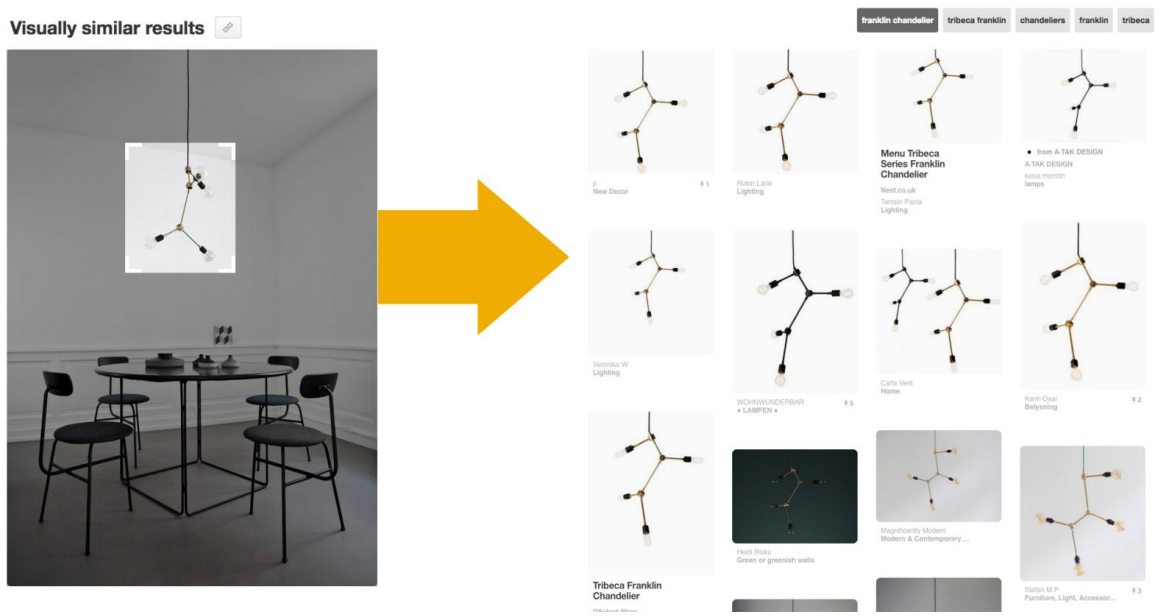
邻近搜索

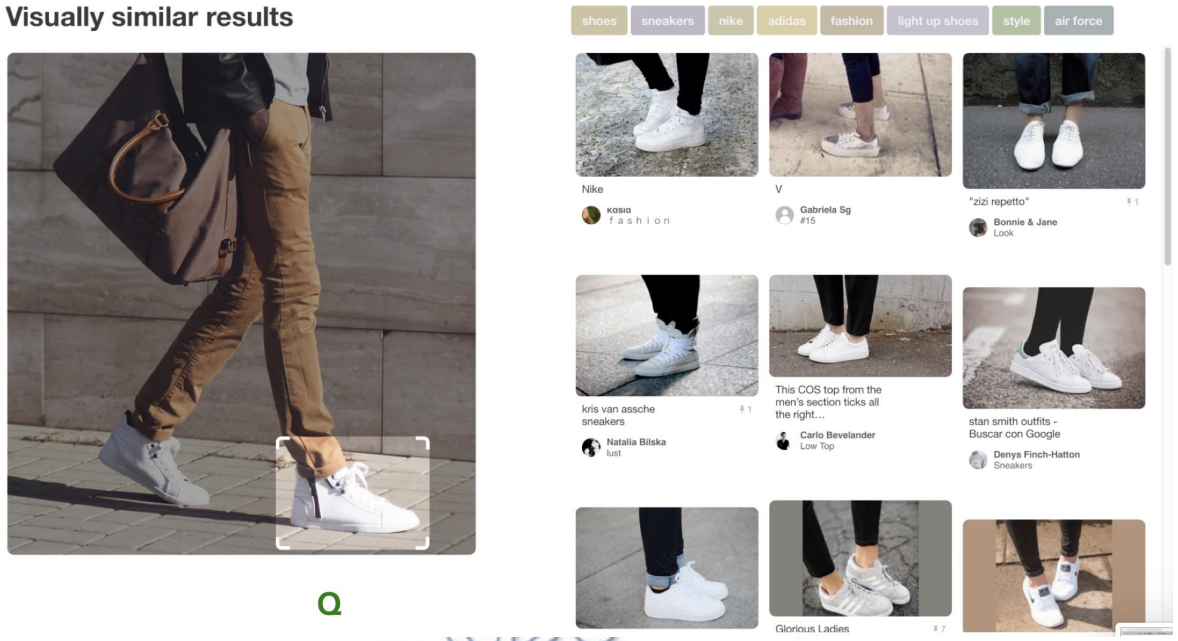
汇报人：扣扣

大纲

Annoy：Approximate Nearest Neighbors Oh Yeah
HNSW：Hierarchcal Navigable Small World graphs
KD Tree：K dimentional Tree
LSH：Locality Sensitive Hashing

大数据情境下的邻近搜索

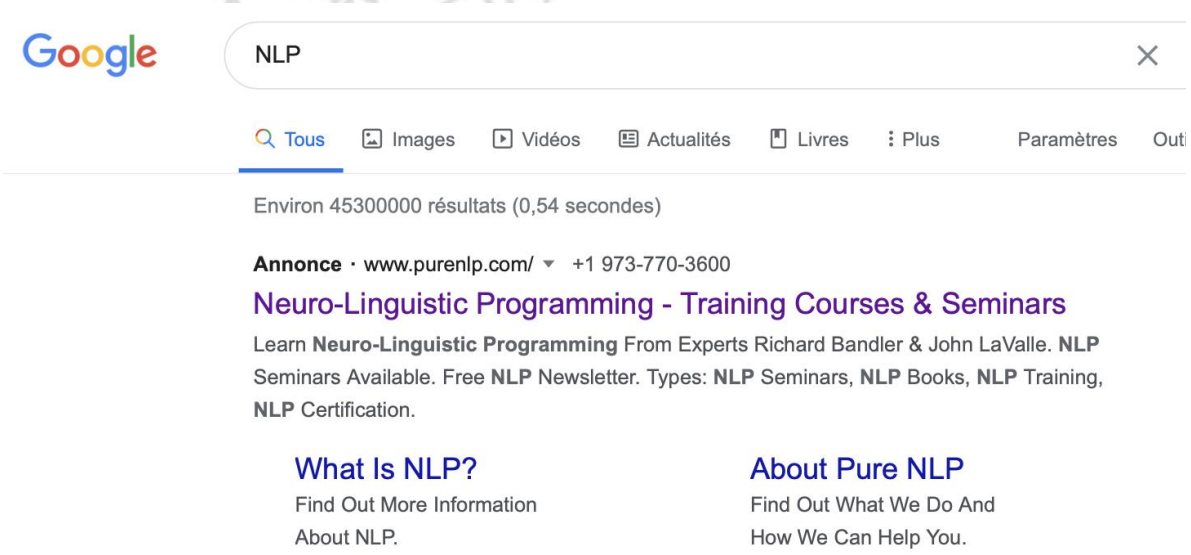




大数据情境下的邻近搜索

大数据情境下的邻近搜索

京东智联云 & 贪心科技



大数据情境下的邻近搜索

如何从海量文本中快速查找出相似文本?

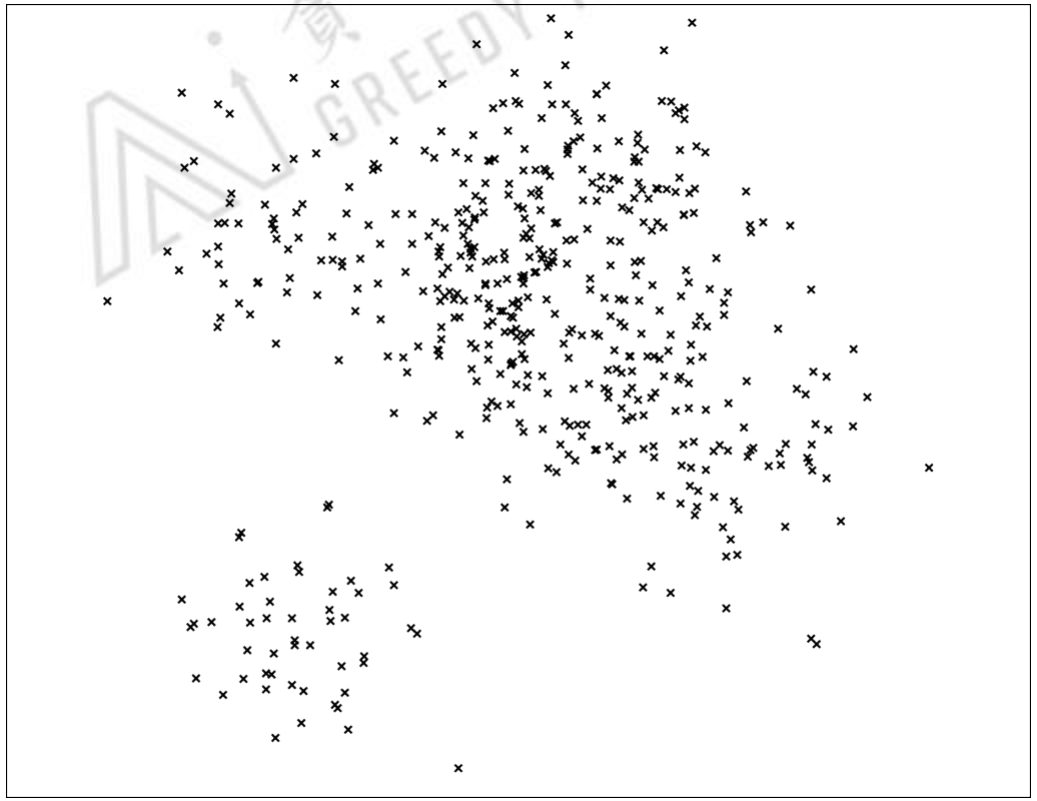
Annoy:Approximate Nearest Neighbors Oh Yeah

如何从海量文本中快速查找出相似的Top N 文本?

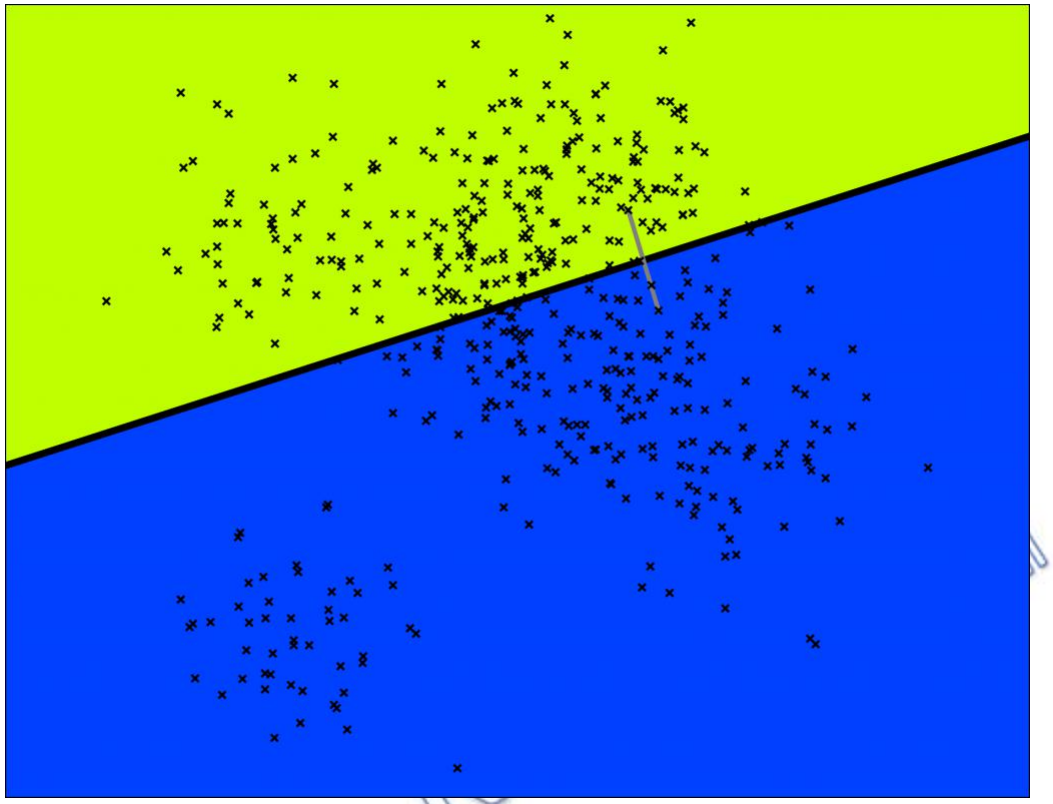
Annoy 是 Spotify 开源的高维空间求近似最近邻的库，在 Spotify 使用它进行音乐推荐。

Annoy通过将海量数据建立成一个二叉树来使得每个数据查找时间复杂度是O(log n)。

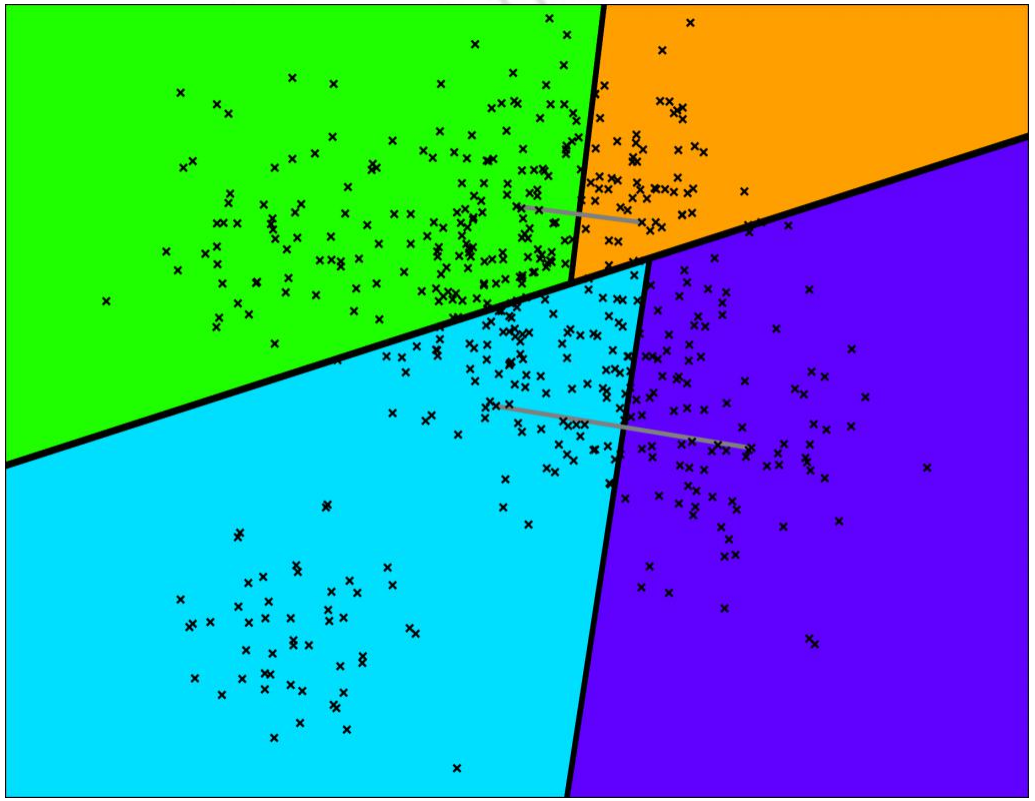
Annoy:Approximate Nearest Neighbors Oh Yeah

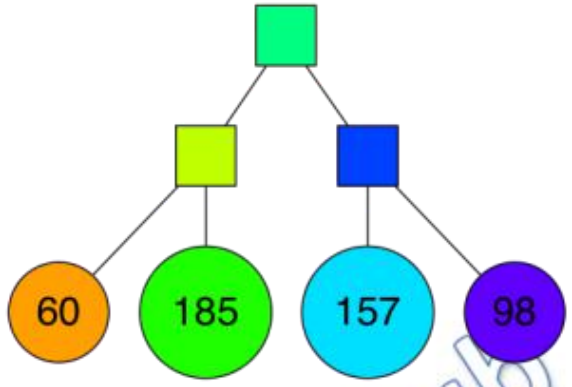


随机取两点进行切分

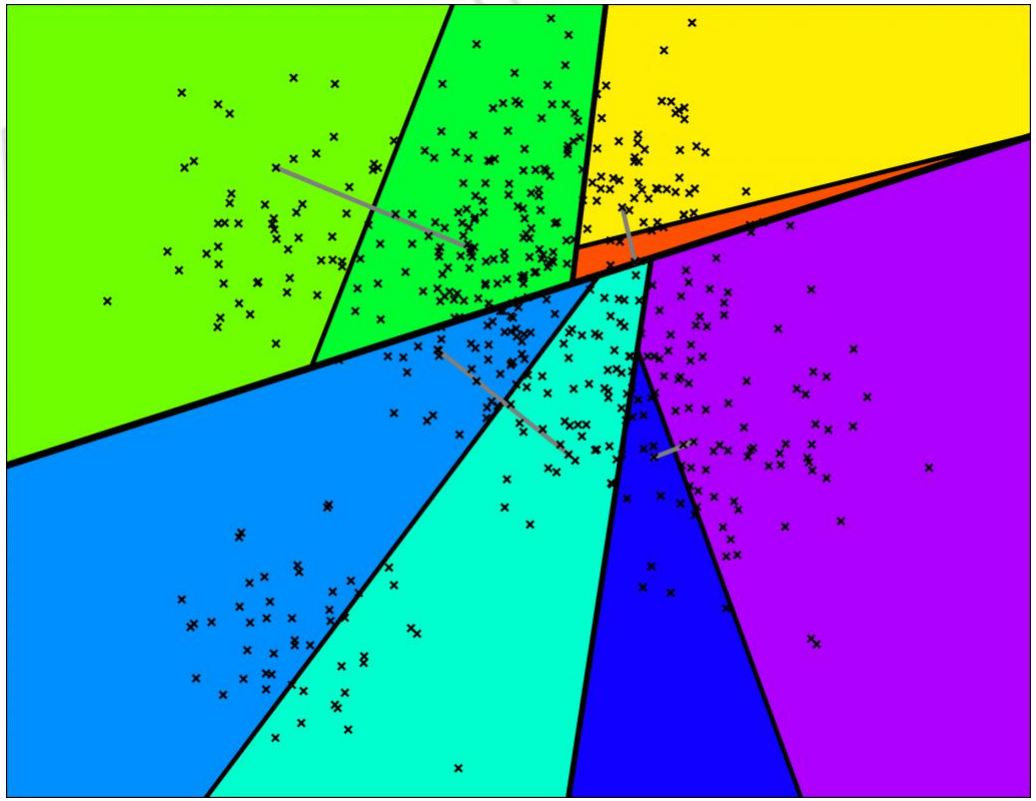


Annoy:Approximate Nearest Neighbors Oh Yeah

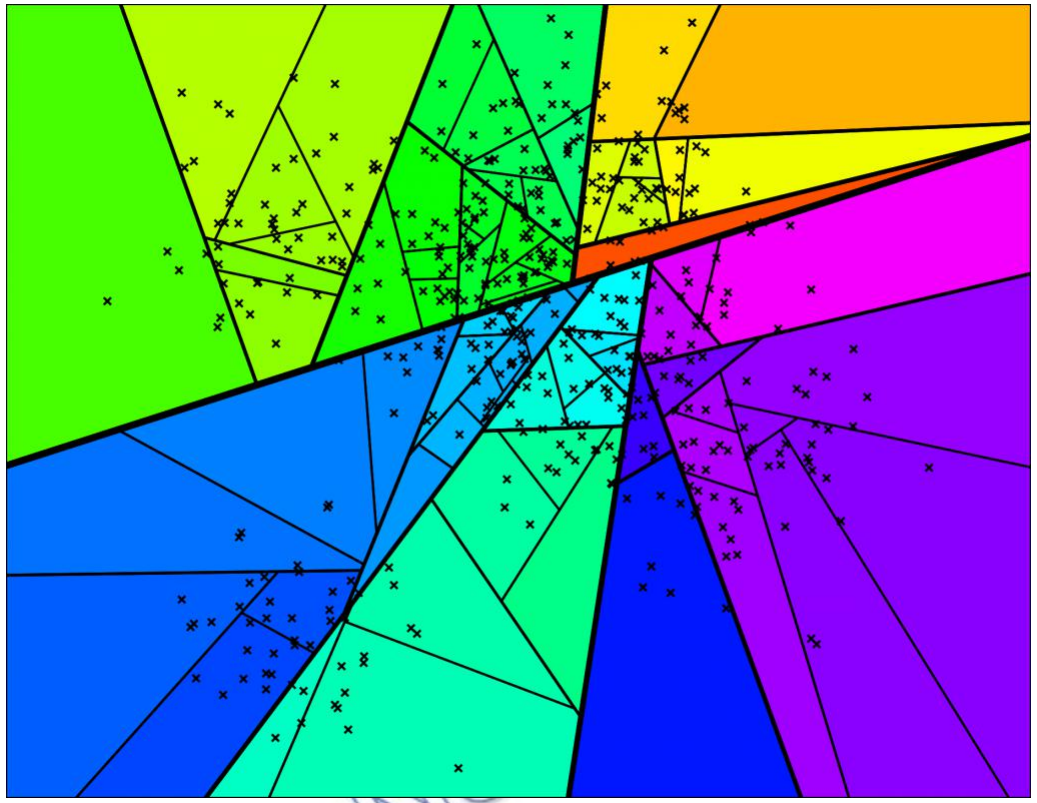




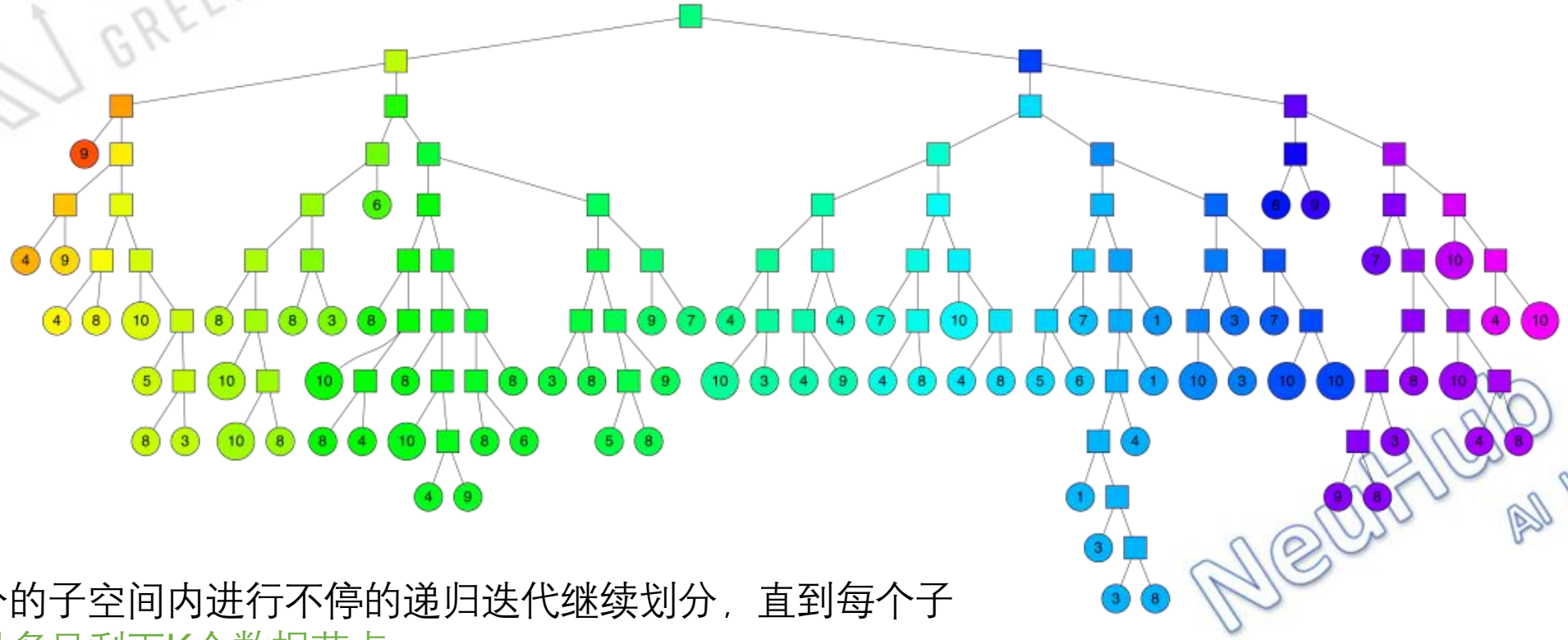
Annoy:Approximate Nearest Neighbors Oh Yeah



继续

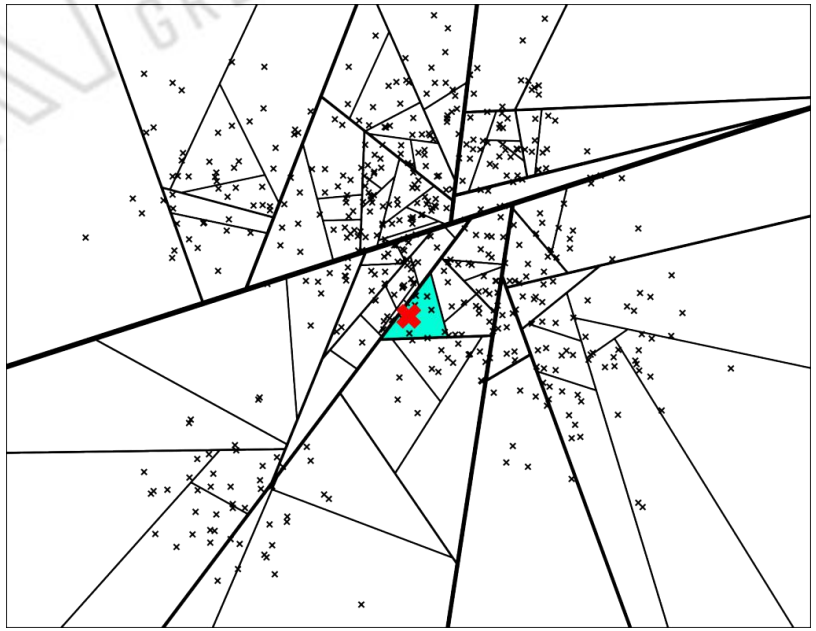


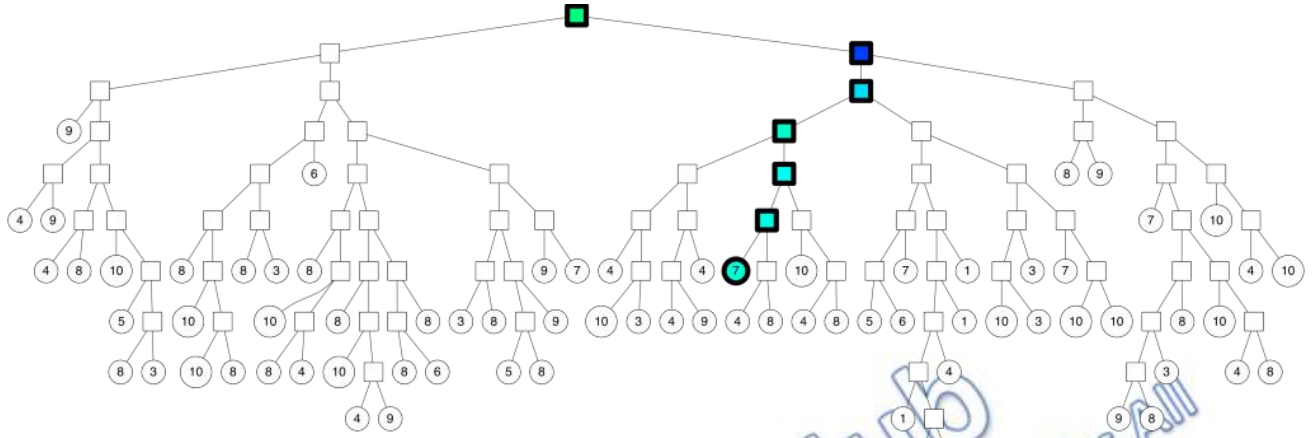
Annoy:Approximate Nearest Neighbors Oh Yeah



在划分的子空间内进行不停的递归迭代继续划分，直到每个子空间最多只剩下K个数据节点。

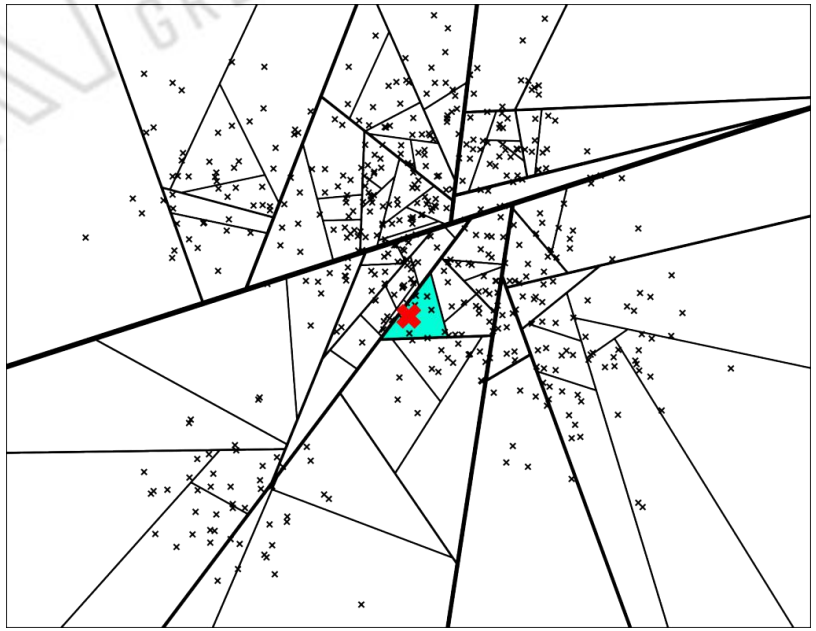
Annoy:Approximate Nearest Neighbors Oh Yeah

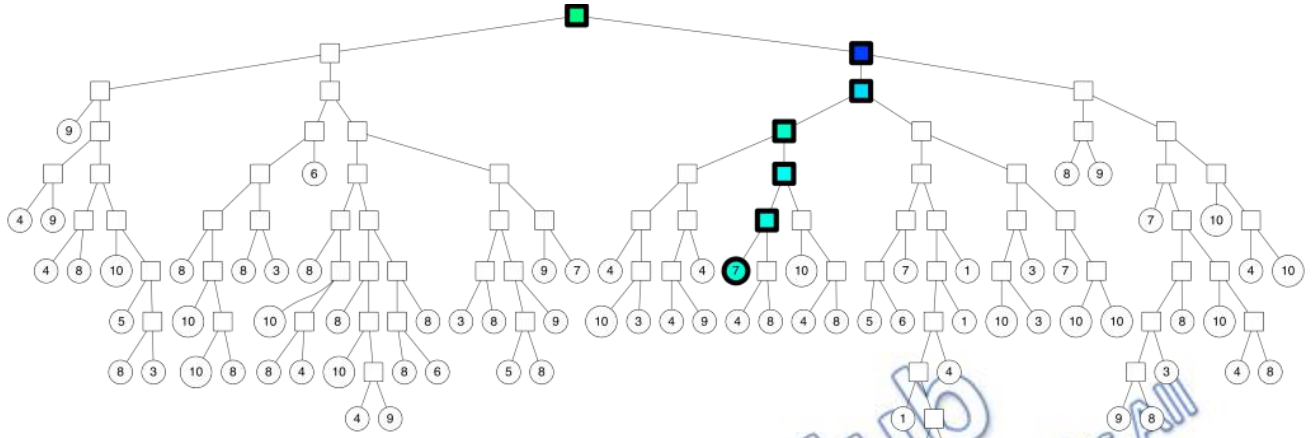




查找：不断看查看此点在分割超平面的哪一边。
从二叉树索引结构来看，即从根节点不停地往叶子节点遍历的过程。

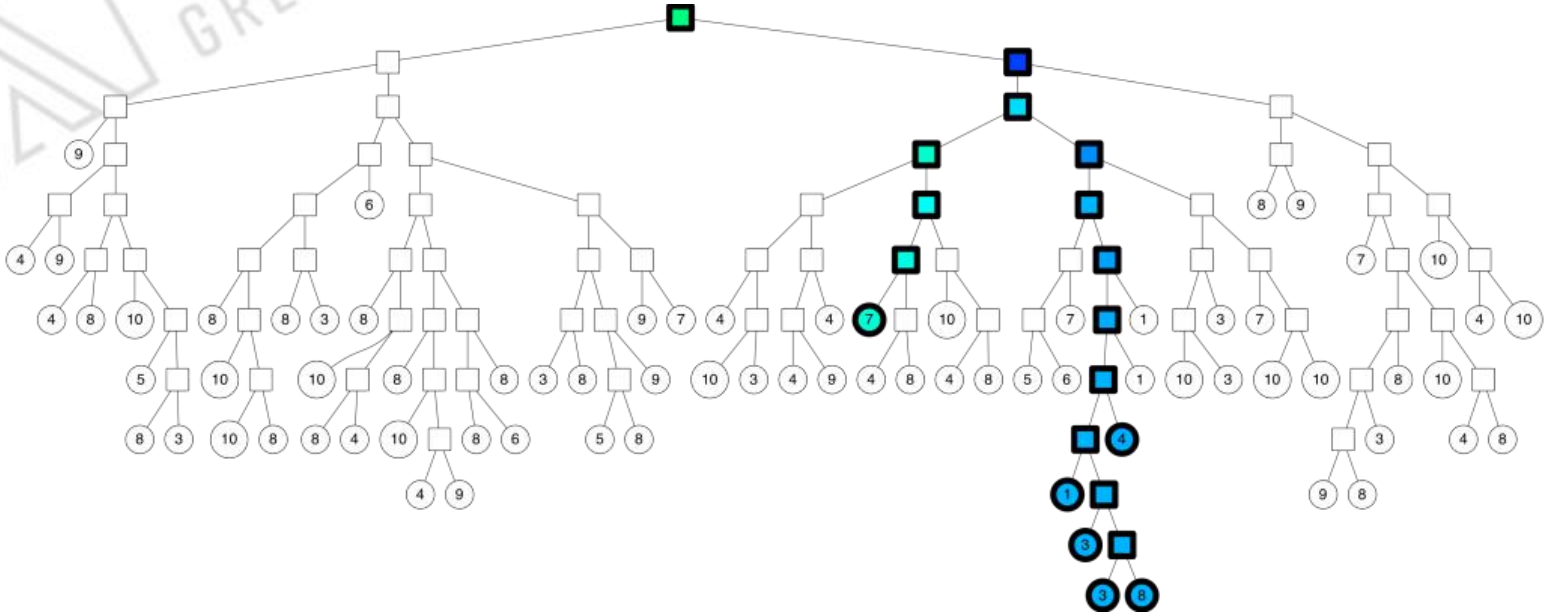
Annoy:Approximate Nearest Neighbors Oh Yeah

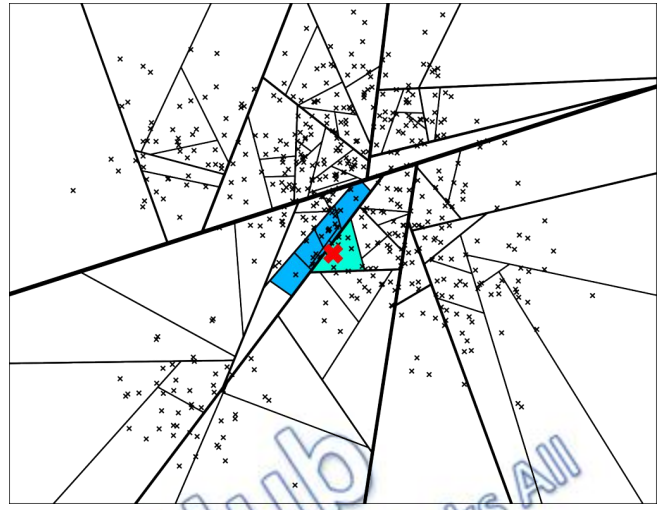




- 查询过程最终落到叶子节点的数据节点数小于我们需要的Top N相似节点数目怎么办？
- 两个相近的数据节点划分到二叉树不同分支上怎么办？

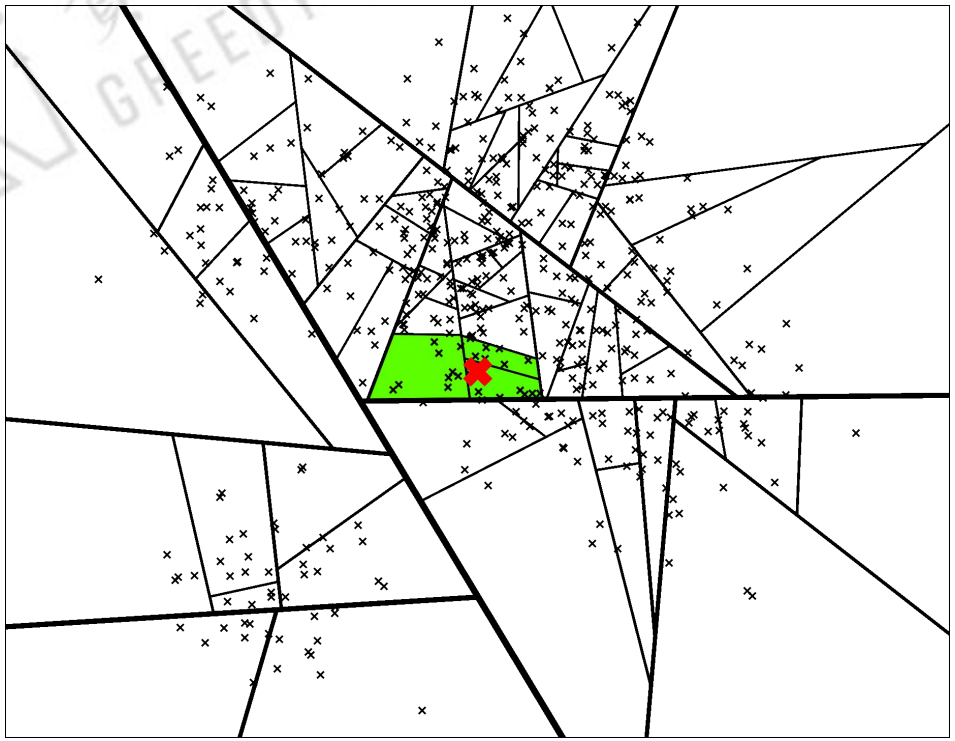
Annoy:Approximate Nearest Neighbors Oh Yeah

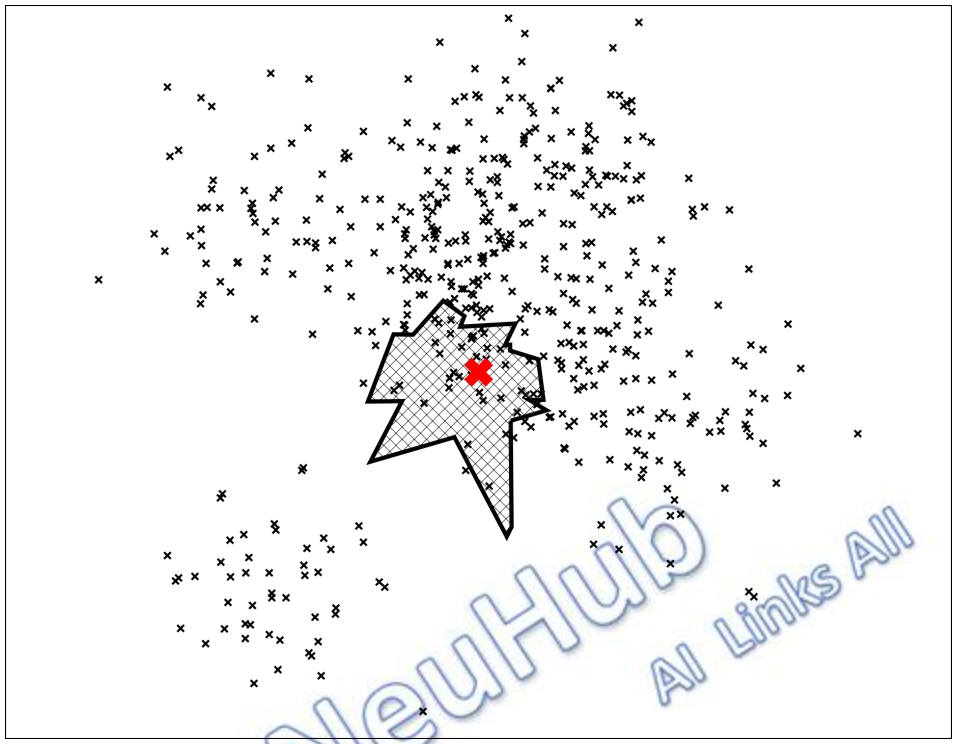




trick1： 两边都遍历

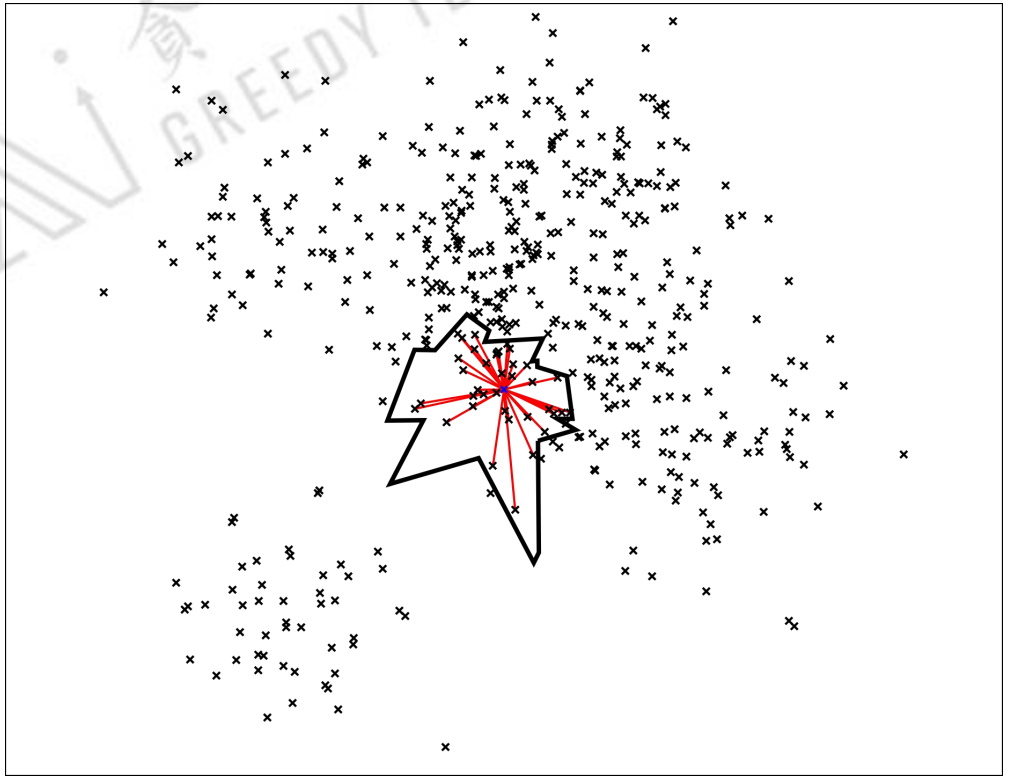
Annoy:Approximate Nearest Neighbors Oh Yeah

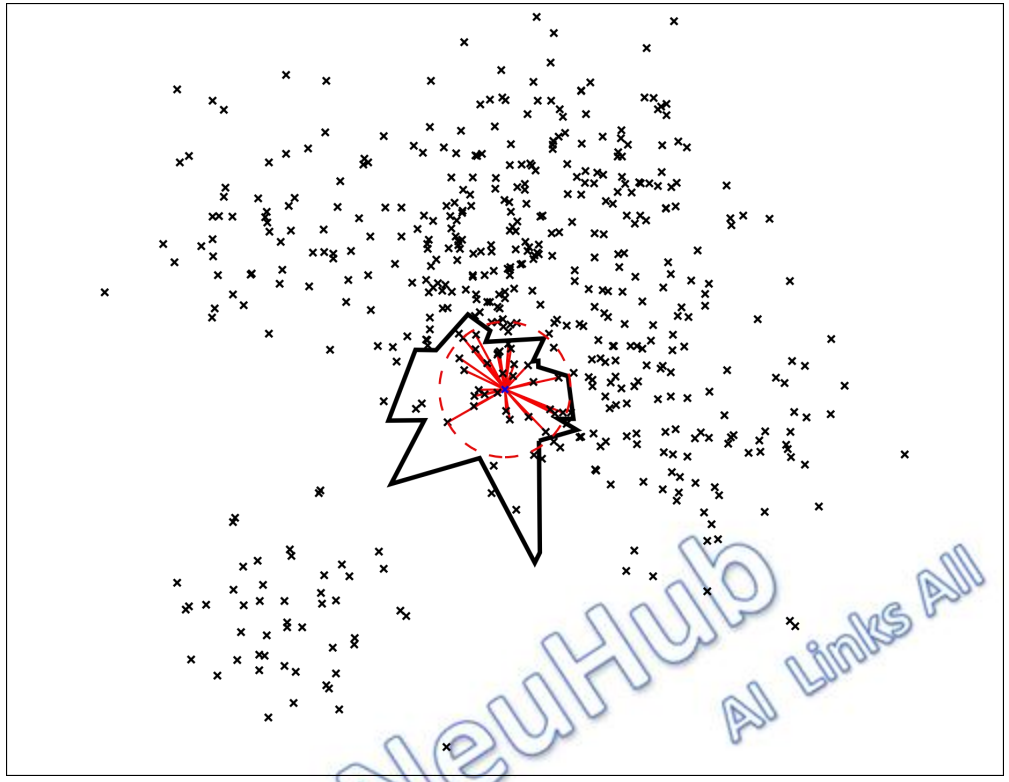




trick2： 多棵数

Annoy:Approximate Nearest Neighbors Oh Yeah





trick2： 多棵数

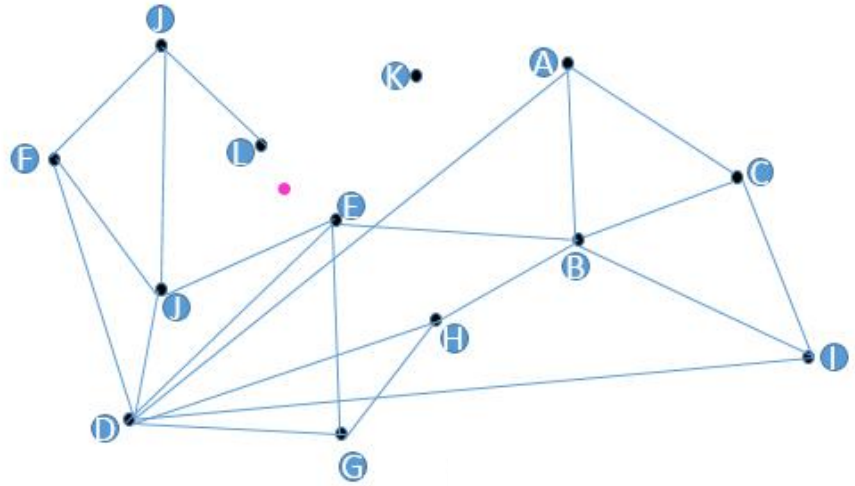
Annoy:Approximate Nearest Neighbors Oh Yeah

(实例)腾讯词向量实战,通过Annoy进行索引和快速查询:
https://www.52nlp.cn/腾讯词向量实战-通过annoy进行索引和快速查询

HNSW：Hierarchcal Navigable Small World graphs

近邻图(Proximity Graph)：最朴素的图算法

思路：构建一张图，每一个顶点连接着最近的 N 个顶点。Target （红点）是待查询的向量。在搜索时，选择任意一个顶点出发。首先遍历它的友节节，找到距离与 Target 最近的某一节点，将其设置为起始节点，再从它的友节点出发进行遍历，反复迭代，不断逼近，最后找到与 Target 距离最近的节点时搜索结束。

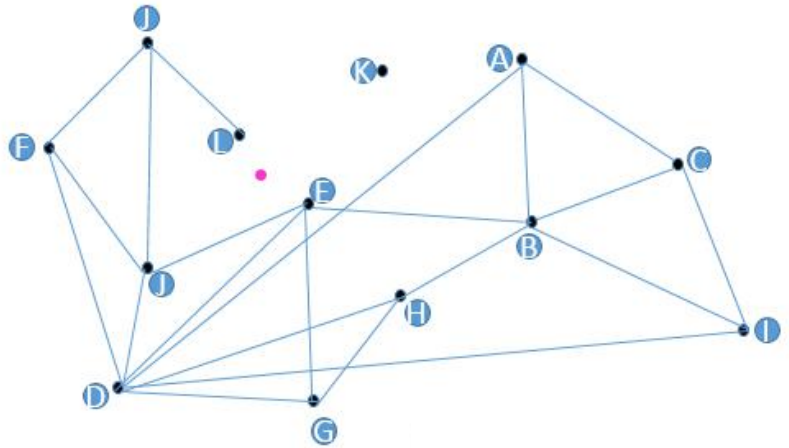


HNSW：Hierarchcal Navigable Small World graphs

近邻图(Proximity Graph)：最朴素的图算法

问题：

- 图中的K点无法被查询到。
- 如果要查找距离粉色点最近的topK个点，而如果点之间无连线，将影响查找效率。
- D点有这么多友节点吗？增加了构造复杂度。谁是谁的友节点如何确定？
- 如果初始点选择地不好（比如很远），将进行多步查找。

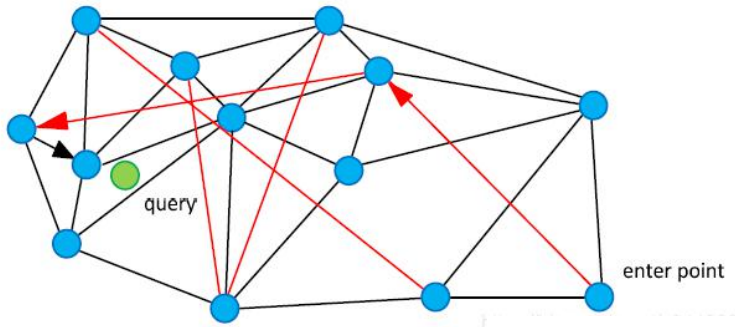


HNSW：Hierarchcal Navigable Small World graphs

NSW(Navigable-Small-World-Graph)

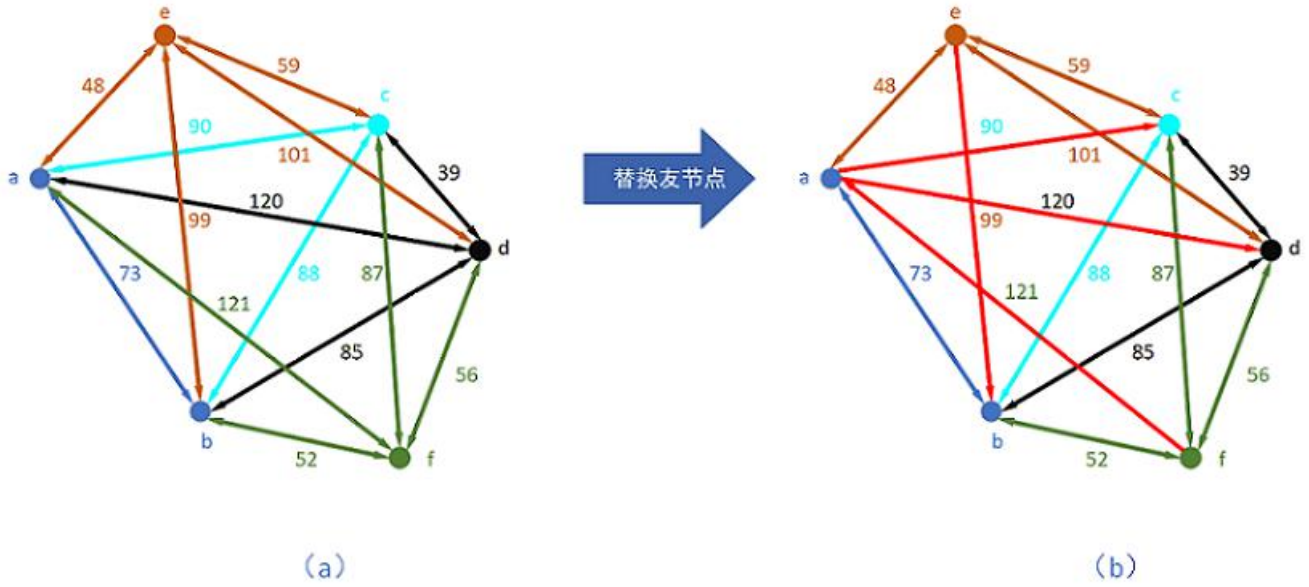
解决：

- 某些点无法被查询到 -> 规定构图时所有节点必须有友节点。
- 相似点不相邻的问题 -> 规定构图时所有距离相近到一定程度的节点必须互为友节点。
- 关于某些点有过多友节点的总是 -> 规定限制每个节点的友节点数量。
- 初始点选择地很远 -> 增加高速公路机制。



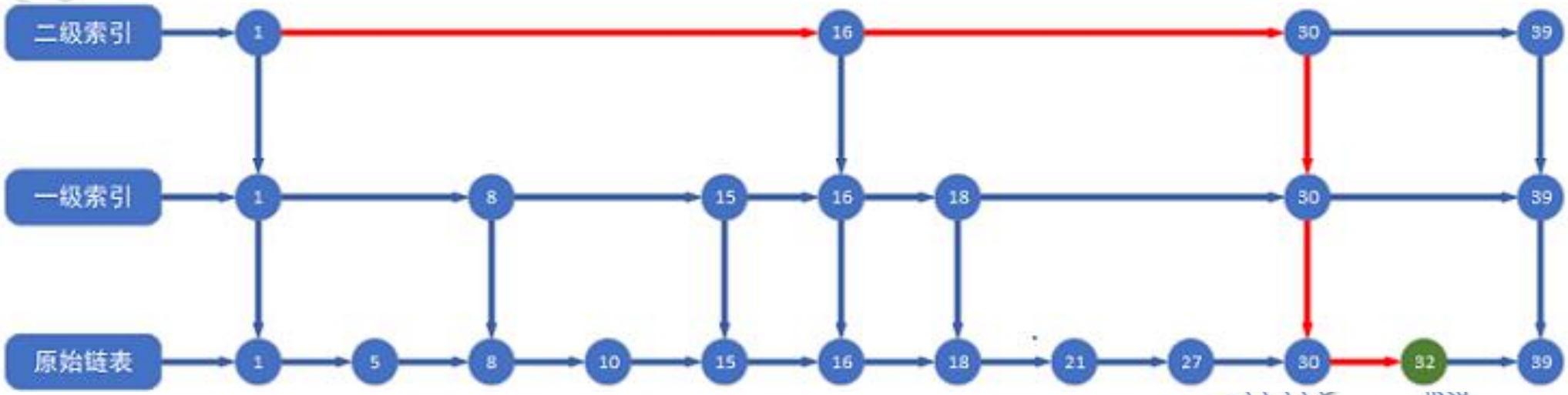
HNSW：Hierarchcal Navigable Small World graphs

建图过程（规定最多四个友节点）：
节点的友节点在新的节点插入的过程中会不断地被更新。



HNSW：Hierarchcal Navigable Small World graphs

如何增加高速公路机制？

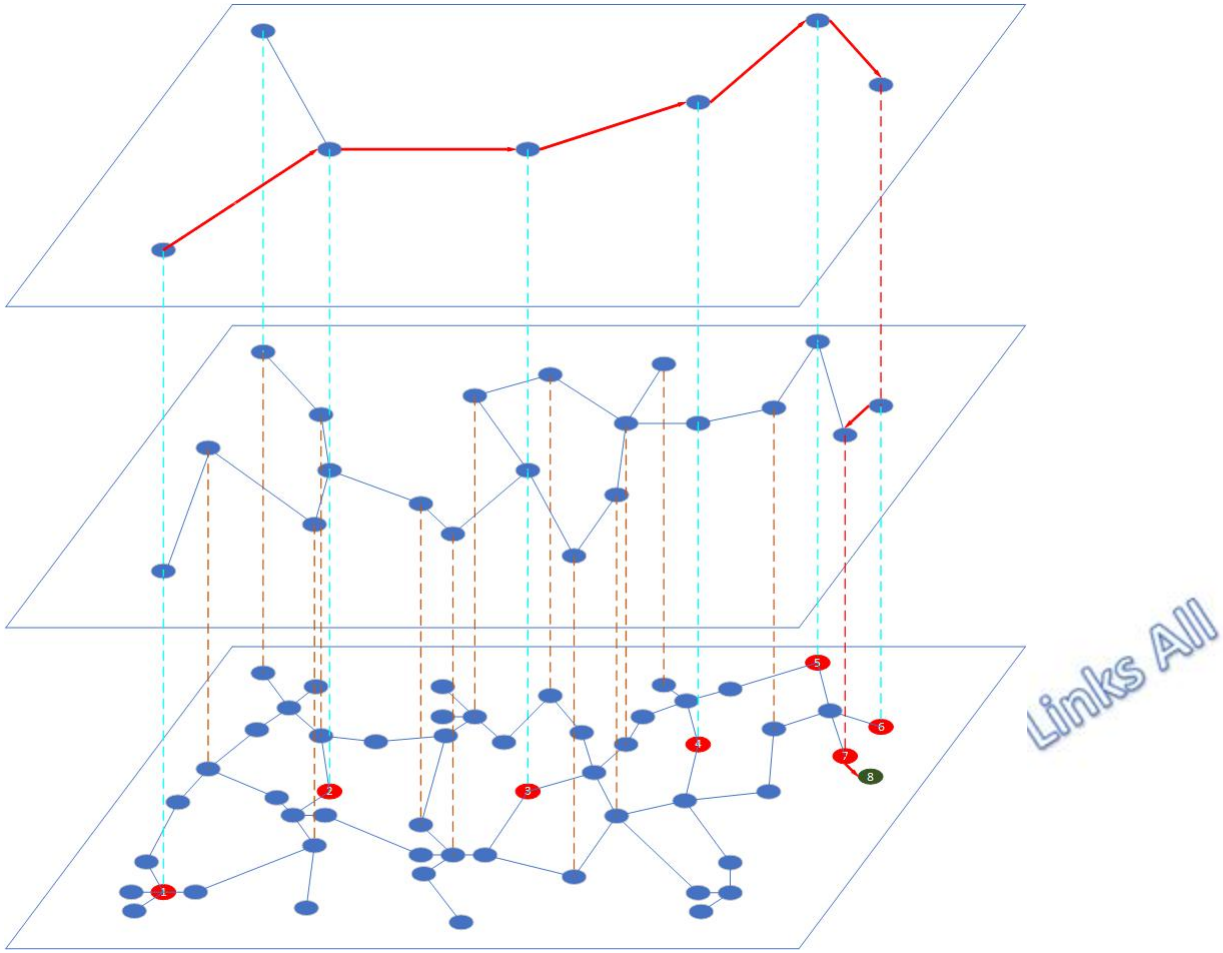


HNSW：Hierarchcal Navigable Small World graphs

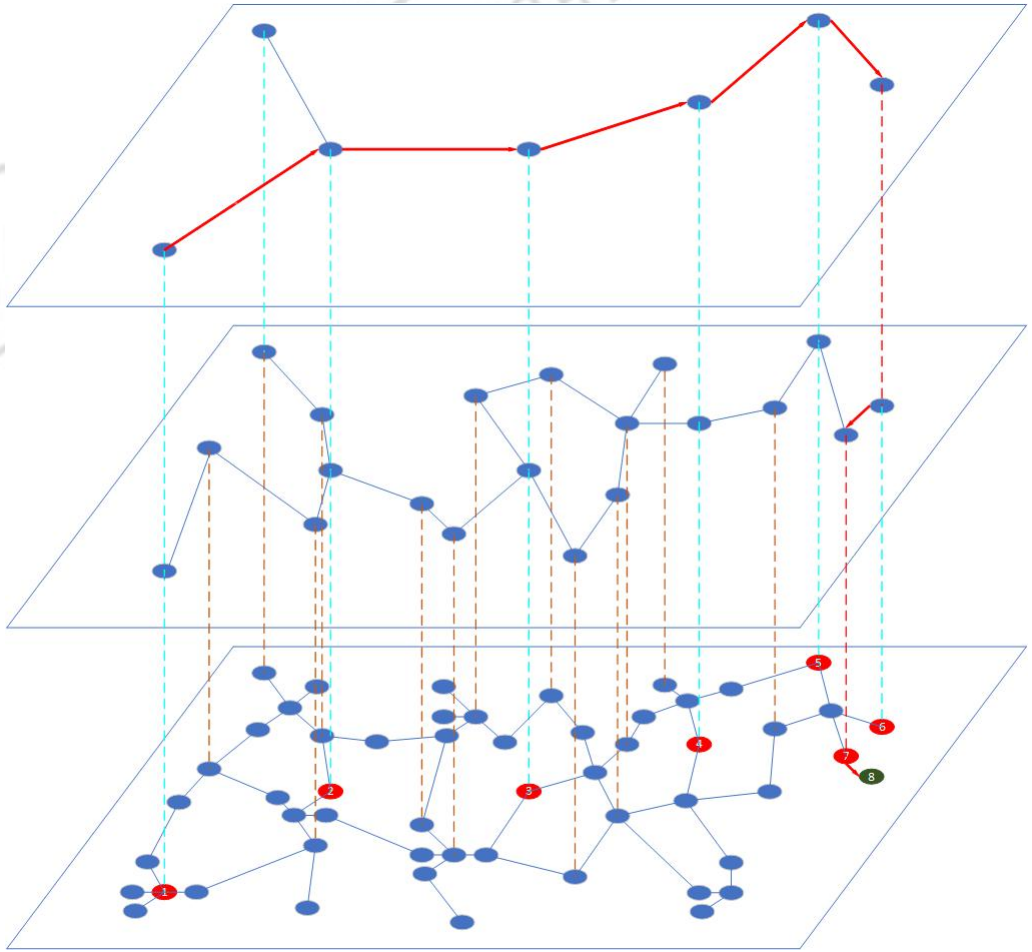
如何增加高速公路机制？

HNSW = NSW + 跳表

code:
https://github.com/nmslib/hnswlib



HNSW：Hierarchcal Navigable Small World graphs



飞机
地铁
步行

京东智联云 & 贪心科技

休息

KD Tree：K dimentional Tree

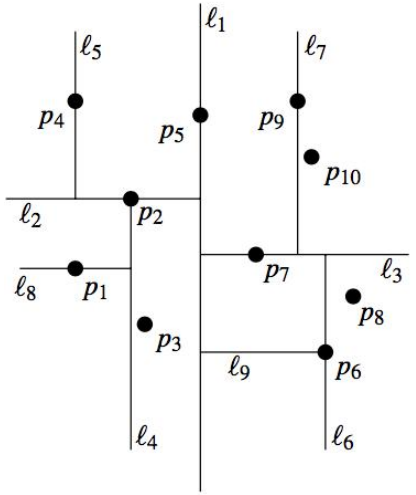
平衡二叉树

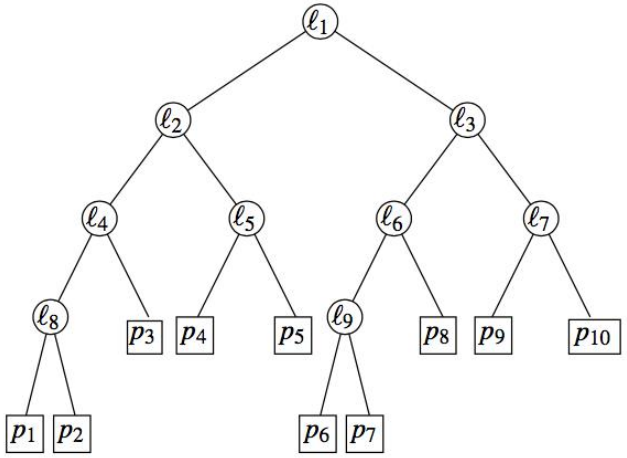
- 又称AVL树，指的是左子树上的所有节点的值都比根节点的值小，而右子树上的所有节点的值都比根节点的值大，且左子树与右子树的高度差最大为1。
- 要查找一个值，不需要遍历整个序列（或者说遍历整棵树），可以根据当前遍历到的结点的值来确定搜索方向。
- 这种思维在搜索中叫做“剪枝”，把不必要的分枝剪掉可以提高搜索效率。

KD Tree：K dimentional Tree

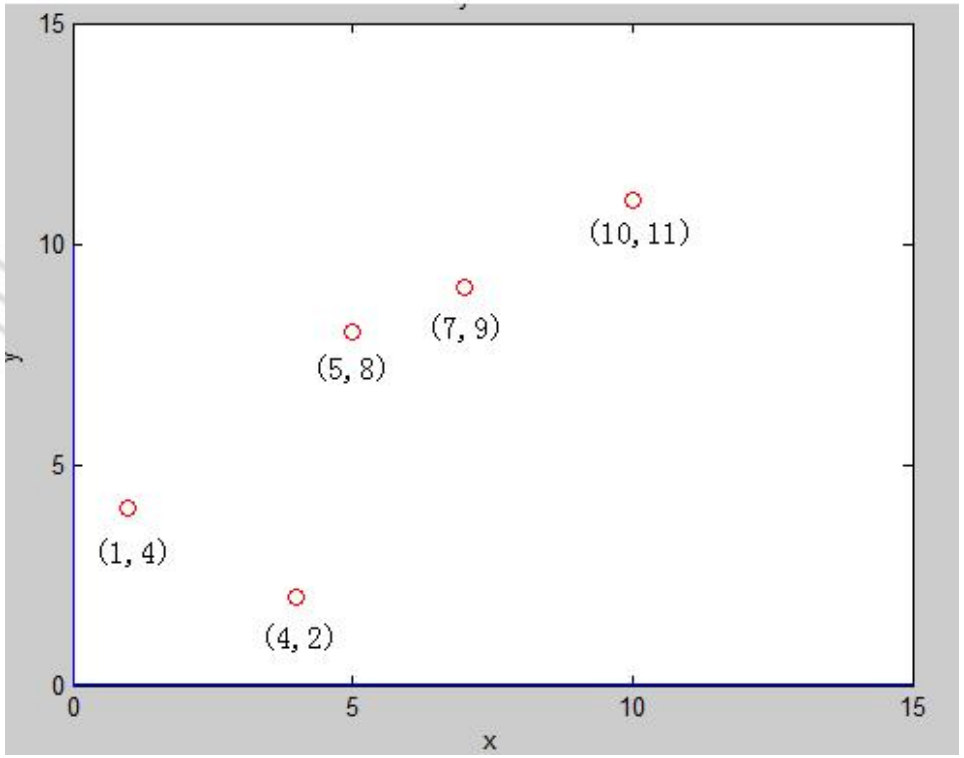
KD Tree

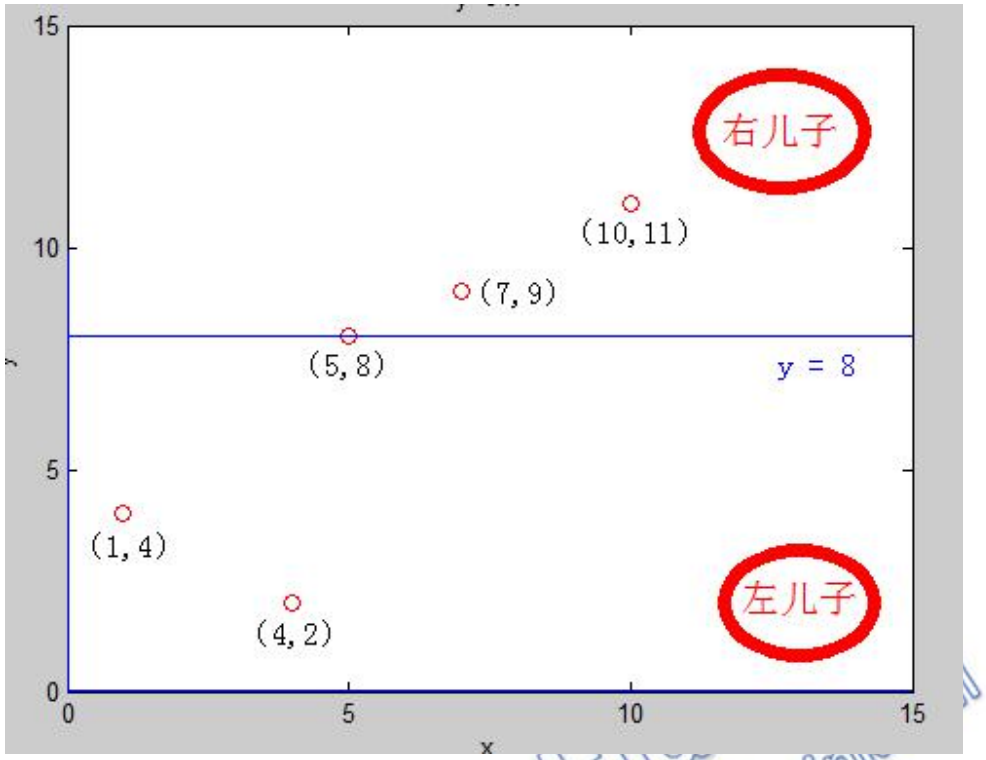
- K：K邻近查询中的k；D：空间是D维空间（Demension）tree：二叉树
- K-D tree的建立就是分裂空间（分成两堆）的过程。
- 计算每个点的坐标的每一维度上的方差，取方差最大的那一维对应的中间值，作为分裂点。
- 以此类推，直到每个空间中最多有一个点。





KD Tree：K dimentional Tree

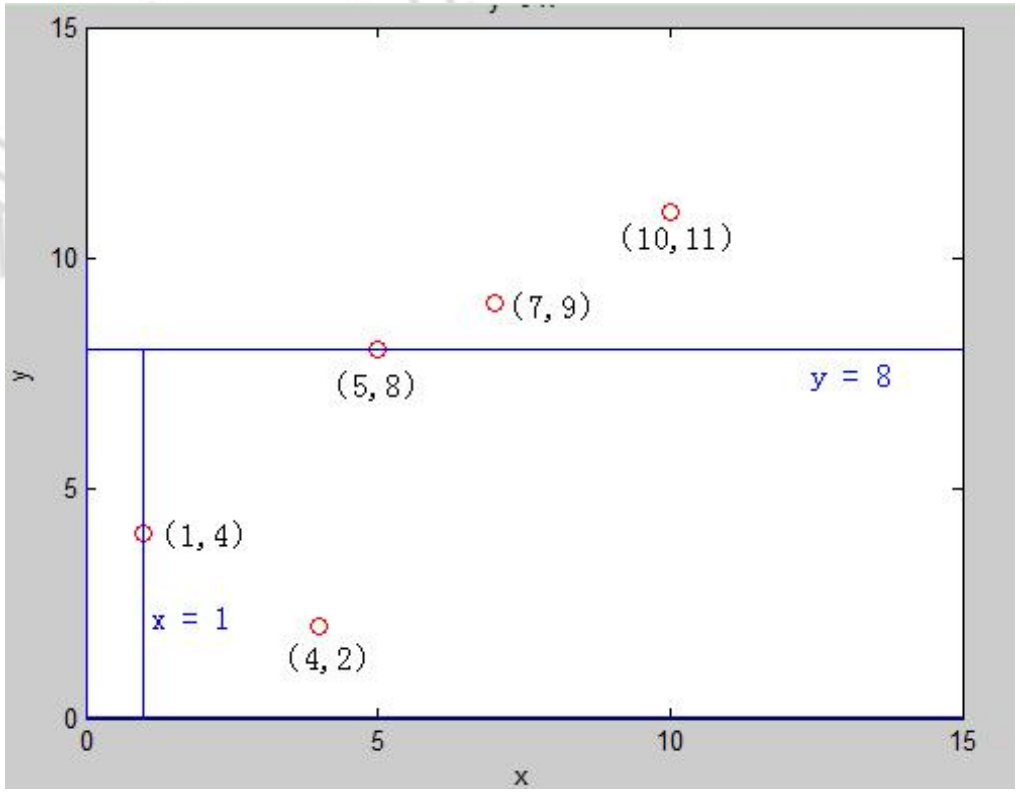


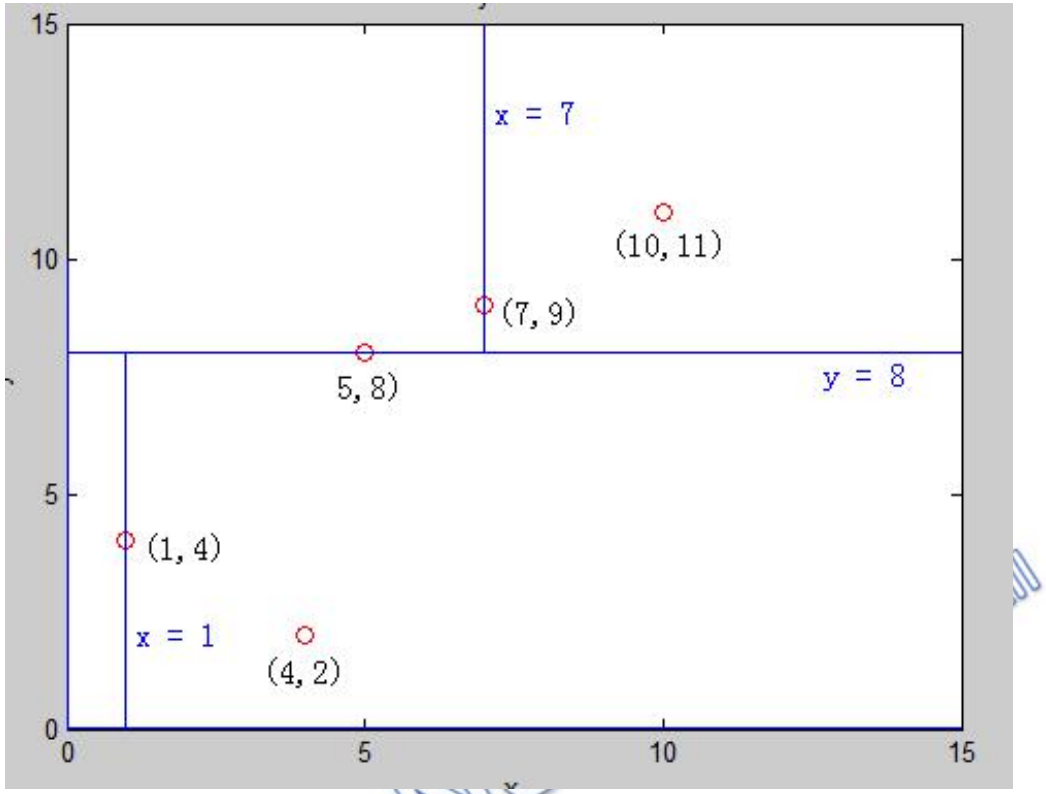


平均值： ave_1 =5.4
方差： varance_1 =9.04

平均值：ave_2 =6.8
方差：varance_2 =10.96

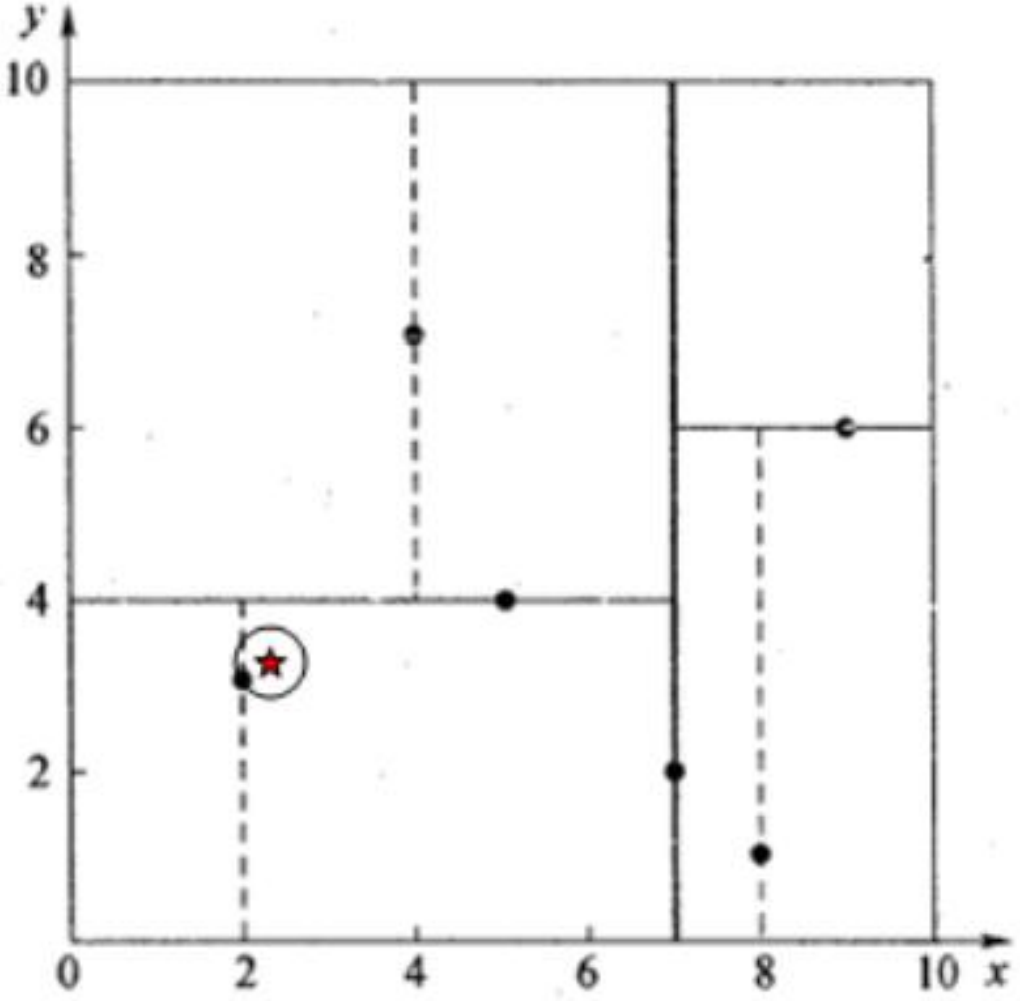
KD Tree：K dimentional Tree





以此类推，直到每个空间中最多有一个点。

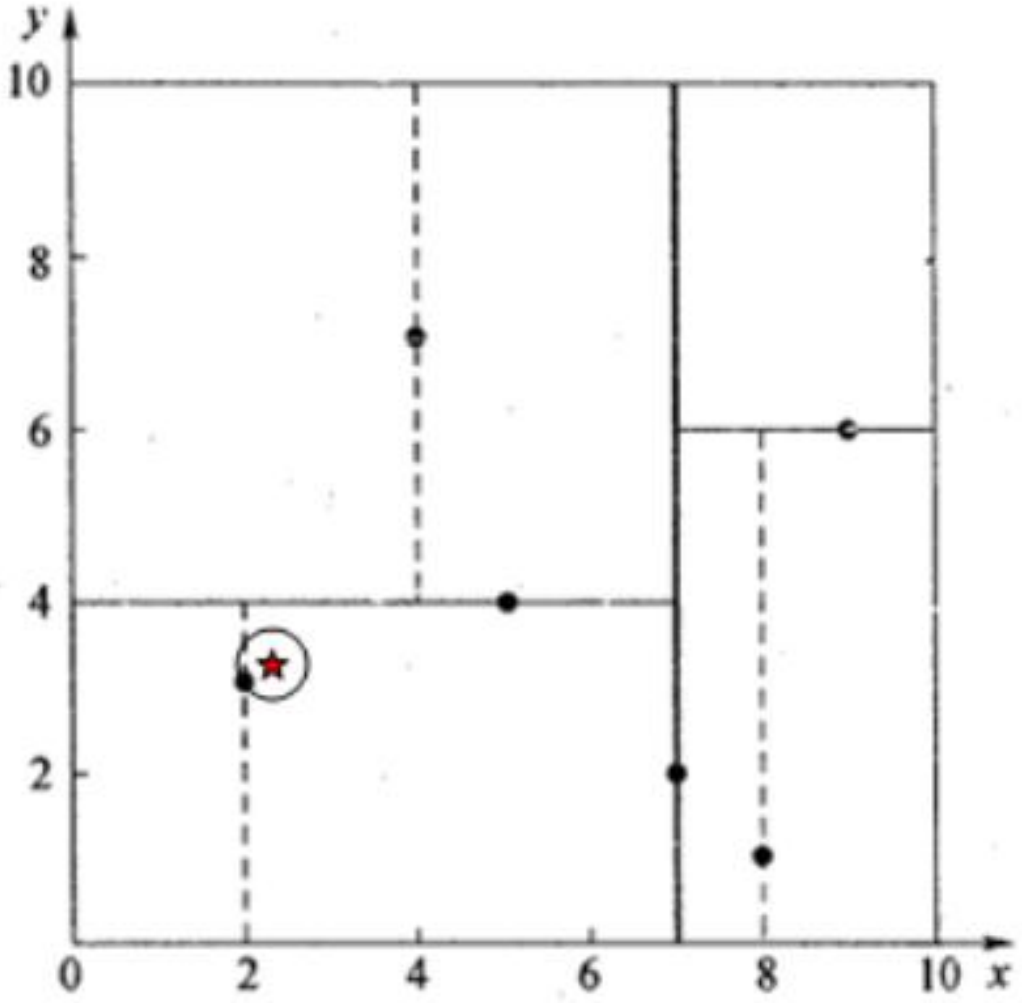
KD Tree：K dimentional Tree



如何查找某个点的临近点？

- 首先通过二叉树搜索（比较待查询节点和分裂节点的分裂维的值，小于等于就进入左子树分支，等于就进入右子树分支直到叶子结点）；
- 顺着搜索路径找到最近邻的近似点，也就是与待查询点处于同一个子空间的叶子结点；
- 回溯搜索路径，并判断搜索路径上的结点的其他子结点空间中是否可能有距离查询点更近的数据点，如果有可能，则需要跳到其他子结点空间中去搜索（也就是将其他子结点加入到搜索路径）。
- 重复这个过程直到搜索路径为空。

KD Tree：K dimentional Tree



如何查找某个点的临近点？

现有一个k-d tree：样本集{(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)}

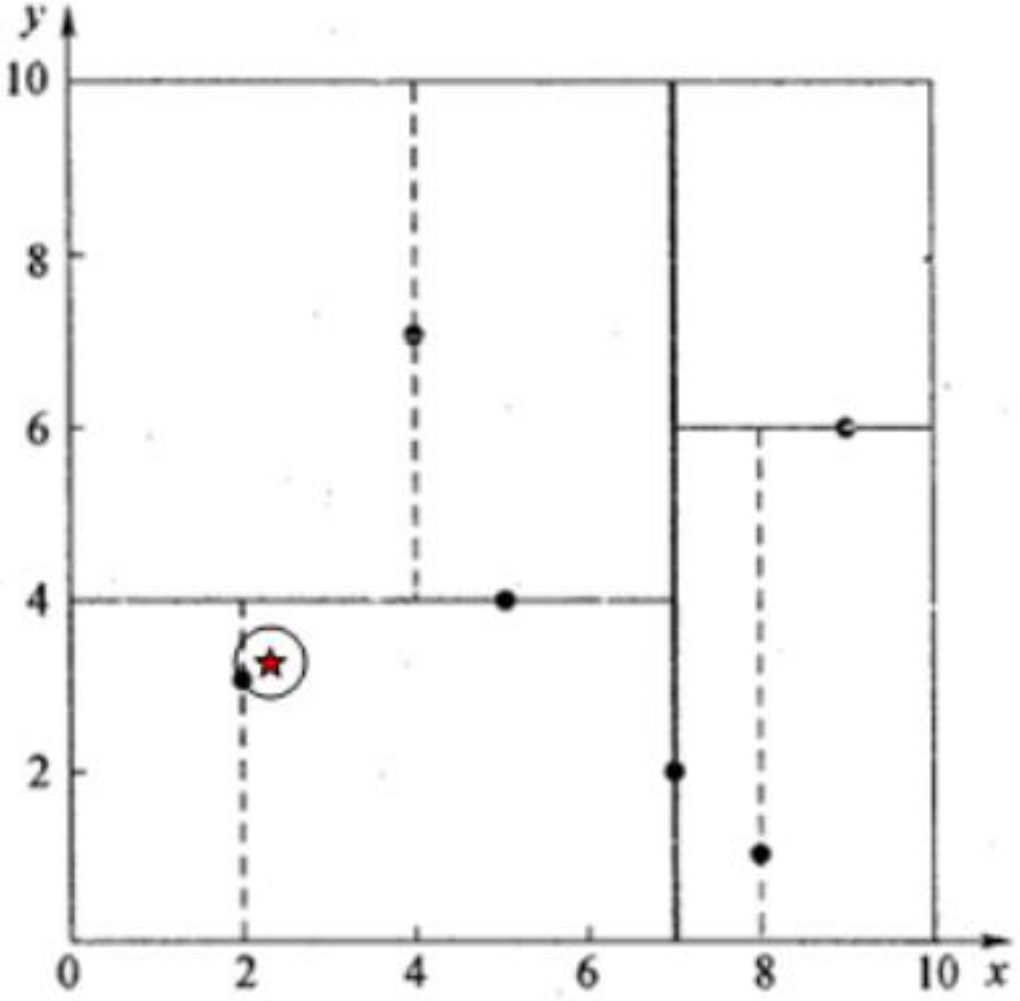
切分轴：粗实线，细实线，虚线

查找点(2.1,3.1)的邻近点？

search_path：<(7,2), (5,4), (2,3)>

(2,3)：作为当前最佳结点nearest, dist为0.141

KD Tree：K dimentional Tree



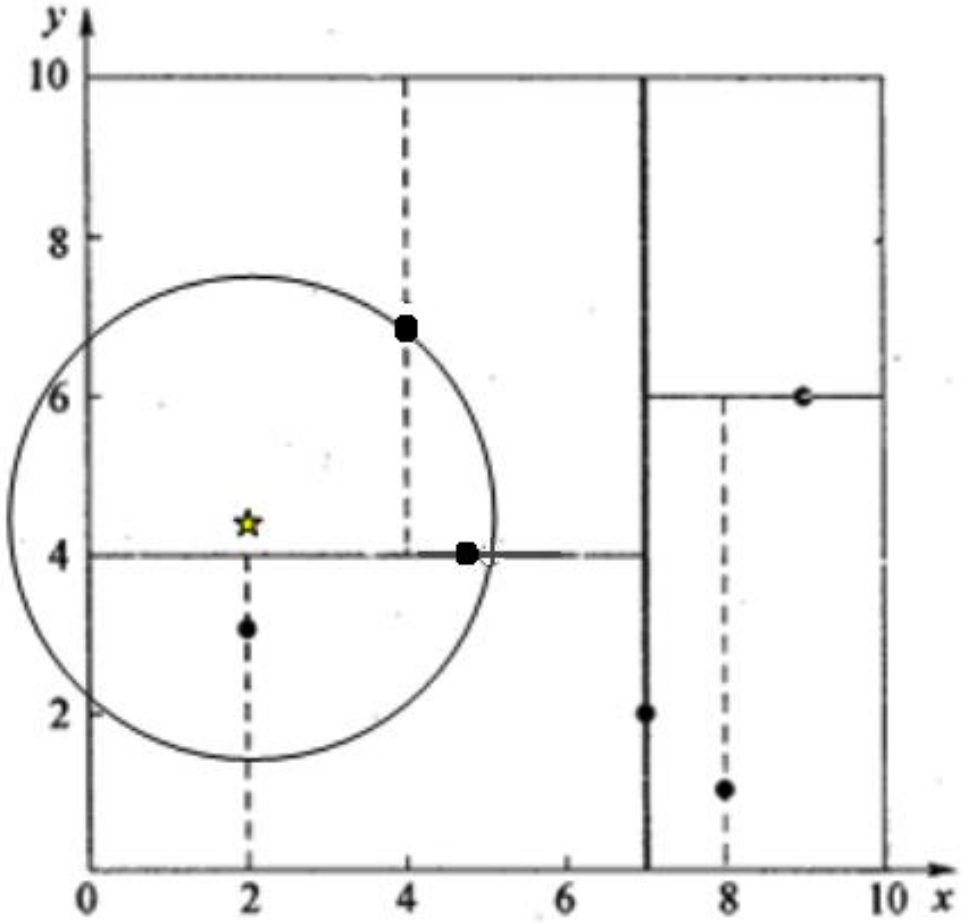
search_path：<(7,2), (5,4), (2,3)>
(2,3)：作为当前最佳结点nearest, dist为0.141

回溯至(5,4)，以(2.1,3.1)为圆心，以dist=0.141为半径画一个圆，并不和超平面y=4相交，所以不必跳到结点(5,4)的右子空间去搜索，因为右子空间中不可能有更近样本点了。

回溯至(7,2)，同理，以(2.1,3.1)为圆心，以dist=0.141为半径画一个圆，并不和超平面x=7相交，所以也不用跳到结点(7,2)的右子空间去搜索。

至此，search_path为空，结束整个搜索，返回nearest(2,3)作为(2.1,3.1)的最近邻点，最近距离为0.141。

KD Tree：K dimentional Tree



查找点(2,4.5)？

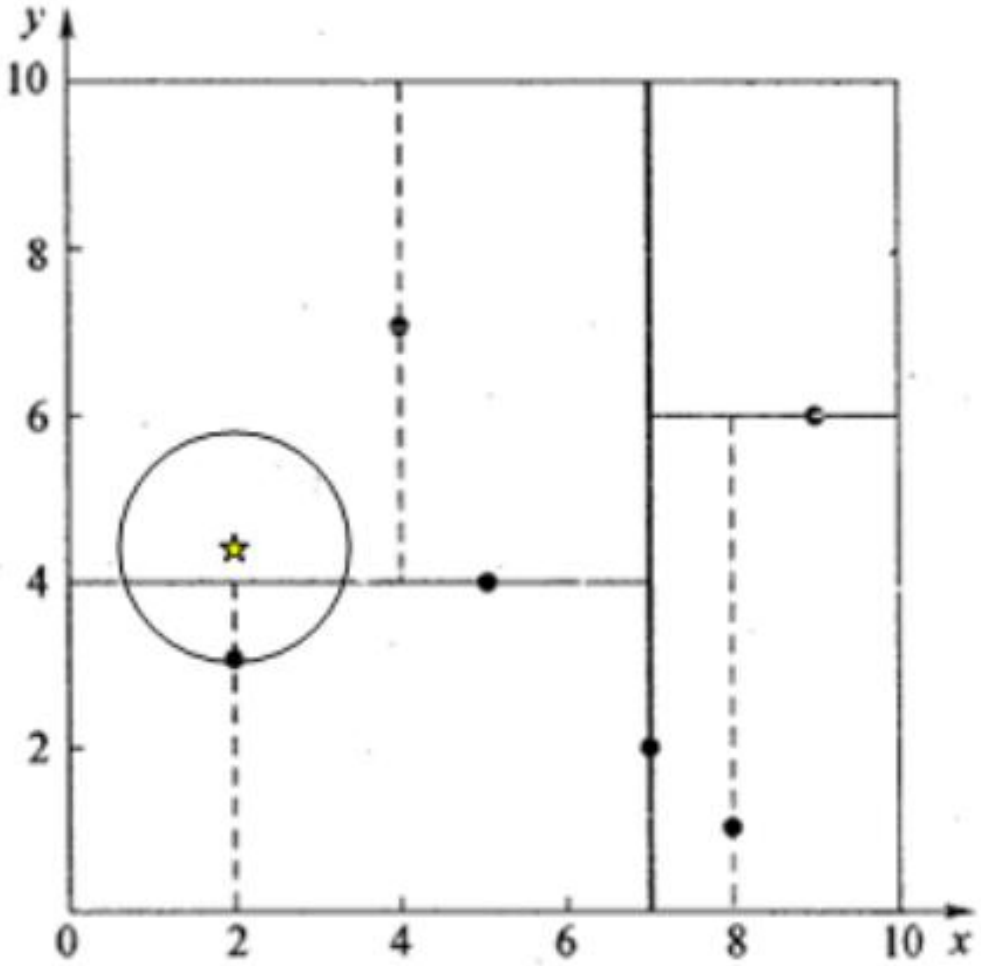
search_path：<(7,2), (5,4), (4,7)>

从search_path中取出(4,7)作为当前最佳结点nearest, dist为3.202；

回溯至(5,4)，以(2,4.5)为圆心，以dist=3.202为半径画一个圆与超平面y=4相交，所以需要跳到(5,4)的左子空间去搜索。

将(2,3)加入到search_path中，
search_path：<(7,2), (2, 3)>；
另，(5,4)与(2,4.5)的距离为3.04 < dist = 3.202，所以将(5,4)赋给nearest，并且dist=3.04。

KD Tree：K dimentional Tree



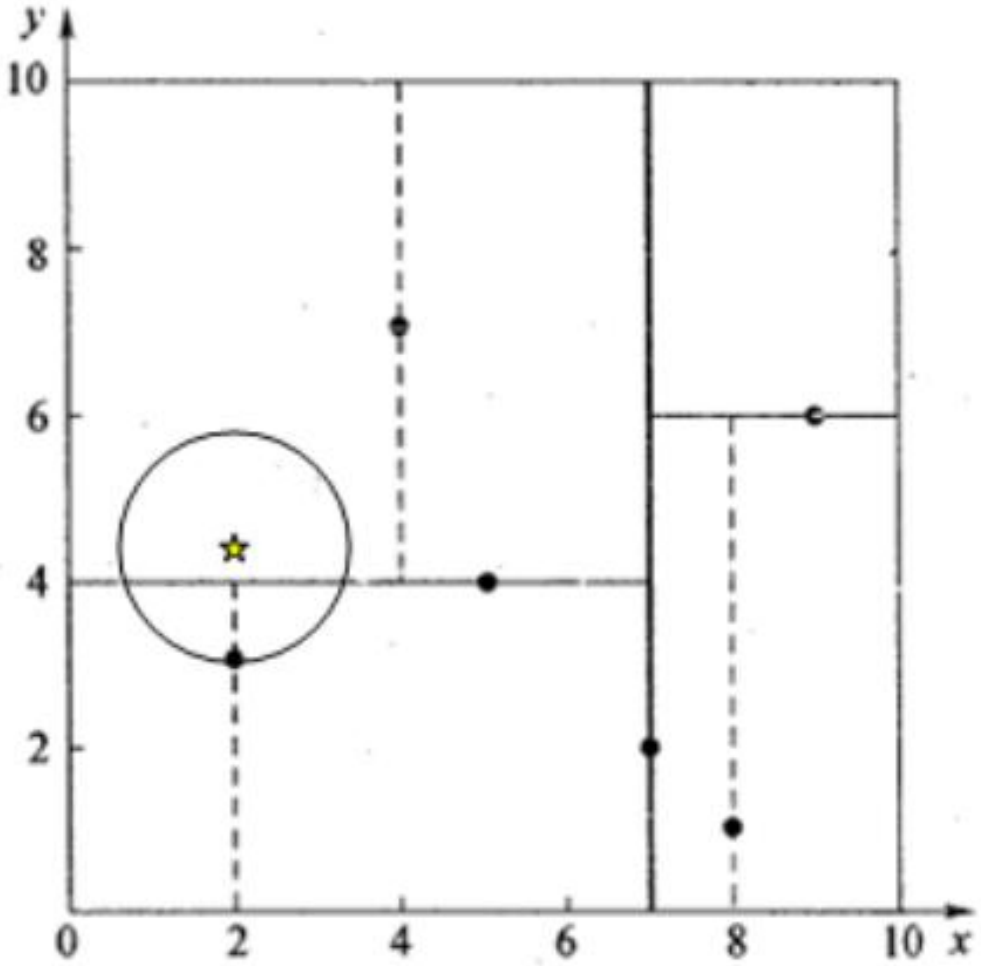
search_path：<(7,2), (2, 3)>
dist=3.04

回溯至(2,3)，(2,3)是叶子节点，直接平判断(2,3)是否离(2,4.5)更近，计算得到距离为1.5，所以nearest更新为(2,3)，dist更新为(1.5)。

回溯至(7,2)，同理，以(2,4.5)为圆心，以dist=1.5为半径画一个圆，不和超平面x=7相交，所以不用跳到结点(7,2)的右子空间去搜索。

至此，search_path为空，结束整个搜索，返回nearest(2,3)作为(2,4.5)的最近邻点，最近距离为1.5。

KD Tree：K dimentional Tree



search_path：<(7,2), (2, 3)>
dist=3.04

回溯至(2,3)，(2,3)是叶子节点，直接平判断(2,3)是否离(2,4.5)更近，计算得到距离为1.5，所以nearest更新为(2,3)，dist更新为(1.5)。

回溯至(7,2)，同理，以(2,4.5)为圆心，以dist=1.5为半径画一个圆，不和超平面x=7相交，所以不用跳到结点(7,2)的右子空间去搜索。

至此，search_path为空，结束整个搜索，返回nearest(2,3)作为(2,4.5)的最近邻点，最近距离为1.5。

京东智联云 & 贪心科技

休息

LSH：Locality Sensitive Hashing

问题提出：

给定1000万条微博，请对相似度大于0.8的条目进行去重。

两两比较所有的微博，对相同的微博值保留一条即可需要两两比较的微博有10^6亿（n*(n-1)/2）对。

LSH：Locality Sensitive Hashing

Shingling（n-gram？）
文档的k-shingle定义为其中长度为k的子串。

例：
s = 从 决心 减肥 的 这 一刻 起 请 做 如下 小 改变
k = 2
shingle = {"从 决心","决心 减肥","减肥 的","的 这","这 一刻","一刻 起","起 请","请 做","做 如下","如下 小","小 改变"}

LSH：Locality Sensitive Hashing

k = 1
s1 = "我 减肥"
s2= "要"
s3 = "他 减肥 成功"
s4 = "我 要 减肥"

元素	S1	S2	S3	S4
我	1	0	0	1
他	0	0	1	0
要	0	1	0	1
减肥	1	0	1	1
成功	0	0	1	0

LSH：Locality Sensitive Hashing

min-hashing

特征矩阵按行进行一个随机的排列后，第一个列值为1的行的行号。

h(S1)=3，h(S2)=5，
h(S3)=1，h(S4)=3

元素	S1	S2	S3	S4
他	0	0	1	0
成功	0	0	1	0
我	1	0	0	1
减肥	1	0	1	1
要	0	1	0	1

LSH：Locality Sensitive Hashing

why min-hashing

- 两列的min-hashing相等的概率就是这两列的Jaccard相似度。
- P(h(Si)=h(Sj)) = sim(Si,Sj)

- 可以把一篇文档看作关于词的集合。
- 集合S，T的Jaccard相似度：

$$SIM(S,T) = \frac{|S交T|}{|S并T|}$$

- S = “我 喜欢 苹果”，T = “我 喜欢 土豆”
- 集合S，T的Jaccard相似度：2/4

LSH：Locality Sensitive Hashing

why min-hashing？

- 考虑Si和Sj两列，所在的行的所有可能结果可以分成如下三类：

- A: 两列的值都为1
- B: 其中一列的值为0，另一列的值为1
- C: 两列的值都为0

- 只有A、B类行决定sim(Si,Sj)，假定A类行有a个，B类行有b个，那么sim(si,sj)=a/(a+b)。
- 现在把C类行都删掉，那么第一行不是A类行就是B类行，如果第一行是A类行那么h(Si)=h(Sj)。
- P(h(Si)=h(Sj))：P(删掉C类行后，第一行为A类) = A类行的数目/所有行的数目 = a/(a+b)

LSH：Locality Sensitive Hashing

- 选择n个随机排列作用于特征矩阵，得到n个min hash值，h1,h2,...,hn。
- 这n个最小hash值组成一个n维向量，称为最小签名，两列的最小签名的相似度即为两列的Jaccard相似度的估计。

元素	S1	S2	S3	S4
他	0	0	1	0
成功	0	0	1	0
我	1	0	0	1
减肥	1	0	1	1
要	0	1	0	1

2	1	2	1
2	1	4	1
1	2	1	2

n = 3

LSH：Locality Sensitive Hashing

- 思考：如何快速得到最小签名表？
- hint: 只关注1的位置

元素	S1	S2	S3	S4
他	0	0	1	0
成功	0	0	1	0
我	1	0	0	1
减肥	1	0	1	1
要	0	1	0	1

2	1	2	1
2	1	4	1
1	2	1	2

n = 3

LSH：Locality Sensitive Hashing

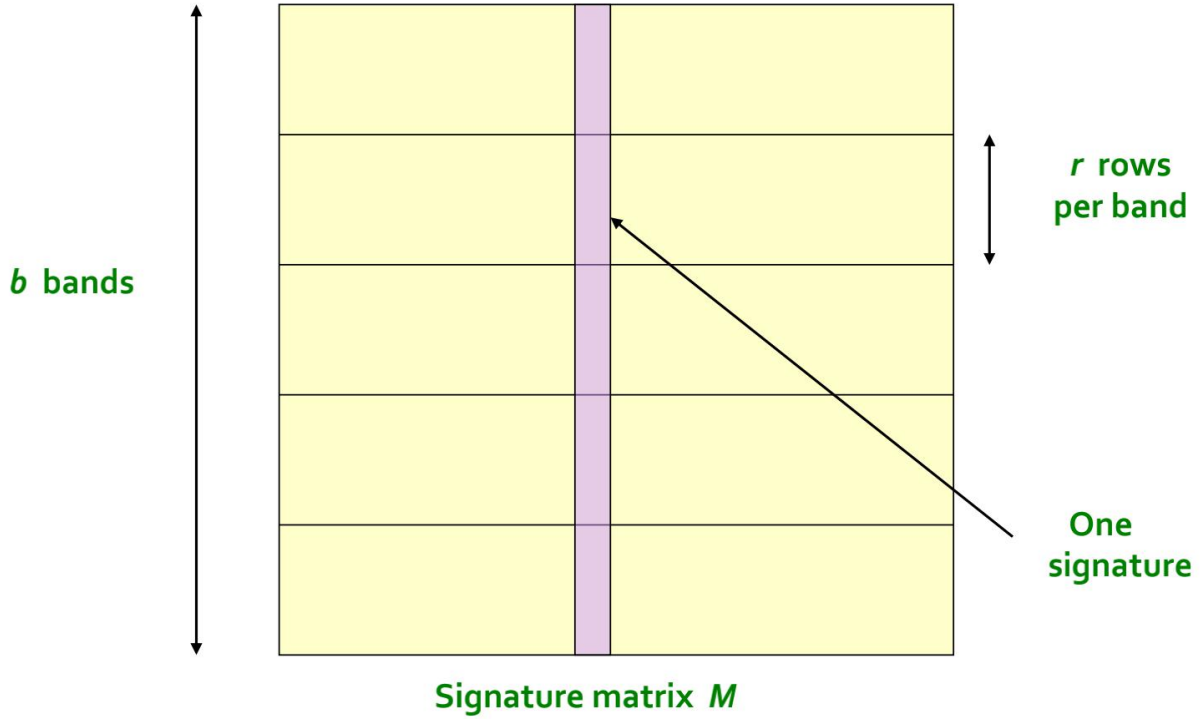
- 虽然利用min hash压缩了表征文本的向量维度。
- 但要计算两列的相似度还是需要两两比较签名矩阵两列的相似度，如果有n篇文档，两个比较的次数是n*(n-1)/2。

LSH：Locality Sensitive Hashing

- 局部敏感hash

对签名矩阵M按行进行分组，将矩阵分成b组，每组由r行组成。

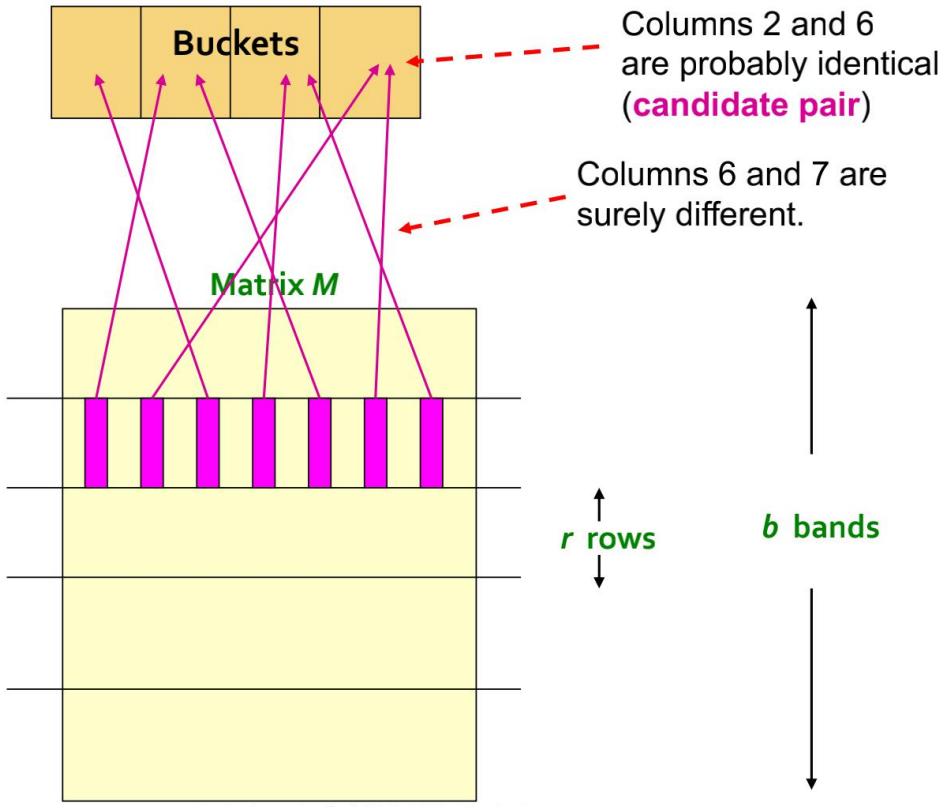
n = b * r



LSH：Locality Sensitive Hashing

- 局部敏感hash

只要两列有一组的最小签名相同，那么这两列就会放到同一个桶而成为候选相似项。



LSH：Locality Sensitive Hashing

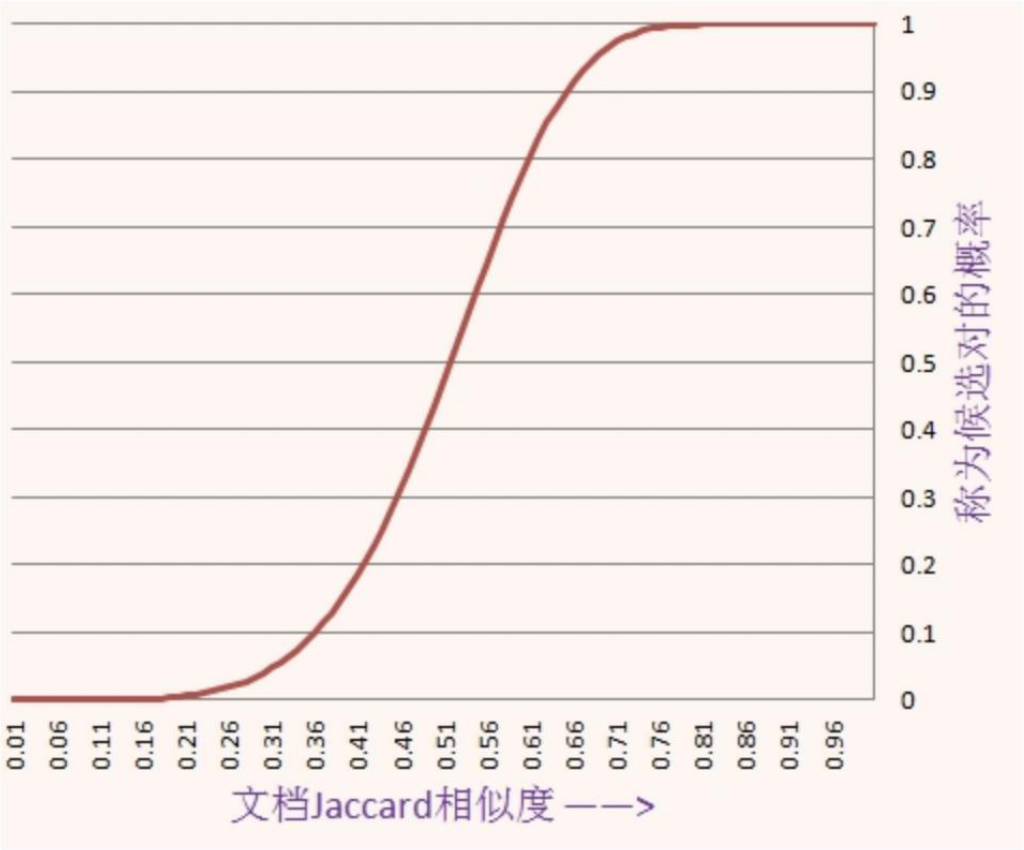
- 局部敏感hash
- 令 p = sim(si,sj)
- 已知两个min-hash相同的概率 =两列的相似度= p
- b组，r行

(1) 在某个组中所有行的两个签名值都相等概率是pr
(2) 在某个组中至少有一对签名不相等的概率是$1-pr$
(3) 在每一组中至少有一对签名不相等的概率是$(1-pr)b$
(4) 至少有一个组的所有对的签名相等的概率是$1-(1-pr)b$

LSH：Locality Sensitive Hashing

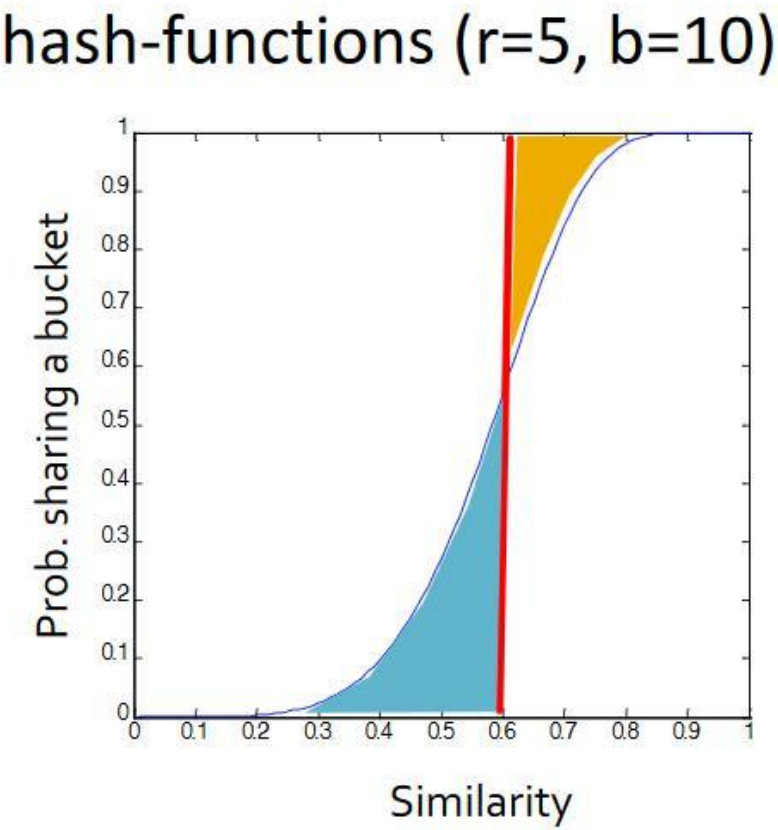
b = 5
r = 20

p	1-(1-p^5)^20
0.1	0.0002
0.2	0.0064
0.3	0.0475
0.4	0.1860
0.5	0.4701
0.6	0.8019
0.7	0.9748
0.8	0.9996
0.9	1.0000



假如两文本相似 度为0.8，有0.9996的可能性被放到一个桶中。

LSH：Locality Sensitive Hashing



false negative
false positive

相关代码

Annoy: https://github.com/spotify/annoy
HNSW: https://github.com/nmslib/hnswlib
LSH: https://github.com/mattilyra/LSH
KD tree: sklearn
ANN测评：http://github.com/erikbern/ann-benchmarks/

参考

ANN Approximate Nearest Neighbor

https://yongyuan.name/blog/ann-search.html

KD树

https://www.cnblogs.com/lesleysbw/p/6074662.html

https://blog.csdn.net/app_12062011/article/details/51986805

https://www.jianshu.com/p/ffe52db3e12b

http://stanford.edu/class/archive/cs/cs106l/cs106l.1162/handouts/assignment-3-kdtree.pdf

https://blog.csdn.net/qp120291570/article/details/41483689

参考

局部敏感哈希（LSH）

https://www.cnblogs.com/fengfenggirl/p/lsh.html

HNSW

https://juejin.im/post/6850418107438071821

https://milvus.io/cn/blogs/2020-01-16-hnsw-nsg-comparison.md

Nearest neighbor methods and vector models annoy

https://erikbern.com/2015/09/24/nearest-neighbor-methods-vector-models-part-1.html

https://zhuanlan.zhihu.com/p/105594786?utm_source=wechat_session