Advances in Applied Mathematics 应用数学进展, **2022, 11(12), 9026-9038**
Published Online December 2022 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2022.1112952

Hans汉斯

一种带矩限制的批量自适应方差缩减算法

朱桂勇[1]，雷家林[2*]

[1]浙江师范大学行知学院，浙江 金华
[2]浙江师范大学，浙江 金华

收稿日期：2022年11月26日；录用日期：2022年12月21日；发布日期：2022年12月29日

摘　要

诸多机器学习模型的训练过程可以归结为求解一个优化问题，传统的梯度下降算法及其变种的收敛速度并不能让人满意。本文在随机方差缩减梯度算法的基础上，结合了自适应学习率和矩限制的特点，并利用批量思想，选取大批量样本代替全部样本进行全局梯度的计算，选取小批量样本进行参数的迭代更新，提出了带矩限制的批量自适应方差缩减算法，并对算法的收敛性进行了说明。通过基于MNIST的数值实验，验证了该算法的有效性，并探究得出实验参数对算法稳定性的影响。

关键词

机器学习，自适应学习率，矩限制，小批量，随机方差缩减梯度法

An Adaptive and Momental Bound Method for Stochastic Variance Reduced Gradient

Guiyong Zhu[1], Jialin Lei[2*]

[1]Xingzhi College, Zhejiang Normal University, Jinhua Zhejiang
[2]Zhejiang Normal University, Jinhua Zhejiang

Received: Nov. 26^{th}, 2022; accepted: Dec. 21^{st}, 2022; published: Dec. 29^{th}, 2022

* 通讯作者。

文章引用: 朱桂勇, 雷家林. 一种带矩限制的批量自适应方差缩减算法[J]. 应用数学进展, 2022, 11(12): 9026-9038.
DOI: 10.12677/aam.2022.1112952

Abstract

The training process of many machine learning models can be reduced to solving an optimization problem, and the convergence speed of the traditional gradient descent algorithm and its variants is not satisfactory. In this paper, based on the stochastic variance reduction gradient algorithm, we combine the characteristics of adaptive learning rate and moment limitation, and use the batch idea to select large batch samples instead of all samples for the calculation of global gradient, and select small batch samples for the iterative update of parameters, then propose the batch adaptive variance reduction algorithm with moment limitation, and illustrate the convergence of the algorithm. The effectiveness of the algorithm is verified through MNIST-based numerical experiments, and the influence of the experimental parameters on the stability of the algorithm is explored.

Keywords

Machine Learing, Adaptive Learning Rate, Momental Bound, Mini-Batch, SVRG

1. 引言

考虑如下无约束极小化问题 [1]

$$\min P(w), \quad P(w) = \frac{1}{n}\sum_{i=1}^{n}\psi_i(w), \tag{1}$$

其中$\psi_i : R^d \to R, i = 1, ...n$.这种模型在机器学习问题中非常普遍, 比如在有监督学习中, n 表示样本数量, ψ_i表示分类或回归模型中样例 i 的损失函数.

对于以上的目标函数, 通常使用梯度下降方法进行求解 [2–5], 但梯度下降有三种变体, 分别是全梯度下降算法、随机梯度下降算法和小批量随机梯度下降算法, 其不同之处在于使用多少样本来

计算目标函数的梯度, 根据样本数量的不同, 在参数更新的准确性和执行更新所需的时间之间进行权衡.

全梯度下降算法 [6]需要计算所有的样本梯度来进行迭代更新, 能够获得极好的梯度下降方向并提供最终收敛速度的保证. 但 FGD 不能在线更新模型, 即不能利用动态的例子, 其更新规则如下

$$w_t = w_{t-1} - \frac{\alpha_t}{n} \sum_{i=1}^{n} \nabla \psi_i \left(w_{t-1} \right). \tag{2}$$

FGD 在面对大数据集时会出现冗余计算的情况, 因为它在为每个参数更新之前都会对之前相同的样本重新计算, 而随机梯度下降算法 [7](Stochastic Gradient Descent, SGD)每次只需要选取单个样本计算近似梯度执行参数更新来消除这种冗余

$$w_t = w_{t-1} - \alpha_t \nabla \psi_{i_t} \left(w_{t-1} \right), \tag{3}$$

SGD 的计算成本仅为 FGD 的 $1/n$, 同时也可用于在线学习, 但由于样本选取的随机性引入了方差, 导致目标函数在收敛的过程中出现剧烈波动.

随后出现了改进的小批量随机梯度下降算法 [8](Mini-Batch Stochastic Gradient Descent, Mini-Batch SGD)

$$w_t = w_{t-1} - \frac{\alpha_t}{|\mathcal{B}|} \sum_{i \in |\mathcal{B}|} \nabla \psi_i \left(w_{t-1} \right), \tag{4}$$

(4) 式充分结合了全梯度下降算法和随机梯度下降算法的特点, 即在每次参数更新时从样本中随机选取部分样本进行梯度估计, 在一定程度上既保证了梯度下降方向, 又降低了样本随机选取的方差. 相较于全梯度下降而言, 小批量随机梯度下降更能胜任大规模机器学习任务的要求, 但仍然弥补不了其收敛速度缓慢的缺点.

(4) 式充分结合了全梯度下降算法和随机梯度下降算法的特点, 即在每次参数更新时从样本中随机选取部分样本进行梯度估计, 在一定程度上既保证了梯度下降方向, 又降低了样本随机选取的方差. 相较于全梯度下降而言, 小批量随机梯度下降更能胜任大规模机器学习任务的要求, 但仍然弥补不了其收敛速度缓慢的缺点.

近年来, 随机梯度下降算法已成为机器学习特别是深度学习的焦点, 随着对梯度下降方向和迭代步长的不断探索, 逐渐涌现出许多基于 SGD 的改进算法. Momentum 算法 [9]在传统随机梯度下降算法的基础上添加动量项, 基于梯度动量累积的思想并结合历史梯度使用指数加权移动平均来有效避免震荡, 加速逼近最优解, Nesterov 加速梯度算法 [10,11]则是在Momentum 算法的基础上对梯度项参数做了一次迭代更新.

在梯度下降的过程当中, 由于优化参数对目标函数的依赖各不相同, 其恒定的学习率会导致参数迭代出现梯度发散或者收敛缓慢的问题, 为此考虑优化算法能否自适应地调节学习率的大小. 自适应梯度法 [12](Adaptive Gradient, AdaGrad) 通过逐步缩小步长来学习; 均方根传播法 [13](Root

Mean SquarePropagation, RMSprop)将学习率分解成平方梯度的指数衰减平均来控制 AdaGrad 算法学习率下降过快的问题; 自适应矩估计法 [14](Adaptive Moment Estimation, Adam) 可以看作 Momentum 算法和 RMSProp 算法的结合体, 利用梯度的一阶矩和二阶矩估计来对学习率进行约束; 自适应矩限制法 [15](Adaptive Momental Bound, AdaMod) 用长期记忆限制过高学习率, 降低了 Adam 算法对学习率的敏感性.

另外随机梯度下降算法在样本的随机取样产生了方差并随着迭代次数的增加而不断累积, 进而无法达到线性收敛. 为此, 研究者们提出了一系列基于方差缩减的随机下降算法, 例如随机平均梯度下降法 [16](Stochastic Average Gradient, SAG) 以“新梯度代替旧梯度”的方式, 充分考虑了历史梯度且达到减少计算量的目的; 随机对偶坐标上升法 [17](Stochastic Dual Coordinate Ascent, SDCA）在参数更新过程中存储对偶变量, 降低梯度方差加速收敛; 随机方差缩减算法 [18](Stochastic Variance Reduction Gradient, SVRG）其核心思想是利用全局梯度信息用于模型更新的梯度进行修正, 理论分析表明, 这几种方法在特定的条件下都能缩减方差并达到线性收敛.

本文第 2 部分基于自适应学习率算法和和方差缩减算法的基础上, 提出了带矩限制的批量自适应方差缩减算法(An Adaptive and Momental Bound Method for Stochastic Variance Reduced Gradient, AmbSVRG), 结合了自适应学习率和矩限制的特点, 并利用批量思想对参数进行迭代更新; 第 3 部分给出了算法的收敛性分析; 第 4 部分基于 MNIST 数据集对算法有效性进行了验证, 接着探究得出实验参数对算法稳定性的影响; 第 5 部分给出了本文的总结.

2. 算法介绍

2.1. 自适应学习率算法

在梯度下降过程中选取一个合适的学习率是很困难的, 过小的学习率会导致收敛缓慢, 而过大的学习率会导致参数会在最小值附近波动或发散, 另外在面对不同的数据集时, 应选取不同的学习率从而适应数据集的特征, 但在梯度下降算法的过程中, 所有参数更新都使用的是恒定的学习率, 由于每个参数更新时的收敛速度都不尽相同, 需要根据参数的收敛情况分别设置不同的学习率, 从而达到对算法加速的目的.

对于固定步长的参数更新公式为

$$w_t = w_{t-1} - \alpha g_t.$$

针对固定步长, AdaGrad 将每次更新的梯度累加, 使学习率适应参数, 对稀疏的数据集执行较大的更新、对稠密的数据集执行较小的更新

$$w_t = w_{t-1} - \frac{\alpha}{\sqrt{G_t + \epsilon}}\, g_t,$$

变量 G_t 存储了历史梯度平方和, ϵ 是为了防止分母为 0, 通过对学习率乘以这样一个分母项来修改每个参数迭代过程中的学习尺度, 从而消除了手动调整学习率的需要. 但 AdaGrad 算法的主要缺点

是由于每一项 G_t 都是正的, 分母会随着训练过程不断增加, 反过来导致学习率越来越小趋近于 0, 从而终止迭代. 为了解决 AdaGrad 学习率急剧下降的问题, RMSProp 引入了指数加权平均数

$$E\left[g^2\right]_t = \beta E\left[g^2\right]_{t-1} + (1-\beta)g_t^2,$$
$$w_t = w_{t-1} - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t,$$

其中 β 为指数衰减系数, AdaGrad 中的梯度平方和也换成了梯度平方衰减平均的期望.

Adam 同时获得了 AdaGrad 和 RMSProp 的优点, Adam 不仅如 RMSProp 那样基于一阶矩计算适应性参数学习率, 它同时还充分利用了梯度的二阶矩均值. 具体表示为

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\, g_t,$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\, g_t^2,$$

其中 m_t 和 v_t 是梯度的一阶矩和二阶矩估计, 由于它们初期会有冷启动的问题, 所以需要对它们做一些偏差修正

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t},$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t},$$

使用这些参数来迭代更新得到 Adam 的更新规则

$$w_t = w_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t.$$

尽管自适应学习率方法在许多情况下都很受欢迎, Adam 也被认为是深度学习框架中使用的默认算法, 但它仍然会遇到稳定性的问题：不稳定和极端的学习率. AdaMod 采用动态边界来限制极端的学习速率, 在训练的同时计算自适应学习率的指数长期平均值, 并使用该平均值来修剪训练过程中过高的学习率

$$\eta_t = \alpha_t / \sqrt{\hat{v}_t} + \epsilon,$$

$$S_t = \beta_3 S_{t-1} + (1-\beta_3)\, \eta_t,$$

$$\hat{\eta}_t = \min\left(\eta_t, s_t\right),$$

通过第二个方程定义了当前平滑值和过去“长期记忆”的关系. 显然, 当 $\beta_3 = 0$ 时, AdaMod 完全等价于 Adam, 在计算出当前平滑值后, 在它和当前 Adam 算出的学习率 η_t 中选出一个最小值, 从而避免了出现过高学习率的情况.

由于 SGD 的随机性引入的方差使得 SGD 在固定步长的情况下只能达到次线性收敛速度, 文献在 [18]指出, 随机算法的精度和采样方差成正相关, 当方差趋于 0 时, 算法的偏差也会为 0. 如何在随机算法中减小随机梯度的方差, 将 SGD 做到和全梯度下降一样的线性收敛呢? 研究者们又提出了一类方差缩减算法.

2.2. 方差缩减算法

近年来, 基于方差缩减的随机梯度下降算法成为了优化算法研究中的热点问题, 研究者们相继提出了 SAG 和 SDCA 等相关算法, 其中 SAG 算法的更新规则为

$$d = d - y_i + \nabla\psi_i(w),$$
$$w_t = w_{t-1} - \frac{\alpha}{n}d.$$

SAG 在内存中为每个样本维护一个旧梯度, 随机选择一个样本来更新参数 d, 而更新项中的 d 是用新梯度替换旧梯度得到的, 所以每次计算中只需要涉及两个梯度. SDCA 本质上通过构造对偶变量进行方差缩减, 依照对偶变量 w 和对偶变量 α 之间的关系, w 的更新可以表示成

$$w_t = w_{t-1} - \gamma(\nabla f_j(w_{t-1}) + \lambda n\alpha_j^k),$$

随着迭代的进行, (w, α) 将会收敛到 (w^*, α^*), 于是 $(\nabla f_j(w) + \lambda n\alpha_j) \to 0$, $\nabla f_j(w) + \lambda n\alpha_j$ 的方差也会趋近于 0. 这就达到了方差缩减的目的. 这两种算法都能达到线性收敛的速度, 但在参数更新过程中都要求存储全部梯度（或对偶变量）, 对内存空间具有一定的要求. SVRG 克服了 SAG 和 SDCA 对大量存储空间需求的问题

$$\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n}\nabla\psi_i(\tilde{w}),$$

$$w_t = w_{t-1} - \alpha(\nabla\psi_{it}(w_{t-1}) - \nabla\psi_{it}(\tilde{w}) + \tilde{\mu}).$$

其核心思想是利用全局梯度信息对参数迭代更新中的梯度进行修正, 每经历一轮内循环计算一次全局梯度, 每次更新最多计算两次梯度. 在 SGD 的收敛性分析中需要假设样本梯度的方差是有常数上界的, 然而正是因为这个上界导致了 SGD 无法线性收敛, 而 SVRG 利用特殊的梯度更新项使样本方差有一个可以不断减少的上界, 从而达到了降低样本方差的目的, 因此也就做到了线性收敛.

批量随机方差缩减梯度算法 [19]（Batching SVRG）利用部分样本的梯度去近似全局梯度, 在一定程度上降低了 SVRG 的计算量. SVRG 和 Batching SVRG 在迭代过程中都使用固定学习率进行优化求解, 选择适合的学习率对优化算法至关重要. 此外, 当上一次的迭代参数离当前迭代参数距离较远, 而当前参数离最优值处接近时, SVRG 和 Batching SVRG 又会过多地利用历史梯度的信息值对当前的梯度下降方向进行修正, 导致算法在最优值处震荡, 降低算法的收敛速度.

对于采用固定的学习率和过度利用历史梯度的信息的问题, 本文基于 SVRG, 根据梯度的一阶矩和二阶矩估计, 计算不同参数下的个体自适应学习速率, 提出了带矩限制的批量自适应方差缩减算法(An Adaptive and Momental Bound Method for Stochastic Variance Reduced Gradient, AmbSVRG).

2.3. AmbSVRG算法

对于问题 (1), 本文将 AdaMod 中一阶和二阶矩估计与 Batching SVRG 中梯度更新方式做结

合, 提出了带矩限制的批量自适应方差缩减算法.

AmbSVRG算法:

输入：全局学习率 α_t, 更新频率 m, 动量系数 β_1, β_2, $\beta_3 \in [0,1)$;

步0：初始化 $\tilde{w}_0 = 0$, $m_0 = 0$, $v_0 = 0$, $s_0 = 0$;

步1：对外循环 $s = 1, 2...T$, 令 $\tilde{w} = \tilde{w}_{s-1}$;

步2：选择全局梯度更新的批量尺寸大小 $|B^s| = B$, 计算全局梯度: $\tilde{\mu} = \frac{1}{B}\sum_{i=1}^{B}\nabla\psi_i(\tilde{w})$;

步3：令 $w_0 = \tilde{w}$, $s := s + 1$;

步4：对内循环 $t = 1, 2, ...m$, 选择参数更新的批量尺寸大小 $|B^t| = b$,

计算梯度项: $g_t = \frac{1}{b}\sum_{i=1}^{b}\left[\nabla\psi_i\left(w_{t-1}\right) - \nabla\psi_i\left(\tilde{w}\right)\right] + \tilde{\mu}$;

步5：计算一阶矩和二阶矩:

$m_t = \beta_1 m_{t-1} + \left(1-\beta_1\right)g_t$, $v_t = \beta_2 v_{t-1} + \left(1-\beta_2\right)g_t^2$;

步6：进行偏差修正: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$, $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$;

步7：对更新项进行指数加权平均, 限制过高学习率:

$\eta_t = \alpha_t/\sqrt{\hat{v}_t} + \epsilon$, $S_t = \beta_3 S_{t-1} + \left(1-\beta_3\right)\eta_t$, $\hat{\eta}_t = \min\left(\eta_t, s_t\right)$;

步8：参数迭代更新: $w_t = w_{t-1} - \hat{\eta}_t\hat{m}_t$;

步9：令 $t := t + 1$,若 $t < m$ 转步 4, 若 $t = m$ 转步 1.

在 AmbSVRG 算法中, 步 2 结合 Batching SVRG 算法中批量的思想, 用部分样本计算梯度去近似全局梯度, 在一定的条件下, 能够减少算法的计算量, 加快收敛; 在步 4 中, 将小批量的思想融入到 SVRG 的梯度更新规则中对梯度做无偏估计, 从而降低迭代过程中的梯度方差, 增加收敛过程中的稳定性; 在步 5 中又利用指数加权平均的思想, 对一阶梯度和二阶梯度加入动量, 充分利用梯度信息, 并在步 6 对 m_t 和 v_t 初期的冷启动问题做了偏差修正, 使得算法在面对不同数据集时能自动调整参数的学习率, 大大提高了算法的稳定性和训练速度; 在步 7, 计算自适应学习率的指数长期平均值, 并选取较小的学习率代入到的参数迭代方程, 抑制了过高学习率的产生, 进一步增加了算法的稳定性.

3. 收敛性分析

本节讨论 AmbSVRG 算法的收敛性, 首先列出问题 (1) 所需要满足的基本假设.

假设 1 对无约束极小化问题 (1), 目标函数 $\psi(w)$ 满足

(1) ψ 为凸函数;

(2) ψ 存在极小值点w^*, 即 $w^* = argmin_w P(w)$;

(3) ψ 为利普希茨连续的, 即

$$\psi(w)-\psi_i(w^*)-\frac{L}{2}\|w-w^*\|^2\leqslant\nabla\psi_i(w^*)^T(w-w^*). \tag{5}$$

其中 $L>0$ 为利普希茨常数.

假设 2 $P(w)$ 是 μ-强凸函数

$$P(w)-P(w^*)-\frac{\mu}{2}\|w-w^*\|^2\geqslant\nabla P(w^*)^T(w-w^*). \tag{6}$$

定理3.1. 在假设 *1*, 假设 *2* 的条件下, 取 $\mu>0$, 假设 m 充分大有

$$\alpha=\frac{b}{\mu mR(b+2LR)}+\frac{2LR}{b+2LR}<1,$$

得到 $\mathbb{E}[P(\tilde{w}_s)-P(w_*)]\leq\alpha^s\mathbb{E}[P(\tilde{w}_0)-P(w_*)]$, 那么 *AmbSVRG* 算法就拥有线性收敛的速度.

证明 对任意的 i, 用 g_t 表示第 t 步时的搜索方向, 则有

$$g_t=\frac{1}{b}\sum_{i=1}^{b}\left[\nabla\psi_i\left(w_{t-1}\right)-\nabla\psi_i(\tilde{w})\right]+\frac{1}{B}\sum_{i=1}^{B}\nabla\psi_i(\tilde{w}),$$

由文献 [18]可知

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla\psi_i(w)-\nabla\psi_i(w_*)\|_2^2\leqslant 2L[P(w)-P(w_*)]. \tag{7}$$

对 g_t 其取期望得到

$$\begin{aligned}
\mathbb{E}\|g_t\|_2^2&=\mathbb{E}\|\frac{1}{b}\sum_{i=1}^{b}[\nabla\psi_i(w_{t-1})-\nabla\psi_i(\tilde{w})]+\frac{1}{B}\sum_{i=1}^{B}\nabla\psi_i(\tilde{w})\|_2^2\\
&\leqslant 2\mathbb{E}\|\frac{1}{b}\sum_{j=1}^{b}\nabla\psi_i(w_{t-1})-\frac{1}{b}\sum_{i=1}^{b}\nabla\psi_i(w^*)\|_2^2+2\mathbb{E}\|\frac{1}{b}\sum_{i=1}^{b}\nabla\psi_i(\tilde{w})-\frac{1}{b}\sum_{i=1}^{b}(w^*)\|_2^2-\nabla p(\tilde{w})\|_2^2\\
&=2\mathbb{E}\|\frac{1}{b}\sum_{j=1}^{b}\nabla\psi_i(w_{t-1})-\frac{1}{b}\sum_{i=1}^{b}\nabla\psi_i(w^*)\|v^2+2\mathbb{E}\|\frac{1}{b}\sum_{i=1}^{b}\nabla\psi_i(\tilde{w})\\
&-\frac{1}{b}\sum_{i=1}^{b}(w^*)\|_2^2-\mathbb{E}[\nabla\psi_i(\tilde{w})-\nabla\psi_i(w^*)]\|_2^2\\
&\leqslant 2\mathbb{E}\|\frac{1}{b}\sum_{j=1}^{b}\nabla\psi_i(w_{t-1})-\frac{1}{b}\sum_{i=1}^{b}\nabla\psi_i(w^*)\|_2^2+2\mathbb{E}\|\frac{1}{b}\sum_{i=1}^{b}\nabla\psi_i(\tilde{w})-\frac{1}{b}\sum_{i=1}^{b}\nabla\psi_i(w^*)\|_2^2\\
&\leqslant 4\frac{L}{b}[P(w_{t-1})-P(w^*)+P(\tilde{w})-P(w^*)].
\end{aligned} \tag{8}$$

第一个不等式使用了 $\|a+b\|_2^2 \leqslant 2\|a\|_2^2 + 2\|b\|_2^2$, 用 $\tilde{\mu} = \frac{1}{B}\sum_{i=1}^{B} \nabla\psi_i(\tilde{w})$ 代替 $\nabla P(\tilde{w})$, 且 $\nabla P(\tilde{w}) = \mathbb{E}[\nabla\psi_i(\tilde{w}) - \nabla\psi_i(w^*)]$; 第二个不等式使用了对任意的 ξ, 满足 $\mathbb{E}\|\xi - \mathbb{E}\xi\|_2^2 = \mathbb{E}\|\xi\|_2^2 - \|\mathbb{E}\xi\|_2^2 \leqslant \mathbb{E}\|\xi\|_2^2$; 第三个不等式使用了 (7) 式.

由 (8) 式可知 $\|g_t\|$ 是有上界的, 不妨设为 G, 在 AmbSVRG 算法的步 5 中, $m_t = \beta_1 m_{t-1} + (1-\beta_1)\, g_t$, 当 $m_0 = 0$ 时, 对式子展开可得

$$m_t = (1-\beta_1)\left(g_t + \beta_1 g_{t-1} + \beta_1^2 g_{t-2} + \cdots + \beta_1^{t-1} g_1\right),$$

求其期望为

$$\begin{aligned}
\mathbb{E}\|\hat{m}_t\|_2^2 &= \mathbb{E}\|\frac{1-\beta_1}{(1-\beta_1^t)}(g_t + \beta_1 g_{t-1} + \beta_1^2 g_{t-2} + \cdots + \beta_1^{t-1} g_1\|_2^2 \\
&= \frac{1-\beta_1}{(1-\beta_1^t)}\mathbb{E}\|g_t + \beta_1 g_{t-1} + \beta_1^2 g_{t-2} + \cdots + \beta_1^{t-1} g_1\|_2^2 \\
&\leqslant \frac{1-\beta_1}{(1-\beta_1^t)}(1 + \beta_1 + \beta_1^2 + \cdots + \beta_1^{t-1})G = G. \qquad (9)
\end{aligned}$$

由 (9) 式可知 $\|\hat{m}_t\|$ 有界, 同理算法步 6 中的 $\|\hat{v}_t\|$ 和步 7 中的 $\|\hat{\eta}_t\|$ 也是有上界的, 即存在 $R > 0$, 使得 $\|\hat{\eta}_t\| \leqslant R$, 有

$$\begin{aligned}
\mathbb{E}\|w_t - w^*\|_2^2 &= \mathbb{E}\|w_{t-1} - \hat{\eta}_t\hat{m}_t - w^*\|_2^2 \\
&= \|w_{t-1} - w^*\|_2^2 + 2R(w_{t-1} - w^*)^\top \mathbb{E}[\widehat{m}_t] + R^2\mathbb{E}\|\hat{m}_t\|_2^2 \\
&\leqslant \|w_{t-1} - w^*\|_2^2 + 2R(w_{t-1} - w^*)^\top \nabla P(w_{t-1}) \\
&+ \frac{4LR^2}{b}[p(w_{t-1}) - p(w^*) + p(\tilde{w}) - p(w^*)] \\
&\leqslant \|w_{t-1} - w^*\|_2^2 - 2R[p(w_{t-1}) - p(w^*)] \\
&+ \frac{4LR^2}{b}[p(w_{t-1}) - p(w^*) + p(\tilde{w}) - p(w^*)] \\
&= \|w_{t-1} - w^*\|_2^2 - 2R(1 + 2\frac{LR}{b})[p(w_{t-1}) - p(w^*)] \\
&+ \frac{4LR^2}{b}[p(\tilde{w}) - p(w^*)],
\end{aligned}$$

其中第一个不等式由 $\mathbb{E}[\widehat{m}_t] \leqslant \mathbb{E}[g_t] = \nabla P(w_{t-1})$ 可得, 第二个不等式使用了 $P(w)$ 的凸性

$$p(w_{t-1}) - p(w^*) \leqslant (w_{t-1} - w^*)^\top \nabla P(w_{t-1}).$$

将上述的不等式对 $t=1,2,...,m$ 求和, 可得

$$
\begin{aligned}
&E\|w_m-w^*\|_2^2+2mR(1+2\frac{LR}{b})\mathbb{E}[P(\tilde{w}_s)-P(w^*)]\\
&\leqslant \mathbb{E}\|w_0-w^*\|_2^2+\frac{4mLR^2}{b}\mathbb{E}[P(\tilde{w})-P(w*)]\\
&=\mathbb{E}\|\tilde{w}-w^*\|_2^2+\frac{4mLR^2}{b}\mathbb{E}[P(\tilde{w})-P(w^*)]\\
&\leqslant \frac{2}{\mu}\mathbb{E}[P(\tilde{w})-P(w^*)]+\frac{4mLR^2}{b}\mathbb{E}[P(\tilde{w})-P(w^*)]\\
&=2(\mu^{-1}+\frac{2mLR^2}{b})\mathbb{E}[P(\tilde{w})-p(w^*)],
\end{aligned}
$$

对第二个不等式使用了 (7) 式, 由此可以得到：

$$
\mathbb{E}[P(\tilde{w}_s)-P(w^*)]\leqslant\left[\frac{b}{\mu mR(b+2LR)}+\frac{2LR}{b+2LR}\right]\mathbb{E}[P(\tilde{w}_{s-1})-P(w^*)],
$$

从而算法的收敛性：$\mathbb{E}[P(\tilde{w}_s)-P(w_*)]\leq\alpha^s\mathbb{E}[P(\tilde{w}_0)-P(w_*)]$ 得证. □

由算法的收敛性可以看出, 算法的收敛速度与参数 b 和 m 有关, 关于小批量参数 b 和更新频率 m 对算法收敛速度及稳定性的影响将在数值实验部分展开说明.

4. 数值实验

本文算法基于Python3.7完成, 整体网络架构参考文献 [20], 其中激活函数为 ReLU 函数, 对于算法的实现, 本设备的参数为 CPU: Intel(R) Core(TM) i5-6200U 2.40 GHz, 内存: 4.00GB. 使用主流数据集 MNIST 对算法的有效性进行验证, 数据集主要的参数见表 1：

Table 1. Data set introduction
表 1. 数据集介绍

8	图片类型	图片大小	训练集数量	测试集数量
MNIST	灰度图	28*28	20000	10000

数值实验总共包括两部分：第 1 部分主要是本文算法与当前主流算法之间的对比; 第 2 部分则是算法参数对算法稳定性的影响.

4.1. 主流算法对比

基于Python3.7完成 AmbSVRG 算法的编写, 将其与主流算法 Mini-Batch SGD、Batching SVRG、AdaGrad、RMSProp 算法进行对比.

从图 1可以看出：

(1)从初始迭代准确率上看, AmbSVRG 在迭代到第一个 epoch 时, 其测试集准确率达到了 60%以上, 而 Batching SVRG 和 RMSProp 分别达到了 53% 和 48%, Mini-Batch SGD 和 Adagrad只达到了40%, 可见, 在初始迭代准确率上, AmbSVRG 要优于其他算法.

(2)从收敛过程上看, 除了 AdaGrad 算法收敛过程不太平滑之外, 其他算法都能稳定地收敛到最优解; 从测试集准确率上升的坡度来看, 在迭代初期, AmbSVRG 上升的坡度最陡, 可见其收敛速度最快.

(3)从算法准确率看, AmbSVRG 算法的准确率达到了 91.4%, RMSProp 只比其低了 0.8 个百分点, Batching SVRG 的准确率达到了 85.7%, 但仍比 AdaGrad 和 Mini-Batch SGD 算法优秀.

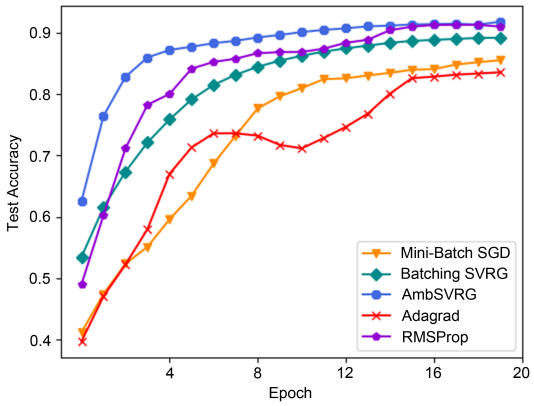

Figure 1. Accuracy comparison of mainstream algorithms
图 1. 主流算法准确率对比

4.2. 稳定性探究

由于算法的稳定性主要受小批量尺寸大小 b 以及更新频率 m 影响, 本文将对不同参数情况下算法测试集的准确率进行探究, 分别从 MNIST 数据集中选取小批量尺寸大小为 100、300、500 的样本, 代入到更新频率为 10、30、50 的算法中, 得出不同参数情况下算法的训练时间和测试集准确率, 实验参数及对应的评价指标见表 2. 为了避免实验的偶然性, 对实验数据集在相同参数条件下进行三次同等实验, 并取三次实验平均值作为最终结果.

Table 2. The influence of experimental parameters on the convergence effect of the algorithm
表 2. 实验参数对算法收敛效果的影响探究

更新频率m	$b=100$		$b=300$		$b=500$	
	train time	test accuracy	train time	test accuracy	train time	test accuracy
$m=10$	19. 34s	85. 2%	22. 26s	88. 9%	23. 86s	91. 4%
$m=30$	24. 22s	87. 5%	28. 69s	89. 1%	35. 12s	89. 5%
$m=50$	32. 05s	86. 0%	36. 71s	87. 2%	46. 89s	86. 7%

由表 2可知, 在训练时间上, 随着小批量尺寸大小 b 和更新频率 m 的增加, 算法的计算代价加大, 其相对的训练时间也逐渐增加; 在测试集准确率上, 条件 $b=500$ 时的算法测试集准确率总体要比 $b=300$ 和 $b=100$ 的情况下更为可观. 由此可以看出, 小批量尺寸大小 b 对算法测试集准确率有直接影响, 且具有一定的正比例关系; 在 b 与 m 的关系探究上, 可以看到当 $m=10$ 和 $m=30$ 时, 随着 b 的增加, 算法测试集准确率也逐渐增加, 但当 $m=50$ 时,b 的增加可能会带来准确率的下降, 可能 m 的增加导致内循环的参数更新梯度和外循环全局梯度相差变大, 对收敛速度和收敛准确率造成了一定的影响.

评论 通过4.1节可以看出, AmbSVRG 算法的收敛速度和收敛精度表现突出, 其收敛精度和 RMSProp 算法相当, 并且在初期算法的收敛速度优于其他算法. 通过4.2节, 了解到小批量尺寸大小 b 和更新频率 m 对算法收敛速度和收敛精度的影响, 对算法收敛效果的调参具有一定的参考意义.

5. 结论

本文基于随机方差缩减梯度算法, 结合了自适应学习率和矩限制的特点, 并利用批量思想, 选取大批量样本代替全部样本进行全局梯度的计算, 选取小批量样本进行参数的迭代更新, 克服了随机梯度下降算法由于样本选取的随机性引入的高方差问题, 以及梯度下降法恒定的学习率导致参数在最优点来回震荡、收敛速度缓慢的缺点, 同时针对自适应学习率算法抑制了过高学习率的产生. 因此 AmbSVRG 算法具有良好稳定性和较快的收敛速度, 并且实验结果表明, 在与主流算法的对比下, AmbSVRG 算法的有效性也得到了验证.

受制于设备的原因, 无法验证算法在复杂网络结构上的有效性, 同时由于参数的选取对算法的计算效率有一定的影响, 因此如何选取参数对提高算法的计算效率也具有一定的现实意义.

参考文献

[1] Bottou, L. (2010) Large-Scale Machine Learning with Stochastic Gradient Descent. In: Lechevallier, Y. and Saporta, G., Eds., *Proceedings of COMPSTAT'2010*, Physica-Verlag HD, 177-186. https://doi.org/10.1007/978-3-7908-2604-3_16

[2] Sebastian, R. (2016) An Overview of Gradient Descent Optimization Algorithms. *Computer Science Machine Learning*, 1-12.

[3] Lecun, Y. and Bottou, L. (1998) Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, **86**, 2278-2324. https://doi.org/10.1109/5.726791

[4] Bubeck, S. (2015) Convex Optimization: Algorithms and Complexity. *Foundations and Trends in Machine Learning*, **8**, 231-357. https://doi.org/10.1561/2200000050

[5] Bottou, L., Curtis, F.E. and Nocedal, J. (2018) Optimization Methods for Large-Scale Machine Learning. *SIAM Review*, **60**, 223-311. https://doi.org/10.1137/16M1080173

[6] Nesterov, Y. (2013) Gradient Methods for Minimizing Composite Functions. *Mathematical Programming*, **140**, 125-161. https://doi.org/10.1007/s10107-012-0629-5

[7] Robbins, H. and Monro, S. (1951) A Stochastic Approximation Method. *Annals of Mathematical Statistics*, **22**, 400-407. https://doi.org/10.1214/aoms/1177729586

[8] Li, M., Zhang, T., Chen, Y., *et al.* (2014) Efficient Mini-Batch Training for Stochastic Optimization. *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, New York, 24-27 August 2014, 661-670.

[9] Qian, N. (1999) On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Networks*, **12**, 145-151. https://doi.org/10.1016/S0893-6080(98)00116-6

[10] Nesterov, Y. (1983) A Method of Solving a Convex Programming Problem with Convergence Rate O(1/k^2). *Soviet Mathematics Doklady*, **27**, 372-376.

[11] Botev, A., Lever, G. and Barber, D. (2017) Nesterov's Accelerated Gradient and Momentum as Approximations to Regularised Update Descent. *2017 International Joint Conference on Neural Networks*, Anchorage, AK, 14-19 May 2017, 1899-1903.

[12] Duchi, J., Hazan, E. and Singer, Y. (2011) Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, **12**, 2121-2159.

[13] Tieleman, T. and Hinton, G. (2012) Lecture 6.5-RMSProp: Divide the Gradient by a Running Average of Its Recent Magnitude. *COURSERA: Neural Networks for Machine Learning*, **4**, 26-30.

[14] Kingma, D. and Ba, J. (2015) Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, 7-9 May 2015, 1-13.

[15] Ding, J., Ren, X., Luo, R., *et al.* (2019) An Adaptive and Momental Bound Method for Stochastic Learning. arXiv: 1910.12249

[16] Schmidt, M., Le Roux, N. and Bach, F. (2017) Minimizing Finite Sums with the Stochastic Average Gradient. *Mathematical Programming*, **162**, 83-112.

[17] Shalev, S. and Zhang, T. (2013) Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization. *Journal of Machine Learning Research*, **14**, 567-599. https://doi.org/10.1007/s10107-016-1030-6

[18] Johnson, R. and Zhang, T. (2013) Accelerating Stochastic Gradient Descent Using Predictive Variance Reduction. *Proceedings of the 27th Neural Information Processing Systems*, Lake Tahoe, 5-10 December 2013, 315-323.

[19] Babanezhad, R., Ahmed, M.O., Virani, A., *et al.* (2015) Stop Wasting My Gradients: Practical SVRG. *Proceedings of the 28th International Conference on Neural Information Processing Systems*, **2**, 2251-2259.

[20] Qiu, X.P. (2020) Neural Networks and Deep Learning. China Machine Press, Beijing, 160-169.