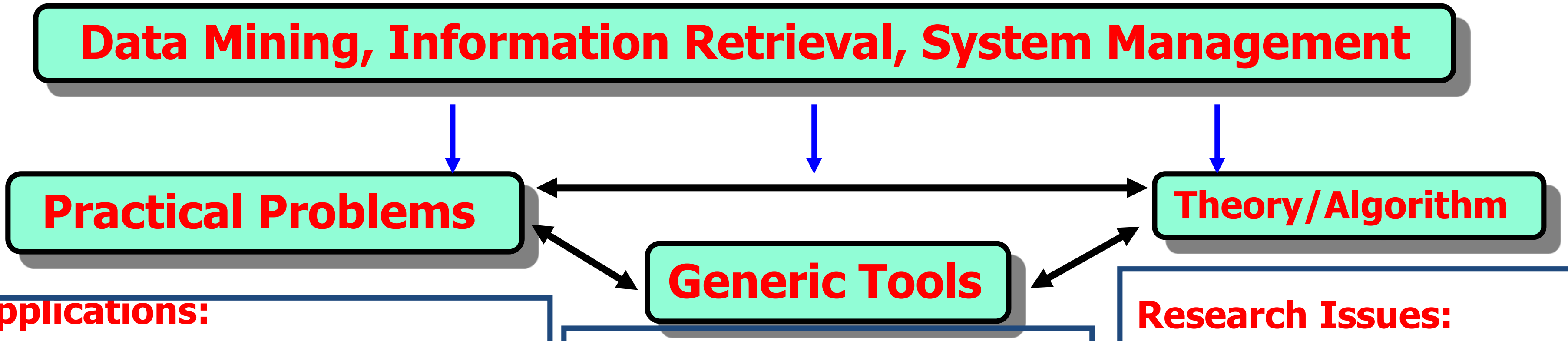
Multi-Armed Bandit Algorithms for Personalized Recommendation

Tao Li (李涛)

南京邮电大学 计算机学院

School of Computer Science, Florida International University

My Research Summary



Applications:

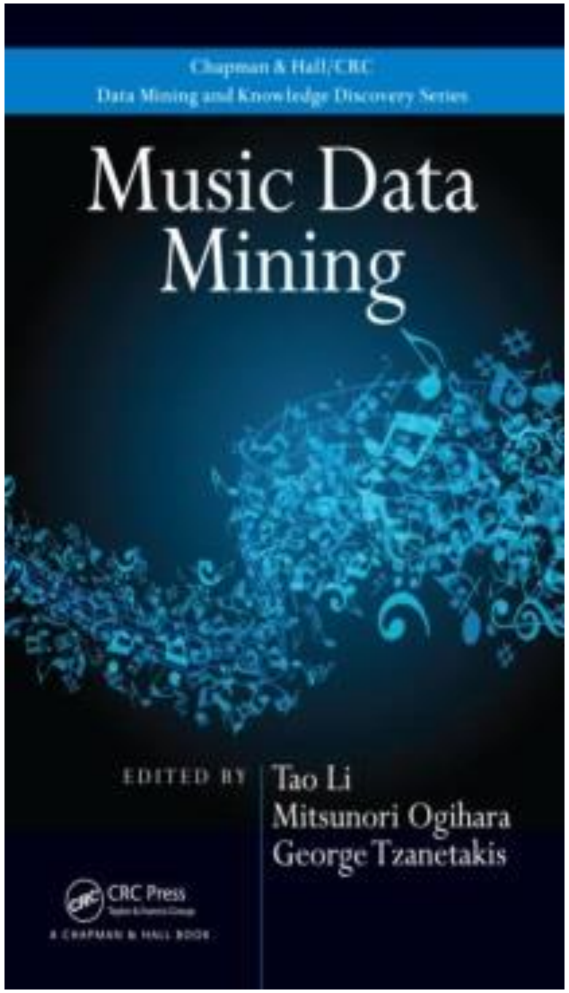
- **System Analytics**
  - Computing System Management, Intelligent Malware Detection
- **Cloud Analytics**
  - Capacity Planning, Demand Prediction
- **Social Media Analytics**
  - Sentiment analysis, influential user identification
- **Service Analytics**
  - Intelligent helpdesk, Cloud Service Marketplace, recommendation
- **Music Information Retrieval**
- **Text Mining**
  - Document categorization & summarization
- **Disaster Management**
- **Bioinformatics**

Tools and Libraries:

- **Computing System Management**
  - Event Mining, Log Parser, Visualization Tools
- **SumView: Online Product Review**
- **iHelp: Intelligent Helpdesk**
- **LIBGS: Gene Selection**
- **Twitter Event Summarization**
- **Information sharing and processing tools in disaster management**
- **Malware scanners**
- **iHR: Online Recruiting System**
- **FIU-Miner: A Fast, Integrated, and User-Friendly System for Data Mining in Distributed Environment**

Research Issues:

- **Matrix-based Learning Framework**
- **Semi-supervised Learning**
- **Learning from heterogeneous data types**
- **Log/Event mining**
- **Stream data mining**
- **Incremental/online ming**
- **Large-scale data analysis**


Chapman & Hall/CRC
Data Mining and Knowledge Discovery Series

Music Data Mining

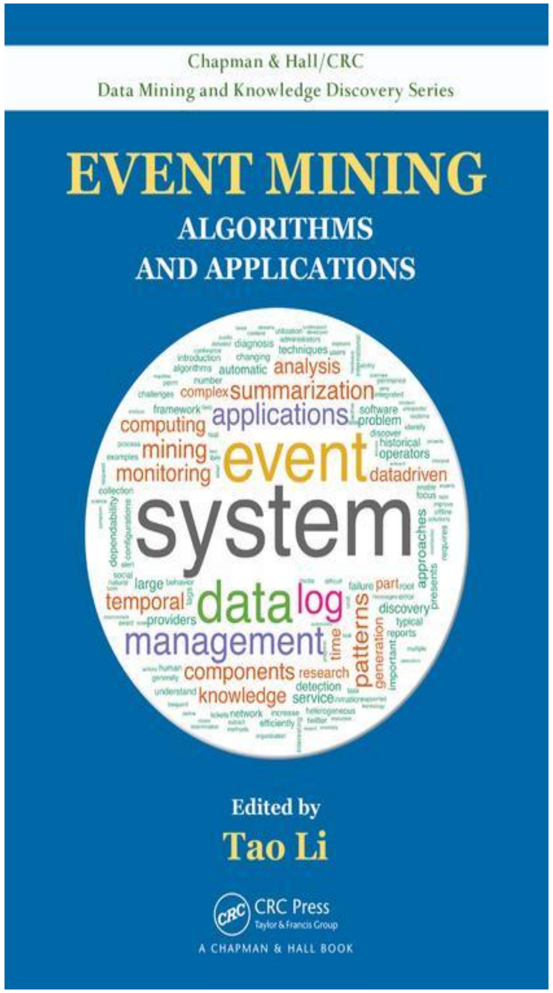
EDITED BY Tao Li
Mitsunori Ogihara
George Tzanetakis

CRC Press
A CHAPMAN & HALL BOOK

DATA MINING
Where Theory Meets Practice

数据挖掘的应用与实践
——大数据时代的案例分析

国际数据挖掘领域知名专家
李涛 等 著


Chapman & Hall/CRC
Data Mining and Knowledge Discovery Series

EVENT MINING
ALGORITHMS
AND APPLICATIONS

Edited by
Tao Li

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Recommender Systems

Who?
When?
Where?
What?

- User
- News
- Product
- Movie
- Music
- Photo
- Location

Outline

- Introduction
- Motivation
- Contextual-free Bandit Algorithms
- Contextual Bandit Algorithms
- Our Recent Studies
  - Ensemble Contextual Bandits for Personalized Recommendation
  - Personalized Recommendation via Parameter-Free Contextual Bandits
- Future Work
- Q&A

What is Personalized Recommendation?

- Personalized Recommendation help users find interesting items based the individual interest of each item.
  - Ultimate Goal: maximize user engagement.

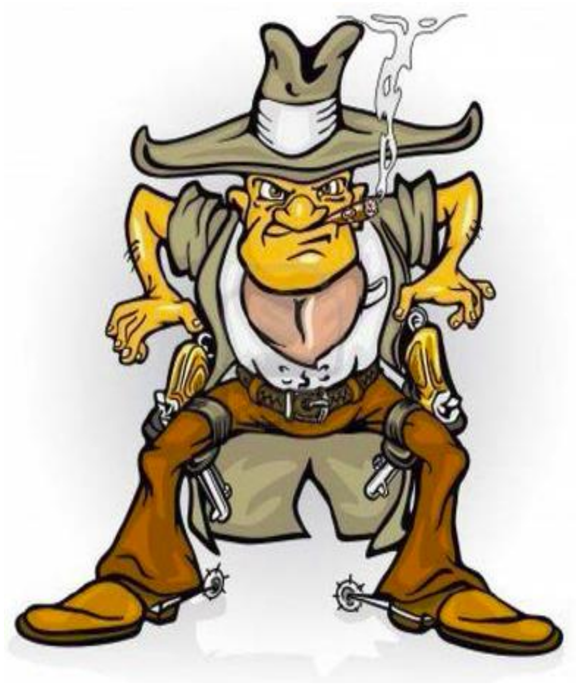
What is Cold Start Problem?

- Do not have enough observations for new items or new users.
  - How to predict the preference of users if we do not have data?

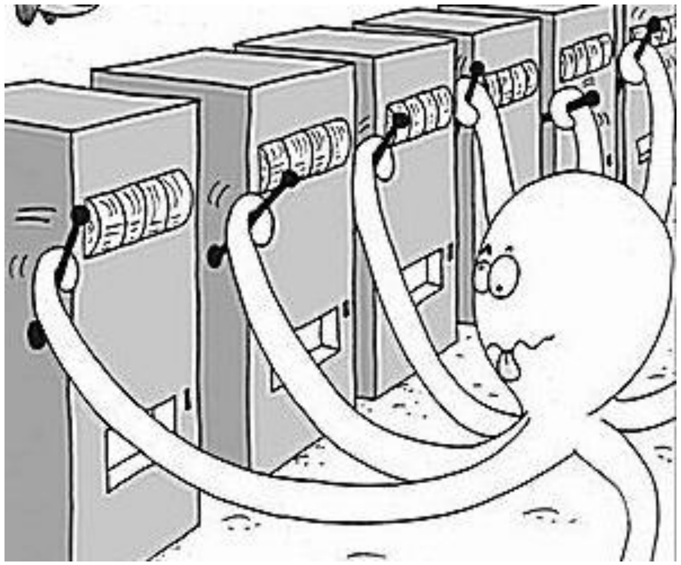
- Many practical issues for offline data
  - Historical user log data is biased.
  - User interest may change over time.

Approach: Multi-armed Bandit Algorithm

- A gambler walks into a casino
- A row of slot machines providing a random rewards

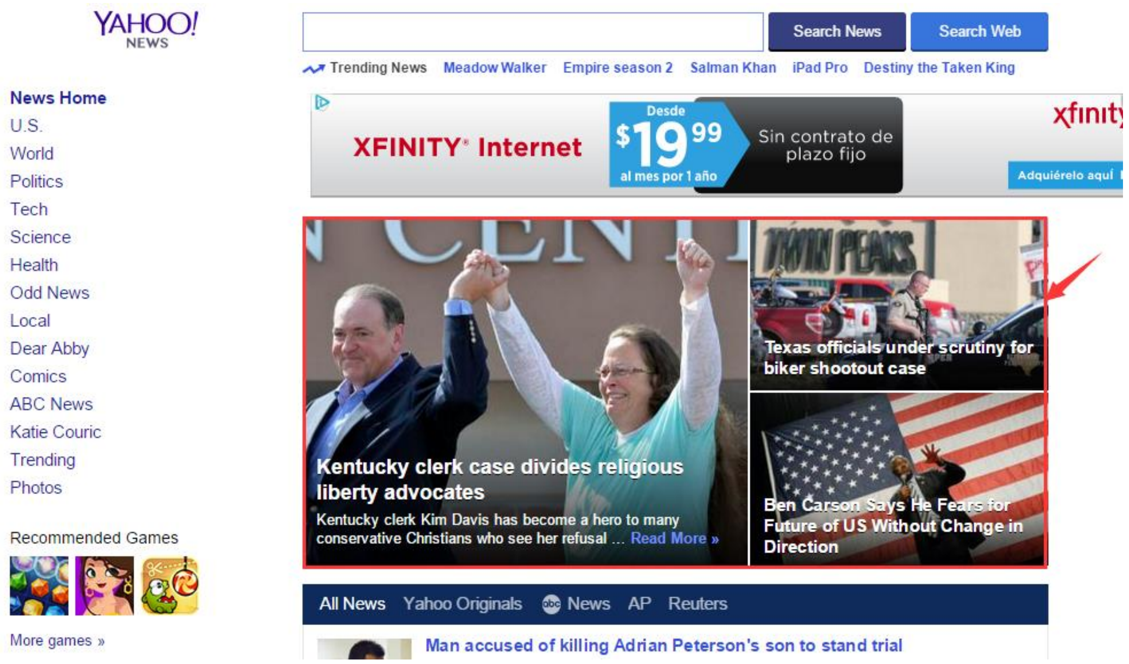


Objective: Maximize the sum of rewards(Money)!



Example: News Personalization

- **Recommend** news based on users' interests.
- **Goal**: Maximize user's Click-Through-Rate.



Kentucky clerk case divides religious liberty advocates

Kentucky clerk Kim Davis has become a hero to many conservative Christians who see her refusal ... Read More »

Texas officials under scrutiny for biker shootout case

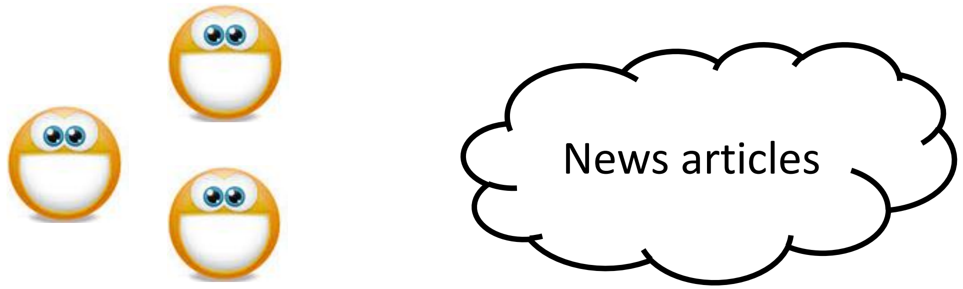
Ben Carson Says He Fears for Future of US Without Change in Direction

Man accused of killing Adrian Peterson's son to stand trial

[1] Li, Lihong, et al. "A contextual-bandit approach to personalized news article recommendation." *Proceedings of the 19th international conference on World wide web*. ACM, 2010.

Example: News Personalization

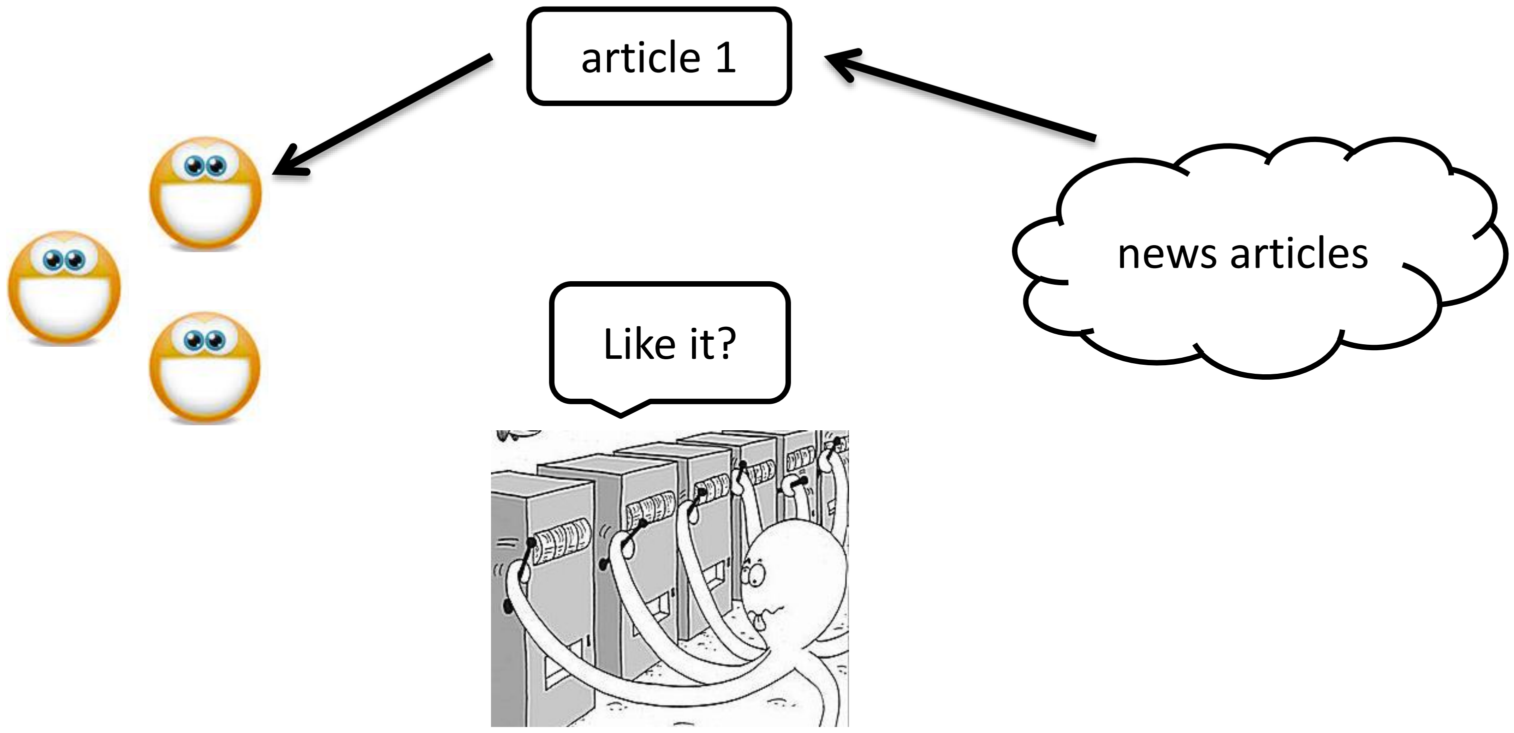
- There are a bunch of articles in the news pool
- Users come sequentially and ready to be entertained



[1] Zhou Li, “News personalization with Multi-armed bandits”.

Example: News Personalization

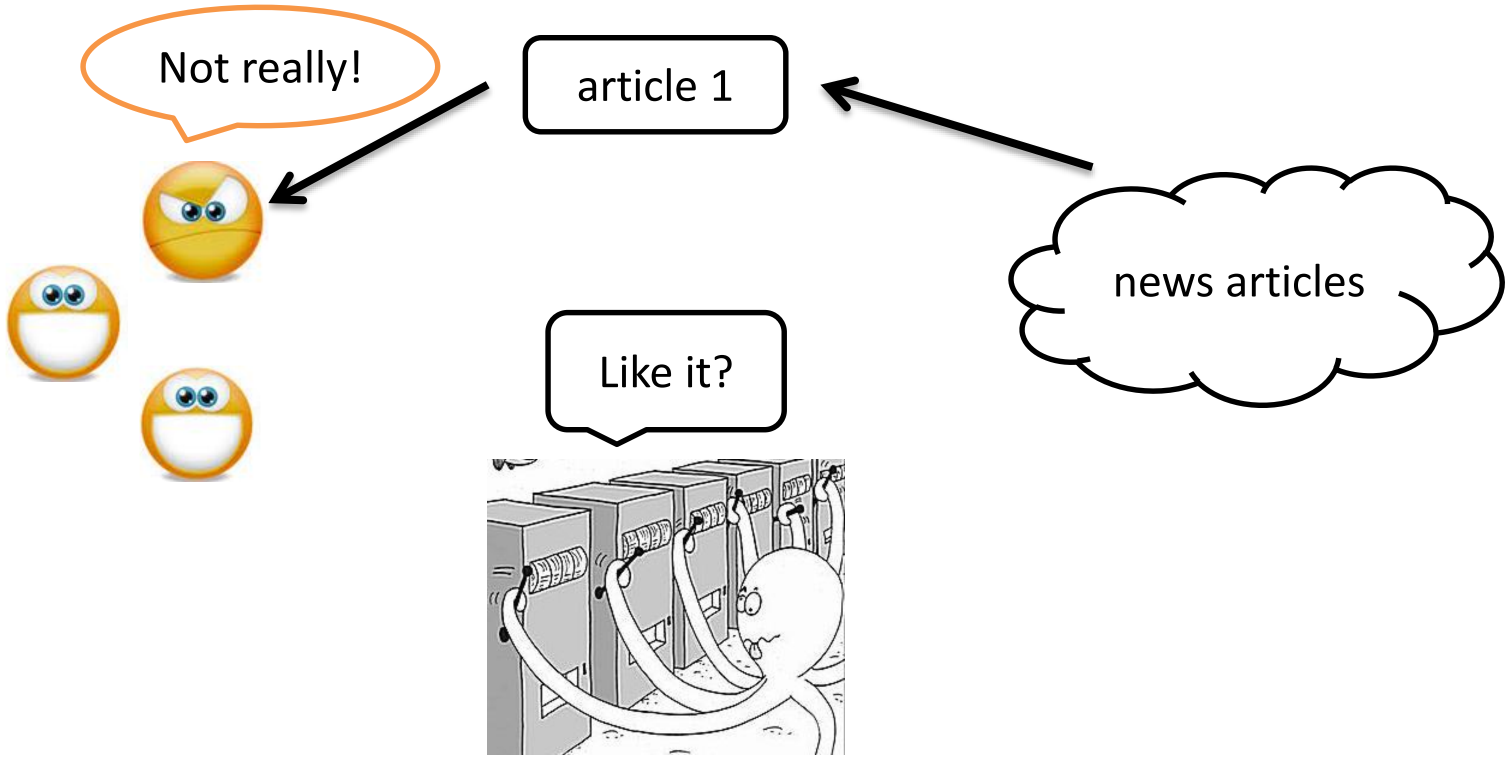
- At each time, we want to select one article for user.



MABs

Example: News Personalization

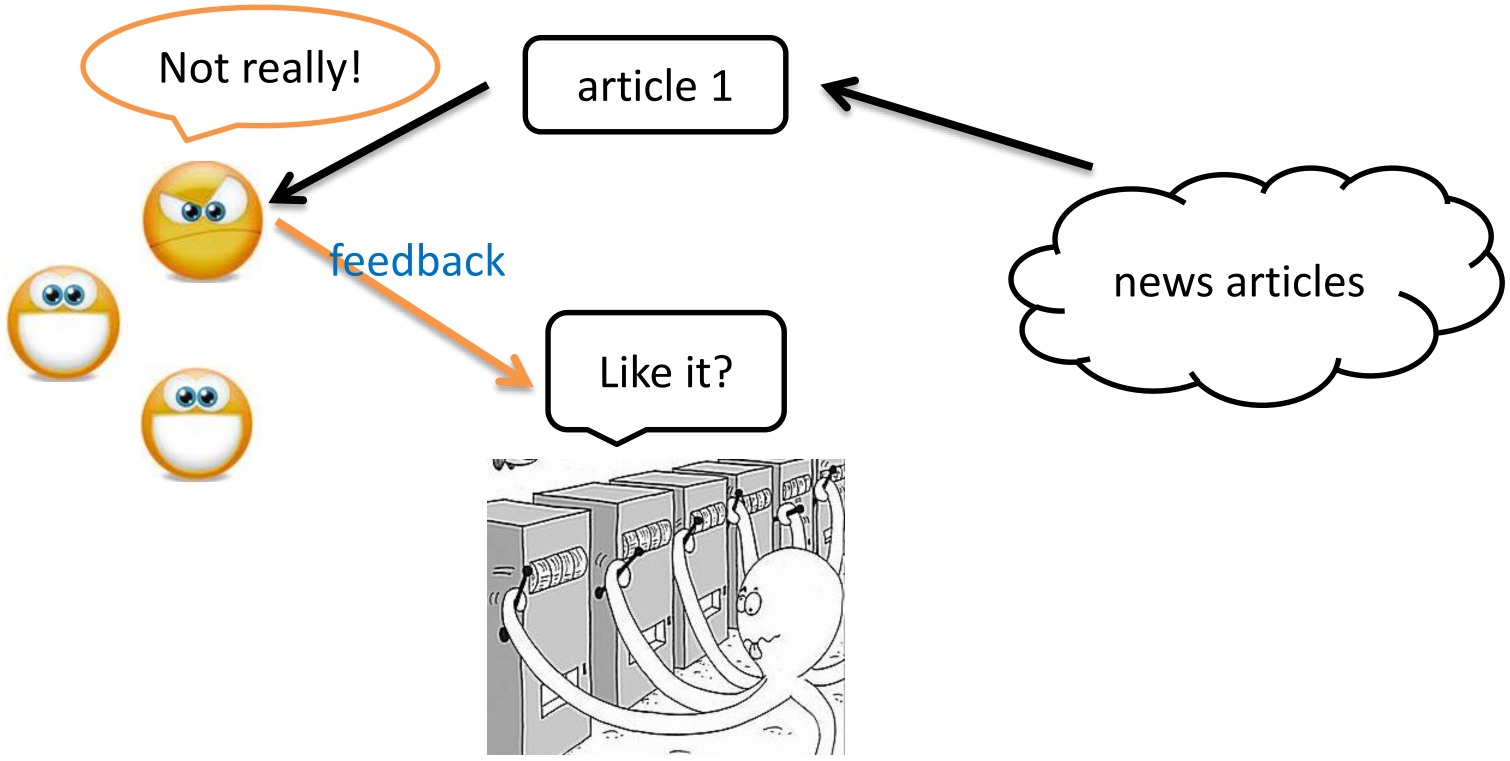
- Goal: maximum CTR.



MABs

Example: News Personalization

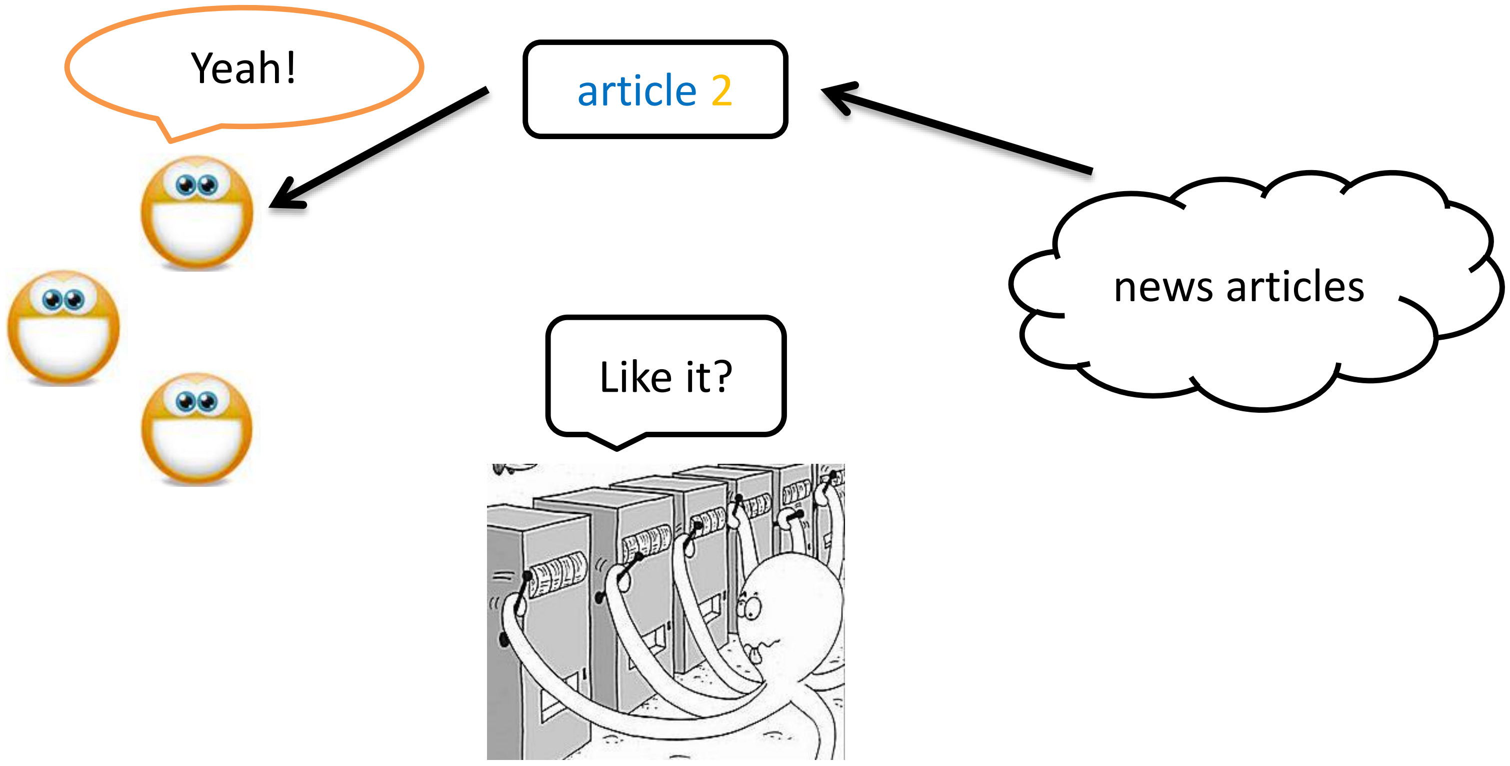
- Update the model with user's feedback



MABs

Example: News Personalization

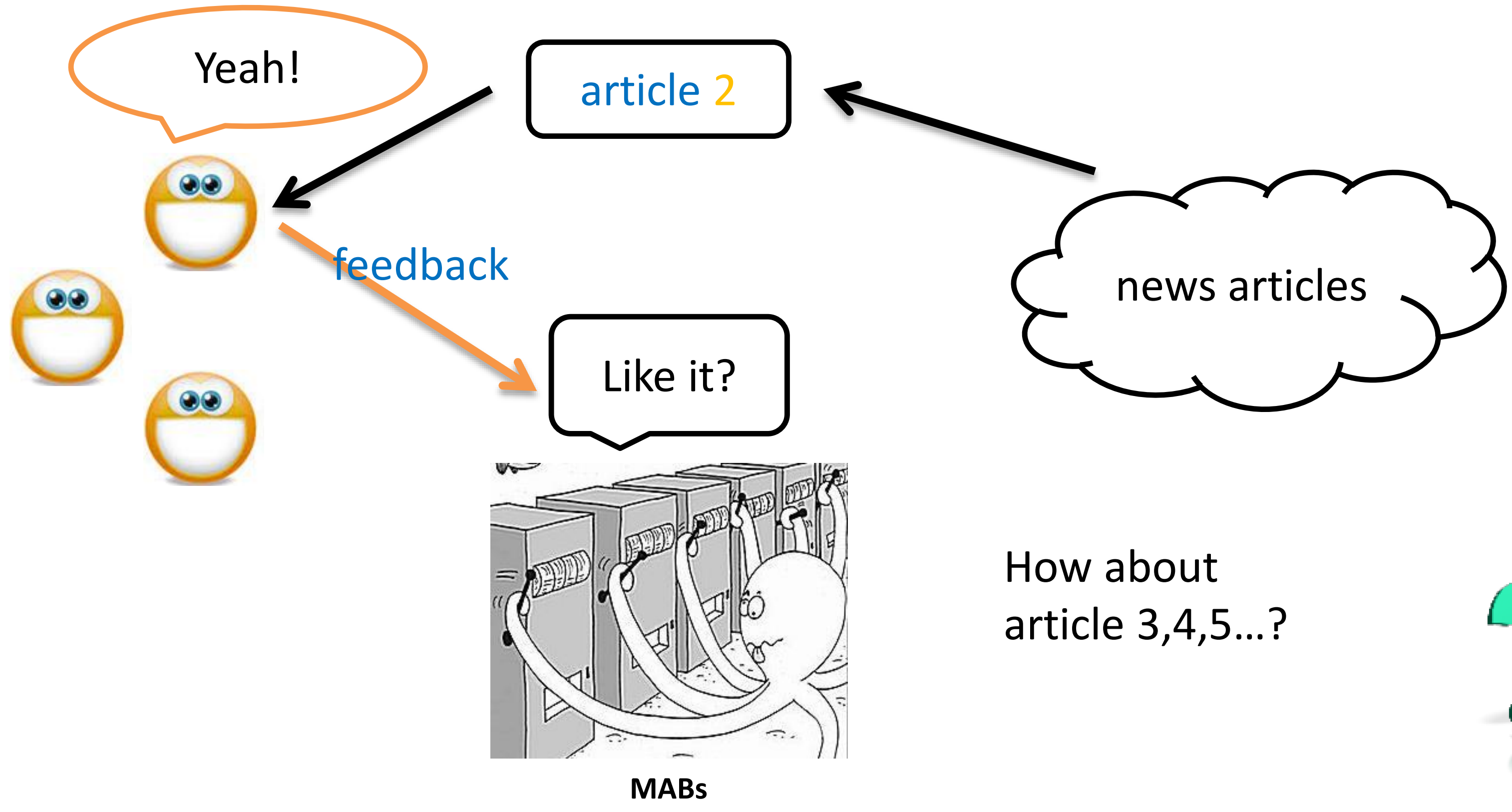
- Update the model once given the feedback



MABs

Example: News Personalization

- Update the model once given the feedback



Multi-Armed Bandit (MAB) Definition

- The MAB problem is a classical paradigm in Machine Learning in which an online algorithm choses from a set of strategies in a sequence of trials so as to maximize the total payoff of the chosen strategies[1].

[1] http://research.microsoft.com/en-us/projects/bandits/

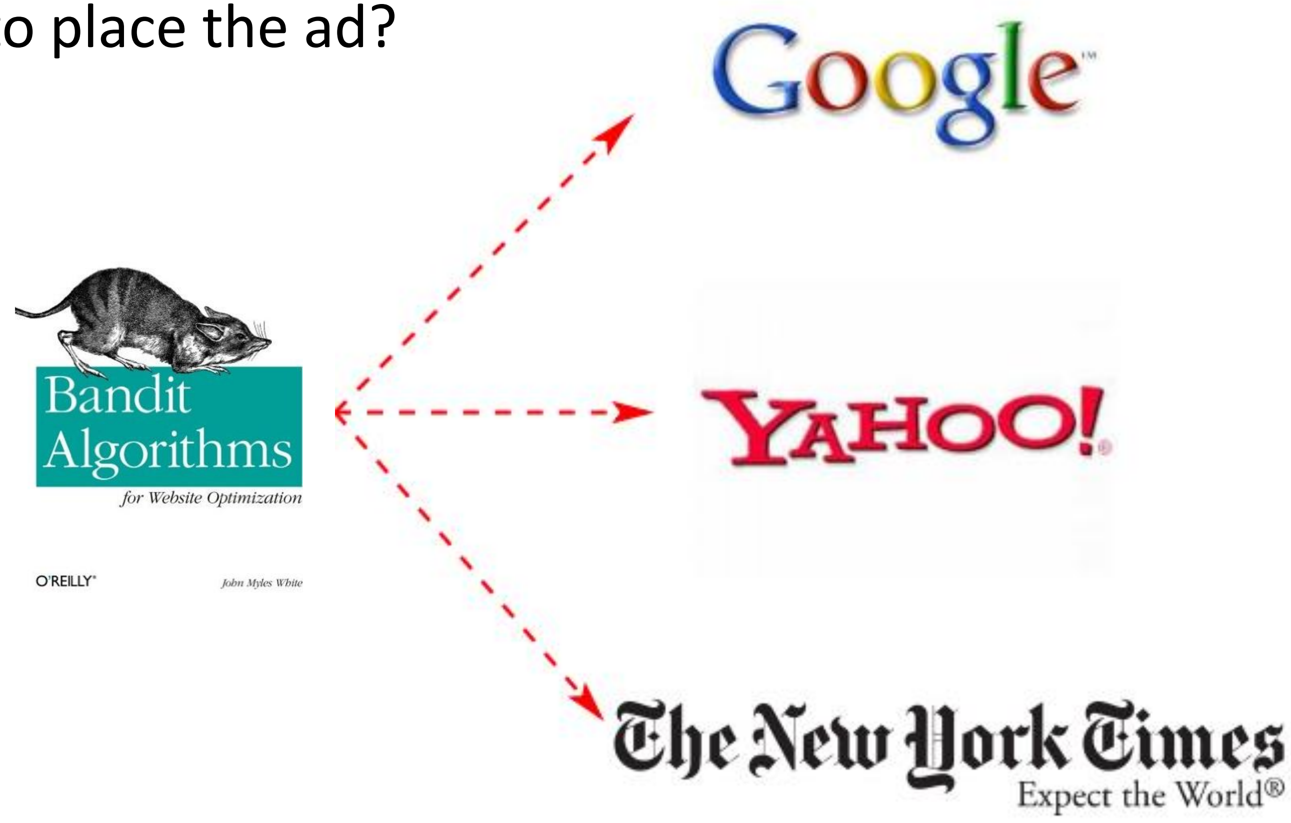
Application: Clinical Trial

- Two treatments with unknown effectiveness

[1] Einstein, A., B. Podolsky, and N. Rosen, 1935, “Can quantum-mechanical description of physical reality be considered complete?”, Phys. Rev. **47**, 777-780

Web advertising

- Where to place the ad?



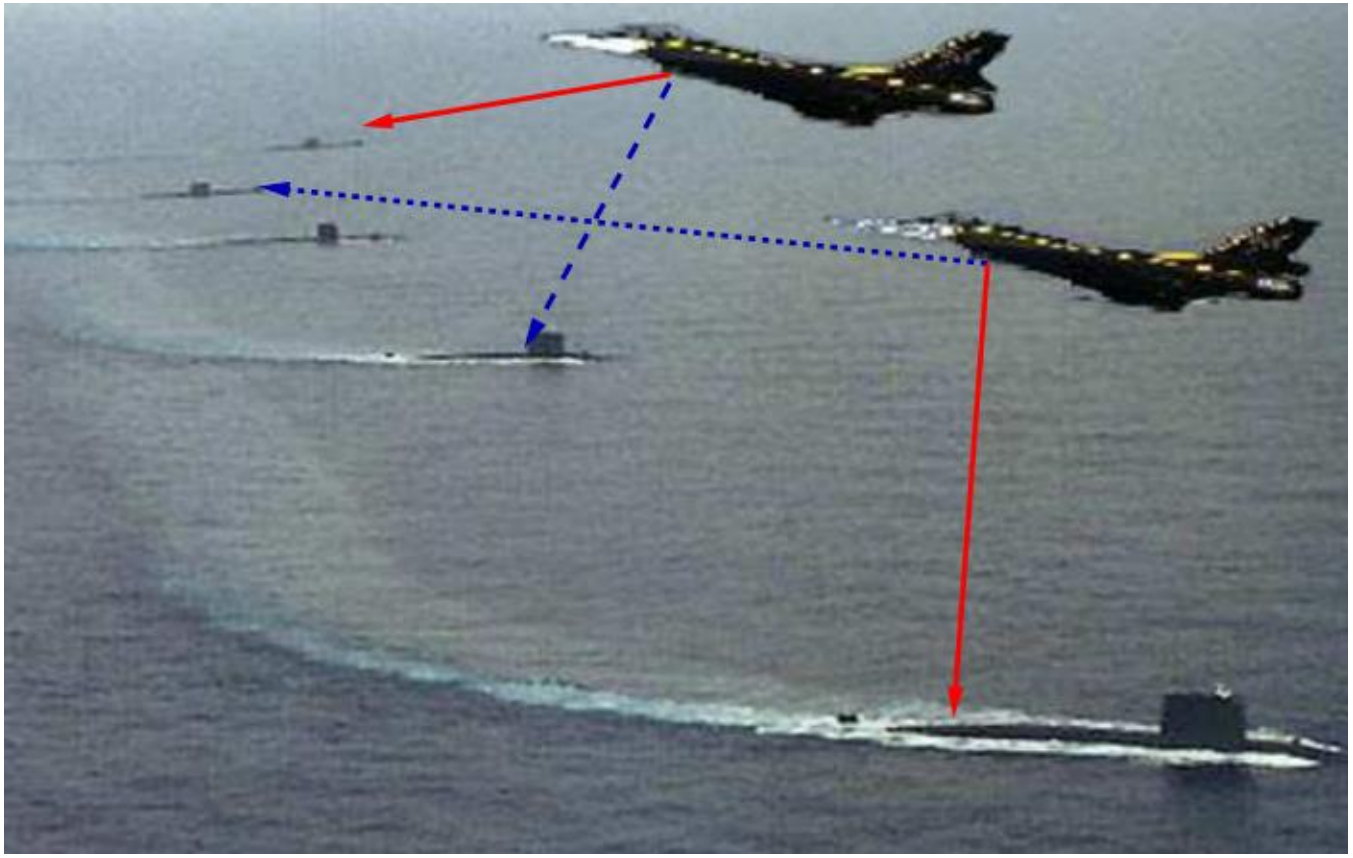
[1] Tang L, Rosales R, Singh A, et al. Automatic ad format selection via contextual bandits[C], Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, 2013: 1587-1594.

Playing Golf with multi-balls

[1] Dumitriu, Ioana, Prasad Tetali, and Peter Winkler. "On playing golf with two balls." *SIAM Journal on Discrete Mathematics* 16.4 (2003): 604-615.

Multi-Agent System

- K agents tracking N (N > K) targets:



[1] Ny, Jerome Le, Munther Dahleh, and Eric Feron. "Multi-agent task assignment in the bandit framework." Decision and Control, 2006 45th IEEE Conference on. IEEE, 2006.

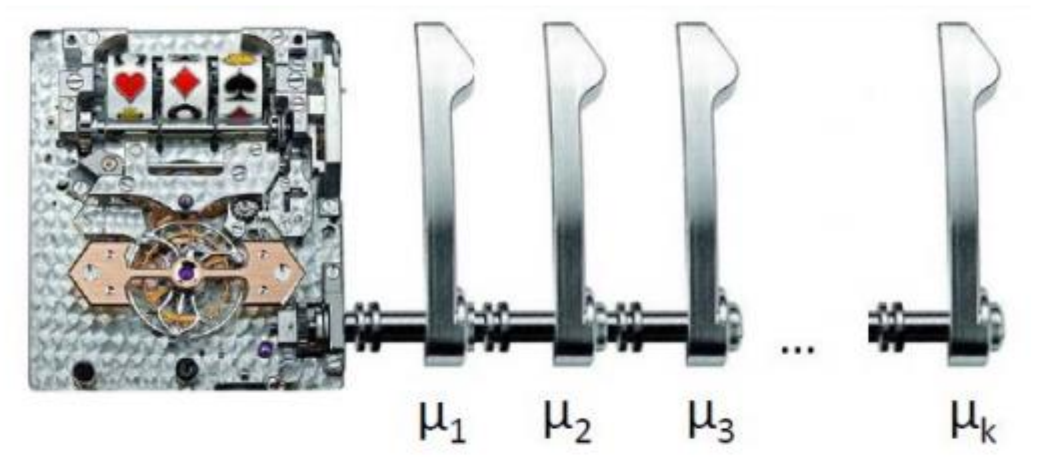
Some Jargon Terms[1]

- Arm: one idea/strategy
- Bandit: A group of ideas(strategies)
- Pull/Play/Trial: One chance to try your strategy
- Reward: The unit of success we measure after each pull
- Regret: Performance Metric

- Learning through experimentation

[1] **Bandit Algorithms for Website Optimization** Developing, Deploying, and Debugging By John Myles White, O'Reilly Media,2012

K-Armed Bandit



- Each Arm a
  - Wins(reward=1) with fixed(unknown) prob. μ_a
  - Loses(reward=0) with fixed(unknown) prob. $(1 - \mu_a)$

- How to pull arms to maximize total reward?(estimate the arm's prob. of winning μ_a)

Model of K-Armed Bandit

- Set of $\boldsymbol{k}$ choices(arms)
- Each choice ***a*** is associated with unknown probability distribution $\boldsymbol{P_a}$ in **[0, 1]**
- We play the game for ***T*** rounds
- In each round ***t :***
  - We pick some arm ***j***
  - We obtain random sample $\boldsymbol{X_t}$ from $\boldsymbol{P_j}$
- Goal: maximize $\sum_{t=1}^{T} \boldsymbol{X_t}$ (without known μ_a)
- However, every time we pull some arm ***a*** we get to learn a bit about μ_a.

Performance Metric: Regret

- Let be μ_a the mean of $\boldsymbol{P_a}$
- Payoff/reward **best arm**: $\mu^* = \boldsymbol{max}\{\mu_a | \boldsymbol{a} = \mathbf{1}, \ldots, \boldsymbol{k}\}$
- Let $i_1, \ldots i_T$ be the sequence of arms pulled
- Instantaneous regret at time t: $r_t = \mu^* - \mu_{a_{it}}$
- Total regret:
    - $\boldsymbol{R_T} = \sum_{t=1}^{T} \boldsymbol{r_t}$
- Typical goal: arm allocation strategy that guarantees :
    - $\frac{\boldsymbol{R_T}}{T} \to 0$ as $T \to \infty$

Allocation Strategies

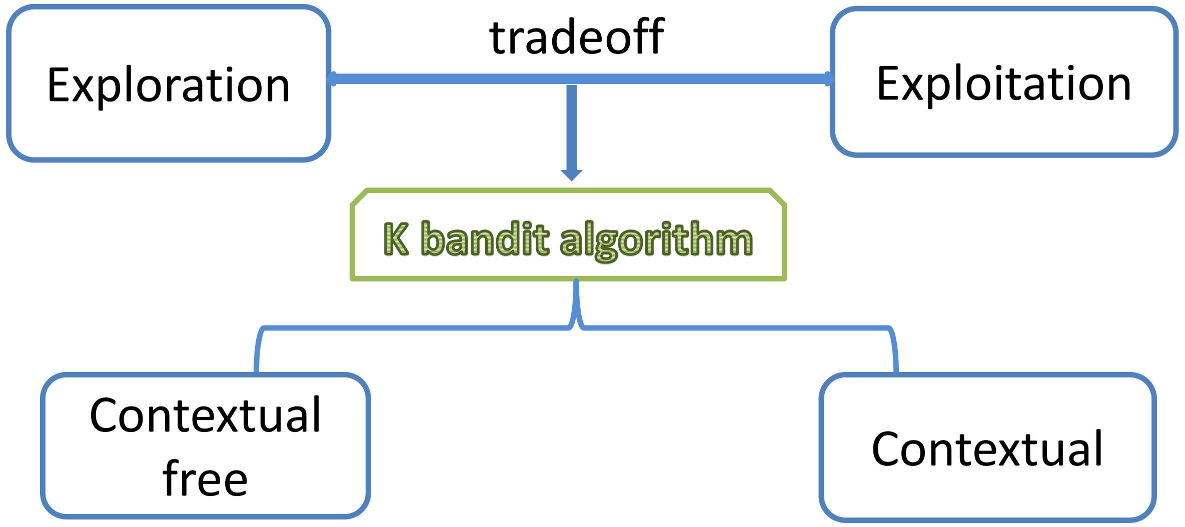
- If we knew the payoffs, which arm should we pull?
  - **best arm**: $\mu^* = \boldsymbol{max}\{\, \mu_a |\, \boldsymbol{a} = \mathbf{1}, \dots, \boldsymbol{k}\}$
- What if we only care about estimating payoff μ_a?
  - Pick each of ***k*** arms equally often : $\frac{\boldsymbol{T}}{\boldsymbol{k}}$
  - **Estimate** : $\widehat{\mu_a} = \sum_{j=1}^{\frac{T}{k}} X_{a,j} \,/ (\frac{\boldsymbol{T}}{k})$ $\Rightarrow$ $\frac{\boldsymbol{k}}{T} \sum_{j=1}^{T/k} X_{a,j}$
  - Total regret:
    - $\boldsymbol{R_T} = \frac{\boldsymbol{T}}{k} \sum_{a=1}^{k} (\mu^* - \mu_a)$

$X_{a,j}$ payoff received when pulling an arm ***a*** for *j-th* time

Exploitation vs. Exploration

- Tradeoff:
  - Only exploitation(making decisions based on history data), you will have bad estimation for “best” items.
    - Exploitation: Pull an arm currently having the highest estimate
  - Only exploration(gathering data about arm payoffs), you will have low user’s engagement.
    - Exploration: Pull an arm never pulled before

Algorithm to Exploration & Exploitation



1. Epsilon algorithm[1]
2. UCB1[2]

1. Ex3, Ex4
2. Tompson Sampling[3]
3. LinUCB[4]

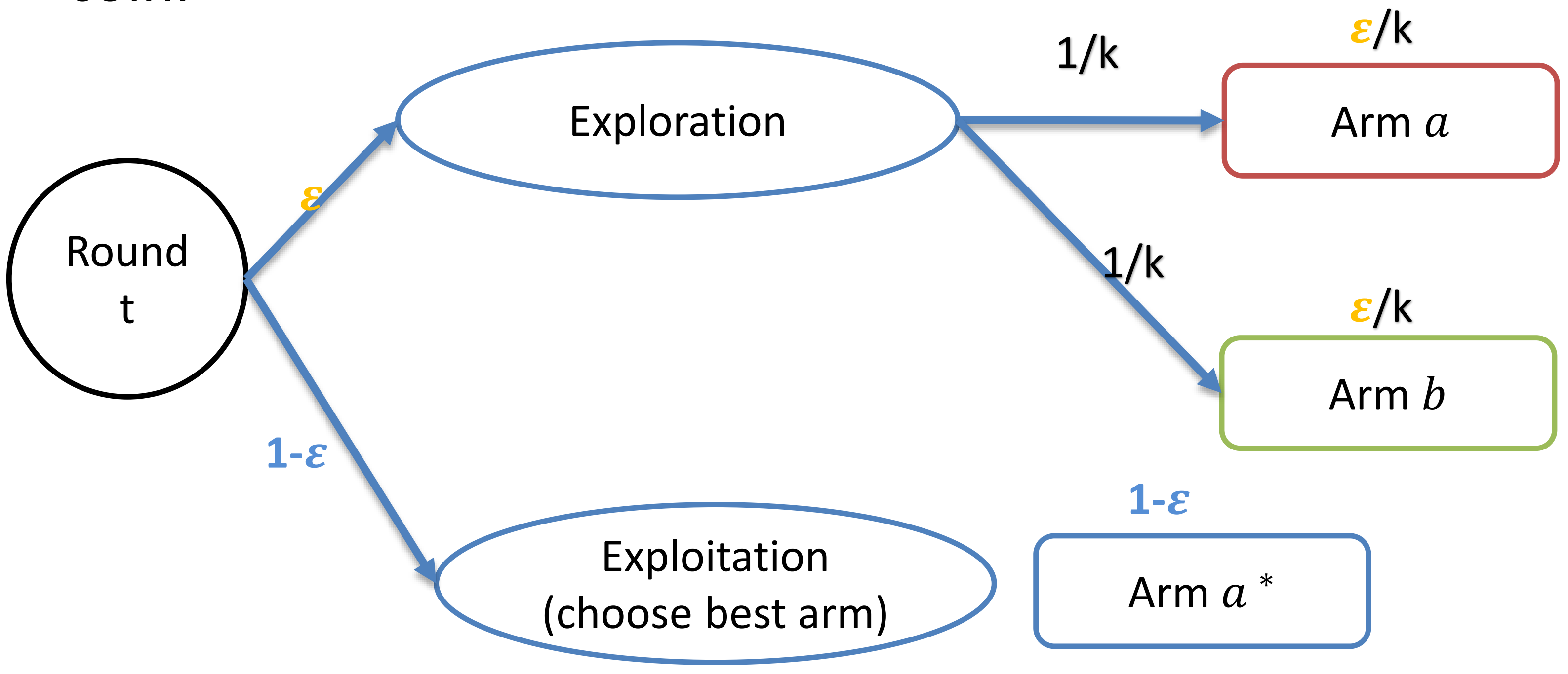
[1] Wynn P. On the convergence and stability of the epsilon algorithm[J]. SIAM Journal on Numerical Analysis, 1966, 3(1): 91-122.
[2] Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multi-armed bandit problem[J]. Machine learning, 2002, 47(2-3): 235-256.
[3] Agrawal S, Goyal N. Analysis of Thompson sampling for the multi-armed bandit problem[J]. arXiv preprint arXiv:1111.1797, 2011.
[4] Li, Lihong, et al. "A contextual-bandit approach to personalized news article recommendation." *Proceedings of the 19th international conference on World wide web*. ACM, 2010.

Contextual and Contextual- Free

- Contextual
  - Every round receives context
  - User features, arm/item features (e.g., articles reviewed before) at each trial
  - Select items to users based on contextual information about the user and the items
  - Bandits with covariate, bandits with side information, associative bandits
- Contextual-free
  - Both the arm set and contexts are constant at every trial

ε-Greedy Algorithm

- It tries to be fair to the two opposite goals of exploration(with prob. ε) and exploitation(1-ε) by using a mechanism: flips a coin.



ε-Greedy Algorithm

- For t=1:T
  - Set $\varepsilon_t = O\left(\frac{1}{t}\right)$
  - With prob. ε_t: Explore by picking an arm chosen uniformly at random
  - With prob. 1-ε_t: Exploit by picking an arm with highest empirical mean payoff
- Theorem [Auer et al. ‘02]
  - For suitable choice of ε_t it holds that

$$R_T = O(k \log T) \Rightarrow \frac{R_T}{T} = O\left(\frac{k \log T}{T}\right) \to 0$$

Issues with ε-Greedy Algorithm

- **Not elegant"** : Algorithm explicitly distinguishes between exploration and exploitation
- **More importantly**: Exploration makes **suboptimal choices**(since it picks any arm equally likely)

- Idea: When exploring/exploiting we need to compare arms.

Example : Comparing Arms

- **Suppose we have done experiments** :
  - **Arm 1**: 1 0 0 1 1 1 0 0 0 1
  - **Arm 2**: 1
  - **Arm 3**: 1 1 0 1 0 0 1 1 1 1
- **Mean arm values**:
  - Arm 1: 5/10 Arm 2: 1 Arm 3: 7/10

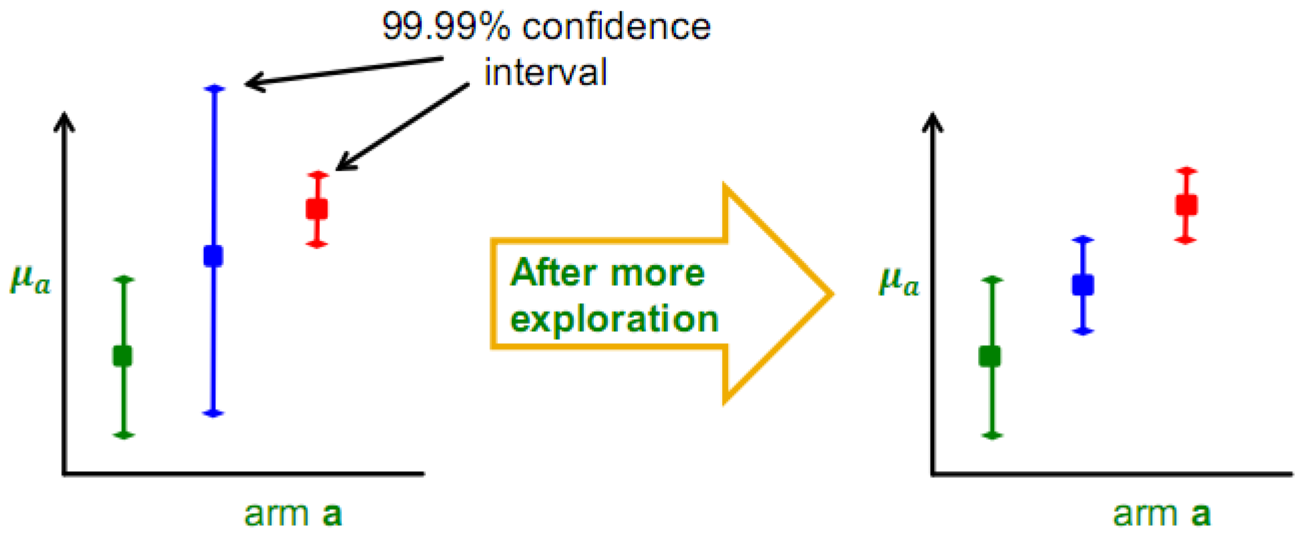
- Which arm would you choose next?
- Idea: Not only look at the mean but also the confidence!

Confidence Intervals

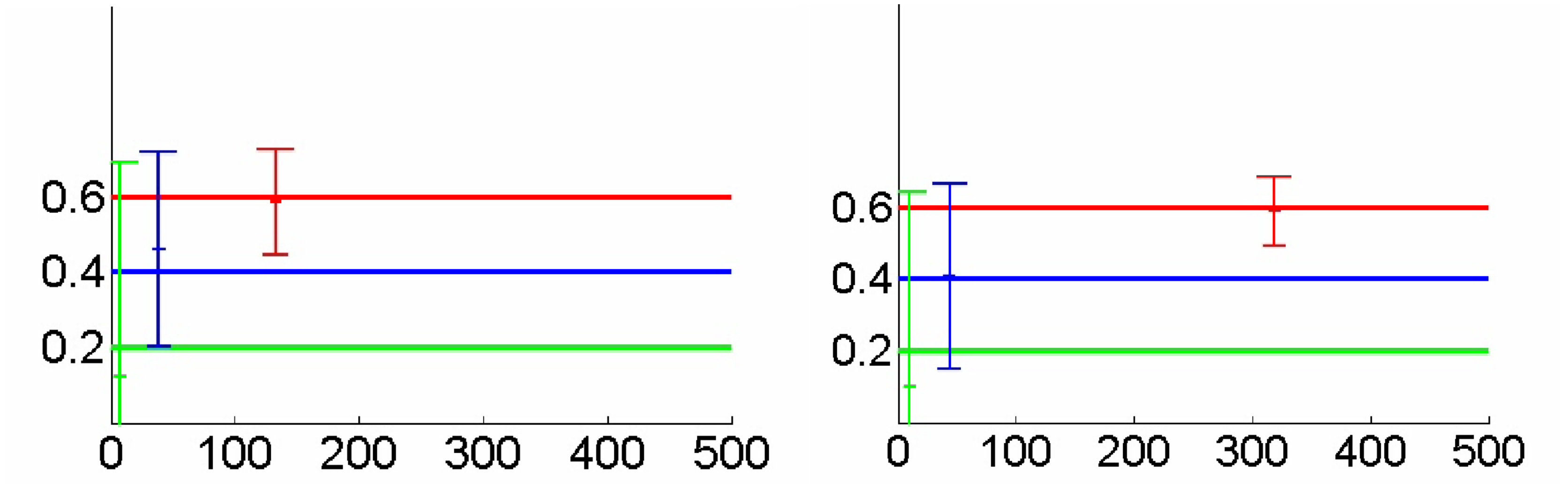
- **A confidence interval is a range of values within which we are sure the mean lies with a certain probability**
  - We could believe μ_a is within [0.2,0.5] with probability 0.95
  - If we would have tried an action less often, our estimated reward is less accurate so the confidence interval is larger
  - Interval shrinks as we get more information (try the action more often)

Confidence Based Selection

- Assuming we know the confidence intervals
- Then, instead of trying the action with the highest mean we can try the action with the highest upper bound on its confidence interval.



Confidence intervals vs Sampling times



The estimation of confidence becomes smaller as the number of pulling times increases.

[1] Jean-Yves Audibert and Remi Munos, Introduction to Bandits: Algorithms and Theory. ICML 2011, Bellevue(WA), USA

Calculating Confidence Bounds

- **Suppose we fix arm a:**
  - Let $r_{a,1}$ ... $r_{a,m}$ be the payoffs of arm a in the first m trials
    - $r_{a,1}$ ... $r_{a,m}$ are i.i.d. taking values in [0,1]
  - Our estimate : $\widehat{\mu_{a,m}} = \frac{\mathbf{1}}{m} \sum_{j=1}^{m} r_{a,j}$
  - Want to find b such that with high probability $\left|\mu_a - \widehat{\mu_{a,m}}\right| \le b$ (want b to be as small as possible)
  - Goal : Want to bound $\mathbf{P}(\left|\mu_a - \widehat{\mu_{a,m}}\right| \le b)$

UCB1 Algorithm

- **UCB1 (Upper confidence sampling) algorithm**
  - Let $\widehat{\mu_1} \ldots = \widehat{\mu_k} = 0$ and $m_1 = \ldots = m_k = 0$
    - $\widehat{\mu_a}$ is our estimate of payoff of arm a
    - m_a is the number of pulls of arm a so far.
  - For t = 1 : T

Hoeffding's Inequality

    - For each arm *a* calculate $\mathrm{UCB(a)} = \widehat{\mu_a} + \alpha\sqrt{\frac{2\ln t}{m_a}}$
    - Pick arm $j = argmax_a UCB(a)$
    - Pull arm *j* and observe y_t
    - $m_j = m_j + 1$ and $\widehat{\mu_j} = 1/m_j(y_t + (m_j - 1)\widehat{\mu_j})$

Hoeffding's Inequality

- **Hoeffding's inequality bounds** $\mathbf{P}\left(\left|\mu_a - \widehat{\mu_{a,m}}\right| \leq b\right)$
  - Let $X_1 \dots X_m$ be **i.i.d.** rnd. vars. taking values in **[0,1]**
  - Let $\mu = E[X]$ and $\widehat{\mu_m} = \frac{1}{m}\sum_{\ell=1}^{m} X_\ell$
  - **Then:** $\mathbf{P}(|\mu - \widehat{\mu_m}| \geq b) \leq 2\, exp(-2b^2 m) = \delta$
- **To find out the confidence interval *b* (for a given confidence level *δ*) we solve:**
  - $2e^{-2b^2 m} \leq \delta$ **then** $-2b^2 m \leq \ln(\delta/2)$
  - **So:** $b \geq \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2\,m}}$

[1] CS246 Mining Massive Data Sets 2015, Stanford University

UCB1 Algorithm: Discussion

- Confidence interval grows with the total number of actions *t* we have taken
- But Shrinks with the number of times m_a we have tried arm *a*
- This ensures each arm is tried infinitely often but still balances exploration and exploitation
- α plays the role of δ: $\alpha = \mathrm{f}\left(\frac{2}{\delta}\right) = 1 + \sqrt{\frac{\ln(2/\delta\ )}{2}}$
- For each arm *a* calculate $\mathrm{UCB(a)} = \widehat{\mu_a} + \alpha\sqrt{\frac{2\ln t}{m_a}}$
    - Pick arm $j = argmax_a UCB(a)$
    - Pull arm *j* and observe y_t
    - m_j = m_j + 1 and $\widehat{\mu_j}$ = $1/m_j(\ y_t + (\ m_j - 1)\,\widehat{\mu_j})$

UCB1 Algorithm Performance

- Theorem [Auer et al. 2002]
  - Suppose optimal mean payoff is $\mu^* = \max_a \mu_a$
  - And for each arm let $\Delta_a = \mu^* - \mu_a$
  - Then it holds that

$$E[R_T] = \left[8 \sum_{a:\mu_a<\mu^*} \frac{\ln T}{\Delta_a}\right] + \left(1 + \frac{\pi^2}{3}\right)\left(\sum_{i=a}^{k} \Delta_a\right)$$

$$O(k \ln T) \qquad\qquad O(k)$$

- So, we get $O\left(\frac{R_T}{T}\right) = k\,\frac{ln\,T}{T}$

Quick Summary

- Multi-armed bandit problem as a formalization the exploration-exploitation tradeoff

- Simple algorithms for context-free bandits are able to achieve no regret (in the limit)
  - ε-Greedy
  - UCB (Upper Confidence Sampling)

Contextual Bandits

- Contextual bandit algorithm in round t
  - Algorithm observers user $\boldsymbol{u_t}$ and a set $\mathbf{A}$ of arms together with their features $\boldsymbol{x_{t,a}}$(context)
  - Based on payoffs from previous trials, algorithm chooses arm $\boldsymbol{a} \in \mathbf{A}$ and receives payoff $\boldsymbol{r_{t,a}}$
  - Algorithm improves arm selection strategy with each observation($\boldsymbol{x_{t,a}}, \boldsymbol{a}, \boldsymbol{r_{t,a}}$)

LinUCB Algorithm[1]

- Contextual bandit algorithm in round t
  - Algorithm observers user $\boldsymbol{u_t}$ and a set $\mathbf{A}$ of arms together with their features $\boldsymbol{x_{t,a}}$(context)
  - Based on payoffs from previous trials, algorithm chooses arm $\boldsymbol{a} \in \mathbf{A}$ and receives payoff $\boldsymbol{r_{t,a}}$
  - Algorithm improves arm selection strategy with each observation($\boldsymbol{x_{t,a}}, \boldsymbol{a}, \boldsymbol{r_{t,a}}$)

[1] Li, Lihong, et al. "A contextual-bandit approach to personalized news article recommendation." *Proceedings of the 19th international conference on World wide web*. ACM, 2010.

LinUCB Algorithm

- Expectation of reward of each arm is modeled as a linear function of the context.

$\boldsymbol{\theta_a^*}$ is the unknown coefficient vector we **aim to learn**

Payoff of arm a : $\mathrm{E}\left[r_{t,a} \middle| x_{t,a}\right] = [x_{t,a}]^T \theta_a^*$

$\boldsymbol{x_{t,a}}$ is a **d**-dimensional feature vector

- The goal is to minimize regret, defined as the difference between the expectation of the reward of best arms and the expectation of the reward of selected arms.

$$R_t(T) \stackrel{\text{def}}{=} E\left[\sum_{t=1}^{T} r_{t,a_t^*}\right] - E[\sum_{t=1}^{T} r_{t,a_t}]$$

LinUCB Algorithm

- $\mathrm{E}\left[r_{t,a} \middle| x_{t,a}\right] = [x_{t,a}]^T \theta_a^*$
  - How to estimate θ_a?
    - Linear regression solution to θ_a is

$$\widehat{\boldsymbol{\theta_a}} = \boldsymbol{argmin_\theta} \sum_{\boldsymbol{m \in D_a}} ([x_{t,a}]^T \theta_a - \boldsymbol{b_a^{(m)}})^2$$

We can get:

$$\widehat{\boldsymbol{\theta_a}} = (\boldsymbol{D_a^T D_a} + \boldsymbol{I_d})^{-1} \boldsymbol{D_a^T b_a}$$

$\boldsymbol{D_a}$ is a m × d matrix of **m** training inputs $[x_{t,a}]$

$\boldsymbol{b_a}$ is a m-dimension vector of responses to $\boldsymbol{a}$(click/no-click)

LinUCB Algorithm

- Using similar techniques as we used for UCB

$$\left|[x_{t,a}]^T\widehat{\boldsymbol{\theta}_{\boldsymbol{a}}} - \mathrm{E}\left[r_{t,a}\middle|x_{t,a}\right]\right| \leq \boldsymbol{\alpha}\sqrt{[x_{t,a}]^T(\boldsymbol{D}_{\boldsymbol{a}}^{\boldsymbol{T}}\boldsymbol{D}_{\boldsymbol{a}}+\boldsymbol{I}_{\boldsymbol{d}})^{-\mathbf{1}}x_{t,a}}$$

$$\boldsymbol{\alpha} = \mathbf{1} + \sqrt{\boldsymbol{ln}(\mathbf{2}/\boldsymbol{\delta})/\mathbf{2}}$$

- For a given context, we estimate the reward and the confidence interval.

$$\underline{\boldsymbol{a}_{\boldsymbol{t}} \overset{\text{def}}{=} \boldsymbol{argmax}_{a\in A_t}\Big(\underbrace{[x_{t,a}]^T\widehat{\boldsymbol{\theta}_{\boldsymbol{a}}}}_{\text{Estimated } \mu_a} + \boldsymbol{\alpha}\underbrace{\sqrt{[x_{t,a}]^T(\boldsymbol{D}_{\boldsymbol{a}}^{\boldsymbol{T}}\boldsymbol{D}_{\boldsymbol{a}}+\boldsymbol{I}_{\boldsymbol{d}})^{-\mathbf{1}}x_{t,a}}}_{\text{Confidence interval}}\Big)}$$

LinUCB Algorithm

$A_a \overset{\text{def}}{=} \boldsymbol{D_a^T D_a + I_d}$

```
Initialization:
    For each arm $a$:
        $A_a = I_d$                                   //identity matrix d ×d
        $b_a = [0]_d$                                 //vector of zeros
Online algorithm:
    For t=[1:T]:
        Observe features for all arms $a$ : $x_{t,a} \in R^d$
        For each arm $a$ :
            $\theta_a = A_a^{-1} b_a$                 //regression coefficients
            $p_{t,a} = [x_{t,a}]^T \theta_a + \boldsymbol{\alpha}\sqrt{[x_{t,a}]^T A_a^{-1} x_{t,a}}$
        Choose arm $a_t = argmax_a p_{t,a}$           //choose arm
        $A_{a_t} = A_{a_t} + x_{t,a_t}[x_{t,a_t}]^T$  //update A for the chosen arm $a_t$
        $b_{a_t} = b_{a_t} + r_t\, x_{t,a_t}$         //update b for the chosen arm $a_t$
```

Different between UCB1 and LinUCB

- **UCB1** directly estimates μ_a through experimentation (without any knowledge about arm *a*)
- **LinUCB** estimates μ_a by regression $\mu_a = [x_{t,a}]^T \boldsymbol{\theta}_{\boldsymbol{a}}^*$
  - The hope is that we will be able to learn faster as we consider the context $x_a(\text{user}, \text{ad})$ of arm *a*
  - $\boldsymbol{\theta}_{\boldsymbol{a}}^*$ unknown coefficient vector we aim to learn

LinUCB: Discussion

- LinUCB computational complexity is
  - Linear in the number of arms and
  - At most cubic in the number of features
- LinUCB works well for a dynamic arm set(arms come and go)
  - For example, in news article recommendation, for instance, editors add/remove articles to/from a pool

Thompson Sampling

- A simple natural Bayesian heuristic
  - Maintain a belief(distribution) for the unknown parameters
  - Each time, pull arm a and observe a reward r

- Initialize priors using belief distribution
  - For t=1:T:
    - Sample random variable X from each arm's belief distribution
    - Select the arm with largest X
    - Observe the result of selected arm
    - Update prior belief distribution for selected arm

[1] Agrawal S, Goyal N. Analysis of Thompson sampling for the multi-armed bandit problem[J]. arXiv preprint arXiv:1111.1797, 2011.

Simple Example

- Coin toss: x ~ Bernoulli(θ)
- Let's assume that
  - θ ~ Beta(α_H, α_T)
  - P(θ) $\propto \theta^{\alpha_H - 1}(1-\theta)^{\alpha_T - 1}$

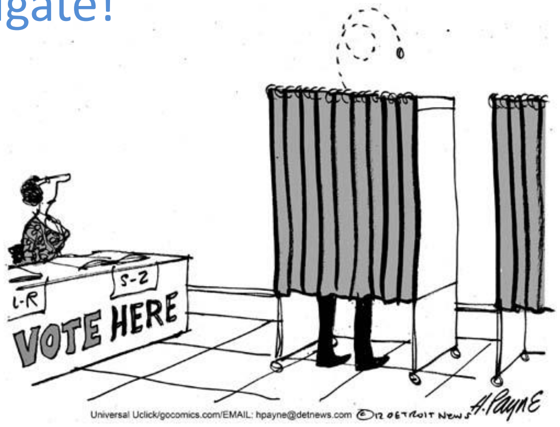
Beta distribution

Prior

- $P(\theta|X) = \frac{P(X|\theta)P(\theta)}{\sum_\theta P(X|\theta)}$

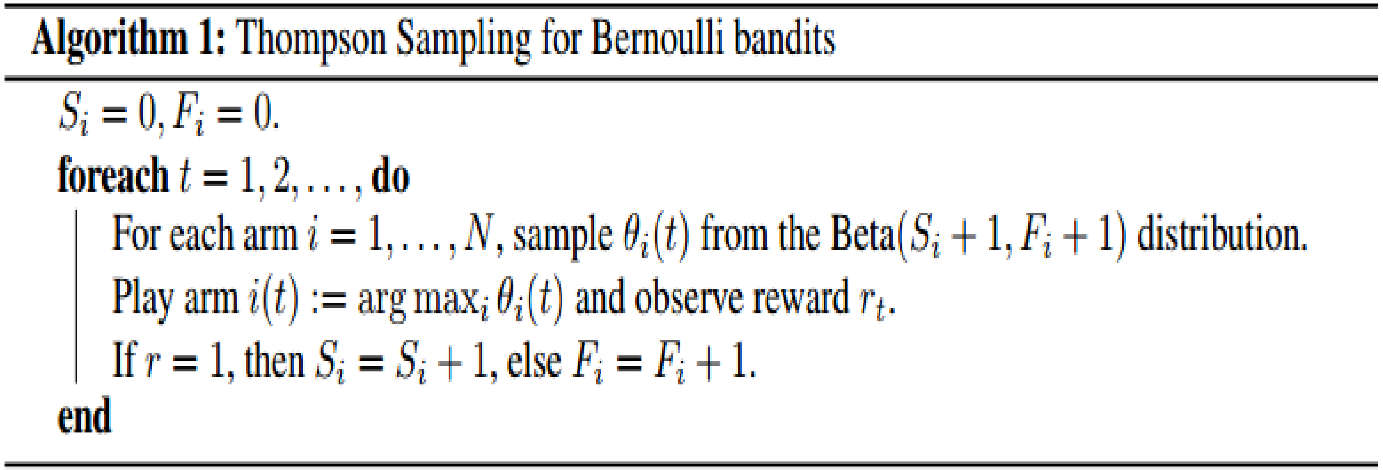
Posterior

The prior is conjugate!


L-R
S-Z
VOTE HERE
Universal Uclick/gocomics.com/EMAIL: hpayne@detnews.com ©12 DETROIT NEWS
H. PAYNE

Thompson Sampling Using Beta belief distribution

- Theorem [Emilie et al. 2012]
  - Initially assumes arm $\boldsymbol{i}$ with prior Beta(1,1) on $\boldsymbol{\mu_i}$
  - S_i = #“Success”, F_i= #“Failure”



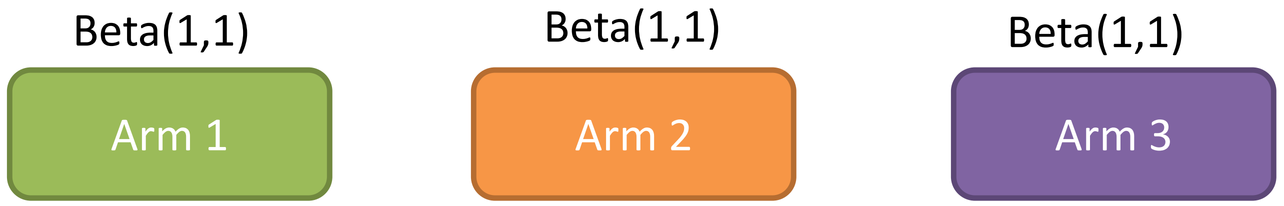
Algorithm 1: Thompson Sampling for Bernoulli bandits

```
$S_i = 0, F_i = 0.$
foreach $t = 1, 2, \ldots,$ do
    For each arm $i = 1, \ldots, N$, sample $\theta_i(t)$ from the Beta$(S_i + 1, F_i + 1)$ distribution.
    Play arm $i(t) := \arg\max_i \theta_i(t)$ and observe reward $r_t$.
    If $r = 1$, then $S_i = S_i + 1$, else $F_i = F_i + 1$.
end
```

- This posterior distribution could then be used as the prior for more samples, with the hyperparameters simply adding each extra piece of information as it comes.

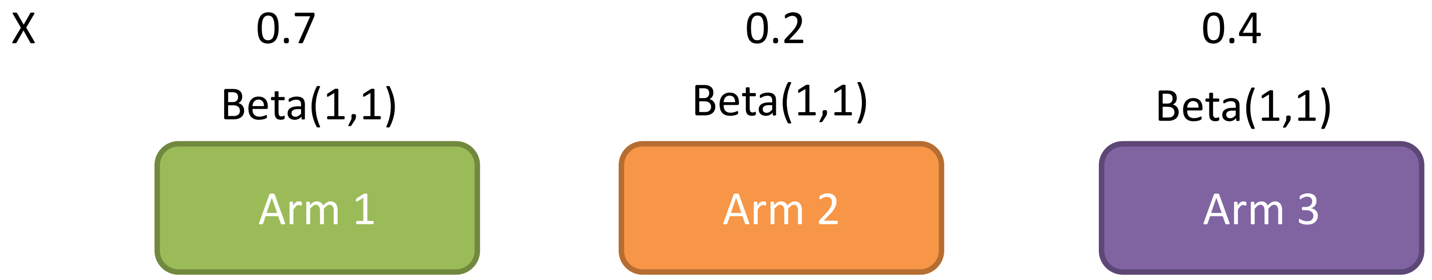
Thompson Sampling
Using Beta belief distribution

- Initialization



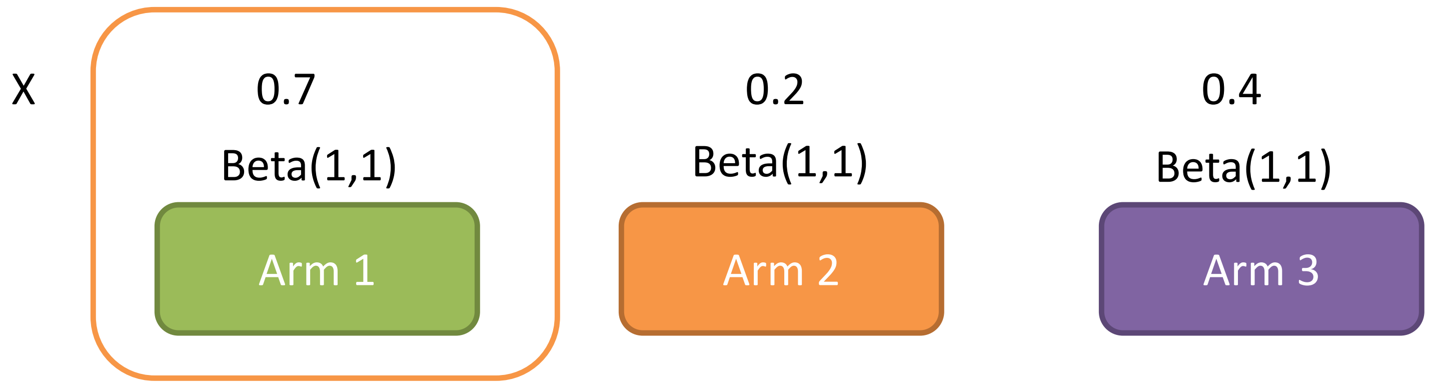
Thompson Sampling
Using Beta belief distribution

- For each round:
    - Sample random variable X from each arm's Beta Distribution



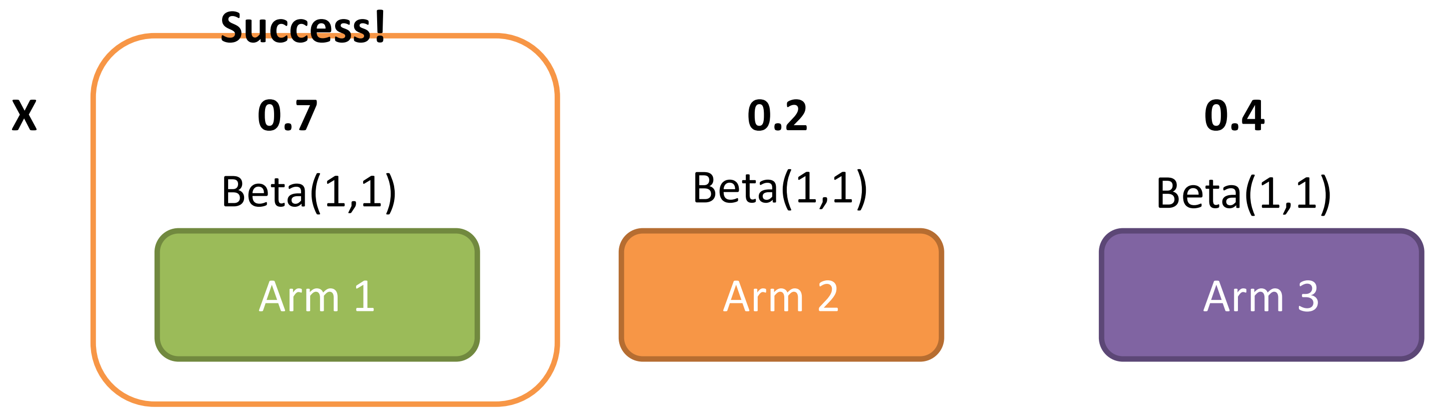
Thompson Sampling
Using Beta belief distribution

- For each round:
  - Sample random variable X from each arm's Beta Distribution
  - Select the arm with largest X



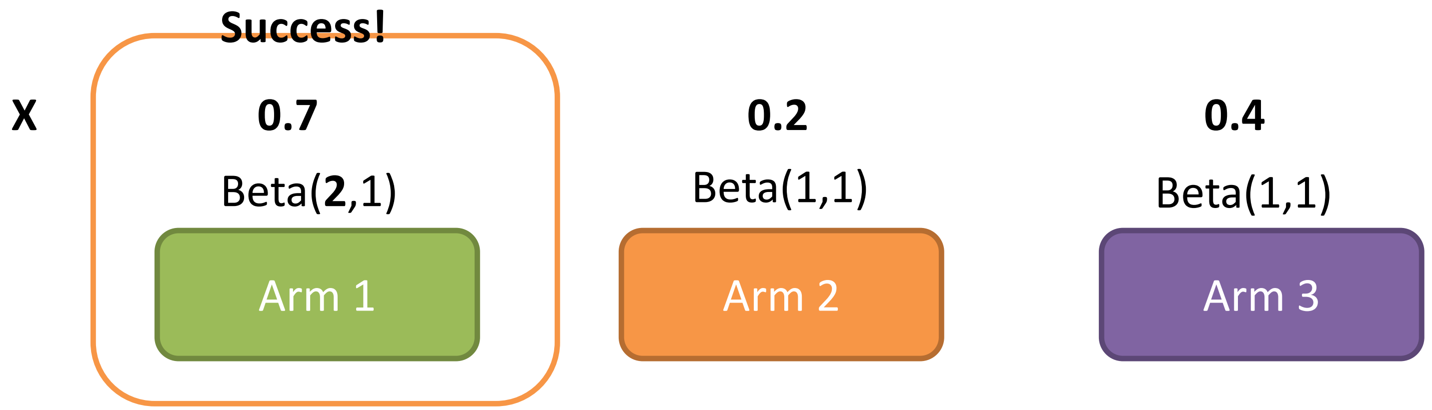
Thompson Sampling
Using Beta belief distribution

- For each round:
  - Sample random variable X from each arm's Beta Distribution
  - Select the arm with largest X
  - Observe the result of selected arm



Thompson Sampling
Using Beta belief distribution

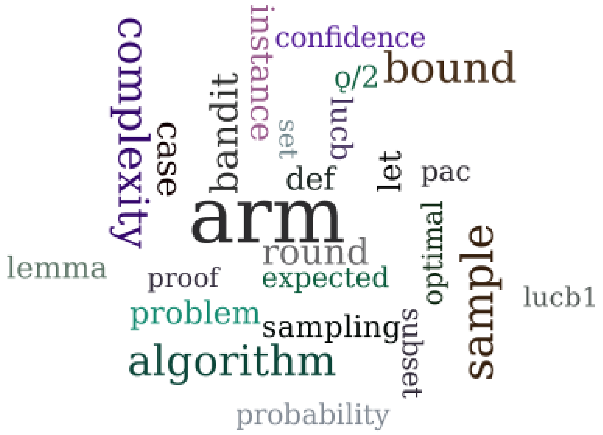
- For each round:
  - Sample random variable X from each arm's Beta Distribution
  - Select the arm with largest X
  - Observe the result of selected arm
  - Update prior Beta distribution for selected arm



Our Recent Research Studies

- Ensemble Contextual Bandits for Personalized Recommendation (RecSys 2014)
- Parameter-free Contextual Bandits for Personalized Recommendation (SIGIR 2015)

Ensemble Contextual Bandits for Personalized Recommendation



[1] Tang, Liang, et al. "Ensemble contextual bandits for personalized recommendation." *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014.

Problem Statement

- **Problem Setting:** have many different recommendation models (or policies):
  - Different CTR Prediction Algorithms.
  - Different Exploration-Exploitation Algorithms.
  - Different Parameter Choices.
- **No data to do model validation**
- **Problem Statement**: how to build an ensemble model that is close to the best model in the cold start situation ?

How Ensemble?

- Classifier ensemble method does not work in this setting
  - Recommendation decision is NOT purely based on the predicted CTR.
- Each individual model only tells us:
  - Which item to recommend.

Not appropriate to adopt majority voting or consensus prediction as the ensemble

Ensemble Method

- Our Method:
  - Allocate recommendation chances to individual models.
- Problem:
  - Better models should have more chances.
  - We do not know which one is good or bad in advance.
  - Ideal solution: allocate all chances to the best one.

Current Practice: Online Evaluation (or A/B testing)

- Let $\pi_1, \pi_2 \ \ldots \pi_m$ be the individual models.
  - Deploy $\pi_1, \pi_2 \ \ldots \pi_m$ into the online system at the same time.
  - Dispatch a small percent user traffic to each model.
  - After a period, choose the model having the best CTR as the production model.

Current Practice: Online Evaluation (or A/B testing)

- Let $\pi_1, \pi_2 \dots \pi_m$ be the individual models.
  - Deploy $\pi_1, \pi_2 \dots \pi_m$ into the online system at the same time.
  - Dispatch a small percent user traffic to each model.
  - After a period, choose the model having the best CTR as the production model.

If we have too many models, this will hurt the performance of the online system.

Our Idea 1 (HyperTS)

- The CTR of model π_i is a random unknown variable, R_i .
- **Goal**:
  - maximize $\frac{1}{N}\sum_{t=1}^{N} r_t$

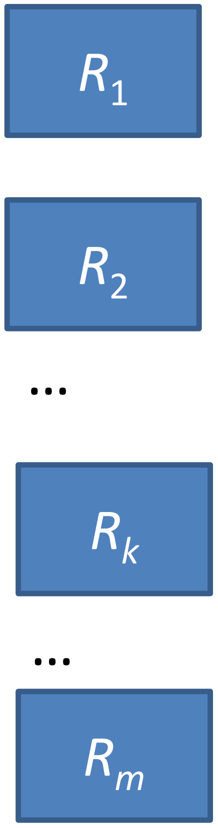
    CTR of our ensemble model

    r_t is a random number drawn from $R_{s(t)}$, $s(t)$=1,2,…, or m. For each t=1,…,N, we decide $s(t)$.
- **Solution**:
  - *Bernoulli Thompson Sampling* (flat prior: beta(1,1)) .

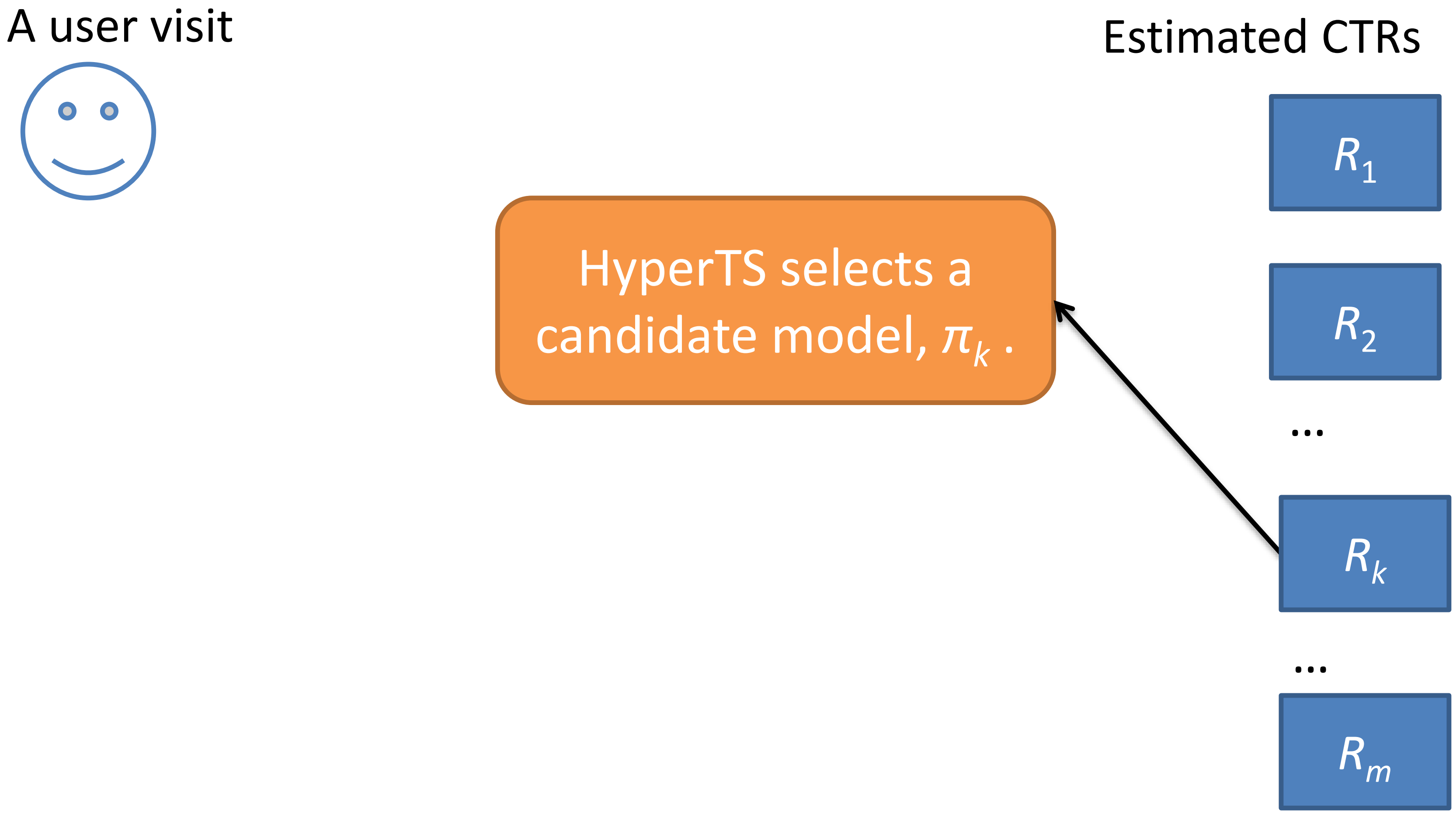
    No tricky parameters
  - $\pi_1, \pi_2 \ldots \pi_m$ are bandit arms.

An Example of HyperTS

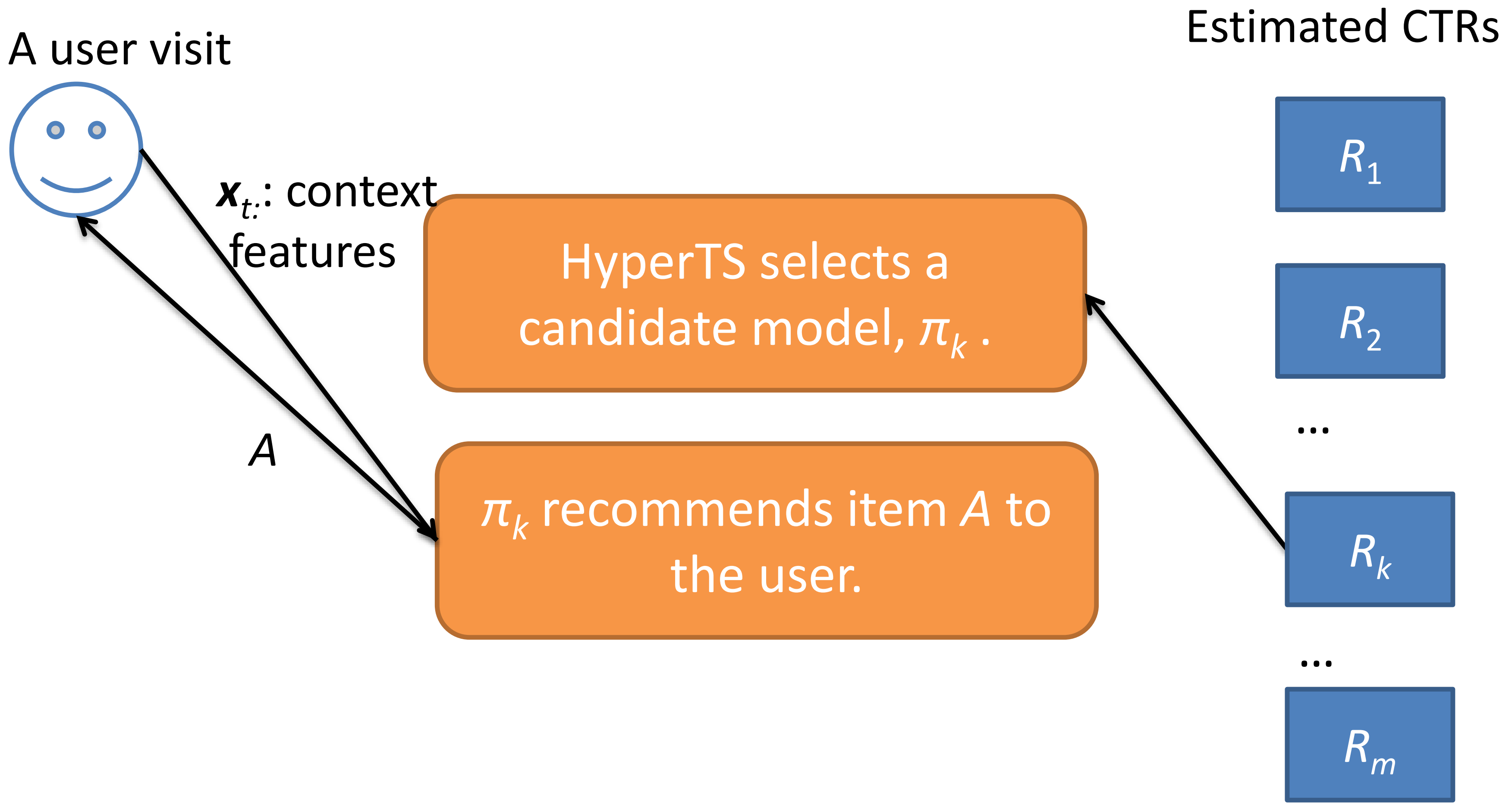
In memory, we keep these estimated CTRs for π_1, π_2 … $\pi_{m.}$



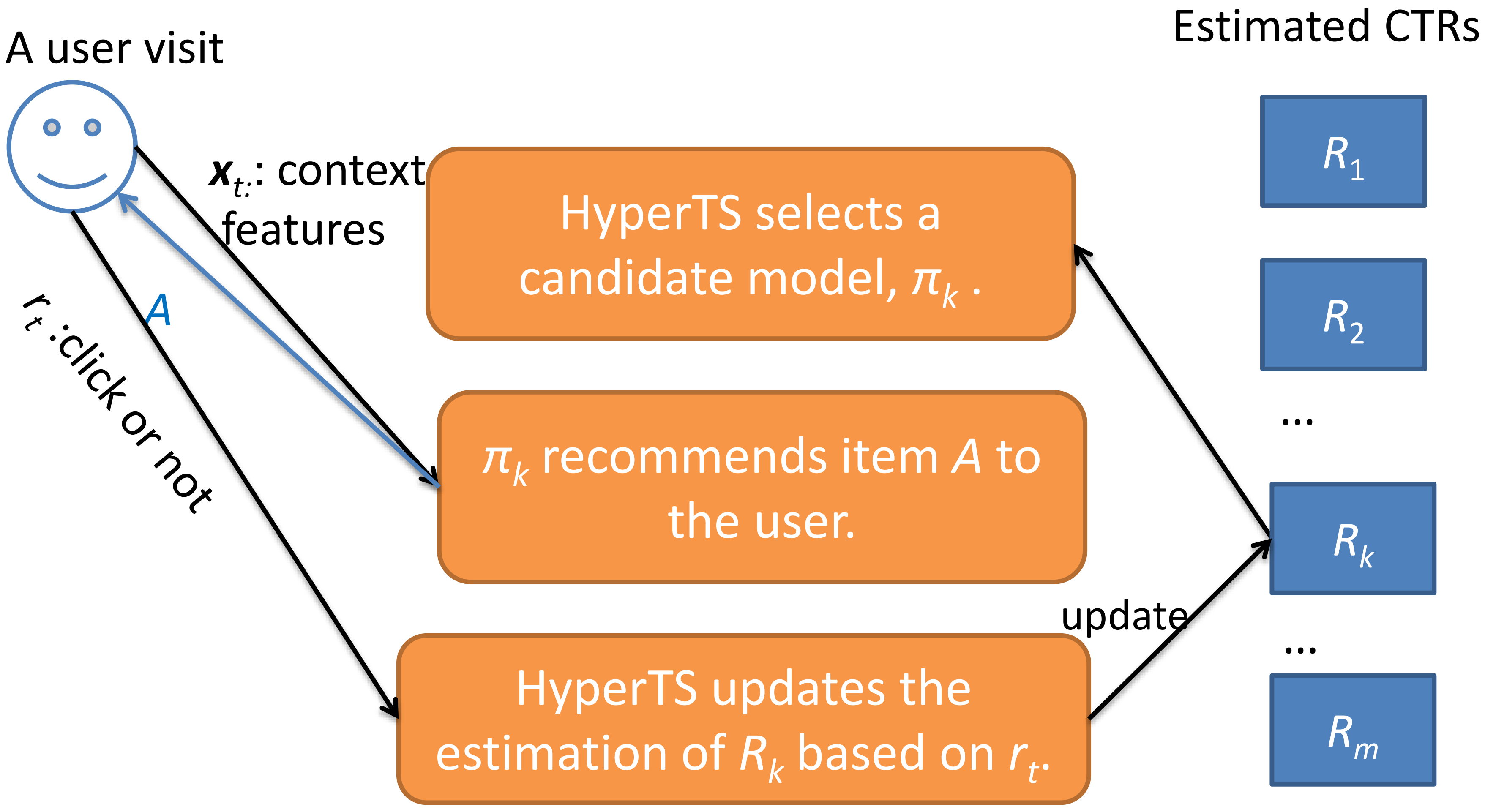
An Example of HyperTS



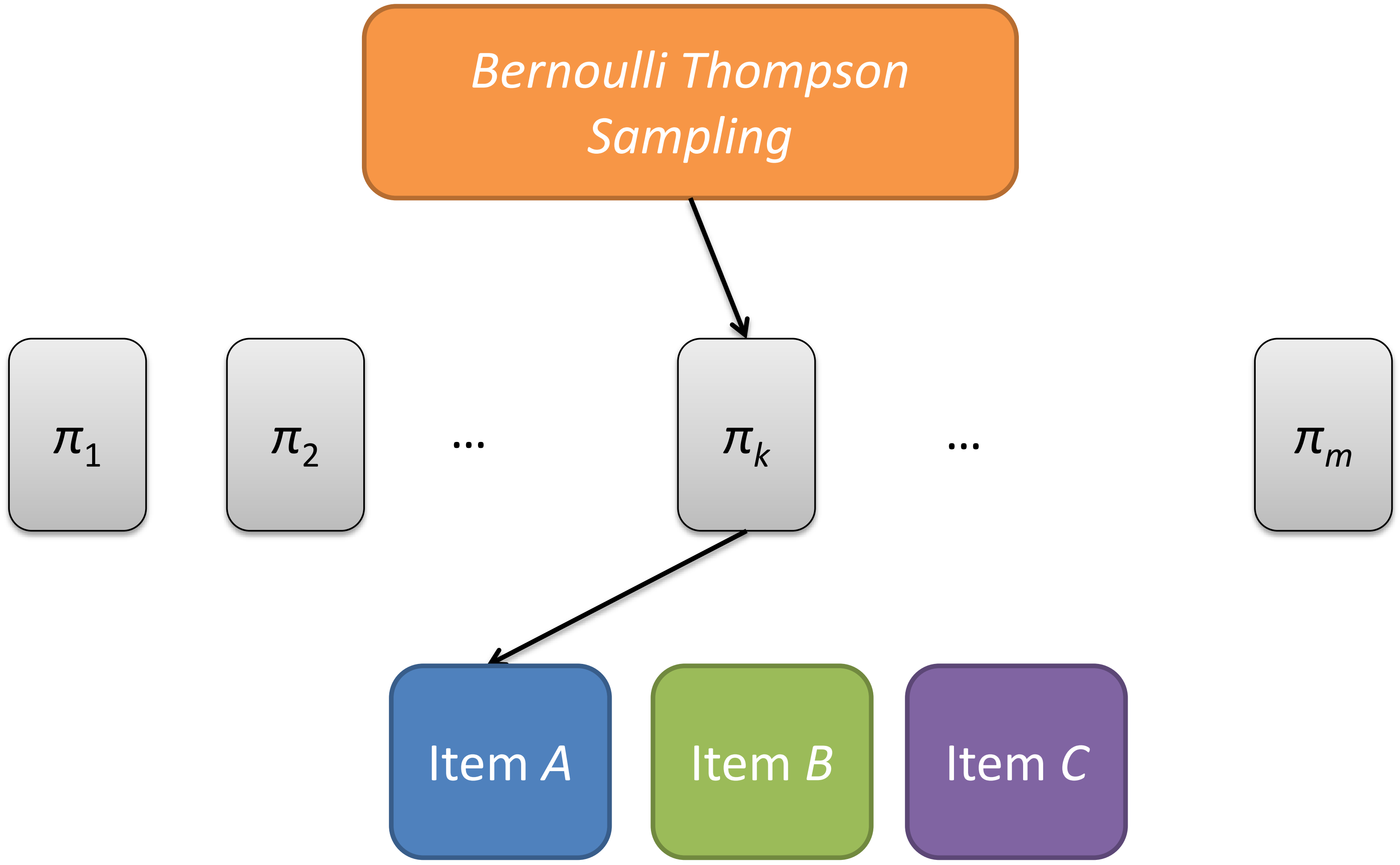
An Example of HyperTS


A user visit
$\boldsymbol{x}_{t:}$: context features
HyperTS selects a candidate model, π_k .
A
π_k recommends item A to the user.
Estimated CTRs
R_1
R_2
…
R_k
…
R_m

An Example of HyperTS


A user visit
Estimated CTRs
$\boldsymbol{x}_{t:}$: context features
HyperTS selects a candidate model, π_k .
π_k recommends item A to the user.
HyperTS updates the estimation of R_k based on r_t.
A
r_t :click or not
update
R_1
R_2
…
R_k
…
R_m

Two-Layer Decision



Our Idea 2 (HyperTSFB)

- Limitation of Previous Idea:
    - For each recommendation, user feedback is used by only one individual model (e.g., π_k).

- Motivation:
    - Can we update all R_1, R_2, …, R_m by every user feedback? (Share every user feedback to every individual model).

Our Idea 2 (HyperTSFB)

- Assume each model can output the probability of recommending any item given $\boldsymbol{x}_t$.
  - E.g., for deterministic recommendation, it is 1 or 0.
- For a user visit $\boldsymbol{x}_t$:
  - π_k is selected to perform recommendation (k=1,2,…, or m).
  - Item A is recommended by π_k given $\boldsymbol{x}_t$.
  - Receive a user feedback (click or not click), $r_{t.}$
  - Ask every model π_1, π_2 … π_m, what is the probability of recommending A given $\boldsymbol{x}_{t.}$

Our Idea 2 (HyperTSFB)

- Assume each model can output the probability of recommending any item given $\boldsymbol{x}_t$.
  - E.g., for deterministic recommendation, it is 1 or 0.

- For a user visit $\boldsymbol{x}_t$:
  - π_k is selected to perforr
  - Item A is recommendec
  - Receive a user feedback (click or not click), r_t.
  - Ask every model $\pi_1, \pi_2 \ldots \pi_m$, what is the probability of recommending A given $\boldsymbol{x}_t$.

Estimate the CTR of $\pi_1, \pi_2 \ldots \pi_m$ (Importance Sampling)

Experimental Setup

- **Experimental Data**
  - Yahoo! Today News data logs (randomly displayed).
  - KDD Cup 2012 Online Advertising data set.

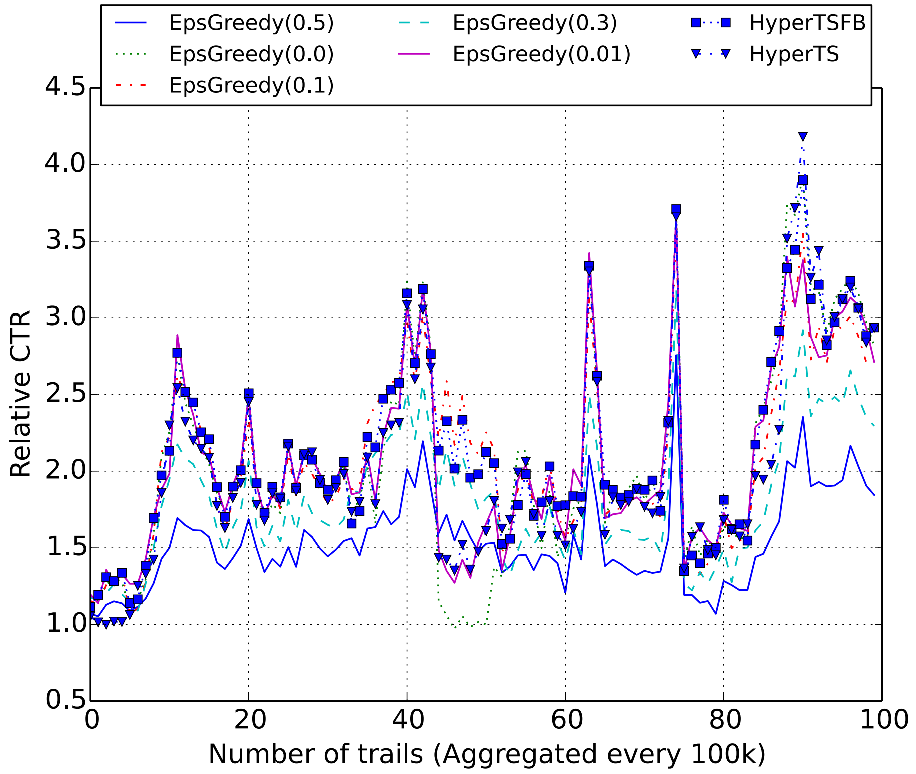
- **Evaluation Methods**
  - Yahoo! Today News: *Replayer* (see Lihong Li et. al's WSDM 2011 paper).
  - KDD Cup 2012 Data: *Simulation* by a Logistic Regression Model.

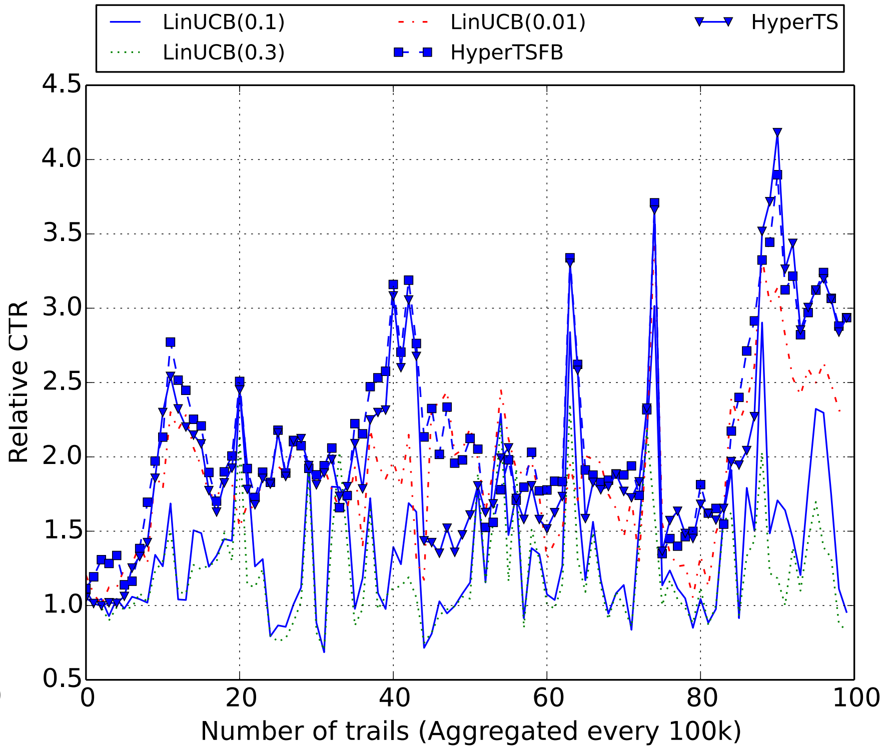
Comparative Methods

- CTR Prediction Algorithm
  - Logistic Regression
- Exploitation-Exploration Algorithms
  - Random, ε-greedy, LinUCB, Softmax, Epoch-greedy, Thompson sampling
- HyperTS and HyperTSFB

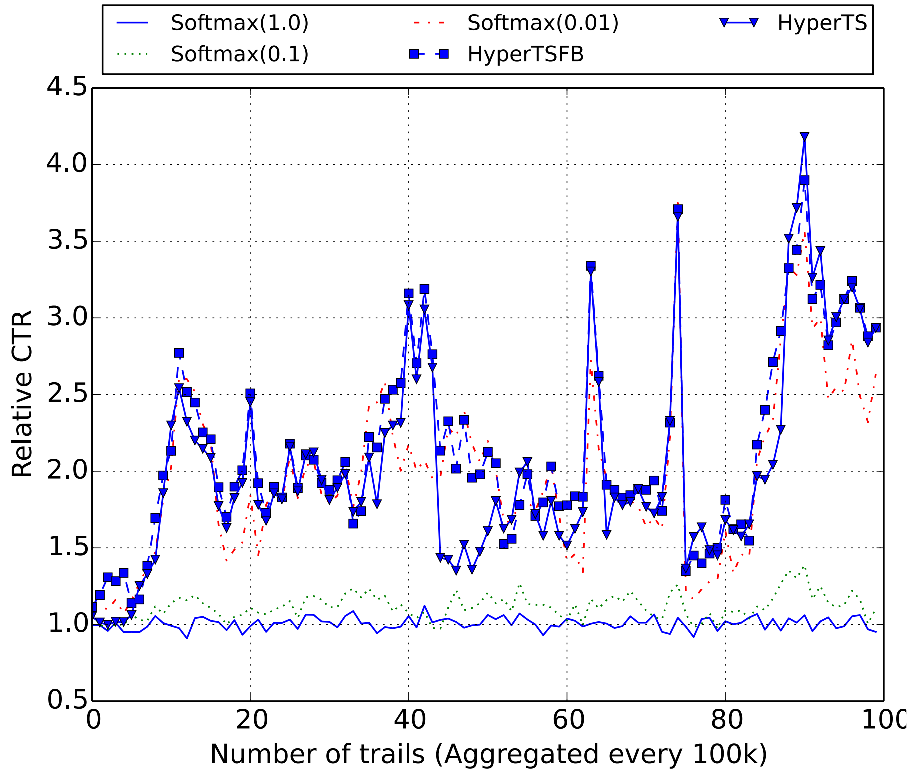
Results for Yahoo! News Data

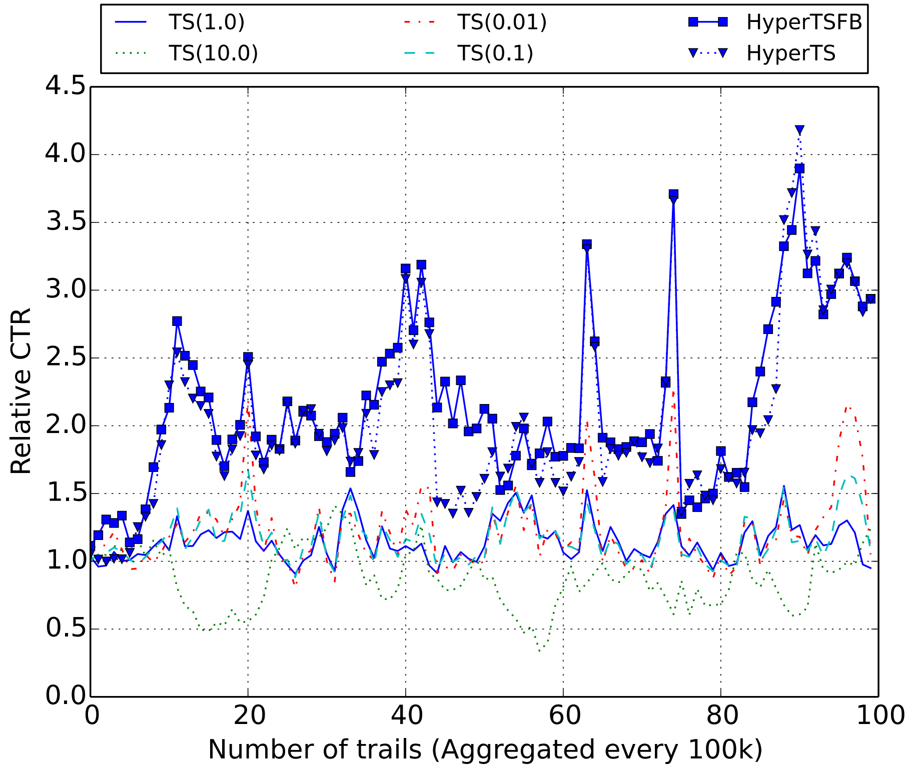
- Every 100,000 impressions are aggregated into a bucket.





Results for Yahoo! News Data (Cont.)





Conclusions

- The performance of baseline exploitation-exploration algorithms is very sensitive to the parameter setting.
  - In cold-start situation, no enough data to tune parameter.
- HyperTS and HyperTSFB can be close to the optimal baseline algorithm (No guarantee be better than the optimal one), even though some bad individual models are included.

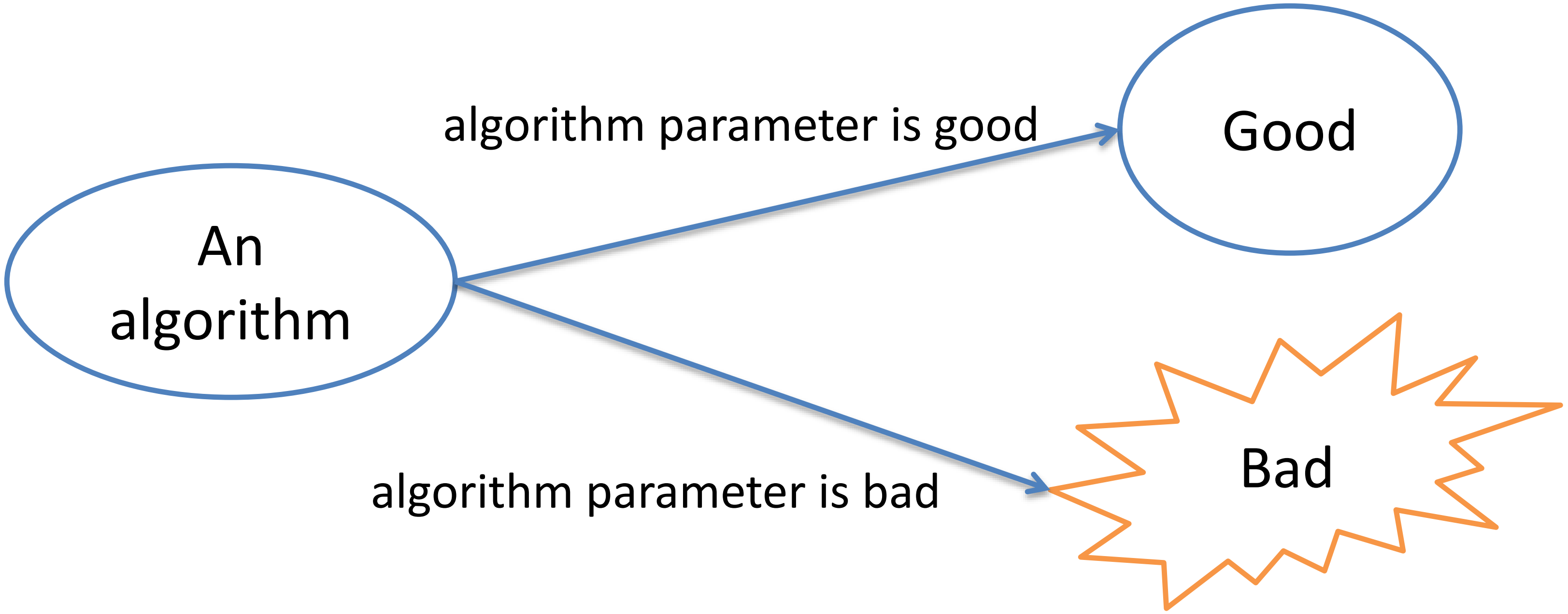
- For contextual Thompson sampling, the performance depends on the choice of prior distribution for the logistic regression.
  - For online Bayesian learning, the posterior distribution approximation is not accurate(cannot store the past data).

Personalized Recommendation via Parameter-Free Contextual Bandits

[1] Tang, Liang, et al. "Personalized recommendation via parameter-free contextual bandits." *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015.

How to Balance Tradeoff

- Performance is mainly determined by the tradeoff. Existing algorithms find the tradeoff by user input parameters and data characteristics (e.g., variance of the estimated reward).
- Existing algorithms are all parameter-sensitive.



Chicken-and-Egg Problem for Existing Bandit Algorithms

- Why we use bandit algorithms?
  - Solve the cold start problem (No enough data for estimating user preferences).
- How to find the best input parameters?
  - Tune the parameters online or offline.

if you already have the data or online traffic to tune the parameters, why do you need bandit algorithms?

Our Work

- Parameter-free:
  - It can find the tradeoff by data characteristics automatically.
- Robust:
  - Existing algorithm can have very bad performance if the input parameter is not appropriate.

Solution

- Thompson Sampling
  - Randomly selecting a model coefficient vector from posterior distribution and find the “best” item.
  - Prior is the input parameter for computing posterior.

- Non-Bayesian Thompson Sampling (**Our Solution**)
  - Randomly selecting a bootstrap sample to find the MLE of model coefficient and find the “best” item.
  - Bootstrapping has no input parameter.

Bootstrap Bandit Algorithm

Input : a feature vector $\boldsymbol{x}$ of the context.
Algorithm:

```
if each article has sufficient observations then {
  for each article $i$=1,…, $k$
      i.   $D^i \Leftarrow$ randomly sample $\boldsymbol{n_k}$ impression data of article $i$ with
           replacement // Generate a bootstrap sample
      ii.  $\boldsymbol{\theta}_i \Leftarrow$ MLE coefficient of $D^i$ // Model estimation on bootstrap sample
  select the article $i^* = \operatorname{argmax}(f(\boldsymbol{x}, \boldsymbol{\theta}_i))$, $i$=1,…, $k$. to show.
}
else
{
  randomly select an article that has no sufficient observations to show.
}
```

Prediction function

Online Bootstrap Bandits

- Why Online Bootstrap?
  - Inefficient to generate a bootstrap sample for each recommendation.
- How to online bootstrap?
  - Keep the coefficient estimated by each bootstrap sample in memory.
  - No need to keep all bootstrap samples in memory.
  - When a new data arrives, incrementally update the estimated coefficient for each bootstrap sample **[1]**.

[1] N. C. Oza and S. Russell. Online bagging and boosting. In IEEE international conference on Systems, man and cybernetics, volume 3, pages 2340–2345, 2005.

Experiment Data

- Two public data sets
  - News recommendation data (Yahoo! Today News)
    - News displayed on the Yahoo! Front Page from Oct. 2nd, 2011 to Oct. 16th 2011.
    - 28,041,015 user visit events.
    - 136 dimensions of feature vector for each event.
  - Online advertising data (KDD Cup 2012, Track 2)
    - The data set is collected by a search engine and published by KDD Cup 2012.
    - 1 million user visit events.
    - 1,070,866 dimensions of the context feature vector.

Offline Evaluation Metric and Methods

- Setup
  - Overall CTR (average reward of a trial).

- Evaluation Method
  - The experiment on Yahoo! Today News is evaluated by the *replay* method **[1]**.
  - The reward on KDD Cup 2012 AD data is simulated with a weight vector for each AD **[2]**.

[1] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In WSDM, pages 297–306, 2011.
[2] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In NIPS, pages 2249–2257, 2011.

Experimental Methods

- Our method
  - Bootstrap(B), where B is the number of bootstrap samples.
- Baselines
  - Random: it randomly selects an arm to pull.
  - Exploit: it only consider the exploitation without exploration.
  - ε-greedy(ε): ε is the probability of exploration.
  - LinUCB(α): it pulls the arm with largest score defined by the parameter α
  - TS(q_0): Thompson sampling with logistic regression, where q_0^{-1} is the prior variance, 0 is the prior mean.
  - TSNR(q_0): Similar to TS(q_0), but the logistic regression is not regularized by the prior.

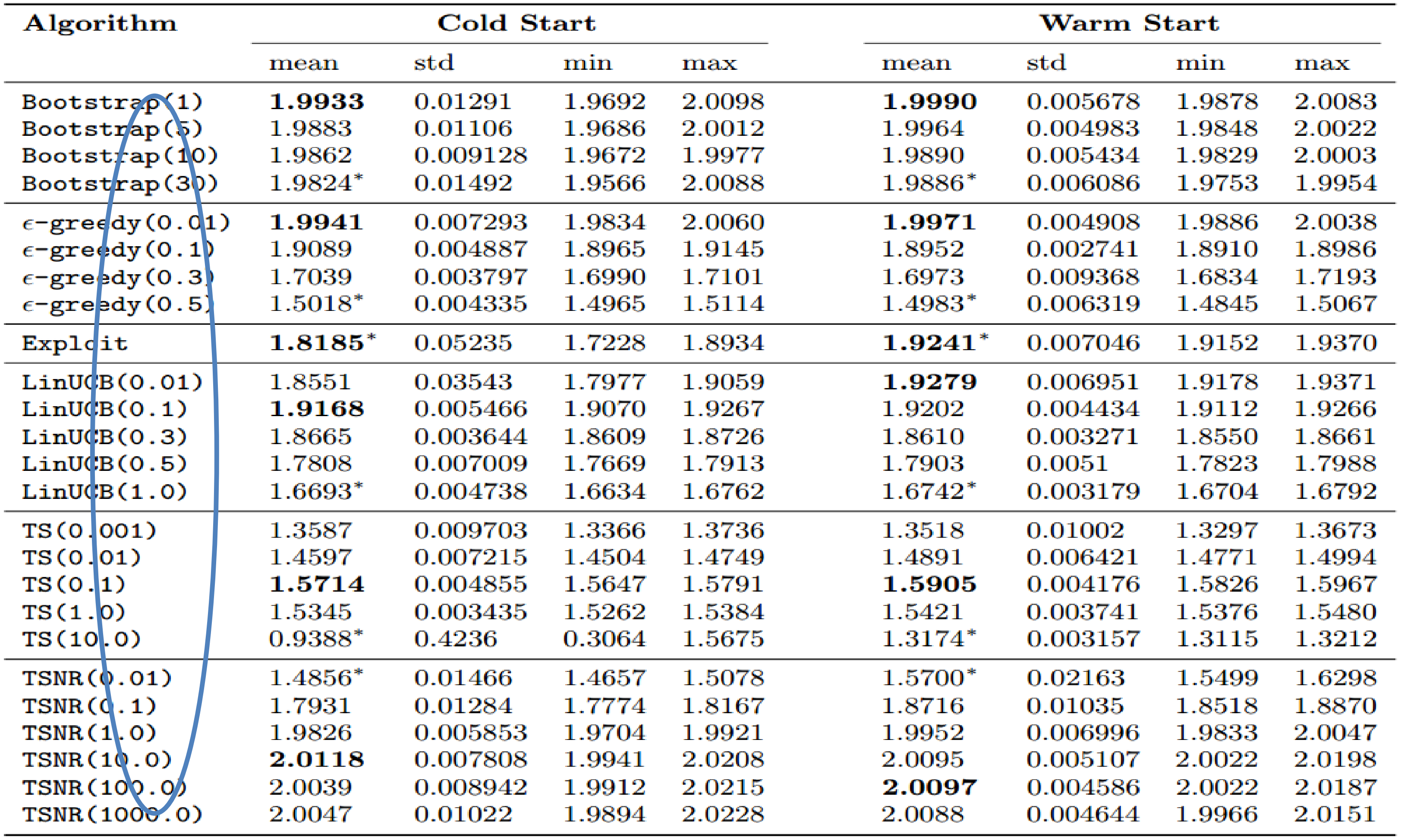
Experiment(Yahoo! News Data)

- All numbers are relative to the random model.

Algorithm	Cold Start				Warm Start			
	mean	std	min	max	mean	std	min	max
Bootstrap(1)	1.7350*	0.08327	1.6032	1.9123	1.7029*	0.1392	1.4299	1.8358
Bootstrap(5)	**1.8025**	0.07676	1.6526	1.9127	1.8366	0.07996	1.7118	1.9514
Bootstrap(10)	1.7536	0.07772	1.6338	1.8814	**1.8403**	0.08518	1.6673	1.9296
Bootstrap(30)	1.7818	0.08857	1.6092	1.9025	1.8311	0.08699	1.7230	1.9396
ϵ-greedy(0.01)	**1.7708**	0.09383	1.6374	1.9503	**1.8466**	0.05494	1.7846	1.9755
ϵ-greedy(0.1)	1.7375	0.04992	1.6452	1.8003	1.8132	0.03502	1.7621	1.8721
ϵ-greedy(0.3)	1.5486	0.03703	1.4812	1.5930	1.5976	0.02739	1.5591	1.6491
ϵ-greedy(0.5)	1.3819*	0.02341	1.3489	1.4169	1.3753*	0.02884	1.3173	1.4020
Exploit	**1.1782***	0.2449	0.9253	1.5724	**1.1576***	0.00198	1.1554	1.1607
LinUCB(0.01)	**1.6349**	0.08967	1.4849	1.7360	**1.8103**	0	1.8103	1.8103
LinUCB(0.1)	1.2037	0.02321	1.1682	1.2577	1.2394	0	1.2394	1.2394
LinUCB(0.3)	1.1661	0.01073	1.1552	1.1926	1.1650	1.863e-08	1.1650	1.1650
LinUCB(0.5)	1.1462	0.01215	1.1136	1.1571	1.1752	1.317e-08	1.1752	1.1752
LinUCB(1.0)	1.1361*	0.01896	1.0969	1.1594	1.1594*	1.317e-08	1.1594	1.1594
TS(0.001)	**1.2203**	0.026	1.1842	1.2670	**1.2725**	0.03175	1.2301	1.3422
TS(0.01)	1.1880	0.02895	1.1585	1.2466	1.2377	0.01886	1.2132	1.2713
TS(0.1)	1.1527	0.01988	1.1289	1.1811	1.1791	0.02225	1.1437	1.2169
TS(1.0)	1.1205	0.0142	1.1009	1.1472	1.1362	0.02203	1.0971	1.1599
TS(10.0)	0.7669*	0.1072	0.5445	0.9526	0.8808*	0.01557	0.8483	0.9031
TSNR(0.01)	1.2173*	0.03369	1.1430	1.2561	1.2972*	0.02792	1.2479	1.3394
TSNR(0.1)	1.2285	0.01948	1.1915	1.2610	1.3028	0.02121	1.2701	1.3461
TSNR(1.0)	1.2801	0.02365	1.2558	1.3303	1.3250	0.03148	1.2486	1.3634
TSNR(10.0)	1.6657	0.03285	1.6025	1.7125	1.6153	0.05608	1.5210	1.7128
TSNR(100.0)	**1.7816**	0.07609	1.7093	1.9278	1.8399	0.1134	1.5240	1.9200
TSNR(1000.0)	1.7652	0.09946	1.6123	1.9346	**1.8769**	0.03731	1.8409	1.9656

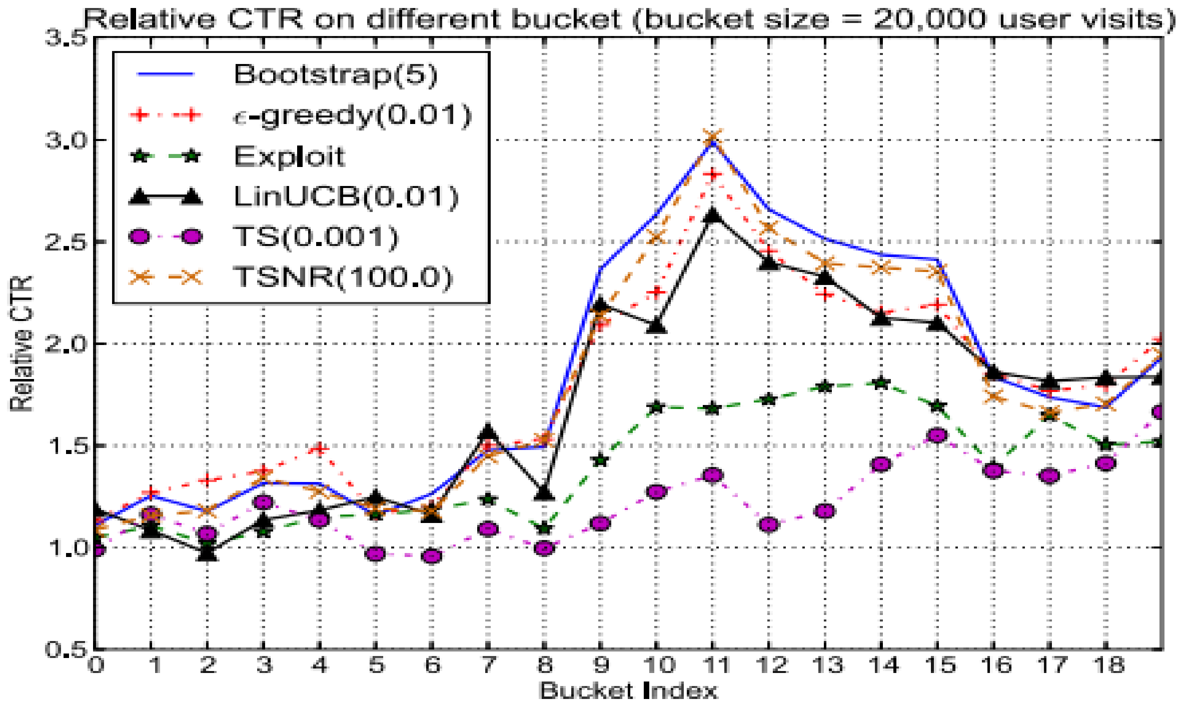
Experiment(AD KDD Cup’12)

- All numbers are relative to the random model.

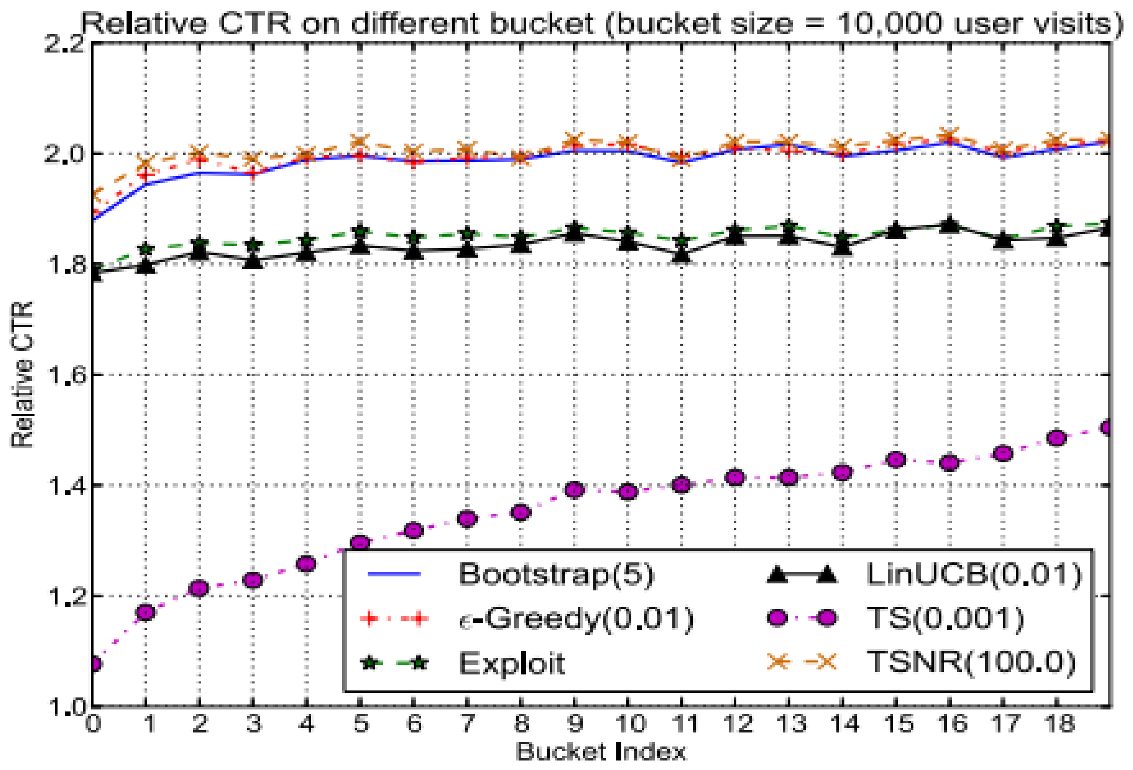


Algorithm	Cold Start				Warm Start			
	mean	std	min	max	mean	std	min	max
Bootstrap(1)	**1.9933**	0.01291	1.9692	2.0098	**1.9990**	0.005678	1.9878	2.0083
Bootstrap(5)	1.9883	0.01106	1.9686	2.0012	1.9964	0.004983	1.9848	2.0022
Bootstrap(10)	1.9862	0.009128	1.9672	1.9977	1.9890	0.005434	1.9829	2.0003
Bootstrap(30)	1.9824*	0.01492	1.9566	2.0088	1.9886*	0.006086	1.9753	1.9954
ϵ-greedy(0.01)	**1.9941**	0.007293	1.9834	2.0060	**1.9971**	0.004908	1.9886	2.0038
ϵ-greedy(0.1)	1.9089	0.004887	1.8965	1.9145	1.8952	0.002741	1.8910	1.8986
ϵ-greedy(0.3)	1.7039	0.003797	1.6990	1.7101	1.6973	0.009368	1.6834	1.7193
ϵ-greedy(0.5)	1.5018*	0.004335	1.4965	1.5114	1.4983*	0.006319	1.4845	1.5067
Exploit	**1.8185***	0.05235	1.7228	1.8934	**1.9241***	0.007046	1.9152	1.9370
LinUCB(0.01)	1.8551	0.03543	1.7977	1.9059	**1.9279**	0.006951	1.9178	1.9371
LinUCB(0.1)	**1.9168**	0.005466	1.9070	1.9267	1.9202	0.004434	1.9112	1.9266
LinUCB(0.3)	1.8665	0.003644	1.8609	1.8726	1.8610	0.003271	1.8550	1.8661
LinUCB(0.5)	1.7808	0.007009	1.7669	1.7913	1.7903	0.0051	1.7823	1.7988
LinUCB(1.0)	1.6693*	0.004738	1.6634	1.6762	1.6742*	0.003179	1.6704	1.6792
TS(0.001)	1.3587	0.009703	1.3366	1.3736	1.3518	0.01002	1.3297	1.3673
TS(0.01)	1.4597	0.007215	1.4504	1.4749	1.4891	0.006421	1.4771	1.4994
TS(0.1)	**1.5714**	0.004855	1.5647	1.5791	**1.5905**	0.004176	1.5826	1.5967
TS(1.0)	1.5345	0.003435	1.5262	1.5384	1.5421	0.003741	1.5376	1.5480
TS(10.0)	0.9388*	0.4236	0.3064	1.5675	1.3174*	0.003157	1.3115	1.3212
TSNR(0.01)	1.4856*	0.01466	1.4657	1.5078	1.5700*	0.02163	1.5499	1.6298
TSNR(0.1)	1.7931	0.01284	1.7774	1.8167	1.8716	0.01035	1.8518	1.8870
TSNR(1.0)	1.9826	0.005853	1.9704	1.9921	1.9952	0.006996	1.9833	2.0047
TSNR(10.0)	**2.0118**	0.007808	1.9941	2.0208	2.0095	0.005107	2.0022	2.0198
TSNR(100.0)	2.0039	0.008942	1.9912	2.0215	**2.0097**	0.004586	2.0022	2.0187
TSNR(1000.0)	2.0047	0.01022	1.9894	2.0228	2.0088	0.004644	1.9966	2.0151

CTR over Time Bucket (Yahoo! News Data)

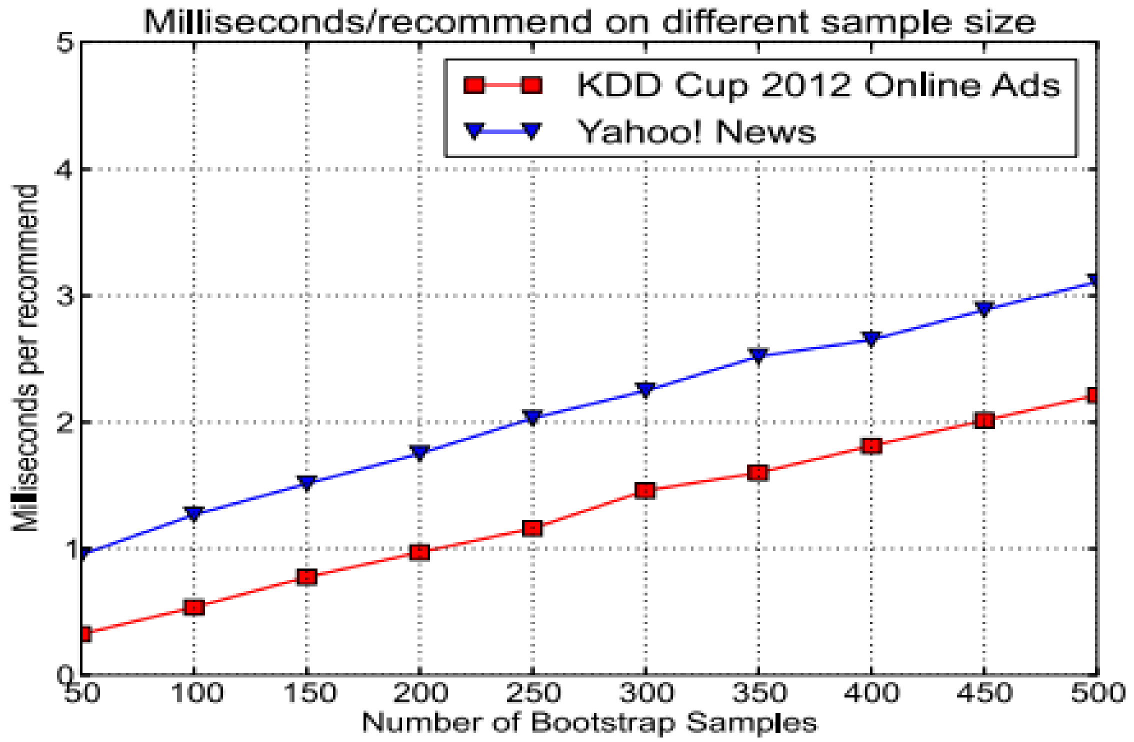


CTR over Time Buckets (KDD Cup Ads Data)



Efficiency

- Time cost on different bootstrap sample sizes



Summary of Experiment

- For solving the contextual bandit problem, the algorithms of ϵ-greedy and LinUCB can achieve the optimal performance, but the input parameters that control the exploration need to be tuned carefully.
- The probability matching/Thomson Sampling strategies highly depend on the selection of the prior.
- Our proposed algorithm is a safe choice of building predictive models for contextual bandit problems under the scenario of cold-start.

Conclusion

- Propose a non-Bayesian Thompson Sampling method to solve the personalized recommendation problem.
- Give both theoretical and empirical analysis to show that the performance of Thompson sampling depends on the choice of the prior.
- Conduct extensive experiments on real data sets to demonstrate the efficacy of the proposed method and other contextual bandit algorithms.

Future Work

- MAB with similarity information
- MAB in a changing environment
- Explore-exploit tradeoff in mechanism design
- Explore-exploit learning with limited resources
- Risk vs. reward tradeoff in MAB

[1] http://research.microsoft.com/en-us/projects/bandits/

Acknowledgements

- **Funding**
  - NSF
  - Department of Homeland Security
  - Army Research Office
  - Faculty Research Awards from Industry

- **Ph.D. Students working on recommendation systems**
  - Dr. Lei Li (Samsung Research)
  - Dr. Yexi Jiang (Facebook)
  - Dr. Liang Tang (LinkedIn)
  - Dr. Chao Shen (Amazon)
  - Dr. Dingding Wang (Florida Atlantic University)
  - Qing Wang (FIU)
  - Chunqiu Zeng (FIU)

Email: taoli@cs.fiu.edu
http://www.cs.fiu.edu/~taoli

谢 谢 ！

Bandit Algorithm: simple greedy

- Regret is defined in terms of average reward
- So if we can estimate avg. reward, we can minimize regret
- Consider a greedy algorithm that takes the arm with the highest avg. reward
  - Example:
    - Arm1 has reward 1 with prob. 0.2
    - Arm 2 has reward 1 with prob. 0.8
  - Play A1, get reward 1
  - Play A2, get reward 0
  - Now avg. reward of A1 will never drop to 0,and we will never play A2