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
    - … may not need to be online if the patterns are stationary

Big Picture

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

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)

  - *Sometimes hard to beat!!*

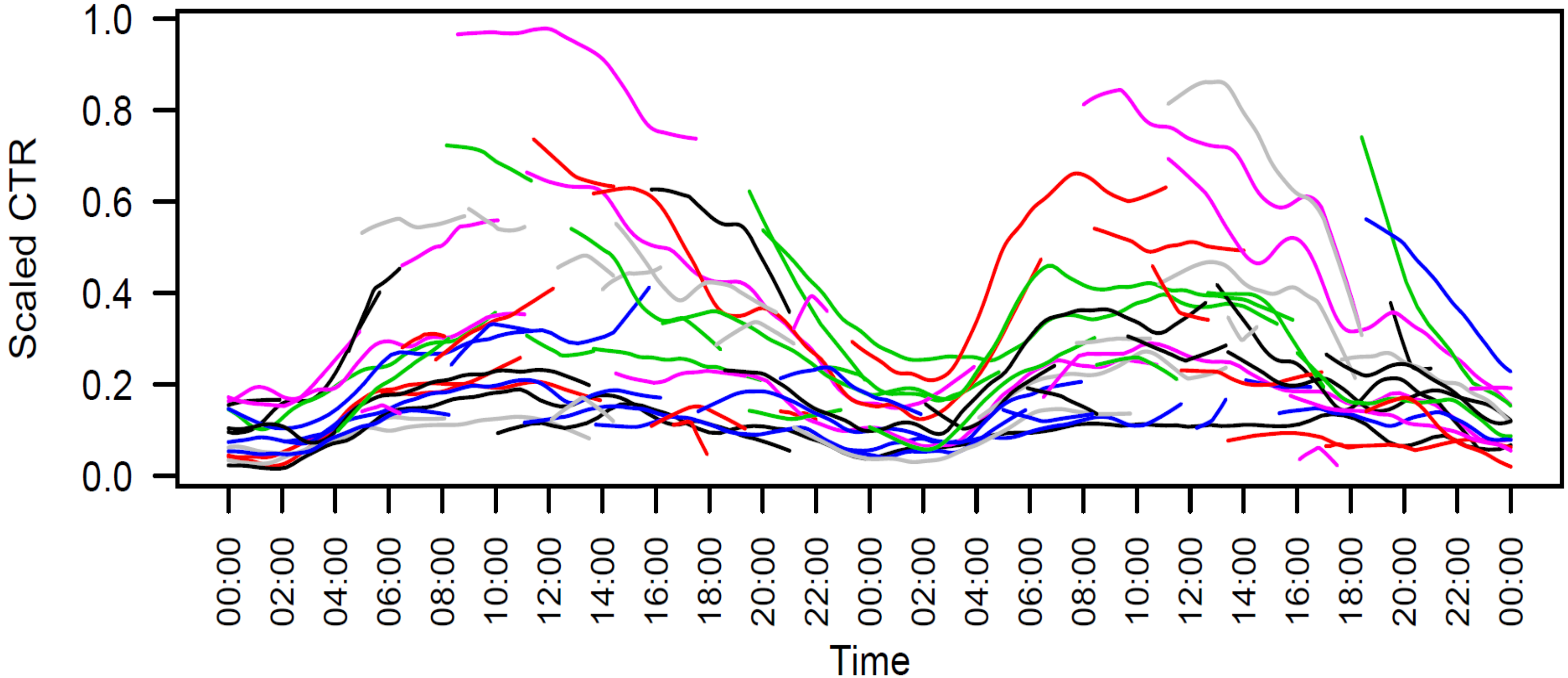
- Segmented most popular recommendation
  - Create user segments/clusters based on user features
  - Provide most popular recommendation for each segment

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 $\rightarrow$ Unreliable CTR estimates
    - Too much $\rightarrow$ 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)

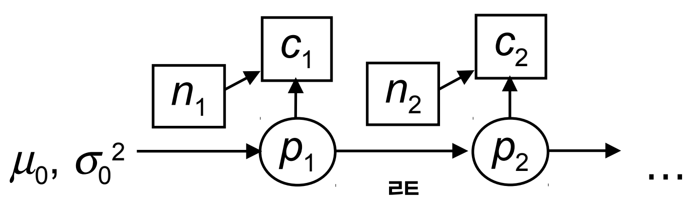
Online Models

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

Online Models: Dynamic Gamma-Poisson

Notation:
- Show the item n_t times
- Receive c_t clicks
- p_t = CTR at time t

- Model-based approach
  - $(c_t | n_t, p_t) \sim \text{Poisson}(n_t p_t)$
  - $p_t = p_{t-1} \varepsilon_t$, 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
    - ∀ η 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)



Online Models: Derivation

Estimated CTR distribution at time *t*

$$(p_t \mid c_1, n_1, \ldots, c_{t-1}, n_{t-1}) \sim Gamma(mean = \mu_t, var = \sigma_t^2)$$

$$\text{Let } \gamma_t = \mu_t / \sigma_t^2 \;\; \text{(effective sample size)}$$

$$(p_t \mid c_1, n_1, \ldots, c_t, n_t) \sim Gamma(mean = \mu_{t|t}, var = \sigma_{t|t}^2)$$

$$\text{Let } \gamma_{t|t} = \gamma_t + n_t \;\; \text{(effective sample size)}$$

$$\mu_{t|t} = (\mu_t \cdot \gamma_t + c_t) / \gamma_{t|t}$$

$$\sigma_{t|t}^2 = \mu_{t|t} / \gamma_{t|t}$$

Estimated CTR distribution at time *t*+1

$$(p_{t+1} \mid c_1, n_1, \ldots, c_t, n_t) \sim Gamma(mean = \mu_{t+1}, var = \sigma_{t+1}^2)$$

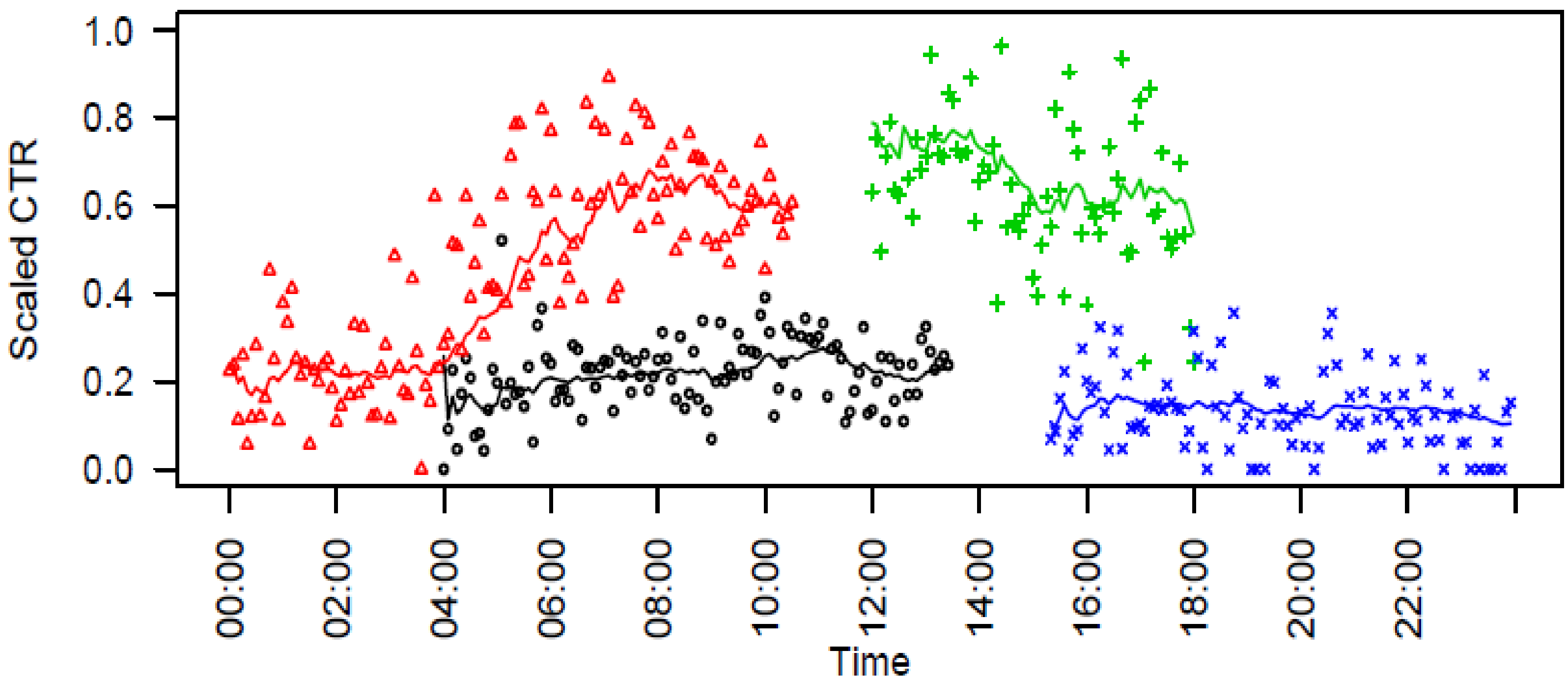
$$\mu_{t+1} = \mu_{t|t}$$

$$\sigma_{t+1}^2 = \sigma_{t|t}^2 + \eta(\mu_{t|t}^2 + \sigma_{t|t}^2)$$

High CTR items more adaptive

Tracking behavior of Gamma-Poisson model

- Low click rate articles – More temporal smoothing



Intelligent Initialization: Prior Estimation

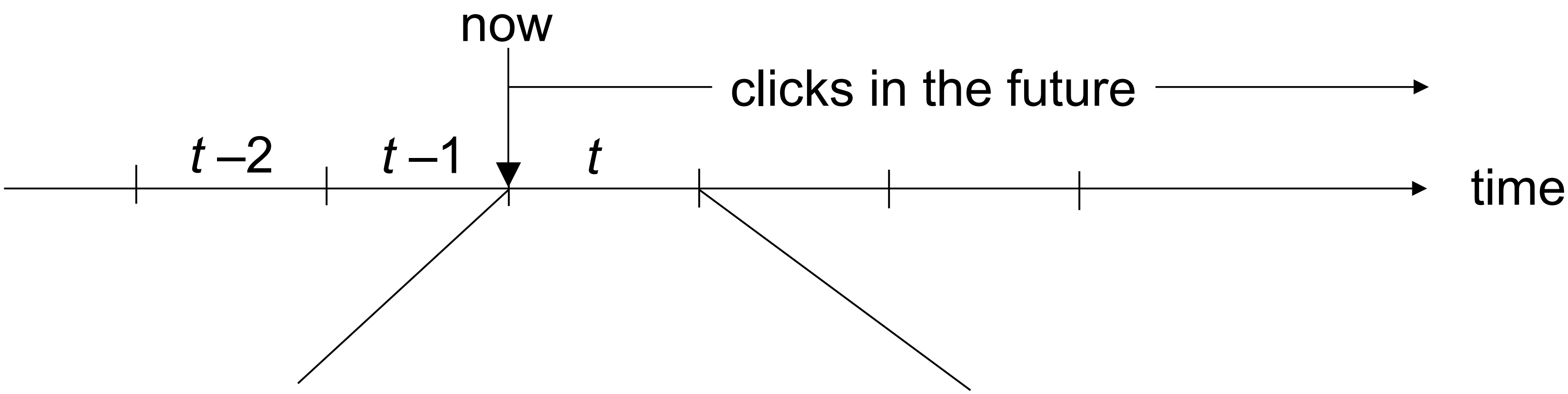
- Prior CTR distribution: Gamma(mean=μ_0, var=σ_0^2)
  - *N* historical items:
    - n_i = #views of item *i* in its first time interval
    - c_i = #clicks on item *i* in its first time interval
  - Model
    - $c_i \sim$ Poisson($n_i p_i$) and $p_i \sim$ Gamma(μ_0, σ_0^2)

      $\Rightarrow$ $c_i \sim$ NegBinomial(μ_0, σ_0^2, n_i)
  - Maximum likelihood estimate (MLE) of (μ_0, σ_0^2)

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

- Better prior: Cluster items and find MLE for each cluster
  - Agarwal & Chen, 2011 (SIGMOD)

Explore/Exploit: Problem Definition

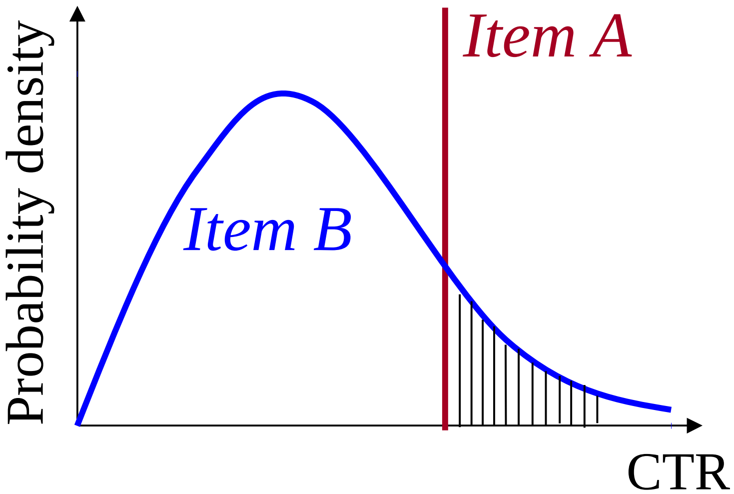


Item 1	x_1% page views
Item 2	x_2% page views
…	…
Item K	x_K% page views

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

p_1

p_2

p_3

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)

p_1

p_2

p_3

Bandit “arms”

(unknown payoff probabilities)

- 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**
  - 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)

Multi-Armed Bandits: Markov Decision Process (1)

- Select an arm **now** at time t=0, to maximize expected total number of clicks in t=0,…,T
- State at time t: $\boldsymbol{\Theta}_t = (\theta_{1t}, \ldots, \theta_{Kt})$
  θ_{it} = State of arm i at time t (that captures all we know about arm i at t)
- Reward function $R_i(\boldsymbol{\Theta}_t, \boldsymbol{\Theta}_{t+1})$
  – Reward of pulling arm i that brings the state from $\boldsymbol{\Theta}_t$ to $\boldsymbol{\Theta}_{t+1}$
- Transition probability Pr[$\boldsymbol{\Theta}_{t+1}$ | $\boldsymbol{\Theta}_t$, pulling arm i]
- Policy π: A function that maps a state to an arm (action)
  $\pi(\boldsymbol{\Theta}_t)$ returns an arm (to pull)
- Value of policy π starting from the current state $\boldsymbol{\Theta}_0$ with horizon T

$$V_T(\pi,\boldsymbol{\Theta}_0) = E\Big[\overbrace{R_{\pi(\boldsymbol{\Theta}_0)}(\boldsymbol{\Theta}_0,\boldsymbol{\Theta}_1)}^{\text{Immediate reward}} + \overbrace{V_{T-1}(\pi,\boldsymbol{\Theta}_1)}^{\text{Value of the remaining } T\text{-1 time slots if we start from state } \boldsymbol{\Theta}_1}\Big]$$

$$= \int \Pr[\boldsymbol{\Theta}_1 \mid \boldsymbol{\Theta}_0, \pi(\boldsymbol{\Theta}_0)] \cdot \left[R_{\pi(\boldsymbol{\Theta}_0)}(\boldsymbol{\Theta}_0,\boldsymbol{\Theta}_1) + V_{T-1}(\pi,\boldsymbol{\Theta}_1)\right] d\boldsymbol{\Theta}_1$$

Multi-Armed Bandits: MDP (2)

$$V_T(\pi, \boldsymbol{\Theta}_0) = E\Big[\underbrace{R_{\pi(\boldsymbol{\Theta}_0)}(\boldsymbol{\Theta}_0, \boldsymbol{\Theta}_1)}_{\text{Immediate reward}} + \underbrace{V_{T-1}(\pi, \boldsymbol{\Theta}_1)}_{\text{Value of the remaining } T\text{-1 time slots if we start from state } \boldsymbol{\Theta}_1}\Big]$$

$$= \int \Pr[\boldsymbol{\Theta}_1 \mid \boldsymbol{\Theta}_0, \pi(\boldsymbol{\Theta}_0)] \cdot \big[R_{\pi(\boldsymbol{\Theta}_0)}(\boldsymbol{\Theta}_0, \boldsymbol{\Theta}_1) + V_{T-1}(\pi, \boldsymbol{\Theta}_1)\big]\, d\boldsymbol{\Theta}_1$$

- Optimal policy: $\arg\max_{\pi} \; V_T(\pi, \boldsymbol{\Theta}_0)$
- 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
    $R_{total} = R(0) + \gamma.R(1) + \gamma^2.R(2) + \ldots \qquad (0<\gamma<1)$
- Theorem [Gittins 1979]: The optimal policy decouples and solves a bandit problem for each arm independently

✗ Policy $\pi(\Theta_t)$ is a function of $(\theta_{1t}, \ldots, \theta_{Kt})$ **One *K*-dimensional problem**

✓ Policy $\pi(\Theta_t) = \text{argmax}_i \{ g(\theta_{it}) \}$ **K one-dimensional problems**

Gittins' Index

Still computationally expensive!!

Multi-Armed Bandits: MDP (4)

Priority 1

Priority 2

Priority 3

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, **whether to 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

      θ_{it} = (#successes, #pulls) for arm i up to time t
    - g maps each possible (#successes, #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
  - Lowest regret in the worst case
- One common policy is UCB1 [Auer 2002]

Number of successes of arm i

Total number of pulls of all arms

$$\text{Priority}_i = \underbrace{\frac{c_i}{n_i}}_{\text{Observed success rate}} + \underbrace{\sqrt{\frac{2 \cdot \log n}{n_i}}}_{\text{Factor representing uncertainty}}$$

Number of pulls of arm i

Multi-Armed Bandits: Minimax Approach (2)

$$\text{Priority}_i = \underbrace{\frac{c_i}{n_i}}_{\textit{Observed payoff}} + \underbrace{\sqrt{\frac{2 \cdot \log n}{n_i}}}_{\textit{Factor representing uncertainty}}$$

- 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)

$$\text{Priority}_i = \underbrace{\frac{c_i}{n_i}}_{\text{Observed payoff}} + \underbrace{\sqrt{\frac{2 \cdot \log n}{n_i}}}_{\text{Factor representing uncertainty}}$$

- Sub-optimal arms are pulled $O(\log n)$ times
- 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

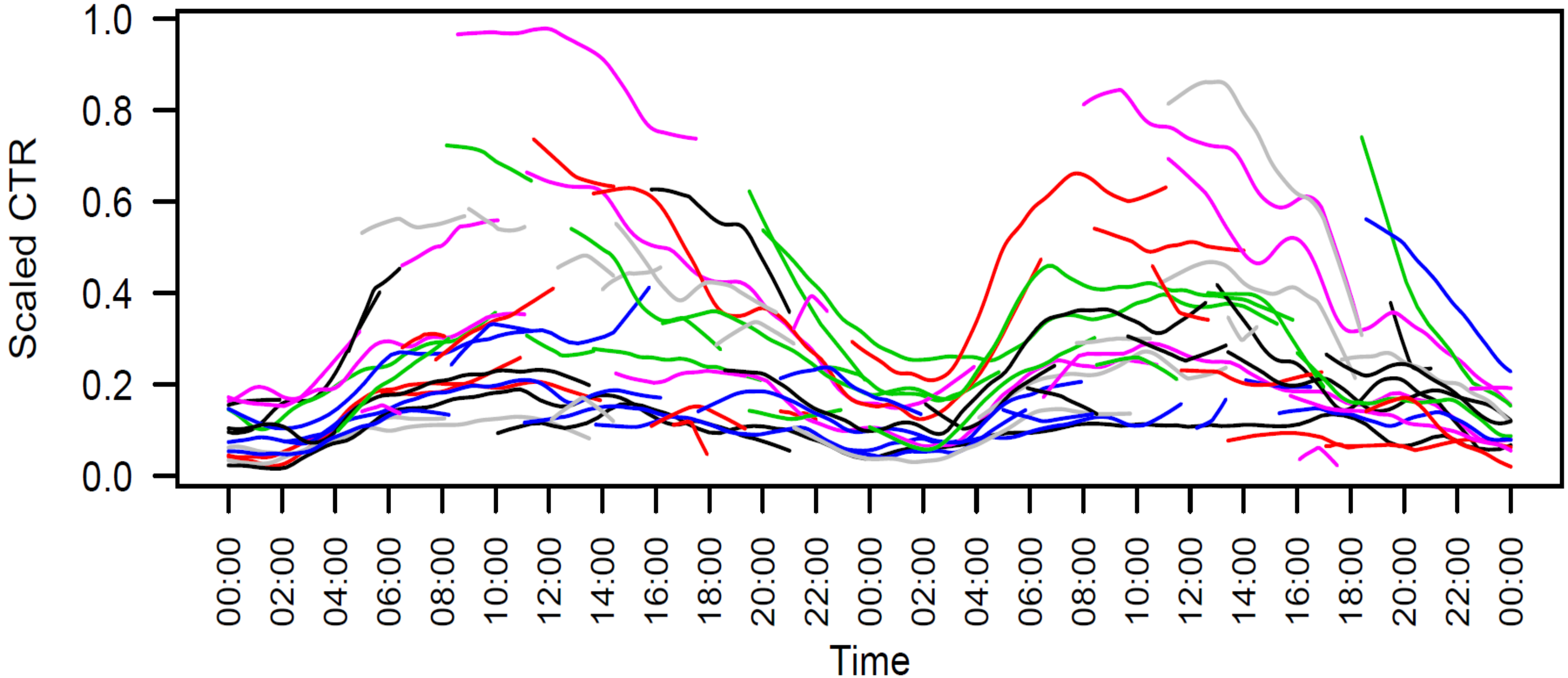
$$\left(8 \sum_{i:\mu_i < \mu_{best}} \frac{\ln n}{\Delta_i}\right) + \left(1 + \frac{\pi^2}{3}\right) \cdot \left(\sum_{j=1}^{K} \Delta_j\right), \text{ where } \Delta_i = \mu_{best} - \mu_i$$

Classical Multi-Armed Bandits: Summary

- Bayesian approach (Markov decision process)
  - Representative work: Gittins' index [Gittins 1979]
    - 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
  - ε-Greedy: Random exploration using fraction ε of traffic
  - Softmax: $\frac{\exp\{\hat{\mu}_i / \tau\}}{\sum_j \exp\{\hat{\mu}_j / \tau\}}$ $\quad \hat{\mu}_i = \text{predicted CTR of item } i, \quad \tau = \text{temperature}$
  - *P*% upper confidence bound (model-based)

Do Classical Bandits Apply to Web Recommenders?

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:

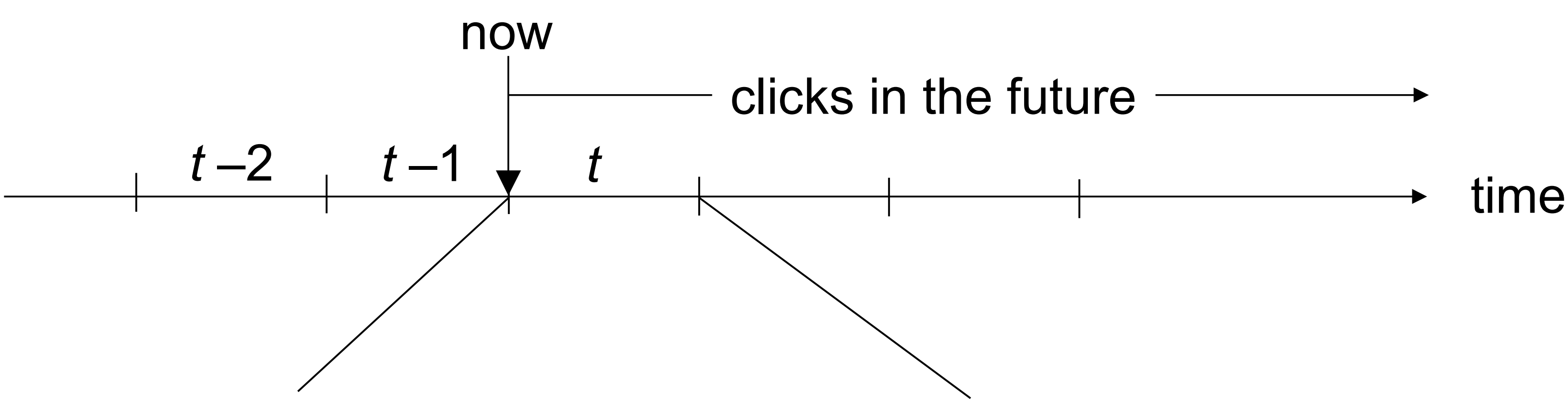
(a) Short lifetimes, (b) temporal effects, (c) often breaking news stories

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 decays 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*

[Agarwal et al., ICDM, 2009]

Explore/Exploit in Recommender Systems



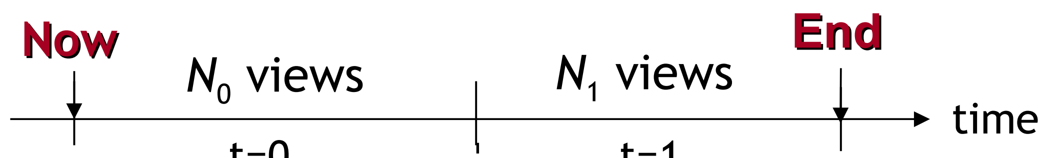
Item 1	x_1% page views
Item 2	x_2% page views
…	…
Item K	x_K% page views

Determine (x_1, x_2, …, x_K) based on clicks and views observed before t in order to maximize the expected total number of clicks in the future

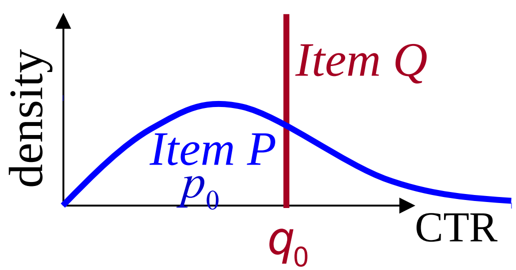
Let's solve this from first principle

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

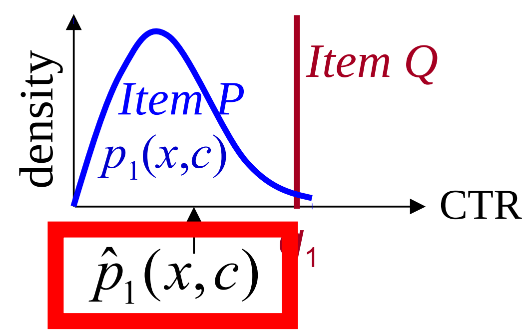


Question:
What fraction ***x*** of N_0 views to *item P*
(1-***x***) to *item Q*



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
  $\underbrace{E[\, p_1(x,c) \mid x, c\,]}_{\hat{p}_1(x,c)} > q_1$



Bayesian Solution: Two Items, Two Time Slots (2)

- Expected total number of clicks in the two time slots

$$\underbrace{\overbrace{\underbrace{N_0 x \hat{p}_0}_{\textit{Item P}} + \underbrace{N_0(1-x)q_0}_{\textit{Item Q}}}^{E[\#\text{clicks}] \text{ at } t=0} + \overbrace{N_1 E_c[\max\{\hat{p}_1(x,c),\ q_1\}]}^{E[\#\text{clicks}] \text{ at } t=1}}$$

Show the item with higher E[CTR]: $\max\{\hat{p}_1(x,c),\ q_1\}$

$$= \underbrace{N_0 q_0 + N_1 q_1}_{\substack{\text{E[\#clicks] if we} \\ \text{always show} \\ \textit{item Q}}} + \underbrace{N_0 x(\hat{p}_0 - q_0) + N_1 E_c[\max\{\hat{p}_1(x,c) - q_1,\ 0\}]}_{Gain(x,\, q_0,\, q_1)}$$

Gain of exploring the uncertain *item P* using ***x***

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: argmax_x 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

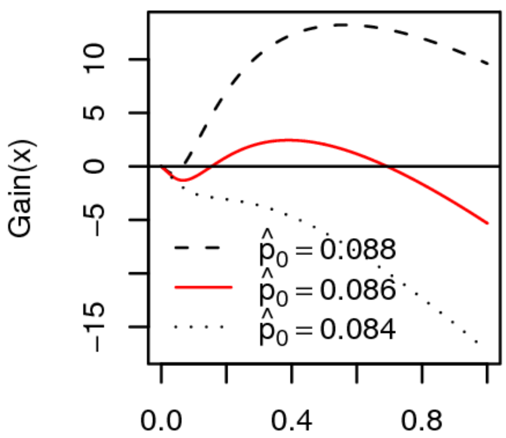
$$Gain(x,q_0,q_1) = N_0 x(\hat{p}_0 - q_0) + N_1\left[\sigma_1(x)\cdot\phi\left(\frac{q_1-\hat{p}_1}{\sigma_1(x)}\right) + \left(1-\Phi\left(\frac{q_1-\hat{p}_1}{\sigma_1(x)}\right)\right)(\hat{p}_1 - q_1)\right]$$

Prior of $p_1 \sim Beta(a,b)$

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

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

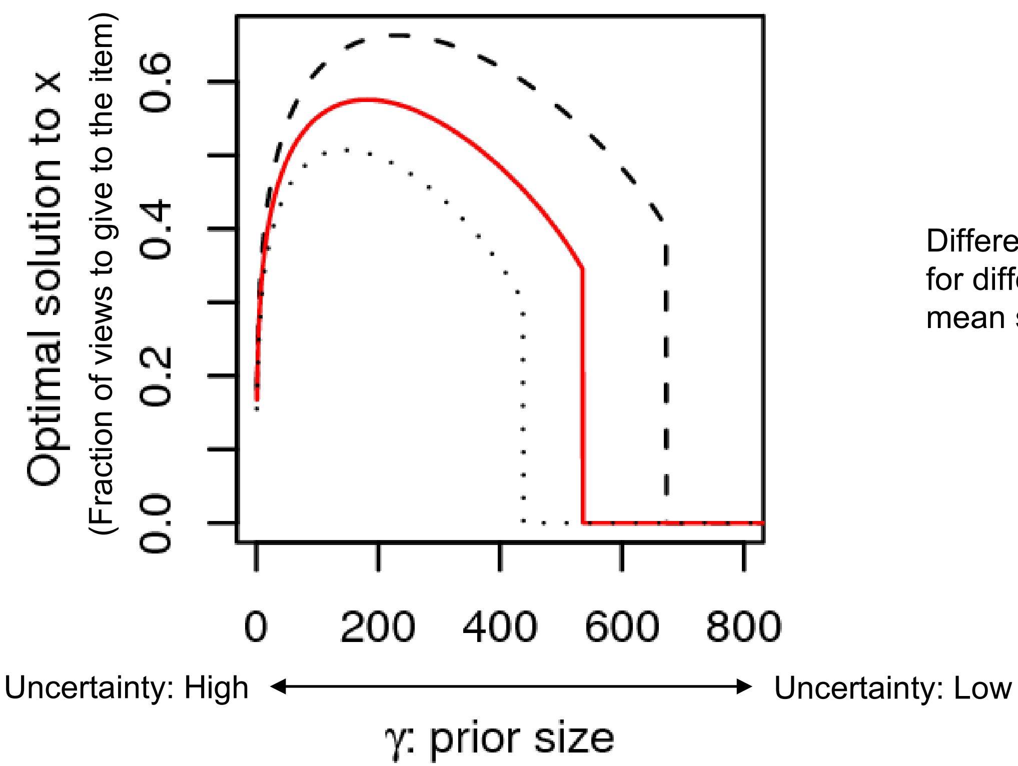
- Proposition: Using the approximation, the Bayes optimal solution *x* can be found in time $O(\log N_0)$



x: Fraction of views for uncertain item

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?



Different curves are for different prior mean settings

Bayesian Solution: General Case (1)

- From two items to *K* items
  - Very difficult problem: $\max_{\substack{\mathbf{x}\geq 0 \\ \sum_i x_i = 1}} \left(N_0 \sum_i x_i \hat{p}_{i0} + N_1 \underbrace{E_{\mathbf{c}}[\max_i \{\hat{p}_{i1}(x_i, c_i)\}]}_{\displaystyle || \; \max_{\mathbf{z}\geq 0} E_{\mathbf{c}}[\textstyle\sum_i z_i(\mathbf{c})\hat{p}_{i1}(x_i, c_i)]} \right)$

    $\sum_i z_i(\mathbf{c}) = 1$, for all possible $\mathbf{c}$

    Note: $\mathbf{c} = [c_1, \ldots, c_K]$

    c_i is a **random variable** representing the # clicks on item *i* we may get
  - Apply Whittle's Lagrange relaxation (1988) to this problem setting
    - Relax $\sum_i z_i(\mathbf{c}) = 1$, for all $\mathbf{c}$, to $E_{\mathbf{c}} [\sum_i z_i(\mathbf{c})] = 1$
    - Apply Lagrange multipliers (q_1 and q_2) to enforce the constraints

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

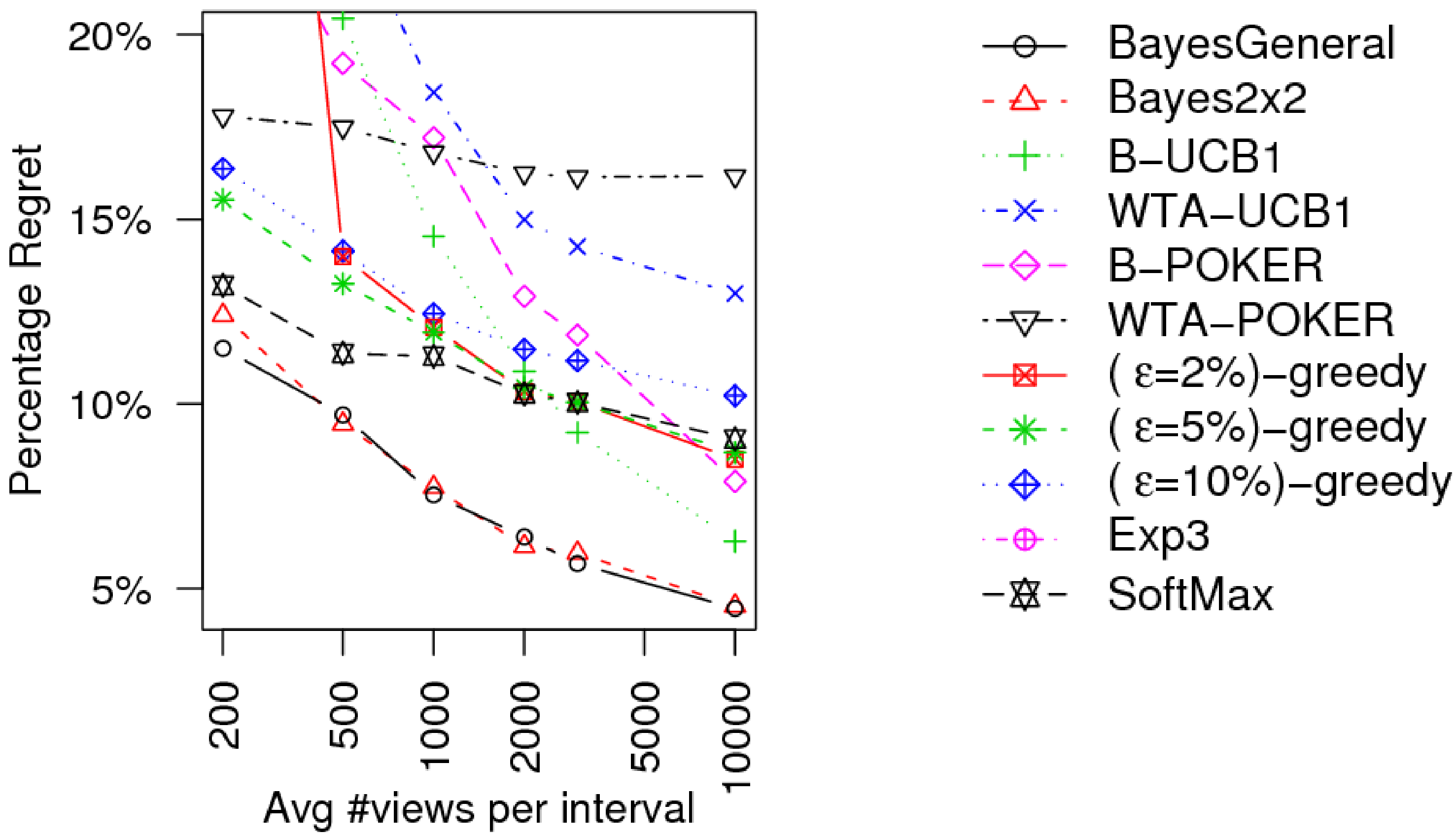
  - We essentially reduce the *K*-item case to *K* independent two-item sub-problems (which we have solved)

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

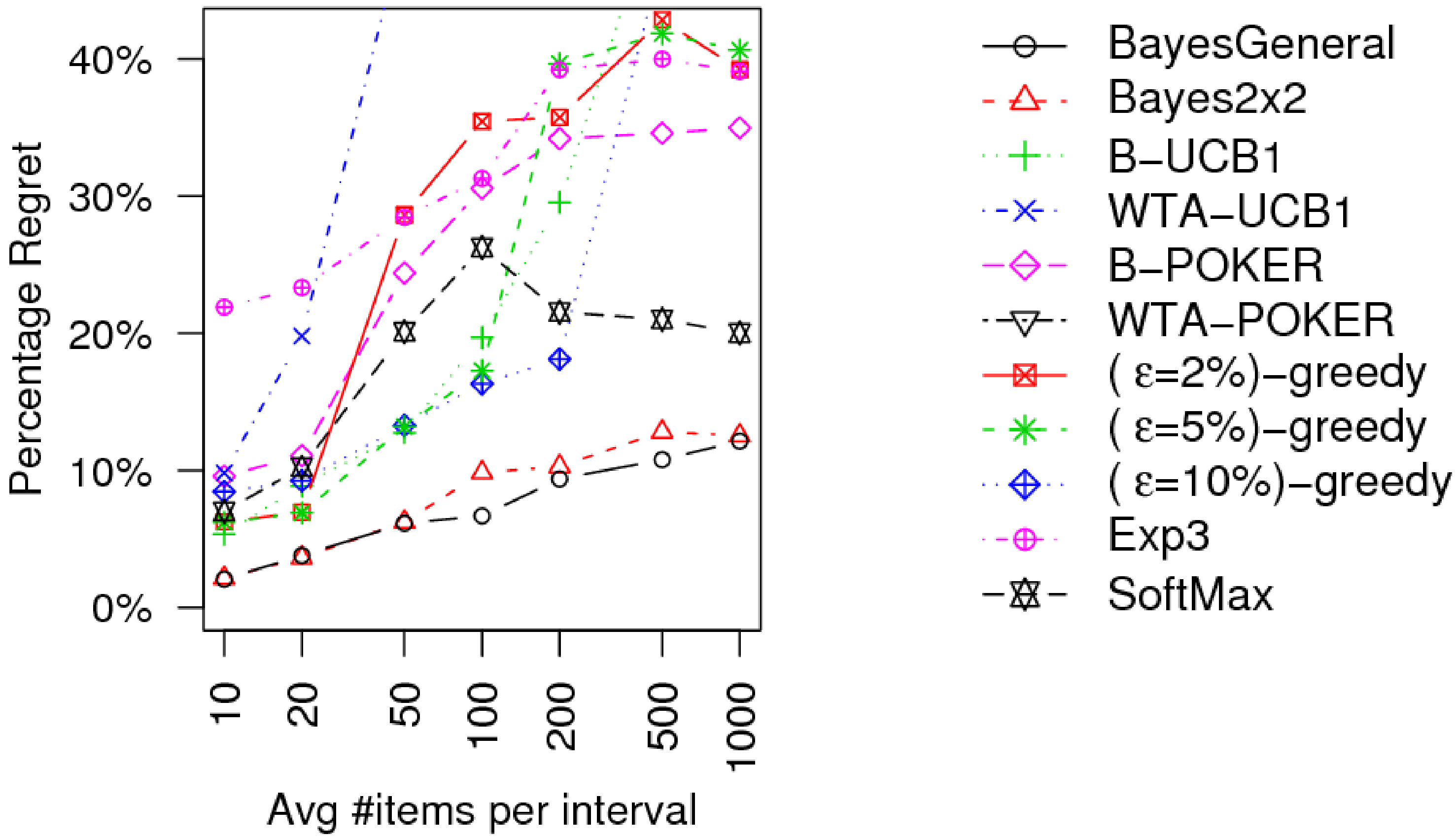
Simulation Experiment: Different Traffic Volume

- 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)



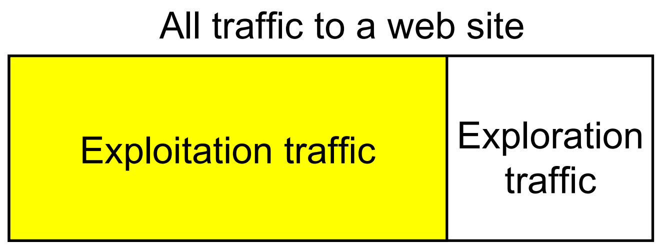
Simulation Experiment: Different Sizes of the Item Pool

- 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)



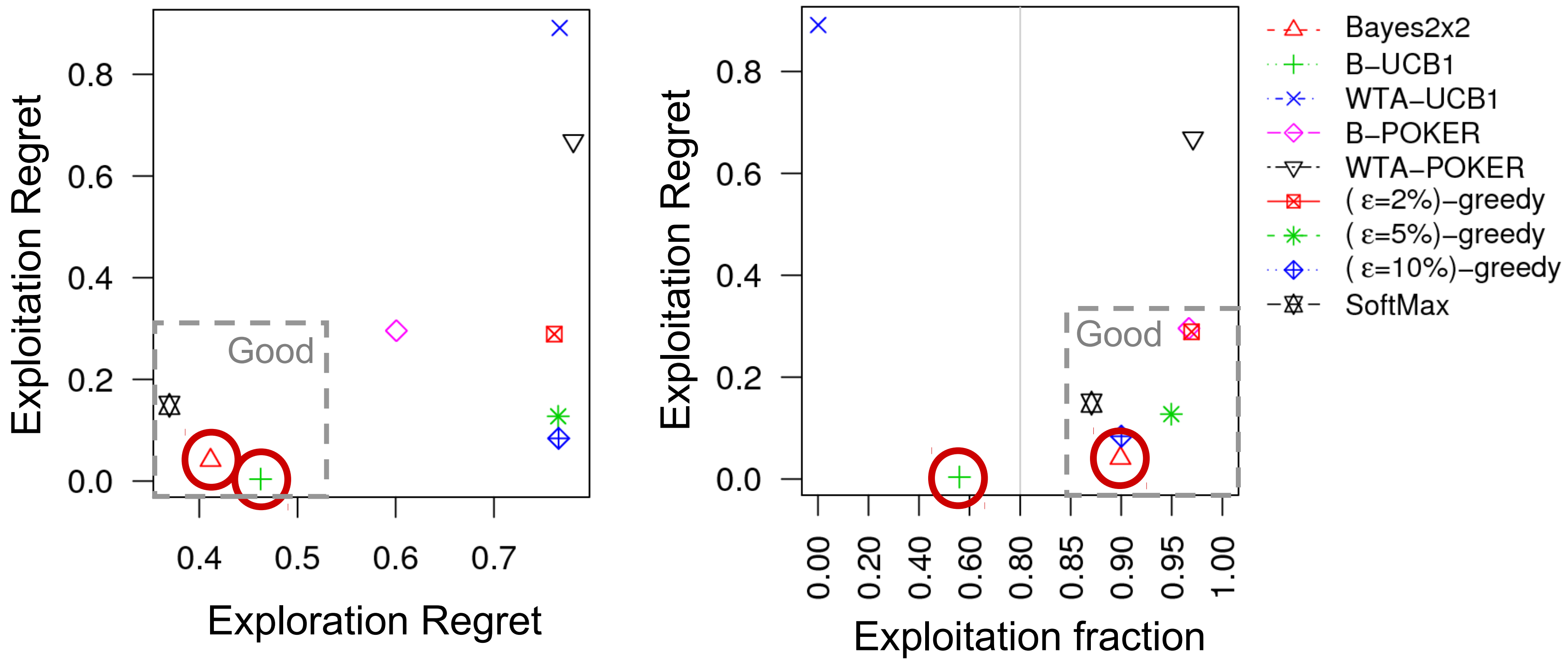
Characteristics of Different Explore/Exploit Schemes (1)

- 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 $\rightarrow$ better)
  - **Fraction of exploitation** (higher $\rightarrow$ 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)
  - Use item features to reduce the **prior variance** of an item
    - E.g., *Var*[item CTR | Sport] < *Var*[item CTR]
  - 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

  $\forall\ S_1, S_2$ and a, $f_a(S_1 \oplus S_2) \leq f_a(S_1)$

  - where $f_a(S) = f_a(S \oplus \langle a \rangle) - f_a(S)$
  - 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 $\times$ 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)
    - First explore/exploit categories/segments
    - Then, switch to individual items

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 $\rightarrow$ 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
  - ε-Greedy: Empirically competitive for tuned ε

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, tensor factorization
    - timeSVD++ [Koren 2009] for Netflix, (not really online)
- 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])

$$corr(i,j) = \frac{\overbrace{\sum_k (r_{ik} - \bar{r}_i)(r_{jk} - \bar{r}_j)}^{B_{ij}}}{\sqrt{\underbrace{\sum_k (r_{ik} - \bar{r}_i)}_{C_i}}\sqrt{\underbrace{\sum_k (r_{jk} - \bar{r}_j)}_{D_j}}}$$

Incrementally maintain three sets of counters: *B*, *C*, *D*

  - Clustering (e.g., [Das 2007])
    - MinHash (for Jaccard similarity)
    - Clusters(user *i*) = $(h_1(\mathbf{r}_i), \ldots, h_K(\mathbf{r}_i))$ ← fixed online (rebuilt periodically)
    - AvgRating(cluster *c*, item *j*) ← updated online

$$score(\text{user } i, \text{item } j) \propto \sum_k AvgRating(h_k(\mathbf{r}_i), j)$$

- Item-item methods (similar ideas)

Online Update for PLSI

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

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

Fixed online
(rebuilt Periodically)

Updated online

$$\frac{\sum_{\text{user } u} I(u \text{ clicks } j)\, p(k \mid u)}{\sum_{\text{user } u} p(k \mid u)}$$

Online Update for Linear/Factorization Model

- Linear model:

the regression weight of item *j* on the *k*th user feature

$$y_{ij} \sim \sum_k x_{ik}\beta_{jk} = x_i'\beta_j$$

rating that user *i* gives item *j*

the *k*th feature of user *i*

  - x_i can be user factor vector (estimated periodically, fixed online) β_j is an item factor vector (updated online)
  - Straightforward to fix item factors and update user factors

- Gaussian model (use vector notation)

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

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

$E[\beta_j]$ and $Var[\beta_j]$ (current estimates)

Update

$$E[\beta_j \mid y] = Var[\beta_j \mid y](V_j^{-1}\mu_j + \sum_i y_{ij}x_i / \sigma^2)$$

$$Var[\beta_j \mid y] = (V_j^{-1} + \sum_i x_i x_i' / \sigma^2)^{-1}$$

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

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

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

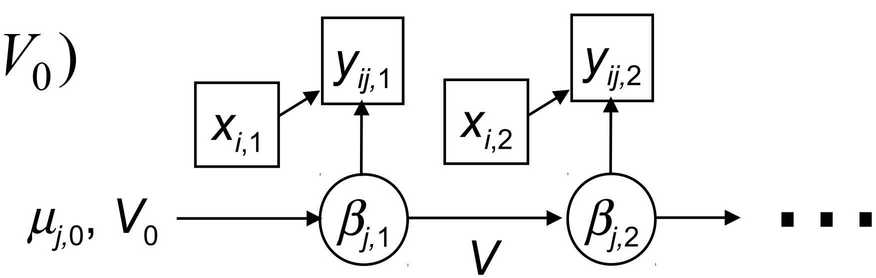
random variable

$$\beta_{j,1} \sim N(\mu_{j,0}, V_0)$$

Static model

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

$$\beta_j \sim N(\underbrace{\mu_j, V_j}_{\text{constants}})$$



  - 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: 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

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

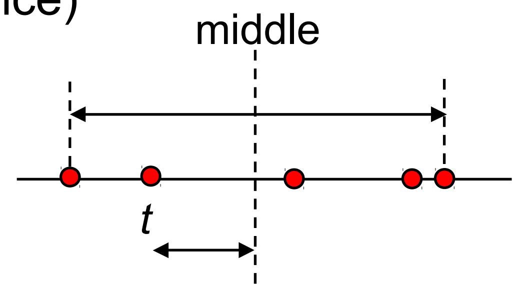
Subscript:
user *i*,
item *j*
time *t*

Temporal Dynamics: timeSVD++

- Explicitly model temporal patterns on historical data to remove bias
- Part of the winning method of Netflix contest [Koren 2009]

$$y_{ij,t} \sim \mu + \underbrace{b_i(t)}_{\text{user bias}} + \overbrace{b_j(t)}^{\text{item popularity}} + \underbrace{u_i(t)}_{\text{user factors (preference)}}' v_j$$

$$b_i(t) = b_i + \alpha_i \underbrace{\mathrm{dev}_i(t)}_{\text{distance to the middle rating time of } i} + b_{it}$$



$$b_j(t) = b_j + b_{j,\underbrace{\mathrm{bin}(t)}_{\text{time bin}}}$$

$$u_i(t)_k = u_{ik} + \alpha_{ik}\,\mathrm{dev}_u(t) + u_{ikt}$$

Model parameters: μ, b_i, α_i, b_{it}, b_j, b_{jd}, u_{ik}, α_{ik}, u_{ikt},
for all user *i*, item *j*, factor *k,* time *t*, time bin *d*

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)

- Linear/factorization model

rating that user *i* gives item *j*

factor vector of item *j*

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

feature/factor vector of user *i*

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

- How to estimate the prior parameters μ_j and Σ
  - 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

Subscript: user *i*, item *j*

FOBFM: Fast Online Bilinear Factor Model

Per-item online model $y_{ij} \sim u_i'\beta_j, \quad \beta_j \sim N(\mu_j, \Sigma)$

Subscript:
user i
item j
Data:
y_{ij} = rating that user i gives item j
u_i = offline factor vector of user i
x_j = feature vector of item j

- Feature-based model initialization

$$\beta_j \sim N(\underbrace{Ax_j}_{\text{predicted by features}}, \Sigma) \quad \dashv \quad y_{ij} \sim u_i'Ax_j + u_i'v_j$$

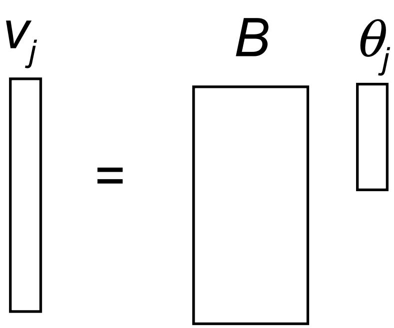
$$v_j \sim N(0, \Sigma)$$

- Dimensionality reduction for fast model convergence

$$v_j = B\theta_j$$

$$\theta_j \sim N(0,\ \sigma_\theta^2 I)$$

B is a $n \times k$ linear projection matrix ($k << n$)
project: high dim(v_j) $\rightarrow$ low dim(θ_j)
low-rank approx of Var[β_j]: $\beta_j \sim N(Ax_j,\ \sigma_\theta^2 BB')$



Offline training: Determine A, B, σ_θ^2 through the EM algorithm (once per day or hour)

FOBFM: Fast Online Bilinear Factor Model

Per-item online model $y_{ij} \sim u_i' \beta_j, \quad \beta_j \sim N(\mu_j, \Sigma)$

Subscript:
user i
item j
Data:
y_{ij} = rating that user i gives item j
u_i = offline factor vector of user i
x_j = feature vector of item j

- Feature-based model initialization

$$\beta_j \sim N(\underbrace{Ax_j}_{\text{predicted by features}}, \Sigma) \quad \Leftrightarrow \quad y_{ij} \sim u_i' A x_j + u_i' v_j, \quad v_j \sim N(0, \Sigma)$$

- Dimensionality reduction for fast model convergence

$$v_j = B\theta_j$$

$$\theta_j \sim N(0, \sigma_\theta^2 I)$$

B is a $n \times k$ linear projection matrix ($k << n$)
project: high dim(v_j) $\rightarrow$ low dim(θ_j)
low-rank approx of Var[β_j]: $\beta_j \sim N(Ax_j, \sigma_\theta^2 BB')$

- Fast, parallel online learning

$$y_{ij} \sim \underbrace{u_i' A x_j}_{\text{offset}} + \underbrace{(u_i' B)}_{\text{new feature vector (low dimensional)}} \theta_j$$

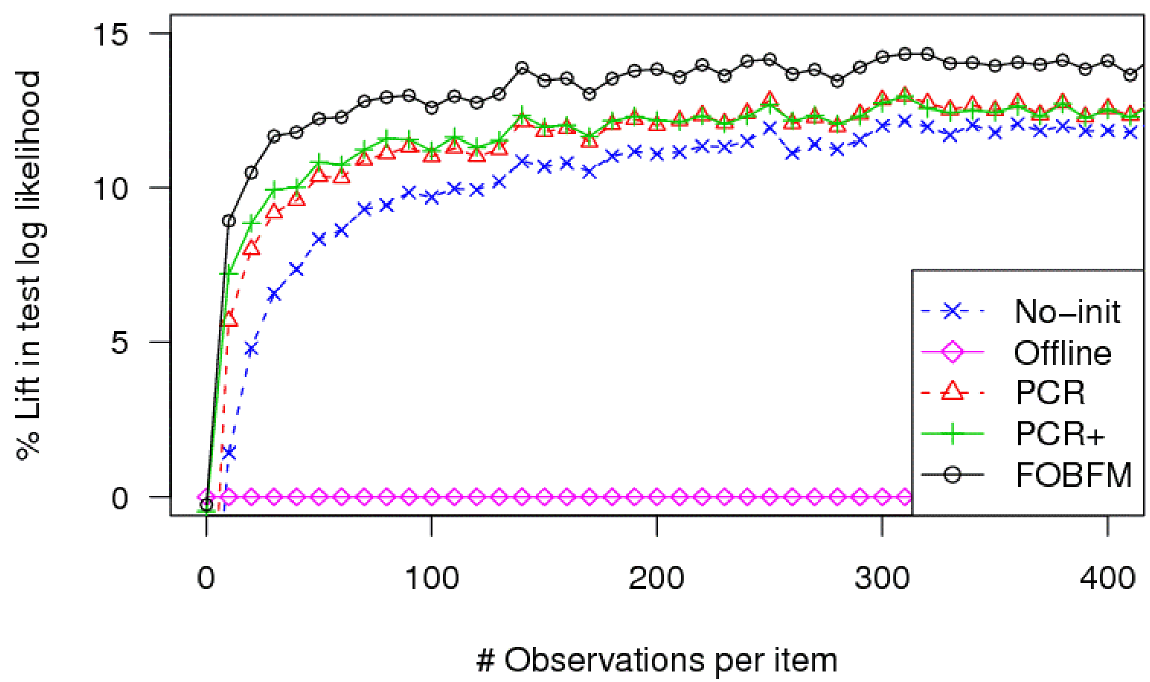
, where θ_j is updated in an online manner

- Online selection of dimensionality ($k = \dim(\theta_j)$)
  - Maintain an ensemble of models, one for each candidate dimensionality

Experimental Results: My Yahoo! Dataset (1)

- My Yahoo! is a personalized news reading site
  - Users manually select news/RSS feeds
- ~12M “ratings” from ~3M users on ~13K articles
  - Click = positive
  - View without click = negative

Experimental Results: My Yahoo! Dataset (2)

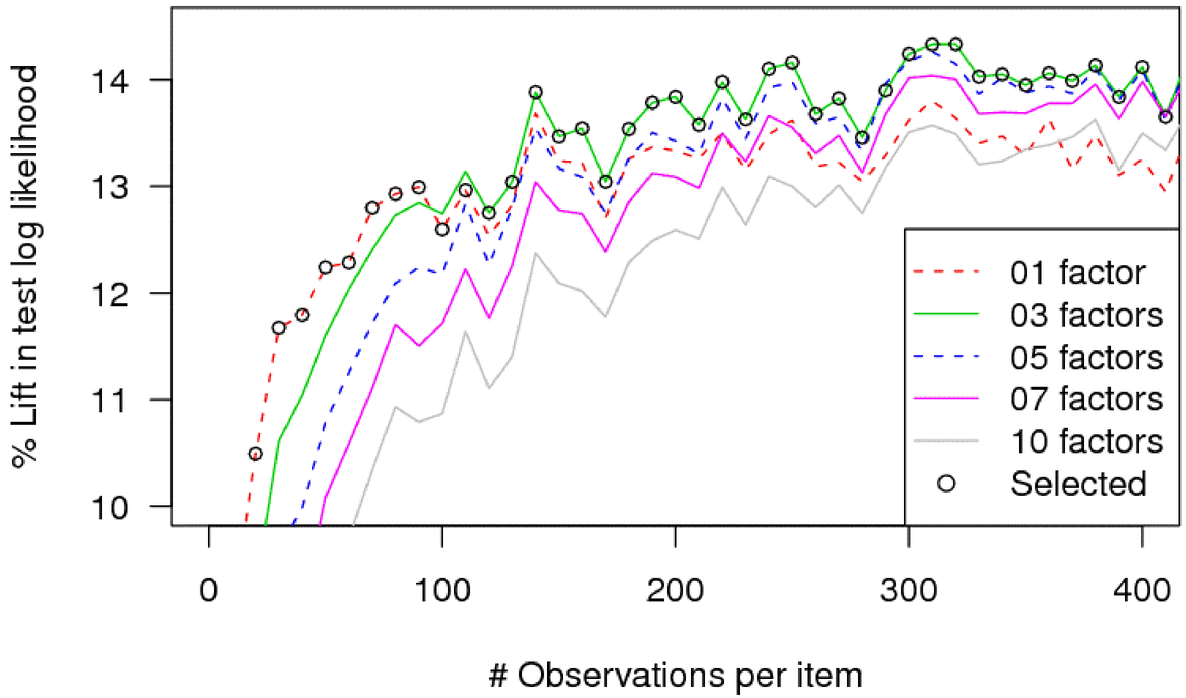


Methods:

- **No-init:** Standard online regression with ~1000 parameters for each item
- **Offline:** Feature-based model without online update
- **PCR, PCR+:** Two principal component methods to estimate *B*
- **FOBFM:** Our fast online method

- Item-based data split: Every item is new in the test data
  - First 8K articles are in the training data (offline training)
  - Remaining articles are in the test data (online prediction & learning)
- Supervised dimensionality reduction (reduced rank regression) significantly outperforms other methods

Experimental Results: My Yahoo! Dataset (3)



factors =
Number of parameters per item updated online

- Small number of factors (low dimensionality) is better when the amount of data for online leaning is small
- Large number of factors is better when the data for learning becomes large
- The online selection method usually selects the best dimensionality

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
  - Covariate bandits

Explore/Exploit for Personalized Recommendation

- One extreme problem formulation
  - One bandit problem per user with one arm per item
  - Bandit problems are correlated: “Similar” users like similar items
  - Arms are correlated: “Similar” items have similar CTRs
- Model this correlation through covariates/features
  - Input: User feature/factor vector, item feature/factor vector
  - Output: Mean and variance of the CTR of this (user, item) pair based on the data collected so far
- Covariate bandits
  - Also known as contextual bandits, bandits with side observations
  - Provide a solution to
    - Large content pool (correlated arms)
    - Personalized recommendation (hint before pulling an arm)

Methods for Covariate Bandits

- Priority-based methods
  - Rank items according to the user-specific "score" of each item; then, update the model based on the user's response
  - **UCB** (upper confidence bound)
    - Score of an item = *E*[posterior CTR] + *k StDev*[posterior CTR]
  - **Posterior draw**
    - Score of an item = a number drawn from the posterior CTR distribution
  - **Softmax**
    - Score of an item = a number drawn according to $\frac{\exp\{\hat{\mu}_i/\tau\}}{\sum_j \exp\{\hat{\mu}_j/\tau\}}$

∀ ε-Greedy
  - Allocate ε fraction of traffic for random exploration (ε may be adaptive)
  - Robust when the exploration pool is small

- Bayesian scheme
  - Close to optimal if can be solved efficiently

Covariate Bandits: Some References

- Just a small sample of papers
  - Hierarchical explore/exploit (Pandey et al., 2008)
    - Explore/exploit categories/segments first; then, switch to individuals
  - Variants of ε-greedy
    - Epoch-greedy (Langford & Zhang, 2007): ε 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): ε decreases over time; example model: histogram, nearest neighbor
  - Variants of UCB methods
    - Linearly parameterized bandits (Rusmevichientong et al., 2008): minimax, based on uncertainty ellipsoid
    - LinUCB (Li et al., 2010): Gaussian linear regression model
    - Bandits in metric spaces (Kleinberg et al., 2008; Slivkins et al., 2009):
      - Similar arms have similar rewards: $|\ reward(i) - reward(j)\ | \leq distance(i,j)$

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

Backup Slides

Intelligent Initialization for Factorization Model (1)

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

Offline model

$$y_{ij} \sim N(\underbrace{u_i' v_j}_{\text{Factorization}},\ \sigma^2 I)$$

$$u_i \sim N(\underbrace{Gx_i}_{\text{Feature-based init}},\ \sigma_u^2 I)$$

$$v_j = \underbrace{Ax_j}_{\text{Feature-based init}} + \overbrace{B\theta_j}^{\text{Dim reduction}}$$

$$\theta_j \sim N(0,\ \sigma_\theta^2 I)$$

(periodic) offline training output: u_i, A, B, σ_θ^2 →

Online model

$$y_{ij,t} \sim N(\overbrace{u_i' Ax_j}^{\text{Offset}} + \overbrace{u_i' B}^{\text{Feature vector}}\theta_{j,t},\ \sigma^2 I)$$

Updated online

$$\theta_{j,t} = \theta_{j,t-1}$$

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

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

$$y_{ij} \sim N(u_i' v_j,\ \sigma^2 I)$$

$$u_i \sim N(G x_i,\ \sigma_u^2 I)$$

$$v_j = A x_j + B\theta_j$$

$$\theta_j \sim N(0,\ \sigma_\theta^2 I)$$

Online

$$y_{ij,t} \sim N(u_i' A x_j + u_i' B\theta_{j,t},\ \sigma^2 I)$$

$$\theta_{j,t} = \theta_{j,t-1}$$

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

- Our observation so far
  - Dimension reduction ($u_i'B$) does not improve much if factor regressions are based on good covariates (σ_θ^2 is small)
    - Small σ_θ^2 $\rightarrow$ strong shrinkage $\rightarrow$ 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)

$$y_{ij,t} \sim N(u'_{i,t} v_{j,t},\ \sigma^2 I)$$

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

$$\delta_{i,t} \sim N(\delta_{i,t-1},\ \sigma_\delta^2 I)$$

$$\delta_{i,1} \sim N(0,\ s_\delta^2 I)$$

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

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

$$\theta_{j,1} \sim N(0,\ s_\theta^2 I)$$

Online computation

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

$$y_{ij,t} \sim N(u'_{i,t} Dx_{j,t} + u'_{i,t} B\theta_{j,t},\ \sigma^2 I)$$

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

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

$$y_{ij,t} \sim N(v'_{j,t} Gx_{i,t} + v'_{j,t} H\delta_{i,t},\ \sigma^2 I)$$

$$\delta_{i,t} \sim N(\delta_{i,t-1},\ \sigma_\delta^2 I)$$

Repeat the above two steps a few times

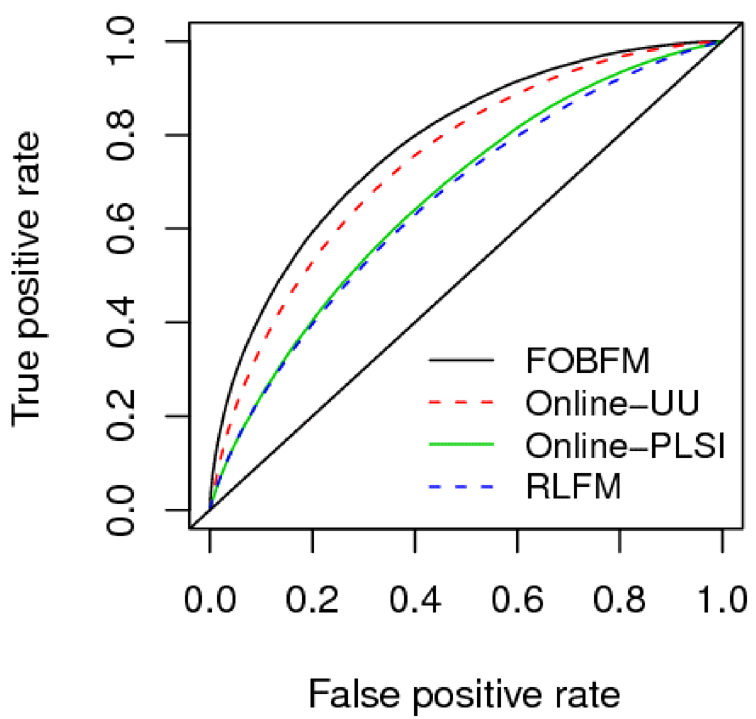
Experimental Results: MovieLens Dataset

- Training-test data split
  - Time-split: First 75% ratings in training; rest in test
  - Movie-split: 75% randomly selected movies in training; rest in test

Model	RMSE Time-split	RMSE Movie-split
FOBFM	0.8429	0.8549
RLFM	0.9363	1.0858
Online-UU	1.0806	0.9453
Constant	1.1190	1.1162

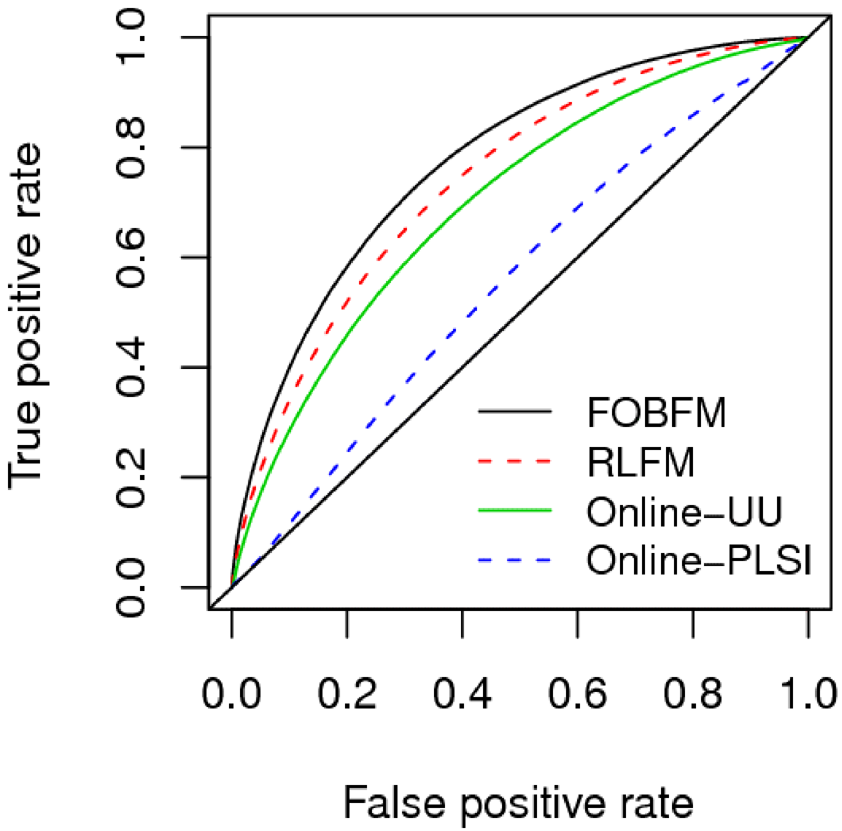
FOBFM: Our fast online method
RLFM: [Agarwal 2009]
Online-UU: Online version of user-user collaborative filtering
Online-PLSI: [Das 2007]

ROC for Movie-split



Experimental Results: Yahoo! Front Page Dataset

- Training-test data split
  - Time-split: First 75% ratings in training; rest in test



–~2M “ratings” from ~30K frequent users to ~4K articles

- Click = positive
- View without click = negative

–Our fast learning method outperforms others

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*[CTR | data] + *k Std*[CTR | 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