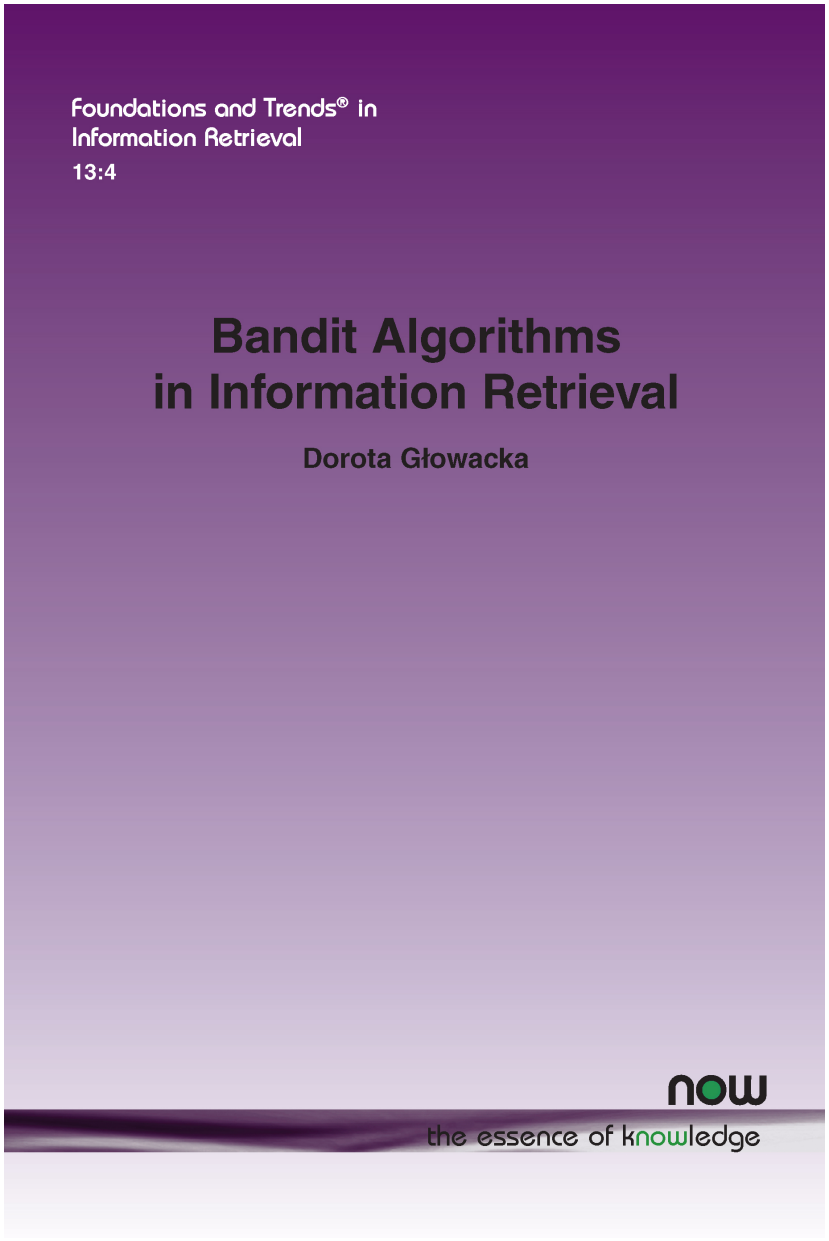
Bandits in Recommender Systems

Dorota Glowacka
University of Helsinki
glowacka@cs.helsinki.fi



Free dowload this week only!

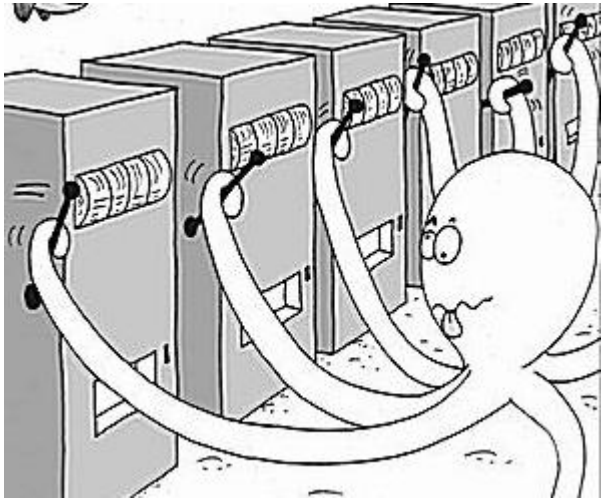
https://nowpublishers.com/article/Details/INR-067

Slides available at:
https://glowacka.org/teaching.html

The Multi-Armed Bandit Problem

You are in a casino. There are many different slot machines (known as 'one-armed bandits', as they are know for robbing you), each with a lever (an arm). You think that some slot machines pay out more than others, so you'd like to maximize this. You only have a limited amount of resources – you can pull each arm a limited number of times. Your goal is to walk out of the casino with most money.

If you knew which lever would pay out the most, you would pull that lever all day.



The question is: *how do you learn which slot machine is the best and get the most money in the shortest amount of time?*

Explore – Exploit Dilemma

You have no initial knowledge about the payoff of the machines and so at each trial you face the following trade-off:

- Play the machine that has given you the highest reward so far (***exploit***)
- Play other machines in the hope of finding one that will give you a higher reward (***explore***)

Exploration – Exploitation Dilemma

- Online decision making involves a fundamental choice:

Exploitation: make the best decision given current information

Exploration: gather more information

- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decision

Examples

- Online Advertising

Exploit: Show the most successful advert

Explore: Show a new advert

- Restaurant Selection:

Exploit: Go to your favourite restaurant

Explore: Try a new restaurant

- Oil drilling

Exploit: Drill at the best known location

Explore: Drill at a new potential oil field

Practical Applications

- Dynamic allocation of resources (which project to work on given the uncertainty about the difficulty and payoff of each project)
- Clinical trials (investigating effects of different experimental treatments while minimizing patient loss)
- Financial portfolio design
- Adaptive routing (to minimize delays in the network)

How to Explore?

- Naïve Exploration

  Add noise to a greedy policy
- Optimistic Initialization

  Assume the best until proven otherwise
- Optimism in the Face of Uncertainty

  Select actions with uncertain values
- Probability Matching

  Select actions according to probability they are best

The Multi-Armed Bandit

- A multi-armed bandit is a tuple $\langle A, R \rangle$
- *A* is a known set of *m* actions (or arms)
- $R^a(r) = \mathrm{P}[r \mid a]$ is an unknown probability distribution over rewards
- At each step *t*, agent selects an action $a_t \in A$
- The environment generates a reward r_t
- The goal is to maximize cumulative reward $\sum_{\tau=1}^{t} r_\tau$

Regret

The *regret* is the difference between the sum of *rewards r obtained so far* and the reward sum associated with *optimal strategy*.

Let $\mu_1, \ldots, \mu_m$ be the mean values associated with the rewards of each arm.

The regret after *t* rounds is:

$$\rho = t\mu^* - \sum_{\tau=1}^{t} r_\tau$$

where μ^* is the maximum reward mean.

Approaches to the Bandit Problem

- Regret is defined in terms of the average reward.
- If we can estimate average reward, then we can minimize regret.
- Let's take the action with the highest average reward:
  - Assume two actions (arms)
  - Action (arm) 1 has reward of 1 with probability 0.3 and otherwise the reward is 0
  - Action (arm) 2has reward of 1 with probability 0.7 and otherwise has reward 0f 0
  - We play the first arm and get reward of 1
  - Next, we play the second arm and get reward 0
  - Now the average reward of arm 1 is higher than that of arm 2.

Greedy Algorithm

- After playing each arm once and observing the reward, we might conclude that the arm 1 gives us a better reward and so play for the rest of the game.
- The greedy algorithm selects arm with the highest value:

$$a_t^* = \underset{a \in A}{\operatorname{argmax}}\, \mu_t(a)$$

- The greedy algorithm can lock onto a suboptimal action forever and exploit it forever.

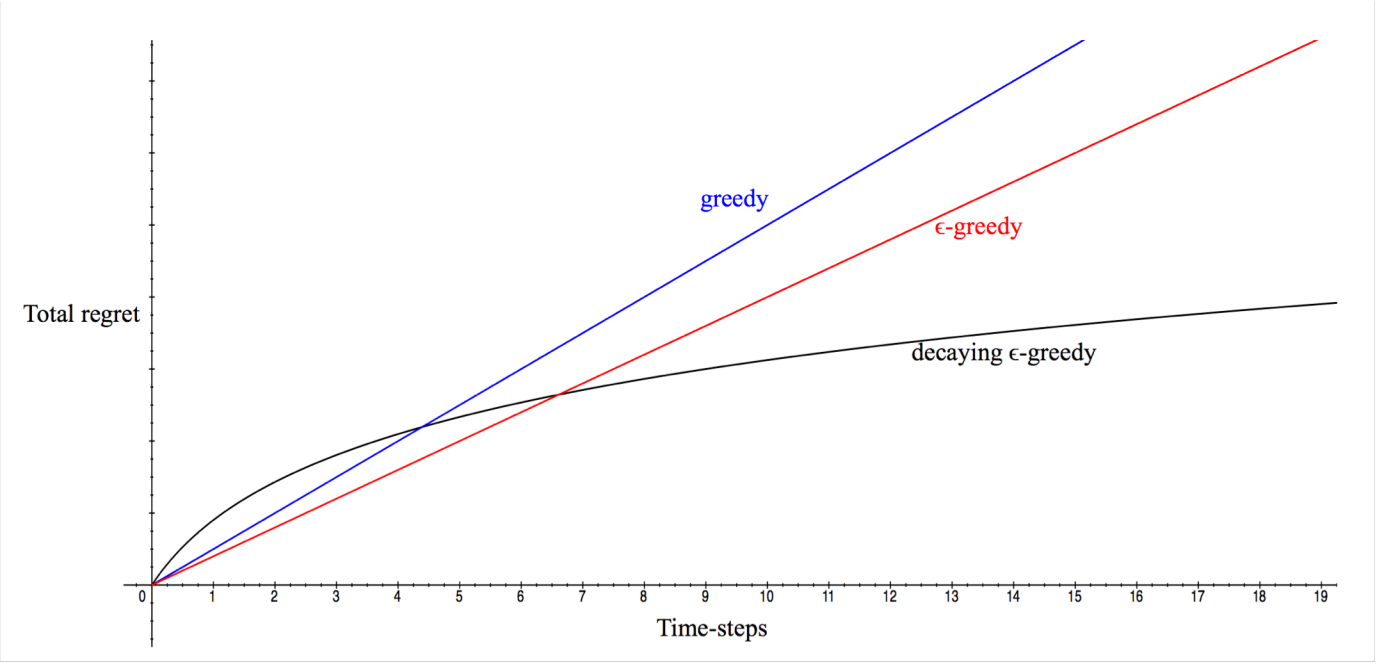
Epsilon-Greedy Algorithm

- With probability 1 –ε select $a = \underset{a \in A}{\operatorname{argmax}}\, \rho(a)$
- With probability *ε* select a random action.
- Typically *ε = 0.1* but this may vary depending on the problem at hand.
- Constant ε ensures minimum regret

$$\rho_t \geq \frac{\varepsilon}{A} \sum_{a \in A} \Delta_a$$

- The *ε*-greedy algorithm continues to explore forever, which means it has a linear total regret.

Regret



If you *only* explore, then the regret will be linear.
If you *never* explore, then the regret will be linear.
Is it possible to achieve sublinear total regret?

Sublinear Regret

- Both the greedy algorithm and the ε-greedy algorithm have linear regret.
- Is it possible to achieve a *sublinear* total regret?
- Is it possible to explore a lot in the initial stage and then gradually move to exploitation as the game progresses?

Decaying ε_t-Greedy Algorithm

- Pick a decay schedule for $\varepsilon_1, \varepsilon_2, \ldots$
- Consider the following schedule:

$$c > 0$$

$$d = \min_{a|\Delta_a > 0} \Delta_a$$

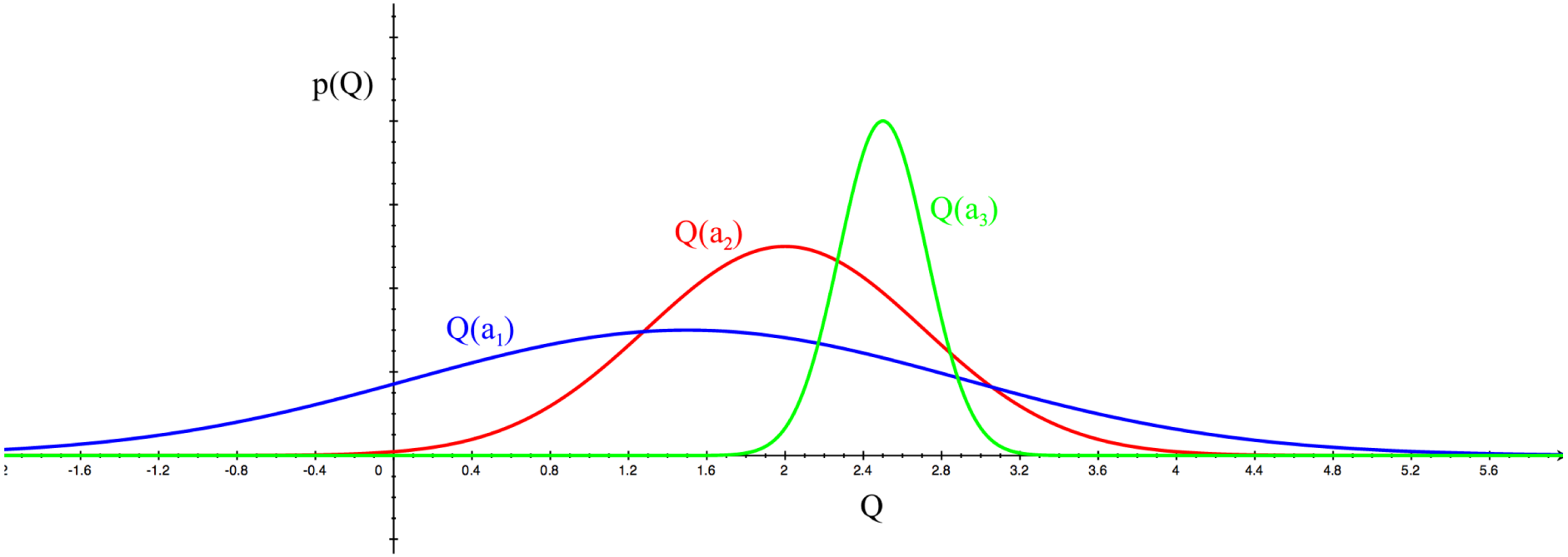
$$\varepsilon_t = \min\left\{1, \frac{c\,|A|}{d^2 t}\right\}$$

- Decaying ε_t-greedy has logarithmic total regret.
- Unfortunately, schedule requires advance knowledge of gaps.
- Can we find an algorithm with sublinear regret without knowledge of gaps?

Optimism in Face of Uncertainty

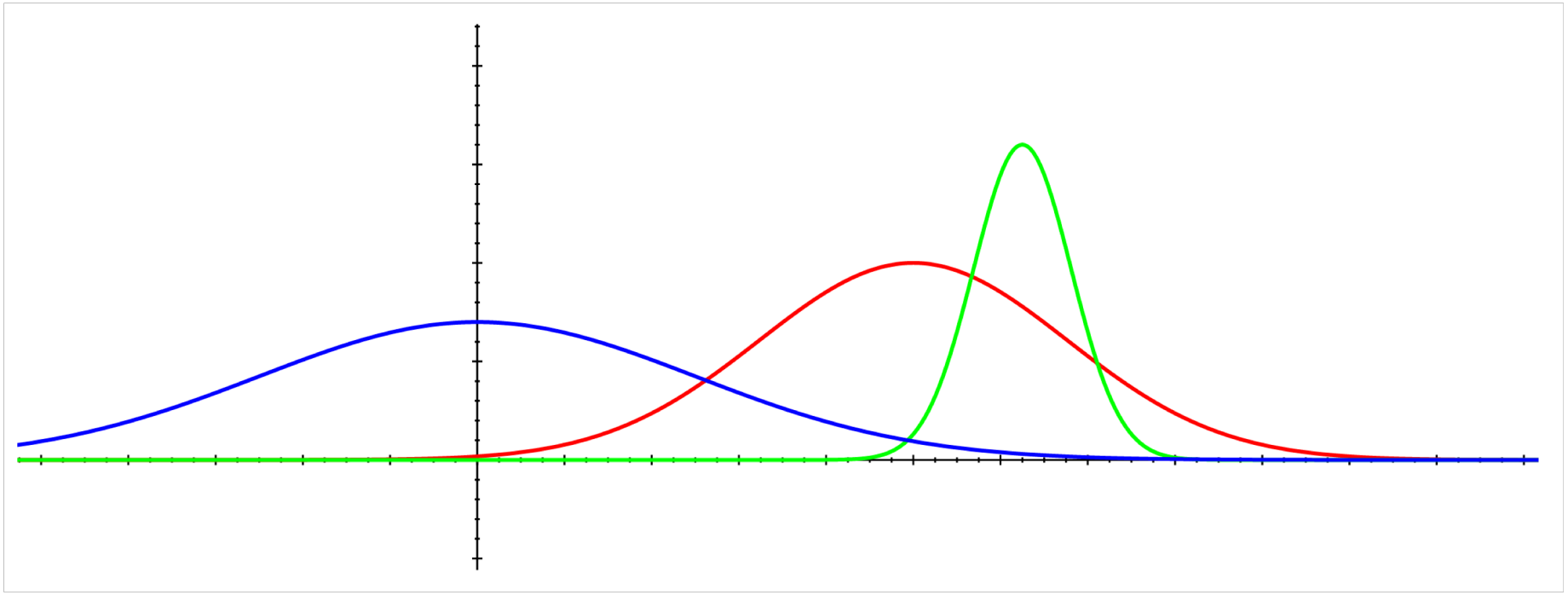
- The problem with the greedy algorithm is that it is *too certain* of its estimates – we should not conclude what the average reward of a given arm is based on one trial.
- The more *uncertain* we are about the reward of an action (arm), the more important it is to *explore* that action.
- It could turn out to be the best action!

Optimism in Face of Uncertainty



Which action should we choose?

Optimism in Face of Uncertainty



After observe reward of the blue action, we are less uncertain about its value.

Confidence Bounds

- Instead of greedily selecting an action (arm) based on one observed reward, we should assume that the average reward lies within some *confidence interval* that we can adjust based on the information that we have.
- A *confidence interval* is a range of values within which we are sure the mean lies with a certain probability.
- If we tried an action less often, then our estimated reward is less accurate so the confidence interval is larger. It shrinks as we get more information (try an action more often).

Lower Bound

- The performance of an algorithm is determined by *similarity* between optimal arm and other arms.
- Hard problems have *similar-looking* arms with *different means*.
- This is formally described by the gap Δ_a and the similarity in distributions $KL(R^a \| R^{a^*})$

Theorem (Lai and Robbins)

Asymptotic total regret is at least logarithmic in number of steps

$$\lim_{t\to\infty} L_t \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(R^a \| R^{a^*})}$$

Upper Confidence Bound

- Estimate upper confidence $U_t(a)$ for each action value
- Such that $\mu(a) \leq \mu_t(a) + U_t(a)$ with high probability
- This depends on number of times $N(a)$ was selected:
  - *Small* $N_t(a)$ ➔ *large* $U_t(a)$ (estimated value is *uncertain*)
  - *Large* $N_t(a)$ ➔ *small* $U_t(a)$ (estimated value is *accurate*)
- Select action maximizing *Upper Confidence Bound* (UCB)

$$a_t = \underset{a \in A}{\operatorname{argmax}}\, \mu_t(a) + U_t(a)$$

Hoeffding's Inequality

- How do we calculate the upper confidence bound?

Theorem (Hoeffding's Inequality)

Let $X_1, \ldots, X_t$ be i.i.d. random variables in [0, 1], and let $\bar{X}_t = \frac{1}{\tau}\sum_{\tau}^{t} X_\tau$ be the sample mean. Then

$$\mathrm{P}[\mathrm{E}[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

Calculating Upper Confidence Bound

- We will apply Hoeffding's Inequality to rewards of the bandit conditioned on selecting action *a*

$$\mathrm{P}[\mu(a) > \hat{\mu}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$$

- Pick a probability *p* with *true value* exceeding UCB
- Now solve for $U_t(a)$

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

Calculating Upper Confidence Bound

- Reduce *p* as we observe more rewards, e.g. $p = t^{-4}$
- Ensures that we select optimal actions as $t \rightarrow \infty$

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}}$$

UCB1

- This leads to the UCB1 algorithm

$$a_t = \underset{a \in A}{\operatorname{argmax}}\, \mu(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

- The confidence bound grows with the total number of actions taken but shrinks with the number of times a particular action has been tried.
- This ensures that each action is tried infinitely often but still balances exploration and exploitation.

UCB1

- For each action *a* record the average reward *μ(a)* and the number of times we have tried it *N(a)*. We write *t* for the total number of actions we have tried so far.
- Try the action that maximizes $\mu(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$
- It is quick and easy to implement.

UCB1 Regret Bound

- The UCB algorithm achieves logarithmic total regret.
- At time *t*, the expected total regret is at most:

$$8 \log T \sum_{a | \Delta_a < \Delta_{a^*}} \frac{1}{\Delta_a} + \left(1 + \frac{\pi^2}{3}\right) \sum_{a \in A} \Delta_a$$

where

$$\Delta_a = \mu^* - \mu_a$$

P.Auer, N. Cesa-Bianchi, P. Fisher: *Finite-Time Analysis of the Multiarmed Bandit Problem*. Machine Learning 47 (2), 2002.

UCB-Tuned

- In practice we can improve the performance of UCB by replacing its upper confidence bound with:

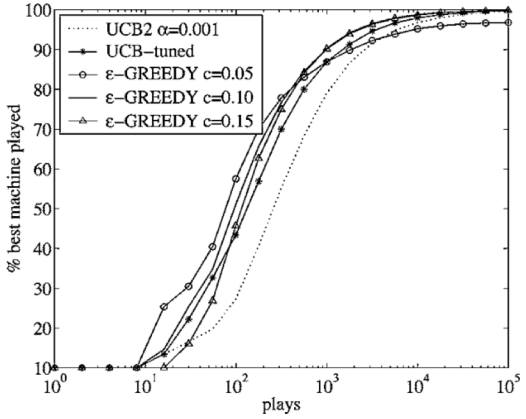
$$\sqrt{\frac{\log t}{N_t(a)} \min\left\{\frac{1}{4}, \left(\sigma_a + \frac{2\log t}{N_t(a)}\right)\right\}}$$

σ_a is the sample variance for each action a

the factor ¼ is an upper bound on the variance of any [0, 1] bounded variable

- UCB-Tuned is not very sensitive to large differences in response rates

Example: UCB vs. ε-Greedy on 10-arm Bandit



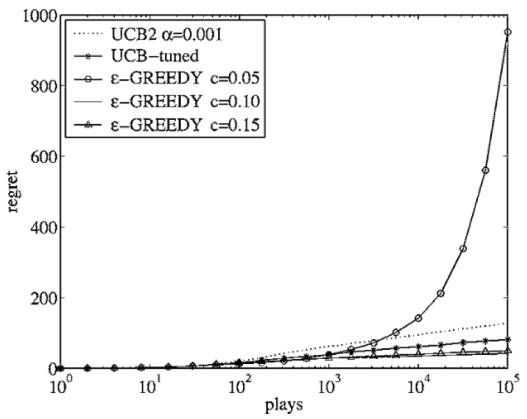
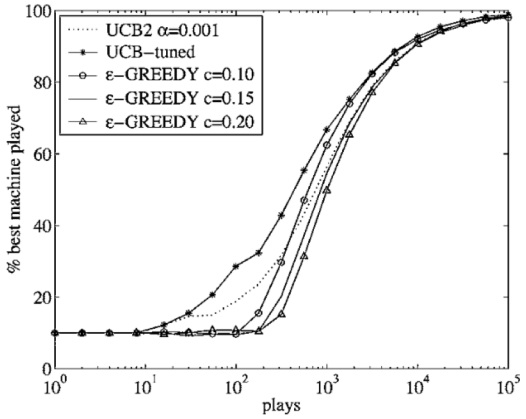


Figure 9. Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, . . . , 0.6).



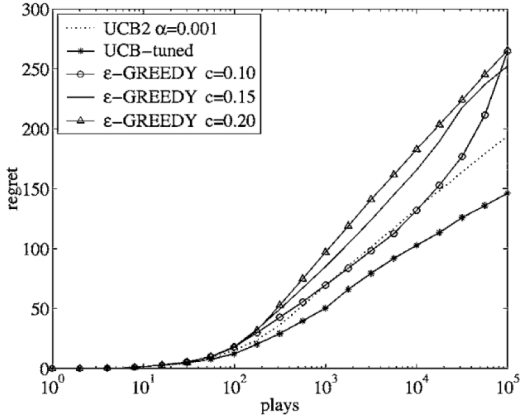


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

Linear Bandits

- What happens if the number of arms is very large and we cannot try all of them?
- Take advantage of similarity between arms, i.e. playing one arm will give you information about similar arms thus reducing the amount of exploration required.
- The assumption is that there is a similarity structure between arms.
- Each arm is represented as a feature vector.

Linear Bandits

- Consider m arms, *m >> T*, where every arm *a* is represented as a vector x_a
- On pulling arm *a* at time *t*, we observe reward

$$r_t = x_a^{\mathrm{T}} \omega$$

Example:

$$x_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, x_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, x_3 = \begin{pmatrix} 0 \\ 0.5 \end{pmatrix}, x_4 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Pulling arm 1 tells us: some information about pulling arm 2, everything about pulling arm 3, and nothing about pulling arm 4

Contextual Bandits

- In each round:

(1). **Observe** context vector x_a of each arm $a \in \mathcal{A}$

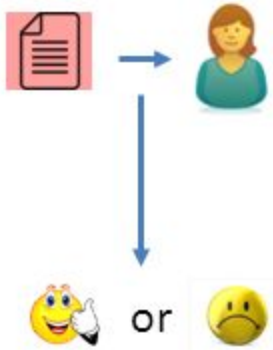
(2). **Picks** an arm a_t

Upper Confidence Bound

$$a_t = \arg max_a (\hat{r}_{a,t} + C_t(a))$$

Estimated reward (exploitation)

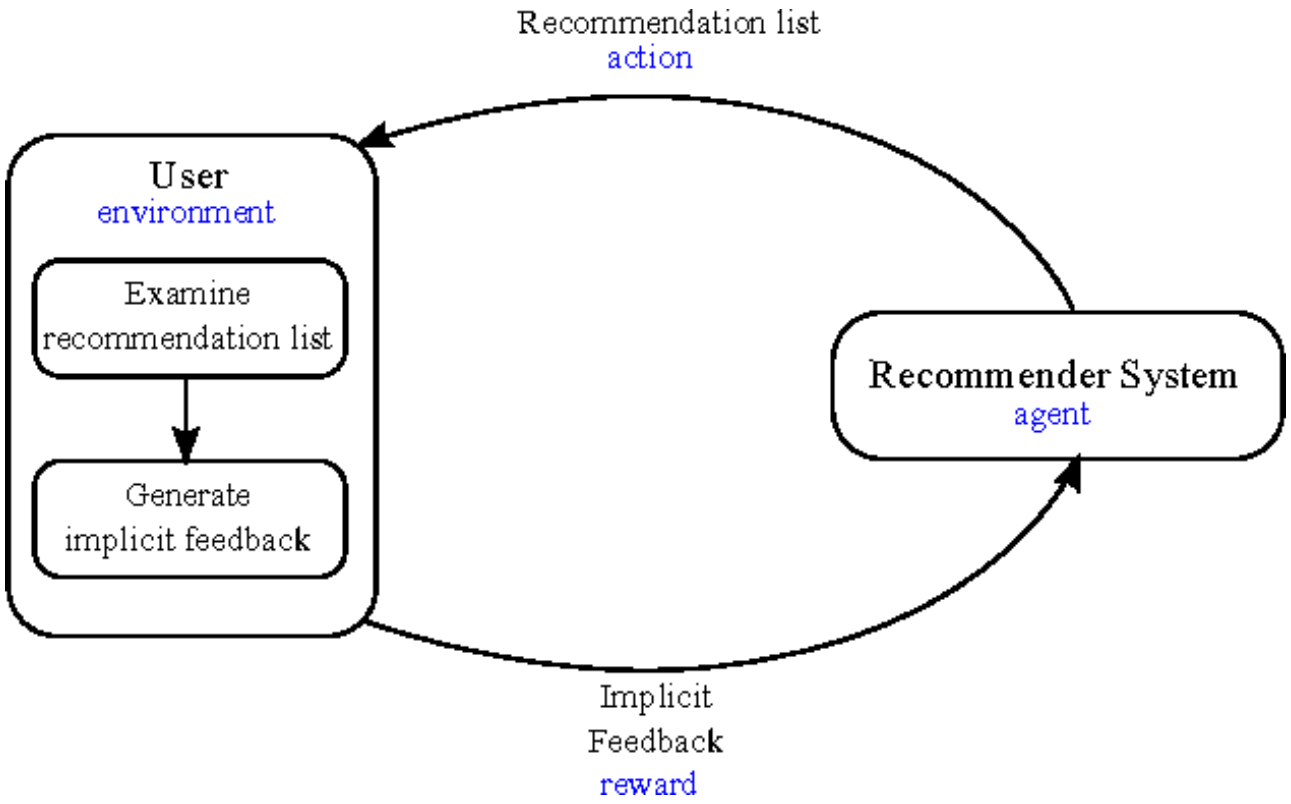
Confidence interval (exploration)



(3). **Receive** corresponding reward $r_{a_t,u}$

(4). **Update** model

Bandits and Recommendation



Diverse Rankings with Bandits

- *Probabilistic ranking* advocates ranking documents in order of *decreasing relevance* to a *query*.
- Result: *similar documents* ranked at *similar positions*, while *users* may prefer *diverse* set of results.

- Problem: users click on few results (increased *abandonment*)

Diverse Ranking

- We want to learn an optimally diverse ranking of documents for a given query and maximize user satisfaction.
- In each round, a user arrives and an algorithm outputs a list of *k* results. The user scans the documents top-down and clicks on the first relevant one.
- Each slot *i* is examined by the user only if the documents in the higher slots are not clicked.

Rank 1: ✗
Rank 2: ✗
Rank 3: ✗
Rank 4: ✗
Rank 5: ✓
STOP
Rank 6
…
…
Rank k:

Ranked Bandits

- Run a bandit for each rank
- Each bandit maintains a value for every document in collection
- Bandits corresponding to each rank are treated independently
- If B_j and B_i select the same document, then a random document is selected for rank j

B_1 – d_1, d_2, d_3, d_4, … d_n
B_2 – d_1, d_2, d_3, d_4, … d_n
B_k – d_1, d_2, d_3, d_4, … d_n

Rank 1: d_2 ✗
Rank 2: d_4 ✓
Rank k: d_1 ✗

B_1 – 0, 0, 0, 0, …, 0
B_2 – 0, 0, 0, **1**, …, 0
B_k – 0, 0, 0, 0, …, 0

Ranked Bandits Algorithm

- No relevance judgments from experts required for training.
- Accounts for dependencies between documents.
- The algorithms learns a utility value for each document at each rank, maximizing the probability that a new user of the search system will find at least one relevant document within the top *k* positions.
- Equivalent to an online learning problem with no distinction between training and testing phases.

R.Kleinberg, F. Radlinski, T. Joachims: *Learning Diverse Rankings with Multi-Armed Bandits*. ICML 2008.

What if the Number of Documents is Large?

- Bandit algorithms are ideal for online settings with exploration/exploitation trade-offs but are impractical at web scales.

- Exploit document *similarity* and ranking *context* to optimize the convergence rate of the bandit algorithm.

- To exploit the *context*, we can factor in *conditional clickthrough rates* (user skips a set of documents) and *correlated clicks* (probability that two documents are ir/relevant to the user).

Ranked Bandits in Metric Spaces

- *Document model*: web documents are organised into *tree* hierarchies, where *closer pairs* are *more similar* and each document *x* is a leaf in the tree.

- The tree is a topic taxonomy on documents such that the *click event* on each *subtopic* is obtained from that on the *parent topic* via probability mutation (distance between child and parent).

- A. Slivkins, F. Radlinski, S. Gollapudi: *Learning Optimally Diverse Rankings Over Large Document Collections*. ICML 2010.
- A. Slivkins, F. Radlinski, S. Gollapudi: *Ranked Bandits in Metric Spaces: Learning Diverse Rankings over Large Document Collections*. JMLR 14 (2013).

Extensions to Recommender Systems

- *Independent Bandit Algorithm* (IBA) (Kohli et al. 2013) – based on RBA with reward of 1 given to any clicked article (RBA gives reward of 1 to the first clicked article only)

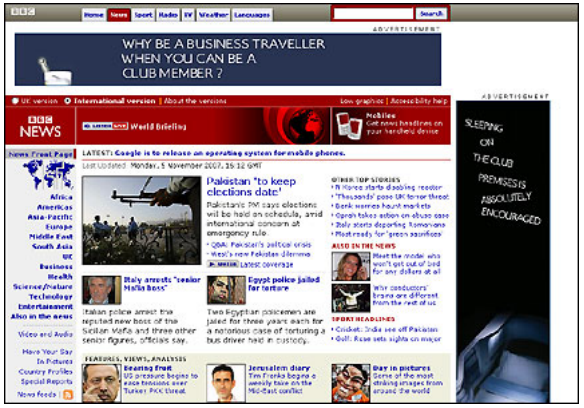
- *DynUCB* (Nguyen & Lauw 2014) – user population divided into multiple clusters based on their interests although no graph structure is used.

Ads and News Recommendation

1. L. Li et al. *A contextual-Bandit Approach to Personalized News Article Recommendation*. WWW 2010
2. O. Chapelle & L. Li. *An Empirical Evaluation of Thompson Sampling*. NIPS 2011

1. W. Li et al. *Exploration and Exploitation in a Performance based Contextual Advertising System.* KDD'10
2. L. Tang et al. *Personalized Recommendation via Parameter-Free Contextual Bandits*. SIGIR'15.

Which article to put on our website?



We have space for only one article on our website but three candidates.

If we had some features about our users, i.e. what type of articles they had clicked, the algorithm could take that into account to find best articles based on their past click behaviors.

arm	clicked_sports	clicked_politics	ct	reward	mean_clk_rt
1	1	0	963	560	0.5815161
1	1	1	392	271	0.6913265
2	0	0	1450	286	0.1972414
3	0	1	606	310	0.5115512

Algorithm 1 LinUCB with disjoint linear models.

0: Inputs: $\alpha \in \mathbb{R}_+$
1: **for** $t = 1, 2, 3, \ldots, T$ **do**
2: Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
3: **for all** $a \in \mathcal{A}_t$ **do**
4: **if** a is new **then**
5: $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
6: $\mathbf{b}_a \leftarrow \mathbf{0}_{d \times 1}$ (d-dimensional zero vector)
7: **end if**
8: $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1} \mathbf{b}_a$
9: $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}$
10: **end for**
11: Choose arm $a_t = \arg\max_{a \in \mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
12: $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t} \mathbf{x}_{t,a_t}^\top$
13: $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t \mathbf{x}_{t,a_t}$
14: **end for**

mean (to exploit)

Variance (to explore)

UCB style

Li, Lihong, Wei Chu, John Langford, and Robert E Schapire. 2010. "A Contextual-Bandit Approach to Personalized News Article Recommendation." In *Proceedings of the 19th International Conference on World Wide Web*, 661–70. ACM.

algorithm	size = 100%		size = 30%		size = 20%		size = 10%		size = 5%		size = 1%	
	deploy	learn	deploy	learn	deploy	learn	deploy	learn	deploy	learn	deploy	learn
ϵ-greedy	1.596 0%	1.326 0%	1.541 0%	1.326 0%	1.549 0%	1.273 0%	1.465 0%	1.326 0%	1.409 0%	1.292 0%	1.234 0%	1.139 0%
ucb	1.594 0%	1.569 18.3%	1.582 2.7%	1.535 15.8%	1.569 1.3%	1.488 16.9%	1.541 5.2%	1.446 9%	1.541 9.4%	1.465 13.4%	1.354 9.7%	1.22 7.1%
ϵ-greedy (seg)	1.742 9.1%	1.446 9%	1.652 7.2%	1.46 10.1%	1.585 2.3%	1.119 −12%	1.474 0.6%	1.284 −3.1%	1.407 0%	1.281 −0.8%	1.245 0.9%	1.072 −5.8%
ucb (seg)	1.781 11.6%	1.677 26.5%	1.742 13%	1.555 17.3%	1.689 9%	1.446 13.6%	1.636 11.7%	1.529 15.3%	1.532 8.7%	1.32 2.2%	1.398 13.3%	1.25 9.7%
ϵ-greedy (disjoint)	1.769 10.8%	1.309 −1.2%	1.686 9.4%	1.337 0.8%	1.624 4.8%	1.529 20.1%	1.529 4.4%	1.451 9.4%	1.432 1.6%	1.345 4.1%	1.262 2.3%	1.183 3.9%
linucb (disjoint)	1.795 12.5%	1.647 24.2%	1.719 11.6%	1.507 13.7%	1.714 10.7%	1.384 8.7%	1.655 13%	1.387 4.6%	1.574 11.7%	1.245 −3.5%	1.382 12%	1.197 5.1%
ϵ-greedy (hybrid)	1.739 9%	1.521 14.7%	1.68 9%	1.345 1.4%	1.636 5.6%	1.449 13.8%	1.58 7.8%	1.348 1.7%	1.465 4%	1.415 9.5%	1.342 8.8%	1.2 5.4%
linucb (hybrid)	1.73 8.4%	1.663 25.4%	1.691 9.7%	1.591 20%	1.708 10.3%	1.619 27.2%	1.675 14.3%	1.535 15.8%	1.588 12.7%	1.507 16.6%	1.482 20.1%	1.446 27%

Table 1: Performance evaluation: CTRs of all algorithms on the one-week evaluation dataset in the deployment and learning buckets (denoted by "deploy" and "learn" in the table, respectively). The numbers with a percentage is the CTR lift compared to ϵ-greedy.

Li, Lihong, Wei Chu, John Langford, and Robert E Schapire. 2010. "A Contextual-Bandit Approach to Personalized News Article Recommendation." In *Proceedings of the 19th International Conference on World Wide Web*, 661–70. ACM.

Similar Approaches

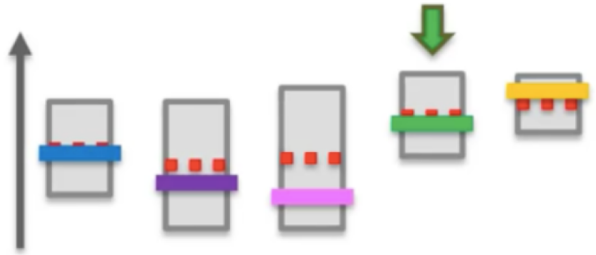
- *Laten Contextual Bandits* (LCB) (Zhou & Brunskill 2016) - in phase one, LCB runs LinUCB on the first *j* users to collect training data and in phase two, LCB trains/re-trains latent models based on mixture of linear regressions using the collected data.

- *C2UCB* (Qin et al. 2014) – contextual combinatorial bandits

- *CGPrank* (Vanchinathan et al. 2014) - exploits prior information specified in terms of a Gaussian process kernel function, which allows to share feedback in three ways: between positions in a list, between items, and between contexts.

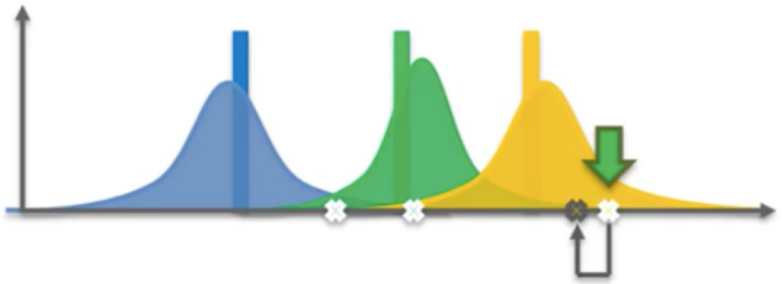
Thompson Sampling Algorithm

UCB



- Deterministic
- Requires update at every round

Thompson Sampling



- Probabilistic
- Can accommodate delayed feedback
- Better empirical evidence

Thompson Sampling

Step 1. At each round n, we consider two numbers for each ad i:

- $N_i^1(n)$ - the number of times the ad i got reward 1 up to round n,
- $N_i^0(n)$ - the number of times the ad i got reward 0 up to round n.

Step 2. For each ad i, we take a random draw from the distribution below:

$$\theta_i(n) = \beta(N_i^1(n) + 1, N_i^0(n) + 1)$$

Step 3. We select the ad that has the highest $\theta_i(n)$.

Bayesian Inference

- Ad i gets rewards $\boldsymbol{y}$ from Bernoulli distribution $p(\boldsymbol{y}|\theta_i) \sim \mathcal{B}(\theta_i)$.
- θ_i is unknown but we set its uncertainty by assuming it has a uniform distribution $p(\theta_i) \sim \mathcal{U}([0,1])$, which is the prior distribution.
- Bayes Rule: we approach θ_i by the posterior distribution

$$\underbrace{p(\theta_i|\boldsymbol{y})}_{\text{posterior distribution}} = \frac{p(\boldsymbol{y}|\theta_i)p(\theta_i)}{\int p(\boldsymbol{y}|\theta_i)p(\theta_i)d\theta_i} \propto \underbrace{p(\boldsymbol{y}|\theta_i)}_{\text{likelihood function}} \times \underbrace{p(\theta_i)}_{\text{prior distribution}}$$

- We get $p(\theta_i|\boldsymbol{y}) \sim \beta(\text{number of successes} + 1, \text{number of failures} + 1)$
- At each round n we take a random draw $\theta_i(n)$ from this posterior distribution $p(\theta_i|\boldsymbol{y})$, for each ad i.
- At each round n we select the ad i that has the highest $\theta_i(n)$.

Algorithm 2 Thompson sampling for the Bernoulli bandit

Require: α, β prior parameters of a Beta distribution
  $S_i = 0, F_i = 0, \ \forall i$. {Success and failure counters}
  for $t = 1, \ldots, T$ **do**
    for $i = 1, \ldots, K$ **do**
      Draw θ_i according to Beta$(S_i + \alpha, F_i + \beta)$.
    end for
    Draw arm $\hat{\imath} = \arg\max_i \theta_i$ and observe reward r
    if $r = 1$ **then**
      $S_{\hat{\imath}} = S_{\hat{\imath}} + 1$
    else
      $F_{\hat{\imath}} = F_{\hat{\imath}} + 1$
    end if
  end for

Chapelle O. and Li L. 2011. *An empirical evaluation of thompson sampling*. In Advances in Neural Information Processing Systems. 2249-2257.

Table 2: CTR regrets on the display advertising data.

Method	TS			LinUCB			ε-greedy			Exploit	Random
Parameter	0.25	0.5	1	0.5	1	2	0.005	0.01	0.02		
Regret (%)	4.45	3.72	3.81	4.99	4.22	4.14	5.05	4.98	5.22	5.00	31.95

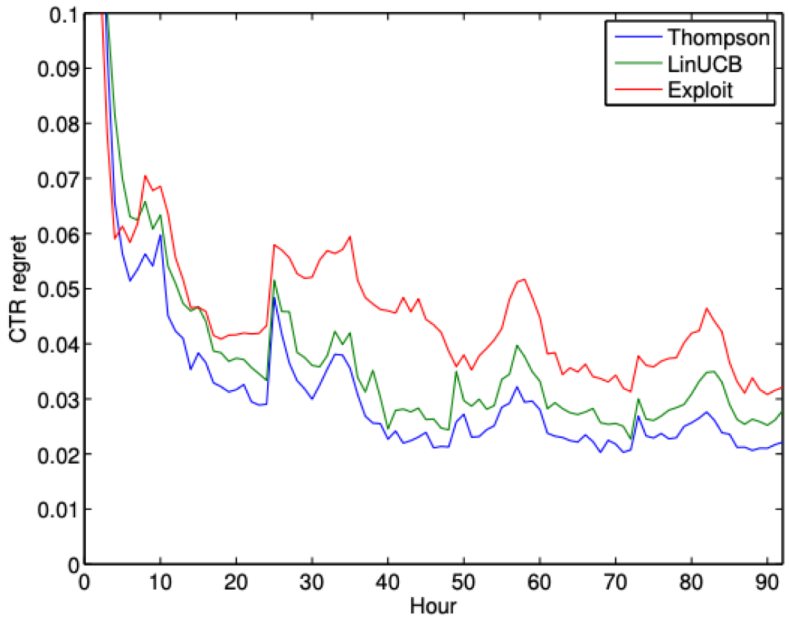
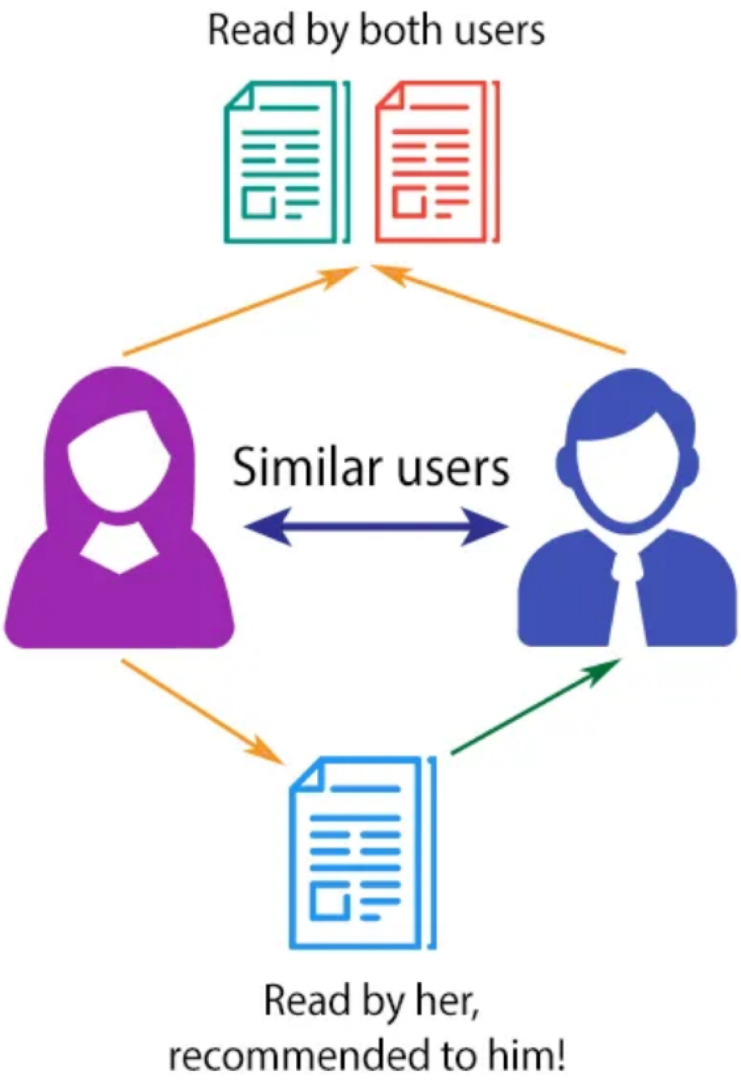


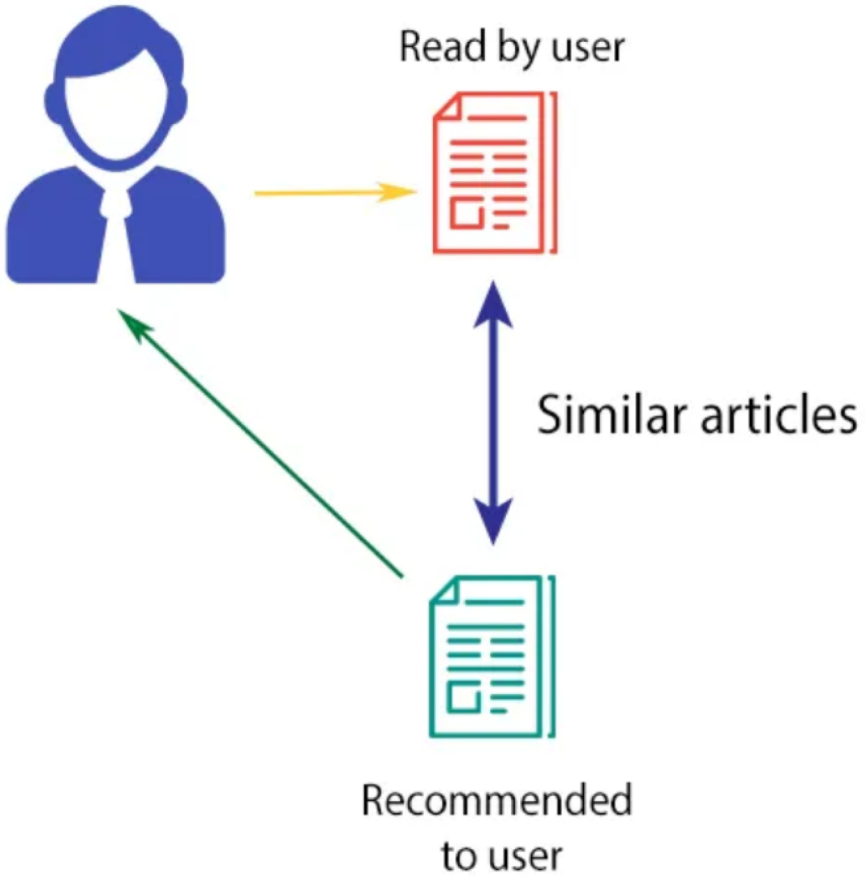
Figure 4: CTR regret over the 4 days test period for 3 algorithms: Thompson sampling with $\alpha = 0.5$, LinUCB with $\alpha = 2$, Exploit-only. The regret in the first hour is large, around 0.3, because the algorithms predict randomly (no initial model provided).

Chapelle O. and Li L. 2011. *An empirical evaluation of thompson sampling*. In Advances in Neural Information Processing Systems. 2249-2257.

COLLABORATIVE FILTERING

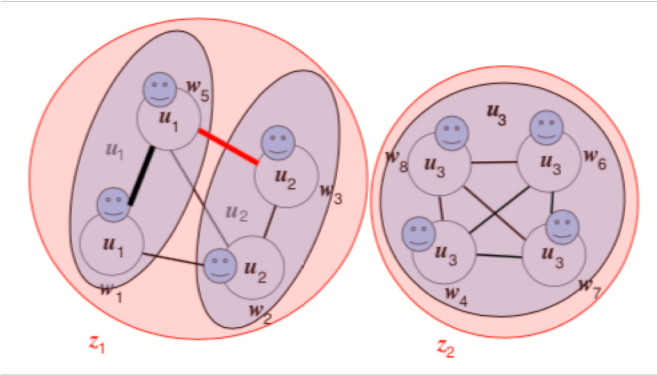


CONTENT-BASED FILTERING



Online Clustering of Bandits (CLUB)

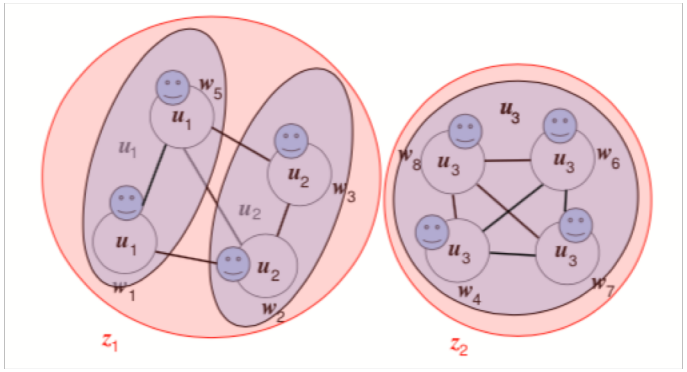
- CLUB is an approach to content recommendation based on adaptive clustering of bandit strategies
- Relevant to group recommendation
- Relies on sequential clustering of users



Gentile, C., Li, S., & Zappella, G. 2014. Online clustering of bandits. In *International Conference on Machine Learning.* 757-765.

The CLUB Algorithm

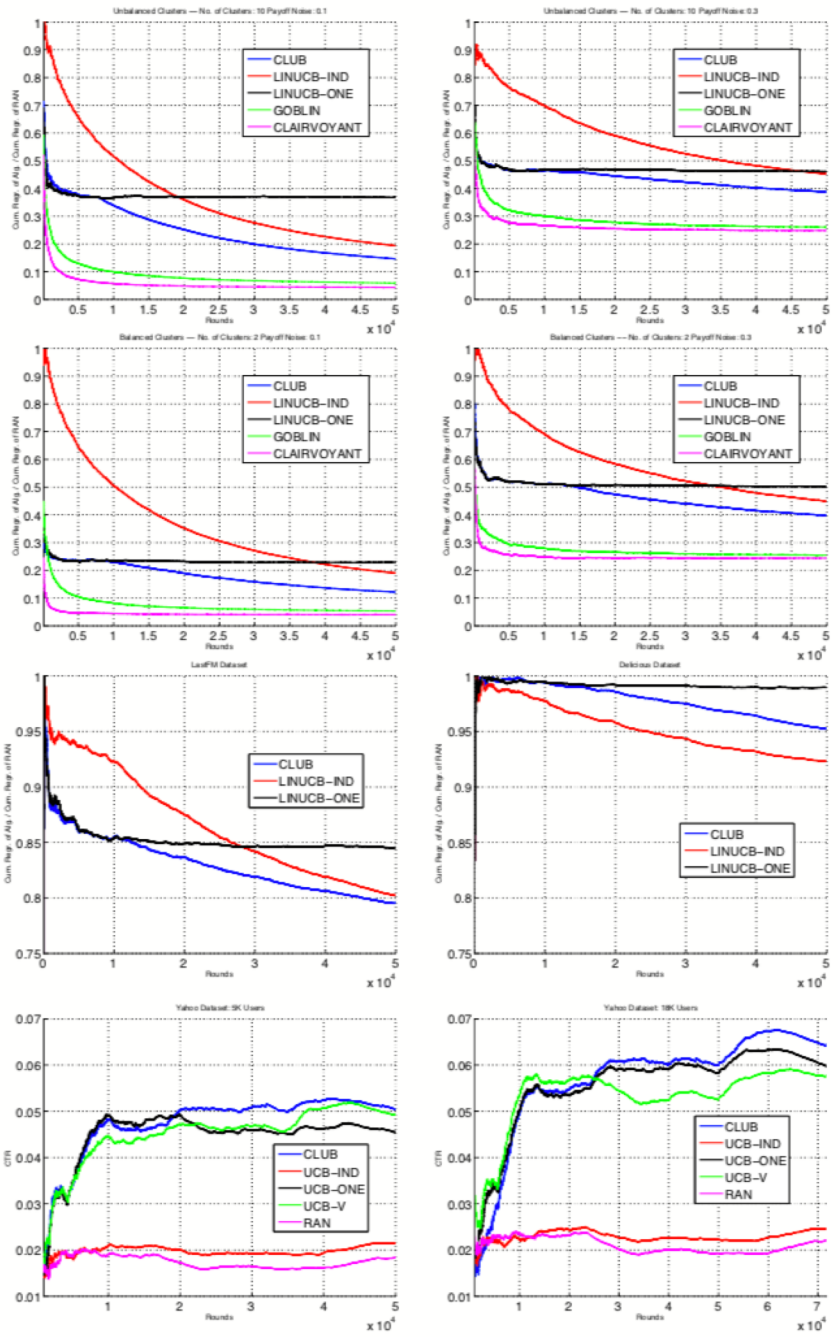
- n users, $m << n$ clusters
- Users' profiles $\boldsymbol{u}_i$, $i = 1 \ldots n$
- Clusters' profiles $\boldsymbol{u}_j$, $j = 1 \ldots m$
- Nodes i within cluster j share same profile $\boldsymbol{u}_j$
- One linear bandit per node and one linear bandit per cluster: node i hosts proxy $\boldsymbol{w}_i$, cluster j hosts proxy $\boldsymbol{z}_j$
- $\boldsymbol{z}_j$ is aggregation of proxies $\boldsymbol{w}_i$
- Nodes served sequentially in random order:



- Start off from full n-node graph (or sparsified version thereof) and single estimated cluster
- If $||\boldsymbol{w}_i - \boldsymbol{w}_j|| > \theta_{(i,j)} \implies$ delete edge (i, j)
- Clusters are current connected components
- When serving user i in estimated cluster j, update node proxy $\boldsymbol{w}_i$ and cluster proxy $\boldsymbol{z}_j$
- Recompute clusters after deleting edges

Experimental Results

1. Synthetic datasets
   - $c_t = 10$, $T = 55,000$, $d = 25$, and $n = 500$
   - Cluster V_j is random unit-norm vector $\boldsymbol{u}_j \in \mathbb{R}^d$
   - Context vectors $\boldsymbol{x}_{t,k} \in \mathbb{R}^d$ generated uniformly with unit-norm
   - Cluster relative size $|V_j| = n \frac{j^{-z}}{\sum_{\ell=1}^{m} \ell^{-z}}$, $j = 1, \ldots, m$, with $z \in \{0, 1, 2, 3\}$
   - Sequence of served users i_t generated uniformly at random over the n users
   - Payoff with each cluster $= \boldsymbol{u}_j^\top \boldsymbol{x}_{t,k}$ plus white noise
2. LastFM & Delicious ("hits" & "niches") datasets
   - $c_t = 25$, $T = 55,000$, and $d = 25$
   - LastFM contains 1,892 users, 17,632 artists
   - Delicious contains 1,861 users, 69,226 URLs
   - Payoff is 1 if the user listened or bookmarked
3. Yahoo! ("ICML 2012 Challenge") dataset
   - $c_t = 41$(med.), $T = 55(75),000$, and $d = 323$
   - $8,362,905$ records, $713,862$ users, 323 news
   - User described by 136D binary feature vector
   - Payoff is 1 if the user clicked the news



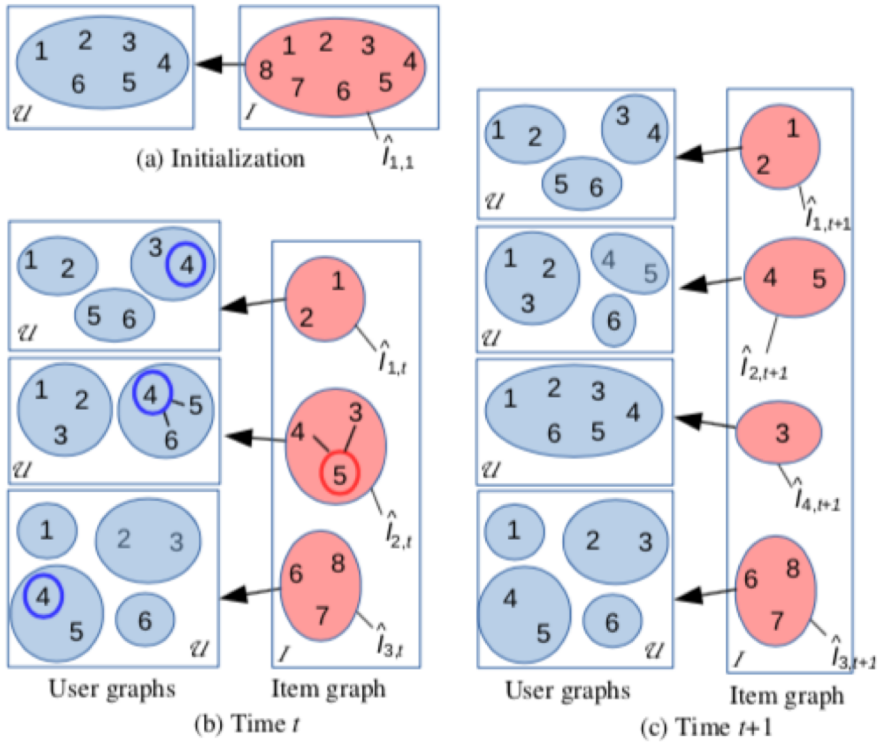
COFIBA

- Generalization of Clustering of Bandits (CLUB) with co-clustering for collaborative effects
- Explore the collaborative effect that arises due to ever-changing interaction of users and products
- Dynamically group users based on items under consideration and group items based on similarity of clusterings induced over users

Li, S., Karatzoglou, A., & Gentile, C. 2016. *Collaborative filtering bandits*. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 539-548.

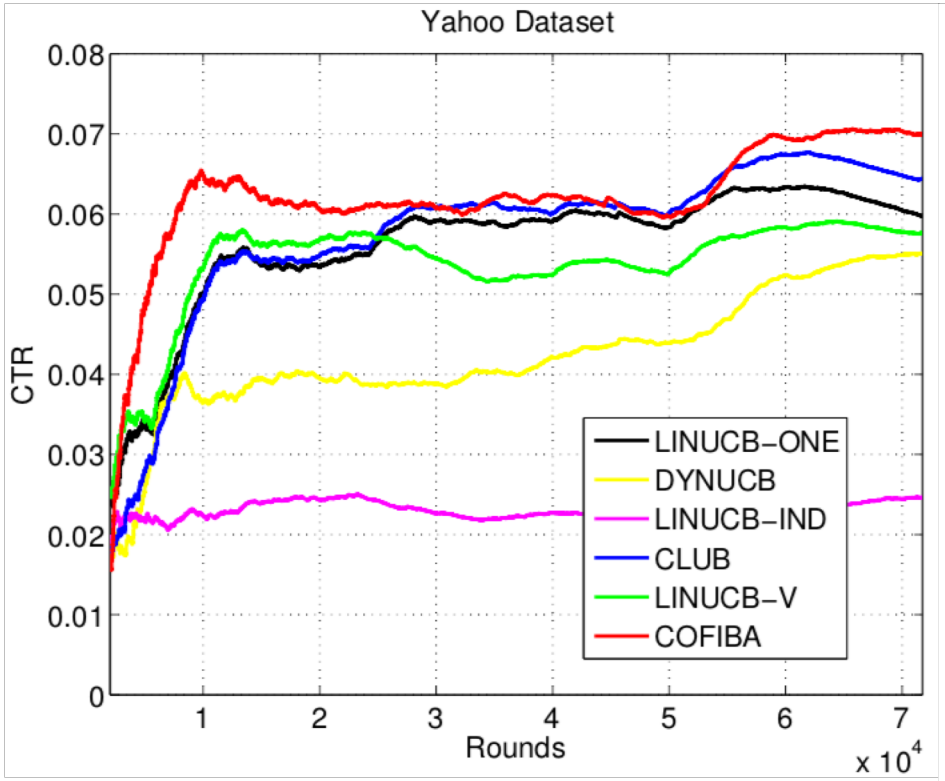
Algorithmic Idea

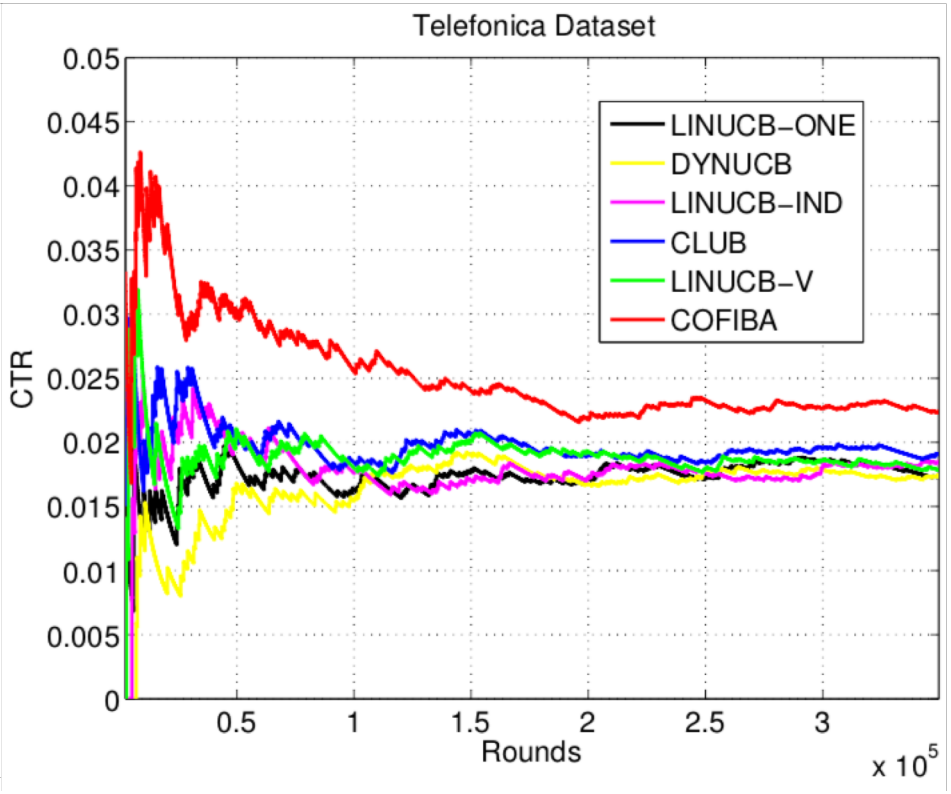
- Group users based on items, and group items based on the clustering induced over users
- Explore the priori: $I = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|I|}\}$
- Multiple clusterings over the set of users U and a single clustering over the set of items I
- $N_{i_t,t}(\boldsymbol{x}_{t,k})$ w.r.t. the items in C_{i_t} are stored into the clusters at the user side pointed to by those items
- Update the clusterings at user side and unique clustering at item side



Li, S., Karatzoglou, A., & Gentile, C. 2016. *Collaborative filtering bandits*. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 539-548.

COFIBA: Experimental Results





Collaborative Filtering Approaches

- *UCB-VB* and *UCB-PMF* (Nakamura 2014) – UCB combined with variational Bayes and stochastic matrix factorization to recommend promotional mail to users

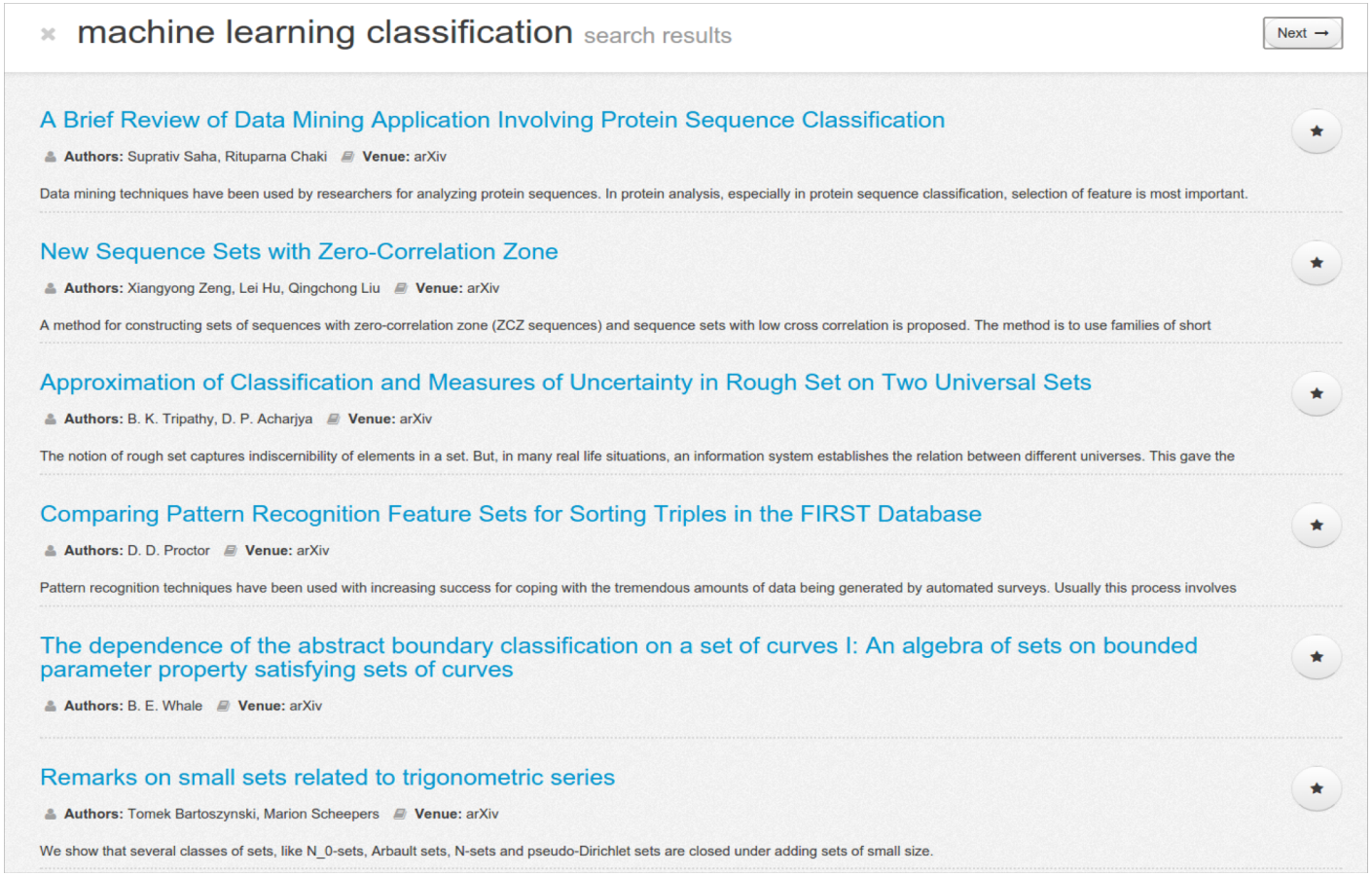
- *FactorUCB* (Wang et al. 2017) – similarity of users incorporated through a weighted graph

- *Particle Thompson Sampling for Matrix Factorization* (PTS) (Kawale et al. 2015) – Thompson Sampling combined with Rao-Blackwellized particle filter.

Recommendations with Limited Lifespan

- *Mortal bandits* (Chakrabarti et al. 2009) – arms have a lifetime after which they expire; the algorithm needs to explore constantly
- *Rotting Bandits* (Levine et al. 2017) – the reward of each arm decays as a function of the number of times it has been pulled
- *Multi-objective ranked bandit* (Lacerda 2017) – dynamically prioritizes different recommendation quality metrics during the life cycle of the user in the system

Personalizing Exploration-Exploitation



Medlar et al. *A System for Exploratory Search of Scientific Literature*. SIGIR 2016.

LinRel

In each iteration *t*, LinRel calculates:

$$a_i = x_i \cdot \left(X_t^T X_t + \mu I \right)^{-1} X_t^T$$

for each document *i* in dataset and selects for presentation top *n* documents that maximize:

$$\arg\max_X \{ a_i \cdot y_t + \frac{c}{2} \| a_i \| \}$$

for some constant $c > 0$

Study Design

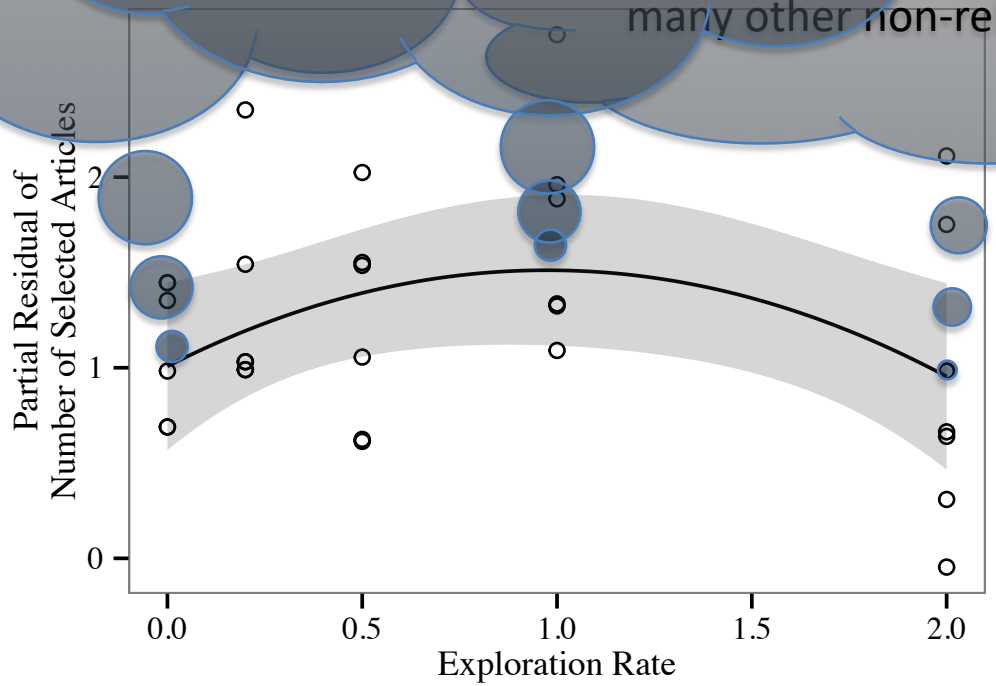
- Simulations: exploration rates to show different numbers of “exploratory” documents
- User study: MSc/PhD researchers in Machine Learning, 5 ML queries using different exploration rates
- Analysis: modelling combined with qualitative analysis of user performance data

Results

"Results are not quite satisfactory and about very specific definitions. I did not get [any] understanding of what topics are in this area"

"I went over several iterations. Results started getting way better [over iterations] and overall I am very satisfied"

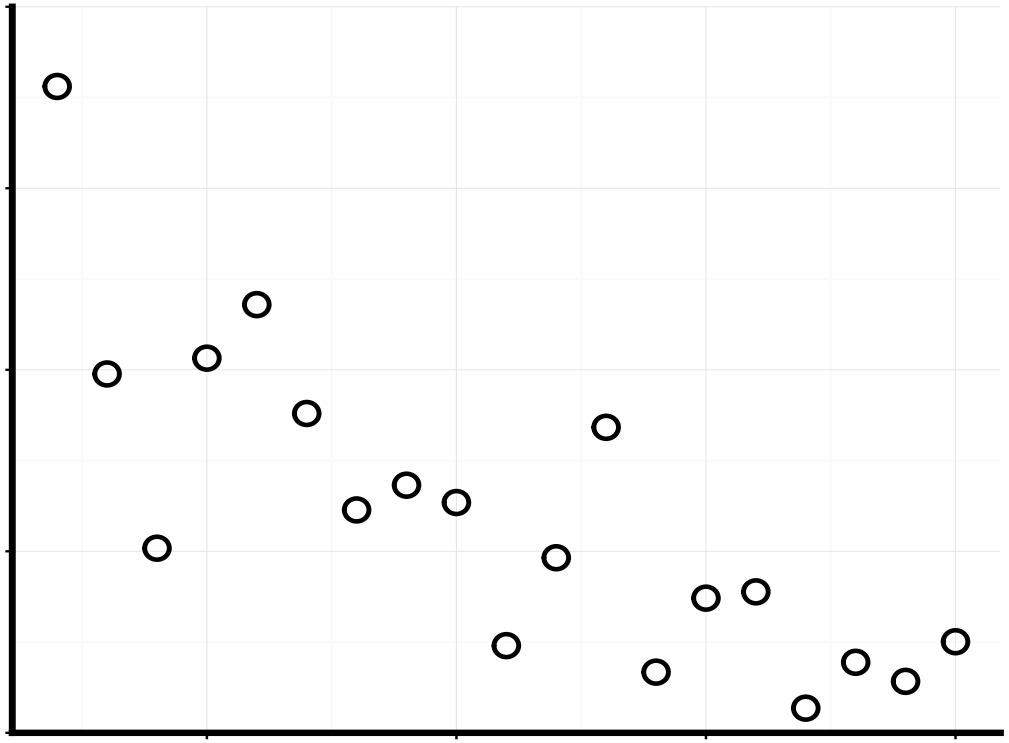
"[results are] too scattered and many other non-related papers"



Can we “personalize” the exploration level for each user for each search session?

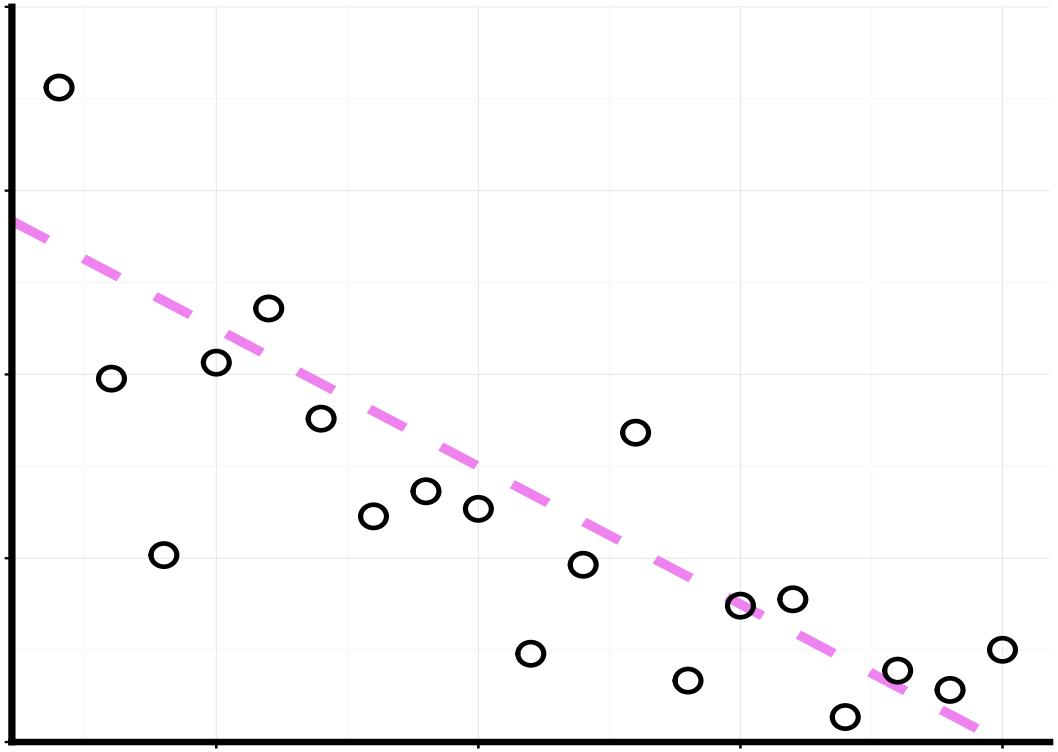
Approach

Ordinary regression fits a model based on linear relationships between response variable and exploratory values



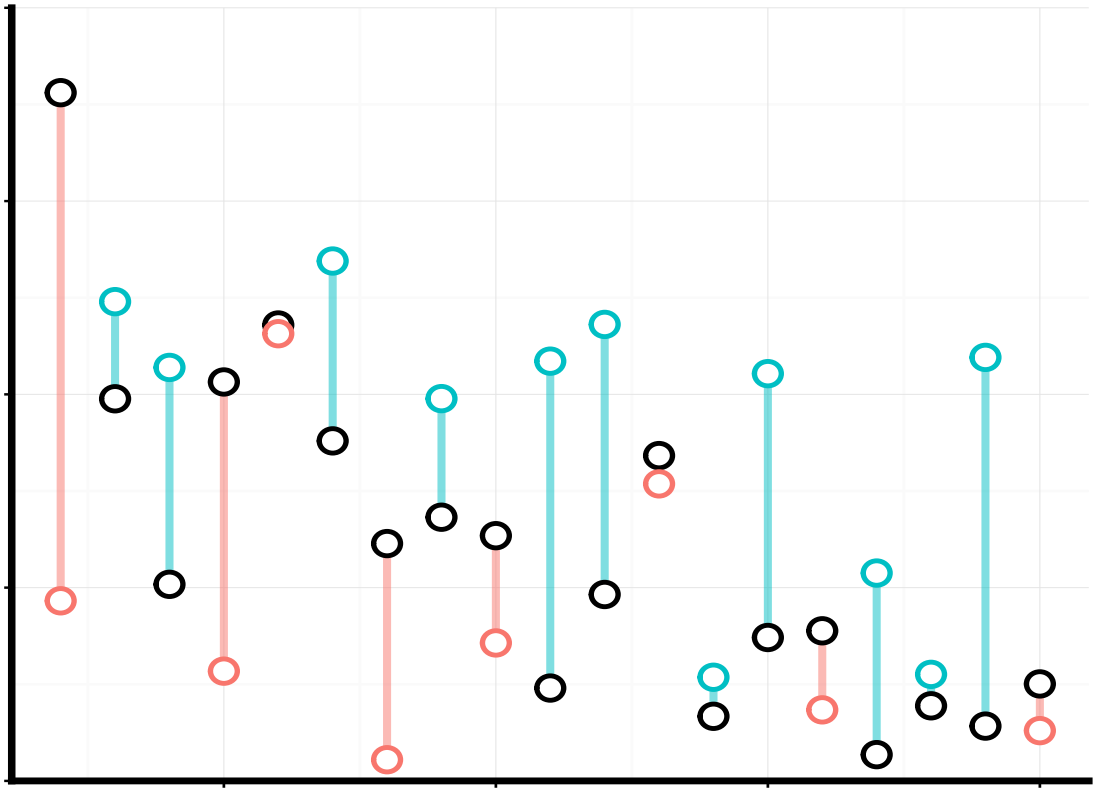
Approach

Ordinary regression fits a model based on linear relationships between response variable and exploratory values



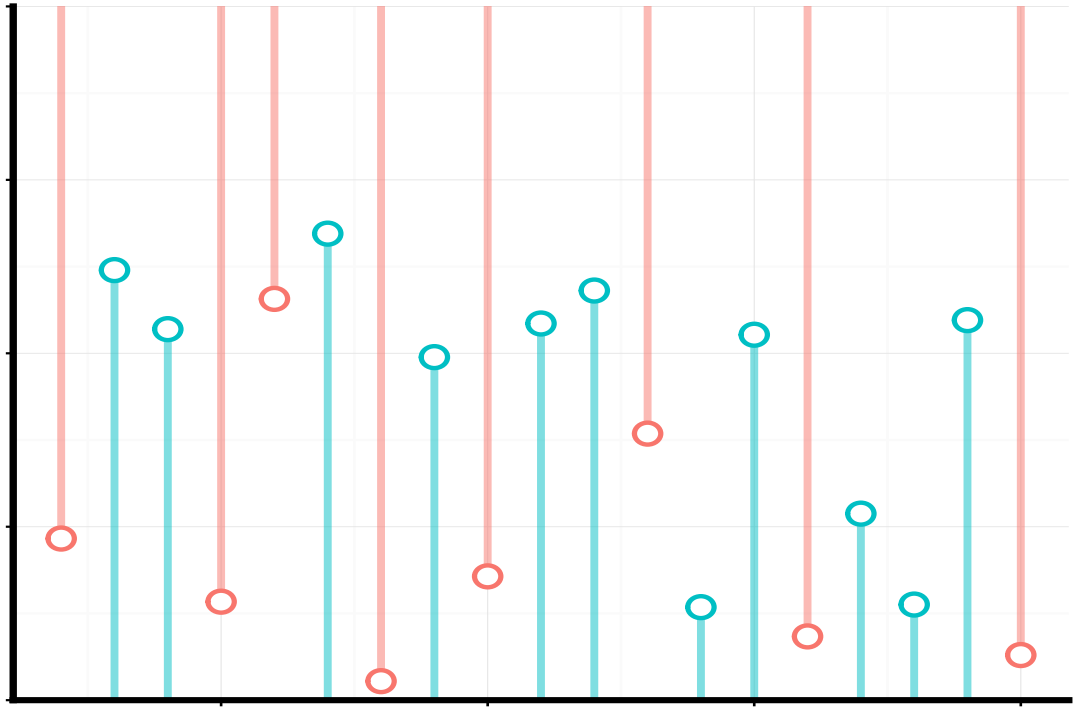
Problem

We don’t know apriori optimal exploration rate for given user and can’t observe their behaviour under these conditions.



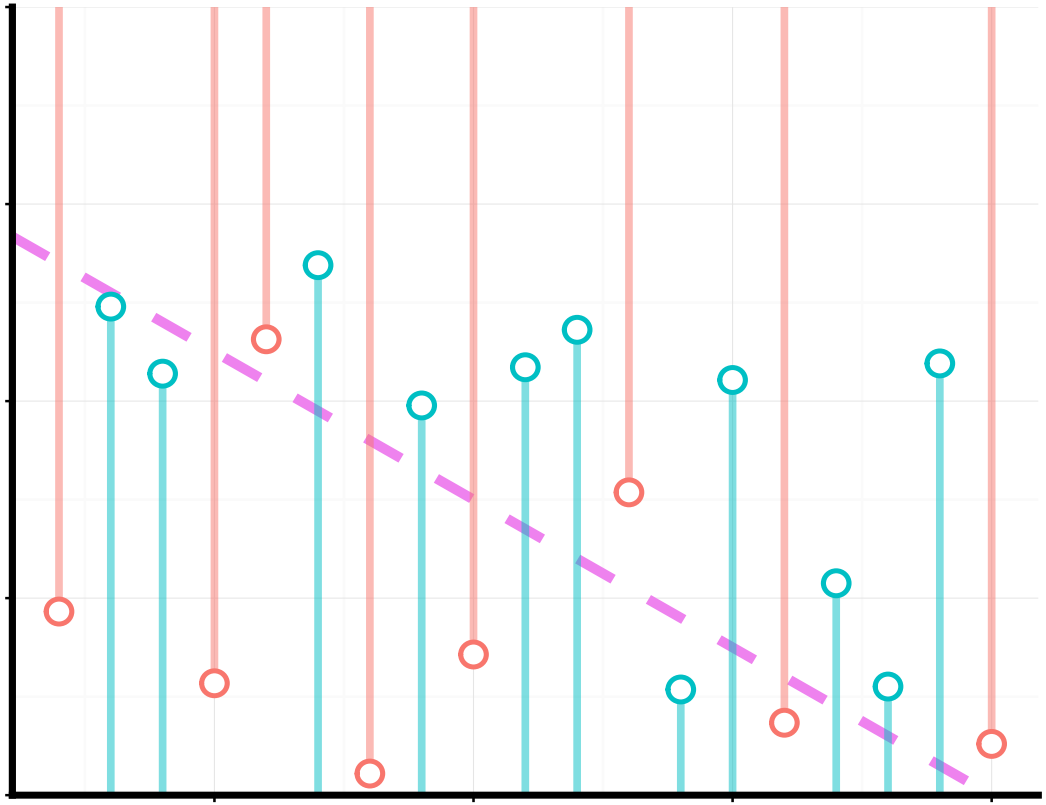
Solution

Instead of using specific exploration rates as the response variable, we created censored intervals based on user feedback.



Solution

Instead of using specific exploration rates as the response variable, we created censored intervals based on user feedback.

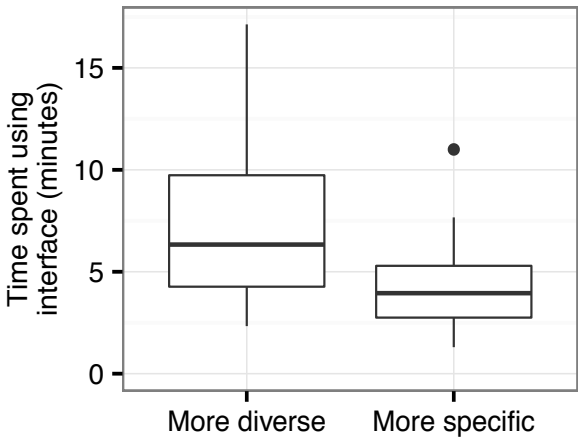


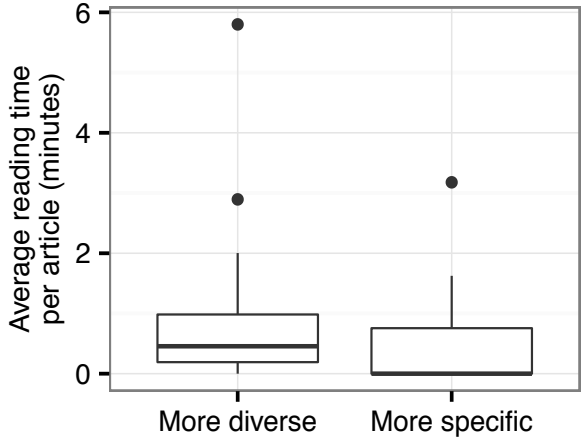
User Study Design

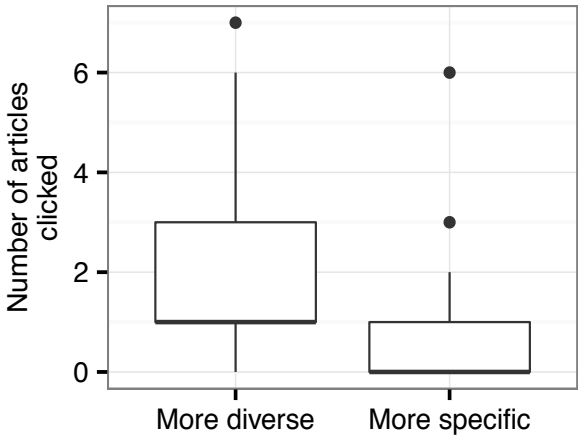
- Random exploration rate
- Participants asked to rate knowledge of topic
- Collect simple metrics: clicks, reading time, etc.
- Participants: 20 MSc students from a CS dept. performing two searches
- Data: 1.1 million arXiv documents
- After each search, participants completed a short questionnaire

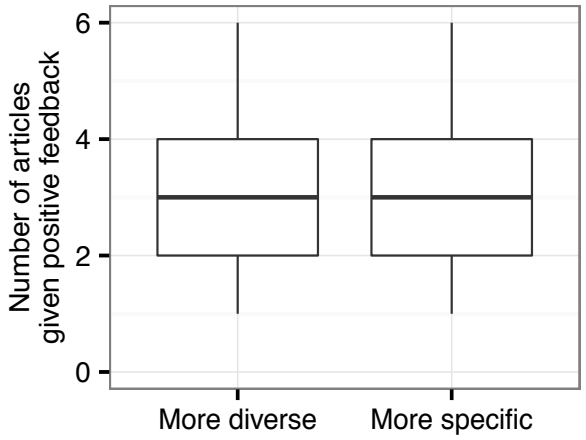
“The search results recommended by the system contained documents closely related to the initial search query as well as articles related to other topics with varying degrees of relevance to the initial query. Based on the search session that you have just completed, would you prefer the search results to contain: a) more articles closely related to the initial search query; b) more articles related to other topics with varying degrees of relevance to the initial search query”

Experimental Results

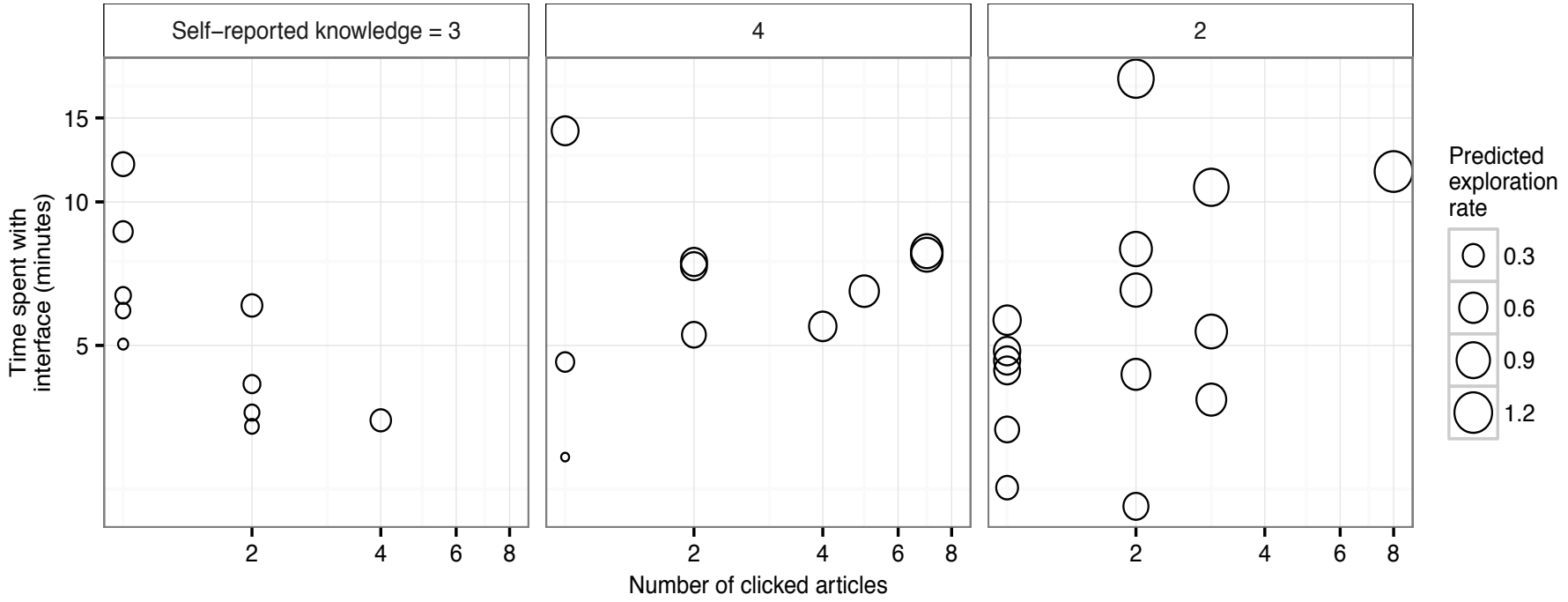




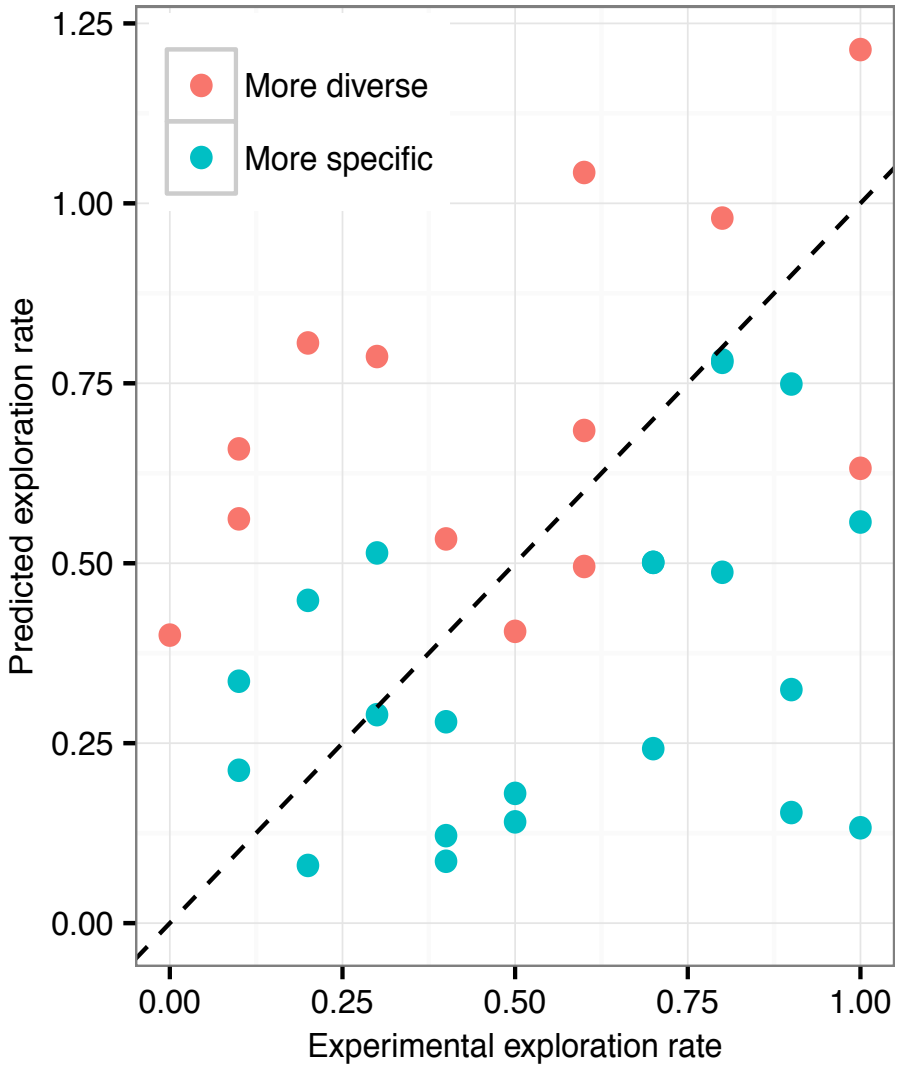




Graphical Representation of Model



Model Predictions and User Feedback



Questions?