

点击率预估

点击率预估

1、CTR简介

2、FTRL

 2.1 Logistic 回归

 2.2 SGD算法

 2.3 Follow The Regularized Leader (FTRL)

 2.3.1 FOBOS

 2.3.2 L1-FOBOS

 2.3.3 RDA

 2.3.4 L1- RDA

 2.3.4 FTRL算法原理

 2.3.5 FTRL与SDG之间的关系

 2.3.6 学习率: Per-Coordinate Learning Rates

 2.3.7 FTRL的Python实现

3、GBDT产生特征

4、FM

 4.1 FM模型

 4.2 模型训练

5、FFM

6、总结与展望

参考文献

1、CTR简介

网络广告每年花费数十亿美元，也是各大互联网公司的主要营业收入。在搜索引擎中，在用户输入他/她的查询之后，针对该查询显示有针对性的广告。图1显示了Google的一个示例。广告的目的是连接广告主和消费者。因为广告大部分是按照点击次数计费（Cost Per Click, CPC）的，搜索引擎只有在用户点击广告时才能获得收入。这不是相关广告的完美代理，但足以说明点击广告是相关的。所以搜索引擎喜欢展示相关的广告。另一方面，用户也更喜欢相关的广告。为了满足这些要求，关键因素是针对特定查询预测特定广告的点击概率。广告点击通过率（Click-Through Rate Prediction, CTR）预估是互联网计算广告中的关键环节，预估准确性直接影响公司广告收入。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。

在互联网展示广告（display advertising）中，点击率是指网络广告的实际点击次数除以广告的展现量。点击率越高，意味着在相同投入的情况下，收获了更多的用户注意力。在该任务中，搜索关键词和广告文字是最重要的功能。点击率（CTR）偏低说明可能用户需求与广告描述相关性不高，无法满足潜在受众的需求。可以通过修改广告文案扩大推广机会（不在课程范围内）或通过算法推广与用户需求（如检索关键词和场景）更相关的广告以提高点击率。

图1：Google Adwords广告

由于互联网广告的特性，点CTR预估任务中的特征包括用户查询关键词、广告描述，搜索上下文信息等，维度通常可达数十亿。因此CTR预估系统要用最少的资源在海量数据上训练大量模型的系统存在一些挑战，这些系统应该能够处理：

- 数十亿特征（模型系数）
- 为每天数十亿次的流量提供预测服务
- 数十亿训练样本

对给定场景下的特定广告，判断该广告是否被点击是一个两类分类分类问题（1表示被点击，0表示不被点击）。因此原则上机器学习分类算法在CTR预估中均可使用。但在实际广告点击率预估的应用中，样本数目和特征（对类别型特征独热编码后，特征维数很好，但数据稀疏）的数目均可以达到上亿维度，因此需要选择合适的模型。CTR预估中用的最多的模型是Logistic回归（Logistic Regression, LR）模型[1]。LR模型是线性模型，这种线性模型很容易并行化，可以处理大规模高维数据（上亿条训练样本不是问题）。Google针对大规模高维数据的LR模型的在线训练，提出了FTRL算法[1][2][3]。但线性模型学习能力有限，需要大量特征工程预先分析出有效的特征、特征组合，从而间接增强LR的非线性学习能力。LR因为其简单和易并行，并且基于复杂的特征工程后也能得到非常好的效果，所以在工业界获得了广泛的应用。

LR模型中的特征组合很关键，但依靠人工经验耗时耗力，同时并不一定会带来效果提升。如何自动发现有效的特征、特征组合，弥补人工经验不足，缩短LR特征实验周期，是亟需解决的问题。Facebook通过GBDT（Gradient Boost Decision Tree）解决LR的特征组合问题[4]，随后Kaggle竞赛也有实践此思路[5][6]，GBDT与LR融合开始引起了业界关注。台大提出FM[7]和FFM[8]实现特征二次多项式组合的有效表示，在点击率预估任务中取得了很好的效果。

推荐CTR预估模型的进化之路blog：<https://www.qcloud.com/community/article/701728>

2、FTRL

2.1 Logistic 回归

Logistic回归（Logistic Regression, LR）可以称之上是CTR预估模型的开山鼻祖，因为在大规模系统中的实现简单和高性能，是工业界使用最为广泛的CTR预估模型。在CTR预估中，由于LR为限行模型，模型表示能力有限，通常很少直接用连续的数值型特征，而是通过特征工程将数值特征先离散化，在进行OneHot编码扩展特征维度，得到海量离散特征。

如果用逻辑回归来建模点击率预测的问题，我们根据以下框架来进行：在 t 时刻，利用特征向量 \mathbf{x}_t 来描述一个实例，同时给定模型权重向量 \mathbf{w} ，CTR预测为

$$p_t = \sigma(\mathbf{w}^T \mathbf{x}_t) \quad (1)$$

其中 $\sigma = \frac{1}{1+exp(-a)}$ 为sigmoid函数。该样本对应的log损失函数(logloss)为

$$l_t(\mathbf{w}) = -y_t \log p_t - (1 - y_t) \log(1 - p_t) \quad (2)$$

通过上式可以很容易计算出log损失函数关于 \mathbf{w} 的梯度

$$\nabla l_t(\mathbf{w}) = (p_t - y_t)\mathbf{x}_t \quad (1)$$

由于CTR预估任务中，特征维数非常高，通常我们希望得到的模型是稀疏的。稀疏模型可以可通过增加L1正则实现，即目标函数变为：

$$J(\mathbf{w}) = L(\mathbf{w}) + R(\mathbf{w}) = \sum_{i=1}^N l_i(\mathbf{w}) + \lambda |\mathbf{w}|_1 \quad (2)$$

其中 $R(\mathbf{w})$ 表示正则项。由于L1正则项在0处不可导，可以采用次梯度(Subgradient)来计算L1正则项的梯度(当函数可导时，次梯度=梯度)。对绝对值函数 $f(x) = |x|$ ，其次梯度为：

$$\partial|x| = \begin{cases} \{-1\} & \text{if } x < 0 \\ [-1, +1] & \text{if } x = 0 \\ \{+1\} & \text{if } x > 0 \end{cases}$$

当 $0 \in \partial J(\mathbf{w})|_{\mathbf{w}^*}$ ， \mathbf{w}^* 为目标函数 $J(\mathbf{w})$ 的最优值。

2.2 SGD算法

批处理梯度下降算法 (Batch Gradient Descent) 算法可用于求解LR模型

$$J(\mathbf{w}) = L(\mathbf{w}) + R(\mathbf{w}) = \sum_{i=1}^N l_i(\mathbf{w}) + \lambda |\mathbf{w}|_1 \quad (3)$$

的参数 \mathbf{w} 。

Algorithm 1. Batch Gradient Decent

Repeat until convergence

{

$$\begin{aligned} \mathbf{w}^{(t+1)} &= \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}_J^{(t)} \\ &= \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}_L^{(t)} - \eta^{(t)} \mathbf{g}_R^{(t)} \\ &= \mathbf{w}^{(t)} - \eta^{(t)} \sum_{i=1}^N \nabla l_i(\mathbf{w}^{(t)}) - \eta^{(t)} \partial R^{(t)} \end{aligned}$$

}

其中 $\eta^{(t)}$ 称为学习率，它随着迭代轮数增多而递减，通常将其设置为 $\frac{1}{\sqrt{t}}$ 的函数； $\mathbf{g}_J^{(t)} \in \partial J(\mathbf{w})$ ， $\mathbf{g}_L^{(t)}$ 表示第 t 次迭代中损失函数 $L(\mathbf{w})$ 的梯度， $\mathbf{g}_R^{(t)} \in \partial R(\mathbf{w})$ 。下面为了描述简洁，我们记 $\mathbf{g}_L^{(t)}$ 为 $\mathbf{g}^{(t)}$ 。

当样本数目很多时，对参数做一轮迭代求解中计算损失函数的梯度 $\mathbf{g}^{(t)} = \sum_{i=1}^N \nabla l_i(\mathbf{w}^{(t)})$ 时都需要遍历所有的样本，这在实际应用中未免迭代速度太慢，模型更新速度也太慢。我们可以取部分训练集作为原训练集的子集，使用这些子集做迭代，并逐步求解 \mathbf{w} 的下降方向，逐步对 \mathbf{w} 进行更新。特别的，如果我们每次取原训练样本的一个训练样本，对 \mathbf{w} 的值逐步进行更新，那么我们就得到了**SGD算法**。SGD的参数更新公式为（每次只处理一样样本，所以样本索引*i*同迭代次数*t*）：

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta^{(t)} \nabla l_t(\mathbf{w}^{(t)}) - \eta^{(t)} \partial R^{(t)} \quad (1)$$

与SGD比较，批处理梯度下降算法需要每次扫描所有的样本以计算一个全局梯度，SGD则每次只针对一个观测到的样本进行更新。通常情况下SGD可以更快的逼近最优值，而且SGD每次更新只需要一个样本，使得它很适合进行增量或者在线计算（也就是所谓的Online learning），因此可以处理大数据量训练。在进行大规模的模型训练时，线性模型的Online Learning算法有很多优势。尽管特征向量可能有多达上亿的维度，但每个样本可能只有几百个维度有非零值（特征多为对类别型特征进行独热编码得到，特征是稀疏的）。因为每个样本只需要考虑一次，这种特性允许从硬盘或者网络流式读取数据来进行有效训练。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。

如果我们希望得到的模型是稀疏的，可以采用L1正则化。在Batch模式下，L1正则化通常产生稀疏模型，系数 \mathbf{w} 的一些维度为0。这些为0的维度就代表了不是很相关的维度，从而起到了特征选择（Feature Selection）的目的，也降低了预测计算的复杂度。不同于Batch训练方法的是，Online中每次权重向量 \mathbf{w} 的更新并不是沿着全局梯度进行下降，而是沿着某个样本的产生的梯度方向进行下降，整个寻优过程变得像是一个随机查找的过程（这就是SGD中Stochastic的来历），这样Online最优化求解即使采用L1正则化的方式，也很难产生稀疏解。在各个在线最优化求解算法中，稀疏性是其中一个主要的追求目标。

2.3 Follow The Regularized Leader (FTRL)

SGD算法是常用的online learning算法，能学习出不错的模型，但学出的模型不是稀疏的。为此，学术界和工业界都在研究这样一种online learning算法，它能学习出有效的且稀疏的模型。

FTRL (Follow the Regularized Leader) 算法正是这样一种算法，它由Google的H. Brendan McMahan在2010年提出的[1]，作者后来在2011年发表了一篇关于FTRL和AOGD、FOBOS、RDA比较的论文[2]，2013年又和Gary Holt, D. Sculley, Michael Young等人发表了一篇关于FTRL工程化实现的论文[3]。如论文[3]的内容所述，FTRL算法融合了RDA算法能产生稀疏模型的特性和SGD算法能产生更有效模型的特性。它在处理诸如LR之类的带非光滑正则化项（例如1范数）的凸优化问题上性能非常出色，国内各大互联网公司都已将该算法应用到实际产品中。

文章[9]是篇很好的介绍FTRL算法的中文资料，从TG算法[10]、FOBOS算法[11]开始，到RDA算法[12]，最后到FTRL算法，一脉相承，而且各个算法都有推导过程，值得认真体会。FTRL综合了RDA和FOBOS，但在L1范数或者其他非光滑的正则项下，FTRL比前两者更加有效。

2.3.1 FOBOS

前向后向分裂方法 (Forward-Backward Splitting, FOBOS)是由John Duchi和Yoram Singer提出的[11]，旨在解决正则函数的 R 的非可微分性。该方法通过交替最小化步骤的解析解和次梯度步骤，解决了如L1正则中的不可微分问题。FOBOS亦可在非正式地被视为类似于投影次梯度方法 (projected subgradient method)，只是用有封闭形式解的即时最小化问题来取代或增强投影步骤。FOBOS非常简洁，每个迭代由以下两个步骤组成：

$$\begin{aligned}\mathbf{w}^{(t+1/2)} &= \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}^{(t)} \\ \mathbf{w}^{(t+1)} &= \underset{\mathbf{w}}{\operatorname{argmin}}\left(\frac{1}{2} \|\mathbf{w} - \mathbf{w}^{(t+1/2)}\|^2 + \eta^{(t+1/2)} R(\mathbf{w})\right)\end{aligned}\quad (8)$$

前一个步骤实际上是一个标准的梯度下降（只考虑损失函数的梯度），后一个步骤可以理解为对梯度下降的结果进行微调。观察第二个步骤，发现对 \mathbf{w} 的微调也分为两部分：(1) 前一部分保证微调发生在梯度下降结果的附近；(2) 后一部分则是用少量代价用于处理正则化，产生稀疏性。

将公式(8)中两个步骤合成一个，得到

$$\mathbf{w}^{(t+1)} = \underset{\mathbf{w}}{\operatorname{argmin}}\left(\frac{1}{2} \|\mathbf{w} - \mathbf{w}^{(t)} + \eta^{(t)} \mathbf{g}^{(t)}\|^2 + \eta^{(t+1/2)} R(\mathbf{w})\right) \quad (9)$$

当 0 向量一定属于目标函数的次梯度集合，即

$$0 \in \partial\left(\frac{1}{2} \|\mathbf{w} - \mathbf{w}^{(t)} + \eta^{(t)} \mathbf{g}^{(t)}\|^2 + \eta^{(t+1/2)} R(\mathbf{w})\right)|_{\mathbf{w}=\mathbf{w}^{(t+1)}} \quad (10)$$

计算次梯度带入，得到

$$0 \in \mathbf{w}^{(t+1)} - \mathbf{w}^{(t)} + \eta^{(t)} \mathbf{g}^{(t)} + \eta^{(t+1/2)} \partial R(\mathbf{w}^{(t+1)}) \quad (11)$$

从而得到

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}^{(t)} - \eta^{(t+1/2)} \mathbf{g}_R^{(t+1)} \quad (12)$$

其中 $\mathbf{g}_R^{(t+1)} \in \partial R(\mathbf{w}^{(t+1)})$ 。上面的等式可以理解为其中新权重向量 $\mathbf{w}^{(t+1)}$ 是前一轮权重向量 $\mathbf{w}^{(t)}$ 、前一轮损失函数的梯度 $\mathbf{g}^{(t)}$ 、以及本轮正则函数 R 的次梯度的矢量 $\mathbf{g}_R^{(t+1)}$ 的线性组合，因此称为前向次梯度（Forward-Looking Subgradient），因此算法被命名为FOBOS 的由来（文章的草稿版本称为 FOward LOoking Subgradients, FOLOS。为了不与之前的版本混淆，同时又与之前名字相近，取名为FOBOS，而不是 Fobas）。

上述解法有两个主要的好处。首先，从算法的角度来看，它几乎不需要额外的计算成本就能实现稀疏解决方案。第二，前向梯度使得我们能够在现有的分析基础上，证明所产生框架也享有许多现有的基于梯度和在线凸规划的形式收敛性质（收敛证明请见[11]）。

2.3.2 L1-FOBOS

FOBOS 算法可以与搭配不同的正则函数（如 L2 正则、L1 正则和 L0 正则）。下面我们来看看 FOBOS 如何在 L1 正则化下取得比较好的稀疏性。

L1 正则为 $R(\mathbf{w}) = \lambda |\mathbf{w}|_1$ 。为了简化描述，下面用向量 $\mathbf{v} = [v_1, v_2, \dots, v_D] \in \mathbb{R}^D$ 来表示 $\mathbf{w}^{(t+1/2)}$ ，用标量 $\tilde{\lambda} \in \mathbb{R}$ 来表示 $\eta^{(t+1/2)} \lambda$ ，并将公式(8)等号右边按维度展开，得到

$$\mathbf{w}^{(t+1)} = \underset{\mathbf{w}}{\operatorname{argmin}}\left(\sum_{j=1}^D \frac{1}{2} (w_j - v_j)^2 + \tilde{\lambda} |w_j|\right) \quad (13)$$

我们可以看到，在求和公式 $\{\sum_{j=1}^D \frac{1}{2} (w_j - v_j)^2 + \tilde{\lambda} |w_j|\}$ 中的每一项都是大于等于 0 的，所以类似 L1 正则在线性回归（Lasso）中的求解，公式(13)可以拆解成对特征权重向量 \mathbf{w} 的每一维单独求解：

$$w_j^{(t+1)} = \underset{w_j}{\operatorname{argmin}} \left(\frac{1}{2} (w_j - v_j)^2 + \tilde{\lambda} |w_j| \right) \quad (1)$$

首先，假设 w_j^* 是 $\underset{w_j}{\operatorname{minimize}} \left(\frac{1}{2} (w_j - v_j)^2 + \tilde{\lambda} |w_j| \right)$ 的最优解，则有 $w_j^* * v_j \geq 0$ ，这是因为：

反证法：假设 $w_j^* * v_j < 0$ ，那么有：

$$\frac{1}{2} v_j^2 < \frac{1}{2} v_j^2 - w_j^* * v_j + \frac{1}{2} (w_j^*)^2 < \frac{1}{2} (w_j^* - v_j)^2 + \tilde{\lambda} |w_j^*|$$

这与 w_j^* 是 $\underset{w_j}{\operatorname{minimize}} \left(\frac{1}{2} (w_j - v_j)^2 + \tilde{\lambda} |w_j| \right)$ 的最优解矛盾，故假设不成立， $w_j^* * v_j \geq 0$ 。

既然有 $w_j^* * v_j \geq 0$ ，那么我们分两种情况 $v_j \geq 0$ 和 $v_j < 0$ 来讨论：

(1) 当 $v_j \geq 0$ 时：由于 $w_j^* * v_j \geq 0$ ，所以 $w_j^* \geq 0$ ，相当于对 $\underset{w_j}{\operatorname{minimize}} \left\{ \frac{1}{2} (w_j - v_j)^2 + \tilde{\lambda} |w_j| \right\}$ 引入了不等式约束条件 $-w_j^* \leq 0$ ；

为了求解这个含不等式约束的最优化问题，引入拉格朗日乘子 $\beta \geq 0$ ，由 KKT 条件，有

$$\frac{\partial}{\partial w_j} \left\{ \frac{1}{2} (w_j - v_j)^2 + \tilde{\lambda} |w_j| - \beta w_j \right\} = 0 \text{ 以及 } \beta w_j^* = 0;$$

根据上面的求导等式可得： $w_j^* = v_j - \tilde{\lambda} + \beta$ 分为两种情况：

1.) $w_j^* > 0$ ：

由于 $\beta w_j^* = 0$ ，所以 $\beta = 0$ ，这时候有： $w_j^* = v_j - \tilde{\lambda}$ ；

又由于 $w_j^* > 0$ ，所以 $v_j - \tilde{\lambda} > 0$ 。

2. $w_j^* = 0$ ：

这时候有： $v_j - \tilde{\lambda} + \beta = 0$ ；

又由于 $\beta > 0$ ，所以 $v_j - \tilde{\lambda} \leq 0$ 。

所以，在 $v_j \geq 0$ 时， $w_j^* = \max(0, v_j - \tilde{\lambda})$

(2) 当 $v_j < 0$ 时：采用相同的分析方法，在 $v_j \geq 0$ 时，有 $w_j^* = -\max(0, -v_j - \tilde{\lambda})$

综上所述，可以得到在 FOBOS 在 L1 正则化条件下，特征权重的各个维度更新方式为：

$$w_j^{(t+1)} = \operatorname{sgn}(v_j) \max(0, |v_j| - \tilde{\lambda}) = \operatorname{sgn}(v_j) [0, |v_j| - \tilde{\lambda}]_+ = \operatorname{soft}(v_j; \tilde{\lambda}) \quad (1)$$

其中 $\operatorname{soft}(a; \delta) \triangleq \operatorname{sgn}(a) (|a| - \delta)_+$ 称为软阈值 (soft thresholding)， $x_+ = \max(x, 0)$ 为 x 的正的部分， sgn 为符号函数。

将 $\mathbf{v} = \mathbf{w}^{(t+1/2)} = \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}^{(t)}$ ， $\tilde{\lambda} = \eta^{(t+1/2)} \lambda$ 代入，得到：

$$\begin{aligned} w_j^{(t+1)} &= \operatorname{sgn} \left(w_j^{(t+1/2)} \right) \left[0, |w_j^{(t+1/2)}| - \tilde{\lambda} \right]_+ \\ &= \operatorname{sgn} \left(w_j^{(t)} - \eta^{(t)} g_j^{(t)} \right) \left[0, |w_j^{(t)} - \eta^{(t)} g_j^{(t)}| - \eta^{(t+1/2)} \lambda \right]_+ \end{aligned} \quad (1)$$

从公式(16)可以看出，L1-FOBOS在每次更新权重向量 \mathbf{w} 的时候，对权重向量 \mathbf{w} 的每个维度都会进行判定。当满足 $|w_j^{(t)} - \eta^{(t)} g_j^{(t)}| - \eta^{(t+1/2)} \lambda \leq 0$ ，即 $|w_j^{(t)} - \eta^{(t)} g_j^{(t)}| \leq \eta^{(t+1/2)} \lambda$ 时，对该维度进行截断，得到 $w_j^{(t+1)} = 0$ 。也就是说，当一个样本产生的梯度不足以令对应维度上的权重值发生足够大的变化 $\eta^{(t+1/2)} \lambda$ 时，认为在本次更新过程中该维度不够重要，应当令其权重为0。

为了更清楚地看到截断效果，公式(16)也可写成：

$$w_j^{(t+1)} = \begin{cases} 0 & \text{if } |w_j^{(t)} - \eta^{(t)} g_j^{(t)}| \leq \eta^{(t+1/2)} \lambda \\ (w_j^{(t)} - \eta^{(t)} g_j^{(t)}) - \eta^{(t+1/2)} \lambda \operatorname{sgn}(w_j^{(t)} - \eta^{(t)} g_j^{(t)}) & \text{otherwise} \end{cases}$$

2.3.3 RDA

FOBOS 建立在 SGD 的基础之上的，属于梯度下降类型的方法。这类型方法的优点就是精度比较高，并且也能在稀疏性上得到提升。正则对偶平均(Regularized Dual Averaging, RDA) 是微软2010年提出的一种算法[12]，从另一个方面来求解 Online Optimization 并且更有效地提升了特征权重的稀疏性。

在 RDA 中，特征权重的更新策略为：

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w}} \left(\frac{1}{t} \sum_{s=1}^t \mathbf{g}^{(s)} \cdot \mathbf{w} + R(\mathbf{w}) + \frac{\beta^{(t)}}{t} h(\mathbf{w}) \right) \quad (1)$$

其中第一项 $\frac{1}{t} \sum_{s=1}^t \mathbf{g}^{(s)} \cdot \mathbf{w}$ 表示梯度 $\mathbf{g}^{(s)}$ 对 \mathbf{w} 的积分（平均值，dual average），第二项 $R(\mathbf{w})$ 为正则项，第三项为额外的正则项， $h(\mathbf{w})$ 为一个辅助的严格凸函数， $\{\beta^{(t)} \mid t \geq 1\}$ 是一个非负且非自减序列。

2.3.4 L1- RDA

我们下面来看看在 L1 正则化下，RDA 中的特征权重更新具有什么样的形式以及如何产生稀疏性。

L1正则为 $R(\mathbf{w}) = \lambda |\mathbf{w}|_1$ 。并且 $h(\mathbf{w})$ 是一个关于 \mathbf{w} 的严格凸函数，不妨令 $h(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2$ ，此外，将非负非自减序列 $\{\beta^{(t)} \mid t \geq 1\}$ 定义为 $\beta^{(t)} = \gamma \sqrt{t}$ 。将这些表示代入公式(17)有：

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w}} \left(\frac{1}{t} \sum_{s=1}^t \mathbf{g}^{(s)} \cdot \mathbf{w} + \lambda |\mathbf{w}|_1 + \frac{\gamma}{2\sqrt{t}} \|\mathbf{w}\|_2^2 \right) \quad (1)$$

类似L1-FOBOS，将公式(19)拆解成对特征权重向量 \mathbf{w} 的每一维单独求解：

$$w_j^{(t+1)} = \arg \min_{w_j} \left(\bar{g}_j^{(t)} * w_j + \lambda |w_j| + \frac{\gamma}{2\sqrt{t}} w_j^2 \right) \quad (2)$$

这里 $\lambda > 0, \gamma > 0, \bar{g}_j^{(t)} = \frac{1}{t} \sum_{s=1}^t g_j^{(s)}$ 。

公式(19)就是一个无约束的非平滑最优化问题。第 2 项 $\lambda |w_j|$ 在 $w_j = 0$ 处不可导。假设 w_j^* 是其最优解，并且定义 $\xi \in \partial(|w_j|) \mid_{w_j=w_j^*}$ 为 $|w_j|$ 在 w_j^* 的次导数，那么有：

$$\partial(|w_j|) \mid_{w_j=w_j^*} = \begin{cases} \{-1\} & \text{if } w_j^* < 0 \\ [-1, +1] & \text{if } w_j^* = 0 \\ \{+1\} & \text{if } w_j^* > 0 \end{cases} \quad (2)$$

如果对公式(20)求导(求次导数)并等于 0, 则有:

$$\bar{g}_j^{(t)} + \lambda \xi + \frac{\gamma}{\sqrt{t}} w_j = 0 \quad (2)$$

由于 $\lambda > 0$, 我们针对(22)分三种情况 $|\bar{g}_j^{(t)}| < \lambda$ 、 $\bar{g}_j^{(t)} > \lambda$ 、和 $\bar{g}_j^{(t)} < -\lambda$ 来进行讨论:

(1) 当 $|\bar{g}_j^{(t)}| < \lambda$ 时:

还可以分为三种情况: 1 如果 $w_j^* = 0$, 由(22)得 $\xi = -\frac{\bar{g}_j^{(t)}}{\lambda} \in \partial(|w_j|)|_{w_j=0}$, 满足(21);

2 如果 $w_j^* > 0$, 由(21)得 $\xi = 1$, 则 $\bar{g}_j^{(t)} + \lambda + \frac{\gamma}{\sqrt{t}} w_j > \bar{g}_j^{(t)} + \lambda \geq 0$, 不满足(22);

3 如果 $w_j^* < 0$, 由(21)得 $\xi = -1$, 则 $\bar{g}_j^{(t)} - \lambda + \frac{\gamma}{\sqrt{t}} w_j < \bar{g}_j^{(t)} - \lambda \leq 0$, 不满足(22);

所以, 当 $|\bar{g}_j^{(t)}| < \lambda$ 时, $w_j^* = 0$.

(2) 当 $|\bar{g}_j^{(t)}| > \lambda$ 时:

采用相同分析方法得到 $w_j^* < 0$, 有 $\xi = -1$ 满足条件, 即 $w_j^* = -\frac{\gamma}{\sqrt{t}}(\bar{g}_j^{(t)} - \lambda)$ 。

(3) 当 $|\bar{g}_j^{(t)}| < -\lambda$ 时:

采用相同分析方法得到 $w_j^* > 0$, 有 $\xi = 1$ 满足条件, 即 $w_j^* = -\frac{\gamma}{\sqrt{t}}(\bar{g}_j^{(t)} + \lambda)$ 。

综上所述, 可以得到在 L1-RDA 特征权重的各个维度更新方式为:

$$w_j^{(t+1)} = \begin{cases} 0 & \text{if } |\bar{g}_j^{(t)}| < \lambda \\ -\frac{\gamma}{\sqrt{t}} (\bar{g}_j^{(t)} - \lambda \operatorname{sgn}(\bar{g}_j^{(t)})) & \text{otherwise} \end{cases} \quad (2)$$

即当某维度满累积梯度平均值 $|\bar{g}_j^{(t)}|$ 小于阈值 λ 时, 对该维度进行截断, 得到 $w_j^{(t+1)} = 0$ 。特征权重的系属性以由此产生。

对比 L1-FOBOS 的截断阈值为 $\eta^{(t+1/2)}\lambda$, 通常 $\eta = \frac{1}{\sqrt{t}}$, 随着 t 的增加, 这个阈值 $\frac{1}{\sqrt{t}}\lambda$ 会逐渐降低。而 L1-RDA 的截断阈值为 λ , 并不随着 t 而变化, 因此可以认为 L1-RDA 比 L1-FOBOS 在截断判定上更加 aggressive。这种性质使得 L1-RDA 更容易产生稀疏性; 此外, RDA 中判定对象是梯度的累加平均值 $|\bar{g}_j^{(t)}|$, 不同于 L1-FOBOS 中针对单次梯度计算的结果进行判定, 避免了由于某些维度由于训练不足导致截断的问题。并且通过调节 λ 一个参数, 很容易在精度和稀疏性上进行权衡。

2.3.4 FTRL 算法原理

本节我们首先从形式上将 L1-FOBOS 和 L1-RDA 进行统一, 然后介绍从这种统一形式产生 FTRL 算法, 以及介绍 FTRL 算法工程化实现的算法逻辑。

在 2.3.3 节中我们从原理上定性比较了 L1-FOBOS 和 L1-RDA 在稀疏性上的表现。有实验证明, L1-FOBOS 这一类基于梯度下降的方法有比较高的精度, 但是 L1-RDA 却能在损失一定精度的情况下产生更好的稀疏性。那么这两者的优点能不能在一个算法上体现出来? 这就是 FTRL 要解决的问题。

L1-FOBOS 的权重每一维更新公式为:

$$w_j^{(t+1)} = \underset{w_j}{\operatorname{argmin}} \left(\frac{1}{2} \left(w_j - w_j^{(t)} + \eta^{(t)} g_j^{(t)} \right)^2 + \eta^{(t+1/2)} \lambda |w_j| \right) \quad (2)$$

令 $\eta^{(t+1/2)} = \eta^{(t)} = \frac{1}{\sqrt{t}}$ ， 得到

$$\begin{aligned} w_j^{(t+1)} &= \underset{w_j}{\operatorname{argmin}} \left(\frac{1}{2} \left(w_j - w_j^{(t)} + \eta^{(t)} g_j^{(t)} \right)^2 + \eta^{(t)} \lambda |w_j| \right) \\ &= \underset{w_j}{\operatorname{argmin}} \left(\frac{1}{2} \left(w_j - w_j^{(t)} \right)^2 + \frac{1}{2} \left(\eta^{(t)} g_j^{(t)} \right)^2 + w_j \eta^{(t)} g_j^{(t)} + w_j^{(t)} \eta^{(t)} g_j^{(t)} + \eta^{(t)} \lambda |w_j| \right) \quad (2) \\ &= \underset{w_j}{\operatorname{argmin}} \left(w_j g_j^{(t)} + \lambda |w_j| + \frac{1}{2\eta^{(t)}} \left(w_j - w_j^{(t)} \right)^2 + \frac{\eta^{(t)}}{2} \left(g_j^{(t)} \right)^2 + w_j^{(t)} g_j^{(t)} \right) \end{aligned}$$

由于 $\frac{\eta^{(t)}}{2} \left(g_j^{(t)} \right)^2 + w_j^{(t)} g_j^{(t)}$ 与变量 w_j 无关，因此上式可以等价于：

$$w_j^{(t+1)} = \underset{w_j}{\operatorname{argmin}} \left(w_j g_j^{(t)} + \lambda |w_j| + \frac{1}{2\eta^{(t)}} \left(w_j - w_j^{(t)} \right)^2 \right) \quad (2)$$

再将这 D 个独立最优化子步骤合并，那么 L1-FOBOS 可以写作：

$$\mathbf{w}^{(t+1)} = \underset{\mathbf{w}}{\operatorname{argmin}} \left(\mathbf{w} \cdot \mathbf{g}^{(t)} + \lambda \|\mathbf{w}\|_1 + \frac{1}{2\eta^{(t)}} \|\mathbf{w} - \mathbf{w}^{(t)}\|_2^2 \right) \quad (2)$$

对比 L1-RDA 的公式(19)，

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w}} \left(\sum_{s=1}^t \mathbf{g}^{(s)} \cdot \mathbf{w} + \lambda \|\mathbf{w}\|_1 + \frac{\gamma}{2\sqrt{t}} \|\mathbf{w}\|_2^2 \right) \quad (1)$$

右边乘以 t ，不影响极值点的位置，得到：

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w}} \left(\sum_{s=1}^t \mathbf{g}^{(s)} \cdot \mathbf{w} + t\lambda \|\mathbf{w}\|_1 + \frac{\gamma}{2\eta^{(t)}} \|\mathbf{w} - \mathbf{0}\|_2^2 \right) \quad (2)$$

令 $\mathbf{g}^{(1:t)} = \sum_{s=1}^t \mathbf{g}^{(s)}$, $\sigma^s = \eta^{(s)} - \eta^{(s-1)}$, $\sigma^{(1:t)} = \frac{1}{\eta^{(t)}}$ ，公式(27)和公式(28)可以写作：

$$\mathbf{w}^{(t+1)} = \underset{\mathbf{w}}{\operatorname{argmin}} \left(\mathbf{g}^{(t)} \cdot \mathbf{w} + \lambda \|\mathbf{w}\|_1 + \frac{1}{2} \sigma^{(1:t)} \|\mathbf{w} - \mathbf{w}^{(t)}\|_2^2 \right) \quad (2)$$

$$\mathbf{w}^{(t+1)} = \underset{\mathbf{w}}{\operatorname{argmin}} \left(\mathbf{g}^{(1:t)} \cdot \mathbf{w} + t\lambda \|\mathbf{w}\|_1 + \frac{1}{2} \sigma^{(1:t)} \|\mathbf{w} - \mathbf{0}\|_2^2 \right) \quad (3)$$

比较(29)和(30)这两个公式，可以看出 L1-FOBOS 和 L1-RDA 的区别在于：(1) 前者对计算的是累加梯度以及 L1 正则项只考虑当前模的贡献，而后者采用了累加的处理方式；(2) 前者的第三项限制 \mathbf{w} 的变化不能离已迭代过的解太远，而后者则限制 \mathbf{w} 不能离 0 点太远。

FTRL 综合考虑了 FOBOS 和 RDA 对于正则项和 \mathbf{w} 限制的区别，其特征权重的更新公式为：

$$\mathbf{w}^{(t+1)} = \underset{\mathbf{w}}{\operatorname{argmin}} \left(\mathbf{g}^{(1:t)} \cdot \mathbf{w} + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2 + \frac{1}{2} \sum_{s=1}^t \sigma^{(s)} \|\mathbf{w} - \mathbf{w}^{(s)}\|_2^2 \right) \quad (3)$$

注意上式中额外增加了L2正则项，L2正则项的引入并不影响FTRL的稀疏性，仅仅相当于对最优化过程多了一

个约束，使得结果求解结果更加“平滑”。

公式(31)看上去很复杂，更新特征权重貌似非常困难的样子。不妨将其进行改写，将最后一项展开，等价于求下面这样一个最优化问题：

$$\mathbf{w}^{(t+1)} = \underset{\mathbf{w}}{\operatorname{argmin}} \left(\left(\mathbf{g}^{(1:t)} - \sum_{s=1}^t \sigma^{(s)} \mathbf{w}^{(s)} \right) \cdot \mathbf{w} + \lambda_1 \|\mathbf{w}\|_1 + \frac{1}{2} \left(\lambda_2 + \sum_{s=1}^t \sigma^{(s)} \right) \|\mathbf{w}\|_2^2 + \frac{1}{2} \sum_{s=1}^t \sigma^{(s)} \|\mathbf{w}^{(s)}\|_2^2 \right) \quad (32)$$

由于 $\frac{1}{2} \sum_{s=1}^t \sigma^{(s)} \|\mathbf{w}^{(s)}\|_2^2$ 相对于 \mathbf{w} 来说是一个常数，并且令 $\mathbf{z}^{(t)} = \mathbf{g}^{(1:t)} - \sum_{s=1}^t \sigma^{(s)} \mathbf{w}^{(s)}$ ，上式等价于：

$$\mathbf{w}^{(t+1)} = \underset{\mathbf{w}}{\operatorname{argmin}} \left(\mathbf{z}^{(t)} \cdot \mathbf{w} + \lambda_1 \|\mathbf{w}\|_1 + \frac{1}{2} \left(\lambda_2 + \sum_{s=1}^t \sigma^{(s)} \right) \|\mathbf{w}\|_2^2 \right) \quad (3)$$

公式(33)的问题同公式(20)

$$w_j^{(t+1)} = \arg \min_{w_j} \left(\bar{g}_j^{(t)} + \lambda |w_j| + \frac{\gamma}{2\sqrt{t}} w_j^2 \right) \quad (2)$$

一样，从而得到FTRL的权重更新公式为：

$$w_j^{(t+1)} = \begin{cases} 0 & \text{if } |z_j^{(t)}| < \lambda_1 \\ -(\lambda_2 + \sum_{s=1}^t \sigma^{(s)})^{-1} (z_j^{(t)} - \lambda_1 \operatorname{sgn}(z_j^{(t)})) & \text{otherwise} \end{cases} \quad (3)$$

从公式(34)可以看出，引入 L2 正则化并没有对 FTRL 结果的稀疏性产生任何影响。

2.3.5 FTRL与SGD之间的关系

为了表示简洁，下面我们对不使用正则化FTRL与SGD进行讨论，发现他们之间其实也是等价的（带上正则还是等价）。

<https://vividfree.github.io/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0/2015/12/05/understanding-FTRL-algorithm>

SGD算法的迭代计算公式为：

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}^{(t)} \quad (3)$$

FTRL算法的迭代算公式如下：

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w}} \left(\sum_{s=1}^t \mathbf{g}^{(s)} \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^t \sigma^{(s)} \|\mathbf{w} - \mathbf{w}^{(s)}\|_2^2 \right) \quad (3)$$

其中 t 为迭代轮数, \mathbf{w} 是模型参数, $\sigma^{(s)}$ 定义为 $\sum_{s=1}^t \sigma^{(s)} = \frac{1}{\eta^{(t)}}$ 。

公式(36)看似比公式(35)复杂, 需要求最优解, 但其实很容易算出 $\mathbf{w}^{(t+1)}$ 的解析解 (close form 表达式)。实际上二者是等价的, 下文对这点进行证明。

首先, 记公式(36)右侧 $\arg \min$ 算子的内容为 $f(\mathbf{w})$, 显然 $f(\mathbf{w})$ 是凸函数, 所以公式(36)存在极值。将其对 \mathbf{w} 求梯度, 得到

$$\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} = \sum_{s=1}^t \mathbf{g}^{(s)} + \sum_{s=1}^t \sigma^{(s)} (\mathbf{w} - \mathbf{w}^{(s)}) \quad (3)$$

令上述梯度等于零向量, 即可得极值, 极值正是 $\mathbf{w}^{(t+1)}$ 。这样也就是有下面的公式成立:

$$\sum_{s=1}^t \mathbf{g}^{(s)} + \sum_{s=1}^t \sigma^{(s)} (\mathbf{w}^{(t+1)} - \mathbf{w}^{(s)}) = 0 \quad (3)$$

将含有 $\mathbf{w}^{(t+1)}$ 的项放到等号左边, 剩下的放在右边, 得到:

$$(\sum_{s=1}^t \sigma^{(s)}) \mathbf{w}^{(t+1)} = \sum_{s=1}^t \sigma^{(s)} \mathbf{w}^{(s)} - \sum_{s=1}^t \mathbf{g}^{(s)} \quad (3)$$

进一步化简得到:

$$\frac{1}{\eta^t} \mathbf{w}^{(t+1)} = \sum_{s=1}^t \sigma^{(s)} \mathbf{w}^{(s)} - \sum_{s=1}^t \mathbf{g}^{(s)} \quad (4)$$

用 $t - 1$ 替换 t , 得到:

$$\frac{1}{\eta^{t-1}} \mathbf{w}^{(t)} = \sum_{s=1}^{t-1} \sigma^{(s)} \mathbf{w}^{(s)} - \sum_{s=1}^{t-1} \mathbf{g}^{(s)} \quad (4)$$

用公式(40)减去公式(41), 即式子的左边右边同时减去, 得到:

$$\frac{1}{\eta^t} \mathbf{w}^{(t+1)} - \frac{1}{\eta^{t-1}} \mathbf{w}^{(t)} = \sigma^{(t)} \mathbf{w}^{(t)} - \mathbf{g}^{(t)} \quad (4)$$

将 $\sigma^{(t)}$ 用 $\eta^{(t)}$ 表示: $\sigma^{(s)} = \frac{1}{\eta^{(t)}} - \frac{1}{\eta^{(t-1)}}$, 带入公式(42), 得到公式:

$$\frac{1}{\eta^{(t)}} \mathbf{w}^{(t+1)} - \frac{1}{\eta^{(t-1)}} \mathbf{w}^{(t)} = \left(\frac{1}{\eta^{(t)}} - \frac{1}{\eta^{(t-1)}} \right) \mathbf{w}^{(t)} - \mathbf{g}^{(t)} \quad (4)$$

对公式(43)化简, 得到

$$\frac{1}{\eta^{(t)}} \mathbf{w}^{(t+1)} = \frac{1}{\eta^{(t)}} \mathbf{w}^{(t)} - \mathbf{g}^{(t)} \quad (4)$$

两边同乘以 $\eta^{(t)}$, 即可得到 SGD 的公式(35)。

通过上面的推导证明, 我们看到公式(36)与公式(35)确实等价。由此可以更好的体会下公式(31)为什么是这样, 即在 SGD 算法的基础上增加正则功能。

FTRL，较SGD有以下优势：

1. 带有L1正则，学习的特征更加稀疏
2. 使用累计的梯度，加速收敛
3. 根据特征在样本的出现频率确定该特征学习率，每个维度上的学习率都是单独考虑的，保证每个特征有充分的学习

2.3.6 学习率：Per-Coordinate Learning Rates

在一个标准的OGD里面使用的是一个全局的学习率策略 $\eta^{(t)} = \frac{1}{\sqrt{t}}$ ，这个策略保证了学习率是一个正的非增长序列，对于每一个特征维度都是一样的。

FTRL 中，每个维度上的学习率都是单独考虑的(Per-Coordinate Learning Rates)。如果特征 1 比特征 2 的变化更快，那么在维度 1 上的学习率应该下降得更快。我们很容易就可以想到可以用某个维度上梯度分量来反映这种变化率。FTRL 采用**Per-Coordinate Learning Rates**可以对于某个特征的完整性比较低的时候，会动态加大学习率，而对于特征完整性高的，会减小学习率，让特征权重慢慢学。学习率的计算公式如下，即步长等于历次梯度的平方和的开方的倒数：

$$\eta_j^{(t)} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^t (g_j^{(s)})^2}} \quad (4)$$

由于 $\sigma^{(1:t)} = \frac{1}{\eta^{(t)}}$ ，所以公式(34)

$$w_j^{(t+1)} = \begin{cases} 0 & \text{if } |z_j^{(t)}| < \lambda_1 \\ -(\lambda_2 + \sum_{s=1}^t \sigma^{(s)})^{-1} (z_j^{(t)} - \lambda_1 \operatorname{sgn}(z_j^{(t)})) & \text{otherwise} \end{cases} \quad (3)$$

中 $\sum_{s=1}^t \sigma^{(s)} = \frac{\beta + \sqrt{\sum_{s=1}^t (g_j^{(s)})^2}}{\alpha}$ 。这里的 α 和 β 是需要输入的参数。公式(34)中学习率写成累加的形式，是为了方便理解后面 FTRL 的迭代计算逻辑。

文献[3]中给出了FTRL算法的伪代码，并且也参数设置的指导原则，在此不再赘述。文献[3]中的实验表明这种每维单独考虑学习率的方式AUC指标可提高11.2%（通常提供1%就被认为是很大的提高）。文章[3]还给出了工程实现上节省内存的一些技巧。

FTRL的伪代码：

Algorithm 1 Per-Coordinate FTRL-Proximal with L1 and L2 Regularization for Logistic Regression

#With per-coordinate learning rates of Eq(34):

Input: parameters $\alpha, \beta, \lambda_1, \lambda_2 (\forall j \in 1, \dots, D)$,

initialize $z_j = \mathbf{0}$ and $n_j = \mathbf{0}$ (累积梯度平方和)

for $t = 1$ to T do

Receive feature vector \mathbf{x}_t and let $J = \{j | x_j \neq 0\}$

For $j \in J$ compute

$$w_j = \begin{cases} 0 & \text{if } |z_j| < \lambda_1 \\ -(\lambda_2 + \sum_{s=1}^t \sigma^{(s)})^{-1} (z_j - \lambda_1 \operatorname{sgn}(z_j)) & \text{otherwise} \end{cases}$$

Predict $p_t = \sigma(\mathbf{x}_t \cdot \mathbf{w})$ using the w_j computed above

Observe label $y_t \in \{0, 1\}$

for all $j \in J$ do

$$\sigma_j = \frac{1}{\alpha} \left(\sqrt{n_j + g_j^2} - \sqrt{n_j} \right) \# \text{equals } \sigma_j^{(t)} = \eta_j^{(t)} - \eta_j^{(t-1)} \quad z_j \leftarrow z_j + g_j - \sigma_j w_j \quad n_j \leftarrow n_j + g_j^2$$

end for

end for

Algorithm 1 Per-Coordinate FTRL-Proximal with L1 and L2 Regularization for Logistic Regression

#With per-coordinate learning rates of Eq:

$$w_j^{(t+1)} = \begin{cases} 0 & \text{if } |z_j^{(t)}| < \lambda_1 \\ -(\lambda_2 + \sum_{s=1}^t \sigma^{(s)})^{-1} (z_j^{(t)} - \lambda_1 \operatorname{sgn}(z_j^{(t)})) & \text{otherwise} \end{cases}$$

Input: parameters $\alpha, \beta, \lambda_1, \lambda_2$ ($\forall j \in 1, \dots, D$), initialize $z_j = 0$ and $n_j = 0$ (累积梯度平方和)

for $t = 1$ to T do

Receive feature vector \mathbf{x}_t and let $J = \{j | x_j \neq 0\}$ For $j \in J$ compute

$$w_j = \begin{cases} 0 & \text{if } |z_j| < \lambda_1 \\ -(\lambda_2 + \sum_{s=1}^t \sigma^{(s)})^{-1} (z_j - \lambda_1 \operatorname{sgn}(z_j)) & \text{otherwise} \end{cases}$$

Predict $p_t = \sigma(\mathbf{x}_t \cdot \mathbf{w})$ using the w_j computed above Observe label $y_t \in \{0, 1\}$ for all $j \in J$ do

$$g_j = (p_t - y_t)x_j \# \text{gradient of loss w.r.t. } w_j$$

$$\sigma_j = \frac{1}{\alpha} \left(\sqrt{n_j + g_j^2} - \sqrt{n_j} \right) \# \text{equals } \sigma_j^{(t)} = \eta_j^{(t)} - \eta_j^{(t-1)} \quad z_j \leftarrow z_j + g_j - \sigma_j w_j \quad n_j \leftarrow n_j + g_j^2$$

end for

```
end for
```

2.3.7 FTRL的Python实现

网上可用的一些字包含的工具包：

<https://github.com/fmfn/FTRLp>

<https://www.kaggle.com/jiweiliu/ftrl-starter-code/output>

<https://www.kaggle.com/c/avazu-ctr-prediction/discussion/10927>

Tensorflow支持FtrlOptimizer。

3、GBDT产生特征

GBDT (Gradient Boost Decision Tree) 是一种常用的非线性模型，作为学习器被经常使用。GBDT的思想使其具有天然优势，可以发现多种有区分性的特征以及特征组合，决策树的路径可以直接作为LR输入特征使用，省去了人工寻找特征、特征组合的步骤。这种通过GBDT生成LR特征的方式 (GBDT+LR)，业界已有实践Facebook[4]；Kaggle竞赛上也有用GBDT提取特征，然后和FM/FFM模型结合 [5][6]，且效果不错。

推荐CTR预估中GBDT与LR融合方案的文章：

http://blog.csdn.net/lilyth_lilyth/article/details/48032119

论文[4]的思想很简单，就是先用已有特征训练GBDT模型，然后利用GBDT模型学习到的树来构造新特征，最后把这些新特征加入原有特征一起训练模型。构造的新特征向量GBDT与LR的融合方式如下图所示，图中Tree1、Tree2为通过GBDT模型学出来的两棵树， \mathbf{x} 为一条输入样本，遍历两棵树后， \mathbf{x} 样本分别落到两棵树的叶子节点上，每个叶子节点对应LR一维特征。那么通过遍历树，就得到了该样本对应的所有LR特征。新特征向量的长度等于GBDT模型里树的数目（当然叶子节点索引还只是类别型编码，输入LR模型还需OneHot编码）。

举例说明。假设样本 \mathbf{x} 对应，第一棵树的第3个叶子节点，第二棵树的第1个叶子节点，那么通过GBDT获得的新特征向量为[3, 1]，其中向量中的前1位元素对应第一棵树的叶子节点索引（3），第2个元素第二棵树的叶子节点索引（1）。由于树的每条路径，是通过最小化均方差/Gini指标等方法最终分割出来的有区分性路径，根据该路径得到的特征、特征组合都相对有区分性，效果理论上不会亚于人工经验的处理方式。

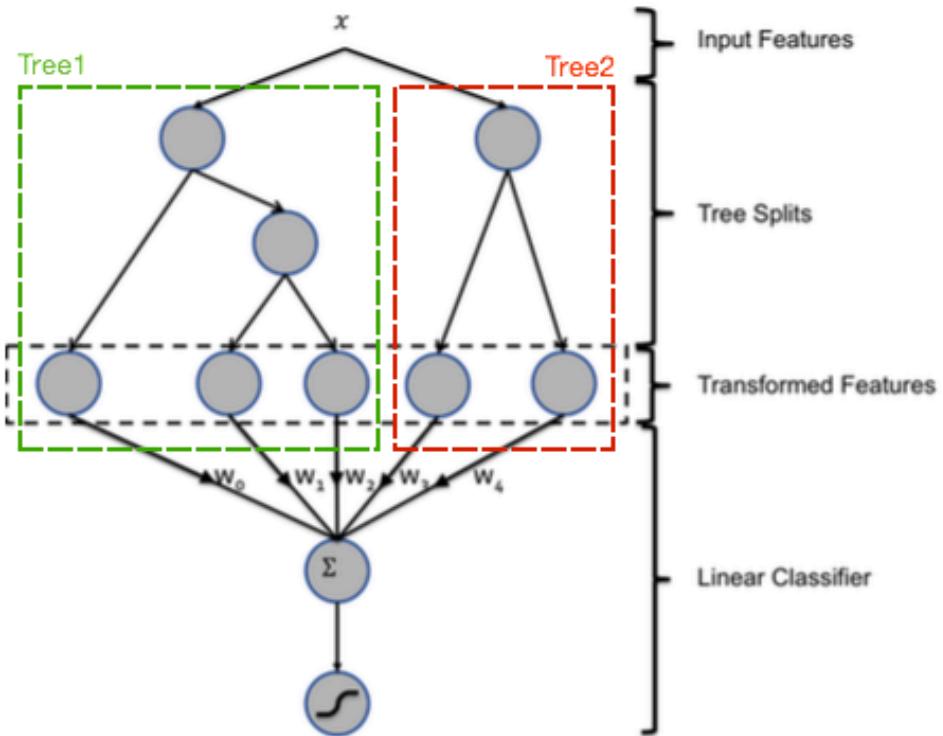


Figure 1: Hybrid model structure. Input features are transformed by means of boosted decision trees. The output of each individual tree is treated as a categorical input feature to a sparse linear classifier. Boosted decision trees prove to be very powerful feature transforms.

4、FM

推荐blog: <http://tracholar.github.io/machine-learning/2017/03/10/factorization-machine.html>

http://www.52caml.com/head_first_ml/ml-chapter9-factorization-family/

因子分解机 (Factorization Machines, FM) 由Steffen Rendle于2010年提出[7], paper地址如下: <https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf> FM主要目标是: 解决数据稀疏的情况下, 特征怎样组合的问题。FM有以下优点:

1. 可以在非常稀疏的数据中进行合理的参数估计
2. FM模型的时间复杂度是线性的

4.1 FM模型

在一般的线性模型中, 是各个特征独立考虑的, 没有考虑到特征与特征之间的相互关系。但实际上, 大量的特征之间是有关联的。一般的线性模型为:

$$y = w_0 + \sum_{j=1}^D w_j x_j \quad (4)$$

从上面的式子中看出，一般的线性模型没有考虑特征之间的关联。为了表述特征间的相关性，我们采用多项式模型。在多项式模型中，特征 x_i 与 x_j 的组合用 $x_i x_j$ 表示（注意之前我们用下标*i*表示样本索引，这里临时用于表示特征索引）。为了简单起见，我们讨论二阶多项式模型。

$$y = w_0 + \sum_{j=1}^D w_j x_j + \sum_{i=1}^D \sum_{j=i+1}^D w_{ij} x_i x_j \quad (4)$$

该多项式模型与线性模型相比，多了特征组合的部分，特征组合部分的参数有 $\frac{D(D-1)}{2}$ 个。如果特征非常稀疏且维度很高的话，模型复杂度和时间复杂度将大大增加。为了降低复杂度，对每一个特征，引入隐含向量 $\mathbf{v}_i = [v_{i,1}, v_{i,2}, \dots, v_{i,K}]^T$ ，模型修改如下：

$$y = w_0 + \sum_{j=1}^D w_j x_j + \sum_{i=1}^D \sum_{j=i+1}^D \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \quad (4)$$

以上就是FM模型的表达式。其中 K 是超参数，即隐含向量的维度，一般取30或40；
 $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{k=1}^K v_{i,k} v_{j,k}$ 。

显而易见，公式(48)是一个通用的拟合方程，可以采用不同的损失函数用于解决回归、二元分类等问题，比如可以采用MSE (Mean Square Error) 损失函数来求解回归问题，也可以采用Hinge/Cross-Entropy损失来求解分类问题。当然，在进行二元分类时，FM的输出需要经过sigmoid变换，这与Logistic回归是一样的。

FM应用场景	损失函数	说明
回归	均方误差 (MSE) 损失	Mean Square Error，与平方误差类似
二类分类	Hinge/Cross-Entropy损失	分类时，结果需要做sigmoid变换
排序	.	.

4.2 模型训练

直观上看，FM的复杂度是 $O(KD^2)$ 。但是，通过公式(48)的二次项进行化简，其复杂度可以优化到 $O(KD)$ 。因此FM可以在线性时间对新样本作出预测。

对FM的交叉项进行化简：

$$\begin{aligned}
& \sum_{i=1}^D \sum_{j=i+1}^D \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \\
&= \frac{1}{2} \sum_{i=1}^D \sum_{j=1}^D \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j - \frac{1}{2} \sum_{i=1}^D \langle \mathbf{v}_i, \mathbf{v}_i \rangle x_i x_j \\
&= \frac{1}{2} \left(\sum_{i=1}^D \sum_{j=1}^D \sum_{k=1}^K v_{i,k} v_{j,k} x_i x_j - \sum_{i=1}^D \sum_{k=1}^K v_{i,k} v_{i,k} x_i x_i \right) \\
&= \frac{1}{2} \sum_{k=1}^K \left(\left(\sum_{i=1}^D v_{i,k} x_i \right) \left(\sum_{j=1}^D v_{j,k} x_j \right) - \sum_{i=1}^D v_{i,k}^2 x_i^2 \right) \\
&= \frac{1}{2} \sum_{k=1}^K \left(\left(\sum_{i=1}^D v_{i,k} x_i \right)^2 - \sum_{i=1}^D v_{i,k}^2 x_i^2 \right) \\
&= \frac{1}{2} \sum_{k=1}^K \left(\left(\sum_{j=1}^D v_{j,k} x_j \right)^2 - \sum_{j=1}^D v_{j,k}^2 x_j^2 \right)
\end{aligned} \tag{4}$$

如果用随机梯度下降法学习模型参数。那么，模型各个参数的梯度如下：

$$\frac{\partial}{\partial \theta} y(\mathbf{x}) = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \quad (\text{常数项}) \\ x_j & \text{if } \theta \text{ is } w_j \quad (\text{线性项}) \\ x_j \sum_{i=1}^D v_{i,k} x_i - v_{j,k} x_j^2, & \text{if } \theta \text{ is } v_{j,k} \quad (\text{交叉项}) \end{cases} \tag{5}$$

由于 $\sum_{i=1}^D v_{i,k} x_i$ 只与 k 有关，在参数迭代过程中，只需要计算一次所有 k 的 $\sum_{i=1}^D v_{i,k} x_i$ ，就能够方便地得到所有 $v_{i,k}$ 的梯度。显然，计算所有 k 的 $\sum_{i=1}^D v_{i,k} x_i$ 的复杂度是 $O(KD)$ ；已知 $\sum_{i=1}^D v_{i,k} x_i$ 时，计算每个参数梯度的复杂度是 $O(D)$ ；得到梯度后，更新每个参数的复杂度是 $O(1)$ ；模型参数一共有 $KD + D + 1$ 个。因此，FM 参数训练的时间复杂度为 $O(KD)$ 。

综上可知，FM 算法可以在线性时间内完成模型训练、以及对新样本做出预测，所以说 FM 是一个非常高效的模型。

FM 模型的训练可以通过 SGD 或 FTRL 进行训练，模型参数为 $w_0, \mathbf{w}, \mathbf{V}$ ，其中 w_0 为截距项，是标量； \mathbf{w} 为 D 维向量， \mathbf{V} 为 $D * K$ 的矩阵。模型训练算法可参

见：<http://blog.csdn.net/itplus/article/details/40536025>。

论文作者提供 FM 工具包 libFM，链接为：<http://libfm.org/>。

另外还有 fastFM 可用：<https://github.com/ibayer/fastFM>。

5、FFM

http://blog.csdn.net/john_xyz/article/details/78933253

FFM 的论文[8]链接：<https://www.csie.ntu.edu.tw/~cjlin/papers/ffm.pdf>

FFM (Field-aware Factorization Machine) 最初的概念来自 Yu-Chin Juan (阮毓钦，毕业于中国台湾大学，现在美国 Criteo 工作) 及其比赛队友，是 FM 的升级版模型。

FFM通过引入field的概念，FFM把相同性质的特征归于同一个field。我们可以把同一个categorical特征经过one-hot编码生成的数值型特征都可以放在同一个field中。在FFM中，每一维特征 x_i ，针对其特征的每一种field f_j ，都会学习一个隐向量 \mathbf{v}_{i,f_j} 。因此，隐向量不仅与特征相关，也与field相关。这也是FFM中“Field-aware”的由来。

假设样本一共有 D 维特征， F 个field，那么FFM的二次项有 $D * F$ 个隐向量。而在FM模型中，每一维特征的隐向量只有1个。FM可以看作FFM的特例，是把所有特征都归属到一个field时的FFM模型。根据FFM的field敏感特性，可以导出其模型方程：

$$y = w_0 + \sum_{j=1}^D w_j x_j + \sum_{i=1}^D \sum_{j=i+1}^D \langle \mathbf{v}_{i,f_j}, \mathbf{v}_{j,f_i} \rangle x_i x_j \quad (5)$$

其中 f_j 是第 j 维特征所属的field。如果隐向量的长度为 K ，那么FFM的二次参数有 DK 个，远多于FM模型的 DK 个。此外，由于隐向量与field相关，FFM二次项并不能够化简，**时间复杂度是 $O(KD^2)$ **。需要注意的是由于FFM中的隐向量只需要学习特定的field，所以通常：
 $K_{FFM} << K_{FM}$ 。

下面以一个例子简单说明FFM的特征组合方式。输入记录如下

Clicked	Publisher(P)	Advertiser(A)	Gender(G)
Yes	ESPN	Nike	Male

对于FM模型来说，其交叉项为：

$$\phi_{FM}(\mathbf{v}, x) = \langle \mathbf{v}_{ESPN}, \mathbf{v}_{Nike} \rangle + \langle \mathbf{v}_{ESPN}, \mathbf{v}_{Male} \rangle + \langle \mathbf{v}_{Nike}, \mathbf{v}_{Male} \rangle \quad (5)$$

因为在FM中，每个特征只有一个隐向量，这个隐向量可以用来学习和其他特征之间的关系。

但是在FFM中，每一个特征有好几个隐向量，取决于其他特征的字段。在这个例子中，FFM的特征交叉项

$$\phi_{FFM}(\mathbf{v}, x) = \langle \mathbf{v}_{ESPN,A}, \mathbf{v}_{Nike,P} \rangle + \langle \mathbf{v}_{ESPN,G}, \mathbf{v}_{Male,P} \rangle + \langle \mathbf{v}_{Nike,G}, \mathbf{v}_{Male,A} \rangle \quad (5)$$

简单来讲，就是说在做隐向量的内积时，必须考虑到其他特征所属的字段。例如 $\langle \mathbf{v}_{ESPN,A}, \mathbf{v}_{Nike,P} \rangle$ 中，因为Nike在字段A中，所以ESPN的这个特征必须考虑到字段A，以区分其他字段；而 $\langle \mathbf{v}_{ESPN,G}, \mathbf{v}_{Male,P} \rangle$ 中因为其交叉的特征Male属于字段G，所以使用了VESPN,G这个隐向量。这样，每个特征都有 F 个隐向量。

论文作者提供FFM的工具包libffm：<https://github.com/guestwalk/libffm>。

6、总结与展望

点击率预估模型涉及的训练样本一般是上亿级别，样本量大，模型常采用速度较快的LR。但LR是线性模型，学习能力有限，此时特征工程尤其重要。现有的特征工程实验，主要集中在寻找到有区分度的特征和特征组合。GBDT算法的特点正好可以用来发掘有区分度的特征、特征组合，FM和FFM则通过隐含向量找到特征组合的低维表示，这些方法能有效减少特征工程中人力成本。另外深度学习也学习到有效的特征表示。

参考文献

- [1] H. Brendan McMahan & M Streter. Adaptive Bound Optimization for Online Convex Optimization. In COLT, 2010
- [2] H. Brendan McMahan. Follow-the-Regularized-Leader and Mirror Descent: Equivalence Theorems and L1 Regularization. In AISTATS, 2011
- [3] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, Jeremy Kubica, Ad Click Prediction: a View from the Trenches. In ACM SIGKDD, 2013
- [4].He X, Pan J, Jin O, et al. Practical lessons from predicting clicks on ads at facebook[C]. Proceedings of 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, 2014: 1-9.
- [5].<http://www.csie.ntu.edu.tw/~r01922136/Kaggle-2014-criteo.pdf>
- [6].<https://github.com/guestwalk/Kaggle-2014-criteo>
- [7] Steffen Rendle, Factorization Machines, Proceedings of the 10th IEEE International Conference on Data Mining (ICDM 2010), Sydney, Australia.
- [8] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, Chih-Jen Lin. Field-aware Factorization Machines for CTR Prediction, Proceedings of the 10th ACM Conference on Recommender Systems, 2016
- [9] 冯扬, 在线最优化求解
[https://github.com/wzhe06/Ad-papers/blob/master/Optimization%20Method/%E5%9C%A8%E7%BA%BF%E6%9C%80%E4%BC%98%E5%8C%96%E6%B1%82%E8%A7%A3\(Online%20Optimization\)-%E5%86%AF%E6%89%AC.pdf](https://github.com/wzhe06/Ad-papers/blob/master/Optimization%20Method/%E5%9C%A8%E7%BA%BF%E6%9C%80%E4%BC%98%E5%8C%96%E6%B1%82%E8%A7%A3(Online%20Optimization)-%E5%86%AF%E6%89%AC.pdf)
- [10] John Langford, Lihong Li & Tong Zhang. Sparse Online Learning via Truncated Gradient. Journal of MachineLearning Research, 2009
- [11] John Duchi & Yoram Singer. Efficient Online and Batch Learning using Forward Backward Splitting. Journal of Machine Learning Research, 2009
- [12] Lin Xiao. Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization. Journal of Machine Learning Research, 2010