Large-Scale Behavioral Targeting

Ye Chen*, Dmitry Pavlov†, John Canny‡

*eBay, †Yandex, ‡UC Berkeley
(This work was conducted at Yahoo! Labs.)

June 30, 2009

Agenda

1. Introduction
2. Linear Poisson Regression
3. Large-Scale Implementation
   - Data Reduction and Information Loss
   - Feature Selection and Indexing
   - Feature Vector Generation in $\mathcal{O}(1n)$
   - Data-driven Weight Initialization
   - Parallel Multiplicative Recurrence
4. Experiments
   - Setup
   - Results
5. Conclusion

Introduction

- Behavioral targeting (BT)
  - leverages historical user behavior to select the most relevant ads.
  - y: predicts click-through rate (CTR).
  - $\mathbf{x}$: ad clicks and views, page views, search queries and clicks.
- Challenges:
  - large scale, e.g., Y! logged 9TB ad data with 500B entries on Aug'08.
  - sparse, e.g., the CTR of automotive display ads is 0.05%.
  - dynamic, i.e., user behavior changes over time.

Non-negative Linear Poisson Regression

- Poisson distribution for counts:

$$p(y) = \frac{\lambda^y \exp(-\lambda)}{y!}, \text{where } \lambda = \mathbf{w}^\top \mathbf{x}.$$

- MLE by multiplicative recurrence:

$$w'_j \leftarrow w_j \frac{\sum_i \frac{y_i}{\lambda_i} x_{ij}}{\sum_i x_{ij}}, \text{where } \lambda_i = \mathbf{w}^\top \mathbf{x}_i.$$

- CTR prediction:

$$\widehat{\mathsf{CTR}}_{ik} = \frac{\lambda_{ik}^{\mathsf{click}} + \alpha}{\lambda_{ik}^{\mathsf{view}} + \beta}.$$

- Notation:

y,λ	the observed and expected counts.
$\mathbf{w}$,$\mathbf{x}$	the weight and bag-of-words feature vector.
i, j, k	the indices of user, feature, and ad category.
α, β	the smoothing constants for clicks and views.

Data Reduction and Information Loss

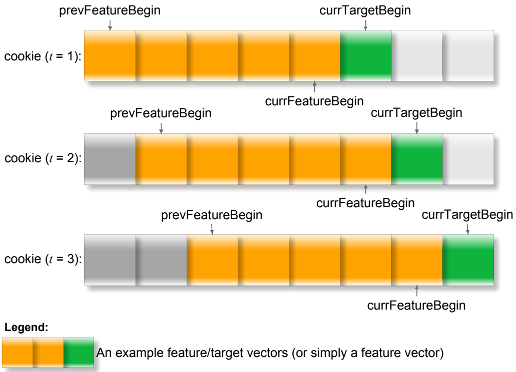
- Many practical learning algorithms are IO-bound and scan-bound.
- For BT, one needs to preprocess 20-30TB raw data feeds of ads and searches.
- Reduce data size at the earliest, by projection, aggregation and merging, e.g., on (cookie, time).
- Data prep should have minimum information loss and redundancy, e.g., time resolution.
- Data prep should be loosely coupled with specific modeling logics for data reusability, e.g., neither decays counts nor categorizes ads.
- After preprocessing, the data size is reduced to 2-3TB.

Feature Selection and Indexing

- A data-driven approach is to use granular events as features.
- Frequency-based feature selection works best in practice for sparse data.
- Frequency is counted in cookie rather than event occurrence (robot filtering).
- Thresholding immediately after summing Mapper, locally and in-memory, thus cut the long tail of the power-law like sparse data.
- Output of feature selection is three dictionaries (ads, pages, queries), which collectively define an indexing of the feature space.

Feature Vector Generation in $\mathcal{O}(1n)$

- Linear time algorithms are of great interest for large-scale learning.
- The scalar c of a linear complexity $O(cn)$ should be seriously taken into account when n is easily in the order of billion.
- To generate $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ in $O(1n)$ time:



Data-driven Weight Initialization

- To exploit the sparseness, one shall use some data-driven approaches.
  1. feature-specific normalization (the idea of tf-idf):

$$w_{kj} \leftarrow \frac{\sum_i \frac{y_{ik} x_{ij}}{\sum_{j'} x_{ij'}}}{\sum_i x_{ij}}.$$

  2. target-specific normalization (respect the highly skewed distribution of traffic over categories):

$$w_{kj} \leftarrow \frac{\sum_i (y_{ik} x_{ij}) \sum_i y_{ik}}{\sum_{j'} \left[\sum_i (y_{ik} x_{ij'}) \sum_i x_{ij'}\right]}.$$

Parallel Multiplicative Recurrence

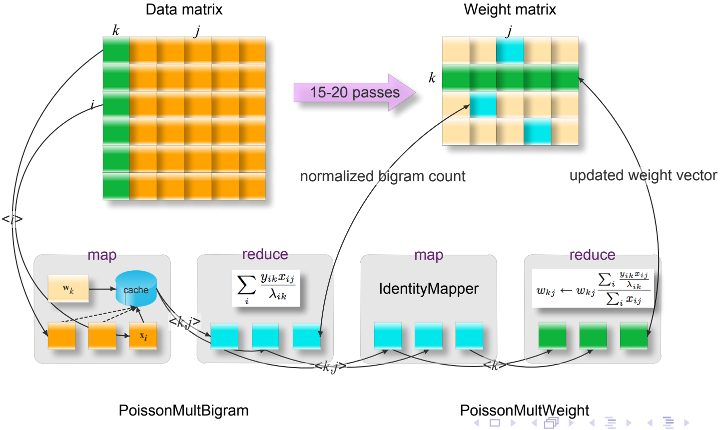
- Given $D = [Y \;\; X]$, solve $W^* = \text{argmax}_W \log p(Y^\top | W X^\top)$.
- An NMF problem $Y^\top \approx W X^\top$ where the quality of factorization is measured by log likelihood.
- Multiplicative update:

$$w'_j \leftarrow w_j \frac{\sum_i \frac{y_i}{\lambda_i} x_{ij}}{\sum_i x_{ij}}, \text{where } \lambda_i = \mathbf{w}^\top \mathbf{x}_i.$$

- Computational bottleneck: $\sum_i \frac{y_i}{\lambda_i} x_{ij}$.
- Parallel iterative algorithms typically suffer from synchronizing model parameters after each iteration.
- For BT, the final multiplicative update of $\mathbf{w}_k$ has to be carried out in a single node.

"Fine-grained Parallelization"

- Scalable data structures: $(\mathbf{x}_i, \mathbf{y}_i)$ sparse vectors, $\mathbf{w}_k$ dense vectors.
- Distribute counting co-occurrences by (k, j) which defines an entry in W.
- In-memory cache input examples (*not weights*), and retrieve relevant weight vectors on demand.



Setup

- Training data: 5-week full-scale Y! user behavioral data, 500 millions training examples, 3TB preprocessed and compressed data.
- Feature space: 150K features comprised of 40K ads ($\times 2$), 40K pages, and 10K queries ($\times 3$).
- Sliding windows: one-day target window over one-week, and 4-week feature window.
- Evaluation: an $1/16$ sample from the next day, and a 6-minute latency between a 5-week feature window and a 6-minute target window.
- Metrics: (1) relative CTR lift, (2) click-view ROC curve, and (3) run-time.
- Baseline model: a sign-constrained linear regression with categorized features.
- Cluster: a 500-node Hadoop cluster of commodity machines ($2\times$ Quad Core 2GHz CPU and 8GB RAM).

Data Size

- To scale up to the entire Yahoo's user data:

Table: The Effect of Training Data Size

# Buckets	32	64	128	256	512
CTR lift	0.1583	0.2003	0.2287	0.2482	0.2598
ROC area	0.8193	0.8216	0.8234	0.8253	0.8267
Run-time	2.95	3.78	6.95	7.43	14.07

Feature Selection

- To examine different feature dimensionalities:

Table: The Effect of Feature Dimensionality

# Features	60K	90K	150K	270K	1.2M
CTR lift	0.2197	0.2420	0.2598	0.2584	0.2527
ROC area	0.8257	0.8258	0.8267	0.8267	0.8261
Run-time	14.87	13.52	14.07	13.08	16.42

Feature Vector Generation

- To verify the scalability of feature vector generation:

Table: Linear-time Feature Vector Generation

Size of tgt. win.	15-min	1-hour	1-day	1-week
CTR lift	0.1829	0.2266	0.2598	−0.0086
ROC area	0.8031	0.8145	0.8267	0.7858
Act. ex. (10^6)	2,176	1,469	535	158
Run-time (fv-gen)	1.5	1.57	1.43	1.38
Run-time (total)	31.03	27.37	14.07	9.23

Stratified Sampling

- To examine different stratified sampling schemes:

Table: Stratified Sampling

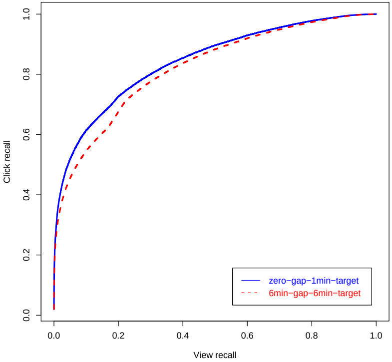
Sampling rates	CTR lift	ROC area	Run-time
neg $= 0$; view $= 1$	0.2598	0.8267	14.07
neg $= 0.2$; view $= 1$	0.2735	0.8243	12.77
neg $= 0.5$; view $= 1$	0.2612	0.8208	13
neg $= 1$; view $= 1$	0.2438	0.8162	11.88
neg $= 0$; view $= 0.5$	0.2579	0.8280	8.9
neg $= 0$; view $= 0.2$	0.2462	0.8266	7.57
neg $= 0$; view $= 0$	-0.0328	0.7736	5.38

Note:

neg examples with zero ad clicks and views;
view examples with view-only events.

Latency

- To validate the potential of latency removal:



Latency	6-min gap 6-min target	no-gap 1-min target
CTR lift	0.2598	0.4295
ROC area	0.8267	0.8413

Contributions

1. A MapReduce statistical learning algorithm and implementation that achieve optimal data parallelism, task parallelism, and load balance in spite of the typically skewed distribution of domain data.
2. An in-place feature vector generation algorithm with linear time complexity $O(n)$ regardless of the granularity of sliding target window.
3. An in-memory caching scheme that significantly reduces the number of disk IOs to make large-scale learning practical.
4. Highly efficient data structures and sparse representations of models and data to enable fast model updates.

We believe that our work makes significant contributions to solving large-scale machine learning problems of industrial relevance in general.

Thanks

Thanks and Comments.