



北京交通大学硕士研究生《机器学习》课件

第10章 K近邻分类模型

不知其子视其父；不知其人视其友；不知其君视其所使；不识其地视其草木。

——《孔子家语·六本》

北京交通大学《机器学习》课程组

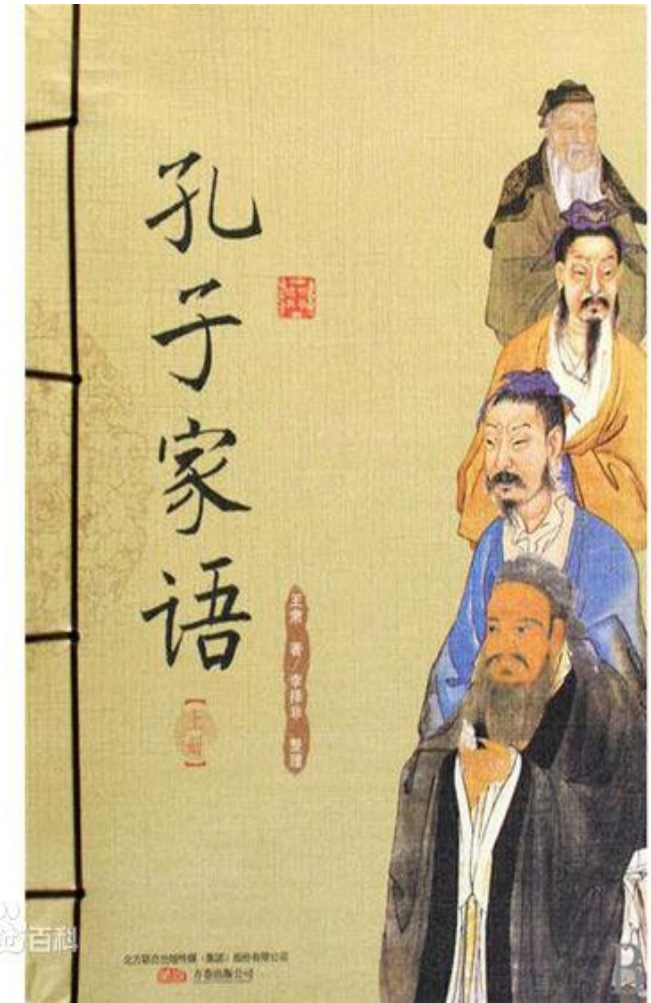




孔子曰：“吾死之后，则商也日益，赐也日损。”

曾子曰：“何谓也？”

子曰：“商也好与贤己者处，赐也好说不若己者。不知其子视其父；**不知其人视其友**；不知其君视其所使；不识其地视其草木。故曰与善人居，如入芝兰之室，久而不闻其香，即与之化矣；与不善人居，如入鲍鱼之肆，久而不闻其臭，亦与之化矣。丹之所藏者赤，漆之所藏者黑。是以君子必慎其所处者焉。”





白箱算法

■ 分类问题->回归问题

- 分类问题复杂，回归函数也变得复杂
- 回归函数不能直接输出，不可见、不可理解，学习方法黑箱化
- 固有缺陷：解释性和泛化性能不可兼得。

■ 需要设计白箱算法

- 在满足泛化性能的同时能够满足解释性的要求
- 解释能力优先
- 解释能力强的分类算法一般没有把分类方法视为回归问题



白箱算法的类表示

■ 归类输入

- 归类输入 $(X, U, \underline{X}, \text{Sim}_X)$
- 外部表示 (X, U)
- 内部表示 $(\underline{X}, \text{Sim}_X)$

■ 归类输出

- 归类输出 $(Y, V, \underline{Y}, \text{Sim}_Y)$
- 外部表示 (Y, V)
- 内部表示 $(\underline{Y}, \text{Sim}_Y)$

■ 问题的简化表示

- 如果 $X = Y$ ，则分类问题简化为 $(X, U, \underline{Y}, \text{Sim}_Y)$
- (X, U) 为训练输入， $(\underline{Y}, \text{Sim}_Y)$ 为待学习的分类器
- 如果分类算法也能输出 $(\underline{Y}, \text{Sim}_Y)$ ，这个算法被认为是解释能力强的**白箱算法**。



K近邻算法的引出

■如果 (Y, Sim_Y) 不学习就能得到，分类算法最简单，解释能力也最强

● 归类输入的外部表示：对象特性输入表示 $X = \{x_1, x_2, \dots, x_N\}$

类外延表示 $\{X_1, X_2, \dots, X_c\}$,

X_i 代表 X 中属于第 i 个输入类的对象子集

● 类的外延表示是已知的，如果类的认知表示就是外延表示，这样不学习就可以得到类的认知表示，就可以得到K近邻算法。



K近邻算法的基本思想

■ K近邻算法的基本思想：类的认知表示是其外延表示

$$\forall i, \underline{Y_i} = X_i$$

- 由于每个类由其各自的外延表示定义，因此判断一个对象属于哪一个类的简单方法，就是看看其近邻的样本类别，即所谓的“不知其人视其友”。
- 需要定义相似度映射，不同的类相似映射，构成了不同的K近邻方法。
- K近邻方法参数不多，类认知表示不需要计算，是最简单的不基于回归的分类算法。



目录

■ 10.1 K近邻算法

- 10.1.1 K近邻算法问题表示
- 10.1.2 K近邻分类算法
- 10.1.3 K近邻分类算法的理论错误率

■ 10.2 距离加权最近邻算法

■ 10.3 K近邻算法加速策略

■ 10.4 kd树

■ 10.5 K近邻算法中的参数问题



10.1 K近邻算法

■ 10.1.1 K近邻算法问题表示

- 归类输入 $(X, U, \underline{X}, \text{Sim}_X)$, 归类输出 $(Y, V, \underline{Y}, \text{Sim}_Y)$
- $X = Y$, 分类问题简化为 $(X, U, \underline{Y}, \text{Sim}_Y)$ 。
- 训练集 (X, U) , $(\underline{Y}, \text{Sim}_Y)$ 为待学习的分类器
- 分类算法能输出 $(\underline{Y}, \text{Sim}_Y)$, 则算法为白箱算法
- 每个类的类认知表示是其外延表示, 即属于该类的所有样本的集合, 则该算法为K近邻算法。

$$\forall i, \underline{Y}_i = X_i, \quad X_i = \{x_k | u_{ik} = 1\}$$

- 需要定义近邻



近邻定义

- 近邻的定义严重依赖对象的**特征描述**。不同的特征描述，近邻定义不同，K近邻算法也不同。
- 假定输入对象可以用 R^p 空间中的点来描述，每个对象表示为 p 维特征向量。**类相似性函数的定义为：**

$$\text{Sim}_Y(y, \underline{Y_i}) = \text{Sim}_Y(x, \underline{Y_i}) = \frac{|N_i(x)|}{K} \quad (10.1)$$

- **分母：表示待考虑的所有近邻整体数目**
- **分子：表示属于第 i 类的近邻个数**

$N_i(x) = \{x_1 | x_1 \in X_i \wedge x_1 \in N^K(x)\}$, $N^K(x)$ 是**所有近邻的集合**



样本归属

- 样本可分性公理：每个对象总有一个与其最相似的类
- 归类等价公理：测试对象所属的类是与其最相似的类

如果 $\arg \max_j \text{Sim}_Y(x, \underline{Y_j}) = \arg \max_j \frac{|N_j(x)|}{K} = i$

则样本 x 被认为属于第 i 类。

- 类表示唯一公理通常不成立，K近邻分类存在误差。



10.1.2 K近邻分类算法

- 训练集: $X = \{x_1, x_2, \dots, x_N\}$, $U = [u_{ik}]_{c \times N}$, U 是硬划分

- 算法10.1 K近邻算法

输入: 训练数据集以及测试对象 x_T

输出: 测试对象 x_T 所属类别 $j \in \{1, 2, \dots, c\}$

步骤:

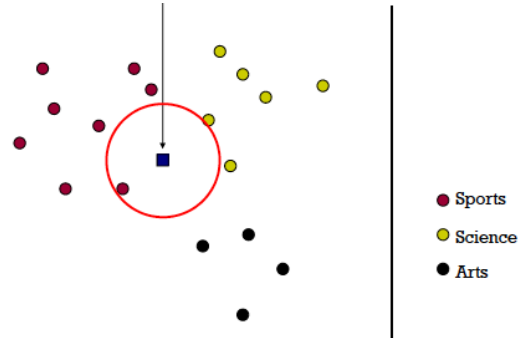
- (1) 根据给定的**距离度量方式**, 在训练集 X 中, 找到与测试对象 x_T 最邻近的**K个对象**的集合 $N^K(x_T)$, 对K个近邻统计属于每个类别的情况 $N_j(x_T)$ 。
- (2) 根据 $\{N_j(x_T)\}_{j=1}^c$ 在训练集中的类别信息来决定测试对象 x_T 的类别 i , 具体条件为

$$i = \arg \max_j \text{Sim}_Y(x_T, \underline{Y_j}) = \arg \max_j \frac{|N_j(x_T)|}{K}$$

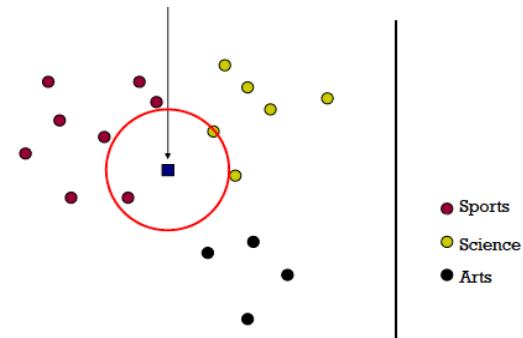


测试对象类别归属

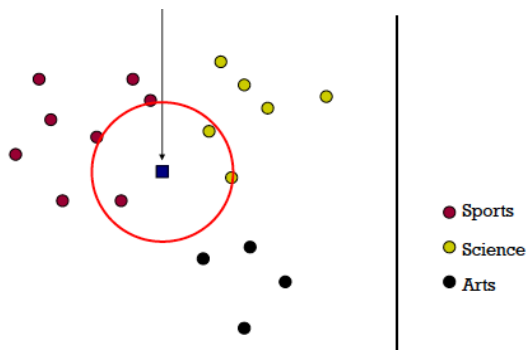
- $K=1$, 最近邻算法: 把与测试对象 x_T 最近的训练对象的类别赋给 x_T
- $K>1$, “多数表决”: K 个最接近的对象中出现频率最高的类别赋给 x_T



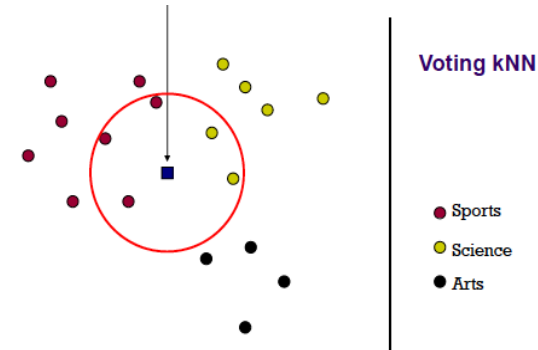
1-NN



2-NN



3-NN

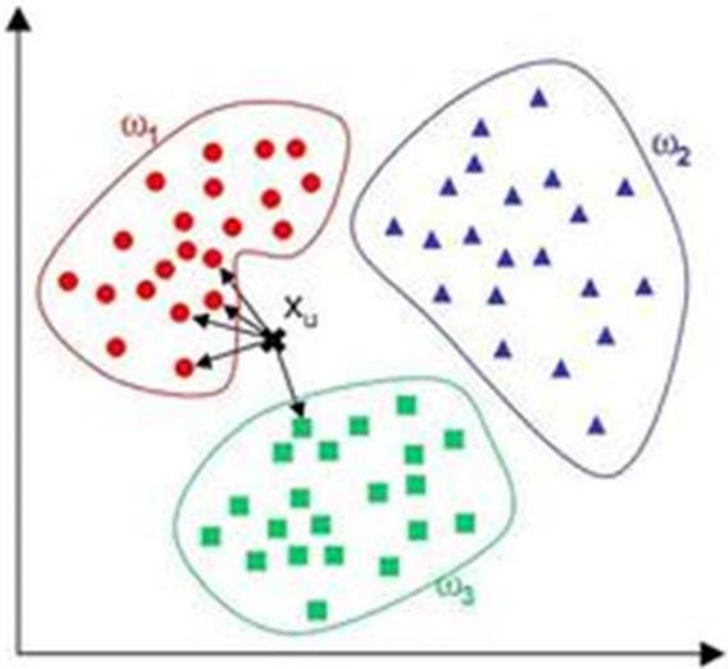


K-NN

Voting kNN



举例



K近邻法克服了最近邻法的偶然性，增加了可靠性。

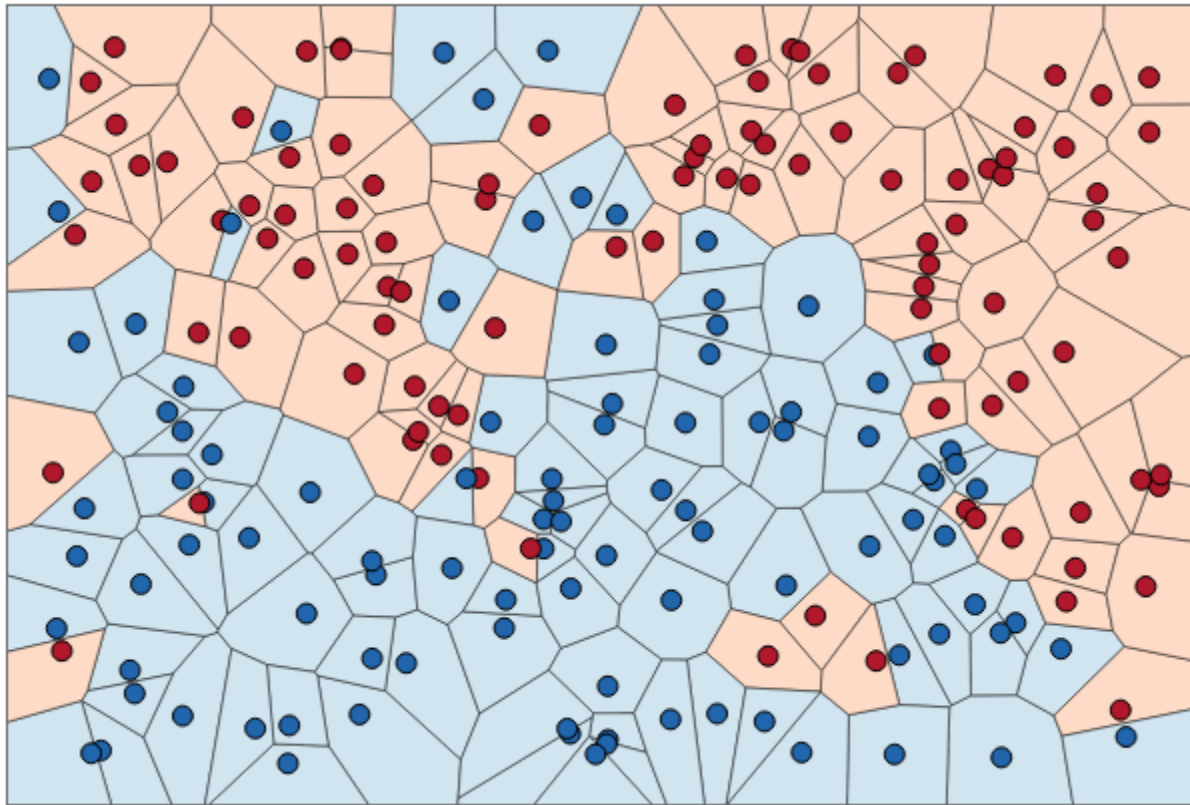
图中设定 $K=5$ ，用欧氏距离度量 \mathbf{x} 到这三类的距离得到：

$K_1 = 4, K_2 = 0, K_3 = 1$ 。

根据判别规则 \mathbf{x} 为 ω_1 类。



K近邻算法决策面



整个算法实际上是利用训练集把特征空间划分成一个个子空间，训练集中的每个样本占据一部分空间。对最近邻而言，当测试样本落在某个训练样本的领域内，就把测试样本标记为这一类。



10.1.3 K近邻算法的理论错误率

- 1967年Cover和Hart对最近邻的错误率给予了理论上证明，分类性能不差。
- 设 N 个样本下最近邻的平均错误率为 $P_N(e)$ ，样本 x 的最近邻为 $x' \in \{x_1, x_2, \dots, x_N\}$ ，平均错误率为

$$P_N(e) = \iint P_N(e|x, x')p(x'|x)dx'p(x)dx \quad (10.2)$$

$$P_N(e|x, x') = 1 - \sum_{i=1}^c P(i|x)P(i|x') \quad (10.3)$$

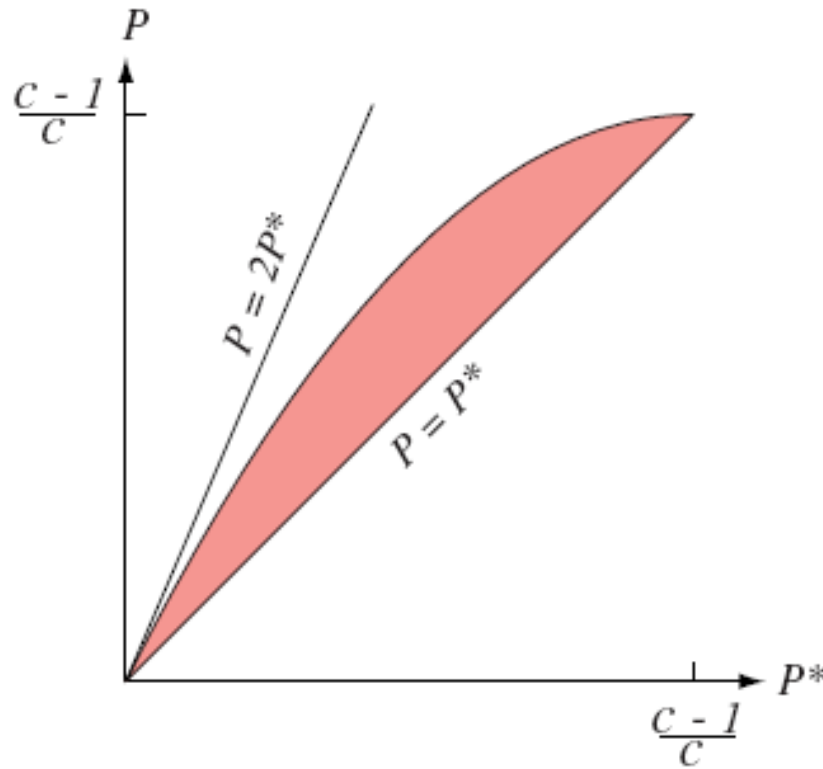
- 当 $N \rightarrow \infty$ 时， $P_N(e)$ 的极限 $P = \lim_{N \rightarrow \infty} P_N(e)$ ，可以证明存在

$$P^* \leq P \leq P^* \left(2 - \frac{c}{c-1} P^* \right) \quad (10.4)$$

其中， P^* 为贝叶斯错误率——理论最优的分类错误率， c 为类别个数， P 为最近邻算法的渐近错误率



最近邻算法渐近错误率的上下界与贝叶斯错误率的关系

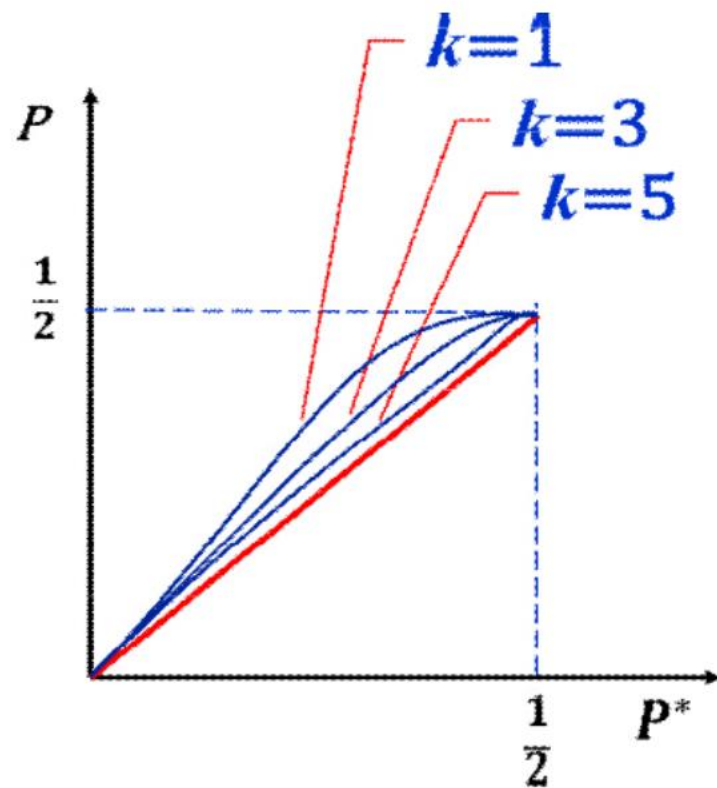


最近邻算法的渐近错误率 P 最坏不会超出两倍的贝叶斯错误率 P^* ，最好有可能接近或达到贝叶斯错误率 P^* ，该结论的条件是**样本趋向于无穷**。样本数有限时会影响最近邻算法的性能。



K近邻算法渐近错误率的上下界与贝叶斯错误率的关系

- **K近邻算法的渐近错误率仍然满足前边的上下界关系；**
- K增加，趋近于下界；K趋于无穷大时，达到了贝叶斯错误率；
- K近邻理论性能不错。K不能太大，不能等N，应该选取合适的K值。
- K近邻影响因素：一是跟边界有关，保留边界样本点，中心非本质样本减少，不影响算法。二是跟距离有关。按什么规则算K近邻，严重影响结果。





10.2 距离加权K近邻算法

■ 距离度量

- 度量 $D(\cdot, \cdot)$ 本质上是一个函数，该函数给出了两个模式之间的标量距离的大小。
一个度量必须满足4个性质：
- 对于任意的向量 a 、 b 和 c ，有
 - 非负性： $D(a,b) \geq 0$
 - 自反性： $D(a,a)=0$ 当且仅当 $a=b$
 - 对称性： $D(a,b)=D(b,a)$
 - 三角不等式： $D(a,b)+D(b,c) \geq D(a,c)$



距离度量

- **欧氏距离 (Euclidean distance)** $D(x, x') = \sqrt{\sum_i (x_i - x'_i)^2}$
- 更为一般的 d 维空间的度量为**明氏 (Minkowski) 距离**

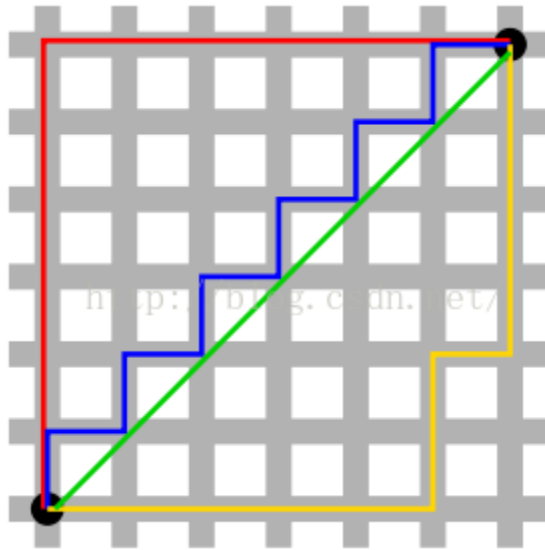
$$L_k(x, x') = \left(\sum_{i=1}^d |x_i - x'_i|^k \right)^{1/k}$$

通常也被称为 L_k 范数, 欧氏距离就是 L_2 范数

- **曼哈顿距离 (Manhattan distance)** L_1 norm
- **契比雪夫距离 (Chebyshev distance)** L_∞ norm



曼哈顿距离和契比雪夫距离



- 绿色：欧氏距离或直线距离；
- 红线：曼哈顿距离；
- 蓝色和黄色：等价的曼哈顿距离。
- 曼哈顿距离表明两个点在标准坐标系上的绝对轴距总和。

常用距离度量

曼哈顿距离：

$$d_{12} = \sum_{k=1}^n |x_{1k} - x_{2k}|$$

切比雪夫距离：

$$d_{12} = \max_i (|x_{1i} - x_{2i}|)$$

- 切比雪夫距离：两个点之间的距离定义为其各坐标数值差绝对值的最大值



马氏距离

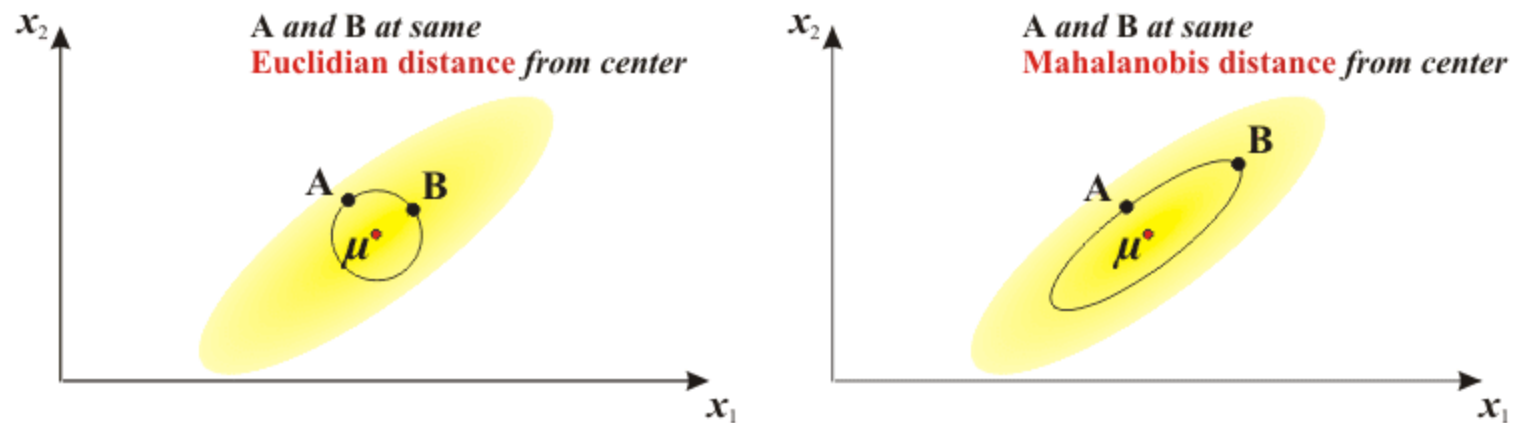
- 马氏距离 (Mahalanobis distances)

$$D(x, x') = \sqrt{(x - x')^T \Sigma^{-1} (x - x')}$$

- Σ^{-1} 协方差矩阵的逆
- 两点之间的马氏距离与原始数据的**测量单位无关**
- 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
- 可以排除变量之间的相关性的干扰



马氏距离



- 上图左为欧氏距离，A和B到 μ 是一样的距离；
- 上图右为马氏距离，A和B到 μ 也是一样的距离；
- 这说明马氏距离的计算中考虑了在不同方向上尺度单位是不同的。



距离加权K近邻算法

经典K近邻算法，每个近邻对最后的决策作用都一样。若考虑距离不同的近邻对决策作用不同，可以设计**距离加权K近邻算法**，距离查询点较近的点的权值较大，距离较远的点的权值较小。

■类相似性映射为

$$Sim_Y(x, \underline{Y}_i) = \frac{|W_i(x)|}{K} \quad (10.5)$$

其中， $W_i(x) = \sum_{x_i^K \in N_i(x)} w_{x_i^K}$ ， $N_i(x) = X_i \cap N^k(x)$ 表示属于第*i*类近邻的集合， $N^k(x)$ 是所有近邻的集合， x_i^K 表示在训练集中属于第*i*类且与*x*近邻的样本， $w_{x_i^K}$ 表示 x_i^K 的权重，与该点到测试对象*x*的距离成反比。



权重定义

- $w_{x_i^K}$ 表示 x_i^K 的权重，与该点到测试点 x 的距离成反比，有两种选择

$$w_{x_i^K} = \frac{1}{d(x, x_i^K)^2} \quad (10.6)$$

$$w_{x_i^K} = \exp(-d(x, x_i^K)^2) \quad (10.7)$$

- 对 $w_{x_i^K}$ 进行归一化

$$\sigma_{x_i^K} = \frac{w_{x_i^K}}{\sum_{x_i^K \in N_i(x)} w_{x_i^K}} \quad (10.8)$$



10.3 K近邻算法加速策略

■ K近邻算法的复杂度

● 算法10.1 K近邻算法

步骤:

(1) 根据给定的距离度量方式, 在训练集 X 中, 找到与测试对象 x_T 最邻近的 K 个对象的集合 $N^K(x_T)$

(2) 根据 $\{N_j(x_T)\}_{j=1}^c$ 在训练集中的类别信息来决定测试对象 x_T 的类别 i , 其具体条件为 $i = \arg \max_j \text{Sim}_Y(x_T, Y_j) = \arg \max_j \frac{|N_j(x_T)|}{K}$

● 步骤(1): 每个测试对象需要计算其与所有 N 个训练对象的距离, 时间复杂度为 $O(pN)$, p 为特征维数;

● 步骤(2): 从 N 个对象中选取 K 个最近邻的训练对象(需要对距离进行排序), 时间复杂度为 $O(N \log N)$;

● N 增大时, 计算量会急剧增长。

► 演示 - 数据影响



Antelope



Jellyfish



German Shepherd

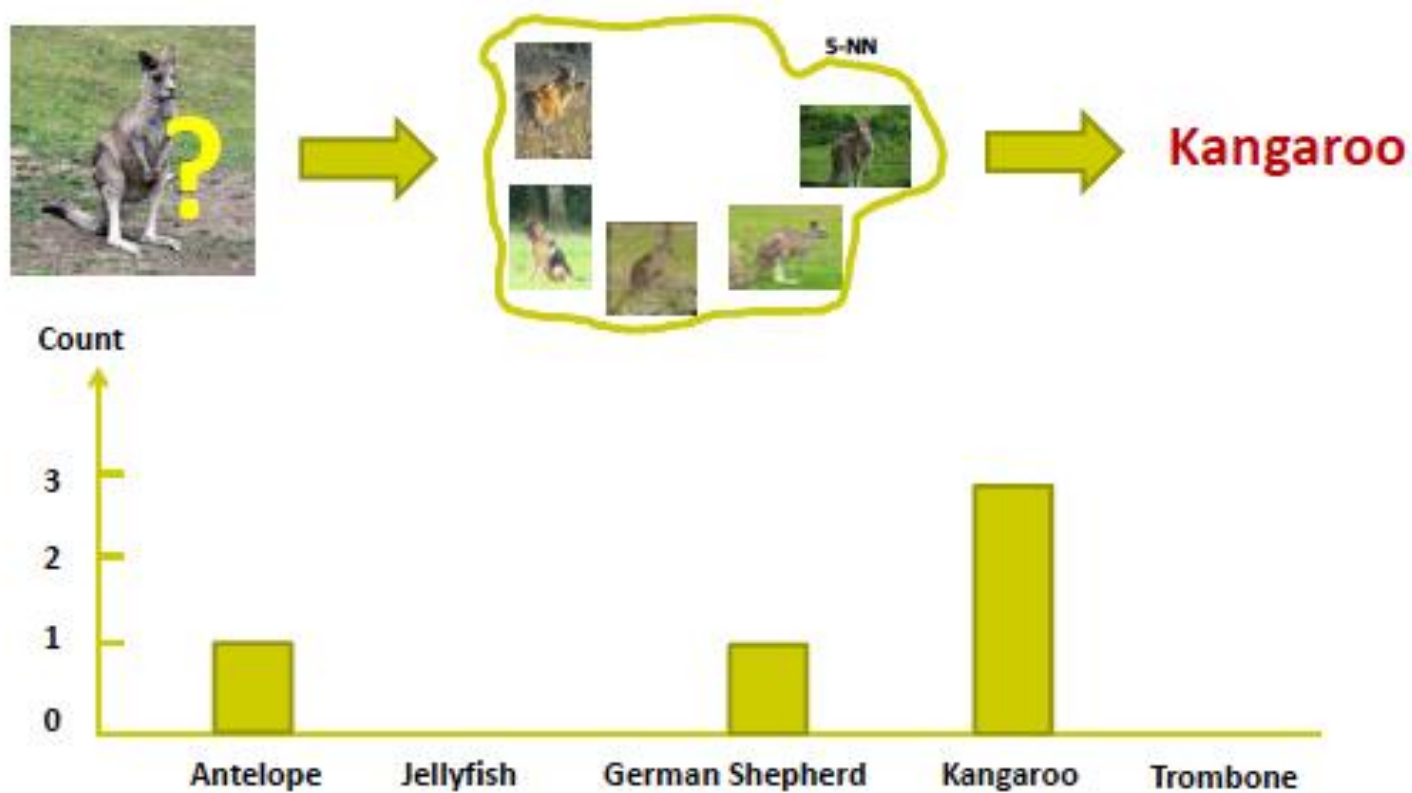


Trombone



Kangaroo

➤ 演示 – 数据影响 (voting)



➤ 演示 – 数据影响 (10K类, 4.5M检索, 4.5M训练数据)





加速策略1-计算部分距离

- 选用部分特征 ($r \ll p$) 计算观测对象之间的距离,复杂度从 $O(pN)$ 降为 $O(rN)$ (子空间学习)

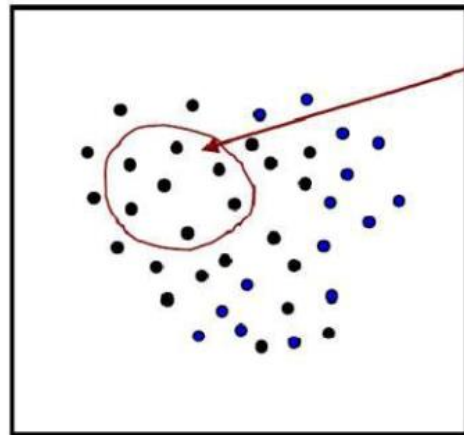
$$d_r(a, b) = (\sum_{i=1}^r (a_i - b_i)^2)^{\frac{1}{2}} \quad (10.9)$$

- 如果优先计算方差较大的维度, 可以减小 r 值的大小, 进一步降低计算量, 因为**方差较大的维度**是反映两点之间真实距离的主要因素。



加速策略2-训练对象剪辑

- 选取代表性训练对象，即消去那些对测试对象类别判定无用的训练对象
- 一个对象a被同类对象包围，此时某测试对象的最近邻为a，那么去掉a之后，该测试对象的最近邻也一定是与a同类别的训练对象，所以去掉a并不影响判定结果，称a为无用对象。



无用的样本

只适用于最近邻算法 (1-NN)
对于K ($K > 1$) 近邻算法并不适用

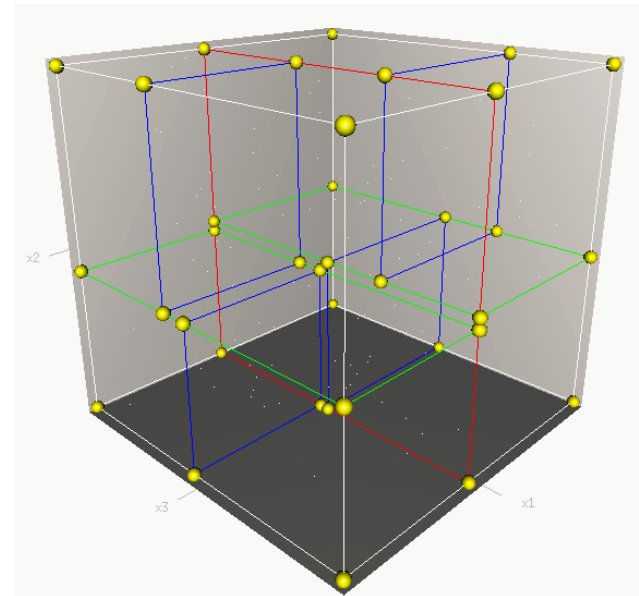


加速策略3-建立预结构

- 对训练集进行预处理，依据**相对距离**将其组织成某种形式的**搜索树**，在寻找测试对象近邻的时候可依据搜索树的结构，**只访问搜索树的部分信息**，从而降低计算量。
- kd树是一种最常见的预建立结构
- kd树由斯坦福大学本科生Jon Louis Bentley于1975年首次提出
- kd树是每个节点都为k维点的二叉树。其中k表示存储的数据的维度，d就是dimension的意思
- 所有非叶子节点可以视作用一个超平面把空间分割成两部分

■kd树的基本思想

- 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结构
- kd树从本质上来说是二叉树，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切分k维空间**，构成一系列的k维超矩形区域，kd树的每一个结点都对应于一个超矩形区域，非叶结点的左右子树分别表示划分得到的两个区域。





kd树的构建方法

- kd树算法可以分为两大部分，一部分是有关k-d树本身这种数据结构**建立的算法**，另一部分是在建立的k-d树上如何进行最邻近**查找的算法**
- kd树每个结点的内容如下：

域名	类型	描述
dom_elt	k维的向量	k维空间中的一个样本点
split	整数	分裂维的序号，也是垂直于分割超平面的方向轴序号
left	kd-tree	由位于该结点分割超平面左子空间内所有数据点构成的kd-tree
right	kd-tree	由位于该结点分割超平面右子空间内所有数据点构成的kd-tree



kd树举例

- 假设有6个二维数据点{ (2,3) , (5,4) , (9,6) , (4,7) , (8,1) , (7,2) }, 数据点位于二维空间内 (如图1中黑点所示) 。
- kd树算法就是要确定图1中这些分割空间的分割线 (多维空间即为分割平面, 一般为超平面) 。
- 下面通过一步步展示k-d树是如何确定这些分割线的。

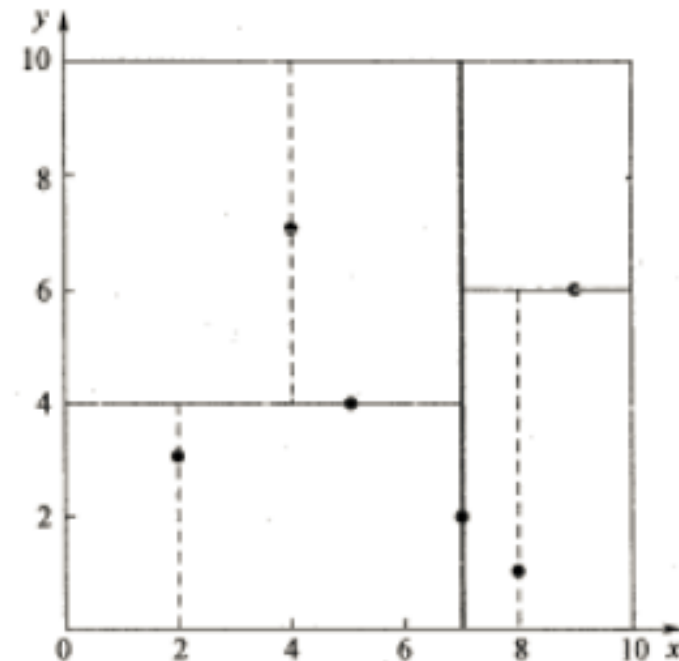


图1 二维数据k-d树空间划分示意图



kd树构建举例

- 可以简单地给 x , y 两个方向轴编号为0,1, 也即 $\text{split}=\{0,1\}$

(1) 确定 split 域首先该取的值。分别计算 x , y 方向上数据的方差, 得知 x 方向上的方差最大, 所以 split 域值首先取0, 也就是 x 轴方向

(2) 确定Node-data的域值。根据 x 轴方向的值2,5,9,4,8,7排序选出中值为7, 所以 $x_l^{(j)} = 7, j=1$ 。这样, 该节点的分割超平面就是通过(7,2)并垂直于 $\text{split} = 0$ (x 轴)的直线 $x = 7$

(3) 确定左子空间和右子空间。分割超平面 $x = 7$ 将整个空间分为两部分, 如图2所示。 $x \leq 7$ 的部分为左子空间, 包含3个节点{(2,3), (5,4), (4,7)}; 另一部分为右子空间, 包含2个节点{(9,6), (8,1)}。

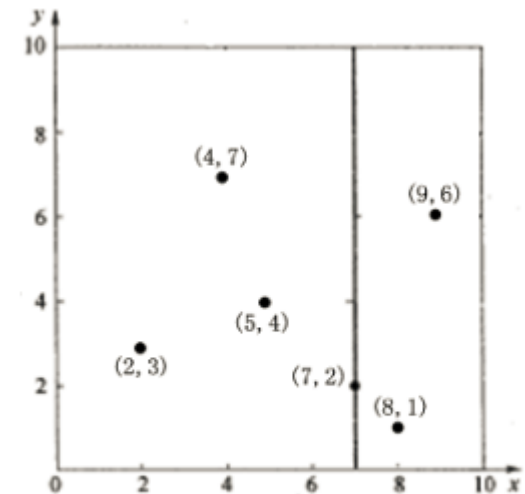


图2 $x=7$ 将整个空间分为两部分



kd树构建举例

- kd树的构建是一个递归的过程。然后对左子空间和右子空间内的数据重复根节点的过程就可以得到下一级子节点 (5,4) 和 (9,6) (也就是左右子空间的'根'节点), 同时将空间和数据集进一步细分。如此反复直到空间中只包含一个数据点, 如图1所示。最后生成的kd树如图3所示。

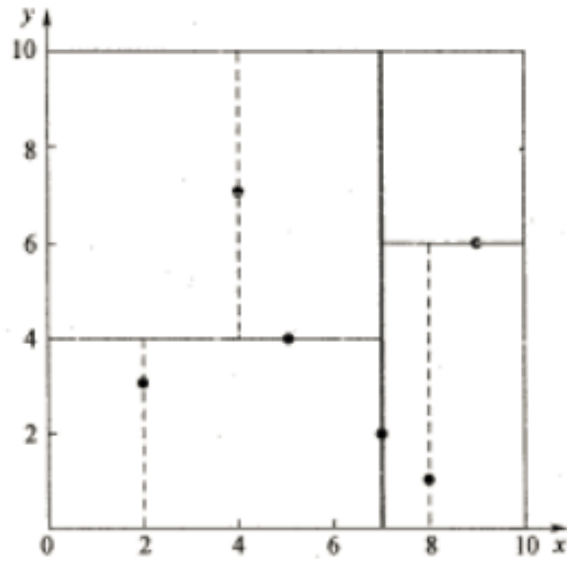


图1 二维数据k-d树空间划分示意图

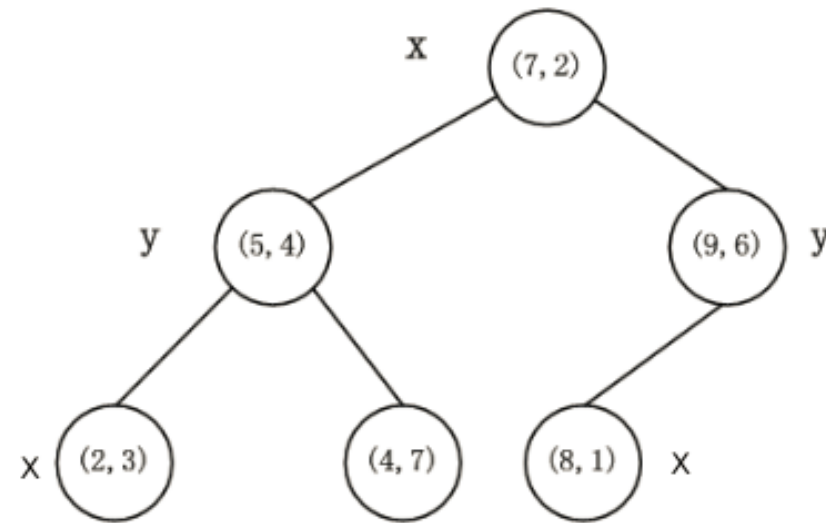


图3 上述实例生成的k-d树



kd树的构建方法

- 构造kd树的过程就是不断对第 j 维取中位数的过程，中位数对应的对象就是二叉树中的节点。
- Step1: 查看全数据集对象 $\{x_k^{(j)}\}_{k=1}^N$ ($j = 1, 2, \dots, p$)，令 $j = 1$ ，找到中位数 $x_1^{(j)}$ ，将其对应的对象 x_1 保存为根节点。左子树：所有 $x_k^{(j)} \leq x_1^{(j)}$ 的对象点组成的集合 L_{x_1} ；右子树：所有 $x_k^{(j)} > x_1^{(j)}$ 的对象点组成的集合 R_{x_1} 。
- 令深度 $t = 1$ ，在上一步形成的各子区域中操作如下：
- Step2: 令 $j = t(\bmod p) + 1$ （轮流沿着每个维度），寻找中位数 $x_{l'}^{(j)}$ ，将 $x_{l'}$ 保存为子树的根节点，由 $x_{l'}$ 分割形成新的两个子区域
- Step3: 令 $t = t + 1$ ，形成的子区域为空，算法停止，输出kd树；否则，返回Step2。



kd树查找举例-1.1

- 查找点 $s(2.1, 3.1)$, search_path 的结点为 $\langle (7, 2), (5, 4), (2, 3) \rangle$, 令 $(2, 3)$ 作为当前最佳结点 a , s_a 为0.141;
- 然后上移至 $(5, 4)$, 以 $s(2.1, 3.1)$ 为圆心, 以 $s_a=0.141$ 为半径画一个圆, 并不和超平面 $y=4$ 相交, 如下图, 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。

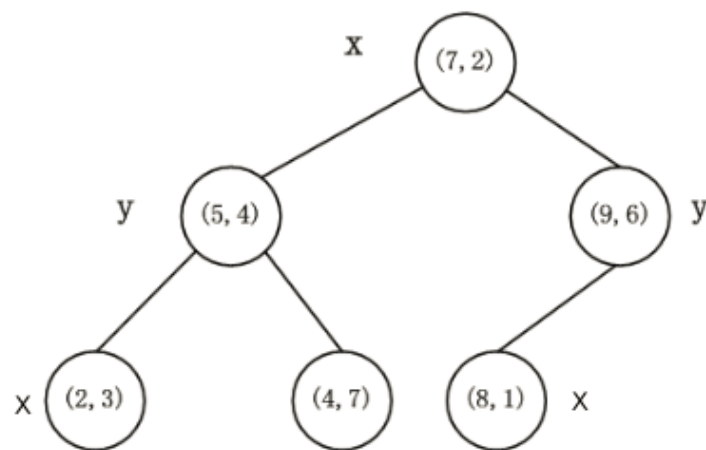


图3 上述实例生成的k-d树

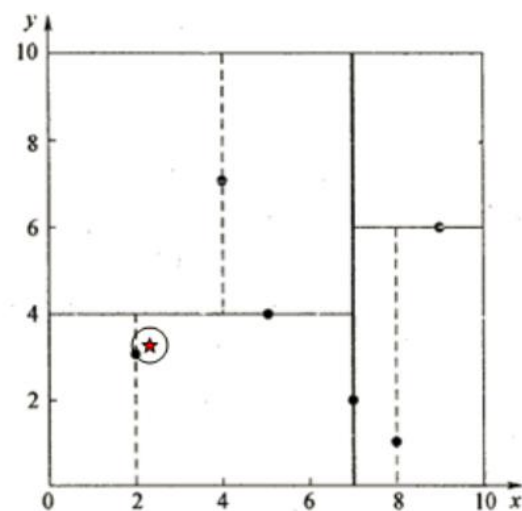


图4 查找 $(2.1, 3.1)$ 点的两次回溯判断



kd树查找举例-1.2

- 于是再上移至(7,2)，同理，以(2.1,3.1)为圆心，以 $s_a=0.141$ 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，(7,2)为根节点，结束整个搜索，返回a(2,3)作为(2.1,3.1)的最近邻点，最近距离为0.141。

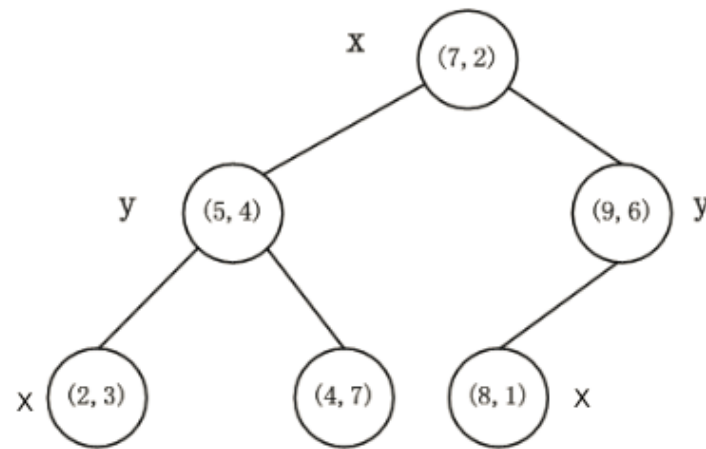


图3 上述实例生成的k-d树

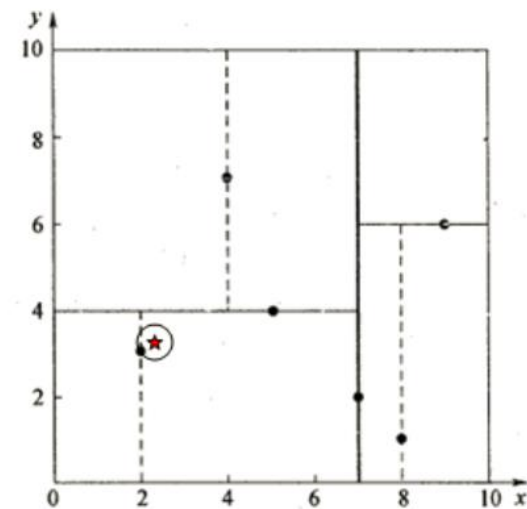


图4 查找 (2.1, 3.1) 点的两次回溯判断



kd树查找举例-2.1

- 查找点为(2,4.5)先从(7,2)查找到(5,4)节点，在进行查找时是由 $y = 4$ 为分割超平面的，由于查找点 y 值为4.5，因此进入右子空间查找到(4,7)， search_path 的结点为 $\langle (7,2), (5,4), (4,7) \rangle$ ，但(4,7)与目标查找点的距离为3.202，而(5,4)与查找点之间的距离为3.041，所以(5,4)为查询点的最佳结点 a ， s_a 为3.041；

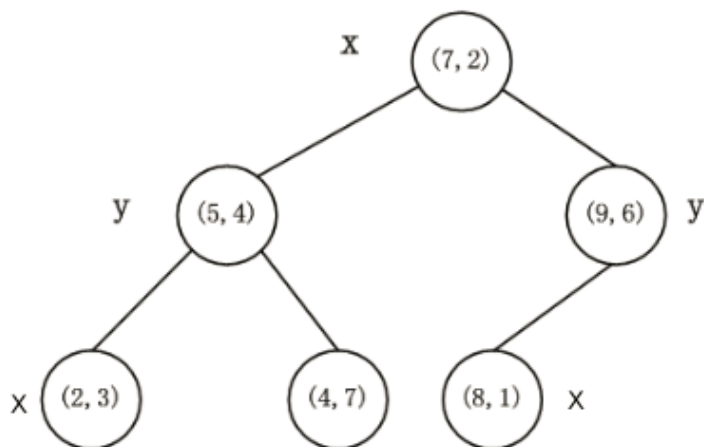
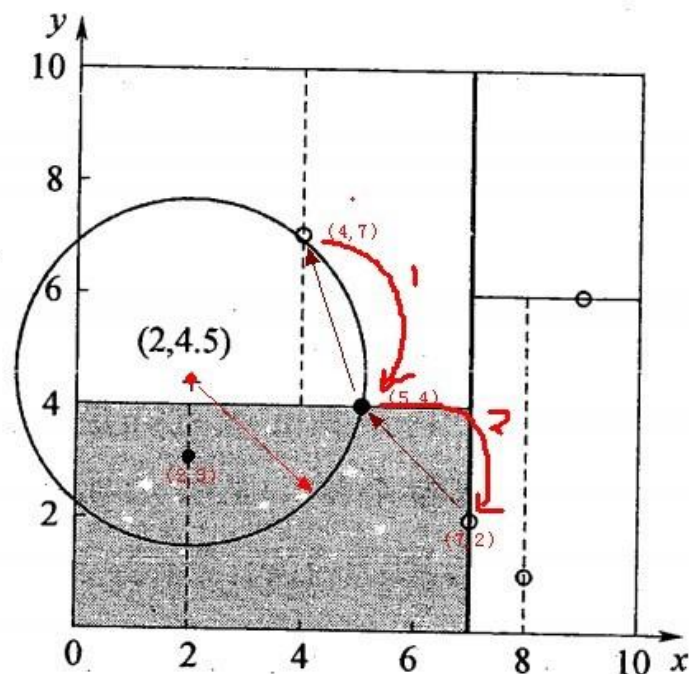


图3 上述实例生成的k-d树





kd树查找举例-2.2

- 以 $(2, 4.5)$ 为圆心，以 3.041 为半径作圆，如下图所示。可见该圆和 $y = 4$ 超平面交割，所以需要进入 $(5, 4)$ 左子空间进行查找，也就是将 $(2, 3)$ 节点加入搜索路径中得 $\langle (7, 2), (5, 4), (2, 3) \rangle$ ；于是接着搜索至 $(2, 3)$ 叶子节点， $(2, 3)$ 距离 $(2, 4.5)$ 比 $(5, 4)$ 要近，所以最近邻点更新为 $(2, 3)$ ，最近距离更新为 1.5 ；

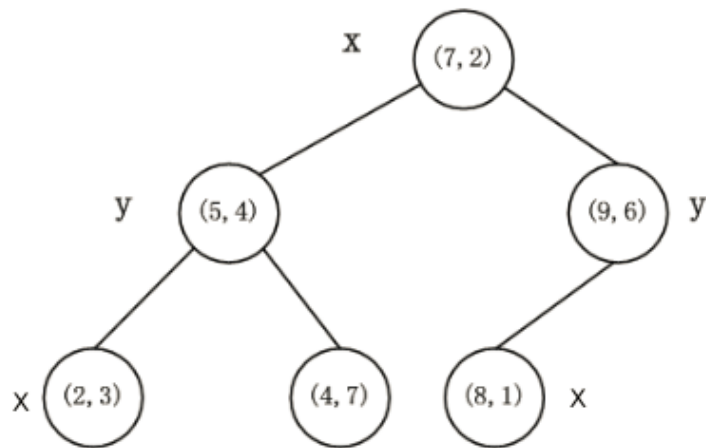
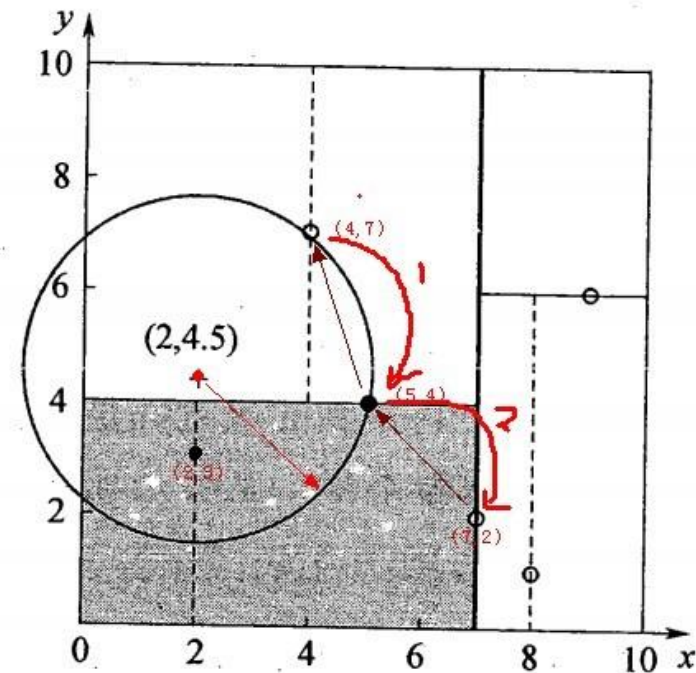


图3 上述实例生成的k-d树





kd树查找举例-2.3

- 回溯查找至(5,4)，直到最后回溯到根结点(7,2)的时候，以(2,4.5)为圆心1.5为半径作圆，并不和 $x = 7$ 分割超平面交割，如下图所示。至此，搜索路径回溯完，返回最近邻点(2,3)，最近距离1.5。

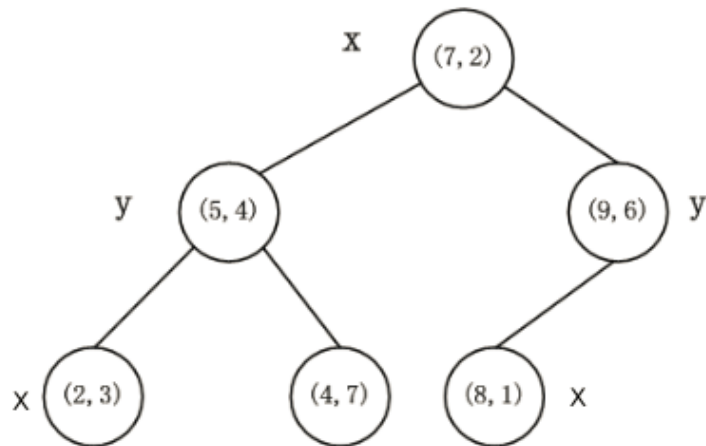
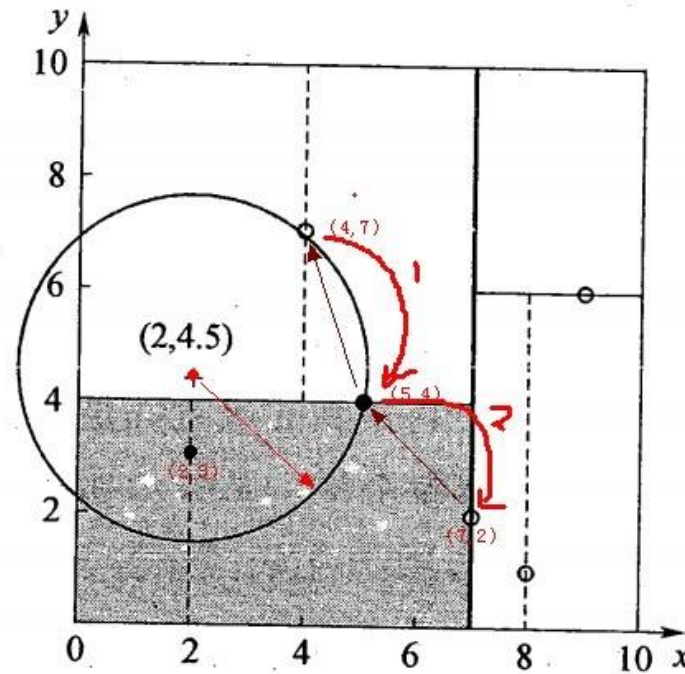


图3 上述实例生成的k-d树



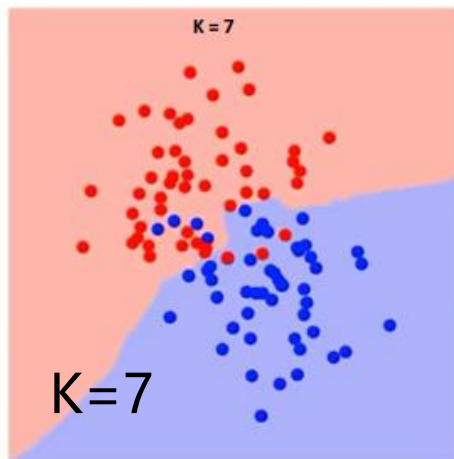
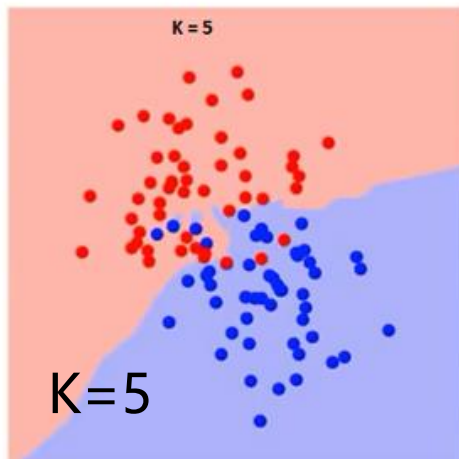
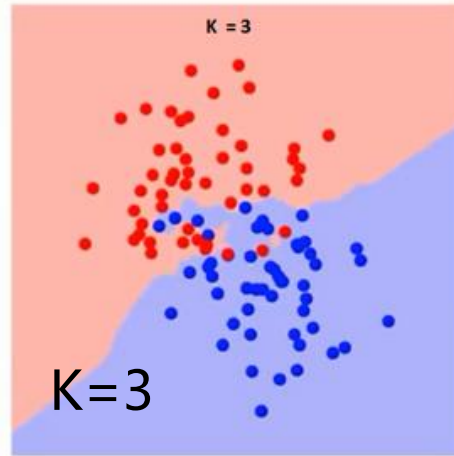
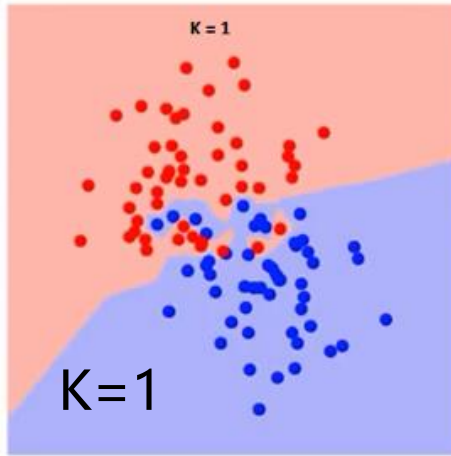


kd树的最近邻搜索

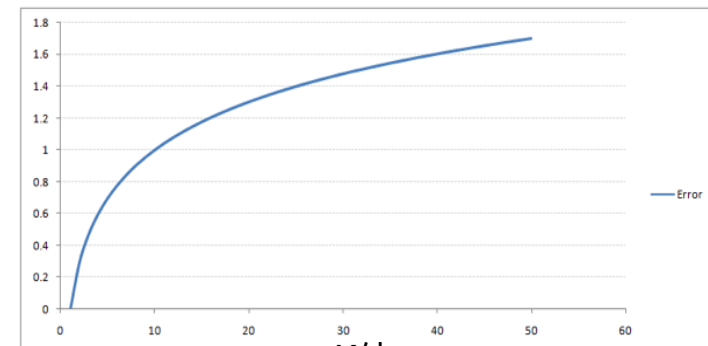
- 设查询点为 s ，寻找其所在区域。从kd树根节点开始，从第1维 $j=1$ 开始。
- Step1: 判断 $s^{(j)}$ 与当前节点 $x_l^{(j)}$ 的关系，若 $s^{(j)} \leq x_l^{(j)}$ ，则查询该节点左子树的根节点，否则查询右子树的根节点。令 $j = (j + 1) \bmod p$;
- Step2: 若该节点为叶子节点，标记为 a ，表示当前找到的最近邻；否则返回上一步
- Step3: 以查询点 s 为圆心， sa 长度为半径得到 p 维“球体”，记当前节点的父节点为 q ，判断当前节点的兄弟节点所在的区域是否与该球体相交。若相交，搜索兄弟节点所在子树，记找到的最近邻节点为 a ；上移一层，判断 q 节点是否比 a 点更接近 s ，若更近，使 $a = q$ ；若 q 为根节点，算法结束，输出 a ，否则返回Step2。



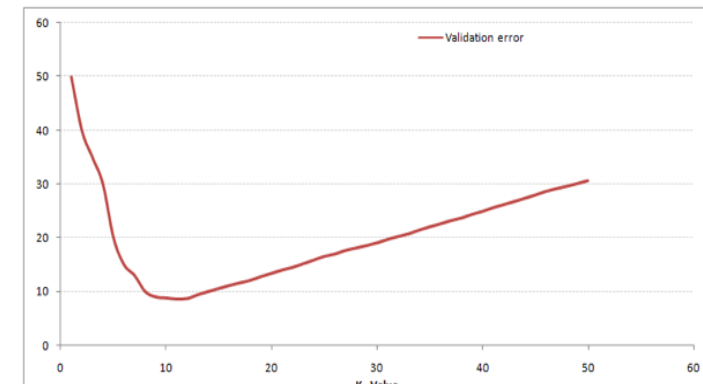
10.5 K近邻算法中的参数问题



随着K的增大,决策边界逐渐变得平滑;



训练样本错误率



测试样本错误率



10.5 K近邻算法中的参数问题

■ 参数K的选取

- K值的选取是影响K近邻算法效果的主要因素。
- 不能使用K个近邻与查询对象的距离平方和来评估：此时总是倾向于选择 $K=1$ ， $K=1$ 时算法易受噪声点和离群点的影响。
- K较小，容易被噪声影响，整体模型变得复杂，容易发生过拟合。
- K较大，近邻中出现很多其他类的点，容易发生错误，整体模型变得简单。
- 一般情况下，将K取一个较小的数值，而后使用交叉验证的方式来选取最优的K值。



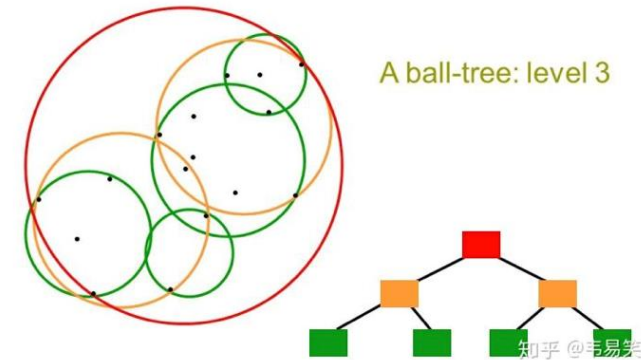
K近邻改进算法

BBF (Best-Bin-First) 算法^[1]:

- 将“查询路径”上的结点进行排序，回溯检查总是从优先级最高 (Best Bin) 的树结点开始。
- David Lowe在1997的一篇文章中针对高维数据提出的一种近似算法，此算法能确保优先检索包含最近邻点可能性较高的空间，此外，BBF机制还设置了一个运行超时限定。采用了BBF查询机制后，kd树便可以有效的扩展到高维数据集上；

超球体空间分割法 (ball-tree)

- 每个分割块都是超球体，而不是KD树里面的超矩形体；
- 当样本数量增加，不规则性上升时，即便映射到高维核空间里，也会出现线性不可分的情况，此时 SVM 的准确度就会下降，而装配了 ball-tree 的核化 kNN 表现出较高的准确性，同时兼具良好的查询性能。





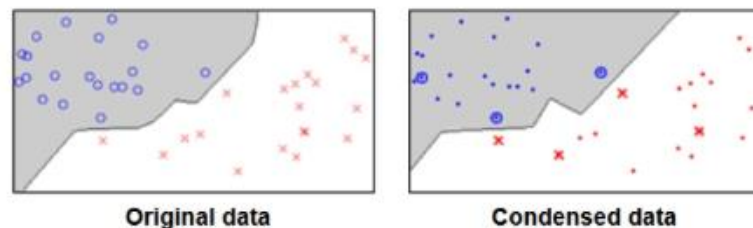
K近邻改进算法

Kernel based kNN^[4]:

- 距离计算时，将线性不可分的低维度特征矢量通过核函数，例如高斯核（RBF）、多项式核（POLY）等，映射到线性可分的高维特征空间中；
- Kai Yu等^[4]论证了基于核方法的 kNN 分类器比传统 kNN 分类器表现的更好，因为仅仅是距离测量方式改变了一下，所以总体时间和传统 kNN 分类器仍然类似，但是效果好了很多；

Condensed Nearest Neighbor^[5]:

- 先将样本点删除，然后用其他样本判断这个点，如果判断结果正确，则认为是一个冗余点，可以删除，如果不正确就要保留；
- 经过 reduction 过后的样本数据和原来的不一样，求解结果是一个近似解，只要误差可控，可以极大的提高 kNN 的搜索性能；

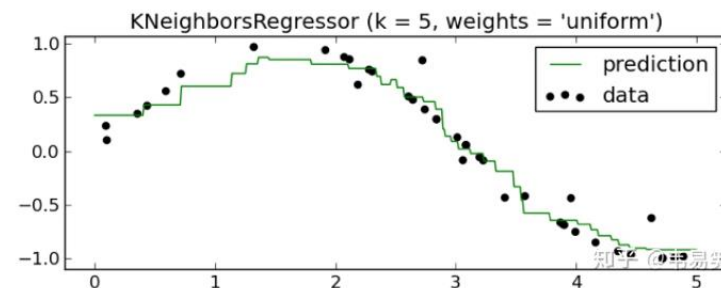




拓展应用

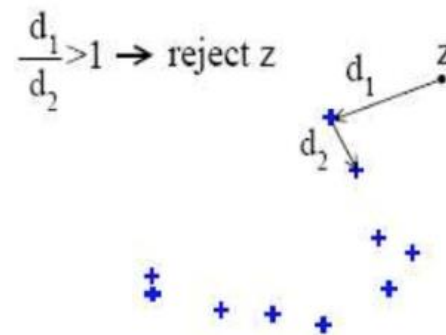
■ 回归

- 给定一个测试点坐标 x ，求回归曲线上对应的 y 值：最简单的做法是取 k 个离 x 最近的样本坐标，然后对他们的 y 值求平均；



■ 异常点/离群点检测

- 对待测试样本 z ，在训练样本中找到一个离他最近的邻居 b ，计算 z 到 b 点的距离为 d_1 ，然后再在训练样本中找到一个离 b 最近的点 c ，计算 b 到 c 距离为 d_2 ；



如果 $d_1 \leq \alpha * d_2$ 那么接受 z 样本，否则拒绝它（识别为异常点）。



拓展应用

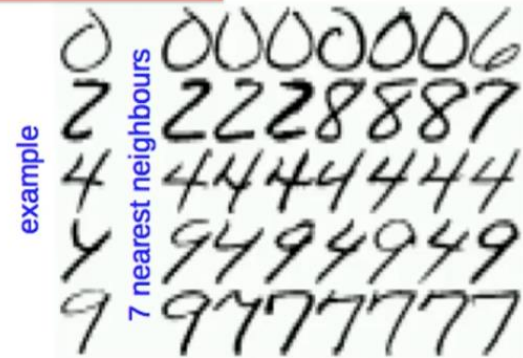
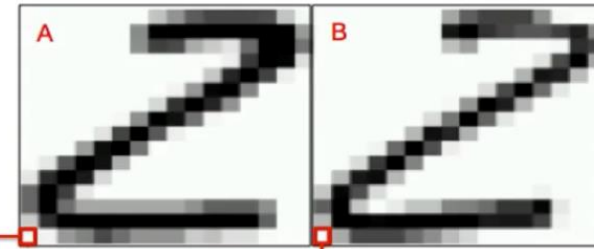
■ 手写数字识别

- 16x16 bitmaps
- 8-bit grayscale
- Euclidian distance

– over raw pixels

$$D(A,B) = \sqrt{\sum_r \sum_c (A_{r,c} - B_{r,c})^2}$$

- Accuracy:
 - 7-NN ~ 95.2%
 - SVM ~ 95.8%
 - humans ~ 97.5%



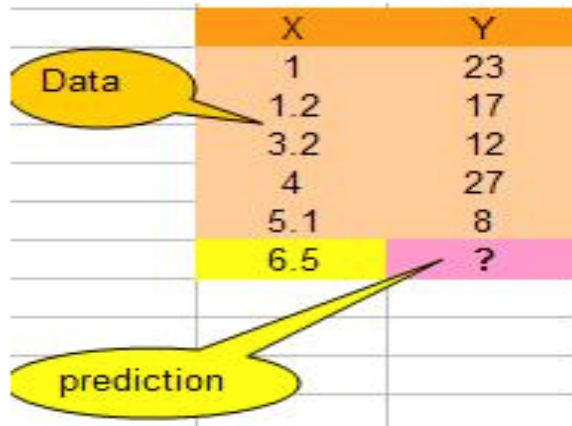
■ 图像分类

<https://cs231n.github.io/classification/#nn>



作业

1. We have 5 data pair (X,Y) as shown below. The data are quantitative in nature. Suppose the data is sorted as in time series. Then the problem is to estimate the value of Y based on K-Nearest Neighbor (KNN) algorithm at X=6.5



	X	Y
	1	23
Data	1.2	17
	3.2	12
	4	27
	5.1	8
	6.5	?
prediction		

Suppose K=2

2. Suppose we know the X data is between 0 and 6 and we would like to compute the value of Y between them.
 1. We define $dx=0.1$ and set the value of $x = 0$ to 6 with increment dx
 2. Compute distance between x (as if it is the query instance) and each of the data X



小结

■ 优点

- ①实现简单，解释性好，非参数方法。属于懒惰学习（lazy learning），不需要学习类认知表示，待收到样本后再进行分类，训练时间为0；
- ②分类结果较好，错误率在一倍和两倍Bayes错误率 P^* 之间。

■ 缺点

- ①需将全部训练样本存入计算机中，每次决策都要计算 x 与全部样本的距离并进行比较。因此存贮量和计算量都很大。
- ②在高维空间，数据集变得稀疏，导致不存在紧密的近邻，很难达到理想效果，即“维数灾难”。
- ③分析是建立在样本无限大的基础上，对有限样本缺乏分析。



小结

Computer Vision: let's say you have an image and you want to find sets of images similar to that image, KNN is used — among many others — to find and predict the most prominent images that resemble the input image.

Marketing: imagine you have purchased a baby trolley on Amazon, next day you will receive an email from the e-commerce company that you might be interested on a set of pacifiers. Amazon can target you with similar products, or products that other customer that have the same buying habits as you.

Content recommendation: probably the most interesting example is Spotify and their legendary recommendation engine. KNN is used on many content recommendation engines, even though more powerful systems are now available, KNN is still relevant to this date.



参考文献

- [1] Kernel Nearest-Neighbor Algorithm
- [2] T. Liu, A. W. Moore, A. Gray, “New Algorithms for Efficient High Dimensional Non-Parametric Classification”, Journal of Machine Learning Research, 2006, pp 1135-1158.
- [3] T. Bailey and A. K. Jain, “A note on Distance weighted k-nearest neighbor rules”, IEEE Trans. Systems, Man Cybernetics, Vol.8, pp 311-313, 1978.
- [4] Kernel Nearest-Neighbor Algorithm
- [5] K. Chidananda and G. Krishna, “The condensed nearest neighbor rule using the concept of mutual nearest neighbor”, IEEE Trans. Information Theory, Vol IT- 25 pp. 488-490, 1979.



小结

1、问题一：近邻间的距离会被大量的不相关属性所支配。

应用k-近邻算法的一个实践问题是，实例间的距离是根据实例的所有属性（也就是包含实例的欧氏空间的所有坐标轴）计算的。这与那些只选择全部实例属性的一个子集的方法不同，例如决策树学习系统。

比如这样一个问题：每个实例由20个属性描述，但在这些属性中仅有2个与它的分类是有关。在这种情况下，这两个相关属性的值一致的实例可能在这个20维的实例空间中相距很远。结果，依赖这20个属性的相似性度量会误导k-近邻算法的分类。近邻间的距离会被大量的不相关属性所支配。这种由于存在很多不相关属性所导致的难题，有时被称为维度灾难（curse of dimensionality）。最近邻方法对这个问题特别敏感。

2、解决方法：当计算两个实例间的距离时对每个属性加权。

这相当于按比例缩放欧氏空间中的坐标轴，缩短对应于不太相关属性的坐标轴，拉长对应于更相关的属性的坐标轴。每个坐标轴应伸展的数量可以通过交叉验证的方法自动决定。



BJTU “Machine Learning” Group

于 剑: jianyu@bjtu.edu.cn;

景丽萍: lpjing@bjtu.edu.cn;

田丽霞: lxtian@bjtu.edu.cn;

黄惠芳: hfhuang@bjtu.edu.cn;

李晓龙: hlli@bjtu.edu.cn;

吴 丹: wudan@bjtu.edu.cn;

万怀宇: hywan@bjtu.edu.cn;

王 晶: wj@bjtu.edu.cn.

