(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 110909865 B
(45)授权公告日 2022.08.30

(21)申请号 201911125638.7

(22)申请日 2019.11.18

(65)同一申请的已公布的文献号
申请公布号 CN 110909865 A

(43)申请公布日 2020.03.24

(73)专利权人 福州大学
地址 350108 福建省福州市闽侯县福州大学城乌龙江北大道2号福州大学

(72)发明人 郑海峰 高敏 马金凤 冯心欣

(74)专利代理机构 福州元创专利商标代理有限公司 35100
专利代理师 陈明鑫 蔡学俊

(51)Int.Cl.
G06N 3/04(2006.01)
G06N 3/08(2006.01)
G06N 20/00(2019.01)
G06F 9/50(2006.01)

(56)对比文件
CN 109740588 A,2019.05.10
CN 110266771 A,2019.09.20
CN 107871160 A,2018.04.03
CN 107798697 A,2018.03.13
CN 110309847 A,2019.10.08
CN 110211064 A,2019.09.06
US 2019114547 A1,2019.04.18
王磊 等.面向嵌入式应用的深度神经网络模型压缩技术综述.《北京交通大学学报》.2017,第41卷(第06期),第34-41页.
纪荣嵘 等.深度神经网络压缩与加速综述.《计算机研究与发展》.2018,第55卷(第09期),第1871-1888页.
Haifeng Zheng et al..A Distributed Hierarchical Deep Computation Model for Federated Learning in Edge Computing.《IEEE Transactions on Industrial Informatics》.2021,第17卷(第12期),第7946-7956页.

审查员 钟艺雯

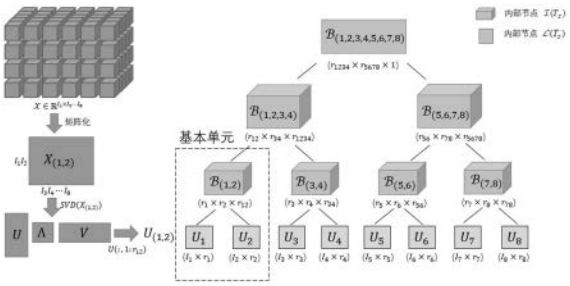
权利要求书3页 说明书8页 附图3页

(54)发明名称

边缘计算中基于分层张量分解的联邦学习方法

(57)摘要

本发明涉及一种边缘计算中基于分层张量分解的联邦学习方法。步骤S1:在云端设计有效的深度神经网络共享模型;步骤S2:根据分层张量分解方法对设计的共享模型进行压缩得到分层共享模型;步骤S3:设计分层共享模型对应的正向传播算法和反向传播算法;步骤S4:在云端对分层共享模型进行初始化并下发至参与训练的边缘节点;步骤S5:参与训练的边缘节点利用本地数据集,并根据S3设计的算法对S2得到的分层共享模型进行学习。步骤S6:在云端通过平均聚合的方式对边缘模型进行聚合。本发明在保护用户隐私的前提下实现了共享模型的分布式训练,减少分布式训练时对网络带宽的需求,降低了边缘节点的通信能耗。



CN 110909865 B

1.一种边缘计算中基于分层张量分解的联邦学习方法，其特征在于，包括如下步骤：

步骤S1、在云端设计深度神经网络共享模型；

步骤S2、根据分层张量分解方法对步骤S1设计的深度神经网络共享模型进行压缩得到分层共享模型；

步骤S3、设计分层共享模型对应的正向传播算法和反向传播算法；

步骤S4、在云端对分层共享模型进行初始化并下发至参与训练的边缘节点；

步骤S5、参与训练的边缘节点利用本地数据集，并根据步骤S3设计的正向传播算法和反向传播算法对步骤S2得到的分层共享模型进行学习；

步骤S6、在云端通过平均聚合的方式对边缘模型进行聚合；

所述步骤S2的具体实现过程如下：

步骤S21、对神经网络的超参$\theta \equiv \{w^{(q)}; b^{(q)} \mid q \in \{1,2,3\}\}$中$w^{(q)}_{q\in\{1,3\}}$进行分层张量分解；假设$w \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$，该张量有d个模M＝{1,2,…,d}，其对应的满秩二叉树为$T_{\mathcal{I}}$，其中满秩二叉树中的每个节点用模M＝{1,2,…,d}的子集表示；设定二叉树从根节点到叶节点的层数l为0到$[\log_2 d]$，则第l层的节点代表的模个数为$\frac{d}{2^l}$，并且该层所有节点代表的模的集合为M的全集；

步骤S22、根据步骤S21所得到的满秩二叉树中各节点所表示的模对张量$\mathcal{X} \in R^{I_1 \times I_2 \times \cdots \times I_d}$进行模展开后进行SVD分解，取其左奇异值的前$k_t$列作为该节点的$U_t$，其中$k_t$满足以下关系：

$$\forall t \in T_n: \ k_t \coloneqq rank(X_{(t)})$$

步骤S23、定义满秩二叉树的任意内部节点为t，其对应的两个子节点分别为t_l和t_r，则t，t_l和t_r三个节点上的U_t满足以下关系：

$$U_t = (U_{t_l} \otimes U_{t_r})B_t$$

其中$B_t \in R^{k_t \times k_{t_l} k_{t_r}}$为节点t上的传输矩阵；因此，$w^{(q)}_{q\in\{1,3\}}$能够被表示为分层分解的形式：

$$w^{(q)}_{q\in\{1,3\}} = \bigotimes_{t\in\mathcal{I}(T_{\mathcal{I}})} (U^{(q)}_{t_l} \otimes U^{(q)}_{t_r})B^{(q)}_t ;$$

步骤S24、利用分层张量分解方法对深度神经网络共享模型进行压缩得到分层共享模型：

$$\mathcal{F} = f\Big(\sum_{j\in l}\sum_{i\in l-1}\Big(\bigotimes_{t\in\mathcal{I}(T_{\mathcal{I}})}(U^{(1)}_{t,ij} \otimes U^{(1)}_{t,ij})B^{(1)}_{tij}\Big)x_i + b^{(1)}_j\Big)$$

$$\mathcal{C} = f\Big(\sum_{j\in l}\Big(\bigotimes_{t\in\mathcal{I}(T_{\mathcal{I}})}(U^{(3)}_{t_l j} \otimes U^{(3)}_{t_r j})B^{(3)}_{tj}\Big)x + b^{(3)}_j\Big)$$

此时神经网络的学习参数为$\theta \equiv \{U^q_t, B^q_t, b^p, w^{(2)} \mid q \in \{1,3\}, p \in \{1,2,3\}, t \in \mathcal{I}(T_{\mathcal{I}})\}$；

所述步骤S3的具体实现过程如下：

步骤S31、利用训练集数据作为神经网络的输入，根据分层共享模型得到神经网络的输出作为预测值；

步骤S32、设计节点k上神经网络的损失函数

$$J_k(\theta)=\frac{1}{2}\sum_{m=1}^{n_k}(h_\theta(x_k^m)-y_k^m)^2$$

其中$\{(x_k^m,y_k^m)\,|\,m\in n_k\}$为节点k上数据集$n_k$中的样本；

步骤S33、通过梯度下降法使得步骤S32中设计的损失函数最小化；

若l为输出层，则

$\varepsilon^{(l)}=(f(z^{(l)})-y)f'(z^{(l)})$

若l为卷积层，则

$\varepsilon^{(l)}=up(\varepsilon^{(l+1)})w^{(l)}f'(z^{(l)})$

$$\nabla b=\varepsilon^{(l)}$$

$$\nabla w^{(l)}=\sum_{j\in l}\sum_{i\in l-1}f(z^{(l-1)})_i\varepsilon_j^{(l)}$$

若l为池化层，则

$\varepsilon^{(l)}=up(\varepsilon^{(l+1)})w^{(2)}f'(z^{(l)})$

$$\nabla b=\varepsilon^{(l)}$$

$$\nabla w^{(2)}=\sum_{j\in l}up(f(z^{(l-1)})_j)\varepsilon_j^{(l)}$$

若l为全连接层，则

$\varepsilon^{(l)}=\varepsilon^{(l+1)}w^{(3)}f'(z^{(l)})$

$$\nabla b=\varepsilon^{(l)}$$

$$\nabla w^{(3)}=\sum_{j\in l}f(z^{(l-1)})_i\varepsilon_j^{(l)}$$

若l为分层张量表示，则

$$\nabla B_t^{(m)}=\frac{\partial((U_{t_l}\otimes U_{t_l})B_t)}{\partial B_t^{(m)}}=\nabla U_t^{(l-1)}\cdot(U_{t_l}\otimes U_{t_r})$$

$$\nabla U_{t_{l,r}}^{(m)}=\frac{\partial((U_{t_l}\otimes U_{t_l})B_t)}{\partial U_{t_{l,r}}^{(m)}}=\nabla U_{t_{l,r}}^{(l-1)}\cdot A\cdot B_t$$

假设g(x)是关于x的函数，则g(x)对x求导可表示为$\frac{\partial g(x)}{\partial x}$，因此上述表达式中$\varepsilon^{(l)}$是输出层对l层的梯度，$z^{(l)}$为l层的输出，f(•)为sigmoid激活函数，A为克罗内克积⊗积的梯度；

假设$X_1\in\mathbb{R}^{a\times b},X_2\in\mathbb{R}^{c\times d}$，则

$$A=\frac{\partial(X_1\otimes X_2)}{\partial X_{m\in\{1,2\}}}=\frac{\partial Y}{\partial X_{m\in\{1,2\}}}=\sum_{i=1}^{ac}\sum_{j=1}^{bd}\frac{\partial Y(i,:)}{\partial X_m(:,j)}$$

步骤S34、通过步骤S33中得到的梯度$\nabla\theta$，采用η的学习率对模型进行更新

$\theta^*=\theta-\eta\nabla\theta$。

2.根据权利要求1所述的边缘计算中基于分层张量分解的联邦学习方法，其特征在于，所述步骤S1的具体实现过程如下：

构造一个深度神经网络共享模型，包括卷积层、池化层和全连接层，其对应的表达式如下：

$$\mathcal{F}(w^{(1)};b^{(1)})=f(\sum_{j\in l}\sum_{i\in l-1}w_{ij}^{(1)}x_i+b_j^{(1)})$$

$$\mathcal{D}(w^{(2)};b^{(2)})=f(\sum_{j\in l}w_j^{(2)}pooling(x_j)+b_j^{(2)})$$

$$\mathcal{C}(w^{(3)};b^{(3)})=f(\sum_{j\in l}w_j^{(3)}x+b_j^{(3)})$$

其中，$w_{ij}^{(1)}$表示输入层l-1第i个神经元x_i和输出层l第j个神经元的权重，b_j表示输出层第j个神经元的偏置，$\mathcal{F}(\cdot)$、$\mathcal{D}(\cdot)$和$\mathcal{C}(\cdot)$分别表示卷积层的特征提取器、池化层的分类器和全连接层的分类器，其中$\theta\equiv\{w^{(q)};b^{(q)}|q\in\{1,2,3\}\}$为神经网络的超参，pooling(•)为池化层的降采样操作。

3.根据权利要求1所述的边缘计算中基于分层张量分解的联邦学习方法，其特征在于，云端在所有具有富余计算力的边缘节点中随机选取K个节点作为参与对象，并将分层共享模型下发。

4.根据权利要求1所述的边缘计算中基于分层张量分解的联邦学习方法，其特征在于，所有参与训练的边缘节点利用本地数据分别独立对分层共享模型进行学习。

5.根据权利要求2所述的边缘计算中基于分层张量分解的联邦学习方法，其特征在于，各个参与计算的边缘节点在学习完成后分别将本地模型发至云端，云端通过平均聚合的方式更新全局模型：

$$\theta^{t+1}=\sum_{k=1}^{K}\frac{n_k}{n}\theta_k^{t+1}$$

进一步根据以上更新的模型重新随机选取K个节点开始新一轮训练，从而实现共享模型的更新；其中θ_k^{t+1}为在第t+1轮通信中第k个节点上的模型，n_k对应为第k个节点上的本地训练数据的样本数。

边缘计算中基于分层张量分解的联邦学习方法

技术领域

[0001] 本发明涉及一种边缘计算中基于分层张量分解的联邦学习方法

背景技术

[0002] 随着物联网技术的飞速发展及其在智能工厂、工业自动化、智能制造等工业领域的广泛应 用,工业物联网技术受到了学术界和工业界的广泛关注。在工业物联网中,各种连接设备生成 的数据呈爆炸式增长。然而,将大量数据直接传输到远程云端平台进行进一步的处理和分析是 不切实际的,这可能会导致严重的网络拥塞和无法忍受的传输延迟。近年来,随着边缘计算技 术的兴起,传感器、工厂网关等边缘设备(节点)具有存储、处理和分析本地数据的能力。此外, 边缘设备还可以与远程云端协作,执行大规模、复杂的任务。

[0003] 与此同时,近年来,深度学习在许多应用中也取得了巨大的成功,尤其是在大数据分析和 机器学习方面。深度学习模型使用多层体系结构从大量原始数据中自动学习固有的特性。然而, 在边缘设备上训练深度学习模型存在以下主要缺点:一方面,由于人们对数据安全和用户隐私 的意识越来越强,将每个边缘设备上的本地数据集上传到云端服务器上存在着数据泄露的风险。 例如在大多数行业,不同的公司甚至同一公司的不同部门之间都禁止共享数据,因此这在现实 生活中并不可行。另一方面,由于边缘设备的计算能力和存储能力有限,在这些低端设备上进 行模型学习非常困难。这是因为深度学习模型的参数非常大,训练这样的模型通常需要昂贵的 硬件资源。

[0004] 为了解决上述的数据安全问题,谷歌最近提出了联邦学习概念,通过将云端计算下沉到各 个边缘节点,从而避免了传输用户数据带来的数据泄露的风险。然而,由于多节点上的分布式 训练需要梯度交换,因此联邦学习需要较大的通信带宽。为了克服联合学习中的交流瓶颈,人 们进行了许多研究。这些方法可以分为三类:第一类是梯度稀疏化方法,根据预定义的梯度阈 值或以固定的稀疏率只选择一小部分参数进行更新。第二类方法是通过梯度量化将梯度量化到 低精度值来降低通信带宽。例如,随机梯度量化方法中对每个参数只取2bits。最后一种方法是 通过降低通信频率来降低通信带宽。例如近似同步并行算法中,只有当参数变化超过预定义的 阈值时才执行聚合。

[0005] 与上述工作不同的是,本发明从权值张量的低秩表示的角度来降低联邦学习中的通信带宽。针对分布式训练中的深度卷积计算模型,提出了一种基于分层分解方法,在压缩效率和分类精 度之间取得了较好的平衡。该方案的优点在于能够利用卷积网络对应于广义分层张量分解的特 性,其中卷积和输出层的网络权值可以直接映射到各自的分层张量分解的参数。一方面,由于 神经网络存在大量的冗余信息,浪费了网络传输的带宽资源和设备的存储资源,本方案利用分 层张量分解将模型参数从高阶张量空间压缩为低维空间,降低了边缘节点分布式训练的带宽消 耗和存储需求。另一方面本方案提出了一种基于梯度下降的分层张量分解模型的反向传播更新 算法,在边缘节点上训练卷积计算模型的参数。该方法采用分层方式直接计算低维参数的梯度, 减少了对边缘设备计算力的消耗。因此,在边缘计算中,利用张量分解方法进行模型压缩从而减 少系统的能量损耗具有潜在

的优势。

发明内容

[0006] 本发明的目的在于提供一种边缘计算中基于分层张量分解的联邦学习方法，在保护用户隐 私的前提下实现了多用户的数据共享，减少分布式训练时对网络带宽的需求，降低了边缘节点 的通信能耗。

[0007] 为实现上述目的，本发明的技术方案是：一种边缘计算中基于分层张量分解的联邦学习方 法，包括如下步骤：

[0008] 步骤S1、在云端设计深度神经网络共享模型；

[0009] 步骤S2、根据分层张量分解方法对步骤S1设计的深度神经网络共享模型进行压缩得到分 层共享模型；

[0010] 步骤S3、设计分层共享模型对应的正向传播算法和反向传播算法；

[0011] 步骤S4、在云端对分层共享模型进行初始化并下发至参与训练的边缘节点；

[0012] 步骤S5、参与训练的边缘节点利用本地数据集，并根据步骤S3设计的正向传播算法和反 向传播算法对步骤S2得到的分层共享模型进行学习；

[0013] 步骤S6、在云端通过平均聚合的方式对边缘模型进行聚合。

[0014] 在本发明一实施例中，所述步骤S1的具体实现过程如下：

[0015] 构造一个深度神经网络共享模型，包括卷积层、池化层和全连接层，其对应的表达式如下：

[0016] $$\mathcal{F}(w^{(1)};b^{(1)})=f(\sum_{j\in l}\sum_{i\in l-1}w_{ij}^{(1)}x_i+b_j^{(1)})$$

[0017] $$\mathcal{D}(w^{(2)};b^{(2)})=f(\sum_{j\in l}w_j^{(2)}pooling(x_j)+b_j^{(2)})$$

[0018] $$\mathcal{C}(w^{(3)};b^{(3)})=f(\sum_{j\in l}w_j^{(3)}x+b_j^{(3)})$$

[0019] 其中，$w_{ij}^{(l)}$表示输入层l-1第i个神经元x_i和输出层l第j个神经元的权重，b_j表示输出层第 j个神经元的偏置，$\mathcal{F}(\cdot)$、$\mathcal{D}(\cdot)$和$\mathcal{C}(\cdot)$分别表示卷积层的特征提取器、池化层的分类器和全连接 层的分类器，其中$\theta\equiv\{w^{(q)};b^{(q)}\mid q\in\{1,2,3\}\}$为神经网络的超参，pooling(•)为池化层的降采样操 作。

[0020] 在本发明一实施例中，所述步骤S2的具体实现过程如下：

[0021] 步骤S21、对$\theta\equiv\{w^{(q)};b^{(q)}\mid q\in\{1,2,3\}\}$中$w_{q\in\{1,3\}}^{(q)}$进行分层张量分解；假设$w\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_d}$，该 张量有d个模M＝{1,2,…,d}，其对应的满秩二叉树为T_I，其中满秩二叉树中的每个节点用模 M＝{1,2,…,d}的子集表示；设定二叉树从根节点到叶节点的层数l为0到[$\log_2$ d]，则第l层的 节点代表的模个数为$\frac{d}{2^l}$，并且该层所有节点代表的模的集合为M的全集；

[0022] 步骤S22、根据步骤S21所得到的满秩二叉树中各节点所表示的模对张量$\mathcal{X}\in R^{I_1\times I_2\times\cdots\times I_d}$进行 模展开后进行SVD分解，取其左奇异值的前$k_t$列作为该节点的$U_t$，其中$k_t$

满足以下关系：

[0023] $\forall t \in T_n : \ k_t := rank(\mathrm{X}_{(t)})$

[0024] 步骤S23、定义满秩二叉树的任意内部节点为t，其对应的两个子节点分别为t_l和t_r，则t，t_l和t_r三个节点上的U_t满足以下关系：

[0025] $$\mathrm{U}_t = (\mathrm{U}_{t_l} \otimes \mathrm{U}_{t_r})\mathrm{B}_t$$

[0026] 其中$B_t \in R^{k_t \times k_{t_l} k_{t_r}}$为节点t上的传输矩阵；因此，$w^{(q)}_{q\in\{1,3\}}$能够被表示为分层分解的形式：$w^{(q)}_{q\in\{1,3\}} = \bigotimes_{t\in\mathcal{I}(T_{\mathcal{I}})} (\mathrm{U}^{(q)}_{t_l} \otimes \mathrm{U}^{(q)}_{t_r})\mathrm{B}^{(q)}_t$；

[0027] 步骤S24、利用分层张量分解方法对深度神经网络共享模型进行压缩得到分层共享模型：

[0028] $$\mathcal{F} = f(\sum_{j\in l}\sum_{i\in l-1}(\bigotimes_{t\in\mathcal{I}(T_{\mathcal{I}})}(\mathrm{U}^{(1)}_{t,ij} \otimes \mathrm{U}^{(1)}_{t,ij})\mathrm{B}^{(1)}_{tij})x_i + b^{(1)}_j)$$

[0029] $$\mathcal{C} = f(\sum_{j\in l}(\bigotimes_{t\in\mathcal{I}(T_{\mathcal{I}})}(\mathrm{U}^{(3)}_{t,j} \otimes \mathrm{U}^{(3)}_{t,j})\mathrm{B}^{(3)}_{tj})x + b^{(3)}_j)$$

[0030] 此时神经网络的学习参数为$\theta \equiv \{U^q_t, B^q_t, b^p, w^{(2)} \mid q \in \{1,3\}, p \in \{1,2,3\}, t \in \mathcal{I}(T_{\mathcal{I}})\}$。

[0031] 在本发明一实施例中，所述步骤S3的具体实现过程如下：

[0032] 步骤S31、利用训练集数据作为神经网络的输入，根据分层共享模型得到神经网络的输出 作为预测值；

[0033] 步骤S32、设计节点k上神经网络的损失函数

[0034] $$J_k(\theta) = \frac{1}{2}\sum\nolimits_{m=1}^{n_k}(h_\theta(x^m_k) - y^m_k)^2$$

[0035] 其中$\{(x^m_k, y^m_k) \mid m \in n_k\}$为节点k上数据集$n_k$中的样本；

[0036] 步骤S33、通过梯度下降法使得步骤S32中设计的损失函数最小化；

[0037] 若l为输出层，则

[0038] $\varepsilon^{(l)} = (f(z^{(l)}) - y) f'(z^{(l)})$

[0039] 若l为卷积层，则

[0040] $\varepsilon^{(l)} = up(\varepsilon^{(l+1)}) w^{(l)} f'(z^{(l)})$

[0041] $\nabla b = \varepsilon^{(l)}$

[0042] $$\nabla w^{(l)} = \sum_{j\in l}\sum_{i\in l-1} f(z^{(l-1)})_i \varepsilon^{(l)}_j$$

[0043] 若l为池化层，则

[0044] $\varepsilon^{(l)} = up(\varepsilon^{(l+1)}) w^{(2)} f'(z^{(l)})$

[0045] $\nabla b = \varepsilon^{(l)}$

[0046] $$\nabla w^{(2)} = \sum_{j\in l} up(f(z^{(l-1)})_j)\varepsilon^{(l)}_j$$

[0047] 若l为全连接层，则

[0048] $\varepsilon^{(l)} = \varepsilon^{(l+1)} w^{(3)} f'(z^{(l)})$

[0049] $\nabla b = \varepsilon^{(l)}$

[0050] $$\nabla w^{(3)} = \sum_{j \in l} f(z^{(l-1)})_i \varepsilon_j^{(l)}$$

[0051] 若l为分层张量表示，则

[0052] $$\nabla B_t^{(m)} = \frac{\partial((U_{t_l} \otimes U_{t_l}) B_t)}{\partial B_t^{(m)}} = \nabla U_t^{(l-1)} \cdot (U_{t_l} \otimes U_{t_r})$$

[0053] $$\nabla U_{t_{l,r}}^{(m)} = \frac{\partial((U_{t_l} \otimes U_{t_l}) B_t)}{\partial U_{t_{l,r}}^{(m)}} = \nabla U_{t_{l,r}}^{(l-1)} \cdot A \cdot B_t$$

[0054] 假设g(x)是关于x的函数，则g(x)对x求导可表示为$\frac{\partial g(x)}{\partial x}$，因此上述表达式中$\varepsilon^{(1)}$是输出 层对l层的梯度，$z^{(1)}$为l层的输出，f(•)为sigmoid激活函数，A为克罗内克积⊗积的梯度；

[0055] 假设$X_1 \in \mathbb{R}^{a \times b}, X_2 \in \mathbb{R}^{c \times d}$，则

[0056] $$A = \frac{\partial(X_1 \otimes X_2)}{\partial X_{m \in \{1,2\}}} = \frac{\partial Y}{\partial X_{m \in \{1,2\}}} = \sum_{i=1}^{ac} \sum_{j=1}^{bd} \frac{\partial Y(i,:)}{\partial X_m(:,j)}$$

[0057] 步骤S34、通过步骤S33中得到的梯度▽θ，采用η的学习率对模型进行更新

[0058] $\theta^* = \theta - \eta \nabla \theta$。

[0059] 在本发明一实施例中，云端在所有具有富余计算力的边缘节点中随机选取K个节点作为参 与对象，并将分层共享模型下发。

[0060] 在本发明一实施例中，所有参与训练的边缘节点利用本地数据分别独立对分层共享模型进 行学习，从而避免了将数据发至云端集中化进行处理导致的数据泄露及网络负载过大等问题，保证了用户数据的隐私性。

[0061] 在本发明一实施例中，各个参与计算的边缘节点在学习完成后分别将本地模型发至云端，云端通过平均聚合的方式更新全局模型：

[0062] $$\theta^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \theta_k^{t+1}$$

[0063] 进一步根据以上的更新模型重新随机选取K个节点开始新一轮训练，从而实现共享模型的 更新；其中θ_k^{t+1}为在第t+1轮通信中第k个节点上的模型，n_k对应为第k个节点上的本地数据。

[0064] 相较于现有技术，本发明具有以下有益效果：本发明利用分层张量分解方法，将联邦学习 中的神经网络模型的冗余参数进行压缩，大大减少了分布式训练时对网络带宽的需求，降低了 边缘节点的通信能耗。另外，该方案将模型参数从高阶张量空间压缩为低维空间，并通过方案 中基于梯度下降的反向传播更新算法，直接计算低维参数的梯度，减少了边缘设备计算的能耗。

附图说明

[0065] 图1是本发明一实施例分层张量分解示意图。

[0066] 图2是本发明一实施例提供的方法与基于其他张量分解方法压缩率的对比示意

图。

[0067] 图3是本发明一实施例提供的方法与基于其他张量分解方法通信能量对比示意图。

[0068] 图4是本发明一实施例提供的方法与基于其他张量分解方法计算能量对比示意图。

[0069] 图5是本发明一实施例提供的方法与基于其他张量分解方法精度损失对比示意图。

具体实施方式

[0070] 下面结合附图,对本发明的技术方案进行具体说明。

[0071] 本发明提供了一种边缘计算中基于分层张量分解的联邦学习方法,包括如下步骤:

[0072] 步骤S1、在云端设计深度神经网络共享模型;

[0073] 步骤S2、根据分层张量分解方法对步骤S1设计的深度神经网络共享模型进行压缩得到分 层共享模型;

[0074] 步骤S3、设计分层共享模型对应的正向传播算法和反向传播算法;

[0075] 步骤S4、在云端对分层共享模型进行初始化并下发至参与训练的边缘节点;

[0076] 步骤S5、参与训练的边缘节点利用本地数据集,并根据步骤S3设计的正向传播算法和反 向传播算法对步骤S2得到的分层共享模型进行学习;

[0077] 步骤S6、在云端通过平均聚合的方式对边缘模型进行聚合。

[0078] 进一步的,所述步骤S1的具体实现过程如下:

[0079] 构造一个深度神经网络共享模型,包括卷积层、池化层和全连接层,其对应的表达式如下:

[0080] $$\mathcal{F}(w^{(1)};b^{(1)})=f(\sum_{j\in l}\sum_{i\in l-1}w_{ij}^{(1)}x_i+b_j^{(1)})$$

[0081] $$\mathcal{D}(w^{(2)};b^{(2)})=f(\sum_{j\in l}w_j^{(2)}pooling(x_j)+b_j^{(2)})$$

[0082] $$\mathcal{C}(w^{(3)};b^{(3)})=f(\sum_{j\in l}w_j^{(3)}x+b_j^{(3)})$$

[0083] 其中,$w_{ij}^{(1)}$表示输入层l-1第i个神经元x_i和输出层l第j个神经元的权重,b_j表示输出层第 j个神经元的偏置,$\mathcal{F}(\cdot)$、$\mathcal{D}(\cdot)$和$\mathcal{C}(\cdot)$分别表示卷积层的特征提取器、池化层的分类器和全连接 层的分类器,其中$\theta\equiv\{w^{(q)};b^{(q)}\mid q\in\{1,2,3\}\}$为神经网络的超参,pooling(•)为池化层的降采样操 作。

[0084] 此时,设置本方案中神经网络架构为:

[0085] 卷积层$\mathcal{F}:3\times3\times64\rightarrow$卷积层$\mathcal{F}:3\times3\times64\rightarrow$池化层$\mathcal{D}:2\times2\rightarrow$卷积层$\mathcal{F}:3\times3\times64\rightarrow$卷积层$\mathcal{F}:3\times3\times128\rightarrow$池化层$\mathcal{D}:2\times2\rightarrow$卷积层$\mathcal{F}:3\times3\times128\rightarrow$卷积层$\mathcal{F}:3\times3\times128\rightarrow$池化层$\mathcal{D}:4\times4\rightarrow$全连接层$\mathcal{C}:128\times1024\rightarrow$全连接层$\mathcal{C}:1024\times10$。

[0086] 进一步的,请参照图1,所述步骤S2的具体实现过程如下:

[0087] 步骤S21、对$\theta \equiv \{w^{(q)}; b^{(q)} \mid q \in \{1,2,3\}\}$中$w^{(q)}_{q\in\{1,3\}}$进行分层张量分解；假设$w \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$，该 张量有d个模M＝{1,2,…,d}，其对应的满秩二叉树为T_I，其中满秩二叉树中的每个节点用模 M＝{1,2,…,d}的子集表示；设定二叉树从根节点到叶节点的层数l为0到[$\log_2$ d]，则第l层的 节点代表的模个数为$\frac{d}{2^l}$，并且该层所有节点代表的模的集合为M的全集；

[0088] 步骤S22、根据步骤S21所得到的满秩二叉树中各节点所表示的模对张量$\mathcal{X} \in R^{I_1 \times I_2 \times \cdots \times I_d}$进行 模展开后进行SVD分解，取其左奇异值的前$k_t$列作为该节点的$U_t$，其中$k_t$满足以下关系：

[0089] $$\forall t \in T_n : \ k_t \coloneqq rank(\mathrm{X}_{(t)})$$

[0090] 步骤S23、定义满秩二叉树的任意内部节点为t，其对应的两个子节点分别为t_l和t_r，则t， t_l和t_r三个节点上的U_t满足以下关系：

[0091] $$\mathrm{U}_t = (\mathrm{U}_{t_l} \otimes \mathrm{U}_{t_r})\mathrm{B}_t$$

[0092] 其中$B_t \in R^{k_t \times k_{t_l} k_{t_r}}$为节点t上的传输矩阵；因此，$w^{(q)}_{q\in\{1,3\}}$能够被表示为分层分解的形式：$w^{(q)}_{q\in\{1,3\}} = \bigotimes_{t\in\mathcal{I}(T_\mathcal{I})} (\mathrm{U}^{(q)}_{t_l} \otimes \mathrm{U}^{(q)}_{t_r})\mathrm{B}^{(q)}_t$；

[0093] 步骤S24、利用分层张量分解方法对深度神经网络共享模型进行压缩得到分层共享模型：

[0094] $$\mathcal{F} = f(\sum_{j\in l}\sum_{i\in l-1}(\bigotimes_{t\in\mathcal{I}(T_\mathcal{I})}(\mathrm{U}^{(1)}_{t,ij} \otimes \mathrm{U}^{(1)}_{t,ij})\mathrm{B}^{(1)}_{tij})x_i + b^{(1)}_j)$$

[0095] $$\mathcal{C} = f(\sum_{j\in l}(\bigotimes_{t\in\mathcal{I}(T_\mathcal{I})}(\mathrm{U}^{(3)}_{t,j} \otimes \mathrm{U}^{(3)}_{t,j})\mathrm{B}^{(3)}_{tj})x + b^{(3)}_j)$$

[0096] 此时神经网络的学习参数为$\theta \equiv \{U^q_t, B^q_t, b^p, w^{(2)} \mid q \in \{1,3\}, p \in \{1,2,3\}, t \in \mathcal{I}(T_\mathcal{I})\}$。

[0097] 进一步的，所述步骤S3的具体实现过程如下：

[0098] 步骤S31、利用训练集数据作为神经网络的输入，根据分层共享模型得到神经网络的输出 作为预测值；

[0099] 步骤S32、设计节点k上神经网络的损失函数

[0100] $$J_k(\theta) = \frac{1}{2}\sum\nolimits_{m=1}^{n_k}(h_\theta(x^m_k) - y^m_k)^2$$

[0101] 其中$\{(x^m_k, y^m_k) \mid m \in n_k\}$为节点k上数据集$n_k$中的样本；

[0102] 步骤S33、通过梯度下降法使得步骤S32中设计的损失函数最小化；

[0103] 若l为输出层，则

[0104] $\varepsilon^{(l)} = (f(z^{(l)}) - y) f'(z^{(l)})$

[0105] 若l为卷积层，则

[0106] $\varepsilon^{(l)} = up(\varepsilon^{(l+1)}) w^{(l)} f'(z^{(l)})$

[0107] $\nabla b = \varepsilon^{(l)}$

[0108] $$\nabla w^{(1)} = \sum_{j\in l}\sum_{i\in l-1} f(z^{(l-1)})_i \varepsilon_j^{(l)}$$

[0109] 若l为池化层,则

[0110] $\varepsilon^{(l)} = up(\varepsilon^{(l+1)})w^{(2)}f'(z^{(l)})$

[0111] $\nabla b = \varepsilon^{(l)}$

[0112] $$\nabla w^{(2)} = \sum_{j\in l} up(f(z^{(l-1)})_j)\varepsilon_j^{(l)}$$

[0113] 若l为全连接层,则

[0114] $\varepsilon^{(l)} = \varepsilon^{(l+1)}w^{(3)}f'(z^{(l)})$

[0115] $\nabla b = \varepsilon^{(l)}$

[0116] $$\nabla w^{(3)} = \sum_{j\in l} f(z^{(l-1)})_i \varepsilon_j^{(l)}$$

[0117] 若l为分层张量表示,则

[0118] $$\nabla B_t^{(m)} = \frac{\partial((U_{t_l}\otimes U_{t_l})B_t)}{\partial B_t^{(m)}} = \nabla U_t^{(l-1)}\cdot(U_{t_l}\otimes U_{t_r})$$

[0119] $$\nabla U_{t_{l,r}}^{(m)} = \frac{\partial((U_{t_l}\otimes U_{t_l})B_t)}{\partial U_{t_{l,r}}^{(m)}} = \nabla U_{t_{l,r}}^{(l-1)}\cdot A\cdot B_t$$

[0120] 假设g(x)是关于x的函数,则g(x)对x求导可表示为$\frac{\partial g(x)}{\partial x}$,因此上述表达式中$\varepsilon^{(l)}$是输出 层对l层的梯度,$z^{(l)}$为l层的输出,f(•)为sigmoid激活函数,A为克罗内克积⊗积的梯度;

[0121] 假设$X_1\in\mathbb{R}^{a\times b}, X_2\in\mathbb{R}^{c\times d}$,则

[0122] $$A = \frac{\partial(X_1\otimes X_2)}{\partial X_{m\in\{1,2\}}} = \frac{\partial Y}{\partial X_{m\in\{1,2\}}} = \sum_{i=1}^{ac}\sum_{j=1}^{bd}\frac{\partial Y(i,:)}{\partial X_m(:,j)}$$

[0123] 步骤S34、通过步骤S33中得到的梯度$\nabla\theta$,采用η的学习率对模型进行更新

[0124] $\theta^* = \theta - \eta\nabla\theta$。

[0125] 进一步的,所述步骤S4中,云端在所有具有富余计算力的N个边缘节点中随机选取α比 例的节点参与训练,即K=max(α•N,1),并将分层共享模型下发。

[0126] 进一步的,所述步骤S5中,所有参与训练的边缘节点利用本地数据分别独立对分层共享模 型进行学习,从而避免了将数据发至云端集中化进行处理导致的数据泄露及网络负载过大等问 题,保证了用户数据的隐私性。此时,将数据集均匀分成N份,用于模拟边缘节点上的本地数 据集,并且根据各个边缘节点上数据集的特征分布是否一致分成了IID=1和IID=0。例如,每个 边缘节点拥有所有的数据类别时,设置为IID=1;当每个边缘节点只拥有所有类别中的一种或者 几种,则设置为IID=0;

[0127] 进一步的,所述步骤S6中,各个参与计算的边缘节点在学习完成后分别将本地模型发至云 端,云端通过平均聚合的方式更新全局模型:

[0128] $$\theta^{t+1} = \sum_{k=1}^{K}\frac{n_k}{n}\theta_k^{t+1}$$

[0129] 进一步根据以上的更新模型重新随机选取K个节点开始新一轮训练，从而实现共享模型的 更新；其中θ_k^{t+1}为在第t+1轮通信中第k个节点上的模型，n_k对应为第k个节点上的本地数据。

[0130] 本发明方法在压缩效率和分类精度之间取得了较好的平衡，并且对比了其他传统的张量分 解的方法，发现与现有算法相比，本方案在保持相似的精度损失的同时，可以获得更好的压缩 效率，实现对能量的最大化利用。

[0131] 以上是本发明的较佳实施例，凡依本发明技术方案所作的改变，所产生的功能作用未超出 本发明技术方案的范围时，均属于本发明的保护范围。

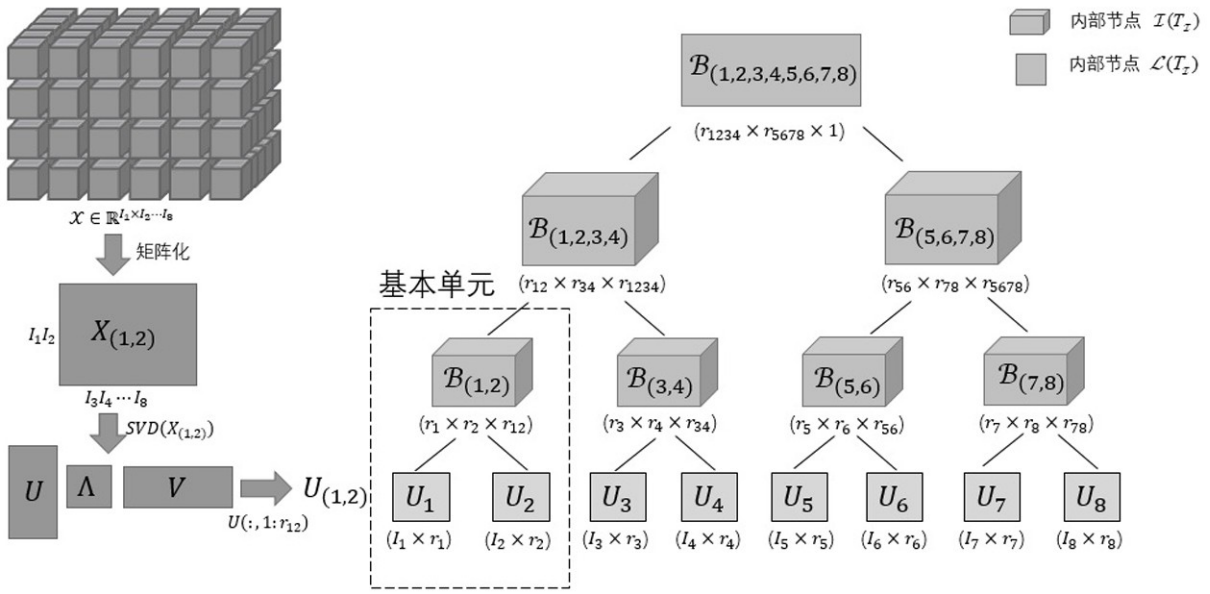

内部节点 $\mathcal{I}(T_\tau)$
内部节点 $\mathcal{L}(T_\tau)$
$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots I_8}$
矩阵化
$I_1 I_2$
$X_{(1,2)}$
$I_3 I_4 \cdots I_8$
$SVD(X_{(1,2)})$
U
Λ
V
$U_{(1,2)}$
$U(:, 1{:}r_{12})$
基本单元
$\mathcal{B}_{(1,2,3,4,5,6,7,8)}$
$(r_{1234} \times r_{5678} \times 1)$
$\mathcal{B}_{(1,2,3,4)}$
$(r_{12} \times r_{34} \times r_{1234})$
$\mathcal{B}_{(5,6,7,8)}$
$(r_{56} \times r_{78} \times r_{5678})$
$\mathcal{B}_{(1,2)}$
$(r_1 \times r_2 \times r_{12})$
$\mathcal{B}_{(3,4)}$
$(r_3 \times r_4 \times r_{34})$
$\mathcal{B}_{(5,6)}$
$(r_5 \times r_6 \times r_{56})$
$\mathcal{B}_{(7,8)}$
$(r_7 \times r_8 \times r_{78})$
U_1 $(I_1 \times r_1)$
U_2 $(I_2 \times r_2)$
U_3 $(I_3 \times r_3)$
U_4 $(I_4 \times r_4)$
U_5 $(I_5 \times r_5)$
U_6 $(I_6 \times r_6)$
U_7 $(I_7 \times r_7)$
U_8 $(I_8 \times r_8)$

图1

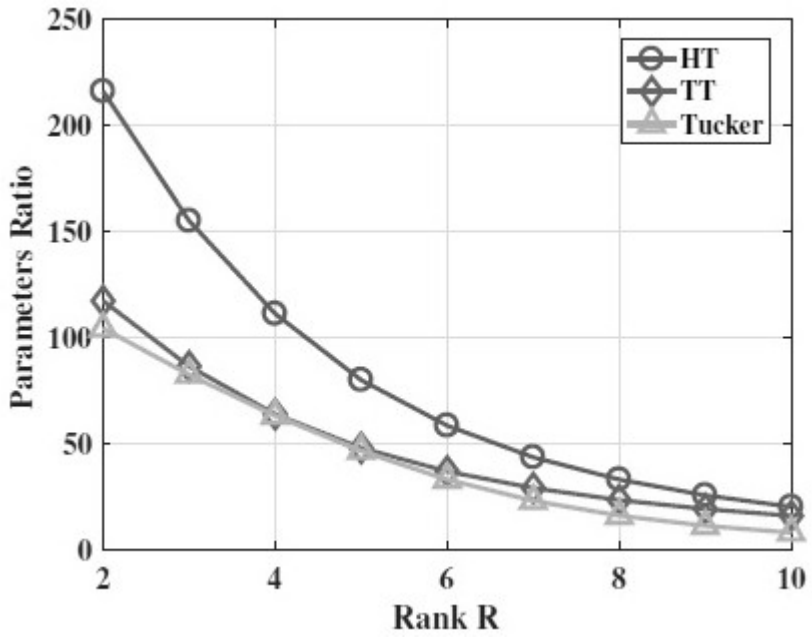

HT
TT
Tucker
Parameters Ratio
Rank R
250
200
150
100
50
0
2
4
6
8
10

图2

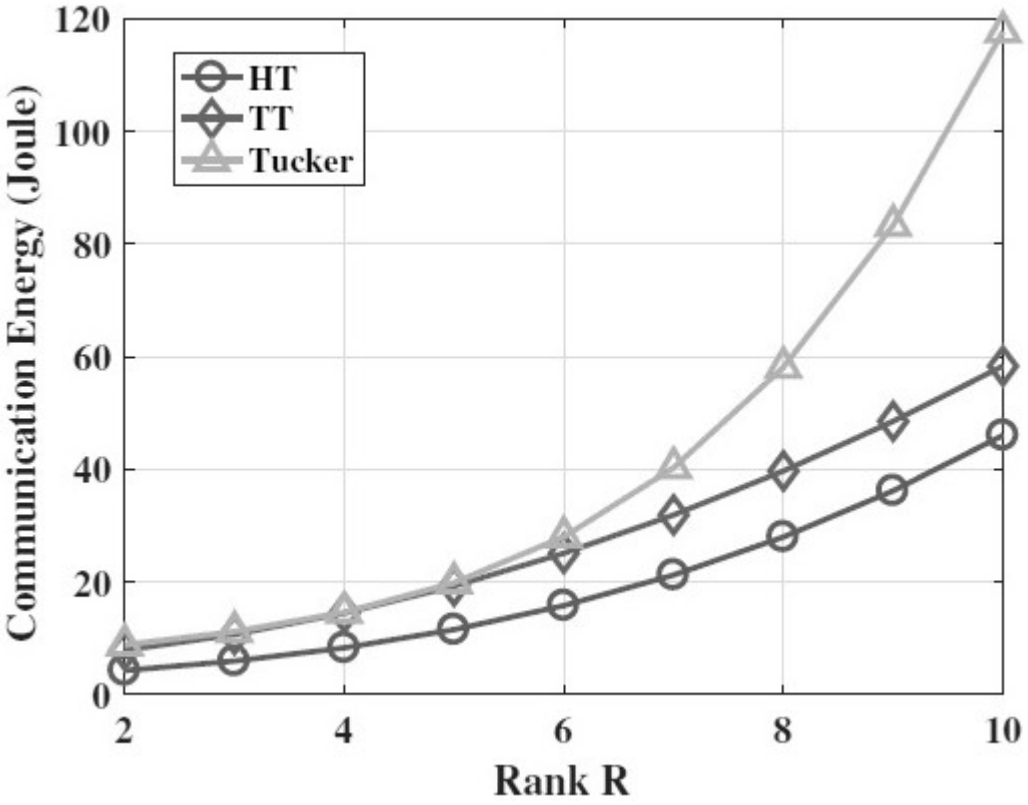

HT
TT
Tucker
Communication Energy (Joule)
Rank R
0
20
40
60
80
100
120
2
4
6
8
10

图3

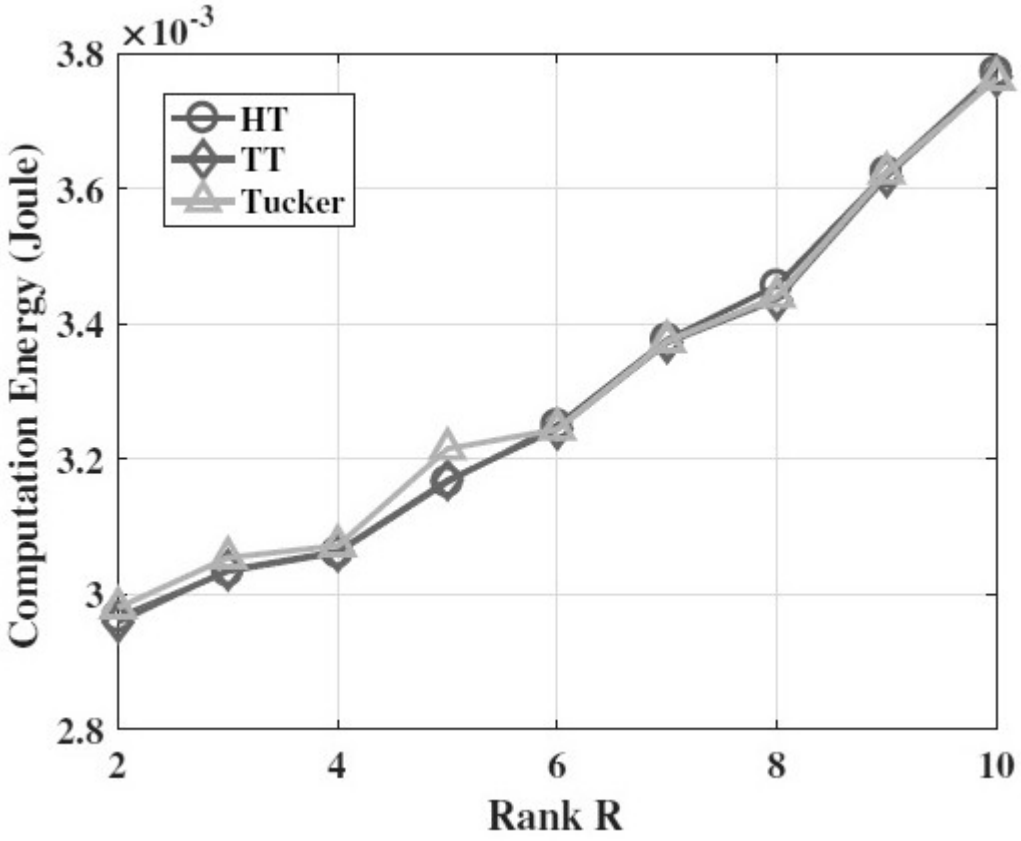

×10^-3
3.8
3.6
3.4
3.2
3
2.8
Computation Energy (Joule)
HT
TT
Tucker
2
4
6
8
10
Rank R

图4

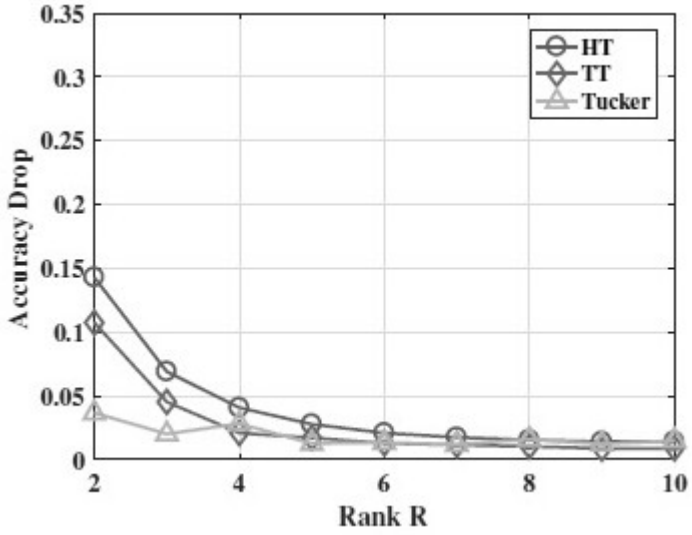

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Accuracy Drop
HT
TT
Tucker
2
4
6
8
10
Rank R

图5