用于深度学习的一种改进L-BFGS算法*

赵　睿　颖

(天津大学数学学院，天津　300350)

摘要:本文提出了一种改进的L-BFGS算法,该算法利用相邻迭代点梯度之差,以及由此定义的矩阵与向量之间的乘积的线性组合来更新向量对,进而构造新的Hessian矩阵近似.改进算法将迭代方向进行单位化,以保证所提算法的稳定性.在MNIST和CIFAR10数据集上进行了实验,结果表明改进的算法具有更好的稳定性,与带动量的随机梯度下降法、AdaGrad以及L-BFGS等算法相比,具有更好的实验表现.

关键词:深度学习;二阶优化方法;拟牛顿法;L-BFGS算法

中图分类号:O29　　**DOI**:10.19789/j.1004-9398.2021.05.002

0　引　言

深度学习,作为解决大规模机器学习问题的领先技术,在图像分类、自然语言处理和大规模回归任务等领域都有非常突出的表现.深度学习算法通常需要解决一个高阶非线性的、非凸的、无约束优化的问题,求解其优化问题常用的方法包括一阶和二阶优化方法.其中,一阶优化方法包括随机梯度下降法(stochastic gradient descent,SGD)[1-4],二阶优化方法包括牛顿法、拟牛顿法[5-8]、Hessian-free方法[9-10]和K-FAC方法[11]等.一阶优化方法,如SGD,在计算中成本较低,但存在收敛速度较慢、需要反复地实验不断调整超参数等问题;二阶优化方法,如牛顿法,使用目标函数的Hessian矩阵来确定搜索方向,通过加速收敛,可以减少大量的迭代次数,且不需要复杂的超参数调整,更容易得到好的训练结果.但是,由于深度学习中模型拥有海量的参数,Hessian矩阵的维度过大,导致无法直接计算其逆矩阵.因此,在深度学习中,牛顿法并不可行,本文利用拟牛顿法进行深度学习算法研究.

拟牛顿法是求解非线性优化问题最有效的算法之一.牛顿法的优点是收敛速度快,但需要计算二阶导数,而且目标函数的Hessian矩阵可能不是正定的.为了克服牛顿法的缺点,提出了拟牛顿法,其基本思想是用不包含二阶导数的矩阵近似牛顿法中的Hessian矩阵的逆矩阵.因此,拟牛顿法只需要一阶导数信息,这使得拟牛顿法成为一个比较受欢迎的算法.同时,拟牛顿法不需要二阶导数的信息,有时比牛顿法优化更为有效.由于构造近似矩阵的方法不同,出现了不同的拟牛顿法.常见的拟牛顿法有BFGS算法、L-BFGS算法以及Hessian-free算法等.

L-BFGS算法是最受欢迎的拟牛顿算法之一.Liu和Nocedal[12]是第一个提出L-BFGS算法概念的;Wang等[13]提出了用于非凸随机优化的SdLBFGS算法,该算法利用了经典L-BFGS算法的框架;Keskar和Berahas[14]提出了用于训练循环神经网络的adaQN算法,该算法在L-BFGS算法的基础上每L(更新周期为20~60次迭代)次更新1次Hessian矩阵;Rafati和Marcia[15]将L-BFGS算法与信赖域算法以及线搜索相结合,提出了TRMinATR算法;Byrd等[16]提出了一种随机拟牛顿方法(stochastic quasi-Newton,SQN),该方法通过对样本进行二次采样,利用矩阵与向量的乘积来形成向量对(而不是使用随机梯度之间的差),取得了比以往方法更好的结果,并证明了在强凸的情况下,其收敛速度是次线性的;Berahas等[17]将上述方法与随机方差缩减梯度方法[18]相结合,证明了算法的线性收敛性;Le等[19]在深度学习的背景下,对现有的优化算法如L-BFGS、共轭梯度法(conjugate gradient,CG)和SGD的优缺

收稿日期:2020-08-17

* 国家自然科学基金项目(11871051)

点进行了实证研究,表明L-BFGS和CG在许多情况下优于SGD;Berahas等[20]介绍了采样的L-BFGS(SL-BFGS),与经典的L-BFGS随着迭代的进行而顺序近似Hessian矩阵(逆矩阵)不同,SL-BFGS是在当前迭代点附近进行随机采样来近似Hessian矩阵(逆矩阵);Rafati等[21]介绍了一种基于L-BFGS使用信赖域的算法(TRMinATR),作为解决深度学习中优化问题的一种新的方法;Gower等[22]利用文献[16]中算法的框架,在优化过程中加入块BFGS更新,收集曲率信息,并证明了块BFGS更新可以加快训练速度.

上述文献大多是在理论上讨论算法的可行性,且通常利用相邻迭代点梯度之差,或矩阵与向量的乘积中的一种来更新向量对,缺乏在常用数据集CIFAR10和MNIST上的实验结果.本文提出了一种改进L-BFGS算法,拟利用梯度之间的差和矩阵与向量之间乘积的线性组合来更新向量对,以实现更新Hessian矩阵的逆矩阵.

1　常用拟牛顿法BFGS和L-BFGS算法

现有拟牛顿法大多为传统拟牛顿法的变形,主要包括BFGS和L-BFGS算法,其区别主要在于如何定义和构造拟牛顿矩阵、计算搜索方向以及更新模型中参数.

1.1　BFGS算法

BFGS算法是构造Hessian矩阵的近似正定矩阵常用的拟牛顿法.在深度学习中,通常需要解决的是高阶非线性的、非凸的无约束优化问题,其公式为

$$\min_{w\in R^d}\left\{F(\boldsymbol{w})\coloneqq\frac{1}{N}\sum_{i\in[N]}f(\boldsymbol{w};x_i,z_i)\triangleq\frac{1}{N}\sum_{i\in[N]}f_i(\boldsymbol{w})\right\},\tag{1}$$

式中$f_i:\mathbf{R}^d\to\mathbf{R}$表示第$i$个样本的损失函数,$\boldsymbol{w}\in\mathbf{R}^d$是模型可训练的参数向量,$d$是训练参数的个数,$d,i\in[N]\triangleq\{1,\cdots,N\}$;$\{(x_i,z_i)\}_{i=1}^{i=N}$是训练样本集.用于解决式(1)的拟牛顿算法,更新公式通常为:

$$\boldsymbol{w}_{k+1}=\boldsymbol{w}_k-\alpha_k\boldsymbol{B}_k^{-1}\boldsymbol{g}_k,\tag{2}$$

$$\boldsymbol{w}_{k+1}=\boldsymbol{w}_k-\alpha_k\boldsymbol{H}_k\boldsymbol{g}_k,\tag{3}$$

式中,$\boldsymbol{B}_k$为Hessian矩阵$\nabla^2 f(\boldsymbol{w}_k)$的近似;$\boldsymbol{H}_k$为Hessian矩阵的逆$\left[\nabla^2 f(\boldsymbol{w}_k)\right]^{-1}$的近似;$\boldsymbol{g}_k:=\nabla f(\boldsymbol{w}_k)$;$\alpha_k$为步长.一般情况下,$N$和$d$都是非常大的数字.因此,直接计算真正的Hessian矩阵是不切实际的.拟牛顿法利用目标函数值和一阶导数的信息,构造合适的$\boldsymbol{H}_k$来逼近真正的Hessian矩阵的逆矩阵,使得不需要计算真正的Hessian矩阵,既减少了计算复杂度,又保持了牛顿法收敛速度快的优点.

拟牛顿法中的一个重要问题是构造合适的$\boldsymbol{H}_k$,需要满足下列条件:

1)$\boldsymbol{H}_k$满足拟牛顿方程:$\boldsymbol{B}_{k+1}\boldsymbol{s}_k=\boldsymbol{y}_k$或$\boldsymbol{H}_{k+1}\boldsymbol{y}_k=\boldsymbol{s}_k$;

2)$\boldsymbol{H}_k$是正定对称矩阵;

3)由$\boldsymbol{H}_k$到$\boldsymbol{H}_{k+1}$的更新是低秩更新.

BFGS方法是拟牛顿方法的一种,在BFGS方法中,$\boldsymbol{B}_k$更新公式为

$$\boldsymbol{B}_{k+1}=\boldsymbol{B}_k+\frac{\boldsymbol{y}_k\boldsymbol{y}_k^{\mathrm{T}}}{\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{y}_k}-\frac{\boldsymbol{B}_k\boldsymbol{s}_k\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{B}_k}{\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{B}_k\boldsymbol{s}_k},\tag{4}$$

式中$\boldsymbol{s}_k=\boldsymbol{w}_{k+1}-\boldsymbol{w}_k$,$\boldsymbol{y}_k=\boldsymbol{g}_{k+1}-\boldsymbol{g}_k$.通过利用Sherman-Morrison-Woodbury公式,可以得到更新公式

$$\boldsymbol{H}_{k+1}=\left(\boldsymbol{I}-\frac{\boldsymbol{s}_k\boldsymbol{y}_k^{\mathrm{T}}}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k}\right)\boldsymbol{H}_k\left(\boldsymbol{I}-\frac{\boldsymbol{y}_k\boldsymbol{s}_k^{\mathrm{T}}}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k}\right)+\frac{\boldsymbol{s}_k\boldsymbol{s}_k^{\mathrm{T}}}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k}.\tag{5}$$

算法1:BFGS算法.

输入:运行误差$\varepsilon>0$;最大迭代次数k_{max}.

初始化:$\boldsymbol{w}_0\in\mathbf{R}^d$;$\boldsymbol{H}_0=\boldsymbol{I}$.

1:$k=1$;

2:计算梯度$\boldsymbol{g}_k=\nabla f(\boldsymbol{w}_k)$;

3:while$\|\boldsymbol{g}_k\|>\varepsilon$;$k<k_{\max}$　do;

4:计算更新方向$\boldsymbol{p}_k=-\boldsymbol{H}_k\boldsymbol{g}_k$;

5:利用线搜索计算步长α_k;

6:$\boldsymbol{r}_k=\alpha_k\boldsymbol{p}_k$;

7:$\boldsymbol{w}_{k+1}=\boldsymbol{w}_k+\boldsymbol{r}_k$;

8:计算梯度$\boldsymbol{g}_{k+1}$;

9:用BFGS更新公式(5)更新拟牛顿矩阵H_{k+1};

10:$k=k+1$;

11:end while.

1.2　L-BFGS算法

BFGS方法需要存储和计算$\boldsymbol{B}_k$,而在深度学习中,N和d的值很大,甚至可能达到上亿级别,因此,存储和计算$\boldsymbol{B}_k$的代价很高.而L-BFGS方法有效地解决了这一问题.在L-BFGS方法中,不再需要直接存储和计算$\boldsymbol{B}_k$,而是使用保存在存储器中的m个最新向量对$\{\boldsymbol{s}_k,\boldsymbol{y}_k\}$对$\boldsymbol{B}_k$进行更新,只需要存储2个$m\times d$的矩阵,节省了大量的存储空间和计算费用.在L-BFGS方法中,可以通过two-loop recursion算法[23],直接计算Hessian矩阵与梯度向量的乘积$(-\boldsymbol{H}_k\boldsymbol{g}_k)$.

算法2:L-BFGS算法.

输入:初始点$\boldsymbol{w}_0$,运行误差$\varepsilon>0$,最大迭代次数$k_{\max}$,存储最近的m次迭代数据.

初始化:$\boldsymbol{w}_0\in\mathbf{R}^d$; $\boldsymbol{H}_0=\boldsymbol{I}$.

1:$k=1$;

2:计算梯度$\boldsymbol{g}_k=\nabla f\left(\boldsymbol{w}_k\right)$;

3:若$\left\|\boldsymbol{g}_{k+1}\right\|\leqslant\varepsilon$,算法终止,返回最优解$\boldsymbol{w}_{k+1}$,否则转入第4步;

4:计算本次迭代的更新方向$\boldsymbol{p}_k=-\boldsymbol{H}_k\boldsymbol{g}_k$;

5:由精确一维线搜索$f\left(\boldsymbol{w}_k+\alpha_k\boldsymbol{p}_k\right)=\min_{\alpha\geqslant0}f\left(\boldsymbol{w}_k+\alpha\boldsymbol{p}_k\right)$,计算步长$\alpha_k>0$;

6:更新$\boldsymbol{w}$:$\boldsymbol{w}_{k+1}=\boldsymbol{w}_k+\alpha_k\boldsymbol{p}_k$;

7:如果$k\geqslant m$,保留最新的向量对$(\boldsymbol{s}_k,\boldsymbol{y}_k)$,删除$(\boldsymbol{s}_{k-m},\boldsymbol{y}_{k-m})$;

8:计算$\boldsymbol{s}_k=\boldsymbol{w}_{k+1}-\boldsymbol{w}_k$; $\boldsymbol{y}_k=\boldsymbol{g}_{k+1}-\boldsymbol{g}_k$;

9:用two-loop recursion算法$\boldsymbol{p}_k=-\boldsymbol{H}_k\boldsymbol{g}_k$;

10:$k=k+1$,转第3步.

算法3:two-loop recursion算法(方向更新).

输入:$\boldsymbol{g}_k$;存储向量对数m;向量对$(\boldsymbol{s}_k,\boldsymbol{y}_k)$;

1:令$\boldsymbol{q}=-\boldsymbol{g}_k$,$\boldsymbol{\rho}_i=\dfrac{1}{\boldsymbol{y}_i^{\mathrm{T}}\boldsymbol{s}_i}$;

2:for $i=k-1,k-2,\cdots,k-m$ do;

3:$\alpha_i=\boldsymbol{\rho}_i\boldsymbol{s}_i^{\mathrm{T}}\boldsymbol{q}$;

4:$\boldsymbol{q}=\boldsymbol{q}-\alpha_i\boldsymbol{y}_i$;

5:end for;

6:$\boldsymbol{r}=\boldsymbol{H}_k^{(0)}\boldsymbol{q}$;

7:for $i=k-m,\cdots,k-1$ do;

8:$\boldsymbol{\beta}=\boldsymbol{\rho}_i\boldsymbol{y}_i^{\mathrm{T}}\boldsymbol{r}$;

9:$\boldsymbol{r}=\boldsymbol{r}+\boldsymbol{s}_i(\alpha_i-\boldsymbol{\beta})$;

10:end for;

输出:$\boldsymbol{r}=-\boldsymbol{H}_k\boldsymbol{g}_k$.

2 改进L-BFGS算法

在BFGS算法中仍然有一些缺陷,如当优化问题规模很大时,矩阵的存储和计算将变得不可行.为了解决这个问题,就有了L-BFGS算法,但其依然有一些不足,如利用全批量数据来计算梯度.为了进一步节省计算训练时间、花费以及提高算法的精度,本节提出了一种改进L-BFGS算法.利用L-BFGS算法的框架,改变向量对的更新规则,利用相邻迭代点梯度之差,矩阵与向量之间的乘积的线性组合来更新向量对.

改进L-BFGS算法中,利用小批量数据而不是全批量数据来计算梯度,即

$$\boldsymbol{g}_k:=\frac{1}{a}\sum\nolimits_{i\in[S]}f(\boldsymbol{w}_k;\ x_i,z_i),$$

式中,$S\subset\{1,2,3,\cdots,N\}$是从训练样本中随机选择的子集,定义$a\triangleq|S|$.改进方法的具体内容与步骤如下:

(1)用$\hat{\boldsymbol{y}}_k$代替$\boldsymbol{y}_k$($\boldsymbol{y}_k=\boldsymbol{g}_{k+1}-\boldsymbol{g}_k$)更新BFGS公式.式中$\hat{\boldsymbol{y}}_k=\theta_k\boldsymbol{y}_k+(1-\theta_k)M_k\boldsymbol{s}_k$,$\boldsymbol{s}_k=\boldsymbol{w}_{k+1}-\boldsymbol{w}_k$;$M_k$为如下定义的对角矩阵

$$\begin{pmatrix}\sqrt{(\boldsymbol{y}_{k-1})_1^2} & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \sqrt{(\boldsymbol{y}_{k-1})_n^2}\end{pmatrix},$$

其中,$(\boldsymbol{y}_{k-1})_i$表示向量$\boldsymbol{y}_{k-1}$的第i个分量.

$\boldsymbol{\theta}_k$的取值如下:

$$\boldsymbol{\theta}_k=\begin{cases}\dfrac{0.75\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{s}_k}{\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{s}_k-\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{y}_k} & \boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{y}_k<0.25\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{s}_k,\\ 1 & \text{其他},\end{cases}$$

此时,更新公式为

$$\boldsymbol{H}_{k+1}=\left(\boldsymbol{I}-\frac{\boldsymbol{s}_k\hat{\boldsymbol{y}}_k^{\mathrm{T}}}{\hat{\boldsymbol{y}}_k^{\mathrm{T}}\boldsymbol{s}_k}\right)\boldsymbol{H}_k\left(\boldsymbol{I}-\frac{\hat{\boldsymbol{y}}_k\boldsymbol{s}_k^{\mathrm{T}}}{\hat{\boldsymbol{y}}_k^{\mathrm{T}}\boldsymbol{s}_k}\right)+\frac{\boldsymbol{s}_k\boldsymbol{s}_k^{\mathrm{T}}}{\hat{\boldsymbol{y}}_k^{\mathrm{T}}\boldsymbol{s}_k}.$$

(2)当直接沿着迭代方向$\boldsymbol{p}_k$进行更新时,算法在几个epoch后停止运行,方法不稳定,精度波动较大,因此,对迭代方向单位化$\boldsymbol{p}_k=\dfrac{\boldsymbol{p}_k}{\left\|\boldsymbol{p}_k\right\|}$使改进算法更加稳定.

算法4:改进算法.

输入:初始点$\boldsymbol{w}_0$、正整数m、存储的最近m次数据、运行误差$\varepsilon>0$;

1:$k=1$;

2:for $k\geqslant1$ do;

3:计算梯度$\boldsymbol{g}_k$,若$\left\|\boldsymbol{g}_k\right\|<\varepsilon$,算法终止,返回最优解$\boldsymbol{w}_k$,否则转入第4步;

4:用two-loop recursion算法计算更新方向$\boldsymbol{p}_k=-\boldsymbol{H}_k\boldsymbol{g}_k$;

5:选择学习率$\alpha_k>0$;更新迭代$\boldsymbol{w}_{k+1}=\boldsymbol{w}_k-\alpha_k\boldsymbol{p}_k$;

6:更新向量对$\boldsymbol{s}_k=\boldsymbol{w}_{k+1}-\boldsymbol{w}_k$; $\hat{\boldsymbol{y}}_k=\boldsymbol{\theta}_k\boldsymbol{y}_k+(1-\boldsymbol{\theta}_k)M_k\boldsymbol{s}_k$;

7:end for;

8:If $k\geqslant m$;

9:用$(\boldsymbol{s}_k,\hat{\boldsymbol{y}}_k)$代替原来的向量对$(\boldsymbol{s}_{k-m},\hat{\boldsymbol{y}}_{k-m})$;

10:else;

11:存储向量对$(\boldsymbol{s}_k,\hat{\boldsymbol{y}}_k)$;

12:方向单位化 $\boldsymbol{p}_k = \boldsymbol{p}_k / \|\boldsymbol{p}_k\|$;

13:$k = k + 1$, 转第3步.

与L-BFGS算法相比,改进L-BFGS算法有2个关键的区别特征:(1)向量对的创建方式;(2)迭代方向的单位化.首先,该方法构造的逆Hessian矩阵的近似矩阵能更好地利用目标函数的当前曲率信息;其次,更新向量对时,不仅利用了梯度之间的差,还利用了连续迭代的迭代点之间的差;再次,利用迭代方向单位化,使改进算法具有更好的稳定性;最后,改进L-BFGS算法不需要直接存储和计算B_k,只需要存储2个$m \times d$的矩阵,节省了大量的存储空间和计算费用.因此,与BFGS算法相比,改进L-BFGS算法大大减少了计算复杂度以及计算费用.

3　实验结果

为了将改进L-BFGS算法与现有的流行算法(SGD、AdaGrad和L-BFGS算法)进行比较,本文采用Li和Liu[24]使用的卷积神经网络进行训练,给出了在MNIST[25]和CIFAR10数据集[26]上的实验结果.MNIST数据集包含70 000个手写数字样本,其中60 000个样本用作训练集,10 000个样本用作测试集,数字范围为0~9,图像包括描述其预期分类的标签;CIFAR 10数据集共有60 000张图像,其中50 000张图像用于训练,10 000张图像用于测试,所有图像分为10类,每类包含图像6 000张.数据集一共分为5个训练批次和1个测试批次,每个批次图像10 000张,测试批次包含从每类中随机选择的1 000张图像,剩下的图像随机排列组成了训练批次,因此,有的训练批次某一类的图像可能多于另一类的图像,但总的来看,训练集每一类都有5 000张图像.

在实验中,设置改进L-BFGS算法的学习率为$\frac{1}{\sqrt{k}}$,批量大小为64,SGDM的动量为0.5,其中SGDM为带动量的随机梯度下降法(stochastic gradient descent with momentum,SGDM).SGDM、AdaGrad和L-BFGS算法的学习率选择范围为{0.001,0.003,…,1,3}.所有的算法均是训练了100个迭代周期(epoch).

SGDM、AdaGrad、L-BFGS和改进L-BFGS算法在CIFAR10和MNIST数据集上的测试精度和训练损失如图1所示.在CIFAR10数据集上,当训练100个迭代周期时,SGDM、AdaGrad、L-BFGS和改进L-BFGS

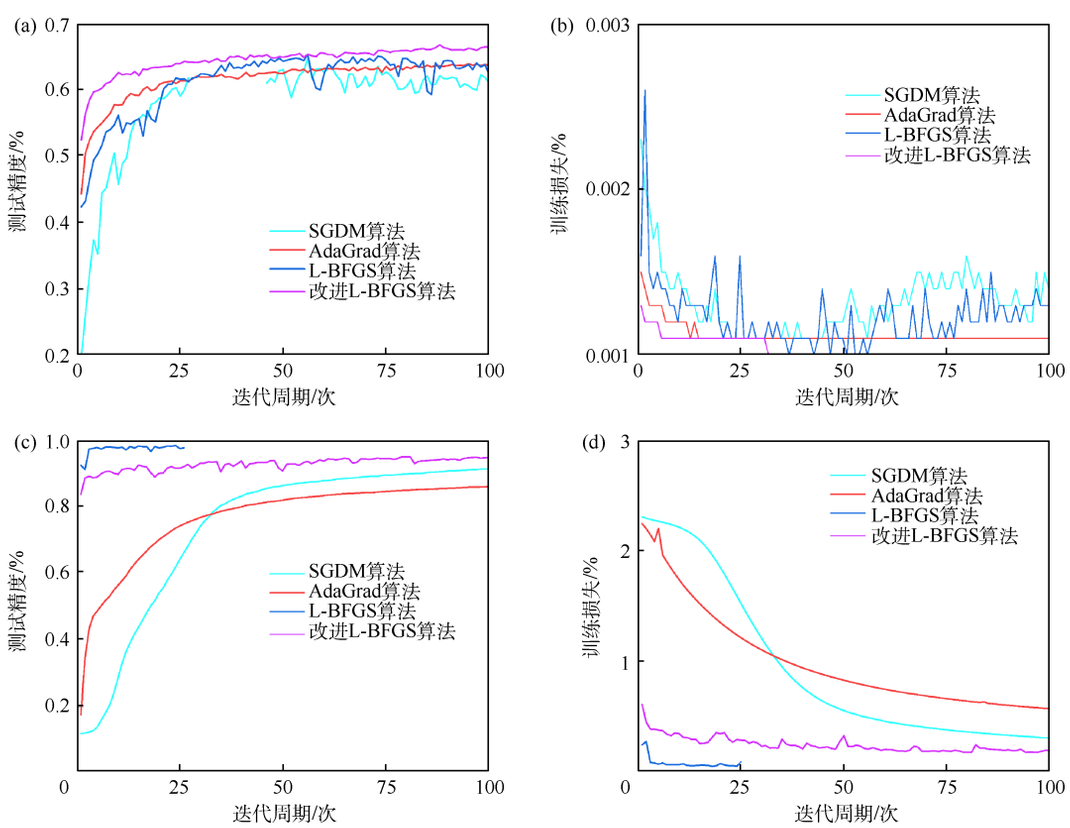


图1　4种算法在2种数据集上的测试精度和训练损失

(a)CIFAR10数据集的测试精度;(b)CIFAR10数据集的训练损失;(c)MNIST数据集的测试精度;(d)MNIST数据集的训练损失

算法的测试精度分别是62%、64%、63%和66%,训练损失分别是0.001 4、0.001 2、0.001 3和0.001 1;与其他3种算法相比,改进L-BFGS算法测试精度最高,最终能达到66%的测试精度,且损失最低.在MNIST数据集上,当训练100个迭代周期时,SGDM、AdaGrad、L-BFGS和改进L-BFGS算法的测试精度分别是92%、86%、None和95%,其中None指无数据,训练损失分别是0.31、0.58、None和0.20;与其他3种算法相比,改进L-BFGS算法的测试效果最优,能够以最快的速度达到最高的测试精度,且其损失始终处于一个比较低的水平.在运行10余个迭代周期之后,SGDM算法能达到比改进算法更低的训练损失.在前25个迭代周期,与改进L-BFGS算法相比,L-BFGS算法的测试精度较优,且训练损失较低,实验效果较佳;但在运行25个迭代周期之后,该算法停止,L-BFGS算法在Li和Liu[24]的研究中也有类似的情况.

超参数存储向量对数m和批量b对改进L-BFGS算法精度的影响如图2所示.当m=50和100时,算法的表现效果相当;当m=150时,其表现略优于前2种情况,且能达到最高精度66%.当b=64和256时,算法的表现相当;当b=128时,其表现明显差于其他2种情况.

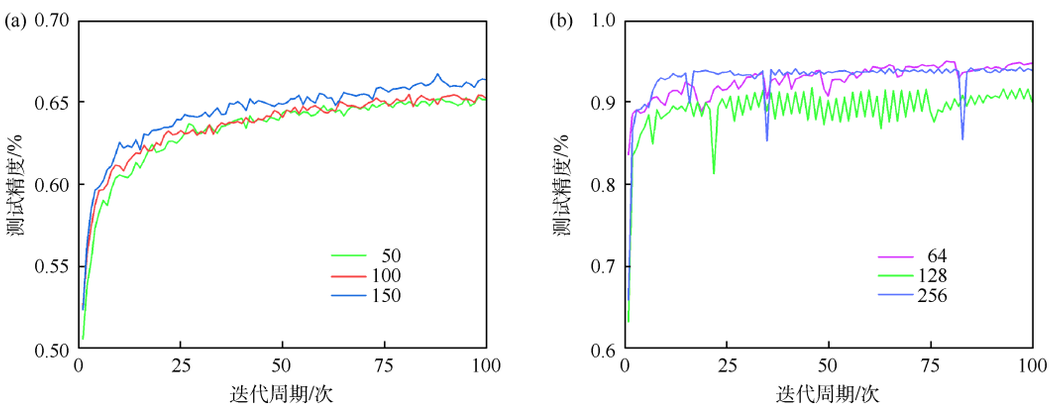


图2　不同大小超参数对改进L-BFGS算法的影响

(a)存储向量对数;(b)批量

4　讨　论

由图1可知,改进L-BFGS算法在CIFAR10数据集上的测试精度始终高于其他3种算法,说明了算法的高效性.改进L-BFGS算法在CIFAR10数据集上的训练损失始终处于一个较低的水平,与SGDM、L-BFGS算法相比,波动较小,更加稳定.虽然在前25个迭代周期内,L-BFGS算法在MNIST数据集上的测试精度高于改进L-BFGS算法,但在运行25个迭代周期后,算法停止.改进L-BFGS算法在MNIST数据集上的训练损失始终低于SGDM和L-BFGS算法,也再一次说明了算法的稳定性.综上,改进L-BFGS算法比原L-BFGS算法更加高效及稳定.图2展示了不同的超参数时,所得结果相差了4%的精度,为了使算法更加高效,在实验的过程中要选择合适的超参数.在本文实验过程中,当选择p_k而不进行单位化来迭代时,算法在运行几个迭代后停止运行,遂采取了对迭代方向单位化.

5　结束语

本文提出了一种基于L-BFGS算法的改进优化算法,作为训练深层神经网络的一个新颖的拟牛顿算法.与已有的拟牛顿算法不同,改进L-BFGS算法利用连续迭代的梯度之差以及矩阵与向量乘积的线性组合更新向量对,并单位化迭代方向,使得该算法具有更好的实验表现.本研究主要是利用数值实验来说明算法的有效性.因此,未来可以利用理论来证明算法的收敛性,讨论算法的理论收敛性分析以及迭代方向单位化的理论必要性.

参 考 文 献

[1] ROBBINS H, MONRO S. A stochastic approximation method[J]. The Annals of Mathematical Statistics, 1951, 22(3): 400-407.

[2] DUCHI J,HAZAN E,SINGER Y. Adaptive subgradient methods for online learning and stochastic optimization[J]. Journal of Machine Learning Research,2011(12):2121-2159.

[3] NIU F,RECHT B,RE C,et al. Hogwild:a lock-free approach to parallelizing stochastic gradient descent[J]. Advances in Neural Information Processing Systems,2011(24):693-701.

[4] KINGMA D P,BA J L. Adam:a method for stochastic optimization[EB/OL]. (2014-01-22)[2017-01-30]. https://arxiv.org/abs/1412.6980.

[5] MORITZ P,NISHIHARA R,JORDAN M. A linearly-convergent stochastic L-BFGS algorithm[C]//GRETTON A. Artificial Intelligence and Statistics. California:Microtome Publishing,2016:249-258.

[6] ERWAY J B,GRIFFIN J D,MARCIA R F. Trust-region algorithms for training responses:machine learning methods using indefinite hessian approximations[J]. Optimization Methods and Software,2020,35(3):460-487.

[7] LEE C P,LIM C H,WRIGHT S J. A distributed quasi-newton algorithm for empirical risk minimization with nonsmooth regularization[C]//ACM. KDD'18:Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. New York:ASSOC Computing Machinery,2018:1646-1655.

[8] RAFATI J,MARCIA R F. Improving L-BFGS initialization for trust-region methods in deep learning[C]//WANI M.2018 17th IEEE international conference on machine learning and applications (ICMLA). New York:IEEE,2018:501-508.

[9] MARTENS J,SUTSKEVER I. Training deep and recurrent networks with hessian-free optimization[J]. Neural Networks (Tricks of the Trade),2012,7700:479-535.

[10] CHO M,DHIR C,LEE J. Hessian-free optimization for learning deep multidimensional recurrent neural networks[C]// CORTES C. Advances in neural information processing systems 28(NIPS 2015). Montreal:Neural Information Processing Systems (NIPS),2015:883-891.

[11] OSAWA K,TSUJI Y,UENO Y,et al. Large-scale distributed second-order optimization using Kronecker-factored approximate curvature for deep convolutional neural networks[C]//IEEE Comp Soc.2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR 2019). New York:IEEE,2019:12351-12359.

[12] LIU D C,NOCEDAL J. On the limited memory BFGS method for large scale optimization[J]. Mathematical Programming,1989,45(1/2/3):503-528.

[13] WANG X,MA S,GOLDFARB D,et al. Stochastic quasi-newton methods for nonconvex stochastic optimization[J]. SIAM Journal on Optimization,2017,27(2):927-956.

[14] KESKAR N S,BERAHAS A S.adaQN:an adaptive quasi-newton algorithm for training RNNs[J]. Machine Learning and Knowledge Discovery in Databases,2016(9851):1-16.

[15] RAFATI J,MARCIA R F. Quasi-newton optimization methods for deep learning applications [EB/OL].(2019-09-04)[2020-4-19].https://arxiv.org/abs/1909.01994.

[16] BYRD R H,HANSEN S L,NOCEDAL J,et al. A stochastic quasi-newton method for large-scale optimization[J]. SIAM Journal on Optimization,2016,26(2):1008-1031.

[17] BERAHAS A S,NOCEDAL J,TAKAC M. A multi-batch L-BFGS method for machine learning[C]//LEE D. Advances in neural information processing systems. Barcelona:Neural Information Processing Systems (NIPS),2016:1055-1063.

[18] JOHNSON R,ZHANG T. Accelerating stochastic gradient descent using predictive variance reduction[C]//LEE D. Advances in neural information processing systems. California:Neural Information Processing Systems (NIPS),2013:315-323.

[19] LE Q V,NGIAM J,COATES A,et al. On optimization methods for deep learning[C]//LISE G,TOBIAS S. Proceedings of the 28th international conference on machine learning. Washington: ICML,2011:265-272.

[20] BERAHAS A S,JAHANI M,TAKAC M. Quasi-newton methods for deep learning:forget the past,just sample[EB/OL].(2019-5-29)[2020-3-11].https://arxiv.org/abs/1901.09997.

[21] RAFATI J,DEGUCHY O,MARCIA R F. Trust-region minimization algorithm for training responses (TRMinATR):the rise of machine learning techniques[C]//MAIORANA E. 2018 26th European signal processing conference. Rome: IEEE computer society,2018.

[22] GOWER R,GOLDFARB D,RICHTARIK P. Stochastic block BFGS:squeezing more curvature out of data[EB/OL].(2016-03-31)[2020-03-20]. https://arxiv.org/abs//603.09649.

[23] NOCEDAL J,WRIGHT S J. Numerical optimization[M]. 2nd ed. Berlin:Springer,2006:135-193.

[24] LI Y K, LIU H D. Implementation of stochastic quasi-newton's method in pytorch[EB/OL]. (2018-05-07)[2020-3-20]. https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html.

[25] YANN L C. The MNIST database of handwritten digits[EB/OL].(2010-10-01)[2020-1-23]. http://yann.lecun.com/exdb/mnist/.

[26] KRIZHEVSKY A, HINTON G. Learning multiple layers of features from tiny images[J]. Handbook of Systemic Autoimmune Diseases, 2009, 1(4): 1-60.

An improved L-BFGS algorithm for deep learning

ZHAO Ruiying

(School of Mathematical Science, Tianjin University, Tianjin 300350)

Abstract: In order to propose an improved L-BFGS algorithm, the algorithm uses the linear combination of the difference between gradients and the product between the matrix and the vector defined thereby to update the curvature pairs, and then construct a new approximation of the Hessian matrix. The improved algorithm has better performance through direction regularization. The main innovation of this paper is to use linear combination to update the curvature pairs and the unitization of the iteration direction. The experiments implemented based on MNIST and CIFAR10 database. The numerical results show that the proposed improved L-BFGS algorithm is efficient, and has better performance compared to the stochastic gradient descent with momentum, AdaGrad and L-BFGS method.

Keywords: deep learning; second-order optimization methods; Quasi-Newton method; L-BFGS method

(责任编辑:马田田)