# 互联网应用下的大规模在线学习算法

陈景东　2013-7-25

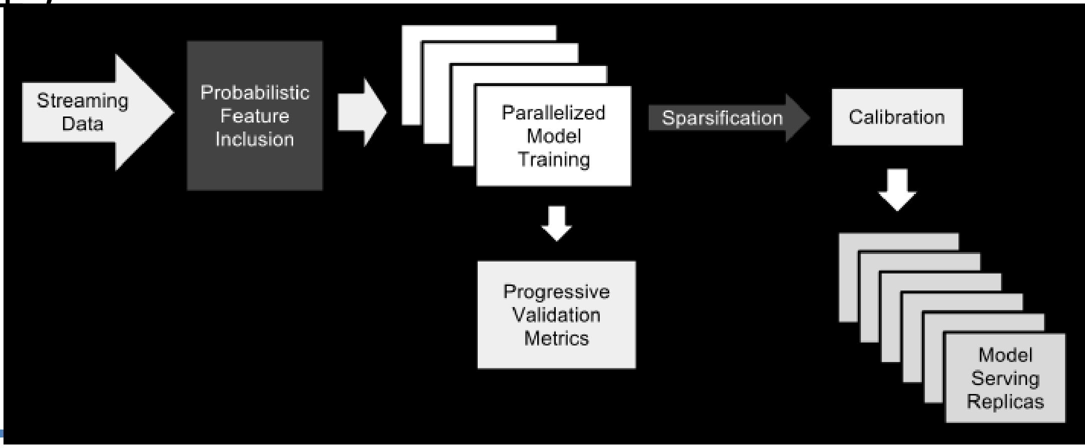
## 大纲

- 通用在线学习算法
  - 在线学习算法的应用背景
  - SGD、mini-batch算法简介（Andrew NG）
  - SGD算法的收敛特性（T. Zhang）
- SGD稀疏化方法
  - 稀疏化方法概要
  - Truncate Gradient
  - Regularized Dual Averaging
  - TFRL-Proximal
- 大规模在线学习系统实践
  - 节省内存
  - 有效地模型评估
  - 其它问题的探讨

## 在线学习算法应用示例

- 视频分析

  - 场景复杂、多变，如何实现运动目标检测
- CTR预估
  - 广告库中选出用户最感兴趣的样本；
  - 数以万亿计的样本及特征；
- 算法适用环境
  - 环境复杂多变；
  - 持续动态输入的大量样本；


Streaming Data
Probabilistic Feature Inclusion
Parallelized Model Training
Sparsification
Calibration
Progressive Validation Metrics
Model Serving Replicas

# Batch gradient descent

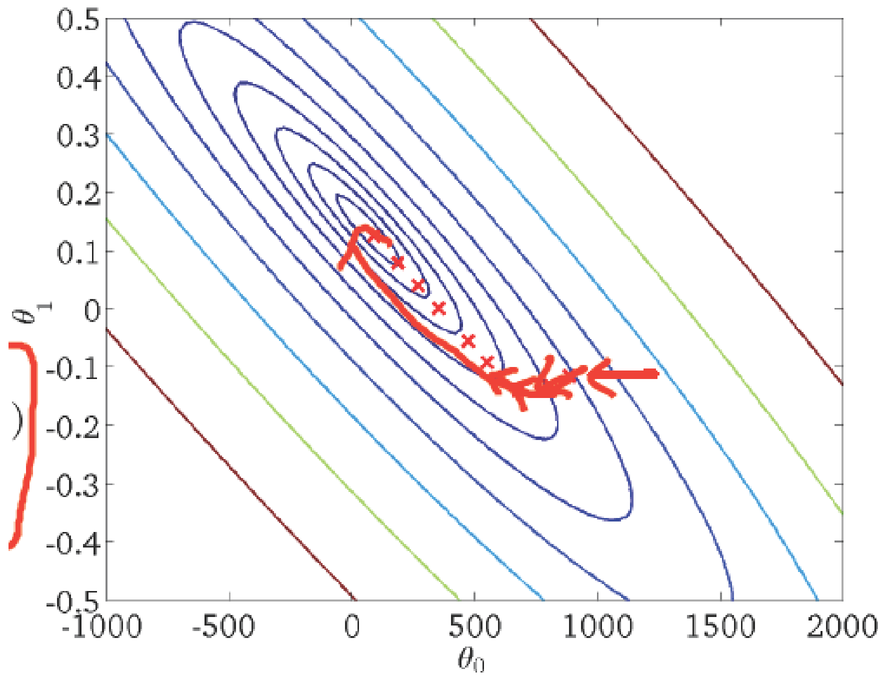
- 以线性回归为例

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

```
Repeat {
    $\theta_j := \theta_j - \alpha \boxed{\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}$
    (for every $j = 0, \ldots, n$)
}
```

$M = \underline{300{,}000{,}000}$



- 每迭代一次需要读取所有样本，计算量大

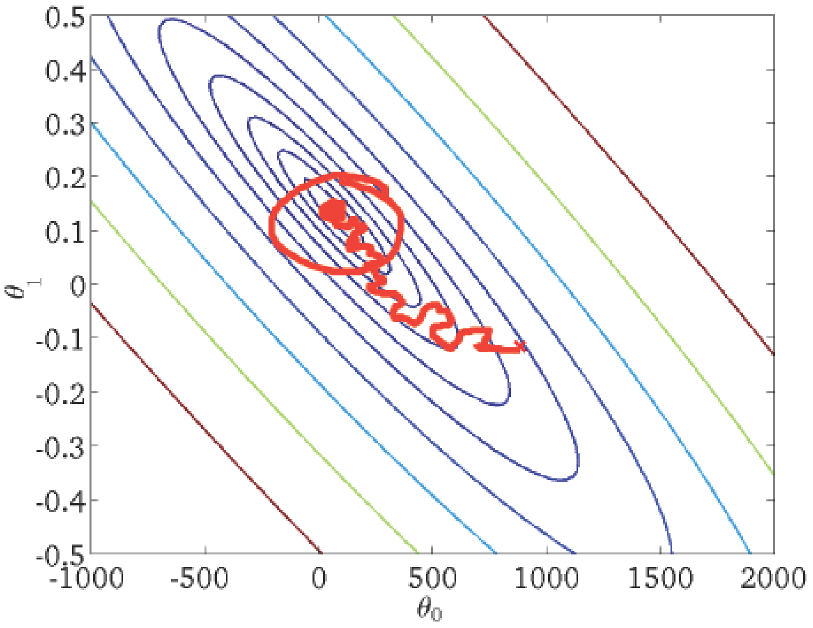
# Stochastic gradient descent

- 以线性回归为例

$$cost(\theta,(x^{(i)},y^{(i)}))=\frac{1}{2}(h_\theta(x^{(i)})-y^{(i)})^2$$

$$J_{train}(\theta)=\frac{1}{2m}\sum_{i=1}^{m}cost(\theta,(x^{(i)},y^{(i)}))$$

```
1.  Randomly shuffle dataset.
2.  Repeat {
        for $i := 1,\ldots,m$        {
            $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}$
                (for $j = 0,\ldots,n$)
        }
    }
```



- 学习率α通常为一个充分小的常数，α也可以按照时间逐步衰减

(E.g. $\alpha = \frac{\texttt{const1}}{\texttt{iterationNumber + const2}}$ )

- 若样本不完全随机？

## Mini-batch gradient descent

Say $\boxed{b = 10}$, $m = \boxed{1000}$.

Repeat {

```
    for $i = 1, 11, 21, 31, \ldots, \boxed{991}$ {
        $\theta_j := \theta_j - \alpha \frac{1}{\boxed{10}} \sum_{k=i}^{\boxed{i+9}} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$
                (for every $j = 0, \ldots, n$)
    }
}
```

$m = \underline{300,000,000}$

- 对比SGD
  - 梯度方向更稳定
  - 收敛速度更慢

## Standard SGD

- 目标函数

$$Q_\lambda(w) = E_{X,Y}\phi(w^T X, Y) + \frac{\lambda}{2}||w||_2^2,$$

- Standard SGD

```
Initialize $\hat{w}_0$
for $t = 1, 2, \ldots$
  Draw $(X_t, Y_t)$ randomly from $D$.
  Update $\hat{w}_{t-1}$ as:
  $\hat{w}_t = \hat{w}_{t-1} - \eta_t S^{-1}(\lambda \hat{w}_{t-1} + \phi_1'(w_{t-1}^T X_t, Y_t) X_t)$
end
```

- 其中 $\phi_1'(p, y) = \frac{\partial}{\partial p}\phi(p, y)$

## Standard SGD

- 事先将学习率固定为一个比较小的常数，是一个简单实用的办法；
- 在数据集上多几次迭代，SGD算法更接近batch最优解；
- 数据集上的迭代次数与正则化效应密切相关
  - 迭代次数越少正则化效应越强
    - Go through the data once ……the variance of the estimator is relatively small
    - Go through the data multiple time…. The variance increase;
  - SGD算法在数据集上1次（少量的几次）即可收敛,有比较好的正则化效应，这是SGD算法高效的重要原因

## Averaged SGD

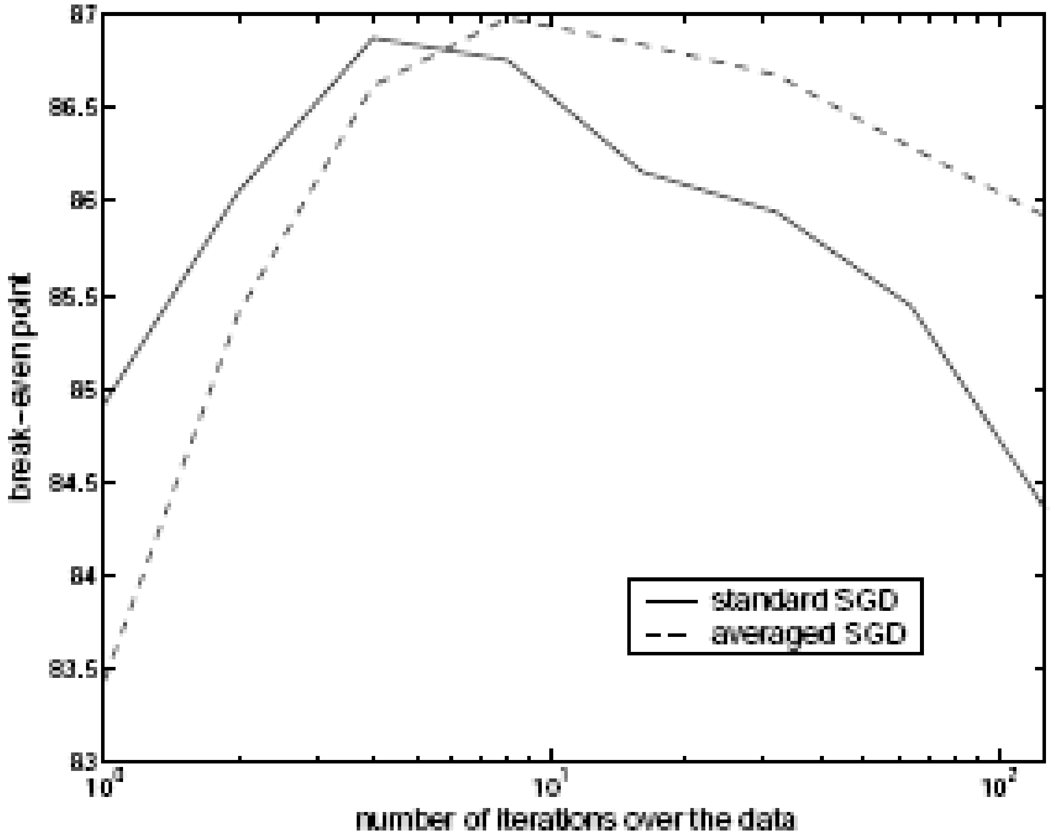
- Averaged SGD

```
Initialize $\hat{w}_0$
Let $\hat{v}_0 = 0$ and $r_0 = 0$
for $t = 1, 2, \ldots$
  Draw $(X_t, Y_t)$ randomly from $D$.
  Update $\hat{w}_{t-1}$ as:
  $\hat{w}_t = \hat{w}_{t-1} - \eta_t S^{-1}(\lambda \hat{w}_{t-1} + \phi_1'(w_{t-1}^T X_t, Y_t) X_t)$
  $r_t = r_{t-1} + \eta_t - \frac{AM^2\eta_t^2}{2}$
  $\hat{v}_t = \frac{r_{t-1}}{r_t}\hat{v}_{t-1} + \frac{r_t - r_{t-1}}{r_t}\hat{w}_{t-1}$
end
```

- Averaged predictor is more stable than non-averaged

## 算法对比

- SGD算法收敛速度快（约10步）；
- Averaging并没有带来收益，SGD源于学习率比较小；
- Early stopping与正则化效果相当；



Baidu推广

## 收敛性证明思路

- In separable case , stochastic update with quadratically  smmothed SVM behaves similar to the perception algorithm
- Using a technique in the online learning literature to extend the traditional proof of perception mistake bounds

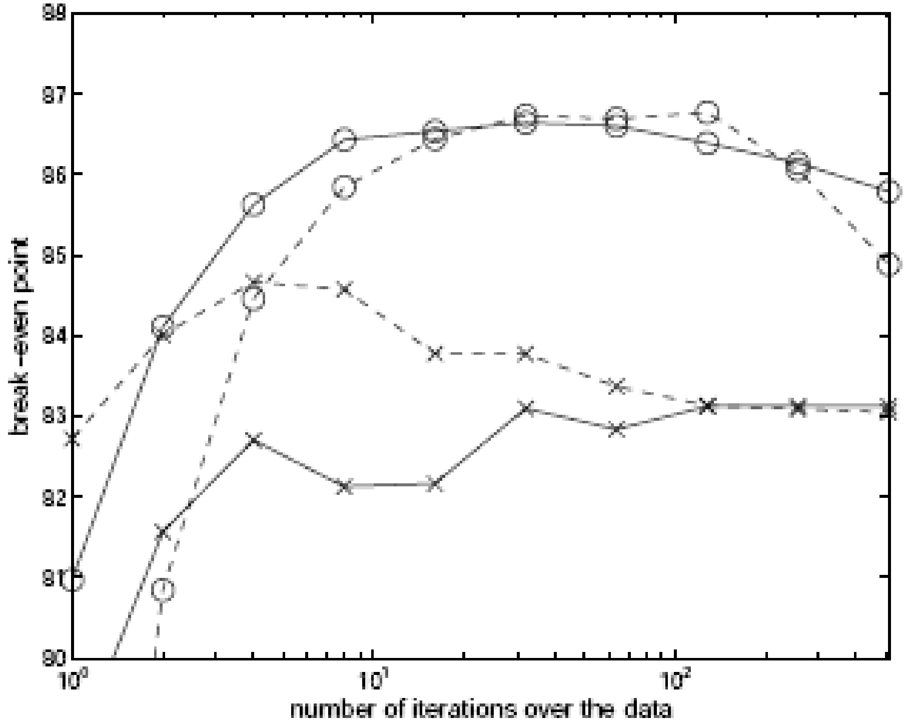


*Figure 2.* Performance of perceptron and SGD with SVM loss versus the number of iterations over the data. non-averaged methods: 'solid lines'; averaged methods: 'dashed lines'; perceptron: 'x', SGD : 'o'

## 大纲

- 通用在线学习算法
  - 在线学习算法的应用背景
  - SGD、mini-batch算法简介（Andrew NG）
  - SGD算法的收敛特性（T. zhang）
- SGD稀疏化方法
  - 稀疏化的必要性
  - Truncate Gradient
  - Regularized Dual Averaging
  - TFRL-Proximal
- 大规模在线学习系统实践
  - 节省内存
  - 有效地模型评估
  - 其它问题的探讨

# 参数稀疏化方法概述

- 参数稀疏化的必要性
  - 内存的限制（训练过程中）；
  - 快速预估服务的需求（训练结束）；
- 一些无效的正则化方法
  - L1 正则化；
  - 简单的四舍五入；
- 有效地稀疏化方法
  - Truncate Gradient（J. Langford et al, JMLR 2009）
  - Regularized Dual Averaging(L. Xiao NIPS 2009)
  - TFRL-Proximal(H. B. McMahan et al, ICML 2013)

## Truncate Gradient

- Simple Coefficient Rounding

$$f(w_i) = T_0(w_i - \eta \nabla_1 L(w_i, z_i), \theta),$$

  - where $T_0(v_j, \theta) = v_j I(|v_j| < \theta)$

  - Too aggressive , a large number of infinitesimal steps;

- Truncate Gradient

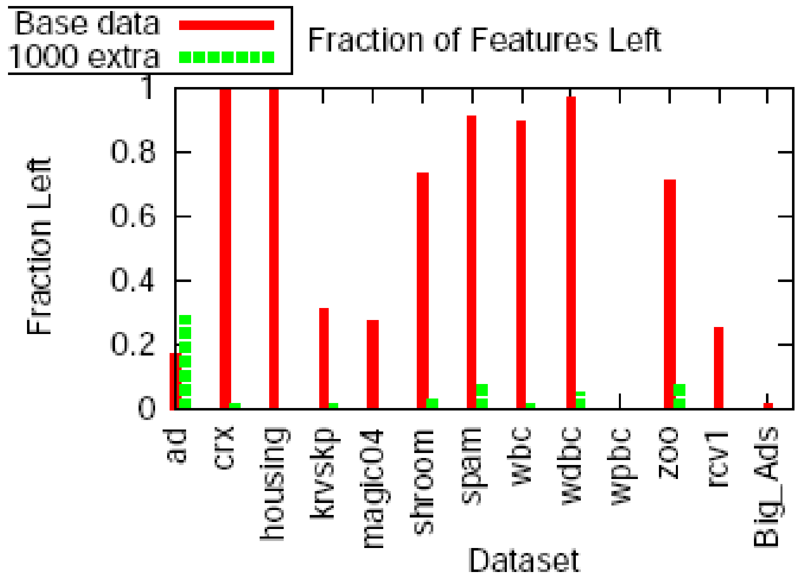
$$f(w_i) = T_1(w_i - \eta \nabla_1 L(w_i, z_i), \eta g_i, \theta),$$

  - Where

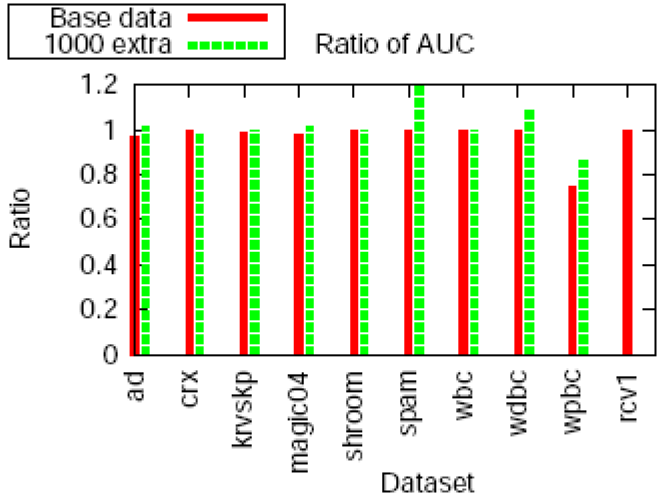
$$T_1(v_j, \alpha, \theta) = \begin{cases} \max(0, v_j - \alpha) & \text{if } v_j \in [0, \theta] \\ \min(0, v_j + \alpha) & \text{if } v_j \in [-\theta, 0] \\ v_j & \text{otherwise} \end{cases}.$$

## Empirical Results

- Left: the amount of features left after sparsification for each dataset without 1% performance loss
- AUC tolerated to 1.3%, achieve 91.4%



- the ratio of AUC with and without sparsification.
- AUC improve due to removal of some irrelevant features



## Empirical Results

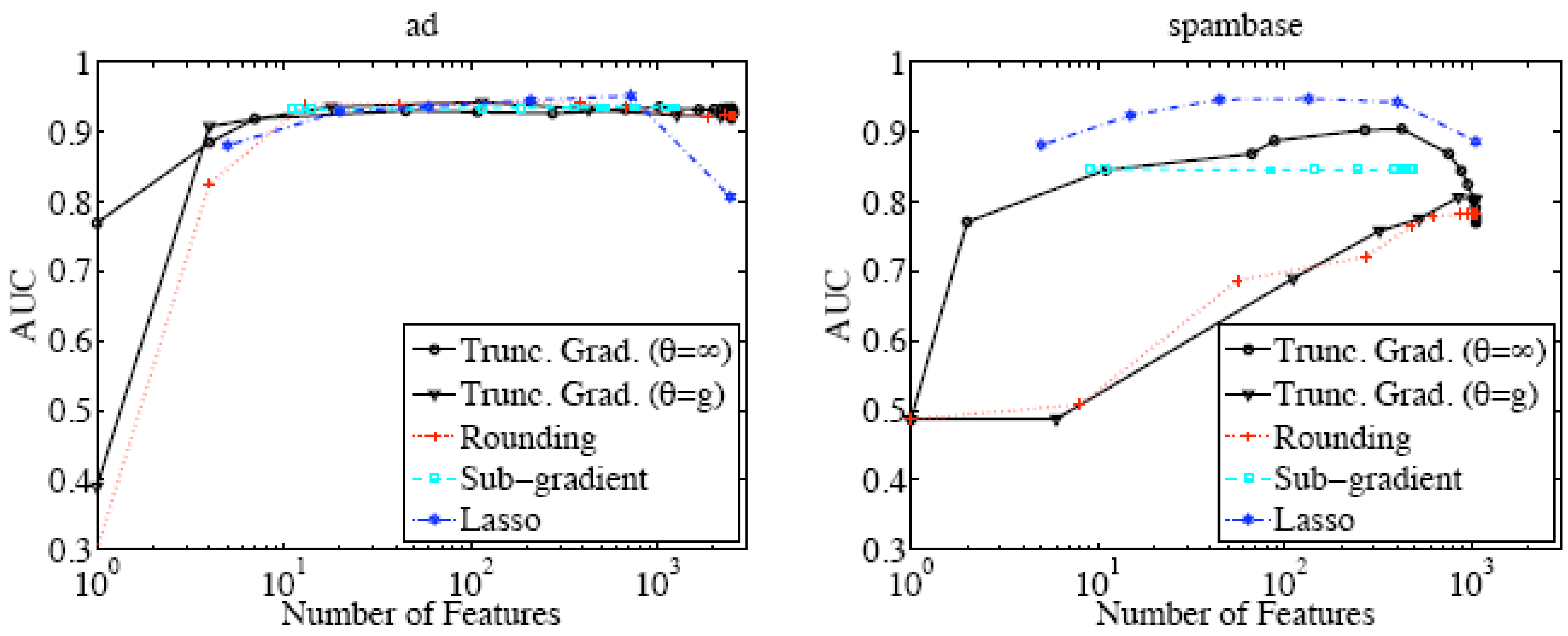
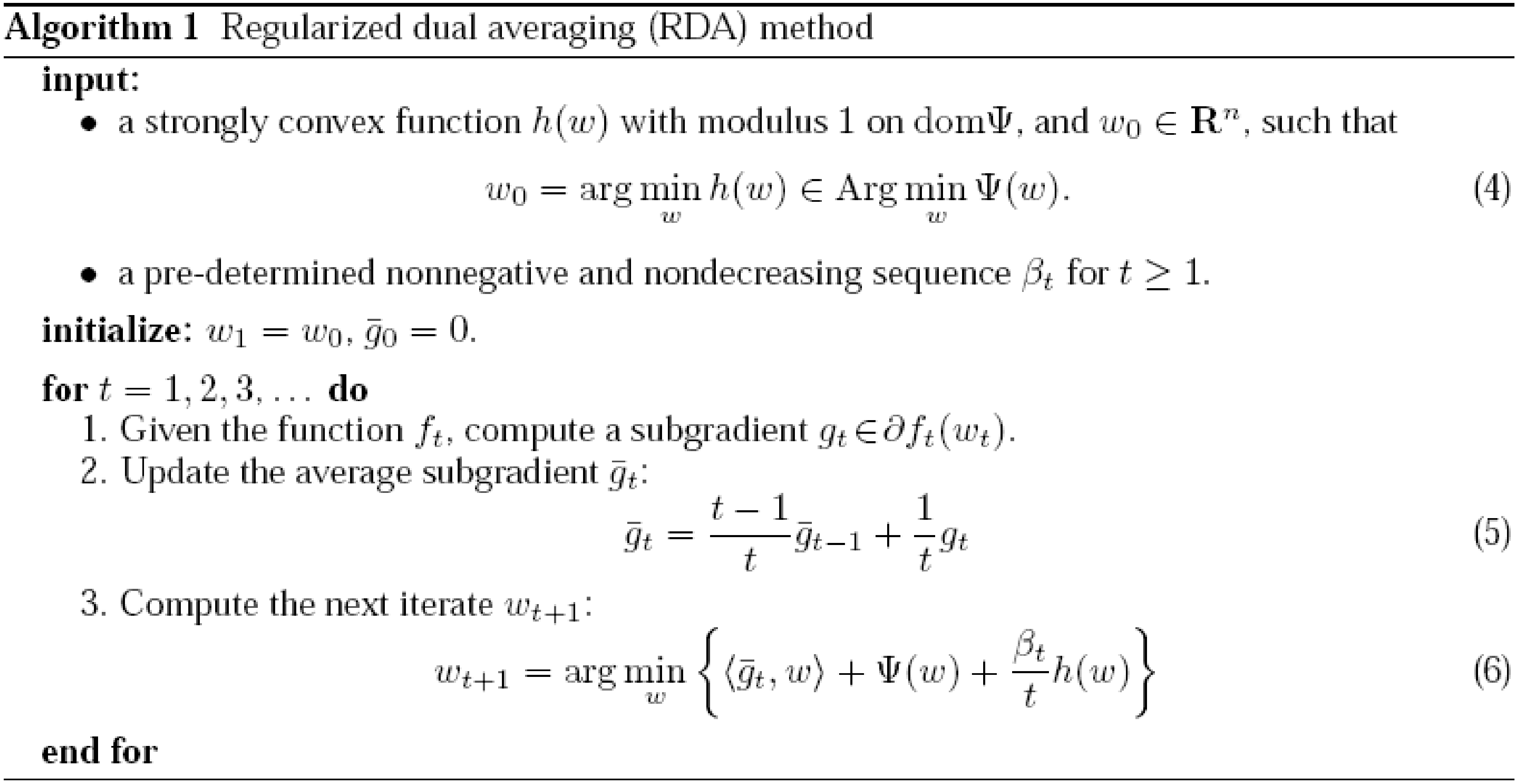


Figure 2: Comparison of the five algorithms in two sample UCI datasets.

- Truncate Gradient, efficient sparsification technique with strong thoretical guarantess

## Regularized Dual Averaging



**Algorithm 1** Regularized dual averaging (RDA) method

```
input:
    • a strongly convex function $h(w)$ with modulus 1 on $\mathrm{dom}\Psi$, and $w_0 \in \mathbf{R}^n$, such that
        $$w_0 = \arg\min_w h(w) \in \mathrm{Arg}\min_w \Psi(w). \qquad (4)$$
    • a pre-determined nonnegative and nondecreasing sequence $\beta_t$ for $t \geq 1$.
initialize: $w_1 = w_0$, $\bar{g}_0 = 0$.
for $t = 1, 2, 3, \ldots$ do
    1. Given the function $f_t$, compute a subgradient $g_t \in \partial f_t(w_t)$.
    2. Update the average subgradient $\bar{g}_t$:
        $$\bar{g}_t = \frac{t-1}{t}\bar{g}_{t-1} + \frac{1}{t}g_t \qquad (5)$$
    3. Compute the next iterate $w_{t+1}$:
        $$w_{t+1} = \arg\min_w \left\{ \langle \bar{g}_t, w \rangle + \Psi(w) + \frac{\beta_t}{t}h(w) \right\} \qquad (6)$$
end for
```

## Regularized Dual Averaging

- For simplicity, we do not need h(w)

*Mixed $\ell_1/\ell_2^2$-regularization.* Let $\Psi(w) = \lambda\|w\|_1 + (\sigma/2)\|w\|_2^2$ with $\lambda,\ \sigma > 0$. Then

$$w_{t+1}^{(i)} = \begin{cases} 0 & \text{if } \left|\bar{g}_t^{(i)}\right| \le \lambda, \\ -\frac{1}{\sigma}\left(\bar{g}_t^{(i)} - \lambda\,\mathrm{sign}(\bar{g}_t^{(i)})\right) & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, n.$$

Of course, setting $\lambda = 0$ gives the algorithm for pure $\ell_2^2$-regularization.

- Different from L1 regulariztion

$$f(w_i) = w_i - \eta\nabla_1 L(w_i, z_i) - \eta g\,\mathrm{sgn}(w_i),$$

  - this method does not produce sparse weights online simply because only in very rare cases do two floats add up to 0

## Computational Experiments

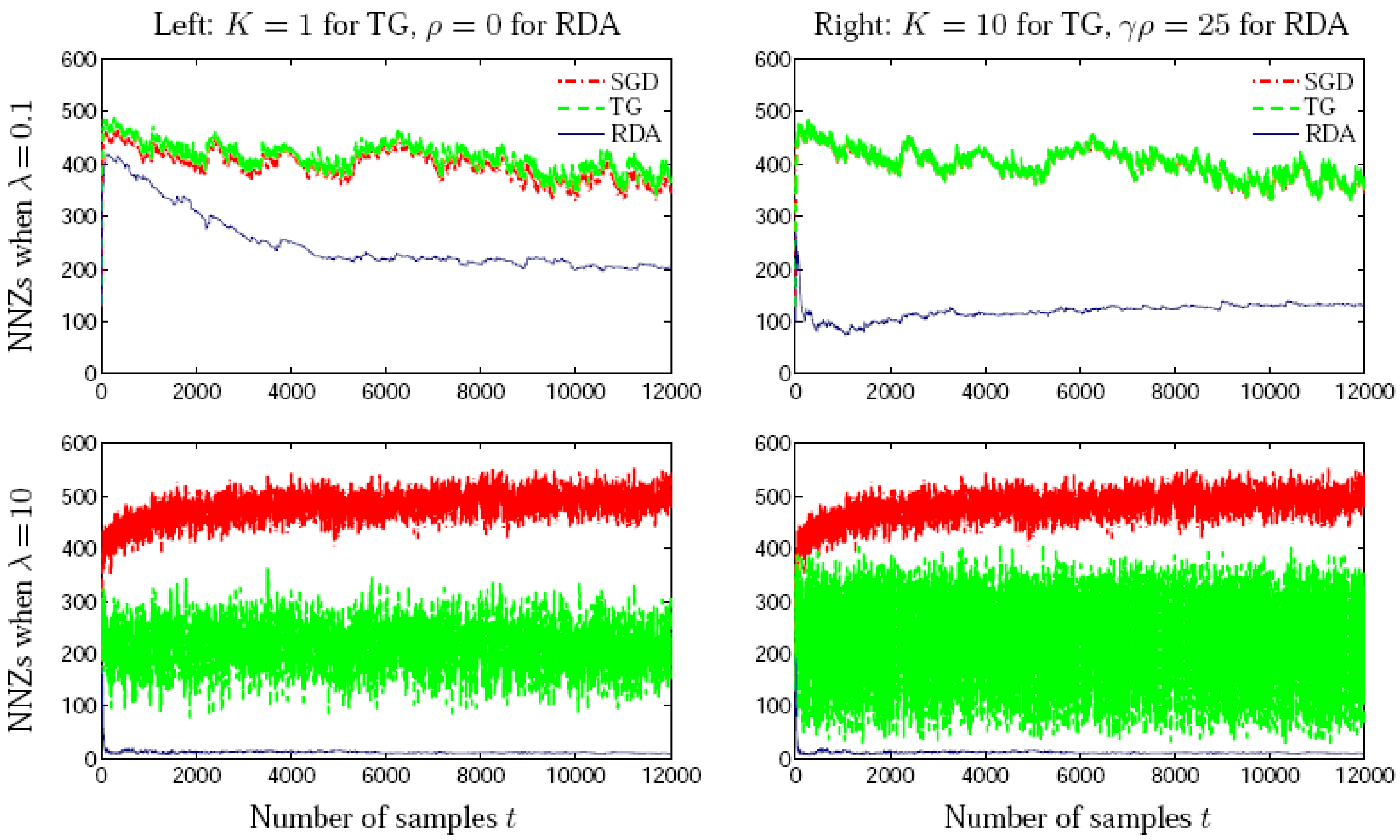


Figure 2: Number of non-zeros (NNZs) in $w(t)$ for the three online algorithms (classifying 6 and 7).

## TFRL-Proximal

- General online gradient descent performs the update

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t,$$

- Lazy representation of the model coefficients ω, using the "Follow The (Prox-imally) Regularized Leader" algorithm

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

  - Where $\mathbf{g}_{1:t} = \sum_{s=1}^{t} \mathbf{g}_s.$ $\sigma_{1:t} = \frac{1}{\eta_t}.$

  - Different form RDA

$$\Psi(w) = \lambda \|w\|_1 + (\sigma/2)\|w\|_2^2$$

## TFRL-Proximal

- New gradient update

$$\mathbf{w}_{t+1} = \begin{cases} 0 & \text{if } |z_t| \le \lambda_1 \\ -\eta_t(z_t - \operatorname{sgn}(z_t)\lambda_1) & \text{otherwise.} \end{cases}$$

  - Where

$$z_{t-1} = \mathbf{g}_{1:t-1} - \textstyle\sum_{s=1}^{t-1} \sigma_s \mathbf{w}_s$$

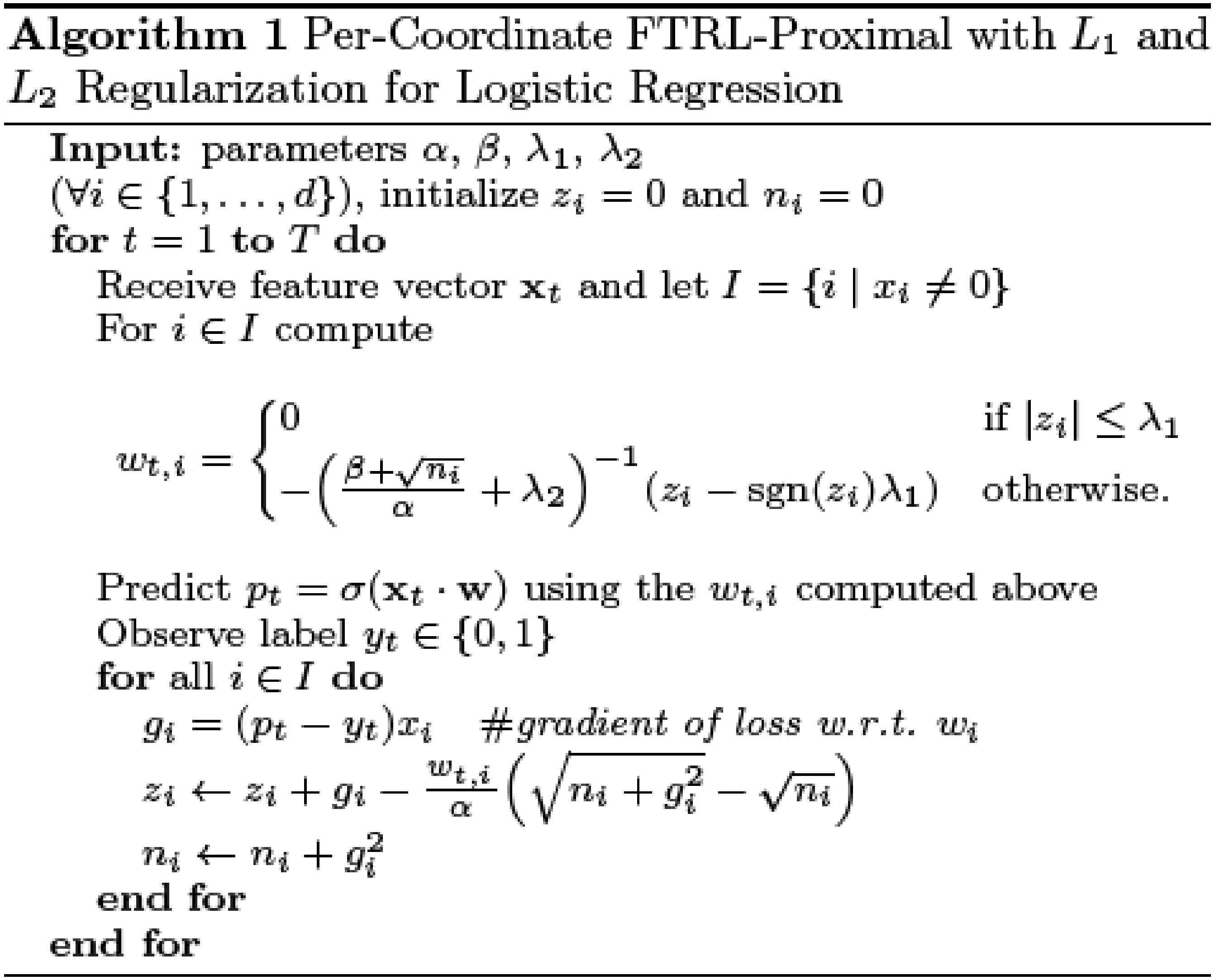
$$z_t = z_{t-1} + g_t + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\omega_t$$

- Per-Coordinate Learning Rates

$$\eta_{t,i} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^{t} g_{t,i}^2}}.$$

Bai du 推广

## TFRL-Proximal



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

```
Input: parameters $\alpha$, $\beta$, $\lambda_1$, $\lambda_2$
($\forall i \in \{1, \ldots, d\}$), 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. $w_i$
        $z_i \leftarrow z_i + g_i - \frac{w_{t,i}}{\alpha}\left(\sqrt{n_i + g_i^2} - \sqrt{n_i}\right)$
        $n_i \leftarrow n_i + g_i^2$
    end for
end for
```

## Empirical Results

| | Num. Non-Zero's | AucLoss Detriment |
|---|---|---|
| FTRL-Proximal | baseline | baseline |
| RDA | +3% | 0.6% |
| FOBOS | +38% | 0.0% |
| OGD-Count | +216% | 0.0% |

## 大纲

- 通用在线学习算法
  - 在线学习算法的应用背景
  - SGD、mini-batch算法简介（Andrew NG）
  - SGD算法的收敛特性（T. Zhang）
- SGD稀疏化方法
  - 稀疏化方法概要
  - Truncate Gradient
  - Regularized Dual Averaging
  - TFRL-Proximal
- 大规模在线学习系统实践
  - 节省内存
  - 有效地模型评估
  - 其它问题的探讨

## Saving Memory

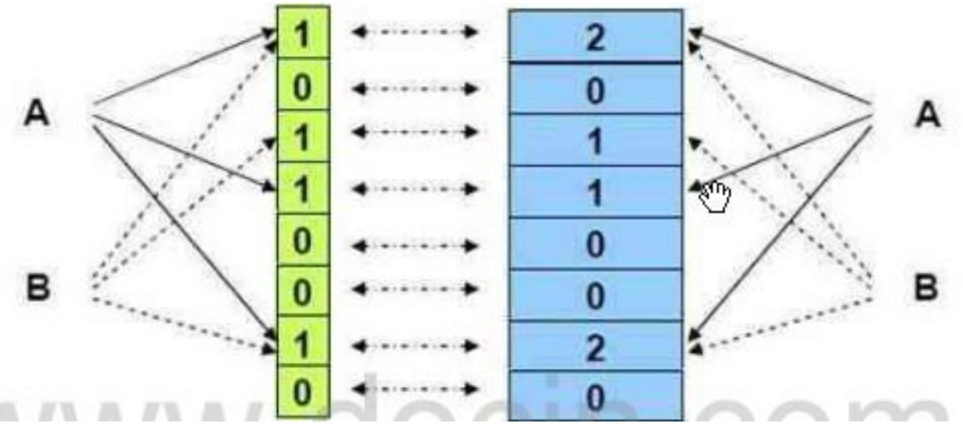
- Subsampling Training Data
  - Any query for which at least one of the ads was clicked.
  - A fraction $r \in (0, 1]$ of the queries where none of the ads were clicked.
  - Bias calibrating

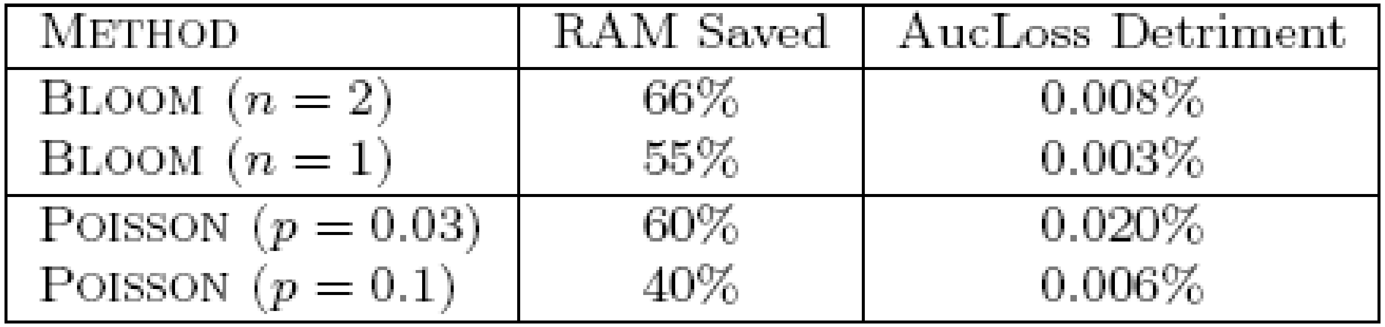
$$\omega_t = \begin{cases} 1 & \text{event } t \text{ is in a clicked query} \\ \frac{1}{r} & \text{event } t \text{ is in a query with no clicks.} \end{cases}$$

  - Applicable for positive and negative samples non-balance

## Saving Memory

- Probabilistic Feature Inclusion
  - Poisson Inclusion
  - Bloom Filter Inclusion

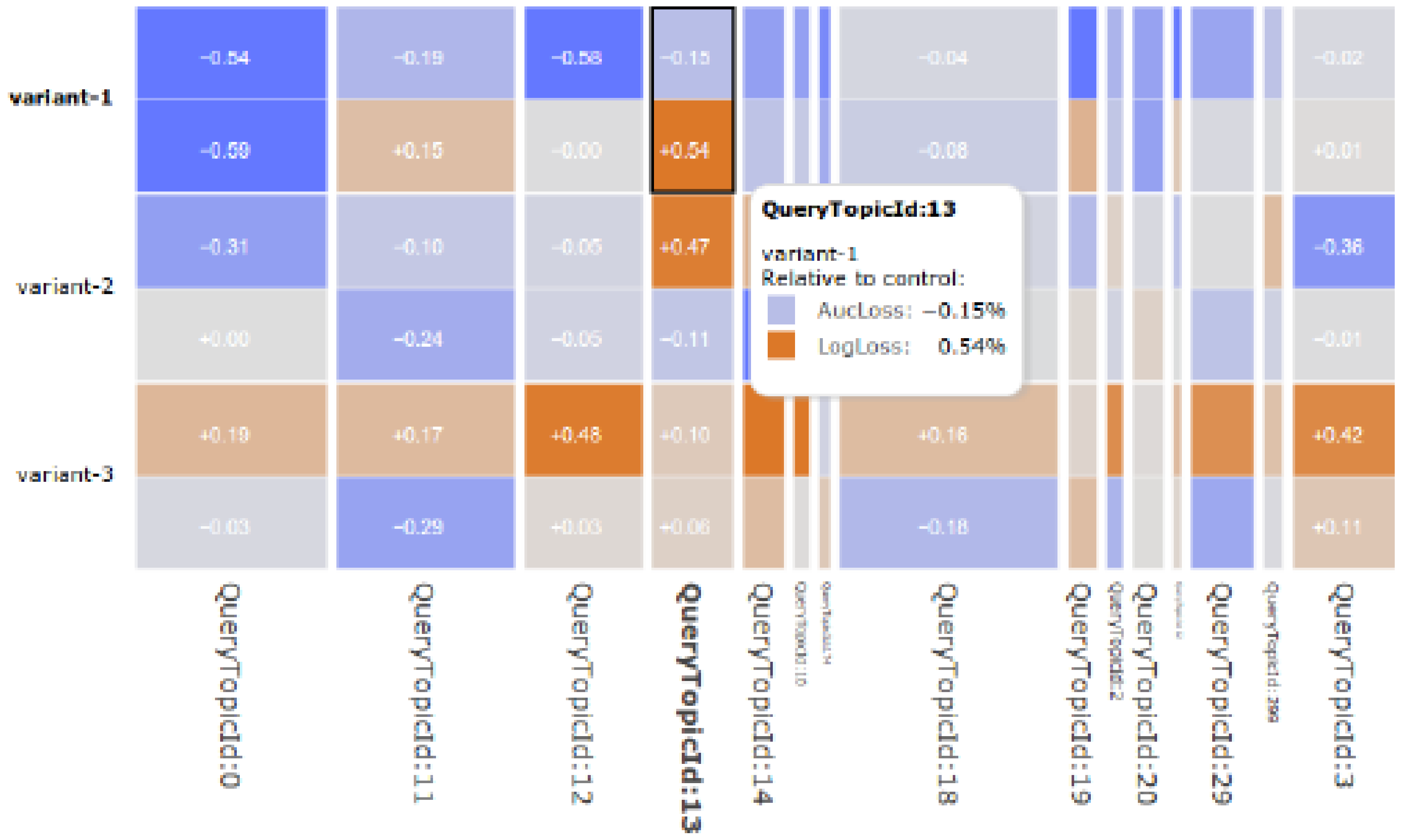




| Method | RAM Saved | AucLoss Detriment |
|---|---|---|
| Bloom ($n = 2$) | 66% | 0.008% |
| Bloom ($n = 1$) | 55% | 0.003% |
| Poisson ($p = 0.03$) | 60% | 0.020% |
| Poisson ($p = 0.1$) | 40% | 0.006% |

- Grafting maybe better
  - Scaling up machine learning: Parallel and distributed approaches.2011. P361

# Evaluating Model Performance



## 其它问题的探讨

- 计算平台的选择
  - Hadoop Map/Reduce 效率低
  - MPI 控制Streaming Data比较困难，节点之间耦合性高
  - 流式计算（Storm、S4）存在丢数据的风险，计算恢复困难
- 系统稳定性如何控制？

## 参考文献

- M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In Proceedings of ICML-03, pages 928–936, 2003.
- T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. InProceedings of ICML-04, pages 919–926, 2004.
- J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. JMLR, 10, 2009.
- L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. In NIPS, 2009.
- Ad Click Prediction: a View from the Trenches. H. Brendan McMahan, et al, ICML 2013
- R. Bekkerman, M. Bilenko, and J. Langford. Scaling up machine learning: Parallel and distributed approaches.2011.