基于MapReduce和Spark的大数据模糊压缩近邻算法[[1]](#footnote-1)\*

王谟瀚1, 翟俊海1,2, 齐家兴1

1(河北大学 数学与信息科学学院, 河北 保定 071002)

2(河北大学 河北省机器学习与计算智能重点实验室, 河北 保定 071002)

通讯作者: 翟俊海, E-mail: mczjh@126.com

摘 要: 在我们之前的工作中，提出了针对模糊K-近邻的压缩近邻算法。本文对这一算法进行了改进，本文的贡献主要包括三个方面：(1)在样例选择阈值设置方面,引入了动态机制，根据当前迭代次数和总迭代次数动态的设置阈值，使得选择出的样例更具代表性；(2)将算法扩展到大数据环境，提出了基于MapReduce和Spark的大数据模糊压缩近邻算法；(3)对基于两种开源平台的大数据模糊压缩近邻算法进行了比较研究。实验结果证明，本文提出的算法是行之有效的。

关键词: 大数据；模糊K-近邻；压缩近邻；样例选择；动态机制

中图法分类号: TP311

Fuzzy Compression Nearest Neighbor Algorithm for Big Data Based on MapReduce and Spark

WANG Mo-Han1, ZHAI Jun-Hai1,2, QI Jia-Xing1

1(College of Mathematics and Information Science, Hebei University, Baoding 071002, China)

2(Hebei Key Laboratory of Machine Learning and Computational Intelligence, Hebei University, Baoding 071002, China)

**Abstract**: In our previous work, the condensed neighbor algorithm for fuzzy K-nearest neighbors was proposed. In this paper, the algorithm is improved. The contribution of this paper mainly includes three aspects: (1) In the aspect sample selection threshold setting, a dynamic mechanism is introduced, and the threshold is dynamically set according to the current iteration number and the total number of iterations, so that the selected examples are more representative. The example is more representative; (2) The algorithm is extended to the big data environment, and the big data condensed fuzzy K-nearest neighbors algorithm based on MapReduce and Spark is proposed. (3) The comparative study of the big data condensed fuzzy K-nearest neighbors algorithm based on two open-source platforms is made. The experimental results show that the proposed algorithm is effective.

**Key words**: big data; fuzzy K-nearest neighbors; condensed nearest neighbor; instance selection; dynamic mechanism

K-近邻（K-Nearest Neighbor，K-NN）[1]是一种常用的监督学习方法，广泛应用于模式识别、数据挖掘、机器学习领域。K-NN思想简单、易于编程实现，但是K-NN存在以下两个问题：

1. 对待分类样例进行分类时，需要对训练集中的所有样例进行存储，且要计算待分类样例与训练集中所有样例之间的距离；
2. 对于一个待分类样例，训练集中的样例对分类时的贡献是相同的，即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年提出的压缩近邻算法（CNN, Condensed Nearest Neighbor）。在CNN算法的基础上，研究人员提出了许多改进K-NN性能的样例选择算法，早期代表性的工作包括RNN（Reduced Nearest Neighbor）[3]、ENN（Edited Nearest Neighbor）[4]、ICF（Iterative Case Filtering）[5]等。在这一框架下，近几年研究人员也提出了一些比较好的解决方法。例如，Hou等人[6]将哈希技术与决策树结合起来，提出了基于树的紧哈希方法，可显著提高搜索近邻样例的效率。沿着哈希这一技术路线，Wan等人[7]基于谱哈希技术，提出了对高维数据进行近似近邻搜索的算法。基于分布式哈希技术，文庆福等人[8]提出了一种近似近邻搜索方法。Alvar等人[9]使用局部敏感哈希技术，提出了针对大规模数据集的样例选择算法，算法的时间复杂度达到了线性级。受交叉验证思想的启发，Zhai等人[10]提出了交叉样例选择算法，也可解决大规模样例选择问题；Song等人[11]将针对分类问题的样例选择问题扩展到回归场景，提出了一种针对K-NN回归问题的排序样例选择算法，扩充了样例选择的应用范围。近几年，大数据是非常火热的研究方向，一些科研人员研究了针对大数据的近邻搜索问题。基于开源大数据平台，Muja等人[12]提出了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[13]提出了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[14]提出了大数据K-近邻搜索算法。Song等人[15]对基于MapReduce的KNN算法进行了理论分析和实验证明，具有较高的参考价值。

针对第二个问题，Keller[16]于1985年提出的模糊K-NN算法。然而，模糊K-NN算法依然存在上述的问题(1)，针对这一问题，Zhai等人[17]提出了压缩模糊近邻算法(Condensed Fuzzy K-NN, CFKNN)。CFKNN是针对中小型数据集的算法，在大数据环境中，CFKNN会遇到计算效率低的问题，甚至会变得不可行。为了解决这一问题，本文提出了大数据模糊压缩近邻算法，使得CFKNN算法在大数据环境下变得可行，旨在利用并行计算框架的优势，达到在短时间内对海量数据进行样例选择的目的。

本文的主要贡献如下：

（1）将CFKNN算法推广到大数据环境，提出了大数据模糊压缩近邻算法，该算法大大降低了CFKNN算法的计算复杂度；

（2）将面向大数据环境的模糊压缩近邻算法在并行计算框架上进行了实现，缩短了算法的运行时间；

（3）针对单机环境下与并行计算框架下不同的运算机制，在阈值设置上提出了动态机制，对阈值进行动态调整，使得大数据模糊压缩近邻算法得以在并行计算框架实现。

# 基础知识

本节将对本文所涉及的基础知识进行简要介绍，包括CFKNN算法，Spark编程模型和MapReduce编程模型。

## CFKNN算法

设T是训练集，S是所选样例的集合，C为训练集的类别属性，共分为*p*类。初始时，从训练集T中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，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**

1. 输入:所选样例集合S.
2. 输出:样例的隶属度.
3. 对于,计算每一类的中心；
4. 对于,计算到各类中心的距离；
5. 对于,按式(1)计算；
6. 返回样例隶属度

(1)

**算法2. F-KNN算法**

1. 输入: 所选样例集,样例x.
2. 输出: 样例x隶属于每一类的隶属度.
3. 利用算法1计算S中每一个样例的类别隶属度，构成一个*n*行*p*列的矩阵*μ*；
4. 在S中找到*x*的*K*个近邻；
5. 利用公式(2)确定*x*的类别隶属度；

(2)

1. 输出

**算法3. CFKNN算法**

1. 输入:训练集T,参数k和(假设T的样本容量为n,T中的样例共分为p类).
2. 输出:.
3. 从T中每类随机选一个样例加入S
4. For x in T-S do
5. 根据算法1计算S中样例的类别隶属度
6. 用算法2确定x的类别隶属度
7. 根据式(3)计算x的熵
8. If
10. End if
11. End for
12. Return S

## MapReduce并行计算框架

MapReduce[18]是由Google公司提出的一种面向大规模数据处理的并行计算模型，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两个阶段，就可以完成对大规模数据的并行化处理工作。

Spark[19]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、可扩展等优势，迅速成为了Apache顶级项目。Spark起初是为了克服Hadoop并行计算框架的不足而提出的。发展到现在，Spark已经成为包含Spark SQL、Spark Streaming、Spark GraphX和Spark MLlib等子项目的生态系统。Spark将MapReduce基于磁盘的存储和容错改为基于内存的机制，提高了Spark的计算速度。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，把多个阶段的任务串联或并行执行，不再需要将不必要的中间结果输出到HDFS(Hadoop Distributed File System)上，以此提高Spark的计算效率。

RDD和算子是Spark的核心与基础。RDD是Spark中的基本数据抽象，它是不可变，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中，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允许用户显式的将数据缓存在内存中，极大的提高了计算速度。在对RDD中的数据进行操作时，需要通过Spark算子来实现相应的数据操作。Spark算子一般根据是否会触发Spark作业执行，将算子分为两类：

1. 转换算子，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，触发Spark执行作业，得到Spark作业的计算结果。

# 大数据模糊压缩近邻算法

本节首先介绍提出的大数据模糊压缩近邻算法，然后分别介绍基于Spark和MapReduce模糊压缩近邻算法的具体实现思路。

## 大数据模糊压缩近邻算法

通过对CFKNN算法进行分析可以发现，原始CFKNN算法所存在的问题以及无法在大数据环境下应用主要包括以下3个问题：

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*μ*，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，随着集合S中样例逐渐增多，对T中的每个样例寻找S中的*k*个近邻，并计算T中每个样例的熵值的过程，导致算法计算量非常巨大，使算法运行时间超出可接受范围；
2. 当数据集为大数据集时，寻找*k*近邻的计算复杂度大大增加；
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，是导致原始CFKNN算法无法在大数据环境下应用的主要原因。

针对以上原始CFKNN算法存在的问题，本文提出的大数据模糊压缩近邻算法对原始CFKNN做出如下改进：

1. 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，大大降低了由于计算S中所有样例的隶属度所带来的复杂度问题；
2. 通过并行计算框架，在每个计算节点上并行的在集合S中寻找样例*x*的*k*个近邻；
3. 对于问题(3)，本文在阈值设置上引入了动态机制，对阈值进行动态调整，将阈值设置为迭代次数*j*的单调递减函数，见式(3)。其中，对于initEntropy的设置应该考虑到对应类别数的最大熵。

通过对原始CKFNN算法进行上述改进，本文提出了基于并行计算框架的大数据模糊压缩近邻算法，算法的伪代码如算法4所示。算法的基本思想如图1所示。

(3)

其中，initEntropy为所设初始阈值，*j*为当前迭代次数，*n*为总迭代次数。

**算法4. 大数据模糊压缩近邻算法**

1. 输入:数据集T,近邻数k,阈值,迭代次数iterations.
2. 输出:数据子集S().
3. for iteration in iterations do
4. 初始化S, T；
5. 根据式(3)，并行计算T中每个样例*x*的信息熵entropy=；
6. if entropy>
7. ；
8. End if
9. 输出S
10. End for



Fig. 1 The basic idea of big data fuzzy compression neighbor algorithm

图 1 大数据模糊压缩近邻算法的基本思想

## 基于MapReduce的压缩模糊近邻算法

根据对CFKNN算法进行的分析，由于T为大数据集，我们发现对T中的每个样例，在S中寻找它的*k*个近邻以及计算T中每个样例的熵值的过程，会导致算法计算量非常巨大，所以我们将此过程通过MapReduce框架并行的执行，对于T中的样例，并行寻找S中k个近邻并计算熵值，从而大大减少了算法的运行时间。随着T中样例数量的增加，可以通过增加计算节点个数，使算法维持可接受的运算时间，因此算法变得非常容易扩展，具体流程如算法5所示。

算法5展示了大数据模糊压缩近邻算法在MapReduce中的计算流程。算法分为两部分，Mapper阶段和Reducer阶段。Mapper阶段包含setup和map两个方法。Reducer阶段只包含一个reduce方法。

在Mapper阶段的setup方法中，首先初始化随机选择或上一次迭代产生数据子集S(算法5第3行)，map方法计算输入的样例t∈T在S中的*k*个近邻(算法5第6行)，由*k*个近邻根据本文所提出的算法计算出t的熵值Entropy(算法5第7行)，若Entropy大于熵的阈值，则输出样例t。Reduce阶段不做任何操作直接将选择的样例输出。

算法5. MR-CFKNN算法

1. 输入:数据集T,近邻数k,阈值.
2. 输出:数据子集S().
3. Class Mapper
4. method setup()
5. load subDataSet S⊂T
6. Initialize k,initEntropy
7. method map(sid id, instance t)
8. Array kNearestNeighbor = findKNN(t, S, k)
9. Entropy = fknn(kNearestNeighbor, t)
10. If Entropy >
11. context.write(NullWritable, t)
12. End if
13. Mapper end
14. Class Reducer
15. method reduce(NullWritable, [])
16. for t in [] do
17. context.write(NullWritable, t)
18. End for
19. Reducer end

## 基于Spark的模糊压缩近邻算法

由于提出的大数据模糊压缩近邻算法为迭代算法，所以在MR-CFKNN的基础上，在Spark平台上对算法进行了实现。具体地，对于给定的大数据训练集T，以一轮迭代为例，首先随机从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(算法6第4行)；其次将T中的其余样例划分为若干子集，部署到*m*个云计算节点上(算法6第5行)，并将D作为广播变量在每个云计算节点上进行存储(算法6第7行)；然后使用map算子计算得到在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)，计算得到熵后，对大于阈值的样例进行筛选，将得到的*m*个所选样例子集合并，得到当前迭代所选样例集合；最后得到最终所选样例集合(算法6第14行至21行)，用S作为新的初始样例集合，将S作为新的广播变量部署在每个云计算节点上。重复算法6第6行至第22行过程，直到完成算法所设置的迭代次数iteration。

算法6. Spark-CFKNN算法

1. 数据集T,近邻数k,阈值,迭代次数iterations.
2. 输出:数据子集S().
3. val trainInitRDD = sc.textFile(T)
4. var dRDD = trainInitRDD.combineByKey().map().flatmap()
5. var tRDD = trainInitRDD.subtract(dRDD)
6. for(i<-0 until iteration) do
7. var dInsbroad = sc.broadcast(dRDD.collect())
8. val distanceRDD = tRDD.map(line => {
9. for (i <- 0 until dInsbroad.value.length) do
10. Distance(dInsbroad.value(i),line)
11. End for
12. })
13. val tEntropyAndSelectRDD= distanceRDD .map(line => {
14. memShipDevide(trainInsMemberShipCalc(kNearestNeighbor))
15. val entropy = calcEntropy()
16. If Entropy > α

19. End if
20. })
21. dRDD = dRDD.union(tEntropyAndSelectRDD)
22. tRDD = tRDD.subtract(tEntropyAndSelectRDD)
23. End for

# 实验结果与分析

为了验证本文算法的有效性，在4个大数据集上进行了实验，4个数据集的基本信息见表1。我们从文件数目、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和Spark-CFKNN进行了对比。此外，还对比了原始CFKNN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的精度。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250000个样例点，共500000个样例，且服从高斯分布，具体参数见表2。第二个人工数据集是一个3类二维数据集，每类包含200000个样本点，且服从概率分布：

在实验中，分别从4个大数据随机选取了部分样例作为测试集，测试集基本信息见表3。实验所用的云计算节点的配置信息见表4，云计算平台节点规划见表5。

**Table1 Basic information of data sets**

表1 数据集基本信息

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集 | 样例个数 | 属性个数 | 类别个数 |
| Gaussian1 | 500000 | 2 | 2 |
| Gaussian2 | 600000 | 2 | 3 |
| Healthy Older People | 75128 | 8 | 4 |
| Skin Segmentation | 240000 | 3 | 2 |

**Table 2 Gaussian distribution parameter**

表2 高斯分布参数

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 |  |  |
| 2 |  |  |

**Table 3 Basic information of test set**

表 3 测试集基本信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据集 | 类别1 | 类别2 | 类别3 | 类别4 | 样本容量 |
| Gaussian1 | 8816 | 9010 | 无 | 无 | 17826 |
| Gaussian2 | 4838 | 4747 | 4924 | 无 | 14509 |
| Healthy Older People | 25 | 25 | 25 | 25 | 100 |
| Skin Segmentation | 2527 | 2528 | 无 | 无 | 5057 |

**Table 4 Node configuration information**

表 4 节点配置信息

|  |  |
| --- | --- |
| 软硬件项目 | 配置情况 |
| CPU | Inter Xeon E5-4603 2.0Ghz（双核） |
| 内存 | 16GB RDIMM |
| 硬盘 | 1TB |
| 网卡 | Broadcom 5720 QP 1Gb 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 |
| 云计算平台 | Hadoop-2.7.1 Spark-2.3.1 |
| JDK版本 | JDK1.8 |

**Table 5 Cloud computing platform node planning**

表 5 云计算平台节点规划

|  |  |  |  |
| --- | --- | --- | --- |
| 节点号 | 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展示了原始CFKNN与本文提出的大数据模糊压缩近邻算法在Guassian1数据集进行实验的结果。如表中所示，由于算法的不同，CFKNN算法只对所有训练样例进行了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，是分别用两种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。导致大数据模糊压缩近邻算法的分类精度优于原始CFKNN算法的分类精度主要有两点。首先，CFKNN算法只对训练集进行1次迭代，为了保证所选出的样例更具代表性，同时考虑阈值为固定值，所以CFKNN算法需要将阈值设置为较为折中的数值，这会导致在算法运行初期，会选入较多的非边界样例；其次，大数据模糊压缩近邻算法考虑到了算法运行初期，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靠近真实值，所以引入了动态阈值策略，使阈值随着迭代次数的增加逐渐衰减，以此来克服原始CFKNN算法的缺点。以上两点使得大数据模糊压缩近邻算法的分类精度要优于原始CFKNN算法的分类精度。大数据模糊压缩近邻算法通过提高计算并行度，使得算法运行时间较原始单机环境下的CFKNN算法有极大的缩短，并通过计算加速比，衡量通过大数据模糊压缩近邻算法取得的并行化效果。该实验结果显示，本文所提出的算法在分类精度上优于CFKNN算法，说明本文所提出的算法所筛选出的样例更具代表性，且极大的缩短了算法的运行时间。综上所述，本文所提出的算法是行之有效的。

**Table 6 Comparison of Classification Accuracy of CFKNN,Spark-CFKNN and MR-CFKNN**

表 6 CFKNN,Spark-CFKNN,MR-CFKNN的分类精度比较

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | |  | | |
| 算法名称 | CFKNN | MR-CFKNN | | | Spark-CFKNN | | |
|  |  |  | | |  | | |
|  |  |  |  |  |  |  |  |
| 迭代次数 | 1 | 3 | 4 | 5 | 3 | 4 | 5 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| 分类精度 | 0.82856 | 0.92135 | 0.92179 | 0.92837 | 0.98996 | 0.98979 | 0.98979 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| 运行时间 | 10882 | 1588 | 2478 | 3374 | 3077 | 3679 | 5458 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| 加速比 | 无 | 6.85264 | 4.39144 | 3.22525 | 3.53656 | 2.95787 | 1.99377 |
|  |  |  |  |  |  |  |  |

根据吴信东[20]等人所做工作，本文对两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了对比，对比结果见表7及表8。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，与数据集无关，顾对该指标的对比不区分数据集。

对于文件数目的对比，主要指的是中间文件数目，因为算法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的增加。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，即分区个数即为中间文件数目。综上所述，结合MR-CFKNN和Spark-CFKNN算法的程序设计，在不同迭代次数下，两种算法所产生的中间文件数目如表7中文件数目栏所示。

对于同步次数，MapReduce为同步模型，即所有的map操作结束后，才能进行reduce操作。而在Spark中，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Spark中算法的局部性能。结合MR-CFKNN和Spark-CFKNN算法的程序设计，在不同迭代次数下，两种算法所产生的中间文件数目如表7中同步次数栏所示。

**Table 7 Comparison of the number of files and synchronizations under different iterations**

表 7 不同迭代次数下文件数目和同步次数的比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 文件数目 | | 同步次数 | |
| 迭代次数 |
|  | MapReduce | Spark | MapReduce | Spark |
| 3 | 14 | 84 | 6 | 5 |
| 4 | 42 | 252 | 8 | 7 |
| 5 | 112 | 672 | 10 | 9 |

关于算法的运行时间的分析中，算法的执行时间T会受到输入文件时间Tread、中间数据排序时间Tsort、中间数据传递时间Ttrans和写输出文件到HDFS时间Twrite的影响。由于对比的主要目的在于由MapReduce和Spark运行机制及调度策略所带来的运行时间的差异，所以只考虑Tsort和Ttrans对T造成的影响。

对于中间数据排序时间Tsort。由于MapReduce的shuffle过程会对中间结果进行排序和归并操作，所以若假设每个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TMR-sort=N\*logN+R=O(NlogN)。而在Spark中,默认没有对中间数据进行预排序的操作，所以Spark的中间数据排序时间TSpark-sort=0。

中间数据传递时间Ttrans主要是指将map任务运算的数据传送到reduce任务所消耗的时间，所以Ttrans主要由map任务输出的中间数据的大小|D|和网络传输速度Ct所决定。显而易见，在网络传输速度相同的情况下，Ttrans与中间数据大小|D|成正比。由此可知，在计算数据量相同的情况下，由于MapReduce的shuffle操作含有预排序和归并功能，所以MapReduce的中间数据量|D|MR要小于Spark的中间数据量|D|Spark。从而在计算数据量相同的情况下，MapReduce的中间数据传输时间TMR-trans要优于Spark的中间数据传输时间TSpark-trans。

综上所述，由于MR-CFKNN与Spark-CFKNN算法的程序设计不同，可以明显的观察到Spark的文件数目要大大多于MapReduce的文件数目，这就导致了虽然Spark的TSpark-sort为0，但TSpark-trans要远远高于TMR-trans，最终导致Spark-CFKNN算法的运行时间要高于MR-CFKNN算法的运行时间，两个算法具体的运行时间见表8。

**Table 8 Comparison of data sets under different iterations**

表 8 各数据集在不同迭代次数下的对比

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Gaussian1 | | | | | |
| 迭代次数 |
| 分类精度 | | 样例选择个数 | | 算法运行时间（单位:s） | |
| MapReduce | Spark | MapReduce | Spark | MapReduce | Spark |
| 3 | 0.92135 | 0.98996 | 76760 | 80955 | 1588 | 3077 |
| 4 | 0.92179 | 0.98979 | 77992 | 81190 | 2478 | 3679 |
| 5 | 0.92837 | 0.98979 | 78360 | 81239 | 3374 | 5458 |
|  | Gaussian2 | | | | | |
| 迭代次数 |
| 分类精度 | | 样例选择个数 | | 算法运行时间（单位:s） | |
| MapReduce | Spark | MapReduce | Spark | MapReduce | Spark |
| 3 | 0.40905 | 0.48784 | 233157 | 228527 | 998 | 2656 |
| 4 | 0.41546 | 0.48500 | 352164 | 259748 | 4088 | 6257 |
| 5 | 0.41450 | 0.48314 | 412549 | 278788 | 5747 | 8520 |
|  | Skin Segmentation | | | | | |
| 迭代次数 |
| 分类精度 | | 样例选择个数 | | 算法运行时间（单位:s） | |
| MapReduce | Spark | MapReduce | Spark | MapReduce | Spark |
| 3 | 0.98835 | 0.98714 | 13325 | 12443 | 186 | 125 |
| 4 | 0.9880 | 0.99031 | 14012 | 13970 | 314 | 252 |
| 5 | 0.98942 | 0.99624 | 14581 | 14194 | 628 | 444 |
|  | Healthy Older People | | | | | |
| 迭代次数 |
| 分类精度 | | 样例选择个数 | | 算法运行时间（单位:s） | |
| MapReduce | Spark | MapReduce | Spark | MapReduce | Spark |
| 3 | 0.68 | 0.64 | 17023 | 16476 | 112 | 88 |
| 4 | 0.83 | 0.79 | 53841 | 51117 | 197 | 154 |
| 5 | 0.80 | 0.80 | 59856 | 59232 | 411 | 300 |

表9为分别使用原始数据集和迭代5次的所选样例集合做训练集，使用KNN算法对分类精度进行测试的结果。如结果所示，除在Healthy Older People数据集外，在其余3个数据集上都实现了精度提升。表明本文提出的算法是行之有效的。

**Table 9 Comparison of classification accuracy between the original dataset and the selected set**

表 9 原始数据集和所选样例集分类精度对比

|  |  |  |
| --- | --- | --- |
| 数据集 | 原始数据集合 | 所选样例集合 |
| Gaussian1 | 0.98985 | 0.98996 |
| Gaussian2 | 0.47384 | 0.48784 |
| Skin Segmentation | 0.99466 | 0.99624 |
| Healthy Older People | 0.82000 | 0.80000 |

# 总结

本文对原始CFKNN算法进行了改进，基于此提出了大数据模糊压缩近邻算法，并对本文提出的算法在两个常用并行计算框架上进行了对比。从实验结果来看，本文的算法除在文件数目和算法运行时间上有较明显的差距外，在其它对比指标上并无太大差异。本文在真实数据集上进行了大量实验，与CFKNN进行了比较，且使用本文算法所选出的样例集合与原始数据集合进行了分类精度对比，并通过计算加速比，衡量了本文提出的算法所取得的并行化效果实验结果表明，本文提出的算法是行之有效的。

参考文献:

1. T. Cover, Hart P. Nearest neighbor pattern classification[J]. IEEE Trans. on Information Theory, 1967, 13(1):21-27.
2. Hart P E. The condensed nearest neighbor rule[J]. IEEE Trans. on Information Theory, 1968, 14(3):515-516.
3. Gates G. The reduced nearest neighbor rule[J]. IEEE Trans. on Systems Man & Cybernetics, 1972, 18(3):431-433.
4. Wilson D R, Martinez T R. Reduction techniques for instance-based learning algorithms[J]. Machine Learning, 2000, 38(3):257-286.
5. Henry Brighton C M. Advances In instance selection for instance-based learning algorithms[M]. Kluwer Academic Publishers, 2002.
6. Hou G, Cui R, Pan Z. Tree-based compact hashing for approximate nearest neighbor search[J]. Neurocomputing, 2015, 166:271-281.
7. Wan J, Tang S, Zhang Y. HDIDX: high-dimensional indexing for efficient approximate nearest neighbor search[J]. Neurocomputing, 2017, 237: 401-404.
8. Wen Q F, Wang J M, Zhu H. Distributed learning to hash for approximate nearest neighbor search[J]. Chinese Journal of Computers, 2017,40(1):192-206.
9. Arnaiz-González, Alvar, Díez-Pastor, José-Francisco, et al. Instance selection of linear complexity for big data[J]. Knowledge-Based Systems, 2016, 107:83-95.
10. Zhai J, Li T, Wang X. A cross-selection instance algorithm[J]. Journal of Intelligent & Fuzzy Systems, 2016, 30(2): 717-728.
11. Song Y, Liang J, Lu J. An efficient instance selection algorithm for k nearest neighbor regression[J]. Neurocomputing, 2017, 251: 26-34.
12. 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.
13. Zhai J H, Wang X Z, Pang X H. Voting-based instance selection from large data sets with MapReduce and random weight networks[J]. Information Sciences, 2016, 367: 1066-1077.
14. Maillo J, Ramírez S, Triguero I. kNN-IS: an iterative Spark-based design of the k-nearest neighbors classifier for big data[J]. Knowledge-Based Systems, 2017, 117: 3-15.
15. Song G, Rochas J, El Beze L. 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.
16. Keller J M, Gray M R, Givens J A. A fuzzy K-nearest neighbor algorithm[J]. IEEE Transactions on Systems Man & Cybernetics, 1985, SMC-15(4):580-585.
17. Zhai J H, Li N, Zhai M Y. The condensed fuzzy k-nearest neighbor rule based on sample fuzzy entropy[C]. In: Proc. of the Int’l Conf. on Machine Learning and Cybernetics. GuiLin: IEEE, 2011. 282-286.
18. Apache. Hadoop [Z/OL]. [2017-12-01]. http://hadoop.apache.org/.
19. Apache. Spark [Z/OL]. [2017-12-05]. http://spark.apache.org/.
20. Wu X D, Ji SW. Comparative study on MapReduce and Spark for big data analytics[J]. Journal of Software, 2008,29(6):1770-1791.

附中文参考文献:

[8] 文庆福, 王建民, 朱晗. 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.

[20] 吴信东, 嵇圣硙. MapReduce与Spark用于大数据分析之比较[J].软件学报,2008,29(6):1770-1791.

1. \*基金项目:河北省自然科学基金项目(F2017201026), 河北大学研究生创新资助项目(hbu2019ss077)资助。

   Foundation item: Natural Science Foundation of Hebei Province (F2017201026), Postgraduate's Innovation Fund Project of Hebei University (hbu2019ss077)

   收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos在线出版时间: 0000-00-00

   CNKI在线出版时间: 0000-00-00 [↑](#footnote-ref-1)