ТЕХНОСФЕРА

Лекция 11
Large scale machine learning

Владимир Гулин

5 мая 2018 г.

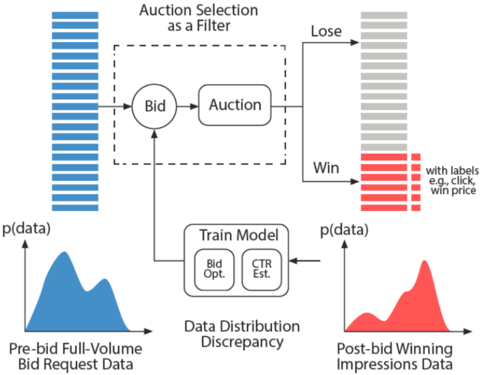
План лекции

Unbiased Learning

Large scale linear models

Large scale decision trees ensembles

Training with Instance Bias



$$\underbrace{p_x(\boldsymbol{x})}_{\text{bid request}} \cdot \underbrace{P(\text{win}|\boldsymbol{x}, b_{\boldsymbol{x}})}_{\text{auction selection}} = \underbrace{q_x(\boldsymbol{x})}_{\text{impression}}$$

$$w(b_{\boldsymbol{x}}) \equiv P(\text{win}|\boldsymbol{x}, b_{\boldsymbol{x}}) = \int_0^{b_{\boldsymbol{x}}} p_z^{\boldsymbol{x}}(z)dz$$

Unbiased Learning

$$l(b_x) = \prod_{b_j < b_x} \frac{n_j - d_j}{n_j}$$

$$w(b_x) = 1 - \prod_{b_j < b_x} \frac{n_j - d_j}{n_j}$$

Table 3.1: An example of data transformation of 8 instances with bid price between 1 and 4. Left: tuples of bid, win and cost $\langle b_i, w_i, z_i \rangle_{i=1...8}$. Right: transformed survival model tuples $\langle b_j, d_j, n_j \rangle_{j=1...4}$ and the calculated winning probabilities. Here we also provide a calculation example of $n_3 = 4$ shown as blue in the right table. The counted cases of n_3 in the left table are 2 winning cases with $z \geq 3 - 1$ and the 2 losing cases with $b \geq 3$, shown highlighted in blue colour. Source [Zhang et al., 2016e].

b_i	w_i	z_i
2	win	1
3	win	2
2	lose	×
3	win	1
3	lose	×
4	lose	×
4	win	3
1	lose	×

b_j	n_j	d_j	$\frac{n_j - d_j}{n_j}$	$w(b_j)$	$w_o(b_j)$
1	8	0	1	$1 - 1 = 0$	0
2	7	2	$\frac{5}{7}$	$1 - \frac{5}{7} = \frac{2}{7}$	$\frac{2}{4}$
3	4	1	$\frac{3}{4}$	$1 - \frac{5}{7}\frac{3}{4} = \frac{13}{28}$	$\frac{3}{4}$
4	2	1	$\frac{1}{2}$	$1 - \frac{5}{7}\frac{3}{4}\frac{1}{2} = \frac{41}{56}$	$\frac{4}{4}$

Unbiased Learning

$$\min_{\boldsymbol{\theta}} \quad \mathbb{E}_{\boldsymbol{x}\sim p_x(\boldsymbol{x})}[\mathcal{L}(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))] + \lambda\Phi(\boldsymbol{\theta})$$

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}\sim p_x(\boldsymbol{x})}[\mathcal{L}(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))] = \int_{\boldsymbol{x}} p_x(\boldsymbol{x})\mathcal{L}(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))d\boldsymbol{x} \\
&= \int_{\boldsymbol{x}} q_x(\boldsymbol{x})\frac{\mathcal{L}(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{w(b_{\boldsymbol{x}})}d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x}\sim q_x(\boldsymbol{x})}\left[\frac{\mathcal{L}(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{w(b_{\boldsymbol{x}})}\right] \\
&= \frac{1}{|D|}\sum_{(\boldsymbol{x},y,z)\in D}\frac{\mathcal{L}(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{w(b_{\boldsymbol{x}})} = \frac{1}{|D|}\sum_{(\boldsymbol{x},y,z)\in D}\frac{\mathcal{L}(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{1-\prod_{b_j<b_{\boldsymbol{x}}}\frac{n_j-d_j}{n_j}}.
\end{aligned}$$

$$\min_{\boldsymbol{\theta}} \frac{1}{|D|}\sum_{(\boldsymbol{x},y,z)\in D}\frac{-y\log f_{\boldsymbol{\theta}}(\boldsymbol{x}) - (1-y)\log(1-f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{1-\prod_{b_j<b_{\boldsymbol{x}}}\frac{n_j-d_j}{n_j}} + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$$

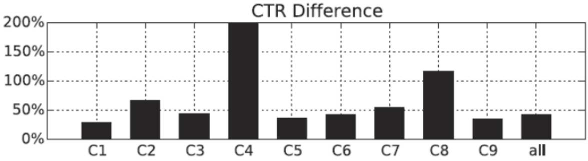
Вопрос:

- Как будет выглядеть правило обновления весов в линейной модели?

Unbiased CTR Estimation Learning

Table : Online A/B testing of CTR estimation (Yahoo!).

Camp.	BIAS AUC.	KMMP AUC	AUC Lift
C1	63.78%	64.12%	0.34%
C2	87.45%	88.58%	1.13%
C3	69.73%	75.52%	5.79%
C4	88.82%	89.55%	0.73%
C5	69.71%	72.29%	2.58%
C6	89.33%	90.70%	1.37%
C7	77.76%	78.92%	1.16%
C8	74.57%	76.98%	2.41%
C9	71.04%	73.12%	2.08%
all	73.48%	76.45%	2.97%



Как обучаться на больших данных?

Линейная модель

$$P(click|x) = \frac{1}{1 + exp(-w^T x)}$$

Как ее обучать?

Как обучаться на больших данных?

Линейная модель

$$P(click|x) = \frac{1}{1 + exp(-w^T x)}$$

Online learning

Инициализировать $w = 0$
Повторять:

1. Получить признаки $x \in R^n$
2. Вычислить предсказания $\hat{p}(x) = \sigma(w^T x)$
3. Обновить вектор весов так, чтобы $\hat{p}(x)$ стало ближе к y.

$$w_i \to w_i - \eta \frac{\partial L(\hat{p}(x), y)}{\partial w_i}$$

Distributed online learning

Входные данные:

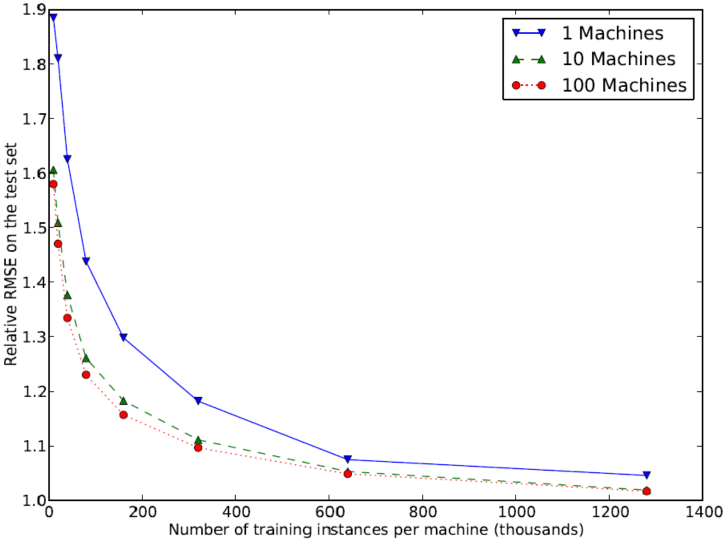
- Обучающая выборка X
- K машин
- Число итераций T

1. Случайно разбить выборку по примерам на K частей: $X^k,\ k = 1, \dots, K$
2. $w^k \to 0, k = 1, \dots, K$
3. Выполнить $t = 1, \dots, T$
4. Выполнить на каждой машине:
5. $w^k \to$ результат онлайн обучения на X^k
6. $w_i \to \sum\limits_{k=1}^{K} A_i^k w_i^k$

Варианты агрегации:

- $A_i^k = \frac{1}{K}$
- $A_i^k = \frac{G_i^k}{\sum\limits_{u=1}^{K} G_i^u}, \quad G_i^u = \sum\limits_{j=1}^{N} \left(\frac{\partial L(\hat{p}(x_j), y_j)}{\partial w_i} \right)^2$

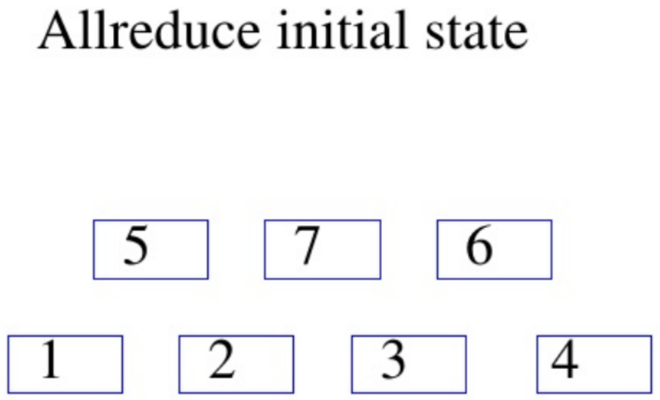
Ускорение онлайн обучения



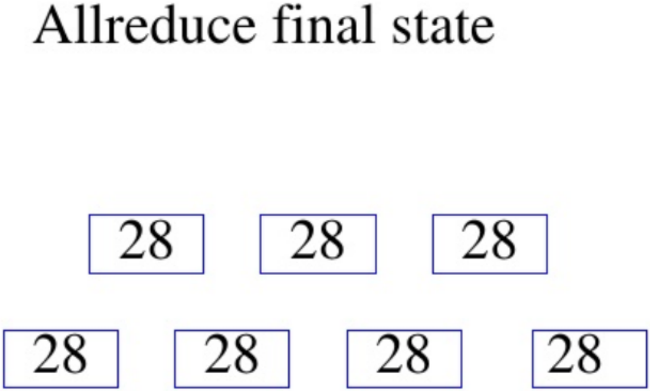
Вопрос:

Как учить линейные модели на hadoop?

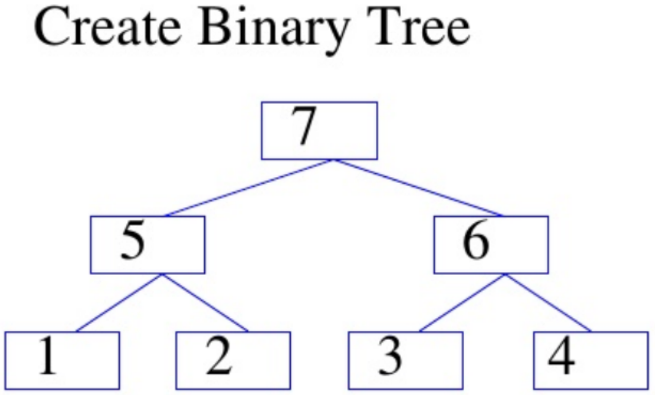
MPI AllReduce



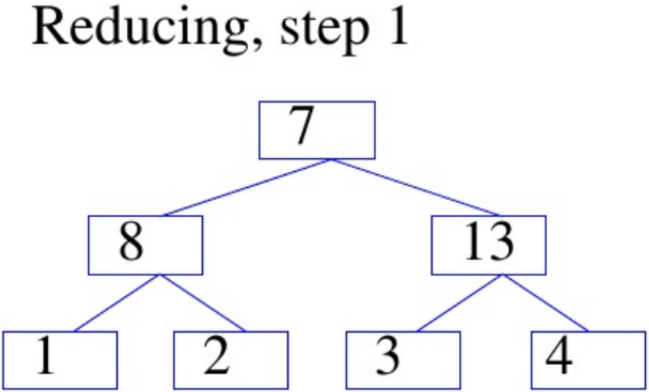
MPI AllReduce



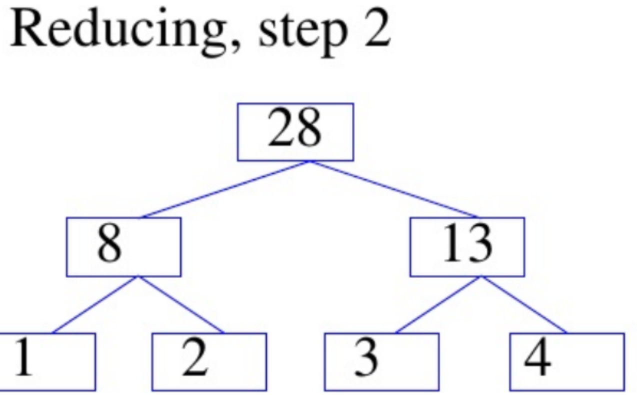
MPI AllReduce



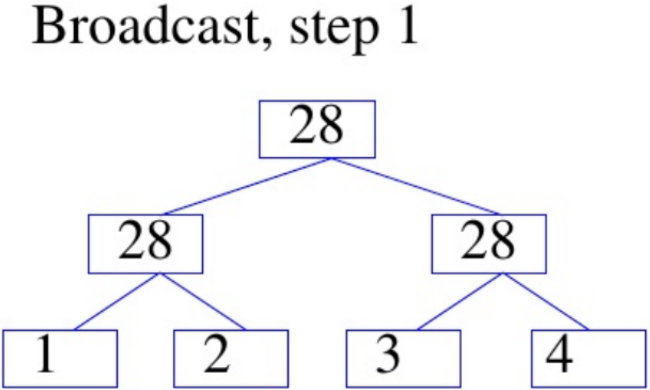
MPI AllReduce



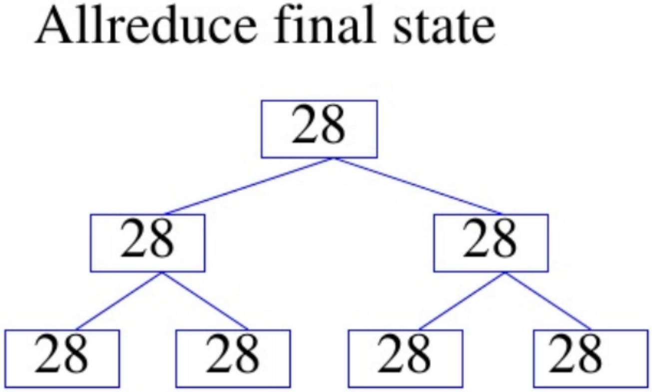
MPI AllReduce



MPI AllReduce



MPI AllReduce



AllReduce = Reduce+Broadcast

Не нужно править код (SPMD)

Метод градиентного спуска

Входные данные:

- Обучающая выборка X
- $w^k \to 0$
- η - шаг обучения

1. Пока не выполнен критерий останова:
2. Вычислить градиент $g = \nabla L(w)$
3.
4. $w \to w - \eta g$
5. Вернуть w

Не нужно править код (SPMD)

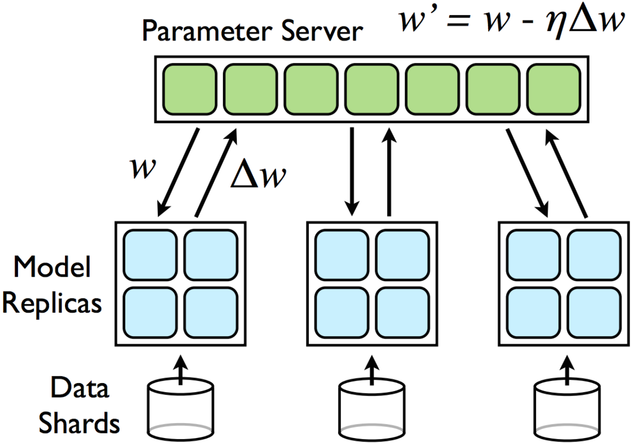
Метод градиентного спуска

Входные данные:

- Обучающая выборка X
- $w^k \to 0$
- η - шаг обучения

1. Пока не выполнен критерий останова:
2. Вычислить градиент $g = \nabla L(w)$
3. $g \to AllReduce(g)$
4. $w \to w - \eta g$
5. Вернуть w

Parameter Server



Как экономить память?

Локальное обновление весов в лог. регрессии

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

Локальное обновление весов в лог. регрессии

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

$$\mathbf{g}_{1:t} = \sum_{s=1}^{t} \mathbf{g}_s, \quad \sigma_{1:t} = \frac{1}{\eta_t}, \quad \eta_t = \frac{1}{\sqrt{t}}$$

Можно записать как

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

FTRL Proximal

Follow The Regularized Leader

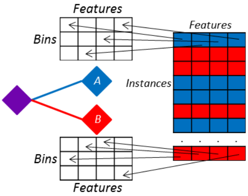
Algorithm 1 Per-Coordinate FTRL-Proximal with L_1 and L_2 Regularization for Logistic Regression

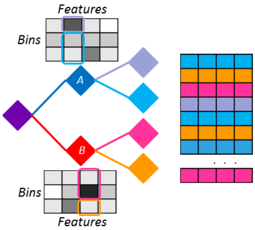
 # With per-coordinate learning rates of Eq. (2).
Input: parameters α, β, λ_1, λ_2
$(\forall i \in \{1, \dots, d\})$, initialize $z_i = 0$ and $n_i = 0$
for $t = 1$ **to** T **do**
  Receive feature vector $\mathbf{x}_t$ and let $I = \{i \mid x_i \neq 0\}$
  For $i \in I$ compute

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

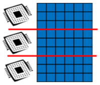
  Predict $p_t = \sigma(\mathbf{x}_t \cdot \mathbf{w})$ using the $w_{t,i}$ computed above
  Observe label $y_t \in \{0, 1\}$
  for all $i \in I$ **do**
    $g_i = (p_t - y_t) x_i$ *#gradient of loss w.r.t.* w_i
    $\sigma_i = \frac{1}{\alpha}\left(\sqrt{n_i + g_i^2} - \sqrt{n_i}\right)$ *#equals* $\frac{1}{\eta_{t,i}}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

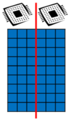
Distributed tree construction





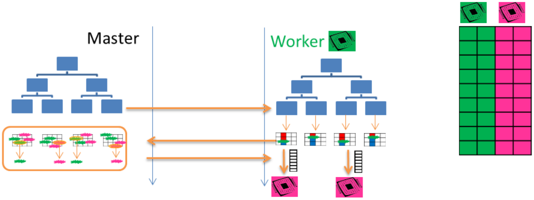
- Наблюдение 1: Одного прохода по данным достаточно на каждый уровень дерева
- Наблюдение 2: Итерироваться можно либо по точкам, либо по фичам





Distributed tree construction

Feature Distributed



Master

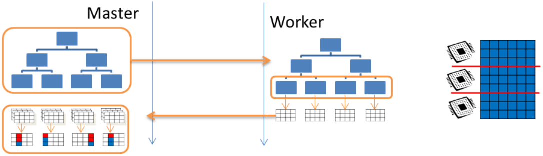
- Request workers to expand a set of nodes
- Wait to receive best per-feature splits from workers
- Select best feature-split for every node
- Request best splits' workers to broadcast per-instance assignments and residuals

Worker

- Pass through all instances for local features, aggregating split histograms for each node
- Select local features' best splits for each node, send to master

Distributed tree construction

Data Distributed



Master

- Send workers current model and set of nodes to expand
- Wait to receive local split histograms from workers
- Aggregate local split histograms, select best split for every node

Worker

- Pass through local data, aggregating split histograms
- Send completed local historograms to master

Distributed tree construction

Data Distributed for sparse features

Algorithm 2 FindBestSplit

```
Input: DataSet D
for all X in D.Attribute do
    ▷ Construct Histogram
    H = new Histogram()
    for all x in X do
        H.binAt(x.bin).Put(x.label)
    end for
    ▷ Find Best Split
    leftSum = new HistogramSum()
    for all bin in H do
        leftSum = leftSum + H.binAt(bin)
        rightSum = H.AllSum - leftSum
        split.gain = CalSplitGain(leftSum, rightSum)
        bestSplit = ChoiceBetterOne(split,bestSplit)
    end for
end for
return bestSplit
```

Algorithm 3 PV-Tree_FindBestSplit

```
Input: Dataset D
localHistograms = ConstructHistograms(D)
▷ Local Voting
splits = []
for all H in localHistograms do
    splits.Push(H.FindBestSplit())
end for
localTop = splits.TopKByGain(K)
▷ Gather all candidates
allCandidates = AllGather(localTop)
▷ Global Voting
globalTop = allCandidates.TopKByMajority(2*K)
▷ Merge global histograms
globalHistograms = Gather(globalTop, localHistograms)
bestSplit = globalHistograms.FindBestSplit()
return bestSplit
```

Вопросы