优化方法

张朝阳

腾讯课堂

2016 年 12 月 1 日

提纲

- 梯度下降及牛顿法
- 拉格朗日乘子法及KKT条件
- 并行随机梯度下降法
- 对梯度下降法的优化
- 高维特征稀疏性的产生方法

点击前往视频课程　　点击前往我的博客

优化问题

如果$f(X)$是定义在N维向量空间上的实值函数，对于在$f(X)$的定义域C 上的任意两个点X_1和X_2，以及任意[0,1] 之间的值λ都有：

$$f(\lambda X_1 + (1-\lambda)X_2) \leq \lambda f(X_1) + (1-\lambda)f(X_2)$$

$$\forall X_1, X_2 \in C, \quad 0 \leq t \leq 1$$

那么称$f(X)$是凸函数。一个函数是凸函数是它存在最优解的充要条件。

无约束最优化问题：$\arg\min_x f(X)$

带约束的一般形式：
$$\begin{cases} h_i(X) = 0 & i = 1, 2, \cdots, m \\ g_j(X) \leq 0 & j = 1, 2, \cdots, n \end{cases}$$

梯度下降法

泰勒级数：$f(x+\Delta x)=f(x)+f'(x)\Delta x+\frac{f^{(2)}(x)}{2!}\Delta x^2+O(\Delta x^3)$，其中$\Delta x$ 可正可负，但要接近于0。

x沿方向D移动步长η后变为$x+\eta D$，由一阶泰勒展开式得$f(x+\eta D)=f(x)+f'(x)\eta D$，我们希望x的本次移动使目标函数$f(x)$下降得尽可能多，即$max\{f(x)-f(x+\eta D)=-f'(x)\eta D\}$，$\eta$确定的情况下即最小化$min\ f'(x)D$。$f'(x)$是梯度的方向，$D$是x移动的方向，要使两个向量的内积最小，当然要使两个向量的方向相反，即x 要沿着负梯度的方向前进。

$$x^{(t+1)}=x^{(t)}-\eta g^{(t)}$$

g是$f(x)$的梯度。

牛顿法

梯度下降法只考虑函数的一次近似，而牛顿迭代法考虑函数的二次近似。一阶泰勒展开式的矩阵形式:

$$f(X^{(t)} + \Delta X) = f(X^{(t)}) + g(X^{(t)})^T \Delta X$$

令$X^{(t)} + \Delta X = X^{(t+1)}$,
得$f(X^{(t+1)}) = f(X^{(t)}) + g(X^{(t)})^T (X^{(t+1)} - X^{(t)})$。$f(X^{(t+1)})$ 取得最小值时其导数为0，即$g(X^{(t+1)}) = g(X^{(t)}) + H(X^{(t)})(X^{(t+1)} - X^{(t)}) = 0$
$H(X)$是Hesse矩阵，可以理解为是二阶梯度。得

$$X^{(t+1)} = X^{(t)} - H(X^{(t)})^{-1} g(X^{(t)})$$

即X 的移动方向为$-H(X^{(t)})^{-1} g(X^{(t)})$，函数值要在此方向上下降，就需要它与梯度的方向相反，即$-g(X^{(t)})^T H(X^{(t)})^{-1} g(X^{(t)}) < 0$，所以要求在每一个迭代点上为上Hesse矩阵必须是正定的。

拟牛顿法

在拟牛顿法中每次迭代并不需要直接去计算Hesse矩阵，而是构造与Hesse 矩阵相近的正定矩阵。一阶泰勒展开式

$$f(X^{(t+1)} - \Delta X) = f(X^{(t+1)}) - \Delta X g(X^{(t+1)})$$

令$X^{(t+1)} - \Delta X = X^{(t)}$，然后对上式两边求导

$$g(X^{(t)}) = g(X^{(t+1)}) - \Delta X H(X^{(t+1)})$$

$$H(X^{(t+1)}) = \frac{g(X^{(t+1)}) - g(X^{(t)})}{X^{(t+1)} - X^{(t)}} \tag{1}$$

(1)式就是所有拟牛顿法的基本出发点。L-BFGS(Limited-Memory BFGS或Limited-Storage BFGS)是工程中最流行的一种拟牛顿法，因为它极大地节约了内存。

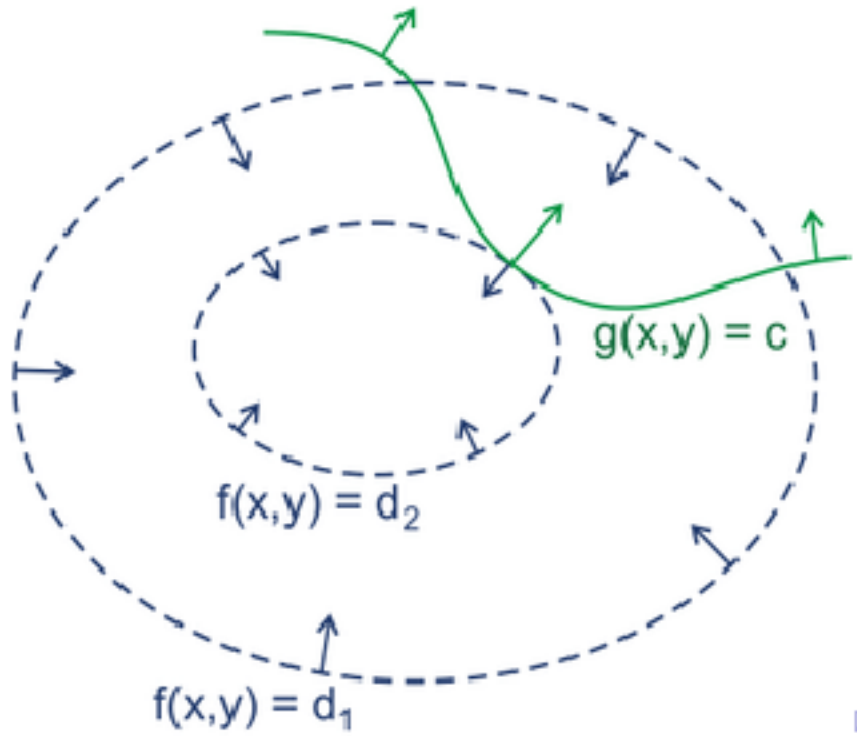
拉格朗日乘子法

带等式约束的优化问题$\begin{cases} min \;\; f(X) \\ s.t. \;\; g_i(X) = 0 \quad i = 1, 2, \cdots, n \end{cases}$

等价于$min \;\; f(X) + \sum_{i=1}^{n} \lambda_i g_i(X)$，其中$\lambda_i \neq 0$称为拉格朗日乘子。

现有二维优化问题：

$$\begin{cases} min \;\; f(x, y) \\ s.t. \;\; g(x, y) = c \end{cases} \tag{2}$$



拉格朗日乘子法

图中绿线是$g(x,y)=c$的轨迹，蓝线是$f(x,y)$的等值线，越往内环$f(x,y)$ 的值越小（即$f(x,y)$是凸函数）。从图中可以直观地看到在(2)式的最优解处$f(x,y)$和$g(x,y)-c$的梯度共线且方向相反，即

$$\bigtriangledown[f(x,y)+\lambda(g(x,y)-c)]=0 \quad \lambda \neq 0 \tag{3}$$

而满足(3)的点同时又是(4)的解。

$$min \quad f(x,y)+\lambda(g(x,y)-c) \tag{4}$$

所以(2)和(4)等价。

KKT条件

对于优化问题$\begin{cases} min \ \ f(x) \\ s.t. \ \ g_k(x) \leq 0 \quad k = 1, 2, \cdots, n \end{cases}$我们构造下式

$$L(x, \mu) = f(x) + \sum_{k=1}^{q} \mu_k g_k(x) \ \ \mu_k \geq 0, g_k(x) \leq 0 \tag{5}$$

$$\begin{aligned} \max_{\mu} \min_{x} L(x, \mu) &= \max_{\mu}[\min_{x} f(x) + \min_{x} \mu g(x)] = \max_{\mu} \min_{x} f(x) \\ &+ \max_{\mu} \min_{x} \mu g(x) = \min_{x} f(x) + \max_{\mu} \min_{x} \mu g(x) \end{aligned} \tag{6}$$

这里的u和g都就向量，所以去掉了下标k以及求和符号。

$$\left.\begin{matrix} \mu \geq 0 \\ g(x) \leq 0 \end{matrix}\right\} \Rightarrow \min_{x} \mu g(x) = \begin{cases} 0 & if \ \mu = 0 \ or \ g(x) = 0 \\ -\infty & if \ \mu > 0 \ and \ g(x) < 0 \end{cases}$$

$\therefore \max_{\mu} \min_{x} \mu g(x) = 0$此时满足$\mu = 0 \ or \ g(x) = 0$，即$\mu g(x) = 0$。

对偶问题

$\because \max_\mu \min_x \mu g(x) = 0$，代入(6) 得

$$\max_\mu \min_x L(x, \mu) = \min_x f(x) \tag{7}$$

$$\left.\begin{array}{r} \mu_k \geq 0 \\ g_k(x) \leq 0 \end{array}\right\} \Rightarrow \mu g(x) \leq 0$$

由(5)得

$$\therefore \max_\mu L(x, \mu) = f(x) \tag{8}$$

$$\therefore \min_x f(x) = \min_x \max_\mu L(x, \mu) \tag{9}$$

联合(7)(9)得

$$\min_x \max_\mu L(x, \mu) = \max_\mu \min_x L(x, \mu) = \min_x f(x)$$

KKT条件

$f(x)$在x^*处最得极小值，把x^*代入(8)得$\max_\mu L(x^*,\mu)=f(x^*)$，由(7)得$\max_\mu \min_x L(x,\mu)=f(x^*)$，所以$L(x^*,\mu)=\min_x L(x,\mu)$，这说明$x^*$也是$L(x,\mu)$的极值点，即$\frac{\partial L(x,\mu)}{\partial x}|_{x=x^*}=0$。

最后总结一下：

$$\left.\begin{array}{c} L(x,\mu)=f(x)+\sum_{k=1}^{q}\mu_k g_k(x) \\ \mu_k \geq 0 \\ g_k(x)\leq 0 \end{array}\right\}$$

$$\Rightarrow \begin{cases} \min_x \max_\mu L(x,\mu)=\max_\mu \min_x L(x,\mu)=\min_x f(x)=f(x^*) \\ \mu_k g_k(x^*)=0 \\ \frac{\partial L(x,\mu)}{\partial x}|_{x=x^*}=0 \end{cases}$$

KKT条件

KKT条件是拉格朗日乘子法的泛化，如果我们把等式约束和不等式约束一并纳入进来则表现为：

$$\left.\begin{array}{c} L(x,\lambda,\mu) = f(x) + \sum_{i=1}^{n} \lambda_i h_i(x) + \sum_{k=1}^{q} \mu_k g_k(x) \\ \lambda_i \neq 0 \\ h_i(x) = 0 \\ \mu_k \geq 0 \\ g_k(x) \leq 0 \end{array}\right\}$$

$$\Rightarrow \begin{cases} \min_x \max_\mu L(x,\lambda,\mu) = \max_\mu \min_x L(x,\lambda,\mu) = \min_x f(x) = f(x^*) \\ \mu_k g_k(x^*) = 0 \\ \frac{\partial L(x,\lambda,\mu)}{\partial x}|_{x=x^*} = 0 \end{cases}$$

注：x, λ, μ都是向量。

并行SGD算法

step1 将训练数据集D均匀地划分到k台机器上，第i台机器上获得子集D_i，$|D_i| = T$。

step2 每台机器上给定学习率η，并令向量$\mathbf{v}=\mathbf{0}$。

step3 对于$i \in 1, 2, \cdots, k$并行地执行

1. 将D_i中的样本随机打乱顺序。
2. 初始化向量$w_0^{(i)}=\mathbf{v}$
3. **for all** $t \in \{1, \cdots, T\}$:**do**

   $$w_t^{(i)} = w_{t-1}^{(i)} = -\eta \frac{\partial Loss(w_{t-1}^{(i)})}{\partial w_{t-1}^{(i)}}$$

   end for

step4 求每台机器上算得的权值向量$w_T^{(i)}$ 的平均值$v = \frac{1}{k}\sum_{i=1}^{k} w_T^{(i)}$。

step5 若达到收敛标准则返回$\mathbf{v}$；否则转到step3。

并行SGD算法

- 只涉及数据并行，不涉及模型并行。各台机器拿到部分样本集后并行训练，但每台机器上都要训练全部的模型参数。适合于样本很多而参数不多的情况。
- 训练完所有样本后，各台机器之间才需要进行一次通信，并行度较高。
- 非常适合用MapReduce框架来实现。

梯度下降优化算法

基于梯度下降的算法存在一些问题：

- 选择一个合适的学习率是很困难的。
- 在训练过程中通常会使用诸如退火的方式动态改变学习率，这种调整策略是在训练之前就已经固定好了的，不能自适应每次迭代时数据集的特点。
- 所有参数都使用相同的学习率也不妥。如果数据是稀疏的或者特征出现的频率不同，我们倾向于让那些罕见的特征每次获得一个较大的更新。

在深度学习领域人们提出很多优化梯度下降的算法，如动量项法(Moment)、Nesterov梯度加速法、Adagrad法、RMSprop法、最后还有自适应动量估计法(Adaptive Moment Estimation, Adam)。

动量法

对于优化问题$min\ J(\theta)$传统的梯度下降法每次迭代采用如下方式更新θ

$$\begin{aligned} v_t &= \eta \bigtriangledown_\theta J(\theta) \\ \theta &= \theta - v_t \end{aligned} \tag{10}$$

v_t表示参数要更新的方向。如果加了动量项则为

$$v_t = \gamma v_{t-1} + \eta \bigtriangledown_\theta J(\theta) \tag{11}$$

γ通常在0.9附近取值。

思想：如果本次的梯度方向跟上次相同，则把步子迈大；如果本次的梯度方向跟上次相反，则把步子缩小。这样有望减少振荡加速收敛。

Nesterov加速法

$$
\begin{aligned}
v_t &= \gamma v_{t-1} + \eta \bigtriangledown_\theta J(\theta - \gamma v_{t-1}) \\
\theta &= \theta - v_t
\end{aligned}
\tag{12}
$$

γ通常在0.9附近取值。

思想：我们希望滚落的小球在预见到前方要上坡时能提前减速。用$\theta - \gamma v_{t-1}$来近似估计下一次迭代点的位置，则$\bigtriangledown_\theta J(\theta - \gamma v_{t-1})$为下一次迭代点的梯度，如果跟之前的梯度$v_{t-1}$方向相反，则步长$v_t$就会被缩小。

Adagrad法

Adagrad法为每个参数设置不同的学习率，使得频繁出现的参数更新小一些，不频繁出现的参数更新大一些。第i个参数第t次迭代时的取值记为$\theta_{t,i}$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\epsilon + \sum_{s=1}^{t} g_{s,i}^2}} g_{t,i} \tag{13}$$

$g_{t,i}$是第t次迭代时梯度在第i个维度上的取值。ϵ是为了防止除0异常，通常取O(1e-8)数量级。η可以取常数0.01。

由于$g_{s,i}^2$始终是正数，越往后累积和越大，可能会出现学习率太小参数不更新的情况。于是就有了RMSprop法。

RMSprop法

Adagrad法中一直累加g_t^2是一种非常激进的方法，RMSprop法在对g_t^2 进行累加时乘以一个小于1的系数。

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\epsilon + E[g^2]_t}} g_t \tag{14}$$

γ通常取0.9，η取0.001。

$E[g^2]$的计算是一个递归的过程，代数隔得越远g^2衰减得越严重。

Adam

$$
\begin{aligned}
m_t &= \gamma_1 m_{t-1} + (1-\gamma_1) g_t \\
v_t &= \gamma_2 v_{t-1} + (1-\gamma_2) g_t^2
\end{aligned}
\tag{15}
$$

m_t是历次g_t乘以相应衰减因子的加权和，v_t是历次g_t^2乘以相应衰减因子的加权和。由于m_0和v_0被初始化为$\mathbf{0}$，且γ_1 和γ_2又非常接近于1，这样就导致m_t和v_t经训练后容易倾向于$\mathbf{0}$。为避免这种后果，我们把g_t和g_t^2前面的系数化为1：

$$
\begin{aligned}
\hat{m}_t &= \frac{\gamma_1}{1-\gamma_1} m_{t-1} + g_t \\
\hat{v}_t &= \frac{\gamma_2}{1-\gamma_2} v_{t-1} + g_t^2
\end{aligned}
\tag{16}
$$

使用类似于RMSprop的方法，$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t$。建议$\gamma_1$取0.9，$\gamma_2$取0.999，$\epsilon$取$10^{-8}$。Adam似乎是上述学习率的优化算法中最好的。

大数据困境

特征：

$$(user\ features \otimes item\ IDs, item\ features \otimes user\ IDs, context\ features)$$

如果你有billion级别的特征，million级别的稀疏样本，进行在线学习时面临2个困境：效率太低；内存中容不下全部的特征。

理论上L_1正则化能够产生稀疏性，最终只保留少数的特征，但事上并实不尽然，因为很少有特征的权重能真正达到0，只是非常小接近0而已。

如果特征权重很小接近于0时就强行将其截断为0，也欠妥，因为权重可能还没有得到充分的训练。

简单系数舍入

通常，在SGD中第$i+1$轮迭代时使用下面的公式来更新权重。

$$w_{i+1} = f(w_i) = w_i - \eta g(w_i)$$

$g(w_i)$是目标函数在w_i处的导数，如果目标函数在w_i处不可导（比如目标函数中含有L_1正则项），则$g(w_i)$就从次梯度集合中任取一个。

简单系数舍入法每隔K轮迭代就使用截断函数T_0对权重进行截断。

$$f(w_i) = T_0(w_i - \eta g(w_i)) \quad if\ i\%K == 0$$

$$T_0(v, \theta) = \begin{cases} 0 & if\ |v| \leq \theta \\ v & otherwise \end{cases}$$

把K设得大一些，给那些很小的权重系数充分的时间让它跳到θ 以上，避免由于训练不充分而错误地丢弃掉了重要的特征。

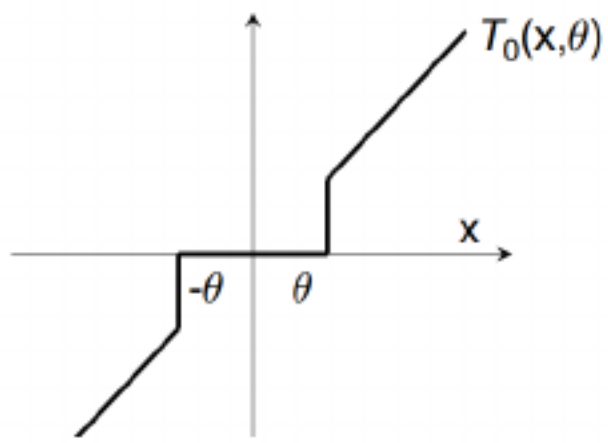
Truncated Gradient

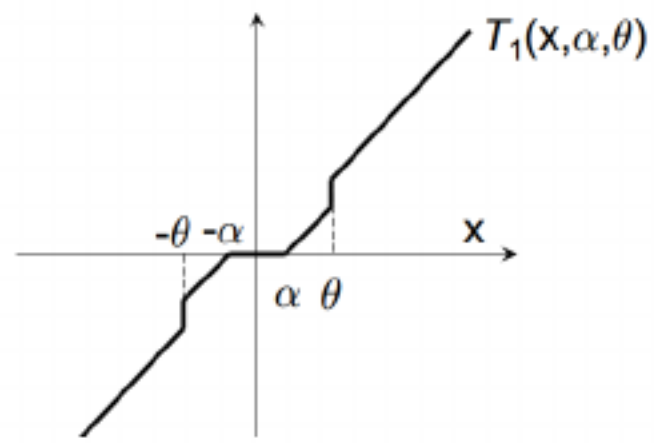
简单系数舍入法未免简单粗暴，Truncated Gradient是一种更温和的方法。

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

$$f(w_i) = T_1(w_i - \eta g(w_i), \eta\beta, \theta) \quad if\ i\%K == 0$$

$\eta\beta < \theta$，如果令$\beta = \theta$则我们只需要调节θ这一个参数就可以控制稀疏性。从下图可以看到$\beta(\beta = \frac{\alpha}{\eta})$和$\theta$设置得越大，稀疏性就越大。





Forward-Backward Splitting

$$w_{t+\frac{1}{2}} = w_t - \eta_t g_t^f \tag{17}$$

$$w_{t+1} = \underset{w}{argmin} \left\{ \frac{1}{2} \| w - w_{t+\frac{1}{2}} \|^2 + \eta_{t+\frac{1}{2}} \gamma(w) \right\} \tag{18}$$

g_t^f是函数$f(w_t)$的次梯度（梯度是次梯度的一个特例），η_t是前进的步长。公式(18)的第一项保证下一次迭代的w 在梯度下降法结果的附近，第二项是正则化函数，用于产生稀疏性。把(17)代入(18) 得

$$w_{t+1} = \underset{w}{argmin} \left\{ \frac{1}{2} \| w - w_t + \eta_t g_t^f \|^2 + \eta_{t+\frac{1}{2}} \gamma(w) \right\}$$

令导数为0

$$0 = w - w_t + \eta_t g_t^f + \eta_{t+\frac{1}{2}} g_{t+1}^{\gamma}$$

FOBOS

$$\therefore w_{t+1} = w_t - \eta_t g_t^f - \eta_{t+\frac{1}{2}} g_{t+1}^{\gamma}$$

我们看到w_{t+1}不仅跟第t轮迭代时的g_t^f有关系，还跟第t+1轮时的g_{t+1}^{γ}有关系，这正是FOBOS名称的由来。

当使用L_1正则项时$\gamma(w) = \lambda\|w\|_1$，权重更新公式为

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

*sgn*是符号函数。我们看到当$\left|w_i^{(t)} - \eta^{(t)} g_i^{(t)}\right| \leq \eta^{(t+\frac{1}{2})}\lambda$ 时$w_i^{(t+1)}$为0，意思是说当一条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$\eta^{(t+\frac{1}{2})}\lambda$时就认为在本次更新过程中该维度不够重要，对其进行“截断”，即令其权重为0。

Regularized Dual Averaging

RDA各个维度上权重的更新公式：

$$w_{t+1} = \underset{w}{argmin}\{\langle \overline{g}_t, w\rangle + \Psi(w) + \frac{\beta_t}{t} h(w)\} \tag{19}$$

$\overline{g}_t$是从第1代到第t代在本维度上次梯度的平均值（dual average），
$\Psi(w)$是正则项，β_t是一个非负且非递减序列，$h(w)$ 是一个严格凸函数。
当$\Psi(w) = \lambda_t \|w\|_1$，$\beta_t = \gamma\sqrt{t}$，$h(w) = \frac{1}{2}\|w\|_2^2$时，有

$$w_{t+1} = \underset{w}{argmin}\{\langle \overline{g}_t, w\rangle + \lambda_t \|w\|_1 + \frac{\gamma}{2\sqrt{t}}\|w\|_2^2\} \tag{20}$$

$$solution: \ w_{t+1} = \begin{cases} 0 & if \ |\overline{g}_t| \le \lambda_t \\ -\frac{\sqrt{t}}{\gamma}(\overline{g}_t - \lambda_t sgn(\overline{g}_t)) & otherwise \end{cases}$$

$\lambda_t = \lambda + \rho/\sqrt{t}$

Follow the Regularized Leader

$L_1 - FOBOS$中的学习率η通常与$\frac{1}{\sqrt{t}}$正相关，即随着t 的增加截断阈值越来越小。而$L_1 - RDA$中每一代的截断阈值至少都是λ，因此RDA在截断方面更激进，更容易产生稀疏性，通过控制λ这个参数可以在精度和稀疏性上进行权衡。

FOBOS这类基于梯度下降的方法有比较高的精度，RDA在损失一定精度的情况下产生更好的稀疏性，FTRL综合了FOBOS 和RDA 对于正则项和W限制的区别，其各维度上权重更新公式为：

$$w^{(t+1)} = \underset{w}{argmin}\left\{ g^{(1:t)}w + \lambda_1\|w\|_1 + \frac{1}{2}\lambda_2\|w\|_2^2 + \frac{1}{2}\sum_{s=1}^{t}\sigma^{(s)}\|w - w^{(s)}\|_2^2 \right\} \tag{21}$$

式(21)中$g^{(1:t)} = \sum_{s=1}^{t} g^{(s)}, \sigma^{(s)} = \frac{1}{\eta^{(s)}} - \frac{1}{\eta^{(s-1)}}, \sum_{s=1}^{t}\sigma^{(s)} = \frac{1}{\eta^{(t)}}$

FTRL

将(21)式展开得

$$w^{(t+1)} = \underset{w}{argmin} \left\{ \left(g^{(1:t)} - \sum_{s=1}^{t} \sigma^{(s)} w^{(s)} \right) w + \lambda_1 \|w\|_1 + \frac{1}{2} \left(\lambda_2 + \sum_{s=1}^{t} \sigma^{(s)} \right) \|w\|_2^2 + \frac{1}{2} \sum_{s=1}^{t} \sigma^{(s)} \|w^{(s)}\|_2^2 \right\} \tag{22}$$

第4项相对于w来说是常数可以去掉。该优化问题的解为：

$$w^{(t+1)} = \begin{cases} 0 & if \ \left|z^{(t)}\right| < \lambda_1 \\ -\left(\lambda_2 + \sum_{s=1}^{t} \sigma^{(s)}\right)^{-1} \left(z^{(t)} - \lambda_1 sgn(z^{(t)})\right) & otherwise \end{cases} \tag{23}$$

其中$z^{(t)} = g^{(1:t)} - \sum_{s=1}^{t} \sigma^{(s)} w^{(s)}$

FTRL

FTRL中每个维度上的学习率是分别考虑的。如果一个维度上的累计梯度比较大，即该维度上的变化比较大，那么这个维度上的学习率应该适当降低。

$$\eta^{(t)} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^{t} \left(g^{(s)}\right)^2}} \tag{24}$$

又$\sum_{s=1}^{t} \sigma^{(s)} = \frac{1}{\eta^{(t)}} = \frac{\beta + \sqrt{\sum_{s=1}^{t} \left(g^{(s)}\right)^2}}{\alpha}$代入(23)式得FTRL单个维度上的权重更新公式

$$w^{(t+1)} = \begin{cases} 0 & if \ \left|z^{(t)}\right| < \lambda_1 \\ -\left(\lambda_2 + \frac{\beta + \sqrt{\sum_{s=1}^{t} \left(g^{(s)}\right)^2}}{\alpha}\right)^{-1} \left(z^{(t)} - \lambda_1 sgn(z^{(t)})\right) & otherwise \end{cases} \tag{25}$$

$\lambda_1, \lambda_2, \alpha$和$\beta$是FTRL算法的参数。