Offline Components:
Collaborative Filtering in Cold-start Situations

Problem

Algorithm selects Item j with item 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}

(explicit rating, implicit click/no-click)

Predict the unobserved entries based on features and the observed entries

Model Choices

- Feature-based (or content-based) approach
  - Use features to predict response
    - (regression, Bayes Net, mixture models, …)
  - Limitation: need predictive features
    - Bias often high, does not capture signals at granular levels
- Collaborative filtering (CF aka Memory based)
  - Make recommendation based on past user-item interaction
    - User-user, item-item, matrix factorization, …
    - See [Adomavicius & Tuzhilin, TKDE, 2005], [Konstan, SIGMOD'08 Tutorial], etc.
  - Better performance for old users and old items
  - Does not naturally handle new users and new items (**cold-start**)

Collaborative Filtering (Memory based methods)

User-User Similarity

$$p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i)$$

Item-Item similarities, incorporating both

Estimating Similarities

- Pearson's correlation
- Optimization based (Koren et al)

How to Deal with the Cold-Start Problem

- Heuristic-based approaches
  - Linear combination of regression and CF models
  - Filterbot
    - Add user features as psuedo users and do collaborative filtering
  - Hybrid approaches
    - Use content based to fill up entries, then use CF
- Matrix Factorization
  - Good performance on Netflix (Koren, 2009)
- Model-based approaches
  - Bilinear random-effects model (probabilistic matrix factorization)
    - Good on Netflix data [Ruslan et al ICML, 2009]
  - Add feature-based regression to matrix factorization
    - (Agarwal and Chen, 2009)
  - Add topic discovery (from textual items) to matrix factorization
    - (Agarwal and Chen, 2009; Chun and Blei, 2011)

Per-item regression models

- When tracking users by cookies, distribution of visit patters could get extremely skewed
  - Majority of cookies have 1-2 visits

- Per item models (regression) based on user covariates attractive in such cases

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

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

$$s_{ijt} = \boldsymbol{x}_{it}^{'} \mathbf{A} \boldsymbol{x}_j + \boldsymbol{x}_{it}^{'} \boldsymbol{v}_{jt}$$

Several per-item regressions: Multi-task learning

- Agarwal,Chen and Elango, KDD, 2010

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

•Low dimension
•(5-10),

•*B* estimated
•retrospective data

Affinity to old items

Per-user, per-item models via bilinear random-effects model

Motivation

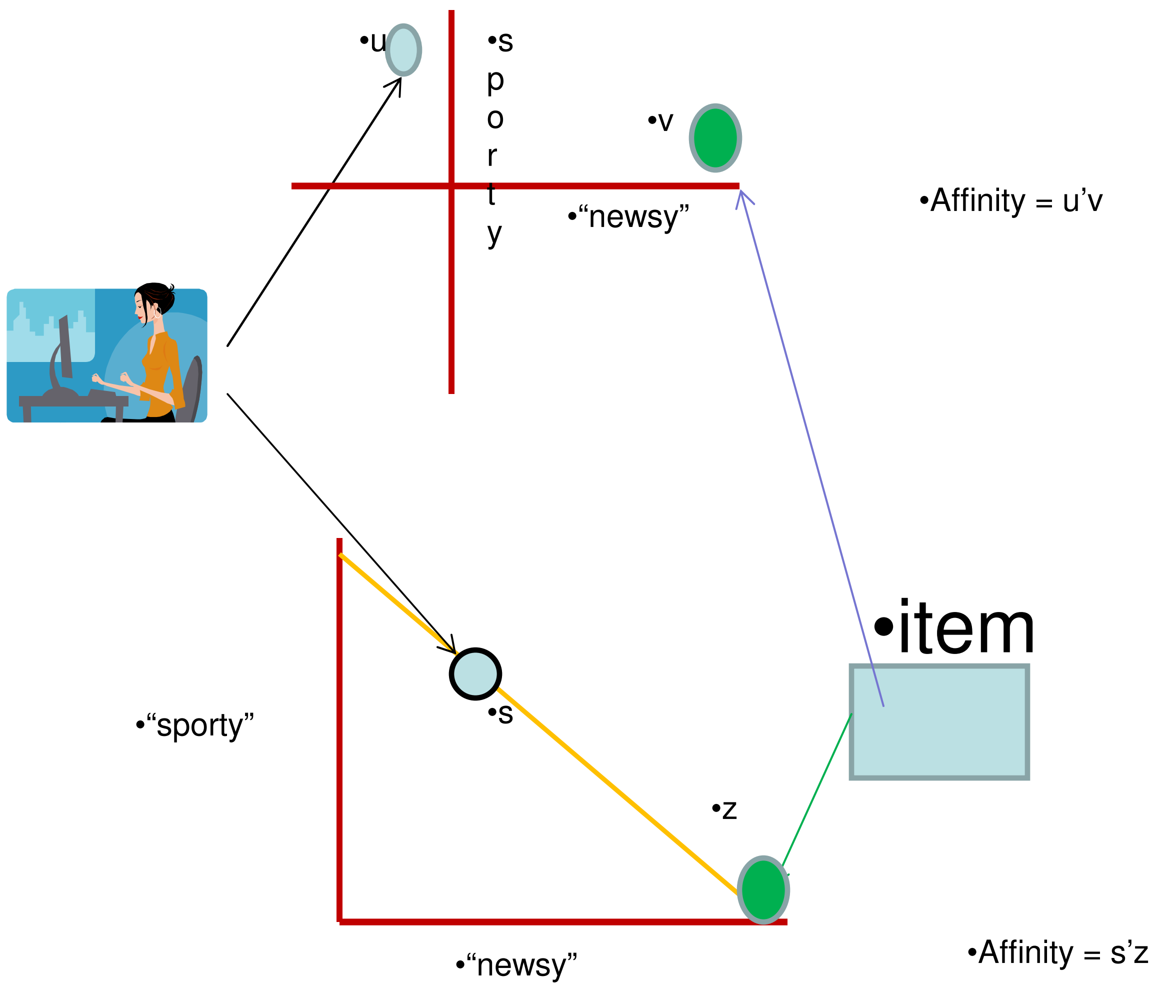
- Data measuring k-way interactions pervasive
  - Consider k = 2 for all our discussions
    - E.g. User-Movie, User-content, User-Publisher-Ads,….
  - Power law on both user and item degrees

- Classical Techniques
  - Approximate matrix through a singular value decomposition (SVD)
    - After adjusting for marginal effects (user pop, movie pop,..)
  - Does not work
    - Matrix highly incomplete, severe over-fitting
  - Key issue
    - Regularization of eigenvectors (factors) to avoid overfitting

Early work on complete matrices

- Tukey's 1-df model (1956)
  - Rank 1 approximation of small nearly complete matrix
- Criss-cross regression (Gabriel, 1978)
- Incomplete matrices: Psychometrics (1-factor model only; small data sets; 1960s)
- Modern day recommender problems
  - Highly incomplete, large, noisy.

Latent Factor Models



Factorization – Brief Overview

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

Interaction

$$E(y_{ij}) = \mu + \alpha_i + \beta_j + u_i' B v_j$$

- (Nn + Mm) parameters → will overfit for moderate values of n,m
- Key technical issue: → *Regularization*

Latent Factor Models: Different Aspects

- Matrix Factorization
  - Factors in Euclidean space
  - Factors on the simplex
- Incorporating features and ratings simultaneously
- Online updates

Maximum Margin Matrix Factorization (MMMF)

- Complete matrix by minimizing loss (hinge,squared-error) on observed entries subject to constraints on trace norm
  - Srebro, Rennie, Jakkola (NIPS 2004)
    - Convex, Semi-definite programming (expensive, not scalable)
- Fast MMMF (Rennie & Srebro, ICML, 2005)
  - Constrain the Frobenious norm of left and right eigenvector matrices, not convex but becomes scalable.
- Other variation: Ensemble MMMF (DeCoste, ICML2005)
  - Ensembles of partially trained MMMF (some improvements)

Matrix Factorization for Netflix prize data

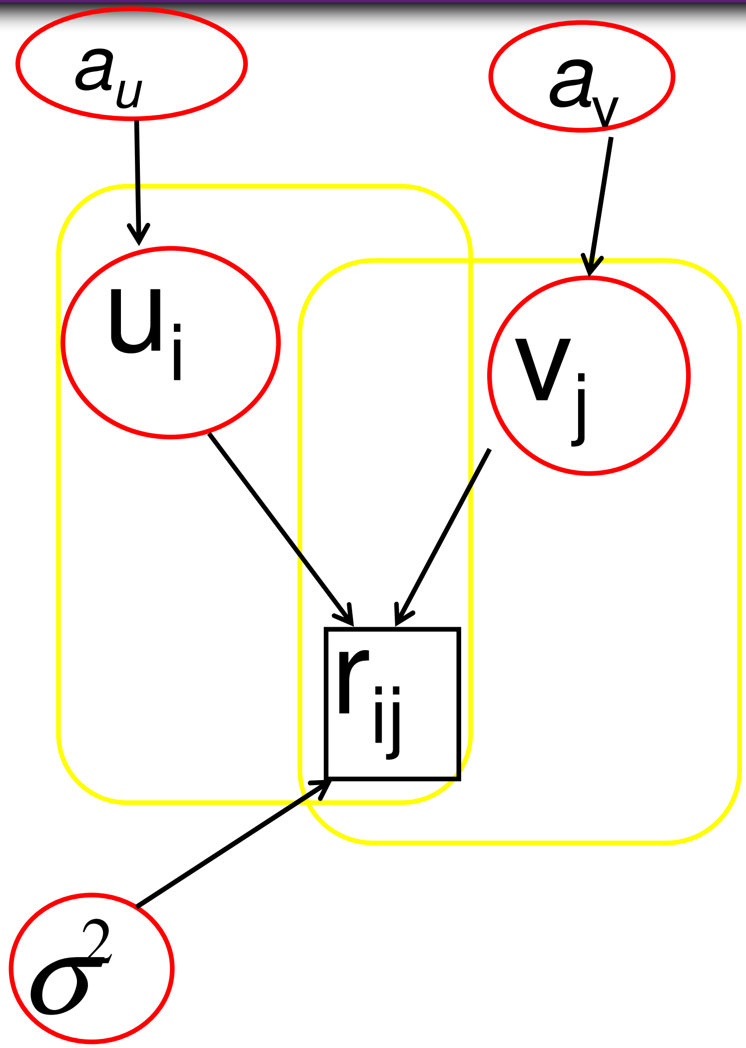
- Minimize the objective function

$$\sum_{ij \in obs} (r_{ij} - u_i^T v_j)^2 + \lambda(\sum_i |u_i|^2 + \sum_j |v_j|^2)$$

- Simon Funk: Stochastic Gradient Descent

- Koren et al (KDD 2007): Alternate Least Squares
  - They move to SGD later in the competition

Probabilistic Matrix Factorization (Ruslan & Minh, 2008, NIPS)



- Optimization is through Gradient Descent (Iterated conditional modes)

- Other variations like constraining the mean through sigmoid, using “who-rated-whom”

- Combining with Boltzmann Machines also improved performance

$$r_{ij} \sim N(\mathbf{u}_i^T \mathbf{v}_j, \sigma^2)$$

$$\mathbf{u}_i \sim MVN(\mathbf{0}, a_u I)$$

$$\mathbf{v}_j \sim MVN(\mathbf{0}, a_v I)$$

Bayesian Probabilistic Matrix Factorization (Ruslan and Minh, ICML 2008)

- Fully Bayesian treatment using an MCMC approach
- Significant improvement

- Interpretation as a fully Bayesian hierarchical model shows why that is the case
  - Failing to incorporate uncertainty leads to bias in estimates
  - Multi-modal posterior, MCMC helps in converging to a better one

Latent dimension r	2	5	10	15
ICM	.9736	.9729	.9799	.9802
MCEM	.9728	.9722	.9714	.9715

MCEM also more resistant to over-fitting

•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

Non-parametric Bayesian matrix completion (Zhou et al, SAM, 2010)

- Specify rank probabilistically (automatic rank selection)

$$y_{ij} \sim N(\sum_{k=1}^{r} z_k u_{ik} v_{jk}, \sigma^2)$$

$$z_k \sim Ber(\pi_k)$$

$$\pi_k \sim Beta(a/r, b(r-1)/r)$$

$$z_k \sim Ber(1, a/(a + b(r-1)))$$

$$E(\#\,\text{Factors}) = ra/(a + b(r-1))$$

How to incorporate features:
Deal with both warm start and cold-start

- Models to predict ratings for *new* pairs
  - Warm-start: (user, movie) present in the training data with large sample size
  - Cold-start: At least one of (user, movie) new or has small sample size
    - Rough definition, warm-start/cold-start is a continuum.

- Challenges
  - Highly incomplete (user, movie) matrix
  - Heavy tailed degree distributions for users/movies
    - Large fraction of ratings from small fraction of users/movies
  - Handling both warm-start and cold-start effectively in the presence of predictive features

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
    - 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

Add Feature-based Regression into Matrix Factorization

RLFM: Regression-based Latent Factor Model

Regression-based Factorization Model (RLFM)

- Main idea: Flexible prior, predict factors through regressions
- Seamlessly handles cold-start and warm-start
- Modified state equation to incorporate covariates

RLFM: Model

- **Rating:** user *i* gives item *j*

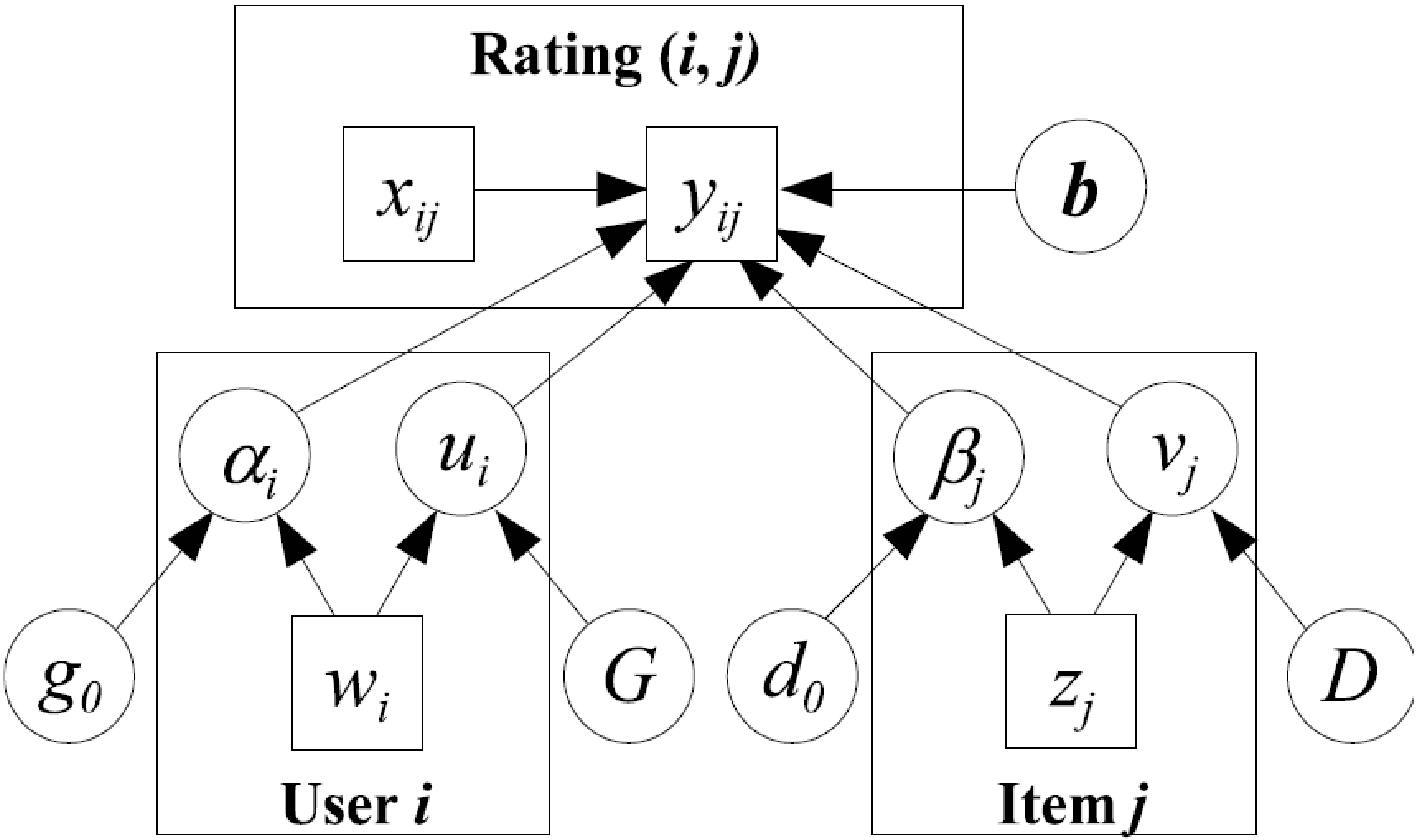
	$y_{ij} \sim N(\mu_{ij}, \sigma^2)$	Gaussian Model
	$y_{ij} \sim Bernoulli(\mu_{ij})$	Logistic Model (for binary rating)
	$y_{ij} \sim Poisson(\mu_{ij} N_{ij})$	Poisson Model (for counts)

$$t(\mu_{ij}) = x_{ij}^t b + \alpha_i + \beta_j + u_i^t v_j$$

- **Bias of user *i*:** $\alpha_i = g_0^t x_i + \varepsilon_i^\alpha, \quad \varepsilon_i^\alpha \sim N(0, \sigma_\alpha^2)$
- **Popularity of item *j*:** $\beta_j = d_0^t x_j + \varepsilon_j^\beta, \quad \varepsilon_j^\beta \sim N(0, \sigma_\beta^2)$
- **Factors of user *i*:** $u_i = G x_i + \varepsilon_i^u, \quad \varepsilon_i^u \sim N(0, \sigma_u^2 I)$
- **Factors of item *j*:** $v_i = D x_j + \varepsilon_i^v, \quad \varepsilon_i^v \sim N(0, \sigma_v^2 I)$

Could use other classes of regression models

Graphical representation of the model



Advantages of RLFM

- Better regularization of factors
  - Covariates "shrink" towards a better centroid

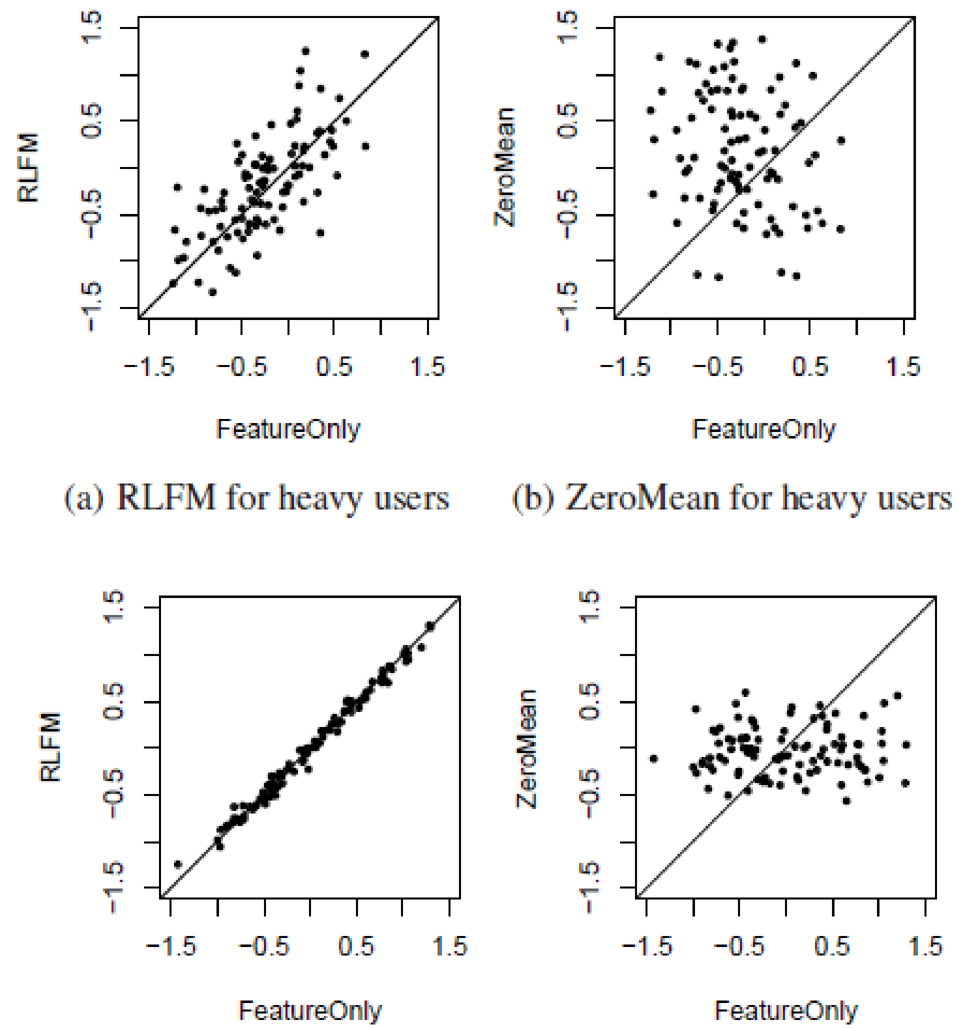
- Cold-start: Fallback regression model (**FeatureOnly**)

$$y_{ij} \sim N(m_{ij}, \sigma^2)$$

$$m_{ij} = x'_{ij}\boldsymbol{b} + g'_0 w_i + d'_0 z_j + w'_i G' D z_j$$

RLFM: Illustration of Shrinkage

Plot the first factor value for each user (fitted using Yahoo! FP data)



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

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

Induced correlations among observations

Hierarchical random-effects model

$$\overline{y_{ij} \sim N(m_{ij}, \sigma^2)}$$

$$\underset{\downarrow}{}$$

$$x'_{ij}\boldsymbol{b} + \alpha_i + \beta_j + u'_i v_j$$

Marginal distribution obtained by integrating out random effects

$$\begin{aligned}
\alpha_i &= g'_0 w_i + \epsilon_i^{\alpha}, & \epsilon_i^{\alpha} &\sim N(0, a_{\alpha}) \\
\beta_j &= d'_0 z_j + \epsilon_j^{\beta}, & \epsilon_j^{\beta} &\sim N(0, a_{\beta}) \\
u_i &= G w_i + \epsilon_i^{u}, & \epsilon_i^{u} &\sim MVN(\mathbf{0}, A_u) \\
v_j &= D z_j + \epsilon_j^{v}, & \epsilon_j^{v} &\sim MVN(\mathbf{0}, A_v)
\end{aligned}$$

Closer look at induced marginal correlations

$$E(y_{ij}) = x'_{ij}\boldsymbol{b} + g'_0 w_i + d'_0 z_j + w'_i G' D z_j$$

$$Var(y_{ij}) = \sigma^2 + a_\alpha + a_\beta + tr(A_u A_v) + z'_j D' A_u D z_j + w'_i G' A_v G w_i$$

$$cov(y_{ij}, y_{ij^*}) = a_\alpha + z'_j D' A_u D z_{j^*}$$

$$cov(y_{ij}, y_{i^*j}) = a_\beta + w'_i G' A_v G w_{i^*}$$

Model fitting: EM for our class of models

$$\boldsymbol{Y} : \text{ Data}$$

$$\boldsymbol{\Delta} : \text{ Latent variables}$$

$$\boldsymbol{\Theta} : \text{ hyper-parameters}$$

$$\text{Model: } p(\boldsymbol{Y}|\boldsymbol{\Delta},\boldsymbol{\Theta})p(\boldsymbol{\Delta}|\boldsymbol{\Theta})$$

$$\text{Output needed: } \text{ Mode: } max_{\boldsymbol{\Theta}} p(\boldsymbol{\Theta}|\boldsymbol{Y})$$

$$p(\boldsymbol{\Delta}|\boldsymbol{Y}) \approx p(\boldsymbol{\Delta}|\boldsymbol{Y},\hat{\boldsymbol{\Theta}})$$

The parameters for RLFM

- Latent parameters

$$\Delta = (\{\alpha_i\}, \{\beta_j\}, \{u_i\}, \{v_j\})$$

- Hyper-parameters

$$\Theta = (\mathbf{b}, \mathrm{G}, \mathrm{D}, \mathrm{A_u} = \mathrm{a_u I}, \mathrm{A_v} = \mathrm{a_v I})$$

Computing the mode

$$log(p(\boldsymbol{\Theta}|\boldsymbol{Y})) = log(p(\boldsymbol{\Theta}, \boldsymbol{\Delta}|\boldsymbol{Y})) - log(p(\boldsymbol{\Delta}|\boldsymbol{\Theta}, \boldsymbol{Y}))$$

$$log(p(\boldsymbol{\Theta}|\boldsymbol{Y})) = E_{old}(log(p(\boldsymbol{\Theta}, \boldsymbol{\Delta}|\boldsymbol{Y}))) - E_{old}(log(p(\boldsymbol{\Delta}|\boldsymbol{\Theta}, \boldsymbol{Y})))$$

$$E_{old} : \text{ Expectation w.r.t. } p(\boldsymbol{\Delta}|\boldsymbol{\Theta}_{old}, \boldsymbol{Y})$$

Second term: Minimized . at $\boldsymbol{\Theta}_{old}$

Find new value of $\boldsymbol{\Theta}$ that increases first term

The EM algorithm

Initialize $\boldsymbol{\Theta}$

Iterate

$$\text{E-step} \quad : E_{old}(log(p(\boldsymbol{\Theta}, \boldsymbol{\Delta}|\boldsymbol{Y})))$$

$$\text{M-step} \quad : argmax_{\boldsymbol{\Theta}} E_{old}(log(p(\boldsymbol{\Theta}, \boldsymbol{\Delta}|\boldsymbol{Y})))$$

Computing the E-step

- Often hard to compute in closed form
- Stochastic EM (Markov Chain EM; MCEM)
  - Compute expectation by drawing samples from

$$p(\boldsymbol{\Delta}|\boldsymbol{\Theta}_{old}, \boldsymbol{Y})$$

  - Effective for multi-modal posteriors but more expensive
- Iterated Conditional Modes algorithm (ICM)
  - Faster but biased hyper-parameter estimates

$$\text{Approximate } E_{old}(log(p(\boldsymbol{\Theta}, \boldsymbol{\Delta}|\boldsymbol{Y})))$$

$$\text{by } log(p(\boldsymbol{\Theta}_{old}, \hat{\boldsymbol{\Delta}}|\boldsymbol{Y}))$$

$$\hat{\boldsymbol{\Delta}} = argmax_{\boldsymbol{\Delta}} log(p(\boldsymbol{\Theta}_{old}, \boldsymbol{\Delta}|\boldsymbol{Y})$$

Monte Carlo E-step

- Through a vanilla Gibbs sampler (conditionals closed form)

$$\text{Let } o_{ij} = y_{ij} - \alpha_i - \beta_j - x'_{ij}\boldsymbol{b}$$

$$Var[u_i|\text{Rest}] = (A_u^{-1} + \sum_{j \in \mathcal{J}_i} \frac{v_j v_j'}{\sigma_{ij}^2})^{-1}$$

$$E[u_i|\text{Rest}] = Var[u_i|\text{Rest}](A_u^{-1} G w_i + \sum_{j \in \mathcal{J}_i} \frac{o_{ij} v_j}{\sigma_{ij}^2})$$

- Other conditionals also Gaussian and closed form
- Conditionals of users (movies) sampled simultaneously
- Small number of samples in early iterations, large numbers in later iterations

M-step (Why MCEM is better than ICM)

- Update G, optimize

$$(E^*(u_{il}) - Gw_i)'(E^*(u_{il}) - Gw_i)$$

- Update $A_u = a_u I$

$$\hat{a_u} = \frac{\sum_{i=1}^{M}(E^*(u_i) - \hat{G}w_i)'(E^*(u_i) - \hat{G}w_i) + \sum_{k=1}^{r} Var^*(u_{ikl})}{Mr}$$

Ignored by ICM, underestimates factor variability
Factors over-shrunk, posterior not explored well

Experiment 1: Better regularization

- MovieLens-100K, avg RMSE using pre-specified splits
- ZeroMean, RLFM and FeatureOnly (no cold-start issues)
- Covariates:
  - Users : age, gender, zipcode (1st digit only)
  - Movies: genres

	RLFM	*ZeroMean*	*FeatureOnly*
MovieLens-100K	0.8956	0.9064	1.0968

Experiment 2: Better handling of Cold-start

- MovieLens-1M; EachMovie
- Training-test split based on timestamp
- Same covariates as in Experiment 1.

	MovieLens-1M			EachMovie		
Model	30%	60%	75%	30%	60%	75%
RLFM	0.9742	0.9528	0.9363	1.281	1.214	1.193
ZeroMean	0.9862	0.9614	0.9422	1.260	1.217	1.197
FeatureOnly	1.0923	1.0914	1.0906	1.277	1.272	1.266
FilterBot	0.9821	0.9648	0.9517	1.300	1.225	1.199
MostPopular	0.9831	0.9744	0.9726	1.300	1.227	1.205
Constant Model	1.118	1.123	1.119	1.306	1.302	1.298

Experiment 4: Predicting click-rate on articles

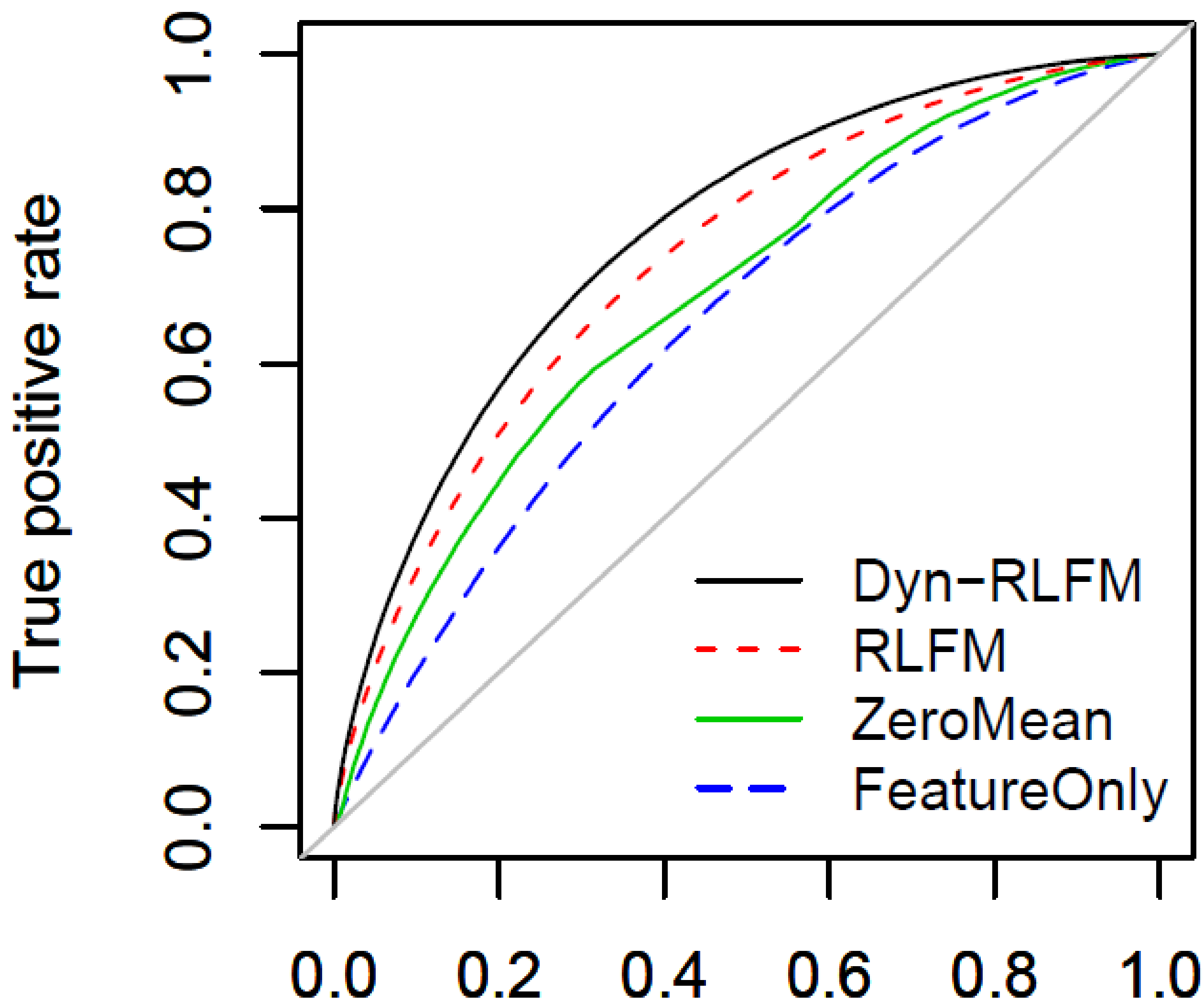
- Goal: Predict click-rate on articles for a user on F1 position
- Article lifetimes short, dynamic updates important

- User covariates:
  - Age, Gender, Geo, Browse behavior

- Article covariates
  - Content Category, keywords

- 2M ratings, 30K users, 4.5 K articles

Results on Y! FP data



Some other related approaches

- Stern, Herbrich and Graepel, WWW, 2009
  - Similar to RLFM, different parametrization and expectation propagation used to fit the models
- Porteus, Asuncion and Welling, AAAI, 2011
  - Non-parametric approach using a Dirichlet process
- Agarwal, Zhang and Mazumdar, Annals of Applied Statistics, 2011
  - Regression + random effects per user regularized through a Graphical Lasso

Add Topic Discovery into Matrix Factorization

fLDA: Matrix Factorization through Latent Dirichlet Allocation

fLDA: Introduction

- 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*

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

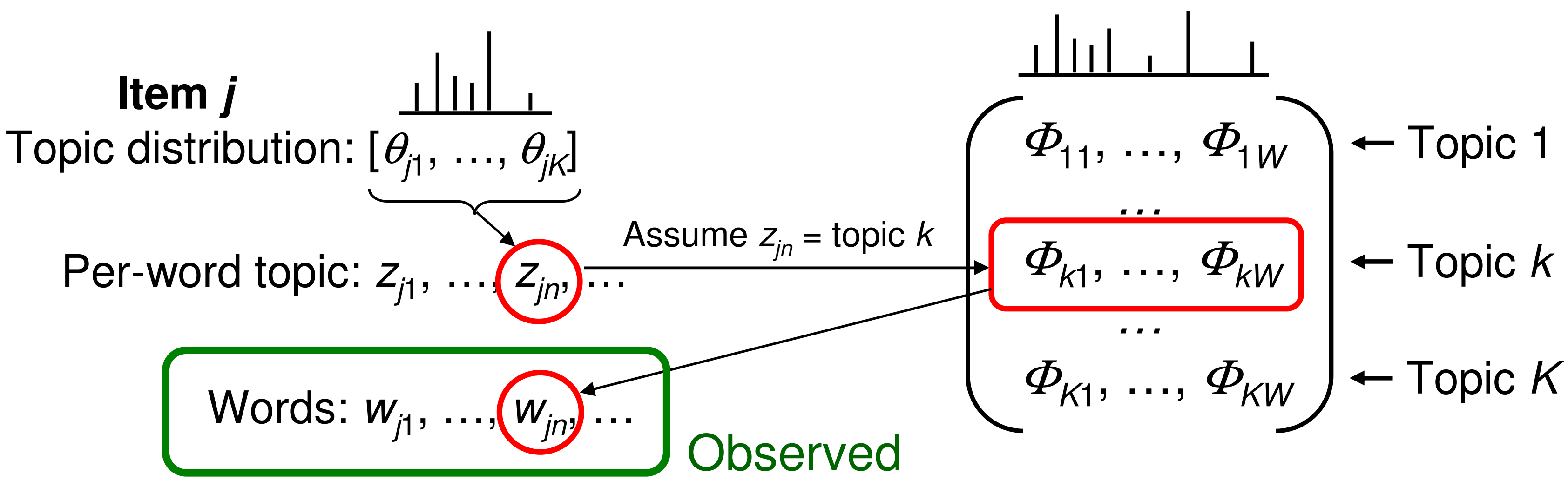
Pr(item *j* has topic *k*) estimated by averaging the LDA topic of each word in item *j*

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
- – fLDA can be thought of as a "multi-task learning" version of the supervised LDA model [Blei'07] for cold-start recommendation

LDA Topic Modeling (1)

- LDA is effective for unsupervised topic discovery [Blei'03]
  - It models the generating process of a corpus of items (articles)
  - For each topic k, draw a word distribution $\Phi_k = [\Phi_{k1}, \ldots, \Phi_{kW}] \sim \text{Dir}(\eta)$
  - For each item j, draw a topic distribution $\theta_j = [\theta_{j1}, \ldots, \theta_{jK}] \sim \text{Dir}(\lambda)$
  - For each word, say the nth word, in item j,
    - Draw a topic z_{jn} for that word from $\theta_j = [\theta_{j1}, \ldots, \theta_{jK}]$
    - Draw a word w_{jn} from $\Phi_k = [\Phi_{k1}, \ldots, \Phi_{kW}]$ with topic $k = z_{jn}$



LDA Topic Modeling (2)

- Model training:
  - Estimate the prior parameters and the posterior topic$\times$word distribution Φ based on a training corpus of items
  - EM + Gibbs sampling is a popular method
- Inference for new items
  - Compute the item topic distribution based on the prior parameters and Φ estimated in the training phase
- Supervised LDA [Blei'07]
  - Predict a target value for each item based on supervised LDA topics

Regression weight for topic *k*

One regression per user

$$\underbrace{y_j}_{\text{Target value of item } j} = \sum_k s_k \bar{z}_{jk} \qquad \text{vs.} \qquad y_{ij} = \ldots + \sum_k s_{ik} \bar{z}_{jk}$$

Target value of item *j*

Same set of topics across different regressions

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

fLDA: Model

- **Rating:** $y_{ij} \sim N(\mu_{ij}, \sigma^2)$ Gaussian Model

  user *i* gives item *j*

  $y_{ij} \sim Bernoulli(\mu_{ij})$ Logistic Model (for binary rating)

  $y_{ij} \sim Poisson(\mu_{ij} N_{ij})$ Poisson Model (for counts)

$$t(\mu_{ij}) = x_{ij}^t b + \alpha_i + \beta_j + \sum_k s_{ik} \bar{z}_{jk}$$

- **Bias of user *i*:** $\alpha_i = g_0^t x_i + \varepsilon_i^\alpha, \quad \varepsilon_i^\alpha \sim N(0, \sigma_\alpha^2)$
- **Popularity of item *j*:** $\beta_j = d_0^t x_j + \varepsilon_j^\beta, \quad \varepsilon_j^\beta \sim N(0, \sigma_\beta^2)$
- **Topic affinity of user *i*:** $s_i = H x_i + \varepsilon_i^s, \quad \varepsilon_i^s \sim N(0, \sigma_s^2 I)$
- **Pr(item *j* has topic *k*):** $\bar{z}_{jk} = \sum_n 1(z_{jn} = k) / (\#\,\text{words in item } j)$

  The LDA topic of the *n*th word in item *j*
- **Observed words:** $w_{jn} \sim LDA(\lambda, \eta, z_{jn})$

  The *n*th word in item *j*

Model Fitting

- Given:
  - Features $X = \{x_i, x_j, x_{ij}\}$
  - Observed ratings $y = \{y_{ij}\}$ and words $w = \{w_{jn}\}$
- Estimate:
  - Parameters: $\Theta = [b, g_0, d_0, H, \sigma^2, a_\alpha, a_\beta, A_s, \lambda, \eta]$
    - Regression weights and prior parameters
  - Latent factors: $\Delta = \{\alpha_i, \beta_j, s_i\}$ and $z = \{z_{jn}\}$
    - User factors, item factors and per-word topic assignment
- Empirical Bayes approach:
  - Maximum likelihood estimate of the parameters:

$$\hat{\Theta} = \arg\max_{\Theta} \Pr[y, w \mid \Theta] = \arg\max_{\Theta} \int \Pr[y, w, \Delta, z \mid \Theta] \, d\Delta dz$$

  - The posterior distribution of the factors:

$$\Pr[\Delta, z \mid y, \hat{\Theta}]$$

The EM Algorithm

- Iterate through the E and M steps until convergence
  - Let $\hat{\Theta}^{(n)}$ be the current estimate
  - E-step: Compute $$f(\Theta) = E_{(\Delta, z \mid y, w, \hat{\Theta}^n)}[\log \Pr(y, w, \Delta, z \mid \Theta)]$$
    - The expectation is not in closed form
    - We draw Gibbs samples and compute the Monte Carlo mean
  - M-step: Find $$\hat{\Theta}^{(n+1)} = \arg\max_{\Theta} f(\Theta)$$
    - It consists of solving a number of regression and optimization problems

Supervised Topic Assignment

The topic of the *n*th word in item *j*

$$\Pr(z_{jn} = k \mid \text{Rest})$$

$$\propto \underbrace{\frac{Z_{kl}^{\neg jn} + \eta}{Z_{k}^{\neg jn} + W\eta} \left(Z_{jk}^{\neg jn} + \lambda\right)}_{\text{Same as unsupervised LDA}} \cdot \underbrace{\prod_{i \text{ rated } j} \overbrace{f(y_{ij} \mid z_{jn} = k)}^{\text{Probability of observing } y_{ij} \text{ given the model}}}_{\text{Likelihood of observed ratings by users who rated item } j \text{ when } z_{jn} \text{ is set to topic } k}$$

Probability of observing y_{ij} given the model

Same as unsupervised LDA

Likelihood of observed ratings by users who rated item *j* when z_{jn} is set to topic *k*

fLDA: Experimental Results (Movie)

- 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

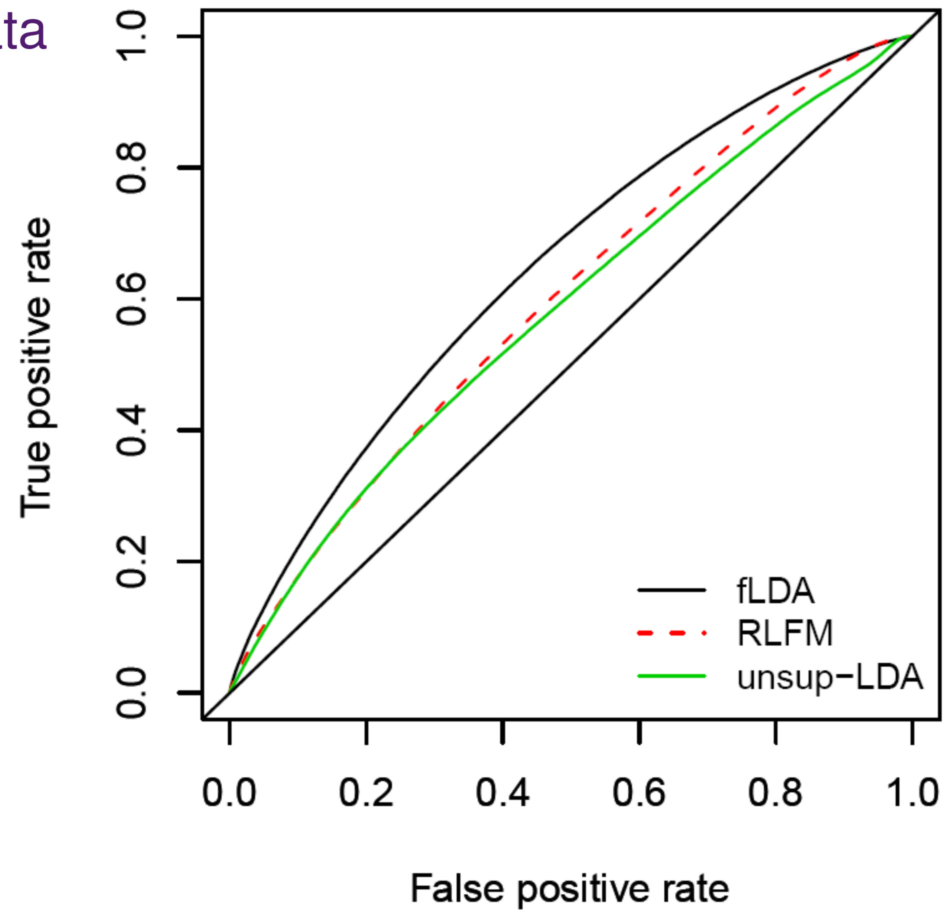
fLDA is as strong as the best method
It does not reduce the performance in warm-start scenarios

fLDA: Experimental Results (Yahoo! Buzz)

- 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



Experimental Results: Buzzing Topics

3/4 topics are interpretable; 1/2 are similar to unsupervised topics

Top Terms (after stemming)	Topic
bush, tortur, interrog, terror, administr, CIA, offici, suspect, releas, investig, georg, memo, al	CIA interrogation
mexico, flu, pirat, swine, drug, ship, somali, border, mexican, hostag, offici, somalia, captain	Swine flu
NFL, player, team, suleman, game, nadya, star, high, octuplet, nadya_suleman, michael, week	NFL games
court, gai, marriag, suprem, right, judg, rule, sex, pope, supreme_court, appeal, ban, legal, allow	Gay marriage
palin, republican, parti, obama, limbaugh, sarah, rush, gop, presid, sarah_palin, sai, gov, alaska	Sarah Palin
idol, american, night, star, look, michel, win, dress, susan, danc, judg, boyl, michelle_obama	American idol
economi, recess, job, percent, econom, bank, expect, rate, jobless, year, unemploy, month	Recession
north, korea, china, north_korea, launch, nuclear, rocket, missil, south, said, russia	North Korea issues

fLDA Summary

- fLDA is a useful model for cold-start item recommendation
- It also provides interpretable recommendations for users
  – User's preference to interpretable LDA topics
- Future directions:
  – Investigate Gibbs sampling chains and the convergence properties of the EM algorithm
  – Apply fLDA to other multi-task prediction problems
    - fLDA can be used as a tool to generate supervised features (topics) from text data

Summary

- Regularizing factors through covariates effective

- Regression based factor model that regularizes better and deals with both cold-start and warm-start in a single framework in a seamless way looks attractive

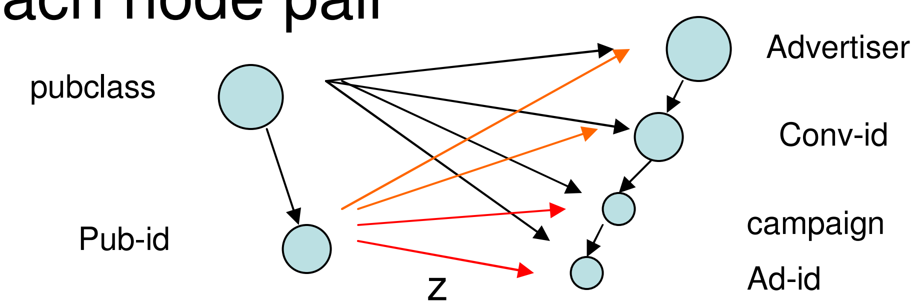
- Fitting method scalable; Gibbs sampling for users and movies can be done in parallel. Regressions in M-step can be done with any off-the-shelf scalable linear regression routine

- Distributed computing on Hadoop: Multiple models and average across partitions

Hierarchical smoothing
Advertising application, Agarwal et al. KDD 10

- Product of states for each node pair

$$\lambda_z = \prod_{s=1}^{m} \prod_{t=1}^{n} \phi_{i_s, j_t}$$



- Spike and Slab prior (S_z, E_z, λ_z)

$$\pi(\phi; a, P) = P1(\phi = 1) + (1 - P)\text{Gamma}(\phi; 1, 1/a)$$

– Known to encourage parsimonious solutions
  - Several cell states have no corrections
– Not used before for multi-hierarchy models, only in regression
– We choose P = .5 (and choose "a" by cross-validation)
  - a – psuedo number of successes