第50卷第8期 中国科学技术大学学报 Vol. 50, No. 8
2020年8月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Aug. 2020
文章编号:0253-2778(2020)08-1156-06

分组随机梯度下降法:掉队和延迟的平衡

高 翔,陈 力

(中国科学技术大学电子工程与信息科学系,安徽合肥 230027)

摘要:分布式随机梯度下降法被广泛应用于大规模机器学习,同步随机梯度下降法和异步随机梯度下降法是两个典型的分布式随机梯度下降法.在同步随机梯度下降法中,所有的工作节点都需要互相等待,导致训练速度受限于最慢的工作节点.在异步随机梯度下降法中,延迟的梯度会造成最终训练得到的模型很差.为此提出一种新的分布式随机梯度下降法:分组随机梯度下降法.该方法将通信和计算性能相近的工作节点划入同一组,这样就会将工作节点划分成若干的组.在同一组的工作节点以同步的方式工作,不同的组之间以异步的方式工作.由于组内的工作节点只需互相等待很短的时间,该方法可以缓解同步随机梯度下降法的掉队问题.由于组的数目远小于工作节点的数目,该方法梯度的延迟也很小.理论分析证明了该方法的收敛性.仿真结果表明,在异质集群中该方法的收敛速度比同步随机梯度下降法和异步随机梯度下降法更快.

关键词:随机梯度下降;分布式机器学习;掉队者;延迟

中图分类号:TP181 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2020.08.016

引用格式:高翔,陈力. 分组随机梯度下降法:掉队和延迟的平衡[J]. 中国科学技术大学学报,2020,50(8):1156-1161.

GAO Xiang, CHEN Li. Group stochastic gradient descent: A tradeoff between straggler and staleness [J]. Journal of University of Science and Technology of China, 2020,50(8):1156-1161.

Group stochastic gradient descent: A tradeoff between straggler and staleness

GAO Xiang, CHEN Li

(*Department of Electronic Engineering and Information Science, University of Science and Technology of China, HeFei* 230027, *China*)

Abstract: Distributed stochastic gradient descent(DSGD)is widely used for large scale distributed machine learning. Two typical implementations of DSGD are synchronous SGD(SSGD)and asynchronous SGD (ASGD). In SSGD, all workers should wait for each other and the training speed will be slowed down to that of the straggler. In ASGD, the stale gradients can result in a poorly trained model. To solve this problem, a new version of distributed SGD method based named group SGD(GSGD)is proposed, which puts workers with similar computation and communication performance in a group and divides them into several groups. The workers in the same group work in a synchronous manner while different groups work in an asynchronous manner. The proposed method can migrate the straggler problem since workers in the same group spend little time waiting for each other. The staleness of the method is small since the number of groups is much smaller than the number of workers. The convergence of the method is proved through theoretical analysis. Simulation results show that the method converges faster than SSGD and ASGD in the heterogeneous cluster.

Key words: Stochastic gradient descent; distributed machine learning; straggler; staleness

0 引言

在机器学习中,随机梯度下降法(SGD)是一种常用的优化方法[1-3],被成功地应用到机器学习的各领域,如图像识别[4]、语音识别[5]和文本处理[6].传统的SGD只能在单个计算节点上运行,然而当数据集很大时,单机上的SGD训练需要很长的时间.此外,有时数据集会非常庞大,单个节点的存储空间可能完全无法容纳整个数据集.分布式随机梯度下降法(DSGD)通过将原本的计算任务分配到多个节点上,很好地解决了上述问题[7-9].在这些节点中,有一个节点是参数服务器节点,其余节点都是工作节点[7,10].训练数据集被随机等分到所有工作节点.参数服务器节点会将模型下发给工作节点,工作节点

收稿日期:2020-07-01;**修回日期**:2020-07-28

作者简介:高翔,男,1995年生,硕士生.研究方向:分布式机器学习.E-mail:xgao0@mail.ustc.edu.cn

通讯作者:陈力,博士/副教授.E-mail:chenli87@ustc.edu.cn

收到来自参数服务器节点的模型后会结合本地的训练数据集计算随机梯度,并将该随机梯度发送至参数服务器节点,参数服务器节点会根据工作节点发送的随机梯度来更新模型.

当所有的工作节点以同步的方式工作时,这种方式被称为同步随机梯度下降法(SSGD). SSGD 的缺点是效率不高. 这是由于在每次迭代中,运行速度快的工作节点需要等待运行速度慢的工作节点,所以会造成运行速度快的工作节点完成计算后进入空闲的等待状态.

当所有的工作节点以异步的方式工作时,这种方式被称为异步随机梯度下降法(ASGD)[7]. ASGD 的缺点是训练得到模型精度低[11-12]. 这是由于在每次迭代中,用于更新模型参数的梯度存在延迟,这个延迟会对模型的收敛产生负面影响.

针对现有研究的缺陷,本文提出了分组随机梯度下降法(GSGD). 在该方法中,性能相同或相似的节点被放入同一组内,每组工作节点的数量不需要相同. 组内的工作节点以同步的方式获取模型和提供用于更新的随机梯度,而组与组之间以异步的方式更新模型. 理论分析证明,本文提出的算法是收敛的. 仿真结果表明,在异质集群中,该算法相比 SSGD 和 ASGD 能够更快的收敛.

1　相关工作

针对 SSGD 中效率不高和 ASGD 中精度不高的问题,研究人员提出了改进方案.

对于 SSGD,文献[13]使用一些备份的工作节点,并且忽略部分速度太慢的工作节点的更新结果. 这种做法仅仅在各节点拥有相似的通信和计算性能的同质集群中有效. 当某些工作节点速度一直很慢,这些工作节点的计算结果可能永远无法被用于更新模型. 文献[14]使用编码的方式解决掉队节点的问题,但这样会造成某些节点计算资源的浪费. 文献[15]提出将运行速度慢的工作节点的计算任务分配到运行速度快的节点上. 由于这一方法需要数据在工作节点之间传输,所以会消耗更多的通信资源.

对于 ASGD,文献[16]提出一种步长调节算法以缓解分布式系统中的延迟问题. 文献[17]对延迟梯度添加了一个指数的惩罚项来降低延迟对收敛的影响. 以上两种方法都会在当延迟很高时,使得学习率很小,从而导致收敛速率很慢. 文献[18]指出延迟的梯度只是真实梯度泰勒展开的零阶项并提出使用泰勒展开中的更多项去逼近真实梯度. 由于仅仅计算真实梯度泰勒展开中的一阶项就涉及计算损失函数的海森矩阵,这会大幅提高计算复杂度.

在 SSGD 中,训练过程精确但是效率不高,这与 ASGD 恰好相反. 有一些工作将 SSGD 和 ASGD 相结合来实现精度和效率的折中. 文献[19]提出了一种延时同步并行的方案. 在该方案中,各工作节点异步执行,但是最快和最慢的节点之间相差的迭代次数不能超过预设的阈值. 一旦迭代次数差值达到阈值,最快的节点就会停止运行并等待,直到最慢的节点完成迭代使得差值小于阈值时,最快的节点才能继续运行. 该方案仅在同质集群中有效,如果某个工作节点的运行速度一直很慢,那么整体的运行速度会被该节点一直拖累. 在文献[20-21]中,参数服务器节点只要收到固定数量的来自工作节点的信息就会更新模型. 更近一步,文献[22]提出将工作节点分成许多组,每组中有相等数量的工作节点. 当一组中工作节点的性能相差较大时,运行速度快的工作节点仍然需要花费大量时间去等待慢的工作节点. 一个直观的解决方案时把性能相同或相近的工作节点划入同一组,但是这样每一组中的工作节点数就不相等了,现有的方法也不再适用.

2　系统模型

本节将介绍应用于分布式机器学习的参数服务器系统模型,对该模型下要解决的机器学习问题进行数学描述. 之后将对该系统模型下 SSGD 和 ASGD 算法进行描述并分析讨论这两种优化方法的优缺点.

2.1　系统模型

参数服务器的系统模型图 1 所示. 假设在该模型中有一个参数服务器节点和 P 个工作节点.

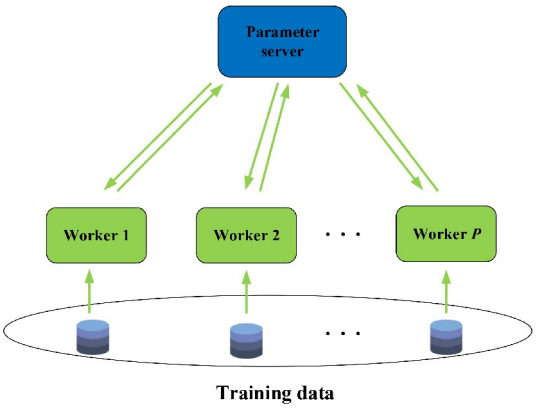

图 1　参数服务器模型

Fig. 1　Parameter server model

在数据并行机器学习中,整个数据集被随机等分为 P 个不相交的子数据集. 这 P 个子数据集被随机分配到各工作节点上,最终每个工作节点都将拥有一份自己的本地数据集. 分布式机器学习要解决的问题就是通过联合这些工作节点来最小化在整个数据集上的损失函数,即

$$\min_{w\in\mathbf{R}^d} F(\boldsymbol{w})=\frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{w};s_i) \tag{1}$$

式中,N 是整个训练数据集中的数据样本个数,s_i 表示第 i 个数据样本,$\boldsymbol{w}$ 代表模型,$f(\boldsymbol{w};s_i)$ 表示在模型 $\boldsymbol{w}$ 和数据样本 s_i 上的损失函数.

为了能够在分布式场景下解决式(1)中的问题,SSGD 和 ASGD 是常用的优化方法.

2.2　SSGD

在 SSGD 每次迭代中,参数服务器节点模型的更新规则为

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t-\eta_t\frac{1}{P}\sum_{i=1}^{P}g(\boldsymbol{w}_t;\xi_{t,i}) \tag{2}$$

式中，$\xi_{t,i}$ 表示第 t 次迭代从第 i 个工作节点抽样得到的批量大小为 m 的数据样本集，$g(\boldsymbol{w}_t;\xi_{t,i})=\frac{1}{m}\sum_{s\in\xi_{t,i}}\nabla f(\boldsymbol{w}_t;s)$ 表示这个数据样本集上计算出来的平均梯度，η_t 表示学习率. 工作节点从参数服务器节点获取模型参数并计算各自的小批量随机梯度，并将该梯度发送给参数服务器节点. 参数服务器节点收到来自所有工作节点的小批量随机梯度时，会按照式(2)更新模型参数，最后将更新后的模型参数发送给所有的工作节点. 工作节点在收到更新后的模型参数后就可以继续计算小批量随机梯度. 整个训练过程如图 2(a)所示. 在 SSGD 中，参数服务器节点需要等待所有工作节点的随机梯度才能更新模型. 由于每个工作节点的计算和通信性能可能是不同的，快的工作节点在每轮迭代中需要等待慢的工作节点，这导致训练的速度受限于速度最慢的节点. 这就是 SSGD 中的掉队者问题.

图 2 当 $P=4$ 时，SSGD 和 ASGD 的训练过程

Fig. 2 For $P=4$, the training process of SSGD and ASGD

2.3 ASGD

在 ASGD 每次迭代中，参数服务器节点模型的更新规则为

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t-\eta_t g(\boldsymbol{w}_{t-\tau_t};\xi_{t,i(t)}) \tag{3}$$

式中，$\xi_{t,i(t)}$ 表示在第 t 次迭代过程中，第 $i(t)$ 个工作节点抽样得到的批量大小为 m 的数据样本集，$g(\boldsymbol{w}_{t-\tau_t};\xi_{t,i(t)})=\frac{1}{m}\sum_{s\in\xi_{t,i(t)}}\nabla f(\boldsymbol{w}_{t-\tau_t};s)$ 表示该数据样本集的平均梯度，τ_t 表示第 t 次迭代的延迟，η_t 表示学习率. 在 ASGD 中，工作节点不需要互相等待. 只要参数服务器节点收到来自任意一个工作节点的随机梯度，它就会根据式(3)更新模型参数并将新的模型参数传回给发送梯度的那个工作节点. 当该工作节点收到更新的模型参数后则可以继续计算随机梯度. 整个训练过程如图 2(b)所示.

在 ASGD 中，在第 t 次迭代中用于更新模型参数 $\boldsymbol{w}_t$ 的小批量随机梯度是基于模型参数 $\boldsymbol{w}_{t-\tau_t}$ 得到的. 延迟 τ_t 将会对模型的收敛产生负面的影响，并将导致最终训练的模型的准确度低于用 SSGD 方法得到的模型的准确度. 这就是 ASGD 中的延迟问题. 在 SSGD 中，训练过程准确但是低效，而在 ASGD 中情况正好相反，因此本文提出 GSGD 来实现精度和效率的折中.

3 GSGD 算法设计

本节将介绍提出的解决分布式机器学习的 GSGD 算法.

3.1 主要思想

SSGD 和 ASGD 分别有掉队者问题和延迟问题. 在 SSGD 中，所有的工作节点以同步的方式工作，训练的速度将会被运行速度最慢的工作节点严重拖累. 在 ASGD 中，所有的工作节点以异步的方式工作. 在大规模分布式机器学习中，工作节点的数量会很多，所以 ASGD 梯度的延迟也会很大，这将严重阻碍模型的收敛并导致最终收敛精度不高. 有一些方法将相等数量的工作节点划分到一个组中，以实现精度和效率的平衡. 然而，当一个组内的工作节点的通信和计算性能相差较大时，掉队者效应仍然不可避免. 如果将性能相同或相近的工作节点划入同一组内，而不要求每一组的工作节点数量相等时，则同一组内工作节点就不需要花费很长的时间去等待. 例如，有许多工作节点位于不同的数据中心. 在同一数据中心的工作节点具有相同版本的 CPU 或 GPU，并且这些工作节点的通信性能也几乎相同，但是在不同的数据中心的工作节点具有不同的计算和通信性能. 如果将位于同一数据中心的工作节点划入一个组内，让同一组内的工作节点以同步的方式运行，同时不同的组以异步的方式去更新模型参数. 这样一来，组内的工作节点将不需要花费大量时间等待. 又因为组的数量远远小于工作节点的数量，用于更新模型参数的随机梯度的延迟也很低. 这就是 GSGD 的主要思想.

3.2 完整算法

在 GSGD 中，性能相同或相近的节点被划入同一组内. 假设一共划分了 $K(K<<N)$ 个组，$\mathcal{G}_k$ 用来表示第 K 个组的所有工作节点的结合，$|\mathcal{G}_k|$ 表示该组内工作节点的数量. GSGD 的系统框架如图 3 所示. 在 GSGD 中，每个工作节点计算小批量随机梯度并将该梯度发送至参数服务器节点. 参数服务器节点会聚集来自工作节点的梯度并进行模型参

数的更新.本文将梯度的聚集和模型的更新分成两个部分:组内聚集和组间更新.在组内聚集时,参数服务器节点将来自同一组的梯度聚集到一起求平均.为了便于后续描述,本文将求平均后的组内的梯度称为组梯度.在组间更新时,参数服务器节点只要得到任意一组的组梯度,就会更新模型参数.这两个步骤可用公式表示为

$$g_{k(t)}=\frac{1}{|\mathcal{G}_{k(t)}|}\sum_{i\in\mathcal{G}_{k(t)}}g(\boldsymbol{w}_{t-\tau_t};\xi_{t,i}) \tag{4}$$

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t-\eta_t\,g_{k(t)} \tag{5}$$

式中,$k(t)$ 表示第 t 次用于提供组梯度的组的索引,$\mathcal{G}_{k(t)}$ 表示第 $k(t)$ 个组的组内所有工作节点的集合,$|\mathcal{G}_{k(t)}|$ 表示该组内节点的个数,$g_{k(t)}$ 表示第 t 次迭代用于更新模型参数的组梯度.模型参数更新之后,参数服务器节点会将新的模型参数传回提供组梯度的对应组的所有工作节点.整个训练过程如图 4 所示.图 4 中,工作节点 1 和工作节点 2 被分在一组,工作节点 3 和工作节点 4 被分在另一组.完整的算法流程如算法 3.1 所示.

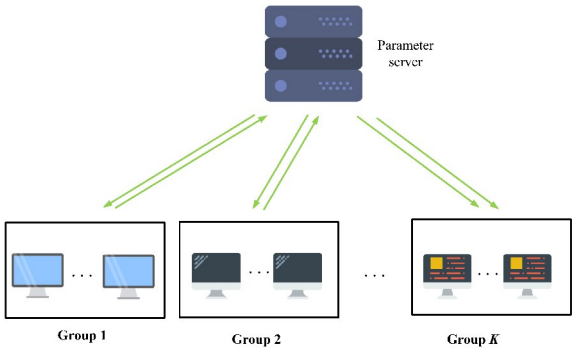

图 3　GSGD 的系统结构

Fig. 3　System structure of GSGD

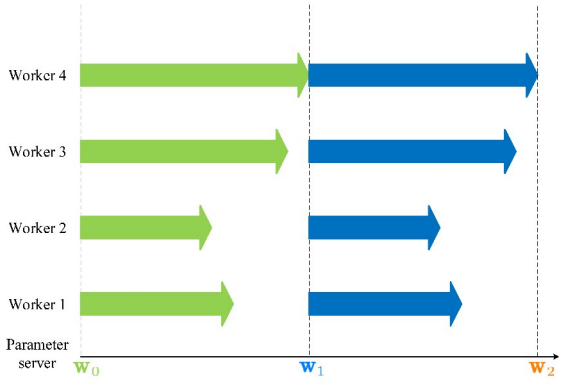

图 4　当 $P=4$ 时,GSGD 的训练过程

Fig. 4　For $P=4$, the training process of GSGD

算法 3.1　GSGD

工作节点 $p=1,2,\cdots,P$:

Repeat

Pull model $\boldsymbol{w}$ from the central server

Randomly select m training samples from local dataset and compute stochastic gradient $g(\boldsymbol{w};\xi)$

Push $g(\boldsymbol{w};\xi)$ to parameter server

UNTIL converge

参数服务器节点:

Input: learning rate η

Initialize: t=0, set the initialize model $\boldsymbol{w}_0$

Put workers with similar performance into the same group and get K groups: $\mathcal{G}_1,\cdots,\mathcal{G}_K$

REPEAT

Execute intragroup aggregation and get group gradient based on Eq. (4)

Execute intergroup update and get new model parameters based on Eq. (5)

$t\leftarrow t+1$

Send $\boldsymbol{w}_t$ back to the workers in group $k(t)$

Until converge

3.3　收敛性分析

为了能够进行收敛性分析,本文先做以下几个常用的假设[3,23-24]:

假设 3.1　$F(\boldsymbol{w})$ 是 μ-光滑的,因此对于任意的 $\boldsymbol{w}$, $\boldsymbol{w}'$,有

$$F(\boldsymbol{w})-F(\boldsymbol{w}')\leqslant\langle\nabla F(\boldsymbol{w}),\boldsymbol{w}'-\boldsymbol{w}\rangle+\frac{\mu}{2}\|\boldsymbol{w}'-\boldsymbol{w}\|^2 \tag{6}$$

假设 3.2　$F(\boldsymbol{w})$ 是 c −强凸的,因此对于任意的 $\boldsymbol{w}$, $\boldsymbol{w}'$,有

$$F(\boldsymbol{w}')-F(\boldsymbol{w})\geqslant\langle\nabla F(\boldsymbol{w}),\boldsymbol{w}'-\boldsymbol{w}\rangle+\frac{c}{2}\|\boldsymbol{w}'-\boldsymbol{w}\|^2 \tag{7}$$

因为 $F(\boldsymbol{w})$ 是强凸的,所以 $F(\boldsymbol{w})$ 拥有一个唯一的最小值,用 $\boldsymbol{w}_*$ 表示其最小值所在点,则对于任意的 $\boldsymbol{w}$,有

$$2c(F(\boldsymbol{w})-F(\boldsymbol{w}_*))\leqslant\|\nabla F(\boldsymbol{w})\|_2^2 \tag{8}$$

上面的不等式证明见文献[23]中式(4.12).

假设 3.3　随机等分到所有工作节点的数据集都是独立同分布的,且任意一个节点上样本的随机梯度都是真实梯度的无偏估计.即对于任意 $\boldsymbol{w}$,有

$$E_s[\nabla f(\boldsymbol{w};s)]=\nabla F(\boldsymbol{w}) \tag{9}$$

式中,s 表示任意一个随机采样的数据样本.

假设 3.4　随机梯度的方差是有界的.即对于任意 $\boldsymbol{w}$,有

$$E_s\|\nabla f(\boldsymbol{w};s)-\nabla F(\boldsymbol{w})\|^2\leqslant\sigma^2 \tag{10}$$

假设 3.5　由文献[24],使用一种通用的延迟界限.对于某个 $\gamma\leqslant1$,有

$$E\|\nabla F(\boldsymbol{w}_t)-\nabla F(\boldsymbol{w}_{t-\tau_t})\|^2\leqslant\gamma E\|\nabla F(\boldsymbol{w}_t)\|^2 \tag{11}$$

式中,γ 是延迟的度量,γ 越小,延迟越低.由以上的假设,可以推导出如下定理.

定理 3.1　如果学习率满足 $\eta\leqslant\frac{1}{\mu}$,则有

$$E[F(\boldsymbol{w}_t)-F(\boldsymbol{w}_*)]\leqslant\alpha^tE[F(\boldsymbol{w}_0)-F(\boldsymbol{w}_*)]+\frac{\mu\sigma^2\eta^2}{2m}\sum_{i=0}^{t-1}\frac{\alpha^i}{|\mathcal{G}_{k(t-1-i)}|} \tag{12}$$

式中,$\alpha=1-\eta\,c(1-\gamma)$.证明见附录 A.

4　仿真结果与分析

4.1　仿真设置

为了能够验证 GSGD 的性能,使用 python 进

行仿真. 本文采用 smooth SVM 和 logistic regression 这两个机器学习模型去进行训练. 这两个机器学习模型的损失函数如表 1 所示. 表 1 损失函数中，第 i 个训练数据样本 s_i 被表示为 (x_i, y_i)，x_i 是机器学习模型的一个输入矢量，y_i 是该训练数据样本的标签. 所有的模型都是在 MINIST 数据集上进行训练的. 对于这两个机器学习模型，本文将 MINIST 数据集中的奇数和偶数看成二元标签. 假设一共有 16 个工作节点连接一个参数服务器节点. 整个 MINIST 数据集被随机等分到所有的工作节点. 对于每个工作节点，设置随机抽样的批量大小为 32. 这 16 个工作节点是异质的，这意味着不同的工作节点在每次更新过程将花费不同的时间. 一次更新过程包括工作节点从参数服务器节点获取新的模型参数，计算随机梯度，将该梯度发送至参数服务器节点.

表 1 机器学习模型的损失函数

Tab. 1 Loss functions of machine learning models

模型	损失函数
smooth SVM	$\frac{\lambda}{2}\|\boldsymbol{w}\|^2+\frac{1}{2N}\sum_{i=1}^{N}\max\{0;1-y_i\boldsymbol{w}^T x_i\}^2$
logistic regression	$\frac{\lambda}{2}\|\boldsymbol{w}\|^2-\frac{1}{N}\sum_{i=1}^{N}[y_i\log\sigma(w,x_i)+(1-y_i)\log(1-\sigma(w,x_i))]$，其中 $\sigma(w,x_i)=\frac{1}{1+e^{-\boldsymbol{w}^T x_i}}$

在本文的仿真中，设 4 个工作节点在一次更新过程中需要 1 个单位的时间，4 个工作节点需要 2 个单位的时间，剩余的 8 个工作节点需要 3 个单位的时间. 在 GSGD 中，在一次更新过程中花费相同时间的工作节点将被划入同一组，所以总共有 3 组. 我们将 GSGD 与 ASGD 和 SSGD 进行比较. 对于 smooth SVM，设置 $\lambda=0.01$，ASGD 的学习率为 $\eta=0.001$，GSGD 的学习率为 $\eta=0.015$，SSGD 的学习率为 $\eta=0.04$. 对于 logistic regression，设置 $\lambda=0.001$，ASGD 的学习率为 $\eta=0.01$，GSGD 的学习率为 $\eta=0.08$，SSGD 的学习率为 $\eta=0.2$. 为了公平起见，所有的仿真都从同一个初始模型开始.

4.2 仿真结果与分析

图 5 和图 6 分别展示了两种机器学习模型下损失函数的值和测试精度随着迭代次数变化的曲线. 图 5(a)和(b)表明 GSGD 在 smooth SVM 下会随着迭代次数的增加而收敛. 图 6(a)和(b)表明 GSGD 在 logistic regression 下也是收敛的. 因此，GSGD 的收敛性得到了验证. 从图 5 和图 6 中可以看出，SSGD 收敛速率最快并且最后的误差很小，ASGD 收敛速率最慢且最后的误差很大. GGSD 的收敛速率略低于 SSGD 且最后的误差略大于 SSGD. 这是因为 ASGD 梯度的延迟很大，导致收敛的速率很慢且最后得到的模型精度很低，而 GSGD 的延迟相对 ASGD 来说要小很多. 结果表明 GSGD 随着迭代次数的收敛相对 ASGD 要稳定的多，且 GSGD 最终得到的模型精度与 SSGD 最终得到的基本相等.

图 7 和图 8 分别展示了两种机器学习模型下损失函数的值和测试精度随着时间的变化曲线. 从两张图中可以看出，随着时间的增加，GSGD 是收敛的. 在图 7 和图 8 中，ASGD 在一开始收敛很快，但是很快收敛速度就会变慢，且最后收敛的模型精度不高. SSGD 能够收敛得到一个好的模型，但是由于所有的节点都需要同步，收敛速度很慢. GSGD 收敛速度最快，且能够得到一个很好的模型精度. 在 GSGD 中，工作节点不需要花费很长的等待时间，且梯度的延迟也很低. 结果表明，GSGD 可以很好地缓解 SSGD 中的掉队者问题和 ASGD 中的延迟过高问题. 在有限的时间内，GSGD 能够训练出最好的模型.

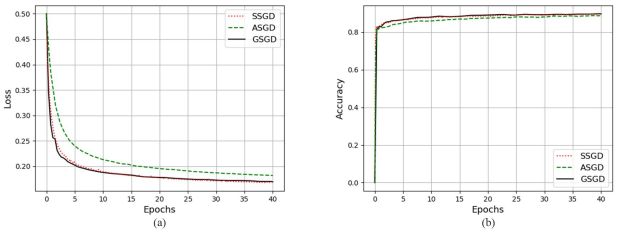

图 5 SVM 的损失函数值和测试精度随迭代次数的关系曲线

Fig. 5 The curves of loss function values and test accuracy w. r. t iteration times for SVM

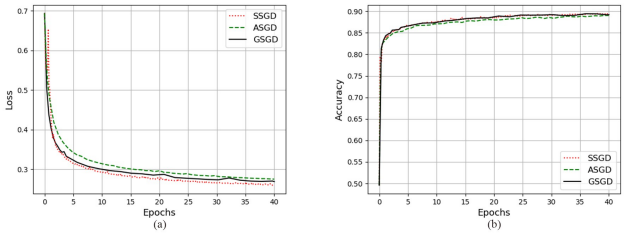

图 6 logistic regression 的损失函数值和测试精度随迭代次数的关系曲线

Fig. 6 The curves of loss function values and test accuracy w. r. t iteration times for logistic regression

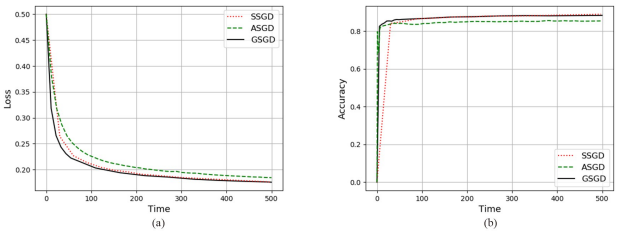

图 7 SVM 的损失函数值和测试精度随时间的关系曲线

Fig. 7 The curves of loss function values and test accuracy w. r. t time for SVM

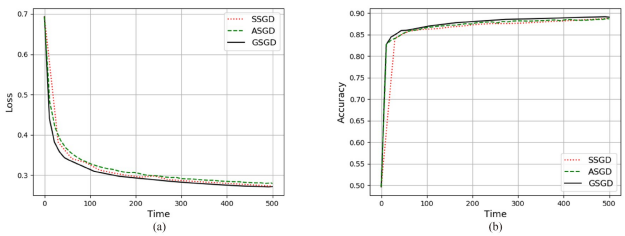

图 8 logistic regression 的损失函数值和测试精度随时间的关系曲线

Fig. 8 The curves of loss function values and test accuracy w. r. t time for logistic regression

5 结论

本文提出了一种新的分布式 SGD 算法(GSGD).该算法将性能相同或相近的工作节点划入同一组内来减少工作节点的空闲等待时间和限制梯度的延迟.通过理论的推导证明了所提算法的收敛性.仿真结果表明,GSGD 是收敛的,且在异质集群中,其随时间的收敛速度要比 SSGD 和 ASGD 要快.当时间有限时,GSGD 能够训练得到最精确的模型.

参考文献(References)

[1] BOTTOU L. Stochastic gradient learning in neural networks[J]. Proceedings of Neuro-Nimes, 1991, 91(8):12.

[2] BOTTOU L. Large-scale machine learning with stochastic gradient descent[C]//Proceedings of COMPSTAT '2010. Berlin, German: Springer, 2010: 177-186.

[3] RAKHLIN A, SHAMIR O, SRIDHARAN K. Making gradient descent optimal for strongly convex stochastic optimization[C]//International Conference on Machine Learning. Basel, Switzerland: MDPI, 2012:449-456.

[4] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet classification with deep convolutional neural networks[C]//Advances in Neural Information Processing Systems. New York, USA: Curran Associates Inc, 2012: 1097-1105.

[5] DAHL G E, YU D, DENG L, et al. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition[J]. IEEE Transactions on audio, speech, and language processing, 2011, 20(1):30-42.

[6] COLLOBERT R, WESTON J. A unified architecture for natural language processing: Deep neural networks with multitask learning[C]//Proceedings of the 25th International Conference on Machine Learning. New York, USA: ACM, 2008: 160-167.

[7] DEAN J, CORRADO G S, MONGA R, et al. Large scale distributed deep networks[J]. Advances in Neural Information Processing Systems, 2012, 2:1223-1231.

[8] XING E P, HO Q, DAI W, et al. Petuum: A new platform for distributed machine learning on big data[J]. IEEE Transactions on Big Data, 2015, 1(2):49-67.

[9] ABADI M, AGARWAL A, BARHAM P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems[J]. 2015, arXiv: 1603.04467.

[10] LI M, ANDERSEN D G, PARK J W, etal. Scaling distributed machine learning with the parameter server[C]//Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2014: 583-598.

[11] ZHANG S, CHOROMANSKA A E, LECUN Y. Deep learning with elastic averaging sgd[C]//Proceedings of the 28th International Conference on Neural Information Processing Systems. Cambridge, MA, USA: MIT Press, 2015: 685-693.

[12] LIAN X, HUANG Y, LI Y, et al. Asynchronous parallel stochastic gradient for nonconvex optimization[C]//Proceedings of the 28th International Conference on Neural Information Processing Systems. Cambridge, MA, USA: MIT Press, 2015: 2737-2745.

[13] CHEN J, PAN X, MONGA R, et al. Revisiting distributed synchronous sgd[J]. arXiv preprint arXiv: 1604.00981, 2016.

[14] TANDON R, LEI Q, DIMAKIS A G, et al. Gradient coding: Avoiding stragglers in distributed learning[C]//Proceedings of the 34th International Conference on Machine Learning. Sydney, Australia: PMLR, 2017: 3368-3376.

[15] HARLAP A, CUI H, DAI W, et al. Addressing the straggler problem for iterative convergent parallel ml[C]//Proceedings of the Seventh ACM Symposium on Cloud Computing. New York, NY, USA: ACM, 2016:98-111.

[16] MCMAHAN H B, STREETER M. Delay-tolerant algorithms for asynchronous distributed online learning[J]. Advances in Neural Information Processing Systems, 2014, 4:2915-2923.

[17] CHAN W, LANE I. Distributed asynchronous optimization of convolutional neural networks[J]. College & Research Libraries, 2014, 76(6):756-770.

[18] ZHENG S, MENG Q, WANG T, et al. Asynchronous stochastic gradient descent with delay compensation[C]//Proceedings of the 34th International Conference on Machine Learning. New York, NY: ACM, 2017: 4120-4129.

[19] HO Q, CIPAR J, CUI H, et al. More effective distributed ml via a stale synchronous parallel parameter server[C]//Proceedings of the 26th International Conference on Neural Information Processing Systems. New York, USA: Curran Associates Inc, 2013: 1223-1231.

[20] GUPTA S, ZHANG W, WANG F. Model accuracy and runtime tradeoff in distributed deep learning: A systematic study[C]//16th International Conference on Data Mining. NEW YORK, NY: IEEE, 2016: 171-180.

[21] ZHANG W, GUPTA S, LIAN X, et al. Staleness-aware async-sgd for distributed deep learning[J]. In International Joint Conference on Artificial Intelligence, 2016: 2350-2356.

[22] BASU S, SAXENA V, PANJA R, et al. Balancing stragglers against staleness in distributed deep learning[C]//25th International Conference on High Performance Computing. NEW YORK, NY: IEEE, 2018: 12-21.

[23] BOTTOU L, CURTIS F E, NOCEDAL J. Optimization methods for large-scale machine learning[J]. Siam Review, 2016, 60(2):223-311.

[24] DUTTA S, JOSHI G, GHOSH S, et al. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd[J]. 2018, arXiv:1803.01113.