逻辑回归与线性回归：

方法	自变量（特征）	因变量（结果）	关系
线性回归	连续或离散	连续实数	线性
Logistic回归	连续或离散	(0,1)之间连续值	非线性

线性回归用一组变量的（特征）的线性组合，来建立与结果之间的关系；在线性回归模型中，输出一般是连续的， 对于每一个输入的x，都有一个对应的输出y。因此模型的定义域和值域都可以是无穷。模型表达y(x,w)=w0+w1x1+...+wnxn；

逻辑回归用于分类，而不是回归；对于逻辑回归，输入可以是连续的[-∞，+∞]，但输出一般是离散的，通常只有两个值{0， 1}；这两个值可以表示对样本的某种分类，高/低、患病/健康、阴性/阳性等，这就是最常见的二分类逻辑回归。因此，从整体上来说，通过逻辑回归模型，我们将在整个实数范围上的x映射到了有限个点上，这样就实现了对x的分类；

逻辑回归与线性回归的关系可以认为逻辑回归的输入是线性回归的输出，将逻辑斯蒂函数（Sigmoid曲线）作用于线性回归的输出得到输出结果；线性回归y = ax + b，其中a和b是待求参数；逻辑回归p = S(ax + b)，其中a和b是待求参数， S是逻辑斯蒂函数，然后根据p与1-p的大小确定输出的值，通常阈值取0.5，若p大于0.5则归为1这类；

线性函数如下：

$$\theta_0+\theta_1x_1+,\ldots,+\theta_nx_n=\sum_{i=1}^{n}\theta_ix_i=\theta^Tx$$

构造预测函数：

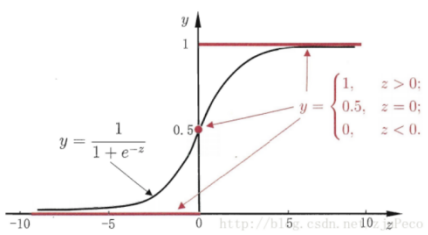
$$h_\theta(x)=g(\theta^Tx)=\frac{1}{1+e^{-\theta^Tx}}$$

逻辑回归的原理：

逻辑回归是利用回归类似的方法来解决分类问题。假设有一个二分类问题，输出y{0,1}，而线性模型的预测值z是实数值，我们希望找到一个阶跃函数将实数z映射为{0,1}，这样我们就能很好的处理分类问题了。sigmoid函数中的z就是线性函数的z，因为g(z)最后输出的时样本类别的概率值，所以我们可以把阈值设为0.5，g(z)大于等于0.5的看作1，小于0.5的看作0，这样我们就能利用逻辑回归来处理二分类问题了。

sigmoid函数的图像：

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



逻辑回归用途：

寻找危险因素：寻找某一疾病的危险因素等；

预测：根据模型，预测在不同的自变量情况下，发生某病或某种情况的概率有多大；
判别：实际上跟预测有些类似，也是根据模型，判断某人属于某病或属于某种情况的概率有多大，也就是看一下这个人有多大的可能性是属于某病。

逻辑回归损失函数推导及优化：

Cost函数和J函数如下，它们是基于最大似然估计推导得到的。

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

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}Cost(h_\theta(x_i),y_i)=-\frac{1}{m}\left[\sum_{i=1}^{m}(y_i log h_\theta(x_i)+(1-y_i)\log(1-h_\theta(x_i)))\right]$$

1） 求代价函数

概率综合起来写成：

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

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：

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

最大似然估计就是求使l(θ)取最大值时的θ，其实这里可以使用梯度上升法求解，求得的θ就是要求的最佳参数。

在Andrew Ng的课程中将J(θ)取为下式，即：

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

2) 梯度下降法求解最小值

θ更新过程：

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

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

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

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

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y_i(1-g(\theta^T x_i)) - (1-y_i) g(\theta^T x_i) \right) x_i^j$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y_i - g(\theta^T x_i) \right) x_i^j$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i) - y_i \right) x_i^j$$

θ更新过程可以写成：

$$\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i) - y_i \right) x_i^j$$

正则化：

正则化是一种处理过拟合问题的有效手段；正则化是结构风险最小化策略的实现，是在经验风险上加一个正则化项或惩罚项。正则化项一般是模型复杂度的单调递增函数，模型越复杂，正则化项就越大。正则项可以取不同的形式，在回归问题中取平方损失，就是参数的L2范数，也可以取L1范数。取平方损失时，模型的损失函数变为：

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{n}(h_\theta(x_i) - y_i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2$$

lambda是正则项系数：

值很大，说明对模型的复杂度惩罚大，对拟合数据的损失惩罚小，这样它就不会过分拟合数据，在训练数据上的偏差较大，在未知数据上的方差较小，但是可能出现欠拟合的现象；

值很小，说明比较注重对训练数据的拟合，在训练数据上的偏差会小，可能会导致过拟合；

正则化后的梯度下降算法θ的更新变为：

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta(x_i) - y_i \right) x_i^j - \frac{\lambda}{m}\theta_j$$

L0正则化的值是模型参数中非零参数的个数。

L1正则化表示各个参数绝对值之和。

L2正则化标识各个参数的平方的和的开方值。

L1正则假设参数的先验分布是Laplace分布，可以保证模型的稀疏性，也就是某些参数等于0；
L2正则假设参数的先验分布是Gaussian分布，可以保证模型的稳定性，也就是参数的值不会太大或太小；
在实际应用过程中，L1会趋向于产生少量的特征，而其他的特征都是0，而L2会选择更多的特征，这些特征都会接近于0。Lasso在特征选择时候非常有用，而Ridge就只是一种规则化而已。在所有特征中只有少数特征起重要作用的情况下，选择Lasso比较合适，因为它能自动选择特征。而如果所有特征中，大部分特征都能起作用，而且起的作用很平均，那么使用Ridge也许更合适。

模型评估标准：

分类问题有通用的评估指标，常用的是准确率(accuary)、召回率(recall)、精度(precision)、ROC曲线、AUC。

逻辑回归的优缺点：

优点：
1）速度快，适合二分类问题
2）简单易于理解，直接看到各个特征的权重
3）能容易地更新模型吸收新的数据
缺点：
1.对数据和场景的适应能力有局限性，不如决策树算法适应性那么强
2.容易欠拟合，分类精度不高
3.数据特征有缺失或者特征空间很大时表现效果并不好

样本不均衡问题解决办法：

样本不均衡指的是数据集中正负例样本比例失衡，不再是1:1。此类问题的解决方法一般是基于数据集的重采样或者基于模型的调整。对于逻辑回归来说可以调整预测函数的临界值，使其适当偏向少数类样本，平衡召回率和精度。

sklearn参数：

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)

api	参数	意义	备注
LogisticRegression的parameters	penalty	正则化项，可选值为'L1'和'L2'，默认为'L2'	解决过拟合：'L2'；特征多希望特征归0：'L1'。 'L2'可选优化算法：'newton-cg'；'lbfgs'；'liblinear'；'sag'； 'L1'可选优化算法：'liblinear'(L1正则化的损失函数不是连续可导的)
	dual	对偶或原始方法，可选值True和False，默认False	dual只适用于正则化项为L2并且损失函数优化算法为liblinear的情况； 当样本数>特征数时默认为False
	tol	优化算法停止的条件，默认1e-4	迭代前后函数差值<=0.0001时停止
	C	正则化系数lambda的倒数，默认为1.0	C越小表示惩罚越大（正则化越强）
	fit_intercept	True和False：是否存在截距项；默认True存在	是否含有b，不含b，则b=0
	intercept_scaling	默认为1；	x=(x, intercept_scaling)，等价于西瓜书x=(x, 1)； 仅当solver='liblinear'，fit_intercept=True时有效
	class_weight	类型权重，不平衡数据/分类代价敏感；dict or 'balanced'；默认为None	解决类别不平衡问题，权重值可以事先计算好再赋值，也可以设置为balanced,即让程序自动根据数据集计算出每个类别对应的权重。 eg：{0:0.8，1:0.2}；二分类，0类权重为0.8，1类权重为0.2
	random_state	随机数种子，仅在solver='sag'和'liblinear'时有用	变量初始值随机产生，种子控制产生什么值
	solver	损失函数优化算法；{'newton-cg', 'lbfgs', 'liblinear', 'sag'}，默认'liblinear	liblinear:坐标轴下降法； lbfgs:拟牛顿法，利用海森矩阵优化； newton-cg:牛顿法； sag:随机平均梯度下降，每次迭代仅用一部分数据计算，适用于样本数据多的时候；
	max_iter	优化算法的迭代次数；整数，默认100	仅在优化函数为newton-cg,lbfgs,sag时有用
	multi_class	选择多分类的策略	ovr和multinomial（manyVSmany）；默认ovr
	verbose	控制是否print训练过程	0：不输出训练过程；1：偶尔输出；>1：对每个子模型都输出
	warm_start	是否热启动，True和False，默认False	True：把之前训练好的参数值赋给变量进行初始化；
	n_jobs	用cpu的几个核来跑程序；默认1	-1: all cores are used