模 式 识 别

第4章：线性判别函数分类器

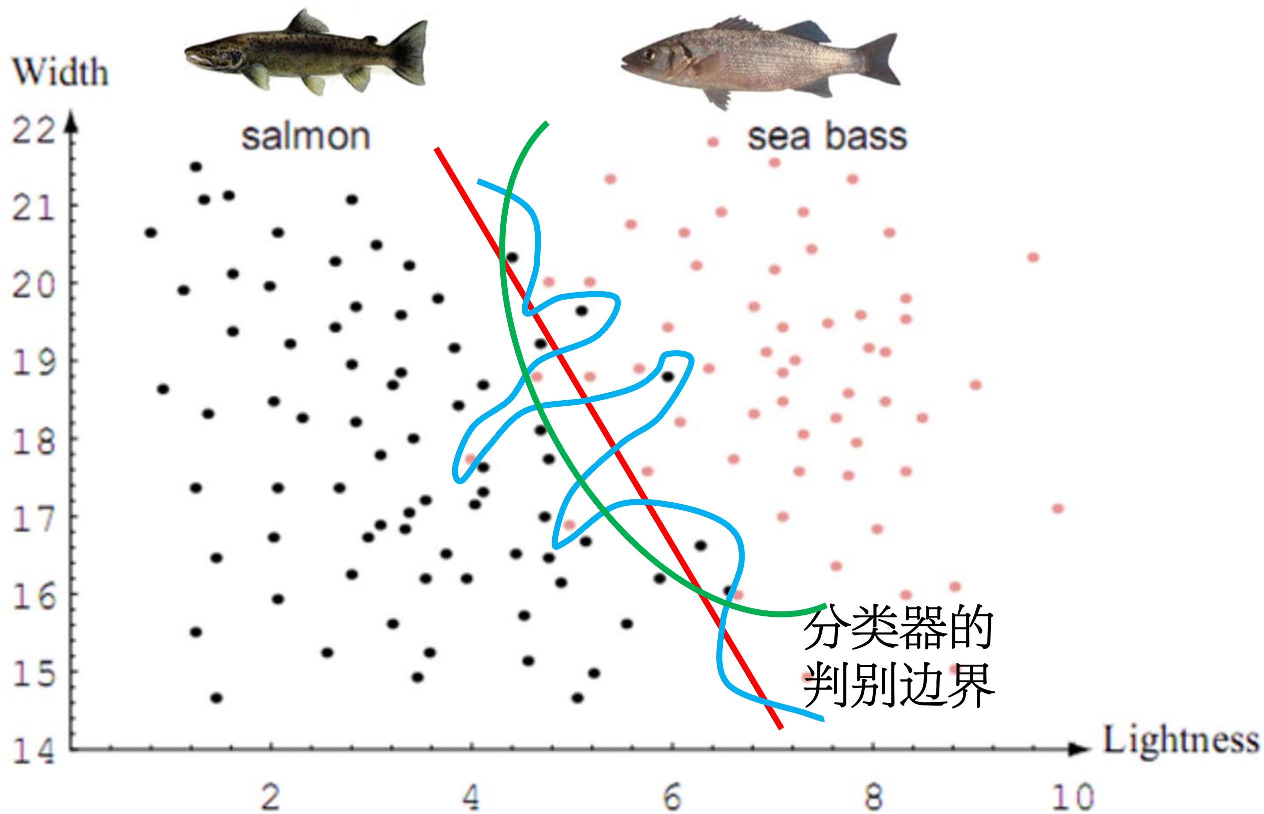
主讲人：张治国
zhiguozhang@hit.edu.cn

本章内容

- 线性判别函数和线性分类界面
- 感知器的准则、算法和问题
- 最小平方误差算法和平方误差准则
- 线性判别函数分类器用于多类别问题
  - 一对多方式
  - 一对一方式
  - 扩展的感知器算法
  - 感知器网络

分类界面

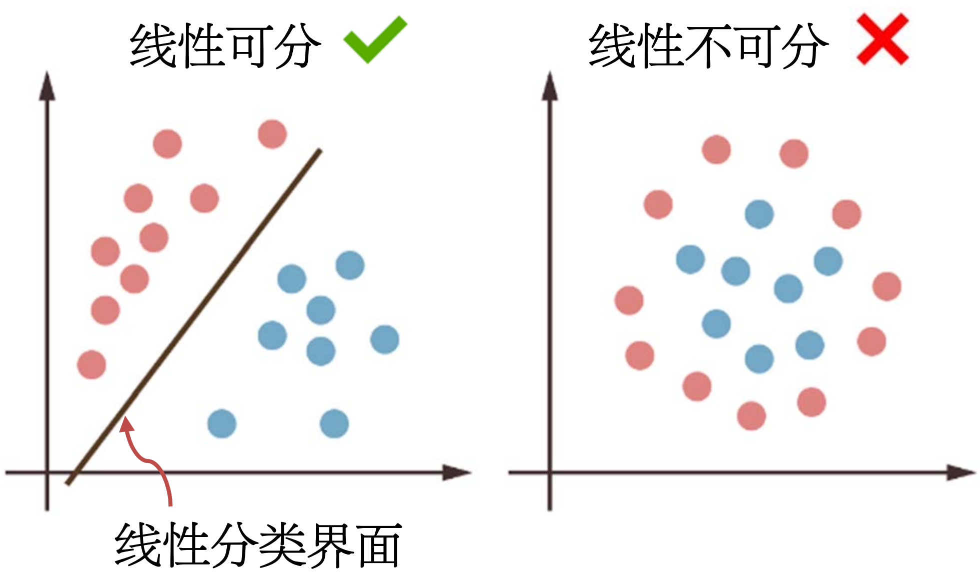
- 使用光泽度和宽度两个特征分类海鲈鱼和鲑鱼，如何确定分类界面？



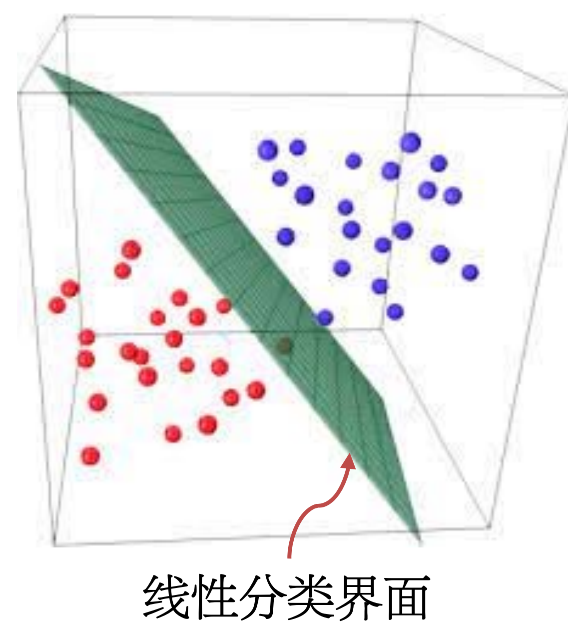
线性分类界面

- 直线、平面和超平面都可以称为线性分类界面，采用线性分类界面区分两类样本的方法称为线性分类器。

二维空间



三维空间



线性判别函数

- d 维空间中超平面 H 的方程为：

$$
\begin{aligned}
g(\boldsymbol{x}) &= w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + w_0 \\
&= \sum_{i=1}^{d} w_i x_i + w_0 \\
&= \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0
\end{aligned}
$$

其中 $\boldsymbol{x} = [x_1, \cdots, x_d]^T$ 是 d 维特征矢量，

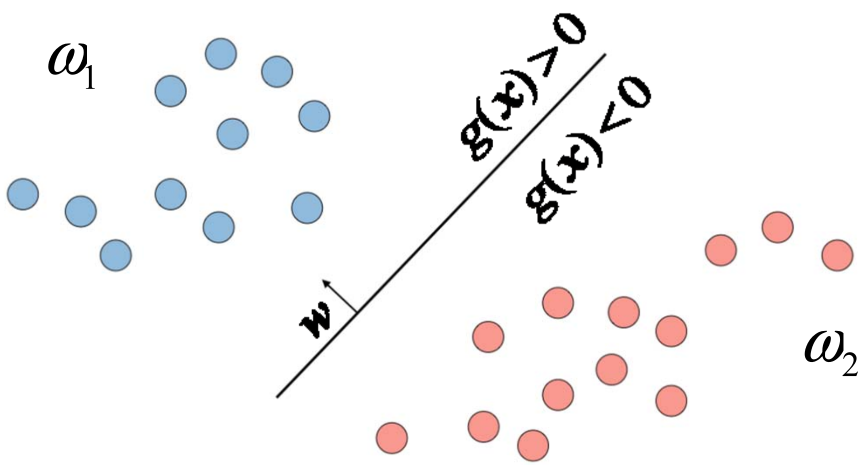
$\boldsymbol{w} = [w_1, \cdots, w_d]^T$ 是 d 维权值矢量，

w_0 是偏置。

线性判别函数

- 根据判别函数 $g(\boldsymbol{x})$ 的正负值可以判断 $\boldsymbol{x}$ 处于线性分类界面划分的哪一个空间。

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 \begin{cases} > 0, & \boldsymbol{x} \in \omega_1 \\ < 0, & \boldsymbol{x} \in \omega_2 \\ = 0, & \text{拒识} \end{cases}$$



线性判别函数

- 关于线性判别函数和线性分类界面的三个断言：

I. 线性分类界面 H 将特征空间划分为两个区域：一个区域中 $g(\boldsymbol{x}) > 0$，另一个区域中 $g(\boldsymbol{x}) < 0$。

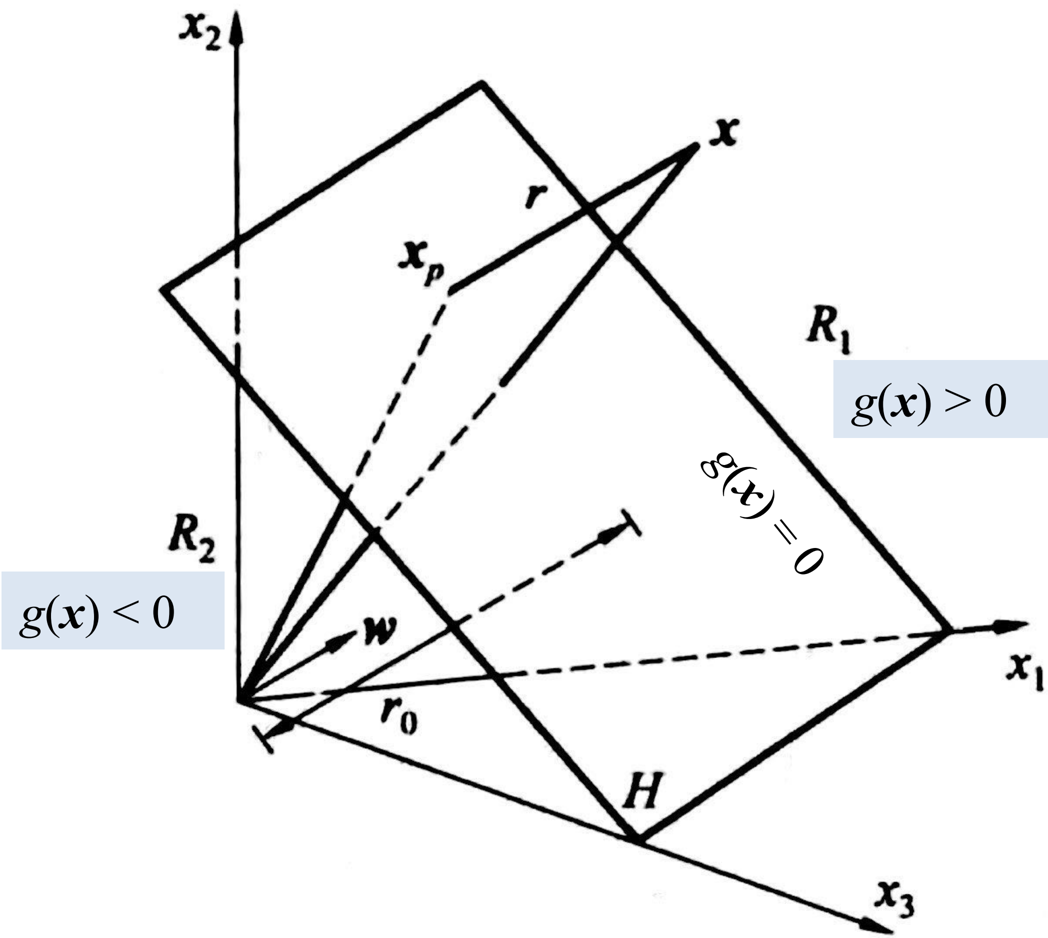
II. 权值矢量垂直正交于分类界面，并且指向 $g(\boldsymbol{x}) > 0$ 的区域。

III. 偏置 w_0 与坐标原点到分类界面 H 的距离 r_0 有关：$r_0 = |w_0| / \|\boldsymbol{w}\|$ 。

证明详见课本73-74页（4.1.2）。

线性判别函数

- 线性判别函数的几何解释



- 线性分类界面 H 是 d 维空间中的一个超平面；
- 分类界面将 d 维空间分成两部分，R_1，R_2分别属于两个类别；
- 判别函数的权矢量 $\boldsymbol{w}$ 是一个垂直于分类界面 H 的矢量，其方向指向区域 R_1，即 $g(\boldsymbol{x}) > 0$的区域；
- 偏置 w_0 与原点到分类界面 H 的距离有关。

感知器

- 假设有一个包含 n 个样本的集合 $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$ ，一些标记为 ω_1 ,另一些标记为 ω_2 。用这些样本来确定一个判别函数 $g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$ 的权矢量 $\boldsymbol{w}$ 和偏置 w_0。
- 在线性可分的情况下，希望得到的判别函数能够将所有的训练样本正确分类。
- 在线性不可分的情况下，判别函数产生错误的概率最小。

感知器

- 为后续分析，将样本和权值增广化和规范化。
- 增广化：增广的权值矢量 $\boldsymbol{a}=[\boldsymbol{w}^T, w_0]^T$

  增广的特征矢量 $\boldsymbol{y}=[\boldsymbol{x}^T, 1]^T$

  则 $g(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{x}+w_0 \Leftrightarrow g(\boldsymbol{y})=\boldsymbol{a}^T\boldsymbol{y}$
- 规范化：将 ω_2 的样本 $\boldsymbol{y}$ 乘以 -1，则有

$$\begin{cases} g(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{x}+w_0>0, & \forall \boldsymbol{x}\in\omega_1 \\ g(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{x}+w_0<0, & \forall \boldsymbol{x}\in\omega_2 \end{cases} \overset{\text{增广化}}{\Leftrightarrow} \begin{cases} g(\boldsymbol{y})=\boldsymbol{a}^T\boldsymbol{y}>0, & \forall \boldsymbol{x}\in\omega_1 \\ g(\boldsymbol{y})=\boldsymbol{a}^T\boldsymbol{y}<0, & \forall \boldsymbol{x}\in\omega_2 \end{cases}$$

$$\overset{\text{规范化}}{\Leftrightarrow} \begin{cases} g(\boldsymbol{y})=\boldsymbol{a}^T\boldsymbol{y}>0, & \forall \boldsymbol{x}\in\omega_1 \\ g(\boldsymbol{y})=\boldsymbol{a}^T\boldsymbol{y}>0, & \forall \boldsymbol{x}\in\omega_2 \end{cases} \Leftrightarrow \underbrace{g(\boldsymbol{y})=\boldsymbol{a}^T\boldsymbol{y}>0}_{\text{统一形式}}$$

感知器

- 线性判别分类器最直观的优化准则函数是被错误分类的样本数量，但这种优化函数很难求解。
- 感知器准则：以被错误分类的样本到分类界面的“距离”之和为准则进行优化

$$J_P(\boldsymbol{a}) = \sum_{\boldsymbol{y}\in Y} (-\boldsymbol{a}^T \boldsymbol{y})$$

其中 $Y = \{\boldsymbol{y} \mid \boldsymbol{y} \in D, \boldsymbol{a}^T \boldsymbol{y} < 0\}$ 是被错分的样本集。

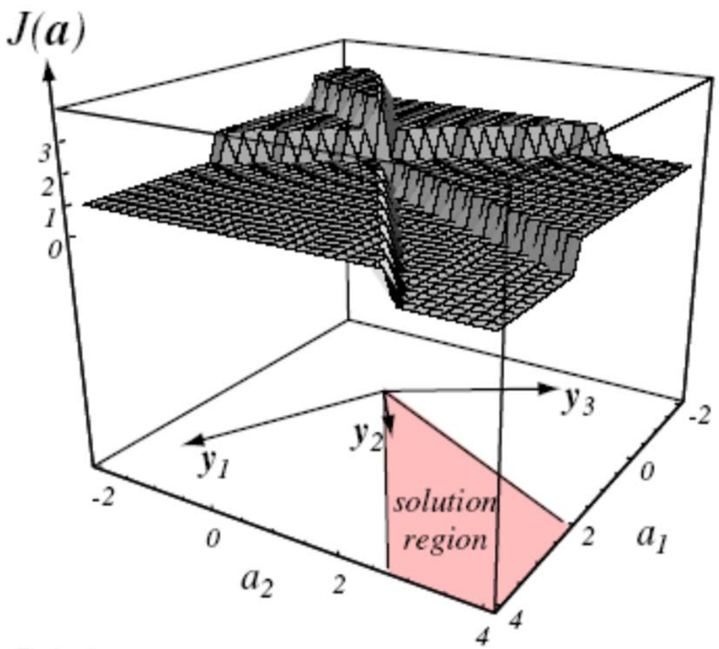
- 因为 $\boldsymbol{y}$ 被规范化，所以被错误识别时 $\boldsymbol{a}^T\boldsymbol{y} < 0$。因此 $J_P \geqslant 0$，且当无错误样本时，J_P有最小值0。

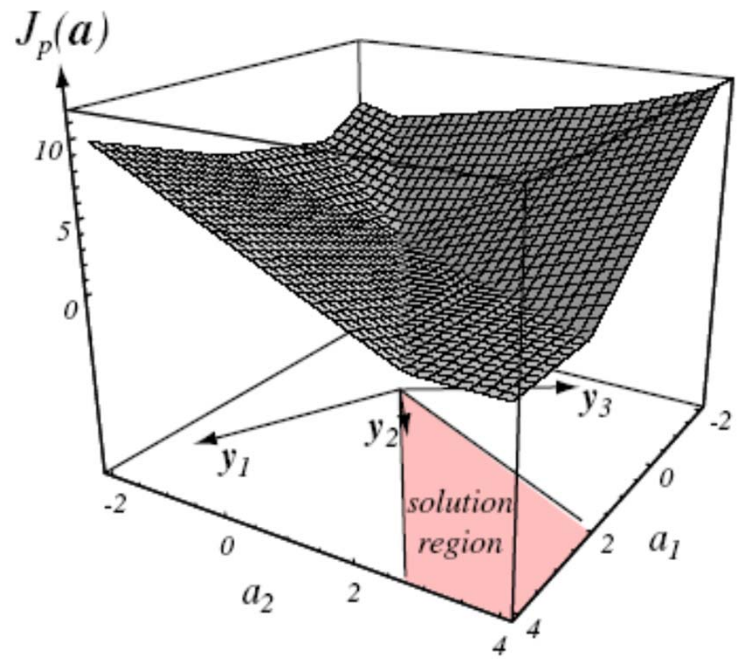
感知器

严格地说，分段线性函数在分段处无法求导。收敛性的证明见课本78-79页。

以错分样本数最少作为准则，分段均值，无法迭代优化

以错分样本到判别界面距离之和作为准则，分段线性，可迭代优化



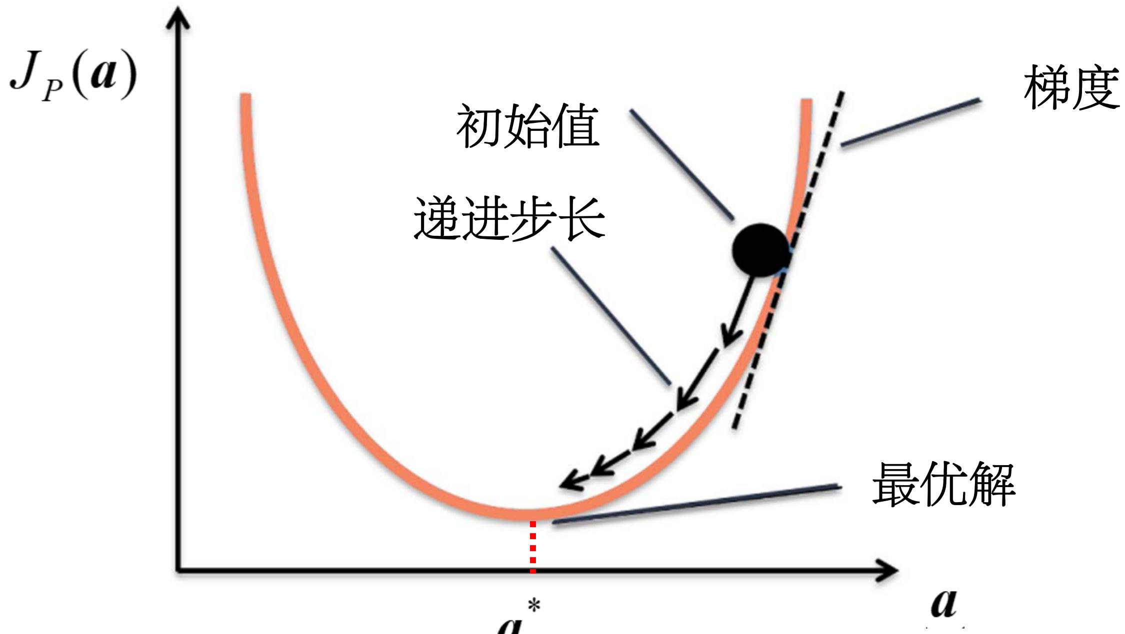


感知器

- 感知器准则函数的优化求解问题为：

$$\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}} J_P(\boldsymbol{a})$$

- 最优化方法采用最多的是梯度下降法。详见课本附录B.2



感知器

- 计算 J_P 关于 $\boldsymbol{a}$ 的梯度矢量：

$$\nabla J_P(\boldsymbol{a}) = -\sum_{\boldsymbol{y}\in Y} \boldsymbol{y}$$

- 采用梯度下降法沿梯度的负方向迭代计算 $\boldsymbol{a}$：

$$\boldsymbol{a}(k+1) = \boldsymbol{a}(k) + \eta(k)\sum_{\boldsymbol{y}\in Y} \boldsymbol{y}$$

其中 $\eta(k)$ 称为学习率，或称步长。

感知器算法

- 感知器算法（批量调整版本）：每一轮迭代输入所有训练样本，由当前权值 $\boldsymbol{a}(k)$ 判断哪些样本被错误识别，所有错误样本求和，然后调整权重。

- 初始化：$\boldsymbol{a}(0)$，学习率 $\eta(k)$，收敛精度 θ，$k=0$；
- do $k=k+1$

$$\boldsymbol{a}(k+1)=\boldsymbol{a}(k)+\eta(k)\sum_{\boldsymbol{y}\in Y}\boldsymbol{y}$$

until $\quad \|\eta(k)\sum_{\boldsymbol{y}\in Y}\boldsymbol{y}\|<\theta$

- 输出：$\boldsymbol{a}$

感知器算法

- 感知器算法（单样本调整版本）：每一轮迭代输入一个训练样本（任意顺序），如果识别正确则输入下一个样本，否则将权值 $\boldsymbol{a}(k)$ 与被错误识别样本 $\boldsymbol{y}$ 求和得到新权值。相当于 $\eta(k) = 1$。

- 初始化：$\boldsymbol{a}(0)$，$k = 0$；
- do $k = k + 1$

  $i = (k+1) \bmod n$

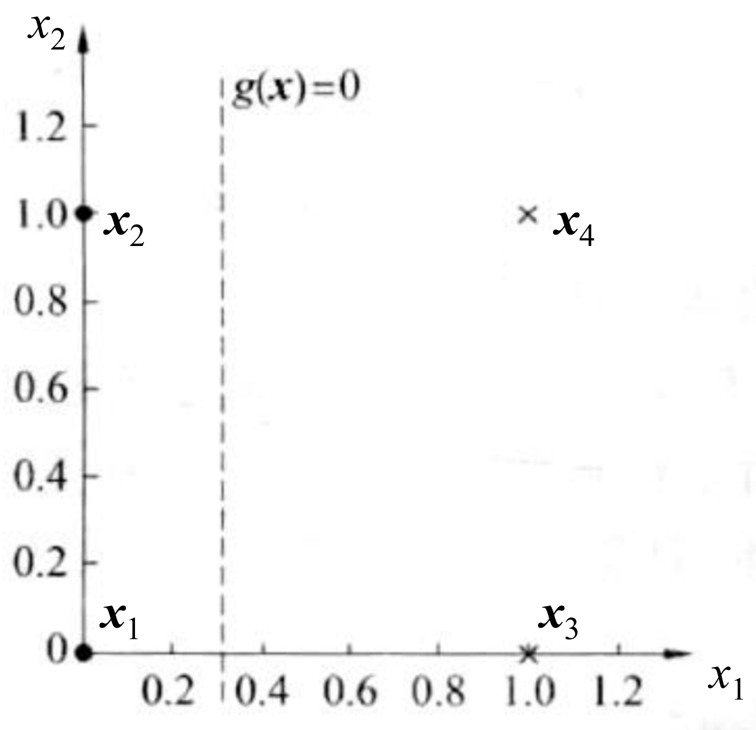
  如果 $\boldsymbol{y}_i$ 被 $\boldsymbol{a}$ 错误识别，则 $\boldsymbol{a}(k+1) = \boldsymbol{a}(k) + \boldsymbol{y}_i$

  until 全部样本被正确识别
- 输出：$\boldsymbol{a}$

感知器算法

- 例：现有两类训练样本，采用单样本调整感知器算法学习线性分类器。

$$\omega_1: \boldsymbol{x}_1=(0,0)^T, \boldsymbol{x}_2=(0,1)^T;\ \ \omega_2: \boldsymbol{x}_3=(1,0)^T, \boldsymbol{x}_4=(1,1)^T$$



详见课本77页例4.1
(图4.4有错，以左图为准)

分类界面如何迭代更新？

感知器算法

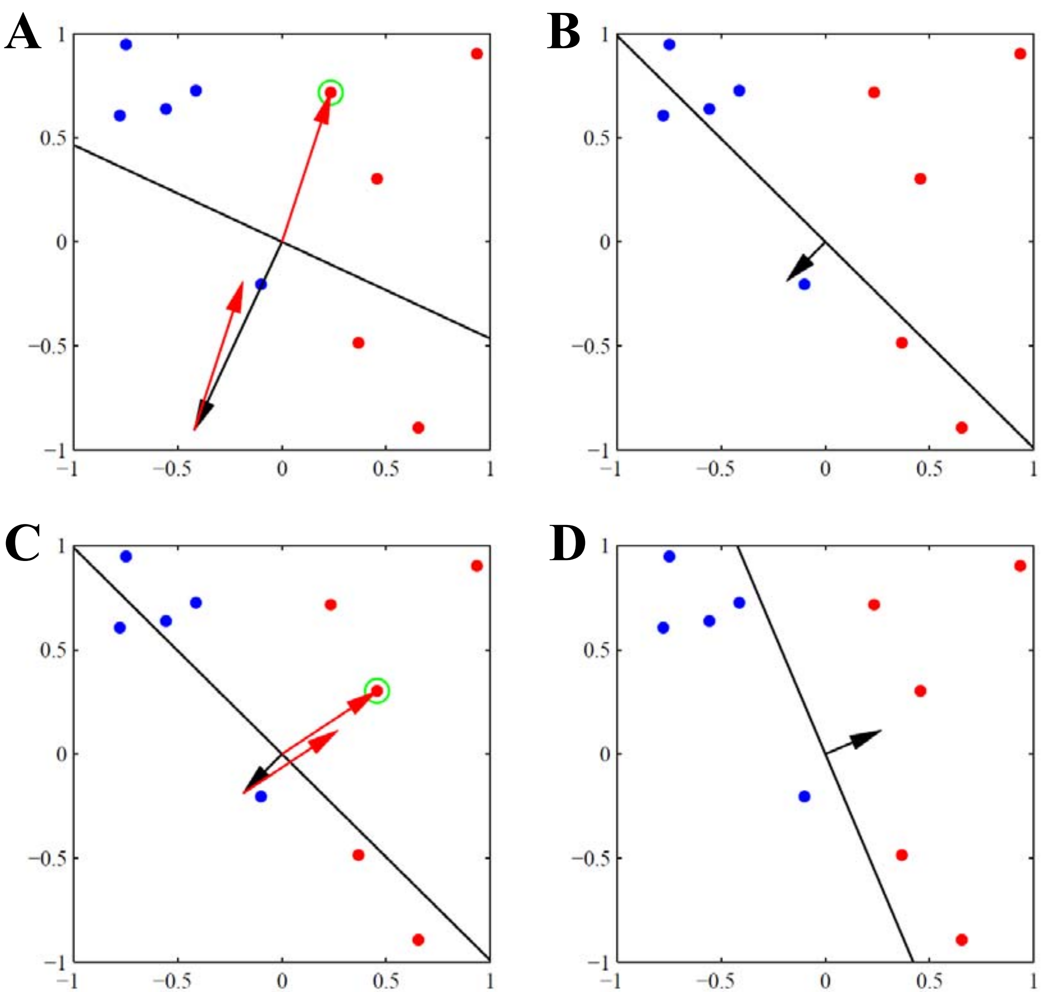
- 感知器算法（单样本调整版本）简单易行，收敛性好（证明略，见课本78-79页）。
- 收敛性的简单说明：

$$\begin{aligned} g_{k+1}(\boldsymbol{y}) &= \boldsymbol{a}_{k+1}^T \boldsymbol{y} \\ &= (\boldsymbol{a}_k + \boldsymbol{y})^T \boldsymbol{y} \\ &= \boldsymbol{a}_k^T \boldsymbol{y} + \|\boldsymbol{y}\|^2 \\ &> \boldsymbol{a}_k^T \boldsymbol{y} = g_k(\boldsymbol{y}) \end{aligned}$$

经过一次迭代后，关于 $\boldsymbol{y}$ 的判别函数值在增大；只要迭代次数足够多，可以使 $g_k(\boldsymbol{y}) > 0$ 成立。

感知器算法

- 感知器算法收敛性的说明



感知器用于分类二维特征空间中的两类的数据点（红色和蓝色）。

A. 给出初始权向量 ***a***（黑色箭头），以及对应的决策边界（黑色直线），其中箭头指向被分类为红色类别的决策区域。用绿色圆标出的数据点被误分类，因此它的特征向量（红色箭头）被加到当前的权向量中。

B. 更新后的参数权向量 ***a*** 和决策边界。

C. 一个误分类的点用绿色圆圈标出，它的特征向量再次被加到权向量上。

D. 再次更新后的决策边界，所有数据点都被正确分类。

感知器算法的特点

- 学习率的选择：当样本线性可分情况下，学习率合适时（如，单样本版本时$\eta(k)=1$），算法具有收敛性。
- 收敛速度：收敛速度和初始权重以及样本集合有关，一般较慢。
- 线性不可分的训练样本集：当样本线性不可分情况下，感知器算法不收敛，且无法判断样本是否线性可分。对于线性不可分的问题，最小平方误差法可以获得比较好的结果。

最小平方误差算法

- 如果对每一个样本给定一个正数 b_i，那么如下线性方程组同样满足不等式 $\boldsymbol{a}^T\boldsymbol{y}_i > 0,\ i=1,\cdots,n$

$$\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1,d+1} \\ y_{21} & y_{22} & \cdots & y_{2,d+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{n,d+1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{d+1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

其中矩阵每一行 $\boldsymbol{y}_i = [y_{i1}, y_{i2}, \cdots, y_{i,d+1}]^T$ 对应一个训练样本的规范化增广矢量，$\boldsymbol{a} = [a_1, a_2, \cdots, a_{d+1}]^T$ 为增广的权值矢量， $\boldsymbol{b} = [b_1, b_2, \cdots, b_n]^T$ 为常数矢量。

最小平方误差算法

- 上述线性方程组的矩阵形式：$\boldsymbol{Ya}=\boldsymbol{b}$
- 如果 $\boldsymbol{Y}$ 是可逆的方阵，则方程组解为：$\boldsymbol{a}=\boldsymbol{Y}^{-1}\boldsymbol{b}$
- 但$\boldsymbol{Y}$一般不是方阵，不存在逆矩阵。而且，一般地，$n>(d+1)$，所以该线性方程组往往是超定方程组，不存在精确解。
- 求近似解时，定义误差矢量 $\boldsymbol{e}=\boldsymbol{Ya}-\boldsymbol{b}$，当 $\boldsymbol{e}$ 为0矢量时方程有精确解，$\boldsymbol{e}$ 越小则近似精度越高。
- 以误差矢量长度的平方作为优化准则函数：

$$J_s(\boldsymbol{a})=\|\boldsymbol{e}\|^2=\|\boldsymbol{Ya}-\boldsymbol{b}\|^2$$

最小平方误差算法

- 线性判别函数的学习变为对上述最小平方（least mean squares，LMS）误差的优化：

$$\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}} J_s(\boldsymbol{a})$$

- 由于 $J_s(\boldsymbol{a}) = \|\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{b}\|^2 = (\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{b})^T(\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{b})$

$$= \boldsymbol{a}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{a} - 2\boldsymbol{a}^T\boldsymbol{Y}^T\boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b}$$

  因此 $\nabla J_s(\boldsymbol{a}) = \dfrac{dJ_s(\boldsymbol{a})}{d\boldsymbol{a}} = 2\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{a} - 2\boldsymbol{Y}^T\boldsymbol{b}$

- 可以解得： $\boldsymbol{a} = (\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{Y}^T\boldsymbol{b} = \boldsymbol{Y}^+\boldsymbol{b}$

  其中 $\boldsymbol{Y}^+ = (\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{Y}^T$ 称为 $\boldsymbol{Y}$ 的伪逆矩阵，$\boldsymbol{a} = \boldsymbol{Y}^+\boldsymbol{b}$ 称为伪拟解。

最小平方误差算法

- 常数矢量 $\boldsymbol{b}$ 的设置

  很难预先设置或者估计 $\boldsymbol{b}$；实际算法中，可以设置 $\boldsymbol{b}=1$，对每个样本相等。

- 伪逆矩阵的存在性和计算

  伪逆矩阵需要计算 $(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}$，但是 $\boldsymbol{Y}^T\boldsymbol{Y}$ 并不一定可逆。当特征维度 d 大于等于样本数 n 时，矩阵不可逆。可以引入正则项计算伪逆矩阵：

  $$\boldsymbol{Y}^+=(\boldsymbol{Y}^T\boldsymbol{Y}+\varepsilon\boldsymbol{I})^{-1}\boldsymbol{Y}^T$$

  其中，$\boldsymbol{I}$ 是单位矩阵，$\varepsilon>0$ 且充分小。

最小平方误差算法

- 最小平方误差准则函数也可以用梯度法进行迭代优化：

$$\nabla J_s(\boldsymbol{a}) = 2\boldsymbol{Y}^T(\boldsymbol{Ya}-\boldsymbol{b}) = 2\sum_{i=1}^{n}(\boldsymbol{a}^T\boldsymbol{y}_i - b_i)\boldsymbol{y}_i$$

- 可得最小平方误差算法的迭代优化公式：

$$\boldsymbol{a}(k+1) = \boldsymbol{a}(k) - \eta(k)\boldsymbol{Y}^T(\boldsymbol{Ya}(k)-\boldsymbol{b})$$

- 迭代算法可以避免复杂度高的矩阵求逆运算（特别是特征维数很大的时候）。
- 迭代法可以每次输入一个样本，调整权重：

$$\boldsymbol{a}(k+1) = \boldsymbol{a}(k) - \eta(k)[\boldsymbol{a}^T(k)\boldsymbol{y}_i - b_i]\boldsymbol{y}_i$$

最小平方误差算法

- Widrow-Hoff算法

- 初始化：$\boldsymbol{a}(0)$，$\boldsymbol{b}$，学习率 $\eta(k)$，收敛精度 θ，$k=0$；
- do $k=k+1$

$$i=(k+1)\bmod n$$

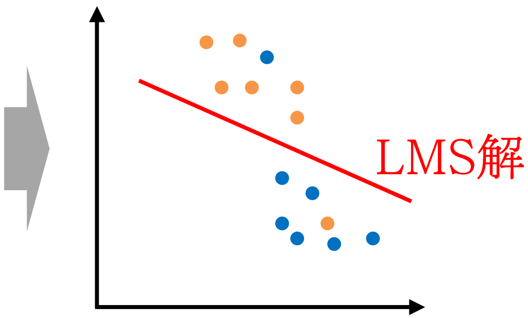
$$\boldsymbol{a}(k+1)=\boldsymbol{a}(k)-\eta(k)[\boldsymbol{a}^T(k)\boldsymbol{y}_i-b_i]\boldsymbol{y}_i$$

until $|\eta(k)[\boldsymbol{a}^T(k)\boldsymbol{y}_i-b_i]\boldsymbol{y}_i|<\theta$

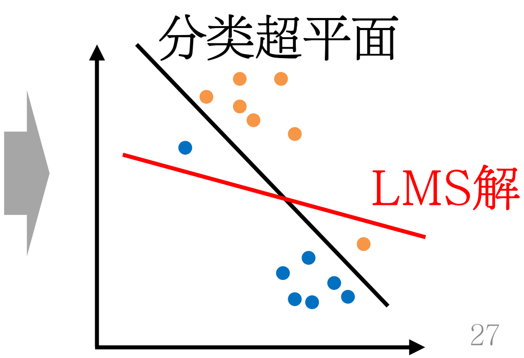
- 输出：$\boldsymbol{a}$

最小平方误差算法的特点

- 算法的收敛依靠适当的学习率 $\eta(k)$。
- 取 $\boldsymbol{b} = 1$ 时，当样本数趋于无穷多时，算法的解以最小均方误差逼近贝叶斯判别函数（略）。
- 对于线性不可分的训练样本，算法能够收敛于一个均方误差最小解；

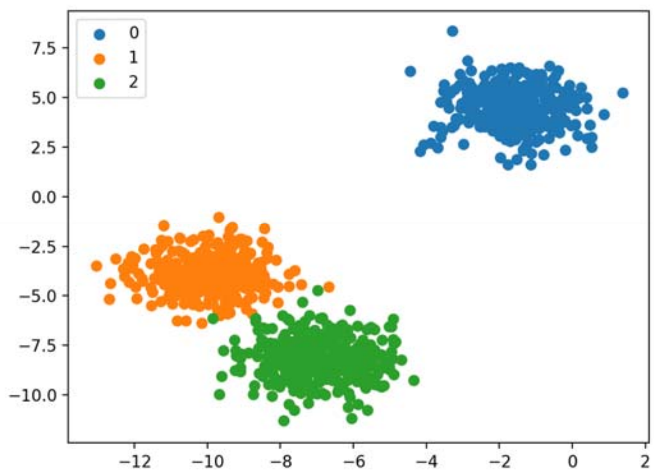


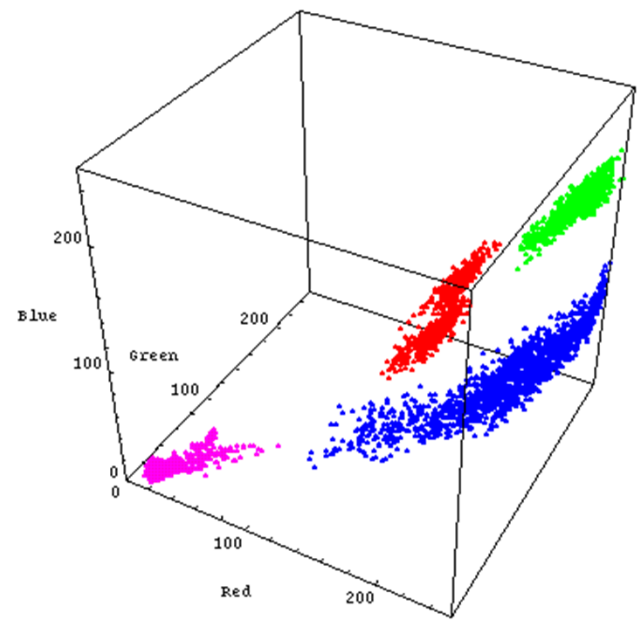
- 但对于线性可分的训练样本，算法未必能够收敛于一个分类超平面。



多类别问题

- 前面介绍的线性判别函数分类器只可以解决两分类问题。当样本有多个类别时，可以将多类别问题转换为两类别问题。
  - 一对多
  - 一对一
  - 扩展的感知器





多类别问题

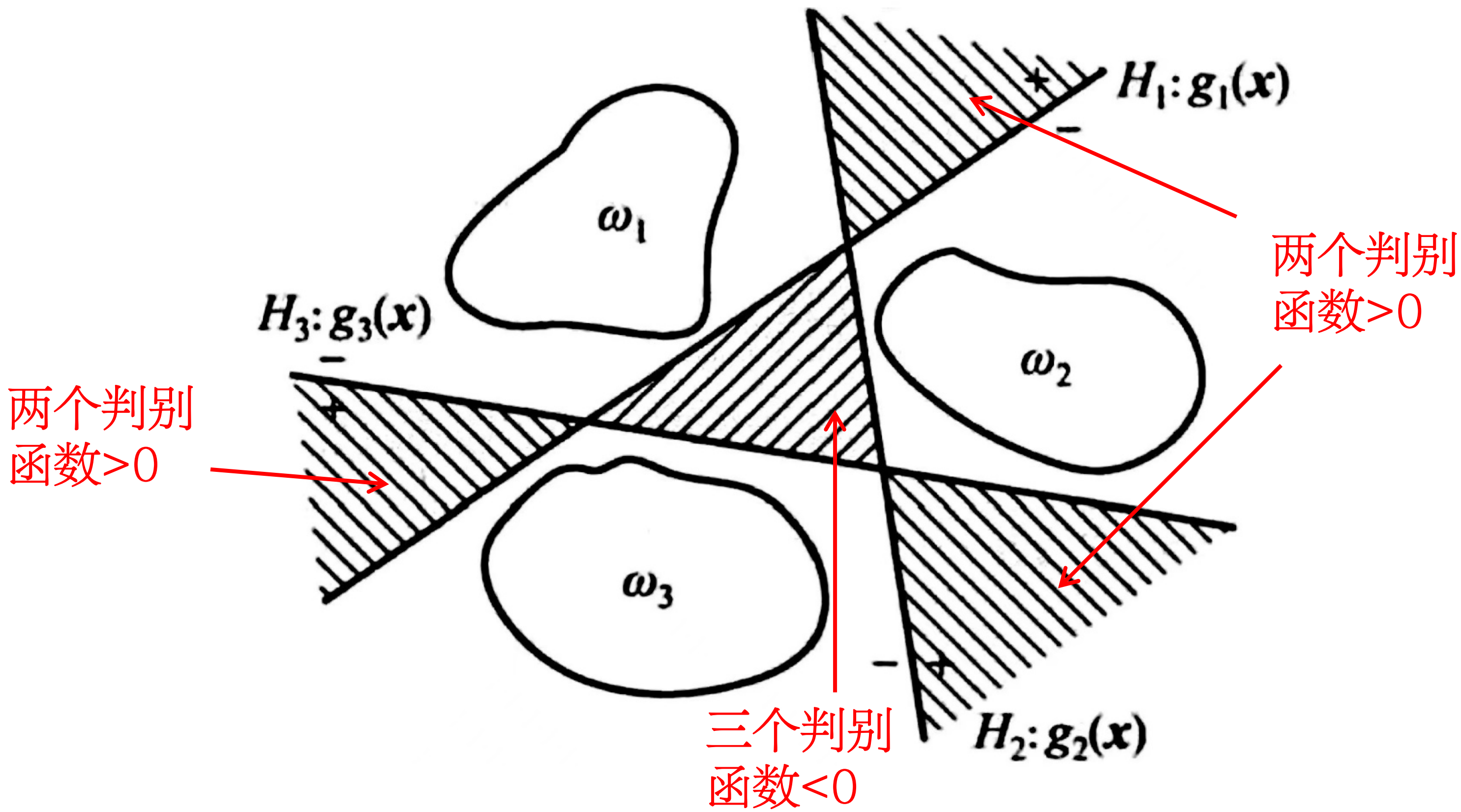
- 一对多：如果样本属于 c 个类别，可以学习 c 个线性判别函数分类器，其中第 i 个分类器将第 i 类的样本同其他类别的样本区分开。

- 一对多方式的判别规则为：

$$\begin{cases} \boldsymbol{x} \in \omega_i, \ g_i(\boldsymbol{x}) > 0, \ g_j(\boldsymbol{x}) < 0, \forall j \neq i \\ \text{拒识，} \quad \text{其他} \end{cases}$$

多类别问题

- 当 c 个判别函数中只有一个大于0，其他都小于0的时候，待识别样本属于该类别；其他情况则无法识别。



多类别问题

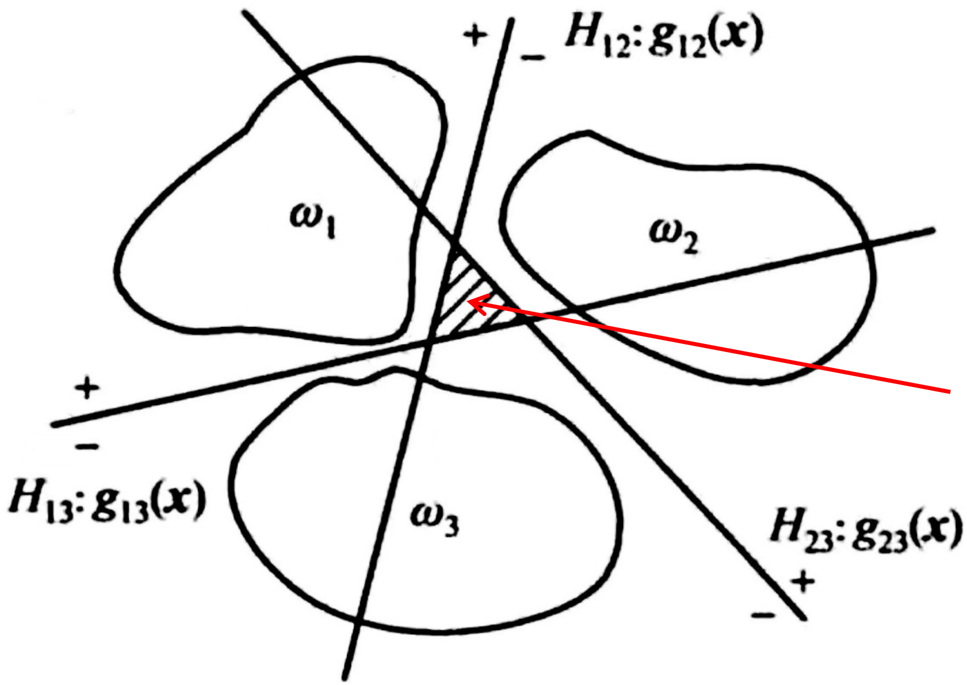
- 一对一：如果样本属于 c 个类别，可以由第 i 类的样本和第 j 类的样本学习得到区分这两类的判别函数 $g_{ij}(\boldsymbol{x})$，共需要 $c(c-1)/2$个分类界面将任意两类分开。

- 一对一方式的判别规则为：

$$\begin{cases} \boldsymbol{x} \in \omega_i, \ g_{ij}(\boldsymbol{x}) > 0, \ \forall j \neq i \\ \text{拒识，其他} \end{cases}$$

多类别问题

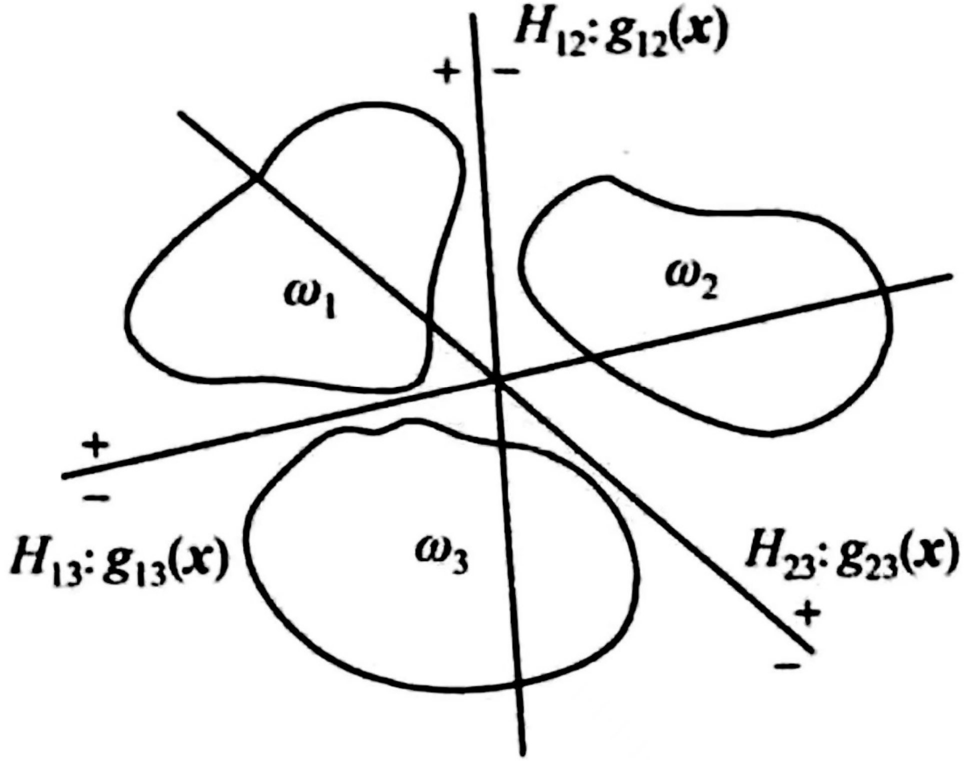
- 如与第 i 类相关的 $c-1$ 个判别函数都大于0时，判别待识别样本属于此类别，否则无法判别。
- 注意： $g_{ij}(\boldsymbol{x}) = -g_{ji}(\boldsymbol{x})$，两者对应同一个分类界面但正负方向相反。



对任一类，一个判别函数>0，另一个<0，无法判别

多类别问题

- 上页图中，可以调整分类界面位置以消除类别不确定区域：



多类别问题

- 在此方式中，需要建立 c 个判别函数分别对应每个类别，判别规则是将样本分到对应判别函数值最大的类别：

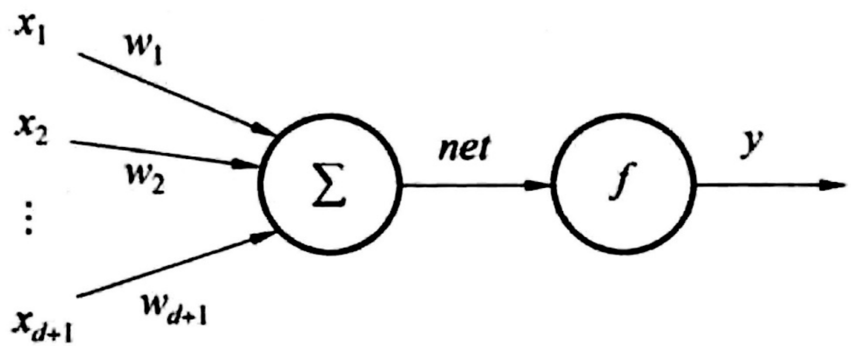
$$i = \arg\max_{1 \le j \le c} g_j(\boldsymbol{x})$$

或，如果 $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x})$，$\forall j \neq i$，则判别 $\boldsymbol{x} \in \omega_i$。

- 此种方式判别函数的学习需要一种特殊的扩展感知器算法（略）。

感知器网络

- 线性判别函数分类器同人工神经元有密切联系。
- 人工神经元模型：



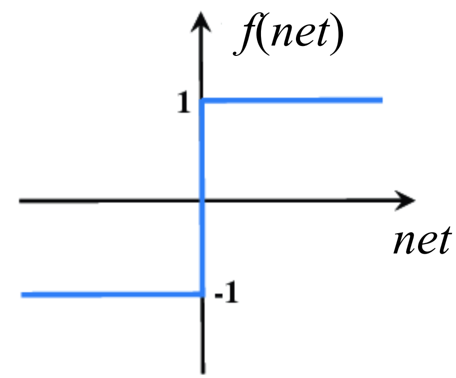
每个神经元有多个输入 $x_1, \cdots, x_{d+1}$和一个输出 y，输入由神经元权值 $w_1, \cdots, w_{d+1}$ 加权求和产生净输入 net，然后由激活函数 f 映射产生输出 y。

$$net = \sum_{i=1}^{d+1} x_i w_i = \boldsymbol{w}^T \boldsymbol{x}, \; y = f(net) = f(\boldsymbol{w}^T \boldsymbol{x})$$

感知器网络

- 当激活函数 f 为符号函数时，神经元模型等价于一个线性判别函数分类器：

$$f(net)=\begin{cases}-1, & net<0\\ +1, & net\geq 0\end{cases}$$

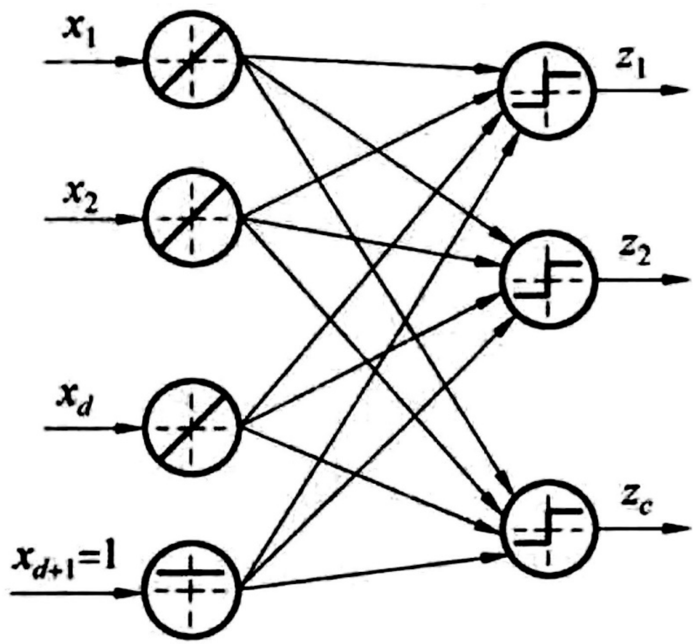


其中输入 $\boldsymbol{x}$ 为增广的特征矢量，$\boldsymbol{w}$ 为判别函数的权值矢量和偏置，根据输出 ± 1判断类别属性。

- 神经元的经典算法即是感知器。
- 单独一个神经元只可以实现二分类，实现多分类需要多个神经元构成两层感知器网络。

感知器网络

- 两层感知器网络由输入层和输出层构成：
  - 输入层包含 $d+1$ 个神经元，激活函数为线性函数 $f(x)=x$，仅起传递作用；
  - 输出层可根据设计采用符号函数或线性函数。



输入层　　输出层

感知器网络

- 输出可用直接编码、最大值和2进制编码等方式确定类别属性。
- 直接编码：输出层的 c 个神经元对应 c 个类别，采用符号函数作为激活函数。当其中某个神经元的输出为 $+1$ 而其他输出为 -1 时，判别 y 属于输出 $+1$ 神经元对应的类别。
  - 例如4个类别：第1类 $(+1,-1,-1,-1)$

    第2类 $(-1,+1,-1,-1)$

    第3类 $(-1,-1,+1,-1)$

    第4类 $(-1,-1,-1,+1)$

感知器网络

- 最大值：输出层的 c 个神经元对应 c 个类别，但是采用线性函数作为激活函数，以输出值最大的神经元判定输入 $\mathbf{y}$ 所属的类别：

$$\mathbf{y} \in \omega_i, i = \arg\max_{1 \le j \le c} z_j$$

- 2进制编码：以输出层的输出作为相应位的编码值来确定输入的类别属性
  - 例如4类： 第1类 $(-1,-1)$；第2类 $(-1,+1)$
    第3类 $(+1,-1)$；第4类 $(+1,+1)$

感知器网络

- 直接编码方式只有当一个神经元的输出为+1，其他神经元的输出为-1时才能够判定输入的类别属性，如果有多个神经元输出+1或所有神经元都输出-1，则无法判别。
- 最大值方式是一种比较常用的方式，一方面它可以对任意样本判定其类别，不会出现无法识别的情况，另一方面对类别的数量没有特殊的要求。
- 2 进制编码方式对任意的输入都可以判别其类别属性，但一般要求类别数 c 是 2 的幂次。

感知器网络

- 两层感知器网络的学习一般是采用最小平方误差法，目的是调整输出层神经元的权值，使得网络能够正确识别训练样本。
- 类似地，两层感知器网络也可以采用梯度法迭代求解。

详见课本92-93页

线性分类器小结

- 本章主要介绍感知器和最小平方误差算法。感知器只在线性可分情况下保证收敛，最小平方误差算法的适应性更好。
- 多数实际问题是线性不可分的，但线性判别分类器函数是非线性判别分类器函数的基础。
- 线性分类器算法以寻找一个可将训练样本分开的线性分类界面为目的建立优化函数，然后用优化方法求解分类器参数。
- 能将训练样本分开的线性分类界面可能很多，是否存在最优界面？以上问题在第6章介绍。

本章小结

- 介绍了线性判别函数的基本原理
- 介绍了感知器准则和算法，以及算法的问题
- 介绍了最小平方误差算法
- 介绍了将线性判别函数分类器用于多类别问题的几种方式