ISAAC NEWTON：基于输入的近似曲率与牛顿法

Felix Petersen[12], **Tobias Sutter**[2], **Christian Borgelt**[3],**Dongsung Huh**[4], **Hilde Kuehne**[45], **Yuekai Sun**[6], **OliverDeussen**[21]斯坦福大学， [2]康斯坦茨大学， [3]萨尔茨堡大学，[4]MIT-IBM沃森人工智能实验室， [5]法兰克福大学， [6]密歇根大学 petersen@cs.stanford.edu

arXiv:2305.00604v1 [cs.LG] 1 May 2023

摘要

我们提出了 ISAAC（基于输入的近似曲率），这是一种新方法，通过选定的二阶信息对梯度进行条件化；在批量大小小于神经元数量的前提下，其计算开销渐近趋于零。我们证明，仅根据相应层的输入，就可以计算出一个良好的条件器，而不会带来显著的计算开销。所提出的方法即使在小批量随机场景中也能实现有效训练，因此既能与一阶方法竞争，也能与二阶方法竞争。

1 引言

在大规模机器学习（ML）应用中，由于内存需求高且每次迭代的计算成本令人望而却步，二阶优化方法历来远不如一阶方法受到关注。然而，近年来二阶优化方法在机器学习中逐渐普及，主要是因为与一阶方法相比，它们具有更快的收敛性质 [1]。牛顿步中计算海森矩阵逆（也称为预条件矩阵）的高昂成本，也通过根据梯度的变化估计曲率得到了解决。笼统地说，这些算法称为 拟牛顿方法；关于全面的介绍，请参见 Nocedal & Wright [2]。近期文献中提出了多种预条件矩阵的近似方法 [3]–[6]。从理论角度来看，二阶优化方法远不如一阶方法那样得到充分理解。弥补这一差距仍是一个活跃的研究方向 [7], [8]。

受训练神经网络这一任务的启发，并基于这样一个观察：利用与神经网络目标函数相关的局部曲率信息，相比标准一阶方法 [9]–[11], 能够在每次迭代中取得快得多的进展，因此提出了多种方法。其中一种受到广泛关注的方法称为克罗内克因子近似曲率（*K-FAC*） [12], ，其核心是对广义高斯-牛顿矩阵和费舍尔信息矩阵进行精细近似，以量化底层神经网络目标函数的曲率；随后即可高效地求其逆矩阵。

受 K-FAC 近似和牛顿法中的蒂霍诺夫正则化启发，我们提出了一种新颖的双参数正则化克罗内克因子化牛顿更新步。该方案将经典的蒂霍诺夫正则化解耦，并在特定极限情况下允许我们利用选定的二阶信息对梯度进行条件化，且其计算开销渐近趋于零。尽管从计算复杂度角度看，这一特性使所提出的方法极具吸引力，我们仍证明，该方法在高维机器学习问题上的实证表现与现有最先进方法相当。

本文的贡献可总结如下：(i) 我们提出了一种新颖的双参数正则化 K-FAC 近似高斯-牛顿更新步骤；(ii) 我们证明，对于任意一对正则化参数，所提出的更新方向始终是损失下降方向；(iii) 当其中一个正则化参数趋于无穷大时，我们得到一种高效有效的梯度条件化方法，其开销渐近趋于零；(iv) 我们对该方法进行了实证分析，发现高效条件化方法能够保持其高开销对应方法的性能；(v) 我们展示了该方法在小批量随机场景中的有效性，并观察到其性能可与一阶方法和拟牛顿方法相媲美。

2 预备知识

本节回顾二阶优化的相关内容，重点介绍广义高斯-牛顿方法。结合克罗内克分解，我们由此得到一种新的正则化更新方案。我们考虑训练一个 L 层神经网络 $f(x;\theta)$，其递归定义如下:

$$z_i \leftarrow a_{i-1}W^{(i)} \quad \text{(pre-activations)}, \qquad a_i \leftarrow \phi(z_i) \quad \text{(activations)}, \tag{1}$$

其中， $a_0 = x$ 是输入向量， $a_L = f(x;\theta)$是输出向量。除非另有说明，我们假设这些向量为行向量（即，在 $\mathbb{R}^{1\times n}$ 中），因为这样可以直接扩展到稍后介绍的（批次）向量化情形（即，在 $\mathbb{R}^{b\times n}$ 中）。对于任意层 i，令 $W^{(i)} \in \mathbb{R}^{d_{i-1}\times d_i}$ 为权重矩阵，令 ϕ 为逐元素非线性函数。我们考虑一个凸损失函数$\mathcal{L}(y,\ y')$，用于衡量 y 与 y'之间的差异。于是，训练优化问题为

$$\arg\min_{\theta} \mathbb{E}_{x,y}\left[\mathcal{L}(f(x;\theta),y)\right], \tag{2}$$

其中， $\theta = \left[\theta^{(1)},\ldots,\theta^{(L)}\right]$，且 $\theta^{(i)} = \mathrm{vec}(W^{(i)})$。

用于求解（2）的经典牛顿法表示为以下更新规则

$$\theta' = \theta - \eta\,\mathbf{H}_\theta^{-1}\nabla_\theta\mathcal{L}(f(x;\theta),y)\,, \tag{3}$$

其中， $\eta > 0$ 表示学习率，$\mathbf{H}_\theta$ 是与式（2）中的目标函数相对应的海森矩阵。通过添加蒂霍诺夫正则化项 [13]，可提高采用牛顿法求解估计问题时的稳定性和效率，从而得到正则化牛顿法

$$\theta' = \theta - \eta\,(\mathbf{H}_\theta + \lambda\mathbf{I})^{-1}\nabla_\theta\mathcal{L}(f(x;\theta),y)\,, \tag{4}$$

其中， $\lambda > 0$ 是所谓的蒂霍诺夫正则化参数。众所周知， [14], [15], 在假设用一阶泰勒展开近似模型 f 的情况下，海森矩阵对应于所谓的广义高斯-牛顿矩阵（GGN）$\mathbf{G}_\theta$，因此式（4）可表示为

$$\theta' = \theta - \eta\,(\mathbf{G}_\theta + \lambda\mathbf{I})^{-1}\nabla_\theta\mathcal{L}(f(x;\theta),y)\,. \tag{5}$$

式（5）的一个主要实际局限在于逆项的计算。缓解这一困难的一种方法称为克罗内克因子近似曲率（K-FAC） [12]，该方法对块对角（即逐层）的经验海森矩阵或广义高斯-牛顿矩阵进行近似。受 K-FAC 启发，另有研究探讨了对$\mathbf{G}_\theta$ 及其逆矩阵 [15]的近似。下文将讨论一种广受欢迎、能够实现（相对）高效计算的方法。

广义高斯-牛顿矩阵$\mathbf{G}_\theta$ 定义为

$$\mathbf{G}_\theta = \mathbb{E}\left[(\mathbf{J}_\theta f(x;\theta))^\top \nabla_f^2\mathcal{L}(f(x;\theta),y)\,\mathbf{J}_\theta f(x;\theta)\right], \tag{6}$$

其中，$\mathbf{J}$ 和 ∇^2 分别表示雅可比矩阵和海森矩阵。相应地，$\mathbf{G}_\theta$ 中对应于第 i 层权重的对角块为

$$\mathbf{G}_{W^{(i)}} = \mathbb{E}\left[(\mathbf{J}_{W^{(i)}} f(x;\theta))^\top \nabla_f^2\mathcal{L}(f(x;\theta),y)\,\mathbf{J}_{W^{(i)}} f(x;\theta)\right].$$

根据反向传播规则 $\mathbf{J}_{W^{(i)}} f(x;\theta) = \mathbf{J}_{z_i} f(x;\theta)\ a_{i-1}$, $a^\top b = a\otimes b$以及混合乘积性质，我们可以将 $\mathbf{G}_{W^{(i)}}$ 改写为

$$\mathbf{G}_{W^{(i)}} = \mathbb{E}\Big[\big((\mathbf{J}_{z_i} f(x;\theta)\,a_{i-1})^\top (\nabla_f^2\mathcal{L}(f(x;\theta),y))^{1/2}\big)\big((\nabla_f^2\mathcal{L}(f(x;\theta),y))^{1/2}\,\mathbf{J}_{z_i} f(x;\theta)\,a_{i-1}\big)\Big] \tag{7}$$

$$= \mathbb{E}\left[(\bar{g}^\top a_{i-1})^\top(\bar{g}^\top a_{i-1})\right] = \mathbb{E}\left[(\bar{g}\otimes a_{i-1})^\top(\bar{g}\otimes a_{i-1})\right] = \mathbb{E}\left[(\bar{g}^\top\bar{g})\otimes(a_{i-1}^\top a_{i-1})\right], \tag{8}$$

其中

$$\bar{g} = (\mathbf{J}_{z_i} f(x;\theta))^\top\,(\nabla_f^2\mathcal{L}(f(x;\theta),y))^{1/2}\,. \tag{9}$$

注 1（针对 $\bar{g}^\top\bar{g}$ 的蒙特卡洛低秩近似）。 由于 $\bar{g}$ 是一个形状为 $m\times d_i$的矩阵，其中 m 是 f 的输出维度，因此通常计算 $\bar{g}$ 的开销很大。因此， *[12]* 使用低秩蒙特卡洛近似来估计 $\nabla_f^2\mathcal{L}$（f（x； θ） y），从而 $\bar{g}^\top\bar{g}$。为此，我们需要使用损失 $\mathcal{L}$ 所依据的概率模型分布（例如，均方误差损失对应高斯分布，交叉熵对应类别分布）。具体而言，通过从该分布中采样

p_f（x），由神经网络输出 f（x；θ）定义，我们可以通过恒等式得到 $\nabla_f^2\mathcal{L}$（f（x；θ） y）的一个估计量

$$\nabla_f^2\mathcal{L}(f(x;\theta),y)=\mathbb{E}_{\hat{y}\sim p_f(x)}\left[\nabla_f\mathcal{L}(f(x;\theta),\hat{y})^\top\nabla_f\mathcal{L}(f(x;\theta),\hat{y})\right]. \tag{10}$$

关于这一方法及其替代方案的详尽参考资料，可见 *Dangel* 等人的附录 *A.2 [15]*。相应的秩一近似（记为 $\triangleq$） $\nabla_f^2\mathcal{L}(f(x;\ \theta))$ 为

$$\nabla_f^2\mathcal{L}(f(x;\theta),y)\triangleq\nabla_f\mathcal{L}(f(x;\theta),\hat{y})^\top\nabla_f\mathcal{L}(f(x;\theta),\hat{y}),$$

其中 $\hat{}y\sim p_f(x)$。相应地，我们可以利用这一秩一近似，通过以下方式估计 $\bar{g}^\top\bar{g}$

$$\bar{g}\triangleq\left(\mathbf{J}_{z_i}f(x;\theta)\right)^\top\nabla_f\mathcal{L}(f(x;\theta),\hat{y})=\nabla_{z_i}\mathcal{L}(f(x;\theta),\hat{y}). \tag{11}$$

In a与 $\bar{g}$ 类似，我们引入训练目标相对于预激活 z_i 的梯度，定义如下：

$$\mathrm{g}_i=\left(\mathbf{J}_{z_i}f(x;\theta)\right)^\top\nabla_f\mathcal{L}(f(x;\theta),y)=\nabla_{z_i}\mathcal{L}(f(x;\theta),y). \tag{12}$$

换言之，对于给定的层，令 $\mathrm{g}\in\mathbb{R}^{1\times d_i}$ 表示输出与真实值之间的损失的梯度，令 $\bar{g}\in\mathbb{R}^{m\times d_i}$ 表示神经网络 f 乘以损失函数海森矩阵平方根所得结果的导数（可根据注释 1 进行近似），二者均相对于给定层 i 的输出 z_i 计算。请注意，$\bar{g}$ 不等于 g，并且计算它们各自都需要一次反向传播遍（对于 $\bar{g}$ 的情况则可能需要多次）。因此，计算 $\bar{g}$ 的成本较高。

将 K-FAC [12] 近似应用于式 (8)，期望的克罗内克积可近似为期望的克罗内克积

$$\mathbf{G}=\mathbb{E}((\bar{g}^\top\bar{g})\otimes(\mathrm{a}^\top\mathrm{a}))\approx\mathbb{E}(\bar{g}^\top\bar{g})\otimes\mathbb{E}(\mathrm{a}^\top\mathrm{a}), \tag{13}$$

为清晰起见，我们在式 (8) 中省略 a_{i-1} 的下标，并记为 a；类似地，我们将 $\mathbf{G}_{W^{(i)}}$ 记为 $\mathbf{G}$。虽然期望的克罗内克积通常不等于期望的克罗内克积，但这一 K-FAC 近似（13）已被证明在实践中相当准确，并能保留广义高斯-牛顿矩阵 [12] 的“粗略结构”。式（13）中的 K-FAC 分解很方便，因为克罗内克积具有如下优良性质：对于两个矩阵 A，B 恒等式 $(A\otimes B)^{-1}=A^{-1}\otimes B^{-1}$，这显著简化了逆矩阵的计算。

在实践中，$\mathbb{E}(\bar{g}^\top\bar{g})$ 和 $\mathbb{E}(\mathrm{a}^\top\mathrm{a})$ 可以通过对大小为 b 的批次取平均来计算，具体如下

$$\mathbb{E}(\bar{g}^\top\bar{g})\simeq\bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}/b,\qquad\mathbb{E}(\mathrm{a}^\top\mathrm{a})\simeq\mathbf{a}^\top\mathbf{a}/b, \tag{14}$$

其中，我们将 g、$\bar{g}$ 和 a 的批次分别记为 $\mathbf{g}\in\mathbb{R}^{b\times d_i}$、$\bar{\boldsymbol{g}}\in\mathbb{R}^{rb\times d_i}$ 和 $\mathbf{a}\in\mathbb{R}^{b\times d_{i-1}}$。我们的层有 d_{i-1} 个输入、d_i 个输出，b 是批量大小，而 r 则表示输出数量 m 或近似的秩，具体取决于备注 1。相应地，GGN 矩阵的 K-FAC近似及其逆矩阵可简洁地表示为

$$\mathbf{G}\approx(\bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}})\otimes(\mathbf{a}^\top\mathbf{a})/b^2\qquad\mathbf{G}^{-1}\approx\left(\bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}\right)^{-1}\otimes\left(\mathbf{a}^\top\mathbf{a}\right)^{-1}\cdot b^2. \tag{15}$$

采用标准术语和设定后，我们现在介绍一种新颖的正则化更新步骤。首先，受 K-FAC 近似 (13) 的启发，Tikhonov正则化高斯-牛顿方法 (5) 可近似为

$$\theta^{(i)\prime}=\theta^{(i)}-\eta(\bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}/b+\lambda\mathbf{I})^{-1}\otimes(\mathbf{a}^\top\mathbf{a}/b+\lambda\mathbf{I})^{-1}\nabla_{\theta^{(i)}}\mathcal{L}(f(x;\theta)), \tag{16}$$

其中正则化参数为 $\lambda>0$。上述更新的结构启发我们注意到一个关键点：将 λ 的两次出现拆分为两个独立的正则化参数 $\lambda_{\mathbf{g}}$，$\lambda_{\mathbf{a}}>0$。将克罗内克因子化高斯-牛顿更新步定义为

$$\boldsymbol{\zeta}=\lambda_{\mathbf{g}}\lambda_{\mathbf{a}}(\bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}/b+\lambda_{\mathbf{g}}\mathbf{I})^{-1}\otimes(\mathbf{a}^\top\mathbf{a}/b+\lambda_{\mathbf{a}}\mathbf{I})^{-1}\nabla_{\theta^{(i)}}\mathcal{L}(f(x;\theta)), \tag{17}$$

我们得到简洁的更新方程

$$\theta^{(i)\prime}=\theta^{(i)}-\eta^*\boldsymbol{\zeta}. \tag{18}$$

当处于 $\eta^*=\frac{\eta}{\lambda_{\mathbf{g}}\lambda_{\mathbf{a}}}$ 和 $\lambda=\lambda_{\mathbf{g}}=\lambda_{\mathbf{a}}$ 的情况下，此更新步骤（18）等价于更新步骤（16）。这种等价性不会以任何方式限制 η^*，$\lambda_{\mathbf{g}}$，$\lambda_{\mathbf{a}}$，并且改变 $\lambda_{\mathbf{g}}$ 或 $\lambda_{\mathbf{a}}$ 并不意味着我们改变了学习率或步长 η^*。在（17）中使用乘法项 $\lambda_{\mathbf{g}}\lambda_{\mathbf{a}}$ 对 $\boldsymbol{\zeta}$ 进行参数化，可以使公式更便于分析。

在本文中，我们研究了迭代更新规则（18）的理论和实证性质，特别展示了正则化参数 $\lambda_{\mathbf{g}}$, $\lambda_{\mathbf{a}}$ 如何影响克罗内克因子化高斯-牛顿更新步 $\boldsymbol{\zeta}$。在分析克罗内克因子化高斯-牛顿更新步 $\boldsymbol{\zeta}$时，一个特别有用的工具是向量乘积恒等式,

$$\left(\left(\bar{\boldsymbol{g}}^{\top}\bar{\boldsymbol{g}}\right)^{-1}\otimes\left(\mathbf{a}^{\top}\mathbf{a}\right)^{-1}\right)\operatorname{vec}(\mathbf{g}^{\top}\mathbf{a})=\operatorname{vec}\left(\left(\bar{\boldsymbol{g}}^{\top}\bar{\boldsymbol{g}}\right)^{-1}\mathbf{g}^{\top}\mathbf{a}\left(\mathbf{a}^{\top}\mathbf{a}\right)^{-1}\right), \tag{19}$$

其中，相对于权重矩阵的梯度为$\mathbf{g}^{\top}\mathbf{a}$。

3 理论保证

本节将探讨式（17）中定义的克罗内克因子化高斯-牛顿更新方向 $\boldsymbol{\zeta}$ 的理论性质。回顾一下，$\boldsymbol{\zeta}$ 引入了蒂霍诺夫正则化，这也是基于二阶信息的方法实现中的常见做法。不出所料，我们证明，随着正则化参数 $\lambda_{\mathbf{g}}, \lambda_{\mathbf{a}}$ 减小，更新规则（18）在极限情况下会退化为经典的高斯-牛顿方法；因此，当 $\lambda_{\mathbf{g}}, \lambda_{\mathbf{a}}$ 较小时，变量 $\boldsymbol{\zeta}$ 表示高斯-牛顿方向。此外，随着正则化强度增大，该方法在极限情况下会收敛到传统的梯度下降更新步。

关键观察在于，当我们将两个克罗内克因子 $\bar{\boldsymbol{g}}^{\top}\bar{\boldsymbol{g}}$ 和 $\mathbf{a}^{\top}\mathbf{a}$ 的正则化分离开来，并考虑只有一个正则项较大的情形（准确地说是 $\lambda_{\mathbf{g}} \to \infty$），便能得到一种可高效计算的更新方向。我们证明，这种情形对应于一种近似的高斯-牛顿更新方案，其优越的数值表现随后在第 4 节中通过实验予以展示。

定理 1 ($\boldsymbol{\zeta}$**的性质)。** 如 (17) 中定义的基于 *K-FAC* 的更新步骤 $\boldsymbol{\zeta}$ 可以表示为

$$\boldsymbol{\zeta}=\left(\mathbf{I}_m-\frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}^{\top}\left(\mathbf{I}_b+\frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}\bar{\boldsymbol{g}}^{\top}\right)^{-1}\bar{\boldsymbol{g}}\right)\cdot\mathbf{g}^{\top}\cdot\left(\mathbf{I}_b-\frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^{\top}\left(\mathbf{I}_b+\frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^{\top}\right)^{-1}\right)\cdot\mathbf{a}\,. \tag{20}$$

此外，$\boldsymbol{\zeta}$ 具有以下渐近性质：

(i) 当 $\lambda_{\mathbf{g}}$ 时，$\lambda_{\mathbf{a}} \to 0$，$\frac{1}{\lambda_{\mathbf{g}}\lambda_{\mathbf{a}}}\boldsymbol{\zeta}$是高斯-牛顿步的 *K-FAC* 近似，即$\lim_{\lambda_{\mathbf{g}},\ \lambda_{\mathbf{a}}\to 0}\frac{1}{\lambda_{\mathbf{g}}\lambda_{\mathbf{a}}}\boldsymbol{\zeta}\approx\mathbf{G}^{-1}\nabla_{\theta^{(i)}}\mathcal{L}(f(x;\ \theta))$，其中 $\approx$表示式 (15) 中的 *K-FAC* 近似。

(ii) 当 $\lambda_{\mathbf{g}}$ 时，$\lambda_{\mathbf{a}} \to \infty$，$\boldsymbol{\zeta}$是梯度，即 $\lim_{\lambda_{\mathbf{g}},\ \lambda_{\mathbf{a}}\to\infty}\boldsymbol{\zeta}=\nabla_{\theta^{(i)}}\mathcal{L}(f(x;\ \theta))$。

证明详见补充材料。

我们希望证明，$\boldsymbol{\zeta}$ 定义良好且方向正确；这不仅适用于数值上接近于零的 $\lambda_{\mathbf{g}}$ 和 $\lambda_{\mathbf{a}}$，因为我们希望探索 $\lambda_{\mathbf{g}}$ 和 $\lambda_{\mathbf{a}}$ 的全部取值范围。因此，我们证明，无论如何选择 $\lambda_{\mathbf{g}}$ 和 $\lambda_{\mathbf{a}}$，$\boldsymbol{\zeta}$ 都是损失增加方向。

定理 2（正确性不受 $\boldsymbol{\zeta}$ 影响，与 $\lambda_{\mathbf{g}}$ 和 $\lambda_{\mathbf{a}}$无关）。 $\boldsymbol{\zeta}$ 是损失增加方向，与 $\lambda_{\mathbf{g}}$ 和 $\lambda_{\mathbf{a}}$的选择无关。

证明。 根据定义，$(\lambda_{\mathbf{g}}\mathbf{I}_m+\bar{\boldsymbol{g}}^{\top}\bar{\boldsymbol{g}}/b)$ 和 $(\lambda_{\mathbf{a}}\mathbf{I}_n+\mathbf{a}^{\top}\mathbf{a}/b)$ 均为正半定矩阵。因此，它们的逆矩阵 $(\lambda_{\mathbf{g}}\mathbf{I}_m+\bar{\boldsymbol{g}}^{\top}\bar{\boldsymbol{g}}/b)^{-1}$ 和 $(\lambda_{\mathbf{a}}\mathbf{I}_n+\mathbf{a}^{\top}\mathbf{a}/b)^{-1}$ 也为正半定矩阵。由于正半定矩阵的克罗内克积仍为正半定矩阵，条件矩阵 $((\lambda_{\mathbf{g}}\mathbf{I}_m+\bar{\boldsymbol{g}}^{\top}\bar{\boldsymbol{g}}/b)^{-1}\otimes(\lambda_{\mathbf{a}}\mathbf{I}_n+\mathbf{a}^{\top}\mathbf{a}/b)^{-1}\approx\mathbf{G}^{-1})$ 为正半定矩阵，因此更新步骤的方向仍然正确。 □

这引出了我们的主要贡献：根据对 $\boldsymbol{\zeta}$的表述，我们可以发现，当$\lambda_{\mathbf{g}} \to \infty$趋于极限时，公式（21）不依赖于 $\bar{\boldsymbol{g}}$。这在计算上非常有利，因为计算$\bar{\boldsymbol{g}}$的代价很高，需要额外进行一次甚至多次反向传播遍。此外，这还允许通过在$\mathbf{g}^{\top}$ 和$\mathbf{a}$之间乘以一个 $b \times b$ 矩阵来对梯度更新进行条件化，这种操作非常快。

定理 3（高效更新方向 / ISAAC）。 当 $\lambda_{\mathbf{g}} \to \infty$ 趋于极限时，更新步骤 $\boldsymbol{\zeta}$ 收敛于 极限 $\lim_{\lambda_{\mathbf{g}}\to\infty}\boldsymbol{\zeta}=\boldsymbol{\zeta}^{*}$，其中

$$\boldsymbol{\zeta}^{*}=\mathbf{g}^{\top}\cdot\left(\mathbf{I}_b-\frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^{\top}\left(\mathbf{I}_b+\frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^{\top}\right)^{-1}\right)\cdot\mathbf{a}\,. \tag{21}$$

(i) 这里，更新方向 $\boldsymbol{\zeta}^*$ 仅基于输入，不需要计算 $\bar{\boldsymbol{g}}$（这需要进行第二遍反向传播），因此效率很高。

(ii) 计算更新量 $\boldsymbol{\zeta}^*$ 的计算成本为 $\mathcal{O}(bn^2 + b^2n + b^3)$，其中 n 是每层的神经元数量。这包括计算梯度 $\nabla = \mathbf{g}^\top \mathbf{x}$ 的常规成本，后者为 $\mathcal{O}(bn^2)$，以及计算 $\boldsymbol{\zeta}^*$ 而非 ∇ 所产生的开销，后者为 $\mathcal{O}(b^2n+b^3)$。假设 $n \gg b$，该开销趋于零。对于 $b > n$，复杂度为 $\mathcal{O}(bn^2 + n^3)$。

证明。 我们首先证明性质 (21)。注意，根据 (22)， $\lambda_{\mathbf{g}} \cdot \left(\lambda_{\mathbf{g}} \mathbf{I}_m + \bar{\boldsymbol{g}}^\top \bar{\boldsymbol{g}}/b\right)^{-1}$ 在 $\lambda_{\mathbf{g}} \to \infty$的极限下收敛至 $\mathbf{I}_m$，因此 (21) 成立。

(i) 该陈述源于这样一个事实：项 $\bar{\boldsymbol{g}}$ 并未出现在等价表征中 (21) $\boldsymbol{\zeta}^*$。

（ii）我们首先注意到，矩阵 $\mathbf{a}\mathbf{a}^\top$ 的维度为 $b \times b$，并且可以在 $\mathcal{O}(b^2n)$ 时间内计算得到。接下来，矩阵

$$\left(\mathbf{I}_b - \frac{1}{b\lambda_{\mathbf{a}}} \mathbf{a}\mathbf{a}^\top \left(\mathbf{I}_b + \frac{1}{b\lambda_{\mathbf{a}}} \mathbf{a}\mathbf{a}^\top\right)^{-1}\right)$$

的形状为 $b \times b$，并且可以在 $\mathcal{O}(b^2n)$ 时间内与 $\mathbf{a}$ 相乘。 □

值得注意的是，（21）可以以趋于零的计算开销计算得到，并且只需对实现进行少量修改。具体而言，在反向传播步骤中，只需将 $\mathbf{g}^\top \mathbf{a}$ 表达式替换为（21）即可。由于这可以针对每一层独立完成，因此也适合仅应用于个别层。

正如我们在实验部分所看到的，在许多小批量场景（即 $b < n$）中， $\lambda_{\mathbf{g}}$ 的最优选择（或一个较好的选择）实际上处于趋于 ∞ 的极限处。这是一个出人意料的结果，由此得到高效且有效的 $\boldsymbol{\zeta}^* = \boldsymbol{\zeta}_{\lambda_{\mathbf{g}} \to \infty}$ 优化器。

注 2 （更新方向之间的关系 $\boldsymbol{\zeta}$ 与 $\boldsymbol{\zeta}^*$）。 将不带正则化的（即 $\lambda_{\mathbf{g}} \to 0, \lambda_{\mathbf{a}} \to 0$）式 (20) 中的更新方向$\boldsymbol{\zeta}$ 与 (21)中给出的 $\boldsymbol{\zeta}^*$ （即 $\lambda_{\mathbf{g}} \to \infty$）进行比较，可以直接看出， $\boldsymbol{\zeta}^*$ 对应于对 $\boldsymbol{\zeta}$ 进行特定的预条件处理，因为对于$M = \frac{1}{b\lambda_{\mathbf{g}}} \bar{\boldsymbol{g}}^\top \bar{\boldsymbol{g}}$. $\boldsymbol{\zeta}^* = M\boldsymbol{\zeta}$

作为我们所提出的更新方向$\boldsymbol{\zeta}^*$最后一个理论性质，我们将证明，在特定的神经网络$\boldsymbol{\zeta}^*$ 中，它与高斯-牛顿更新方向一致。

定理 4 （$\boldsymbol{\zeta}^*$ 在最后一层上是精确的）。 对于线性回归，或者更一般地，对于采用均方误差的网络最后一层， $\boldsymbol{\zeta}^*$ 是高斯-牛顿更新方向。

证明。 均方误差损失的海森矩阵是单位矩阵。相应地， $\bar{\boldsymbol{g}}^\top \bar{\boldsymbol{g}}$ 的期望值为 $\mathbf{I}$。因此， $\boldsymbol{\zeta}^* = \boldsymbol{\zeta}$。 □

备注 3。 方向 $\boldsymbol{\zeta}^*$ 对应于高斯-牛顿更新方向，其中 $\mathbf{G}$ 的近似形式可表示为 $\mathbf{G} \approx \mathbb{E}\left[\mathbf{I} \otimes (\mathrm{a}^\top \mathrm{a})\right]$.

备注 4(自然梯度的扩展)。 在某些情况下，相较于高斯-牛顿方法，使用基于费舍尔信息的自然梯度可能更为理想。与此设置的区别在于，在 (5) 中，广义高斯-牛顿矩阵 $\mathbf{G}$ 被经验费舍尔信息矩阵$\mathbf{F}$替代。

我们注意到，我们的理论也适用于 $\mathbf{F}$，并且 $\boldsymbol{\zeta}^*$ 也能高效近似自然梯度更新步骤 $\mathbf{F}^{-1}\nabla$。$\mathbf{F}$ 的第 i 个对角块（$\mathbf{F}_{\theta^{(i)}} = \mathbb{E}\left[(\mathrm{g}_i^\top \mathrm{g}_i) \otimes (a_{i-1}^\top a_{i-1})\right]$） 与广义高斯-牛顿矩阵 $\mathbf{G}$ （$\mathbf{G}_{\theta^{(i)}} = \mathbb{E}\left[(\bar{g}_i^\top \bar{g}_i) \otimes (a_{i-1}^\top a_{i-1})\right]$）的一个块具有相同的形式。因此，在我们的理论结果中，我们可以用$\mathbf{g}$ 替换 $\bar{\boldsymbol{g}}$，从而得到其关于$\mathbf{F}$的对应结果。

4 实验[1]

在上一节中，我们讨论了所提出的更新方向$\boldsymbol{\zeta}$ 和 $\boldsymbol{\zeta}^*$ 的理论性质，重点在于在小批量场景下，计算 $\boldsymbol{\zeta}^*$ 实际上“无需额外成本”。本节中，我们提供了实验证据，表明即使在深度学习中，$\boldsymbol{\zeta}^*$ 也是一个良好的更新方向。具体而言，我们证明：

[1]代码将在以下地址公开：GitHub.com/Felix-Petersen/isaac

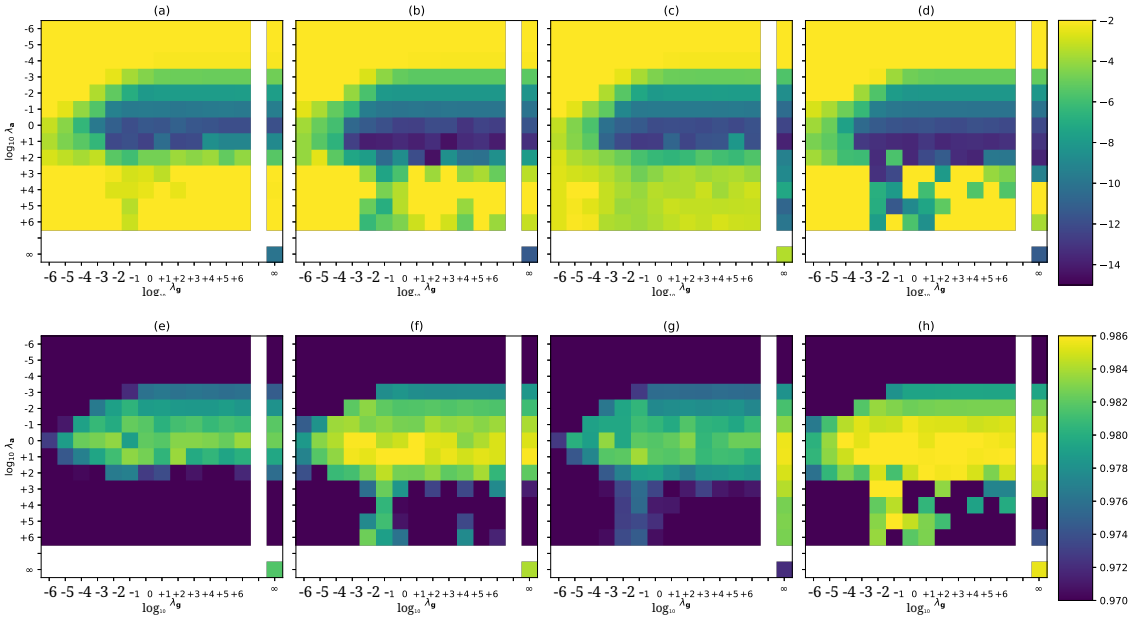


图1：MNIST 分类任务上的对数训练损失（上）和测试准确率（下）。坐标轴表示正则化参数 $\lambda_{\mathbf{g}}$ 和 $\lambda_{\mathbf{a}}$，采用以 10 为底的对数尺度。使用一个由 ReLU 激活的 5 层神经网络进行训练，每层分别包含 100 个（左侧，a、e）、400 个（中间，b、c、f、g）和 1600 个（右侧，d、h）神经元。除（c、g）使用带动量的 SGD 外，优化器均为 SGD。左上区域为 ζ，右上列为 ζ^*，右下角为 ∇ （梯度下降）。对于每个实验和三个区域中的每一个，我们都使用一个学习率，即 ζ、ζ^*、∇ 各自使用自己的学习率，以便公平比较不同方法；每个区域内的学习率保持不变。我们可以观察到，在 $\lambda_{\mathbf{g}} \to \infty$ 的极限下（即向右的极限），性能仍然良好，这表明了 ζ^*的效用。

(E1) ζ^* 的性能与 K-FAC 相近，但计算成本显著更低。(E2) 实验证据表明，我们提出的方法在小批量场景（$n \gg b$）下仍能保持性能。(E3) ζ^* 可用于单独的层，而其他层则只使用梯度 ∇。这仍能带来性能提升。(E4) ζ^* 还能提升训练更大型模型（如 BERT 和 ResNet）时的性能。(E5) ζ^* 的运行时间和内存需求与梯度下降相当。

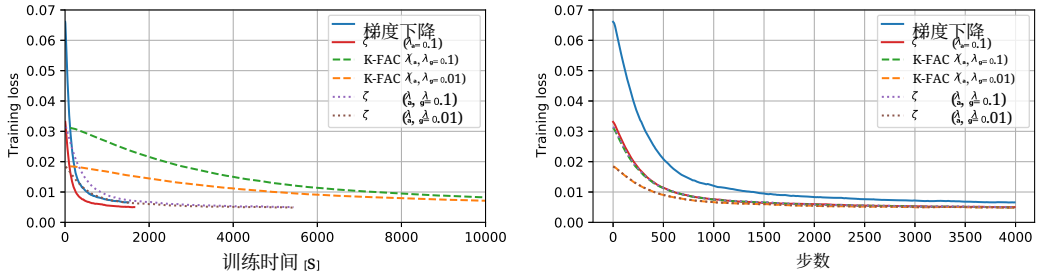
E1：正则化参数的影响

对于（E1），我们研究模型性能对正则化参数 $\lambda_{\mathbf{g}}$和 $\lambda_{\mathbf{a}}$的依赖关系。这里，我们在 MNIST 分类任务 [16]上训练一个 5 层深度神经网络，批量大小为 60，共训练 40 个训练轮次或 40000 步。

图1中的曲线表明，通过利用曲率信息进行条件化训练所带来的优势，既可以通过同时考虑层输入 $\mathbf{a}$ 和相对于随机样本的梯度 $\bar{\boldsymbol{g}}$来实现，也可以仅通过使用层输入 $\mathbf{a}$来实现。图中展示了 ζ 在不同 $\lambda_{\mathbf{g}}$ 和 $\lambda_{\mathbf{a}}$取值下的表现，它们的取值范围均为 10^{-6} 到 10^{6}。右侧一列展示了不同 $\lambda_{\mathbf{a}}$下的 ζ^*，即 $\lambda_{\mathbf{g}} = \infty$。右下角对应梯度下降，其对应于 $\lambda_{\mathbf{g}} = \infty$ 和$\lambda_{\mathbf{a}} = \infty$。

牛顿法或一般的 K-FAC 近似对应于较小的 $\lambda_{\mathbf{g}}$ 和 $\lambda_{\mathbf{a}}$区域。这里有趣的发现是，随着 $\lambda_{\mathbf{g}}$ 增大至 ∞，性能并未下降，也就是说，在图中从左向右移动时，性能没有受到影响。

此外，在图3中，我们考察了在 MNIST [16] 和 Fashion-MNIST [17]上使用均方误差损失训练的自编码器进行回归的情况。这里，我们遵循与上文相同的原则，也发现 ζ^* 表现良好。



Figu图2：使用梯度下降、K-FAC、ζ和 ζ^* 训练的 MNIST 自编码器的训练损失。比较 g the 按实际运行时间（左）和更新步骤数（右）衡量的性能。运行时间以一个 CPU 核心为准。

在图2中，我们比较了不同方法的损失。这里，我们分别考察单位时间内的损失（左）和每个步数对应的损失（右）。可以观察到，对于 $\lambda = 0.1$，K-FAC、ζ 和ζ^* 在每个更新步骤上的表现几乎相同（右）；而 ζ^* 的速度明显最快，其次是 ζ，传统的 K-FAC 实现则最慢（左）。另一方面，对于$\lambda = 0.01$，与 $\lambda = 0.1$ 相比，我们能够实现更快的收敛，但在这种情况下，只有 K-FAC 和 ζ 方法在数值上稳定，而ζ^* 不稳定。这意味着，在极小的 λ情况下，ζ^* 不如 K-FAC 和 ζ稳健；不过，当λ较小但适中（如 $\lambda = 0.1$）时，它的表现良好。对于 $\lambda < 0.01$，在这种设置下，K-FAC 和 ζ 在数值上也会变得不稳定。总的来说，我们观察到，K-FAC 可用的最小 λ 为 0。01 或 0。001，具体取决于模型和任务。考虑运行时间，ζ^* 表现最佳，其速度几乎与梯度下降一样快，而性能与 K-FAC 和 ζ相当。具体来说，梯度下降的一步只比 ζ^*快约 10 %。

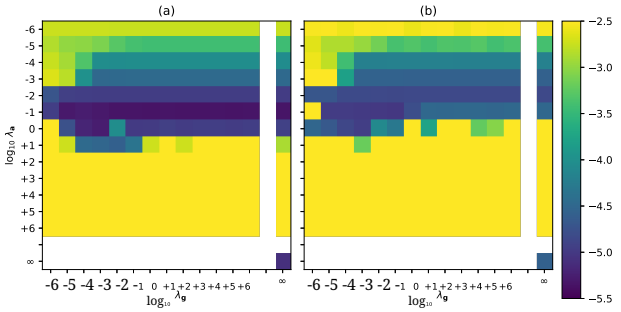


图3：在 MNIST（左）和 Fashion-MNIST（右）上训练自编码器。模型与 Botev 等人 [18], 所用模型相同，即一个采用 ReLU 激活的 6 层全连接模型，维度为 784-1000-500-30-500-1000-784。图中展示的是对数训练损失。

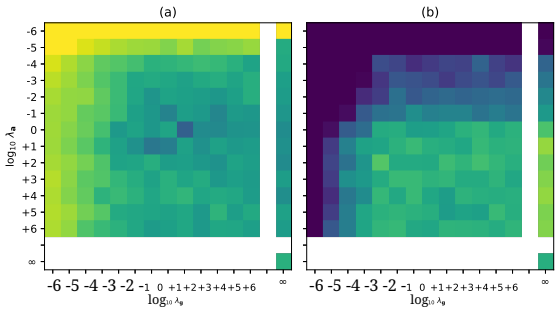


图4：在 MNIST 分类任务上训练一个每层含 400 个神经元的 5 层 ReLU 网络（与图1相同），但使用 Adam 优化器 [19]。

E2：小批量训练机制

对于（E2），从图1中可以看出，每层有 $n \in \{100, 400, 1\,600\}$ 个神经元时，即使批量大小仅为 60，训练效果也很好。此外，在所有其他实验中，我们使用的批量大小都较小，介于 8 和 100 之间。

E3：ζ^* 在单独的层中

在图5中，我们训练每层含 400 个神经元的 5 层全连接模型。这里，我们考虑在部分层中使用 ζ^*，而在其他层中使用默认梯度 ∇的设置。具体来说，我们考虑

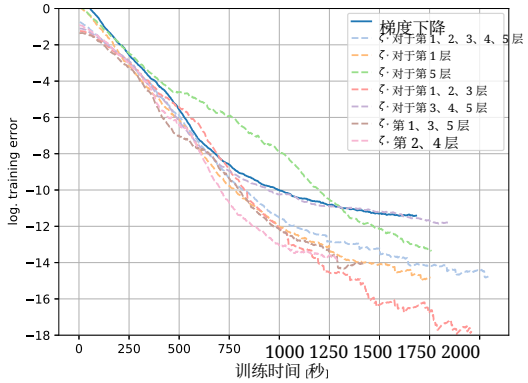


图5：仅在选定层中使用 ζ^*，在 MNIST 分类任务上进行训练。运行时间是在 CPU 上测得的。

设置中，分别更新所有层、第一层、最后一层、前三层、最后三层、奇数编号层和偶数编号层，更新方式为 ζ^*。我们观察到，所有带有 ζ^* 的设置表现都优于

表 1：在 CoLA、MRPC 和 STSB 文本分类任务上，对预训练的 BERT-Base（B-B）和 BERT-Mini（B-M）模型进行微调的结果。所有指标的数值越大越好。MCC 为马修斯相关系数。结果为 10 次运行的平均值。

方法 / 设置	CoLA（B-B）	CoLA（B-M）	MRPC（B-B）		STS-B（B-M）	
指标	MCC	MCC	Acc.	F1	皮尔逊相关系数	斯皮尔曼相关系数
梯度基线	54.20 ± 7.56	21.08 ± 2.88	82.52 ± 1.22	87.88 ± 0.74	76.98 ± 1.10	76.88 ± 0.79
ζ^*	57.62 ± 1.59	24.67 ± 2.62	83.28 ± 0.89	88.28 ± 0.70	81.09 ± 1.58	80.82 ± 1.57

普通梯度下降，但“ζ^* 针对第 3、4、5 层”这一设置除外，其表现与梯度下降大致相当。

E4：大规模模型

BERT 为展示 ζ^* 在大规模模型中同样具有实用价值，我们在三个自然语言任务上评估了它在 BERT [20]微调中的表现。表 1 总结了 BERT 微调任务的结果。对于“语言可接受性语料库”（CoLA） [21] 数据集，我们分别对 BERT-Base 和 BERT-Mini 模型进行微调，发现两种情况下的表现均优于梯度下降基线。对于“微软研究释义语料库”（MRPC） [22] 数据集，我们对 BERT-Base 模型进行微调，发现无论准确率还是 F1 分数都优于基线。最后，在“语义文本相似度基准”（STS-B） [23] 数据集上，我们对 BERT-Mini 模型进行微调，取得了高于基线的皮尔逊相关系数和斯皮尔曼相关系数。在 CoLA 和 MRPC 的训练中，我们使用了 Adam 优化器 [19]（该优化器推荐用于此任务和模型），并以 ζ^* 替代梯度；但 Adam 在 STS-B 上效果不佳。因此，对于 STS-B，我们使用带动量的随机梯度下降优化器进行评估。对于每种方法，我们都对超参数进行了网格搜索。值得注意的是，所有 BERT 实验的批量大小均为 8。

ResNet 此外，我们还进行了一项实验：使用 ζ^*训练 ResNet 的最后一层，同时使用梯度 ∇ 更新模型的其余部分。在此，我们使用批量大小为 100 的随机梯度下降，在 CIFAR-10 [25] 上训练 ResNet-18 [24] 。图 6 展示了测试准确率随训练轮次的变化。各方法的耗时相差不超过 1%。我们考虑了三种设置：使用动量和权重衰减的常规设置、仅使用动量的设置，以及不使用动量的原始随机梯度下降设置。结果表明，在这三种情况下，所提出的方法都优于随机梯度下降。虽然在默认训练设置下提升较小，但在不使用权重衰减和动量时，提升尤为显著。

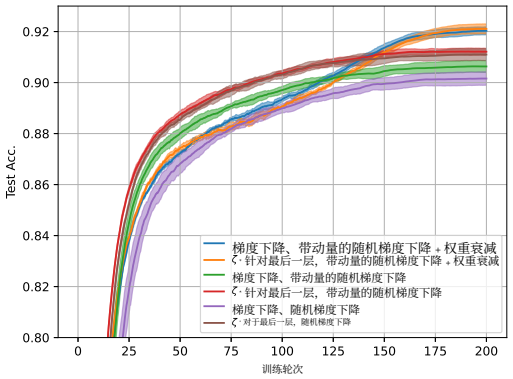


图 6：使用图像增强和余弦学习率调度在 CIFAR-10 上训练的 ResNet-18。为消融优化器，额外加入了两种设置：不使用权重衰减和不使用动量。结果取 5 次运行的平均值，彩色区域表示标准差。

E5：运行时间与内存

最后，我们还评估了各方法的运行时间和内存需求。运行时间评估结果见表 2。我们报告了使用 PyTorch [26] 以及（对于 K-FAC） backpack 库 [15] 测得的 CPU 和 GPU 运行时间。请注意，CPU 运行时间更能代表纯粹的计算成本，因为在 GPU 运行时间的前几行中，调用 GPU 的开销占主导地位。比较 GPU 上梯度与 ζ^* 的运行时间时，可以看到，无论模型大小如何，开销都在 $2.5\,s$ 左右。CPU 时间的开销也非常小：对于最大的模型，不到 1%；对于最小的模型，仅为 $1.3\,s$ 。相比之下，ζ^* 的运行时间约为梯度计算的 4 倍，而 K-FAC 的运行时间则要长得多。在内存方面，ζ^* （与其他方法相比）也只需要少量额外内存。

表 2：不同模型的运行时间和内存需求。运行时间是批量大小为 60 时在 MNIST 上每个 epoch 的训练时间，即 1 000 个训练步数。K-FAC 实现来自 backpack 库 [15]。GPU 为 Nvidia A6000。

	梯度			K-FAC			ζ			ζ^*		
模型	CPU 时间	GPU时间	内存	CPU 时间	GPU t.	内存	CPU 时间	GPU t.	内存	CPU 时间。	GPU 时间。	内存
5 层，每层 100 个神经元。	2.05 s	1.79 s	1。0 MB	62.78 s	17.63 s	11.5 MB	8.65 s	11.76 s	1.6 MB	3.34 s	4.07 s	1。0 MB
5 层，含 400 个神经元。	23.74 s	1.84 s	4.8 MB	218.48 s	32.00 s	22.4 MB	38.67 s	12.62 s	7。7 MB	13.62 s	4.19 s	4.9 MB
5 层，含 1,600 个 n。	187.87 s	1.93 s	51.0 MB	6985.48 s	156.48 s	212。2 MB	665.80 s	12.53 s	85.8 MB	291.01 s	4.49 s	51.4 MB
5 层，w/ 6 400 n.	3439.59 s	8.22 s	691.0 MB	—	1320.81 s	3155.3 MB	9673 s	31.87 s	1197.8 MB	3451.61 s	10.24 s	692.5 MB
自编码器	78.61 s	2.20 s	16。2 MB	1207.58 s	74.09 s	70.7 MB	193.25 s	14.19 s	33。8 MB	87.39 s	4.93 s	16.5 MB

备注 5 (实现)。 实现 ζ^* 时，可以将相应层的反向传播步骤替换为 (21)。由于所需的所有“要素”在常用的深度学习框架中都已具备，因此只需进行少量修改（与 *K-FAC* 和 ζ相比，后两者至少还需要额外进行一次反向传播）。

我们将发布实现的源代码。在附录中，我们给出了所提出方法的 PyTorch [26]实现（ζ^*）。

5 相关工作

我们的方法与 Martens 和 Grosse 提出的 K-FAC [12]相关。K-FAC 使用近似式（13）来近似神经网络经验风险的海森矩阵各个块。在 K-FAC 的大多数实现中，海森矩阵的非对角块也被设为零。K-FAC 声称的主要优势之一是，在大批量训练中，其速度快于随机梯度下降。尽管如此，近期的实证研究表明，计入大批量训练额外的超参数调优计算成本后，K-FAC 的这一优势便不复存在。有一系列研究将 K-FAC 的基本思想扩展到了卷积层 [27]。Botev 等 [18] 进一步拓展了这些思路，提出了 KFLR（一种克罗内克分解低秩近似）和 KFRA（一种对高斯-牛顿步进行克罗内克分解递归近似的方法）。Singh 和 Alistarh [28] 提出了 WoodFisher，这是一种基于伍德伯里矩阵逆的海森矩阵逆估计方法，并将其应用于神经网络压缩。Yao 等 [29] 提出了 AdaHessian，这是一种通过自适应估计海森矩阵来纳入损失函数曲率的二阶优化器。Frantar 等 [6] 提出了 M-FAC，这是一种通过维护近期（例如，1 000 个）梯度队列来实现的无矩阵自然梯度近似。这些研究与我们的方法有本质区别：它们的目标是近似费 Fisher 矩阵或高斯-牛顿矩阵逆与向量的乘积。相比之下，本文提出仅用高斯-牛顿矩阵的一个克罗内克因子来近似该矩阵；我们发现，这种方法在显著提升计算速度、减少内存占用的同时，能够取得良好性能。关于这一领域的综述，请参阅 Kunstner 等 [30] 和 Martens [31]。关于反向传播二阶量的技术细节综述，请参阅 Dangel等 [15], [32]
退一步来看，K-FAC 是众多用于训练神经网络的牛顿类方法之一。这类方法的其他重要例子包括子采样牛顿法 [33], [34] （通过对经验风险函数中的各项进行子采样，并计算子采样项的海森矩阵来近似海森矩阵）以及草图牛顿法 [3]–[5] （通过草图技术近似海森矩阵，例如将海森矩阵乘以一个随机矩阵，将其投影到更低维的空间）。另一种拟牛顿法 [35] 则利用梯度和海森矩阵的结构，将海森矩阵近似为分块对角矩阵，并进一步近似这些块。
K-FAC 与这一类方法的主要区别在于其经验性能更优，但缺乏理论依据。

6 结论

本文介绍了 ISAAC Newton，一种基于层输入的新型近似曲率方法。我们证明了它是正则化广义高斯-牛顿方法的一个特例，并通过实验证明了它的实用性。具体而言，我们的方法在小批量场景下的计算开销渐近趋于零，同时在各种基准问题上取得了具有竞争力的实证表现。

致谢

本研究得到了 IBM-MIT Watson 人工智能实验室、德国研究联合会卓越集群 EXC 2117“集体行为高级研究中心”（项目编号 390829875）、萨尔茨堡州 WISS 2025 项目 IDA 实验室（20102-F1901166-KZP 和 20204-WISS/225/197-2019），以及美国国家科学基金会（NSF）（资助编号 1916271、2027737 和 2113373）的支持。

参考文献

[1] N. Agarwal、B. Bullins 和 E. Hazan，《用于机器学习的线性时间二阶随机优化》，《机器学习研究期刊》（JMLR），2017年。[2] J. Nocedal 和 S. J. Wright，《数值优化》，第 2 版。美国纽约州纽约：Springer，2006年。[3] A. Gonen 和 S. Shalev-Shwartz，《使用草图预条件加速随机梯度下降》，arXiv 计算研究文献库（CoRR），2015年。[4] M. Pilanci 和 M. J. Wainwright，《牛顿草图：一种具有线性—二次收敛性的近线性时间优化算法》，《SIAM 优化期刊》，第 27 卷，2017年。[5] M. A. Erdogdu 和 A. Montanari，《子采样牛顿方法的收敛速度》，载于神经信息处理系统（NeurIPS）会议论文集，2015年。[6] E. Frantar、E. Kurtic 和 D. Alistarh，《M-FAC：二阶信息的高效无矩阵近似》，载于神经信息处理系统（NeurIPS）会议论文集，2021年。[7] N. Doikov 和 Y. Nesterov，《基于全局下界二阶模型的凸优化》，载于神经信息处理系统（NeurIPS）会议论文集，2020年。[8] Y. Nesterov 和 B. T. Polyak，《牛顿法的三次正则化及其全局性能》，《数学规划》，第 108 卷，2006年。[9] S. Becker 和 Y. Lecun，《利用二阶方法提升反向传播学习的收敛性》，1989年。[10] T. Schaul、S. Zhang 和 Y.LeCun，《不再为学习率烦恼》，载于国际机器学习会议（ICML），2013年。[11] Y. Ollivier，《神经网络的黎曼度量 I：前馈网络》，《信息与推断》，第 4 卷，第 108–153 页，2015年 6 月。[12] J. Martens 和 R. Grosse，《使用克罗内克因子近似曲率优化神经网络》，载于国际机器学习会议（ICML），2015年。[13] A. N. Tikhonov 和 V. Y. Arsenin，《病态问题的解》。W. H. Winston，1977年。[14] P. Chen，《海森矩阵与高斯—牛顿海森矩阵》，《SIAM 数值分析期刊》，2011年。[15] F. Dangel、F. Kunstner 和 P. Hennig，《Backpack：在反向传播中装入更多内容》，载于国际表征学习会议，2020年。[16] Y. LeCun、C. Cortes 和 C. Burges，《MNIST 手写数字数据库》，ATT 实验室，2010年。[17] H. Xiao、K. Rasul 和 R. Vollgraf，《Fashion-MNIST：用于机器学习算法基准测试的新型图像数据集》，arXiv 计算研究文献库（CoRR），2017年。[18] A. Botev、H. Ritter 和 D. Barber，《用于深度学习的实用高斯—牛顿优化》，载于国际机器学习会议（ICML），2017年。[19] D. Kingma 和 J. Ba，《Adam：一种随机优化方法》，载于国际表征学习会议（ICLR），2015年。[20] J. Devlin、M.-W. Chang、K. Lee 和 K. Toutanova，《BERT：用于语言理解的深度双向变换器预训练》，载于北美计算语言学协会分会：人类语言技术会议（NAACL-HLT），2018年。[21] A. Warstadt、A. Singh 和 S. R. Bowman，《神经网络可接受性判断》，《计算语言学协会汇刊》（ACL），2019年。[22] W. B. Dolan 和 C. Brockett，《自动构建句子释义语料库》，载于第三届国际释义研讨会（IWP2005）论文集，2005年。[23] D. Cer、M. Diab、E. Agirre、I. Lopez-Gazpio 和 L. Specia，《SemEval-2017 任务 1：语义文本相似度多语言与跨语言专项评测》，载于第十一届国际语义评测研讨会（SemEval-2017）论文集，加拿大温哥华：计算语言学协会，2017年。

[24] K. He、X. Zhang、S. Ren 和 J. Sun，《用于图像识别的深度残差学习》，载于国际计算机视觉与模式识别会议（*CVPR*）论文集，2016年。[25] A. Krizhevsky、V. Nair 和 G. Hinton，《CIFAR-10（加拿大高等研究院）》，2009年。[26] A. Paszke、S. Gross、F. Massa、A. Lerer、J. Bradbury、G. Chanan、T. Killeen、Z. Lin、N. Gimelshein、L. Antiga、A. Desmaison、A. Kopf、E. Yang、Z. DeVito、M. Raison、A. Tejani、S. Chilamkurthy、B. Steiner、L. Fang、J. Bai 和 S. Chintala，《PyTorch：一款命令式风格的高性能深度学习库》，载于神经信息处理系统（*NeurIPS*）会议论文集，2019年。[27] R. Grosse 和 J. Martens，《用于卷积层的克罗内克因子化近似费舍尔矩阵》，载于 国际机器学习会议（*ICML*），2016年。[28] S. P. Singh 和 D. Alistarh，《WoodFisher：用于神经网络压缩的高效二阶近似》，载于神经信息处理系统（*NeurIPS*）会议论文集，2020年。[29] Z. Yao、A. Gholami、S. Shen、M. Mustafa、K. Keutzer 和 M. W. Mahoney，《AdaHessian：一种用于机器学习的自适应二阶优化器》，载于 人工智能促进协会年会，2021年。[30] F. Kunstner、L. Balles 和 P. Hennig，《经验费舍尔近似在自然梯度下降中的局限性》，载于神经信息处理系统（*NeurIPS*）会议论文集，2019年。[31] J. Martens，《关于自然梯度方法的新见解与新视角》，《机器学习研究期刊》，2020年。[32] F. Dangel、S. Harmeling 和 P. Hennig，《用于前馈架构的模块化分块对角曲率近似》，载于 国际人工智能与统计会议（*AISTATS*），2020年。[33] F. Roosta-Khorasani 和 M. W. Mahoney，《子采样牛顿方法 I：全局收敛算法》，*arXiv* 计算研究文献库（*CoRR*），2016年。[34] P. Xu、J. Yang、F. Roosta、C. R´e 和 M. W. Mahoney，《采用非均匀采样的子采样牛顿方法》，载于 神经信息处理系统（*NeurIPS*）会议论文集，2016年。[35] D. Goldfarb、Y. Ren 和 A. Bahamou，《训练深度神经网络的实用拟牛顿方法》，载于第 *34* 届神经信息处理系统国际会议论文集，2020年。

一个 PyTorch 实现

下面展示了 ISAAC 在全连接层中的 PyTorch [26] 实现。我们用红色矩形标出了重要部分（即样板代码以外的部分）。

导入 torch

类 ISAACLinearFunction(torch.autograd.Function)：@staticmethod 定义 forward(ctx, input, weight, bias, la, inv_类型)：ctx.save_用于_反向传播(input, weight, bias) ctx.la = la 如果 inv_类型 == 'cholesky_逆矩阵'：ctx.inverse = torch.cholesky_逆矩阵 elif inv_类型 == '逆矩阵'：ctx.inverse = torch.inverse 否则：引发 NotImplementedError(inv_类型) 返回 input @ weight.T + (bias if bias i s not None else 0

)

```
@staticmethod
def 反向传播(ctx,grad_output):
    输入, 权重, 偏置 = ctx.已保存_张量
    如果 ctx.需要_输入_梯度[0]:
        梯度_0 = 梯度_输出 @ 权重
    否则:
        梯度_0 =无

    如果ctx.需要_输入_梯度[1]:
        aaT = 输入 @ input.T / 梯度_输出.shape[0]I_b = torch.eye(aaT.shape[0], device=aaT.device, dtype=aaT.dtype) aaT_IaaT_逆矩阵 = aaT @ ctx.inverse(aaT / ctx.la + I_b) 梯度_1 = 梯度_输出.T @ ( I_b - 1. / ctx.la* aaT_IaaT_逆矩阵 ) @ 输入

    否则:
        梯度_1 =无

    返回( 梯度_0, 梯度_1, 梯度_输出.求和(0, keepdim=真) 如果bias 不为无否则无, 无, 无 , 无 , )
```

类 ISAACLinear(torch.nn.Linear): 定义 __初始化__(self, 输入_特征, 输出_特征, la, 逆矩阵_类型='逆矩阵', **kwargs): super(ISAACLinear, self).__初始化__(输入_特征=输入_特征, 输出_特征=输出_特征,**kwargs) self.la = la self.逆矩阵_类型 = 逆矩阵_类型

```
def forward(self, input: torch.Tensor) ->torch.Tensor: return
ISAACLinearFunction.apply( input, self.weight, self.bias.unsqueeze(0) ifself.bias i
s not None else None, self.la, self.inv_type )
```

B 实现细节

除非另有说明，对于所有实验，我们都在以下网格中调整学习率：(1, 0.3,0.1,0.03,0.01,0.003,0.001)。我们确认，该范围涵盖了所有合理的学习率范围。具体而言，对于每一项实验，我们都确保不存在范围之外且效果更好的学习率。

对于所有语言模型实验，我们均使用相应的 Huggingface PyTorch 实现。

其他所有超参数细节均见正文。

C 补充证明

定理 1 的证明。我们首先证明，按（17）定义的 $\boldsymbol{\zeta}$ 可以表示为（20）的形式。事实上，利用（19）、Woodbury 矩阵恒等式并对逆矩阵进行正则化后，可以看出

$$
\begin{aligned}
\boldsymbol{\zeta} &= \lambda_{\mathbf{g}}\lambda_{\mathbf{a}}(\bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}/b + \lambda_{\mathbf{g}}\mathbf{I})^{-1} \otimes (\mathbf{a}^\top\mathbf{a}/b + \lambda_{\mathbf{a}}\mathbf{I})^{-1}\mathbf{g}^\top\mathbf{a} \\
&= \lambda_{\mathbf{g}}\lambda_{\mathbf{a}} \cdot \left(\lambda_{\mathbf{g}}\mathbf{I}_m + \bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}/b\right)^{-1}\mathbf{g}^\top\mathbf{a}\left(\lambda_{\mathbf{a}}\mathbf{I}_n + \mathbf{a}^\top\mathbf{a}/b\right)^{-1} \\
&= \lambda_{\mathbf{g}}\lambda_{\mathbf{a}} \cdot \left(\frac{1}{\lambda_{\mathbf{g}}}\mathbf{I}_m - \frac{1}{b{\lambda_{\mathbf{g}}}^2}\bar{\boldsymbol{g}}^\top\left(\mathbf{I}_b + \frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}\bar{\boldsymbol{g}}^\top\right)^{-1}\bar{\boldsymbol{g}}\right) \\
&\quad \mathbf{g}^\top\mathbf{a}\left(\frac{1}{\lambda_{\mathbf{a}}}\mathbf{I}_n - \frac{1}{b{\lambda_{\mathbf{a}}}^2}\mathbf{a}^\top\left(\mathbf{I}_b + \frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^\top\right)^{-1}\mathbf{a}\right) \\
&= \left(\mathbf{I}_m - \frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}^\top\left(\mathbf{I}_b + \frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}\bar{\boldsymbol{g}}^\top\right)^{-1}\bar{\boldsymbol{g}}\right) \cdot \mathbf{g}^\top \\
&\quad \cdot \mathbf{a} \cdot \left(\mathbf{I}_n - \frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}^\top\left(\mathbf{I}_b + \frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^\top\right)^{-1}\mathbf{a}\right) \\
&= \left(\mathbf{I}_m - \frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}^\top\left(\mathbf{I}_b + \frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}\bar{\boldsymbol{g}}^\top\right)^{-1}\bar{\boldsymbol{g}}\right) \cdot \mathbf{g}^\top \\
&\quad \cdot \left(\mathbf{a} - \frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^\top\left(\mathbf{I}_b + \frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^\top\right)^{-1}\mathbf{a}\right) \\
&= \left(\mathbf{I}_m - \frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}^\top\left(\mathbf{I}_b + \frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}\bar{\boldsymbol{g}}^\top\right)^{-1}\bar{\boldsymbol{g}}\right) \cdot \mathbf{g}^\top \\
&\quad \cdot \left(\mathbf{I}_b - \frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^\top\left(\mathbf{I}_b + \frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}\mathbf{a}^\top\right)^{-1}\right) \cdot \mathbf{a}
\end{aligned}
$$

为证明断言 (i)，我们注意到，根据式 (17)

$$
\begin{aligned}
&\lim_{\lambda_{\mathbf{g}},\lambda_{\mathbf{a}}\to 0}\frac{1}{\lambda_{\mathbf{g}}\lambda_{\mathbf{a}}}\boldsymbol{\zeta} \\
&= \lim_{\lambda_{\mathbf{g}},\lambda_{\mathbf{a}}\to 0}(\bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}/b + \lambda_{\mathbf{g}}\mathbf{I})^{-1} \otimes (\mathbf{a}^\top\mathbf{a}/b + \lambda_{\mathbf{a}}\mathbf{I})^{-1}\mathbf{g}^\top\mathbf{a} \\
&= (\bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}})^{-1} \otimes (\mathbf{a}^\top\mathbf{a})^{-1}\mathbf{g}^\top\mathbf{a} \\
&\approx \mathbf{G}^{-1}\mathbf{g}^\top\mathbf{a},
\end{aligned}
$$

其中，第一个等式使用了（17）中 $\boldsymbol{\zeta}$ 的定义。第二个等式源于矩阵求逆的连续性，最后一个近似等式则由 K-FAC 近似（15）得到。

为证明断言（ii），我们分别考虑 $\lim_{\lambda_{\mathbf{g}}\to\infty}$ 和 $\lim_{\lambda_{\mathbf{a}}\to\infty}$，即

$$\lim_{\lambda_{\mathbf{g}}\to\infty} \lambda_{\mathbf{g}} \cdot \left(\lambda_{\mathbf{g}}\mathbf{I}_m + \bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}/b\right)^{-1} \tag{22}$$
$$= \lim_{\lambda_{\mathbf{g}}\to\infty} \left(\mathbf{I}_m + \frac{1}{b\lambda_{\mathbf{g}}}\bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}\right)^{-1} = \mathbf{I}_m,$$

and

$$\lim_{\lambda_{\mathbf{a}}\to\infty} \lambda_{\mathbf{a}} \cdot \left(\lambda_{\mathbf{a}}\mathbf{I}_n + \mathbf{a}^\top\mathbf{a}/b\right)^{-1} \tag{23}$$
$$= \lim_{\lambda_{\mathbf{a}}\to\infty} \left(\mathbf{I}_n + \frac{1}{b\lambda_{\mathbf{a}}}\mathbf{a}^\top\mathbf{a}\right)^{-1} = \mathbf{I}_n.$$

这意味着

$$\lim_{\lambda_{\mathbf{g}},\lambda_{\mathbf{a}}\to\infty} \lambda_{\mathbf{g}} \left(\lambda_{\mathbf{g}}\mathbf{I}_m + \bar{\boldsymbol{g}}^\top\bar{\boldsymbol{g}}/b\right)^{-1} \cdot \mathbf{g}^\top \tag{24}$$
$$\cdot \mathbf{a} \cdot \lambda_{\mathbf{a}} \left(\lambda_{\mathbf{a}}\mathbf{I}_n + \mathbf{a}^\top\mathbf{a}/b\right)^{-1}$$
$$= \mathbf{I}_m \cdot \mathbf{g}^\top\mathbf{a} \cdot \mathbf{I}_n = \mathbf{g}^\top\mathbf{a},$$

由此得证。 □

D 附加实验

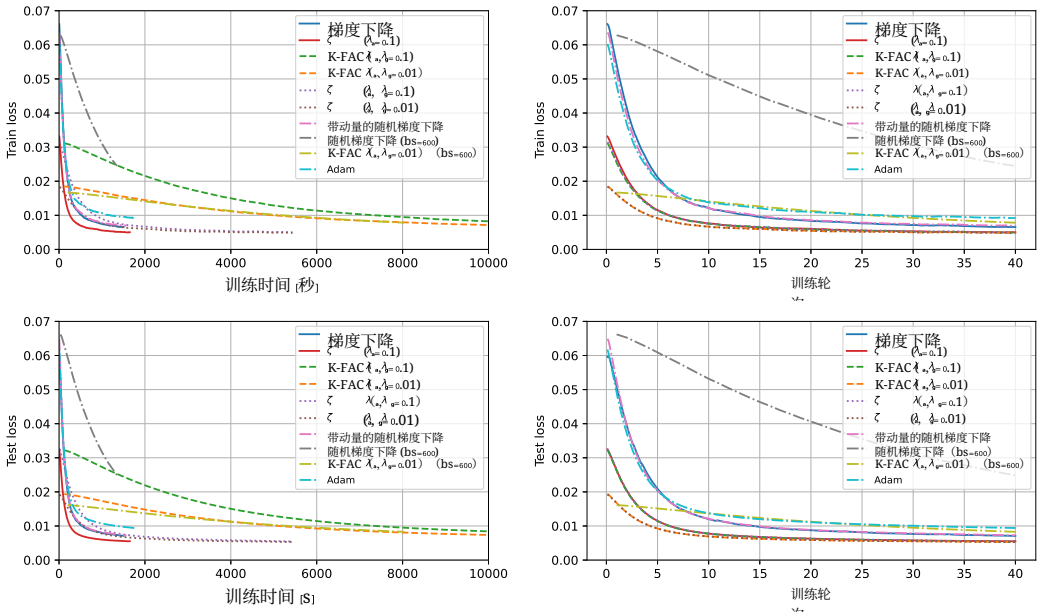


图 7：使用梯度下降、K-FAC、ζ、ζ*、带动量的随机梯度下降、采用更大批量大小（600）的随机梯度下降、采用更大批量大小（600）的 K-FAC，以及 Adam 训练的 MNIST 自编码器的训练损失。比较按实际运行时间（左）和训练轮次（右）衡量的性能。我们同时展示训练损失（上）和测试损失（下）。运行时间按一个 CPU 核心计算。

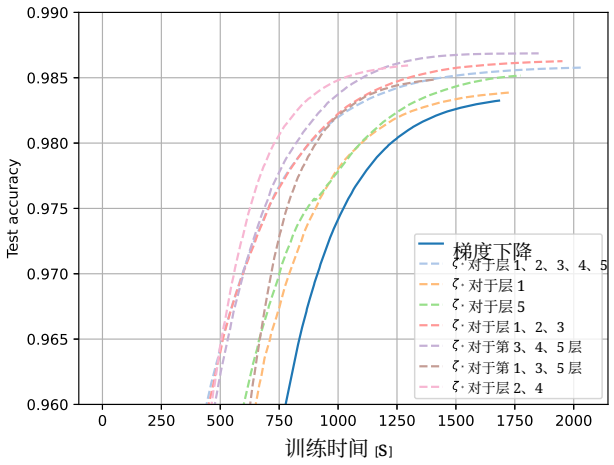


图8：在选定的层中仅使用 ζ^* 训练 MNIST 分类任务时的测试准确率。运行时间为 CPU 上的测量结果。

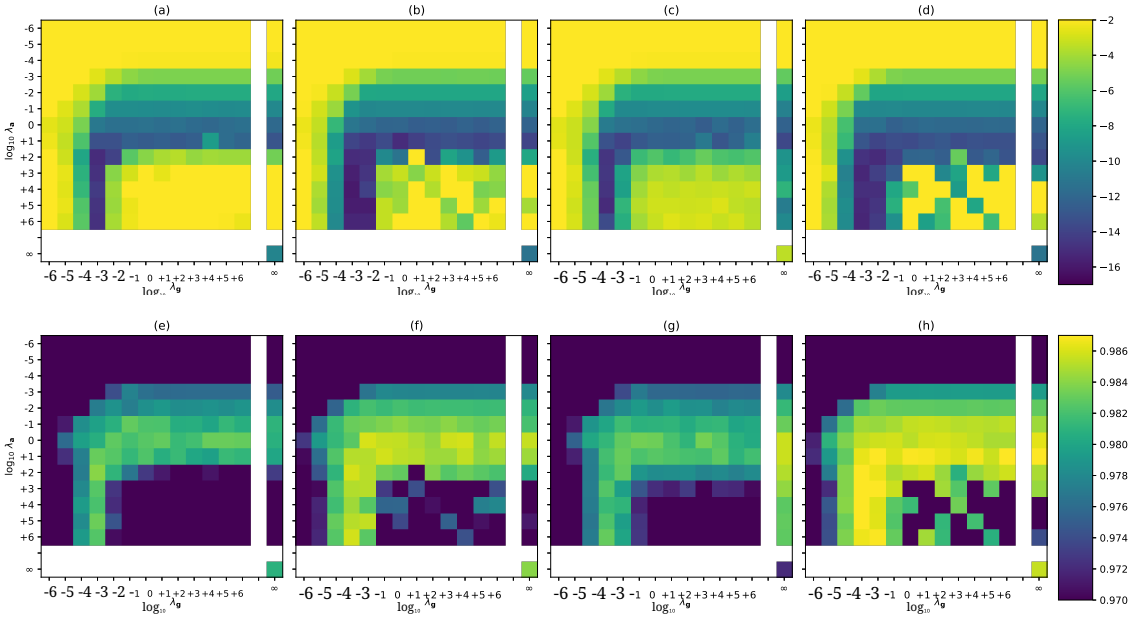


图9：使用备注 4 中的基于 Fisher 的自然梯度公式，重现图1中的实验。实验设置的说明见图1的图注。我们观察到，当 $\lambda_{\mathbf{g}}$ 较大时，其表现与图1相似；这符合预期，因为当 $\lambda_{\mathbf{g}} \to \infty$ 趋于极限时，两者相同。此外，我们观察到，在这种基于费舍尔信息的自然梯度的情形下，不仅在 $\lambda_{\mathbf{g}} \to \infty$ 趋于极限时，而且在 $\lambda_{\mathbf{a}} \to \infty$ 趋于极限时，都能取得良好表现。此外，在这一特定实验中，与 $\lambda_{\mathbf{g}} \to \infty$相比，$\lambda_{\mathbf{a}} \to \infty$ 的最优表现略好；但与 $\lambda_{\mathbf{g}} \to \infty$ 情形下对 $\lambda_{\mathbf{a}}$ 变化的敏感度相比， $\lambda_{\mathbf{a}} \to \infty$ 对 λ_g 的变化更为敏感。这一现象在图1的实验中也有所体现（程度较轻）。值得指出的是，与 $\lambda_{\mathbf{a}} \to \infty$相比，$\lambda_{\mathbf{g}} \to \infty$ 的情形（即 $\boldsymbol{\zeta}^*$）在计算上更高效。