



基于 MapReduce 和 Spark 的大规模压缩模糊 K-近邻算法

王谟瀚^a, 翟俊海^{a,b}, 齐家兴^a

(河北大学 a. 数学与信息科学学院; b. 河北省机器学习与计算智能重点实验室, 河北 保定 071002)

摘 要: 压缩模糊 K-近邻(CFKNN)算法仅适用于中小数据环境,且其样例选择采用静态机制,导致算法不能对阈值进行动态调整从而选出最优样例。为此,对 CFKNN 算法进行改进,将其扩展到大规模数据环境,提出分别基于 MapReduce 和 Spark 的 2 种大规模压缩模糊 K-近邻算法。在样例选择阈值设置方面,引入动态机制,使得所选样例更具代表性。在具有 7 个数据节点的大数据平台上进行实验,结果表明,与 CFKNN 算法相比,所提 2 种算法具有更高的分类精度和加速比。2 个平台相比,MapReduce 产生的中间文件数目多于 Spark,而 Spark 在运行时间和同步次数上优于 MapReduce。

关键词: MapReduce 平台;Spark 平台;模糊 K-近邻;样例选择;动态机制

开放科学(资源服务)标志码(OSID):



中文引用格式:王谟瀚,翟俊海,齐家兴. 基于 MapReduce 和 Spark 的大规模压缩模糊 K-近邻算法[J]. 计算机工程, 2020,46(11):139-147.

英文引用格式:WANG Mohan, ZHAI Junhai, QI Jiaxing. Large-scale condensed fuzzy K-nearest neighbor algorithm based on MapReduce and Spark[J]. Computer Engineering, 2020,46(11):139-147.

Large-Scale Condensed Fuzzy K-Nearest Neighbor Algorithm Based on MapReduce and Spark

WANG Mohan^a, ZHAI Junhai^{a,b}, QI Jiaxing^a

(a. College of Mathematics and Information Science; b. Hebei Key Laboratory of Machine Learning and Computational Intelligence, Hebei University, Baoding, Hebei 071002, China)

[Abstract] The Condensed Fuzzy K-Nearest Neighbor (CFKNN) algorithm is applicable only to small-sized and medium-sized data sets, and its mechanism of instance selection is static, so the algorithm can not adjust the threshold dynamically to select the optimal sample. To address the problems, this paper improves the CFKNN algorithm to make it applicable it to large-scale data environment, and on this basis proposes two kinds of large-scale condensed fuzzy K-nearest neighbor algorithms based on MapReduce and Spark. A dynamic mechanism is introduced into the setting of the instance selection threshold to make the selected instances more representative. Experiments are carried out on a big data platform with 7 data nodes. The experimental results show that compared with the CFKNN algorithm, the two proposed algorithms have better classification accuracy and acceleration ratio. Results of comparison between the two proposed algorithms show that MapReduce produces more intermediate files than Spark, yet Spark outperforms MapReduce in terms of running time and synchronization times.

[Key words] MapReduce platform; Spark platform; fuzzy K-Nearest Neighbor (K-NN); instance selection; dynamic mechanism

DOI:10.19678/j.issn.1000-3428.0055670

0 概述

K-近邻(K-Nearest Neighbor, K-NN)^[1]是一种常用的分类方法,广泛应用于模式识别、数据挖掘和

机器学习等领域。K-NN 方法简单且易于编程实现,但是其存在 2 个问题,一是对测试样例进行分类时需要存储训练集中的所有样例,并计算测试样例与训练集中所有样例之间的距离,二是对于每一个测

基金项目:国家自然科学基金(71371063);河北省自然科学基金(F2017201026);河北省科技计划重点研发基金(19210310D);河北大学研究生创新项目基金(hbu2019ss077)。

作者简介:王谟瀚(1992—),男,硕士研究生,主研方向为分布式计算、机器学习;翟俊海(通信作者),教授;齐家兴,硕士。

收稿日期:2019-08-05 **修回日期:**2019-11-10 **E-mail:**mczjh@126.com

试样例,用训练集中的样例对它们进行分类时,每个训练样例被认为是同等重要的。

针对第一个问题,HART^[2]于1968年提出了压缩近邻(Condensed Nearest Neighbor, CNN)算法。但是,CNN算法对噪声非常敏感,且其结果与样例选择顺序有关。对此,研究人员提出了较多改进算法,如RNN(Reduced Nearest Neighbor)^[3]、ENN(Edited Nearest Neighbor)^[4]和ICF(Iterative Case Filtering)^[5]等。近年来,针对K-NN对测试样例进行分类时需存储训练集中所有样例的问题,学者们也提出了一些较好的解决方法。HOU等人^[6]将哈希技术与决策树相结合,提出基于树的紧哈希方法,该方法可显著提高近邻样例的搜索效率。梁聪等人^[7]提出一种基于参考点的改进K-NN分类算法。该算法依据点到样本距离的方差选择参考点,并赋予参考点自适应权重,与基本K-NN算法及kd-tree近邻算法相比,其具有较高的分类精度及较低的时间复杂度。基于谱哈希技术,WAN等人^[8]提出针对高维数据的近似近邻搜索算法。基于分布式哈希技术,文庆福等人^[9]提出一种近似近邻搜索方法。ALVAR等人^[10]使用局部敏感哈希技术,提出针对大规模数据集的样例选择算法,该算法的时间复杂度达到线性级。针对投影哈希中投影误差较大、二进制编码时原始信息丢失严重等问题,杨定中等人^[11]提出一种近似最近邻搜索方法,该方法通过多阶段量化策略降低编码过程中的投影及量化误差。罗辛等人^[12]提出一种基于相似度支持度的最近邻度量方法,其在保证分类精度的前提下降低了计算复杂度。乔玉龙等人^[13]利用向量的方差和小波域中的逼近系数得出2个重要不等式,利用不等式排除不可能成为K-近邻的向量,进而降低了计算复杂度。受交叉验证思想的启发,ZHAI等人^[14]提出交叉样例选择算法,该算法可解决大规模样例的选择问题。SONG等人^[15]将针对分类问题的样例选择算法移植到回归场景,提出一种针对K-NN回归问题的排序样例选择算法,其扩大了样例选择的应用范围。

近年来,大数据技术在很多领域得到广泛关注与应用,一些科研人员针对大数据的近邻搜索问题进行了研究。基于开源大数据平台,MUJA等人^[16]提出具有可扩展性的最近邻算法。基于MapReduce大数据计算平台,ZHAI等人^[17]设计基于投票机制和随机权网络的大数据样例选择算法。基于Spark大数据计算平台,MAILLO等人^[18]提出大数据K-近邻搜索算法。SONG等人^[19]对基于MapReduce的K-NN算法进行了具体的理论分析。

针对第二个问题,KELLER^[20]于1985年提出了模糊K-NN算法。然而,模糊K-NN算法依然存在上述第一个问题。为此,ZHAI等人^[21]提出了压缩

模糊K-近邻(Condensed Fuzzy K-NN, CFKNN)算法。但是,CFKNN算法仅适用于中小数据场景,在大数据环境中,CFKNN会出现计算效率低的问题,甚至不可实现。此外,CFKNN的样例选择采用静态机制,导致该算法的性能提升存在局限性。

为了解决上述问题,本文基于MapReduce和Spark提出2种大规模压缩模糊K-近邻算法。将CFKNN算法扩展到大数据环境,在MapReduce和Spark 2种并行计算框架上实现面向大规模数据环境的压缩模糊K-近邻算法,以降低CFKNN的计算复杂度并缩短算法的运行时间。在样例选择过程中,对阈值进行动态调整,从而提高算法的动态特性。

1 相关知识

1.1 CFKNN 算法

设 T 是训练集, S 是所选样例的集合, C 为训练集类别属性,训练集共分为 p 类。在初始时,从训练集 T 中的每类随机选择一个样例加入 S ,然后根据算法1计算 S 中样例的模糊隶属度,用算法2确定 x 的类别隶属度,并通过类别隶属度计算样例 x 的信息熵。如果样例 x 的信息熵大于所设阈值,则将样例 x 加入到 S 中,否则丢弃 x 。当训练集 T 为空时,算法终止,输出所选样例集合 S 。CFKNN算法的伪代码如算法3所示。

算法1 模糊隶属度算法

输入 所选样例集合 S

输出 样例的隶属度 $\mu_{ij} = \mu_j(x_i), 1 \leq i \leq n, 1 \leq j \leq p$

1. 对于 $\forall j, 1 \leq j \leq p$, 计算每一类的中心 C_j
2. 对于 $\forall i, j$, 计算 x_i 到各类中心 C_j 的距离 d_{ij}
3. 对于 $\forall i, j$, 按式(1)计算 $\mu_{ij} = \mu_j(x_i), 1 \leq i \leq n, 1 \leq j \leq p$
4. 返回样例隶属度

$$\mu_j(x_i) = \frac{d_{ij}^2}{\sum_{j=1}^p d_{ij}^2}, 1 \leq i \leq n, 1 \leq j \leq p \quad (1)$$

算法2 F-KNN 算法

输入 所选样例集合 $S = \{(x_i, y_i) | x_i \in R^d; y_i \in Y\}, 1 \leq i \leq n$, 样例 x

输出 样例 x 隶属于每一类的隶属度 $\mu_j(x), 1 \leq j \leq p$

1. 利用算法1计算 S 中每一个样例的类别隶属度 $\mu_{ij} = \mu_j(x_i)$, 构成一个 n 行 p 列的矩阵 μ
2. 在 S 中找到 x 的 K 个近邻
3. 利用式(2)确定 x 的类别隶属度:

$$\mu_j(x) = \frac{\sum_{i=1}^k \mu_{ij} \left(\frac{1}{\|x - x_i\|^{\frac{1}{\alpha-1}}} \right)}{\sum_{i=1}^k \left(\frac{1}{\|x - x_i\|^{\frac{1}{\alpha-1}}} \right)}, 1 \leq j \leq p \quad (2)$$

4. 输出 $j(x)$

算法3 CFKNN 算法

输入 训练集 T , 参数 k 和 α (假设 T 的样本容量为 n , T 中的样例共分为 p 类)

输出 $S \subseteq T$

1. 从 T 中的每类随机选一个样例加入 S 中
 2. For x in $T - S$ do
 3. 根据算法 1 计算 S 中样例的类别隶属度
 4. 用算法 2 确定 x 的类别隶属度 $(\mu(x, C_1), \mu(x, C_2), \dots, \mu(x, C_p))$
 5. 根据式(3)计算 x 的熵 $\text{Entr}(x)$:
- $$\text{Entr}(x) = - \sum_{i=1}^p \mu_i(x) \lg \mu_i(x) \quad (3)$$
6. If $\text{Entr}(x) > \alpha$
 7. $S = S \cup \{x\}$
 8. End if
 9. End for
 10. Return S

1.2 MapReduce 和 Spark 并行计算框架

MapReduce^[22]是由 Google 公司提出的一种面向大规模数据的并行计算模型, MapReduce 继承了函数式编程语言 LISP 中 map 函数和 reduce 函数的思想, 采用分治策略处理大数据。在初始阶段, MapReduce 自动将大数据集划分为若干子集部署到云计算节点上, map 阶段将数据变换为键值对数据。reduce 阶段在 map 阶段的基础上, 对已经归纳好的数据做进一步处理, 得到最终计算结果。通过 map 和 reduce 2 个阶段, 完成对大规模数据的并行化处理。

Spark^[23]是处理大数据的快速通用引擎, 2009 年由加州大学伯克利分校对外开源, 随后凭借其快速、通用以及可扩展等优势, 迅速成为 Apache 顶级项目。Spark 起初是为了克服 Hadoop 并行计算框架的不足而被提出, 发展至今, Spark 已经成为包含 SparkSQL、Spark Streaming、Spark GraphX 和 Spark MLlib 等子项目在内的生态系统。Spark 将 MapReduce 基于磁盘的存储和容错机制改为基于内存的机制, 提高了计算速度。通过将执行模型抽象为有向无环图(Directed Acyclic Graph, DAG), 并根据弹性分布式数据集(Resident Distributed Dataset, RDD)间的宽依赖和窄依赖关系, 串联或并行执行多个阶段的任务, 无需将不必要的中间结果输出到 HDFS(Hadoop Distributed File System)上, 以此提高计算效率。

RDD 和算子是 Spark 的核心与基础。RDD 是 Spark 中的基本数据抽象, 其为不可变、可分区、可并行计算的数据集合。在具体的逻辑实现上, RDD 将数据分为若干分区, 分区以分布式方式保存在云节点上, 既可以存储在内存中, 也可以存储在外存中。当某些数据需要重复使用时, RDD 允许用户显式地将数据缓存在内存中, 从而有效提高了计算速度。在对 RDD 中的数据进行处理时, 需要通过 Spark 算子来实现相应的数据操作。一般根据是否会触发 Spark 作业执行将 Spark 算子分为如下两类:

- 1) 转换算子, 其对 RDD 进行转换操作, 将一个

RDD 转换为另一个 RDD。转换算子的转换操作是延时加载的, 它们不会直接返回计算结果, 只记录转化动作。

- 2) 行动算子, 其触发 Spark 作业执行, 得到 Spark 作业的计算结果。

2 大规模压缩模糊 K-近邻算法

2.1 算法描述

通过对原始 CFKNN 算法进行分析可以得出, 该算法难以在大数据环境下进行应用的 3 个主要原因具体如下:

- 1) 在确定 T 中样例 x 的类别隶属度时, 首先需要计算集合 S 的隶属度矩阵 m , 然后寻找 k 个近邻, 计算样例 x 的类别隶属度。当训练集 T 为大数据集时, 集合 S 中样例增多, 对 T 中的每个样例寻找 S 中的 k 个近邻并计算 T 中每个样例的熵值时, 算法计算量很大, 算法运行时间超出可接受范围。

- 2) 当训练集为大数据集时, 寻找 k 个近邻的计算复杂度大幅增加。

- 3) 对所选样例集合 S 不能实时更新, 进而导致对当前样例 x 的隶属度和信息熵计算不准确, 这是导致原始 CFKNN 算法无法在大数据环境下应用的主要原因。

针对以上问题, 本文提出大规模压缩模糊 K-近邻算法, 该算法对原始 CFKNN 做出如下改进:

- 1) 针对第一个问题, 大规模压缩模糊 K-近邻算法在计算样例 x 的类别隶属度时, 先在 S 中寻找样例 x 的 k 个近邻, 然后只计算 k 个样例的类别隶属度, 从而大幅降低了由于计算 S 中所有样例的隶属度所引起的计算复杂度(对应算法 4 第 5 行)。

- 2) 利用并行计算框架, 在每个计算节点上并行地寻找集合 S 中样例 x 的 k 个近邻, 从而解决第二个问题。

- 3) 对于第三个问题, 本文在阈值设置上引入动态机制, 对阈值进行动态调整, 将阈值 α 设置为迭代次数 j 的单调递减函数, 如式(4)所示。其中, 设置初始阈值 initEntropy 时考虑到对应类别数的最大熵(对应算法 4 第 6 行)。通过引入动态阈值机制, 使得本文算法训练出的分类器较原始 CFKNN 算法具有更好的分类精度。

$$\alpha(j, n) = \begin{cases} \text{initEntropy}, & j = 1 \\ \alpha(j-1, n) - \frac{e^{\frac{j}{n}}}{10n}, & 1 < j \leq n \end{cases} \quad (4)$$

其中, j 为当前迭代次数, n 为总迭代次数。

本文基于并行计算框架的大规模压缩模糊 K-近邻算法伪代码如算法 4 所示, 算法流程如图 1 所示。

算法 4 大规模压缩模糊 K-近邻算法

输入 数据集 T , 近邻数 k , 阈值 α , 迭代次数 iterations

输出 数据子集 $S \subseteq T$

1. for iteration in iterations do

2. 初始化 S, T

3. 根据式(3)并行计算 T 中每个样例 x 的信息熵 $\text{Entr}(x)$

4. 根据式(4)计算 $\alpha(j, n)$

5. if $\text{Entropy} > \alpha(j, n)$

6. $S = S \cup \{x\}$

7. End if

8. 输出 S

9. End for

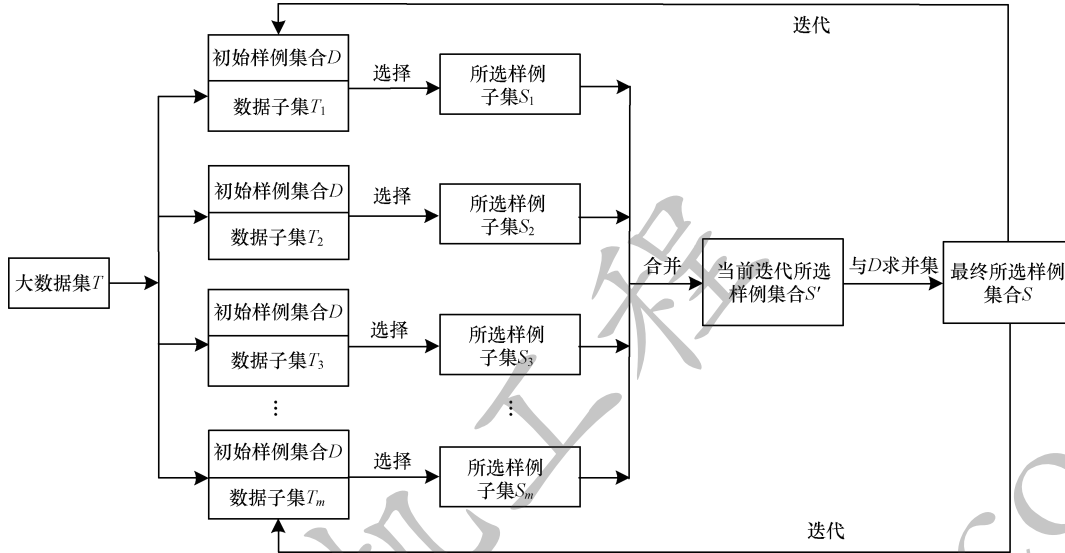


图1 大规模压缩模糊 K-近邻算法流程

Fig.1 Procedure of large-scale condensed fuzzy K-nearest neighbor algorithm

2.2 基于 MapReduce 的压缩模糊 K-近邻算法

在原始 CFKNN 算法中,当 T 为大数据集时,针对 T 中的每个样例,在 S 中寻找其 k 个近邻以及计算 T 中每个样例熵值的过程,会大幅提高算法的计算量。因此,本文将此过程通过 MapReduce 框架进行并行执行。对于 T 中的样例,并行寻找 S 中 k 个近邻并计算熵值,从而大幅缩短算法的运行时间。随着 T 中样例数量的增加,可以通过增加计算节点个数,使得算法维持可接受的运算时间并容易扩展。大规模压缩模糊 K-近邻算法在 MapReduce 中的计算,即基于 MapReduce 的压缩模糊 K-近邻(MR-CFKNN)算法流程如算法 5 所示。该算法分为 Mapper 阶段和 Reducer 阶段 2 个部分,Mapper 阶段包含 setup 和 map 2 个方法,Reducer 阶段只包含 reduce 方法。在 Mapper 阶段的 setup 方法中,首先对数据子集 S 进行初始化(算法 5 第 3 行),map 方法计算输入的样例 $t \in T$ 在 S 中的 k 个近邻(算法 5 第 6 行),由 k 个近邻计算出 t 的熵值 Entropy (算法 5 第 7 行),若 Entropy 大于熵的阈值 α ,则输出样例 t 。Reducer 阶段不做任何操作直接输出所选样例。

算法 5 MR-CFKNN 算法

输入 数据集 T , 近邻数 k , 阈值 α

输出 数据子集 $S \subseteq T$

1. ClassMapper

2. 执行 $\text{setup}()$ 方法,对所需资源进行初始化

3. 初始化数据子集 $S \subseteq T$

4. 初始化参数 $k, \text{initEntropy}$

5. 使用 map 方法对样例进行格式化操作, $\text{map}(\text{sid id}, \text{instance } t)$

6. 使用 findKNN 方法找到当前样例的 k 个近邻, $\text{Array kNearestNeighbor} = \text{findKNN}(t, S, k)$

7. 使用 fKNN 方法计算当前样例的熵, $\text{Entropy} = \text{fKNN}(\text{kNearestNeighbor}, t)$

8. If $\text{Entropy} > \alpha$

9. $\text{context.write}(\text{NullWritable}, t)$

10. End if

11. Mapper end

12. Class Reducer

13. 使用 reduce 方法对样例进行格式化, $\text{reduce}(\text{NullWritable}, [t^{(1)}, t^{(2)}, \dots])$

14. for t in $[t^{(1)}, t^{(2)}, \dots]$ do

15. 输出所选样例, $\text{context.write}(\text{NullWritable}, t)$

16. End for

17. Reducer end

2.3 基于 Spark 的压缩模糊 K-近邻算法

由于本文大规模压缩模糊 K-近邻算法为迭代算法,因此在 MR-CFKNN 的基础上,实现基于 Spark 的压缩模糊 K-近邻(Spark-CFKNN)算法,即大规模压缩模糊 K-近邻算法在 Spark 中的计算算法。具体地,对于给定的大数据训练集 T ,以一轮迭代为例,首先从 T 的每类样例中随机取出 c 个样例,加入到初始样例集合 D 中(算法 6 第 4 行),其次将 T 中的其余样例划分为若干子集 T_1, T_2, \dots, T_m ,部署到 m 个计算节点上(算法 6 第 5 行),并将 D 作为广播变量存储在每个计算节点上(算法 6 第 7 行),然后使用

map 算子计算得到 t 在 S 中的 k 个近邻,基于 k 个近邻计算 t 的类别隶属度(算法 6 第 14 行),在计算得到熵 $\text{Entr}(t)$ 后,对大于阈值 α 的样例进行筛选,将得到的 m 个所选样例子集 S_1, S_2, \dots, S_m 合并,得到当前迭代所选样例集合 $S' = \cup_{i=1}^m S_i$,最后将当前迭代所选样例集合 S' 与 D 求并集,得到所选样例集合 $S = D \cup S'$ (算法 6 第 14 行~第 21 行),将 S 作为新的初始样例集合以及新的广播变量部署在每个计算节点上。重复算法 6 第 6 行~第 22 行,直到完成算法所设置的迭代次数 iterations。

算法 6 Spark-CFKNN 算法

输入 数据集 T , 近邻数 k , 阈值 α , 迭代次数 iterations

输出 数据子集 $S \subseteq T$

1. 对数据集 T 进行初始化 RDD 操作, $\text{val trainInitRDD} = \text{sc. textFile}(T)$
2. 得到初始样例集合 D , $\text{var dRDD} = \text{trainInitRDD. combineByKey}(). \text{map}(). \text{flatMap}()$
3. 得到 T 与 D 的差集, $\text{var tRDD} = \text{trainInitRDD. subtract}(\text{dRDD})$
4. **for** ($i < -0$ until iterations) **do**
5. 对数据集 D 进行广播操作, $\text{var dInsbroad} = \text{sc. broadcast}(\text{dRDD. collect}())$
6. $\text{val distanceRDD} = \text{tRDD. map}(\text{line} = > \{$
7. **for** ($i < -0$ until $\text{dInsbroad. value. length}$) **do**
8. 计算当前样例与 D 中样例的距离, $\text{Distance}(\text{dInsbroad. value}(i), \text{line})$
9. **End for**
10. $\})$
11. $\text{val tEntropyAndSelectRDD} = \text{distanceRDD. map}(\text{line} = > \{$
12. 计算当前样例的隶属度, $\text{memShipDevide}(\text{trainInsMemberShipCalc}(k\text{NearestNeighbor}))$
13. 计算当前样例的熵值, $\text{val Entropy} = \text{calcEntropy}()$
14. **If** $\text{Entropy} > \alpha$
15. $S_m = S_m \cup \{x\}$
16. $T_m = T_m - \{x\}$
17. **End if**
18. $\})$
19. 将当前迭代所选样例与 D 求并集, $\text{dRDD} = \text{dRDD. union}(\text{tEntropyAndSelectRDD})$
20. 将 T 与当前迭代所选样例做差集, $\text{tRDD} = \text{tRDD. subtract}(\text{tEntropyAndSelectRDD})$
21. **End for**

3 实验结果与分析

为了验证本文算法的有效性,在 4 个大数据集上进行实验,数据集的基本信息如表 1 所示。4 个大数据集包括 2 个人工数据集和 2 个 UCI 数据集,第一个人工数据集是二分类数据集,每类包含 250 000 个样例点,共 500 000 个样例,且服从高斯分布 $p(x|\omega_i) \sim N(\mu_i, \sum_i)$, $i=1,2$,具体参数如表 2 所示。第二个人工数据集是一个三分类二维数据集,每

类包含 200 000 个样本点,且服从如下概率分布:

$$p(x|\omega_1) \sim N(0, I)$$

$$p(x|\omega_2) \sim N\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, I\right)$$

$$p(x|\omega_3) \sim \frac{1}{2}N\left(\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, I\right) + \frac{1}{2}N\left(\begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix}, I\right)$$

表 1 数据集基本信息

Table 1 Basic information of datasets

数据集	样例个数	属性个数	类别个数
Gaussian1	500 000	2	2
Gaussian2	600 000	2	3
Healthy Older People	75 128	8	4
Skin Segmentation	240 000	3	2

表 2 高斯分布参数

Table 2 Parameters of Gaussian distribution

i	μ	Σ
1	$\begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}$	$\begin{bmatrix} 0.6 & -0.2 \\ 0.2 & 0.6 \end{bmatrix}$
2	$\begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix}$	$\begin{bmatrix} 0.2 & -0.1 \\ 0.1 & 0.2 \end{bmatrix}$

本文从文件数目、同步次数、分类精度、所选样例个数以及算法运行时间等方面,将 MR-CFKNN 算法和 Spark-CFKNN 算法进行对比。此外,分别将原始 CFKNN 算法和本文算法所筛选出的样例作为训练集,使用 KNN 算法对测试集进行分类并比较分类精度。在实验过程中,分别从 4 个大数据集中随机选取部分样例作为测试集,测试集基本信息如表 3 所示。实验所用的大数据平台环境的配置信息如表 4 所示,大数据平台环境的节点规划如表 5 所示。

表 3 测试集基本信息

Table 3 Basic information of test sets

数据集	类别 1	类别 2	类别 3	类别 4	样本容量
Gaussian1	8 939	8 887	无	无	17 826
Gaussian2	4 838	4 747	4 924	无	14 509
Healthy Older People	25	25	25	25	100
Skin Segmentation	2 527	2 528	无	无	5 057

表 4 实验配置信息

Table 4 Experimental configuration information

软硬件项目	配置情况
CPU	Inter Xeon E5-4603 2.0 GHz(双核)
内存	16 GB RDIMM
硬盘	1 TB
网卡	Broadcom 5720 QP 1 Gb 网络子卡(四端口)
网络设备	华为 S3700 系列以太网交换机
操作系统	CentOS 6.4
Hadoop 平台	Hadoop-2.7.1 Spark-2.3.1
JDK 版本	JDK1.8

表 5 大数据平台环境的节点规划

Table 5 Nodes planning of big data platform environment

节点编号	主机名	IP 地址	节点类型
1	Master1	10.187.86.242	Master, Namenode, ResourceManager
2	Node1	10.187.86.243	Worker, DataNode, NodeManager
3	Node2	10.187.86.244	Worker, DataNode, NodeManager
4	Node3	10.187.86.245	Worker, DataNode, NodeManager
5	Node4	10.187.86.246	Worker, DataNode, NodeManager
6	Node5	10.187.86.247	Worker, DataNode, NodeManager
7	Node6	10.187.86.248	Worker, DataNode, NodeManager
8	Node7	10.187.86.249	Worker, DataNode, NodeManager

表 6 所示为原始 CFKNN 算法与本文大规模压缩模糊 K-近邻算法在 Gaussian1 数据集上的实验结果。从表 6 可以看出,CFKNN 算法只对所有训练样例进行 1 次迭代,MR-CFKNN 算法和 Spark-CFKNN 算法分别进行了 3 次、4 次和 5 次迭代。表中所示的分类精度是分别将原始 CFKNN 算法、本文算法所筛选出的样例集合作为训练集,使用 KNN 算法进行分类精度测试的结果。大规模压缩模糊 K-近邻算法的分类精度优于原始 CFKNN 算法的主要原因有 2 点:首先,CFKNN 算法只对训练集进行 1 次迭代,为了保证所选样例更具代表性,同时考虑到阈值为固定值,所以 CFKNN 算法需要将阈值设置为中间数值,这会导致在算法运行初期选入较多的非边界样例;其次,大规模压缩模糊 K-近邻算法考虑算法运行初期训练样例的熵值普遍较高的情况,且随着算法的不断迭代,训练样例的熵值逐渐接近真实值,所以其引入了动态阈值策略,使阈值随着迭代次数的增加而逐渐衰减,以此来克服原始 CFKNN 算法的缺点。以上 2 点使得大规模压缩模糊 K-近邻算法的分类精度优于原始 CFKNN 算法,说明本文算法所筛选出的样例更具代表性。

表 6 3 种算法的分类精度比较

Table 6 Comparison of classification accuracy of three algorithms

算法	迭代次数	分类精度
CFKNN	1	0.828 56
	3	0.987 71
MR-CFKNN	4	0.987 71
	5	0.987 71
Spark-CFKNN	3	0.987 71
	4	0.987 71
	5	0.987 67

加速比 (SpeedUp) 是衡量并行计算算法性能和效果的常用指标,传统的加速比计算方式如式(5)所示,其中, T_{best} 为串行算法在一台计算机上的最优运行时间, $T(N)$ 为并行算法在 N 台计算机上的运行时间。

$$\text{SpeedUp} = \frac{T_{best}}{T(N)} \quad (5)$$

由于本文大规模压缩模糊 K-近邻算法改变了 CFKNN 算法对训练集只进行 1 次迭代的计算方式,因此本文通过式(6)来计算 Spark-CFKNN 算法的加速比,以此为例来衡量通过大规模压缩模糊 K-近邻算法取得的并行化效果,其中, $T_{single-best}$ 为 Spark-CFKNN 算法在单一节点上的最优运行时间。

$$\text{SpeedUp}_{single} = \frac{T_{single-best}}{T(N)} \quad (6)$$

表 7 所示为 Spark-CFKNN 算法在单一节点和 7 个节点上的运行时间,从表 7 可以看出,本文算法大幅缩短了运行时间,具有较高的加速比。

表 7 Spark-CFKNN 算法在 Gaussian1 数据集上的加速比

Table 7 SpeedUp of Spark-CFKNN algorithm on Gaussian1 dataset

节点个数	迭代次数	运行时间/s	加速比
1	3	5 064	无
	4	5 796	无
	5	5 808	无
7	3	769	6.585 18
	4	796	7.281 41
	5	834	6.964 02

图 2 所示为不同迭代次数下 Spark-CFKNN 和 MR-CFKNN 的分类精度。从图 2 可以看出,Spark-CFKNN 和 MR-CFKNN 的分类精度变化趋势大致相同,说明 2 个算法在实现细节上具有逻辑一致性,同时也没有因为平台间运行机制的差异而对分类结果造成影响。此外,由图 2 可知,分类精度和迭代次数并不成简单的正比关系,即分类精度并不会随着迭代次数的增加而持续增长,当达到一定迭代次数时,分类精度会趋于恒定,这也说明为了提高分类精度而持续增加迭代次数的做法不可行。

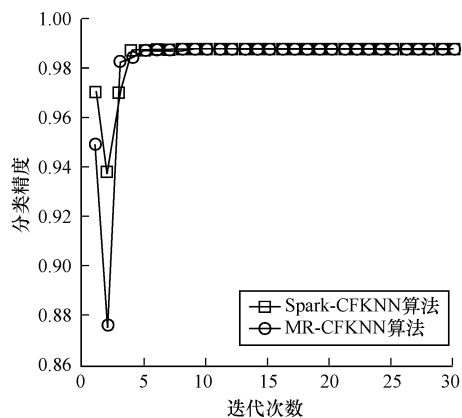


图 2 Gaussian1 数据集上分类精度随迭代次数的变化情况

Fig. 2 Variation of classification accuracy with iteration times on Gaussian1 dataset

根据吴信东等人^[24]的研究成果,本文对 2 种大数据平台在不同迭代次数下的文件数目、同步次数和算法运行时间进行对比,结果如表 8 ~ 表 12 所示。由于文件数目和同步次数只与大数据平台的调度机制有关而与数据集无关,因此对该指标进行对比时不区分数据集。

表 8 不同迭代次数下文件数目和同步次数的比较

Table 8 Comparison of the number of files and synchronization times under different iteration times

迭代次数	文件数目		同步次数	
	MapReduce	Spark	MapReduce	Spark
3	14	280	6	5
4	42	280	8	6
5	112	280	10	7

表 9 2 种平台在 Gaussian1 数据集下的性能对比

Table 9 Performance comparison of two platforms under Gaussian1 dataset

迭代次数	分类精度		样例选择个数		算法运行时间/s	
	MapReduce	Spark	MapReduce	Spark	MapReduce	Spark
3	0.987 71	0.987 71	32 292	37 726	5 499	769
4	0.987 71	0.987 71	32 323	38 053	14 351	796
5	0.987 71	0.987 67	32 329	38 248	19 127	836

表 10 2 种平台在 Gaussian2 数据集下的性能对比

Table 10 Performance comparison of two platforms under Gaussian2 dataset

迭代次数	分类精度		样例选择个数		算法运行时间/s	
	MapReduce	Spark	MapReduce	Spark	MapReduce	Spark
3	0.473 09	0.487 15	271 257	269 569	34 409	5 088
4	0.473 57	0.485 70	290 478	286 900	48 665	5 199
5	0.473 77	0.485 22	310 589	299 008	63 142	10 910

表 11 2 种平台在 Skin Segmentation 数据集下的性能对比

Table 11 Performance comparison of two platforms under Skin Segmentation dataset

迭代次数	分类精度		样例选择个数		算法运行时间/s	
	MapReduce	Spark	MapReduce	Spark	MapReduce	Spark
3	0.988 35	0.987 14	13 325	12 443	186	125
4	0.988 00	0.990 31	14 012	13 970	314	252
5	0.989 42	0.996 24	14 581	14 194	628	444

表 12 2 种平台在 Healthy Older People 数据集下的性能对比

Table 12 Performance comparison of two platforms under Healthy Older People dataset

迭代次数	分类精度		样例选择个数		算法运行时间/s	
	MapReduce	Spark	MapReduce	Spark	MapReduce	Spark
3	0.68	0.64	17 023	16 476	112	88
4	0.83	0.79	53 841	51 117	197	154
5	0.80	0.80	59 856	59 232	411	300

文件数目主要指中间文件数目,因为算法在运行过程中所产生的中间文件数量不仅会占用内存空间,还会影响磁盘的 I/O 性能,最终导致算法运行时间延长。在 MapReduce 中,每次的 shuffle 操作会对 map 产生的中间结果进行排序和归并操作,MapReduce 通过归并和排序操作减少了中间结果传输的数据量,以此保证每一个 map 只产生一个中间数据文件,达到减少文件数目的目的。在 Spark 中,默认没有对中间数据进行预排序和归并操作,所以只能将不同分区的数据分别保存在单个文件中,分区个数即为中间文件数目。从表 8 可以看出,Spark-

CFKNN 的中间文件数目明显高于 MR-CFKNN,且分区数并不随着迭代次数的增加而增加,原因是 Spark-CFKNN 通过增加分区数目降低了每个分区所需的内存空间,减少了每个 task 的执行时间,但同时也造成了中间文件数目过多的问题,此外,Spark-CFKNN 通过设置 Spark 的环境变量及对 reparation 算子进行重分区操作,使得文件数目保持了恒定。

对于同步次数,MapReduce 为同步模型,即在所有的 map 操作结束后才能进行 reduce 操作。而 Spark 通过 RDD 间的宽依赖、窄依赖关系以及管道化操作(pipeline),提高了其并行化程度及 Spark 中

算法的局部性能。

算法的执行时间 T 会受到输入文件时间 T_{read} 、中间数据排序时间 T_{sort} 、中间数据传递时间 T_{trans} 和输出文件到 HDFS 时间 T_{write} 的影响。因为对每一轮的计算结果都进行了广播, 2 个算法的执行时间差异主要受到 MapReduce 和 Spark 运行机制及调度策略的影响, 所以最终只考虑 T_{sort} 和 T_{trans} 对 T 造成的影响。对于中间数据排序时间 T_{sort} , 由于 MapReduce 的 shuffle 过程会对中间结果进行排序和归并操作, 因此若假设每个 map 任务有 N 条数据, 每个 reduce 任务有 M 条数据, 则 MapReduce 的中间数据排序时间 $T_{\text{MR-sort}} = N \cdot \log_a N + R = O(N \cdot \log_a N)$ 。而在 Spark 中, 默认没有对中间数据进行预排序的操作, 因此, Spark 的中间数据排序时间 $T_{\text{Spark-sort}} = 0$ 。中间数据传递时间 T_{trans} 主要指将 map 任务运算的数据传送到 reduce 任务所消耗的时间, T_{trans} 主要由 map 任务输出的中间数据大小 $|D|$ 和网络传输速度 C 所决定。可以看出, 在网络传输速度相同的情况下, T_{trans} 与中间数据大小 $|D|$ 成正比。由此可知, 在相同的迭代次数下, 中间数据传递时间 T_{trans} 主要受同步次数的影响。由于 Spark 引入了管道化操作 (pipeline), 因此可以减少同步次数, 提高并行化程度。由表 8 中的同步次数可以看出, 随着迭代次数的增加, 相比 MapReduce, Spark 在中间数据传递时间 T_{trans} 上的优势越来越明显。

综上, 由于 MR-CFKNN 与 Spark-CFKNN 算法的程序设计不同, Spark-CFKNN 虽然因为增加了分区数而导致 Spark 的文件数目远多于 MapReduce, 但因为 2 个大数据框架的调度机制存在差异, Spark 通过引入管道化操作减少了同步次数, 使得中间数据传输时间随着迭代次数的增加而越来越优于 MapReduce。基于内存的 Spark 可以将算法运行过程中重复用到的中间数据结果缓存到内存中, 减少因为重复计算所消耗的时间, 最终使得 Spark-CFKNN 算法的运行效率优于 MR-CFKNN 算法。此外, 从表 9 ~ 表 12 可以看出, Skin Segmentation 和 Healthy Older People 数据集在算法运行时间上的差异小于 Gaussian1 和 Gaussian2 数据集, 导致该现象的原因是因为前两者的数据规模小于后两者。

分别使用原始数据集和本文算法迭代 5 次的所选择例集合做训练集, 使用 KNN 算法对分类精度进行测试, 结果如表 13 所示。从表 13 可以看出, 除了 Healthy Older People 数据集以外, 在其余 3 个数据集上本文算法都实现了精度提升, 表明该算法具有有效性。

表 13 原始数据集和本文算法所选择例集合的最优分类精度对比

Table 13 Comparison of optimal classification accuracy between the original dataset and the sample set selected by this algorithm

数据集	分类精度	
	原始数据集	本文算法所选择例集合
Gaussian1	0.987 71	0.989 85
Gaussian2	0.473 84	0.487 84
Skin Segmentation	0.994 66	0.996 24
Healthy Older People	0.820 00	0.800 00

4 结束语

本文对 CFKNN 算法进行改进, 提出一种在大数据环境下具有良好可扩展性的大规模压缩模糊 K-近邻算法。该算法在样例选择的过程中引入动态机制, 使得所选样例更具代表性。实验结果表明, 与原始 CFKNN 算法相比, 该算法具有更高的分类精度和加速比, 将基于 MapReduce 和 Spark 的 2 种大规模压缩模糊 K-近邻算法相比, 两者在样例选择个数和分类精度上相近, 但在文件数目、同步次数和运行时间上存在比较明显的差异。下一步将对动态选择的阈值是否为最优阈值以及样例初始化是否影响算法性能等问题进行深入研究。

参考文献

- [1] COVER T, HART P. Nearest neighbor pattern classification[J]. IEEE Transactions on Information Theory, 1967, 13(1): 21-27.
- [2] HART P E. The condensed nearest neighbor rule[J]. IEEE Transactions on Information Theory, 1968, 14(3): 515-516.
- [3] GATES G. The reduced nearest neighbor rule[J]. IEEE Transactions on Information Theory, 1972, 18(3): 431-433.
- [4] WILSON D, MARTINEZ T. Reduction techniques for instance-based learning algorithms[J]. Machine Learning, 2000, 38(3): 257-286.
- [5] BRIGHTON H, MELLISH C. Identifying competence-critical instances for instance-based learners[M]. Berlin, Germany: Springer, 2001.
- [6] HOU Guangdong, CUI Runpeng, PAN Zheng, et al. Tree-based compact hashing for approximate nearest neighbor search[J]. Neurocomputing, 2015, 166: 271-281.
- [7] LIANG Cong, XIA Shuyin, CHEN Zizhong. Improvement k-nearest neighbor classification algorithm based on reference points[J]. Computer Engineering, 2019, 45(2): 173-178. (in Chinese)
梁聪, 夏书银, 陈子忠. 基于参考点的改进 k 近邻分类算法[J]. 计算机工程, 2019, 45(2): 173-178.
- [8] WAN Ji, TANG Sheng, ZHANG Yongdong, et al. HDIdx: high-dimensional indexing for efficient approximate nearest neighbor search[J]. Neurocomputing, 2017, 237: 401-404.
- [9] WEN Qingfu, WANG Jianmin, ZHU Han, et al. Distributed hash learning method for approximate neighbor query[J].

- Chinese Journal of Computers, 2017, 40(1): 192-206. (in Chinese)
- 文庆福,王建民,朱晗,等. 面向近似近邻查询的分布式哈希学习方法[J]. 计算机学报, 2017, 40(1): 192-206.
- [10] ARNAIZ-GONZÁLEZ Á, DÍEZ-PASTOR J, RODRÍGUEZ J J, et al. Instance selection of linear complexity for big data[J]. Knowledge-Based Systems, 2016, 107: 83-95.
- [11] YANG Dingzhong, CHEN Xinhao. Approximate nearest neighbor search based on projection residual quantization hash[J]. Computer Engineering, 2015, 41(12): 161-165, 170. (in Chinese)
- 杨定中,陈心浩. 基于投影残差量化哈希的近似最近邻搜索[J]. 计算机工程, 2015, 41(12): 161-165, 170.
- [12] LUO Xin, OUYANG Yuanxin, XIONG Zhang, et al. Optimization of K-nearest neighbor based collaborative filtering algorithm by similarity support degree[J]. Chinese Journal of Computers, 2010, 33(8): 1437-1445. (in Chinese)
- 罗辛,欧阳元新,熊璋,等. 通过相似度支持度优化基于 K 近邻的协同过滤算法[J]. 计算机学报, 2010, 33(8): 1437-1445.
- [13] QIAO Yulong, PAN Zhengxiang, SUN Shenghe. An improved fast k-nearest neighbor classification algorithm[J]. Acta Electronica Sinica, 2005, 33(6): 1146-1149. (in Chinese)
- 乔玉龙,潘正祥,孙圣和. 一种改进的快速 k-近邻分类算法[J]. 电子学报, 2005, 33(6): 1146-1149.
- [14] ZHAI Junhai, LI Ta, WANG Xizhao. A cross-selection instance algorithm[J]. Journal of Intelligent and Fuzzy Systems, 2016, 30(2): 717-728.
- [15] SONG Yunsheng, LIANG Jiye, LU Jing, et al. An efficient instance selection algorithm for k nearest neighbor regression[J]. Neurocomputing, 2017, 251: 26-34.
- [16] MUJA M, LOWE D G. Scalable nearest neighbor algorithms for high dimensional data[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014, 36(11): 2227-2240.
- [17] ZHAI Junhai, WANG Xizhao, PANG Xiaohe. Voting-based instance selection from large data sets with MapReduce and random weight networks[J]. Information Sciences, 2016(367/368): 1066-1077.
- [18] MAILLO J, RAMÍREZ S, TRIGUERO I, et al. kNN-IS: an iterative Spark-based design of the k-nearest neighbors classifier for big data[J]. Knowledge-Based Systems, 2017, 117: 3-15.
- [19] SONG G, ROCHAS J, BEZE L E, et al. K nearest neighbour joins for big data on MapReduce: a theoretical and experimental analysis[J]. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(9): 2376-2392.
- [20] KELLER J M, GRAY M R, GIVENS J A. A fuzzy K-nearest neighbor algorithm[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1985, 15(4): 580-585.
- [21] ZHAI Junhai, LI Na, ZHAI Mengyao. The condensed fuzzy k-nearest neighbor rule based on sample fuzzy entropy[C]//Proceedings of 2011 International Conference on Machine Learning and Cybernetics. Washington D. C., USA: IEEE Press, 2011: 16-27.
- [22] APACHE. Hadoop[EB/OL]. [2019-07-20]. <http://hadoop.apache.org/>.
- [23] APACHE. Spark[EB/OL]. [2019-07-20]. <http://spark.apache.org/>.
- [24] WU Xindong, JI Shengqiao. Comparison of MapReduce and Spark in big data analysis[J]. Journal of Software, 2018, 29(6): 1770-1791. (in Chinese)
- 吴信东,嵇圣颢. MapReduce 与 Spark 用于大数据分析之比较[J]. 软件学报, 2018, 29(6): 1770-1791.

编辑 吴云芳