

# 第四章 分类线性模型

主讲人 王星

中国人民大学统计学院

April 6, 2022

# 大纲

- ▶ 分类器，判别函数和决策面
- ▶ 基本形式
- ▶ 线性回归(review)\*
- ▶ 二次判别
- ▶ Fisher 判别
- ▶ LDA线性判别分析
- ▶ 多分类学习
- ▶ 类别不平衡问题

# 分类的基本问题

1. 回归问题有很好的数学性质和计算性质，包括3个基本要素预测分布、估计过程和证据近似；
2. 分类的基本问题是将输入变量分到 $C_k$ 个类别中。最常见的情况是不同的类别之间互不相交，当然也有不同类别之间存在交叠的问题。在第一种情况中(本章重点)，每个数据点分到唯一的类别中，输入空间为决策区域，边界称为决策边界或决策面。
3. 本章考虑的线性分类模型,指的是决策面是线性模型。如果数据集可以被精准地分类，那么这个数据集称为线性可分的。

$$y(x, \omega) = f(x^T \omega + \omega_0).$$

4. 在分类中，二分类就是一个目标变量 $t \in \{0, 1\}$ . 其中 $t = 1$ 表示类别 $C_1$  而 $t = 0$ 表示类别 $C_2$ , 这样除了关心每个数据的类别 $C_1, C_2$ , 我们还关心属于某个类别的概率。
5. 对于 $K > 2$ 的情形，比较方便的表示方法是”1-of-K”的编码方式。 $t = c(0/1, 0/1, \dots, 0/1, 0/1)^T$ , 我们也关心每个类别的概率。

# Least Squares for Classification

- Consider a general classification problem with  $K$  classes using 1-of- $K$  encoding scheme for the target vector  $\mathbf{t}$ .
- Remember: **Least Squares approximates the conditional expectation**  $\mathbb{E}[\mathbf{t}|\mathbf{x}]$ .

- Each class is described by its own linear model:

$$y_k(\mathbf{x}) = \mathbf{x}^T \mathbf{w}_k + w_{k0}, \text{ where } k = 1, \dots, K.$$

- Using vector notation, we can write:

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}$$

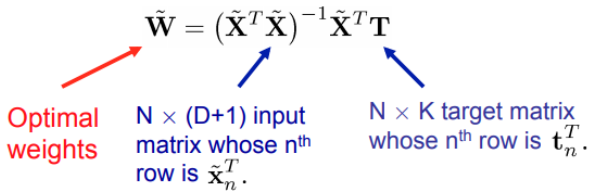
$(D+1) \times K$  matrix whose  $k^{\text{th}}$  column comprises of  $D+1$  dimensional vector:

$$\tilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^T)^T.$$

corresponding augmented input vector:

$$\tilde{\mathbf{x}} = (1, \mathbf{x}^T)^T.$$

- Consider observing a dataset  $\{\mathbf{x}_n, \mathbf{t}_n\}$ , where  $n=1, \dots, N$ .
- We have already seen how to do least squares. Using some matrix algebra, we obtain the **optimal weights**:

$$\tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{T}$$


Optimal weights

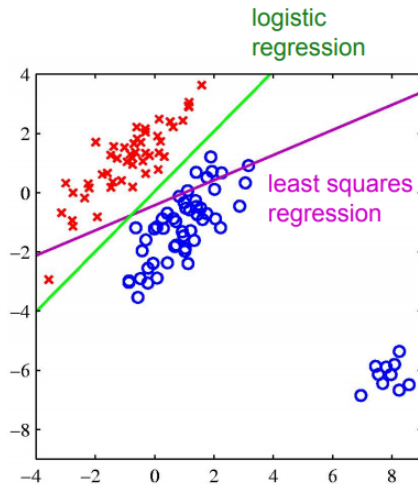
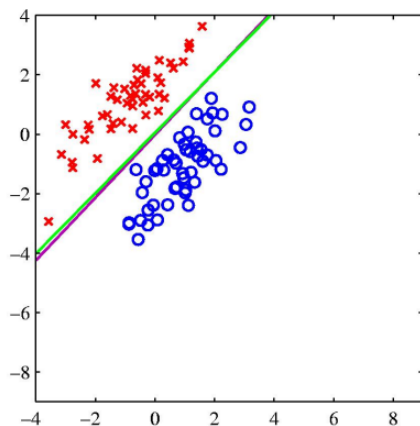
$N \times (D+1)$  input matrix whose  $n^{\text{th}}$  row is  $\tilde{\mathbf{x}}_n^T$ .

$N \times K$  target matrix whose  $n^{\text{th}}$  row is  $\mathbf{t}_n^T$ .

- A new input  $\mathbf{x}$  is assigned to a class for which  $y_k = \tilde{\mathbf{x}}^T \tilde{\mathbf{w}}_k$  is largest.
- There are however several problems when using least squares for classification.

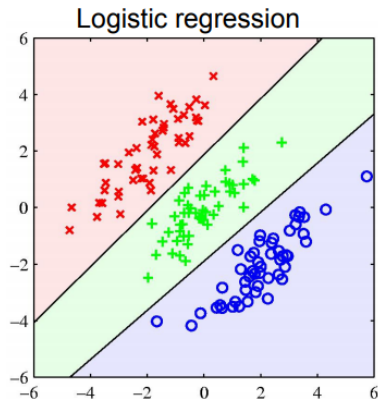
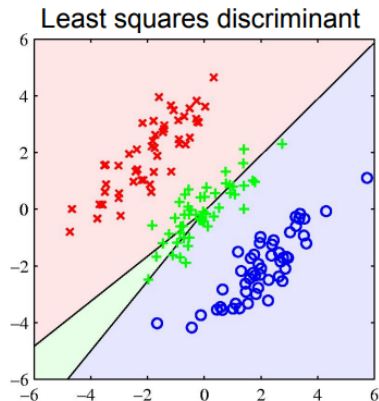
# 用LS直接分类的问题1，在二分类中抗干扰能力差

Least squares is highly sensitive to outliers,  
unlike logistic regression



## 用LS直接分类的问题2，在多分类

Example of synthetic dataset containing 3 classes, where lines denote decision boundaries.



Many green points are misclassified.

# 分类问题-推断和决策

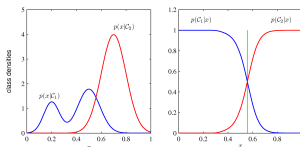
有3种构造判别函数的方法

- ▶ 分布依赖型（得分依赖型）：分类问题可以分为两个阶段：推断和决策，在推断的阶段训练分布，在决策的阶段结合损失函数给出最优的分类。通常又可以分为两种方法：
  - ▶ 概率生成法（**generative**）
    - ▶ 用一个生成模型去推断 $p(x|C_k)$ ;
    - ▶ 将先验分布 $p(C_k)$  和 $p(x|C_k)$  联合得到 $p(C_k|x)$ ;
  - ▶ 判别模型法(**discriminant**): 直接推断 $p(C_k|x)$ ;
- ▶ 非概率依赖型（非得分依赖型）:直接学习一个判别函数 $g(x)$ .
  - ▶ 直接解一个有分类变量的映射函数;
  - ▶ 而分类问题而言，相当于一个 $\{+1, -1\}$  的函数，比如直接的最小二乘、决策树、SVM,NN



# Decision theory—inference and decision

- ▶ 分布依赖型生成式模型:
  - ▶ **pros**: access to  $p(x) \rightarrow$  easy detection of outliers | i.e., low-confidence predictions |;
  - ▶ **cons**: estimating the joint probability  $p(x, C_k)$  can be computational and data demanding.
- ▶ 分布依赖型判别模型:
  - ▶ **pros**: less demanding than the generative approach;



- ▶ 判别函数:
  - ▶ **pros**: a single learning problem (vs inference + decision);
  - ▶ **cons**: no access to  $p(C_k|x)$  which can have many advantages in practice for (e.g.) rejection and model combination – see page 45.

# 分类问题中的三个空间: Three different spaces that are easy to confuse

- ▶ 权值空间 **M Weight-space**: 参数空间
  - ▶ Each axis corresponds to a weight;
  - ▶ A point is a weight vector;
  - ▶ Dimensionality:  $= \# \text{weights} + 1$  extra dimension for the loss.
- ▶ 数据空间 **D Data-space**: 有效视角
  - ▶ Each axis corresponds to an input value;
  - ▶ A point is a data vector;
  - ▶ A decision surface is a plane;
  - ▶ Dimensionality = dimensionality of a data vector;
- ▶ 样例空间 **N Case-space**: 证据近似
  - ▶ Each axis corresponds to a training case;
  - ▶ A point assigns a scalar value to every training case.
  - ▶ So it can represent the 1-D targets or it can represent the value of one input component over all the training data.
  - ▶ Dimensionality  $= \# \text{training cases}$ .

# 决策论-推断与决策

- ▶ The decision problem:
  - ▶ given  $x$ , predict  $t$  according to a probabilistic model  $p(x, t)$
- ▶ binary classification:  $t \in \{0, 1\} \leftrightarrow \{C_1, C_2\}$ ;
- ▶ Important quantity:  $p(C_k|x)$ .

$$p(C_k|x) = \frac{p(x, C_k)}{p(x)} = \frac{p(x, C_k)}{\sum_{k=1}^2 p(x, C_k)}$$

→ getting  $p(x, C_i)$  is the (central!) inference problem

$$= \frac{p(x|C_k)p(C_k)}{p(x)} \propto \text{likelihood} \times \text{prior}$$

- ▶ Intuition: choose  $k$  that maximizes  $p(C_k|x)$ .

# 决策论-二分类

- ▶ 决策域:  $\mathcal{R}_i = \{x : \text{pred}(x) = C_i\}$ .
- ▶ 判错率:

$$\begin{aligned} p(\text{misclassification}) &= p(x \in \mathcal{R}_1, C_2) + p(x \in \mathcal{R}_2, C_1) \\ &= \int_{\mathcal{R}_1} p(x, C_2) dx + \int_{\mathcal{R}_2} p(x, C_1) dx \end{aligned}$$

- ▶  $\Leftrightarrow p(C_1|x) > p(C_2|x)$ .
- ▶ 对于 $k$ 类而言: 极小化  $\sum_j = \int_{\mathcal{R}_j} (\sum_{k \neq j} p(x, C_k)) dx$   
 $\Leftrightarrow \text{pred}(x) = \operatorname{argmax}_k p(C_k|x)$ .

# 分类问题—推断和决策

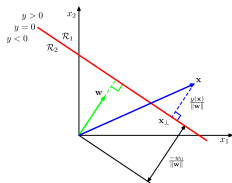
- ▶ 假设 $\{C_i\}, i = 1, \dots, K$  个不同的类, 如果 $g_i(x) > g_j(x), i \neq j, g_i(x), i = 1, \dots, K$  称为判别函数;
- ▶ 例如:  $g_i(x) = -R(\delta_i|x)$ (这对应于后验风险最小准则), 那么根据错误最小化原理,  $g_i(x) = p(C_i|x)$ , 于是

$$g_i(x) = p(x|C_i)p(C_i).$$

- ▶ 判别函数可以替换为任意一个单调增函数 $f(g(.))$ ,

$$g_i(x) = \ln p(x|C_i) + \ln p(C_i).$$

# 判别函数



- ▶ The planar decision surface in data-space for the simple linear discriminant function:  $w^T x + w_0$
- ▶ 线性判别函数  $y(x) = w^T x + w_0$ ; 考虑两个决策面上的点  $x_A, x_B$ , 应该有  $w^T (x_A - x_B) = 0$ . 权值向量与决策面上的任意向量都正交。
- ▶ 考虑  $y(x) = 0, x \in$  决策面上, 于是从原点出发到决策面的垂直距离是

$$\frac{w^T x}{\|w\|} = -\frac{w_0}{\|w\|}.$$

- ▶ 任意一点和它在决策面的投影  $x_{\perp}$   
有  $x = x_{\perp} + r \frac{w}{\|w\|}$ ,  $r = \frac{y(x)}{\|w\|}$ .

### 3.4.1 LDA学习的Bayes解决方案

对于多分类问题，假设有 $C$ 个判别函数 $g_c(x) : c = 1, \dots, C$ , 将 $x$ 判为 $c$ 类，如果满足

$$g_c(x) > g_i(x), \forall i \neq c.$$

- ▶ 令 $g_c(x) = P(c|x)$ ,  $g_c(x) = P(x|c)P(c)$
- ▶ 将这个判别函数替换成任意一个单调函数:

$$g_c(x) = \ln P(x|c) + \ln P(c)$$

- ▶ 对于二分类问题 $\{0, 1\}$ , 可以定义一个决策函数

$$g(x) = g_1(x) - g_0(x)$$

- ▶ 定义: 判别函数

$$\delta(x) = \begin{cases} i, & g(x) > 0 \\ j, & \text{otherwise} \end{cases}$$

- ▶ 二分类问题中通常会有两类决策函

数 $g(x) : g_1(x) = P(1|x) - P(0|x)$ ;

$$g_2(x) = \ln P(1|x) - \ln P(0|x) = \ln \frac{P(x|1)}{P(x|0)} + \ln \frac{P(1)}{P(0)};$$

### 3.4.1 多元正态分布

- ▶ 如果 $P(x|c)$  服从多元正态分布

$$P(x|c) = \frac{1}{(2\pi)^{d/2} |\Sigma_c|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu_c)^T \Sigma_c^{-1} (x - \mu_c) \right]$$

这里 $x = (x_1, \dots, x_d)^T$ ,  $\mu_c = (\mu_{c1}, \dots, \mu_{cd})^T$ ,  $\Sigma_{d \times d}$  是一个协方差矩阵,  $|\Sigma_c|$  是协方差矩阵的行列式,  $\Sigma_c^{-1}$  是逆。

- ▶ 决策函数可以表达为(QDA):

$$g_c(x) = -\frac{1}{2} (x - \mu_c)^T \Sigma_c^{-1} (x - \mu_c) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma_c| + \ln P(c)$$

- ▶ 令 $r_c^2(x) = \frac{1}{2} (x - \mu_c)^T \Sigma_c^{-1} (x - \mu_c)$ ,

$$\delta(x) = \begin{cases} 1, & r_1^2(x) < r_0^2(x) + 2 \ln \frac{P(1)}{P(0)} + \ln \frac{|\Sigma_0|}{|\Sigma_1|} \\ 0, & otherwise \end{cases}$$

$P(0)$ 和 $p(1)$ 分别是0类和1类的先验概率



### 3.4.1 推广到多类

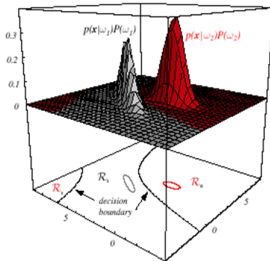
- ▶ 当多类 $\{1, \dots, C\}$ , 若 $p(x|c)$  是正态, Bayes Detection function
- ▶  $\delta(x) = \operatorname{argmax}(g_c(x))$

$$g_c(x) = -\frac{1}{2}(x - \mu_c)^T \Sigma_c^{-1}(x - \mu_c) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma_c| + \ln P(c)$$

- ▶ 用极大似然估计表示

$$\hat{g}_c(x) = -\frac{1}{2}(x - \hat{\mu}_c)^T \hat{\Sigma}_c^{-1}(x - \hat{\mu}_c) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\hat{\Sigma}_c| + \ln P(c)$$

$$\text{其中 } \hat{\mu}_c = \frac{1}{n_c} \sum_{i=1}^{n_c} x_{ci}, \quad \hat{\Sigma}_c = \frac{1}{n_c} \sum_{i=1}^{n_c} (x_{ci} - \hat{\mu}_c)(x_{ci} - \hat{\mu}_c)^T$$



**FIGURE 2.6.** In this two-dimensional two-category classifier, the probability densities are Gaussian, the decision boundary consists of two hyperbolas, and thus the decision region  $\mathcal{R}_2$  is not simply connected. The ellipses mark where the density is  $1/e$  times that at the peak of the distribution. From: Richard O. Duda, Peter E. Hart, and David G.

### 3.4.1 情形1 $\Sigma_c = \sigma^2 I$ , $I$ stands for the identity matrix(1)



$$g_c(x) = -\frac{\|x - \mu_c\|^2}{2\sigma^2} + \ln P(c);$$



$$\|x - \mu_c\|^2 = (x - \mu_c)^T (x - \mu_c) = x^T x - 2\mu_c^T x + \mu_c^T \mu_c;$$

- ▶  $g_c(x) = w_c^T x + w_{c0}$  (LDA: Linear Discriminant function 线性判别函数)

$$w_c = \frac{\mu_c}{\sigma^2}; \quad w_{c0} = -\frac{1}{2\sigma^2} \mu_c^T \mu_c + \ln P(c);$$

$w_{c0}$  是第  $c$  类的阈值

- ▶ 决策面 (The decision surfaces) for a linear machine are pieces of hyperplanes defined by:

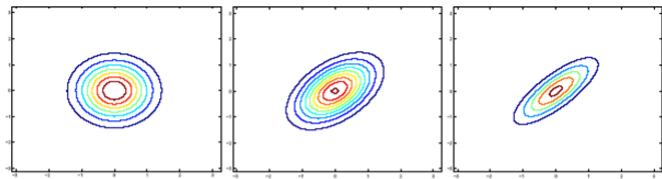
$$g_i(x) = g_j(x); i \neq j \in \{1, \dots, C\}$$

$$w^T (x - x_0) = 0$$

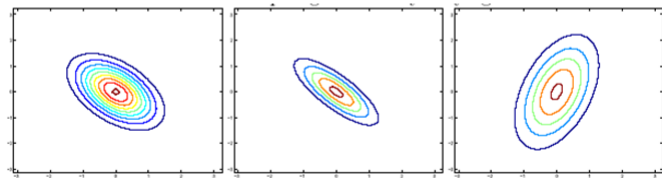
这里

$$w = \mu_i - \mu_j; \quad x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(i)}{P(j)} (\mu_i - \mu_j)$$

### 3.4.1 情形1 $\Sigma_c = \sigma^2 I$ , $I$ stands for the identity matrix(2)



$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$



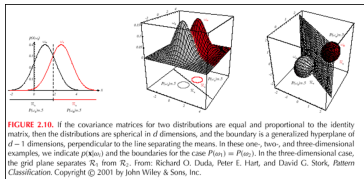
$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

### 3.4.1 情形1 $\Sigma_c = \sigma^2 I(3)$

- 用来分隔两个不同类*i*和*j*的决策面过一点 $x_0$

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(i)}{P(j)} (\mu_i - \mu_j)$$

- 如果 $P(i) = P(j)$ , 那么 $x_0 = \frac{1}{2}(\mu_i + \mu_j)$
- 先验概率相等和不等的分界面

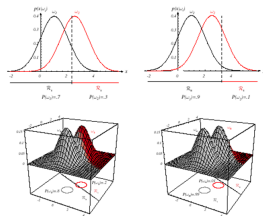
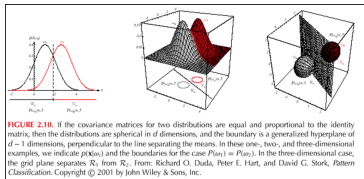


### 3.4.1 情形1 $\Sigma_c = \sigma^2 I(3)$

- 用来分隔两个不同类*i*和*j*的决策面过一点 $x_0$

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(i)}{P(j)} (\mu_i - \mu_j)$$

- 如果 $P(i) = P(j)$ , 那么 $x_0 = \frac{1}{2}(\mu_i + \mu_j)$
- 先验概率相等和不等的分界面



### 3.4.1 情形1 $\Sigma_c = \sigma^2 I$ 求二元正态分布的分界面(4)

$$\mu_1 = \begin{pmatrix} 5 \\ 0 \end{pmatrix}, \mu_2 = \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \Sigma = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$
$$p(\omega_1) = p(\omega_2) = 0.5$$

解答:

$$x = (u, v),$$

$$w^t(x - x_0) = 0$$

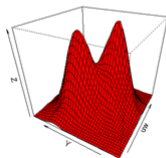
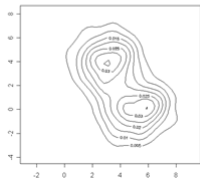
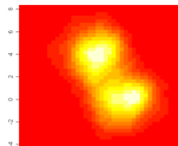
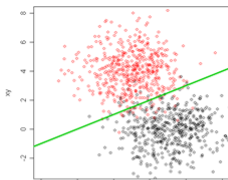
where:

$$w = \mu_1 - \mu_2 = (2, -4)^t$$

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(\omega_i)}{P(\omega_j)} (\mu_i - \mu_j) = (4, 2)^t$$

$$v = 0.5u$$

### 3.4.1 情形1 $\Sigma_c = \sigma^2 I$ 求二元正态分布的分界面(6)



### 3.4.1 情形2 $\Sigma_c = \Sigma$ 线性判别函数,协方差相等(1)



$$g_c(x) = -\frac{1}{2}(x - \mu_c)^T \Sigma^{-1}(x - \mu_c) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma| + \ln P(c);$$

- ▶  $g_c(x) = w_c^T x + w_{c0}$  (LDA: Linear Discriminant function 线性判别函数)

$$w_c = \Sigma^{-1} \mu_c; \quad w_{c0} = -\frac{1}{2} \mu_c^T \Sigma^{-1} \mu_c + \ln P(c);$$

- ▶  $w_c^T (x - x_0) = 0$  这里  $w = \Sigma^{-1}(\mu_i - \mu_j)$

- ▶ 用于区分第*i*类和第*j*类的分界面:

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln(P(i)/P(j))}{(\mu_i - \mu_j)^T \Sigma^{-1}(\mu_i - \mu_j)}(\mu_i - \mu_j)$$



### 3.4.1 情形2 $\Sigma_c = \Sigma(2)$

当 $\mu$ 和 $\Sigma$ 未知时,可以使用极大似然估计求解

$$\hat{\mu}_{MLE} = \frac{1}{n} \sum_{i=1}^n X_i, \quad \hat{\Sigma}_{MLE} = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})^t$$

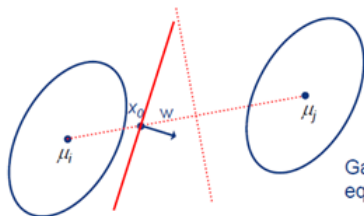
- for a classification problem with Gaussian classes of equal covariance  $\Sigma_i = \Sigma$ , the BDR boundary is the plane of normal

$$w = \Sigma^{-1}(\mu_i - \mu_j)$$

- if  $\Sigma_i = \Sigma_0$ , this is also the LDA solution

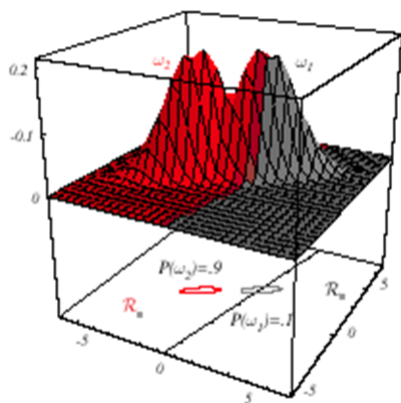
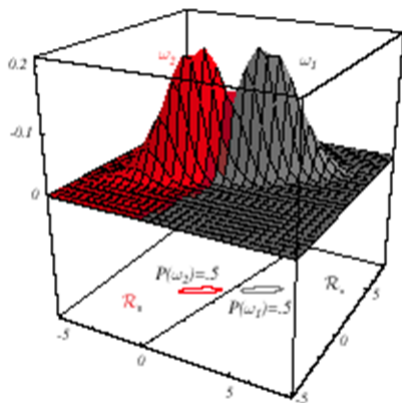
this gives two different interpretations of LDA

- it is optimal if and only if the classes are Gaussian and have equal covariance
- better than PCA, but not necessarily good enough
- a classifier on the LDA feature, is equivalent to
  - the BDR after the approximation of the data by two Gaussians with equal covariance



Gaussian classes,  
equal covariance  $\Sigma$

### 3.4.1 情形2 $\Sigma_c = \Sigma(3)$



### 3.4.1 情形2 $\Sigma_c = \Sigma(4)$

练习:求二元正态分布的分界面

$$\mu_1 = \begin{pmatrix} 5 \\ 0 \end{pmatrix}, \mu_2 = \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \Sigma = \begin{pmatrix} 1 & 2 \\ 2 & 9 \end{pmatrix}$$
$$P(w_1) = 0.7, P(w_2) = 0.3$$

解答提示:

$$g_i(x) = w_i^t x + w_{i0} \quad w_i = \Sigma^{-1} \mu_i$$
$$w_{i0} = -\frac{1}{2} \mu_i^t \Sigma^{-1} \mu_i + \ln P(w_i)$$
$$w_i^t (x - x_0) = 0$$

where :

$$w = \Sigma^{-1} (\mu_i - \mu_j)$$

$$x_0 = \frac{1}{2} (\mu_i + \mu_j) - \frac{\ln [P(\omega_i) / P(\omega_j)]}{(\mu_i - \mu_j)^t \Sigma^{-1} (\mu_i - \mu_j)} (\mu_i - \mu_j)$$

### 3.4.1 情形3 $\Sigma_c$ 任意(1)

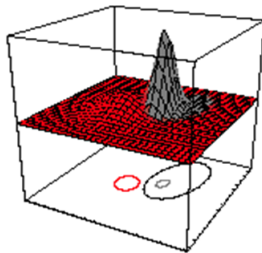
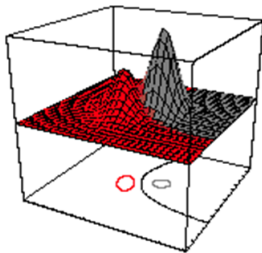
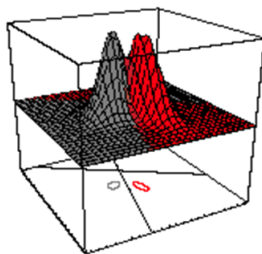
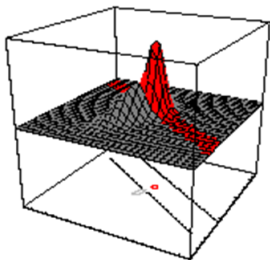
- ▶ 每一类的协方差矩阵都不同

$$g_c(x) = x^T W_c x + w_c^T x + w_{c0}$$

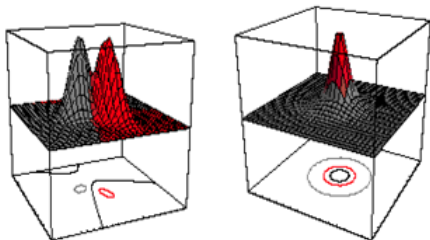
这里

- ▶  $W_c = -\frac{1}{2}\Sigma_c^{-1}$ ;
- ▶  $w_c = \Sigma_c^{-1}\mu_c$ ;
- ▶  $w_{c0} = -\frac{1}{2}\mu_c^T \Sigma_c^{-1} \mu_c - \frac{1}{2} \ln |\Sigma_c| + \ln P(c)$ ;
- ▶ 超二次曲面Hyperquadrics which are: hyperplanes, pairs of hyperplanes, hyperspheres, hyperellipsoids, hyperparaboloids, hyperhyperboloids

### 3.4.1 情形3 $\Sigma_c$ 任意(2)



### 3.4.1 情形3 $\Sigma_c$ 任意(3)



**FIGURE 2.14.** Arbitrary Gaussian distributions lead to Bayes decision boundaries that are general hyperquadrics. Conversely, given any hyperquadric, one can find two Gaussian distributions whose Bayes decision boundary is that hyperquadric. These variances are indicated by the contours of constant probability density. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

### 3.4.1 情形3 $\Sigma_c = \text{任意}(4)$

练习:求二元正态分布的分界面

$$\mu_1 = \begin{pmatrix} 3 \\ 6 \end{pmatrix}, \mu_2 = \begin{pmatrix} 3 \\ -2 \end{pmatrix}, \Sigma_1 = \begin{pmatrix} 0.5 & 0 \\ 0 & 2 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, p(\omega_1) = p(\omega_2)$$

解答提示:

$$g_i(x) = x^T W_i x + w_{i0}^T x + w_{i0}$$

where :

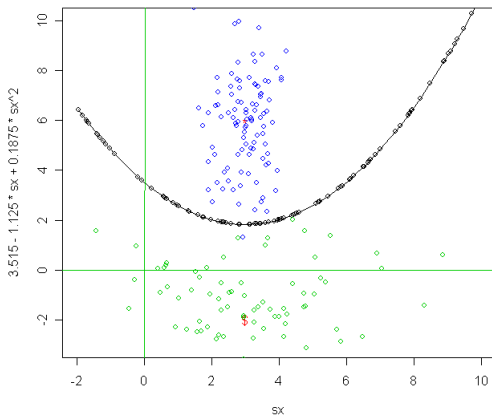
$$W_i = -\frac{1}{2} \Sigma_i^{-1}$$

$$w_i = \Sigma_i^{-1} \mu_i$$

$$w_{i0} = -\frac{1}{2} \mu_i^T \Sigma_i^{-1} \mu_i - \frac{1}{2} \ln |\Sigma_i| + \ln P(\omega_i)$$

### 3.4.1 情形3 $\Sigma_c = \text{任意}(4)$

$$x_2 = 3.514 - 1.125x_1 + 0.1875x_1^2$$



练习:求二元正态分布的分界面

$$\mu_1 = \begin{pmatrix} 3 \\ 6 \end{pmatrix}, \mu_2 = \begin{pmatrix} 3 \\ -2 \end{pmatrix}, \Sigma_1 = \begin{pmatrix} 0.5 & 0 \\ 0 & 2 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, P(\omega_1) = P(\omega_2)$$

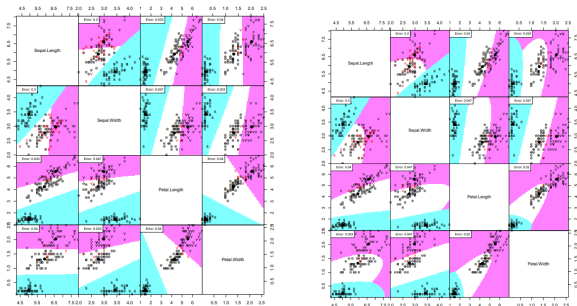
解答提示:

$$\begin{aligned} \mathbf{g}(\mathbf{x}) &= \mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{w}^T \mathbf{x} + w_0 \\ \text{where: } & \\ \mathbf{W} &= -\frac{1}{2} \Sigma^{-1} \\ \mathbf{w} &= \Sigma_1^{-1} \mu_1 \\ w_0 &= -\frac{1}{2} \mu_1^T \Sigma_1^{-1} \mu_1 - \frac{1}{2} \ln \left| \Sigma_1 \right| + \ln P(\omega_1) \end{aligned}$$



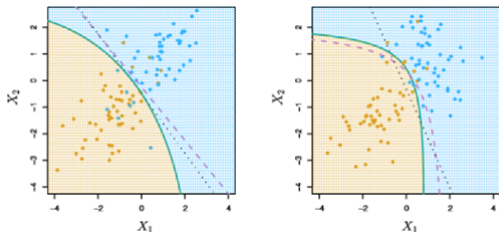
# LDA at example IRIS data

LDA(左)和QDA(右), 会发现并非每个变量都有效果(第3个变量和第4个变量用来区分不同类, 作用明显), 这样是不是可以通过降维的方式获得视角空间, 估计权值, 再找出代表性的数据点



### 3.4.1 总结

- ▶ QDA will work best when the variances are very different between classes and we have enough observations to accurately estimate the variances.
- ▶ LDA will work best when the variances are similar among classes or we don't have enough data to accurately estimate the variances.



# LDA under multivariate normal model

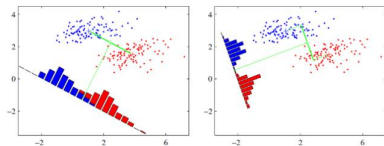
W.l.o.g., we assume data have been centered:

$$X|y \sim \begin{cases} \mathcal{N}_p(\mu, \Sigma), & \text{if } y = +1, \\ \mathcal{N}_p(-\mu, \Sigma), & \text{if } y = -1. \end{cases}$$

- ▶ Training data: *iid* samples from both classes
- ▶ Goal: A linear classifier which predicts label of  $X$  by  $\text{sgn}(\hat{\eta}^\top X)$ , where  $\hat{\eta}$  is from training data
- ▶ Bayes classifier  $\implies \eta \propto \Sigma^{-1}\mu$
- ▶ **Key:** Estimate  $\Sigma^{-1}\mu$  from training data

## 3.4.2 线性判别分析(1)

- ▶ 线性判别分析Linear Discriminant Analysis, LDA, 是一种经典的线性学习方法, 二分类问题上最早由Fisher,1936 提出, 亦称“Fisher 判别分析”。
- ▶ LDA的思想: 给定训练样例集, 设法将样例投影到一条直线上使得同类样例的投影点尽可能接近、异类样例的投影点尽可能远离, 在对新样本进行分类时, 将其投影到同样的这条直线上, 再根据投影点的位置来确定新样本的类别。
- ▶ 给定数据集 $D = \{(x_i, y_i)_{i=1}^m, y_i = \{0, 1\}\}$ , 假设有两类数据, 分别为红色和蓝色, 如图所示, 这些数据特征是二维的, 希望将这些数据投影到一维的一条直线, 让每一种类别数据的投影点尽可能的接近, 而红色和蓝色数据中心之间的距离尽可能大。



$$J = \frac{\|w^T \mu_0 - w^T \mu_1\|_2^2}{w^T \Sigma_0 w + w^T \Sigma_1 w}$$

## 3.4.2 线性判别分析2

- ▶ 最优化目标函数

$$\begin{aligned} J &= \frac{\|\mathbf{w}^T \mu_0 - \mathbf{w}^T \mu_1\|_2^2}{\mathbf{w}^T \Sigma_0 \mathbf{w} + \mathbf{w}^T \Sigma_1 \mathbf{w}} \\ &= \frac{\mathbf{w}^T (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T \mathbf{w}}{\mathbf{w}^T (\Sigma_0 + \Sigma_1) \mathbf{w}} \end{aligned}$$

- ▶ 类内散度矩阵

$$\begin{aligned} S_w &= \Sigma_0 + \Sigma_1 \\ &= \sum_{x \in X_0} (x - \mu_0)(x - \mu_0)^T + \sum_{x \in X_1} (x - \mu_1)(x - \mu_1)^T \end{aligned}$$

- ▶ 类间散度矩阵

$$S_b = (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T.$$

- ▶ 最优化目标函数写作：

$$J = \frac{\mathbf{w}^T S_b \mathbf{w}}{\mathbf{w}^T S_w \mathbf{w}}$$

LDA 最优化目标，经常称为“广义瑞丽商”（generalized Rayleigh Quotient）

### 3.4.2 线性判别分析求解3

- ▶ 不失一般性，假设  $\mathbf{w}^T \mathbf{S}_w \mathbf{w} = 1$ ，于是目标函数等价于

$$\min -\mathbf{w}^T \mathbf{S}_b \mathbf{w}, \quad \text{s.t. } \mathbf{w}^T \mathbf{S}_w \mathbf{w} = 1$$

- ▶ 由拉格朗日乘子法，上式等价于

$$\mathbf{S}_b \mathbf{w} = \lambda \mathbf{S}_w \mathbf{w}$$

- ▶ 其中  $\lambda$  是拉格朗日乘子因子，注意到  $\mathbf{S}_b \mathbf{w}$  的方向恒为  $\mu_0 - \mu_1$ ，不妨令

$$\mathbf{S}_b \mathbf{w} = \lambda(\mu_0 - \mu_1).$$

- ▶ 于是

$$\mathbf{w} = \mathbf{S}_w^{-1}(\mu_0 - \mu_1)$$

### 3.4.2 线性判别分析多分类3

- ▶ 考虑到数值解的稳定性，在实践中常常是对  $\mathbf{S}_w$  进行奇异值分解化为对角矩阵， $\mathbf{S}_w = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ ,  $\mathbf{S}_w^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T$

假定有  $N$  个类

□ 全局散度矩阵  $\mathbf{S}_t = \mathbf{S}_b + \mathbf{S}_w = \sum_{i=1}^m (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T$

□ 类内散度矩阵  $\mathbf{S}_w = \sum_{i=1}^N \mathbf{S}_{w_i}$      $\mathbf{S}_{w_i} = \sum_{\mathbf{x} \in X_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T$

□ 类间散度矩阵  $\mathbf{S}_b = \mathbf{S}_t - \mathbf{S}_w = \sum_{i=1}^N m_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$

多分类LDA有多种实现方法：采用  $\mathbf{S}_b$ ,  $\mathbf{S}_w$ ,  $\mathbf{S}_t$  中的任何两个

例如,  $\max_{\mathbf{W}} \frac{\text{tr}(\mathbf{W}^T \mathbf{S}_b \mathbf{W})}{\text{tr}(\mathbf{W}^T \mathbf{S}_w \mathbf{W})} \implies \mathbf{S}_b \mathbf{W} = \lambda \mathbf{S}_w \mathbf{W}$

$$\mathbf{W} \in \mathbb{R}^{d \times (N-1)}$$

$\mathbf{W}$  的闭式解是  $\mathbf{S}_w^{-1} \mathbf{S}_b$  的  $N-1$  个最大广义特征值所对应的特征向量组成的矩阵

## 3.4.2 举例1/3

- ▶ iris案例数据，有三个类别，每个类别观察了50例，目标是寻找一个线性决策面，可以将三个类比较完整地分开



- ▶ 加载数据集分析数据loaddata() 结果返回特征向量矩阵X以及类别列向量y

```
1 def loaddata():
2     feature_dict = {i:label for i,label in zip(
3         range(4),
4         ('sepal length in cm',
5          'sepal width in cm',
6          'petal length in cm',
7          'petal width in cm'))}
8
9     df = pd.io.parsers.read_csv(
10         filepath_or_buffer='https://archive.ics.uci.edu/m
11         header=None,
12         sep=',',
13
14         df.columns = [i for i,l in sorted(feature_dict.items())]
15         df.dropna(how='all', inplace=True) # to drop the empty I
16         df.tail()
17
18     from sklearn.preprocessing import LabelEncoder
19     # 取出特征向量矩阵
20     X = df[['sepal length in cm',
21            'sepal width in cm',
22            'petal length in cm',
23            'petal width in cm']].values
24     # 取出类别向量
25     y = df['class label'].values
26     enc = LabelEncoder()
27     label_encoder = enc.fit(y)
28     y = label_encoder.transform(y) + 1
29     return X,y
```



## 3.4.2 举例2/3

- ▶ 根据每个变量制作直方图，并以不同颜色标记不同的类别

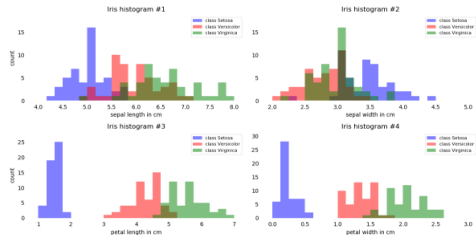
```
1 def showdata(X, y):
2     from matplotlib import pyplot as plt
3     import numpy as np
4     import math
5     label_dict = {1: 'Setosa', 2: 'Versicolor', 3: 'Virginica'}
6     fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12,6))
7     for ax,cnt in zip(axes.ravel(), range(4)):
8         # get data lines
9         min_b = math.floor(np.min(X[:,cnt]))
10        max_b = math.ceil(np.max(X[:,cnt]))
11        bins = np.linspace(min_b, max_b, 25)
12        # plotting the histograms
13        for lab,col in zip(range(1,4), ('blue', 'red', 'green')):
14            ax.hist(X[y==lab, cnt],
15                  color=col,
16                  label='class %s' %label_dict[lab],
17                  bins=bins,
18                  alpha=0.5,)
19        ylims = ax.get_ylim()
20        # plot annotation
21        leg = ax.legend(loc='upper right', fancybox=True, for
22        ax.set_ylim([0, max(ylims)+2])
23        ax.set_xlabel(feature_dict[cnt])
24        ax.set_title('Iris histogram # %s' %str(cnt+1))
25        # hide axis ticks
26        ax.tick_params(axis="both", which="both", bottom="off",
27                      labelbottom="on", left="off", right="off")
28        # remove axis spines
29        ax.spines["top"].set_visible(False)
30        ax.spines["right"].set_visible(False)
31        ax.spines["bottom"].set_visible(False)
32        ax.spines["left"].set_visible(False)
33        axes[0][0].set_ylabel('count')
34        axes[1][0].set_ylabel('count')
35        fig.tight_layout()
36        plt.show()
```

- 根据每个变量制作直方图，并以不同颜色标记不同的类别

```

1 def showdata(X, y):
2     from matplotlib import pyplot as plt
3     import numpy as np
4     import math
5     label_dict = {'1':'Setosa', '2':'Versicolour', '3':'Virginica'}
6     fig, axes = plt.subplots(nrows=2, ncol=1, figsize=(12,6))
7     for ax,cnt in zip(axes.ravel(), range(4)):
8         # set bin sizes
9         min_b = math.floor(np.min(X[:,cnt]))
10        max_b = math.ceil(np.max(X[:,cnt]))
11        bins = np.linspace(min_b, max_b, 25)
12        # plotting the histograms
13        for lab,col in zip(range(1,4), ('blue', 'red', 'green')):
14            ax.hist(X[:,y==lab], cnt,
15                    color=col,
16                    label='class %s' %label_dict[lab],
17                    bins=bins,
18                    alpha=0.5,)
19        ylims = ax.get_ylim()
20        # plot annotation
21        leg = ax.legend(loc='upper right', fancybox=True, for
22        leg.set_frame().set_alpha(0.5)
23        ax.set_ylim([0, max(ylims)-2])
24        ax.set_xlabel('feature dict[cnt]')
25        ax.set_title('Iris histogram %s' %str(cnt+1))
26        # tick axis label
27        ax.tick_params(axis="both", which="both", bottom="off",
28                        labelbottom="on", left="off", right="off")
29        # remove axis spines
30        ax.spines["top"].set_visible(False)
31        ax.spines["right"].set_visible(False)
32        ax.spines["bottom"].set_visible(False)
33        ax.spines["left"].set_visible(False)
34        axes[0][0].set_ylabel('count')
35        axes[1][0].set_ylabel('count')
36        fig.tight_layout()
37        plt.show()

```



## 3.4.2 举例(LDA)3/3

- ▶ Step 1: Computing the d-dimensional mean vectors 计算各类别均值 $\mu_i$ ;

```
1 def meanvector(X, y):
2     np.set_printoptions(precision=4)
3     mean_vectors = []
4     for cl in range(1,4):
5         mean_vectors.append(np.mean(X[y==cl], axis=0))
6         print('Mean Vector class %s: %s\n' % (cl, mean_vectors[cl]))
7     return mean_vectors
```

- ▶ Step 2: Computing the Scatter Matrices 计算散度矩阵,类内散度矩阵计算 $S_w$ , 类间散度矩阵 $S_b$ ;
- ▶ Step 3: Solving the generalized eigenvalue problem for the matrix 求特征值分解;

```
1 def within_class_scatter(X, y, mean_vectors):
2     S_W = np.zeros((4,4))
3     for cl, mv in zip(range(1,4), mean_vectors):
4         class_sc_mat = np.zeros((4,4)) # scatter for this class
5         for row in X[y == cl]:
6             row, mv = row.reshape(4,1), mv.reshape(4,1) # make column vectors
7             class_sc_mat += (row-mv).dot((row-mv).T) # sum of squares
8         S_W += class_sc_mat # sum for this class
9     print('within-class Scatter Matrix:\n', S_W)
10    return S_W
```

```
1 def between_class_scatter(X, y, mean_vectors):
2     overall_mean = np.mean(X, axis=0)
3     S_B = np.zeros((4,4))
4     for i, mean_vec in enumerate(mean_vectors):
5         n = X[y==i+1,:].shape[0]
6         mean_vec = mean_vec.reshape(4,1) # make column vector
7         overall_mean = overall_mean.reshape(4,1) # make column vector
8         S_B += n * (mean_vec - overall_mean).dot((mean_vec - overall_mean).T)
9     print('between-class Scatter Matrix:\n', S_B)
10    return S_B
```

## 3.4.2 举例(LDA)

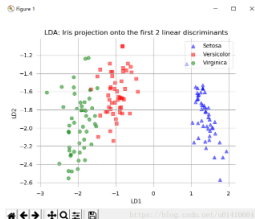
- ▶ Step 4: Selecting linear discriminants for the new feature subspace;
- ▶ 返回最大的 $k$ 个特征值对应的特征向量组成的矩阵，即为 $W$ 法向量子空间；
- ▶ Step 5: Transforming the samples onto the new subspace, 将样本转为到 $W$ 的向量空间内，实现降维操作,并绘出样本分布图

```
1 def eigenvalue(S_W, S_B):  
2     eig_vals, eig_vecs = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))  
3     for i in range(len(eig_vals)):  
4         eigvec_sc = eig_vecs[:,i].reshape(4,1)  
5         print('\nEigenvector {}: \n{}'.format(i+1, eigvec_sc.real))  
6         print('Eigenvalue {}: {:.2e}'.format(i+1, eig_vals[i].real))  
7     return eig_vals, eig_vecs
```

## 3.4.2 举例(LDA)

- ▶ Step 4: Selecting linear discriminants for the new feature subspace;
- ▶ 返回最大的 $k$ 个特征值对应的特征向量组成的矩阵，即为 $W$ 法向量子空间；
- ▶ Step 5: Transforming the samples onto the new subspace, 将样本转为到 $W$ 的向量空间内，实现降维操作,并绘出样本分布图

```
1 def eigenvalue(S_W, S_B):  
2     eig_vals, eig_vecs = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))  
3     for i in range(len(eig_vals)):  
4         eigvec_sc = eig_vecs[:,i].reshape(4,1)  
5         print('\nEigenvector {}: {}'.format(i+1, eigvec_sc.real))  
6         print('Eigenvalue {}: {}'.format(i+1, eig_vals[i].real))  
7     return eig_vals, eig_vecs
```



## 4.3 判别式模型的基本代表-Logistic回归

假设连续情形下,

- ▶ 对于二分类问题 $\{C_1, C_2\}$  而言,

$$\begin{aligned} p(C_1|x) &= \frac{p(x, C_1)}{p(x)} = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} \\ &= \frac{1}{1 + \exp(-a(x))} = \sigma(a); \end{aligned}$$

- ▶ 对数优势比函数(log odds):  $a(x) = \ln \frac{p(C_1|x)}{p(C_2|x)}$ ;
- ▶  $\sigma(a)$  逻辑挤压变形(Logistic sigmoid) 函数:  $\sigma(a) = \frac{1}{1+\exp(-a)}$ ;
- ▶  $\sigma(a)$  的性质: 对称性 $\sigma(-a) = 1 - \sigma(a)$ ;  $a = \ln(\frac{\sigma}{1-\sigma})$ ;
- ▶ 对于多分类问题 $K > 2$ ,  $\{C_1, \dots, C_K\}$ ,  $k \in 1, \dots, K$  而言,

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum p(x|C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)};$$

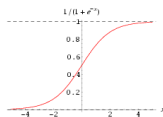
- ▶  $a_k = \ln p(x|C_k)p(C_k)$ ;

# Logistic 回归1

如果要进行的是分类任务，就需要将线性回归模型的预测值与分类任务的真实标记联系起来。

- ▶ 对于二分类任务来说，输出  $y \in \{0, 1\}$ ，线性回归模型产生的预测值是实值，需要将实值转换为0/1 值。最理想的方式就是给预测值加上一个单位阶跃函数，若预测值大于0 就判为正类，小于0 则判为负类，预测值为临界值0 则判别为任意。

$$y = \begin{cases} 0, & z \leq 0 \\ 0.5 & z = 0 \\ 1, & z > 0 \end{cases}$$



- ▶ 阶跃函数的缺点：不连续，无法微分，于是找到替代函数，对数几率函数(S 型函数)

$$y = \frac{1}{1 + e^{-x}}.$$

## 算法

- ▶ 初始化模型，输

入  $(x, y) = ((x_1, y_1), \dots, (x_m, y_m)) \in R^{m \times d}, x_i \in \mathbb{R}^d, y_i \in \{0, 1\}$ .

- ▶ 输出过程:

- ▶ 初始化模型参数  $w \in \mathbb{R}^d, b \in \mathbb{R}$
- ▶ 建立逻辑回归模型

$$p_1(x) = p(y = 1|x) = \frac{e^{w^T x + b}}{1 + e^{w^T x + b}}; \quad p(y = 0|x) = \frac{1}{1 + e^{w^T x + b}}.$$

令  $\beta = (w, b), \tilde{x}_i = (x_i, 1)$ , 那么

$$p_1(x) = p(y = 1|x) = \frac{e^{\beta^T \tilde{x}}}{1 + e^{\beta^T \tilde{x}}}; \quad p(y = 0|x) = \frac{1}{1 + e^{\beta^T \tilde{x}}}.$$

$$\Rightarrow \ln \frac{p_1}{1 - p_1} = \beta^T \tilde{x} \quad \ln(1 - p_1) = -\ln(1 + e^{\beta^T \tilde{x}})$$

- ▶ 两点分布  $P(y_i|x_i, \beta) = p_1^{y_i}(1 - p_1)^{1-y_i}$ ;

$$\ln P(y_i|x_i, \beta) = y_i \ln p_1 + (1 - y_i) \ln(1 - p_1) = y_i \left( \ln \frac{p_1}{1 - p_1} \right) + \ln(1 - p_1).$$



- ▶ 记  $l(\beta) = \ln \prod_{i=1}^m P(y_i|x_i, \beta)$  为对数似然函数;
- ▶ 负对数似然函数  $-l(\beta) = \sum_{i=1}^m (-y_i \beta^T \tilde{x}_i + \ln(1 + \exp \beta^T \tilde{x}_i))$
- ▶ 负对数似然函数为**损失函数**, 为求最小值,  $l(\beta)$  对  $\beta$  求导数

$$\frac{\partial l(\beta)}{\partial \beta_j} = \sum_{i=1}^m \tilde{x}_{ij}(y_i - p_1(\tilde{x}_i; \beta));$$

$$\frac{\partial^2 l(\beta)}{\partial \beta_j \partial \beta_k^T} = \sum_{i=1}^m \tilde{x}_i \tilde{x}_i^T p_1(\tilde{x}_i; \beta)(1 - p_1(\tilde{x}_i; \beta))$$

- ▶ 注意:以上求导过程中,  $\ln(1 + \beta^T \tilde{x}_i)$  对  $\beta_j$  求导时,  $p(x_i; \beta) = \frac{e^{\beta^T \tilde{x}}}{1 + e^{\beta^T \tilde{x}}}$
- ▶ 特别的,

$$\frac{\partial^2 l(\beta)}{\partial \beta_0 \partial \beta_j^T} = \sum_{i=1}^m \tilde{x}_i p_1(\tilde{x}_i; \beta)(1 - p_1(\tilde{x}_i; \beta))$$

▶

$$\frac{\partial^2 l(\beta)}{\partial \beta_0 \partial \beta_0^T} = \sum_{i=1}^m p_1(\tilde{x}_i; \beta)(1 - p_1(\tilde{x}_i; \beta))$$

# 梯度下降法gradient descent最速下降法steepest descent, (1)参见李航附录A

- ▶  $f(x)$  是 $\mathbb{R}^m$  上有一阶连续偏导数的函数, 要求其无约束最优化问题

$$\min_{x \in \mathbb{R}^m} f(x), \quad x^* = \operatorname{argmin}_{x \in \mathbb{R}^m} f(x).$$

- ▶ 梯度下降法的基本原理: 根据泰勒展开,

$$f(x) = f(x^{(k)}) + g_k^T(x - x^{(k)}). g_k = \nabla f(x^{(k)}) \text{称为梯度}$$

- ▶ 选取适当的初值 $x^{(0)}$ , 不断按照以下公式迭代, 直到收敛:

$$x^{(k+1)} \leftarrow x^{(k)} + \lambda_k p_k.$$

- ▶ 其中 $p_k$  是搜索方向, 取负梯度方向 $p_k = -\nabla f(x^{(k)})$ ,  $\lambda_k$  是步长, 由一维搜索确定, 即 $\lambda_k$  使得

$$f(x^{(k)} + \lambda_k p_k) = \min_{\lambda \geq 0} f(x^{(k)} + \lambda p_k)$$

## 梯度下降法gradient descent最速下降法steepest descent(2)

- ▶ **输入**: 目标函数 $f(x)$ , 梯度函数 $g(x) = \nabla f(x)$ , 计算精度 $\epsilon$ .
- ▶ **输出**:  $f(x)$  的极小点 $x^*$ .

1. 取初始值 $x^{(0)}$ , 置 $k = 0$
2. 计算 $f(x^{(k)})$
3. 计算梯度 $g_k = g(x^{(k)})$ , 当 $\|g_k\| \leq \epsilon$  时, 停止迭代, 令 $x^* = x^{(k)}$ ; 否则, 令 $p_k = -g(x^{(k)})$ , 求 $\lambda_k$  使得

$$f(x^{(k)} + \lambda_k p_k) = \min_{\lambda \geq 0} f(x^{(k)} + \lambda p_k)$$

4. 置 $x^{(k+1)} \leftarrow x^{(k)} + \lambda_k p_k$ , 计算 $f(x^{(k+1)})$   
当 $\|f(x^{(k+1)}) - f(x^{(k)})\| < \epsilon$  或 $\|x^{(k+1)} - x^{(k)}\| < \epsilon$  时, 停止迭代, 令 $x^* = x^{(k+1)}$
5. 否则, 置 $k = k + 1$  转(3)

# 牛顿法和拟牛顿法(1)

- ▶  $f(x)$  是  $\mathbb{R}^m$  上有二阶连续偏导数的函数, 要求其无约束最优化问题

$$\min_{x \in \mathbb{R}^m} f(x), \quad x^* = \operatorname{argmin}_{x \in \mathbb{R}^m} f(x).$$

- ▶ 根据泰勒展开,

$$f(x) = f(x^{(k)}) + g_k^T(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})H(x^{(k)})(x - x^{(k)}).$$

$g_k = \nabla f(x^{(k)})$  是  $f(x)$  的梯度在点  $x^{(k)}$  的值,  $H(x^{(k)})$  是  $f(x)$  的海森矩阵(Hessen Matrix)

$$H(x) = \left[ \frac{\partial^2 f}{\partial x_i \partial x_j} \right]_{n \times n}$$

- ▶ 牛顿法利用了极小点的必要条件是  $\nabla f(x) = 0$ , 每次迭代从点  $x^{(k)}$  开始, 极小点满足  $\nabla f(x^{(k+1)}) = 0$
- ▶ 由泰勒展开:

$$\nabla f(x) = g_k + H_k(x - x^{(k)}) \Rightarrow g_k + H_k(x^{(k+1)} - x^{(k)}) = 0$$

## 牛顿法和拟牛顿法(2)

- ▶ 因此,  $x^{(k+1)} = x^{(k)} - H_k^{-1} g_k \Leftrightarrow x^{(k+1)} = x^{(k)} + p_k$
- ▶ **输入:** 目标函数 $f(x)$ , 梯度函数 $g(x) = \nabla f(x)$ , 海森矩阵 $H(x)$ , 计算精度 $\epsilon$ .
- ▶ **输出:**  $f(x)$  的极小点 $x^*$ .
  1. 取初始值 $x^{(0)}$ , 置 $k = 0$ ;
  2. 计算 $g_k = g(x^{(k)})$ ;
  3. 若 $H_k = H(x^{(k)})$ , 并求 $p_k$ ,

$$H_k p_k = -g_k;$$

4. 置 $x^{(k+1)} = x^{(k)} + p_k$ ,
5. 否则, 置 $k = k + 1$  转(2)

$$p_k = -H_k^{-1} g_k.$$

## 算法

- ▶ 用凸优化理论:  $\beta^{t+1} = \beta^t - \frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T}^{-1} \frac{\partial l(\beta)}{\partial \beta}$
- ▶ 其中关于  $\beta$  的一阶二阶导数分别为:

$$\frac{\partial l(\beta)}{\partial \beta_j} = \sum_{i=1}^m \tilde{x}_{ij}(y_i - p_1(\tilde{x}_i; \beta));$$
$$\frac{\partial^2 l(\beta)}{\partial \beta_j \partial \beta_k^T} = \sum_{i=1}^m \tilde{x}_i \tilde{x}_i^T p_1(\tilde{x}_i; \beta)(1 - p_1(\tilde{x}_i; \beta))$$

- ▶ 特别的,

$$\frac{\partial^2 l(\beta)}{\partial \beta_0 \partial \beta_j^T} = \sum_{i=1}^m \tilde{x}_i p_1(\tilde{x}_i; \beta)(1 - p_1(\tilde{x}_i; \beta))$$

▶

$$\frac{\partial^2 l(\beta)}{\partial \beta_0 \partial \beta_0^T} = \sum_{i=1}^m p_1(\tilde{x}_i; \beta)(1 - p_1(\tilde{x}_i; \beta))$$

# 迭代重加权平方Iterative reweighted least squares(IRIS)

设计用牛顿法优化。先计算Hessian矩阵

$$\mathbf{H} = \nabla^2 E(\mathbf{w}) = \sum_{n=1}^N y_n(1 - y_n) \phi_n \phi_n^T = \Phi^T \mathbf{R} \Phi$$

其中 $\mathbf{R}$ 是对角矩阵,  $\mathbf{R}_{nn} = y_n(1 - y_n)$ . 注意 $\mathbf{H}$ 正定.

$$\mathbf{w}^{(new)} = \mathbf{w}^{(old)} - \mathbf{H}^{-1} \nabla E(\mathbf{w}) = (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T \mathbf{R} \mathbf{z}$$

其中

$$\mathbf{z} = \Phi \mathbf{w}^{(old)} - \mathbf{R}^{-1}(\mathbf{y} - \mathbf{t})$$

注意, 每次迭代,  $\mathbf{R}$ 要重新计算, 实际上 $R$ 可以被解释为预测值的协方差

$$\begin{aligned} \mathbb{E}[t] &= \sigma(\mathbf{w}^T \phi) = y \\ \text{var}[t] &= \mathbb{E}[t^2] - \mathbb{E}[t]^2 = \sigma(\mathbf{w}^T \phi) - \sigma(\mathbf{w}^T \phi)^2 = y(1 - y) \end{aligned}$$

# 迭代重加权平方的矩阵表示(IRIS)

- Hessian Matrix

- ... 
$$\frac{\partial l(\beta)}{\partial \beta_j} = \sum_{i=1}^n x_{ij} (y_i - p(x_i; \beta)) = 0$$

$$X^T S = 0 \quad S = y - \pi(y)$$

$$\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} = \sum_{i=1}^n x_i x_i^T p(x_i; \beta) (1 - p(x_i; \beta))$$

$$\frac{dX^T S}{d\beta} = -X^T V X^T$$

$$\frac{\partial^2 l(\beta)}{\partial \beta_0 \partial \beta_j^T} = \frac{\partial^2 l(\beta)}{\partial \beta_j \partial \beta_0^T} = \sum_{i=1}^n x_i p(x_i; \beta) (1 - p(x_i; \beta))$$

$$\frac{\partial^2 l(\beta)}{\partial \beta_0 \partial \beta_0^T} = \sum_{i=1}^n p(x_i; \beta) (1 - p(x_i; \beta))$$

$$\beta_i^{(t+1)} = \beta_i^{(t)} + \left( \frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial l(\beta)}{\partial \beta}$$

$$\begin{aligned} \hat{\beta}^{(t+1)} &= \hat{\beta}^{(t)} + (X^T V X)^{-1} X^T S \\ &= (X^T V X)^{-1} X^T V (Z^t) \end{aligned}$$



# 计算后检验

In Gaussian case:

$$\hat{\beta} = (X'X)^{-1}X'Y, \quad Y \sim N(X\beta, \sigma^2 I).$$

Since Gaussian vectors remain Gaussian under linear transforms,

$$\hat{\beta} \sim N(\beta, (X'X)^{-1}\sigma^2).$$

For logistic regression,  $\hat{\beta}$  is no longer linear in  $Y$ . However, *asymptotically* (i.e.  $n$  large), it is Gaussian. It's covariance can be estimated by

$$\widehat{\text{cov}}(\hat{\beta}) = (X' \text{diag}[\hat{\pi}_i(1 - \hat{\pi}_i^{(t)})]X)^{-1}.$$

From the square root of the diagonal elements of the above matrix you can get s.e.( $\hat{\beta}$ ).

**Example 10** We apply the logistic regression on the Coronary Risk-Factor Study (CORIS) data and yields the following estimates and Wald statistics  $W_j$  for the coefficients:

Covariate	$\hat{\beta}_j$	se	$W_j$	p-value
Intercept	-6.145	1.300	-4.738	0.000
sbp	0.007	0.006	1.138	0.255
tobacco	0.079	0.027	2.991	0.003
ldl	0.174	0.059	2.925	0.003
adiposity	0.019	0.029	0.637	0.524
famhist	0.925	0.227	4.078	0.000
typea	0.040	0.012	3.233	0.001
obesity	-0.063	0.044	-1.427	0.153
alcohol	0.000	0.004	0.027	0.979
age	0.045	0.012	3.754	0.000

## Logistic回归的做法步骤总结:

- ▶ (表示) 取一个二项分布的似然函数, 再最大似然函数, 转换成求最小二乘法;
- ▶ (求解) 再求导出 $\beta$  向量的解析解, 用梯度下降, 牛顿等方法估计参数 $\beta$  的最优解;
- ▶ (不足) 线性模型重点就是放在求某个参数上。但是, 当数据量少的时候, **logistic** 回归的点估计很容易造成过拟合**overfitting**。
- ▶ (解决) **贝叶斯估计**: 估计目标是一个分布。而不是一个最优化的值 $\beta_{MAP}$ ), 通过似然函数 $\times$  先验求出后验概率分布之后, 再用它去积分进行类别预测, 考虑的是全局的所有 $\beta$ , 这样就可以消除过拟合。

## 3.5 多分类学习

LDA算法既可以用来降维，又可以用来分类，但是目前来说，主要还是用于降维。在进行图像识别相关的数据分析时，LDA 是一个有力的工具。

优点:

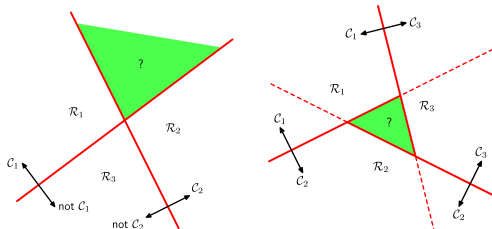
- ▶ 在降维过程中可以使用类别的先验知识经验;
- ▶ LDA在样本分类信息依赖均值而不是方差。

缺点:

- ▶ LDA不适合对非高斯分布样本进行降维。
- ▶ LDA 降维最多降到类别数 $k - 1$ 的维数，如果降维的维度大于 $k - 1$ ，则不能使用LDA。当然目前有一些LDA 的进化版算法可以绕过这个问题。
- ▶ LDA 在样本分类信息依赖方差而非均值时，降维效果不好。
- ▶ LDA 可能过度拟合数据。

# Discriminant functions for $N > 2$ classes

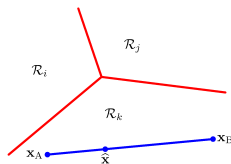
- ▶ One possibility is to use  $N$  two-way discriminant functions. Each function discriminates one class from the rest.
- ▶ Another possibility is to use  $N(N - 1)/2$  two-way discriminant functions, Each function discriminates between two particular classes.
- ▶ Both these methods have problems



# A simple solution

- 使用  $K$  个分类函数  $g_1(x), \dots, g_k(x)$ , 每个线性判别函数有下面的形式

$$g_k(x) = w_k^T x + w_{k0}$$



**Figure:** 如果两个点在同一个决策区域，决策区域满足单连通和凸性质

- 如果  $g_k(x_A) > g_l(x_A), g_k(x_B) > g_l(x_B), \forall \hat{x} = \lambda x_A + (1 - \lambda)x_B, 0 \leq \lambda \leq 1$ ,

$$g_k(\hat{x}) > g_l(\hat{x})$$

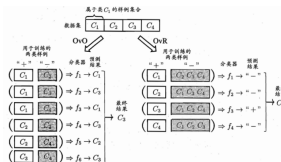
The simplest representation of a linear discriminant function can be expressed as:

$$y(x) = f(\omega^T x + \omega_0)$$

在机器学习文献中， $f(\cdot)$ 被称为激活函数。它的反函数是链接函数(link function). 决策面对应于 $y(x) = \text{const}$ . The normal distance from the origin to the decision surface is given by

$$\frac{\omega^T x}{\|\omega\|} = -\frac{\omega_0}{\|\omega\|}$$

- ▶ 现实中常遇到多分类学习任务，有些二分类学习方法可直接推广到多分类，但在更多情况下，我们是基于一些基本策略，根据二分类学习器来解决多分类问题。假设有  $N$  类别  $C_1, C_2, \dots, C_N$ ，多分类学习的基本思路是“拆解法”，将多分类任务拆分为若干个二分类任务求解。具体来说，先对问题进行拆分，然后为拆出的每个二分类任务训练一个分类器。在测试的时候，对这些分类器的预测结果进行集成以获得最终的多分类结果。因此，如何对多分类任务进行拆分是关键。
  - ▶ 拆分策略：一对一（One vs One，简称OvO）；OvO将  $N$  个类别两两配对，从而产生  $N(N-1)/2$  个二分类任务，例如OvO将为区分类别  $C_i, C_j$  训练一个分类器，该分类器把  $D$  中的  $C_i$  类样例作为正例， $C_j$  类样例作为反例。测试阶段，新样本将同时提交给所有分类器，于是我们将得到  $N(N-1)/2$  个分类结果，最终结果可以通过投票产生
  - ▶ 一对其余（One vs Rest，简称OvR）；OvR则是每次将一个类的样例作为正例，所有其他类的样例作为反例来训练  $N$  个分类器，在测试时若仅有一个分类器预测为正类，则对应的类别标记作为最终分类结果。



# MvM 策略

- ▶ **MvM** 每次将若干个类作为正类，若干个其它类作为反类。**OvO** 和 **OvR** 可以看成是它的特例。**MvM** 的正、反类构造必须有特殊的设计，不能随意选取。
- ▶ 最常用的**MvM** 技术：纠错输出码（**ECOC**）。**ECOC**是将编码的思想引入类别拆分，并尽可能在解码的过程中具有容错性。**ECOC** 工作过程主要分为两步：
  - （1）编码：对  $N$  个类别做  $M$  次划分，每次划分将一部分类别划为正类，一部分划为反类，从而形成一个二分类训练集，这样一共产生  $M$  个训练集，可训练出  $M$  个训练器  $f_m, m = 1, \dots, M$ 。
  - （2）解码： $M$  个分类器分别对测试样本进行预测，这些预测标记组成一个编码。将这个与此编码与每个类别各自的编码进行比较，返回其中距离最小的类别为最终预测结果。



# ECOC 编码

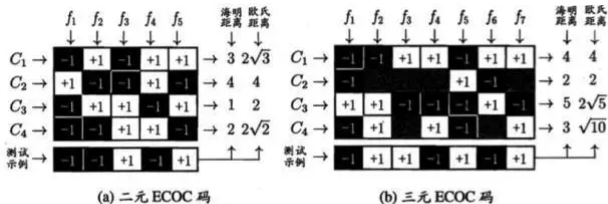


图 3.5 ECOC 编码示意图。“+1”、“-1”分别表示学习器  $f_i$  将该类样本作为正、反例；三元码中“0”表示  $f_i$  不使用该类样本

- ▶ 二元:  $N = 4, M = 5$ ; 三元  $N = 4, M = 7$ .
- ▶ 海明距离: 统计所有分类器  $f_m$  对测试样本的分类结果和  $C_i$  类不一致的数目叫做第  $i$  类的海明距离, 不一致计数加1, 否则不加, 结果为:  $0+1+1+1+0 = 3$
- ▶ 欧式距离: 每个分类器对测试样本的分类结果减去  $C_i$  类的分类划分, 差值的平方和的开方, 结果为: 根号下  $\sqrt{((-1 - (+1))^2 + 2^2 + 2^2 + 2^2 + 0)} = \sqrt{12} = 2\sqrt{3}$ .

# 类别不平衡问题

- ▶ 类别不平衡性问题：分类任务中不同类别的训练样例数差别很大。
- ▶ 从线性分类器的角度讨论，使用  $y = w^T x + b$  对新样本进行分类时，用预测的  $y$  与一个阈值进行比较，当  $y > 0.5$  即判别为正例，否则判别为负例。这里的  $y$  实际表达了正例的可能性（ $1-y$  是反例的可能性）， $0.5$  表明分类器认为正反例可能性相同，即

$$\delta(x) = \begin{cases} +, & \frac{y}{1-y} > 1 \\ -, & otherwise \end{cases}$$

- ▶ 如果训练集中正反例数目相差悬殊，令  $m+$  表示正例数目， $m-$  表示反例数目，则观测几率就代表了真实几率，只要分类器的预测几率高于观测几率就判定为正例，即

$$\delta(x) = \begin{cases} +, & \frac{y}{1-y} > \frac{m+}{m-} \\ -, & otherwise \end{cases}$$

## ▶ 欠采样

- ▶ 对训练集里的反例样本进行“欠采样”，即去除一些反例使得正反例数目接近，再进行学习。由于丢弃了很多反例，会使得训练集远小于初始训练集，所以有可能导致欠拟合；
- ▶ 代表算法：**EasyEnsemble**；
- ▶ 利用集成学习机制，每次从大多数类中抽取和少数类数目差不多的重新组合，总共构成 $n$ 个新的训练集，基于每个训练集训练出一个**AdaBoost** 分类器(带阈值),最后结合之前训练分类器结果加权求和减去阈值确定最终分类类别。

## ▶ 过采样

- ▶ 增加一些正例使得正反例数目接近，然后再学习,需要注意的是不能只是对初始正例样本重复采样，否则将导致严重的过拟合。
- ▶ 代表算法：**SMOTE**
- ▶ 合成新的少数样本的策略是，对每个少类 $a$  样本，从最近邻中随机选一个样本 $b$ ，在 $a$ 、 $b$  之间连线上随机选一点作为合成新样本。
- ▶ 基于算法的改进：**SMOTE**可能导致初始样本分布有的部分更加稠密，有的部分更加稀疏，而且使得正反例的边界模糊。所以有学者提出**Borderline-SMOTE** 算法，将少数类样本根据距离多数类样本的距离分为noise,safe,danger 三类样本集，只对danger中的样本集合使用**SMOTE** 算法。

## 2020.4.6 作业

- ▶ 从正态分布  $C_1 = N(\begin{pmatrix} -1.5 \\ -1 \end{pmatrix}, \Sigma_1)$  和  $C_2 = N(\begin{pmatrix} 2 \\ 1 \end{pmatrix}, \Sigma_2)$ ,  
 $\Sigma_1 = \begin{pmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{pmatrix}$  和  $\Sigma_2 = \begin{pmatrix} 1 & \rho_2 \\ \rho_2 & 1 \end{pmatrix}$  进行如下比较分析:
  1. 令  $\rho_1 = \rho_2 = 0$ ,  $C_1$  类产生随机数100个,  $C_2$  类选择80个作为训练数据, 另从  $C_1$  中产生20个和  $C_2$  类产生20例作为测试数据, 比较  $LDA$  和  $QDA$  在测试数据上的分类错误率, 分析这两个模型哪个模型更适用?
  2. 令  $\rho_1 = -0.3, \rho_2 = 0.8$ , 重新进行实验, 实验方法和1类似, 比较  $LDA$  和  $QDA$  在测试数据上的分类错误率, 分析这两个模型哪个模型更适用?
  3. 从  $t(2)$  分布中产生  $X_1, X_2$  生成50个观测作为  $C_3$  类, 此时需要将  $C_3$  和  $C_2$  分开  $LDA$  和  $QDA$  哪一种比较理想? 以上实验中的  $P(C_i), i = 1, 2, 3$  请用每一组参与训练的数据量来估计。
- ▶ 用iris数据, 用Species(S:setosa;V:versicolor)对四个输入变量进行Fisher判别建模, 求法向量 $w$ . 尝试BayesLDA和QDA 建模, 进行比较, 实验中的  $P(C_S) : P(C_V) = 5 : 2$  。
- ▶ 用上次作业的sa心脏病数据, 用chd(1:得病, 0:正常)对tobacco(吸烟量)+ldl(肥胖指数)+age(年龄)进行logistic回归建模, 和用FisherLDA 进行比较, 实验中的  $P(C_i), i = 1, 2, 3$  请用每一组参与训练的数据量来估计。