An Introduction to Bandit Algorithm

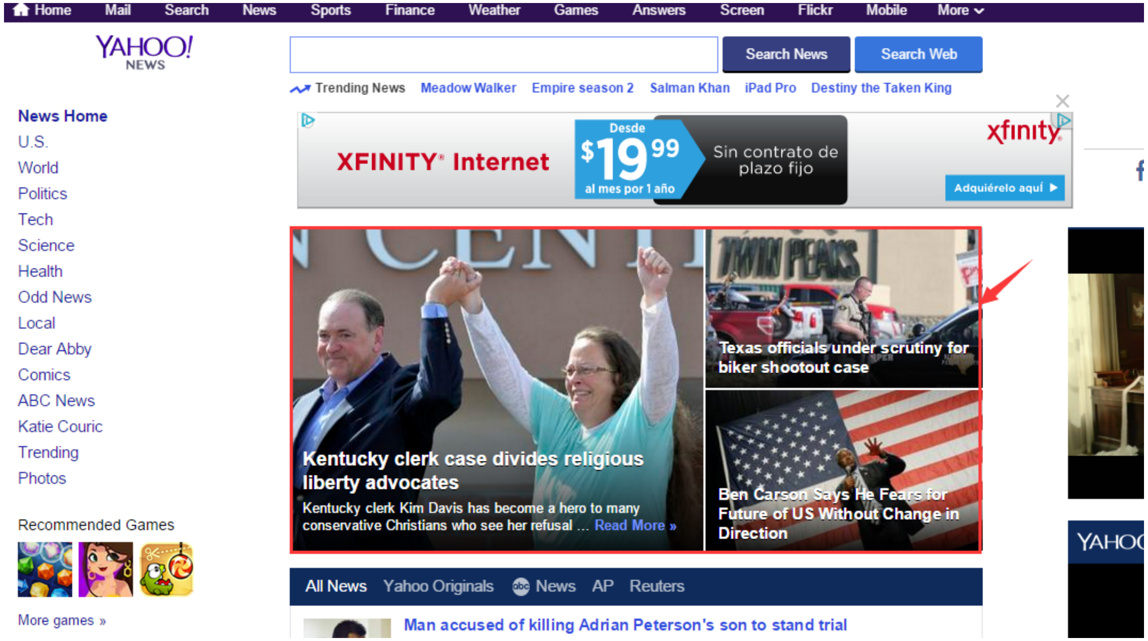
Qing Wang, Ph. D. student, 2016
Some slides from Li Zhou and Jure Leskovec

AGENDA

- Motivation
- Background
- Contextual-free MAB
- Contextual MAB
- Results and Future work
- Question

What is news personalization?

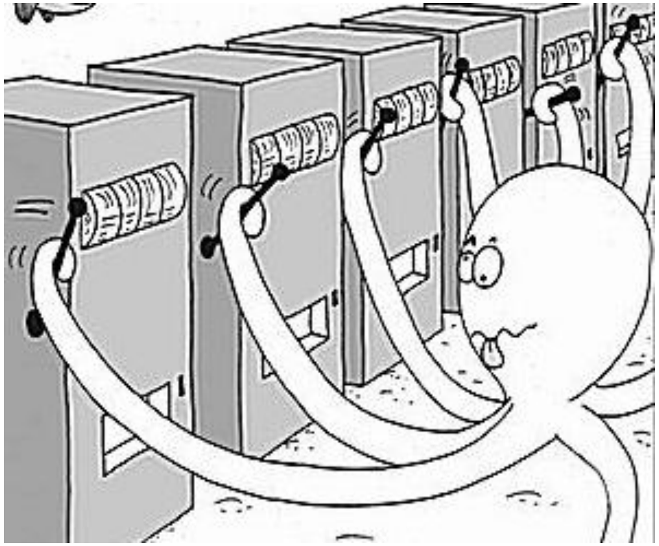
- **Customize** news feed based on users' interests.
- Particularly, **Cold Start** problem: How to personalize news for a new user?
- **Goal**: Maximize user engagement



Multi-armed Bandit Algorithm

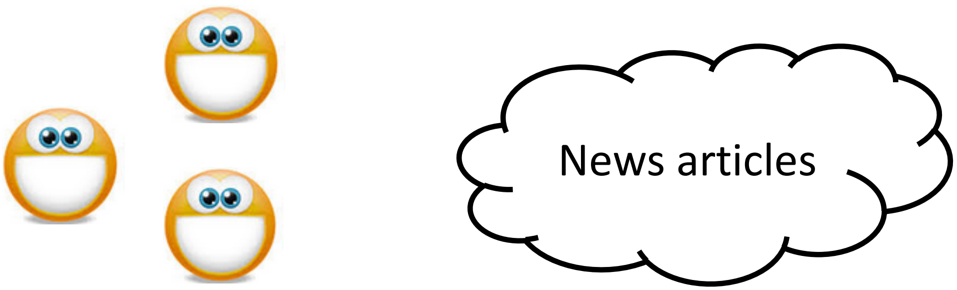
- A gambler →casino
- A row of slot machines providing a random rewards

Objective: Maximize the sum of rewards(Money)!



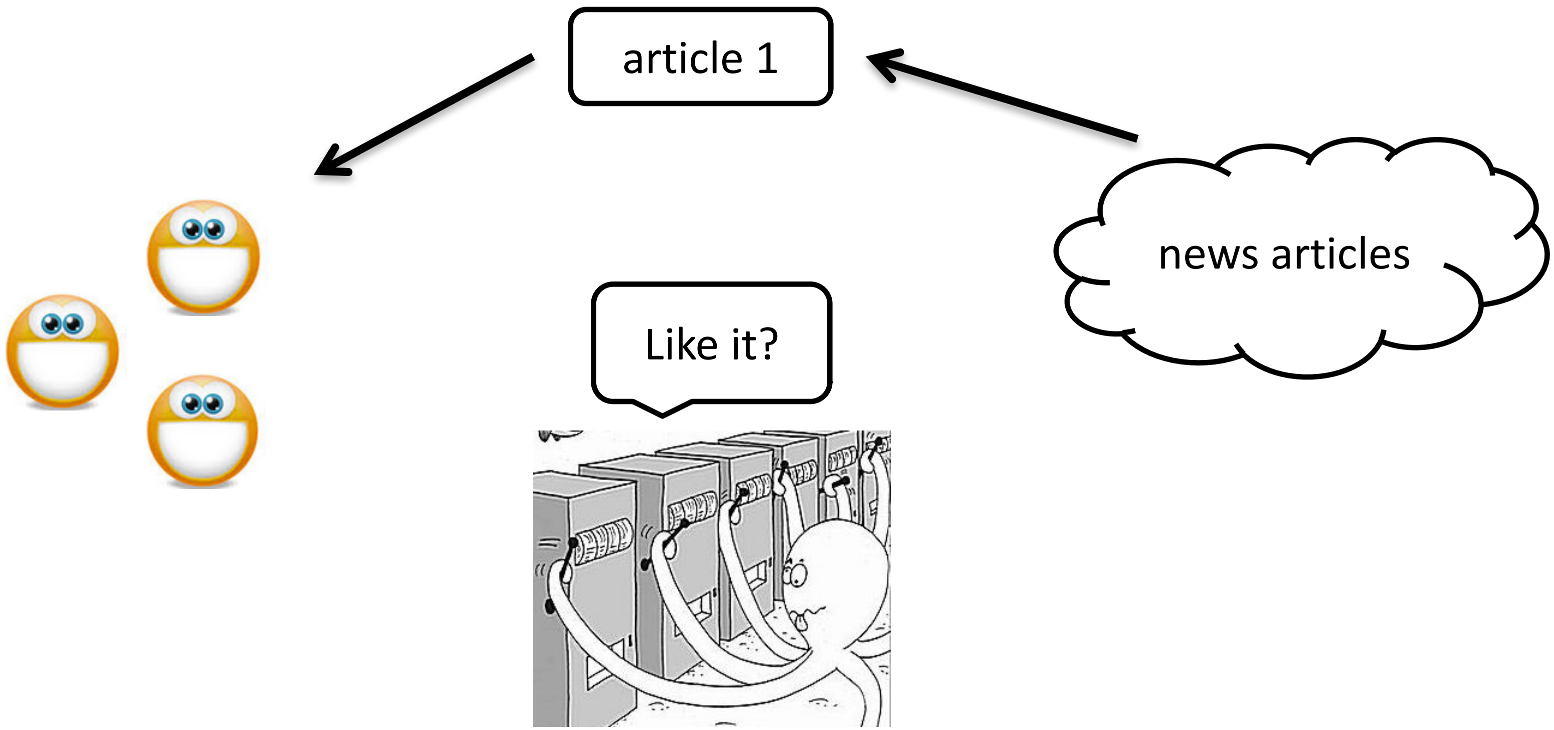
Multi-armed Bandit

- Take news personalization as an example
  - There are a bunch of articles in the news pool
  - Users come sequentially and ready to be enter



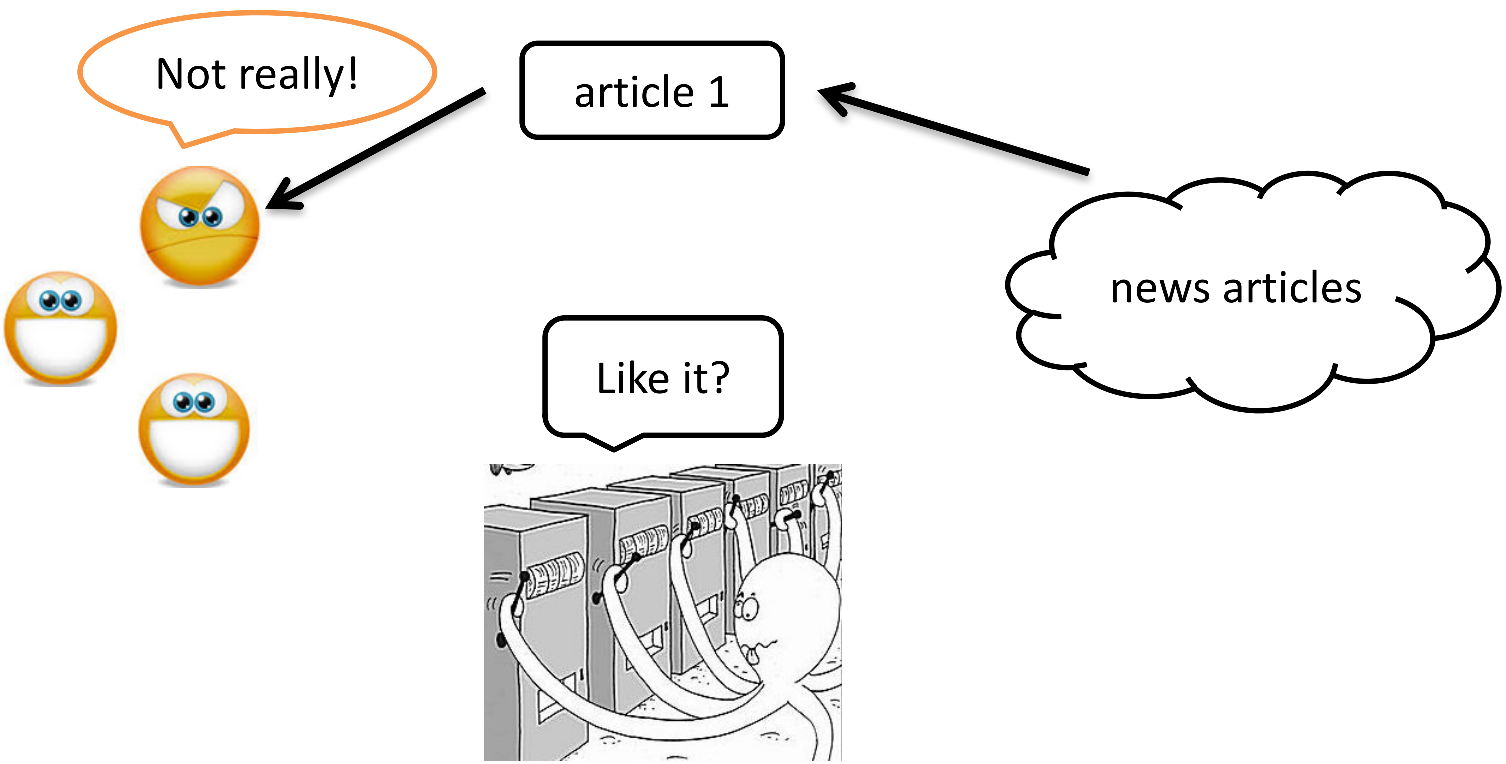
Multi-armed Bandit

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



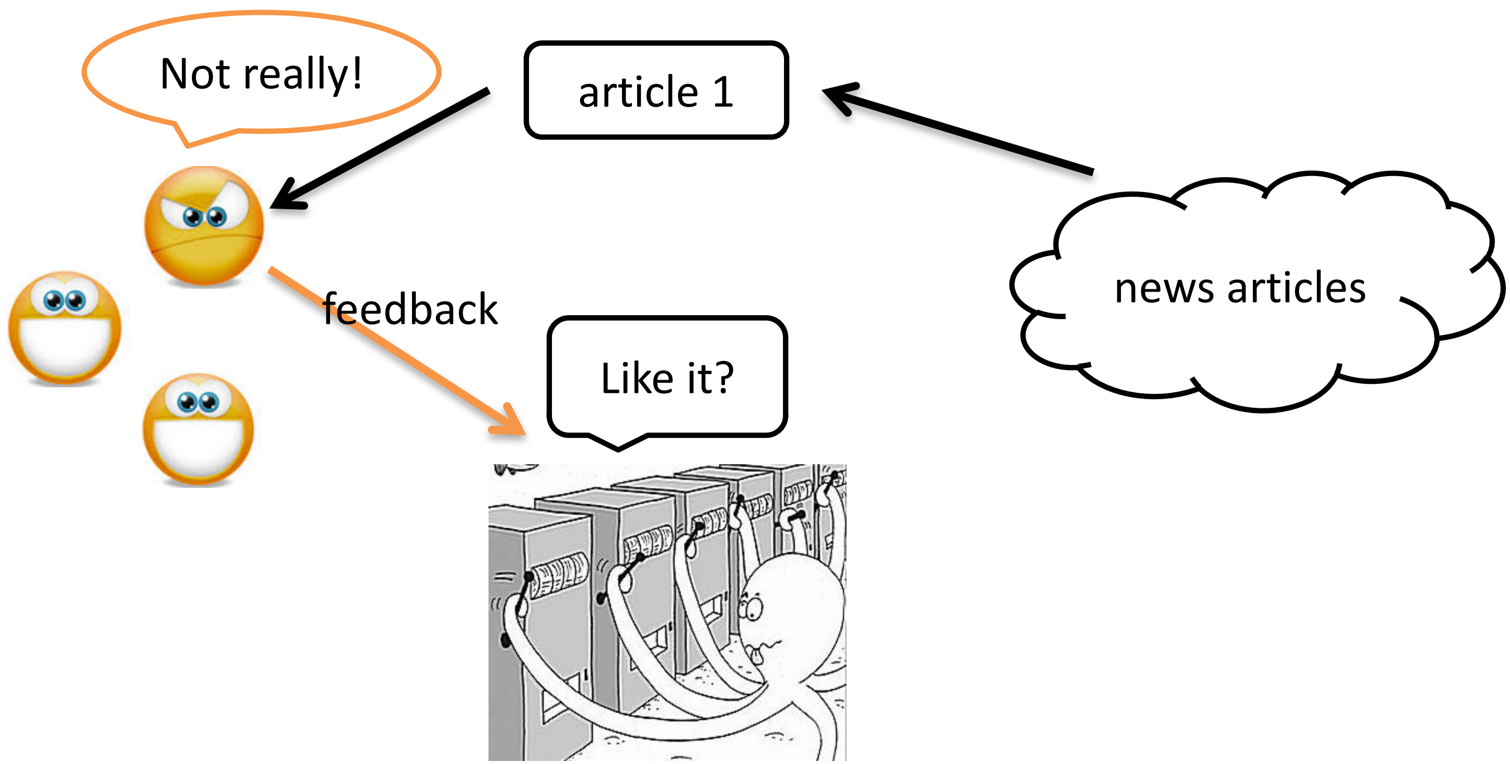
Multi-armed Bandit

- Goal: maximize CRT(click through rate)



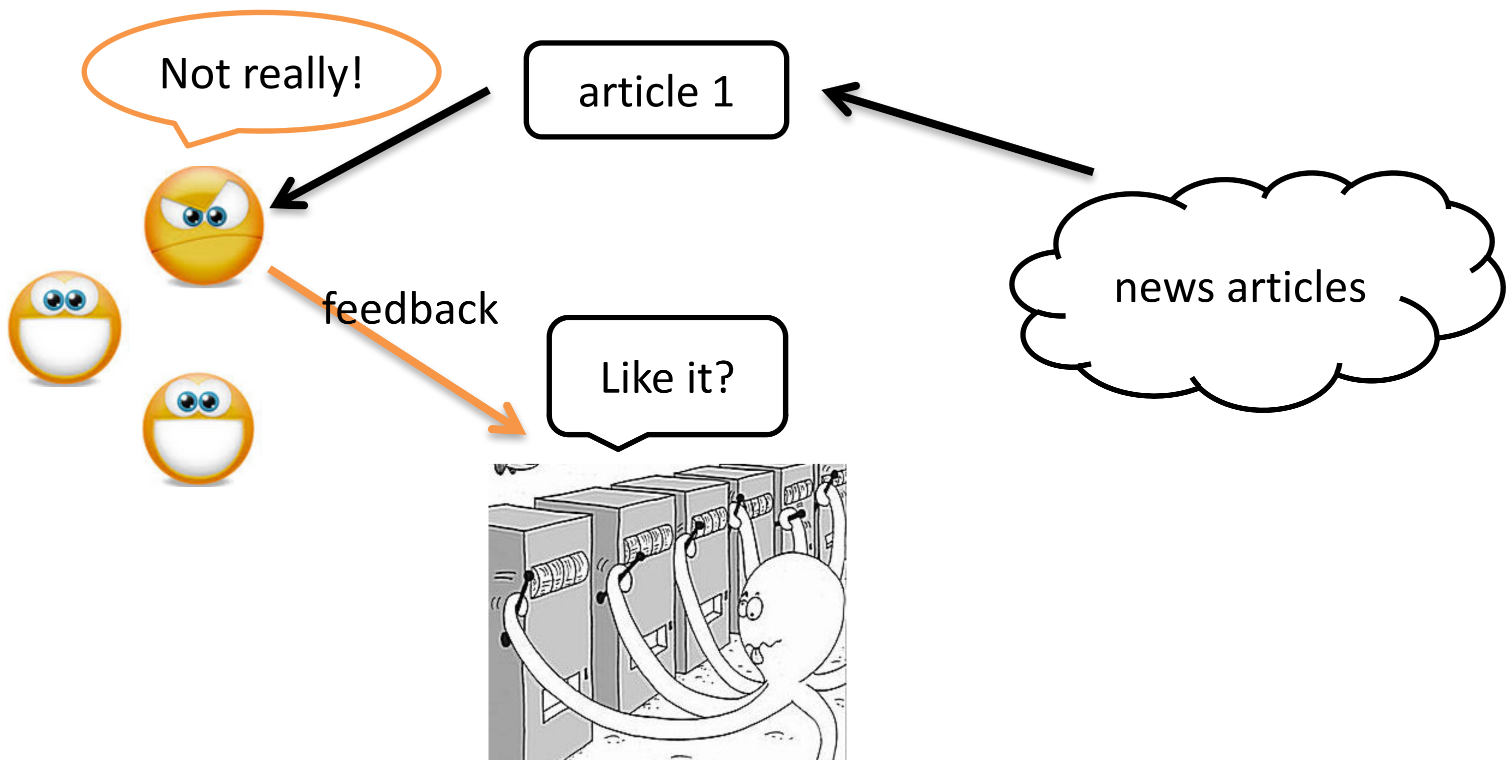
Multi-armed Bandit

- Update the model with user's feedback



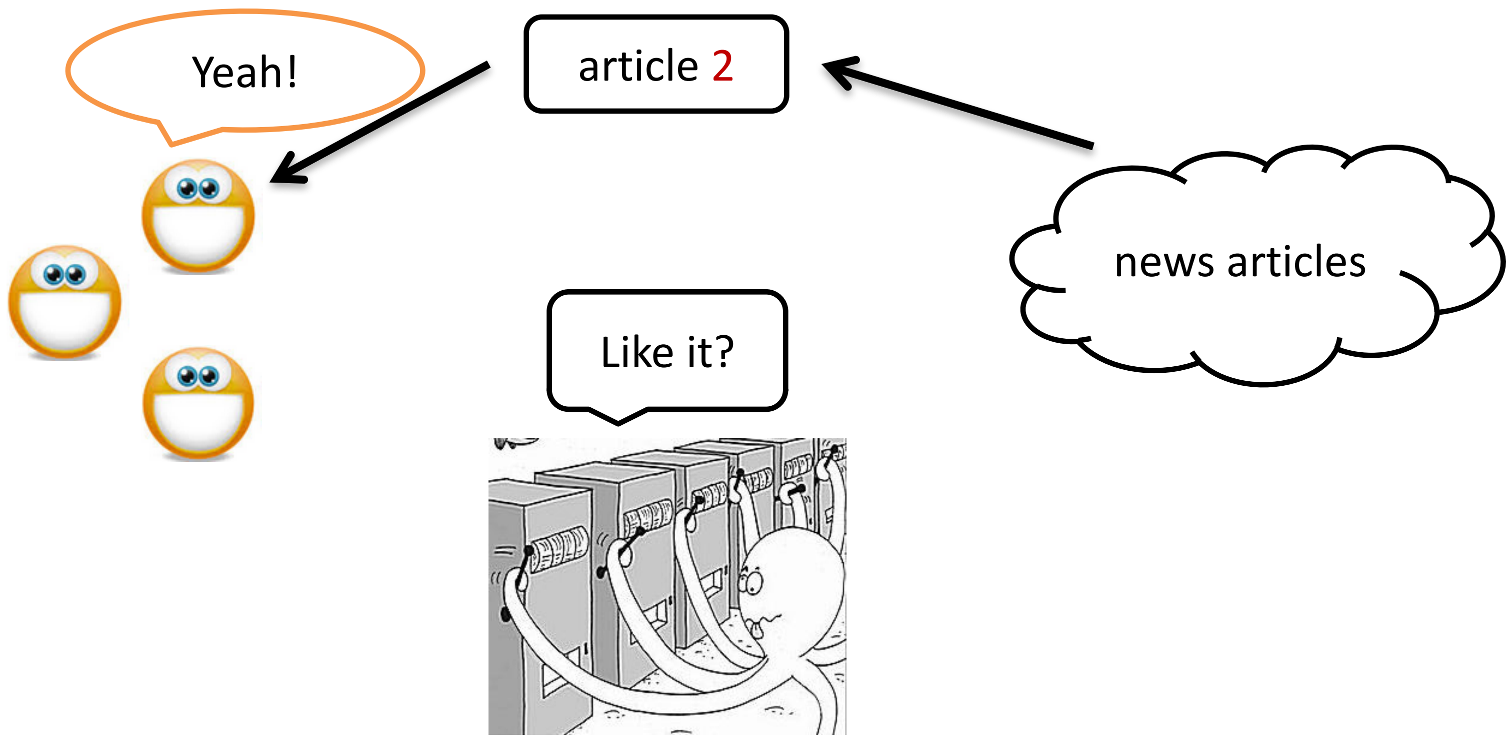
Multi-armed Bandit

- Update the model with user's feedback



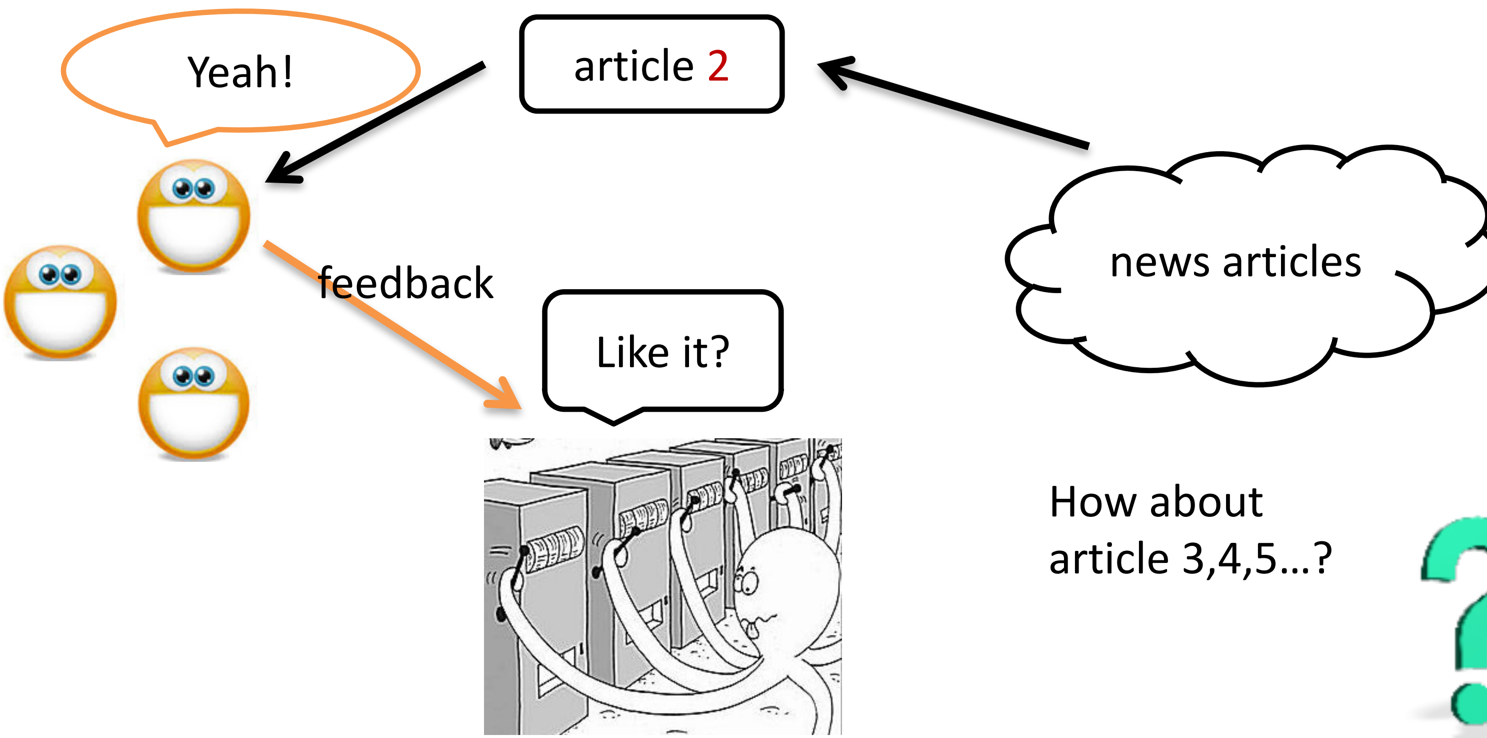
Multi-armed Bandit

- Update the model once the user gives the feedback



Multi-armed Bandit

- Update the model once the user gives the feedback



How about
article 3,4,5…?

Background

Multi-armed Bandit 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/

Other Application

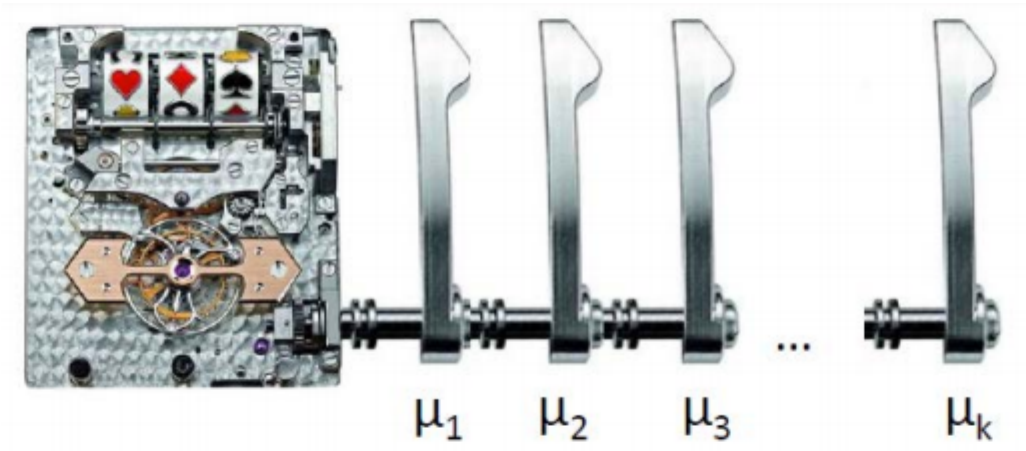
- Clinical trials:
  - Investigate effects of different treatments while minimizing patient losses
- Adaptive routing:
  - Minimize delay in the network by investigating different routes
- Asset pricing:
  - Figure out product prices while trying to make optimal profit

Some Jargon Terms

- 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

[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. $(1 - \mu_a)$
- All draws are independent given $\mu_1 \ldots \mu_k$
- How to pull arms to maximize total reward?(estimate the arm's prob. of winning μ_a)

Model of Stochastic 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 $\boldsymbol{P_j}$(reward is independent of previous draws)
- 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**: $\mu^* = \boldsymbol{max}\{\,\mu_a |\ \boldsymbol{a} = \mathbf{1}, \dots, \boldsymbol{k}\}$
- Let $i_1, \dots i_T$ be the sequence of arms pulled
- Instantaneous regret at time t: $r_t = \mu^* - \mu_{a_t}$
- Total regret:
  - $\boldsymbol{R_T} = \sum_{t=1}^{T} \boldsymbol{r_t}$
- Typical goal: arm allocation strategy that guarantees :

$$\frac{\boldsymbol{R_T}}{T} \rightarrow 0 \text{ as } \mathrm{T} \rightarrow \infty$$

Allocation Strategies

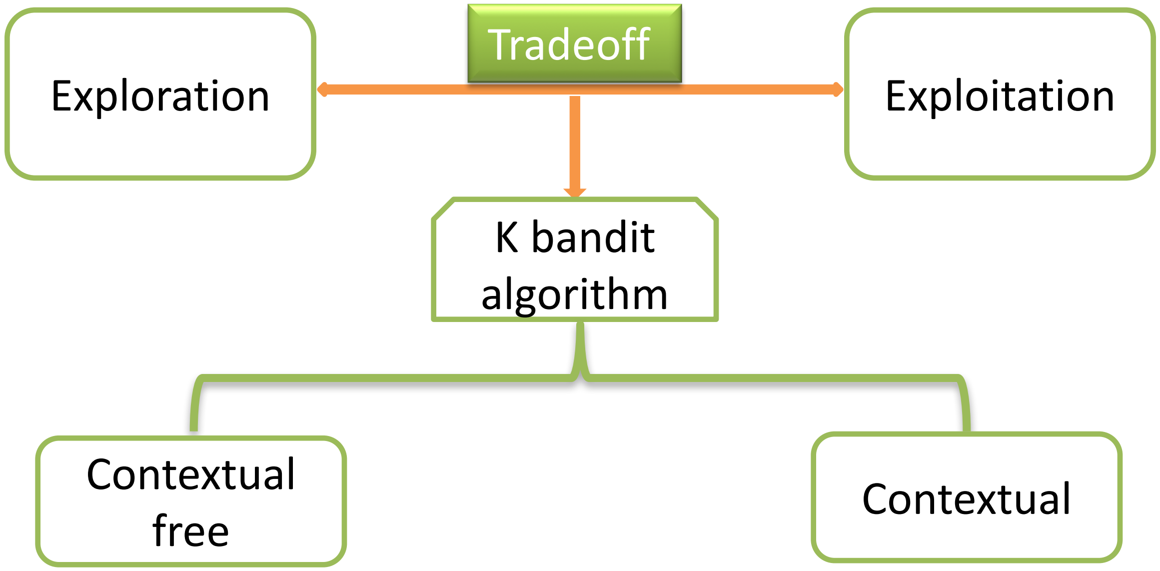
- If we knew the payoffs, which arm should we pull?
  - **best arm**: $\mu^* = \boldsymbol{max}\{\, \mu_a | \, \boldsymbol{a} = \mathbf{1}, \dots, \boldsymbol{k}\}$
- What if we only care about estimating payoff μ_a?
  - Pick each of ***k*** arms equally often : $\frac{\boldsymbol{T}}{k}$
  - **Estimate**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

Exploration vs. Exploitation

- **Trade off** between exploration (gathering data about arm payoffs) and exploitation (making decisions based on history data) in decision making.

Algorithm to Exploration & Exploitation



1. Epsilon algorithm
2. UCB1

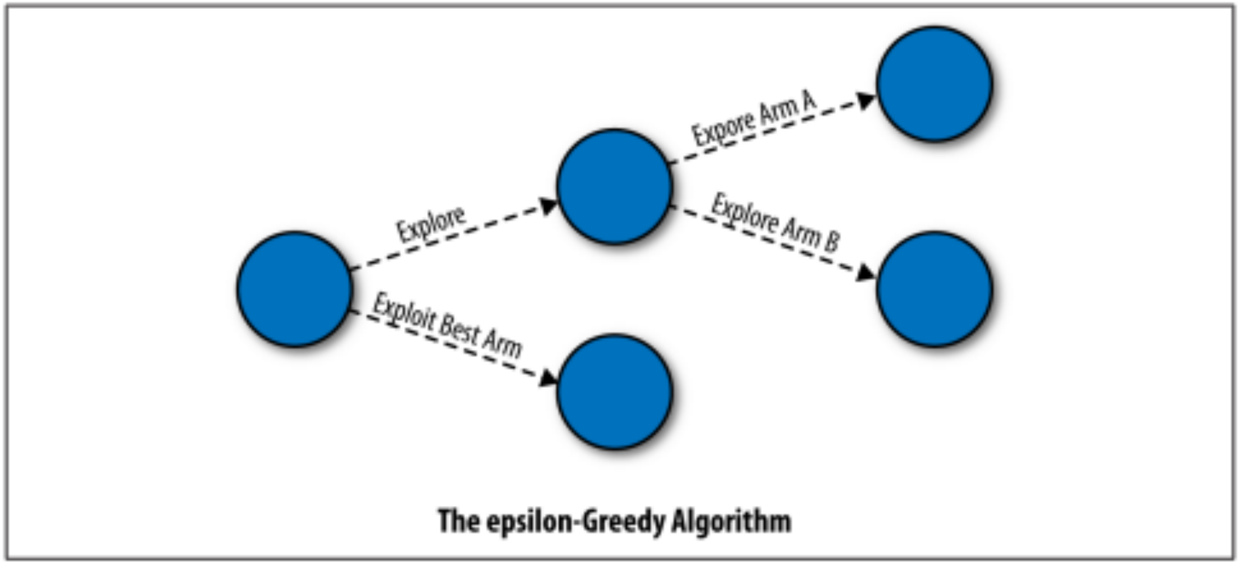
1. Ex3, Ex4
2. Tompson Sampling
3. LinUCB

Existing Work

- LinUCB (Li, Lihong et al. 2010), a contextual-bandit approach to news personalization
- Thompson sampling (Chapelle, Olivier et al. 2011), An Empirical Evaluation of Thompson Sampling
- tUCB (Lazaric, Alessandro et al. 2013) Sequential transfer in multi-armed bandit with finite set of models

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.



The epsilon-Greedy Algorithm

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!

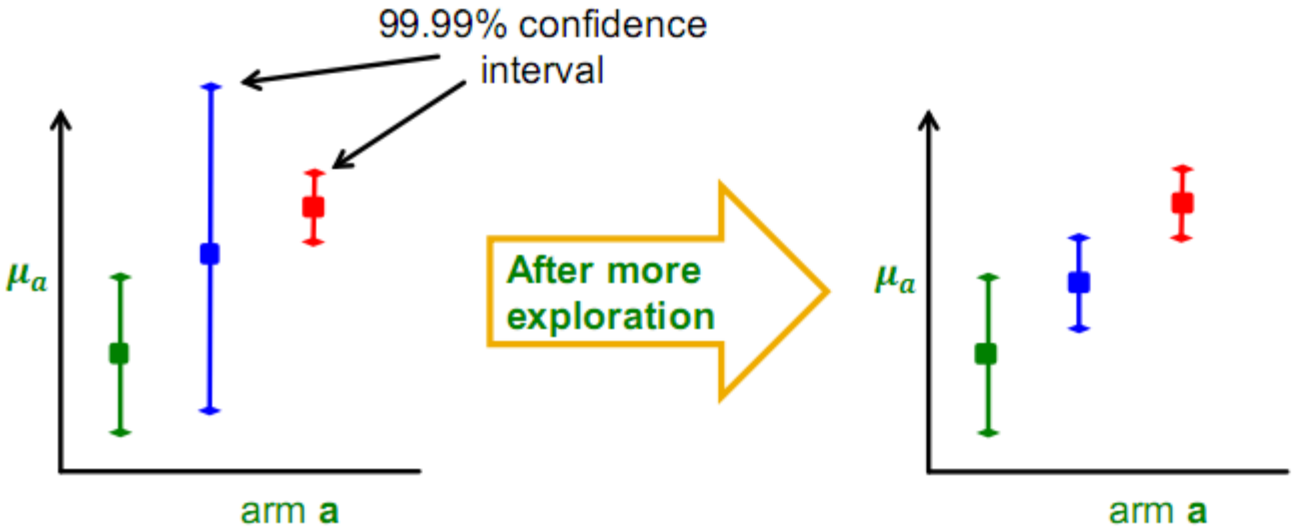
Confidence Intervals

- **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 Intervals

- 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 Based Selection



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 $\left|\mu_a - \widehat{\mu_{a,m}}\right| \leq b$ (want b to be as small as possible)
  - Goal : Want to bound $\mathbf{P}\left(\left|\mu_a - \widehat{\mu_{a,m}}\right| \leq b\right)$

Hoeffding's Inequality

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

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 i
    - m_a is the number of pulls of arm i so far.
  - For t = 1 : T
    - 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: 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

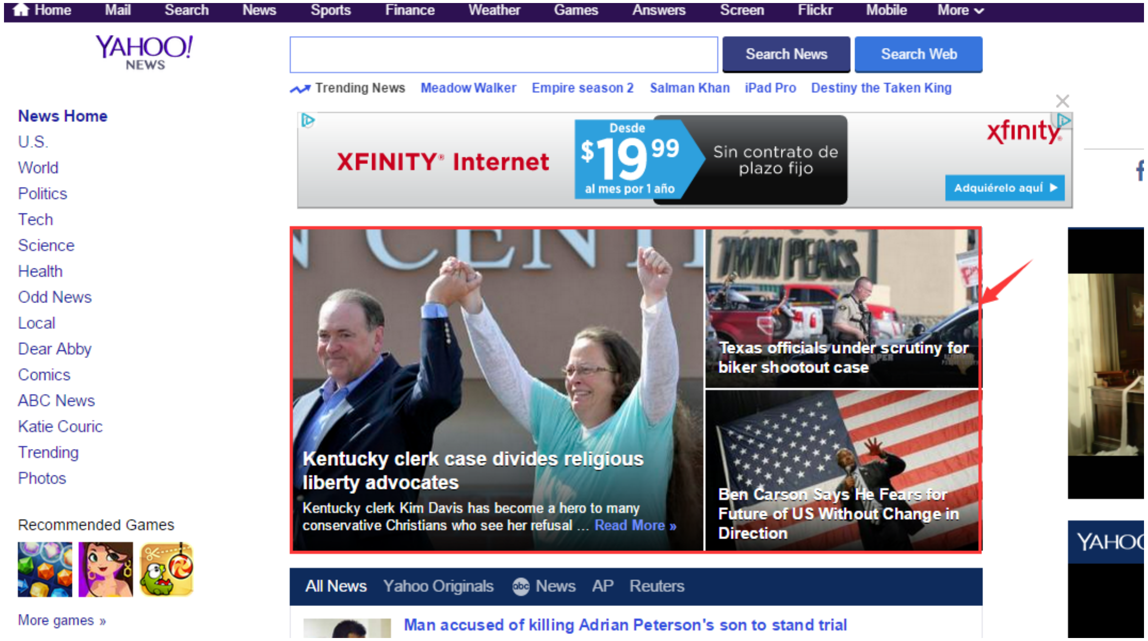
- Theorem [Auer et al. 2002]
  - Suppose optimal mean payoff is $\mu^* = \max_a \mu_a$
  - And for each arm let $\Delta_a = \mu^* - \mu_a$
  - Then it holds that

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

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

What is news personalization?

- **Customize** news feed based on users' interests.
- Particularly, **Cold Start** problem: How to personalize news for a new user?
- **Goal**: Maximize user engagement



Modeling News Personalization as Contextual Multi-armed Bandit Problem

- Select news articles based on current context such as users' profile and articles content
- **Pros**: Trade-off between acquiring new information (**exploration**) and capitalizing on the information available so far (**exploitation**). Able to handle **cold start** problem.

LinUCB (Li, Lihong 2010) for News Personalization

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

$$\mathbf{E}[r_{t,a}|\mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^{\top}\boldsymbol{\theta}_a^*$$

- 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.

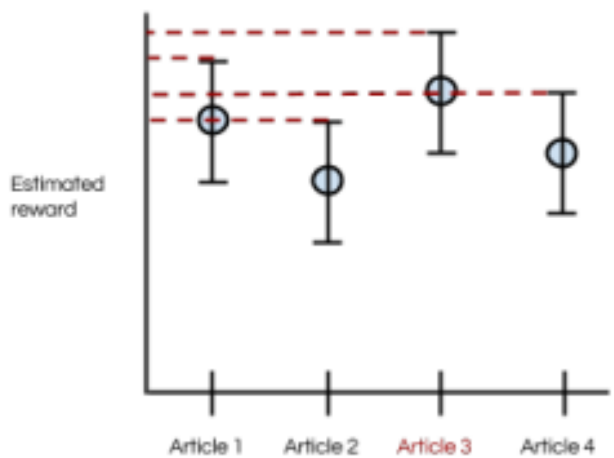
$$R_{\mathsf{A}}(T) \stackrel{\text{def}}{=} \mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t^*}\right] - \mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t}\right]$$

LinUCB (Li, Lihong 2010) for News Personalization

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

$$a_t \stackrel{\text{def}}{=} \arg\max_{a \in \mathcal{A}_t} \left(\underbrace{\mathbf{x}_{t,a}^\top \hat{\boldsymbol{\theta}}_a}_{\text{estimated reward}} + \underbrace{\alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}}_{\text{half-size confidence interval}} \right)$$

- Select arm based on the upper bound of confidence interval



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 com and go)
  - For example, in news article recommendation, for instance, editors add/remove articles to/from a pool

Different between UCB1 and LinUCB

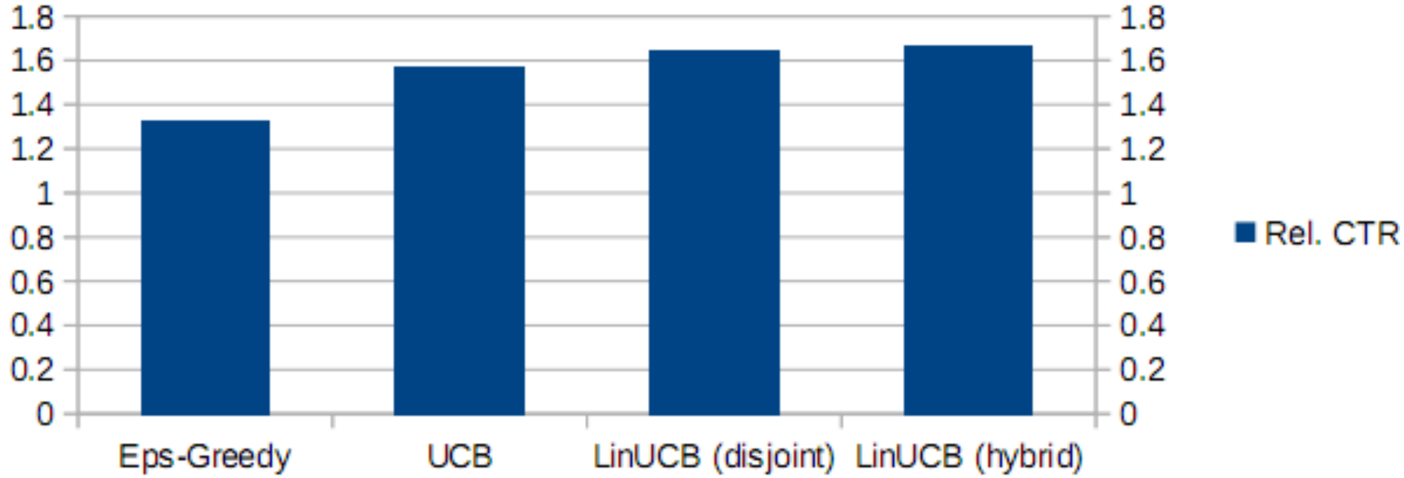
- UCB1 directly estimates μ_a through experimentation (without any knowledge about arm *a*)
- LinUCB estimates μ_a by regression $\boldsymbol{\mu_a = x_{t,a}^T \cdot \theta_a^*}$
  - The hope is that we will be able to learn faster as we consider the context $x_a(\mathrm{user}, \mathrm{ad})$ of arm *a*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

Empirical Results

- Scenario
  - 4.7m events(featured article, infos, click) in tuning set
  - 36m events in test set
  - Articles and Users clustered into 5 categories

Empirical Results

- Results



Questions?