Click modeling for display advertising

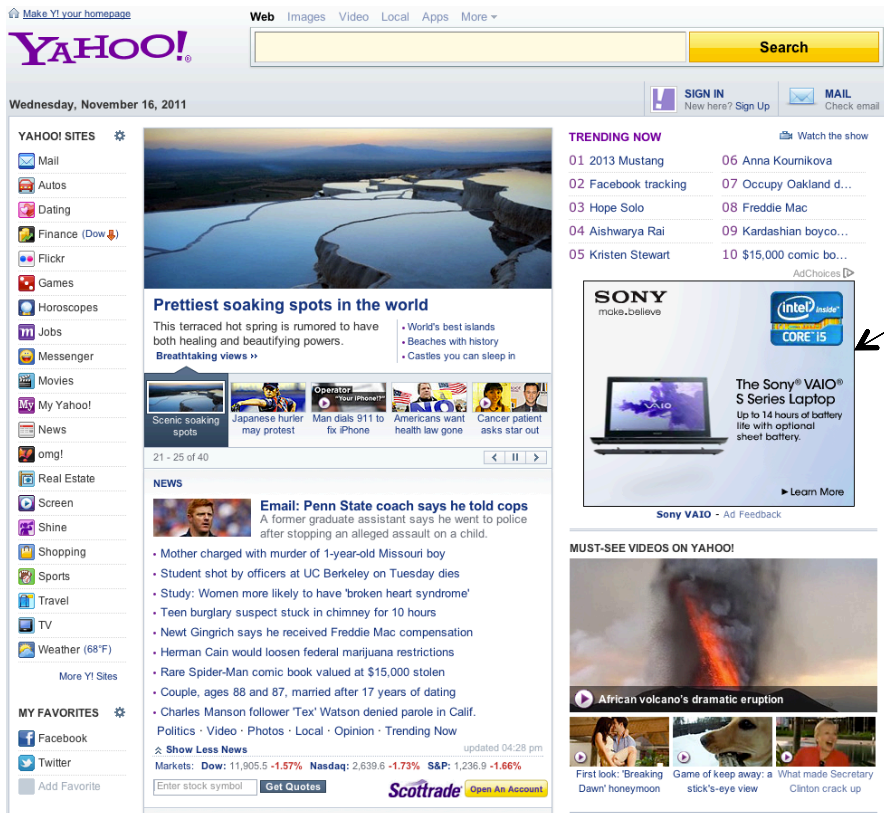
June 30, 2012

Olivier Chapelle

Outline

1. Display advertising
2. Modeling
3. Large scale learning
4. Explore / exploit

Display advertising



Display ad

Display advertising

- Rapidly growing multi-billion dollar business (38% of internet advertising revenue in 2010).
- Marketplace between:
  - Publishers: sell display opportunities
  - Advertisers: pay for showing their ad
- Two different modes:
  - Guaranteed delivery: opportunities are defined and sold in advance.
  - Non guaranteed delivery (this talk): real-time auction amongst advertisers is held at the moment when a user generates a display opportunity by visiting a publisher's web page.

Pricing type

- CPM (Cost Per Mille): advertiser pays per thousand impressions
- CPC (Cost Per Click): advertiser pays only when the user clicks
- CPA (Cost Per Action): advertiser pays only when the user performs a predefined action such as a purchase.

Generalized second price auction

$$\text{Winner}: \quad \arg\max_i \quad b_i p_i$$

$$b_i = \text{bid}$$

$$p_i = \text{probability (click or action)}$$

$$\text{Price charged:} \quad \frac{b_2 p_2}{p_1}$$

$$\text{with } b_1 p_1 > b_2 p_2 > \ldots$$

Click prediction

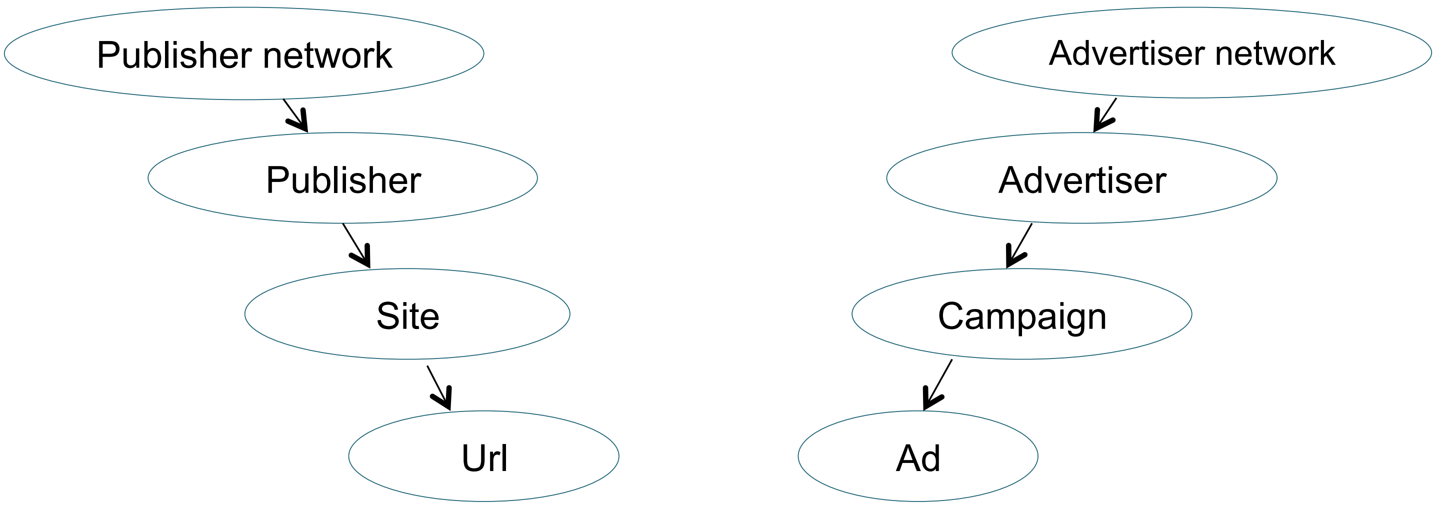
- Click prediction is a critical aspect of display advertising.
- Highly unbalanced problems (clickthrough rates < 1%).
- Related problem: action prediction for CPA (even more unbalanced).
- Very large amount of data: about 9B daily impressions.

Outline

1. Display advertising
2. Modeling
3. Large scale learning
4. Explore / exploit

Features

- Three sources of features: user, ad, page
- In this talk: categorical features on ad and page.
- Two hierarchies of features:



Hashing trick

- Standard representation of categorical features: “one-hot” encoding

Site: (0 … 0 1 0 … 0)

cnn.com news.yahoo.com

- Dimensionality equal to the number of different values: can be very large
- Hashing to reduce dimensionality (made popular by John Langford in VW)

$$h : \text{string} \rightarrow [0 \ldots 2^b - 1]$$

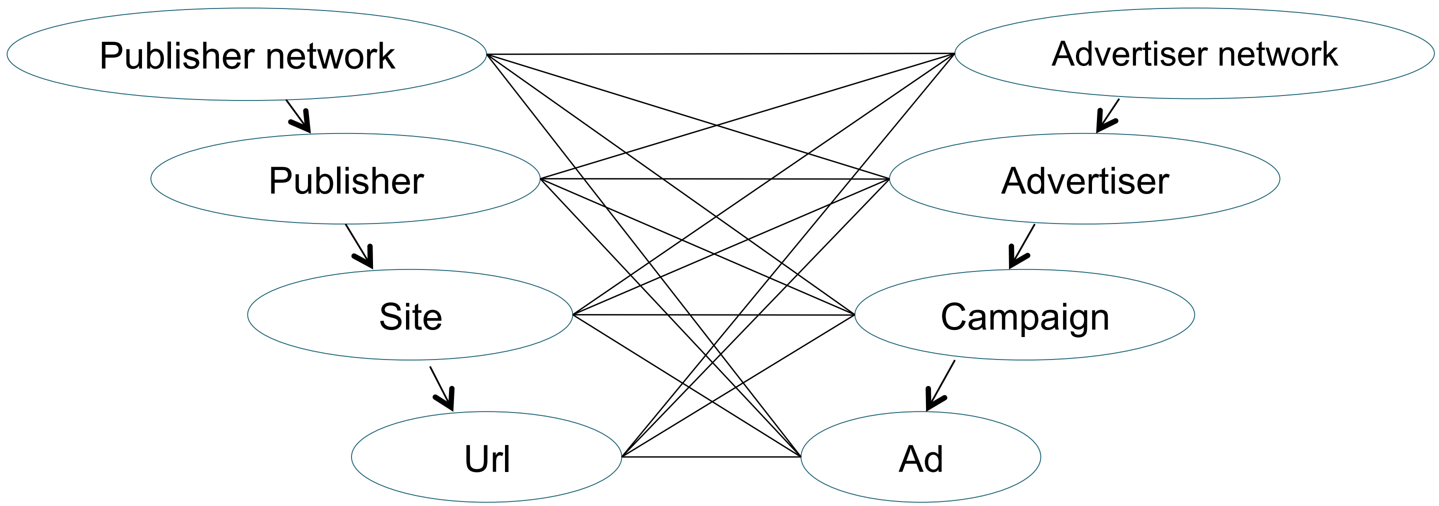
$$x_{h(v)} = 1$$

- Still one-hot encoding but dimensionality independent of number of values
- All features are hashed into the same space
- Risk of collision
- Collisions are form of regularization: infrequent feature values are washed away by the frequent ones.

Quadratic features

- Outer product between two features.
- Example: between *site* and *advertiser,*

  Feature = 1 $\Leftrightarrow$ site=finance.yahoo.com & advertiser=bank of america



- Similar to a polynomial kernel of degree 2
- Large number of values $\longrightarrow$ *hashing trick*

Advantages of hashing

- Statistical
  - Regularization
- Practical
  - Straightforward implement; no need to maintain dictionaries
- Most powerful when combined with quadratic features

Quote of John Langford about hashing:

At first it’s scary, then you love it

Learning

- Regularized logistic regression
  - Vowpal Wabbit open source package

- Regularization with hierarchical features $\approx$ backoff smoothing

$$w_{parent} \quad + \quad w_{child}$$

Well estimated

Small if rare value

- Negative data subsampled:
  - Statistical advantage (better balance)
  - Computational advantage

Evaluation

- Comparison with (Agarwal et al. ’10)
  - Probabilistic model for the same display advertising prediction problem
  - Leverages the hierarchical structures on the ad and publisher sides
  - Sparse prior for smoothing
- Model trained on three weeks of data, tested on the 3 following days
- Metrics: area under the ROC and PR curves, log likelihood

auROC	auPRC	Log likelihood
+ 3.1%	+ 10.0%	+ 7.1%

D. Agarwal et al., Estimating Rates of Rare Events with Multiple Hierarchies through Scalable Log-linear Models, KDD, 2010

Bayesian logistic regression

- Regularized logistic regression = MAP solution (Gaussian prior, logistic likelihood)
- Posterior is not Gaussian
- Diagonal Laplace approximation:

$$\Pr(w \mid D) \approx \mathcal{N}(\mu, \Sigma)$$

with:

$$\mu = \arg\min L(w)$$
$$\Sigma_{ii} = \frac{\partial^2 L}{\partial w_i^2}$$

and:

$$L(w) = -\log \Pr(w \mid D) = \sum_{j=1}^{n} \log(1 + \exp(-y_j w \cdot x_j)) + \lambda \|w\|^2$$

Model update

- Needed because ads / campaigns keep changing.
- The posterior distribution of a previously trained model can be used as the prior for training a new model with a new batch of data.

Require: Regularization parameter $\lambda > 0$.
$m_i = 0,\ q_i = \lambda.$ {Each weight w_i has an independent prior $\mathcal{N}(m_i, q_i^{-1})$}
for $t = 1, \ldots, T$ **do**
Get a new batch of training data $(\mathbf{x}_j, y_j),\quad j = 1, \ldots, n.$
Find $\mathbf{w}$ as the minimizer of: $\displaystyle \frac{1}{2}\sum_{i=1}^{d} q_i (w_i - m_i)^2 + \sum_{j=1}^{n} \log(1 + \exp(-y_j \mathbf{w}^\top \mathbf{x}_j)).$
$m_i = w_i$
$\displaystyle q_i = q_i + \sum_{j=1}^{n} x_{ij}^2 p_j (1 - p_j),\quad p_j = (1 + \exp(-\mathbf{w}^\top \mathbf{x}_j))^{-1}$ {Laplace approximation}
end for

- Influence of the update frequency (auPRC)

1 day	6 hours	2 hours
+3.7%	+5.1%	+5.8%

Outline

1. Display advertising
2. Modeling
3. Large scale learning
4. Explore / exploit

Parallel learning

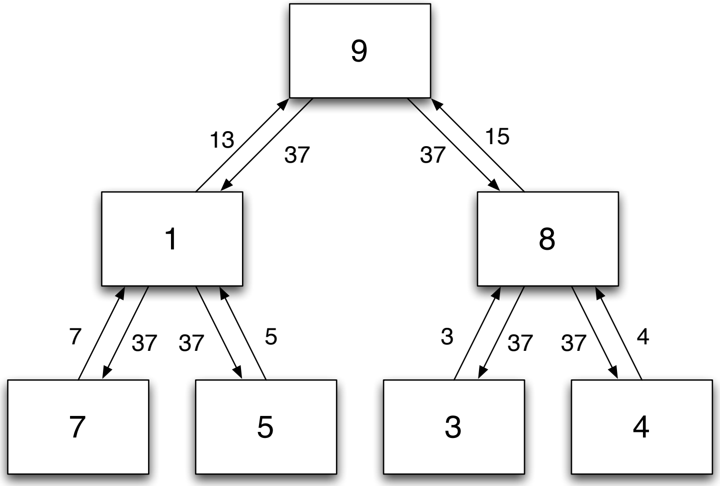
- 2B training samples
- 16M parameters
- Training set size = 400GB (compressed)
- Less than one hour with 500 machines

- Optimize: $$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \log(1 + \exp(-y_i w \cdot x_i)) + \lambda \|w\|^2$$

- Stochastic gradient descent (SGD) is fast on a single machine, but difficult to parallelize
- Batch (quasi-Newton) methods are straightforward to parallelize
  - L-BFGS with distributed gradient computation.

AllReduce

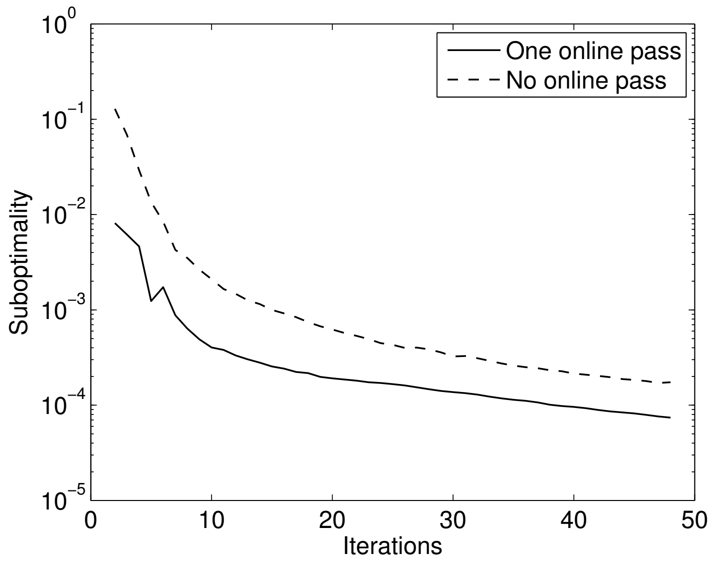
- Aggregate and broadcast across nodes



- Very little modification to existing code: just insert several AllReduce operations
- Compatible with Hadoop / MapReduce
  - Build a spanning tree on the gateway
  - Single MapReduce job
  - Leverage speculative execution to alleviate the slow node issue

Online initialization

- Hybrid approach:
  - One pass of online learning on each node
  - Average the weights from each node to get a warm start for batch optimization
- Best of both (online / batch) worlds.



Robustness and Scaling

- Slowest node is the bottleneck
- Speculative execution: when a node appears to be slow, start a duplicate job

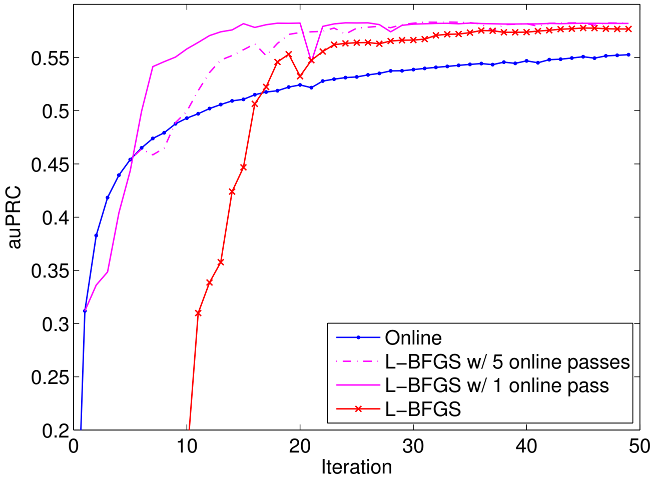
Distribution of computing time over 1000 nodes, with and without speculative execution

	5%	50%	95%	Max	Comm. time
Without	29	34	60	758	26
With	29	33	49	63	10

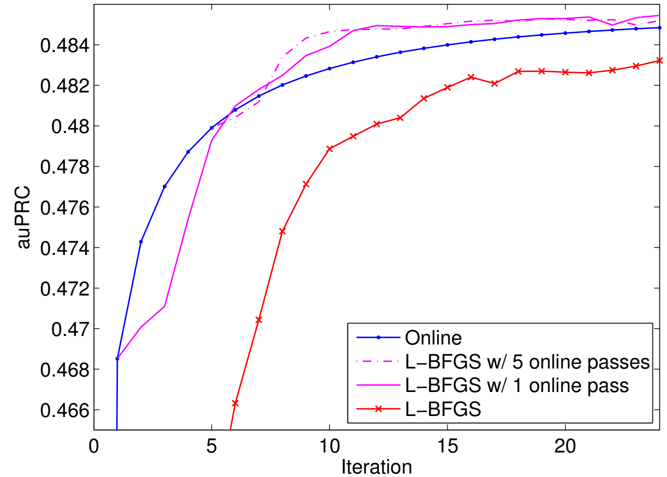
- Can scale up to 1000 nodes

Nodes	100	200	500	1000
Comm time / pass	5	12	9	16
Median comp time / pass	167	105	43	34
Max comp time / pass	462	271	172	95
Wall clock time	3677	2120	938	813

Test accuracy



Splice site prediction (Sonnenburg et al. '10)

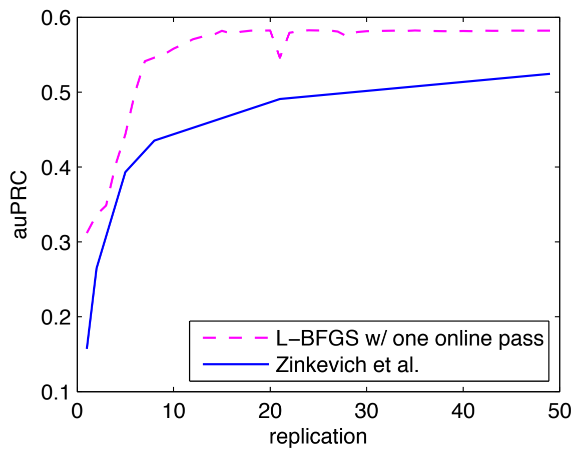


Display advertising

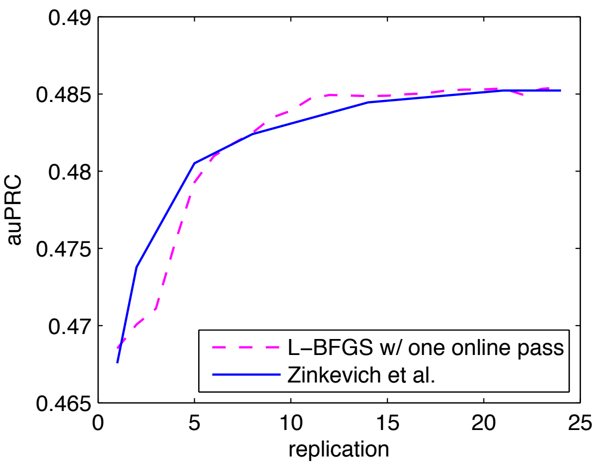
One online pass followed by L-BFGS is the best combination

S. Sonnenburg and V. Franc, COFFIN: A Computational Framework for Linear SVMs, ICML 2010

- Comparison with (Zinkevich et al. ‘10):
  - Replicate the data, do one pass of online learning on each node, average the solution
  - Potential drawbacks:
    - Single pass over the data might hurt accuracy
    - More data to communicate over the network



Splice site prediction



Display advertising

M. Zinkevich et al., Parallelized stochastic gradient descent, NIPS 2010

Outline

1. Display advertising
2. Modeling
3. Large scale learning
4. Explore / exploit

Thompson sampling

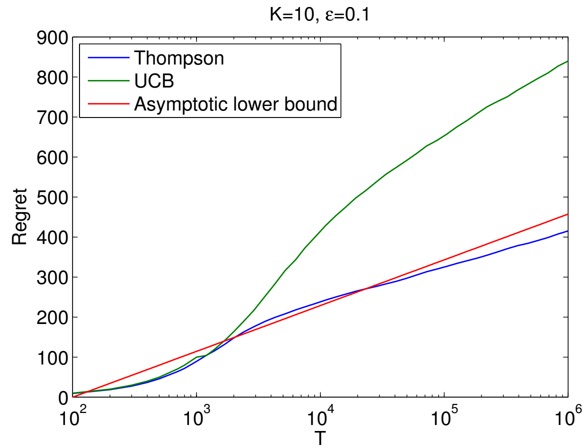
- Heuristic to address the Explore / Exploit problem, dating back to Thompson (1933)
- Simple to implement
- Good performance in practice (Graepel et al. '10, Chapelle and Li '11)
- Rarely used, maybe because of lack of theoretical guarantee.

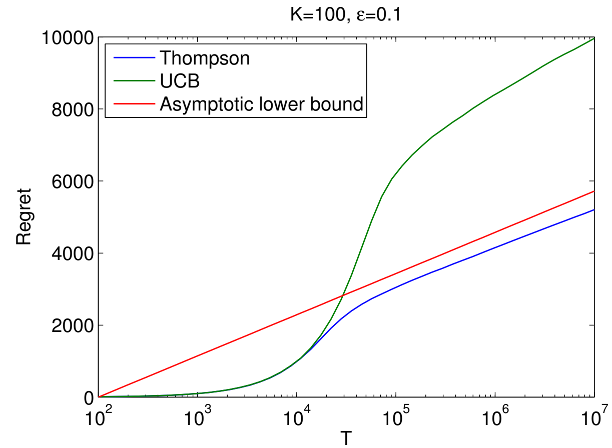
$D = \emptyset$
for $t = 1, \ldots, T$ **do**
  Receive context x_t
  Draw θ^t according to $P(\theta|D)$
  Select $a_t = \arg\max_a \mathbb{E}_r(r|x_t, a, \theta^t)$
  Observe reward r_t
  $D = D \cup (x_t, a_t, r_t)$
end for

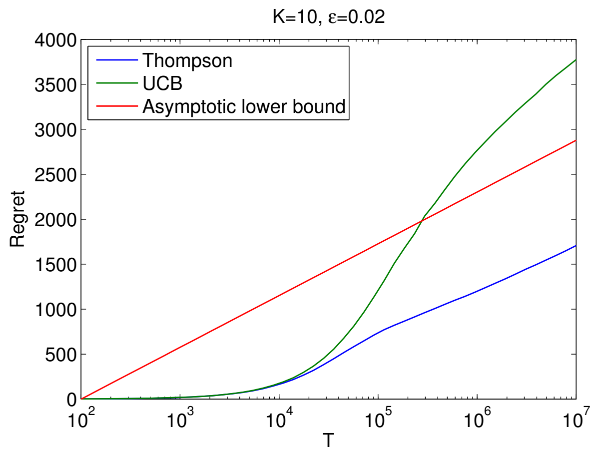
T. Graepel et al., Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine, ICML 2010
O. Chapelle and L. Li, An Empirical Evaluation of Thompson Sampling, NIPS 2011

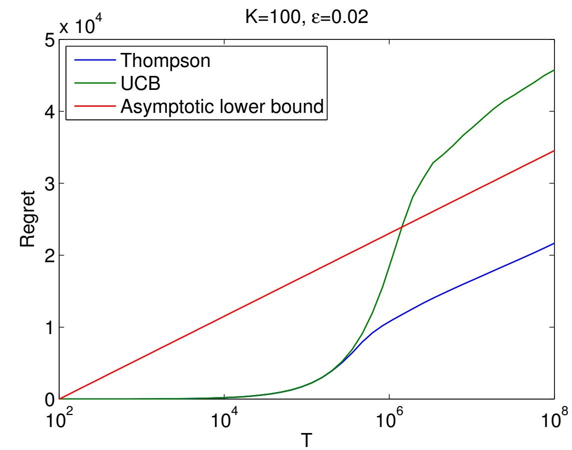
E/E simulations

- MAB with K arms
- Best arm has mean reward = 0.5, others have 0.5 − ε.



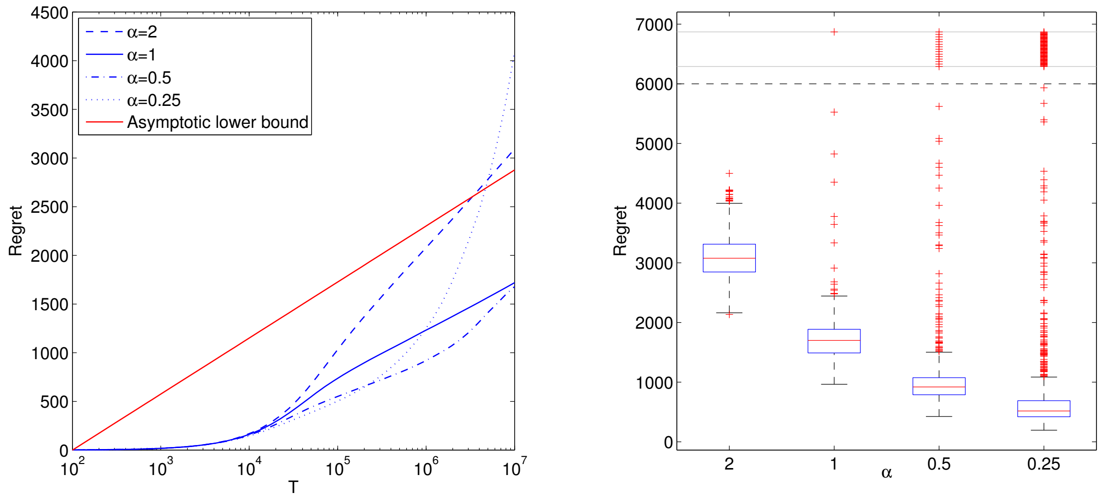






Posterior reshaping

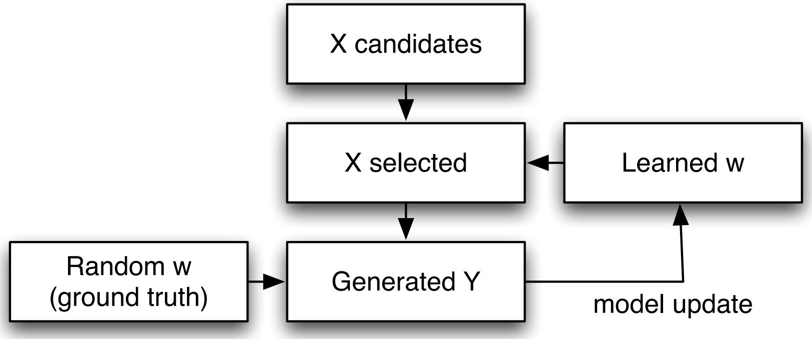
- Add a knob to control the exploration / exploitation trade-off.
- Multiply the variance by α
  - $\alpha > 1$: wider posterior $\longrightarrow$ more exploration
  - $\alpha < 1$: narrower posterior $\longrightarrow$ more exploitation



- $\alpha < 1$ tends to achieve smaller regrets, but is riskier in the long run.

Evaluation

- 13,000 hourly opportunities
- Set of eligible ads varies from 1 to 5,910 (mean 1,364). Total ads = 66,373
- Semi-simulated environment: real input features, but labels generated.
- Comparison of E/E algorithms:
  - 4 days of data
  - Cold start
- Algorithms:
  - UCB: mean + α std. dev.
  - ε-greedy
  - Thompson sampling

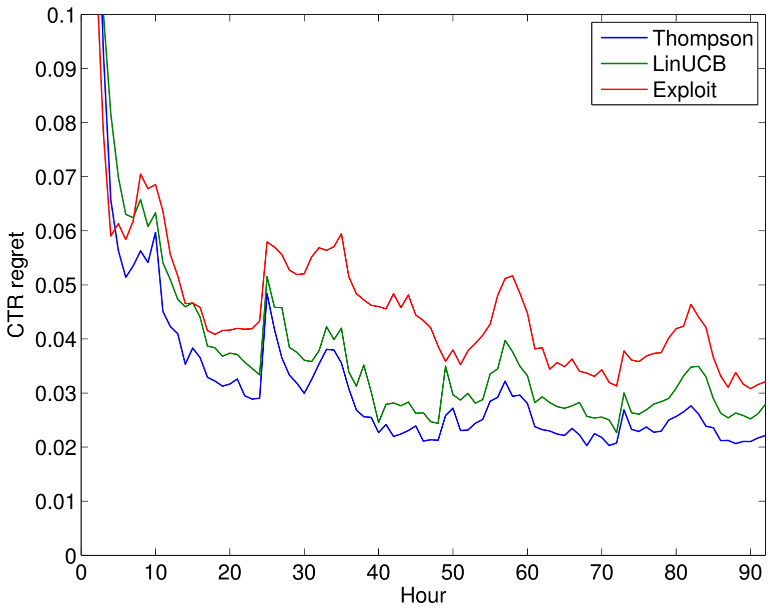


- CTR regret (in percentage):

Thompson (0.5)	UCB (2)	ε-greedy (0.005)	Exploit-only	Random
3.72	4.14	4.98	5.00	31.95

Best parameter value in parenthesis

- Regret over time:



Open questions

- Hashing
  - Theoretical performance guarantees
- Sample selection bias
  - System is trained only on selected ads, but all ads are scored.
  - Possible solution: inverse propensity scoring
  - But we still need to bias the training data toward good ads.
- Explore / exploit
  - Evaluation framework
  - Regret analysis of Thompson's sampling
  - E/E with a budget; with multiple slots
  - Delayed feedback → automatic throttling

Conclusion

- Simple yet efficient techniques for click prediction

- Main difficulty in applied machine learning: avoid the bias (because of academic papers) toward complex systems
  - It's *easy* to get lured into building a *complex* system
  - It's *difficult* to keep it *simple*