

# 模式识别

---

Pattern Recognition

第2章 距离分离器

## 2.1.1 距离分类器的一般形式

■ 将待识别样本 $\mathbf{x}$ 分类到与其最相似的类别中

- 输入：需要识别的样本 $\mathbf{x}$ ；
- 计算 $\mathbf{x}$ 与所有类别的相似度  $s(\mathbf{x}, \omega_i)$ ,  $i = 1, \dots, c$ ；
- 输出：相似度最大的类别  $\omega_j$

$$j = \arg \max_{1 \leq i \leq c} s(\mathbf{x}, \omega_i)$$


**关键问题**：如何度量样本 $\mathbf{x}$ 与类别  $\omega_i$  的相似程度？

## 2.1.2 模板匹配

- 每个类别的“先验知识”就是一个样本（模板） $\mu_i$
- 利用 $\mathbf{x}$ 与模板 $\mu_i$ 的相似度，作为 $\mathbf{x}$ 与类别 $\omega_i$ 的相似度
- 用“距离”度量样本之间的相似度

$$\omega_i \rightarrow \mu_i$$

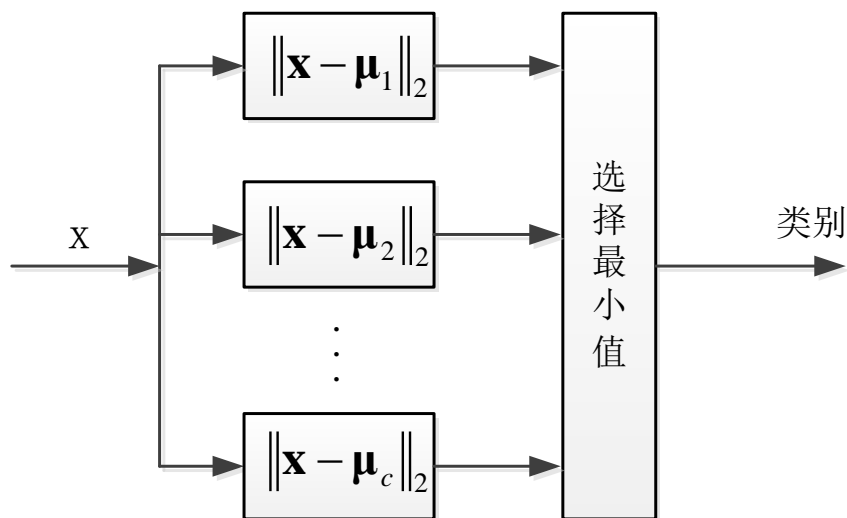
$$s(\mathbf{x}, \omega_i) = s(\mathbf{x}, \mu_i)$$

$$\begin{aligned} s(\mathbf{x}, \mu) &= -d(\mathbf{x}, \mu) \\ &= -\|\mathbf{x} - \mu\|_2 = -\sqrt{\sum_{i=1}^d (x_i - \mu_i)^2} \end{aligned}$$

欧氏距离

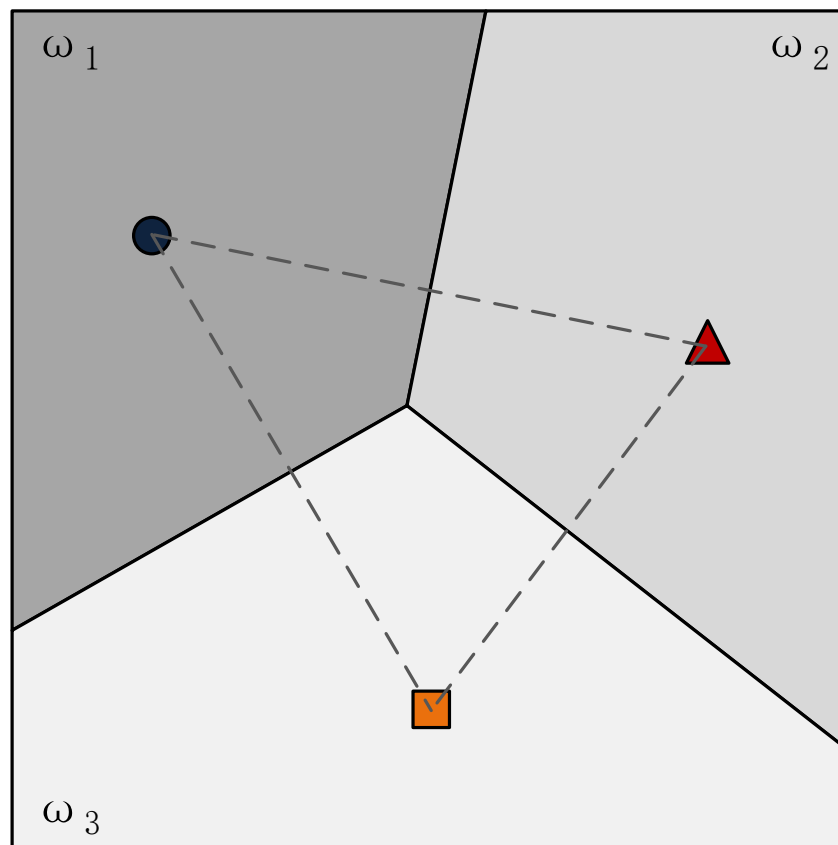
## 2.1.2 模板匹配

- $\|\cdot\|_2$  矢量的“ $l_2$ 范数”——矢量的长度
- 差矢量  $\mathbf{x} - \boldsymbol{\mu}$  的  $l_2$  范数——两个点之间的欧氏距离
- 模板匹配过程：
$$j = \arg \min_{1 \leq i \leq c} d(\mathbf{x}, \boldsymbol{\mu}_i)$$



## 2.1.2 模板匹配——判别界面

- $c$ 个模板将特征空间划分成了 $c$ 个区域
- 两个区域的交界称为“判别界面”



## 2.1.3 最近邻分类器

- 每一个类别有多个训练样本  $D_i = \{\mathbf{x}_1^{(i)}, \dots, \mathbf{x}_{n_i}^{(i)}\}$

类别  $i = 1, \dots, c$ .      $n_i$  : 第 $i$ 类训练样本个数

- 样本 $\mathbf{x}$ 与类别  $\omega_i$  之间相似程度:

$$s(\mathbf{x}, \omega_i) = -\min_{\mathbf{y} \in D_i} d(\mathbf{x}, \mathbf{y})$$

## 2.1.3 最近邻分类器

- 
- 输入：需要识别的样本  $\mathbf{x}$ ，训练样本集

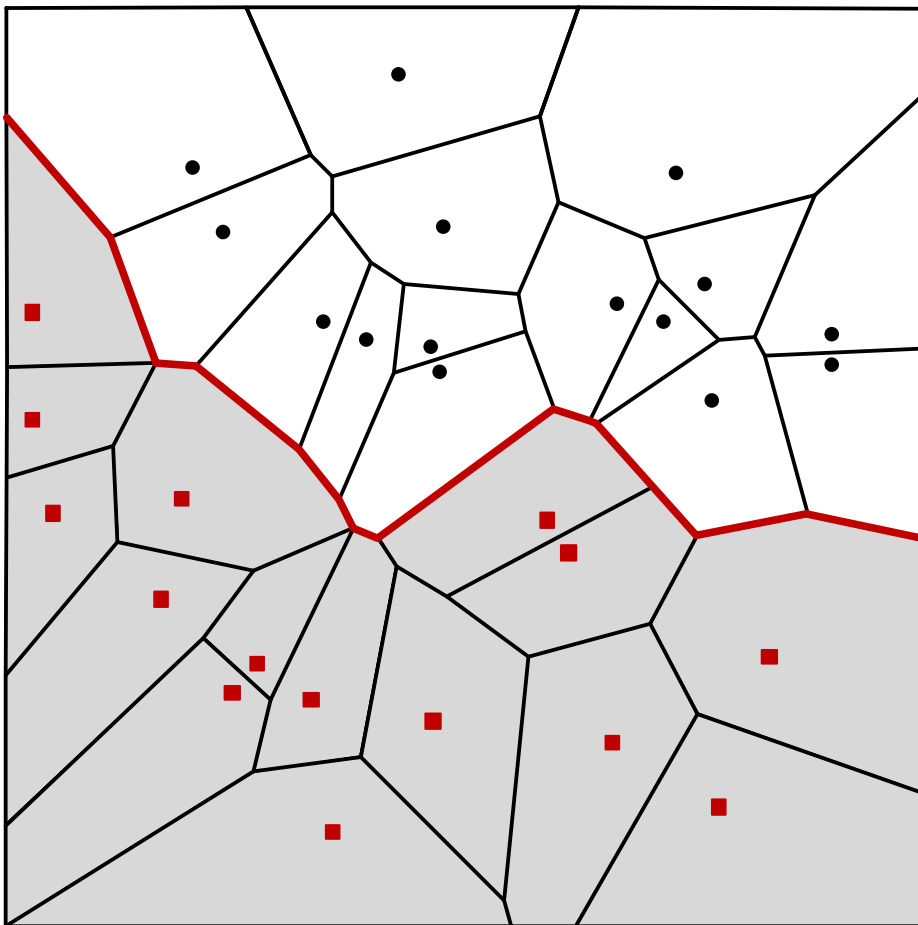
$$D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\};$$

- 寻找  $D$  中与  $\mathbf{x}$  距离最近的样本：

$$\mathbf{y} = \arg \min_{\mathbf{x}_i \in D} d(\mathbf{x}, \mathbf{x}_i);$$

- 输出：  $\mathbf{y}$  所属的类别；
-

## 2.1.3 最近邻分类器-Voronoi网格





## 2.1.3 最近邻分类器-Matlab实现

```
function output = NNClassify( X, S, T )  
Dist = pdist2( X, S );  
[dmin,id] = min( Dist' );  
output = T(id);
```

参数：X--识别样本 ( $m \times d$ ), S--训练样本矩阵 ( $n \times d$ )

T--样本的类别标号 ( $n \times 1$ ),  $1, \dots, c$

返回值：对 X 的分类结果 ( $m \times 1$ )

函数功能：最近邻分类算法

**Pdist2(X,Y)**

%X与Y的行向量，两两求距离

**[dmin,id]=min(D)**

%行向量dmin: D的各列向量的最小值

% id : 最小值所在行号

## 2.1.3 最近邻分类器——特点分析

- 训练样本数量较多时效果良好。
- 计算量大：
  - 每次识别时需要同所有训练样本计算距离
- 占用存储空间大：
  - 需要保存所有的训练样本，也比较
- 易受样本噪声影响：
  - 最近邻算法只依赖最近训练样本，当训练样本某些特征有偏差、或者标注错误， 导致错误

## 2.1.4 最近邻分类器的加速

### ■ 转化为单模板匹配

○ 用每个类别的训练样本学习出一个模板  $\mu$

问题：什么样的  $\mu_i$  最适合作为代表模板？

思路：距离训练样本距离都比较近的点。

模型：

$$\mu_i = \arg \min_{\mu \in R^d} \sum_{k=1}^{n_i} d(\mathbf{x}_k^{(i)}, \mu)$$

求解：

1) 构造准则函数：

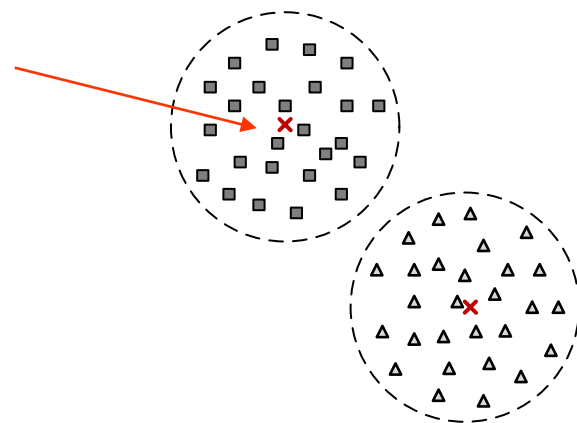
$$J_i(\mu) = \sum_{k=1}^{n_i} \left\| \mathbf{x}_k^{(i)} - \mu \right\|^2$$

误差平方和准则

2) 最优化：

$$\mu_i = \arg \min_{\mu \in R^d} J_i(\mu)$$

准则函数何处取得极值？



## 2.1.4 最近邻分类器的加速

### ■ 转化为单模板匹配


误差平方和准则函数：

$$J_i(\boldsymbol{\mu}) = \sum_{k=1}^{n_i} \underbrace{\left\| \mathbf{x}_k^{(i)} - \boldsymbol{\mu} \right\|^2}_{\text{范数的平方}} = \sum_{k=1}^{n_i} \underbrace{\left( \mathbf{x}_k^{(i)} - \boldsymbol{\mu} \right)^t \left( \mathbf{x}_k^{(i)} - \boldsymbol{\mu} \right)}_{\text{内积}}$$

极值点导数为0：

$$\nabla J_i(\boldsymbol{\mu}) = \frac{\partial J_i(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \sum_{k=1}^{n_i} 2 \left( \mathbf{x}_k^{(i)} - \boldsymbol{\mu} \right) (-1) = 2n_i \boldsymbol{\mu} - 2 \sum_{k=1}^{n_i} \mathbf{x}_k^{(i)} = 0$$

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{k=1}^{n_i} \mathbf{x}_k^{(i)}$$

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^t \mathbf{x}}$$


结论：计算每个类别训练样本的均值作为匹配模板

## 2.1.4 最近邻分类器的加速

### ■ 单模板匹配——训练

学习的模板 ( $c \times d$ )

模板对应的类别标签 ( $c \times 1$ )

对应样本的类别标签 ( $n \times 1$ )

**function** [Templates Labels] = OneTemplateTrain( **X**, **T** )

训练样本矩阵 ( $n \times d$ )

```
n = size(X,1);           %样本数
Labels = unique( T );    %标签合并 (每类取一个)
c = length(Labels);      %类别数
dim = size(X,2);         %特征维数
Templates = zeros(c,dim);
for i = 1:c
    id = find(T==Labels(i)); %找出所有该类别的样本
    Templates(i,:) = mean(X(id,:)); %求均值得到模版
end
```

## 2.1.4 最近邻分类器的加速

### ■ 单模板匹配——识别

分类结果 ( $m \times 1$ )

样本矩阵( $m \times d$ )

模板对应的类别标签  
( $c \times 1$ )

**function** Out = OneTemplatesClassify( X, Templates, Labels )

Dist = pdist2(X,Templates);

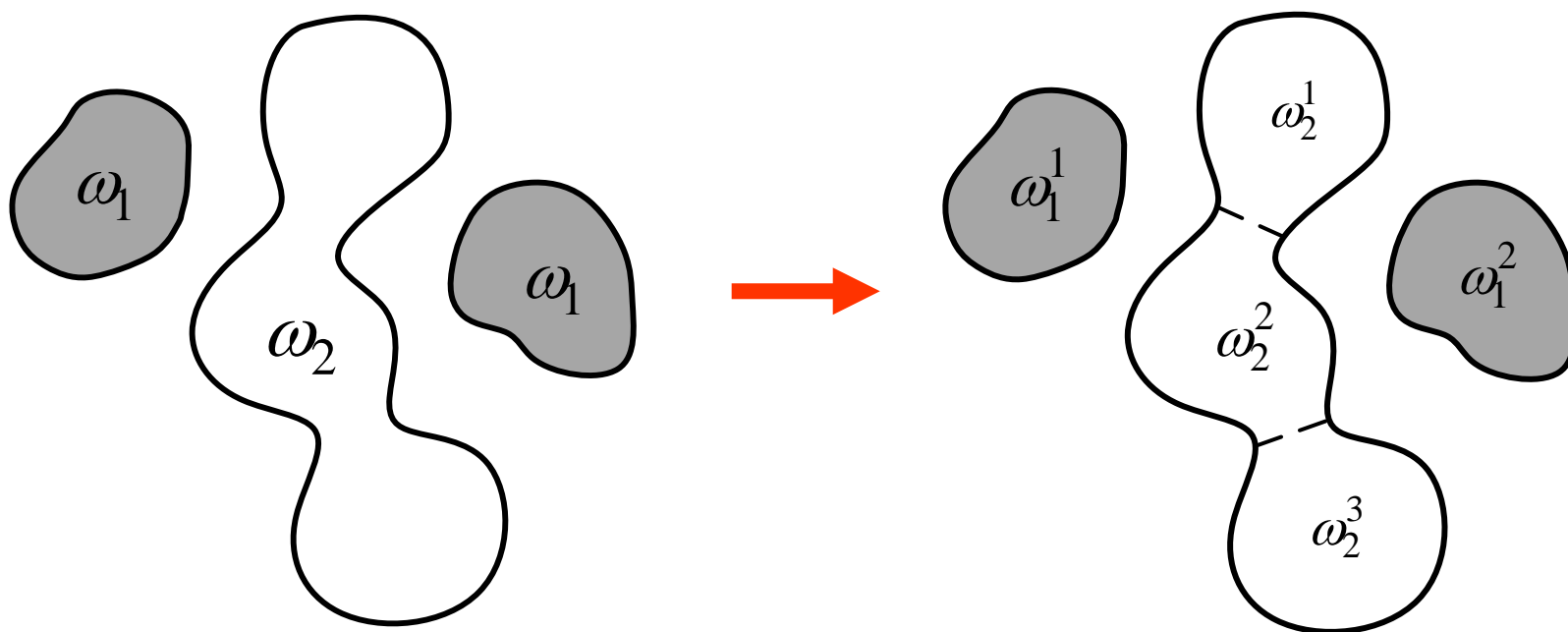
[y,id] = min(Dist,[],2); %Dist第二维最小值  
%即各行最小值，组成列向量y

模板矩阵 ( $c \times d$ )

Out = Labels(id);

## 2.1.4 最近邻分类器的加速

### ■ 从“单模板”到“多模板”

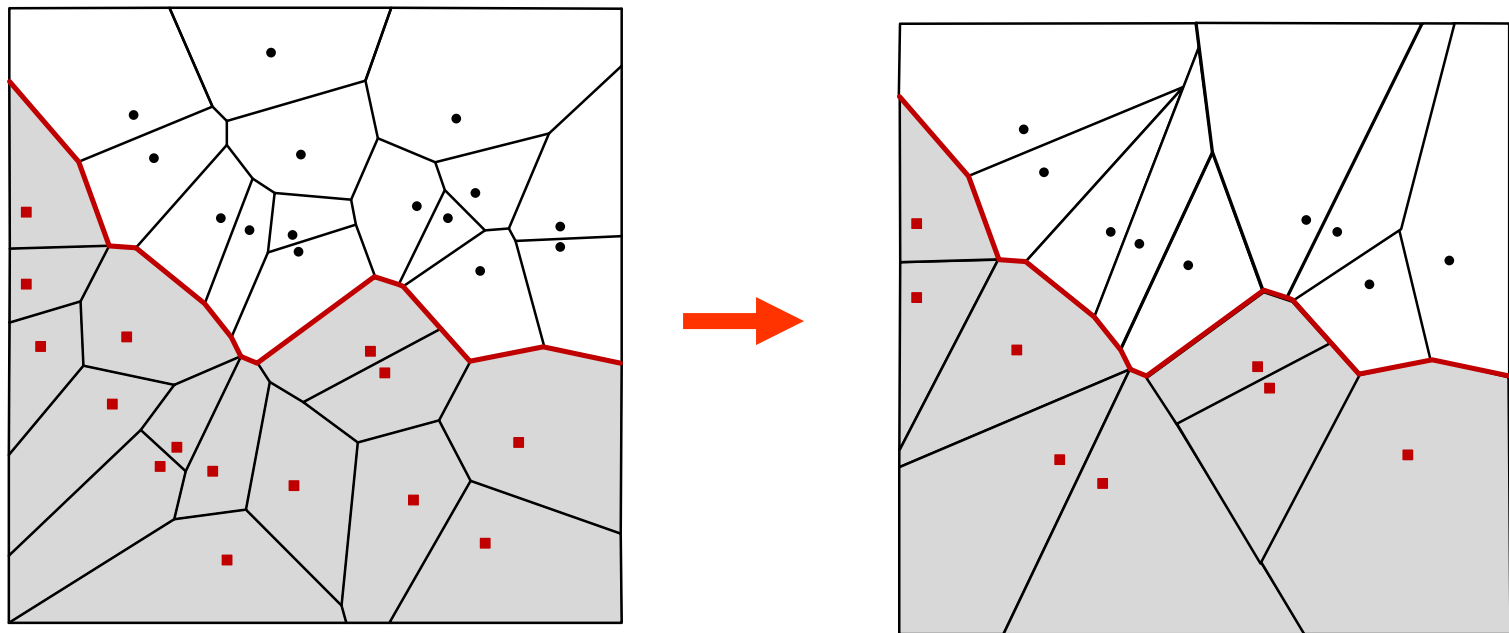


关键问题：高维特征空间如何划分？

——聚类分析

## 2.1.4 最近邻分类器的加速

- 使用一个或多个模板
  - 能够提高计算效率——减少匹配次数
  - 不能保证准确率——改变了分类面形状
- 近邻剪辑
  - 去掉对分类面“无用”的点——不改变分类面形状

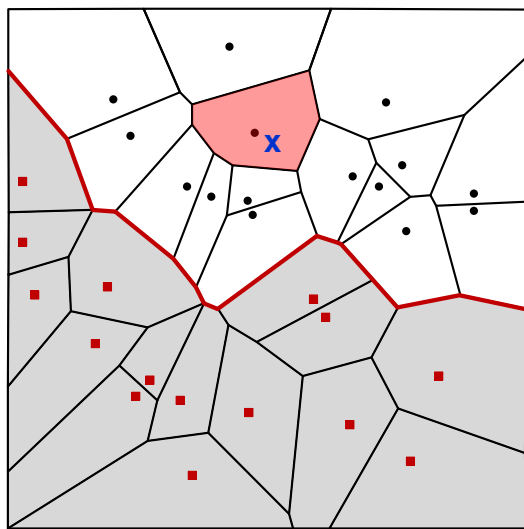




## 2.1.5 K-近邻分类器

### ■ 最近邻

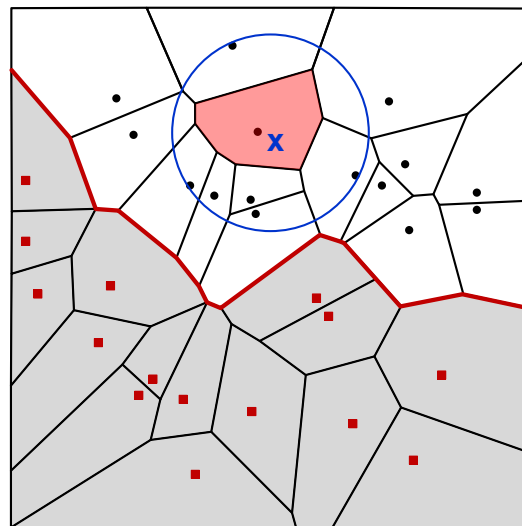
只依据距离最近的“一个”  
样本的类别



一个噪声样本就将导致  
一片错误分类区域

### ■ k-近邻

由距离最近的“K个”  
样本投票来决定



寻找样本x最近的 $k=7$ 个样本  
投票6:1，正确识别

## 2.1.5 K-近邻分类器——特点

### ■ K的选择

- K值选择的过小，算法的性能接近于最近邻分类；
- K值选择的过大，距离较远的样本也会对分类结果产生作用，这样也会引起分类误差。
- 适合的K值需要根据具体问题来确定。

### ■ 非平衡样本集 (某一类样本数量很大，而其它类别样本的数量相对较少)

- 样本数多的类别总是占优势

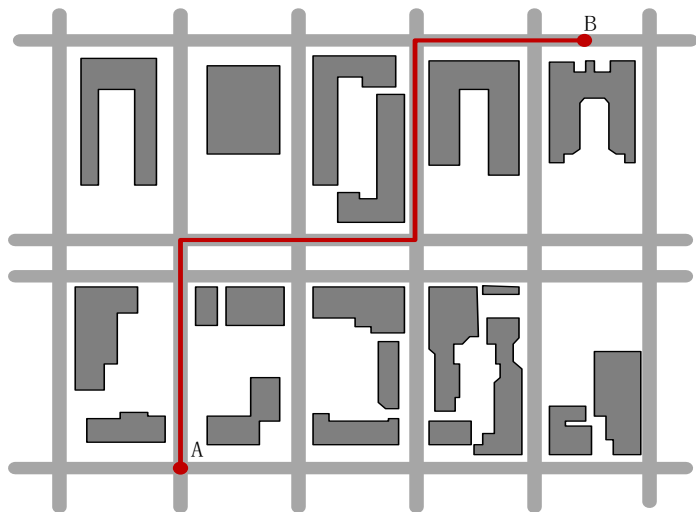
### ■ 计算量

- 需要与每一个训练样本计算距离。
- 寻找与其最相近K个样本。

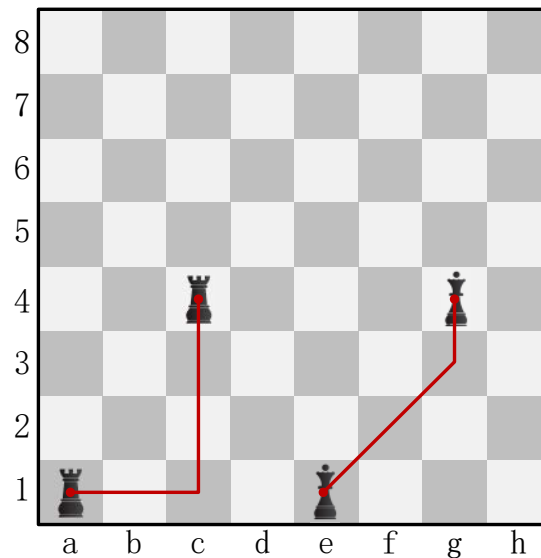
快速查找？先排序再查找——K-D树

## 2.2 距离和相似性度量

“距离”有很多种，如何选择？如何定义？



——街区距离



——切比雪夫距离

世界上最遥远的距离，不是生与死，而是我就站在你面前，你却不知道我爱你。

世界上最遥远的距离，不是我就站在你面前你却不知道我爱你，而是明明知道彼此相爱，却又不能在一起；

——泰戈尔，反正不是欧氏距离

## 2.2.1 距离度量

对于任意一个定义在两个矢量 $\mathbf{x}$ 和 $\mathbf{y}$ 上的函数 $d(\mathbf{x}, \mathbf{y})$ 只要满足如下4个性质就可以称作“距离度量”：

■ 非负性： $d(\mathbf{x}, \mathbf{y}) \geq 0$

■ 对称性： $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$

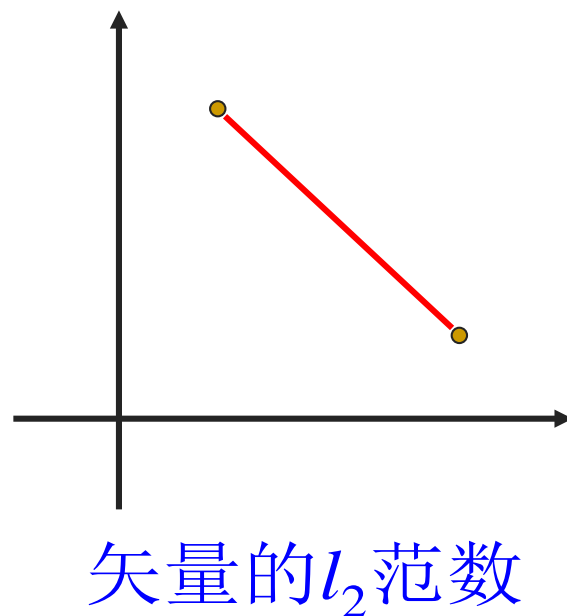
■ 自反性： $d(\mathbf{x}, \mathbf{y}) = 0$ , 当且仅当 $\mathbf{x} = \mathbf{y}$

■ 三角不等式： $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z})$

# 常用的距离函数

## ■ 欧氏距离: (Euclidean Distance)

$$\begin{aligned} d(\mathbf{x}, \mathbf{y}) &= \|\mathbf{x} - \mathbf{y}\|_2 \\ &= \left[ \sum_{i=1}^n (x_i - y_i)^2 \right]^{\frac{1}{2}} \\ &= \left[ (\mathbf{x} - \mathbf{y})^t (\mathbf{x} - \mathbf{y}) \right]^{\frac{1}{2}} \end{aligned}$$



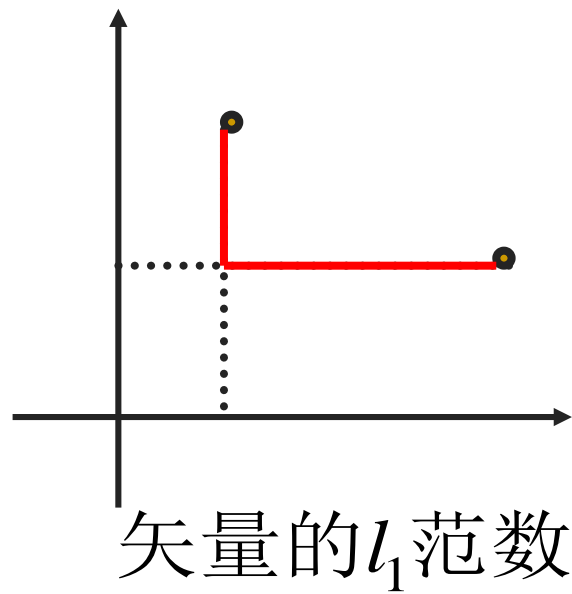
$$\mathbf{x}^t \mathbf{y} = \sum_{i=1}^n x_i y_i \text{ 是 } \mathbf{x} \text{ 与 } \mathbf{y} \text{ 之间的内积}$$

# 常用的距离函数

■ 街市距离：

(**Manhattan/city block/taxicab distance**)

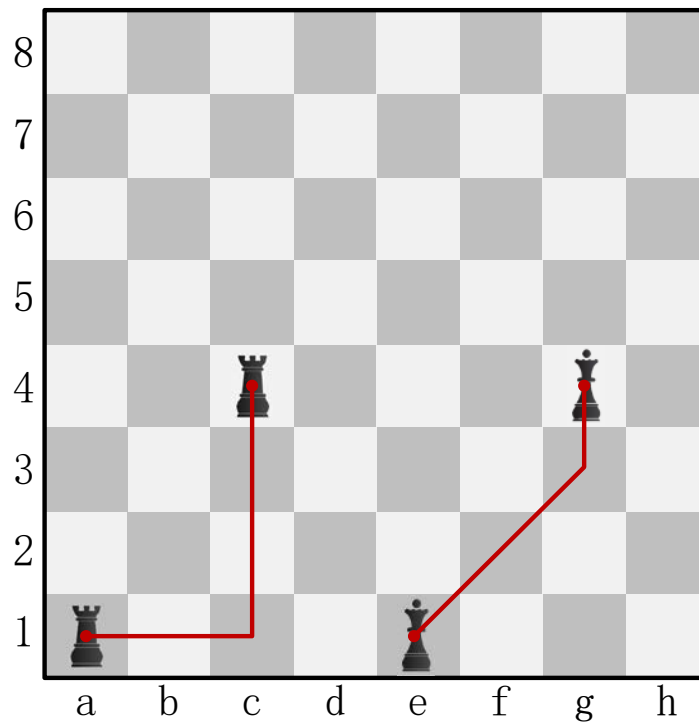
$$\begin{aligned} d(\mathbf{x}, \mathbf{y}) &= \|\mathbf{x} - \mathbf{y}\|_1 \\ &= \sum_{i=1}^n |x_i - y_i| \end{aligned}$$



# 常用的距离函数

## ■ 切比雪夫距离： (Chebyshev Distance)

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_{\infty}$$
$$= \max_{1 \leq i \leq d} |x_i - y_i|$$



# 常用的距离函数

## ■ 闵可夫斯基距离 (Minkowski Distance)

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

$q = 1$ : 街市距离

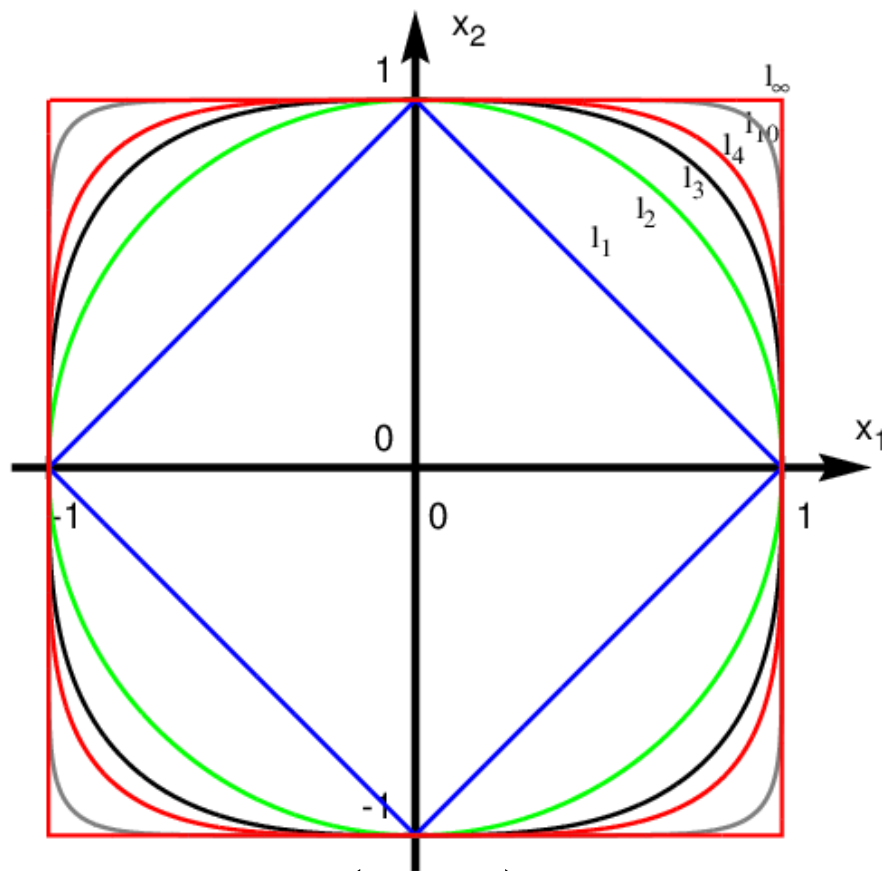
$q = 2$ : 欧氏距离

$q = \infty$ : 切比雪夫距离



闵氏距离具有平移不变性旋转

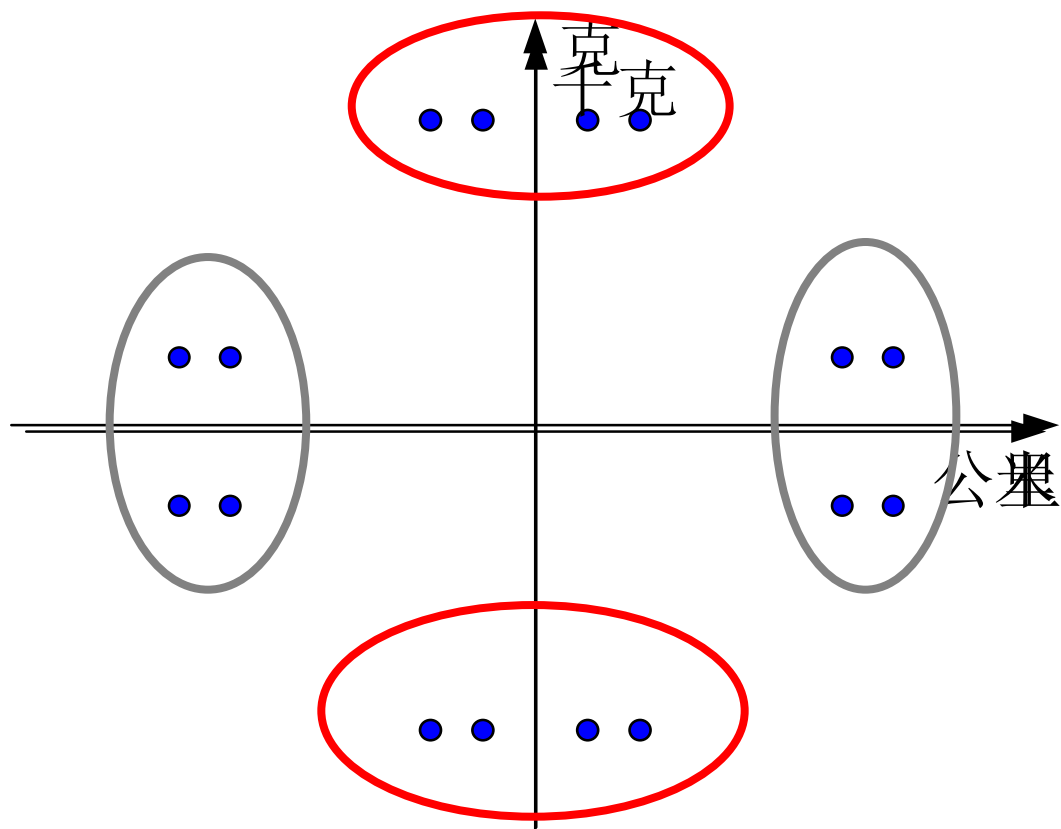
仅当  $q=2$  时，具有旋转不变性



$$d(\mathbf{x}, 0) = 1$$

# 样本规格化

## ■ 特征量纲的影响（缩放坐标轴）



# 样本规格化

- 使样本每一维特征都分布在相同或相似的范围內，计算距离度量时每一维特征上的差异都会得到相同的体现
- 方法1-均匀缩放：假设各维特征服从均匀分布将每一维特征都平移和缩放到  $[0,1]$ 內

1) 计算样本集每一维特征的最大、最小值

$$x_{j\min} = \min_{1 \leq i \leq n} x_{ij}, \quad x_{j\max} = \max_{1 \leq i \leq n} x_{ij}, \quad j = 1, \dots, d$$

2) 平移和缩放样本的每一维特征：

$$x'_{ij} = \frac{x_{ij} - x_{j\min}}{x_{j\max} - x_{j\min}}, \quad i = 1, \dots, n, \quad j = 1, \dots, d$$

# 样本规格化

- 使样本每一维特征都分布在相同或相似的范围内，计算距离度量时每一维特征上的差异都会得到相同的体现
- 方法2-高斯缩放：假设每一维特征都符合高斯分布，平移、缩放为标准高斯分布

1) 计算每一维特征的均值和标准差：

$$\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij}, \quad s_j = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \mu_j)^2}, \quad j = 1, \dots, d$$

进一步考虑各维特征间的协方差关系？  
——马氏距离

2) 规格化每一维特征：

$$x'_{ij} = \frac{x_{ij} - \mu_j}{s_j}, \quad i = 1, \dots, n, \quad j = 1, \dots, d$$

# 加权距离

关键问题：  
如何确定每一  
维特征的权重

- 计算距离时为“不同特征”引入“不同权重”

- 克服量纲影响
- 体现不同特征重要性

- 样本规格化，可以看做加权距离的特例：

加权欧氏距离：
$$d(\mathbf{x}, \mathbf{y}) = \left( \sum_{j=1}^d w_j (x_j - y_j)^2 \right)^{\frac{1}{2}}, \quad w_j \geq 0$$

均匀缩放：
$$x'_{ij} = \frac{x_{ij} - x_{j\min}}{x_{j\max} - x_{j\min}} \quad \rightarrow \quad w_j = \frac{1}{(x_{j\max} - x_{j\min})^2}$$

高斯缩放：
$$w_j = \frac{1}{s_j^2} \quad \rightarrow \quad x'_{ij} = \frac{x_{ij} - \mu_j}{s_j}$$

# 汉明距离 (Hamming Distance)

- 二值矢量  $\mathbf{x}, \mathbf{y} \in \{0,1\}^d$  (每个元素只取0或1)  
计算两个矢量对应位置元素不同的数量

$$d(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^d (x_j - y_j)^2$$

例:  $\mathbf{x} = (1, 1, 0, 0, 1, 1, 1)^t$ ,  $\mathbf{y} = (1, 0, 0, 0, 0, 0, 1)^t$

$$d(\mathbf{x}, \mathbf{y}) = 3$$

## 2.2.2 相似性度量

- 衡量相似度，不一定需要距离，可以选择更直接的方法衡量相似度
  - 两向量的夹角——角度相似性
  - 相关系数

相似性度量随着样本间相似程度的增加而增大，距离则是随着相似程度的增加而减小。为了保持一致性可以将相似度和距离进行转换：

$$d(\mathbf{x}, \mathbf{y}) = 1 - s(\mathbf{x}, \mathbf{y})$$

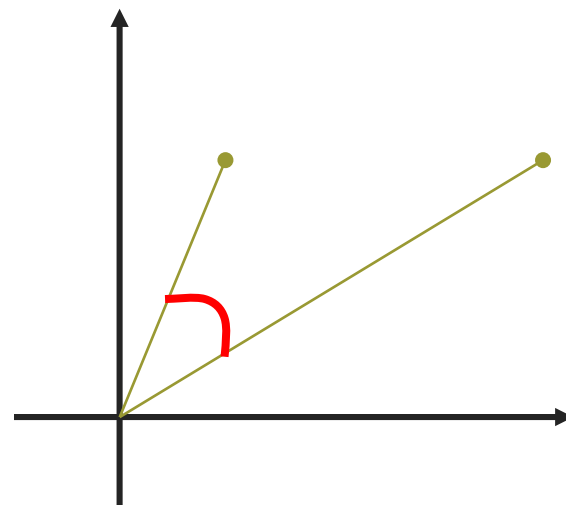
## ■ 角度相似性

- 当两个样本之间的相似程度只与它们之间的夹角有关、与矢量的长度无关时，可以使用矢量夹角的余弦来度量相似性。

$$s(\mathbf{x}, \mathbf{y}) = \cos \theta_{\mathbf{xy}} = \frac{\mathbf{x}^t \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$$

将向量归一为  
单位向量后做  
内积。

$$= \frac{\sum_{i=1}^d x_i y_i}{\sqrt{\sum_{i=1}^d x_i^2} \cdot \sqrt{\sum_{i=1}^d y_i^2}}$$





## ■ 相关系数

- 认为矢量 $\mathbf{x}$ 、 $\mathbf{y}$ 分别来自于两个样本集

样本集的均值分别为 $\mu_{\mathbf{x}}$ 、 $\mu_{\mathbf{y}}$ ，相关系数定义为：

$$s(\mathbf{x}, \mathbf{y}) = \frac{(\mathbf{x} - \mu_{\mathbf{x}})^t (\mathbf{y} - \mu_{\mathbf{y}})}{\|\mathbf{x} - \mu_{\mathbf{x}}\| \cdot \|\mathbf{y} - \mu_{\mathbf{y}}\|} = \frac{\sum_{i=1}^d (x_i - \mu_{xi})(y_i - \mu_{yi})}{\sqrt{\sum_{i=1}^d (x_i - \mu_{xi})^2} \cdot \sqrt{\sum_{i=1}^d (y_i - \mu_{yi})^2}}$$

- 将矢量 $\mathbf{x}$ 、 $\mathbf{y}$ 视为两个一维信号

$$\mu_x = \frac{1}{d} \sum_{i=1}^d x_i, \quad \mu_y = \frac{1}{d} \sum_{i=1}^d y_i, \quad \mathbf{e}: \text{所有元素均为1的} d \text{维矢量}$$

$$s(\mathbf{x}, \mathbf{y}) = \frac{(\mathbf{x} - \mu_x \mathbf{e})^t (\mathbf{y} - \mu_y \mathbf{e})}{\|\mathbf{x} - \mu_x \mathbf{e}\| \cdot \|\mathbf{y} - \mu_y \mathbf{e}\|} = \frac{\sum_{i=1}^d (x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^d (x_i - \mu_x)^2} \cdot \sqrt{\sum_{i=1}^d (y_i - \mu_y)^2}}$$

## 2.2.3 Matlab实现

### ■ knnclassify

(Bioinformatics Toolbox K-近邻算法分类函数)

---

Class = `knnclassify`( Sample, Training, Group, k, distance, rule )

---

参数:

Sample --  $m \times d$ 矩阵;

Training --  $n \times d$ 矩阵;

Group --  $n \times 1$ 矩阵, 与Training每一行对应的类别标签;

k -- K-近邻算法参数, 缺省为1 (最近邻算法);

distance -- 距离度量参数 euclidean: 欧几里得距离度量;

cityblock: 街市距离度量; hamming: 汉明距离度量;

cosine: 角度相似性度量, correlation: 相关系数;

rule -- 分类参数; k设置为偶数时, 出现票数相同情况下的处理规则

返回:

class --  $m \times 1$ 矩阵, 对应Sample每一行的分类结果。

---

## 2.2.3 Matlab实现

### ■ ClassificationKNN

(Statistics Toolbox中更复杂的K-近邻算法的实现)

先训练一个**ClassificationKNN**模型，然后调用**predict**函数分类

---

```
model = ClassificationKNN.fit( X, Y, Name, Value )
```

---

参数：

**X** --  $n \times d$ 矩阵，样本集矩阵，**n**个训练样本，**d**维特征；

**Y** --  $n \times 1$ 矩阵，类别标签；

**Name** -- 参数名称； **Value** -- 参数的值；

返回：

**model** -- K-近邻分类器。

---

使用一组**Name**和对应的**Value**设置K-近邻分类器的参数

<b>Name</b>	<b>Value</b>	<b>说明</b>
<b>Distance</b>	<b>euclidean</b>	欧几里得距离度量
	<b>cityblock</b>	街市距离度量
	<b>chebychev:</b>	切比雪夫距离度量;
	<b>minkowski:</b>	闵可夫斯基距离度量;
	<b>seuclidean:</b>	规格化样本的欧氏距离度量
	<b>hamming</b>	汉明距离
	<b>cosine</b>	角度相似性度量
	<b>correlation</b>	相关系数
<b>DistParameter</b>	<b>minkowski</b>	闵可夫斯基距离中的指数p
	<b>seuclidean</b>	规格化样本的标准差
<b>NumNeighbors</b>	<b>k</b>	<b>K-近邻算法参数</b>

Name	Value	
分类策略 BreakTies	nearest	先用K-近邻原则分类，如果出现数量相同的情况则以最近邻的原则分类
	random	先用K-近邻原则分类，如果出现数量相同的情况则在样本数量最多的几个类别中随机的选择一个；
	smallest	先用K-近邻原则分类，如果出现数量相同的情况则选择类别标号最小的类别；

## 2.2.3 Matlab实现

- ClassificationKNN  
(Statistics Toolbox中更复杂的K-近邻算法的实现)

---

**$Z = \text{predict}(\text{model}, X)$**

---

参数:

**model** -- K-近邻分类器;

**X** --  $m \times d$ 矩阵, 测试样本矩阵,  $m$ 个样本,  $d$ 维特征数

返回:

**Z** --  $m \times 1$ 矩阵, K-近邻分类器识别结果

---

## 2.3 分类器性能评价

### ■ 2.3.1 评价指标

- 拒识: 只对有把握的样本判别类别, 对没有把握的样本拒绝识别.
- 对  $M$  个样本分类,  $m_r$  个被拒识,  $m_e$  个被分类错误

错误率:  $P_e \approx \frac{m_e}{m - m_r}$

拒识率:  $P_r \approx \frac{m_r}{m}$

## 2.3 分类器性能评价

### ■ 2.3.1 敏感性与特异性

		分类结果	
		阳性	阴性
实际类别	患者	$a$	$b$
	正常人	$c$	$d$

○ 敏感性（真阳率）：患者被诊断出来的比率

$$P_s \approx \frac{a}{a+b}$$

○ 特异性（1-假阳率）：正常人不被误诊的比率

$$P_n \approx \frac{d}{c+d}$$

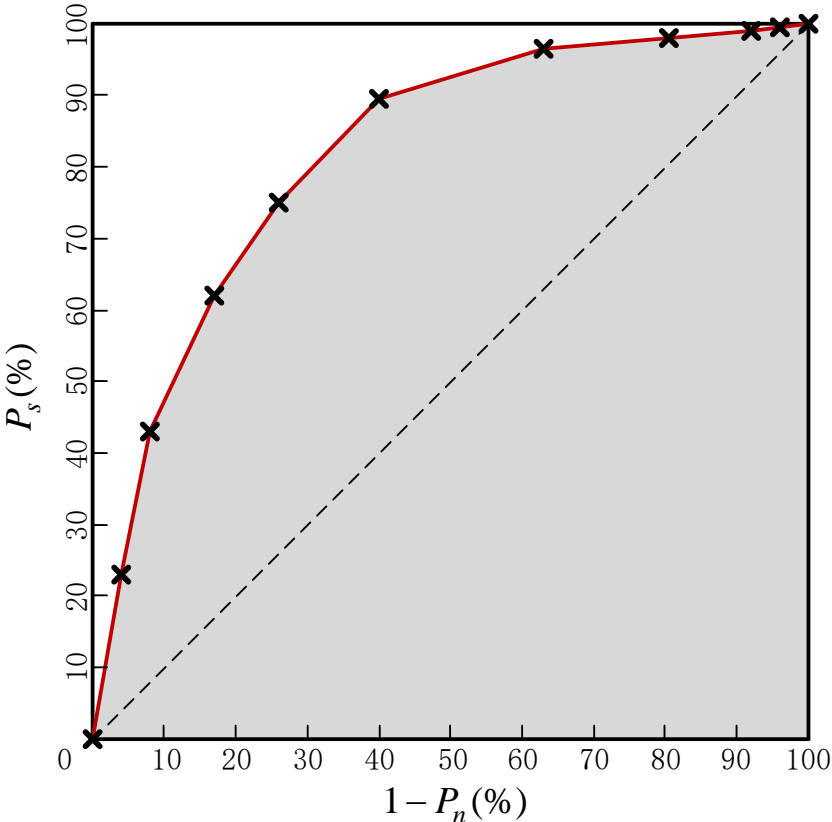


# 2.3 分类器性能评价

## 2.3.1 ROC曲线

	0	1	2	3	4	5	6	7	8	9	10
$P_s$	0	23.1	43.8	62.2	72.3	89.6	96.1	98	98.9	99.5	100
$1-P_n$	0	4.3	8.6	17.1	25.9	40.4	63.5	80.2	92.3	96.5	100

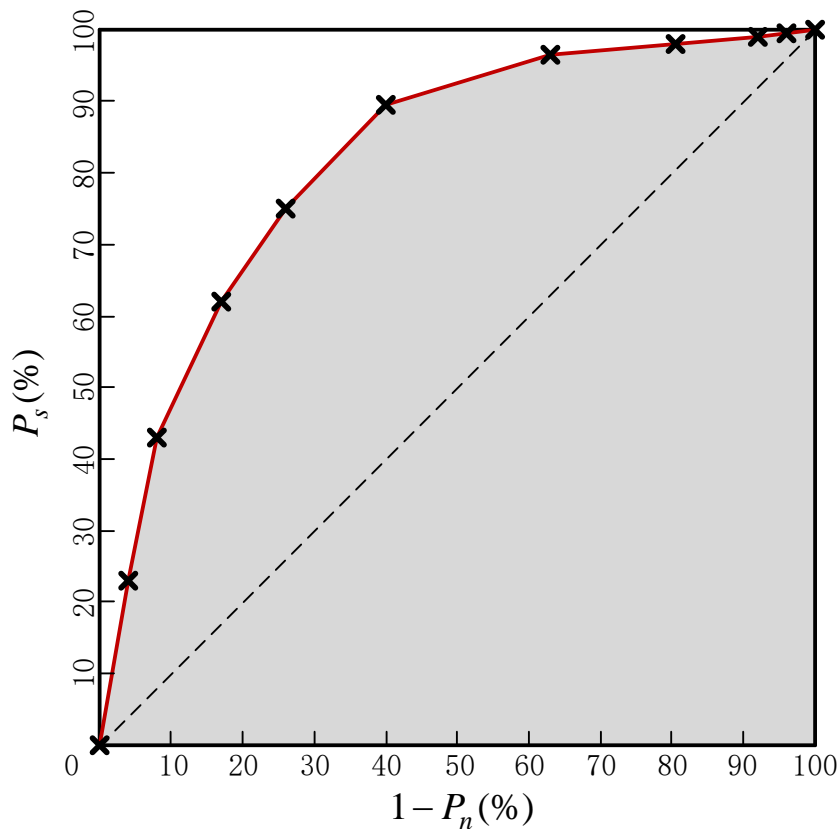
纵轴为敏感性



横轴为假阳率

## 2.3 分类器性能评价

### ■ 2.3.1 ROC曲线



- 判断不同分类方法的优劣：  
**ROC**曲线下方的面积越大性能越好。

- 选择分类器参数：  
一般应用：  $P_s = 1 - P_n$

敏感性要求高的场合：

$$P_s > 95\%$$

## 2.3 分类器性能评价

### ■ 2.3.1 召回率和准确率

		分类结果	
		检索到	未检索到
实际类别	相关	$a$	$b$
	不相关	$c$	$d$

○ 召回率（查全率）：相关的信息中被检索出来的比例

$$P_s \approx \frac{a}{a+b} \quad \text{=敏感性}$$

○ 准确率：检索到的信息中，与主题相关的比例

$$P \approx \frac{a}{a+c}$$

## 2.3 分类器性能评价

### ■ 2.3.1 召回率和准确率——调和平均

○ 兼顾召回率和准确率，使用二者的调和平均

$$F_1 = \frac{2}{\frac{1}{R} + \frac{1}{P}} = \frac{2RP}{R + P}$$

## 2.3 分类器性能评价

### ■ 2.3.2评价方法——如何进行性能评价试验？

- 两分法：随机将训练样本集划分为不相交的两个子集
  - 两子集分别用于训练、测试。重复 $k$ 次取均值
  - 只能用一部分样本学习分类器，另一部分样本测试性能
- 交叉验证：随机地分成不相交的 $k$ 个子集
  - 每个子集中的样本数量相同。使用 $k-1$ 个训练，剩余样本测试。
  - 以 $k$ 次的平均值作为分类器的性能评价指标

## 2.3 分类器性能评价

### ■ 2.3.2 评价方法——如何进行性能评价试验？

#### ○ Bootstrap方法（20世纪70年代提出）

1. 对样本集  $D$  中有放回地抽取  $n$  个样本，  
组成 Bootstrap 样本集  $A$ 、 $B$ （样本可能重复）；
2. 用集合  $A$  训练、 $B$  测试得到性能指标估计；
3. 上述过程重复  $k$  次，取  $k$  次的平均值作为分类器性能的评价。

# 新手作业：手写数字识别

## ■ UCI 数据集

- Semeion Handwritten Digit Data Set

## ■ MatLab 或 Python (NumPy, SciPy, Matplotlib, scikit-learn )

## ■ 分类方法？

- 距离分类器

## ■ 如何评价？

- 如何评价识别结果？

- 如何改进？

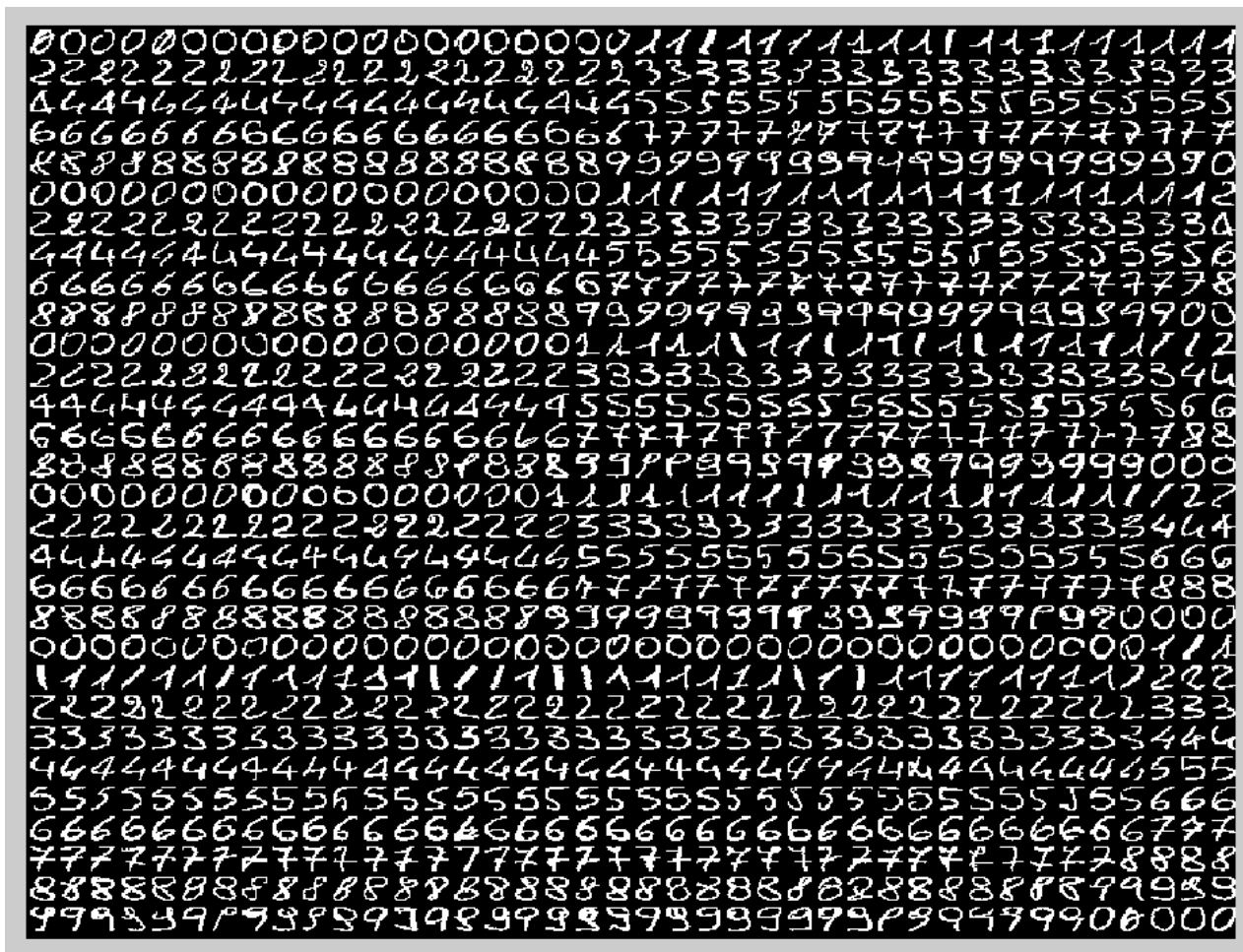
➤ 如果你从来没有做过一个模式识别或机器学习程序，那么请你务必练习一下。

➤ 如果你不熟悉基于Python的机器学习库，请开始熟悉。

➤ 高手请忽略😊

UCI 数据集 <http://archive.ics.uci.edu/ml/>

## Semeion Handwritten Digit Data Set





# 样本库 matlab基本操作

`load e:\char\semeion.data` 加载数据文件semeion.data 每行266个元素构成一个样本，共1593个

`img=semeion(:,1:256);` 每个样本的前256维为16\*16大小的像素点阵，0：黑 1：白

`label=semeion(:,257:266);` 每个样本的后10维为类别标签  
例： 01000 00000 代表第二类

`tmp = find(label(:,i));` 获取第i类所有样本的行号

`u= mean(img(tmp,:));` 获取第i类的样本均值

# 新手作业：手写数字识别

## ■ 用Matlab或Python实现该字符集的识别

- 代码+注释，规范命名
- 模版匹配，最近邻，k近邻
- 如何实验并评价各类方法的效果？
- 你有哪些想法或改进？能使识别率更高？

- 大家自行分组，找几位高手做组长、副组长
- 组长们负责协助新手童鞋调程序、做实验
- 组长做得好，平时成绩、实验加分☺
- 新手作业自愿完成，按组打包，下周1前发到群文件

# 性能不好怎么办？

- 理想主义者：让我多想想，规划一个好方案再动手。
- 实践主义者：先设计一个基本框架，快速迭代！
  - 与其花大量时间考虑如何构建一个完美的机器学习系统，还不如尽快构建一个简单的原型。
  - 在几天内构建第一个原型，然后就会看到新的线索，告诉你该选择什么样的方向去改进原型的性能。
  - 在下一次迭代中，可以根据这些线索改进系统，并构建系统的下一个版本。一次接一次，循环往复。

# 性能不好怎么办？

## ■ 可以从那几个方向改进？

- 特征：更精巧、更有效的特征计算
- 方法：更复杂的模型、更有针对性的机器学习方法
- 样本：更大的样本集（收集、变换、仿真）

■ 高手作业：阅读《机器学习训练秘籍》

<https://accepteddoge.com/machine-learning-yearning-cn/docs/ch58/>