KDD’10 Tutorial: Recommender Problems for Web Applications

Deepak Agarwal and Bee-Chung Chen

Yahoo! Research
Agenda

• Focus:
  – Recommender problems for **dynamic, time-sensitive** applications
  • Content Optimization

• Introduction (20 min, Deepak)
  – Content optimization, match-making, example applications

• Offline components (40 min, Deepak)
  – Collaborative filtering (CF), methods for cold-start

• Online components + initialization (70 min, Bee-Chung)
  – Time-series, online/incremental methods, explore/exploit (bandit)

• Evaluation methods (15 min, Deepak)

• Challenges (5 min, Deepak)
Content Optimization

• Goal
  – Effectively and “pro-actively” learn from user interactions with content that are displayed to maximize our objectives.

• A new scientific discipline at the interface of
  – Large scale Machine Learning & Statistics
    • Offline Models
    • Online Models
    • Collaborative Filtering
    • Explore/Exploit
  – Multi-Objective Optimization in the presence of Uncertainty
    • Click-rates (CTR), Engagement,....
  – User Understanding
    • Profile construction
  – Content Understanding
    • Topics, “aboutness”, entities, follow-up of something, breaking news,....
Content Optimization: High level flowchart

• Flow
  – Understand content (Offline)
  – Serve content to optimize our objectives (Online)
  – quickly learn from feedback obtained using ML/Statistics (Offline + Online)
  – Constantly enhance our content inventory to improve future performance (Offline)
  – Constantly enhance our user understanding to improve future performance (Offline + Online)
  – Iterate
Some examples

- **Simple version**
  - I have an important module on my page, content inventory is obtained from a third party source which is further refined through editorial oversight. Can I algorithmically recommend content on this module? I want to drive up total CTR on this module.

- **More advanced**
  - I got X% lift in CTR. But I have additional information on other downstream utilities (e.g. dwell time). Can I increase downstream utility without losing too many clicks?

- **Highly advanced**
  - There are multiple modules running on my website. How do I take a holistic approach and perform a simultaneous optimization?
Recommend search queries
Recommend packages:
  - Image
  - Title, summary
  - Links to other pages
Pick 4 out of a pool of $K$
  $K = 20 \sim 40$
Dynamic
Routes traffic other pages
Recommend news article
Recommend applications
Problems in this example

• Optimize CTR on different modules together in a holistic way
  – Today Module, Trending Now, Personal Assistant, News, Ads
  – Treat them as independent?

• For a given module
  – Optimize some combination of CTR, downstream engagement and perhaps revenue.
Single module CTR optimization problem

- Display “best” articles for each user visit
- Best - Maximize User Satisfaction, Engagement
  - BUT Hard to obtain quick feedback to measure these

- Approximation
  - Maximize utility based on immediate feedback (click rate) subject to constraints (relevance, freshness, diversity)

- Inventory of articles?
  - Created by human editors
  - Small pool (30-50 articles) but refreshes periodically
Recommendation: A Match-making Problem

- Recommendation problems
- Search: Web, Vertical
- Online advertising
- ...

**Item Inventory**
- Articles, web page, ads, ...

**Opportunity**
- Users, queries, pages, ...

Use an automated algorithm to select item(s) to show

Get feedback (click, time spent,..)
Refine the models

Repeat (large number of times)
Measure metric(s) of interest
(Total clicks, Total revenue,..)
Important Factors affecting solution in Match-making Problems

- **Items**: Articles, web pages, ads, modules, queries, users, updates, etc.

- **Opportunities**: Users, query keywords, pages, etc.

- **Metric** (e.g., editorial score, CTR, revenue, engagement)
  - Currently, most applications are single-objective
  - May be multi-objective optimization (maximize $X$ subject to $Y$, $Z$,..)

- **Properties of the item pool**
  - Size (e.g., all web pages vs. 40 stories)
  - Quality of the pool (e.g., anything vs. editorially selected)
  - Lifetime (e.g., mostly old items vs. mostly new items)
Factors affecting Solution continued

• **Properties of the opportunities**
  – **Pull**: Specified by explicit, user-driven query (e.g., keywords, a form)
  – **Push**: Specified by implicit context (e.g., a page, a user, a session)
  – **Size** (e.g., user base); **continuity** (e.g., session vs. single event)

• **Properties of the feedback on the matches made**
  – **Types and semantics** of feedback (e.g., click, vote)
  – **Latency** (e.g., available in 5 minutes vs. 1 day)
  – **Volume** (e.g., 100K per day vs. 300M per day)

• **Constraints specifying legitimate matches** (e.g., business rules)

• **Available Metadata** (e.g., link graph, various user/item attributes)
## Recommendation vs. Other Match-Making Problems

<table>
<thead>
<tr>
<th></th>
<th>Recommendation</th>
<th>Search</th>
<th>Advertising</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Metric</strong></td>
<td>User engagement</td>
<td>Relevance to the query</td>
<td>Revenue</td>
</tr>
<tr>
<td><strong>Items</strong></td>
<td>Anything (except for ads)</td>
<td>Anything (except for ads)</td>
<td>Ads</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Opportunities</strong></td>
<td>Push (implicit)</td>
<td>Pull (explicit)</td>
<td>Push</td>
</tr>
<tr>
<td></td>
<td>The system guesses users info needs</td>
<td>Users specify their info needs</td>
<td></td>
</tr>
<tr>
<td><strong>Examples</strong></td>
<td>Recommend articles, friends, feeds to users</td>
<td>Web search</td>
<td>Sponsored search</td>
</tr>
<tr>
<td></td>
<td>Recommend related items given an item</td>
<td>Vertical search</td>
<td>Content match</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Behavior targeting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Display advertising (non-guaranteed)</td>
</tr>
</tbody>
</table>
More on Recommendation vs. Search

• Recommendation
  – User intent: See something “interesting” (browse mode, implicit)
    • The system tries to guess what a user likes on an entity/topic page
  – No query reformulation (unless we suggest related topics/entities)
  – False +ve more costly than false –ve
    • Showing a bad article is a worse than missing a good one

• Search
  – User intent: Explicit, users express what they want
  – Users can reformulate queries
  – False –ve more costly (but depends on the query)
    • Users want to get the results they are looking for
Modeling: Key components

Offline
(Logistic, GBDT,..)

Feature construction
Content: IR, clustering, taxonomy, entity,..
User profiles: clicks, views, social, community,..

Initialize

Explore/Exploit
(Adaptive sampling)

Online
(Fine resolution Corrections)
(item, user level)
(Quick updates)
Modeling Problems that has received attention

- Univariate response (e.g. click); single objective (e.g. maximize CTR)

- Our solution
  - Initialize online through offline models
  - Learn “corrections” to offline models at very granular levels (user, item) and learn rapidly in an online fashion
  - Online correction models have reduced dimension through clever representations of parameters and by exploiting the fallback mechanism to coarser models
  - The models are tightly coupled with Explore-exploit to ensure fast convergence to areas of high valued response
Example Application:
Today Module on Yahoo! Homepage

Currently in production powered by some methods discussed in this tutorial
17 Deepak Agarwal & Bee-Chung Chen @ KDD’10

Recommend packages:
- Image
- Title, summary
- Links to other pages

Pick 4 out of a pool of $K$
- $K = 20 \sim 40$
- Dynamic
- Routes traffic other pages
Problem definition

• Display “best” articles for each user visit
• Best - Maximize User Satisfaction, Engagement
  – BUT Hard to obtain quick feedback to measure these

• Approximation
  – Maximize utility based on immediate feedback (click rate) subject to constraints (relevance, freshness, diversity)

• Inventory of articles?
  – Created by human editors
  – Small pool (30-50 articles) but refreshes periodically
Where are we today?

- Before this research
  - Articles created and selected for display by editors

- After this research
  - Article placement done through statistical models

- How successful?

  "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)"
Main Goals

• Methods to select most popular articles
  – This was done by editors before

• Provide personalized article selection
  – Based on user covariates
  – Based on per user behavior

• Scalability: Methods to generalize in small traffic scenarios
  – Today module part of most Y! portals around the world
  – Also syndicated to sources like Y! Mail, Y! IM etc
Similar applications

- Goal: Use same methods for selecting most popular, personalization across different applications at Y!
- Good news! Methods generalize, already in use
Next few hours

<table>
<thead>
<tr>
<th></th>
<th>Most Popular Recommendation</th>
<th>Personalized Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Offline Models</strong></td>
<td></td>
<td>Collaborative filtering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(cold-start problem)</td>
</tr>
<tr>
<td><strong>Online Models</strong></td>
<td>Time-series models</td>
<td>Incremental CF,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>online regression</td>
</tr>
<tr>
<td><strong>Intelligent Initialization</strong></td>
<td>Prior estimation</td>
<td>Prior estimation,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dimension reduction</td>
</tr>
<tr>
<td><strong>Explore/Exploit</strong></td>
<td>Multi-armed bandits</td>
<td>Bandits with covariates</td>
</tr>
</tbody>
</table>
Offline Components:
Collaborative Filtering in Cold-start Situations
Problem Definition

Algorithm selects Item $j$ with item features $x_j$
(keywords, content categories, ...)

User $i$ visits with user features $x_i$
(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
Model Choices

- Feature-based (or content-based) approach
  - Use features to predict response (regression, Bayes Net, mixture models, …)
  - Bottleneck: need predictive features
    - 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

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

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 feature-based and CF models
    - Learn weights adaptively at user level
  - Filterbot
    - Add user features as pseudo users and do collaborative filtering

- **Hybrid approaches**
  - Use content based to fill up entries, then use CF

- **Model-based approaches**
  - Mixed kernel learnt jointly
    - Popularity, features, user-user similarities, item-item similarities
  - Bayesian mixed-effects models
    - Given modeling assumptions are reasonable: state-of-the-art

- **Drilldown**
  - Matrix factorization
    - Superior than others on Netflix data [Koren, 2009], also on our Yahoo! data
    - Add feature-based regression to matrix factorization
    - Add topic discovery (from textual items) to matrix factorization
Per-user, per-item models via bilinear random-effects model

Matrix Factorization
Motivation

• Data measuring k-way interactions pervasive
  – Consider k = 2 for all our discussions
  – E.g. User-Movie, User-content, User-Publisher-Ads,….

• 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, overfit very easily
  – Key issue
    • Putting constraints on the eigenvectors (factors) to avoid overfitting
Early work in the literature

• 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 web datasets
  – Highly incomplete, large, noisy.
Factorization – Brief Overview

• Latent user factors: \((\alpha_i, \mathbf{u}_i=(u_{i1}, \ldots, u_{in}))\)
• Latent movie factors: \((\beta_j, \mathbf{v}_j=(v_{j1}, \ldots, v_{jn}))\)

\[ E(y_{ij}) = \mu + \alpha_i + \beta_j + u'_i B v_j \]

• (Nn + Mm) parameters
• Will overfit for moderate values of n,m

• Key technical issue: \(Regularization\)
Model: Different choices of factors

- Bi-Clustering
  - Hard, Soft

- Matrix Factorization
  - Factors in Euclidean space
  - Regularization

- Incorporating features
- Online updates
BICLUSTERING: Iterative row and column k-means

RAW DATA

After ROW CLUSTERING

Smooth Rows using column clusters; reduce variance

After COLUMN Clustering

Iterate Until convergence
Bi-Clustering can be represented as factorization

- m user clusters, n item clusters

\[ u_i, v_j: \text{Cluster membership weights.} \]
\[ 1^\prime u_i = 1^\prime v_j = 1 \]

- \( B \): bi-cluster means
- Hard-clustering
  - Each row(col) belongs to exactly one cluster
- Soft-clustering
  - Weighted assignment to several clusters
Factors in Euclidean space

- Latent user factors: \((\alpha_i, \mathbf{u}_i=(u_{i1},\ldots,u_{ir}))\)
- Latent movie factors: \((\beta_j, \mathbf{v}_j=(v_{j1},\ldots,v_{jr}))\)

\[
\alpha_i + \beta_j + u_i^\prime v_j
\]

- \((N + M)(r+1)\) parameters
- will overfit for moderate values of \(r\)

- Key technical issue: \(\text{Regularization}\)
- Usual approach: Gaussian ZeroMean prior
Existing Zero-Mean Factorization Model

Observation Equation
\[ y_{ij} \sim N(m_{ij}, \sigma^2) \]
\[ x'_{ij} b + \alpha_i + \beta_j + u'_i v_j \]

State Equation
\[ \alpha_i \sim N(0, a_\alpha) \]
\[ \beta_j \sim N(0, a_\beta) \]
\[ u_i \sim MVN(0, A_u) \]
\[ v_j \sim MVN(0, A_v) \]

Predict for new cell:
\[ (x_{ij}^{new})' \hat{b} + \hat{\alpha}_i + \hat{\beta}_j + \hat{u}'_i \hat{v}_j \]
PROBLEM DEFINITION

• Models to predict ratings for new pairs
  – Warm-start: (user, movie) present in the training data
  – Cold-start: At least one of (user, movie) new

• 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
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
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 $j$
  - $y_{ij} \sim N(\mu_{ij}, \sigma^2)$  \hspace{1cm} \text{Gaussian Model}
  - $y_{ij} \sim \text{Bernoulli}(\mu_{ij})$  \hspace{1cm} \text{Logistic Model (for binary rating)}
  - $y_{ij} \sim \text{Poisson}(\mu_{ij} N_{ij})$  \hspace{1cm} \text{Poisson Model (for counts)}

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

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

Could use other classes of regression models
Advantages of RLFM

- Better regularization of factors
  - Covariates “shrink” towards a better centroid

- Cold-start: Fallback regression model (FeatureOnly)

\[ y_{ij} \sim N(m_{ij}, \sigma^2) \]
\[ m_{ij} = x'_{ij} b + g'_{0} w_{i} + d'_{0} z_{j} + w'_{i} G' D z_{j} \]
Graphical representation of the model

Rating \((i, j)\)

\[ x_{ij} \rightarrow y_{ij} \rightarrow b \]

User \(i\)

- \(\alpha_i\)
- \(u_i\)
- \(w_i\)
- \(G\)
- \(d_0\)

Item \(j\)

- \(\beta_j\)
- \(v_j\)
- \(z_j\)
- \(D\)
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

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

\[x'_{ij} b + \alpha_i + \beta_j + u'_{i} v_j\]

\[
\alpha_i = g'_0 w_i + \epsilon^\alpha_i, \quad \epsilon^\alpha_i \sim N(0, a_\alpha)
\]

\[
\beta_j = d'_0 z_j + \epsilon^\beta_j, \quad \epsilon^\beta_j \sim N(0, a_\beta)
\]

\[
u_i = G w_i + \epsilon^u_i, \quad \epsilon^u_i \sim MVN(0, A_u)
\]

\[
u_j = D z_j + \epsilon^\nu_j, \quad \epsilon^\nu_j \sim MVN(0, A_v)
\]
Closer look at induced marginal correlations

\[ E(y_{ij}) = x_{ij}' b + g_0' w_i + d_0' z_j + w_i' G' D z_j \]

\[ V ar(y_{ij}) = \sigma^2 + a_\alpha + a_\beta + tr(A_u A_v) + z_j' D' A_u D z_j + w_i' G' A_v G w_i \]

\[ cov(y_{ij}, y_{ij}^*) = a_\alpha + z_j' D' A_u D z_j^* \]

\[ cov(y_{ij}, y_{i^*j}) = a_\beta + w_i' G' A_v G w_i^* \]
Overview: EM for our class of models

\[ Y : \text{Data} \]

\[ \Delta : \text{Latent variables} \]

\[ \Theta : \text{hyper-parameters} \]

Model: \[ p(Y | \Delta, \Theta)p(\Delta | \Theta) \]

Output needed: Mode: \[ \max_{\Theta} p(\Theta | Y) \]

\[ p(\Delta | Y) \approx p(\Delta | Y, \hat{\Theta}) \]
The parameters for RLFM

- Latent parameters

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

- Hyper-parameters

\[ \Theta = (b, G, D, A_u = a_u I, A_v = a_v I) \]
Computing the mode

\[
\log(p(\Theta|Y)) = \log(p(\Theta, \Delta|Y)) - \log(p(\Delta|\Theta, Y))
\]

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

\[E_{old} : \text{Expectation w.r.t. } p(\Delta|\Theta_{old}, Y)\]

Second term: Maximized at \(\Theta_{old}\)
Find new value of \(\Theta\) that increases first term
The EM algorithm

Initialize $\Theta$

Iterate

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

M-step : $\arg\max_{\Theta} E_{old}(\log(p(\Theta, \Delta|Y)))$
Computing the E-step

- Often hard to compute in closed form
- Stochastic EM (Markov Chain EM; MCEM)
  - Compute expectation by drawing samples from
    \[ p(\Delta | \Theta_{old}, Y) \]
  - Effective for multi-modal posteriors but more expensive
- Iterated Conditional Modes algorithm (ICM)
  - Faster but biased hyper-parameter estimates

\[
\text{Approximate } E_{old}(\log(p(\Theta, \Delta | Y))) \text{ by } \log(p(\Theta_{old}, \hat{\Delta} | Y))
\]
\[
\hat{\Delta} = \arg \max_{\Delta} \Delta \log(p(\Theta_{old}, \Delta | Y))
\]
Model Fitting

- Challenging, multi-modal posterior
- Monte-Carlo EM (MCEM)
  - E-step: Sample factors through Gibbs sampling
  - M-step: Estimate regressions through off-the-shelf linear regression routines using sampled factors as response
    - We used t-regression, others like LASSO could be used
- Iterated Conditional Mode (ICM)
  - Replace E-step by CG: conditional modes of factors
  - M-step: Estimate regressions using the modes as response
- Incorporating uncertainty in factor estimates in MCEM helps

<table>
<thead>
<tr>
<th>Latent dimension</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICM</td>
<td>.9736</td>
<td>.9729</td>
<td>.9799</td>
<td>.9802</td>
</tr>
<tr>
<td>MCEM</td>
<td>.9728</td>
<td>.9722</td>
<td>.9714</td>
<td>.9715</td>
</tr>
</tbody>
</table>
Monte Carlo E-step

- Through a vanilla Gibbs sampler (conditionals closed form)

\[
L \text{et } o_{ij} = y_{ij} - \alpha_i - \beta_j - x'_{ij} b \\
\text{Var}[u_i | \text{Rest}] = (A_u^{-1} + \sum_{j \in \mathcal{J}_i} \frac{v_j v'_j}{\sigma_{i,j}^2})^{-1} \\
E[u_i | \text{Rest}] = \text{Var}[u_i | \text{Rest}](A_u^{-1} \tilde{G} w_i + \sum_{j \in \mathcal{J}_i} \frac{o_{ij} v_j}{\sigma_{i,j}^2})
\]

- 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 $A_u = a_u I$

\[
\hat{a}_u = \frac{\sum_{i=1}^{M} (E^* (u_i) - \hat{G}w_i)' (E^* (u_i) - \hat{G}w_i) + \sum_{k=1}^{r} Vari (u_{ikl})}{Mr}
\]

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

<table>
<thead>
<tr>
<th></th>
<th>RLFM</th>
<th>ZeroMean</th>
<th>FeatureOnly</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens-100K</td>
<td>0.8956</td>
<td>0.9064</td>
<td>1.0968</td>
</tr>
</tbody>
</table>
Experiment 2: Better handling of Cold-start

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

<table>
<thead>
<tr>
<th>Model</th>
<th>MovieLens-1M</th>
<th></th>
<th></th>
<th>EachMovie</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30%</td>
<td>60%</td>
<td>75%</td>
<td></td>
<td>30%</td>
<td>60%</td>
</tr>
<tr>
<td>RLFM</td>
<td>0.9742</td>
<td>0.9528</td>
<td>0.9363</td>
<td>1.281</td>
<td>1.214</td>
<td>1.193</td>
</tr>
<tr>
<td>ZeroMean</td>
<td>0.9862</td>
<td>0.9614</td>
<td>0.9422</td>
<td>1.260</td>
<td>1.217</td>
<td>1.197</td>
</tr>
<tr>
<td>FeatureOnly</td>
<td>1.0923</td>
<td>1.0914</td>
<td>1.0906</td>
<td>1.277</td>
<td>1.272</td>
<td>1.266</td>
</tr>
<tr>
<td>FilterBot</td>
<td>0.9821</td>
<td>0.9648</td>
<td>0.9517</td>
<td>1.300</td>
<td>1.225</td>
<td>1.199</td>
</tr>
<tr>
<td>MostPopular</td>
<td>0.9831</td>
<td>0.9744</td>
<td>0.9726</td>
<td>1.300</td>
<td>1.227</td>
<td>1.205</td>
</tr>
<tr>
<td>Constant Model</td>
<td>1.118</td>
<td>1.123</td>
<td>1.119</td>
<td>1.306</td>
<td>1.302</td>
<td>1.298</td>
</tr>
</tbody>
</table>
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
Another Interesting Regularization on the factors

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

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

\[ (u_1, \cdots, u_N) \sim MVN \left( 0, (D - \rho W) \otimes I \right) \]
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

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

User $i$’s affinity to topic $k$

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 $n$th 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 $k = z_{jn}$

Item $j$

Topic distribution: $[\theta_{j1}, \ldots, \theta_{jk}]$

Per-word topic: $z_{j1}, \ldots, z_{jm}, \ldots$

Assume $z_{jn} = \text{topic } k$

Words: $w_{j1}, \ldots, w_{jn}, \ldots$

Observed

$\Phi_{11}, \ldots, \Phi_{1W}$

$\Phi_{k1}, \ldots, \Phi_{kW}$

$\Phi_{K1}, \ldots, \Phi_{KW}$

Topic 1

Topic $k$

Topic $K$
LDA Topic Modeling (2)

- **Model training:**
  - Estimate the prior parameters and the posterior topic×word distribution $\Phi$ 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 $\Phi$ estimated in the training phase

- **Supervised LDA [Blei’07]**
  - Predict a target value for each item based on supervised LDA topics

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

  - Regression weight for topic $k$
  - One regression per user
  - Same set of topics across different regressions

  Target value of item $j$

  Pr(item $j$ has topic $k$) estimated by averaging the topic of each word in item $j$
fLDA: Model

- **Rating:**
  \[ y_{ij} \sim N(\mu_{ij}, \sigma^2) \]
  \[ y_{ij} \sim \text{Bernoulli}(\mu_{ij}) \]
  \[ y_{ij} \sim \text{Poisson}(\mu_{ij} N_{ij}) \]
  Gaussian Model
  Logistic Model (for binary rating)
  Poisson Model (for counts)

- **Bias of user i:**
  \[ \alpha_i = g^t x_i + \varepsilon_i^\alpha, \quad \varepsilon_i^\alpha \sim N(0, \sigma^2_\alpha) \]

- **Popularity of item j:**
  \[ \beta_j = d^t_0 x_j + \varepsilon_j^\beta, \quad \varepsilon_j^\beta \sim N(0, \sigma^2_\beta) \]

- **Topic affinity of user i:**
  \[ s_i = H x_i + \varepsilon_i^s, \quad \varepsilon_i^s \sim N(0, \sigma^2_s I) \]

- **Pr(item j has topic k):**
  \[ z_{jk} = \sum_n 1(z_{jn} = k) / (# \text{words in item j}) \]
  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 $\Theta^{(n)}$ be the current estimate
  – E-step: Compute $f(\Theta) = E_{(\Delta, z | y, w, \Theta^n)} [\log \Pr(y, w, \Delta, z | \Theta)]$
    • The expectation is not in closed form
    • We draw Gibbs samples and compute the Monte Carlo mean
  – M-step: Find $\Theta^{(n+1)} = \arg \max_{\Theta} f(\Theta)$
    • It consists of solving a number of regression and optimization problems
Supervised Topic Assignment

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

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

$$\propto \frac{Z_{kl}^{jn} + \eta}{Z_k^{jn} + W\eta} (Z_{jk}^{jn} + \lambda) \cdot \prod_{i \text{ rated}} f(y_{ij} \mid z_{jn} = k)$$

Same as unsupervised LDA

Probability of observing $y_{ij}$ given the model

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Test RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLFM</td>
<td>0.9363</td>
</tr>
<tr>
<td>fLDA</td>
<td>0.9381</td>
</tr>
<tr>
<td>Factor-Only</td>
<td>0.9422</td>
</tr>
<tr>
<td>FilterBot</td>
<td>0.9517</td>
</tr>
<tr>
<td>unsup-LDA</td>
<td>0.9520</td>
</tr>
<tr>
<td>MostPopular</td>
<td>0.9726</td>
</tr>
<tr>
<td>Feature-Only</td>
<td>1.0906</td>
</tr>
<tr>
<td>Constant</td>
<td>1.1190</td>
</tr>
</tbody>
</table>

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

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

• We presented a regression based factor model that regularizes better and deals with both cold-start and warm-start in a single framework in a seamless way

• 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
Online Components:
Online Models, Intelligent Initialization,
Explore / Exploit
Why Online Components?

• Cold start
  – New items or new users come to the system
  – How to obtain data for new items/users (explore/exploit)
  – Once data becomes available, how to quickly update the model
    • Periodic rebuild (e.g., daily): Expensive
    • Continuous online update (e.g., every minute): Cheap

• Concept drift
  – Item popularity, user interest, mood, and user-to-item affinity may change over time
  – How to track the most recent behavior
    • Down-weight old data
  – How to model temporal patterns for better prediction
## Big Picture

<table>
<thead>
<tr>
<th>Offline Models</th>
<th>Most Popular Recommendation</th>
<th>Personalized Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Collaborative filtering (cold-start problem)</td>
</tr>
<tr>
<td><strong>Online Models</strong></td>
<td></td>
<td>Incremental CF, online regression</td>
</tr>
<tr>
<td>Real systems are dynamic</td>
<td>Time-series models</td>
<td>Incremental CF, online regression</td>
</tr>
<tr>
<td><strong>Intelligent Initialization</strong></td>
<td>Prior estimation</td>
<td>Prior estimation, dimension reduction</td>
</tr>
<tr>
<td>Do not start cold</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Explore/Exploit</strong></td>
<td>Multi-armed bandits</td>
<td>Bandits with covariates</td>
</tr>
<tr>
<td>Actively acquire data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Extension: **Segmented Most Popular Recommendation**
Online Components for
Most Popular Recommendation

Online models, intelligent initialization & explore/exploit
Most popular recommendation: Outline

• Most popular recommendation (no personalization, all users see the same thing)
  – Time-series models (online models)
  – Prior estimation (initialization)
  – Multi-armed bandits (explore/exploit)

• Segmented most popular recommendation
  – Create user segments/clusters based on user features
  – Provide most popular recommendation for each segment
Most Popular Recommendation

• Problem definition: Pick \( k \) items (articles) from a pool of \( n \) to maximize the total number of clicks on the picked items
• Easy! Pick the items having the highest click-through rates (CTRs)
• But …
  – The system is highly dynamic:
    • Items come and go with short lifetimes
    • CTR of each item changes over time
  – How much traffic should be allocated to explore new items to achieve optimal performance
    • Too little → Unreliable CTR estimates
    • Too much → Little traffic to exploit the high CTR items
CTR Curves for Two Days on Yahoo! Front Page

Each curve is the CTR of an item in the Today Module on www.yahoo.com over time

Traffic obtained from a controlled randomized experiment (no confounding)

Things to note:
(a) Short lifetimes, (b) temporal effects, (c) often breaking news stories
For Simplicity, Assume ...

- Pick only **one** item for each user visit
  - Multi-slot optimization later
- No user segmentation, no personalization (discussion later)
- The pool of candidate items is **predetermined** and is relatively small ($\leq 1000$)
  - E.g., selected by human editors or by a first-phase filtering method
  - Ideally, there should be a feedback loop
  - Large item pool problem later
- Effects like user-fatigue, diversity in recommendations, multi-objective optimization not considered (discussion later)
Online Models

• How to track the changing CTR of an item
• For a given item, at time $t$, we
  – Show the item $n_t$ times (i.e., $n_t$ views)
  – Receive $c_t$ clicks
• Problem Definition: Given $c_1$, $n_1$, …, $c_t$, $n_t$, predict the CTR (click-through rate) $p_{t+1}$ at time $t+1$
• Potential solutions:
  – Instant CTR: $c_t / n_t \rightarrow$ highly unstable ($n_t$ is usually small)
  – Cumulative CTR: $(\sum_{i} c_i) / (\sum_{i} n_i) \rightarrow$ react to changes very slowly
  – Moving window CTR: $(\sum_{i=\text{last} K} c_i) / (\sum_{i=\text{last} K} n_i) \rightarrow$ reasonable
    • But, no estimation of $\text{Var}[p_{t+1}]$ (useful for explore/exploit)
Online Models: Dynamic Gamma-Poisson

- Model-based approach
  - \( (c_t \mid n_t, p_t) \sim \text{Poisson}(n_t p_t) \)
  - \( p_t = p_{t-1} \varepsilon_t \), where \( \varepsilon_t \sim \text{Gamma} \text{mean}=1, \text{var}=\eta \)
  - Model parameters:
    - \( p_1 \sim \text{Gamma} \text{mean} = \mu_0, \text{var} = \sigma_0^2 \) is the offline CTR estimate
    - \( \eta \) specifies how dynamic/smooth the CTR is over time
  - Posterior distribution \( (p_{t+1} / c_1, n_1, \ldots, c_t, n_t) \sim \text{Gamma}(?,?) \)
    - Solve this recursively (online update rule)

- Show the item \( n_t \) times
- Receive \( c_t \) clicks
- \( p_t = \text{CTR at time } t \)
Online Models: Derivation

Estimated CTR distribution at time $t$
\[
(p_t \mid c_1, n_1, \ldots, c_{t-1}, n_{t-1}) \sim \text{Gamma}(\text{mean} = \mu_t, \text{var} = \sigma_t^2)
\]
Let $\gamma_t = \mu_t / \sigma_t^2$ (effective sample size)
\[
(p_t \mid c_1, n_1, \ldots, c_t, n_t) \sim \text{Gamma}(\text{mean} = \mu_{t\mid t}, \text{var} = \sigma_{t\mid t}^2)
\]
Let $\gamma_{t\mid t} = \gamma_t + n_t$ (effective sample size)
\[
\mu_{t\mid t} = (\mu_t \cdot \gamma_t + c_t) / \gamma_{t\mid t}
\]
\[
\sigma_{t\mid t}^2 = \mu_{t\mid t} / \gamma_{t\mid t}
\]

Estimated CTR distribution at time $t+1$
\[
(p_{t+1} \mid c_1, n_1, \ldots, c_t, n_t) \sim \text{Gamma}(\text{mean} = \mu_{t+1}, \text{var} = \sigma_{t+1}^2)
\]
\[
\mu_{t+1} = \mu_{t\mid t}
\]
\[
\sigma_{t+1}^2 = \sigma_{t\mid t}^2 + \eta(\mu_{t\mid t}^2 + \sigma_{t\mid t}^2)
\]
High CTR items more adaptive
Tracking behavior of Gamma-Poisson model

- Low click rate articles – More temporal smoothing
Intelligent Initialization: Prior Estimation

- Prior CTR distribution: Gamma(mean=$\mu_0$, var=$\sigma_0^2$)
  - $N$ historical items:
    - $n_i$ = #views of item $i$ in its first time interval
    - $c_i$ = #clicks on item $i$ in its first time interval
  - Model
    - $c_i \sim$ Poisson($n_i \rho_i$) and $\rho_i \sim$ Gamma($\mu_0$, $\sigma_0^2$)
      $\Rightarrow$ $c_i \sim$ NegBinomial($\mu_0$, $\sigma_0^2$, $n_i$)
  - Maximum likelihood estimate (MLE) of ($\mu_0$, $\sigma_0^2$)

$$\arg\max_{\mu_0, \sigma_0^2} N \frac{\mu_0^2}{\sigma_0^2} \log \frac{\mu_0}{\sigma_0} - N \log \Gamma\left(\frac{\mu_0^2}{\sigma_0^2}\right) + \sum_i \log \Gamma\left(c_i + \frac{\mu_0^2}{\sigma_0^2}\right) - \left(c_i + \frac{\mu_0^2}{\sigma_0^2}\right) \log \left(n_i + \frac{\mu_0}{\sigma_0^2}\right)$$

- Better prior: Cluster items and find MLE for each cluster
Explore/Exploit: Problem Definition

Determine \((x_1, x_2, \ldots, x_K)\) based on clicks and views observed before \(t\) in order to maximize the expected total number of clicks in the future.
Modeling the Uncertainty, NOT just the Mean

Simplified setting: Two items

If we only make a **single** decision, give 100% page views to **Item A**

If we make **multiple** decisions in the future, explore **Item B** since its CTR can potentially be higher

\[
\text{Potential} = \int_{p>q} (p - q) \cdot f(p) \, dp
\]

- CTR of **item A** is \( q \)
- CTR of **item B** is \( p \)
- Probability density function of **item B**’s CTR is \( f(p) \)

We know the CTR of **Item A** (say, shown 1 million times)
We are uncertain about the CTR of **Item B** (only 100 times)
Multi-Armed Bandits: Introduction (1)

For now, we are attacking the problem of choosing best article/arm for all users

Bandit “arms”
(unknown payoff probabilities)

- “Pulling” arm i yields a reward:
  - reward = 1 with probability \( p_i \) (success)
  - reward = 0 otherwise (failure)
Multi-Armed Bandits: Introduction (2)

- Goal: Pull arms sequentially to maximize the total reward
- Bandit scheme/policy: Sequential algorithm to play arms (items)
- Regret of a scheme = Expected loss relative to the "oracle" optimal scheme that always plays the best arm
  - "Best" means highest success probability
  - But, the best arm is not known ... unless you have an oracle
  - Hence, the regret is the price of exploration
  - Low regret implies quick convergence to the best
Multi-Armed Bandits: Introduction (3)

- **Bayesian approach**
  - Seeks to find the Bayes optimal solution to a Markov decision process (MDP) with assumptions about probability distributions
  - Representative work: Gittins’ index, Whittle’s index
  - Very computationally intensive

- **Minimax approach**
  - Seeks to find a scheme that incurs bounded regret (with no or mild assumptions about probability distributions)
  - Representative work: UCB by Lai, Auer
  - Usually, computationally easy
  - But, they tend to explore too much in practice (probably because the bounds are based on worse-case analysis)
• Select an arm now at time \( t=0 \), to maximize expected total number of clicks in \( t=0, \ldots, T \)

• State at time \( t \): \( \Theta_t = (\theta_1, \ldots, \theta_{Kt}) \)
  - \( \theta_{it} \) = State of arm \( i \) at time \( t \) (that captures all we know about arm \( i \) at \( t \))

• Reward function \( R_i(\Theta_t, \Theta_{t+1}) \)
  - Reward of pulling arm \( i \) that brings the state from \( \Theta_t \) to \( \Theta_{t+1} \)

• Transition probability \( \Pr [\Theta_{t+1} | \Theta_t, \text{pulling arm } i] \)

• Policy \( \pi \): A function that maps a state to an arm (action)
  - \( \pi(\Theta_i) \) returns an arm (to pull)

• Value of policy \( \pi \) starting from the current state \( \Theta_0 \) with horizon \( T \)

\[
V_T (\pi, \Theta_0) = E \left[ R_{\pi(\Theta_0)} (\Theta_0, \Theta_1) + V_{T-1} (\pi, \Theta_1) \right] \\
= \int \Pr [\Theta_1 | \Theta_0, \pi(\Theta_0)] \cdot \left[ R_{\pi(\Theta_0)} (\Theta_0, \Theta_1) + V_{T-1} (\pi, \Theta_1) \right] d\Theta_1
\]
Multi-Armed Bandits: MDP (2)

\[ V_T(\pi, \Theta_0) = E[R_{\pi(\Theta_0)}(\Theta_0, \Theta_1) + V_{T-1}(\pi, \Theta_1)] \]

\[ = \int \Pr[\Theta_1 | \Theta_0, \pi(\Theta_0)] \left[ R_{\pi(\Theta_0)}(\Theta_0, \Theta_1) + V_{T-1}(\pi, \Theta_1) \right] d\Theta_1 \]

- Optimal policy: \( \arg \max_{\pi} V_T(\pi, \Theta_0) \)

- Things to notice:
  - Value is defined recursively (actually \( T \) high-dim integrals)
  - Dynamic programming can be used to find the optimal policy
  - But, just evaluating the value of a fixed policy can be very expensive

- Bandit Problem: The pull of one arm does not change the state of other arms and the set of arms do not change over time
Multi-Armed Bandits: MDP (3)

- Which arm should be pulled next?
  - Not necessarily what looks best right now, since it might have had a few lucky successes
  - Looks like it will be a function of successes and failures of all arms

- Consider a slightly different problem setting
  - Infinite time horizon, but
  - Future rewards are geometrically discounted
    \[ R_{\text{total}} = R(0) + \gamma R(1) + \gamma^2 R(2) + \ldots \quad (0 < \gamma < 1) \]

- Theorem [Gittins 1979]: The optimal policy decouples and solves a bandit problem for each arm independently

  \[ \text{Policy } \pi(\Theta_i) \text{ is a function of } (\theta_{1t}, \ldots, \theta_{Kt}) \]

\[ \text{One } K\text{-dimensional problem} \]

\[ \text{Policy } \pi(\Theta_i) = \arg\max_i \{ g(\theta_i) \} \]

\[ K \text{ one-dimensional problems} \]

Still computationally expensive!!
Multi-Armed Bandits: MDP (4)

**Bandit Policy**

1. Compute the priority (Gittins’ index) of each arm based on its state
2. Pull arm with max priority, and observe reward
3. Update the state of the pulled arm
Multi-Armed Bandits: MDP (5)

- Theorem [Gittins 1979]: The optimal policy decouples and solves a bandit problem for each arm independently
  - Many proofs and different interpretations of Gittins’ index exist
    - The index of an arm is the fixed charge per pull for a game with two options, **pull the arm or not**, so that the charge makes the optimal play of the game have zero net reward
  - Significantly reduces the dimension of the problem space
  - But, Gittins’ index $g(\theta_{it})$ is still hard to compute
    - For the Gamma-Poisson or Beta-Binomial models
      - $\theta_{it} = (\#\text{successes}, \#\text{pulls})$ for arm $i$ up to time $t$
      - $g$ maps each possible $(\#\text{successes}, \#\text{pulls})$ pair to a number
  - Approximate methods are used in practice
  - Lai et al. have derived these for exponential family distributions
Multi-Armed Bandits: Minimax Approach (1)

- Compute the priority of each arm $i$ in a way that the regret is bounded
  - Best regret in the worst case
- One common policy is UCB1 [Auer 2002]

$$\text{Priority}_i = \frac{c_i}{n_i} + \sqrt{\frac{2 \cdot \log n}{n_i}}$$

- Number of successes of arm $i$
- Total number of pulls of all arms
- Number of pulls of arm $i$
- Observed success rate
- Factor representing uncertainty
Multi-Armed Bandits: Minimax Approach (2)

\[ \text{Priority}_i = \frac{c_i}{n_i} + \sqrt{\frac{2 \cdot \log n}{n_i}} \]

- **Observed payoff**
- **Factor representing uncertainty**

As total observations \( n \) becomes large:
- Observed payoff tends asymptotically towards the true payoff probability
- The system never completely “converges” to one best arm; only the rate of exploration tends to zero
Multi-Armed Bandits: Minimax Approach (3)

\[
\text{Priority}_i = \frac{c_i}{n_i} + \sqrt{\frac{2 \cdot \log n}{n_i}}
\]

- Sub-optimal arms are pulled \(O(\log n)\)
- Hence, UCB1 has \(O(\log n)\) regret
- This is the lowest possible regret (but the constants matter 😊)
- E.g. Regret after \(n\) plays is bounded by

\[
\left( 8 \sum_{i: \mu_i < \mu_{\text{best}}} \frac{\ln n}{\Delta_i} \right) + \left( 1 + \frac{\pi^2}{3} \right) \cdot \left( \sum_{j=1}^{K} \Delta_j \right), \text{ where } \Delta_i = \mu_{\text{best}} - \mu_i
\]
Classical Multi-Armed Bandits: Summary

- Bayesian approach (Markov decision process)
  - Representative work: Gittins’ index
    - Gittins’ index is optimal for a fixed reward distribution
    - Idea: Pull the arm currently having the highest index value
  - Representative work: Whittle’s index [Whittle 1988]
    - Extend Gittins’ index to a changing reward distribution
    - Only near optimal; approximate by Lagrange relaxation
    - Computationally intensive
- Minimax approach (providing guaranteed regret bounds)
  - Representative work: UCB1 [Auer 2002]
    - Index = Upper confidence bound (model agnostic)
- Heuristics
  - \( \varepsilon \)-Greedy: Random exploration using fraction \( \varepsilon \) of traffic
  - Softmax:
    \[
    \frac{\exp\left\{ \hat{\mu}_i / \tau \right\}}{\sum_j \exp\left\{ \hat{\mu}_j / \tau \right\}}
    \]
    - \( P \)% upper confidence bound (model-based)
Characteristics of Real Recommender Systems

- Dynamic set of items (arms)
  - Items come and go with short lifetimes (e.g., a day)
  - Asymptotically optimal policies may fail to achieve good performance when item lifetimes are short

- Non-stationary CTR
  - CTR of an item can change dramatically over time
    - Different user populations at different times
    - Same user behaves differently at different times (e.g., morning, lunch time, at work, in the evening, etc.)
    - Attention to breaking news stories decay over time

- Batch serving for scalability
  - Making a decision and updating the model for each user visit in real time is expensive
  - Batch serving is more feasible: Create time slots (e.g., 5 min); for each slot, decide what fraction $x_i$ of the visits in the slot should give item $i$
Let's solve this from first principle
Bayesian Solution: Two Items, Two Time Slots (1)

- Two time slots: \( t = 0 \) and \( t = 1 \)
  - **Item P**: We are uncertain about its CTR, \( p_0 \) at \( t = 0 \) and \( p_1 \) at \( t = 1 \)
  - **Item Q**: We know its CTR exactly, \( q_0 \) at \( t = 0 \) and \( q_1 \) at \( t = 1 \)
- To determine \( x \), we need to estimate what would happen in the future

**Question:**
What fraction \( x \) of \( N_0 \) views to **Item P** (1-\( x \)) to **Item Q**

Obtain \( c \) clicks after serving \( x \) (not yet observed; random variable)

- Assume we observe \( c \); we can update \( p_1 \)
- If \( x \) and \( c \) are given, optimal solution:
  - Give all views to **Item P** iff
    \[
    E[ p_1(x,c) \mid x, c ] > q_1
    \]
    \( \hat{p}_1(x,c) \)
Bayesian Solution: Two Items, Two Time Slots (2)

- Expected total number of clicks in the two time slots

\[
\begin{align*}
E[\text{#clicks}] \text{ at } t = 0 &= N_0 x \hat{p}_0 + N_0 (1 - x) q_0 \\
E[\text{#clicks}] \text{ at } t = 1 &= N_1 E_c \left[ \max \{ \hat{p}_1 (x, c), q_1 \} \right]
\end{align*}
\]

\[
\begin{align*}
\text{Item } P & \quad \text{Item } Q & \\
N_0 x \hat{p}_0 + N_0 (1 - x) q_0 & \quad + N_1 E_c \left[ \max \{ \hat{p}_1 (x, c), q_1 \} \right]
\end{align*}
\]

\[
\begin{align*}
= N_0 q_0 + N_1 q_1 + N_0 x (\hat{p}_0 - q_0) + N_1 E_c \left[ \max \{ \hat{p}_1 (x, c) - q_1, 0 \} \right]
\end{align*}
\]

Gain\((x, q_0, q_1)\) = Expected number of additional clicks if we explore the uncertain item \(P\) with fraction \(x\) of views in slot 0, compared to a scheme that only shows the certain item \(Q\) in both slots

Solution: \(\arg\max_x \text{Gain}(x, q_0, q_1)\)
Bayesian Solution: Two Items, Two Time Slots (3)

- Approximate \( \hat{p}_1(x, c) \) by the normal distribution
  - Reasonable approximation because of the central limit theorem

\[
\text{Gain}(x, q_0, q_1) = N_0 x(\hat{p}_0 - q_0) + N_1 \left[ \sigma_1(x) \cdot \phi \left( \frac{q_1 - \hat{p}_1}{\sigma_1(x)} \right) + \left( 1 - \Phi \left( \frac{q_1 - \hat{p}_1}{\sigma_1(x)} \right) \right) (\hat{p}_1 - q_1) \right]
\]

Prior of \( p_1 \sim \text{Beta}(a, b) \)

\[
\hat{p}_1 = E_c[\hat{p}_1(x, c)] = \frac{a}{a + b}
\]

\[
\sigma_1^2(x) = \text{Var}[\hat{p}_1(x, c)] = \frac{xN_0}{(a + b + xN_0)} \frac{ab}{(a + b)^2(1 + a + b)}
\]

- Proposition: Using the approximation, the Bayes optimal solution \( x \) can be found in time \( O(\log N_0) \)
Bayesian Solution: Two Items, Two Time Slots (4)

- Quiz: Is it correct that the more we are uncertain about the CTR of an item, the more we should explore the item?
Bayesian Solution: General Case (1)

- From two items to $K$ items
  - Very difficult problem:
    \[
    \max_{\sum x_i = 1} \left( N_0 \sum x_i \hat{p}_{i0} + N_1 E_c \left[ \max_i \{ \hat{p}_{i1}(x_i, c_i) \} \right] \right)
    \]
    
    Note: $c = [c_1, ..., c_K]$  
    $c_i$ is a random variable representing the # clicks on item $i$ we may get

    \[
    \max_{\sum z_i(c) = 1} E_c \left[ \sum_i z_i(c) \hat{p}_{i1}(x_i, c_i) \right]
    \]
    
    - Apply Whittle’s Lagrange relaxation (1988) to this problem setting
      - Relax $\sum z_i(c) = 1$, for all $c$, to $E_c \left[ \sum_i z_i(c) \right] = 1$
      - Apply Lagrange multipliers ($q_1$ and $q_2$) to enforce the constraints

    \[
    \min_{q_0, q_1} \left( N_0 q_0 + N_1 q_1 + \sum_i \max_{x_i} \text{Gain}(x_i, q_0, q_1) \right)
    \]

    - We essentially reduce the $K$-item case to $K$ independent two-item sub-problems (which we have solved)
Bayesian Solution: General Case (2)

• From two intervals to multiple time slots
  – Approximate multiple time slots by two stages

• Non-stationary CTR
  – Use the Dynamic Gamma-Poisson model to estimate the CTR distribution for each item
Simulation Experiment: Different Traffic Volume

- Simulation with ground truth estimated based on Yahoo! Front Page data
- Setting: 16 live items per interval
- Scenarios: Web sites with different traffic volume (x-axis)
Simulation Experiment: Different Sizes of the Item Pool

- Simulation with ground truth estimated based on Yahoo! Front Page data
- Setting: 1000 views per interval; average item lifetime = 20 intervals
- Scenarios: Different sizes of the item pool (x-axis)
Characteristics of Different Explore/Exploit Schemes (1)

- Why the Bayesian solution has better performance
- Characterize each scheme by three dimensions:
  - **Exploitation regret**: The regret of a scheme when it is showing the item which *it thinks* is the best (may not actually be the best)
    - 0 means the scheme always picks the *actual* best
    - It quantifies the scheme’s ability of finding good items
  - **Exploration regret**: The regret of a scheme when it is exploring the items which it feels *uncertain* about
    - It quantifies the price of exploration (lower → better)
  - **Fraction of exploitation** (higher → better)
    - Fraction of exploration = 1 – fraction of exploitation
Characteristics of Different Explore/Exploit Schemes (2)

- Exploitation regret: Ability of finding good items (lower → better)
- Exploration regret: Price of exploration (lower → better)
- Fraction of Exploitation (higher → better)
Discussion: Large Content Pool

• The Bayesian solution looks promising
  – ~10% from true optimal for a content pool of 1000 live items
  • 1000 views per interval; item lifetime ~20 intervals

• Intelligent initialization (offline modeling)
  – Obtain a better item-specific prior (based on features)
  – Linear models that estimate CTR distributions
  – Hierarchical smoothing: Estimate the CTR distribution of a random article of a item category for a user segment
    • Use existing hierarchies of items and users
    • Create supervised clusters via extended version of LDA

• Feature-based explore/exploit
  – Estimate model parameters, instead of per-item CTR
  – More later
Discussion: Multiple Positions, Ranking

• Feature-based approach
  – reward(page) = model(ϕ(item 1 at position 1, … item k at position k))
  – Apply feature-based explore/exploit

• Online optimization for ranked list
  – Ranked bandits [Radlinski et al., 2008]: Run an independent bandit algorithm for each position
  – Dueling bandit [Yue & Joachims, 2009]: Actions are pairwise comparisons

• Online optimization of submodular functions
  – ∀ S₁, S₂ and a, f_a(S₁ ⊕ S₂) ≤ f_a(S₁)
    • where f_a(S) = f_a(S ⊕ ⟨a⟩) − f_a(S)
  – Streeter & Golovin (2008)
Discussion: Segmented Most Popular

- Partition users into segments, and then for each segment, provide most popular recommendation
- How to segment users
  - Hand-created segments: AgeGroup × Gender
  - Clustering based on user features
    • Users in the same cluster like similar items
- Segments can be organized by taxonomies/hierarchies
  - Better CTR models can be built by hierarchical smoothing
    • Shrink the CTR of a segment toward its parent
    • Introduce bias to reduce uncertainty/variance
  - Bandits for taxonomies (Pandey et al., 2008)
    • Explore/exploit categories/segments first; then, switch to individuals
Most Popular Recommendation: Summary

• Online model:
  – Estimate the mean and variance of the CTR of each item over time
  – Dynamic Gamma-Poisson model

• Intelligent initialization:
  – Estimate the prior mean and variance of the CTR of each item
    cluster using historical data
  – Cluster items → Maximum likelihood estimates of the priors

• Explore/exploit:
  – Bayesian: Solve a Markov decision process problem
    • Gittins’ index, Whittle’s index, approximations
    • Better performance, computation intensive
  – Minimax: Bound the regret
    • UCB1: Easy to compute
    • Explore more than necessary in practice
  – \( \varepsilon \)-Greedy: Empirically competitive for tuned \( \varepsilon \)
Online Components for Personalized Recommendation

Online models, intelligent initialization & explore/exploit
Personalized recommendation: Outline

• Online model
  – Methods for online/incremental update (cold-start problem)
    • User-user, item-item, PLSI, linear model
  – Methods for modeling temporal dynamics (concept drift problem)
    • State-space model, timeSVD++ [Koren 2009] for Netflix, tensor factorization

• Intelligent initialization (cold-start problem)
  – Feature-based prior + reduced rank regression (for linear model)

• Explore/exploit
  – Bandits with covariates
Online Update for Similarity-based Methods

- **User-user methods**
  - Key quantities: Similarity(user \(i\), user \(j\))
  - Incremental update (e.g., [Papagelis 2005])
    
    \[
    corr(i, j) = \frac{B_{jk} = \sum_k (r_{ik} - \bar{r}_i)(r_{jk} - \bar{r}_j)}{\sqrt{C_i = \sum_k (r_{ik} - \bar{r}_i)} \sqrt{D_j = \sum_k (r_{jk} - \bar{r}_j)}}
    \]

    Incrementally maintain three sets of counters: \(B, C, D\)

  - Clustering (e.g., [Das 2007])
    - MinHash (for Jaccard similarity)
    - Clusters(user \(i\)) = \((h_1(r_i), \ldots, h_K(r_i))\) ← fixed online (rebuilt periodically)
    - AvgRating(cluster \(c\), item \(j\)) ← updated online
      
      \[
      score(user i, item j) \propto \sum_k AvgRating(h_k(r_i), j)
      \]

- **Item-item methods** (similar ideas)
Online Update for PLSI

- Online update for probabilistic latent semantic indexing (PLSI) [Das 2007]

\[ p(\text{item } j \mid \text{user } i) = \sum_k p(\text{cluster } k \mid i) p(j \mid \text{cluster } k) \]

\[ \sum_{\text{user } u} I(u \text{ clicks } j) p(k \mid u) \]

\[ \sum_{\text{user } u} p(k \mid u) \]
Online Update for Linear/Factorization Model

- **Linear model:**
  \[ y_{ij} \sim \sum_k x_{ik} \beta_{jk} = x_i' \beta_j \]
  - Rating from user \( i \) to item \( j \)
  - The regression weight of item \( j \) on the \( k \)th user feature
  - \( x_i \) can be user factor vector (estimated periodically, fixed online)
  - \( \beta_j \) is a item factor vector (updated online)
  - Straightforward to fix item factors and update user factors

- **Gaussian model** (use vector notation)
  \[ \begin{align*}
  y_{ij} &\sim N(x_i' \beta_j, \sigma^2) \\
  \beta_j &\sim N(\mu_j, V_j)
  \end{align*} \]
  \[ \begin{align*}
  E[\beta_j | y] &= \text{Var}[\beta_j | y](V_j^{-1} \mu_j + \sum_i y_{ij} x_i / \sigma^2) \\
  \text{Var}[\beta_j | y] &= (V_j^{-1} + \sum_i x_i x_i' / \sigma^2)^{-1}
  \end{align*} \]
Temporal Dynamics: State-Space Model

- Item factors $\beta_{j,t}$ change over time $t$
  - The change is smooth: $\beta_{j,t}$ should be close to $\beta_{j,t-1}$

**Dynamic model**

$$y_{ij,t} \sim N(x_{i,t} \beta_{j,t}, \sigma^2)$$

$$\beta_{j,t} \sim N(\beta_{j,t-1}, V)$$

**Static model**

$$y_{ij} \sim N(x_i' \beta_j, \sigma^2)$$

$$\beta_j \sim N(\mu_j, V_j)$$

- Use standard Kalman filter update rule
- It can be extended to Logistic (for binary data), Poisson (for count data), etc.

Subscript:
user $i$,
item $j$,
time $t$
Temporal Dynamics: timeSVD++

- Explicitly model temporal patterns
- Part of the winning method of Netflix contest [Koren 2009]

\[
y_{ij,t} \sim \mu + b_i(t) + b_j(t) + u_i(t)^	op v_j
\]

- item popularity
- user bias
- user factors (preference)

\[
b_i(t) = b_i + \alpha_i \text{dev}_i(t) + b_{it}
\]

- distance to the middle rating time of \(i\)

\[
b_j(t) = b_j + b_{j,\text{bin}(t)}
\]

- time bin

\[
u_i(t)_k = u_{ik} + \alpha_{ik} \text{dev}_u(t) + u_{ikt}
\]

Model parameters: \(\mu, b_i, \alpha_i, b_{it}, b_j, b_{j,\text{bin}}, u_{ik}, \alpha_{ik}, u_{ikt}\)

for all user \(i\), item \(j\), factor \(k\), time \(t\), time bin \(d\)
Temporal Dynamics: Tensor Factorization

- Decompose ratings into three components [Xiong 2010]
  - User factors \( u_{ik} \): User \( i \)'s membership to type \( k \)
  - Item factors \( v_{jk} \): Item \( j \)'s affinity to type \( k \)
  - Time factors \( z_{tk} \): Importance/weight of type \( k \) at time \( t \)

Regular matrix factorization

\[
y_{ij} \sim \sum_k u_{ik} v_{jk} = u_{i1} v_{j1} + u_{i2} v_{j2} + \ldots + u_{iK} v_{jK}
\]

Tensor factorization

\[
y_{ij,t} \sim \sum_k u_{ik} v_{jk} z_{tk} = u_{i1} v_{j1} z_{t1} + u_{i2} v_{j2} z_{t2} + \ldots + u_{iK} v_{jK} z_{tK}
\]

Time-varying weights on different types/factors

\[
z_{t,k} \sim N(z_{t-1,k}, \sigma^2) \quad \text{Time factors are smooth over time}
\]

Subscript:
- user \( i \)
- item \( j \)
- time \( t \)
Online Models: Summary

• Why online model? Real systems are dynamic!!
  – Cold-start problem: New users/items come to the system
    • New data should be used a.s.a.p., but rebuilding the entire model is expensive
    • How to efficiently, incrementally update the model
      – Similarity-based methods, PLSI, linear and factorization models
      – Concept-drift problem: User/item behavior changes over time
        • Decay the importance of old data
          – State-space model
        • Explicitly model temporal patterns
          – timeSVD++ for Netflix, tensor factorization ← Not really online models!!
  
• Next
  – Initialization methods for factorization models (for cold start)
    • Start from linear regression models
Intelligent Initialization for Linear Model (1)

- Dynamic linear model

\[ y_{ij,t} \sim N(x'_{i,t} \beta_{j,t}, \sigma^2) \]
\[ \beta_{j,t} \sim N(\beta_{j,t-1}, V) \]
\[ \beta_{j,1} \sim N(\mu_{j,0}, V_0) \]

- How to estimate the prior parameters $\mu_{j,0}$ and $V_0$
  - Important for cold start: Predictions are made using prior
  - Leverage available features

- How to learn the weights/factors quickly
  - High dimensional $\beta_j \rightarrow$ slow convergence
  - Reduce the dimensionality
### Intelligent Initialization for Linear Model (2)

#### Original model

- \( y_{ij,t} \sim N(x'_{i,t} \beta_{j,t}, \sigma^2) \)
- \( \beta_{j,t} \sim N(\beta_{j,t-1}, V) \)
- \( \beta_{j,1} \sim N(\mu_{j,0}, V_0) \)

**Subscript:**
- user \( i \)
- item \( j \)
- time \( t \)

**Features:**
- \( x_{i,t} \): Feature vector of user \( i \) at time \( t \)
- \( x_{j,t} \): Feature vector of item \( j \) at time \( t \)

#### Revised model

- \( y_{ij,t} \sim N(x'_{i,t} A x_{j,t} + x'_{i,t} \beta_{j,t}, \sigma^2) \)

**Revised model details:**

- Feature-based regression
- Correction to regression
- Can be 0 mean
- Smaller scale

\[
\sum_{k\ell} A_{k\ell} \cdot x_{i,t,k} x_{j,t,\ell}
\]

**Matrix of regression weights**

- \( \beta_{j,t} = B \theta_{j,t} \)
- \( p \times r \) where \( r << p \)
- \( B \) projects the high dim space to a low dim one

- \( \theta_{j,t} \sim N(\theta_{j,t-1}, \sigma_{\theta}^2 I) \)
- \( \theta_{j,1} \sim N(0, \sigma_0^2 I) \)

**Low rank approx of var-cov matrix**

- OK!! \( \beta_{j,1} \sim N(0, \sigma_0^2 BB') \)

- (because of \( A \))
Intelligent Initialization for Linear Model (3)

- Model fitting
  - Offline training
    - Determine $\eta = (A, B, \sigma, \sigma_{\theta}, \sigma_0)$
      - Regression weights: $A, B$
      - Prior variances: $\sigma, \sigma_{\theta}, \sigma_0$
    - Latent factors $\Theta = \{ \theta_{j,t} \}$
    - Given data $\mathbf{D}$, find maximum likelihood estimate (MLE) of $\eta$
      \[
      \arg \max_\eta p(\mathbf{D} \mid \eta) = \int p(\mathbf{D}, \Theta \mid \eta) \, d\Theta
      \]
  - Use the EM algorithm
  - Online update
    - Fix $A, B, \sigma, \sigma_{\theta}, \sigma_0$
    - Update $\theta_{j,t}$ (low dimensional, use Kalman filter)
    - $\theta_{j,t}$ for each item $j$ can be updated independently in parallel

\[
\begin{align*}
  y_{i,j,t} & \sim N(x_{i,t} A x_{j,t} + x_{i,t} \beta_{j,t}, \sigma^2) \\
  \beta_{j,t} & = B \theta_{j,t}  \\
  \theta_{j,t} & \sim N(\theta_{j,t-1}, \sigma_\theta^2 I)  \\
  \theta_{j,1} & \sim N(0, \sigma_0^2 I)
\end{align*}
\]
Intelligent Initialization for Linear Model (4)

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

- **Data**: My Yahoo! data

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

Intelligent init
Intelligent Initialization for Factorization Model (1)

- Online update for item cold start (no temporal dynamics)

**Offline model**

\[ y_{ij} \sim N(u_i^{\prime}v_j, \sigma^2 I) \]

\[ u_i \sim N(Gx_i, \sigma^2 I) \]

\[ v_j = Ax_j + B\theta_j \]

\[ \theta_j \sim N(0, \sigma^2 \theta I) \]

**Feature-based init**

**Dim reduction**

**Factorization**

**(periodic)** offline training output:

\[ u_i, A, B, \sigma^2 \theta \]

**Online model**

\[ y_{ij,t} \sim N(u_i^{\prime}Ax_j + u_i^{\prime}B\theta_{j,t}, \sigma^2 I) \]

\[ \theta_{j,t} = \theta_{j,t-1} \] **Updated online**

\[ \theta_{j,1} \sim N(0, \sigma^2 \theta I) \]

**Offset**

**Feature vector**

**Scalability:**

- \( \theta_{j,t} \) is low dimensional
- \( \theta_{j,t} \) for each item \( j \) can be updated independently in parallel
Intelligent Initialization for Factorization Model (2)

Offline
\[ y_{ij} \sim N(u_i^t v_j, \sigma^2 I) \]
\[ u_i \sim N(Gx_i, \sigma_u^2 I) \]
\[ v_j = Ax_j + B\theta_j \]
\[ \theta_j \sim N(0, \sigma_\theta^2 I) \]

Online
\[ y_{ij,t} \sim N(u_i^t Ax_j + u_i^t B\theta_{j,t}, \sigma^2 I) \]
\[ \theta_{j,t} = \theta_{j,t-1} \]
\[ \theta_{j,1} \sim N(0, \sigma_\theta^2 I) \]

• Our observation so far
  – Dimension reduction \((u_i^t B)\) does not improve much if factor regressions are based on good covariates \((\sigma_\theta^2 \text{ is small})\)
    • Small \(\sigma_\theta^2 \rightarrow \text{strong shrinkage} \rightarrow \text{small effective dimensionality} \)
      (soft dimension reduction)
  – Online updates help significantly: In MovieLens (time-based split), reduced RMSE from .93 to .86
Intelligent Initialization for Factorization Model (3)

- Include temporal dynamics

**Offline computation**
(rebuilt periodically)

\[ y_{ij,t} \sim N(u_{i,t}', v_{j,t}, \sigma^2 I) \]

\[ u_{i,t} = Gx_{i,t} + H\delta_{i,t}, \]

\[ \delta_{i,t} \sim N(\delta_{i,t-1}, \sigma^2 I) \]

\[ \delta_{i,1} \sim N(0, s^2 I) \]

\[ v_{j,t} = Dx_{j,t} + B\theta_{j,t} \]

\[ \theta_{j,t} \sim N(\theta_{j,t-1}, \sigma^2 I) \]

\[ \theta_{j,1} \sim N(0, s^2 I) \]

**Online computation**

Fix \( u_{i,t} \) and update \( \theta_{j,t} \)

\[ y_{ij,t} \sim N(u_{i,t}', Dx_{j,t} + u_{i,t} B\theta_{j,t}, \sigma^2 I) \]

\[ \theta_{j,t} \sim N(\theta_{j,t-1}, \sigma^2 I) \]

Fix \( v_{j,t} \) and update \( \delta_{i,t} \)

\[ y_{ij,t} \sim N(v_{j,t}', Gx_{i,t} + v_{j,t} H\delta_{i,t}, \sigma^2 I) \]

\[ \delta_{i,t} \sim N(\delta_{i,t-1}, \sigma^2 I) \]

Repeat the above two steps a few times
Intelligent Initialization: Summary

• Online models are useful for cold start and concept drift
• Whenever historical data is available, do not start cold
• For linear / factorization models
  – Use available features to setup the starting point
  – Reduce dimensionality to facilitate fast learning

• Next
  – Explore/exploit for personalization
  – Users are represented by covariates
    • Features, factors, clusters, etc
  – Bandits with covariates
Explore/Exploit with Covariates/Features

- It provides solution to
  - Large content pool (correlated arms)
  - Personalized recommendation (hint before pulling an arm)
    • Covariate bandits, contextual bandits, bandits with side observations

- Models: Reward (CTR) is a (stochastic) function of covariates
  - Hierarchical model: CTR of a child is centered around its parent
  - Linear model: $f(CTR) = \text{weighted sum of covariate values}$
  - Similarity model: Similar items have similar CTRs
  - More general models or model agnostic

- Approaches:
  - Hierarchical explore/exploit
  - Variants of upper confidence bound methods
    • Model-based (Bayesian) vs. model-agnostic (minimax)
  - Variants of $\epsilon$-greedy ($\epsilon$ depends on observed data)
  - Variants of softmax
    \[
    \frac{\exp\{\hat{\mu}_i / \tau\}}{\sum_j \exp\{\hat{\mu}_j / \tau\}}
    \]
Are Covariate Bandits Difficult?

- When features are predictive and different users/items have different features, the myopic scheme is near optimal
  - Myopic scheme: Pick the item having the highest predicted CTR (without considering the explore/exploit problem at all)
  - Sarkar (1991) and Wang et al. (2005) studied this for the two-armed bandit case

- Simple predictive upper confidence bound gave good empirical results
  - Pick the item having highest $E[\text{CTR} \mid \text{data}] + k \cdot \text{Std}[\text{CTR} \mid \text{data}]$
  - Pavlidis et al. (2008) studied this for Gaussian linear models
  - Preliminary experiments (Gamma linear model)
    - Bayesian scheme is better when features are not very predictive
    - Simple predictive UCB is better when features are predictive
Covariate Bandits

- Related work ... just a small sample of papers
  - Hierarchical explore/exploit (Pandey et al., 2008)
    - Explore/exploit categories/segments first; then, switch to individuals
  - Variants of $\varepsilon$-greedy
    - Epoch-greedy (Langford & Zhang, 2007): $\varepsilon$ is determined based on the generalization bound of the current model
    - Banditron (Kakade et al., 2008): Linear model with binary response
    - Non-parametric bandit (Yang & Zhu, 2002): $\varepsilon$ decreases over time; example model: histogram, nearest neighbor
  - Variants of UCB methods
    - Linearly parameterized bandits (Rusmevichientong et al., 2008): minimax, based on uncertainty ellipsoid
    - Bandits in metric spaces (Kleinberg et al., 2008; Slivkins et al., 2009):
      - Similar arms have similar rewards: $|\text{reward}(i) - \text{reward}(j)| \leq \text{distance}(i,j)$
Online Components: Summary

• Real systems are dynamic
• Cold-start problem
  – Incremental online update (online linear regression)
  – Intelligent initialization (use features to predict initial factor values)
  – Explore/exploit (pick posterior mean + \( k \) posterior standard dev)

• Concept-drift problem
  – Tracking the current behavior (state-space models, Kalman filter)
  – Modeling temporal patterns
Evaluation Methods and Challenges
Evaluation Methods

• Ideal method
  – Experimental Design: Run side-by-side experiments on a small fraction of randomly selected traffic with new method (treatment) and status quo (control)
  – Limitation
    • Often expensive and difficult to test large number of methods

• Problem: How do we evaluate methods offline on logged data?
  – Goal: To maximize clicks/revenue and not prediction accuracy on the entire system. Cost of predictive inaccuracy for different instances vary.
    • E.g. 100% error on a low CTR article may not matter much because it always co-occurs with a high CTR article that is predicted accurately
Usual Metrics

• Predictive accuracy
  – Root Mean Squared Error (RMSE)
  – Mean Absolute Error (MAE)
  – Area under the Curve, ROC

• Other rank based measures based on retrieval accuracy for top-k
  – Recall in test data
    • What Fraction of items that user actually liked in the test data were among the top-k recommended by the algorithm (fraction of hits, e.g. Karypsis, CIKM 2001)

• One flaw in several papers
  – Training and test split are not based on time.
    • Information leakage, results not valid
    • Even in Netflix, this is the case to some extent
      – Time split per user, not per event. For instance, information will leak if models are based on user-user similarity.
Metrics continued..

- **Recall per event based on Replay-Match method**
  - Fraction of clicked events where the top recommended item matches the clicked one.

- **This is good if logged data collected from a randomized serving scheme, with biased data this will be a problem**
  - We will be inventing algorithms that provide recommendations that are similar to the current one
    - No reward for novel recommendations
Details on Replay-Match method (Li, Langford, et al)

- $x$: feature vector for a visit
- $r = [r_1, r_2, \ldots, r_K]$: reward vector for the $K$ items in inventory
- $h(x)$: recommendation algorithm to be evaluated
- Goal: Estimate expected reward for $h(x)$

$$E_{(x,r) \sim \mathcal{P}} \left[ \sum_i \Pr(h(x) = i) \cdot r_i \right]$$

- $s(x)$: recommendation scheme that generated logged-data
- $x_1, \ldots, x_T$: visits in the logged data
- $r_{ti}$: reward for visit $t$, where $i = s(x_t)$
Replay-Match continued

• Estimator

\[
\frac{1}{T} \sum_{t} \sum_{i} I(h(x_t) = i \text{ and } s(x_t) = i) \cdot r_{ti} \cdot \alpha_t
\]

• If importance weights and \((x_t, r_t) \text{ iid } \sim \mathcal{P}\).

  – It can be shown estimator is unbiased

• E.g. if \(s(x)\) is random serving scheme, importance weights are uniform over the item set

• If \(s(x)\) is not random, importance weights have to be estimated through a model
Recall: Some examples

• Simple version
  – I have an important module on my page, content inventory is obtained from a third party source which is further refined through editorial oversight. Can I algorithmically recommend content on this module? I want to drive up total CTR on this module

• More advanced
  – I got X% lift in CTR. But I have additional information on other downstream utilities (e.g. dwell time). Can I increase downstream utility without losing too many clicks?

• Highly advanced
  – There are multiple modules running on my website. How do I take a holistic approach and perform a simultaneous optimization?
For the simple version

• Multi-position optimization
  – Explore/exploit, optimal subset selection

• Explore/Exploit strategies for large content pool and high dimensional problems
  – Some work on hierarchical bandits but more needs to be done

• Better offline evaluation strategies
  – This is important for progress in this area

• Constructing user profiles from multiple sources with less than full coverage

• Content understanding

• Metrics to measure user engagement (other than CTR)
Other problems

- Whole page optimization
  - Challenging, open area

- Content programming
  - How should we generate content to enhance our inventory?

- Incentivizing User generated content

- Incorporating Social information for better recommendation