基于密度峰值的top-k空间文本查询

李艳红,涂锐
(中南民族大学 计算机科学学院,武汉 430074)

摘 要 由于普通的空间关键词查询通常导致查询结果过多,人们往往倾向于搜索结果集中且文本匹配度较高的地点.提出了一种基于密度峰值的空间文本查询问题,以获取空间对象密度集中且文本相似度较高的空间典型对象.利用TF-IDF结合Cosine相似度评估方法计算查询条件与其他空间关键词的相关度,再基于密度峰值聚类(DPC)算法,在满足空间文本条件的对象中,设计了TS-DPC算法将中间的结果集根据密度要求分为若干簇集,一方面可以获取给定范围内满足密度要求的空间对象簇;另一方面可以获取不同空间对象簇的中心,为研究所需.而后,对该算法进行了优化,提出了TS-DPC-IMP算法,在保持其他参数不变的情况下,通过网格算法,减少了该算法的运行时间.

关键词 空间数据库;聚类算法;密度峰值;密度聚类;cosine相似度
中图分类号 TP339 **文献标志码** A **文章编号** 1672-4321(2025)02-0260-09
doi:10.20056/j.cnki.ZNMDZK.20250216

Top-k spatial text query based on density peak

LI Yanhong, TU Rui
(College of Computer Science, South-Central Minzu University, Wuhan 430074, China)

Abstract Due to the fact that ordinary spatial keyword queries often result in too many query results, and people tend to search for locations with high text matching in the result set, a spatial text query problem based on density peak is proposed to obtain typical spatial objects with concentrated spatial object density and hightext similarity. TF-IDF similarity and Cosine similarity evaluation methods are used to calculate the correlation etween query conditions and other spatial keywords. Then, based on DPC (Clustering by Fast Search and Find of Density Peaks) algorithm, the TS-DPC algorithm is designed to divide the intermediate result set into several clusters according to density requirements among objects that meet spatial text conditions. On the one hand, it can obtain spatial object clusters that meet density requirements within a given range. On the other hand, it can obtain the centers of different spatial object clusters for research purposes. Then, the algorithm is optimized and the TS-DPC-IMP algorithm is proposed which greatly reduces the running time of the algorithm by using a grid algorithm while keeping other parameters unchanged.

Keywords spatial database; clustering algorithm; density peak; density clustering; cosine similarity

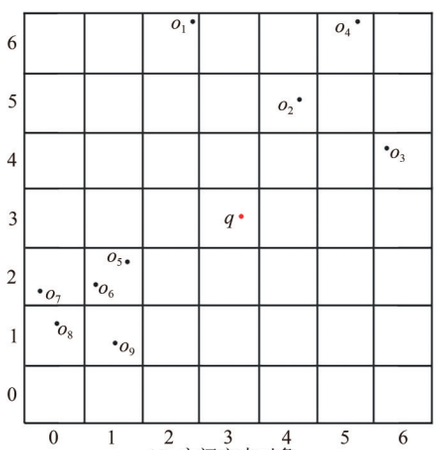
随着Internet的普遍应用以及智能手机和移动终端的普及,对于兴趣点(Point of Interests,POIs,比如:商场、饭店、旅游景点等)的空间关键词查询已成为当前空间数据库的研究热点[1].而以空间地理位置-文本数据为背景的空间关键词查询技术在多个领域得到了广泛的应用,比如:给定一个空间查询对象,包括待查询的空间地理位置(一般用经度和纬度表示)和待查询的一组关键词,返回距离查询位置以及空间文本相关性最高的前k个结果.但是由于现在每天会产生海量的空间数据,任意一个空间关键词的搜索可能会产生大量的查询结果,而且这样返回的空间对象均零散地分布在空间地理

收稿日期 2023-11-05
作者简介 李艳红(1973-),女,教授,博士,研究方向:时空数据处理、网络大数据、DB与AI融合,E-mail: liyanhong@mail.scuec.edu.cn
基金项目 湖北省自然科学基金资助项目(2017CFB135);中央高校基本科研业务费专项资金资助项目(CZY23019);网络创新及应用型人才课程实践教学研究资助项目(2019年第一批)

位置中，从而忽略了空间对象之间的相关性.

聚类算法可以根据数据之间的相似特征对数据集进行归类与划分，已经广泛应用于数据分析、地理位置分析等领域[2-3].根据给定的规则按照对象的属性将其划分成不同的簇集，相同簇集下的对象具有属性相关性，而不同的簇集中的对象则存在较弱的相关性.在聚类方法发展的进程中，出现了很多经典的算法，比如基于密度聚类[4-6](Density-Based)的典型代表算法有DBSCAN[7](Density-Based Spatial Clustering of Applications with Noise)和OPTICS[8](Ordering Points to Identify the Clustering Structure)，基于划分聚类的k-means[9]均值聚类算法以及基于模型聚类[10-11]的HMM算法，这些算法因为计算简单而被广泛使用，但是存在一定的局限性.

尽管SKQ[12](Spatial Keyword Query)技术可以从海量的空间文本数据中查询并返回搜索结果，但是这种搜索并不能捕获用户的真实需求，往往会返回大量的无效查询结果.可能存在一种极端情况，就是这k个对象之间的距离较大，结果是降低了用户对搜索结果的可选择性且增加了用户往返的路程代价.这些查询方法都忽略了对象之间的位置相关性，因为用户往往倾向于搜索结果较为集中的地点.以图1为例，右边为其具体的坐标位置，o_1~o_9为9个空间文本对象，均有地理位置坐标及其对应关键词集合.其中查询点q的位置为(3.6,3.5)，关键词为{cinema,food}，在查询点q处想吃点东西，然后去看电影，观察图1(a)中的对象，很明显，对象o_1的关键词与查询对象q的关键词匹配度最高，但是它们的距离较远.继续观察对象o_5到o_9，发现对象o_5和o_6除了coffee这个相同的关键词，它们另外的关键词组成的集合为{cinema,food}，可以知道对象集合{o_5,o_6}的关键词是满足查询点q的关键词，而且它们的距离比q到o_1的距离更小，另外对象o_7到o_9的关键词也与查询对象q的关键词存在交集，而且o_5到o_9对象的空间位置较为集中，这样也为用户提供了更多选择，距离代价也远小于q与对象o_1的距离代价，而且在对象o_1周围也缺少可供选择的对象.如果用户更看重品质，那么基于密度峰值的文本相似度top-k查询(即对象集合{o_5,o_6,o_7,o_8,o_9})将更符合用户的品质要求.



(a) 空间文本对象

o_1	(2.8,2.8)	cinema,food
o_2	(4.6,6.6)	cinema,game
o_3	(6.2,4.7)	food,game,hotel
o_4	(5.6,6.9)	cinema,food
o_5	(1.7,2.8)	coffee,food
o_6	(1.2,2.4)	coffee,cinema
o_7	(0.2,2.2)	food,ride,game
o_8	(0.5,1.6)	cinema,hotel
o_9	(1.5,1.4)	bread,food,water.

(b) 对象的坐标和文本信息

图1　空间文本查询示例

Fig. 1　Space text query example

基于密度峰值的聚类算法具有计算效率高、能检测非球形簇等优点[13-14].结合top-k[15]空间文本查询会考虑搜索结果密度最为集中的，并且其周围的文本相似度与查询对象关键词的文本相似度最高的前k个簇集以及每个簇集的中心点.

本文采用IR-Tree[16]来构建空间文本索引，它是一种混合索引模型，以倒排索引和R-Tree索引为基础.IR-Tree混合索引将两种索引结构有机结合在一起实现文本相似度和位置相关度的融合.因此，设计TS-DPC(Spatial Textual Similar-Based on Clustering by Fast Search and Find of Density Peaks)算法，该算法在经典的密度峰值聚类算法DPC(Clustering by Fast Search and Find of Density Peaks)的基础上，不仅考虑结果对象之间的距离相似度，而且考虑结果对象与其他对象的关键词集合与查询对象的关键词的相似度，这样在整个空间中，就会得到多个以密度峰值对象为中心的簇集.最后，再计算查询对象与得到的多个簇集中包含的对象的空间文本相似度，选择前k个结果.

1 相关技术介绍

DPC(Density Peak Clustering)是一种基于粒度计算的模型,该方法不仅能够快速识别聚类的数目,而且能够找到聚类的中心.它不仅在机器学习领域引起了强烈的反响,而且也吸引了其他领域的科研人员对它进行研究.该算法是基于两个假设:

(1)聚类中心被局部密度较低的近邻数据点包围;

(2)任意聚类中心与比它密度更高的数据点之间的距离都较远.

该算法与典型的密度聚类算法(DBSCAN)相比,只需要一个参数d_c,称为截断距离;而DBSCAN需要两个参数:Eps(代表以该点为圆心的Eps领域)和minPts(代表在Eps领域内需要包含的最小对象个数).

设有数据集$X=\{x_1,x_2,\cdots,x_N\}$,$i=1,2,\cdots,N$.对于空间任意的数据点x_i,DPC需要计算出每个数据点的局部密度ρ_i以及它与具有更高密度的最邻近点的距离δ_i,数据点x_i的局部密度ρ_i的计算公式定义为:

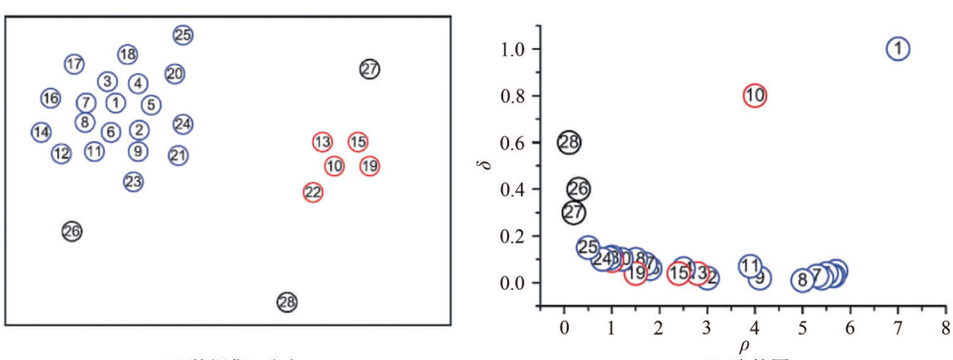
$$\rho_i=\sum_{i=1}^{n}\chi(d_{ij}-d_c), \tag{1}$$

其中:N为数据点的个数;$\chi(\cdot)$是一个逻辑判断函数,其表达的含义为:

$$\chi(\cdot)=\begin{cases}1, (\cdot)<0\\0, (\cdot)\geqslant 0\end{cases}. \tag{2}$$

$d_{ij}=\mathrm{dist}(x_i,x_j)$表示的是$x_i$与$x_j$这两个数据点之间的欧氏距离;截断距离$d_c$是本算法中需要传入的唯一的参数.可以看出,对于数据集中任意一点x_i的局部密度ρ_i等于x_i与数据集中不同于自身的任意点的欧氏距离小于d_c的点的个数,也即分布在x_i的d_c领域范围内的点的数目.

这样,对于数据集X中的每一个样本点,都存在其对应的ρ_i和δ_i,使用其构造数据集X的决策图(Decision Graph),如图2所示.很容易看到,编号为1和10的数据点的ρ值和δ值均较大,因此,可以确定其为数据集X的两个密度峰值的点.为了能更加准确方便地找出密度峰值的点,提出了一个综合考虑ρ和δ的度量γ,即$\gamma_i=\rho_i\delta_i$.



(a) 数据集X分布 (b) 决策图

图2 数据集X及其决策图

Fig. 2 Dataset X and its decision graph

1.1 空间对象之间距离相似度的度量

现有方法计算空间对象两点间的距离通常使用欧氏距离(Euclidean Distance),它是在m维空间中两点之间的真实距离,计算公式如下:

$$\mathrm{dist}(X,Y)=|X_i-Y_i|=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}. \tag{3}$$

那么两点间的空间相似度可以定义为:

$$\mathrm{SimDist}(X_i,Y_j)=1-\frac{\mathrm{dist}(X_i,Y_j)}{\max \mathrm{Dist}}, \tag{4}$$

其中,maxDist代表所有空间对象之间的最大距离,这是归一化后的结果.从公式(4)中可以看出,两个数据点之间的距离越大,那么它们的空间相似度越小.

1.2 空间对象与空间区域距离相似度的度量

假设空间区域为任意的形状,其中$R=\{o_1,o_2,\cdots,o_n\}$,那么对于查询点q与R的空间距离相似度定义为:

$$\mathrm{SimDist}(q,R)=1-\frac{1}{\max \mathrm{Dist}}\sum_{i=1}^{n}\mathrm{dist}(q,o_i).$$

可以看出q与R的相似度表述为q与目标区域包含的所有的点的欧氏距离的均值归一化后的值.

1.3 空间文本相似度的度量

定义1 文本相似度. 两个字符或者两个词语等的相似程度,相似度一般可用[0,1]之间的实数表示,该实数可通过语义距离计算获得.相似度与语义距离呈反比,语义距离越小,那么相似度越高,

通常使用下列公式表示相似度与语义距离的关系:

$$\mathrm{Sim}(S_A, S_B) = \frac{\alpha}{\mathrm{dist}(S_A + S_B) + \alpha}.$$

计算两个空间对象之间的文本相似度.首先,通过TF-IDF计算各关键词的权重,再通过Consin(余弦)相似度或Jaccard相似度计算两个对象之间的文本相度.TF-IDF算法是一种通过某个关键词在某文章中出现频次从而判断其重要程度的统计方法,这里主要包含两个概念:TF(Term Frequency,词频)和IDF(Inverse Document Frequency,逆文档词频).它的算法思想是先对TF进行统计,认为词语在某篇文章中出现的次数越多,则该词语与文档的正向关联性越大;其次,计算该词语的IDF,也就是该词语在所有的文章中出现的次数越多,那么IDF就越小,计算公式如下:

$$\mathrm{TF-IDF}_{i,j} = \mathrm{TF}_{i,j} \times \mathrm{IDF}_{i,j},$$

其中TF-IDF表示词频和逆文档词频的乘积,TF-IDF值越大,则表示词语对当前文本的重要性越大.

2　问题定义与分析

2.1　问题定义

定义 2(空间文本对象)　一个空间文本对象表示为o=(id, loc, k),其中o.id表示对象的唯一标识,o.loc表示对象的空间地理位置,用经纬度(lon, lat)表示,$o.k$表示对象的关键词,多个关键词表示为$o.k_1$={$o.k_1$, $o.k_2$, $o.k_3$,…, $o.k_n$}.在整个空间范围内所包含的空间文本对象集记为O,O={o_1, o_2, o_3,…, o_n}.

定义 3(空间文本查询对象)　用户查询表示为q=(loc, k, d_c, n),其中$q.loc$和$q.k$类似于空间文本对象的含义,分别表示查询q的空间位置信息以及待查询的关键词,$q.d_c$表示空间对象之间存在密度关联的最大距离,$q.n$表示查询检索对象的数量.

定义 4(密度峰值top-k空间文本查询)　给定一个对象集O,该查询检索所有$o\in O$且满足以下条件的点:

(1)γ={$\gamma_1,\gamma_2,\gamma_3,\cdots,\gamma_n$}且$\gamma_i>M$,其中$M$为满足密度峰值点的阈值;

(2)在满足该点为密度峰值点的前提下,获取该点d_c领域内关键词的合集,与查询关键词在空间文本相似度最高的前k个簇集.

此即为本研究需要查询的对象.

2.2　问题分析

定理 1　若存在一个空间文本查询对象Q,对空间任意不同于查询对象的两个空间文本对象X与Y,有:

$$\mathrm{CosinSim}(Q, \{X, Y\}) \geqslant \max\{\mathrm{CosinSim}(Q, X), \mathrm{Cosin}Sim(Q, Y)\}.$$

证明　设有两个空间文本对象X和Y,X={$x_1, x_2, x_3, \cdots, x_n | x_i \neq x_j$}和$Y$={$y_1, y_2, y_3, \cdots, y_n | y_i \neq y_j$},目标对象$Q$={$q_1, q_2, q_3, \cdots, q_n | q_i \neq q_j$},其中:$x_i$, y_i, q_i语句分词之后的词语的频度使用TF来表示,则有:

$$\mathrm{CosinSim}(Q, X) = \frac{\sum_{i=1}^{n}(x_i \times q_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2 \times \sum_{i=1}^{n}(q_i)^2}},$$

同理:

$$\mathrm{CosinSim}(Q, Y) = \frac{\sum_{i=1}^{n}(y_i \times q_i)}{\sqrt{\sum_{i=1}^{n}(y_i)^2 \times \sum_{i=1}^{n}(q_i)^2}}.$$

那么对于Z=X∪Y,其中Z={$z_1, z_2, z_3, \cdots, z_i$},有TF($x_i$)+TF($y_i$)=TF($z_i$),则:

$$\mathrm{CosinSim}(Q, Z) \geqslant \mathrm{CosinSim}(Q, X) + \mathrm{CosinSim}(Q, Y).$$

而 CosinSim (Q, X) +CosinSim (Q, Y) ≥max {CosinSim(Q,X), CosinSim(Q,Y)},所以 CosinSim (Q,{X,Y})≥max{CosinSim(Q,X),CosinSim(Q,Y)}.

证毕.

TS-DPC算法首先在DPC算法的基础上引入了数据点之间文本相似度这个概念,针对文本相似度的数据集X,修改了ρ和δ的计算方法.其中对于密度ρ的计算考虑了数据点之间的文本相似度,这里不仅需要考虑数据点与查询对象之间的文本相似度,还需要考虑数据点之间的文本相似度.比如,查询对象q的关键词={咖啡,瑞幸,电影},数据点o_1的关键词={咖啡,瑞幸},数据点o_2的关键词={电影},数据点o_3的关键词={咖啡,星巴克},那么通过余弦相似度计算查询对象与各数据点之间的文本相似度,$\mathrm{sim}_{\mathrm{text}}(q,o_1)$=0.67,$\mathrm{sim}_{\mathrm{text}}(q,o_2)$=0.33,$\mathrm{sim}_{\mathrm{text}}(q,o_3)$=0.33,但是$\mathrm{sim}_{\mathrm{text}}(q, \{o_1,o_2\})$=1,$\mathrm{sim}_{\mathrm{text}}(q, \{o_2,o_3\})$=0.33,$\mathrm{sim}_{\mathrm{text}}(q, \{o_1,o_3\})$=0.67,用户可能希望买一杯咖啡,然后去电影院,所以将这部分因素加入到DPC算法中,于是有:

$$\rho_i = \sum_{i=1}^{n}\chi(d(o_i,o_j) - d_c, \mathrm{sim}_{\mathrm{text}}(q, \{o_i,o_j\}) - \mathrm{sim}(q,o_i) - \alpha),$$

上式中,$d(o_i,o_j)$为空间对象o_i与o_j之间的欧氏距离,d_c为空间截断距离,用户可以根据经验或自身需要来定义其值;$\mathrm{sim}_{\mathrm{text}}(q,\{o_i,o_j\})$为查询对象$q$的关键词

与o_i和o_j关键词的集合的文本相似度,α为相似度调节系数,其范围为[0,1],表达的含义为空间对象o_i与查询q的关键词的相似度,在其截断距离内,加入了空间对象o_j后,其文本相似度的前后的差值在α邻域范围内,也可以称α为相似度距离.

同样,$\chi(x,y)$表示0~1函数,当x和y同时小于0时,$\chi(x,y)$=1,否则$\chi(x,y)$=0,即在文本相似度约束下的ρ_i表示同时小于空间邻域d_c和文本相似度邻域α的空间文本对象的个数.

其次,可以采用空间对象的相似度距离计算δ,两个对象的相似度距离α越大,越可以表示其空间文本与查询对象的相似程度,于是δ的计算方法如下:

$$\delta_i=\begin{cases}\min\{\alpha_{ij}\}, i\geqslant 2\\ \max\{\delta_j\}, i=1\end{cases}.$$

于是,可以根据ρ和δ的决策图,首先计算出密度最大的若干空间对象,这里称为核心点C,然后通过距离核心点的相似度距离可将其它的非核心点且非边界点分类到核心点,这样就形成若干的簇集.

引理1 若存在一个空间文本查询对象Q,对空间任意不同于查询对象的3个空间文本对象X,Y,Z,有:

$$\text{CosinSim}(Q,\{X,Y,Z\})\geqslant \max\{\text{CosinSim}(Q,X),\text{CosinSim}(Q,Y),\text{CosinSim}(Q,Z)\}.$$

证明 设有3个空间文本对象X,Y,Z,其中$X=\{x_1,x_2,x_3,\cdots,x_n|x_i\neq x_j\}$和$Y=\{y_1,y_2,y_3,\cdots,y_n|y_i\neq y_j\}$以及$Z=\{z_1,z_2,z_3,\cdots,z_n|z_i\neq z_j\}$,目标对象$Q=\{q_1,q_2,q_3,\cdots,q_n|q_i\neq q_j\}$,其中:$x_i$, y_i, z_i, q_i语句分词之后的词语的频度用TF来表示.

根据定理1有:$\text{CosinSim}(Q,\{X,Y,Z\})\geqslant\max\{\text{CosinSim}(Q,X),\text{CosinSim}(Q,\{Y,Z\})\}\geqslant\max\{\text{CosinSim}(Q,X),\max\{\text{CosinSim}(Q,Y),\text{CosinSim}(Q,Z)\}\}=\max\{\text{CosinSim}(Q,X),\text{CosinSim}(Q,Y),\text{CosinSim}(Q,Z)\}\}$.

证毕.

3 基于密度峰值的空间文本查询TS-DPC算法

假设q表示一个空间关键词的查询对象,空间文本对象的集合用O来表示,查询的结果表示为满足密度峰值要求的各簇集中密度最大的数据点的领域内,与查询点空间文本相似度最高的数据点以及其所在簇集的前k个结果.

本算法分为3个步骤,首先,构建基于空间文本倒排索引的IR-Tree,搜索出与查询对象相关的所有空间文本对象;其次,根据DPC算法,对上一步产生的结果集进行密度峰值聚类,将满足条件的对象分别归类,这样就产生若干的簇集,再找出每个簇集中密度最大的点(这里将该点定义为质心);最后,计算质心的ε领域内的空间文本对象的相似度以及质心到它们的距离的标准差,并根据总和评分进行排序,取前k个结果.

3.1 索引构建

传统的IR-Tree只是在R-Tree的基础上,根据各区域(MBR)所包含的对象构建倒排索引,也就是将R-Tree的每个非叶子节点和一个倒排文档相结合,如图3所示.

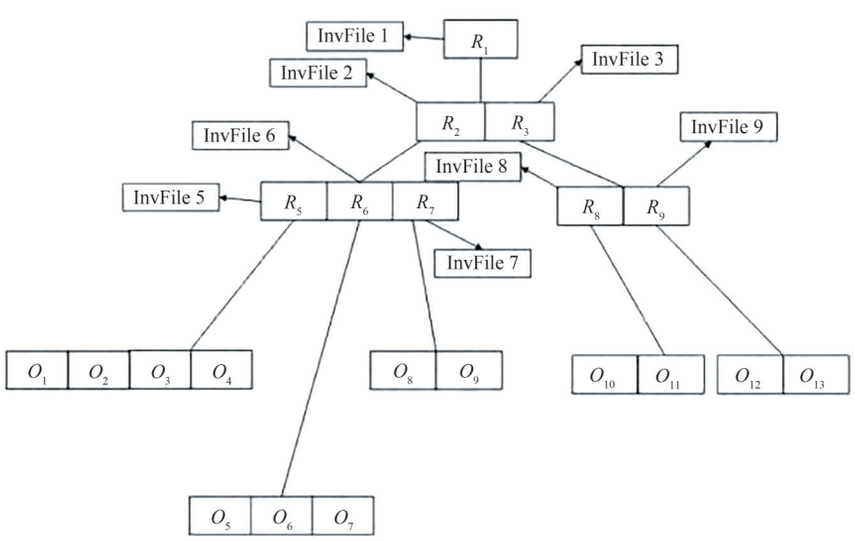


图3 基于空间文本的IR-Tree

Fig. 3 IR-Tree based on spatial text

给定空间查询对象q，在IR-Tree中查询出与其相关的对象，并且根据余弦相似度计算出查询对象q与空间文本对象o之间的文本相似度Similarity(q, o)，即$R=\{o_1, o_2, \cdots, o_j\}$.

3.2　根据DPC将空间对象分类并得出各类别密度峰值点

给定输入数据集R(即3.1中计算出的结果)，根据DPC算法，可以将R分为若干个簇集，并且得到每个簇集密度最大的点，定义聚类的结果为$C=\{C_1, C_2, \cdots, C_n\}$，其中$C_i=\{o_i \in R | <O, D_i>\}$.具体见算法1.

算法1　CalculateDensityPeak(R, q, d_c)

Input：数据集R，查询对象q，截断距离d_c，调节系数α

Output：任意数据点的局部密度ρ及其与之对应的最近邻对象的距离δ

```
1  ρ={},δ={},sim-dist={}
2  //根据数据集R计算任意两点的欧式距离d
3  for(i=0; i<R.length; i++)
4    for (j=0; j<R.length; j++)
5      if(i ! =j)
6      //获取两个对象之间的欧式距离
7      d_ij=euclidean_distance(o_i, o_j)
8        if(d_ij<d_c)
9          //计算查询对象到空间文本对象关键词的文本相似度
10           sim-dist_ij=cal_text_similarity(q.k, key_union(o_i.k, o_j.
k)) - cal_text_similarity(q.k, key_union(o_i.k)) - α
11         end if
12       end if
13   end for
14 end for
15 //获取数据集R中每个对象的局部密度
16 for(i=0; i<R.length; i ++)
17   if(d_ij<d_c&&sim-dist_ij>0)
18     ρ_i++
19   end if
20 end for
21 //对于任意的数据点o_i，计算出其最邻近对象的距离δ_i
22 //根据数据集R中任意点的局部密度ρ_i进行排序
23 ρ'_i =binarySort(ρ_i)
24 for(i=0; i<R.length; i ++)
25   if(ρ_i=max(ρ))
26     δ_i=max(d_ij)
27   end if
28 end for
29 return ρ, δ
```

算法1中第10行可以看到计算查询点与任意两个空间对象的文本相似度，也是本算法计算的关键所在.

3.3　找出以密度峰值点为中心的簇集

通过DPC算法计算出各数据点的局部密度ρ_i和距离δ_i来确定密度峰值点，当ρ_i和δ_i都较大时，γ_i的值会更大，中心点离非中心点的γ_i值会更远，在γ决策图中会出现点的跳跃.所以这里计算数据点的$\gamma_i=\rho_i\delta_i$，然后将$\{\gamma\}_{i=1}^{N}$进行降序排列，使用启发式方法可以在γ决策图中确定聚类中心个数.具体见算法2.

算法2　CalculateDecisionNum(ρ, δ)

Input：数据点局部密度ρ的集合以及其对应的δ

Output：数据点对应的决策值γ的集合

```
1  γ={}
2  for(i=0; i<ρ.length; i++)
3    for (j=0; j<δ.length; j++)
4      //计算每个空间点的决策值
5      γ_i=ρ_i·δ_i
6    end for
7  end for
8  //将集合γ_i进行排序
9  γ_i=sort(γ_i)
```

算法2第4行计算该点的决策值，然后对所有点的值进行倒序排列，这即是需要寻找的空间对象.

3.4　时间复杂度与空间复杂度分析

TS-DPC算法的时间复杂度由两部分组成：样本点的局部密度ρ和距离δ，其中ρ和δ的计算都涉及样本点之间的距离.在数据集规模为N的情况下，密度峰值聚类中心选择：(1)由算法1中的第7行，可以知道，计算任意两个样本点之间的距离，时间复杂度为$O(N^2)$；(2)计算N个样本点的局部密度ρ，时间复杂度为$O(N^2)$；(3)计算γ时间复杂度，同样为$O(N^2)$；(4)密度可达剩余点分配，时间复杂度为$O(m\cdot N)$，m为聚类中心数目，N为数据点个数.由于m是常数且通常较小，所以时间复杂度为$O(N)$.因此总的时间复杂度为$O(N^2)$.

空间复杂度：(1)距离矩阵D存储$O(N^2)$；(2)簇矩阵C复杂度$O(m\cdot N)\approx O(N)$.因此总的空间复杂度为$O(N^2)$.

4　改进的TS-DPC-IMP算法

上述的算法时间复杂度较高的地方主要在DPC计算密度峰值时，DPC算法的步骤1中每次查询都需要计算其中的任一数据点和其他所有数据点的距离，其时间复杂度为$O(N^2)$，对于空间大量的数据来说，这里的代价是比较大的.另外，算法在运

行时,所有的数据都在内存中进行计算,当数据量增大时,需要比较大的内存才能支持该算法,而且磁盘I/O消耗也大.另外通过对该算法步骤2的研究可以发现数据点的截断距离领域之外的点,并不加入其密度ρ的计算.于是,提出了改进的TS-DPC-IMP算法(Spatial Textual Similar-Based on Clustering by Fast Search and Find of Density Peaks-Improvment),通过网格划分的方法来对上述计算进行剪枝,优化区域查询的操作,以减少时间的开销.

4.1 相关概念

定义5(网格单元) 用户需要确定一个空间网格划分的参数e,根据这个参数,将数据空间在每一行上划分为N个相同长度的区间,每一列上做相同的划分,从而将整个数据空间划分为边长为e的网格.设网格单元Grid=$\{g_1,g_2,\cdots,g_n\}$,$g_i=\{1\leqslant i,j\leqslant n\mid[\mathrm{col}_{ij},\mathrm{row}_{ij}]\}$,其中$i$代表行,$j$代表列.在网格单元划分的过程中,对于每行或者每列的末尾的网格单元,可以称之为边缘网格单元.可能长度不足e,但是仍可以将该边长看作e,因为对于边缘网格单元来说,不会存在其相同方向延伸的点,这样既可以保证计算的准确性,又可以保证算法的一致性.

定义6(邻近网格单元) 对于任意网格单元Grid,若符合以下条件则认为其为g_i的邻近网格单元,(a)两个网格单元存在公共的边edge;(b)两个网格单元存在公共的顶点,也即以自身为中心的九宫格包含的网格单元集,记作N-Grid.

4.2 算法描述

4.2.1 优化数据点密度ρ_i的计算

为了更加方便地计算存在于网格单元中任意数据点的局部密度,这里取网格单元的边长e与截断距离d_c的关系为$e=d_c$.如图4所示,从上至下、从左至右依次给网格单元Grid的邻近网格编号,即N-Grid=$\{g_1,g_2,\cdots,g_7\}$.根据DPC局部密度的ρ_i的计算方法,对于Grid中的任意点,均需要计算其与N-Grid中任意点的欧式距离,从而可以大幅减少点与点之间欧式距离的计算.

如图4所示,$g_{43}=\{o_6,o_9\}$,$g_{34}=\{o_4\}$,$g_{44}=\{o_7,o_8\}$,$g_{35}=\{o_2\}$,$g_{45}=\{o_3\}$,$g_{66}=\{o_5\}$,$g_{57}=\{o_1\}$,其中红色虚线的圆为以o_3为圆心、d_c为半径的圆.根据定义5可以知道,计算o_3的局部密度,只需要计算与其相邻的网格中的空间文本对象的欧式距离,再与d_c进行比较,即可算出ρ_3的值.特别的,对于网格边上的点,也可以简化计算,例如:o_6和o_9均在g_{43}这个单元格内且处于对边,就不需要计算它们之间的距离.

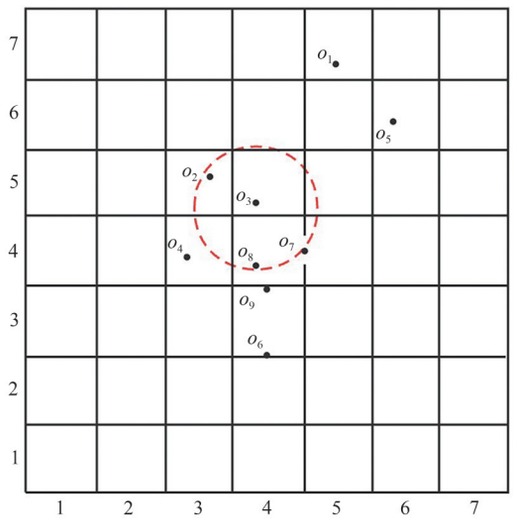


图4 基于网格计算对象的截断距离

Fig. 4 Truncation distance of objects based on grid computing

4.2.2 时间复杂度与空间复杂度分析

TS-DPC算法的时间复杂度定性描述了该算法的运行时间.很明显,使用网格可以减少对任意两点的欧氏距离的计算以及局部密度ρ的计算,时间复杂度为$O(N\log N)$,但是增加了空间复杂度,因为增加了存储网格单元所消耗的空间,空间复杂度为$O(N^2)$.

5 实验结果与讨论

在本节中,将展示使用随机生成的数据来测试数据集总量R、截断距离d_c以及k值变化对算法性能的影响,同时也会将本算法与经典的k-means、DBSCAN、DPC算法、改进的TS-DPC算法进行比较.

本实验在一台装有Intel(R) Core(TM) i7-10750H, 2.6 GHz CPU,64 GB RAM,Window 10的计算机上完成,算法在JDK-11.0.15.1的JAVA环境下运行.

5.1 实验环境、数据集与评价指标

为验证TS-DPC在凸数据集与非凸数据集上都有较好的聚类性能,将TS-DPC与DPC、DBSCAN、k-means等经典聚类算法在人工数据集和真实数据集上进行比较,并使用聚类准确率(Acc)衡量聚类,其公式为:

$$\mathrm{Acc}=\sum_{i=1}^{n}\frac{\delta(y_i,\mathrm{map}(z_i))}{n},\tag{5}$$

$$\delta(x,y)=\begin{cases}1, & x=y\\ 0, & \text{otherwise}\end{cases},\tag{6}$$

式中:map(x)是一个映射函数,以真实标签y_i作为参考标签,用来解决标签不一致问题;N是簇的个数;n表示数据集中的点数;z_i为聚类结果标签.

数据集见表1、表2.此外，为避免其它因素影响，这里使用了人工数据集和真实数据集，人工数据集中的样本点可以多样化，真实数据集是经典的数据集，可以让空间对象更有代表性.实验中距离使用欧式距离.

表1　人工数据集

Tab. 1　Synthetic datasets

DataSets	Samples	Attributes	Clusters
Aggregation	788	2	7
Flame	240	2	2
Two-moon	1502	2	2
Two-circle	1200	2	2

表2　真实数据集

Tab. 2　Real dataset

DataSets	Samples	Attributes	Clusters
Iris	150	4	3
Ecoli	336	8	8
Wine	178	13	3
WDBC	569	30	2

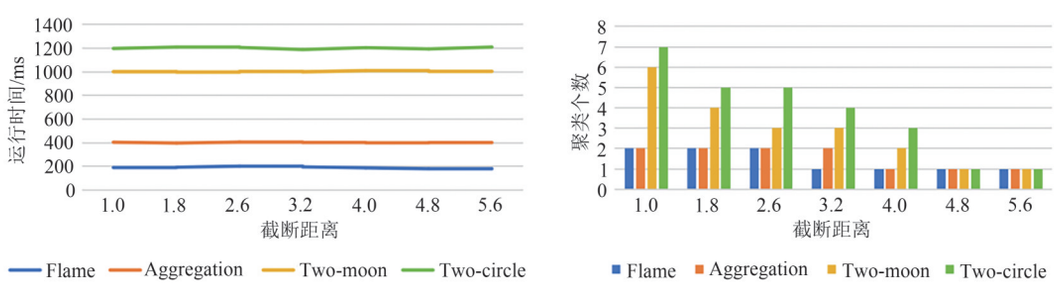
5.2　截断距离d_c变化对测试结果的影响

由上可知，截断距离d_c对于算法的准确率影响较大，为了能更准确地测试该参数对算法的影响，按照d_c根据同等Step=2(初始为1)逐步递增的方式来计算聚类中心，并且结合查询点与空间对象的空间文本相似性，来计算获取的簇集数，测试结果见图5.

由图5(a)可见：对于相同的截断距离，聚合的耗时随着数据集对象数量的增多而增加，而不同的截断距离对聚合的耗时影响不大；由图5(b)可见：聚合的簇集数在正常范围内波动，但随着截断距离的不断增加，集合的簇集数会减少，因为此时更多的数据点会合并.

5.3　查询关键词数量对测试结果的影响

用户在查找自己感兴趣的点时，可能会输入一个或者多个关键词，因为这样会更好地匹配到自己需要的结果，但同时也会使查询到的结果更少，甚至可能会将自己期望的结果排除掉，本节会测试$q.k$的数量为1,2,3,4,5时的准确率.由图6可见：随着查询关键词数量的增加，查询出的对象的个数会逐渐减少，这也和本研究预期的一致，但是随着关键词数量进一步减少，最终会无法返回结果.所以本文在查询时，需要优化客户传入的关键词，这样才可以得出较优的结果.



(a) 不同数据集在不同截断距离中聚合的耗时　　(b) 不同数据集在不同截断距离中聚类的个数

图5　不同截断距离d_c对测试结果的影响

Fig. 5　Influence of different truncation distance d_c on test results

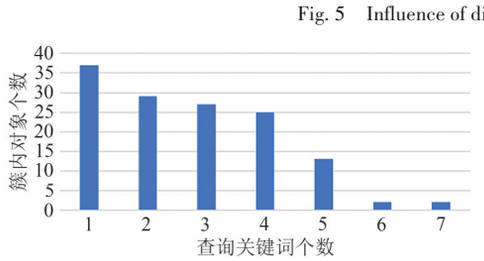


图6　查询关键词数量对查询簇集内对象数据的影响

Fig. 6　Influence of the number of query keywords on querying object data within a cluster

5.4　改进算法与原算法效率的对比

TS-DPC-IMP改进算法主要在计算欧式距离时进行了剪枝，减少了很多不必要的计算，所以在一定程度上提高了效率，另外，增加了经典的k-means、DBSCAN、DPC算法与之比较，图7展示了不同的算法在真实数据集下的运行时间.

从图7中可以看出，对于相同的数据集，在不同的算法中执行时，TS-DPC-IMP算法的效率明显要优于其它算法.对比TS-DPC算法，TS-DPC-IMP算法的运行时间要少，这与其中的网格算法来减少两点间的距离计算是明显相关的.

显而易见，算法的耗时是随着空间对象的增加而增加的，呈上升趋势.但是就不同的算法相对于不同的数据集而言，仍然是TS-DPC-IMP算法在效率上要优于其他算法.

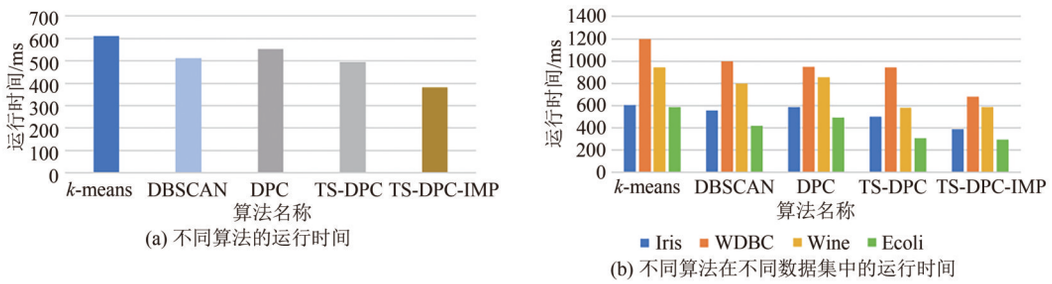


图7 不同算法的运行时间

Fig. 7 Running time of different algorithms

6 结语

针对DPC由于分配策略导致在非凸集上聚类效果不佳的问题,提出了一种基于密度峰值聚类的算法TS-DPC.实验表明:该算法在凸数据集与非凸数据集上均表现良好,能够处理离群点以及簇边缘扩散等问题,较*k*-means、DBSCAN、DPC聚类算法更加稳定和灵活,因为该算法优先考虑密度峰值处的对象的分布情况,再选出符合用户搜索条件的对象.另外,本文还提出了改进的TS-DPC-IMP算法,不仅大大减少了计算时间,而且在准确率上也有较大的提升.但是该算法未考虑数据对象较大时的解决方案,这样对用户感受的影响较大,所以未来会在多机上以分布式集群的方式来运行算法,尽可能地给用户提供准实时的结果返回.

参 考 文 献

[1] LI F, ZHOU M, LI S, et al. A new density peak clustering algorithm based on cluster fusion strategy[J]. IEEE Access,2022, 10: 98034-98047.

[2] DUAN B, WEI P, MA W. An improved density peak clustering algorithm [C]//International Conference on Computer Graphics, Artificial Intelligence, and Data Processing. Guangzhou: SPIE, 2023: 1044-1052.

[3] CHEN H, ZHOU Y, MEI K, et al. A new density peak clustering algorithm with adaptive clustering center based on differential privacy [J]. IEEE Access, 2023, 11: 1418-1431.

[4] JORDAN M I, MITCHELL T M. Machine learning: Trends, perspectives, and prospects[J]. Science, 2015, 349(6245):255-260.

[5] YU J, HONG R, WANG M, et al. Image clustering based on sparse patch alignment framework [J]. Pattern Recognition, 2014, 47(11):3512-3519.

[6] BUCZAK A L, GUVEN E. A survey of data mining and machine learning methods for cyber security intrusion detection[J].IEEE Communications Surveys & Tutorials, 2016, 18(2):1153-1176.

[7] ESTER M, KRIEGEL H, SANDER J, et al. A density-based algorithm for discovering clusters in large spatial databaseswith noise [C]//National Conferences on Aritificial Intelligence. Munich: AAAI, 1999: 836-841.

[8] ANKERST M, BREUNIG M M, KRIEGEL H P, et al. OPTICS: Ordering points to identify the clusteringstructure[C]// Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data. Philadelphia: ACM, 1999: 49-60.

[9] LIKAS A, VLASSIS N, VERBEEK J J. The global k-means clustering algorithm[J]. Pattern Recognition, 2003, 36(2): 451-461.

[10] MCPARLAND D, GORMLEY I C. Model based clustering for mixed data: ClustMD [J]. Advances in Data Analysis and Classification,2016,10(2):155-169.

[11] ROKACH L. A survey of clustering algorithms [M]// MAIMON O, ROKACH L, eds. Data Mining and Knowledge Discovery Handbook. Boston: Springer US, 2009: 269-298.

[12] CHEN Z, ZHAO T, LIU W. Time-aware collective spatial keyword query [J]. Computer Science and Information Systems, 2021,18(3):1077-1100.

[13] RODRIGUEZ A, LAIO A. Clustering by fast search and find of density peaks[J]. Science, 2014, 344(6191): 1492-1496.

[14] 吴亮, 刘国英. 局部密度峰聚类耦合字典学习的图像融合算法[J].计算机工程与设计, 2018, 39(7): 2008-2014.

[15] GUTTMAN A. R-trees: A dynamic index structure for spatial searching [C]//Proceedings of ACM SIGMOD Conference on Management of Data. Boston: ACM, 1984:47-57.

[16] CONG G, JENSEN C S, WU D. Efficient retrieval of the top-*k* most relevant spatial web objects[J]. Proceedings of the VLDB Endowment, 2009, 2(1): 337-348.

(责编 曹东,校对 刘钊)