TERA-SCALE MACHINE LEARNING

Achieving Scale and Speed in Computational Advertising

February 12, 2015

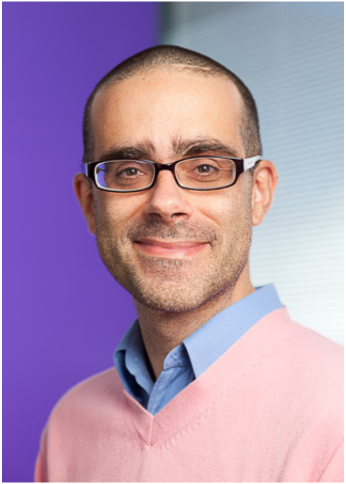
Bruno Fernandez-Ruiz, Senior Fellow
<brunofr@yahoo-inc.com>

Yahoo Inc.

ABOUT THE AUTHOR

Bruno Fernandez-Ruiz

- Senior Fellow at Yahoo, working in computational advertising, recommender systems and search.
- Previously part of Andersen Consulting CStaR in Sophia Antipolis.
- M.Sc.Operations Research and Research Affiliate at MIT.; M.Eng. from Universidad Politécnica de Madrid and Ecole Nationale des Ponts et Chaussées.



DISCLAIMER

- This talk presents the opinions of the author. It does not necessarily reflect the views of Yahoo Inc or any other entity.
- Algorithms, techniques, features, etc. mentioned here might or might not be in use by Yahoo or any other company.
- This lecture benefitted from the valuable contributions of many colleagues at Yahoo and elsewhere.

OVERVIEW

1. Computational Advertising
2. Machine Learning
3. The Big-Data Machine Learning Toolset
4. Design Patterns Enabled
5. Conclusion

COMPUTATIONAL ADVERTISING

Advertising: form of marketing communication used in order to persuade an audience to take an action.

- A publisher creates media.
- An audience reads, views, listens, ... such media.
- The publisher makes *inventory* available, i.e. ad spaces.
- An advertiser buys ad spaces to insert advertising creatives that target the audience.

Advertiser campaign goals vary:

- *Brand Advertising*: create a distinct and favorable brand (and/or product/campaign) image, e.g. Global Recall Points (GRP) ...
- *Direct Marketing*: obtain a direct response from the audience, e.g. conversions, installs, purchases, ...

A GAME OF MULTIPLE OBJECTIVES

- **Publishers**' objective is to optimize short-term revenue (*yield*) and long-term revenue (*user engagement*):
  - Percentage of inventory sold, e.g. sell-through-rate.
  - Revenue per unit of inventory, e.g. revenue per mille, revenue per search.
  - Media reach, e.g. readership, page views, number of searches, minutes online.
- **Advertisers**' objective is to optimize budgets' ROI *and* volume:
  - Maximize GRP lift per campaign.
  - Reduce cost per acquisition.
  - Ensure minimum audience reach.

EVOLUTION OF ONLINE ADVERTISING: THE BEGINNINGS

1. Initially, property owners sold media inventory banner advertising slots in large exclusivity contracts, e.g. Yahoo Sports sells all daily impressions to Nike. To buy sports-enthusiasts, one buys Yahoo Sports, ESPN, etc. So far, same as the printed world.
2. Yahoo Sports grows in readership more than any advertiser can buy; Yahoo slices the inventory and enters into upfront contracts with multiple advertisers, providing volume guarantees to each of them. Premium guaranteed contracts ensure minimum delivery, within a given timeline, and associate penalties of under-delivery.
3. Online media competitors start appearing, and eventually offer access to similar audience. Yahoo starts offering "behavioral targeting" to allow advertisers to better select their audience (a more targeted audience results in higher ROI, more GRP-lift and higher conversion rates).

EVOLUTION OF ONLINE ADVERTISING: THE AD NETWORKS

4. Large media owners like Yahoo create "ad networks" to package and sell an audience across multiple properties.
5. Search and contextual advertising appears and allows direct response advertisers to optimize for clicks and conversions.
6. Advertisers start buying in portfolios across multiple media and ad networks, across display, search and video; they optimize global media spent.
7. Publishers allocate part of their inventory to exclusive direct sales deals and part of the inventory to the ad network(s). All are upfront contracts that carry a pricing premium for the guarantees. Publishers end up with "remnant" inventory.

EVOLUTION OF ONLINE ADVERTISING: PROGRAMMATIC BUYING

8. Ad exchanges appear as a spot market to liquidate remnant inventories. Ad exchanges use Real-Time Bidding to programmatically connect multiple publishers (supply) and advertisers (demand) together: every impressions at a publisher is an auction, with multiple advertisers bidding per impression.
9. Advertisers get an opportunity to peak into the impression and cherry-picking appears. Cookie syncing, data mapping, data cooking allows tracking and building audience profiles. Data Management Platforms appear and provide intelligence to the bidders to buy the right audience, at the right time, at the cheapest possible price.
10. Large media publishers get price eroded.
11. We are currently seeing a move from large, upfront, premium contracts with guarantees based on sales rate cards, to an efficient, real-time, programmatic spot market.

THE TASKS OF COMPUTATIONAL ADVERTISING

- Identify and track users online (and offline) across media.
- Cluster users to optimize for clicks and conversions.
- Forecast supply to minimize under-delivery penalties.
- Select the "best" representation for users and ads.
- Extract query intent, publisher page context, etc.
- Design pricing to motivate advertisers to bid truthfully.
- Select the ads with highest yield.
- Select the ads with highest quality.
- Ensure a "fair distribution" of impressions.
- Dynamically optimize the ad creative.
- Decide when to bid.
- ...

COMPUTATIONAL ADVERTISING IN COMPUTER SCIENCE

New scientific sub-discipline bringing together:

- Microeconomics
- Game theory
- Auction theory
- Mechanism design
- Information retrieval
- Natural language processing
- Large scale systems engineering
- Computer vision
- **Machine learning**
- ...

SO, WHAT IS COMPUTATIONAL ADVERTISING?

Optimization program: find the "best" advertising,

- Given a user, e.g. an online profile associated with a cookie.
- Given a context, e.g. search query, publisher page, video stream, etc.
- Given a corpus of ad offers and contracts, e.g. sponsored search, premium display banners, video pre-rolls, etc.
- Subject to a set of publisher yield constraints.
- Subject to a set of marketplace constraints.

A CONCRETE FORMULATION

For every impression:

1. select the top-N ad offer *candidate slate* (advertiser specific, position independent),
2. ranked for *short-term revenue* ($eCPM$),
3. discounted for *negative externalities* (qs).

$$eCPM_k = bid_k \cdot pCTR_k$$
$$ar_k = eCPM_k \cdot qs_k$$

- $eCPM_k$ effective cost-per-mille (revenue ex-TAC)
- ar_k ad score (rank) for the ad offer in position k in the slate,
- bid_k maximum cost per click advertiser is willing to pay,
- $pCTR_k$ predicted click-through rate,
- qs_k quality score for the ad, capturing future negative externalities, pre- and post-click.

GENERALIZED SECOND PRICE AND CLICK-THROUGH RATES

Given the ranked slate of ads:

$$ar_1 < ar_2 < ... < ar_k < ... < ar_N$$

we price ad offer ar_k at the minimum that advertiser would have to pay to outbid the next position offer with ar_{k+1}:

$$PPC_k \cdot pCTR_k \cdot qs_k = bid_{k+1} \cdot pCTR_{k+1} \cdot qs_{k+1}$$

so the effective price-per-click is:

$$PPC_k = bid_{k+1} \cdot \frac{pCTR_{k+1}}{pCTR_k} \cdot \frac{qs_{k+1}}{qs_k}$$

Accurate Predictions: It’s Down to the Money

Click-Through Rate and *Quality Score* estimation biases have huge impact on marketplace efficiency (operator), yield (publisher), pricing (advertiser) and long-term user retention/satisfaction

FOCUSING ON CLICK PREDICTION

Conditional probability, unknown to us:

$$pCTR = P(click|user, context, ad)$$

Model clicks and relevance scores using attributes from:

- User
  - Geo / location
  - Behavior (views, searches, clicks ...)
  - Techno- and demographic (age, gender, device, network ...)
- Context
  - Page content (words, phrases, category, ...)
  - Meta information (URL, referral query, web rank, ...)
- Ad
  - Creative (title, abstract, URL, ...)
  - Bid terms
  - Categories
  - Targeting geo
  - Bid amount

CLICK MODELING

Feature engineering:

- Unigrams, Phrases, Categories, Geo, Bidterm, keywords
- Weights adjust the contribution of each score to the final score
- User/page and ad are represented by vectors in different spaces
- Score is the cosine distance between vectors in each space
- Final score is linear combination of individual scores

Click modeling score is the estimate P(click | user, context, ad):

- Intensity of word or phrases based on tf-idf.
- Intensity of categories based on categorizer score.
- Editorially judged page-ad pairs, optimize weights.
- Rule weights are learned during training.
- Rules can match features from channels of query and channels of ad.

MACHINE LEARNING

GENERALIZATION ERROR

A training example is a pair (x, y) composed of an input vector of features x and a scalar or label output y. In the binary case of clicks (-1 no click; 1 click):

$$x \rightarrow y \in \{-1, +1\}$$

Given an unseen example, our objective is to estimate the output:

$$x \rightarrow \hat{y}$$

The quality of a learning system is determined by the generalization error (E) [and a loss function (L)]:

$$E_n(f) = \int L(\hat{y}, y) dp(y) = \int L(f_w(x), y) dp(x, y)$$

Our objective is to find the function that minimizes E.

OPTIMIZATION PROBLEM

The solution to the learning problem is the function $f_w(x)$ with a weight vector such that:

$$\hat{w} = \arg\min_{w} \left(\sum_{i=1}^{n} L(f_w(x_i), y_i) + \lambda R(w) \right)$$

where we introduce R and λ for regularization (control to avoid overfitting the parameters).

Much machine learning work focuses on problems of this form (e.g. Adaline, Perceptron, K-Means, SVM, Lasso, ...), and it's also applicable to other problems such as large-scale matrix factorization (LDA, LFA, random indexing, etc.), collaborative filtering, deep networks, etc.

EXAMPLE OPTIMIZATION PROBLEMS

For example, an Adaline learns by selecting a family of linear functions, and minimizing the mean square errors:

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

In the case of ad click prediction, it's common to estimate the click probability by maximizing the entropy (logistic regression). We assume the regression is a sigmoid (logistic) function, and we use L2 regularization. The weight vector is in this case:

$$\hat{w} = \arg\min_{w} \left(\sum_{i=1}^{n} log(1 + exp(-y_i w^T x_i)) + \frac{\lambda}{2} \|w\|^2 \right)$$

BATCH LEARNING METHODS: GRADIENT DESCENT

GD is an iterative batch method that updates in each step the weight vector w_k in the direction of the gradient of $E_n(f_w)$ by a small amount ϵ_k (learning step):

$$w_{k+1} = w_k - \epsilon_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w L(x_i, y_i, w_k)$$

- GD is slow but accurate to converge.
- Calculating the gradient is computationally burdening since we need to estimate the gradient over the entire training set at every step of the algorithm until we reach convergence.
- Industrially, we use L-BFGS (limited memory BFGS), a method similar to GD.

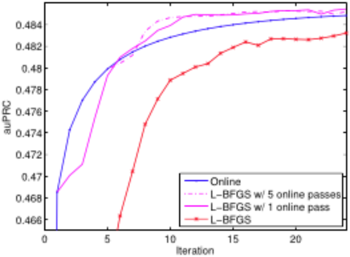
ONLINE LEARNING METHODS: STOCHASTIC GRADIENT DESCENT

The SGD algorithm estimates the gradient on the basis of a randomly picked example:

$$w_{k+1} = w_k - \epsilon_k \nabla_w L(x_k, y_k, w_k)$$

- SGD converges faster than GD and can escape local minimum and works for infinite training sets (e.g. datastreams).
- SGD can be easily parallelized by splitting the weight vectors into different CPU cores.
- Given the sequential nature of the SGD algorithm, the ability to scale is bound by the maximum number of cores and available memory a single computer.

HYBRID TRAINING METHODS



Agarwal et Al., A Reliable Effective Terascale Linear Learning System

Effect of initializing the L-BFGS optimization by an average solution from online runs on individual nodes. Test auPRC for 4 different learning strategies. Note that the online and hybrid curves overlap during the warmstart phase (of either 1 or 5 online passes).

THE SCALING CHALLENGES

Scale:

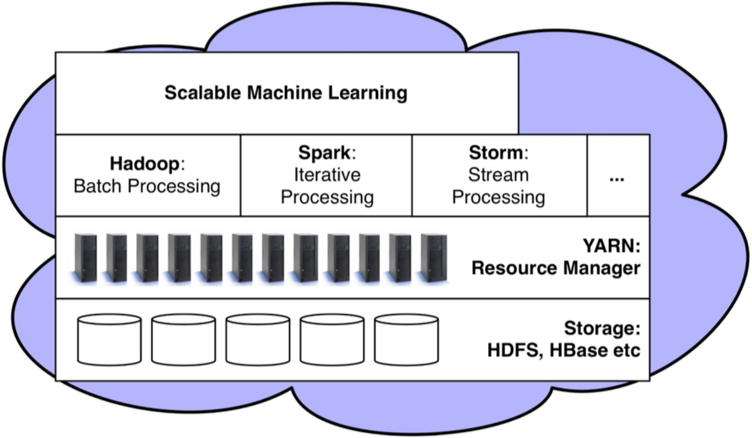
- 1,000,000,000 examples
- 100,000,000 features
- 10,000 models
- 10 algorithms

Speed:

- Naïve solutions spend days/hours in model training
- Items discovery within minutes, e.g. breaking news
- Temporal nature of user interests, e.g. query intent

THE BIG-DATA MACHINE LEARNING TOOLSET

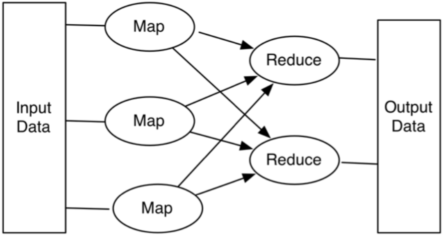
BIG-DATA MACHINE LEARNING



APACHE HADOOP

http://hadoop.apache.org

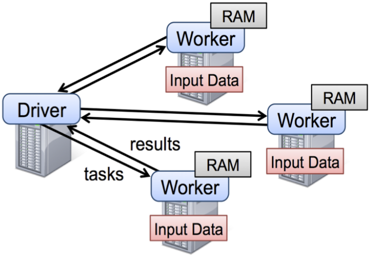
- Popular framework for running applications on large cluster built of commodity hardware
- Designed for very high throughput and reliability
- YARN resource manager supports Map/Reduce, Tez and beyond



APACHE SPARK

http://spark.apache.org

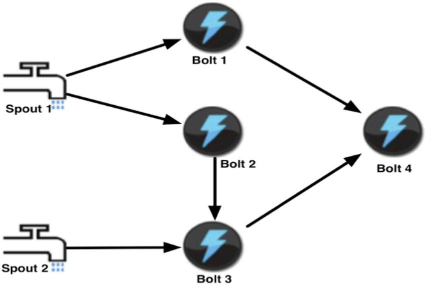
- Fast and expressive cluster computing system compatible with Apache Hadoop
- Support general execution DAGs; Include iterative programming
- Resilient distributed datasets (RDDs)
- In-memory storage



APACHE STORM

http://storm.apache.org

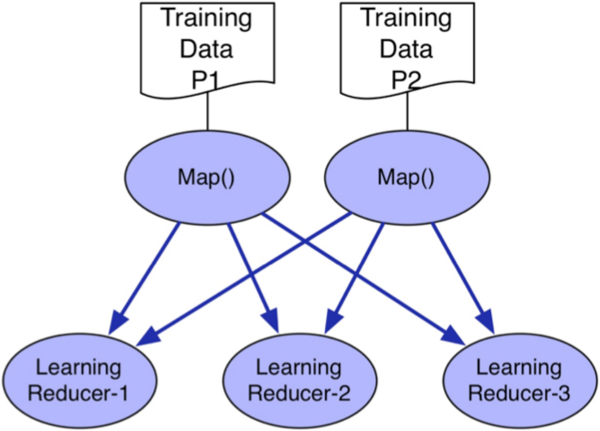
- Hadoop for Realtime
- Distributed, fault-tolerant, and high-performance streaming computation
- Top-level Apache project since Sept. 2014



DESIGN PATTERNS ENABLED

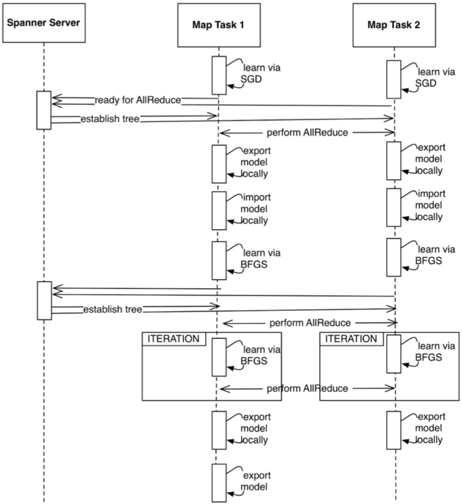
BATCH TRAINING: 1,000+ MODELS

Hadoop: Training data for each model could be loaded into a single machine.



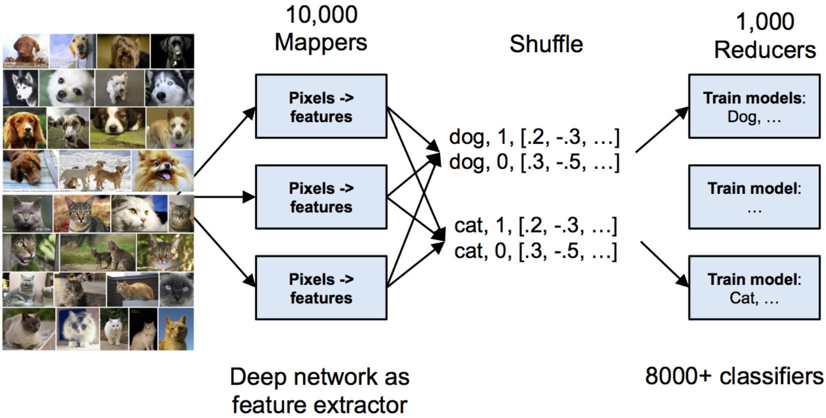
BATCH TRAINING: LARGE TRAINING DATASET

Hadoop + MPI AllReduce: Training data are too large to be loaded into a single machine.



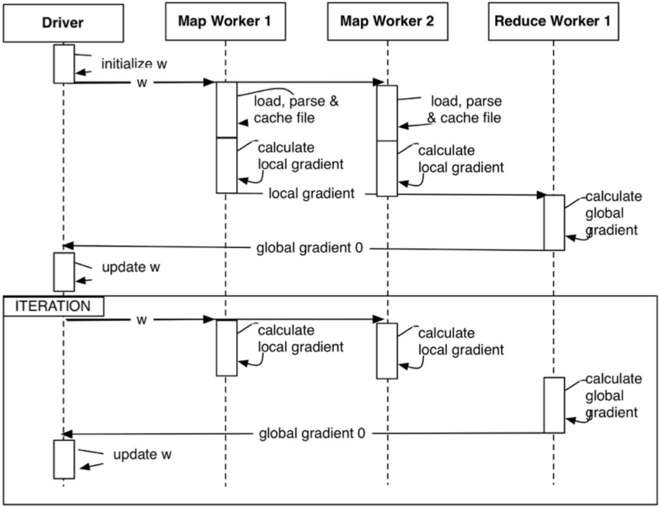
BATCH TRAINING: LARGE DATASET AND 1000+ MODELS

Example: ad landing page image classification.



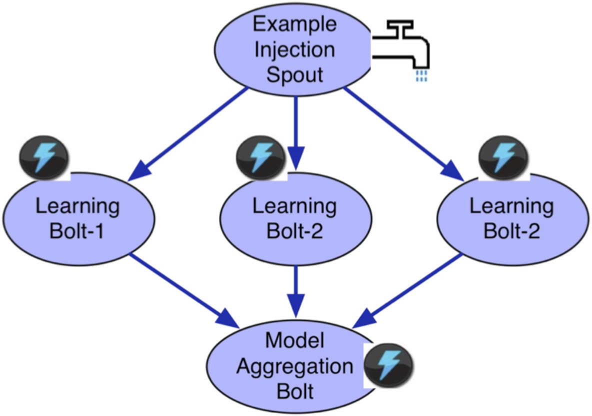
BATCH TRAINING: MEDIUM BUT DISTRIBUTED DATASET

Spark: behavioral targeting segments.



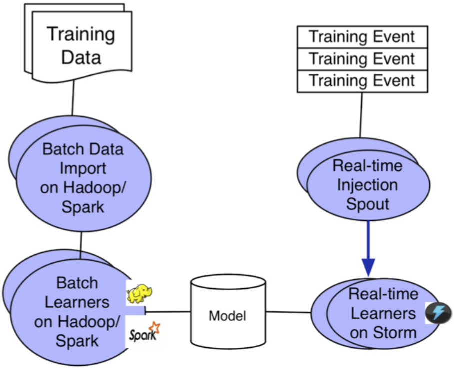
ONLINE TRAINING: SUB-MINUTE MODEL FRESHNESS

Storm: search query intent.

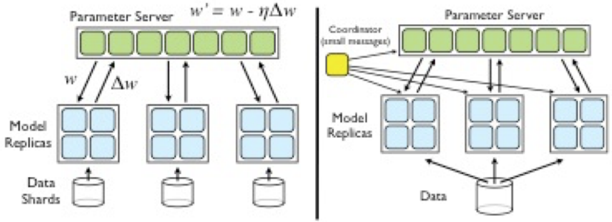


HYBRID TRAINING: RE-TRAIN AT SPEED

- Bootstrap models via batch learning from large datasets, update models via realtime learning from latest events
- Bootstrap learning online, switch to batch for accuracy
- ML in Hadoop + Storm
- ML in Spark + Storm



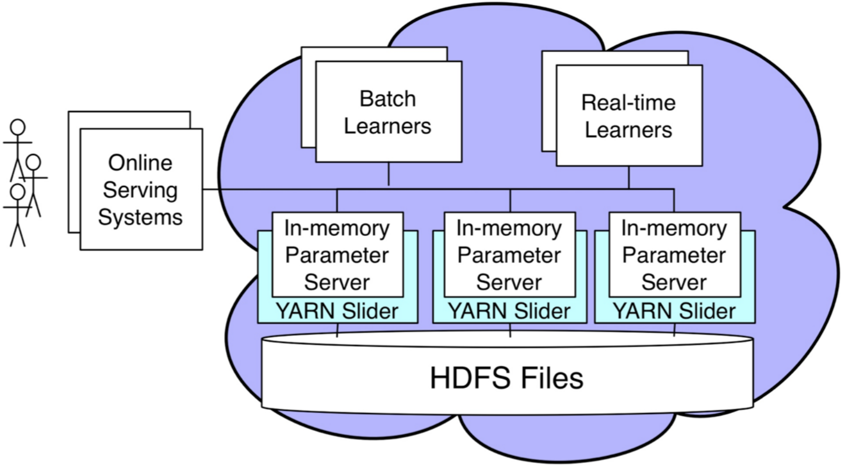
HYBRID TRAINING: LARGE SCALE AT SPEED



Dean et Al., Large Scale Distributed Deep Networks

- billions of features per model; millions of operation per second
- asynchronous gradient descent: no consistency or order guarantees
- practically, it works for non-linear, non-convex, global minima

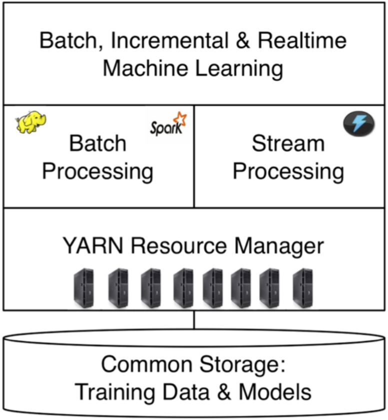
HYBRID TRAINING: PARAMETER SERVER ON HADOOP



CONCLUSION

CONCLUSION

- Scalable machine learning is critical for big-data tech evolution.
- Computational advertising (and search) is pushing the boundaries of machine learning.
- It's possible to achieve large scale machine learning with an open source strategy; Yahoo committers:
  - Apache Hadoop 15
  - Apache Storm 5
  - Apache Spark 4



REFERENCES: COMPUTATIONAL ADVERTISING

- Hal R. Varian
  Position auctions
  International Journal of Industrial Organization, 2006
- Hal R. Varian
  Online ad auctions
  The American Economic Review, 2009
- Olivier Chapelle, et al.
  Simple and scalable response prediction for display advertising
  Transactions on Intelligent Systems and Technology, 2014
- H. Brendan McMahan, et al.
  Ad Click Prediction: a View from the Trenches
  Proceedings of the 19th ACM international conference on Knowledge Discovery and Data mining, 2013

REFERENCES: MACHINE LEARNING

- Yoshua Bengio
  Deep learning of representations: looking forward
  Proceedings of the First international conference on Statistical Language and Speech Processing, 2013
- Jeffrey Dean, et al.
  Large Scale Distributed Deep Networks
  Advances in Neural Information Processing Systems, 2012
- Mu Li, et al.
  Scaling distributed machine learning with the parameter server
  Operating Systems Design and Implementation (OSDI), 2014.
- Alekh Agarwal, et al.
  A reliable effective terascale linear learning system
  The Journal of Machine Learning Research, January 2014.

ABOUT

Thoughts, suggestions, feedback, questions, comments:

- `brunofr@yahoo-inc.com`