

# 机器学习入门

## K近邻法

K-Nearest Neighbours, K-NN

K近邻分类、K近邻回归

*Non – Generalizing Machine Learning Method*

河北师范大学软件学院

2017.03.15-03.22



# PART1. K近邻分类

## K-Nearest Neighbors Classification

### 关键词:

分类问题的定义

KNN分类模型

距离度量

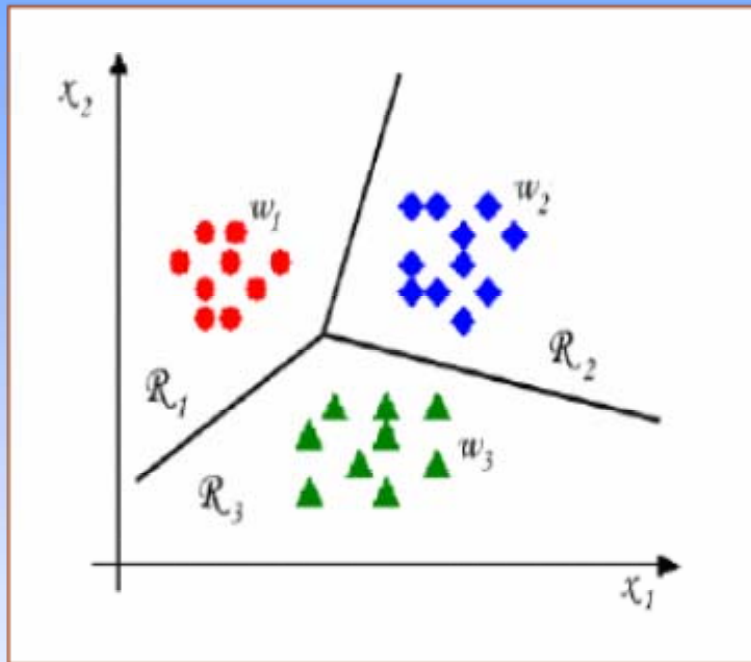
超参数

交叉验证

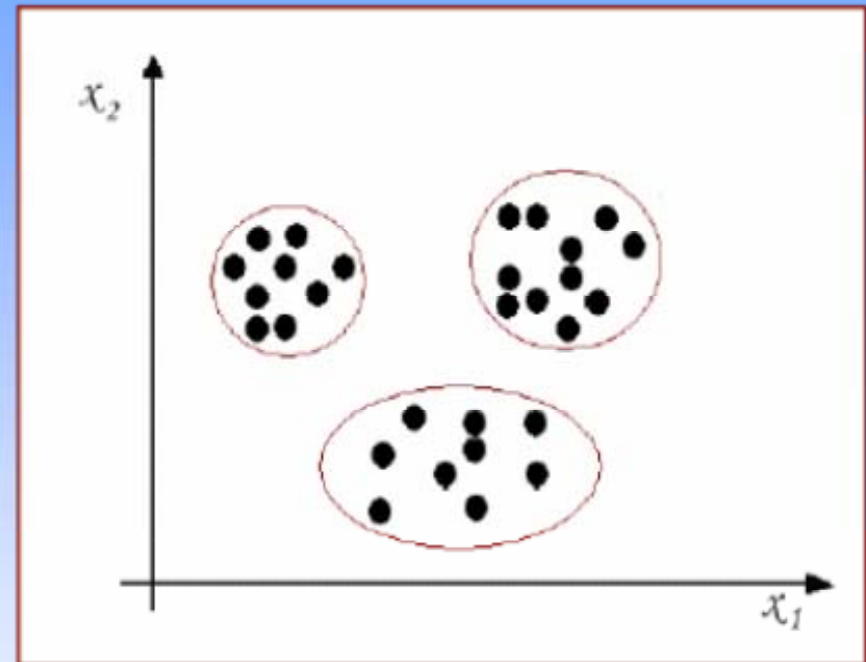
性能评价



## 分类 (Classification) 与 聚类 (Clustering) 的区别



Given labeled training patterns,  
construct decision boundaries  
or partition the feature space



Given some patterns, discover the  
underlying structure (categories)  
in the data

## 1. 分类问题的一般描述

## 2. K近邻分类算法的描述

## 3. K近邻分类的三个基本要素

距离度量

超参数K值的确定

决策规则

## 4. K近邻算法的实现—kd树(k-dimensional Tree)

Kd树的构建

待决策样本的K近邻搜索分支定界搜索

## 5. K近邻分类模型的评价

## 1. 分类问题的一般描述

给定带有类别标记的训练样本集  $\{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$ .

其中:

$\mathbf{x}_i$ ----第*i*个观测样本的特征向量,  $\mathbf{x}_i \in \mathbf{X} \subseteq R^d$

$y_i$ ----第*i*个观测样本的类别标号  $\begin{cases} C=2, & y_i \in Y = \{\omega_1, \omega_2\} \\ C>2, & y_i \in Y = \{\omega_1, \omega_2, \dots, \omega_C\} \end{cases}$

要求:

基于上述样本集, 设计分类模型--**分类模型的监督式学习**;

对特征空间的任意观测 $\mathbf{x}$ 进行类别决策--**模型的使用**

1. 分类问题的一般描述

2. K近邻分类算法的描述

3. K近邻分类的三个基本要素

距离度量方式

超参数K值的确定

决策规则

4. K近邻算法的实现—kd树(k-dimensional Tree)

5. KNN分类模型的评价

## 2.1 K近邻分类模型的引入

**近朱者赤，近墨者黑**

**懒惰学习**

无显式的训练过程，直接进行基于样本的空间划分

**非参数法**分类模型

**准备工作轻松**

训练样本集、近邻数**K**、距离度量

## 2.2 K近邻分类算法的描述

**输入 :** (1) **训练样本集**  $D = \{(x_i, y_i), i = 1, \dots, N\}$ , 并且有:

$x_i$  ----第*i*个训练样本的特征向量,  $x_i \in X \subseteq R^d$

$y_i$  ----第*i*个训练样本的类别标号  $\begin{cases} C=2, & y_i \in Y = \{\omega_1, \omega_2\} \\ C>2, & y_i \in Y = \{\omega_1, \omega_2, \dots, \omega_C\} \end{cases}$

(2) **观测样本**  $x$

**输出 :** 观测样本 $x$ 所属的类别 $y$ .

**STEP1.** 指定**距离度量**, **选择K**

**STEP2.** 在训练集 $D$ 内找到样本 $x$ 的**K个近邻**, 记为 $N_K(x)$

$$N_K(x) = N_{K, \omega_1}(x) \cup \dots \cup N_{K, \omega_C}(x)$$

**STEP3.** 结合指定的**分类规则**, 预测 $x$ 的类别 $y$ .

$$\hat{y} = \arg \max_{\omega \in \{\omega_1, \omega_2, \dots, \omega_C\}} \text{Sim}(x, \omega)$$





不同的距离度量方式、  
不同的K值、  
不同的决策规则，

会导致不同的分类结果

1. 分类问题的一般描述
2. K近邻分类算法的描述
3. K近邻分类的三个基本要素

## 3.1 距离度量

典型的距离度量方式

样本的规范化预处理

## 3.2 超参数K值的确定

## 3.3 决策规则

4. K近邻算法的实现—kd树(k-dimensional Tree)
5. K近邻分类模型的评价

### 3.1 距离度量



对于  $\forall \mathbf{x}_i, \mathbf{x}_j \in X \subseteq \mathbf{R}^d$   $\mathbf{x}_i = [x_{i1} \cdots x_{id}]^T$   $\mathbf{x}_j = [x_{j1} \cdots x_{jd}]^T$

**A.  $L_p$  距离**  $d_p(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_p = \left( \sum_{k=1}^d |x_{ik} - x_{jk}|^p \right)^{\frac{1}{p}} \quad p > 0$

**B. 绝对值距离 (manhattan distance,  $L_1$  距离)**

$$d_1(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_1 = \sum_{k=1}^d |x_{ik} - x_{jk}|$$

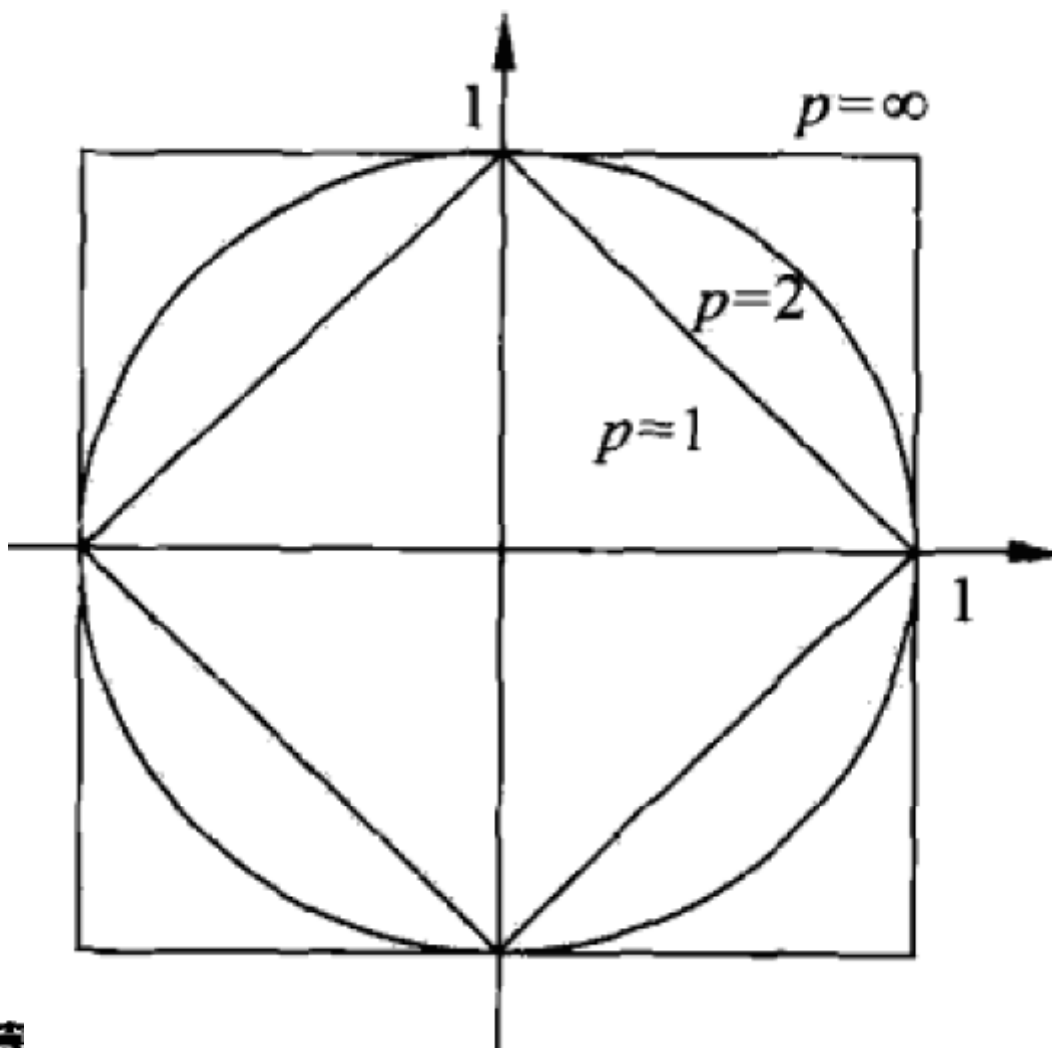
**C. 欧式距离 ( $L_2$  距离)**  $d_2(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \left( \sum_{k=1}^d |x_{ik} - x_{jk}|^2 \right)^{\frac{1}{2}}$

**D. 切氏距离 ( $L_\infty$  距离, 拉格朗日距离)**

$$d_\infty(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_\infty = \max_{1 \leq k \leq d} |x_{ik} - x_{jk}|$$

例：平面中与原点的 $L_p$ 距离为1的点的集合

- $p=1$
- $p=2$
- $p$ 为无穷大



## 用于距离度量的样本的“标准化预处理”



### “标准化预处理”的必要性：

$$\|x_i - x_j\|_p = \left( \sum_{k=1}^d |x_{ik} - x_{jk}|^p \right)^{\frac{1}{p}}$$

距离计算时，通常对各个特征变量均等对待。

若**大数量级、大动态范围的特征**以及**小数量级、小动态范围特征**同时参与距离估算，常面临如下风险：

- (1)起主导作用的前者可能淹没后者特征变化；
- (2)起主导作用的特征所含的类鉴别信息不一定明显

不同特征的量纲不同、固定量纲下采用不同度量单位，也会导致不同数量级的特征取值。有必要**去量纲化**。

因此，需要对所有样本的特征描述部分，进行标注化预处理。

## 样本的“标准化预处理”的方式：

特征的平移、尺度调整



$$D = \{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$$

### 方式1--0均值、1方差的标准化预处理(推荐使用)

首先，利用训练集估计

$$\begin{cases} \mu_k = \frac{1}{N} \sum_{i=1}^N x_{ik} \\ \sigma_k = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_{ik} - \mu_k)^2} \end{cases} \quad k = 1, \dots, d$$

然后，任意观测样本  $\mathbf{x} \in X \subseteq \mathbf{R}^d$  的预处理：

处理前  $\mathbf{x} = [x_1 \quad \dots \quad x_d]^T$

处理后  $\mathbf{x}' = [x_1' \quad \dots \quad x_d']^T$  其中  $x_k' = \frac{x_k - \mu_k}{\sigma_k} \quad k = 1, \dots, d$

## 方式2--将原始样本特征取值进行线性映射

首先，利用训练集确定

$$\begin{cases} x_{k\_min} = \min_{i=1,...,N} \{x_{ik}\} \\ x_{k\_max} = \max_{i=1,...,N} \{x_{ik}\} \end{cases} \quad k = 1, \dots, d$$

然后，任意观测样本  $\mathbf{x} \in \mathbf{X} \subseteq \mathbf{R}^d$  的预处理：

处理前  $\mathbf{x} = [x_1 \quad \dots \quad x_d]^T$

处理后  $\mathbf{x}' = [x_1' \quad \dots \quad x_d']^T$

若线性映射至  $[0,1]$  则  $x_k' = \frac{x_k - x_{k\_min}}{x_{k\_max} - x_{k\_min}} \quad k = 1, \dots, d$

若线性映射至  $[-1,1]$  则  $x_k' = \frac{2(x_k - x_{k\_min})}{x_{k\_max} - x_{k\_min}} - 1 \quad k = 1, \dots, d$

1. 分类问题的一般描述
2. K近邻分类算法的描述
3. K近邻分类的三个基本要素

## 3.1 距离度量

典型的距离度量方式、样本的规范化预处理

## 3.2 超参数K值的确定

m-fold cross validation + 分类模型的评价指标

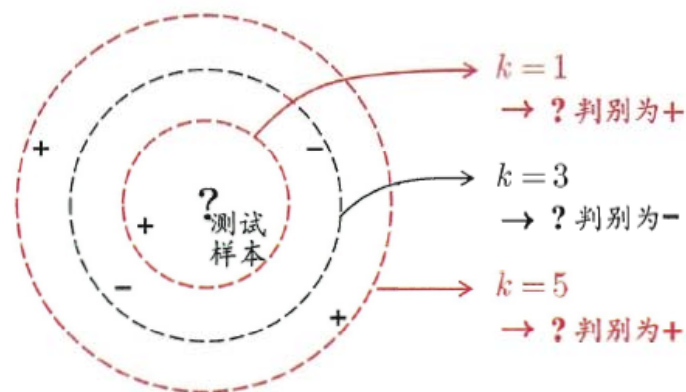
## 3.3 决策规则

4. K近邻算法的实现—kd树(k-dimensional Tree)
5. K近邻分类模型的评价



## A. 对超参数K值进行选择的意义

任意观测 $\mathbf{x}$ 的类别预测：

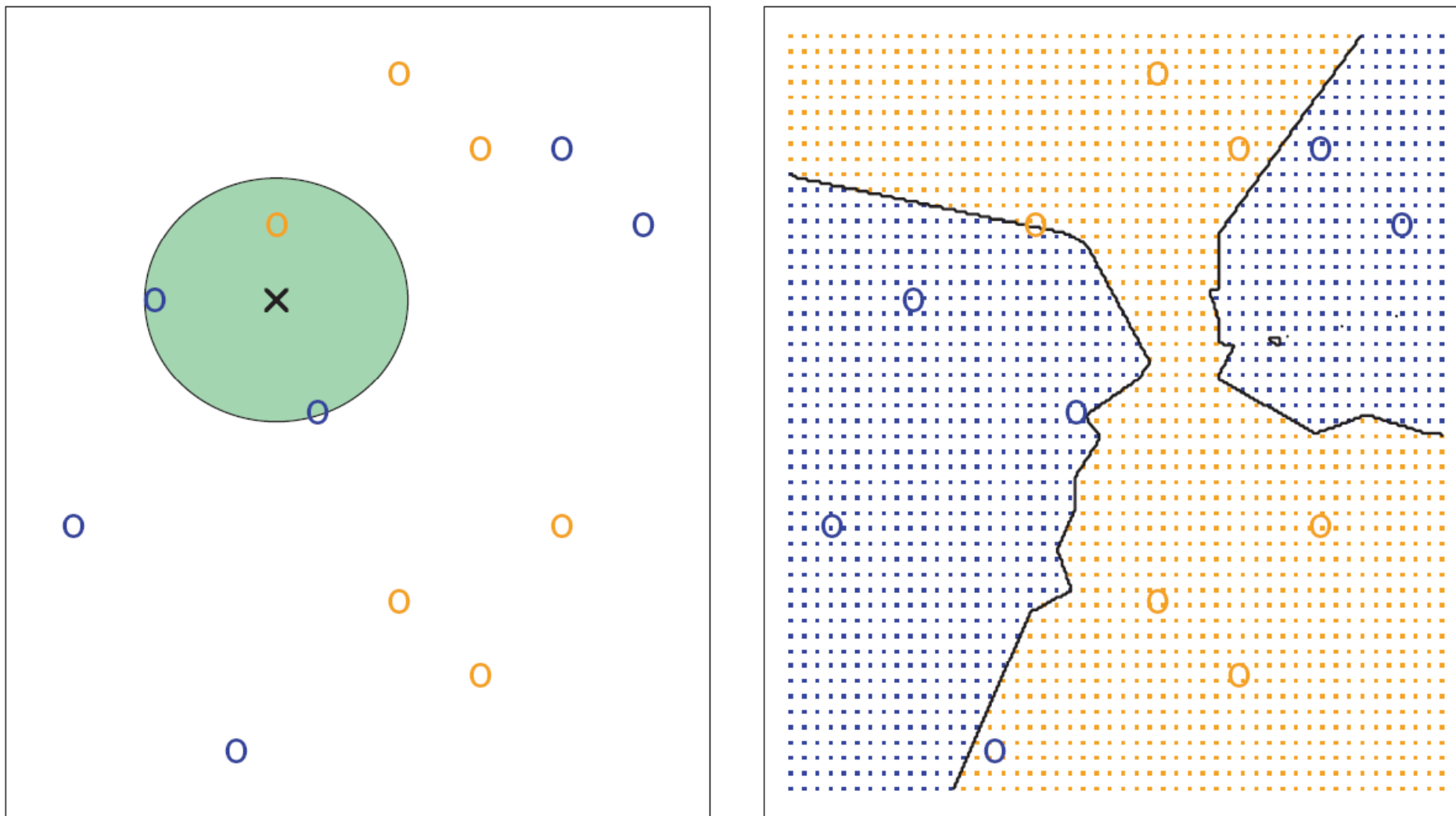


若使用**较小的K值**，则利用 $\mathbf{x}$ 较小邻域训练样本进行类别预测，只有**更接近 $\mathbf{x}$** 的训练样本**(更相似)**才对预测结果有作用，预测结果对近邻的训练样本类别更为敏感。导致训练误差降低，预测误差增大。

若数据分布复杂、或噪声影响严重，易导致高的“预测错误率”。模型复杂。**最小 $K=1$ ，为最近邻分类。**

若使用**较大的K值**，则需要利用 $\mathbf{x}$ 较大邻域的训练样本进行类别预测，使得**远离 $\mathbf{x}$** 的训练样本**(更相异)**对预测结果也有作用，这会增大训练误差，但降低预测误差。模型更简单。

**最大 $K=N$ ，每个位置的预测结果为具有最大训练样本数目的类别。**



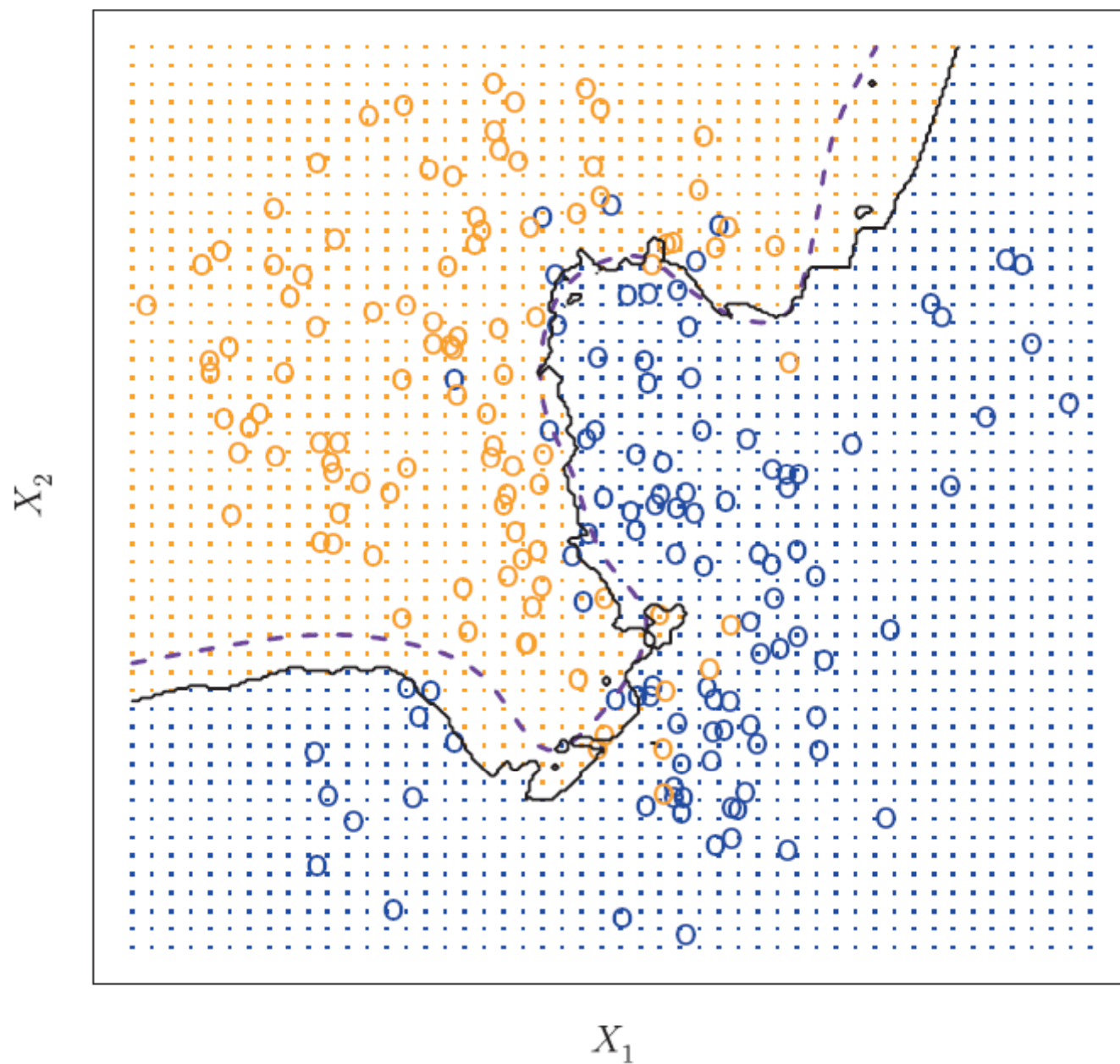
图：K-最近邻规则、两类问题，基于训练样本的特征空间 网格划分。

$K=3$ .

左图，训练样本； 右图，二维特征空间的决策结果

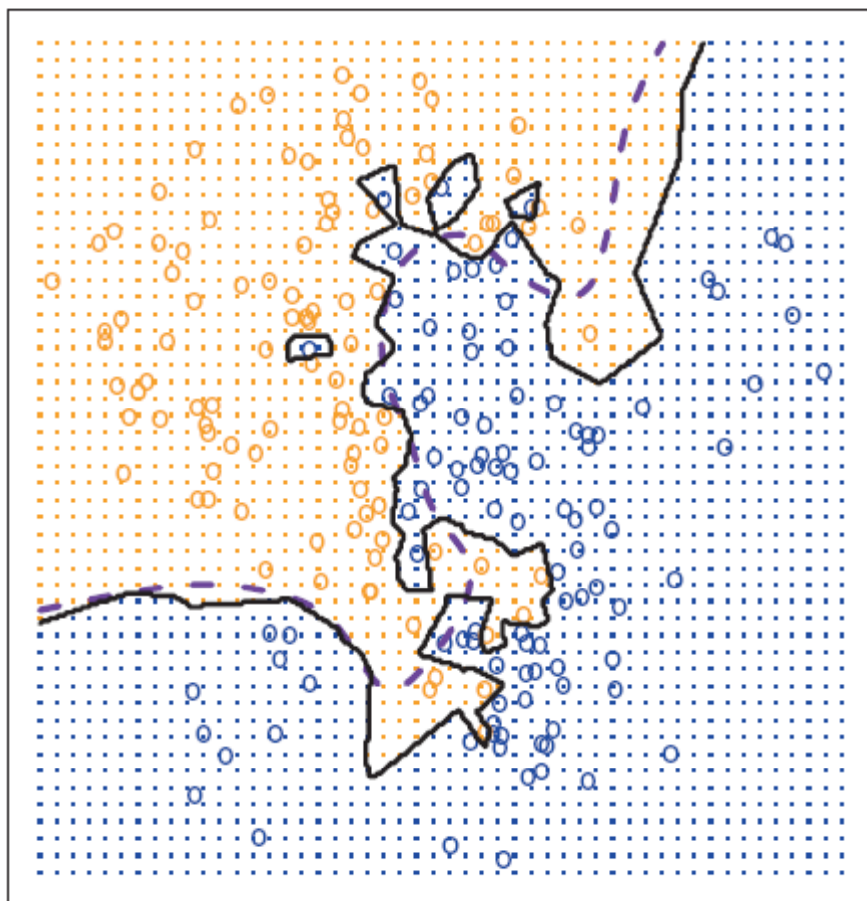


KNN: K=10

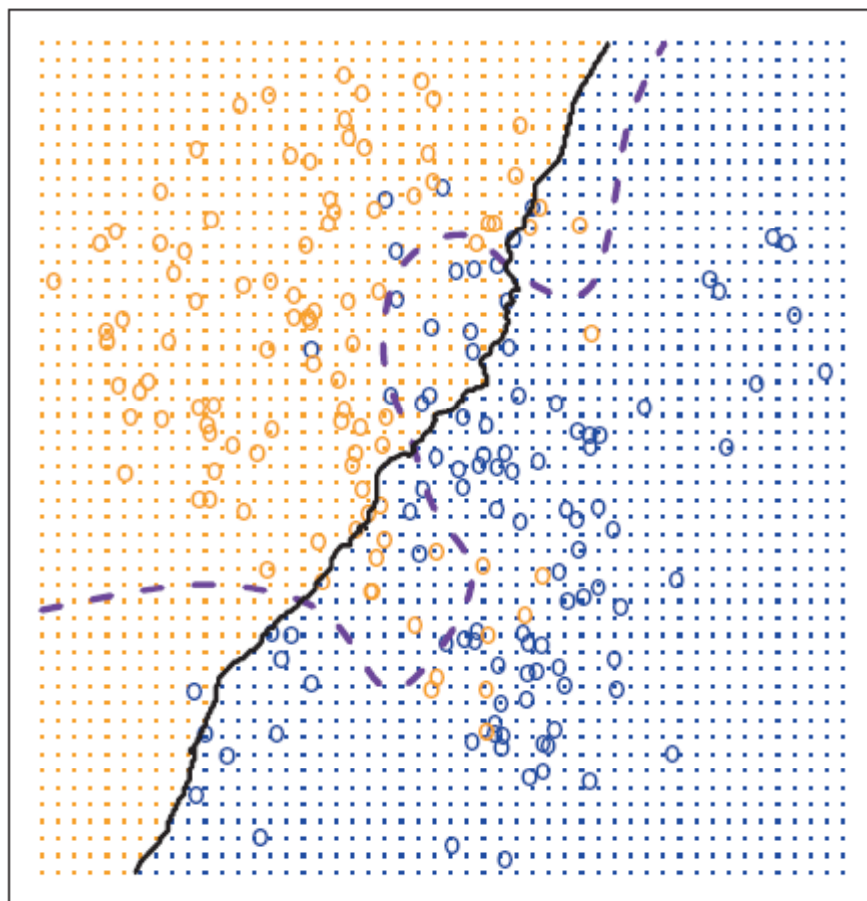




KNN:  $K=1$



KNN:  $K=100$



## B. 基于 $m$ -fold cross validation 的 $K$ 值选择

超参数空间：离散/连续  
数据集的使用：交叉验证  
超参数空间的某点的评价：  
错误率

以单轮  $m$ -fold cross validation 为例：

**STEP1. 训练集** 随机打乱，均分成  $m$  等份，每一份的训练样本数目  $\frac{N}{m}$ .

**STEP2.** 对于每个备选  $K$  值，

2-1. for  $i = 1, \dots, m$  do

拿出第  $i$  份作为 **验证集**，其余  $m-1$  份构成 **估计集**

利用 **估计集**，对 **验证集** 的每个样本进行类别预测，得 **验证集** 预测错误率  $Err_i(K)$

2-2. 估计对应于该备选  $K$  值的

$$\left\{ \begin{array}{l} \text{平均错误率 } \mu_{Err(K)} = \frac{1}{m} \sum_{i=1}^m Err_i(K) \\ \text{标准差 } \sigma_{Err(K)} = \sqrt{\frac{1}{m} \sum_{i=1}^m [Err_i(K) - \mu_{Err(K)}]^2} \end{array} \right.$$

表示为  $\mu_{Err(K)} \pm \sigma_{Err(K)}$

**STEP3.** 取最小  $\mu_{Err(K)}$  对应的  $K$  值为最终选择结果；

若同时有多个  $K$  值有最小  $\mu_{Err(K)}$ ，则取较小  $\sigma_{Err(K)}$  对应的  $K$  值。

1. 分类问题的一般描述
2. K近邻分类算法的描述
3. K近邻分类的三个基本要素

## 3.1 距离度量

典型的距离度量方式、样本的规范化预处理

## 3.2 超参数K值的确定

m-fold cross validation + 分类模型的评价指标

## 3.3 决策规则

胜者为王(多数表决)、加权投票

4. K近邻算法的实现—kd树(k-dimensional Tree)
5. K近邻分类模型的评价

## 方式1. 多数表决(胜者为王)——传统的K近邻决策方式

观测 $\mathbf{x}$ 的**K个近邻** $N_K(\mathbf{x}) = N_{K, \omega_1}(\mathbf{x}) \cup \dots \cup N_{K, \omega_C}(\mathbf{x})$

来自第 $\omega_j$ 类的近邻数:

$$k_j = \sum_{\mathbf{x}_i \in N_K(\mathbf{x})} I(y_i = \omega_j) = |N_{K, \omega_j}(\mathbf{x})| \quad j = 1, \dots, C$$

**决策规则**：若 $k_n = \max_{j=1, \dots, C} k_j$ , 则将 $\mathbf{x}$ 决策为第 $\omega_n$ 类.

样本 $\mathbf{x}$ 与 $\omega_j$ 类的相似度:  $\text{Sim}(\mathbf{x}, \omega_j) = \frac{|N_{K, j}(\mathbf{x})|}{K} = \frac{k_j}{K}$

则 $\mathbf{x}$ 的类别 $y$ 预测:  $\hat{y} = \arg \max_{\omega_j \in \{\omega_1, \omega_2, \dots, \omega_C\}} \text{Sim}(\mathbf{x}, \omega_j)$

## 方式1. 多数表决 (胜者为王)--续



### 特殊情况的处理

$x$ 的**K个近邻** $N_K(x)$ 中，多个类别训练样本数目同时最大

即：存在多个 $n \in \{1, \dots, C\}$ ，同时满足 $k_n = \max_{j=1, \dots, C} k_j$ ,

则可采取如下方式之一，进行 $x$ 的最终类别预测：

- A.** 随机选择其中一个类别
- B.** 将 $x$ 分到其最近邻的那个类别
- C.** 针对这些类别 $k_n$ 个近邻，分别计算相应均值向量，将 $x$ 分到与其最近的均值向量的那个类别.
- D.** 将 $x$ 分到最紧凑的那个类别 (即：  $k_n$  近邻距离最小)



## 方式2. 基于距离的加权投票

观测 $\mathbf{x}$ 的**K个近邻** $N_K(\mathbf{x}) = N_{K, \omega_1}(\mathbf{x}) \cup \dots \cup N_{K, \omega_C}(\mathbf{x})$

由近及远,  $N_K(\mathbf{x})$ 中**K个训练样本**类别标号 $a_1, \dots, a_K$ ,

**K个训练样本**关于 $\mathbf{x}$ 的距离为 $\delta_{a_1} \leq \delta_{a_2} \leq \dots \leq \delta_{a_K}$

**加权投票**: 统计第 $\omega_j$ 类的训练样本的**决策权重**

$$W_j = \sum_{i=1}^K w(\delta_{a_i}) I(a_i = \omega_j) \quad j = 1, \dots, C$$

**类别决策**: 若 $W_n = \max_{j=1, \dots, C} W_j$ , 则将 $\mathbf{x}$ 决策为第 $\omega_n$ 类.

如何确定K个近邻的投票权重?

$x$ 的前**K个近邻** $N_K(x)$ 的投票权重的估计方式

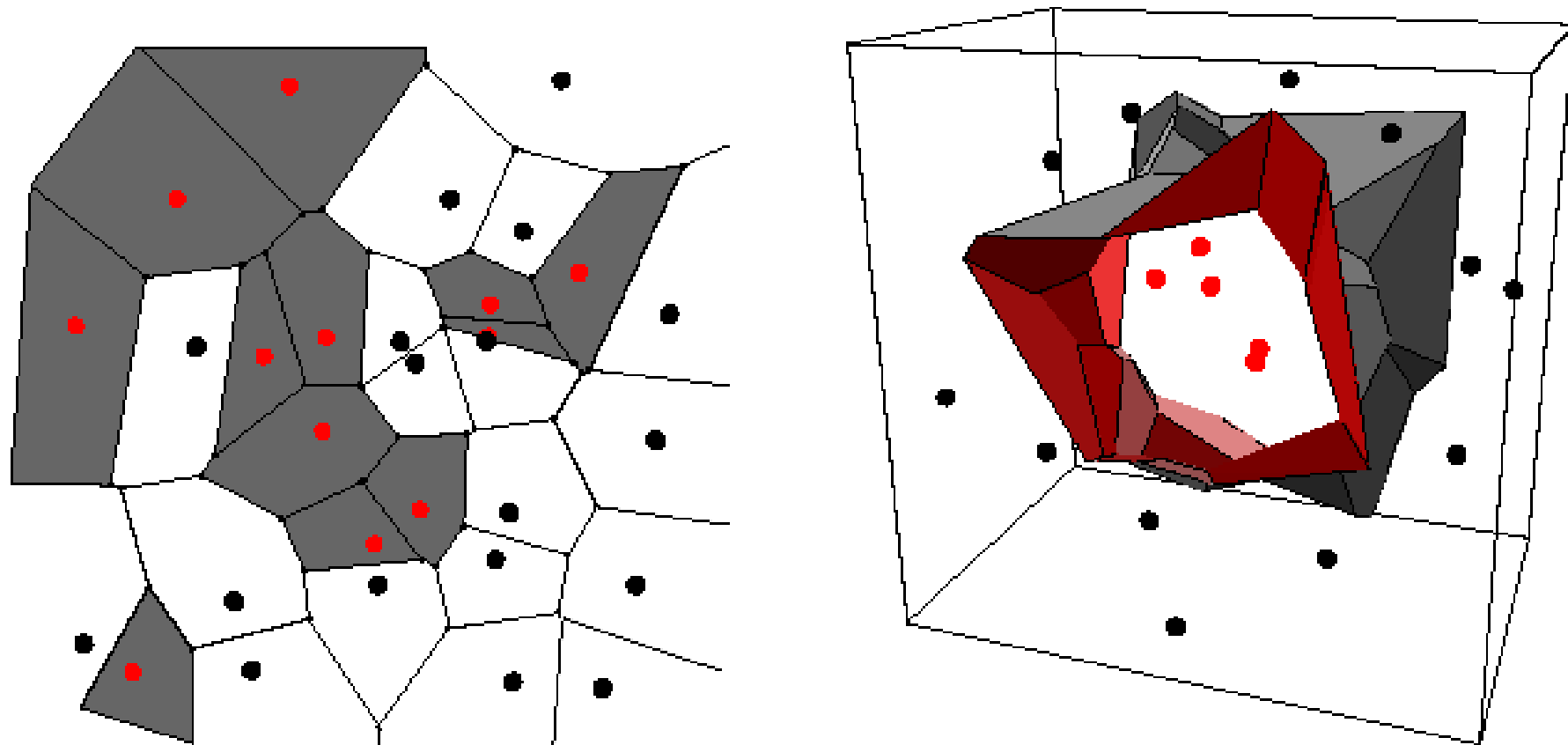
$$A. w(\delta_{a_i}) = \begin{cases} 0, & \delta_{a_i} = \delta_{a_K} \\ \frac{\delta_{a_K} - \delta_{a_i}}{\delta_{a_K} - \delta_{a_1}}, & \delta_{a_i} \neq \delta_{a_K} \\ 1, & \delta_{a_i} = \delta_{a_1} \end{cases}$$

$$B. w(\delta_{a_i}) = \frac{1}{\delta_{a_i}^2}$$

$$C. w(\delta_{a_i}) = \exp[-\delta_{a_i}^2]$$

$$D. w(\delta_{a_i}) = \frac{1}{\delta_{a_i}}, \quad \delta_{a_i} \neq 0$$

各近邻的归一化权重： $w(\delta_{a_i}) \leftarrow \frac{w(\delta_{a_i})}{\sum_{j=1}^K w(\delta_{a_j})}, i = 1, \dots, K$



图：最近邻规则、两类问题，基于训练样本点的特征空间的Voronoi网格划分。

左图，二维特征空间；右图，三维特征空间

1. 分类问题的一般描述
2. K近邻分类算法的描述
3. K近邻分类的三个基本要素
  - 3.1 距离度量
  - 3.2 超参数K值的确定
  - 3.3 决策规则
4. K近邻算法的实现—kd树(k-dimensional Tree, k维树)  
树的构建、基于树的搜索
5. K近邻分类模型的评价

基于**训练样本集** $D = \{(x_i, y_i), i = 1, \dots, N\}$ ,  $x_i \in R^d$

进行 **$K$** 近邻分类, 时间复杂度为 $O(Nd)$

**$K$** 近邻法省略了“监督式学习”阶段

其对任意观测样本 $x$ 的类别决策, 均依赖于**训练集的“记忆”**

**$K$** 近邻法也称为:

*Non – Generalizing Machine Learning Method*

如何 **“有效记忆”** 训练集 ?

如何在“记住”的训练集内**搜索** $x$ 的 **$K$** 近邻 ?

注意区分:

k近邻的“k”、kd树的“k”

训练样本集  $D = \{(x_i, y_i), i = 1, \dots, N\}$ , 并且  $x_i \in R^d$

## [1] kd树的构建----训练集的“有效记忆”

**构建递归二叉树**, 将  $d$  维特征空间的训练集按层划分, 形成若干子集, 每个子集对应一个子区域

## [2] 分枝定界法 (Branch and Bound) 近邻搜索 ---- $x$ 的近邻搜索

对于任意  $x \in R^d$ , 将分枝定界法用于  $kd$  树, 快速搜索测试样本  $x$  的近邻。

## kd树算法涉及两个部分：构建kd树、搜索kd树



### 算法1----kd树的构建 CreateKDTree

(平衡)递归二叉树

输入：

(1) 样本集  $D = \{(x_i, y_i), i = 1, \dots, N\}$

其所在空间超矩形体  $Range$

其中：

$x_i = [x_i^{(1)}, \dots, x_i^{(d)}]^T$  ----第  $i$  个训练样本的特征向量， $x_i \in R^d$

kd树是一个以划分方式存储  $k$  维空间数量有限的观测点集的数据结构--二叉树结构

(2) 结点分裂的终止条件  $m_0$ ：

到达该结点的训练样本数  $< m_0$  例： $m_0 - 1$

输出： kd树结构

例：Scikit-learn默认值 = 30

**STEP1. 开始：令 $j=0$ ，构造根节点。**

(1) **确定切分特征***split*.

$l=j \bmod d + 1 = 1$ , 选择以 $x^{(l)}=x^{(1)}$ 特征为切分特征*split*坐标轴.

(2) **确定切分点.**

将 $D$ 内所有样本按照切分特征*split*取值升序排列，称*split*取值的中位数为**切分点** $s$ .

采用垂直于*split*坐标轴、并经过切分点的超平面为**切分面**.

(3) **确定 $D$ 内经过切分面的样本点，并保存在根结点处.**

(4) **切分超矩形区域** $Range$ ，得左、右子区域.

采用**切分面**，将 $Range$ 区域切分为：

左子区域  $LeftRange = \{x | x \in Range \text{ 并且 } x[split] < s\}$

右子区域  $RightRange = \{x | x \in Range \text{ 并且 } x[split] > s\}.$



(5)数据集 $D$ 的左右子集的生成。

左子集  $LeftD = \{(x, y) | (x, y) \in D, \text{ 并且 } x[split] < s\}$

右子集  $RightD = \{(x, y) | (x, y) \in D, \text{ 并且 } x[split] > s\}$

(6)由根结点生成深度为1的左右子结点.

左子结点, 其对应数据集以及超矩形区域为  $LeftD, LeftRange$

右子结点, 其对应数据集以及超矩形区域为  $RightD, RightRange$

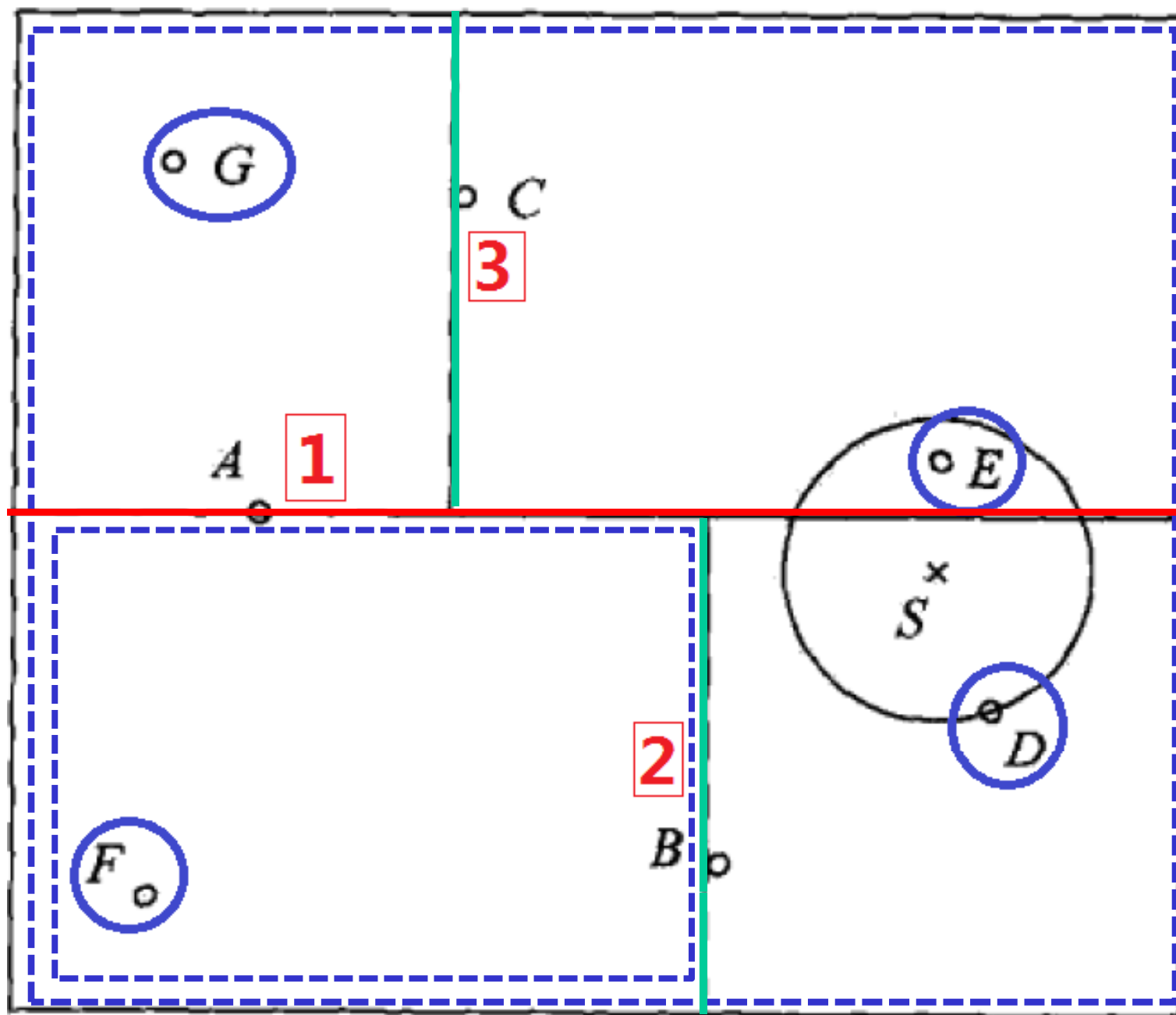
**STEP2. 根结点的左子树生成.**

若 $|LeftD| \geq m_0$ , 对于深度为 $j = 1$ 的左子结点, 基于 $LeftD$ 以及 $LeftRange$ , 调用**CreateKDTree**, 递归生成左子树.

**STEP3. 根结点的右子树生成.**

若 $|RightD| \geq m_0$ , 对于深度为 $j = 1$ 的右子结点, 基于 $RightD$ 以及 $RightRange$ , 调用**CreateKDTree**, 递归生成右子树.

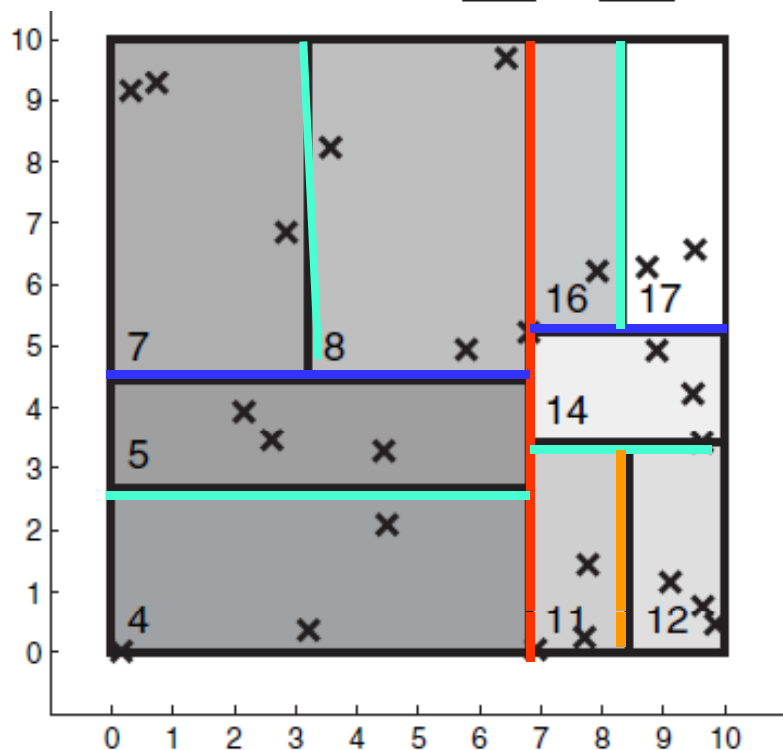
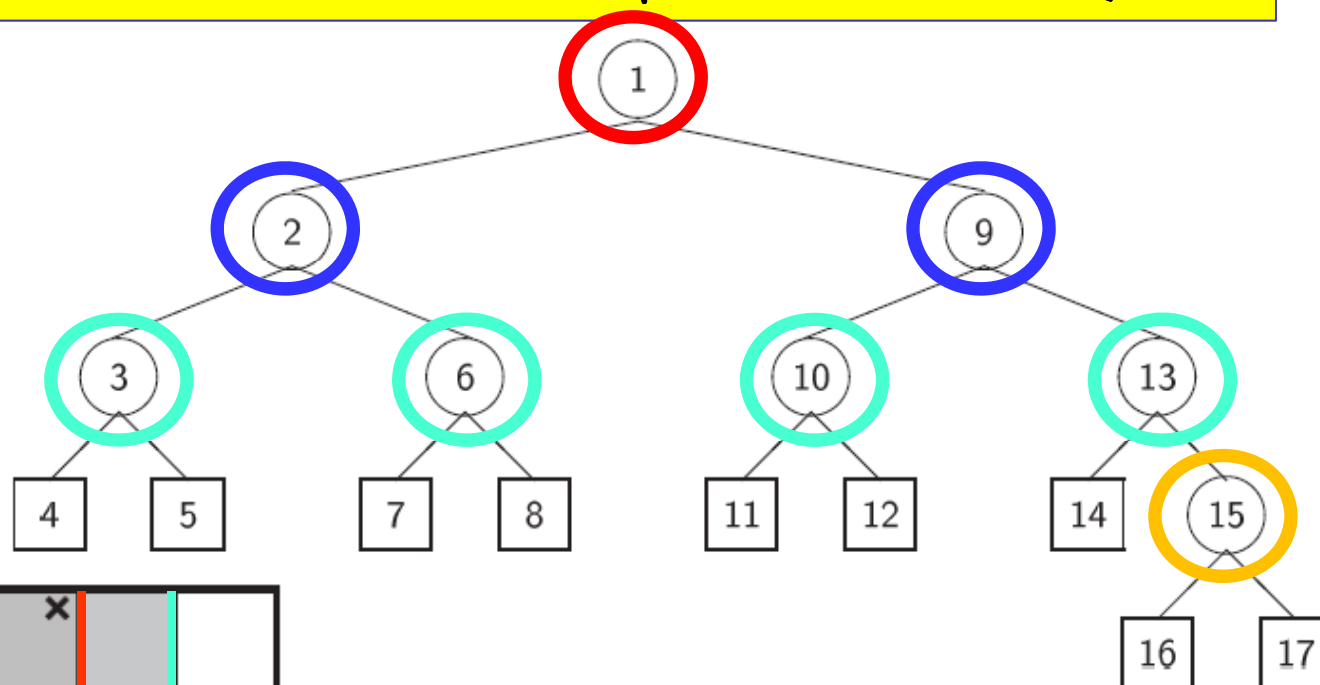
例:基于训练样本A,B,C,D,E,F,G构建二维特征空间的kd树.

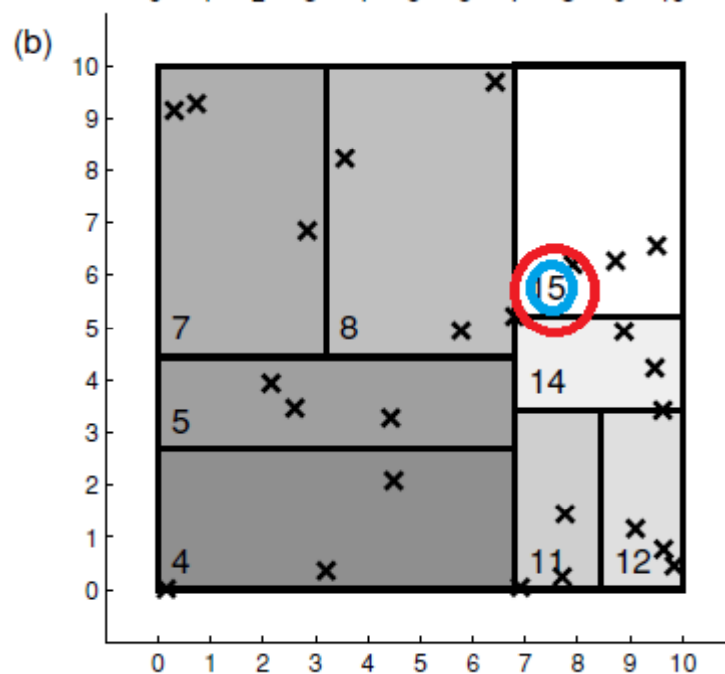
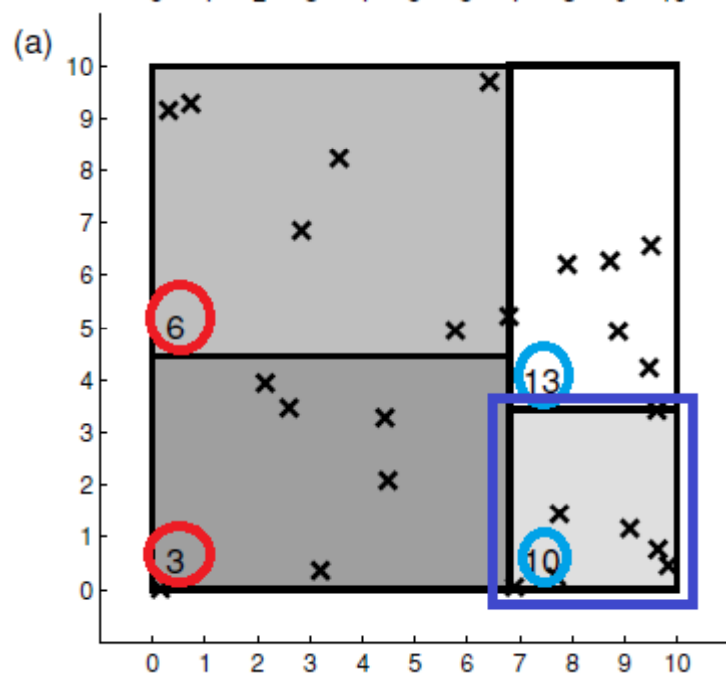
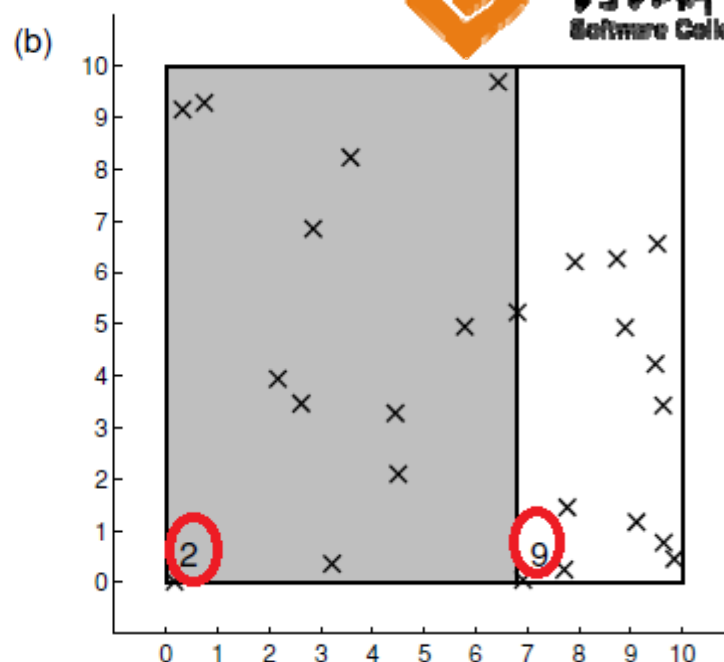
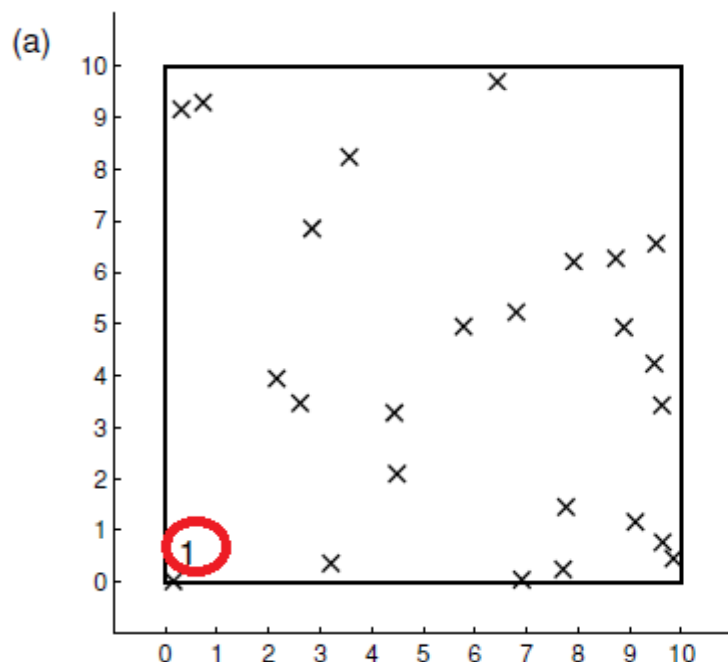


$m_0=1$

例：基于二维特征空间25个训练样本，构建kd树。

$m_0=3$





## 4.2 搜索kd树

算法：基于分支定界法 (*Branch and Bound*) 的kd树搜索

输入： (1)已经构造的kd树

(2)任意观测样本  $\mathbf{x} = [x^{(1)}, \dots, x^{(d)}]^T$

输出：  $\mathbf{x}$ 的近邻

## STEP1. 二叉树搜索, 找到含样本 $x$ 的叶结点

从根结点出发, 沿树自上向下,

将样本 $x$ 的**切分特征** $split$ 取值与当前结点切分点 $s$ 比较

$\left\{ \begin{array}{l} \text{若 } x[split] < s, \text{ 则该结点的左子结点胜出} \\ \text{若 } x[split] > s, \text{ 则该结点的右子结点胜出} \end{array} \right.$

逐级进入生成结点, 直到找到 $kd$ 树中含样本 $x$ 的叶结点

## STEP2. 在该叶结点对应的样本子集内搜索, 找到初始近邻。

将其初始化为样本 $x$ 的“**当前最近邻** $x_0$ ”

得到二者的距离 $d_0 = d(x, x_0)$

### STEP3. 从该叶结点开始，递归向上回溯



河北师范大学软件学院  
Software College of Hebei Normal University

对每个结点重复操作如下：

- (1) 记**当前结点**的父结点为 $q$
- (2) 以样本 $x$ 为中心，以当前最近距离 $d_0$ 为半径，构建**超球**.

则 $x$ 的**真正最近邻** $x_0^*$ 一定在该超球内.

- (3) 检查**超球**是否与**当前结点**的**兄弟节点**对应的**超矩形体**相交.

**若是相交：**

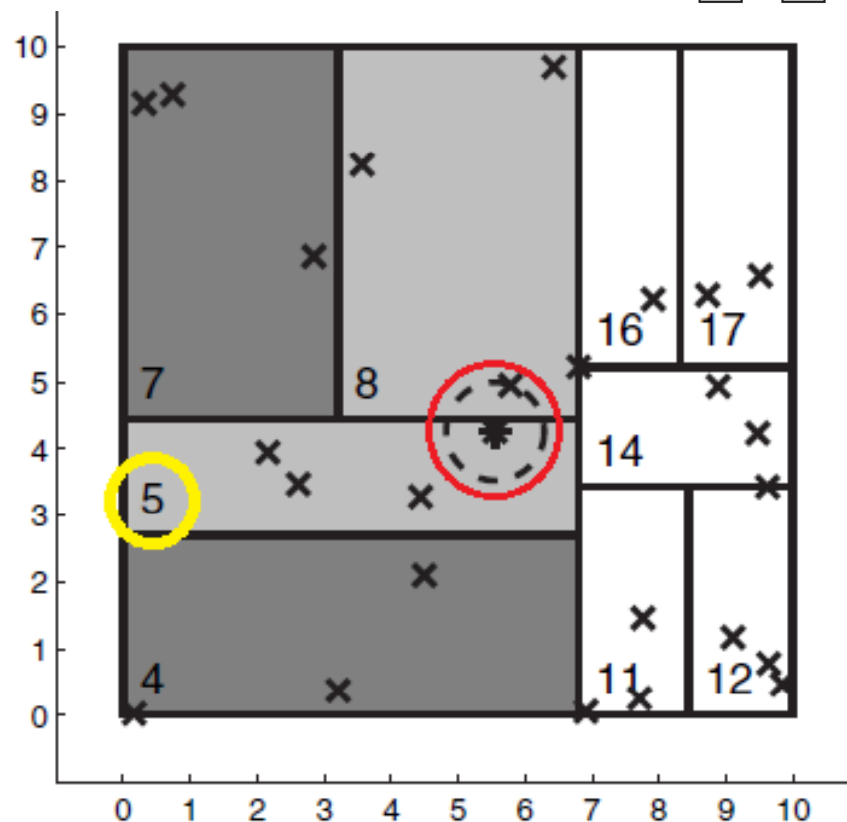
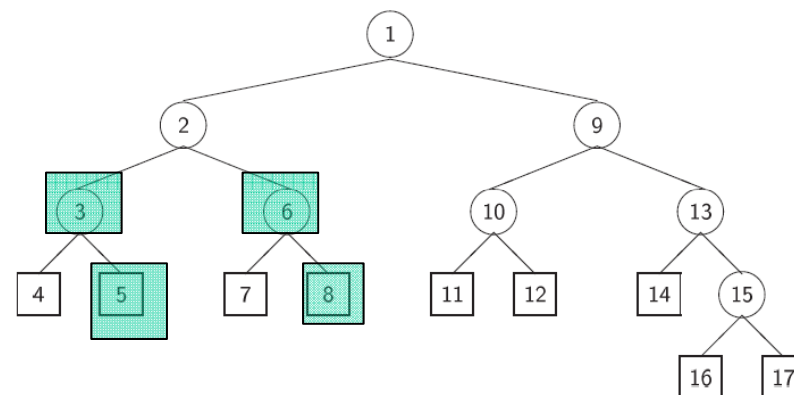
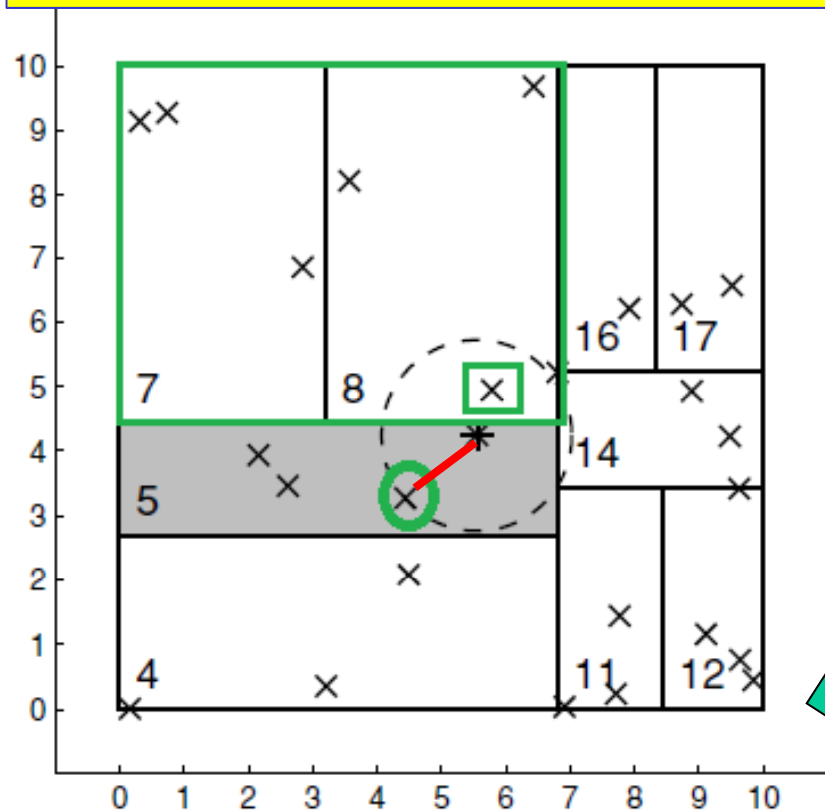
该**兄弟节点**所对应超矩形区域内，可能存在关于 $x$ 更近的样本. 移动至该**兄弟节点**，递归进行近邻搜索，若存在更近的近邻，则更新近邻及超球.

**若不相交：**

则向上回退, 更新当前节点、父结点

**STEP4.** 若回退至根节点，则搜索结束，输出最终的近邻 $x_0^*$

例：基于分支定界法，搜索测试样本(\*)的最近邻。





1. 分类问题的一般描述
2. K近邻分类算法的描述
3. K近邻分类的三个基本要素
  - 3.1 距离度
  - 3.2 超参数K值的确定
  - 3.3 决策规则
4. K近邻算法的实现—kd树(k-dimensional Tree)  
树的构建、基于树的搜索
5. (K近邻)分类模型的评价

测试样本集  $D_{Test} = \{(x_i, y_i), i = 1, \dots, N\}$

分类正确率  $Acc = \frac{\sum_{i=1}^N I(y_i = \hat{y}_i)}{N} \times 100\%$

分类错误率  $Err = \frac{\sum_{i=1}^N I(y_i \neq \hat{y}_i)}{N} \times 100\%$

## 5.1 基于混淆矩阵的两类别分类模型的性能评价

混淆矩阵: confusion matrix


(1) 样本的两种自然类别状态

通常设定感兴趣的一类为正类

类别标号	类别各种名称			
1	Positive (P)	正	阳性	Case Samples (病理样本)
0或-1或2	Negetive (N)	负	阴性	Control Samples (对照样本)

## (2) 两类决策的混淆矩阵(confusion matrix)

正确分类；错误分类；决策阈值



自然状态 True Value Actual Value	预测输出(Predicted Outcome)	
	Positive (Predicated 1)	Negative (Predicated 0 or -1)
Positive (True 1)	<i>a</i> <b>True Positive (TP)</b> 真阳性 Hits	<i>b</i> <b>False Negative (FN)</b> 假阴性 Misses
Negative (True 0 或 -1)	<i>c</i> <b>False Positive (FP)</b> 假阳性 False Alarms	<i>d</i> <b>True Negative (TN)</b> 真阴性 True Rejections

## (2) 两类决策的混淆矩阵(confusion matrix)—概率形式

正确分类；错误分类

自然状态 True Value Actual Value	预测输出(Predicated Outcome)	
	Positive (Predicated 1)	Negetive (Predicated 0 or -1)
Positive (True 1)	$S_n$ --灵敏度 $P(\text{Predicated } P   \text{True } P)$	$\beta$ --假阴性率 $P(\text{Predicated } N   \text{True } P)$
Negetive (True 0 或-1)	$\alpha$ --假阳性率 $P(\text{Predicated } P   \text{True } N)$	$S_p$ --特异度 $P(\text{Predicated } N   \text{True } N)$

$$P(\text{Predicated } P | \text{True } P) + P(\text{Predicated } N | \text{True } P) = 1$$

$$P(\text{Predicated } P | \text{True } N) + P(\text{Predicated } N | \text{True } N) = 1$$

### (3) 基于两类别混淆矩阵的评价指标

[1] 总体**正确率 / 准确率 / 准确度** (OverallAccuracy)  $Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$

[2] 总体**错误率** (ErrorRate)  $Error = \frac{FP + FN}{TP + FP + TN + FN}$

[3] (阳性类) **查准率 / 精度** (Precision)  $Precision = \frac{TP}{TP + FP}$

[4] (阳性类) **查全率 / 召回率** (Recall) / **灵敏度** (Sensitivity)

/ (阳性类) **命中率 / 真阳性率** (TruePositiveRate)  $S_n = Recall = \frac{TP}{TP + FN}$

[5] **特异度** (Specificity) / **真阴性率** (TrueNegativeRate)  $S_p = \frac{TN}{TN + FP}$

[6] **假阴性率** (FalseNegativeRate) / (阳性类) **漏报率** (MissedRate)  $\beta = \frac{FN}{FN + TP}$

[7] **假阳性率** (FalsePositiveRate) / (阳性类) **虚警率** (FalseAlarmRate)  $\alpha = \frac{FP}{TN + FP}$

[8](阳性类的) $F_\beta$  - Score以及 $F_1$  - Score ---- Precision与Recall的调和平均

$$F = \frac{1}{\frac{1}{\beta^2 + 1} \left( \beta^2 \frac{1}{Recall} + 1 \frac{1}{Precision} \right)} = \frac{(\beta^2 + 1) Precision \cdot Recall}{\beta^2 Precision + Recall}$$

$\beta=1$ 时,  $F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} = \frac{2 Precision \cdot Recall}{Precision + Recall}$

[9]马修相关系数(Matthews Correlation Coefficient, MCC)

类别不均衡的两类别分类, 公平度量模型预测能力

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FN)(TN + FP)(TP + FP)(TN + FN)}}$$

# [10] *Kappa*系数 (*Cohen's kappa*)

*Cohen's kappa* measures **the agreement between two raters** who each classify  $N$  items into  $C$  mutually exclusive categories.

$$K = \frac{p_o - p_e}{1 - p_e}$$
$$= \frac{\frac{TP + TN}{TP + FN + TN + FP} - \frac{(TP + FN)(TP + FP) + (TN + FP)(TN + FN)}{(TP + FN + TN + FP)^2}}{1 - \frac{(TP + FN)(TP + FP) + (TN + FP)(TN + FN)}{(TP + FN + TN + FP)^2}}$$

理想情况，完全一致， $K = 1$ ； $K$ 值越小，越不一致，甚至为负。



# [10] *Kappa*系数(*Cohen's kappa*)--续

关于 $p_0, p_e$ 的具体说明:

(1)  $p_0$ --实际一致率 
$$p_0 = \frac{TP + TN}{TP + FN + TN + FP}$$

(2)  $p_e$ --理论一致率

$p_e$ 表示一个样本被抽取的类别与决策类别一致的概率

从随机抽取的角度:

$$\left\{ \begin{array}{l} \text{一个样本真实类别是正类、并被决策为正类的概率:} \\ p_{e1} = \frac{TP + FN}{TP + FN + TN + FP} \cdot \frac{TP + FP}{TP + FN + TN + FP} \\ \text{一个样本真实类别为负类、并被决策为负类的概率:} \\ p_{e2} = \frac{TN + FP}{TP + FN + TN + FP} \cdot \frac{TN + FN}{TP + FN + TN + FP} \end{array} \right.$$
$$p_e = p_{e1} + p_{e2} = \frac{(TP + FN)(TP + FP) + (TN + FP)(TN + FN)}{(TP + FN + TN + FP)^2}$$

## 5.2 基于混淆矩阵的多类别分类模型的性能评价

C类别的分类，混淆矩阵为C行\*C列

	class $j$ predicted by a classifier									
true class $i$	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
'0'	97	0	0	0	0	0	1	0	0	1
'1'	0	98	0	0	1	0	0	1	0	0
'2'	0	0	96	1	0	1	0	1	0	0
'3'	0	0	2	95	0	1	0	0	1	0
'4'	0	0	0	0	98	0	0	0	0	2
'5'	0	0	0	1	0	97	0	0	0	0
'6'	1	0	0	0	0	1	98	0	0	0
'7'	0	0	1	0	0	0	0	98	0	0
'8'	0	0	0	1	0	0	1	0	96	1
'9'	1	0	0	0	3	1	0	0	0	95

$C$ 类别分类, **混淆矩阵** $A = [a_{ij}]_{C \times C}$

$a_{ij}$  -- 矩阵 $A$ 的第 $i$ 行第 $j$ 列, 真实类别为第 $i$ 类但决策为第 $j$ 类的样本数.

则由矩阵 $A = [a_{ij}]_{C \times C}$  可知:

参与决策的第 $i$ 类样本总数  $= \sum_{j=1}^C a_{ij}$

决策类别为第 $j$ 类样本总数  $= \sum_{i=1}^C a_{ij}$

[1] **正确率 / 准确率 / 准确度 (Accuracy)**

$$Accuracy = \frac{\sum_{i=1}^C a_{ii}}{\sum_{i=1}^C \sum_{j=1}^C a_{ij}}$$

[2] **错误率 (ErrorRate)**

$$Error = 1 - Accuracy$$

[3] **第*i*类查准率 / 精度 (Precision)**

$$Precision\_i = \frac{\text{正确决策为第}i\text{类的样本总数}}{\text{决策为第}i\text{类的样本总数}} \times 100\% = \frac{a_{ii}}{\sum_{k=1}^C a_{ki}} \times 100\%$$

[4] **第*i*类查全率 / 召回率 (Recall) / 第*i*类命中率**

$$Recall\_i = \frac{\text{正确决策为第}i\text{类的样本总数}}{\text{参与决策的第}i\text{类样本总数}} \times 100\% = \frac{a_{ii}}{\sum_{k=1}^C a_{ik}} \times 100\%$$

[5] **第*i*类  $F_1$  - Score** ---  $Precision\_i$  与  $Recall\_i$  的调和平均

$$F_1^{(i)} = \frac{2Precision\_i \cdot Recall\_i}{Precision\_i + Recall\_i} \quad i = 1, \dots, C$$

[6]  $C$  个类别的 **宏平均  $F_1$  - Score (Macro - averaging  $F_1$ )**

$$Macro\_F_1 = \frac{1}{C} \sum_{i=1}^C F_1^{(i)}$$

[7] *Kappa*系数 (*Cohen's kappa*)

*Cohen's kappa* measures **the agreement between two raters** who each classify  $N$  items into  $C$  mutually exclusive categories.

$$K = \frac{p_0 - p_e}{1 - p_e} = \frac{\frac{\sum_{i=1}^C a_{ii}}{\sum_{i=1}^C \sum_{j=1}^C a_{ij}} - \frac{\sum_{i=1}^C \left[ \frac{\sum_{j=1}^C a_{ij} \sum_{k=1}^C a_{ki}}{\sum_{i=1}^C \sum_{j=1}^C a_{ij}} \right]}{\left( \sum_{i=1}^C \sum_{j=1}^C a_{ij} \right)^2}}{1 - \frac{\sum_{i=1}^C \left[ \frac{\sum_{j=1}^C a_{ij} \sum_{k=1}^C a_{ki}}{\sum_{i=1}^C \sum_{j=1}^C a_{ij}} \right]}{\left( \sum_{i=1}^C \sum_{j=1}^C a_{ij} \right)^2}}$$

理想情况，完全一致， $K = 1$ ； $K$ 值越小，越不一致，甚至为负。

关于 $p_0, p_e$ 的具体说明:

$$(1) \text{ } p_0 \text{ -- 实际一致率} \quad p_0 = \frac{\sum_{i=1}^C a_{ii}}{\sum_{i=1}^C \sum_{j=1}^C a_{ij}}$$

$$(2) \text{ } p_e \text{ -- 理论一致率} \quad p_e = \sum_{i=1}^C p_{ei} = \frac{\sum_{i=1}^C \left[ \sum_{j=1}^C a_{ij} \sum_{k=1}^C a_{ki} \right]}{\left( \sum_{i=1}^C \sum_{j=1}^C a_{ij} \right)^2}$$

$p_e$ 表示一个样本被抽取的类别与决策类别一致的概率

样本被随机抽取的真实类别是第 $i$ 类、并同时被决策为第 $i$ 类的概率:

$$p_{ei} = \frac{\sum_{j=1}^C a_{ij}}{\sum_{i=1}^C \sum_{j=1}^C a_{ij}} \cdot \frac{\sum_{k=1}^C a_{ki}}{\sum_{i=1}^C \sum_{j=1}^C a_{ij}} \quad i = 1, \dots, C$$



## PART2. K近邻回归

K-Nearest Neighbors Regression, KNN回归

1. 回归问题的一般描述
2. K近邻回归算法的描述
3. K近邻回归的预测规则
4. K近邻回归模型的评价

高维空间K近邻回归模型的性能远不如低维空间



## 1. 回归问题的一般描述

给定训练样本集  $\{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$ .

其中:

$\mathbf{x}_i$ ----第*i*个观测样本的输入  $\mathbf{x}_i \in R^d$

$y_i$ ----第*i*个观测样本的输出  $y_i \in R$

**要求:**

基于上述样本集, 构建预测模型  $\mathbf{y} = f(\mathbf{x})$ --**模型的回归学习**;

对特征空间的任意观测 $\mathbf{x}$ 的输出 $y$ 进行预测--**模型的使用**

1. 回归问题的一般描述
2. K近邻回归算法的描述
3. K近邻回归的预测规则
4. K近邻回归模型的评价

## 2.1 K近邻回归模型的引入

### 非参数法回归.

无需明确预测模型参数化形式，直接由训练样本数据预测任意观测 $x$ 所导致的输出 $\hat{f}(x)$ .

### 懒惰学习

### 准备工作轻松

训练样本集、近邻数 $K$ 、欧式距离

## 2.2 K近邻回归算法的描述

**输入 :** (1) **训练样本集**  $D_{\text{train}} = \{(x_i, y_i), i = 1, \dots, N\}$ , 并且有:

$x_i \in R^d$  ----第*i*个训练样本的特征向量

$y_i \in R$  ----第*i*个训练样本的目标输出

(2) **观测样本**  $x$

**输出 :** 观测样本 $x$ 应导致的输出 $y$ .

**STEP1. 选择K**

**STEP2.** 在训练集 $D_{\text{train}}$ 内找到样本 $x$ 的**K个近邻**, 记为 $N_K(x)$

**STEP3.** 结合指定的**预测规则**, 对 $x$ 应导致输出 $y$ 进行预测

$$\hat{y} = \frac{1}{K} \sum_{x_i \in N_K(x)} y_i$$

不同的K值、  
不同的预测规则，

会导致不同的预测结果

1. 回归问题的一般描述
2. K近邻回归算法的描述
3. K近邻回归的预测规则
4. K近邻回归模型的评价

## 方式1. 等权平均——传统的K近邻决策方式

对于给定的观测 $\mathbf{x}$ ，利用距离度量，在训练样本集

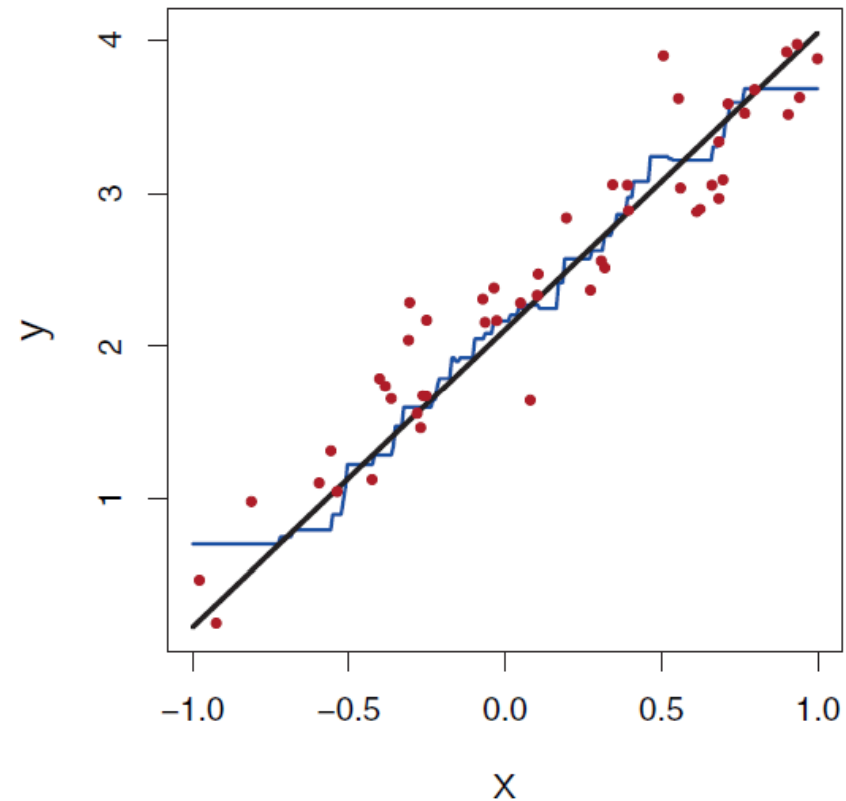
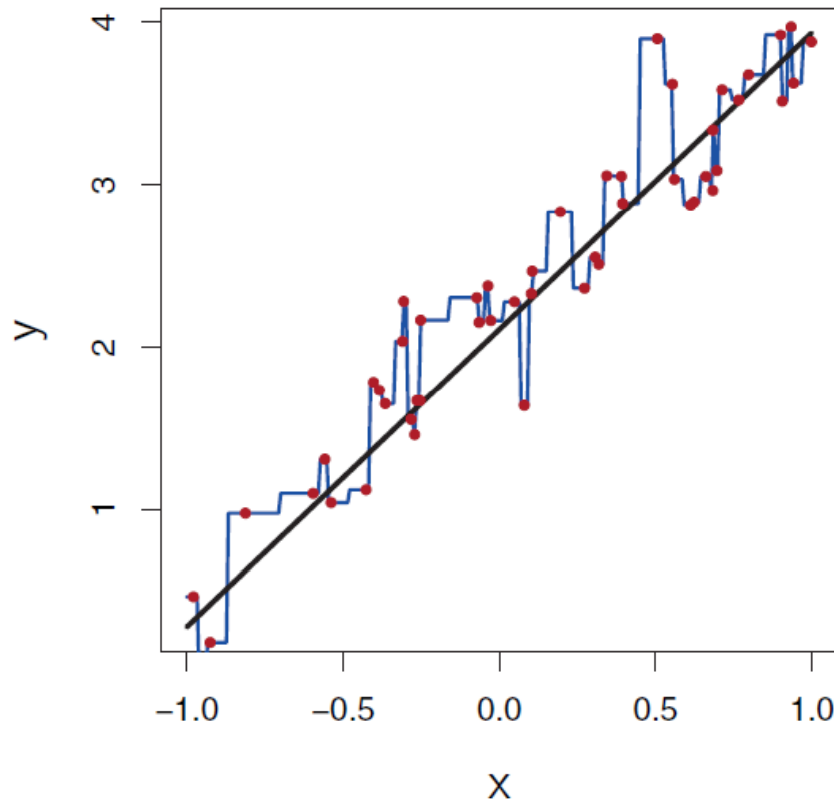
$D_{train} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N\}$ 内确定 $\mathbf{x}$ 的前**K个近邻** $N_K(\mathbf{x})$

则 $\mathbf{x}$ 应的输出 $\mathbf{y}$ 预测为

$$\hat{\mathbf{y}} = \frac{1}{K} \sum_{\mathbf{x}_i \in N_K(\mathbf{x})} \mathbf{y}_i$$

## 例1. K近邻回归模型

100个训练样本(左  $K=1$ , 右  $K=9$ )



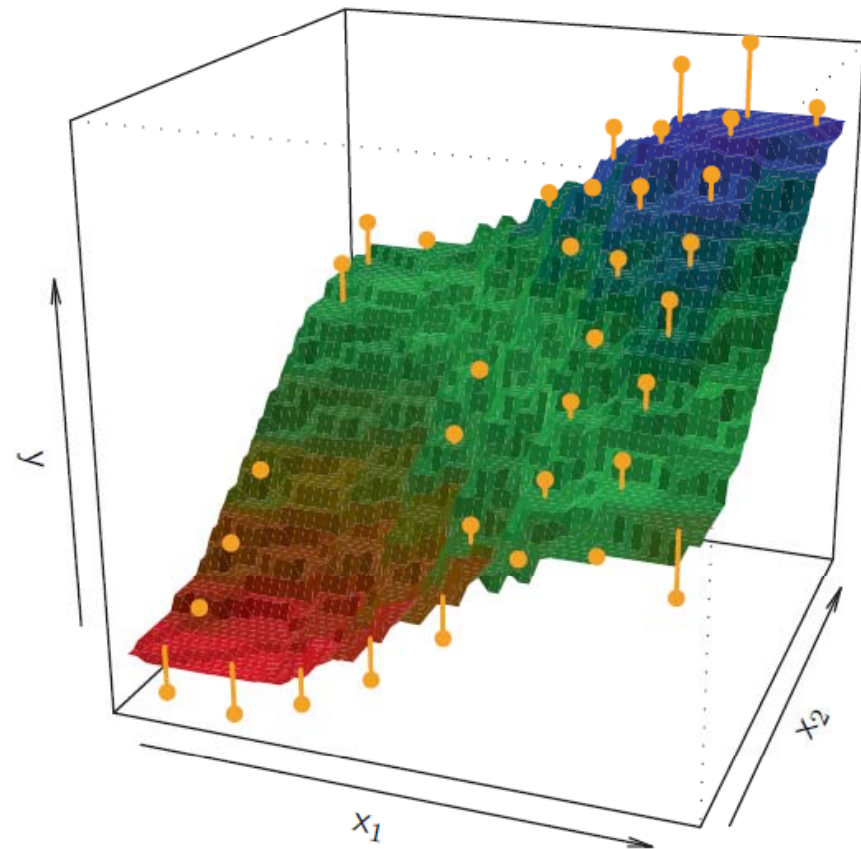
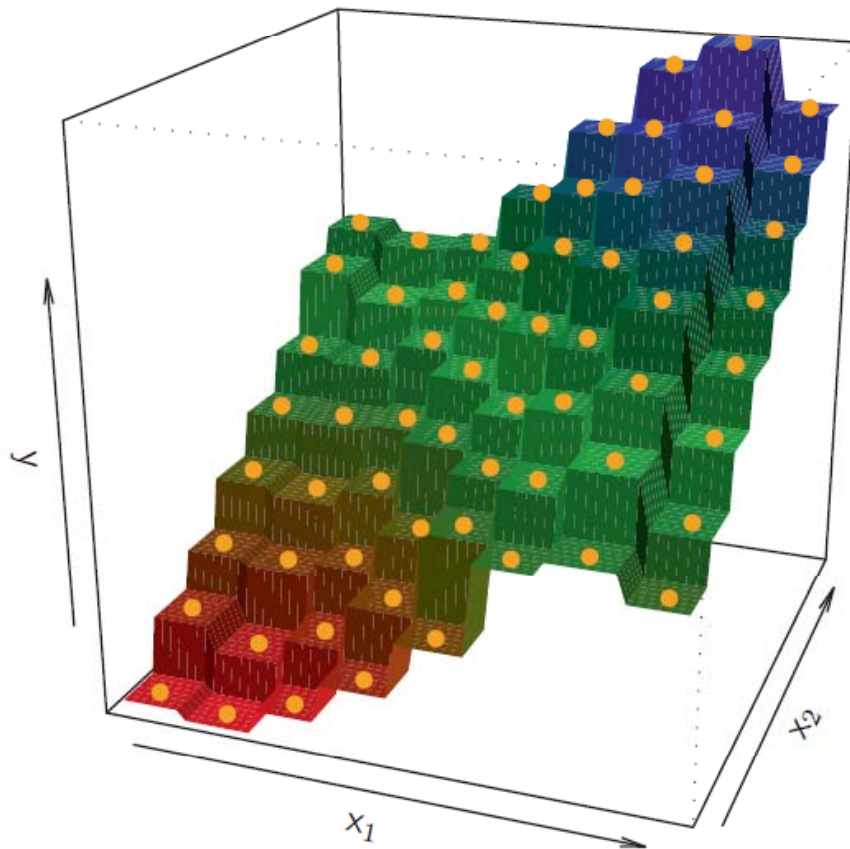
*The blue curve corresponding to  $K = 9$  represents a smoother fit.*



例: KNN regression on a two-dimensional data set with 64 observations (orange dots).

Left:  $K = 1$  results in a rough step function fit.

Right:  $K = 9$  produces a much smoother fit



## 方式2. 基于距离的加权平均



对于给定的观测 $\mathbf{x}$ , 利用距离度量, 在训练样本集  
 $D = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N\}$ 内确定 $\mathbf{x}$ 的前**K个近邻** $N_K(\mathbf{x})$

对于 $\forall \mathbf{x}_j \in N_K(\mathbf{x})$ , 该训练样本关于 $\mathbf{x}$ 的距离为 $d(\mathbf{x}, \mathbf{x}_j)$   
引入基于距离的预测权值, 如:

$$w(d) = \frac{1}{d^2} \quad \text{或} \quad w(d) = \exp[-d^2]$$

则 $\mathbf{x}$ 的输出 $\mathbf{y}$ 预测为

$$\hat{\mathbf{y}} = \frac{\sum_{\mathbf{x}_i \in N_K(\mathbf{x})} w(d(\mathbf{x}, \mathbf{x}_i)) \mathbf{y}_i}{\sum_{\mathbf{x}_i \in N_K(\mathbf{x})} w(d(\mathbf{x}, \mathbf{x}_i))}$$

1. 回归问题的一般描述
2. K近邻回归算法的描述
3. K近邻回归的预测规则
4. K近邻回归模型的评价

思考：如何采用m-fold CV法，对K近邻回归模型进行K值的优选？

测试样本集  $D_{Test} = \{(x_i, y_i), i = 1, \dots, N\}$

### 1. 均方误差 (*Mean Squared Error, MSE*)

$$MSE = \frac{1}{N} \sum_{i=1}^N [y_i - \hat{y}_i]^2$$

### 2. 均方根误差 (*Root Mean Squared Error, RMSE*)

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N [y_i - \hat{y}_i]^2}$$

### 3. 平均绝对误差 (*Mean Absolute Error, MAE*)

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

### 4. 中值绝对误差 (*Median Absolute Error*)

$$Median\_AE = median_{i=1, \dots, N} \{|y_i - \hat{y}_i|\}$$



## K近邻小结

1. 分类问题的描述、分类模型的性能评价
2. 回归问题的描述、回归模型的性能评价
3. 距离度量
4. 样本的规范化预处理

### 5. K近邻分类

- K近邻分类的算法描述
- 如何面向分类，进行K值优选
- 决策规则

### 6. K近邻回归

- K近邻回归的算法描述
- 如何面向回归问题，进行K值优选
- 预测规则

## 关于KD树的说明(参见Scikit-learn)

Regarding the Nearest Neighbors algorithms, if two neighbors, have identical distances but different labels, the results will depend on the ordering of the training data.

Though the KD tree approach is very fast for low-dimensional ( $d < 20$ ) neighbors searches, it becomes inefficient as grows very large: this is one manifestation of the so-called “curse of dimensionality”.

For small data sets ( less than 30 or so), brute force algorithms can be more efficient than a tree-based approach.