个人资料

skyline0623

访问： 96296次
积分： 1239
等级： BLOG 4
排名： 千里之外

原创： 21篇 转载： 3篇
译文： 2篇 评论： 86条

【机器学习】K-means聚类算法初探

标签： k-means 数据挖掘 机器学习 聚类算法

2012-11-06 20:38 20631人阅读 评论(2) 收藏 举报

分类： 机器学习与应用

算法代码 Github传送门：K-MeansCluster@skyline0623

数据聚类是对于静态数据分析的一门技术，在许多领域内都被广泛地应用，包括机器学习、数据挖掘、模式识别、图像分析、信息检索以及生物信息等。聚类是把相似的对象通过静态分类的方法分成不同的组别或者更多的子集，这样让在同一个子集中的成员对象都有相似的一些属性，常见的包括在坐标系中更加短的空间距离等。

我们拿2维特征的实例作为例子。我们以这个2维特征向量作为坐标，在一个2维空间中用点标注出这些实例，如图1所示，这里是随机生成的100个实例。图1中带颜色的方框同样是实例，在之后介绍的K-means算法中，这些方框是初始被随机选择出来的聚类中心点。

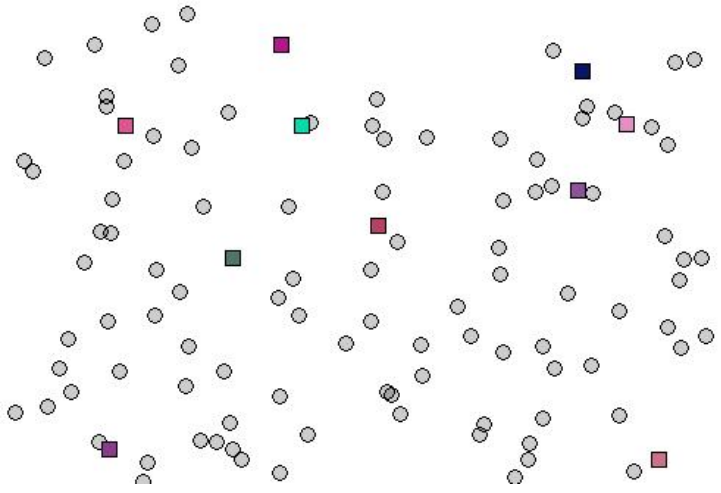


图1 在二维空间中100个具有两个特征的实例

我们希望聚类算法能够将特征相近的实例聚集成一个集合，最后形成多个由特征相近的实例聚集成的聚类。如图2所示，我们将上面的数据通过聚类算法聚集成了10个类，分别用10种颜色表示，其中点的颜色标示它属于哪一个聚类。这里使用了点间的欧式距离作为评价两个点特征相似程度的度量。

关于二叉树，我们的中国 (16111)
如何在C语言中实现简单 (7314)
哈夫曼树的初始化，编码 (5901)
让多核CPU占用率曲线呀 (5040)
让多核CPU占用率曲线呀 (4467)
一组数字的全排列按序输 (3225)
《Ph.D Grind》阅读笔记 (2554)
关于Python的WEB开发框 (2373)
C语言中的位运算&结构体 (2217)

给一些我常拜读的价值博客链接
Mind Hacks 思维改变生活|刘未鹏 MSRA
4G spaces|徐宥 Google
Matrix67|数学大牛

评论排行
关于二叉树，我们的中国 (38)
学习重要，还是经营人脉 (11)
如何在C语言中实现简单 (11)
哈夫曼树的初始化，编码 (6)
我的处女博文——乱78糟 (4)
可怜之人必有可恨之处！ (4)
成为究极程序员的艰难的 (3)
如何在C语言中实现简单 (2)
让多核CPU占用率曲线呀 (2)
关于Facebook Graph Se (2)

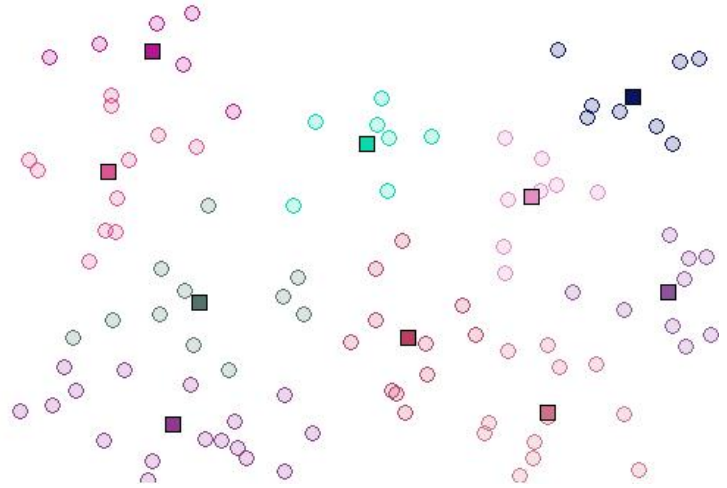


图2 聚成10个类的效果

通常，我们将聚类分析作为一种无监督机器学习算法来看待。与有监督学习不同，如常见的分类问题，我们有标注好分类类别的训练数据，通过这些数据训练出一个模型来对新的数据的分类进行预测。在聚类分析中，我们的数据是没有标注类别的，我们通过数据中实例的特征值相似程度将相似的实例划分到一类中。

在接下来的内容中，我们将介绍一种简单的聚类算法----K-means，第二部分它的原理、过程以及可能存在的问题和解决方案。第三部分介绍K-means算法的具体实现中的流程与细节，以及实验结果的分析。

1 K-Means算法介绍

1.1 算法初探

K-means是一种基于距离的迭代式算法[1]。它将n个观察实例分类到k个聚类中，以使得每个观察实例距离它所在的聚类的中心点比其他的聚类中心点的距离更小。

其中，距离的计算方式可以是欧式距离（2-norm distance），或者是曼哈顿距离（Manhattan distance,1-norm distance）或者其他。这里我们使用欧式距离。

要将每个观测实例都归类到距离它最近的聚类中心点，我们需要找到这些聚类中心点的具体位置。然而，要确定聚类中心点的位置，我们必须知道该聚类中包含了哪些观测实例。这似乎是一个“先有蛋还是先有鸡”的问题。从理论上来说，这是一个NP-Hard问题[3]。

但是，我们可以通过启发式的算法来近似地解决问题。找到一个全局最优的方案是NP-hard问题，但是，降低问题的难度，如果能够找到多个局部最优方案，然后通过一种评价其聚类结果优劣的方式，把其中最优的解决方案作为结果，我们就可以得到一个不错的聚类结果[2]。

这就是为什么我们说K-means算法是一个迭代式的算法。算法[2]的过程如下：

1）所有的观测实例中随机抽取出k个观测点，作为聚类中心点，然后遍历其余的观测点找到距离各自最近的聚类中心点，将其加入到该聚类中。这样，我们就有了一个初始的聚类结果，这是一次迭代的过程。

2）我们每个聚类中心都至少有一个观测实例，这样，我们可以求出每个聚类的中心点（means），作为新的聚类中心，然后再遍历所有的观测点，找到距离其最近的中心点，加入到该聚类中。然后继续运行2）。

3）如此往复2），直到前后两次迭代得到的聚类中心点一模一样。

这样，算法就稳定了，这样得到的k个聚类中心，和距离它们最近的观测点构成k个聚类，就是我们要的结果。

实验证明，算法试可以收敛的[2]。

计算聚类的中心点有三种方法如下：

1）Minkowski Distance 公式 —— λ 可以随意取值，可以是负数，也可以是正数，或是无穷大。

$$d_{ij} = \sqrt[\lambda]{\sum_{k=1}^{n} |x_{ik} - x_{jk}|^{\lambda}}$$

公式（1）

2）Euclidean Distance 公式 —— 也就是第一个公式 λ=2 的情况

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - x_{jk} \right)^2}$$

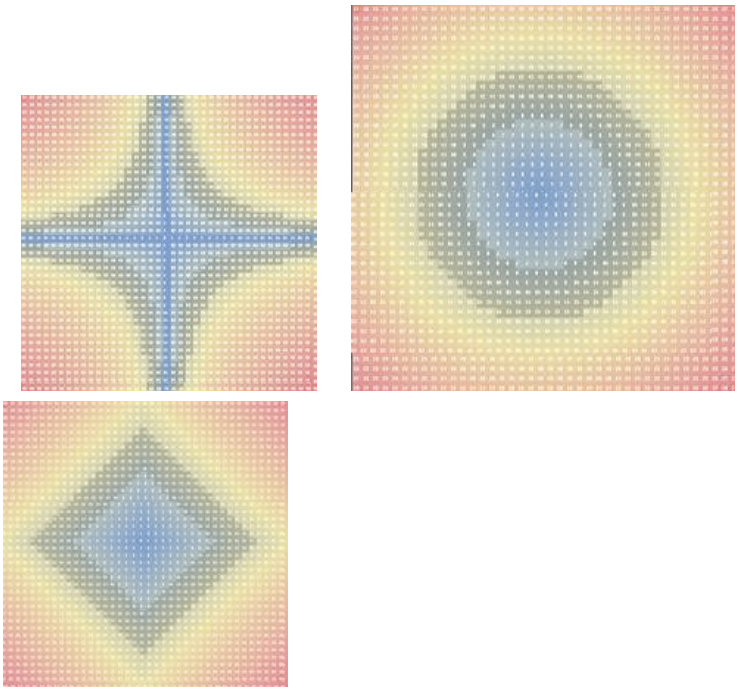
公式（2）

3）CityBlock Distance 公式 —— 也就是第一个公式 λ=1 的情况

$$d_{ij} = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|$$

公式（3）

这三个公式的求中心点有一些不一样的地方，我们看下图（对于第一个 λ 在 0-1之间）。



（1）Minkowski Distance （2）Euclidean Distance

（3） CityBlock Distance

上面这几个图的大意是他们是怎么个逼近中心的，第一个图以星形的方式，第二个图以同心圆的方式，第三个图以菱形的方式。我们采用第二种方式，在实际实现中，我们并没有开根号。

我们算法十分的简单。下面对100个随机生成的2维特征的观测实例在2维空间中运行这个K-means算法的过程用图表示出来。整个过程用了10次迭代后收敛。假设k=10。

100个观测点的初始分布如图1所示。方块为随机挑出的10个聚类中心点。

图3显示了将各个观测值划分到距离最近的聚类后的结果。

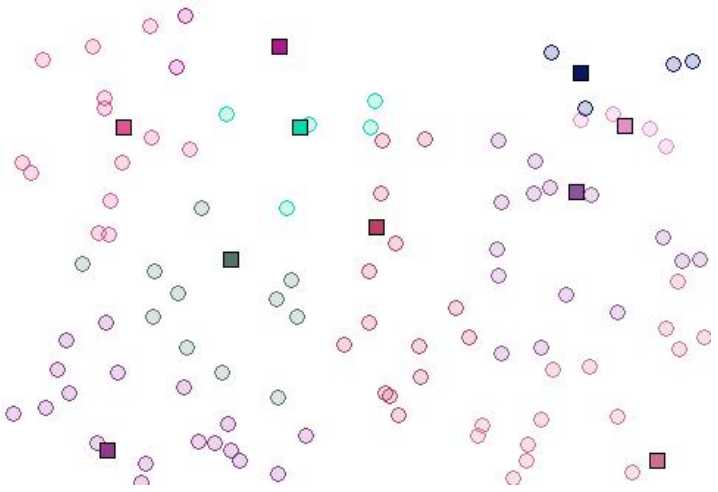


图3

然后根据每个聚类的点的情况，计算出新的中心点，得到新的聚类中心，再以此往复，直到达到收敛的条件。

在第9次迭代和第10次迭代中，聚类中心没有任何变化，所以算法收敛，结束。从图5中可以看出，聚类中心在这10次迭代中位置变化的过程。黑色方块为初始位置，白色为收敛后位置，其余同一颜色的为中间过程中的位置。

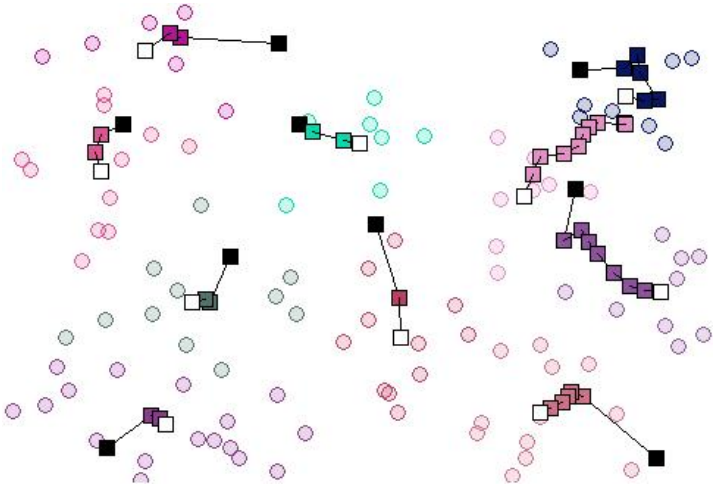


图5 聚类中心位置变化的过程

那么，如何评价一个聚类结果呢？我们计算所有观测点距离它对应的聚类中心的距离的平方和即可，我们称这个评价函数为evaluate(C)。它越小，说明聚类越好。

1.2 K-means的问题及解决方案

K-means算法非常简单，然而却也有许多问题。

1）首先，算法只能找到局部最优的聚类，而不是全局最优的聚类。而且算法的结果非常依赖于初始随机选择的聚类中心的位置。我们通过多次运行算法，使用不同的随机生成的聚类中心点运行算法，然后对各自结果C通过evaluate(C)函数进行评估，选择多次结果中evaluate(C)值最小的那一个。

2）关于初始k值选择的问题。首先的想法是，从一个起始值开始，到一个最大值，每一个值运行k-means算法聚类，通过一个评价函数计算出最好的一次聚类结果，这个k就是最优的k。我们首先想到了上面用到的evaluate(C)。然而，k越大，聚类中心越多，显然每个观测点距离其中心的距离的平方和会越小，这在实践中也得到了验证。第四节中的实验结果分析中将详细讨论这个问题。

3）关于性能问题。原始的算法，每一次迭代都要计算每一个观测点与所有聚类中心的距离。有没有方法能够提高效率呢？是有的，可以使用k-d tree或者ball tree这种数据结构来提高算法的效率。特定条件下，对于一定区域内的观测点，无需遍历每一个观测点，就可以把这个区域内所有的点放到距离最近的一个聚类中去。这将在第三节中详细地介绍。

2 实现

2.1 k-d tree 与 ball tree

k-d tree[5]或者ball tree[4]这个数据结构在K近邻算法中被用到来提高算法的效率，在K-means中能否使用这些数据结构呢？当然能，而且，它们用在这里更加高效。在这里，我们使用ball tree来优化算法的性能。我们首先介绍k-d tree。

1）k-d tree[5]

把n维特征的观测实例放到n维空间中，k-d tree每次通过某种算法选择一个特征(坐标轴)，以它的某一个值作为分界做超平面，把当前所有观测点分为两部分，然后对每一个部分使用同样的方法，直到达到某个条件为止。
上面的表述中，有几个地方下面将会详细说明：（1）选择特征（坐标轴）的方法　　（2）以该特征的哪一个为界　（3）达到什么条件算法结束。

(1)选择特征的方法

计算当前观测点集合中每个特征的方差，选择方差最大的一个特征，然后画一个垂直于这个特征的超平面将所有观测点分为两个集合。

（2)以该特征的哪一个值为界 即垂直选择坐标轴的超平面的具体位置。

第一种是以各个点的方差的中值（median）为界。这样会使建好的树非常地平衡，会均匀地分开一个集合。这样做的问题是，如果点的分布非常不好地偏斜的，选择中值会造成连续相同方向的分割，形成细长的超矩形(hyperrectangles)。

替代的方法是计算这些点该坐标轴的平均值，选择距离这个平均值最近的点作为超平面与这个坐标轴的交点。这样这个树不会完美地平衡，但区域会倾向于正方地被划分，连续的分割更有可能在不同方向上发生。

（3）达到什么条件算法结束

实际中，不用指导叶子结点只包含两个点时才结束算法。你可以设定一个预先设定的最小值，当这个最小值达到时结束算法。

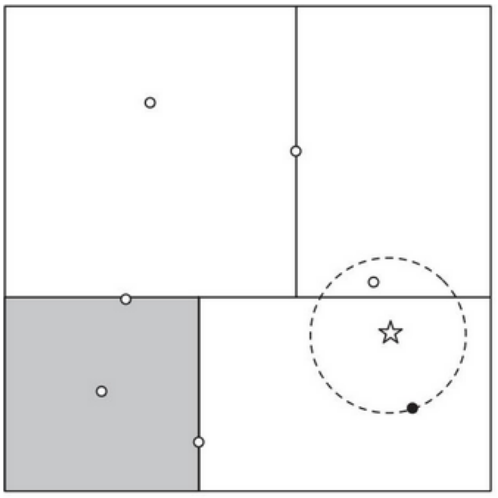


图 6 一个k-d tree划分二维空间

图6中，星号标注的是目标点，我们在k-d tree中找到这个点所处的区域后，依次计算此区域包含的点的距离，找出最近的一个点（黑色点），如果在其他region中还包含更近的点则一定在以这两个点为半径的圆中。假设这个圆如图中所示包含其他区域。先看这个区域兄弟结点对应区域，与圆不重叠；再看其双亲结点的兄弟结点对应区域。从它的子结点对应区域中寻找（图中确实与这个双亲结点的兄弟结点的子结点对应区域重叠了）。在其中找是否有更近的结点。

k-d tree的优势是可以递增更新。新的观测点可以不断地加入进来。找到新观测点应该在的区域，如果它是空的，就把它添加进去，否则，沿着最长的边分割这个区域来保持接近正方形的性质。这样会破坏树的平衡性，同时让区域不利于找最近邻。我们可以当树的深度到达一定值时重建这棵树。

然而，k-d tree也有问题。矩形并不是用到这里最好的方式。偏斜的数据集会造成我们想要保持树的平衡与保持区域的正方形特性的冲突。另外，矩形甚至是正方形并不是用在这里最完美的形状，由于它的角。如果图6中的圆再大一些，即黑点距离目标点点再远一些，圆就会与左上角的矩形相交，需要多检查一个区域的点，而且那个区域是当前区域双亲结点的兄弟结点的子结点。

为了解决上面的问题，我们引入了ball tree。

2）ball tree[4]

解决上面问题的方案就是使用超球面而不是超矩形划分区域。使用球面可能会造成球面间的重叠，但却没有关系。ball tree就是一个k维超球面来覆盖这些观测点，把它们放到树里面。图7（a)显示了一个2维平面包含16个观测实例的图,图7（b）是其对应的ball tree，其中结点中的数字表示包含的观测点数。

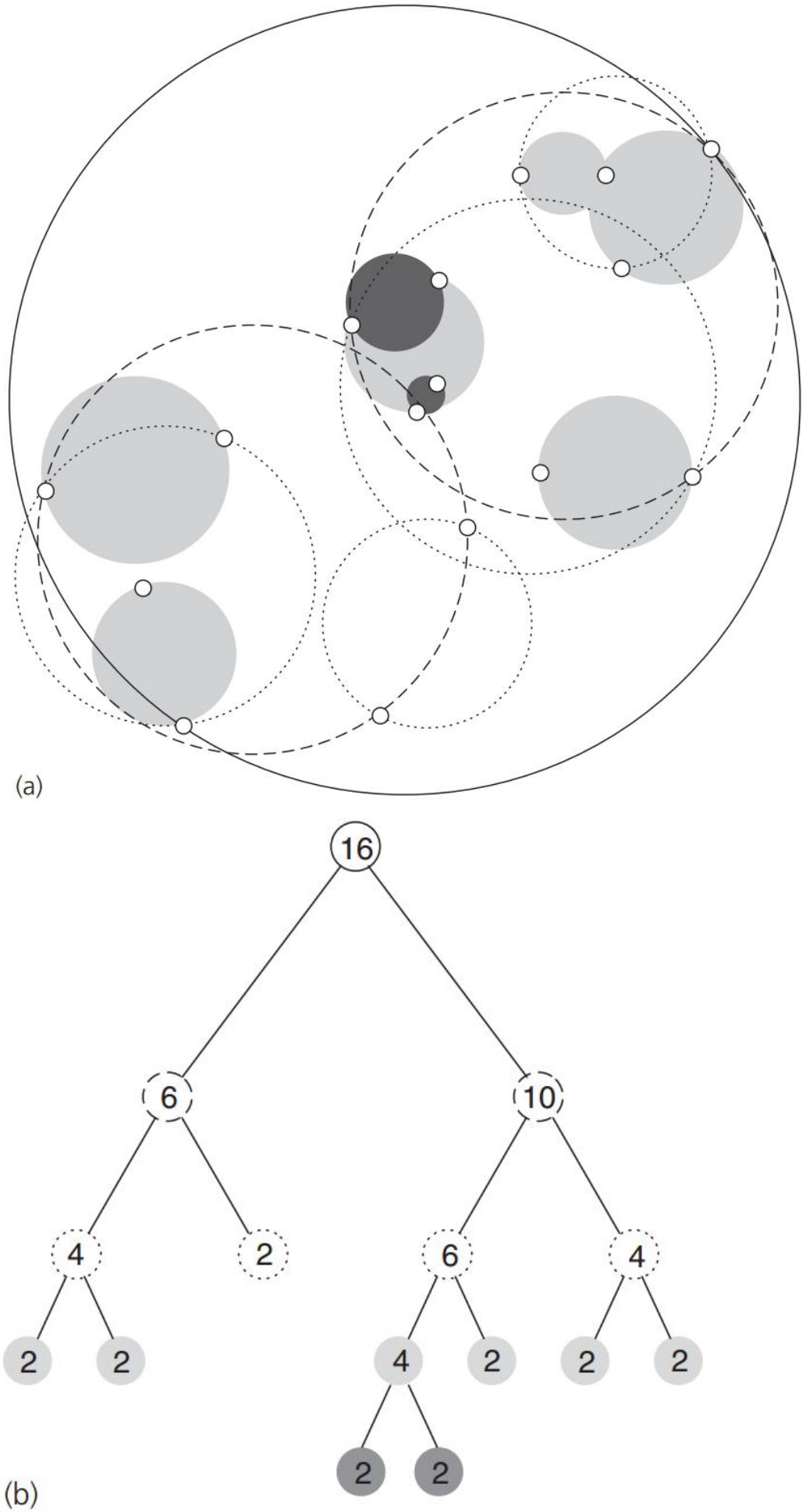


图 7 ball tree对二维平面的划分和ball tree

不同层次的圆被用不同的风格画出。树中的每个结点对应一个圆，结点的数字表示该区域保含的观测点数，但不一

定就是图中该区域囊括的点数，因为有重叠的情况，并且一个观测点只能属于一个区域。实际的ball tree的结点保存圆心和半径。叶子结点保存它包含的观测点。

使用ball tree时，先自上而下找到包含target的叶子结点，从此结点中找到离它最近的观测点。这个距离就是最近邻的距离的上界。检查它的兄弟结点中是否包含比这个上界更小的观测点。方法是：如果目标点距离兄弟结点的圆心的距离大于这个圆的圆心加上前面的上界的值，则这个兄弟结点不可能包含所要的观测点。（如图8）否则，检查这个兄弟结点是否包含符合条件的观测点。

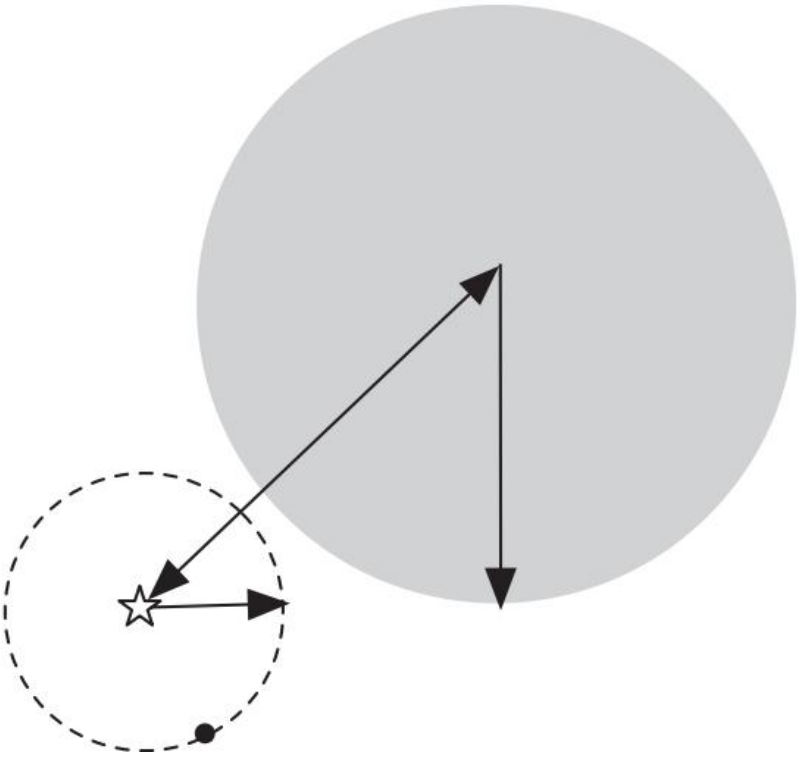


图 8 点与超圆

那么，ball tree的分割算法是什么呢？

选择一个距离当前圆心最远的观测点i1，和距离i1最远的观测点 i2，将圆中所有离这两个点最近的观测点都赋给这两个簇的中心，然后计算每一个簇的中心点和包含所有其所属观测点的最小半径。对包含n个观测点的超圆进行分割，只需要线性的时间。

与k-d tree一样，如果结点包含的观测点到达了预先设定的最小值，这个顶点就可以不再分割了。

2.2 在K-means算法中使用ball tree

在k-means算法中使用k-d tree和ball tree效率更高，因为在此所有的观测点都是一起处理，而在k近邻中，测试的观测点单独处理。

首先，根据所有的观测点创建一个包含它们的k-d tree 或者是ball tree。

在k-means算法的每一次迭代中，设定的聚类中心集合为C={Ci，i= 1,...,k},每个簇中心有一个对应其归属它的观测点集合，初始为空。从根结点开始遍历树，直到找到叶子结点为止，判断其中的观测点距离哪一个中心近，然后赋给那个中心。有可能在浏览一个内部结点时，它所包含的点完全包含在一个聚类的区域内。这时只需要直接把其下所有的观测点加入到该聚类中即可。

我们可以在每个簇的中心点保存一个向量，保存聚类中点各个坐标之和和点的个数。在计算其中心点时只需一除即可。

在多维空间中，如何判断一个内部结点包含的观测点全部落在一个聚类中呢？ 我们采用下面描述的方法。

Stackoverflow上的讨论见这里：点击这里

如果一个空间容器比如这里的超球面与某一簇中心的最大距离小于其与其它所有簇中心的最小距离，即：

$$\exists mean_i \; max\_dist(mean_i, container) < min \text{ of all } j!=i \; min\_dist(mean_j, container)$$ 公式（4）

其中，表示聚类的中心点，如果上式成立，则可推出：

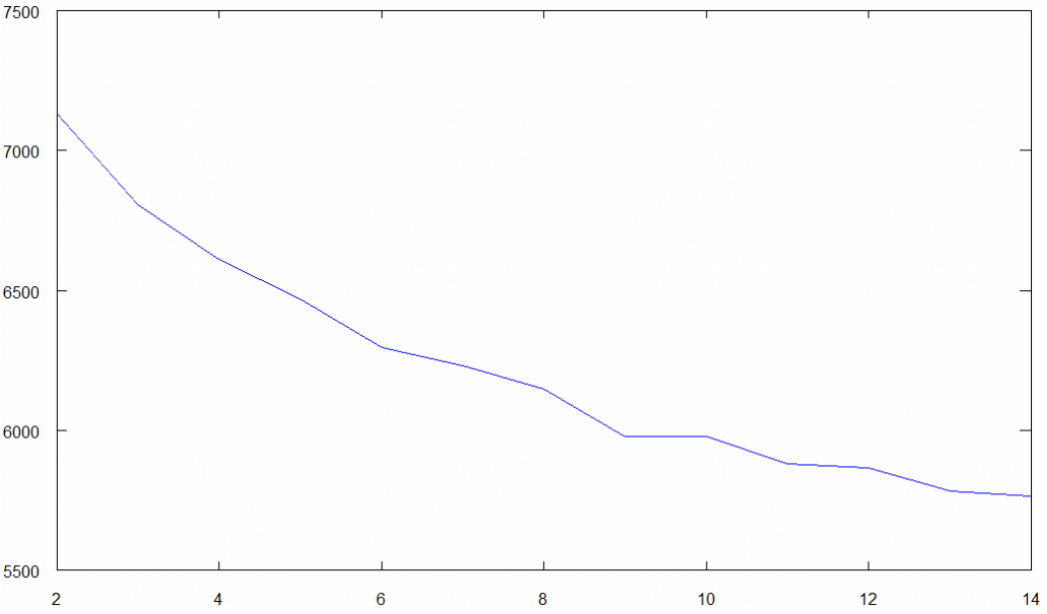
$$\forall obj \text{ in the container } dist(mean_i, obj) < min \text{ of all } j!=i \; dist(mean_j, obj)$$ 公式（5）

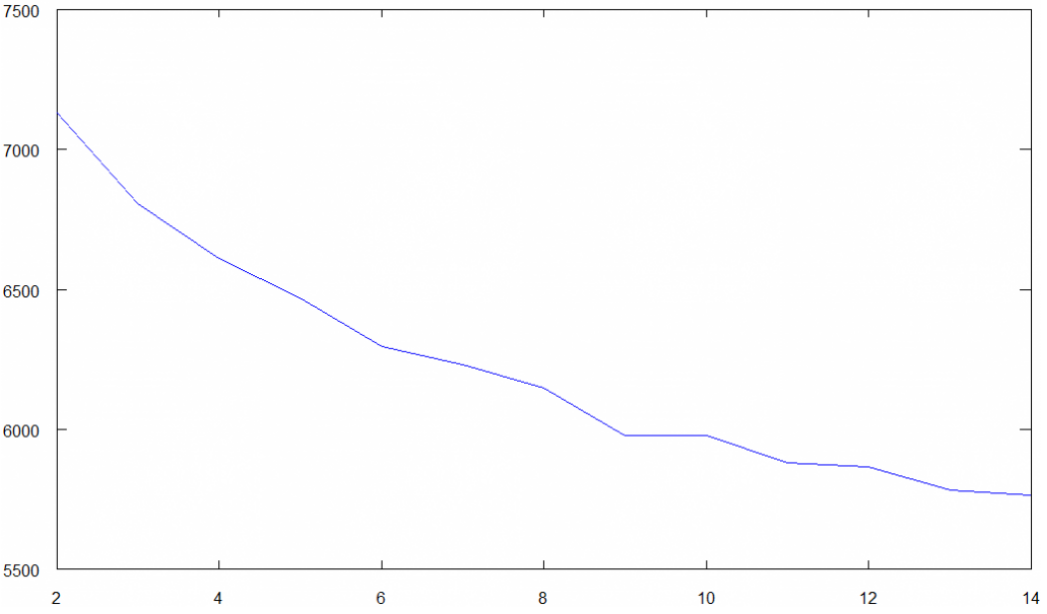
即此容器中的任意观测点距离这个的距离都小于距离其他聚类中心点的距离。此时，可以断定这个容器中的所有观测点都在这个所在的聚类中。

在我们这个环境即ball tree中，任意点与超球面的最大距离为其与圆心距离加圆的半径，最小距离为其与圆心距离减去圆的半径。在实际的运行中，这个策略可以明显地提高程序运行的性能。经大量的实验得出，在对ball tree中的结点访问时，能够避免继续向下探索的次数占所有访问结点次数平均达到17.8764%。每次避免访问其子树时，就大量减少了访问总结点的时间。

3 实验结果分析

我们从k=2开始，一直调用K-means算法（每次调用10次，选择evaluate最小那一次聚类的结果返回）直到k=14.我们记录下每个k聚类后评估函数evaluate(Ck)的值，然后画出其变化的曲线。理论上，随着k的增大，evaluate(Ck)的值一定是非递增的。因为中心越多，各个观测点距离中心的平均距离越近。所以，我们通过另一种方式来确定k。我们在一个要聚类数据集上我们按照上面的方法运行多次，其evaluate关于k的趋势图如图9所示：





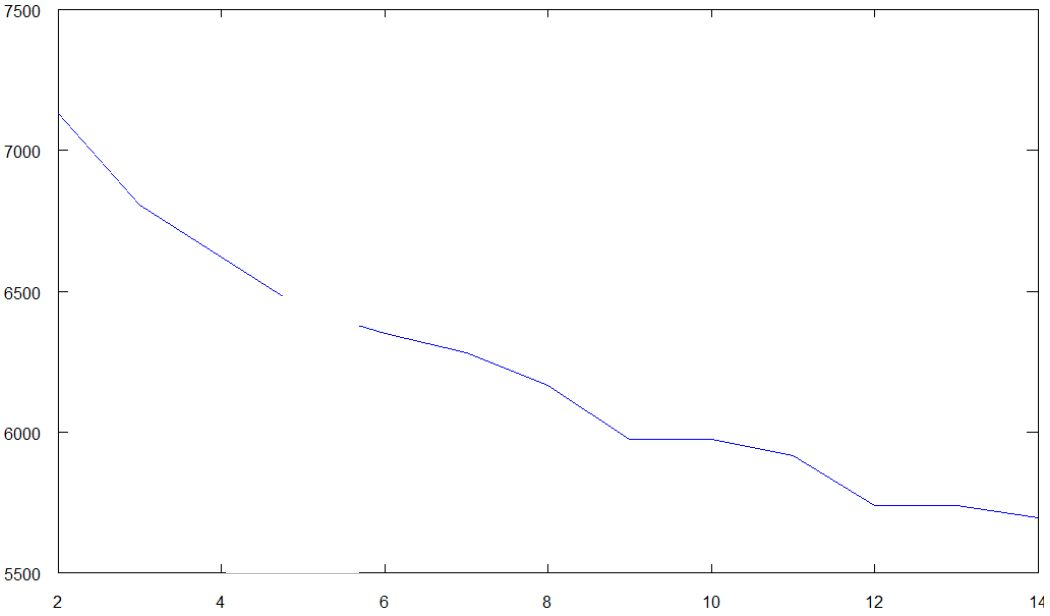


图9 多次运行上述过程的evaluate关于k的变化图

通过我们的多次试验（上面只给出了3次的evaluate变化过程），多次结果的共同特征是，每次k=9~k=10间，evaluate函数的计算值都有极小的变化，而k<9时变化又极大，在k=9之后明显下降变缓。所以，我们k=9或者k=10作为初始的聚类个数，即k-means中的k比较好。根据MDL原则，这里选择k = 9。

上面的例子中以实际的过程提供了一种选择k的思路，就是根据evaluate(C)值下降的“拐点”和MDL原则来选择k。另外，还有许多更科学的方法选择k，见这里：点击这里

我的同类文章

- ssh 免密码设置后，还需要输入密码的问题
- 2.3 - 变量
- 2.5.8 – 函数定义
- 1 - 介绍
- Java视线的blog网址
- SIGH, 半途而废
- 2 - 语言
- 2.4.5 - for 语句
- 牛人

C语言系列之 快速排序与全排列算法　　大数据分析机器学习算法实现的演化

查看评论

2楼 GISQZC 2015-01-07 15:31发表


很不错，就是代码怎么下载不了？

1楼 付 2014-10-15 15:28发表


您好你这个实验的曲线图怎么画出来的？