先看logistic推导：

**Logistic回归原理及公式推导**

转载 2014年11月20日 11:48:39

* 103859

原文见 http://blog.csdn.net/acdreamers/article/details/27365941

**Logistic回归**为概率型非线性回归模型，是研究二分类观察结果http://img.blog.csdn.net/20140528190117703与一些影响因素http://img.blog.csdn.net/20140528190239031之间关系的一种多

变量分析方法。通常的问题是，研究某些因素条件下某个结果是否发生，比如医学中根据病人的一些症状来判断它是

否患有某种病。

在讲解**Logistic回归**理论之前，我们先从LR分类器说起。LR分类器，即**Logistic Regression Classifier。**

在分类情形下，经过学习后的LR分类器是一组权值http://img.blog.csdn.net/20140528191606953，当测试样本的数据输入时，这组权值与测试数据按

照线性加和得到

http://img.blog.csdn.net/20140528191853296

这里http://img.blog.csdn.net/20140528191952140是每个样本的http://img.blog.csdn.net/20140528192118671个特征。

之后按照**sigmoid函数**的形式求出

http://img.blog.csdn.net/20140528192344156

由于**sigmoid函数**的定义域为http://img.blog.csdn.net/20140528192601796，值域为http://img.blog.csdn.net/20140528192701921，因此最基本的LR分类器适合对两类目标进行分类。

所以**Logistic回归**最关键的问题就是研究如何求得http://img.blog.csdn.net/20140528191606953这组权值。这个问题是用**极大似然估计**来做的。

下面正式地来讲Logistic回归模型。

考虑具有http://img.blog.csdn.net/20140528192102375个独立变量的向量http://img.blog.csdn.net/20140528193551562，设条件慨率http://img.blog.csdn.net/20140528193705953为根据观测量相对于某事件http://img.blog.csdn.net/20140528193813921发生的

概率。那么**Logistic回归**模型可以表示为

http://img.blog.csdn.net/20140528194036875

这里http://img.blog.csdn.net/20140528194524250称为**Logistic函数。**其中http://img.blog.csdn.net/20140528201109953

那么在http://img.blog.csdn.net/20140528201222125条件下http://img.blog.csdn.net/20140528190117703不发生的概率为

http://img.blog.csdn.net/20140528201513671

所以事件发生与不发生的概率之比为

http://img.blog.csdn.net/20140528201725562

这个比值称为事件的发生比（the odds of experiencing an event），简记为odds。

对odds取对数得到

http://img.blog.csdn.net/20140528202106515

可以看出Logistic回归都是围绕一个Logistic函数来展开的。接下来就讲如何用**极大似然估计**求分类器的参数。

假设有http://img.blog.csdn.net/20140528202727078个观测样本，观测值分别为http://img.blog.csdn.net/20140528202846812，设http://img.blog.csdn.net/20140528203021781为给定条件下得到http://img.blog.csdn.net/20140528203128218的概率，同样地，

http://img.blog.csdn.net/20140528203228140的概率为http://img.blog.csdn.net/20140528203404875，所以得到一个观测值的概率为http://img.blog.csdn.net/20140528203619875。

因为各个观测样本之间相互独立，那么它们的联合分布为各边缘分布的乘积。得到似然函数为

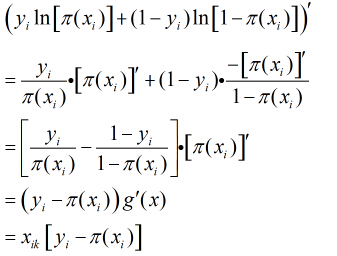
http://img.blog.csdn.net/20140528204102218

然后我们的目标是求出使这一似然函数的值最大的参数估计，最大似然估计就是求出参数http://img.blog.csdn.net/20140528191606953，使得http://img.blog.csdn.net/20140528204318984

取得最大值，对函数http://img.blog.csdn.net/20140528204318984取对数得到

http://img.blog.csdn.net/20140528204640203

继续对这http://img.blog.csdn.net/20140528204822781个http://img.blog.csdn.net/20140528204907171分别求偏导，得到http://img.blog.csdn.net/20140528204822781个方程，比如现在对参数http://img.blog.csdn.net/20140528205058093求偏导，由于



所以得到

http://img.blog.csdn.net/20140528210021312

这样的方程一共有http://img.blog.csdn.net/20140528204822781个，所以现在的问题转化为解这http://img.blog.csdn.net/20140528204822781个方程形成的方程组。

上述方程比较复杂，一般方法似乎不能解之，所以我们引用了**牛顿-拉菲森迭代**方法求解。

利用牛顿迭代求多元函数的最值问题以后再讲。。。

简单牛顿迭代法：**[http://zh.m.wikipedia.org/wiki/%E7%89%9B%E9%A1%BF%E6%B3%95](http://zh.m.wikipedia.org/wiki/%E7%89%9B%E9%A1%BF%E6%B3%95" \t "_blank)**

实际上在上述似然函数求最大值时，可以用梯度上升算法，一直迭代下去。梯度上升算法和牛顿迭代相比，收敛速度

慢，因为梯度上升算法是一阶收敛，而牛顿迭代属于二阶收敛。

**再看FTRL：**

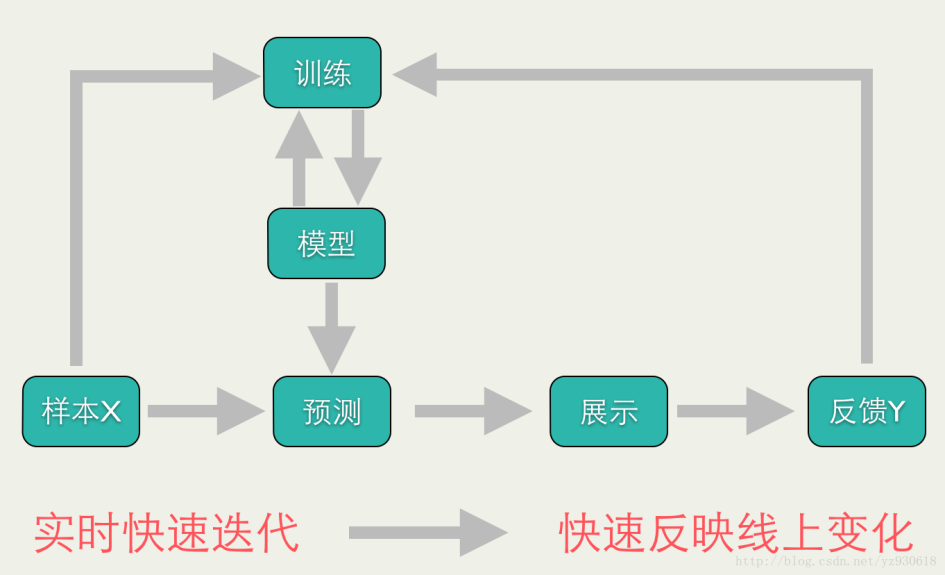
**基于FTRL的在线CTR预测算法**

* https://blog.csdn.net/yz930618/article/details/75270869

在程序化广告投放中，一个优秀的CTR预测算法会给广告主、Adx以及用户都将带来好处。Google公司2013在《ResearchGate》上发表了一篇“Ad click prediction: a view from the trenches”论文，这篇论文是基于FTRL的在线CTR预测算法，下面将讲解该算法的主要思想以及Java实现。

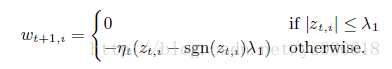
**什么是Online Learning**

传统的**批量算法**的每次迭代是对全体训练数据集进行计算（例如计算全局梯度），优点是精度和收敛还可以，缺点是无法有效处理大数据集（此时全局梯度计算代价太大），且没法应用于数据流做在线学习。而**在线学习算法**的特点是：每来一个训练样本，就用该样本产生的loss和梯度对模型迭代一次，一个一个数据地进行训练，因此可以处理大数据量训练和在线训练。准确地说，Online Learning并不是一种模型，而是一种模型的训练方法，Online Learning能够根据线上反馈数据，实时快速地进行模型调整，使得模型及时反映线上的变化，提高线上预测的准确率。Online Learning的流程包括：将模型的预测结果展现给用户，然后收集用户的反馈数据，再用来训练模型，形成闭环的系统。如下图所示：



这篇论文提出的基于FTRL的在线CTR预测算法，就是一种Online Learning算法。即，针对每一个训练样本，首先通过一种方式进行预测，然后再利用一种损失函数进行误差评估，最后再通过所评估的误差值对参数进行更新迭代。直到所有样本全部遍历完，则结束。那么，如何选择模型预测方法、评估指标以及模型更新公式就是该算法的重点所在。下面将介绍论文中这三部分内容：

1. 预测方法：在每一轮tt中，针对特征样本xt∈Rdxt∈Rd，以及迭代后(第一此则是给定初值)的模型参数wtwt，我们可以预测该样本的标记值：pt=σ(wt,xt)pt=σ(wt,xt)，其中σ(a)=1/(1+exp(−a))σ(a)=1/(1+exp(−a))是一个sigmoid函数。
2. 损失函数：对一个特征样本xtxt，其对应的标记为yt∈0,1yt∈0,1，则通过LogLoss(logistic loss)来作为损失函数，即： lt(wt)=−ytlogpt−(1−yt)log(1−pt)lt(wt)=−ytlogpt−(1−yt)log(1−pt)
3. 迭代公式：我们的目的是使得损失函数尽可能的小，即可以采用极大似然估计来求解参数。首先求梯度 gt=dltdw=(σ(w∗xt)−yt)xt=(pt−yt)xtgt=dltdw=(σ(w∗xt)−yt)xt=(pt−yt)xt，使用FTRL进行迭代：   
   这里写图片描述   
   其中，σsσs为学习率且σ1:t=1ntσ1:t=1nt，g1:t=∑ts=1gtg1:t=∑s=1tgt，λ1λ1为正则化参数。该最优化公式可以化简为：   
   这里写图片描述   
   则，如果我们令zt−1=g1:t−1−∑t−1s=1σswszt−1=g1:t−1−∑s=1t−1σsws，则在第tt轮迭代前，令zt=zt−1+gt−(1nt−1nt−1)wtzt=zt−1+gt−(1nt−1nt−1)wt(**此处和论文中的公式不一致，我觉得应该是减去最后一项，而不是加，作者在后面伪代码中也改成了减，故此处可能是作者笔误**)

下面令梯度为0，则可以得到该优化问题的解析解：   


到此就叙述完该算法的理论部分了，我想大部分人对这部分也不太感兴趣吧，下面直接上伪代码和Java实现吧（过程和理论部分其实是一致的，嘿嘿，想深入的还是研究下理论部分吧）：

