FTRL算法学习总结（1）-jinshang(尚晋)

在使用机器学习进行广告推送预测（如某用户购买某商品的可能性，点击某广告的可能性）时，逻辑回归（logistic regression）是一种常用的学习方法。本文介绍了Google提出的FTRL（Follow-the-regularized-leader）算法，可以在线地处理逻辑回归等凸优化问题，兼顾稀疏性（sparsity）和准确性（accuracy）。

本文第一部分介绍算法本身，第二部分介绍该算法的工程实现。

一、相关背景

（1）数学表示

出于简洁的考虑，本文采用一些简化的数学表示方法。粗体字母如$\mathbf{g} \in \mathbb{R}^d$表示一个向量（vector），$g_{t,i}$表示向量$\mathbf{g}$的第$i$个值，$\mathbf{g}_{1:t}$表示求和,即$\sum_{s=1}^{t} \mathbf{g}_s$。

（2）离线学习与在线学习

离线算法在每次迭代对全体训练数据集进行梯度计算，虽然计算精度高，结果准确，但是由于广告点击数据量巨大，且增长速度快，在每次迭代都进行全局计算的代价太大，因此不适合对数据流进行计算。

在线学习算法对每个新加入的训练实例进行单独计算。对每个的特征向量$\mathbf{x}$，计算出它的分类，并根据其正确的分类给出一个损失，并对分类器模型进行更新，目标是使整个训练过程中的总损失最小。

（3）广告投放问题实例

当用户搜索条目$\mathbf{q}$时，给出一个广告的集合A，在决定是否投放广告$\mathbf{a}$时，我们需要预测用户点击广告的概率$P(\text{click}|\mathbf{q},\mathbf{a})$。因此，搜索条目$\mathbf{q}$和广告$\mathbf{a}$共同够了我们的输入特征向量，其特征包括：搜索条目文字，广告的文字等其他广告的元数据。

（4）逻辑回归模型

在第t轮迭代，我们根据特征向量$\mathbf{x} \in \mathbb{R}^d$和模型参数$\mathbf{w}_t$,我们做出预测$p_t = \sigma(\mathbf{w}_t \cdot \mathbf{x}_t)$,其中$\sigma(a) = 1/(1+exp(-a))$被称作sigmoid函数。根据标签$y_t \in \{0,1\}$，计算出逻辑损失（LogLoss）

$$\ell_t\left(\mathbf{w}_t\right) = -y_t \log p_t - \left(1-y_t\right) \log(1-p_t) \tag{1}$$

我们可以计算出其梯度为$\nabla \ell_t(\mathbf{w}) = \left(\sigma\left(\mathbf{w} \cdot \mathbf{x}_t\right) - y_t\right) \mathbf{x}_t = \left(p_t - y_t\right) \mathbf{x}_t$,我们根据这个梯度对模型进行优化。

（5）OGD\SGD算法（Online\Stochastic Gradient Descent）

OGD即对逻辑回归问题最基础的在线学习算法，对每个新数据，根据其梯度进行优化,即$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t$。该算法的准确度非常高，但是稀疏性并不好，即$\mathbf{w}$的非零参数非常多，导致模型占用大量的内存。为了解决这个问题，产生了几个可以提供稀疏解的算法。其中最简单的就是在梯度中加入正则项L1范数$\lambda||\mathbf{w}||_1$,加入该正则项后我们使参数的L1范数最小化，也就是最接近零。但是由于计算机的浮点数表示，我们很难得到真正的零，因此还需要更多的途径来产生稀疏解。

（6）简单截断和Truncated Gradient 算法

其中最简单最直观的一种方法就是设定一个阈值，低于这个阈值的参数设定为零，我们称之为简单截断法。但是这个方法有两个问题，一是参数值小可能是这个参数才被更新一次（因为训练刚开始或者是训练数据集比较小），二是这种直接的截断方式会破坏算法的理论，影响训练效果。在此基础上，还有不那么粗暴的TG算法：给出一个缓冲值a，当参数$x_{t,i} < \theta$时，$x_{t+1,i} = \max\{0, x_{t,i} - a\}$。

（7）FOBOS算法 （Forward Backward Splitting）

FOBOS算法将权重更新分为两个步骤

$$\begin{aligned} \mathbf{w}^{(t+0.5)} &= \mathbf{w}^{(t)} - \eta^{(t)} G^{(t)} \\ \mathbf{w}^{(t+1)} &= \operatorname{argmin}_{\mathbf{w}} \left\{ \frac{1}{2} \left\| \mathbf{w} - \mathbf{w}^{(t+0.5)} \right\|_2^2 + \eta^{(t+0.5)} \Psi(\mathbf{w}) \right\} \end{aligned} \tag{2}$$

第一个步骤就是普通的梯度下降，第二个是根据第一个步骤进行局部调整，第一项保证距离梯度下降的解不能太远，第二项是一个正则化。如果正则化项$\Psi(\mathbf{w}) = ||\mathbf{w}||_1$，即L1范数，则我们可以对每个维度分别求解。通过数学计算我们得到L1-FOBOS的更新算法：

$$w_i = \operatorname{sgn}\left(w_i^{(t)} - \eta^{(t)} g_i^{(t)}\right) \max\left\{0, \left|w_i^{(t)} - \eta^{(t)} g_i^{(t)}\right| - \eta^{(t+0.5)} \lambda\right\} \text{ for all } 1 \le i \le N \tag{3}$$

不难看出，L1-FOBOS也是截断法的一种特定形式，当本轮迭代产生的梯度值$\left|w_i^{(t)} - \eta^{(t)} g_i^{(t)}\right|$低于阈值时将其截断。

（8）RDA算法 （Regularized Dual Averaging）

前面的几个算法本质上都是根据每轮的梯度进行更新，没有将历史梯度考虑在内。它们的优势是算法的精度较高，同时也产生了一定的稀疏性。RDA算法则考虑每个权重的历史梯度平均值，在提供稀疏性的同时避免了因为刚开始训练和数据集小而被截断的问题。RDA的更新公式是：

$$\mathbf{w}^{(t+1)} = \operatorname{argmin}_{\mathbf{w}} \left\{ \frac{1}{t} \sum_{r=1}^{t} G^{(r)} \cdot \mathbf{w} + \Psi(\mathbf{w}) + \frac{\beta^{(t)}}{t} h(\mathbf{w}) \right\} \tag{4}$$

其中第一项是历史梯度平均值，第二项是正则项，第三项是一个辅助正则项，为一个非递减非负序列和一个严格凸函数的乘积。

我们考虑L1正则的情况，并令$h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2, \beta^{(t)} = \gamma\sqrt{t}$，则各项权重的更新被转化成一个优化问题：

$$w_i^{t+1} = \operatorname{argmin}_{w_i} \left\{ \overline{g}_i^{(t)} + \lambda \left|w_i\right| + \frac{\gamma}{2\sqrt{t}} w_i^2 \right\} \tag{5}$$

经过数学推导我们可以得出：

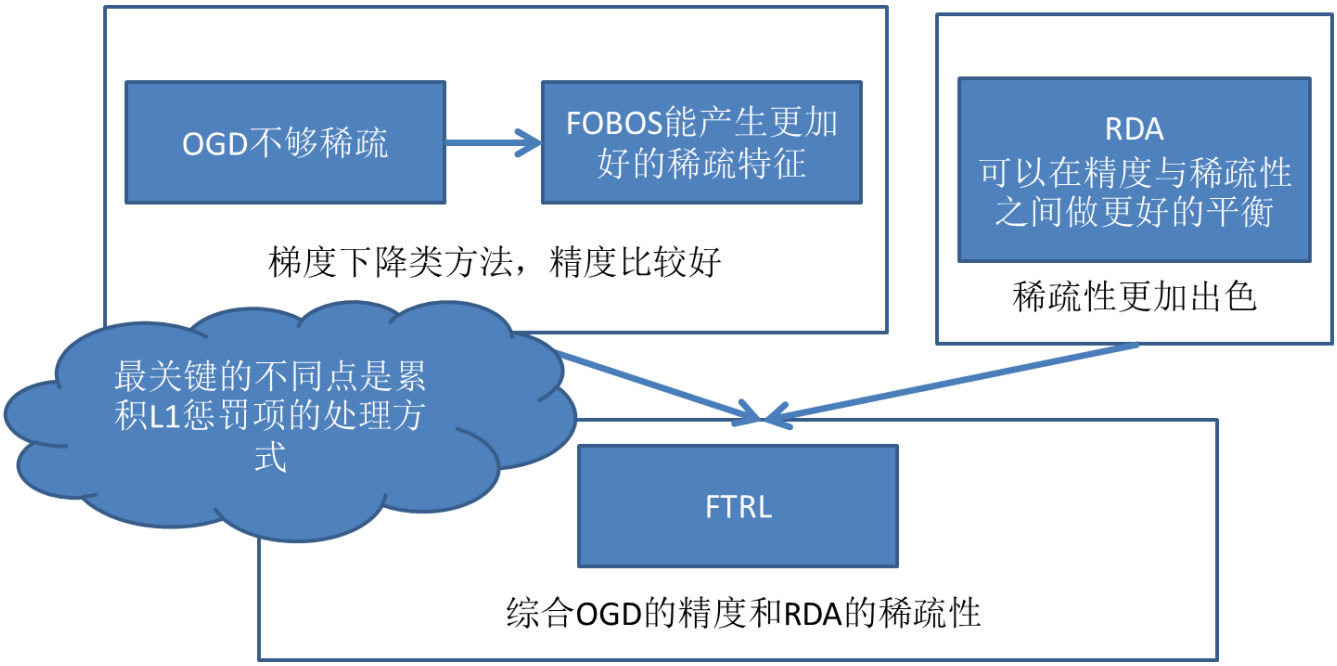
$$\begin{aligned} w_i^{(t+1)} &= 0, \text{ if } \left|\overline{g}_i^{(t)}\right| < \lambda, \\ w_i^{(t+1)} &= -\frac{\sqrt{t}}{\gamma}\left(\overline{g}_i^{(t)} - \lambda \cdot \operatorname{sgn}\left(\overline{g}_i^{(t)}\right)\right), \text{ otherwise.} \end{aligned} \tag{6}$$

从公式可以看出，当一个参数的历史平均低于λ时，我们对其进行截断。相比之前的方法，RDA产生稀疏解的方法更为合理。我们可以通过调整λ来控制精度和稀疏性的平衡。

与FOBOS相比，RDA主要有两点不同：（1）RDA考虑历史累加梯度和L1正则项而FOBOS只考虑当前迭代，（2）RDA限制新权重不能离0太远，而FOBOS限制新权重不能离老权重太远。

二、FTRL算法

（1）基本思想



FTRL的基本思想就是结合两者的优点，考虑历史累加梯度和L1正则项的同时限制新权重不能离老权重太远。因此FTRL的迭代公式为

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left(\mathbf{g}_{1:t} \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s \|\mathbf{w} - \mathbf{w}_s\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 \right) \tag{7}$$

令$\mathbf{z}_{t-1} = \mathbf{g}_{1:t-1} - \sum_{s=1}^{t-1} \sigma_s \mathbf{w}_s$可得最优解，

$$w_{t+1,i} = \begin{cases} 0 & \text{if } |z_{t,i}| \le \lambda_1 \\ -\eta_t \left(z_{t,i} - \text{sgn}(z_{t,i}) \lambda_1 \right) & \text{otherwise.} \end{cases} \tag{8}$$

（2）算法实现

Google在[2]中给出了FTRL算法的伪代码实现，相较于原始公式进行了一些修改，比如加入了L2正则项，其具体的工程实现和细节将在本文下半部分中介绍。

Algorithm 1 Per-Coordinate FTRL-Proximal with L_1 and L_2 Regularization for Logistic Regression

#With per-coordinate learning rates of Eq. (2).
Input: parameters α, β, λ_1, λ_2
$(\forall i \in \{1, \dots, d\})$, initialize $z_i = 0$ and $n_i = 0$
for $t = 1$ **to** T **do**
  Receive feature vector $\mathbf{x}_t$ and let $I = \{i \mid x_i \neq 0\}$
  For $i \in I$ compute

$$w_{t,i} = \begin{cases} 0 & \text{if } |z_i| \leq \lambda_1 \\ -\left(\frac{\beta + \sqrt{n_i}}{\alpha} + \lambda_2\right)^{-1} (z_i - \text{sgn}(z_i)\lambda_1) & \text{otherwise.} \end{cases}$$

  Predict $p_t = \sigma(\mathbf{x}_t \cdot \mathbf{w})$ using the $w_{t,i}$ computed above
  Observe label $y_t \in \{0, 1\}$
  for all $i \in I$ **do**
    $g_i = (p_t - y_t)x_i$ *#gradient of loss w.r.t. w_i*
    $\sigma_i = \frac{1}{\alpha}\left(\sqrt{n_i + g_i^2} - \sqrt{n_i}\right)$ *#equals $\frac{1}{\eta_{t,i}} - \frac{1}{\eta_{t-1,i}}$*
    $z_i \leftarrow z_i + g_i - \sigma_i w_{t,i}$
    $n_i \leftarrow n_i + g_i^2$
  end for
end for

参考资料

1. 各大公司广泛使用的在线学习算法详解

2. Ad Click Prediction: a View from the Trenches (Google关于FTRL技术实现的论文)

3. Follow-the-Regularized-Leader and Mirror Descent: Equivalence Theorem and L1 Regularization （Google关于FTRL与前面几个算法比较的论文）

4. 在线最优化求解