

机器学习-KNN算法

本章目录

2

01 距离度量

02 KNN算法

03 KD树划分

04 KD树搜索

1.距离度量

3

01 距离度量

02 KNN算法

03 KD树划分

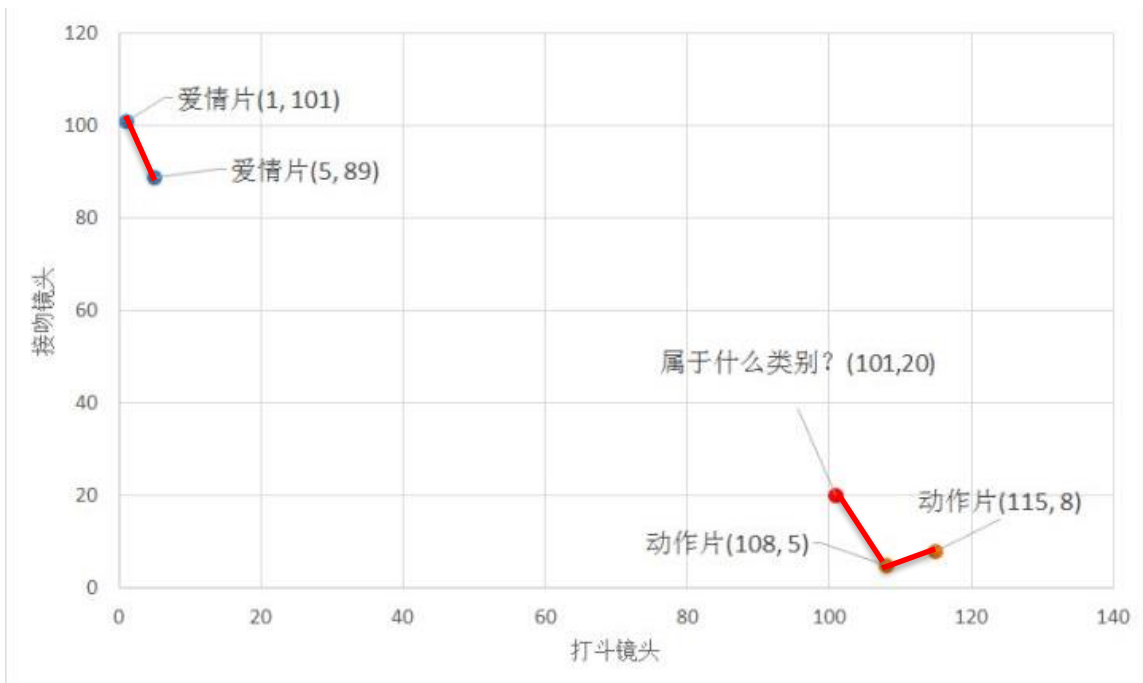
04 KD树搜索

距离度量

4

欧氏距离(Euclidean distance)

电影分类



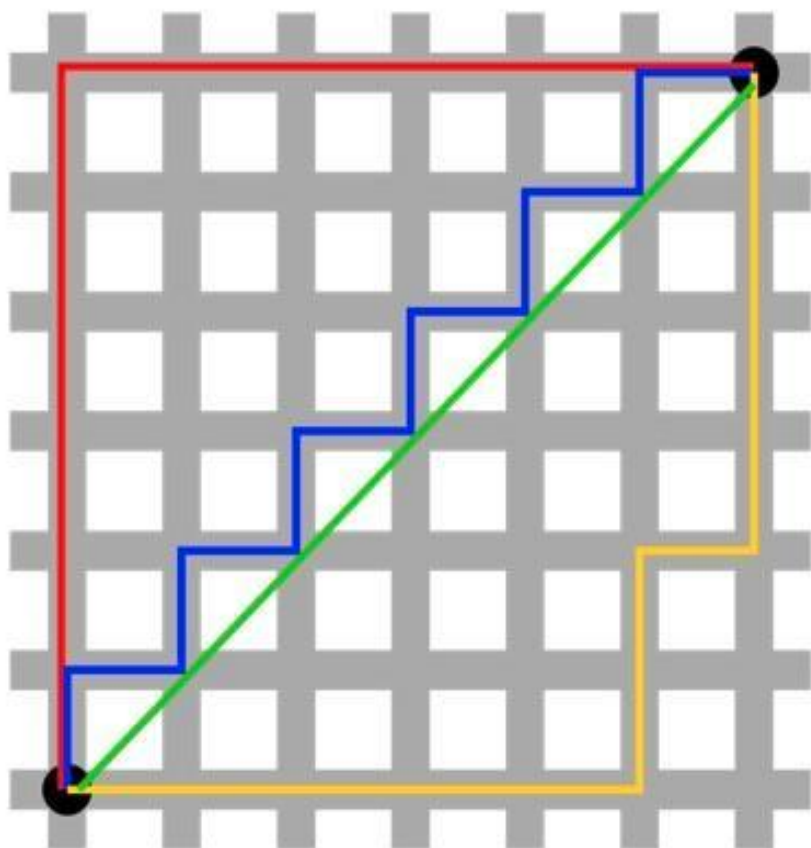
$$d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

欧几里得度量 (Euclidean Metric) (也称欧氏距离) 是一个通常采用的距离定义, 指在 m 维空间中两个点之间的真实距离, 或者向量的自然长度 (即该点到原点的距离)。在二维和三维空间中的欧氏距离就是两点之间的实际距离。

距离度量

5

曼哈顿距离(Manhattan distance)



$$d(x, y) = \sum_i |x_i - y_i|$$


想象你在城市道路里，要从一个十字路口开车到另外一个十字路口，驾驶距离是两点间的直线距离吗？显然不是，除非你能穿越大楼。实际驾驶距离就是这个“曼哈顿距离”。而这也是曼哈顿距离名称的来源，曼哈顿距离也称为城市街区距离(City Block distance)。

距离度量

6

切比雪夫距离(Chebyshev distance)

$$d(x, y) = \max_i |x_i - y_i|$$

	a	b	c	d	e	f	g	h	
8	5	4	3	2	2	2	2	2	8
7	5	4	3	2	1	1	1	2	7
6	5	4	3	2	1		1	2	6
5	5	4	3	2	1	1	1	2	5
4	5	4	3	2	2	2	2	2	4
3	5	4	3	3	3	3	3	3	3
2	5	4	4	4	4	4	4	4	2
1	5	5	5	5	5	5	5	5	1
	a	b	c	d	e	f	g	h	

二个点之间的距离定义是其各坐标数值差绝对值的最大值。

国际象棋棋盘上二个位置间的切比雪夫距离是指王要从一个位子移至另一个位子需要走的步数。由于王可以往斜前或斜后方向移动一格，因此可以较有效率的到达目的的格子。上图是棋盘上所有位置距f6位置的切比雪夫距离。

距离度量

7

闵可夫斯基距离(Minkowski distance)

p 取1或2时的闵氏距离是最为常用的

$p = 2$ 即为欧氏距离,

$p = 1$ 时则为曼哈顿距离。

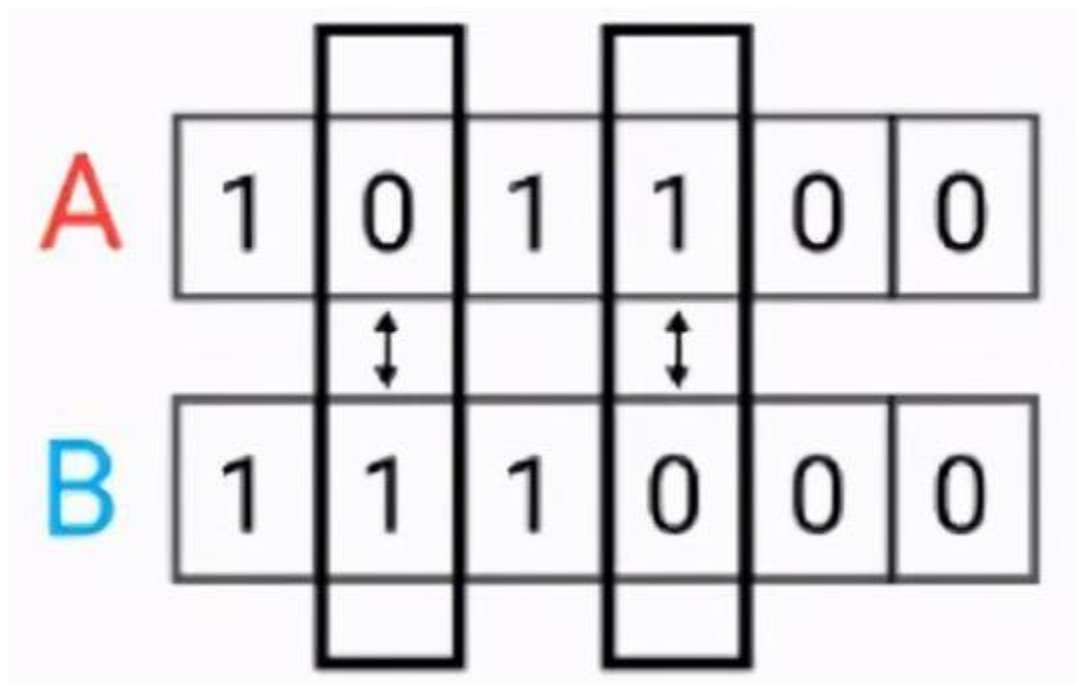
当 p 取无穷时的极限情况下, 可以得到切比雪夫距离

$$d(x, y) = \left(\sum_i |x_i - y_i|^p \right)^{\frac{1}{p}}$$

距离度量

8

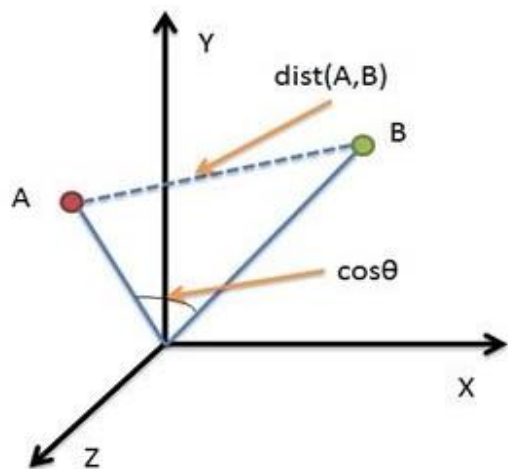
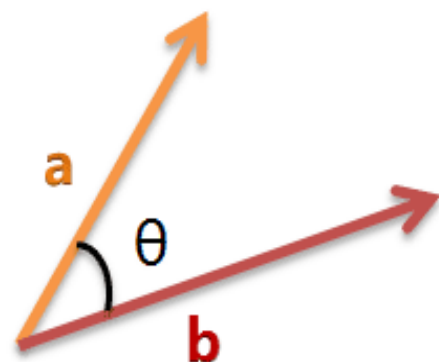
汉明距离(Hamming distance)



$$d(x, y) = \frac{1}{N} \sum_i 1_{x_i \neq y_i}$$

汉明距离是使用在数据传输差错控制编码里面的，汉明距离是一个概念，它表示两个（相同长度）字对应位不同的数量，我们以表示两个字之间的汉明距离。对两个字符串进行异或运算，并统计结果为1的个数，那么这个数就是汉明距离。

余弦相似度



两个向量有相同的指向时，余弦相似度的值为1；两个向量夹角为 90° 时，余弦相似度的值为0；两个向量指向完全相反的方向时，余弦相似度的值为-1。

假定 A 和 B 是两个 n 维向量， A 是 $[A_1, A_2, \dots, A_n]$ ， B 是 $[B_1, B_2, \dots, B_n]$ ，则 A 和 B 的夹角的余弦等于：

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

2.KNN算法

10

01 距离度量

02 KNN算法

03 KD树划分

04 KD树搜索

2.KNN算法

11

k 近邻法 (k-Nearest Neighbor, kNN) 是一种比较成熟也是最简单的机器学习算法，可以用于基本的分类与回归方法。

算法的主要思路：

如果一个样本在特征空间中与 k 个实例最为相似(即特征空间中**最邻近**)，那么这 k 个实例中大多数属于哪个类别，则该样本也属于这个类别。

对于分类问题：对新的样本，根据其 k 个最近邻的训练样本的类别，通过多数表决等方式进行预测。

对于回归问题：对新的样本，根据其 k 个最近邻的训练样本标签值的均值作为预测值。

2.KNN算法

12

k 近邻法 (k-Nearest Neighbor, kNN) 是一种比较成熟也是最简单的机器学习算法，可以用于基本的分类与回归方法。

k 近邻法的三要素：

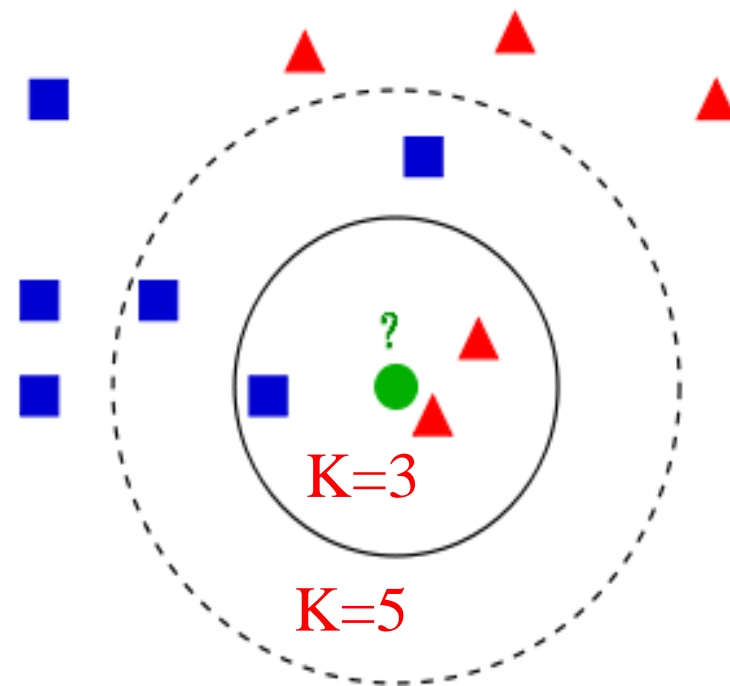
- k 值选择。
- 距离度量。
- 决策规则。

2.KNN算法

13

算法流程如下：

- 1.计算测试对象到训练集中每个对象的距离
- 2.按照距离的远近排序
- 3.选取与当前测试对象最近的k的训练对象，作为该测试对象的邻居
- 4.统计这k个邻居的类别频次
- 5.k个邻居里频次最高的类别，即为测试对象的类别



3.K-D-Tree划分

14

01 距离度量

02 KNN算法

03 KD树划分

04 KD树搜索

KD树划分

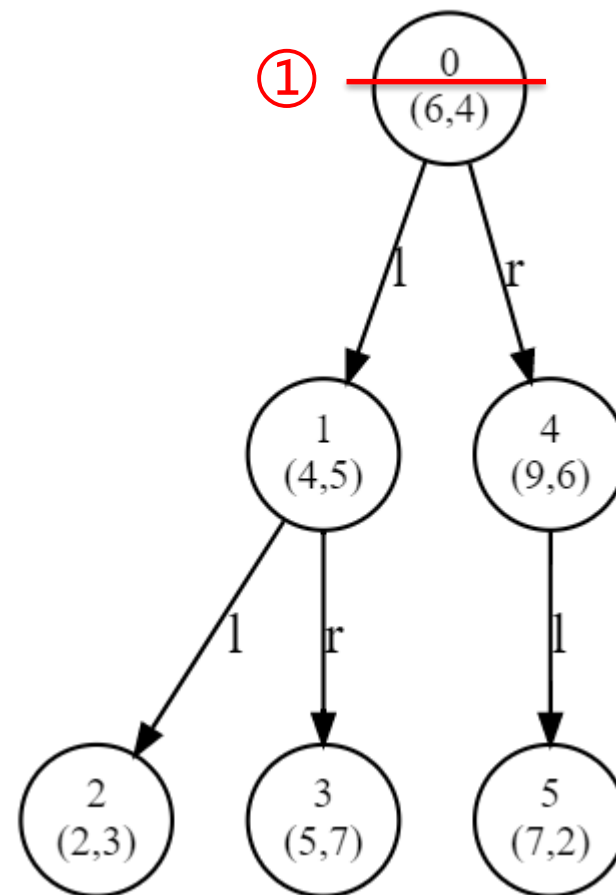
15

KD树(K-Dimension Tree), , 也可称之为K维树, 可以用更高的效率来对空间进行划分, 并且其结构非常适合寻找最近邻居和碰撞检测。

假设有 6 个二维数据点, 构建KD树的过程:

$D = \{(2,3), (5,7), (9,6), (4,5), (6,4), (7,2)\}$ 。

①从 x 轴开始划分, 根据 x 轴的取值2,5,9,4,6,7得到中位数为6, 因此切分线为: $x = 6$ 。



KD树划分

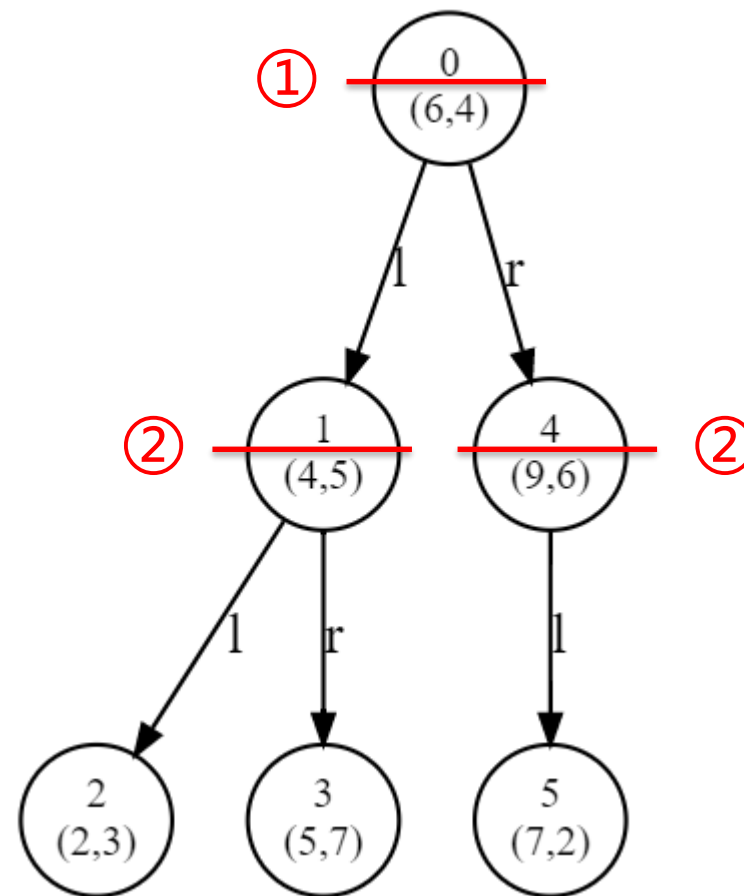
16

$D = \{(2,3), (5,7), (9,6), (4,5), (6,4), (7,2)\}$ 。

②可以根据 x 轴和 y 轴上数据的方差，选择方差最大的那个轴作为第一轮划分轴。

左子空间（记做 D_1 ）包含点 $(2,3), (4,5), (5,7)$ ，切分轴轮转，从 y 轴开始划分，切分线为： $y = 5$ 。

右子空间（记做 D_2 ）包含点 $(9,6), (7,2)$ ，切分轴轮转，从 y 轴开始划分，切分线为： $y = 6$ 。



KD树划分

17

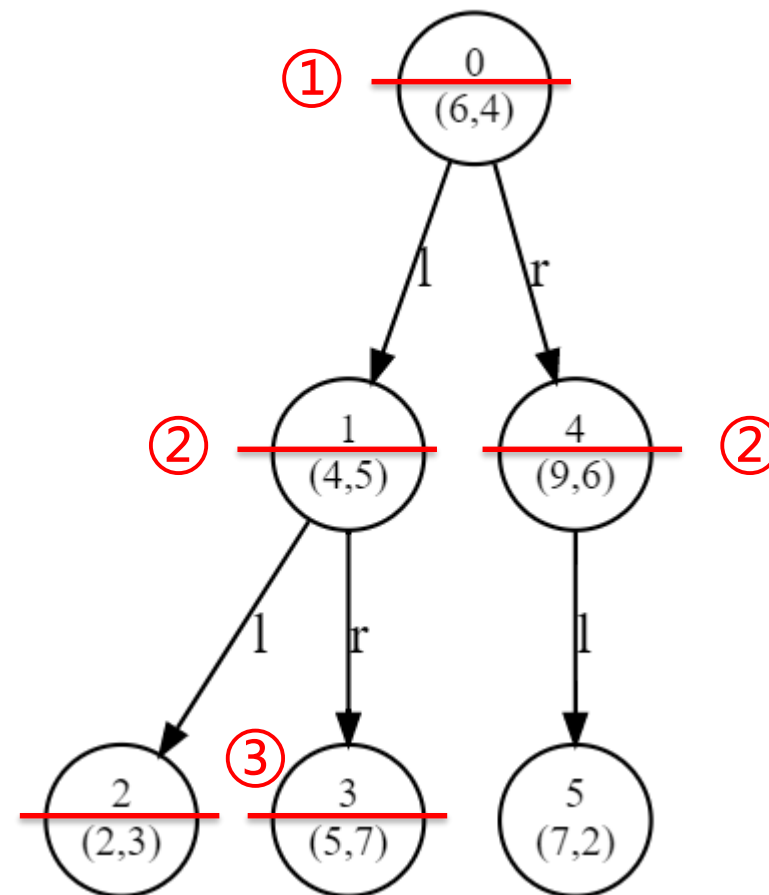
$D = \{(2,3), (5,7), (9,6), (4,5), (6,4), (7,2)\}$ 。

③ D_1 的左子空间（记做 D_3 ）包含点(2,3)，切分轴轮转，从x轴开始划分，切分线为： $x = 2$ 。

其左子空间记做 D_7 ，右子空间记做 D_8 。由于 D_7, D_8 都不包含任何点，因此对它们不再继续拆分。

D_1 的右子空间（记做 D_4 ）包含点(5,7)，切分轴轮转，从x轴开始划分，切分线为： $x = 5$ 。

其左子空间记做 D_9 ，右子空间记做 D_{10} 。由于 D_9, D_{10} 都不包含任何点，因此对它们不再继续拆分。



KD树划分

18

$D = \{(2,3), (5,7), (9,6), (4,5), (6,4), (7,2)\}$ 。

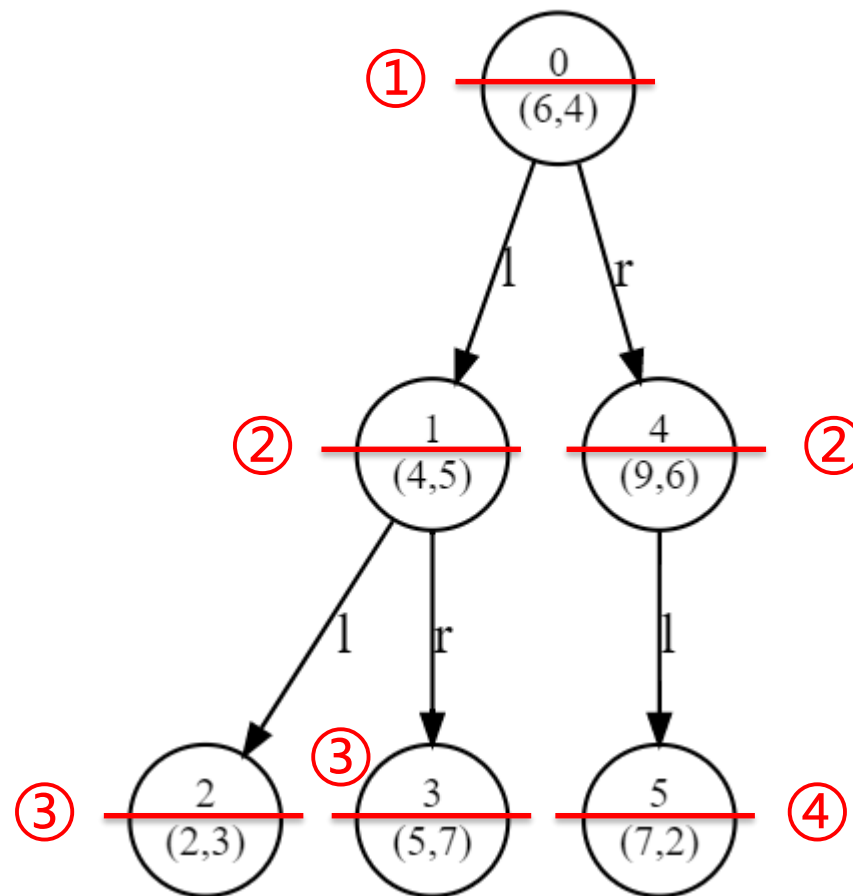
④ D_2 的左子空间（记做 D_5 ）包含点(7,2)，切分轴轮转，从x 轴开始划分，切分线为： $x = 7$

。

其左子空间记做 D_{11} ，右子空间记做 D_{12} 。

由于 D_{11}, D_{12} 都不包含任何点，因此对它们不再继续拆分。

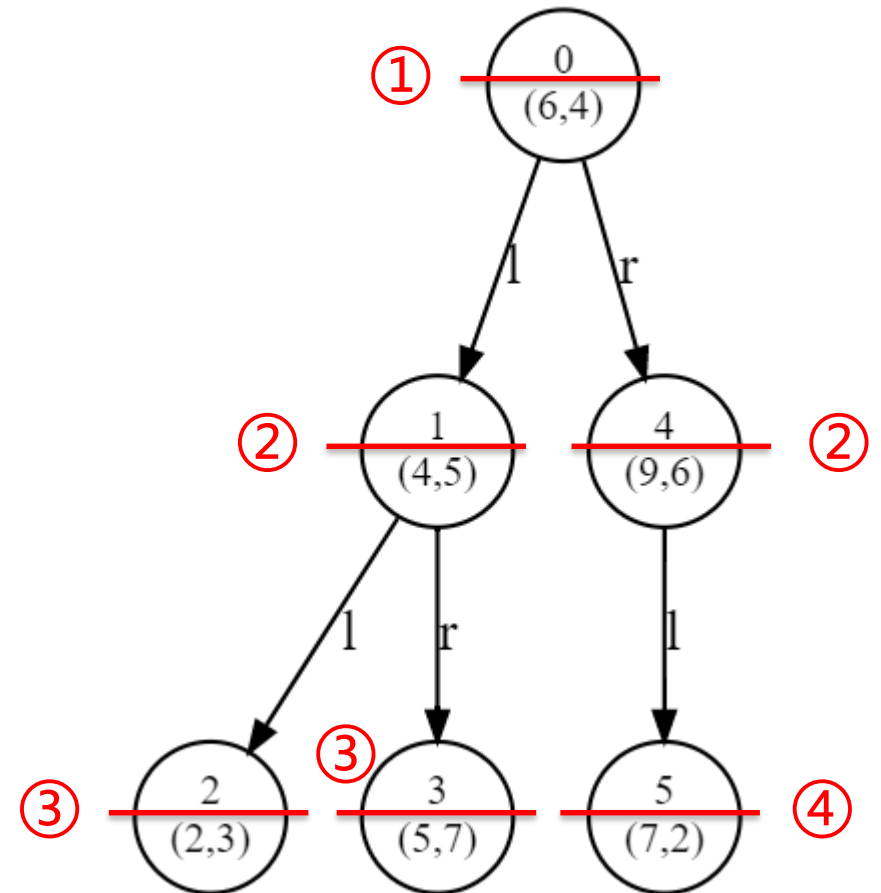
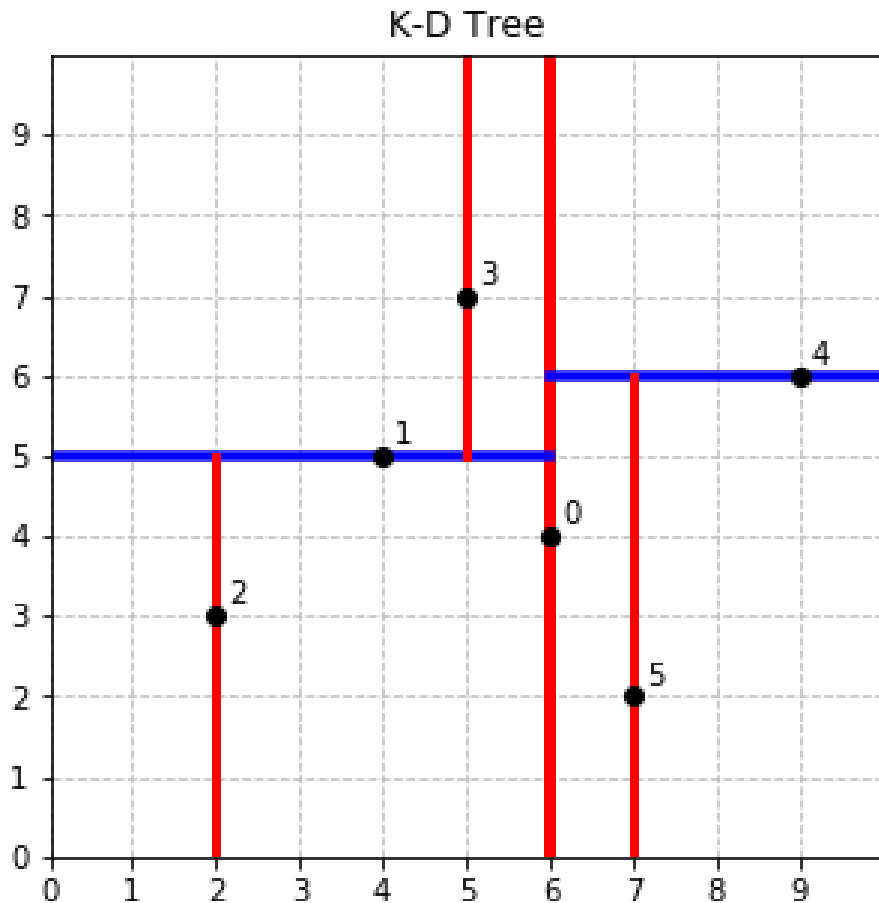
D_2 的右子空间（记做 D_6 ）不包含任何点，停止继续拆分。



KD树划分

19

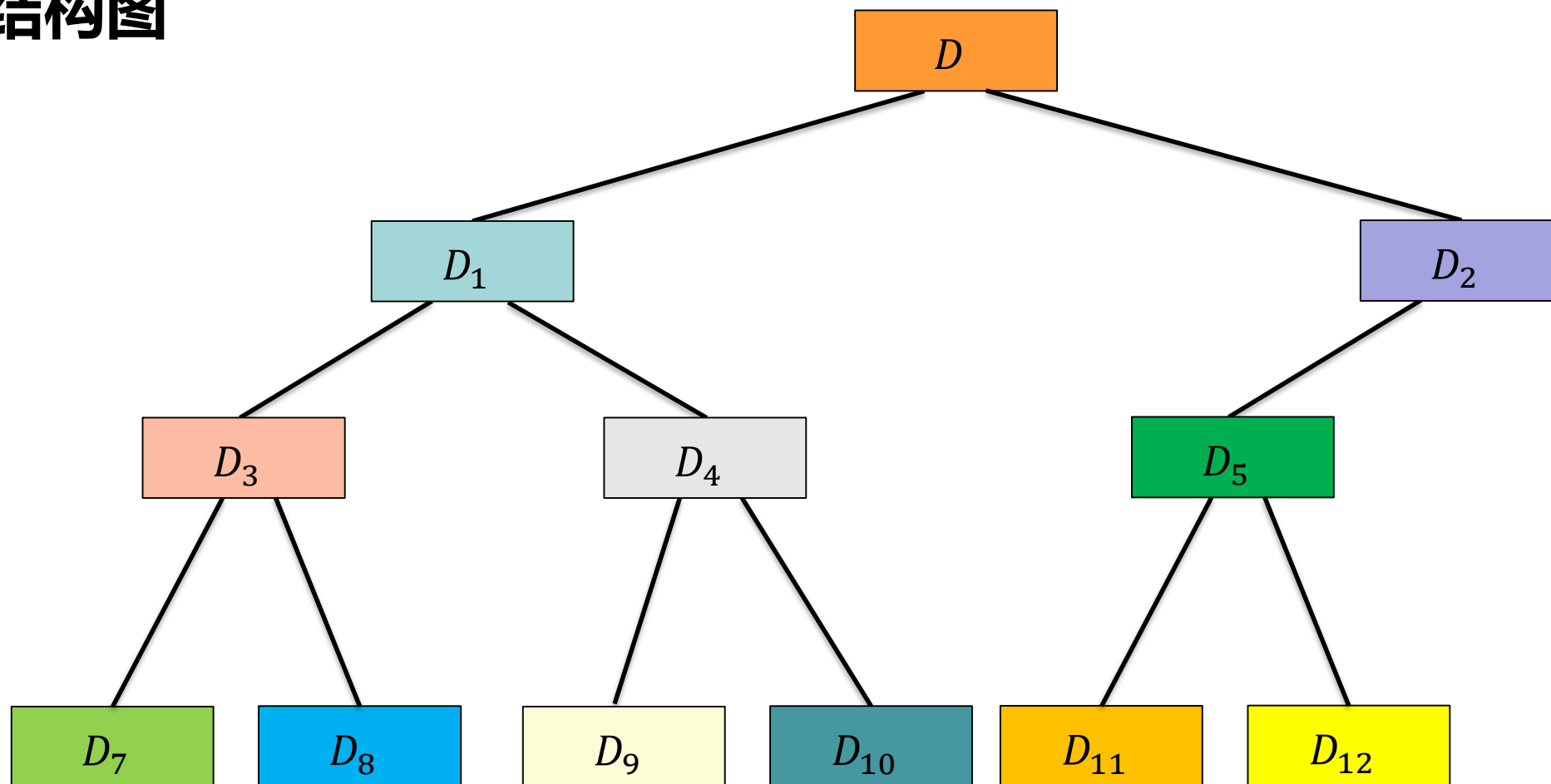
$D = \{(2,3), (5,7), (9,6), (4,5), (6,4), (7,2)\}$ 。



KD树划分

20

样本空间结构图



4.K-D-Tree搜索

21

01 距离度量

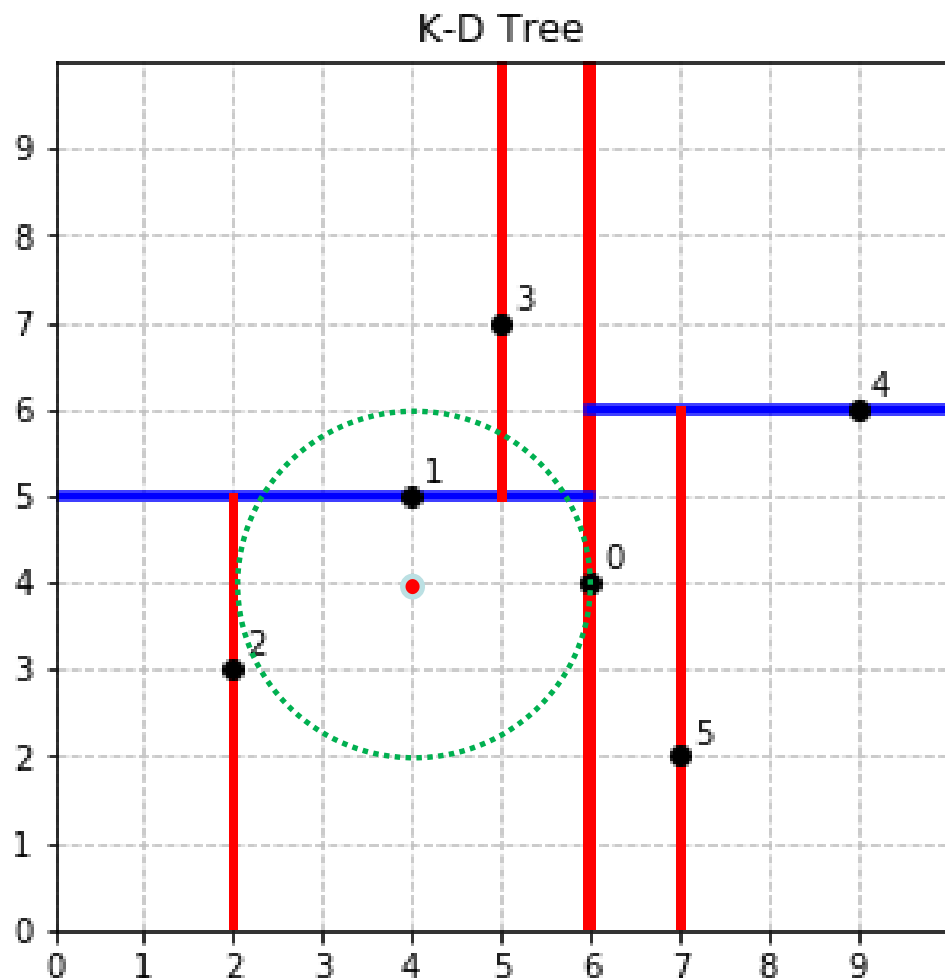
02 KNN算法

03 KD树划分

04 KD树搜索

KD树搜索

22

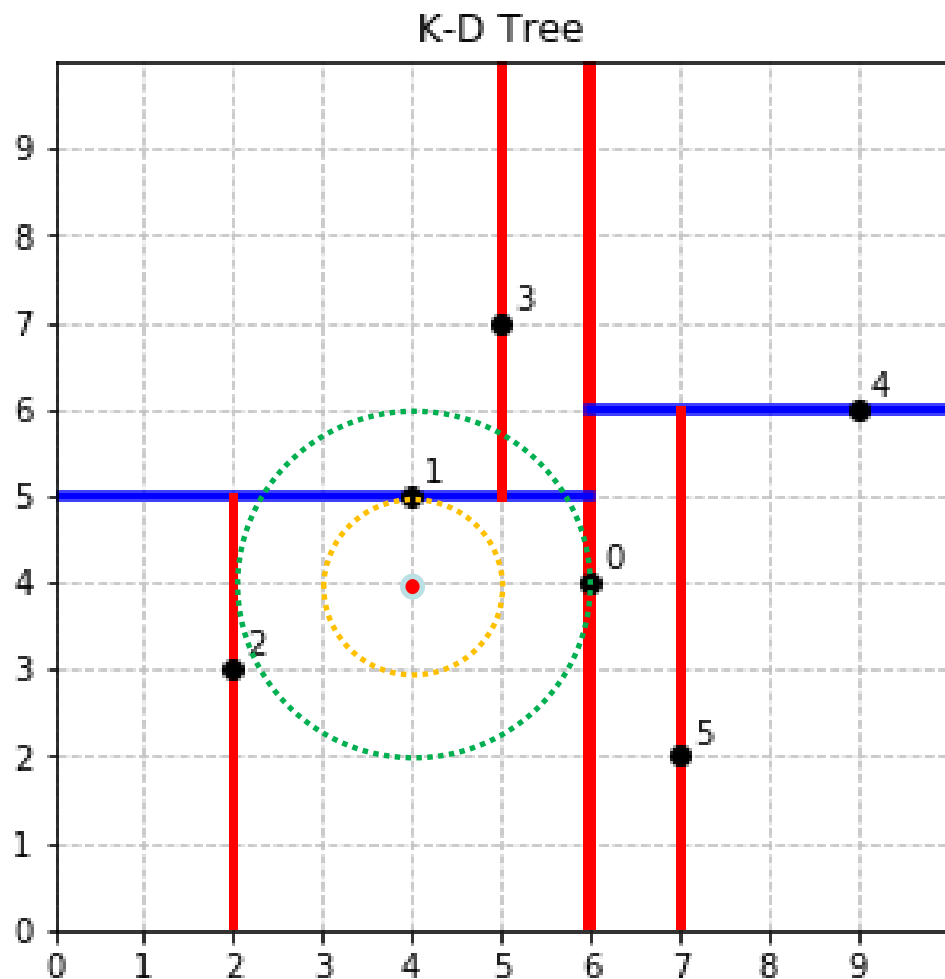


1.首先要找到该目标点的叶子节点，然后以目标点为圆心，目标点到叶子节点的距离为半径，建立一个超球体，我们要找寻的最近邻点一定是在该球体内部。

搜索 (4,4) 的最近邻时。首先从根节点 (6,4) 出发，将当前最近邻设为 (6,4)，对该KD树作深度优先遍历。以 (4,4) 为圆心，其到 (6,4) 的距离为半径画圆（多维空间为超球面），可以看出 (7,2) 右侧的区域与该圆不相交，所以 (7,2) 的右子树全部忽略。

KD树搜索

23



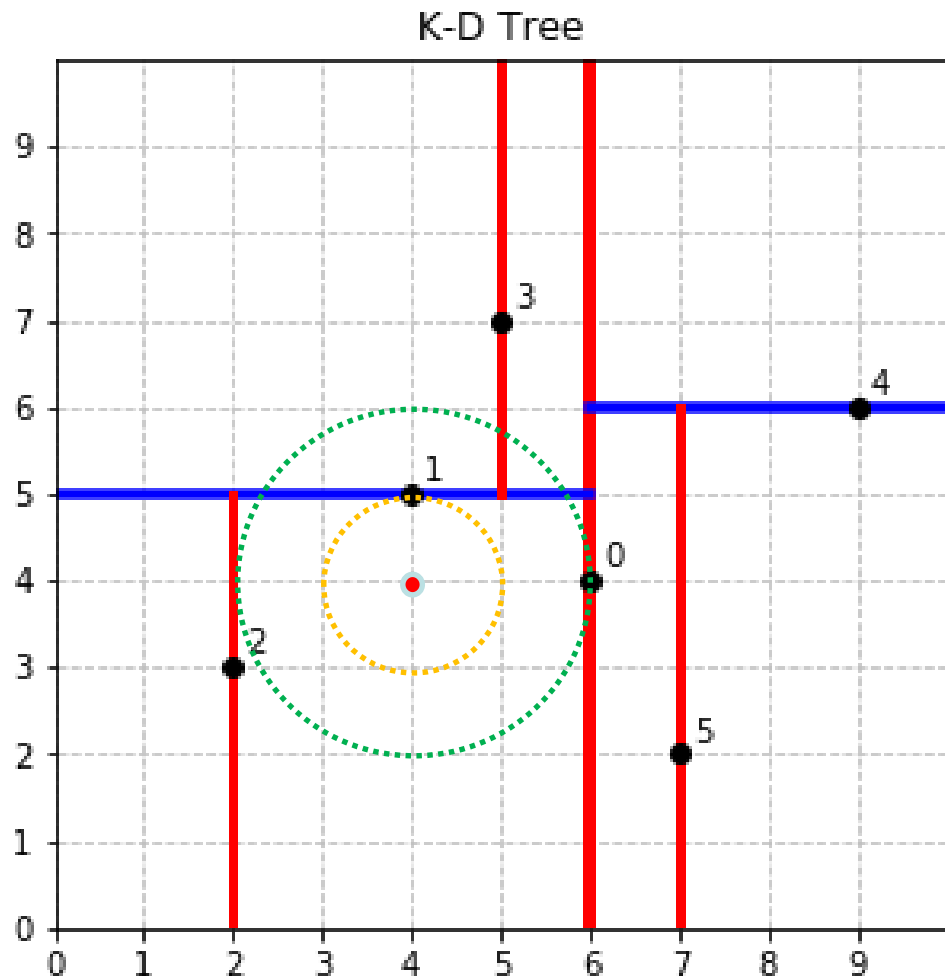
2.返回叶子结点的父节点，检查另一个子结点包含的超矩形体是否和超球体相交，如果相交就到这个子节点寻找是否有更加近的近邻，有的话就更新最近邻。

接着走到 (6,4) 左子树根节点 (4,5)，与原最近邻对比距离后，更新当前最近邻为 (4,5)。

以 (4,4) 为圆心，其到 (4,5) 的距离为半径画圆，发现 (6,4) 右侧的区域与该圆不相交，忽略该侧所有节点，这样 (6,4) 的整个右子树被标记为已忽略。

KD树搜索

24



3.如果不相交直接返回父节点，在另一个子树继续搜索最近邻。

4.当回溯到根节点时，算法结束，此时保存的最近邻节点就是最终的最近邻。

遍历完 (4,5) 的左右叶子节点，发现与当前最优距离相等，不更新最近邻。

所以 (4,4) 的最近邻为 (4,5) 。

- [1] Andrew Ng. Machine Learning[EB/OL]. Stanford University,2014.
<https://www.coursera.org/course/ml>
- [2] 李航. 统计学习方法[M]. 清华大学出版社,2019.
- [3] 周志华. 机器学习[M]. 清华大学出版社,2016.
- [4] Cover T M , Hart P E . Nearest neighbor pattern classification[J]. IEEE Trans.inf.theory, 1953, 13(1):21-27.
- [5] Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning[M]. Springer, New York, NY, 2001.
- [6] CHRISTOPHER M. BISHOP. Pattern Recognition and Machine Learning[M]. Springer,2006.
- [7] Stephen Boyd, Lieven Vandenberghe, Convex Optimization[M]. Cambridge University Press, 2004.