大作业选题报告

夏武 2016210943 孙禹楠 P16206025 曾华 2016213578

基于金字塔技术的 k 最近邻搜索算法

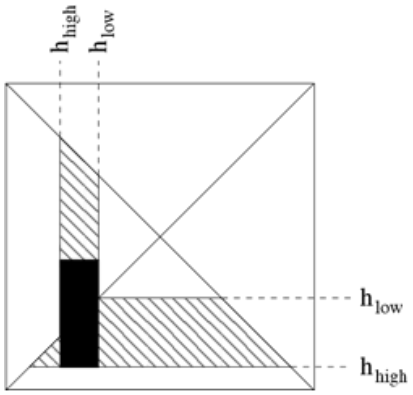
K 最近邻搜索是计算几何中的一个重要问题，应用于很多方面，例如计算机图形学，多媒体数据库，地理信息系统，知识发现和数据挖掘以及计算机辅助设计系统。为了找到查询点的 k 个最近邻居，k 个最近邻居的最大距离是搜索邻居所需的最小半径，我们无法提前知道这个距离。如果采用增加半径的方法：从查询点为中心，一个初始半径很小的球，这个球里包含的可能是 k 个最近邻，然后不断扩大球的半径，直到 k 个备选答案是真正的 k 个最近邻。与这个半径增加的 k 最近邻搜索方法相比，本文实现的是一种减小半径的 k 最近邻搜索算法。

金字塔技术将数据空间[0,1]d 分为二维金字塔。查询球的半径 r 初始化值为 k 个候选最近邻居中最远数据点的距离。以查询点为中心的查询矩形 W，侧边长度为 2r 然后在剩余的 2d-1 金字塔之一中执行正交范围搜索。如果搜索到与查询点距离小于 r 的一些数据点，则更新 k 个候选最近邻居，并且将 r 重置为 k 个候选点的当前最远距离。继续搜索未检查的金字塔，只要发现新的候选点，则半径 r 减小。重复该过程直到检查所有金字塔。这个算法是为金字塔技术设计的，却适用于其他基于映射的索引方案。

算法实现

金字塔技术的基本思想是将 d 维（d-d）数据点转换为 1-d 值，然后使用 B +树存储和访问 1-d 值。数据空间分为两个步骤：首先，将数据空间分为具有数据空间中心点（0.5,0.5，...，0.5）作为其顶部和（d-1）-d 面的 2d 金字塔的数据空间作为它们的基础。然后，每个 2d 金字塔被分成几个分区，每个对应于 B +树的一个数据块。

给定查询矩形 W，正交范围搜索找到与 W 相交的点。金字塔技术将 d-d 范围查询映射为 1d 范围查询的并集。首先我们确定哪个金字塔与 W 相交。然后我们确定交叉金字塔内的哪个金字塔值相交 W.当我们达到 B +树的叶级别时，我们检查存储在叶节点中的数据点是否相交。执行正交范围搜索时所访问的区域如图所示：



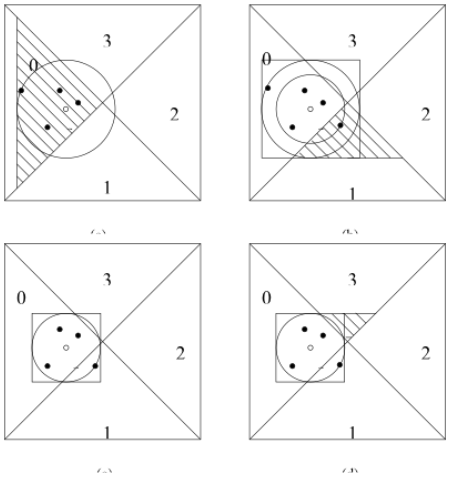
给定一个查询点 q，k 最近邻搜索找到最接近于 q 的 k 个数据点。我们使用列表 A 记录按照与 q 递减顺序排列的 k 个当前候选最近邻居。为了不失一般性，我们使用欧式距离。令 D（v，q）是 d-d 空间中点 v 和点 q 之间的欧式距离，Dmax 是 A 和 q 中数据点之间的最大距离。此外，令 C（q，r）是以 q 为中心的圆，半径为 r。

第一步，我们使用如下的算法确定金字塔 q 和金字塔值 pvq。假设 q 在金字塔 pi 中，使用 LocateLeaf 函数搜索 B+ 树找到键值为 pvq 的叶节点，如果都小于 pvq 则取最大的叶节点。确定叶节点后，我们使用函数 SearchLeft（SearchRight）来检查节点向左（右）的数据点，确定它们是否在 k 个最近邻居中，并相应的更新 A. 注意，如果点 v 与 q 在相同的金字塔，则它们的金字塔值之间的差不大于它们的欧式距离，即| pvq -pvv | ≤D（q，v）。 当叶节点的键值小于（大于）i（i + 0.5）时，SearchLeft（SearchRight）停止，或者 A 中存在 k 个数据点，节点中当前键值与金字塔值之间的差值 q 大于 Dmax。

第二步，我们生成包围 C（q，r）的边长为 Δ= 2r 的查询矩形 W，以执行正交范围搜索，保证查询结果的正确性。如果 A 中存在 k 个数据点，则半径 r 初始化为 Dmax;否则 r 初始化为

√d，使得 C（q，r）覆盖整个数据空间[0,1]d。为了简单起见，我们假设第一步之后在 A 中有 k 个数据点。我们以任意顺序检查金字塔的其余部分。如果金字塔与 W 相交，使用 Interval 算法确定 hlow 和 hhigh，之后执行 RangeSearch 来检查这个金字塔中与 W 相交的数据点是否是 k 个最近邻。W 的中心是固定的，但是在检查金字塔之后每次更新其边长 Δ。如果金字塔不与 W 相交，我们可以修正这个金字塔的搜索。当检查所有金字塔时，k 最近邻搜索停止。

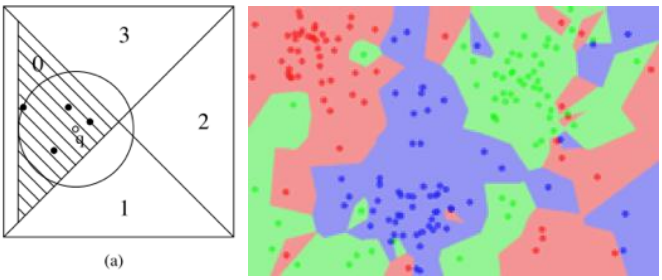
下图示出了 2-d 数据空间中的 k 最近邻搜索示例。这时，最近邻居数 k = 4，表示未填充圆的数据点 q 是查询点。首先，我们可以确定 q 在金字塔 p0 中，然后我们使用区间[0,0.5]搜索 B+ 树，在 p0 中找到 q 个候选，并将它们存储在列表 A 中，即数据点在 C（q，r）中，如图（左上）所示，其中 r 是 A 中最远数据点与 q 的距离（我们假设 A 中有 k 个数据点）。其次，生成包围 C（q，r）的查询矩形 W。我们以逆时针顺序检查其余的金字塔。图（右上）中金字塔 p1 的交叉阴影区域是当我们在 B +树上执行 1-d 范围搜索时的搜索区域。找到一个候选，并更新 A，r 也是如此。查询矩形 W 被更新以包围更新的 C（q，r）。当找到更接近 q 的数据点时，我们搜索半径较小，搜索代价更低。因为 W 和金字塔 p2 之间没有交点（左下），所以我们不需要检查这个金字塔的数据点。当我们到达金字塔 p3 时，我们检查图（右下）中交叉阴影区域的数据点，发现它们都不在 k 个最近邻居之间。在检查 4 个金字塔后，我们得到结果，即图（右下）中的 C（q，r）中除 q 之外的数据点。



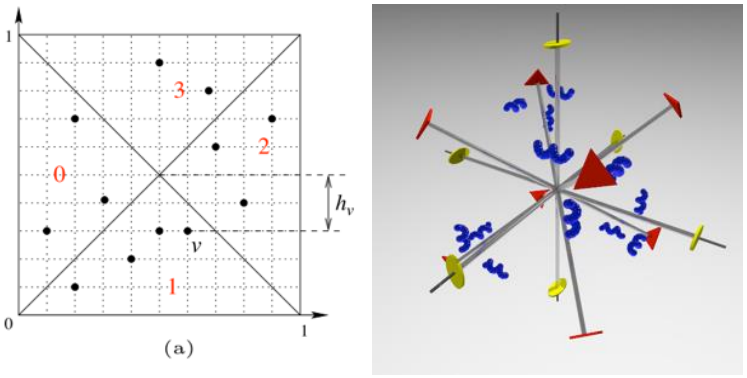
展示方案

低维数据可视化

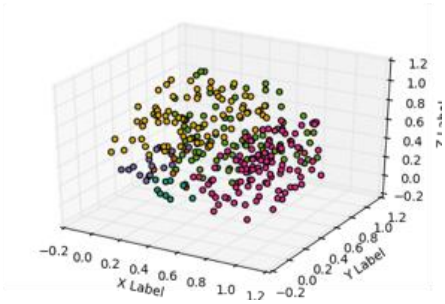
k-NN 通常在二维平面上展示如原论文的 2 维展示或来源于维基百科的分类结果，如下。



对于二维和三维空间，2d 的金字塔 id 可以直接显示，二维的每一个金字塔为三角形，三维的金字塔为对角线相交形成的四棱柱（以红色三角形状标注对角线为轴），如下：



对于三维，对金字塔 ID 测试如下：



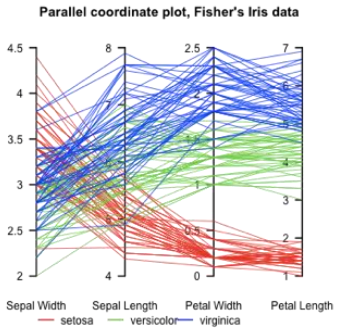
可以使用所属空间标注金字塔 ID，在搜寻时，高亮显示搜索起点和当前搜索点。

高维数据可视化

点在高维空间的位置不能直观显示，通用的方法有 Parallel Coordinates、Line Graph 等。

其中 Parallel Coordinates 非常适合本次可视化。

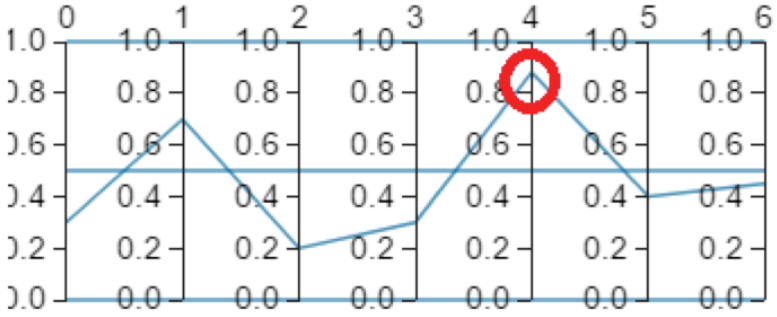
Parallel Coordinates 是将相互垂直的轴，改变为平行，从而达到可以显示高维数据。 其中，每一个点数据，转变为线，线与每一个轴的交点为对应数据。如下图（维基百科）。



关键的点为 Parallel Coordinates 与 2d 金字塔编号相互匹配。取（0.5, 0.5, 0.5, …, 0.5）为界，自然将空间分解为 2d 个部分。

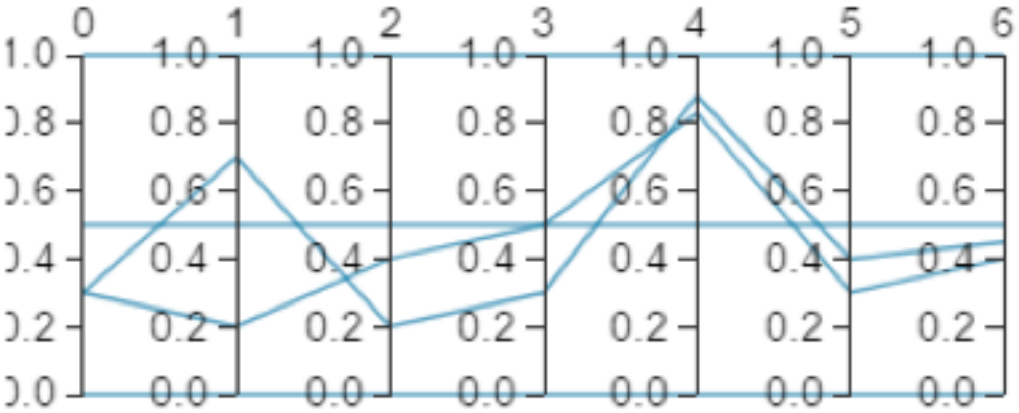
金字塔 ID 选择：

从图上可以看出，将与（0.5, 0.5, 0.5, …, 0.5）的距离分解后，最大值所在区域为其编号，最大距离+编号为其索引值。



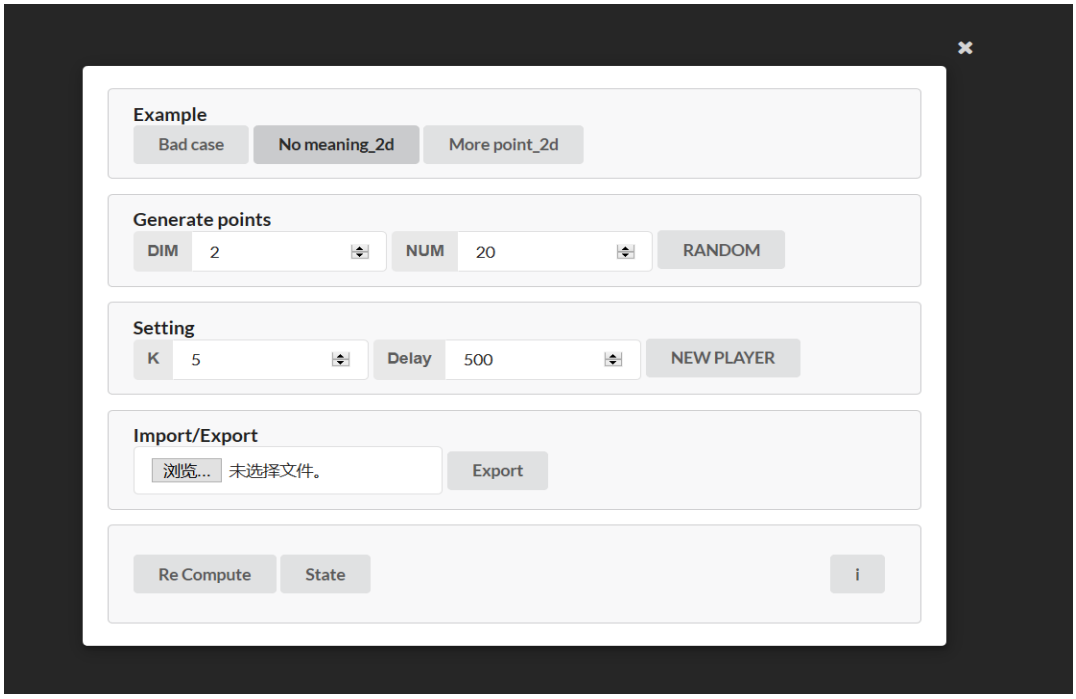
搜索过程：

在金字塔 ID 其轴与线相交附近开始搜索 KNN，可以得到初始搜索边界，显示为该轴上的高亮区域，从而开始便利各个金字塔内部是否有更近的点。

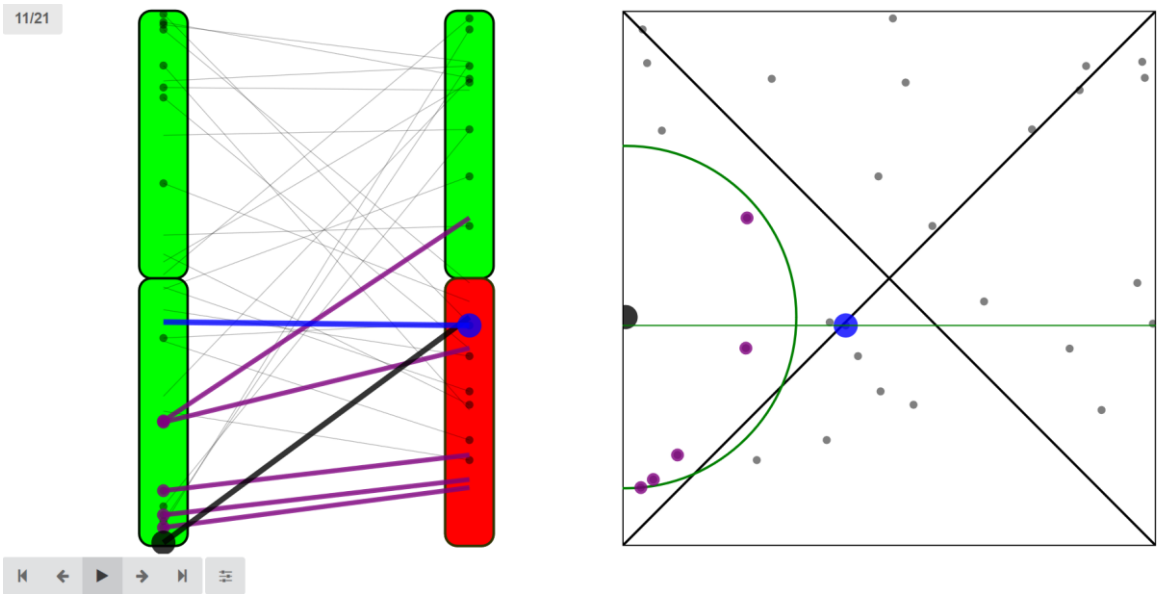


演示程序说明

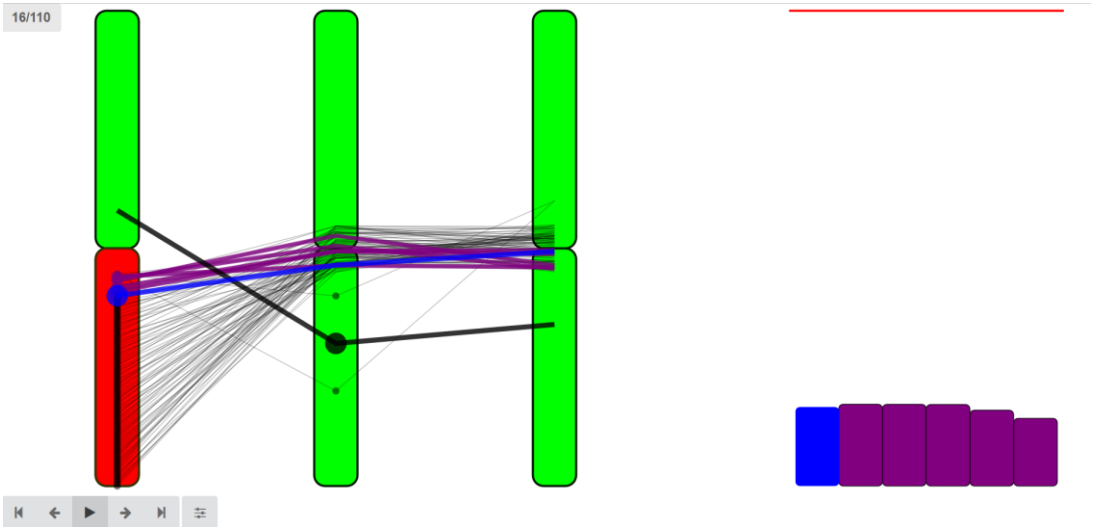
在二维的情况下，准备了两个默认的例子，展示了少数点和多数点的搜索过程。



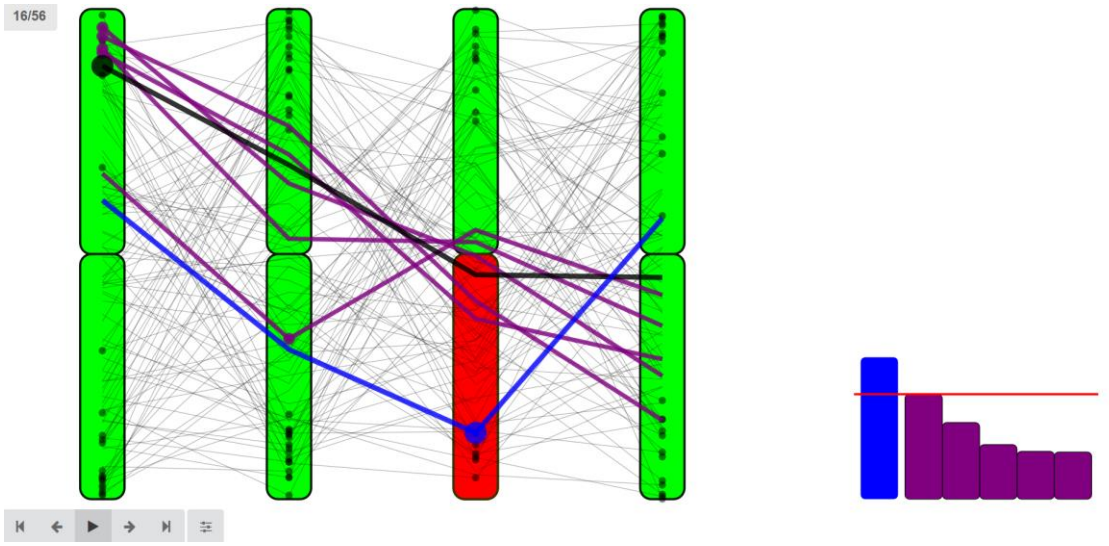
下图展示的是二维且点数不多时的搜索过程，蓝色的点是正在搜索的点，紫色是已经完成搜索的点。在右侧三角形区域可以看到平面上的效果。



下面分别是最坏情况和一个四维情况下的例子。



每当正在搜索的蓝点符合搜索的条件，便替换为确认的紫点，右侧的紫色柱形显示了候选邻居同中心点的距离，当蓝色点距离更近时便替换掉过去最远的点。



在不同数据量，维度，K 取值下的性能如下所示

size = 1000000	dim = 10	k = 10	building time:149.105	query time for 10000 quries:78.402
size = 100000	dim = 20	k = 10	building time:12.403	query time for 10000 quries:93.467
size = 100000	dim = 10	k = 10	building time:12.349	query time for 10000 quries:44.701
size = 100000	dim = 10	k = 20	building time:12.312	query time for 10000 quries:44.863
size = 100000	dim = 20	k = 20	building time:12.54	query time for 10000 quries:92.415

参考文献

[1] Nickerson, Bradford G., and Qingxiu Shi."K-Nearest Neighbor Search using the Pyramid Technique." In CCCG. 2006.

[2] S. Berchtold, C. B¨ohm, and H.-P. Kriegel. The Pyramidtechnique: Towards breaking the curse of dimensionality. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pages 142– 153, Seattle, Washington, USA, June 2-4 1998.