软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(11):2847−2855 [doi: 10.13328/j.cnki.jos.004895]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种基于近邻表示的聚类方法*

周国兵, 吴建鑫, 周 嵩

(计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)

通讯作者: 吴建鑫, E-mail: wujx2001@nju.edu.cn, http://cs.nju.edu.cn/wujx/

摘 要: 当今社会处在信息急剧膨胀的时代,数据的规模和维度都在不断增大,传统的聚类方法有很多难以适应这一趋势.尤其是移动计算平台的高速发展,其平台自身的特性限制了算法的内存使用规模,因此,以往的很多方法若不进行改进,在这类平台上将无法运行.提出了一种基于近邻表示的聚类方法,该方法基于近邻的思想构造出新的表示形式,这种表示可以进行压缩,因此有效地减少了聚类所需要的存储开销.实现了直接对近邻表示压缩后的数据进行聚类的算法,称为 Bit *k*-means.实验结果表明,该方法取得了较好的效果,在提高准确率的同时,大幅度降低了存储空间开销.

关键词: 近邻;聚类

中图法分类号: TP181

中文引用格式: 周国兵,吴建鑫,周嵩.一种基于近邻表示的聚类方法.软件学报,2015,26(11):2847−2855. http://www.jos.org.cn/1000-9825/4895.htm

英文引用格式: Zhou GB, Wu JX, Zhou S. Clustering method based on nearest neighbors representation. Ruan Jian Xue Bao/ Journal of Software, 2015,26(11):2847−2855 (in Chinese). http://www.jos.org.cn/1000-9825/4895.htm

Clustering Method Based on Nearest Neighbors Representation

ZHOU Guo-Bing, WU Jian-Xin, ZHOU Song

(State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

Abstract: With the rapid expansion of information, scale and dimensionality of data are constantly increasing. Traditional clustering methods are difficult to adapt to this trend. Especially, given the fast development of mobile computing platforms, its properties limit the scale of memory that algorithms can use, so many algorithms cannot run on such platforms without making improvements. This paper proposes a clustering method based on nearest neighbor representation. This method uses the idea of nearest neighbors to construct the new representation. This new representation is compressible, thus effectively reducing the storage cost required for clustering. An algorithm called Bit *k*-means in implemented to perform clustering directly on the compressed nearest neighbors representation. Experimental results show that the new method achieves higher accuracy and substantially reduces the storage cost.

Key words: nearest neighbor; clustering

聚类(clustering)在数据挖掘、模式识别等领域中有着重要的作用,在识别数据的内在结构方面尤其有效.聚类主要应用于语音识别、字符识别、图像分割、数据压缩和信息检索等[1].迄今为止,聚类还没有一个公认的形式化定义,本文使用如下描述:聚类是将一个数据集划分为若干组或类的过程,通过聚类,使得同一组内的数据对象具有较高的相似度,而不同组中的数据对象则是不相似的[2,3].聚类通常需要计算数据集中数据之间的距离,因此当数据的维度很大时,传统的聚类方法计算速度会急速下降,同时会消耗大量的内存.当今社会处在数据急剧膨胀的时代,数据的规模和维度都在不断增大,传统的聚类方法很多难以适应这一趋势.尤其是移动计算平台的高速发展,其平台自身的特性限制了算法的内存使用规模,因此,以往的很多方法若不进行改进,在这类

* 基金项目: 国家自然科学基金(61422203)
收稿时间: 2015-05-24; 修改时间: 2015-07-14, 2015-08-11; 定稿时间: 2015-08-26

平台上将无法运行.然而,聚类分析本身依然具有很多优势,所以需要一种可以快速分析高维数据的聚类方法.

本文提出了一种新型的基于近邻(nearest neighbors)的数据表示,在此基础上进行聚类,实验中取得了很好的效果.我们使用近邻查找,找到样本的 k 最近邻,在新的数据表示中,将近邻所在位置赋值为 1,非近邻位置赋值为 0.实验表明:这种表示具有较高的准确率;同时,使用我们的压缩方法后,可以在准确率不变的情况下有效降低存储空间和运行时内存使用.

本文第 1 节对聚类算法进行简单的介绍.第 2 节详细介绍我们的方法.第 3 节为具体的实验和结果分析.第 4 节给出结论.

1 聚 类

由于数据的类型、维度等多种因素,没有一种聚类方法可以对所有的数据都适用[3].从最基本的聚类思想衍生出了多种聚类方法,可以根据聚类规则、距离计算方式等标准对聚类算法进行分类[4].常见的聚类算法主要有基于划分的算法、基于层次的算法、基于密度的算法、基于网格的算法和基于模型的算法等,其中具代表性的是 K-means 问题的 Lloyd 方法.

K-means 这个词最早由 MacQueen 于 1967 年使用[5],该问题的提出可以追溯到 1957 年,标准算法最早由 Lloyd 在 1957 年给出,但直到 1982 年,才发表了算法详细步骤和数学证明[6].K-means 的算法思想是:在给定的数据集 $D=[x_1,x_2,\ldots,x_n]$ 中,每个向量均为 d 维,将 n 个数据分成 k 个簇的集合 $S=[s_1,s_2,\ldots,s_k]$,其中,s_i 是集合 $\{x_1,x_2,\ldots,x_n\}$ 的子集,任 $s_i\cap s_j=\varnothing$ 且 $\bigcup_{i=1}^{k} s_i = D$. K-means 的目标是使簇内距离和最小(within-cluster sum of squares),该目标用公式表示为

$$\arg\min_S \sum_{i=1}^{k}\sum_{x\in s_i} \| x-\mu_i \|^2 \tag{1}$$

其中,μ_i 是属于 s_i 的数据的中心,用公式表示为

$$\mu_i = \frac{\sum_{x\in s_i} x}{|s_i|} \tag{2}$$

求解 K-means 问题的 Lloyd 方法具体算法描述如下:

算法 1.

输入:簇个数 k,数据集 D.

输出:k 个簇的集合 S.

1) 从 D 中随机选择 k 个点作为每个簇的中心
2) repeat
3) 对于每个数据 $x\in D$,找到距离最近的簇的中心 μ_i,标记 x 属于第 i 簇
4) 根据公式(2)更新每个簇中心的值
5) Until 目标函数(公式(1))值的变化在阈值内或达到循环次数,退出

Lloyd 方法是最经典的聚类算法,实现简单且具有很好的性能.同时,Lloyd 方法是具有可伸缩性的,可以适应分布式计算.但它也有一些缺点:(1) 对聚类中心的初始值很敏感;(2) 对于分类属性不适用;(3) 对于“噪音”和孤立点(outlier)数据是敏感的.

使用传统的 Lloyd 方法进行聚类时需要存储完整的数据集,同时将数据集全部或部分加载到内存中进行计算,在数据量大或平台存储空间限制时,算法难以发挥作用.本文提出的基于近邻表示的聚类算法可以较好地解决这个问题.

2 基于近邻表示的聚类算法

2.1 简 介

本文提出了一种基于近邻表示的聚类方法.该方法可以取得更高的准确率,同时使用的内存较少.第 3 节将

对该方法的性能进行分析.在文献[7]中使用数据集本身作为过匹配的字典,代替通用字典来表示数据集当中的数据,文中提到:如果每一类数据都可以找到足够的训练数据来构造字典,那么所有的测试数据将可以由字典中与它同类的数据线性表示.所以我们有理由相信,一个样例可以由与它相似的样例来表示.另一篇文献[8]阐述了近邻结构信息包含了较强的类别信息,可以有效地判断数据相似与否.由此我们提出:通过存储数据的近邻结构关系来替代数据本身,而这种近邻关系本身是以 0-1 向量的形式存在,故可以在该表示上使用数据压缩,压缩后所需要的存储量将远远小于原始数据本身,这就是基于近邻表示的聚类方法的基本思想.

为使文章表述清晰,下面我们约定本文中将使用的一些符号.$D\in\mathbb{R}^{n\times m}$表示 n 个 m 维数据组成的数据集,也可以表示$[x_1,x_2,\ldots,x_n]$,并且 $x_i=[x_{i1},x_{i2},\ldots,x_{in}]$,$x_{ij}$ 表示数据 x_i 在第 j 维的值.

2.2 方法细节

对于数据集 D 中的任意数据 x_i,首先我们计算它的近邻 N_i,$N_i=[n_{i1},n_{i2},\ldots,n_{ik}]$表示向量 x_i 的邻居集合,n_{ij} 为第 j 个邻居在 D 中的下标.若 $j\in N_i$,则设置 x_i 的近邻表示 x_i' 中的 x_{ij}' 为 1,否则为 0.因此,x_i 基于近邻的表示为 $x_i'=[x_{i1}',x_{i2}',\ldots,x_{in}']$,其中,$x_{ij}'$ 等于 0,表示 j 不在 N_i 中;等于 1,表示 j 在 N_i 中.即,x_i 和 x_j 是邻居.上文提到,D 中的数据 x_i 可以由与它相似的数据表示,因此我们相信,x_i 的近邻数据包含了这个数据本身的有效信息.故,我们使用 x_i' 来代替 x_i.若两个数据具有相似的近邻表示,也就意味着它们很有可能是同一个类别中的数据,因此可以在近邻表示上进行聚类.

在图 1 中,我们使用 k=5 的情况举例说明了如何生成一个近邻表示.一张 16×16 图片的原始数据 x_i 是一个 256×1 的向量,我们找到 x_i 的 5 个近邻,将近邻所在的位置赋为 1,非近邻所在位置赋为 0,这样就得到了基于近邻表示的向量 x_i'.x_i' 具有如下优点:(1) 是一个 0-1 向量,因此可以压缩存储,节省空间;(2) 可以通过修改聚类算法,直接计算两个压缩后向量之间的距离,使用压缩后的向量可以有效节约内存空间,降低运算时间,下一节我们将详细描述数据压缩和使用压缩后的向量进行计算的细节;(3) 包含了原始数据 x_i 的有效信息.

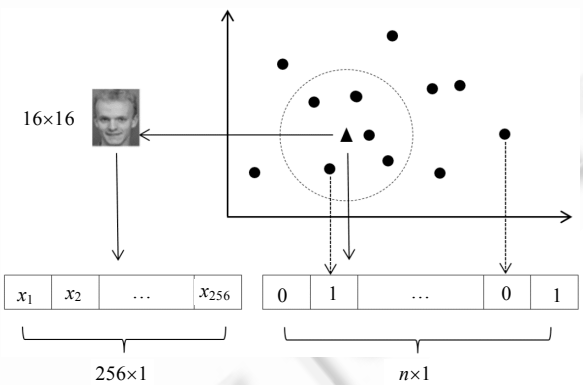


Fig.1 How to generate NN-based representation (k=5)

图 1 如何得到基于近邻的表示(k=5)

但是,这样做会有两个困难:(1) 当数据量增大时,寻找近邻本身会需要很长的计算时间,若计算 x_i 与其他所有点之间的距离并从中找到 k 个距离最小的点,则时间复杂度为 $O(nm-m+n)$,n 为数据集大小,m 为数据的维度;(2) 若只存储基于近邻的 0-1 向量,对于新数据将无法计算它的最近邻,而存储原始数据则耗费大量的存储空间.

对于上述两个问题,我们的解决方法是选择 FLANN(fast library for approximate nearest neighbors)[9]替代传统的 KNN 算法.FLANN 是一个高维空间中进行快速近似近邻查找的库,它包含了一系列高效的近邻查找算法和一个根据数据集自动匹配最优算法和参数的系统.在 FLANN 作者提供的实验中,在众多的数据集上,与现有的近似近邻搜索算法相比,FLANN 的查询效率是最高的.FLANN 的高效使得第 1 个问题得到很好的解决,同时,

FLANN 会生成一个模型文件,通过这个模型文件,我们可以找到任何数据在原始数据中的最近邻,而无需原始数据,这样就不需要存储原始数据.这个模型文件通常都非常小,不会带来额外的存储负担.

2.3 数据压缩与距离计算

在解决上述两个问题后,我们将产生的 0-1 向量进行压缩存储,将 0-1 向量的每一个值仅用一个比特(bit)表示,这样,存储的效率比最初的 0-1 向量提高了 8 倍.但是在计算压缩后向量之间的相似度时,需要使用大量的除法和移位操作,为了提高运算效率,我们使用了一个 256×8 的表,通过查表可以将一个字节(byte)还原成 8 个 0-1 值,只需要一个数组下标访问操作.

压缩近邻表示具体算法描述如下,其中,S 为 N 中的近邻表示经过压缩后组成的集合:

算法 2.

输入:近邻表示集合 N.

输出:压缩表示集合 S.

1) for i←1:n
2) for j←1:k
3) x_i 的第 j 个近邻为 $index$←N_{ij}
4) $block$←⌊$index$/8⌋
5) bit←$index$ mod 8
4) S[i][$block$]的第 bit 位赋为 1
7) end
8) end

压缩的数据在距离计算时需要将比特(bit)数据替换成真实数据,压缩数据中,一个字节(byte)称为一个 block,在计算过程中需要用到一张映射表 *dataMap*,这张表中存储的是一个字节与其包含的 8 个 0-1 比特的映射关系,可以直接通过字节对应的数值查询,具体算法描述如下:

算法 3.

输入:压缩向量 v,映射表 *dataMap*.

输出:真实数据 d.

1) repeat
2) $index$←v[$block$]
3) $realValue$[0,…,7]←$dataMap$[$index$]
4) for i←0:7
5) d[8×$block$+i]←$realValue$[i]
6) end
7) Until 遍历 v 所有的 $block$

恢复一个 Byte 查表需要 9 次运算,包括 1 次寻址和 8 次赋值;而位运算对于每个 bit 都需要先移位再获取 bit 信息,至少需要 16 次运算.因此,使用查表与直接使用位运算相比有明显的优势.

在解决了上述问题之后,我们的方法变得切实可行,并且理论上运行速度和节约使用空间都有很大的提升.下面我们在实验中对方法进行了验证.

3 实 验

在这一节,我们将展现我们方法在公开数据集上的具体表现.本节将详细阐述数据集的选取、实验参数设置和评价指标等细节.代码使用 C++实现,运行平台为 Visual Studio,CPU 为 Intel i7-4770,内存 8G.

3.1 数据集

本次实验中我们使用了 6 个数据集:COIL20[10],USPS,ATT face database(ORL)[11],Yale database[12],COIL100[13]和 MNIST[14].Yale 和 ORL 是人脸数据集,USPS 和 MNIST 是手写数据集,COIL20 和 COIL100 是物体数据集.

- COIL20:我们使用经过预处理版本的 COIL20 数据集,其中所有物品照片中的背景均被忽略了,并且照片均为可以覆盖物体的最小方形区域.数据集有 20 个不同物体,每个物体有 72 张照片,每张图片由 32×32 的 256 灰度值组成,因此,每张照片可以由一个 1 024 维的向量表示.
- USPS:一个手写数据集,我们使用它的一个常用子集,共 9 298 张照片.每张图片由 16×16 位的 256 灰度值组成,因此,每张照片可以由一个 256 维的向量表示.
- Yale Database:包含了 15 个个体的 gif 格式图片,每个人有 11 张照片,分别是不同的面部表情或场景,包括中间照光、左侧照光、右侧照光、戴眼镜、不戴眼镜、高兴、沮丧、瞌睡、惊奇和眨眼.每张图片由 32×32 的 256 灰度值组成,因此,每张照片可以由一个 1 024 维的向量表示.
- ATT face database(ORL):有 40 个不同的人,每个人有 10 张照片,每个人的照片有灯光、面部表情和面部细节的不同.所有的照片均是在昏暗的背景中面部朝前完成的.每张图片由 32×32 的 256 灰度值组成,因此,每张照片可以由一个 1 024 维的向量表示.
- COIL100:COIL20 的扩充版本,拥有 100 个不同物体,每个物体拥有 72 张图片,每张图片由 32×32 位的 256 灰度值组成,因此,每张照片可以由一个 1 024 维的向量表示.
- MNIST:一个手写数据集,拥有 60 000 张训练照片和 10 000 张测试照片,我们使用了其中的子集共 10 000 张.每张图片由 28×28 的 256 灰度值组成,因此,每张照片可以由一个 784 维的向量表示.

3.2 评价指标

实验的评价指标采用准确率(accuracy,简称 AC):对于一个给定的数据 x_i,r_i 和 s_i 分别是其聚类结果和原始标记值,则 AC 通过以下公式计算:

$$AC=\sum_{i=1}^{n}\frac{\delta(s_i,map(r_i))}{n} \tag{3}$$

其中,n 为数据集的大小;$\delta(x,y)$是一个二值函数,在 $x=y$ 时等于 1,否则为 0;Map 是一个映射函数,将聚类得到的标记映射到对应的真实标记上.目前,通常使用的映射方法是 Kuhn-Munkres 算法[15−17].

3.3 参数设置

为了能够准确地评价我们方法的效果,我们选取了最经典的聚类算法:k-means(Lloyd 方法).比较两种方法的结果:k-means,Bit k-means.k-means 的输入为原始数据,Bit k-means 的输入为压缩后的近邻表示,两个算法的具体逻辑相同,只是适应不同表示的数据输入.所有的这些方法只需要一个输入参数,即,聚类的个数 K.同时,由于我们选用 FLANN 作为近邻查找算法,需要确定两个参数:近邻个数 k 和检查次数(check number).因为聚类算法的初始化使用了随机函数,我们将每次实验的初始随机数种子设置为 0,运行 10 次.

为了能够公平地比较运行结果,我们需要为所有的 FLANN 设置相同的参数,所以需要找 FLANN 的相对最佳参数.对于检查次数,我们选取 COIL20 的结果作为标准,结果见表 1.

Table 1 Check number experiments

表 1 检查次数实验

Check	1	2	3	4	5	6	7	8	9	10	Avg.
64	0.54	0.56	0.54	0.50	0.49	0.58	0.48	0.50	0.58	0.56	0.53
128	0.54	0.50	0.55	0.52	0.53	0.62	0.55	0.53	0.60	0.60	0.55
256	0.55	0.48	0.59	0.52	0.58	0.59	0.55	0.63	0.63	0.55	0.57
512	0.58	0.46	0.58	0.57	0.62	0.63	0.62	0.70	0.66	0.59	0.60
1 024	0.56	0.68	0.62	0.67	0.56	0.67	0.59	0.63	0.70	0.62	**0.63**
inf	0.51	0.52	0.56	0.50	0.49	0.56	0.48	0.47	0.56	0.54	0.52

由表 1,我们可以确定检查次数选择 1 024.

由于每个数据集的大小均不相同,我们还需要找到一个合理的近邻个数 k,使得结果的准确性较高并且是稀疏的.实验中,我们从整个数据集的 2.5%,5%,7.5%和 10%中做选择,具体结果见表 2~表 7;

Table 2 KNN size experiment: COIL20

表 2 近邻个数实验:COIL20

Check	1	2	3	4	5	6	7	8	9	10	Avg.
2.50%	0.61	0.37	0.57	0.50	0.53	0.60	0.51	0.58	0.58	0.49	0.54
5%	0.56	0.68	0.62	0.67	0.56	0.67	0.59	0.63	0.70	0.62	**0.63**
7.50%	0.55	0.47	0.53	0.55	0.57	0.53	0.56	0.57	0.57	0.56	0.54
10%	0.46	0.46	0.49	0.52	0.51	0.56	0.53	0.49	0.53	0.54	0.51

Table 3 KNN size experiment: USPS

表 3 近邻个数实验:USPS

Check	1	2	3	4	5	6	7	8	9	10	Avg.
2.50%	0.60	0.60	0.60	0.60	0.45	0.73	0.60	0.61	0.61	0.52	0.59
5%	0.59	0.44	0.49	0.44	0.46	0.68	0.57	0.68	0.49	0.72	0.56
7.50%	0.57	0.57	0.64	0.64	0.69	0.66	0.60	0.67	0.59	0.52	0.61
10%	0.55	0.59	0.62	0.56	0.69	0.68	0.60	0.68	0.59	0.60	**0.62**

Table 4 KNN size experiment: Yale

表 4 近邻个数实验:Yale

Check	1	2	3	4	5	6	7	8	9	10	Avg.
2.50%	0.33	0.37	0.48	0.35	0.35	0.35	0.28	0.39	0.32	0.48	0.37
5%	0.26	0.45	0.53	0.42	0.31	0.35	0.41	0.33	0.44	0.36	**0.38**
7.50%	0.35	0.42	0.44	0.28	0.39	0.42	0.33	0.38	0.35	0.34	0.37
10%	0.31	0.36	0.41	0.37	0.32	0.38	0.34	0.28	0.38	0.35	0.35

Table 5 KNN size experiment: ORL

表 5 近邻个数实验:ORL

Check	1	2	3	4	5	6	7	8	9	10	Avg.
2.50%	0.50	0.48	0.49	0.51	0.50	0.50	0.47	0.49	0.45	0.52	**0.49**
5%	0.48	0.48	0.43	0.46	0.52	0.50	0.46	0.51	0.42	0.42	0.47
7.50%	0.48	0.52	0.48	0.49	0.47	0.50	0.49	0.47	0.44	0.44	0.48
10%	0.46	0.48	0.43	0.45	0.45	0.45	0.49	0.42	0.43	0.44	0.45

Table 6 KNN size experiment: COIL100

表 6 近邻个数实验:COIL100

Check	1	2	3	4	5	6	7	8	9	10	Avg.
2.50%	0.44	0.35	0.48	0.39	0.39	0.39	0.38	0.49	0.42	0.48	0.42
5%	0.46	0.47	0.47	0.50	0.47	0.42	0.49	0.46	0.48	0.44	**0.47**
7.50%	0.45	0.42	0.44	0.38	0.39	0.42	0.43	0.41	0.40	0.45	0.41
10%	0.41	0.39	0.44	0.39	0.42	0.42	0.39	0.38	0.39	0.40	0.39

Table 7 KNN size experiment: MNIST

表 7 近邻个数实验:MNIST

Check	1	2	3	4	5	6	7	8	9	10	Avg.
2.50%	0.50	0.48	0.49	0.51	0.50	0.50	0.47	0.49	0.45	0.52	0.49
5%	0.55	0.51	0.50	0.47	0.51	0.50	0.51	0.53	0.51	0.50	**0.51**
7.50%	0.48	0.52	0.48	0.49	0.47	0.50	0.49	0.47	0.44	0.44	0.48
10%	0.46	0.48	0.43	0.45	0.45	0.45	0.49	0.42	0.43	0.44	0.45

由表 2~表 7,综合在各个数据集上的表现,我们选取整个数据集的 5%作为近邻个数(k)的值,即 k=0.05n.

3.4 实验结果

在检查次数为 1 024、k 为整个数据集的 5%的参数设置下,我们对两种方法进行了实验,每种方法运行 10

次,得到实验结果图 2(*x* 轴为次数,*y* 轴为准确率).同时,针对 10 次的准确率统计得到最高准确率(max AC)和平均准确率(average AC),详情见表 8.

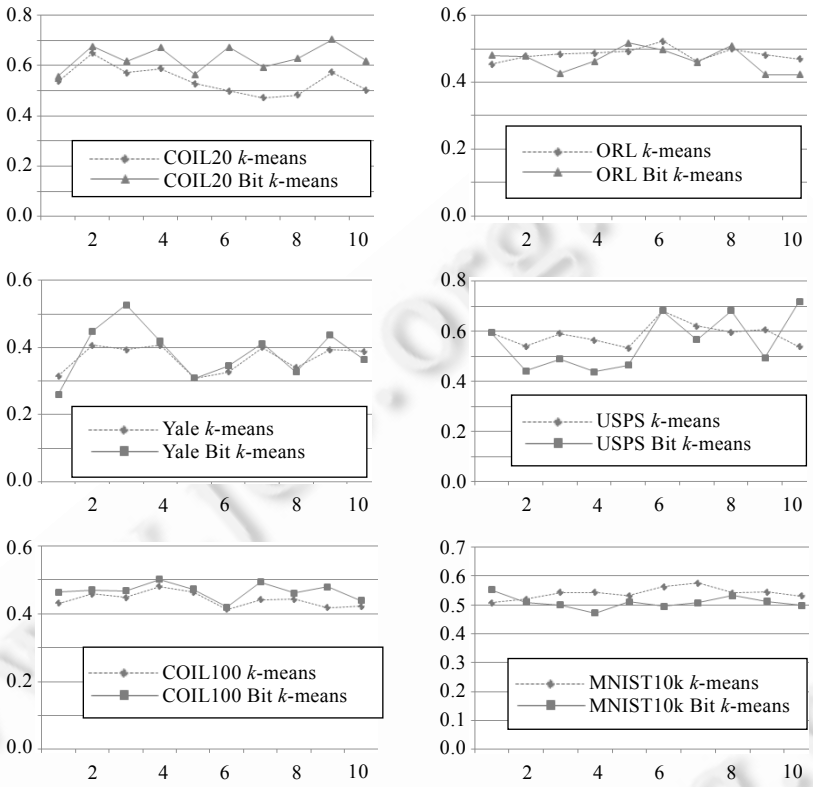


Fig.2 Results on six datasets

图 2 6 个数据集的实验结果

Table 8 Average AC and max AC

表 8 平均准确率和最高准确率

	Avg-AC		Max-AC	
	k-means	Bit *k*-means	*k*-means	Bit *k*-means
COIL20	0.540±0.055	**0.630±0.050**	0.649	**0.703**
ORL	**0.484±0.019**	0.468±0.035	**0.523**	0.518
Yale	0.368±0.040	**0.385±0.078**	0.406	**0.527**
USPS	**0.586±0.045**	0.557±0.107	0.682	**0.719**
COIL100	0.442±0.022	**0.467±0.024**	0.481	**0.502**
MNIST	**0.540±0.020**	0.509±0.021	**0.576**	0.552

在实验结果图中,从左上到右下分别为以下数据集的结果:COIL20,ORL,Yale,USPS,COIL100 和 MNIST.从表 8 平均准确率来看,在 COIL20,Yale 和 COIL100 上,我们的方法比现有方法有提升,在 ORL,USPS 和 MNIST 上准确率稍有下降.ORL 中准确率略有下降,与 ORL 本身每个类别的数据过少有关.由于每个类的数据过少,这样,每个数据的近邻个数相对较多,因此被误判为同一类的概率就会相应地增加,最终导致了整体准确率的下降.而真实世界的数据集中,单类样例个数往往远大于 ORL 中的 10,因此这种情况在实际中发生的概率较小.

从表 8 最高准确率来看,我们的方法在 COIL20,Yale,USPS,COIL100 均取得了更高的准确率,其中,COIL20 数据集上 Bit *k*-means 相对 *k*-means 提高了 8%,Yale 数据集上提高了 29.8%.综合而言,基于近邻表示的聚类方法在准确率上取得了更好的表现.

MNIST,USPS 数据集为手写数据集,数字一共有 10 种(0~9),某些数字在增加了手写的误差后,差别变得很小,如 0,9 和 8,7 和 1,5 和 6.因此,这些数据的近邻都很相似,这时候再对基于近邻表示的向量进行相似度计算时,极少的近邻差异就会被归入不同类别,这大大增加了出错的概率.因此,我们的方法在数据集的数据近邻均非常相似时会有一定的局限性,这是以后需要加以改进的.

3.5 内存空间分析

在实际的运行过程中,*k*-means 需要的数据部分内存大小为 $n\times m\times 8$ 字节,其中,n 为样本(instance)个数,m 为属性(dimension)个数,8 表示 double 类型的长度.Bit *k*-means 需要的数据部分内存大小为 $n\times n/8$ 字节,当 $n\leqslant 64\times m$ 时,我们的方法使用的内存空间会更小,这个分析同样适用于数据在硬盘上的存储.因此,该方法在属性较多的数据,如文本、图像等,具有较好的应用前景.具体的压缩结果见表 9,其中,space 表示原始数据使用的空间,compress 表示使用我们的近邻表示方法并进行压缩后的空间,单位为 KB.

Table 9 Compress result

表 9 压缩效果

	Dimension	Instance	Space	Compress
COIL20	1 024	1 440	11 520	253
ORL	1 024	400	3 200	19
Yale	1 024	165	1 320	3
USPS	256	9 298	18 596	10 553
COIL100	1 024	7 200	57 600	6 328
MNIST	784	10 000	61 250	12 207

3.6 运行时间分析

我们对程序的运行时间也进行了统计,具体结果见表 10,时间的单位是 s.从结果我们可以看出,当 $n>m$ 时,新的表示在压缩前是比原始数据大的,这样在计算距离时需要更大的计算量,因此运行时间有一定的增加.我们的程序针对新的表示的二值特性做了一些优化,提升了运行效率,但依然无法消除数据维度变大带来的影响.因此,我们的表示更适用于属性维度较高的场景.

Table 10 Running time

表 10 运行时间

	Dimension	Instance	*K*-means	Bit *k*-means
COIL20	1 024	1 440	405	446
ORL	1 024	400	91	38
Yale	1 024	165	19	7
USPS	256	9 298	1 183	8 227
COIL100	1 024	7 200	2 230	6 415
MNIST	784	10 000	3 156	10 113

4 结束语

本文使用基于近邻的聚类方法,大大降低了数据存储和内存开销;同时,在一定程度上提高了聚类的准确性.但该方法对于数据差异较小的数据集有一定的局限性,仍需要加以改进.对于属性较多数据,我们的方法可以显著降低运行内存和存储开销.

References:

[1] Sun JG, Liu J, Zhao LY. Clustering algorithms research. Ruan Jian Xue Bao/Journal of Software, 2008,19(1):48−61 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/48.htm [doi: 10.3724/SP.J.1001.2008.00048]

[2] Zhu M. Introduction to Data Mining. Hefei: Press of University of Science and Technology of China, 2002. 138−139 (in Chinese).

[3] Jain AK, Dubes RC. Algorithms for Clustering Data. Prentice-Hall, Inc., 1988. 1−334.

[4] Gelbard R, Goldman O, Spiegler I. Investigating diversity of clustering methods: An empirical comparison. Data & Knowledge Engineering, 2007,63(1):155−166. [doi: 10.1016/j.datak.2007.01.002]

[5] MacQueen J. Some methods for classification and analysis of multivariate observations. In: Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability. 1967. 281−297.

[6] Lloyd S. Least squares quantization in PCM. IEEE Trans. on Information Theory, 1982,28(2):129−137. [10.1109/TIT.1982.1056489]

[7] Wright J, Yang AY, Ganesh A, Sastry SS, Ma Y. Robust face recognition via sparse representation. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2009,31(2):210−227. [doi: 10.1109/TPAMI.2008.79]

[8] Wu JX. Balance support vector machines locally using the structural similarity kernel. In: Proc. of the Pacific-Asia Conf. on Knowledge Discovery and Data Mining. 2011. 112−123. [doi: 10.1007/978-3-642-20841-6_10]

[9] Muja M, Lowe DG. Fast approximate nearest neighbors with automatic algorithm configuration. In: Proc. of the Int'l Conf. on Vision Theory and Applications. 2009. 331−340.

[10] Nene SA, Nayar SK, Murase H. Columbia object image library (COIL-20). Technical Report, CUCS-005-96, New York: Department of Computer Science, Columbia University, 1996.

[11] Samaria FS, Harter AC. Parameterisation of a stochastic model for human face identification. In: Proc. of the 2nd IEEE Workshop on the Applications of Computer Vision. 1994. 138−142. [doi: 10.1109/ACV.1994.341300]

[12] Belhumeur PN, Hespanha JP, Kriegman D. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1997,19(7):711−720. [doi: 10.1109/34.598228]

[13] Nayar SK, Nene SA, Murase H. Columbia object image library (coil 100). Technical Report, CUCS-006-96, New York: Department of Computer Science, Columbia University, 1996.

[14] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-Based learning applied to document recognition. Proc. of the IEEE, 1998, 86(11):2278−2324. [doi: 10.1109/5.726791]

[15] Kuhn HW. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2005,52(1):7−21. [doi: 10.1002/nav.20053]

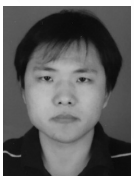
[16] Kuhn HW. Variants of the Hungarian method for assignment problems. Naval Research Logistics Quarterly, 1956,3(4):253−258. [doi: 10.1002/nav.3800030404]

[17] Munkres J. Algorithms for the assignment and transportation problems. Journal of the Society for Industrial & Applied Mathematics, 1957,5(1):32−38. [doi: 10.1137/0105003]

附中文参考文献:

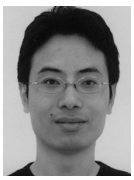
[1] 孙吉贵,刘杰,赵连宇.聚类算法研究.软件学报,2008,19(1):48−61. http://www.jos.org.cn/1000-9825/19/48.htm [doi: 10.3724/SP.J.1001.2008.00048]

[2] 朱明.数据挖掘导论.合肥:中国科学技术大学出版社,2002.138−139.



周国兵(1990−),男,安徽马鞍山人,硕士生,主要研究领域为机器学习.

周嵩(1976−),男,博士,讲师,CCF 会员,主要研究领域为数据库,数据仓库,数据挖掘.



吴建鑫(1978−),男,博士,教授,博士生导师,CCF 会员,主要研究领域为计算机视觉,机器学习.