文章编号:1001-506X(2019)05- -07 网址:www. sys-ele. com

基于 Bagging-Down SGD 算法的分布式深度网络

秦 超[1],高晓光[1],陈大庆[2]
(1. 西北工业大学电子信息学院,陕西 西安 710100;2. 南岸大学,英国 伦敦 SE10AA)

摘 要:通过对大量数据进行训练并采用分布式深度学习算法可以学习到比较好的数据结构,而传统的分布式深度学习算法在处理大数据集时存在训练时间比较慢或者训练精度比较低的问题。提出 Bootstrap 向下聚合随机梯度下降(Bootstrap aggregating-down stochastic gradient descent,Bagging-Down SGD)算法重点来提高分布式深度网络的学习速率。Bagging-Down SGD 算法通过在众多单机模型上加入速度控制器,对单机计算的参数值做统计处理,减少了参数更新的频率,并且可以使单机模型训练和参数更新在一定程度上分开,在保证训练精度的同时,提高了整个分布式模型的训练速度。该算法具有普适性,可以对多种类别的数据进行学习。

关键词:深度网络;分布式;Bootstrap 向下聚合随机梯度下降;速度控制器
中图分类号:TP 301.6 **文献标志码**:A **DOI**:10.3969/j.issn.1001-506X.2019.05.13

Distributed deep networks based on Bagging-Down SGD algorithm

QIN Chao[1], GAO Xiaoguang[1], CHEN Daqing[2]
(*1. School of Electronics and Information, Northwestern Poly-technical University, Xi'an 710100, China; 2. London South Bank University, London SE10AA, England*)

Abstract: As a cutting-edge disruptive technology, deep learning and unsupervised learning have attracted a significant research attention, and it has been widely acknowledged that training big data with a distributed deep learning algorithm can get better structures. But there are two main problems with traditional distributed deep learning algorithms: the speed of training is slow and the accuracy of training is low. The Bootstrap aggregating-down stochastic gradient descent (Bagging-Down SGD) algorithm is proposed to solve the speed problem mainly. We add a speed controller to update the parameters of the single machine statistically, and to split model training and parameters updating to improve the training speed with the assurance of the same accuracy. It is to be proved in the experiment that the algorithm has the generality to learn the structures of different kinds of data.

Keywords: deep network; distributed; Bootstrap aggregating-down stochastic gradient descent (Bagging-Down SGD); speed controller

0 引 言

针对迅速扩大的数据规模,机器学习的算法结构,从原来的浅层,到现在的深层,可以通过大数据集的实验看出深层结构相对于浅层结构的优势[1-2]。但深度学习仍存在着诸多不足,其中最大的问题是在训练的时候初始值难以确定,使利用反向传播梯度下降的方法训练容易进入局部极值,导致训练结果不如预期。2006 年,Hinton[3] 提出对深层网络的新训练算法,将深度网络进行预训练(无监督)来确定初始值,进而进行整体的监督训练。由此开始,对深度网络的研究探讨成为国内外重要机器学习会议或者期刊的主要内容,有关深度结构的机器学习的算法得到巨大发展并且逐步完善。尤其是深层结构的基本单元方面,受限玻尔兹曼机[4-5] 和自适应编码[6-7] 已经比较成熟,使得二十世纪 80 年代提出的卷积神经网络[8] 作为深层结构也借势重新焕发青春。

在深度概念的基础上,学者们做了结构上的改进,使其能更好地适应现如今逐渐扩大的数据集。一部分学者选择加深网络结构[6],使整个深度网络在不过拟合的前提下可学到更确切的数据内容。输入数据的维数和规模增大,相应的数据结构所需的节点也逐渐增多,通过深度结构的加深可比较准确地得到分类(回归)结果[1, 9-10]。但这种做法对单个机器的性能提出了很高的要求[11]。另一部分学者希望可以将深度结构进行分布化[12-14],将整个深度网络运行在具备一定拓扑结构的多计算机上来协同完成整个网络的训练,但其在做结构优化的同时缺少考虑时间损耗和因

收稿日期:2018-01-04; **修回日期**:2019-02-21; **网络优先出版日期**:2019-03-21。
网络优先出版地址:http://kns.cnki.net/kcms/detail/11.2422.TN.20190321.1527.002.html
基金项目:国家自然科学基金(61573285)资助课题

单机模型运算速度慢而拖慢整个结构的运算速度问题。

在研究分布化处理深度结构这个问题时，有两类算法。一部分学者放宽了同步性需求，采取延时梯度下降法来解决该问题[15-16]，但同步性不能得到满足就不可避免地出现一些重复训练，从而导致速度变慢。另外一些学者通过稀疏梯度下降法来解决在共享内存结构（单机）下的 lock-less 同步随机梯度下降问题[17-18]，将并行计算放在一系列单隐层模型上，通过逐层训练来降低模型的训练复杂度，但该算法只是针对一些特定的稀疏性和凹凸性的模型，并不能处理大规模计算问题。一般而言，得到一个良好的大规模计算结构需要考虑计算效率和模型计算方法的普适性[19]。

针对这两个主要的分布式训练算法，学者们的主要工作是通过制定一些指标，对两个模型进行适当选择[15-16,18,20]。提出新的大规模分布式深度学习算法，即 Bootstrap 向下聚合随机梯度下降（Bootstrap aggregating-down stochastic gradient descent，Bagging-Down SGD）算法。对于普适性和计算效率两个问题，Bagging-Down SGD 算法采取不改变单机训练算法结构，而只是对每次更新参数做优化的方法来解决。Bagging-Down SGD 算法在计算节点上加入了速度控制器使单机模型训练和参数更新在一定程度上分开，在保证大数据集训练精度同时，重点研究如何提高模型的学习速率。且该算法具有普适性，可以很好地训练不同的数据类型。

实现分布式学习算法的时候，需要考虑所使用的工具箱。现在主要基础工具箱有两种，MapReduce[21] 和 GraphLab[22]。它们在处理并行化方面都比较优秀，但 MapReduce在处理迭代计算方面的不足导致其在深度学习模型中的用处不大。目前流行的高度集成的分布式深度平台有 TensorFlow[23-24]，Mxnet，pytorch，Caffe 等[25]，学者们主要使用的分布式深度结构平台为 TensorFlow 和 Mxnet。TensorFlow 具有高度的代码集成程度，并对分布式之间的数据交流的方式做了一些优化，大幅度提升了数据交流的速率；Mxnet 在代码集成度方面不如 TensorFlow，但其在商业领域已经得到广泛使用，并且其对异构大数据集的支持度比较好。由于当前现存两种主要的分布式深度结构算法主要应用在 Graphlab 上，并且 GraphLab 作为为图模型计算而生的工具箱，可以很好地处理分布式深度模型问题，应用该工具箱实现深度模型的并行化计算。

对于计算所需要的底层结构，主要有两种，中央处理器（central process unit，CPU）和图形处理器（graphic process unit，GPU）[3-4,6,12]。GPU 对比 CPU 来说有着速度上的巨大优势，在现在的大数据环境中，随着 GPU 的广泛使用，使得深度学习训练时间大大缩短[12]。选择 GPU 作为处理计算的底层结构。

1 模型的并行化概念

针对模型并行化处理，使用图 1 的拓扑结构。对需要大规模处理数据的模型而言，并行化处理需要将不同的节点放入不同的机器中进行计算。这个分布式结构可以在不同计算机中自动并行化处理数据，将效率最大化，并可以提高训练的精度。

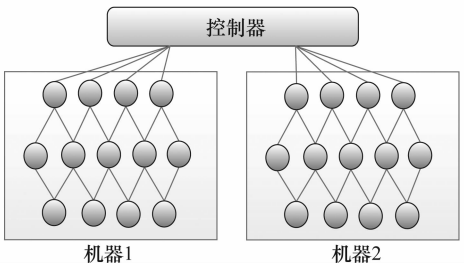

图 1 分布式模型的拓扑结构图

Fig. 1 Topological structure of distributed model

2 分布式优化算法 Bagging-Down SGD

2.1 分布式优化算法

通过并行化计算方法，可以更有效地训练大规模深度模型。考虑到训练时间，需要协调多个模型同时计算，不只是简单的模型之间互联，还需要优化算法对模型之间的速度差异进行整体控制。整个分布式模型需要由多个单机模型组成，具体来说，需要考虑 4 个问题[20,26-27]：

（1）单机模型的结构是否一样[28-29]；

（2）单机模型的输入训练集是否一样[30-32]；

（3）单机模型之间的速度调配问题[33-34]；

（4）普适性问题，即输入训练集是否需要满足一定的条件。

2.2 Bagging-Down SGD

针对第 2.1 节提出的 4 个问题，设计了 Bagging-Down SGD 算法。该算法在原随机梯度下降算法基础上加入离线批处理步骤，使其可以适应多机离线训练，对已有输入训练集进行分布式训练。该算法每个单机模型都是相同的（实验中采用 AutoEncoder 模型），运用统计抽样的方法将输入数据分出几个训练集（输入参数集不相同）。运用速度控制器进行速度的调配。并对输入参数集没有其他限制条件，算法普适性较好。

SGD 是训练神经网络最常用的方法之一[9,35-36]。传统的 SGD 在计算顺序方面的严格性和大数据处理时的时序要求，导致其在处理大数据集方面表现乏力。Bagging-Down SGD 算法是在 SGD 算法的基础上加入了 Bagging（Bootstrap aggregating 抽样）控制器，采用顶层计算权值，向下（down）输出到各单机模型中更新权值的策略进行速度调配，来更好地处理大数据集问题。另外，Bagging-Down SGD 对输入数据集没要求，所以算法普适性方面比较好。

Bagging-Down SGD 算法结构图如图 2 所示。

该算法的主要流程：拿到大数据集后，通过 Bootstrap 方法将数据集 $\mathcal{L}=\{(y_n,x_n),n=1,2,\cdots,N\}$ 采样成不同的训练子集 $L_k=\{(y_n,x_n),n=k_1,k_2,\cdots,k_m\}$，并将其作为单机模型的输入。模型和速度控制器进行参数交流，每个模

型需要向控制器索取当前的参数值 W'，而将通过模型应用 SGD 算法计算得到的更新量 ΔW 输送给控制器，控制器负责对单机模型计算量进行处理，该部分算法的结构如图 3 所示。

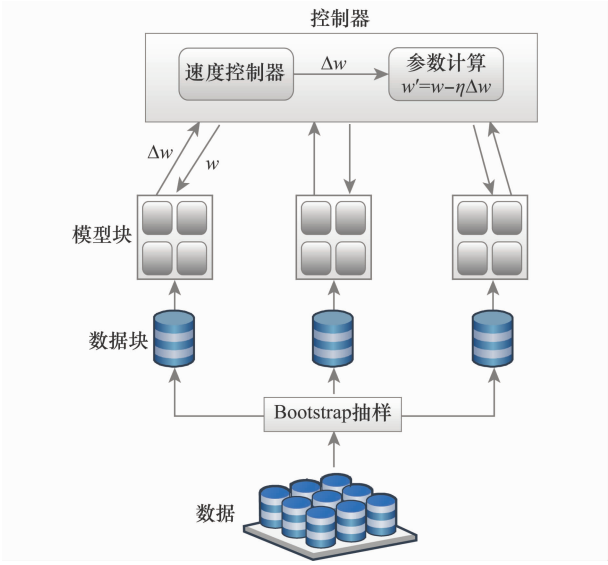

图 2　Bagging-Down SGD 算法结构

Fig. 2　Algorithm structure of Bagging-Down SGD

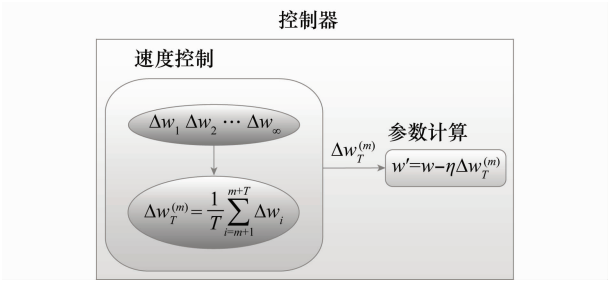

图 3　速度控制器部分算法结构

Fig. 3　Partial algorithmic structure of speed controller

Bagging 速度控制器收集模型传递过来的参数，生成一个不断增加的模型参数序列(ΔW_1，ΔW_2，…，ΔW_∞)，控制器从头开始抽取前 T 个数据，求其平均值，输入到参数计算模块计算更新后的 W'。每次计算完之后，将这 T 个数据删除，便于下一次抽取不同的值。而 Bagging 速度控制器可以适当改进因单机模型运算速度慢而拖慢整个结构的运算速度问题。

在每一次算法迭代中，每一个单机模型需要向控制器请求最新的 W'。并且在 Bagging-Down SGD 算法中每一个单机模型都是交叉同步进行，只需要与自己的训练子集进行数据交换，使得模型之间相对独立，大大减少整个结构的通讯负载进而降低整个分布式模型的运算速度。在单机模型得到新的参数值 W'后，计算梯度下降的变化值 ΔW，并将其输送给速度控制器，进行下一次的迭代计算。

Bagging-Down SGD 算法比普通的同步 SGD 算法稳定性更强，对输入大数据集的噪声鲁棒性更好。对于普通的同步 SGD 算法来说，单机模型训练出现错误，会导致整个分布式模型的训练出现延迟甚至停止[40]；通过 Bagging-Down SGD 算法进行训练，一台参数训练机器出现错误，不会影响整体的运算进度和运算速度。Bagging-Down SGD 算法里的模型不同步训练方法，使得优化过程会更加稳定。参数更新和单机模型训练的不同步也会带来一些问题。由于每个单机模型得到的参数值都有一定的滞后，而且每个单机模型得到的参数值不一定相同，因此不能保证每个单机模型得到的参数都是经过相同的计算之后得到的。而且，因为 W'参数的输出和 ΔW 参数的更新在控制器不同的进程，也会因为处于不相同的时间迭代步骤而产生问题。这些都是一些非凸函数问题，在实际使用中可以通过改变 Bagging 速度控制器里的 T 参数，将影响尽量降低。

考虑可以通过调节速度控制器里面的参数 η(学习速率)来提高 Bagging-Down SGD 算法的鲁棒性[26]。使用 $\eta_{i,k}$ 来代替传统的 η，其中 i 表示第 i 个参数，k 表示第 k 迭代。$\eta_{i,k}$ 的计算式为

$$\eta_{i,k} = Y/\sqrt{\sum_{j=1}^{K}\Delta W_{i,j}{}^2} \tag{1}$$

式中，Y 是定值，表征模型的规模因子。这个参数的改进增加了分布式模型的可扩展性，可以同时容纳更多的单机模型和更大的输入数据集。

2.3　Bagging-Down SGD 算法有效性

由于 Bagging-Down SGD 算法只是对权值进行更新，而速度控制器是整个算法权值更新的核心，所以整个算法的有效性可以用单机模型的有效性和速度控制器的有效性来进行说明。

单机模型选取的 Autoencoder 算法，文献[6]已经说明其有效性。下面对速度控制器的有效性进行证明。

假设(ΔW，x)取自分布 P 且相互独立，$\Delta\varphi(x,\mathcal{L})$为观测值。令 $\Delta\varphi_A(x)=E_{\mathcal{L}}\Delta\varphi(x,\mathcal{L})$，表示 Bagging 速度控制器最终更新的 ΔW，则有

$$E_{\mathcal{L}}(\Delta W-\Delta\varphi(x,\mathcal{L}))^2 = \Delta W^2-2\Delta W\Delta\varphi(x,\mathcal{L})+E_{\mathcal{L}}\Delta\varphi^2(x,\mathcal{L}) \tag{2}$$

令 $E_{\mathcal{L}}\Delta\varphi(x,\mathcal{L})=\Delta\varphi_A(x)$，并且应用不等式 $EZ^2\geqslant(EZ)^2$，可得

$$E_{\mathcal{L}}(\Delta W-\Delta\varphi(x,\mathcal{L}))^2\geqslant(\Delta W-\Delta\varphi_A(x))^2 \tag{3}$$

从式(3)可以看出，在大多数情况下，通过 Bagging-Down SGD 算法中的速度控制器计算得出$\Delta\varphi_A(x)$的均方误差比没有经过该控制器计算的均方误差要小，这表明 Bagging 速度控制器可以有效选择比较优的 ΔW 参数值，由此证明算法能够有效地对大规模数据集进行学习。

3　实验与分析

3.1　实验设计和参数选择

实验选择稀疏自动编码器作为每个分模块的学习模型。选择稀疏自适应编码器有比较明显的优点，就是在处理数据时，可以将隐层节点尽可能地多，相应地，其隐层层

数就会比较少。验证时采用 3 台性能相同的电脑进行子模型的学习(每一台计算机为一个输入节点,训练一个子模型),而另加一台性能较优的电脑作为速度控制器(控制节点)。每一个计算节点(子模型)一共有 3 层,输入层有 28×28 个输入结点,隐层有 500 个结点,因为稀疏自适应的结构对称性,所以最终输出层也是 28×28 个结点。输入训练集采用 MINIST 手写字体库。针对 MINIST 手写字体数据库,采用 bootstrap 方法,即有放回地从数据中随机抽取多个训练 MINIST 数据集。实验选择的重建误差函数为

$$L_H(x,z) = -\sum_{k=1}^{d}[x_k \lg z_k + (1-x_k)\lg(1-z_k)] \quad (4)$$

对所有分布式平台来说,训练模型的参数都一样,便于研究每个分布式平台的运算速度,选取速度快的计算节点做对比实验。

做单机训练的 3 台计算机的具体配置为:2G 内存,CPU 为 Intel Core i3-8100,GPU 为 Geforce GTX 750。做速度控制器的计算机的具体配置为:16G 内存,CPU 为 Intel Core i7-4710MQ,GPU 为 Geforce GTX 765M。

通过 K 均值聚类算法进行预训练,使每个学习到的中心作为单机模型中的初始参数[44],大幅度提高训练时间且可有效解决训练时的局部最优问题,使权值可以收敛到全局最优解。

在整个分布式平台中,认为最重要的参数为速度控制器里面的 T 参数(时间控制参数),T 参数的主要作用是控制速度控制器中输出参数更新的速度,选取 T 值越大,则速度控制器每次更新输出参数时要接受单个模型送来的输入参数越多,输出参数更新越快。而 T 值选取过大,导致训练每一步的更新量过大,使得训练达不到全局最优。在实际应用中,一般选取 T 值在单机模型数量的一到 3 倍。T 在实验中的取值为 0,1,2,4,6,8。对每一个 T 值,迭代训练 15 次(针对单机来说)。在这里,约定 T 为 0 时表示没有进行分布式训练。

接着使用延时梯度下降法和稀疏梯度下降法来进行 MINIST 数据集的学习,并与 Bagging-Down SGD 算法(时间控制参数为 8)学习的实验结果进行对比。

另外,采用 ImageNet 作为输入数据集[45]。由于数据集 Imagenet 包含的图像类别比较全面,基本涵盖了现在所有主流的图像类别,如果算法可以在 Imagenet 数据集上验证通过,说明该算法可以训练的图像数据类别较多,则可以证明算法的普适性。

选取不同的模型规模因子(0.2,0.4,0.6 和 0.8)进行对比试验,研究规模因子的不同取值对算法性能的影响。针对不同的计算规模(计算节点数量为 2 和 3),进行对比试验,研究不同计算节点规模对算法性能的影响。

3.2 分布式优化算法 Bagging-Down SGD 效果对比分析

为提高实验运行结果的可信度,先进行预实验(迭代计算 1 h),3 台机器实验结束后做对比,运算最快的计算机作为采样绘图所用机器。图 4 显示了这 6 次实验中每一次迭代计算后的重建误差函数值。

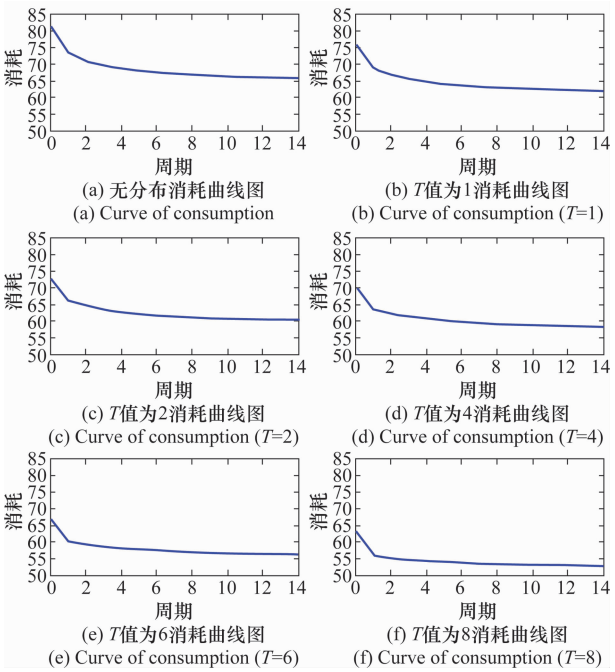

图 4 不同 T 值的训练实例

Fig. 4 Training examples of different T values

可以看到,模型的优化收益于 T 值的增加。选择速度最快的分布式平台进行测量,从得到的结果可以看出随着 T 值的增加,训练速度加快。Bagging-Down SGD 算法时间控制参数越大,则相应的重建误差越小,且减少得越快。

首先,分析每一次迭代计算出的重建误差值,重建误差值可一定程度上反映算法的优化速度和算法的结果可行性。从所得的结果来看,随着时间控制参数的增加,可以让重建误差的变化更加显著,说明在误差控制方面,分布式优化算法的重要参数选取的有效性,并且也说明整个算法的有效性。当 $T=8$ 时,重建误差下降梯度更大了。说明,随着 T 的增加,Bagging-Down SGD 算法越来越优。

图 5 的结果显示的是随着 T 的增加,训练时间的变化。采样于 3 台机器中最快的那一台,训练总 15 次的时间都相差比较大,在 $T=4$ 时开始比较稳定。

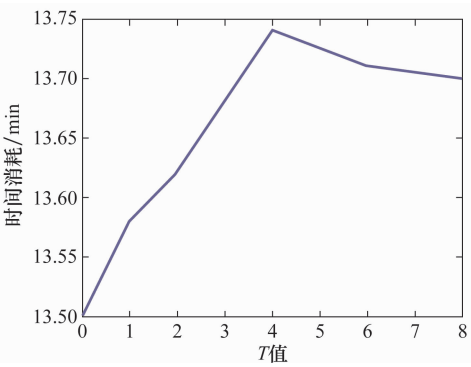

图 5 训练总时间随 T 变化曲线

Fig. 5 Curve of total training time versus T

在图 5 中可以看到，当 $T\geqslant 2$ 时，随着 T 增加，整个结构迭代 15 次时间的变化量。$T=2$ 到 $T=4$ 时，总时间是增加的，不难想到，是通信交换次数过于频繁导致通信负载过高问题，并且加上由于速度控制器参数更新过快，使得同一批次拿到的参数值不相同，速度慢的机器在做速度快机器已经做过的工作，导致整体效率的降低，但结合图 4 的数据，表明整体的运算速率还是在一定程度上得到增加。从 $T=4$ 开始，总体时间开始往下降，说明 Bagging-Down SGD 算法可以通过时间控制变量的变化来适当改善因为速度快慢导致的机器训练同一批次时拿到数据不同的状况，也会一定程度上降低了通信交流的频繁程度继而降低了通信负载。

图 6 显示了使用延时梯度下降法和 Bagging-Down SGD 算法训练的结果对比。

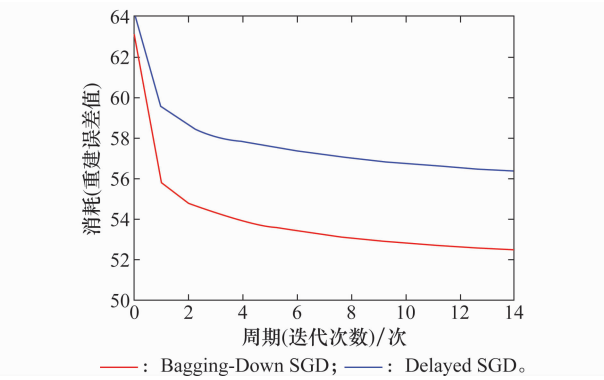

图 6　Bagging-Down SGD 与延时梯度下降算法对比

Fig. 6　Comparison of Bagging-Down SGD and delay gradient decline algorithms

由图 6 可以看出，Bagging-Down SGD 算法每次迭代的下降速率比延时梯度下降算法迭代的下降速率大。统计出了延时梯度下降算法整体运行所需要的时间为 13.74 min，采用公式 Δ(重建误差)/(训练时间)来计算算法的训练速度，计算得 Bagging-Down SGD 算法训练速度比延时梯度下降法降低了 15.8%，说明 Bagging-Down SGD 算法比延时梯度下降算法运算速度快，效果好。

图 7 显示了当参数集为 MINIST 时，Bagging-Down SGD 算法与稀疏梯度下降算法训练的结果对比。

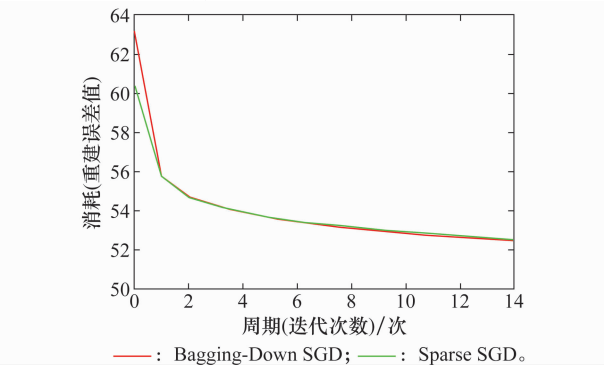

图 7　Bagging-Down SGD 与稀疏梯度下降算法对比

Fig. 7　Comparison of Bagging-Down SGD and sparse gradient decline algorithms

由图 7 可以看出，稀疏梯度下降算法所得最终的重建误差结果与 Bagging-Down SGD 算法所得的结果相近，两者在学习 MNIST 数据集时效果相近。图 8 显示了当参数集为 ImageNet 的时候，Bagging-Down SGD 算法和延时梯度下降算法的对比结果。

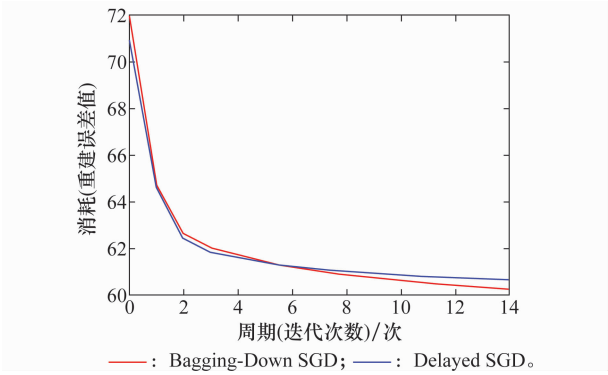

图 8　输入参数集为 ImageNet 的算法对比结果

Fig. 8　Comparisons of algorithms with ImageNet as input parameter set

可以看出，使用在 ImageNet 参数集训练时，Bagging-Down SGD 算法每次迭代的下降速率比延时梯度下降算法的下降速率更快。

记录了 Bagging-Down SGD 算法和延时梯度下降算法所用的时间分别为 381.48 min 和 414.72 min，说明 Bagging-Down SGD 算法比延时梯度下降算法所耗时间更少。计算得到 Bagging-Down SGD 算法的训练速度比延时梯度下降算法下降了 25.0%。由于该参数集不能满足稀疏梯度下降算法所需要的稀疏性和凹凸性要求，且数据规模较大，所以不能使用稀疏梯度下降法进行训练。由此可以看出 Bagging-Down SGD 算法的普适性比较好。

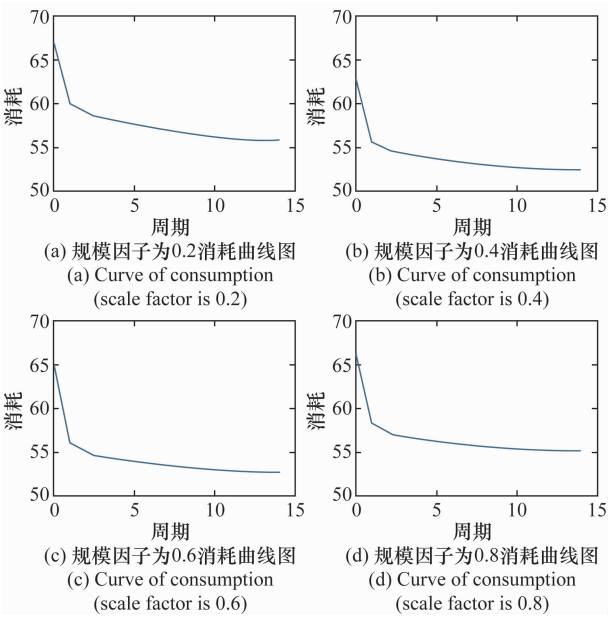

图 9　不同模型规模因子的训练结果

Fig. 9　Training results of scale factors of different models

图 9 显示了模型规模因子对模型的影响。可以看出，当选取模型规模因子为 0.4 时算法的重建误差下降速度最快。模型规模因子跟算法的规模有关，算法规模越大，则需要的模型规模因子越大。当选取的模型规模因子过大时，会导致模型训练速度降低。对于 3 个计算节点的算法规模而言，选取模型规模因子为 0.4 时算法达到最优的效果。

图 10 显示了不同的计算规模(参与单机模型训练的计算机数量)对模型的影响程度。

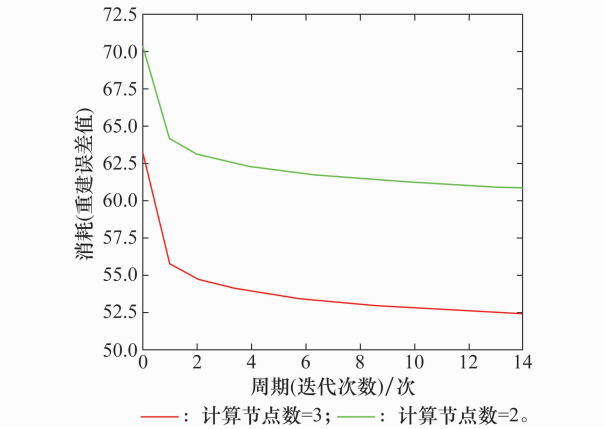

图 10 不同计算节点数的训练结果

Fig. 10 Training results of different calculation nodes

可以看出，参与单机模型训练的计算机为 3 台时的模型训练速度明显快于参与训练的计算机为 2 台时的模型训练速度。计算节点数量增多，使得训练速度明显加快。

4 实验结果讨论与展望

在分析当前大数据集分布式学习的两种主要算法的优缺点后，提出了 Bagging-Down SGD 算法，通过引入速度控制器，将原有单机模型基础上的模型计算和参数更新分开，在控制器中，重点通过速度调配器进行整个分布式模型学习速率的调节，达到了保证模型计算精度的同时提高模型训练速度的目的。第 1 个实验结果表明，Bagging-Down SGD 算法比延时梯度下降法速度提高了 15.8%，与稀疏自适应算法速度相差不大；第 2 个实验结果表明，Bagging-Down SGD 算法比稀疏梯度下降法更具有普适性，可以处理一些比较复杂的问题。较好地解决了当前两个主要大数据分布式学习算法存在的问题。

实验虽然选择的是静态的训练集，但整个结构和 Bagging-Down SGD 算法却可以使得动态参数集也能得到很好的训练。可以应用于动态模型的原因主要是该算法给动态模型的训练提供了结构基础。因为 Bagging-Down SGD 算法选择控制器作为参数更新的唯一方式，可以同时存储和处理数据，并且在处理输入集分配的时候借用了 Bagging 方法，这样即使输入数据集逐渐增加，也不会影响参数的训练。

采用包含 3 台计算机的分布式平台，考虑扩展到具有更多计算机的分布式平台上。这会导致单机之间的通讯问题。因为算法使用的计算机越多，作为子模型训练的计算机和速度控制器之间的交流越频繁，会导致越高的负载。

参考文献：

[1] NGIAM J, COATES A, LAHIRI A, et al. On optimization methods for deep learning[C]// Proc. of the 28th International Conference on Machine Learning (ICML-11), 2011: 265 - 272.

[2] MARTENS J. Deep learning via Hessian-free optimization[C]// Proc. of the 27th International Conference on Machine Learning (ICML-10), 2010: 735 - 742.

[3] HINTON G E, OSINDERO S, TEH Y W. A fast learning algorithm for deep belief nets[J]. Neural Computation, 2006, 18(7): 1527 - 1554.

[4] HINTON G, DENG L, YU D, et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups[J]. Signal Processing Magazine, 2012, 29(6): 82 - 97.

[5] 张亚军，刘宗田，周文，等. 基于深度信念网络的事件识别[J]. 电子学报，2017，45(6)：1415 - 1423.
ZHANG Y J, LIU Z T, ZHOU W, et al. Event recognition based on deep belief network[J]. Acta Electronica Sinica, 2017, 45(6): 1415 - 1423.

[6] DAHL G E, YU D, DENG L, et al. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition[J]. IEEE Trans. on Audio, Speech and Language Processing, 2012, 20(1): 30 - 42.

[7] BENGIO Y. How auto-encoders could provide credit assignment in deep networks via target propagation[J]. ArXiv Preprint ArXiv: 1407.7906, 2014.

[8] O'NEILL M. Neural network for recognition of handwritten digits[J]. Standard Reference Data Program National Institute of Standards and Technology, 2006.

[9] CIRESAN D C, MEIER U, GAMBARDELLA L M, et al. Deep big simple neural nets excel on handwritten digit recognition[J]. Corr, 2010, 22(12): 3207 - 3220.

[10] COATES A, NG A Y, LEE H. An analysis of single-layer networks in unsupervised feature learning[C]// Proc. of the International Conference on Artificial Intelligence and Statistics, 2011: 215 - 223.

[11] BENGIO Y. Deep learning of representations: looking forward [M]// Statistical Language and Speech Processing, Berlin Heidelberg: Springer, 2013: 1 - 37.

[12] RAINA R, MADHAVAN A, NG A Y. Large-scale deep unsupervised learning using graphics processors[C]// Proc. of the 26th Annual International Conference on Machine Learning, 2009: 873 - 880.

[13] MCDONALD R, MOHRI M, SILBERMAN N, et al. Efficient large-scale distributed training of conditional maximum entropy models[C]// Proc. of the Advances in Neural Information Processing Systems, 2009: 1231 - 1239.

[14] MCDONALD R, HALL K, MANN G. Distributed training strategies for the structured perceptron[C]// Proc. of the Annual Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies, 2010: 456 - 464.

[15] NOKLEBY M, BAJWA W U. Distributed mirror descent for stochastic learning over rate-limited networks[C]// Proc. of the IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2017: 1 - 5.

[16] AGARWAL A, DUCHI J C. Distributed delayed stochastic optimization[C]// Proc. of the Advances in Neural Information Processing Systems, 2011: 873 - 881.

[17] BENGIO Y, DUCHARME R, VINCENT P, et al. A neural probabilistic language model[J]. Journal of Machine Learning Research, 2000, 3(6): 932 - 938.

[18] NIU F, RECHT B, RE C, et al. HOGWILD!: a lock-free approach to parallelizing stochastic gradient descent[J]. Optimization and Control, 2011, 6: 693 - 701.

[19] KRIZHEVSKY A, HINTON G. Learning multiple layers of features from tiny images[R]. Canada: University of Toronto, 2009.

[20] LAN G, ZHOU Y. Random gradient extrapolation for distributed and stochastic optimization[J]. SIAM Journal on Optimization, 2018, 28(4): 2753 - 2782.

[21] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107 - 113.

[22] LOW Y, BICKSON D, GONZALEZ J, et al. Distributed graphLab: a framework for machine learning and data mining in the cloud[J]. Proceedings of the VLDB Endowment, 2012, 5(8): 716 - 727.

[23] ABADI M, AGARWAL A, BARHAM P, et al. Tensorflow: large-scale machine learning on heterogeneous distributed systems[J]. ArXiv Preprint ArXiv: 1603.04467, 2016

[24] AGARWAL A, CHAPELLE O, DUDiK M, et al. A reliable effective terascale linear learning system[J]. The Journal of Machine Learning Research, 2014, 15(1): 1111 - 1133.

[25] BERGSTRA J, BREULEUX O, BASTIEN F, et al. Theano: a CPU and GPU math expression compiler[C]// Proc. of the Python for Scientific Computing Conference (SciPy), 2010: 3.

[26] MOKHTARI A, RIBEIRO A. DSA: decentralized double stochastic averaging gradient algorithm[J]. The Journal of Machine Learning Research, 2016, 17(1): 2165 - 2199.

[27] PAN X, PAPAILIOPOULOS D, OYMAK S, et al. Parallel correlation clustering on big graphs[C]// Proc. of the Advances in Neural Information Processing Systems, 2015: 82 - 90.

[28] CIRESAN D, MEIER U, SCHMIDHUBER J. Multi-column deep neural networks for image classification[C]// Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012: 3642 - 3649.

[29] GROSSE R B, MADDISON C J, SALAKHUTDINOV R R. Annealing between distributions by averaging moments[C]// Proc. of the Advances in Neural Information Processing Systems, 2013: 2769 - 2777.

[30] DENG L, YU D, PLATT J. Scalable stacking and learning for building deep architectures[C]// Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012: 2133 - 2136.

[31] DUCHI J, HAZAN E, SINGER Y. Adaptive sub-gradient methods for online learning and stochastic optimization[J]. The Journal of Machine Learning Research, 2011, 12: 2121 - 2159.

[32] BELTRAMELLI T, RISI S. Deep-spying: Spying using smart watch and deep learning[J]. ArXiv Preprint ArXiv: 1512.05616, 2015.

[33] LANGFORD J, SMOLA A J, ZINKEVICH M. Slow learners are fast[C]// Proc. of the International Conference on Neural Information Processing Systems, 2009.

[34] ZINKEVICH M, WEIMER M, LI L, et al. Parallelized stochastic gradient descent[C]// Proc. of the Advances in Neural Information Processing Systems, 2010: 2595 - 2603.

[35] LEE S, NEDIC A, RAGINSKY M. Stochastic dual averaging for decentralized online optimization on time-varying communication graphs[J]. IEEE Trans. on Automatic Control, 2017, 62(12): 6407 - 6414.

[36] ALLEN-ZHU Z. Katyusha: the first direct acceleration of stochastic gradient methods[J]. The Journal of Machine Learning Research, 2017, 18(1): 8194 - 8244.

[37] LECUN Y A, BOTTOU L, ORR G B, et al. Efficient backprop[M]// Neural Networks: Tricks of the Trade, 2012: 9 - 48.

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

[39] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[J]. ArXiv Preprint ArXiv: 1409.1556, 2014.

[40] MCMAHAN H B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data[J]. ArXiv Preprint ArXiv: 1602.05629, 2016.

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

[42] LÄNGKVIST M, KARLSSON L, LOUTFI A. A review of unsupervised feature learning and deep learning for time-series modeling[J]. Pattern Recognition Letters, 2014, 42: 11 - 24.

[43] NALISNICK E, RAVI S. Infinite dimensional word embeddings[J]. ArXiv Preprint ArXiv: 1511.05392, 2015.

[44] COATES A, NG A, LEE H. An analysis of single-layer networks in unsupervised feature learning[C]// Proc. of the 14th International Conference on Artificial Intelligence and Statistics, 2011: 215 - 223.

[45] SHI Q, PETTERSON J, DROR G, et al. Hash kernels[C]// Proc. of the International Conference on Artificial Intelligence and Statistics, 2009: 496 - 503.

作者简介：

秦　超(1991 -)，男，博士研究生，主要研究方向为复杂系统建模与仿真、深度学习。

E-mail: woshiqchi@mail.nwpu.edu.cn

高晓光(1957 -)，女，教授，博士研究生导师，博士，主要研究方向为复杂系统建模与仿真、复杂系统控制理论与控制工程、大系统效能分析及智能信息处理。

E-mail: cxg2012@nwpu.edu.cn

陈大庆(1959 -)，男，高级讲师，博士研究生导师，博士，主要研究方向为数据挖掘、大数据分析、模式识别和人工神经网络。

E-mail: chend@lsbu.ac.uk