第4章　概率密度函数的非参数估计

刘家锋

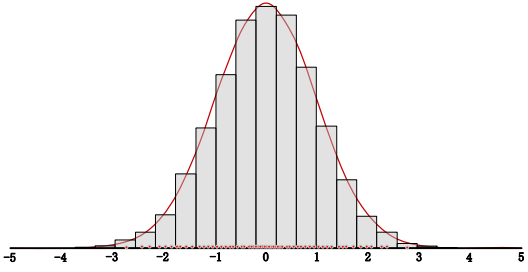
哈尔滨工业大学

第4章　概率密度函数的非参数估计

4.1 非参数估计

非参数估计的基本思路

- **概率密度估计的简单方法**
  - 将特征空间划分为一系列的区域(bin)，统计每个区域内训练样本出现的频率，作为概率密度的粗糙近似



非参数估计的基本思路

- **非参数估计的一般性描述**
  - 令R是包含样本点$\mathbf{x}$的一个区域，体积为V；n个训练样本中有k个落在区域R中，可对$\mathbf{x}$的概率密度作出估计

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

  - 此估计以区域R内的平均概率密度近似$\mathbf{x}$点的密度
  - 如果假设区域R内每一点的概率密度均为$p(\mathbf{x})$，则新的采样点$\mathbf{x}'$落在R内的概率为

$$P(\mathbf{x}' \in R) = \int_R p(\mathbf{x})d\mathbf{x} = p(\mathbf{x}) \int_R d\mathbf{x} = Vp(\mathbf{x}) \approx k/n$$

估计的有效性

- **给定样本数n**
  - 区域R的体积V影响估计的效果
  - 区域过大，每一点的概率密度值差异大，以平均值近似$p(\mathbf{x})$不够精确
  - 区域过小，可能导致区域内的采样点少，甚至$k = 0$
- **有效性**
  - 非参数估计的有效性取决于样本数n，以及与其相适应的区域体积V

估计的收敛性

- **非参数估计的收敛性**
  - 构造一系列包含$\mathbf{x}$的区域$R_1, R_2, \cdots$，对应$n = 1, 2, \cdots$，对$p(\mathbf{x})$有一系列的估计

$$p_n(\mathbf{x}) = \frac{k_n/n}{V_n}, \quad n = 1, 2, \cdots$$

  - 当满足下列条件时，$\lim_{n\to\infty} p_n(\mathbf{x}) = p(\mathbf{x})$

$$\lim_{n\to\infty} V_n = 0$$
$$\lim_{n\to\infty} k_n = \infty$$
$$\lim_{n\to\infty} \frac{k_n}{n} = 0$$

区域选定的两个途径

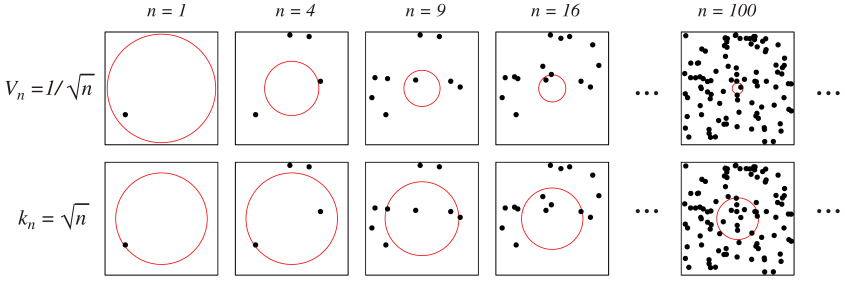
- **Parzen窗法**
  - 区域体积V是样本数n的函数，例如

$$V_n = \frac{1}{n}$$

- **K-近邻法**
  - 落在区域R内的样本数k是总样本数n的函数，例如

$$k_n = \sqrt{n}$$

Parzen窗法和K-近邻法

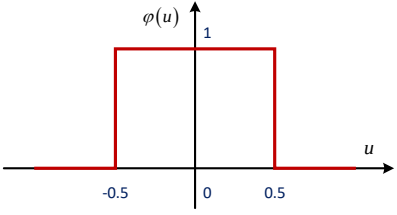


4.2 Parzen窗方法

窗函数

- **窗函数**
  - Parzen窗法使用窗函数来划定区域R
  - 超立方体窗函数

$$\varphi(\mathbf{u}) = \begin{cases} 1, & |u_i| \le \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}$$



窗函数

- **窗函数的位置和尺度**
  - 以$\mathbf{x}$为中心，边长为h_n的窗函数

$$\varphi\left(\frac{\mathbf{u}-\mathbf{x}}{h_n}\right)=\begin{cases}1, & |u_i-x_i|\le\frac{h_n}{2}\\ 0, & \text{otherwise}\end{cases}$$

  - 窗函数（超立方体）的体积

$$V_n=h_n^d$$

- **判断样本$\mathbf{x}_i$是否在超立方体R之内**

$$\varphi\left(\frac{\mathbf{x}_i-\mathbf{x}}{h_n}\right)=\begin{cases}1, & \mathbf{x}_i\in R\\ 0, & \mathbf{x}_i\notin R\end{cases}$$

窗函数

- **超立方体内的样本数**

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

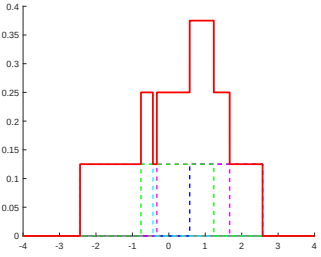
窗函数是中心对称的，因此可以互换$\mathbf{x}$和$\mathbf{x}_i$的位置

- **概率密度函数的估计**

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

窗函数

- **对Parzen窗的理解**
  - 概率密度的估计是n个窗函数的叠加平均
  - 也可以理解为是n个以训练样本为中心的窗函数，对概率密度函数的插值逼近



窗函数的选择

- **更多的窗函数**
  - Parzen窗估计概率密度，距离$\mathbf{x}$近的样本$\mathbf{x}_i$对估计结果的贡献大，距离远的贡献小
  - 超立方体窗函数是不连续的，估计结果是用“阶梯”函数来逼近概率密度函数
  - 使用平滑的窗函数可以得到平滑的概率密度函数
- **窗函数的条件**
  - 满足如下条件的函数，均可以作为窗函数

$$\varphi(\mathbf{u}) \geq 0, \quad \int \varphi(\mathbf{u}) d\mathbf{u} = 1$$

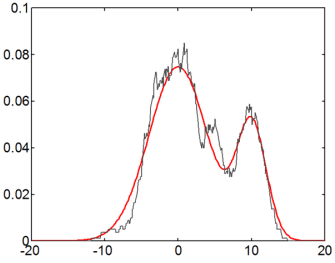
  - 最常使用的窗函数是Gauss函数

$$\phi(\mathbf{u}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\|\mathbf{u}\|^2}{2\sigma^2}}$$

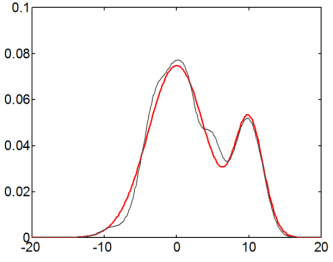
σ^2为窗函数的宽度

窗函数的选择

- **方形窗和高斯窗函数**



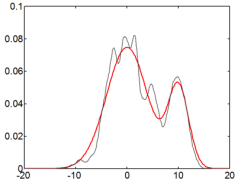
方形窗函数



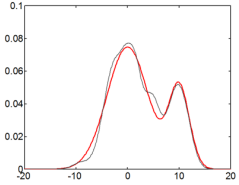
高斯窗函数

窗函数的选择

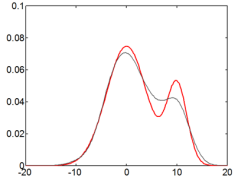
- 不同的窗函数宽度



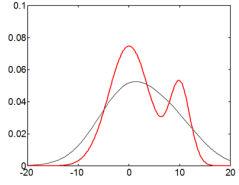
$h_n = 0.5$



$h_n = 1$



$h_n = 2$



$h_n = 5$

Parzen窗识别

Algorithm 1 Parzen窗识别

Input: 训练集D，窗函数$\varphi(\mathbf{x})$，窗函数宽度h

Output: 待识别模式$\mathbf{x}$所属类别

1: 保存每个类别的训练样本集$D_i = \{\mathbf{x}_1^i, \cdots, \mathbf{x}_{n_i}^i\}$

2: **for** $i = 1$ to c **do**

3: 计算ω_i类的概率密度函数值：

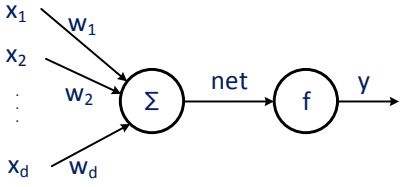
$$p(\mathbf{x}|\omega_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{1}{V_n} \varphi\left(\frac{\mathbf{x} - \mathbf{x}_j^i}{h}\right)$$

4: **end for**

5: 最小错误率贝叶斯判别

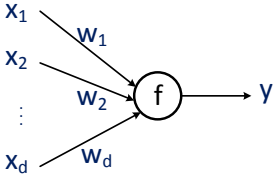
概率神经网络

- **神经元模型**
  - 1943年McCulloch和Pitt模仿生物的神经系统提出了人工神经元模型，是现代人工神经网络的基础
  - 很多模式识别方法都可以采用神经网络的方式实现，包括Parzen窗法



概率神经网络

- **简化的神经元模型**
  - 输入：$\mathbf{x} = (x_1, \cdots, x_d)^t$
  - 权值：$\mathbf{w} = (w_1, \cdots, w_d)^t$
  - 净输入：$net = \mathbf{w}^t\mathbf{x}$
  - 输出：$y = f(net) = f(\mathbf{w}^t\mathbf{x})$，$f$称为激活函数



概率神经网络

- **Gauss窗函数的神经元表示**
  - 输入样本的长度归一化

$$\mathbf{x} \leftarrow \frac{\mathbf{x}}{\|\mathbf{x}\|}, \quad \mathbf{x}_i \leftarrow \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}$$

  - 设置神经元的权值为训练样本

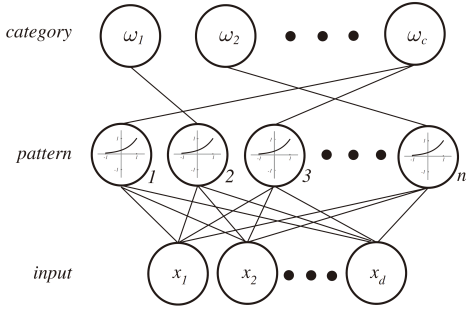
$$\mathbf{w}_i = \mathbf{x}_i, \quad \|\mathbf{w}_i\|^2 = \mathbf{w}_i^t \mathbf{w}_i = 1$$

  - 窗函数可以用神经元实现

$$\begin{aligned} \varphi\left(\frac{\mathbf{x}-\mathbf{w}_i}{\sigma}\right) &\propto \exp\left[-\frac{(\mathbf{x}-\mathbf{w}_i)^t(\mathbf{x}-\mathbf{w}_i)}{2\sigma^2}\right] \\ &= \exp\left(-\frac{\mathbf{x}^t\mathbf{x}+\mathbf{w}_i^t\mathbf{w}_i-2\mathbf{w}_i^t\mathbf{x}}{2\sigma^2}\right) \\ &= \exp\left(\frac{net_i-1}{\sigma^2}\right) = f(net_i) \end{aligned}$$

概率神经网络

- **Probabilistic Neural Networks(PNN)**



PNN的学习算法

Algorithm 2 PNN Training

Input: 属于c个类别的训练集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$

Output: 模式层权值$\{\mathbf{w}_1, \cdots, \mathbf{w}_n\}$，类别层权值$\{\mathbf{a}_1, \cdots, \mathbf{a}_c\}$

1: 初始化：$\mathbf{a}_j = \mathbf{0}, j = 1, \cdots, c$

2: **for** $i = 1$ to n **do**

3: 归一化样本长度：$\mathbf{x}_i \leftarrow \mathbf{x}_i / \|\mathbf{x}_i\|$

4: 学习模式层权值：$\mathbf{w}_i = \mathbf{x}_i$

5: **if** $\mathbf{x}_i \in \omega_j$ **then**

6: 学习类别层权值：$a_{ji} \leftarrow 1$

7: **end if**

8: **end for**

PNN的分类算法

Algorithm 3 PNN Classification

Input: PNN的权值$\{\mathbf{w}_1, \cdots, \mathbf{w}_n\}$，$\{\mathbf{a}_1, \cdots, \mathbf{a}_c\}$

Output: 待识别模式$\mathbf{x}$所属类别

归一化样本长度：$\mathbf{x} \leftarrow \mathbf{x}/\|\mathbf{x}\|$

初始化网络输出：$y_j = 0, j = 1, \cdots, c$

for $i = 1$ to n **do**

    计算模式层净输入：$net_i = \mathbf{w}_i^t \mathbf{x}$

    if $a_{ji} = 1$ **then**

        累加类别层输出：$y_j \leftarrow y_j + \exp\left(\frac{net_i - 1}{\sigma^2}\right)$

    end if

end for

return $\arg\max_{1 \le j \le c} [y_j]$

4.3 近邻分类器

近邻分类器

- **最小错误率分类的3个途径**
  - 条件概率判别：$\max_i p(\mathbf{x}|\omega_i)P(\omega_i)$
  - 联合概率判别：$\max_i p(\mathbf{x}, \omega_i)$
  - 后验概率判别：$\max_i p(\omega_i|\mathbf{x})$
- **Parzen窗和近邻分类**
  - Parzen窗法估计的是每个类别的条件概率$p(\mathbf{x}|\omega_i)$
  - 近邻法估计的是每个类别的后验概率$p(\omega_i|\mathbf{x})$

近邻分类器

- **后验概率的估计**
  - 构造一个以待识别模式$\mathbf{x}$为中心的区域R，体积为V
  - 所有类别的n个训练样本中有k个处于区域R内，其中k_i个样本属于ω_i类
  - ω_i类联合概率的估计

$$p(\mathbf{x}, \omega_i) \approx \frac{k_i/n}{V}$$

  - ω_i类后验概率的估计

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}, \omega_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}, \omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}, \omega_j)} \approx \frac{k_i}{\sum_{j=1}^{c} k_j} = \frac{k_i}{k}$$

K-近邻分类算法

Algorithm 4 KNN：k-Nearest Neighbor

Input: 训练集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，参数$k$

Output: 待识别模式$\mathbf{x}$所属类别

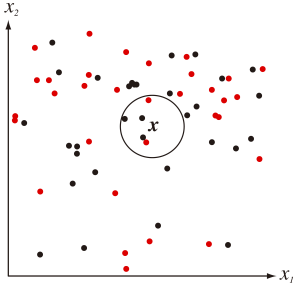
1: 计算待识别模式与每个训练样本的距离：$D(\mathbf{x}, \mathbf{x}_i)$

2: 选择距离最小的前k个样本，统计其中包含各个类别的样本数k_j

3: **return** $\arg\max_{1 \le j \le c} [k_j]$

K-近邻分类算法

$k = 5$的近邻分类



距离度量

- **距离度量**
  - 距离是数学上的一个重要概念
  - 满足如下条件的函数均可称为距离
    1. 非负性：$D(\mathbf{x}, \mathbf{y}) \geq 0$
    2. 自反性：$D(\mathbf{x}, \mathbf{y}) = 0$ 当且仅当$\mathbf{x} = \mathbf{y}$
    3. 对称性：$D(\mathbf{x}, \mathbf{y}) = D(\mathbf{y}, \mathbf{x})$
    4. 三角不等式：$D(\mathbf{x}, \mathbf{z}) + D(\mathbf{y}, \mathbf{z}) \geq D(\mathbf{x}, \mathbf{y})$
  - 具体应用中，需要根据情况选择合适的距离度量

常用的距离度量

- **Eucidean距离**：对应矢量的l_2范数

$$D(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2 = \left[\sum_{i=1}^{d}(x_i - y_i)^2\right]^{\frac{1}{2}}$$

- **Manhattan距离**：对应矢量的l_1范数

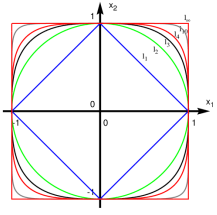
$$D(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_1 = \sum_{i=1}^{d}|x_i - y_i|$$

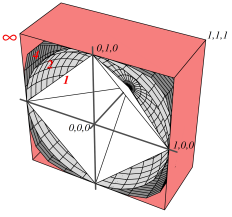
常用的距离度量

- **Minkowski距离**：对应矢量的l_p范数

$$D(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_p = \left[\sum_{i=1}^{d}|x_i - y_i|^p\right]^{\frac{1}{p}}$$

- **等距面**：不同p值对应的“单位球面”





常用的距离度量

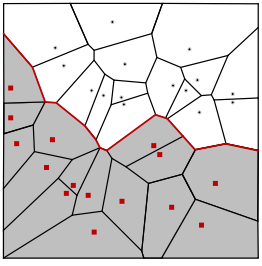
- **相似度**
  - 在某些应用以相似度来代替距离，距离越小相似度越大
  - 相似度的定义比距离函数灵活、宽松
- **角度相似性**

$$D(\mathbf{x},\mathbf{y}) = \cos\theta = \frac{\mathbf{x}^t\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$$

其中，θ是矢量$\mathbf{x}$和$\mathbf{y}$之间的夹角

最近邻分类

- **最近邻分类**
  - 分类规则：在训练样本集中寻找与待识别样本$\mathbf{x}$距离最近的样本$\mathbf{x}'$，将$\mathbf{x}$分类到$\mathbf{x}'$所属的类别
  - 最近邻分类相当于$k=1$的k-近邻分类，其分类界面可以用Voronoi网格表示
  - 每个网格包含一个训练样本，网格区域内的点以该样本为最近邻



最近邻分类的简化

- **最近邻分类的错误率**
  - 可以证明，当训练样本数$n \to \infty$时，最近邻分类错误率P与贝叶斯分类错误率P^*之间的关系满足

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

  - 一般来说，训练样本数n越大，最近邻分类的性能越好
- **最近邻分类的计算复杂度**
  - 时间复杂度和空间复杂度均为：$O(dn)$，d为特征维数
  - 最近邻分类的简化：部分距离法，预分类法，剪辑近邻法

例4.1 KNN算法分类MNIST数据

- **数据集**
  - 使用实验2的17维MNIST数据
  - 类别：10个手写数字
  - 训练集：30000，测试集：10000
- **分类器**
  - 设置参数$k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$
  - 测试分类器性能

读取数据

```
import numpy as np
import pandas as pd

data = pd.read_csv("MNIST-Train-Samples.csv", header=None)
X_train = data.to_numpy()
data = pd.read_csv("MNIST-Train-Labels.csv", header=None)
y_train = data.to_numpy().ravel()

data = pd.read_csv("MNIST-Test-Samples.csv", header=None)
X_test = data.to_numpy()
data = pd.read_csv("MNIST-Test-Labels.csv", header=None)
y_test = data.to_numpy().ravel()

print("Train samples:", X_train.shape, y_train.shape)
print("Test samples:", X_test.shape, y_test.shape)
```

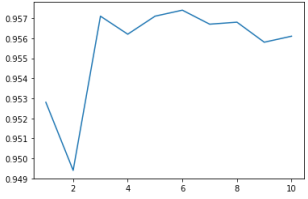
KNN算法分类MNIST数据

```
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt

Score = np.zeros(10)
for k in range(1,11):
    knn = KNeighborsClassifier(n_neighbors=k).fit(X_train, y_train)
    Score[k-1] = knn.score(X_test,y_test)
    print("k=%d: %f" %(k,Score[k-1]))

plt.plot(np.linspace(1,10,10),Score)
plt.show()
```

```
k=1: 0.952800
k=2: 0.949400
k=3: 0.957100
k=4: 0.956200
k=5: 0.957100
k=6: 0.957400
k=7: 0.956700
k=8: 0.956800
k=9: 0.955800
k=10: 0.956100
```



最近邻分类的简化

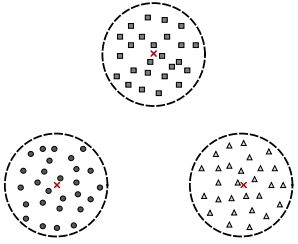
- **部分距离法**
  - 定义部分距离

$$D_r(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{r}(x_i - y_i)^2\right)^{\frac{1}{2}}$$

  - $D_r(\mathbf{x},\mathbf{y})$是$r$的单调不减函数
  - 令D_{min}为当前搜索到的最近邻距离
  - 如果$D_r(\mathbf{x},\mathbf{x}_i) > D_{min}$，则$D_d(\mathbf{x},\mathbf{x}_i) > D_{min}$
  - $\mathbf{x}_i$不是$\mathbf{x}$的最近邻，无需继续计算$D_d(\mathbf{x},\mathbf{x}_i)$

最近邻分类的简化

- **预分类**
  - 选择m个代表样本点，各代表一部分样本
  - 待识别模式$\mathbf{x}$在代表样本中寻找最近邻，然后在其代表的样本集中寻找实际的最近邻
  - 这种方法是一个次优的搜索算法



最近邻分类的简化

Algorithm 5 最近邻剪辑算法

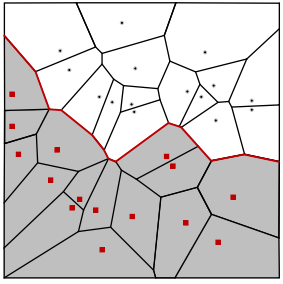
Input: 训练集D

Output: 简化的Voronoi图

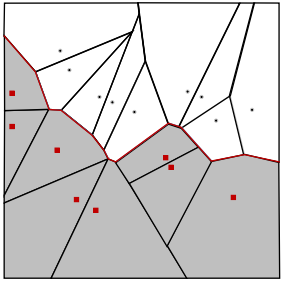
1: 构造训练集D的完整Voronoi图
2: **for** $j = 1$ to n **do**
3: 寻找样本$\mathbf{x}_j$在Voronoi图上的所有邻居
4: 如果邻居中存在与$\mathbf{x}_j$不同类别的样本，则标记$\mathbf{x}_j$
5: **end for**
6: 样本集中删除未被标记的样本
7: 用剩余的样本重新构造Voronoi图

剪辑近邻法

靠近分类边界的样本被保留，远离的样本被剪辑



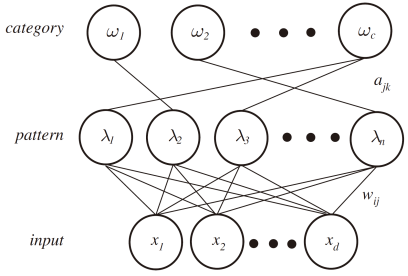
剪辑前



剪辑后

RCE网络

- 最近邻分类也可以采用神经网络实现



RCE网络

RCE网络的学习

Algorithm 6 RCE Training

Input: 训练集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，最大半径$\lambda_m$，距离裕量$\epsilon$

Output: 模式层权值$\{\mathbf{w}_i\}$，半径$\{\lambda_i\}$，类别层权值$\{\mathbf{a}_j\}$

初始化类别层权值：$\mathbf{a}_j = \mathbf{0}, j = 1, \cdots, c$

for $i = 1$ to n **do**

　学习模式层权值：$\mathbf{w}_i = \mathbf{x}_i$

　如果$\mathbf{x}_i \in \omega_j$，则类别层权值$a_{ji} = 1$

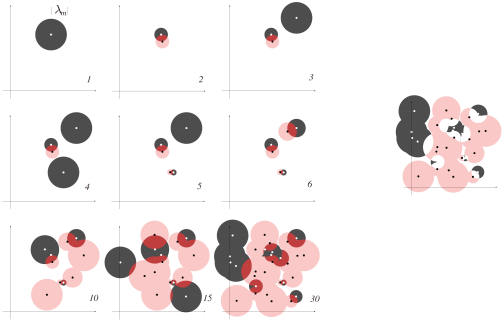
　寻找非ω_j类的最近邻：$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}' \notin \omega_j} D(\mathbf{x}_i, \mathbf{x}')$

　设置半径：$\lambda_i = \min\left[D(\mathbf{x}_i, \hat{\mathbf{x}}) - \epsilon, \lambda_m\right]$

end for

RCE网络

RCE网络对空间的划分，空白区域无法判别



RCE网络的分类

Algorithm 7 RCE Classification

Input: 模式层权值$\{\mathbf{w}_i\}$，半径$\{\lambda_i\}$，类别层权值$\{\mathbf{a}_j\}$

Output: 待识别模式$\mathbf{x}$的类别

1: 初始化类别层输出：$y_j = 0, \quad j = 1, \cdots, c$

2: 计算模式层输出：$z_i = \begin{cases} 1, & D(\mathbf{x}, \mathbf{w}_i) < \lambda_i \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, \cdots, n$

3: 计算类别层输出：$y_j = \mathbf{a}_j^t \mathbf{z} = \sum_{i=1}^{n} a_{ji} z_i, \quad j = 1, \cdots, c$

4: **if** 存在$y_k \neq 0$，而$y_j = 0, \forall j \neq k$ **then**

5: 判别：$\mathbf{x} \in \omega_k$

6: **else**

7: 无法判别

8: **end if**

End