自动梯度下降：无需超参数的深度学习

Jeremy Bernstein* **Chris Mingard*** **Kevin Huang Navid Azizan Yisong Yue**麻省理工学院
牛津大学 华盛顿大学 麻省理工学院 加州理工学院

* 表示贡献相同。

arXiv:2304.05187v1 [cs.LG] 11 Apr 2023

摘要

深度神经网络的架构由层数、每层的宽度以及整体网络拓扑结构明确界定。现有优化框架忽略了这些信息，转而采用隐式架构信息（例如二阶方法）或与架构无关的距离函数（例如镜像下降）。与此同时，实践中最流行的优化器——Adam——基于启发式方法。本文构建了一个用于推导优化算法的新框架，能够明确利用神经架构。该理论将镜像下降扩展至非凸复合目标函数：其核心思想是转换 Bregman 散度，以考虑神经架构的非线性结构。对深度全连接网络进行详细推导后，得到自动梯度下降：一种无需任何超参数的一阶优化器。自动梯度下降能够开箱即用地训练全连接网络和卷积网络，并可扩展至 ImageNet 规模。PyTorch 实现可在 `https://github.com/jxbz/agd` 获取，也见附录B。总体而言，本文为下一代架构依赖型优化器提供了严谨的理论基础，这类优化器无需超参数即可自动运行。

关键词： 主化-最小化元算法、算子扰动理论、架构感知优化

目录

1 引言 **2**
1.1 相关工作.3 1.2 预备知识.4

2 主化-最小化用于一般学习问题 5
2.1 线性化误差的分解.5 2.2 函数展开与函数主化.7 2.3 恢复现有框架.7

3 主化-最小化用于深度学习问题 8
3.1 推导自动梯度下降.8 3.2 收敛性分析.10 3.3 实验.11

4 讨论 **12**

A 证明 **18**

B PyTorch实现 23

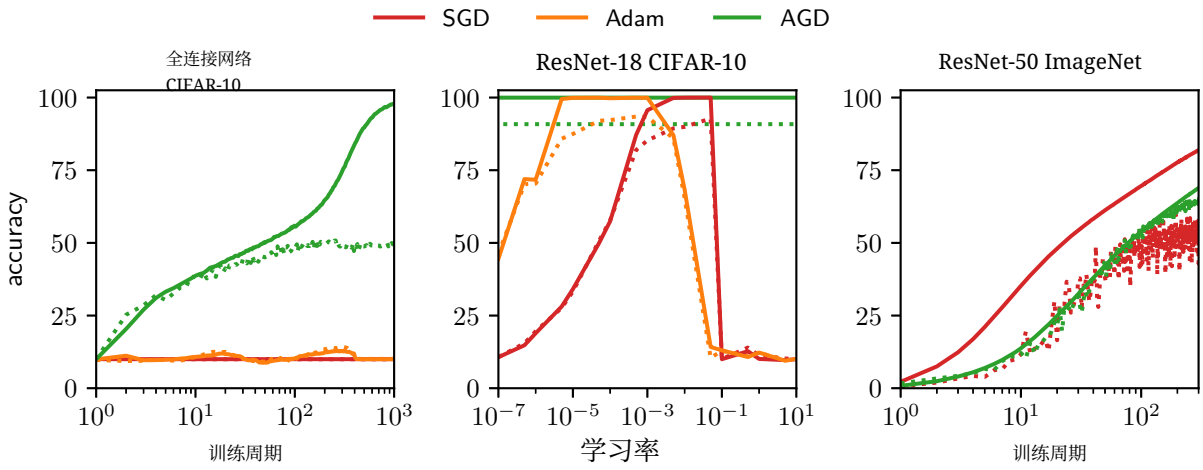

图 1：自动梯度下降无需超参数即可可靠地训练神经网络。实线表示训练准确率，虚线表示测试准确率。网络未进行正则化，并禁用了偏置和仿射参数，因为自动梯度下降目前尚不支持这些特性。在**左图**中，与自动梯度下降不同，Adam 和随机梯度下降使用各自的默认学习率（Adam 为 0.001，随机梯度下降为 0.1）时，未能在 CIFAR-10 上训练一个 32 层全连接网络。**中图**展示了在 CIFAR-10 上训练 ResNet-18 时的学习率网格搜索结果。自动梯度下降的表现与经过最佳调参的 Adam 和随机梯度下降相当。在**右图**中，自动梯度下降在 ImageNet 上将 ResNet-50 训练至 65.5% 的 Top-1 测试准确率。ImageNet 基线采用随机梯度下降，学习率为 0.1，且不使用学习率衰减计划。

1 引言

自动微分推动了深度学习领域的快速创新。PyTorch（Paszke 等人，2019 年）和 Theano（Al-Rfou 等人，2016 年）等软件包推动了一种编程范式：用户（1）通过组合可微算子来定义神经网络架构，并（2）提供训练数据。随后，软件包会通过递归应用链式法则，自动计算训练数据上的误差梯度。此时，用户必须再次参与：（3）从众多优化算法中选择一种，并（4）手动调整其超参数，尤其是初始学习率和学习率衰减计划（Goodfellow 等人，2016 年）。

但手动调整超参数令人厌烦。超参数数量繁多，使得我们难以对不同深度学习算法的表现进行排序（Lucic et al., 2017；Schmidt et al., 2021），也难以复现文献中的结果（Henderson et al., 2018）。超参数妨碍了我们建立对深度学习泛化能力的科学认识（Jiang et al., 2020；Farhang et al., 2022）。此外，当我们为了追求更强大的人工智能而以最大规模训练神经网络时，超参数网格搜索可能造成数百万美元的计算成本（Sharir et al., 2020）。

超参数只是生活中的既定事实吗？本文的论点是不：它们不是。深度学习涉及通过最小化已知目标函数，将已知函数拟合到已知数据。如果我们能够分别刻画这些组成部分，以及刻画它们之间的交互方式，那么——原则上——就不应再有需要调节的剩余自由度（Orabona & Cutkosky, 2020）。沿着这一思路继续发展，便产生了自动梯度下降（AGD）：一种不含任何超参数的神经网络优化器。自动梯度下降与自动微分互为补充，有助于实现通用机器学习工作流的自动化。

我们的推导以两种现有工具为核心，本文的主要理论贡献正是对二者进行创新性结合。首先，我们使用凸分析中的经典工具*Bregman*散度（Bregman，1967年；Dhillon 和 Tropp，2008）来刻画神经网络与损失函数之间的交互。其次，我们使用一种名为深度相对信任 的工具（Bernstein 等，2020），来刻画权重与网络输出之间高度非线性的交互。借助这些工具

优化器	参考文献	超参数 Free	宽度缩放	深度缩放	自动调度	内存 Cost
Adam	Kingma 与 Ba（2015）	✗	✗	✗	✗	3 ×#权重
随机梯度下降 + 动量。	Bottou 等人（2018年）	✗	✗	✗	✗	2 ×#权重
随机梯度下降 + μP	Yang & Hu（2021）	✗	✓	✗	✗	1 ×#权重
AGD	本文	✓	✓	✓	✓	1 ×#权重

表 1：实用优化器比较。Adam 和动量随机梯度下降使用梯度统计量的滑动估计，因此比自动梯度下降占用更多内存。此外，Adam 和随机梯度下降无法指导如何根据网络架构调整超参数；不过，对于宽度缩放，μP 解决了这一问题。

有了这些，我们可以应用主化-最小化元算法（Lange，2016）来推导出一种专门针对深度网络目标函数的优化器。总而言之，自动梯度下降的推导遵循三个主要步骤：

步骤 1：函数展开。我们使用*Bregman*散度来表示目标函数$\mathcal{L}(\boldsymbol{w})$ 的线性化误差，其表达基于网络 $\boldsymbol{f}$的函数扰动 $\Delta \boldsymbol{f}$ 。

步骤 2：架构扰动界。我们使用深度相对信任，将权重扰动 $\Delta \boldsymbol{w}$ 的大小和结构与其引起的函数扰动 $\Delta \boldsymbol{f}$的大小联系起来。

步骤 3：主化-最小化。我们将深度相对信任代入 Bregman散度，从而得到显式依赖于架构的主化形式。对 $\Delta \boldsymbol{w}$ 进行最小化即可得到一个优化器。

贡献总结 本文通过将极大化-最小化元算法应用于深度网络目标函数，推导出自动梯度下降（AGD）。AGD 无需超参数即可训练所有经过测试的网络架构，并且能够扩展到 ResNet-50 等深度网络以及 ImageNet 等大型数据集。即使 Adam 和随机梯度下降（SGD）使用默认超参数训练失败，AGD 也能开箱即用地完成训练。

1.1 相关工作

优化理论 一阶优化器利用目标函数$\mathcal{L}(\boldsymbol{w})$的一阶泰勒展开，尤其是梯度 $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$。理论研究包括镜像下降（Nemirovsky & Yudin，1983）、自然梯度下降（Amari，1998年）和高斯-牛顿法（Björck，1996年）。这些方法已在深度学习领域得到探索（Pascanu & Bengio，2014；Azizan & Hassibi，2019；Sun 等，2022）。一阶方法适用于深度学习，因为目标函数的梯度可以通过递归应用链式法则获得，即误差反向传播（Rumelhart 等，1986）。

二阶优化器利用目标函数的二阶泰勒展开$\mathcal{L}(\boldsymbol{w})$—特别是梯度 $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$ 和 Hessian $\nabla^2_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$。示例包括牛顿法（Nocedal & Wright，1999）和立方正则化牛顿法（Nesterov & Polyak，2006）。朴素地说，二阶方法不太适用于深度学习，因为在高维情况下，相关 Hessian计算的成本高得令人难以承受。尽管如此，人们仍已努力绕过这一问题（Agarwal 等，2017）。

主化-最小化元算法（Lange，2016）是一种可用于推导优化器的算法模式。要应用该元算法，必须首先推导出目标函数的一个上界，使其对于某个整数 k，在泰勒级数中截至 k 阶都与目标函数相匹配。然后可以将此 主化 最小化，以此作为降低原始目标函数的代理。图 2 展示了针对 $k = 1$ 的元算法。

深度学习理论 利普希茨光滑性假设——即对 Hessian 特征值施加的全局约束——常用于推导和分析神经网络优化器（Agarwal 等，2016）。然而，这一假设受到质疑（Zhang 等，2020），甚至有证据表明存在相反的关系：Hessian 谱对优化器的选择高度敏感（Cohen 等，2021）。

这些考量推动了更明确地针对神经架构的理论发展。例如，Bernstein 等（2020）使用了一种称为深度相对信任的架构扰动界来彩票娱乐

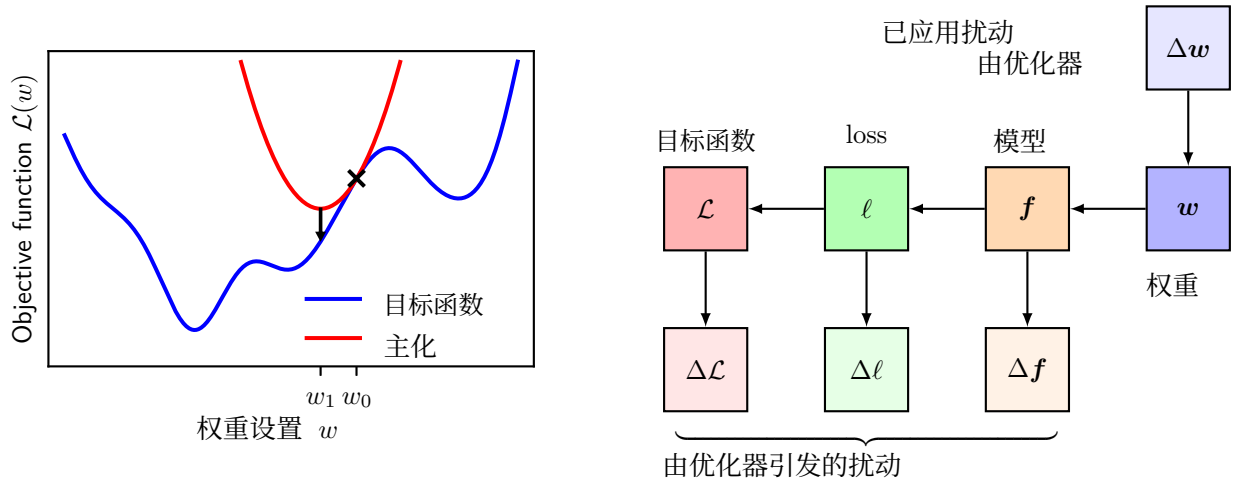

图 2：主化-最小化与扰动层级。左图展示了主化-最小化元算法（Lange，2016），这是一种通过最小化一系列上界（其中一个以红色显示）来降低目标函数（蓝色）的算法模式。上界也称为主化，必须与目标函数相切，才能保证元算法在一步内取得改进。**右图**展示了通用机器学习模型的扰动层级：优化器扰动权重，由此引发模型输出、单个训练样本上的损失，最终乃至整体目标函数的扰动。要对机器学习目标函数进行主化，就必须处理完整的扰动层级。

将神经网络优化景观描述为网络深度的函数。类似地，Yang & Hu（2021）试图理解宽度的作用，从而提出了他们的最大更新参数化。表 1 和表 2 对自动梯度下降与这些及其他框架进行了比较。

1.2 预备知识

给定一个向量 $\boldsymbol{v}$ in $\mathbb{R}^n$，我们需要用三种不同的方式来度量它的大小：
定义1（曼哈顿范数）曼哈顿范数 $\|\cdot\|_1$ 向量 $\boldsymbol{v}$ 的定义为 $\|\boldsymbol{v}\|_1 \coloneqq \sum_i |\boldsymbol{v}_i|$。**定义 2**（欧几里得范数）欧几里得范数 $\|\cdot\|_2$ 向量 $\boldsymbol{v}$ 的定义为 $\|\boldsymbol{v}\|_2 \coloneqq \sqrt{\sum_i \boldsymbol{v}_i^2}$。**定义3**（无穷范数）无穷范数 $\|\cdot\|_\infty$ 向量 $\boldsymbol{v}$ 的定义为 $\|\boldsymbol{v}\|_\infty \coloneqq \max_i |\boldsymbol{v}_i|$。

对于矩阵 $\boldsymbol{M}$，其属于 $\mathbb{R}^{m\times n}$，读者应注意它具有奇异值分解：
事实1 (SVD)每个矩阵 $\boldsymbol{M}$ 在 $\mathbb{R}^{m\times n}$ 中都存在一个奇异值分解 (SVD)，其形式为 $\boldsymbol{M} = \sum_{i=1}^{\min(m,n)} \sigma_i(\boldsymbol{M}) \cdot \boldsymbol{u}_i \boldsymbol{v}_i^\top$ 其中左奇异向量 $\{\boldsymbol{u}_i\}$ 是 $\mathbb{R}^m$ 中的标准正交向量，右奇异向量 $\{\boldsymbol{v}_i\}$ 是 $\mathbb{R}^m$ 中的标准正交向量，而奇异值 $\{\sigma_i(\boldsymbol{M})\}$ 是非负标量。

奇异值分解使我们能够通过两种不同的方式来衡量矩阵的大小：**定义4**（弗罗贝尼乌斯范数）弗罗贝尼乌斯范数 $\|\cdot\|_F$ 矩阵 $\boldsymbol{M}$ 的 $\|\boldsymbol{M}\|_F$ 为 $= \sqrt{\sum_i \sigma_i(\boldsymbol{M})^2}$。**定义5**（算子范数）算子范数 $\|\cdot\|_*$ 矩阵 $\boldsymbol{M}$ 的 $\|\boldsymbol{M}\|_*$ 为 $= \max_i \sigma_i(\boldsymbol{M})$。

算子范数 $\|\boldsymbol{M}\|_*$ 报告最大奇异值，而量 $\|\boldsymbol{M}\|_F/\sqrt{\min(m,n)}$ 报告奇异值均方根。最后，我们需要理解矩阵条件性的两个方面：
Defi定义 6（秩）矩阵的秩 计算非零奇异值的数量 .
定义 7（稳定秩）矩阵 $\boldsymbol{M}$ 的稳定秩 定义为 $\text{秩}_{\text{stable}}\,\boldsymbol{M} \coloneqq \|\boldsymbol{M}\|_F^2/\|\boldsymbol{M}\|_*^2$。

稳定秩提供了秩的一种近似，忽略了极小奇异值的存在。我们来看看两种极端情况。正交矩阵 $\boldsymbol{O} \in \mathbb{R}^{m\times n}$ 既是满秩矩阵，也是满稳定秩矩阵：$\text{rank}\ \boldsymbol{O} = \text{rank}_{\text{stable}}\,\boldsymbol{O} = \min(m,\ n)$。秩一矩阵 $\boldsymbol{P}$ 的稳定秩为 1，并满足 $\|\boldsymbol{P}\|_* = \|\boldsymbol{P}\|_F$。

理论	参考文献	处理损失函数 $\mathcal{L} \leftarrow \ell \leftarrow f$	非线性网络 $f \leftarrow w$
镜像下降	Nemirovsky 和 Yudin（1983）	✓	✗
高斯-牛顿法	Björck（1996年）	✓	✗
自然梯度下降	Amari（1998年）	✓	✗
神经切线核	Jacot 等（2018年）	✓	✗
深度相对信任	Bernstein 等（2020）	✗	✓
张量程序	Yang & Hu（2021）	✗	✓
自动梯度下降	本文	✓	✓

表 2：一阶优化理论中常用框架的比较。各框架在处理模型输出 $\boldsymbol{f}$ 与目标函数 $\mathcal{L}$之间的相互作用，以及权重 $\boldsymbol{w}$ 与模型输出 $\boldsymbol{f}$之间复杂的非线性相互作用方面各有不同。我们的框架能够处理这两个方面。

2 面向通用学习问题的主化-最小化

本节将建立一个框架，将主化-最小化元算法应用于机器学习中的一般优化问题。具体而言，本节将介绍一种新颖的函数展开技术。第 3 节将把这项技术应用于深度神经网络。所有证明均见附录 A。

给定一个机器学习模型和一组训练数据，我们的目标是最小化模型在训练数据上的平均误差。形式上，我们希望最小化以下函数：

定义 8（复合目标函数）考虑一个机器学习模型 $\boldsymbol{f}$，它将输入 $\boldsymbol{x}$ 和权重向量 $\boldsymbol{w}$ 映射为输出 $\boldsymbol{f}(\boldsymbol{x};\ \boldsymbol{w})$。给定数据 S 和凸损失函数 ℓ，目标函数 $\mathcal{L}(\boldsymbol{w})$ 定义如下：

$$\mathcal{L}(\boldsymbol{w}) := \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \ell(\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}),\boldsymbol{y}).$$

我们将此目标函数称为复合目标函数，因为损失函数 ℓ 与机器学习模型$\boldsymbol{f}$复合在一起。尽管损失函数本身是凸的，但由于机器学习模型具有非线性，整体复合目标函数通常是非凸的。常见的凸损失函数包括平方损失和交叉熵损失：

Exam例 1（平方损失）平方损失定义为：$\ell(\boldsymbol{f}(\boldsymbol{x};\ \boldsymbol{w}),\ \boldsymbol{y}) := \frac{1}{2d_L}\|\boldsymbol{f}(\boldsymbol{x};\ \boldsymbol{w}) - \boldsymbol{y}\|_2^2$。

例 2（交叉熵损失）交叉熵（*xent*）损失定义为：$\ell(\boldsymbol{f}(\boldsymbol{x}),\ \boldsymbol{y}) := -\log[\mathrm{softmax}(\boldsymbol{f}(\boldsymbol{x}))]^\top \boldsymbol{y}$，其中 softmax 函数定义为 $\mathrm{softmax}(\boldsymbol{f}(\boldsymbol{x})) := \exp\ \boldsymbol{f}(\boldsymbol{x})/\|\exp\ \boldsymbol{f}(\boldsymbol{x})\|_1$。

2.1 线性化误差的分解

一阶优化器利用当前迭代点处目标函数的线性化。要设计此类方法，我们必须了解这种线性化的有效范围。为此，我们推导出机器学习系统线性化误差的一个非常一般的分解。该结果以扰动层级的形式表述。具体而言，扰动机器学习模型的权重向量 $\boldsymbol{w} \to \boldsymbol{w} + \Delta\boldsymbol{w}$会引起模型输出 $\boldsymbol{f} \to \boldsymbol{f} + \Delta\boldsymbol{f}$、单个数据样本上的损失 $\ell \to \ell + \Delta\ell$ 以及最终整体目标函数 $\mathcal{L} \to \mathcal{L} + \Delta\mathcal{L}$ 的扰动。形式上，权重扰动 $\Delta\boldsymbol{w}$ 会引起：

$$\begin{aligned}
\Delta\boldsymbol{f}(\boldsymbol{x}) &:= \boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}+\Delta\boldsymbol{w}) - \boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}); && \text{（函数扰动）}\\
\Delta\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}) &:= \ell(\boldsymbol{f}(\boldsymbol{x})+\Delta\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}) - \ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}); && \text{(loss perturbation)}\\
\Delta L(\boldsymbol{w}) &:= \tfrac{1}{|\mathsf{S}|}\textstyle\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \Delta\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}). && \text{（目标函数扰动）}
\end{aligned}$$

我们采用了一种简洁的记法，有时会省略 $\boldsymbol{f}(\boldsymbol{x};\ \boldsymbol{w})$ 对 $\boldsymbol{w}$ 的依赖。通用机器学习模型和深度神经网络的扰动层次结构分别如图 2 和图 3 所示。目标函数扰动 $\Delta\mathcal{L}$ 的线性化误差可分解为：

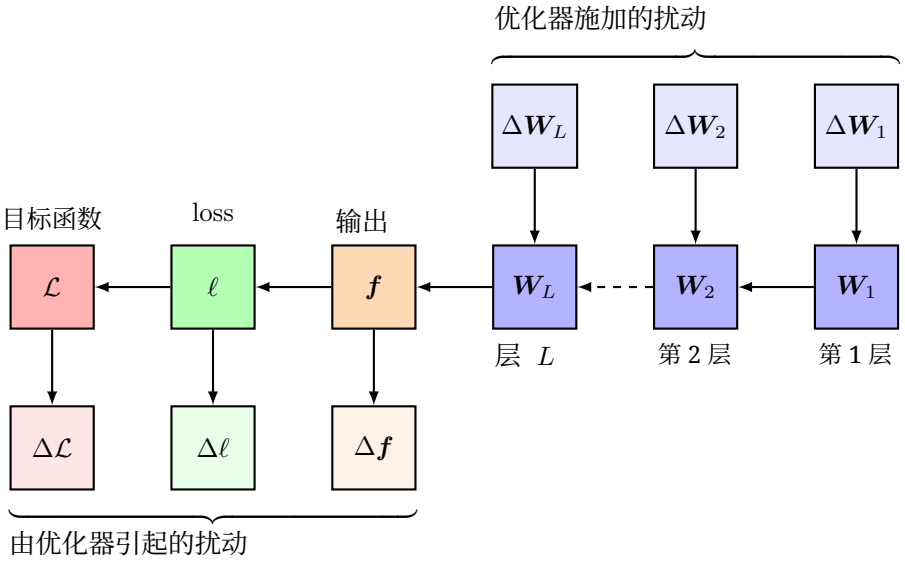

图3：深度神经网络的扰动层级。 训练神经网络时，优化器会以结构化的方式对权重施加扰动，即每个权重矩阵对应一个扰动矩阵 $\Delta \boldsymbol{W}_k$ $\boldsymbol{W}_k$。深度相对信任（Bernstein 等，2020）提供了一种工具，可用于理解这种形式的结构化权重扰动如何影响网络输出 $\boldsymbol{f}$。将深度相对信任与 Bregman散度（Bregman，1967年）结合起来，可以分析完整的扰动层级。

命题1 （线性化误差的分解）对于任意可微损失 ℓ 和任意可微机器学习模型 $\boldsymbol{f}$，目标函数 $\mathcal{L}$ 的线性化误差可分解如下：

$$\underbrace{\Delta \mathcal{L}(\boldsymbol{w}) - \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})^\top \Delta \boldsymbol{w}}_{\text{linearisation error of objective}} = \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathsf{S}} \nabla_{\boldsymbol{f}(\boldsymbol{x})} \ell(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y})^\top \underbrace{[\Delta \boldsymbol{f}(\boldsymbol{x}) - \nabla_{\boldsymbol{w}} \boldsymbol{f}(\boldsymbol{x}) \Delta \boldsymbol{w}]}_{\text{linearisation error of model}} + \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathsf{S}} \underbrace{\Delta \ell(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y}) - \nabla_{\boldsymbol{f}(\boldsymbol{x})} \ell(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y})^\top \Delta \boldsymbol{f}(\boldsymbol{x})}_{\text{linearisation error of loss}}.$$

换言之，目标函数的线性化误差可分解为两项。第一项取决于机器学习模型的线性化误差，第二项则取决于损失。这种分解仅依赖于可微性。对于凸损失，第二项可解释为 Bregman散度：**定义9** （损失的 Bregman散度）对于任意凸损失 ℓ：

$$\text{bregman}_{\ell(\cdot,\boldsymbol{y})}(\boldsymbol{f}(\boldsymbol{x}), \Delta \boldsymbol{f}(\boldsymbol{x})) := \Delta \ell(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y}) - \nabla_{\boldsymbol{f}(\boldsymbol{x})} \ell(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y})^\top \Delta \boldsymbol{f}(\boldsymbol{x}).$$

Bregman散度就是凸函数的线性化误差。两个重要的例子是：**引理1** （平方损失的 Bregman散度）当 ℓ 设为平方损失时，则：

$$\text{bregman}_{\ell(\cdot,\boldsymbol{y})}(\boldsymbol{f}(\boldsymbol{x}), \Delta \boldsymbol{f}(\boldsymbol{x})) = \tfrac{1}{2d_L} \|\Delta \boldsymbol{f}(\boldsymbol{x})\|_2^2.$$

引理2 （交叉熵损失的Bregman散度）当 ℓ 设为交叉熵损失，且若 $\boldsymbol{y}^\top \mathbf{1} = 1$，则：

$$\begin{aligned} \text{bregman}_{\ell(\cdot,\boldsymbol{y})}(\boldsymbol{f}(\boldsymbol{x}), \Delta \boldsymbol{f}(\boldsymbol{x})) &= D_{\text{KL}}\Big(\text{softmax}(\boldsymbol{f}(\boldsymbol{x})) \Big\| \text{softmax}(\boldsymbol{f}(\boldsymbol{x}) + \Delta \boldsymbol{f}(\boldsymbol{x}))\Big) \\ &\leqslant \tfrac{1}{2} \|\Delta \boldsymbol{f}(\boldsymbol{x})\|_\infty^2 + \mathcal{O}(\Delta \boldsymbol{f}^3). \end{aligned}$$

通过计算或界定其他凸损失的 Bregman散度，我们的方法也可以应用于这些损失。

2.2 函数展开与函数主化

在继续之前，我们先作一个简化假设。注意，命题1右侧的第一项是两个向量之间的高维内积。由于没有明显理由认为这两个向量应该对齐，我们假设它们的内积为零:

Assumption 1 （模型线性化误差的正交性）在与命题1相同的设定下:

$$\frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}}\nabla_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})^\top\underbrace{[\Delta\boldsymbol{f}(\boldsymbol{x})-\nabla_{\boldsymbol{w}}\boldsymbol{f}(\boldsymbol{x})\Delta\boldsymbol{w}]}_{\text{linearisation error of model}}=0.$$

尽管可以不采用这一假设（Bernstein，2022），但我们发现，加入该假设能简化分析，而且在实践中并未明显削弱最终算法的效果。无论如何，这一假设远比文献中常见的假设宽松得多；后者认为模型线性化误差本身为零（Pascanu & Bengio，2014；Lee 等，2019）： $[\Delta\boldsymbol{f}(\boldsymbol{x})-\nabla_{\boldsymbol{w}}\boldsymbol{f}(\boldsymbol{x})\Delta\boldsymbol{w}]=0$。

有了命题1和假设 1，我们现在可以介绍函数展开与主化：**定理1** （函数展开）考虑一个凸可微损失 ℓ 和一个可微机器学习模型 $\boldsymbol{f}$。在假设 1 下，相应的复合目标函数 $\mathcal{L}$ 可展开为:

$$\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w})=\underbrace{\mathcal{L}(\boldsymbol{w})+\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w}}_{\text{first-order Taylor series}}+\frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}}\operatorname{bregman}_{\ell(\cdot,\boldsymbol{y})}(\boldsymbol{f}(\boldsymbol{x}),\Delta\boldsymbol{f}(\boldsymbol{x})).$$

因此，扰动后的目标函数 $\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w})$ 可写成其一阶泰勒展开与模型输出上的 Bregman 散度之和，其中该散度在整个训练集上取平均。只需代入不同损失对应的 Bregman 散度，即可将这一结果直接应用于各种损失:

推论 1 （均方误差的函数展开）在假设 1 下，对于平方损失:

$$\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w})=\mathcal{L}(\boldsymbol{w})+\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w}+\frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}}\tfrac{1}{2d_L}\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_2^2.$$

推论2（函数　　主化适用于交叉熵损失）在假设 1 下，对于交叉熵损失，如果 $\boldsymbol{y}^\top\mathbf{1}=$　　1:

$$\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w})\leqslant\mathcal{L}(\boldsymbol{w})+\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w}+\frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}}\tfrac{1}{2}\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_\infty^2+\mathcal{O}(\Delta\boldsymbol{f}^3).$$

当函数扰动足够分散时，我们预期 $\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_\infty^2\approx\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_2^2/d_L$。在这种情况下，交叉熵损失的函数主化与均方误差的函数展开在二阶近似下相一致。虽然论文推导了平方损失的自动梯度下降，但这一观察证明了将其应用于交叉熵损失的合理性，ImageNet 实验就是一个例子。

2.3 恢复现有框架

我们简要观察到　　三个现有的优化框架可以从定理1中高效恢复　　:

镜像下降对于线性模型 $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{W}):=\boldsymbol{W}\boldsymbol{x}$，Bregman散度 $\operatorname{bregman}_{\ell(\cdot,\ \boldsymbol{y})}(\boldsymbol{f}(\boldsymbol{x}),\ \Delta\boldsymbol{f}(\boldsymbol{x}))$ 可以写作 $\operatorname{bregman}_{\ell(\cdot,\ \boldsymbol{y})}(\boldsymbol{W}\boldsymbol{x},\ \Delta\boldsymbol{W}\boldsymbol{x})$。这是权重扰动 $\Delta\boldsymbol{W}$ 的凸函数。将其代入定理1，并对 $\Delta\boldsymbol{W}$ 求最小值，是镜像下降的起点。

高斯-牛顿法 将线性化的函数扰动 $\Delta\boldsymbol{f}(\boldsymbol{x})\approx\nabla_{\boldsymbol{w}}\boldsymbol{f}(\boldsymbol{x})\Delta\boldsymbol{w}$ 代入推论 1，并针对 $\Delta\boldsymbol{w}$ 进行最小化，是高斯-牛顿法的起点。

自然梯度下降 将线性化的函数扰动 $\Delta\boldsymbol{f}(\boldsymbol{x})\approx\nabla_{\boldsymbol{w}}\boldsymbol{f}(\boldsymbol{x})\Delta\boldsymbol{w}$ 代入推论2，并对 $\Delta\boldsymbol{w}$ 求最小值，是自然梯度下降的起点。

3 深度学习问题的主化-最小化

本节将重点推导一种用于采用平方损失训练的深度全连接网络的优化器。交叉熵损失的推导方法类似。证明见附录 A。

定义10（全连接网络）一个全连接网络（全连接网络） $\boldsymbol{f}$ 深度为 L 的网络将输入 $\boldsymbol{x} \in \mathbb{R}^{d_0}$映射到输出 $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w}) \in \mathbb{R}^{d_L}$ 通过 L 矩阵乘法，并在其中穿插非线性函数 $\mathrm{ReLU}(z) := \max(0, z)$:

$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w}) := \boldsymbol{W}_L \circ (\mathrm{relu} \circ \boldsymbol{W}_{L-1}) \circ (\mathrm{relu} \circ \boldsymbol{W}_{L-2}) \circ \cdots \circ (\mathrm{relu} \circ \boldsymbol{W}_1 \boldsymbol{x}).$$

在此表达式中， $\boldsymbol{w}$ 表示矩阵元组 $\boldsymbol{w} = (\boldsymbol{W}_1, ..., \boldsymbol{W}_L)$，其中第 k 个矩阵 $\boldsymbol{W}_k$ 属于 $\mathbb{R}^{d_k \times d_{k-1}}$。在下文中，我们会发现以下维度缩放尤为方便:

处方 1 (维度缩放**)** 对于 $\eta > 0$，数据 $(\boldsymbol{x}, \boldsymbol{y})$，权重 $\boldsymbol{W}_k$ 和更新量 $\Delta \boldsymbol{W}_k$ 应遵循:

$$\begin{aligned}
\|\boldsymbol{x}\|_2 &= \sqrt{d_0}; && \text{(input scaling)} \\
\|\boldsymbol{W}_k\|_* &= \sqrt{d_k / d_{k-1}} \qquad \text{for all } k = 1, ..., L; && \text{(weight scaling)} \\
\|\Delta \boldsymbol{W}_k\|_* &= \sqrt{d_k / d_{k-1}} \cdot \tfrac{\eta}{L} \qquad \text{for all } k = 1, ..., L; && \text{(update scaling)} \\
\|\boldsymbol{y}\|_2 &= \sqrt{d_L}. && \text{(target scaling)}
\end{aligned}$$

虽然不采用处方 1 也能推导出结果，但这些缩放大大简化了我们的公式。其原因之一是，在处方 1 下，我们具有望远镜性质 $\prod_{k=1}^{L} \|\boldsymbol{W}_k\|_* = \sqrt{d_L / d_0}$。为了具体说明这一点如何发挥作用，请看下面关于网络输出范数的界:

引理 3(输出　　t 界）全连接网络的输出范数 $\boldsymbol{f}$ 满足以下界:

$$\|\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w})\|_2 \leqslant \left[\prod_{k=1}^{L} \|\boldsymbol{W}_k\|_* \right] \times \|\boldsymbol{x}\|_2 = \sqrt{d_L} \text{ under Prescription 1.}$$

因此，在处方 1下，该界很简单。此外，使用单个参数η 对更新进行缩放，将求解优化器的问题简化为一个单参数问题。为了解这可能如何使问题变得更容易，请考虑下面这个将权重扰动与函数扰动联系起来的引理:

引理 4 （深度相对信任）当通过 $\Delta \boldsymbol{w} = (\Delta \boldsymbol{W}_1, ..., \Delta \boldsymbol{W}_L)$调整全连接网络$\boldsymbol{f}$的权重 $\boldsymbol{w} = (\boldsymbol{W}_1, ..., \boldsymbol{W}_L)$ 时，所诱导的函数扰动 $\Delta \boldsymbol{f}(\boldsymbol{x}) := \boldsymbol{f}(\boldsymbol{x};\ \boldsymbol{w} + \Delta \boldsymbol{w}) - \boldsymbol{f}(\boldsymbol{x};\ \boldsymbol{w})$满足:

$$\|\Delta \boldsymbol{f}(\boldsymbol{x})\|_2 \leqslant \left[\prod_{k=1}^{L} \|\boldsymbol{W}_k\|_* \right] \times \|\boldsymbol{x}\|_2 \times \left[\prod_{k=1}^{L} \left(1 + \frac{\|\Delta \boldsymbol{W}_k\|_*}{\|\boldsymbol{W}_k\|_*} \right) - 1 \right] \leqslant \sqrt{d_L} \times (\exp \eta - 1) \text{ under Prescription 1.}$$

因此，在处方 1 下，单个参数 η 直接控制函数扰动的幅度。

在实际中落实处方 1 时，数据 $(\boldsymbol{x}, \boldsymbol{y})$ 的范数可以通过预处理来设定，更新 $\Delta \boldsymbol{W}_k$ 的范数可以通过优化算法来设定，而权重矩阵 $\boldsymbol{W}_k$ 的范数则可以通过选择初始化方式来设定。虽然 $\|\boldsymbol{W}_k\|_*$ 在训练过程中可能会发生漂移，但这种变化的幅度受到 Weyl（1912）的奇异值不等式限制。具体来说，经过一步后，扰动后的算子范数 $\|\boldsymbol{W}_k + \Delta \boldsymbol{W}_K\|_*$ 被夹在 $(1 - \eta/L) \cdot \|\boldsymbol{W}_k\|_* \leqslant \|\boldsymbol{W}_k + \Delta \boldsymbol{W}_K\|_* \leqslant (1 + \eta/L) \cdot \|\boldsymbol{W}_k\|_*$之间。

3.1 推导自动梯度下降

有了函数主化和深度相对信任，我们便可以对深度网络目标函数进行主化：**引理 5** （指数上界）对于采用平方损失的全连接网络，在假设 1 和处方 1 的条件下:

$$\mathcal{L}(\boldsymbol{w} + \Delta \boldsymbol{w}) \leqslant \mathcal{L}(\boldsymbol{w}) + \frac{\eta}{L} \sum_{k=1}^{L} \left[\sqrt{d_k / d_{k-1}} \times \mathrm{tr} \frac{\Delta \boldsymbol{W}_k^\top \nabla_{\boldsymbol{W}_k} \mathcal{L}}{\|\Delta \boldsymbol{W}_k\|_*} \right] + \tfrac{1}{2} (\exp \eta - 1)^2.$$

```
def initialise_weights():
    for layer k in {1, ..., L}:
        W_k ~ UNIFORM(orthogonal(d_k, d_{k-1}))        # sample a semi-orthogonal matrix
        W_k ← W_k · sqrt(d_k / d_{k-1})                # rescale its singular values
def update_weights():
    G ← (1/L) Σ_{l=1}^{L} ||∇_{W_k} L||_F · sqrt(d_k / d_{k-1})    # get gradient summary
    η ← log((1 + sqrt(1 + 4G)) / 2)                    # set automatic learning rate
    for layer k in {1, ..., L}:
        W_k ← W_k − (η / L) · (∇_{W_k} L / ||∇_{W_k} L||_F) · sqrt(d_k / d_{k-1})    # update weights
```

算法1：自动梯度下降。矩阵 $\boldsymbol{W}_k$ in $\mathbb{R}^{d_k \times d_{k-1}}$ 是第 k 层的权重矩阵。梯度 $\nabla_{\boldsymbol{W}_k}\mathcal{L}$ 是相对于在训练样本小批次 B 上计算的目标函数 $\mathcal{L}$ 而言的。

请注意，主化仅取决于标量 η 的大小，以及扰动矩阵 $\Delta\boldsymbol{W}_k$ 与梯度矩阵 $\nabla_{\boldsymbol{W}_k}\mathcal{L}$ 之间某种角度概念 tr $\Delta\boldsymbol{W}_k^\top \nabla_{\boldsymbol{W}_k}\mathcal{L}/\|\Delta\boldsymbol{W}_k\|_*$。为了推导出一个优化器，接下来我们希望针对 η 和这个角度最小化该主化。首先，我们引入一个额外的假设和一个额外的定义：

假设2（梯度条件）在所有层 $k = 1, ..., L$ 上，梯度均满足 $\text{rank}_{\text{stable}}\, \nabla_{\boldsymbol{W}_k}\mathcal{L} = 1$。

这项假设意味着，第 k 层的梯度的弗罗贝尼乌斯范数 $\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_F$ 与算子范数 $\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_*$ 相等。为什么这是一个合理的假设，并非一目了然。毕竟，梯度是 $|\mathsf{S}|$ 个秩一矩阵之和：$\nabla_{\boldsymbol{W}_k}\mathcal{L} = \frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \nabla_{\boldsymbol{h}_k}\ell(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y}) \otimes \boldsymbol{h}_{k-1}$，其中 $\boldsymbol{h}_{k-1}(\boldsymbol{x})$ 和 $\boldsymbol{h}_k(\boldsymbol{x})$ 分别表示第 k 层权重矩阵 $\boldsymbol{W}_k$ 的输入和输出，而 $\otimes$ 表示外积。因此，直观地看，人们可能会预期梯度 $\nabla_{\boldsymbol{W}_k}\mathcal{L}$ 的稳定秩为 $\min(d_k, d_{k-1}, |\mathsf{S}|)$。但事实证明，这在实践中是一个合理的假设（Yang 与 Hu，2021；Yang 等，2021）。接下来给出定义：

定义 11（梯度摘要）在权重设置 $\boldsymbol{w}$ 下，梯度摘要 G 由下式给出：

$$G := \frac{1}{L}\sum_{k=1}^{L}\sqrt{d_k/d_{k-1}}\cdot\|\nabla_{\boldsymbol{W}_k}\mathcal{L}(\boldsymbol{w})\|_F.$$

梯度摘要是各层梯度范数的加权平均值。可以将其视为一种衡量梯度大小的方法，同时考虑到不同层的权重矩阵可能处于不同的尺度。这与 Philipp 等人（2017 年）的梯度尺度系数概念有关。

我们现在拥有一切 ing 我们需要通过主化-最小化原理推导自动梯度下降 e:

定理 2（自动梯度下降）对于深度全连接网络，在假设 1 和 2 以及处方 1 下，引理 5 中给出的平方损失的主化通过设定以下内容而最小化：

$$\eta = \log\frac{1+\sqrt{1+4G}}{2}, \qquad \Delta\boldsymbol{W}_k = -\frac{\eta}{L}\cdot\sqrt{d_k/d_{k-1}}\cdot\frac{\nabla_{\boldsymbol{W}_k}\mathcal{L}}{\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_F}, \qquad \text{for all layers } k = 1, ..., L.$$

我们在算法1中给出了该定理的伪代码，并在附录B中提供了 PyTorch 实现。自动梯度下降通过基于明确算法原理的简单推导，统一了文献中出现的各种启发式和理论思想：

• 相对更新量。 更新量会根据其所作用的权重矩阵的范数进行缩放——前提是权重矩阵按照处方 1 进行缩放。You 等人（2017）提出了这种缩放方式，随后 Carbonnelle 和 Vleeschouwer（2019）以及 Bernstein 等（2020）对其进行了进一步研究。有证据表明，神经科学中可能存在这种相对突触更新（Loewenstein 等，2011）。

• 深度缩放。 Bernstein 等人（2020）基于通过深度相对信任进行的分析，从理论角度提出，对于深度为 L 的网络，将扰动强度按 $1/L$ 进行缩放。

• 宽度缩放。 d_k 和 d_{k-1} 的维度因子与 Yang 和 Hu（2021）提出的最大更新参数化密切相关，该参数化旨在确保超参数迁移能够跨越网络宽度实现。

• 梯度裁剪。 更新量对梯度汇总的对数依赖性可以被视为一种自动形式的自适应梯度裁剪 （Brock 等人，2021）——这是一种当梯度幅度超过由超参数设定的某个阈值时对梯度进行裁剪的技术。

3.2 收敛分析

本节给出自动梯度下降的理论收敛速率。尽管分析的基本思路在优化理论中较为标准，但其细节仍可能因对深度网络优化性质的详细刻画而颇具意义。例如，我们提出了一个针对深度网络算子结构定制的新型 Polyak- Łojasiewicz 不等式。我们从两个观察结果开始：

引理 6 （目标函数有界）对于平方损失，目标函数有如下界：

$$\mathcal{L}(\boldsymbol{w}) \leqslant \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{\|\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})\|_2^2 + \|\boldsymbol{y}\|_2^2}{2d_L} \leqslant 1 \text{ under Prescription 1.}$$

引理 7 （梯度有界）对于平方损失，第 k 层的梯度范数有如下界：

$$\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_F \leqslant \frac{\prod_{l=1}^{L}\|\boldsymbol{W}_l\|_*}{\|\boldsymbol{W}_k\|_*} \cdot \sqrt{\frac{2\mathcal{L}(\boldsymbol{w})}{d_L}} \cdot \sqrt{\frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}}\|\boldsymbol{x}\|_2^2} \leqslant \sqrt{2\cdot\frac{d_{k-1}}{d_k}} \text{ under Prescription 1.}$$

这些结果有助于我们证明，自动梯度下降会收敛到梯度为零的点：**引理 8** （收敛到临界点的速率）考虑一个使用自动梯度下降（定理 2）和平方损失训练了 T 次迭代的全连接网络。令 G_t 表示第 $t \leqslant T$ 步的梯度汇总（定义 11）。在假设 1 和 2 以及处方 1 下，自动梯度下降以如下速率收敛：

$$\min_{t\in\{1,\ldots,T\}} G_t^2 \leqslant \frac{11}{T}.$$

加入一个额外假设后，可将此引理转化为收敛到全局最小值的速率：**假设 3** （深度 Polyak-Łojasiewicz 不等式）对于某个 $\alpha > 0$，梯度范数有如下下界：

$$\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_F \geqslant \alpha \times \frac{\prod_{l=1}^{L}\|\boldsymbol{W}_l\|_*}{\|\boldsymbol{W}_k\|_*} \cdot \sqrt{\frac{2\mathcal{L}(\boldsymbol{w})}{d_L}} \cdot \sqrt{\frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}}\|\boldsymbol{x}\|_2^2} = \alpha \times \sqrt{2\cdot\mathcal{L}(\boldsymbol{w})\cdot\frac{d_{k-1}}{d_k}} \text{ under Prescription 1.}$$

这个下界与引理 7 中上界的结构相呼应。参数 α 刻画了权重中的小奇异值以及未激活的 ReLU 单元对梯度的衰减程度。尽管 Polyak-Łojasiewicz 不等式在文献中很常见（Liu 等，2022），我们的假设新颖之处在于它关注网络的算子结构。假设 3 导出以下定理：

定理 3 （收敛到全局极小值的速率）对于与引理 8 相同设置但增加了假设 3 的自动梯度下降（定理 2），第 T 步的均方误差目标函数满足：

$$\mathcal{L}(\boldsymbol{w}_T) \leqslant \frac{1}{\alpha^2} \times \frac{6}{T}.$$

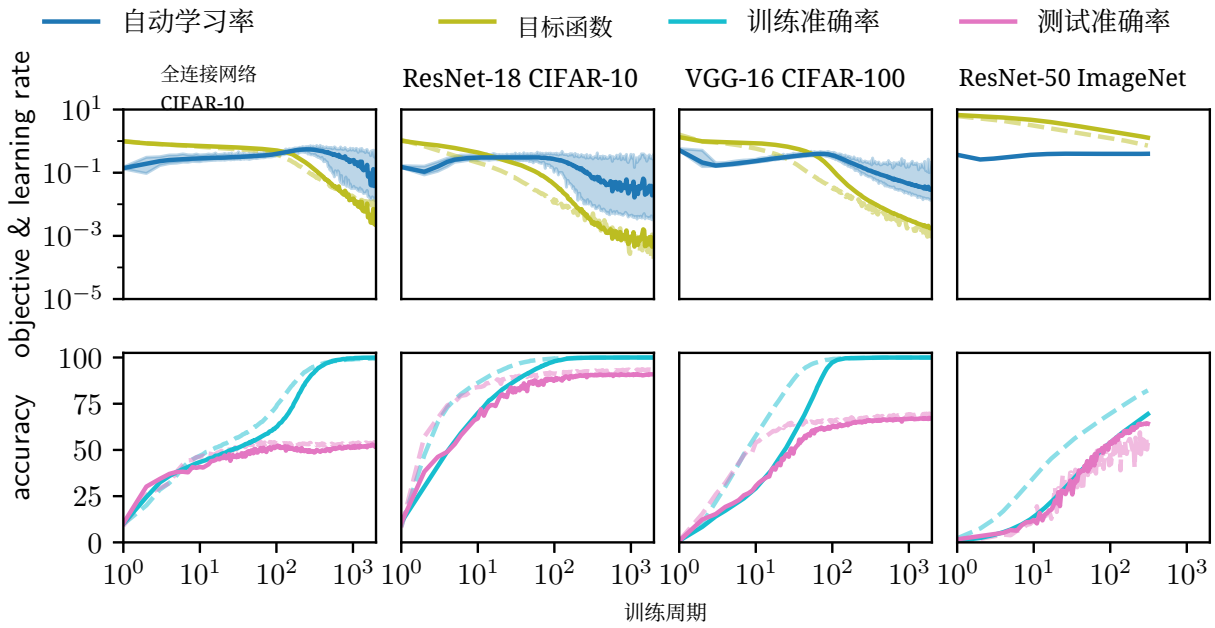

图4：在一系列架构和数据集上对自动梯度下降进行基准测试。实线表示自动梯度下降，浅色虚线表示经过调优的 Adam；但对于 ImageNet，虚线表示使用固定学习率 0.1 的随机梯度下降。ImageNet 使用批量大小为 1024 的交叉熵损失。其他实验使用批量大小为 128 的平方损失。**顶行**绘制了自动学习率（正文中的 η）和目标值。除前三个图中的平均值外，还包含每个训练轮次的最大学习率和最小学习率。**底行**显示了训练准确率和测试准确率。

3.3 实验

我们的实验目标有两个方面。首先，我们希望在广泛的网络架构和数据集上测试自动梯度下降（AGD，算法1），以检验它是否确实有效。具体而言，我们在 CIFAR-10、CIFAR-100（Krizhevsky，2009）和ImageNet（Deng 等，2009，ILSVRC2012）数据集上，使用标准数据增强，对全连接网络（FCNs，定义10）以及VGG 风格（Simonyan 与Zisserman，2015）和 ResNet 风格（He 等，2015）的卷积神经网络进行了 AGD 测试。其次，为了了解AGD 除现有方法之外还能提供什么，我们希望将 AGD 与经过调优的 Adam 和随机梯度下降基线进行比较，同时也与使用默认超参数运行的 Adam 和随机梯度下降进行比较。

为了使自动梯度下降能够与卷积层配合使用，我们采用了逐子矩阵归一化方案。具体而言，对于滤波器大小为 $\mathtt{k_x} \times \mathtt{k_y}$ 的卷积张量，我们分别对 $\mathtt{k_x} \times \mathtt{k_y}$ 个维度为 $\mathtt{channels_{in}} \times \mathtt{channels_{out}}$ 的子矩阵实施归一化。由于自动梯度下降目前尚不支持 batchnorm 中的偏置或仿射参数，我们在所有架构中禁用了这些参数。为了至少在初始化时遵循处方1，自动梯度下降从均匀半正交的初始权重矩阵开始，并将其按 $\sqrt{\mathtt{fan_in}/\mathtt{fan_out}}$ 的因子重新缩放。Adam 和随机梯度下降基线使用 PyTorch 的默认初始化。附录B给出了一个反映这些细节的自动梯度下降 PyTorch 实现。除 ImageNet 外，所有实验均使用平方损失；ImageNet使用交叉熵损失。对于类别数量较多的数据集，研究发现交叉熵损失优于平方损失（Demirkaya 等，2020；Hui 和 Belkin，2021）。

我们的实验结果分布在五张图中：

- 图 1 展示了我们研究结果的一些亮点：首先，自动梯度下降能够训练出使用默认超参数的Adam和随机梯度下降无法训练的网络。其次，对于CIFAR-10 上的 ResNet-18，自动梯度下降达到了与 Adam 和随机梯度下降经过最佳调优后的性能相当的水平。第三，自动梯度下降可以扩展到 ImageNet。

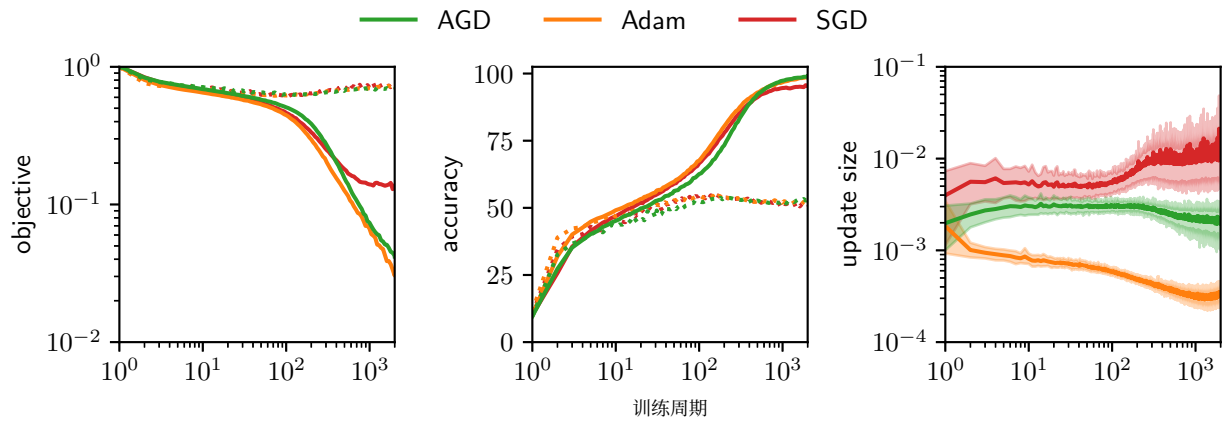

图5：比较自动梯度下降与调优后的 Adam 和随机梯度下降。使用平方损失在CIFAR-10上训练了一个八层全连接网络。虚线表示测试性能，实线表示训练性能。**左图**显示目标值：自动梯度下降和 Adam达到的训练目标值小于随机梯度下降。**中图**显示训练和测试准确率。**右图**显示在各层上取平均的相对更新幅度：$\frac{1}{L}\sum_{k=1}^{L}\|\Delta \boldsymbol{W}_k\|_F/\|\boldsymbol{W}_k\|_F$。我们绘制了一个训练轮次内的最大值、最小值和平均值。

- 图4展示了我们实验的广泛性：从在 CIFAR-10 上训练一个16层全连接网络，到在ImageNet上训练 ResNet-50。Adam 的学习率在对数网格 $\{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$ 上进行调优；而对于ImageNet，我们对随机梯度下降使用了0.1的默认学习率，且未进行任何手动衰减。AGD和 Adam 在深度16、宽度512的全连接网络上表现几乎相当：AGD 的测试准确率为52.7%，而 Adam为53.5%。对于 CIFAR-10上的 ResNet-18，Adam达到了92.9%的测试准确率，而 AGD 为91.2%。在这一基准测试中，经过充分调优的随机梯度下降，结合学习率调度、权重衰减、交叉熵损失以及偏置和仿射参数，可达到93.0%的测试准确率（Liu，2017）。对于 CIFAR-100 上的 VGG-16，AGD 达到了67.4%的测试准确率，而Adam 为69.7%。最后，在ImageNet上，AGD经过350个训练周期后达到了65.5%的 Top-1测试准确率。

- 图5比较了自动梯度下降、Adam 和随机梯度下降在训练一个宽度为 256的八层全连接网络时的表现。Adam 和随机梯度下降的学习率在对数网格 $\{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$上进行了调节。Adam的最优学习率为 10^{-4}，比随机梯度下降的最优学习率10^{-1}小三个数量级。随机梯度下降未能达到像 Adam 或自动梯度下降那么低的目标值。

- 图6显示，自动梯度下降可以训练宽度从64到2048、深度从2到32的全连接网络；图7则显示，自动梯度下降成功训练了一个四层全连接网络，其小批量大小从32到4096不等。

4 讨论

本文提出了一个用于推导非凸复合目标函数优化算法的新框架，这类目标函数在机器学习领域，尤其是深度学习这一子领域中十分常见。我们提出的确实是一个框架：只需写出新损失函数的 Bregman散度，或新机器学习模型的架构扰动界，即可将其应用于该损失函数或模型。该框架恰当地置于现有框架的背景下，例如主化-最小化元算法、镜像下降和自然梯度下降。

近期论文提出了一种超参数迁移 范式：先调整一个小型网络，再将得到的超参数迁移到更大型的网络（Yang 等，2021；Bernstein，2022）。本文的方法和结果表明，还可以采用一种更强的超参数消除范式：通过详细分析机器学习系统中不同组件的结构及其相互作用，我们或许可以——即便无法彻底禁绝超参数——至少减少其数量并降低其不透明性。

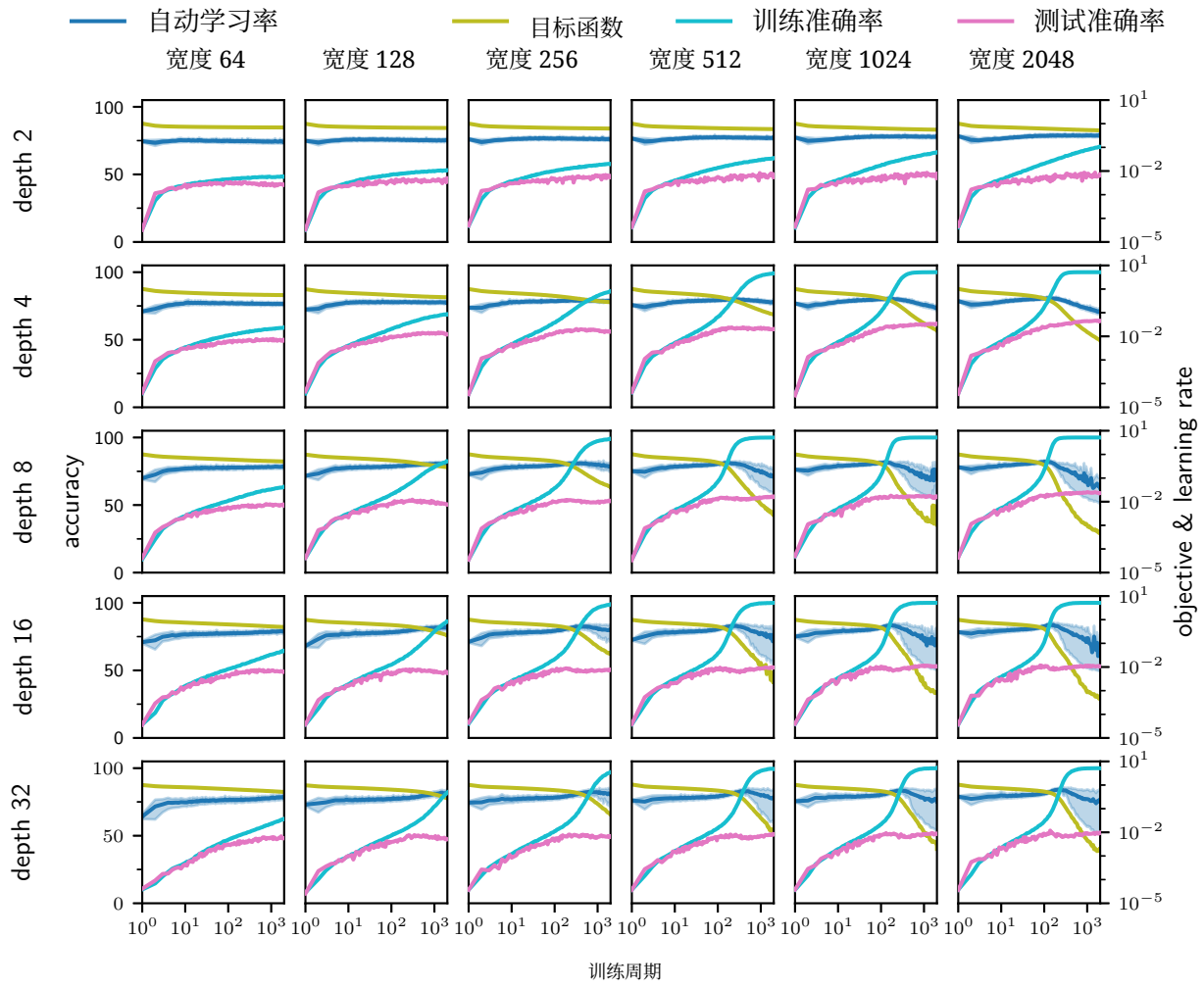

图6：在不同宽度和深度的网络上对自动梯度下降进行基准测试。 我们使用平方损失和 128 的小批量大小，在 CIFAR-10 上训练了全连接网络。深度范围为 2 到 32，宽度范围为 64 到 2048，均按 2 的幂次递增。从训练性能来看，宽度越大效果始终越好，而深度为 8 和 16 的网络优于深度为 32 的网络。从测试准确率来看，深度为 4、宽度为 2048 的网络表现最佳，达到 63.7%。测试性能最差的是最小的网络，深度为 2、宽度为 64，准确率为 42.55%。较大的网络在训练过程中呈现出两个明显不同的阶段：自动学习率缓慢上升，同时目标函数缓慢下降；随后自动学习率和目标函数迅速下降。第二阶段通常与训练准确率达到 100% 同时发生。关于 256 宽度、8 层全连接网络中 Adam、随机梯度下降和自动梯度下降的比较，请参见图5。

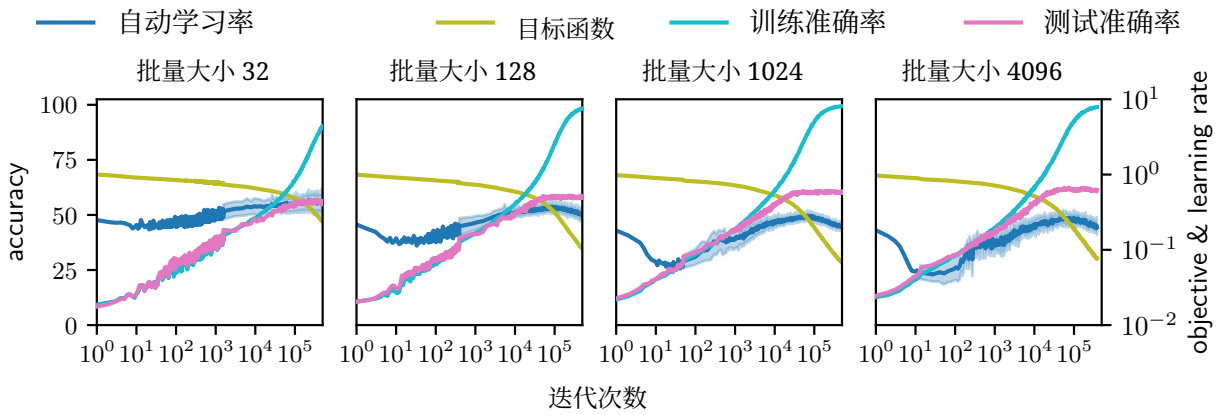

图7：在不同小批量大小下对自动梯度下降进行基准测试。 我们在 CIFAR-10 上训练了四层全连接网络。小批量大小范围为32至4096。测试准确率通常会随着小批量大小的增加而提高：按照小批量大小递增的顺序，最终测试准确率分别为55.0%、58.0%、60.0%和 59.8%。自动学习率似乎最初有所下降，并且对于较大的小批量大小，这一现象更加明显。在第一个训练轮次期间，每次迭代都会计算指标，此后每个训练轮次计算一次——这解释了图中可见的曲线折点。

这项研究的主要成果是自动梯度下降（自动梯度下降），算法1中给出了伪代码，附录B中给出了 PyTorch代码。我们发现自动梯度下降确实很有用，并相信它可以通过帮助实现通用机器学习工作流的自动化来补充自动微分。

导向自动梯度下降的分析是基础的：我们利用线性代数中的基本概念，例如矩阵范数和向量范数，并使用一些简单的界，例如向量—向量求和所用的三角不等式，以及矩阵—向量乘积所用的算子范数界。该分析是非渐近的：它不依赖于将维度取到无穷大；同时也是确定性的：不涉及随机矩阵理论。我们相信，这种分析的易理解性可以使本文成为未来研究发展的良好起点。

未来工作方向 在此我们列出一些有前景的理论和实践研究方向。我们正在开发代码库中探索其中一些想法：`https://github.com/C1510/agd_exp`。

• 随机优化。自动梯度下降是在全批量优化设置下推导出来的，但该算法是在小批量设置下进行实验评估的。尝试扩展我们的理论和实践方法，以更忠实地处理随机优化，将会很有意义。

• 更多架构。 自动梯度下降针对全连接网络推导而来，并以启发式方式扩展到卷积网络。我们希望将这些方法扩展到更多样化的架构，例如Transformer（Vaswani 等，2017），以及偏置等架构组件。由于大多数神经网络在某种意义上都类似于全连接网络，因为它们都只是深层复合算子，我们预计，本文所介绍的自动梯度下降的大部分结构都能够延续下去。

• 正则化。本文纯粹探讨深度神经网络的优化结构，几乎未考虑泛化或正则化。未来的工作可以研究自动梯度下降的理论和实践正则化方案。尝试在不引入超参数的情况下实现这一点将会很有趣，不过我们怀疑，至少在正则化方面，可能有必要使用一个超参数。

• 加速。 我们在一些初步实验中发现，通过使用增益超参数略微增大自动梯度下降的更新幅度，或引入动量超参数，都可以实现更快的收敛。我们强调，本文中的实验均未使用此类超参数。不过，这些观察结果或许能为未来改进自动梯度下降提供一个有价值的起点。

- **算子扰动理论。** 本文的部分灵感来源于将算子扰动理论应用于深度学习的想法。虽然扰动理论在线性算子的背景下已有深入研究（Weyl，1912；Kato，1966；Stewart，2006），但在深度学习中，我们关注的是非线性复合算子。尝试将扰动理论中的结果进一步扩展到深度神经网络，或许会很有意思。人们可以设想对不同神经网络构件的扰动结构进行分类，并利用类似于深度相对信任（引理 4）的结果来描述它们如何复合。

致谢

作者感谢 MIT SuperCloud、Oxford Hydra、NVIDIA 和 Virgile Richard 提供图形处理器。感谢 Greg Yang 和 Jamie Simon 提供有益的讨论。我们正在与 Greg 和 Jamie 撰写一篇论文，以阐释 μP（Yang 与 Hu，2021）与算子范数之间的关系。

参考文献

Naman Agarwal、Zeyuan Allen Zhu、Brian Bullins、EladHazan 和 Tengyu Ma。寻找比梯度下降更快的近似局部极小值。计算理论研讨会，2016 年。第 3 页引用。

Naman Agarwal、BrianBullins 和 Elad Hazan。《线性时间机器学习二阶随机优化》。《机器学习研究杂志》，2017 年。引用于第 3 页。

Rami Al-Rfou,Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, AnatolyBelikov, AlexanderBelopolsky等人。Theano：用于快速计算数学表达式的 Python 框架。*arXiv:1605.02688*，2016 年。引用见第 2 页。

Shun-ichi Amari。自然梯度在学习中高效运作。《神经计算》，1998年。引用见第 3 和 5 页。

Navid Azizan 和 Babak Hassibi。随机梯度/镜像下降：极小极大最优性与隐式正则化。载于国际学习表征会议，2019。引用见第 3 页。

Jeremy Bernstein。《神经元网络中的优化与泛化》。博士论文，加州理工学院，2022。引用见第 7 和 12 页。

Jeremy Bernstein、Arash Vahdat、Yisong Yue 和 Ming-Yu Liu。关于两个神经网络之间的距离与学习稳定性。载于神经信息处理系统，2020。引用见第 2、3、5、6 和 10 页。

Åke Björck。《最小二乘问题的数值方法》。工业与应用数学学会，1996年。引用见第 3 和 5 页。

Léon Bottou、Frank E. Curtis 和 Jorge Nocedal。大规模机器学习的优化方法。《*SIAM*评论》，2018年。参见第 3 页。

Lev M. Bregman。求凸集公共点的松弛法及其在凸规划问题求解中的应用。苏联计算数学与数学物理，1967年。参见第 2 页和第 6 页。

Andy Brock、Soham De、Samuel L. Smith 和 Karen Simonyan。无需归一化的高性能大规模图像识别。发表于国际机器学习会议，2021年。参见第 10 页。

Simon Carbonnelle 和 Christophe De Vleeschouwer。层旋转：深度网络泛化能力的一个出人意料的简单指标？发表于*ICML* 深度学习现象识别与理解研讨会，2019。参见第 10 页。

Jeremy Cohen、Simran Kaur、Yuanzhi Li、J. Zico Kolter 和 Ameet Talwalkar。神经网络上的梯度下降通常发生在稳定性边缘。在国际学习表征会议，2021。引用见第 3 页。

Ahmet Demirkaya、Jiasi Chen和 Samet Oymak。探索损失函数在多类别分类中的作用。信息科学与系统会议，2020。引用于第 11 页。

Jia Deng、Wei Dong、RichardSocher、Li-Jia Li、KaiLi和 Li Fei-Fei。《ImageNet：一个大规模分层图像数据库》。发表于计算机视觉与模式识别，2009 年。引用于第 11 页。

Inderjit S. Dhillon 和 Joel A. Tropp。Bregman散度下的矩阵逼近问题。*SIAM* 矩阵分析与应用杂志，2008年。第 2 页引用。

Alexander R. Farhang、Jeremy Bernstein、Kushal Tirumala、Yang Liu 和 Yisong Yue。通过控制归一化间隔研究泛化。发表于 国际机器学习会议，2022 年。引用于第 2 页。

Ian Goodfellow、Yoshua Bengio 和 Aaron Courville。《深度学习》。麻省理工学院出版社，2016。引自第 2 页。

Kaiming He、X. Zhang、Shaoqing Ren 和 Jian Sun。用于图像识别的深度残差学习。计算机视觉与模式识别，2015。引用于第 11 页。

Peter Henderson、Riashat Islam、Philip Bachman、Joelle Pineau、Doina Precup 和 David Meger。《真正重要的深度强化学习》。发表于*AAAI*人工智能会议，2018年。引自第 2 页。

Like Hui和MikhailBelkin。使用平方损失与交叉熵训练的神经架构在分类任务中的评估。载于 国际学习表征会议，2021 年。引用见第 11 页。

Arthur Jacot、Franck Gabriel 和 Clement Hongler。《神经切线核：神经网络中的收敛与泛化》。载于神经信息处理系统，2018年。引于第5页。

Yiding Jiang、Behnam Neyshabur、Hossein Mobahi、Dilip Krishnan 和 Samy Bengio。《出色的泛化度量及其寻找之处》。发表于国际学习表征会议，2020。引用见第 2 页。

Tosio Kato。《线性算子的扰动理论》。Springer，1966 年。第 15 页引用。

Diederik P. Kingma 和 Jimmy Ba。Adam：一种用于随机优化的方法。载于 国际学习表征会议，2015 年。第 3 页引用。

Alex Krizhevsky。《从微小图像中学习多层特征》。技术报告，多伦多大学，2009 年。引用见第 11 页。

Kenneth Lange。《*MM*优化算法》。工业与应用数学学会，2016 年。引用于第 3 页和第 4 页。

Jaehoon Lee、Lechao Xiao、Samuel Schoenholz、Yasaman Bahri、Roman Novak、Jascha Sohl-Dickstein 和 Jeffrey Pennington。任意深度的宽神经网络在梯度下降下演化为线性模型。发表于神经信息处理系统，2019。引用见第 7 页。

Chaoyue Liu、Libin Zhu 和 Mikhail Belkin。过参数化非线性系统和神经网络中的损失景观与优化。应用与计算调和分析，2022。引用见第 10 页。

KuangLiu。使用PyTorch 训练 CIFAR-10。 `https://github.com/kuangliu/pytorch-cifar`，2017 年。引用于第 12 页。

Yonatan Loewenstein、AnneroseKuras和SimonRumpel。乘法动力学是体内新皮层中棘突大小对数正态分布出现的基础。《神经科学杂志》，2011 年。引自第 10 页。

Mario Lucic、Karol Kurach、Marcin Michalski、Sylvain Gelly和Olivier Bousquet。《生成对抗网络是否生而平等？一项大规模研究》。载于 神经信息处理系统，2017年。引用于第 2 页。

Arkady S. Nemirovsky 和 David B. Yudin。《优化中的问题复杂度与方法效率》。Wiley，1983 年。引用于第 3 页和第 5 页。

Yurii Nesterov 和 Boris Polyak。牛顿法的三次正则化及其全局性能。*Mathematical Programming*，2006。引文见第 3 页。

Jorge Nocedal 和 Stephen J. Wright。《数值优化》。Springer，1999。引用于第 3 页。

Francesco Orabona和 Ashok Cutkosky。《ICML2020 参数无关在线优化教程》，2020 年。第 2 页引用。

Razvan Pascanu 和 Yoshua Bengio。《重新审视深度网络的自然梯度》。发表于国际学习表征会议，2014。引用见第 3 页和第 7 页。

Adam Paszke、Sam Gross、FranciscoMassa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、LucaAntiga 等。PyTorch：命令式风格、高性能深度学习库。载于 神经信息处理系统，2019 年。第 2 页和第 23 页引用。

George Philipp、Dawn Xiaodong Song 和 Jaime G. Carbonell。揭开梯度爆炸问题的神秘面纱。*arXiv:1712.05577*，2017。第 9 页引用。

David E. Rumelhart、Geoffrey E. Hinton 和 Ronald J. Williams。通过误差反向传播学习表征。*Nature*，1986。第 3 页引用。

Robin M. Schmidt、Frank Schneider 和 Philipp Hennig。沿着拥挤谷下降——深度学习优化器的基准测试。载于国际机器学习会议，2021。第 2 页引用。

Or Sharir、Barak Peleg 和 Yoav Shoham。训练自然语言处理模型的成本：简明概述。*arXiv:2004.08900*，2020。第 2 页引用。

Karen Simonyan 和 Andrew Zisserman。用于大规模图像识别的超深卷积网络。载于国际学习表征会议，2015。第 11 页引用。

Michael Stewart。存在小奇异值时奇异值分解的扰动。*Linear Algebra and itsApplications*，2006。第 15 页引用。

Haoyuan Sun、Kwangjun Ahn、Christos Thrampoulidis 和 Navid Azizan。镜像下降最大化广义间隔，并且可以高效实现。载于 神经信息处理系统，2022。第 3 页引用。

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。《Attention is all you need》。发表于 神经信息处理系统，2017 年。引用见第 14 页。

Hermann Weyl。《线性偏微分方程特征值的渐近分布定律（及其在空腔辐射理论中的应用）》数学年刊，1912年。引用见第8页和第15页。

Greg Yang 和 Edward J. Hu。《张量程序 IV：无限宽神经网络中的特征学习》。发表于国际机器学习会议，2021。引用见第 3、4、5、9、10 和 15 页。Greg Yang、Edward J. Hu、Igor Babuschkin、Szymon Sidor、Xiaodong Liu、David Farhi、Nick Ryder、Jakub Pachocki、Weizhu Chen 和 Jianfeng Gao。《通过零样本超参数迁移调优大型神经网络》。发表于神经信息处理系统，2021。引用见第 9 和 12 页。Yang You、Igor Gitman 和 Boris Ginsburg。《将随机梯度下降批量大小扩展至 32K，以进行 ImageNet 训练》。技术报告，加州大学伯克利分校，2017。引用见第 10 页。

Jingzhao Zhang、Tianxing He、Suvrit Sra 和 Ali Jadbabaie。《为什么梯度裁剪能加速训练：自适应性的理论论证》。载于 国际学习表征会议，2020。第 3 页引用。

A 证明

以下是正文中理论结果的证明。

命题1（线性化误差的分解）对于任意可微损失 ℓ 以及任意可微机器学习模型 $\boldsymbol{f}$，目标函数$\mathcal{L}$ 的线性化误差可分解如下：

$$\underbrace{\Delta\mathcal{L}(\boldsymbol{w}) - \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w}}_{\text{linearisation error of objective}} = \frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \nabla_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})^\top \underbrace{[\Delta\boldsymbol{f}(\boldsymbol{x}) - \nabla_{\boldsymbol{w}}\boldsymbol{f}(\boldsymbol{x})\Delta\boldsymbol{w}]}_{\text{linearisation error of model}} + \frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \underbrace{\Delta\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}) - \nabla_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})^\top\Delta\boldsymbol{f}(\boldsymbol{x})}_{\text{linearisation error of loss}}.$$

证明。 根据链式法则，$\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w} = \frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \nabla_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})^\top\nabla_{\boldsymbol{w}}\boldsymbol{f}(\boldsymbol{x})\Delta\boldsymbol{w}$。因此：

$$\Delta\mathcal{L}(\boldsymbol{w}) - \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w} = \frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \Delta\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}) - \nabla_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})^\top\nabla_{\boldsymbol{w}}\boldsymbol{f}(\boldsymbol{x})\Delta\boldsymbol{w}.$$

加法与减法 从右侧减去 $\frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \nabla_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})^\top\Delta\boldsymbol{f}(\boldsymbol{x})$ 即可得到结果。 □

引理1（平方损失的Bregman散度）当将 ℓ 设为平方损失时：

$$\mathrm{bregman}_{\ell(\cdot,\boldsymbol{y})}(\boldsymbol{f}(\boldsymbol{x}), \Delta\boldsymbol{f}(\boldsymbol{x})) = \tfrac{1}{2d_L}\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_2^2.$$

证明。 展开损失扰动中的欧几里得范数 $\Delta\ell$，得到：

$$\begin{aligned}\Delta\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}) &= \tfrac{1}{2d_L}\|\boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 - \tfrac{1}{2d_L}\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 \\ &= \tfrac{1}{2d_L}\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_2^2 + (\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y})^\top\Delta\boldsymbol{f}(\boldsymbol{x}).\end{aligned}$$

通过识别出 $\nabla_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\ \boldsymbol{y})^\top\Delta\boldsymbol{f}(\boldsymbol{x}) = (\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y})^\top\Delta\boldsymbol{f}(\boldsymbol{x})$，即可得出结果。 □

引理2（Bregman 散度的交叉熵损失）当 ℓ 设为交叉熵损失，且若 $\boldsymbol{y}^\top\mathbf{1} = 1$，则：

$$\begin{aligned}\mathrm{bregman}_{\ell(\cdot,\boldsymbol{y})}(\boldsymbol{f}(\boldsymbol{x}), \Delta\boldsymbol{f}(\boldsymbol{x})) &= D_{\mathrm{KL}}\Big(\operatorname{softmax}(\boldsymbol{f}(\boldsymbol{x}))\Big\|\operatorname{softmax}(\boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{f}(\boldsymbol{x}))\Big) \\ &\leqslant \tfrac{1}{2}\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_\infty^2 + \mathcal{O}(\Delta\boldsymbol{f}^3).\end{aligned}$$

证明。 首先，由于 $\sum_i \boldsymbol{y}_i = 1$，交叉熵损失可重写为：

$$\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}) := -\log[\operatorname{softmax}(\boldsymbol{f}(\boldsymbol{x}))]^\top\boldsymbol{y} = -\boldsymbol{f}(\boldsymbol{x})^\top\boldsymbol{y} + \log\|\exp\boldsymbol{f}(\boldsymbol{x})\|_1.$$

线性项 $-\boldsymbol{f}(\boldsymbol{x})^\top\boldsymbol{y}$ 不会影响线性化误差，因此可以忽略。于是：

$$\begin{aligned}&\Delta\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}) - \nabla_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})^\top\Delta\boldsymbol{f}(\boldsymbol{x}) \\ &\quad= \log\|\exp(\boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{f}(\boldsymbol{x}))\|_1 - \log\|\exp\boldsymbol{f}(\boldsymbol{x})\|_1 - \nabla_{\boldsymbol{f}(\boldsymbol{x})}\log\|\exp\boldsymbol{f}(\boldsymbol{x})\|_1^\top\Delta\boldsymbol{f}(\boldsymbol{x}) \\ &\quad= \log\frac{1/\|\exp\boldsymbol{f}(\boldsymbol{x})\|_1}{1/\|\exp(\boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{f}(\boldsymbol{x}))\|_1} - \frac{\exp\boldsymbol{f}(\boldsymbol{x})^\top}{\|\exp\boldsymbol{f}(\boldsymbol{x})\|_1}\Delta\boldsymbol{f}(\boldsymbol{x}) \\ &\quad= \frac{\exp\boldsymbol{f}(\boldsymbol{x})^\top}{\|\exp\boldsymbol{f}(\boldsymbol{x})\|_1}\log\frac{\exp\boldsymbol{f}(\boldsymbol{x})/\|\exp\boldsymbol{f}(\boldsymbol{x})\|_1}{\exp(\boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{f}(\boldsymbol{x}))/\|\exp(\boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{f}(\boldsymbol{x}))\|_1}.\end{aligned}$$

最后一行等价于 $D_{\mathrm{KL}}\Big(\operatorname{softmax}(\boldsymbol{f}(\boldsymbol{x}))\Big\|\operatorname{softmax}(\boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{f}(\boldsymbol{x}))\Big)$，从而证明了第一个等式。

为证明该不等式，令 $\otimes$ 表示外积，并定义 $p := \mathsf{softmax}(f(\boldsymbol{x}))$。于是我们有：

$$\begin{aligned}\Delta\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y}) - \nabla_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})^\top\Delta\boldsymbol{f}(\boldsymbol{x}) &= \frac{1}{2}\Delta\boldsymbol{f}(\boldsymbol{x})^\top\nabla^2_{\boldsymbol{f}(\boldsymbol{x})}\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})\Delta\boldsymbol{f}(\boldsymbol{x}) + \mathcal{O}(\Delta\boldsymbol{f}^3)\\ &= \frac{1}{2}\Delta\boldsymbol{f}(\boldsymbol{x})^\top\nabla^2_{\boldsymbol{f}(\boldsymbol{x})}\log\|\exp\boldsymbol{f}(\boldsymbol{x})\|_1\Delta\boldsymbol{f}(\boldsymbol{x}) + \mathcal{O}(\Delta\boldsymbol{f}^3)\\ &= \frac{1}{2}\Delta\boldsymbol{f}(\boldsymbol{x})^\top[\mathrm{diag}(p) - p\otimes p]\Delta\boldsymbol{f}(\boldsymbol{x}) + \mathcal{O}(\Delta\boldsymbol{f}^3)\\ &\leqslant \frac{1}{2}\Delta\boldsymbol{f}(\boldsymbol{x})^\top\mathrm{diag}(p)\Delta\boldsymbol{f}(\boldsymbol{x}) + \mathcal{O}(\Delta\boldsymbol{f}^3)\\ &\leqslant \frac{1}{2}\|\Delta\boldsymbol{f}(\boldsymbol{x})\|^2_\infty + \mathcal{O}(\Delta\boldsymbol{f}^3),\end{aligned}$$

其中我们利用了 $p\otimes p$ 是正定的这一性质，然后对 $\|p\|_1 = 1$ 应用了 Hölder 不等式。 □

定理 1（函数展开）考虑凸可微损失 ℓ 和可微机器学习模型 $\boldsymbol{f}$。在假设 1 下，相应的复合目标函数 $\mathcal{L}$ 可展开为：

$$\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w}) = \underbrace{\mathcal{L}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w}}_{\text{first-order Taylor series}} + \frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}}\mathrm{bregman}_{\ell(\cdot,\boldsymbol{y})}(\boldsymbol{f}(\boldsymbol{x}),\Delta\boldsymbol{f}(\boldsymbol{x})).$$

证明。 将假设 1 代入命题 1，并应用定义 9，即可得到该结果。 □

推论 1（均方误差的函数展开）在假设 1 下，对于平方损失：

$$\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w}) = \mathcal{L}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w} + \frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}}\tfrac{1}{2d_L}\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_2^2.$$

证明。 结合引理 1 和定理 1，即可得到该结果。 □

推论2（交叉熵损失的函数主化）在假设 1 下，对于交叉熵损失，若 $\boldsymbol{y}^\top\mathbf{1} = 1$：

$$\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w}) \leqslant \mathcal{L}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w} + \frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}}\tfrac{1}{2}\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_\infty^2 + \mathcal{O}(\Delta\boldsymbol{f}^3).$$

证明。 结合引理2与定理1即可得到该结果。 □

引理 3（输出界）全连接网络 $\boldsymbol{f}$ 的输出范数满足以下界：

$$\|\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})\|_2 \leqslant \left[\prod_{k=1}^{L}\|\boldsymbol{W}_k\|_*\right]\times\|\boldsymbol{x}\|_2 = \sqrt{d_L}\text{ under Prescription 1.}$$

证明。 对于任意维度相容的向量 $\boldsymbol{v}$和矩阵 $\boldsymbol{M}$，有 $\|\boldsymbol{M}\boldsymbol{v}\|_2 \leqslant \|\boldsymbol{M}\|_*\cdot\|\boldsymbol{v}\|_2$和$\|\mathsf{relu}\,\boldsymbol{v}\|_2 \leqslant \|\boldsymbol{v}\|_2$。沿着网络深度递归地应用这些结果，即可得到该引理。 □

引理 4（深度相对信任）将全连接网络$\boldsymbol{f}$的权重 $\boldsymbol{w} = (\boldsymbol{W}_1,...,\boldsymbol{W}_L)$ 调整为 $\Delta\boldsymbol{w} = (\Delta\boldsymbol{W}_1,...,\Delta\boldsymbol{W}_L)$时，所产生的函数扰动 $\Delta\boldsymbol{f}(\boldsymbol{x}) := \boldsymbol{f}(\boldsymbol{x};\,\boldsymbol{w}+\Delta\boldsymbol{w}) - \boldsymbol{f}(\boldsymbol{x};\,\boldsymbol{w})$ 满足：

$$\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_2 \leqslant \left[\prod_{k=1}^{L}\|\boldsymbol{W}_k\|_*\right]\times\|\boldsymbol{x}\|_2\times\left[\prod_{k=1}^{L}\left(1+\frac{\|\Delta\boldsymbol{W}_k\|_*}{\|\boldsymbol{W}_k\|_*}\right)-1\right] \leqslant \sqrt{d_L}\times(\exp\eta - 1)\text{ under Prescription 1.}$$

证明。 我们通过归纳法进行证明。首先，考虑一个有 $L = 1$ 层的网络： $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W}_1\boldsymbol{x}$。注意到 $\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_2 = \|\Delta\boldsymbol{W}_1\boldsymbol{x}\|_2 \leqslant \|\Delta\boldsymbol{W}_1\|_*\cdot\|\boldsymbol{x}\|_2$，这正是所需的结果。接下来，假设该结论对网络 $\boldsymbol{g}(\boldsymbol{x})$ 成立。

使用 $L-1$ 层，并考虑添加一层以获得 $\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{W}_L\circ\mathrm{ReLU}\ \circ\boldsymbol{g}(\boldsymbol{x})$。然后：

$$\begin{aligned}\|\Delta\boldsymbol{f}(\boldsymbol{x})\|_2&=\|(\boldsymbol{W}_L+\Delta\boldsymbol{W}_L)\circ\mathrm{relu}\circ(\boldsymbol{g}(\boldsymbol{x})+\Delta\boldsymbol{g}(\boldsymbol{x}))-\boldsymbol{W}_L\circ\mathrm{relu}\circ\boldsymbol{g}(\boldsymbol{x})\|_2\\&=\|\boldsymbol{W}_L\left(\mathrm{relu}\circ(\boldsymbol{g}(\boldsymbol{x})+\Delta\boldsymbol{g}(\boldsymbol{x}))-\mathrm{relu}\circ\boldsymbol{g}(\boldsymbol{x})\right)+\Delta\boldsymbol{W}_L\left(\mathrm{relu}\circ(\boldsymbol{g}(\boldsymbol{x})+\Delta\boldsymbol{g}(\boldsymbol{x}))-\mathrm{relu}(0)\right)\|_2\\&\leqslant\|\boldsymbol{W}_L\|_*\cdot\|\Delta\boldsymbol{g}(\boldsymbol{x})\|_2+\|\Delta\boldsymbol{W}_L\|_*\cdot(\|\boldsymbol{g}(\boldsymbol{x})\|_2+\|\Delta\boldsymbol{g}(\boldsymbol{x})\|_2)\\&=(\|\boldsymbol{W}_L\|_*+\|\Delta\boldsymbol{W}_L\|_*)\cdot\|\Delta\boldsymbol{g}(\boldsymbol{x})\|_2+\|\Delta\boldsymbol{W}_L\|_*\cdot\|\boldsymbol{g}(\boldsymbol{x})\|_2,\end{aligned}$$

其中，不等式是通过应用三角不等式、算子范数界、ReLU 是 1-利普希茨函数这一事实，以及再次应用三角不等式得到的。但根据归纳假设和引理 3，右侧有如下界：

$$\begin{aligned}&(\|\boldsymbol{W}_L\|_*+\|\Delta\boldsymbol{W}_L\|_*)\left[\prod_{k=1}^{L-1}\left(1+\frac{\|\Delta\boldsymbol{W}_k\|_*}{\|\boldsymbol{W}_k\|_*}\right)-1\right]\times\left[\prod_{k=1}^{L-1}\|\boldsymbol{W}_k\|_*\right]\times\|\boldsymbol{x}\|_2+\|\Delta\boldsymbol{W}_L\|_*\times\left[\prod_{k=1}^{L-1}\|\boldsymbol{W}_k\|_*\right]\times\|\boldsymbol{x}\|_2\\&\quad=\left[\prod_{k=1}^{L}\left(1+\frac{\|\Delta\boldsymbol{W}_k\|_*}{\|\boldsymbol{W}_k\|_*}\right)-1\right]\times\left[\prod_{k=1}^{L}\|\boldsymbol{W}_k\|_*\right]\times\|\boldsymbol{x}\|_2.\end{aligned}$$

归纳过程至此完成。为进一步在处方 1 下界定这一结果，注意到乘积$\left[\prod_{k=1}^{L}\|\boldsymbol{W}_k\|_*\right]\times\|\boldsymbol{x}\|_2$会逐项消去，最终只剩下 $\sqrt{d_L}$，而另一个乘积满足：

$$\left[\prod_{k=1}^{L}\left(1+\frac{\|\Delta\boldsymbol{W}_k\|_*}{\|\boldsymbol{W}_k\|_*}\right)-1\right]=\left(1+\frac{\eta}{L}\right)^L-1\leqslant\lim_{L\to\infty}\left(1+\frac{\eta}{L}\right)^L-1=\exp\eta-1.$$

结合这些观察即可得到结果。 □

引理 5（指数 主化）对于采用平方损失的全连接网络，在假设 1 和处方 1 下 ：

$$\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w})\leqslant\mathcal{L}(\boldsymbol{w})+\frac{\eta}{L}\sum_{k=1}^{L}\left[\sqrt{d_k/d_{k-1}}\times\mathrm{tr}\,\frac{\Delta\boldsymbol{W}_k^\top\nabla_{\boldsymbol{W}_k}\mathcal{L}}{\|\Delta\boldsymbol{W}_k\|_*}\right]+\tfrac{1}{2}\left(\exp\eta-1\right)^2.$$

证明。将引理 4 代入推论 1，并分解 $\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top\Delta\boldsymbol{w}=\sum_{k=1}^{L}\mathrm{tr}(\Delta\boldsymbol{W}_k^\top\nabla_{\boldsymbol{W}_k}\mathcal{L})$。注意到在处方 1 下，扰动满足 $\|\Delta\boldsymbol{W}_k\|_*=\sqrt{d_k/d_{k-1}}\cdot\frac{\eta}{L}$，即可得出结果。 – □

定理 2（自动梯度下降）对于深度全连接网络，在假设 1 和 2 及处方 1 的条件下，通过如下设置可使引理 5 给出的平方损失主化达到最小：

$$\eta=\log\frac{1+\sqrt{1+4G}}{2},\qquad\Delta\boldsymbol{W}_k=-\frac{\eta}{L}\cdot\sqrt{d_k/d_{k-1}}\cdot\frac{\nabla_{\boldsymbol{W}_k}\mathcal{L}}{\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_F},\qquad\text{for all layers }k=1,...,L.$$

证明。 出现在引理 5 中的内积$\mathrm{tr}\Delta\boldsymbol{W}_k^\top\nabla_{\boldsymbol{W}_k}\mathcal{L}/\|\Delta\boldsymbol{W}_k\|_*$在扰动$\Delta\boldsymbol{W}_k$满足 $\Delta\boldsymbol{W}_k/\|\Delta\boldsymbol{W}_k\|_*=-\nabla_{\boldsymbol{W}_k}\mathcal{L}/\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_*$ 时最为负。将这一结果代回引理 5 可得：

$$\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w})\leqslant\mathcal{L}(\boldsymbol{w})-\frac{\eta}{L}\sum_{k=1}^{L}\left[\sqrt{d_k/d_{k-1}}\times\frac{\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_F^2}{\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_*}\right]+\tfrac{1}{2}\left(\exp\eta-1\right)^2.$$

在假设下 n 2 中，我们有 $\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_F^2/\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_*=\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_F$，因此这个不等式可简化为 ：

$$\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w})\leqslant\mathcal{L}(\boldsymbol{w})-\eta\cdot G+\tfrac{1}{2}\left(\exp\eta-1\right)^2.$$

对右侧关于 η 求导并令其为零，得到 $(\exp\eta-1)\exp\eta=G$。应用求根公式并保留正解，得到 $\exp\eta=\frac{1}{2}(1+\sqrt{1+4G})$。结合关系 $\Delta\boldsymbol{W}_k/\|\Delta\boldsymbol{W}_k\|_*=-\nabla_{\boldsymbol{W}_k}\mathcal{L}/\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_*$，并应用 $\|\Delta\boldsymbol{W}_k\|_*=\sqrt{d_k/d_{k-1}}\cdot\frac{\eta}{L}$（根据处方 1），即可得到结果。 □

引理 6（有界目标函数）对于平方损失，目标函数有如下界：

$$\mathcal{L}(\boldsymbol{w}) \leqslant \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{\|\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})\|_2^2 + \|\boldsymbol{y}\|_2^2}{2d_L} \leqslant 1 \text{ under Prescription 1.}$$

证明。 该结果可由以下不等式链推出：

$$\mathcal{L}(\boldsymbol{w}) := \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{1}{2d_L}\|\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y}\|_2^2 \leqslant \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{1}{2d_L}(\|\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})\|_2^2 + \|\boldsymbol{y}\|_2^2) \leqslant \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{d_L + d_L}{2d_L} = 1,$$

其中，第二个不等式在处方 1 下成立。 □

引理 7 （有界梯度）对于平方损失，第 k 层的梯度范数满足以下界：

$$\|\nabla_{\boldsymbol{W}_k}\mathcal{L}\|_F \leqslant \frac{\prod_{l=1}^L \|\boldsymbol{W}_l\|_*}{\|\boldsymbol{W}_k\|_*} \cdot \sqrt{\frac{2\mathcal{L}(\boldsymbol{w})}{d_L}} \cdot \sqrt{\frac{1}{|\mathsf{S}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \|\boldsymbol{x}\|_2^2} \leqslant \sqrt{2\cdot\frac{d_{k-1}}{d_k}} \text{ under Prescription 1.}$$

证明。 根据链式法则，均方误差目标函数的梯度可写为：

$$\nabla_{\boldsymbol{W}_k}\mathcal{L}(\boldsymbol{w}) = \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{1}{d_L}(\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y})^\top \boldsymbol{W}_L \cdot \boldsymbol{D}_{L-1}\boldsymbol{W}_{L-1}\ldots\boldsymbol{D}_{k+1}\boldsymbol{W}_{k+1}\cdot\boldsymbol{D}_k \otimes \boldsymbol{D}_{k-1}\boldsymbol{W}_{k-1}\ldots\boldsymbol{D}_1\boldsymbol{W}_1\boldsymbol{x},$$

其中， $\otimes$ 表示外积， $\boldsymbol{D}_k$ 表示一个对角矩阵，其对角元素在 ReLU 激活时为 1，未激活时为 0。由于算子范数 $\|\boldsymbol{D}_k\|_* = 1$，我们有弗罗贝尼乌斯范数$\|\nabla_{\boldsymbol{W}_k}\mathcal{L}(\boldsymbol{w})\|_F$ 的上界为：

$$\begin{aligned}
&\frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{1}{d_L}\|(\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y})^\top \boldsymbol{W}_L \cdot \boldsymbol{D}_{L-1}\boldsymbol{W}_{L-1}\ldots\boldsymbol{D}_{k+1}\boldsymbol{W}_{k+1}\cdot\boldsymbol{D}_k \otimes \boldsymbol{D}_{k-1}\boldsymbol{W}_{k-1}\ldots\boldsymbol{D}_1\boldsymbol{W}_1\boldsymbol{x}\|_F \\
&\quad= \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{1}{d_L}\|(\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y})^\top \boldsymbol{W}_L \cdot \boldsymbol{D}_{L-1}\boldsymbol{W}_{L-1}\ldots\boldsymbol{D}_{k+1}\boldsymbol{W}_{k+1}\cdot\boldsymbol{D}_k\|_2 \cdot \|\boldsymbol{D}_{k-1}\boldsymbol{W}_{k-1}\ldots\boldsymbol{D}_1\boldsymbol{W}_1\boldsymbol{x}\|_2 \\
&\quad\leqslant \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{1}{d_L}\|\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y}\|_2 \cdot \|\boldsymbol{W}_L\|_* \cdot \|\boldsymbol{W}_{L-1}\|_* \ldots \|\boldsymbol{W}_{k+1}\|_* \cdot \|\boldsymbol{W}_{k-1}\|_* \ldots \|\boldsymbol{W}_1\|_* \cdot \|\boldsymbol{x}\|_2 \\
&\quad= \frac{\prod_{l=1}^L \|\boldsymbol{W}_l\|_*}{\|\boldsymbol{W}_k\|} \times \frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{1}{d_L}\|\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y}\|_2 \cdot \|\boldsymbol{x}\|_2 \\
&\quad\leqslant \frac{\prod_{l=1}^L \|\boldsymbol{W}_l\|_*}{\|\boldsymbol{W}_k\|_*} \cdot \frac{1}{\sqrt{d_L}} \sqrt{\frac{2}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \frac{1}{2d_L}\|\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y}\|_2^2} \cdot \sqrt{\frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \|\boldsymbol{x}\|_2^2} \\
&\quad= \frac{\prod_{l=1}^L \|\boldsymbol{W}_l\|_*}{\|\boldsymbol{W}_k\|_*} \cdot \sqrt{\frac{2\mathcal{L}(\boldsymbol{w})}{d_L}} \cdot \sqrt{\frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \|\boldsymbol{x}\|_2^2}.
\end{aligned}$$

在上述论证中，第一个不等式是通过递归应用算子范数上界得到的，第二个不等式则源于柯西-施瓦茨不等式。在处方 1 下，右侧可化简，并且我们可以应用引理 6 得到：

$$\|\nabla_{\boldsymbol{W}_k}\mathcal{L}(\boldsymbol{w})\|_F \leqslant \frac{\prod_{l=1}^L \|\boldsymbol{W}_l\|_*}{\|\boldsymbol{W}_k\|_*} \cdot \sqrt{\frac{2\mathcal{L}(\boldsymbol{w})}{d_L}} \cdot \sqrt{\frac{1}{|\mathsf{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathsf{S}} \|\boldsymbol{x}\|_2^2} \leqslant \frac{\sqrt{d_L/d_0}}{\sqrt{d_k/d_{k-1}}} \cdot \sqrt{\frac{2}{d_L}} \cdot \sqrt{d_0} = \sqrt{2} \cdot \sqrt{\frac{d_{k-1}}{d_k}}.$$

引理 8 （收敛至临界点的速率）考虑一个通过自动梯度下降（定理 2）和平方损失训练的全连接网络，经过 T 次迭代。令 G_t 表示步骤 $t \leqslant T$处的梯度汇总（定义 11）。在假设 1 和 2 以及处方 1 下，自动梯度下降以如下速率收敛：

$$\min_{t\in\{1,\dots,T\}} G_t^2 \leqslant \frac{11}{T}.$$

证明。 定理 2 规定 $\exp\eta = \frac{1}{2}(1+\sqrt{1+4G})$，因此 $\eta = \log\left(1+\frac{\sqrt{1+4G}-1}{2}\right)$。我们先证明一些有用的辅助界。根据引理 7 和处方 1，梯度汇总有如下界：

$$G := \frac{1}{L}\sum_{k=1}^{L}\sqrt{d_k/d_{k-1}}\cdot\|\nabla_{\boldsymbol{W}_k}\mathcal{L}(\boldsymbol{w})\|_F \leqslant \frac{1}{L}\sum_{k=1}^{L}\sqrt{2} < 2.$$

梯度汇总 G 小于二这一事实很重要，因为对于 $x\leqslant 1$，我们有 $\log(1+x)\geqslant x\log 2$。进而，这意味着由于 $G<2$，我们有 $\eta = \log\frac{1+\sqrt{1+4G}}{2} \geqslant \frac{\sqrt{1+4G}-1}{2}\log 2$。我们还需要知道，对于 $G<2$，有 $\frac{1}{2}\cdot G \leqslant \frac{\sqrt{1+4G}-1}{2} \leqslant G$。

有了这些界，分析就相当标准了。根据定理 2 证明中的一个中间步骤，单步目标函数的变化有如下界：

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}+\Delta\boldsymbol{w})-\mathcal{L}(\boldsymbol{w}) &\leqslant -\eta\cdot G + \tfrac{1}{2}(\exp\eta-1)^2\\ &\leqslant -\tfrac{\sqrt{1+4G}-1}{2}\left(G\log 2 - \tfrac{1}{2}\tfrac{\sqrt{1+4G}-1}{2}\right)\\ &\leqslant -\tfrac{1}{2}\cdot(\log 2-\tfrac{1}{2})\cdot G^2 \leqslant -G^2/11,\end{aligned}$$

其中，第二个和第三个不等式由我们的辅助界推出。令 G_t 表示第 t 步的梯度汇总，对各个时间步取此界的平均值，并应用望远镜性质，可得：

$$\min_{t\in[1,\dots,T]} G_t^2 \leqslant \frac{1}{T}\sum_{t=1}^{T} G_t^2 \leqslant \frac{11}{T}\sum_{t=1}^{T}\mathcal{L}(\boldsymbol{w}_t)-\mathcal{L}(\boldsymbol{w}_{t+1}) = \frac{11}{T}\cdot(\mathcal{L}(\boldsymbol{w}_1)-\mathcal{L}(\boldsymbol{w}_T)) \leqslant \frac{11}{T},$$

其中，最后一个不等式由引理 6 以及 $\mathcal{L}(\boldsymbol{w}_T)\geqslant 0$ 这一事实推出。

定理 3 （收敛到全局极小值的速率）对于与引理8 相同设置下、但增加了假设 3的自动梯度下降（定理 2），第 T 步的均方误差目标满足：

$$\mathcal{L}(\boldsymbol{w}_T) \leqslant \frac{1}{\alpha^2}\times\frac{6}{T}.$$

证明。 根据假设 3，时间步 t 的梯度摘要必须满足 $G_t \geqslant \alpha\times\sqrt{2\cdot\mathcal{L}(\boldsymbol{w}_t)}$。因此，时间步 t 的目标函数有界于 $\mathcal{L}(\boldsymbol{w}_t)\leqslant G_t^2/(2\alpha^2)$。结合引理 8 可得：

$$\mathcal{L}(\boldsymbol{w}_T) = \min_{t\in[1,\dots,T]}\mathcal{L}(\boldsymbol{w}_t) \leqslant \frac{1}{2\alpha^2}\min_{t\in[1,\dots,T]} G_t^2 \leqslant \frac{6}{\alpha^2 T}.$$

证明完毕。 □

B PyTorch 实现

以下代码在 PyTorch（Paszke 等人，2019 年）中实现了自动梯度下降。我们加入了一个增益超参数，用于控制更新幅度；将其从默认值 1.0 略微调高，或许可以加快训练。我们强调，论文中报告的所有结果均使用了单位增益。

```
import math
import torch

from torch.nn.init import orthogonal_

def singular_value(p):
    sv = math.sqrt(p.shape[0] / p.shape[1])
    if p.dim() == 4:
        sv /= math.sqrt(p.shape[2] * p.shape[3])
    return sv

class AGD:
    @torch.no_grad()
    def __init__(self, net, gain=1.0):

        self.net = net
        self.depth = len(list(net.parameters()))
        self.gain = gain

        for p in net.parameters():
            if p.dim() == 1: raise Exception("Biases are not supported.")
            if p.dim() == 2: orthogonal_(p)
            if p.dim() == 4:
                for kx in range(p.shape[2]):
                    for ky in range(p.shape[3]):
                        orthogonal_(p[:,:,kx,ky])
            p *= singular_value(p)

    @torch.no_grad()
    def step(self):

        G = 0
        for p in self.net.parameters():
            G += singular_value(p) * p.grad.norm(dim=(0,1)).sum()
        G /= self.depth

        log = math.log(0.5 * (1 + math.sqrt(1 + 4*G)))

        for p in self.net.parameters():
            factor = singular_value(p) / p.grad.norm(dim=(0,1), keepdim=True)
            p -= self.gain * log / self.depth * factor * p.grad
```