

基于 Spark 框架的并行聚类算法

李淋淋¹,倪建成²,曹 博¹,于莘莘¹,姚彬修¹

(1. 曲阜师范大学 信息科学与工程学院,山东 日照 276826;

2. 曲阜师范大学 软件学院,山东 曲阜 273100)

摘 要:针对传统 K -means 算法在处理海量数据时存在距离计算瓶颈及因迭代计算次数增加导致内存不足的问题,提出了一种基于 Spark 框架的 SBTICK-means (Spark Based Triangle Inequality Canopy- K -means) 并行聚类算法。为了更好地解决 K 值选取的盲目性和随机性的问题,该算法利用 Canopy 进行预处理得到初始聚类中心点和 K 值;在 K -means 迭代计算过程中进一步利用距离三角不等式定理减少冗余计算、加快聚类速度,结合 Spark 框架实现算法的并行化,充分利用 Spark 的内存计算优势提高数据的处理速度,缩减算法的整体运行时间。实验结果表明,SBTICK-means 算法在保证准确率的同时大大提高了聚类效率,与传统的 K -means 算法、Canopy- K -means 算法和基于 MapReduce 框架下的该算法相比,在加速比、扩展比以及运行速率上都有一定的提高,从而更适合应用于海量数据的聚类研究。

关键词: K -means; Spark; 大数据; Hadoop; MapReduce

中图分类号: TP301.6

文献标识码: A

文章编号: 1673-629X(2017)05-0097-05

doi: 10.3969/j.issn.1673-629X.2017.05.021

Parallel Clustering Algorithm with Spark Framework

LI Lin-lin¹, NI Jian-cheng², CAO Bo¹, YU Ping-ping¹, YAO Bin-xiu¹

(1. College of Information Science and Engineering, Qufu Normal University, Rizhao 276826, China;

2. College of Software, Qufu Normal University, Qufu 273100, China)

Abstract: In view of the issues that when processing massive data, traditional K -means algorithm has the bottlenecks of distance computation and causes memory overflow due to increase of iterative calculation, the SBTICK-means (Spark Based Triangle Inequality Canopy- K -means) parallel clustering algorithm based on Spark framework has been proposed. In order to better solve the problem of blindness and randomness about K value's selection, initial cluster centers and K value have been preprocessed by Canopy. During K -means iterative calculation, redundant computation has been reduced and clustering speed has been accelerated by the triangle inequality theorem. Combined with Spark framework and made full use of memory computing advantages, the data processing speed has been improved and the overall running time of this algorithm has been decreased. Experimental results show that the proposed algorithm has improved clustering efficiency while ensuring the accuracy rate, and that at the same time, the size-up rate, scale-up rate and operating speed have been improved compared with the traditional K -means algorithm and Canopy- K -means and this algorithm based on MapReduce framework. Therefore it can be more suitable for clustering research of massive data.

Key words: K -means; Spark; big data; Hadoop; MapReduce

0 引言

K -means 聚类算法因其执行简单、快速、易于并行化,并可以提供直观结果等优点而成为数据挖掘和非监督式学习中的流行算法^[1],但它也存在以下问题^[2]: K 值(簇数)是人为确定的,在对数据不了解的情况下很难给出合理的 K 值;初始中心点的选择是随

机的,若选择到较为孤立的点,会严重影响聚类结果;算法在每次迭代时都需要进行大量的距离计算来确定新的聚类中心,然而其中许多计算都是冗余的;传统的串行 K -means 算法在处理大规模数据集时运行效率非常低。

针对以上问题,学者们不断地进行研究和优化。

收稿日期: 2016-06-29

修回日期: 2016-10-12

网络出版时间: 2017-03-07

基金项目: 国家自然科学基金(青年基金)(61402258); 山东省本科高校教学改革研究项目(2015M102); 校级教学改革研究项目(jg05021*)

作者简介: 李淋淋(1991-),女,硕士研究生,CCF 会员,研究方向为并行与分布式计算、数据挖掘;倪建成,博士,教授,CCF 会员,研究方向为分布式计算、机器学习。

网络出版地址: <http://cnki.net/kcms/detail/61.1450.TP.20170307.0922.096.html>

例如,张石磊等^[3]提出使用 Hadoop 平台下的 MapReduce 框架实现 K -means 的分布式聚类,提高了聚类速度;衣治安等^[4]提出在 MapReduce 框架的基础上利用 Canopy 算法进行优化,提高了聚类的准确率;王永贵等^[5]提出基于 MapReduce 的随机抽样 K -means 算法,通过对数据集进行多次随机采样,优化初始聚类中心,增强了聚类的稳定性;毛典辉^[6]提出了一种基于 MapReduce 的 Canopy- K -means 优化算法,采用“最大最小原则”对该算法进行改进,避免了 Canopy 算法选取的盲目性,进一步提高了聚类的准确率和“抗噪”的能力。

为了进一步减少数据聚类迭代过程中的冗余计算,加快聚类速度,提高聚类准确率,故综合以上聚类优化算法的各种优点,结合 Spark 计算框架,提出了 SBTICK-means 并行聚类算法。通过 Spark 框架的内存迭代计算优势^[7]提高了数据处理的速度,利用 Canopy 算法解决了 K -means 算法初始中心点的预选取和孤立点处理问题,利用距离三角不等式定理^[8-9]解决了聚类过程中距离冗余计算的问题。以 UCI 标准数据集进行测试,并与 MapReduce 框架^[10]下的结果进行比较、分析,以此验证 SBTICK-means 算法具有较好的聚类质量和较快的聚类速度,更适合应用于海量数据的聚类分析中。

1 相关工作

1.1 Spark 框架

Spark 是一套开源的、基于内存的,可以运行在分布式集群上的并行计算框架^[11-12],与 MapReduce 相比,有以下几点优势^[13]:

(1)中间结果不输出到磁盘上,而是使用有向无环图将各阶段的任务串联起来或者并行执行。

(2)使用 RDD 作为分布式索引来对数据进行分区和处理。

(3)MapReduce 在数据 Shuffle 时每次都花费大量时间来排序,而 Spark 任务在 Shuffle 中不是所有情况都需要排序。

1.2 基本概念

定义 1: Canopy 集合。给定数据集合 $D = \{d_i | i = 1, 2, \dots, n\}$, 对 $\forall x_i \in D$, 若满足 $P_c = \{x_i | \exists \|x_i - c_j\| \leq T_1, c_j \in D, i \neq j\}$, 则称 P_c 是以 c_j 为中心点、以 T_1 为半径的 Canopy 集合。

定义 2: Canopy 中心点。给定数据集合 $D = \{d_i | i = 1, 2, \dots, n\}$, 对 $\forall x_i \in D$, 若满足 $\{Q_m | \exists \|x_i - Q_m\| \leq T_2, T_2 < T_1, Q_m \in D, i \neq m\}$, 则称 Q_m 为 Canopy 候选中心点的数据点集合。

定义 3(准确率): $f = \frac{\sum_{t=1}^r a_t}{n} \times 100\%$ 。其中, n 为数据点总数; r 为分类个数; a_t 为聚到 t 簇的数据点数。

定义 4(加速比, Sizeup): $S = \frac{T_s}{T_p}$ 。其中, T_s 为单节点运行时间; T_p 为 p 个节点的运行时间。

定义 5(扩展比, Scaleup): $E = \frac{S}{p}$ 。其中, S 为加速比; P 为集群节点数目。

定理 1(三角不等式定理): 假设空间中任意三个数据点 x, y, z , 如果 $d(y, z) \geq 2d(x, y)$, 则 $d(x, z) \geq d(x, y)$ 。

推论 1: 由定理 1 可以得出, 对于数据点 x , 数据聚类中心点 Q_1, Q_2 , 如果 $d(Q_1, Q_2) \geq 2d(x, Q_1)$, 则数据点 x 被标记到 Q_1 所属的类中。

由推论 1 可知, 若 x_p 为任意一个未被聚类的数据点, $Q_m \in \text{List}$, 且 $\exists Q_n \in \text{List}(m \neq n)$, 使得 $S(c) = \min(d(Q_m, Q_n))$, 如果 $S(c) \geq 2d(x_p, Q_m)$, 就可以断定 Q_m 是距离数据点 x_p 最近的聚类中心, 因此就无需再计算其余聚类中心与 x_p 的距离, 这样就避免了有关 x_p 与其他所有聚类中心点的冗余距离计算, 大大减少了聚类过程中的迭代计算次数, 提高了聚类速度。

2 SBTICK-means 并行聚类算法

2.1 算法思想

该算法首先利用 Canopy 算法对 K -means 算法进行优化, 得到初始聚类中心点和 K 值, 然后在 K -means 聚类过程中引入三角不等式定理进行“精”聚类, 最后使用 Spark 框架实现该算法的并行化。因此, 基于 Spark 框架的 SBTICK-means 算法主要分为两个阶段, 其流程如图 1 所示。

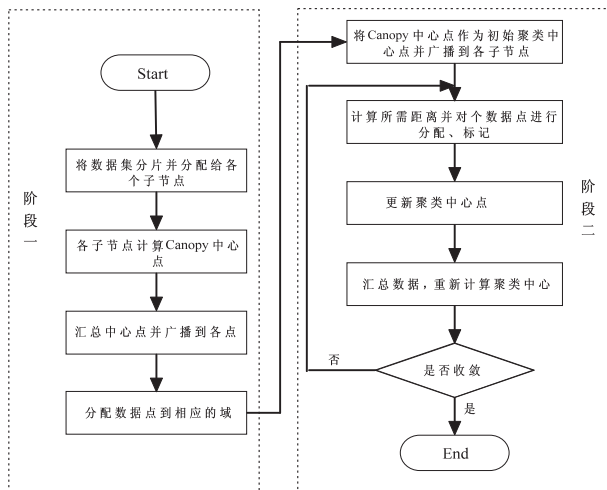


图 1 SBTICK-means 并行聚类算法流程
阶段一为 Canopy 算法在 Spark 框架中的执行

过程:

(1)从分布式文件系统 HDFS^[14-15] 上读取原始数据集 D 生成 RDD,并通过 map 操作将数据集格式化为向量。

(2)在每个子节点分别 map 计算数据点与 Canopy 中心点间的欧氏距离。

(3)根据 Canopy 算法规则对每个数据点进行标记,直到数据集 D 为空时结束,得出局部 Canopy 中心点。

(4)将各子节点得出的 Canopy 中心点通过 reduce 汇总得出全局的 Canopy 中心点。

阶段二为基于三角不等式定理的 K -means 聚类算法在 Spark 框架中的执行过程:

(1)将第一阶段计算得出的 Canopy 中心点作为初始聚类中心。

(2)启动各子节点的 map 任务,分别计算各数据点与聚类中心点的距离、各聚类中心点彼此间的距离。

(3)按照三角不等式定理对每个满足条件的数据点进行相应簇的划分,对不满足条件的数据点按照最短距离原则进行相应簇的划分。

(4)由 Combine 合并中间结果,并通过 Reduce 进行局部聚类中心点的更新和汇总。

(5)判断结果是否收敛,若收敛则算法结束,否则返回步骤(2)。

2.2 基于 Spark 框架的 SBTICK-means 算法并行化实现

2.2.1 Canopy 算法的并行化实现

由于 K -means 算法中的 K 值选取具有一定的盲目性和随机性,因此使用 Canopy 算法对其进行优化,以确定 K -means 的初始聚类中心点和 K 值。该算法可以高效地将数据划分为几个可重叠子集(即 canopy),从而避免聚类过程中局部最优的出现,有效提高了大规模数据的处理效率。该阶段主要实现 Canopy 算法的并行化,其具体实现的伪代码如下所示:

Input:待处理数据集 $D(d_1, d_2, \cdots, d_n)$, 参数 $T_1, T_2(T_1 > T_2)$, 采用交叉检验方式获得)

Output:Canopy 中心点集合 $Q(Q_1, Q_2, \cdots, Q_n)$

```
1)If(  $Q = \text{Null}$  )
2)任选一数据点作为 Canopy 中心点,并从  $D$  中删除
3)While(  $D \neq \text{Null}$  )
4)遍历  $D$ , 计算每个数据点到  $Q$  中点的距离
5)If(  $\text{dist}[i] < T_2$  ) 将该点加入  $Q$  中此点所属的 Canopy, 并从  $D$  中删除
6)Else If(  $\text{dist}[i] > T_1$  ) 将该点加入  $Q$  中, 并从  $D$  中删除
7)Else 将该点加入  $Q$  中此点所属的 Canopy
8)End If
9)End While
```

2.2.2 基于三角不等式定理的 K -means 算法并行化实现

该阶段主要实现 K -means 算法的并行化,利用预处理得到的 Canopy 中心点作为初始中心点完成“精”聚类。此外,为进一步加快聚类速度,引入三角不等式定理。由推论可知,通过三角不等式定理的判定,可以有效减少迭代过程中不必要的距离计算,将各数据点快速划入所属簇中,从而大大加快了聚类速度。其具体实现的伪代码如下所示:

Map 函数:

Input:Canopy 中心点集合 $\text{List}(c_1, c_2, \cdots, c_n)$, K 值, 数据集 $D(d_1, d_2, \cdots, d_n)$

Output:聚类中心点集合 W'

```
1) While(  $W' \neq \text{List}$  )
2) 计算 List 任意两点间的距离  $d(c, c')$ 
3) 将最短距离  $S(c) = \min(d(c, c'))$  保存到数组  $\text{Array}_1$ 
4) 依次计算  $D$  中每一个数据点到 List 中各点的距离  $\text{dist}[i]$ 
5) If  $2 \text{ dist}[i] \leq S$  记录此点属于第  $i$  个 Canopy 中心点的簇, 然后从  $D$  中删掉此点, 并对不满足条件的数据点, 将其到该中心点的距离保存到数组  $\text{Array}_2$ 
6) End If
7) If(  $D \neq \text{Null}$  )
8) 将上述不满足条件的数据点, 根据计算的点到中心点的距离, 将其划分给距离最小的簇心并进行标记
9) End If
10) 重新计算所有被标记点的新簇心  $W'$ 
11) If  $W = W'$  Break
12) Else 返回 2)
13) End While
```

Combine 函数:

Input: D 中数据点所对应下标 key, key 值所属 $\langle \text{key}, \text{value} \rangle$

Output: D 中数据点所对应下标 key, 每个簇中已被标记的所有数据点的各维累加值, key 值所属 $\langle \text{key}, \text{value} \rangle$

解析各维坐标值, 求出各维累加值和, 并保存到对应列表中

Reduce 函数:

Input: D 中数据点所对应下标 key, key 值所属 $\langle \text{key}, \text{value} \rangle$

Output: D 中数据点所属簇的下标 key, 最终簇心 W'

```
1) While(  $D. \text{hasNext}()$  )
2) 初始化  $\text{num} = 0$  记录所属簇内数据点的数目
3) 解析  $D. \text{next}()$  中的各维下标值, 计算  $\text{num}$ 
4) 计算各维下标值累加和并存储到对应列表中
5)  $\text{num}++$ 
6) End While
7) 汇总得出最终簇心  $W'$ , 并返回生成全局结果
```

在每一次 Reduce 执行完成后,比较新簇心与前一次的簇心,如果不发生变化(即收敛)则算法结束,否则继续执行上述过程直到收敛。最后集群上每个节点的数据点都被划分到对应的簇中,主节点汇总信息得

出最终聚类结果。

SBTICK-means 并行聚类算法在保持较高聚类准确率的前提下,不仅解决了聚类中心预选取和减少迭代冗余计算的问题,而且将其成功应用于 Spark 框架中,充分利用其内存计算优势,大大提高了聚类速度。与基于 MapReduce 的框架相比,该算法在处理海量数据时,运行时间更短,因此所提出的 SBTICK-means 算法更适合应用于大数据的聚类分析中。

3 实验

3.1 实验环境、测试数据集及评价指标

实验平台是在 Hadoop2.6.0 的 YARN 基础上部署 Spark 框架,在 Vcenter 中创建 6 台虚拟机,包含 1 个 Master 节点和 5 个 Slave 节点。操作系统版本均为 Ubuntu 14.04.3-Server-amd64,Hadoop 版本为 2.6.0,Spark 版本为 1.4.0,Java 开发包版本为 jdk1.7.0_79,程序开发工具为 Eclipse Mars.1 Release (4.5.1),算法使用 Java 实现。

实验测试数据集利用 UCI 数据集下的 Iris (数据对象 150,属性 4)、Wine (数据对象 178,属性 13)及 Balance Scale (数据对象 625,属性 4)来验证算法的有效性。同时将数据集 Poker-hand-testing (数据对象 1025010,属性 11)按照数据集的形式随机生成大小不同的 5 个数据样本, D_1 (100 万个样本点)、 D_2 (200 万个样本点)、 D_3 (500 万个样本点)、 D_4 (800 万个样本点)、 D_5 (1 000 万个样本点),来验证算法加速比和扩展比,并与 MapReduce 框架下的结果进行比较。

为了测试 SBTICK-means 算法的整体性能,采用以下几个评价指标:准确率、Sizeup、Scaleup 以及 MapReduce 框架下的运行时间比。

3.2 实验结果及分析

为了验证 SBTICK-means 算法的有效性,使用实验平台中的 3 个节点,利用 UCI 数据集,对 K -means 算法^[7]、Canopy- K -means 算法^[4]与 SBTICK-means 算法的有效性(准确率%)进行比较,结果如表 1 所示。

由表 1 结果可以看出,SBTICK-means 算法、Canopy- K -means 算法分别与 K -means 算法相比,在准确率上都有一定的提高,说明使用 Canopy 算法进行预处理可以有效提高聚类的准确率,同时也验证了 SBTICK-means 算法的聚类有效性。

另一方面,为了验证 SBTICK-means 算法的可扩展性,测试不同大小的数据集 D_1 、 D_2 、 D_3 、 D_4 、 D_5 在不同节点数下的加速比和扩展比,实验结果如图 2、图 3 所示。

由图 2、图 3 可知,该算法在处理少量数据时,随着节点数的增加,加速比呈线性增长。但当节点数增

加到 3 个后,加速比曲线趋向平缓或逐渐下降,扩展比也随之下降。这是因为当数据集的规模较小时,随着子节点数目的增加,集群运行时间、任务调度时间、数据通信时间的增加,降低了计算速度。而 D_5 数据样本的加速比基本呈线性增长,并在 6 个节点时达到最大值 5.6,扩展比变化趋势也较平缓,在 5 个节点时才稍有下降。这说明,样本规模越大,SBTICK-means 算法的处理效率越高,聚类效果越明显。

表 1 不同算法 20 次随机实验的测试结果平均值

数据集(.data)	算法类型	准确率/%
Iris	K -means	78.5
	Canopy- K -means	85.4
	SBTICK-means	86.3
Wine	K -means	93.1
	Canopy- K -means	94.5
	SBTICK-means	95.2
Balance Scale	K -means	81.9
	Canopy- K -means	85.1
	SBTICK-means	85.6

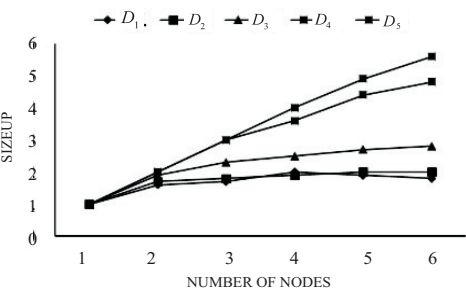


图 2 SBTICK-means 算法的加速比

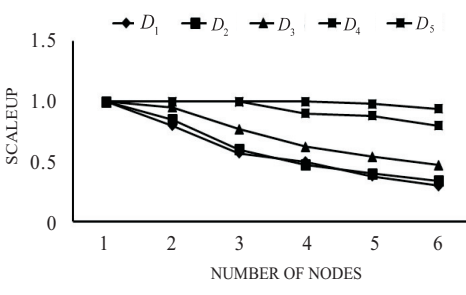


图 3 SBTICK-means 算法的扩展比

为进一步验证该算法的性能优势以及在 Spark 框架的计算优势,利用实验平台中的 6 个节点,测试在不同大小的数据样本下,SBTICK-means 算法与 Canopy- K -means 算法的平均运行时间(ms),结果如图 4 所示。

由图 4 可以看出,两种算法对不同大小的数据样本的执行时间是不同的,SBTICK-means 算法具有更短的运行时间,与 Canopy- K -means 算法相比平均可缩

短 18%, 并且数据样本越大, 优势越明显。原因是在迭代计算过程中引入三角不等式原理, 减少了迭代计算次数, 从而加快了聚类速度。

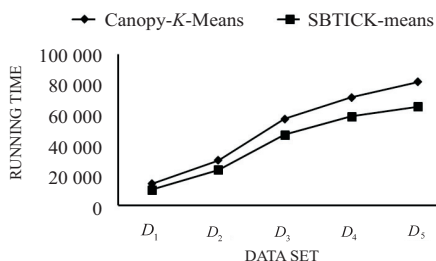


图4 两种算法运行时间的对比

另外还测试了在处理相同大小的数据样本 D_5 时, 该算法分别在 Spark 框架和 MapReduce 框架下的运行时间(ms), 结果如图5所示。

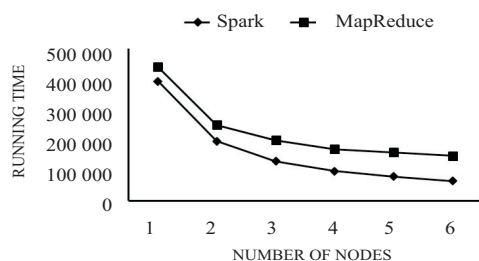


图5 两种框架运行时间的对比

由图5可以看出, Spark 的性能要优于 MapReduce, 在6个节点时运行效率最高可提高 1.261 倍。这是因为 Spark 使用 RDD 进行迭代计算的特点, 不需要将中间数据存在磁盘中, 从而大大减少了运行时间, 提高了聚类速度。

4 结束语

针对传统 K -means 算法在处理海量数据时存在距离计算瓶颈及因迭代计算次数增加导致内存不足的问题, 对传统 K -means 算法进行了改进, 将其与 Canopy 算法、三角形不等式定理相结合, 并应用于 Spark 框架中, 提出了一种优化的 SBTICK-means 算法, 提高了原算法的整体性能。实验结果表明, 该算法具有较高的聚类准确率和较快的聚类速度, 并且在加速比、扩展

比上都有所提高, 更适合应用于大数据的聚类研究中。

参考文献:

- [1] Jain A K. Data clustering: a review[J]. ACM Computing Surveys, 1999, 31(3): 264-323.
- [2] 孙吉贵, 刘杰, 赵连宇. 聚类算法研究[J]. 软件学报, 2008, 19(1): 48-61.
- [3] 张石磊, 武装. 一种基于 Hadoop 云计算平台的聚类算法优化的研究[J]. 计算机科学, 2012, 39(S2): 115-118.
- [4] 衣治安, 王月. 基于 MapReduce 的 K -means 并行算法及改进[J]. 计算机系统应用, 2015, 24(6): 188-192.
- [5] 王永贵, 武超, 戴伟, 等. 基于 MapReduce 的随机抽样 K -means 算法[J]. 计算机工程与应用, 2016, 52(8): 74-79.
- [6] 毛典辉. 基于 MapReduce 的 Canopy-Kmeans 改进算法[J]. 计算机工程与应用, 2012, 48(27): 22-26.
- [7] 梁彦. 基于分布式平台 Spark 和 YARN 的数据挖掘算法的并行化研究[D]. 广州: 中山大学, 2014.
- [8] 张顺龙, 库涛, 周浩. 针对多聚类中心大数据集的加速 K -means 聚类算法[J]. 计算机应用研究, 2016, 33(2): 413-416.
- [9] Elkan C. Using the triangle inequality to accelerate k -means[C]//ICML. [s.l.]: [s.n.], 2003: 147-153.
- [10] 孟海东, 任敬佩. 基于云计算平台的聚类算法[J]. 计算机工程与设计, 2015, 36(11): 2990-2994.
- [11] Meng X, Bradley J, Yuvaz B, et al. MLlib: machine learning in apache spark[J]. JMLR, 2016, 17(34): 1-7.
- [12] Armbrust M, Das T, Davidson A, et al. Scaling spark in the real world: performance and usability[J]. Proceedings of the VLDB Endowment, 2015, 8(12): 1840-1843.
- [13] Armbrust M, Xin R S, Lian C, et al. Spark sql: relational data processing in spark[C]//Proceedings of the 2015 ACM SIGMOD international conference on management of data. [s.l.]: ACM, 2015: 1383-1394.
- [14] 廖彬, 于炯, 张陶, 等. 基于分布式文件系统 HDFS 的节能算法[J]. 计算机学报, 2013, 36(5): 1047-1064.
- [15] Wu J, Hong B. Multicast-based replication for Hadoop HDFS[C]//IEEE/ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing. [s.l.]: IEEE, 2015: 1-6.

勘 误

我刊于 2017 年第 27 卷第 4 期刊载张明辉的论文《基于 FCM 的无检测器交叉口短时交通流量预测》因作者人数有误, 现更正为: 张明辉, 宗智嵩, 王夏黎。特此更正。