编号：2024-2-0001 级别：公开

优化基本理论与方法课程研究报告

Optimal Transport Based Distributed Optimization Research

（2025 年 1 月）

周楠 （3220102535）

王晓宇 （3220104364）

浙江大学计算机科学与技术学院

Contents

Abstract

We analyze Newton's method with lazy Hessian updates for solving general possibly non-convex optimization problems. We propose to reuse a previously seen Hessian for several iterations while computing new gradients at each step of the method. This significantly reduces the overall arithmetic complexity of second-order optimiza- tion schemes. By using the cubic regularization technique, we establish fast global convergence of our method to a second-order stationary point, while the Hessian does not need to be updated each iteration. For convex problems, we justify global and local superlinear rates for lazy Newton steps with quadratic regularization, which is easier to compute. The optimal frequency for updating the Hessian is once every d iterations, where d is the dimension of the problem. This provably improves the total arithmetic complexity of second-order algorithms by a factor $\sqrt{d}$.

我们分析了用 Lazy Hessian 更新牛顿方法来解决一般的可能非凸优化问题。我们建议在方法的每一步计算新梯度的同时，在多次迭代中重复使用之前看到的 Hessian。这大大降低了二阶优化方案的整体算术复杂度。通过使用立方正则化技术，我们建立了我们的方法对二阶静止点的快速全局收敛性，同时不需要每次迭代更新 Hessian。对于凸问题，我们证明了使用二次正则化的 Lazy 牛顿迭代法的全局和局部超线性率，这更容易计算。更新 Hessian 的最佳频率是每 d 次迭代一次，其中 d 是问题的维度。这可以证明，二阶算法的总算术复杂度提高了 $\sqrt{d}$ 倍。

1 论文研究的问题背景

在优化问题中，二阶优化算法（如牛顿法）在处理条件不佳问题时表现出色，尤其是在局部范围内能够达到快速的二次收敛速度。然而，牛顿法的全局收敛性依赖于初始点的选择，且每一步都需要计算梯度和 Hessian 矩阵，并进行矩阵分解，计算成本较高。特别是在大规模问题中，Hessian 矩阵的计算和存储成本非常高，这限制了牛顿法在实际应用中的广泛使用。

本文的核心动机是减少二阶优化算法的计算复杂度，特别是减少 Hessian 矩阵的计算频率。作者提出了一种简单但有效的方法：Lazy Hessian 更新。具体来说，算法在多次迭代中重用先前计算的 Hessian 矩阵，而每一步都使用新的梯度。通过这种方式，可以显著减少 Hessian 矩阵的计算频率，从而降低总体的计算复杂度。

作者指出，Hessian 矩阵的计算成本通常是梯度计算的 d 倍，其中 d 是问题的维度。因此，通过减少 Hessian 矩阵的计算频率，可以显著加速算法的运行速度。本文通过理论分析和实验验证，证明了 Lazy Hessian 更新方法在保持收敛速度的同时，能够显著减少计算成本。

2 论文的贡献

1. 提出了带有立方正则化的 Lazy 牛顿迭代法（第5.2节）：该方法在当前点计算梯度，而在过去的轨迹中使用 Hessian 矩阵。量化了二阶信息不精确性带来的误差，并形式化了单步方法的进展（定理5.1）。展示了如何通过按比例增加正则化参数来平衡 m 次连续 Lazy 牛顿迭代法的误差。

2. 基于此，开发了带有 Lazy Hessian 的立方牛顿法（Algorithm 1），并证明了其快速全局收敛到二阶稳定点（第5.3节的定理5.2）：这避免了方法陷入鞍点的问题。当 $m := 1$（每次迭代都更新 Hessian 矩阵）时，方法恢复为经典的立方牛顿法（Nesterov & Polyak, 2006）。

   在考虑 Hessian 计算的实际算术成本的情况下，方法的最优选择是每 $m := d$ 次迭代更新一次 Hessian 矩阵，从而将立方牛顿法的总复杂度提高了 $\sqrt{d}$ 倍（见推论5.2）。

3. 展示了如何在问题是凸问题时改进方法（第5.4节）：开发了带有 Lazy Hessian 的正则化牛顿法（Algorithm 2），它将模型中的立方正则化替换为二次正则化。这使得子问题更容易求解，仅涉及一次标准的矩阵求逆，同时保持了原始立方正则化方法的快速收敛速度。

4. 研究了新算法的局部收敛性（第5.5节，见定理5.4和定理5.5）：证明了这些算法都具有超线性收敛速度。作为一个特例，还证明了经典牛顿法（无正则化）在 Lazy Hessian 更新下的局部二次收敛性。

5. 提供了数值实验：通过数值实验验证了所提出方法的有效性。

3 论文的章节组织

1. Introduction（引言）: 介绍了二阶优化算法的背景和动机，特别是牛顿法在处理病态问题时的优势及其计算成本高的局限性。提出了 Lazy Hessian 更新的核心思想，并概述了本文的主要贡献。

2. Lazy Newton Steps（Lazy 牛顿迭代法）: 详细介绍了 Lazy 牛顿迭代法的定义和数学模型。通过在当前点计算梯度，而在过去的轨迹中使用 Hessian 矩阵，提出了带有立方正则化的 Lazy 牛顿迭代法。量化了二阶信息不精确性带来的误差，并形式化了单步方法的进展（定理5.1）。

3. Global Convergence Rates（全局收敛性分析）: 基于 Lazy 牛顿迭代法，提出了**带有 Lazy Hessian 的立方牛顿法（Algorithm 2）**，并证明了其快速全局收敛到二阶稳定点（定理5.2）。特别地，当 Hessian 每次迭代都更新时（$m := 1$），该方法恢复为经典的立方牛顿法。通过理论分析，证明了最优的 Hessian 更新频率是每 d 次迭代更新一次（$m := d$），从而将总复杂度提高了 $\sqrt{d}$ 倍（推论5.2）。

4. Minimizing Convex Functions（凸问题的优化）: 针对凸问题，提出了**带有 Lazy Hessian 的正则化牛顿法（Algorithm 2**，使用二次正则化替代立方正则化。这使得子问题更容易求解，仅涉及一次标准的矩阵求逆，同时保持了快速收敛速度。证明了该方法的全局复杂度与立方牛顿法相同，但每次迭代的计算成本更低。

5. Local Superlinear Convergence（局部超线性收敛）: 研究了新算法的局部收敛性，证明了这些算法在局部范围内具有超线性收敛速度（定理5.4和定理5.5）。特别地，对于经典牛顿法（无正则化），在 LazyHessian 更新下也具有局部二次收敛性。

6. Practical Implementation（实际实现）: 讨论了算法的实际实现细节，包括矩阵分解和自适应搜索策略。提出了**自适应立方牛顿法（Algorithm 3）**，通过动态调整正则化参数来进一步提高算法的性能。

7. Experiments（实验）: 通过数值实验验证了所提出方法的有效性。实验包括 Soft Maximum 问题、Logistic 回归问题和对角神经网络训练问题。结果表明，Lazy Hessian 更新方法在保持收敛速度的同时显著减少了计算成本。

8. Discussion（讨论）: 总结了本文的主要贡献，并提出了未来可能的研究方向，包

括研究具有特定 Hessian 结构的问题、探索 LazyHessian 更新与拟牛顿法之间的联系，以及将分析推广到高阶优化方案。

4 论文的相关工作

LazyHessian 更新的思想并非全新，早在 1967 年，Shamanskii 就提出了在非线性方程组求解中使用旧 Hessian 矩阵的局部收敛性分析。Shamanskii 证明了在更新 Hessian 矩阵时，迭代具有局部二次收敛速度，而在不更新 Hessian 矩阵时，迭代具有线性收敛速度。

此后，这一思想被广泛应用于各种正则化方法中，如 Levenberg-Marquardt 正则化、阻尼牛顿迭代法、近端牛顿法等。这些方法通常具有渐近全局收敛性，但没有显式的非渐近收敛速度保证。

本文的工作与这些方法的不同之处在于，作者使用了现代全局化技术（如立方正则化和梯度正则化），并证明了带有 LazyHessian 更新的二阶优化算法在广泛的凸和非凸优化问题中具有快速的全局收敛速度。特别是，本文提出了非渐近的全局复杂度保证，证明了 LazyHessian 更新方法在达到二阶稳定点时的全局复杂度为 $O(1/\epsilon^{3/2})$ 同时显著减少了 Hessian 矩阵的计算频率。

此外，本文还与最近提出的分布式牛顿型方法进行了对比。这些方法在联邦学习等场景中使用了 Hessian 矩阵的概率聚合和压缩技术。虽然这些方法也减少了 Hessian 矩阵的计算频率，但它们通常需要在每次迭代中评估 Hessian 矩阵，并根据某种准则决定是否使用新的 Hessian 矩阵。相比之下，本文的方法在每 m 次迭代中只计算一次 Hessian 矩阵，从而进一步减少了计算成本。

5 论文的技术方法

5.1 问题描述和常用记号

我们考虑以下优化问题：

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}), \tag{1}$$

其中 f 是一个多次可微的函数，不一定是凸函数，我们假设其有下界：$\inf_{\mathbf{x}} f(\mathbf{x}) \geq f^*$。我们用 $\nabla f(\mathbf{x}) \in \mathbb{R}^d$ 表示其梯度，用 $\nabla^2 f(\mathbf{x}) \in \mathbb{R}^{d\times d}$ 表示其 Hessian 矩阵，计算于某个 $\mathbf{x} \in \mathbb{R}^d$。

我们的分析既适用于标准的欧几里得空间，也适用于更一般的情况，即由任意固定的对称正定矩阵 $\mathbf{B} = \mathbf{B}^\top \succ 0$ 定义的范数。在这种情况下，范数定义为

$$\|\mathbf{x}\| \stackrel{\text{def}}{=} \langle \mathbf{B}\mathbf{x}, \mathbf{x}\rangle^{1/2}, \quad \mathbf{x} \in \mathbb{R}^d. \tag{2}$$

因此，矩阵 $\mathbf{B}$ 负责固定我们问题的坐标系。在最简单的情况下，我们可以选择 $\mathbf{B} := \mathbf{I}$（单位矩阵），这恢复了经典的欧几里得几何。更高级的 $\mathbf{B}$ 选择可以考虑问题的特定结构（见例 3.2）。对偶对象（梯度）的范数以标准方式定义：

$$\|\mathbf{g}\|_* \stackrel{\text{def}}{=} \sup_{\mathbf{x}:\|\mathbf{x}\|\leq 1} \langle \mathbf{g}, \mathbf{x}\rangle = \left\langle \mathbf{g}, \mathbf{B}^{-1}\mathbf{g}\right\rangle^{1/2}, \quad \mathbf{g} \in \mathbb{R}^d.$$

矩阵 $\mathbf{A} \in \mathbb{R}^{d\times d}$ 的诱导范数定义为

$$\|\mathbf{A}\| \stackrel{\text{def}}{=} \sup_{\mathbf{x},\mathbf{y}:\|\mathbf{x}\|\leq 1,\|\mathbf{y}\|\leq 1} \langle \mathbf{A}\mathbf{x}, \mathbf{y}\rangle = \sup_{\mathbf{x}:\|\mathbf{x}\|\leq 1} \|\mathbf{A}\mathbf{x}\|_*.$$

我们假设 f 的 Hessian 是 Lipschitz 连续的，具有某个常数 $L > 0$：

$$\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \tag{3}$$

5.2 Lazy 牛顿迭代法

Lazy 牛顿迭代法的核心思想是重用先前计算的 Hessian 矩阵，而不是在每一步都重新计算。具体来说，算法在当前点计算梯度，而在过去的轨迹中使用 Hessian 矩阵。

Lazy 牛顿迭代法的数学模型定义如下：

$$Q_{\mathbf{x},\mathbf{z}}(\mathbf{y}) = \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{1}{2} \langle \nabla^2 f(\mathbf{z})(\mathbf{y} - \mathbf{x}), \mathbf{y} - \mathbf{x} \rangle$$

$$\mathbf{T}_M(\mathbf{x}, \mathbf{z}) = \arg\min_{\mathbf{y} \in \mathbb{R}^d} \left\{ Q_{\mathbf{x},\mathbf{z}}(\mathbf{y}) + \frac{M}{6} \|\mathbf{y} - \mathbf{x}\|^3 \right\} \tag{4}$$

- $\mathbf{x}$ 是当前点，$\nabla f(\mathbf{x})$ 是在当前点计算的梯度。
- $\mathbf{z}$ 是过去某次迭代的点，$\nabla^2 f(\mathbf{z})$ 是在该点计算的 Hessian 矩阵。
- M 是正则化参数，用于控制步长。

由于 Hessian 矩阵是过去某次迭代的计算结果，与当前点的 Hessian 矩阵可能存在差异，这种差异会引入误差。为了平衡这种误差，作者引入了**立方正则化**，并通过增加正则化参数 M 来抵消误差的影响。

定义意味着点 $\mathbf{T} = \mathbf{T}_M(\mathbf{x}, \mathbf{z})$ 是立方正则化模型的全局最小值，尽管该模型通常是非凸的。然而，事实证明，可以使用最初为信任域方法开发的标准技术（Conn et al., 2000）高效地计算该点。

定义 $r \stackrel{\text{def}}{=} \|\mathbf{T} - \mathbf{x}\|$。$\mathbf{T} = \mathbf{T}_M(\mathbf{x}, \mathbf{z})$ 的解满足以下平稳条件：

$$\begin{cases} \nabla f(\mathbf{x}) + \nabla^2 f(\mathbf{z})(\mathbf{T} - \mathbf{x}) + \frac{Mr}{2} \mathbf{B}(\mathbf{T} - \mathbf{x}) = \mathbf{0}, \\ \nabla^2 f(\mathbf{z}) + \frac{Mr}{2} \mathbf{B} \succeq 0. \end{cases}$$

因此，在非退化情况下，第一步可以表示为以下形式：

$$\mathbf{T} = \mathbf{x} - \left(\nabla^2 f(\mathbf{z}) + \frac{Mr}{2} \mathbf{B} \right)^{-1} \nabla f(\mathbf{x}), \tag{5}$$

并且可以通过求解相应的单变量非线性方程（Nesterov & Polyak, 2006）找到值 $r > 0$。这可以通过预先计算的 Hessian 矩阵的特征值或三对角分解非常高效地完成。通常，其成本与经典牛顿迭代法中的矩阵求逆相似。

定义以下量，对于 $\mathbf{y} \in \mathbb{R}^d$：

$$\xi(\mathbf{y}) \overset{\text{def}}{=} \left[-\lambda_{\min}\left(\mathbf{B}^{-1/2}\nabla^2 f(\mathbf{y})\mathbf{B}^{-1/2}\right)\right]_+, \tag{6}$$

其中 $[t]_+ \overset{\text{def}}{=} \max\{t, 0\}$ 表示正部，$\lambda_{\min}(\cdot)$ 是对称矩阵的最小特征值。如果对于某个 $\mathbf{y} \in \mathbb{R}^d$ 有 $\nabla^2 f(\mathbf{y}) \succeq 0$，则 $\xi(\mathbf{y}) = 0$。否则，$\xi(\mathbf{y})$ 表示 Hessian 矩阵的最小特征值相对于固定矩阵 $\mathbf{B} \succ 0$ 的大小（绝对值）。

Theorem 5.1. 设 $M \geq L$。那么，对于一次立方步 (4)，有以下结论：

$$f(\mathbf{x}) - f(\mathbf{T}) \geq \max\left\{\frac{1}{648M^2}\xi(\mathbf{T})^3, \frac{1}{72\sqrt{2M}}\|\nabla f(\mathbf{T})\|_*^{3/2}\right\} + \frac{M}{48}r^3 - \frac{11L^3}{M^2}\|\mathbf{z}-\mathbf{x}\|^3. \tag{7}$$

定理5.1展示了可以从一次带有立方正则化的 Lazy 牛顿迭代法中期望的进展。使用 Lazy Hessian 的代价是最后一项，如果 $\mathbf{z} := \mathbf{x}$（为当前迭代更新 Hessian 矩阵），则该项消失。

5.3 全局收敛速率

我们考虑算法的一个阶段，即我们在当前点 $\mathbf{z} := \mathbf{x}_0$ 计算一次 Hessian 矩阵，然后连续执行 m 次立方正则化步长 (4)：

$$\mathbf{x}_i = \mathbf{T}_M(\mathbf{x}_{i-1}, \mathbf{z}), \quad i = 1, \ldots, m. \tag{8}$$

我们使用某个固定的正则化常数 $M > 0$。从 (7) 中可以看出，使用旧的 Hessian 矩阵带来的误差与到起始点的距离的立方成正比：$\|\mathbf{x}_{i-1} - \mathbf{x}_0\|^3$ 对于每个 $1 \le i \le m$ 我们这里可以通过选择一个足够大的正则化参数 M 来平衡累积误差 (*accumulating error*)，并同时聚合 (7) 中的正项。在这里，我们的目标是要保证下式成立：

$$\frac{M}{48} \sum_{i=1}^{m} \|\mathbf{x}_i - \mathbf{x}_{i-1}\|^3 \ge \frac{11L^3}{M^2} \sum_{i=1}^{m} \|\mathbf{x}_{i-1} - \mathbf{x}_0\|^3. \tag{9}$$

事实上，我们证明只需选择 $\boxed{M \ge 6mL}$ 即可保证上式成立。这里的证明我们放到5.3.1节证明。

因此，我们能够处理 (8) 中 m 次连续 Lazy 步长的累积误差。

在每次使用旧 Hessian 矩阵的立方步长 (8) 内部阶段之后，我们在新的外部轮次中重新计算 Hessian 矩阵。每个内部阶段的长度 $m \ge 1$ 是算法的关键参数。为了方便表示，我们用 $\pi(k)$ 表示小于或等于 k 的最大的 m 的倍数：

$$\pi(k) \stackrel{\text{def}}{=} k - k \bmod m, \qquad k \ge 0. \tag{10}$$

主算法定义如下，根据我们之前的分析，我们将使用一个简单的固定规则来选择正则化参数：

$$\boxed{M := 6mL}. \tag{11}$$

Algorithm. 1 带有 Lazy Hessian 的立方牛顿法

输入：$\mathbf{x}_0 \in \mathbb{R}^d$，$m \ge 1$，$L > 0$，选择 $M \ge 0$。

for $k = 0, 1, \ldots$ do

设置最后一个快照点 $\mathbf{z}_k = \mathbf{x}_{\pi(k)}$

计算 Lazy 立方步长 $\mathbf{x}_{k+1} = \mathbf{T}_M(\mathbf{x}_k, \mathbf{z}_k)$

end for

Theorem 5.2. 假设 M 如 (1.4) 所示固定。假设算法1的前 k 次迭代的梯度高于所需的误差水平 $\varepsilon > 0$：

$$\|\nabla f(\mathbf{x}_i)\|_* \geq \varepsilon. \tag{12}$$

那么，达到精度 $\|\nabla f(\mathbf{x}_{k+1})\|_* \leq \varepsilon$ 所需的迭代次数最多为

$$k \leq \mathcal{O}\left(\frac{\sqrt{mL}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}}\right), \tag{13}$$

在迭代期间，Hessian 更新的总次数为

$$t \leq \mathcal{O}\left(\frac{\sqrt{L}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}\sqrt{m}}\right). \tag{14}$$

对于所有 Hessian 矩阵的最小特征值，有

$$\min_{1\leq i\leq k} \xi(\mathbf{x}_i) \leq \mathcal{O}\left(\frac{M^2(f(\mathbf{x}_0) - f^*)}{k}\right)^{1/3}. \tag{15}$$

假设所需的精度水平较小，即

$$\varepsilon \leq (f(\mathbf{x}_0) - f^*)^{2/3}\left(\frac{L}{m}\right)^{1/3},$$

这要求我们使用 *several Hessian* 方法

为了选择参数 m，我们需要考虑算法所有迭代中实际所需的计算量。我们用 *GradCost* 表示一次梯度步长的算术复杂度，用 *HessCost* 表示计算 Hessian 矩阵及其适当分解的成本。一般来说，我们有

$$\mathit{HessCost} = d \cdot \mathit{GradCost}, \tag{16}$$

其中 d 是问题 (1) 的维度。

Example 5.1. 设 $f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \phi((\mathbf{a}_i, \mathbf{x}))$，其中 $\mathbf{a}_i \in \mathbb{R}^d, 1 \leq i \leq n$ 是给定的数据，ϕ 是固定的单变量光滑函数。在这种情况下，

$$\nabla f(\mathbf{x}) = \mathbf{A}^\top \mathbf{s}(\mathbf{x}), \quad \text{且} \quad \nabla^2 f(\mathbf{x}) = \mathbf{A}^\top \mathbf{Q}(\mathbf{x})\mathbf{A},$$

其中 $\mathbf{A} \in \mathbb{R}^{n \times d}$ 是以 $\mathbf{a}_1, \ldots, \mathbf{a}_n$ 为行的矩阵；$\mathbf{s}(\mathbf{x}) \in \mathbb{R}^n$ 和 $\mathbf{Q}(\mathbf{x}) \in \mathbb{R}^{n \times n}$ 是由下式给出的对角矩阵：

$$\left[\mathbf{s}(\mathbf{x})\right]_{(i)} \stackrel{\text{def}}{=} \frac{1}{n}\phi'((\mathbf{a}_i, \mathbf{x})),$$

$$\left[\mathbf{Q}(\mathbf{x})\right]_{(i,i)} \stackrel{\text{def}}{=} \frac{1}{n}\phi''((\mathbf{a}_i, \mathbf{x})).$$

由于 Hessian 矩阵的结构，使用矩阵 $\mathbf{B} := \mathbf{A}^\top\mathbf{A}$ 来定义我们的原始范数（即二范数）非常方便。实际上，对于这种范数的选择，我们可以证明 Hessian 的 Lipschitz 常数仅依赖于损失函数 ϕ（即它是一个绝对数值常数，不依赖于数据）。同时，当使用单位矩阵 $\mathbf{I}$ 时，相应的 Lipschitz 常数依赖于数据矩阵 $\|\mathbf{A}\|$ 的大小。因此，在后一种情况下，Lipschitz 常数的值可能非常大。

假设计算 $\phi'(\cdot)$ 和 $\phi''(\cdot)$ 的成本为 $\mathcal{O}(1)$，且不依赖于问题参数。我们用 $\text{nz}(\mathbf{A})$ 表示矩阵 $\mathbf{A}$ 的非零元素数量。那么，

$$\textit{HessCost} = \mathcal{O}\left(d \cdot \text{nz}(\mathbf{A}) + d^3\right),$$

其中最后的三次项来自矩阵分解的计算（见5.3.2矩阵分解），且

$$\textit{GradCost} = \mathcal{O}\left(\text{nz}(\mathbf{A}) + d^2\right),$$

其中第二项来自使用分解。因此，关系（16）成立。

Example 5.2. 假设我们的目标函数由计算图表示（例如神经网络）。在这种情况下，我们可以通过自动微分技术（Nocedal & Wright, 2006）计算 Hessian-向量乘积 $\nabla^2 f(\mathbf{x})\mathbf{h}$，对于任何 $\mathbf{x}, \mathbf{h} \in \mathbb{R}^d$，其成本与计算梯度 $\nabla f(\mathbf{x})$ 相同。因此，通常我们可以通过 d 次 Hessian-向量乘积来计算 Hessian 矩阵，

$$\nabla^2 f(\mathbf{x}) = \left[\nabla^2 f(\mathbf{x})\mathbf{e}_1 \mid \ldots \mid \nabla^2 f(\mathbf{x})\mathbf{e}_d\right],$$

其中 $\mathbf{e}_1, \ldots, \mathbf{e}_d$ 是标准基向量。这满足（16）。

Example 5.3. 对于任何可微函数，我们可以使用有限差分来近似其 Hessian 矩阵。具体来说，对于固定参数 $\delta > 0$，我们可以形成 $\mathbf{H}_{x,\delta} \in \mathbb{R}^{\delta \times d}$，其元素为

$$[\mathbf{H}_{x,\delta}]^{(i,j)} := \frac{1}{\delta}\left[\nabla f(\mathbf{x} + \delta\mathbf{e}_i) - \nabla f(\mathbf{x})\right]^{(j)},$$

其中 $\mathbf{e}_1, \ldots, \mathbf{e}_d$ 是标准基向量，并使用以下对称化作为 true Hessian 的近似（见（Nocedal & Wright, 2006; Cartis et al., 2012; Grapiglia et al., 2022）了解更多细节）：

$$\frac{1}{2}\left[\mathbf{H}_{x,\delta} + \mathbf{H}_{x,\delta}^{\top}\right] \approx \nabla^2 f(\mathbf{x}).$$

因此，形成 Hessian 近似需要 $d+1$ 次梯度计算，这与（16）一致。

根据定理5.2，算法1的总算术复杂度可以估计为

$$\begin{aligned}
&\text{Arithmetic Complexity} && (1)\\
&= k \times \text{GradCost} + t \times \text{HessCost} && (2)\\
&\overset{(1.6),(1.7),(1.9)}{\leq} \mathcal{O}\left(\left(\sqrt{m} + \frac{d}{\sqrt{m}}\right)\frac{\sqrt{L}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}}\right) && (3)\\
&\times \textit{GradCost}. && (17)
\end{aligned}$$

Corrollary 5.1. 对于 $\boxed{m = 1}$，我们在每一步都更新 Hessian 矩阵。这对应于完整的立方牛顿法（Nesterov & Polyak, 2006），定理5.2恢复了其全局迭代复杂度：

$$k = t \leq \mathcal{O}\left(\frac{\sqrt{L}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}}\right).$$

因此，该方法的总算术复杂度为

$$\mathcal{O}\left(d \cdot \frac{\sqrt{L}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}}\right) \cdot \textit{GradCost}. \tag{18}$$

Corrollary 5.2. 对于 $\boxed{m = d}$，我们得到了一个阶段的最优选择，它最小化了（17）的右侧。算法1的总算术复杂度变为

$$\mathcal{O}\left(\sqrt{d} \cdot \frac{\sqrt{L}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}}\right) \cdot \text{GradCost}.$$

这比完整的立方牛顿法提高了 $\sqrt{d}$ 倍。

现在，我们可以查看算法生成的所有 Hessian 矩阵的最小特征值。

Corrollary 5.3. 固定某个 $\varepsilon > 0$，

$$k = \frac{\sqrt{mL}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}} \tag{19}$$

并执行 k 次算法1的迭代。根据定理5.2，我们因此确保 $\min_{1\le i\le k}\|\nabla f(\mathbf{x}_i)\|_* \le \mathcal{O}(\varepsilon)$。同时，

$$\min_{1\le i\le k}\xi(\mathbf{x}_i) \overset{(15)}{\le} \left(\frac{648M^2(f(\mathbf{x}_0)-f^*)}{k}\right)^{1/3} \overset{(19)}{\cong} 2^{5/3}3^2\sqrt{mL\varepsilon}.$$

因此，Hessian 矩阵的负特征值不会太大。对于 $\mathcal{O}(\varepsilon)$ 级别的梯度范数，我们保证最小特征值(6)的级别为 $\mathcal{O}(\sqrt{mL\varepsilon})$。

5.3.1 $M \ge 6mL$ 取法的证明

我们用 $r_{k+1} := \|\mathbf{x}_{k+1}-\mathbf{x}_k\|$ 表示相应的步长。根据定理5.1，对于每个 $0\le k\le m-1$，我们有以下关于目标函数的不等式：

$$f(\mathbf{x}_k)-f(\mathbf{x}_{k+1}) \tag{4}$$

$$\ge \max\left\{\frac{1}{648M^2}\xi(\mathbf{x}_{k+1})^3, \frac{1}{72\sqrt{2M}}\|\nabla f(\mathbf{x}_{k+1})\|_*^{3/2}\right\} \tag{5}$$

$$+\frac{M}{48}r_{k+1}^3 - \frac{11L^3}{M^2}\|\mathbf{x}_0-\mathbf{x}_k\|^3. \tag{6}$$

将这个不等式对不同的 k 进行累加，并使用最后一个负项的三角不等式，

$$\|\mathbf{x}_0-\mathbf{x}_k\| \le \sum_{i=1}^k r_i,$$

我们得到

$$f(\mathbf{x}_0)-f(\mathbf{x}_m) \tag{7}$$

$$\ge \sum_{k=1}^m \max\left\{\frac{1}{648M^2}\xi(\mathbf{x}_k)^3, \frac{1}{72\sqrt{2M}}\sum_{k=1}^m\|\nabla f(\mathbf{x}_k)\|_*^{3/2}\right\} \tag{8}$$

$$+\frac{M}{48}\sum_{k=1}^m r_k^3 - \frac{11L^3}{M^2}\sum_{k=1}^{m-1}\left(\sum_{i=1}^k r_i\right)^3. \tag{B.47}$$

接下来，我们使用以下简单的引理。

Lemma 1. 对于任何正数序列 $\{r_k\}_{k\ge1}$，对于任何 $m\ge1$，有

$$\sum_{k=1}^{m-1}\left(\sum_{i=1}^k r_i\right)^3 \le \frac{m^3}{3}\sum_{k=1}^{m-1} r_k^3. \tag{B.48}$$

Proof. 引理证明如下。

我们通过归纳法证明(1)。对于 $m = 1$，显然成立，这是我们的归纳奠基。

假设对于某个任意的 $m \geq 1$ 成立。那么

$$\sum_{k=1}^{m}\left(\sum_{i=1}^{k} r_i\right)^3 = \sum_{k=1}^{m-1}\left(\sum_{i=1}^{k} r_i\right)^3 + \left(\sum_{i=1}^{m} r_i\right)^3 \leq \frac{48}{3}\sum_{k=1}^{m-1} r_k^3 + \left(\sum_{i=1}^{m} r_i\right)^3. \tag{B.49}$$

对第二个项应用凸函数 $t \mapsto t^3$ 的 Jensen 不等式，我们得到

$$\left(\sum_{i=1}^{m} r_i\right)^3 = m^3\left(\frac{1}{m}\sum_{i=1}^{m} r_i\right)^3 \leq m^2\sum_{i=1}^{m} r_i^3. \tag{B.50}$$

因此，

$$\sum_{k=1}^{m}\left(\sum_{i=1}^{k} r_i\right)^3 \leq \left(\frac{m^3}{3} + m^2\right)\sum_{k=1}^{m} r_k^3 \leq \frac{(m+1)^3}{3}\sum_{k=1}^{m} r_k^3.$$

□

使用这个不等式，我们确保(7)的右侧为正。为此，我们需要使用一个增加的正则化参数值。因此，我们证明了以下保证。

Corrollary 5.4. 设 $M \geq 6mL$。那么，

$$f(\mathbf{x}_0) - f(\mathbf{x}_m) \geq \sum_{k=1}^{m} \max\left\{\frac{1}{648M^2}\xi(\mathbf{x}_k)^3, \frac{1}{72\sqrt{2M}}\|\nabla f(\mathbf{x}_k)\|_*^{3/2}\right\}. \tag{B.51}$$

Theorem 5.3. 假设目标函数有下界：$\inf_{\mathbf{x}} f(\mathbf{x}) \geq f^*$。设正则化参数 M 如(11)所示固定。假设算法1的前 k 次迭代的梯度高于所需的误差水平 $\varepsilon > 0$：

$$\|\nabla f(\mathbf{x}_i)\|_* \geq \varepsilon. \tag{B.52}$$

那么，达到精度 $\|\nabla f(\mathbf{x}_{k+1})\|_* \leq \varepsilon$ 所需的迭代次数最多为

$$k \leq \mathcal{O}\left(\frac{\sqrt{mL}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}}\right), \tag{B.53}$$

其中 $\mathcal{O}(\cdot)$ 隐藏了一些绝对数值常数。在这些迭代期间，Hessian 更新的总次数 t 满足

$$t \leq \mathcal{O}\left(\frac{\sqrt{L}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}\sqrt{m}}\right). \tag{B.54}$$

对于所有 Hessian 矩阵的最小特征值，有

$$\min_{1\leq i\leq k}\left[-\lambda_{\min}\left(\mathbf{B}^{-1/2}\nabla^2 f(\mathbf{x}_i)\mathbf{B}^{-1/2}\right)\right]_+ \leq \left(\frac{648M^2(f(\mathbf{x}_0) - f^*)}{k}\right)^{1/3}. \tag{B.55}$$

Proof. 不失一般性，我们假设 k 是 m 的倍数：$k = tm$。那么，对于方法的第 i 个阶段（$1 \leq i \leq t$），我们有

$$f(\mathbf{x}_{m(i-1)}) - f(\mathbf{x}_{mi}) \geq \frac{m}{72\sqrt{2M}}\varepsilon^{3/2} = \frac{\sqrt{m}}{144\sqrt{3L}}\varepsilon^{3/2}. \tag{B.56}$$

对所有阶段进行累加，我们得到

$$f(\mathbf{x}_0) - f^* \geq f(\mathbf{x}_0) - f(\mathbf{x}_k) \geq \frac{t\sqrt{m}}{144\sqrt{3L}}\varepsilon^{3/2}. \tag{B.57}$$

这给出了 t 的所需上界（14）。考虑到 $k = tm$，（B.53）立即成立。对于 Hessian 矩阵的最小特征值 $\xi(\cdot)$，累加（B.51）得到

$$f(\mathbf{x}_0) - f^* \geq f(\mathbf{x}_0) - f(\mathbf{x}_k) \geq \frac{k}{648M^2}\min_{1\leq i\leq k}\xi(\mathbf{x}_i)^3. \tag{B.58}$$

因此，

$$\min_{1\leq i\leq k}\left[-\lambda_{\min}\left(\mathbf{B}^{-1/2}\nabla^2 f(\mathbf{x}_i)\mathbf{B}^{-1/2}\right)\right]_+ \equiv \min_{1\leq i\leq k}\xi(\mathbf{x}_i) \leq \left(\frac{648M^2(f(\mathbf{x}_0) - f^*)}{k}\right)^{1/3},$$

□

5.3.2 矩阵分解

在每 m 次迭代计算一次 Hessian 矩阵后，我们希望能够在方法的每次迭代中高效地解决相应的子问题。

为了求解立方牛顿法（4）中的子问题，我们需要找到参数 $r > 0$，在非退化情况下，它是以下非线性方程的根（见 Nesterov & Polyak, 2006）：

$$\varphi(r) \overset{\text{def}}{=} \|\mathbf{s}(r)\| - r = 0, \tag{24}$$

其中 $\mathbf{s}(r) \overset{\text{def}}{=} \left(\nabla^2 f(\mathbf{z}) + \frac{Mr}{2}\mathbf{B}\right)^{-1} \nabla f(\mathbf{x})$，对于使得矩阵正定的 r。对（24）应用标准的二分法或单变量牛顿法（步长为 $r^+ = r - \varphi(r)/\varphi'(r)$），我们需要执行的主要操作是求解线性方程：

$$\left(\nabla^2 f(\mathbf{z}) + \tau\mathbf{B}\right)\mathbf{h} = -\nabla f(\mathbf{x}), \tag{25}$$

对于不同的 $\tau > 0$。这与我们在算法2中每次迭代需要执行的操作类型相同。

现在，假设我们有以下 Hessian 矩阵的分解：

$$\nabla^2 f(\mathbf{z}) = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top, \quad \text{其中} \quad \mathbf{U}\mathbf{U}^\top = \mathbf{B}, \tag{26}$$

且 $\mathbf{\Lambda} \in \mathbb{R}^{d\times d}$ 是一个**对角**或**三对角**矩阵。因此，$\mathbf{U}$ 是关于 $\mathbf{B}$ 正交的一组向量。在 $\mathbf{B} = \mathbf{I}$（单位矩阵）且 $\mathbf{\Lambda}$ 为对角矩阵的情况下，（26）被称为**特征值分解**，并在大多数标准线性代数库中实现。

一般来说，分解（26）可以在 $\mathcal{O}(d^3)$ 次算术操作中计算。具体来说，我们可以对以下矩阵应用标准的正交化分解：

$$\mathbf{B}^{-1/2}\nabla^2 f(\mathbf{z})\mathbf{B}^{-1/2} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top, \quad \text{其中} \quad \mathbf{V}\mathbf{V}^\top = \mathbf{I},$$

然后设 $\mathbf{U} := \mathbf{B}^{1/2}\mathbf{V}$，得到（26）。注意，（25）的解可以表示为

$$\mathbf{h} = -\mathbf{U}^{-\top}\left(\mathbf{\Lambda} + \tau\mathbf{I}\right)^{-1}\mathbf{U}^{-1}\nabla f(\mathbf{x}),$$

对于任何给定的 $\tau > 0$ 和 $\nabla f(\mathbf{x})$，它可以在 $\mathcal{O}(d^2)$ 次算术操作中计算。使用三对角分解在实践中可能更高效。实际上，对于三对角矩阵 $\mathbf{\Lambda}$，矩阵 $\mathbf{\Lambda} + \tau\mathbf{I}$ 的逆运算仍然只需要 $\mathcal{O}(d)$ 次操作，而计算这种分解所需的计算资源和浮点精度较少。在实践中，利用 Hessian 矩阵的结构（例如矩阵是稀疏的）也可以进一步减少每一步的算术成本。

5.4 凸问题的优化

在凸优化问题中，目标函数的 Hessian 矩阵是半正定的（即 $\nabla^2 f(\mathbf{x}) \succeq 0$）。这使得可以使用更简单的正则化技术来简化子问题的求解。本文针对凸问题，提出了 **带有 LazyHessian 的正则化牛顿法（Algorithm 2）**，使用二次正则化替代立方正则化，从而使得子问题更容易求解。

Algorithm. 2 Regularized Newton with Lazy Hessians

Require: $\mathbf{x}_0 \in \mathbb{R}^d$, $m \geq 1$, $L > 0$
Ensure: Sequence $\{\mathbf{x}_k\}_{k\geq 0}$ converging to a second-order stationary point
Choose $M := 3mL$ {Set regularization parameter}
for $k = 0, 1, \ldots$ **do**
  Set last snapshot point $\mathbf{z}_k = \mathbf{x}_{\pi(k)}$ {Reuse Hessian from $\pi(k)$-th iteration}
  Compute $\lambda_k = \sqrt{M\|\nabla f(\mathbf{x}_k)\|_*}$ {Compute regularization parameter}
  Compute lazy Newton step:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\nabla^2 f(\mathbf{z}_k) + \lambda_k \mathbf{B}\right)^{-1} \nabla f(\mathbf{x}_k)$$

end for

算法2的全局复杂度界与算法1的复杂度界相同（最多相差一个对数项）。然而，算法2的每次迭代更容易实现，因为它仅涉及求解一个线性系统。

$$M := 3mL, \tag{20}$$

- **迭代次数：**达到精度 $\|\nabla f(\mathbf{x}_{k+1})\|_* \leq \varepsilon$ 所需的迭代次数为：

$$k \leq \mathcal{O}\left(\frac{\sqrt{mL}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}} + \ln\frac{\|\nabla f(\mathbf{x}_0)\|_*}{\varepsilon}\right). \tag{22}$$

- **Hessian 更新次数：**在达到精度 ε 的过程中，Hessian 矩阵的更新次数为：

$$t \leq \mathcal{O}\left(\frac{\sqrt{L}(f(\mathbf{x}_0) - f^*)}{\varepsilon^{3/2}\sqrt{m}} + \frac{1}{m}\ln\frac{\|\nabla f(\mathbf{x}_0)\|_*}{\varepsilon}\right). \tag{23}$$

5.5 局部超线性收敛证明 (强凸函数)

在本节中，我们讨论了带有 Lazy Hessian 更新的二阶优化算法的局部收敛性。我们展示了在每个阶段内具有恒定因子的快速线性收敛速率，并考虑了 Hessian 更新后的超线性收敛。

假设我们的目标是**强凸**的，即 $\nabla^2 f(\mathbf{x}) \succeq \mu \mathbf{B}, \forall \mathbf{x} \in \mathbb{R}^d$，其中 $\mu > 0$ 是强凸参数。为了简化分析，我们要求在整个空间中的所有点都满足强凸性，尽管也可以在解的邻域内分析牛顿迭代法长的强凸性。

Theorem 5.4. 设 $M \geq 0$。假设初始梯度足够小：$\|\nabla f(\mathbf{x}_0)\|_* \leq \frac{1}{2}\mathcal{G}_0$，其中 $\mathcal{G}_0 := \frac{\mu^2}{3L+M}$。那么，算法1在梯度范数上具有超线性收敛性，对于 $k \geq 0$：

$$\|\nabla f(\mathbf{x}_k)\|_* \leq \mathcal{G}_0 \cdot \left(\frac{1}{2}\right)^{(1+m)^{\pi(k)}(1+k \bmod m)},$$

其中 $\pi(k)$ 由规定术语 (10) 定义。

Corollary 5.5. 结合定理5.2和定理5.4，我们得出结论：对于使用正则化参数 M 如(11)所示的算法1，从任意 $\mathbf{x}_0$ 开始，最小化强凸函数所需的 Lazy 步长次数为

$$k \leq \mathcal{O}\left(\frac{m^2 L^2 (f(\mathbf{x}_0) - f^*)}{\mu^3} + \frac{1}{\ln(1+m)} \ln\ln \frac{\mu^2}{mL\varepsilon}\right),$$

以达到 $\|\nabla f(\mathbf{x}_k)\|_* \leq \varepsilon$。

我们还展示了算法2在局部具有稍差但仍然超线性的收敛速率，这在实践中已经非常快了。例如，对于 $m = 1$，算法 2 具有阶数为 3/2 的超线性收敛速率，而算法1的阶数为 2。

Theorem 5.5. 设 $M \geq 0$。假设初始梯度足够小：$\|\nabla f(\mathbf{x}_0)\|_* \leq \frac{1}{2^2}\mathcal{G}_0$，其中 $\mathcal{G}_0 := \frac{\mu^2}{2^3(3L+4M)}$。那么，算法2在梯度范数上具有超线性收敛性，对于 $k \geq 0$：

$$\|\nabla f(\mathbf{x}_k)\|_* \leq \mathcal{G}_0 \cdot \left(\frac{1}{2}\right)^{2(1+m/2)^{\pi(k)}(1+(k \bmod m)/2)},$$

其中 $\pi(k)$ 由(10)定义。

Corollary 5.6. 结合定理5.4和定理5.5，我们得出结论：对于使用正则化参数 M 如(20)所示的算法2，从任意 $\mathbf{x}_0$ 开始，最小化强凸函数所需的 Lazy 步长次数为

$$k \leq \mathcal{O}\left(\frac{m^2L^2(f(\mathbf{x}_0)-f^*)}{\mu^3} + \ln\frac{mL\|\nabla f(\mathbf{x}_0)\|_*}{\mu^2} + \frac{1}{\ln(1+m/2)}\ln\ln\frac{\mu^2}{mL\varepsilon}\right),$$

以达到 $\|\nabla f(\mathbf{x}_k)\|_* \leq \varepsilon$。

5.6 实践实现之自适应搜索

前一小节的矩阵分解已经在全局收敛速率处提到，这里便不再赘述，只说明自适应搜索的实践实现。

为了得到我们的结果，我们需要选择正则化参数 M 与 mL 成正比，其中 $m \geq 1$ 是一个阶段的长度，$L > 0$ 是 Hessian 的 Lipschitz 常数。这种选择的一个缺点是，我们实际上需要知道 Lipschitz 常数，而这在实践中并不总是可行的。此外，使用固定的正则化参数会使方法变得保守，阻止了大步长的可能性。同时，从局部角度来看，目标函数的最佳二次近似是纯二阶泰勒多项式。因此，在解的邻域内，最佳策略是使 M 趋近于零，从而获得最快的局部超线性收敛速率（见章节5.5）。

在二阶优化算法中使用自适应搜索已经研究了多年（Nesterov & Polyak, 2006; Cartis et al., 2011a,b; Grapiglia & Nesterov, 2017, 2019; Doikov & Nesterov, 2021a,b）。众所周知，这种方案在实际中表现非常好，同时自适应搜索使方法具有**通用性**（Grapiglia & Nesterov, 2017; Doikov & Nesterov, 2021a），即能够适应 Hessian 的最佳 Hölder 度，甚至**超通用性**（Doikov et al., 2022），即自动选择目标函数的二阶或三阶导数的 Hölder 度，从而在广泛的问题类上具有最佳的全局复杂度保证。

我们提出了算法3，它自适应地改变 M 的值，并在 m 步后检查函数进展以验证其选择。重要的是，不需要知道 Lipschitz 常数 L。

Algorithm. 3 带有 Lazy Hessian 的自适应立方牛顿法

Require: $\mathbf{x}_0 \in \mathbb{R}^d$, $m \geq 1$。固定某个 $M_0 > 0$。
for $t = 0, 1, \ldots$ **do**
  计算快照 Hessian $\nabla^2 f(\mathbf{x}_{tm})$
  更新 $M_t = 2 \cdot M_t$
  for $i = 1, \ldots, m$ **do**
    Lazy 立方步长 $\mathbf{x}_{tm+i} = T_{M_t}(\mathbf{x}_{tm+i-1}, \mathbf{x}_{tm})$
  end for
  if $f(\mathbf{x}_{tm}) - f(\mathbf{x}_{tm+m}) \geq \frac{1}{\sqrt{M_t}} \sum_{i=1}^{m} \|\nabla f(\mathbf{x}_{tm+i})\|_*^{3/2}$ **then**
    设置 $M_{t+1} = \frac{1}{4} \cdot M_t$
  end if
end for

M_0 是正则化常数的**初始猜测**，可以进一步动态增加或减少。

Theorem 5.6. 假设在前 t 个阶段中，前 tm 次迭代的梯度高于所需的误差水平 $\varepsilon > 0$：

$$\|\nabla f(\mathbf{x}_i)\|_* \geq \varepsilon. \tag{27}$$

那么，达到精度 $\|\nabla f(\mathbf{x}_{tm+1})\|_* \leq \varepsilon$ 所需的阶段数最多为

$$t \leq \mathcal{O}\left(\sqrt{\max\left\{\frac{M_0}{m}, L\right\}} \cdot \frac{f(\mathbf{x}_0) - f^*}{\varepsilon^{3/2}\sqrt{m}}\right). \tag{28}$$

在这些阶段期间，梯度调用的总次数 N 满足

$$N \leq 2tm + \max\left\{1, \log_2 \frac{2^9 3^5 mL}{M_0}\right\} m. \tag{29}$$

Remark 1. 根据定理5.6，自适应算法的全局复杂度与定理5.2中给出的固定正则化常数方法的复杂度相同。由于(29)，每个阶段尝试不同 M_t 的平均次数仅为两次。

显然，类似的自适应搜索也可以纳入算法2中，用于凸情况。

6 论文的实验结果

通过数值实验展示了所提出的带有 LazyHessian 更新的二阶方法的性能。考虑以下凸最小化问题，目标函数为 **Soft Maximum**（log-sum-exp）：

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) := \mu \ln\left(\sum_{i=1}^n \exp\left(\frac{\langle \mathbf{a}_i, \mathbf{x}\rangle - b_i}{\mu}\right)\right) \approx \max_{1\le i\le n} \left[\langle \mathbf{a}_i, \mathbf{x}\rangle - b_i\right].$$

为了生成数据，随机采样向量 $\bar{\mathbf{a}}_1, \ldots, \bar{\mathbf{a}}_n \in \mathbb{R}^d$ 和 $\mathbf{b} \in \mathbb{R}^n$，元素来自 $[-1, 1]$ 的均匀分布。然后，我们使用这些向量构建辅助目标函数 $\bar{f}$，并设置 $\mathbf{a}_i := \bar{\mathbf{a}}_i - \nabla \bar{f}(\mathbf{0})$。这确保了最优解位于原点，因为 $\nabla f(\mathbf{0}) = \mathbf{0}$。初始点为 $\mathbf{x}_0 = (1, \ldots, 1)$。

对于原始范数（2），使用矩阵：

$$\mathbf{B} := \sum_{i=1}^n \mathbf{a}_i \mathbf{a}_i^\top + \delta \mathbf{I} \succ 0,$$

其中 $\delta > 0$ 是一个小的扰动参数，用于确保正定性。然后，Hessian 矩阵的 Lipschitz 常数由以下公式界定：$L = 2/\mu^2$，其中 $\mu > 0$ 是平滑参数。

由于问题是凸的，我们可以应用带有梯度正则化的牛顿法（Algorithm 2）。在图 1 中，比较了不同参数 m 值（即 Hessian 更新的频率）的性能。正则化参数固定为 $M := 1$。还展示了梯度法（Gradient Method, GM）作为标准基线的性能。

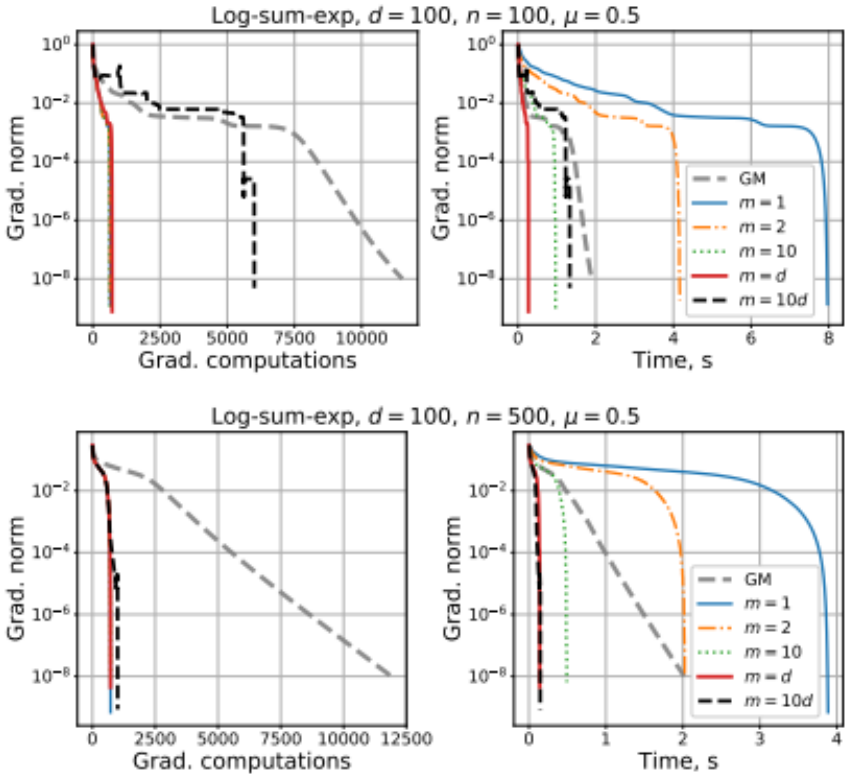

Fig. 1: Hessian 更新频率对算法性能的影响

实验结果表明：

1. 通过增加 Hessian 更新的频率（即增加 m 的值），可以显著提高算法的总体性能。最优的更新频率是 $m = d$，这与理论分析一致。

2. 与每次迭代都更新 Hessian 矩阵的经典牛顿法相比，Lazy Hessian 更新方法在保持收敛速度的同时，显著减少了计算成本。

3. 二阶方法（如立方牛顿法和正则化牛顿法）在收敛速度和计算效率上均优于经典的梯度法。

7 论文的结论

在本文中，开发了新的带有 **LazyHessian 更新**的二阶优化算法，用于解决一般的非凸优化问题。展示了在多次迭代中重用先前计算的 Hessian 矩阵可以显著提高算法的效率，而不需要每次迭代都更新 Hessian 矩阵。

通过使用**立方正则化**和**梯度正则化**技术，证明了带有 LazyHessian 更新的二阶方法在广泛的凸和非凸优化问题中具有快速的全局和局部收敛速度。证明了最优的 Hessian 更新策略是每 d 次迭代更新一次 Hessian 矩阵，其中 d 是问题的维度。这显著减少了二阶算法的总算术复杂度，提高了 $\sqrt{d}$ 倍。

8 未来可行的研究

1. **特定 Hessian 结构的研究：**研究具有特定 Hessian 结构（如稀疏性或某些谱聚类）的问题，可能需要不同的 Hessian 更新策略。

2. **与拟牛顿法的联系：**探索 LazyHessian 更新与经典拟牛顿法之间的联系。拟牛顿法通过逐步更新 Hessian 矩阵的近似来优化问题。最近发现的拟牛顿法的非渐近复杂度界限（Rodomanov & Nesterov, 2021; Rodomanov, 2022）可能对实现这一目标特别有用。

3. **高阶优化方案的推广：**将分析推广到高阶优化方案（Nesterov, 2021）。高阶方法使用更高阶的导数信息来加速收敛。主要挑战是如何高效地重用高阶张量，这可能需要使用一些高级的张量分解技术。

4. **凸优化的进一步研究：**在凸优化中，除了梯度范数外，另一个常见的精度度量是函数残差。可能可以证明使用函数残差作为度量时，LazyHessian 更新方法具有更好的收敛速度。还可以研究加速和超通用的二阶方法。

9 参考

参考凸优化教材牛顿迭代法部分内容 [1]

Reference

[1] NESTEROV Y. Introductory lectures on convex optimization: A basic course: volume 87[M]. Springer Science & Business Media, 2013.