

一种 Spark 下分布式 DBN 并行加速策略

黄 震¹, 钱育蓉¹, 于 炯^{1,2}, 英昌甜^{1,3}, 赵京霞¹

(1 新疆大学 软件学院, 新疆 乌鲁木齐 830008;

2 新疆大学 信息科学与工程学院, 新疆 乌鲁木齐 830046;

3 新疆大学 电气工程学科博士后科研流动站, 新疆 乌鲁木齐 830000)

摘 要: Spark 下分布式深度信念网络(Distributed Deep Belief Network, DDBN)存在数据倾斜、缺乏细粒度数据置换、无法自动缓存重用度高的数据等问题, 导致了 DDBN 计算复杂高、运行时效性低的缺陷. 为了提高 DDBN 的时效性, 提出一种 Spark 下 DDBN 数据并行加速策略, 其中包含基于标签集的范围分区(Label Set based on Range Partition, LSRP)算法和基于权重的缓存替换(Cache Replacement based on Weight, CRW)算法. 通过 LSRP 算法解决数据倾斜问题, 采用 CRW 算法解决 RDD(Resilient Distributed Datasets)重复利用以及缓存数据过多造成内存空间不足问题. 结果表明: 与传统 DBN 相比, DDBN 训练速度提高约 2.3 倍, 通过 LSRP 和 CRW 大幅提高了 DDBN 分布式并行度.

关键词: 分布内存计算框架; 缓存替换; 范围分区; 深度信念网络; 数据倾斜

中图分类号: TP316.4

文献标识码: A

文章编号: 1000-7180(2018)11-0100-06

A Parallel Acceleration Strategy for Distributed DBN in Spark

HUANG Zhen¹, QIAN Yu-rong¹, YU Jiong^{1,2}, Ying Chang-tian^{1,3}, Zhao Jing-xia¹

(1 School of Software, Xinjiang University, Urumqi 830008, China;

2 School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China;

3 Postdoctoral Research Station of Electrical Engineering, Xinjiang University, Urumqi 830000, China)

Abstract: DDBN(Distributed Deep Belief Network, DDBN) has many problems in Spark, such as data skew, lack of fine-grained data replacement, and unable to cache data with high re-usability automatically, resulting in high complexity and low timeliness of DDBN computing. In order to improve the timeliness of DDBN, a parallel acceleration strategy is proposed for DDBN in Spark, which includes LSRP(Label Set based on Range Partition, LSRP) algorithm and CRWS(Cache Replacement based on Weight Statistics, CRWS) algorithm. The problem of data skew is solved by LSRP algorithm, and CRW algorithm is used to solve the problem of RDD reuse and cached data caused by insufficient memory space. The results show that compared with the traditional DBN, the training speed of DDBN is increased by about 2.3 times, and the distributed parallelism of DDBN is greatly improved through LSRP and CRW.

Key words: distributed memory computing framework; cache replacement; range partition; deep belief network; data skew

1 引言

随着大数据时代到来,对大数据的精准分析需

求凸显,传统机器学习算法^[1~4]不能满足对数据的精确分析.近年来深度学习在各领域取得很大进展,解决了很多人工智能领域很多方面的难题,但深度

收稿日期: 2018-02-13; 修回日期: 2018-05-09

基金项目: 国家自然科学基金资助项目(61562086,61462079),新疆自治区研究生科研创新项目(XJGRI2016029),新疆维吾尔自治区教育厅项目(XJEDU2016S035),新疆大学博士科研启动基金项目(BS150257).

学习擅长发现密集复杂的高维数据,导致深度学习算法执行时间较长缺陷,无法实时处理数据。

现阶段一般在 Hadoop^[5~7]下提高算法的运行速度,但 Hadoop 存在频繁访问 I/O 的问题,导致了算法执行速度慢的缺陷。由于 Spark 是分布式内存计算平台,避免了频繁访问 I/O 的问题,如文献[8~10]基于 Spark 下实现传统算法的分布式并行化,提高算法的执行速度。在 Spark 下,对深度学习算法的分布式加速研究,如 2015 年 Yahoo 让 Caffe 在分布式环境 Spark 运行,文献[11]表明 TensorFlow 在分布系统中,把任务调度组织分解成图结构等,来提升分布式算法分布式并行化,现阶段研究缺乏对深度学习算法的细粒度加速。

本研究针对深度信念网络(Deep Belief Network, DBN)运行时效性低的问题;在 Spark 下,实现分布式深度信念网络(Distributed DBN, DDBN)算法,在 DDBN 训练过程中,存在数据倾斜、缺乏细粒度数据置换、无法自动缓存重用度高的数据等问题,导致了 DDBN 算法执行速度慢的缺陷。本文提出了一种基于 Spark 下 DDBN 数据并行加速策略,来提高 DDBN 算法的训练速度。

2 相关概念

2.1 任务执行框架

Spark 是分布式内存计算框架,弹性式分布式数据集 RDD (Resilient Distributed Datasets) 和 DAG(Directed Acyclic Graph)是 Spark 核心基础,通过 Transformation 和 Action 对 RDD 进行操作,由 RDD 血统路线形成 RDD 的 DAG,使用 DAG-Scheduler 依据宽依赖把 DAG 图划分成多个 Stage,通过 TaskScheduler 根对 Stage 划分成 TaskSet,提交给 Worker 节点的 Executor 进行计算,数据上传到 HDFS (Hadoop Distributed File System, HDFS)中,作业执完将最终结果数据保存到 HDFS 内。

2.2 模型定义

定义 1 DDBN 的 RDD 执行时间。设每个 RDD 有 n 个分区, RDD 的各分区空间大小不一,因此记其分区集合为 $RDD = \text{新}\{p_1, p_2, \dots, p_n\}$, 其中 p_i 表示为 RDD 的第 i 个分区。因此 RDD 执行时间为所有分区计算时间的最大值,执行时间 T_{RDD} 即:

$$T_{RDD} = \max(T_{p_1}, T_{p_2}, \dots, T_{p_n}) \quad (1)$$

定理 1 分区大小一致原则。由定义 1 知, RDD 最大分区执行时间决定 RDD 执行时间,因此在分

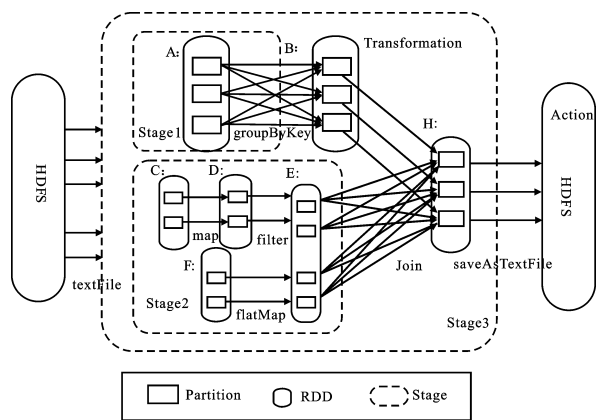


图 1 Spark 下作业的 DAG 图

区数一定的情况下,各分区大小一致, RDD 执行时间最短。

定义 2 缓存 RDD 节约重复计算时间。RDD_i 总执行次数为 r_i , RDD_i 执行 r_i 次所耗计算时间为 $r_i * T_{RDD_i}$, 那么缓存 RDD_i 所节约计算时间为 $(r_i - 1) * T_{RDD_i}$ 。

定理 2 节约时间原则。由定义 2 知,缓存 RDD 减少 RDD 重复计算的节省时间,优于读取内存中缓存的 RDD 所耗时间。

3 基于 DDBN 的并行加速策略

本节通过提出 DDBN 并行加速策略,提高 Spark 下 DDBN 算法的执行速度。首先,通过实现 DDBN 算法,提升 DBN 算法的执行速度。其次,由定理 1 提出基于标签集的范围分区算法(Label Set based on Range Partition, LSRP)解决数据倾斜问题,避免 RDD 单个分区过大,造成 DDBN 执行速度慢的问题。最后,由定理 2 提出基于权重的缓存替换算法(Cache Replacement based on Weight, CRW)解决缺乏缓存重用度高的 RDD 机制,通过缓存 RDD 减少同一 RDD 重复计算时间,并依据 RDD 的频数、血统作为 RDD 的权重,解决 Spark 下 LRU 算法粗略置换 RDD 问题。

3.1 基于 Spark 下 DDBN 的并行加速策略

基于 Spark 下 DDBN 并行加速策略,首先, DBN 算法结合 Spark 下分布式并行模式,实现分布式 DBN 算法。其次,在 DDBN 执行过程中,采用 LSRP 算法对 RDD 重新分区,避免 RDD 中单个分区过大, RDD 执行时间过长。最后,通过 CRW 算法缓存重用度高的 RDD,依据 RDD 频数和血统长度作为 RDD 的权重,当内存空间不足时,依次替换权重较小的 RDD。DDBN 并行加速策略,运行 LSRP

算法和 CRW 算法,判断 RDD 是否需要重新分区、缓存和细粒度置换问题. DDBN 加速策略执行流程图,如图 2 所示.

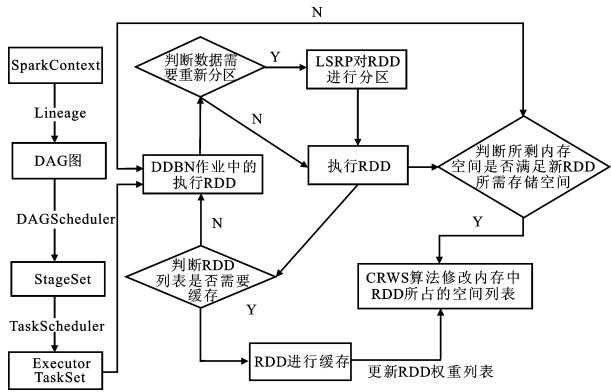


图 2 DDBN 的并行加速策略执行流程图

3.2 分布式 DBN 算法实现

DBN 是由多层受限玻耳曼兹机(Restricted Boltzmann Machine,RBM)构成、样本标签结合 softmax 分类函数进行监督训练,通过反向传播(Back Propagation,BP)来微调优 DBN 深度网络模型. DBN 将所有 RBM 按训练顺序串联形成深层信念网络,调整训练过程是根据输入数据和重构数据的损失函数,利用 BP 算法重新调整网络的参数,最终使网络达到全局最优的过程. 设 DBN 的隐为 $3(h_1, h_2, h_3)$,DBN 构建图如 3 所示.

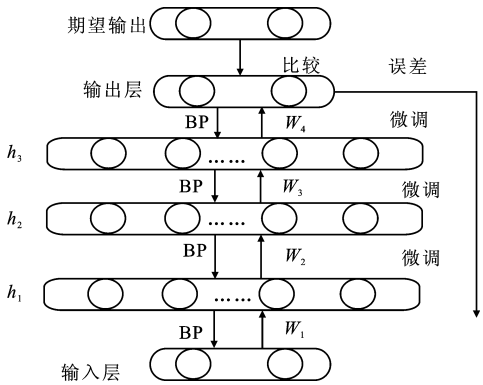


图 3 DBN 模型原理图

根据上述 DBN 算法总体架构,结合 Spark 分布式平台并行编程模式,设计了基于 Spark 下的 DBN 算法分布式并行化实现方案,算法详细实现步骤如图 4 所示. DDBN 算法训练开始之前. 首先,将数据集上传到 HDFS 内,Spark 集群任务资源调度根据 RDD 的 lineage 形成 DAG 图,使用任务调度系统中的 DAGScheduler 调度器,将整个作业分割为每个数据子集,在 Executor 进程里创建新任务,资源调

度器将计算资源分配给相应的任务. 在 Spark 上,分层并行训练,并将局部 DDBN 权重进行合并,训练完成输出全局最优模型到 HDFS.

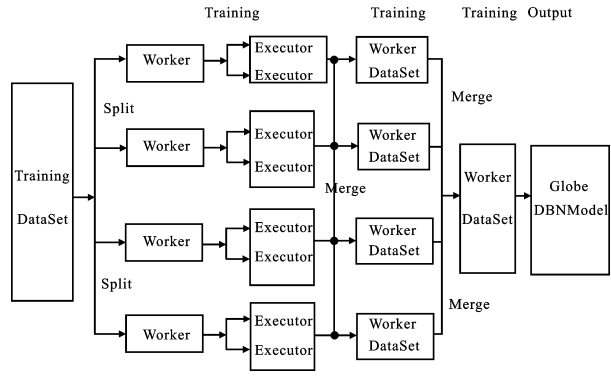


图 4 基于 Spark 下 DDBN 算法训练流程图

DDBN 算法步骤:

- 步骤 1:初始化数据路径和数据分区个数. 通过 textFile()算子,把数据从 HDFS 加载到 Spark 平台内.
- 步骤 2:DDBN 中创建 RBMtrain()方法计算权重,其中包含 backPropagation(),向后计算的结果作为下一层 RBM 输入数据, priorPropagation()向前更新权重,通过向前计算更新前向权重减小误差.
- 步骤 3:在 DDBN 中,初始化权重参数,计算第一层隐含层权重,结合步骤 2 中 DBNtrain()函数循环计算隐含层权重,然后合并各节点权重值.
- 步骤 4:将训练得到最优 DDBN 算法权重参数保存到 HDFS 内.

算法 1 DDBN 算法

Input: TrainingDataset,Filepath,partition

Output: DBNModel

```
1. val sc← new SparkConf(conf)
2. val DataRDD←sc. textFile(Filepath,partition)
3. val DBNmodel ← new DBN(). DBNtrain(DataRDD)
4. return DBNModel;
5. class DBN{
    def RBMtrain(train_t:RDD[(BDM[Double],
        BDM[Double])],opts: rray[Double],dbnconfig:
        DBNConfig,weight: DBNweigh)={
        train_t. randomSplit()//对样本数据进行切分
        backPropagation()//向后运算
        priorPropagation()//向前更新权重
    }
    //通过训练得到 DDBNModel 模型
    def DBNtrain(train_d:RDD[(BDM[Double],
        BDM[Double])],opts:Array[Double])
        :DDBNModel ={
        dbnconfig←sc. sc. broadcast(weight)
```

```
weight ← DBN.Initial()//初始化权重
weight1←DBN.RBMtrain(train_d,weight,opts ,
    dbnconfig,weight )
//训练第 2 层至 n 层
for (i ← 2 to dbnconfig.layer-1) {
    // 训练第 i 层的权重
    val weighti ←DBNweigh()
    val weight2←DBN.RBMtrain(train_d,opts,
        dbnconfig, weight )
}
}
```

3.3 基于标签集的范围分区算法

在 Spark 下,DDBN 算法训练样本集中包含不同数据类型,各种类型所占的比例不同,导致 DDBN 算法执行过程中,存在数据倾斜的问题.在 Spark 环境下,依据哈希分区算法对 RDD 分区,造成 DDBN 中 RDD 的分区大小不一,由定义 1 可知 RDD 最大分区执行时间决定 RDD 执行时间,因此可知 RDD 分区大小不一,影响了 DDBN 的执行速度.由定理 1 提出 LSRP 算法,解决数据倾斜问题,避免 DDBN 中 RDD 的单个分区数据过大,使 RDD 各个分区大小保持基本一致,减少 RDD 的执行时间,提高 DD-BN 整个算法的训练速度.LSRP 算法通过设定标签集,把每条数据依据 key 大小,分配到 RDD 各分区内.

- LSRP 算法步骤如下:
- 步骤 1:样本数据集进行采样,判断数据是否存在数据倾斜问题.
- 步骤 2:如果数据存在数据倾斜问题,重新对数据进行分区,依据 LSRP 算法对数据进行重新分区.
- 步骤 3:通过一系列分割标记分割数据段,如果数据有 n 个分区,则需要有 $n-1$ 个分割标签,分割标签集为 $(l_1, l_2, \dots, l_{n-1})$.
- 步骤 4:在数据分区时,按标签把数据加载到不同的分区中,如 $key_1 < l_1, l_2 < key_2 < l_3, \dots, l_{n-2} < key_{n-1} < l_{n-1}, l_{n-1} < key_n$,实现对 RDD 重新分区.

算法 2 LSRP 算法

```
count←0
dataList←sc.textFile(“hdfs:/data.txt”)
    .map(x=>(count++, x))
labelList←(l1, l2, l3, ..., ln-1)
4. partitionList.length←n
5. for i=0 to labelList.length-1 do
6.   for j=0 to dataList.Length do
7.     if(dataList.first._1 < labelList(i))
8.       then
9.         partitionList.add(dataList.first)
10.        dataList←dataList.next
```

```
11.   end if
12.   end for
end for
14. partitionList(n)←dataList
```

3.4 基于权重的缓存替换算法

在 DDBN 执行过程中,重用度高的 RDD 过多,Spark 缺乏对 RDD 自动缓存机制,导致对重用度高的 RDD 进行重复计算,造成了计算资源的浪费,同时也增加了整个作业的执行时间,数据缓存过多会导致内存空间不足问题,缺乏细粒度置换 RDD 问题,导致延长 DDBN 作业执行时间,由定理 2 提出 CRW 算法,缓存重复度高的 RDD,能减少整个作业执行时间.CRW 算法通过统计 RDD 的出现频数和血统长度,缓存频数大于等于 2 的 RDD,当内存空间不足时,CRW 算法依据 RDD 的频数和血统作为权重,按照权重大小依次替换,直到内存空间满足新 RDD 所需空间.

- CRW 算法步骤如下:
- 步骤 1:通过得到 RDD 数据集合,统计各个 RDD 重复度 f ,RDD 的血统长度为 l .记为 Map $(name, (f, l))$.
- 步骤 2:当 RDD 的 $f \geq 2$ 时,对 RDD 进行缓存,并统计 RDD 的 lineage 的长度 l .
- 步骤 3:首先,依据 RDD 频数从小到大顺序排序.其次,再根据 RDD 的血统长度从小到大的顺序排序,得到 RDD 权重表.

- 步骤 4:在整个 DDBN 作业中,已缓存的 RDD 每执行 1 次,Map 中相应的 RDD 的 f 减去 1,直到 RDD 频数 f 为零时,回收频数为零的 RDD.
- 步骤 5:当内存空间不足时,释放权重最小的 RDD,按照权重列表的顺序,依次替换直到满足新的 RDD 所需空间大小.

算法 3 CRW 算法

```
1. rddSet←getAllRDDs()
//统计 rdd 被调用次数与血统长度
2. for i=0 to rddSet.length do
3.   if rddMap.key.contains(rddSet(i)) then
4.     rddMap.key.contains(rddSet(i)).f++
5. rddMap.key.contains(rddSet(i)).l++
6.   else
7.     rddMap.add((rddSet(i),1))
8.   end if
9. end for
//缓存被调用次数大于等于 2 的 RDD
10. for i=0 to rddMap.length do
11.   if rddMap.first.value >=2 then
```

```
12. rdd←getRDDByKey(rddMap, first, key)
13. rdd. cache
14. end if
15. end for
//缓存的 rdd 空间每被调用 1 次减去 1
if rddName == rddMap. first, key then
17. rddMap. key. contain(rddName). value--
18. end if
//依据频数排序
19. rddMap. getValues. sort(f)
//在频数排序的基础上,依据血统长度排序
20. rddMap. getValues. sort(l)
//按照顺序依次替换 RDD 直到满足新的 RDD 所需存储空间大小
21. for i=0 to rddMap. length do
22. if newRDD. size >= free. size then
23. free. size+=getRDDByKey(rddMap, first, key) . size
24. getRDDByKey(rddMap, first, key). release
25. rddMap. fist. remove
26. end if
27. rddMap. add((newRDD,1))
28. end for
```

4 实验结果与分析

4.1 实验数据集

考虑到遥感影像数据计算量大、信息量丰富等特征,因此本实验采用遥感影像数据作为数据源,数据源自于 Landsat8 卫星. 本文研究区是新疆维吾尔自治区玛纳斯县,位于东经 85. 7°—86. 7°、北纬 43. 5°—45. 6°. 使用 ENVI5. 2 软件对遥感影像进行辐射校正、大气定标与几何校正等预处理. 本研究提取常用的归一化植被指数(NDVI)、比值植被指数(RVI)、差值植被指数(DVI)、增强型植被指数(EVI),基于这 4 中植被指数参数,提取 5 种不同的文本样本集,样本集数据中其它数据包含沙漠、河流等,如表 1 所示.

表 1 不同特征组合的样本表

训练集名称	参量	草地	其它
样本 1	NDVI+RVI+DVI	10000	10000
样本 2	NDVI+RVI+EVI	10000	10000
样本 3	NDVI+DVI+EVI	10000	10000
样本 4	RVI+DVI+EVI	10000	10000
样本 5	NDVI+RVI+DVI+EVI	10000	10000

4.2 实验及结果

本节共 3 组实验,实验 1 验证在相同数量级下,DDBN 算法训练执行速度优于单机下 DBN 训练执

行速度. 实验 2 验证采用 LSRP 算法解决数据倾斜问题,提升 DDBN 执行速度. 实验 3 验证采用 CRW 算法解决缺乏细粒度数据置换、无法自动缓存重用度高的数据问题.

实验 1:DDBN 与单机下 DBN 算法的执行速度本实验选用样本 1 作为训练集,数据量为 10 000 条和 20 000 条两个数量级下,测试 DDBN 和单机下 DBN 训练数据所耗时间,如表 2 所示.

表 2 DDBN 与 DBN 算法的执行时间表

数据量	10000(条)	20000(条)
单机下 DBN 耗时(s)	168. 68	341. 35
DDBN 耗时(s)	73. 65	148. 41
加速比约为	2. 29	2. 30

由表 2 可知,1)在不同的数量级下,DBN 算法执行时间高于 DDBN 算法执行时间. 2)在相同数据量级下,DDBN 算法执行速度约是单机下 DBN 算法的执行速度 2. 3 倍.

实验 2:采用 LSRP 算法前后 DDBN 执行时间在 DDBN 的隐含层数 3 下,输入数据采用不同的样本即特征选择不同,测试采用 LSRP 算法前后,DDBN 算法同一样本集下执行时间如图 5 所示.

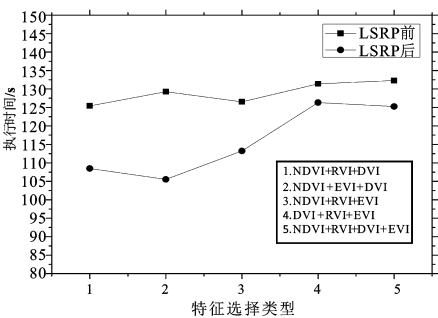


图 5 利用 LSRP 算法前后 DDBN 执行时间图

由图 5 知:(1)在不同的样本集下,DDBN 的执行速度不同;2)同一样本集下,采用 LSRP 算法后,提高了 DDBN 算法的执行速度;3)在不同样本集下,采用 LSRP 算法前后 DDBN 执行时间之差不同,提高 DDBN 执行速度的程度不同.

实验 3:采用 CRW 算法前后 DDBN 的执行时间

本实验采用样本集 5 作为 DDBN 算法的训练样本集,通过改变隐含层数,缓存更多的 RDD,测试采用 CRW 算法前后 DDBN 的执行时间如图 6 所示.

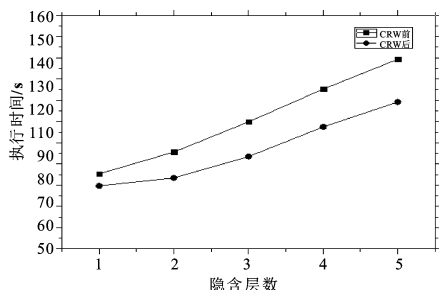


图6 采用CRW算法前后DBN执行时间图

由图6可知:(1)通过采用CRW算法,缩短了DBN算法执行时间,提高了DBN算法执行速度;(2)采用CRW算法前后DBN执行时间的差值,随着隐含层数的增加不断变大,提升DBN执行速度越来越显著。

5 结束语

本文采用基于Spark下DBN的并行加速策略,解决数据倾斜、缺乏细粒度数据置换、无法自动缓存重用度高的数据等问题,提高DBN算法的执行速度,采用LSRP算法解决DBN训练样本集倾斜的问题,使得RDD各个分区所包含数据量大小比较均匀,提高了DBN训练速度,通过CRW算法,缓存DBN算法中重用度高的RDD,减少RDD重复计算的工作,数据缓存过多导致内存空间不足时,依据RDD权重从小到大的顺序依次移出RDD,直到满足新RDD所需内存空间大小。本研究下一步工作,基于Spark Streaming下DBN算法处理流式数据的加速策略。

参考文献:

- [1] 毛毅,陈稳霖,郭宝龙,等. 基于密度估计的逻辑回归模型[J]. 自动化学报,2014,40(1):62-72.
- [2] 谭熊,余旭初,秦进春,等. 高光谱影像的多核SVM提取[J]. 信息安全与技术,2014,35(2):405-411.
- [3] 张伐伐,李卫忠,卢柳叶,等. SVM多窗口纹理土地利用信息提取技术[J]. 遥感学报,2012,16(1):67-78.

- [4] 饶萍,王建力,王勇,等. 基于多特征决策树的建设用地信息提取[J]. 农业工程学报,2014,30(12):233-240.
- [5] Alham N K, Li M, Liu Y, et al. A Map Reduce-based distributed SVM algorithm for automatic image annotation[J]. Computers & Mathematics with Applications, 2011, 62(7):2801-2811.
- [6] Hodge V J, O'Keefe S, Austin J. Hadoop neural network for parallel and distributed feature selection[J]. Neural Networks the Official Journal of the International Neural Network Society, 2016 :78 :24-35.
- [7] Veetil S, Gao Q. Chapter 18-Real-time Network Intrusion Detection Using Hadoop-Based Bayesian Classifier[J]. Emerging Trends in Ict Security, 2014:281-299.
- [8] 王诏远,王宏杰,邢焕来,等. 基于Spark的蚁群优化算法[J]. 计算机应用,2015, 35(10):2777-2780.
- [9] Arias J, Gamez J A, Puerta J M. Learning distributed discrete Bayesian Network Classifiers under Map Reduce with Apache Spark[J]. Knowledge-Based Systems, 2017, 16-26[10] Maillo J, Ramirez S, Triguero I, et al. KNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors Classifier for BigData[J]. Knowledge-Based Systems, 2016.
- [11] Seide F, Fu H, Droppo J, et al. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs[J]. Interspeech, 2014: 1058-1062.

作者简介:

- 黄震** 男,(1989-),硕士研究生.研究方向为内存计算.
- 钱育蓉(通信作者)** 女(满族),(1980-),博士,教授.研究方向为网络计算和遥感图像处理.
- E-mail:qyr@xju.edu.cn.
- 于炯** 男,(1964-),博士,教授,博士生导师.研究方向为网络安全,网络与分布式计算等.
- 英昌甜** 女,(1989-),博士.研究方向为分布式并行计算、分布式并行系统、内存计算.
- 赵京霞** 女(满族),(1995-),硕士研究生.研究方向为深度学习在遥感图像处理方面的应用.