随机梯度下降算法研究进展

史加荣[1,2] 王 丹[1] 尚凡华[3] 张鹤于[4]

摘 要 在机器学习领域中, 梯度下降算法是求解最优化问题最重要、最基础的方法. 随着数据规模的不断扩大, 传统的梯度下降算法已不能有效地解决大规模机器学习问题. 随机梯度下降算法在迭代过程中随机选择一个或几个样本的梯度来替代总体梯度, 以达到降低计算复杂度的目的. 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研究的焦点. 随着对搜索方向和步长的不断探索, 涌现出随机梯度下降算法的众多改进版本, 本文对这些算法的主要研究进展进行了综述. 将随机梯度下降算法的改进策略大致分为动量、方差缩减、增量梯度和自适应学习率等四种. 其中, 前三种主要是校正梯度或搜索方向, 第四种对参数变量的不同分量自适应地设计步长. 着重介绍了各种策略下随机梯度下降算法的核心思想、原理, 探讨了不同算法之间的区别与联系. 将主要的随机梯度下降算法应用到逻辑回归和深度卷积神经网络等机器学习任务中, 并定量地比较了这些算法的实际性能. 文末总结了本文的主要研究工作, 并展望了随机梯度下降算法的未来发展方向.

关键词 随机梯度下降算法, 机器学习, 深度学习, 梯度下降算法, 大规模学习, 逻辑回归, 卷积神经网络

引用格式 史加荣, 王丹, 尚凡华, 张鹤于. 随机梯度下降算法研究进展. 自动化学报, 2021, **47**(9): 2103−2119

DOI 10.16383/j.aas.c190260

Research Advances on Stochastic Gradient Descent Algorithms

SHI Jia-Rong[1,2] WANG Dan[1] SHANG Fan-Hua[3] ZHANG He-Yu[4]

Abstract In the field of machine learning, gradient descent algorithm is the most significant and fundamental method to solve optimization problems. With the continuous expansion of the scale of data, the traditional gradient descent algorithms can not effectively solve the problems of large-scale machine learning. Stochastic gradient descent algorithm selects one or several sample gradients randomly to represent the overall gradients in the iteration process, so as to reduce the computational complexity. In recent years, stochastic gradient descent algorithm has become the research focus of machine learning, especially deep learning. With the constant exploration of search directions and step sizes, numerous improved versions of the stochastic gradient descent algorithm have emerged. This paper reviews the main research advances of these improved versions. The improved strategies of stochastic gradient descent algorithm are roughly divided into four categories, including momentum, variance reduction, incremental gradient and adaptive learning rate. The first three categories mainly correct gradient or search direction and the fourth designs adaptively step sizes for different components of parameter variables. For the stochastic gradient descent algorithms under different strategies, the core ideas and principles are analyzed emphatically, and the difference and connection between different algorithms are investigated. Several main stochastic gradient descent algorithms are applied to machine learning tasks such as logistic regression and deep convolutional neural networks, and the actual performance of these algorithms is numerically compared. At the end of the paper, the main research work of this paper is summarized, and the future development direction of the stochastic gradient descent algorithms is prospected.

Key words Stochastic gradient descent algorithm, machine learning, deep learning, gradient descent algorithm, large-scale learning, logistic regression, convolutional neural networks

Citation Shi Jia-Rong, Wang Dan, Shang Fan-Hua, Zhang He-Yu. Research advances on stochastic gradient descent algorithms. *Acta Automatica Sinica*, 2021, **47**(9): 2103−2119

作为人工智能目前最活跃的一个研究分支, 机器学习根据经验数据来设计、开发算法, 其目的是

收稿日期 2019-03-28 录用日期 2019-07-30
Manuscript received March 28, 2019; accepted July 30, 2019
国家自然科学基金 (61876220, 61876221), 中国博士后科学基金 (2017M613087) 资助
Supported by National Natural Science Foundation of China (61876220, 61876221) and China Postdoctoral Science Foundation (2017M613087)
本文责任编委 王鼎
Recommended by Associate Editor WANG Ding

1. 西安建筑科技大学理学院 西安 710055 2. 省部共建西部绿色建筑国家重点实验室 西安 710055 3. 西安电子科技大学人工智能学院智能感知与图像理解教育部重点实验室 西安 710071 4. 西安电子科技大学计算机科学与技术学院 西安 710071

1. School of Science, Xi′an University of Architecture and Technology, Xi′an 710055 2. State Key Laboratory of Green Building in Western China, Xi′an University of Architecture and Technology, Xi′an 710055 3. Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, School of Artiflcial Intelligence, Xidian University, Xi′an 710071 4. School of Computer Science and Technology, Xidian University, Xi′an 710071

探索数据的生成模式, 并用来发现模式和进行预测[1-2]. 深度学习是一类更广的机器学习方法, 允许由多个处理层组成的计算模型来学习具有多个抽象级别的数据表示[3]. 伴随着深度学习的崛起, 机器学习重新受到了学术界和工业界的广泛关注. 机器学习技术已广泛地应用在计算机视觉、推荐系统、语音识别和数据挖掘等领域中.

回归与分类等监督学习是机器学习中最常见的一类学习问题, 它提供了包含输入数据和目标数据的训练数据集. 为了探讨输入与目标之间的关系, 需要先建立含参数的表示模型, 再通过最小化所有样本的平均损失函数来获得最优的参数, 此处的优化模型通常为经验风险最小化 (Empirical risk minimization, ERM)[4]. 梯度下降法是求解 ERM 模型最常用的方法, 也是二阶方法和黎曼优化的重要基础. 传统的梯度下降法在计算目标函数的梯度时, 需计算每个样本对应的梯度, 总计算复杂度线性地依赖于样本数目. 随着数据规模的日益增大, 求解所有样本的梯度需要相当大的计算量, 因而传统的梯度下降算法在解决大规模机器学习问题时往往不再奏效[5].

随机梯度下降算法 (Stochastic gradient descent, SGD) 源于 1951 年 Robbins 和 Monro[6] 提出的随机逼近, 最初应用于模式识别[7] 和神经网络[8]. 这种方法在迭代过程中随机选择一个或几个样本的梯度来替代总体梯度, 从而大大降低了计算复杂度. 1958 年 Rosenblatt 等研制出的感知机采用了随机梯度下降法的思想, 即每轮随机选取一个误分类样本, 求其对应损失函数的梯度, 再基于给定的步长更新参数[9]. 1986 年 Rumelhart 等分析了多层神经网络的误差反向传播算法, 该算法每次按顺序或随机选取一个样本来更新参数, 它实际上是小批量梯度下降法的一个特例[9]. 近年来, 随着深度学习的迅速兴起, 随机梯度下降算法已成为求解大规模机器学习优化问题的一类主流且非常有效的方法. 目前, 随机梯度下降算法除了求解逻辑回归、岭回归、Lasso、支持向量机[10] 和神经网络等传统的监督机器学习任务外, 还成功地应用于深度神经网络[11-12]、主成分分析[13-14]、奇异值分解[13, 15]、典型相关分析[16]、矩阵分解与补全[17-18]、分组最小角回归[19-20]、稀疏学习和编码[21-22]、相位恢复[23] 以及条件随机场[24] 等其他机器学习任务.

随着大数据的不断普及和对优化算法的深入研究, 衍生出随机梯度下降算法的许多不同版本. 这些改进算法在传统的随机梯度下降算法的基础上引入了许多新思想, 从多个方面不同程度地提升了算法性能. 搜索方向的选取和步长的确定是梯度下降算法研究的核心. 按照搜索方向和步长选取的方式不同, 将随机梯度下降算法的改进策略大致分为动量、方差缩减、增量梯度和自适应学习率等四种类型. 其中, 前三类方法主要是校正梯度或搜索方向, 适用于逻辑回归、岭回归等凸优化问题; 第四类方法针对参数变量的不同分量自适应地设置步长, 适用于深度神经网络等非凸优化问题.

在传统梯度下降算法的基础上添加动量项可以有效避免振荡, 加速逼近最优解. 采用动量更新策略的方法主要包括经典动量算法 (Classical momentum, CM)[25] 和 Nesterov 加速梯度算法 (Nesterov's accelerated gradient, NAG)[26-27]. 简单版本的随机梯度下降算法在随机取样的过程中产生了方差并且随着迭代次数的增加而不断累加, 无法保证达到线性收敛. 为此, 研究者们相继提出了一系列基于方差缩减的随机梯度下降算法, 主要包括随机方差缩减梯度算法 (Stochastic variance reduced gradient, SVRG)[28]、近端随机方差缩减梯度算法 (Proximal stochastic variance reduction gradient, Prox-SVRG)[29]、Katyusha[30] 和 MiG[31] 等. 前述方法没有充分利用历史梯度信息, 而增量梯度策略通过 "以新梯度替代旧梯度"的方式, 充分考虑了历史梯度且达到了减少梯度计算量的目的, 该类型的主要算法包括随机平均梯度算法 (Stochastic average gradient, SAG)[32]、SAGA[33] 和 Point-SAGA[34]. Allen-Zhu[30] 根据算法在强凸条件下的复杂度将前三类随机梯度下降算法分为三代, 复杂度随代数的增加而降低. 在深度神经网络中, 自适应学习率的随机梯度下降法通过使用反向传播所计算出的梯度来更新参数[35]. 与前三类算法不同, 自适应算法在训练过程中会根据历史梯度信息, 针对参数的不同分量自动调整其对应的学习率. 这类算法主要包括 Adagrad[36]、Adadelta[37]、Adam (Adaptive moment estimation)[38] 和 Nadam (Nesterov-accelerated adaptive moment estimation)[39] 等.

目前, 各种版本的随机梯度下降算法大多以黑箱优化器的形式在 TensorFlow、PyTorch 和 MxNet 等各大主流平台供用户调用, 但背后的算法原理却鲜为人知. 2017 年 Ruder 在文献 [40] 中介绍了几种深度学习领域中的随机梯度下降算法, 但缺乏一些最新的研究成果、算法之间的联系以及实际数据集上的实验对比. 国内学者提出过一些随机梯度下降算法的改进策略, 但尚未有人发表过此方向的综述性论文. 因此, 本文的工作对于关注梯度下降算法理论及其在深度学习中应用的研究者具有参

考意义. 本文针对随机梯度下降算法展开研究, 讨论了动量、方差缩减、增量梯度和自适应学习率等四类更新策略下主要算法的核心思想、迭代公式以及算法之间的区别与联系. 对于逻辑回归、岭回归、Lasso 和深度神经网络等机器学习任务, 设计了相应的数值实验, 并对比了几种具有代表性的随机梯度下降算法的性能. 文末对研究工作进行了总结, 并展望了随机梯度下降算法面临的挑战与未来的发展方向.

1　预备知识

本节先引入经验风险最小化模型, 再简要介绍凸优化的基本知识, 最后给出了 3 类梯度下降算法.

1.1　经验风险最小化

在监督学习中, 考虑 n 个独立同分布的样本构成的训练集 $\{(\boldsymbol{x}_1,y_1),\cdots,(\boldsymbol{x}_n,y_n)\}$, 其中 $\boldsymbol{x}_i\in\mathbf{R}^r$ 是第 i 个样本的输入特征向量, $y_i\in\mathbf{R}$ 是第 i 个样本的标签 (或目标), $i=1,2,\cdots,n$. 对于样本 $\boldsymbol{x}$, 可通过决策函数来预测其标签 $\hat{y}=q(\boldsymbol{x},\boldsymbol{\theta})$, 其中 $\boldsymbol{\theta}\in\mathbf{R}^d$ 为可学习的模型参数. 记第 i 个样本关于 $\boldsymbol{\theta}$ 的损失函数为 $f_i(\cdot):\mathbf{R}^d\to\mathbf{R}$, 显然 $f_i(\boldsymbol{\theta})$ 是由 $q(\boldsymbol{x}_i,\boldsymbol{\theta})$ 和 y_i 构成的函数.

通过最小化训练集的平均损失函数可学习到最优的参数 $\boldsymbol{\theta}^*$, 即求解经验风险最小化 (ERM)

$$\min_{\boldsymbol{\theta}\in\mathbf{R}^d} J(\boldsymbol{\theta})=\frac{1}{n}\sum_{i=1}^{n}f_i(\boldsymbol{\theta}) \tag{1}$$

模型 (1) 对应的期望风险为 $\mathrm{E}_{\boldsymbol{\xi}}[J(\boldsymbol{\theta};\boldsymbol{\xi})]$, 此处将 $(\boldsymbol{x}_i^{\mathrm{T}},y_i)^{\mathrm{T}}$ 视为 $r+1$ 维随机变量 $\boldsymbol{\xi}$ 的第 i 组观测值. 为了避免过拟合、降低模型复杂度, 通常对平均损失函数 $J(\boldsymbol{\theta})$或每个损失函数 $f_i(\boldsymbol{\theta})$添加正则函数 $h(\boldsymbol{\theta})$. 正则函数一般为简单的凸函数, 但可能是非光滑的, 例如 ℓ_2 范数和 ℓ_1 范数正则.

1.2　凸优化基础知识[41]

定义 1. 设 S 为实向量空间 $\mathbf{R}^d$ 中的非空凸集, $f:S\to\mathbf{R}$ 为一可微函数. 若 $\exists\sigma>0$, 对 $\forall\boldsymbol{x}_1,\boldsymbol{x}_2\in S$, 有

$$f(\boldsymbol{x}_1)\geq f(\boldsymbol{x}_2)+(\nabla f(\boldsymbol{x}_2))^{\mathrm{T}}(\boldsymbol{x}_1-\boldsymbol{x}_2)+\frac{\sigma}{2}\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2^2 \tag{2}$$

则称 f 在 S 上是 $\sigma-$ 强凸的, 其中, $\nabla f(\boldsymbol{x})$ 表示函数 f 在 $\boldsymbol{x}$ 处的梯度, $\|\cdot\|_2$ 表示向量的 ℓ_2 范数. 若 $\sigma=0$ 时上式仍然成立, 称 f 在 S 上是凸的.

定义 2. 设 $S\subset\mathbf{R}^d$ 为非空凸集, 函数 $f:S\to\mathbf{R}$. 给定 $\boldsymbol{x}_0\in S$, 若对 $\forall\boldsymbol{x}\in S$, $\exists\boldsymbol{g}\in\mathbf{R}^d$ 满足

$$f(\boldsymbol{x})-f(\boldsymbol{x}_0)\geq\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}_0) \tag{3}$$

则称向量 $\boldsymbol{g}$ 为函数 f 在点 $\boldsymbol{x}_0$ 处的梯度 (或次梯度). 特别地, 如果 f 是凸函数, 则其在任意 $\boldsymbol{x}_0\in S$ 处的梯度 (或次梯度) 总是存在的.

定义 3. 若存在一个常数 $L>0$, 对 $\forall\boldsymbol{x}_1,\boldsymbol{x}_2\in\mathbf{R}^d$ 都有

$$\|\nabla f(\boldsymbol{x}_1)-\nabla f(\boldsymbol{x}_2)\|_2\leq L\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2 \tag{4}$$

则称函数 f 是 $L-$光滑的或 ∇f 是 $L-$Lipschitz 连续的, 其中 L 为 Lipschitz 常数.

定义 4. 若函数 f 是 $L-$光滑且 $\sigma-$强凸的, 则称 f 的 Lipschitz 常数 L 与强凸系数 σ 的比值为 f 的条件数 κ , 即 $\kappa=L/\sigma$.

1.3　梯度下降算法

求解模型 (1) 的梯度下降算法包括全梯度下降 (Full gradient descent, FGD)[42]、随机梯度下降 (SGD)[43] 和小批量梯度下降 (Mini-batch gradient descent, Mini-batch)[44-45].

FGD 以目标函数的全梯度 (即全部子成分函数梯度的平均值) 迭代求解, 参数更新式为

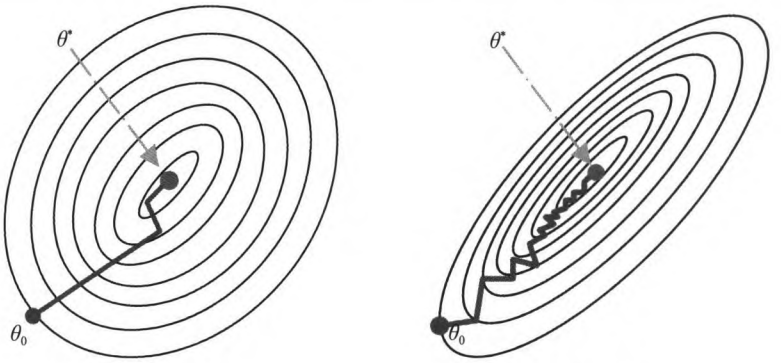
$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\frac{\alpha_t}{n}\sum_{i=1}^{n}\nabla f_i(\boldsymbol{\theta}_t) \tag{5}$$

其中, α_t 为第 t 轮迭代的学习率, 用于调整参数更新的幅度. 为防止学习率过大而错过最优解, 常将其设置为一个较小的常量或递减的序列. 当目标函数 J 为凸函数时, FGD 可取得次线性收敛速度 $\mathrm{O}(1/t)$; 当 J 为强凸函数时, FGD 可取得线性收敛速度 $\mathrm{O}(\rho^t)$, 其中常数 $\rho\in(0,1)$. ρ 的值取决于 J 的条件数 κ, 且条件数越小收敛速度越快[46], 如图 1 所示. FGD 具有较快的收敛速度, 但其迭代成本线性地依赖于样本总数 n . 在求解大规模机器学习问题时, FGD 运行时间长, 优化效率低.

SGD 在每轮更新参数时, 仅随机抽取一个样本或子成分函数计算其梯度, 并以此梯度为全局梯度的估计值. SGD 的参数更新式为

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\alpha_t\nabla f_{i_t}(\boldsymbol{\theta}_t) \tag{6}$$

其中, $i_t\in\{1,2,\cdots,n\}$ 表示第 t 轮迭代中按均匀分布随机抽取的序号. SGD 的参数更新过程简单、高效, 且迭代成本独立于 n . 但由于实际数据存在噪声, 使用 SGD 常难以沿着最佳的更新方向逼近最优参数. 当目标函数 J 分别为凸函数和强凸函数时, SGD 可取得次线性收敛速度 $\mathrm{O}(1/\sqrt{t})$ 和 $\mathrm{O}(1/t)$. 对于 FGD 和 SGD, 图 2 展示了目标参数由初始点向最优参数逐渐逼近的轨迹, 可以看出: 在每次迭代时, FGD 都能改善目标函数, 而 SGD 却振荡地收敛至最优解.

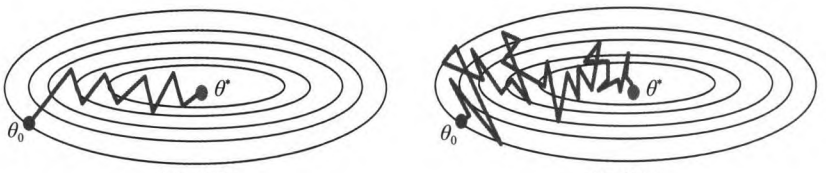


(a) 较小的 κ
(a) Small κ

(b) 较大的 κ
(b) Large κ

图 1 条件数对收敛速度的影响

Fig. 1 Effect of conditional number on convergence speed



(a) FGD (b) SGD

图 2 FGD 和 SGD 的优化轨迹示意图

Fig. 2 Schematic diagram of optimization process of FGD and SGD

作为 FGD 和 SGD 的折中方案, Mini-batch 在一定程度上兼顾了两种算法的优点. Mini-batch 在每轮迭代中随机抽取若干个样本, 以这些样本的梯度平均值作为本轮全局梯度的估计值. 称所抽取样本的数目为批容量 (batch-size), 若 batch-size $= 1$, 则 Mini-batch 变成了 SGD; 若 batch-size $= n$, 则变成了 FGD. Mini-batch 的参数更新式为

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\alpha_t}{B_t}\sum_{k=1}^{B_t}\nabla f_{i_k}(\boldsymbol{\theta}_t) \tag{7}$$

其中, B_t 表示第 t 轮更新所选样本的批容量. 在实际操作中, 通常将全体训练样本随机分成数目大致相等的若干组, 一般每组含 $10 \sim 100$ 个样本. 在每轮迭代时, 依次抽取其中一组样本用于更新目标参数; 待所有分组样本使用完毕后, 再对全体训练样本重新随机分组, 继续下一轮更新[47]. 当 J 为凸函数时, Mini-batch 可实现次线性收敛速度 $\mathrm{O}(1/\sqrt{B_t t}+1/t)$ [45].

2 基于动量的随机梯度下降算法

SGD 所生成的梯度方向常与目标函数的峡谷长轴垂直, 并沿其短轴来回振荡, 因此目标参数在长轴上缓慢移动, 无法快速到达目标函数的谷底. 物理学中的 "动量"可以有效地避免峡谷中的振荡, 从而加快在长轴上的位移. Qian[25] 证明了结合动量的梯度下降算法与保守力场中的牛顿粒子运动具有统一性, 从而得出了在梯度下降算法基础上添加动量项可以提升优化效率的结论.

2.1 随机经典动量算法

随机经典动量算法 (CM) 在 SGD 的基础上添加了动量项, 综合了历史参数改变量, 以加快优化进程[25, 48]. 在 SGD 的更新式中, 令 $\Delta\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t$, 则有 $\Delta\boldsymbol{\theta}_t = -\alpha_t \nabla f_{i_t}(\boldsymbol{\theta}_t)$. CM 的动量更新式为

$$\Delta\boldsymbol{\theta}_t = -\alpha_t \nabla f_{i_t}(\boldsymbol{\theta}_t) + \rho\Delta\boldsymbol{\theta}_{t-1} \tag{8}$$

其中, ρ 为动量系数 (一般取 0.9).

关于添加动量项的有效性, 一些学者认为可以将梯度看作施加在粒子上的力, 将 $\boldsymbol{v}_t = -\Delta\boldsymbol{\theta}_t$ 看作速度, 通过力改变速度, 从而改变位置[40]. 结合式 (8), 有

$$\boldsymbol{v}_t = \alpha_t \nabla f_{i_t}(\boldsymbol{\theta}_t) + \rho\boldsymbol{v}_{t-1} \tag{9}$$

其中, $\rho\,\boldsymbol{v}_{t-1}$ 为动量项, $\boldsymbol{v}_{t-1}$ 为历史累积梯度. 因此, CM 的更新式亦可写为

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{v}_t \tag{10}$$

2.2 Nesterov 加速梯度算法

CM 存在这样一个问题: 不断累积速度, 无限加速, 可能会错过最优解. Nesterov 加速梯度算法

(NAG) 在计算梯度时考虑了历史梯度, 亦可看作向 CM 添加了一个校正因子[26], 参数更新式为

$$\boldsymbol{v}_t = \alpha_t \nabla f_{i_t}(\boldsymbol{\theta}_t - \rho \boldsymbol{v}_{t-1}) + \rho \boldsymbol{v}_{t-1} \tag{11}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{v}_t \tag{12}$$

NAG 通过计算 $\boldsymbol{\theta}_t - \rho \boldsymbol{v}_{t-1}$ 来粗略估计下一次的位置, 以至于在坡度再次上升前能及时减速, 这种动量形式被称为 Nesterov 动量[27].

图 3 直观地展示了 SGD、CM 和 NAG 的参数更新过程. 在 SGD 的第 t 轮迭代中, 参数 $\boldsymbol{\theta}_t$ 沿负梯度方向移动步长 α_t, 即可得到更新参数 $\boldsymbol{\theta}_{t+1}^{(\mathrm{SGD})}$. 在 CM 的第 t 轮迭代中, $\boldsymbol{\theta}_t$ 在 SGD 的更新基础上又沿历史累积梯度的负方向移动步长 ρ, 得到更新参数 $\boldsymbol{\theta}_{t+1}^{(\mathrm{CM})}$. 在 NAG 的第 t 轮迭代中, $\boldsymbol{\theta}_t$ 先沿历史累积梯度的负方向移动步长 ρ, 得到点 $\boldsymbol{\phi}_t$; 再沿 $\boldsymbol{\phi}_t$ 处的负梯度方向移动步长 α_t, 得到更新参数 $\boldsymbol{\theta}_{t+1}^{(\mathrm{NAG})}$. 在 NAG的这一轮更新过程中, $\boldsymbol{\phi}_t$ 的梯度方向与历史累积梯度方向的夹角为锐角, 亦可看作是向前"试探" 一步后并未发现前方梯度出现逆转, 故 $-\alpha_t \nabla f_{i_t}(\boldsymbol{\theta}_t - \rho \boldsymbol{v}_{t-1})$ 项起到了加强更新的效果. 而在 NAG 的第 $t+1$ 轮迭代中, 点 $\boldsymbol{\phi}_{t+1}$ 的梯度方向与历史累积梯度方向的夹角为钝角, 故 $-\alpha_{t+1} \times \nabla f_{i_{t+1}}(\boldsymbol{\theta}_{t+1} - \rho \boldsymbol{v}_t)$ 项起到了减弱更新的效果. 上述现象直观地解释了 "NAG 在坡度再次上升前能及时减速"这一结论.

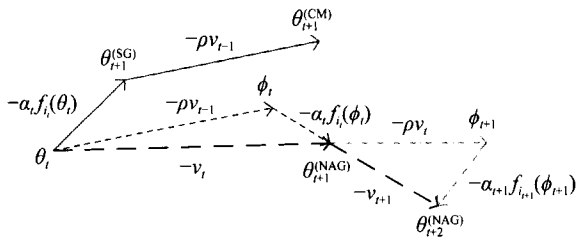


图 3　SGD、CM 和 NAG 的参数更新示意图

Fig. 3　Schematic diagram of parameters update of SGD, CM and NAG

CM 与 NAG 都是基于动量的随机梯度下降算法, 二者的区别在于: CM 只在更新参数时添加了动量项, 而 NAG 在更新参数和计算梯度时都应用了动量技巧. 表 1 比较了 CM 和 NAG 的更新式. 为了统一 CM 与 NAG 中的梯度计算公式, 表 1 也给出了 NAG 的第 2 种写法[39]. NAG 的写法 2 在计算第 t 轮梯度时不再需要上一轮动量 $\rho \boldsymbol{v}_{t-1}$, 这是因为在第 $t-1$ 轮更新参数时已经借助中间变量 $\bar{\boldsymbol{v}}_{t-1}$ 向目标参数施加了动量技巧.

表 1　CM 与 NAG 的更新公式比较

Table 1　Comparison of update formulas between CM and NAG

	CM	NAG (写法1)	NAG (写法2)
梯度	$\boldsymbol{g}_t = \nabla f_{i_t}(\boldsymbol{\theta}_t)$	$\boldsymbol{g}_t = \nabla f_{i_t}(\boldsymbol{\theta}_t - \rho \boldsymbol{v}_{t-1})$	$\boldsymbol{g}_t = \nabla f_{i_t}(\boldsymbol{\theta}_t)$
动量	$\boldsymbol{v}_t = \alpha_t \boldsymbol{g}_t + \rho \boldsymbol{v}_{t-1}$	$\boldsymbol{v}_t = \alpha_t \boldsymbol{g}_t + \rho \boldsymbol{v}_{t-1}$	$\boldsymbol{v}_t = \alpha_t \boldsymbol{g}_t + \rho \boldsymbol{v}_{t-1}$ $\bar{\boldsymbol{v}}_t = \alpha_t \boldsymbol{g}_t + \rho \boldsymbol{v}_t$
参数迭代	$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{v}_t$	$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{v}_t$	$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \bar{\boldsymbol{v}}_t$

3　基于方差缩减的随机梯度下降算法

在 SGD 中, 单个样本的梯度是全体样本平均梯度的无偏估计, 但梯度方差往往随着迭代次数的增加而不断累加, 这使得 SGD 无法保证能够达到线性收敛[49]. "方差缩减" 策略通过构造特殊的梯度估计量, 使得每轮梯度的方差有一个不断缩减的上界, 从而取得较快的收敛速度.

3.1　随机方差缩减梯度算法

随机方差缩减梯度算法 (SVRG) 提出了方差缩减算法的一般框架[28], 后续相继涌现的方差缩减算法大多在此版本上进行改进. 参数 $\boldsymbol{\theta}$ 每经过若干次更新就为它保留一个 "快照", 记为 $\tilde{\boldsymbol{\theta}}$. 例如, 初始化参数 $\boldsymbol{\theta}_0$ 经过 s 次随机迭代更新得到 $\boldsymbol{\theta}_s$, 则令 $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}_s$ (一般设置 $s = n$ 或 $s = 2n$). 计算 $\tilde{\boldsymbol{\theta}}$ 处全体样本的平均梯度 $\tilde{\boldsymbol{\mu}} = \dfrac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{\boldsymbol{\theta}})$, 则有 $\mathrm{E}[\nabla f_i(\tilde{\boldsymbol{\theta}}) - \tilde{\boldsymbol{\mu}}] = 0$. 将 $\tilde{\nabla}_t = \nabla f_{i_t}(\boldsymbol{\theta}_t) - \nabla f_{i_t}(\tilde{\boldsymbol{\theta}}) + \tilde{\boldsymbol{\mu}}$ 作为第 t 轮迭代中的梯度估计量, 可得 SVRG的参数更新式为

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \tilde{\nabla}_t \tag{13}$$

Johnson 等[28] 证明了 SVRG 梯度近似算子产生的方差随着迭代次数的增加不断缩减, 且小于 SGD 对应的方差, 即

$$\mathrm{E}\left[\left\| \tilde{\nabla}_t - \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\boldsymbol{\theta}_t) \right\|_2^2\right] < \mathrm{E}\left[\left\| \nabla f_{i_t}(\boldsymbol{\theta}_t) - \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\boldsymbol{\theta}_t) \right\|_2^2\right] \tag{14}$$

此外, 随机梯度下降算法通常使用递减的学习率, 而 SVRG 在固定学习率下便可取得较快的收敛速度.

3.2　近端随机方差缩减梯度算法

SVRG 能够加快收敛速度, 但只适用于光滑的目标函数. 对于添加了非光滑正则项的目标函数, 虽然可以通过计算次梯度来近似替代梯度, 但收敛速度较慢, 实际意义较小. 随机近端梯度下降算法 (Stochastic proximal gradient descent, SPGD) 通

过计算投影算子间接地估计目标参数, 从而巧妙地避开了正则项不光滑的问题[50-51]. 近端随机方差缩减梯度算法 (Prox-SVRG) 将 SVRG 与 SPGD 相结合, 为含非光滑正则项的目标函数使用方差缩减技巧提供了解决方案[29]. 考虑模型 (1) 的正则化版本

$$\min_{\boldsymbol{\theta}\in\mathbf{R}^d} J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta}) + h(\boldsymbol{\theta}) \tag{15}$$

Prox-SVRG 的目标参数更新式为

$$\boldsymbol{\theta}_{t+1} = \text{prox}_{\alpha_t}^{h}\left(\boldsymbol{\theta}_t - \alpha_t\widetilde{\nabla}_t\right) \tag{16}$$

其中, $\text{prox}_{\alpha_t}^{h}$ 为函数 h 关于超参数 α_t 的近端投影算子, 其计算式为

$$\text{prox}_{\alpha_t}^{h}(\boldsymbol{y}) = \arg\min_{\boldsymbol{\theta}\in\mathbf{R}^d}\left(h(\boldsymbol{\theta}) + \frac{1}{2\alpha_t}\|\boldsymbol{\theta}-\boldsymbol{y}\|_2^2\right) \tag{17}$$

Prox-SVRG 的算法思路可归结为以下三步. 首先, 暂不考虑正则项, 沿用 SVRG 的方差缩减技巧计算第 t 轮梯度的估计量 $\tilde{\nabla}_t$; 其次, 计算目标参数的近似点 $\hat{\boldsymbol{\theta}}_{t+1} = \boldsymbol{\theta}_t - \alpha_t\tilde{\nabla}_t$; 最后, 计算投影算子 $\text{prox}_{\alpha_t}^{h}(\hat{\boldsymbol{\theta}}_{t+1})$. 近端投影算子在保证目标参数 $\boldsymbol{\theta}_{t+1}$ 与近似估计点 $\hat{\boldsymbol{\theta}}_{t+1}$ 距离很近的前提下, 尽可能使 h 达到最小.

3.3 Katyusha 算法

经典动量和 Nesterov 动量的作用都是使目标参数获得更大程度地更新, 以加速逼近最优解. 但对于噪声较多的数据, 计算的梯度常常不准确, 故使用这两种动量算法有时会造成更大的偏差, 从而影响收敛速度. Allen-Zhu[30] 提出了一种消极动量–Katyusha 动量, 并将其与 SVRG 结合形成了 Katyusha 算法. 该算法的主要参数更新式为

$$\boldsymbol{q}_t = \boldsymbol{\theta}_t - \frac{1}{3L}\widetilde{\nabla}_t \tag{18}$$

$$\boldsymbol{z}_t = \boldsymbol{z}_{t-1} - \alpha\widetilde{\nabla}_t \tag{19}$$

$$\boldsymbol{\theta}_{t+1} = \tau_1\boldsymbol{z}_t + \tau_2\widetilde{\boldsymbol{\theta}} + (1-\tau_1-\tau_2)\boldsymbol{q}_t \tag{20}$$

其中, L 为 Lipschitz 常数, $\alpha = 1/(3\tau_1 L)$ 为固定步长, $\tau_1, \tau_2 \in [0,1]$ 为动量系数. 建议设置 $\tau_1 = \min\{\sqrt{n\sigma/L}, 0.5\}$, $\tau_2 = 0.5$, 此处 σ 为强凸系数. 在这种动量形式下, 目标参数 $\boldsymbol{\theta}$ 是 $\tilde{\boldsymbol{\theta}}$ 和中间变量 $\boldsymbol{z}_t$, $\boldsymbol{q}_t$ 三者的凸组合. 在每轮更新参数时, "快照" $\tilde{\boldsymbol{\theta}}$ 像 "磁铁" 一样将目标参数以 τ_2 的权重吸附在它周围, 以至于更新参数不会离 $\tilde{\boldsymbol{\theta}}$ 太远. 若令 $\tau_1 = \tau_2 = 0$, 则 Katyusha 变为了 SVRG. 故可将式 (20) 看作在 SVRG 更新参数后, 进一步对目标参数进行修正, 将其限制在一个合理的范围内.

3.4 MiG 算法

Katyusha 算法有效地提高了 SVRG 的收敛速度, 但由于在迭代过程中添加了两个中间变量, 因此算法结构比较复杂. 对于这类结构复杂的算法, 很难根据各种实际问题对其进行改进 (例如, 异步版本和稀疏版本), 且在多线程并发计算时难以保证收敛速度. MiG 算法是 SVRG 的另一个改进算法, 它仅引入一个中间变量, 其迭代过程简单明晰[31]. MiG 的参数更新式为

$$\boldsymbol{b}_t = \zeta\boldsymbol{\theta}_t + (1-\zeta)\widetilde{\boldsymbol{\theta}} \tag{21}$$

$$\widetilde{\nabla}_t = \nabla f_{i_t}(\boldsymbol{b}_t) - \nabla f_{i_t}(\widetilde{\boldsymbol{\theta}}) + \widetilde{\boldsymbol{\mu}} \tag{22}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t\widetilde{\nabla}_t \tag{23}$$

其中, 中间变量 $\boldsymbol{b}_t$ 是目标参数 $\boldsymbol{\theta}_t$ 与 "快照" $\tilde{\boldsymbol{\theta}}$ 的凸组合, 用于计算梯度估计量 $\tilde{\nabla}_t$; 参数 ζ 可按照以下规则进行设置:

$$\zeta = \begin{cases} \sqrt{\dfrac{n}{3\kappa}}, & \dfrac{n}{\kappa} \le \dfrac{3}{4} \\ \dfrac{1}{2}, & \dfrac{n}{\kappa} > \dfrac{3}{4} \end{cases} \tag{24}$$

MiG 算法结构精简且收敛速度快, 即使在非强凸条件下仍可实现 $O(1/t^2)$ 的收敛速度. 文献 [31] 进一步提出了 MiG 的稀疏改进算法和异步稀疏改进算法. 由于 ζ 与 Katyusha 动量形式中的 τ_2 作用相似, 因此 MiG 亦可看作是在 SVRG 的基础上添加了 Katyusha 动量. Nitanda[52] 提出的 Acc-Prox-SVRG 则是通过添加 Nesterov 动量来提升 Prox-SVRG 的算法性能. 将方差缩减技术与动量形式结合是目前提升随机梯度下降算法收敛速度的有效方法之一, 表 2 列出了几种方差缩减算法及使用的动量类型.

表 2 几种方差缩减算法的动量类型

Table 2 Momentum types of several variance reduced algorithms

算法	动量类型
SVRG[28], Prox-SVRG[29]	无
Katyusha[30]	Nesterov动量, Katyusha动量
MiG[31]	Katyusha动量
Acc-Prox-SVRG[52]	Nesterov动量

Shamir[13] 将方差缩减技术应用在主成分分析中, 并提出了方差缩减主成分分析 (Variance reduced principal component analysis, VR-PCA). Shang 等[53] 通过调整 SVRG 的关键点设置, 提出了适用于大步长的方差缩减随机梯度下降算法 (Vari-

ance reduced SGD, VR-SGD). 此外, 包含方差缩减思想的随机梯度下降算法还有随机对偶坐标上升 (Stochastic dual coordinate ascent, SDCA)[54] 以及随机原对偶坐标 (Stochastic primal-dual coordinate, SPDC)[55] 等.

4 基于增量梯度的随机梯度下降算法

"增量梯度" 策略源于增量聚合梯度算法 (Incremental aggregated gradient, IAG)[56], 该算法为每个样本保留一个相应的梯度值, 在迭代过程中, 依次抽取样本并用新梯度替代旧梯度.

4.1 随机平均梯度算法

随机平均梯度算法 (SAG)[32] 是 IAG 的随机版本. SAG 每轮随机抽取一个样本并计算梯度, 其他样本的梯度维持上一轮的值, 使用全体样本的平均梯度来更新目标参数. 分别对每个样本使用一次简单的梯度下降法或其他方法, 生成 n 个初始化参数 $\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_n$. 初始化聚合梯度 $\boldsymbol{d}_n=\sum_{i=1}^{n}\nabla f_i(\boldsymbol{\theta}_i)$ (注意: 此处的样本序号和参数下标是对应的). 当迭代次数 $t>n$ 时, SAG 的参数更新式为

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\frac{\alpha_t}{n}\boldsymbol{d}_t \tag{25}$$

$$\boldsymbol{d}_{t+1}=\boldsymbol{d}_t-\nabla f_{i_t}(\boldsymbol{\theta}_t)+\nabla f_{i_t}(\boldsymbol{\theta}_{t+1}) \tag{26}$$

SAG 的更新过程可看作先将聚合梯度中某个样本的旧梯度替换为新梯度, 再使用全局梯度估计量 $\boldsymbol{d}_t/n$ 更新目标参数[57]. SAG 的迭代成本与 SGD 相当, 但收敛速度却与 FGD 相同.

4.2 SAGA

SAGA 融合了 SAG 中 "以新梯度替换旧梯度" 的思想和 SVRG 中的方差缩减技巧[33]. 它先使用一次简单的梯度下降法或其他方法, 生成初始化参数 $\boldsymbol{\theta}_0$; 再引入中间变量 $\boldsymbol{\psi}^i$, 并初始化 $\boldsymbol{\psi}_0^i=\boldsymbol{\theta}_0$, 其中 $i=1,\cdots,n$. SAGA 的参数更新式为

$$\boldsymbol{\psi}_{t+1}^{i_t}=\boldsymbol{\theta}_t \tag{27}$$

$$\boldsymbol{w}_{t+1}=\boldsymbol{\theta}_t-\alpha_t\left(\nabla f_{i_t}\left(\boldsymbol{\psi}_{t+1}^{i_t}\right)-\nabla f_{i_t}\left(\boldsymbol{\psi}_t^{i_t}\right)+\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(\boldsymbol{\psi}_t^i\right)\right) \tag{28}$$

$$\boldsymbol{\theta}_{t+1}=\mathrm{prox}_{\alpha_t}^{h}(\boldsymbol{w}_{t+1}) \tag{29}$$

对于添加了非光滑正则项的目标函数, 可按照式 (17) 计算 $\boldsymbol{w}_{t+1}$ 的投算影子来更新目标参数.

虽然 SAGA 和 SAG 均采用了增量梯度的更新策略, 但二者仍有区别: SAG 先将聚合梯度中的某个样本的旧梯度替换为新梯度, 再将聚合梯度求平均; 而 SAGA 则是在聚合梯度求平均的基础上对某个样本的梯度进行更新. 显然, 计算单个样本梯度对 SAGA 的参数更新影响更大. 此外, SAGA 可看作 SVRG 的简化版本, 它将 SVRG 中的 "快照" $\tilde{\boldsymbol{\theta}}$ 替换为所选样本上一轮的中间变量 $\boldsymbol{\psi}_t^{i_t}$, 从而减少了计算量[58]. SAGA 在强凸条件下实现了较 SAG 和 SVRG 更快的线性收敛速度.

4.3 Point-SAGA

Defazio 将近端点算法[59] 融入到 SAGA 中, 提出了一种新的增量算法 Point-SAGA[34]. 与大多数随机梯度下降算法不同, Point-SAGA 不再根据梯度估计值来更新目标参数, 而是通过构造一个特殊的投影算子来完成参数更新. 它为第 i 个样本保留梯度 $\boldsymbol{g}^i$, 并初始化 $\boldsymbol{g}_0^i=\nabla f_i(\boldsymbol{\theta}_0)$, 其中, $i=1,\cdots,n$. Point-SAGA 的参数更新式为

$$\boldsymbol{z}_t^{i_t}=\boldsymbol{\theta}_t+\alpha_t\left(\boldsymbol{g}_t^{i_t}-\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{g}_t^i\right) \tag{30}$$

$$\boldsymbol{\theta}_{t+1}=\mathrm{prox}_{\alpha_t}^{f_{i_t}}\left(\boldsymbol{z}_t^{i_t}\right) \tag{31}$$

$$\boldsymbol{g}_{t+1}^i=\begin{cases}\dfrac{\boldsymbol{z}_t^{i_t}-\boldsymbol{\theta}_{t+1}}{\alpha_t}, & i=i_t\\ \boldsymbol{g}_t^i, & i\neq i_t\end{cases} \tag{32}$$

其中, $\boldsymbol{z}_t^{i_t}$ 为中间变量. 由式 (31) 知, $\boldsymbol{\theta}_{t+1}=\boldsymbol{z}_t^{i_t}-\alpha_t\nabla f_{i_t}(\boldsymbol{\theta}_{t+1})$, 它是关于 $\boldsymbol{\theta}_{t+1}$ 的隐式方程.

在更新 $\boldsymbol{\theta}_{t+1}$ 之前, Point-SAGA 就通过构造投影算子计算出随机样本在 $\boldsymbol{\theta}_{t+1}$ 处的近似梯度. 这一更新技巧使 Point-SAGA 在强凸条件下取得了比 SAGA 更快的收敛速度, 且在非光滑条件下依然适用. 在不考虑添加正则项的情况下, 可将 SVRG、SAG、SAGA 以及 Point-SAGA 的参数更新式进行统一处理[33], 如表 3 所示.

表 3　SVRG 与增量算法参数更新公式对比

Table 3　Comparison of parameters updating formulas among SVRG and incremental algorithms

算法名称	参数更新公式
SVRG	$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\alpha_t\left(\nabla f_{i_t}(\boldsymbol{\theta}_t)-\nabla f_{i_t}(\tilde{\boldsymbol{\theta}})+\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\tilde{\boldsymbol{\theta}})\right)$
SAG	$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\frac{\alpha_t}{n}\left(\nabla f_{i_t}(\boldsymbol{\theta}_t)-\nabla f_{i_t}\left(\boldsymbol{\psi}_t^{i_t}\right)+\sum_{i=1}^{n}\nabla f_i\left(\boldsymbol{\psi}_t^i\right)\right)$
SAGA	$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\alpha_t\left(\nabla f_{i_t}(\boldsymbol{\theta}_t)-\nabla f_{i_t}\left(\boldsymbol{\psi}_t^{i_t}\right)+\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(\boldsymbol{\psi}_t^i\right)\right)$
Point-SAGA	$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\alpha_t\left(\nabla f_{i_t}(\boldsymbol{\theta}_{t+1})-\nabla f_{i_t}\left(\boldsymbol{\psi}_t^{i_t}\right)+\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(\boldsymbol{\psi}_t^i\right)\right)$

对于前述几种非自适应学习率的随机梯度下降算法, Allen-Zhu[30] 根据各算法在强凸条件下的复杂度将其分为 3 代. 第 1 代算法包括 SGD 和 NAG 等; 第 2 代包括 SVRG 和 Prox-SVRG 等; 第 3 代包括 Katyusha 和 Point-SAGA 等. 其中, 第 1 代算法的复杂度最高. 在多数情形下, 第 3 代算法的复杂度优于第 2 代算法的复杂度, 当 $n < \kappa$ 时优势尤其显著. 对于动量、方差缩减和增量梯度策略下的随机梯度下降算法, 表 4 比较了各算法取得 $\varepsilon-$ 近似解的复杂度和收敛速度, 其中, 常数 $\rho \in (0,1)$, $T = t/n$ 表示全局迭代次数.

5 自适应调节学习率的随机梯度下降算法

在大规模机器学习中, 为了减少振荡、提高优化效率, 每经过数轮迭代就需要更换一个较小的学习率. 手动调节学习率工作量较大且很难快速找到当前模型环境下的最佳值. 若设置的学习率过小, 会使得优化进程缓慢; 若学习率过大, 会导致振荡且难以逼近最优解甚至逐渐远离最优解, 如图 4 所示. 为了解决此问题, 一些学者提出了以下几种自适应调节学习率的随机梯度下降算法, 这些算法在深度神经网络中表现出了极佳性能.

5.1 Adagrad

对于数据特征不平衡的问题, 若使用 SGD, 则稀疏特征对应的梯度分量的绝对值很小甚至为零, 这使得目标参数难以逼近最优解. Adagrad[35] 是一种自适应调节学习率的随机梯度下降算法, 它将目标参数 $\boldsymbol{\theta}$ 的分量进行拆分, 对每个分量使用不同的学习率进行更新. 对稀疏特征相应的参数分量使用较大的学习率进行更新, 进而识别出那些非常具有预测价值但易被忽略的特征.

为了描述方便, 引入以下符号: $\theta_{t,k}$ 表示第 t 轮迭代时 $\boldsymbol{\theta}$ 的第 k 个参数分量, 其中 $k \in \{1,2,\cdots,d\}$; $g_t \in \mathbf{R}^d$ 表示第 t 轮迭代时的梯度, 即 $\boldsymbol{g}_t = \nabla f_{i_t}(\boldsymbol{\theta}_t)$; $g_{t,k}$ 表示 $\theta_{t,k}$ 对应的梯度分量, 即 $g_{t,k} = \nabla f_{i_t}(\theta_{t,k})$. Adagrad 的参数分量更新式为

$$\theta_{t+1,k} = \theta_{t,k} - \frac{\alpha}{\sqrt{\hat{g}_{t,k} + \varepsilon}} g_{t,k} \tag{33}$$

其中, $\hat{g}_{t,k} = \sum_{q=1}^{t} g_{q,k}^2$ 表示对前 t 轮的第 k 个梯度分量进行平方和累加; α 是全局学习率; ε 是一个很小的正数 (一般取 10^{-8}), 其作用是避免分母为 0; 可将 $\alpha/\sqrt{\hat{g}_{t,k} + \varepsilon}$ 看作自适应调节的学习率.

Adagrad 对不同的参数分量使用不同的学习率. 对于数据稀疏程度较弱的特征, 分母中历史梯度的累积值较大, 从而学习率较小; 对于数据稀疏程度较强的特征, 分母中历史梯度的累积值较小, 故学习率较大, 这种方法在处理稀疏数据时起到了捕捉罕见特征的作用. 将单个元素向量化, Adagrad 的参数更新式可写为

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\alpha}{\sqrt{\boldsymbol{G} + \varepsilon}} \odot \boldsymbol{g}_t \tag{34}$$

其中, $\boldsymbol{G} = (\hat{g}_{t,1}, \cdots, \hat{g}_{t,d})^{\mathrm{T}}$, $1/\sqrt{\boldsymbol{G}+\varepsilon} = (1/\sqrt{\hat{g}_{t,1}+\varepsilon}, \cdots, 1/\sqrt{\hat{g}_{t,d}+\varepsilon})^{\mathrm{T}}$, $\odot$ 表示向量的对应元素相乘. 图 5 绘出了 Adagrad 在 Abalone 数据集上训练 Lasso 模型时, 目标参数的 8 个分量对应的学习率的变化曲线. 从此图可以看出, 虽然学习率整体呈缓慢下降趋势, 但各分量的变化程度不同.

Dean 等[60] 发现 Adagrad 能显著提高 SGD 的鲁棒性, 并将其用于谷歌的大型神经网络训练; Pennington等[61] 在训练词嵌入技术时使用了 Adagrad, 这是因为不常用的词比常用的词需要更大的更新. Chen 等[62] 提出了 SAdagrad, 它将 Adagrad 看作一个子程序并周期性调用, 在强凸条件下具有更好的收敛速度. Adagrad 的提出使人们摆脱了手动调节

表 4 随机梯度下降算法的复杂度与收敛速度

Table 4 Complexity and convergence speed of stochastic gradient descent algorithms

	算法名称	复杂度 (强凸)	复杂度 (凸)	收敛速度 (强凸)	收敛速度 (凸)
第1代	SGD[43]	$O((\kappa/\varepsilon)\log\frac{1}{\varepsilon})$	$O(1/\varepsilon^2)$	$O(1/T)$	$O(1/\sqrt{T})$
	NAG[26]	$O(n\sqrt{\kappa}\log\frac{1}{\varepsilon})$	—	$O(\rho^T)$	$O(1/T^2)$
第2代	SVRG[28]	$O((n+\kappa)\log\frac{1}{\varepsilon})$	—	$O(\rho^T)$	—
	Prox-SVRG[29]	$O((n+\kappa)\log\frac{1}{\varepsilon})$	—	$O(\rho^T)$	—
	SAG[32]	$O((n+\kappa)\log\frac{1}{\varepsilon})$	—	$O(\rho^T)$	$O(1/T)$
	SAGA[33]	$O((n+\kappa)\log\frac{1}{\varepsilon})$	$O(n/\varepsilon)$	$O(\rho^T)$	$O(1/T)$
第3代	Katyusha[30]	$O((n+\sqrt{n\kappa})\log\frac{1}{\varepsilon})$	$O(n+\sqrt{n/\varepsilon})$	$O(\rho^T)$	$O(1/T^2)$
	Point-SAGA[34]	$O((n+\sqrt{n\kappa})\log\frac{1}{\varepsilon})$	—	$O(\rho^T)$	—
	MiG[31]	$O((n+\sqrt{n\kappa})\log\frac{1}{\varepsilon})$	$O(n+\sqrt{n/\varepsilon})$	$O(\rho^T)$	$O(1/T^2)$

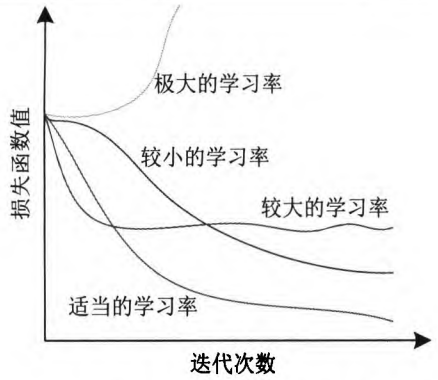


图 4　学习率对优化过程的影响

Fig. 4　Effect of learning rates on optimization process

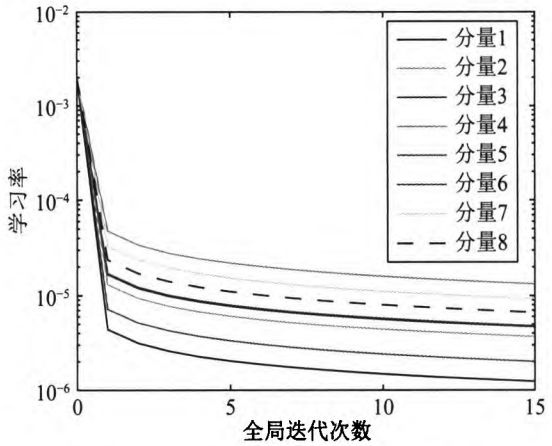


图 5　Adagrad 的学习率变化示意图

Fig. 5　Schematic diagram of learning rate changes for adagrad

学习率的困扰, 但仍存在以下缺点: 1) 历史梯度无节制地累加导致了学习率不断衰减至一个极小的数, 这使得训练后期优化效率很低; 2) 需要预先设置全局学习率; 3) 保留梯度的内存成本过大.

5.2　Adadelta

Adadelta[37] 是 Adagrad 的改进版本, 它不再低效率地存储历史梯度的平方和, 而是通过引入一个新的统计量 "衰减平均"递归地计算平方梯度的指数加权滑动平均, 从而大大降低了存储成本. 每轮平方梯度的衰减平均只与上一轮平方梯度的衰减平均和本轮平方梯度有关, 计算式为

$$\mathrm{E}\left[\boldsymbol{g}^2\right]_t = \gamma \mathrm{E}\left[\boldsymbol{g}^2\right]_{t-1} + (1-\gamma)\boldsymbol{g}_t^2 \tag{35}$$

其中, $\boldsymbol{g}_t^2 = \boldsymbol{g}_t \odot \boldsymbol{g}_t$, $\mathrm{E}[\boldsymbol{g}^2]_t$ 表示第 t 轮平方梯度的衰减期望值 ($\mathrm{E}[\boldsymbol{g}^2]_0$可初始化为 d 维零向量), $\gamma \in (0,1)$ 为衰减常数 (一般取 0.9), 与动量方法中的动量系数类似.

将 Adagrad 参数更新式中的 $\boldsymbol{G}$ 替换为平方梯度的衰减期望值 $\mathrm{E}[\boldsymbol{g}^2]_t$, 可得

$$\Delta\boldsymbol{\theta}_t = -\frac{\alpha}{\sqrt{\mathrm{E}\left[\boldsymbol{g}^2\right]_t + \varepsilon}} \odot \boldsymbol{g}_t \tag{36}$$

式 (36) 亦为 Rmsprop[63] 的参数更新式. 将式 (36) 中的分母用梯度的均方根误差 (Root mean square error, RMSE) 近似代替, 则有

$$\Delta\boldsymbol{\theta}_t = -\frac{\alpha}{RMSE[\boldsymbol{g}]_t} \odot \boldsymbol{g}_t \tag{37}$$

Zeiler 等[37] 认为此更新式中分子与分母的单位不一致, 故可使用上一轮参数的均方根误差 $RMSE[\Delta\boldsymbol{\theta}]_{t-1} = \sqrt{\mathrm{E}[(\Delta\boldsymbol{\theta})^2]_{t-1} + \varepsilon}$ 来替换 α. 于是 Adadelta 的参数更新式为

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{RMSE[\Delta\boldsymbol{\theta}]_{t-1}}{RMSE[\boldsymbol{g}]_t} \odot \boldsymbol{g}_t \tag{38}$$

其中, $RMSE[\Delta\boldsymbol{\theta}]_{t-1}/RMSE[\boldsymbol{g}]_t$ 可看作自适应调节的学习率, "/" 表示向量的对应元素之商.

Adadelta 针对 Adagrad 的三个缺点进行了改进. 1) 更新式 (38) 中的分母不再是一个无限累加的数, 衰减常数 γ 通过调整历史梯度与当前梯度的比例, 使学习率的分母保持在一定的范围内; 2) 此方法无需事先设置全局学习率; 3) 无需存储历史梯度, 故大大节省了内存成本.

5.3　Adam

Adam 结合了矩估计思想, 通过计算并修正每轮梯度的一阶矩和二阶矩来动态调节学习率[38]. 在使用 Adam 时, 参数改变量 $\Delta\boldsymbol{\theta}$ 近似受到全局学习率 α 的限制且$|\Delta\theta_{t,k}| <\approx \alpha$, 其中, $<\approx$ 表示小于或约等于, $\theta_{t,k}$ 是 $\Delta\boldsymbol{\theta}_t$ 第 k 个分量. 在实际应用中, 根据 α 的数量级可以大致推测出参数逼近最优解所需的更新次数; 也可以将 α 看作$|\Delta\theta_{t,k}|$的置信区间的上限, 当某一轮参数变化量远大于 α 时, 可以判断该样本为噪声点或此轮更新不具有价值[64]. 分别计算 $\boldsymbol{g}_t$ 的一阶矩和二阶矩的估计量 $\boldsymbol{m}_t$ 和 $\boldsymbol{u}_t$

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \tag{39}$$

$$\boldsymbol{u}_t = \beta_2 \boldsymbol{u}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2 \tag{40}$$

其中, $\beta_1, \beta_2 \in [0,1)$是衰减常数. 建议设置为 $\beta_1 = 0.9$, $\beta_2 = 0.999$, 初始化 $\boldsymbol{m}_0$ 和 $\boldsymbol{u}_0$ 均为 d 维零向量. 分别对 $\boldsymbol{m}_t$ 和 $\boldsymbol{u}_t$ 的偏差进行修正

$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t}, \qquad \hat{\boldsymbol{u}}_t = \frac{\boldsymbol{u}_t}{1-\beta_2^t} \tag{41}$$

Adam 的参数更新式为

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\alpha}{\sqrt{\hat{\boldsymbol{u}}_t} + \varepsilon} \odot \hat{\boldsymbol{m}}_t \tag{42}$$

Adam 的迭代式与 Adadelta 中的式 (36) 的形

式相似, 即将式 (36) 中的梯度改为修正后的一阶矩, 平方梯度的衰减平均值改为修正后的二阶矩. 称 $\hat{\boldsymbol{m}}_t/\sqrt{\hat{\boldsymbol{u}}_t}$ 为信噪比 (Signal-noise ratio, SNR), 随着 SNR 不断减小, $\Delta\theta_{t,k}$ 不断接近于 0, 这个性质亦是一种自动退火形式[65]. Adam 结合了 Adagrad 善于处理稀疏梯度和 Rmsprop 善于处理非平稳目标的优点, 适用于大数据集和高维空间.

5.4 AdaMax

在 Adam 的参数更新式中, 当 t 较大时, $\sqrt{\hat{\boldsymbol{u}}_t}+\epsilon \approx (\beta_2\boldsymbol{u}_{t-1}+(1-\beta_2)\boldsymbol{g}_t^2)^{1/2}$ 可近似看作由历史梯度组成的向量 $\boldsymbol{g}^{(t)}=(\boldsymbol{g}_1^{\mathrm{T}},\cdots,\boldsymbol{g}_t^{\mathrm{T}})^{\mathrm{T}}$ 的缩放 ℓ_2 范数, 故参数改变量 $|\Delta\theta_{t,k}|$ 与 $\|\boldsymbol{g}^{(t)}\|_2$ 近似成反比. AdaMax 将 Adam 中的 ℓ_2 范数拓展到了 ℓ_∞ 范数, 使得全局学习率 α 对参数改变量 $\Delta\boldsymbol{\theta}$ 的限制范围更加清晰, 即 $|\Delta\theta_{t,k}|\leq\alpha$ [38].

先将 ℓ_2 范数拓展到 ℓ_p 范数, 重新定义滑动均值

$$\boldsymbol{u}_t=\beta_2^p\boldsymbol{u}_{t-1}+(1-\beta_2^p)\,|\boldsymbol{g}_t|^p=(1-\beta_2^p)\sum_{i=1}^{t}\beta_2^{p(t-i)}\times|\boldsymbol{g}_i|^p \tag{43}$$

其中, β_2^p 为衰减常数, $|\boldsymbol{g}_t|^p=(|g_{t,1}|^p,\cdots,|g_{t,d}|^p)^{\mathrm{T}}$. 再将 ℓ_p 范数拓展到 ℓ_∞ 范数, 得

$$\begin{aligned}\boldsymbol{U}_t=&\lim_{p\to\infty}(\boldsymbol{u}_t)^{\frac{1}{p}}=\\&\lim_{p\to\infty}(1-\beta_2^p)^{\frac{1}{p}}\left(\sum_{i=1}^{t}\beta_2^{p(t-i)}\times|\boldsymbol{g}_i|^p\right)^{\frac{1}{p}}=\\&\lim_{p\to\infty}\left(\sum_{i=1}^{t}\left(\beta_2^{(t-i)}\times|\boldsymbol{g}_i|\right)^p\right)^{\frac{1}{p}}=\\&\max\{\beta_2^{t-1}|\boldsymbol{g}_1|,\beta_2^{t-2}|\boldsymbol{g}_2|,\cdots,|\boldsymbol{g}_t|\}\end{aligned} \tag{44}$$

式 (44) 等价于一个简单的递归公式

$$\boldsymbol{U}_t=\max\{\beta_2\boldsymbol{U}_{t-1},|\boldsymbol{g}_t|\} \tag{45}$$

可初始化 $\boldsymbol{U}_0$ 为 d 维零向量. 用 $\boldsymbol{U}_t$ 替换 Adam 参数更新式中的 $\sqrt{\hat{\boldsymbol{u}}_t}+\epsilon$, 得到 AdaMax 的参数更新式

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\frac{\eta}{\boldsymbol{U}_t}\odot\boldsymbol{m}_t \tag{46}$$

其中, η 为全局学习率, 建议与 Adam 中的全局学习率 α 按照关系 $\eta=\alpha/(1-\beta_1^t)$ 进行设置[66]. 值得注意的是, AdaMax 无需对梯度的一阶矩 $\boldsymbol{m}_t$ 进行修正.

5.5 Nadam

Nadam 将 Nesterov 动量加入到 Adam 中, 以提升 Adam 的算法性能, 亦可看作是 Adam 和 NAG 的结合[39]. 从表 1 可以看出: 要将 CM 转化为 NAG, 需要添加中间变量 $\bar{\boldsymbol{v}}$. 只要统一 Adam 与 CM 的更新形式, 便可使用这种修改技巧实现从 Adam 向 Nadam 的转换.

暂不考虑偏差修正, Adam 的更新式可写为

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\frac{\alpha}{\sqrt{\boldsymbol{u}_t}+\varepsilon}\odot\boldsymbol{m}_t \tag{47}$$

因为 $\beta_2\approx1$, 所以 $\boldsymbol{u}_t\approx\boldsymbol{u}_{t-1}$.

将式 (39) 和式 (40) 代入式 (47), 得到

$$\begin{aligned}\boldsymbol{\theta}_{t+1}=&\boldsymbol{\theta}_t-\left(\frac{\alpha(1-\beta_1)}{\sqrt{\beta_2\boldsymbol{u}_{t-1}+(1-\beta_2)\boldsymbol{g}_t^2}+\varepsilon}\odot\boldsymbol{g}_t+\right.\\&\left.\frac{\alpha\beta_1}{\sqrt{\beta_2\boldsymbol{u}_{t-1}+(1-\beta_2)\boldsymbol{g}_t^2}+\varepsilon}\odot\boldsymbol{m}_{t-1}\right)\approx\\&\boldsymbol{\theta}_t-\left(\frac{\alpha(1-\beta_1)}{\sqrt{\beta_2\boldsymbol{u}_{t-1}+(1-\beta_2)\boldsymbol{g}_t^2}+\varepsilon}\odot\boldsymbol{g}_t+\right.\\&\left.\frac{\alpha\beta_1}{\sqrt{\boldsymbol{u}_{t-1}}+\varepsilon}\odot\boldsymbol{m}_{t-1}\right)\end{aligned} \tag{48}$$

式 (48) 与 CM 的更新形式相同, 括号中的第 2 项可看作动量项, 其中 $\alpha\beta_1/(\sqrt{\boldsymbol{u}_{t-1}}+\epsilon)$ 为动量系数. 对 Adam 使用表 1 中的动量转换技巧, 通过引入中间变量 $\tilde{\boldsymbol{m}}_t=\beta_1\boldsymbol{m}_t+(1-\beta_1)\boldsymbol{g}_t$, 得到参数更新式

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\frac{\alpha}{\sqrt{\boldsymbol{u}_t}+\varepsilon}\odot\widetilde{\boldsymbol{m}}_t \tag{49}$$

根据 Adam 的式 (41), 分别对 $\boldsymbol{m}_t$ 和 $\boldsymbol{u}_t$ 进行偏差修正得到 $\hat{\boldsymbol{m}}_t$ 与 $\hat{\boldsymbol{u}}_t$. 鉴于梯度与动量的作用对象不同步, 对 $\boldsymbol{g}_t$ 进行偏差修正得到 $\hat{\boldsymbol{g}}_t=\boldsymbol{g}_t/(1-\beta_1^t)$. 最终 Nadam 的参数更新式为

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\frac{\alpha}{\sqrt{\hat{\boldsymbol{u}}_t}+\varepsilon}\odot\overline{\boldsymbol{m}}_t \tag{50}$$

其中, $\overline{\boldsymbol{m}}_t=\beta_1\hat{\boldsymbol{m}}_t+(1-\beta_1)\hat{\boldsymbol{g}}_t$.

随着深度学习网络模型越来越复杂, 自适应算法面临着泛化性能差、学习率逐渐极端化等问题. 在最近的几个自然语言处理和计算机视觉项目中, 自适应算法仅在训练初期的优化效率较高, 而在训练后期和测试集上常常出现停滞不前的情况, 且整体效果不及 SGD 与 NAG[67-68]. Reddi 等[69] 认为可以通过对过去梯度的 "长期记忆" 来解决自适应算法收敛性差的问题, 并提出 Adam 的改进算法 AmSGDrad, 此算法有效地提升了收敛速度. Luo 等[70] 提出的自适应算法 Adabound 为学习率设置了一个动态上界, 且初始化学习率上界为无穷大, 随着迭代次数的增加, 学习率最终平稳地递减至一个恒定的值. Adabound在训练初期的优化速度与 Adam 相当,

在训练后期性能稳定且泛化能力较强, 能够在复杂的深层网络中发挥良好的性能.

6　数值实验

先使用动量、方差缩减和增量梯度 3 种策略下的随机梯度下降算法解决逻辑回归等机器学习任务, 再将自适应学习率的随机梯度下降算法应用到深度卷积神经网络中.

6.1　非自适应学习率算法的性能对比

本节实验对非自适应学习率的随机梯度下降算法在机器学习模型中的性能进行对比. 采用 MATLAB 与 C 语言混合编程, 涉及的算法程序均在 4 核 AMD A10-7300 Radeon R6 处理器上运行. 实验所用的 Adult 等 4 个数据集均来自于 UCI 机器学习库 (http://archive.ics.uci.edu/ml/datasets.html). 表 5 给出了这些数据集的详细信息.

表 5　数据集描述

Table 5　Description of datasets

数据集	样本总数	样本特征	占用内存
Adult	32562	123	734 KB
Covtype	581012	54	52 MB
Abalone	4177	8	71 KB
MNIST	60000	784	12 MB

建立 ℓ_2 逻辑回归 (ℓ_2-Logistic regression)、岭回归 (Ridge regression) 和 Lasso 等 3 种经典机器学习模型. 表 6 列出了 3 种优化模型的公式, 其中 $\|\cdot\|_1$ 为向量的 ℓ_1 范数, λ_1 和 λ_2 为取正值的正则化系数. 显然, 前两种是光滑且强凸的优化模型, 第 3 种是非光滑凸的优化模型. 表 7 列出了 4 个实验数据集对应的优化模型及正则参数设置. 选取 SGD、NAG、SVRG、SAGA、Katyusha 以及 MiG共 6 种具有代表性的方法进行实验, 通过研究 6 种算法的迭代效率和时间效率, 对比、分析算法的实际性能. 为公平起见, 初始化目标参数为零向量, 每种方法的超参数和学习率均先按照理论最优值进行设置, 再结合具体实验进行微调, 以使算法的实际性能达到最佳.

图 6 比较了 6 种随机梯度下降算法的迭代效率, 其中, 横坐标表示全局迭代次数, 纵坐标表示当前目标函数值与最优目标函数值之差. 从图 6 可以看出, 第 1 代随机梯度下降算法 SGD 和 NAG 的迭代效率较低, 难以在较少的迭代次数内逼近最优解, 这可能是因为梯度的方差不断累积且实际数据集存在噪声. 作为简单版本的随机梯度下降算法, SGD

表 6　3 种机器学习任务的优化模型

Table 6　Optimization models for 3 machine learning tasks

模型名称	模型公式
ℓ_2 逻辑回归	$\min\limits_{\boldsymbol{\theta}\in\mathbf{R}^d} J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\log\left(1+\exp\left(-y_i\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\theta}\right)\right)+\frac{\lambda_2}{2}\|\boldsymbol{\theta}\|_2^2$
岭回归	$\min\limits_{\boldsymbol{\theta}\in\mathbf{R}^d} J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\theta}-y_i\right)^2+\frac{\lambda_2}{2}\|\boldsymbol{\theta}\|_2^2$
Lasso	$\min\limits_{\boldsymbol{\theta}\in\mathbf{R}^d} J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\theta}-y_i\right)^2+\lambda_1\|\boldsymbol{\theta}\|_1$

表 7　数据集对应的优化模型与正则参数

Table 7　Optimization model and regularization parameter for each dataset

编号	数据集	优化模型	正则参数
(a)	Adult	ℓ_2 逻辑回归	$\lambda_2 = 10^{-5}$
(b)	Covtype	ℓ_2 逻辑回归	$\lambda_2 = 10^{-6}$
(c)	Abalone	Lasso	$\lambda_1 = 10^{-4}$
(d)	MNIST	岭回归	$\lambda_2 = 10^{-5}$

迭代效率最低; NAG 添加了 Nesterov 动量, 尽管在一定程度上提升了 SGD 的优化效率, 但效果并不显著. 第 2 代随机梯度下降算法的迭代效率在第 1 代的基础上产生了质的飞跃. SVRG 和 SAGA 分别通过采用方差缩减与增量梯度的策略构造相应的梯度估计量, 有效限制了方差的累积, 极大地提升了算法性能, 且能够在较少的迭代次数内逼近最优解. 第 3 代随机梯度下降算法 Katyusha 和 MiG 在方差缩减技术的基础上添加了消极动量, 因此在目标参数的优化过程中更加稳健、精准、高效, 能够在极少的迭代次数内逼近最优解.

图 7 对比了 6 种随机梯度下降算法的时间效率, 其中: 横坐标表示运行时间, 纵坐标表示当前目标函数值与最优目标函数值之差. 从图 7 可以看出: 第 2 代和第 3 代随机梯度下降算法的时间效率明显强于第 1 代, 这与图 6 中迭代效率的对比情况相似. 此外, 第 3 代算法的整体时间效率虽然稍强于第 2 代算法, 但其优势并不像迭代效率那样显著. 第 3 代算法虽然在复杂度和收敛速度上都有明显优势, 但由于自身结构相对复杂, 故在实际应用中单次迭代的计算量较大、运行时间较长.

6.2　自适应学习率算法的性能对比

本节实验对自适应学习率的随机梯度下降算法在深度学习中的性能进行对比. 实验环境为 MxNet-gluon 1.0, 工作站配置了 10 核 Intel Xeon E5-2640v4 处理器和两块 GTX 1080ti 11 GB 显卡. 使用的 CIFAR-10 数据集 (http://www.cs.toronto.edu/~kriz/cifar.html) 包含 60 000 幅 32 × 32 × 3 的彩色图

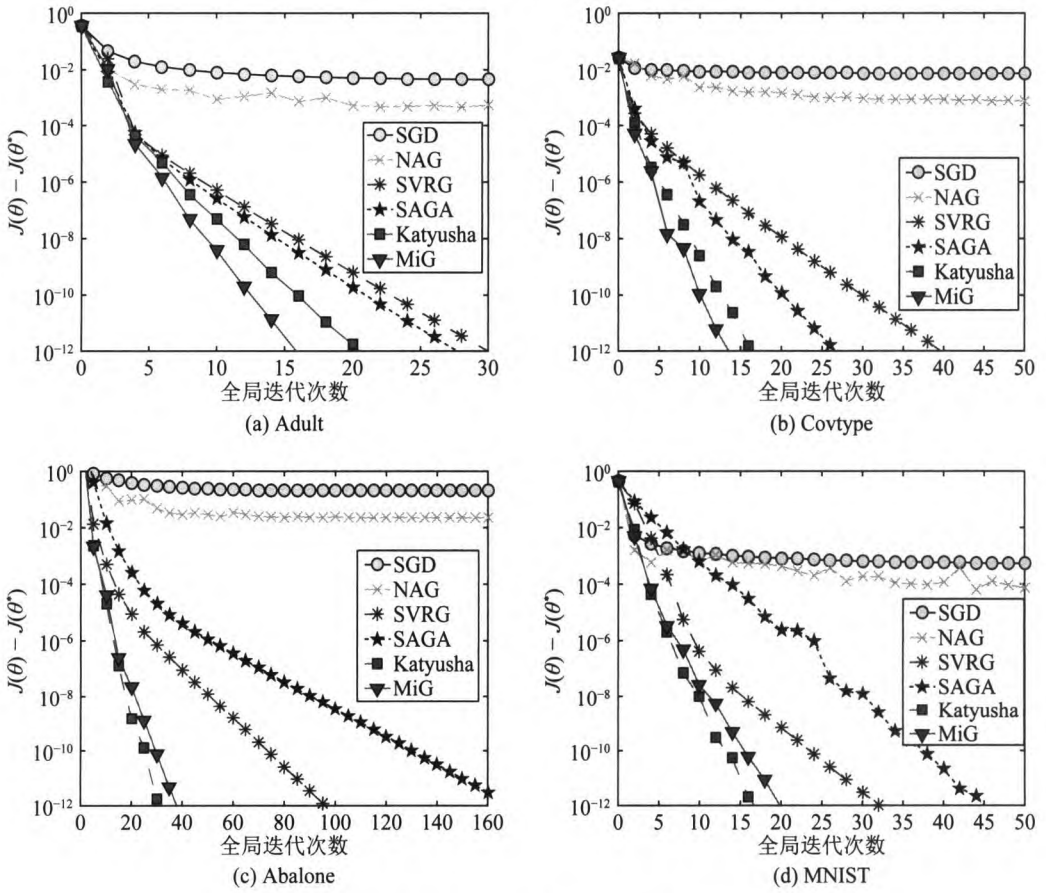


图 6 随机梯度下降算法的迭代效率对比

Fig. 6 Comparison of iterative efficiency of stochastic gradient descent algorithms

像. 根据图像内容可将其分为 "飞机"、"鸟" 和 "猫" 等 10 个类别, 其中每个类别均包含6 000 幅图像. 从每类图像中随机选取 5 000 幅作为训练样本, 剩余的 1 000 幅作为测试样本, 并在实验过程中采取数据增广策略.

在 ResNet-18 卷积神经网络模型[71] 下, 对 Adagrad、Rmsprop、Adadelta 以及 Adam 的实际性能进行比较. 此外, 实验还考虑了 SGD 和 CM 两种非自适应算法. ResNet-18 网络模型的权重参数按均值为 0、标准差为 0.01 的正态分布随机初始化. 训练时采用交叉熵损失函数, 批容量为 32, 除 Adadelta 外, 其余算法全局学习率的初始值均为 10^{-3}, 且每隔 30 代衰减至之前的 1/10, 共训练 90 代.

图 8 展示了实验中训练集损失函数值、训练集精度以及测试集精度的变化情况. 从此图可以看出, 4 种自适应学习率的梯度下降算法在实验中的性能整体优于 SGD. Adagrad 的性能最差, 仅略优于 SGD, 这可能是因为 Adagrad 生成的自适应学习率无节制减小, 从而导致后期学习率微小, 以至于无法突破局部最优点. Rmsprop 与 Adadelta 的性能比较接近, 且优于 Adagrad. Adam 在训练集与测试集中的优化效率都是最高的, 这说明矩估计思想和自动退火形式在随机梯度下降算法中起到了提升算法性能的作用. 此外, CM 在训练集中的表现与 Rmsprop相当, 在测试集中的性能甚至略优于 Rmsprop.

7 总结与展望

本文对近年来随机梯度下降算法的研究进展及主要研究成果进行了综述. 根据算法的更新策略, 将几种具有代表性的随机梯度下降算法分为四类, 包括基于动量的算法、基于方差缩减的算法、基于增量梯度的算法以及自适应学习率的算法. 本文介绍了这些算法的原理、核心思想以及相互之间的区别与联系, 并通过数值实验对算法的实际性能进行了对比. 实验结果表明: 在逻辑回归等经典机器学习任务中, Katyusha 和 MiG 等第 3 代算法普遍具有较好的性能, 而 SGD 和 NAG 等第 1 代算法的实验性能最差; 在深度卷积神经网络中, Adam 的实验性能最好. 当前, 国内外提出的随机梯度下降算法种类繁多, 但理论不完善且评价标准尚未统一. 因此, 随机梯度下降算法仍将是未来的研究热点.

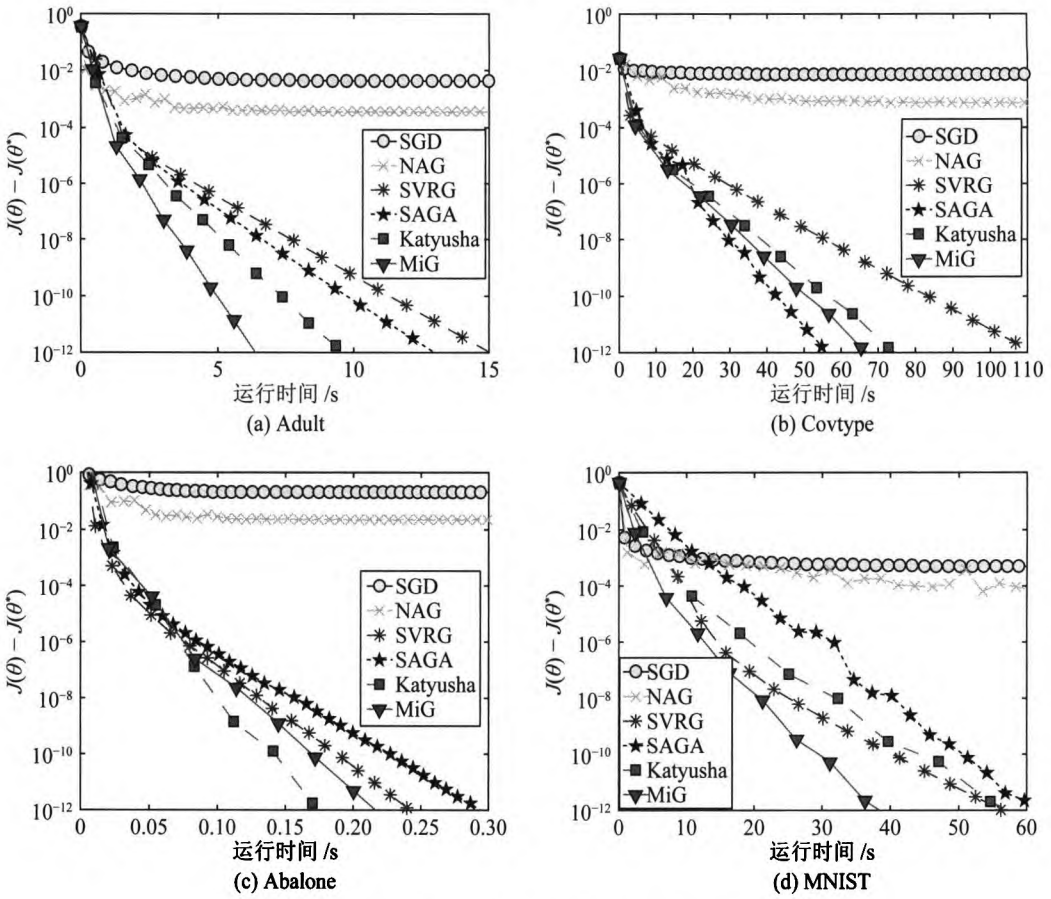


图 7　随机梯度下降算法的时间效率对比

Fig. 7　Comparison of time efficiency of stochastic gradient descent algorithms

下面几个方向在今后的研究中值得关注.

1) 与二阶算法相比, 随机梯度下降算法的收敛速度相对较慢, 且需要更多的迭代次数. 第 2 代和第 3 代改进算法虽然有效地提升了收敛速度, 但耗费了较大的时间成本和内存成本. 随着数据规模的扩大和模型复杂度的提升, 单线程下的随机梯度下降算法已经不能满足大规模机器学习应用的需求[72-73]. Zinkevich 等[74] 提出了首个并行式随机梯度下降算法 SimuParallelSGD, 这种方法适合于大规模学习范式和 MapReduce 框架. 文献 [75] 提出了 Hogwild! 算法, 它对稀疏数据采用无锁异步更新策略, 从而有效地减少了特征更新时的冲突. 陈振宏等[76] 提出了一种基于差异合并的分布式随机梯度下降算法 DA-DSGD, 从性能的加权合并、模型规范化两方面提高了算法性能. 目前, 研究者们已经实现了 SVRG、SAGA 和 MiG 等改进算法的分布式和并行化版本, 但收敛速度却有待进一步提升[31, 77-78]. 如何根据算法特点、数据对象和应用平台, 设计并实现不同改进策略下的随机梯度下降算法的分布式与并行化版本, 使其在实际应用中发挥出较高的性能水平, 这是未来值得探索的问题.

2) 学习率是随机梯度下降算法中一个非常重要的超参数, 直接影响算法在实际应用中的性能. 一些学者按照 $\alpha_t = c_1/(t^v + c_2)$ 的形式对每轮学习率进行设置, 但这种形式在实际应用中收敛速度较慢, 且 c_1, c_2 和 v 的取值仍难以确定[79]. 另一些学者则认为应先在固定步长下快速寻找最优解的邻域, 再考虑更为精确的优化方案[80]. 目前, 寻找最优学习率在理论和实践上仍是一个巨大的挑战.

3) 随机梯度下降算法在每轮迭代过程中计算复杂度较低, 但只利用了一阶梯度, 忽略了目标函数的二阶信息及曲率, 从而限制了实际性能和收敛速度[81]. 如何结合一阶与二阶方法各自的长处, 进一步设计迭代效率俱佳的随机梯度下降算法, 是未来值得研究的问题.

4) 近年来, 研究者们将目光投向非凸的 ERM 模型, 并且提出了一些行之有效的解决方案[82-85], 其中具有代表性的策略包括添加动量跳出局部最优解[86]、使用方差缩减技术减少梯度方差[87] 和添加梯度噪声逃离鞍点[88] 等. 然而, 对于更为一般的非凸、非光滑的优化问题却并未取得太大的突破, 目前仅有 Prox-SAGA[89]、Prox-SVRG+[90] 等算法, 但性能并

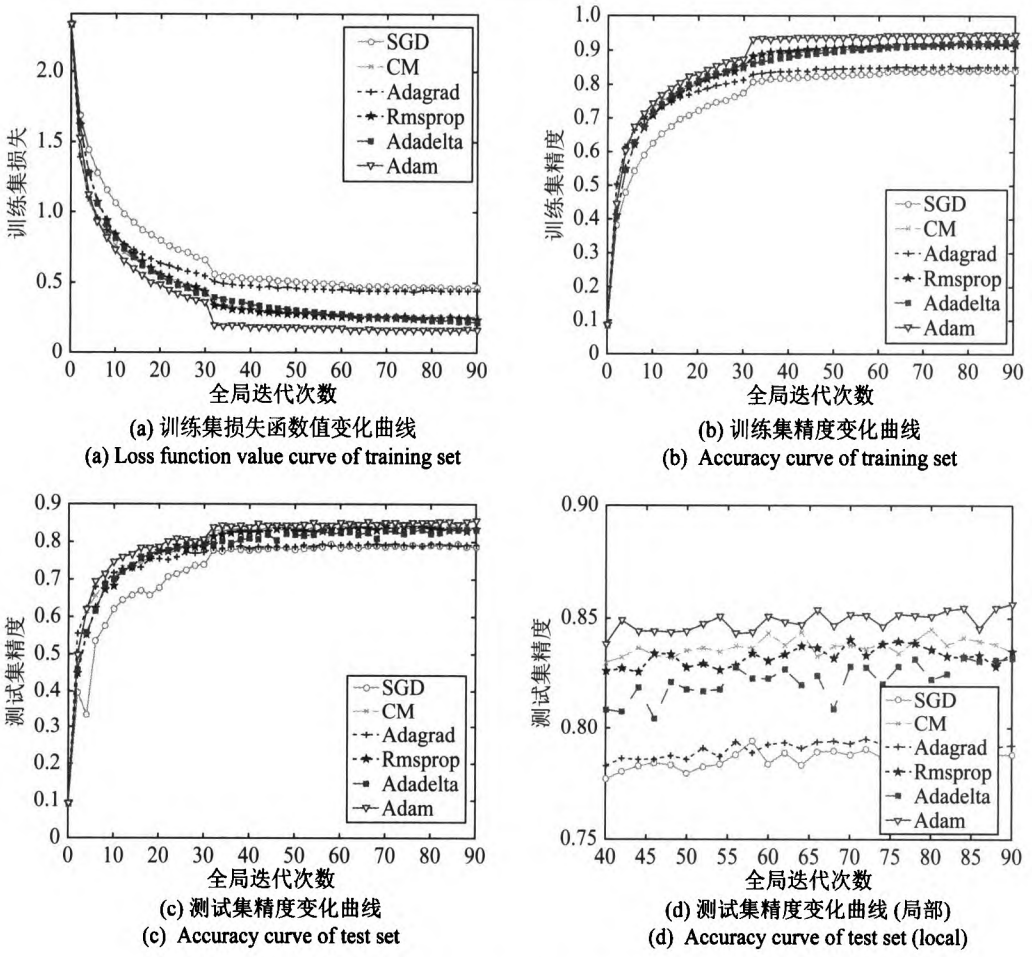


图 8 自适应学习率的随机梯度下降算法性能比较

Fig. 8 Performance comparison of stochastic gradient descent algorithms with adaptive learning rates

不理想. 随机梯度下降算法在非凸、非光滑条件下的策略研究, 不仅是当前面临的困局, 也是未来最具有应用价值的研究方向.

5) 对于非凸的优化问题, 梯度下降法通常存在两个缺陷: 易于陷入局部最优、无法逃离鞍点. 而演化计算/智能计算无需计算梯度和确定步长, 且往往具有较好的全局收敛性. 如何将随机梯度下降算法与演化计算/智能计算方法相结合, 将是一个非常值得关注的研究方向.

References

1 Jordan M I, Mitchell T M. Machine learning: Trends, perspectives, and prospects. *Science*, 2015, **349**(6245): 255–260

2 Lin Yi-Lun, Dai Xing-Yuan, Li Li, Wang Xiao, Wang Fei-Yue. The new frontier of AI research: Generative adversarial networks. *Acta Automatica Sinica*, 2018, **44**(5): 775–792
(林懿伦, 戴星原, 李力, 王晓, 王飞跃. 人工智能研究的新前线: 生成式对抗网络. 自动化学报, 2018, **44**(5): 775–792)

3 LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*, 2015, **521**(7553): 436–444

4 Bottou L, Curtis F E, Nocedal J. Optimization methods for large-scale machine learning. *SIAM Review*, 2018, **60**(2): 223–311

5 Jiao Li-Cheng, Zhao Jin, Yang Shu-Yuan, Liu Fang. *Deep Learning, Optimization and Recognition*. Beijing: Tsinghua University Press, 2017.
(焦李成, 赵进, 杨淑媛, 刘芳. 深度学习、优化与识别. 北京: 清华大学出版社, 2017.)

6 Robbins H, Monro S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951, **22**(3): 400–407

7 Amari S. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 1967, **EC-16**(3): 299–307

8 Bottou L. Online algorithms and stochastic approximations. *Online Learning and Neural Networks*. Cambridge, UK: Cambridge University Press, 1998.

9 Jiao Li-Cheng, Yang Shu-Yuan, Liu Fang, Wang Shi-Gang, Feng Zhi-Xi. Seventy years beyond neural networks: Retrospect and prospect. *Chinese Journal of Computers*, 2016, **39**(8): 1697–1717
(焦李成, 杨淑媛, 刘芳, 王士刚, 冯志玺. 神经网络七十年: 回顾与展望. 计算机学报, 2016, **39**(8): 1697–1717)

10 Kasiviswanathan S P, Jin H X. Efficient private empirical risk minimization for high-dimensional learning. In: Proceedings of the 33rd International Conference on Machine Learning. New York, USA: JMLR, 2016. 488–497

11 Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems. Red Hook, NY, United States: Curran Associates Inc., 2012. 1097–1105

12 Sutskever I, Martens J, Dahl G, Hinton G. On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on Machine Learning. Atlanta, USA: ACM, 2013. III-1139–III-1147

13 Shamir O. Fast stochastic algorithms for SVD and PCA: Convergence properties and convexity. In: Proceeding ofs the 33rd International Conference on Machine Learning. New York, USA: JMLR, 2016. 248–256

14 Shamir O. Convergence of stochastic gradient descent for PCA. In: Proceedings of the 33rd International Conference on Machine Learning. New York, USA: JMLR, 2016. 257–265

15 Garber D, Hazan E, Jin C, Kakade S M, Musco C, Netrapalli P, et al. Faster eigenvector computation via shift-and-invert preconditioning. In: Proceedings of the 33rd International Conference on Machine Learning. New York, USA: JMLR, 2016. 2626–2634

16 Allen-Zhu Z, Li Y Z. Doubly accelerated methods for faster CCA and generalized eigendecomposition. In: Proceedings of the 34th International Conference on Machine Learning. Sydney, Australia: JMLR, 2017. 98–106

17 Kasai H. Stochastic variance reduced multiplicative update for nonnegative matrix factorization. In: Proceedings of the 44th IEEE International Conference on Acoustics, Speech, and Signal Processing. Alberta, Canada: IEEE, 2018. 6338–6342

18 Zhang X, Wang L X, Gu Q Q. Stochastic variance-reduced gradient descent for low-rank matrix recovery from linear measurements [Online], available: https://arxiv.org/pdf/1701.00481.pdf, January 16, 2017

19 Ouyang H, He N, Tran L Q, Gray A. Stochastic alternating direction method of multipliers. In: Proceedings of the 30th International Conference on Machine Learning. Atlanta, USA: JMLR, 2013. I-80–I-88

20 Liu Y Y, Shang F H, Cheng J. Accelerated variance reduced stochastic ADMM. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence. California, USA: AAAI Press, 2017. 2287–2293

21 Qu C, Li Y, Xu H. Linear convergence of SVRG in statistical estimation [Online], available: https://arxiv.org/pdf/1611.01957.pdf, July 27, 2017

22 Paquette C, Lin H Z, Drusvyatskiy D, Mairal J, Harchaoui Z. Catalyst acceleration for gradient-based non-convex optimization [Online], available: https://arxiv.org/pdf/1703.10993.pdf, December 31, 2018

23 Duchi J C, Ruan F. Stochastic methods for composite optimization problems [Online], available: https://arxiv.org/pdf/1703.08570.pdf, September 21, 2018

24 Schmidt M, Babanezhad R, Ahmed M O, Defazio A, Clifton A, Sarkar A. Non-uniform stochastic average gradient method for training conditional random fields. In: Proceedings of the 18th International Conference on Artificial Intelligence and Statistics. California, USA: JMLR, 2015. 819–828

25 Qian N. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 1999, **12**(1): 145–151

26 Nesterov Y. A method for solving a convex programming problem with convergence rate $O(1/k2)$. *Soviet Mathematics Doklady*, 1983, **27**(2): 372–376

27 Botev A, Lever G, Barber D. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In: Proceedings of the 30th International Joint Conference on Neural Networks. Alaska, USA: IEEE, 2017.1899–1903

28 Johnson R, Zhang T. Accelerating stochastic gradient descent using predictive variance reduction. In: Proceedings of the 26th International Conference on Neural Information Processing Systems. Red Hook, NY, United States: Curran Associates Inc., 2013. 315–323

29 Xiao L, Zhang T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 2014, **24**(4): 2057–2075

30 Allen-Zhu Z. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 2018, **18**: 1–51

31 Zhou K W, Shang F H, Cheng J. A simple stochastic variance reduced algorithm with fast convergence rates [Online], available: https://arxiv.org/pdf/1806.11027.pdf, July 28, 2018

32 Schmidt M, Le Roux N, Bach F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2017, **162**(1-2): 83–112

33 Defazio A, Bach F, Lacoste-Julien S. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In: Proceedings of the 27th International Conference on Neural Information Processing Systems. Cambridge, MA, United States: MIT Press, 2014. 1646–1654

34 Defazio A. A simple practical accelerated method for finite sums. In: Proceedings of the 29th Conference on Neural Information Processing Systems. Barcelona, Spain: MIT Press, 2016. 676–684

35 Goodfellow I, Bengio Y, Courville A. *Deep Learning*. Cambridge: MIT Press, 2016. 267–309

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

37 Zeiler D M. ADADELTA: An adaptive learning rate method [Online], available: https://arxiv.org/pdf/1212.5701.pdf, December 22, 2012

38 Kingma D, Ba J. Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations. San Diego, USA: Workshop Track, 2015. 1–13

39 Dozat T. Incorporating Nesterov momentum into Adam. In: Proceedings of the 4th International Conference on Learning Representations. San Juan, Puerto Rico, USA: Workshop Track, 2016.

40 Ruder S. An overview of gradient descent optimization algorithms [Online], available: https://arxiv.org/pdf/1609.04747.pdf, June 15, 2017

41 Nesterov Y. Smooth convex optimization. *Introductory Lectures on Convex Optimization: A Basic Course*. Boston, MA: Springer, 2004. 51–110

42 Nesterov Y. Gradient methods for minimizing composite functions. *Mathematical Programming*, 2013, **140**(1): 125–161

43 Bottou L. Large-scale machine learning with stochastic gradient descent. In: Proceedings of the 19th International Conference on Computational Statistics. Paris, France: Physica-Verlag HD, 2010. 177–186

44 Hinton G, Srivastava N, Swersky K. Neural networks for machine learning: Lecture 6a overview of mini-batch gradient descent [Online], available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2012.

45 Li M, Zhang T, Chen Y Q, Smola A J. Efficient mini-batch training for stochastic optimization. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM, 2014. 661–670

46 Bach F. Linearly-convergent stochastic gradient algorithms [Online], available: http://homepages.laas.fr/henrion/aime@cz18/bach.pdf, February 22, 2018

47 Dekel O, Gilad-Bachrach R, Shamir O, Xiao L. Optimal distributed online prediction using mini-batches. *Journal of Machine*

Learning Research, 2012, **13**(6): 165–202

48 Li Fei, Gao Xiao-Guang, Wan Kai-Fang. Research on RBM accelerating learning algorithm with weight momentum. *Acta Automatica Sinica*, 2017, **43**(7): 1142–1159
(李飞, 高晓光, 万开方. 基于权值动量的RBM加速学习算法研究. 自动化学报, 2017, **43**(7): 1142–1159)

49 Zhu Xiao-Hui, Tao Qing, Shao Yan-Jian, Chu De-Jun. Stochastic optimization algorithm with variance reduction for solving non-smooth problems. *Journal of Software*, 2015, **26**(11): 2752–2761
(朱小辉, 陶卿, 邵言剑, 储德军. 一种减小方差求解非光滑问题的随机优化算法. 软件学报, 2015, **26**(11): 2752–2761)

50 Atchadé Y F, Fort G, Moulines E. On perturbed proximal gradient algorithms [Online], available: https://arxiv.org/pdf/1402.2365.pdf, November 19, 2016

51 Parikh N, Boyd S. Proximal algorithms. *Foundations and Trends® in Optimization*, 2014, **1**(3): 127–239

52 Nitanda A. Stochastic proximal gradient descent with acceleration techniques. In: Proceedings of the 27th International Conference on Neural Information Processing Systems. Cambridge, MA, United States: MIT Press, 2014. 1574–1582

53 Shang F H, Zhou K W, Liu H Y, Cheng J, Tsang I W, Zhang L J, et al. VR-SGD: A simple stochastic variance reduction method for machine learning. *IEEE Transactions on Knowledge and Data Engineering*, 2020, **32**(1): 188–202

54 Shalev-Shwartz S, Zhang T. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 2013, **14**(1): 567–599

55 Zhang Y C, Xiao L. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *Journal of Machine Learning Research*, 2017, **18**: 1–42

56 Gürbüzbalaban M, Ozdaglar A, Parrilo P A. On the convergence rate of incremental aggregated gradient algorithms. *SIAM Journal on Optimization*, 2017, **27**(2): 1035–1048

57 Tseng P, Yun S. Incrementally updated gradient methods for constrained and regularized optimization. *Journal of Optimization Theory and Applications*, 2014, **160**(3): 832–853

58 Poon C, Liang J W, Schönlieb C B. Local convergence properties of SAGA/Prox-SVRG and acceleration [Online], available: https://arxiv.org/pdf/1802.02554.pdf, November 1, 2018

59 Rockafellar R T. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 1976, **14**(5): 877–898

60 Dean J, Corrado G S, Monga R, Chen K, Devin M, Le Q V, et al. Large scale distributed deep networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems. Red Hook, NY, United States: Curran Associates Inc., 2012. 1223–1231

61 Pennington J, Socher R, Manning C. GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Doha, Qatar: ACL, 2014. 1532–1543

62 Chen Z Y, Xu Y, Chen E H, Yang T B. SADAGRAD: Strongly adaptive stochastic gradient methods. In: Proceedings of the 35th International Conference on Machine Learning. Stockholm, Swedish: ACM, 2018. 913–921

63 Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning*, 2021, **4**(2): 26–30

64 Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: Proceedings of the 2017 IEEE Symposium on Security and Privacy. San Jose, USA: IEEE, 2017. 39–57

65 Zhang K, Zuo W M, Chen Y J, Meng D Y, Zhang L. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 2017, **26**(7): 3142–3155

66 Hubara I, Courbariaux M, Soudry D, Yaniv R E. Binarized neural networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. Red Hook, NY, United States: Curran Associates Inc., 2016. 4114–4122

67 Luo L C, Huang W H, Zeng Q, Nie Z Q, Sun X. Learning personalized end-to-end goal-oriented dialog. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence. Honolulu, Hawaii, USA: AAAI, 2019.

68 Wilson A C, Roelofs R, Stern M, Srebro N, Recht B. The marginal value of adaptive gradient methods in machine learning. In: Proceedings of the 31st Conference on Neural Information Processing Systems. Long Beach, USA: MIT Press, 2017. 4148–4158

69 Reddi S J, Kale S, Kumar S. On the convergence of adam and beyond. In: Proceedings of the 6th International Conference on Learning Representations. Vancouver, Canada: Workshop Track, 2018.

70 Luo L C, Xiong Y H, Liu Y, Sun X. Adaptive gradient methods with dynamic bound of learning rate. In: Proceedings of the 7th International Conference on Learning Representations. New Orleans, USA: Workshop Track, 2019.

71 He K M, Zhang X Y, Ren S Q, Sun J. Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA: IEEE, 2016. 770–778

72 Xie Pei, You Ke-You, Hong Yi-Guang, Xie Li-Hua. A survey of distributed convex optimization algorithms over networks. *Control Theory and Applications*, 2018, **35**(7): 918–927
(谢佩, 游科友, 洪奕光, 谢立华. 网络化分布式凸优化算法研究进展. 控制理论与应用, 2018, **35**(7): 918–927)

73 Kang Liang-Yi, Wang Jian-Fei, Liu Jie, Ye Dan. Survey on parallel and distributed optimization algorithms for scalable machine learning. *Journal of Software*, 2018, **29**(1): 109–130
(亢良伊, 王建飞, 刘杰, 叶丹. 可扩展机器学习的并行与分布式优化算法综述. 软件学报, 2018, **29**(1): 109–130)

74 Zinkevich M A, Weimer M, Smola A, Li L H. Parallelized stochastic gradient descent. In: Proceedings of the 23rd International Conference on Neural Information Processing Systems. Red Hook, NY, United States: Curran Associates Inc., 2010. 2595–2603

75 Niu F, Recht B, Re C, Wright S J. HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent. In: Proceedings of the 24th International Conference on Neural Information Processing Systems. Red Hook, NY, United States: Curran Associates Inc., 2011. 693–701

76 Chen Zhen-Hong, Lan Yan-Yan, Guo Jia-Feng, Cheng Xue-Qi. Distributed stochastic gradient descent with discriminative aggregating. *Chinese Journal of Computers*, 2015, **38**(10): 2054–2063
(陈振宏, 兰艳艳, 郭嘉丰, 程学旗. 基于差异合并的分布式随机梯度下降算法. 计算机学报, 2015, **38**(10): 2054–2063)

77 Zhao S Y, Li W J. Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence. Phoenix, USA: AAAI Press, 2016. 2379–2385

78 De S, Goldstein T. Efficient distributed SGD with variance reduction. In: Proceedings of the 16th IEEE International Conference on Data Mining. Barcelona, Spain: IEEE, 2016. 111–120

79 Bach F, Moulines E. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In: Proceedings of the 24th International Conference on Neural Information Pro-

cessing Systems. Red Hook, NY, United States: Curran Associates Inc., 2011. 451–459

80 Babichev D, Bach F. Constant step size stochastic gradient descent for probabilistic modeling [Online], available: https://arxiv.org/pdf/1804.05567.pdf, November 21, 2018

81 Andrychowicz M, Denil M, Colmenarejo S G, Hoffman M W, Pfau D, Schaul T, et al. Learning to learn by gradient descent by gradient descent. In: Proceedings of the 30th Neural Information Processing Systems. Barcelona, Spain: MIT Press, 2016. 3981–3989

82 Allen-Zhu Z. Katyusha X: Practical momentum method for stochastic sum-of-nonconvex optimization [Online], available: https://arxiv.org/pdf/1802.03866.pdf, February 12, 2018

83 Allen-Zhu Z, Hazan E. Variance reduction for faster non-convex optimization. In: Proceedings of the 33rd International Conference on Machine Learning. New York, USA: JMLR, 2016. 699–707

84 Lei L, Ju C, Chen J, Jordan M. Non-convex finite-sum optimization via SCSG methods. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Red Hook, NY, United States: Curran Associates Inc., 2017. 2345–2355

85 Allen-Zhu Z. Natasha 2: Faster non-convex optimization than SGD. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. Red Hook, NY, United States: Curran Associates Inc., 2018. 2680–2691

86 Li Q W, Zhou Y, Liang Y B, Varshney P K. Convergence analysis of proximal gradient with momentum for nonconvex optimization. In: Proceedings of the 34th International Conference on Machine Learning. Sydney, Australia: JMLR, 2017. 2111–2119

87 Zhou D R, Xu P, Gu Q Q. Stochastic nested variance reduced gradient descent for nonconvex optimization. In: Proceedings of the 32nd Conference on Neural Information Processing Systems. Montreal, Canada: MIT Press, 2018. 3921–3932

88 Jin C, Ge R, Netrapalli P, Kakade S M, Jordan M I. How to escape saddle points efficiently. In: Proceedings of the 34th International Conference on Machine Learning. Sydney, Australia: JMLR, 2017. 1724–1732

89 Reddi S J, Sra S, Póczos B, Smola A J. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. Red Hook, NY, United States: Curran Associates Inc., 2016. 1153–1161

90 Li Z Z, Li J. A simple proximal stochastic gradient method for nonsmooth nonconvex optimization. In: Proceedings of the 32nd Conference on Neural Information Processing Systems. Montreal, Canada: MIT Press, 2018. 5569–5579

史加荣 西安建筑科技大学教授. 主要研究方向为机器学习. 本文通信作者.
E-mail: shijiarong@xauat.edu.cn
(**SHI Jia-Rong** Professor at Xi'an University of Architecture and Technology. His main research interest is machine learning. Corresponding author of this paper.)

王 丹 西安建筑科技大学硕士研究生. 主要研究方向为机器学习和随机优化算法.
E-mail: wangdan_edu@163.com
(**WANG Dan** Master student at Xi'an University of Architecture and Technology. Her research interest covers machine learning and stochastic optimization algorithm.)

尚凡华 西安电子科技大学教授. 主要研究方向为机器学习, 并行/分布式计算.
E-mail: fhshang@xidian.edu.cn
(**SHANG Fan-Hua** Professor at Xidian University. His research interest covers machine learning, parallel/distributed computing.)

张鹤于 西安电子科技大学硕士研究生. 主要研究方向为深度学习和分布式随机优化.
E-mail: heyuzhang@stu.xidian.edu.cn
(**ZHANG He-Yu** Master student at Xidian University. Her research interest covers deep learning and distributed stochastic optimization.)