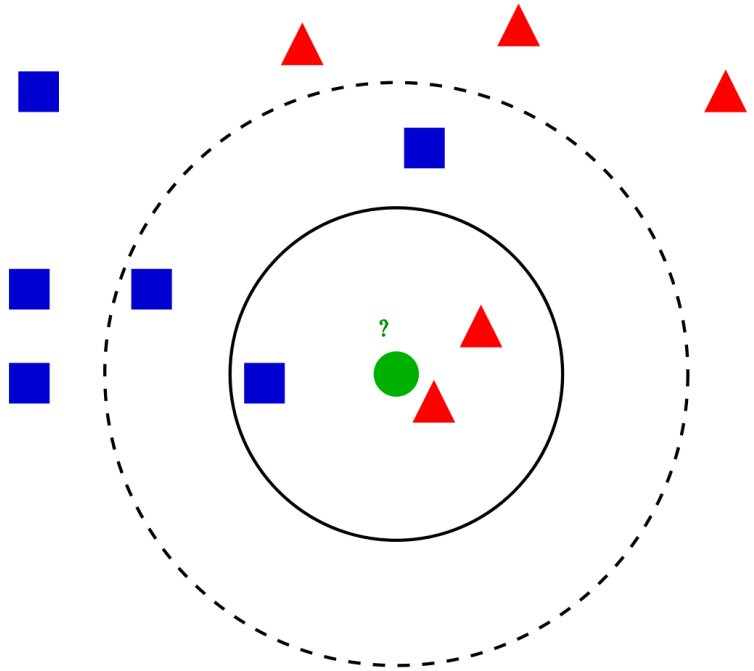
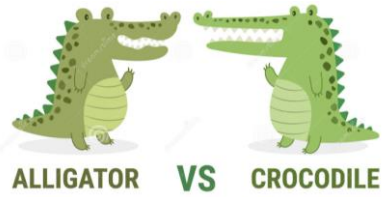
非参数估计与K近邻法

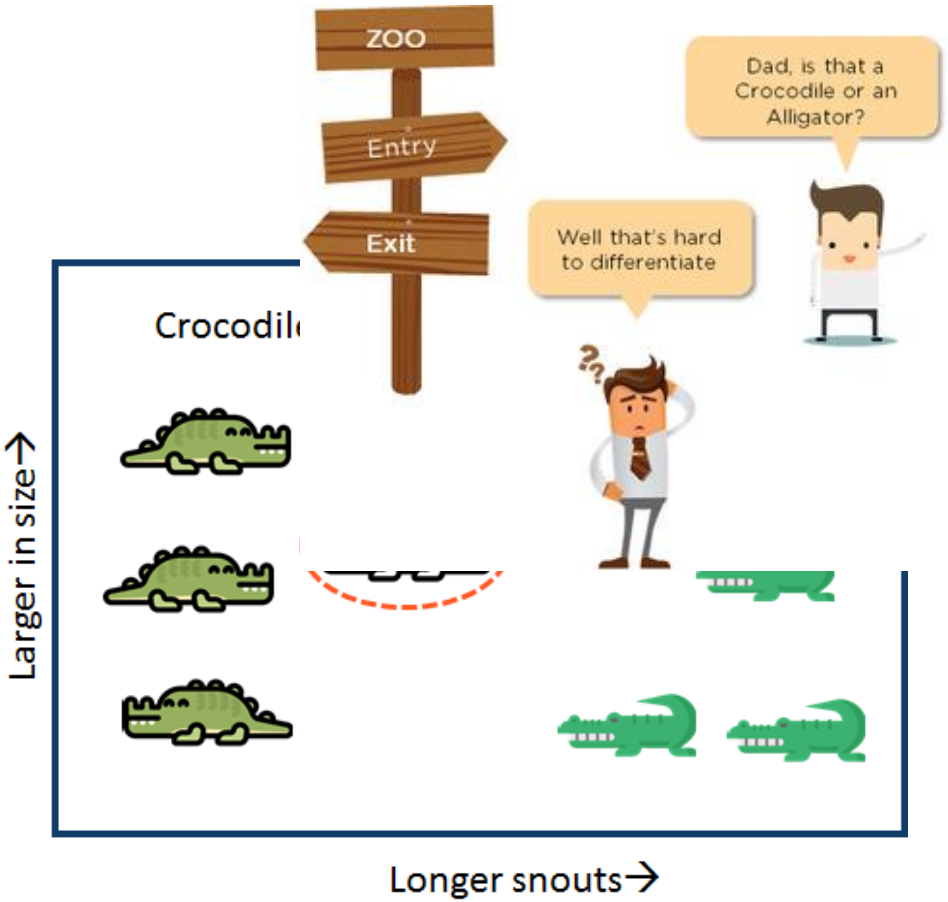


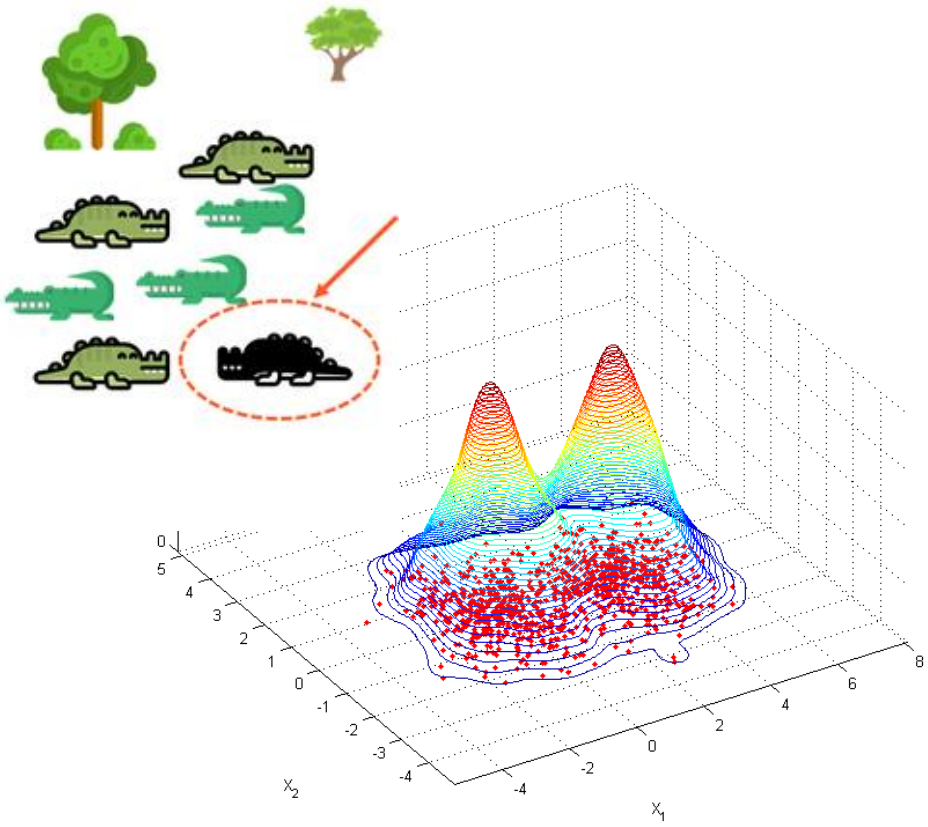
赵海涛

haitaozhao@ecust.edu.cn

开课一张图


ALLIGATOR
VS
CROCODILE


ZOO
Entry
Exit
Dad, is that a Crocodile or an Alligator?
Well that's hard to differentiate
Crocodil
Larger in size→
Longer snouts→


0
5
4
3
2
1
0
-1
-2
-3
-4
X2
-4
-2
0
2
4
6
8
X1

大纲

- 核密度估计与K近邻估计
- K近邻分类器
- 例子

非参数密度估计

- 对概率密度函数建模，无需对其函数形式做任何假设。

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})}$$

- 任何非参数密度估计技术都必须处理控制估计密度平滑度的“平滑”参数的选择。
- 讨论基于以下三种方法

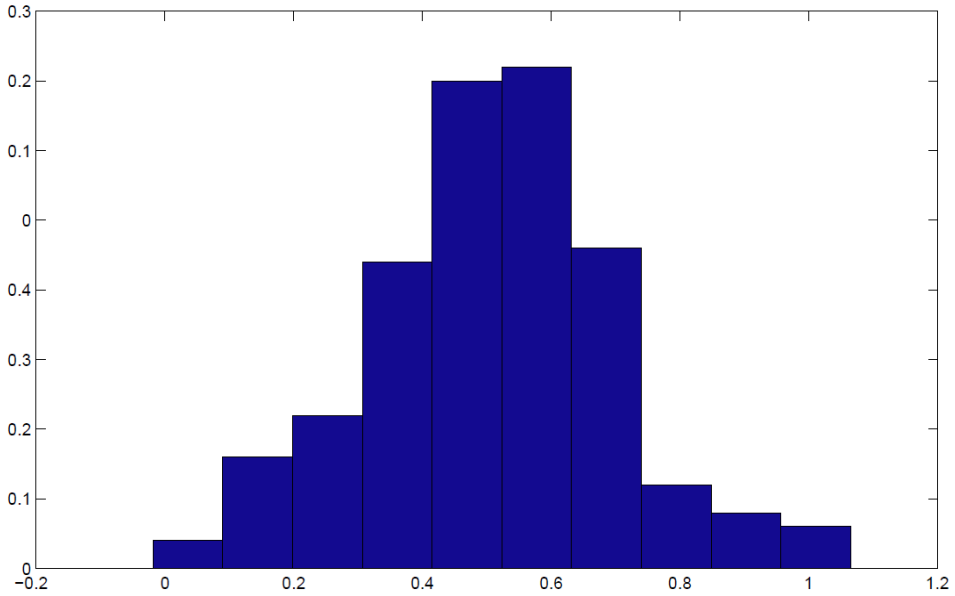
  a. 直方图法

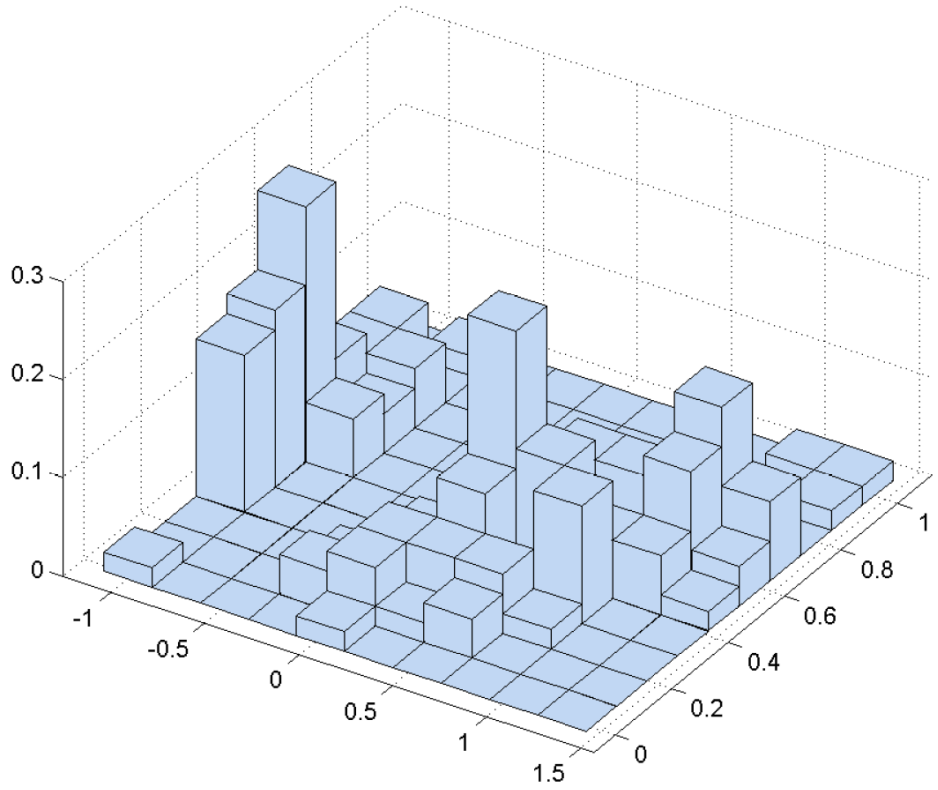
  b. 核密度估计

  c. K近邻估计

基于直方图的密度估计

- 假设每个数据点$\boldsymbol{x}$由n维特征向量表示 $(x_1, x_2, \cdots, x_n)$.
- 直方图法是通过将每个x_i轴划分成 M个区间，并通过落在相应区域内的点的比例来估计概率密度的方法





基于直方图的密度估计

- 区间的个数（或区间的大小）可以作为平滑参数
- 区间的大小过小（如M 的值很大），那么估计的密度有尖刺（即噪音）
- 区间的大小过大（如M 的值很小）， 估计的密度将过于平滑
- 在实际应用中，需要选择合适的M 的值在这两种情况中取得折衷

基于直方图的密度估计的优点

- 一旦构建了直方图，就不再需要数据（即内存效率高）
- 只需要有关直方图区间大小和位置的信息
- 直方图可以顺序构建.....（即一次考虑一个数据，然后丢弃）

基于直方图的密度估计的缺点

- 估计的密度不平滑，并且在直方图区间的边界处具有不连续性。
- 很难扩展到高维数据
  - 考虑 d 维特征空间
  - 如果对每个特征维度划分M 个区间，我们最终得到M^d 个区域
  - 需要非常大规模的样本才能获得较好的密度估计 (否则，大量的区域不包含任何样本，估计的密度值为0)

密度估计

很多估计未知概率密度的核心思想都是非常简单的，尽管关于收敛性的严格证明可能需要较多技巧。最基本的一个事实是：向量$\boldsymbol{x}$落在区域R中的概率为：

$$P = \int_R p(\boldsymbol{x}')d\boldsymbol{x}'$$

我们可以通过估计概率P来估计概率密度函数$p(\boldsymbol{x})$。假设n个样本$\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$都是根据概率密度函数$p(\boldsymbol{x})$独立同分布的抽取而得到的。显然，其中$k$个样本落在区域$R$中的概率服从二项式定理：

$$P_k = \binom{n}{k} P^k (1-P)^{n-k}$$

密度估计

k的期望值为

$$E[k] = nP$$

如果将n看做给定值，则有

$$E\left[\frac{k}{n}\right] = P$$

方差

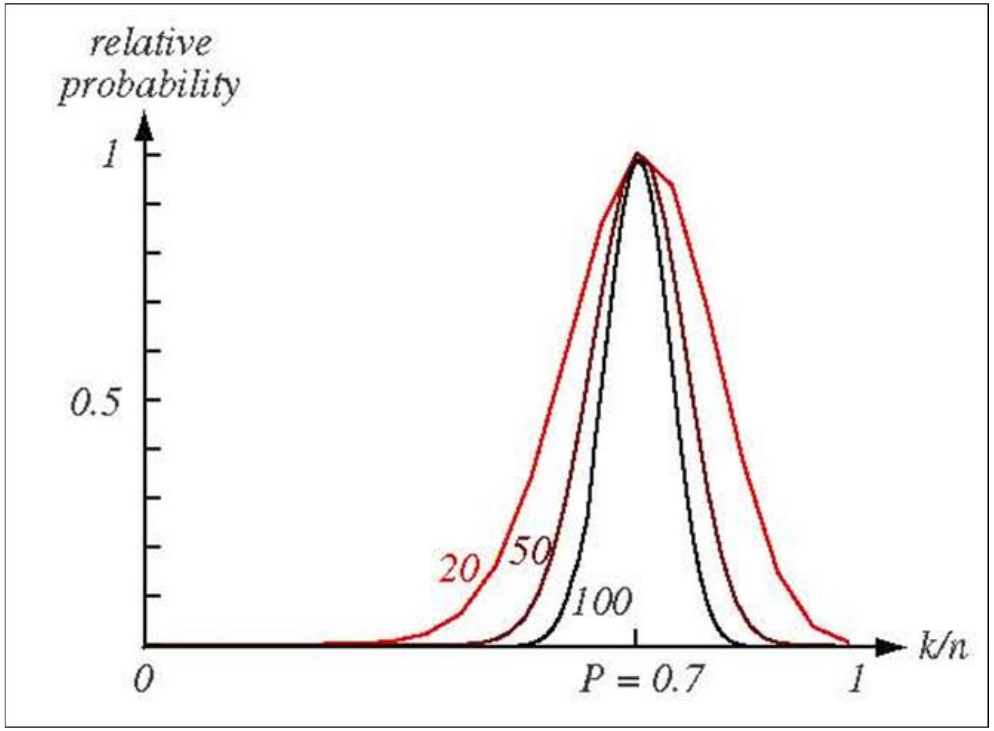
$$Var\left(\frac{k}{n}\right) = E\left[\left(\frac{k}{n} - P\right)^2\right] = \frac{P(1-P)}{n}$$

密度估计

当$n \to +\infty$, 概率将会急剧地达到峰值，因此:

$$P \approx \frac{k}{n}$$

(Approximation 1)



密度估计

如果我们假设$p(\boldsymbol{x})$是连续的，并且区域R足够小，以至于在这个区间中$p(\boldsymbol{x}')$几乎没有变化，那么有

$$P = \int_R p(\boldsymbol{x}')d\boldsymbol{x}' \approx p(\boldsymbol{x})V$$

其中$\boldsymbol{x}$为一个点，而V则是区域R所包含的体积 (Approximation 2)

密度估计

结合这两个近似我们可以得到

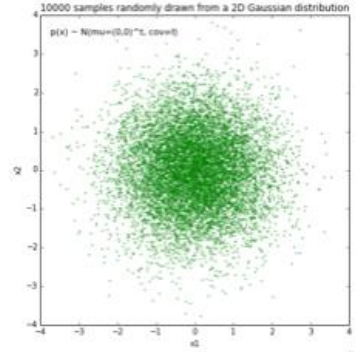
$$p(\boldsymbol{x}) \approx \frac{k/n}{V}$$

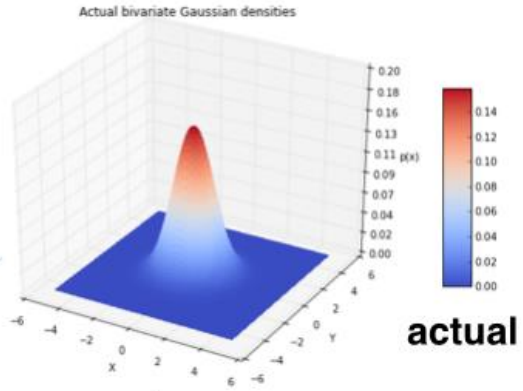
上面的近似是基于两个矛盾的假设

- R相对较大（即它包含许多样本，因此P_k会急剧地达到峰值）
  - Approximation 1
- R相对较小，因此$p(x)$在积分区域内近似恒定
  - Approximation 2

Parzen窗方法 (kernel-based estimation)

Our goal:

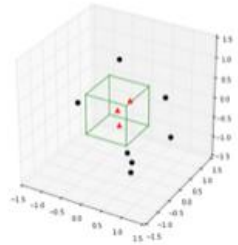


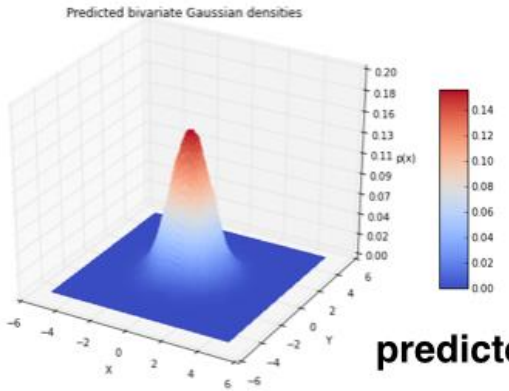


actual

Assuming we have samples drawn from a **unknown** distribution (here: bivariate Gaussian)

We want to **estimate** probability densities at certain points





predicted

Parzen窗方法 (kernel-based estimation)

问题：给定n个样本$\{\boldsymbol{x}_i\}_{i=1}^n$，对于给定向量$\boldsymbol{x}$，估计$p(\boldsymbol{x})$

为了说明估计概率密度函数的Parzen窗方法，

- 我们暂时假设区间R_n是一个d维的超立方体。如果令h_n表示超立方体一条边的长度，那么体积就是

$$V_n = h_n^d$$

- 通过定义如下的窗函数，我们能够解析地得到落在窗中的样本个数k_n的表达式：

$$K(\boldsymbol{u}) = \begin{cases} 1 & |u_j| \le \frac{1}{2}; \quad j = 1, \ldots, d \\ 0 & \text{其它} \end{cases}$$

Parzen窗方法

- 如果$\boldsymbol{x}_i$落在边长为h_n的体积为V_n的超立方体中，那么$K\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_n}\right)=1$，否则便为0。因此，超立方体中的样本个数就是

$$k_n=\sum_{i=1}^{n}K\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_n}\right)$$

- 代入

$$p_n(\boldsymbol{x})\approx\frac{k_n/n}{V_n}$$

- 我们可得到

$$p_n(\boldsymbol{x})=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}K\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_n}\right)$$

Parzen窗方法

- 密度估计是核函数和样本x_i的叠加
- $K(\boldsymbol{u})$ 用于采样进行插值
- 每个样本$\boldsymbol{x}_i$基于其与$\boldsymbol{x}$的距离对估计做出贡献

$K(u)$的性质

- 核函数 $K(\boldsymbol{u})$ 可以有更一般的形式（不只是超立方体）
- 为了让$p_n(\boldsymbol{x})$的估计合理，$K(\boldsymbol{u})$自身必须是有效的

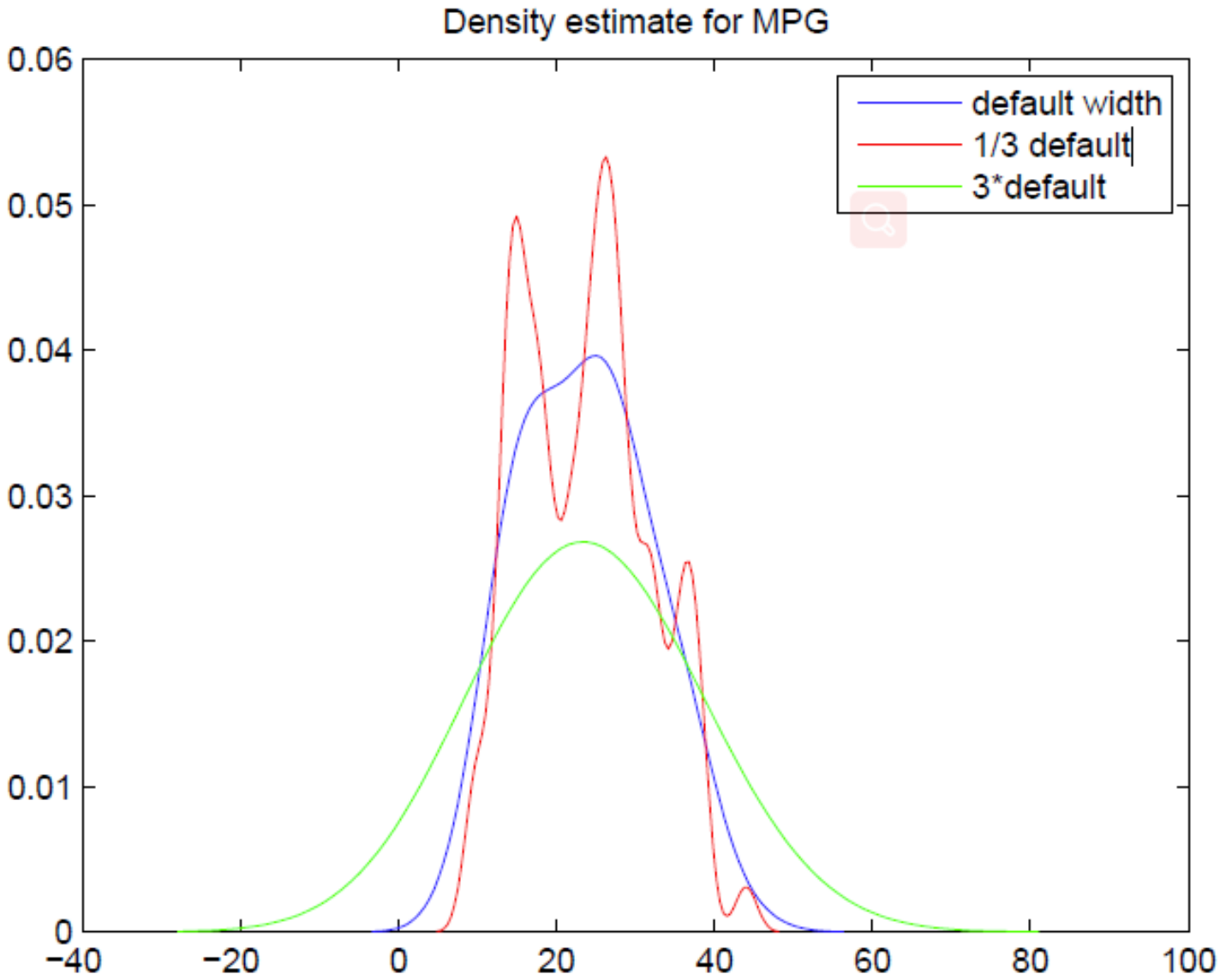
$$K(\boldsymbol{u}) \geq 0$$
$$\int K(\boldsymbol{u}) d\boldsymbol{u} = 1$$

h_n的选择

h_n是作为一个平滑参数，需要被优化

- 如果h_n非常大，$p_n(\boldsymbol{x})$是n个宽的，慢变的函数的叠加，因此$p_n(\boldsymbol{x})$是对$p(\boldsymbol{x})$的估计非常平滑，或者称为“散焦”的估计。
- 如果h_n很小，$p_n(\boldsymbol{x})$是n个以样本点为中心的尖脉冲的叠加——也就是一个充满噪声的估计。

h_n的选择


Density estimate for MPG
default width
1/3 default
3*default
0.06
0.05
0.04
0.03
0.02
0.01
0
−40
−20
0
20
40
60
80
100

$p_n(x)$估计的期望和方差

- 当 $V_n \to 0$，期望值将会趋近于 $p(\boldsymbol{x})$：

$$E(p_n(\boldsymbol{x})) = \int \frac{1}{V_n} K(\frac{\boldsymbol{x}-\boldsymbol{x}'}{h}) p(\boldsymbol{x}') d\boldsymbol{x}'$$

- 估计的方差

$$\mathrm{Var}(p_n(\boldsymbol{x})) \leq \frac{sup(K(\cdot))E(p_n(\boldsymbol{x}))}{nV_n}$$

- 使方差变小，可以构造

$$nV_n \to \infty (e.g., V_n = \frac{1}{\sqrt{n}})$$

kernel-based density estimation

- Data $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}, \boldsymbol{x}_i \in \mathbb{R}^d$
- Kernel function $K(\cdot)$
- Parameter kernel width h (is a smoothness parameter)
- We can obtain a probability density $f(\boldsymbol{x})$ over $\mathbb{R}^d$

$$f(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h^d} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right)$$

Mean shift algorithm

- **Idea** find points with $\nabla f(\boldsymbol{x}) = 0$
- Assume $K(\boldsymbol{z}) = \frac{1}{\sqrt{2\pi}} \exp(-\|\boldsymbol{z}\|^2/2)$ Gaussian kernel
- Then

$$\nabla f(\boldsymbol{x}) = -\frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right)(\boldsymbol{x} - \boldsymbol{x}_i)/h$$

- Local maximum of f is solution of implicit equation

$$x = \underbrace{\frac{\sum_{i=1}^{n} x_i K\left(\frac{x - x_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)}}_{\text{the mean shift } m(x)}$$

Mean shift algorithm (cont'd)

Input Data $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, kernel $K(\boldsymbol{z})$, h

1. for $i = 1{:}n$
   - $\boldsymbol{x} \leftarrow \boldsymbol{x}_i$

   a. iterate $\boldsymbol{x} \leftarrow m(\boldsymbol{x})$ until convergence to $\boldsymbol{m}_i$
2. group points with same $\boldsymbol{m}_i$ in a cluster

- Remarks
  - mean shift iteration guaranteed to converge to a max of f
  - computationally expensive

Mean shift (cont’d)



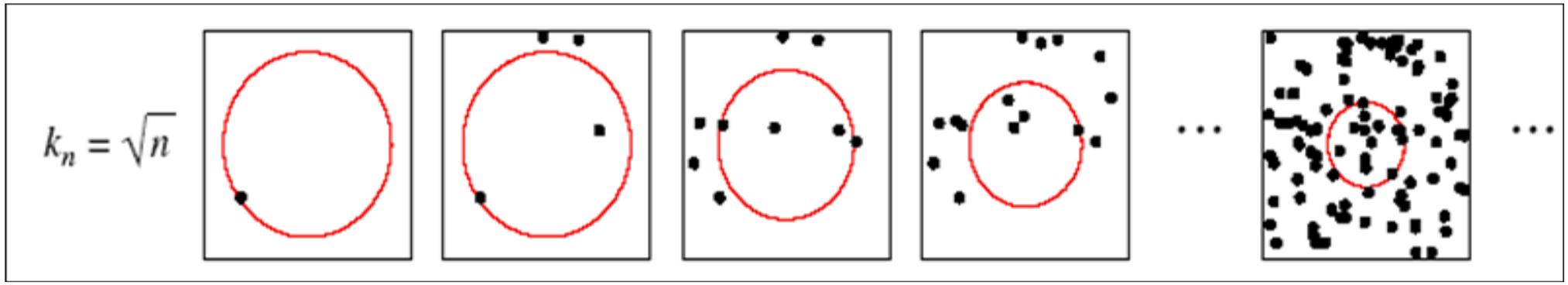

基于核函数方法的不足

- 需要大量样本
- 要求将所有样品保存
- 如果数据点的数量很大，则密度的评估可能会非常缓慢
- 可能的解决方案：使用更少的核，并根据数据调整位置和宽度（例如，混合高斯）

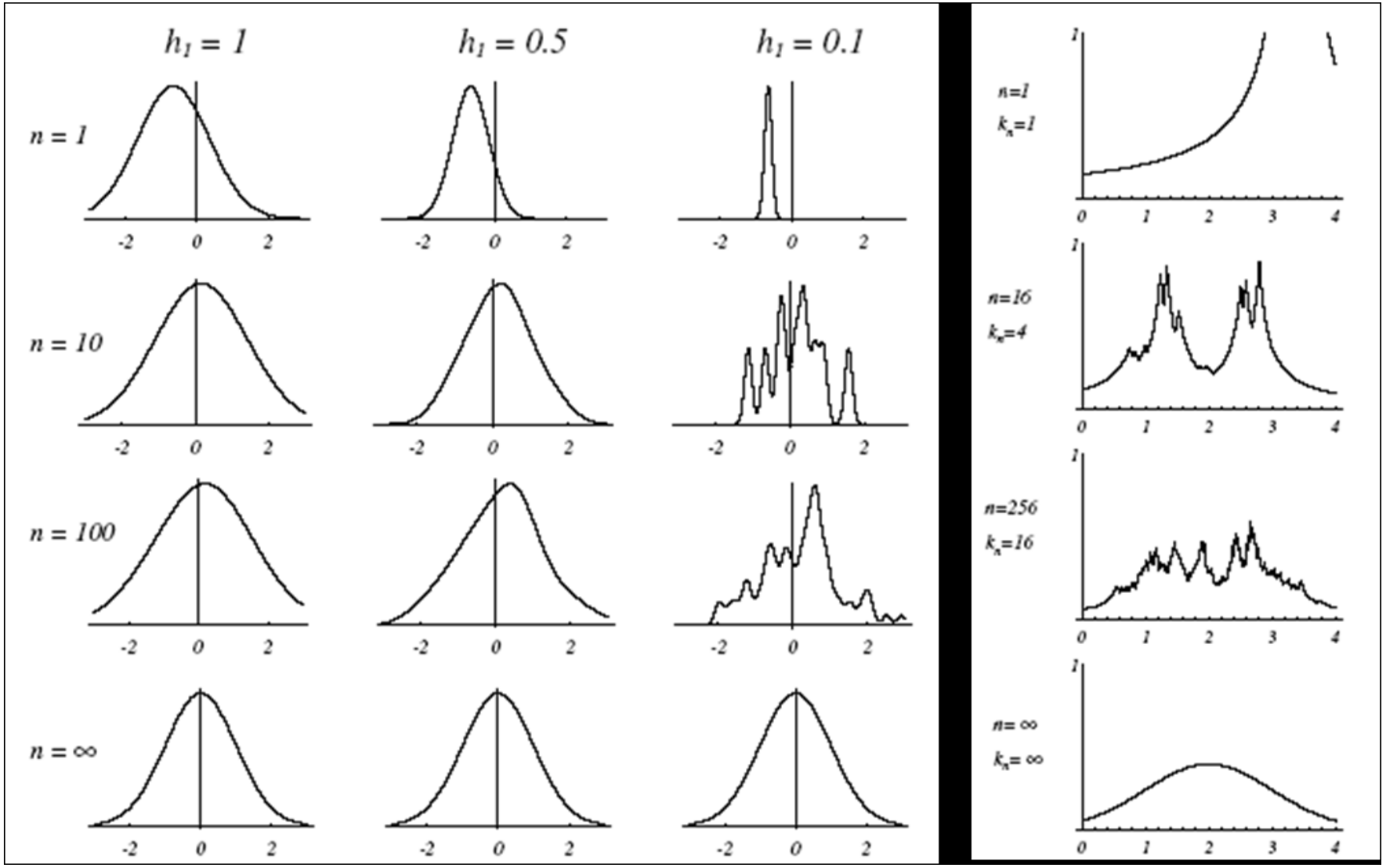
k_n近邻估计

- 固定 k_n 使对应的V_n改变：
  - 考虑包含特征$\boldsymbol{x}$超球体
  - 使球体的半径增加，直到超球体包含k_n 个数据为止.
  - V_n 即为对应超球体的体积

$$p_n(\boldsymbol{x}) \approx \frac{k_n/n}{V_n}$$



Parzen 窗法 vs k_n近邻估计



k_n近邻估计

- 实际的密度估计问题中，经常样本已经给定，如$\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n \sim p, \boldsymbol{x}_i \in \mathbb{R}^d$
- 对某个给定的$\boldsymbol{x}$，首先计算每个样本到$\boldsymbol{x}$的距离。令$D_k(\boldsymbol{x})$表示$\boldsymbol{x}$到它的第k个近邻点的距离，则k近邻密度估计如下

$$p_{knn}(\boldsymbol{x}) = \frac{k}{n} \cdot \frac{1}{V_d \cdot D_k^d(\boldsymbol{x})}$$

其中，$V_d = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}$是$d$维单位超球体的体积，$\Gamma(\cdot)$为Gamma函数。

- 假设一维数据点$1, 2, 6, 11, 13, 14, 20, 33$。分别用$k = 2$和$k = 5$估计$x = 5$出的密度 (距离为$4, 3, 1, 6, 8, 9, 15, 28$)
  - ✓ $p_{2nn}(5) = \frac{1}{24}$
  - ✓ $p_{5nn}(5) = \frac{5}{128}$

k_n近邻分类器

- 假设我们有c个类别，类别ω_i包含n_i个点

$$P(\omega_i|\boldsymbol{x}) = \frac{p_n(\boldsymbol{x}|\omega_i)P(\omega_i)}{p_n(\boldsymbol{x})}$$

- 对于给定的$\boldsymbol{x}$，我们找到 k_n个近邻数据点，假设 其中k_i 个点 属于ω_i类，那么我们有:

$$p_n(\boldsymbol{x}|\omega_i) = \frac{k_i}{n_i V_n}$$

k_n近邻分类器

- 先验概率

$$P(\omega_i) = \frac{n_i}{n}$$

- 后验概率

$$P(\omega_i|\boldsymbol{x}) = \frac{p_n(\boldsymbol{x}|\omega_i)P(\omega_i)}{p_n(\boldsymbol{x})} = \frac{k_i}{k_n}$$

其中 $p_n(\boldsymbol{x}) = \frac{k_n}{nV_n}$

k_n 近邻分类器

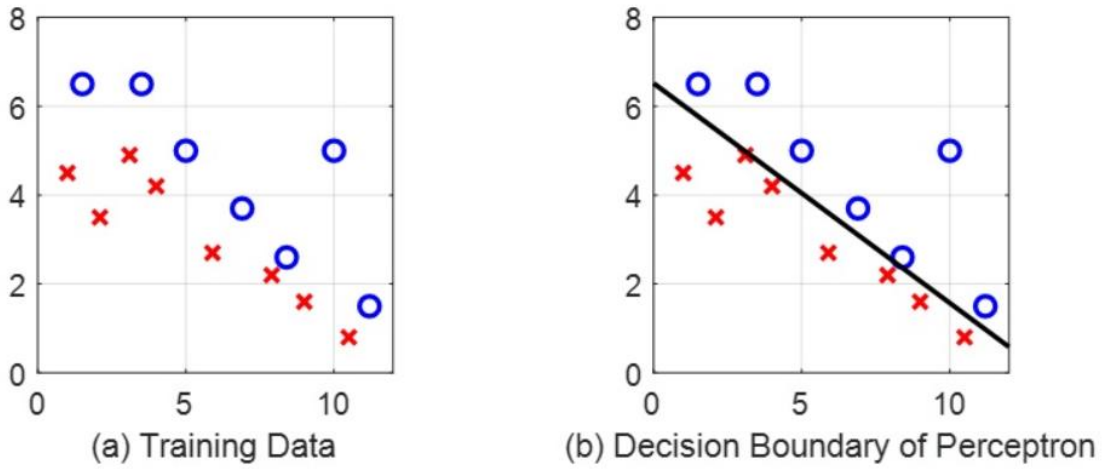
- k_n 近邻分类准则: 给定数据点$\boldsymbol{x}$，找到包含k_n个点的超球体，将$\boldsymbol{x}$分到第i类，如果$k_i = \max\{k_j\}$，其后验概率的估计为

$$P(\omega_i|\boldsymbol{x}) = \frac{p_n(\boldsymbol{x}|\omega_i)P(\omega_i)}{p_n(\boldsymbol{x})} = \frac{k_i}{k_n}$$

- 当$k_n = 1$，为最近邻分类准则

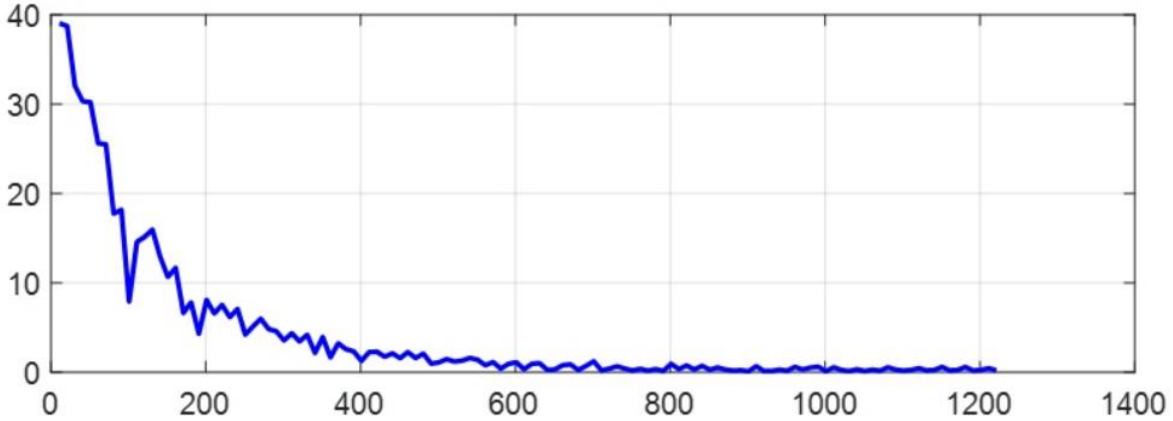
k_n 近邻分类器例子

- 线性感知机中我们的实验



(a) Training Data

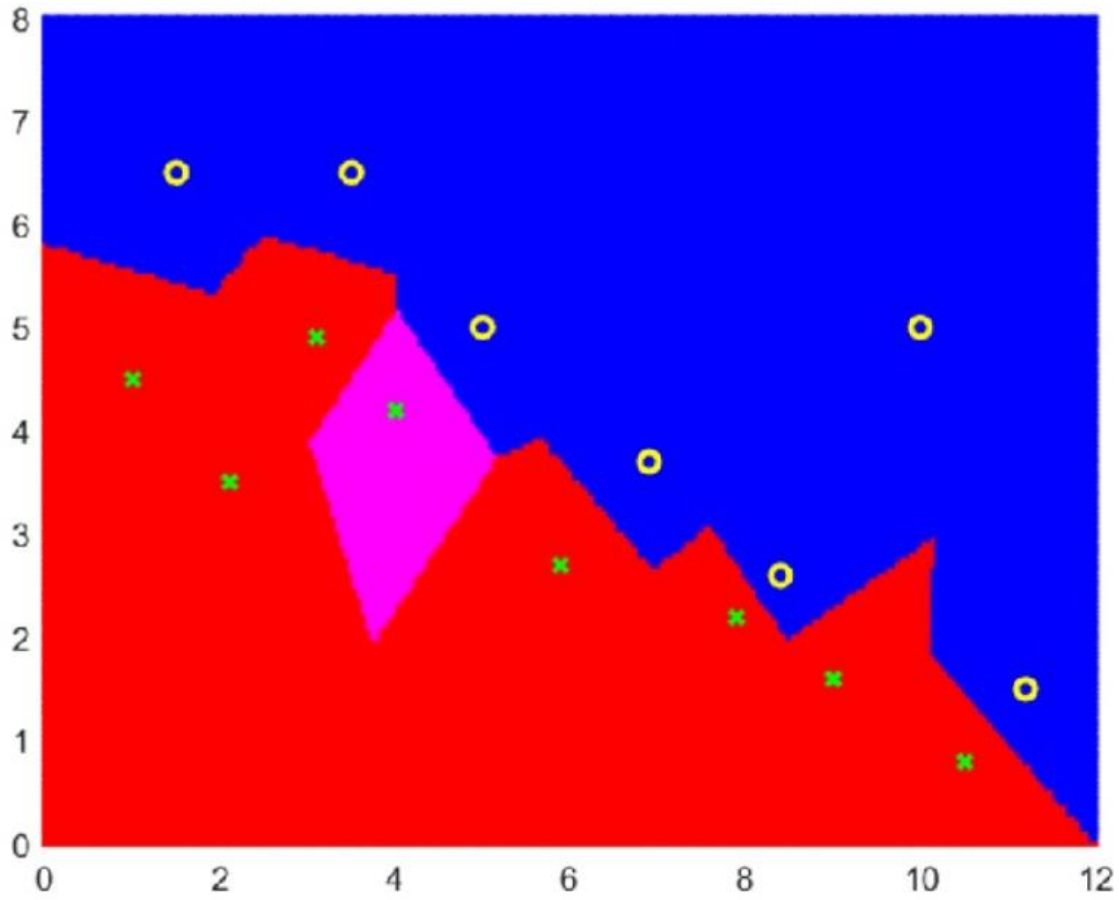
(b) Decision Boundary of Perceptron



(c) Objective function

k_n 近邻分类器例子

- 最近邻分类器的分类边界



*KNN分类器错误概率

- 对样本数为n的数据集，令KNN分类器的条件错误概率为$P_n(e|\boldsymbol{x})$，渐进平均错误概率为

$$P(e) = \lim_{n\to\infty} \int P_n(e|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}$$

- 针对1NN分类器 （c为类别数）

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

*KNN分类器错误概率

- Denote input by $\mathbf{x}$ and its NN by $\mathbf{x}'_n$ (for a finite set)
- Assumption

$$\lim_{n\to\infty} p(\mathbf{x}'_n|\mathbf{x}) = \delta(\mathbf{x}, \mathbf{x}'_n)$$

- The error needs to be conditioned on $\mathbf{x}$ and $\mathbf{x}'_n$

$$P_n(e|\mathbf{x}, \mathbf{x}'_n) = 1 - \sum_{i=1}^{c} P(\omega_i|\mathbf{x})P(\omega_i|\mathbf{x}'_n)$$

- The conditional error

$$\lim_{n\to\infty} P_n(e|\mathbf{x}) = 1 - \sum_{i=1}^{c} P^2(\omega_i|\mathbf{x})$$

- The asymptotic overall error

$$P(e) = \int \left[1 - \sum_{i=1}^{c} P^2(\omega_i|\mathbf{x})\right] p(\mathbf{x})d\mathbf{x}$$

*KNN分类器错误概率

- It is clear that the lower bound of $P(e)$ is $P^*(e)$
- Cover&Hart gave a tight upper bound (1967)
- It is easy to see that $\sum_i P^2(\omega_i|\mathbf{x})$ is minimized if

$$P(\omega_i|\mathbf{x}) = \begin{cases} \frac{P^*(e|\mathbf{x})}{c-1} & i \neq m \\ 1 - P^*(e|\mathbf{x}) & i = m \end{cases}$$

$$1 - \sum_{i=1}^{c} P^2(\omega_i|\mathbf{x}) \leq 2P^*(e|\mathbf{x}) - \frac{c}{c-1} P^{*2}(e|\mathbf{x})$$

- A clever trick

$$Var[P^*(e|\mathbf{x})] = \int [P^*(e|\mathbf{x}) - P^*]^2 p(\mathbf{x}) d\mathbf{x} = \int P^{*2}(e|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} - P^{*2} \geq 0$$

- Then we have

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

K-Nearest Neighbors 算法一般流程

- 存在一个样本数据集合，也称作训练样本集，并且样本集中每个数据都存在标签，即我们知道样本集中每个数据与所属分类的对应关系。
- 输入没有标签的新数据后，将新数据的每个特征与样本集中数据对应的特征进行比较，然后算法提取样本集中特征最相似数据（最近邻）的分类标签。
- 一般来说，只选择样本数据集中前N个最相似的数据。K一般不大于20，最后，选择k个中出现次数最多的分类，作为新数据的分类

K-Nearest Neighbors 算法一般流程

- 收集数据：可以使用任何方法
- 准备数据：距离计算所需要的数值，最后是结构化的数据格式
- 分析数据：可以使用任何方法
- ~~训练算法：　　（此步骤kNN）中不适用~~
- 测试算法：计算错误率
- 使用算法：首先需要输入样本数据和结构化的输出结果，然后运行K-近邻算法判定输入数据分别属于哪个分类，最后应用对计算出的分类执行后续的处理。

K-Nearest Neighbors 算法距离度量

距离度量

$$x_i = (x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(n)})^{\mathrm{T}}$$

- Lp距离：

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

- 欧式距离：

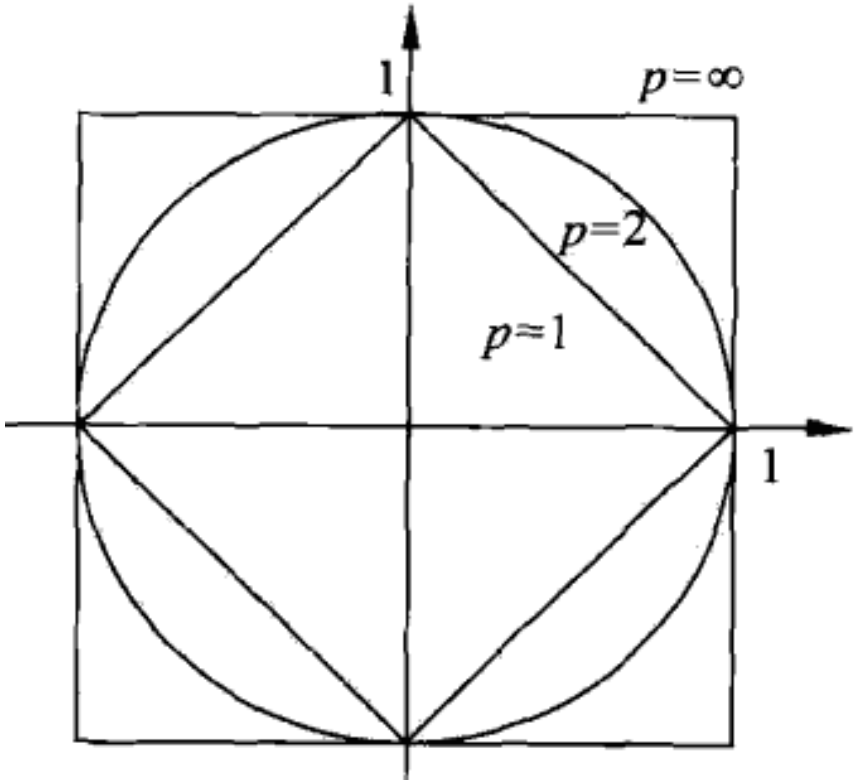
$$L_2(x_i, x_j) = \left(\sum_{l=1}^{n} | x_i^{(l)} - x_j^{(l)} |^2 \right)^{\frac{1}{2}}$$

- 曼哈顿距离

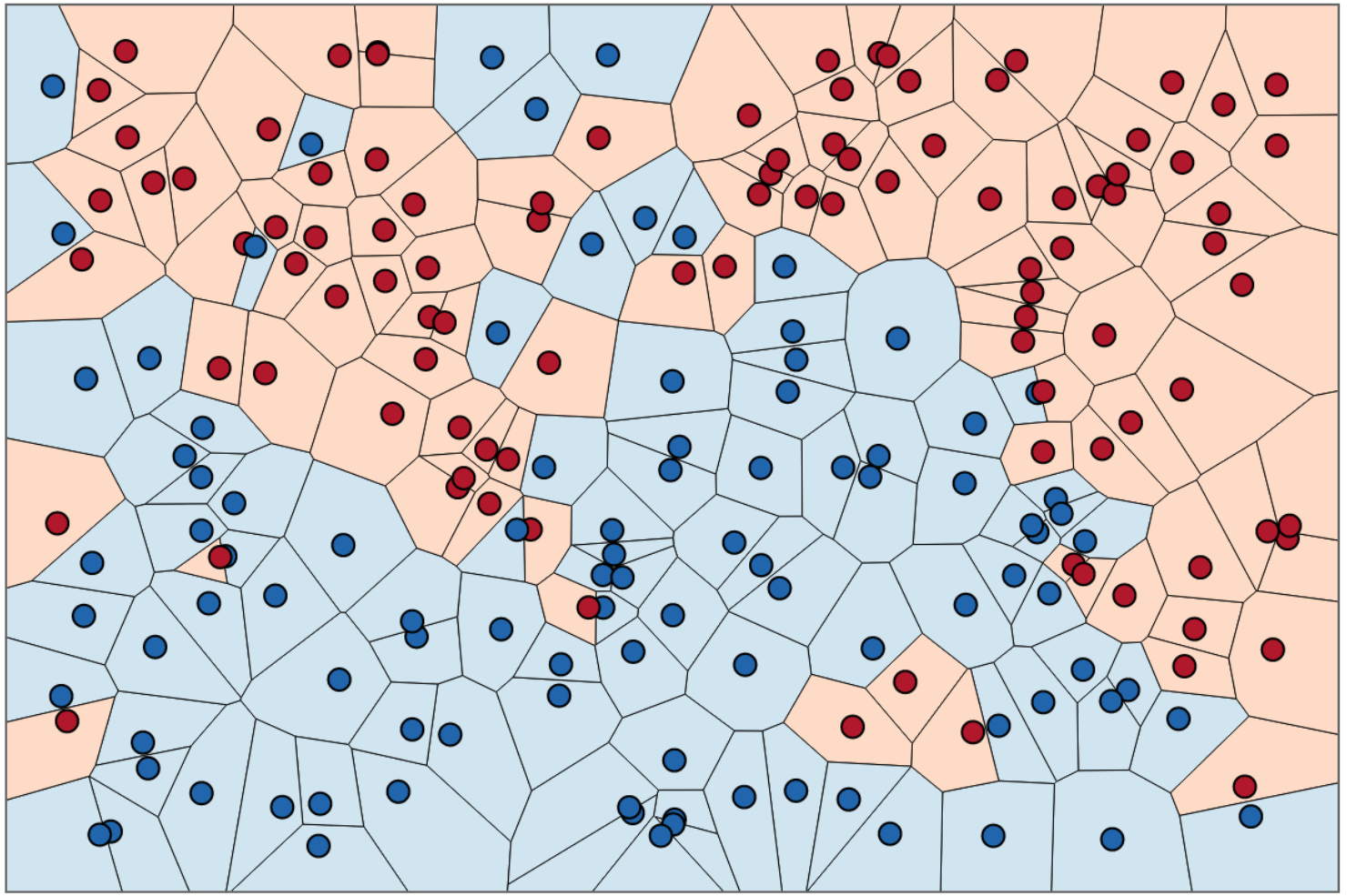
$$L_1(x_i, x_j) = \sum_{l=1}^{n} | x_i^{(l)} - x_j^{(l)} |$$

- L∞距离

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



K-Nearest Neighbors 的判别函数

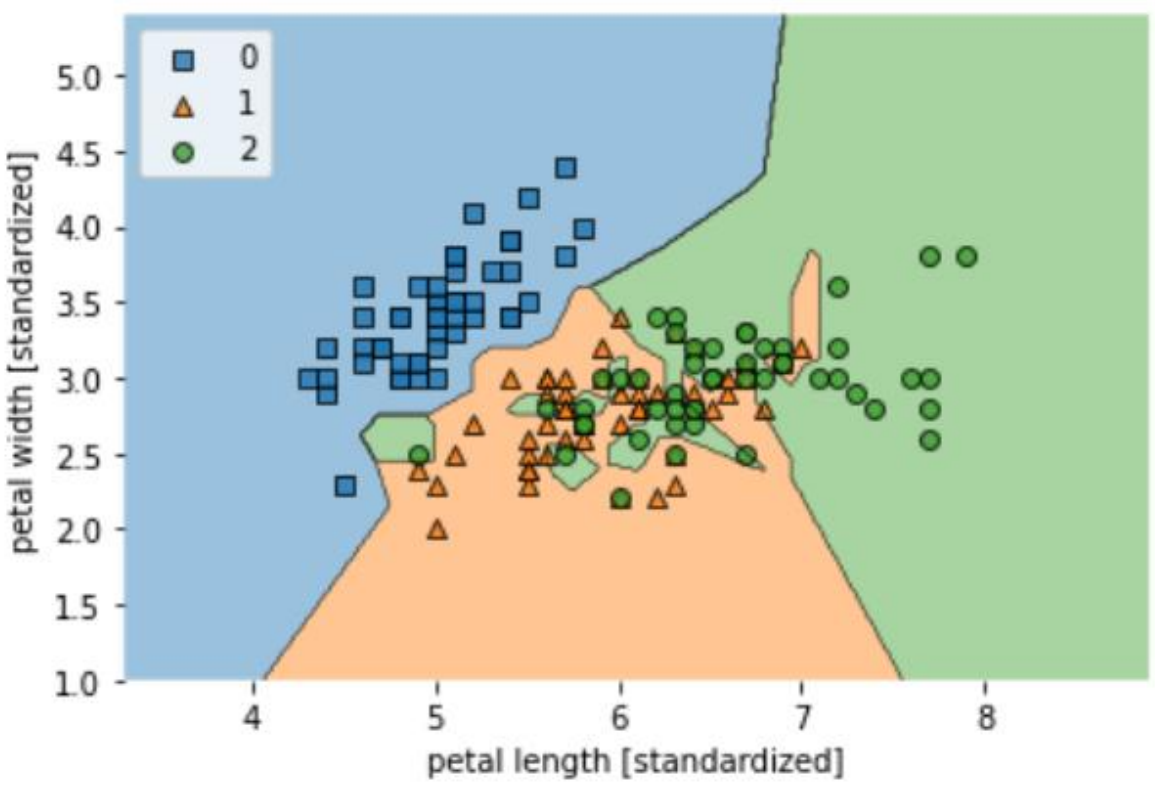


K-Nearest Neighbors 算法K值的选择

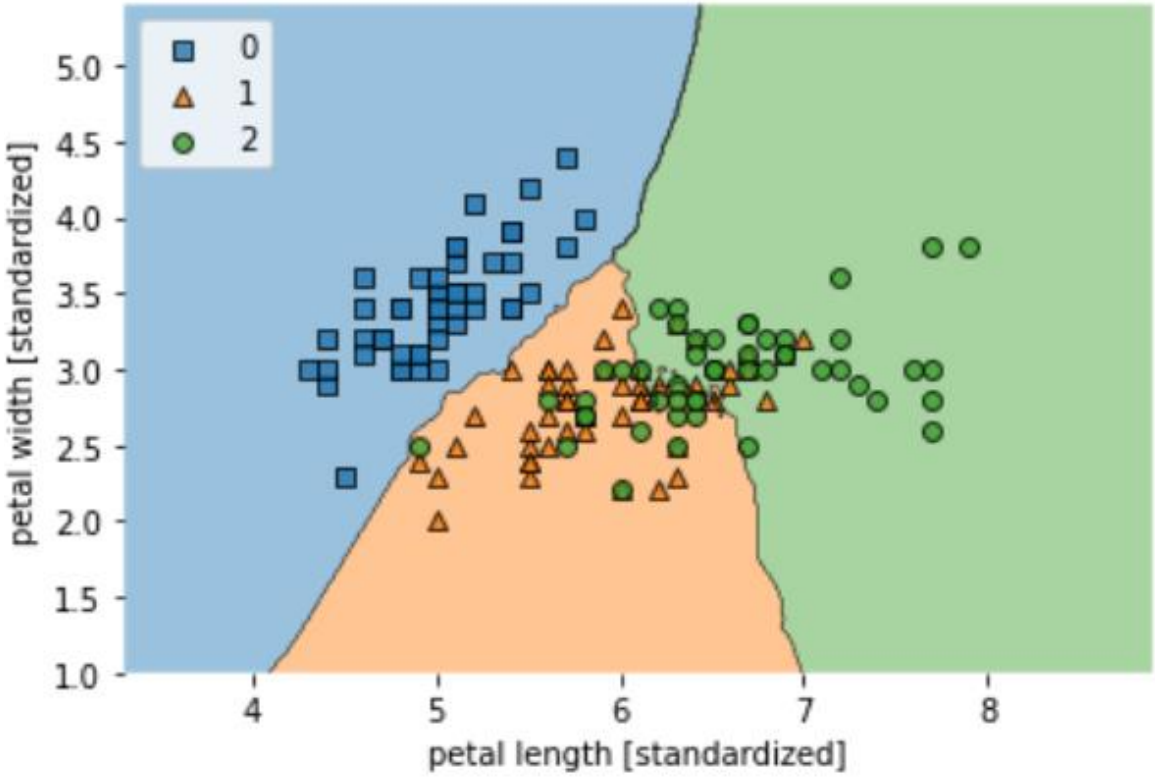
- 如果选择较小的K值
  - ✓ 噪声敏感
  - ✓ K值的减小就意味着整体模型变得复杂，容易发生过拟合

- 如果选择较大的K值
  - ✓ K值的增大 就意味着整体的模型变得简单

K-Nearest Neighbors 算法K值的选择



K=1



K=20

*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

*KD树

- 构造kd树

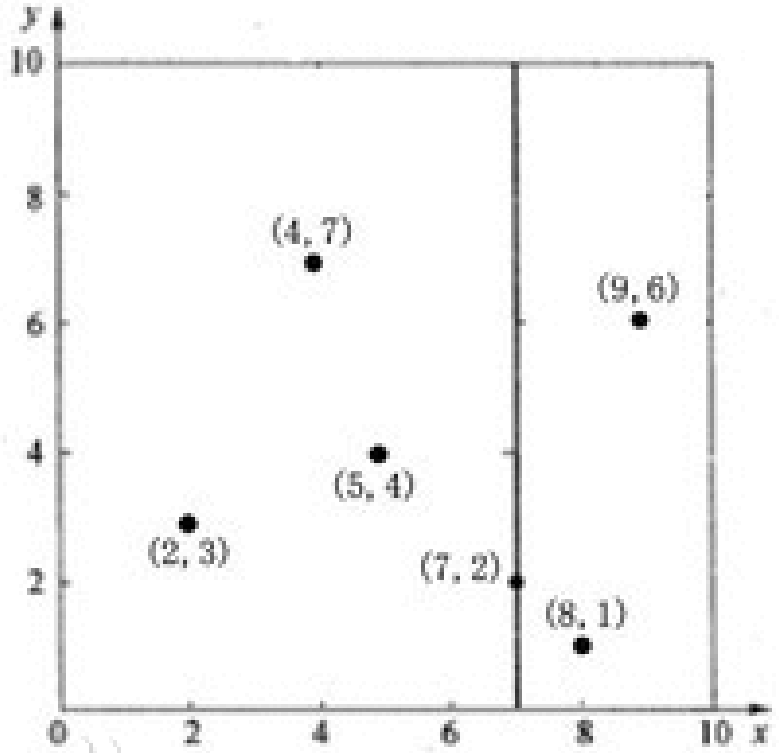


图2 x=7将整个空间分为两部分

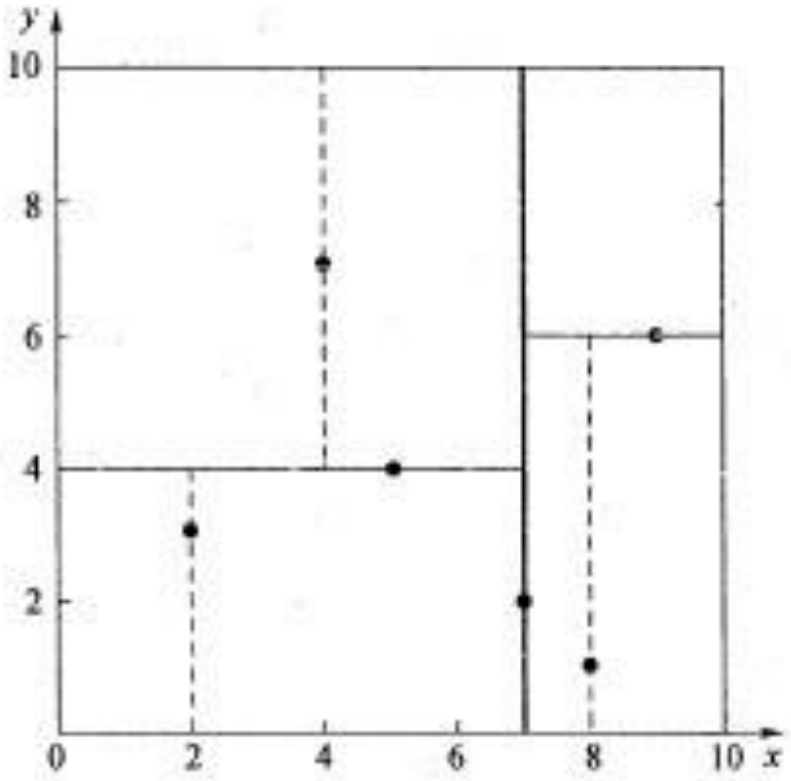
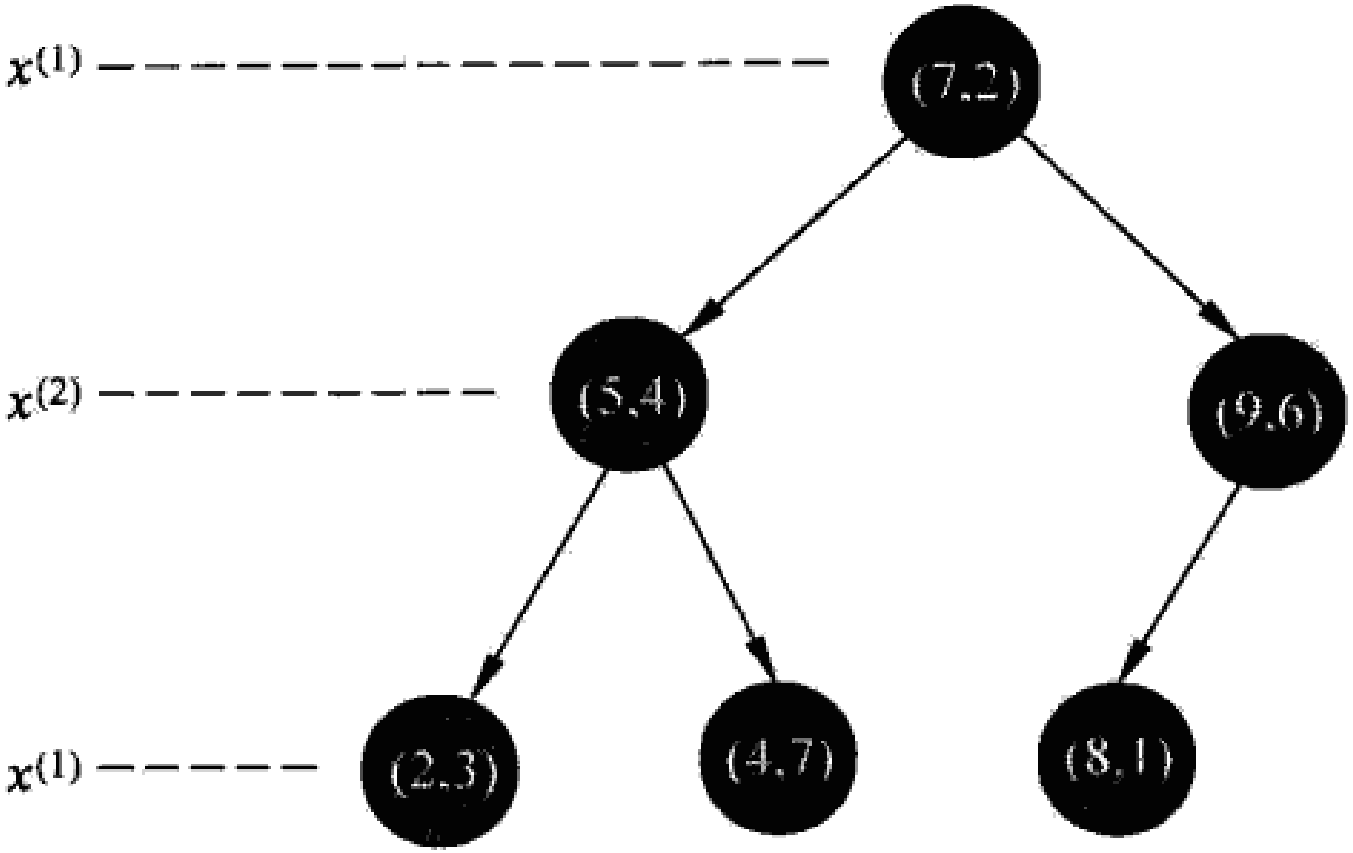


图1 二维数据k-d树空间划分示意图

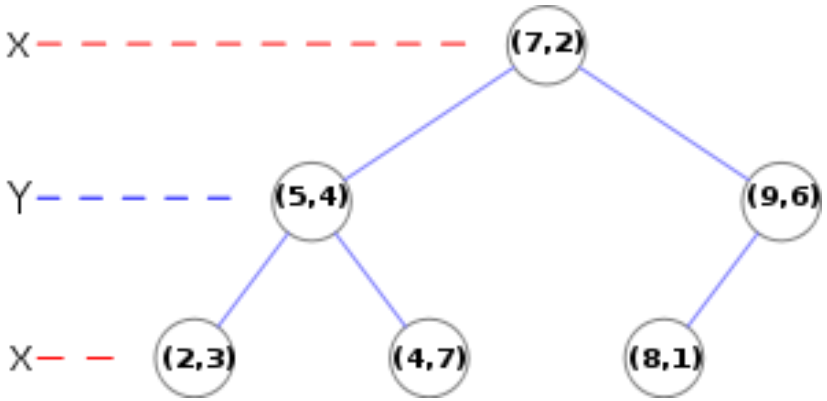
*KD树

- 建立索引



*KD树

- kd树搜索



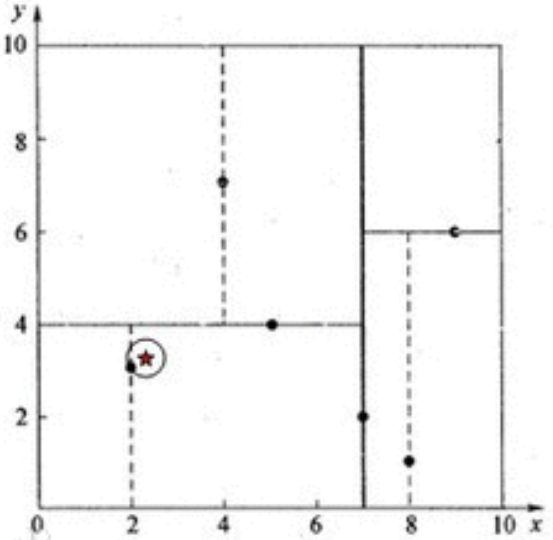


图4 查找（2.1，3.1）点的两次回溯判断

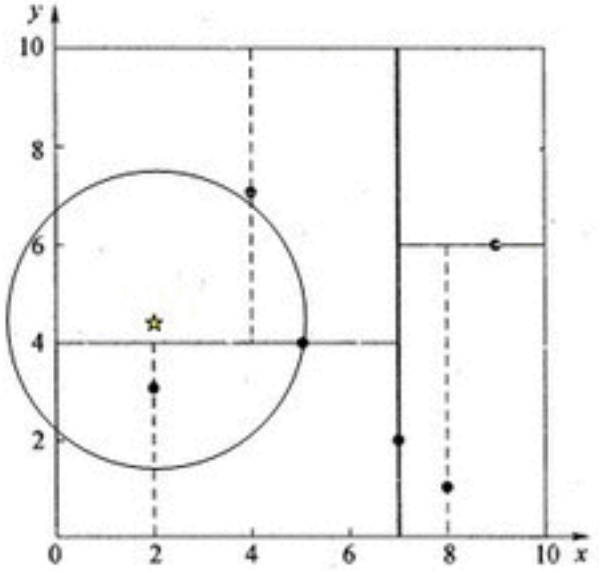


图5 查找（2，4.5）点的第一次回溯判断

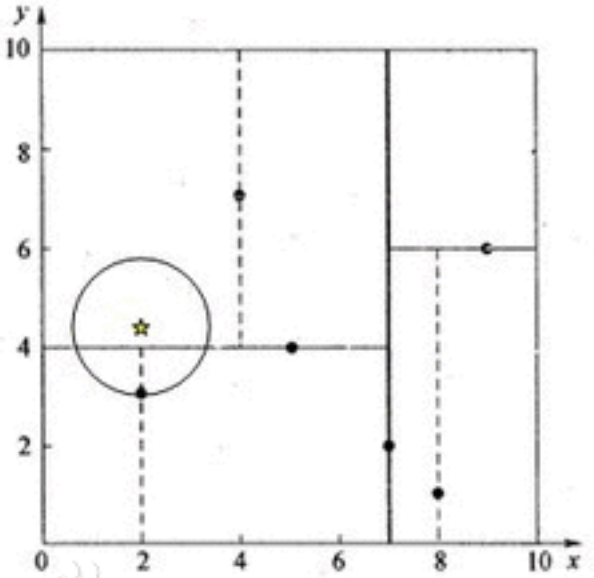
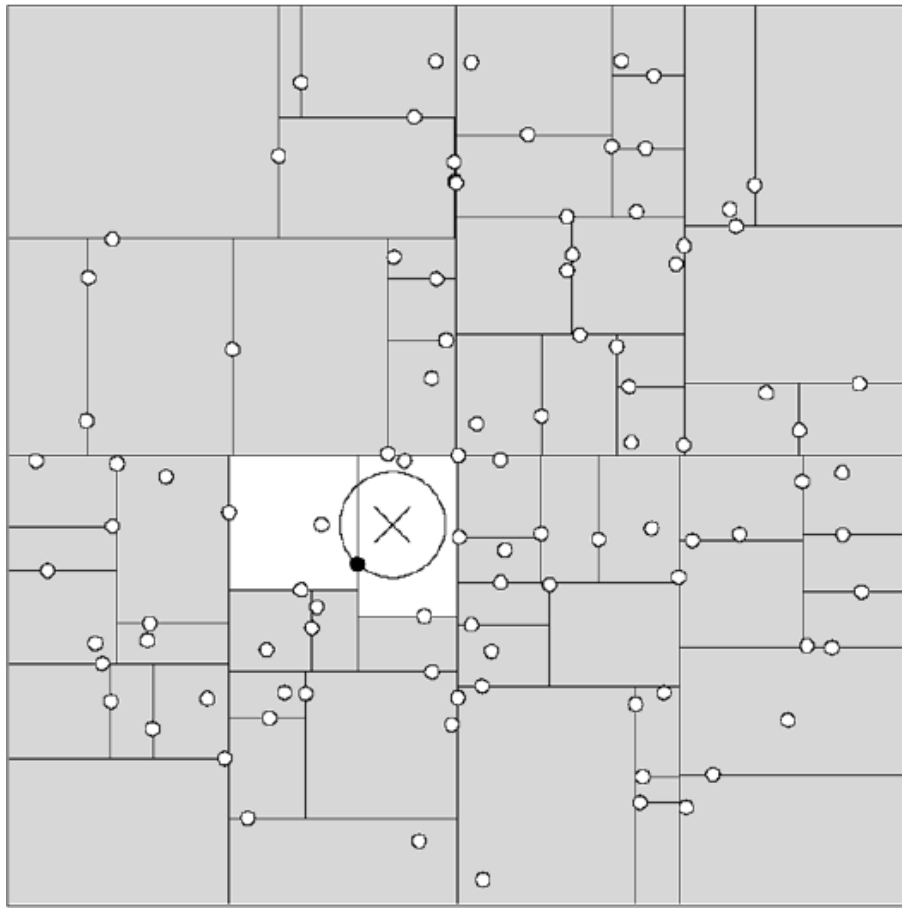
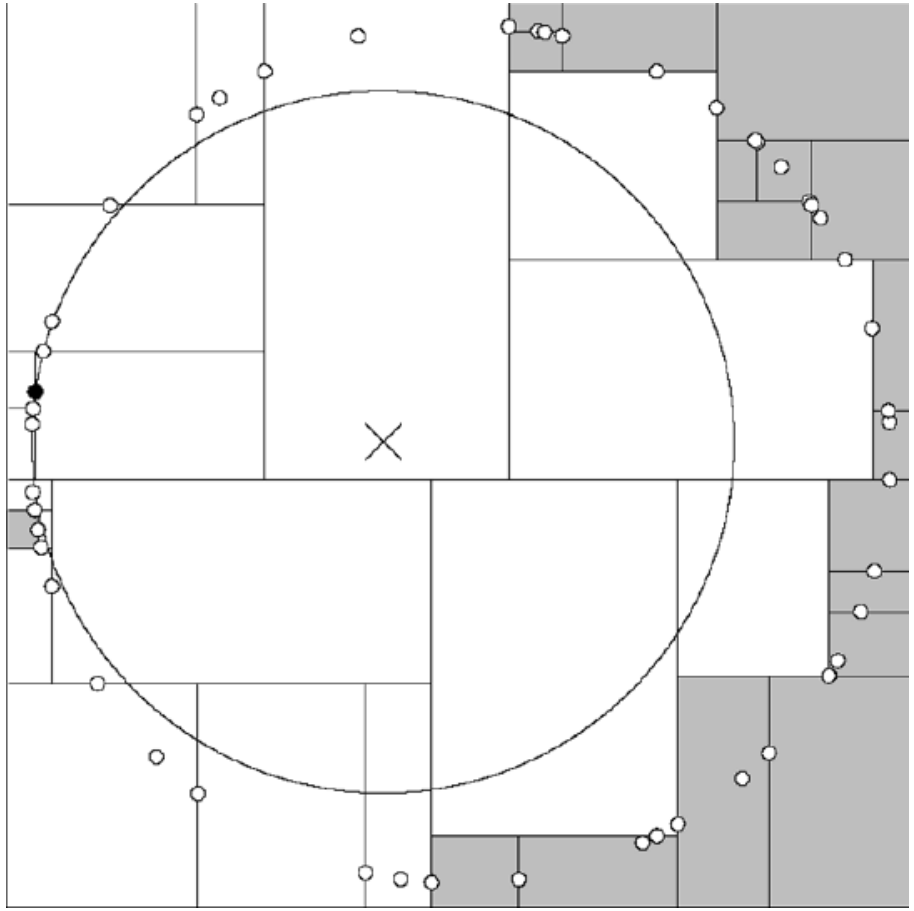


图6 查找（2，4.5）点的第二次回溯判断

*KD 树的缺点





程序函数

class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs)

参数说明:

n_neighbors： knn算法中指定以最近的几个最近邻样本具有投票权，默认参数为5

algrithm：即内部采用什么算法实现。有以下几种选择参数:

'ball_tree':球树、 'kd_tree':kd树、 'brute':暴力搜索、

'auto':自动根据数据的类型和结构选择合适的算法。默认情况下是‘auto’。

KD 树

- 暴力搜索
- KD树是对依次对K维坐标轴，以中值切分构造的树,每一个节点是一个超矩形，在维数小于20时效率高
- ball tree 是为了克服KD树高维失效而发明的，其构造过程是以质心C和半径r分割样本空间，每一个节点是一个超球体。一般低维数据用kd_tree速度快，用ball_tree相对较慢。超过20维之后的高维数据用kd_tree效果反而不佳，而ball_tree效果要好，具体构造过程及优劣势的理论大家有兴趣可以去具体学习。

谢谢各位同学！