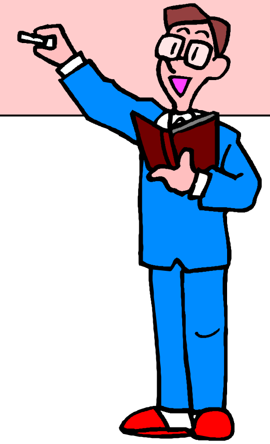
计算机模式识别与机器学习

—— 近邻算法 *k*-NN



主讲：图像处理与模式识别研究所
赵群飞

邮　　箱：zhaoqf@sjtu.edu.cn
办公室：电院 2-441
电　　话：13918191860

第8章　K-近邻算法 k-NN

- **本章学习目标**
  - ✓ 掌握距离度量的定义
  - ✓ 理解K-NN算法原理
  - ✓ 能够熟练运用KD树划分技术
  - ✓ 了结 KD树搜索

目录

K-NN

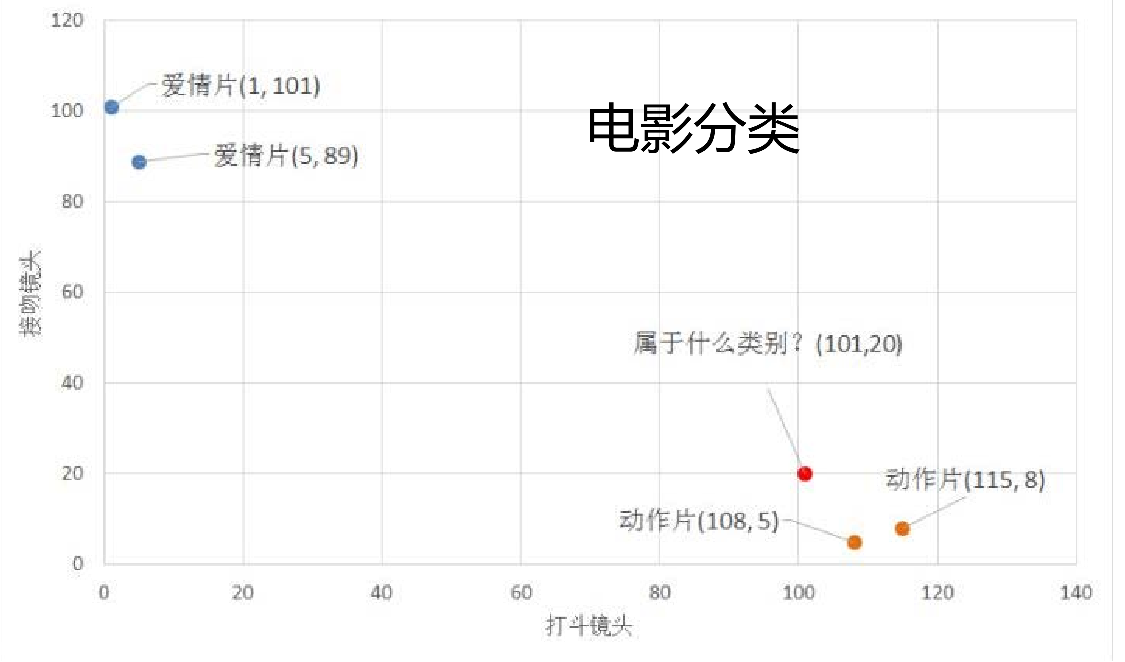
- 分类算法
- 监督学习
- 数据集是带Label的数据
- 没有明显的训练过程，Memory-based learning
- k值含义 - 对于一个样本**x**，要给它分类，首先从数据集中，在**x**附近找**距离**它最近的k个数据点，将它划分为归属于类别最多的一类。

8.1 距离度量

欧几里得度量（Euclidean Metric）（也称欧氏距离）

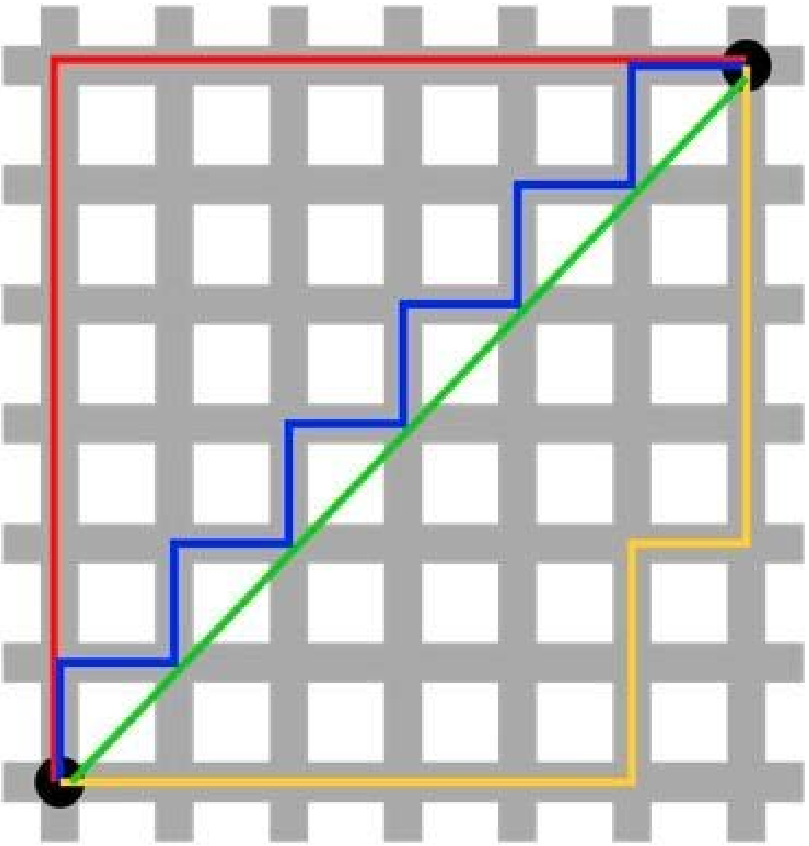
是一个通常采用的距离定义，指在m维空间中两个点之间的真实距离，或者向量的自然长度（即该点到原点的距离）。在二维和三维空间中的欧氏距离就是两点之间的实际距离。

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



曼哈顿距离(Manhattan distance)

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)。

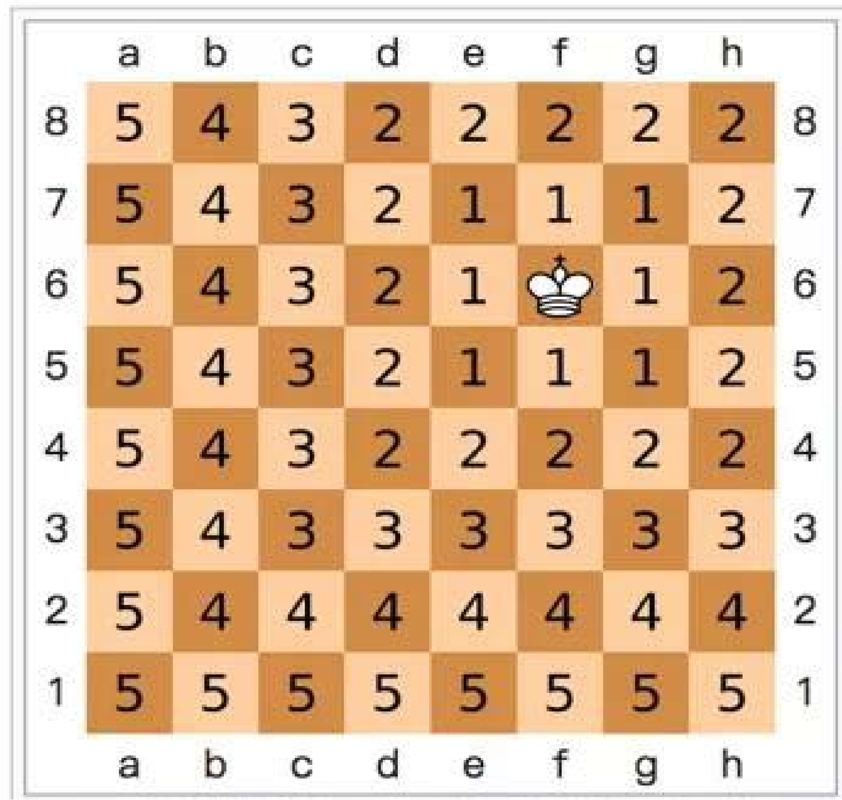


$$d(x,y)=\sqrt{\sum_i (x_i-y_i)^2}$$

切比雪夫距离(Chebyshev distance)

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。
- 右图是棋盘上所有位置距**f6**位置的切比雪夫距离。



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

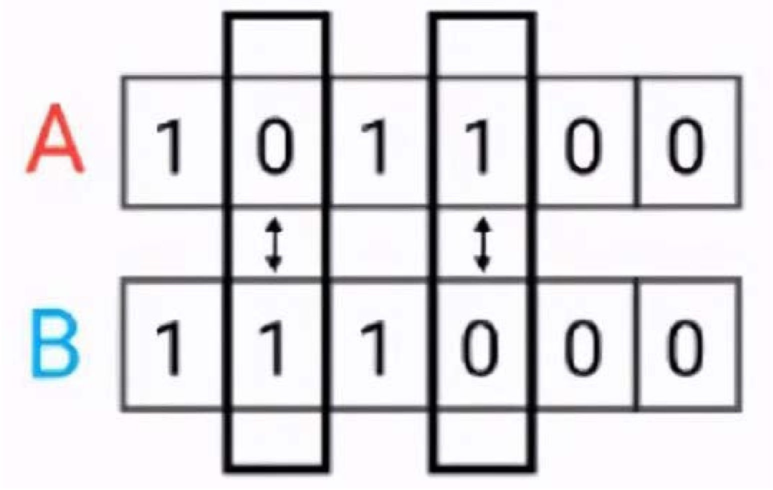
闵可夫斯基距离(Minkowski distance)

- 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。

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

汉明距离(Hamming distance)

- 使用在数据传输差错控制编码中。
- 是一个概念，它表示两个（相同长度）字对应位不同的数量。
- 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。



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

马氏距离(Mahanlanobis Distance)

- 马氏距离是由印度统计学家马哈拉诺比斯(P. C. Mahalanobis)提出的，表示数据的协方差距离。
- 它是一种有效的计算两个未知样本集的相似度的方法。与欧式距离不同的是它考虑到各种特性之间的联系（例如：一条关于身高的信息会带来一条关于体重的信息，因为两者是有关联的），并且是尺度无关的(scale-invariant)，即独立于测量尺度。
- 马氏距离不受量纲的影响，两点之间的马氏距离与原始数据的测量单位无关；由标准化数据和中心化数据(即原始数据与均值之差）计算出的二点之间的马氏距离相同。
- 马氏距离还可以排除变量之间的相关性的干扰。它的缺点是夸大了变化微小的变量的作用。

协方差：

$$C = \begin{bmatrix} C_{11} \cdots C_{1n} \\ \vdots \\ C_{n1} \cdots C_{nn} \end{bmatrix} = E[(X-m)(X-m)^T]$$

C为对称阵且正定 $C_{ij} = E[(x_i - m_i)(x_j - m_j)]$

$$d^2 = (X-m)^T C^{-1}(X-m) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}(x_i - m_i)(x_j - m_j) \geq 0$$

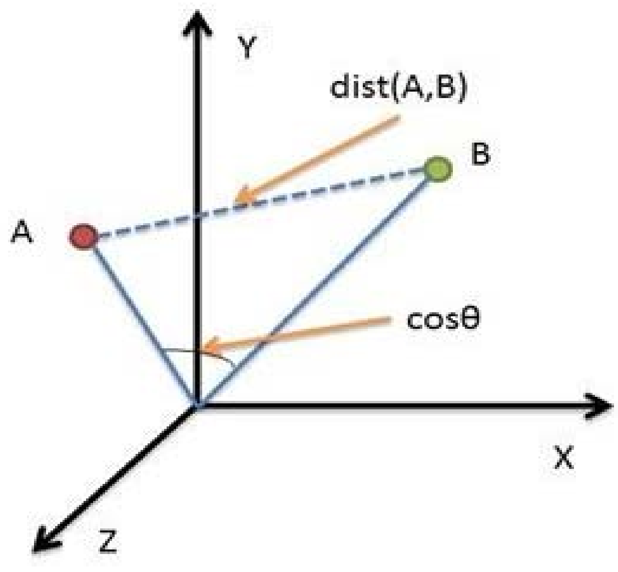
C为单位矩阵式马氏距离就变为欧氏距离。

余弦相似度

➢给定两个n维向量，A=$[A_1, A_2, \ldots, A_n]$，B=$[B_1, B_2, \ldots, B_n]$，则A和B的夹角的余弦等于

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

- 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。



目录

算法的主要思路：

- k近邻法（k-Nearest Neighbor, k-NN）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。

- 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。

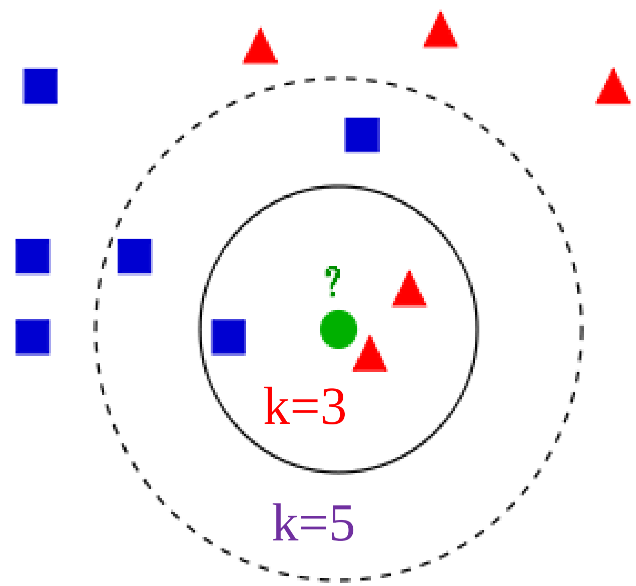
- 对于回归问题：对新的样本，根据其k个最近邻的训练样本标签值的均值作为预测值。

k近邻法的三要素：

- k值选择。
- 距离度量。
- 决策规则。

K值选择的意义

- 从图中可以看到，数据集是打好了label的数据，一类是蓝色的正方形，一类是红色的三角形，绿色的圆形我们待分类的数据。
- 如果k=3，那么离绿色点最近的有2个红色三角形和1个蓝色的正方形，这3个点投票，于是绿色的这个待分类点属于红色的三角形。
- 如果k=5，那么离绿色点最近的有2个红色三角形和3个蓝色的正方形，这5个点投票，于是绿色的这个待分类点属于蓝色的正方形。



😷 如果选择较小的K值

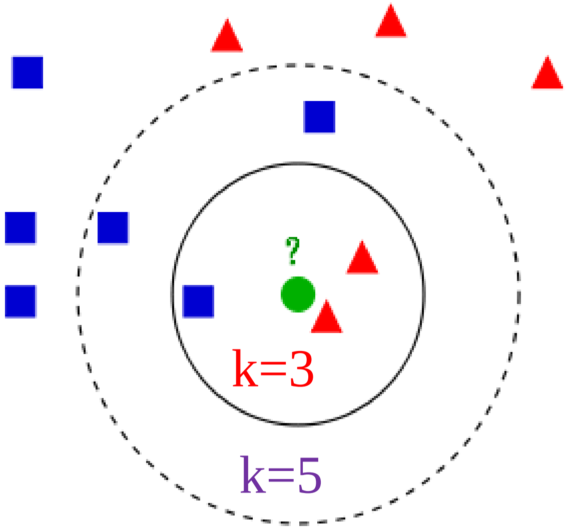
- “学习”的近似误差（approximation error)会减小，但 “学习”的估计误差（estimation error）会增大；
- 噪声敏感；
- 意味着整体模型变得复杂，容易发生过拟合。

😷 如果选择较大的K值

- 减少学习的估计误差，但缺点是学习的近似误差会增大；
- 意味着整体的模型变得简单。

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。



目录

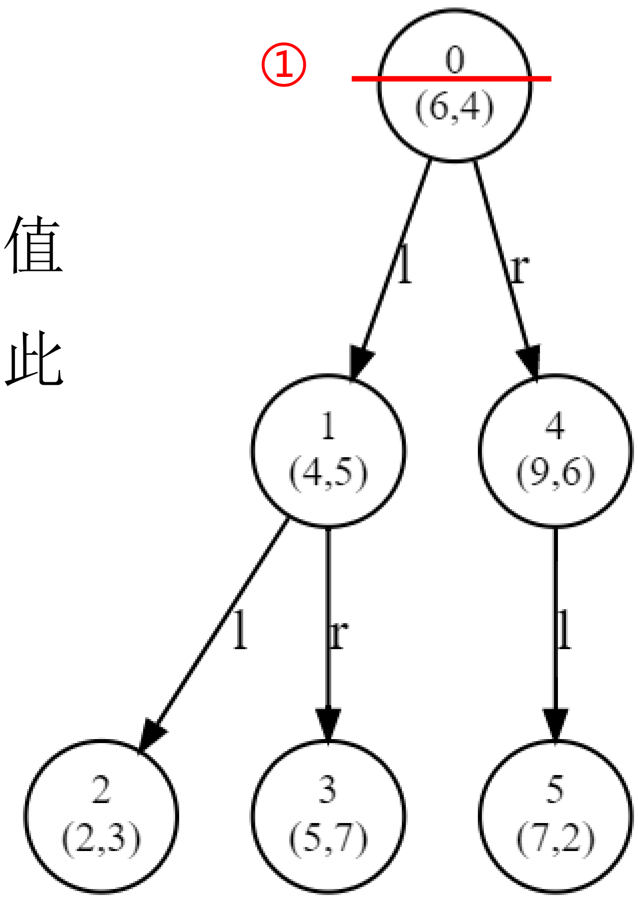
KD树(K-Dimension Tree)，也称之为K维树，可以用来更高效地对空间进行划分，并且其结构非常适合寻找最近邻居和碰撞检测。

- 一种对K维空间中的实例点进行快速检索的树形数据结构，是二叉树，表示对K维空间的一个划分。
- 构造KD树相当于不断地用垂直于坐标轴的超平面将K维空间切分，构成一系列的K维超矩形区域，每个结点对应的于一个K维超矩形区域。

假设有 6 个二维数据点，

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

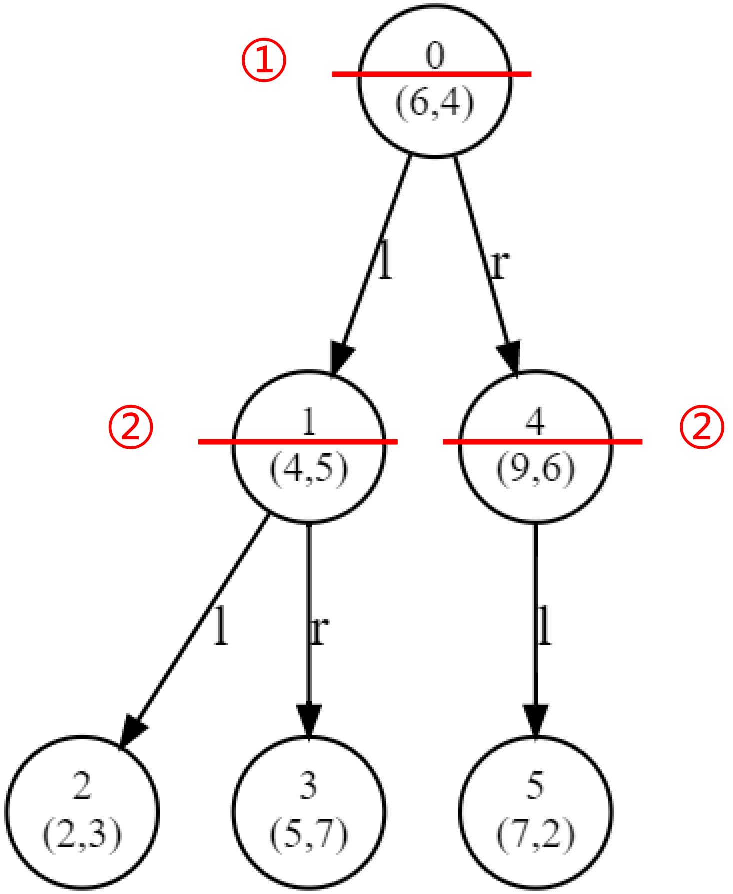
构建KD树的过程如下：



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：$x=6$ 。

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：$y=6$。



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

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：

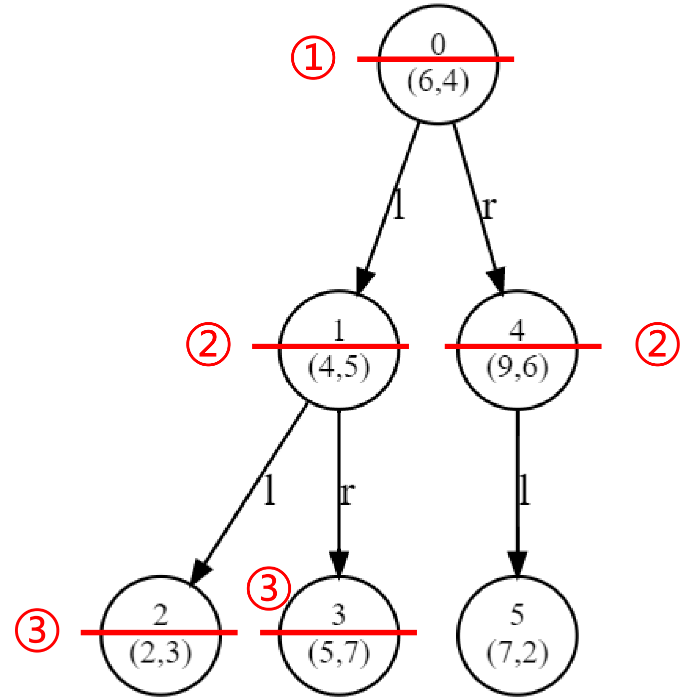
$x = 2$。

➤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。



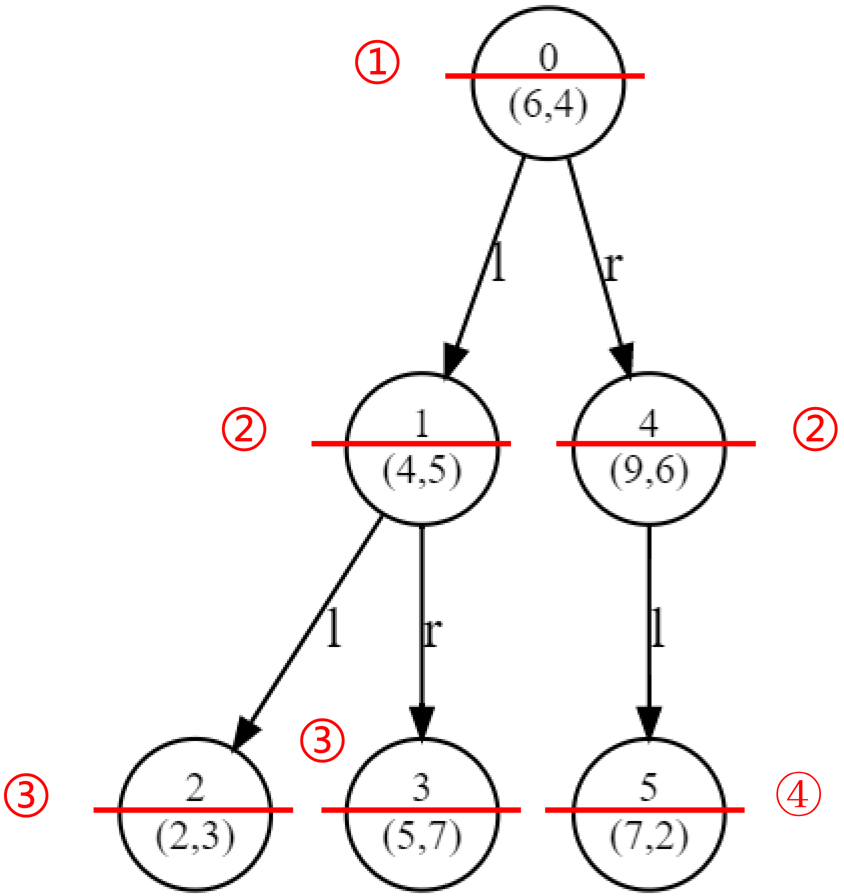
D= {(2, 3), (5, 7), (9, 6), (4, 5), (6, 4), (7, 2)}

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} 。

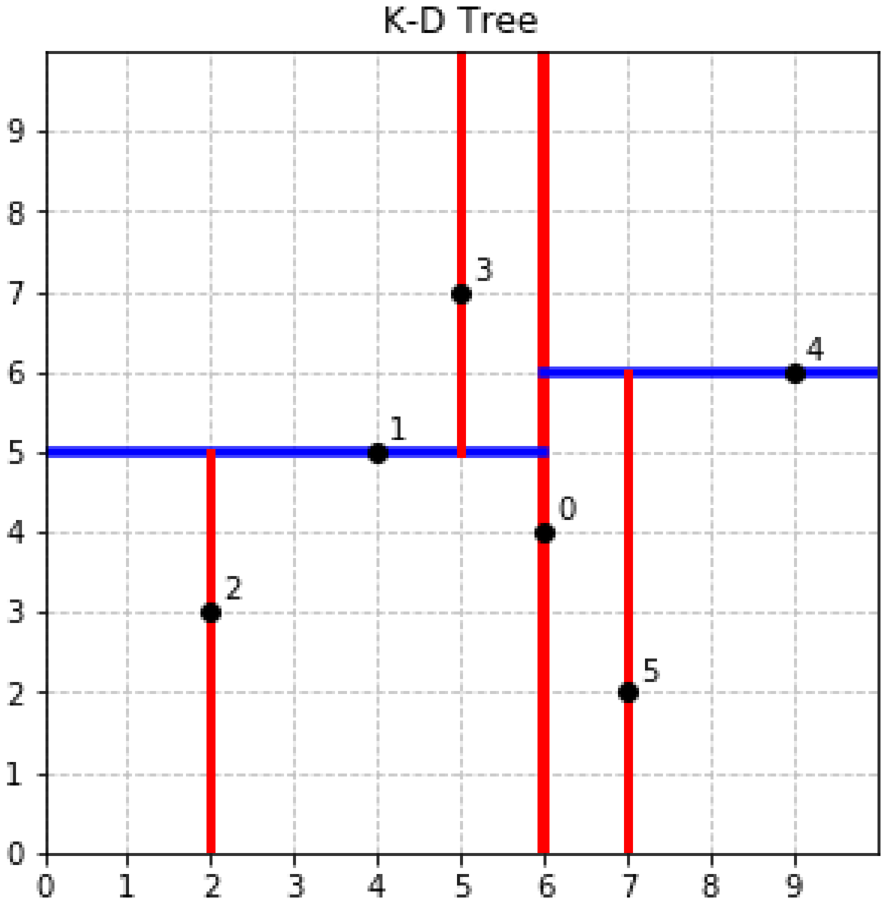
由于D_{11}, D_{12} 都不包含任何点，因此对它们不再继续拆分。

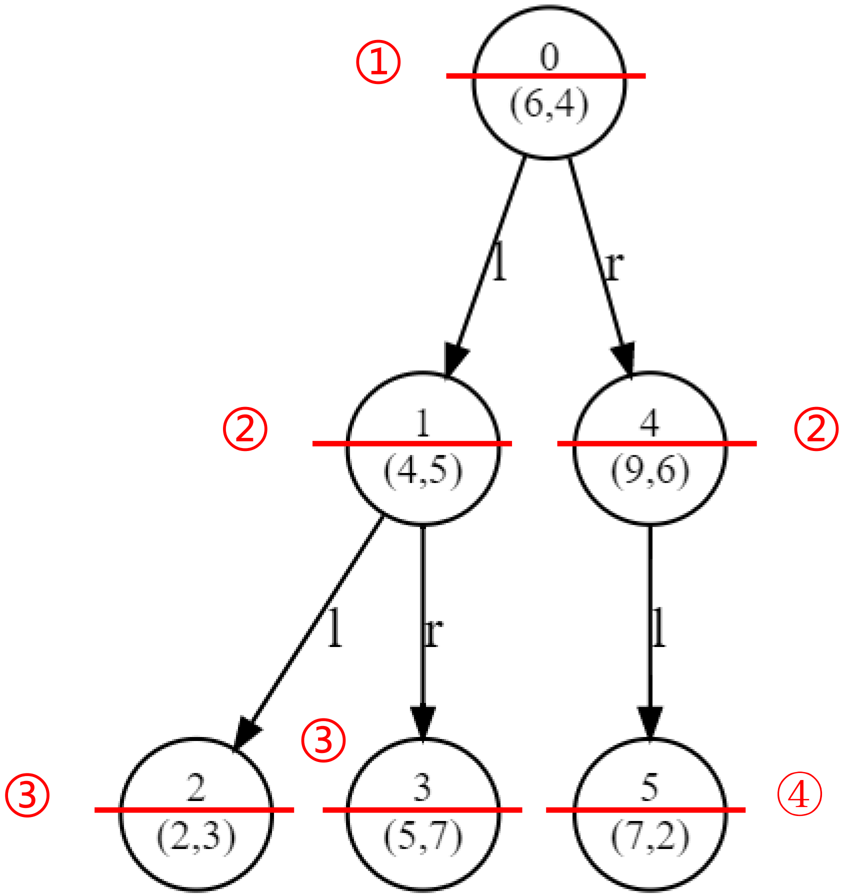
D_2的右子空间（记做 D_6）不包含任何点，停止继续拆分。



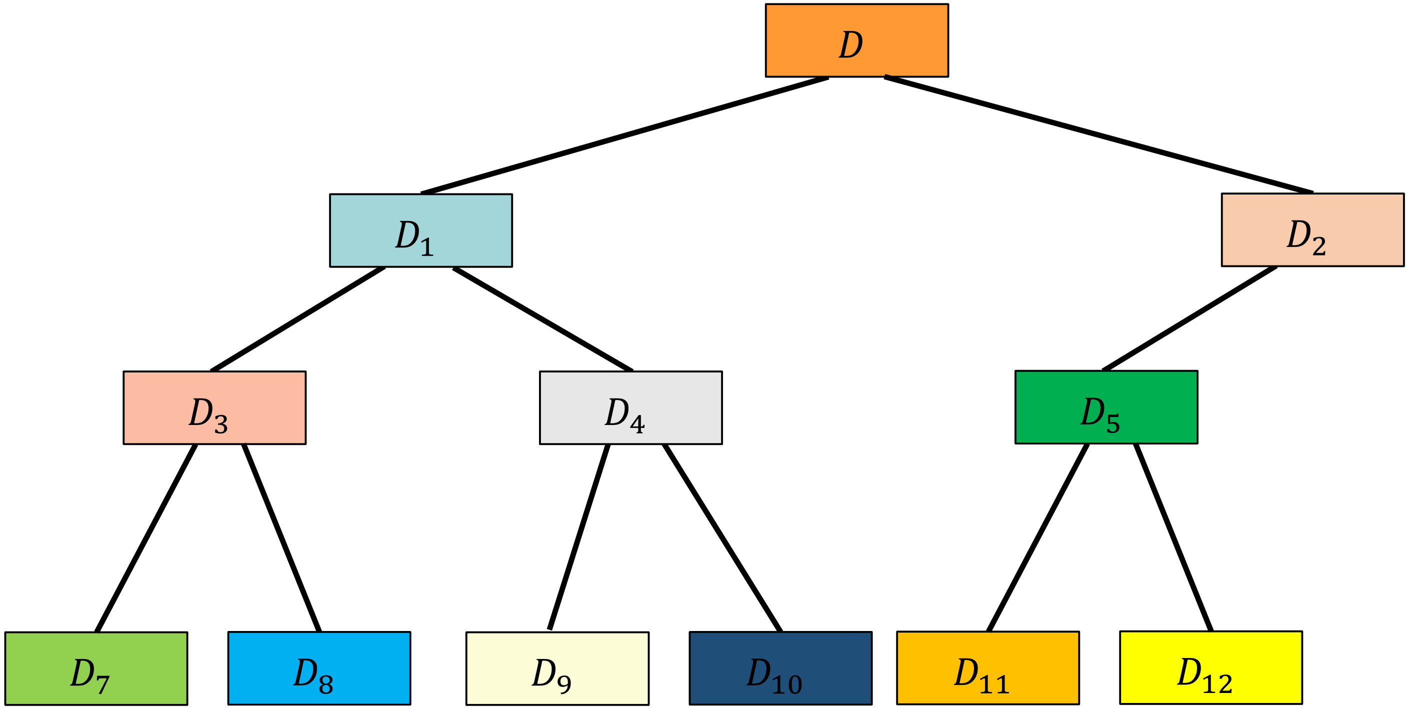
$D=\{(2,3),(5,7),(9,6),(4,5),(6,4),(7,2)\}$

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



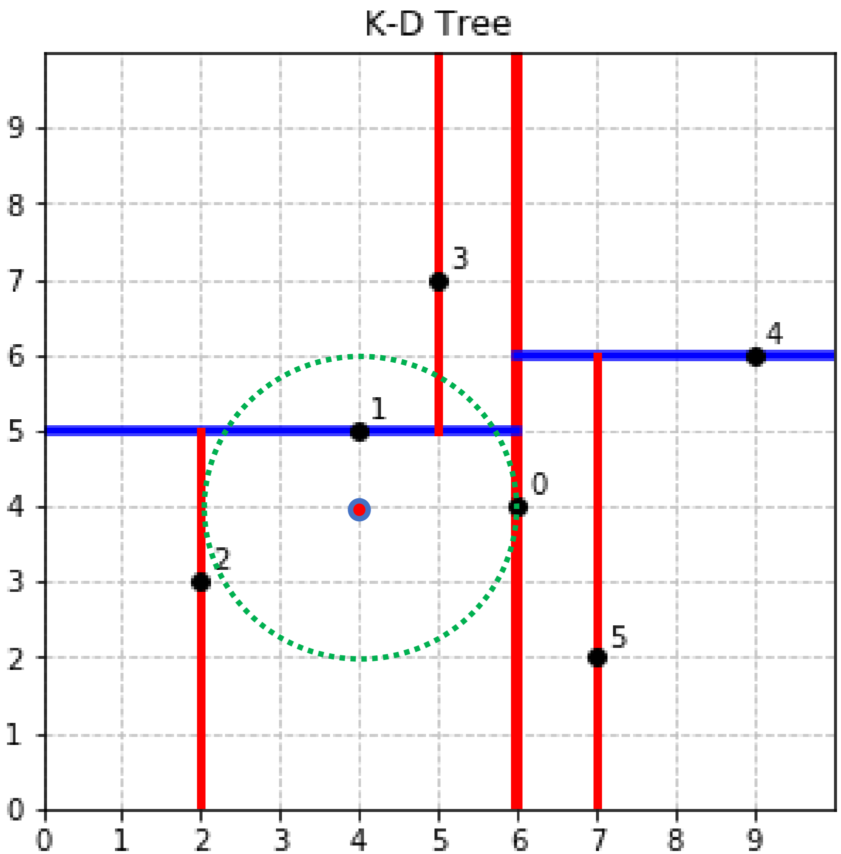


样本空间结构图



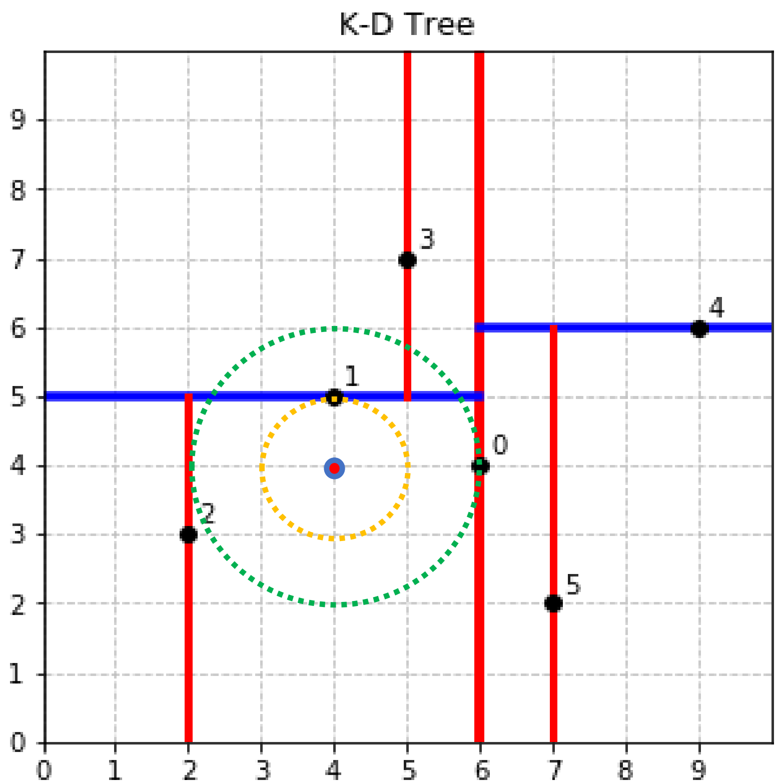
目录

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。

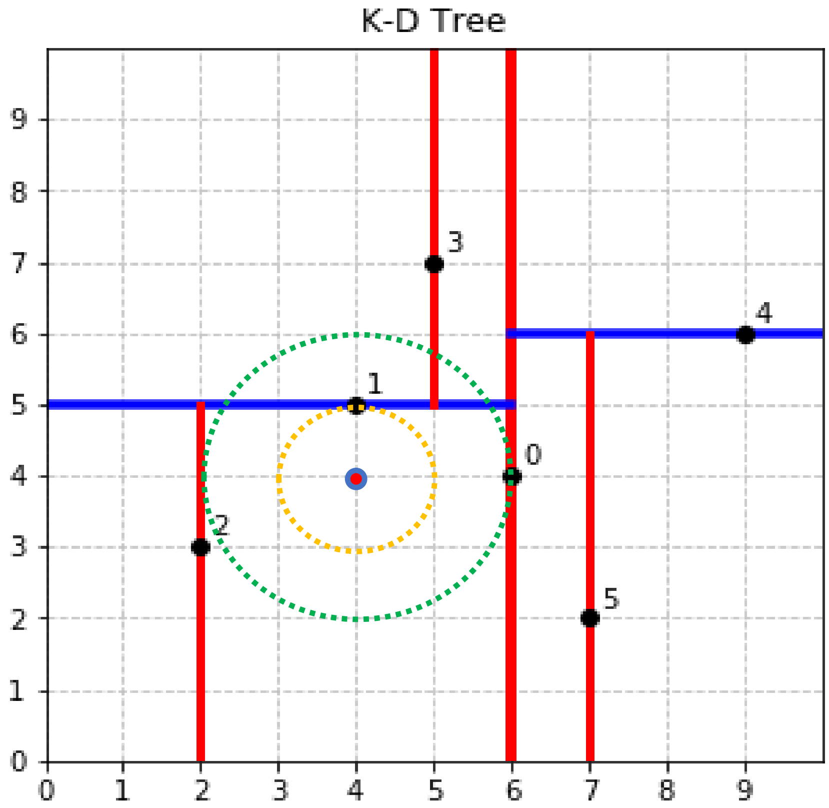
2.返回叶子结点的父节点，检查另一个子结点包含的超矩形体是否和超球体相交，如果相交就到这个子节点寻找是否有更加近的近邻, 有的话就更新最近邻。



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。

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）。

k-NN近邻法总结：

- k-NN是一种 memory-based learning，也叫instance-based learning，属于lazy learning。即它没有明显的前期训练过程，而是程序开始运行时，把数据集加载到内存后，不需要进行训练，就可以开始分类。
- k-NN本质是基于一种数据统计的方法。
- 参数k常常根据经验人为设定。
- 对于二分类问题，k一般取奇数，以避免因两种票数相同时出现难以决策的局面。

K-NN的缺点：

- 每次决策都要计算待识别样本与与全部训练样本之间的距离。
- 并且需要存储整个样本集。
- 寻找近邻的过程所需要的计算量较大。

谢 谢！

本课件制作过程中，多处引用了国内外同行的网页、教材、以及课件PPT的内容或图片，没有随处标注，特此说明，并在此向各位作者表示感谢！