计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(4):118−124 [doi: 10.15888/j.cnki.csa.007869]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Sherman-Morrison 公式的 K-FAC 算法①

刘小雷, 高凯新, 王　勇

(天津大学 数学学院, 天津 300350)
通讯作者: 王　勇, E-mail: wang_yong@tju.edu.cn

摘　要: 二阶优化方法可以加速深度神经网络的训练, 但是二阶优化方法巨大的计算成本使其在实际中难以被应用. 因此, 近些年的研究提出了许多近似二阶优化方法的算法. K-FAC 算法提供了一种近似自然梯度的有效方法. 在 K-FAC 算法的基础上, 结合拟牛顿方法的思想, 提出了一种改进的 K-FAC 算法. 在开始的少量迭代中利用 K-FAC 算法计算, 在后续迭代中构造秩–1 矩阵, 通过 Sherman-Morrison 公式进行计算, 大大降低了计算复杂度. 实验结果表明, 改进的 K-FAC 算法比 K-FAC 算法有相似甚至是更好的实验表现. 特别的, 改进的 K-FAC 算法与 K-FAC 算法相比减少了大量的训练时间, 而且与一阶优化方法相比, 在训练时间上仍具有一定的优势.

关键词: 深度学习; 二阶优化方法; K-FAC 算法; Sherman-Morrison 公式; Fisher 信息矩阵

引用格式: 刘小雷,高凯新,王勇.基于 Sherman-Morrison 公式的 K-FAC 算法.计算机系统应用,2021,30(4):118–124. http://www.c-s-a.org.cn/1003-3254/7869.html

K-FAC Algorithm Based on Sherman-Morrison Formula

LIU Xiao-Lei, GAO Kai-Xin, WANG Yong

(School of Mathematics, Tianjin University, Tianjin 300350, China)

Abstract: Second-order optimization can accelerate the training of deep neural networks, but its huge computational cost hinders it from applications. Therefore, many algorithms have been proposed to approximate second-order optimization in recent studies. The K-FAC algorithm can approximate natural gradient, based on which an improved K-FAC algorithm is proposed according to the quasi-Newton method. The K-FAC algorithm is applied to the first few iterations. Then, a rank-one matrix is built, and its inverse matrix is computed by the Sherman-Morrison formula, greatly reducing computational complexity. The experimental results prove that the improved K-FAC algorithm has similar or even better performance than the original K-FAC, especially with much less training time. It also has the advantage over first-order optimization in regard to training time.

Key words: deep learning; second-order optimization methods; K-FAC algorithm; Sherman-Morrison formula; Fisher information matrix

近些年来, 深度学习已经在计算机视觉和自然语言处理等领域取得了重要的进展. 然而随着研究的深入, 模型越来越复杂, 往往需要耗费大量的训练时间和计算成本. 因此, 采用有效的训练方法是十分有必要的. 以随机梯度下降 (SGD) 为代表的一阶优化方法是当前深度学习中最常用的方法. 近些年来, 一系列 SGD 的改进算法被提出并也被广泛于应用深度学习中, 比如, 动量 SGD (SGDM[1]), Adagrad[2], Adam[3]. 这些一阶优

① 基金项目: 国家自然科学基金 (11871051)
Foundation item: National Natural Science Foundation of China (11871051)
收稿时间: 2020-08-16; 修改时间: 2020-09-10; 采用时间: 2020-09-18; csa 在线出版时间: 2021-03-30

化方法具有更新速度快, 计算成本低等优点, 但是也具有收敛速度慢, 需要进行复杂调参等缺点.

通过曲率矩阵修正一阶梯度, 二阶优化方法可以得到更为有效的下降方向, 使得收敛速度大大加快, 减少了迭代次数和训练时间. 对于有着上百万甚至更多参数的深度神经网络而言, 其曲率矩阵的规模是十分巨大的, 这样大规模的矩阵的计算, 存储和求逆在实际计算中是难以实现的. 因此, 对曲率矩阵的近似引起了广泛的研究. 其中最基本的方法是对角近似, 其在实际计算中取得了较好的效果, 但是在近似过程中丢失了很多曲率矩阵的信息, 而且忽略了参数之间的相关性. 在对角近似的基础上, 一些更为精确的算法也被提出, 这些算法不再局限于曲率矩阵的对角元素, 同时也考虑了非对角元素的影响. 这些方法对曲率矩阵的研究都取得的一定了进展[4–8]. 但是如何在深度学习中更加有效地利用曲率矩阵得到更有效的算法, 仍然是应用二阶优化方法面临的重要挑战.

自然梯度下降可以被视为一种二阶优化方法, 其中自然梯度定义为梯度与模型的 Fisher 信息矩阵的乘积. 该方法最初在文献 [9] 中被提出, 其在深度学习中有着重要的应用. 文献 [10] 中提出了一种近似全连接神经网络中自然梯度的有效方法, 称之为 K-FAC. K-FAC 算法首先通过假设神经网络各层之间的数据是独立的, 将 Fisher 信息矩阵近似为块对角矩阵; 将每个块矩阵近似为两个更小规模矩阵的克罗内克乘积, 通过克罗内克乘积的性质可以有效计算近似后的 Fisher 信息矩阵及其逆矩阵. K-FAC 有效地减少了自然梯度下降的计算量并取得了很好的实验效果. 这一方法也被应用到其他的神经网络中, 包括卷积神经网络[11–13], 循环神经网络[14], 变分贝叶斯神经网络[15,16]. 通过设计 K-FAC 的并行计算框架, 其在大规模问题中的有效性也得到了验证[17,18].

K-FAC 算法在众多问题中都有着很好的表现, 在保持 K-FAC 算法有效性的前提下, 进一步降低计算成本和减少计算时间是非常值得研究的问题. 在本文中, 我们基于 K-FAC 算法的近似思想, 结合拟牛顿法的思想, 提出了一种校正 Fisher 信息矩阵的有效方法. 该方法的主要思想是先用 K-FAC 方法进行若干次迭代, 保存逆矩阵的信息; 在后续迭代中利用该逆矩阵以及新的迭代中产生的信息, 结合 Sherman-Morrison 公式进行求逆计算, 大大减少了迭代时间. 实验中, 改进的 K-FAC 算法比 K-FAC 算法有相同甚至更好的训练效果, 同时大大减少了计算时间.

1 K-FAC 算法

神经网络的训练目标是获得合适的参数θ来最小化目标函数$h(\theta)$, 给定损失函数:

$$\mathbb{E}[-\log p(y|x,\theta)]$$

其中, x,y分别是训练输入和标签, θ是模型的参数向量, $p(y|x,\theta)$表示预测分布的密度函数. 那么 Fisher 信息矩阵的定义如下:

$$F=\mathbb{E}\left[\nabla_\theta \log p(y|x,\theta)(\nabla_\theta \log p(y|x,\theta))^{\mathrm{T}}\right] \tag{1}$$

文献 [10] 中给出了自然梯度的定义: $F^{-1}\nabla_\theta h$. 在实际计算中面临的主要挑战是计算自然梯度, 也就是计算逆矩阵F^{-1}及其与$\nabla_\theta h$的乘积. 在深度学习中, 由于 F 的规模太大, 直接计算F^{-1}是不切实际的. K-FAC 提供了一种近似F^{-1}的有效方法, 其近似过程可以分为两个步骤. 首先, K-FAC 将矩阵 F 按网络的各层分割成块矩阵, 通过假设不同层之间数据是独立的, 将 F 近似为块对角矩阵; 其次将每个块矩阵近似为两个更小规模矩阵的克罗内克乘积, 根据克罗内克乘积求逆及运算的相关性质, 大大减少了计算量.

考虑一个有 L 层的神经网络, a_{l-1}, s_l分别表示第l层的输入和输出, 那么$s_l=W_l a_{l-1}$, 其中W_l为第l层的权重矩阵, $l\in\{1,2,\cdots,L\}$. 方便起见, 定义如下的符号:

$$\begin{cases}\theta_l=\mathrm{vec}(W_l)\\ \mathcal{D}\theta=-\nabla_\theta \log p(y|x,\theta)\\ g_l=\mathcal{D}_{s_l}=-\nabla_{s_l}\log p(y|x,\theta)\end{cases}$$

那么 Fisher 信息矩阵可以被表示为:

$$F=\mathbb{E}\left[\mathcal{D}\theta\mathcal{D}\theta^{\mathrm{T}}\right]$$

即为:

$$\begin{bmatrix}\mathbb{E}[\mathrm{vec}(\mathcal{D}\theta_1)\mathrm{vec}(\mathcal{D}\theta_1)^{\mathrm{T}}] & \cdots & \mathbb{E}[\mathrm{vec}(\mathcal{D}\theta_1)\mathrm{vec}(\mathcal{D}\theta_L)^{\mathrm{T}}]\\ \vdots & \ddots & \vdots\\ \mathbb{E}[\mathrm{vec}(\mathcal{D}\theta_L)\mathrm{vec}(\mathcal{D}\theta_1)^{\mathrm{T}}] & \cdots & \mathbb{E}[\mathrm{vec}(\mathcal{D}\theta_L)\mathrm{vec}(\mathcal{D}\theta_L)^{\mathrm{T}}]\end{bmatrix}$$

因此 F 可以被看作一个$L\times L$的块矩阵. 定义$F_{ij}=\mathbb{E}\left[\mathrm{vec}(\mathcal{D}\theta_i)\mathrm{vec}(\mathcal{D}\theta_i)^{\mathrm{T}}\right]$, $i,j\in\{1,2,\cdots,L\}$. 通过假设神经网络各层之间数据的独立性, 也就是$F_{ij}=0(i\neq j)$, 那么 F 可以被近似为:

$$F=\mathrm{diag}(F_{11},F_{22},\cdots,F_{LL})$$

然而由于每个块矩阵F_{ll}的规模仍然很大, 因此需

要进一步近似. 每个块矩阵F_{ll}可以被写作:

$$\begin{aligned}F_{ll}=&\mathbb{E}\left[\mathrm{vec}(\mathcal{D}\theta_l)\mathrm{vec}(\mathcal{D}\theta_l)^{\mathrm{T}}\right]\\=&\mathbb{E}\left[(a_{l-1}\otimes g_l)(a_{l-1}\otimes g_l)^{\mathrm{T}}\right]\\=&\mathbb{E}\left[\left(a_{l-1}a_{l-1}^{\mathrm{T}}\right)\otimes\left(g_lg_l^{\mathrm{T}}\right)\right]\\\approx&\mathbb{E}\left[a_{l-1}a_{l-1}^{\mathrm{T}}\right]\otimes\mathbb{E}\left[g_lg_l^{\mathrm{T}}\right]\end{aligned}\tag{2}$$

其中, $\otimes$表示克罗内克乘积. 令, $A_{l-1,\,l-1}=\mathbb{E}\left[a_{l-1}a_{l-1}^{\mathrm{T}}\right]$, $G_{ll}=\mathbb{E}\left[g_lg_l^{\mathrm{T}}\right]$. 根据克罗内克乘积求逆的性质可以得到:

$$(A_{l-1,l-1}\otimes G_{ll})^{-1}=A_{l-1,\,l-1}^{-1}\otimes G_{ll}^{-1}$$

因此 Fisher 信息矩阵的逆矩阵可以被近似为:

$$\begin{aligned}F^{-1}=&\mathrm{diag}\left(F_{11}^{-1},F_{22}^{-1},\cdots,F_{LL}^{-1}\right)\\=&\mathrm{diag}\left(A_{00}^{-1}\otimes G_{11}^{-1},A_{11}^{-1}\otimes G_{22}^{-1},\cdots,A_{L-1,\,L-1}^{-1}\otimes G_{LL}^{-1}\right)\end{aligned}\tag{3}$$

为了保持训练的稳定性, 需要对克罗内克因子$A_{l-1,\,l-1}$和G_{ll}添加阻尼如下:

$$A_{l-1,\,l-1}+\pi_l\sqrt{\lambda}I,G_{ll}+\frac{\sqrt{\lambda}}{\pi_l}I\tag{4}$$

其中, λ是阻尼参数, π_l理论上可以是一个任意的正数, 但在实验中发现根据以下公式计算的π_l是一个更好的选择.

$$\pi_l=\sqrt{\frac{\mathrm{tr}(A_{l-1,\,l-1})/(d_{l-1}+1)}{\mathrm{tr}(D_{ll})/d_l}}$$

其中, d_{l-1}和d_l分别是矩阵$A_{l-1,\,l-1}$和G_{ll}的维度. 因此由上述内容可以得到训练中第l层参数的更新规则如下:

$$\theta_l^{(m+1)}\leftarrow\theta_l^{(m)}-\eta\left(\left(A_{l-1,l-1}^{(m)}+\pi_l\sqrt{\lambda}I\right)^{-1}\otimes\left(G_{ll}^{(m)}+\frac{\sqrt{\lambda}}{\pi_l}I\right)^{-1}\right)\nabla_\theta h^{(m)}\tag{5}$$

其中, η是学习率, m 表示迭代次数.

图 1 是 K-FAC 算法近似过程示意.

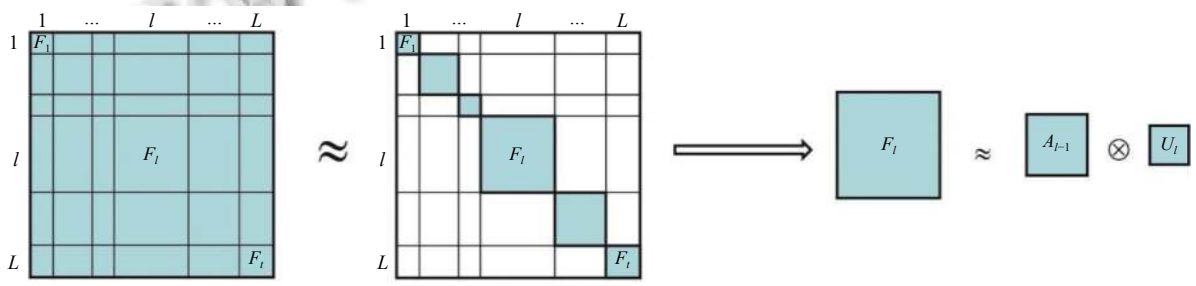

图 1 K-FAC 算法近似过程示意

2 算法改进

在实际计算中, Fisher 信息矩阵 F 的主对角线上的每个块矩阵的规模仍然很大, 直接对矩阵$\{F_{11},F_{22},\cdots,F_{LL}\}$进行求逆的计算成本很高. K-FAC 算法采用的方法是将 Fisher 信息矩阵近似为两个小规模矩阵的克罗内克乘积, 从而将求解大规模矩阵的逆转化为求解两个小规模矩阵的逆, 大大降低了求逆的成本. 在本文中我们结合 K-FAC 方法采取的这种近似方法, 给出了 Fisher 信息矩阵 F 的另一种近似, 其求逆的费用更低. 我们基于下面的 Sherman-Morrison 公式来改进 K-FAC 算法.

Sherman-Morriso 公式: 假设$X\in\mathbb{R}^{m\times m}$是可逆矩阵, $p,q\in\mathbb{R}^m$为任意列向量, 则可逆当且仅当$1+q^{\mathrm{T}}X^{-1}p\neq0$, 而且如果$X+pq^{\mathrm{T}}$可逆, 逆矩阵可以由以下公式得到:

$$\left(X+pq^{\mathrm{T}}\right)^{-1}=X^{-1}-\frac{X^{-1}pq^{\mathrm{T}}X^{-1}}{1+q^{\mathrm{T}}X^{-1}p}\tag{6}$$

由上述公式可以看出, 如果矩阵 X 的逆已知 (或者很容易求), 那么利用 Sherman-Morrison 公式可以将矩阵的求逆运算转化为矩阵向量乘积, 从而可以减少大量的计算时间. 因此我们根据 Sherman-Morrison 公式, 结合拟牛顿的算法思想, 提出了一种 K-FAC 算法的改进算法. 在实际计算中, 每次迭代均更新逆矩阵需要很高的计算成本, 因此实验中一般设置若干次迭代更新一次逆矩阵. 在下文中, 我们用 k 表示逆矩阵更新的次数. 我们提出的算法主要是对 K-FAC 算法的求逆运算进行了进一步的改进. 其主要思想是用 K-FAC 算法先进行 k 次求逆运算, 保存第 k 次求逆得到的逆矩阵信息; 后续迭代中利用该逆矩阵的信息以及在新的迭代中产生的信息, 结合 Sherman-Morrison 公式进行求逆运算. 下面我们以矩阵A_{00}, G_{11}为例说明改进的方法, 对于矩阵 A_{ll} ($l\in\{1,2,\cdots,L-1\}$) 和G_{ll} ($l\in\{2,3,\cdots,L\}$) 均采用相同的近似方法. 为方便表示, 我们在下文中省略下标.

(1) 首先, 按照 K-FAC 算法进行 k 次求逆运算, 在求出逆矩阵$\left(A^{(k)}\right)^{-1}$和$\left(G^{(k)}\right)^{-1}$后保留逆矩阵的信息.

(2) 其次, 矩阵$A^{(k+1)}$和$G^{(k+1)}$表示的是在第 k+1 次更新逆矩阵时计算得到的矩阵, 利用矩阵$A^{(k+1)}$和$G^{(k+1)}$构造向量$u^{(k+1)}$和$v^{(k+1)}$. 从现有文献中可以看出, 矩阵$A^{(k+1)}$和$G^{(k+1)}$的主对角线上的元素占主导性的信息, 因此我们选取其主对角线上的元素来构造向量$u^{(k+1)}$和$v^{(k+1)}$. 我们选取:

$$\begin{cases} (u^{(k+1)})_i = \sqrt{(A^{(k+1)})_{ii}} \\ (v^{(k+1)})_i = \sqrt{(G^{(k+1)})_{ii}} \end{cases}$$

那么$u^{(k+1)}\left(u^{(k+1)}\right)^{\mathrm{T}}$和$v^{(k+1)}\left(v^{(k+1)}\right)^{\mathrm{T}}$就是保留矩阵$A^{(k+1)}$和$G^{(k+1)}$主对角线元素的对角矩阵.

(3) 最后, 利用 Sherman-Morrison 公式可以得到:

$$\begin{aligned}\left(A^{(k+1)}\right)^{-1} &\approx \left(A^{(k)}+\alpha u^{(k+1)}\left(u^{(k+1)}\right)^{\mathrm{T}}\right)^{-1} \\ &= \left(A^{(k)}\right)^{-1} - \frac{\left(A^{(k)}\right)^{-1}u^{(k+1)}\left(u^{(k+1)}\right)^{\mathrm{T}}\left(A^{(k)}\right)^{-1}}{\alpha^{-1}+\left(u^{(k+1)}\right)^{\mathrm{T}}\left(A^{(k)}\right)^{-1}u^{(k+1)}}\end{aligned} \tag{7}$$

$$\begin{aligned}\left(G^{(k+1)}\right)^{-1} &\approx \left(G^{(k)}+\beta v^{(k+1)}\left(v^{(k+1)}\right)^{\mathrm{T}}\right)^{-1} \\ &= \left(G^{(k)}\right)^{-1} - \frac{\left(G^{(k)}\right)^{-1}v^{(k+1)}\left(v^{(k+1)}\right)^{\mathrm{T}}\left(G^{(k)}\right)^{-1}}{\beta^{-1}+\left(v^{(k+1)}\right)^{\mathrm{T}}\left(G^{(k)}\right)^{-1}v^{(k+1)}}\end{aligned} \tag{8}$$

其中, α, β是两个合适的正数.

在改进的 K-FAC 算法中, 主要是结合 Sherman-Morriso 公式对 Fisher 信息矩阵进行了近似, 因此改进的算法在计算矩阵A,G及其逆矩阵部分与 K-FAC 算法有所区别, 其余部分与 K-FAC 算法相同. 文献 [1] 中对 K-FAC 算法整体计算复杂度进行了详细分析, 因此在表 1 中, 我们主要给出了两种方法在计算矩阵A,G及其逆矩阵的计算复杂度对比.

表 1 K-FAC 和改进的 K-FAC 算法的求逆计算复杂度对比

算法	K-FAC算法	改进的K-FAC算法
计算矩阵A, G	$\mathrm{O}\left(tn^2\right)$	$\mathrm{O}\left(kn^2+(t-k)n\right)$
计算逆矩阵	$\mathrm{O}\left(tn^3\right)$	$\mathrm{O}\left(kn^3+(t-k)n^2\right)$

对于改进的 K-FAC 算法, 前k次与 K-FAC 算法一致. 在实际计算中, k的取值远远小于t. 比如在实验中我们选择k = 10, t = 39 100. 因此前t次的计算成本占比很低. 在后续的更新过程中, 改进的 K-FAC 算法一方面矩阵求逆部分都转化成了矩阵乘法, 计算复杂度由$\mathrm{O}\left(n^3\right)$降为$\mathrm{O}\left(n^2\right)$; 另一方面, 在后续迭代中, 我们仅需要计算矩阵 A 和G的主对角线元素, 可以直接将向量元素对应相乘, 不再需要进行矩阵乘法, 计算复杂度由$\mathrm{O}\left(n^2\right)$降为$\mathrm{O}(n)$. 因此, 通过上述两个方面改进的 K-FAC 算法可以减少大量的计算时间. 算法 1 总结了改进的 K-FAC 算法的流程.

算法 1. 改进的 K-FAC 算法

输入: 训练集 T, 学习率η, 阻尼参数λ, Fisher 信息矩阵及其逆矩阵的更新频率T_{FIM},T_{inv}, 逆矩阵更新次数k
输出: 模型参数θ

初始化参数A_{ll} ($l\in\{0,1,\cdots,L-1\}$)和G_{ll} ($l\in\{1,2,\cdots,L\}$), m=0,t=0;
While 未达到终止条件 do
  if $m\equiv 0(\mathrm{mod}\ T_{\mathrm{FIM}})$ **then**
    if $t<=k$ **then**
      根据式 (2) 计算因子$\{A_{ll}\}_{l=0}^{L-1}$, $\{G_{ll}\}_{l=1}^{L}$
  if $m\equiv 0(\mathrm{mod}\ T_{\mathrm{inv}})$ **then**
    if $t<=k$ then
      根据式 (4) 计算逆矩阵$\{A_{ll}^{-1}\}_{l=0}^{L-1}$, $\{G_{ll}^{-1}\}_{l=1}^{L}$
    else
      计算向量u,v, 根据式 (7) 和式 (8) 计算逆矩阵$\{A_{ll}^{-1}\}_{l=0}^{L-1}$, $\{G_{ll}^{-1}\}_{l=1}^{L}$
    end if
    t=t+1
  end if
  根据式 (5) 更新参数$\{\theta_l\}_{l=1}^{L}$
  m=m+1
end While

3 实验

为了说明改进的 K-FAC 算法的有效性, 我们在常用的图像分类数据集上进行了实验. 实验中, 数据集选取的是 CIFAR-10 和 CIFAR-100 数据集[19]. 这两个数据集都是由 60 000 张分辨率为32×32的彩色图像组成, 训练集和测试集分别有 50 000 和 10 000 张彩色图像. CIFAR-10 数据集中图像有 10 个不同的类, 每类有 6000 张图像. CIFAR-100 数据集中图像有 100 个不同的类, 每类有 600 张图像. 实验中我们对两个数据集的图像都采用数据增强技术, 包括随机裁剪和水平翻转. 我们选择动量随机梯度下降 (SGDM) 和 K-FAC 算法作为对比标准, 在 ResNet20[20] 上比较了这两个方法和改进的 K-FAC 算法的表现.

实验中我们采用的深度学习框架是 TensorFlow, 训练的硬件环境为单卡 NVIDIA RTX 2080Ti GPU. 实验中批量大小 (batch-size) 设置为 128, 动量为 0.9, 最

大迭代次数为 39 100, 初始学习率 SGDM 设置为 0.03, K-FAC 算法和改进的 K-FAC 算法设置为 0.001, 学习率每 16 000 次迭代衰减为原来的 0.1. 对于 K-FAC 算法和改进的 K-FAC 算法, Fisher 信息矩阵及其逆矩阵的更新频率分别为$T_{\mathrm{FIM}}=10$, $T_{\mathrm{inv}}=100$, 阻尼为 0.001. 在改进的 K-FAC 算法中, 我们令$\alpha=\beta=0.1$. 对于所有的方法, 我们均没有采用权重衰减.

在表 2 中, 我们给出了在 CIFAR-10 数据集上 SGDM, K-AFC 算法和改进的 K-FAC 算法的训练精度及时间比较, 其中, K-FAC 算法给出了每次迭代均更新逆矩阵 (1:1) 和 100 次迭代更新逆矩阵 (100:100) 的实验结果. 表中第一行给出了各种算法的训练精度比较, 其余各行别给出了各种方法每次迭代的平均训练时间以及测试精度首次达到 89%, 90%, 91%, 92%, 93% 的训练时间, 表中最后一列给出了改进的 K-FAC 算法 (100:100) 比 K-FAC 算法 (100:100) 减少的训练时间. 因为 CIFAR-100 数据集和 CIFAR-10 数据集图像数量和分辨率相同, 这两个数据集上每次迭代的训练时间几乎相同, 所以我们仅给出了在 CIFAR-10 数据集上的结果, 在 CIFAR-100 数据集也有类似的结果.

表 2 SGDM, K-FAC 和改进的 K-FAC 算法的训练精度及时间比较

算法	SGDM	K-FAC(1:1)	K-FAC(100:100)	改进的K-FAC(100:100)	改进的K-FAC减少的训练时间(100:100)
训练精度 CIFAR-10 (%)	92.69	93.64	93.59	93.16	–
训练精度CIFAR-100 (%)	73.02	74.45	74.34	74.64	–
每次迭代的平均时间(s)	0.063	2.99	0.092	0.069	0.023
Time (89%)(s)	838	1508	543	411	132
Time (90%)(s)	1035	2409	722	491	231
Time (91%)(s)	1035	3160	1043	679	364
Time (92%)(s)	1035	6306	1472	1137	335
Time (93%)(s)	–	6456	1508	1271	237

从表 2 可以看出, K-FAC 在不同的逆矩阵更新频率下 ((1:1) 和 (100:100)) 的测试精度相差不大, 但每次迭代均更新逆矩阵耗费了大量的计算时间 (每次迭代平均增加了 2.07 s). 结合之前的相关工作, 在本文中我们更多关注若干次迭代更新逆矩阵的实验结果. 因此, 在后文中, 我们主要基于 K-FAC 算法 (100:100) 的实验结果进行讨论.

从测试精度看, 改进的 K-FAC 算法与 K-FAC 算法相差不大. 在 CIFAR-10 数据集上, 改进的 K-FAC 算法的测试精度略低于 K-FAC 算法, 但在 CIFAR-100 数据集上, 改进的 K-FAC 算法的测试精度高于 K-FAC 算法. 从训练时间看, SGDM 从 89% 到 90%, K-FAC 算法从 91% 到 92% 以及改进的 K-FAC 算法从从 91% 到 92% 的训练时间差距较大, 这是因为在学习率衰减之前, 测试精度在较多的迭代中变化不大, 衰减后才达到了相应的测试精度. 改进的 K-FAC 算法每个迭代的平均训练时间与 SGDM 相比, 仅增加了 0.006 s, 比 K-FAC 减少了 0.023 s. 从到达各个测试精度的时间看, 改进的 K-FAC 算法均比 K-FAC 算法减少了大量的训练时间. 比如在测试精度达到 91% 时, K-FAC 算法比 SGDM 多花费了 8 s, 而我们改进的 K-FAC 算法比 SGDM 减少了 356 s. 从表格最后一行看, SGDM 最终的测试精度达不到 93%, K-FAC 算法和改进的 K-FAC 算法都可以达到 93%, 而且改进的 K-FAC 算法减少了 237 s. 从这些结果可以看出, 我们改进的 K-FAC 算法可以达到与 K-FAC 算法相近的训练精度, 同时减少了大量的训练时间, 而且与一阶优化方法相比在速度与精度上都具有一定的优势.

图 2 给出了在 CIFAR-10 数据集上的实验结果, 分别给出了 SGDM, K-FAC 算法 (100:100) 和改进的 K-FAC 算法 (100:100) 的训练损失, 训练精度和测试精度随迭代的变化曲线. 在图中可以看出二阶优化方法 (K-FAC 算法和改进的 K-FAC 算法) 收敛速度明显快于 SGDM, 改进的 K-FAC 算法与 K-FAC 收敛速度相近. 从训练损失看, 所有的方法都可以达到较低的训练损失, SGDM 的训练损失略高; 从精度看, 所有的方法都可以达到很高的测试精度, 我们改进的 K-FAC 算法在前期表现好于 K-FAC.

图 3 分别给出了 SGDM, K-FAC 算法和改进的 K-FAC 算法在 CIFAR-100 数据集上的训练损失, 训练精度和测试精度随迭代的变化曲线. 从图中可以看出, CIFAR-100 数据集和 CIFAR-10 数据集有着相似的实验结果. 但从测试精度看, 改进的 K-FAC 算法好于 K-FAC. 从这些结果我们可以看出, 我们改进的 K-FAC

算法与 K-FAC 算法相比, 有着相似甚至更好的实验效果, 说明我们提出的 Fisher 信息矩阵的逆矩阵进一步近似的方法是有效的.

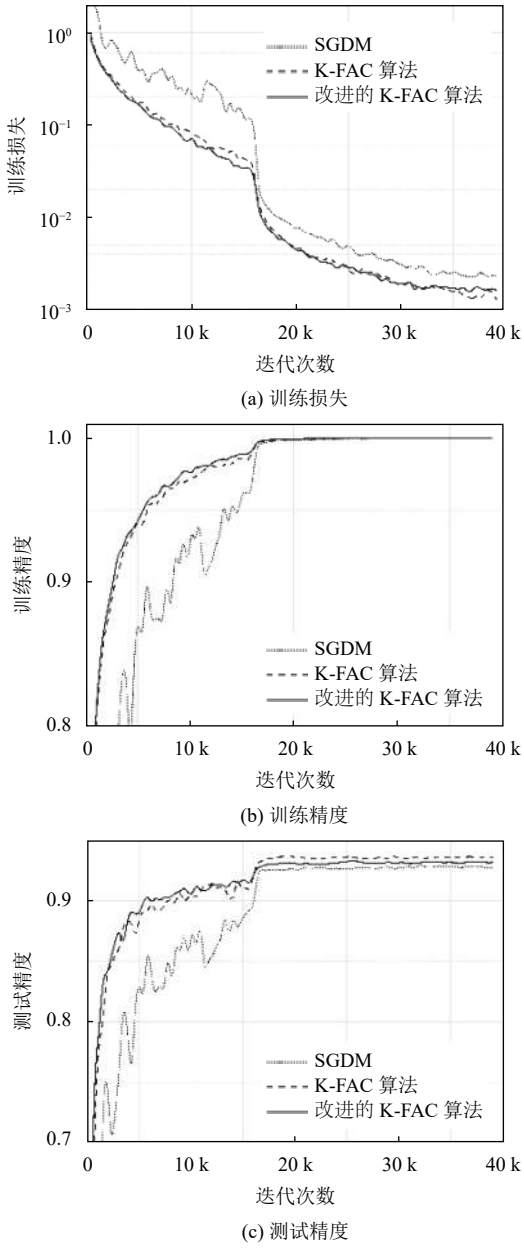

图 2 CIFAR-10 数据集上的实验结果

4 结论

在深度学习中应用二阶优化问题面临的一个重要挑战是计算曲率矩阵的逆矩阵, 由于深度神经网络拥有海量的参数导致其曲率矩阵的规模巨大而难以求逆. 在本文中, 我们基于 K-FAC 算法对 Fisher 信息矩阵的近似方法, 结合拟牛顿方法的思想, 在前期少量迭代中利用原方法训练, 后续迭代利用新计算的矩阵信息构造秩–1 矩阵进行近似. 利用 Sherman-Morrison 公式大大降低了计算复杂度. 实验结果表明, 我们改进的 K-FAC 算法与 K-FAC 算法有着相似甚至更好的实验效果. 从训练时间看, 我们的方法比原方法减少了大量的计算时间, 与一阶优化方法相比我们改进的方法仍具有一定的优势. 但如何在深度学习中更加有效地利用曲率矩阵的信息, 得到更有效更实用的算法, 仍然是在深度学习中应用二阶优化方法面临的重要挑战.

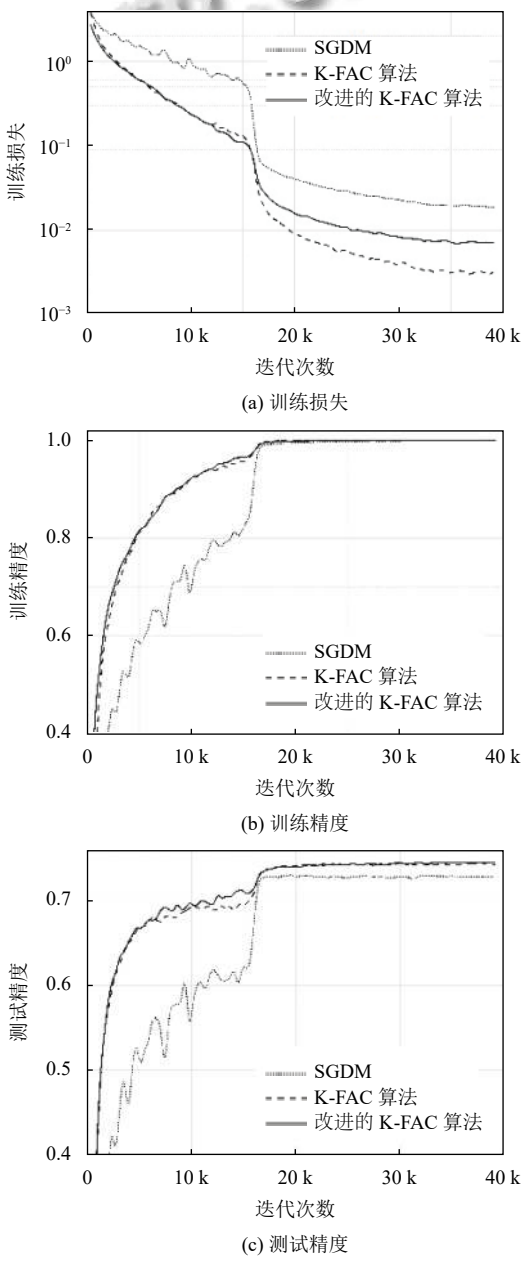

图 3 CIFAR-100 数据集上的实验结果

参考文献

1 Qian N. On the momentum term in gradient descent learning algorithms. Neural Networks, 1999, 12(1): 145–151. [doi: 10.1016/S0893-6080(98)00116-6]

2 Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 2011, 12: 2121–2159.

3 Kingma DP, Ba J. Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference for Learning Representations. San Diego, CA, USA. 2015.

4 Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. Mathematical Programming, 1989, 45(1–3): 503–528.

5 Ollivier Y. Riemannian metrics for neural networks I: Feedforward networks. Information and Inference, 2015, 4(2): 108–153. [doi: 10.1093/imaiai/iav006]

6 Keskar NS, Berahas AS. adaQN: An adaptive quasi-Newton algorithm for training RNNs. Proceedings of 2016 European Conference on Machine Learning and Knowledge Discovery in Databases. Riva del Garda, Italy. 2016. 1–16.

7 Setiono R, Hui LCK. Use of a quasi-Newton method in a feedforward neural network construction algorithm. IEEE Transactions on Neural Networks, 1995, 6(1): 273–277. [doi: 10.1109/72.363426]

8 Xu DP, Dong J, Zhang CD. Convergence of quasi-Newton method for fully complex-valued neural networks. Neural Processing Letters, 2017, 46(3): 961–968. [doi: 10.1007/s11063-017-9621-7]

9 Amari SI. Natural gradient works efficiently in learning. Neural Computation, 1998, 10(2): 251–276. [doi: 10.1162/089976698300017746]

10 Martens J, Grosse R. Optimizing neural networks with kronecker-factored approximate curvature. arXiv: 1503.05671, 2015.

11 Grosse R, Martens J. A kronecker-factored approximate fisher matrix for convolution layers. Proceedings of the 33rd International Conference on International Conference on Machine Learning. New York City, NY, USA. 2016. 573–582.

12 Laurent C, George T, Bouthillier X, *et al*. An evaluation of fisher approximations beyond kronecker factorization. Proceedings of the 6th International Conference on Learning Representations. Vancouver, BC, Canada. 2018. 1–4.

13 George T, Laurent C, Bouthillier X, *et al*. Fast approximate natural gradient descent in a kronecker factored eigenbasis. Proceedings of the 32nd International Conference on Neural Information Processing Systems. Montreal, QC, Canada. 2018. 9573–9583.

14 Martens J, Ba J, Johnson M. Kronecker-factored curvature approximations for recurrent neural networks. Proceedings of the 6th International Conference on Learning Representations. Vancouver, BC, Canada. 2018. 1–25.

15 Zhang GD, Sun SY, Duvenaud D, *et al*. Noisy natural gradient as variational inference. Proceedings of the 35th International Conference on Machine Learning. Stockholm, Sweden. 2018. 5847–5856.

16 Bae J, Zhang GD, Grosse RB. Eigenvalue corrected noisy natural gradient. arXiv: 1811.12565, 2018.

17 Ba J, Grosse RB, Martens J. Distributed second-order optimization using kronecker-factored approximations. Proceedings of the 5th International Conference on Learning Representations. Toulon, France. 2017. 1–17.

18 Osawa K, Tsuji Y, Ueno Y, *et al*. Large-scale distributed second-order optimization using kronecker-factored approximate curvature for deep convolutional neural networks. Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Long Beach, CA, USA. 2019. 12351–12359.

19 Krizhevsky A. Learning multiple layers of features from tiny images. Toronto: University of Toronto, 2009.

20 He KM, Zhang XY, Ren SQ, *et al*. Deep residual learning for image recognition. Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, NV, USA. 2016. 770–778.