基于不规则区域划分方法的 k-Nearest Neighbor 查询算法①

张清清, 李长云, 李 旭, 周玲芳, 胡淑新, 邹豪杰

(湖南工业大学 计算机与通信学院, 株洲 412007)

摘 要: 随着越来越多的数据累积, 对数据处理能力和分析能力的要求也越来越高. 传统 k-Nearest Neighbor (kNN)查询算法由于其容易导致计算负载整体不均衡的规则区域划分方法及其单个进程或单台计算机运行环境的较低数据处理能力. 本文提出并详细介绍了一种基于不规则区域划分方法的改进型 kNN 查询算法, 并利用对大规模数据集进行分布式并行计算的模型 MapReduce 对该算法加以实现. 实验结果与分析表明, MapReduce 框架下基于不规则区域划分方法的 kNN 查询算法可以获得较高的数据处理效率, 并可以较好的支持大数据环境下数据的高效查询.

关键词: k-Nearest Neighbor (kNN)查询算法; 不规则区域划分方法; MapReduce; 大数据

Irregular Partitioning Method Based k-Nearest Neighbor Query Algorithm Using MapReduce

ZHANG Qing-Qing, LI Chang-Yun, LI Xu, ZHOU Ling-Fang, HU Shu-Xin, ZOU Hao-Jie

(College of Computer and Science, Hunan University of Technology, Zhuzhou 421007, China)

Abstract: With the constant accumulation of data, there is much higher desire for processing and analysis power to handle these data. Since the traditional k-Nearest Neighbor (kNN) query algorithm is easy to cause load imbalance on account of the regular partitioning method and its current platform is single process or single machine platform which cannot obtain high enough overall performance today, an irregular partitioning method based kNN algorithm is presented and being executed on the distributed parallel computing model which positioning to process large scale datasets in a distributed parallel way— MapReduce in this paper. Experimental results and analysis show that the irregular partitioning method based kNN algorithm can realize much significant operational efficiencies and support efficient query of big data much better.

Key words: k-Nearest Neighbor (kNN) query algorithm; irregular partitioning method; MapReduce; big data

1 引言

k 最近邻 k-Nearest Neighbor (kNN) 查询算法是一种简单且准确率较高的非参数分类方法, 该算法最初于 1967 年由 Cover 和 Hart 提出[1], 现已广泛应用于文本及图像处理等多领域之中, 并成为使用最为广泛的查询算法之一. 为更好地对该算法加以利用, 国内外许多学者对该算法做出了改进, 这些改进大致可以分为两类, 一类是独立 kNN 算法的改进, 即对 kNN 算法本身而不结合其它算法进行的改进, 另一类是集成 kNN 算法的改进, 即将该算法结合其它算法共同进行的改进. 在独立 kNN 算法的改进方面, Ahmad Basheer Hassanat 等提出了一种基于集成学习的 kNN 算法, 该算法将弱分类与加权求和规则相集成, 取得了较理想的结果[2], Jianping Gou 等提出了一种基于距离加权的改进 kNN 算法, 该算法使用对偶距离加权函数解决 k 值的上下文敏感问题[3]等等; 在集成 kNN 算法的改进

① 基金项目: 2013 年度国家科技部科技支撑计划(2013BAJ10B14-5);2014 年湖南工业大学自然科学研究项目(2014HZX19)
收稿时间:2015-01-13;收到修改稿时间:2015-03-12

方面, Bruno Trstenjak 等提出了一种将 kNN 算法与 TF-IDF 方法及文本过滤框架相结合的新型 kNN 查询算法[4], Yazdan Jamshidi 等提出了一种将引力搜索算法、间隔度量点阵与 kNN 算法三者结合在一起的一种混合型 GSALNkNN 算法[5]等等. 上述改进算法虽各有不同, 但是它们对待处理的文本均按照文本在二维坐标系(在此只讨论二维空间的情形)中的投影面积划分成面积完全相等的若干均等文本分片(每个网格内的文本即一个文本分片), 对每个分片分配相同数量的处理器, 这种划分方法即规则区域划分方法. 这种划分方法简单且易于实现, 然而, 这种算法在用于大规模数据处理的时候, 由于数据本身密集程度不同等多种的因素影响, 该算法极易引起计算负载的不均衡, 进而极大地影响算法的综合性能.

另一方面, 随着数据量的逐渐增大, kNN 查询算法也面临着处理效率需要越来越高的要求, 而传统的单个进程或单台机器的数据查询处理效率已经不能满足需求.

针对上述两个问题, 本文提出了一种基于不规则区域划分方法的 kNN 查询算法, 同时, 鉴于以分布的方式并行处理数据的大规模数据处理模型 MapReduce 已成为目前使用最为广泛的大数据处理模型, 本文使用该模型对基于不规则区域划分方法的 kNN 查询算法加以实现. 以不规则网格划分的方法进行目标文本区域划分可以避免由数据分布不均造成的数据倾斜问题[6], 而且, 用 MapReduce 模型运行此 kNN 算法可以有效提高大数据环境下 kNN 查询算法的数据处理效率.

本文内容将按照以下方式组织: 第二部分介绍论文相关工作, kNN 查询算法, 以及用于大规模数据处理的并行计算模型—MapReduce, 第三部分详细介绍了基于不规则区域划分方法的 kNN 查询算法, 并将该算法运行在 MapReduce 框架下, 第四部分介绍实验结果并进行结果分析, 第五部分进行总结.

2 相关工作

2.1 K 最近邻查询算法

kNN 查询算法的主要思想是根据一个训练样本集中 A 包含的 n 个已知的训练样本($T_1,T_2,\ldots,T_n$)的类别预测给定的目标文本 X 的类别, 并筛选出与给定文本最相似或者距离(本文采用的均是欧氏距离)最近的 k (k>=1) 个文本. 假定目标文本 X 的坐标值为($C_1,C_2,\ldots,C_n$), 任意一个训练文本 T 的坐标值为($W_1,W_2,\ldots,W_n$), 则 X 和 T 之间的距离可以由下式计算:

$$\cos(X,T)=\frac{\sum_{i=1}^{n}(C_i*W_i)}{(\sum_{i=1}^{n}C_i^2*\sum_{i=1}^{n}W_i^2)^{\frac{1}{2}}} \quad (1)$$

$\cos(X,T)$ 的值最大的那个训练文本为目标文本的“最近邻”, 若按上述余弦值的由小到大对训练文本进行排序, 则排在前 k 个的即为目标文本的 k“最近邻”.

2.2 MapReduce 模型

广泛应用于大数据处理领域[7,8]中的分布式并行计算模型 MapReduce 于 2004 年由 Google 提出[9], 该抽象模型十分简洁且易于使用, Hadoop[10]就是其应用非常广泛的一个开源实现. MapReduce 的运行基础即是 Hadoop 分布式文件系统 Hadoop Distributed File System 简称为 HDFS[11]. Map 函数和 Reduce 函数是 MapReduce 中包含的两个用户自定义函数. 其工作过程大致为: Map 函数接收 HDFS 传来的数据分片并将其处理成 key/value 对的形式并输入, 然后对 key/value 对进行处理, 产生一系列 key/value 对形式的中间结果, 这些中间结果经 shuffle 等处理后发送给 Reduce 函数; Reduce 函数接收由 Map 函数传送过来的 key/value 对并加以处理, 然后输出最终结果.

由于 MapReduce 这种将大片的数据分割成若干小片数据逐个处理的“分而治之”的思想, 可以使得编程人员可以仅考虑 Map 函数和 Reduce 函数即可实现大量数据的并行计算, 数据处理的复杂度也得到极大降低, 处理效率得到了大幅提高.

3 基于不规则区域划分方法的kNN查询算法

基于不规则区域划分方法的 kNN 查询算法的主要思想是: 假设待分类的样本是 t, 给定的数据集 D 是由欧式空间中的数据点构成的. 将数据集 D 投影到一个二维坐标系, 并假设该数据集在此坐标系中被均等的标记为 N_x*N_y 个网格(东西方向为 N_x 个, 南北方向为 N_y 个)、共有 M 个参与运算的处理器. 将这些处理器其分成 G 个组, 则 q 组中有 G_q 个处理器[12]:

$$G=[\sqrt{M\frac{N_x}{N_y}}] \quad (2)$$

$$G_q=\begin{cases}[\frac{M}{G}]+1 & q\le Mod(M,G)\\ [\frac{M}{G}] & q>Mod(M,G)\end{cases}\quad q\in[1,G] \tag{3}$$

用$(p.i,p.j)$表示区域 D 中的一个点 p, 将它的负载表示为 L_{ij}, 区域 D 的总负载表示为 L:

$$L_{ij}=\begin{cases}1,\sqrt{(i-i_0)^2+(j-j_0)^2}>10\\ 10,\sqrt{(i-i_0)^2+(j-j_0)^2}\le 10\end{cases}\quad i_0=N_x/2,j_0=N_y/2 \tag{4}$$

$$L=\sum_{i,j\in D}L_{ij} \tag{5}$$

设 $L_q=\sum_{n=1}^{q}\frac{L}{M}G_n\quad n\in[1,G-1]$ (6), 所有 L_{ij} 值中的最小值为 L_{min}, 按照下述方法将 D 划分成 G 个区域 D_q:

```
Begin
  q = 1, j = N_y
  while j ≥ 1
    i = 0
    while (i++) ≤ N_x
      mark (i, j) with q and calculate Σ_{i,j∈D} L_ij
      if |Σ_{i,j∈D} L_ij − L_q| = L_min
        q = q + 1
      else
        break
    j = j − 1
End
```

图 1　区域划分伪代码

图 1 描述了将区域 D 以该区域的负载为依据划分成了编号用 q 表示的 G 个区域的过程. 外层循环的循环变量 j 的值从 N_y 逐步递减到 *1*, 内层循环的循环变量 i 的值从 *1* 逐步递增到 N_x, 若某点处在规定区域内, 则计算出它的 $L=\sum_{i,j\in D}L_{ij}$ 并与 $L_q=\sum_{n=1}^{q}\frac{L}{M}G_n$ 进行比较, 只有在二者的差值等于 L_{min} 时, 区域 D 才可以被划分成 G 个区域.

4　基于不规则区域划分方法的kNN查询算法在MapReduce中的应用

当今的大数据形势对广泛使用的 kNN 查询算法提出了更优性能和更高效率的要求, 上述区域划分方法将整个文本区域划分为负载基本均衡的 G 个小区域并且进行了编号, 由于对每个区域文本的处理时间基本相同, 这种划分方法完全契合 HDFS 中 master 节点根据任务大小来选择 slaver 节点然后将作业交给 MapReduce 来执行的运行机制, 而且 MapReduce 目前已被广泛用于大数据的处理, 因此将基于不规则区域划分方法的 kNN 查询算法用于 MapReduce 框架下可以显著提高大数据环境下 kNN 查询算法的综合性能.

首先将 MapReduce 中用户自定义的 Map 函数中 getSplits()和 getRecordReader()方法根据上述伪代码中的描述进行改写, MapReduce 框架下基于不规则区域划分方法的 kNN 查询算法的执行过程如下:

Step 1: 用户上传文件至 HDFS;

Step 2: MapReduce 模型中的 map 任务从 HDFS 上读取编号为 q 的输入数据分片 D_q, 将 q 作为 Map 函数中输入 key/value 对的 key 值、D_q 作为其 key/value 对的 value 值, 输入 key/value 对的形式为*(q, D_q)*;

Step 3: split 函数读取该文件, 将其转换成特定格式的文件;

Step 4: 函数遍历该样本文件并根据公式(1)计算该样本与给定样本 t 的相似度值 s_q, 并建立一个包含 t 与 s_q 的集合 S, 其形式为 S(t,s_q);

Step 5: 将 q 作为 Map 函数中输出 key/value 对的 key 值、*S(t,s_q)*作为其 key/value 对的 value 值, 以*(q, S(t,s_q))*形式的输出 key/value 对输出中间结果;

Step 6: Map 函数的输出即 Reduce 函数的输入, Reduce 函数将中间结果读入之后对其按照具有相同 key 值的 value 值中 s_q 的大小进行一次升序排序, 并记录排在前 k 位的分片;

Step 7: 系统执行 shuffle 操作;

Step 8: Reduce 函数对所有结果 value 值中 s_q 的大小再进行一次升序排序并选择前 k 个分片;

Step 9: 将所有选择出的 k 个分片中 s_q 对应的 D_q 组成一个形为 *O(D_1,D_2,....,D_k)*集合后统计该集合中每一个 D 出现的次数, 出现次数最多的 D 的分类号(即下标号码)m 就是样本 t 的分类号;

Step 10: 将 m 作为 Reduce 函数中输出 key/value 对的 key 值、*S(t,s_m)*作为其 key/value 对的 value 值, 输出最终结果*(m, S(t,s_m))*.

该过程示意图如图 2.

5　实验

本小节描述了传统 kNN 算法和改进后的 kNN 算法分别在不同实验平台上使用不同数据集进行测试的实验结果, 并进行了比较分析.

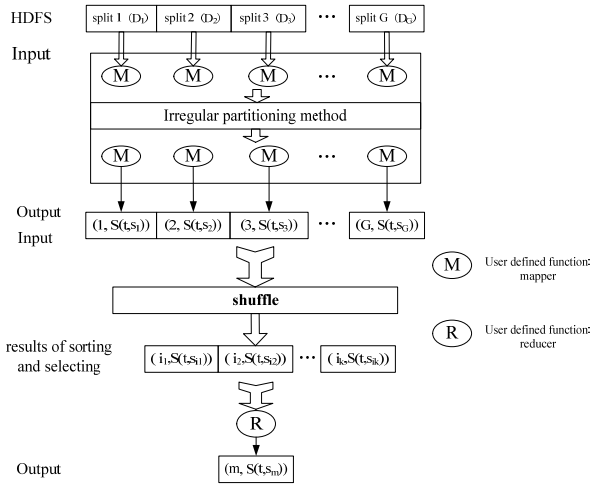

图2 MapReduce中上述改进kNN算法的执行过程

5.1 实验条件

本实验的实验环境为在三台配置均为 4.00GHz CPU、8GRAM、50G 硬盘的台式机上建立的虚拟机, Hadoop 集群为建立在该三台虚拟机上的 3 个节点—1 个 master 节点和 2 个 slaver 节点, 其中每个 slaver 节点可运行的 Mapper 和 Reducer 数目均设置为 4. 操作系统为 Cent OS Linux 6.5, Hadoop 版本为 Hadoop-0.20.1.

本实验采用来源于 UCI[13] 数据集中 Thyroid Disease 数据集的标准测试数据集 T_1 和人工合成的数据集 T_2 两种, 标准测试数据集 T_1 共包含 7200 个样本; 人工合成的数据集 T_2 是由 MATLAB 中的随机函数 $R = gamrnd(A,B,v)$ 产生的二分类数据.

5.2 实验结果与分析

首先, 本文使用标准数据集 T_1 在普通单机平台和 MapReduce 平台上分别对改进型 kNN 算法进行了实验, 以突出 MapReduce 框架下改进型 kNN 算法的优势. 为进一步强调该改进算法在提高数据处理速度方面的优势, 对传统 kNN 算法用同一数据集 T_1 在普通单机平台和 MapReduce 平台上也分别进行了实验, 实验结果如下图 3 所示.

实验结果显示, 运行在单机平台上的改进型 kNN 算法仅消耗了 18 分钟, 而传统 kNN 算法对数据集的 T1 处理却消耗了 21 分钟, 这说明了在相同平台上处理相同数据集时, 改进型的 kNN 算法的效率更高. 在 MapReduce 平台上, 改进型的 kNN 算法的运行效率仍然比较高, 无论节点数量较少还是较多. 例如, 当节点数目为 4 时, 改进型的 kNN 算法仅消耗了 16 分钟, 而传统的 kNN 算法消耗的时间却为 20 分钟. 上述实验结果表明, 在相同平台(无论单机还是分布式)上, 传统的 kNN 算法在处理相同数据时的效率都低于改进型的 kNN 算法.

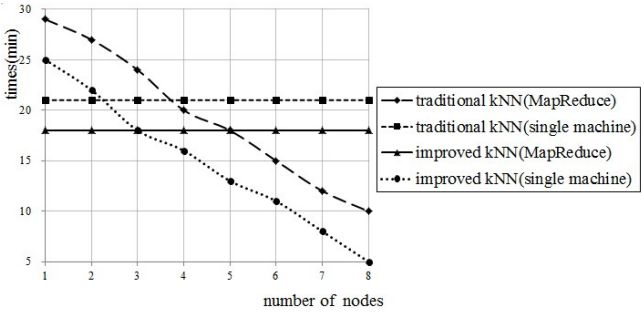

图 3 传统和改进型 kNN 查询算法在不同平台上的运行时间

其次, 衡量程序并行化性能和效率的重要指标为加速比, 此处加速比的计算方法为:

$$Speedup = \frac{\sum_{i,j \in D} L_{ij}}{Max\left(\sum_{i,j \in D} L_{ij}\right)} \tag{7}$$

本文在 MapReduce 框架下分别针对传统 kNN 算法和本文提出的改进型 kNN 算法在标准数据集 T_1 上进行了实验, 为增强实验的准确性及对比性, 两种算法都分别针对人工合成数据集 T_2 进行了实验, 实验结果如图 4 所示.

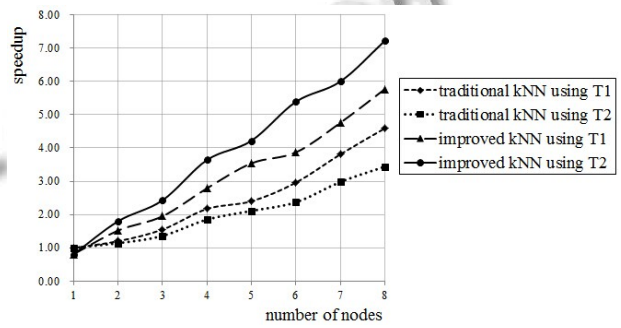

图 4 传统和改进型 kNN 查询算法在不同数据集上的加速比

该实验结果显示了加速比随节点数量变化而变化的情况, 可见, 无论在标准数据集还是在人工合成的数据集上, 传统 kNN 算法和改进型 kNN 算法的加速比均随着计算节点数目的增加而逐渐增大, 而且改进型 kNN 算法的加速比始终优于传统 kNN 算法. 例如, 当节点数为 6 的时候, 传统 kNN 算法和改进型 kNN 算法在数据集 T_2 和数据集 T_1 上的加速比分别为(2.35,2.95,3.

81,5.32), 由此可见, 在 MapReduce 框架下基于不规则区域划分方法的 kNN 算法的加速比高于传统 kNN 算法的加速比.

再次, 由于当 k 取不同值时, 查询结果的精确度也随之不同; 而且, 即使 k 值相同, 针对不同的数据集查询结果的精确度也是不一样的. 下图中表 1 展示了 MapReduce 框架下本实验所涉及改进型 kNN 算法以及传统 kNN 算法在取不同 k 值(k=1 和 k=3)时对数据集 T_1 查询所得结果精确度的百分比, 同时针对每一组 k 值实验对数据集 T_2 查询所得结果精确度也进行了比较, 以获得更为精确的实验结果.

表 1 传统和改进型 kNN 查询算法的精确度

precision of query (%)	traditional kNN using T_1	improved kNN using T_1	traditional kNN using T_2	improved kNN using T_2
k=1	89.6	91.7	91.3	92.9
k=3	77.8	79.5	83.2	87.3

当 k=3 时, 改进型 kNN 算法在处理数据集 T_1 和数据集 T_2 时获得的精确度分别为 79.5%和 87.3%, 而传统 kNN 算法处理此两个数据集后得到的精确度分别为 77.8%和 83.2%, 可见, 改进型 kNN 算法得到结果的精确度高于传统 kNN 算法, 当 k=1 时, 改进型 kNN 算法仍然具有这种优势. 以上数据表明, 无论 k 取哪个值, 改进型 kNN 算法查询结果的精确度均高于传统 kNN 算法.

最后, 在处理维度为 2、规模为 X 的数据 D 时, 传统 kNN 算法的运算复杂度为 $O(X^2)$, 而改进型 kNN 算法并行性明显增强, 由于其将规模为 X 的数据 D 划分成 G 个区域, 分布在 G 个处理机上, 故满足 $X=2^G$, 则算法的运算复杂度变为 $O(logX)$.

综上所述, MapReduce 框架下的基于不规则区域划分方法的 kNN 查询算法的查询效率、加速比及查询精确度均比传统 kNN 算法有所提高, 运算复杂度也比传统 kNN 查询算法要低, 这一事实表明 MapReduce 框架下基于不规则区域划分方法的 kNN 查询算法可以显著提高传统 kNN 查询算法的综合性能, 而且根据上述趋势可以发现, 数据量越大改进型算法的这种优越性将会愈加明显.

6 结语

随着累积的数据量逐渐增大, 越来越多的数据亟需快速、高效处理, 为提高数据处理效率, 本文针对空间数据库领域中基本而又重要的 k-Nearest Neighbor (kNN)查询算法所面临的高效处理大量数据的问题提出了 MapReduce 框架下的改进型 kNN 算法. 该改进型算法摒弃了传统 kNN 算法中基于规则区域划分文本的方法而采用了一种基于不规则区域划分文本的文本划分方法, 该方法不仅可以均衡计算负载, 而且由于该方法以数据密集程度为依据对数据进行划分而较好的避免了 MapReduce 面临的数据倾斜问题的出现, 从而极大的提高了 kNN 查询算法的性能. 然而, 对节点负载的计算需要进一步的研究, 文本集的映射研究需要向更高维度扩展, 基于不规则区域划分方法的 kNN 查询算法在 MapReduce 框架下的执行还需要面向更高的维度和更大量的数据.

参考文献

1 Cover T, Hart P. Nearest neighbor pattern classification. IEEE Trans. on Information Theory, 1967, 6(3): 21–27.

2 Hassanat AB, Abbadi MA, Altarawneh GA, et al. Solving the problem of the K parameter in the kNN classifier using an ensemble learning approach. Int'l Journal of Computer Science and Information Security, 2014, 8(12): 1802–1812.

3 Gou JP, Du L, Zhang YH, et al. A new distance-weighted k-nearest neighbor classifier. Journal of Information & Computational Science, 2012, 9 (6): 1429–1436.

4 Trstenjak B, Mikac S, Donko D. KNN with TF-IDF based framework for text categorization. Procedia Engineering, 2014, 69(3): 1356–1364.

5 Jamshidi Y, Kaburlasos VG. A GSA optimized, lattice computing kNN classifier. Engineering Applications of Artificial Intelligence, 2014, 10(35): 277–285.

6 Kolb L, Rahm E. Parallel entity resolution with dedoop. Datenbank-Spektrum, 2013, 13(1): 23–32.

7 Pandey S, Tokekar V. Prominence of MapReduce in big data processing. Communication Systems and Network Technologies (CSNT), 2014, 4: 555–560.

8 Fernández A, del Río S, López V, et al. Big data with cloud computing: an insight on the computing environment, MapReduce, and programming frameworks. Data Mining and Knowledge Discovery, 2014, 4(5): 385–409.

9 Lämmel R. Google's MapReduce programming model-revisited. Science of Computer Programming, 2007, 68(3): 1–30.

10 Hadoop.官方网站: http://hadoop.apache.org/.

11 White T. Hadoop 权威指南.周敏奇,王晓玲,金澈清等译.第 2 版.北京:清华大学出版社,2010.

12 金之雁,王鼎兴.一种适用于有限差分模式的负载平衡区域分解方法.气象学报,2002,60(2):150–156.

13 UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets.html.