当边缘满足学习：资源受限分布式机器学习的自适应控制

王世强，蒂凡尼·图尔，西奥多罗斯·萨洛尼迪斯，梁建基，克里斯蒂安·马卡娅，何婷，陈凯文∗†∗†∗‡§

∗IBM T.J.沃森研究中心，约克敦高地，纽约，美国。电子邮件：{wangshiq，tsaloni，cmakaya}@美国ibm.com

†英国伦敦帝国理工学院。电子邮件：{蒂芙尼.图尔14, 梁健}@imperial.ac.uk

‡宾夕法尼亚州立大学，宾夕法尼亚大学公园，美国。he@cse.psu.edu美国陆军研究院医学博士kevin Hi。陈先生@邮件.mil§

*摘要*-新兴技术和应用，包括物联网（IoT）、社交网络和众包在网络边缘产生大量数据。机器学习模型通常是根据收集到的数据建立起来的，以便能够检测、分类和预测未来的事件。由于带宽、存储和隐私问题，将所有数据发送到集中位置通常是不切实际的。在本文中，我们考虑从分布在多个边缘节点的数据中学习模型参数的问题，而不需要将原始数据发送到一个集中的地方。我们的重点是使用基于梯度下降的方法训练的一类通用机器学习模型。本文从理论上分析了分布式梯度下降算法的收敛速度，并在此基础上提出了一种在给定资源预算下，在局部更新和全局参数聚合之间进行最佳折衷以使损失函数最小化的控制算法。通过对实际数据集的大量实验，在网络化原型系统和更大规模的仿真环境中对该算法的性能进行了评估。实验结果表明，在不同的机器学习模型和不同的数据分布下，我们提出的方法性能接近最优。

一、 简介

物联网（IoT）和社交网络应用的快速发展导致网络边缘产生的数据呈指数级增长。据预测，在不久的将来，数据生成速率将超过当今互联网的容量[1]。由于网络带宽和数据隐私问题，将所有数据发送到远程云是不切实际的，而且通常是不必要的。因此，研究机构估计超过90%的数据将在本地存储和处理[2]。新兴的移动边缘计算技术（MEC）[3]，[4]使得具有全局协调性的本地数据存储和处理成为可能，其中边缘节点，如传感器、家庭网关、微型服务器和小型单元，都具有存储和计算能力。多个边缘节点与远程

这是论文的扩展版本，标题与IEEE INFOCOM 2018接受的标题相同。

这项研究由美国陆军研究实验室和英国国防部根据W911NF-16-3-001协议号赞助。本文件中的观点和结论是作者的观点，不应被解释为代表美国军事研究实验室、美国政府、英国国防部或英国政府的官方政策。美国政府和英国政府有权复制和分发政府的转载，尽管此处有任何版权说明。

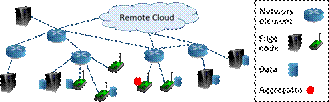


图1：系统架构。

云来执行大规模的分布式任务，包括本地处理和远程协调/执行。

为了分析大量的数据并获得有用的信息，以便对未来事件进行检测、分类和预测，通常采用机器学习技术。机器学习的定义非常广泛，从线性回归的简单数据摘要到支持向量机（SVM）和深层神经网络的多类分类[5]，[6]。后者近年来在图像分类等复杂任务中表现出了很好的性能。机器学习的一个关键因素是使用大量数据学习（训练）模型的能力。随着新应用程序生成的数据量不断增加，以及越来越多的应用程序成为数据驱动的，可以预见机器学习任务将成为未来分布式MEC系统的主要工作负载。然而，在资源受限的MEC系统上进行分布式机器学习是一个挑战。

在本文中，我们探讨如何有效地利用有限的计算和通讯资源，以达到最佳的学习效果。我们考虑一个典型的边缘计算架构，其中边缘节点通过网络元素（如网关和路由器）与远程云互连，如图1所示。在多个边缘节点收集和存储原始数据，并根据分布式数据训练机器学习模型，而不需要将原始数据从节点发送到中心位置。

我们主要研究基于梯度下降的分布式学习算法，这些算法具有广泛的机器学习模型的普遍适用性。学习过程包括局部更新步骤，每个边缘节点通过梯度下降来调整（局部）模型参数，以最小化在其自身数据集上 定义的损失函数。它还包括全局聚合步骤，其中在不同边缘节点获得的模型参数被发送到聚合器，聚合器是可以在远程云、网络元素或边缘节点上运行的逻辑组件。聚合器聚合这些参数（例如，通过取加权平均值），并将更新后的参数发送回边缘节点，以进行下一轮迭代。全局聚合的频率是可配置的；可以以一个或多个本地更新的间隔进行聚合。每个局部更新消耗边缘节点的计算资源，每个全局聚集消耗网络的通信资源。消耗的资源量可能会随时间而变化，并且全局聚集频率、模型训练精度和资源消耗之间存在复杂的关系。

我们提出了一种算法来确定全局聚集的频率，以便最有效地利用可用资源。这一点很重要，因为机器学习模型的训练通常是资源密集型的，学习任务的非优化操作可能会浪费大量的资源。本文的主要贡献如下：

1） 从理论上分析了基于梯度下降的分布式学习算法的收敛速度，得到了一个新的收敛界，该收敛界包含节点间的非i.i.d.数据分布和两个全局聚集之间任意数量的局部更新。

2） 利用上述理论结果，我们提出了一种学习数据分布、系统动态和模型特性的控制算法，并在此基础上实时动态调整全局聚集的频率，以在固定的资源预算下最小化学习损失。

3） 我们通过在硬件原型和模拟环境中使用真实数据集的大量实验来评估所提控制算法的性能，这证实了我们提出的方法对于不同的数据分布、不同的机器学习模型和，以及具有不同数量边缘节点的系统配置。

二。相关工作

现有的关于MEC的工作主要集中在通用应用程序上，针对应用程序卸载[7]、[8]、工作负载调度[9]、[10]和由用户移动性触发的服务迁移[11]、[12]提出了解决方案。然而，它们并没有解决机器学习应用中通信、计算和训练精度之间的关系，这对于优化学习任务的性能非常重要。

文献[13]、[14]、[15]从理论上研究了基于梯度下降的分布式机器学习，得到了训练收敛性和通信代价的渐近界。然后将结果推广到[16]中的分布式学习方法的一般类。在某种程度上，这些结果描述了通信和计算的权衡。然而，它们都没有考虑到全球聚集频率的适应性。许多分析也有不切实际的假设，例如不同节点的独立和同分布（i.i.d.）数据[13]，[14]，而涉及非i.i.d.数据分布的更一般的情况更难分析。

从实用的角度，[17]提出了一种分布式梯度下降的变体，称为联合学习，其中全局聚合是以同步的方式进行的。使用不同数据集的实验证实了该方法的有效性。全局聚集频率在[17]中是固定的。它不提供任何理论上的收敛保证，实验也不是在网络环境下进行的。最近[18]提出了网络系统中全局聚集频率动态变化的唯一工作。为了减少网络流量，只有当模型参数变化超过经验选择的阈值时，才会进行聚合。然而，没有考虑节点间的非i.i.d.数据分布，也没有理论分析阈值对学习的影响。文献[18]中的阈值化方法只是为了减少网络流量，这在计算资源有限的MEC系统中是不够的。

与上述研究不同，本文正式讨论了MEC系统中在给定资源预算的情况下动态确定全局聚合频率以优化学习的问题。这是一个非常重要的问题，因为每个学习步骤和之前的学习步骤之间存在复杂的依赖关系，很难通过分析来捕捉。由于不同节点的非i.i.d.数据分布和系统的实时动态性，这也是一个挑战。我们提出了一种适用于实时系统动力学的理论分析和算法。

在下一节中，我们将从总结分布式机器学习的基础知识开始。在第四节中，我们描述了我们的问题公式。第五节和第六节分别介绍了收敛性分析和控制算法。实验结果见第七节，结论见第八节。

三、 序言和定义

*A、 损失函数*

机器学习模型包括一组基于训练数据学习的参数。训练数据样本通常由两部分组成。一个是作为机器学习模型的输入的向量（例如图像的像素）；另一个是模型期望输出的标量。为了便于学习，每个模型在每个数据样本的参数向量上都定义了一个损失函数。损失函数捕捉模型对训练数据的误差，模型学习过程是在一组训练数据样本上最小化损失函数。对于每个数据样本，我们将损失函数定义为（w，xj，yj），简写为（w）[1]。*j***xw公司***jyj公司jjf福建*

表一：常用机器学习模型的损失函数

|  |  |
| --- | --- |
| 模型 | 损失函数（w，xj，yj）（，（w））*f福建* |
| 平滑支持向量机 | 是常数。） |
| 线性回归 | **wx公司**T*j*k2 |
| 聚类 | 式中，[w（1）T（2）T]T**栈单***,,...* |
| 卷积神经网络 | 级联线性和非线性变换的交叉熵，见[6] |

表一总结了常用机器学习模型的损失函数示例[2]，为了方便起见，本文假设所有向量都是列向量，并用以表示的转置。我们用“，”表示“定义为等于”，用k·k表示L2范数。**二十**T

假设我们有具有本地数据集D1，D2，…，Di，…，DN的边缘节点。对于节点上的每个数据集Di，此节点上收集数据样本的损失函数为*N我*

*.* （一）

为了方便起见，我们定义| Di |，其中|··|*Di公司*表示集合的大小，以及。假设Di∩Di0=∅对于6=i0，我们将所有分布式数据集的全局损失函数定义为*我*

*.* （二）

不需要在多个节点之间共享信息，直接计算全局损失函数w。*F不能*

*B。学习问题*

学习的问题是

**w**∗=argminF（w）。（三）

由于大多数机器学习模型固有的复杂性，通常不可能找到（3）的闭式解。因此，（3）常采用梯度下降法求解。

*C。分布梯度下降*

我们提出了一种求解（3）的正则分布梯度下降算法，该算法在当前的研究领域（如[17]）中得到了广泛的应用。每个节点都有其局部模型参数（t），其中=0,1,2，。。。表示迭代索引。当=0时，所有节点的局部参数初始化为相同的值。对于0，（t）的新值根据局部损失函数的梯度下降更新规则，基于上一次迭代−1中的参数值计算。对每个节点的局部损失函数（在局部数据集上定义）的这种梯度下降步骤称为局部更新。一次或多次局部更新后，通过聚合器进行全局聚合，将每个节点的局部参数更新为所有节点参数的加权平均值。我们定义每个迭代都包含一个本地更新，然后是一个可能的全局聚合。*我***栈单***我tt我t>我t*

|  |  |  |
| --- | --- | --- |
| 操作： | **A**局部迭代 | **B**全局聚合 |



图2：节点处（t）和（t）值的图示。**栈单***我*e*我我*

算法1：分布式梯度下降

输入：，*τT*

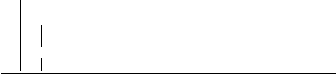
产量：（T）**w**

1                        将（0）和（0）初始化为相同的值；**栈单***我*e*我我*

2                        对于=1,2，…，T do*t*

3                        对于每个节点，使用（4）计算局部更新；*我并行*

4                        如果那样的话*t是的整数倍τ*

5                        （全局聚合）集合ei（t）←w（t），其中（t）**栈单***我*

定义见（5）；

6                        其他的

7                        （无全局聚集）集合ei（t）←wi（t）；**w***我*

全局聚合后，每个节点的局部参数t通常会发生变化。为了方便起见，我们使用ei（t）来表示可能的全局聚集后节点处的参数。如果在迭代中没有进行聚合，我们有ei（t）=wi（t）。如果在迭代时进行聚合，则通常ei（t）6=wi（t），我们设置（t）=w（t），其中（t）是下文（5）中定义的（t）的加权平均值。这些定义的示例如图2所示。**wwwwwww***我我我tt*e*我我*

每次迭代中的局部更新是在上一次迭代中可能的全局聚合之后对参数执行的。对于每个节点，更新规则如下：*我*

**w***我*（t） =w（t−1）−η∇Fi（w（t−1））（4），其中≥0为步长。对于任何迭代（可能包括也可能不包括全局聚合步骤），我们定义e*我*e*我ηt*

**w***.* （五）

如果在迭代过程中进行全局聚合，那么这个全局模型参数（t）只对系统中的节点可见，但是我们为所有节点定义了它以便于以后的分析。**w***tt*

我们定义系统在每两个全局聚合之间的每个节点执行局部更新步骤。我们定义为迭代的总数。为了简单起见，我们假设它是的整数倍，当我们在第VI-B节中讨论实际问题时，这将被放宽。算法1中提出了分布式梯度下降算法，它忽略了聚合器和边缘节点之间的通信等细节。这些实施细节将在后面的第VI-B节中讨论。*τTTτ*

算法1的基本原理是当=1时，即当我们在每个局部更新步骤之后进行全局聚合时，分布式梯度下降（忽略通信方面）相当于集中式梯度下降，后者假设所有的数据样本都在一个集中的位置可用，并且可以直接观察到全局损失函数及其梯度。这是由于梯度算子的线性。详见附录A。*τ*

四、 问题表述

当大量的数据（通常是训练一个精确的模型所需要的）分布在大量的节点上时，分布式学习过程会消耗大量的资源。这里“资源”的概念是通用的，可以包括与计算和通信相关的时间、能量、货币成本等。为了不积压系统并保持较低的运行成本，人们常常不得不限制用于学习每个模型的资源量。在计算和通信资源不如数据中心丰富的边缘计算环境中，这一点尤为重要。

因此，一个自然的问题就是如何有效地利用一定数量的资源，使模型训练的损失函数最小化。对于上面提出的基于梯度下降的分布式学习方法，问题缩小到确定和的最优值，从而使（w）在给定的资源约束下最小化。*TτF*

形式上，我们定义所有参与节点的局部更新的每一步消耗资源单位，全局聚合的每一步消耗资源单位，其中≥0和≥0都是实数。对于给定和*cbcbTτ*，则消耗的资源总量为。让*R*表示总资源预算。我们寻求以下问题的解决方案：

最小F（w（T））（6）*τ、 T*

                                    s、 t。

为了求解（6），我们需要找出损失函数（迭代后）（w（T））的值是如何影响的。通常不可能找到一个与（w（T））相关的精确解析表达式，因为它取决于梯度下降的收敛性（只有上下界已知[19]）和全局聚集频率的影响。此外，资源消耗和在实践中可能是时变的，这使得问题比（6）更具挑战性。*τTTFτTFcb*

在第五节中，我们分析了分布式梯度下降算法（算法1）的收敛速度上界，然后用该上界作为近似解（6），并在第六节中提出了一种自适应选择最优值并达到近似最优资源利用率的控制算法。*τT*

五、 收敛性分析

在这一节中，我们分析了算法1的收敛性，并找到了（w（T））−F（w∗）的上界，在这里我们回顾一下，这是迭代的总数。为了便于分析，我们首先介绍一些符号。*FT*

*A、 定义*

我们用来表示迭代中全局聚合的总数。因为我们之前假设是*KTTτ*，我们有。我们可以进一步划分

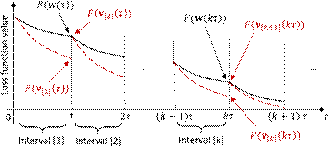


图3：不同间隔的定义说明。

如图3所示，将迭代分为不同的间隔，每个间隔中只有第一次和最后一次迭代包含全局聚合。我们使用速记符号[k]来表示迭代间隔[（k−1）τ，kτ]，对于=1,2，…，k。*TKk*

对于每个区间[k]，我们使用（t）来表示一个辅助参数向量，它遵循一个集中梯度下降，根据**五**[克]

**vv型**[克]（t） η（k）仅为−1（k）定义]（其中τ∈1）。该更新规则基于全局损失函数（w），该函数仅当所有数据样本在中心位置可用时才可观测（因此是集中梯度下降），而（4）中的迭代基于局部损失函数（w）。[克]*tkF金融机构*

我们定义（t）在每个间隔[k]开始时与（t）“同步”，即（（k−1）τ），（（k−1）τ），其中（t）是（5）中定义的局部参数的平均值。注意，我们还有ei（（k−1）τ）=w（（k−1）τ），因为全局聚集（或当=1时初始化）是在迭代（k−1）τ中执行的。**大众汽车**[克][克]*我k*

通过以上定义，我们可以通过两步的方法找到算法1的收敛界。第一步是找出（kτ）和（kτ）之间的差距，即在没有全局聚集的局部更新步骤后，分布梯度下降和集中梯度下降之间的差异。第二步是在每个区间[k]内将此间隙与（t）的收敛速度相结合，得到（t）的收敛速度。**wvvw公司**[克]*kτ*[克]

为了分析的目的，我们对损失函数作了如下假设。

假设1。我们假设如下*一：*

*（一）金融机构*（w）*是凸的*

*（二）金融机构*（w） kFi（w）−Fi（w0）k≤ρkw−w0k*是ρ-Lipschitz，即。，对于任何人***栈单***,*0

*（三）金融机构*（w） k∇Fi（w）–∇Fi（w0）k≤βkw−w0k*是β-平滑，即。，对于任何人***栈单***,*0

对于光滑支持向量机和线性回归，上述假设是满足的（见表一）。第七节将介绍的实验结果表明，对于损失函数不满足上述假设的模型（如神经网络），我们的控制算法也能很好地工作。

引理1。（w）*F是凸的，ρ-Lipschitz，和β-平滑。*

*证据。*这从假设1（w）的定义和三角不等式可以很容易地理解。*F*

我们还定义了以下度量来捕获局部损失函数的梯度和全局损失函数的梯度之间的分歧。这种差异与数据在不同节点上的分布方式有关。

定义1。（梯度散度）对于任何k∇Fi（w）–∇F（w）k*我和***w***，我们定义δi作为，即。，*

                                                                                *.* （八）

*我们还定义*

*B、 主要成果*

下面的定理给出了区间[k]内（t）与（t）之差的上界。**wv型**[克]*t*

定理1。对于任何区间[k]∈[k]*和t，我们有*

（九）

*哪里*

（十）

*对于任何人x轴>*0*. 此外，作为F*(·)*是ρ-Lipschitz，我们有F*（wv（t））−F（[]k（t））≤ρh（t−（k−1）τ）*.*

*证据。*见附录B。

注意，我们总是有0和0，否则梯度下降过程或损失函数变得微不足道。因此，由于Bernoulli不等式，对于≥0，我们有（ηβ+1）x≥ηβx+1。将其代入（10）可证明我们始终具有（x）≥0。*η >β >十h*

很容易看出我们有（0）=h（1）=0。因此，当=（k−1）τ时，即在区间[k]的开始处，（9）中的上界为零。这与（（k−1）τ）=w（（k−1）τ）的定义一致。当=1+（k−1）τ时，（9）中的上限也是零。这与第III-C节中的讨论一致，表明当在全局聚集之后只执行一次局部更新时，分布式梯度下降和集中梯度下降之间没有差距。如果=1，我们得到−（k−1）τ对于任何区间[k]是0或1，并且∈[k]。因此，（9）中的上界对于=1变得精确。*ht***五**[克]*ktτttτ*

对于1，=t−（k−1）的值可以更大。当较大时，（10）中的（ηβ+1）x的指数项占优势，（t）与（t）之间的间隙随着∈[k]而呈指数增大。我们还注意到（x）与梯度散度成正比（见（10）），这是直观的，因为局部梯度与全局梯度（对于同一参数）的差异越大，间隙就越大。这种差距是由于在每个全局聚集之后的第二次局部更新开始的不同节点的局部梯度的差异造成的。在极端情况下，当所有节点具有完全相同的数据样本（因此具有相同的局部损失函数），梯度将始终相同且=0，在这种情况下（t）和（t）始终相等。*τ >十十***wvwwv公司**[克]*tthδδ*[克]

定理1给出了每个迭代区间[k]的分布梯度下降和集中梯度下降之间的差的上界，假设集中梯度下降中的（t）与每个[k]开始时的（t）同步。基于这个结果，我们得到了以下定理。**大众汽车**[克]

定理2。算法1的收敛上界*T迭代由*



*满足下列条件时：*

*（一）*

*（二）*

*3） 对所有人t和k代表哪个***五**[克]（吨）*定义了*

*（四）F*（w（T））−F（w∗）≥ε0*哪里ε >和**.*

*证据。*见附录C。

在上面的定理中，条件1规定为了保证收敛，步长应该足够小。更平滑的函数允许较大的步长（即较小的步长）。当条件1成立时，条件2始终为=1，因为（1）=0。当1时，它适用于足够小的（τ），这意味着为了保证收敛，（9）中的间隙必须很小。这意味着全局聚集需要足够频繁地执行，这样局部梯度就不会偏离全局梯度太多，否则算法可能无法收敛。的值与条件3和4一起指定了最优性差距的下限。当1和*βτhτ >hετ >ε*是小的，即当损失函数接近其最优值时，项大，界变松。什么时候？*ε*低于某个阈值，条件2不再成立。因为（w（T））−F（w∗）通常在变大时变小，从而产生一个更小的直观结果，这意味着当1时，收敛性只保证到非零的最优性差距为→∞。保证收敛到零最优性间隙为1（因为（1）=0，在这种情况下，边界不依赖于）。本文考虑的不是→∞情形，而是有限的资源预算。因此，在我们考虑的大多数问题场景中，（T）和（T）的值并不“太接近”最优值，并且存在一个值，使得定理2中的所有条件都成立。*FTετ >TτhεTT***wv公司**[克]*ε*

由于在迭代过程中进行全局聚合，系统中的所有节点都知道（T）的值，这是分布式梯度下降的最终结果。我们还注意到（11）中的边界对数据如何分布在不同的节点没有限制。*T***w**

六、 控制算法

在本节中，我们提出了一个近似求解（6）的算法。我们首先假设资源消耗和已知，然后求解和的值。然后，我们考虑实际情况，其中，和其他一些参数是未知的，并且可能随时间变化，我们提出了一个控制算法，估计参数和动态调整的值，实时。*cbτTcbτ*

*A、 （6）的近似解*

我们使用（11）中的上界作为（w（T））−F（w\*的近似值）。因为对于给定的全局损失函数（w），其最小值（w∗）是一个常数，（6）中的（w（T））的最小值等于最小化（w（T））−F（w∗）。利用这个近似值并重新排列（6）中的不等式约束，我们可以将（6）改写为*FFFFF*

（十二）

s、 和定理2中的条件。

在下面，我们假设*ηε*选择足够小，以满足定理2中的条件1、3和4。当条件2成立时，（12）中的分母为正，因此，的最佳值。

将其代入（12）并取目标函数的倒数，我们得到

（十三）

                     s、 定理2中的条件1，3，4。

（13）的最优解总是满足定理2中的条件2，因为=1（在这种情况下（τ）=0*τh*)使目标函数为正，从而满足条件。（13）中的目标函数除以产量

（十四）

其中，全局聚合的相对资源消耗被本地更新的相对资源消耗标准化。

基于上述推理，我们知道最小化（12）和最大化（13）和（14）的最优值是相同的。

因此，我们可以解决

*τ*∗=argmaxG（τ）（十五）

*τ*

我们也可以从中得到。下面的命题表明*G*（τ） 有一个唯一的最大值。

命题1。什么时候是严格凹函数*τ*≥1个*.*

*证据。*见附录D。

该方程没有闭式解，因此没有闭式表达式*τ*∗. 因为取整数值，我们可以使用类似于[12]中的二元搜索过程来求解，其复杂度为（logτ），其中是搜索中的最大值，将在下一小节中进一步讨论。*ττ*∗*O*最大值*τ*最大值*τ*

算法2：聚合器处的过程

输入：资源预算、控制参数、搜索范围参数。*Rϕγ*

产量：（T）**w**

1                          ←0，←0为资源，←a*τtss*

2                          将（0）初始化为常量或随机向量；**w**

3                          重复

4                          发送（t）和到所有边缘节点，如果设置了停止，也发送停止；**w***τ*

5                          *t*0 ←t；//保存上次传输的迭代索引**w**（吨）

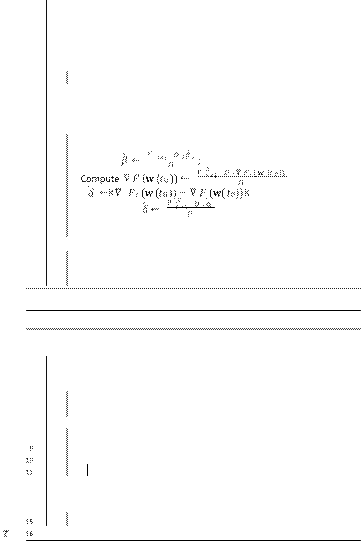
6                          *t*←t+τ；//下一个全局聚合在之后*τ*迭代

7                          从每个节点接收（t），ˆi；**w***我c我*

8                          按（5）计算（t）；**w**

9                          如果设置了停止标志，则

10                        返回（t）作为最终结果；**w**

11                        根据从所有节点接收到的ˆi和聚集器的本地测量值估计资源消耗量ˆ；*c*ˆ*bc我*

12                        *s*←s+cτˆ+ˆ*b*;

13                        如果0，则*t*0 *>*

14                        从每个节点接收，∇Fi（w（t0））；*β*ˆ*我我*

*N*15            估计

16                                                                                                                                ，估算

每一个，从哪一个*我*

                    我们估计；

17                                                                                                                                根据（15）对[1，τ]内的整数值进行二元搜索，计算出的新值，其中设←γτ；*τ*最大值*τ*最大值

18                                                                                                                                如果+cτˆ+≥R，则*s*ˆ*b*

19                                                                                                                                减小到最大可能值，以便*τ*

剩余迭代的估计资源消耗在预算内，设置停止标志；*R*

算法3：每个边节点的过程*我*

1                        初始化←0；*t*

2                        重复

3                        从聚合器接收（t）和新的，设置ei（t）←w（t）；**栈单***τ*

4                        *t*0 ←t；//保存上次传输的迭代索引**w**（吨）

5                        如果0，则*t>*

6                        估计

*β*ˆ*我*←k∇Fi（wi（t））-∇Fi（w（t））k/kwi（t）−w（t）k；

7                        对于=1,2，…，τdo*米*

8                        *t*←t+1；//开始下一次迭代

执行本地更新并根据（4）获得（t）；如果**w***我m<τ*

设置ei（t）←wi（t）；**w**

12              估计节点上一次本地更新的资源消耗ˆi；*c我*

13              发送至（t）聚合器；**w***我c*

14              如果0，则*t*0 *>*

发送，∇Fi（w（t0））到聚合器，直到收到停止标志；*β*ˆ*我*

*B、 动态控制算法*

（τ）的表达式（包括（τ））有一些在实际中可能未知的参数。其中，and（and）与资源消耗有关，与损失函数特性有关。这四个参数是在学习过程中实时估计出来的，稍后将进一步解释。参数是预先指定和已知的梯度下降步长。其余参数，在（τ）的定义中都出现在一个术语中*Ghcb一βδηρεωG*（见（14）），我们定义为手动选择的控制参数，对于同一机器学习模型，该参数保持不变。下一节中给出的实验结果表明，一个固定值*ϕ*适用于不同的数据分布和不同数量的节点。我们注意到，较大的值放大了梯度散度（与（τ）成比例）并产生较小的值，反之亦然。因此，在实践中，不难在一个小而简单的设置上调整的值，然后将其应用于一般情况（也可参见第VII-B4节中关于灵敏度的结果）。在实际实现中，我们不关心定理2的条件。*ϕδhτ*∗*ϕϕ*

如前所述，本地更新在边缘节点上运行，而全局聚合是通过聚合器（也可以在其中一个边缘节点上运行的逻辑组件）的帮助来执行的。算法2和算法3分别介绍了聚合器和每个边缘节点处的过程，其中算法3的第7-11行用于局部更新，其余部分被视为全局聚合或初始化的一部分。我们假设聚合器启动学习过程，初始模型参数（0）由聚合器发送到所有边缘节点。**w**

的值在每个全局聚合步骤中根据最新的参数估计值重新计算。如算法2的第17行所示，我们搜索最高达当前值（即=γτ）倍的新值，并找到最大化（τ）的值，其中0是一个固定参数。它的存在限制了搜索空间，也避免了由于初始参数估计可能不准确而增长过快。的新值与（t）一起发送到每个节点（算法2的第4行）。*ττγττ*最大值*Gγ >γττ***w**

和的值基于边缘节点和聚合器的资源消耗的测量值来估计（算法2的第11行）。估算方法取决于所考虑的资源类型。例如，当资源为能量时，所有节点的总能量消耗（每次本地更新）被视为；当资源为时间时，所有节点（每次本地更新）的最大计算时间被视为。聚合器还根据估计值监视总资源消耗，并将总资源消耗量与资源预算进行比较（算法2的第18行）。如果消耗的资源达到预算限制，则停止学习并返回最终结果。*cbccR*

和的值根据在（t）和（t）处计算的局部和全局梯度估计（见算法2的第14-16行和算法3的第6和15行）。为了进行估计，每个边缘节点需要在同一次迭代中同时访问其局部模型参数（t）和全局模型参数（t）（见算法3第6行），这只有在迭代中进行全局聚集时才可能。因为（t）只有在全局聚合之后每个节点才可观测到，所以和的估计值仅可用于从初始化后的第二个全局聚合步骤开始重新计算，该步骤使用在上一个全局聚合步骤中获得的估计值[3]。*βδ***wwwww网站***我我ttβδτ*

七。实验结果

*A、 设置*

为了评价所提出的控制算法的性能，我们在5个节点的网络化原型系统和5到500个节点的仿真环境中进行了大量的实验。

该原型系统由三台Raspberry Pi（版本3）设备和两台笔记本电脑组成，它们都通过Wi-Fi在办公楼内互连。这表示一个边缘计算环境，其中边缘节点的计算能力是异构的。所有这5个节点都有本地数据集（如下所述），在这些数据集上进行模型训练。聚合器位于一台笔记本电脑上，因此与一个本地数据集位于同一位置。

我们把时间当作实验的资源。对于原型系统，我们训练每个模型在固定的时间预算内。和的值分别对应于每次本地更新和全局聚合所用的实际时间。仿真环境使用模拟资源消耗进行模型训练，资源消耗被设定为原型测量值的平均值。*cb*

当训练数据量很大时，计算整个（局部）数据集上定义的损失函数的梯度通常在计算上是困难的。在这种情况下，通常使用随机梯度下降法（SGD）[5]，[6]，它使用在随机抽样数据子集上定义的损失函数计算的梯度来近似真实梯度。虽然本文的理论分析是基于确定性梯度下降（DGD），但我们在实验中同时考虑了DGD和SGD来评估所提出算法的一般适用性。

*（一）模型和数据集：*我们评估了四个不同模型在三个不同的数据集上的训练。模型包括平滑支持向量机、线性回归、K均值和深卷积神经网络（CNN）[4]。这些模型的损失函数汇总见表一，详见[5]、[6]。其中，smoothSVM（以下简称SVM）和线性回归的损失函数满足假设1，而K均值和CNN的损失函数是非凸的，因此不满足假设1。

SVM和CNN在MNIST数据集[20]上进行训练，该数据集包含70000个手写数字（60000个用于训练，10000个用于测试）。对于支持向量机，我们使用偶数和奇数作为二值标签，而对于CNN，我们使用10个不同数字的多类标签。由于DGD无法处理大量的数据，因此通过DGD进行的SVM训练只需要从整个数据集中抽取1000个训练样本和1000个测试数据样本。SVM和CNN的SGD变量使用整个MNIST数据集。线性回归是在Facebook metrics数据集[21]上进行的，该数据集有500个样本，具有多个与化妆品品牌发布的帖子相关的属性。该模型发现总交互数与所有其他属性之间存在线性关系。K-means在用户知识建模数据集[22]上执行，该数据集有403个样本，每个样本具有5个属性，总结了用户与web环境的交互。样本可以分成4个代表不同知识水平的聚类，但是我们假设我们没有这个分组的先验知识。

*（二）不同节点的数据分布（情况1-4）：*我们考虑了四种不同的将数据分布到不同节点的方法。在案例1中，每个数据样本随机分配给一个节点，因此每个节点都有无偏（但不是完整）的信息。在案例2中，每个节点中的所有数据样本都有相同的标签[5]。这表示每个节点都有偏差信息的情况，因为整个数据集具有多个不同标签的样本。在案例3中，每个节点都有完整的数据集（因此是完整的信息）。在情况4中，具有前半部分标签的数据样本如情况1一样分布到节点的前半部分；其他样本如情况2一样分布到节点的后半部分。这是一个有偏见和无偏见的结合。

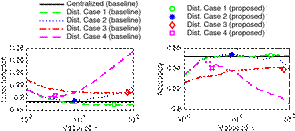
*（三）训练和控制参数：*在我们所有的实验中，我们将搜索范围参数设为10。除非另有规定，我们将SVM、线性回归和K均值的控制参数设置为0.2，CNN的控制参数设置为10-4。对于SVM、线性回归和CNN，梯度下降步长为=0.01，对于K均值，梯度下降步长=0.1。*γϕϕηη*

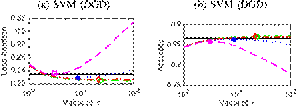
*B、 结果*

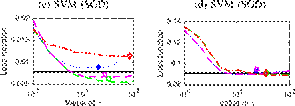
*1） 损失和精度值：*In our first set of experiments, the SVM, linear regression, and K-means models were trained on the prototype system, where the resource (time) budget for each model training instance is fixed to 15 seconds. Due to the resource limitation of Raspberry Pi devices, the CNN model was trained in a simulated environment of 5 nodes, with a total budget equivalent to 200 local updates, and a global aggregation consuming = 5.0 times the resource of one local update. The value = 5.0 was obtained from the time measurements of SVM (SGD) on the prototype.*a a*

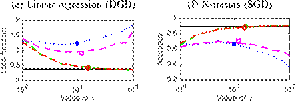
We compare the loss function values of our proposed algorithm (with adaptive ) to baseline approaches that include centralized training and distributed training with fixed . We also compare the classification accuracies for the SVM and CNN classifiers. The centralized baseline is obtained in a simulated environment by running only local updates on a*ττ*

single node subject to the total resource budget, where the local update resource consumption is estimated based on measurements on the prototype system averaged over all cases of each model. The distributed baselines run on the prototype system (except for CNN as discussed earlier) under the same setup but with different fixed values of (i.e., no adaptation). Note that this setup with non-adaptive *τ τ is the same as the*







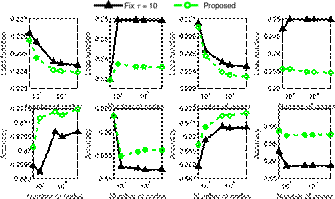


                    (g) CNN (SGD) (h) CNN (SGD)

Fig. 4: Loss function values and classification accuracy with different . Only SVM and CNN classifiers have accuracy values. The curves show the results from the baseline with different fixed values of . Our proposed solution (represented by a single marker for each case) yields an average and loss/accuracy that is close to the optimum in all cases.*τττ*

*federated learning approach in [17]*. When fixing = 1, it is also equivalent to the synchronous SGD approach in [23].*τ*

The average results of 30 different experiments (15 for CNN) are shown in Fig. 4. We note that the proposed approach only has one data point (represented by a single marker in the figure) in each case, because the value of is adaptive in this case and the marker location shows the average with the corresponding loss or accuracy. The centralized case also only has one data point but we show a flat line across different values of for the ease of comparison. We see that the proposed approach performs close to the optimal point for all cases and all models[6]. We also see that the optimal value of is different for different cases and models, so a fixed value of does not work well for all cases. In some cases, the distributed*τ τ τ τ τ*



        (a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4

Fig. 5: Loss function values and classification accuracy with different numbers of nodes for SVM (SGD).

approach can perform better than the centralized approach, because for a given amount of time budget, distributed learning is able to make use of the computation resource at multiple nodes. It does not always perform better than centralized because the resource consumption of the centralized approach is estimated as the average over all distributed cases for each model, so some will perform worse. For DGD approaches, Case 3 does not perform as well as Case 1, because the amount of data at each node in Case 3 is larger than that in Case 1, and DGD processes the entire amount of data thus Case 3 requires more resource (time) for each local update.

*2) Varying Number of Nodes:* Results of SVM (SGD) for the number of nodes varying from 5 to 500 are shown in Fig. 5, which are obtained in the simulated environment with = 5.0 and total budget equal to 868 local updates (both parameters are obtained from measurements on the prototype system). Our proposed approach outperforms the fixed = 10 baseline in all cases.*a τ*

*3) Instantaneous Behavior:* We further study the instantaneous behavior of our system. Results for SVM (DGD) is shown in Fig. 6 for a single run of 30 seconds (for each case) on the prototype system. Further results of SVM (SGD) are available in Appendix E. We see that except for Case 3 of SVM (DGD), the value of converges after a certain amount of time, showing that the control algorithm is stable. The value of keeps increasing in Case 3 of SVM (DGD) because all nodes have exactly the same data in this case and DGD uses all the data samples (i.e., no random sampling). There is no gradient deviation in this case and the optimal value of is infinity. As expected, the gradient deviation is larger for Cases 2 and 4 where the data samples at different nodes are biased. The same is observed for , indicating that the model parameter is in a less smooth region for Cases 2 and 4. Case 3 of SVM (DGD) has a much larger value of because it processes more data than in other cases and thus takes more time, as explained before. The value of exhibits fluctuations because of the randomness of the wireless channel.*τ*∗ *τ*∗ *τ δ β***w** *c b*

*4) Sensitivity of ϕ:* The sensitivity of the control parameter is shown in Fig. 7, where the experimentation settings are*ϕ*

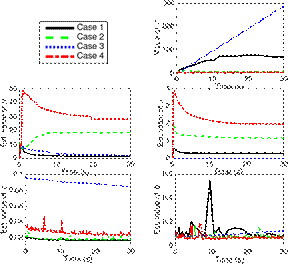
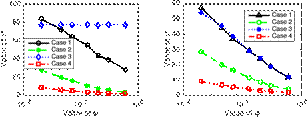


Fig. 6: Instantaneous results of SVM (DGD) with the proposed algorithm.



                  (a) SVM (DGD) (b) SVM (SGD)

Fig. 7: Impact of on the average value of .*ϕ τ*∗

the same as for Fig. 4. We see that the relationship among in different cases is mostly maintained with different values of . The value of decreases approximately linearly with logϕ, which is consistent with the fact that there is an exponential term w.r.t. in (τ) (and thus (τ)). For Case 3 of SVM (DGD), remains the same with different , because = 0 (thus (τ) = 0) in this case and the value of does not affect (τ) (see (14)). We also see that small changes of does not change much, indicating that one can take big steps when tuning in practice and the tuning is not difficult.*τ*∗ *ϕτ*∗ *τ hGτ*∗ *ϕδ hϕ Gϕ τ*∗ *ϕ*

VIII. CONCLUSION

In this paper, we have focused on gradient-descent based distributed learning that include local update and global aggregation steps. Each step of local update and global aggregation consumes resources. We have analyzed the convergence bound for distributed learning with non-i.i.d. data distributions. Using this theoretical bound, a control algorithm has been proposed to achieve the desirable trade-off between local update and global aggregation in order to minimize the loss function under a resource budget constraint. Extensive experimentation results confirm the effectiveness of our proposed algorithm.

REFERENCES

[1] M. Chiang and T. Zhang, “Fog and IoT: An overview of research opportunities,” IEEE Internet of Things Journal, vol. 3, no. 6, pp. 854– 864, Dec. 2016.

[2] R. Kelly, “Internet of Things data to top 1.6 zettabytes by 2020,” Apr. 2015. [Online].

Available: https://campustechnology.com/articles/2015/04/15/internetof-things-data-to-top-1-6-zettabytes-by-2020.aspx

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “Mobile edge computing: Survey and research outlook,” CoRR, vol. abs/1701.01090, 2017. [Online]. Available: http://arxiv.org/abs/1701.01090

[4] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” IEEE Communications Surveys Tutorials, vol. PP, no. 99, 2017.

[5] S. Shalev-Shwartz and S. Ben-David, Understanding machine learning: From theory to algorithms. Cambridge university press, 2014.

[6] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.

[7] Y. Xiao and M. Krunz, “QoE and power efficiency tradeoff for fog computing networks with fog node cooperation,” in IEEE INFOCOM 2017, May 2017, pp. 1–9.

[8] L. Tong and W. Gao, “Application-aware traffic scheduling for workload offloading in mobile clouds,” in IEEE INFOCOM 2016, Apr. 2016.

[9] L. Tong, Y. Li, and W. Gao, “A hierarchical edge cloud architecture for mobile computing,” in IEEE INFOCOM 2016, Apr. 2016.

[10] H. Tan, Z. Han, X.-Y. Li, and F. Lau, “Online job dispatching and scheduling in edge-clouds,” in IEEE INFOCOM, May 2017.

[11] L. Wang, L. Jiao, J. Li, and M. Muhlhauser, “Online resource allocation for arbitrary user mobility in distributed edge clouds,” in Proc. of ICDCS, June 2017, pp. 1281–1290.

[12] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, “Dynamic service placement for mobile micro-clouds with predicted future costs,” IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.

[13] A. Agarwal and J. C. Duchi, “Distributed delayed stochastic optimization,” in Advances in Neural Information Processing Systems, 2011, pp. 873–881.

[14] Y. Zhang, M. J. Wainwright, and J. C. Duchi, “Communication-efficient algorithms for statistical optimization,” in Advances in Neural Information Processing Systems, 2012, pp. 1502–1510.

[15] Y. Arjevani and O. Shamir, “Communication complexity of distributed convex learning and optimization,” in Advances in neural information processing systems, 2015, pp. 1756–1764.

[16] C. Ma, J. Konecnˇ y, M. Jaggi, V. Smith, M. I. Jordan, P. Richt` arik,´ and M. Taka´c, “Distributed optimization with arbitrary local solvers,”ˇ Optimization Methods and Software, vol. 32, no. 4, pp. 813–848, 2017.

[17] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proc. of AISTATS, 2016.

[18] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, “Gaia: Geo-distributed machine learning approaching LAN speeds,” in 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), 2017, pp. 629–647.

[19] S. Bubeck, “Convex optimization: Algorithms and complexity,” Foundations and trends in Machine Learning, vol. 8, no. 3-4, 2015.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] S. Moro, P. Rita, and B. Vala, “Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach.” Journal of Business Research, vol. 69, no. 9, pp. 3341–3351, 2016.

[22] H. Kahraman, S.Sagiroglu, and I.Colak, “Developing intuitive knowledge classifier and modeling of users&apos; domain dependent data in web,” Knowledge Based Systems, vol. 37, pp. 283–295, 2013.

[23] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz, “Revisiting distributed synchronous SGD,” in Proc. of ICLR Workshop Track, 2016.

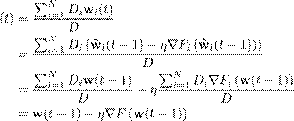
APPENDIX

*A. Distributed vs. Centralized Gradient Descent*

Proposition 2. When = 1(t)*τ , Algorithm 1 yields the following recurrence relation for* **w***:*

**w**(t) = ww(t − 1) − η∇F ((t − 1)) (16)

*Proof.* When = 1, we have ei(t) = w(t) for all . Thus,*τ* **w***t*

**w**

where the second term in the last equality is because



due to the linearity of the gradient operator. 

Eq. (16) is the recurrence relation for centralized gradient decent on the global loss (w). Therefore, the distributed gradient descent algorithm presented in Algorithm 1 is optimal (when comparing to centralized gradient descent) for = 1.*Fτ*

*B. Proof of Theorem 1*

To prove Theorem 1, we first introduce the following lemma.

Lemma 2. For any interval [k]∈ [(k − 1)τ,kτ)*, and t , we have*



*where we define the function gi*(x) *as*



*Proof.* We show by induction that (k − 1)τ) for all ∈ ((k − 1)τ,kτ].*t*

When = (k − 1)τ*t* , we know that by the definition of **v**[k](t), and we have .

We note that because there is no global aggregation within this interval. Combining this with (4), for *t* ∈ ((k − 1)τ,kτ), we have ei(t) = w(t − 1) − η∇Fi(w(t − 1)) (17)**w**e*i*e*i*

For the induction, we assume that



holds for some ∈ ((k − 1)τ,kτ)*t* . We now show that holds for *t*. We have

|  |
| --- |
|  |
|  |  |

      (from (7), (17))

(adding a zero term and rearranging)





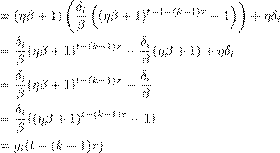
(from triangle inequality)

e

(from the -smoothness of (·) and (8))*βFi*

≤ (ηβ + 1)gi(t − 1 − (k − 1)τ) + ηδi

(from the induction assumption in (18))



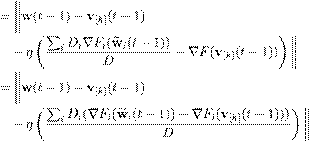
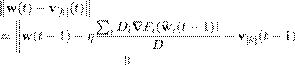
Using the above induction, we have shown that for all *t* ∈ [(k − 1)τ,kτ). 

We are now ready to prove Theorem 1.

*Proof.* (Theorem 1)

From (4) and (5), we have

**w**

Then, for ∈ ((k − 1)τ,kτ], we have*t*

**v**

!

(from triangle inequality)

!

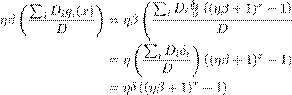
(because (·) is -smooth)*Fβ*



(from Lemma 2)

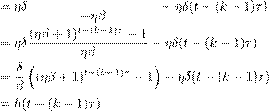
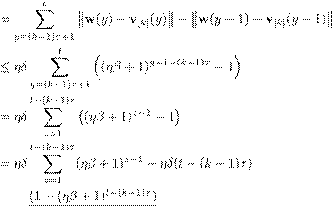


where the last equality is because for any ,*x*

!

Equivalently,

(20)

When = (k − 1)τ, we have (t) = v[k](t) *t* **wv**according to the definition, thus. For ∈ ((k − 1)τ,kτ], by summing up (20) over different values of , we have*t t*



*C. Proof of Theorem 2*

To prove Theorem 2, we first introduce the following definitions and lemmas.

Definition 2. For an interval [k](t) =*, we define θ*[k]

*F*(v[k](t)) − F(w∗)(k − 1)τ ≤ t ≤ kτ*, for a fixed k, t is defined between .*

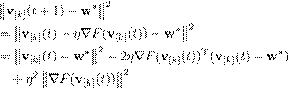
According to the convergence lower bound of gradient descent given in [19, Theorem 3.14], we always have

*θ*[k](t) > 0 (21)

for any finite and .*t k*

Lemma 3. When , for any *k, and t* ∈ [(k − 1)τ,kτ]*, we have that* *does not increase with t, where* **w**∗ *is the optimal parameter defined in (3).*

*Proof.*



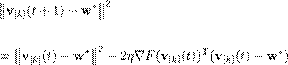
Because (·) is -smooth, from (21) and [19, Lemma 3.5], we have*Fβ*



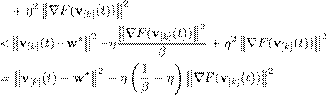
Thus,



Therefore,

(from (7))

(expanding the squared norm)



When, we obtain





Lemma 4. For any ≤ β1 ∈ [(k − 1)τ,kτ]*k, when η and t , we have*

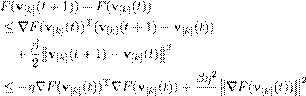
(22)

*where*

*Proof.* Because (·) β-smooth, from [19, Lemma 3.4], we have*F*



for arbitrary and . Substituting this into (7) yields**x y**

2

(from (7))

(23)

By definition, (t) = F(v[k](t)) − F(w∗) and (t + 1) = F(v[k](t+1))−F(w∗) Substituting this into (23) yields*θ*[k]*θ*[k]



Equivalently,

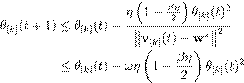


The convexity condition gives



where the last inequality is from the Cauchy-Schwarz inequality. Hence,

(25) Substituting (25) into (24), we get



where the last inequality in the above is explained as follows.

From Lemma 3, we know that for each interval of [k], does not increase with *t* when ∈ [(k −*t*

1)τ,kτ]. Hence, .

Recall that we defined, we have

and the inequality

follows.

As (t + 1)θ[k](t) > 0 according to (21), dividing both sides by (t + 1)θ[k](t), we obtain*θ*[k]*θ*[k]



We have 0 < θ[k](t + 1) ≤ θ[k](t) from (21) and (24), thus

. Hence,



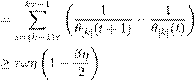


We are now ready to prove Theorem 2.

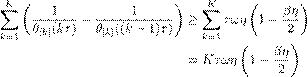
*Proof.* (Theorem 2)

Using Lemma 4 and considering ∈ [(k−1)τ,kτ], we have*t*

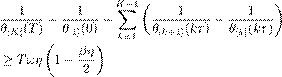




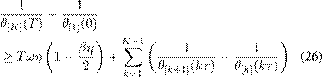
Summing up the above for all = 1,2...,K yields*k*



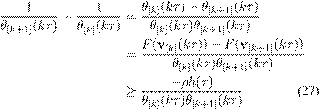
Rewriting the left-hand side and noting that = Kτ yields*T*



which is equivalent to



Each term in the sum in right-hand side of (26) can be further expressed as



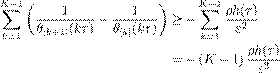
where the last inequality is obtained using Lemma 1 and noting that, according to the definition, (kτ) = w(kτ), thus**v**[k+1]

*F*(v[k+1](kτ)) = F(w(kτ)).

It is assumed that (t) = F(v[k](t))−F(w∗) ≥ ε for all and for which (t) is defined. Consequently,*θ*[k]*t k* **v**[k]

(28)

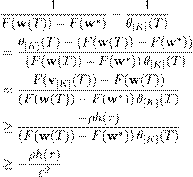
Combining (28) with (27), the sum in the right-hand side of (26) can be bounded by



It is also assumed that (w(T)) − F(w∗) ≥ ε. Using the same argument as for obtaining (28), we have*F*

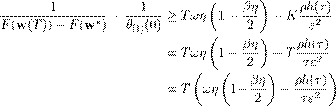
(30)

We then have

(31)

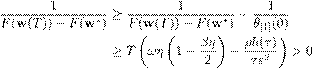
where the first inequality is from Lemma 1 and the second inequality is from (30).

Summing up (29) and (31), we have



where the first equality is because.

We note that



where the first inequality is because (0) = F(v[1](0)) − F(w∗) > 0*θ*[1], and the last inequality is due to the assumption that. Taking the reciprocal of the above inequality yields



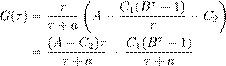


|  |
| --- |
| ! |

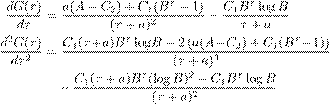
*D. Proof of Proposition 1 Proof.* We note that

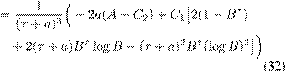
We define

. Then, we can rewrite (τ) as*G*



Taking the derivative, we get



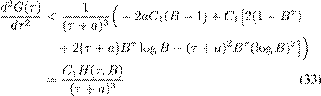


where log denotes the natural logarithm.

When ≤ β1 , we always have − C1(B − 1) − C2 =*η A*

because (1) = 0 *h*and

. Thus, −C2 > C1(B−1). Combining this with (32), we have*A*



where we define

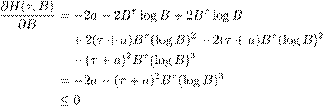
*H*(τ,B) , −2a(B − 1) + 2(1 − Bτ) + 2(τ + a)Bτ logB

− (τ + a)2Bτ(logB)2

We note that ≥ 0 and (τ + a)3 > 0*C*1 . From (33), to show that , we only need to show that *H*(τ,B) ≤ 0. When = 1, we have*B*

*H*(τ,1) = 0

for any . When ≥ 1, we have*τB*



which means that (τ,B) is a non-increasing function for any fixed and varying ≥ 1.*Hτ B*

We always have ≥ 1 according to the definition. We can thus conclude that we always have (τ,B) ≤ 0 for ≥ 1 and ≥ 1*B Hτ B* . It follows that and hence *G*(τ) is a strictly concave function. 

*E. Instantaneous Results of SVM (SGD)*

The instantaneous results of SVM (SGD) are shown in

Fig. 8.

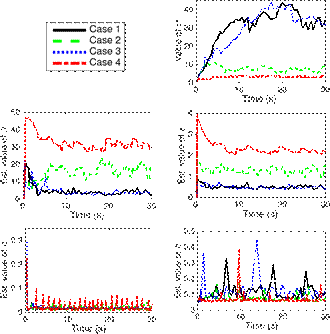


Fig. 8: Instantaneous results of SVM (SGD) with the proposed algorithm.

[[1]](" \l "_ftnref1" \o ") Note that some unsupervised models (such as K-means) only learn on and do not require the existence of in the training data. In such cases, the loss function value only depends on .**xx***j yj j*

[[2]](" \l "_ftnref2" \o ") While our focus is on non-probabilistic learning models, similar loss functions can be defined for probabilistic models where the goal is to minimize the negative of the log-likelihood function, for instance.

[[3]](" \l "_ftnref3" \o ") See the condition in Line 13 of Algorithm 2 and Lines 5 and 14 of Algorithm 3. Also note that the parameters and ∇Fi(w(t0)) sent in Line 15 of Algorithm 3 are obtained at the previous global aggregation step (and are obtained in Lines 4–6 of Algorithm 3).*β*ˆ*i t*0 *β*ˆ*i*

[[4]](" \l "_ftnref4" \o ") The CNN has 7 layers with the following structure: 5 × 5 × 32 Convolutional → 2 × 2 MaxPool → 5 × 5 × 32 Convolutional → 2 × 2 MaxPool → 1568 × 256 Fully connected → 256 × 10 Fully connected → Softmax.

[[5]](" \l "_ftnref5" \o ") When there are more labels than nodes, each node may have data with more than one label, but the number of labels at each node is no more than the total number of labels divided by the total number of nodes rounded to the next integer.

[[6]](" \l "_ftnref6" \o ") Note that the loss and accuracy values shown in Fig. 4 can be improved if we allow a longer training time. For example, the accuracy of CNN on MNIST data can become close to 1.0 if we allow a long enough time for training. The goal of our experiments here is to show that our proposed approach can operate close to the optimal point with a fixed and limited amount of training time (resource budget) as defined at the beginning of Section VII-B1.