

第3章 Logistic回归

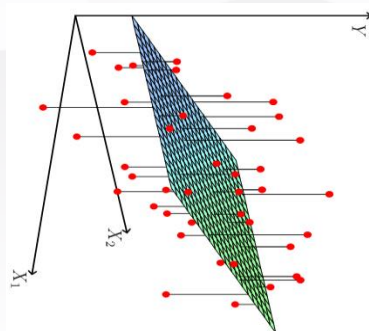
Outline

- 分类任务
- Logistic回归
- Logistic回归的损失函数
- Logistic回归的优化算法
- Logistic回归的多类分类任务
- 分类任务中的样本不均衡问题
- 分类模型的评价指标
- Sklearn中的Logistic回归API
- Logistic回归案例分析

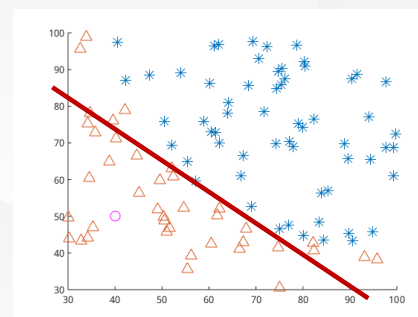
➤ 监督学习 (Supervised Learning)

- 训练数据：给定训练数据 $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ ，其中 N 为训练样本数目， i 为样本索引， \mathbf{x}_i 为第 i 个样本的输入特征， y_i 为对应的输出/响应
- 根据训练样本 \mathcal{D} ，学习一个从输入 \mathbf{x} 到输出 y 的映射 f
- 测试：对新的测试数据 \mathbf{x} ，用学习到的映射 f 对其进行预测： $\hat{y} = f(\mathbf{x})$

■ 回归： $y_i \in \mathbb{R}$



■ 分类： $y_i \in \mathcal{C}$, $\mathcal{C} = \{1, \dots, C\}$



- 考虑两类分类任务，样本的输出 $y_i \in \{0,1\}$ 。
- 在概率分布中，贝努利 (Bernoulli) 试验的输出为 $\{0,1\}$ ，
- 贝努利分布： $y \sim \text{Bernoulli}(\mu)$ ，其中 μ 为分布的期望，表示 $y = 1$ 的概率。
- 概率密度函数为： $p(y; \mu) = \mu^y (1 - \mu)^{(1-y)}$ 。
- 在分类任务中，在给定输入 \mathbf{x} 的情况下，输出 y 用贝努利分布描述：
 $y|\mathbf{x} \sim \text{Bernoulli}(\mu(\mathbf{x}))$ ，其中期望 $\mu(\mathbf{x})$ 表示在给定 \mathbf{x} 的情况下， $y = 1$ 的概率。概率密度函数为： $p(y|\mathbf{x}; \mu) = \mu(\mathbf{x})^y (1 - \mu(\mathbf{x}))^{1-y}$ ，即

$$p(y = 1|\mathbf{x}) = \mu(\mathbf{x}), \quad p(y = 0|\mathbf{x}) = 1 - \mu(\mathbf{x})$$

- 在分类任务中，在给定输入 \mathbf{x} 的情况下，概率密度函数为：

$$p(y|\mathbf{x}; \mu) = \mu(\mathbf{x})^y (1 - \mu(\mathbf{x}))^{1-y}$$

- 期望 $\mu(\mathbf{x})$ 如何表示？

- 最简单的模型：线性模型

$$\mu(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

- 但是 $\mu(\mathbf{x})$ 表示给定 \mathbf{x} 的情况下， $y = 1$ 的概率，取值为区间 $[0,1]$ 。

- 所以，将 $\mathbf{w}^T \mathbf{x}$ 的输出范围转换到 $[0,1]$ ：**sigmoid**函数（S形函数）

➤ Sigmoid函数

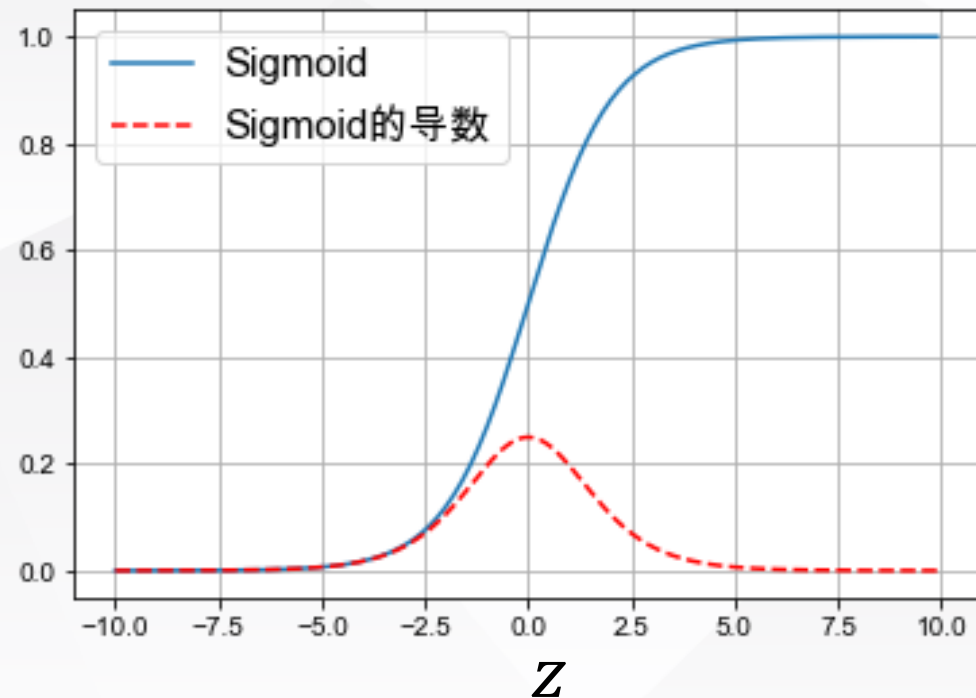
■ Sigmoid函数为:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z))$$

■ 亦被称为logistic函数或logit函数。

■ Logistic回归亦被称为logit回归。



Logistic回归虽然从名字上来看是回归算法，但其实际上是一个分类算法。

- 在Logistic回归模型中:

$$p(y = 1|\mathbf{x}) = \mu(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}),$$

$$p(y = 0|\mathbf{x}) = 1 - \mu(\mathbf{x}) = 1 - \sigma(\mathbf{w}^T \mathbf{x})$$

- 定义一个事件的几率(odds)为该事件发生的概率与不发生的概率的比值:

$$\frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} = \frac{\sigma(\mathbf{w}^T \mathbf{x})}{1 - \sigma(\mathbf{w}^T \mathbf{x})} = \frac{1/(1 + e^{-\mathbf{w}^T \mathbf{x}})}{e^{-\mathbf{w}^T \mathbf{x}}/(1 + e^{-\mathbf{w}^T \mathbf{x}})} = e^{\mathbf{w}^T \mathbf{x}}$$

- 两边同取log运算, 得到对数几率: $\ln \frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} = \ln(e^{\mathbf{w}^T \mathbf{x}}) = \mathbf{w}^T \mathbf{x}$

■ 对数几率: $\ln \frac{p(y=1|x)}{p(y=0|x)} = \ln(e^{w^T x}) = w^T x$

■ 当 $p(y=1|x) > p(y=0|x)$ 时, 如果取最大后验概率, x 的类别取 $y=1$:

$$\frac{p(y=1|x)}{p(y=0|x)} > 1, \quad \ln \frac{p(y=1|x)}{p(y=0|x)} = w^T x > 0。$$

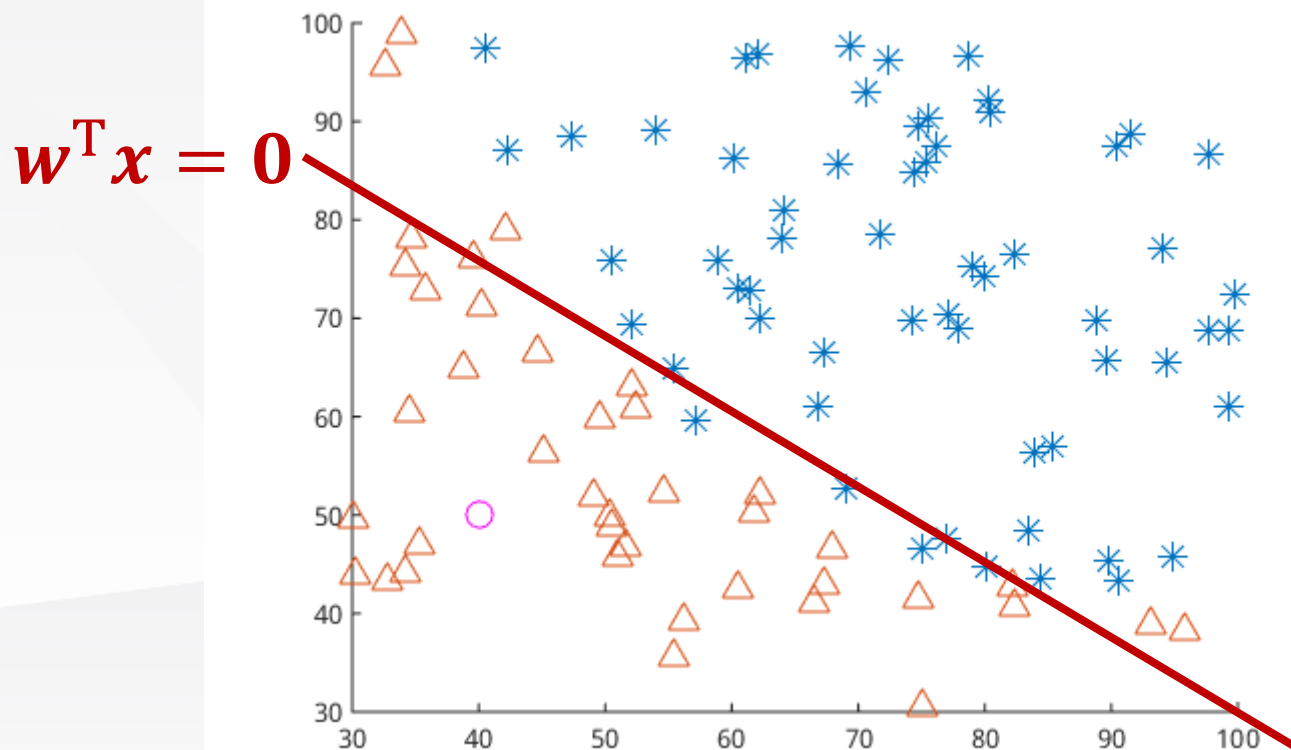
■ 当 $w^T x > 0$ 时, x 的类别取 $y=1$;

■ 当 $w^T x < 0$ 时, x 的类别取 $y=0$;

■ 当 $w^T x = 0$ 时, $y=1$ 的概率和 $y=0$ 的概率相等, 此时 x 位于决策面上。

可将 x 分类到任意一类, 或拒绝作出判断。

- 决策函数 $f(x) = w^T x$ 根据 $w^T x$ 的符号将输入空间 x 分出两个区域。
- $w^T x$ 为输入 x 的线性函数，所以 Logistic 回归模型是一个线性分类模型。



➤ 决策边界

- 更一般地：根据需要划分的类别，分类器将输入空间 \mathcal{X} 划分为一些互不相交的区域。这些区域的边界叫做决策边界(decision boundaries)。
- 预测函数的形式不同，会使得决策面或光滑、或粗糙。
 - 决策面是输入的线性函数，称为线性决策面，对应的分类器就是线性分类器。
- 分类器为每个类别分配一个判别函数，根据判别函数来判断一个新样本属于该类别的可能性。
- 假设有 C 个类别，则有得到 C 个判别函数： $\delta_c(\mathbf{x}), c \in \{1, \dots, C\}$ 。
 - 对一个新的样本 \mathbf{x} ，一般是找到最大的 $\delta_c(\mathbf{x})$ ，即该样本的类别为：

$$\hat{y} = \operatorname{argmax}_c \delta_c(\mathbf{x})$$

Outline

- 分类任务
- Logistic回归
- **Logistic回归的损失函数**
- Logistic回归的优化算法
- Logistic回归的多类分类任务
- 分类任务中的样本不均衡问题
- 分类模型的评价指标
- Sklearn中的Logistic回归API
- Logistic回归案例分析

分类任务的损失函数

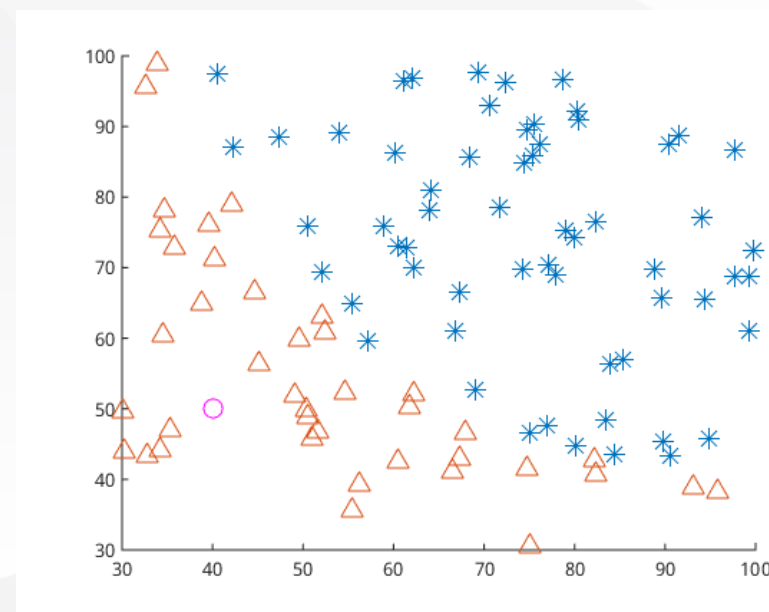
■ 0/1 损失：预测类别正确损失为0，否则为1，记为

$$L(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ 1 & y \neq \hat{y} \end{cases}$$

■ 0/1 损失不连续，优化计算不方便。

■ 寻找其他替代损失函数（Surrogate Loss Function）

- 通常是凸函数，计算方便且和0/1损失是一致的



■ Logistic回归模型: $y|\mathbf{x} \sim \text{Bernoulli}(\mu(\mathbf{x}))$

$$p(y|\mathbf{x}; \mu(\mathbf{x})) = \mu(\mathbf{x})^y (1 - \mu(\mathbf{x}))^{(1-y)}$$

$$\mu(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

■ log似然函数为:

$$\ell(\mu) = \ln p(\mathcal{D}) = \sum_{i=1}^N \ln p(y_i|\mathbf{x}_i)$$

$$= \sum_{i=1}^N \ln(\mu(\mathbf{x}_i)^{y_i} (1 - \mu(\mathbf{x}_i))^{(1-y_i)})$$

- 定义负log似然损失为：

$$L(y, \mu(\mathbf{x})) = -y \ln(\mu(\mathbf{x})) - (1 - y) \ln(1 - \mu(\mathbf{x}))$$

- 则极大似然估计等价于最小训练集上的负log似然损失：

$$\begin{aligned} J(\mu) &= \sum_{i=1}^N L(y_i, \mu(\mathbf{x}_i)) = \sum_{i=1}^N -y_i \ln(\mu(\mathbf{x}_i)) - (1 - y_i) \ln(1 - \mu(\mathbf{x}_i)) \\ &= -\ell(\mu) \end{aligned}$$

交叉熵损失

- 负log似然损失 $L(y, \mu(x)) = -y \ln(\mu(x)) - (1 - y) \ln(1 - \mu(x))$
- 亦被称为**交叉熵损失** (Cross Entropy Loss)。
- 交叉熵：两个分布之间的差异（已知真实分布情况下，预测分布与真实分布之间的差异）
 - 假设已知真值 $y = 1$ ，即 $y|x \sim \text{Bernoulli}(1)$,
 - 假设 $\hat{y} = 1$ 的概率为 $\mu(x)$ ， $\hat{y}|x \sim \text{Bernoulli}(\mu(x))$
 - 即这两个分布之间的交叉熵为：

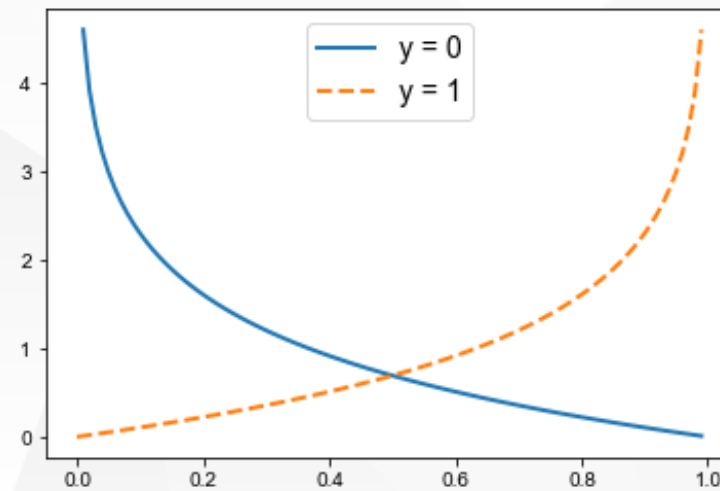
$$\begin{aligned} CE(y, \mu(x)) &= - \sum_y p(y|x) \ln p(\hat{y}|x) = -p(y = 1|x) \ln p(\hat{y} = 1|x) - p(y = 0|x) \ln p(\hat{y} = 0|x) \\ &= -\ln \mu(x) \end{aligned}$$

综合： $CE(y, \mu(x)) = \begin{cases} -\ln \mu(x) & \text{if } y = 1 \\ -\ln(1 - \mu(x)) & \text{otherwise} \end{cases}$

定义Probability of ground truth class为：

$$p_t = \begin{cases} \mu(x) & \text{if } y = 1 \\ 1 - \mu(x) & \text{otherwise} \end{cases}$$

则 $CE(y, \mu(x)) = -\ln(p_t)$



预测输出与真值 y 差得越多，损失的值越大。

Logistic回归的目标函数

- Logistic回归的损失函数采用交叉熵损失

$$L(y, \mu(\mathbf{x})) = -y \ln(\mu(\mathbf{x})) - (1 - y) \ln(1 - \mu(\mathbf{x}))$$

- 其中 y 为真值, $\mu(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$ 为预测值为1的概率。

- 同其他机器学习模型一样, Logistic回归的目标函数也包括两项: 训练集上的损失和正则项

$$J(\mathbf{w}; \lambda) = \sum_{i=1}^N L(y_i, \mu(\mathbf{x}_i; \mathbf{w})) + \lambda R(\mathbf{w})$$

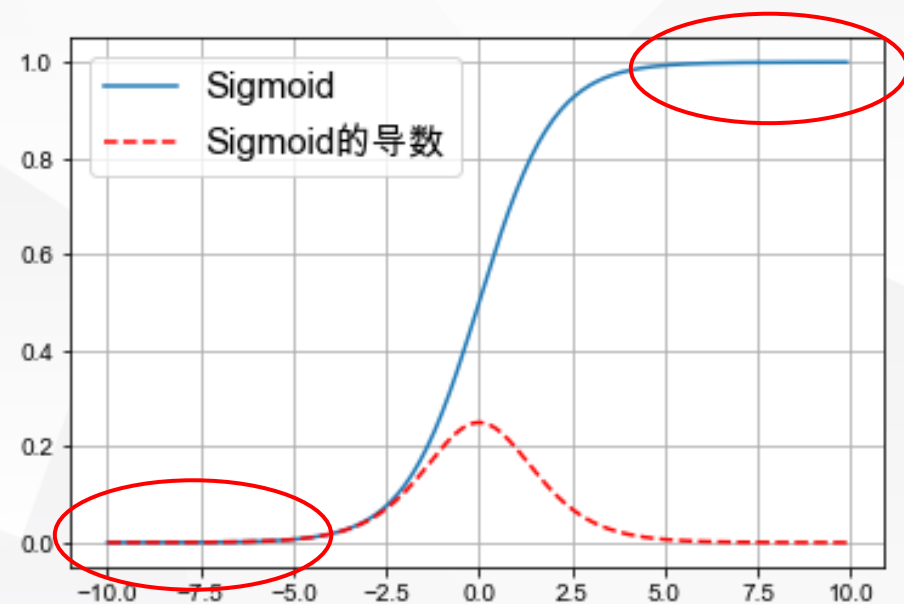
- 类似回归任务, 正则项 $R(\mathbf{w})$ 可为L1正则、L2正则、L1正则+L2正则

Logistic回归中正则的必要性

■ 线性回归：目标函数可以不加正则项（OLS）

■ 但Logistic回归：必须加正则

- 当训练完全可分时，为了使logistic损失和最小（0），每个 $L(y_i, \mu(x_i)) = 0$ ， $\mu(x_i) = 1/0$ 。
- 而 $\mu(x_i) = \sigma(\mathbf{w}^T \mathbf{x}_i) = 1/0$ ，要求 $\mathbf{w}^T \mathbf{x}_i = \infty / -\infty$ ， $|w_j| = \infty$ 。
- 因此必须加正则，找到与训练数据拟合好的最简单模型。



➤ Sklearn中的Logistic回归

Class sklearn.linear_model.LogisticRegression(*penalty*='l2', *dual*=False, *tol*=0.0001, *C*=1.0, *fit_intercept*=True, *intercept_scaling*=1, *class_weight*=None, *random_state*=None, *solver*='liblinear', *max_iter*=100, *multi_class*='ovr', *verbose*=0, *warm_start*=False, *n_jobs*=1)

■实现的目标函数为

$$J(\mathbf{w}; C) = C \sum_{i=1}^N L(y_i, \mu(\mathbf{x}_i; \mathbf{w})) + R(\mathbf{w})$$

■其中超参数C起到正则作用，C越大，正则越少。

Outline

- 分类任务
- Logistic回归
- Logistic回归的损失函数
- Logistic回归的优化算法
 - 梯度下降法
 - 牛顿法
- Logistic回归的多类分类任务
- 分类任务中的样本不均衡问题
- 分类模型的评价指标
- Sklearn中的Logistic回归API
- Logistic回归案例分析

Logistic回归的梯度下降求解

■ Logistic回归模型没有解析解，迭代求解

- 一阶：梯度下降、随机梯度下降（SGD）、随机平均梯度法（SAG）、随机平均梯度法改进版（SAGA）、共轭梯度、坐标轴下降
- 二阶：牛顿法及拟牛顿法（BFGS、L-BFGS）

■ 目标函数：

$$J(\mathbf{w}, \lambda) = \sum_{i=1}^N L(y_i, \mu(\mathbf{x}_i; \mathbf{w})) + \lambda R(\mathbf{w})$$

■ 正则项处理同线性回归。

- L2正则函数连续，可采用梯度下降法/牛顿法等优化方法
- 由于L1正则函数不连续，所以梯度下降法/牛顿法不适用。类似Lasso求解，可采用坐标轴下降法

■我们先来看训练集上的损失函数和部分

$$J_1(\mathbf{w}) = \sum_{i=1}^N (-y_i \ln(\mu(\mathbf{x}_i, \mathbf{w})) - (1 - y_i) \ln(1 - \mu(\mathbf{x}_i, \mathbf{w})))$$

■则梯度

$$\begin{aligned} g_1(\mathbf{w}) &= \frac{\partial J_1(\mathbf{w})}{\partial \mathbf{w}} = - \sum_{i=1}^N \left(y_i \frac{1}{\mu(\mathbf{x}_i, \mathbf{w})} - (1 - y_i) \frac{1}{1 - \mu(\mathbf{x}_i, \mathbf{w})} \right) \frac{\partial \mu(\mathbf{x}_i, \mathbf{w})}{\partial \mathbf{w}} \\ &= - \sum_{i=1}^N \left(y_i \frac{1}{\mu(\mathbf{x}_i, \mathbf{w})} - (1 - y_i) \frac{1}{1 - \mu(\mathbf{x}_i, \mathbf{w})} \right) \mu(\mathbf{x}_i, \mathbf{w})(1 - \mu(\mathbf{x}_i, \mathbf{w})) \frac{\partial (\mathbf{w}^T \mathbf{x}_i)}{\partial \mathbf{w}} \\ &= - \sum_{i=1}^N (y_i(1 - \mu(\mathbf{x}_i, \mathbf{w})) - (1 - y_i)\mu(\mathbf{x}_i, \mathbf{w})) \mathbf{x}_i \\ &= \mathbf{X}^T (\boldsymbol{\mu} - \mathbf{y}) \end{aligned}$$

形式同线性回归模型

$$\begin{aligned} \mu(\mathbf{x}, \mathbf{w}) &= \sigma(\mathbf{w}^T \mathbf{x}) \\ \sigma(z) &= \frac{1}{1 + e^{-z}} \\ \frac{d\sigma(z)}{dz} &= \sigma(z)(1 - \sigma(z)) \\ \frac{\partial (\mathbf{w}^T \mathbf{x})}{\partial \mathbf{w}} &= \mathbf{x} \end{aligned}$$

➤ Hessian矩阵

■ 损失函数和的梯度为（为书写简单，省略下标1）： $g(\mathbf{w}) = \mathbf{X}^T(\boldsymbol{\mu} - \mathbf{y})$

■ 则Hessian矩阵为：

$$H(\mathbf{w}) = \frac{\partial g(\mathbf{w})}{\partial \mathbf{w}^T} = \frac{\partial (\sum_{i=1}^N (\mu(\mathbf{x}_i, \mathbf{w}) - y_i) \mathbf{x}_i)}{\partial \mathbf{w}^T}$$

$$= \sum_{i=1}^N \mu_i(1 - \mu_i) \mathbf{x}_i \mathbf{x}_i^T$$

$$= \mathbf{X}^T \mathbf{S} \mathbf{X}$$

$$\boxed{\frac{\partial (\mathbf{a}^T \mathbf{y})}{\partial \mathbf{y}} = \mathbf{a}^T}$$

$$\boxed{\frac{\partial \mu(\mathbf{x}_i; \mathbf{w})}{\partial \mathbf{w}^T} = \mathbf{x}_i^T \mu_i(1 - \mu_i)}$$

$$\boxed{\mathbf{S} \triangleq \text{diag}(\mu_i(1 - \mu_i))}$$

■ $H(\mathbf{w})$ 为正定矩阵，梯度下降能找到全局最小值。

Outline

- 分类任务
- Logistic回归
- Logistic回归的损失函数
- **Logistic回归的优化算法**
 - 梯度下降法
 - **牛顿法**
- Logistic回归的多类分类任务
- 分类任务中的样本不均衡问题
- 分类模型的评价指标
- Sklearn中的Logistic回归API
- Logistic回归案例分析

➤ 牛顿法

- 牛顿法亦被称为牛顿-拉夫逊（Newton-Raphson）方法。牛顿在17世纪提出用来求解方程的根。
- 假设点 x^* 为函数 $f(x)$ 的根，则 $f(x^*) = 0$ 。
- 现在我们将函数 $f(x)$ 在点 $x^{(t)}$ 处进行一阶泰勒展开有

$$f(x) \approx f(x^{(t)}) + (x - x^{(t)})f'(x^{(t)})$$

- 假设点 $x^{(t+1)}$ 为函数 $f(x)$ 的根，则有

$$f(x^{(t+1)}) \approx f(x^{(t)}) + (x^{(t+1)} - x^{(t)})f'(x^{(t)}) = 0$$

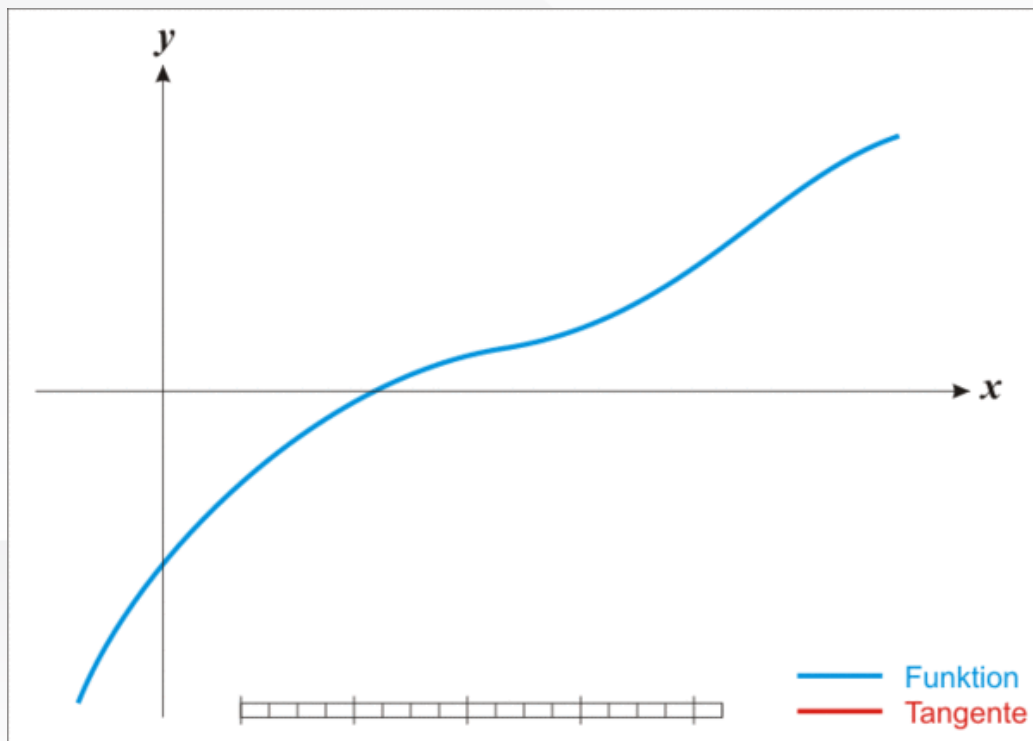
- 那么可以得到

$$x^{(t+1)} = x^{(t)} - \frac{f(x^{(t)})}{f'(x^{(t)})}。$$

通过迭代的方式求解方程的解

■ 牛顿法通过通过迭代的方式求解方程 $f(x) = 0$ 的解

$$x^{(t+1)} = x^{(t)} - \frac{f(x^{(t)})}{f'(x^{(t)})}。$$



➤ 牛顿法求解目标函数极值

- 对于最优化问题，极值点处函数的一阶导数为0。
- 因此我们可以对一阶导数 $g(x) = f'(x)$ 利用牛顿法通过迭代的方式来求得最优解，即相当于求一阶导数对应函数的根。

$$x^{(t+1)} = x^{(t)} - \frac{f(x^{(t)})}{f'(x^{(t)})} \xrightarrow{g(x) = f'(x)} x^{(t+1)} = x^{(t)} - \frac{g(x^{(t)})}{g'(x^{(t)})} = x^{(t)} - \frac{f'(x^{(t)})}{f''(x^{(t)})}$$

牛顿法是二阶最优化算法。

➤ 牛顿法求解目标函数极值

- 对多元函数 $f(x_1, \dots, x_D)$, 一阶导数换成梯度: $\mathbf{g} = \nabla f(x_1, \dots, x_D)$, 二阶导数换成海森 (Hessian) 矩阵 \mathbf{H}

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial^2 x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_D} & \frac{\partial^2 f}{\partial x_D \partial x_2} & \cdots & \frac{\partial^2 f}{\partial^2 x_D^2} \end{bmatrix}$$

- 则牛顿法迭代公式为: $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \mathbf{H}^{-1}(\mathbf{x}^{(t)})\mathbf{g}(\mathbf{x}^{(t)})$

➤ 牛顿法求解目标函数极值

- 1. 从 $t = 0$ 开始, 初始化 $\mathbf{x}^{(0)}$ 为随机值;
- 2. 计算目标函数 $f(\mathbf{x})$ 在点 $\mathbf{x}^{(t)}$ 的梯度 $\mathbf{g}^{(t)} = \nabla f(\mathbf{x}^{(t)})$ 和海森矩阵 $\mathbf{H}^{(t)} = \mathbf{H}(\mathbf{x}^{(t)})$;
- 3. 计算移动方向: $\mathbf{d}^{(t)} = (\mathbf{H}^{(t)})^{-1} \mathbf{g}^{(t)}$ (一般用线性方程组计算 $\mathbf{d}^{(t)}$: $\mathbf{H}^{(t)} \mathbf{d}^{(t)} = -\mathbf{g}^{(t)}$ 。线性方程组求解可用共轭梯度等方法求解);
- 4. 根据迭代公式, 更新 \mathbf{x} 的值: $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \mathbf{d}^{(t)}$;
- 5. 判断是否满足迭代终止条件。如果满足, 循环结束, 返回最佳参数 $\mathbf{x}^{(t+1)}$ 和目标函数极小值 $f(\mathbf{x}^{(t+1)})$; 否则转到第2步。

一阶梯度法: $\mathbf{d}^{(t)} = -\eta \mathbf{g}^{(t)}$,

二阶牛顿法: $\mathbf{d}^{(t)} = -(\mathbf{H}^{(t)})^{-1} \mathbf{g}^{(t)}$

- 牛顿法比一般的梯度下降法收敛速度快。
- 但在高维情况下，计算目标函数的二阶偏导数的复杂度高，而且有时候目标函数的海森矩阵无法保持正定，不存在逆矩阵，此时牛顿法将不再能使用。
- 因此人们提出了**拟牛顿法**：不用二阶偏导数构造出可以近似Hessian矩阵(或Hessian矩阵的逆矩阵)的正定对称矩阵，进而再逐步优化目标函数。
- 不同的Hessian矩阵构造方法产生了不同的拟牛顿法
 - BFGS / L-BFGS

推荐阅读：

1. 谈谈常见的迭代优化方法

<https://blog.csdn.net/aws3217150/article/details/50548177>

2. Mathematical optimization: finding minima of functions

https://www.scipy-lectures.org/advanced/mathematical_optimization/

Logistic回归的牛顿求解: IRLS

■ 损失函数和的梯度为: $g(\mathbf{w}) = \mathbf{X}^T(\boldsymbol{\mu} - \mathbf{y})$

■ Hessian矩阵为: $H(\mathbf{w}) = \mathbf{X}^T \mathbf{S} \mathbf{X}$

■ 迭代公式: $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \left(H(\mathbf{w}^{(t)}) \right)^{-1} g(\mathbf{w})$

$$= \mathbf{w}^{(t)} - \left(\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X} \right)^{-1} \mathbf{X}^T (\boldsymbol{\mu} - \mathbf{y})$$

$$= \left(\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X} \right)^{-1} \left[\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X} \mathbf{w}^{(t)} - \mathbf{X}^T (\boldsymbol{\mu} - \mathbf{y}) \right]$$

$$= \left(\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{S}^{(t)} \left[\mathbf{X} \mathbf{w}^{(t)} - \left(\mathbf{S}^{(t)} \right)^{-1} (\boldsymbol{\mu} - \mathbf{y}) \right]$$

$$= \left(\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{S}^{(t)} \mathbf{z}^{(t)}$$

$$\mathbf{z}^{(t)} \triangleq \mathbf{X} \mathbf{w}^{(t)} - \left(\mathbf{S}^{(t)} \right)^{-1} (\boldsymbol{\mu} - \mathbf{y})$$

Iteratively Reweighted least Squares

最小二乘: $\hat{\mathbf{w}}_{OLS} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$

加权最小二乘 (每个样本的权重为 $s_i^{-1/2}$): $\hat{\mathbf{w}}_{OLS_weight} = (\mathbf{X}^T \mathbf{S} \mathbf{X})^{-1} \mathbf{S} \mathbf{X}^T \mathbf{y}$

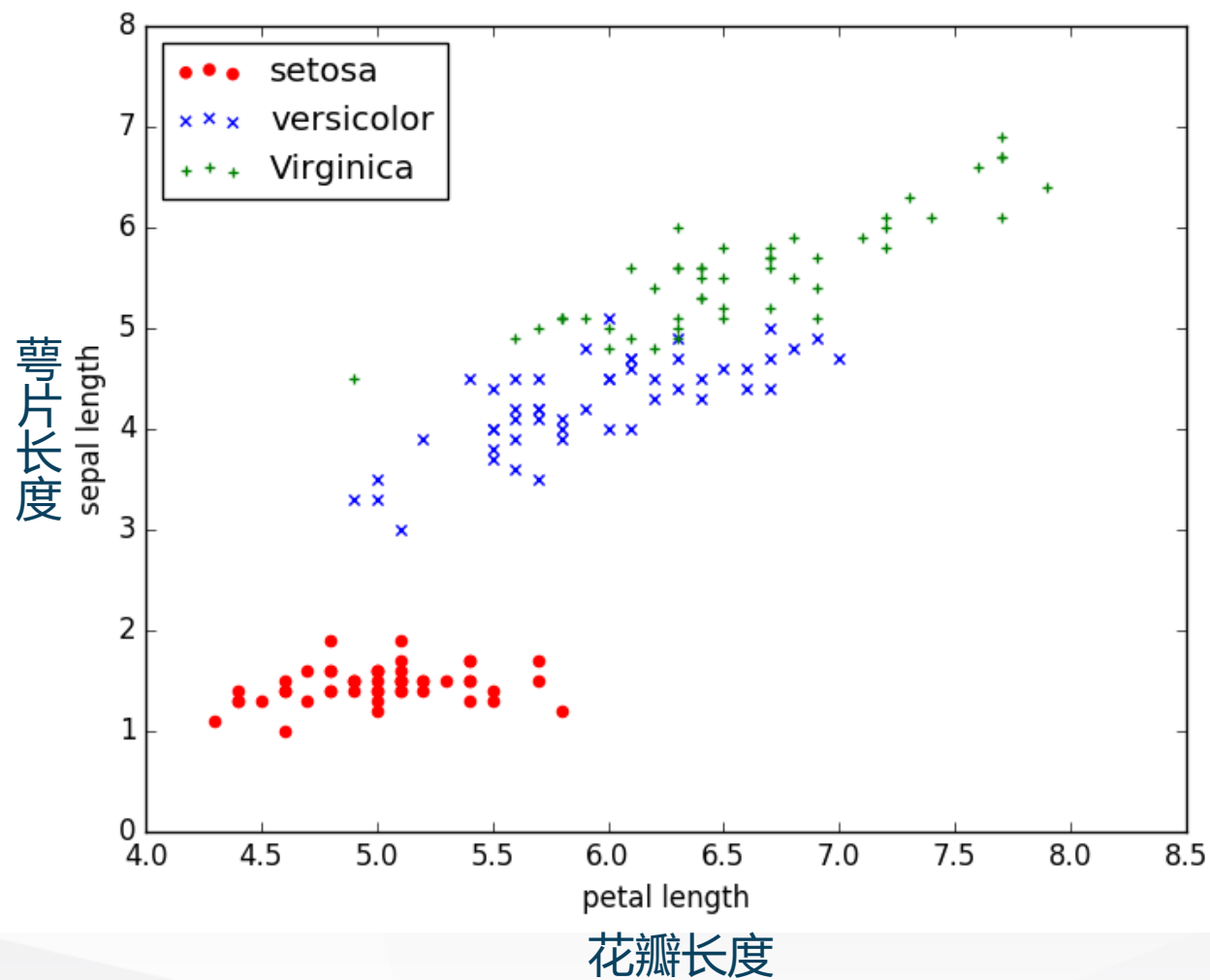
Logistic回归的牛顿求解：IRLS

- 初始值： $\mathbf{w}^{(0)} =$
- IRLS代入迭代公式为： $\mathbf{w}^{(t+1)} = (\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{S}^{(t)} \mathbf{z}^{(t)}$
- 加权最小二乘又通过共轭梯度法求解，因此Scikit-Learn中的采用牛顿法求解的算法取名为“Newton-CG”。

Outline

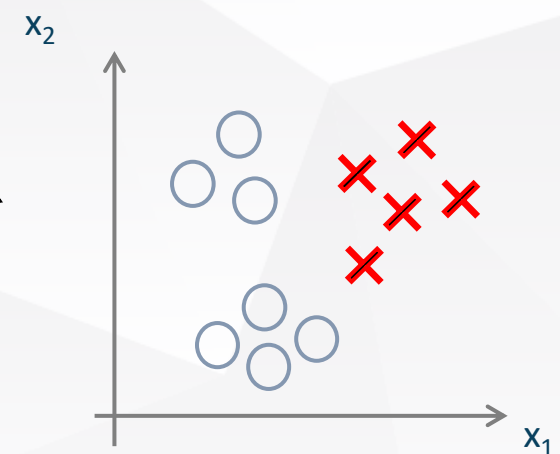
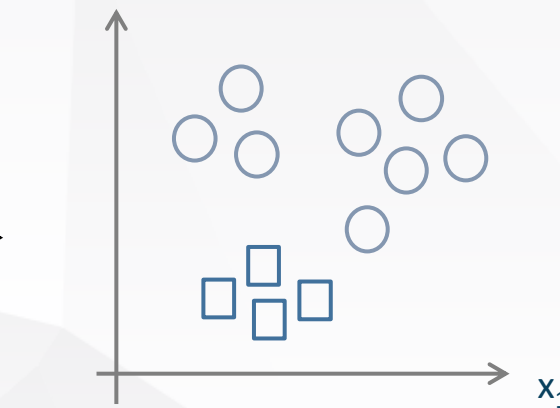
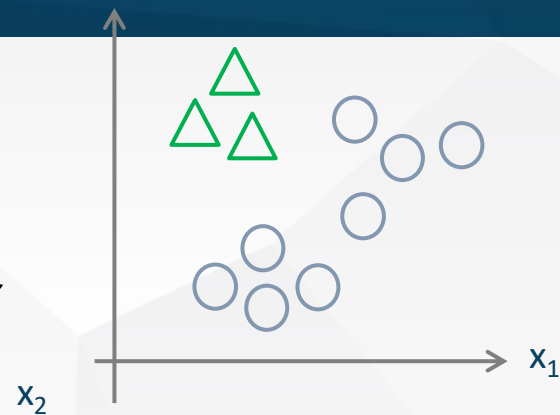
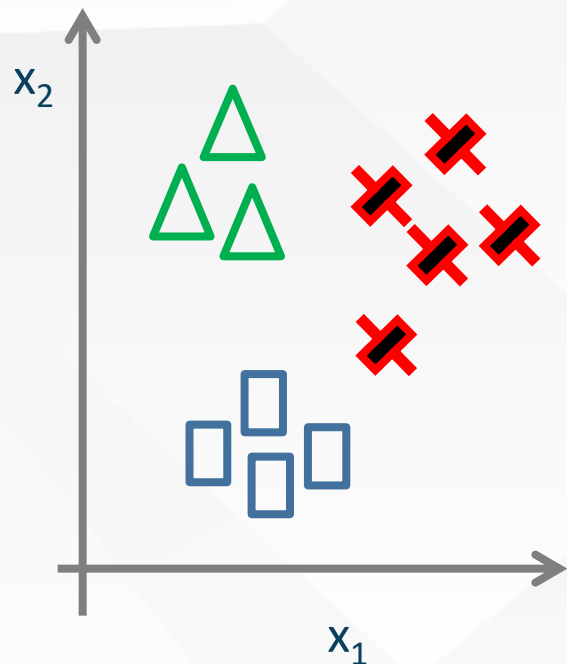
- 分类任务
- Logistic回归
- Logistic回归的损失函数
- Logistic回归的优化算法
- ▶ ■ Logistic回归的多类分类任务
- 分类任务中的样本不均衡问题
- 分类模型的评价指标
- Sklearn中的Logistic回归API
- Logistic回归案例分析

■ 例：Iris花分类中，一共有3种鸢尾花



➤ 1 vs. 其他

■ 一对其他(One-vs-Rest, OvR)



$$f_w^c(\mathbf{x}) = p(y = c | \mathbf{x}, W), c = 1, 2, 3$$

每类的模型都有自己正则参数和权重参数

➤ 1 vs. 其他

- 对每个类别 c ，训练一个Logistic回归分类器 $f_w^c(x)$ ，预测 $y = c$ 概率
- 对新的输入 x ，选择使得 $f_w^c(x)$ 最大的类别作为预测（最大后验估计，MAP)

$$\hat{y} = \operatorname{argmax}_c f_w^c(x)$$

➤ 多项分布直接实现多类分类

- 贝努利 (Bernoulli) 分布的输出只有两种取值。
- Multinoulli分布, 或称为范畴分布(categorical distribution), 输出有 K 种取值。
- 类似用Bernoulli分布描述两类分类的概率分布, 可用Multinoulli分布描述多类分类的概率分布, 其参数为向量 $\theta = (\theta_1, \dots, \theta_C)$, 其中 $\sum_{c=1}^C \theta_c = 1$, 其中每一个分量 θ_c 表示第 c 个状态的概率。我们用符号 $Cat(y; \theta)$ 表示

- 注意:
- Multinoulli是个人造词, 经典概率书并没有将其作为一种概率分布单独表示, 而是用多项 (Multinomial) 分布统称 (Scikit-Learn采用的是Multinoulli)。有些参考书种称其为广义Bernoulli分布, 或离散分布。
- 多重贝努利试验成功次数输出用二项 (Binomial) 分布表示
- 多重Multinoulli (如掷骰子) 的输出用多项 (Multinomial) 分布表示: Multinoulli 是多项Multi+oulli (Bernoulli的后几个字母)



- 将类别 y 用独热编码（编码为 C 维向量，当 $y = c$ 时，第 c 维为1，其他元素均为0），记为向量 \mathbf{y}
 - 则Multinoulli分布的概率函数为： $Cat(\mathbf{y}; \boldsymbol{\theta}) = \prod_{c=1}^C \theta_c^{y_c}$ ，
 - 其中 y_c 表示向量 \mathbf{y} 的第 c 个元素。
-
- 或者用标量形式记为： $Cat(y; \boldsymbol{\theta}) = \prod_{c=1}^C \theta_c^{I(y=c)}$ ，
 - 其中 $I(\cdot)$ 为示性函数，当括号中条件满足时函数值为1，否则为0。



- 类似两类分类模型推导，假设输出 $y = c$ 的概率可以由 \mathbf{x} 的线性组合再经过sigmoid函数变换得到

$$p(y = c | \mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x})}$$

- 上述等式右边为**Softmax函数**。Softmax函数为sigmoid函数的推广，将 C 维向量的每个元素转换为 $[0,1]$ 的数，且变换后元素之和为1：

$$\sigma(z_c) = \frac{e^{z_c}}{\sum_{c'=1}^C e^{z_{c'}}}$$

- 因此得到的分类器被称为Softmax分类器。



■将类别 y 用独热编码为向量 \mathbf{y} : $y_c = I(y = c)$

■向量 $\boldsymbol{\mu}$ 表示Multinoulli分布的参数:

$$\mu_c = p(y = c | \mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x})}$$

■令 $\mu_{i,c} = \frac{\exp(\mathbf{w}_c^T \mathbf{x}_i)}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x}_i)}$, $y_{i,c}$ 为第 i 个样本的类别, 则Softmax分类模型的log似然函数为:

$$\begin{aligned} \ell(\mathbf{W}) &= \sum_{i=1}^N \ln \left(\prod_{c=1}^C \mu_{i,c}^{y_{i,c}} \right) \\ &= \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \ln(\mu_{i,c}) \end{aligned}$$

$$\text{Cat}(\mathbf{y}; \boldsymbol{\theta}) = \prod_{c=1}^C \theta_c^{y_c}$$

- 定义Softmax损失为负log似然:

$$L(\mathbf{y}, \boldsymbol{\mu}) = - \sum_{c=1}^C y_c \ln(\mu_c)$$

- 则极大似然估计等价于最小训练集上的Softmax损失/负log似然损失:

$$J(\mathbf{W}) = - \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \ln(\mu_{i,c})$$

$$\mu_c = p(y = c | \mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x})}$$

$$= - \sum_{i=1}^N \left(\sum_{c=1}^C y_{i,c} \mathbf{w}_c^T \mathbf{x} \left(-\log \left(\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x}) \right) \right) \right)$$

Scikit-Learn中实现Logistic回归的类

```
class sklearn.linear_model.LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_classes='ovr', verbose=0, warm_start=False, n_jobs=1)
```

- 参数multi_class决定了多类分类的实现方式，可选：

- a) 'ovr' : 即1对其他 (one-vs-rest, OvR) , 将多类分类转化为多个二类分类任务。为了完成第*c*类的分类决策，将所有第*c*类的样本作为正例，除了第*c*类样本以外的所有样本都作为负例，每个类别的二分类器单独训练。
- b) 'multinomial' : Softmax回归分类，对多项分布概率整体进行训练。

注意：multi_class选择会影响优化算法solver参数的选择

OvR: 可用所有的solver

Multinomial: 只能选择newton-cg, lbfgs和sag / saga (liblinear不支持)

Outline

- 分类任务
- Logistic回归
- Logistic回归的损失函数
- Logistic回归的优化算法
- Logistic回归的多类分类任务
- 分类任务中的样本不均衡问题
- 分类模型的评价指标
- Sklearn中的Logistic回归API
- Logistic回归案例分析

➤ 不平衡数据的出现场景

- 搜索引擎的点击预测：点击的网页往往占据很小的比例
- 电子商务领域的商品推荐：推荐的商品被购买的比例很低
- 信用卡欺诈检测
- 网络攻击识别
- 医疗中得病的案例少

- 从数据的角度：抽样，从而使得不同类别的数据相对均衡
- 从算法的角度：考虑不同误分类情况代价的差异性对算法进行优化

- 随机下抽样：从多数类中随机选择少量样本再合并原有少数类样本作为新的训练数据集
 - 有放回抽样
 - 无放回抽样
 - 会造成一些信息缺失，选取的样本可能有偏差
- 随机上抽样：随机复制少数类样本
 - 扩大数据集，造成模型训练复杂度加大，另一方面也容易造成模型的过拟合

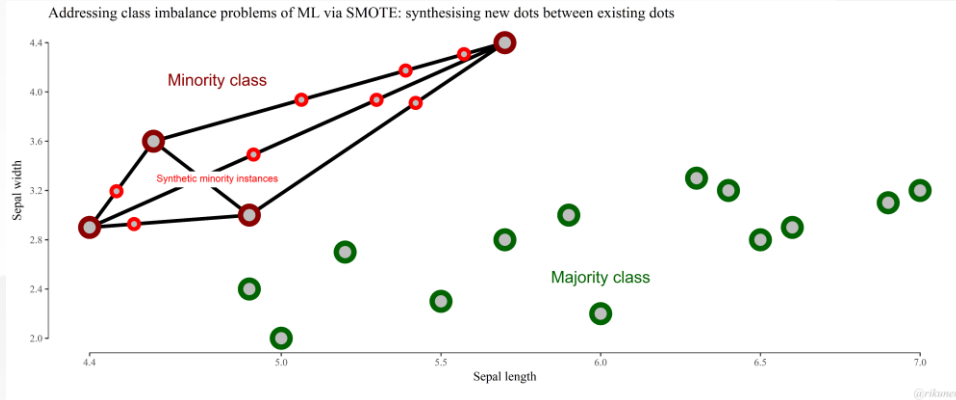
■ EasyEnsemble算法

- 对于多数类样本，通过 n 次有放回抽样生成 n 份子集
- 少数类样本分别和这 n 份样本合并训练一个模型： n 个模型
- 最终模型： n 个模型预测结果的平均值

■ BalanceCascade（级联）算法

- 从多数类中有效地选择一些样本与少数类样本合并为新的数据集进行训练
- 训练好的模型每个多数类样本进行预测。若预测正确，则不考虑将其作为下一轮的训练样本
- 依次迭代直到满足某一停止条件，最终的模型是多次迭代模型的组合

- SMOTE (Synthetic Minority Over-sampling Technique) : 基于“插值”为少数类合成新的样本
- 对少数类的一个样本 i , 其特征向量为 \mathbf{x}_i
 - 1. 从少数类的全部 N 个样本中找到样本 \mathbf{x}_i 的 K 个近邻 (如欧氏距离), 记为 $\mathbf{x}_{i(near)}$, $near \in \{1, \dots, K\}$
 - 2. 从这 K 个近邻中随机选择一个样本 $\mathbf{x}_{i(nn)}$, 再生成一个 0到 1之间的随机数 ζ , 从而合成一个新样本: $\mathbf{x}_{i1} = (1 - \zeta)\mathbf{x}_i + \zeta\mathbf{x}_{i(near)}$ 。



http://rikunert.com/SMOTE_explained

- SMOTE算法摒弃了随机过采样复制样本的做法，可以防止随机过采样易过拟合的问题。实践证明此方法可以提高分类器的性能。
- SMOTE对高维数据不是很有效。
- 当生成合成性实例时，SMOTE并不会把来自其他类的相邻实例考虑进来，这导致了类重叠的增加，并会引入额外的噪音。
- 为了解决SMOTE算法的这一缺点提出一些改进算法，如Borderline-SMOTE算法。

➤ 代价敏感学习

- 在算法层面上解决不平衡数据学习的方法主要是基于代价敏感学习算法(Cost-Sensitive Learning)。
- 代价敏感学习方法的核心要素是代价矩阵：不同类型的误分类情况导致的代价不一样。

Predict \ True	0	1
	0	1
0	c_{00}	c_{01}
1	c_{10}	c_{11}

■ 基于代价矩阵分析，代价敏感学习方法主要有以下三种实现方式

- 从贝叶斯风险理论出发，把代价敏感学习看成是分类结果的一种后处理，按照传统方法学习到一个模型，以实现损失最小为目标对结果进行调整
 - 不依赖所用具体的分类器
 - 缺点：要求分类器输出值为概率
- 从学习模型出发，对具体学习方法的改造，使之能适应不平衡数据下的学习
 - 如：代价敏感的支持向量机，决策树，神经网络
- 从预处理的角度出发，将代价用于权重的调整，使得分类器满足代价敏感的特性
 - 如不同类别的样本的权值不同

➤ Scikit-Learn中的不平衡样本分类处理

- 类别权重class_weight
- 样本权重sample_weight

类别权重class_weight

- class_weight参数用于标示分类模型中各类别样本的权重
- 1. 不考虑权重，即所有类别的权重相同
- 2. balanced：自动计算类别权重
 - 某类别的样本量越多，其权重越低；样本量越少，则权重越高
 - 类权重计算方法为： $n_samples / (n_classes * np.bincount(y))$ ：
n_samples为样本数，n_classes为类别数量，np.bincount(y)输出每个类的样本数
- 3. 手动指定各个类别的权重
 - 如对于0,1二类分类问题，可以定义class_weight={0:0.9, 1:0.1}，即类别0的权重为90%，而类别1的权重为10%

➤ 样本权重sample_weight

- 模型训练: `fit($X, y, sample_weight=None$)`
 - 其中参数sample_weight为样本权重参数
- 如果既用了class_weight, 又用了sample_weight, 那么样本的真正权重是: `class_weight*sample_weight`



- 判断一个分类器对所用样本的分类能力或者在不同的应用场合时，需要有不同的指标。
- Scikit-Learn中，评价指标计算可对每个样本施加权重，权重通过参数`sample_weight`指定。

Outline

- 分类任务
- Logistic回归
- Logistic回归的损失函数
- Logistic回归的优化算法
- Logistic回归的多类分类任务
- 分类任务中的样本不均衡问题
- 分类模型的评价指标
- Sklearn中的Logistic回归API
- Logistic回归案例分析

➤ 分类模型的评价指标

■ 1. 正确率：Sklearn中分类任务的默认评价指标

$$\text{Accuracy}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=0}^N I(\hat{y}_i = y_i)$$

- 其中 $I(\cdot)$ 为示性函数，当括号中条件满足时函数值为1，否则为0

■ 2. 负log似然损失（交叉熵损失）

$$\text{logloss}(\mathbf{Y}, \hat{\mathbf{Y}}) = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \ln(\hat{y}_{i,c})$$

■ 3. 合页损失 (SVM中的损失函数)

$$\text{HingeLoss}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=0}^N \max(1, \hat{y}_i y_i)$$

分类模型的评价指标

■ 混淆矩阵：对 C 类分类问题，混淆矩阵为 $C \times C$ 的矩阵

例：手写数字识别的混淆矩阵

真值	[87	0	0	0	1	0	0	0	0	0]
	[0	88	1	0	0	0	0	0	1	1]
	[0	0	85	1	0	0	0	0	0	0]
	[0	0	0	79	0	3	0	4	5	0]
	[0	0	0	0	88	0	0	0	0	4]
	[0	0	0	0	0	88	1	0	0	2]
	[0	1	0	0	0	0	90	0	0	0]
	[0	0	0	0	0	1	0	88	0	0]
	[0	0	0	0	0	0	0	0	88	0]
	[0	0	0	1	0	1	0	0	0	90]]
预测值											

矩阵的第 i 行第 j 列的元素值表示将真实类别标签为 i 类的样本预测为第 j 类的样本数目。

对角线元素越大越好。

➤ 两类分类任务的评价指标

■ 混淆矩阵

真值

预测值

y	$\hat{y} = 1$	$\hat{y} = 0$	Σ
y = 1	TP	FN	N_+
y = 0	FP	TN	N_-
Σ	\hat{N}_+	\hat{N}_-	

$$\text{Precision} = \frac{TP}{\hat{N}_+}$$

精度：检测结果真正为正的比例（质）

$$\text{Recall} = \frac{TP}{N_+}$$

召回率：被正确检测到的正样本的比例（量），TPR（True Positive Rate）

$$\text{FPR} = \frac{FP}{N_-}$$

假阳率：负样本被预测成了正样本的比例，FPR（False Positive Rate）

$$F1 = \frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

F1分数： Precision 和Recall调和平均值

注意：准确率和召回率是互相影响的。一般情况下准确率高、召回率就低；召回率低、准确率高。

两类分类任务的评价指标

■ 混淆矩阵

真值

预测值

y	预测值		Σ
	$\hat{y} = 1$	$\hat{y} = 0$	
$y = 1$	TP	FN	N_+
$y = 0$	FP	TN	N_-
Σ	\hat{N}_+	\hat{N}_-	

PR曲线

精度、准确率：预测结果为真的样本中真正为真的比例

$$\text{Precision} = \frac{TP}{\hat{N}_+}$$

召回率：预测结果召回了多少真正的真样本；
真阳率：有多少真正的正样本被预测为真

$$\text{TPR} = \text{Recall} = \frac{TP}{N_+}$$

假阳率：预测结果将多少假的样本预测预测成了真

$$\text{FPR} = \frac{FP}{N_-}$$

ROC曲线

➤ 两类分类任务的评价指标

- Matthews相关性系数用一个值综合混淆矩阵，度量真实值和预测值之间的相关性，定义为

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- 分母中任意一对括号相加之和如果为0，那么整个MCC的值就为0。
- MCC值在 $[-1, 1]$ 之间
 - 1: 完美分类器
 - 0: 随机分类器
 - -1: 分类器是最差，所有预测结果和实际相反

真值

预测值

y	$\hat{y} = 1$	$\hat{y} = 0$	Σ
y = 1	TP	FN	N_+
y = 0	FP	TN	N_-
Σ	\hat{N}_+	\hat{N}_-	

➤ 例：垃圾邮件分类

- 假如我们有100封测试邮件，其中50封为垃圾50封为非垃圾，某分类器的预测结果：TP=40封，TN=40封，FP = 10封，FN = 10封

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} = 0.8$$

$$\text{Precision} = \frac{TP}{TP + FP} = 0.8, \quad \text{Recall} = \frac{TP}{TP + FN} = 0.8$$

$$F1 = \frac{2(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} = 0.8$$

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} = 0.6$$

➤ 例：垃圾邮件分类

- 还是100封测试邮件，其中垃圾邮件为98封，非垃圾邮件为2封（分布不均衡）。
- 假设如果有一个傻傻的分类器，永远只做出垃圾邮件判断，那么 $TP=98$, $TN=0$, $FN=0$, $FP=2$

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} = 0.98$$

$$\text{Precision} = \frac{TP}{TP + FP} = 0.98, \quad \text{Recall} = \frac{TP}{TP + FN} = 1$$

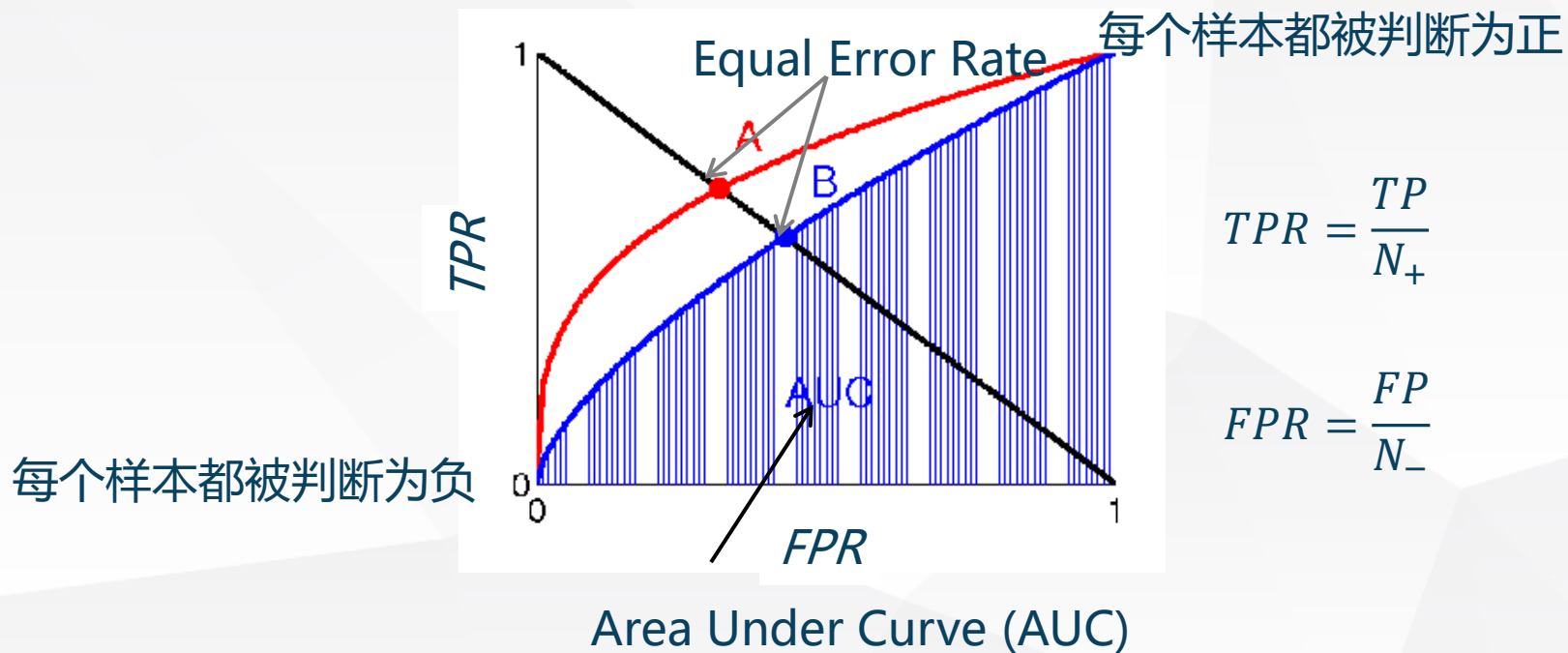
$$F1 = \frac{2(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} = 0.989$$

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} = 0$$

随机分类器

Receiver Operating Characteristic (ROC)曲线

- 上面我们讨论给定阈值 τ 的 TPR (真阳率) 和 FPR (假阳率)。
- 如果不是只考虑一个阈值, 而是在一些列阈值上运行检测器, 并画出 TPR 和 FPR 为阈值 τ 的隐式函数, 得到 ROC 曲线。

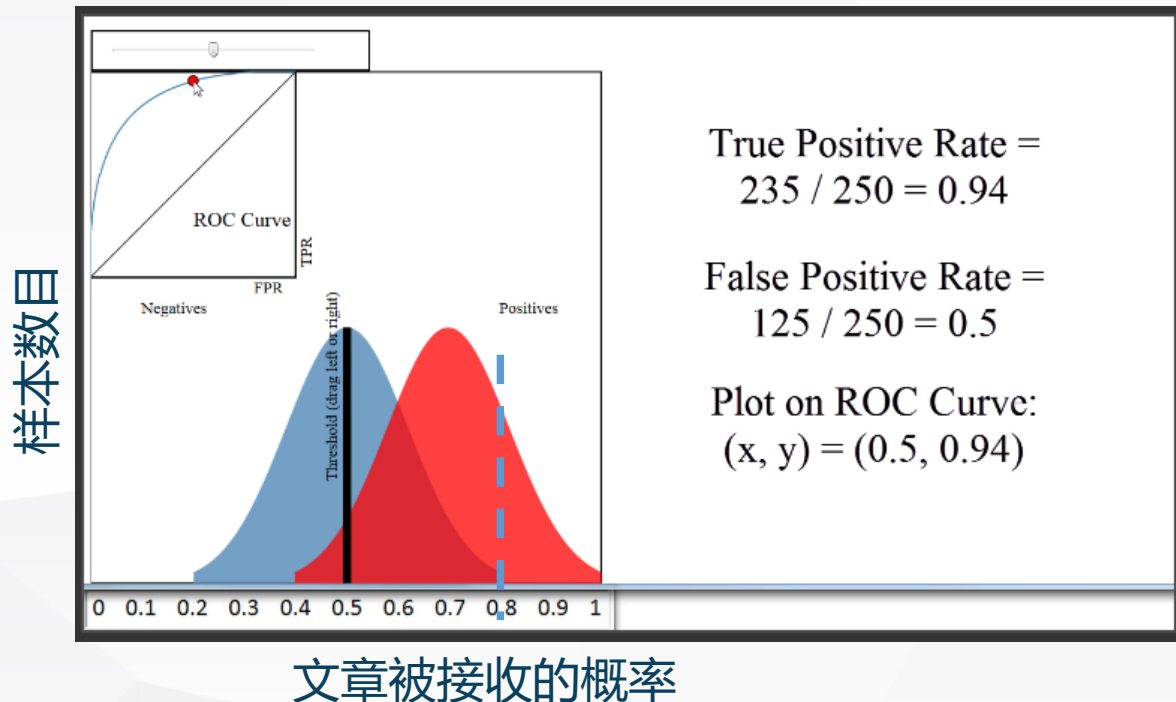


$$TPR = \frac{TP}{N_+}$$

$$FPR = \frac{FP}{N_-}$$

例：ROC曲线

- 例：根据文章特征 x （文章长度、作者的数目、作者之前投递给该杂志的文章数据、...），判断该文章是否会杂志被接收。
- 测试样本数目：500，其中250篇被接收（红色），250被拒绝（蓝色）。
- 现有一个分类器1，给定文章特征，输出该文章被接收的概率。下图为分类器1输出的被接收概率对应的正样本数目和负样本数目

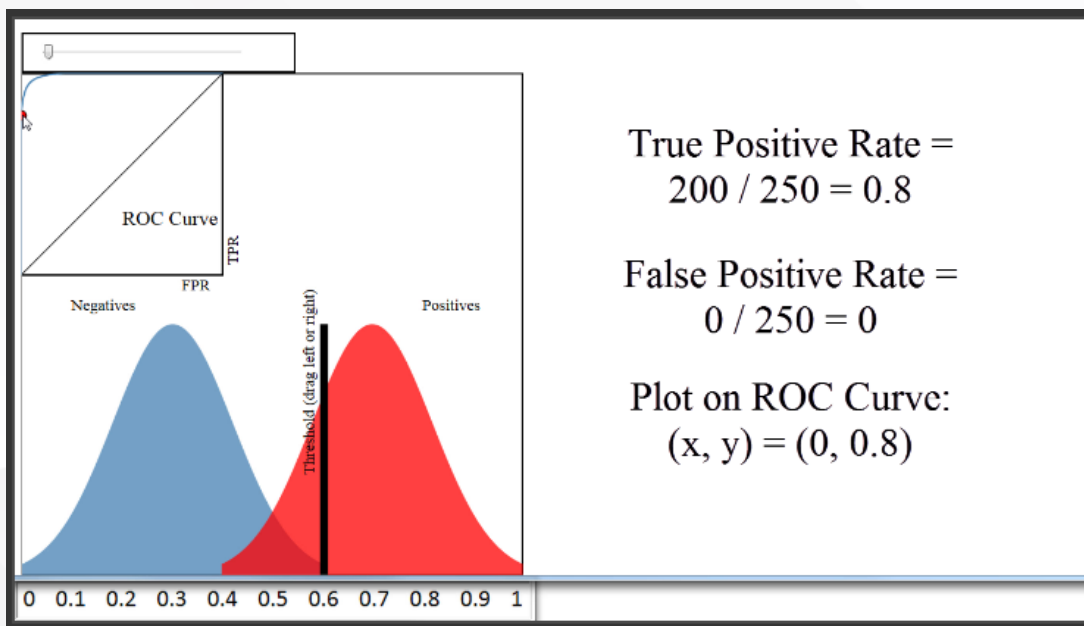


- 假设取阈值为概率阈值0.5: 判断140篇文章被拒绝, 360篇文章被接收。线右边共有235个红色样本被正确接收, 125个蓝色样本被错误接收。 $TPR = \frac{235}{250} = 0.94$, $FPR = \frac{125}{250} = 0.5$, 对应ROC曲线上的点 $(x, y) = (0.5, 0.94)$ 。
- 假设取阈值为概率阈值0.8: 判断50篇文章被接收, 450篇被拒绝。线的右边有50个红色被正确接收, 线的右边有0个蓝色样本被错误接收, $TPR = \frac{50}{250}$, $FPR = 0$, 对应ROC曲线上的点 $(x, y) = (0, 0.2)$ 。
- ...

例：ROC曲线

- 例：根据文章特征 x （文章长度、作者的数目、作者之前投递给该杂志的文章数据、...），判断该文章是否会杂志被接收。
- 测试样本数目：500，其中250篇被接收（红色），250被拒绝（蓝色）。
- 现有一个分类器2，给定文章特征，输出该文章被接收的概率。下图为分类器2输出的被接收概率对应的正样本数目和负样本数

样本数目



$$\text{True Positive Rate} = 200 / 250 = 0.8$$

$$\text{False Positive Rate} = 0 / 250 = 0$$

$$\text{Plot on ROC Curve: } (x, y) = (0, 0.8)$$

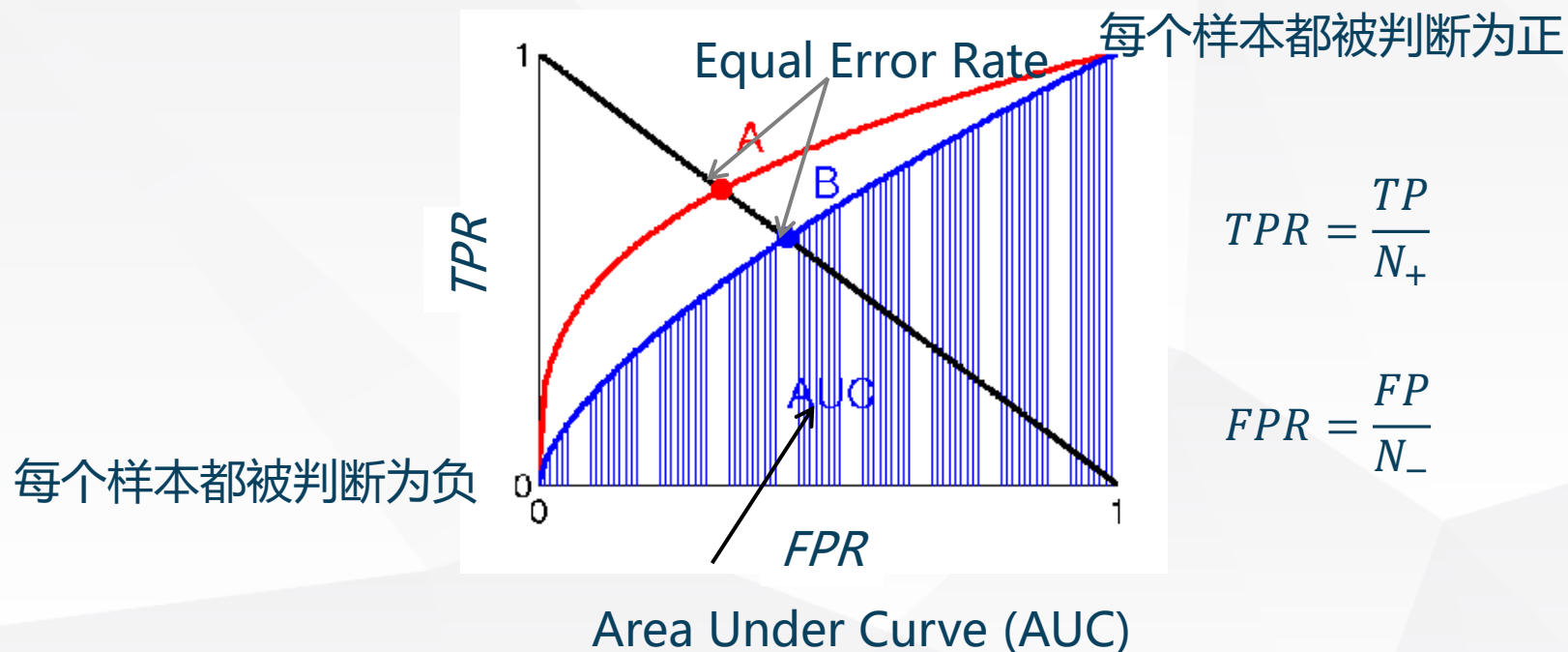
- 假设取阈值为概率阈值0.6: 线的右边有200个红色样本, 线的右边有0个蓝色样本, $\text{TPR} = \frac{200}{250} = 0.8$, $\text{FPR} = 0$, 对应ROC曲线上的点 $(x, y) = (0, 0.8)$ 。

• ...

文章被接收的概率

ROC曲线

- ROC曲线越偏左上角表示分类器性能越好
- AUC (Area Under Curve) : ROC曲线下的面积, 取值在 $[0.5, 1.0]$, 0.5表示随机猜测分类器, 1表示完美分类



$$TPR = \frac{TP}{N_+}$$

$$FPR = \frac{FP}{N_-}$$

ROC曲线有个很好的特性：当测试集中的正负样本的分布变化的时候，ROC曲线能够保持不变。在实际的数据集中经常会出现类不平衡现象，即负样本比正样本多很多（或者相反），而且测试数据中的正负样本的分布也可能随着时间变化。



- PR曲线
 - Precision and Recall (PR曲线): 用于稀有事件检测, 如目标检测、信息检索、推荐系统。

- 负样本非常多 (N_- 很大), 因此 $FPR = FP/N_-$ 很小, 比较TPR和FPR没有太大意义 (ROC曲线中只有左边很小一部分有意义) → 只讨论正样本

- Precision (精度, 查准率): 预测结果为真的样本中真正为真的比例

- Recall (召回率, 查全率): 预测结果召回了多少真正的真样本

➤ Precision and Recall (PR)曲线

■ PR曲线：阈值变化时的Precision和Recall

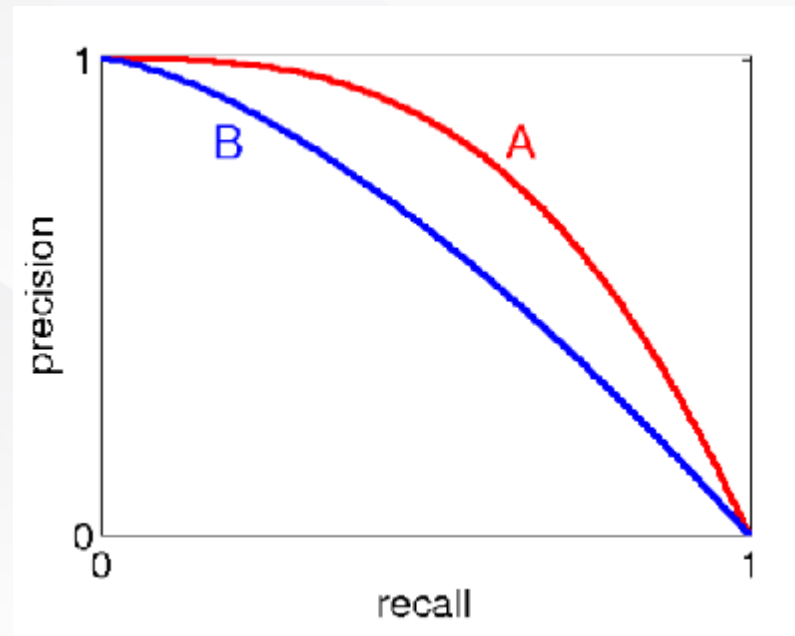
Precision: the fraction of our detection are actually positive
Recall: the fraction of the positives we actually detected

Precision = $\frac{TP}{\hat{N}_+}$: 检测结果真正为正的比率

Recall = $\frac{TP}{N_+}$: 被正确检测到的正样本的比率

F1分数: PR 合一:

$$F1 = \frac{2(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})}$$



注意：准确率和召回率是互相影响的。一般情况下准确率高、召回率低；召回率低、准确率高。

➤ AP: Average Precision

- Precision只考虑了返回结果中相关文档的数目，没有考虑文档之间的顺序。
- 对一个搜索引擎或推荐系统而言，返回的结果是有序的，且越相关的文档越靠前越好，于是有了AP（Average Precision）的概念。
- AP: 对不同召回率点上的精度进行平均

$$AP = \int_0^1 P(R) dR = \sum_{k=0}^n P(k) \Delta R(k)$$

- 即PR曲线下的面积（Recall: AUC为ROC下的面积）。

mAP: Mean AP

- 平均AP (Mean Average Percision, mAP) : 多个AP的平均。
- 如: 物体检测中经常用mAP评价模型性能: 多个物体类别的AP的平均

➤ Scikit-Learn中分类模型性能评价

- 类似回归模型的性能评价，Scikit-Learn中对分类模型性能评价提供3 不同的 API
 - estimator的score方法：每个学习器都有score方法，提供一个缺省的评估方法（分类为**正确率**）
 - Scoring参数：使用交叉验证评估模型的工具具有Scoring 参数
 - Metric： metrics模块实现了一些函数，用来评估预测误差

➤ 多分类与多标签

- 多类 (Multiclass) 分类: 相对于两类 (binary) 分类而言, 标签 y 的取值有多种, 如对水果进行分类, 水果可能是橙子、苹果或梨。多类分类的假设是, 每个样本被分配到一个和唯一——一个标签: 一个水果可以是苹果或梨, 但不是同时两个。
- 多标签 (Multilabel) 分类: 给每个样本分配一套目标标签, 这些标签不相互排斥 (类似物体属性, 可以有多种属性)。例如与文档相关的主题, 文本可能同时是关于宗教、政治、金融或教育的, 或着不属于这些主题。

➤ Scikit-Learn中metrics的分类模型评价指标

■ 只能用于两类分类

[`precision_recall_curve`](#)(y_true, probas_pred) 计算不同概率阈值对应的precision-recall对

[`roc_curve`](#)(y_true, y_score[, pos_label, ...]) 计算收器操作特性(ROC)曲线

■ 亦可用于多类分类

<code>cohen_kappa_score</code> (y1, y2[, labels, weights, ...])	Kappa分数 (Cohen 's kappa) : 衡量两种标注结果的吻合程度
<code>confusion_matrix</code> (y_true, y_pred[, labels, ...])	计算混淆矩阵
<code>hinge_loss</code> (y_true, pred_decision[, labels, ...])	平均合页损失 (非正则的)
<code>matthews_corrcoef</code> (y_true, y_pred[, ...])	计算Matthews相关系数 (MCC)



■ 可用于多标签

[`accuracy_score`](#)(y_true, y_pred[, normalize, ...])

正确率

[`classification_report`](#)(y_true, y_pred[, ...])

创建一个包含主要分类评价指标的文本报告

[`f1_score`](#)(y_true, y_pred[, labels, ...])

F1分数

[`fbeta_score`](#)(y_true, y_pred, beta[, labels, ...])

F-beta 分数

$$F_{\beta} = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{\beta^2 \text{precision} + \text{recall}}$$

[`hamming_loss`](#)(y_true, y_pred[, labels, ...])

平均Hamming损失

[`jaccard_similarity_score`](#)(y_true, y_pred[, ...])

Jaccard相似度分数

[`log_loss`](#)(y_true, y_pred[, eps, normalize, ...])

Logloss, logistic损失或交叉熵损失

[`precision_recall_fscore_support`](#)(y_true, y_pred)

计算每个类的precision, recall, F-measure and support

[`precision_score`](#)(y_true, y_pred[, labels, ...])

precision

[`recall_score`](#)(y_true, y_pred[, labels, ...])

recall

[`zero_one_loss`](#)(y_true, y_pred[, normalize, ...])

0/1分类损失



■ 可用于两类和多标签场合（但不能用于多类场合）：

[`average_precision_score`](#)(y_true, y_score[, ...]) 计算平均精度 (AP)

[`roc_auc_score`](#)(y_true, y_score[, average, ...]) 计算ROC曲线下的面积 (AUC)

■ 更多请见Scikit-Learn文档

- http://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics

➤ 从两类分类到多类分类和多标签

- 好些评价指标（如 [f1_score](#), [roc_auc_score](#)）都是用于二分类任务。
- 将两类分类指标扩展到多类或多标签问题：在多类或多标签问题中，可将其视为多个两类分类问题的集合，每个类别一个
- 采用不同的平均方式（通过average参数指定），可将多个两类分类指标合并得到多类或多标签问题相应的评价指标：
 - 宏观 “macro”：每个类别（二分类）评价指标，再求平均，每个类别的权重相同，会放大少数类（样本数目较少的类的影响）
 - 微观 “micro”：对每个样本（不分类别）计算全局的评价指标，每个样本的权重相同。多标签任务中首选，也可用于多类分类
 - 加权 “weighted”：计算每个类别的指标，每类的权重与该类样本数目有关，可处理不同类别样本数目不均衡问题
 - 样本 “samples”：计算每个样本的评价指标，然后再求平均，仅适用于多标签问题
 - average=None：返回一个数组，包含每个类的分数

Scikit-Learn中的Scoring参数

Scoring	Function	Comment
Classification		
'accuracy'	metrics.accuracy_score	
'average_precision'	metrics.average_precision_score	
'f1'	metrics.f1_score	for binary targets
'f1_micro'	metrics.f1_score	micro-averaged
'f1_macro'	metrics.f1_score	macro-averaged
'f1_weighted'	metrics.f1_score	weighted average
'f1_samples'	metrics.f1_score	by multilabel sample
'neg_log_loss'	metrics.log_loss	requires predict_proba support
'precision' etc.	metrics.precision_score	suffixes apply as with 'f1'
'recall' etc.	metrics.recall_score	suffixes apply as with 'f1'
'roc_auc'	metrics.roc_auc_score	

注意：scoring越大的模型性能越好，所以如果采用损失 / 误差，需要加neg，如 'neg_log_loss'

Outline

- 分类任务
- Logistic回归
- Logistic回归的损失函数
- Logistic回归的优化算法
- Logistic回归的多类分类任务
- 分类任务中的样本不均衡问题
- 分类模型的评价指标
- Sklearn中的Logistic回归API
- Logistic回归案例分析

➤ Scikit-Learn中Logistic回归分类器

- LogisticRegression: 给定正则参数C对应的Logistic回归
 - 可以和GridSearchCV结合进行正则超参数调优
- LogisticRegressionCV: 在一组正则参数Cs中寻找最佳C的Logistic回归
- SGDClassifier: 可实现采用随机梯度下降优化的Logistic回归

LogisticRegression

```
class sklearn.linear_model.LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)
```

- 支持两种正则：L2正则和L1正则。但L1正则函数不连续，优化器只能选择 'liblinear' 和 'saga'。正则参数为：*penalty*、*C*
- 支持两种多类分类方式：'ovr' 和 'multinomial'。模型参数为*multi_class*。
- 支持多种优化方式，需注意不同优化方法的适用范围。优化器参数为：*solver*
- 支持对不同类别样本权重设置，权重参数：*class_weight*

>> Scikit-Learn中实现Logistic回归的类

■ 参数solver表示优化算法，可选

- a) liblinear: 使用开源的liblinear库实现，内部使用了坐标轴下降法来迭代优化损失函数，支持L1正则和L2正则。
- b) lbfgs: 拟牛顿法，只支持L2正则。
- c) newton-cg: 牛顿法（每个大迭代中的加权最小二乘回归部分采用共轭梯度算法实现），只支持L2正则。
- d) sag: 随机平均梯度下降，梯度下降法的变种，适合于样本数据多（如大于5万）的时候，只支持L2正则。
- f) saga: 改进的随机平均梯度下降，支持L1正则。

➤ Scikit-Learn中实现Logistic回归的类

正则	求解算法	应用场景
L1	liblinear	如果模型的特征非常多，希望一些不重要的特征系数归零，从而让模型系数稀疏的话，可以使用L1正则化。liblinear适用于小数据集。
L1	saga	当数据量较大，且选择L1，只能采用saga
L2	liblinear	liblinear只支持多类Logistic回归的OvR，不支持多项分布损失（MvM），但MvM相对精确。
L2	lbfgs/newton-cg/sag	较大数据集，支持OvR和 MvM两种多类Logistic回归。
L2	sag / saga	如果样本量非常大，支持OvR和 MvM两种多类Logistic回归。

对于大数据集，可以考虑使用SGDClassifier，并使用logloss。（类似回归中的SGDRegressor）

sag/saga只在特征尺度大致相等时才能保证收敛，需对数据做缩放（如标准化）

LogisticRegression的参数

参数	说明	备注
penalty	惩罚函数 / 正则函数，支持L2正则和L1正则，缺省：L2	L1正则的优化器可选 'liblinear' 和 'saga'
dual	原问题 (primal) 还是对偶问题求解。对偶只支持L2正则和 'liblinear' 优化器。缺省：False	当样本数n_samples>特征数目n_features时，原问题求解更简单。
tol	迭代终止判据的误差范围。缺省:1e-4	
C	交叉熵损失函数系数，缺省：1	$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^n \log(\exp(-y_i(X_i^T w + c)) + 1).$
fit_intercept	是否在决策函数中加入截距项。缺省：True	如果数据已经中心化，可以不用。
intercept_scaling	截距缩放因子，当fit_intercept为True且liblinear solver有效。输入为[x, self.intercept_scaling]，即对输入特征加入1维常数项	增加的常数项系数也受到1/l2 正则的惩罚，所以要适当增大常数项。
class_weight	不同类别样本的权重，用户指定每类样本权重或 'balanced'（每类样本权重与该类样本出现比例成反比）。缺省：None	
random_state	数据洗牌时的伪随机数。缺省：None	如希望每次运行结果相同，设置random_state为一整数。

>> LogisticRegression的参数

参数	说明	备注
solver	优化求解算法, 可为 'newton-cg' , 'lbfgs' , 'liblinear' , 'sag' , 'saga' 。缺省: liblinear	L1正则的优化器可选 'liblinear' 和 'saga'
max_iter	最大迭代次数, 当solver为newton-cg, sag, lbfgs 时有效。缺省: 100	
multi_class	多类分类处理策略, 可为 'ovr' , 'multinomial' 。 'ovr' 为1对多, 将多类分类转化为多个两类分类问题, multinomial为softmax分类。缺省: 'ovr'	multinomial的优化器只支持newton-cg, lbfgs和sag。OvR相对简单, 但分类效果相对略差。MvM分类相对精确, 但是分类速度没有OvR快。
verbose	是否详细输出	
warm_start	是否热启动 (用之前的结果作为初始化) , 对 liblinear solver无效。缺省: False	
n_jobs	多线程控制, 缺省值1。取-1时算法自动检测可用CPU核, 并使用全部核	

LogisticRegression的属性

属性	说明	备注
coef_	回归系数/权重，与特征维数相同。	如果是多任务回归，标签 y 为二维数组，则回归系数也是二维数组。
intercept_	截距项。	
n_iter_	每个类的迭代次数	

LogisticRegression的方法

方法	说明
<code>fit(X, y[, sample_weight])</code>	模型训练。参数X,y为训练数据，也可以通过sample_weight设置每个样本的权重。
<code>predict(X)</code>	返回X对应的预测值（类别标签）
<code>_predict_log_proba(X)</code>	返回X对应的预测值（每个类别对应的概率的log值）
<code>_predict_proba(X)</code>	返回X对应的预测值（每个类别对应的概率）
<code>decision_function(X)</code>	返回X对应的预测值（置信值，样本到决策超平面的带符号距离）
<code>score(X, y[, sample_weight])</code>	评估模型预测性能，返回模型预测的正确率。
<code>decision_function(X)</code>	预测的置信度（样本到分类超平面的带符号距离）
<code>densify()</code>	如果之前将系数矩阵变成了稀疏模式，再将其变回稠密模式（fit函数的格式）
<code>sparsify()</code>	将系数矩阵变成了稀疏模式

```
class sklearn.linear_model.LogisticRegressionCV(Cs=10, fit_intercept=True, cv=None, dual=False, penalty='l2',  
scoring=None, solver='lbfgs', tol=0.0001, max_iter=100, class_weight=None, n_jobs=1, verbose=0, refit=True, i  
ntercept_scaling=1.0, multi_class='ovr', random_state=None)
```

■ 在一组正则参数 Cs 中找最佳的 C

■ 额外与CV有关的参数

- Cs : 在 $[10^{-4}, 10^4]$ 区间的log域均匀取 Cs 个值作为正则参数 C 的搜索空间
- cv : 交叉校验划分策略
- 注意:
 - 默认的solver变成了 'lbfgs'
 - 对多类分类问题, 采用的是 'ovr' 的方式, 用交叉验证得到每个类的最佳正则参数。

- 类似SGDRegressor, Scikit-Learn中的实现了随机梯度下降分类: SGDClassifier。
- SGDClassifier对于大数据量训练集 (样本数 $N > 10^5$, 特征数目 $N > 10^5$) 问题合适。

```
class sklearn.linear_model.SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=None, tol=None, shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5, class_weight=None, warm_start=False, average=False, n_iter=None))
```

■ 支持的损失函数`loss`包括

- `hinge` : 合页损失(SVM)
- `llog`: 负log似然损失 (Logistic回归)
- `modified_huber`: 对噪声不损失
- `squared_hinge`
- `perceptron`
- 回归中用到的loss: `squared_loss`, `huber`, `epsilon_insensitive`, or `squared_epsilon_insensitive`

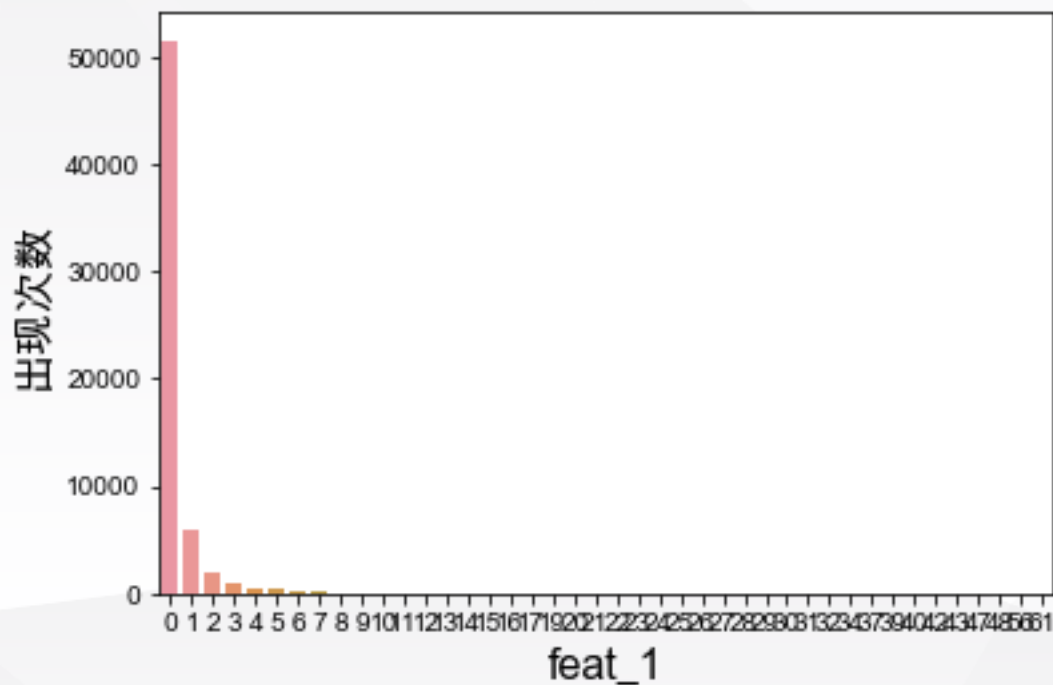
参数`epsilon`是某些损失函数 (`huber`、`epsilon_insensitive`、`squared_epsilon_insensitive`) 需要的额外参数。

Outline

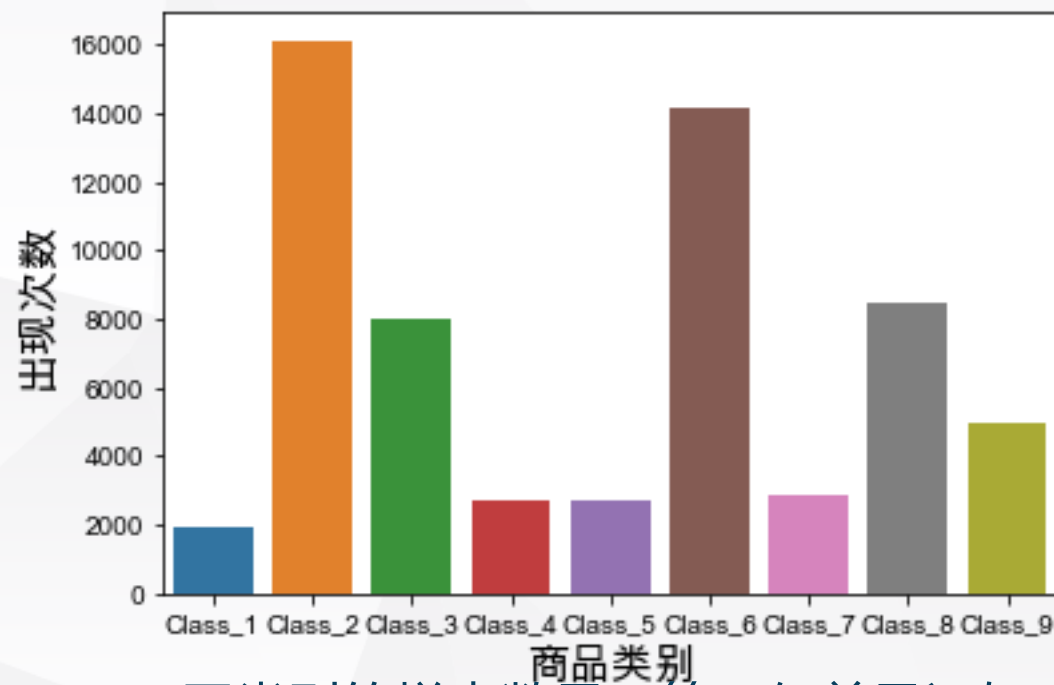
- 分类任务
- Logistic回归
- Logistic回归的损失函数
- Logistic回归的优化算法
- Logistic回归的多类分类任务
- 分类任务中的样本不均衡问题
- 分类模型的评价指标
- Sklearn中的Logistic回归API
- Logistic回归案例分析

➤ Otto商品分类——数据分析

- 训练集 61,878个样本，测试集144,368个样本
 - 每个样本有93个数值型特征： 表示某种事件发生的次数
 - 标签： 9种商品类别



特征稀疏：大部分样本的特征值为0



不同类别的样本数目不等，但差异还好
(不少于0.1倍)

■ 特征均为数值型特征

- 计数：可以考虑log变换
- TF-IDF：事件计数可类比文档分类任务中的词频特征，可进一步更有判别力的 (Term Frequency–Inverse Document Frequency, TF-IDF)
- 标准化处理

➤ Otto商品分类——模型选择

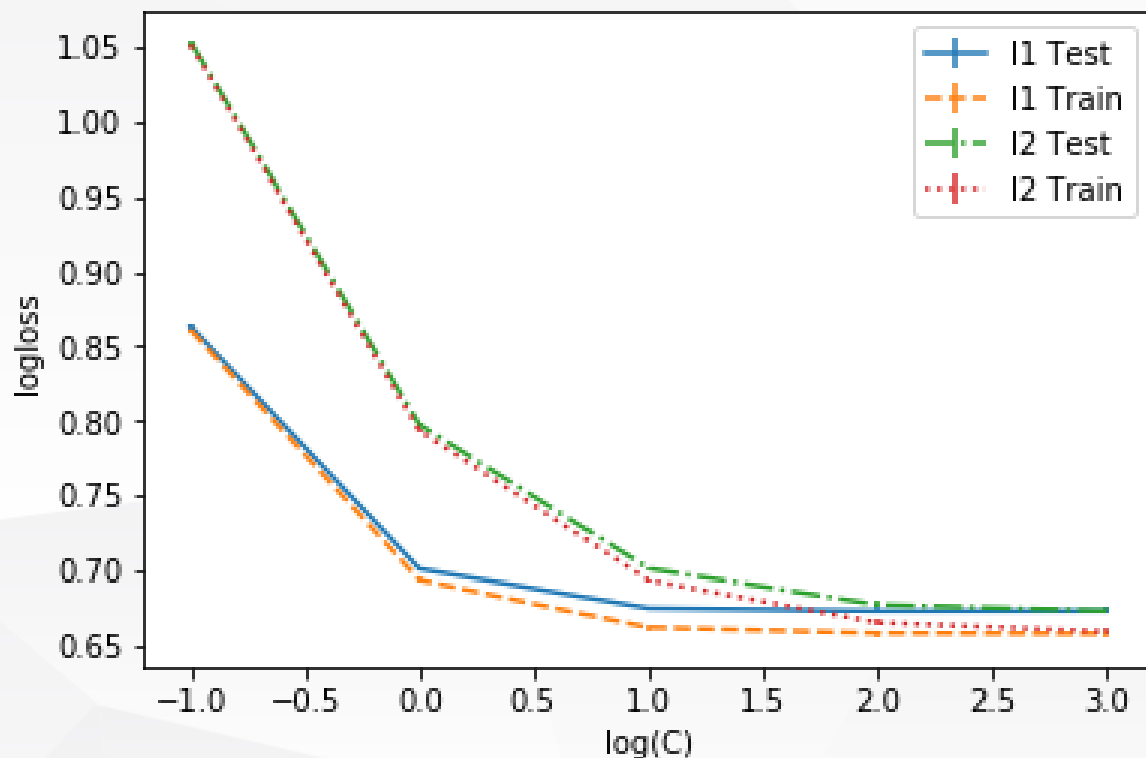
- 正则函数：L1正则、L2正则

- 正则超参数 C

- GridSearchCV (3折交叉验证)

- 评价指标：logloss

3_Train_Test_Advertising.ipynb



可以看出在训练集上 C 越大（正则越少）的模型性能越好；
但在测试集上（交叉验证估计）当 $C = 100$ 时性能最好（L1正则）

■ 测试数据上的性能

特征编码方案	分类性能 (Logloss)
原始特征	0.66683
log特征编码	0.67317
TF-IDF特征	0.63319

Logistic回归在该任务上的性能并不好，可能是该任务比较复杂，Logistic回归只是一个线性模型，不足以处理这么复杂的任务

➤ 本章总结

■ Logistic回归

- 函数集合：输入特征的线性组合
- 目标函数
 损失函数：交叉熵损失
 正则项：L2正则、L1正则
- 优化方法：梯度下降、牛顿法

■ Sklearn中的线性回归实现

- LogisticRegression/ LogisticRegressionCV
- SGDClassification

■ 分类模型评价指标

■ 类别不均衡的分类