**逻辑回归分类算法原理**

# 逻辑回归模型

Logistic函数（或称为Sigmoid函数）即为所谓的逻辑回归模型：

 （1）

式中*x*为样本特征；*θ*为模型参数。*h*(*x*)定义域为(﹣∞,+∞)，值域为（0, 1），如图1所示。

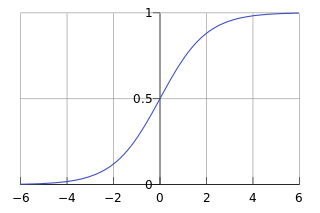


图1 Sigmoid函数曲线图

对于一个二分类问题，令分类结果*y*={1，0}。Sigmoid函数*h*(*x*)的值有特殊的含义，它表示分类结果*y*取1的概率，即：

 （2）

# 代价函数：极大似然函数

假设有*n*个独立的训练样本*x*={(*x*1，*y*1)，(*x*2，*y*2)，...，(*x*n，*y*n)}，*y*={1，0}。那么对每一个观察到的样本（*xi*，*yi*）出现的概率为：

 （3）

对于整个样本集而言，*n*个独立的样本出现的概率为各个样本出现概率的乘积，该乘积表达式也被称为似然函数：

 （4）

对式（4）等号两遍取对数，结合式（2）得到代价函数如式（5）所示：

 （5）

对式（5）所示函数求最大值（极大似然法），得到模型参数*θ*（求最大值的原因：**使得每个样本属于其真实标记的概率越大越好**）*。*在求解模型参数*θ*的过程中，对式（4）求导有时很难得到似然函数的最大值。此时，通常采梯度上升算法迭代求解最优解。

# 梯度上升算法gradient

1. 模型参数*θ*0初始化；
2. 模型参数*θt*+1更新；

 （6）

 （7）

 （8）

 （9）

由式（6）至式（9）得到模型参数*θt*+1更新公式：

 （10）

式中，*yi*为样本标签，*a*为步长。

在实际应用中，样本标签Y，样本特征X常为一个矩阵/向量，于是式（10）可以写成矩阵运行形式：

 （11）

式中，*E*和*X*为：

，



# 随机梯度下降算法SGD (stochastic gradient descent)

梯度下降法（批量梯度下降算法）的代价函数计算需要遍历所有样本，而且是每次迭代都要遍历，直至达到局部最优解，在样本量庞大时就显得收敛速度比较慢了，计算量非常庞大。随机梯度下降仅以当前样本点进行最小值求解，通常无法达到真正局部最优解，但可以比较接近。属于大样本兼顾计算成本的折中方案。

**参考资料：**

<https://www.cnblogs.com/pinard/p/5970503.html>

<https://blog.csdn.net/u014568921/article/details/44856915>

for k in arange(maxIter):

if opts['optimizeType'] == 'gradDescent': #批量梯度下降算法

output = sigmoid(train\_x \* weights)

error = train\_y - output

weights = weights + alpha \* train\_x.transpose() \* error

elif opts['optimizeType'] == 'stocGradDescent':

for i in arange(numSamples):

output = sigmoid(train\_x[i, :] \* weights)

error = train\_y[i, 0] - output

weights = weights + alpha \* train\_x[i, :].transpose() \* error