Coupled Group Lasso for Web-Scale CTR Prediction in Display Advertising (Yan et al.)

Presented by Jongjin Lee.

Seoul National University

ga0408@snu.ac.kr

April 25, 2018

CTR Prediction

- CTR : Click Through Rate
- Estimating the probalility that an advertisement is clicked when displayed to a user in a specific context
- Web-scale CTR Prediction in display advertising(large scale data sets)

Notation and Task

- Estimating $P(Y = 1|X)$
- $x^T = ({x_u}^T, {x_a}^T, {x_o}^T)$
  $Y = 1$ *or* 0 whether the ad is clicked
- Focus on the senarios we can collect both user and ad features.
  - User : job, buying history of other products, ...
  - Advertisement : description words, ...
  - Context : daytime, weekdays, window size ...

Logistic Regression

- Due to its easy implementation and promising performance, LR model has been widely used for CTR prediction
- Logistic Regression

$$h(x) = P(y = 1|x, W, V, b) = \frac{1}{1 + exp(-W^T x)}$$

- The loss

$$\sum_{i=1}^{N} \xi(W, V, B; , x^{(i)}, y^{(i)}) + \lambda\Omega(W, V)$$

$$,\xi(W, V, B; , x^{(i)}, y^{(i)}) = -log([h(x^{(1)})]^{y^{(i)}}[1 - h(x^{(i)})]^{1-y^{(i)}})$$

$$,\Omega(W, V) \ is \ regularization \ term$$

- It can not capture the conjucntion information between user features and ad features

Coupled Group Lasso

- The likelihood of CGL is formulated as follows

$$h(x) = P(y = 1|x, W, V, b)$$
$$= \sigma(({x_u}^T W ({x_a}^T V)^T + b^T x_o), \ \sigma(x) = \frac{1}{1 + exp(-x)}$$

- Loss is as follows

$$\sum_{i=1}^{N} \xi(W, V, B; , x^{(i)}, y^{(i)}) + \lambda\Omega(W, V)$$

$$,\xi(W, V, B; , x^{(i)}, y^{(i)}) = -log([h(x^{(1)})]^{y^{(i)}}[1 - h(x^{(i)})]^{1-y^{(i)}})$$

$$,\Omega(W, V) = \|W\|_{2,1} + \|V\|_{2,1}, \|M\|_{2,1} = \sum_{i=1}^{l} \sqrt{\sum_{j=1}^{k} M_{ij}^2}$$

- W,V,B is $l \times k$ matrix, $s \times k$ matrix, d vector
- k ,λ is hyperparameter

Advantages of CGL

- CGL can capture the conjunction information from user features and ad features.
  - $x_u{}^T W (x_u{}^T V)^T = x_u^T (W V^T) x_a$
- CGL can automatically eliminate useless features for both users and ads, which may facilitate fast online prediction.
  - Each row is a group.

Learning

- ${x_u}^T W ({x_u}^T V)^T$ makes objective function non-convex
- Each time we optimize one parameter with other parameters fixed
- First fix V optimize(L-BFGS) W,b until converge, next fix W, ...
  $\rightarrow$ objective function convex

Algorithm

Algorithm 1 Alternate Learning for CGL

Input: Data set $\{(\mathbf{x}^{(i)}, y^{(i)}) \mid i = 1, ..., N\}$, and hyper-parameters $k \in \mathcal{N}^+$ and $\lambda \in \mathbb{R}^+$.
Output: $\mathbf{W}^*, \mathbf{V}^*, \mathbf{b}^*$
Initialize $\mathbf{b} = \mathbf{0}$.
Initialize $\mathbf{W} = random(\mathbb{R}^{l \times k})$, $\mathbf{V} = random(\mathbb{R}^{s \times k})$.
repeat
  Fix $\mathbf{V}$.
  repeat
    Compute gradient $\mathbf{g}(\mathbf{W}, \mathbf{b})$
    Compute the approximate Hessian $\tilde{\mathbf{H}}_{\mathbf{W},\mathbf{b}}$ w.r.t. $(\mathbf{W}, \mathbf{b})$.
    $\mathbf{d}(\mathbf{W}, \mathbf{b}) = -\tilde{\mathbf{H}}_{\mathbf{W},\mathbf{b}} * \mathbf{g}(\mathbf{W}, \mathbf{b})$.
    Perform line search in the direction of $\mathbf{d}(\mathbf{W}, \mathbf{b})$ and update $\mathbf{W}, \mathbf{b}$.
  until convergence on $\mathbf{W}, \mathbf{b}$
  Fix $\mathbf{W}$.
  repeat
    Compute gradient $\mathbf{g}(\mathbf{V}, \mathbf{b})$
    Compute the approximate Hessian $\tilde{\mathbf{H}}_{\mathbf{V},\mathbf{b}}$ w.r.t. $(\mathbf{V}, \mathbf{b})$.
    $\mathbf{d}(\mathbf{V}, \mathbf{b}) = -\tilde{\mathbf{H}}_{\mathbf{V},\mathbf{b}} * \mathbf{g}(\mathbf{V}, \mathbf{b})$.
    Perform line search in the direction of $\mathbf{d}(\mathbf{V}, \mathbf{b})$ and update $\mathbf{V}, \mathbf{b}$.
  until convergence on $\mathbf{V}, \mathbf{b}$
until convergence

Figure: Algorithm

Web-Scale implementation: Hashing

- Web-scale applications always contain a huge number of users and ad, with billions of impression instances.
- The data are mainly categorical, the number of which is typically very large.
- Using hashing technique for efficient feature mapping and istance generating.

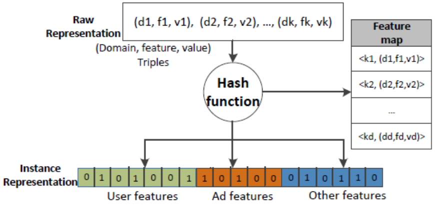

Figure: The hashing framework

Web-Scale implementaition: Sub-sampling

- The data sets are typically highly unbalanced, with only a very small proportion of positive instances.
- Sample negative instances with a probability of $\gamma = 10\%$ and keep all the positive instances.
- After sampling, give a weight $\frac{1}{\gamma}$ to each negative instance during learning to make the objective calculation unbiased

Web-Scale implementaion: Distributed Learning

- Need to compute the gradient of all the paremeters.
- Implement a distributed learning framework : MPI(Message Passing Inference)
- Master node, Slaver nodes.
  - Evenly distribute the whole data set to each node(number of P).
  - Calculate gradient $g_p^{'} = \sum_{i=1}^{p_n} \frac{\partial \xi}{\partial t},\ t = W_{ij}\ or\ V_{ij}$

Experiment on real data

- Three data sets from Taobao of Alibaba group
  - Three datasets contain log information of display ads across different time preriods with different time window sizes
  - The subsequent day's log information is used as test data
- Three datasets contain training data of 4 days, 10 days, and 7 days from difrrent time periods, respectively

Data set	# Instances (in Billion)	CTR (in %)	# Ads	# Users (in Million)	Storage (in TB)
Train 1	1.011	1.62	21,318	874.7	1.895
Test 1	0.295	1.70	11,558	331.0	0.646
Train 2	1.184	1.61	21,620	958.6	2.203
Test 2	0.145	1.64	6,848	190.3	0.269
Train 3	1.491	1.75	33,538	1119.3	2.865
Test 3	0.126	1.70	9,437	183.7	0.233

Figure: Datasets

- MPI-cluster with 80 nodes, each of which is a 24-corserver with 2.2GHz ...

Experiment on real data

- $RelaImpr = \frac{AUC(model)-0.5}{AUC(baseline)-0.5}$

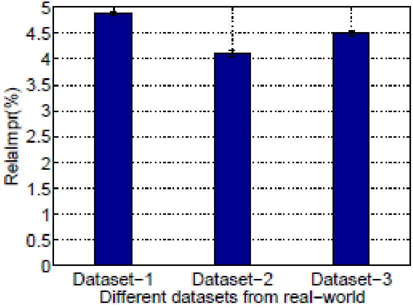

Figure: RelaImpr

Experiment on real data

- Hyperparamters k, λ
- Larger k implies more parmeters. $\rightarrow$ because of memory and speed. Choose k=50
- λ controls the tradeoff between the prediction accuract and number of eliminated features

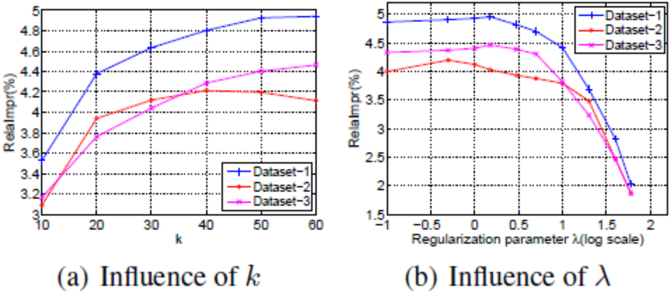

Figure: GSparsity

Experiment on real data

- $GSparsity = \frac{v}{l+s} \times 100\%$
  ,v is the total number of all-zero rows in parameter matrices W and V.
- A GSparsity of 3% - 15% will be a godd trade off for both feature elimination and prediction accuracy
  $\rightarrow$ choose corresponding λ

GSPARSITY	2%	3%	5%	15%	20%
RELAIMPR	3.90%	3.42%	3.02%	2.5%	1.97%

Figure: GSparsity for Dataset-2

The End

The End