# 线性回归------预测连续数值型变量

## 简介

线性回归分析是一种用于预测连续数值型变量的统计学和机器学习方法.

它假设特征变量与目标变量之间存在线性关系, 基于此, 它通过一个线性方程(线性回归模型)描述特征变量与目标变量之间的关系:



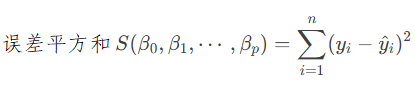
* y是因变量, x 是自变量
* *β*0​ 是截距，*β*1​ 是斜率
* *ϵ* 是误差项, 服从均值为0的正态分布。

## 参数估计

### 最小二乘法

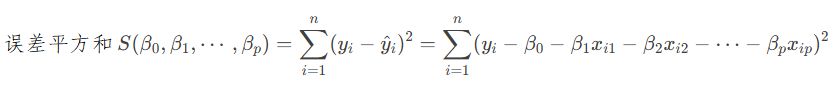
通常使用最小二乘法进行参数估计, 其基本思想是观测值与预测值之间的误差平⽅和越小, 模型对所有样本的预测越准确, 所以在给定观测数据的情况下，最小化样本的预测值与观测值之间的误差和S, 即可以找到使模型最符合观测数据的参数向量***β.***

预测值与观测值之间的误差和S：



### 损失函数

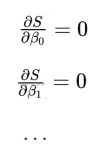
将描述特征变量与目标变量之间关系的线性方程带入最小二乘法误差平⽅和S公式:



两边同时除以n, 则得到线性回归损失函数（均方误差Mean Squared Error, MSE损失函数）.

### 参数估计公式---最小二乘估值公式

因为线性函数是一个连续可微的函数，假设变量之间没有约束条件，显然当且仅当每个参数的偏导数都为0时, 误差平方和S有最小值. 因此对每个参数求偏导数并令偏导数为0, 即可求解参数向量***β*** ( β₀,  β₁):



通过一系列矩阵运算和求导运算推导, 可以得出模型参数计算公式(最小二乘估值公式) :



* ***β*** 是包含β₀,  β₁ ..... β​*p*​ 的参数向量
* **X**是包含自变量观测值的设计矩阵（第一列通常为 1，对应截距项）
* **y**是目标变量的观测值向量

## 适用场景

适用于目标变量为连续数值型变量， 且特征向量与目标变量之间符合线性关系的场景。例如:

· **房价预测**（根据房屋面积、地理位置等特征预测房价）

· **销售预测**（根据广告投放量预测销售额）

· **股票价格预测**（基于历史数据预测未来价格）

· **医学研究**（如根据患者特征预测血压）

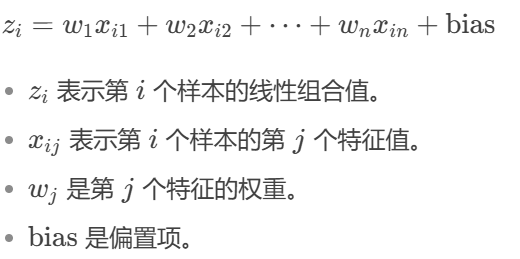
如果实际关系是非线性的，线性回归的效果可能会较差。因此，在使用线性回归分析时，通常需要验证特征向量与目标变量之间是否存在线性关系，或者通过特征工程将非线性关系转化为线性关系。

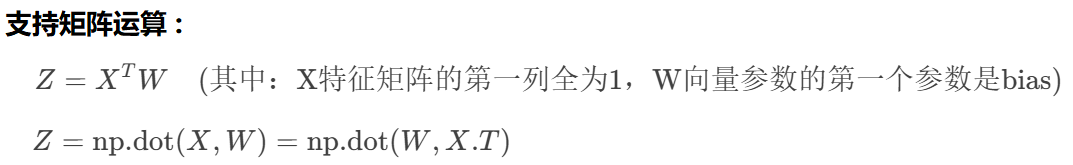
# 逻辑回归-----二分类问题

## 简介

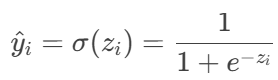
逻辑回归分析（Logistic Regression）是一种用于二分类（Binary Classification）问题的统计学和机器学习方法.

它假设特征变量与目标变量之间存在某种线性关系, 且每一个样本 i是正类的预测概率符合伯努利分布, 基于此, 它对样本的特征向量进行加权求和：

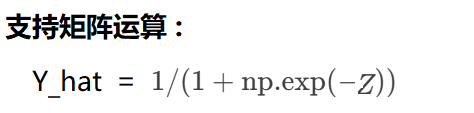




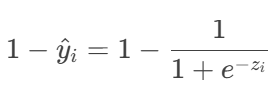
为了让输出是一个概率值（0到1之间），将 z通过 Sigmoid 函数进行变换并推导, 得到样本是正类的预测概率（逻辑回归概率模型）为:



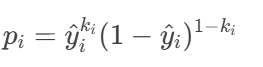
* y\_hat表示样本是正类的预测概率，它实际上是一个条件概率
* z是样本的特征向量的加权求和，其值越大，则样本是正类的预测概率也越大

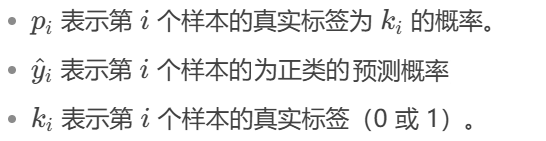


根据伯努利分布, 样本是负类的预测概率则为：



样本是正类或父类的预测概率可以合并为如下公式：





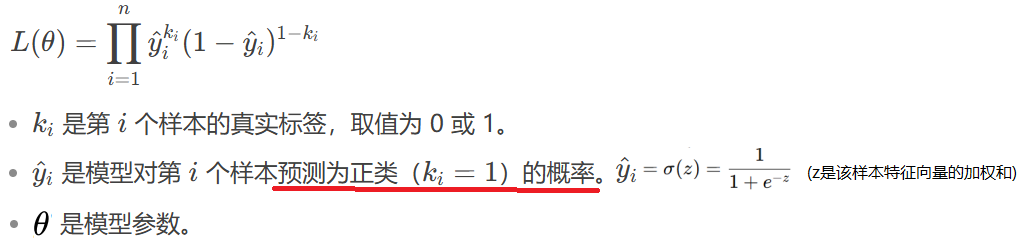
## 参数估计

### 最大似然估计 & 梯度下降

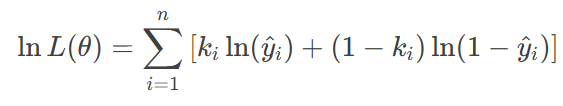
通常使用最大似然估计（Maximum Likelihood Estimation, MLE）进行参数估计，并通过梯度下降（Gradient Descent） 进行优化. 其基本思想是似然函数的值越大, 模型对所有样本的预测越准确. 所以在给定观测数据的情况下，最大化所有样本的预测概率的乘积(联合概率、似然函数 L(θ∣X))或者最小化其负值,  即可以找到使模型最符合观测数据的参数向量***β.***

### 损失函数

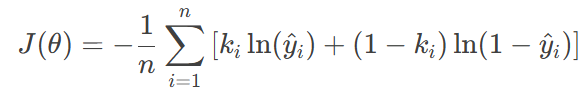
将样本是正类或父类的预测概率带入联合概率（似然函数 L(θ∣X)）公式得到：

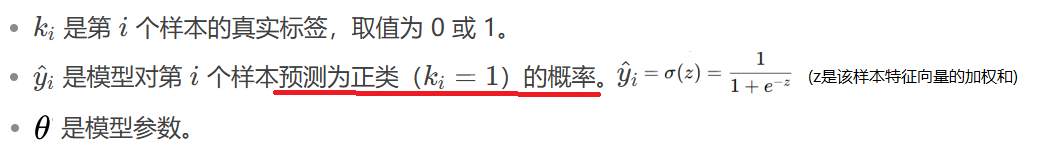


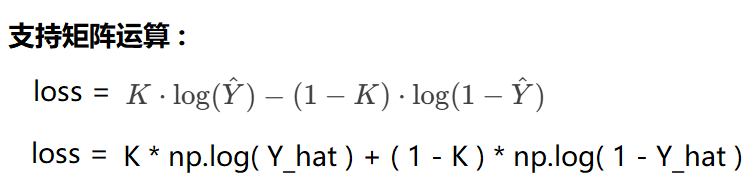
两边取对数, 将乘法变加法, 得到对数似然函数：



加负号, 将求最⼤转换为取最⼩, 然后除以n, 则得到逻辑回归损失函数：



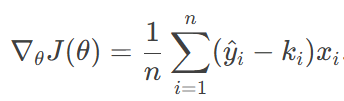


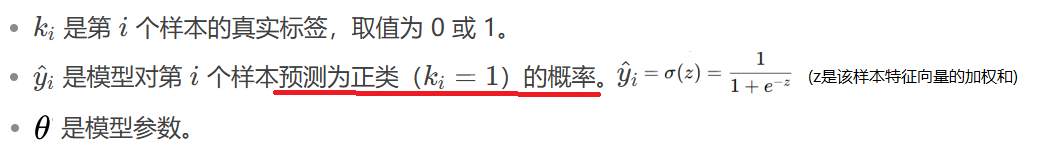


### 损失函数的梯度

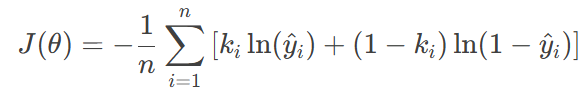
损失函数的梯度是损失函数中每个参数的偏导数构成的向量, 表示损失函数在当前参数值处的变化率.

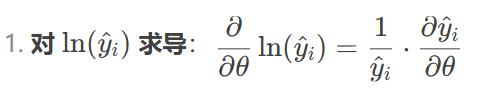
逻辑回归损失函数的梯度等于:

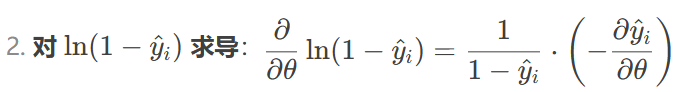


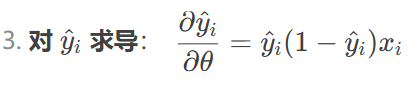


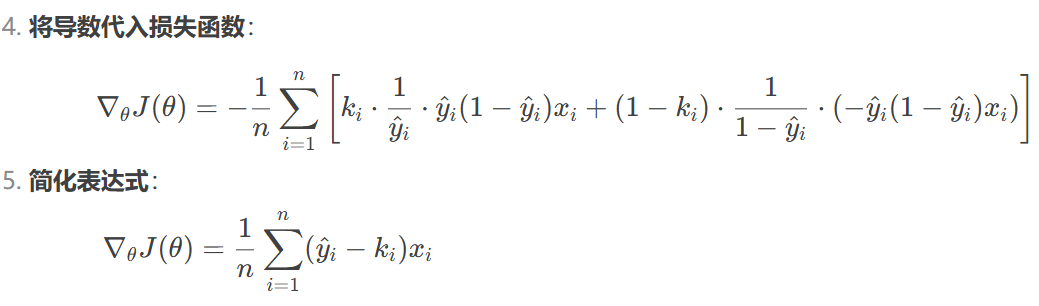
其推导过程如下:











### 参数估计步骤---梯度下降法

梯度表示函数在各个自变量方向上的变化率， 梯度越⼩，说明函数的变化率越⼩，趋近于0时，函数取极值（对于损失函数则是取最小值），因此通过梯度下降法反复迭代不断调整模型参数最小化损失函数的结果值，直至达到最大迭代次数或损失函数收敛， 即可求解模型参数θ , 具体步骤如下:

#### 准备数据集

# 准备数据集, 并在数据集的前面增加1列全1列

X, y = sklearn.datasets.make\_classification(sample\_num, feature\_num)

column\_ones = np.ones((X.shape[0], 1))         # 参数是一个元组，表示行和列

X = np.concatenate((column\_ones, X), axis=1)  # axis=1水平方向连接， axis=0 竖直...

# 拆分数据集: 训练数据集、测试数据集

X\_train, X\_test, y\_train, y\_test = sklearn.model\_selection.train\_test\_split(X, y, test\_size=test\_sample\_size)

#### 初始化模型参数

# 注意要包含偏置项bias

theta = np.random.randn(1, feature\_num + 1)

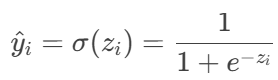
# theta = np.zeros(feature\_num + 1)

#### 模型训练---使用训练数据集

# 通过梯度下降法循环反复迭代不断调整模型参数

for i in range(max\_epoch):

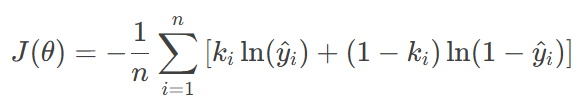
* 使用当前参数向量, 计算所有样本是正类的预测概率:



    z = np.dot(X, theta.T)

    y\_hat = sigmoid(z) = 1 / (1 + np.exp(-z))

* 将预测概率带入损失函数，计算损失值:



np.clip(y\_hat, 1e-15, 1 - 1e-15) #避免出现log(0)、log(1)情况

loss\_value = -1 \* np.mean( y \* np.log(y\_hat) + ( 1 - y ) \* np.log( 1 - y\_hat ) )

* 输出准确率

    if i % 100 == 0:

    # 将概率转换为类别标签: 将大于等于 0.5 的值转换为 1，小于 0.5 的值转换为 0

    y\_pred = (y\_hat > 0.5).astype(int)

        correct\_pred\_num = np.sum(y\_pred == y)  # [False,True,...,False] -> [0,1,...,0]

accuracy = correct\_pred\_num / len(y)

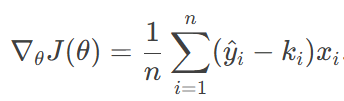
        print(f"epoch: {i}, loss: {loss\_value}, acc: {accuracy}")

* 如果损失函数已收敛，则结束循环；

if abs(last\_loss\_value - loss\_value) < convergence\_threshold:

        break

* 计算损失函数 J(θ)在当前参数 θ处的梯度(变化率):



n = X.shape[0]

gradient = np.dot((y\_hat - y), X) / n

* 根据当前梯度值调整模型参数:

theta = theta - learn\_step \* gradient # 一般learn\_step取值1e-2 ~ 1e-3

#### 模型验证---使用测试数据集

    # 使用模型参数，计算测试集的预测概率

    z = np.dot(X, theta.T)

    y\_hat = sigmoid(z) = 1 / (1 + np.exp(-z))

    # 将概率转换为类别标签: 将大于等于 0.5 的值转换为 1，小于 0.5 的值转换为 0

    y\_pred = (y\_hat > 0.5).astype(int)

# 计算准确率

correct\_pred\_num = np.sum(y == y\_pred)

    accuracy = correct\_pred\_num / len(y)

print("Validation Accuracy:", accuracy)

#### 保存模型参数

np.save('theta.npy', theta)

## 适用场景

适用于二分类问题. 每一个样本 i的预测概率符合伯努利分布， 且特征向量与目标变量之间存在某种线性关系的场景。 如果关系是非线性的，逻辑回归的表现可能会受到限制，需要借助其他方法或扩展模型来处理。

# 重要概念

## 截距 & 偏差（***bias***）

是线性回归模型中的常数项，截距的值反映了除⾃变量之外的其他所有因素对目标变量的综合影响（当 x = 0 时）。在机器 学习领域也被称为偏差（bias）。

## 参数估计

参数估计是机器学习的关键或目标, 其目的是找到一组参数使得模型最符合观测数据(拟合数据)的同时，能够平衡拟合能力与泛化能力，并满足特定的统计性质或应用需求。具体包括：

* 最小化预测误差；在大多数情况下，如上下文是简单的线性回归或预测任务这种简单场景, 参数估计的目标是找到一组参数来拟合数据, 使得预测值与真实值之间的误差尽可能小。通常是找到模型的损失函数并最小化损失函数, 并基于观测数据求解参数向量.
* 控制模型复杂度；
* 满足无偏性、一致性等统计性质；
* 在特定问题中可能还需要考虑计算效率、鲁棒性等。

## 概率函数 & 似然函数

二者都是针对离散的因变量的. 其数学形式相同P ( Y∣θ)=L ( θ∣Y), 但关注的变量不同:

* 如果已知观测数据, 模型参数是未知的, 此时函数是似然函数L ( θ∣Y)
* 如果模型参数θ 是已知的, 此时函数是概率函数P ( Y∣θ)

离散型因变量的P ( Y∣θ)函数, 称为概率密度函数.

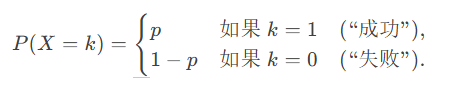
## 伯努利分布

伯努利分布是一种简单的离散概率分布, 具体地说, 如果

* 一个随机事件只有两种可能的互斥结果
* 每次事件的结果是独立的
* 且每次事件中，“成功”和“失败”的概率都是固定的, 如果“成功”概率为 p，而“失败”的概率则为 1−p,

那么结果的概率分布符合伯努利分布。

其数学表达式(伯努利分布的概率质量函数)为:





## 损失函数

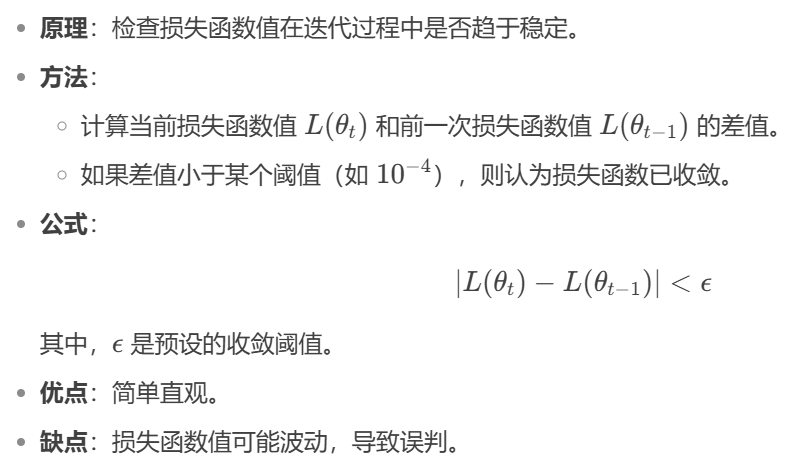
损失函数是针对单个样本计算的误差度量, 用于衡量模型预测值与真实值之间偏差的函数. 在机器学习中, 通常是先找到一个损失函数作为目标函数，然后通过最小化损失函数的结果值求解模型参数.

通常, 最小化损失函数求解模型参数并不是一簇而就的, 而是通过优化算法(如梯度下降法)反复迭代不断调整模型参数最小化损失函数.

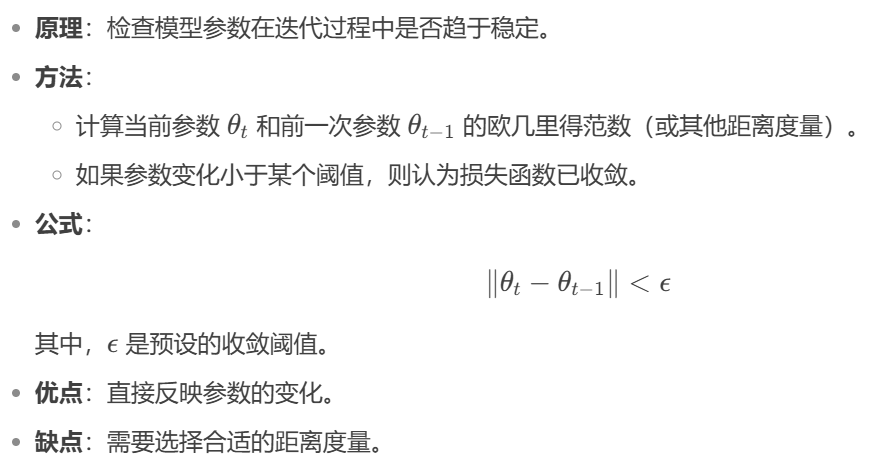
### 损失函数的收敛

在梯度下降法（或其他迭代优化算法）中，判断损失函数是否收敛是优化过程的关键。以下是常用的收敛判断方法：

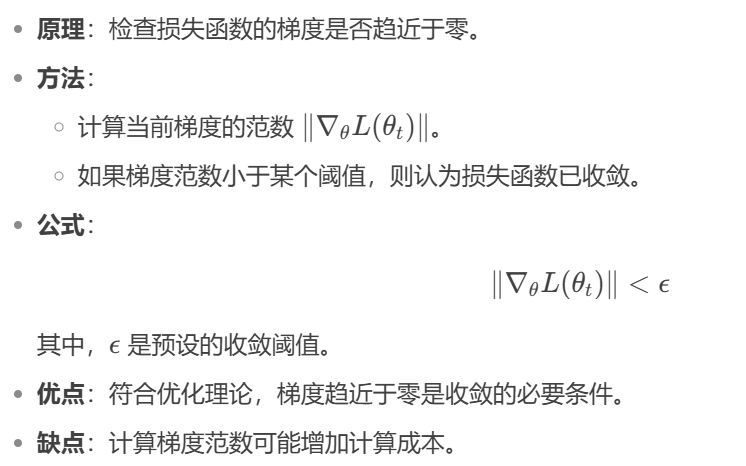
#### 损失函数值的变化



#### 参数值的变化

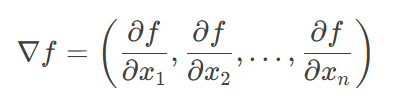


#### 梯度的范数



## 梯度

梯度是多元函数对所有变量的偏导数构成的向量, 即梯度是偏导数的向量形式, 表示函数在各个自变量方向上的变化率, 数学表示为:



梯度的方向是函数在该点上升最快的方向. 梯度的大小表示函数在该方向上的变化率（即斜率）. 梯度越⼩，说明变化率越⼩，趋近于0时，函数取极值, 此时称参数已经收敛了。

梯度的反方向是下降最快的方向

偏导数是梯度的分量.