Recommender Systems: The Art and Science of Matching Items to Users

Deepak Agarwal

Rutgers University, 4rd May, 2011

Collaborators

- Bee-Chung Chen (Yahoo! Research)
- Raghu Ramakrishnan (Yahoo! Research)

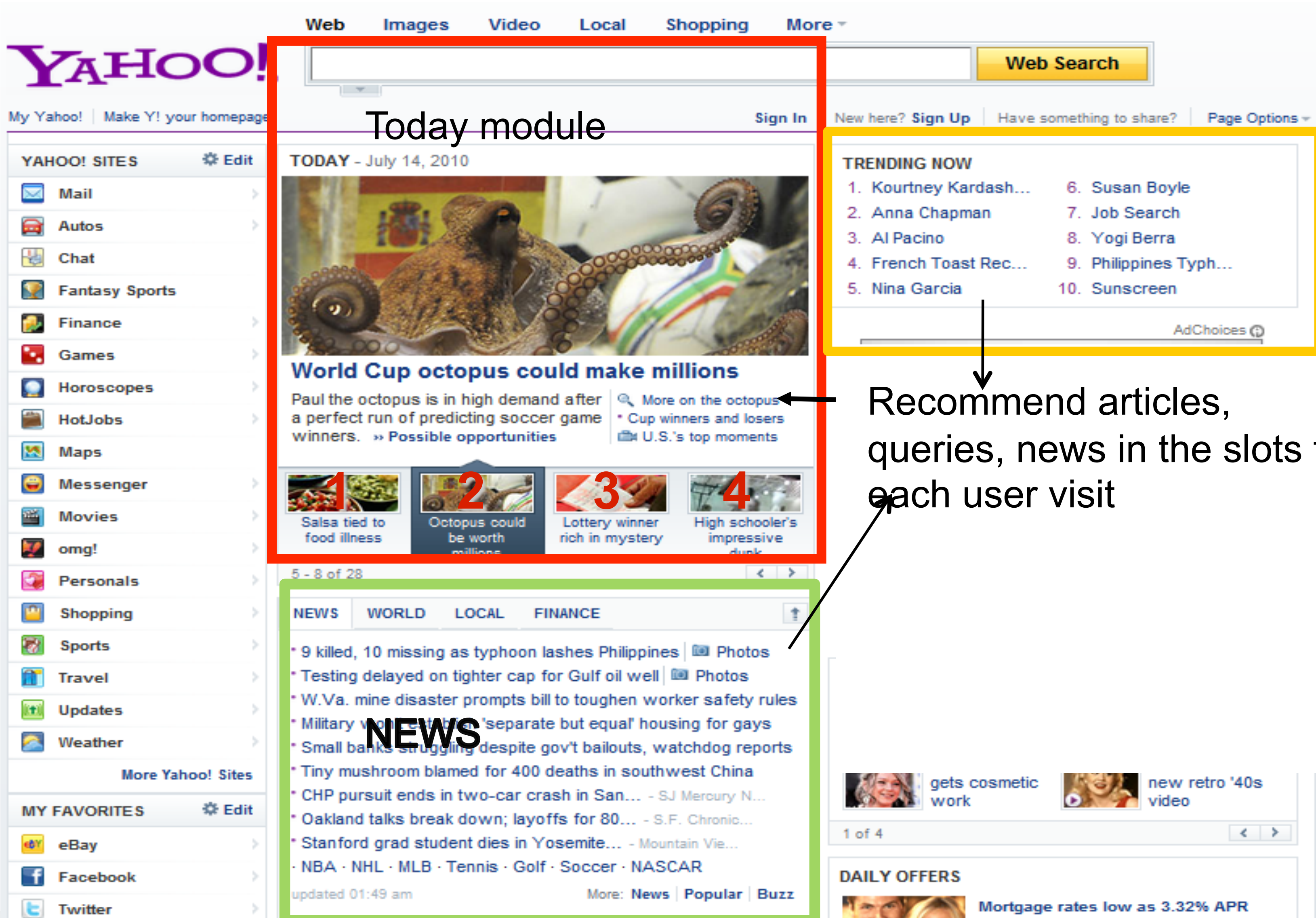
- Pradheep Elango (Yahoo! Labs)
- Liang Zhang (Yahoo! Labs)

- Many others in Engineering, Product and Labs have collaborated to deploy several research methods at various locations on Yahoo! websites around the globe.

Roadmap

- Introduction
  - Motivating example: Yahoo! front page (www.yahoo.com)
- Explore/Exploit problem
- Modeling
  - Covariate based personalization
    - Reduced Rank Regression
  - Personalization at user granularity
    - Regression based latent factor models
- Open Problems

Motivating application: Yahoo! front page



Recommend articles, queries, news in the slots for each user visit

General Approach

- Show content that maximizes user engagement
  - If users are happy, they will spend more time on Yahoo!, they will come back more often and revenue will follow…
- Some proxies for measuring user engagement
  - Click-rate (CTR): Most widely used (despite imperfections)
  - Time spent on landing page (important but not well studied)
  - Session behavior after a click (difficult to model, rarely used)
- In this talk, we will focus on CTR maximization problem

Problem definition

Today Module
4 slots

K1

Trending
10 slots

K3

NEWS
8 slots

K2

Goal: Display content

Maximize CTR in long time-horizon

User covariate vector $\mathbf{x}_{it}$ (includes declared and inferred)

(Age=old, Finance=T, Sports=F)

Difficult problem

- Too many choices even for moderate value of K's
  - Brute force experimentation inefficient, desirable to have adaptive experimentation that rapidly converges to the best choice
- Personalization
  - One size fits all does not work well, personalization based on user covariates desirable.
  - Important to learn at item level, leads to data sparseness
- Dynamic
  - Urn pool changes, user covariates evolve, CTRs change over time

How do we solve the multi-module, multi-slot?

- Reweight observations from minor positions in a module using global weights (estimated through randomized data)
  - E.g. Position 2 twice worse compared to position 1
  - Choose top-m based on scores at position 1

- Assuming CTRs across modules independent, solve each module independently
  - This is not too bad for front page but is clearly not the best, we could do better (work in progress)

Single module, single slot

- Display articles on Today Module for every user visit to
  - Maximize total clicks subject to constraints
    - (Voice, freshness, diversity)

- Inventory of articles?
  - Created by human editors
  - Small pool (30-50 articles) but refreshes periodically
    - Article lifetime short (6-24 hours)
- In this talk, for ease of exposition, assume content recommendation on a single slot
  - (the one with maximum exposure)

Where are we today?

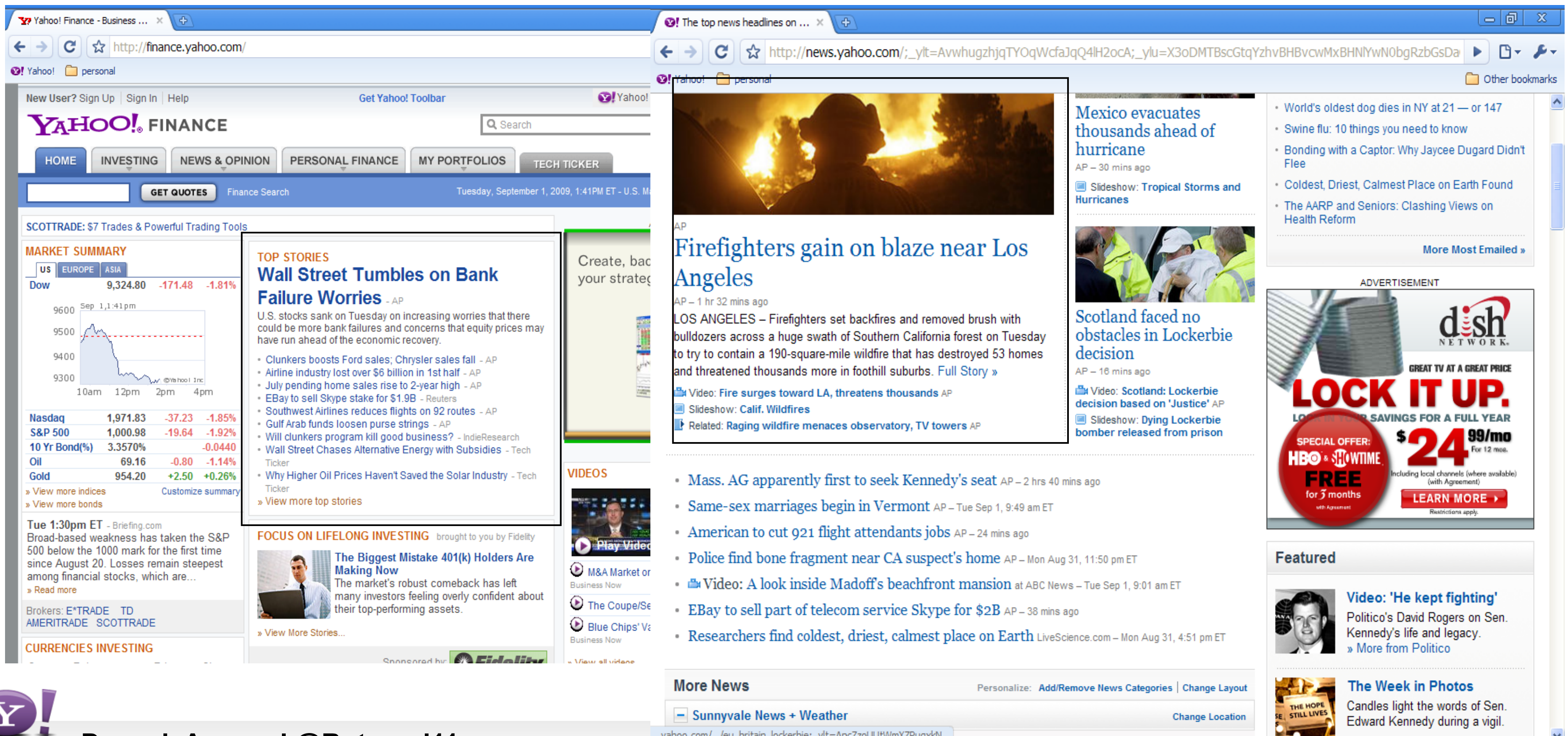
- Before this research
  - Articles *created* and *selected* for display by editors
- After this research
  - Article selection done through statistical models
- Methods
  - Explore/exploit with elaborate statistical models

- How successful ? (significant increase in clicks)

"Just look at our homepage, for example. Since we began pairing our content optimization technology with editorial expertise, we've seen click-through rates in the Today module more than double. ----- Carol Bartz, CEO Yahoo! Inc (Q4, 2009)

Examples of applications similar to front page

- Good news! Methods generalize, already deployed at various locations on Y!



This is an Explore/Exploit Problem

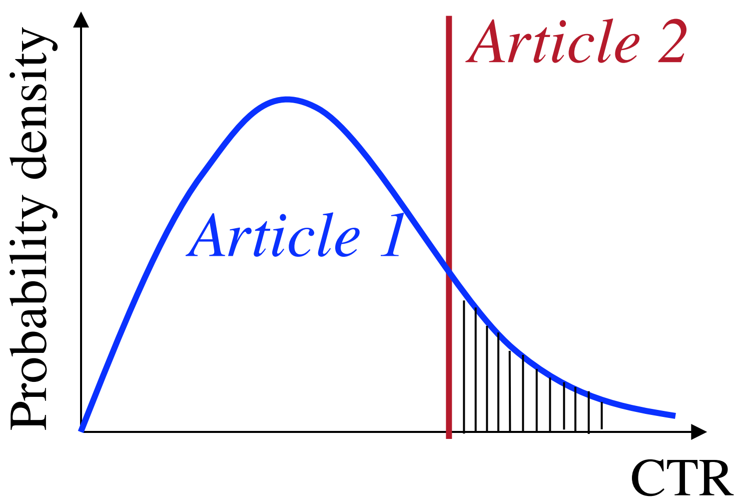
Explore/Exploit high level idea

- *Two Items*: Item 1 CTR= 2/100 ; Item 2 CTR= 25/1000
  - *Greedy:* Show Item 2 to all; not a good idea
  - Item 1 CTR estimate noisy; item could be potentially better
    - Invest in Item 1 for better overall performance on average
  - Show both Item 1 and Item 2
    - Optimal choice of design is the Explore/Exploit problem

- Classical solutions: Multi-armed bandit
  - Gittins' approach (maximize discounted cumulative reward)
  - Upper confidence bound schemes (minimize regret from best)

Bandit problem
(Robbins, Gittins, Whittle, Lai, Berry, Auer, ….)

- There is positive utility in showing articles that currently have low mean but high variance
- E.g. Consider 2 articles
  - Goal: Select most popular
  - CTR_1 ~ (mean=.01,var=.1), CTR_2~ (mean=.05,var~0)



If we only take a **single** decision,
give 100% visits to *Article 2*

If we take **multiple** decisions in the future,
explore *Article 1* since true CTR_1
may be larger.

Bandit Problem: quick tutorial

- Consider a slot machine with two arms

$p_1 > p_2$ (unknown payoff probabilities)

The gambler has 1000 plays, what is the best way to experiment ?
To maximize total expected reward

- Solution to this innocuous looking problem notoriously difficult.

- Gittins' provided a principled solution under some assumptions
- Lai later provided what are called Upper confidence bound policies (UCB)

Optimal bandit solutions

- At any given time in the game, compute priority for each arm *independently* and play the arm with *max priority*
  - Priority essentially represents the future potential of an arm given all the uncertainty about it now
- Upper confidence bound policy (UCB)
  - Mean + uncertainty-estimate
    - mean + k*sd(estimator)
- Gittins'/Whittle kind of policies
  - Priority is expected arm reward based on s-step *future lookahead*
    - *(computation is hard)*
- Thompson
  - randomization by drawing CTRs from the posterior
  - Simple when working in a Bayesian framework

Content optimization: bandit problem

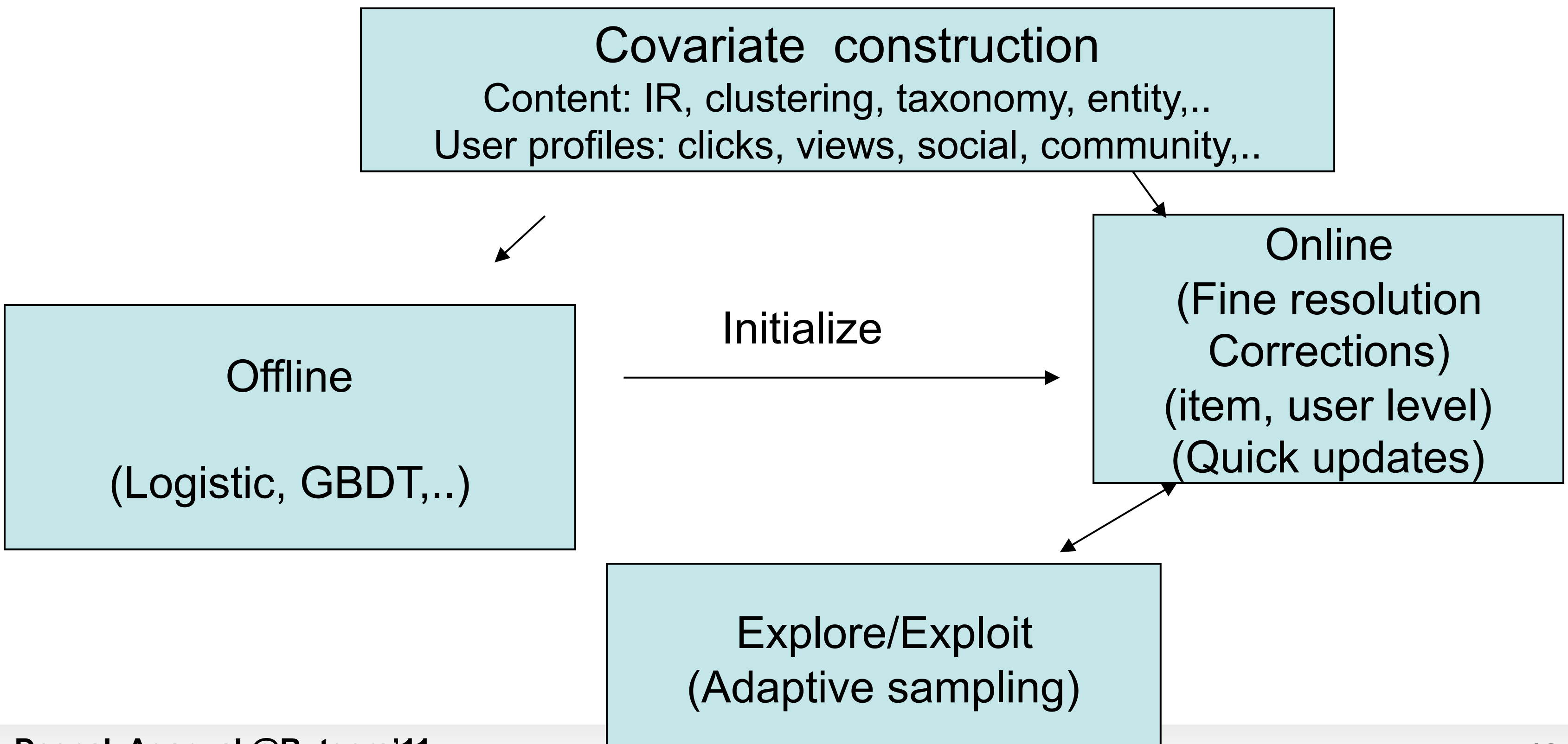
- Articles are arms of bandits, clicks are rewards , CTRs are unknown payoffs
  - Goal is to converge to the best CTR article quickly
  - But this assumes user visits are *iid, no personalization*
- Personalization
  - Each user is a separate bandit
  - Hundreds of millions of bandits (huge casino, multi-armed mafia)
- Other differences
  - Set of arms not fixed
  - Delayed response, need batched updates
    - Scheme to serve items for next epoch of 5 minutes

Our Approach : Reduce dimension via Models

- Users/articles have informative covariates, can be used to group the bandits and bandit arms
  - E.g. Sports articles do well for users who like Y! Sports
- Learn models at very fine granularities using clever representations
  - E.g dimension reduction via latent factor models (matrix factorization as in Netflix)
- Couple this with simple bandit (Explore/Exploit) schemes to avoid “starving” some items
  - ε-greedy, Upper confidence bound (UCB), Posterior draws
  - Gittins’ style lookahead (more principled but computationally tractable only for some class of models)

Our approach (2)

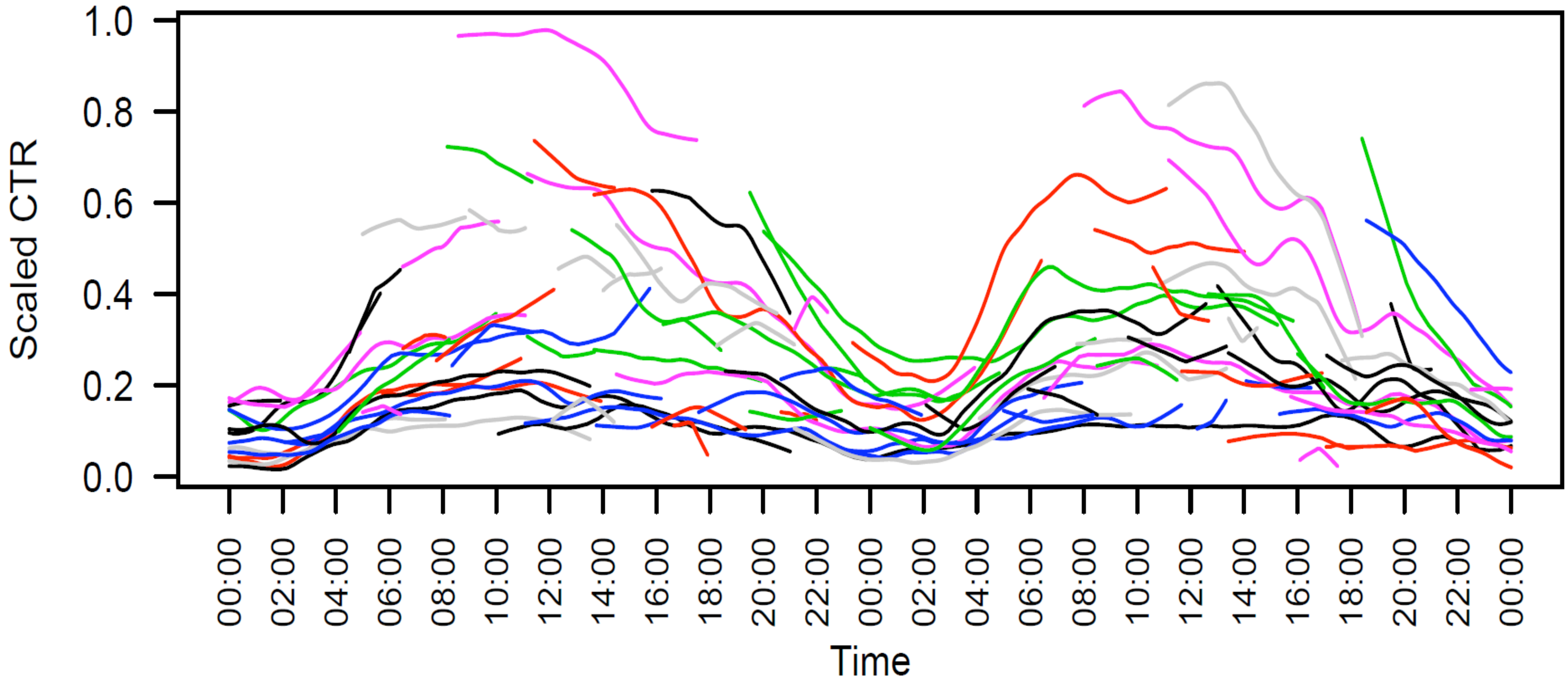
- Massively high dimensional bandit problem, extreme data sparseness
- Initialize through covariate based models+ fast online corrections at granular levels (item and/or user) + randomization through e/e schemes provide a powerful framework



Different kinds of recommendations

- Webpage has several modules, some may publish best of the web, some may do light personalization and some may do deep personalization
  - Selecting most popular with dynamic content pool
    - Time series, multi-armed bandits
  - Personalization using user covariates
    - Online logistic regression, reduced rank regression
  - Personalization based on covariates and past activity
    - Matrix factorization (bilinear random-effects model)

Article click rates over 2 days on Today module



No confounding, traffic obtained from a controlled randomized experiment

Things to note:

a) Short lifetimes b) temporal effects c) often breaking news story

Statistical Issues

- Temporal variations in article click-rates
- Short article lifetimes → quick reaction important
  - Cannot miss out on a breaking news story
  - Cold-start : rapidly learning click-rates of new articles

- Approach
  - Temporal - Time-series models coupled with
  - E/E(multi-armed bandits)
    - To handle cold-start

  - (Agarwal et al , NIPS 08, WWW 09, ICDM 09)

Time series Model: Kalman filter

- Dynamic Gamma-Poisson: click-rate evolves over time in a multiplicative fashion

$$c_t \mid n_t, p_t \sim \text{Poisson}(n_t p_t)$$

$$p_{t+1} = p_t \epsilon_{t+1}$$

$$\epsilon_{t+1} \sim \mathcal{D}(\text{mean} = 1, \text{var} = \eta) \longrightarrow \text{Gamma}$$

- Estimated Click-rate distribution at time t+1
  - Prior mean: $E(p_{t+1} \mid D_t) = \hat{p}_{t|t}$
  - Prior variance: $Var(p_{t+1} \mid D_t) = \hat{\sigma}^2_{t|t} + \eta(\hat{p}^2_{t|t} + \hat{\sigma}^2_{t|t})$

High CTR items more adaptive

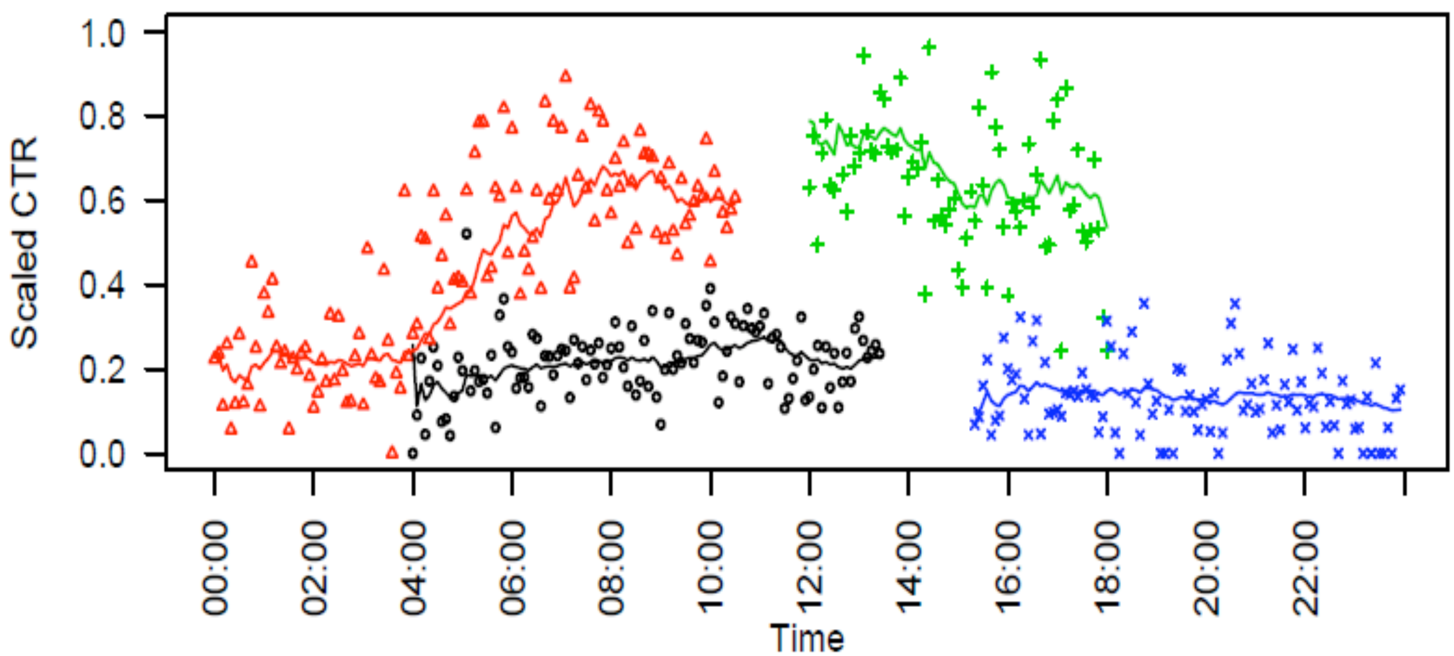
Updating the parameters at time t+1

- Fit a Gamma distribution to match the prior mean and prior variance at time t

- Combine this with Poisson likelihood at time t to get the posterior mean and posterior variance at time t+1
  - Combining Poisson with Gamma is easy, hence we fit a Gamma distribution to match moments

Tracking behavior of Gamma-Poisson model

- Low click rate articles – More temporal smoothing



Time series solution

- Works well and easy to generalize to
  - Segmented Most popular
  - If user population can be clustered into large number of sub-populations, this will work
    - (e.g. Decision Trees, Hierarchical clustering,..)

REGRESSION BASED PERSONALIZATION

Modeling

Algorithm selects article j with features $\boldsymbol{x}_j$

(keywords, content categories, ...)

User i visits with user features $\boldsymbol{x}_i$

(demographics, browse history, search history, …)

(i, j) : response y_{ij}

(rating or click/no-click)

Predict the unobserved entries based on features and the observed entries

Online Logistic regression

- Estimating (user, item) interactions for a large, unbalanced and massively incomplete 2-way binary response matrix
- Natural (simple) statistical model

$$y_{ijt} \sim \text{Bernoulli}(p_{ijt})$$

$$s_{ijt} = \log \frac{p_{ijt}}{1 - p_{ijt}}$$

Item coefficients

$$s_{ijt} = x'_{it} A x_j + x'_{it} v_{jt}$$

Offline initialization

High dimensional item parameters
In our examples, dimension ~ 1000
Online corrections

- Per-item online model
  - **must estimate quickly for new items**

Reduced Rank for our new article problem

- Generalize reduced rank for large incomplete matrix

$$s_{ijt} = \boldsymbol{x}'_{it}\mathbf{A}\boldsymbol{x}_j + \boldsymbol{x}'_{it}B\boldsymbol{\theta}_j$$

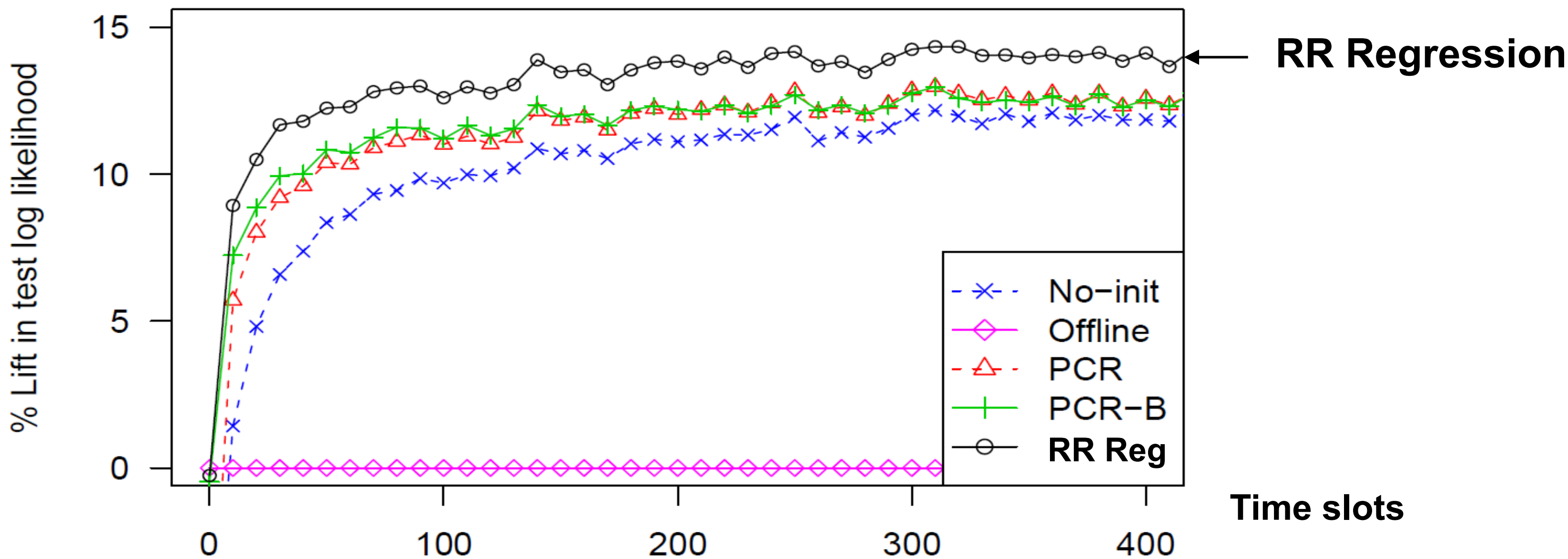
Low dimension (5-10),

B estimated retrospective data

Affinity to old items

- Application different than in classical reduced rank literature
  - Cold-start problem in recommender problems
  - Rank selected online through predictive log-likelihood

Per Item/User Feature model results (Agarwal et al, KDD 2010)



- Methods
  - **No-init:** Regular online logistic with ~1000 parameters for each item
  - **Offline:** Covariate-based model without online update
  - **PCR, PCR-B**: Principal component methods to estimate *B*
  - **RR Reg**: Reduced rank procedure

- Summary:
  - Reduced rank regression significantly improves performance compared to other baseline methods

MATRIX FACTORIZATION

PER USER, PER ARTICLE PERSONALIZATION

PROBLEM DEFINITION

- Models to predict CTR for *new* pairs
  - Warm-start: (user, item) present in the training data
  - Cold-start: At least one of (user, item) new

- Challenges
  - Highly incomplete (user, item) matrix
  - Heavy tailed degree distributions for users/items
    - Large fraction of ratings from small fraction of users/items
  - Handling both warm-start and cold-start effectively

Possible approaches

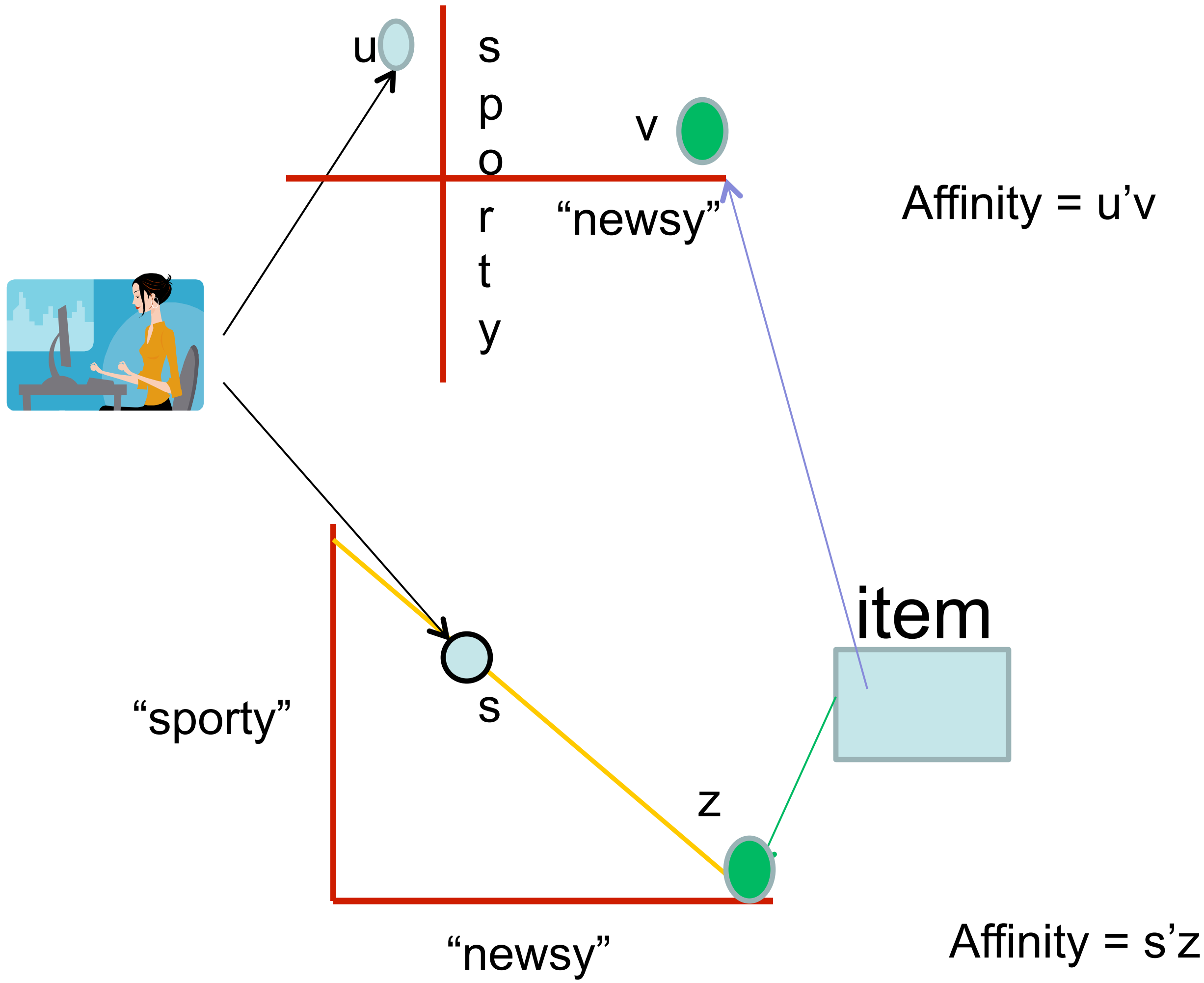
- Large scale regression based on covariates
  - Does not provide good estimates for heavy users/movies
  - Large number of predictors to estimate interactions
- Collaborative filtering
  - Neighborhood based
  - Factorization (our approach)
  - Good for warm-start; cold-start dealt with separately
- Single model that handles cold-start and warm-start
  - Heavy users/movies → User/movie specific model
  - Light users/movies → fallback on regression model
  - **Smooth** fallback mechanism for good performance

Classical work in Recommender Problems

- Combine feature based regressions with collaborative filtering style algorithms (item-item, user-user similarity based methods)
  - E.g. Linear combination of regression and CF per user
  - “Fill-up” based on regression, then apply CF
  - Filterbots : add psuedo users/items that are described through predictor combinations

- Drawbacks
  - Performing collaborative filtering using “imputation” from another content based model leads to bias in estimation
  - For linear combinations, a global combiner is not often suitable
    - Different (user, item) pair types require different combiners

Latent Factor Models



Euclidean Latent Factor Models (Srebro et al ICML 2004,Koren et al, KDD 2007, Ruslan et al, ICML 2008)

- Latent user factors: (α_i , $\mathbf{u_i}$=(u_{i1},…,u_{ir}))
- Latent item factors: (β_j , $\mathbf{v_j}$=(v_{j1},….,v_{jr}))

User's taste

Interaction

Item type

$$\alpha_i + \beta_j + u_i' v_j$$

- Too many parameters → will overfit (learn idiosyncrasies instead of pattern)
- Key technical issue: → *Regularization*
- Usual approach: → Constrain the length of factors

Cold-start and Warm start?

- Estimate new user latent factor using group average
  - E.g Users interested in Sports living in San Francisco

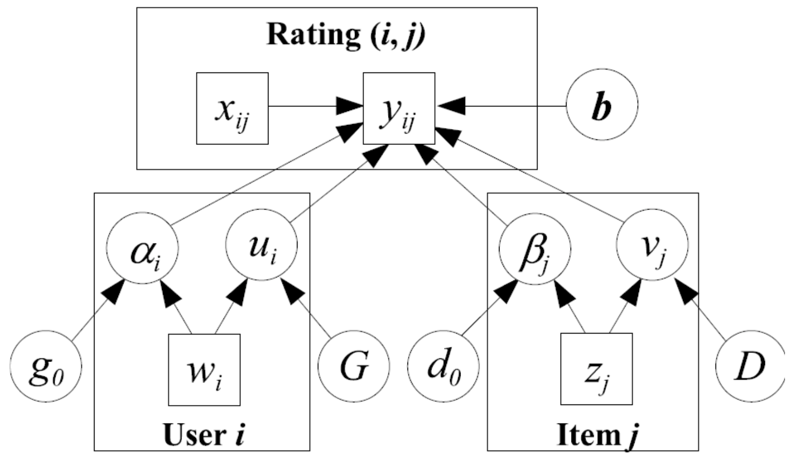
- Don’t map new users/items to origin a-priori, map them to centroids of user/item groups described through covariates
- What is the right grouping?
  - Different class of functions but all estimated from data

Formal description of how to constrain factors
Regression based latent factors (RLFM)

- Multi-level hierarchical model (*i* = user, *j* = article)
- First Stage (Observation equation)

$$y_{ij} \mid b, \alpha_i, \beta_j, \mathbf{u}_i, \mathbf{v}_j \sim N(\text{mean} = x_{ij}'b + \alpha_i + \beta_j + \mathbf{u}_i'\mathbf{v}_j)$$

- Second stage (State equation)



$$\mathbf{u}_i^{r \times 1} \sim MVN(G^{r \times p}\mathbf{w}_i, a_u I)$$

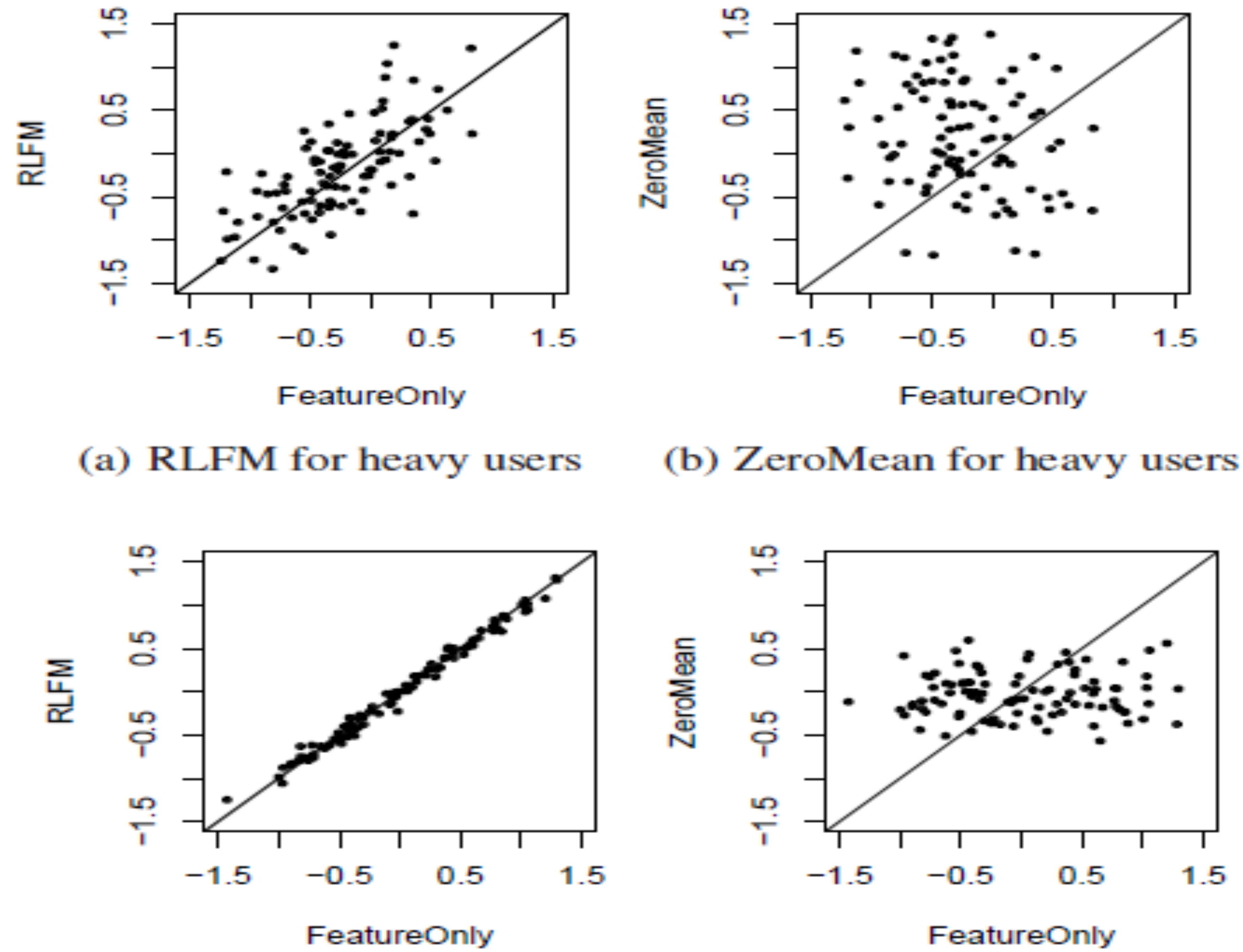
$$\mathbf{v}_j^{r \times 1} \sim MVN(D^{r \times q}\mathbf{z}_i, a_v I)$$

$$\alpha_i \sim N(g'\mathbf{w}_i, a_\alpha), \beta_j \sim N(d'\mathbf{z}_i, a_\beta)$$

$$\boldsymbol{\theta} = (G, D, g, d, a_u, a_v, a_\alpha, a_\beta)$$

Advantages of RLFM illustrated on Yahoo! FP

Only the first user factor plotted in the comparisons



(a) RLFM for heavy users (b) ZeroMean for heavy users

(c) RLFM for light users (d) ZeroMean for light users

Remarks

- Matrix factorization (successful in Netflix) is special case
  - Priors all centered around 0
- Multi-modal posterior, model fitting algorithm important
  - Stochastic EM (MCEM)
    - E-step: MCMC samples
    - M-step: Independent regressions run separately for user and article random effects using off-the-shelf packages
- Selecting rank *r*
  - Higher is typically better, more shrinkage of factors with large *r*
  - Computational dependency on *r* --- $O(r^3)$

r	1	3	5	10	20	30	40
Var-comp: a_u	0.234	0.123	0.075	0.047	0.028	0.020	0.017

Cold start and fast online updates

- For new user/article, factor estimates based on features

$$\mathbf{u}_{new} = \widehat{G}\mathbf{w}_{new}, \ \mathbf{v}_{new} = \widehat{D}\mathbf{z}_{new}$$

- For old user/article, factor estimates obtained through shrinkage

$$R_{ij} = y_{ij} - \widehat{\mu} - \widehat{\alpha}_i - \widehat{\beta}_j$$

$$E(\mathbf{u}_i \mid \text{Rest}) = \Big(\frac{\widehat{\sigma}^2}{\widehat{a}_u} + \sum_{j \in N_i} \mathbf{v}_j \mathbf{v}_j'\Big)^{-1} \Big(\frac{\widehat{\sigma}^2}{\widehat{a}_u} \widehat{G}\mathbf{w}_i + \sum_{j \in N_i} R_{ij} \mathbf{v}_j\Big)$$

- Linear combination of regression and user "ratings"
- Cold-start & warm-start dealt in ad-hoc ways in literature
  - Statistical models provide a rigorous and principled way

Fast Online Updates to factor estimates

- Important to learn quickly from recent data since process is non-stationary
  - This is accomplished by updating user/article factors quickly (e.g. every 5 minutes)
  - Posterior until previous epoch used as prior to initialize our regressions, this is combined with likelihood at current epoch to obtain new posteriors
    - Only E-step is required

Scalable computation

- User (resp. item) random effects for given item(resp. user) random effects conditionally independent
  - Computations can be parallelized in the E-step

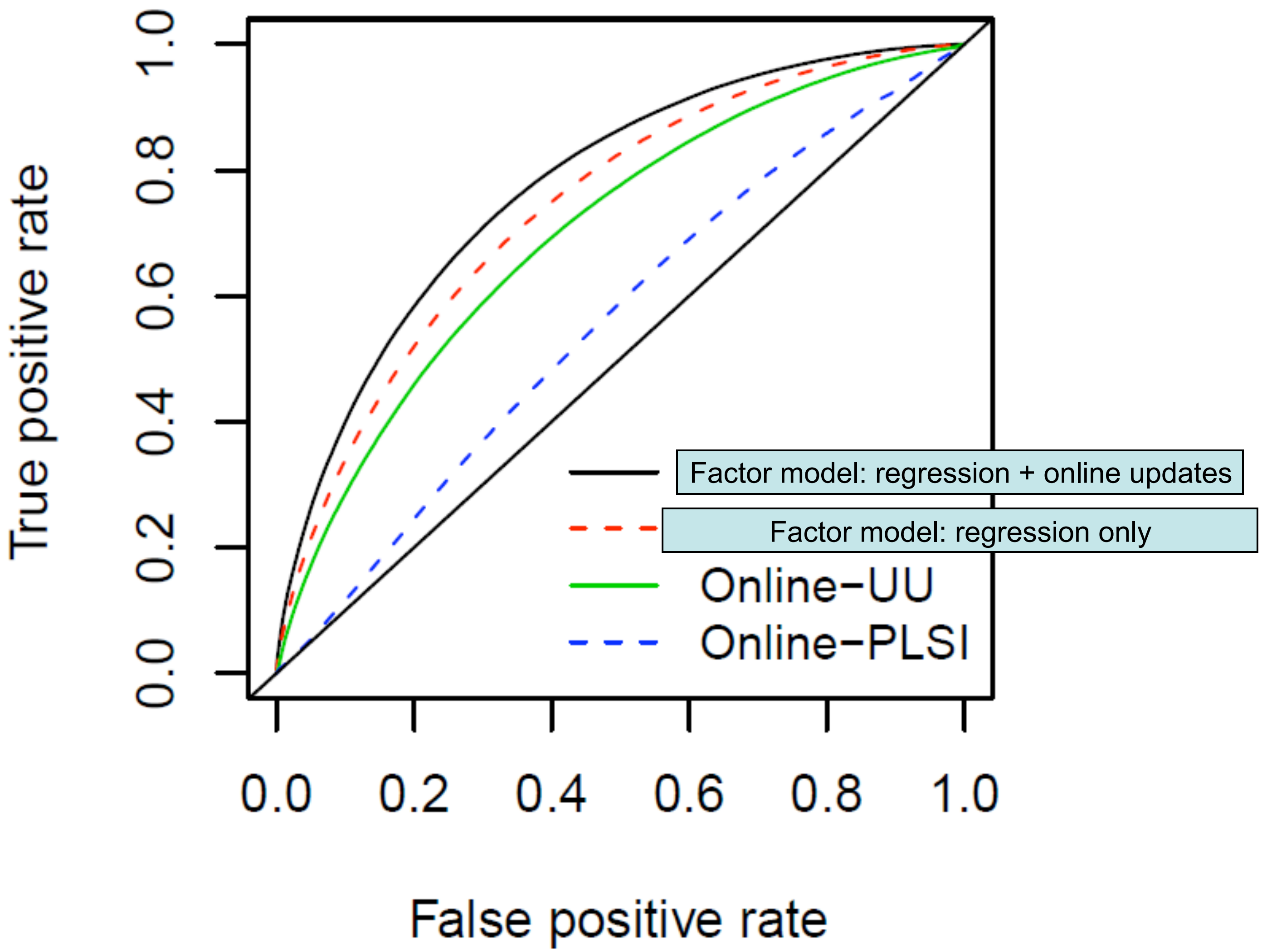
- Scalable off-the-shelf regression code in M-step

- For data that does not fit into memory, we use map-reduce
  - Distributed computing on massive cluster of commodity PCs

Data Example

- 2M binary observations by 30K heavy users on 4K articles
  - Heavy user ---- at least 30 visits to the portal in last 5 months
- Article covariates
  - Editorially labeled category information (~50 covariates)
- User covariates
  - Demographics, browse behavior (~1K covariates)
- Training/test split by timestamp of events (75/25)
- Methods
  - Factor model with regression, no online updates
  - Factor model with regression + online updates
  - Online model based on user-user similarity (Online-UU)
  - Online probabilistic latent semantic index (Online-PLSI)

ROC curve



True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
Factor model: regression + online updates
Factor model: regression only
Online−UU
Online−PLSI

Other functions to get groupings

- We ran several other non-linear regressions in M-step

$$\mathbf{u}_i^{r \times 1} \sim MVN(G^{r \times p}\mathbf{w}_i, a_u I)$$

$$\mathbf{v}_j^{r \times 1} \sim MVN(D^{r \times q}\mathbf{z}_i, a_v I)$$

$$\alpha_i \sim N(g'\mathbf{w}_i, a_\alpha), \beta_j \sim N(d'\mathbf{z}_i, a_\beta)$$

$$\boldsymbol{\theta} = (G, D, g, d, a_u, a_v, a_\alpha, a_\beta)$$

Replace Linear models *Gw* and *Dz* by non-linear functions

- LASSO, Random forests, Gradient Boosted DT, BART,
- Also tried LDA (discrete factors instead of continuous), Markov random field priors on factors
  - The improvements on our datasets are small
  - Random forests, BART, GBDT appear to be the best methods
  - On simulated data where we assume non-linearity, improvements are substantial

Results on Benchmark movie dataset (RMSE)

- 1M observations
  - User rating on movies
- User covariates
  - Age, gender, zipcode, occupation
- Movies
  - Genre + other information obtained from IMDb
- Time-based training/test split (75/25)

Method	RMSE
Model with Online Updates & regression	0.8429
Linear Model in regression	0.9363
Zero-mean prior	0.9422
Regression only	1.091
Item popularity only	0.9726
FilterBot	0.9517
BART in regression	0.9340
Random forest In regression	0.9343
GB Decision trees In regression	0.9344

Another Interesting Regularization on the factors

To incorporate neighborhood information like social network, hierarchies etc to regularize the factor estimates

Lu, Agarwal and Dhillon RecSys 2009

$$u_i | u_{-i} \sim MVN\left(\sum_{j:j \in \mathcal{N}_i} \rho w_{ij} u_j / w_{i+}, \tau^2 / w_{i+} \right)$$

$$(u_1, \cdots, u_N) \sim MVN(\mathbf{0}, (D - \rho W) \otimes I)$$

Item factors on the simplex: fLDA

- Model the rating y_{ij} that user *i* gives to item *j* as the user's affinity to the topics that the item has

User *i* 's affinity to topic *k*

$$y_{ij} = \ldots + \sum_k s_{ik} \bar{z}_{jk}$$

Pr(item *j* has topic *k*) estimated by averaging the LDA topic of each word in item *j*

Old items: z_{jk}'s are Item latent factors learnt from data with the LDA prior
New items: z_{jk}'s are predicted based on the bag of words in the items

- Unlike regular unsupervised LDA topic modeling, here the LDA topics are learnt in a supervised manner based on past rating data
- Our model can be thought of as a "multi-task learning" version of the supervised LDA model [Blei'07] for cold-start recommendation

Experimental Results: MovieLens 1M Dataset

- Task: Predict the rating that a user would give a movie
- Training/test split:
  - Sort observations by time
  - First 75% → Training data
  - Last 25% → Test data
- Item warm-start scenario
  - Only 2% new items in test data

Model	Test RMSE
RLFM	0.9363
fLDA	0.9381
Factor-Only	0.9422
FilterBot	0.9517
unsup-LDA	0.9520
MostPopular	0.9726
Feature-Only	1.0906
Constant	1.1190

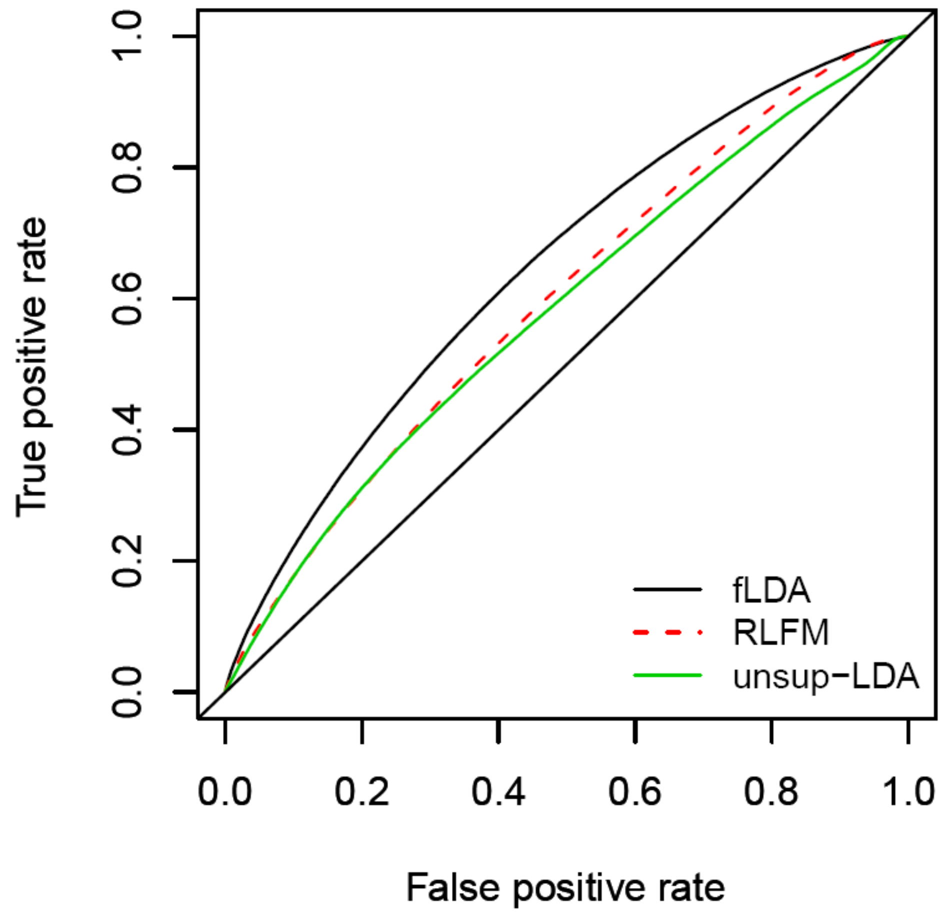
fLDA is as strong as the best method
It does not reduce the performance in warm-start scenarios

Experimental Results: Yahoo! Buzz Dataset

- Task: Predict whether a user would buzz-up an article
- Severe item cold-start
  - All items are new in test data

fLDA significantly outperforms other models

Data Statistics
1.2M observations
4K users
10K articles



Post-click utilities

Recommender + EDITORIAL

content

Clicks on FP links influence downstream supply distribution

SPORTS

NEWS

OMG

FINANCE

AD SERVER

PREMIUM DISPLAY (GUARANTEED)

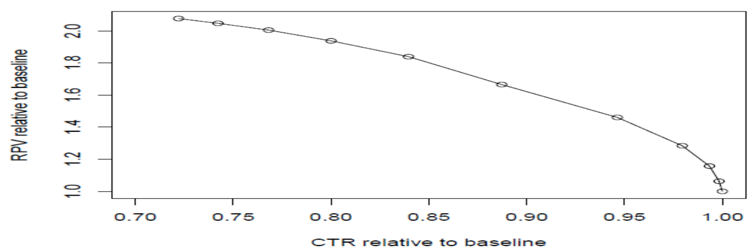
NETWORK PLUS (Non-Guaranteed)

Downstream engagement

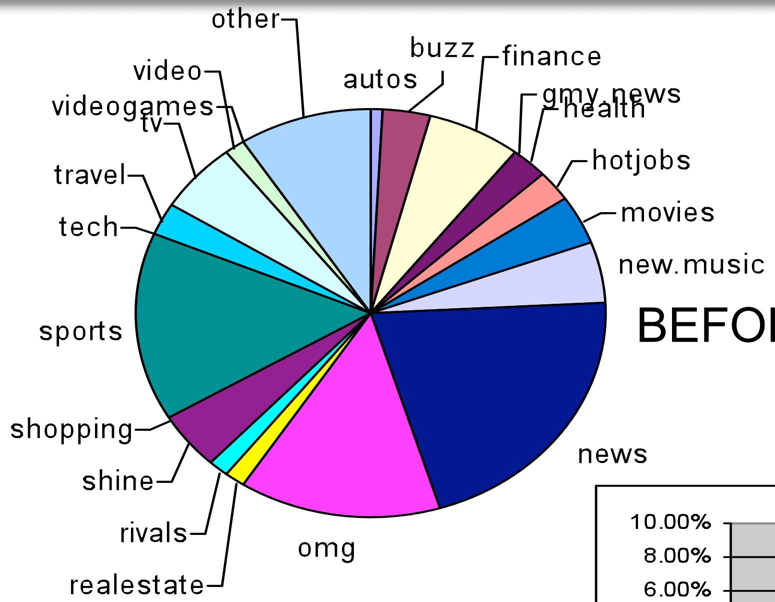
(Time spent)

Serving Content on Front Page: Click Shaping

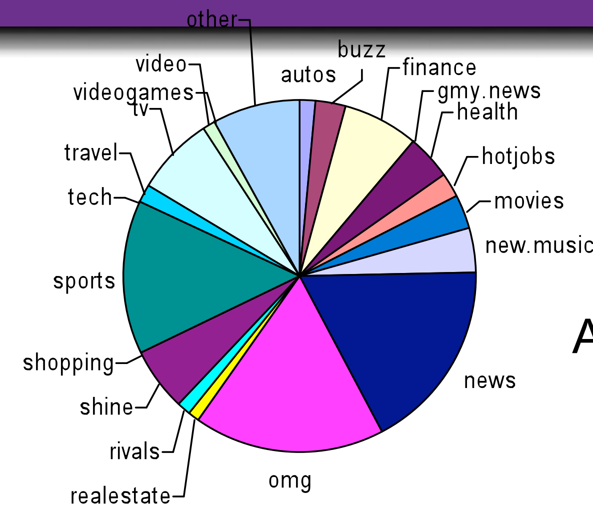
- What do we want to optimize?
- Current: Maximize clicks (maximize downstream supply from FP)
- But consider the following
  - Article 1: CTR=5%, utility per click = 5
  - Article 2: CTR=4.9%, utility per click=10
    - By promoting 2, we lose 1 click/100 visits, gain 5 utils
- If we do this for a large number of visits --- lose some clicks but obtain significant gains in utility?
  - E.g. lose 5% relative CTR, gain 40% in utility (revenue, engagement, etc)



Why call it Click Shaping?

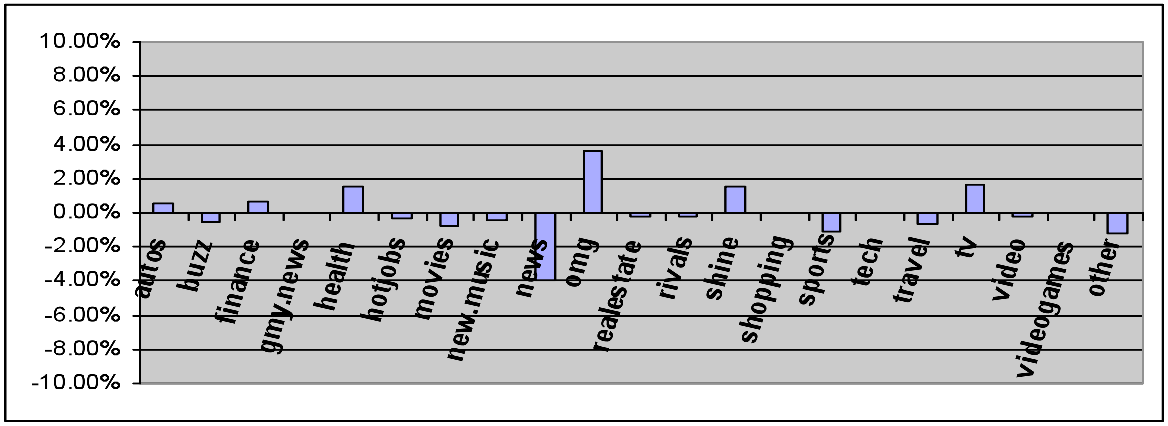


BEFORE



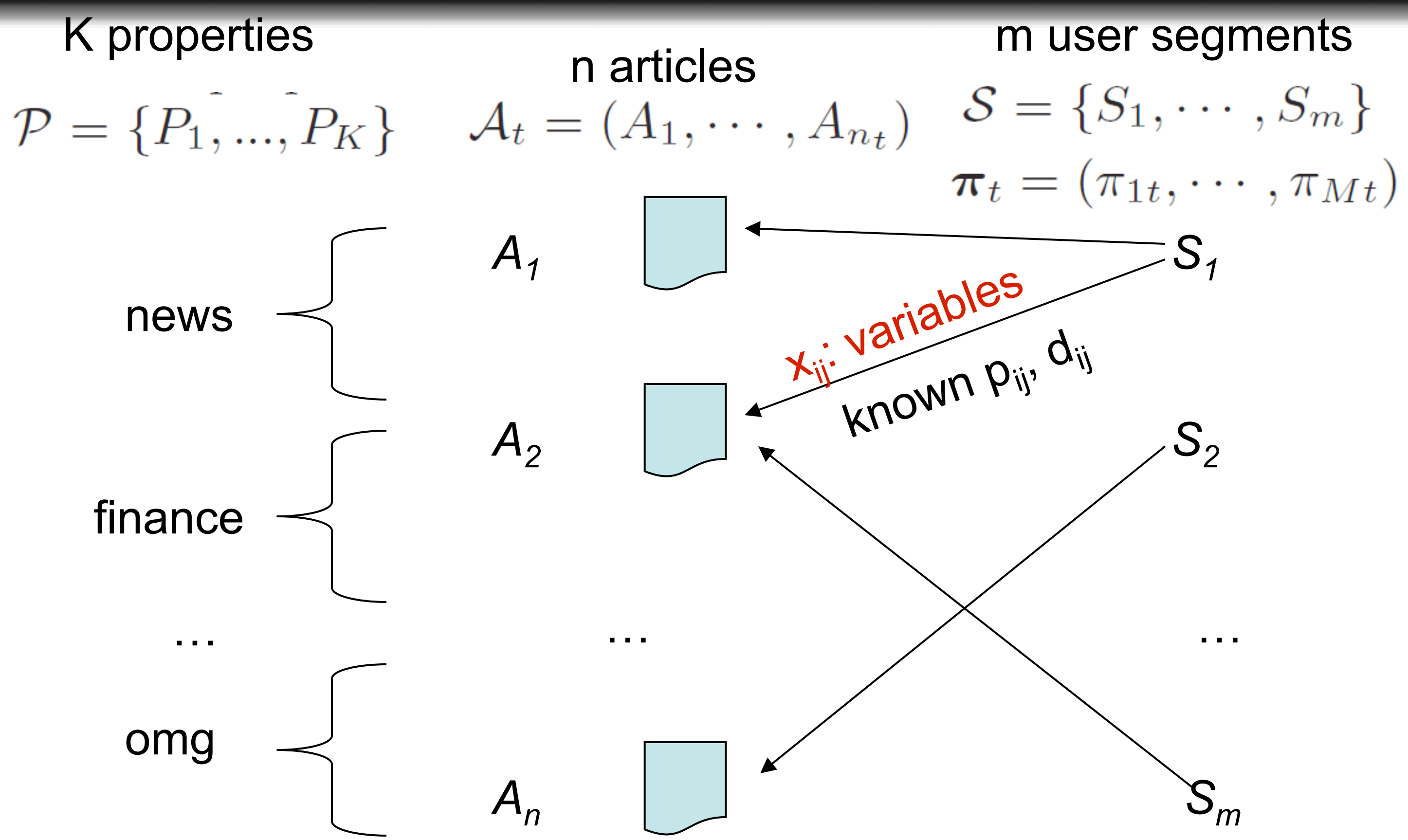
AFTER

Supply distribution
Changes



SHAPING can happen with respect to any downstream metrics (like engagement)

Multi-Objective Optimization



- CTR of user segment *i* on article *j*: p_{ij}
- Time duration of *i* on *j*: d_{ij}

Multi-Objective Program

- Scalarization (s-MOP)

$$\lambda \cdot TotalClicks(\mathbf{x}) + (1-\lambda) \cdot Downstream(\mathbf{x})$$

$$x_{ij} = \begin{cases} 1, & \text{if } j = \arg\max_J \lambda \cdot p_{iJ} + (1-\lambda) \cdot p_{iJ} d_{iJ} \\ 0, & \text{otherwise} \end{cases}$$

- Global Programming (g-MOP)

$$\begin{aligned} &\text{maximize } Downstream(\mathbf{x}) \\ &\quad \text{s.t. } TotalClicks(\mathbf{x}) \geq \alpha \cdot TotalClicks^* \end{aligned}$$

Simplex constraints on x_{iJ} is always applied

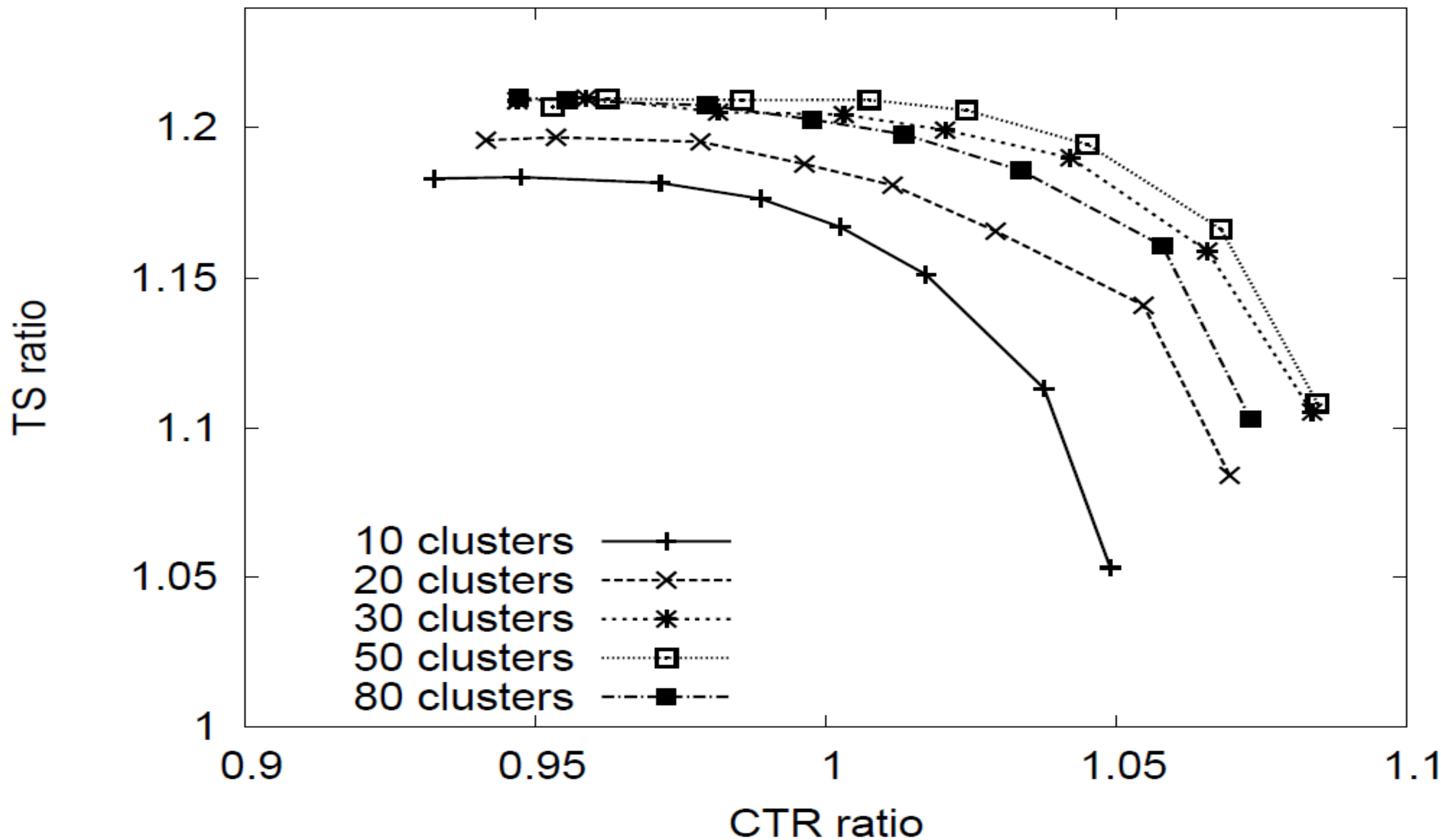
Constraints are linear

Every 10 mins, solve x

Use this x as the serving scheme in the next 10 mins

56

Pareto-optimal solution (more in KDD 2011)



57

Scaling up Regression Matrix Factorization through Map-reduce

- Hundreds of millions of observations, several million users
  - Data cannot fit into memory, stored in distributed file systems.
- Run our full fitting procedure every day
- Challenging
  - 100's of millions of users
- Divide and conquer + model averaging
  - Partition users into K balanced, random clusters
  - Run our model on each cluster
  - Combine estimates of regression coefficients, re-do E-step for each local model
  - Run ensemble of such models (each ensemble member formed by using different random clustering); perform model averaging
- Fast online updates every 5 minutes for factors

Other problems

- Multi-module (Joint work with Mike West at Duke statistics)
  - Incorporate correlation

- Multi-Context learning (submitted to KDD 2011)
  - Learn using feedback obtained from different Yahoo! pages
    - E.g. can we perform better on Front Page by using feedback on News page?

- Model based Item-item similarity through partial correlations using Graphical LASSO
  - To appear in Annals of Applied Statistics

- Recommending related content to increase engagement
  - Users who read article about Obama also like Palin Video

Summary

- Recommending Content on Web is a new scientific discipline with several challenging problems
- Large data, extreme heterogeneity, ability to run experiments on live traffic, blending statistical output with optimization
- Online models initialized via offline models and tightly coupled with explore/exploit provide an effective strategy
- Yahoo! is an ideal place to pursue such a research
  - 600M users/month, leader in several verticals (Sports, Finance, News, Entertainment,..) and Y! Front Page most visited content page on the planet