DOI:10.13718/j.cnki.xsxb.2022.06.002

基于自适应指数蝙蝠和 SAE 的并行大数据分类①

钱真坤[1], 周思吉[2]

1. 四川文理学院 后勤服务处,四川 达州 635000;
2. 四川文理学院 信息化建设与服务中心,四川 达州 635000

摘要:为解决深度学习进行大数据分类时效率低的问题,本文提出一种基于自适应指数蝙蝠和堆叠自编码器(SAE)的并行大数据分类方法.在并行计算框架中,Map 阶段使用自适应指数蝙蝠算法进行特征选择,自适应指数加权移动平均值蝙蝠算法(AEB)由指数加权移动平均值(*EWMA*)和自适应权重策略得到.将选择的特征作为 Reduce 输入进行大数据分类,Reduce 阶段使用 AEB 算法训练的深度堆叠自动编码器(SAE)进行分类,进一步提高了分类精度.实验结果表明,针对不同的训练数据百分比,本文所提方法在准确度和真正例率(*TPR*)性能方面优于其他现有方法.

关 键 词:大数据;MapReduce;自适应指数蝙蝠算法;深度堆叠自动编码器

中图分类号:TP393 **文献标志码**:A **文章编号**:1000-5471(2022)06-0008-07

Parallel Big Data Classification Method Based on Adaptive Exponential Bat and Stacked Autoencoder

QIAN Zhenkun[1], ZHOU Siji[2]

1. *Logistics Service, Sichuan University of Arts and Science, Dazhou Sichuan 635000, China*;
2. *Informatization Construction and Service Center, Sichuan University of Arts and Science, Dazhou Sichuan 635000, China*

Abstract: A parallel big data classification method based on adaptive exponential bats and SAE has been proposed to solve the problem of low efficiency when classifying big data by deep learning. In the parallel computing framework, AEB algorithm is used to select features in the Map stage. AEB is obtained according to the exponential weighted moving average (*EWMA*) and adaptive weight strategy. Then the selected features are used as the input of Reduce for big data classification. In the Reduce stage, the deep stacked autoencoders trained by AEB algorithm is used for classification, which further improves the classification accuracy. The experimental results show that the proposed method is superior to other existing methods in terms of accuracy and *TPR* performance for different percentage of training data.

Key words: big data; MapReduce; adaptive exponential bat algorithm; deep stacked autoencoders

在信息技术高速发展的社会,数据正以前所未有的速度增长[1-2],大数据作为一种新的战略资源正在推动创新,改变不同领域的研究以及人们的生活、思维方式[3-4].分布式计算是一种大数据策略,常用的大

① 收稿日期:2022-03-03
基金项目:四川省高校后勤协会 2022—2023 年度立项课题(20220602).
作者简介:钱真坤,硕士,实验师,主要从事计算机应用及软件工程研究.

数据框架之一是 Hadoop，在 Hadoop 分布式文件系统(Hadoop Distributed File System，HDFS)上实现 MapReduce 并行计算[5-6]. Apache Spark 是另外一个常用的并行计算框架，Spark 基于 MapReduce 算法实现分布式计算，与 Hadoop 框架的不同之处是：Job 中间输出和结果可以保存在内存中，因此不需要读写 HDFS，Spark 的核心依然是 MapReduce. Spark 对于数据挖掘与机器学习等需要迭代的算法更友好，适应性更强[7-8]. MapReduce 由两个阶段组成：Map 和 Reduce，Map 阶段处理输入的数据拆分，生成不同的键值对，Reduce 阶段按键汇总在映射阶段获得的结果[9].

大数据分类研究已经应用到各个行业，如金融、医疗、工业等. 文献[10]针对大数据分类中的噪声问题，提出两种消除噪声样本的大数据预处理方法：同质集合和异类集合过滤器，通过对大数据中噪声的处理得到高质量和干净的数据. 文献[11]提出一种 Spark 框架下 K 最邻近(KNN)分类器的网络大数据分类处理方法，该方法通过 Map 阶段分区 K 近邻操作，并通过 Reduce 阶段确定最终 K 近邻，同时对近邻的标签集合进行聚合，得出分类结果，但是该方法分类准确度较低. 文献[12]提出了物联网大数据的随机森林分类方法，并根据蜻蜓优化选取特征对电子医疗数据进行分类，但该方法仅考虑了目标和当前特征变量的数据. 文献[13]设计了一种线性支持向量机大数据分类方法，相较于传统支持向量机，该方法在训练速度和分类精度上具有明显的优势，但用于更大数据集时会影响性能. 文献[14]提出了蚁群优化-人工神经网络联合算法，该算法使用了深度人工神经网络，并进行了蚁群优化，提升了分类准确度. 文献[15]使用蝙蝠算法优化人工神经网络，提高了分类准确率.

本文在研究了大数据处理框架和现有大数据分类方法的基础上，提出了基于自适应指数蝙蝠和堆叠自编码器(Stacked AutoEncoder，SAE)的并行大数据分类方法，该方法根据大数据分类方法的 MapReduce 并行实现，设计了基于自适应指数蝙蝠算法. 在 Map 阶段进行特征选择，在 Reduce 阶段，使用 AEB 训练的深度堆叠自动编码器分类，得到分类结果. 实验结果表明，该方法能够实现较高精度的大数据分类.

1　分布式处理框架

MapReduce 是一个编程范例，用于在分布式环境中处理大数据集，MapReduce 框架为大数据提供了一个可扩展的容错环境. 在 MapReduce 范例中，Map 阶段执行过滤和排序，Reduce 阶段执行分组和聚合操作.

图 1 给出了 MapReduce 框架流程. 一个节点被选为负责分配工作的 Master，其余的都是 Worker，输入数据分为多个 Split，Master 设备分配给 Map Worker. 每个 Worker 处理相应的输入 Split，生成＜键/值＞对并将其写入中间文件(在磁盘或内存中). Master 通知 Reduce 阶段 Worker 关于中间文件的位置和读取数据，根据 Reduce 阶段处理它. 最后，将数据写入输出文件.

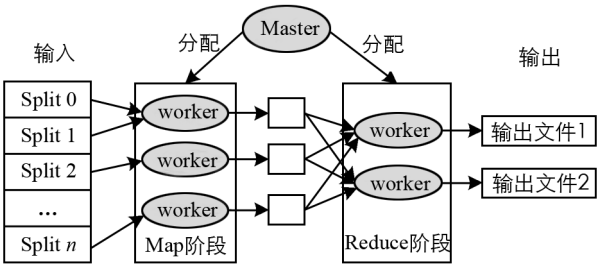

图 1　MapReduce 框架

MapReduce 范例的主要贡献是可扩展性，因为 MapReduce 允许在大量节点上进行高度并行化和分布式执行任务. 在 MapReduce 范例中，Map 或 Reduce 任务分成多个作业，这些作业被分配给网络中的节点，通过将任何失败节点的作业重新分配到另一个节点来实现可靠性. 开源 MapReduce 在 Hadoop 分布式文件系统(HDFS)之上实现.

Hadoop MapReduce 的主要优点之一是允许非专家用户轻松地对大数据运行分析任务. Hadoop MapReduce 使用户可以完全控制输入数据集的处理方式，用户使用 Java 编写查询代码，便于多数没有数据库背景，只需具备 Java 基础知识的开发人员使用.

2　基于自适应指数蝙蝠和 SAE 的并行大数据分类

本文基于自适应指数蝙蝠和 SAE 的并行大数据分类方法，Map 阶段使用 AEB 算法从数据库中选择特征. 然后，将选定的特征提供给 Reduce 进行大数据分类，Reduce 阶段使用深度堆叠自动编码器进行大数据分类，该编码器由 AEB 算法选择特征训练而成. 图 2 给出了基于 AEB 堆叠自动编码器算法进行大数据分

类的框图.

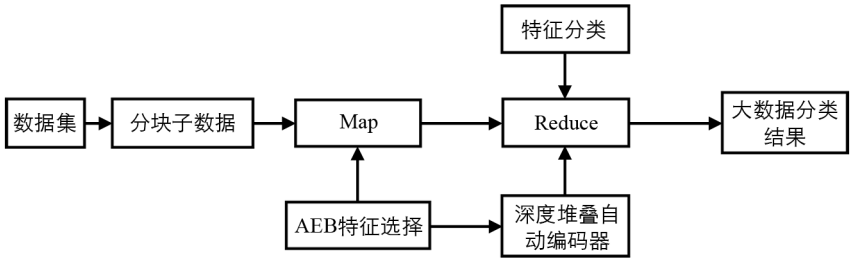

图2 自适应深度堆叠自动编码器大数据分类

将输入的大数据集合视为 H，并将多个属性表示为 $H=x_{nm}$，$1\leqslant m\leqslant G$，$1\leqslant n\leqslant q$. 其中，x_{nm} 表示第 m 个数据的第 n 个属性的大数据，所有数据均由 G 个数据点和 q 个属性组成.

2.1 Map阶段

特征提取是Map阶段的重要功能，其中选择相关特征通过减小维度来最大程度地提高分类精度. Map阶段使用AEB算法选择特征.

AEB算法是将指数加权移动平均值(Exponential Weighted Moving Average，*EWMA*)和自适应权重策略引入到蝙蝠算法中得到的. 蝙蝠算法作为一种著名的启发式算法，模拟了蝙蝠回声定位的行为，具有模型简单，收敛速度快，分布均匀等特点. 对于第 a 个蝙蝠在第 $h+1$ 次迭代中的蝙蝠速度 v_a^{h+1} 和位置 A_a^{h+1}，表示为：

$$\begin{cases} v_a^{h+1}=v_a^h+(A_a^h-A_{bat}^*)\times f_a \\ A_a^{h+1}=A_a^h+v_a^{h+1} \end{cases} \tag{1}$$

其中，A_a^{h+1} 表示第 a 个蝙蝠在第 $h+1$ 次迭代中的位置，A_a^h 表示第 a 个蝙蝠在第 h 次迭代中的位置，A_{bat}^* 表示蝙蝠的最佳位置，v_a^h 表示蝙蝠在第 h 次迭代中的速度，f_a 表示第 a 个蝙蝠的频率，$f_a=f_{\min}+(f_{\max}-f_{\min})\times\omega$，$\omega\in[0,1]$ 是遵循正态分布的随机数. 蝙蝠算法的位置更新公式为：

$$A_a^h=A_{bat}^*+\varepsilon L_{mean}^h \tag{2}$$

$\varepsilon\in[-1,1]$ 是一个随机数，L_{mean}^h 表示第 h 次迭代中所有响度的平均值. 随着蝙蝠接近其目标，响度 L 和发射速率 r，如以下公式进行更新.

$$\begin{cases} L_a^{h+1}=\psi L_a^h \\ r_a^{h+1}=r_a^0/[1-\exp(-\gamma h)] \end{cases} \tag{3}$$

其中，r_a^0 为初始发射速率，γ 和 ψ 是一个常数，γ 表示脉冲频数增加系数，$\gamma>0$，ψ 表示脉冲响度减弱系数，$0<\psi<1$. 将式(1)中两个式子融合得到蝙蝠算法表达式为：

$$A_a^{h+1}=A_a^h(1+f_a)+v_a^h-A_{bat}^*\times f_a \tag{4}$$

指数加权移动平均值(*EWMA*)是平均数据的过程，以获得较少的权重随时间而被删除的数据，可表示为：

$$A_a^h=\frac{1}{\beta}[A_{EWMA}^h-(1-\beta)\times A_{EWMA}^{h-1}] \tag{5}$$

A_{EWMA}^h 和 A_{EWMA}^{h-1} 表示当前和上一次迭代的记录，β 为常数，取值范围为0到1. 当前的预测取决于历史值乘权重，变化的采样间隔会迅速生成有关过程的信息，可以防止连续产生不良结果. 指数蝙蝠算法是将 *EWMA* 融合进蝙蝠算法，可表示为：

$$A_a^{h+1}=\frac{1}{\beta}[A_{EWMA}^h-(1-\beta)\times A_{EWMA}^{h-1}]\times(1+f_a)+v_a^h-A_{bat}^*\times f_a \tag{6}$$

AEB算法是通过引入自适应速度的概念而得到的，蝙蝠个体继承上一代的速度，并获得最佳个体飞行，当算法进行到后期时，大多数蝙蝠个体都会陷入局部优化，而蝙蝠的速度无法使其逃脱局部优化，为了解决这个问题，本文提出了自适应蝙蝠速度算法. 该算法基于每个蝙蝠的适应度(距离越近适应的最佳距离值越高，距离越远值越低)，给其速度加上权重，如果个体比其他个体更接近最优解，则其适应价值更高，此时迅速放慢个体速度，使个体收敛到最优解，在这种情况下给出较小的权重值. 相反，如果个体的适

应价值非常低，则意味着个体距离最佳解决方案还很远，需要给它更大的权重值. 权重计算方法设计为：

$$w_a^h = \frac{e_a^{score\,h}}{e_a^{score1} + e_a^{score2} + \cdots + e_a^{scoren}} \tag{7}$$

$e_a^{score\,h}$ 定义为：

$$e^{score\,h} = \begin{cases} 1, \ if(f_{worst} = f_{best}) \\ (f_{worst} - f_i)/(f_{worst} - f_{best}), \ otherwise \end{cases} \tag{8}$$

大数据被分为数据子集，表示为 $x_{nm} = \{q_e\}$，$1 \leqslant e \leqslant F$，数据子集的总数等于 Map 阶段中的映射器数量，表示为 $D = \{D1, \cdots, DF\}$，其中 F 是从大数据开发的总子集数. Map 的输入由 E 个总属性组成，表示为：

$$q_e = \{S_{j,q}\}, \ 1 \leqslant j \leqslant M, \ 1 \leqslant q \leqslant E \tag{9}$$

其中，$M < G$ 表示第 e 个子集的总数据点. 本文所提出的 AEB 算法在 Map 阶段用于选择最佳特征. 将数据输入到 Reduce 阶段，使用 AEB 算法处理生成最优特征. 为了提供更好的性能，选择具有最大值的适应度输出作为最佳解. AEB 特征选择被描述为：

$$u = \{u_1, u_2, \cdots, u_b, \cdots, u_l\} \tag{10}$$

其中，选择特征的总数表示为 l. 解向量是特征尺寸为 $[1 \times l]$ 的向量.

在 MapReduce 处理框架中，首先对大数据集进行不同子集的划分，每个子集对应不同的 Split，然后在 Map 进行特征选择，输出键值对为 < 分类准确率，特征 >，对所有分类准确率进行排序，选择最大的分类准确率，其对应的特征即为最终特征，所有特征以集合形式输入 Reduce 阶段.

在 Map 阶段将获得的 AEB 算法用于从数据库中选择特征. 并且在 Reduce 阶段使用的深度堆叠自动编码器由 AEB 训练而成.

2.2　Reduce 阶段

使用最常用的深度学习方法深度堆叠自动编码器对 Map 中选定的特征进行分类，深度堆叠自动编码器是一个对称的神经网络，用于无人监督方式学习数据集的特征，分为 3 层：输入层、隐藏层和输出层. 输入层完全连接到隐藏层，隐藏层进一步连接到输出层，如图 3 所示.

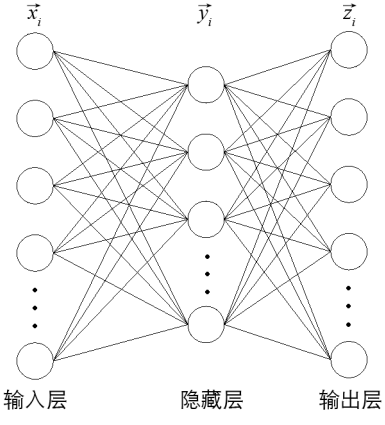

图 3　深度堆叠自动编码器网络体系结构

给定一组训练样本 $\{\vec{x}_1, \vec{x}_2, \vec{x}_3, \cdots\}$，其中 $\vec{x}_i \in R^n$，n 表示输入层的大小. 堆叠自动编码器首先通过应用线性映射和非线性激活函数 $f(x)$ 将输入向量 $\vec{x}_i$ 编码为隐藏表示 $\vec{y}_i$，其中 $\vec{y}_i \in R^m$，m 表示隐藏层的大小. 然后，通过应用线性映射和非线性激活函数 $g(x)$ 将 $\vec{y}_i$ 解码重建为 $\vec{z}_i$，这两个步骤可由式(11) 和式(12) 表示为：

$$\vec{y}_i = f(\boldsymbol{W}_1 \vec{x}_i + \boldsymbol{b}_1) \tag{11}$$

$$\vec{z}_i = g(\boldsymbol{W}_2 \vec{y}_i + \boldsymbol{b}_2) \tag{12}$$

其中，$\boldsymbol{W}_1$ 是编码权向量，$\boldsymbol{b}_1$ 是编码偏置向量，$\boldsymbol{W}_2$ 是解码权向量，$\boldsymbol{b}_2$ 是解码偏置向量. $f(\cdot)$ 和 $g(\cdot)$ 表示非线性的输入和输出映射函数，采用 Sigmoid 函数可表示为 $f(x) = 1/[1 + \exp(-x)]$. 深度自动编码器为了使隐藏层稀疏，使用稀疏约束最小化重构误差，这种架构被称为稀疏自编码器.

$$L(X, Z) + \alpha \sum_{j=1}^{m} KL(\rho \parallel \hat{\rho}_j) \tag{13}$$

其中，α 表示惩罚系数，ρ 表示稀疏参数(通常是一个非常小的值，接近零). $\hat{\rho}_j = \sum_{i=1}^{N} \vec{y}_i[j]/N$ 表示第 N 个训练样本上第 j 个隐藏单元的平均激活度. $KL(\rho \parallel \hat{\rho}_j)$ 表示 Kullback-Leibler (KL) 散度，KL 是用来测量两个不同分布之间差异的标准函数，可表示为：

$$KL(\rho \parallel \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \tag{14}$$

其中，ρ 被设置为接近 0 的值. $KL(\rho \| \hat{\rho}_j)$ 随着 ρ 和 $\hat{\rho}_j$ 之间的差减小而单调减小，如果 $\rho = \hat{\rho}_j$，则 $KL(\rho \| \hat{\rho}_j) = 0$.

多层自动编码器是多个自动编码器的串联，是将连续层输入与堆叠在该层上的自动编码器输出连接的过程. 对于第 p 层，激活输出为：

$$g^G_{p,m} = (g^G_{p-1,m} M_{p-1} + d^G_{p-1}) \tag{15}$$

式(15)中，$g^G_{p-1,m}$ 为 $p-1$ 层的输出，M_{p-1} 表示 $p-1$ 层的权重，d^G_{p-1} 表示 $p-1$ 层的偏置. 式(16)中，K 表示神经网络层数. $C_{M,A}(\cdot)$ 表示多层稀疏自动编码器函数，$g_{1,m} = q_m$. 考虑到 $C_{M,A}(q^G_m) = g^G_{ap,m}$，成本函数可以表示为：

$$Jac(M, A) = \frac{1}{2K} \sum_{m=1}^{K} \| C_{M,A}(q^G_m) - q_m \|^2 \tag{16}$$

深层自动编码器包含 3 个过程. ① 初始化成本函数局部最小值附近的各个自动编码器参数. ② 学习到下一个自动编码器隐藏层的输入. ③ 执行反向传播进行微调，通过同时更改所有模型参数，可以改善整体分类性能.

3 实验分析

将本文所提 AEB 堆叠自动编码器的大数据分类方法进行实验，并通过现有方法对本文方法进行比较分析，以证明本文方法的有效性. 本文所有实验在 Windows 10 操作系统的电脑中运行. MapReduce 实验环境由 6 台集群结点组成，选取 1 台作为主节点，另外 5 台作为从节点，每个节点的配置如下：Intel Core i5-8GB，RAM 为 64 G，硬盘容量为 1 T，软件配置为 CentOS 6.5，Hadoop 版本为 5.4.5，6 个节点都连接在 100 Mb 的交换机上.

本文采用准确度($Accuracy$)和真正例率(TPR)这两个指标评估模型的性能，数学定义为：

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

$$TPR = \frac{TP}{TP + FN} \tag{19}$$

当一个样本为正类，且被预测为正类时，称为真正类(TP)；假若负类被预测为正类，称为假正类(FP)；假若是负类被预测成负类，称为真负类(TN)；假若正类被预测为负类，称为假负类(FN).

本文使用的两个标准数据集分别是加州大学欧文分校(UCI)机器存储库中 Cleveland 数据集和 Pima India 数据集. 对比算法有：文献[11]中的 KNN 大数据分类方法，文献[13]中 SVM 分类方法，文献[14]中蚁群优化-人工神经网络联合算法以及文献[15]中蝙蝠算法 + 人工神经网络. 对比实验通过改变组合数据集中训练数据的百分比来对不同方法进行分析. 经过多次实验，本文对于自适应蝙蝠算法的参数设置为：种群 200，迭代次数 70，初始发射速率 0.95，响度衰减系数 0.95，频度增加系数 0.5.

图 4 和图 5 给出了 Cleveland 数据集中不同训练数据在百分比条件下准确度和 TPR 的对比结果.

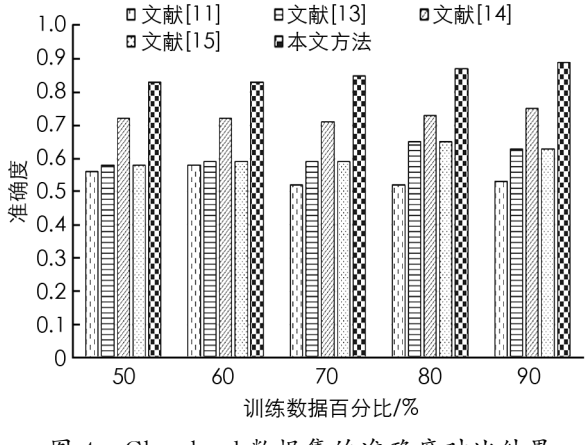

图 4 Cleveland 数据集的准确度对比结果

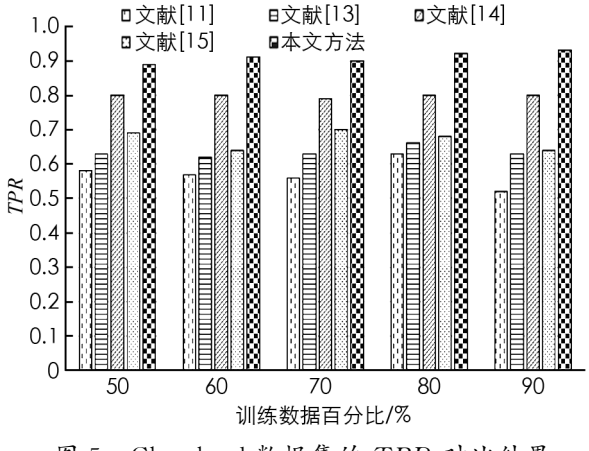

图 5 Cleveland 数据集的 TPR 对比结果

从图 4 和图 5 中的对比数据可以看出，在 Cleveland 数据集上针对不同训练数据百分比，本文所提算法的准确度都优于其他 4 种方法. 在训练数据百分比为 90%时，本文所提算法具有最高的准确度，达到 0.887 5，这是因为本文所提算法设计了自适应指数蝙蝠算法来选择最优特征，增加了分类准确度，同时深度堆叠自动编码器通过 AEB 训练得到的分类结果更加准确. 文献[14]中蚁群优化-人工神经网络联合算法准确度次之，这是因为使用了深度人工神经网络，并进行了蚁群优化，提升了分类准确度. 文献[15]中蝙蝠算法＋人工神经网络算法使用了蝙蝠算法优化人工神经网络，但是由于蝙蝠算法的局限性，导致其分类准确度与 SVM 分类性能接近. 在 Cleveland 数据集上，针对不同训练数据百分比，本文所提算法的 *TPR* 都优于其他 4 种方法，在训练数据百分比为 90%时，本文所提算法具有最高的真正例率，达到 0.928 9. 这是因为本文 AEB＋堆叠自动编码器对分类性能的提升.

图 6 和图 7 给出了 Pima India 数据集上不同训练数据在百分比条件下准确度和 *TPR* 的对比结果.

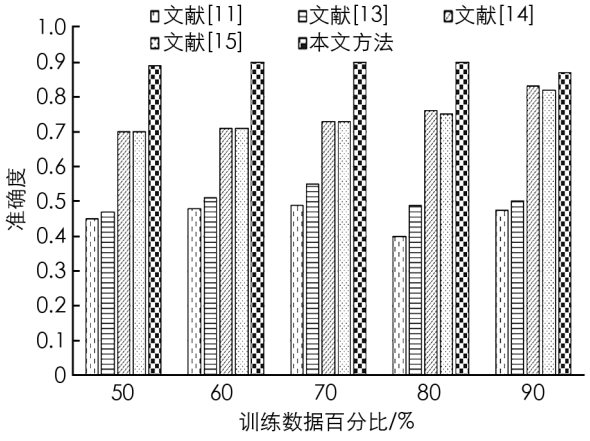

图 6　Pima India 数据集的准确度对比结果

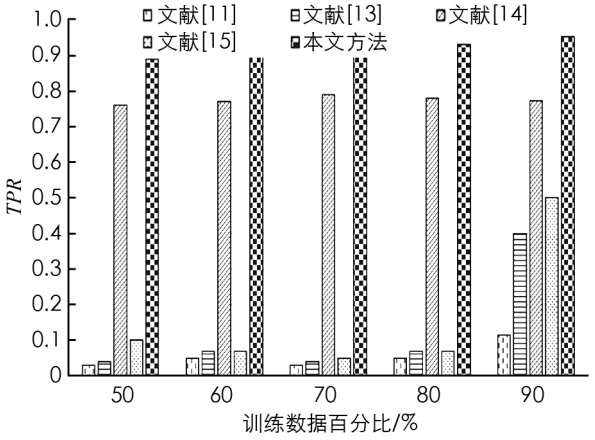

图 7　Pima India 数据集的 *TPR* 对比结果

从图 6 和图 7 中数据可以看出，当训练数据百分比达到 80%时，Pima India 数据集上本文所提算法的准确度最高；当训练数据百分比达到 90%时，*TPR* 最高. 而且，在所有不同训练数据百分比条件下，本文所提算法的性能都比其他算法高，说明了本文方法的有效性.

为了验证本文方法在大数据集上的优越性能，在 Higgs 大数据集上给出性能对比结果. Higgs 大数据集样本数量为 11 000 000，特征向量 28，标签数量 42.

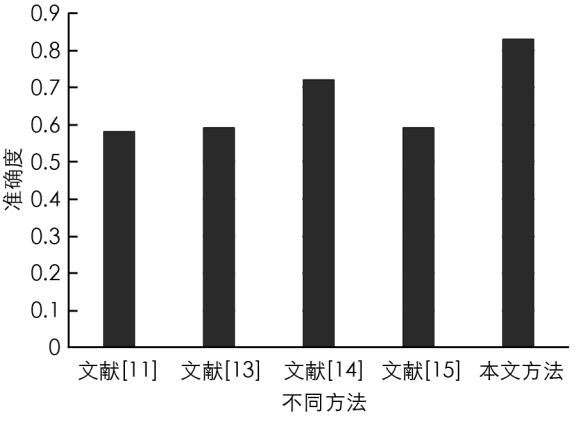

图 8　Higgs 大数据集的准确度对比结果

从图 8 中可以看出，本文方法在 Higgs 大数据集上的准确度能够达到 0.83，比文献[14]算法准确度提高了 0.11. 针对不同的大数据集，本文所提方法能够以高精度实现大数据分类，且分类性能优于其他方法，说明了本文方法的普适性和优越性.

4　结论

针对大数据分类性能低的问题，本文提出一种自适应指数蝙蝠和 SAE 的并行大数据分类方法，该方法在 Map 阶段使用设计的自适应指数蝙蝠算法进行特征选择；在 Reduce 阶段使用经过 AEB 算法训练的深度堆叠自动编码器进行分类，进一步提升了分类性能. 使用不同的实验数据集对本文所提方法进行实验，不同百分比条件下的分类性能结果显示，本文所提方法能够以高精度实现大数据分类，且在准确度和 *TPR* 性能方面都优于现有其他方法，说明本文方法的有效性和优越性. 未来的工作将通过扩展本文所提出的方法来处理安全约束.

参考文献:

[1] 孙倩,陈昊,李超.基于改进人工蜂群算法与MapReduce的大数据聚类算法[J].计算机应用研究,2020,37(6):1707-1710,1764.

[2] VARATHARAJAN R, MANOGARAN G, PRIYAN M K. A Big Data Classification Approach Using LDA with an Enhanced SVM Method for ECG Signals in Cloud Computing [J]. Multimedia Tools and Applications, 2018, 77(8): 10195-10215.

[3] XIE A, YIN F, XU Y, et al. Distributed Gaussian Processes Hyperparameter Optimization for Big Data Using Proximal ADMM [J]. IEEE Signal Processing Letters, 2019, 26(8): 1197-1201.

[4] QURESHI N M F, SIDDIQUI I F, UNAR M A, et al. An Aggregate MapReduce Data Block Placement Strategy for Wireless IoT Edge Nodes in Smart Grid [J]. Wireless Personal Communications, 2019, 106(4): 2225-2236.

[5] THIND J S, SIMON R. Implementation of Big Data in Cloud Computing with Optimized Apache Hadoop [C]//2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA). Coimbatore: IEEE, 2019.

[6] SAXENA A, CHAURASIA A, KAUSHIK N, et al. Handling Big Data Using MapReduce over Hybrid Cloud [C]. Ostrave: International Conference on Innovative Computing and Communications, 2019.

[7] LUNGA D, GERRAND J, YANG L X, et al. Apache Spark Accelerated Deep Learning Inference for Large Scale Satellite Image Analytics [J]. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2020, 13: 271-283.

[8] SHI L Z, MENG X D, TSENG E, et al. SpaRC: Scalable Sequence Clustering Using Apache Spark [J]. Bioinformatics, 2018, 35(5): 760-768.

[9] VENKATESH G, ARUNESH K. Map Reduce for Big Data Processing Based on Traffic Aware Partition and Aggregation [J]. Cluster Computing, 2019, 22(5): 12909-12915.

[10] GARCÍA-GIL D, LUENGO J, GARCÍA S, et al. Enabling Smart Data: Noise Filtering in Big Data Classification [J]. Information Sciences, 2019, 479: 135-152.

[11] 张龙翔,曹云鹏,王海峰.面向大数据复杂应用的GPU协同计算模型[J].计算机应用研究,2020,37(7):2049-2053.

[12] LAKSHMANAPRABU S K, SHANKAR K, ILAYARAJA M, et al. Random Forest for Big Data Classification in the Internet of Things Using Optimal Features [J]. International Journal of Machine Learning and Cybernetics, 2019, 10(10): 2609-2618.

[13] ZOU H S, JIN Z Y. Comparative Study of Big Data Classification Algorithm Based on SVM [C]//2018 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC). Xuzhou: IEEE, 2018.

[14] JOSEPH MANOJ R, ANTO PRAVEENA M D, VIJAYAKUMAR K. An ACO - ANN Based Feature Selection Algorithm for Big Data [J]. Cluster Computing, 2019, 22(2): 3953-3960.

[15] ISCAN H, KAMAL L L, KODAZ H. A New Hybrid Classifier based on Bat Algorithm and Artificial Neural Networks [J]. Journal of Industrial Engineering Research, 2018, 4(4): 27-33.

责任编辑 夏娟