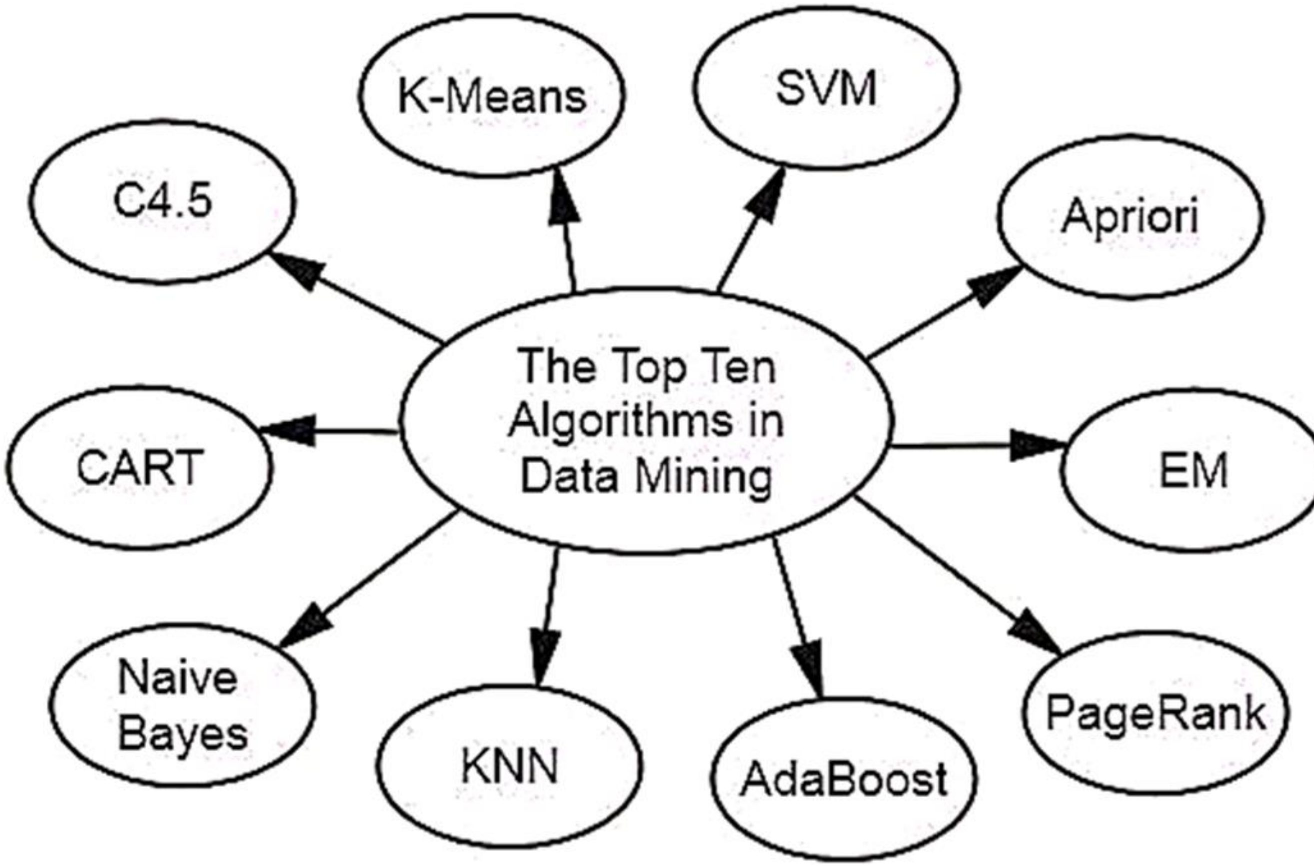
第四章：k-近邻
Ch4: k-nearest neighbor

概述



The Top Ten Algorithms in Data Mining
C4.5
K-Means
SVM
Apriori
CART
EM
Naive Bayes
KNN
AdaBoost
PageRank

概述

*k*近邻法（*k*-nearest neighbor, *k*-NN）1968年由Cover和Hart提出。

*k*近邻法是一种基本**分类**与**回归**方法。*k*近邻法的**输入为实例的特征向量**，对应于特征空间的点；**输出为实例的类别**，可以取多类。近邻法假设给定一个训练数据集，其中的实例类别已定。分类时，对新的实例，根据其*k*个最近邻的训练实例的类别，通过**多数表决**等方式进行预测。因此，*k*近邻法**不具有显式**的学习过程（也称为**Lazy learning**）。*k*近邻法实际上利用训练数据集对特征向量空间进行划分，并作为其分类的“模型”。***k*值的选择**、**距离度量**及**分类决策规则**是*k*近邻法的三个基本要素。

k近邻法的思维：

近朱者赤，近墨者黑

分类问题

熊出没重返地球 7.3
硬核科幻带你遨游宇宙

雄狮少年粤语版 8.1
舞狮少年追梦为自己而战！

五个扑水的少年 9.0
热血高中男生挑战花样游泳

我和我的祖国 9.2
燃爆了！为祖国疯狂打call

怒火重案 9.2
甄子丹谢霆锋激战厮杀

再见少年 8.9
张子枫演绎少年殊途

1921 9.3
百星演绎！致敬百年征程！

新逃学威龙 9.1
张浩英雄救美

狗果定理 8.7
于谦贾冰邂逅“狗界”大佬

急先锋 8.9
成龙杨洋艾伦火力全开！

盛夏未来 9.2
吴磊张子枫交换青春秘密

悬崖之上 9.4
张艺谋首次尝试谍战大片！

动作片、剧情片、艺术片、古装片、科幻片、爱情片、...

影片分类

序号	电影名称	搞笑镜头	拥抱镜头	打斗镜头	电影类型
1	功夫熊猫	39	0	31	喜剧片
2	叶问3	3	2	65	动作片
3	二次曝光	2	3	55	爱情片
4	代理情人	9	38	2	爱情片
5	新步步惊心	8	34	17	爱情片
6	谍影重重	5	2	57	动作片
7	美人鱼	21	17	5	喜剧片
8	宝贝当家	45	2	9	喜剧片
9	唐人街探案	23	3	17	?

本讲内容

- ☐ k近邻算法
- ☐ k近邻法模型及三要素
- ☐ kd树：kd树构造和kd树搜索

k近邻算法

k近邻算法简单、直观：给定一个训练数据集，对新的输入实例，在训练数据集中找到与该实例邻近的k个实例，这k个实例的多数属于某个类，就把该输入实例分为这个类。

k近邻法算法

算法3.1 （k近邻法）

输入：训练数据集

$$T=\{(x_1,y_1),(x_2,y_2),...,(x_N,y_N)\}$$

其中，$x_i\in\mathcal{X}\subseteq\mathbf{R}^n$为实例的特征向量，$y_i\in\mathcal{Y}=\{c_1,c_2,...,c_K\}$为实例的类别，$i=1,2,...,N$；实例特征向量$x$。

输出：实例x所属的类y。

(1) 根据给定的距离度量，在训练集T中找出与x最邻近的k个点，涵盖这k个点的x的邻域记做$N_k(x)$。

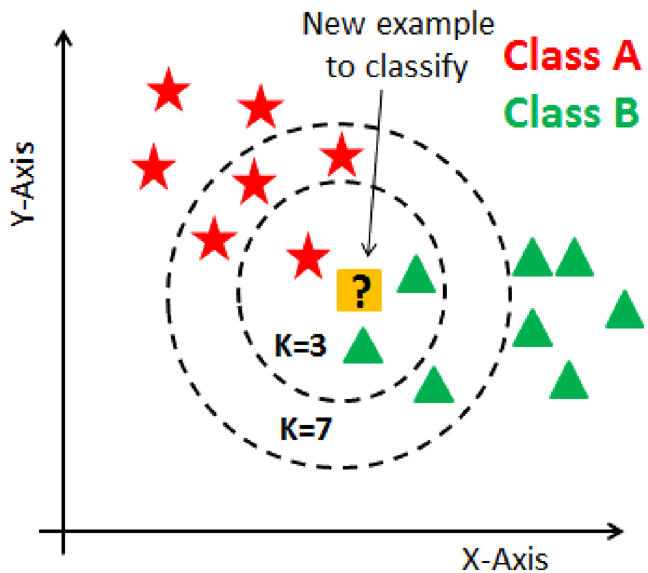
(2) 在$N_k(x)$中根据分类决策规则（如多数表决）决定x的类别y：

$$y=arg\max_{c_j}\sum_{x_i\in N_k(x)}I(y_i=c_j),i=1,2,...,N;j=1,2,...,K \quad (3.1)$$

式（3.1）中，I为指示函数，即当$y_i=c_j$时I为1，否则I为0。

k近邻法算法

下图中有两种类型的样本数据，一类是**红色的五角星**，另一类是绿色的三角形，中间那个**黄色**的圆形是待分类数据：



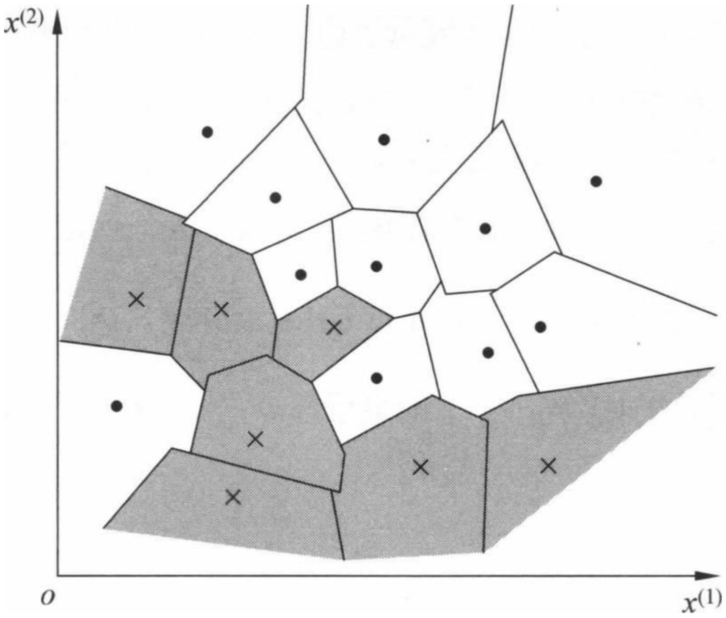
k=1，称为**最近邻**算法：对于输入的实例点*x*，将训练数据集中与*x*最邻近点的类作为*x*的类。

如果*k*=3，那么离黄色点最近的有2个绿色的三角形和1个红色五角星，这3个点进行投票，于是黄色的待分类点就属于**绿色**的三角形。

如果*k*=7，那么离黄色点最近的有4个红色的五角星和3个绿色的三角方形，这7个点进行投票，于是黄色的待分类点就属于**红色**的五角星。

K近邻模型

*k*近邻法使用的模型实际上对应于对**特征空间的划分**。



voronoi图 (最近邻举例)

模型由三个基本要素：**距离度量**、***k*值**的选择和**分类决策规则**决定。

度量距离

特征空间中两个实例点的**距离**是两个实例点**相似程度**的反映。k近邻模型的特征空间一般是n维实数向量空间$\boldsymbol{R}^n$。使用的距离是欧氏距离，但也可以是其他距离，如更一般的L_p距离(L_p distance)或Minkowski距离(Minkowski distance) 。

设特征空间n维实数向量空间 $\boldsymbol{R}^n$, x_i, $x_j \in X$, $x_i = \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right)^T$, $x_j = \left(x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(n)}\right)^T$, x_i, x_j 的L_p距离定义为，

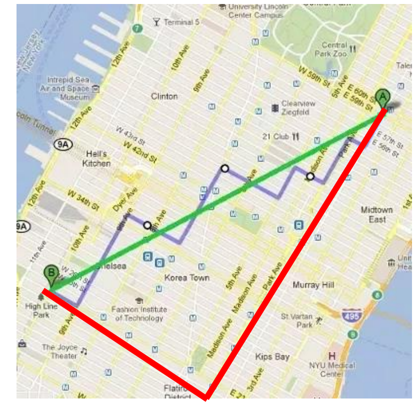
$$L_p(x_i, x_j) = \left(\sum_{l=1}^{n} |x_i^{(l)} - x_j^{(l)}|^p\right)^{\frac{1}{p}}$$

其中，$p \geq 1$

度量距离

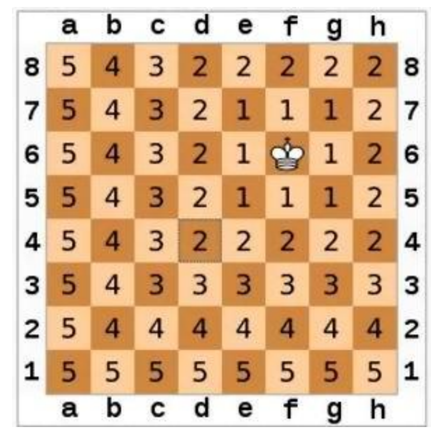
当 p = 2 时，称为欧氏距离 (Euclidean distance)，即，

$$L_2(x_i, x_j) = \left(\sum_{l=1}^{n} |x_i^{(l)} - x_j^{(l)}|^2 \right)^{\frac{1}{2}} \quad (3.3)$$



当 p = 1 时，称为曼哈顿距离 (Manhattan distance)，即，

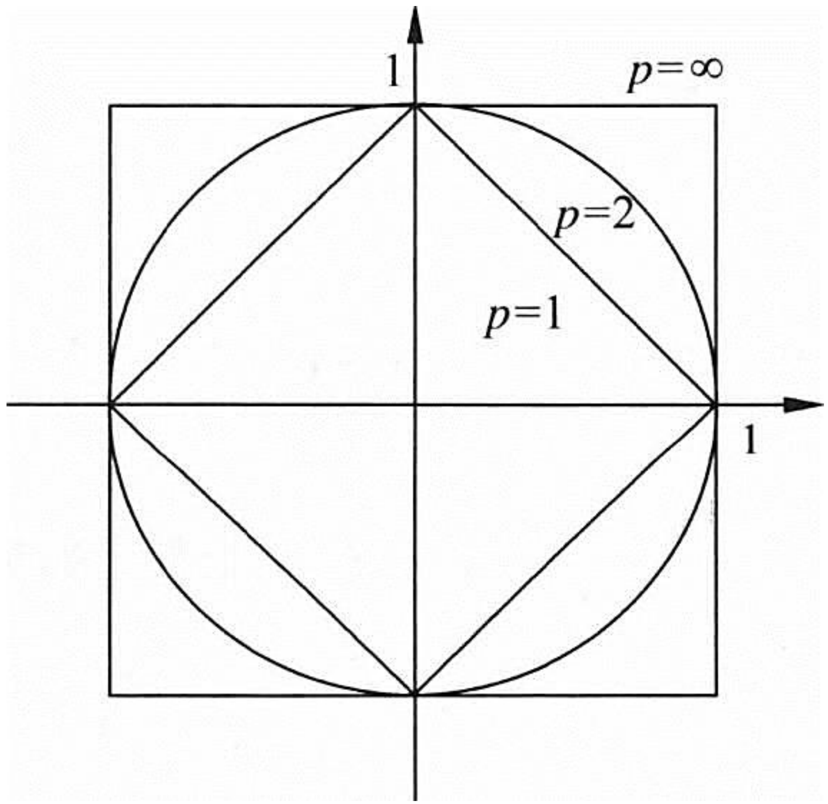
$$L_1(x_i, x_j) = \sum_{l=1}^{n} |x_i^{(l)} - x_j^{(l)}| \quad (3.4)$$



当 p = ∞时，成为切比雪夫距离 (Chebyshev distance), 它是各个坐标距离的最大值，即，

$$L_\infty(x_i, x_j) = max_l |x_i^{(l)} - x_j^{(l)}| \quad (3.5)$$

度量距离



二维空间中p取不同值时，与原点的L_p距离为1(L_p=1)的点的图形

度量距离

例题 3.1： 已知二维空间的3个点 $x_1=(1, 1)^T$, $x_2=(5, 1)^T$, $x_3= (4, 4)^T$, 试求在p取不同值时 L_p 距离下x_1的最近邻点。

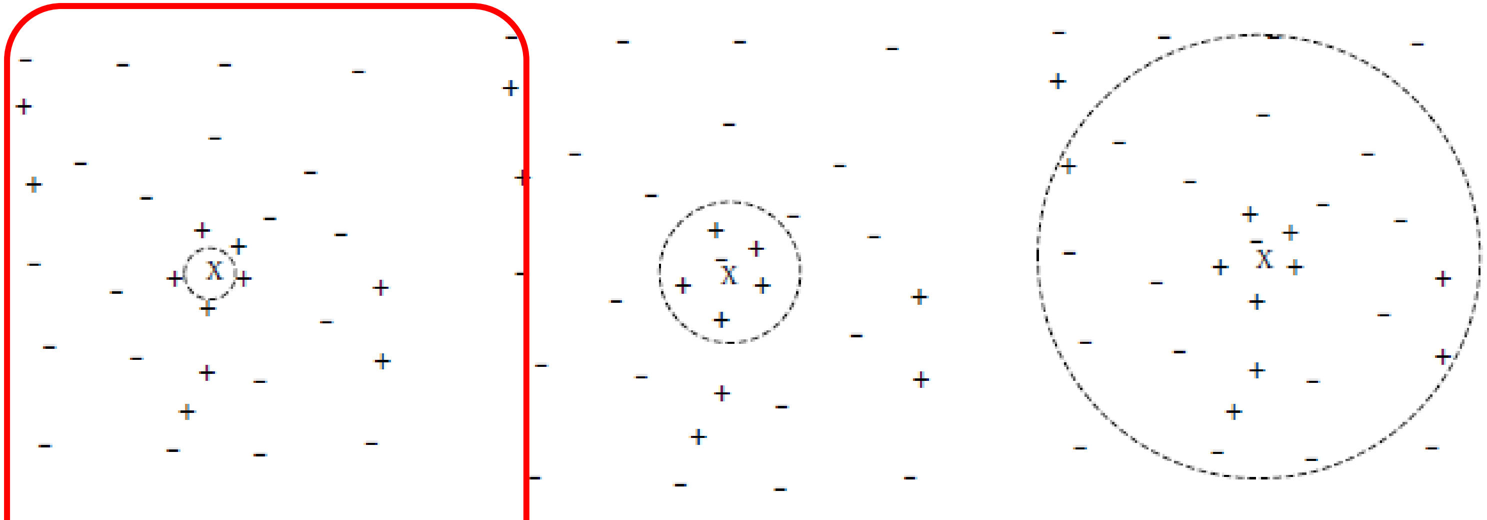
解： 因为x_1和x_2只有第一维的值不同，所以p为任何值时，$L_p(x_1, x_2)=4$。而$L_1(x_1, x_3)=6$, $L_2 (x_1, x_3)=4.24$, $L_3(x_1, x_3)=3.78$, $L_4(x_1, x_3)=3.57$

于是得到：p等于1或者2时，x_2是x_1的最近邻点；p大于等于3时，x_3是x_1的最近邻点。

上面的例子说明，由**不同的距离度量所确定的最近邻点是不同的**。

k值的选择

*k*值的选择也会对*k*近邻法的结果产生重大影响



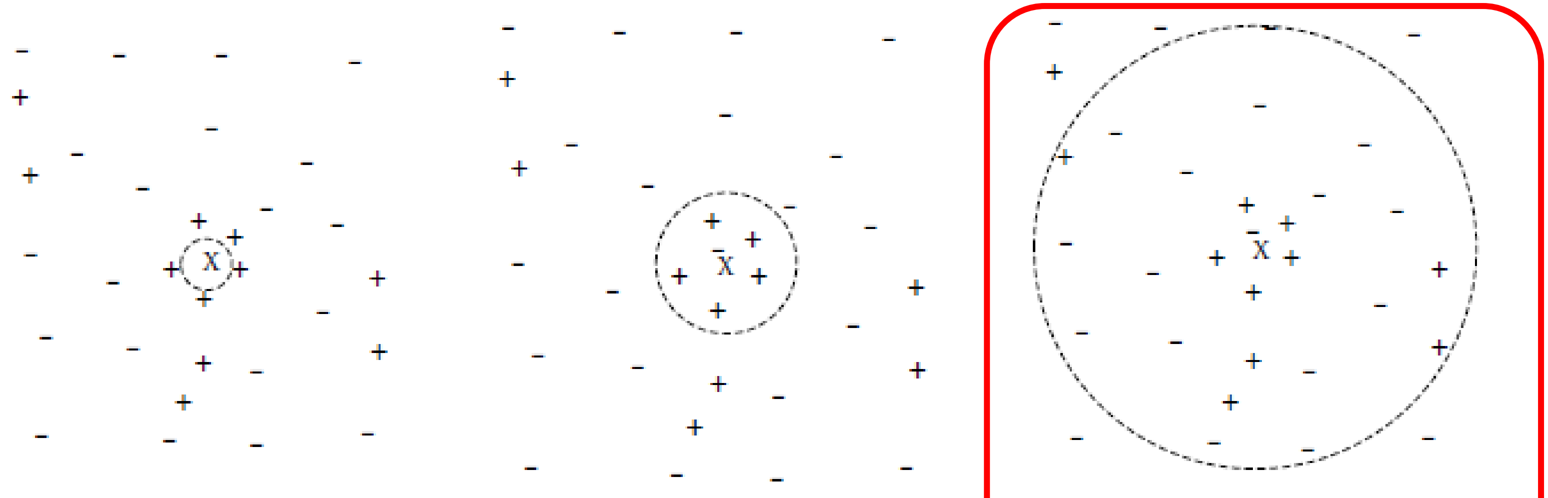
(a) Neighborhood too small.

(b) Neighborhood just right.

(c) Neighborhood too large.

*k*值的减小就意味着整体模型变得复杂，容易发生过拟合

k值的选择



(a) Neighborhood too small.

(b) Neighborhood just right.

(c) Neighborhood too large.

k值的增大就意味着整体的模型变得简单

问题1： 如何选择最优k？

问题2： k 为奇数还是偶数？

分类决策规则

k近邻法中的分类决策规则往往是**多数表决**，即由输入实例的k个邻近的训练实例中的多数类决定输入实例的类。

多数表决规则（majority voting rule）有如下解释：如果分类的损失函数为0-1失函数，分类函数为，

$$f:\mathbf{R}^n \to \{c_1, c_2, ..., c_K\}$$

那么误分类的概率是：

$$P(Y \neq f(X)) = 1 - P(Y = f(X))$$

分类决策规则

对给定的实例$x \in \mathcal{X}$，其最近邻的k个训练实例点构成集合$N_k(x)$。如果涵盖$N_k(x)$的区域的类别是c_j，那么误分类率是

$$\frac{1}{k}\sum_{x_i \in N_k(x)} I(y_i \neq c_j) = 1 - \frac{1}{k}\sum_{x_i \in N_k(x)} I(y_i = c_j)$$

要使误分类率最小即经验风险最小，就要使$\sum_{x_i \in N_k(x)} I(y_i = c_j)$最大，所以<u>**多数表决规则等价于经验风险最小化**</u>。

近邻法的实现：kd树

实现*k*近邻法时，主要考虑的问题是如何对训练数据进行**快速*k*近邻搜索。**这点在特征空间的维数（**n**）大及训练数据容量（**N**）大时尤其必要。

*k*近邻法最简单的实现方法是线性扫描（linear scan）。这时要计算输入实例与每一个训练实例的距离。当训练集很大时，计算非常耗时，这种方法是不可行的。

为了**提高近邻搜索的效率**，可以考虑使用特殊的结构存储训练数据，以减少计算距离的次数。具体方法很多，下面介绍其中的*kd*树（*k*-dimension tree）方法。

构造kd树

*kd*树是一种对*k*维空间中的**实例点进行存储**以便对其进行快速检索的树形数据结构。***kd*树是二叉树**，表示对*k*维空间的一个划分（partition）。

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。*kd* 树的每个结点对应于一个*k*维超矩形区域。

通常，依次选择坐标轴对空间切分，选择训练实例点在选定坐标轴上的中位(median)为切分点，这样得到的*kd*树是平衡的。注意，平衡的*kd*树搜索时的效率未必是最优的。

构造平衡kd树

输入： k维空间数据集$T=\{x_1,x_2,...,x_N\}$，$x_i=\left(x_i^{(1)},x_i^{(2)},...,x_i^{(k)}\right)^T, i=1,2,...,N;$

输出： kd树

(1) 开始：构造根结点，根结点对应于包含T的k维空间的超矩形区域。

选择$x^{(1)}$为坐标轴，以T中所有实例的$x^{(1)}$坐标的中位数为切分点，将根结点对应的超矩形区域切分为两个子区域。切分由通过切分点并与坐标轴$x^{(1)}$垂直的超平面实现。

由根结点生成深度为 1 的左、右子结点：左子结点对应坐标$x^{(1)}$小于切分点的子区域，右子结点对应于坐标$x^{(1)}$大于切分点的子区域。

将落在切分超平面上的实例点保存在根结点。

构造平衡kd树

输入： *k*维空间数据集$T=\{x_1,x_2,...,x_N\}$，$x_i=\left(x_i^{(1)},x_i^{(2)},...,x_i^{(k)}\right)^T, i=1,2,\dots,N$;

输出： *kd*树。

(2) 重复：对深度为*j*的结点，选择$x^{(l)}$为切分的坐标轴，$l = j(\text{mod } k)+1$，以该结点的区域中所有实例的$x^{(l)}$坐标的中位数为切分点，将该结点对应的超矩形区域切分为两个子区域。切分由通过切分点并与坐标轴$x^{(l)}$垂直的超平面实现。

由该结点生成深度为*j*+1的左、右子结点：左子结点对应坐标$x^{(l)}$小于切分点的子区域，右子结点对应坐标$x^{(l)}$大于切分点的子区域。

将落在切分超平面上的实例点保存在该结点。

(3) 直到两个子区域没有实例存在时停止。从而形成*kd*树的区域划分。

例子

例子3.2： 给定二维空间的数据集：

$T=\{(2,3)^T,(5,4)^T,(9,6)^T,(4,7)^T,(8,1)^T,(7,2)^T\}$

构造一个平衡 *kd* 树。

解： 根结点对应包含数据集T的矩形，选择$x^{(1)}$轴，6个数据点的$x^{(1)}$坐标的中位数是7，以平面$x^{(1)}=7$将空间分为左、右两个子矩形(子结点)；接着，左矩形以$x^{(2)}=4$分为两个子矩形，右矩形以$x^{(2)}=6$分为两个子矩形，如此递归，最后得到如图3.3所示的特征空间划分和如图3.4所示的*kd*树。

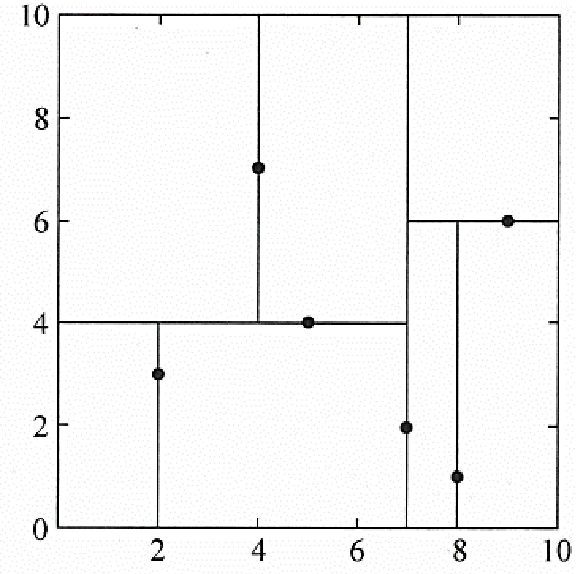


图 3.3　特征空间划分

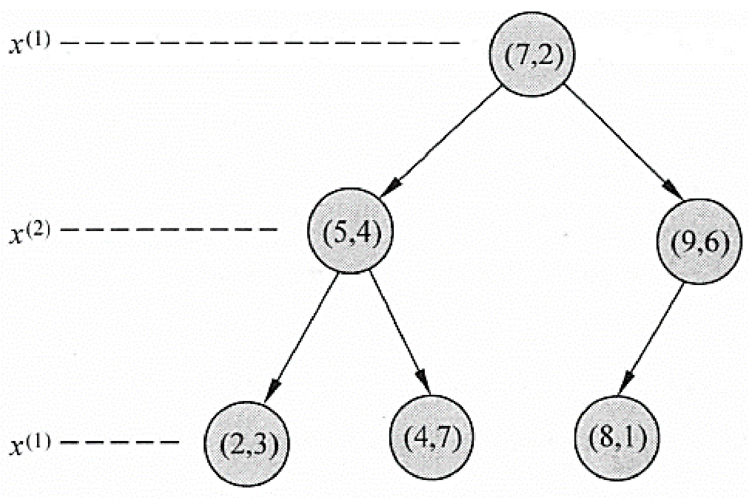


图 3.4　*kd* 树示例

搜索kd树

利用*kd*树可以省去对大部分数据点的搜索，从而减少搜索的计算量。

给定一个目标点，搜索其最近邻。首先找到包含目标点的叶结点；然后从该叶结点出发，依次回退到父结点；不断查找与目标点最邻近的结点，当确定不可能存在更近的结点时终止。这样搜索就被限制在空间的局部区域上，效率大为提高。

包含目标点的叶结点对应包含目标点的最小超矩形区域。以此叶结点的实例点作为当前最近点。目标点的最近邻一点在以目标点为中心并通过当前最近点的超球体的内部。然后返回当前结点的父结点，如果父结点的另一子结点的超矩形区域与超球体相交，那么在相交的区域内寻找与目标点更近的实例点。如果存在这样的点，将此点作为新的当前最近点。算法转到更上一级的父结点，继续上述过程。如果父结点的另子结点的超矩形区域与超球体不相交，或不存在比当前最近点更近的点，则停止搜索。

kd 树的最近邻搜索算法

算法3.3： 用*kd*树的最近邻搜索

输入： 已构造的*kd*树，目标点x

输出： x的最近邻

① 在*kd*树中找出包含目标点x的叶结点：从根结点出发，递归地向下访问*kd*树。若目标点x当前维的坐标小于切分点的坐标，则移动到左子结点，否则移动到右子结点。直到子结点为叶结点为止。

② 以此叶结点为“当前最近点”。

kd 树的最近邻搜索算法

算法3.3： 用*kd*树的最近邻搜索

输入： 已构造的*kd*树，目标点*x*

输出： *x*的最近邻

③ 向上回溯，如果该回溯结点保存的实例点比“当前最近点”距离目标点更近，则以该回溯结点为新的“当前最近点”。

④ 检查该回溯点的另一子结点对应的区域是否有更近的点，具体地：

If 如果另一子结点对应的区域与以目标点为球心、以目标点与“当前最近点”间的距离为半径的超球体相交，则可能在另一个子结点对应的区域内存在距目标点更近的点，移动到另一个子结点。接着，在该子节点（子树）递归地进行最近邻搜索（像整个程序开始一样）；

If else， 继续向上回溯。

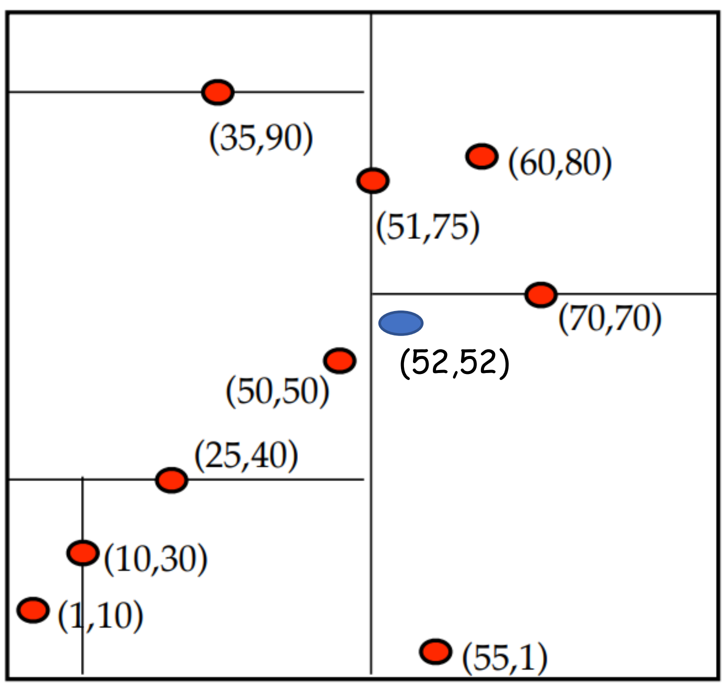
⑤ 当回溯到根结点时，搜索结束。最后的“当前最近点”即为*x*的最近邻点。

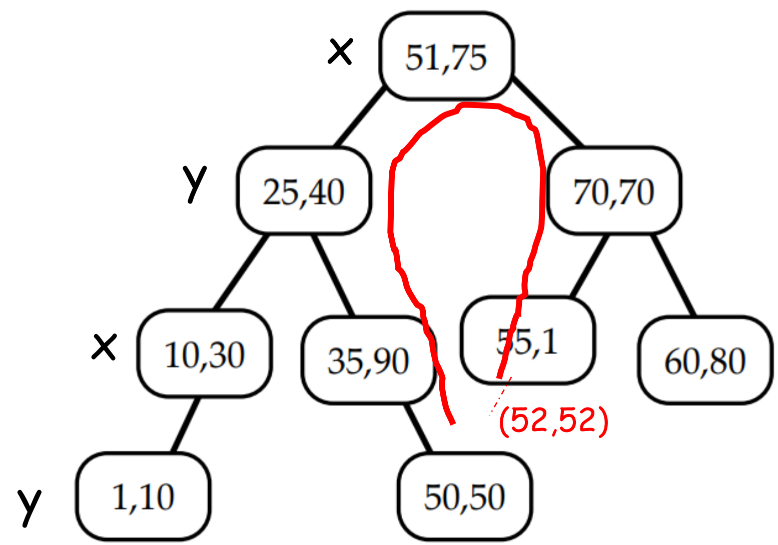
参考：https://yasenh.github.io/post/kd-tree/

为什么回溯到根节点?

因为： 特征空间两个点很近，可能在*kd*树上很远。

比如，点（50,50）和（52,52）空间很近，却分在不同子树上，相距路径很远。





例子

例3.3： 给定一个如图3.5所示的*kd*树，根结点为*A*，其子结点为*B*，*C*等。树上共存储7个实例点；另有一个输入目标实例点*S*，求*S*的最近邻。

解： 首先在*kd*树中找到包含点*S*的叶结点*D*，以点*D*作为近似最近邻。真正最近邻一定在以点*S*为中心通过点*D*的圆的内部。然后返回结点*D*的父结点*B*，在结点*B*的另一子结点*F*的区域内搜索最近邻。结点*F*的区域与圆不相交，不可能有最近邻点。继续返回上一级父结点*A*，在结点*A*的另一子结点*C*的区域内搜索最近邻。结点*C*的区域与圆相交；该区域在圆内的实例点有点*E*，点*E*比点*D*更近，成为新的最近邻近似。最后得到点*E*是点*S*的最近邻。

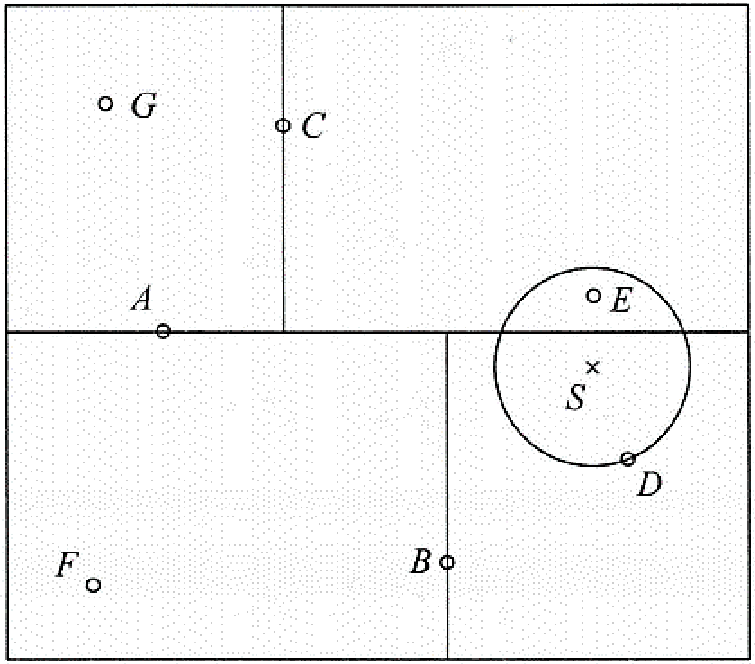
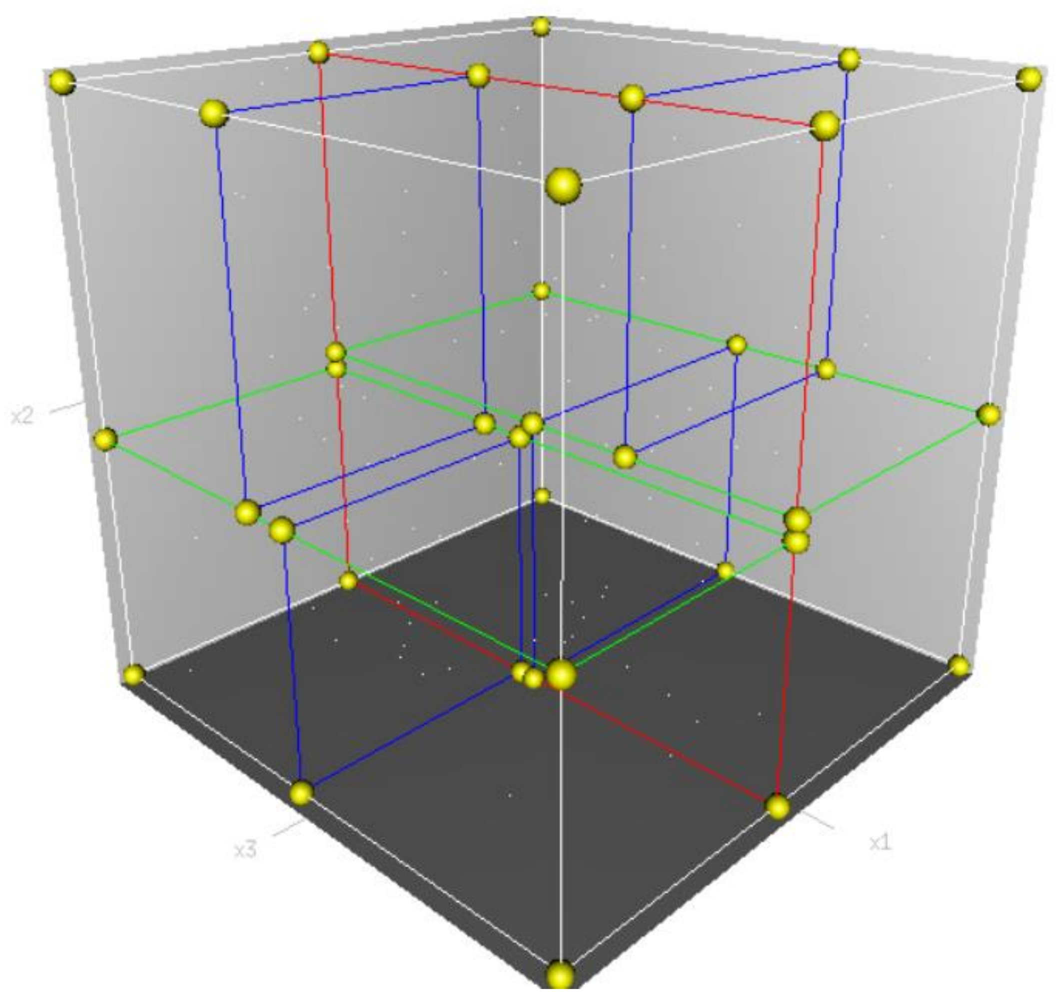


图 3.5　通过 *kd* 树搜索最近邻

三维的例子



课后作业

作业：请根据下列问卷调查，利用K-NN（K=5）算法预测小明（161CM&61KG）应该穿多大尺码的衣服。请给出算法经过。交作业时间：5月16日课堂。

身高（CM）	体重（KG）	T恤尺码
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L