**W2:LR 模型**

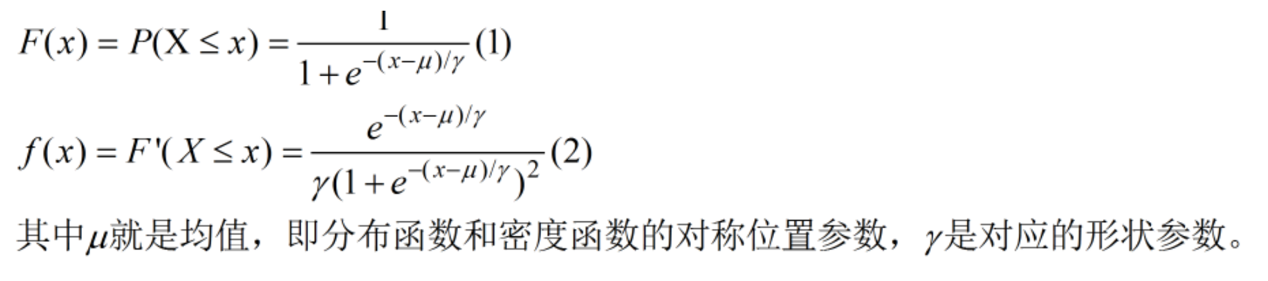
W2:LR 模型W2.1 LR模型理论学习W2.2 特征处理2.2.1 MeanEncoder2.2.2 Poly2.2.3 特征处理方法3W2.3 FFM模型训练2.3.1 数据样集2.3.2 超参数假定W2.4 FFM结果分析W2.5 FFM填坑总结参考文献

**W2.1 LR模型理论学习**

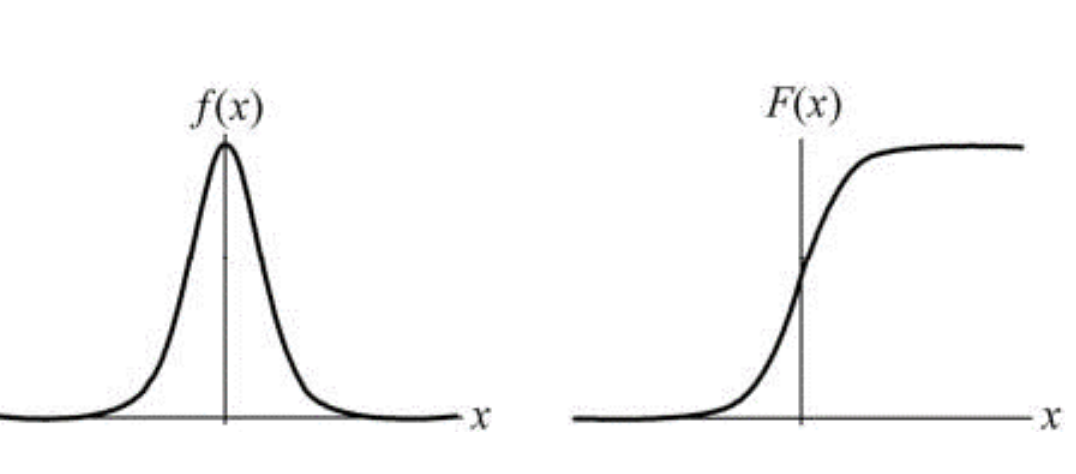
LR模型即是逻辑回归模型，在简单的线性回归模型上，增加了的一层逻辑函数。但是逻辑函数的适用范围非常的狭窄，在复杂的数据集中的预测效果非常不理想。

logistic回归是一种广义线性回归（generalized linear model），因此与多重线性回归分析有很多相同之处。它们的模型形式基本上相同，都具有 w‘x+b，其中w和b是待求参数，其区别在于他们的因变量不同，多重线性回归直接将w‘x+b作为因变量，即y =w‘x+b，而logistic回归则通过函数L将w‘x+b对应一个隐状态p，p =L(w‘x+b),然后根据p 与1-p的大小决定因变量的值。如果L是logistic函数，就是logistic回归，如果L是多项式函数就是多项式回归

逻辑斯蒂是一种变量的分布方式，和常见的指数分布、高斯分布等类似，它也有自己的概率分布函数和概率密度函数.



他们的图形如下图所示：



Logistic回归模型的适用条件

1 因变量为二分类的分类变量或某事件的发生率，并且是数值型变量。但是需要注意，重复计数现象指标不适用于Logistic回归。

2 残差和因变量都要服从二项分布。二项分布对应的是分类变量，所以不是正态分布，进而不是用最小二乘法，而是最大似然法来解决方程估计和检验问题。

3 自变量和Logistic概率是线性关系

4 各观测对象间相互独立。

Logistic回归实质：发生概率除以没有发生概率再取对数。就是这个不太繁琐的变换改变了取值区间的矛盾和因变量自变量间的曲线关系。究其原因，是发生和未发生的概率成为了比值 ，这个比值就是一个缓冲，将取值范围扩大，再进行对数变换，整个因变量改变。不仅如此，这种变换往往使得因变量和自变量之间呈线性关系，这是根据大量实践而总结。所以，Logistic回归从根本上解决因变量要不是连续变量怎么办的问题。还有，Logistic应用广泛的原因是许多现实问题跟它的模型吻合。例如一件事情是否发生跟其他数值型自变量的关系。

**W2.2 特征处理**

**2.2.1 MeanEncoder**

​

​

# 根据特征的fiel特性，进行field编号  
# fields编号字典  
fields\_index={}  
​  
fields\_index['app\_domain'] = 1  
fields\_index['device\_model'] = 2  
fields\_index['C17'] = 3  
fields\_index['C20'] = 4  
​  
fields\_index['C1'] = 5  
fields\_index['banner\_pos'] = 6  
fields\_index['site\_category'] = 7  
fields\_index['app\_category'] = 8  
fields\_index['device\_type'] = 9  
fields\_index['device\_conn\_type'] = 10  
fields\_index['C15'] = 11  
fields\_index['C16'] = 12  
fields\_index['C18'] = 13  
fields\_index['C19'] = 14  
fields\_index['C21'] = 15  
​  
fields\_index['hour'] = 16  
fields\_index['device\_id'] = 17  
fields\_index['device\_ip'] = 18

**2.2.2 Poly**

# 不剔除特征app\_id,site\_id,site\_domain,c14  
​  
# 对特征取值中计数大于10的进行独热编码,小于10次的统统归为一起,独热编码  
count\_category\_features = ['site\_id','site\_domain',  
                       'app\_id','app\_domain',  
                       'device\_model',  
                       'C14','C17', 'C20'  
                      ]  
​  
# 对特征的每个取值进行独热编码  
each\_category\_features = ['C1','banner\_pos',  
                       'site\_category',  
                       'app\_category',  
                       'device\_type','device\_conn\_type',  
                       'C15', 'C16', 'C18', 'C19','C21'  
                      ]  
​  
# 连续值,对取值不进行独热编码,而是填入数值  
continuous\_value\_features = ['hour',#填入点击率  
                           'device\_id','device\_ip' #填入点击率,测试集中点击率等于0  
                          ]

​

# 根据特征的fiel特性，进行field编号  
# fields编号字典  
fields\_index={}  
​  
fields\_index['site\_id'] = 1  
fields\_index['site\_domain'] = 2  
fields\_index['app\_id'] = 3  
fields\_index['app\_domain'] = 4  
fields\_index['device\_model'] = 5  
fields\_index['C14'] = 6  
fields\_index['C17'] = 6  
fields\_index['C20'] = 7  
​  
fields\_index['C1'] = 8  
fields\_index['banner\_pos'] = 9  
fields\_index['site\_category'] = 10  
fields\_index['app\_category'] = 11  
fields\_index['device\_type'] = 12  
fields\_index['device\_conn\_type'] = 13  
fields\_index['C15'] = 14  
fields\_index['C16'] = 14  
fields\_index['C18'] = 15  
fields\_index['C19'] = 16  
fields\_index['C21'] = 17  
​  
fields\_index['hour'] = 18  
fields\_index['device\_id'] = 19  
fields\_index['device\_ip'] = 19

**2.2.3 特征处理方法3**

# 不剔除特征app\_id,site\_id,site\_domain,c14  
# 增加C15XC16，hour\_int，day\_week，hour\_day四个特征  
​  
# 对特征取值中计数大于10的进行独热编码,小于10次的统统归为一起,独热编码  
count\_category\_features = ['site\_id','site\_domain',  
                       'app\_id','app\_domain',  
                       'device\_model',  
                       'C14','C17', 'C20'  
                      ]  
​  
# 对特征的每个取值进行独热编码  
each\_category\_features = ['C1','banner\_pos',  
                       'site\_category',  
                       'app\_category',  
                       'device\_type','device\_conn\_type',  
                       'C15', 'C16', 'C15XC16','C18', 'C19','C21',  
                       'day\_week','hour\_day'  
                      ]  
​  
# 连续值,对取值不进行独热编码,而是填入数值  
continuous\_value\_features = ['hour\_int',#距离第一天的小时数,最后采用最大最小值归一化填数  
                           'device\_id','device\_ip' #填入点击率,测试集中点击率等于0  
                          ]

​

# 根据特征的fiel特性，进行field编号  
# fields编号字典  
fields\_index={}  
​  
fields\_index['site\_id'] = 1  
fields\_index['site\_domain'] = 2  
fields\_index['app\_id'] = 3  
fields\_index['app\_domain'] = 4  
fields\_index['device\_model'] = 5  
fields\_index['C14'] = 6  
fields\_index['C17'] = 6  
fields\_index['C20'] = 7  
​  
fields\_index['C1'] = 8  
fields\_index['banner\_pos'] = 9  
fields\_index['site\_category'] = 10  
fields\_index['app\_category'] = 11  
fields\_index['device\_type'] = 12  
fields\_index['device\_conn\_type'] = 13  
fields\_index['C15'] = 14  
fields\_index['C16'] = 14  
fields\_index['C15XC16'] = 14  
fields\_index['C18'] = 15  
fields\_index['C19'] = 16  
fields\_index['C21'] = 17  
fields\_index['day\_week'] = 18  
fields\_index['hour\_day'] = 18  
​  
fields\_index['hour\_int'] = 18  
fields\_index['device\_id'] = 19  
fields\_index['device\_ip'] = 19

**W2.3 FFM模型训练**

**2.3.1 数据样集**

将train.csv按照日期141021~141030分成了10份数据集，并且这10份数据集进行了随机打乱。

模型的训练集train\_sample200w.csv从141021~1410229这9份数据集中随机挑选5.6%的数据，最终整合为一个大约200w的随机训练样集。

从141030这份数据集中随机抽取1/40的数据，分别当作验证样集validate\_sample10w.csv和测试样集test\_sample10w.csv。

**2.3.2 超参数假定**

# 采用的优化方法为adagrad  
# 学习率lr分别设定了0.001,0.01,0.1三个取值  
# 采用L2正则，正则化参数分别设定了0.002,0.001,0.003三个取值  
​  
param =[]  
param.append({'task':'binary', 'lr':0.1, 'lambda':0.002, 'metric':'auc','epoch':20})  
param.append({'task':'binary', 'lr':0.01, 'lambda':0.002, 'metric':'auc','epoch':20})  
param.append({'task':'binary', 'lr':0.001, 'lambda':0.002, 'metric':'auc','epoch':20})  
param.append({'task':'binary', 'lr':0.1, 'lambda':0.001, 'metric':'auc','epoch':20})  
param.append({'task':'binary', 'lr':0.01, 'lambda':0.003, 'metric':'auc','epoch':20})  
param.append({'task':'binary', 'lr':0.001, 'lambda':0.001, 'metric':'auc','epoch':20})

**W2.4 FFM结果分析**

从图中可以发现：

特征工程2得到了在测试集上最好的结果：0.36001，

特征工程3得到了最差的结果:0.41111。

从三个不同的特征工程走势发现：

1. 学习率不易太小，0.1附近比较好
2. 正则参数不易太大，0.001附近比较好

对于同样的特征工程，喂给FM模型和LR模型，得到结果如下：

FM模型：特征工程2得到了在测试集上最好的结果:0.3644,特征工程3得到了最差的结果:0.41281

LR模型：特征工程3得到了在测试集上最好的结果：0.30227,特征工程1得到了最差的结果：0.35296

综合上面的实验：

1. 学习率不易太小，0.1附近比较好
2. 正则参数不易太大，0.001附近比较好

**W2.5 FFM填坑总结**

1. 轮子要选择好，原来从github上fork了一个充满坑的代码，浪费了好多时间；
2. 训练集，验证集和测试集的各个特征取值不完全一样，在进行编码的时候要统一考虑；
3. 在开始写代码前，务必要领会模型的数学意义，如果能推到出来，就更好了。否则根本就没办法在出问题时候进行问题的解析；

**参考文献**

1. [【github】libffm下载](https://www.csie.ntu.edu.tw/~r01922136/libffm/)
2. [【博客】美团深入FFM原理与实践](http://tech.meituan.com/deep-understanding-of-ffm-principles-and-practices.html)
3. [【论文】libffm原理](http://www.csie.ntu.edu.tw/~r01922136/slides/ffm.pdf)
4. [【博客】CTR预估算法之FM, FFM, DeepFM及实践](https://blog.csdn.net/john_xyz/article/details/78933253)
5. [【API】xlearn文档](http://xlearn-doc.readthedocs.io/en/latest/command_line.html)
6. [【博客】推荐系统遇上深度学习 (一) FM模型理论和实践](https://mp.weixin.qq.com/s?__biz=MzI1MzY0MzE4Mg==&mid=2247483878&idx=1&sn=0a94aff9156bf2902096c77ca2122372&chksm=e9d01127dea79831de1d1d126549cba1996994c210d0be2ae0004b718fd348909b425fe15fe2&scene=21#wechat_redirect)
7. [【博客】推荐系统遇上深度学习(二) -FFM模型理论和实践](https://mp.weixin.qq.com/s?__biz=MzI1MzY0MzE4Mg==&mid=2247483886&idx=1&sn=d333c173163c4909bcd92f07bae5963e&chksm=e9d0112fdea798393a883496197cf680dc5b06c0f53c723ae45c49553e28dd1bf718d48a0646&scene=21#wechat_redirect)
8. [【博客】CTR预估特征工程](https://blog.csdn.net/supinyu/article/details/52351465)
9. [【博客】推荐系统遇上深度学习（四）-多值离散特征的embedding解决方案](https://cloud.tencent.com/developer/news/188316)
10. [【博客】平均数编码：针对高基数定性特征（类别特征）的数据预处理/特征工程](https://blog.csdn.net/juzexia/article/details/78581462)
11. [【博客】通过 sklearn 进行大规模机器学习](http://wulc.me/2017/08/08/%E9%80%9A%E8%BF%87%20sklearn%20%E8%BF%9B%E8%A1%8C%E5%A4%A7%E8%A7%84%E6%A8%A1%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0/)