A Gang of Bandits

Will Knospe, Paul Reich,
Bryce Bern, Dawson d'Almeida

The Problem

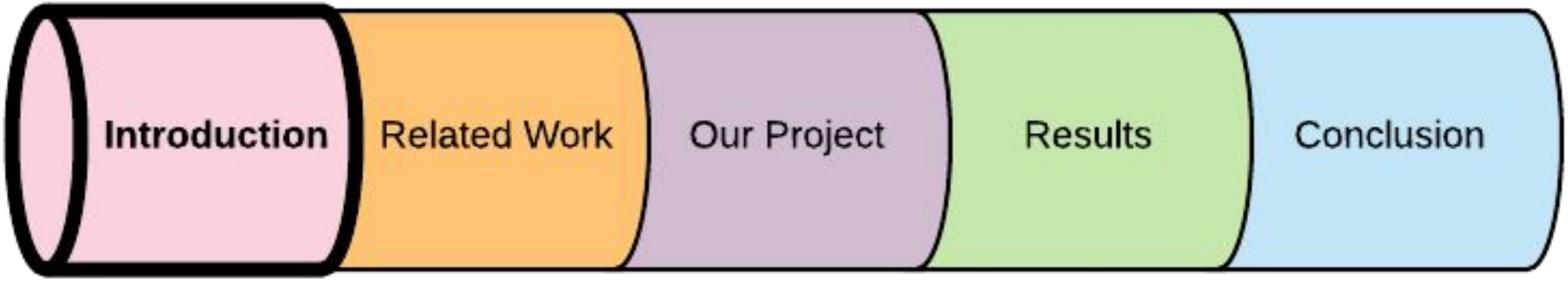
Trying to make a recommendation from thousands of choices

Only understand users' preferences as we recommend them shows

MyHouse Friends

Tags that identify what shows have in common

Road Map



Introduction
Related Work
Our Project
Results
Conclusion

Introduction to our Project

Replicating a paper that tries to solve this problem: A Gang of Bandits

Why replicate papers?

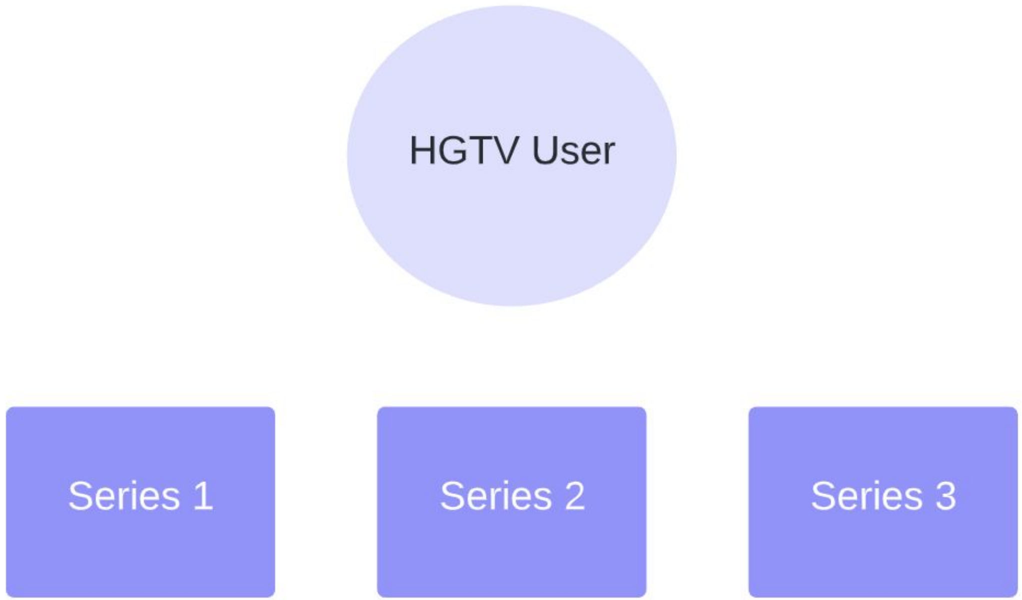
- Ensure papers’ processes are repeatable
- Validate findings as basis for new research in the future
- Avoid replication crises faced by other fields

Basic Multi-Armed Bandit Problem

The user might enjoy an episode from a series based on some set probability

Choose a series and observe whether or not the user enjoyed the episode

Update the probabilities associated with that series

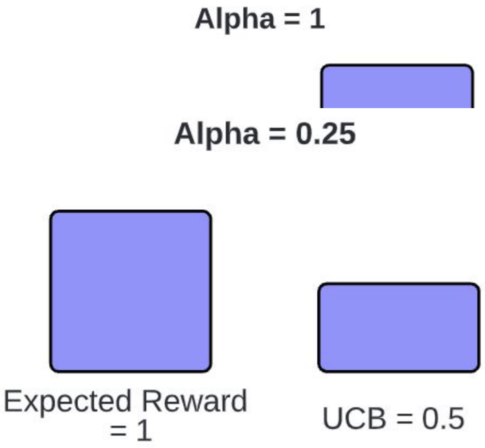


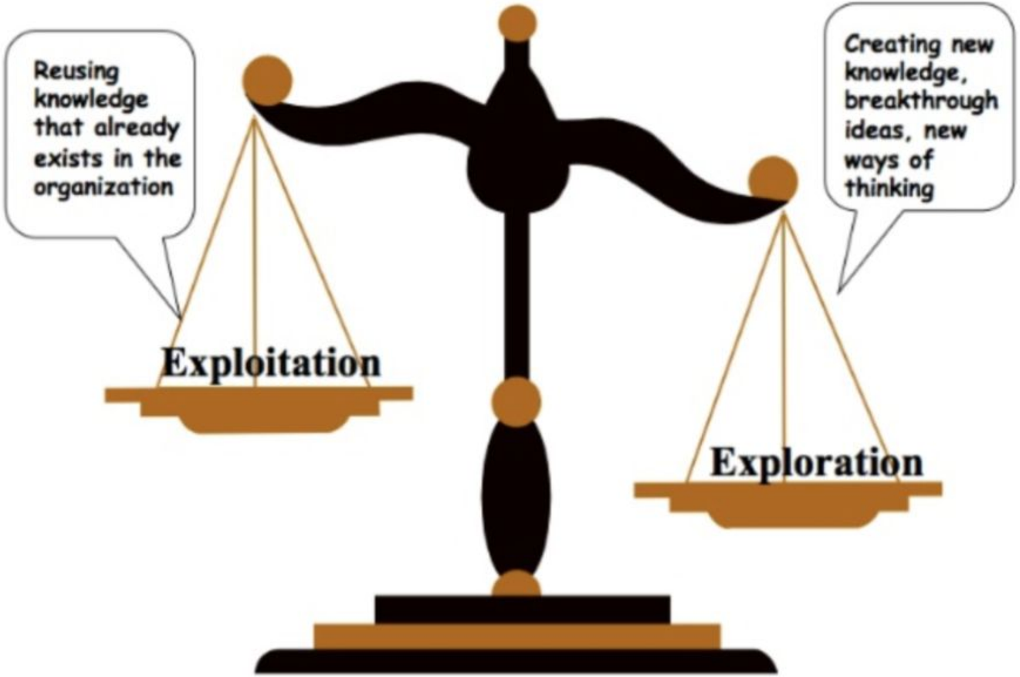
Multi-Armed Bandit - Exploration Vs. Exploitation

How does the algorithm balance the need to exploit and explore?

Score = expected reward + UCB

α: exploration factor





Terminology

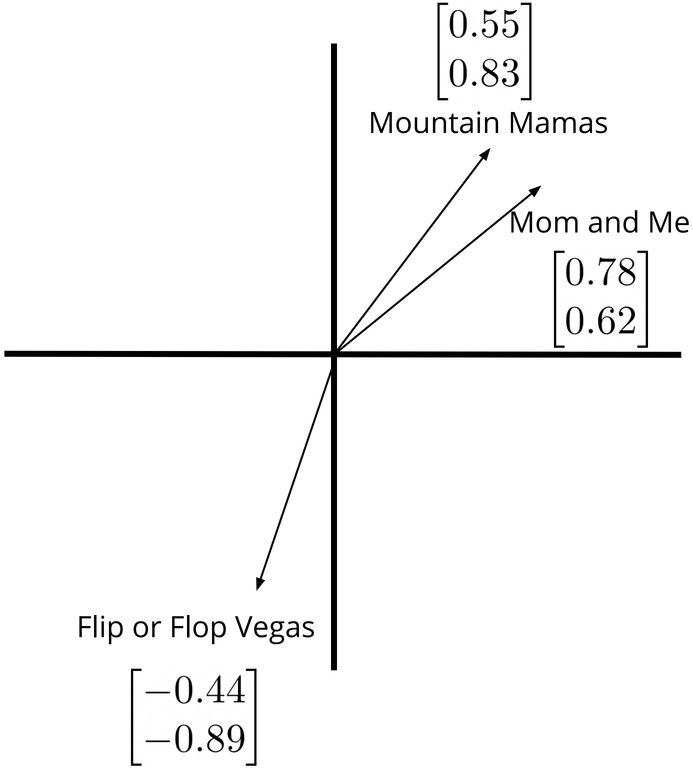
Learner: An instance of a MAB algorithm that is making recommendation decisions

Context: Represents a recommendation (i.e. song, website, etc...) that a learner can choose

- Represented as a vector - this 'summarizes' the context information

User: Who the learner is recommending to

Reward: Measure of how good a recommendation decision is

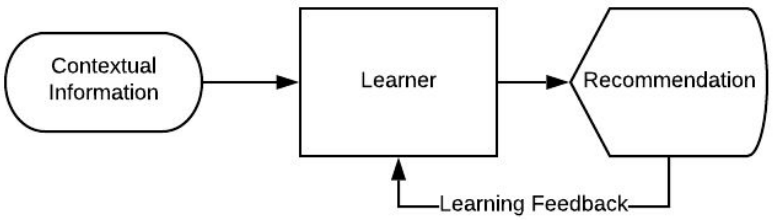


Formalization of the problem

There are **T** time steps and **K** possible contexts at each time step **t**

At each **t:**

- The learner <u>chooses</u> one of the possible contexts
- The learner receives a reward **r**
- The learner <u>updates</u> its knowledge
  - What contexts it has chosen and what the subsequent rewards were



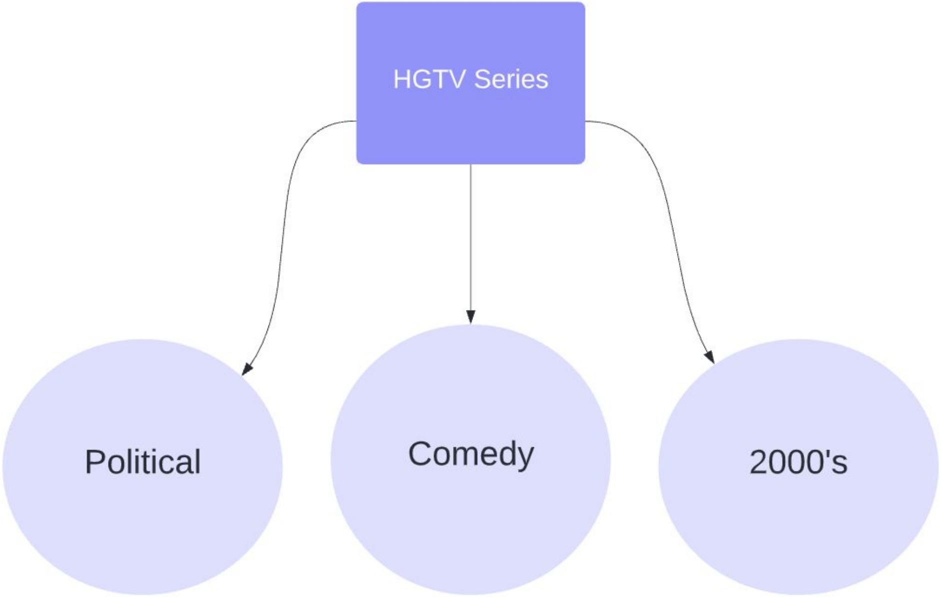
Road Map


Introduction
Related Work
Our Project
Results
Conclusion

Related Work - Contextual Bandits[1]

We are once again recommending a series to a user

- But each series is comprised of a list of tags: a political, comedy released in the 2000’s
- If the user enjoyed the series, update the user so that similarly tagged series will have higher scores in the future



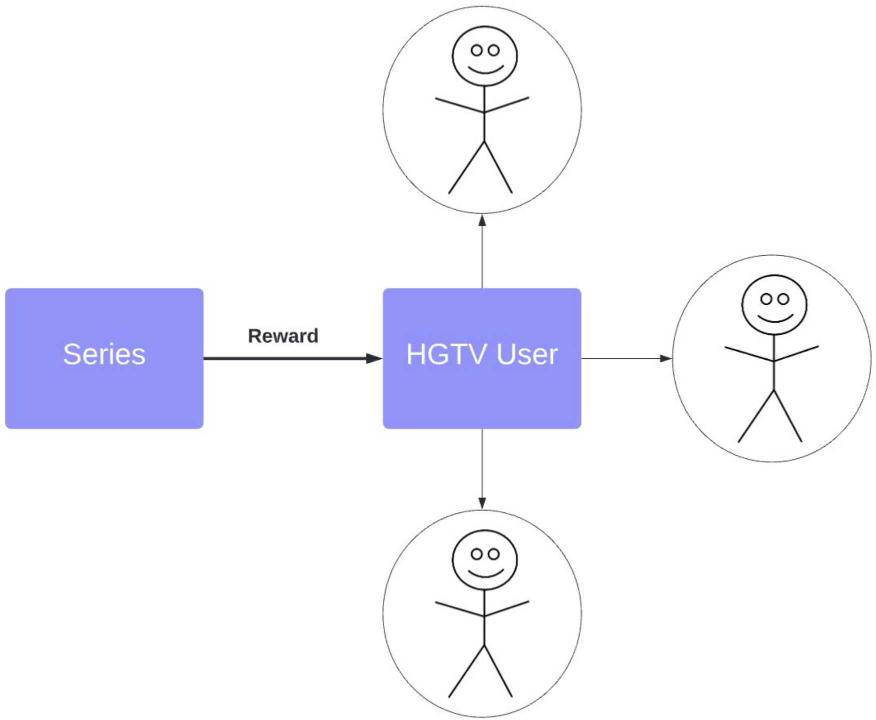
[1] Chu, Wei, et al.. "Contextual bandits with linear payoff functions." 2011.

Related Work - Network Based Bandits[1]

There is a network in which the HGTV user has three friends

Choose a series for the HGTV user and observe the reward

Update not only the HGTV user, but also the connected friends

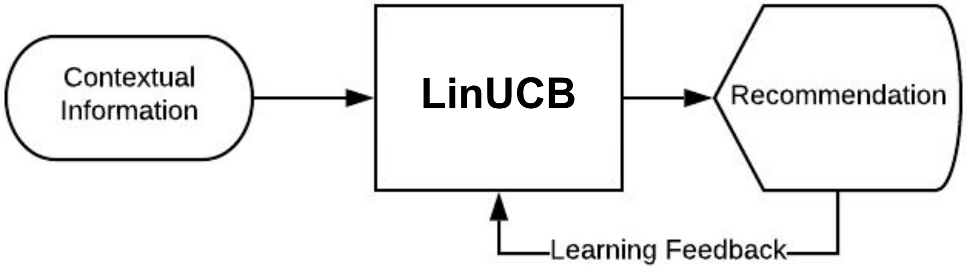


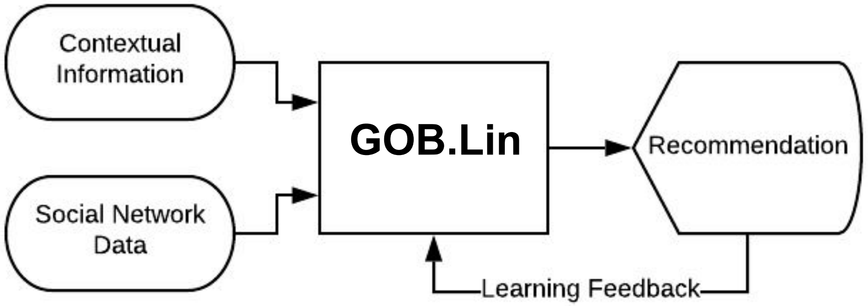
[1] Swapna Buccapatnam, Atilla Eryilmaz, and Ness B. Shroff. "Multi-armed Bandits in the Presence of Side Observations in Social Networks", 2013.

Road Map


Introduction
Related Work
Our Project
Results
Conclusion

Overview of A Gang of Bandits





LinUCB[2]

Contextual MAB (MAB problem with expert advice)

Primary point of comparison for GOB.Lin

Maintains a <u>bias vector</u> **b** and a <u>context matrix</u> **M**

- **b**: remembers how well the learner has done with certain contexts

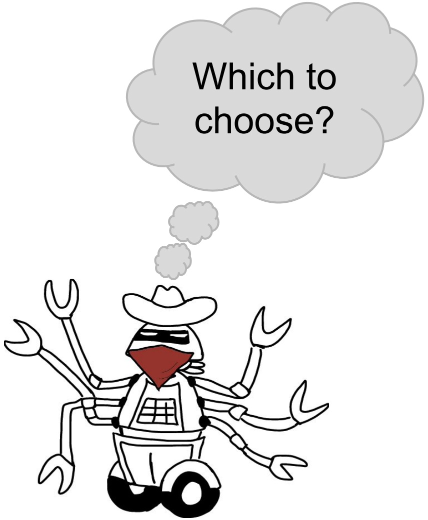
$$\begin{bmatrix} a \\ . \\ . \\ . \\ z \end{bmatrix}$$

- **M**: remembers how many times the learner has chosen certain contexts

$$\begin{bmatrix} a_1 & ... & a_d \\ . & . & . \\ . & . & . \\ . & . & . \\ z_1 & ... & z_d \end{bmatrix}$$

[2] Chu, Li, Reyzin, Schapire

Choosing an Action

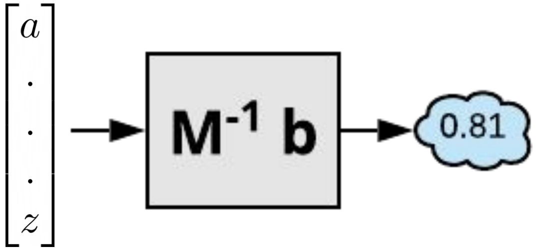


Learner observes **K** context vectors ($\mathbf{x}_\mathbf{k}$)

$$\begin{bmatrix} a_1 \\ \cdot \\ \cdot \\ \cdot \\ z_1 \end{bmatrix} \begin{bmatrix} a_2 \\ \cdot \\ \cdot \\ \cdot \\ z_2 \end{bmatrix} \cdots \begin{bmatrix} a_K \\ \cdot \\ \cdot \\ \cdot \\ z_K \end{bmatrix}$$

Learner constructs a vector $\mathbf{w} = \mathbf{M}^{-1}\,\mathbf{b}$

- Approximates the theoretical linear function from context vectors to context payoffs



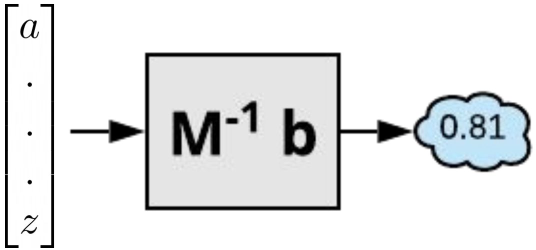
Calculating score

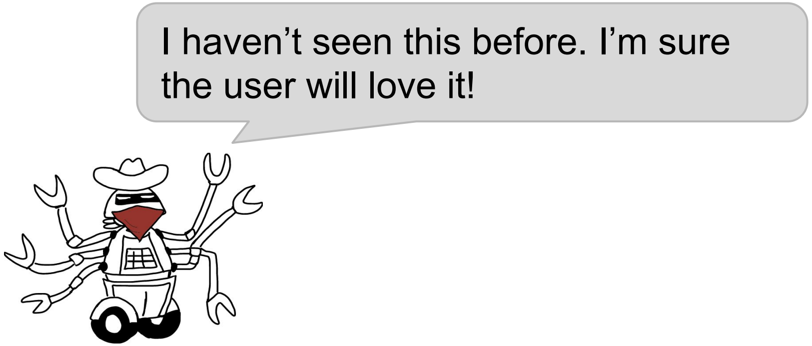
For each context vector, it calculates a **score**:

Expected payoff **P** + Confidence bound **CB**

$$w^T x_k$$

$$\alpha\sqrt{x_k^T M^{-1} x_k \, \log(t+1)}$$





Updating Knowledge

From chosen context $\mathbf{x_t}$ receive a payoff $\mathbf{a_t}$

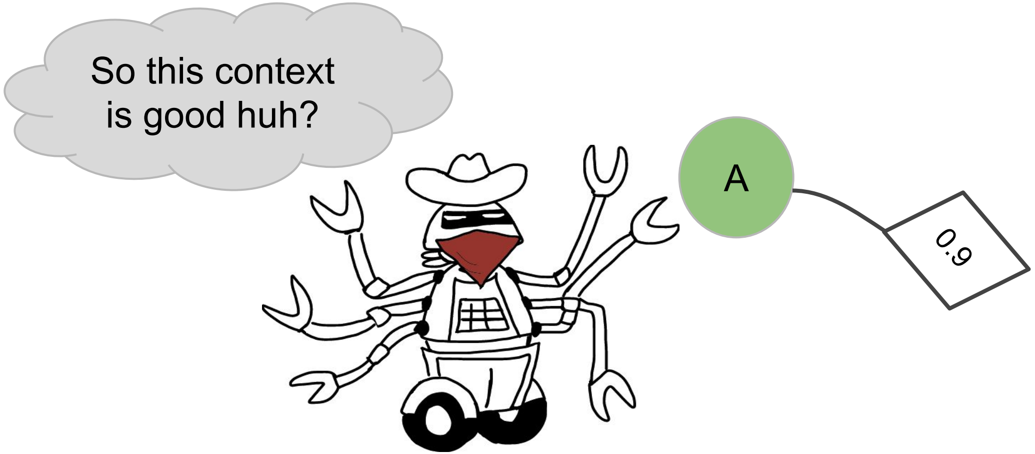
M: Adjust by outer product of context vector

$$M_t = M_{t-1} + x_t x_t^T$$

b: Adjust by context vector scaled by payoff

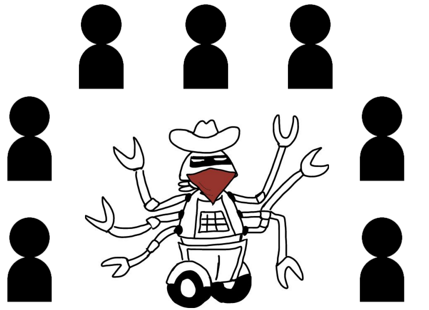
$$b_t = b_{t-1} + a_t x_t$$

This updating leads to more accurate scores in future choosing rounds!



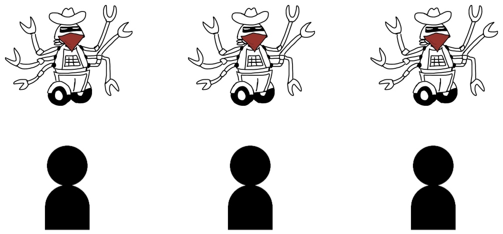
Implementations

LinUCB-SIN



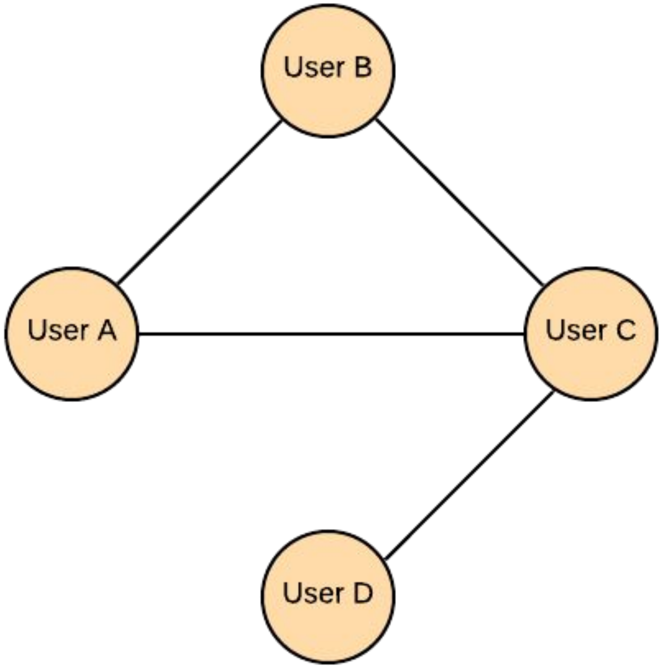
- The learner maintains only one context matrix and bias vector for all users
- Advantage: It learns quickly and accurately if users are similar

LinUCB-IND



- The learner maintains a separate context matrix and bias vector for each user
- Advantage: It learns accurately if users are different

GOB.Lin



$$+ \quad \begin{bmatrix} a_1 \\ . \\ . \\ . \\ z_1 \end{bmatrix} \begin{bmatrix} a_2 \\ . \\ . \\ . \\ z_2 \end{bmatrix} \cdots \begin{bmatrix} a_K \\ . \\ . \\ . \\ z_K \end{bmatrix}$$

Incorporating the Social Network

$$\begin{bmatrix} A_{1,1} & 0 & 0 & \dots & 0 \\ 0 & A_{2,2} & 0 & \dots & 0 \\ 0 & 0 & A_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & A_{dn,dn} \end{bmatrix} \qquad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_1 \\ \vdots \\ a_n \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

“Spread” Context Vector

$$A_{\otimes}^{-\frac{1}{2}} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_1 \\ \vdots \\ a_n \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{dn} \end{bmatrix}$$

Choosing an Action

Observe **K** context vectors

For each context vector, calculate a score:

- Sum of confidence bound **CB** and projected payoff **P**

Calculating a Score

Expected Payoff **P** + Confidence Bound **CB**

$$w^T \phi_k + \alpha \sqrt{\phi_{t,k}^T M_{t-1}^{-1} \phi_{t,k} \log(t+1)}$$

Updating Knowledge

M: add outer product of modified vectors -- encodes which **context** was seen with which **user**, and spreads the learned information across multiple blocks

$$M = M + \phi_{t,k}\phi_{t,k}^T$$

b: add modified context vector multiplied by payoff (same as LinUCB)

Issues With GOB.Lin

Relies on a matrix inversion scaling with the number of users ($O(n^2)$)

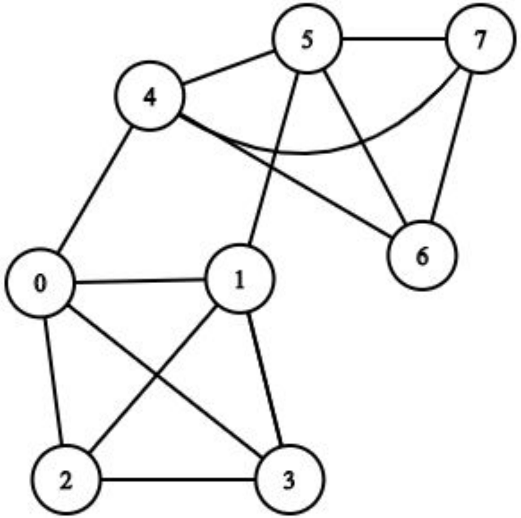
How to solve matrix inversion problem?

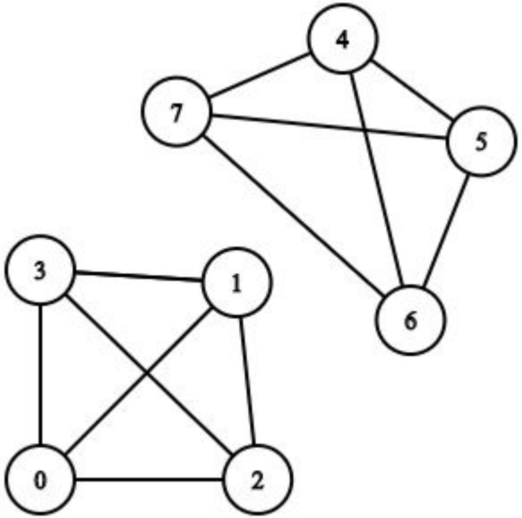
- Clustering to reduce number of users!

Two methods for using clustering

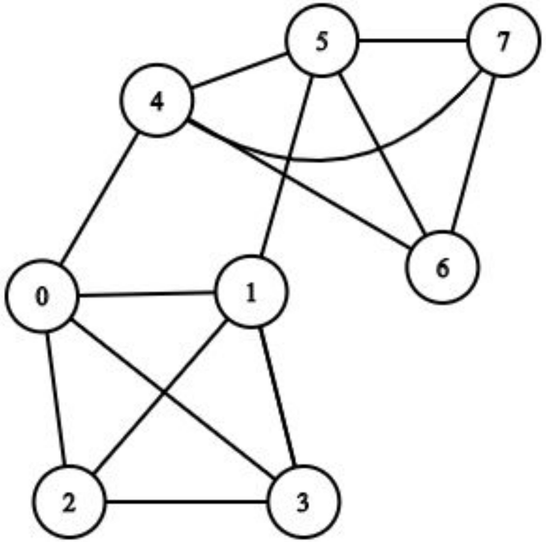
- GOB.Lin BLOCK
- GOB.Lin MACRO

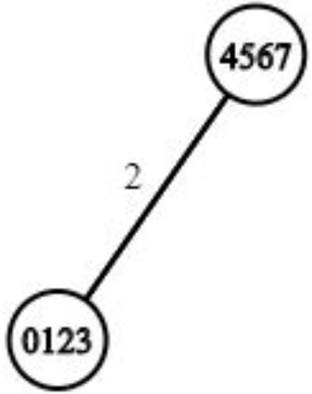
GOB.Lin BLOCK





GOB.Lin MACRO





Road Map


Introduction
Related Work
Our Project
Results
Conclusion

Data-Sets

4Cliques

- Small Artificial dataset

Last.fm

- Data from music streaming streaming service
- Fewer but more popular items (artists)

Delicious

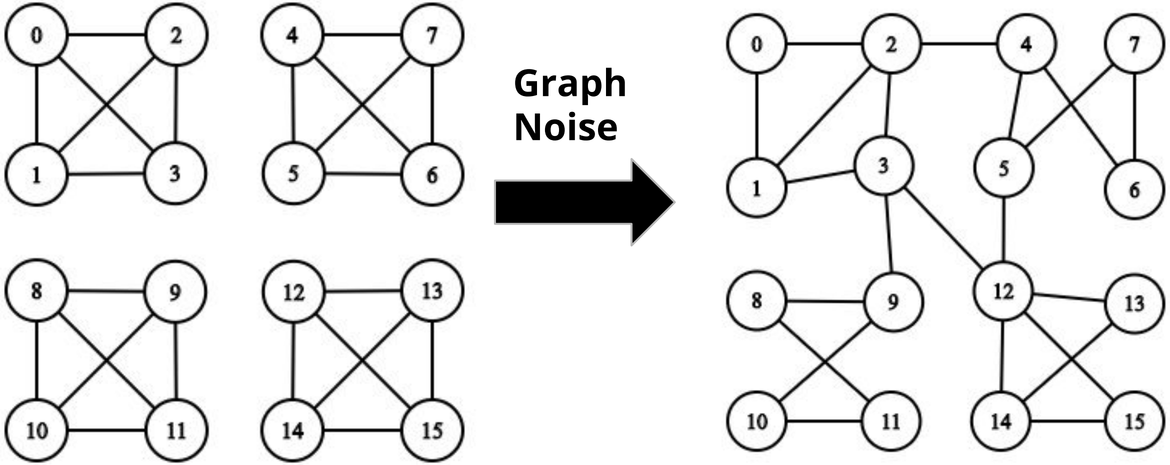
- Data from social bookmarking web service
- Many moderately popular items (websites)

4 Cliques

Graph starts as 4 cliques of 25 nodes each

Every node i in a clique is assigned the same preference vector u_i

Then add Graph Noise



4 Cliques

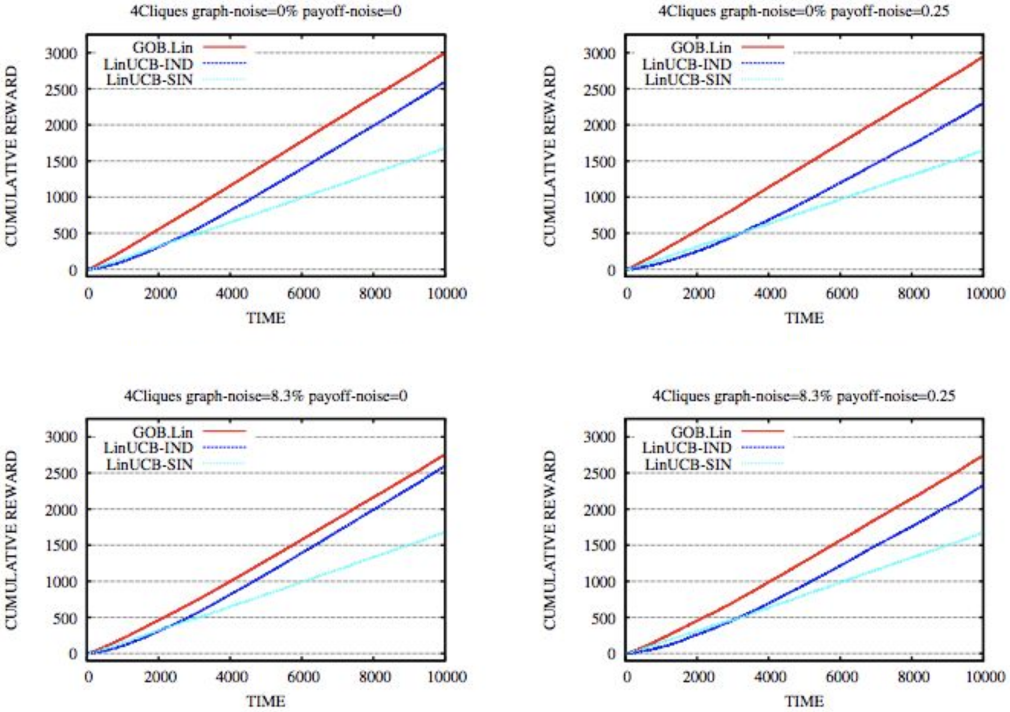
At every timestep, learner picks a random user and generates 10 random context vectors

Payoffs are calculated $a_i(x) = u_i^T x + \varepsilon$ where x is the chosen context and ε is the payoff noise uniformly distributed in a bounded interval around 0

4Cliques' Original Results

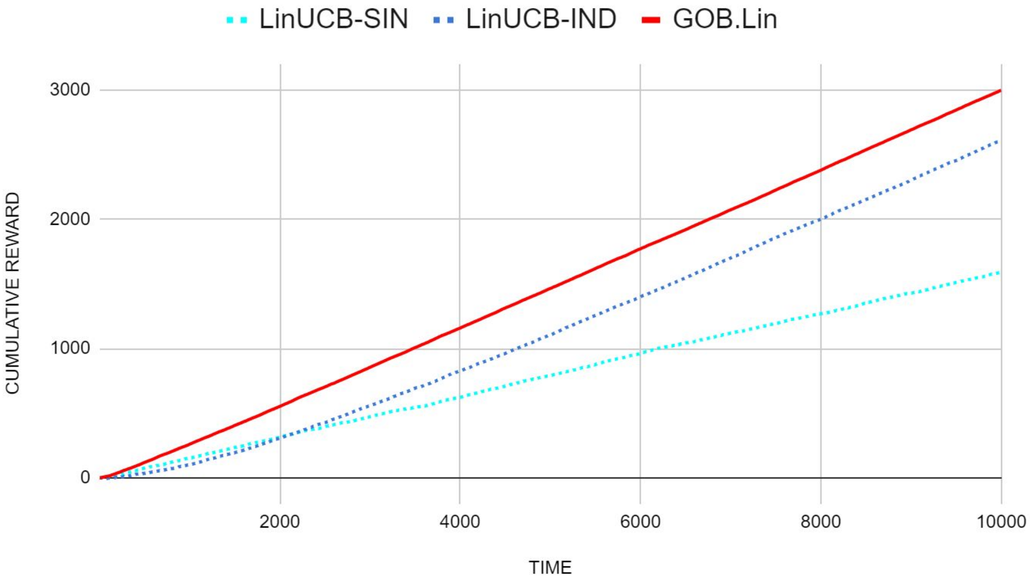
GOB.Lin robust to payoff noise

LinUCB not impacted by graph noise

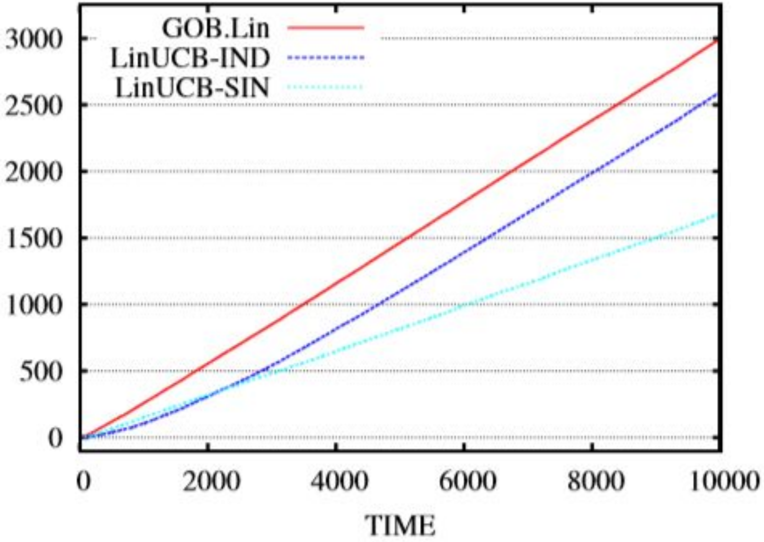


4 Cliques

Our Results



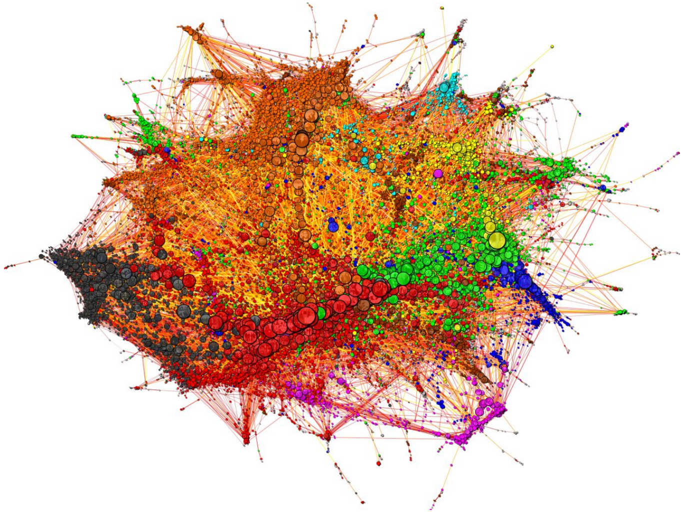
Their Results

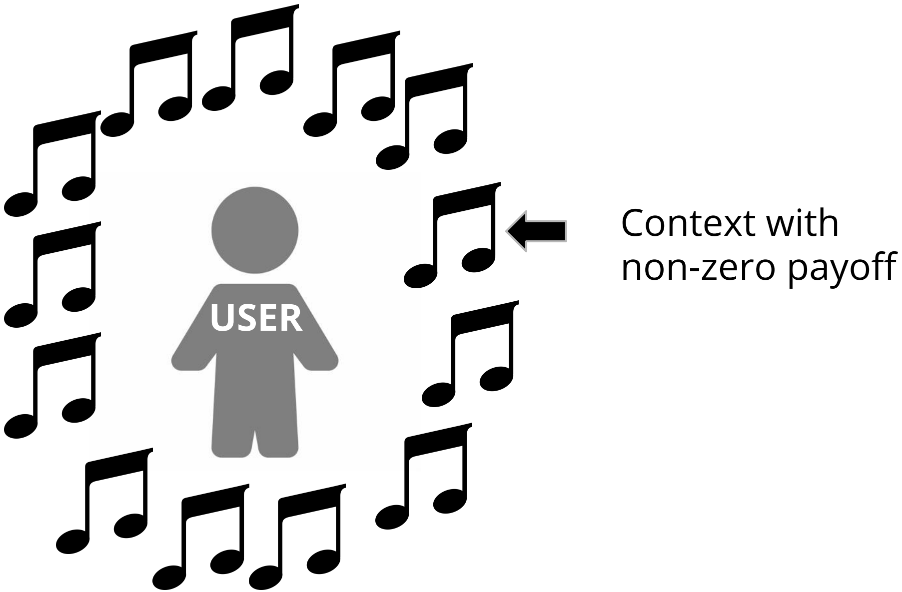


Last.fm and Delicious

1 Random User

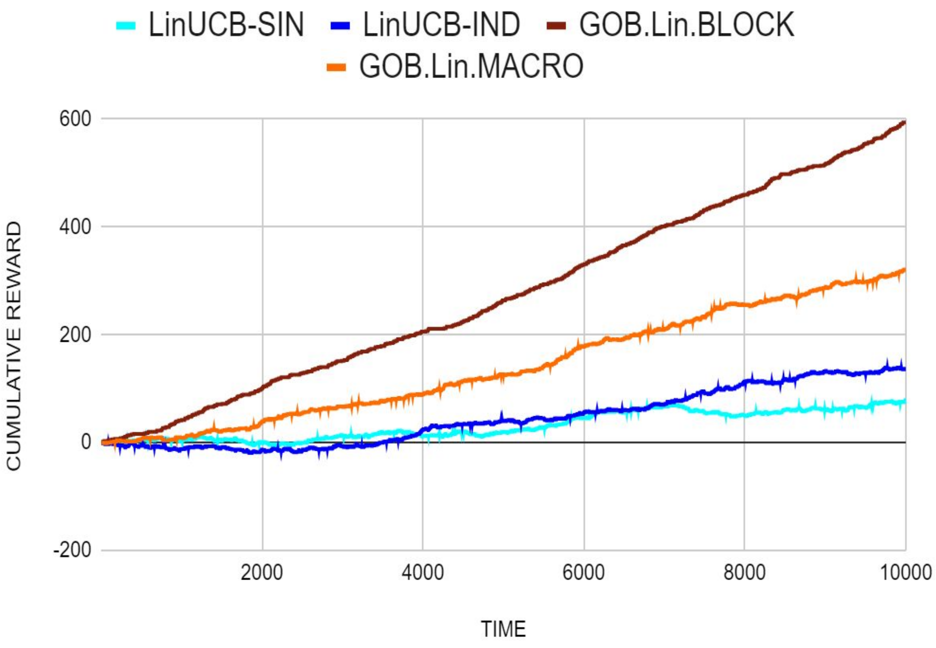
25 Random Contexts



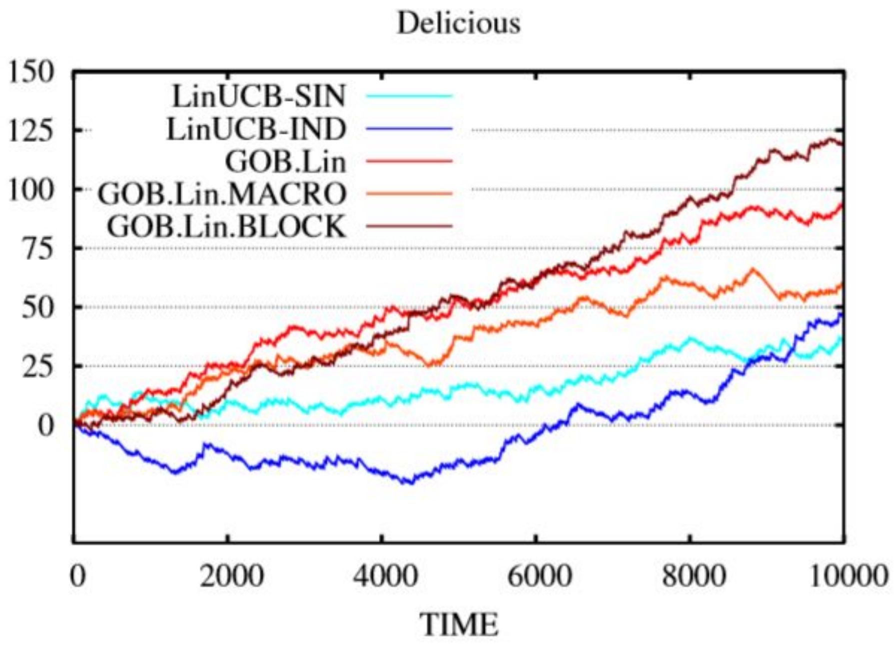


Delicious

Our Results

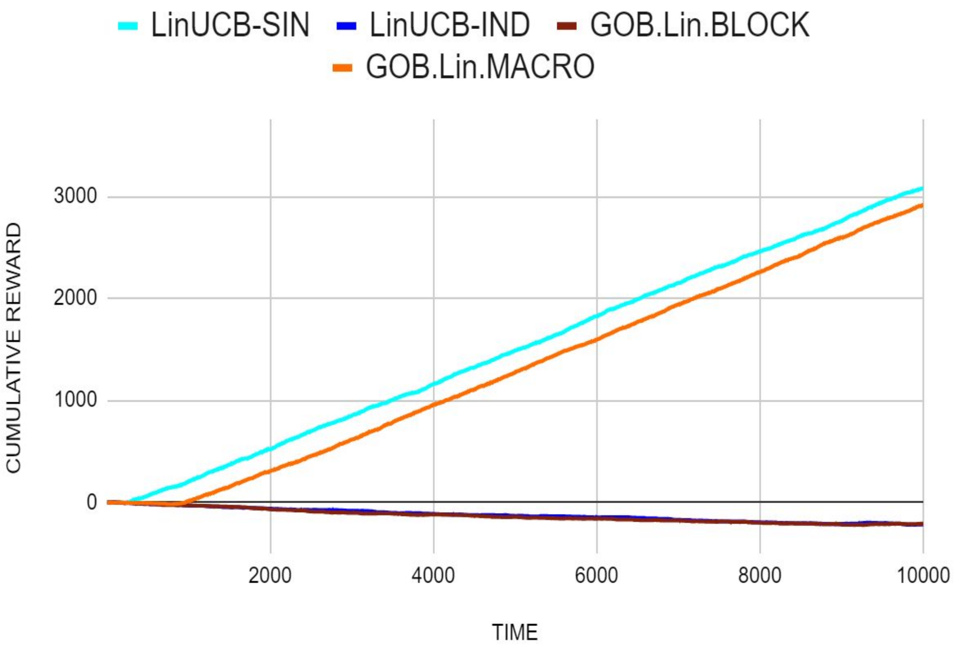


Their Results

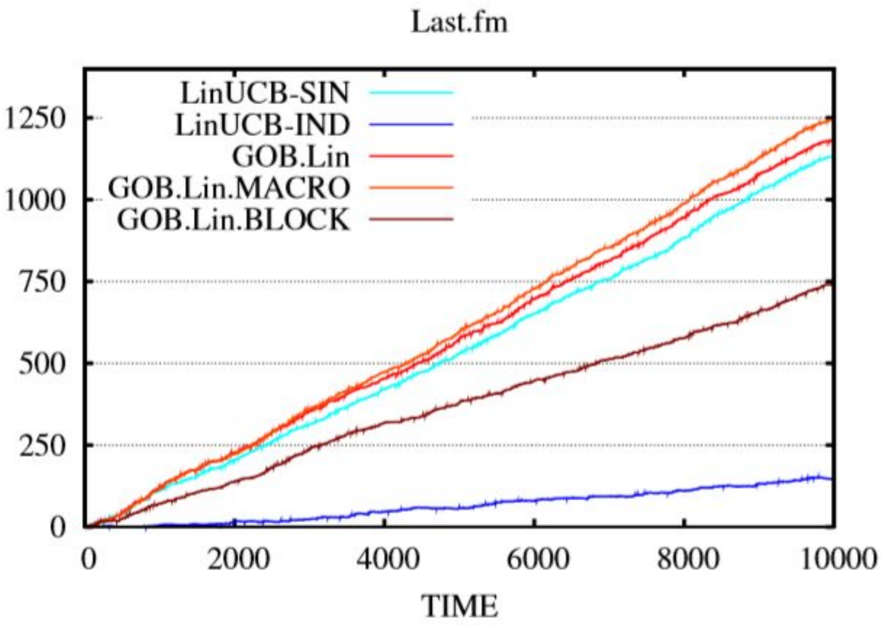


LastFM

Our Results



Their Results



Road Map


Introduction
Related Work
Our Project
Results
Conclusion

Successes

We implemented two linear bandit algorithms, as well as their variations

- LinUCB (Sin and Ind)
- GOB.Lin
  - Additionally implemented Block and Macro

On every dataset, our algorithms demonstrated the ability to learn

- This shows that the algorithms could be applicable to other recommendation-based scenarios

Challenges and Next Steps

GOB.Lin on Last.fm and Delicious was prohibitively slow and memory intensive

- We could not obtain results for GOB.Lin on these datasets

Ambiguity in paper

- Which **α** (exploration rate) to use
- How data from Last.fm and Delicious was processed
    - TFIDF
    - PCA
    - Clustering

Main Takeaways of Replication

Our results on Delicious and Last.fm differ from the researchers’ findings, but follow the same trends

- On Delicious, Block outperforms Macro
- On Last.fm, Macro outperforms Block
- Discrepancy in results may mean that Macro and Block are not as robust to changes in the dataset as the researchers make them out to seem

Our findings on 4Cliques validate what the researchers found

- This acts to bolster the foundation for more research to be conducted

Thank yous

Anna Rafferty’s server :(

Mike Tie

Paul, Hal, and Paul’s Pal for participating in our lightning talk

Anna Rafferty

- Fall term
- Winter term pre-tenure
- Winter term tenured
- All future Anna Raffertys

Work Cited

Cesa-Bianchi, Nicolo, Claudio Gentile, and Giovanni Zappella. "A gang of bandits." In *Advances in Neural Information Processing Systems*, pp. 737-745. 2013.

Chu, Wei, et al. "Contextual bandits with linear payoff functions." *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.* 2011.

Swapna Buccapatnam, Atilla Eryilmaz, and Ness B. Shroff. “Multi-armed Bandits in the Presence of Side Observations in Social Networks”. *52nd IEEE Conference on Decision and Control.* 2013.

Questions?