李宏志,李苡兰,赵生慧. 基于 Spark 的大规模文本 KNN 并行分类算法[J]. 湖南科技大学学报(自然科学版),2019,35(1):90-97. doi:10.13582/j.cnki.1672-9102.2020.01.013

Li H Z, Li X L, Zhao S H. Large Scale Text KNN Parallel Classification Algorithm Based on Spark [J]. Journal of Hunan University of Science and Technology(Natural Science Edition), 2019,35(1):90-97. doi:10.13582/j.cnki.1672-9102.2020.01.013

基于 Spark 的大规模文本 KNN 并行分类算法

李宏志[1,2*],李苡兰[2],赵生慧[1]

(1.滁州学院 信息学院,安徽 滁州 239000;2.福建师范大学 光电与信息工程学院,福建 福州 350000)

摘 要:在使用 KNN 算法进行大规模文本分类,需要处理频繁的迭代运算,针对现有 Hadoop 平台迭代运算效率较低的问题,本文提出一种基于 Spark 平台的并行优化 KNN 算法.主要从 3 个方面对算法进行优化,首先,对于训练数据集通过剪枝算法控制有效数据的规模,从而减少迭代运算的次数;其次,针对高维数据集采用 ID3 算法利用信息熵进行属性降维,减少文本相似度的运算量;最后,使用 Spark 并行计算平台,引入内存计算最大限度地减少了迭代运算的 I/O 次数,提高处理速度.通过实验,与常用的 KNN 算法相比,基于 Spark 的 KNN 文本并行分类算法在加速比、扩展性等主要性能指标上表现较优,能够较好地满足大规模文本分类的需求.

关键词:KNN; 并行化;文本分类; Spark; RDD; MapReduce

中图分类号:TP311 **文献标志码**:A **文章编号**:1672-9102(2020)01-0090-08

Large Scale Text KNN Parallel Classification Algorithm Based on Spark

Li Hongzhi[1,2], Li Xianlan[2], Zhao Shenghui[1]

(1. College of Computer and Information Engineering, Chuzhou University, Chuzhou 239000, China;
2. College of Photonic and Electronic Engineering, Fujian Normal University, Fuzhou 350000, China)

Abstract: Aiming at the problem in the use of KNN algorithm for large-scale text classification, the Hadoop platform had a low efficiency in operating frequent iterative computation, a parallel optimization KNN algorithm based on the spark platform was proposed. Firstly, for training dataset, the size of effective data was controlled by branch reduction algorithm. Secondly, aiming at high-dimensional dataset, ID3 algorithm was used to reduce the dimension of attributes and reduced the computational complexity of text similarity. Finally, using the Spark platform, in-memory computing was introduced to minimize the number of I/O iterations and improved the computational speed. Compared with the traditional KNN algorithm, the KNN parallel classification algorithm based on the Spark platform had better performance on the main performance indexes, such as acceleration ratio and extensibility, and was better used on the large-scale text classification.

Keywords: KNN; parallelization; text classification; spark platform ; RDD ; MapReduce

文本分类作为信息处理的关键技术之一,在信息检索、知识工程,人工智能等领域有着广泛的应用[1-2].目前应用较广泛文本分类算法主要有朴素贝叶斯、决策树、支持向量机(Support Vector Machine,

收稿日期:2018-08-09

基金项目:安徽省自然科学基金资助面上项目(1408085MF126)

*通信作者,李宏志, E-mail: 1071260932@qq.com

SVM)、K 最近邻算法(k-NearestNeighbor,KNN)等,其中由于 KNN 算法具有稳定性强,准确率高等优点得到了广泛的应用.随着互联网的普及,其产生的文本数量呈现指数增长;如何对海量的文本信息进行有效价值挖掘与分类成为当前的研究热点之一.

以 MPI(message passing interface)、网格计算等为代表的传统并行计算方法存在开发复杂、扩展性不好等问题,已经无法满足日益增长的大规模数据处理需求[3].随着云计算技术的普及,大规模数据处理更适合在云环境中进行,Map-Reduce 编程模型因其具备良好的扩展性、可用性、容错性成了数据处理领域的热点技术之一,Hadoop 的 MapReduce 框架能够处理多至百万个节点和 ZB 级别的数据量[4-5].Hadoop 平台的 MapReduce 编程模型适用数据量较大但实时性要求不高的离线批处理作业,对于递归、迭代和嵌套等运算利用 MapReduce 主要存在如下问题:

1)每次迭代运算都作为 Hadoop 的一次独立的 Job 执行,需要重新完成数据的初始化、读入和网络传输,会增加不必要的系统开销.

2)迭代运算中存在不变的数据集在下一次运算的时候依然需要重新装载原来的数据,会浪费大量的 I/O 和 CPU 资源和网络带宽.

新型并行计算平台 Spark 允许将迭代的数据存于内存中,减少了数据加载的 I/O 和网络带宽的开销,使参与迭代运算的数据能够重复使用,适用实时迭代数据运算[6].

本文主要研究 KNN 分类算法的并行化,采用 K-means 聚类的方法减少样本的计算量,并引入了特征信息熵对 KNN 分类算法进行了改进,通过计算特征属性的信息增益,降低数据集的计算维度,减少 KNN 分类算法中相似度的冗余计算,提高了执行效率,同时对并行 KNN 分类算法在大规模文本分类中的应用做了有益的探索.

1　文本分类 KNN 算法及其改进

1.1　分类文本预处理

文本作为非结构化的信息载体,用于信息处理的各个方面,然而计算机无法直接对非结构化的数据进行处理,应对非结构化的文本先进行出了结构化预处理[7].文本预处理是进行文本分类的首要步骤,文本预处理步骤包括分词、去停词、词频计算等.非结构化的文本一般转换为结构化的空间向量模型(VSM)$\{<t_0,w_0>\cdots<t_6,w_6>\cdots<t_i,w_i>\}$,词组的权重采用 TF-IDF(term frequency-inverse document frequency)算法计算.

权重 w_i 可通过式(1)来计算:

$$w(d,t_i)=\frac{tf(d,t_i)\ln\left(\frac{N}{n}+1\right)}{\sum_{t_i\in d}\left[tf(d,t_i)\ln\left(\frac{N}{n}+1\right)\right]}. \tag{1}$$

式中:$w(d,t_i)$ 为特征项 t_i 在文档 d 中的权重;$tf(d,t_i)$ 为特征项 t_i 在文档 d 中的词频;N 为训练文本的数量;n 为特征项 t_i 在样本中出现的次数.

如图 1 所示,非结构化的文本经过预处理之后转换成结构化的便于计算机处理的向量数据.

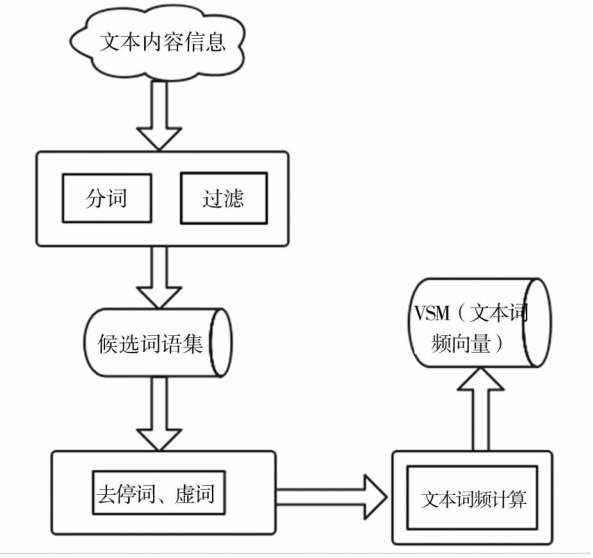

图 1　文本预处理过程

1.2　KNN 分类算法及其改进

KNN 分类算法是一种基本的分类处理算法,其基本思路:计算待测样本与已知样本的相似距离,选择距离最近的 K 份样本;K 份样本中多数所属的类别为待测样本的分类类别.

KNN 具体实现的步骤:

1) 对于训练样本集 $\boldsymbol{S}$, 其中有 N 种类别 $c_1, c_2, c_3, \cdots, c_N$. 样本数据集可表示为

$$\boldsymbol{S} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}. \tag{2}$$

式中: $\boldsymbol{x}_i \in \boldsymbol{X} \subseteq \boldsymbol{R}^n$ 为实例的文本词频向量; $y_i \in C = \{c_1, c_2, c_3 \cdots c_N\}$ 为实例类别.

对于待测数据集 $\boldsymbol{T}$ 可表示为 $\boldsymbol{T} = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \cdots, \boldsymbol{t}_N\}$, 其中 $\boldsymbol{t}_i \subseteq \boldsymbol{R}^n$ 为样本的文本词频向量.

2) 根据给定的距离的定义,计算待测样本集 $\boldsymbol{T}$ 中的向量 $\boldsymbol{t}$ 与训练样本集中的向量 $\boldsymbol{x} \in \boldsymbol{X}$ 的距离,距离一般采用余弦相似度:

$$\mathrm{sim}(\boldsymbol{t}, \boldsymbol{x}) = \frac{\boldsymbol{t} \cdot \boldsymbol{x}}{|\boldsymbol{t}| \times |\boldsymbol{x}|}. \tag{3}$$

3) 选取和待测样本距离最近的 K 份样本,其中所属最多的分类为待测样本的所属类别.

传统的 KNN 算法虽然简单有效,但计算过程中需要与所有的样本进行距离的计算,算法的复杂度与样本的数量大小成正比,而向量距离运算的复杂度又与向量的维度成正比;因此在大数据的环境下,KNN 算法的时间效率问题尤为明显[8].

K-means 聚类算法作为基于统计学原理的聚类分析算法,算法简单易实现,收敛性较好,但传统 K-means 算法在初始化时对中心点的选取较为敏感.因此本文采用文献[9]提出的 K-means 算法对训练数据集进行裁剪.通过 K-means 算法筛选出分类样本数据集中的"质心"数据集,"质心"数据集作为新的训练样本,降低算法的复杂度.训练数据集的裁剪算法如下.

算法 1　基于优化 K-means 聚类的训练数据裁剪算法 .

名称　DataSetCut

输入　已分类的训练数据集 $\boldsymbol{S}$

输出　裁剪之后的训练数据集 $\boldsymbol{S}'$

选取已分类的 n 个样本簇 $\boldsymbol{S}\{s_1, s_2, s_3 \cdots s_n\}$, $\boldsymbol{S}\{s_1 = (\boldsymbol{x}_1, y_1), s_2 = (\boldsymbol{x}_2, y_2), \cdots, s_n = (\boldsymbol{x}_n, y_n)\}$ 对于每个样本簇执行:

Step1 K-means 初始化中心点选择与优化

a)计算样本簇中向量距离的方差,选取方差最小的样本作为第一个初始质心;

b)计算样本簇向量距离的平均值 d;

Step2 选取与质心的距离不大于 d 的样本数据集构成新的样本簇 $\boldsymbol{W}$;

Step3 执行 Step1,Step2,直至每个样本簇的质心不再变化;

Step4 输出经过裁剪之后的训练数据集 $\boldsymbol{S}'\{s_1', s_2', s_3' \cdots s_n'\}$

通过 K-means 算法实现对训练数据集的裁剪,能够减少大量与样本训练集的相似度运算;但对于高维度样本数据而言,向量相似度计算的时间复杂度较高,因此对于高维度向量的相似度计算需要进一步降低训练数据集的维度[10].目前主要通过特征提取来降低向量的维度,常用的方法主要有基于成分分析算法(Principal Component Analysis,PCA)、线性判别分析算法(Linear Discriminant Analysis,LDA)以及基于信息增益的 ID3 算法.其中 PCA 算法属于无监督学习,无法有效地利用已有的样本分类信息;而传统 LDA 算法的能够将维度降低到类别数减一,不利于处理类别较少的高维度数据[11].

基于信息熵的 ID3 算法既能够有效地利用已分类的信息,同时不受向量维度大小的制约,在高维度数据集的降维处理中有着广泛的应用[12].

对于训练数据集 $\boldsymbol{S}$ 有 s 份样本可分成 m 类,第 i 类的实例个数为 s_i 则信息熵的计算公式定义为

$$\mathrm{Info}(\boldsymbol{S}) = -\sum_{i=1}^{m} p_i \log_2 p_i. \tag{4}$$

式中: p_i 是 $\boldsymbol{S}$ 中属于第 i 类的概率, $p_i = s_i / s$.

若选择属性 A, 将划分训练集 $\boldsymbol{S}$ 为 $\{s_1, s_2, \cdots s_i, s_n\}$, 则每个划分 s_i 的信息熵的计算公式定义为

$$\mathrm{Info}(S_i) = -\sum_{i=1}^{k} c_i \log_2 c_i. \tag{5}$$

根据选择的结果的不同,每个划分 s_i 被划分成 $\{d_1,d_2,\cdots d,d_k\}$,式(5)中的 $c_i = d_i/s_i$.

在属性 $\boldsymbol{A}$ 下的条件信息熵定义为

$$\mathrm{Info}(\boldsymbol{S}_i \mid \boldsymbol{A}) = \sum_{i=1}^{n} \frac{s_i}{S}\mathrm{Info}(\boldsymbol{S}_i)\ . \tag{6}$$

最终得到特征属性 $\boldsymbol{A}$ 带来的信息增益为

$$\mathrm{Gain}(\boldsymbol{A},\boldsymbol{S}) = \mathrm{Info}(\boldsymbol{S}) - \mathrm{Info}(\boldsymbol{S}_i \mid \boldsymbol{A})\ . \tag{7}$$

信息增益是在进行属性选择划分前和划分后信息的差值,而信息增益越大,区分样本的能力就越强,越具有代表性.利用属性的信息增益对于数据集进行降维处理的步骤如下.

算法 2　基于信息增益的属性剪枝

名称　DescendDim

输入　已分类的训练数据集 $\boldsymbol{S}$

输出　降维后的训练数据集 $\boldsymbol{S}'$

选取已分类的训练样本属性集 $\boldsymbol{A}$ $\{a_1,a_2,a_3\cdots a_n\}$:

Step1　计算属性集 $\boldsymbol{A}$ 中各属性的信息增益

a)选择 $\boldsymbol{A}$ 中信息增益最大的属性 a_g;

b)对属性 a_g 的每个取值 $a_g = n$ 将数据集 $\boldsymbol{S}$ 分割为若干非空子集 $\boldsymbol{S}_i$;

Step2　对于非空子集 $\boldsymbol{S}_i$, 以 $\{\boldsymbol{A} - a_g\}$ 为特征集执行步骤 Step1,直到筛选出 k 个属性的属性集 $\boldsymbol{A}'\{a_1,a_2,a_3\cdots.a_k\}$;

Step3　经过降维处理之后的属性集 $\boldsymbol{A}'$, 对训练数据集 $\boldsymbol{S}$ 规约得到训练数据集 $\boldsymbol{S}'$.

训练数据集在经过算法 1 处理之后实现了数据集的在量上面的规约,再通过算法 2 处理之后实现训练集在属性维度上的实现了规约;本文通过从训练数据集的数据量和属性维度规约 2 个方面来减少 KNN 分类算法的计算复杂度,实现 KNN 分类算法的改进.

2　KNN 分类算法的并行化

KNN 算法中涉及大量与训练数据集的向量相似度的计算,通过算法的并行化可大大提高运算速度.文中改进的 KNN 算法有很多的迭代计算,而 Spark 框架中 RDD 底层接口是基于迭代器的,从而使得数据访问变得更加高效同时避免了大量中间结构对内存的消耗, 不需要进行磁盘间的 I /O 操作,可以大大提高计算效率[13-14].Spark 并行计算的核心是使用 RDD 进行数据的分发和处理,算法的总体逻辑并没有太大的改变,本文所使用的算法的总体逻辑如图 2 所示.

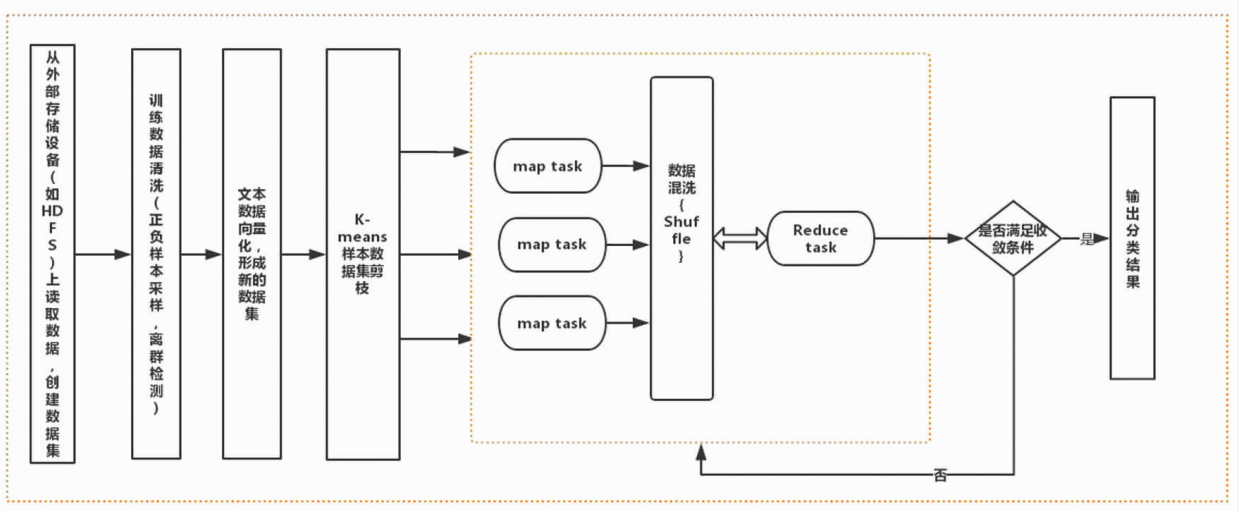

图 2　基于 Spark 的 KNN 并行分类的算法实现总流程

2.1 KNN 分类并行化算法设计

基于 Spark 的 KNN 分类算法可主要划分为 3 个主要处理阶段:

1) 文本数据预处理阶段,这一阶段包括文本数据的清洗,文本空间向量模型的构建.

2) 文本向量数据剪枝阶段,运用文中的算法 1 实现训练数据集的剪切与规范,减少迭代运算的次数;通过使用算法 2 实现向量维度的规约,最终得到比较规范的数据集.

3) Spark Map-Reduce 的并行计算阶段,在 Map 阶段通过并行的运算方式计算训练数据集与待测样本的向量相似度,并将中间结果缓存到集群中[15-16];在 Reduce 阶段,使用 Map 阶段的输出作为 Reduce 的输入,提取 K 个最相近的结果集,作为待测数据分类判别的依据.

基于 Spark 的 KNN 并行分类算法如算法 3 所示:

算法 3 KNN 并行分类算法

名称 ParallelKNN

输入 已分类的训练数据集 $\boldsymbol{S}$, 待测样本数据 $\boldsymbol{X}$, 类型数量 k

输出 待测样本数据 $\boldsymbol{X}$ 的分类类别 Y

选取已分类的 n 个样本簇 $\boldsymbol{S}\{s_1,s_2,s_3,\cdots,s_n\}$

Step1 使用文中算法 1 获取经过样本剪枝的样本簇 $\boldsymbol{S}'$;

Step2 使用文中算法 2 获取经过属性剪枝的样本簇 $\boldsymbol{S}''$;

Step3 通过样本簇 $\boldsymbol{S}''$ 创建 Spark 弹性分布式数据集 rdd_s;

Step4 定义 Spark 中 map 函数:通过式(3)计算样本间距离;

Step5 定义 Spark 中的 reduce 函数:选择余弦相似度最大的 k 个样本簇;

Step6 弹性数据集 rdd_s,待测样本数据 $\boldsymbol{X}$ 作为 map-reduce 输入,计算最近的 k 个样本簇;

Step7 筛选出 k 个样本簇中的最多的分类作为待测数据 X 的类别输出.

传统的 KNN 算法利用 Spark 计算平台经过并行化处理之后,能够有效地减少迭代运算的次数,提高算法的执行效率.

2.2 并行化实现的关键步骤

Spark 平台支持多种编程语言,文中的算法 3 在 Spark 中实现的关键步骤如下:

Step1 Spark 集群初始化;

Step2 从外存读取数据训练数据集,并 RDD 化;

Step3 训练数据集广播到集群各计算节点,提供并行计算的数据集;

Step4 KNN 距离运算并行化,获取计算结果数据集 rs_set;

Step5 处理最终的数据分类时,将 Step4 的计算结果集 rs_set 转化为 pair RDD(类别=>出现次数;

Step6 Spark 的 ReduceByKey 计算出近邻样本中出现的次数的最多的分类,即为待测样本所属的分类.

如图 3 所示,在 Map 阶段,测试数据集被拷贝成不同的分片分发到集群中的各计算节点中,同时待测数据集被分发到集群的计算节点中,以并行的方式与测试数据集进行相似度计算.完成集群 Map 运算之后进入 Reduce 阶段,汇总各节点的计算结果,筛选出符合要求的记录,统计其所属的类别,出现次数最多的类别为待测样本所属的分类.

3 实验设计与分析

3.1 实验环境与数据集

本文实验平台在 Hadoop 平台的 YARN 上部署 Spark 框架,框架结构如图 4 所示.通过 Vmware 创建了 8 台虚拟机,其中包含 1 个 Master 节点和 7 个 Slave 节点.虚拟机中操作系统均为 Ubuntu 14.04.3-Server-amd64 版本,Hadoop 版本为 2.6.2, Spark 版本为 1.4.3, Java 开发包版本为 jdk1.7.0_79, 程序开发工具为 Eclipse Mars.1. 其中 Master 节点担任 NameNode 角色, 主要负责管理与调用并维护着所有文件的元数据,

Slave 节点担任 DataNode 角色，根据 NameNode 的调用检索和处理数据.

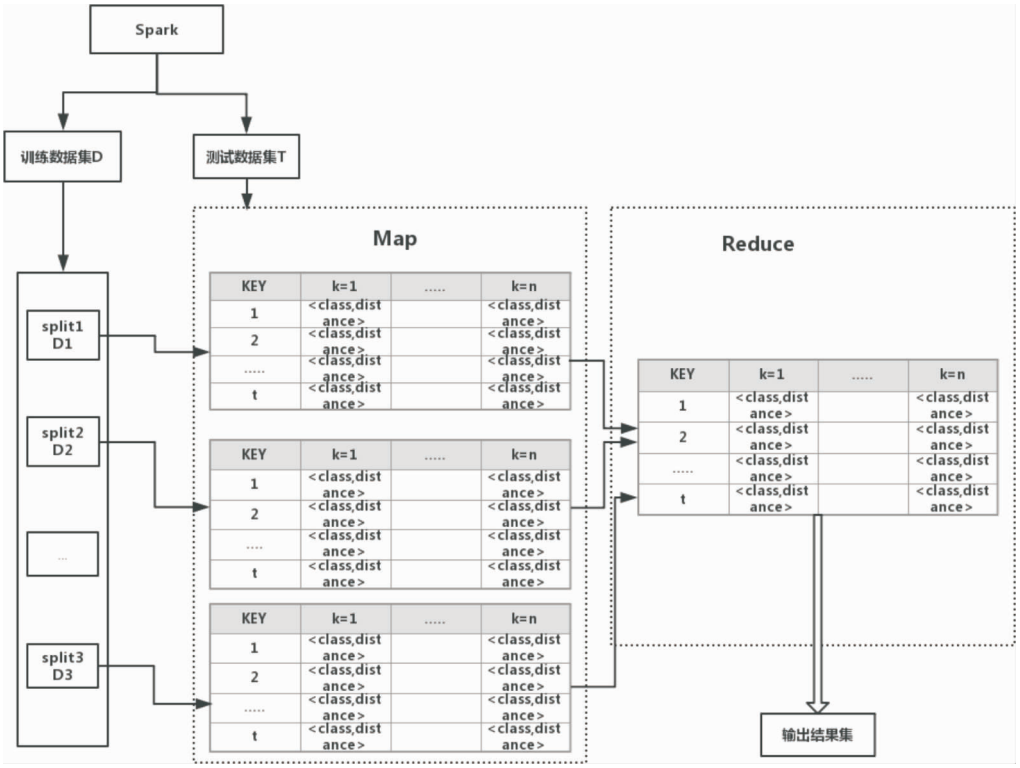

图 3　Spark 框架下 KNN 算法的并行化过程

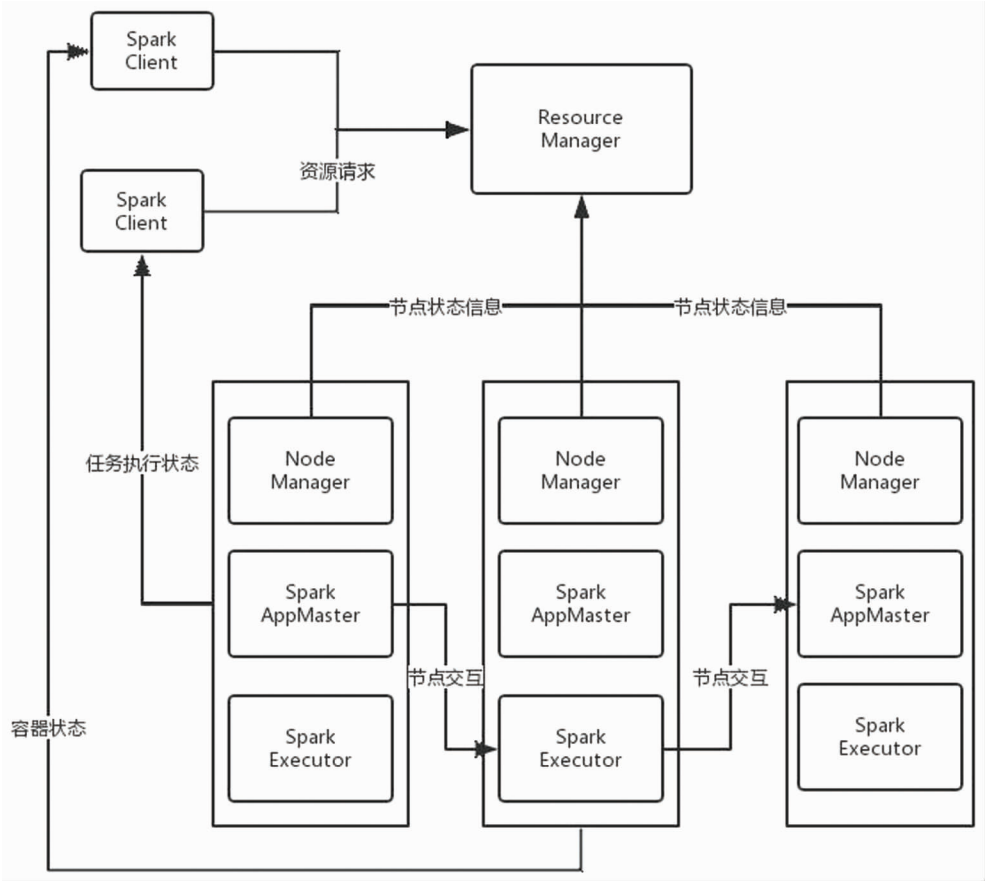

图 4　Spark On YARN 分布式计算架构

数据集采用由 Sogou 实验室提供的分类语料库,实验中使用的语料分为教育、文化、体育、招聘、旅游

等 10 个分类主题,每个分类中包含 3 000 篇文章,总计包括 30 000 篇文档,每个分类中随机选取 800 篇,组成训练文本集.剩余文本划分成不同大小,不同分类组合的测试集.

3.2 实验结果分析

在 Spark 计算框架中对得到的新训练文本集进行下一步的 KNN 文本分类处理(如表 1 所示),并在分类过程中迭代 K 值并获取最终分类结果. 最后通过比较准确率、运行时间、加速比来验证算法的执行性能.加速比公式为

$$\text{Speedup}(g) = \frac{\text{在一个节点上的执行时间}}{\text{在 } g \text{ 个节点上的执行时间}}. \tag{8}$$

准确率公式:

$$P_c = \frac{\text{分类正确的文档}}{\text{分类正确的文档} + \text{分类错误的文档}}. \tag{9}$$

实验首先以准确率(P_c)与运行时间(T)为指标在单一节点上对传统 KNN 分类算法、本文优化 KNN 分类算法以及基于 Spark 框架的优化 KNN 进行比较(见表 2).

表 1 文本测试集分类

测试集	规模篇数	说明
D_1	1 000	10 个分类,每分类随机抽取 100 篇
D_2	5 000	10 个分类,每类随机抽取 500 篇
D_3	5 600	8 个分类,每类随机抽取 700 篇
D_4	10 000	10 个分类,每类随机抽取 1 000 篇
D_5	20 000	10 个分类,每类随机抽取 2 000 篇

表 2 3 种算法性能比较

测试集	传统的 KNN 算法		本文优化的 KNN 算法		基于 Spark 框架的优化 KNN 算法	
	P_c /%	T /s	P_c /%	T /s	P_c /%	T /s
D_1	80.8	265.3	78.3	277.2	79.1	54.5
D_2	82.3	320.5	79.4	287.8	80.1	56.3
D_3	85.1	457.8	84.8	312.3	83.2	62.7
D_4	84.7	865.7	85.4	556.5	85.2	77.9
D_5	86.9	1 542.0	87.3	796.4	86.5	86.4

通过表 2 的数据可以看出在数据量较少的时候,优化的 KNN 算法的准确率低于传统的 KNN 算法,主要是由于训练数据集经过两次剪枝之后,训练数据集的数据量过小,向量维度降低,影响了测试数据的准确性,但随着数据量的增加准确率逐渐上升,最终逼近传统 KNN 分类的准确度,但经过优化的 KNN 算法的耗时要明显小于传统的算法.且在 Spark 环境下,处理大数据量时算法的准确率较好,并且时间效率得到了明显的提高;而在数据规模偏小时,算法准确率不是很高,主要因为并行算法采用分区计算,数据规模偏小影响了分类的准确.为了直观的比较,给出了 3 种算法的耗时折线图如图 5 所示.

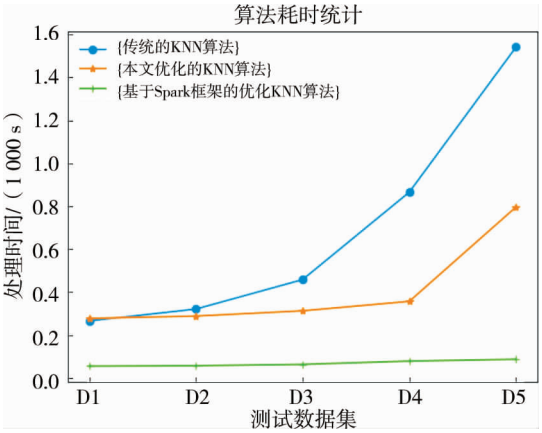

图 5 3 种算法运行时间比较

加速比是用来衡量并行系统性能的重要指标,为了验证 Spark 集群在文本分类过程相比较 Hadoop 的 MapReduce 集群是否具有性能优势,分别针对 2 种方案设计出具备相同节点规模的计算集群,运行文中提出的优化的 KNN 算法处理从 $D_1 \sim D_5$ 不同规模数据集,根据式(8)计算出的集群加速比,如图 6 所示.

通过图 6 的折线图,可以看出随着数据集规模的不断增大,基于 Spark 框架并行算法的系统处理能力

逐渐凸显出来，系统加速比逐渐增大；对比 Spark 和 Hadoop 集群，Spark 集群的加速性能优于 Hadoop 集群，并且随着数据规模的增大，Hadoop 集群的加速比趋于稳定而 Spark 集群依旧保持了一定的增长趋势.

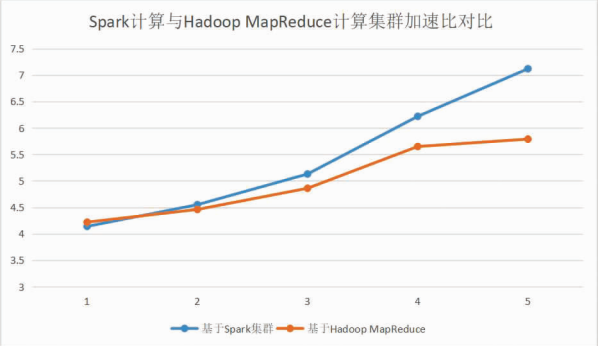

图 6　基于 Spark 框架优化的 KNN 分类算法的加速比

4　结论

1）从分类准确度来看，本文提出的算法在数据规模较小时，不占明显优势，但对于大规模数据集，相比实验中的对照组准确度更高，并且稳定性较好. 从计算耗时来看，本文采用基于内存运算的 Spark 框架相对于传统的 MapReduce 具备明显的优势，尤其在数据规模较大的场景中.

2）相对于简单的算法并行化，本文着重提出了训练数据的优化方法，通过对训练数据集的减枝和优化进一步提高了算法的效率和准确度.综合来看，本文提出的解决方案适用于解决对实时性要求较高的大规模文本分类问题.后续的研究主要考虑集成更多的分本分类技术进一步提高分类的准确度.

参考文献：

[1] Dean J, Ghemawat S. MapReduce：simplified data processing on large clusters[C]//Proceedings of the 2004 Conference on Symposium on Operating Systems Design&Implementation. Berkeley, CA：USENIX Association, 2004：10-10.

[2] Qu W, Cheng S, Wang H. Efficientfile accessing techniques on Hadoop distributed file systems[C]// Social Computing. Springer Singapore, 2016.

[3] 张承畅，张华誉，罗建昌，等.基于云计算和改进 K-means 算法的海量用电数据分析方法[J].计算机应用，2018，38(1)：159-164.

[4] 胡鹏威，倪志伟，伍章俊，等.基于 MapReduce 离散萤火虫群优化算法的服务选择方法[J].计算机工程，2018，44(1)：211-218.

[5] Yang S, Guo J Z, Jin J W. An improved Id3 algorithm for medical data classification[J]. Computers & Electrical Engineering, 2017：S0045790617325 17X.

[6] 王连喜，蒋盛益.一种基于特征聚类的特征选择方法[J].计算机应用研究，2015，32（5）：1305-1308.

[7] 张承畅，张华誉，罗建昌，等.基于云计算和改进 K-means 算法的海量用电数据分析方法[J].计算机应用，2018，38(1)：159-164.

[8] 黄震，钱育蓉，范迎迎，等.Spark 下遥感大数据特征提取的加速策略[J].计算机工程与设计，2017，38(12)：3279-3283.

[9] Qi R Z, Wang Z J, Li S Y. Aparallel genetic algorithm based on Spark for pairwise test suite generation[J]. Journal of Computer Science and Technology, 2016, 31(2)：417-427.

[10] 黄文辉，冯瑞.基于 Spark Streaming 的视频/图像流处理与新的性能评估方法[J].计算机工程与科学，2015，37(11)：2055-2060.

[11] 王磊，曾诚，奚雪峰，等.基于 Spark 的海量文本评论情感分析[J].苏州科技大学学报(自然科学版)，2018，35(1)：71-75.

[12] 刘鹏，王学奎，黄宜华，等.基于 Spark 的极限学习机算法并行化研究[J].计算机科学，2017，44(12)：33-37.

[13] Chen J, Chen H, Wan X Y, et al. MR-ELM：a MapReducebased framework for large-scale ELM training in big data era[J]. Neural Computing&Applications, 2016, 27（1）：101-110.

[14] 赵文芳，刘旭林.Spark Streaming 框架下的气象自动站数据实时处理系统[J].计算机应用，2018，38(1)：38-43.

[15] Zhao S, Yang X, Li X, et al. A Hadoop-based visualization and diagnosis framework for earth science data[C]//Proceedings of the 2015 IEEE International Conference on Big Data. Piscataway, NJ：IEEE, 2015：1972-1977.

[16] An C Q, Feng L I, Yue C, et al. Parameter optimization for Spark jobs based on runtime data analysis[J]. Computer Engineering & Science, 2016, 38（1）：11-19.