

# Stochastic Optimization

April 30, 2018

## 1 Variants of Stochastic Gradients Algorithms

consider problem:

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(w) + \lambda \|w\|_1,$$

where  $f_i(w) = \log(1 + \exp(-y^i w^T x_i))$ , and  $\lambda > 0$ . the prediction is:

$$p(y = 1|x; w) = \frac{1}{1 + \exp(-w^T x)},$$

$$p(y = -1|x; w) = \frac{1}{1 + \exp(w^T x)}.$$

### 1.1 adagrad

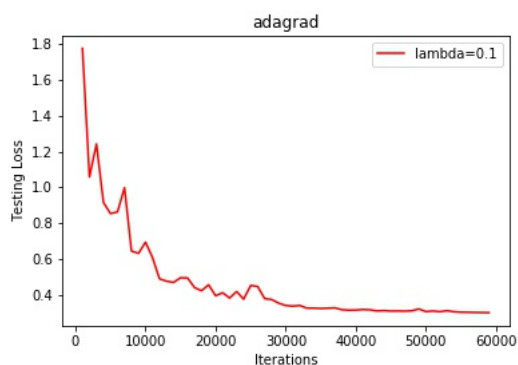
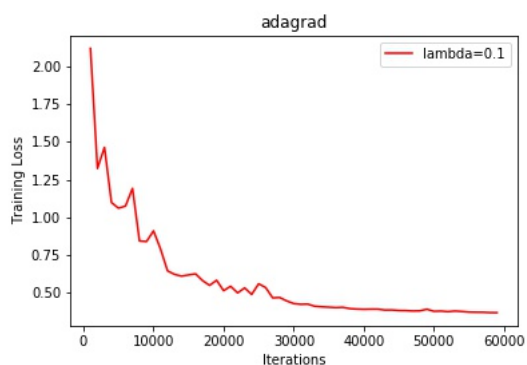
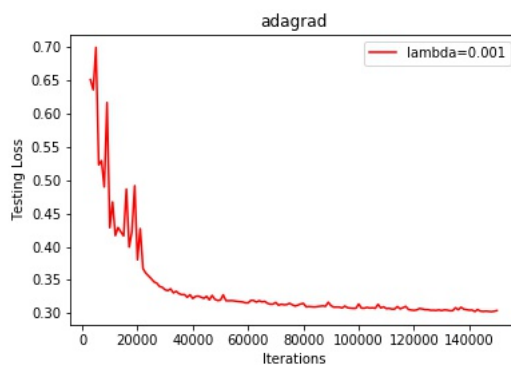
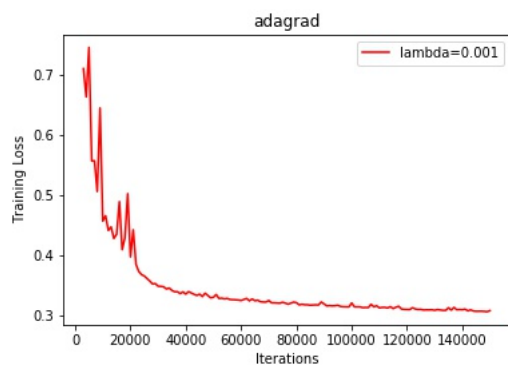
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t + \epsilon I}} \cdot g_t, (\epsilon = 10^{-8}).$$

其中,  $G_t$  是对角矩阵, 对角元素  $e_{ii}$  为过去到当前第  $i$  个参数  $\theta_i$  的梯度的平方和。

在训练过程中, 首先取步长 0.003 ( $\lambda = 0.1$  or 0.001 时) 当训练误差累计超过 10 次下降不超过某个阈值时, 我们将步长除以 10, 继续迭代。

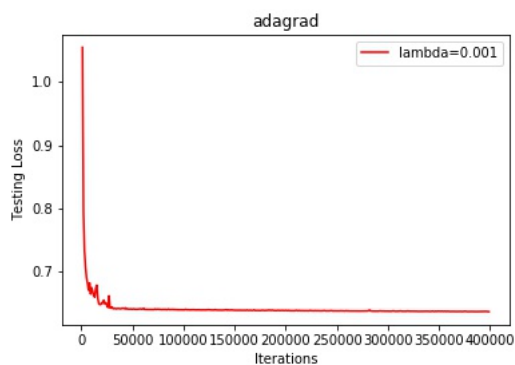
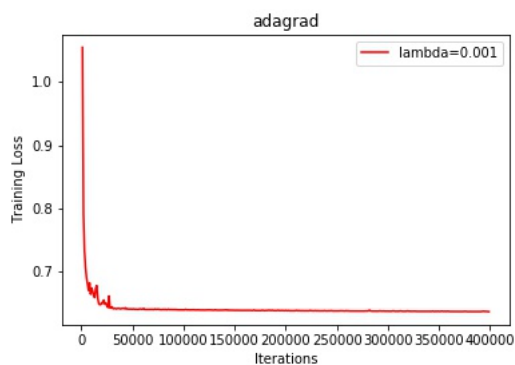
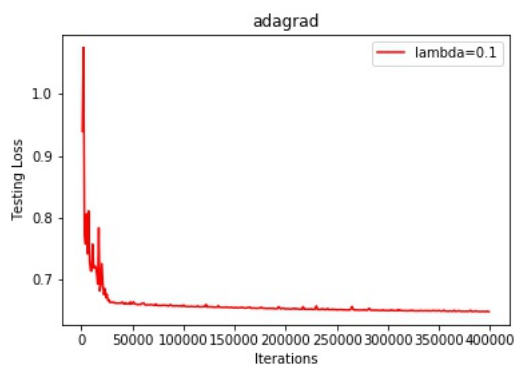
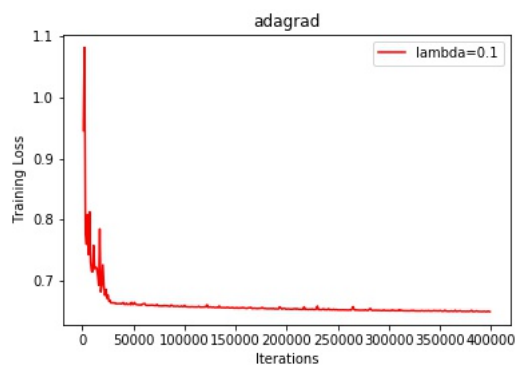
mnist 数据集上的结果:

$\lambda$	train_loss	test_loss
0.001	0.307	0.304
0.1	0.370	0.300



coverttype 数据集上的结果:

$\lambda$	train_loss	test_loss
0.001	0.636	0.636
0.1	0.650	0.648



## 1.2 adam

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

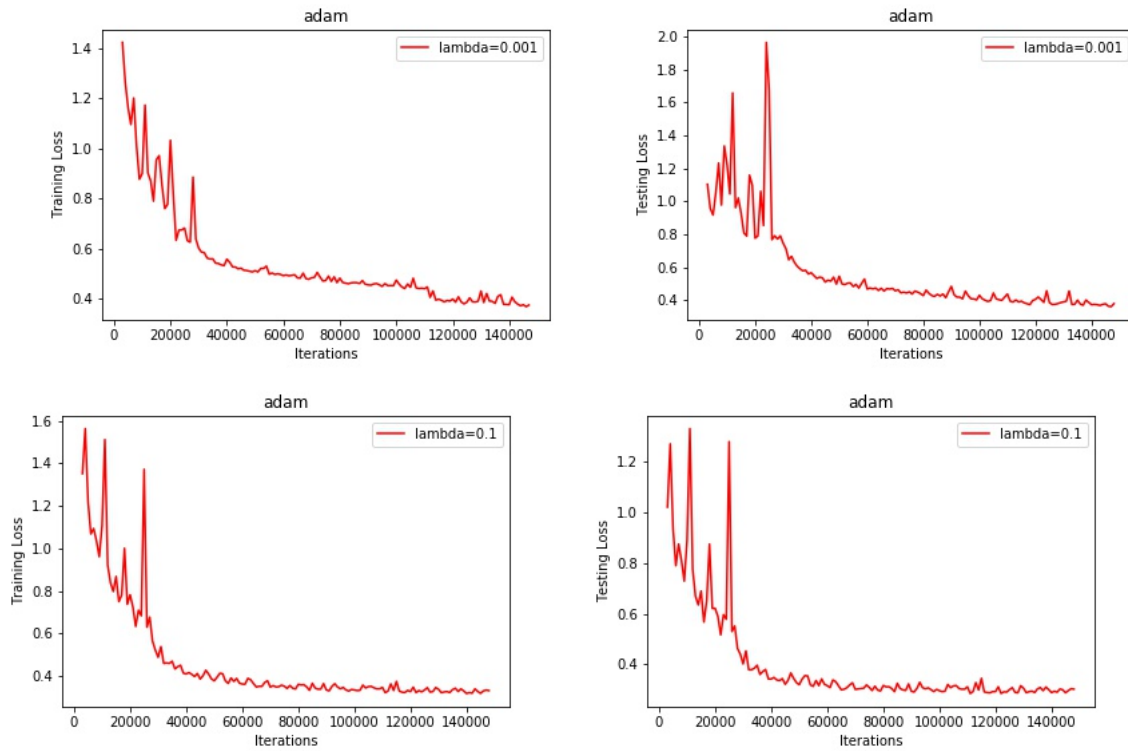
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{(\hat{v}_t) + \epsilon}} \hat{m}_t$$

$$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$$

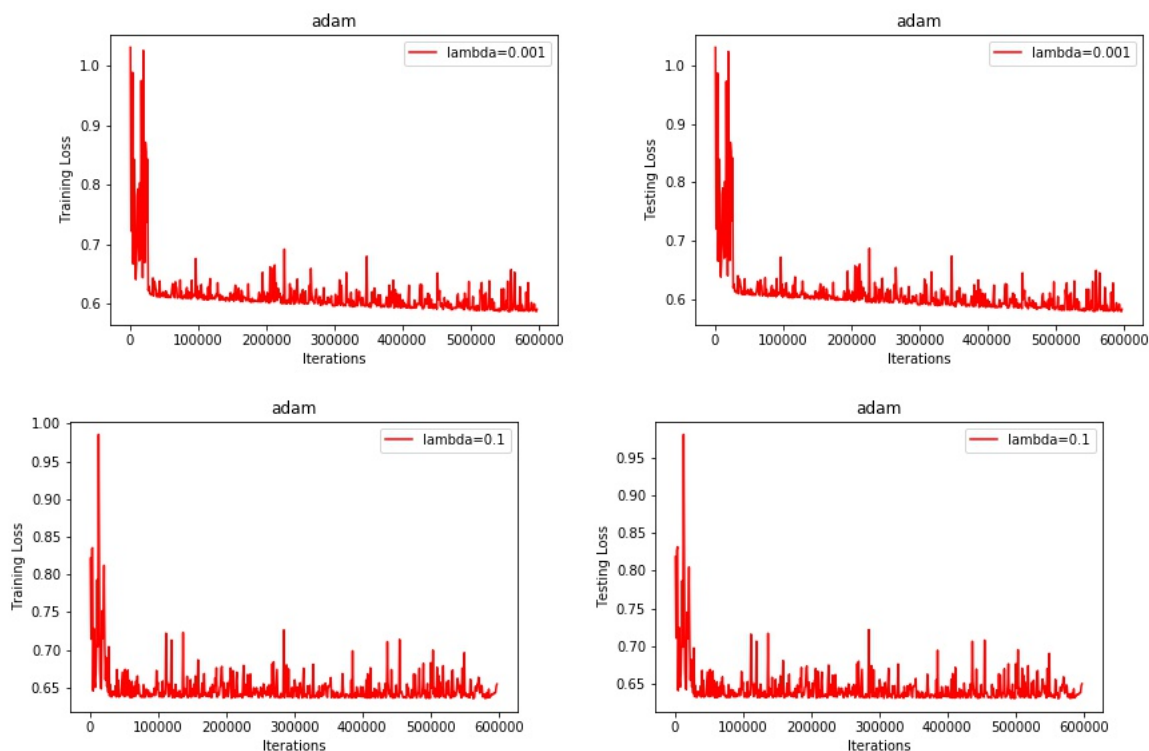
mnist 数据集上的结果：在训练过程中，首先取步长 0.0008 ( $\lambda = 0.1$ )，0.001 ( $\lambda = 0.001$ ) 当训练误差累计超过 10 次下降不超过某个阈值时，我们将步长除以 10，继续迭代。

$\lambda$	train_loss	test_loss
0.001	0.358	0.360
0.1	0.332	0.301



coverttype 数据集上的结果：

$\lambda$	train_loss	test_loss
0.001	0.586	0.579
0.1	0.636	0.631



### 1.3 SVRG

ref: <https://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction.pdf>

ref: <https://arxiv.org/pdf/1403.4699.pdf> proxSVRG 算法描述:

#### Algorithm 2: SVRG method

Input:  $\tilde{x}_0, \alpha, m$

**for**  $e = 1 : E$  **do**

    ▶  $y \leftarrow \tilde{x}_{e-1}, x_1 \leftarrow \tilde{x}_{e-1}.$

    ▶  $g \leftarrow \nabla f(y)$  (full gradient)

    ▶ **for**  $k = 1 : m$  **do**

        • pick  $s_k \in \{1, \dots, n\}$  uniformly at random.

        •  $v_k = \nabla f_{s_k}(x_k) - \nabla f_{s_k}(y) + \nabla f(y)$

        •  $x_{k+1} = x_k - \alpha v_k$

    ▶ **end for**

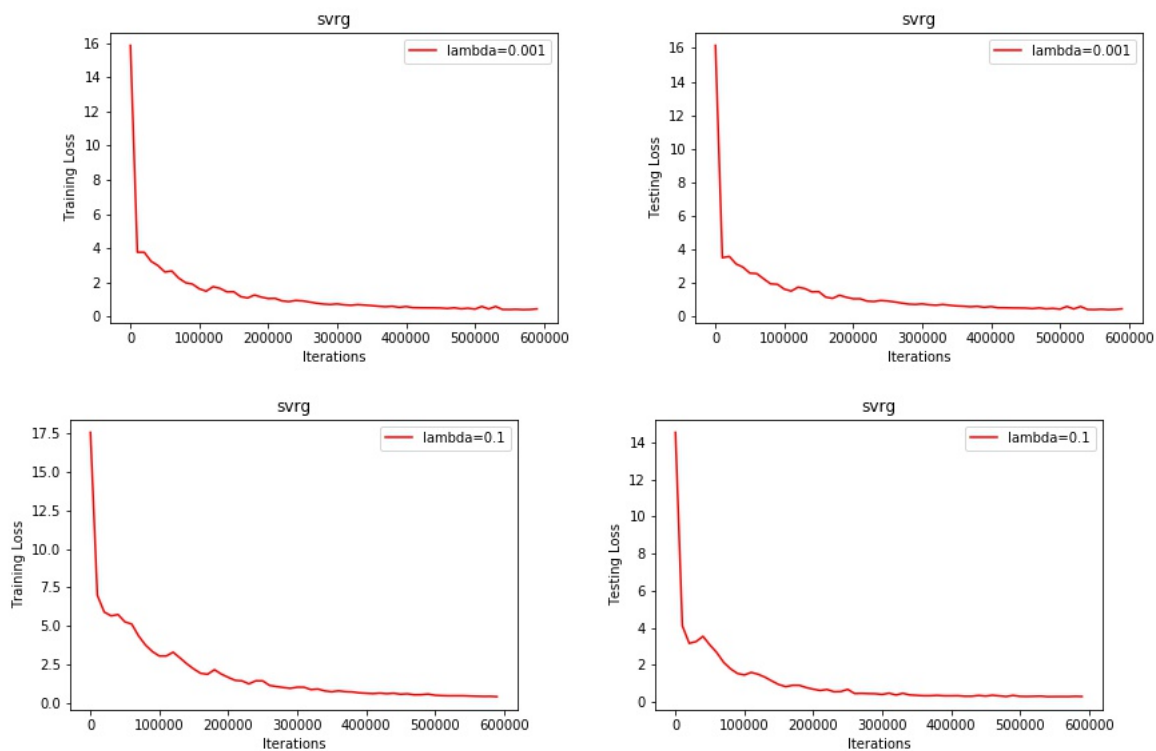
    ▶  $\tilde{x}_e \leftarrow \frac{1}{m} \sum_{k=1}^m x_k$

**end for**

在算法迭代中，我们先做几步 SGD，然后再进行 SVRG 更新，当内层循环结束后，将内层循环得到的变量进行平均。

mnist 数据集上的结果:

$\lambda$	train_loss	test_loss
0.001	0.397	0.402
0.1	0.405	0.274



## 1.4 SVRG-BB

ref: 《BarzilaiBorwein Step Size for Stochastic Gradient Descent》 NIPS2016

<http://lanl.arxiv.org/pdf/1605.04131v2>

ref: [https://github.com/tanconghui/Stochastic\\_BB](https://github.com/tanconghui/Stochastic_BB)

算法描述:

---

**Algorithm 2** SVRG with BB step size (SVRG-BB)

---

**Parameters:** update frequency  $m$ , initial point  $\tilde{x}_0$ , initial step size  $\eta_0$  (only used in the first epoch)

**for**  $k = 0, 1, \dots$  **do**

$$g_k = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x}_k)$$

**if**  $k > 0$  **then**

$$\eta_k = \frac{1}{m} \cdot \|\tilde{x}_k - \tilde{x}_{k-1}\|_2^2 / (\tilde{x}_k - \tilde{x}_{k-1})^\top (g_k - g_{k-1}) \quad (3.2)$$

**end if**

$$x_0 = \tilde{x}_k$$

**for**  $t = 0, \dots, m-1$  **do**

Randomly pick  $i_t \in \{1, \dots, n\}$

$$x_{t+1} = x_t - \eta_k (\nabla f_{i_t}(x_t) - \nabla f_{i_t}(\tilde{x}_k) + g_k)$$

**end for**

$$\tilde{x}_{k+1} = x_m$$

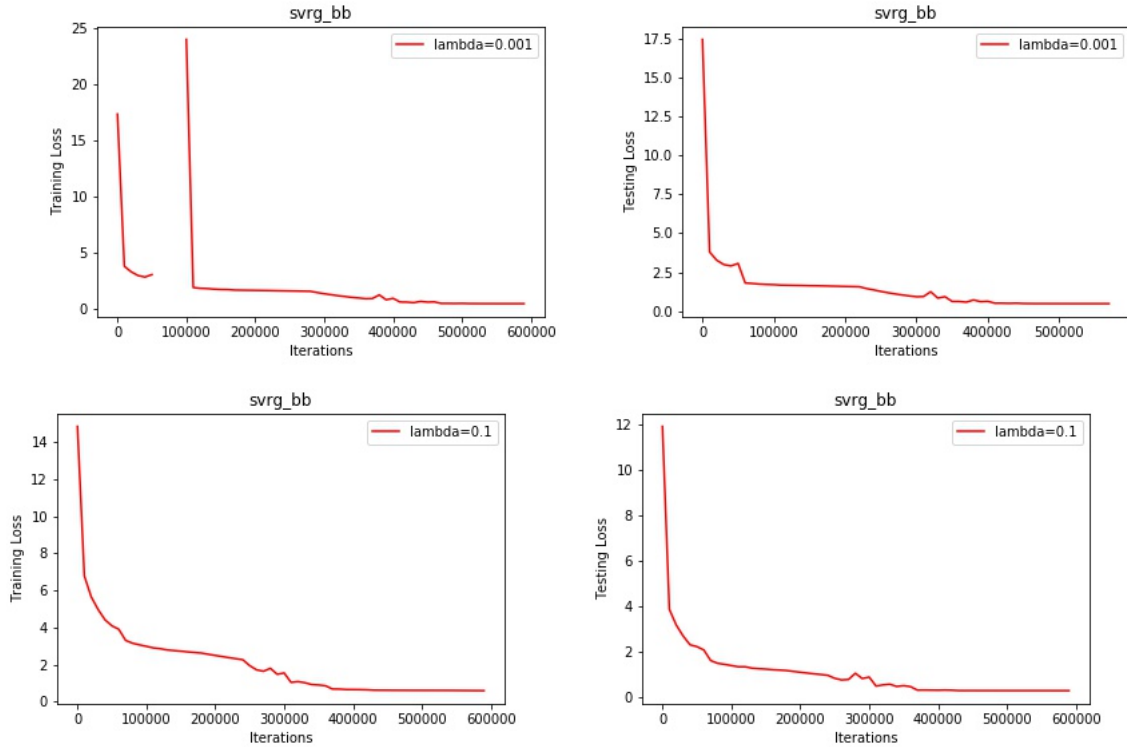
**end for**

---

在算法迭代中，我们先做几步 SGD，然后再进行 SVRBB 更新。

mnist 数据集上的结果：

$\lambda$	train_loss	test_loss
0.001	0.470	0.476
0.1	0.594	0.292



## 1.5 FTRL: follow the regularized leader

Reference:

1. 《Online Learning and Online Convex Optimization1》 section 2.2,2.3

URL: <http://www.cs.huji.ac.il/~shais/papers/OLsurvey.pdf>

3. 原始论文:

URL: <https://www.eecs.tufts.edu/~dsculley/papers/ad-click-prediction.pdf>

2. from 美团点评技术团队

URL: <https://tech.meituan.com/online-learning.html>

4. 其他:

URL: <http://www.datakit.cn/blog/2016/05/11/ftrl.html>

---

**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  $\alpha, \beta, \lambda_1, \lambda_2$

$(\forall i \in \{1, \dots, d\})$ , 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  $\frac{1}{\eta_{t,i}} - \frac{1}{\eta_{t-1,i}}$*

$z_i \leftarrow z_i + g_i - \sigma_i w_{t,i}$

$n_i \leftarrow n_i + g_i^2$

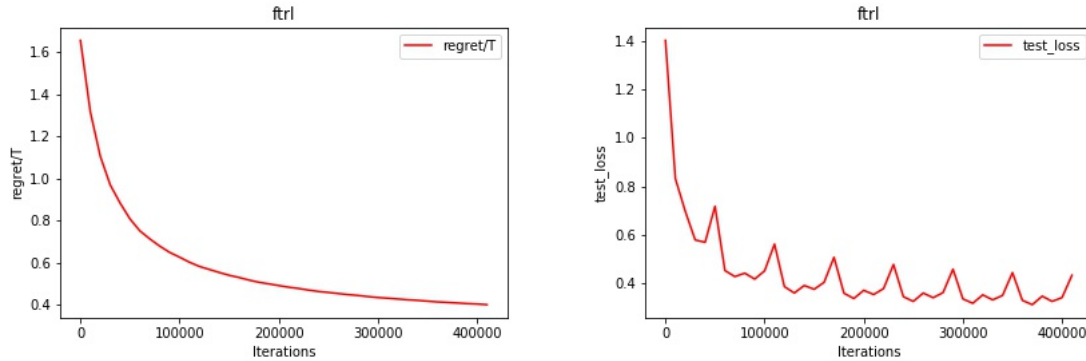
**end for**

**end for**

---

以上算法是针对 0, 1 分类问题, 对于 1, -1 分类, 需稍加改动。

由于在线算法测评方式不太一样, 我们采用  $\text{regret}/T$ , 和  $\text{test\_loss}$  两种指标。



## 2 future work

由于 mnist dataset 的样本向量比较稀疏，所以可以考虑稀疏矩阵运算将其加速，我了解到 xlearn 这个大规模稀疏数据机器学习库 (<https://github.com/aksnzhy/xlearn>) 实现了 SGD, AdaGrad, FTRL 等，可以借鉴。