在线学习

2018.09.25 王珏

场景

在线学习框架，每一次迭代，环境给出一个样本，模型作出预测并产生loss，更新样本
广告场景：用户请求->模型预估->产生loss(用户是否点击)->更新模型
业务实践：用户请求->模型预估->等待一天(或小时级)->产生loss(用户是否点击)->更新模型

For $t = 1, \ldots, T$

- Player chooses $w_t \in \mathcal{W}$, where $\mathcal{W}$ is a *convex* set in $\mathbb{R}^n$.
- Environment chooses a *convex* loss function $f_t : \mathcal{W} \to \mathbb{R}$.
- Player incurs a loss $\ell_t = f_t(w_t) = f_t(w_t; (x_t, y_t))$.
- Player receives feedback f_t.

在线学习算法评估

$$R(T) = \sum_{t=1}^{T} f_t(w_t) - \min_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w).$$

解释：评估后悔程度，w为全局最优解，w_t为每次更新的权重，R(T)表示了在线学习和批量学习的gap。当R(T)随T增长低于线性时，效果随迭代而增加.
分析方法： 对于在线学习算法，证明其R(T)<=a，得到效果下界。

举例：
最简单的在线学习算法，在线梯度下降，R(T)分析略。

Algorithm 2. Stochastic Gradient Descent

Loop {

for j=1 to M {

$$W^{(t+1)} = W^{(t)} - \eta^{(t)} \nabla_W \ell\left(W^{(t)}, Z_j\right)$$

}

}

Algorithm 1. Batch Gradient Descent

Repeat until convergence {

$$W^{(t+1)} = W^{(t)} - \eta^{(t)} \nabla_W \ell\left(W^{(t)}, Z\right)$$

}

关于稀疏性

一般是通过约束区域和损失函数等高线，证明l1比l2更容易获得稀疏解。
另外一种解释： l2在靠近0的附近变化变小，而l1变化量不变。

Consider the vector $\vec{x} = (1, \varepsilon) \in \mathbb{R}^2$ where $\varepsilon > 0$ is small. The l_1 and l_2 norms of $\vec{x}$, respectively, are given by

$$||\vec{x}||_1 = 1 + \varepsilon, \quad ||\vec{x}||_2^2 = 1 + \varepsilon^2$$

Now say that, as part of some regularization procedure, we are going to reduce the magnitude of one of the elements of $\vec{x}$ by $\delta \leq \varepsilon$. If we change x_1 to $1 - \delta$, the resulting norms are

$$||\vec{x} - (\delta, 0)||_1 = 1 - \delta + \varepsilon, \quad ||\vec{x} - (\delta, 0)||_2^2 = 1 - 2\delta + \delta^2 + \varepsilon^2$$

On the other hand, reducing x_2 by δ gives norms

$$||\vec{x} - (0, \delta)||_1 = 1 - \delta + \varepsilon, \quad ||\vec{x} - (0, \delta)||_2^2 = 1 - 2\varepsilon\delta + \delta^2 + \varepsilon^2$$

The thing to notice here is that, for an l_2 penalty, regularizing the larger term x_1 results in a much greater reduction in norm than doing so to the smaller term $x_2 \approx 0$. For the l_1 penalty, however, the reduction is the same. Thus, when penalizing a model using the l_2 norm, it is highly unlikely that anything will ever be set to zero, since the reduction in l_2 norm going from ε to 0 is almost nonexistent when ε is small. On the other hand, the reduction in l_1 norm is always equal to δ, regardless of the quantity being penalized.

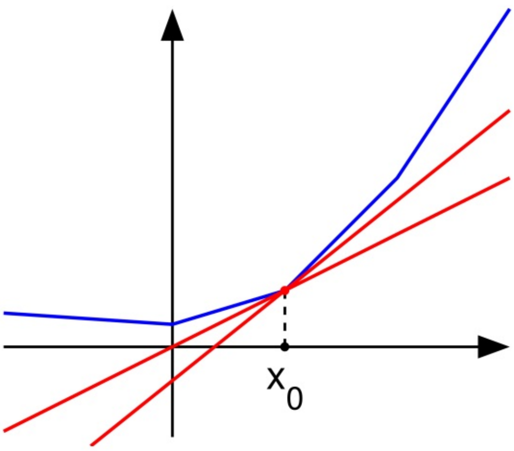
Another way to think of it: it's not so much that l_1 penalties encourage sparsity, but that l_2 penalties in some sense **discourage** sparsity by yielding diminishing returns as elements are moved closer to zero.

优化问题没有解析解时，即使使用l1也很难获得稀疏解。l1只能保证快速逼近到0附近。在线学习只使用l1不容易获得稀疏解。需要额外手段来保证稀疏。

在线梯度下降l1正则化

$$W^{(t+1)} = W^{(t)} - \eta^{(t)} G^{(t)} - \eta^{(t)} \lambda \operatorname{sgn}(W^{(t)})$$

次梯度



简单截断法

以k为窗口，当t/k不为整数时采用标准的 SGD 进行迭代，当t/k为整数时，采用如下权重更新方式：

$$W^{(t+1)} = T_0\left(W^{(t)} - \eta^{(t)} G^{(t)}, \theta\right)$$

$$T_0(v_i, \theta) = \begin{cases} 0 & if\ |v_i| \leq \theta \\ v_i & otherwise \end{cases} \tag{3-1-2}$$

注意，这里面$\theta \in \mathbb{R}$是一个标量，且$\theta \geq 0$；如果$V = [v_1, v_2, \ldots, v_N] \in \mathbb{R}^N$是一个向量，$v_i$是向量的一个维度，那么有$T_0(V, \theta) = [T_0(v_1, \theta), T_0(v_2, \theta), \ldots, T_0(v_N, \theta)] \in \mathbb{R}^N$。

直观的获得稀疏性的方法，当权重小于阈值时设置为0

截断梯度法

$$W^{(t+1)} = T_1\left(W^{(t)} - \eta^{(t)} G^{(t)}, \eta^{(t)} \lambda^{(t)}, \theta\right)$$

$$T_1(v_i, \alpha, \theta) = \begin{cases} \max(0, v_i - \alpha) & if\ v_i \in [0, \theta] \\ \min(0, v_i + \alpha) & if\ v_i \in [-\theta, 0] \\ v_i & otherwise \end{cases} \tag{3-1-3}$$

Algorithm 3. Truncated Gradient

input θ

initial $W \in \mathbb{R}^N$

for $t = 1,2,3\ldots$ *do*

$G = \nabla_W \ell\left(W, X^{(t)}, y^{(t)}\right)$

refresh W *according to*

$$w_i = \begin{cases} \max(0, w_i - \eta^{(t)} g_i - \eta^{(t)} \lambda^{(t)}) & if \left(w_i - \eta^{(t)} g_i\right) \in [0, \theta] \\ \max(0, w_i - \eta^{(t)} g_i + \eta^{(t)} \lambda^{(t)}) & if \left(w_i - \eta^{(t)} g_i\right) \in [-\theta, 0] \\ w_i - \eta^{(t)} g_i & otherwise \end{cases}$$

end

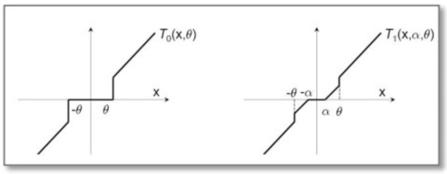
return W

l1正则化和简单截断法都是梯度截断的特殊形式。

$$w_i^{(t+1)} = \begin{cases} Trnc\left(\left(w_i^{(t)} - \eta^{(t)} g_i^{(t)}\right), \lambda_{TG}^{(t)}, \theta\right) & if\ mod(t,k) = 0 \\ w_i^{(t)} - \eta^{(t)} g_i^{(t)} & otherwise \end{cases}$$

$$\lambda_{TG}^{(t)} = \eta^{(t)} \lambda k \qquad (3\text{-}1\text{-}4)$$

$$Trnc\left(w, \lambda_{TG}^{(t)}, \theta\right) = \begin{cases} 0 & if\ |w| \le \lambda_{TG}^{(t)} \\ w - \lambda_{TG}^{(t)} sgn(w) & if\ \lambda_{TG}^{(t)} \le |w| \le \theta \\ w & otherwise \end{cases}$$



FOBOS

权重更新分两个步骤：

$$W^{(t+\frac{1}{2})} = W^{(t)} - \eta^{(t)} G^{(t)}$$

$$W^{(t+1)} = \underset{W}{argmin}\left\{\frac{1}{2}\left\|W - W^{(t+\frac{1}{2})}\right\|^2 + \eta^{(t+\frac{1}{2})}\Psi(W)\right\}$$

第一步为梯度下降，第二步在第一步梯度下降得到权重附近加入正则进行最优化。
合并公式有：

$$W^{(t+1)} = \underset{W}{argmin}\left\{\frac{1}{2}\left\|W - W^{(t)} + \eta^{(t)} G^{(t)}\right\|^2 + \eta^{(t+\frac{1}{2})}\Psi(W)\right\}$$

考虑正则为l1的FOBOS：

$$W^{(t+1)} = \underset{W}{argmin} \sum_{i=1}^{N}\left(\frac{1}{2}(w_i - v_i)^2 + \tilde{\lambda}|w_i|\right)$$

使用技巧求解，得到

$$w_i^{(t+1)} = \underset{w_i}{argmin}\left(\frac{1}{2}(w_i - v_i)^2 + \tilde{\lambda}|w_i|\right)$$

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

反证法：
假设：$w_i^* v_i < 0$，那么有：

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

这与w_i^*是$\text{minimize}_{w_i}\left(\frac{1}{2}(w_i - v_i)^2 + \tilde{\lambda}|w_i|\right)$的最优解矛盾，故假设不成立，$w_i^* v_i \geq 0$

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

(1) 当$v_i \geq 0$ 时：

由于$w_i^* v_i \geq 0$，所以$w_i^* \geq 0$，相当于对$\text{minimize}_{w_i}\left(\frac{1}{2}(w_i - v_i)^2 + \tilde{\lambda}|w_i|\right)$ 引入了不等式约束条件 $-w_i \leq 0$；

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：$\frac{\partial}{\partial w_i}\left(\frac{1}{2}(w_i - v_i)^2 + \tilde{\lambda}w_i - \beta w_i\right)\Big|_{w_i = w_i^*} = 0$ 以及 $\beta w_i^* = 0$；

根据上面的求导等式可得：$w_i^* = v_i - \tilde{\lambda} + \beta$；

分为两种情况：

① $w_i^* > 0$ ：

由于 $\beta w_i^* = 0$ 所以 $\beta = 0$

这时候有：$w_i^* = v_i - \tilde{\lambda}$

又由于$w_i^* > 0$，所以$v_i - \tilde{\lambda} > 0$

② $w_i^* = 0$ ：

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

又由于$\beta \geq 0$，所以$v_i - \tilde{\lambda} \leq 0$

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

(2) 当$v_i < 0$ 时：

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

综合上面的分析，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的方式为：

$$\begin{aligned} w_i^{(t+1)} &= sgn(v_i)max(0, |v_i| - \tilde{\lambda}) \\ &= sgn\left(w_i^{(t)} - \eta^{(t)}g_i^{(t)}\right)max\left\{0, \left|w_i^{(t)} - \eta^{(t)}g_i^{(t)}\right| - \eta^{(t+\frac{1}{2})}\lambda\right\} \end{aligned} \quad (3\text{-}2\text{-}3)$$

$$\begin{aligned} w_i^{(t+1)} &= sgn(v_i)max(0, |v_i| - \tilde{\lambda}) \\ &= sgn\left(w_i^{(t)} - \eta^{(t)}g_i^{(t)}\right)max\left\{0, \left|w_i^{(t)} - \eta^{(t)}g_i^{(t)}\right| - \eta^{(t+\frac{1}{2})}\lambda\right\} \end{aligned}$$

写成梯度截断的形式：

$$w_i^{(t+1)} = \begin{cases} 0 & if\ \left|w_i^{(t)} - \eta^{(t)}g_i^{(t)}\right| \leq \eta^{(t+\frac{1}{2})}\lambda \\ \left(w_i^{(t)} - \eta^{(t)}g_i^{(t)}\right) - \eta^{(t+\frac{1}{2})}\lambda sgn\left(w_i^{(t)} - \eta^{(t)}g_i^{(t)}\right) & otherwise \end{cases}$$

显式的获取稀疏性。

FTL

Follow-The-Leader

w_1 is set arbitrarily
for $t = 1, 2, \ldots, T$
$\quad w_t = \underset{w \in \mathcal{W}}{\operatorname{argmin}} \sum_{s=1}^{t-1} f_s(w)$

在线学习中每一轮将w更新为在全部历史轮数上的最优解。

FTRL

$$w_{t+1} = \underset{w \in W}{\operatorname{argmin}}(f_{1:t}(w) + R(w)).$$

FTRL-Proximal

考虑l1-FOBOS的更新公式：

$$x^{(t+1)} = argmin_x\{(x - (x^{(t)} - \eta \nabla l(x^{(t)})))^2 + \lambda|x|\}$$

展开并忽略其中一个常数项 $(\eta \nabla l(x^{(t)}))^2$ 后得到等价的形式

$$x^{(t+1)} = argmin_x\{2\eta \nabla l(x^{(t)})(x - x^{(t)}) + (x - x^{(t)})^2 + \lambda|x|\}$$

定义FTRL损失函数
$f_(t+1)(x) = g_t(x - x_t) + (1/n_t - 1/n_{t-1})(x - x_t)^2$

套用FTRL优化方法：
$argmin \sum f_t + R$
得到FTRL的优化函数

$$W^{(t+1)} = \underset{W}{argmin}\left\{G^{(1:t)} \cdot W + \lambda_1\|W\|_1 + \lambda_2\frac{1}{2}\|W\|_2^2 + \frac{1}{2}\sum_{s=1}^{t}\sigma^{(s)}\left\|W - W^{(s)}\right\|_2^2\right\}$$

将最后一项展开，可以得到

$$W^{(t+1)} = \underset{W}{argmin}\left\{\left(G^{(1:t)} - \sum_{s=1}^{t}\sigma^{(s)}W^{(s)}\right) \cdot W + \lambda_1\|W\|_1 + \frac{1}{2}\left(\lambda_2 + \sum_{s=1}^{t}\sigma^{(s)}\right)\|W\|_2^2 + \frac{1}{2}\sum_{s=1}^{t}\sigma^{(s)}\left\|W^{(s)}\right\|_2^2\right\}$$

由于$\frac{1}{2}\sum_{s=1}^{t}\sigma^{(s)}\left\|W^{(s)}\right\|_2^2$相对于$W$来说是一个常数，并且令$Z^{(t)} = G^{(1:t)} - \sum_{s=1}^{t}\sigma^{(s)}W^{(s)}$，上式等价于：

$$W^{(t+1)} = \underset{W}{argmin}\left\{Z^{(t)} \cdot W + \lambda_1\|W\|_1 + \frac{1}{2}\left(\lambda_2 + \sum_{s=1}^{t}\sigma^{(s)}\right)\|W\|_2^2\right\}$$

解法如下：

首先仅考虑第 i 维即 $w_i^{(t+1)}$ ，$\vec{z}^{(t)}\vec{w} + \frac{1}{2\eta_t}||\vec{w}||_2^2 + \lambda||\vec{w}||_1$ 对于 $w_i^{(t+1)}$ 的次梯度为集合 $\{z_i^{(t)} + \frac{1}{\eta_t}w_i + \lambda\partial r(w_i)\}$ ，其中 $r(w_i) = |w_i|$

当 $w_i^{(t+1)}$ 为最优解时，$0 \in \{z_i^{(t)} + \frac{1}{\eta_t}w_i + \lambda\partial r(w_i)\}$ ，即存在 $g_r(w_i^{(t+1)}) \in \partial r(w_i^{(t+1)})$ 满足 $0 = z_i^{(t)} + \frac{1}{\eta_t}w_i^{(t+1)} + \lambda g_r(w^{(t+1)})$ ，即 $g_r(w^{(t+1)}) = -\frac{1}{\lambda}(z_i^{(t)} + \frac{1}{\eta_t}w_i^{(t+1)})$.

根据 $g_r(w_i^{(t+1)}) \in \partial r(w_i^{(t+1)})$ 在 $w_i^{(t+1)}$ 大于0时取值1、小于0时取值-1、等于0时取值 $[-1,1]$ ，得到以下3种情况：

- $w_i^{(t+1)} = 0$ 且 $g_r(w^{(t+1)}) = -\frac{1}{\lambda}(z_i^{(t)} + \frac{1}{\eta_t}w_i^{(t+1)}) \in [-1,1]$
- $w_i^{(t+1)} > 0$ 且 $g_r(w^{(t+1)}) = -\frac{1}{\lambda}(z_i^{(t)} + \frac{1}{\eta_t}w_i^{(t+1)}) = 1$
- $w_i^{(t+1)} < 0$ 且 $g_r(w^{(t+1)}) = -\frac{1}{\lambda}(z_i^{(t)} + \frac{1}{\eta_t}w_i^{(t+1)}) = -1$

整理后就得到这一轮的解析解：

$w_i^{(t+1)} = 0$ ，当 $|z_i^{(t)}| < \lambda$

$w_i^{(t+1)} = -\eta_t(z_i^{(t)} - \lambda)$ ，当 $z_i^{(t)} > \lambda$

$w_i^{(t+1)} = -\eta_t(z_i^{(t)} + \lambda)$ ，当 $z_i^{(t)} < -\lambda$

结果：

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

FTRL学习率

FTRL的学习率在每维特征上单独更新

$$\eta_i^{(t)} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^{t}\left(g_i^{(s)}\right)^2}}$$

由于$\sigma^{(1:t)} = \frac{1}{\eta^{(t)}}$，所以公式(3-4-4)中$\sum_{s=1}^{t}\sigma^{(s)} = \frac{1}{\eta_i^{(t)}} = \left(\beta + \sqrt{\sum_{s=1}^{t}\left(g_i^{(s)}\right)^2}\right.$

和β是需要输入的参数，(3-4-4)中学习率写成累加的形式，是为了方便理解后面

计算逻辑。

保证了在训练一定轮数之后，新的有意义的特征也能对模型起到影响，同时避免训练较多的特征振荡。

FTRL算法逻辑

Algorithm 6. FTRL-Proximal with L1 & L2 Regularization

input $\alpha,\ \beta,\ \lambda_1,\ \lambda_2$

initialize $W \in \mathbb{R}^N,\ Z = 0 \in \mathbb{R}^N,\ Q = 0 \in \mathbb{R}^N$

for t =1,2,3... do

  $G = \nabla_W \ell\left(W, X^{(t)}, y^{(t)}\right)$ *# gradient of loss function*

  for i in 1,2,...,N do *# for each coordinate*

$$\sigma_i = \frac{1}{\alpha}\sqrt{q_i + g_i^2} - \sqrt{q_i}\ \ \&\ \ q_i = q_i + g_i^2 \quad \text{\# equals } \frac{1}{\eta^{(t)}} - \frac{1}{\eta^{(t-1)}}$$

$$z_i = z_i + g_i - \sigma_i w_i$$

$$w_i = \begin{cases} 0 & \text{if } \left|z_i^{(t)}\right| < \lambda_1 \\ -\left(\lambda_2 + \frac{\beta + \sqrt{q_i}}{\alpha}\right)^{-1} (z_i - \lambda_1 sgn(z_i)) & otherwise \end{cases}$$

  end

end

return W

组内代码

```cpp
void FTRL::train(const std::vector<pair<std::string, double>> &fea, int label)
{
    std::vector<ModelUnit *> tempvec(fea.size(), NULL);
    double p = 0.0;
    for (int i = 0; i < fea.size(); ++i)
    {
        const std::string &index = fea[i].first;
        tempvec[i] = WGSZN->getOrInitDB(index);
        ModelUnit &modelUnit = *(tempvec[i]);
        modelUnit.mtx.lock();
        if (fabs(modelUnit.z) <= lambda1)
        {
            modelUnit.w = 0.0;
        }
        else
        {
            modelUnit.w = (-1) *
                          (1 / (lambda2 + (beta + sqrt(modelUnit.n)) / alpha)) *
                          (modelUnit.z - utils::sgn(modelUnit.z) * lambda1);
        }
        p += modelUnit.w * fea[i].second;
        modelUnit.mtx.unlock();
    }
    p = utils::sigmoid(p);
    for (int i = 0; i < fea.size(); ++i)
    {
        ModelUnit &modelUnit = *(tempvec[i]);
        modelUnit.mtx.lock();
        modelUnit.g = (p - label) * (fea[i].second);
        modelUnit.s = 1 / alpha * (sqrt(modelUnit.n + modelUnit.g * modelUnit.g) - sqrt(modelUnit.n));
        modelUnit.z += modelUnit.g - modelUnit.s * modelUnit.w;
        modelUnit.n += modelUnit.g * modelUnit.g;
        modelUnit.mtx.unlock();
    }
}
```