基于近似牛顿法的分布式卷积神经网络训练

王雅慧[1,2] 刘 博[3] 袁晓彤[1,2]
(南京信息工程大学信息与控制学院 南京 210044)[1] (江苏省大数据分析技术重点实验室 南京 210044)[2]
(罗格斯大学计算机科学学院 新泽西州 08854)[3]

摘 要 大多数机器学习问题可以最终归结为最优化问题(模型学习)。它主要运用数学方法研究各种问题的优化途径及方案,在科学计算和工程分析中起着越来越重要的作用。随着深度网络的快速发展,数据和参数规模也日益增长。尽管近些年来GPU硬件、网络架构和训练方法均取得了重大的进步,但单一计算机仍然很难在大型数据集上高效地训练深度网络模型,分布式近似牛顿法作为解决这一问题的有效方法之一被引入到分布式神经网络的研究中。分布式近似牛顿法将总体样本平均分布到多台计算机,减少了每台计算机所需处理的数据量,使计算机之间互相通信,共同协作完成训练任务。文中提出了基于近似牛顿法的分布式深度学习,在相同的网络中利用分布式近似牛顿法训练,随着GPU数目呈2的指数次幂增加,训练时间呈近乎2的指数次幂减少。这与研究的最终目的一致,即在保证估计精度的前提下,利用现有分布式框架实现近似牛顿法,分布式训练神经网络,从而提升训练效率。

关键词 最优化问题,近似牛顿法,分布式框架,神经网络
中图法分类号 TP181 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.07.028

Distributed Convolutional Neural Networks Based on Approximate Newton-type Mothod

WANG Ya-hui[1,2] LIU Bo[3] YUAN Xiao-tong[1,2]
(Department of Information and Control, Nanjing University of Information Science and Technology, Nanjing 210044, China)[1]
(Jiangsu Key Laboratory of Big Data Analysis Technology, Nanjing 210044, China)[2]
(Department of Computer Science, Rutgers University, New Jersey 08854, USA)[3]

Abstract Most machine learning problems can ultimately be attributed to optimization problems (model learning). It mainly uses mathematics methods to study the optimal ways and solutions for various problems and plays an increasingly important role in scientific computing and engineering analysis. With the rapid development of deep networks, the scale of data and parameters also increases. Although significant advances have been made in GPU hardware, network architecture and training methods in recent years, it is still difficult for a single computer to efficiently train deep network models on large data sets. The distributed approximation Newton-type method is one of the effective methods to solve this problem. It is introduced into the study of distributed neural networks. Distributed approximation Newton-type method distributes the average sample evenly across multiple computers, the amount of data to be processed by each computer is reduced, and computers communicate with each other to complete the training task. This paper proposed distributed deep learning based on Approximation Newton-type method. The DANE algorithm is used to train in the same network. As the number of GPUs increases exponentially by 2, the training time decreases exponentially by nearly 2. This is consistent with ultimate goal, that is, on the premise of ensuring the estimation accuracy, the existing distributed framework is used to implement the approximate Newton-like algorithm, and the algorithm is used to train the neural network in a distributed manner to improve the operating efficiency.

Keywords Optimization problem, Approximate Newton-typemethod, Distributed framework, Neural network

1 引言

最优化方法(即运筹学方法)是近几十年形成的,它主要运用数学方法研究各种系统的优化途径及方案,为决策者提供科学决策的依据。在许多机器学习问题中,往往需要研究大规模数据的组合最优化问题,最优化方法在科学计算和工程分析中起着越来越重要的作用。实践表明,随着科学技术的日益进步和生产经营的日益发展,最优化方法已成为现代

到稿日期:2018-08-02 返修日期:2018-10-10 本文受国家自然科学基金(61876090,61522308)资助。
王雅慧(1993-),女,硕士生,主要研究方向为机器学习与分布式深度学习,E-mail:20162283677@nuist.edu.cn;**刘 博**(1984-),男,博士生,主要研究方向为机器学习与计算机视觉;**袁晓彤**(1980-),男,博士后,教授,CCF会员,主要研究方向为机器学习与计算机视觉,E-mail:xtyan1980@qmail.com(通信作者)。

管理科学的重要理论基础和不可或缺的方法，被广泛地应用到公共管理、经济管理、工程建设、国防等领域，并发挥着越来越重要的作用。

在大型数据集上进行训练的现代神经网络架构在很多领域取得了良好的应用效果，这些领域包括语音和图像认知、自然语言处理、欺诈检测和推荐系统等。但是训练这些神经网络模型面临着巨大的运算量。尽管近些年来 GPU 硬件、网络架构和训练方法均取得了重大的进步，但事实是在单一机器上，网络训练所需要的时间仍然过长。因此我们迫切需要通过多计算机分布式优化学习方法来有效提升深度学习的训练速度。

在机器学习和深度学习领域，分布式的优化已经成为一种先决条件，这是因为单机已经解决不了目前快速增长的数据和参数量。分布式处理框架，不仅能用于处理大规模数据，而且能将很多繁琐的细节隐藏起来，比如，自动并行化、负载均衡和灾备管理等，这将极大地简化程序员的开发工作。在分布式系统中，机器通过网络通信来共同完成任务。而网络速度是本地内存读写的数十分之一或数百分之一，机器间的网络通信大大降低了网络通信开销。因此分布式优化的主要目的在于：1）解决单个计算节点内存不足的问题，以保证能够处理 TB 级及以上的数据量；2）利用并行加速模型训练，将原本数月的训练时间大幅缩短。

现有的分布式框架包括基于 Hadoop[1] 的 Mahout 和基于 Spark[2] 的 MLI，它们采用的 Iterative 方式都是基于 MapReduce[3] 的架构。其能够保持迭代之间的状态，并且执行策略也更加优化。但是，由于这两种方法采用的都是同步迭代的通信方式，因此它们很容易因为个别机器的低性能而使得全局性能降低。为了解决这个问题，Graphlab[4] 采用图形抽象的方式进行异步调度通信。但是它缺少了以 MapReduce 为基础架构的弹性扩展性，并且它使用粗粒度的 snapshots 来进行恢复，这两点都会阻碍可扩展性。parameter server[5] 吸取了 Graphlab 异步机制的优势，并解决了其在可扩展性方面的劣势。Mxnet[6] 是可用于分布式训练的深度学习框架，而提供分布式训练特性的正是 parameter server。本文就是利用 Mxnet 框架下 parameter server 的分布式训练特性实现各个节点之间的通信交流。

分布式优化算法的学习和研究一直是分布式研究的重点，将样本数据进行行拆分，每条数据分别存储在不同的节点上，每个节点利用本地数据进行全部模型参数训练，集群节点并行更新参数。在分布式优化中模型参数需要被所有的 worker 节点频繁访问，这就带来了很多问题和挑战：1）访问这些数据和参数，需要大量的网络带宽支持；2）很多机器学习算法都是连续型的，只有上一次迭代完成，即各个 worker 都完成之后才能进行下一次迭代，这就导致了机器之间的性能存在差距，将会造成性能的极大损失；3）许多分布式的容错能力差，从而导致收敛效果差。

近年来，由 Zinkevich 等[7] 提出的整个训练过程只有一次参数平均，而后，Zhang 等[8] 进一步分析并提出了偏差调整，但由于次优化只在每个局部节点上进行，而节点之间仅有一次交流，因此该方法最终在最小化全局目标时没有得到很好的效果。在分布式学习中模型需要平衡计算时间和交流时间，仅进行一次交流是不够的[9]。Shalev-Shwartz 等[10] 提出了多次通信次数的梯度下降的分布式实现，分布式梯度下降所需的迭代次数与总体样本大小 N 有关[11]。除了梯度下降方法，还存在一些更复杂的利用梯度信息的方法，例如 Najafabadi 等[12] 提出了使用 L-BFGS 的分布式实现，但是不能保证其效果优于梯度下降。另一种流行的方法是 ADMM[13]，其中机器以分布式方式交替计算共享的对偶变量，并解决了关于其本地数据的增广拉格朗日问题。但是，ADMM 的收敛速度可能会很慢。尽管最近的工作在有利的假设下证明了线性收敛速度[14-15]，但没有与迭代次数和总体样本数相关的分析。

本文提出的近似牛顿法是一种基于牛顿法的优化算法，在每次迭代中利用目标函数及其一阶导能够达到快速收敛的效果。梯度下降法主要利用目标函数的局部性质，即求解目标函数的一阶导数时，梯度下降法具有一定的“盲目性”。相比于牛顿法，近似牛顿法避免了 Hessian 矩阵的计算，减少了计算量，同时保持了牛顿法的快速收敛速率，不用二阶偏导数即可构造出可近似海森矩阵的正定对称阵，因此近似牛顿法是一类比较有效的算法。DANE 方法是一个多交流的分布式方法，适用于很多凸优化问题，其将总体样本平均分给多个节点，这样每台机器会更加均衡地进行计算，而最终实现收敛的交流次数与机器个数相关。

2　分布式近似牛顿法

2.1　分布式优化

在分布式优化问题中，我们考虑到有 m 个机器，而每个机器都拥有一个局部损失函数，因此最终目标是最小化它们的平均值，即：

$$\emptyset(w)=\frac{1}{m}\sum_{i=0}^{m}\emptyset_i(w) \tag{1}$$

在随机优化设置中，最终目标是最小化一些随机目标（如期望损失和正则化误差）：

$$F(w)=E_{z\sim D}[f(w,z)] \tag{2}$$

设有 m 台机器，而且每个机器都拥有 n 个样本，第 i 个机器的样本在 $z_i^1,\cdots,z_i^n$ 在分布源 D 中是独立同分布的，因此有 $N=nm$ 个独立样本在 m 个机器中均匀且随机分布，每个机器可以构建一个局部实验估计 $F(w)$：

$$\emptyset_i(w)=\hat{F}_i(w)=\frac{1}{n}\sum_{j=1}^{n}f(w,z_i^j) \tag{3}$$

整体的经验目标是：

$$\emptyset(w)=\hat{F}(w)=\frac{1}{m}\sum_{i=1}^{m}\hat{F}_i(w)=\frac{1}{nm}\sum_{i,j}f(w,z_i^j) \tag{4}$$

然后，将经验风险最小化值：

$$\hat{w}=\arg\min\hat{F}(w) \tag{5}$$

作为 $F(w)$ 的近似最小值，也就是将随机优化设置的函数最终转化为 $\hat{w}=\arg\min\emptyset(w)$ 的表达方式，因此优化目标 $\emptyset(w)$

不是一个随机目标的变化而是一个计算随机目标的近似值。

在考虑分布式优化时，我们认为有两种资源在发挥作用：1)每台机器上的处理量；2)各机器之间的通信。本文重点关注在每台机器上局部优化过程之间交替的算法，以及涉及简单的 map-reduce 操作(如 R^d 中向量的分布式平均)的通信回合。由于分布式的交流成本在实践中非常高[16]，因此如何使用最小数量的迭代来开发快速优化经验目标$\hat{F}(\cdot)$的方法是我们的目标。

2.2 近似牛顿法的优化

近似牛顿法可以看作一种类似于牛顿法的方法，在每次迭代时，我们采取了一个问题几何的步骤，而不是一个梯度步骤。特别地，对于二次目标，该方法可以看作牛顿法，其中每台机器 i 隐式使用其局部 Hessian $\nabla^2 \mathcal{O}_i(w)$(尽管没有显式计算 Hessians)。与交替方向乘子法(ADMM)不同的是，本文所提方法可以利用以下事实：对于机器学习应用，其子问题通常是相似的，即 $\mathcal{O}_i \approx \mathcal{O}$。我们将这种方法称为 DANE[17]——分布式近似牛顿法。

近似牛顿法适用于任何平滑和强凸的问题。然而，正如牛顿和类似牛顿的典型方法，它的一般分析并不是很明显。对于一般函数，我们可以显式收敛，但不能严格证明基于梯度下降的提升。相反，为了证明 DANE 的优势并给出其好处，我们将目光集中在二次目标 $f(w,z)$ 上，其中二次目标是 L-光滑和 λ-强凸。当 L/λ 固定，并且每台机器的样本 n 很大(Zhang 等[18]于 2013 年考虑的方案)时在固定的迭代/通信轮次数下建立收敛。对于分布式近似牛顿法，当 λ 等于 $1/nm$ 时，在多次迭代中产生经验最小值的收敛，这些迭代与机器的数量 m 大致呈线性关系，但与样本大小 $N=nm$ 无关。近似牛顿法是第一个可证明具有递代次数与机器数 m 相关的算法。

2.3 分布式近似牛顿法

本文描述了一种新的分布式优化迭代方法。该方法每次迭代执行两次分布式平均计算，并输出预测变量 $w^{(t)}$，该变量在合适的参数选择下收敛于最优$\hat{w}$，我们称之为分布式近似牛顿法(DANE)。算法 1 描述了分布式近似牛顿法的主要步骤。

算法 1 分布式近似牛顿法

参数：学习率 $\eta>0$，正则化项 $\mu>0$

初始化：从 $w^{(0)}$ 开始，例如 $w^{(0)}=0$

FOR t=1,2,…

计算$\nabla\mathcal{O}(w^{(t-1)})=\frac{1}{m}\sum_{i=1}^{m}\nabla\mathcal{O}_i(w^{(t-1)})$并将结果分发给所有机器

每个机器 i 解决：

$$w_i^{(t)}=\arg\min_w\left[\mathcal{O}_i(w)-(\nabla\mathcal{O}_i(w^{(t-1)})-\eta\nabla\mathcal{O}(w^{(t-1)}))^{\mathrm{T}}w+\frac{\mu}{2}\|w-w^{(t-1)}\|_2^2\right]$$

计算 $w^{(t)}=\frac{1}{m}\sum_{i=1}^{m}w_i^{(t)}$并将结果分发给所有机器

END

分布式近似牛顿法维护一致性的迭代 $w^{(t)}$，它在每次计算结束时在所有机器中同步。在每次迭代中，我们首先通过平均局部梯度 $\nabla\mathcal{O}_i(w^{(t-1)})$ 来计算当前迭代的梯度 $\nabla\mathcal{O}(w^{(t-1)})$。然后，每台机器根据其自身的局部目标 $\mathcal{O}_i(w)$ 和计算的全局梯度$\nabla\mathcal{O}(w^{(t-1)})$执行单独的局部优化，以获取局部迭代 $w_i^{(t)}$。这些局部迭代被平均以获得集中迭代 $w^{(t)}$。

该方法的关键是在每次迭代时在每台机器上执行局部优化：

$$w_i^{(t)}=\arg\min_w\left[\mathcal{O}_i(w)-(\nabla\mathcal{O}_i(w^{(t-1)})-\nabla\mathcal{O}(w^{(t-1)}))^{\mathrm{T}}w+\frac{\mu}{2}\|w-w^{(t-1)}\|_2^2\right] \tag{6}$$

为了理解这个局部最优化问题，回想一下对应于一个强凸函数 φ 的 Bregman 散度定义：

$$D_\varphi(w';w)=\varphi(w')-\varphi(w)-\langle\nabla\varphi(w),w'-w\rangle \tag{7}$$

现在，对于每个局部目标 $\mathcal{O}_i$，考虑正则化的局部目标，并将其定义为：

$$h_i(w)=\mathcal{O}_i(w)+\frac{\mu}{2}\|w\|^2 \tag{8}$$

它的相应 Bregman 散度为：

$$D_i(w';w)=D_{h_i}(w';w)=D_{\mathcal{O}_i}(w';w)+\frac{\mu}{2}\|w'-w\|^2 \tag{9}$$

因此，本地优化问题 $w_i^{(t)}$ 可以写成：

$$w_i^{(t)}=\arg\min_w\mathcal{O}(w^{(t-1)})+\langle\nabla\mathcal{O}(w^{(t-1)}),w-w^{(t-1)}\rangle+\frac{1}{\eta}D_i(w;w^{(t-1)}) \tag{10}$$

这里还添加了不依赖于 w 且不影响优化的项 $\mathcal{O}(w^{(t-1)})+\langle\nabla\mathcal{O}(w^{(t-1)}),w^{(t-1)}\rangle$，因此，式(10)的前两项是关于当前迭代 $w^{(t-1)}$ 的整体目标 $\mathcal{O}(w)$ 的线性近似，并且不依赖于机器 i。每台机器的不同之处为用于定位线性逼近的隐藏函数，上式实际上是一个利用隐藏函数 h_i 和步长 η 的镜像下降更新。

3 基于 DANE 的分布式深度训练

DANE 算法由于具有通信高效的性质，被广泛应用于凸优化问题，并且取得了良好的效果。但 DANE 算法是用来解决最优化凸问题的，对于非凸优化问题的解决还未做尝试。本文基于 DANE 做分布式深度训练，利用 DANE 算法通信高效的性质来解决神经网络的非凸优化问题。最后在可用于分布式训练的深度学习框架 MXNET 下进行实验。

MXNet 是一款具有效率和灵活性的深度学习框架。它允许混合符号编程和命令式编程，从而最大限度地提高效率和生产力。其核心是一个动态的依赖调度，它能够自动并行符号和命令的操作。其包含一个图形优化层，能加快符号的执行速度，提高内存的使用效率。这个库便携、轻量，而且能够扩展到多个 GPU 和多台机器。多设备间的数据交互是通过 KVStore 进行的，而 KVStore 的实现是基于 parameter server[19]的。

parameter server 包含两种节点：worker 和 server。每个 server 节点只负责分到的部分参数，每个 server 可以与其他

server 节点通信，servers 共同维持一个全局的共享参数；每个 worker 节点也只分到部分数据和处理任务，但 worker 节点之间没有通信，只能与对应的 server 节点进行通信。workers 跟 servers 之间通过 push 和 pull 通信。worker 通过 push 将计算好的梯度发送到 server，然后通过 pull 将 server 更新的参数发送给 workers。

本文基于 DANE 算法分布式地训练神经网络，其中 $\emptyset(w)$ 为整体目标函数，$\emptyset_i(w)$ 为局部目标函数。首先，每个 worker 节点基于本地数据前向传播经过神经网络得到的局部梯度 $\nabla\emptyset_i(w)$，每个 worker 节点通过 push 将计算好的梯度发送给 server 节点，在 server 节点中通过求和取平均值得到全局梯度 $\nabla\emptyset(w)$，然后每个 worker 节点通过 pull 从 server 节点获取全局梯度 $\nabla\emptyset(w)$。每个 worker 节点利用式(6)根据 server 节点发送的全局梯度进行计算，然后经过神经网络后向传播更新局部参数，再通过 push 将局部参数 w_i 发送给 server 节点，在 server 节点中通过求和取平均得到最终的全局参数 w。最后每个 worker 节点通过 pull 从 server 节点获取全局参数 w，并进入下一次迭代。worker 节点和 server 节点的交流过程如图 1 所示。

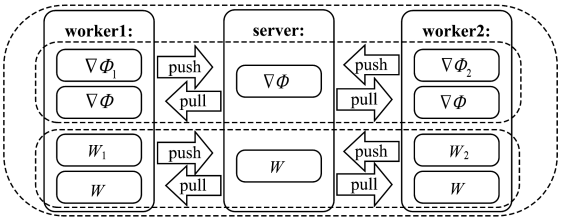

图 1　workers 和 servers 之间的通信流程

Fig. 1　Communication process between workers and servers

4　实验

本文首先呈现的是采用分布式近似牛顿算法进行 Matlab 仿真的初步实验，然后在 Linux 系统下，利用分布式框架 parameter server 实现分布式近似牛顿法。

4.1　算法仿真

我们先利用一个合成的数据集来解决一个简单的二次平方式问题，合成数据集的所有参数都已知。我们对分布式牛顿法进行算法仿真，根据模型 $y=\langle x,\boldsymbol{w}^*\rangle+\varepsilon$，产生 N 个独立同分布训练样本 (x,y)，$x\sim N(0,1)$，$\varepsilon\sim N(0,1)$，这里 x 是一个 500 维的数据，$\boldsymbol{w}^*$ 是全一向量。给定一组样本 (x,y)，假设它们被随机分配到不同的机器，然后用分布式近似牛顿法来解决标准岭回归问题：$\min_w\frac{1}{N}\sum_{i=1}^{N}(\langle x,w\rangle-y)^2+0.005w^2$。

图 2 显示了基于总数为 N 的合成数据集，在不同数目的机器 m 上算法的收敛行为。我们分别在 1 台、2 台、4 台和 8 台机器上进行了分布式近似牛顿法的仿真实验，图 2(a)－图 2(c)给出了在 6000×500，10000×500，14000×500 的 3 个合成数据集上进行分布式优化的结果，横坐标表示机器个数(进程数)，纵坐标表示时间。从实验结果中发现，当增加进程数时，分布式近似牛顿法所需时间明显减少，但该时间并不像预期的那样按比例减少。图 2(d)给出样本矩阵为 18000×500 时的实验结果，可以发现：造成上述现象的主要原因是部分进程的运算速度较慢，从而延长了整个估计过程的时间。这是由每组数据在计算梯度以及更新局部参数时计算量的不同造成的；并且程序实现过程中，存在 server 节点对全局梯度和参数进行计算的程序，这部分程序是无法进行并行的，这也是耗时没有按理想比例下降的主要原因之一。

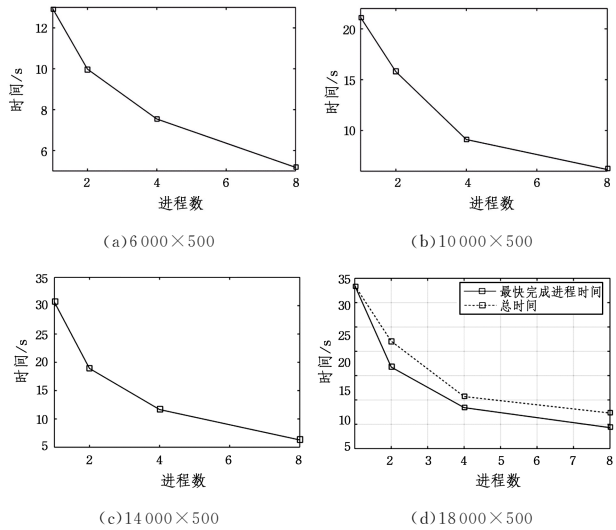

图 2　4 组合成数据的并行加速曲线

Fig. 2　Parallel acceleration curves of four groups synthetic data

4.2　真实数据集实验

每个节点利用本地数据对分配的参数进行更新，集群节点并行更新参数。采用两个真实数据集 MNIST 和 Cifar10。MNIST 是一个基本的手写字体识别数据集，MNIST 训练集由 250 个人手写的数字构成，其中 50% 是高中学生，50% 来自人口普查局的工作人员；MNIST 的测试集也是同样比例的手写数字数据，包含 50000 个训练样本、10000 个测试样本和对应的两个 label 集，共有 10 种 label 类型。

Cifar10 是一个包含 10 种 label 类型的图像数据集，该数据集共有 60000 张彩色图像，这些图像的大小为 32 * 32，并被分为 10 类，每类 6000 张图。这里将其中 50000 张图用于训练，另外 10000 张图用于测试。用于测试的数据取自 10 类中的每一类，且每一类随机取 1000 张。剩下的随机排列组成训练集。

在不同的分布式计算模型下，优化算法通常可以分为数据划分和模型划分。数据划分是指将样本数据进行行拆分，每条数据分别存储在不同的节点上，每个节点利用本地数据进行全部模型参数训练，集群节点并行更新参数。模型划分是指将模型参数进行列拆分，不同列的数据分别存储在不同的节点上，本文实验就是一个数据划分的分布式优化算法训练的实验。实验中使用了一个 MXNET 计算机集群，在 MXNET 计算机集群中分别用 1，2，3 台机器进行训练实验，每台机器都拥有 2 个 gpu，本文基于 DANE 算法分别分布式训练一个 3 层和一个 5 层的神经网络。

首先分别将 MNIST 数据集里的训练样本和测试样本分

别拆分成 2,4,6 条数据,由于 MNIST 是一个 0－9 手写数字识别的数据集,我们将网络输入设为大小为(1,32,32)的单通道灰度图,批量大小设为 100。Cifar10 是一个包含 10 类图像的图像数据集,要将 Cifar10 数据集输入训练网络,首先要把文件转换成.jpg 格式的样本,然后根据样本生成.list 文件,再用.list 文件生成.rec 格式的文件,为了不丢失颜色信息,我们需要将网络输入设为大小为(3,32,32)的三通道彩色图,批量大小设为 100。图 3、图 4 分别给出了在 MNIST 数据集下经过 Lenet3,Lene5 网络的训练结果,图 5、图 6 分别给出了在 Cifar10 数据集下经过 Lenet3,Lenet5 的训练结果。图中 x 轴是 gpu 的个数,y 轴是时间。

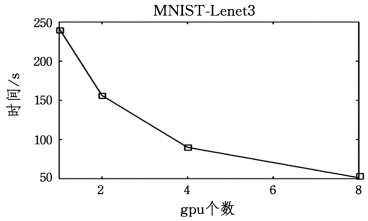

图 3 MNIST-Lenet3 的实验结果

Fig. 3 Experimental results of MNIST-Lenet3

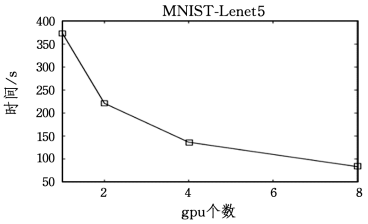

图 4 MNIST-Lenet5 的实验结果

Fig. 4 Experimental results of MNIST-Lenet5

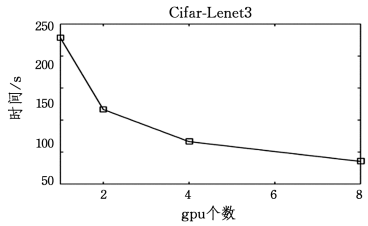

图 5 Cifar10-Lenet3 的实验结果

Fig. 5 Experimental results of Cifar10-Lenet3

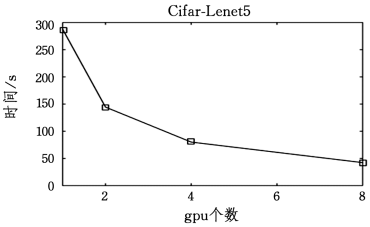

图 6 Cifar10-Lenet5 的实验结果

Fig. 6 Experimental results of Cifar10-Lenet5

将 DANE 算法与基础的分布式算法进行比较。在 MNIST 数据集上进行实验,随着 gpu 个数的增加,训练时间不断减少。随着迭代次数的增加,平均损失逐渐减少。将 DANE 算法与 ADMM 以及实际平均损失最小值的结果进行比较,结果如图 8 所示,其中横轴表示迭代次数,纵轴表示测试集的平均损失。可以看出:DANE 可以迅速收敛到最优值,而 ADMM 所需的迭代次数明显增加。

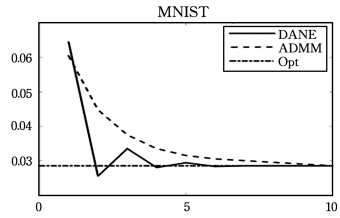

图 7 算法准确度之间比较

Fig. 7 Comparison between accuracy of algorithms

通过多次实验可以获得训练模型,接着用测试集对训练模型进行测试,得到数据集训练的准确度,MNIST 数据集和 Cifar 数据集训练的精度分别如表 1 和表 2 所列。可以看出:MNIST 数据集经过 Lenet3 训练的准确度低于 Lenet5 训练的准确度,这说明卷积神经网络越复杂,准确度越高。同时 Cifar10 数据集的准确度也达到了当前最新算法的准确度。

表 1 MNIST 样本精度分析

Table 1 Sample accuracy analysis of MNIST

gpu	2	4	6	8
MNIST-Lenet3	0.983100	0.981600	0.981300	0.981000
MINST-Lenet5	0.991400	0.989700	0.989100	0.988700

表 2 Cifar10 样本精度分析

Table 2 Sample accuracy analysis of Cifar10

gpu	2	4	6	8
Cifart-Lenet3	0.882500	0.874500	0.866700	0.861300
Cifar-Lenet5	0.890600	0.887500	0.886300	0.885400

结束语 针对大规模数据集的深度网络模型的训练,目前已有很多学者提出了多种提高其性能的方法。传统的计算模型都是基于单一机器的,由于这些模型的训练时间长、效率低,已无法满足实际需求。本文基于 DANE 算法对神经网络进行分布式训练。在对这种方法进行推理分析的过程中,不难发现数据的独立性,这是分布式优化的前提和基础。该分布式算法具有收敛快、稳定性好、通信开销小的优点。在合成数据和真实数据上的实验结果充分表明:在不影响结构估计准确率的前提下,本文提所提方法可以很好地实现分布式数据的处理以及高效并行计算。同时,parameter server 并行框架的快速发展也使得本文的研究得到保障。

参 考 文 献

[1] GANDHI A, THOTA S, DUBE P, et al. Autoscaling for Hadoop Clusters[C]// IEEE International Conference on Cloud Engineering. IEEE, 2016: 109-118.

[2] YUAN Y, SALMI M F, YIN H, et al. Spark-GPU: An accelerated in-memory data processing engine on clusters[C]// IEEE International Conference on Big Data. IEEE, 2017: 273-283.

[3] SAMADDAR S, SINHA R, DE R K. A MODEL for DISTRIB-

UTED PROCESSING and ANALYSES of NGS DATA under MAP-REDUCE PARADIGM[J]. IEEE/ACM Transactions on Computational Biology & Bioinformatics,2018,PP(99):1.

[4] NASR M M,SHAABAN E M,HAFEZ A M. Building Sentiment analysis Model using Graphlab[J]. International Journal of Scientific & Engineering Research,2017,8(6):1155-1160.

[5] JIANG J,CUI B,ZHANG C,et al. Heterogeneity-aware Distributed Parameter Servers[C]// ACM International Conference. ACM,2017:463-478.

[6] CHEN T,LI M,LI Y,et al. Mxnet:A flexible and efficient machine learning library for heterogeneous distributed systems[J]. arXiv preprint arXiv:1512. 01274,2015.

[7] ZINKEVICH M,WEIMER M,SMOLA A J,et al. Parallelized-Stochastic Gradient Descent[C]// Advances in Neural Information Processing Systems 23,Conference on Neural Information Processing Systems 2010. DBLP,2010:2595-2603.

[8] ZHANG Y,DUCHI J C,WAINWRIGHT M J. Communication-efficient algorithms for statistical optimization[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. ,2012:1502-1510.

[9] GUPTA S,ZHANG W,WANG F. Model Accuracy and Runtime Tradeoff in Distributed Deep Learning:A Systematic Study [C]// IEEE,International Conference on Data Mining. IEEE,2017:171-180.

[10] SHALEV-SHWARTZ S,SHAMIR O,SREBRO N,et al. Stochastic convex optimization[C]// Annual Conference on Learning Theory. 2009.

[11] SRIDHARAN K,SHALEV-SHWARTZ S,SREBRO N. Fast rates for regularized objectives[C]// Advances in Neural Information Processing Systems. 2009:1545-1552.

[12] NAJAFABADI M M,KHOSHGOFTAAR T M,VILLANUSTRE F,et al. Large-scale distributed L-BFGS[J]. Journal of Big Data,2017,4(1):22.

[13] ERSEGHE T. Distributed Optimal Power Flow Using ADMM [J]. IEEE Transactions on Power Systems,2014,29(5):2370-2380.

[14] TAYLOR G,BURMEISTER R,XU Z,et al. Training neural networks without gradients:a scalable ADMM approach[C]// International Conference on International Conference on Machine Learning. JMLR. org,2016:2722-2731.

[15] WANG Y,YIN W,ZENG J. Global convergence of ADMM in nonconvex nonsmooth optimization[J]. Journal of Scientific Computing,2015(1-2):1-35.

[16] FENG X,CHANG L,LIN X,et al. Distributed computing connected components with linear communication cost[J]. Distributed and Parallel Databases,2018,36(3):555-592.

[17] SHAMIRO,SREBRO N,ZHANG T. Communication-efficient distributed optimization using an approximate Newton-type method[C]// International Confe-rence on International Conference on Machine Learning. JMLR. org,2014:II-1000.

[18] ZHANG Y,WAINWRIGHT M J,DUCHI J C. Communication-efficient algorithms for statistical optimization[C]// Advances in Neural Information Processing Systems. 2012:1502-1510.

[19] LI M. Scaling Distributed Machine Learning with the Parameter Server[C]// International Conference on Big Data Science and Computing. ACM,2014:3.

[20] CHAUDHARI P,BALDASSI C,ZECCHINA R,et al. Parle:parallelizing stochastic gradient descent[J]. arXiv preprint arXiv:1707. 00424,2017.