 的取值不以任何方式依赖于计算时间$\{\tau_1, \ldots, \tau_n\}$。

证明。 根据定理2.4.1，RingmasterASGD 的迭代复杂度为

$$K = \left\lceil\frac{8RL\Delta}{\epsilon}+\frac{16\sigma^2 L\Delta}{\epsilon^2}\right\rceil. \tag{2.9}$$

根据引理2.4.2，我们知道，RingmasterASGD 至多需要 $t$（$R$）秒来完成迭代序列的任意 $R$ 次连续更新。因此，总时间至多为

$$t(R)\times\left\lceil\frac{K}{R}\right\rceil.$$

不失一般性，假设[1] $L\Delta > \varepsilon/2$。因此，利用（2.9），可得

$$t(R)\times\left\lceil\frac{K}{R}\right\rceil = \mathcal{O}\left(t(R)\left(\frac{L\Delta}{\epsilon}+\frac{\sigma^2 L\Delta}{R\epsilon^2}\right)\right). \tag{2.10}$$

将我们选择的（2.8）代入（2.10），即可得到（2.7）。我们取（2.8），注意到这一选择使 $R$ 在通用常数范围内最小化（2.10）。 □

据我们所知，这是文献中首个确立完全异步随机梯度下降变体最优性的结果：时间复杂度（2.7）是最优的，并与Tyurin &Richt´arik（2024年）所确立的下界一致。

推导出的时间复杂度（2.7）具有许多良好且令人满意的特性。

[1]否则，利用 $L$-光滑性， $\|\nabla f(x^0)\|^2 \le 2L\Delta \le \varepsilon$，且初始点是一个 $\varepsilon$-驻点。参见第A.6节。

首先，它对慢速工作器具有鲁棒性：如果 $\tau_n \to \infty$，则该表达式等于

$$\min_{m\in[n-1]} \left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{m\varepsilon^2}\right),$$

实际上忽略了速度最慢的工作节点。接下来，假设 $m_*$ 是使（2.7）最小的最小索引。在这种情况下，（2.7）可进一步简化为

$$\left(\frac{1}{m_*}\sum_{i=1}^{m_*}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{\sigma^2 L\Delta}{m_*\varepsilon^2}\right).$$

这表明，该方法的实际运行效果就好像只有最快的 $m_*$ 个工作节点参与优化过程，这与算法2.2.1中的思路类似。不过，重要的是，该方法能够自适应地 $m_*$ 并自动地确定。

### 2.4.1 阈值的选择

该方法以及定理2.4.3 中的时间复杂度结果还有一个值得注意的性质：阈值 $R$ 与单个计算时间 $\tau_i$无关。因此，该方法可应用于异构分布式系统（其中工作节点的速度各不相同），同时仍能达到（2.10）给出的最优时间复杂度，误差至多相差一个常数因子。

如果希望得到更精确的最优阈值常数级表达式，可以显式计算 $R$。在这种情况下，它将取决于 $\tau_i$的值。最优的 $R$ 满足

$$\arg\min_{R\geq 1}\ t(R)\left(1+\frac{\sigma^2}{R\varepsilon}\right),$$

这可由 (2.10) 推得，方法是舍弃与 $R$无关的常数。这里，$t(R)$ 表示 $R$ 次连续迭代所需的总时间，并且其上界为 (2.6)。

代入此上界，可得最优 $R$的表达式如下：

$$R=\max\left\{\sigma\sqrt{\frac{m^*}{\varepsilon}},1\right\},$$

其中

$$m^*=\arg\min_{m\in[n]}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(1+2\sqrt{\frac{\sigma^2}{m\varepsilon}}+\frac{\sigma^2}{m\varepsilon}\right).$$

如该表达式所示，最优阈值 $R$ 取决于 $m^*$，而后者又由 $\tau_i$ 值决定。

### 2.4.2 证明技术

要得到最终结果，必须解决若干数学难题。引理2.4.2和2.5.1 是新的，因为估计异步 RingmasterASGD 方法的时间复杂度，需要采用不同于

半同步 RennalaSGD 方法的 RingmasterASGD 更加混乱且难以预测。与 Koloskova 等人 (2022年);Mishchenko 等人 (2022a)的工作相比，定理 2.4.1 的证明更严谨、细致，因为我们更仔细地分析了总和

$$\sum_{k=0}^{K} \mathbb{E}\left[\left\|x^k - x^{k-\delta^k}\right\|^2\right]$$

在引理 A.3.2中，我们认为，RingmasterASGD的简洁性，加上新的引理、更精细的分析以及新颖的 $R$ 选择，共同构成了一系列重要的进展，使我们得以达到最优时间复杂度。

## 2.5 任意计算动态下的最优性

在前面的章节中，我们在固定计算模型(2.1)框架下介绍了动机、改进、比较和理论结果。现在，我们证明 RingmasterASGD 在工作节点几乎任意的计算行为下都是最优的。形式化描述这些行为的一种方法是采用通用计算模型 (Tyurin, 2024年)。

对于每个工作节点 $i \in [n]$，我们关联一个计算能力函数

$$p_i : \mathbb{R}_+ \to \mathbb{R}_+ .$$

工作节点 $i$在 $[T^0, T^1]$中计算的随机梯度数量，由其计算能力 $p_i$的积分再经取整操作得到：

$$\text{“\# of stoch. grad. in } [T^0, T^1]\text{”} = \left\lfloor \int_{T^0}^{T^1} p_i(\tau) d\tau \right\rfloor . \tag{2.11}$$

计算能力 $p_i$ 刻画了工作节点的行为，其中考虑了硬件问题或网络延迟导致的潜在断连、处理能力随时间的变化，以及计算速度的波动或趋势。在给定区间内对 $p_i$ 积分，表示已完成的计算工作。如果在 $[T^0, T^1]$ 内 $p_i$ 较小，则积分相应较小，说明工作节点 $i$ 完成的计算较少。反之，如果在 $[T^0, T^1]$ 内 $p_i$ 较大，则该工作节点能够完成更多计算。

在我们的分析中，我们只假设 $p_i$ 非负且几乎处处连续[2]。应用黎曼积分时，我们会隐式地用到这一假设。计算能力 $p_i$ 甚至可以随机变化，并且本文讨论的所有结果均是在给定 $\{p_i\}$随机性的条件下成立的。

我们来看一些示例。如果工作节点 $i$ 在最初的 $t$秒内一直处于非活动状态，随后变为活动状态，则对应于对于所有 $\tau \leq t$， $p_i(\tau) = 0$ ；对于所有 $\tau > t$， $p_i(\tau) > 0$ 。此外，我们允许 $p_i$ 表现出周期性甚至混沌行为[3]。通用计算模型 可简化为固定

[2]因此，它可以在一个可数集上不连续并发生“跳跃”。[3]例如， $p_i(t)$可能呈现如下的不连续行为：$p_i(t) = 0.5t + \sin(10t)$ 对于$t \leq 10$， $p_i(t) = 0$ 对于 $10 < t \leq 20$，并且 $p_i(t) = \max\{80 - 0.5t, 0\}$。

计算模型 当 $p_i(t) = 1/\tau_i$ 对于所有 $t \geq 0$ 和 $i \in [n]$。在这种情况下，区间 $[T^0, T^1]$ 中计算出的随机梯度数量，由 (2.11) 给出，变为 $\lfloor (T^1 - T^0)/\tau_i \rfloor$，这表明工作节点 $i$在 $T^0 + \tau_i$秒后计算出一个随机梯度，在 $T^0 + 2\tau_i$秒后计算出两个随机梯度，依此类推。

注意，定理2.4.1在任何计算模型下都成立。接下来，我们为2.4.2引理介绍一种适用于通用计算模型的替代方案。

**引理 2.5.1** （证明见附录 A.2）。 设工作节点计算时间满足通用计算模型。设 $R$ 为算法 2.3.1 或算法 2.3.2的延迟阈值。假设某次迭代在时间 $T^0$开始。从这次迭代开始， $R$ 次连续的算法 2.3.1或算法 2.3.2 迭代点更新将在时间

$$T(R, T^0) := \min \left\{ T \geq 0 : \sum_{i=1}^{n} \left\lfloor \frac{1}{4} \int_{T^0}^{T} p_i(\tau) d\tau \right\rfloor \geq R \right\}.$$

该结果推广了引理 2.4.2。具体而言，若 $p_i(t) = 1/\tau_i$ 对所有 $t \geq 0$以及$i \in [n]$，则可以证明 $T(R, T^0) - T^0 = \Theta(t(R))$ 对所有 $T^0 \geq 0$均成立。

结合定理 2.4.1和引理 2.5.1，我们现在可以给出主要结果。

**定理 2.5.2** (RingmasterASGD的最优性；证明见附录 A.4)。 设假设 2.1.1、2.1.2和 2.1.3 成立。设 RingmasterASGD(算法 2.3.1或算法 2.3.2) 中的步长为

$$\gamma = \min \left\{ \frac{1}{2RL}, \frac{\varepsilon}{4L\sigma^2} \right\},$$

且延迟阈值为

$$R = \max \left\{ 1, \left\lceil \frac{\sigma^2}{\varepsilon} \right\rceil \right\}.$$

那么，在通用计算模型下，Ringmaster ASGD 至多经过 $T^{\bar{K}}$ 秒即可找到一个 $\varepsilon$– 驻点，其中

$$\bar{K} := \left\lceil \frac{48L\Delta}{\epsilon} \right\rceil$$

且 $T^{\bar{K}}$ 是以下递归定义序列中的第 $\bar{K}$ 个元素：

$$T^k := \min \left\{ T \geq 0 : \sum_{i=1}^{n} \left\lfloor \frac{1}{4} \int_{T^{k-1}}^{T} p_i(\tau) d\tau \right\rfloor \geq R \right\}$$

对所有 $k \geq 1$ 和 $T^0 = 0$均成立。

诚然，所得定理不如定理2.4.3明确。然而，为了使结果在其允许的计算时间动态方面具有普适性，这是必须付出的代价。确定时间复杂度 $T^K$ 需要依次计算 $T^1$、$T^2$、等等。不过，在某些情况下，$T^K$ 可以显式导出。例如，如果 $p_i$（$t$） $= 1/\tau_i$ 对于所有 $t \geq 0$ 和 $i \in [n]$，那么 $T^K$ 由（2.7）给出。此外， $T^1$ 表示计算前

个随机梯度所需的秒数，$T^2$ 表示计算前 $2 \times R$ 个随机梯度所需的时间，依此类推。自然地，为了计算$T^2$，必须首先确定 $T^1$，鉴于该过程的顺序性质，这是合理的。

所得结果是最优的，与Tyurin (2024年)所建立的下界一致，并且无法通过任何异步并行方法进一步改进。

## 2.6 实验

异步随机梯度下降 始终展现出其有效性和实用性，在各种应用中都取得了良好表现（Recht 等人，2011年；Lian 等人，2018年；Mishchenko 等人，2022a），此外还有大量其他研究支持其效用。本文的主要目标是进一步改进该方法，并证明它不仅具有实用性，而且是理论上最优的。

我们使用所提出的RingmasterASGD 方法进行一个玩具实验，并将其与两个基线方法进行比较：先前的朴素异步随机梯度下降 （遵循Mishchenko 等人（2022a）的做法，称为延迟自适应异步随机梯度下降）以及RennalaSGD。该优化任务基于一个凸二次函数 $f\colon\ \mathbb{R}^d \to \mathbb{R}$ 定义为

$$f(x) = \frac{1}{2}x^\top Ax - b^\top x\ , \qquad \forall x \in \mathbb{R}^d\ ,$$

其中

$$A = \frac{1}{4}\begin{bmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{d\times d}\ , \quad b = \frac{1}{4}\begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^d\ .$$

我们设定 $d = 1729$ 和 $n = 6174$。每个 $n$ 工作节点计算形式为以下的无偏随机梯度

$$\nabla f(x,\xi) = \nabla f(x) + \xi\ ,$$

其中 $\xi \sim \mathcal{N}(0,\ 0.01_2)$。

实验采用 Python 实现。分布式环境通过使用配备 Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz 的机器进行模拟。每个工作节点的计算时间模拟为 $\tau_i = i + |\eta_i|$ 对所有$i \in [n]$，其中 $\eta_i \sim \mathcal{N}(0,\ i)$。我们从集合 $\{5^p :\ p \in [-5,5]\}$中调整步长。Rennala SGD 的批量大小以及 Ringmaster ASGD 的延迟阈值均从集合 $\{\lceil n/4^p \rceil :\ \ p \in \mathbb{Z}_+\}$中调整。实验结果如图 2.6.1所示。

所得结果证实，在所考虑的设置下，RingmasterASGD 确实比 Delay-AdaptiveASGD 和RennalaSGD 更快。可以看出，数值实验支持我们的理论结果，并且我们显著提高了Naive AsynchronousSGD（Delay-Adaptive ASGD）这一先前版本的收敛速率。

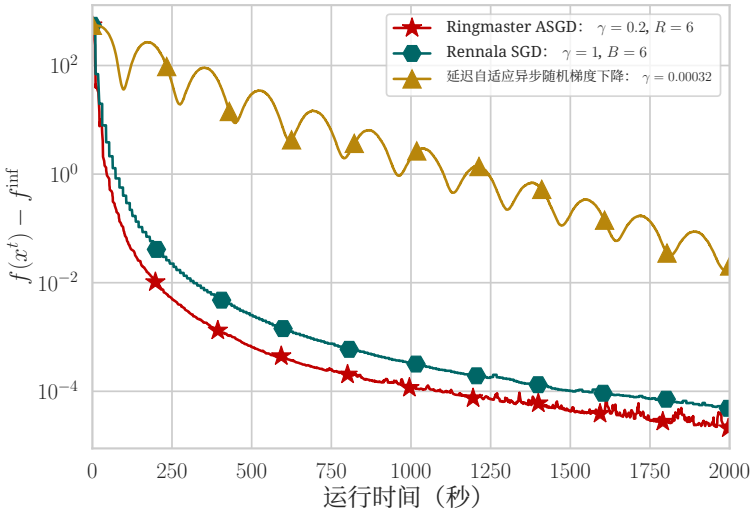



图 2.6.1：使用 $n = 6174$ 和 $d = 1729$ 进行的实验，展示了RingmasterASGD、延迟自适应异步随机梯度下降和RennalaSGD的收敛情况。

### 2.6.1 神经网络实验

为展示我们的方法同样适用于神经网络，我们在 MNIST 数据集（LeCun 等人，1998）上训练了一个采用 ReLU 激活的小型 20 层神经网络。我们使用了与前一项实验相同数量的工作节点（$n = 6174$），并保持相同的时间分布。结果如图 2.6.2所示。

## 2.7 其他相关工作

在联邦学习中，通信通常比本地计算更耗时，因此促使人们采用本地训练策略（McMahan 等人，2016年；Mishchenko 等人，2022b；Malinovsky 等人，2022年；Maranjyan 等人，2025c）。类似地，在我们的设定中，工作节点无需在同一个模型点上反复计算梯度，而是可以执行若干本地梯度下降步骤，并在收到请求时将更新后的模型发送给服务器。以往研究已探索了此类方案的异步变体，包括对其进行明确时间复杂度分析的方案（Tyurin 和 Sivtsov，2025年；Fradin 等人，2025年）。

## 2.8 结论与未来工作

在本研究中，我们开发了首个可实现最优时间复杂度的异步随机梯度下降方法，称为Ring-masterASGD。通过采用精心设计的算法方法（可解释为具有自适应步长的异步随机梯度下降（2.4）），我们成功实现了这一目标。通过在算法 $R$ 中选择适当的延迟阈值2.3.1，该方法达到了Tyurin & Richt´arik (2024年）和Tyurin (2024年）所确立的理论下界。

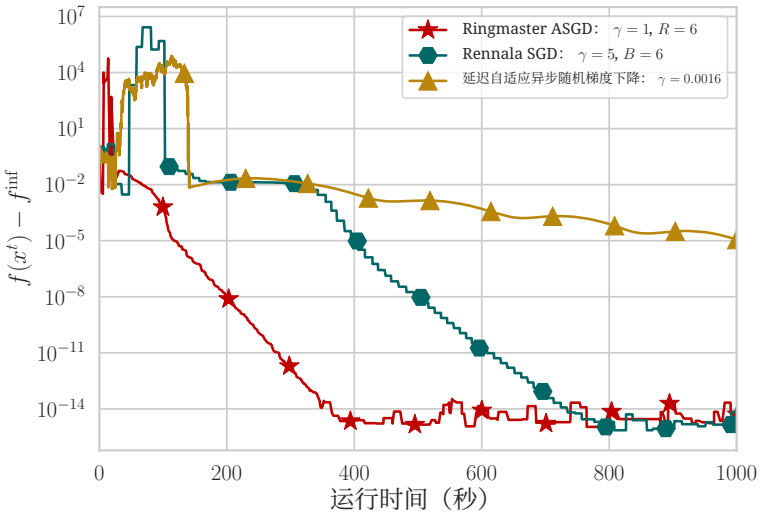



图 2.6.2：我们在一个采用 ReLU 激活的简单双层神经网络上使用 MNIST 数据集进行了实验，结果表明，我们的方法 RingmasterASGD 更加稳健，且优于 延迟自适应异步随机梯度下降 和 RennalaSGD。

未来工作可以探索异构场景，即每台设备上的数据来自不同的分布；这对于联邦学习（联邦学习）（Kone˘cn´y 等人，2016b；McMahan 等人，2016年；Kairouz 等人，2021年）尤其重要。在联邦学习中，通信成本也至关重要，因此值得研究类似的扩展（Alistarh 等人，2017年；Tyurin 等人，2024b；Tyurin& Richt´arik，2024年）。此外，设计一种完全异步的对应方法，以替代（Tyurin 等人，2024a）中的方法，也很有意义。此外，正如Maranjyan 等人(2025a)所讨论的，计算时间和通信时间可以视为随机变量。进一步研究这些情况并推导各种分布的闭式表达式，将很有价值。

# 第 3 章

## Ringleader ASGD：数据异质性条件下首个具有最优时间复杂度的异步随机梯度下降

本章基于以下研究成果Maranjyan 与 Richt´arik （2025）：

> Artavazd Maranjyan 和 Peter Richt´arik. Ringleader ASGD：数据异质性条件下首个具有最优时间复杂度的异步随机梯度下降。*arXiv* 预印本 *arXiv:2509.22860*，2025年

## 3.1 引言

现代机器学习日益依赖于在集群中进行大规模分布式训练，这些集群配备数百甚至数千个图形处理器（Shoeybi等人，2019年；Brown 等人，2020年；Narayanan 等人，2021年）。然而，经典的同步训练方法在这类场景中难以扩展，因为设备故障、网络不稳定和同步开销会造成显著的效率低下（Chen 等人，2016a；Grattafiori 等人，2024年）。在计算能力异构的环境中，例如联邦学习（FL），这些问题更加突出；在此类环境中，设备涵盖高端数据中心图形处理器和资源受限的边缘硬件（Kone˘cn´y 等人，2016b；McMahan 等人，2016年；Li 等人，2020b；Kairouz 等人，2021年）。由于同步方法受最慢参与方的制约，较快的设备只能闲置，导致计算资源遭到严重闲置；当掉队节点——因计算或通信而变慢的节点——远远落后时，这种情况尤为明显。

减少同步瓶颈的一种方法是为数据中心配备同构图形处理器。然而，这种方法成本高昂且难以扩展：升级到速度更快的图形处理器需要同时更换所有设备，因为异构硬件无法高效地组合使用。即便如此，同质性也无法消除同步问题，因为训练期间的硬件故障和设备掉线仍会导致掉队节点和空闲时间。此外，该解决方案仅适用于受控的数据中心环境，而在联邦学习中不可行，因为边缘设备不受服务器控制。

一种更有前景的方法是采用异步优化方法，从硬件解决方案转向算法解决方案。这些方法无需同步，使快速工作节点无需等待较慢的工作节点即可贡献更新（Tsitsiklis 等人，1986年；Recht 等人，2011年；Agarwal 和 Duchi，2011年；Dean 等人，2012年；Li 等人，2014年）。尽管异步方法颇具吸引力，但分析起来更为困难。具体而言，要进行有意义的分析，就必须研究收敛时间，而不能只关注迭代复杂度。虽然迭代复杂度是优化领域的传统指标，但它未必能反映并行环境中的实际训练速度：如果某种方法的迭代无需等待慢速工作节点即可完成，那么即使它执行的迭代次数更多，也可能在实际运行时间上更快

这一点引出了一个根本问题：在所有并行方法中，理论上哪些方法可以证明速度最快？为了明确这一问题，我们将研究范围限定为光滑非凸问题和随机一阶方法，其中包括采用同步或不采用同步的算法。本文仅考虑这一设定。

最近，Tyurin &Richt´arik （2024年）研究了这一特定情形，并推导出了下界。随后，他们提出了两种算法：RennalaSGD，专为齐次数据设置设计，即所有工作节点都从同一分布中抽取样本；以及MaleniaSGD，适用于异构数据环境，即不同工作节点的数据分布各不相同。他们证明，这两种方法都是最优的——均达到了下界；也许令人意外的是，两者都是同步的（会定期同步工作节点）。这两种方法的核心思想是，通过让工作节点持续忙碌，充分利用可用的计算资源：每个工作节点独立计算，只有在累积了足够多的梯度计算后才进行同步。

起初，Tyurin & Richt´arik （2024年）的结果似乎预示着异步方法前景相当悲观：尽管异步方法在实践中颇具吸引力，他们还是证明，现有的异步方法并非最优，而达到下界的方法是同步的。这使人们认为，最优性必然与同步密不可分。然而，Maran- jyan 等人（2025d）推翻了这一观点。他们在齐次数据设置下提出了Ringmas-terASGD——首个达到与同步RennalaSGD相同最优时间复杂度的异步随机梯度下降 方法。尽管这两种方法具有相同的理论保证，RingmasterASGD 在实践中可能比RennalaSGD 更快，因为它无需同步，并能受益于更频繁的更新。

然而，Maranjyan 等人（2025d）仅在同质数据环境中证明了最优性。对于不依赖同步（即采用异步方式）的并行方法，是否存在某种变体也能在更一般的异构数据环境中达到最优，这个问题仍未解决。本文填补了这一空白，并对此给出了肯定的回答。

异构数据环境既重要，在实践中也有现实意义。例如，在联邦学习中，参与方持有不同的数据集，因此自然会产生这种异质性（Zhao 等人，2018年；Li 等人，2020b；Tan 等人，2022年）。然而，这种环境比同质数据环境更具挑战性。异步随机梯度下降的标准做法——每次计算梯度后都更新模型——在此可能有害：快速工作节点贡献更新的频率更高，导致优化过程偏向其本地数据。为缓解这一问题，大多数现有异步方法都假设客户端分布相似（Mishchenko 等人，2022a；Koloskova 等人，2022年；Nguyen 等人，2022年；Islamov 等人，2024年）。尽管这一假设简化了分析，但在实际应用中往往并不现实，因为客户端可能涉及人口构成截然不同的群体（例如，人口统计特征各异的医院、来自不同国家的移动用户，或受到不同法规约束的金融机构）。

Wang 等人Wang 等人（2025年）通过提出增量聚合异步随机梯度下降（IA$^2$SGD，朝着消除这些限制性假设迈出了重要一步。这种方法无需相似性假设即可保证收敛。然而，其时间复杂度与标准的朴素小批量-

表 3.1：固定计算时间模型下并行一阶方法的时间复杂度比较。在该模型中，每个工作节点 $i$ 都需要固定时间 $\tau_i$ 来计算随机梯度，且各时间按顺序排列，使得 $\tau_n$ 最大（3.2）。我们用 $\tau_{\text{avg}} := \frac{1}{n}\sum_{i=1}^{n}\tau_i$ 表示所有工作节点的平均计算时间。该表展示了不同算法的时间复杂度如何取决于关键问题参数：初始函数次优性 $\Delta := f(x^0) - f^*$（假设 3.3.5）、目标平稳性 $\varepsilon$、随机梯度的方差上界 $\sigma^2$（假设 3.3.1）以及光滑常数。具体而言，$L_f$是 $f$ 的光滑常数（定义 3.3.3）；$L_{\max} := \max_{i\in[n]} L_{f_i}$，其中$L_{f_i}$ 是 $f_i$的光滑常数；$L$ 是与我们提出的新光滑性类假设（假设 3.3.2）相关的常数。它们满足 $L_f \le L \le L_{\max}$（引理 3.3.4）。所有列出的时间复杂度均省略通用常数因子。

每一列表示某种方法是否满足以下理想属性：**最优的：** 对于异构数据环境中的并行一阶随机方法，达到由Tyurin&Richt´arik （2024年）推导出的**理论下界**。**无需同步：** 无需同步，因此是异步的。**无空闲工作节点：** 所有工作节点始终处于忙碌状态，无需等待，因此计算资源得到充分利用。**无丢弃的工作：** 不会浪费任何计算，也不会让任何工作节点在计算过程中途停止。我们的新方法RingleaderASGD是首个实现最优时间复杂度的异步方法，同时还能确保资源得到充分利用（无空闲工作节点），且不会丢弃计算或工作。

| 算法 | Time<br>复杂度 | 最优的 | No<br>同步。 | No<br>Idle<br>工作节点 | No<br>已丢弃<br>Work |
|---|---|---|---|---|---|
| 朴素小批量随机梯度下降<br>（第 3.4.1节） | $\frac{L_f\Delta}{\varepsilon}\left(\tau_n + \tau_n\frac{\sigma^2}{n\varepsilon}\right)$ | ✘ | ✘ | ✘ | ✔ |
| IA²随机梯度下降<br>（Wang 等人，2025年）<br>（第 B.3节） | $\frac{L_{\max}\Delta}{\varepsilon}\left(\tau_n + \tau_n\frac{\sigma^2}{n\varepsilon}\right)$ (†) | ✘ | ✔ | ✔ | ✔ |
| Malenia SGD<br>(Tyurin &Richt´arik, 2024年) | $\frac{L_f\Delta}{\varepsilon}\left(\tau_n + \tau_{\text{avg}}\frac{\sigma^2}{n\varepsilon}\right)$ | ✔ | ✘ | ✔ | ✘ |
| RingleaderASGD （新）<br>（算法 3.5.1；<br>定理 3.6.5） | $\frac{L\Delta}{\varepsilon}\left(\tau_n + \tau_{\text{avg}}\frac{\sigma^2}{n\varepsilon}\right)$ | ✔(‡) | ✔ | ✔ | ✔ |

(†)Wang 等人 （2025年）给出的分析假设每个 $f_i$ 都具有相同的光滑常数，即每个 $f_i$ 都是 $L_{f_i}$–光滑的，然后对所有 $L_{f_i}$ 使用上界 $L_{\max}$。然而，这一严格假设并非必要：他们可以改用我们的宽松光滑性类型假设（假设 3.3.2），这样常数就会改进为 $L$，且其分析保持不变。 (‡) Ringleader ASGD 和 Malenia SGD 的时间复杂度仅在光滑常数上有所不同。由于 Malenia SGD 是最优的，当 $L$ 至多比 $L_f$ 大一个通用常数因子时，Ringleader ASGD 也同样是最优的，即 $L = \mathcal{O}(L_f)$。

批量随机梯度下降 （见表 3.1 的前两行）——这是最简单的同步 随机梯度下降基线方法，它在每次更新之前都要等待从每个工作节点收集一个梯度，因此无法提供最初促使人们采用异步方法的计算优势。

据我们所知，在异构数据环境 下，唯一已被证明具有最优性的同步算法 Malenia SGD 是Tyurin &Richt´arik (2024年)提出的算法。值得注意的是，该算法不依赖相似性假设。然而，同步在实践中是一个主要瓶颈：尽管同步方法和异步方法的理论复杂度可能相同，但异步方法在实践中通常更快，因为它们避免了高成本的同步，并能受益于更频繁的更新；正如Maranjyan 等人 (2025d)在同构情形中所展示的那样。

这引出了一个根本性问题：能否设计一种无需任何相似性假设、同时仍能实现最优时间复杂度的异步方法？本文对此给出了肯定的回答，并提出了 RingleaderASGD，这是首个在异构数据环境下实现最优时间复杂度的异步随机梯度下降方法[1] 。重要的是，RingleaderASGD无需依赖严格的相似性假设即可达到这一目标。

### 3.1.1 贡献

我们的主要贡献如下：

- **数据异质性下的最优异步随机梯度下降**。据我们所知，我们证明 RingleaderASGD （算法3.5.1）是首个在固定计算模型（3.2）下、适用于异构环境且达到并行方法下界的异步方法；该下界由Tyurin &Richt´arik （2024年）确立（表3.1）。至关重要的是，它不依赖客户端数据集之间的任何相似性假设。

- **其他有用特性**。除了实现最优时间复杂度外，我们的方法 RingleaderASGD 还具备两项额外的理想特性：(i) 所有工作节点始终处于活跃状态（没有空闲工作节点）；(ii) 每个计算得到的梯度都会纳入更新（不丢弃任何工作）。这些特性在实践中至关重要，因为它们能确保资源利用率最大化：所有工作节点始终参与计算，其计算成果也绝不会浪费。表 3.1 根据这些特性，对比了RingleaderASGD 与基准算法。

- **无参数设计**。与最优同步方法MaleniaSGD (Tyurin & Richt´arik，2024年)不同，该方法在固定计算时间模型下运行，无需此类参数（步长除外；步长仅用于达到理论上的最优速率）。这使其在实际部署中实用得多，因为这些量通常未知或难以估算。同样的无参数

[1]在全文中，我们将我们的方法称为最优的。形式上，只要常数 $L$——与我们新提出的光滑性类假设（假设 3.3.2相关——不超过推导所得下界中所用光滑常数的一个常数因子 $L_f$ （Tyurin & Richt´arik，2024年）。详情见表 3.1 。

改进也可以扩展到MaleniaSGD，正如我们在第B.4节所讨论的那样。

- **通用计算模型**。在第B.1节中，我们将分析从固定计算时间模型扩展到计算时间任意变化的一般情形，从而涵盖几乎任何计算行为，包括随机或对抗模式，同时保持最优时间复杂度。

- **经验验证**。在第3.7节中，我们在具有代表性的玩具问题上，将RingleaderASGD与基准方法进行比较。结果验证了我们的理论发现，并表明该方法相较于基线具有明显的实践优势。

## 3.2 相关工作

异步随机梯度方法的研究可追溯至 Tsitsiklis 等人（1986年）的开创性工作，并随着 HOGWILD! 算法的提出（Recht 等，2011）而重新受到关注。HOGWILD! 本质上是一种异步坐标下降方法：更新以无锁方式执行，并伴有不一致的读写；其收敛保证依赖于稀疏性假设，而这些假设在现代大规模机器学习中往往并不现实。此后，这一范式也得到了一些改进（J Reddi 等人，2015年；Zhao 和 Li，2016年；Pan 等人，2016年；Mania 等，2017；Leblond 等，2018；Nguyen 等，2018；Zhou 等人，2018年），但这些工作仍局限于存在不一致内存访问的坐标下降设置，因此与我们的研究重点有很大不同。

与我们的设定更为接近的是基于梯度进行更新且更新方式一致的研究。早期的贡献通常采用同质数据假设（所有工作节点均从同一分布中采样），其中包括 Agarwal 和 Duchi（2011年）对凸目标函数的研究，以及后来拓展至非凸情形的研究，例如 Lian 等人（2015年）和 Dutta 等人（2018年）的工作；后者分析了计算时间服从指数分布的情况。这一研究方向上的其他相关结果包括（Feyzmahdavian 等人，2016年；Zheng 等人，2017年；Arjevani 等人，2020年；Feyzmahdavian 和 Johansson，2023年），这些研究均假设延迟固定。近年来，人们提出了延迟自适应方法，旨在通过降低极为陈旧的梯度的权重来提升性能（Cohen 等人，2021；Wu 等人，2022；Koloskova 等人，2022；Mishchenko 等人，2022a）。

与我们工作尤其相关的是SAGA的Leblond 等人（2018年）沿着HOGWILD!的思路开发了共享参数版本，而Glasgow 与 Wootters（2022年）研究了采用全梯度的分布式设置，这与我们的随机梯度视角形成对比。

近期大量研究探讨了联邦学习（FL）中的异步方法，其中客户端持有来自异质分布的数据。重要贡献包括（Aytekin 等，2016年；Xie 等，2019年；Mishchenko 等人，2022a；Koloskova 等人，2022；Wang 等人，2022a,b；Glasgow 与 Wootters，2022；Fraboni 等，2023年；Zhang 等，2023年；Wang 等人，2024年；Islamov 等人，2024年；Alahyane 等，2025年）。

更广泛地说，Assran 等人 (2020年) 对异步优化方法进行了全面综述。

另一条研究脉络始于 Tyurin & Richt´arik(2024年)，他们为并行方法建立了下界，并提出了最优的同步算法及其异步对应方法。此后，若干后续论文将这一半异步框架扩展到了其他场景（Tyurin 等人，2024a，b；Tyurin& Richt´arik，2024年；Maranjyan 等人，2025a）。

最后，除了异步方法之外，一些同步方法还通过根据工作节点速度调整本地训练来应对系统异构性。典型方法 FedAvg (McMahan 等人，2017年) 会在每个工作节点上执行多个本地步数。一些变体会调整本地步数，以匹配工作节点的计算速度（Li 等人，2020b；Maranjyan 等人，2025c），从而有效平衡异构系统中的任务分配。最近，Maranjyan 等人(2025b) 提出动态调整本地步数，而无需预先了解工作节点的速度。

## 3.3 问题设置

我们考虑一个包含 $n$ 个工作节点的分布式学习场景，其中每个工作节点 $i$都拥有自己的本地数据分布 $\mathcal{D}_i$。我们的目标是求解以下分布式优化问题：

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \left\{ f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \right\}, \quad \text{where} \quad f_i(x) := \mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left[f_i(x;\xi_i)\right]. \tag{3.1}$$

其中 $f_i\colon\ \mathbb{R}^d \to \mathbb{R}$ 表示工作节点 $i$的局部目标函数，定义为样本损失 $f_i(x;\xi_i)$ 在从其本地分布$\mathcal{D}_i$中抽取的数据点 $\xi_i$ 上的期望。

### 3.3.1 工作者异质性模型

我们首先关注工作节点具有恒定计算速度的情况，因为这种设置更直观，也为理解异步分布式优化的动态过程提供了基础模型。任意计算时间情形的扩展将在第 B.1节介绍。

遵循 固定计算模型 (Mishchenko 等人，2022a)，我们将其形式化如下：

$$\begin{aligned} &\text{Each worker } i \text{ requires } \tau_i \text{ seconds}^{2} \text{ to compute one stochastic gradient } \nabla f_i(x;\xi_i).\\ &\text{Without loss of generality, we assume}\\ &\qquad 0 < \tau_1 \le \tau_2 \le \cdots \le \tau_n\ . \end{aligned} \tag{3.2}$$

我们假设通信速度无限快（耗时 0 秒），无论是从工作节点到服务器，还是从服务器到工作节点[3]。这是建模上的一种选择——可以说，

[2]另一种假设是，每个工作节点所需时间至多为 $\tau_i$ 秒。在这种表述下，我们的所有上界仍然成立；然而，下界将不再有效。因此，我们采用每个工作节点恰好需要 $\tau_i$ 秒的假设。[3]或者，也可以将 $\tau_i$ 定义为工作节点计算梯度并将其发送到服务器所需的时间，同时假设服务器到工作节点的通信时间为无限

这是最简单的一种——也是以往研究中的标准假设，Mishchenko 等人（2022a）；Koloskova 等人（2022年）；Tyurin & Richt´arik （2024年）；Maranjyan 等人（2025d），即使这一假设并不总是被明确提出。Tyurin 等人Tyurinet al. （2024b）的一项相关研究考虑了通信不可忽略的情况，并提出了应对这一问题的技术。

最后，我们用 $\tau_{\text{avg}} := \frac{1}{n}\sum_{i=\overline{1}}^{n}\tau_i$ 表示所有工作节点的平均计算时间。

### 3.3.2 符号

我们将 $\mathbb{R}^d$ 中的标准内积记为

$$\langle x, y\rangle = \sum_{i=1}^{d} x_i y_i \ ,$$

并将相应的欧几里得范数记为 $\|x\| := \sqrt{\langle x,\ x\rangle}$。我们用 $\overline{[n]} := \{1, 2, \ldots, n\}$ 表示索引集，用 $\mathbb{E}[\cdot]$ 表示数学期望。对于函数 $\phi,\ \psi\colon \mathcal{Z} \to \mathbb{R}$，如果存在常数 $C > 0$，使得对所有 $z \in \mathcal{Z},\ \phi = \mathcal{O}(\psi) \leq C\psi(z)$，则记作 $\phi(z)$。

### 3.3.3 假设

我们考虑非凸情形下的以下标准假设。

**假设 3.3.1。** 对于每个 $i \in [n]$ 和任意 $\xi_i$，函数 $f_i(x;\xi_i)$ 关于其第一个参数 $x$ 可微。此外，随机梯度是无偏的，且具有有界方差 $\sigma^2 \geq 0$，即

$$\mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left[\nabla f_i(x;\xi_i)\right] = \nabla f_i(x), \quad \forall x \in \mathbb{R}^d,\ \ \forall i \in [n] \ ,$$
$$\mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left[\|\nabla f_i(x;\xi_i) - \nabla f_i(x)\|^2\right] \leq \sigma^2, \quad \forall x \in \mathbb{R}^d,\ \ \forall i \in [n] \ .$$

**假设 3.3.2。** 每个函数 $f_i$ 都可微。存在一个常数$L > 0$，使得对于所有 $x \in \mathbb{R}^d$ 和 $y_1$、……、$y_n \in \mathbb{R}^d$，

$$\left\|\nabla f(x) - \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(y_i)\right\|^2 \leq \frac{L^2}{n}\sum_{i=1}^{n}\|x - y_i\|^2 \ .$$

回顾光滑性的标准定义

**定义 3.3.3**（光滑性）。 若可微函数 $\phi\colon \mathbb{R}^d \to \mathbb{R}$ 满足以下条件，则称其为 $L_\phi$–光滑：

$$\|\nabla\phi(x) - \nabla\phi(y)\| \leq L_\phi\|x - y\|, \quad \forall x, y \in \mathbb{R}^d \ .$$

按照惯例，$L_\phi$ 表示这样的最小常数。

速度很快。在这种表述下，我们的上界仍然成立，但下界不再适用，因此我们采用更简单的模型。

注意，假设 3.3.2 比要求 $f$ 本身是 $L_f$–光滑更强，但比要求所有 $f_i$ 都是 $L_{f_i}$–光滑更弱。下面的引理阐明了常数 $L_f$、$L$和 $L_{f_i}$之间的关系。

**引理 3.3.4**（光滑性界；证明见第 B.2.1节）。假设假设 3.3.2 以常数 $L > 0$ 成立。那么 $f$ 是 $L_f$–光滑的，且 $L_f \leq L$。此外，若每个 $f_i$ 都是 $L_{f_i}$–光滑的，则满足假设 3.3.2，并且有

$$L_f \;\leq\; L \;\leq\; \sqrt{\frac{1}{n}\sum_{i=1}^{n} L_{f_i}^2} \;\leq\; \max_{i\in[n]} L_{f_i} =: L_{\max}\,.$$

最后，如果所有 $f_i$ 都相同，即对所有 $i \in [n]$都有 $f_i = f$，则 $L = L_f$。

据我们所知，先前关于数据异质性下异步 随机梯度下降 的研究始终假设每个 $f_i$（Koloskova 等人，2022年；Nguyenetal.，2022年；Wang etal.，2025年）。

**假设 3.3.5**。存在 $f^* > -\infty$，使得对所有 $x \in \mathbb{R}^d$，$f(x) \geq f^*$。我们定义 $\Delta := f(x^0) - f^*$，其中 $x^0$是优化方法的起始点。

在这些假设下；我们的目标是找到一个 $\varepsilon$–驻点，即满足 $\mathbb{E}[\|\nabla f(x)\|^2] \leq \varepsilon$ 的（可能是随机的）向量 $x$。

## 3.4 背景与动机

本节回顾分布式优化领域的相关现有方法，并讨论其局限性，以此引出我们的算法设计。我们首先介绍朴素小批量随机梯度下降 这一典型的同步基线方法，接着考察 Male-niaSGD (Tyurin & Richt´arik，2024年)，这是首个达到最优时间复杂度的同步随机梯度下降方法。随后，我们转向异步方法面临的挑战，重点关注近期提出的 IA$^2$随机梯度下降(Wang 等人，2025年)及其局限性。最后，我们概述如何克服这些局限性，并介绍我们算法背后的核心思想。

### 3.4.1 朴素小批量随机梯度下降

朴素小批量随机梯度下降 是在分布式设置下求解问题（3.1）最直接的方法。

**算法描述**。在每次迭代 $k$ 中，算法执行以下更新：

$$x^{k+1} = x^k - \gamma \frac{1}{n}\sum_{i=1}^{n} \nabla f_i\left(x^k; \xi_i^k\right).$$

该算法采用同步方式：在每次迭代中，服务器等待从每个 $n$ 工作节点接收一个随机梯度，这些梯度均在当前迭代点 $x^k$处计算。收集到所有梯度后，服务器通过对这些随机梯度取平均，构造全梯度的无偏小批量估计量，并执行标准的梯度下降步。

**局限性**。 这种同步方法存在显著的计算瓶颈。每次迭代都需要等待最慢的工作节点完成梯度计算，因此迭代时间为 $\tau_n := \max_{i\in[n]} \tau_i$ (3.2)。因此，较快的工作节点在等待掉队节点时会处于空闲状态，导致可用计算资源无法得到有效利用。

**理论性能**。就收敛速率而言，朴素小批量随机梯度下降达到以下迭代复杂度

$$\mathcal{O}\left(\frac{L_f\Delta}{\varepsilon}\left(1+\frac{\sigma^2}{n\varepsilon}\right)\right)$$

以达到一个 $\varepsilon$–驻点（Cotter 等人，2011年；Goyal 等人，2017年；Gower 等人，2019年）。相应的时间复杂度为

$$\mathcal{O}\left(\frac{\tau_n L_f\Delta}{\varepsilon}\left(1+\frac{\sigma^2}{n\varepsilon}\right)\right).$$

这促使人们开发能够更充分利用快速工作节点、而无需等待掉队节点的方法。

## 3.4.2 Malenia SGD

MaleniaSGD （Tyurin 与 Richt´arik，2024年）通过确保工作节点持续工作，而不是在每次迭代中等待最慢的工作节点，从而解决了朴素小批量随机梯度下降中的掉队节点问题。然而，从根本上说，它仍是一种小批量随机梯度下降算法：每次迭代中，它都根据多个梯度构造一个无偏梯度估计器，然后执行同步更新。关键区别在于批次的收集方式——MaleniaSGD 异步收集梯度，因此在单次迭代中可能会包含来自同一工作节点的多份贡献，这与朴素小批量随机梯度下降不同。

**算法描述**。在每次迭代 $k$中，所有工作节点持续在当前点 $x^k$ 计算随机梯度。服务器不断累积这些梯度，直至满足停止条件

$$\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{b_i^k}\right)^{-1} \geq \max\left\{1, \frac{\sigma^2}{n\varepsilon}\right\} \tag{3.3}$$

得到满足，其中 $b_i^k$ 表示从工作节点 $i$接收到的梯度数量。一旦满足此条件，服务器便执行更新

$$x^{k+1} = x^k - \gamma\bar{g}^k := x^k - \gamma\frac{1}{n}\sum_{i=1}^{n}\bar{g}_i^k\ ,$$

其中 $\bar{g}_i^k$ 是从工作节点 $i$接收到的随机梯度的平均值

$$\bar{g}_i^k = \frac{1}{b_i^k} \sum_{j=1}^{b_i^k} \nabla f_i \left( x^k; \xi_i^{k,j} \right).$$

由于随机梯度是异步收集的，但只有在收到所有所需梯度后才会执行更新，因此该算法可视为半异步算法——梯度收集是异步的，而参数更新是同步的。

**停止条件依据**。 条件（3.3）的左侧出现在梯度估计量 $\bar{g}^k$ 的方差中。确保这一量足够大，可使算法控制每一步的方差。此外，条件（3.3）保证每个工作节点至少计算一个梯度，因此其方差小于朴素小批量随机梯度下降中的估计量方差。

**理论性能**。 这种异步梯度收集策略可达到最优的时间复杂度

$$\mathcal{O}\left( \frac{L_f \Delta}{\varepsilon} \left( \tau_n + \tau_{\text{avg}} \frac{\sigma^2}{n\varepsilon} \right) \right),$$

如Tyurin & Richt´arik （2024年）所示。相较于朴素小批量随机梯度下降，其关键改进在于方差项 $\sigma^2$ 现在乘以 $\tau_{\text{avg}}$ ，而不是 $\tau_n$。由于 $\tau_{\text{avg}} \ll \tau_n$ 在计算异构环境中，MaleniaSGD 在高度异构的场景下有望带来显著的加速。

该算法的优势在高噪声情形下最为显著（此时$\sigma^2/n\varepsilon$ 较大）。在低噪声情形下，或当 $\sigma = 0$ 时，MaleniaSGD 并不比朴素小批量随机梯度下降更有优势，因为每个工作节点收集多个梯度并不能带来方差缩减方面的益处。

**局限性**。MaleniaSGD 的主要局限在于其同步特性。收集到所需的梯度后，服务器必须同时向所有工作节点广播更新后的模型。这种一次性同步会产生大量通信开销，在大规模或带宽受限环境中可能成为可扩展性的主要瓶颈。

此外，同步过程会迫使服务器丢弃广播发生时仍在积极计算的工作节点正在进行的梯度计算。由于工作节点在梯度收集阶段会持续运行，因此收到新模型后必须放弃当前计算，浪费宝贵的计算资源，而这些资源原本可以用于促进收敛。

此外，MaleniaSGD 需要预先了解噪声水平 $\sigma$ 和目标精度 $\varepsilon$ ，以评估停止条件（3.3）。这种对问题特定参数的依赖显著限制了其实际适用性，而这些参数在实践中往往未知或难以估计。

这些同步瓶颈促使人们开发异步优化方法。异步方法不仅可以避免协调开销，还能在梯度到达时立即更新模型，从而通过更频繁的参数更新加快收敛。这种即时处理对于稀疏模型尤其有益，因为梯度通常影响彼此不重叠的参数子集，因此可以进行并行更新而不会相互干扰（Recht 等人，2011年）。

### 3.4.3 迈向异步方法

一种实现异步优化的简单方法，是在收到任何梯度后立即更新模型，使用

$$x^{k+1} = x^k - \gamma \nabla f_{i^k}\left(x^{k-\delta^k}; \xi_{i^k}^{k-\delta^k}\right),$$

其中，$i^k$ 表示在迭代 $k$时发送梯度的工作节点，$\delta^k \geq 0$ 表示其延迟，即计算梯度期间发生的服务器更新次数。延迟在异步执行中自然产生：快速工作节点会迅速返回梯度，并使用更新后的模型继续工作；较慢的工作节点则基于陈旧迭代点进行计算。等它们返回梯度时，服务器可能已经进行了多次更新。因此，只有当服务器收到梯度时才能确定 $\delta^k$ ：此时，服务器既知道工作节点使用的模型迭代点，也知道当前的服务器迭代点，而 $\delta^k$ 就是两者之差。

**局限性**。当工作节点的数据分布存在异质性时，这种方法会面临根本性的偏差问题。较快的工作节点更频繁地发送更新，导致其局部目标函数在优化过程中占据主导地位，并将模型拉向各自的极小点。较慢的工作节点的更新最终到达时，则会将模型推回不同的解。因此，迭代序列可能发生振荡或停滞，无法收敛到驻点。

这种偏差也使理论分析变得困难。经典的 随机梯度下降式证明依赖于每一步都朝着最小化全局函数的方向推进，但在这里，每个更新方向都反映了不同的局部目标函数。若没有额外的数据相似性假设（Mishchenko 等人，2022a；Koloskova 等人，2022年；Nguyen 等人，2022年；Islamov 等人，2024年），就无法将分析推广到全局函数；然而，当不同机器或组织之间的数据可能存在任意异质性时，这类假设往往并不现实。

根本原因在于，每次更新只使用一个工作节点的梯度。更好的策略是整合所有工作节点的信息，即使其中一些梯度是在陈旧迭代点上计算得到的。这一思想促成了增量聚合梯度（IAG）（Blatt 等人，2007年；Gurbuzbalaban 等人，2017年；Vanli 等人，2018年）和随机平均梯度（SAG）（Roux 等人，2012年；Schmidt 等人，2017年）等方法，这些方法会维护并聚合所有工作节点的梯度。

### 3.4.4 IA$^2$随机梯度下降

如上文所述，关键思想是在每次模型更新时利用所有工作节点的信息来构造梯度估计量。 IA$^2$SGD （Wang 等人，2025年）通过在服务器上维护一个梯度表来实现这一点，这与SAG 或IAG类似，但采用异步表更新。

**算法概述**。 服务器维护一个梯度表 $\{g_i\}_{i=1}^n$ ，用于存储从每个 $n$ 工作节点收到的最新梯度。该表通过收集每个工作节点在起始点$x^0$处的一个梯度进行初始化。随后，服务器计算第一次模型更新

$$x^1 = x^0 - \gamma \bar{g} := x^0 - \gamma \frac{1}{n} \sum_{i=1}^n g_i$$

并将其广播 $x^1$ 给所有工作节点。之后，工作节点并行计算随机梯度，并在计算完成后立即异步更新相应的表项。

在每次迭代 $k$中，服务器执行以下步骤:

1. 接收来自工作节点 $i^k$的梯度 $\nabla f_{i^k}\left(x^{k-\delta_{i^k}^k};\ \xi_{i^k}^{k-\delta_{i^k}^k}\right)$

2. 更新梯度表条目： $g_{i^k} = \nabla f_{i^k}\left(x^{k-\delta_{i^k}^k};\ \xi_{i^k}^{k-\delta_{i^k}^k}\right)$

3. 执行模型更新： $x^{k+1} = x^k - \gamma \bar{g} = x^k - \gamma \frac{1}{n} \sum_{i=1}^n g_i$

4. 将更新后的迭代点 $x^{k+1}$ 发送给工作节点 $i^k$ ，供其进行下一次计算

梯度估计量 $\bar{g}$ 结合了每个工作节点最近的梯度，确保尽管采用异步执行，每次更新仍能反映整个工作节点集合的信息。这样，该方法既保留了利用所有工作节点数据的统计优势，又允许各工作节点独立运行，从而避免限制可扩展性的同步瓶颈。

请注意，由于异步性，表中存储的随机梯度通常对应模型的不同迭代点。因此，我们记录每个工作节点 $i$的延迟 $\delta_i^k$，以跟踪其梯度计算时所对应的迭代点。

**理论性能**。该算法的迭代复杂度由Wang等人(2025年)确立，通过直接转换为固定计算时间模型（见第B.3节，我们得到相应的时间复杂度

$$\mathcal{O}\left(\frac{\tau_n L_{\max} \Delta}{\varepsilon}\left(1 + \frac{\sigma^2}{n\varepsilon}\right)\right),$$

这与朴素小批量随机梯度下降的复杂度相当。这表明，仅异步执行并不会比同步方法带来计算优势。因此，梯度延迟如何影响收敛是一个根本性挑战，接下来我们将对此进行探讨。

### 3.4.5 动机

异步算法的主要局限源于梯度延迟，它可能显著降低收敛性能。较大的延迟会使优化步骤沿着次优轨迹进行，从而干扰收敛。

即使在所有函数$f_i$都相等（$f_i \equiv f$对所有 $i \in [n]$）的同质环境中，也会出现这种延迟问题。针对这种情况，最先进的解决方案RingmasterASGD (Maranjyan 等人，2025d）通过显式控制延迟以防止其变大，从而实现最优时间复杂度。RingmasterASGD通过丢弃到达时延迟较大的梯度来实现这一点。

遗憾的是，这种丢弃梯度的策略在IA$^2$SGD的异质环境中行不通。根本问题在于，受计算限制影响，慢速工作节点不可避免地会经历较大的延迟。如果忽略或丢弃它们的延迟梯度，相应的表项就会一直过时；如果后续梯度也延迟到达并被丢弃，这些表项甚至可能再也不会更新。这会造成持续的信息瓶颈，降低梯度估计器的质量并损害收敛。

这个问题表明，我们应通过控制快速工作节点执行的更新次数，来避免此类情况发生。最简单的方法是忽略快速工作节点的一些更新，但这与异步方法的核心原则相矛盾：所有工作节点都持续计算梯度。

相反，我们的方法会缓冲 快速工作节点产生的梯度，而不是立即应用这些梯度，这与MaleniaSGD中的策略类似。通过缓冲梯度，并仅在收集到足够数量的梯度后才进行模型更新，我们能够控制慢速工作节点造成的延迟，同时让所有工作节点持续保持活跃。这种缓冲机制还带来另一项优势：聚合在同一迭代点计算出的多个梯度时，可以得到方差更低的估计，从而提高收敛性。

## 3.5 Ringleader ASGD

现在，我们正式介绍我们的算法RingleaderASGD。

RingleaderASGD 基于现有方法的三项关键见解。首先，受IA$^2$SGD启发，我们维护一张梯度表[4] ，以确保每次更新都纳入所有工作节点的信息，从而无需假设工作节点之间的数据具有相似性。其次，借鉴RingmasterASGD，我们认识到，控制梯度延迟对于实现高效的异步优化至关重要。第三，受MaleniaSGD启发，我们采用缓冲随机梯度的方式，而非丢弃延迟到达的梯度，从而在控制延迟的同时保留有价值的计算，使所有资源都能持续得到利用。

该算法的一个重要特性是，所有工作节点始终持续运行，不断计算随机梯度。工作节点一旦完成梯度计算，便立即将其发送给服务器。回想一下，我们假设

[4]并非一定要在服务器上维护一张表。例如，人工智能$^2$随机梯度下降(Wang 等人，2025年)采用了另一种方案：在每个工作节点上存储一个向量，并在服务器上额外存储一个向量。然而，当工作节点的内存有限时，这种方案可能会带来问题。为清晰起见并简化表述，我们在本文中采用服务器端表述。

通信是瞬时完成的，即耗时为零（第3.3.1节）。服务器收到梯度后，会将其缓存以供后续使用，或者立即应用它来执行模型更新。如果梯度被缓存，则不采取进一步动作，工作节点只需继续计算并发送新的梯度。如果服务器决定执行更新，就会更新模型，并将更新后的模型发送回提供该梯度的工作节点。我们同样假设从服务器到工作节点的通信是瞬时完成的；随后，工作节点便在新的模型点继续计算梯度，从而确保工作节点始终不会处于空闲状态。

算法按轮次进行。每轮恰好执行 $n$ 次模型更新——每个工作节点一次。具体而言，当工作节点发送随机梯度时，服务器可以应用一次更新，并将更新后的模型返回给该工作节点，但服务器会确保每轮中每个工作节点最多收到一次更新后的模型。重复此过程 $n$次，可确保每个工作节点在每轮中恰好获得一个新的模型，从而将延迟控制在有界范围内。为避免丢弃快速工作节点的计算结果，服务器会缓冲它们的梯度，并仅在适当时机应用这些梯度，从而确保每轮恰好包含 $n$ 次更新。

每轮包含两个阶段：

- **阶段 1：** 将随机梯度缓存在表中，直到每个工作节点至少有一个梯度可用。

- **阶段 2：** 恰好执行 $n$ 次更新（每个工作节点一次），然后丢弃表中的旧随机梯度，并返回阶段 1 开始收集新的梯度。

完整算法的形式化描述见算法 3.5.1。

**算法 3.5.1** RingleaderASGD （服务器算法）

**1: 输入：步长** $\gamma > 0$**，初始点** $x^0 \in \mathbb{R}^d$**2: 初始化：将** $x^0$ **广播给所有工作节点，然后各工作节点并行运行算法 3.5.2 3: 设置** $k = 0$**,** $S = \emptyset$**；为所有** $i \in [n]$ **初始化** $G_i = 0$**,** $b_i = 0$ **4: 当尚未终止时执行 5: — 阶段 1：等待所有工作节点发送随机梯度 — 6: 当** $S \neq [n]$ **时执行**

7 $g_j^k \left( x^{k-\delta_j^k} \right)$ $j \in [n]$**: 从**

8

**某个工作节点接收位于 的随机梯度：** $G_j = G_j + g_j^k$**；** $b_j = b_j + 1$**；** $S = S \cup \{j\}$ **9: 结束循环 10: — 阶段 2：为每个工作节点恰好执行一次更新 —**

11 $x^{k+1} = x^k - \gamma \frac{1}{n} \sum_{i=1}^n G_i / b_i$**: 更新模型：：向工作节点**

12

$j$ $x^{k+1}$ **◇广播 阶段 1 中最后完成的工作节点 13:** $k = k + 1$**；** $S = S \setminus \{j\}$ **14: 为所有** $i \in [n]$ **设置** $g_i^+ = 0$**,** $b_i^+ = 0$ **；** $S^+ = \emptyset$ **◇初始化临时缓冲区 15: 当** $S \neq \emptyset$ **时执行 16: 从某个工作节点** $j \in [n]$**接收位于** $x^{k-\delta_j^k}$**) 的随机梯度** $g_j^k$ **(17: 如果** $j \in S$ **则**18 $G_j = G_j + g_j^k$ $b_j = b_j + 1$**：；：更新模型：**

19 $x^{k+1} = x^k - \gamma \frac{1}{n} \sum_{i=1}^n G_i / b_i$

20 $x^{k+1}$ **: 向工作节点** $j$**广播**

**21:** $k = k + 1$**；** $S = S \setminus \{j\}$**22: 否则 23:** $G_j^+ = G_j^+ + g_j^k$**；** $b_j^+ = b_j^+ + 1$**；** $S^+ = S^+ \cup \{j\}$ **◇缓冲区 24: 结束条件判断 25: 结束循环 26: 对所有** $i \in [n]$**执行** $G_i = G_i^+$ **；** $S = S^+$**27: 结束循环**

---

**算法 3.5.2** 工作节点 $i$的子程序

1: **输入：**模型 $x$2: **当**未终止 **时**3: 使用新采样的数据点计算 $g_i = \nabla f_i$ $(x;\ \xi_i)$ $\xi_i \sim \mathcal{D}_i$4: 将 $g_i$ 发送至服务器5: **结束循环**

---

## 3.5.1 详细说明

**初始化。** 算法首先将初始点 $x^0 \in \mathbb{R}^d$广播给所有工作节点，随后这些工作节点开始执行工作节点子程序（算法 3.5.2）。每个工作节点持续在其当前点计算随机梯度，并将其发送给服务器，直到收到停止指令；此时，服务器会提供一个新点以便工作节点继续执行。这种设计确保工作节点始终得到充分利用，绝不会处于空闲状态。

服务器维护着一个梯度表，其中包含条目 $\{(G_i,\ b_i)\}_{i=1}^n$，且所有条目初始值均为零。这里，$G_i$ 用于累积梯度，而 $b_i$ 用于跟踪随机

工作节点 $i$传来的梯度数量，以形成适当的小批量估计量；开始时，所有 $i \in [n]$ 的 $b_i = 0$均为

在阶段 1 开始之前，我们还会初始化集合 $S = \emptyset$，用于跟踪哪些表项至少包含一个随机梯度。由于尚未收到任何梯度， $S$ 为空。

**阶段 1 — 梯度收集**。在此阶段，服务器持续接收来自工作节点的随机梯度，并将其存储在梯度表$\{(G_i,\ b_i)\}_{i=1}^n$中。我们用 $g_j^k$表示工作节点 $j$ 在迭代$k$发送的随机梯度，该梯度是在点 $x^{k-\delta_j^k}$ 处使用独立同分布样本 $\xi_j \sim D_j$计算得到的。

我们不指定延迟 $\delta_j^k$ 如何变化，因为运行算法并不需要这些信息：必要时，可以通过当前模型迭代点与计算随机梯度时的迭代点之差得到延迟。延迟只会在分析中发挥作用，而不会影响方法的执行。

收到 $g_j^k$ 来自工作节点 $j$ （第7行）后，服务器按如下方式更新相应的表项和随机梯度计数器（第 8行）

$$G_j = G_j + g_j^k\ ,\quad b_j = b_j + 1\ ,\quad S = S \cup \{j\}\ .$$

此过程持续到 $S = [n]$，也就是说，服务器已从每个工作节点至少收集到一个梯度。在此阶段不会执行模型更新，工作节点也不会收到新的点；因此，给定工作节点的所有随机梯度都是在同一点计算的。

**阶段 2 — 顺序更新**。 在此阶段，服务器会为每个工作节点恰好执行一次模型更新 $i$，共计 $n$ 次更新。阶段 2 从最后一个完成阶段 1 的工作节点开始，也就是其梯度使表完整的工作节点。服务器首先通过对表 $\{(G_i, b_i)\}_{i=1}^n$ 中累积的随机梯度求平均值来计算一次更新11，并使用该估计值执行下降步（第 12 行）。随后将更新后的模型发送给该工作节点（第 $S$ 行），并将该工作节点从集合中移除，同时递增迭代计数器（第 13 行）。

接下来，服务器必须更新剩余的 $n-1$ 个工作节点。这些更新会在每个工作节点完成当前计算后立即按顺序执行。在此等待期间，来自不属于 $S$ 的工作节点的新梯度可能会到达——例如，最后一个已更新的工作节点可能会在其他工作节点完成计算之前再发送一个随机梯度。丢弃这些梯度会浪费信息，因此会将它们缓冲起来，留待下一轮使用。

**临时表管理**。 为此，服务器维护一个临时表 $\{(G_i^+、b_i^+)\}_{i=1}^n$，将其初始化为零（第 14 行），并配有一个集合$S^+$，用于记录哪些工作节点已向该表贡献数据。每当来自不在 $S$ 中的工作节点的梯度到达时，便会将其存储在临时表中（第 23 行）。

如果梯度来自某个工作节点 $j \in S$——也就是说，来自仍需更新其模型的工作节点之一——服务器会先用这个新的随机梯度更新主表$\{(G_i,\ b_i)\}_{i=1}^n$ （第 18 行）。随后，它再次对累积的随机梯度求平均值，从而执行模型更新，更新

表（第 19 行），将新模型广播给工作节点 $j$ （第20 行），递增迭代计数器，并从集合 $j$ 中移除 $S = S \setminus \{j\}$ （第 21 行）。

**为下一轮做准备**。当 $S$ 中的所有工作节点都已更新且阶段 2 已完成（$S = \emptyset$）后，服务器会将临时表 $\{(G_i^+, b_i^+)\}_{i=1}^n$ 中的内容复制到主表$\{(G_i, b_i)\}_{i=1}^n$ （第 26 行），从而为下一轮做准备。集合 $S$ 也会重置为 $S = S^+$，因为这些工作节点已在其更新后的模型上贡献了梯度。主表中与不在 $S^+$ 中的工作节点对应的条目仍为零，因为临时表在阶段 2 开始时已初始化为零（第 14 行）。

服务器现在可以返回阶段 1 并开始新的梯度收集阶段，从而进入下一轮。

### 3.5.2 延迟控制分析

RingleaderASGD采用两个阶段的结构，专门用于防止标准异步方法中出现的无界延迟。要理解其原理，请注意，在每一轮中，我们都会执行恰好 $n$ 次更新——每个工作节点一次——然后进入下一轮。这确保任何工作节点相对于当前迭代都不会落后超过整整一轮。延迟的精确界限由以下引理给出

**引理 3.5.1** （有界延迟）。 在 RingleaderASGD （算法 3.5.1）中，延迟 $\delta_i^k$ 满足

$$\delta_i^k \leq 2n - 2 \,,$$

对于任意工作节点 $i \in [n]$ 和任意迭代 $k \geq 0$。

证明。 我们通过分析RingleaderASGD的结构来证明这一点。该算法按轮次运行，每一轮由阶段 1（梯度收集）和阶段 2（顺序更新）组成。在每个阶段 2 中，服务器恰好执行 $n$ 次更新，每个工作节点执行一次。阶段 2 从迭代 0、 $n$、 $2n$、$3n$、.....开始，即从 $n$的倍数处开始。

**第一轮（迭代 0 至 $n-1$）：** 起初，所有工作节点都在点 $x^0$处计算梯度，因此在迭代 0、1、.....、 $n-1$期间，服务器接收到的是在 $x^0$处计算的梯度。对于该范围内的任意迭代 $k$，服务器处理的是在点 $x^{k-\delta_i^k}$处计算的随机梯度，因此 $\delta_i^k = k \leq n-1$。所以，在第一个阶段 2 期间，延迟只会递增。

在本轮结束时，每个工作节点 $i$ 都已收到一个新模型 $x^j$，用于某些 $j \in \{1, 2, \ldots, n\}$，并且这些更新迭代在各工作节点之间各不相同。

**第二轮（迭代 $n$ 至 $2n-1$）：** 在第二个阶段 2 开始时（迭代 $n$），梯度表包含在点 $x^{n-\delta_i^n}$ 处为工作节点 $i$计算的梯度，其中 $\delta_i^n \in \{0, 1, \ldots, n-1\}$。这些延迟值在各工作节点之间各不相同，因为每个工作节点在上一轮的不同迭代中收到了其更新。

在迭代 $n$ 到 $2n-1$期间，由于与第一阶段 2 中相同的原因，这些延迟在每次迭代时都会增加 1，因此在本轮结束时得到 $\delta_i^{2n-1} \in \{n-1, n, \ldots, 2n-2\}$。

与此同时，所有工作节点都会收到新的点来计算梯度，因此在下一个阶段 2 期间，所有工作节点的延迟将再次各不相同，并且为 $\{0, 1, \ldots, n-1\}$。

**一般模式：** 通过归纳法可知，在从迭代 $cn$（对于整数 $c \geq 1$）开始的每一轮次开始时，延迟 $\delta_i^{cn}$在 $\{0, 1, \ldots, n-1\}$ 中取互不相同的值。在每个阶段 2 期间，这些延迟最多增加 $n-1$，因此有界为

$$\delta_i^k \leq (n-1) + (n-1) = 2n - 2 \, .$$

□

### 3.5.3 与 IA²随机梯度下降的比较

我们的方法是延迟控制版本的 IA²随机梯度下降。通过移除阶段 1（梯度收集）和阶段 2（结构化更新），我们可以还原 IA²随机梯度下降，从而采用朴素方式执行更新——在梯度到达后立即更新。相比之下，我们的算法以结构化轮次运行，在每个轮次中对每个工作节点恰好执行一次更新，从而提供 IA²随机梯度下降 所缺乏的关键延迟控制。

在 IA²随机梯度下降 中，慢速工作器的延迟可能无限增长，因为服务器持续使用快速工作器的梯度更新模型，导致慢速工作器越来越落后。我们的方法通过缓冲快速工作器的梯度，而不是立即应用这些梯度，避免了这一问题，从而确保所有工作节点都能在后续 $n$ 次迭代内收到更新后的模型。

### 3.5.4 与 MaleniaSGD 的比较

MaleniaSGD 同样是一种分为两个阶段的算法。在阶段 1 中，Malenia SGD 采用与我们的方法类似的方式收集梯度，但使用了不同的终止条件（3.3），该条件需要已知噪声参数 $\sigma$ 和目标平稳性水平 $\varepsilon$，因此不切实际。在阶段 2，MaleniaSGD 通过对收集到的所有梯度取平均值来执行同步更新，并在返回阶段 1 之前，同时向所有工作节点广播新模型。这种同步会迫使 MaleniaSGD 丢弃广播期间仍在活跃工作节点上进行的梯度计算。

相比之下，我们的方法以异步地方式执行阶段 2：随着工作节点完成当前的梯度计算，我们依次更新这些工作节点，从而确保不会浪费任何计算工作。

关于MaleniaSGD的终止条件（3.3），我们在第B.4 节中证明，该条件可以替换为一个更简单的要求：从每个工作节点至少获取一个梯度。经过这一修改，MaleniaSGD在固定时间区间（3.2）内仍保持最优的，同时实现无参数化，从而无需先验知识 $\sigma$ 和 $\varepsilon$。在这一无参数变体中，MaleniaSGD 与Ringleader ASGD 之间的唯一区别在于阶段 2：我们异步地执行更新而不丢弃梯度，而 MaleniaSGD 则以同步方式运行。

## 3.6 理论结果

在介绍理论结果之前，我们首先以紧凑形式写出该算法。表中每个工作节点的梯度都是在同一点计算的；对于工作节点 $i$ 在第 $k$ 次迭代时，该点为 $x^{k-\delta_i^k}$。更新规则可以简洁地写成

$$x^{k+1} = x^k - \gamma \bar{g}^k \ ,$$

其中，梯度估计量 $\bar{g}^k$ 定义为

$$\bar{g}^k := \frac{1}{n}\sum_{i=1}^{n} \bar{g}_i^k := \frac{1}{n}\sum_{i=1}^{n} \frac{1}{b_i^k}\sum_{j=1}^{b_i^k} g_i^{k,j} \ .$$

由于可能从同一个工作节点收到多个梯度，我们用 $g^{k,i}{}^{j}$ 表示迭代 $k$时来自工作节点 $i$ 的第 $j$个梯度。这里，索引 $j$ 对应于独立同分布采样的数据点，具体而言

$$g_i^{k,j} := \nabla f_i\left(x^{k-\delta_i^k}; \xi_i^{k-\delta_i^k,j}\right) \ .$$

量 $b_i^k$表示迭代 $k$时存储在表中的来自工作节点 $i$的梯度数量，即第 $b_i$ 行 11 和 19 中的值。因此，$(b_i^k, \delta_i^k)$ 这一对完全决定了该方法在迭代 $k$ 时的行为。

请注意，序列 $\{b_i^k\}$ 仅取决于计算时间 $\{\tau_i\}$ 和算法设计（即收集梯度的停止规则）。这些因素一旦确定，对于每个 $i \in [n]$ 和迭代 $k$，所有 $b_i^k$ 都已确定。关键的是，$b_i^k$ 的值不取决于该方法的超参数 $\gamma$、$x^0$，也不取决于方差参数 $\sigma$ 或平稳性水平 $\varepsilon$。

### 3.6.1 迭代复杂度

我们的收敛性分析遵循Maranjyan 等人(2025d采用的结构化方法，将证明分解为两个关键部分：用于跟踪目标函数进展的下降引理，以及用于控制系统中累积延迟的残差估计引理。

我们首先为各轮次批次大小的调和平均数建立记号：

$$B^k = \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{b_i^k}\right)^{-1}, \quad \text{and} \quad B = \inf_{k\geq 0} B^k \ .$$

注意，$B \geq 1$，因为根据算法的设计，每个 $b_i^k \geq 1$。稍后将在引理 $B$ 中建立关于3.6.4的更紧界。

第一个引理量化了目标函数在每次迭代中的下降幅度，同时考虑标准梯度下降带来的进展，以及异步更新引入的额外复杂性。

**引理 3.6.1**（下降引理；证明见第 B.2.3节。在假设 3.3.1和 3.3.2下，如果算法 3.5.1 中的步长满足 $\gamma \leq 1/4L$，则有以下

不等式成立

$$\mathbb{E}\left[f\left(x^{k+1}\right)\right] \leq \mathbb{E}\left[f\left(x^{k}\right)\right] - \frac{\gamma}{2}\mathbb{E}\left[\left\|\nabla f\left(x^{k}\right)\right\|^2\right] - \frac{\gamma}{4}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] + \frac{\gamma L^2}{2n}\sum_{i=1}^{n}\mathbb{E}\left[\left\|x^k - x^{k-\delta_i^k}\right\|^2\right] + \frac{3\gamma^2 L\sigma^2}{2B} + \gamma^2 L \sum_{\ell=k-(k \bmod n)}^{k-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{\ell-\delta_i^\ell}\right)\right\|^2\right].$$

这个下降引理与Maranjyan 等人（2025d）在同质情形下分析的对应引理结构相似，但多了一个关键项。上界中的最后一个求和项反映了使用梯度表中的陈旧梯度所产生的影响——我们将这一现象称为“表延迟”。同质情形下不存在这一项，因为无需维护梯度表。事实上，当 $n = 1$时，我们的设定就退化为同质情形，梯度表也变得多余，这个附加项随之消失，从而恢复了Maranjyan 等人（2025d）提出的原始下降引理。

接下来，与Maranjyan 等人（2025d）的工作类似，我们推导一个引理，以界定涉及当前点与旧点之差的项。

**引理 3.6.2** （残差估计；证明见第B.2.4节）。在假设3.3.1下，RingleaderASGD （算法3.5.1）的迭代序列在步长 $\gamma \leq 1/4nL$下满足以下界

$$\frac{1}{K}\sum_{k=0}^{K-1}\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\left\|x^k - x^{k-\delta_i^k}\right\|^2\right] \leq \frac{2\gamma n}{LK}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^{n}\nabla f_j\left(x^{k-\delta_j^k}\right)\right\|^2\right] + \frac{2\gamma\sigma^2}{LB}.$$

最后，结合这两个引理，我们得到迭代复杂度。

**定理 3.6.3 (迭代复杂度)。** 在假设 3.3.1、3.3.2 和3.3.5成立的条件下，令 RingleaderASGD （算法 3.5.1）中的步长为

$$\gamma = \min\left\{\frac{1}{8nL}, \frac{\varepsilon B}{10L\sigma^2}\right\}.$$

那么，

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla f\left(x^{k}\right)\right\|^2\right] \leq \varepsilon\,,$$

for

$$K \geq \frac{32nL\Delta}{\varepsilon} + \frac{40L\Delta\sigma^2}{B\varepsilon^2} = \mathcal{O}\left(\frac{nL\Delta}{\varepsilon}\left(1 + \frac{\sigma^2}{Bn\varepsilon}\right)\right).$$

证明。 我们首先对引理 3.6.1在 $K$ 次迭代上的不等式取平均

并将两边除以 $\gamma/2$

$$\begin{aligned}\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right]+\frac{1}{2K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right]\\ \leq\frac{2\Delta}{\gamma K}+\frac{3\gamma L\sigma^2}{B}\\ +\frac{L^2}{n}\frac{1}{K}\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}\left[\left\|x^k-x^{k-\delta_i^k}\right\|^2\right]\\ +2\gamma L\frac{1}{K}\sum_{k=0}^{K-1}\sum_{\ell=k-(k \bmod n)}^{k-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{\ell-\delta_i^\ell}\right)\right\|^2\right].\end{aligned}$$

现在，我们利用引理3.6.2对右侧第三项进行界定。

$$\begin{aligned}\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right]+\frac{1}{2K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right]\\ \leq\frac{2\Delta}{\gamma K}+\frac{3\gamma L\sigma^2}{B}+\frac{2\gamma L\sigma^2}{B}\\ +2\gamma Ln\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^{n}\nabla f_j\left(x^{k-\delta_j^k}\right)\right\|^2\right]\\ +2\gamma L\frac{1}{K}\sum_{k=0}^{K-1}\sum_{\ell=k-(k \bmod n)}^{k-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{\ell-\delta_i^\ell}\right)\right\|^2\right]\\ \leq\frac{2\Delta}{\gamma K}+\frac{5\gamma L\sigma^2}{B}\\ +2\gamma Ln\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^{n}\nabla f_j\left(x^{k-\delta_j^k}\right)\right\|^2\right]\\ +2\gamma Ln\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right].\end{aligned}$$

现在，利用界 $\gamma\leq 1/8nL$，我们得到

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right]\leq\frac{2\Delta}{\gamma K}+\frac{5\gamma L\sigma^2}{B}.$$

最后，代入步长以及 $K$ 的表达式，可确保右侧不超过 $\varepsilon$。

□

对于并行和异步方法而言，迭代复杂度不如时间复杂度重要。真正重要的是我们能多快完成训练。只要能更快完成整个过程，我们愿意执行更多迭代并进行额外计算。在确定了迭代复杂度后，我们

现在转而讨论时间复杂度。

### 3.6.2 时间复杂度

由于该算法按轮次运行，每轮包含 $n$ 步，而且其迭代复杂度已知，因此接下来需要确定每轮的持续时间。我们有以下引理

**引理 3.6.4.** 每一组 $n$算法3.5.1 的连续迭代（每一轮）最多需要 $2\tau_n$ 秒。此外，我们有

$$B \geq \frac{\tau_n}{2}\left(\frac{1}{n}\sum_{i=1}^{n}\tau_i\right)^{-1} = \frac{\tau_n}{2\tau_{\text{avg}}}\,.$$

证明。 $2\tau_n$ 的上界可由算法的结构直接得出，该算法包含两个阶段。在第一阶段，服务器等待所有工作节点至少完成一次梯度计算，这最多需要 $\tau_n$ 秒。在第二阶段，服务器应用收到的梯度，并等待所有正在进行的计算结束——这同样最多需要 $\tau_n$ 秒。因此， $n$ 次迭代的总时间不超过 $2\tau_n$。

现在我们证明引理的第二部分。回顾

$$B = \inf_{k\geq 0} B^k = \inf_{k\geq 0}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{b_i^k}\right)^{-1} \geq \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{b_i}\right)^{-1},$$

其中我们定义

$$b_i = \inf_{k\geq 0} b_i^k\,.$$

我们关注迭代$k$时表中存储的梯度数量。该数量包括阶段 1 期间计算的梯度，以及阶段 2 额外计算的一个梯度（阶段 1 最后完成的工作节点除外）。

由于每个工作节点在阶段 1 期间都至少需要计算一个梯度，因此速度最慢的工作节点需要花费 $\tau_n$ 秒完成一次梯度计算。在这段持续 $\tau_n$ 秒的区间内，速度较快的工作节点 $i < n$ 可能仍在完成上一轮阶段 2 的梯度，然后才开始计算当前轮次阶段 1 的梯度。

完成所有剩余的阶段 2 工作后（最多需要 $\tau_i$ 秒），工作节点 $i$ 至少还剩 $\tau_n - \tau_i$ 秒，可为当前轮次的阶段 1 计算额外的梯度。工作节点$i$ 能够计算的梯度数量满足

$$b_i \geq \max\left\{1, \left\lceil\frac{\tau_n - \tau_i}{\tau_i}\right\rceil\right\} \geq \max\left\{1, \frac{\tau_n}{\tau_i} - 1\right\}.$$

对于满足 $\tau_n \geq 2\tau_i$ 条件的工作节点 $i$，我们有

$$\frac{\tau_n}{\tau_i} - 1 \geq \frac{\tau_n}{2\tau_i}\,,$$

因此

$$b_i \geq \frac{\tau_n}{2\tau_i} .$$

将此界代入 $B$ 的表达式，即可得到所声称的结果。 □

基于此引理，我们推导出该算法的时间复杂度保证

**定理 3.6.5。** 设假设3.3.2、3.3.5和3.3.1 均成立。设Ringleader ASGD （算法3.5.1）中的步长为 $\gamma = \min\left\{1/8nL, \varepsilon B/10L\sigma^2\right\}$。那么，在固定计算模型 （3.2）下，RingleaderASGD 达到最优时间复杂度

$$\mathcal{O}\left(\frac{L\Delta}{\varepsilon}\left(\tau_n + \tau_{\text{avg}}\frac{\sigma^2}{n\varepsilon}\right)\right).$$

证明。 我们从定理3.6.3的迭代复杂度出发

$$K \geq \frac{32nL\Delta}{\varepsilon} + \frac{40L\Delta\sigma^2}{B\varepsilon^2} = \mathcal{O}\left(\frac{nL\Delta}{\varepsilon}\left(1 + \frac{\sigma^2}{Bn\varepsilon}\right)\right).$$

执行 $n$ 步所需的时间至多为 $2\tau_n$ 根据引理 3.6.4。因此，时间复杂度为

$$2\tau_n \times \frac{K}{n} = \mathcal{O}\left(\tau_n\frac{L\Delta}{\varepsilon}\left(1 + \frac{\sigma^2}{Bn\varepsilon}\right)\right).$$

剩下的工作是代入 $B \geq \tau_n/2\tau_{\text{avg}}$ ，该项来自引理 3.6.4。 □

所得的时间复杂度由两个关键项组成，揭示了算法的行为。第一项取决于最慢的设备，这是根本性的，因为所有设备都必须参与解决问题。然而，第二项涉及 $\tau_{\text{avg}}$ ，而不是 朴素小批量随机梯度下降 中的 $\tau_n$ （见表 3.1）——这一替换体现了异步执行的核心优势。具体而言，当 $\sigma$ 相对较大时，这一优势会更加明显。直观地说，在高噪声情形下，从工作节点收集大量梯度对于收敛至关重要，而异步方法能够更有效地利用速度较快的工作节点。相反，在低噪声情形下，较少的梯度计算就足以取得良好性能，因此<style id='8'>朴素小批量随机梯度下降 已经相当有效，使得异步性的额外复杂度变得没有必要。

这一时间复杂度结果与 Tyurin &Richt´arik （2024年）推导出的下界相匹配，从而使 RingleaderASGD 成为首个实现最优性的异步算法。

## 3.7 实验

为验证我们的理论结果，我们进行了一项玩具模拟。

我们研究 MNIST（LeCun 等人，2010）和 Fashion-MNIST（Xiao 等人，2017年）上的图像分类任务，并对两个数据集采用标准归一化。为使客户端大小相同，我们首先裁剪数据集，使样本总数能够被客户端数量整除： $n = 100$。为在各客户端之间构造异构数据集，随后我们使用浓度参数为 $\alpha = 0.1$（Yurochkin 等人，

2019年）。对于每个客户端 $j \in [n]$，我们在各类别上抽取比例 $p_j \sim$狄利克雷分布($\alpha$，……，$\alpha$)，并计算一个取整后的类别分配向量，使其各项之和恰好等于 $N/n$，其中 $N$ 为裁剪后的数据集大小。由此得到非独立同分布数据，每个客户端的类别分布都存在偏斜（当 $\alpha = 0.1$时，客户端通常只会频繁观察到 1–2 个类别）。

分配样本时，我们从每个类别样本池中为客户端 $j$取出指定数量的样本。如果某个类别样本池中剩余的样本不足以满足指定数量，客户端将获得该类别中剩余的全部样本，缺少的部分则通过补充 其他仍有可用样本的类别中的样本来填足。

我们的模型是一个两层多层感知机 Linear($d$，128) $\to$ReLU $\to$Linear(128，10)，使用均值交叉熵进行训练。客户端上的随机梯度使用大小为 4 的小批次计算，而报告的梯度范数则在完整 数据集上计算。

我们通过为每个客户端分配 $i$ 一个基础延迟和随机抖动，来模拟异构计算：

$$\tau_i = i + |\eta_i| , \qquad \eta_i \sim \mathcal{N}(0, i) , \quad \text{for all } i \in [n] .$$

对于每种方法，我们在固定的实际时间预算内调整步长 $\gamma$ 。我们遍历

$$\gamma \in \{0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0\} ,$$

然后为每种方法固定最佳的 $\gamma$，用于评估。

我们报告全批量梯度范数平方 $\|\nabla f(x^k)\|^2$随实际运行时间的变化。每种方法使用不同的随机种子运行 30 次。我们报告中位数和四分位距（*IQR*）。为减少高频噪声，我们对聚合后的曲线应用居中移动平均平滑（聚合后处理），同时保持初始点不变。

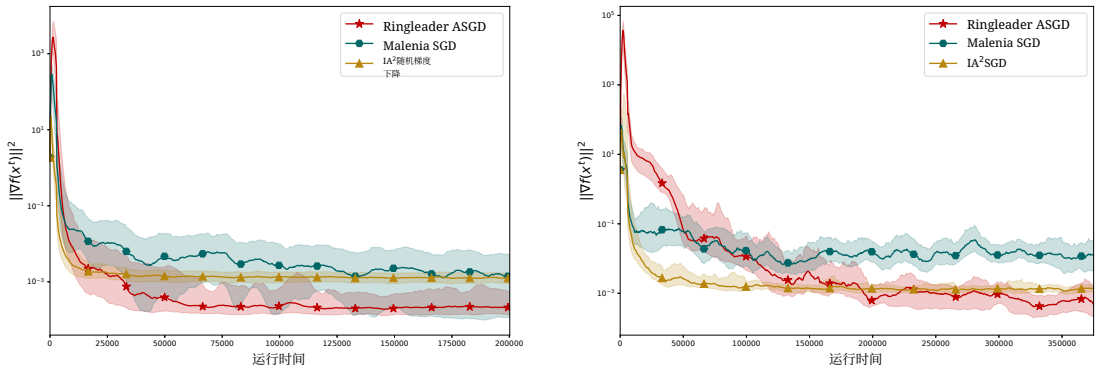
图3.7.1 展示了结果。我们观察到，与Malenia SGD 和IA$^2$SGD相比，RingleaderASGD 收敛更快。尽管RingleaderASGD 和MaleniaSGD 的理论时间复杂度相同，但在实践中，Ringleader ASGD表现更好，因为阶段 2 采用了 $n$异步更新，而非单次同步更新。这种设计使其能够在相同的实际时间预算内执行更多优化步骤，在更新稀疏时尤其有利。

## 3.8 结论

我们提出了RingleaderASGD，这是首个在任意数据异质性和分布式学习中任意异质的计算时间条件下，无需工作节点数据集之间的相似性假设，即可实现最优时间复杂度的异步随机梯度方法。

其核心创新是在每一轮中采用两阶段结构：每个工作节点更新一次模型（总计 $n$次更新），同时通过缓冲机制管理梯度延迟，并保持异步执行的效率。通过维护梯度表，并交替进行梯度收集和顺序更新，RingleaderASGD 避免了朴素异步方法中常见的无界延迟。服务器收到的每个梯度要么在当前轮次中使用，要么存储起来以备后续使用，从而确保不浪费任何计算。

我们的分析表明，RingleaderASGD 达到了最优时间复杂度




（a）MNIST　　（b）Fashion-MNIST

图 3.7.1：收敛比较，显示梯度范数平方的中位数$\|\nabla f(x^k)\|^2$（实线）及四分位距（阴影区域）随实际运行时间的变化，结果取 30 个随机种子的平均值。**设置：**使用架构为 Linear($d$，128) $\to$ ReLU $\to$ Linear(128，10) 的两层多层感知机，在（a）MNIST 和（b）Fashion-MNIST 数据集上进行训练。**客户端延迟：**异构延迟模拟为 $\tau_i = i + |\eta_i|$，其中每个客户端的 $\eta_i \sim \mathcal{N}(0,\ i)$ 为 $i \in [n]$，且 $n = 100$。**结果：**在经过最优调优的步长下，RingleaderASGD 的收敛速度快于MaleniaSGD 和IA$^2$SGD，尽管Ringleader ASGD 和MaleniaSGD 具有相同的理论时间复杂度保证。

由 Tyurin & Richt´arik (2024年)确立的界限。与最优但需要同步执行的 MaleniaSGD 方法不同，RingleaderASGD 是异步的，在算法设计中无需预先了解问题参数，因此适用于实际部署。

最后，经过小幅修改，RingleaderASGD 在计算时间任意变化这一更一般的情形下也能达到最优性（第B.1节）。

# 第4章

# ATA：面向分布式机器学习的高效资源管理的自适应任务分配

本章节基于Maranjyan 等人（2025b）的研究：

> Artavazd Maranjyan、El Mehdi Saad、Peter Richt´arik 和 Francesco Orabona。《ATA：分布式机器学习的自适应任务分配与高效资源管理》。载于国际机器学习会议，2025b

## 4.1 引言

在这项工作中，我们研究了一个非常普遍但又基础且重要的问题，该问题存在于各种情境和领域中。具体而言，有 $n$ 个工作节点/节点/设备协作运行某种迭代算法，该算法具有以下结构：

- 为了执行算法的一次迭代，需要完成一定数量（$B$）的任务。

- 每项任务都可以由任意工作节点计算，且任务之间没有时间上的关联。也就是说，任务可以按任意顺序计算，也可以并行计算，等等。
- 每当要求某个工作节点执行一项任务时，该任务都需要一定的时间才能完成；这一时间被建模为一个非负随机变量，其取值来自专属于该工作节点的未知分布。随机假设是合理的，因为在实际系统中，计算时间并非固定不变，每次迭代都可能有所不同（Dean 与 Barroso，2013年；Chen 等人，2016年；Dutta 等人，2018年；Maranjyan 等人，2025a）。
- 每个工作节点一次只能处理一项任务。也就是说，工作节点必须依次处理所有需要执行的任务。不同的工作节点则并行工作。

### 4.1.1 贪心任务分配

在这种情况下，一个自然的目标是确保尽可能快地完成所有任务（以期望时间衡量）。如果任务完成时间是每次迭代的主要耗时因素，这就能缩短（预期）完成算法一次迭代所需的时间。只要我们愿意浪费资源，这个问题就有一个简单的解决方案：采用贪心任务分配（GTA）策略，遵循以下原则：确保所有工作节点始终都在执行某项任务，并在 $B$ 项任务完成后停止。在GTA中，我们最初让所有 $n$ 个工作节点开始执行任务；一旦有工作节点

完成一项任务，我们就让它开始执行另一项任务。重复这一过程，直到 $B$ 项任务全部完成。

### 4.1.2 GTA 的浪费

虽然 GTA 能最大限度地缩短完成时间，但就收集全部 $B$ 项任务所需的工作者利用总时间而言，它可能造成极大的浪费。事实上，考虑有 $n = 1{,}010$ 个工作节点和 $B = 10$ 项任务的情形。在这种情况下，GTA 会导致每次迭代至少有 $n - B = 1{,}000$ 项不必要的任务被执行！在工作节点还要承担运行上述迭代算法之外的其他多项工作的情况下，这一点尤其不可取。

### 4.1.3 本文的目标

我们的目标是设计新的任务分配策略，并为其提供严谨的理论支持，在完全消除这种浪费的*constraint* 条件下，尽可能缩短预期完成时间。也就是说，我们确保每轮完成的任务不超过 $B$ 项。

### 4.1.4 一个启发性示例：最优并行 SGD

我们这项工作的一个重要灵感来源，也是上述一般任务收集问题的典型示例，与并行随机梯度下降（SGD）方法领域的近期进展有关。考虑寻找优化问题的近似驻点这一问题：

$$\underset{x\in\mathbb{R}^d}{\text{minimize}}\left\{f(x) \coloneqq \mathbb{E}_{\xi\sim\mathcal{D}}\left[f_\xi(x)\right]\right\},$$

其中 $f_\xi : \mathbb{R}^d \to \mathbb{R}$ 是光滑非凸函数，并假设 $f$ 有下界。我们假设

$$\mathbb{E}_{\xi\sim\mathcal{D}}\left[\left\|f_\xi(x) - \nabla f(x)\right\|^2\right] \leq \sigma^2$$

对所有 $x \in \mathbb{R}^d$均成立。

在最近的一项突破性研究中，Tyurin & Richt´arik （2024年）开发了一种并行随机梯度下降方法。就一种称为时间复杂度的新型复杂度概念而言，该方法对于使用 $n$ 个并行工作节点解决上述问题是最优的，其假设为：工作节点 $i$ 计算 $f$ 的一个随机梯度需要$\tau_i > 0$ 秒（这对应于一个任务）。他们的方法RennalaSGD相当于批量大小为 $B$ 的小批量随机梯度下降（该批量大小仅取决于目标精度和 $\sigma$ ），其中 $B$ 个任务（随机梯度）通过GTA完成。尽管就时间复杂度而言达到了极小极大最优，但RennalaSGD中采用的GTA 任务分配策略可能造成浪费，具体如上文所述。

最近，Maranjyan 等人(2025d) 提出了RingmasterASGD，这是一种完全异步的随机梯度下降方法，其时间复杂度与RennalaSGD相匹配，并且对于与任务（随机梯度）相关的任意计算时间模式（包括我们设定中考虑的随机模式），都能达到最优性。然而，RingmasterASGD 也采用了贪心任务分配策略，导致资源浪费。

许多其他并行/分布式方法也涉及任务分配策略的实施，包括随机近端点方法（任务 =对随机近端算子的评估）、高阶方法（任务 =对随机海森矩阵的评估）等。因此，通过解决一般的任务分配问题，我们旨在减少所有这些方法固有的资源浪费。

### 4.1.5 贡献

在本文中，我们将任务分配问题形式化为一个具有部分反馈和非线性损失的组合在线学习问题。随后，我们提出ATA，这是一种基于下置信界的算法，旨在解决所提出的分配问题。ATA 不依赖于工作节点的计算时间；我们的理论分析表明，我们的方法所达到的总计算时间与最优计算时间（即在充分了解工作节点的臂分布时可达到的计算时间）相比，仅相差一个较小的乘法因子。此外，我们还提出了ATA-Empirical，这是ATA的一个变体，它利用一种新颖的数据依赖集中不等式，并取得了更好的经验结果。最后，我们通过数值模拟验证了所提出的方法。

## 4.2 相关工作

大多数关于异步方法的文献都着重展示其相较于同步对应方法的优势。对于最简单的方法，随机梯度下降，这一点直到最近才由Tyurin& Richt´arik(2024年)确立。随着这一结果的确立，学界如今可以将重点转向降低异步性的开销。我们的工作或许是朝这一方向迈出的第一步。

在联邦学习（人工智能）中（Kone˘cn´y 等人，2016b；McMahan 等人，2016年；Kairouz 等人，2021年），已有多项研究考虑了系统异构性。最为人熟知的联邦学习方法 FedAvg （McMahan 等人，2017年）通过让工作节点执行多个本地步数来运行，其中每一步都可视为一个任务。一些研究根据工作器计算时间调整本地步数（Li 等人，2020b；Maranjyan 等人，2025c），从而有效地根据工作节点的速度调整任务分配。然而，这些方法依赖于对这些时间的先验知识，而不是像我们这样对其进行自适应学习。

我们将问题重新表述为一个在线赌博机问题。关于老虎机算法的文献十分丰富，读者可参阅 Lattimore &Szepesv´ari （2020年）对该主题的介绍。我们的算法基于对臂的真实均值使用下置信界（LCB）。这一思想最初由Auer （2002年）针对经典多臂老虎机（MAB）设置提出，此后已广泛应用于随机组合赌博机文献中（Gai等人，2012年；Chen 等人，2013年；Combes等人，2015年；Kveton等人，2015年）。与均值的经验估计相比，使用下置信界能够在探索与利用之间实现最优的权衡。

我们采用的“贪心”方法，是根据下置信界而非未知均值，选择使损失函数最小化的动作。这是文献中的一种标准技术（Chen 等人，2013年；Lin 等人，2015年）。不过请注意，我们更大的动作空间以及损失函数的不连续性

使得我们需要进行更有针对性的分析。据我们所知，这是首篇研究随机组合多臂老虎机框架中非连续损失函数的工作。为应对这一挑战，我们利用损失函数和动作空间的特定结构，控制选择次优动作的轮次数。此外，我们的方法在计算上十分高效。

## 4.3 问题设置

本节将正式描述问题设置。

### 4.3.1 任务分配协议

我们考虑一个由 $n$个工作节点组成的系统，每个工作节点都负责重复执行同一项任务（例如，计算梯度）。在每一轮中，分配算法的预算为 $B$ 个单位，必须将其分配给 $n$ 个工作节点。分配一个单位相当于执行一次该任务。

设 $K$ 表示轮次总数，并假设学习者不知道该数值。对于工作节点 $i \in [n]$: $= 1$，2，...，$n$，设 $X^{k,\ u}_{i}$ 为其在第 $k \in [K]$ 轮完成第 $u$项任务所需的计算时间。因此，工作节点 $i$ 在第 $k$ 轮执行 $a_i^k$ 项任务所需的总计算时间为

$$\sum_{u=1}^{a_i^k} X_i^{k,u}$$

约定当 $a_i^k = 0$时，该总和等于0。

在每一轮 $k$，算法仅根据截至第 $k-1$ 轮可获取的信息，选择一个分配向量 $a^k = (a_1^k, \ldots, a_n^k) \in \mathbb{Z}_+^n$，使得 $\left\|a^k\right\|_1 = \sum_{i=1}^n a_i^k = B$。反馈包括所有所选工作节点的 $a_i^k$ 观测时间。我们将动作集记为

$$\mathcal{A} := \left\{a \in \mathbb{Z}_+^n : \|a\|_1 = B\right\} .$$

每一轮 $k$ 中，分配策略的目标是最小化总计算时间。形式化地，我们定义

$$C : \mathcal{A} \to \mathbb{R}_+ ,$$

为优化器必须等待接收 $B$ 个已完成任务（例如梯度）所需的计算时间。在给定分配向量 $a \in \mathcal{A}$ 的情况下，该时间定义为

$$C(a^k) := \max_{i \in \mathrm{supp}(a^k)} \sum_{u=1}^{a_i^k} X_i^{k,u} . \tag{4.1}$$

### 4.3.2 建模假设

我们假设每个工作节点 $i \in [n]$ 的计算时间均为独立同分布，它们取自一个随机变量 $X_i$，该随机变量服从概率分布 $\nu_i$。我们用

$\mu = (\mu_1、\ldots\ldots、\mu_n)$ 表示未知均值向量。因此，随机变量 $X^k{}_i{}^u$（其中 $u \in \{1, \ldots, a_i^k\}$）是从 $\nu_i$ 中抽取的独立同分布样本 $a_i^k$。

我们假设计算时间的分布 $\nu_i$ 是次指数随机变量。为量化这一假设，我们回顾中心化实值随机变量 $X$的次指数范数（亦称 Orlicz 范数）的定义：

$$\|X\|_{\psi_1} := \inf \left\{ C > 0 : \mathbb{E}\left[\exp\left(\frac{|X|}{C}\right)\right] \leq 2 \right\} . \tag{4.2}$$

因此，我们正式作出如下假设。

**假设 4.3.1.** 对于每个 $i \in [n]$， $X_i$ 都是一个正随机变量，且满足

$$\|X_i - \mu_i\|_{\psi_1} \leq \alpha_i , \quad \alpha_i \geq 0 .$$

在本文余下部分，我们记 $\alpha := \max^{i \in [n]} \alpha_i$。

所考虑的类别涵盖了文献中其他几个众所周知的分布类别，例如有界支撑分布和次高斯分布。此外，它还包括指数分布；在排队论以及大型分布式系统中的资源分配研究中，指数分布常用于对等待时间或计算时间建模（Gelenbe & Mitrani， 2010； Gross 等人，2011年； Hadjis 等人， 2016年； Mitliagkas 等人， 2016年；Dutta 等人， 2018年； Nguyen 等人， 2022年）。

### 4.3.3 分配算法的目标

本工作的主要目标是开发一种在线分配策略，以实现较小的期望总计算时间，其定义为

$$\mathcal{C}_K := \sum_{k=1}^{K} \mathbb{E}\left[C(a^k)\right] .$$

I如果事先已知各臂的分布，则会选择最优分配

$$a^* \in \arg\min_{a \in \mathcal{A}} \mathbb{E}\left[C(a)\right]$$

以最小化每轮期望计算时间，并在所有 $K$ 轮次中持续使用相同的分配。相应的最优总计算时间为

$$\mathcal{C}_K^* := K\mathbb{E}\left[C(a^*)\right] .$$

我们的目标是设计一种策略，使计算时间 $\mathcal{C}_K$ 保持在最优时间 $\mathcal{C}_K^*$的一个较小乘法因子范围内，此外仅增加一个可忽略项。具体而言，我们希望满足

$$\mathcal{C}_K \leq c \cdot \mathcal{C}_K^* + \mathcal{E}_K , \tag{4.3}$$

其中， $c \geq 1$ 是一个接近 1 的常数，而当 $K \to \infty$ 时，与 $\mathcal{C}_K^*$ 相比， $\mathcal{E}_K$ 是一个可忽略项。这将确保我们在极限情况下相差一个常数

乘法因子，才能达到完全了解工作节点计算时间分布的最优分配策略的性能。

寻找能够解决式（4.3）中目标的策略面临着若干技术挑战。首先，动作空间 $\mathcal{A}$ 是离散的，而且计算时间函数 $C(\cdot)$ 的非线性使得我们无法将目标转化为凸优化问题。其次， $\mathcal{A}$的规模呈组合式增长，增长量级为 $\binom{n+B-1}{B}$，因此必须利用问题固有的结构来制定高效策略。第三，由于工作节点的计算时间是随机的，任何解决方案都必须考虑不确定性。最后，在线场景要求学习者在有限的分配预算下平衡探索与利用：每轮预算为 $B$个单位，且只能获得部分反馈——我们只能观察到获得分配的工作节点的计算时间。最后这一点自然表明可以采用多臂老虎机方法。

在下一节中，我们将说明如何将这个问题简化为多臂老虎机问题，并高效地解决它。

## 4.4 自适应任务分配

这里，我们首先说明如何将式（4.3）中的问题简化为一个非线性随机多臂老虎机问题。然后，我们针对这一问题提出一种高效算法。

### 4.4.1 化简为多臂老虎机与代理损失

随机多臂老虎机问题是不确定性下序贯决策的基本框架。该问题涉及这样一种情形：智能体必须从一组臂中进行选择，每个臂都对应一个未知的奖励分布。智能体旨在通过平衡探索（收集有关奖励分布的信息）与利用（利用已知表现最佳的臂），随时间推移最大化累积奖励（或等价地，最小化累积损失）。其挑战在于如何权衡探索次优臂以改进奖励估计，与利用观测到奖励最高的臂；这一权衡还受到结果随机性质的影响。根据多臂老虎机文献中的术语，本文将每个工作节点称为一个“臂”。

然而，与标准多臂老虎机问题不同，我们面临的问题更难，因为 $\mathbb{E}[C(a^k)]$ 取决于 $a^k$ 支持集内所有臂的联合分布，而不仅仅取决于它们的期望。由于集合 $\mathcal{A}$ 具有组合性质，这种依赖关系可能使得针对 $a \in \mathcal{A}$ 依赖 $\mathbb{E}[C(a)]$ 的估计在计算上具有挑战性。

为解决这一问题，我们首先想到引入一个代理损失 $\ell\colon \mathcal{A}\times\mathbb{R}_+^n \to \mathbb{R}_+$，其定义如下：

$$\ell(a,\mu) \coloneqq \max_{i\in[n]} a_i\mu_i \ . \tag{4.4}$$

由于 $C(\cdot)$ 的凸性，引入的代理损失低估了期望计算时间。然而，在第 C.4.3 节中，我们证明，该量还可以在一个取决于臂分布的常数范围内给出期望计算时间的上界。具体而言，对于任意 $a \in \mathcal{A}$，我们证明

$$\ell(a,\mu) \le \mathbb{E}\left[C(a)\right] \le (1+4\eta\ln(B))\ell(a,\mu) \ , \tag{4.5}$$

其中 $\eta$ 定义为

$$\eta := \max_{i\in[n]} \frac{\alpha_i}{\mu_i} . \tag{4.6}$$

换言之， $\eta$ 为臂的标准差与均值之比提供了一个上界。请注意，在文献中，通常考虑指数分布、厄兰分布或伽马分布，此时比值 $\eta$ 通常[1] 有 1 作为上界。

上述界使我们能够根据分配策略在代理损失 $\ell(\cdot)$ 上的保证，推导其总计算时间的保证，误差至多相差一个 $1 + 4\eta\ln(B)$ 量级的因子。值得指出的是，在臂的分布为确定性分布（$\eta = 0$）或查询预算为单位值（$B = 1$）这一特殊情况下，两个目标 $\mathbb{E}[C(a)]$ 与 $\ell$ 完全一致。

### 4.4.2 与组合多臂老虎机设置的比较

我们的设置与组合多臂老虎机（CMAB）框架密切相关（Cesa-Bianchi 和 Lugosi，2012年），这主要是因为动作空间具有组合性质，并采用半老虎机反馈，学习者能够观察所有所选臂的结果。然而，我们的表述在两个重要方面有所不同。首先，CMAB 通常涉及从 $n$个臂中选取一个子集，因此动作空间的最大基数为 $2^n$，而我们的动作空间$\mathcal{A}$ 的基数为 $\binom{n+B-1}{B}$。这两者之比可能非常大，甚至可能随 $n$呈指数级增长。其次，尽管该领域的大多数研究假设损失函数关于臂的均值是线性的，但也有一些值得注意的例外研究了非线性奖励函数（Chen 等人，2013年；Lin 等人，2015年；Chen 等人，2016b；Wang 和 Chen，2018 年）。然而，这些方法通常依赖于奖励函数满足光滑性、利普希茨连续性或高阶可微性等假设。相比之下，我们的损失函数 $\ell(\cdot, \mu)$ 关于臂的均值并不连续。最后，出于我们设置的实际需求，我们特别强调计算效率，因此排除了大多数基于 CMAB 的方法。

### 4.4.3 自适应任务分配算法

现在，我们介绍自适应任务分配算法（ATA）。ATA不需要时域 $K$ 的先验知识，只依赖于一个上界 $\alpha$，该上界满足 $\alpha \geq \max_{i\in[n]} \|X_i - \mu_i\|_{\psi_1}$，用于臂分布的 Orlicz 范数。回想一下， $\|X_i - \mu_i\|_{\psi_1} \leq 2\|X_i\|_{\psi_1}$，因此， $\|X_i\|_{\psi_1}$ 的上界也为 $\|X_i - \mu_i\|_{\psi_1}$ 提供了一个上界。该过程的核心思想是根据臂均值的置信下界估计 $(\mu_i)_{i\in[n]}$来分配工作节点，从而平衡探索与利用。

对于每个臂 $i \in [n]$ 和轮次 $k \in [K]$，令 $K_i^k$ 表示截至轮次 $k$ 从臂 $i$ 的分布中收集的样本数量。在每轮次 $k$，我们使用迄今获得的 $K_i^k$ 个样本计算经验均值，记为 $\hat{}\mu_i^k$。基于这些经验均值，我们定义下置信界

[1]对于伽马分布$(\alpha, \lambda)$， $\sigma/\mu = 1/\sqrt{\alpha}$，因此该结论对 $\alpha \geq 1$ 成立。

**算法 4.4.1** ATA （自适应任务分配）

1: **输入**：分配预算 $B$， $\alpha > 0$2: **初始化**：经验均值 $\hat{}\mu_i^1 = 0$、使用次数 $K_i^1 = 0$和使用时间$T_i^1 = 0$，对于所有 $i \in [n]$3: **对于** $k = 1, \ldots,$ $K$ **执行**4: 使用（$s_i^k$）计算所有 $i \in [n]$ 的下置信界4.7)5: 求出分配：

$$a^k \in \arg\min_{a \in \mathcal{A}} \ell(a, s^k)$$

6: 分配 $a_i^k$ 个任务给每个工作节点 $i \in [n]$7: 更新优化参数8: 对于所有满足 $a_i^k \neq 0$ 的 $i$，更新：

$$K_i^{k+1} = K_i^k + a_i^k$$
$$T_i^{k+1} = T_i^k + \sum_{j=1}^{a_i^k} X_i^{k,j}$$
$$\hat{\mu}_i^{k+1} = \frac{T_i^{k+1}}{K_i^{k+1}}$$

9: **结束循环**

$s_i^k$ 作为

$$s_i^k = \left(\hat{\mu}_i^k - \mathrm{conf}(i,k)\right)_+ , \tag{4.7}$$

其中 $(x)_+ = \max\{x, 0\}$，并且 conf$(\cdot, \cdot)$ 定义为

$$\mathrm{conf}(i,k) = \begin{cases} 2\alpha\left(\sqrt{\frac{\ln(2k^2)}{K_i^k}} + \frac{\ln(2k^2)}{K_i^k}\right), & K_i^k \geq 1 , \\ +\infty, & K_i^k = 0 . \end{cases}$$

术语 conf$(\cdot, \cdot)$ 源自一个已知的次指数分布变量集中不等式，其 Orlicz 范数的上界为 $\alpha$ （引理 C.5.1，见附录）。

给定置信界 $s^k := (s_1^k, \ldots, s_n^k)$，学习者在第$a^k \in \mathcal{A}$ 轮次 $k$ 选择使损失 $\ell(\cdot, s^k)$最小化的动作，该损失定义见（4.4）。尽管该优化问题是非凸的，我们在第C.3 节证明，可以使用递归例程求解该问题，其计算效率为

$$\mathcal{O}\left(n \ln\left(\min\{B, n\}\right) + \min\{B, n\}^2\right) .$$

最后一步是，使用应用分配 $a^k$ 后获得的反馈来更新下置信界。 ATA 的完整伪代码见算法4.4.1。

**备注 4.4.1**。 算法第 7 行是优化方法的占位符，其中优化参数使用工作节点计算出的量进行更新（例如，在随机梯度下降的情况下为梯度）。从这个角度来看，分配算法与所选优化

算法的具体细节无关。详情请参阅第 *C.2* 节。

### 4.4.4 总计算时间的上界

我们以所需的期望总计算时间上界的形式，为 ATA 提供保证，以执行优化过程的 $K$ 次迭代。回顾一下，代理损失 $\ell(\cdot,\mu)$ 与期望计算时间之间的关系由式（4.5）给出。利用这一关系和定理4.6.1，我们可以推导出期望总计算时间的保证，记为

$$\mathcal{C}_K := \sum_{k=1}^{K} \mathbb{E}\left[C(a^k)\right] \ .$$

回顾一下，在此框架下，最优预期总计算时间为

$$\mathcal{C}_K^* := K\mathbb{E}\left[C(a^*)\right] \ ,$$

with

$$a^* \in \arg\min_{a\in\mathcal{A}} \ \mathbb{E}\left[C(a)\right] \ .$$

现在陈述关于 ATA 所达到的总期望计算时间上界的结果。

**定理 4.4.2**（证明见第C.4.3节）。假设假设条件 4.3.1成立，且 $\eta := \max_{i\in[n]} {}^{\alpha_i}/_{\mu_i}$。那么，经过 $K$轮次后，使用ATA 根据输入（$B,\alpha$）所规定的分配，其总期望计算时间满足

$$\mathcal{C}_K \leq (1 + 4\eta \ \ln(B))\,\mathcal{C}_K^* + \mathcal{O}(\ln K) \ .$$

**注释 4.4.3**。该 $\mathcal{O}(\cdot)$ 项隐藏了一个依赖于具体实例的因子。我们将在定理 *4.6.1*的遗憾上界中给出其完整细节。

定理4.4.2中的界表明，ATA 的总期望计算时间始终处于最优计算时间 $\mathcal{C}_K^*$ 的 $1 + 4\eta\ln(B)$ 倍以内，并带有一个随 $K$ 对数增长的额外余数项。由于 $\mathcal{C}_K = \Omega(K)$，与 $\mathcal{C}_K^*$ 相比，该加性项可以忽略不计。在实际场景中，当计算时间服从指数分布或伽马分布等常见分布时，因子 $\eta$ 通常为 1 阶，而对于优化算法（如SGD）中常用的批次大小，$\ln(B)$ 仍然相对较小。

读者可能会想知道，能否实现一个更具雄心的目标，即推导出乘法因子恰好为 1 的界。然而，要实现这一目标，就需要对所有 $a \in \mathcal{A}$ 的期望计算时间 $\mathbb{E}[C(a)]$ 进行更加精确的估计。由于 $\mathbb{E}[C(a)]$ 取决于支持集 $a$ 中所有工作节点的联合分布，获得如此精确的估计将以牺牲分配策略的计算效率为代价。

我们注意到，$\eta$ 出现在定理4.4.2的上界中并不令人意外，因为重尾分布会通过 $C(\cdot)$ 的凸性，增大$\ell(a,\mu)$ 与 $\mathbb{E}[C(a)]$ 之间的差距。相反，因子 $\ln(B)$ 的出现，是因为 $C(\cdot)$ 表示为至多 $B$ 个随机变量的最大值。

此外，在边界情形下，若 $\eta = 0$ （确定性情形）或 $B = 1$ （线性成本函数），我们保证期望计算时间至多比最优时间多一个加性因子。

## 4.5 经验性自适应任务分配

该ATA 流程基于下置信界方法，并依赖集中不等式。这些界对性能起着关键作用：集中界越紧，估计就越准确，也能减少对次优选项的探索。由于工作节点计算时间服从次指数分布，其集中行为由相应变量的奥尔利奇范数决定。在ATA中，唯一可用的先验知识是所有臂中最大奥尔利奇范数的上界。当各臂分布的奥尔利奇范数差异显著时，这个统一的上界可能导致置信区间过宽，进而造成探索效率低下。

为了解决这一问题，我们提出了ATA-Empirical，它能够更好地适应每个臂的分布，尤其是其奥尔利奇范数。这种适应性通过一种针对次指数变量的新型数据依赖集中不等式实现。与依赖最大奥尔利奇范数的ATA不同，ATA-Empirical会考虑所有臂各自的奥尔利奇范数，记为 $(\alpha_i)_{i\in[n]}$。这一改进体现在第4.6节给出的遗憾上界中。在实践中，这至少在某些设置下能提升性能，如第4.7节的模拟结果所示。不过，这种更强的适应性也伴随着权衡：由于 ATA-Empirical 需要量$\eta = \max_i \alpha_i/\mu_i$的上界，而不是最大奥尔利奇范数的上界。尽管如此，对于许多感兴趣的分布，不同臂之间的比值 $\alpha_i/\mu_i$ 通常处于同一数量级，而它们的奥尔利奇范数则可能差异显著。ATA-Empirical 流程与ATA 的区别仅在于所用的下置信界。这些界由引理4.6.2中的新型集中不等式推导得出，并定义为第 $i \in [n]$ 轮 $k \in [K]$ 中臂

$$\hat{s}_i^k = \hat{\mu}_i^k \left[1 - 2\,\eta \left(\sqrt{\frac{\ln(2k^2)}{K_i^k}} + \frac{\ln(2k^2)}{K_i^k}\right)\right]_+, \tag{4.8}$$

其中 $\eta = \max_{i\in[n]} \alpha_i/\mu_i$。

ATA-Empirical 的期望总计算时间 $\mathcal{C}_K$ 满足定理4.6.1中给出的相同保证，但我们改进了加性对数项的乘法因子。这些因子的精确表达式将在下一节给出，并且表明，ATA-Empirical 的保证取决于每个臂的 Orlicz 范数 $\|X_i\|_{\psi_1}$，而ATA 的保证则取决于最大的 Orlicz 范数 $\max_i \|X_i\|_{\psi_1}$。

## 4.6 理论结果

本节简要说明定理 4.4.2 的推导过程，针对 ATA 和 ATA- 经验性，通过对代理损失进行遗憾分析。我们定义期望

经过 $K$ 轮次后，代理损失 $\ell(\cdot,\ \mu)$ 的累积遗憾

$$\mathcal{R}_K := \sum_{k=1}^{K} \mathbb{E}\left[\ell(a^k, \mu)\right] - K \cdot \ell(\bar{a}, \mu)\ , \tag{4.9}$$

其中，$\bar{a} \in \arg\min_{a\in\mathcal{A}} \ell(a, \mu)$ 表示工作节点上的最优分配。如果存在多个最优动作，我们采用ATA (算法 C.2.1第 5 行中使用的优化子程序所返回的动作。

我们推导出期望累积遗憾 $\mathcal{R}_K$ 的上界。基于这些上界，我们给出完成优化过程的 $K$ 次迭代所需的期望总计算时间的保证。

### 4.6.1 ATA 的保证

对于每个工作节点 $i \in [n]$，回顾一下，$\bar{a}_i$ 表示最优动作 $\bar{a}$ 的指定分配。定义 $k_i$ 为满足以下条件的最小整数：

$$(\bar{a}_i + k_i)\mu_i > \ell(\bar{a}, \mu)\ . \tag{4.10}$$

根据上述定义可知，如果学习者在第 $a^k$ 轮$k$ 执行动作 $a_i^k \geq \bar{a}_i + k_i$，使得 $\ell(a^k, \mu) \geq \ell(\bar{a}, \mu$，那么，$k_i$ 可以解释为分配给工作节点 $i$ 的、会导致次优损失的最小额外单位数。此外，对于每个工作节点 $i \in [n]$，都有 $k_i \in \{1, 2\}$（参见附录中的引理 C.4.1 ）。

接下来的结果给出了AT 的期望遗憾上界 A.

**定理 4.6.1** （证明见C.4.1 节）。 假设4.3.1成立。那么，输入为 $(B, \alpha)$ 的ATA 的期望遗憾满足

$$\mathcal{R}_K \leq 2n \max_{i\in[n]}\{B\mu_i - \ell(\bar{a}, \mu)\} + c \cdot \sum_{i=1}^{n} \frac{\alpha^2(\bar{a}_i + k_i)(B\mu_i - \ell(\bar{a}, \mu))}{((\bar{a}_i + k_i)\mu_i - \ell(\bar{a}, \mu))^2} \cdot \ln K\ ,$$

其中 $\alpha \geq \max_{i\in[n]} \|X_i - \mu_i\|_{\psi_1}$，且 $c$ 是一个数值常数。

遗憾上界中的第一项与轮次数量$K$无关。然而，第二项随 $K$ 呈对数增长，这与文献中随机老虎机问题所观察到的行为一致。

当 $B = 1$时，我们的问题可化为标准多臂老虎机的遗憾最小化问题。注意，在这种情况下$\ell(\bar{a}, \mu) = \min^{i\in[n]} \mu_i$, $k_i = 1$ 对所有 $i \in [n]$均成立。因此，定理4.6.1 恢复了已知的最优上界

$$\mathcal{O}\left(\sum_i \frac{\ln(K)}{\Delta_i}\right)$$

标准多臂老虎机设定，其中 $\Delta_i := \mu_i - \min_j \mu_j$。

**证明概要**。在标准多臂老虎机问题和组合多臂老虎机问题中，通常通过控制学习者选择次优臂的轮次数来推导遗憾界。这些界通常为 $\ln(K)/\Delta^2$ 的量级，其中 $\Delta$ 表示次优间隙，并量化了区分最优动作与次优动作所需的探索成本。

在我们的设定中，问题更加复杂，因为学习者不仅必须选择拉取哪些臂，还必须确定如何在所选臂之间分配资源。基于这一点，我们提出以下关键论证，从而得到定理 4.6.1 中的上界。

我们将工作节点 $i$ 在轮次 $k$ 的过度分配定义为事件 $a_i^k \geq \bar{a}_i + k_i$。根据 $k_i$ 的定义（见 (4.10)），这意味着 $\ell(a^k, \mu) > \ell(\bar{a}, \mu)$。我们将满足 $\ell(a^k, \mu) > \ell(\bar{a}, \mu)$ 的轮次定义为坏轮次；当 $a_i^k \mu_i = \ell(a^k, \mu) > \ell(\bar{a}, \mu)$ 时，我们称坏轮次由臂 $i$ 触发。于是，证明的核心是对坏轮次总数建立一个上界。

为推导该上界，我们考虑验证工作节点 $i$ 发生过度分配时，其平均计算时间是否超过最优等待时间 $\ell(\bar{a}, \mu)$ 所需的样本数。具体而言，我们需要检验相应分布的均值（至少为 $(\bar{a}_i + k_i)\mu_i$）是否超过阈值 $\ell(\bar{a}, \mu)$。这等价于检验

$$\left\{ \mu_i > \frac{\ell(\bar{a}, \mu)}{\bar{a}_i + k_i} \right\}.$$

根据分析中应用的集中不等式，该检验所需的样本数量级为：

$$\alpha_i^2 \left( \mu_i - \frac{\ell(\bar{a}, \mu)}{\bar{a}_i + k_i} \right)^{-2} = \frac{\alpha_i^2 (\bar{a}_i + k_i)^2}{\left( (\bar{a}_i + k_i)\mu_i - \ell(\bar{a}, \mu) \right)^2}. \tag{4.11}$$

在工作节点 $i$ 被过度分配的轮次中，学习者会从相应分布中至少收集 $\bar{a}_i + k_i$ 个样本。因此，累积足够样本以停止过度分配工作节点 $i$ 所需的总轮次数可上界为

$$\frac{\alpha_i^2 (\bar{a}_i + k_i)}{\left( (\bar{a}_i + k_i)\mu_i - \ell(\bar{a}, \mu) \right)^2}.$$

在定理4.6.1的遗憾界中，出现的是 $\alpha^2$ 而非 $\alpha_i^2$，这是因为学习者的先验知识仅限于臂分布的最大 Orlicz 范数的上界 $\alpha \geq \max_i \|X_i - \mu_i\|_{\psi_1}$。最后，考虑到过度分配工作节点 $i$ 所造成的最坏情况额外损失为 $B\mu_i - \ell(\bar{a}, \mu)$，我们得到所述上界。

## 4.6.2 ATA-Empirical 的保证

我们通过给出期望累积遗憾的上界（ATA-Empirical ），为 4.9提供理论保证。如第 4.4 节所述，ATA-Empirical利用一种新颖的数据依赖型集中不等式所导出的下置信界，该不等式将在下文介绍。该结果的证明详见第 C.5 节。

**引理 4.6.2**。 设 $X_1, \ldots, X_n$ 是均值为 $\mu$ 的独立同分布正随机变量，且满足 $\alpha = \|X_1 - \mu\|_{\psi_1} < +\infty$。 令 $X_n$ 表示经验均值。对于 $\delta \in (0,1)$，令

$$C_{n,\delta} := 2\sqrt{\frac{\log(2/\delta)}{n}} + 2\frac{\log(2/\delta)}{n} ,$$

其中 $\eta = \alpha/\mu$。 那么，以至少 $1-\delta$ 的概率，我们有

$$\mu \geq \hat{X}_n \left(1 - \eta C_{n,\delta}\right)_+ .$$

此外，如果 $\eta C_{n,\delta} \leq 1/4$，那么，以至少 $1-\delta$ 的概率，我们有

$$\hat{X}_n \left(1 - \eta C_{n,\delta}\right)_+ \leq \mu \leq \hat{X}_n \left(1 + \frac{4}{3}\eta C_{n,\delta}\right) .$$

利用上述集中不等式，我们构造下置信界 $\hat{s}_i^k$，其定义见（4.8）。下述定理给出了ATA-Empirical遗憾的上界。

**定理 4.6.3** （证明见第C.4.2节）。假设4.3.1成立。那么，输入为（$B$, $\eta$）的 ATA-Empirical 的期望遗憾满足

$$\begin{aligned}\mathcal{R}_K \leq\ & 2n \max_{i\in[n]}\{B\mu_i - \ell(\bar{a}, \mu)\} \\ & + c\eta^2 \cdot \sum_{i=1}^{n} \frac{\mu_i^2(\bar{a}_i + k_i)(B\mu_i - \ell(\bar{a}, \mu))}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a}, \mu)\right)^2} \cdot \ln K ,\end{aligned}$$

其中 $\eta \geq \max_{i\in[n]} \alpha_i/\mu_i$， 且 $c$ 是一个数值常数。

比较ATA-Empirical 和ATA的上界，我们发现了一个关键差异。与定理4.6.1中的上界不同，后者对上界中的所有项都施加了平方最大 Orlicz 范数惩罚 $\alpha^2$ ； 而 ATA-Empirical 则受益于其自适应特性，因此产生了针对各项的因子 $\eta^2\mu_i^2$。当臂分布的比率 $\alpha_i/\mu_i$ 处于同一数量级时（例如指数分布）， 定理4.6.3 的上界表明，ATA-Empirical 能够适应量 $\alpha_i$ ， 因为在最后一种情形下，我们有 $\eta\mu_i = \alpha_i$。

## 4.7 实验

在本节中，我们通过模拟一个包含 $n$ 个工作节点的场景来验证我们的算法，在该场景中，我们使用 随机梯度下降 解决一个简单问题。每次迭代中，我们从工作节点收集 $B = 23$ 个梯度，并执行一个梯度下降步。

目标函数 $f$： $\mathbb{R}^d \to \mathbb{R}$ 是一个定义为如下形式的凸二次函数

$$f(x) = \frac{1}{2}x^\top Ax - b^\top x ,$$

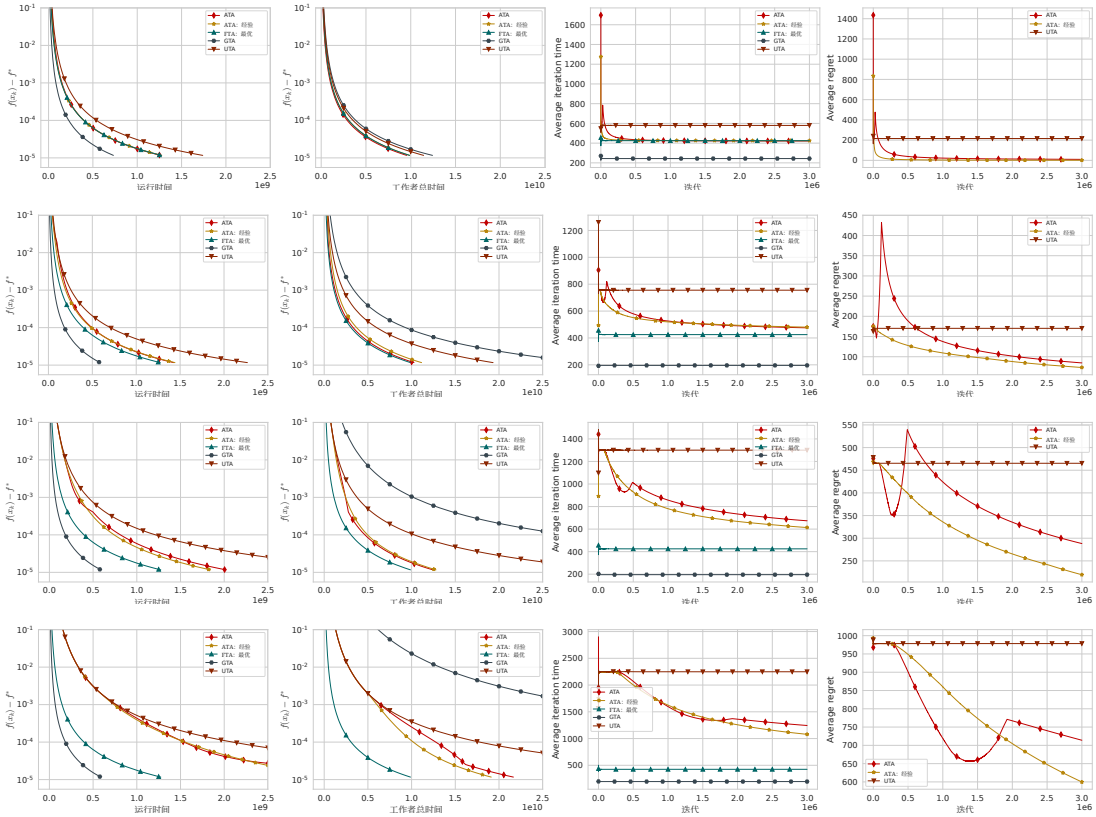



图 4.7.1：每一行的工作节点数从 17 开始，依次增加 3 倍，即 $n = 17,\ 51,\ 153,\ 459$（从上到下）。第一列展示运行时间与次优性的关系。第二列同样绘制次优性，但横轴为工作者总时间，即 $\sum_{i=1}^{n} T_i^k$，见算法4.4.1。第三列展示平均迭代时间，其计算方式为 $c_k/k$ 在所有迭代上 $k$。最后一列展示平均累积遗憾，其定义见（4.9）。

其中

$$A = \frac{1}{4}\begin{bmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{d\times d}, \quad b = \frac{1}{4}\begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{d}.$$

我们将 $f^*$ 表示为函数 $f$ 的最小值。每个 $n$ 工作节点都能够计算无偏随机梯度 $g(x)$，且满足

$$\mathbb{E}\left[\|g(x) - \nabla f(x)\|^2\right] \le 0.01^2 .$$

这是通过向 $f$ 的梯度添加高斯噪声实现的。

工作节点 $i$ 的计算时间由以下分布建模

$$\nu_i = 29\sqrt{i} + \mathrm{Exp}\left(29\sqrt{i}\right),$$

对所有 $i \in [n]$，其中 $\mathrm{Exp}(\beta)$ 表示尺度参数为 $\beta$ 的指数分布。该分布的期望值为 $\mu_i = 2 \cdot 29\sqrt{i}$。此外，

Orlicz 范数满足上界 $\alpha_i \leq 2\mu_i$。

我们考虑三种基准算法。GTA-SGD，最初由RennalaSGD 提出，作者为 Tyurin & Richt´arik (2024年)。此外，我们还纳入了OFTA （最优固定任务分配）。该算法假设已知平均计算时间这一预言机知识，并在每次迭代中使用（4.9）中的最优分配 $\bar{a}$ 。我们还纳入了UTA （均匀任务分配），它将 $B$ 个任务均匀分配给 $n$ 个工作节点。如果 $n > B$，则在UTA 中，我们会随机选择 $B$ 个工作节点，并为每个节点分配一个随机梯度计算任务。我们的算法旨在达到接近OFTA的性能，而无需了解真实均值的任何先验知识。

对于ATA，我们设定 $\alpha = \alpha_n = 4\sqrt{\cdot 29}\sqrt{n}$；对于ATA-Empirical ，我们使用 $\eta = 1$。实验结果见图4.7.1。正如预期，GTA 在运行时间方面（第一列）最快，但在工作者总时间方面（第二列）表现不佳。这是因为它使用了所有设备，其中大多数执行的无用计算都不会被采用；随着工作节点数量增加，性能因此变差。实际上，其性能可能会变得任意差。另一方面，OFTA 在工作者总时间方面表现最佳。尽管它的运行时间较长，但差距仅为一个常数因子，且不会随 $n$ 增长而扩大。这是因为增加的工作节点效率较低，无法为GTA带来显著收益。

将注意力转向我们的算法，ATA 和 ATA-Empirical 最初的表现都与UTA相似，这符合它们需要通过均匀分配进行初始探索阶段的预期。然而，在这一阶段之后，它们开始收敛至OFTA的性能。

最后两列中的图表验证了我们的理论推导。第三幅图验证了定理 4.4.2，表明 ATA 和ATA-Empirical 收敛至 OFTA ，误差至多为一个常数。最后一列展示了平均累积遗憾，正如定理 4.6.1 和 4.6.3所预测的那样，遗憾会随时间趋于零。

表 4.1：达到 $f(x) - f^* < 10^{-5}$ 所需的工作者总时间与运行时间之比。对于工作者总时间，我们将总工作时间除以其他所列算法相应的总工作时间。对于运行时间，则采用相反的计算方式，即将其他算法的运行时间除以GTA的运行时间，因为GTA 速度最快。为简化名称，我们将ATA-Empirical简称为ATA-E。

| $n$ | Tot. worker time ratio | | | Runtime ratio | | |
|---|---|---|---|---|---|---|
| | ATA | ATA-E | OFTA | ATA | ATA-E | OFTA |
| 17 | 1.3 | 1.26 | 1.26 | 1.73 | 1.75 | 1.74 |
| 51 | 2.91 | 2.69 | 3.03 | 2.43 | 2.45 | 2.17 |
| 153 | 7.22 | 7.02 | 9.1 | 3.44 | 3.14 | 2.17 |
| 459 | 12.45 | 14.1 | 27.3 | 6.36 | 5.51 | 2.17 |

在表 4.1 中，我们对结果进行了数值比较。随着 $n$的增加，工作节点总时间比率和运行时间比率都在上升。工作节点总时间比率增加，是因为GTA 的效率降低，使用了超出必要范围的资源。

对于ATA 和ATA-Empirical ，运行时间比率会随工作节点数量增加而上升，因为更多的工作节点需要进行更多探索。然而，对于OFTA，该比率保持不变，正如前文所述。

需要说明的是，在这些实验中，我们启动所有运行时，ATA 和 ATA-Empirical 都没有关于计算时间分布的先验知识。然而，在实际系统中，这些算法会被多次使用，因此可能会有基于先前运行结果得出的计算时间先验估计。借助这些信息，ATA 和ATA-Empirical 会快得多，因为它们在探索上花费的时间会更少，从而更快地接近OFTA 的性能。我们通过第C.1.5节中介绍的实验对此进行了验证。

在第 C.1.1 节中，我们采用不同的时间分布进行了类似实验，其中各臂的均值时间呈线性变化。在第 C.1.2 节中，我们考察了客户端时间分布各异的情形。此外，在第 C.1.3 节中，我们分析了遗憾性能，证实其如定理 4.6.1 和 4.6.3 所预测的那样呈对数变化。最后，在第 C.1.4 节中，我们使用 Adam（Kingma 和 Ba，2014年）在 CIFAR-100 数据集（Krizhevsky 等人，2009年）上训练了一个简单的卷积神经网络。

# 第5章

# 总结

异步方法解决了同步分布式训练的根本效率低下问题：同步屏障会使较快的工作节点在等待较慢的工作节点时处于空闲状态。然而，在本研究之前，异步方法缺乏严谨的理论基础，也未能证明其相较于同步方法具有任何理论优势。

为弥补这一空白，我们从一个简洁的理论设定入手，将异步方法旨在解决的核心问题单独抽离出来：工作节点之间的计算时间存在异质性。我们研究了数据并行模式——所有工作节点都维护完整模型的副本，但处理不同的数据——并采用一阶随机方法进行光滑非凸优化。这样的设定使我们能够排除与异步性无关的因素（例如通信延迟和内存一致性），专注于一个根本问题：异步方法能否在理论上取得更好的表现?

在这一设定下，现有分析往往依赖严格的假设，而且关键的是，尚无已知的异步算法能够达到最优时间复杂度——这一基准此前只有同步方法才能达到（Tyurin & Richt´arik，2024年）。

本学位论文证明，异步方法确实能够达到最优时间复杂度。通过精心的算法设计和严谨的分析，我们表明，陈旧性——异步优化面临的核心挑战——可以通过选择性梯度丢弃和结构化梯度缓冲等有原则的机制加以控制。我们的工作奠定了基础理论，证明在数据并行随机优化这一基本设定下，异步方法既能在理论上达到最优，也能在实践中保持高效。

## 5.1 贡献总结

在这一理论框架下，我们提出了三项主要贡献：

### 5.1.1 Ringmaster ASGD（章节 2）

我们提出了首个在同质数据设置下实现最优时间复杂度的异步随机梯度下降算法。RingmasterASGD （算法2.3.2）可视为带有选择性梯度丢弃的异步随机梯度下降 （算法2.1.1）：它会拒绝延迟超过阈值 $R$ 的梯度，从而过滤掉原本会损害收敛的陈旧更新。通过适当选择$R$，RingmasterASGD 在固定计算时间（2.1）和计算时间任意变化（第2.5节）的设置下，均能实现最优时间复杂度。此外，我们的实验表明，RingmasterASGD 在实践中可能比

最优的同步方法 RennalaSGD （Tyurin & Richt´arik，2024年）更快，这得益于更频繁的模型更新以及没有同步开销。

### 5.1.2 Ringleader ASGD（第3章）

我们将最优异步方法扩展到异质数据场景，在该场景中，不同工作节点的局部目标函数各不相同——这正是联邦学习的典型情形（Kone˘cn´y 等人，2016年，a；McMahan 等人，2017年）。

关键的算法创新结合了两种机制。首先，我们维护一个梯度表，按工作节点存储梯度，并使用所有已存储梯度的平均值来更新模型。这种方法消除了对局部目标函数之间严格相似性假设的需求；由于客户端数据多样，这类假设在联邦学习中很少成立。其次，我们通过结构化轮次控制陈旧性：每轮恰好执行 $n$ 次更新（每个工作节点一次），确保局部模型和表中的梯度都保持最新。这种设计无需显式设置延迟阈值，即可将延迟限制在一定范围内。

借助这些机制，RingleaderASGD 在固定计算时间和任意变化的计算时间两种情形下，均实现了最优时间复杂度。重要的是，服务器收到的每个梯度要么立即使用，要么存储起来供后续更新使用——不会浪费任何计算；而同步方法在同步屏障处会丢弃工作成果。由此，RingleaderASGD 成为首个在无需相似性假设的异质数据场景下被证明最优的异步方法。此外，我们的数值实验（第3.7节）表明，RingleaderASGD 在实践中能够胜过理论上最优的同步方法 MaleniaSGD （Tyurin & Richt´arik，2024年），其原因恰恰在于它避免了同步所固有的计算浪费。

### 5.1.3 自适应任务分配（章节4）

本章探讨了一个不同但同样重要的问题：计算效率。前面的最优方法通过利用所有可用的计算资源来最小化实际运行时间。在大规模系统中，如此充分的利用可能会导致不必要且浪费的计算，而无法带来相称的收益。这引出了一个自然的问题：我们能否在不显著增加总实际运行时间的情况下减少计算浪费？

为研究这一问题，我们聚焦于同质数据设置，并以Ringmaster ASGD 和 RennalaSGD为基础。我们假设工作器计算时间服从未知的概率分布。如果已知这些分布，就可以为每个工作节点分配最优数量的任务，从而最小化预期的实际运行时间。我们将这一最优固定分配作为竞争方法——即用于衡量性能的基准。

我们的目标是设计一种算法，在没有工作节点时间分布先验知识的情况下，表现几乎与最佳固定竞争基准一样出色。我们通过提出的方法 ATA 实现了这一目标。ATA 将任务分配形式化为一个多臂老虎机问题，在平衡探索与利用的同时，在线学习工作节点的计算时间分布。我们证明，ATA 的性能在期望意义下与最佳固定竞争基准相差至多一个常数乘法

因子，这是在线学习中的标准保证，称为常数遗憾。

## 5.2 未来方向

未来研究仍有若干重要方向。我们将这些方向分为两类：一类是针对本学位论文中提出的各个算法的扩展，另一类是更广泛的研究方向，适用于本文介绍的所有方法，或更一般的异步优化。

### 5.2.1 特定算法扩展

以下方向是对各个算法的自然扩展，旨在解决其当前局限性或增强其特定能力。

**利用 Ringmaster ASGD 中被舍弃的梯度**。目前，Ringmaster ASGD 会完全舍弃延迟超过阈值的梯度 $R$。一个值得探讨的问题是，能否像依赖延迟的异步随机梯度下降 方法（Koloskova 等人，2022年；Mishchenko 等人，2022a）那样，以较低权重纳入这些陈旧梯度，而不是将其完全舍弃。此类方法或许能更充分地利用可用计算资源，从而缩短实际收敛时间。此外，在数据中心环境中，如果慢速工作节点的梯度总是超过延迟阈值，它们可能始终无法做出贡献，这意味着其数据永远不会被纳入训练。若能开发机制纳入这些工作节点（例如采用依赖延迟的加权或自适应阈值），便有望在维持近乎最优的收敛保证的同时，提高数据利用率和模型质量。

**Ringleader ASGD 中的客户端采样**。RingleaderASGD 在执行更新之前需要来自所有 工作节点的梯度，这在联邦学习中可能不切实际，因为客户端经常会由于网络问题、电池限制或用户不活跃而变得不可用。一种自然的扩展方式是允许RingleaderASGD 在每一轮中仅使用部分可用客户端，而不是等待所有工作节点。不过，为了维持收敛保证，这种修改可能需要重新引入局部目标函数之间的相似性假设。另一个互补的研究方向，是在这些相似性假设下建立异步方法的下界。

**弥合 Ringleader ASGD 的最优性差距**。我们的分析表明，当新的光滑常数（假设 3.3.2）与标准光滑参数相差不超过一个常数因子时，RingleaderASGD 达到了现有下界。然而，仍存在一个差距：尚不清楚这个常数因子是否是根本性的，还是说 RingleaderASGD 的复杂度还能进一步降低。未来工作可以通过以下任一方式解决这一问题：(i) 放宽确立最优性所依据的假设，或 (ii) 为异构环境中的异步方法推导更紧的下界。我们认为后者更有希望——针对异步算法提出新的、更精确的下界，或许能够

最终确定 RingleaderASGD 是否真正达到最优，还是仍有进一步改进的可能。

**面向异质数据的自适应任务分配**。该ATA算法是为齐次数据设置开发的，可提高Ringmaster ASGD 和RennalaSGD (Tyurin 与 Richt´arik，2024年）的计算效率。一种自然的扩展是将这些思路应用于异质数据场景，从而提升RingleaderASGD 和MaleniaSGD (Tyurin 与 Richt´arik，2024年）的计算效率。此类扩展需要考虑不同的梯度收集停止标准；在这种情况下，这些标准取决于梯度数量的调和平均数。

**应对对抗性延迟的自适应任务分配**。现有的ATA框架假设工作器计算时间服从未知但固定的分布。然而，在联邦学习中，工作器的行为可能极难预测：设备可能变得不可用、遭遇极端延迟，或呈现不符合任何稳定分布的非平稳模式。开发能够抵御此类对抗性或非随机延迟的分配策略，将显著提升实际适用性。这一方向与对抗性在线学习和老虎机算法文献相关（Auer等人，2002年）。

### 5.2.2 更广泛的研究方向

除了扩展单个算法之外，还有若干基本问题可以改进本学位论文中的所有方法，或更广泛地推进异步优化。

**超越随机梯度下降**。 我们的分析仅关注随机梯度下降这一最基础的一阶优化方法。然而，现代深度学习经常依赖更先进的优化器，例如 Adam(Kingma 和 Ba，2014年）、AdamW (Loshchilov & Hutter，2019年），以及 Shampoo (Gupta等人，2018年）等预条件梯度方法，以及较新的算法，例如 Muon(Jordan等人，2024年）。在保留理论保证的同时，将我们的异步框架扩展到这些方法，将显著拓宽本研究的实际适用范围。

**扩展至其他并行形式**。 本学位论文仅关注数据并行。然而，现代大规模训练系统日益将数据并行与模型并行和流水线并行结合起来（Narayanan 等人，2021年；Shoeybi等人，2019年）。将异步优化方法扩展到这类混合设置，是顺理成章的下一步。尤其值得关注的是，为异步流水线并行开展时间复杂度分析。在这种并行方式中，模型的不同阶段由不同工作节点执行，并受到阶段间延迟的影响。近期已有多项研究探讨了这一背景下的异步流水线训练（Yang 等人，2019年；Ajanthan等人，2025年；Kong等人，2025年）。

**通信成本和网络延迟**。在本文中，我们假设通信是即时的——这一简化使我们能够单独研究异步性的基本挑战。

然而，在联邦系统和地理分布式系统中，通信往往占据计算时间的主要部分。为缓解这一问题，现有方法采用通信压缩（Alistarh 等人，2017年）或本地训练（McMahan 等人，2016年；Mishchenko 等人，2022b；Malinovsky 等人，2022年；Maranjyan 等人，2025c）来降低传输更新的频率或数量。将异步性与这些技术——本地训练或通信压缩——相结合，并分析由此产生的时间复杂度，仍是一个重要的开放问题。两个方向都已取得初步进展，近期已有关于异步压缩（Tyurin 等人，2024a）和异步本地训练（Tyurin 和 Sivtsov，2025年；Fradin 等人，2025年）的研究结果。

**无锁共享内存算法**。在整个研究过程中，我们假设读写具有原子性，且通信瞬时完成。另一种模型见于 HOGWILD! （Recht 等人，2011年）等研究，它考虑的是一种共享内存系统：多个工作节点可以在没有锁的情况下同时读写。在这种情形下，读取参数向量的工作节点可能会观察到不一致的值——当前服务器状态与其他工作节点部分更新的混合。近期研究（Mania 等人，2017年；Leblond 等人，2018年；Nguyen 等人，2018年）改进了对 HOGWILD! 风格算法的分析。值得探究的是，本学位论文中提出的技术能否适用于这种无锁环境，以及即使存在不一致读，是否仍能达到最优时间复杂度。

**大规模实证验证**。本学位论文中的实验主要是小规模演示，旨在验证理论预测。未来工作的一个重要方向是在现实环境中进行大规模实证验证：在实际的数据中心或联邦学习环境中训练大语言模型或其他现代架构。此类实验能够揭示理论模型未能涵盖的实际瓶颈，并有助于缩小理论与部署之间的差距。

## 5.3 结束语

本学位论文证明，异步方法能够达到最优时间复杂度，为其在分布式优化中的应用提供了首个严谨的理论基础。虽然我们的结果聚焦于将数据并行随机梯度下降应用于光滑非凸目标函数这一基本设定，但它们标志着构建全面的异步优化理论迈出了至关重要的第一步。

除了具体的最优性结果外，本研究还提出了具有广泛适用性的技术：用于控制陈旧性的原则性机制（例如选择性梯度丢弃和结构化梯度缓冲）、用于分析异步性下收敛与时间复杂度的严谨框架，以及平衡实际运行时间与计算成本的策略。这些理念共同为将异步方法拓展至更复杂、更贴近现实的环境奠定了基础。

这项研究的意义不止于理论层面。异步算法在实践中的应用历来有限，原因在于理论保证薄弱、软件支持不足。本学位论文证明了异步算法既能达到理论最优，又能在实践中高效运行，为开发稳健实现并将这些方法集成到大规模训练系统中提供了有力动机。随着人工智能系统规模不断扩大，训练基础设施也日益异质化，涵盖数据中心、边缘设备和联邦网络，高效异步优化的重要性只会与日俱增。

这篇学位论文的核心观点强调的是可能性，而非局限性。同步算法具有可预测性，但会因强制闲置而浪费资源。异步算法消除了这种浪费，却带来了管理陈旧信息的挑战。本论文表明，这种权衡并非不可避免：通过精心的算法设计，可以在保持理论最优性和实际效率的同时控制陈旧性。大规模机器学习的未来，不在于强迫所有工作节点步调一致地行动，而在于让它们能够异步地作出贡献，并通过有原则的机制加以协调，从而确保收敛和效率。正如本文所示，这种协调不仅可行，而且是最优的。

# 参考文献

Josh Achiam、Steven Adler、Sandhini Agarwal、Lama Ahmad、Ilge Akkaya、Florencia Leoni Aleman、Diogo Almeida、Janko Altenschmidt、Sam Altman、Shyamal Anadkat 等。GPT-4 技术报告。*arXiv* 预印本 *arXiv:2303.08774*，2023年。

Alekh Agarwal 和 John C Duchi。分布式延迟随机优化。神经信息处理系统进展，24，2011年。

国际能源署。能源与人工智能，2025年4月。网址：https://www.iea.org/reports/energy-and-ai。

Thalaiyasingam Ajanthan、Sameera Ramasinghe、Yan Zuo、Gil Avraham 和 Alexander Long。异步流水线并行优化的 Nesterov 方法。载于国际机器学习会议，2025年。

´Abdelkrim Alahyane、C´elineComte、Matthieu Jonckheere 和 Eric Moulines。优化异步联邦学习：模型参数陈旧性与更新频率之间的微妙权衡。*arXiv* 预印本 *arXiv:2502.08206*，2025年。

Dan Alistarh、Demjan Grubic、Jerry Li、Ryota Tomioka 和 Milan Vojnovic。QSGD：通过梯度量化与编码实现高通信效率的随机梯度下降。载于神经信息处理系统进展（*NIPS*），第1709–1720页，2017年。

Ganesh Ananthanarayanan、Ali Ghodsi、Scott Shenker 和 Ion Stoica。《有效的掉队者缓解：克隆的攻击》。载于第十届 *USENIX* 网络系统设计与实现大会论文集，nsdi'13，第 185–198 页，美国，2013年。USENIX 协会。

Yossi Arjevani、Ohad Shamir 和 Nathan Srebro。《延迟更新下随机梯度下降的紧致收敛性分析》。载于算法学习理论，第 111–132 页。PMLR，2020年。

Yossi Arjevani、Yair Carmon、John C Duchi、Dylan J Foster、Nathan Srebro 和 Blake Woodworth。非凸随机优化的下界。*Mathematical Programming*，第 1–50 页，2022年。

Mahmoud Assran、Arda Aytekin、Hamid Reza Feyzmahdavian、Mikael Johans- son 和 Michael G Rabbat。异步并行分布式优化的进展。*IEEE*会刊，108(11):2013–2031，2020年。

P Auer。《多臂老虎机问题的有限时间分析》，2002年。

Peter Auer，Nicol`o Cesa-Bianchi、Yoav Freund 和 Robert E. Schapire。《非随机多臂老虎机问题》。*SIAM*计算杂志，32（1）：48–77，2002年。doi：10.1137/S0097539701398375。URL https://doi.org/10.1137/S0097539701398375。

Arda Aytekin、Hamid Reza Feyzmahdavian 和 MikaelJohansson。参数服务器异步优化算法的分析与实现。*arXiv*预印本 *arXiv:1610.05507*，2016年。

Doron Blatt、Alfred O Hero 和 Hillel Gauchman。采用恒定步长的收敛增量梯度方法。*SIAM* 优化期刊，18（1）：29–51，2007年。

MariuszBojarski、DavideDel Testa、Daniel Dworakowski、Bernhard Firner、Beat Flepp、Prasoon Goyal、Lawrence D Jackel、Mathew Monfort、Urs Muller、Ji- akai Zhang 等。自动驾驶汽车的端到端学习。*arXiv*预印本 *arXiv:1604.07316*，2016年。

L´eon Bottou、Frank E. Curtis 和 Jorge Nocedal。大规模机器学习的优化方法。*SIAM*评论，60(2):223–311，2018年。doi: 10. 1137/16M1080173。URL https://doi.org/10.1137/16M1080173。

St´ephane Boucheron、Olivier Bousquet、G´abor Lugosi 和 Pascal Massart。独立随机变量函数的矩不等式。2005年。

Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Ka-plan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、AmandaAskell、Sandhini Agarwal、Ariel Herbert-Voss、GretchenKrueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel Ziegler、Jeffrey Wu、Clemens Winter、Chris Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray、Ben-jamin Chess、Jack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。语言模型是少样本学习器。在 H. Larochelle、M. Ranzato、R. Hadsell、M. F. Balcan 和 H. Lin（编）所著的《神经信息处理系统进展》中，第 33 卷，第 1877–1901 页。Curran Associates, Inc.，2020年。URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf。

Nicolo Cesa-Bianchi 和 G´abor Lugosi。组合多臂老虎机。《计算机与系统科学杂志》，78(5)：1404–1422，2012年。

Jianmin Chen、Xinghao Pan、Rajat Monga、Samy Bengio 和Rafal Jozefowicz。重新审视分布式同步随机梯度下降。*arXiv* 预印本 *arXiv:1604.00981*，2016年。

Wei Chen、Yajun Wang 和 Yang Yuan。组合多臂老虎机：一般框架及应用。载于国际机器学习会议，第151–159页。PMLR，2013年。

Wei Chen、Wei Hu、Fu Li、Jian Li、Yu Liu 和 Pinyan Lu。具有一般奖励函数的组合多臂老虎机。神经信息处理系统进展，29，2016b。

Pranav Singh Chib 和 Pravendra Singh。使用深度学习实现端到端自动驾驶的最新进展：综述。*IEEE*智能车辆汇刊，9(1)：103–118，2023年。

Aakanksha Chowdhery、Sharan Narang、Jacob Devlin、Maarten Bosma、Gaurav Mishra、Adam Roberts、Paul Barham、Hyung Won Chung、Charles Sutton、Sebastian Gehrmann 等。PaLM：利用路径扩展语言建模。机器学习研究期刊，24(240)：1–113，2023年。

Alon Cohen、Amit Daniely、Yoel Drori、Tomer Koren 和 Mariano Schain。对任意延迟具有鲁棒性的异步随机优化。《神经信息处理系统进展》，34：9024–9035，2021年。

Richard Combes、M. Sadegh Talebi、Alexandre Proutiere 和 Marc Lelarge。组合老虎机再探。《神经信息处理系统进展》，28，2015年。

Laurent Condat、Artavazd Maranjyan 和 Peter Richt´arik。LoCoDL：采用本地训练与压缩的通信高效分布式学习。载于国际学习表征会议论文集（*ICLR*），2025年。

Andrew Cotter、Ohad Shamir、Nati Srebro 和 Karthik Sridharan。通过加速梯度方法改进小批次算法。《神经信息处理系统进展》，24，2011年。

Ben Cottier、Robi Rahman、Loredana Fattorini、Nestor Maslej、Tamay Besiroglu 和DavidOwen。前沿人工智能模型训练成本的不断上升。*arXiv* 预印本 *arXiv:2405.21015*，2024年。

Paul Covington、Jay Adams 和 Emre Sargin。用于 YouTube 推荐的深度神经网络。载于第十届 *ACM* 推荐系统会议论文集，第 191–198 页，2016年。Alex De Vries。人工智能能源足迹的不断增长。*Joule*，7 (10):2191–2194，2023年。Jeffrey Dean 和 Luiz Andr´e Barroso。规模化尾部效应。*ACM*通讯，56(2):74–80，2013年。

Jeffrey Dean、Greg Corrado、Rajat Monga、Kai Chen、Matthieu Devin、Mark Mao、Marc'aurelio Ranzato、Andrew Senior、Paul Tucker、Ke Yang、Quoc Le 和 Andrew Ng。《大规模分布式深度网络》。载于 F. Pereira、C.J. Burges、L. Bottou 和 K.Q. Weinberger（编），神经信息处理系统进展，第 25 卷。Curran Associates, Inc.，2012年。网址 https://proceedings.neurips.cc/paper_files/paper/2012/file/6aca97005c68f1206823815f66102863-Paper.pdf。

Arthur Douillard、Qixuan Feng、Andrei A. Rusu、Rachita Chhaparia、Yani Donchev、Adhiguna Kuncoro、Marc'Aurelio Ranzato、Arthur Szlam 和 Jiajun Shen。《DiLoCo：语言模型的分布式低通信训练》，2024年。网址 https://arxiv.org/abs/2311.08105。

Sanghamitra Dutta、Gauri Joshi、Soumyadip Ghosh、Parijat Dube 和 Priya Nagpurkar。缓慢且陈旧的梯度也能赢得竞赛：分布式随机梯度下降中的误差与运行时间权衡。发表于国际人工智能与统计学会议，第 803–812 页。PMLR，2018年。

Cooper Elsworth、Keguo Huang、David Patterson、Ian Schneider、Robert Sedivy、Savannah Goodman、Ben Townsend、Parthasarathy Ranganathan、Jeff Dean、Amin Vahdat 等。衡量在 Google 规模下提供人工智能服务的环境影响。*arXiv* 预印本 *arXiv:2508.15734*，2025年。

Andre Esteva、Brett Kuprel、Roberto A Novoa、Justin Ko、Susan M Swetter、Helen M Blau 和 Sebastian Thrun。利用深度神经网络实现皮肤癌的皮肤科医生级分类。《自然》，542(7639):115–118，2017年。

Hamid Reza Feyzmahdavian 和 Mikael Johansson。《优化中的异步迭代：新的序列结果与更精确的算法保证》。机器学习研究期刊，24(158):1–75，2023年。

Hamid Reza Feyzmahdavian、Arda Aytekin 和 Mikael Johansson。用于正则化随机优化的异步小批量算法。*IEEE*自动控制汇刊，61(12):3740–3754，2016年。

Yann Fraboni、Richard Vidal、Laetitia Kameni 和 Marco Lorenzi。异步异构客户端更新的联邦优化一般理论。机器学习研究期刊，24(110):1–43，2023年。

Adrien Fradin、Peter Richt´arik 和 Alexander Tyurin。通过时间复杂度的视角审视本地随机梯度下降与联邦平均。*arXiv* 预印本 *arXiv:2509.23207*，2025年。

Yi Gai、Bhaskar Krishnamachari 和 Rahul Jain。具有未知变量的组合网络优化：具有线性奖励和个体观测的多臂老虎机。*IEEE/ACM*网络汇刊，20(5)：1466–1478，2012年。

Erol Gelenbe 和 Isi Mitrani。计算机系统的分析与综合，第4卷。WorldScientific，2010年。

Saeed Ghadimi 和 Guanghui Lan。用于非凸随机规划的随机一阶与零阶方法。*SIAM* 优化期刊，23(4)：2341–2368，2013年。

Margalit R Glasgow 和 Mary Wootters。具有随机延迟的异步分布式优化。载于国际人工智能与统计学会议，第9247–9279页。PMLR，2022年。

Robert Mansel Gower、Nicolas Loizou、Xun Qian、Alibek Sailanbayev、Egor Shulgin 和 Peter Richt´arik。《随机梯度下降：一般分析与改进的收敛速率》。载于国际机器学习会议，第 5200–5209 页。PMLR，2019年。

Priya Goyal、PiotrDoll´ar、Ross Girshick、Pieter Noordhuis、Lukasz Wesolowski、Aapo Kyrola、Andrew Tulloch、Yangqing Jia 和何恺明。《精确的大批次随机梯度下降：1 小时内训练 ImageNet》。*arXiv* 预印本 *arXiv:1706.02677*，2017年。

Aaron Grattafiori、Abhimanyu Dubey、Abhinav Jauhri、Abhinav Pandey、Ab- hishek Kadian、Ahmad Al-Dahle、Aiesha Letman、Akhil Mathur、AlanSchel- ten、Alex Vaughan 等。《Llama 3 模型家族》。*arXiv* 预印本 *arXiv:2407.21783*，2024年。

Martin Grigoryan 和 Artavazd Maranjyan。关于一般 Haar 系统中傅里叶级数的发散。亚美尼亚数学杂志，13(6):1–10，2021年a。

Martin Grigoryan、Anna Kamont 和 Artavazd Maranjyan。局部化算子序列发散集的 Menshov 型定理。当代数学分析（亚美尼亚科学院），58(2):81–92，2023年。

Tigran M Grigoryan 和 Artavazd Maranjyan。关于 $L^1$ 中 Faber-Schauder 级数的无条件收敛。《*YSU A*辑：物理与数学科学学报》，55(1 (254)):12–19，2021年b。

Donald Gross、John F Shortle、James M Thompson 和 Carl M Harris。排队论基础，第 627 卷。John Wiley & Sons，2011年。

Vineet Gupta、Tomer Koren 和 Yoram Singer。Shampoo：预条件随机张量优化。载于国际机器学习大会，第 1842–1850 页。PMLR，2018年。

Mert Gurbuzbalaban、Asuman Ozdaglar 和 Pablo A Parrilo。关于增量聚合梯度算法的收敛速率。*SIAM* 优化期刊，27(2):1035–1048，2017年。

Stefan Hadjis、Ce Zhang、Ioannis Mitliagkas、Dan Iter 和 Christopher R´e。Omnivore：一种用于 CPU 和图形处理器上多设备深度学习的优化器。*arXiv* 预印本 *arXiv:1606.04487*，2016年。

J. Edward Hu、Yelong Shen、Phillip Wallis、Zeyuan Allen-Zhu、Yuanzhi Li、Shean Wang 和 Weizhu Chen。低秩适配（LoRA）：大型语言模型的低秩适配。2022年。

Rustem Islamov、Mher Safaryan 和 Dan Alistarh。AsGrad：对异步随机梯度下降算法的尖锐统一分析。发表于国际人工智能与统计学会议，第649–657页。PMLR，2024年。

Sashank J Reddi、AhmedHefny、Suvrit Sra、Barnabas Poczos 和 Alexander J Smola。关于随机梯度下降及其异步变体中的方差缩减。神经信息处理系统进展，第28卷，2015年。

Keller Jordan、Yuchen Jin、Vlado Boza、Jiacheng You、Franz Cesista、Laker New- house 和 Jeremy Bernstein。Muon：一种用于神经网络隐藏层的优化器，2024年。URL https://kellerjordan.github.io/posts/muon/。

Norman P. Jouppi、Cliff Young、Nishant Patil、David Patterson、Gaurav Agrawal、Raminder Bajwa、Sarah Bates、Suresh Bhatia、Nan Boden、Al Borchers 和 Robert Boyle。《张量处理单元的数据中心内性能分析》。

载于第*44*届年度国际计算机体系结构研讨会论文集，第1–12页，2017年6月。

Peter Kairouz、H Brendan McMahan、Brendan Avent、Aur´elien Bellet、Mehdi Bennis、Arjun Nitin Bhagoji、Kallista Bonawitz、Zachary Charles、Graham Cormode、Rachel Cummings 等。联邦学习的进展与开放问题。*Foundations and Trends*® 机器学习，14(1–2):1–210，2021年。

Jared Kaplan、Sam McCandlish、Tom Henighan、Tom B Brown、Benjamin Chess、RewonChild、Scott Gray、Alec Radford、JeffreyWu 和 DarioAmodei。神经语言模型的缩放定律。*arXiv* 预印本 *arXiv:2001.08361*，2020年。

Diederik P Kingma 和 Jimmy Ba。Adam：一种随机优化方法。*arXiv* 预印本 *arXiv:1412.6980*，2014年。

Anastasiia Koloskova、Sebastian U Stich 和 MartinJaggi。分布式学习和联邦学习中异步随机梯度下降的更强收敛保证。神经信息处理系统进展，35:17202–17215，2022年。

Jakub Koneˇcn´y、H Brendan McMahan、Daniel Ramage 和 Peter Richt´arik。《联邦优化：面向端侧智能的分布式机器学习》。*arXiv* 预印本 *arXiv:1610.02527*，2016年。

Jakub Koneˇcn´y、H. Brendan McMahan、Felix X. Yu、Peter Richt´arik、Ananda Theertha Suresh 和 Dave Bacon。《联邦学习：提升通信效率的策略》。发表于*NIPS* 私有多方机器学习研讨会，2016b。

Jakub Koneˇcn´y、H. Brendan McMahan、Felix X. Yu、Peter Richt´arik、Ananda Theertha Suresh 和 Dave Bacon。《联邦学习：提升通信效率的策略》。发表于*NIPS* 私有多方机器学习研讨会，2016年。

Boao Kong、Xu Huang、Yuqi Xu、Yixuan Liang、Bin Wang 和 Kun Yuan。《Clapping：通过通信压缩，在流水线并行分布式优化中移除逐样本存储》。*arXiv* 预印本 *arXiv:2509.19029*，2025年。

Alex Krizhevsky、Geoffrey Hinton 等人。从微型图像中学习多层特征。技术报告，多伦多大学，多伦多，2009年。

Alex Krizhevsky、Ilya Sutskever 和 Geoffrey E Hinton。使用深度卷积神经网络进行 ImageNet 分类。神经信息处理系统进展，25，2012年。

Branislav Kveton、Zheng Wen、Azin Ashkan 和 Csaba Szepesvari。随机组合半老虎机的紧遗憾界。载于人工智能与统计，第 535–543 页。PMLR，2015年。T. Lattimore 和 C. Szepesv´ari。老虎机算法。剑桥大学出版社，2020年。

Remi Leblond、Fabian Pedregosa 和 Simon Lacoste-Julien。针对随机增量方法的改进型异步并行优化分析。机器学习研究期刊，19(81):1–68，2018年。网址 http://jmlr.org/papers/v19/17-650.html。

Yann LeCun、L´eon Bottou、Yoshua Bengio 和 Patrick Haffner。应用于文档识别的基于梯度的学习。*IEEE*会刊，86(11): 2278–2324，1998年。

Yann LeCun、Corinna Cortes 和 CJ Burges。MNIST手写数字数据库。*ATT*实验室 *[*在线*]*。网址：*http://yann.lecun.com/exdb/mnist*，2，2010年。

Mu Li、David G Andersen、Alexander Smola 和 Kai Yu。使用参数服务器实现通信高效的分布式机器学习。神经信息处理系统进展，27，2014年。

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania 等。PyTorch分布式：加速数据并行训练的经验。*arXiv*预印本 *arXiv:2006.15704*，2020a。

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, AmeetTalwalkar 和 Virginia Smith。异构网络中的联邦优化。机器学习与系统会议论文集，2：429–450，2020b。

Xiangru Lian、Yijun Huang、Yuncheng Li 和 Ji Liu。用于非凸优化的异步并行随机梯度。神经信息处理系统进展，28，2015年。

Xiangru Lian、Wei Zhang、Ce Zhang 和 Ji Liu。异步去中心化并行随机梯度下降。发表于国际机器学习会议，第3043–3052页。PMLR，2018年。

Tian Lin、Jian Li 和 Wei Chen。具有半 bandit 反馈的随机在线贪心学习。神经信息处理系统进展，28，2015年。

Ilya Loshchilov 和 Frank Hutter。解耦权重衰减正则化。2019年。

Yucheng Lu 和 Christopher De Sa。去中心化训练中的最优复杂度。载于《国际机器学习会议》，第 7111–7123 页。PMLR，2021年。

Grigory Malinovsky、Kai Yi 和 Peter Richt´arik。方差缩减 ProxSkip：面向联邦学习的算法、理论与应用。《神经信息处理系统进展》，35:15176–15189，2022年。

Horia Mania、Xinghao Pan、Dimitris Papailiopoulos、Benjamin Recht、Kannan Ramchandran 和 Michael I Jordan。异步随机优化的扰动迭代分析。《SIAM 优化期刊》，27(4):2202–2229，2017年。

Artavazd Maranjyan 和 Peter Richt´arik。Ringleader ASGD：首个在数据异质性下具有最优时间复杂度的异步随机梯度下降方法。*arXiv* 预印本 *arXiv:2509.22860*，2025年。

Artavazd Maranjyan、Abdurakhmon Sadiev 和 Peter Richt´arik。差分隐私随机块坐标下降。*arXiv* 预印本 *arXiv:2412.17054*，2024年。

Artavazd Maranjyan、Omar Shaikh Omar 和 Peter Richt´arik。MindflayerSGD：存在异质且随机的工作节点计算时间时的高效并行随机梯度下降。发表于第 *41* 届人工智能不确定性会议，2025a。

Artavazd Maranjyan、El Mehdi Saad、Peter Richt´arik 和 Francesco Orabona。UTA：用于分布式机器学习中高效资源管理的自适应任务分配。发表于国际机器学习会议，2025b。

Artavazd Maranjyan、Mher Safaryan 和 Peter Richt´arik。GradSkip：通信加速的局部梯度方法，具有更优的计算复杂度。机器学习研究汇刊，2025c。ISSN 2835- 8856。URL https://openreview.net/forum?id=6R3fRqFfhn。

Artavazd Maranjyan、Alexander Tyurin 和 Peter Richt´arik。《Ringmaster ASGD：首个具有最优时间复杂度的异步随机梯度下降》。发表于国际机器学习会议，2025d。

Andreas Maurer 和 Massimiliano Pontil。次高斯和次指数条件下的集中不等式。神经信息处理系统进展，34:7588–7597，2021年。

Jos´e Maur´ıcio、Inˆes Domingues 和 Jorge Bernardino。用于图像分类的视觉变换器与卷积神经网络比较：文献综述。*Applied Sciences*，13(9):5521，2023年。

Brendan McMahan、Eider Moore、Daniel Ramage、Seth Hampson 和 Blaise Aguera y Arcas。《利用去中心化数据进行深度网络的通信高效学习》。载于人工智能与统计，第 1273–1282 页。PMLR，2017年。

H BrendanMcMahan、Eider Moore、Daniel Ramage 和 Blaise Ag¨uera y Arcas。《使用模型平均进行深度网络的联邦学习》。*arXiv* 预印本 *arXiv:1602.05629*，2:2，2016年。

Konstantin Mishchenko、Francis Bach、Mathieu Even 和 Blake E Woodworth。《任意延迟下，异步随机梯度下降优于小批量随机梯度下降》。神经信息处理系统进展，35:420–433，2022a。

Konstantin Mishchenko、Grigory Malinovsky、Sebastian Stich 和 Peter Richtarik。ProxSkip：是的！局部梯度步可证明能够加速通信！终于做到了！载于 Kamalika Chaudhuri、Stefanie Jegelka、Le Song、Csaba Szepesvari、Gang Niu 和 Sivan Sabato（编），第*39*届国际机器学习会议论文集，第162卷，收录于机器学习研究论文集，第15750–15769页。PMLR，2022年7月17日至23日。网址 https://proceedings.mlr.press/v162/mishchenko22b.html。

Ioannis Mitliagkas、Ce Zhang、Stefan Hadjis 和 Christopher R´e。异步性带来动量及其在深度学习中的应用。载于*2016*年第*54*届 *Allerton*通信、控制与计算会议（*Allerton*），第 997–1004 页。IEEE，2016年。

Deepak Narayanan、Mohammad Shoeybi、Jared Casper、Patrick LeGresley、Mostofa Patwary、Vijay Korthikanti、Dmitri Vainbrand、Prethvi Kashinkunti、Julie Bernauer、Bryan Catanzaro 等。使用 Megatron-LM 在 GPU集群上高效进行大规模语言模型训练。载于高性能计算、网络、存储与分析国际会议论文集，第 1–15 页，2021年。

Yurii Nesterov。凸优化讲义，第 137 卷。Springer，2018年。

John Nguyen、Kshitiz Malik、Hongyuan Zhan、Ashkan Yousefpour、Mike Rab-bat、Mani Malek 和 Dzmitry Huba。采用缓冲异步聚合的联邦学习。载于国际人工智能与统计学会议，第 3581–3607 页。PMLR，2022年。

Lam Nguyen、Phuong Ha Nguyen、Marten Dijk、Peter Richt´arik、Katya Schein-berg 和 Martin Tak´ac。无需有界梯度假设的随机梯度下降和 HOGWILD! 收敛。载于国际机器学习会议，第 3750–3758 页。PMLR，2018年。

James O'Donnell 和 Casey Crownhart。我们计算了人工智能的能耗足迹。这是你尚未听说过的故事。《*MIT Technology Review*》，2025年5月。网址 https://www.technologyreview.com/2025/05/20/1116852/ai-energy-footprint/。访问日期：2025年10月18日。

Xinghao Pan、Maximilian Lam、Stephen Tu、Dimitris Papailiopoulos、Ce Zhang、Michael I Jordan、Kannan Ramchandran 和 Christopher R´e。Cyclades：无冲突异步机器学习。载于 D. Lee、M. Sugiyama、U. Luxburg、I. Guyon 和 R. Garnett（编），《神经信息处理系统进展》，第 29 卷。Curran Associates, Inc.，2016年。网址 https://proceedings.neurips.cc/paper_files/paper/2016/file/28e209b61a52482a0ae1cb9f5959c792-Paper.pdf。

Samyam Rajbhandari、Jeff Rasley、Olatunji Ruwase 和 Yuxiong He。ZeRO：面向万亿参数模型训练的内存优化。载于*SC20*：高性能计算、网络、存储与分析国际会议，第 1–16 页。IEEE，2020年。

Reenu Rajpoot、Mahesh Gour、Sweta Jain 和 VijayBhaskarSemwal。用于从 CT扫描和 X射线图像诊断 COVID-19 的集成卷积神经网络与可解释人工智能。《科学报告》，14(1):24985，2024年。

Benjamin Recht、Christopher Re、Stephen Wright 和 Feng Niu。HOGWILD!：一种用于并行化随机梯度下降的无锁方法。神经信息处理系统进展，24，2011年。

HerbertRobbins 和 SuttonMonro。随机近似方法。《数学统计年鉴》，22(3):400–407，1951年。

Nicolas Roux、Mark Schmidt 和 Francis Bach。针对有限训练集的具有指数收敛率的随机梯度方法。神经信息处理系统进展，25，2012年。

Kevin Scaman、Francis Bach、S´ebastien Bubeck、Yin Tat Lee 和 Laurent Mas- souli´e。《网络中平滑且强凸分布式优化的最优算法》。载于国际机器学习会议，第 3027–3036 页。PMLR，2017年。

Mark Schmidt、Nicolas Le Roux 和 Francis Bach。《使用随机平均梯度最小化有限和》。《Mathematical Programming》，162：83–112，2017年。

Mohammad Shoeybi、Mostofa Patwary、Raul Puri、Patrick LeGresley、Jared Casper 和 Bryan Catanzaro。Megatron-LM：利用模型并行训练数十亿参数的语言模型。*arXiv*预印本 *arXiv:1909.08053*，2019年。

Sebastian U Stich 和 Sai Praneeth Karimireddy。误差反馈框架：采用延迟梯度的随机梯度下降。机器学习研究期刊，21(237)：1–36，2020年。

Emma Strubell、Ananya Ganesh 和 Andrew McCallum。现代深度学习研究中的能源与政策考量。载于*AAAI*人工智能大会论文集，第34卷，第13693–13696页，2020年。

Alysa Ziying Tan、Han Yu、Lizhen Cui 和 Qiang Yang。迈向个性化联邦学习。*IEEE*神经网络与学习系统汇刊，34(12)：9587–9603，2022年。

Gemini团队、Rohan Anil、Sebastian Borgeaud、Jean-Baptiste Alayrac、Jiahui Yu、Radu Soricut、Johan Schalkwyk、Andrew M Dai、Anja Hauth、Katie Mil- lican 等人。Gemini：一系列能力强大的多模态模型。*arXiv*预印本 *arXiv:2312.11805*，2023年。

John Tsitsiklis、Dimitri Bertsekas 和 MichaelAthans。分布式异步确定性与随机梯度优化算法。*IEEE*自动控制汇刊，31(9):803–812，1986年。

Alexander Tyurin。任意计算动态下并行随机优化的紧确时间复杂度。*arXiv*预印本 *arXiv:2408.04929*，2024年。

Alexander Tyurin 和 Peter Richt´arik。固定计算模型下并行随机优化方法的最优时间复杂度。神经信息处理系统进展，36，2024年。

Alexander Tyurin 和 Peter Richt´arik。《去中心化随机异步优化中的最优时间复杂度》。神经信息处理系统进展，37，2024年。

Alexander Tyurin 和 Danil Sivtsov。Birch随机梯度下降：局部异步随机梯度下降方法的树图框架。*arXiv* 预印本 *arXiv:2505.09218*，2025年。

Alexander Tyurin、Kaja Gruntkowska 和 Peter Richt´arik。Freya PAGE：异构异步计算下大规模非凸有限和优化的首个最优时间复杂度。神经信息处理系统进展，37，2024年a。

Alexander Tyurin、Marta Pozzi、Ivan Ilin 和 Peter Richt´arik。Shadowheart SGD：在任意计算与通信异构性下具有最优时间复杂度的分布式异步随机梯度下降。神经信息处理系统进展，37，2024年b。

N Denizcan Vanli、Mert Gurbuzbalaban 和 Asuman Ozdaglar。近端增量聚合梯度方法的全局收敛速率。*SIAM* 优化期刊，28(2)：1282–1300，2018年。

Zhongwei Wan、Xin Wang、Che Liu、Samiul Alam、Yu Zheng、Jiachen Liu、Zhongnan Qu、Shen Yan、Yi Zhu、Quanlu Zhang、Mosharaf Chowdhury 和 Mi Zhang。高效大语言模型：一项综述。机器学习研究汇刊，2024年。ISSN 2835-8856。URL https://openreview.net/forum?id=bsCCJHbO8A。综述认证。

Qiyuan Wang、Qianqian Yang、Shibo He、Zhiguo Shi 和 Jiming Chen。AsyncFedED：基于欧几里得距离的自适应权重聚合异步联邦学习。*arXiv* 预印本 *arXiv:2205.13797*，2022a。

Siwei Wang 和 Wei Chen。组合半赌博机的汤普森采样。发表于国际机器学习会议，第 5114–5122 页。PMLR，2018年。

Xiaolu Wang、Zijian Li、Shi Jin 和 Jun Zhang。异质客户端异步联邦学习中的线性加速。*IEEE* 移动计算汇刊，2024年。

Xiaolu Wang、Yuchang Sun、Hoi To Wai 和 JunZhang。适用于任意异构数据的增量聚合异步随机梯度下降，2025年。URL https://openreview.net/forum?id=m3x4kDbYAK。

Zhongyu Wang、Zhaoyang Zhang、Yuqing Tian、Qianqian Yang、Hangguan Shan、Wei Wang 和 Tony QS Quek。无线通信网络上的异步联邦学习。*IEEE* 无线通信汇刊，21(9):6961–6978，2022b。

Blake E Woodworth、Jialei Wang、Adam Smith、Brendan McMahan 和 Nati Srebro。《并行随机优化的图预言机模型、下界及差距》。神经信息处理系统进展，第31卷，2018年。

Xuyang Wu、Sindri Magnusson、Hamid Reza Feyzmahdavian 和 Mikael Johansson。《异步学习的延迟自适应步长》。载于 Kamalika Chaudhuri、Stefanie Jegelka、Le Song、Csaba Szepesvari、Gang Niu 和 Sivan Sabato （编），第 *39* 届国际机器学习会议论文集，机器学习研究论文集第162卷，机器学习研究论文集，第24093–24113页。PMLR，2022年7月17–23日。网址： https://proceedings.mlr.press/v162/wu22g.html。

Han Xiao、Kashif Rasul 和 Roland Vollgraf。Fashion-MNIST：一种用于机器学习算法基准测试的新型图像数据集。*arXiv* 预印本 *arXiv:1708.07747*，2017年。

CongXie、SanmiKoyejo 和 Indranil Gupta。异步联邦优化。*arXiv* 预印本 *arXiv:1903.03934*，2019年。

Bowen Yang、Jian Zhang、Jonathan Li、Christopher R´e、Christopher R Aberger 和 Christopher De Sa。Pipemare：异步流水线并行深度神经网络训练。*arXiv* 预印本 *arXiv:1910.05124*，2019年。

Mikhail Yurochkin、Mayank Agarwal、Soumya Ghosh、Kristjan Greenewald、Nghia Hoang 和 Yasaman Khazaeni。神经网络的贝叶斯非参数联邦学习。载于Kamalika Chaudhuri 和 Ruslan Salakhut-dinov（编），第*36*届国际机器学习会议论文集，第97卷，机器学习研究论文集，第7252–7261页。PMLR，2019年6月9日至15日。网址：https://proceedings.mlr.press/v97/yurochkin19a.html。

Feilong Zhang、Xianming Liu、Shiyi Lin、Gang Wu、Xiong Zhou、Junjun Jiang 和 Xiangyang Ji。《无人闲置：采用并行边缘与服务器计算的高效异构联邦学习》。载于 国际机器学习会议，第 41399–41413 页。PMLR，2023年。

Shuai Zhang、Lina Yao、Aixin Sun 和 Yi Tay。《基于深度学习的推荐系统：综述与新视角》。*ACM*计算综述（*CSUR*），52（1）：1–38，2019年。

Shen-Yi Zhao 和 Wu-Jun Li。《快速异步并行随机梯度下降：一种具有收敛保证的无锁方法》。第 30 卷，2016年。

Yue Zhao、Meng Li、Liangzhen Lai、Naveen Suda、Damon Civin 和 Vikas Chan-dra。非独立同分布数据下的联邦学习。*arXiv* 预印本 *arXiv:1806.00582*，2018年。

Shuxin Zheng、Qi Meng、Taifeng Wang、Wei Chen、Nenghai Yu、Zhi-Ming Ma 和 Tie-Yan Liu。带延迟补偿的异步随机梯度下降。载于 Doina Precup 和 Yee Whye Teh（编），第*34*届国际机器学习会议论文集，机器学习研究论文集第*70*卷，第4120–4129页。PMLR，2017年8月6日至11日。网址https://proceedings.mlr.press/v70/zheng17b.html。

Kaiwen Zhou、Fanhua Shang 和 James Cheng。一种具有快速收敛速率的简单随机方差缩减算法。载于国际机器学习会议，第 5980–5989 页。PMLR，2018年。

# 附录

# 附录 A

# 第 2 章附录

## A.1 引理 2.4.2 的证明

为了分析RingmasterASGD的时间复杂度，我们首先给出完成固定数量更新所需时间的一个上界。为完整起见，我们先重述该引理，再给出其证明。

**引理 2.4.2。** 设工作节点的计算时间满足 固定计算模型 (2.1)。设 $R$ 为算法 2.3.1 或算法 2.3.2的延迟阈值。算法 $R$ 2.3.1 或算法 2.3.2 的任意连续迭代更新2.3.1 或算法2.3.2所需时间至多为

$$t(R) := 2 \min_{m \in [n]} \left( \frac{1}{m} \sum_{i=1}^{m} \frac{1}{\tau_i} \right)^{-1} \left( 1 + \frac{R}{m} \right). \tag{2.6}$$

证明。 现在我们聚焦于算法 2.3.2。考虑 $t(R)$ 的一种简化记法：

$$t := t(R) = 2 \min_{m \in [n]} \left( \sum_{i=1}^{m} \frac{1}{\tau_i} \right)^{-1} (R + m) = 2 \min_{m \in [n]} \left( \frac{1}{m} \sum_{i=1}^{m} \frac{1}{\tau_i} \right)^{-1} \left( 1 + \frac{R}{m} \right).$$

固定任意一次迭代 $k$，并考察从 $k$ 到$k + R - 1$的连续迭代。根据算法2.3.2的设计，每个工作节点最多会暂停一次，也就是说，每个工作节点最多会有一个随机梯度被忽略。此后，在这些迭代期间，后续随机梯度的延迟不会超过$R - 1$ 。只要某个工作节点完成一个随机梯度的计算，就会立即开始计算新的随机梯度。

用反证法，假设算法2.3.2 无法在 $t$之前完成第 $k + R - 1$ 次迭代。同时，到 $t$时，所有工作节点都至少会计算

$$\sum_{i=1}^{n} \max \left\{ \left\lfloor \frac{t}{\tau_i} \right\rfloor - 1, 0 \right\} \tag{A.1}$$

延迟小于 $R$ 的随机梯度，因为 $\lfloor t/\tau_i \rfloor$ 是工作节点 $i$ 在 $t$ 秒后能够计算出的随机梯度数量。我们减去 1，以考虑到一个具有较大延迟的随机梯度可能会被

忽略。我们取一个索引

$$j^* = \arg\min_{m\in[n]} \left(\sum_{i=1}^{m} \frac{1}{\tau_i}\right)^{-1} (R+m).$$

由于对所有 $x \geq 0$ 都有 $\lfloor x \rfloor \geq x-1$，因此我们得到

$$\begin{aligned}
\sum_{i=1}^{n} \max\left\{\left\lfloor \frac{t}{\tau_i}\right\rfloor - 1, 0\right\} &\geq \sum_{i=1}^{j^*} \max\left\{\left\lfloor \frac{t}{\tau_i}\right\rfloor - 1, 0\right\} \geq \sum_{i=1}^{j^*} \left(\left\lfloor \frac{t}{\tau_i}\right\rfloor - 1\right) \\
&\geq \sum_{i=1}^{j^*} \frac{t}{\tau_i} - 2j^* \\
&= 2\left(\sum_{i=1}^{j^*} \frac{1}{\tau_i}\right)\left(\left(\sum_{i=1}^{j^*} \frac{1}{\tau_i}\right)^{-1} (R+j^*)\right) - 2j^* \\
&= 2R + 2j^* - 2j^* \geq R.
\end{aligned}$$

我们可以得出结论：到时间（2.6）时，算法将计算 $R$ 个延迟小于 $R$ 的随机梯度，并完成第 $k+R-1$ 次迭代，这与假设矛盾。

算法 2.3.1 的证明本质上相同。在算法 2.3.1 中，每个工作节点有一个延迟较大的随机梯度将被忽略，其余所有随机梯度都将用于优化过程。

□

## A.2 引理 2.5.1 的证明

为处理工作节点速度随时间变化的动态设置，我们现在在通用计算模型下证明引理 2.4.2 的对应结论。

**引理 2.5.1。** 设工作节点的计算时间满足 通用计算模型。设 $R$ 为算法 2.3.1 或算法 2.3.2 的延迟阈值。假设某次迭代在时间 $T^0$ 开始。从这次迭代开始，算法 $R$ 的连续迭代更新将于时间 2.3.1 或算法 2.3.2 之前完成

$$T(R,T^0) \coloneqq \min\left\{T \geq 0 : \sum_{i=1}^{n} \left\lfloor \frac{1}{4}\int_{T^0}^{T} p_i(\tau)d\tau \right\rfloor \geq R\right\}.$$

证明。设

$$T \coloneqq T(R,T^0) = \min\left\{T \geq 0 : \sum_{i=1}^{n} \left\lfloor \frac{1}{4}\int_{T^0}^{T} v_i(\tau)d\tau \right\rfloor \geq R\right\}.$$

一开始，这个证明遵循引理 2.4.2 的证明，直到（A.1）。这里我们也使用反证法，并假设算法 2.3.2 到时间 $R$ 时将无法执行连续 $T$ 次迭代更新。

与（A.1）不同，所有工作节点至少会计算

$$N := \sum_{i=1}^{n} \max\left\{\left\lfloor \int_{T^0}^{T} v_i(\tau)d\tau \right\rfloor - 1, 0\right\}$$

截至时间 $T$，由于（2.11），$\left\lfloor \int_{T^0}^{T} v_i\left(\tau\right) d\tau \right\rfloor$ 表示工作节点 $i$ 将在区间 $[T^0, T]$ 内计算的随机梯度数量。我们定义 $V_i\left(T\right) := \int_{T^0}^{T} v_i\left(\tau\right) d\tau$ 因此，

$$N = \sum_{i=1}^{n} \max\left\{\lfloor V_i(T) \rfloor - 1, 0\right\} \text{ and } T = \min\left\{T \geq 0 : \sum_{i=1}^{n} \left\lfloor \frac{V_i(T)}{4} \right\rfloor \geq R\right\}.$$

我们进一步定义

$$S := \left\{i \in [n] \,:\, \frac{V_i(T)}{4} \geq 1\right\}. \tag{A.2}$$

注意

$$\sum_{i=1}^{n} \left\lfloor \frac{V_i(T)}{4} \right\rfloor = \sum_{i \in S} \left\lfloor \frac{V_i(T)}{4} \right\rfloor, \tag{A.3}$$

由于对所有 $i \notin S$ 都有 $\lfloor V_i\left(T\right) / 4 \rfloor = 0$，利用简单代数可得

$$\begin{aligned} N &= \sum_{i=1}^{n} \max\{\lfloor V_i(T) \rfloor - 1, 0\} \geq \sum_{i \in S} \max\{\lfloor V_i(T) \rfloor - 1, 0\} \\ &\geq \sum_{i \in S} \lfloor V_i(T) \rfloor - |S| \geq \sum_{i \in S} V_i(T) - 2|S| \,, \end{aligned}$$

其中，最后一个不等式是由于 $\lfloor x \rfloor \geq x - 1$ 对所有 $x \in \mathbb{R}$ 成立。接下来，利用（A.2），我们有

$$\begin{aligned} N &\geq \frac{1}{2} \sum_{i \in S} V_i(T) + \frac{1}{2} \sum_{i \in S} 4 - 2|S| = \sum_{i \in S} \frac{V_i(T)}{2} \\ &\geq \sum_{i \in S} \left\lfloor \frac{V_i(T)}{4} \right\rfloor \overset{\text{(A.3)}}{=} \sum_{i=1}^{n} \left\lfloor \frac{V_i(T)}{4} \right\rfloor \geq R \,, \end{aligned}$$

其中，最后一个不等式是由 $T$ 的定义得出的。如引理 2.4.2 的证明中所述，我们可以得出：截至时间 $T = T\left(R,\ T^0\right)$ 时，该算法将计算 $R$ 个延迟小于 $R$ 的随机梯度，并完成迭代 $k + R - 1$，这与假设矛盾。

□

## A.3 定理证明 2.4.1

下面重述一个结果，该结果给出了 RingmasterASGD 达到所需平稳性水平所需的迭代次数上界，随后给出其证明。

**定理 2.4.1**。 在假设2.1.1、2.1.2和2.1.3成立的条件下，令RingmasterASGD （算法 2.3.1 或算法 2.3.2）中的步长为

$$\gamma = \min\left\{\frac{1}{2RL}, \frac{\varepsilon}{4L\sigma^2}\right\}.$$

那么，以下结论成立

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right] \leq \varepsilon\ ,$$

for

$$K \geq \frac{8RL\Delta}{\epsilon} + \frac{16\sigma^2 L\Delta}{\epsilon^2}\ , \tag{2.5}$$

其中， $R \in \{1, 2, \dots\}$ 是任意的延迟阈值。

我们采用 Koloskova 等人 (2022年) 的证明策略，并依赖以下两个引理。

**引理 A.3.1** (下降引理；证明见附录 A.3.1)。 在假设 2.1.1 和 2.1.3下，如果 RingmasterASGD (算法 2.3.1 或算法 2.3.2) 中的步长满足 $\gamma \leq 1/2L$，则以下不等式成立：

$$\mathbb{E}_{k+1}\left[f\left(x^{k+1}\right)\right] \leq f\left(x^k\right) - \frac{\gamma}{2}\left\|\nabla f\left(x^k\right)\right\|^2 - \frac{\gamma}{4}\left\|\nabla f\left(x^{k-\delta^k}\right)\right\|^2 + \frac{\gamma L^2}{2}\left\|x^k - x^{k-\delta^k}\right\|^2 + \frac{\gamma^2 L}{2}\sigma^2\ ,$$

其中， $\mathbb{E}_{k+1}[\cdot]$ 表示以迭代 $k$ 为止的所有随机性为条件的期望。

**引理 A.3.2** (残差估计；证明见附录 A.3.2)。 在假设 2.1.1 和 2.1.3下，RingmasterASGD (算法 2.3.1 或算法 2.3.2) 的迭代序列在步长为 $\gamma \leq 1/2RL$ 时满足以下界：

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\left\|x^k - x^{k-\delta^k}\right\|^2\right] \leq \frac{1}{2L^2(K+1)}\sum_{k=0}^{K}\mathbb{E}\left[\left\|\nabla f\left(x^{k-\delta^k}\right)\right\|^2\right] + \frac{\gamma}{L}\sigma^2.$$

有了这些结果，我们现在可以证明定理2.4.1。

定理 *2.4.1* 的证明。 我们先对 $K+1$ 次迭代取平均，并将不等式两边除以 $\gamma$ ，该不等式来自引理 A.3.1：

$$\frac{1}{K+1}\sum_{k=0}^{K}\left(\frac{1}{2}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right] + \frac{1}{4}\mathbb{E}\left[\left\|\nabla f(x^{k-\delta^k})\right\|^2\right]\right) \leq \frac{\Delta}{\gamma(K+1)} + \frac{\gamma L}{2}\sigma^2 + \frac{1}{K+1}\frac{L^2}{2}\sum_{k=0}^{K}\mathbb{E}\left[\left\|x^k - x^{k-\delta^k}\right\|^2\right].$$

接下来，将引理 A.3.2 应用于最后一项，可得：

$$\frac{1}{K+1}\sum_{k=0}^{K}\left(\frac{1}{2}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right]+\frac{1}{4}\mathbb{E}\left[\left\|\nabla f(x^{k-\delta^k})\right\|^2\right]\right)\leq\frac{\Delta}{\gamma(K+1)}+\frac{\gamma L}{2}\sigma^2 + \frac{1}{4(K+1)}\sum_{k=0}^{K}\mathbb{E}\left[\left\|\nabla f\left(x^{k-\delta^k}\right)\right\|^2\right]+\frac{\gamma L}{2}\sigma^2.$$

进一步简化可得：

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right]\leq\frac{2\Delta}{\gamma(K+1)}+2\gamma L\sigma^2.$$

现在，我们选择步长 $\gamma$ 为

$$\gamma=\min\left\{\frac{1}{2RL},\frac{\varepsilon}{4L\sigma^2}\right\}\leq\frac{1}{2RL}\ .$$

选择 $\gamma$后，接下来只需选择

$$K\geq\frac{8\Delta RL}{\epsilon}+\frac{16\Delta L\sigma^2}{\epsilon^2}$$

以确保

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right]\leq\epsilon\ .$$

证明至此完成。 □

## A.3.1 引理A.3.1

下面重述 Ringmaster ASGD 收敛分析中用到的下降引理，并给出其证明。

**引理A.3.1** （下降引理）。在假设2.1.1 和2.1.3下，如果RingmasterASGD （算法2.3.1 或算法2.3.2）中的步长满足 $\gamma\leq1/2L$，则有以下不等式成立：

$$\mathbb{E}_{k+1}\left[f\left(x^{k+1}\right)\right]\leq f\left(x^k\right)-\frac{\gamma}{2}\left\|\nabla f\left(x^k\right)\right\|^2-\frac{\gamma}{4}\left\|\nabla f\left(x^{k-\delta^k}\right)\right\|^2 + \frac{\gamma L^2}{2}\left\|x^k-x^{k-\delta^k}\right\|^2+\frac{\gamma^2L}{2}\sigma^2,$$

其中， $\mathbb{E}_{k+1}\left[\cdot\right]$ 表示在迭代 $k$ 之前所有随机性条件下的期望。

证明。 假设我们在计算 $x^{k+1}$ 时，从索引为 $i_k$ 的工作节点获得了一个随机梯度。由于函数 $f$ 是 $L$-光滑的（假设2.1.1），我们

有（Nesterov，2018年）：

$$\mathbb{E}_{k+1}\left[f\left(x^{k+1}\right)\right] \leq f\left(x^{k}\right) - \gamma \underbrace{\mathbb{E}_{k+1}\left[\left\langle \nabla f\left(x^{k}\right), \nabla f\left(x^{k-\delta^{k}}; \xi_{i_k}^{k-\delta^{k}}\right)\right\rangle\right]}_{=:t_1} + \frac{L}{2}\gamma^{2} \underbrace{\mathbb{E}_{k+1}\left[\left\|\nabla f\left(x^{k-\delta^{k}}; \xi_{i_k}^{k-\delta^{k}}\right)\right\|^{2}\right]}_{=:t_2}.$$

根据假设 2.1.3，我们将第二项估计为

$$\begin{aligned} t_1 &= \left\langle \nabla f\left(x^{k}\right), \nabla f\left(x^{k-\delta^{k}}\right)\right\rangle \\ &= \frac{1}{2}\left[\left\|\nabla f\left(x^{k}\right)\right\|^{2} + \left\|\nabla f\left(x^{k-\delta^{k}}\right)\right\|^{2} - \left\|\nabla f\left(x^{k}\right) - \nabla f\left(x^{k-\delta^{k}}\right)\right\|^{2}\right]. \end{aligned}$$

利用方差分解等式和假设2.1.3，可得

$$\begin{aligned} t_2 &= \mathbb{E}_{k+1}\left[\left\|\nabla f\left(x^{k-\delta^{k}}; \xi_{i_k}^{k-\delta^{k}}\right) - \nabla f\left(x^{k-\delta^{k}}\right)\right\|^{2}\right] + \left\|\nabla f\left(x^{k-\delta^{k}}\right)\right\|^{2} \\ &\leq \sigma^{2} + \left\|\nabla f\left(x^{k-\delta^{k}}\right)\right\|^{2}. \end{aligned}$$

结合 $t_1$ 和 $t_2$的结果，并利用 $L$光滑性对$\|\nabla f(x^k) - \nabla f(x^{k-\delta^k})\|^2$进行界定，可得

$$\begin{aligned} \mathbb{E}_{k+1}\left[f\left(x^{k+1}\right)\right] \leq f\left(x^{k}\right) &- \frac{\gamma}{2}\left\|\nabla f\left(x^{k}\right)\right\|^{2} - \frac{\gamma}{2}(1-\gamma L)\left\|\nabla f\left(x^{k-\delta^{k}}\right)\right\|^{2} \\ &+ \frac{\gamma L^{2}}{2}\left\|x^{k} - x^{k-\delta^{k}}\right\|^{2} + \frac{\gamma^{2} L}{2}\sigma^{2}. \end{aligned}$$

最后，应用条件 $\gamma \leq 1/2L$ 即可完成证明。 □

## A.3.2 引理A.3.2的证明

下面我们重述用于界定RingmasterASGD 相邻迭代点之间期望残差的引理，并给出其证明。

**引理A.3.2** (残差估计)。在假设2.1.1 和2.1.3下，RingmasterASGD (算法2.3.1 或算法2.3.2）的迭代序列在步长$\gamma \leq 1/2RL$ 下满足以下界：

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\left\|x^{k} - x^{k-\delta^{k}}\right\|^{2}\right] \leq \frac{1}{2L^{2}(K+1)}\sum_{k=0}^{K}\mathbb{E}\left[\left\|\nabla f\left(x^{k-\delta^{k}}\right)\right\|^{2}\right] + \frac{\gamma}{L}\sigma^{2}.$$

证明。 假设计算 $x^{k+1}$时，我们从索引为 $i_k$的工作节点获得了一个随机梯度。我们首先展开差值并应用

tower 性质、假设 2.1.3、杨氏不等式和詹森不等式：

$$\begin{aligned}\mathbb{E}\left[\left\|x^k - x^{k-\delta^k}\right\|^2\right] &= \mathbb{E}\left[\left\|\sum_{j=k-\delta^k}^{k-1} \gamma \nabla f\left(x^{j-\delta_j}; \xi_{i_j}^{j-\delta_j}\right)\right\|^2\right] \\ &\leq 2\mathbb{E}\left[\left\|\sum_{j=k-\delta^k}^{k-1} \gamma \nabla f\left(x^{j-\delta_j}\right)\right\|^2\right] \\ &\quad + 2\mathbb{E}\left[\left\|\sum_{j=k-\delta^k}^{k-1} \gamma\left(\nabla f\left(x^{j-\delta_j}; \xi_{i_j}^{j-\delta_j}\right) - \nabla f\left(x^{j-\delta_j}\right)\right)\right\|^2\right] \\ &\leq 2\mathbb{E}\left[\left\|\gamma \sum_{j=k-\delta^k}^{k-1} \nabla f\left(x^{j-\delta_j}\right)\right\|^2\right] + 2\delta^k\gamma^2\sigma^2 \\ &\leq 2\delta^k\gamma^2 \sum_{j=k-\delta^k}^{k-1} \mathbb{E}\left[\left\|\nabla f\left(x^{j-\delta_j}\right)\right\|^2\right] + 2\delta^k\gamma^2\sigma^2 \ .\end{aligned}$$

利用 $\gamma \leq 1/2RL$ 和 $\delta^k \leq R-1$ （根据设计），我们得到：

$$\mathbb{E}\left[\left\|x^k - x^{k-\delta^k}\right\|^2\right] \leq \frac{1}{2L^2R} \sum_{j=k-\delta^k}^{k-1} \mathbb{E}\left[\left\|\nabla f\left(x^{j-\delta_j}\right)\right\|^2\right] + \frac{\gamma}{L}\sigma^2 \ .$$

接下来，对所有迭代 $k=0$、.....、$K$求和，可得：

$$\sum_{k=0}^{K} \mathbb{E}\left[\left\|x^k - x^{k-\delta^k}\right\|^2\right] \leq \frac{1}{2L^2R} \sum_{k=0}^{K} \sum_{j=k-\delta^k}^{k-1} \mathbb{E}\left[\left\|\nabla f\left(x^{j-\delta_j}\right)\right\|^2\right] + (K+1)\frac{\gamma}{L}\sigma^2 \ .$$

注意，右侧求和中的每个平方范数 $\|\nabla f(x^{j-\delta_j})\|^2$由于算法的设计，最多出现 $R$ 次。具体而言，对于所有$k \geq 0$， $\delta^k \leq R-1$，从而确保求和中出现的平方范数不超过 $R$ 个。因此：

$$\sum_{k=0}^{K} \mathbb{E}\left[\left\|x^k - x^{k-\delta^k}\right\|^2\right] \leq \frac{1}{2L^2} \sum_{k=0}^{K} \mathbb{E}\left[\left\|\nabla f\left(x^{k-\delta^k}\right)\right\|^2\right] + (K+1)\frac{\gamma}{L}\sigma^2.$$

最后，将不等式除以 $K+1$，证明即告完成。 □

## A.4 定理 2.5.2 的证明

下面重述用于建立RingmasterASGD在通用计算模型下收敛性的定理，然后给出其证明。

**T定理2.5.2**。令假设2.1.1、2.1.2和2.1.3 成立。令步长 ze

在RingmasterASGD （算法2.3.1 或算法2.3.2）中为

$$\gamma = \min\left\{\frac{1}{2RL}, \frac{\varepsilon}{4L\sigma^2}\right\},$$

和延迟阈值

$$R = \max\left\{1, \left\lceil\frac{\sigma^2}{\varepsilon}\right\rceil\right\}.$$

那么，在通用计算模型下，Ringmaster ASGD 至多经过 $T^{\bar{K}}$ 秒后找到一个 $\varepsilon$– 驻点，其中

$$\bar{K} := \left\lceil\frac{48L\Delta}{\epsilon}\right\rceil$$

并且 $T^{\bar{K}}$ 是以下递归定义序列中的第 $\bar{K}$ 个元素:

$$T^k := \min\left\{T \geq 0 : \sum_{i=1}^{n}\left\lfloor\frac{1}{4}\int_{T^{k-1}}^{T} p_i(\tau)d\tau\right\rfloor \geq R\right\}$$

对所有 $k \geq 1$ 和 $T^0 = 0$均成立。

证明。 根据定理2.4.1，RingmasterASGD 的迭代复杂度为

$$K = \left\lceil\frac{8RL\Delta}{\epsilon} + \frac{16\sigma^2 L\Delta}{\epsilon^2}\right\rceil. \tag{A.4}$$

不失一般性，我们假设 $L\Delta > \varepsilon/2$。 [1] 因此，

$$\left\lceil\frac{8RL\Delta}{\epsilon} + \frac{16\sigma^2 L\Delta}{\epsilon^2}\right\rceil \leq \frac{16RL\Delta}{\epsilon} + \frac{32\sigma^2 L\Delta}{\epsilon^2}$$

and

$$K \leq R \times \left\lceil\frac{K}{R}\right\rceil = R \times \left\lceil\frac{16L\Delta}{\epsilon} + \frac{32\sigma^2 L\Delta}{R\epsilon^2}\right\rceil.$$

使用 $R$, 的选择，我们得到

$$K \leq R \times \left\lceil\frac{48L\Delta}{\epsilon}\right\rceil.$$

总而言之，这些算法需要通过 $R$ 次 $x^k$ 的连续更新，花费 $\lceil 48L\Delta/\epsilon\rceil$ ，才能找到一个 $\varepsilon$–驻点。定义 $K := \lceil 48L\Delta/\epsilon\rceil$。根据引理2.5.1，我们知道RingmasterASGD 最多需要

$$T^1 := T(R, 0) = \min\left\{T \geq 0 : \sum_{i=1}^{n}\left\lfloor\frac{1}{4}\int_{0}^{T} v_i(\tau)d\tau\right\rfloor \geq R\right\}$$

s秒来完成迭代序列的前 $R$ 次连续更新。由于该算法 o-

[1] 否则，利用 $L$–光滑性，$\left\|\nabla f(x^0)\right\|^2 \leq 2L\Delta \leq \varepsilon$，且初始点是一个 $\varepsilon$–驻点。

法将在最多 首次 $R$ 次连续更新后完成 $T^1$ 秒，并将在时间 $R+1$ 之前开始迭代 $T^1$。因此，再次利用引理 2.5.1 ，所需次数至多为

$$T^2 \coloneqq T(R, T^1) = \min\left\{T \geq 0 : \sum_{i=1}^{n} \left\lfloor \frac{1}{4} \int_{T^1}^{T} v_i(\tau) d\tau \right\rfloor \geq R\right\}$$

秒内完成前 $2 \times R$ 次连续更新。根据相同的推理，他们将在至多

$$T_{\bar{K}} \coloneqq T(R, T_{\bar{K}-1}) = \min\left\{T \geq 0 : \sum_{i=1}^{n} \left\lfloor \frac{1}{4} \int_{T_{\bar{K}-1}}^{T} v_i(\tau) d\tau \right\rfloor \geq R\right\}$$

秒内。 □

## A.5 第 2.2 节示例的推导2.2

令 $\tau_i = \sqrt{i}$ 对所有 $i \in [n]$，则

$$T_{\mathrm{R}} = \Theta\left(\min_{m\in[n]}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\sqrt{i}}\right)^{-1}\left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right)\right).$$

使用

$$\sum_{i=1}^{m}\frac{1}{\sqrt{i}} = \Theta\left(\sqrt{m}\right)$$

对于 $m \geq 1$，我们将该项化简为：

$$\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\sqrt{i}}\right)^{-1} = \Theta\left(\sqrt{m}\right) \ ,$$

$$T_{\mathrm{R}} = \Theta\left(\min_{m\in[n]}\sqrt{m}\left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right)\right) = \Theta\left(\min_{m\in[n]}\left(\frac{L\Delta\sqrt{m}}{\varepsilon} + \frac{\sigma^2 L\Delta}{\sqrt{m}\varepsilon^2}\right)\right).$$

当这两项达到平衡时，最小值即取得，即在

$$m = \min\left\{\left\lceil\frac{\sigma^2}{\varepsilon}\right\rceil, n\right\}.$$

代入 $m$ 的这个值，我们得到：

$$T_{\mathrm{R}} = \Theta\left(\max\left[\frac{\sigma L\Delta}{\varepsilon^{3/2}}, \frac{\sigma^2 L\Delta}{\sqrt{n}\varepsilon^2}\right]\right).$$

现在我们考虑 $T_{\mathrm{A}}$：

$$T_{\mathrm{A}} = \Theta\left(\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right)\right).$$

使用

$$\sum_{i=1}^{n} \frac{1}{\sqrt{i}} = \Theta\left(\sqrt{n}\right)$$

对于 $n \geq 1$，我们简化该项：

$$\left(\frac{1}{n}\sum_{i=1}^{n} \frac{1}{\sqrt{i}}\right)^{-1} = \Theta\left(\sqrt{n}\right).$$

将此结果代入 $T_{\mathrm{A}}$，我们得到：

$$\begin{aligned} T_{\mathrm{A}} &= \Theta\left(\sqrt{n}\left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right)\right) = \Theta\left(\frac{L\Delta\sqrt{n}}{\varepsilon} + \frac{\sigma^2 L\Delta}{\sqrt{n}\varepsilon^2}\right) \\ &= \Theta\left(\max\left[\frac{L\Delta\sqrt{n}}{\varepsilon}, \frac{\sigma^2 L\Delta}{\sqrt{n}\varepsilon^2}\right]\right). \end{aligned}$$

## A.6 当初始点是一个 $\varepsilon$–平稳点时

在 $L$–光滑性（假设 2.1.1）的条件下，我们有：

$$f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2$$

对于所有 $x$， $y \in \mathbb{R}^d$。取 $y = x - \frac{1}{L}\nabla f(x)$，

$$f\left(x - \frac{1}{L}\nabla f(x)\right) \leq f(x) - \frac{1}{2L}\|\nabla f(x)\|^2.$$

由于 $f\left(x - \frac{1}{L}\nabla f(x)\right) \geq f^*$，并取 $x = x^0$，我们得到

$$\left\|\nabla f(x^0)\right\|^2 \leq 2L\Delta\,.$$

因此，如果 $2L\Delta \leq \varepsilon$，那么 $\|\nabla f(x^0)\|^2 \leq \varepsilon$。

# 附录 B

## 章节附录 3

## B.1 任意变化的计算时间

在实践中，固定计算模型 (3.2) 往往并不成立。设备的计算能力会随时间变化，原因可能是临时断连、硬件或网络延迟、处理能力波动或其他瞬态效应（Maranjyan 等人，2025a）。

本节将我们的理论扩展到任意变化的计算时间这一更一般的情形。

### B.1.1 通用计算模型

为形式化描述这一情形，我们采用由 *Tyurin*于 Tyurin（2024年）提出的通用计算模型。

对于每个工作节点 $i \in [n]$，我们定义一个计算能力函数

$$p_i : \mathbb{R}_+ \to \mathbb{R}_+ \ ,$$

假设其几乎处处非负且连续（允许存在可数个跳跃点）。对于任意 $T_2 \geq T_1 \geq 0$，工作节点 $i$ 在 $[T_1, T_2]$ 上已完成的随机梯度数量为

$$\#\text{gradients in } [T_1, T_2] \ = \ \left\lfloor \int_{T_1}^{T_2} p_i(t)\, dt \right\rfloor .$$

这里，$p_i(t)$ 描述了工作节点时变的计算能力：在某个区间内，取值越小，已完成的梯度就越少；取值越大，已完成的梯度就越多。

例如，如果工作节点 $i$ 在最初的 $T$ 秒内一直处于空闲状态，之后才开始活动，那么这对应于 $p_i(t)$ 在 $t \leq T$ 时为 $= 0$，且 $p_i(t)$ 在 $t > T$ 时为 $> 0$。更一般地，$p_i(t)$ 可能呈现周期性或非规则模式，从而导致计算能力出现活动突发、暂停或混沌变化。过程 $p_i(t)$ 甚至可能是随机的，并且所有结果均在给定 $\{p_i\}$ 的实现样本路径的条件下成立。

该通用计算模型在固定计算模型（3.2）情形下退化为 $p_i(t) = 1/\tau_i$ 对所有 $t \geq 0$ 和 $i \in [n]$均成立。在这种情况下，

$$\#\text{gradients in } [T_1, T_2] = \left\lfloor \frac{T_2 - T_1}{\tau_i} \right\rfloor ,$$

这意味着工作节点 $i$ 在 $T_1 + \tau_i$ 秒后计算出一个随机梯度，在 $T_1 + 2\tau_i$ 秒后计算出两个梯度，依此类推。

### B.1.2 迈向最优方法

在计算时间任意变化的一般情形下，算法 3.5.1并非最优的。为了解原因，请考虑以下对抗性时间模式。

假设有两个工作节点。在较慢的工作节点完成一次梯度计算期间，较快的工作节点计算 $s$ 个梯度。紧接着，两者交换角色：先前较快的工作节点速度减慢为原来的 $s$ 倍，而先前较慢的工作节点则以相同倍数提速。每当较慢的工作节点完成一次梯度计算时，这种模式就会重复。

在此设置下，如果运行算法 3.5.1，服务器在每个阶段 1 中都会等待每个工作节点计算出一个梯度。因此，较慢的工作节点始终只贡献一个梯度，批次大小的调和平均数满足

$$1 \;\leq\; B^k \;\leq\; 2\,.$$

根据定理3.6.3，迭代复杂度为

$$\mathcal{O}\left(\frac{nL\Delta}{\varepsilon}\left(1+\frac{\sigma^2}{Bn\varepsilon}\right)\right).$$

当 $\sigma^2/n\varepsilon$ 远大于 $B$ 时，这种依赖关系可能会远非最优。

相反，假设服务器等待上述过程完整进行一轮，并从每个工作节点收集 $s+1$ 个梯度。此时调和平均数满足 $B^k \geq s+1$，该值可以任意大于 2。由于实际中 $s$ 和 $\sigma^2/n\varepsilon$ 都可能非常大，因此，在任意时间设置下，只等待每个工作节点计算出一个梯度的朴素策略（如算法 3.5.1）不可能是最优的。

### B.1.3 一种最优方法

解决方案很简单，可直接由迭代复杂度界得出。根据

$$\mathcal{O}\left(\frac{nL\Delta}{\varepsilon}\left(1+\frac{\sigma^2}{Bn\varepsilon}\right)\right),$$

我们可以看出，要平衡各项，只需确保

$$B \;\geq\; \frac{\sigma^2}{n\varepsilon}\,.$$

因此，我们修改算法 3.5.1 阶段 1 的停止条件。不再要求服务器从每个工作节点至少接收一个梯度，而是要求满足 MaleniaSGD 中采用的更强条件，即

$$\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{b_i}\right)^{-1} \;\geq\; \max\left\{1, \frac{\sigma^2}{n\varepsilon}\right\}\,, \tag{3.3}$$

其中 $b_i$ 是从工作节点 $i$ 接收到的梯度数量。

在低噪声区域，当 $\sigma^2/n\varepsilon \leq 1$ 时，该条件可简化为要求$b_i \geq 1$ 对所有 $i$ 都成立，因此该算法与原始算法 3.5.1一致。在高噪声区域，该算法会在阶段 1 收集更多梯度，确保 $B$ 足够大，从而实现最优收敛。

经过这一改动，我们算法的阶段 1 与MaleniaSGD相同。区别在于阶段 2：我们的算法继续利用所有工作节点正在进行的梯度计算来执行 $n$ 更新，而 MaleniaSGD 会丢弃所有尚未完成的梯度，执行一次更新，然后进入下一轮。

以下定理给出了我们的算法在通用计算模型下的时间复杂度。

**定理 B.1.1**。 在假设 3.3.1、3.3.2 和3.3.5下，令RingleaderASGD 中的步长为

$$\gamma = \frac{1}{10nL} .$$

那么，在通用计算模型下，Ringleader ASGD 至多用 $T^{\bar{K}}$ 秒即可找到一个 $\varepsilon$ 阶驻点，其中

$$\bar{K} := \left\lceil \frac{160L\Delta}{\varepsilon} \right\rceil ,$$

且 $T^{\bar{K}}$ 表示递归定义序列中的第 $\bar{K}$ 个元素

$$T^k = \min \left\{ T \geq 0 : \left( \frac{1}{n} \sum_{i=1}^{n} \left\lfloor \int_{T_{k-1}}^{T} p_i(t)\, dt \right\rfloor^{-1} \right)^{-1} \geq \max\left\{1, \frac{\sigma^2}{n\varepsilon}\right\} \right\} ,$$

对所有 $k \geq 1$ 均成立，初始化为 $T^0 = 0$。

该结果与 Tyurin (2024年) 推导出的下界一致，因此所提出的方法是最优的。

证明。 在条件（3.3）下，该算法的每个梯度型步骤都满足

$$B^k = \left( \frac{1}{n} \sum_{i=1}^{n} \frac{1}{b_i^k} \right)^{-1} \geq \max\left\{1, \frac{\sigma^2}{n\varepsilon}\right\} .$$

在定理3.6.3中，我们可以使用任何有效的下界来替代 $B$ 。这里我们选择

$$B = \max\left\{1, \frac{\sigma^2}{n\varepsilon}\right\} .$$

进行这一替代后，迭代复杂度变为

$$K = \frac{80\, nL\Delta}{\varepsilon} .$$

为推导时间复杂度，考虑执行 $n$ 次迭代所需的时间。每一批 $n$ 次更新都在阶段 1 收集梯度后，于阶段 2 执行。从时间 $T = 0$开始，当累积的梯度数量满足条件（3.3）时，阶段 1 结束，此时的时间为

$$T_+^1 = \min \left\{ T \geq 0 : \left( \frac{1}{n} \sum_{i=1}^{n} \left\lfloor \int_{0}^{T} p_i(t)\, dt \right\rfloor^{-1} \right)^{-1} \geq \max\left\{1, \frac{\sigma^2}{n\varepsilon}\right\} \right\} .$$

阶段 1 结束后，为在阶段 2 完成 $n$ 次更新，我们必须等待正在进行的计算完成。这最多需要

$$T^1 = \min\left\{T \geq 0 : \left(\frac{1}{n}\sum_{i=1}^{n}\left\lfloor\int_{T_+^1}^{T} p_i(t)\,dt\right\rfloor^{-1}\right)^{-1} \geq 1\right\}.$$

因此，完成所有 $K$ 次迭代所需的总时间有界于

$$T^{\lceil 2K/n\rceil},$$

其中，序列 $\{T^k\}_{k\geq 0}$ 递归定义为

$$T^k = \min\left\{T \geq 0 : \left(\frac{1}{n}\sum_{i=1}^{n}\left\lfloor\int_{T_{k-1}}^{T} p_i(t)\,dt\right\rfloor^{-1}\right)^{-1} \geq \max\left\{1, \frac{\sigma^2}{n\varepsilon}\right\}\right\}.$$

且 $T^0 = 0$。 □

## B.2 辅助引理

这里我们给出正文中省略的引理证明，以及后文将用到的辅助结果。

### B.2.1 引理的证明3.3.4

我们先给出一个引理，说明不同光滑常数之间的关系。

**引理 3.3.4** (光滑性界)。 令 $L_f$ 表示$f$的光滑常数， $L_{f_i}$ 表示 $f_i$的光滑常数，而 $L$ 表示假设 3.3.2 中的常数。则有

$$L_f \;\leq\; L \;\leq\; \sqrt{\frac{1}{n}\sum_{i=1}^{n} L_{f_i}^2} \;\leq\; \max_{i\in[n]} L_{f_i} =: L_{\max}.$$

此外，如果所有的 $f_i$ 都相同，即对所有 $i \in [n]$， $f_i = f$，那么 $L = L_f$。

回顾假设 3.3.2，我们假设了以下广义光滑性条件：存在某个常数 $L > 0$，使得对所有 $x \in \mathbb{R}^d$和 $y_1, \ldots, y_n \in \mathbb{R}^d$，

$$\left\|\nabla f(x) - \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(y_i)\right\|^2 \;\leq\; \frac{L^2}{n}\sum_{i=1}^{n}\|x - y_i\|^2. \tag{B.1}$$

回顾一下，如果函数 $\phi$ 满足以下条件，则称其为 $L_\phi$–光滑:

$$\|\nabla\phi(x) - \nabla\phi(y)\| \leq L_\phi\|x - y\|, \quad \forall x, y \in \mathbb{R}^d.$$

这里， $L_\phi$ 表示满足条件的最小常数。现在我们可以证明该引理了 .

证明。 对于第一个不等式，取 $y_1 = \cdots = y_n = y$。那么（B.1）化简为 t o

$$\|\nabla f(x) - \nabla f(y)\|^2 \leq L^2\|x - y\|^2,$$

因此 $f$ 是 $L$–光滑的。根据 $L_f$ 作为最小光滑常数的定义，这意味着 $L_f \leq L$。

对于第二个不等式，先应用三角不等式，再利用每个 $f_i$的光滑性，最后应用柯西–施瓦茨，

$$\left\|\nabla f(x) - \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(y_i)\right\| \leq \frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(x) - \nabla f_i(y_i)\| \leq \frac{1}{n}\sum_{i=1}^{n} L_{f_i}\|x - y_i\|$$
$$\leq \sqrt{\frac{1}{n}\sum_{i=1}^{n} L_{f_i}^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\|x - y_i\|^2}\,.$$

两边平方可得（B.1）在 $L = \sqrt{\frac{1}{n}\sum_{i=1}^{n} L_{f_i}^2}$ 的条件下成立。~~最后，~~假设所有 $f_i$ 都相同： $f_i \equiv f$ 对所有 $i$ 均成立。于是

$$\left\|\nabla f(x) - \frac{1}{n}\sum_{i=1}^{n}\nabla f(y_i)\right\| \leq \frac{1}{n}\sum_{i=1}^{n}\|\nabla f(x) - \nabla f(y_i)\| \leq \frac{L_f}{n}\sum_{i=1}^{n}\|x - y_i\|$$
$$\leq L_f\sqrt{\frac{1}{n}\sum_{i=1}^{n}\|x - y_i\|^2}\,,$$

其中，最后一步用到了柯西–施瓦茨。两边平方可得

$$\left\|\nabla f(x) - \frac{1}{n}\sum_{i=1}^{n}\nabla f(y_i)\right\|^2 \leq \frac{L_f^2}{n}\sum_{i=1}^{n}\|x - y_i\|^2,$$

即，B.1）在 $L \leq L_f$ 的条件下成立。结合 $L_f \leq L$，可得 $L = L_f$。 □

## B.2.2 方差项

下一个引理给出了 RingleaderASGD 中所用梯度估计量方差的一个上界。

**引理 B.2.1** (方差上界)。 在假设3.3.1下，算法 3.5.1 中使用的梯度估计量满足以下方差型不等式

$$\mathbb{E}\left[\left\|\bar{g}^k - \frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] \leq \frac{\sigma^2}{B^k n}\,.$$

证明。 回顾一下，梯度估计量定义为

$$\bar{g}^k = \frac{1}{n}\sum_{i=1}^{n}\bar{g}_i^k = \frac{1}{n}\sum_{i=1}^{n} g_i^{k,j} = \frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k};\xi_i^{k-\delta_i^k,j}\right)\,.$$

令 $\mathcal{F}^k$ 表示包含当前轮次开始前所有随机性的 σ-代数，即截至迭代 $k-$ （$k$ mod $n$）。在给定 $\mathcal{F}^k$ 的条件下，评估点 $x^{k-\delta_i^k}$ 是确定的，而随机梯度 $g^k_{,}$ $_i{}^j$ 在不同工作节点 $i$和样本 $j$之间相互独立。

根据全期望公式和随机梯度的独立性，我们有

$$\begin{aligned}\mathbb{E}\left[\left\|\bar{g}^k-\frac{1}{n}\sum_{i=1}^n\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right]&=\mathbb{E}\left[\mathbb{E}\left[\left\|\bar{g}^k-\frac{1}{n}\sum_{i=1}^n\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\,\middle|\,\mathcal{F}^k\right]\right]\\&=\mathbb{E}\left[\frac{1}{n^2}\sum_{i=1}^n\mathbb{E}\left[\left\|\bar{g}_i^k-\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\,\middle|\,\mathcal{F}^k\right]\right].\end{aligned}$$

对于每个工作节点 $i$，小批量梯度估计量的条件方差为

$$\begin{aligned}\mathbb{E}\left[\left\|\bar{g}_i^k-\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\,\middle|\,\mathcal{F}^k\right]&=\mathbb{E}\left[\left\|\frac{1}{b_i^k}\sum_{j=1}^{b_i^k}g_i^{k,j}-\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\,\middle|\,\mathcal{F}^k\right]\\&=\frac{1}{b_i^k}\mathbb{E}\left[\left\|g_i^{k,1}-\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\,\middle|\,\mathcal{F}^k\right]\leq\frac{\sigma^2}{b_i^k},\end{aligned}$$

其中，最后一个不等式由假设3.3.1得出。

结合这些结果，我们得到

$$\mathbb{E}\left[\left\|\bar{g}^k-\frac{1}{n}\sum_{i=1}^n\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right]\leq\frac{1}{n^2}\sum_{i=1}^n\frac{\sigma^2}{b_i^k}=\frac{\sigma^2}{n}\frac{1}{n}\sum_{i=1}^n\frac{1}{b_i^k}=\frac{\sigma^2}{B^kn},$$

其中，最后一个等式用到了调和平均数的定义

$$B^k=\left(\frac{1}{n}\sum_{i=1}^n\frac{1}{b_i^k}\right)^{-1}.$$

□

### B.2.3 引理3.6.1 的证明

现在我们来证明下降引理。

**引理 3.6.1** （下降引理）。 在假设 3.3.1 和 3.3.2下，如果算法 3.5.1中的步长满足 $\gamma\leq 1/4L$，则有以下不等式成立

$$\begin{aligned}\mathbb{E}\left[f\left(x^{k+1}\right)\right]\leq\mathbb{E}\left[f\left(x^k\right)\right]&-\frac{\gamma}{2}\mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right]-\frac{\gamma}{4}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^n\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right]\\&+\frac{\gamma L^2}{2n}\sum_{i=1}^n\mathbb{E}\left[\left\|x^k-x^{k-\delta_i^k}\right\|^2\right]+\frac{3\gamma^2L\sigma^2}{2B}\\&+\gamma^2L\sum_{\ell=k-(k\bmod n)}^{k-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^n\nabla f_i\left(x^{\ell-\delta_i^\ell}\right)\right\|^2\right].\end{aligned}$$

证明。 一些证明技术改编自以下研究成果： Maranjyan 等人 [1].

（2025d）和 Wang 等人 （2025年）。

根据假设 3.3.2 和引理 3.3.4，我们知道 $f$ 是 $L$–光滑的。因此，以下标准不等式成立（Nesterov，2018年）

$$\mathbb{E}\left[f(x^{k+1})\right] \leq \mathbb{E}\left[f(x^k) - \gamma\left\langle \nabla f(x^k), \bar{g}^k\right\rangle + \frac{L\gamma^2}{2}\left\|\bar{g}^k\right\|^2\right]. \tag{B.2}$$

回忆一下，梯度估计量定义为

$$\bar{g}^k = \frac{1}{n}\sum_{i=1}^{n}\bar{g}_i^k = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{b_i^k}\sum_{j=1}^{b_i^k} g_i^{k,j}\ .$$

令 $\mathcal{F}^k$ 表示包含截至当前阶段 2 开始时所有随机性的 σ-代数，即截至迭代 $k-$（$k$ 模 $n$）。一个关键观察是，当前梯度表中的所有梯度都是在当前轮次期间计算并获得的。由于这些梯度是在当前轮次内先前迭代的点上计算的，因此对所有 $i \in [n]$，我们有 $k - \delta_i^k \leq k -$（$k$ 模 $n$）。在给定 $\mathcal{F}^k$ 的条件下，点 $x^{k-\delta_i^k}$ 是确定的。因此，我们可以计算梯度估计量的条件期望：

$$\mathbb{E}\left[\bar{g}^k \mid \mathcal{F}^k\right] = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{b_i^k}\sum_{j=1}^{b_i^k}\mathbb{E}\left[g_i^{k,j} \mid \mathcal{F}^k\right] = \frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k}\right).$$

最后一个等号来自随机梯度估计量的无偏性（假设 3.3.1）。

利用这个条件期望和全期望公式，我们可以 n

现在简化式（B.2）中的内积项：

$$\begin{aligned}
\mathbb{E}\left[\left\langle \nabla f\left(x^k\right), \bar{g}^k\right\rangle\right] &= \mathbb{E}\left[\left\langle \nabla f\left(x^k\right), \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\rangle\right] \\
&\quad + \mathbb{E}\left[\left\langle \nabla f\left(x^k\right), \bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\rangle\right] \\
&= \mathbb{E}\left[\left\langle \nabla f\left(x^k\right), \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\rangle\right] \\
&\quad + \mathbb{E}\left[\left\langle \nabla f\left(x^k\right) - \nabla f\left(x^{k-(k \bmod n)}\right), \bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\rangle\right] \\
&\quad + \mathbb{E}\left[\left\langle \nabla f\left(x^{k-(k \bmod n)}\right), \bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\rangle\right] \\
&= \underbrace{\mathbb{E}\left[\left\langle \nabla f\left(x^k\right), \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\rangle\right]}_{T_1} \\
&\quad + \underbrace{\mathbb{E}\left[\left\langle \nabla f\left(x^k\right) - \nabla f\left(x^{k-(k \bmod n)}\right), \bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\rangle\right]}_{T_2}.
\end{aligned}$$

接下来，根据假设3.3.2，我们有

$$\begin{aligned}
2T_1 &= \mathbb{E}\left[2\left\langle \nabla f\left(x^k\right), \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\rangle\right] \\
&= \mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2 + \left\|\frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] \\
&\quad - \mathbb{E}\left[\left\|\nabla f\left(x^k\right) - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] \\
&\geq \mathbb{E}\left[\left\|\nabla f\left(x^k\right)\right\|^2\right] + \mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] \\
&\quad - \frac{L^2}{n}\sum_{i=1}^n \mathbb{E}\left[\left\|x^k - x^{k-\delta_i^k}\right\|^2\right].
\end{aligned}$$

接下来，我们分析 $T_2$

$$
\begin{aligned}
T_2 &= \mathbb{E}\left[\left\langle \nabla f\left(x^k\right) - \nabla f\left(x^{k-(k \bmod n)}\right), \bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\rangle\right] \\
&\geq -\mathbb{E}\left[\left\|\nabla f\left(x^k\right) - \nabla f\left(x^{k-(k \bmod n)}\right)\right\|\left\|\bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|\right] \\
&\geq -L\mathbb{E}\left[\left\|x^k - x^{k-(k \bmod n)}\right\|\left\|\bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|\right] \\
&= -L\mathbb{E}\left[\left\|\gamma \sum_{\ell=k-(k \bmod n)}^{k-1} \bar{g}^\ell\right\|\left\|\bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|\right] \\
&\geq -L\gamma \sum_{\ell=k-(k \bmod n)}^{k-1} \mathbb{E}\left[\left\|\bar{g}^\ell\right\|\left\|\bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|\right] \\
&\geq -L\gamma \sum_{\ell=k-(k \bmod n)}^{k-1} \frac{1}{2}\left(\mathbb{E}\left[\left\|\bar{g}^\ell\right\|^2\right] + \mathbb{E}\left[\left\|\bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right]\right) \\
&\geq -\frac{L\gamma}{2} \sum_{\ell=k-(k \bmod n)}^{k-1} \mathbb{E}\left[\left\|\bar{g}^\ell\right\|^2\right] - (k \bmod n)\frac{L\gamma\sigma^2}{2B^k n} \\
&\geq -\frac{L\gamma}{2} \sum_{\ell=k-(k \bmod n)}^{k-1} \mathbb{E}\left[\left\|\bar{g}^\ell\right\|^2\right] - \frac{L\gamma\sigma^2}{2B^k} \ .
\end{aligned}
$$

这些不等式由柯西-施瓦茨不等式、 $L$的－光滑性、 $f$、三角不等式、杨氏不等式、引理 B.2.1，最后还有（$k$ 模 $n$） $\leq n-1<n$。

剩下要界定项 $\mathbb{E}\left[\left\|\bar{g}^k\right\|^2\right]$。利用杨氏不等式，可得

$$
\begin{aligned}
\mathbb{E}\left[\left\|\bar{g}^k\right\|^2\right] &\leq 2\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] + 2\mathbb{E}\left[\left\|\bar{g}^k - \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] \\
&\leq 2\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^n \nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] + \frac{2\sigma^2}{B^k n} \ ,
\end{aligned}
$$

其中最后一步用到了引理B.2.1。

现在，将式（B.2）中的所有项合并，得到

$$\begin{aligned}
\mathbb{E}\left[f\left(x^{k+1}\right)\right] &\leq \mathbb{E}\left[f\left(x^{k}\right)\right] - \frac{\gamma}{2}\mathbb{E}\left[\left\|\nabla f\left(x^{k}\right)\right\|^2\right] - \frac{\gamma}{2}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] \\
&\quad + \frac{\gamma L^2}{2n}\sum_{i=1}^{n}\mathbb{E}\left[\left\|x^k - x^{k-\delta_i^k}\right\|^2\right] \\
&\quad + \frac{\gamma^2 L}{2}\sum_{\ell=k-(k \bmod n)}^{k-1}\mathbb{E}\left[\left\|\bar{g}^{\ell}\right\|^2\right] + \frac{\gamma^2 L\sigma^2}{2B^k} + \frac{\gamma^2 L}{2}\mathbb{E}\left[\left\|\bar{g}^{k}\right\|^2\right] \\
&\leq \mathbb{E}\left[f\left(x^{k}\right)\right] - \frac{\gamma}{2}\mathbb{E}\left[\left\|\nabla f\left(x^{k}\right)\right\|^2\right] - \frac{\gamma}{2}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] \\
&\quad + \frac{\gamma L^2}{2n}\sum_{i=1}^{n}\mathbb{E}\left[\left\|x^k - x^{k-\delta_i^k}\right\|^2\right] \\
&\quad + \gamma^2 L\sum_{\ell=k-(k \bmod n)}^{k-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{\ell-\delta_i^{\ell}}\right)\right\|^2\right] \\
&\quad + \gamma^2 L\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(x^{k-\delta_i^k}\right)\right\|^2\right] \\
&\quad + \gamma^2 L\sum_{\ell=k-(k \bmod n)}^{k-1}\frac{\sigma^2}{B^{\ell}n} + \frac{\gamma^2 L\sigma^2}{2B^k} + \frac{\gamma^2 L\sigma^2}{B^k n}\,.
\end{aligned}$$

这就完成了在步长条件 $\gamma \leq 1/4L$ and $B := \inf_{k\geq 0} B^k$ 下的证明。

□

### B.2.4 引理 3.6.2

下面的引理给出了延迟导致的残差误差上界。

**引理3.6.2** (残差估计)。在假设3.3.1下，Ringleader ASGD (算法3.5.1）的迭代序列在步长 $\gamma \leq 1/4nL$ 下满足以下界

$$\frac{1}{K}\sum_{k=0}^{K-1}\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\left\|x^k - x^{k-\delta_i^k}\right\|^2\right] \leq \frac{2\gamma n}{LK}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^{n}\nabla f_j\left(x^{k-\delta_j^k}\right)\right\|^2\right] + \frac{2\gamma\sigma^2}{LB}\,.$$

证明。根据杨氏不等式，我们有

$$\begin{aligned}\mathbb{E}\left[\left\|x^k - x^{k-\delta_i^k}\right\|^2\right] &= \mathbb{E}\left[\left\|\gamma \sum_{\ell=k-\delta_i^k}^{k-1} \bar{g}^\ell\right\|^2\right] \\ &\leq 2\gamma^2 \mathbb{E}\left[\left\|\sum_{\ell=k-\delta_i^k}^{k-1} \frac{1}{n}\sum_{j=1}^n \nabla f_j\left(x^{\ell-\delta_j^\ell}\right)\right\|^2\right] \\ &\quad + 2\gamma^2 \mathbb{E}\left[\left\|\sum_{\ell=k-\delta_i^k}^{k-1} \left(\bar{g}^\ell - \frac{1}{n}\sum_{j=1}^n \nabla f_j\left(x^{\ell-\delta_j^\ell}\right)\right)\right\|^2\right] \\ &\leq 2\gamma^2 \underbrace{\delta_i^k \sum_{\ell=k-\delta_i^k}^{k-1} \mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^n \nabla f_j\left(x^{\ell-\delta_j^\ell}\right)\right\|^2\right]}_{T_{ik}} + 2\gamma^2 (\delta_i^k)^2 \frac{\sigma^2}{Bn}\,.\end{aligned}$$

在最后一个不等式中，我们使用了詹森不等式和引理 B.2.1。接下来，我们估计 $T_{ik}$

$$\begin{aligned}\sum_{k=0}^{K-1}\frac{1}{n}\sum_{i=1}^n T_{ik} &= \sum_{k=0}^{K-1}\frac{1}{n}\sum_{i=1}^n \delta_i^k \sum_{\ell=k-\delta_i^k}^{k-1} \mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^n \nabla f_j\left(x^{\ell-\delta_j^\ell}\right)\right\|^2\right] \\ &= \frac{1}{n}\sum_{i=1}^n \sum_{k=0}^{K-1} \delta_i^k \sum_{\ell=k-\delta_i^k}^{k-1} \mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^n \nabla f_j\left(x^{\ell-\delta_j^\ell}\right)\right\|^2\right] \\ &\leq 2\sum_{i=1}^n \sum_{k=0}^{K-1} \sum_{\ell=k-\delta_i^k}^{k-1} \mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^n \nabla f_j\left(x^{\ell-\delta_j^\ell}\right)\right\|^2\right] \\ &\leq 2\sum_{i=1}^n \delta_i^{\max} \sum_{k=0}^{K-1} \mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^n \nabla f_j\left(x^{k-\delta_j^k}\right)\right\|^2\right] \\ &\leq 4n^2 \sum_{k=0}^{K-1} \mathbb{E}\left[\left\|\frac{1}{n}\sum_{j=1}^n \nabla f_j\left(x^{k-\delta_j^k}\right)\right\|^2\right].\end{aligned}$$

在第一个和最后一个不等式中，我们使用了引理 $\delta_i^{\max} \leq 2n$ 中的界 3.5.1。最后，应用步长条件 $\gamma \leq 1/4nL$ 即可得到结果。 □

## B.3 IA²随机梯度下降的时间复杂度

IA²随机梯度下降（Wang 等人，2025年）的迭代复杂度为

$$K = \mathcal{O}\left(\frac{\delta^{\max} L\Delta}{\varepsilon}\left(1 + \frac{\sigma^2}{n\varepsilon}\right)\right).$$

现在，我们分析 固定计算模型 （3.2）下相应的实际运行时间。由于算法会在单个工作节点

完成一次计算后执行更新，因此我们寻求满足以下条件的最短时间 $T$

$$\sum_{i=1}^{n} \left\lfloor \frac{T}{\tau_i} \right\rfloor \geq K \ .$$

注意到

$$\sum_{i=1}^{n} \frac{T}{\tau_i} \geq \sum_{i=1}^{n} \left\lfloor \frac{T}{\tau_i} \right\rfloor .$$

因此，若我们定义 $T'$ 为

$$\sum_{i=1}^{n} \frac{T'}{\tau_i} = K \ ,$$

then

$$T' = \left( \sum_{i=1}^{n} \frac{1}{\tau_i} \right)^{-1} K \ .$$

由此可知，最短时间 $T$ 必然大于 $T'$。

接下来需要对 $\delta^{\max}$ 给出上界。在初始化时，所有工作节点同时开始计算各自的第一个梯度。当最慢的工作节点完成其第一个梯度时（时间为 $\tau_n$），其他工作节点可能已经各自完成了多个梯度。特别地，

$$\delta^{\max} \geq \sum_{i=1}^{n} \left\lfloor \frac{\tau_n}{\tau_i} \right\rfloor .$$

结合迭代复杂度界，可得总运行时间满足

$$T \geq c \times \frac{\tau_n L \Delta}{\varepsilon} \left( 1 + \frac{\sigma^2}{n\varepsilon} \right),$$

其中 $c > 0$ 是某个通用常数。

请注意，上述表达式不应被视为运行时间的精确上界。更恰当地说，它是对 $T$ 的简化估计，足以满足我们的目的，并为比较提供更清晰的依据。

## B.4 改进的 MaleniaSGD

MaleniaSGD 的迭代复杂度如下（Tyurin & Richt´arik，2024年）

$$K \geq \frac{12\Delta L_f}{\varepsilon} + \frac{12\Delta L_f \sigma^2}{\varepsilon^2 n S} \ ,$$

其中 $S$ 是批次大小调和平均数的下界，即

$$\left( \frac{1}{n} \sum_{i=1}^{n} \frac{1}{b_i^k} \right)^{-1} \geq S \ ,$$

对于所有迭代 $k$。在原始的 MaleniaSGD 分析（Tyurin & Richt´arik，2024年）中，该界由（3.3）中的条件推出，该条件在所有迭代中固定了相同的 $S$。

然而，在固定时间区间（3.2）中，这一条件不再是必要的。

采用与 RingleaderASGD （算法 3.5.1）相同的策略——即等待每个工作节点至少返回一个梯度——我们实际上将速率中的 $S$ 替换为$B$ 。由此得到以下时间复杂度

$$\tau_n K = \frac{12\tau_n \Delta L_f}{\varepsilon}\left(1 + \frac{\sigma^2}{\varepsilon n B}\right).$$

代入 $B$ 引理 3.6.4 中的表达式，并按照定理 3.6.5的证明中的步骤进行推导，我们得到与之前相同的总体时间复杂度——这次 无需 满足条件（3.3），该条件要求预先知道 $\sigma$并预先固定 $\varepsilon$。

最后需要指出的是，这种改进仅在固定时间区间内有效。在计算时间任意变化的情况下，由于与第 RingleaderASGD 节中讨论的B.1相同的原因，无法进行同样的优化。

# 附录 C

# 第 4 章节的附录

## C.1 补充实验

目标函数是一个凸二次函数 $f:\ \mathbb{R}^d \to \mathbb{R}$ 定义为

$$f(x) = \frac{1}{2}x^\top Ax - b^\top x\ ,$$

其中

$$A = \frac{1}{4}\begin{bmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{d\times d}\ , \quad \text{and} \quad b = \frac{1}{4}\begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^d\ .$$

我们将 $f^*$ 记为函数 $f$ 的最小值。 $n$ 个工作节点中的每一个都能够计算无偏随机梯度 $g(x)$，并满足

$$\mathbb{E}\left[\|g(x) - \nabla f(x)\|^2\right] \le 0.01^2\ .$$

这是通过向 $f$ 的梯度添加高斯噪声实现的。

实验使用 Python 实现。通过在配备 Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz 的机器上模拟分布式环境。

### C.1.1 线性噪声

在本节中，我们将工作节点 $i$ 的计算时间建模为如下分布

$$\nu_i = 29i + \mathrm{Exp}(29i), \quad \text{for all} \quad i \in [n]\ .$$

该分布的期望值为 $\mu_i = 2\cdot 29i$ 。此外，Orlicz 范数满足上界 $\alpha_i \le 2\mu_i$。

我们再次设定 $B = 23$ 并运行与第 4.7 节类似的模拟。结果如图 C.1.1 所示。

与前一幅图 4.7.1 相比，重要区别在于此处 ATA-Empirical的表现优于 ATA。这是因为 Orlicz 范数 $\alpha = 4\cdot 29n$ 大得多。

同样，我们在表C.1中提供了数值比较。

### C.1.2 异质时间分布

到目前为止，我们只考虑了客户端遵循相同分布但均值不同的情况。在本节中，我们将实验扩展到以下情况：

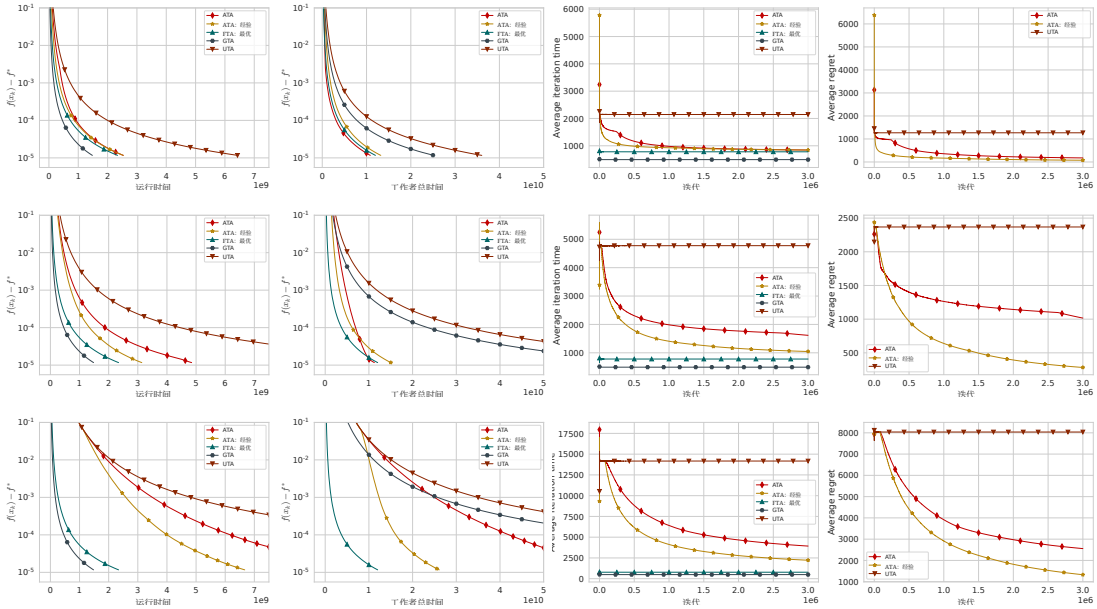



图 C.1.1：每一行的工作节点数量从 17 开始，每次增加至原来的 3 倍，即 $n = 17,\ 51,\ 153,\ 459$，从上到下依次排列。第一列展示运行时间与次优性之间的关系。第二列也绘制了次优性，但横轴为工作者总时间，即 $\sum_{i=1}^{n} T_i^k$在算法 4.4.1中的定义。第三列展示平均迭代时间，即 $C_k/k$ 在所有迭代 $k$上的平均值。最后一列展示平均累积遗憾，其定义见（4.9）。

分布本身存在差异。我们考虑五种分布：指数分布、均匀分布、半正态分布、对数正态分布和伽马分布。我们将遵循这五种分布的五个工作节点分为一组，使每组的均值相同，然后让不同组的均值各不相同。具体来说，我们采用：

- $\mathrm{Exp}(c(5g+1))$,
- 均匀分布 $\left(\frac{c}{(5g+1)2}, \frac{3c}{(5g+1)}\right)$,
- $\left|\mathcal{N}\left(0, c(5g+1)\sqrt{\frac{\pi}{2}}\right)\right|$,
- 对数正态分布 $\left(\log(c(5g+1))/2,\ \sqrt{\log(c(5g+1))}\right)$,
- 伽马分布 $\left((c(5g+1))^2,\ \frac{c}{(5g+1)}\right)$ 形状参数和尺度参数。

接下来，我们将常数 $c(5g+1)$ 加到所有分布中，其中 $c = 29$，而 $g$表示组编号，从 0 开始。客户端被分为 $n/5$组。

实验结果如图 C.1.2所示。这些图表表明，算法在不同分布下都具有稳健性。

## C.1.3 遗憾

在本节中，我们通过模拟验证关于遗憾的定理 4.6.1和 4.6.3。我们设置 $n = 20$ 和 $B = 5$，工作节点 $i$ 的计算时间遵循

表 C.1：达到 $f(x) - f^* < 10^{-5}$ 所需的工作者总时间和运行时间之比。对于工作者总时间，我们将GTA 的工作者总时间除以所列其他算法对应的工作者总时间。对于运行时间则反之，将其他算法的运行时间除以GTA的运行时间，因为 GTA 速度最快。

| $n$ | Total worker time ratio | | | Runtime ratio | | |
|---|---|---|---|---|---|---|
| | ATA | ATA-Empirical | OFTA | ATA | ATA-Empirical | OFTA |
| 17 | 2.32 | 1.91 | 2.1 | 1.71 | 1.71 | 1.58 |
| 51 | 6.71 | 5.02 | 6.29 | 3.27 | 2.12 | 1.58 |
| 153 | 3.41 | 8.68 | 18.87 | 7.96 | 4.5 | 1.58 |

分布

$$\nu_i = \text{Exp}(2i), \quad \text{for all} \quad i \in [n] \ .$$

我们进行了五次模拟，图中包含标准差误差棒，但这些误差棒不可见。结果如图 C.1.3 所示。

正如预期，遗憾呈对数增长。

### C.1.4 真实数据集

在本节中，我们展示一个实验：在 CIFAR-100 数据集（Krizhevsky 等人，2009年）上训练卷积神经网络（卷积神经网络）。该网络由三个卷积层和两个全连接层组成，共有 160k 个参数。

我们使用 Adam 优化器（Kingma 和 Ba，2014年），步长为恒定的 $8 \cdot 10^{-5}$。工作节点的计算时间设置与图 C.1.1 相同。结果如图 C.1.4 所示。

### C.1.5 先验知识对时间分布的影响 S

在训练多个机器学习模型的真实系统中，可能会有先前运行所得的计算时间估计值。借助这些先验知识，ATA和ATA-Empirical 可以显著加快速度，因为它们减少了探索所花的时间，并能迅速接近OFTA的性能。

为说明这一点，我们改变先前运行的次数 $P$。由于我们的算法独立于底层优化过程运行，因此我们首先只关注多臂老虎机组件，在多个迭代中更新机器的置信度分数。然后，我们将损失曲线应用于多臂老虎机阶段的不同部分，并比较结果随 $P$ 增加的变化。 $P$ 越大，估计就越准确。

优化设置与图 C.1.4中的相同，其中 $B = 23$和 $n = 51$。结果见图 C.1.5。

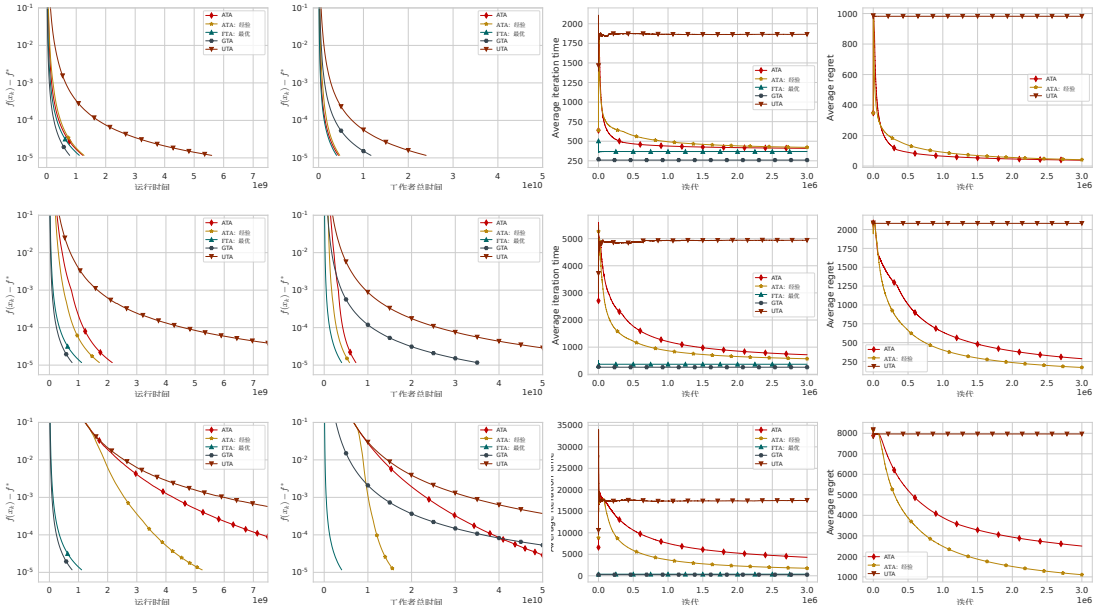


图 C.1.2：每一行对应数量递增的工作节点，从上到下分别为$n = 15$、45、150。我们考虑五种分布——指数分布、均匀分布、半正态分布、对数正态分布和伽马分布——将它们分组，使每组具有相同的均值，然后改变不同组之间的均值。结果表明，这些算法在不同分布下仍保持稳健。各列所表示的内容与图 C.1.1相同。

## C.2 具体优化算法

本节将通过具体示例介绍采用 ATA和 GTA 分配策略的优化算法。

对于优化问题，我们重点关注随机梯度下降 和异步随机梯度下降。其他方法，例如随机近端点方法和高阶方法，也可以采用类似的方式进行构建。

### C.2.1 随机梯度下降

考虑求解以下优化问题的近似驻点：

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \ \{f(x) := \mathbb{E}_{\xi\sim\mathcal{D}}\left[f(x;\xi)\right]\}. \tag{C.1}$$

我们假设每个工作节点都能够计算满足 $\mathbb{E}_{\xi\sim\mathcal{D}}\left[\|f(x;\ \xi)\ -\nabla f(x)\|^2\right] \le \sigma^2$ 的随机梯度 $f(x;\ \xi)$，对所有 $x \in \mathbb{R}^d$ 均成立。

在这种情况下，随机梯度下降 配合分配预算 $B$ 变为 小批量随机梯度下降，批量大小为 $B$。下一步是确定如何收集批次。对于 ATA，我们将此方法称为 SGD-ATA，如算法C.2.1所述。

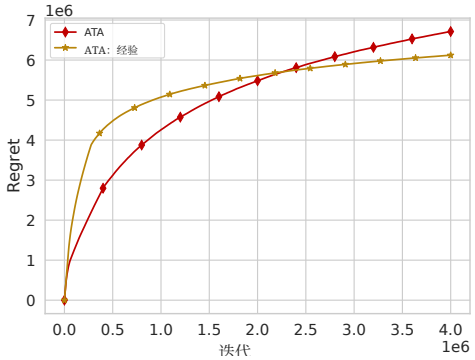



图 C.1.3：迭代过程中的遗憾增长。

**算法 C.2.1** SGD-ATA

1: **优化输入**：初始点 $x^1 \in \mathbb{R}^d$，步长 $\gamma > 0$2: **分配输入**：分配预算 $B$3: **初始化**：对所有 $i \in [n]$，设置经验均值 $\hat{}\mu_i^1 = 0$、使用次数 $K_i^1 = 0$ 和使用时间 $T_i^1 = 0$4: **对于** $k = 1, \ldots\ldots, K$ **执行**5: 为所有 $i \in [n]$ 计算下置信界（$s_i^k$）6: 确定分配：

$$a^k \in \arg\min_{a \in \mathcal{A}} \ell(a, s^k)$$

7: 为每个工作节点分配 $a_i^k$ 个任务 $i \in [n]$
8: 更新 $x$：

$$x^{k+1} = x^k - \frac{\gamma}{B} \sum_{i=1}^{n} \sum_{j=1}^{a_i^k} \nabla f\left(x^k; \xi_i^j\right)$$

9: 对于所有满足 $a_i^k \neq 0$ 的 $i$，执行更新：

$$K_i^{k+1} = K_i^k + a_i^k$$
$$T_i^{k+1} = T_i^k + \sum_{j=1}^{a_i^k} X_i^{k,j}$$
$$\hat{\mu}_i^{k+1} = \frac{T_i^{k+1}}{K_i^{k+1}}$$

10: **结束循环**

在这种情况下，每个任务都包括使用设备的本地数据计算梯度，这些数据被假定与所有其他设备上的数据具有相同的分布。因此，由哪个设备执行该任务并不重要。随后，该方法对这些梯度取平均，得到一个无偏梯度估计器，并执行一次梯度下降步。

现在，让我们给出使用贪心分配算法 小批量随机梯度下降 的版本——算法 C.2.2。

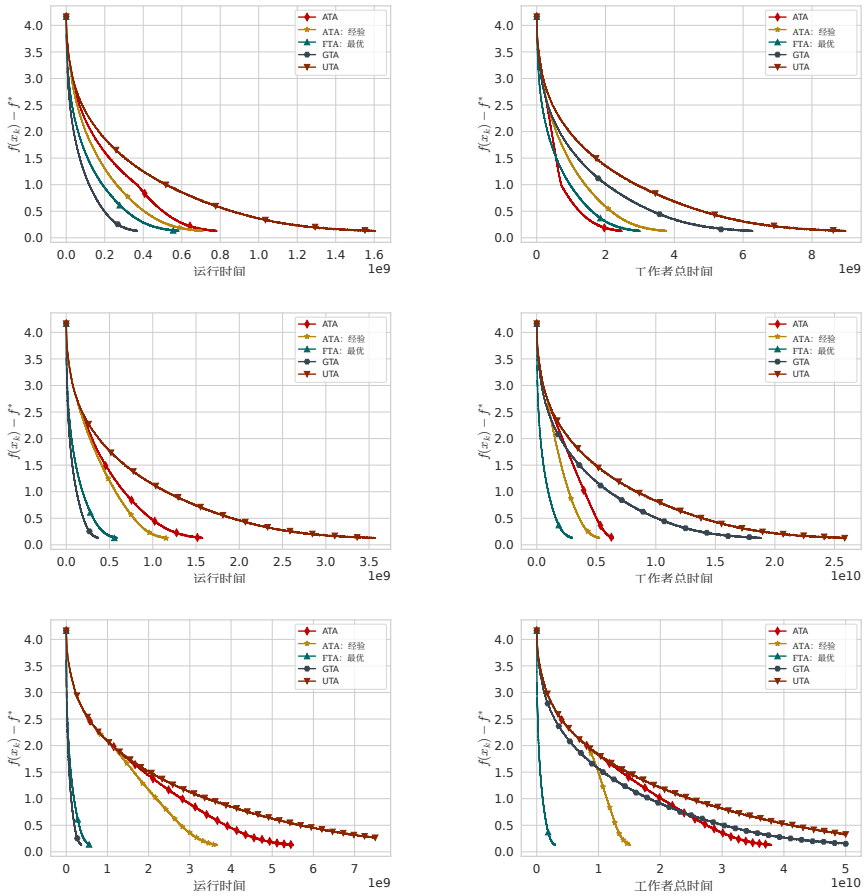



图 C.1.4：我们使用 CIFAR-100 数据集（Krizhevsky 等人，2009年）。模型是一个包含三层卷积层和两层全连接层的卷积神经网络，共有 160k 个参数。使用 Adam 优化器（Kingma 和 Ba，2014年），步长为恒定的 $8 \cdot 10^{-5}$。工作节点的计算时间设置与图 C.1.1相同，其中均值随时间线性增长。批量大小保持为 $B = 23$。每一行对应不同数量的工作节点，从上到下依次为 $n = 17$、51、153。

**算法 C.2.2** SGD-GTA

1：**输入**：初始点 $1 \in \mathbb{R}^d$、步长 $\gamma > 0$、分配预算 $B$2：**对** $k = 1, \ldots\ldots, K$ **执行**3：$b = 0$4：从每个工作节点查询单个梯度 $i \in [n]$5：**当** $b < B$ **执行**

6 $\nabla f\left(x^k\ \xi^{k\,b}\right)$ $i^{k\,b}$：梯度；，来

7 $g^k = g^k + \nabla f\left(x^k\ \xi^{k\,b}\right)$

自工作节点的梯度，：；，8：$b = b + 1$9：在 $x^k$ 处从工作节点 $i^k$ 查询梯度，$b$10：**结束当**11：更新模型：$x^{k+1} = x^k - \gamma \frac{g}{B}$

—

12：**结束循环**

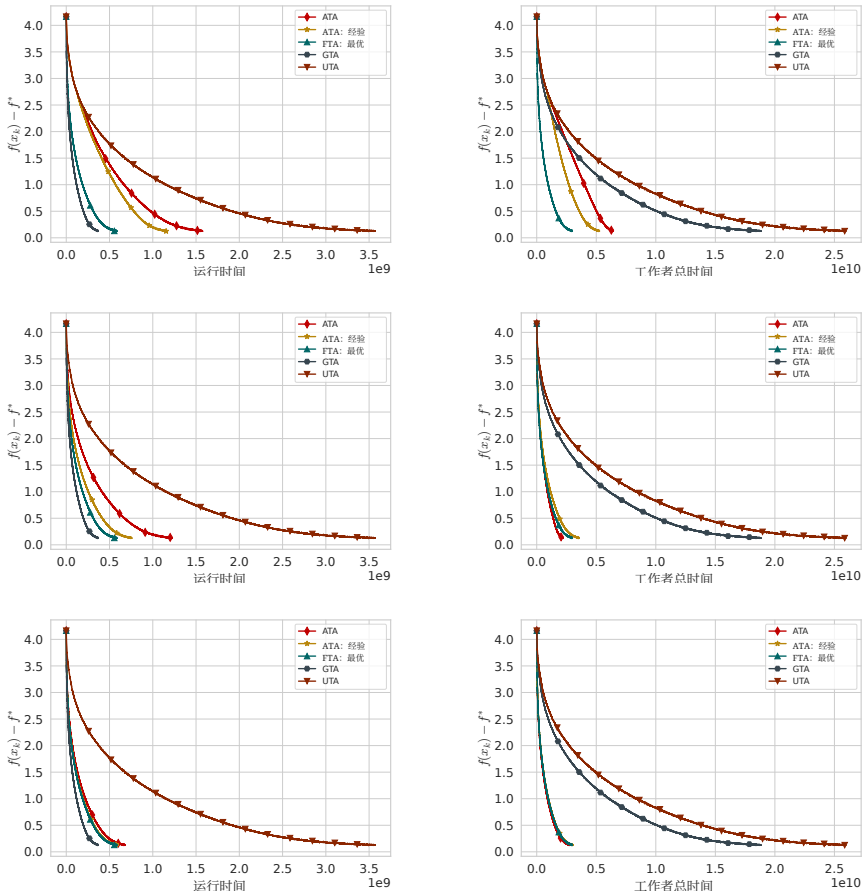



图 C.1.5：我们使用与图 C.1.4相同的优化设置，其中 $B = 23$和 $n = 51$。先前迭代的数量 $P$ 在各行中有所不同，从最上方一行开始依次为 $P = 0$、$5 \cdot 10^5$、$5 \cdot 10^6$。随着先前迭代次数增加，我们观察到 ATA 和 ATA-Empirical的训练速度加快，使其性能更接近 OFTA的最优性能。

算法 C.2.2 与 RennalaSGD 方法完全相同，该方法由Tyurin & Richt´arik (2024年），当目标函数非凸且光滑时，具有最优时间复杂度。

如果计算时间是确定的，那么 GTA 会在每次迭代中进行相同的分配。在这种情况下，SGD-ATA 将收敛到这一固定分配。如果计算时间是随机的，GTA 找到的分配可能在每次迭代中有所不同。在这种情况下，SGD-ATA 将趋近于针对期望时间的最佳分配。

## C.2.2 异步 SGD

这里，我们关注公式（C.1）中给出的齐次问题。贪心变体 RingmasterASGD 由 Maranjyan 等人 （2025d）提出，与RennalaSGD一样，具有最佳运行时间。

现在给出其带有 ATA 的版本，见算法 C.2.3。

**算法 C.2.3** ASGD-ATA

1: 优化输入：初始点 $x^1 \in \mathbb{R}^d$，步长 $\gamma > 0$2: 分配输入：分配预算 $B$3: 初始化：对所有 $i \in [n]$，经验均值 $\hat{}\mu_i^1 = 0$、使用次数 $K_i^1 = 0$ 和使用时间 $T_i^1 = 0$4: 对于 $k = 1, \ldots, K$，执行 5: 计算所有 $i \in [n]$ 的下置信界 $(s_i^k)$6: 求分配： $a^k = \text{RAS}(s^k; B)$ 7: 使用分配 $a^k$，根据算法 C.2.4 更新 $x^k$ 8: 对所有满足 $a_i^k \neq 0$ 的 $i$，进行更新：

$$
\begin{aligned}
K_i^{k+1} &= K_i^k + a_i^k \\
T_i^{k+1} &= T_i^k + \sum_{j=1}^{a_i^k} X_i^{k,j} \\
\hat{\mu}_i^{k+1} &= \frac{T_i^{k+1}}{K_i^{k+1}}
\end{aligned}
$$

9: **结束循环**

**算法 C.2.4** ASGD更新

**1: 输入：初始点 $x^1 \in \mathbb{R}^d$，步长 $\gamma > 0$，分配向量 $a$，其中 $\|a\|_1 = B$2: 工作节点在 $a_i > 0$ 处开始计算随机梯度 $x^1$3: 对于 $s = 1, 2, \ldots, B$，执行 4: 接收来自工作节点 $i^s$的梯度 $\nabla f\left(x^{s+\delta^s}; \xi_{i^s}^{s+\delta^s}\right)$ 5: 更新： $x^{s+1} = x^s - \gamma \nabla f\left(x^{s+\delta^s}; \xi_{i^s}^{s+\delta^s}\right)$6: 如果 $a_{i^s} > 0$，则 7: 工作节点 $i^s$ 开始计算 $\nabla f\left(x^{s+1}; \xi_{i^s}^{s+1}\right)$8: 将工作节点 $i^s$ 的剩余分配量减一：$a_{i^s} = a_{i^s} - 1$9: 结束 if 10: 结束循环 11: 返回： $x^{B+1}$**

序列 $\{\delta^s\}$ 表示延迟，其中 $\delta^s \geq 0$ 是工作节点 $i^s$ 开始计算梯度的迭代与应用该梯度的迭代 $s$之间的差值。

这里，任务仍然是梯度计算，但每个工作节点后续任务使用不同的点来计算梯度。这些点取决于实际计算时间和该方法的异步特性，因此得名 异步随机梯度下降。

## C.3 递归分配选择算法

本节将介绍一种用于寻找最佳分配的高效方法。给定下置信界 $s^k$ 和分配预算 $B$，ATA （算法 4.4.1）的每次迭代

通过求解来确定分配

$$a^k \in \arg\min_{a\in\mathcal{A}} \ \ell(a, s^k) \ ,$$

其中

$$\ell(a,\mu) \coloneqq \max_{i\in[n]} \ a_i\mu_i = \|a \odot \mu\|_\infty \ ,$$

其中 $\odot$ 表示逐元素乘积。根据上下文明确时，我们用$\ell(a)$代替 $\ell(a,\ \mu)$。

在早期迭代中，当某些 $s_i$ 值为 0 时，ATA 会在这些臂之间均匀分配，直到所有 $s_i$ 值都变为正数。此后，分配由算法 C.3.1中的递归例程确定。

**算法 C.3.1** 递归分配选择（RAS）

1: **输入：分数** $s_1$，……，$s_n$，分配预算 $B$2: 不失一般性，假设 $s_1 \le s_2 \le \cdots \le s_n$（即对分数排序）3: **如果** $B = 1$ **则**4: **返回：**（1，0，……，0）5: **结束如果**6: 求出先前的最佳分配：

$$a = (a_1, \ldots, a_n) = \mathsf{RAS}\,(s_1, \ldots, s_n; B-1)$$

7: 确定第一个零分配：

$$r = \begin{cases} \min\{i \mid a_i = 0\}, & \text{if } \ a_n = 0, \\ n, & \text{otherwise} \end{cases} \tag{C.2}$$

8: 找到下一个最佳查询分配集：

$$M = \arg\min_{i\in[r]} \ \|(a + e_i) \odot s\|_\infty \, ,$$

其中 $e_i$ 是方向为 $i$的单位向量。

9: 选择 $j \in M$，使得

$$\arg\max_{i\in[r]} \ (a_i + e_{j,i})s_i$$

的基数最小

10: **返回：** $a + e_j$

**注释 C.3.1**。 RAS 的迭代复杂度为

$$\mathcal{O}\left(n \ln\left(\min\{B, n\}\right) + \min\{B, n\}^2\right) \ .$$

事实上，第一个 $n\ln(\min\{B, n\})$ 项源于找出最小的 $B$分数。对于第二项，注意在 (C.2)中，我们有 $r \le \min\{B, n\}$。

## C.3.1 最优性

现在我们证明，RAS 能够找到最优分配，正如下述引理所述。

**引理 C.3.2.** 对于正分数 $0 < s_1 \leq s_2 \leq \ldots \leq s_n$，RAS (算法 C.3.1) 找到一个最优分配 $h \in \mathcal{A}$，满足

$$h \in \arg\min_{a \in \mathcal{A}} \|a \odot s\|_\infty \ .$$

证明。 我们通过对分配预算 $B$ 使用归纳法来证明这一结论。

**基础情形 ($B = 1$)：** 当 $B = 1$，RAS (算法 C.3.1) 将任务分配给分数最低的工作节点（第 9 行）。因此，基础情形成立。

**归纳步骤：** 假设 RAS 在预算$B - 1$下找到一个最优分配，记为

$$\bar{h} = \mathsf{RAS}(s_1, \ldots, s_n; B - 1) \ .$$

我们需要证明，对于预算 $B$ 返回的解（记为 $h = h + e_r$）也是最优的。

反设存在 $a \in \mathcal{A}$ 使得 $a \neq h$ 且$\ell(a) < \ell(h)$。令 $a = \bar{a} + e_q$ 为某个 $q \in [n]$。注意到 $\|\bar{a}\|_1 = B - 1$，因为 $a \in \mathcal{A}$。

根据 $\ell\left(\bar{h} + e_r\right)$的值，我们分两种情况讨论：

- $\ell\left(\bar{h} + e_r\right) = h_k s_k$为某个 $k \neq r$。在这种情况下，将索引 $r$ 增加一个单位不会改变最大值，即 $\ell\left(\bar{h}\right) = \ell\left(\bar{h} + e_r\right)$。根据归纳假设，$h$ 使预算为 $B - 1$时的 $\ell(\boldsymbol{x})$ 最小。因此，我们有

$$\ell(a) \geq \ell\left(\bar{a}\right) \geq \ell\left(\bar{h}\right) = \ell\left(\bar{h} + e_r\right) = \ell(h) \ ,$$

这与假设 $\ell(a) < \ell(h)$ 矛盾。

- $\ell\left(\bar{h} + e_r\right) = \left(\bar{h}_r + 1\right) s_r$。根据算法的逻辑，$\left(\bar{h}_r + 1\right) s_r \leq \left(\bar{h}_i + 1\right) s_i$对所有 $i \neq r$。由于 $\ell(h + e_r) \leq \ell(h + e_q)$，并且我们假设 $\ell(\bar{a} + e_q) = \ell(a) < \ell(h) = \ell(h + e_r)$，那么 $\bar{a} \neq h$否则 $\ell(\bar{a} + e_q) < \ell(\bar{a} + e_r)$。鉴于这一点 $\|h\|_1 = \|\bar{a}\|_1$，这意味着存在某个 $u \in [n]$ 使得 $0 \leq \bar{a}_u \leq h_u - 1$，以及另一个索引 $v \in [n]$，其中 $\bar{a}_v \geq h_v + 1$。

此外，注意到 $r$ 的选择使得 $\ell\left(\bar{h} + e_r\right)$ 最小。利用 $\ell\left(\bar{h} + e_r\right) = \left(\bar{h}_r + 1\right) s_r$ 这一事实，对于任意索引 $q$，我们也必然有 $\ell\left(\bar{h} + e_q\right) = \left(\bar{h}_q + 1\right) s_q$。由此可推得

$$\ell(h) = \ell\left(\bar{h} + e_r\right) \leq \ell\left(\bar{h} + e_v\right) = \left(\bar{h}_v + 1\right) s_v \leq \max_i \ \bar{a}_i s_i = \ell\left(\bar{a}\right) \leq \ell(a) \ ,$$

其中，在第二个不等式中我们利用了 $\bar{a}_v \geq h_v + 1$ 这一事实；在最后一个不等式中，我们利用了这样一个事实：向向量中添加一个元素时，损失不会减小。这一连串不等式再次与假设 $\ell(a) < \ell(h)$ 矛盾。

由于这两种情况都会导致矛盾，我们得出结论：不存在满足$\ell(a) < \ell(h)$的 $a \in \mathcal{A}$。因此，RAS 为预算 $B$产生最优分配。 □

### C.3.2 最小基数

在所有可能的分配中，RAS 选择一个始终使该集合的基数最小的分配：

$$\arg\max_{i\in[n]} \ a_i s_i \ .$$

选择这一方式的原因纯粹是技术性的，因为这样可以使引理C.4.1 成立。

**引理 C.3.3.** RAS 的输出保证该集合具有最小基数：

$$\arg\max_{i\in[n]} \ a_i s_i$$

在所有最优分配 $a$ 中。

证明。 此证明采用了与前一个证明类似的推理。

设 $h = \mathsf{RAS}(s; B)$，并将分配 $a$ 的集合 $\arg\max^{i\in[n]} \ a_i s_i$ 的基数记为

$$C_B(a) = \left| \arg\max_{i\in[n]} \ a_i s_i \right| \geq 1 \ .$$

我们通过对 $B$进行归纳法来证明这一结论。

**基础情形**（$B = 1$）：</style}对于 $B = 1$，只有一种坐标分配，因此$C_1(h) = 1$，其基数是可能的最小值。

**归纳步骤：**假设 RAS 对预算$B - 1$ 求得的最优分配具有最小基数，并将其输出记为

$$\bar{h} = \mathsf{RAS}(s_1, \ldots, s_n; B - 1) \ .$$

¯我们需要证明，在预算为 $B$的所有最优分配中， $h = h + e_r$使 $C_B(a)$达到最小值。

反设存在 $a \in \mathcal{A}$，使得 $a \neq h$、 $\ell(a) = \ell(h)$，且 $C_B(a) < C_B(h)$。令 $a = \bar{a} + e_q$，其中 $q \in [n]$。我们分三种情况讨论：

- $C_B(h) = 1$。由于最小基数恰为 1，因此必有$C_B(a) \geq 1 = C_B(h)$，这与我们的假设矛盾。
- $C_B(h) = C_{B-1}\left(\bar{h}\right) > 1$。当 $\ell(h) = \ell\left(\bar{h}\right) \neq \left(\bar{h}_r + 1\right) s_r$ 时，就会出现这种情况。根据 $h$ 的最优性，我们有 $\ell\left(\bar{h}\right) \leq \ell(\bar{a}) \leq \ell(a) = \ell(h) = \ell(\bar{h})$，这意味着 $\ell(\bar{a}) = \ell(a)$。因此， $C_{B-1}(\bar{a}) \leq C_B(a)$。由于归纳假设对 $B - 1$ 成立，我们有 $C_{B-1}\left(\bar{h}\right) \leq C_{B-1}(\bar{a})$。因此，

$$C_B(h) = C_{B-1}\left(\bar{h}\right) \leq C_{B-1}\left(\bar{a}\right) \leq C_B(a) \ ,$$

  这就导致了矛盾。
- $C_B(h) = C_{B-1}\left(\bar{h}\right) + 1$。当 $\ell(h) = \ell\left(\bar{h}\right) = \left(\bar{h}_r + 1\right) s_r$ 时会出现这种情况。按照前一种情况的处理方式，我们有 $\ell(\bar{a}) = \ell(a)$，因此 $C_{B-1}(\bar{a}) \leq C_B(a)$。由于对 $B-1$ 成立归纳假设，我们知道 $C_{B-1}\left(\bar{h}\right) \leq C_{B-1}(\bar{a})$。

现在还有其他情况：

– If $C_{B-1}(\bar{a}) = C_{B-1}\left(\bar{h}\right) + 1$, then

$$C_B(h) = C_{B-1}\left(\bar{h}\right) + 1 = C_{B-1}(\bar{a}) \leq C_B(a) \ ,$$

这就导致了矛盾。

– 现在假设 $C_{B-1}(\bar{a}) = C_{B-1}\left(\bar{h}\right)$。我们将证明，在这种情况下，$C_B(a) = C_{B-1}\,\bar{a} + 1$。反证法：假设 $C_B(a) = C_{B-1}\,\bar{a}$，这意味着 $(\bar{a}_q + 1)s_q < \ell(a)$。令 $k$ 为满足 $\bar{a}_k s_k = \ell(a)$ 的索引。构造一个新的分配 $a' = \bar{a} + e_q - e_k$。于是，

$$C_{B-1}(a') = C_{B-1}(\bar{a}) - 1 < C_{B-1}\left(\bar{h}\right) \ ,$$

这与归纳假设矛盾。因此，

$$C_B(a) = C_{B-1}(\bar{a}) + 1 \ .$$

利用这一点，我们得到

$$C_B(h) = C_{B-1}\left(\bar{h}\right) + 1 = C_{B-1}(\bar{a}) + 1 = C_B(a) \ ,$$

这再次与 $C_B(a) < C_B(h)$ 矛盾。

证明至此完成。 □

## C.4 定理4.6.1、定理4.6.3以及定理4.4.2的证明

我们首先回顾相关记号。对于 $i \in [n]$ 和 $k \in [K]$，$(X_{i\,k}^{(u)}\ ,\ )^{u\in[B]}$ 表示在轮次 $k$ 从分布 $\nu_i$ 中获得的 $B$ 独立样本。使用分配向量 $a^k \in \mathcal{A}$ 时，工作节点 $i$ 在轮次 $k$ 的总计算时间为 $\sum_{u=1}^{a_i^k} X_{i\,k}^{(u)}\ ,$，其中 $a_i^k > 0$。$\mu = (\mu_1, \ldots\ldots, \mu_K)$ 是均值向量。对于每个 $k \in [K]$，使用分配向量 $a^k$ 时，我们回顾代理损失 $\ell$ 的定义：$\mathcal{A} \times \mathbb{R}_+^n \to \mathbb{R}_+$，即

$$\ell(a^k, \lambda) = \max_{i\in[n]} a_i^k \lambda_i \ ,$$

其中 $\lambda = (\lambda_1, \ldots\ldots, \lambda_n)$ 是一个由非负分量组成的向量。当 $\lambda = \mu$ 时，我们省略对 $\ell$ 第二个输入的依赖。对于每个 $\lambda$，令 $\bar{a}_\lambda \in \mathcal{A}$ 为使该损失最小的动作

$$\bar{a}_\lambda \in \arg\min_{a\in\mathcal{A}} \ \ell(a, \lambda) \ .$$

为简化符号，我们省略 $\bar{a}_\mu$ 对 $\mu$ 的依赖。第 $k$ 轮实际的（随机）计算时间记为 $C:\ \mathcal{A} \to \mathbb{R}_+$：

$$C(a^k) := \max_{i\in[n]} \sum_{u=1}^{a_i^k} X_i^{k,u} \ . \tag{4.1}$$

令 $a^*$ 为使期望时间最小的动作

$$a^* \in \arg\min_{a\in\mathcal{A}} \ \mathbb{E}\left[C(a)\right] \ .$$

$K$ 轮次后的期望遗憾定义如下

$$\mathcal{R}_K := \sum_{t=1}^{K} \mathbb{E}\left[\ell(a^k) - \ell(\bar{a})\right] \ .$$

在本分析的其余部分，我们考虑使用RAS 过程找到的 $\bar{a} \in \mathsf{argmin}_{a\in\mathcal{A}} \ \ell(a)$。对于每个 $i \in [n]$，回忆一下， $k_i$ 是满足以下条件的最小整数

$$(\bar{a}_i + k_i)\mu_i > \ell(\bar{a}) \ . \tag{C.3}$$

下面给出一个用于定理4.6.1 和4.6.3的技术引理。

**引理 C.4.1**。设 $x = (x_1, \ldots, x_n) \in \mathbb{R}^n_{\geq 0}$。设 $a$ 为 RAS($x$; $B$) 的输出。对于每个 $i$, $j \in [n]$，我们有

$$a_j x_j \leq (a_i + 1)\, x_i \ .$$

证明。固定 $x \in \mathbb{R}^n_+$，并令 $a = \mathsf{RAS}(x; B)$。当 $\min_{i\in[n]} x_i = 0$ 时，结果显然成立。

假设对所有 $i \in [n]$， $x_i > 0$ 成立。令 $s \geq 1$ 表示其基数

$$s := \left| \arg\max_{i\in[n]} \ a_i x_i \right| \ .$$

固定 $i$， $j \in [n]$，令 $k \in \arg\max^{i\in[n]} \ a_i x_i$。我们需要证明

$$a_k x^k \leq (a_i + 1) x_i \ .$$

我们使用反证法。假设我们有 $a_k x^k > (a_i+1)x_i$，考虑由 $a'_k = a_k - 1$、 $a'_i = a_i + 1$ 以及当 $u \notin \{i, k\}$ 时的 $a'_u = \bar{a}_u$ 给出的分配向量 $a' \in \mathcal{A}$。令 $R := \max_{u\neq i,\ k}\{a_u x_u\}$。我们有

$$\ell(a', x) = \max_{u\in[n]} \ a'_u x_u = \max\{(a_i + 1)x_i, (a_k - 1)x^k, R\} \ .$$

我们考虑两种情况：

- 假设 $s = 1$ （即， $[n]$ 中唯一满足 $a_u x_u = \ell(a,\ x)$ 的元素是 $k$），那么必然有 $R < a_k x^k$。此外，根据矛盾假设，$(a_i + 1)x_i < a_k x^k$。因此，

$$\ell(a', x) = \max\{(a_i + 1)x_i, (a_k - 1)x^k, R\} < a_k x^k = \ell(a, x) \ ,$$

这与 $a$的定义矛盾。

- 假设 $s \geq 2$，由于根据假设 $a_k x^k > (a_i + 1)x_i$，显然我们有$a_i x_i < \ell(a, x)$，因此集合 $[n] \setminus \{k, i\}$ 中恰有 $s - 1$

个元素满足 $a_u x_u = \ell(a, x)$。特别地，这意味着

$$\ell(a', x) = \max_{u \in [n]} \{(a_i + 1)x_i, (a_k - 1)x^k, R\} = R = \ell(a, x) \ .$$

因此， $a' \in \arg\min_{a \in \mathcal{A}} \ell(a, x)$以及满足$a'_i x_i = \ell(a', x) = \ell(a, x)$的元素个数至多为 $s - 1$，这与 $s$在给定RAS 选择和引理 C.3.3的情况下是最小的这一事实相矛盾。

作为结论，我们得到 $a_k x^k \leq (a_i + 1)x_i$。 □

**备注 C.4.2**。 回顾一下，引理*C.4.1* 保证 $k_i$ 在(C.3)中定义的量满足： $k_i \in \{1, 2\}$对每个 $i \in [n]$均成立。

## C.4.1 定理4.6.1的证明

下面我们重述该定理。

**定理 4.6.1**。 假设假设条件 4.3.1 成立。令 $\bar{a} \in \arg\min_{a \in \mathcal{A}} \ell(a)$，若存在多个最优动作，则选取 RAS 以 $\mu$ 为输入时输出的动作。则 ATA 在输入 $(B, \alpha)$ 时的期望遗憾满足

$$\mathcal{R}_K \leq 2n \max_{i \in [n]} \{B\mu_i - \ell(\bar{a})\} + c \cdot \sum_{i=1}^{n} \frac{\alpha^2(\bar{a}_i + k_i)(B\mu_i - \ell(\bar{a}))}{((\bar{a}_i + k_i)\mu_i - \ell(\bar{a}))^2} \cdot \ln K \ ,$$

其中 $\alpha = \max_{i \in [n]} \|X_i - \mu_i\|_{\psi_1}$， 且 $c$ 是一个常数。

证明。 令 $K_i^k$表示在第$k$ 轮之前查询臂 $i$ 的轮次数（我们取 $K_i^1 = 0$）。回想一下，我们选择了以下置信界：若$K_i^k \geq 1$，则

$$\mathrm{conf}(i, k) = 2\alpha \sqrt{\frac{\ln(2k^2)}{K_i^k}} + 2\alpha \frac{\ln(2k^2)}{K_i^k} \ ,$$

否则，$\mathrm{conf}(i, k) = \infty$ 。回想一下，$\hat{}\mu_i^k$ 表示在 $k$ 之前观测到的来自 $\nu_i$ 的样本的经验均值，其中若 $K_i^k \geq 0$ 则取$\hat{}\mu_i^k = 0$ ，若 $K_i^k = 0$则取相应值。令 $s_i^k$表示算法中使用的下置信界：

$$s_i^k = \left(\hat{\mu}_i^k - \mathrm{conf}(i, k)\right)_+ \ .$$

我们引入事件 $\mathcal{E}_i^k$（对应于 $i \in [n]$ ）以及由 $k \in [K]$ 定义的事件：

$$\mathcal{E}_i^k := \left\{\left|\hat{\mu}_i^k - \mu_i\right| > \mathrm{conf}(i, k)\right\} .$$

Let

$$\mathcal{E}_k = \bigcup_{i \in [n]} \mathcal{E}_i^k \ .$$

我们来证明，对于每个 $k \in [K]$ 和 $i \in [n]$： $\mathbb{P}\left(\mathcal{E}_i^k\right) \leq 1/k^2$，利用并集界可得 $\mathbb{P}(\mathcal{E}_k) \leq n/k^2$。令 $i \in [n]$，利用引理 C.5.1 并取 $\delta = 1/k^2$，可得

$$\mathbb{P}(\mathcal{E}_i^k) = \mathbb{P}\left\{\left|\hat{\mu}_i^k - \mu\right| > \mathrm{conf}(i, k)\right\} \leq \frac{1}{k^2} \ .$$

我们将满足 $\ell(a^k) > \ell(\bar{a})$ 的轮次 $k$ 称为“坏轮次”。下面给出坏轮次数量的上界。

注意到，在一个坏轮次中，必然存在一个臂 $i \in [K]$ 使得 $a_i^k \mu_i > \ell(\bar{a})$。对于每个 $i \in [n]$，令 $N_i(k)$ 表示满足 $q \in \{1, \ldots, k\}$且 $a_i^q \mu_i > \ell(\bar{a})$ 和 $i \in \arg\max_{j \in [n]} a_j^q \mu_j$ 的轮次数量（这对应于由工作节点 $i$ 触发的坏轮次）

$$N_i(k) := \left| \{ q \in \{1, \ldots, k\} : a_i^q \mu_i > \ell(\bar{a}) \text{ and } a_i^q \mu_i = \ell(a_q) \} \right| .$$

我们证明，在 $\ell(a^k) > \ell(\bar{a})$ 的情况下，以下事件将成立：存在 $i \in [n]$ 使得

$$E_{i,k} := \mathcal{E}_k \text{ or } \left\{ N_i(k-1) \leq \frac{24\alpha^2(\bar{a}_i + k_i)\ln(2K^2)}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2} \right\} .$$

为证明这一点，我们使用反证法。假设对于每个 $i \in [n]$，$\neg E_{i,\,k}$ 都成立，并且 $\ell(a^k) > \ell(\bar{a})$。这意味着 $k$ 是一个坏轮次，令 $i$ 为触发该坏轮次的臂（即 $i \in \arg\max_{j \in [n]} a_j^k \mu_j$）。事件 $\neg E_{i,\,k}$ 特别给出

$$N_i(k-1) > \frac{24\alpha^2(\bar{a}_i + k_i)\ln(2K^2)}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2} . \tag{C.4}$$

注意，在每个 $N_i(\cdot)$ 增加的轮次中，从分布 $\nu_i$ 中收到的样本数至少增加 $\bar{a}_i + k_i$。因此，我们有 (C.4) 蕴含

$$K_i^k > \frac{24\alpha^2(\bar{a}_i + k_i)^2 \ln(2K^2)}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2} = \frac{24\alpha^2 \ln(2K^2)}{\left(\mu_i - \frac{\ell(\bar{a})}{\bar{a}_i + k_i}\right)^2} .$$

然后，利用 $\mathrm{conf}(\cdot)$ 的表达式和上述界，我们得到

$$\begin{aligned} 2\mathrm{conf}(i,k) &= 4\alpha\sqrt{\frac{\ln(2k^2)}{K_i^k}} + 4\alpha\frac{\ln(2k^2)}{K_i^k} \\ &\leq \mu_i - \frac{\ell(\bar{a})}{\bar{a}_i + k_i} . \end{aligned} \tag{C.5}$$

矛盾假设表明$a_i^k \mu_i > \ell(\bar{a})$，于是，利用 $k_i$ 的定义，我们得到$a_i^k \geq \bar{a}_i + k_i$。因此，(C.5) 给出

$$2\mathrm{conf}(i,k) < \mu_i - \frac{\ell(\bar{a})}{a_i^k} . \tag{C.6}$$

注意到，在每一轮次中，$\|a^k\|_1 = B$，因此，如果我们有 $a_i^k \geq \bar{a}_i + k_i > \bar{a}_i$ 对于某个 $i$，那么必然存在 $j \in [n] \setminus \{i\}$ 使得 $a_j^k \leq \bar{a}_j - 1$。利用 $\ell(\bar{a}) \geq \bar{a}_j \mu_j$ 以及（C.6），我们得到

$$a_i^k(\mu_i - 2\mathrm{conf}(i,k)) > \bar{a}_j \mu_j . \tag{C.7}$$

由于 $\neg\mathcal{E}_i^k$和 $\neg\mathcal{E}_{j,\ k}$ 均成立（因为 $\neg E_{i,\ k}$ 蕴含 $\neg\mathcal{E}_k$），我们有

$$\mu_i - 2\mathrm{conf}(i,k) \le \hat{\mu}_{i,k} - \mathrm{conf}(i,k) \le s_i^k \ , \tag{C.8}$$

且 $\mu_j \geq \hat{}\mu_{j,k} - \mathrm{conf}(j,k)$。回想一下，$\mu_j \geq 0$，因此

$$\mu_j \geq (\hat{\mu}_{j,k} - \mathrm{conf}(j,k))_+ = s_{j,k} \ . \tag{C.9}$$

利用 (C.8) 和 (C.9) 中的界限，我们得到 (C.7)

$$a_i^k s_i^k > \bar{a}_j s_{j,k} \geq (a_j^k + 1) s_{j,k} \ ,$$

其中在第二个不等式中用到了 $j$ 的定义。这与引理 C.4.1 的陈述矛盾，从而完成反证法。因此，$k$ 是坏轮次这一事件意味着至少存在一个 $i \in [n]$ 使得 $E_{i,\ k}$ 成立。我们称一个坏轮次是由臂 $i$ 触发的，即在该轮次中 $N_i(\cdot)$ 增加。注意，若 $k \in [K]$ 不是坏轮次，则 $\mathbb{E}\left[\ell(a^k)\right] - \ell(\bar{a}) = 0$；否则，若 $k$ 是由 $i \in [n]$ 触发的坏轮次，则

$$\mathbb{E}\left[\ell(a^k)\right] - \ell(\bar{a}) \le B\mu_i - \ell(\bar{a}) \ .$$

为简化记号，对于 $i \in [n]$，我们引入

$$H_i := \frac{24\alpha^2(\bar{a}_i + k_i)\ln(2K^2)}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2} \ .$$

我们用$\mathbf{1}(\cdot)$表示示性函数：当花括号内的条件为真时取值为1，否则取值为0。

期望遗憾满足

$$
\begin{aligned}
\mathcal{R}_K &= \sum_{i=1}^{K} \mathbb{E}\left[\ell(a^k) - \ell(\bar{a})\right] \\
&\le \sum_{i=1}^{n} (B\mu_i - \ell(\bar{a}))\mathbb{E}[N_i(K)] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} (B\mu_i - \ell(\bar{a}))\mathbb{E}\left[\mathbb{1}(k \text{ is a bad round triggered by } i)\right] \\
&\le \max_{i\in[n]}\{(B\mu_i - \ell(\bar{a}))\} \cdot \sum_{t=1}^{K} \mathbb{P}(\mathcal{E}_k) \\
&\quad + \sum_{i=1}^{n} (B\mu_i - \ell(\bar{a})) \sum_{k=1}^{K} \mathbb{E}\left[\mathbb{1}(k \text{ is a bad round triggered by } i) \mid \neg\mathcal{E}_k\right] \\
&\le \max_{i\in[n]}\{(B\mu_i - \ell(\bar{a}))\} \cdot \sum_{t=1}^{K} \mathbb{P}(\mathcal{E}_k) \\
&\quad + \sum_{i=1}^{n} (B\mu_i - \ell(\bar{a})) \sum_{k=1}^{K} \mathbb{E}\left[\mathbb{1}(N_i(k) = 1 + N_i(k-1) \text{ and } N_i \le H_i) \mid \neg\mathcal{E}_k\right] \\
&\le \max_{i\in[n]}\{(B\mu_i - \ell(\bar{a}))\} \cdot \sum_{k=1}^{K} \mathbb{P}(\mathcal{E}_k) + \sum_{i=1}^{n} (B\mu_i - \ell(\bar{a}))H_i \\
&\le 2n \max_{i\in[n]}\{(B\mu_i - \ell(\bar{a}))\} + \sum_{i=1}^{n} \frac{24\alpha^2(\bar{a}_i + k_i)(B\mu_i - \ell(\bar{a}))\ln(2K^2)}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2} \,.
\end{aligned}
$$

□

## C.4.2 定理4.6.3 的证明

下面重述为ATA-Empirical给出遗憾界的定理，并随后给出其证明。

**定理 4.6.3。** 假设假设条件 4.3.1 成立。令 $\bar{a} \in \arg\min_{a\in\mathcal{A}} \ell(a)$；若存在多个最优动作，则考虑RAS 接收 $\mu$时输出的动作。则ATA-Empirical 使用输入 $(B, \eta)$ 所得经验置信界的期望遗憾满足

$$
\mathcal{R}_K \le 2n \max_{i\in[n]}\{B\mu_i - \ell(\bar{a})\} + c \cdot \sum_{i=1}^{n} \frac{\eta^2\mu_i^2(\bar{a}_i + k_i)(B\mu_i - \ell(\bar{a}))}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2} \cdot \ln K \,,
$$

其中 $\eta = \max_{i\in[n]} \alpha_i/\mu_i$，且 $c$ 是一个常数。

证明。 我们采用定理4.6.1证明中使用的技巧。回顾 $\eta$ 的表达式：

$$
\eta = \max_{i\in[n]} \frac{\alpha_i}{\mu_i} \,.
$$

定义以下量 $C_{i,\ k}$：

$$C_{i,k} = 2\sqrt{\frac{\ln(2k^2)}{K_i^k}} + 2\frac{\ln(2k^2)}{K_i^k} \ .$$

回顾一下，这里使用的下置信界定义如下

$$\hat{s}_i^k = \hat{\mu}_i^k \left(1 - \eta C_{i,k}\right)_+ \ .$$

我们还定义以下量

$$\hat{u}_{i,k} := \hat{\mu}_i^k \left(1 + \frac{4}{3}\eta C_{i,k}\right) \ .$$

我们引入事件 $\mathcal{E}_i^k$（针对 $i \in [n]$ 和 $k \in [K]$），其定义如下

$$\mathcal{E}_i^k := \left\{\mu_i < \hat{s}_i^k\right\} \quad \text{or} \quad \left\{\eta C_{i,k} \le \frac{1}{4} \quad \text{and} \quad \mu_i > \hat{u}_{i,k}\right\} \ .$$

Let

$$\mathcal{E}_k = \bigcup_{i\in[n]} \mathcal{E}_i^k \ .$$

根据引理 C.5.3，对于每个 $k \in [K]$ 和 $i \in [n]$: $\mathbb{P}\left(\mathcal{E}_i^k\right) \le 1/k^2$，利用并集界可得 $\mathbb{P}(\mathcal{E}_k) \le n/k^2$。

仿照定理 4.6.1 的证明，我们将满足 $k$ 且 $\ell(a^k) > \ell(\bar{a})$ 的轮次称为“坏轮次”。下面给出坏轮次数量的上界。

注意，在一个坏轮次中，必然存在一个臂 $i \in [K]$ 使得$a_i^k\mu_i > \ell(\bar{a})$。对于每个 $i \in [n]$，令 $N_i(k)$ 表示满足 $q \in\{1,\dots,k\}$ 且 $a_i^q\mu_i > \ell(\bar{a})$ 的轮次数，其中 $i \in \arg\max_{j\in[n]}\{a_j^q\mu_j\}$（这对应于由工作节点 $q$ 触发的坏轮次）：

$$N_i(k) := \left|\{q \in \{1,\dots,k\} : a_i^q\mu_i > \ell(\bar{a}) \text{ and } a_i^q\mu_i = \ell(a_q)\}\right| \ .$$

我们证明，在 $\ell(a^k) > \ell(\bar{a})$ 的情况下，以下事件将成立：存在 $i \in [n]$，使得

$$E_{i,k} := \mathcal{E}_k \text{ or } \left\{N_i(k-1) \le \frac{185\eta^2\mu_i^2(\bar{a}_i + k_i)\ln(2K^2)}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2}\right\} \ .$$

为证明这一点，假设反证法中存在某个 $i \in[n]$: $\neg E_{i,\ k}$，且 $k$ 是由臂 $i$ 触发的坏轮次（即 $\ell(a^k) > \ell(\bar{a})$ 且$i \in \arg\max_{j\in[n]} a_j^k\mu_j$）。

由此特别得到

$$N_i(k-1) > \frac{185\eta^2\mu_i^2(\bar{a}_i + k_i)\ln(2K^2)}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2} \ . \tag{C.10}$$

请注意，每当 $N_i(\cdot)$ 增加时，从分布 $\nu_i$ 中收到的样本数量至少增加 $\bar{a}_i + k_i$。因此，我们有

(C.10) 蕴含

$$K_i^k > \frac{185\eta^2\mu_i^2(\bar{a}_i+k_i)^2\ln(2K^2)}{\left((\bar{a}_i+k_i)\mu_i-\ell(\bar{a})\right)^2} = \frac{185\eta^2\mu_i^2\ln(2K^2)}{\left(\mu_i-\frac{\ell(\bar{a})}{\bar{a}_i+k_i}\right)^2} .$$

因此，利用 $C_{i,k}$ 的表达式以及上述界

$$\begin{aligned} C_{i,k} &= 2\sqrt{\frac{\ln(2k^2)}{K_i^k}} + 2\frac{\ln(2k^2)}{K_i^k} \\ &\leq \frac{3}{19\eta\mu_i}\left(\mu_i - \frac{\ell(\bar{a})}{\bar{a}_i+k_i}\right) . \end{aligned} \tag{C.11}$$

上述最后一个界尤其意味着 $\eta C_{i,k} \leq 3/19$，因此 $(1-\eta C_{i,k})_+ = 1-\eta C_{i,k}$。

我们有

$$\begin{aligned} \hat{\mu}_i^k - \hat{s}_i^k &= \hat{\mu}_i^k\left(1-(1-\eta C_{i,k})_+\right) \\ &\leq \eta C_{i,k}\hat{\mu}_i^k \\ &\leq \eta C_{i,k}\frac{\mu_i}{1-\eta C_{i,k}} \\ &\leq \frac{1}{5}\left(\mu_i - \frac{\ell(\bar{a})}{\bar{a}_i+k_i}\right) , \end{aligned} \tag{C.12}$$

其中，在倒数第二个不等式中我们使用了事件 $\neg\mathcal{E}_i^k$（尤其是 $\hat{s}_i^k = \hat{\mu}_i^k(1-\eta C_{i,k})_+ \leq \mu_i$），并在最后一个不等式中使用了界（C.11）。

根据假设 $\ell(a^k) > \ell(\bar{a})$ 且 $\ell(a^k) = a_i^k\mu_i$，我们必然有 $a_i^k \geq \bar{a}_i + k_i$。因此，界（C.12）

$$5\hat{\mu}_i^k - 5\hat{s}_i^k < \mu_i - \frac{\ell(\bar{a})}{a_i^k} .$$

注意，在每一轮次 $\|a^k\|_1 = B$，因此，如果对于某个 $i$，我们有 $a_i^k \geq \bar{a}_i + k_i > \bar{a}_i$，那么必然存在 $j \in [n]\setminus\{i\}$ 使得 $a_j^k \leq \bar{a}_j - 1$。因此，利用 $\ell(\bar{a}) \geq \bar{a}_j\mu_j$ 这一事实，我们得到

$$a_i^k(\mu_i + 5\hat{s}_i^k - 5\hat{\mu}_i^k) > \ell(\bar{a}) \geq \bar{a}_j\mu_j . \tag{C.13}$$

由于 $\neg\mathcal{E}_i^k$ 和 $\neg\mathcal{E}_{j,\,k}$ 成立（因为 $\neg E_{i,\,k}$ 蕴含 $\neg\mathcal{E}_k$），我们得到

$$\begin{aligned} \mu_i + 5\hat{s}_i^k - 5\hat{\mu}_i^k &= \hat{s}_i^k + \mu_i - \hat{\mu}_i^k + 4\left(\hat{s}_i^k - \hat{\mu}_i^k\right) \\ &= \hat{s}_i^k + \mu_i - \hat{\mu}_i^k + 4\hat{\mu}_i^k\left((1-\eta C_{i,k})_+ - 1\right) \\ &\leq \hat{s}_i^k + \mu_i - \hat{\mu}_i^k - 4\hat{\mu}_i^k\eta C_{i,k} \\ &\leq \hat{s}_i^k + \mu_i - \hat{u}_{i,k} , \end{aligned}$$

总之，注意到给定 $\eta C_{i,k} \leq 3/19$，事件 $\neg\mathcal{E}_i^k$ 意味着 $\mu_i \leq \hat{u}_{i,k}$，因此

$$\mu_i + 5\hat{s}_i^k - 5\hat{\mu}_i^k \leq \hat{s}_i^k .$$

由于$\neg\mathcal{E}_{j,k}$成立，我们也有

$$\mu_j \geq \hat{s}_{j,k} \ .$$

利用 (C.13) 中最后两个界，我们得到

$$a_i^k \hat{s}_i^k > \bar{a}_j \hat{s}_{j,k} \geq (a_j^k + 1)\hat{s}_{j,k} \ ,$$

在第二个不等式中，我们使用了 $j$ 的定义，即满足 $\bar{a}_j \geq 1 + a_j^k$ 的臂。这与引理C.4.1的结论矛盾，从而完成反证法。因此， $k$ 是坏轮次这一事件意味着 $E_{i,k}$ 对至少一个 $i \in [n]$ 成立。我们称坏轮次由臂 $i$ 触发；在该轮次中， $N_i(\cdot)$ 被递增。注意，如果 $k \in [K]$ 不是坏轮次，那么 $\mathbb{E}\left[\ell(a^k)\right] - \ell(\bar{a}) = 0$；否则，如果 $k$ 是由 $i \in [n]$ 触发的坏轮次，那么 $\mathbb{E}\left[\ell(a^k)\right] - \ell(\bar{a}) \leq B\mu_i - \ell(\bar{a})$。为简化记号，我们对 $i \in [n]$ 引入

$$H_i := \frac{185\eta^2\mu_i^2(\bar{a}_i + k_i)\ln(2K^2)}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2} \ .$$

期望遗憾满足

$$\begin{aligned}
\mathcal{R}_K &= \sum_{i=1}^{K} \mathbb{E}\left[\ell(a^k) - \ell(\bar{a})\right] \\
&\leq \sum_{i=1}^{n} (B\mu_i - \ell(\bar{a}))\mathbb{E}[N_i(K)] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} (B\mu_i - \ell(\bar{a}))\mathbb{E}\left[\mathbb{1}(k \text{ is a bad round triggered by } i)\right] \\
&\leq \max_{i\in[n]}\{(B\mu_i - \ell(\bar{a}))\} \cdot \sum_{t=1}^{K} \mathbb{P}(\mathcal{E}_k) \\
&\quad + \sum_{i=1}^{n} (B\mu_i - \ell(\bar{a})) \sum_{k=1}^{K} \mathbb{E}\left[\mathbb{1}(k \text{ is a bad round triggered by } i) \mid \neg\mathcal{E}_k\right] \\
&\leq \max_{i\in[n]}\{(B\mu_i - \ell(\bar{a}))\} \cdot \sum_{t=1}^{K} \mathbb{P}(\mathcal{E}_k) \\
&\quad + \sum_{i=1}^{n} (B\mu_i - \ell(\bar{a})) \sum_{k=1}^{K} \mathbb{E}\left[\mathbb{1}(N_i(k) = 1 + N_i(k-1) \text{ and } N_i \leq H_i) \mid \neg\mathcal{E}_k\right] \\
&\leq \max_{i\in[n]}\{(B\mu_i - \ell(\bar{a}))\} \cdot \sum_{k=1}^{K} \mathbb{P}(\mathcal{E}_k) + \sum_{i=1}^{n} (B\mu_i - \ell(\bar{a}))H_i \\
&\leq 2n \max_{i\in[n]}\{(B\mu_i - \ell(\bar{a}))\} + \sum_{i=1}^{n} \frac{185\eta^2\mu_i^2(\bar{a}_i + k_i)(B\mu_i - \ell(\bar{a}))\ln(2K^2)}{\left((\bar{a}_i + k_i)\mu_i - \ell(\bar{a})\right)^2} \ .
\end{aligned}$$

□

### C.4.3 定理4.4.2的证明

我们先重述该定理。

**定理 4.4.2**。假设 4.3.1 成立，且令 $\eta := \max_{i\in[n]} \alpha_i/\mu_i$，其中 $\alpha_i = \|X_i - \mu_i\|_{\psi_1}$。那么，使用 ATA 根据输入 $(B, \alpha)$ 规定的分配，经过 $K$轮次后的总期望计算时间满足

$$\mathcal{C}_K \leq (1 + 4\eta \ln(B))\, \mathcal{C}_K^* + \mathcal{O}(\ln K)\ .$$

证明。令 $\mathbb{E}_k$ 表示关于截至并包括 $k$ 时所观测变量的期望，并令 $\mathcal{F}_k$ 表示相应的滤过。根据塔式法则，我们有

$$\sum_{k=1}^{K} \mathbb{E}\left[C(a^k)\right] = \mathbb{E}\left[\sum_{k=1}^{K} \mathbb{E}_{k-1}\left[C(a^k)\right]\right].$$

考虑第 $k \in [K]$轮次，我们用 $\mathbb{E}_{k-1}[C(a_t)]$ 来给 $\mathbb{E}_{k-1}[\ell(a^k)]$一个上界。回忆 $a^k \in \mathcal{F}_{k-1}$，令 $Y_i = \sum_{u=1}^{a_i^k} X_{i\,k}^{(u)}$，，由于 $Y_i$ 是 $a_i^k$ 个独立同分布样本之和，因此我们有 $\mathbb{E}_{k-1}[Y_i] = a_i^k\mu_i$ 以及 $\|Y_i - a_i^k\mu_i\|_{\psi_1} \leq a_i^k\|X_i - \mu_i\|_{\psi_1}$。于是，根据引理C.5.4，得到

$$\begin{aligned}
\mathbb{E}_{k-1}\left[C(a^k)\right] &= \mathbb{E}_{k-1}\left[\max_{i\in\mathrm{supp}(a^k)} \sum_{u=1}^{a_i^k} X_{i,k}^{(u)}\right] \\
&\leq \max_{i\in\mathrm{supp}(a^k)} a_i^k\mu_i + 4 \max_{i\in\mathrm{supp}(a^k)} a_i^k\alpha_i \cdot \ln(B) \\
&\leq \max_{i\in\mathrm{supp}(a^k)} a_i^k\mu_i + 4 \max_{i\in\mathrm{supp}(a^k)} a_i^k\,\eta\mu_i \cdot \ln(B) \\
&= (1 + 4\eta\ln(B)) \max_{i\in\mathrm{supp}(a^k)} a_i^k\mu_i.
\end{aligned}$$

此外，利用詹森不等式，我们有

$$\max_{i\in[n]} a_i^*\mu_i \leq \mathbb{E}\left[\max_{i\in[n]} \sum_{u=1}^{a_{k,i}} X_{i,k}^{(u)}\right] = \mathbb{E}\left[C(a^*)\right]\ .$$

结合最后两个界与定理 4.6.1 的结果，我们便得到所需结果。 □

## C.5 技术引理

下面的引理给出了次指数变量的集中界。请注意，这一结果可以从 Maurer& Pontil(2021年 的工作中的命题 7 推导得出；不过，直接应用该结果需要假设这些变量为正，而在一维情形下，他们的证明并不需要这一假设。为完整起见，我们在下文给出完整证明。

**引理 C.5.1**。设 $Y_1, \ldots, Y_n$ 为独立同分布随机变量，且满足 $\mathbb{E}[Y_1] = 0$ 和 $\alpha = \|Y_1\|_{\psi_1} < +\infty$。那么对于任意 $\delta \in (0,1)$，以至少.....的概率，我们有

$1-\delta$

$$\left|\frac{1}{n}\sum_{i=1}^{n} Y_i\right| \le 2\alpha\left(\sqrt{\frac{\ln(2/\delta)}{n}} + \frac{\ln(2/\delta)}{2n}\right) .$$

证明。 设 $v := 2n\alpha^2$。根据引理 C.5.5 ，我们有

$$\sum_{i=1}^{n} \mathbb{E}\left[Y_i^2\right] \le 2n\alpha^2 \quad \text{and} \quad \sum_{i=1}^{n} \mathbb{E}\left[(Y_i)_+^q\right] \le \frac{q!}{2} v\alpha^{q-2} .$$

因此，利用伯恩斯坦集中不等式（命题 C.5.2），可得

$$\mathbb{P}\left(\left|\sum_{i=1}^{n} Y_i\right| \ge 2\alpha\sqrt{nt} + \alpha t\right) \le 2\exp(-t) .$$

取 $t = \ln(2/\delta)$，即可得到所需结果。 □

**命题 C.5.2** （Boucheron 等人著作中的定理 2.10 （2005））。设 $X_1, \ldots, X_n$ 为独立实值随机变量。假设存在正数 $v$ 和 $c$ 使得

$$\sum_{i=1}^{n} \mathbb{E}\big[X_i^2\big] \le v \quad \textit{and} \quad \sum_{i=1}^{n} \mathbb{E}\big[(X_i)_+^q\big] \le \frac{q!}{2} v\, c^{q-2} , \qquad \textit{for all integers } q \ge 3 ,$$

其中 $x_+ :=$ 最大值$\{x, 0\}$。定义中心化和

$$S := \sum_{i=1}^{n} \big(X_i - \mathbb{E}X_i\big).$$

那么，对每个 $t > 0$,

$$\mathbb{P}\left(S \ge \sqrt{2vt} + ct\right) \le e^{-t} .$$

**引理 C.5.3**。 设 $X_1, \ldots, X_n$ 为均值为 $\mu$且满足 $\|X_1\|_{\psi_1} < +\infty$的独立同分布的正随机变量。记 $\alpha := \|X_1 - \mu\|_{\psi_1}$。记 $\hat{} X_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ ，并令$\delta \in (0,1)$。定义 $\eta, C_{\cdot,\cdot}$ 如下：

$$\eta := \frac{\alpha}{\mu} \qquad \text{and} \qquad C_{n,\delta} := 2\sqrt{\frac{\ln(2/\delta)}{n}} + 2\frac{\ln(2/\delta)}{n} .$$

则以至少 $1-\delta$ 的概率，有

$$\mu \ge \hat{X}_n(1 - \eta \cdot C_{n,\delta})_+ ,$$

其中我们使用记号 $(a)_+ = \max\{0, a\}$。此外，如果 $\eta\, C_{n,\,\delta} \le 1/4$，则以至少 $1-\delta$ 的概率

$$\hat{X}_n(1 - \eta \cdot C_{n,\delta})_+ \le \mu \le \hat{X}_n\left(1 + \frac{4}{3}\eta \cdot C_{n,\delta}\right) .$$

证明。 固定 $n$， $\delta$。我们考虑事件

$$\mathcal{E}_{n,\delta} = \left\{ \left| \hat{X}_n - \mu \right| \leq \alpha \cdot C_{n,\delta} \right\}$$

以至少 $1-\delta$ 的概率成立；若将引理 C.5.1 应用于 $X_i - \mu$。 **ˆ证明** $\mu \geq X_n(1 - \eta \cdot C_{n,\delta})_+$**：**

若 $\eta C_{n,\delta} \geq 1$，则有 $(1 - \eta \cdot C_{n,\delta})_+ = 0$，结论由 $X$ 非负这一事实得出，从而得到 $\mu \geq 0$。

现在假设 $\eta C_{n,\delta} < 1$。回想一下，事件 $\mathcal{E}_{n,\delta}$ 意味着

$$\hat{X}_n \leq \mu(1 + \eta C_{n,\delta}) \ .$$

因此，我们有

$$\frac{\hat{X}_n}{1 + \eta C_{n,\delta}} \leq \mu \ .$$

结合上述界，利用 $1 - \eta C_{n,\delta} \leq \frac{1}{1+\eta C_{n,\delta}}$，可得

$$\hat{X}_n(1 - \eta C_{n,\delta})_+ = \hat{X}_n(1 - \eta C_{n,\delta}) \leq \frac{\hat{X}_n}{1 + \eta \cdot C_{n,\delta}} \leq \mu \ .$$

**证明** $\mu \leq \left(1 + \frac{4}{3}\eta \cdot C_{n,\delta}\right)$. 回想一下，事件 $\mathcal{E}_{n,\delta}$ 给出

$$\hat{X}_n \geq \mu - \alpha C_{n,\delta} = \mu(1 - \eta C_{n,\delta}) \ .$$

假设 $\eta C_{n,\delta} \leq 1/4$。因此，我们有

$$\mu \leq \frac{\hat{X}_n}{1 - \eta C_{n,\delta}} \ .$$

接下来，我们利用以下事实：对于任意 $x \in [0, 1/4]$，都有

$$\frac{1}{1-x} \leq 1 + \frac{4}{3}x \ ,$$

由此可得

$$\mu \leq \hat{X}_n \left(1 + \frac{4}{3}\eta \cdot C_{n,\delta}\right) \ ,$$

当 $\eta C_{n,\delta} \leq 1/4$ 时。 □

以下引理给出了独立次指数随机变量最大值的期望的一个有用上界。

**引理 C.5.4.** 设 $X_1, \ldots, X_n$ 是一列相互独立且 Orlicz 范数有限的随机变量，其中 $\|X_i\|_{\psi_1} < +\infty$，并设 $\mathbb{E}[X_i] = \mu_i$。则有

$$\mathbb{E}\left[\max_{i\in[n]} X_i\right] \leq \max_{i\in[n]} \mu_i + 4\alpha \ln(n) \ ,$$

其中 $\alpha = \text{最大值}_{i\in[n]} \|X_i - \mu_i\|_{\psi_1}$。

证明。若 $n = 1$该界显然成立，则设 $n \geq 2$。令 $Y_i := X_i - \mu_i$，则 $\alpha =$最大值$_{i\in[n]} \|Y_i\|_{\psi_1}$。令 $\lambda \in (0, 1/\alpha)$，我们有

$$\begin{aligned}\max_{i\in[n]} Y_i &= \frac{1}{\lambda} \ln\left(\exp\left(\lambda \max_{i\in[n]} Y_i\right)\right) \\ &\leq \frac{1}{\lambda} \ln\left(\sum_{i=1}^{n} \exp\left(\lambda Y_i\right)\right) .\end{aligned}$$

取期望并应用引理C.5.5，可得

$$\begin{aligned}\mathbb{E}\left[\max_{i\in[n]} Y_i\right] &\leq \frac{1}{\lambda} \ln\left(\sum_{i=1}^{n} \mathbb{E}\left[\exp\left(\lambda Y_i\right)\right]\right) \\ &\leq \frac{1}{\lambda} \ln\left(\frac{n}{1-\lambda\alpha}\right) .\end{aligned}$$

我们选择

$$\lambda = \frac{1 - 1/n}{\alpha} ,$$

由此得到

$$\begin{aligned}\mathbb{E}\left[\max_{i\in[n]} Y_i\right] &\leq \frac{\alpha}{1-\frac{1}{n}} \ln(n^2) \\ &= 2\alpha \frac{n}{n-1} \ln(n) \\ &\leq 4\alpha \ln(n) .\end{aligned} \tag{C.14}$$

令 $i^* \in \arg\max_{i\in[n]} X_i$，可得

$$\begin{aligned}\max_{i\in[n]} X_i - \max_{i\in[n]} \mu_i &= X_{i^*} - \max_{i\in[n]} \mu_i \\ &\leq X_{i^*} - \mu_{i^*} \leq \max_{i\in[n]} (X_i - \mu_i) = \max_{i\in[n]} Y_i .\end{aligned}$$

将最后一个界与（C.14）结合，可得

$$\mathbb{E}\left[\max_{i\in[n]} X_i\right] \leq \max_{i\in[n]} \mu_i + 4\alpha \ln(n) .$$

□

下面的引理基于一个标准论证；为完整起见，我们在此给出该论证 s.

**引理 C.5.5.** 令 $Y$ 为一个满足 $\alpha = \|Y\|_{\psi_1} < +\infty$的变量。则对于任意 $\lambda \in (-1/\alpha, 1/\alpha)$，有

$$\mathbb{E}\left[\exp\left(\lambda Y\right)\right] \leq \frac{1}{1 - |\lambda|\,\alpha} .$$

此外，对于任意 $q \geq 3$，有

$$\mathbb{E}\left[Y^2\right] \leq 2\alpha^2 \quad \text{and} \quad \mathbb{E}\left[(Y)_+^q\right] \leq \frac{q!}{2} \cdot (2\alpha^2) \cdot \alpha^{q-2} \ .$$

证明。令 $Z = |Y|/\alpha$。首先注意到，我们有

$$\sum_{k\geq 0} \frac{\mathbb{E}\left[Z^k\right]}{k!} = \mathbb{E}\left[\exp(Z)\right] = \mathbb{E}\left[\exp\left(\frac{|Y|}{\alpha}\right)\right] \leq 2 \ ,$$

so,

$$\sum_{k\geq 1} \frac{\mathbb{E}\left[Z^k\right]}{k!} \leq 1 \ .$$

这意味着

$$\mathbb{E}\left[|Y|^k\right] \leq k!\alpha^k$$

对所有 $k \geq 1$ 均成立。利用这个界，我们估计矩母函数

$$\begin{aligned}
\mathbb{E}\left[\exp(\lambda Y)\right] &\leq \mathbb{E}\left[\exp\left(|\lambda|\,|Y|\right)\right] \\
&= \sum_{k\geq 0} \frac{|\lambda|^k\, \mathbb{E}\left[|Y|^k\right]}{k!} \\
&\leq 1 + \sum_{k\geq 1} |\lambda|^k\, \alpha^k \\
&= \frac{1}{1 - |\lambda|\,\alpha} \ .
\end{aligned}$$

其余的界可由下式得到

$$\mathbb{E}\left[|Y|^k\right] \leq k!\alpha^k \ .$$

□