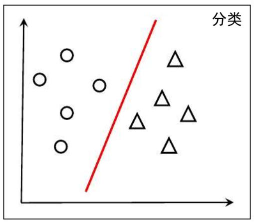
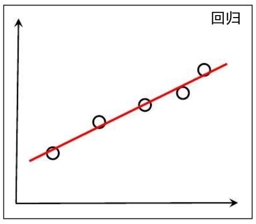
四、线性模型

目录

线性模型 (linear regression)





线性模型(linear model)试图学得一个通过属性的线性组合来进行预测的函数

$$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$

向量形式：$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$　　简单、基本、可理解性好

线性回归 (linear regression)

$$\boxed{f(x) = wx_i + b \quad \text{使得} \quad f(x_i) \simeq y_i}$$

离散属性的处理：若有“序”(order)，则连续化；
否则，转化为 k 维向量

令均方误差最小化，有 $(w^*, b^*) = \underset{(w,b)}{\arg\min} \sum_{i=1}^{m} (f(x_i) - y_i)^2$

$$= \underset{(w,b)}{\arg\min} \sum_{i=1}^{m} (y_i - wx_i - b)^2$$

对 $E_{(w,b)} = \sum_{i=1}^{m} (y_i - wx_i - b)^2$ 进行最小二乘参数估计

线性回归

分别对 w 和 b 求导：

$$\frac{\partial E_{(w,b)}}{\partial w} = 2\left(w\sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m}(y_i - b)\,x_i\right)$$

$$\frac{\partial E_{(w,b)}}{\partial b} = 2\left(mb - \sum_{i=1}^{m}(y_i - wx_i)\right)$$

令导数为 0, 得到闭式(closed-form)解：

$$w = \frac{\sum_{i=1}^{m} y_i(x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}\left(\sum_{i=1}^{m} x_i\right)^2} \qquad b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)$$

多元(multi-variate)线性回归

$$\boxed{f(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \quad \text{使得} \quad f(\boldsymbol{x}_i) \simeq y_i}$$

$$\boldsymbol{x}_i = (x_{i1}; x_{i2}; \ldots; x_{id}) \qquad y_i \in \mathbb{R}$$

把 $\boldsymbol{w}$ 和 b 吸收入向量形式 $\hat{\boldsymbol{w}} = (\boldsymbol{w}; b)$，数据集表示为

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} & 1 \\ \boldsymbol{x}_2^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_m^{\mathrm{T}} & 1 \end{pmatrix} \qquad \boldsymbol{y} = (y_1; y_2; \ldots; y_m)$$

线性模型的变化

对于样例 $(\boldsymbol{x}, y)$, $y \in \mathbb{R}$, 若希望线性模型的预测值逼近真实标记，则得到线性回归模型 $y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

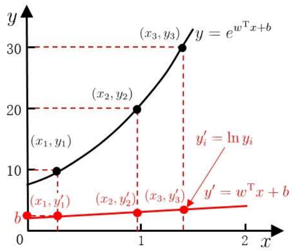
令预测值逼近 y 的衍生物？

若令 $\ln y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

则得到对数线性回归

(log-linear regression)

实际是在用 $e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}$ 逼近 y



广义(generalized)线性模型-逻辑回归

一般形式: $y = g^{-1}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right)$

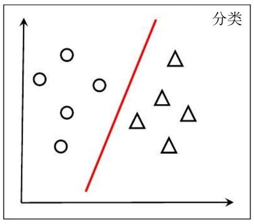
单调可微的 联系函数 (link function)

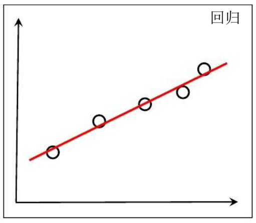
令 $g(\cdot) = \ln(\cdot)$ 则得到 对数线性回归

$$\ln y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

$\dots\ \dots$

线性模型做“分类”





如何“直接”做分类？

广义线性模型：通过“联系函数”

例如，对率回归

二分类任务

线性回归模型产生的实值输出 $z = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

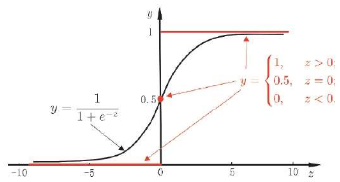
期望输出 $y \in \{0, 1\}$

找 z 和 y 的联系函数

理想的“单位阶跃函数”

(unit-step function)

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



性质不好，需找“替代函数”

(surrogate function)

单调可微、任意阶可导

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

对数几率函数

(logistic function)

简称“对率函数”

对率回归-逻辑回归

逻辑回归

以对率函数为联系函数：

$$y = \frac{1}{1 + e^{-z}} \quad \text{变为} \quad y = \frac{1}{1 + e^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)}}$$

即: $\ln \frac{y}{1-y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

“对数几率”

(log odds, 亦称 logit)

几率(odds), 反映了 $\boldsymbol{x}$ 作为正例的相对可能性

“对数几率回归”(logistic regression)

简称“对率回归”

注意：它是分类学习算法！

- 无需事先假设数据分布
- 可得到“类别”的近似概率预测
- 可直接应用现有数值优化算法求取最优解

求解思路

逻辑回归

若将 y 看作类后验概率估计 $p(y = 1 \mid \boldsymbol{x})$，则

$$\ln \frac{y}{1-y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b \quad \text{可写为} \quad \ln \frac{p(y = 1 \mid \boldsymbol{x})}{p(y = 0 \mid \boldsymbol{x})} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

$\theta^T x$

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

对于输入x分类结果为类别1和类别0的概率分别为：

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

“极大似然法”

求解思路

逻辑回归

给定数据集 $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$

最大化“对数似然”(log-likelihood)函数:

$$l(w,b)=\sum_{i=v}^{m} lnp(yi|xi;\theta)$$

X被正确分类的概率：

$$P(y\mid x;\theta)=(h_\theta(x))^y(1-h_\theta(x))^{1-y}$$

求解思路

逻辑回归

$$P(y\mid x;\theta)=(h_\theta(x))^y(1-h_\theta(x))^{1-y}$$

取似然函数为:

$$L(\theta)=\prod_{i=1}^{m}P(y^{(i)}\mid x^{(i)};\theta)=\prod_{i=1}^{m}(h_\theta(x^{(i)}))^{y^{(i)}}(1-h_\theta(x^{(i)}))^{1-y^{(i)}}$$

对数似然函数为:

$$l(\theta)=\log L(\theta)=\sum_{i=1}^{m}\left(y^{(i)}\log h_\theta(x^{(i)})+(1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right)$$

对数似然函数为:

$$l(\theta)=\log L(\theta)=\sum_{i=1}^{m}\left(y^{(i)}\log h_\theta(x^{(i)})+(1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right)$$

逻辑回归的损失函数为:

$$J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}logh_\theta(x^{(i)})+(1-y^{(i)})log(1-h_\theta(x^{(i)}))]$$

$$Cost(h_\theta(x),y)=\begin{cases}-\log(h_\theta(x)) & if\ y=1\\ -\log(1-h_\theta(x)) & if\ y=0\end{cases}$$

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{n}Cost(h_\theta(x^{(i)}),y^{(i)})=-\frac{1}{m}\sum_{i=1}^{n}\left(y^{(i)}\log h_\theta(x^{(i)})+(1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right)$$

$$J(\theta)=-\frac{1}{m}l(\theta)$$

逻辑回归

$$h_\theta(x)=g(\theta^Tx)=\frac{1}{1+e^{-\theta^Tx}}$$

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{n}Cost(h_\theta(x^{(i)}),y^{(i)})=-\frac{1}{m}\sum_{i=1}^{n}\left(y^{(i)}\log h_\theta(x^{(i)})+(1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right)$$

梯度下降法求的最小值： $\theta_j=\theta_j-\alpha\frac{\delta}{\delta\theta_j}J(\theta)$

$$\frac{\delta}{\delta\theta_j}J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{h_\theta(x^{(i)})}\frac{\delta}{\delta_{\theta_j}}h_\theta(x^{(i)})-(1-y^{(i)})\frac{1}{1-h_\theta(x^{(i)})}\frac{\delta}{\delta_{\theta_j}}h_\theta(x^{(i)})\right)$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^Tx^{(i)})}-(1-y^{(i)})\frac{1}{1-g(\theta^Tx^{(i)})}\right)\frac{\delta}{\delta_{\theta_j}}g(\theta^Tx^{(i)})$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^Tx^{(i)})}-(1-y^{(i)})\frac{1}{1-g(\theta^Tx^{(i)})}\right)g(\theta^Tx^{(i)})(1-g(\theta^Tx^{(i)}))\frac{\delta}{\delta_{\theta_j}}\theta^Tx^{(i)}$$

逻辑回归

$$\frac{\delta}{\delta\theta_j}J(\theta)$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^Tx^{(i)})}-(1-y^{(i)})\frac{1}{1-g(\theta^Tx^{(i)})}\right)g(\theta^Tx^{(i)})(1-g(\theta^Tx^{(i)}))\frac{\delta}{\delta_{\theta_j}}\theta^Tx^{(i)}$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}(1-g(\theta^Tx^{(i)}))-(1-y^{(i)})g(\theta^Tx^{(i)})\right)x_j^{(i)}$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}-g(\theta^Tx^{(i)})\right)x_j^{(i)}$$

$$=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j^{(i)}$$

求解思路

逻辑回归

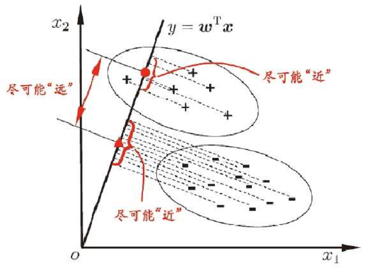
$$\theta_j:=\theta_j-\alpha\frac{\delta}{\delta_{\theta_j}}J(\theta)$$

$$\frac{\delta}{\delta_{\theta_j}}J(\theta):=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j^{(i)}$$

$$\theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j^{(i)}$$

目录

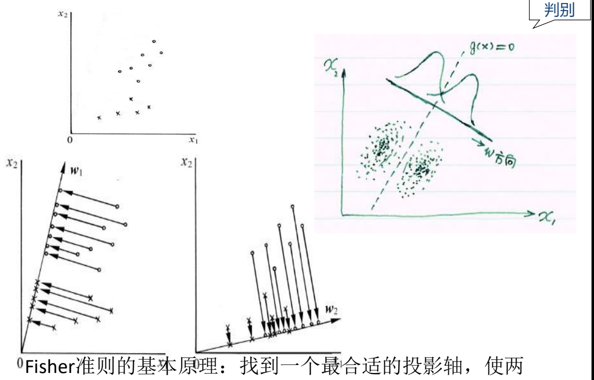
线性判别分析 (Linear Discriminant Analysis)



由于将样例投影到一条直线（低维空间），因此也被视为一种“监督降维”技术 降维 ⟶ 第10章

线性判别分析

Fisher 判别



Fisher准则的基本原理：找到一个最合适的投影轴，使两类样本在该轴上投影之间的距离尽可能远，而每一类样本的投影尽可能紧凑，从而使分类效果为最佳。

21

*d*维空间样本分布的描述量

Fisher 判别

- 各类样本均值向量$\mathbf{m}_i$　$\mathbf{m}_i = \frac{1}{N_i}\sum_{x\in K_i}\mathbf{x} \quad i=1,2$
- 样本**类内离散度**矩阵$\mathbf{S}_i$与总类内离散度矩阵$\mathbf{S}_w$

$$\mathbf{S}_i = \sum_{\mathbf{x}\in\chi_i}(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^T, \quad i=1,2 \qquad \mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2$$

- 样本**类间离散度**矩阵$\mathbf{S}_b$：$\mathbf{S}_b = (\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^T$

离散度矩阵在形式上与协方差矩阵很相似，但协方差矩阵是一种期望值，而离散矩阵只是表示有限个样本在空间分布的离散程度

22

一维*Y*空间样本分布的描述量

Fisher 判别

- 各类**样本均值**　$\tilde{m}_i = \frac{1}{N_i}\sum_{y\in\psi_i} y, \quad i=1,2$
- 样本**类内离散度**和总类内离散度

$$\tilde{S}_i = \sum_{y\in\psi_i}(y-\tilde{m}_i)^2, \quad i=1,2 \qquad \tilde{S}_w = \tilde{S}_1 + \tilde{S}_2$$

- 样本**类间离散度**　$\tilde{S}_b = (\tilde{m}_1 - \tilde{m}_2)^2$

以上定义描述*d*维空间样本点到一向量**投影的分散情况**，因此也就是对某向量**w**的投影在**w**上的分布。样本离散度的定义与随机变量方差相类似

23

样本与其投影统计量间的关系

Fisher 判别

- 样本**x**与其投影*y*的统计量之间的关系：

$$\tilde{m}_i = \frac{1}{N_i}\sum_{y\in\psi_i} y = \frac{1}{N_i}\sum_{y\in K_i}\mathbf{w}^T\mathbf{x} = \mathbf{w}^T\mathbf{m}_i, \quad i=1,2$$

$$\tilde{S}_b = (\tilde{m}_1-\tilde{m}_2)^2 = (\mathbf{w}^T\mathbf{m}_1 - \mathbf{w}^T\mathbf{m}_2)^2$$
$$= \mathbf{w}^T(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{w} = \mathbf{w}^T S_b \mathbf{w}$$

24

样本与其投影统计量间的关系

Fisher 判别

$$\begin{aligned}\tilde{S}_i &= \sum_{y\in\psi_i}(y-\tilde{m}_i)^2 \\ &= \sum_{x\in K_i}(\mathbf{w}^T\mathbf{x}-\mathbf{w}^T\mathbf{m}_i)^2 \\ &= \mathbf{w}^T\left[\sum_{x\in K_i}(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^T\right]\mathbf{w} \\ &= \mathbf{w}^T S_i \mathbf{w}\end{aligned}$$

$$\tilde{S}_1+\tilde{S}_1 = \mathbf{w}^T(S_1+S_2)\mathbf{w} = \mathbf{w}^T S_w \mathbf{w}$$

25

Fisher准则函数

Fisher 判别

- 评价投影方向w的原则，使原样本向量在该方向上的投影能兼顾类间分布尽可能分开，类内尽可能密集的要求
- Fisher准则函数的定义：

$$J_F(w) = \frac{\tilde{S}_b}{\tilde{S}_1+\tilde{S}_2} = \frac{\mathbf{w}^T S_b \mathbf{w}}{\mathbf{w}^T S_w \mathbf{w}}$$

- Fisher最佳投影方向的求解

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}}\, J_F(\mathbf{w})$$

26

Fisher最佳投影方向的求解

Fisher 判别

- 采用拉格朗日乘子算法解决

$$\mathbf{w}^* = S_w^{-1}(\mathbf{m}_1-\mathbf{m}_2)$$

$\mathbf{m}_1$-$\mathbf{m}_2$是一向量，对与($\mathbf{m}_1$-$\mathbf{m}_2$)平行的向量投影可使两均值点的距离最远。但是如从使类间分得较开，同时又使类内密集程度较高这样一个综合指标来看，则需根据两类样本的分布离散程度对投影方向作相应的调整，这就体现在对$\mathbf{m}_1$-$\mathbf{m}_2$ 向量按S_w^{-1}作一线性变换，从而使Fisher准则函数达到极值点

27

Fisher公式的推导

Fisher 判别

$$J_F(w) = \frac{\tilde{S}_b}{\tilde{S}_1+\tilde{S}_2} = \frac{\mathbf{w}^T S_b \mathbf{w}}{\mathbf{w}^T S_w \mathbf{w}}$$

令 $\mathbf{w}^T S_w \mathbf{w} = c \neq 0$

定义Lagrange函数: $L(\mathbf{w},\lambda) = \mathbf{w}^T S_b \mathbf{w} - \lambda(\mathbf{w}^T S_w \mathbf{w} - c)$

令: $\dfrac{\partial L(\mathbf{w},\lambda)}{\partial \mathbf{w}} = S_b\mathbf{w} - \lambda S_w \mathbf{w} = 0$

$$S_w^{-1}S_b\mathbf{w} = \lambda\mathbf{w}$$

$$\begin{aligned}\lambda\mathbf{w} &= S_w^{-1}S_b\mathbf{w} = S_w^{-1}(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{w} \\ &= S_w^{-1}(\mathbf{m}_1-\mathbf{m}_2)R\end{aligned}$$

$$\mathbf{w}^* = \frac{R}{\lambda}S_w^{-1}(\mathbf{m}_1-\mathbf{m}_2) = S_w^{-1}(\mathbf{m}_1-\mathbf{m}_2)$$

28

Fisher 判别

决策规则： 若 $g(x) = w^T x + w_0 \begin{matrix} >0 \\ <0 \end{matrix}$，则 $\begin{matrix} x\in\omega_1 \\ x\in\omega_2 \end{matrix}$

选阈值 w_0：

（1）d 和 N 很大时，y 近似正态分布，可在 Y 空间内用贝叶斯分类器。

（2）经验，如：

$$w_0 = -\frac{1}{2}(\tilde{m}_1+\tilde{m}_2)$$

$$w_0 = -\tilde{m}$$

$$w_0 = -\frac{1}{2}(\tilde{m}_1+\tilde{m}_2) - \frac{1}{N_1+N_2-2}\ln\frac{P(\omega_1)}{P(\omega_2)}$$

2．已知有两类数据,分别为

Fisher 判别

2．已知有两类数据,分别为

ω_1：(1，0)，(2，0)，(1，1)
ω_2：(−1，0)，(0，1)，(−1，1)

试求：该组数据的类内及类间离散矩阵 S_w 及 S_b。

2．解：第一类的均值向量为

$$m_1 = \left(\frac{4}{3},\frac{1}{3}\right),\quad m_2 = \left(\frac{-2}{3},\frac{2}{3}\right)$$

$$\therefore S_1 = \frac{1}{9}\begin{pmatrix}6 & -5\\ -5 & 6\end{pmatrix},\quad S_2 = \frac{1}{3}\begin{pmatrix}2 & 1\\ 1 & 2\end{pmatrix}$$

$$S_w = S_1 + S_2 = \frac{1}{9}\begin{pmatrix}12 & -2\\ -2 & 12\end{pmatrix}$$

$$S_b = \begin{pmatrix}\frac{6}{3}\\ -\frac{1}{3}\end{pmatrix}\begin{pmatrix}\frac{6}{3} & -\frac{1}{3}\end{pmatrix} = \frac{1}{9}\begin{pmatrix}36 & -6\\ -6 & 1\end{pmatrix}$$

Sw^{-1}=[0.7714 0.1286
0.1286 0.7714]

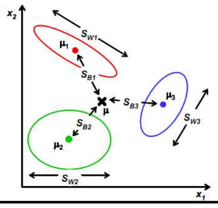
W*= Sw^{-1} (m1-m2)
=[2.7407 -0.8889]^T

推广到多类

$\mathbf{W} \in \mathbb{R}^{d\times(N-1)}$　　W=[w_1, w_2, ……w_{N-1}]

假定有 N 个类

- 全局散度矩阵 $\mathbf{S}_t = \mathbf{S}_b + \mathbf{S}_w = \sum_{i=1}^{m} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$
- 类内散度矩阵 $\mathbf{S}_w = \sum_{i=1}^{N} \mathbf{S}_{w_i}$　$\mathbf{S}_{w_i} = \sum_{\boldsymbol{x}\in X_i} (\boldsymbol{x} - \boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i)^T$
- 类间散度矩阵 $\mathbf{S}_b = \mathbf{S}_t - \mathbf{S}_w = \sum_{i=1}^{N} m_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$



推广到多类

W=[w_1, w_2, ……w_{N-1}]

假定有 N 个类

- 全局散度矩阵 $\mathbf{S}_t = \mathbf{S}_b + \mathbf{S}_w = \sum_{i=1}^{m} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$
- 类内散度矩阵 $\mathbf{S}_w = \sum_{i=1}^{N} \mathbf{S}_{w_i}$　$\mathbf{S}_{w_i} = \sum_{\boldsymbol{x}\in X_i} (\boldsymbol{x} - \boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i)^T$
- 类间散度矩阵 $\mathbf{S}_b = \mathbf{S}_t - \mathbf{S}_w = \sum_{i=1}^{N} m_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$

$\mathbf{W} \in \mathbb{R}^{d\times(N-1)}$　　W=[w_1, w_2, ……w_{N-1}]

我们将样本点在这N-1维向量投影后结果表示为：y=[y_1, y_2, ……y_{N-1}]

$y_i = w_i^T x$

$Y = W^T x$

推广到多类

假定有 N 个类

- 全局散度矩阵 $\mathbf{S}_t = \mathbf{S}_b + \mathbf{S}_w = \sum_{i=1}^{m} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$
- 类内散度矩阵 $\mathbf{S}_w = \sum_{i=1}^{N} \mathbf{S}_{w_i}$　$\mathbf{S}_{w_i} = \sum_{\boldsymbol{x}\in X_i} (\boldsymbol{x} - \boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i)^T$
- 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 $\mathbf{S}_b$，$\mathbf{S}_w$，$\mathbf{S}_t$

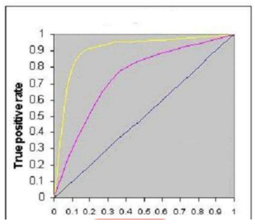
例如，$\max_{\mathbf{W}} \frac{\mathrm{tr}(\mathbf{W}^T\mathbf{S}_b\mathbf{W})}{\mathrm{tr}(\mathbf{W}^T\mathbf{S}_w\mathbf{W})} \Rightarrow \mathbf{S}_b\mathbf{W} = \lambda \mathbf{S}_w \mathbf{W}$

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

逻辑回归主要用来做回归吗？

逻辑回归中可以用以下哪种方法来训练数据？
A.最小二乘法 B最大似然估计 C.杰卡德距离

10.下图是3个逻辑回归模型的AUC-ROC曲线。不同的颜色表示不同的超参值，哪一个会产生最佳的结果？



目录

多类问题

多分类

- 两类别问题可以推广到多类别问题
- 多个分类器
  - $\omega_i/\sim\omega_i$ 法：将C类别问题化为(C-1)个两类（第i类与所有非i类）问题，按两类问题确定其判别函数与决策面方程 (one-vs-rest)
  - ω_i/ω_j 法：将C类中的每两类别单独设计其线性判别函数，因此总共有C(C-1)/2个线性判别函数方程 (one-vs-one)

36

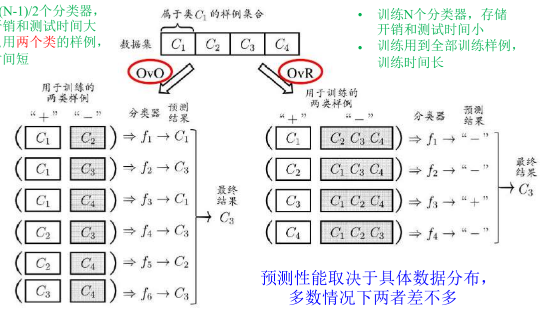
多分类学习

多分类

拆解法：将一个多分类任务拆分为若干个二分类任务求解

- 训练N(N-1)/2个分类器，存储开销和测试时间大
- 训练只用两个类的样例，训练时间短

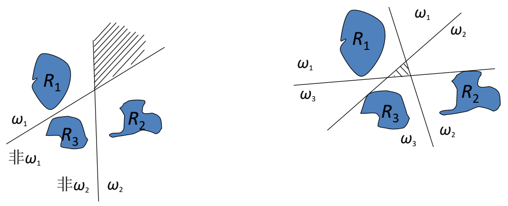
- 训练N个分类器，存储开销和测试时间小
- 训练用到全部训练样例，训练时间长



预测性能取决于具体数据分布，多数情况下两者差不多

多分类问题

多分类



38

多分类问题:sklearner

多分类

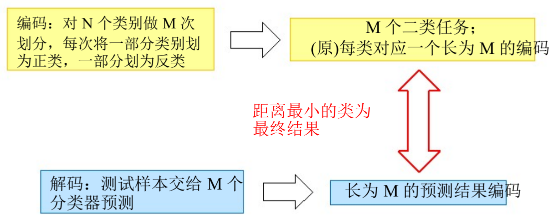
Multiclass as `One-Vs-One`:
sklearn.svm.NuSVC sklearn.svm.SVC
sklearn.gaussian_process.GaussianProcessClassifier(setting multi_class = “one_vs_one”)
Multiclass as One-Vs-All:
sklearn.ensemble.GradientBoostingClassifiersklearn.gaussian_process.GaussianProcessClassifier(setting multi_class = “one_vs_rest”)
sklearn.svm.LinearSVC(setting multi_class=” ovr”)
sklearn.linear_model.LogisticRegression(setting multi_class=” ovr”)
sklearn.linear_model.SGDClassifier sklearn.linear_model.Perceptron

纠错输出码 (ECOC)

多分类

多对多(Many vs Many, MvM): 将若干类作为正类，若干类作为反类

一种常见方法：纠错输出码 (Error Correcting Output Code)



纠错输出码

多分类

	f_1	f_2	f_3	f_4	f_5	海明距离	欧氏距离
C_1	−1	+1	−1	+1	+1	3	$2\sqrt{3}$
C_2	+1	−1	−1	+1	−1	4	4
C_3	−1	+1	+1	−1	+1	1	2
C_4	−1	−1	+1	+1	−1	2	$2\sqrt{2}$
测试示例	−1	−1	+1	−1	+1		

(a) 二元 ECOC 码

[Dietterich and Bakiri,1995]

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	海明距离	欧氏距离
C_1	−1	−1	+1	+1	−1	+1	+1	4	4
C_2	−1	0	0	0	+1	−1	0	2	2
C_3	+1	+1	−1	−1	−1	+1	−1	5	$2\sqrt{5}$
C_4	−1	+1	0	+1	−1	0	+1	3	$\sqrt{10}$
测试示例	−1	+1	+1	−1	+1	−1	+1		

(b) 三元 ECOC 码

[Allwein et al. 2000]

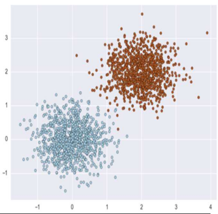
- ECOC编码对分类器错误有一定容忍和修正能力，编码越长、纠错能力越强
- 对同等长度的编码，理论上来说，任意两个类别之间的编码距离越远，则纠错能力越强

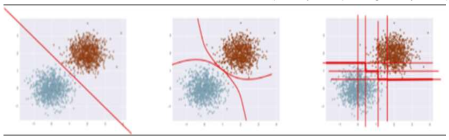
目录

非平衡分类问题

非平衡分类

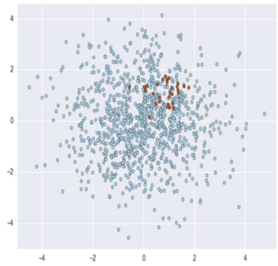
- 教课书中常见的分类问题



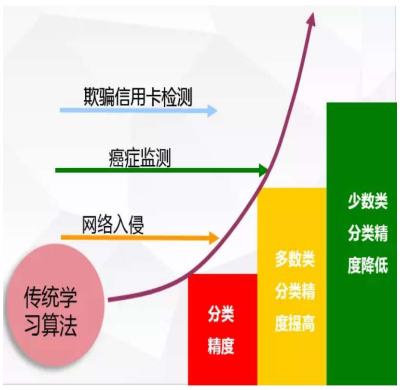


非平衡问题

非平衡分类



- 研究不平衡类通常认为不平衡意味着少数类只占比10~20%。实际上，一些数据集远比这更不平衡。例如：
  - 每年大约有2%的信用卡账户被欺骗。（大多数欺诈检测领域严重不平衡。）
  - 状态医疗甄别通常在大量不存在此状态的人口中检测极少数有此状态的人（比如美国的HIV携带者仅占0.4%）。
  - 磁盘驱动器故障每年约1%。
  - 网络广告的转化率估计在10^{-3}到10^{-6}之间。



评价标准

非平衡分类

- 混淆矩阵

状态与决策的可能关系

决策 \ 状态	阳性	阴性
阳性	真阳性（TP）	假阳性（FP）
阴性	假阴性（FN）	真阴性（TN）

- 精度

$$Precision = \frac{TP}{TP+FP}$$

- 召回率

$$Recall = \frac{TP}{TP+FN}$$

- Kappa（衡量分类精度）和Roc曲线（曲线面积就是AUC）

类别不平衡 (class-imbalance)

非平衡分类

不同类别的样本比例相差很大；“小类”往往更重要

基本思路：

若 $\frac{y}{1-y} > 1$ 则 预测为正例. ⇨ 若 $\frac{y}{1-y} > \frac{m^+}{m^-}$ 则 预测为正例.

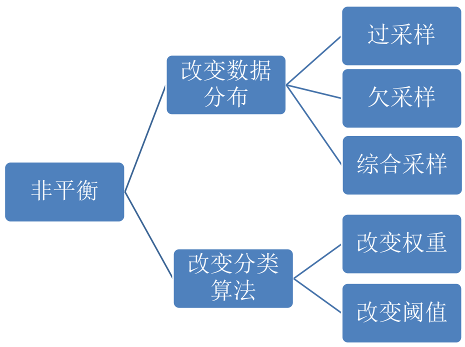
基本策略
—— “再缩放” (rescaling)：

$$\frac{y'}{1-y'} = \frac{y}{1-y} \times \frac{m^-}{m^+}$$

然而，精确估计 m_-/m_+ 通常很困难！

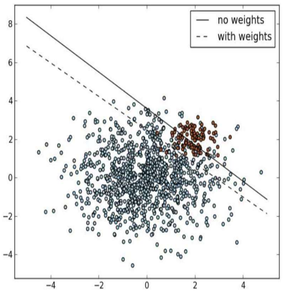
非平衡处理的方法

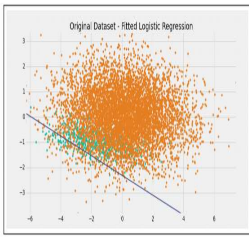
非平衡分类



改变分类算法

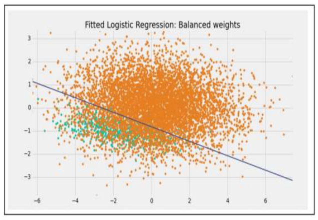
- 一个处理非平衡数据常用的方法就是设置损失函数的权重，使得少数类判别错误的损失大于多数类判别错误的损失。在python的scikit-learn中我们可以使用class_weight参数来设置权重。
- 指定样本各类别的的权重，主要是为了防止训练集某些类别的样本过多，导致训练的决策树过于偏向这些类别。这里可以自己指定各个样本的权重，或者用“balanced”，如果使用“balanced”，则算法会自己计算权重，样本量少的类别所对应的样本权重会高。当然，如果你的样本类别分布没有明显的偏倚，则可以不管这个参数，选择默认的"None"





The performance on the test set is as follows.

precision on L	recall on S
0.90	0.12

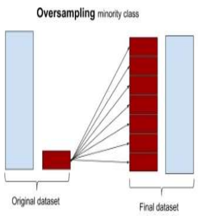


precision on L	recall on S
0.98	0.89

过采样

非平衡分类

- 随机过采样
  - 采取简单复制样本的策略来增加少数类样本
  - 容易产生模型过拟合的问题
- SMOTE全称是Synthetic Minority Oversampling Technique即合成少数类过采样技术
  - 2002年Chawla提出了SMOTE算法
  - 对少数类样本进行分析并根据少数类样本人工合成新样本添加到数据集中



SMOTE算法流程

非平衡分类

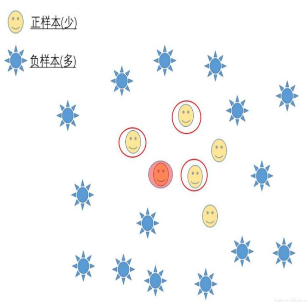
- SMOTE算法流程：
  - 1、对于少数类中每一个样本a，以欧氏距离为标准计算它到少数类样本集中所有样本的距离，得到其k近邻。
  - 2、根据样本不平衡比例设置一个采样比例以确定采样倍率N，对于每一个少数类样本a，从其k近邻中随机选择若干个样本，假设选择的近邻为b。
  - 3、对于每一个随机选出的近邻b，分别与原样本a按照如下的公式构建新的样本：

c=a+rand(0,1)∗|a−b|

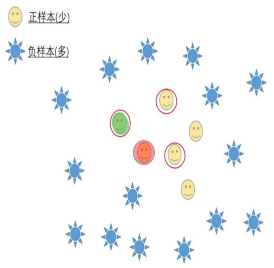
SMOTE 算法是建立在相距较近的少数类样本之间的样本仍然是少数类的假设基础上的。

- **步骤__1.选一个正样本**

- **步骤2：找到该正样本的K个近邻（假设K = 3）**



- 步骤__3.随机从K个近邻中选出一个样本

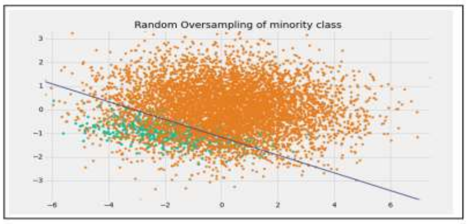


- 步骤__4.在正样本和随机选出的这个近邻之间的连线上，随机找一点。这个点就是人工合成


```
Algorithm SMOTE(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE N%; Number of nearest
    neighbors k
Output: (N/100) * T synthetic minority class samples
(* If N is less than 100%, randomize the minority class samples as only a random
    percent of them will be SMOTEd. *)
if N < 100
    then Randomize the T minority class samples
        T = (N/100) * T
        N = 100
endif
N = (int)(N/100) (* The amount of SMOTE is assumed to be in integral multiples of
    100. *)
k = Number of nearest neighbors
numattrs = Number of attributes
Sample[ ][ ]: array for original minority class samples
newindex: keeps a count of number of synthetic samples generated, initialized to 0
Synthetic[ ][ ]: array for synthetic samples
(* Compute k nearest neighbors for each minority class sample only. *)
for i ← 1 to T
        Compute k nearest neighbors for i, and save the indices in the nnarray
        Populate(N, i, nnarray)
endfor

Populate(N, i, nnarray) (* Function to generate the synthetic samples. *)
while N ≠ 0
        Choose a random number between 1 and k, call it nn. This step chooses one of
        the k nearest neighbors of i.
        for attr ← 1 to numattrs
                Compute: dif = Sample[nnarray[nn]][attr] − Sample[i][attr]
                Compute: gap = random number between 0 and 1
                Synthetic[newindex][attr] = Sample[i][attr] + gap * dif
        endfor
        newindex++
        N = N − 1
endwhile
return (* End of Populate. *)
End of Pseudo-Code.
```


	$\lvert L\rvert$	$\lvert S\rvert$
Before resampling	6320	680
After resampling	6320	3160



precision on L	recall on S
0.97	0.76

Smote算法总结

非平衡分类

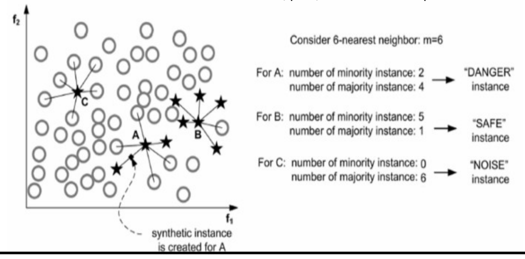
- 总结：
  - 对于少数类的每个样本寻找其同类样本中k个最近邻。其中，k通常是大于1的奇数。
  - 重复上述插值过程，使得新生成的训练数据集数据达到均衡，最后利用新的训练样本集进行训练
- 优点：
  - 有助于简单打破过抽样所产生的关系
  - 使得分类器的学习能力得到显著提高
- 缺陷：
  - 在近邻选择时,存在一定的盲目性。
  - 边界模糊化

Borderline-SMOTE

非平衡分类

- Han（2005）等人在SMOTE算法基础上进行了改进,提出了Borderline-SMOTE算法,解决了生成样本重叠(Overlapping)的问题。算法在运行的过程中,查找一个适当的区域,该区域可以较好地反应数据集的性质,然后在该区域内进行插值,以使新增加的“人造”样本更有效。

https://sci2s.ugr.es/keel/keel-dataset/pdfs/2005-Han-LNCS.pdf



非平衡分类

- Borderline-SMOTE

For each point p in S:

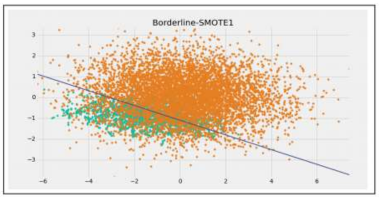
1. 计算点p在训练集T上的m个最近邻。我们称这个集合为Mp，然后设 m'= |Mp ∩ L| (表示点p的最近邻中属于L的数量).
2. If m'= m, p 是一个噪声,不做任何操作.
3. If 0 ≤m'≤m/2, 则说明p很安全，不做任何操作.
4. If m/2 ≤ m'≤ m, 那么点p就很危险了，我们需要在这个点附近生成一些新的少数类点，所以我们把它加入到DANGER中.

最后，对于每个在DANGER中的点d，使用SMOTE算法生成新的样本.

非平衡分类

- 应用Borderline-SMOTE的参数设置为k=5

	$\lvert L \rvert$	$\lvert S \rvert$
Before resampling	6320	680
After resampling	6320	3160

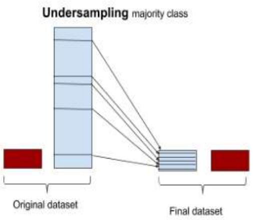


precision on L	recall on S
0.97	0.78

欠采样

非平衡分类

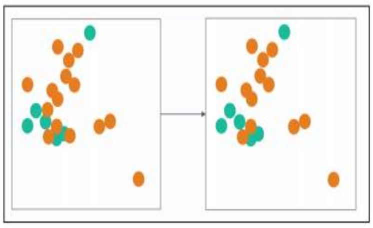
- 随机欠采样
  - 从多数类中随机抽取样本从而减少多数类样本的数量，使数据达到平衡。



欠采样

非平衡分类

- TomLink
  - 如果有两个不同类别的样本，它们的最近邻都是对方，也就是A的最近邻是B，B的最近邻是A，那么A,B就是Tomek link。我们要做的就是讲所有Tomek link都删除掉。那么一个删除Tomek link的方法就是，将组成Tomek link的两个样本，如果有一个属于多数类样本，就将该多数类

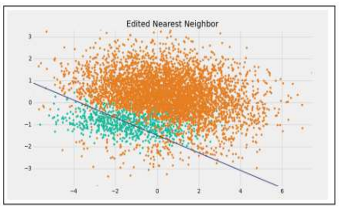


欠采样

非平衡分类

- TomLink
  - 针对minority的样本考察紧邻

	$\lvert L \rvert$	$\lvert S \rvert$
Before resampling	6320	680
After resampling	5120	680

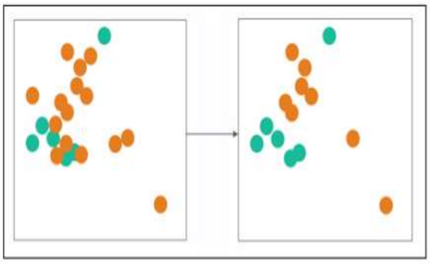


precision on L	recall on S
0.95	0.59

欠采样

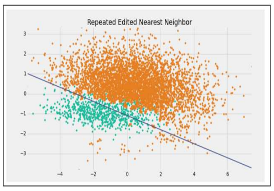
非平衡分类

- **ENN方法**
  - 对于那些多数类的样本，如果他的大部分k近邻样本都跟他自己本身的类别不一样，我们就将他删除。



非平衡分类

- **Repeated Edited Nearest Neighbor**
- 这个方法就是不断的重复上述的删除过程，直到无法再删除为止。

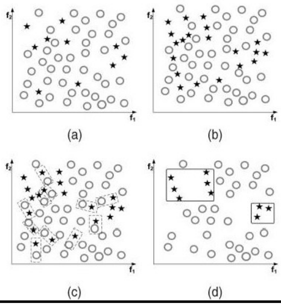


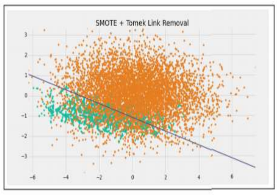
precision on L	recall on S
0.97	0.80

综合

非平衡分类

- SMOTE + Tomek
  – 先合成SMOTE点， 然后根据Tomek思想去除Tomek links。

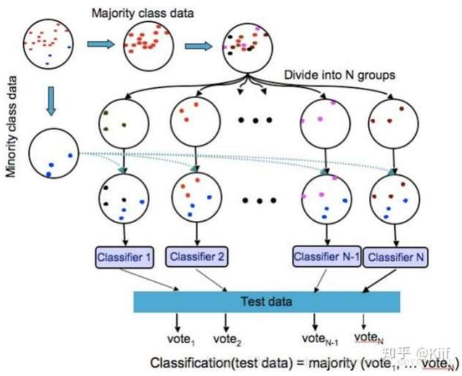




precision on L	recall on S
0.97	0.80

easyEnsemble

非平衡分类



前往第五站……