模式识别

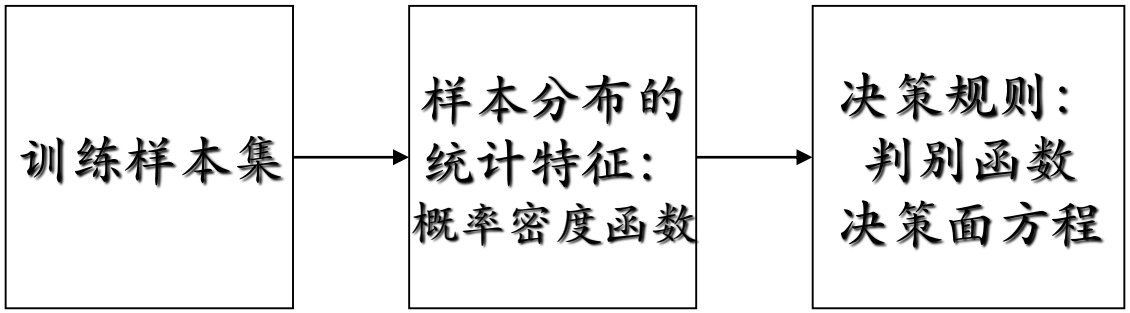
第四章 线性判别函数

内容

* 引言
* 线性判别函数的基本概念
* Fisher线性判别函数
* 感知准则函数
* 最小平方误差准则函数
* 多类问题

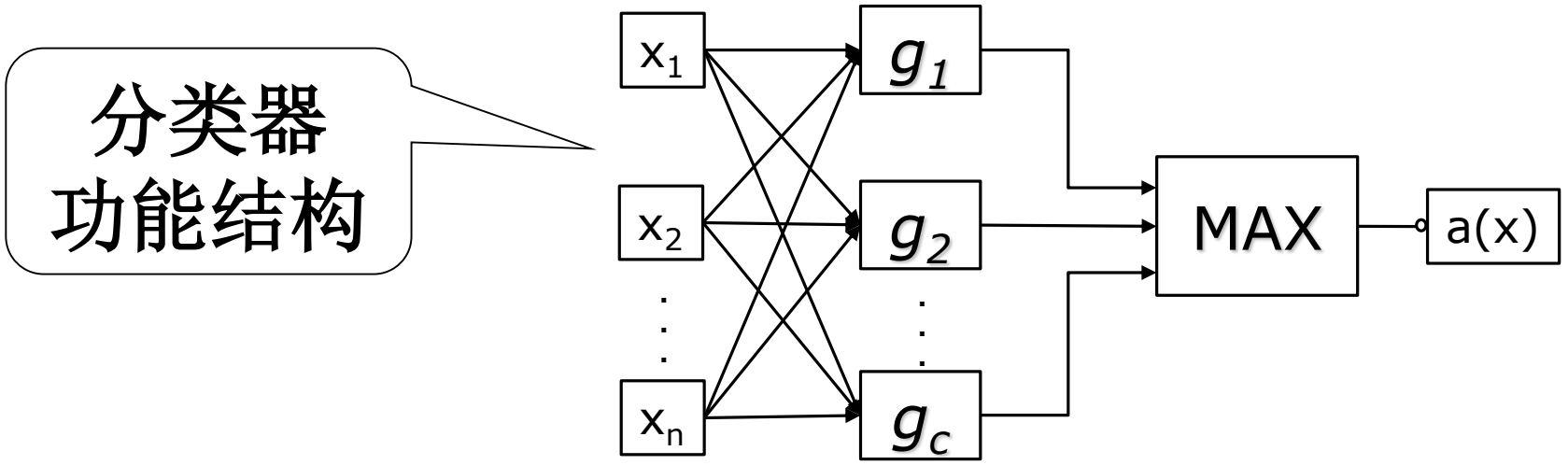
引言

* 第三章主要讲了类条件概率密度函数的估计
  * 参数估计方法
    * 最大似然估计
    * 贝叶斯估计
  * 非参数估计方法



引言

* 基于样本的Bayes分类器：通过估计类条件概率密度函数，设计相应的判别函数



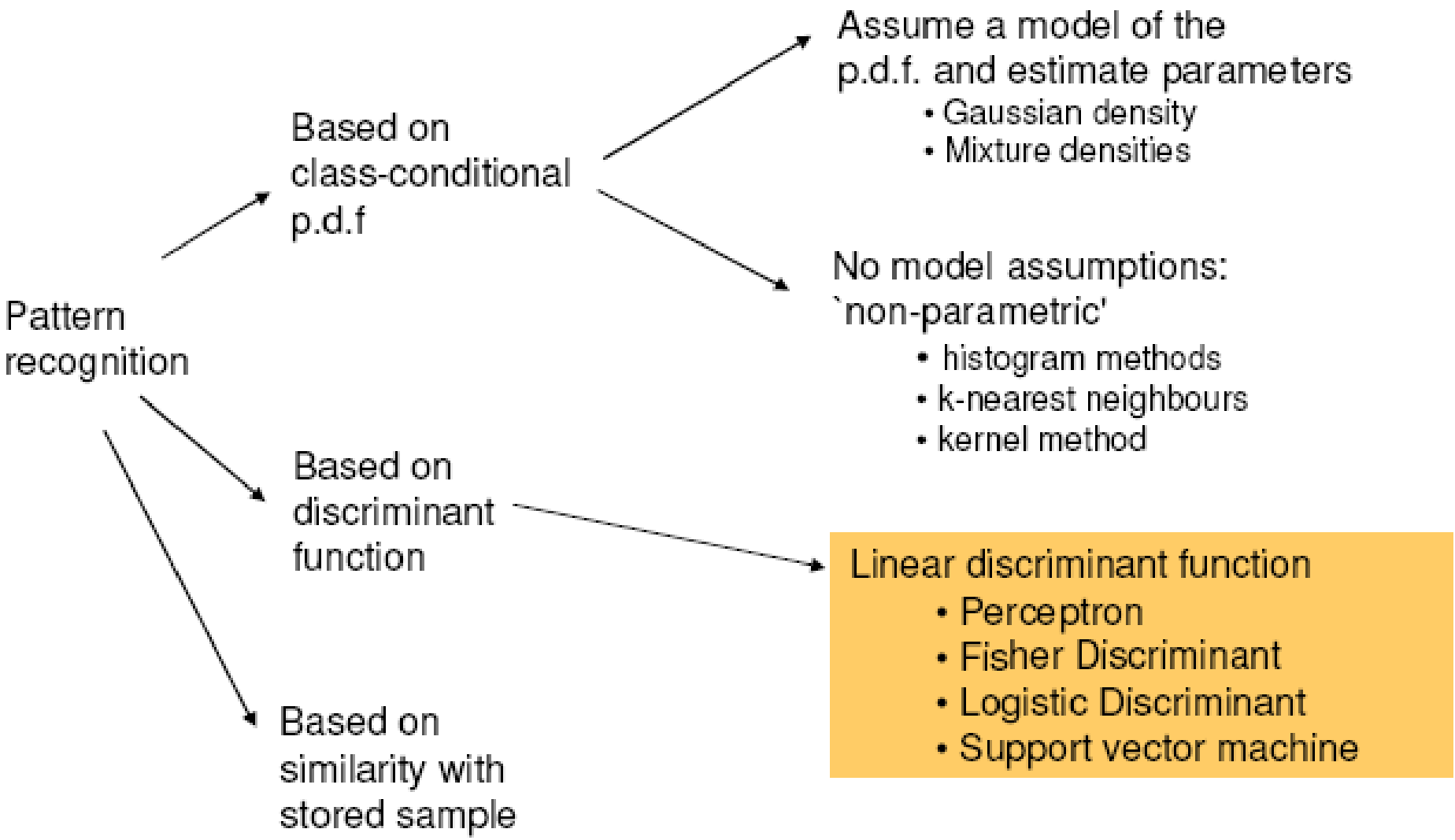
引言

* 如果可以估计概率密度函数，则可以使用贝叶斯决策来最优的实现分类
* 最一般情况下适用的“最优”分类器：错误率最小，对分类器设计在理论上有指导意义。
* 获取统计分布及其参数很困难，实际问题中并不一定具备获取准确统计分布的条件。
  * 特征空间维度高
  * 样本较少
* 如果估计不了呢?

引言

* 模式识别还有更多方法!
* 注意：模式识别（分类）的最终目的是在特征空间中找到分类面!
* 从样本数据中直接估计参数->根据对样本/问题的理解直接设定判别函数形式，直接求解!
  * 线性
  * 非线性

Linear discriminant functions

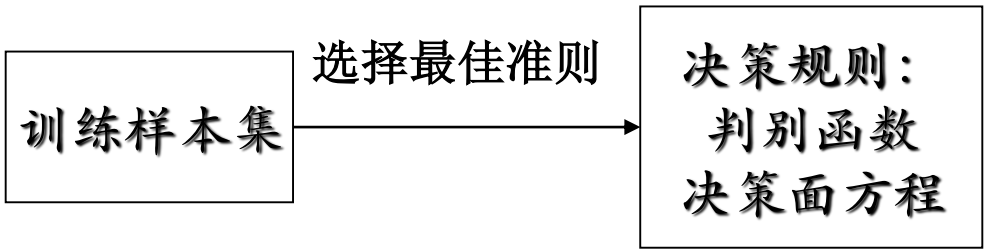


引言

- Parametric Methods
  - Underlying pdfs are known
  - Training samples used to estimate pdf parameters
- Linear Discriminant Functions
  - Forms of discriminant functions are known
  - Similar to non-parametric techniques
  - Sub-optimal, but simple to use

引言

* 基于样本的直接确定判别函数方法：
  * 针对各种不同的情况，使用不同的准则函数，设计出满足这些不同准则要求的分类器。
  * 这些准则的“最优”并不一定与错误率最小相一致：次优分类器。
  * 实例：正态分布最小错误率贝叶斯分类器在特殊情况下，是线性判别函数$\mathbf{g}(\mathbf{x}) = \boldsymbol{w}^T\boldsymbol{x}$（决策面是超平面），能否基于样本直接确定$\boldsymbol{w}$?



线性判别函数的基本概念

* d维空间中的线性判别函数的一般形式：

$$\mathbf{g}(\mathbf{x}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{w}_{\boldsymbol{0}}$$

* 其中：

x是样本向量-样本在d维特征空间中的描述，$\boldsymbol{w}$是权向量，$\boldsymbol{w}_{\boldsymbol{0}}$是一个常数(阈值权)。

$$\boldsymbol{x} = [\boldsymbol{x}_{\boldsymbol{1}}, \boldsymbol{x}_{\boldsymbol{2}}, \dots, \boldsymbol{x}_{\boldsymbol{d}}]^{\boldsymbol{T}}$$

$$\boldsymbol{w} = [\boldsymbol{w}_{\boldsymbol{1}}, \boldsymbol{w}_{\boldsymbol{2}}, \dots, \boldsymbol{w}_{\boldsymbol{d}}]^{\boldsymbol{T}}$$

线性判别函数的基本概念

* 两类问题的分类决策规则：
* 令 $\mathbf{g(x)} = \mathbf{g}(x_1) - \mathbf{g}(x_2)$

$$\text{如果}\begin{cases} g(\mathbf{x})>0, & \text{则决策}\mathbf{x}\in\omega_1 \\ g(\mathbf{x})<0, & \text{则决策}\mathbf{x}\in\omega_2 \\ g(\mathbf{x})=0, & \text{可将其任意分类或拒绝} \end{cases}$$

线性判别函数的基本概念

$$\mathbf{g(x) = 0}$$

* 定义了一个决策面，分开了两类
* 如果g(x)是线性的，则决策面是一个超平面
* 如果x_1和x_2都在决策面上，则

$$\boldsymbol{w^T x_1 + w_0 = w^T x_2 + w_0}$$

* 或

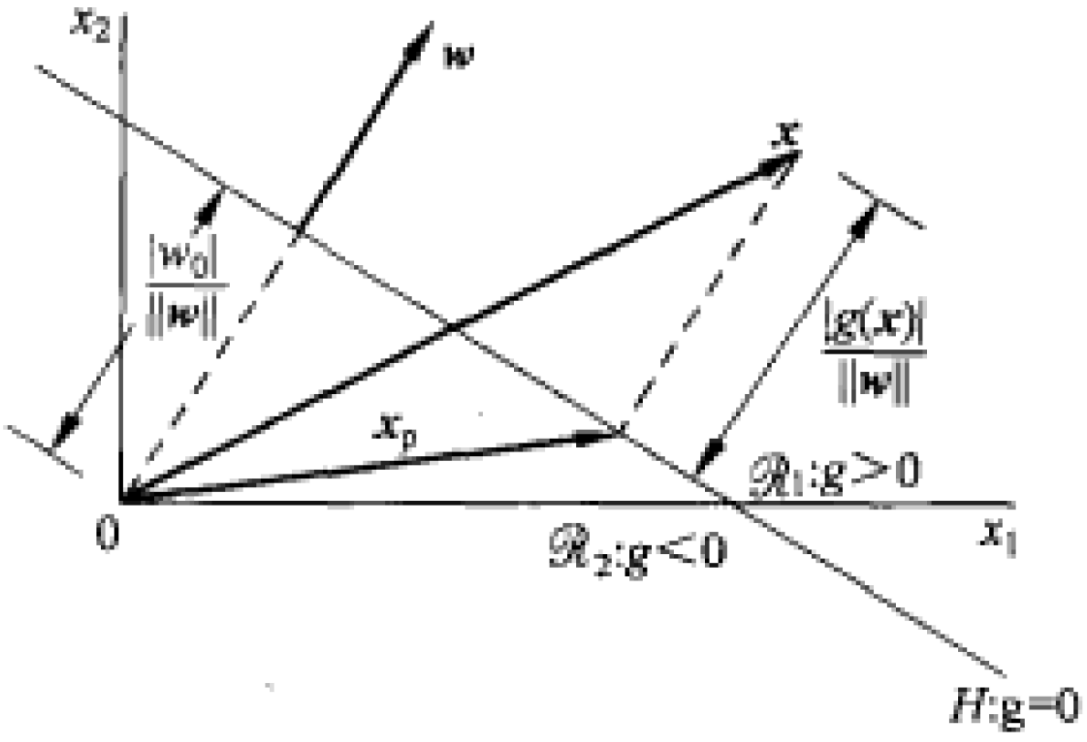
$$\boldsymbol{w^T(x_1 - x_2) = 0}$$

线性判别函数的基本概念

* 说明w与超平面上任一向量正交 - w是H的法向量

一般来说H将空间分成两个决策域

正侧 vs 负侧



线性判别函数的基本概念

* 判别函数g(x)也可以看作是特征空间中某个点x到超平面距离的一种代数度量
* 把x表示为

$$\mathbf{x} = x_p + r\frac{\boldsymbol{w}}{||\boldsymbol{w}||}$$

x_p是x到H的投影向量

r是x到H的垂直距离

$\frac{\boldsymbol{w}}{||\boldsymbol{w}||}$是w方向上的向量

线性判别函数的基本概念

$$\mathbf{g(x)} = \boldsymbol{w}^{\boldsymbol{T}}\left(\boldsymbol{x}_{\boldsymbol{p}} + \boldsymbol{r}\frac{\boldsymbol{w}}{||\boldsymbol{w}||}\right) + \boldsymbol{\omega_0}$$

$$= \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{p}} + \boldsymbol{\omega_0} + \boldsymbol{r}\frac{\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}}{||\boldsymbol{w}||} = \boldsymbol{r}||\boldsymbol{w}||$$

* 上式也可写为$\boldsymbol{r} = \frac{\mathbf{g(x)}}{||\boldsymbol{w}||}$

线性判别函数的基本概念

* 特殊情况 – 若x为原点

$$g(x) = \omega_0$$

* 则，原点到超平面的距离为

$$r_0 = \frac{\omega_0}{||w||}$$

* $\omega_0 > 0$ – H在原点的负侧，原点在?
* $\omega_0 < 0$ – H在原点的正侧，原点在?
* $\omega_0 = 0$ – 超平面通过原点

线性判别函数的基本概念

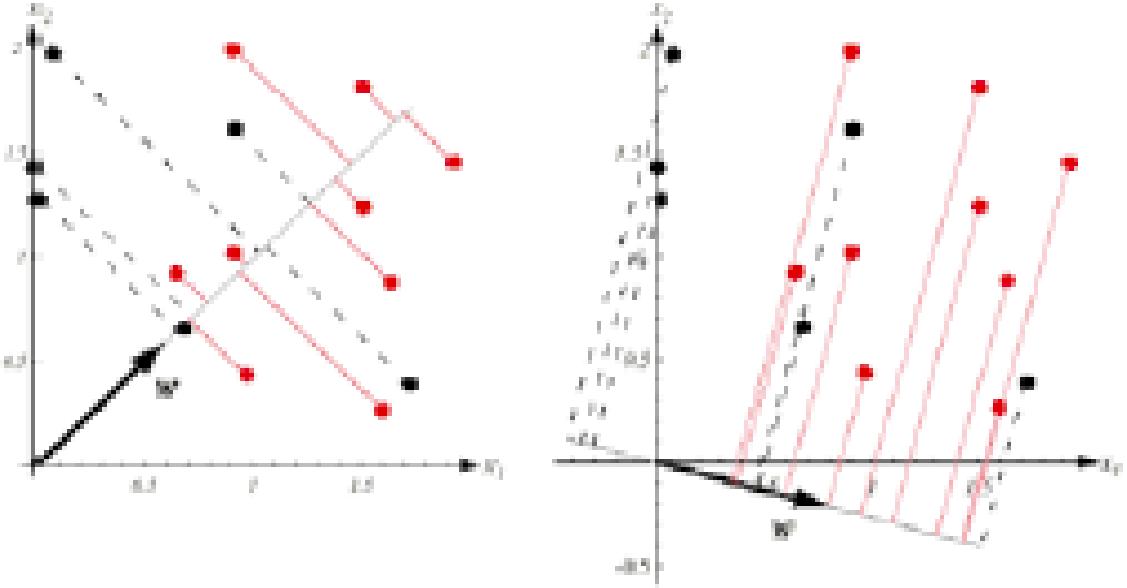
* 设计线性分类器的主要步骤:
  * 有一组具有类别标志的样本集
  * 根据实际情况确定一个准则函数J,满足:
    * J是样本集和w,w_0,a的函数
    * J的值能反映分类器的性能，它的极值解对应于“最好”的决策
  * 利用最优化方法求出准则函数的极值解和w,w_0,a，进而得到$g(x)$

Fisher线性判别分析

* 又叫Linear discriminant Analysis (LDA)
* 1936年的方法
* R. A. Fisher所提出
* 沿用至今

Fisher线性判别分析

- The Fisher's approach projects d-dimensional data onto a line, *w*
- It is expected that these projections are well separated by class



Fisher线性判别分析

* Fisher线性判别的基本思想：

  希望投影后的一维数据满足

  两类之间的·距离尽可能远

  每一类自身尽可能紧凑

如何实现？如何写公式？

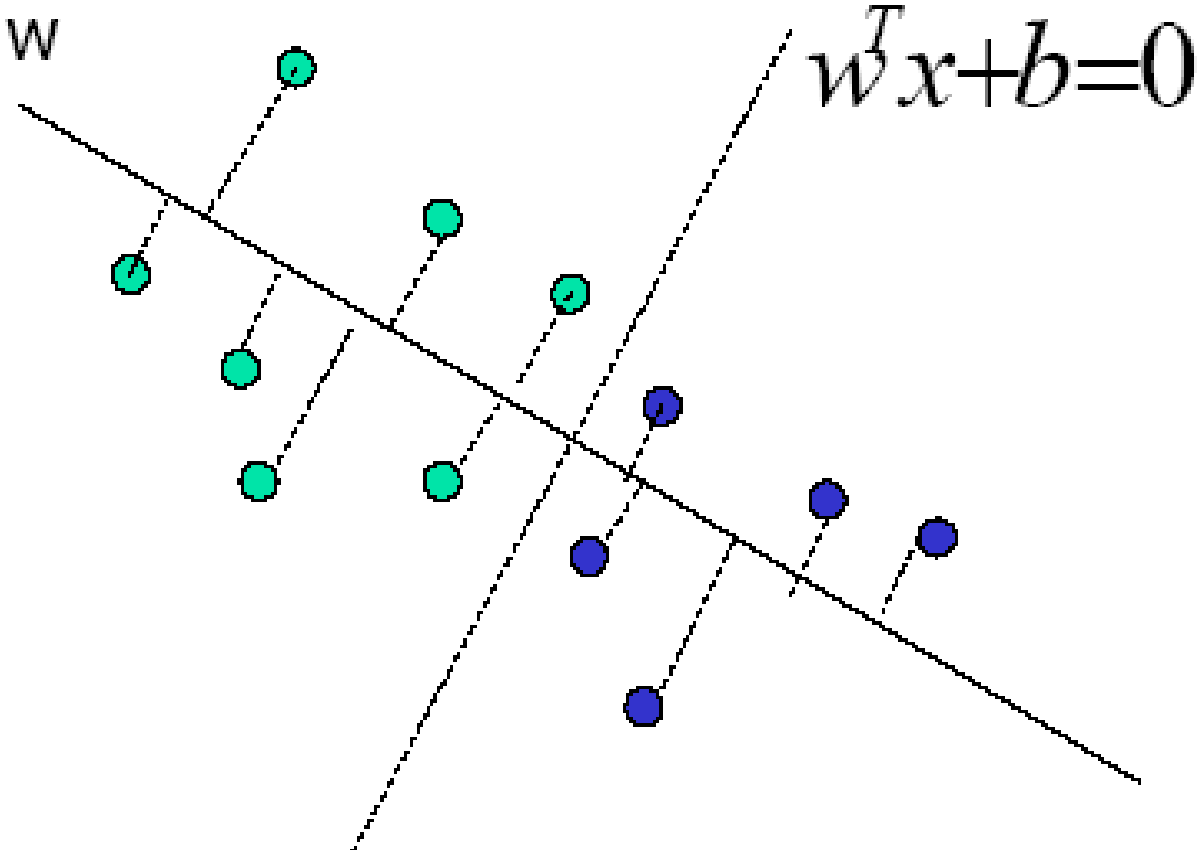
Fisher线性判别分析

准则的描述

用投影后数据的统计性质（均值和离散度）的函数作为判别优劣的标准

Fisher线性判别分析

- 线性判别函数的几何意义：



Fisher线性判别分析

符号含义：

- m_1, m_2 ——两类（原始）数据的均值向量。
- S_1, S_2 分别表示两类（原始）数据的离散度矩阵。
- μ_1, μ_2 分别表示两类（投影后，一维）数据的均值。
- σ_1, σ_2 分别表示两类（投影后，一维）数据的离散度。

Fisher线性判别分析

- 均值向量和离散度矩阵

$$m_i = \frac{1}{N}\sum x \qquad i = 1,2$$

$$S_i = \sum\left((x - m_i)(x - m_i)^T\right) \qquad i = 1,2$$

Fisher线性判别分析

原始数据与做 w 方向投影后数据统计量之间的关系：

$$\mu_i = w^T m_i, \qquad i = 1,2.$$

$$\sigma_i^2 = \sum (w^T x - \mu_i)^2$$

$$= w^T \sum (x - m_i)(x - m_i)^T w$$

$$= w^T S_i w.$$

Fisher线性判别分析

- Fisher准则函数：

$$J_F(w)=\frac{(\mu_1-\mu_2)^2}{\sigma_1^{\ 2}+\sigma_2^{\ 2}}.$$

类间距

总类内离散度

$$w_{opt}=\arg\max J_F(w).$$

Fisher线性判别分析

- 称 $S_b = (m_1 - m_2)(m_1 - m_2)^T$

  类间离散度矩阵。

- 称 $S_t = S_1 + S_2$ 类内总离散度矩阵。

$$J_F(w) = \frac{w^T S_b w}{w^T S_t w}.$$

Fisher线性判别分析

- Fisher准则的合理性：

$J_F(w)$只与投影方向有关，与w大小无关—若w是一个最优解，kw也是最优解，k是任何不为零的常数。

Fisher线性判别分析

- Fisher最佳投影方向的求解：
  - 要求：

$$S_t = S_1 + S_2 \quad \text{正定。}$$

否则，存在投影方向 w ，使得

$$w^T S_t w = 0. \quad \text{所有数据被投影到一点上！}$$

$J_F(w)$ 没有极大值。

Fisher线性判别分析

- 求出最佳投影方向上任何一个 w 即可。

- $J_F(w)$有上界，最佳投影方向一定存在！

$$J_F(w) \le \frac{\lambda(S_b)_{\max}}{\lambda(S_t)_{\min}}.$$

$\lambda(S_w)_{\min}, \lambda(S_b)_{\max}$ 分别是矩阵 S_t, S_b 的最小、最大的特征根。

Fisher线性判别分析

- 一定存在一个最优的 w ，满足：

$$w^T S_t w = 1.$$ 因为 S_t 正定！

- 无约束最优化：$\max \dfrac{w^T S_b w}{w^T S_t w}.$

等价于带约束的最优化：

$$\max w^T S_b w$$

$$s.t. \quad w^T S_t w = 1.$$

Fisher线性判别分析

- 带等式约束的最优化，用Lagrange乘子法：

$$L(w,\lambda) = w^T S_b w - \lambda(w^T S_t w - 1).$$

$$\frac{\partial L(w,\lambda)}{\partial w} = S_b w - \lambda S_t w.$$

最优解满足： $S_b w_{opt} - \lambda S_t w_{opt} = 0.$

Fisher线性判别分析

根据类间离散度定义：

$$(m_1 - m_2)(m_1 - m_2)^T w_{opt} = \lambda S_t w_{opt}.$$

注意 $(m_1 - m_2)^T w_{opt}$ 是一个数，记作 c ，

$$w_{opt} = \frac{c}{\lambda} S_t^{-1}(m_1 - m_2).$$

Fisher线性判别分析

- 只关心投影的方向：

$$w_{opt} = S_t^{-1}(m_1 - m_2)$$

$$= (S_1 + S_2)^{-1}(m_1 - m_2).$$

Fisher线性判别分析

* 至此，我们还没有解决分类问题，只是将d维映射到1维，将d维分类问题转划为1维分类问题，如何分类?

* 确定阈值

Fisher线性判别分析

- 分类阈值 b 的确定：
  - 两类均值的中点：

$$b = \frac{\mu_1 + \mu_2}{2}.$$

  - 投影后数据的均值（n_1, n_2是两类样本的个数）

$$b = \frac{n_1\mu_1 + n_2\mu_2}{n_1 + n_2}.$$

感知准则函数

* Perceptron
* 感知准则函数是五十年代由Rosenblatt提出的一种自学习判别函数生成方法，由于Rosenblatt企图将其用于脑模型感知器，因此被称为感知准则函数。其特点是随意确定的判别函数初始值，在对样本分类训练过程中逐步修正直至最终确定。

感知准则函数-基本概念

* 为了讨论方便，把向量x增加一维，为

$$y = [1, x_1, x_2, \dots, x_d]^T$$

* y为增广的样本向量
* 增广的权向量为

$$a = [w_0, w_1, w_2, \dots, w_d]^T$$

* 线性判别函数变为

$$g(y) = a^T y$$

感知准则函数-基本概念

* 线性可分性：训练样本集中的两类样本在特征空间可以用一个线性分界面正确无误地分开。在线性可分条件下，对合适的(广义)权向量a应有：

$$\text{如果 } \mathbf{y} \in \omega_1, \text{则} \mathbf{a}^T\mathbf{y} > 0$$

$$\text{如果 } \mathbf{y} \in \omega_2, \text{则} \mathbf{a}^T\mathbf{y} < 0$$

感知准则函数

* 线性可分:
  * 模式分类如可用任一个线性函数来划分，则这些模式就称为线性可分的，否则就是非线性可分的。
  * 一旦线性函数的系数w_k被确定，这些函数就可用作模式分类的基础。

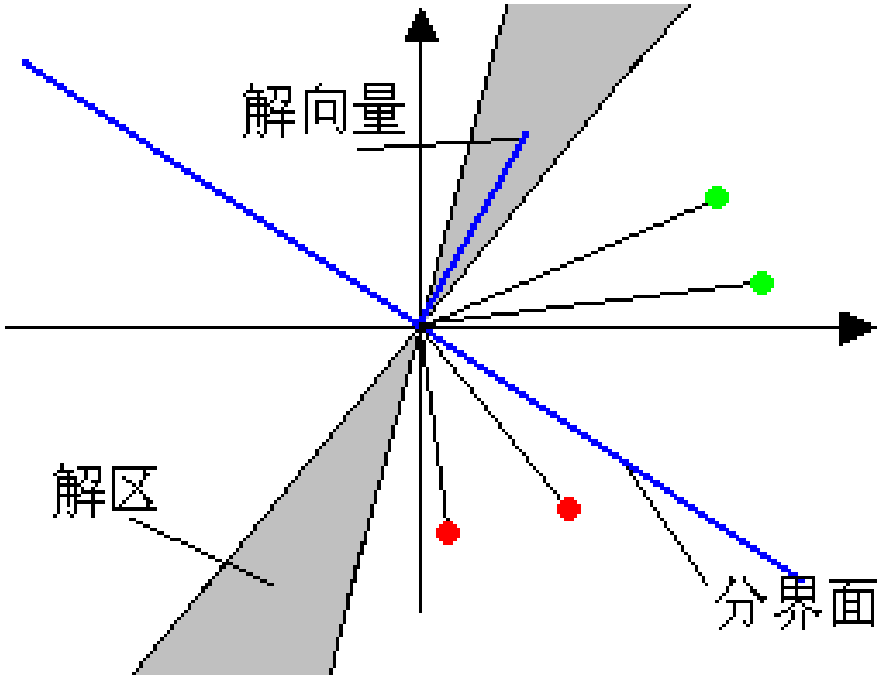
感知准则函数-基本概念

* 规范化样本向量：将第二类样本取其反向向量

$$\mathbf{y}'=\begin{cases}\mathbf{y} & \text{如果 } \mathbf{y}\in\omega_1\\ -\mathbf{y} & \text{如果 } \mathbf{y}\in\omega_2\end{cases}$$

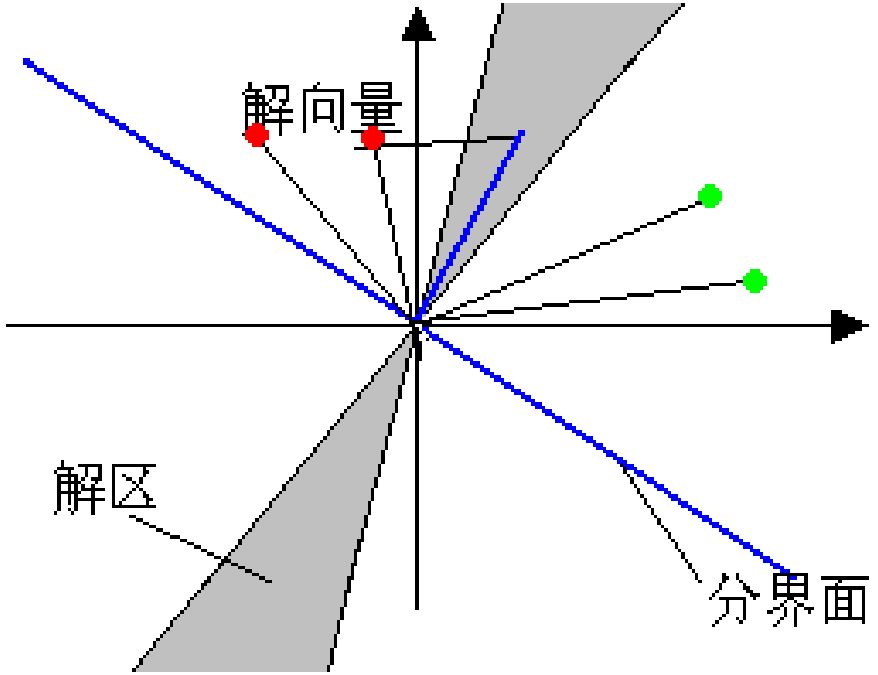
$$\Rightarrow \quad \mathbf{a}^T\mathbf{y}_i'>0 \quad i=1,\ldots,N$$

感知准则函数-解向量与解区



a: 第一类样本

(a) 未规范化



b: 第二类样本

(b) 规范化

感知准则函数

* 实际中直接用下式会有什么问题么？

$$\mathbf{a}^T\mathbf{y}_i' > 0 \quad i = 1, \ldots, N$$

* 噪声、计算误差可能使得很接近于0的结果出现问题！
* 怎么办？

感知准则函数

* 引入余量b>0

$$a^T y_i' > 0, i = 1, 2, \ldots, N$$

* 如何求解解向量呢?

To find this solution we must first define an objective function *J(a)*

- A good choice is what is known as the **Perceptron** criterion function

$$J_p(a) = \sum_{y \in Y_M} (-a^T y)$$

  - where Y_M is the set of examples *misclassified* by *a*.
  - Note that $J_P(a)$ is non-negative since $a^T y < 0$ for all misclassified samples

感知准则函数

To find the minimum of this function, we use gradient descent

- The gradient is defined by

$$\nabla_a J_P(a) = \sum_{y \in Y_M} (-y)$$

- And the gradient descent update rule becomes

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

Perceptron rule

- This is known as the ***perceptron batch update rule***.
  - The weight vector may also be updated in an "on-line" fashion, this is, after the presentation of each individual example

$$a(k+1) = a(k) + \eta y^{(i)}$$

  - where $y^{(i)}$ is an example that has been misclassified by $a(k)$

感知准则函数

- **If classes are linearly separable, the perceptron rule is guaranteed to converge to a valid solution**
  - Some version of the perceptron rule use a variable learning rate η(k)
    - In this case, convergence is guaranteed only under certain conditions (for details refer to [Duda, Hart and Stork, 2001], pp. 232-235)
- **However, if the two classes are not linearly separable, the perceptron rule will not converge**
  - Since no weight vector *a* can correctly classify every sample in a non-separable dataset, the corrections in the perceptron rule will never cease
  - One ad-hoc solution to this problem is to enforce convergence by using variable learning rates $\eta(k)$ that approach zero as k approaches infinite

感知准则函数－梯度下降

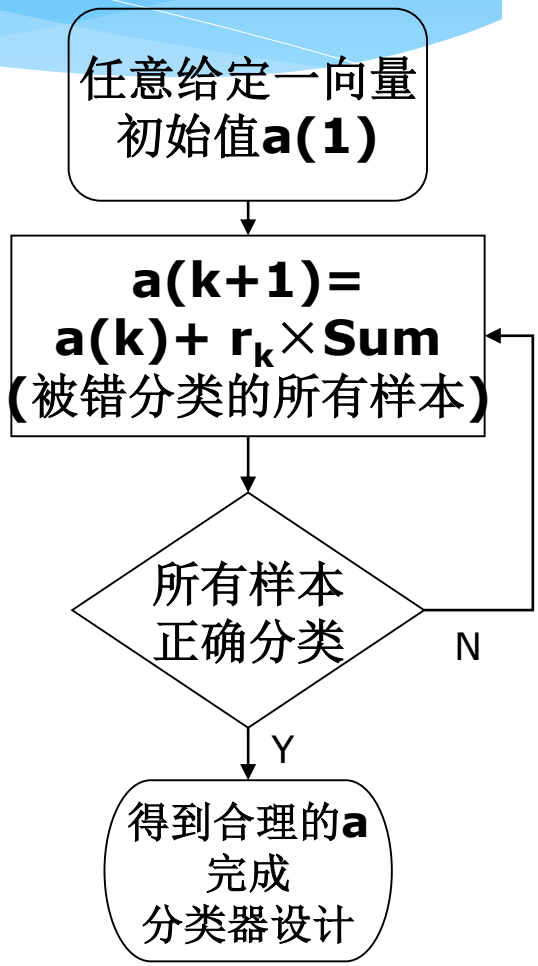
* 梯度下降算法：对(迭代)向量沿某函数的负梯度方向修正，可较快到达该函数极小值。

$$\nabla J_p(\mathbf{a}) = \frac{\delta J_p(\mathbf{a})}{\delta \mathbf{a}} = \sum_{\mathbf{y} \in Y^k} (-\mathbf{y})$$

$$\mathbf{a}(k+1) = \mathbf{a}(k) - r_k \nabla J_p(\mathbf{a}) = \mathbf{a}(k) + r_k \sum_{\mathbf{y} \in Y^k} \mathbf{y}$$

感知准则函数－梯度下降

* 算法具体步骤
* 初值：任意给定一向量初始值a(1)
* 迭代：第k+1次迭代时的权向量a(k+1)等于第k次的权向量a(k)加上被错分类的所有样本之和与r_k的乘积
* 终止：对所有样本正确分类



Fixed-Increment Single-Sample Perceptron Algorithm

- 1 begin initialize $\mathbf{a}, \quad k = 0$
- 2 do $k \leftarrow (k+1)$
- 3 if $\mathbf{y}^{(k)}$ is misclassified by ***a***
  then $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{y}^{(k)}$
- 4 until all patterns properly classified
- 5 return
- 6 end

$$\Delta \mathbf{a} = -\eta \nabla J_p(\mathbf{a}) = -\eta \left(\sum_{\mathbf{y} \in Y} (-\mathbf{y}) \right)$$

a: Modify the weight vector whenever it mis-classifies a single sample

An Example

- Sample set for two-class case
  - Class 1

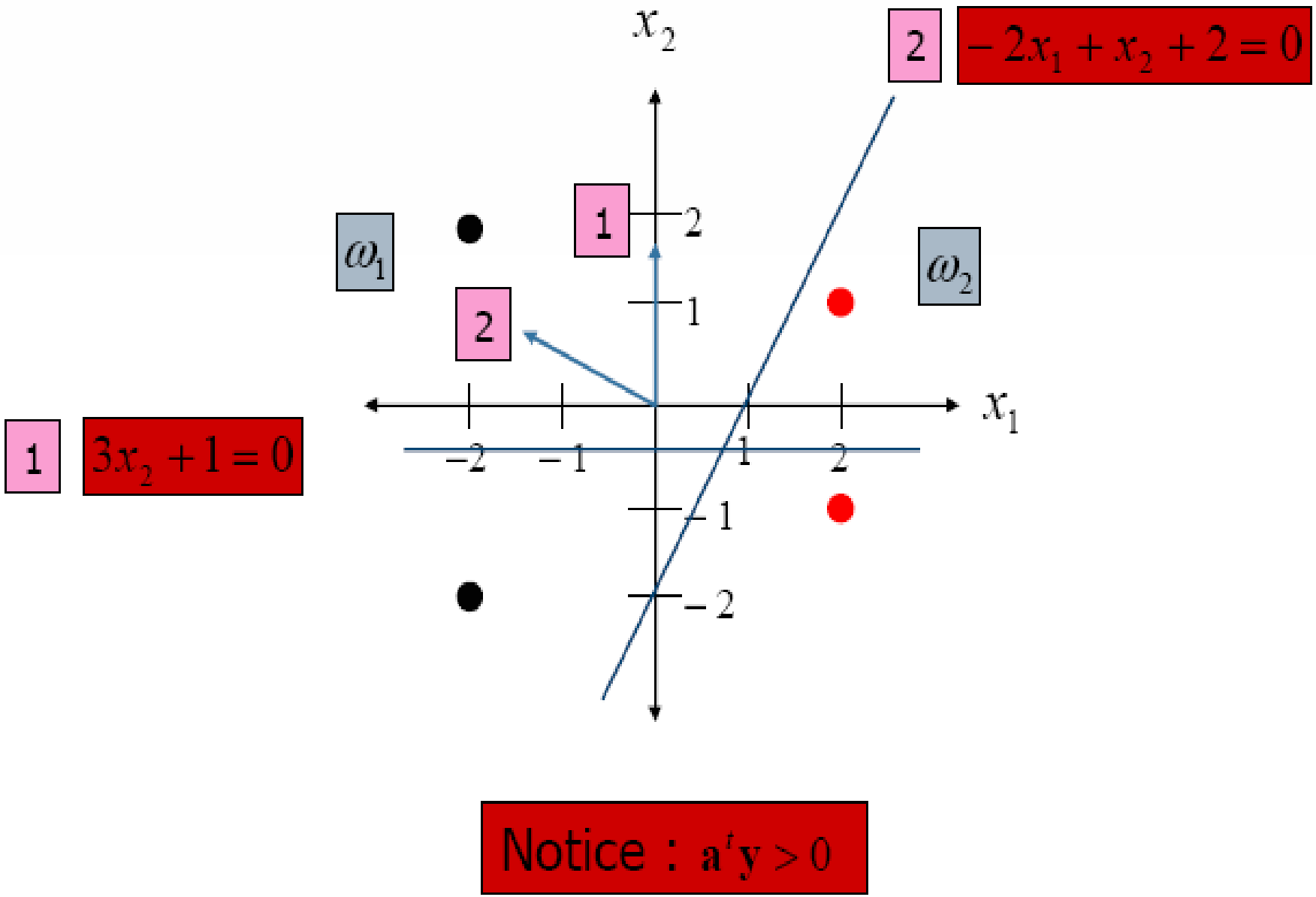
$$\mathbf{x}_1 = \begin{pmatrix} -2 \\ 2 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} -2 \\ -2 \end{pmatrix} \qquad \mathbf{y}_1 = \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix}, \mathbf{y}_2 = \begin{pmatrix} 1 \\ -2 \\ -2 \end{pmatrix}$$

  - Class 2

$$\mathbf{x}_3 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \qquad \mathbf{y}_3 = \begin{pmatrix} -1 \\ -2 \\ -1 \end{pmatrix}, \mathbf{y}_4 = \begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix}$$

  - Initial weight vector

$$\mathbf{a}(1) = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} \qquad \mathbf{a}(1)^t = \begin{pmatrix} 0 & 2 & 1 \end{pmatrix}$$


x_2
2
$-2x_1 + x_2 + 2 = 0$
1
2
ω_1
ω_2
2
1
x_1
1
$3x_2 + 1 = 0$
-2
-1
1
2
-1
-2
Notice : $\mathbf{a}^t\mathbf{y} > 0$

感知准则函数－梯度下降示例

- Example (Cont.)
  - Iterative procedure
    - (1) $\mathbf{y}^{(1)t} = \begin{pmatrix} 1 & -2 & 2 \end{pmatrix}$

$$\mathbf{a}(1)^t \mathbf{y}^{(1)} = \begin{pmatrix} 0 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix} = -2 < 0$$

$$\mathbf{a}(2)^t = \begin{pmatrix} 0 & 2 & 1 \end{pmatrix} + \begin{pmatrix} 1 & -2 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 3 \end{pmatrix}$$

感知准则函数－梯度下降示例

- Example (Cont.)
  - Iterative procedure
    - (2) $\mathbf{y}^{(2)t} = (1 \quad -2 \quad -2)$

$$\mathbf{a}(2)^t \mathbf{y}^{(2)} = (1 \quad 0 \quad 3)\begin{pmatrix} 1 \\ -2 \\ -2 \end{pmatrix} = -5 < 0$$

$$\mathbf{a}(3)^t = (1 \quad 0 \quad 3) + (1 \quad -2 \quad -2) = (2 \quad -2 \quad 1)$$

感知准则函数－梯度下降示例

- Example (Cont.)
  - Iterative procedure
    - (3)

$$\mathbf{y}^{(3)t} = \begin{pmatrix} -1 & -2 & -1 \end{pmatrix}$$

$$\mathbf{a}(3)^t \mathbf{y}^{(3)} = \begin{pmatrix} 2 & -2 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ -2 \\ -1 \end{pmatrix} = 1 > 0$$

$$\mathbf{a}(4)^t = \begin{pmatrix} 2 & -2 & 1 \end{pmatrix} \text{ (no change)}$$

感知准则函数－梯度下降示例

- Example (Cont.)
  - Iterative procedure
    - (4) $\mathbf{y}^{(4)t} = (-1 \quad -2 \quad 1)$

$$\mathbf{a}(4)^t \mathbf{y}^{(3)} = (2 \quad -2 \quad 1)\begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix} = 3 > 0$$

$$\mathbf{a}(5)^t = (2 \quad -2 \quad 1) \quad \text{(no change)}$$

感知准则函数－梯度下降示例

- Example (Cont.)
  - Iterative procedure
    - (5)

$$\mathbf{y}^{(5)t} = \begin{pmatrix} 1 & -2 & 2 \end{pmatrix}$$

$$\mathbf{a}(5)^t \mathbf{y}^{(1)} = \begin{pmatrix} 2 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix} = 8 > 0$$

$$\mathbf{a}(6)^t = \begin{pmatrix} 2 & -2 & 1 \end{pmatrix} \qquad \text{(no change)}$$

感知准则函数－梯度下降示例

- Example (Cont.)
  - Iterative procedure
    - (6)

$$\mathbf{y}^{(6)t} = \begin{pmatrix} 1 & -2 & -2 \end{pmatrix}$$

$$\mathbf{a}(6)^t \mathbf{y}^{(2)} = \begin{pmatrix} 2 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ -2 \end{pmatrix} = 4 > 0$$

$$\mathbf{a}(7)^t = \begin{pmatrix} 2 & -2 & 1 \end{pmatrix} \text{ (no change)}$$

感知准则函数－梯度下降示例

- Example (Cont.)
  - Iterative procedure
    - (7)

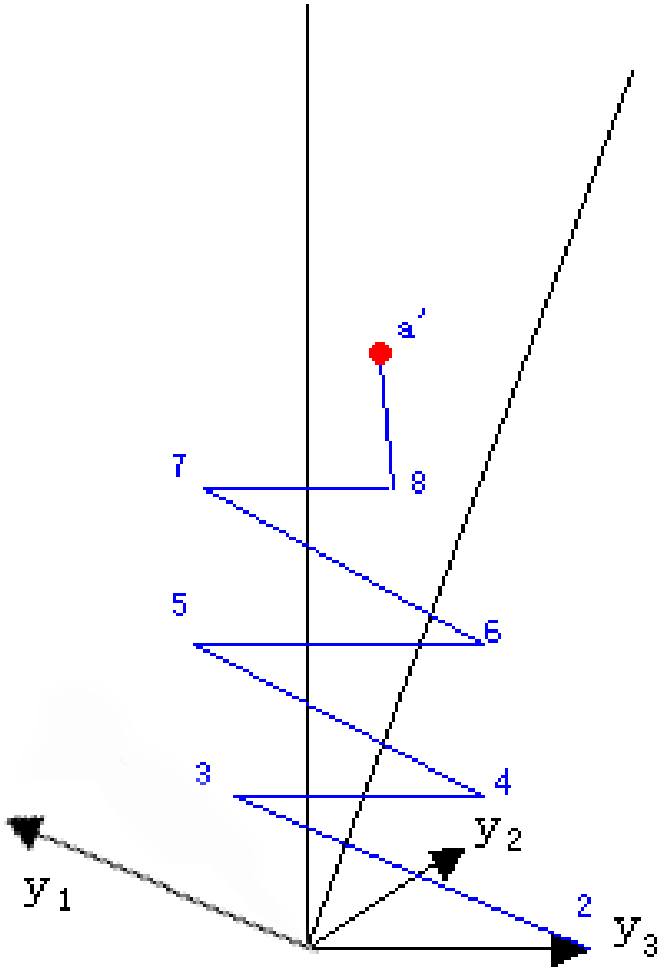
$$\mathbf{y}^{(7)t} = \begin{pmatrix} -1 & -2 & -1 \end{pmatrix}$$

$$\mathbf{a}(7)^t \mathbf{y}^{(7)} = \begin{pmatrix} 2 & -2 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ -2 \\ -1 \end{pmatrix} = 1 > 0$$

$$\mathbf{a}(8)^t = \begin{pmatrix} 2 & -2 & 1 \end{pmatrix} \text{ (no change)}$$

感知准则函数

* 固定增量法与可变增量法
* 批量样本修正法与单样本修正法
  * 单样本修正法：样本集视为不断重复出现的序列，逐个样本检查，修正权向量
  * 批量样本修正法：样本成批或全部检查后，修正权向量



感知准则函数－小结

* 感知准则函数方法的思路：
* 先随意找一个初始向量a(1)，然后用训练样本集中的每个样本来计算。
* 若发现一个y出现$a^Ty<0$，则只要$a(k+1) = a(k) + r_ky$，r_k为正(步长系数)，则必有$a(k+1)^Ty = a(k)^Ty + r_ky^Ty$，就有趋势做到使$a(k+1)^Ty >0$。
* 当然，修改后的a(k+1)还可以使某些y出现$a(k+1)^Ty <0$的情况，理论证明，只要训练样本集线性可分，无论a(1)的初值是什么，经过有限次叠代，都可收敛。

最小平方误差准则函数

* 规范化增广样本向量y_i，增广权向量a，正确分类要求： $a^T y_i>0, i=1,\ldots,N$
* 线性分类器设计↔求一组N个线性不等式的解

最小平方误差准则函数

* 样本集增广矩阵Y及一组N个线性不等式的矩阵表示：

$$Y=\begin{bmatrix}\mathbf{y}_1^T\\ \mathbf{y}_1^T\\ \cdots\\ \mathbf{y}_N^T\end{bmatrix}=\begin{bmatrix}y_{11} & y_{12} & \cdots & y_{1\hat{d}}\\ y_{21} & y_{22} & \cdots & y_{2\hat{d}}\\ \cdots & \cdots & \cdots & \cdots\\ y_{N1} & y_{N2} & \cdots & y_{N\hat{d}}\end{bmatrix}$$

最小平方误差准则函数

* 引入余量(目标向量) $b=[b_1, b_2, \ldots, b_N]^T$， b_i任意给定正常数， $a^T y_i = b_i > 0$
* N个线性方程的的矩阵表示：

$$Ya > 0$$

$$Ya = b$$

最小平方误差准则函数

* 定义误差向量 e=Ya-b
* 定义平方误差准则函数$J_s(a)$:

$$J_s(\mathbf{a}) = \|\mathbf{e}\|^2 = \|Y\mathbf{a} - \mathbf{b}\|^2 = \sum_{i=1}^{N} (\mathbf{a}^T \mathbf{y}_i - b_i)^2$$

* SSE & MSE

最小平方误差准则函数
函数解

* 最小平方误差准则函数解：

$$\mathbf{a}^* = \underset{\mathbf{a}}{\operatorname{argmin}} J_s(\mathbf{a})$$

MSE方法的思想：对每个样本，设定一个"理想"的判别函数输出值，以最小平方误差为准则求最优权向量

最小平方误差准则函数
MSE准则函数的伪逆解

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

$$\nabla J_s(\mathbf{a}^*) = 0 \Leftrightarrow Y^T Y \mathbf{a}^* = Y^T \mathbf{b}$$

$$\mathbf{a}^* = (Y^T Y)^{-1} Y^T \mathbf{b} = Y^+ \mathbf{b}$$

Y的
伪逆矩阵

最小平方误差准则函数
MSE方法与其他方法的关系

* 与Fisher方法的关系： 当

$$\mathbf{b}=\begin{bmatrix} N/N_1 \\ \cdots \\ N/N_1 \\ N/N_2 \\ \cdots \\ N/N_2 \end{bmatrix}$$

N_1个

N_2个

* MSE解等价于Fisher解

最小平方误差准则函数
MSE方法与其他方法的关系

* 与Bayes方法的关系： 当N→∞，$b=u_N=[1,1,\ldots,1]^T$ 时，则它以最小均方误差逼近Bayes判别函数：

$$g(\mathbf{x}) = P(\omega_1 \mid \mathbf{x}) - P(\omega_2 \mid \mathbf{x})$$

最小平方误差准则函数
MSE方法的迭代解

* 伪逆解$\mathbf{a}^*=Y^+\mathbf{b}$, $Y^+=(Y^TY)^{-1}Y^T$，计算量大
* 实际中常用梯度下降法：

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

$$\begin{cases}\mathbf{a}(1),\text{任意初始化}\\ \mathbf{a}(k+1)=\mathbf{a}(k)-r_kY^T(Y\mathbf{a}-\mathbf{b})\end{cases}$$

* 满足一定条件时梯度下降结束

最小平方误差准则函数
MSE方法的迭代解

* 单样本修正调整权向量
* Widrow-Hoff算法/最小均方根算法/LMS算法

$$a(k+1) = a(k) + r_k(b_i - a(k)^T y_i)y_i$$

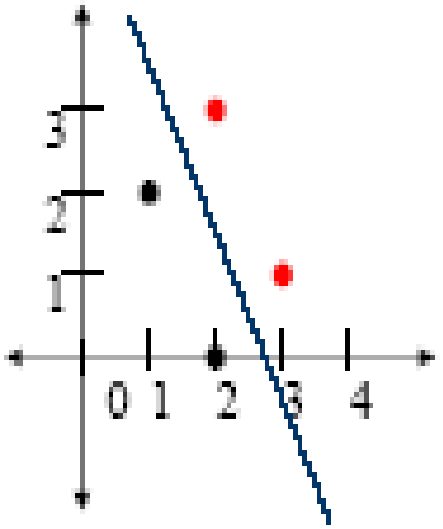
其中y_i是使得$a(k)^T y_i \neq b_i$的样本

Example 1

- Sample set for two-class case
  - Class 1
  - Class 2

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \qquad \boxed{\mathbf{x}_1 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}}$$

$$\mathbf{Y} = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ -1 & -3 & -1 \\ -1 & -2 & -3 \end{pmatrix} \qquad \mathbf{Y}^+ = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t = \begin{pmatrix} \frac{5}{4} & \frac{13}{12} & \frac{3}{4} & \frac{7}{12} \\ -\frac{1}{2} & -\frac{1}{6} & -\frac{1}{2} & -\frac{1}{6} \\ 0 & -\frac{1}{3} & 0 & -\frac{1}{3} \end{pmatrix}$$



$$\mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \Rightarrow \mathbf{a} = \mathbf{Y}^+\mathbf{b} = \begin{pmatrix} \frac{11}{3} \\ -\frac{4}{3} \\ -\frac{2}{3} \end{pmatrix} \Rightarrow \mathbf{a}^t \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} = 0 \Rightarrow \frac{11}{3} - \frac{4}{3}x_1 - \frac{2}{3}x_2 = 0 \Rightarrow \frac{4}{11}x_1 + \frac{2}{11}x_2 = 1$$

最小平方误差准则函数
MSE方法的迭代解示例

☐ Verification

$$\mathbf{Ya} = \mathbf{b} \Rightarrow$$

$$\begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ -1 & -3 & -1 \\ -1 & -2 & -3 \end{pmatrix} \begin{pmatrix} 11 \\ -4 \\ -2 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \end{pmatrix}$$

$$\mathbf{Y} = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ -1 & -3 & -1 \\ -1 & -2 & -3 \end{pmatrix} \quad \mathbf{a} = \begin{pmatrix} \frac{11}{3} \\ -\frac{4}{3} \\ -\frac{2}{3} \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

最小平方误差准则函数 MSE方法的迭代解示例

- Sample set for two-class case
  - Class 1

$$\mathbf{x}_1=\begin{pmatrix}0\\0\end{pmatrix},\mathbf{x}_2=\begin{pmatrix}1\\1\end{pmatrix},\mathbf{x}_3=\begin{pmatrix}2\\2\end{pmatrix} \Rightarrow \mathbf{y}_1=\begin{pmatrix}1\\0\\0\end{pmatrix},\mathbf{y}_2=\begin{pmatrix}1\\1\\1\end{pmatrix},\mathbf{y}_3=\begin{pmatrix}1\\2\\2\end{pmatrix}$$

  - Class 2

$$\mathbf{x}_1=\begin{pmatrix}-1\\0\end{pmatrix},\mathbf{x}_2=\begin{pmatrix}-1\\1\end{pmatrix},\mathbf{x}_3=\begin{pmatrix}-1\\2\end{pmatrix} \Rightarrow \mathbf{y}_1=\begin{pmatrix}-1\\1\\0\end{pmatrix},\mathbf{y}_2=\begin{pmatrix}-1\\1\\-1\end{pmatrix},\mathbf{y}_3=\begin{pmatrix}-1\\1\\-2\end{pmatrix}$$

Example 2

$$y_1 = \begin{pmatrix}1\\0\\0\end{pmatrix}, y_2 = \begin{pmatrix}1\\1\\1\end{pmatrix}, y_3 = \begin{pmatrix}1\\2\\2\end{pmatrix}$$

- $(\mathbf{Y}'\mathbf{Y})^{-1} = \left(\sum_{i=1}^{n} y_i y_i'\right)^{-1}$

$$y_1 = \begin{pmatrix}-1\\1\\0\end{pmatrix}, y_2 = \begin{pmatrix}-1\\1\\-1\end{pmatrix}, y_3 = \begin{pmatrix}-1\\1\\-2\end{pmatrix}$$

$$\sum_{i=1}^{n} y_i y_i' = \begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0 & 0\end{pmatrix} + \begin{pmatrix}1\\1\\1\end{pmatrix}\begin{pmatrix}1 & 1 & 1\end{pmatrix} + \begin{pmatrix}1\\2\\2\end{pmatrix}\begin{pmatrix}1 & 2 & 2\end{pmatrix}$$

$$+ \begin{pmatrix}-1\\1\\0\end{pmatrix}\begin{pmatrix}-1 & 1 & 0\end{pmatrix} + \begin{pmatrix}-1\\1\\-1\end{pmatrix}\begin{pmatrix}-1 & 1 & -1\end{pmatrix} + \begin{pmatrix}-1\\1\\-2\end{pmatrix}\begin{pmatrix}-1 & 1 & -2\end{pmatrix} = \begin{pmatrix}6 & 0 & 6\\0 & 8 & 2\\6 & 2 & 10\end{pmatrix}$$

$$\left(\sum_{i=1}^{n} y_i y_i'\right)^{-1} = \begin{pmatrix}6 & 0 & 6\\0 & 8 & 2\\6 & 2 & 10\end{pmatrix}^{-1} \approx \begin{pmatrix}3 & 0 & 3\\0 & 4 & 1\\3 & 1 & 5\end{pmatrix}^{-1} \approx \begin{pmatrix}19 & 3 & -12\\3 & 6 & -3\\-12 & -3 & 12\end{pmatrix}$$

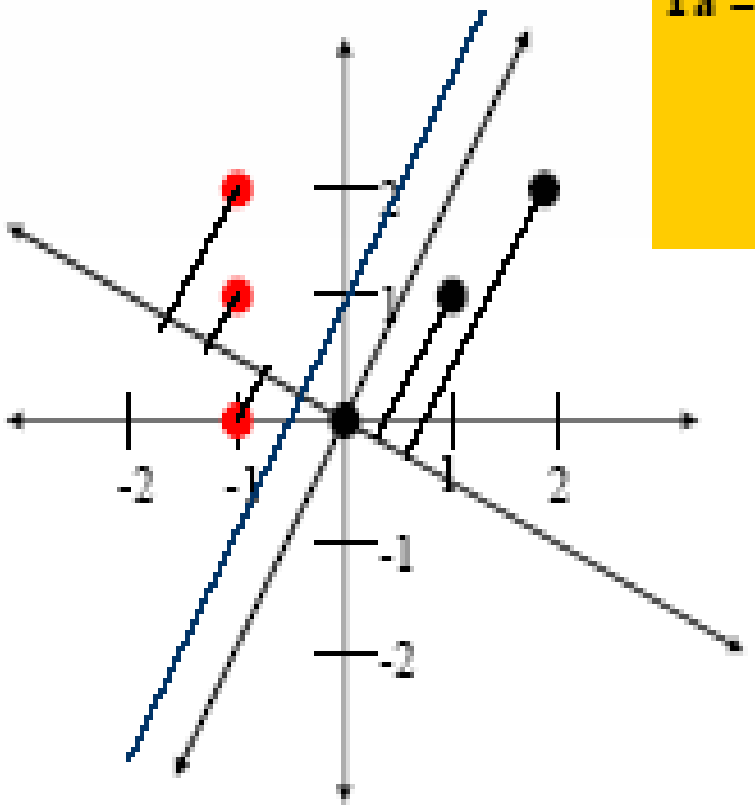
□ $\mathbf{a} = (\mathbf{Y}'\mathbf{Y})^{-1}\mathbf{Y}'\mathbf{b} = \mathbf{Y}^{+}\mathbf{b}$

$$\mathbf{Y}'\mathbf{b} = \begin{pmatrix} 1 & 1 & 1 & -1 & -1 & -1 \\ 0 & 1 & 2 & 1 & 1 & 1 \\ 0 & 1 & 2 & 0 & -1 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 6 \\ 0 \end{pmatrix} \qquad (\mathbf{Y}'\mathbf{Y})^{-1} = \begin{pmatrix} 19 & 3 & -12 \\ 3 & 6 & -3 \\ -12 & -3 & 12 \end{pmatrix}$$

$$\mathbf{a} = (\mathbf{Y}'\mathbf{Y})^{-1}\mathbf{Y}'\mathbf{b} = \begin{pmatrix} 19 & 3 & -12 \\ 3 & 6 & -3 \\ -12 & -3 & 12 \end{pmatrix} \begin{pmatrix} 0 \\ 6 \\ 0 \end{pmatrix} = \begin{pmatrix} 18 \\ 36 \\ -18 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$$

$$\mathbf{a}'\mathbf{y} = 0 \qquad (1 \quad 2 \quad -1) \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} = 0 \qquad 2x_1 - x_2 + 1 = 0$$

$$Ya = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 2 \\ -1 & 1 & 0 \\ -1 & 1 & -1 \\ -1 & 1 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 3 \end{pmatrix}$$



$$2x_1 - x_2 + 1 = 0$$

☐ Verification

$$\mathbf{Ya} = \mathbf{b}$$

$$\mathbf{Ya} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 2 \\ -1 & 1 & 0 \\ -1 & 1 & -1 \\ -1 & 1 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\mathbf{Y} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 2 \\ -1 & 1 & 0 \\ -1 & 1 & -1 \\ -1 & 1 & -2 \end{pmatrix}$$

$$\mathbf{a} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

最小平方误差准则函数
Perceptron Vs. MSE

- Sample set for two-class case
  - Class 1
  - Class 2

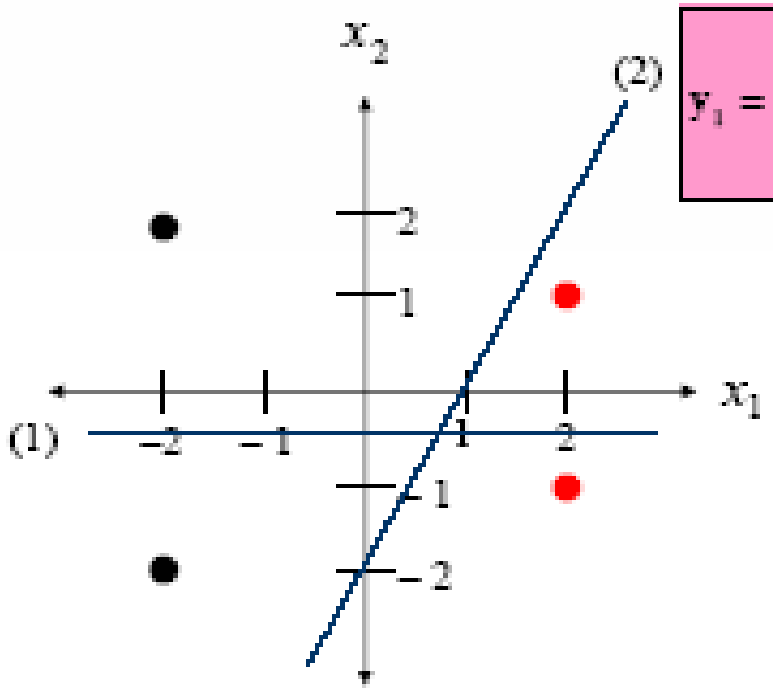
$$\mathbf{x}_1 = \begin{pmatrix} -2 \\ 2 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} -2 \\ -2 \end{pmatrix} \qquad \mathbf{y}_1 = \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix}, \mathbf{y}_2 = \begin{pmatrix} 1 \\ -2 \\ -2 \end{pmatrix}$$

$$\mathbf{x}_3 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \qquad \mathbf{y}_3 = \begin{pmatrix} -1 \\ -2 \\ -1 \end{pmatrix}, \mathbf{y}_4 = \begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix}$$

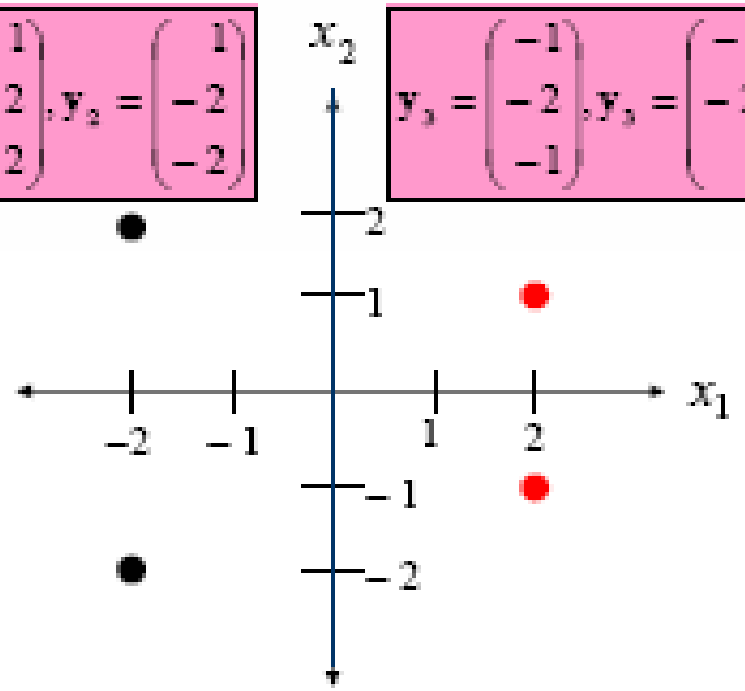
最小平方误差准则函数

Perceptron

MSE



$$y_1 = \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix}, y_2 = \begin{pmatrix} 1 \\ -2 \\ -2 \end{pmatrix}$$



$$y_3 = \begin{pmatrix} -1 \\ -2 \\ -1 \end{pmatrix}, y_4 = \begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix}$$

$Ya = b$

$$\begin{pmatrix} 1 & -2 & 2 \\ 1 & -2 & -2 \\ -1 & -2 & -1 \\ -1 & -2 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 1 \\ 3 \end{pmatrix}$$

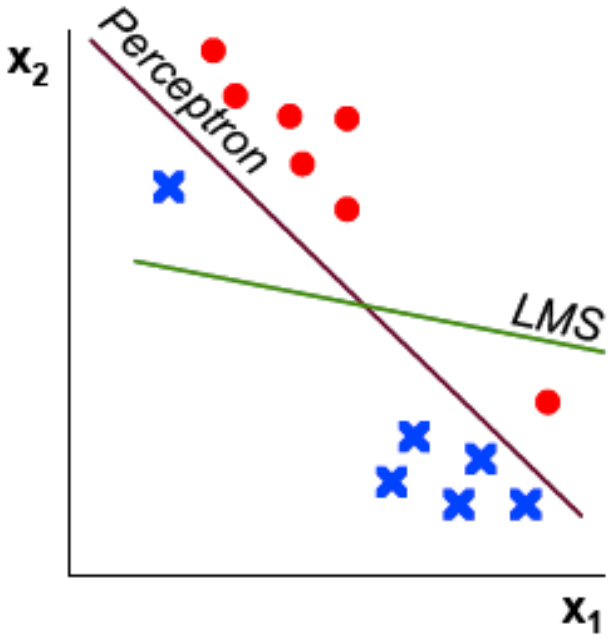
$$a = \begin{pmatrix} 2 \\ -2 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & -2 & 2 \\ 1 & -2 & -2 \\ -1 & -2 & -1 \\ -1 & -2 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -20 \\ 0 \end{pmatrix} = \begin{pmatrix} 40 \\ 40 \\ 40 \\ 40 \end{pmatrix}$$

$$a = \begin{pmatrix} 0 \\ -20 \\ 0 \end{pmatrix}$$

最小平方误差准则函数
Perceptron Vs. MSE

- **Perceptron rule**
  - The perceptron rule always finds a solution if the classes are linearly separable, but does not converge if the classes are non-separable
- **MSE criterion**
  - The MSE solution has guaranteed convergence, but it may not find a separating hyperplane if classes are linearly separable
    - Notice that MSE tries to minimize the sum of the squares of the distances of the training data to the separating hyperplane, as opposed to finding this hyperplane



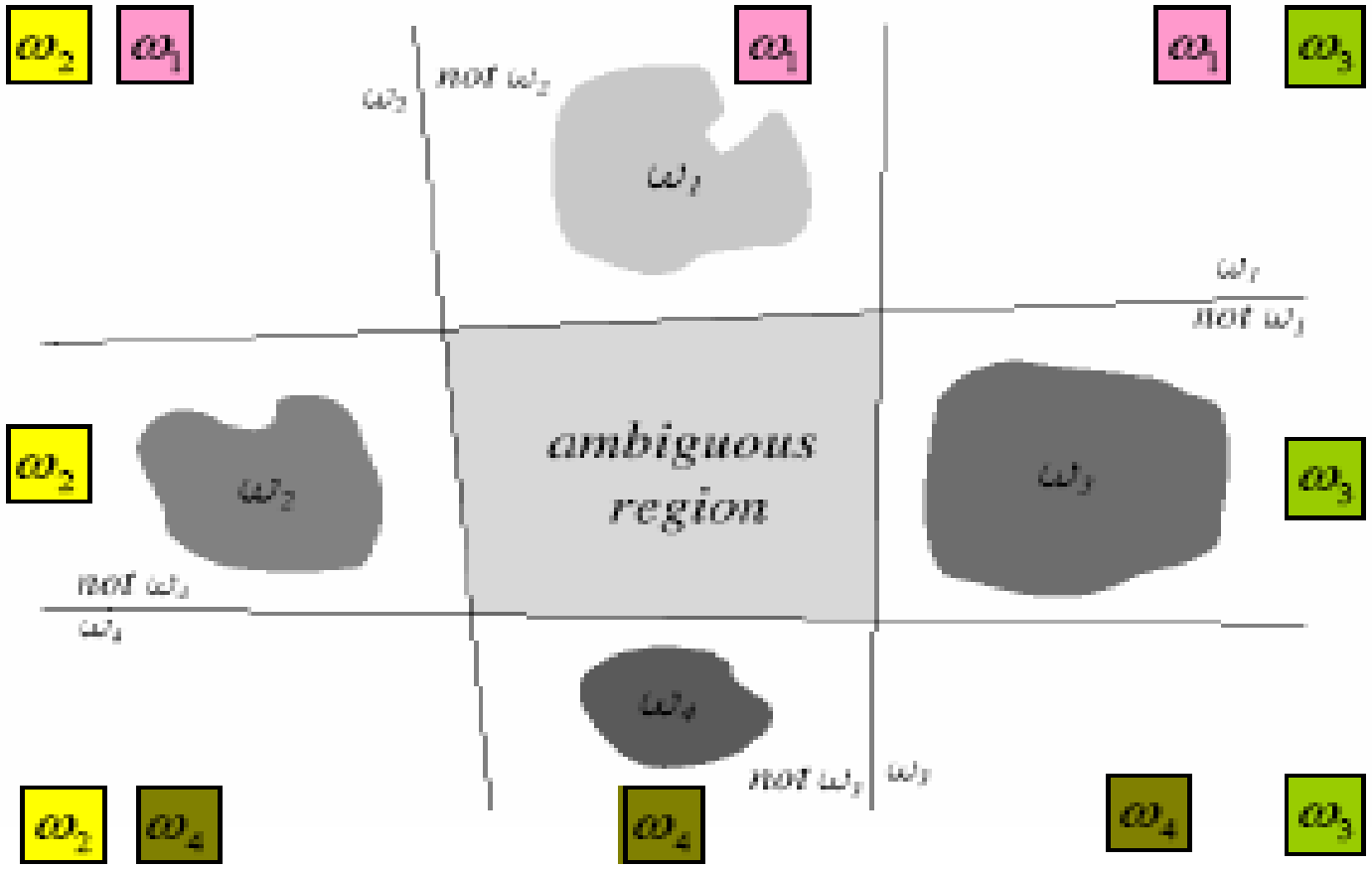
多类问题

- Three approaches
  - c-1 two-class problems
    - Separate points assigned to ω_i from those not assigned to ω_i .
  - $c(c\text{-}1)/2$ two-class problems
    - Separate every pair of classes.
  - Define c linear discriminant functions
    - Assign x to ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0} \qquad i = 1, \cdots, c$$

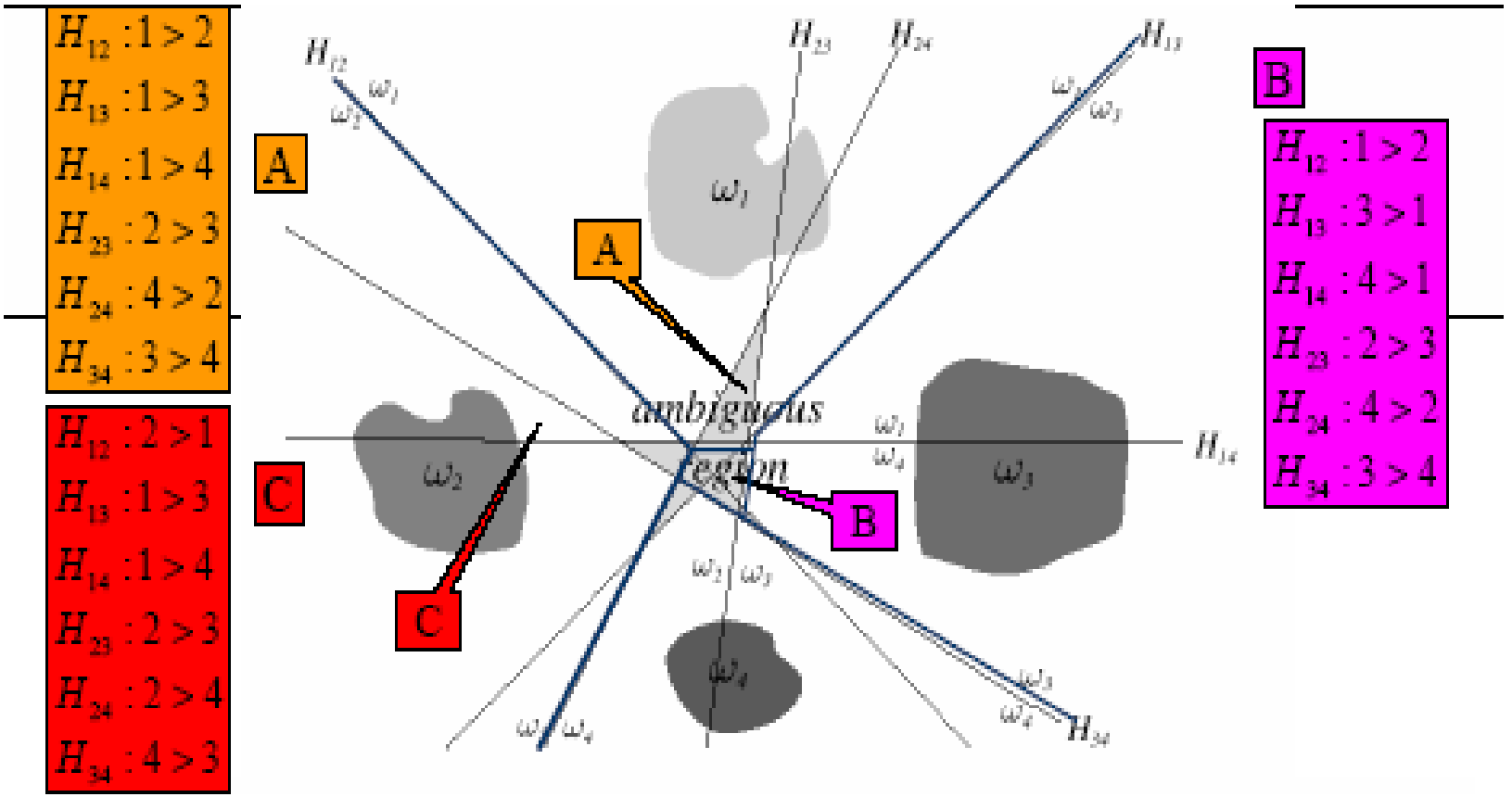
多类问题

Ambiguous region in c-1 two-class problems



多类问题

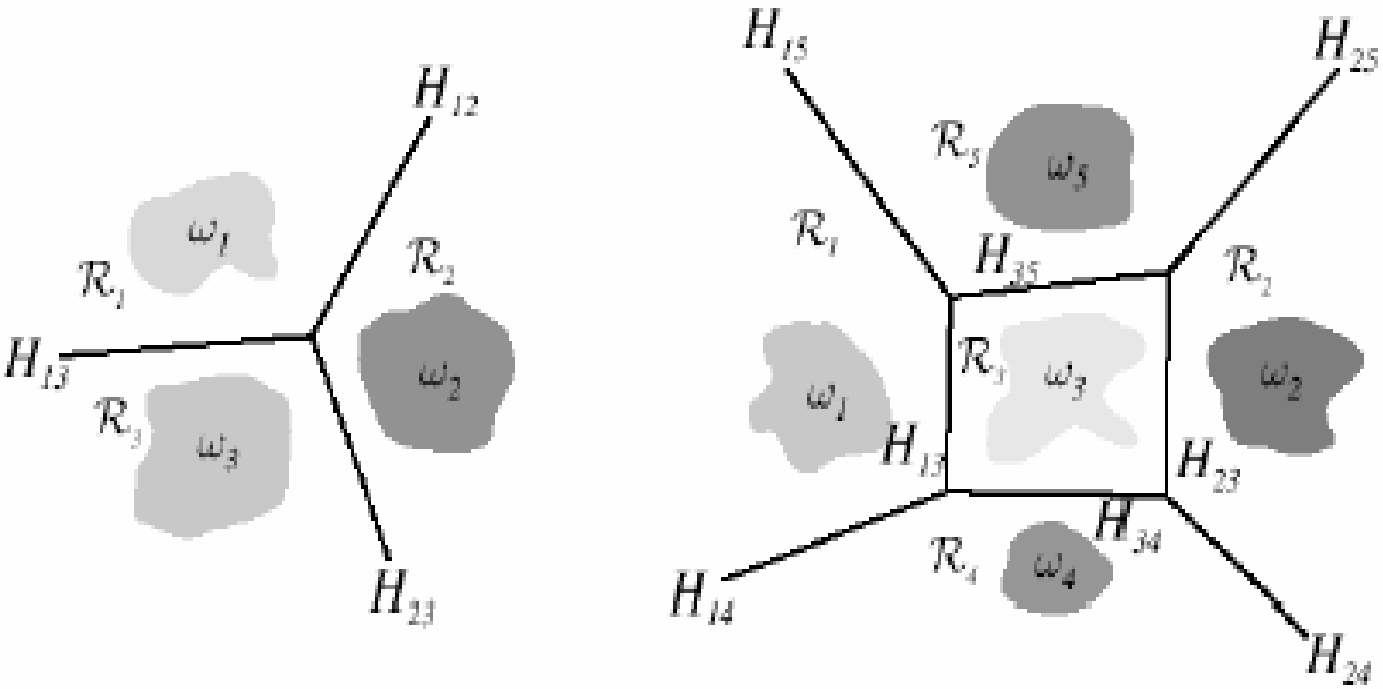
Ambiguous region in $c(c-1)/2$ two-class problems



多类问题

Linear machine

Use c linear discriminant functions



决策树

* 模式识别 第三版 P132-P139

谢谢