Learning with Limited Samples

Shouyuan Chen

The Chinese University of Hong Kong

October 29, 2014

Introduction

samples are sometimes very expensive.

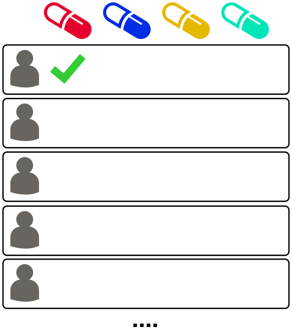
- decision making / prediction using a limited number of samples.

Introduction

samples are sometimes very expensive.

- decision making / prediction using a limited number of samples.

clinical trials

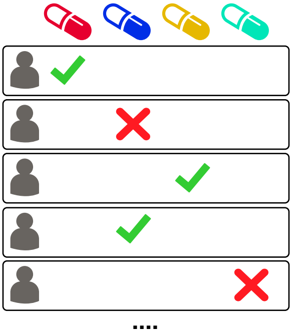


Introduction

samples are sometimes very expensive.

- decision making / prediction using a limited number of samples.

clinical trials

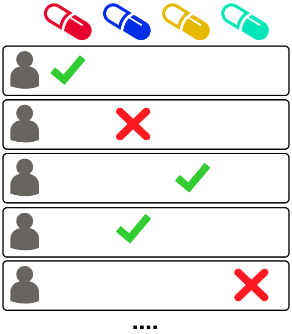


Introduction

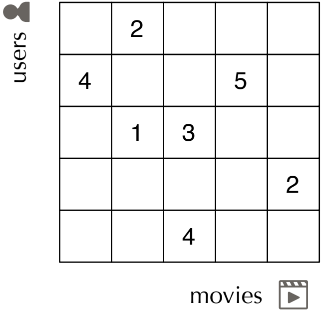
samples are sometimes very expensive.

- decision making / prediction using a limited number of samples.

clinical trials



movie recommendation

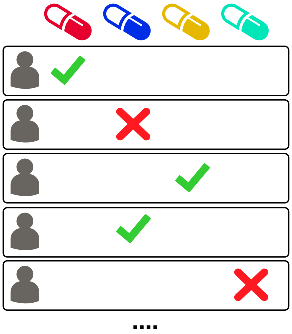


Introduction

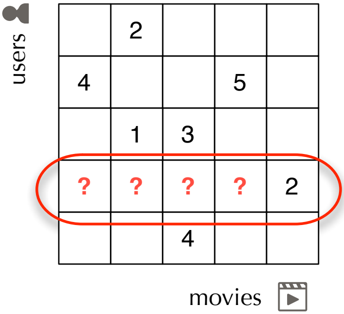
samples are sometimes very expensive.

- decision making / prediction using a limited number of samples.

clinical trials



movie recommendation

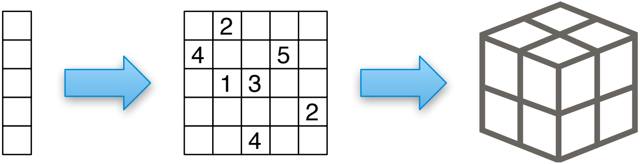


Introduction

multi-armed bandits

- sequential decision-making problem
- exploration and exploitation using limited samples

tensor completion



compressed sensing matrix completion tensor completion

Outline

multi-armed bandits

- Part II: Combinatorial pure exploration of multi-armed bandits
- Part III: Linear combinatorial bandits & Fast approximation for ridge regression

tensor completion

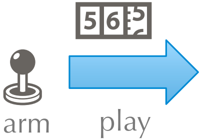
- Part IV: Exact and stable recovery for pairwise interaction Tensors

Part II

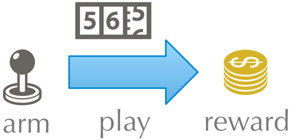
Combinatorial Pure Exploration of Multi-Armed Bandits

Single-armed bandit

Single-armed bandit



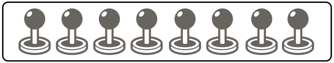
Single-armed bandit



sampled independently from
an **unknown** distribution
(reward distribution)

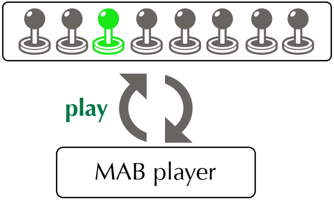
Multi-armed bandit

n arms

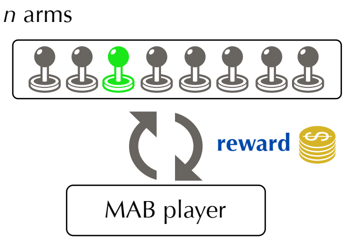


Multi-armed bandit

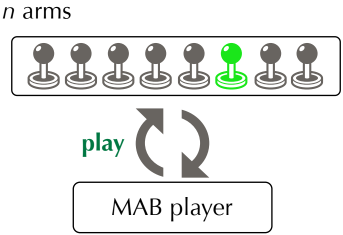
n arms



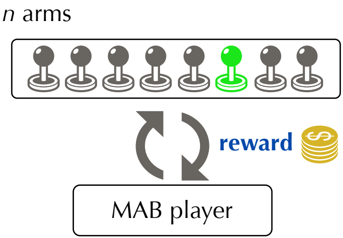
Multi-armed bandit



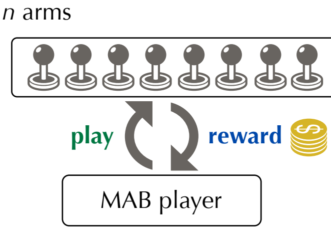
Multi-armed bandit



Multi-armed bandit

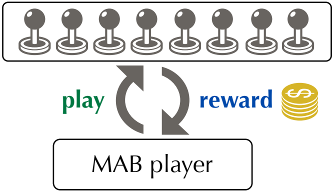


Multi-armed bandit

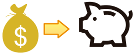


Multi-armed bandit

n arms



in the end...

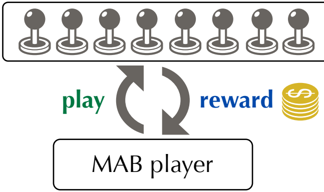
take all rewards 

goal: maximize the cumulative reward

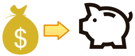
exploitation v.s. exploration

Multi-armed bandit

n arms



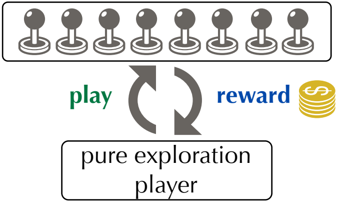
in the end...

take all rewards 

goal: maximize the cumulative reward

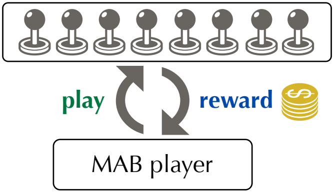
exploitation v.s. exploration

n arms **pure exploration**

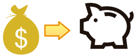


Multi-armed bandit

n arms



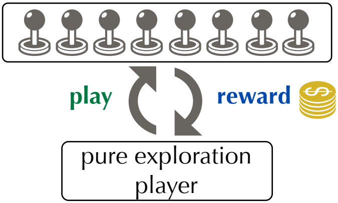
in the end...

take all rewards 

goal: maximize the cumulative reward

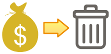
exploitation v.s. exploration

n arms **pure exploration**

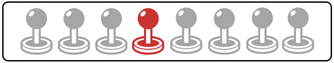


in the end...

(1) forfeit all rewards
(2) output **1** arm



goal: find the single **best arm**
(largest expected reward)



Combinatorial Pure Exploration of MAB

Combinatorial Pure Exploration (CPE)

- play one arm at each round
- find the optimal **set** of arms $M_* \in \mathcal{M}$
  - maximize the sum of expected rewards of arms in the set.
  - $\mathcal{M} \subseteq 2^{[n]}$ is the collection of admissible sets.

Combinatorial Pure Exploration of MAB

Combinatorial Pure Exploration (CPE)

- play one arm at each round
- find the optimal **set** of arms $M_* \in \mathcal{M}$
  - maximize the sum of expected rewards of arms in the set.
  - $\mathcal{M} \subseteq 2^{[n]}$ is the collection of admissible sets.

What kind of admissible sets?

Combinatorial Pure Exploration of MAB

Combinatorial Pure Exploration (CPE)

- play one arm at each round
- find the optimal **set** of arms $M_* \in \mathcal{M}$
  - maximize the sum of expected rewards of arms in the set.
  - $\mathcal{M} \subseteq 2^{[n]}$ is the collection of admissible sets.

What kind of admissible sets?

k-sets

Combinatorial Pure Exploration of MAB

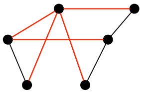
Combinatorial Pure Exploration (CPE)

- play one arm at each round
- find the optimal **set** of arms $M_* \in \mathcal{M}$
  - maximize the sum of expected rewards of arms in the set.
  - $\mathcal{M} \subseteq 2^{[n]}$ is the collection of admissible sets.

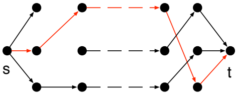
What kind of admissible sets?

k-sets

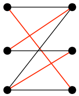
spanning trees



paths



matchings

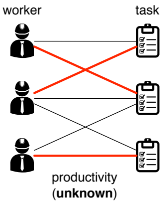


Motivating Examples

- *k*-sets
  - finding the top-k arms.

Motivating Examples

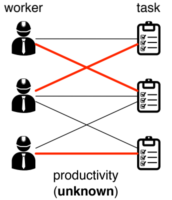
- *k*-sets
  - finding the top-*k* arms.
- matching



Goal:
1) estimate the productivities from tests.
2) find the optimal **1-1 assignment**.

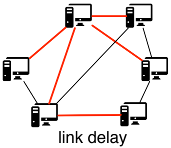
Motivating Examples

- *k*-sets
  - finding the top-*k* arms.
- matching



Goal:
1) estimate the productivities from tests.
2) find the optimal **1-1 assignment**.

- spanning trees and paths



Goal:
1) estimate the delays from measurements
2) find the minimum spanning tree
or shortest path.

Our Results

- Algorithms
  - two general learning algorithms for a wide range of $\mathcal{M}$.

Our Results

- Algorithms
  - two general learning algorithms for a wide range of $\mathcal{M}$.
- Upper bounds
  - sample complexity / probability of error.
  - **exchange class**: a new tool for analysis.

Our Results

- Algorithms
  - two general learning algorithms for a wide range of $\mathcal{M}$.
- Upper bounds
  - sample complexity / probability of error.
  - **exchange class**: a new tool for analysis.
- Lower bound
  - algorithms are optimal (within log factors) for many types of $\mathcal{M}$ (in particular, bases of a matroid).

Related Work

- Combinatorial bandits
  - sets of arms are played at each round.
  - minimizing the cumulative regret, instead of finding the best set.
    - the two problems are fundamentally different.

Related Work

- Combinatorial bandits
  - sets of arms are played at each round.
  - minimizing the cumulative regret, instead of finding the best set.
    - the two problems are fundamentally different.
- Pure exploration of multi-armed bandits
  - finding single best arm: matching upper and lower bounds are known.
  - finding top-k arms: only upper bounds are known.

Related Work

- Combinatorial bandits
  - sets of arms are played at each round.
  - minimizing the cumulative regret, instead of finding the best set.
    - the two problems are fundamentally different.
- Pure exploration of multi-armed bandits
  - finding single best arm: matching upper and lower bounds are known.
  - finding top-k arms: only upper bounds are known.
- **Our results**
  - the first lower bound of top-k problem.
  - the first upper and lower bounds for other combinatorial constraints.

Two Settings

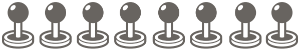
- Fixed budget
  - play for T rounds.
  - make the prediction after finished.
  - **goal**: minimize the probability of error

Two Settings

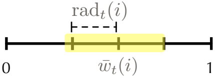
- Fixed budget
  - play for T rounds.
  - make the prediction after finished.
  - **goal**: minimize the probability of error
- Fixed confidence
  - play for any number of rounds.
  - make the prediction after finished
  - guarantee that probability of error $< \delta$.
  - **goal**: minimize the number of rounds (sample complexity).

CLUCB: Fixed confidence algorithm

all arms

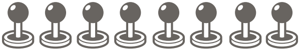


maintain: for all i and t

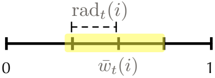


CLUCB: Fixed confidence algorithm

all arms



maintain: for all i and t

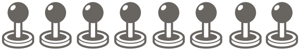


notations

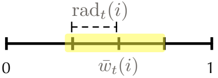
- for each arm $i \in [n]$ in each round t
  - empirical mean: $\bar{w}_t(i)$
  - confidence interval: $\mathrm{rad}_t(i)$ (proportional to $1/\sqrt{n_t(i)}$)

CLUCB: Fixed confidence algorithm

all arms



maintain: for all i and t



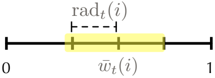
notations

- for each arm $i \in [n]$ in each round t
  - empirical mean: $\bar{w}_t(i)$
  - confidence interval: $\mathrm{rad}_t(i)$ (proportional to $1/\sqrt{n_t(i)}$)
- maximization oracle: $\mathrm{Oracle}(\cdots)$
  - $\mathrm{Oracle}(v) = \max_{M \in \mathcal{M}} \sum_{i \in M} v(i)$ for any n-dimensional vector v

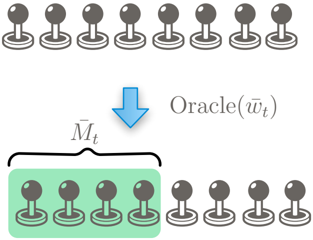
CLUCB: Fixed confidence algorithm

all arms

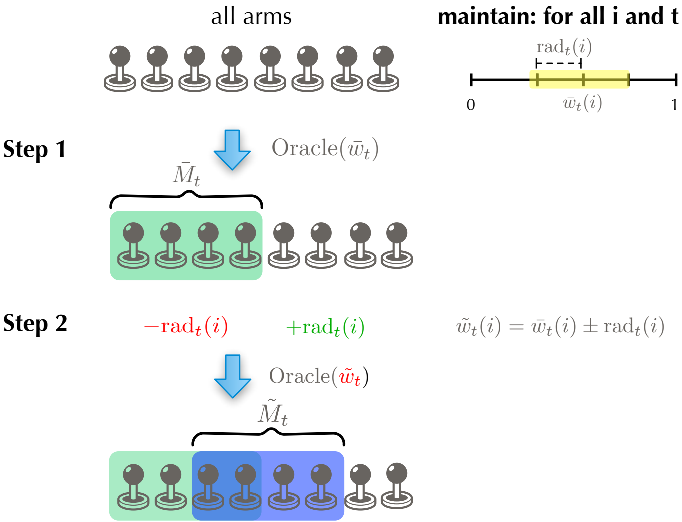
maintain: for all i and t



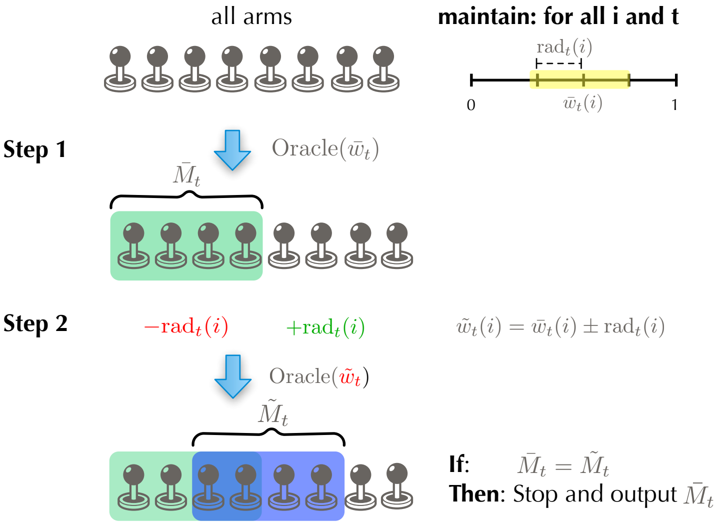
Step 1



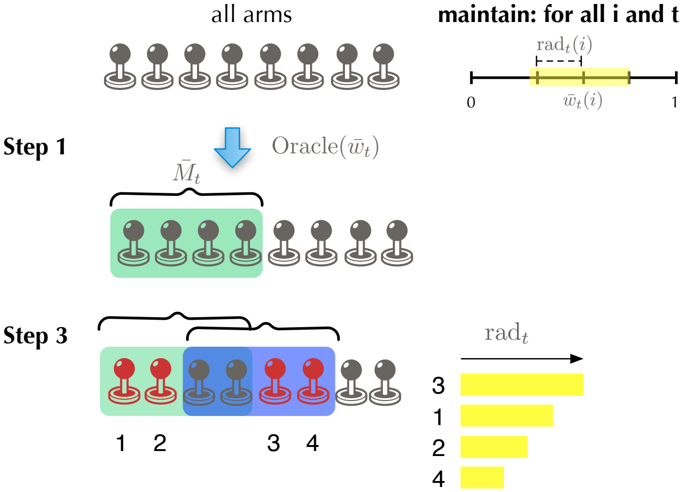
CLUCB: Fixed confidence algorithm



CLUCB: Fixed confidence algorithm

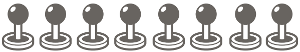


CLUCB: Fixed confidence algorithm

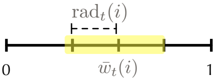


CLUCB: Fixed confidence algorithm

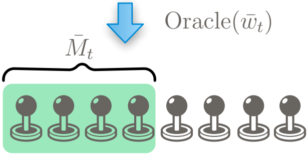
all arms



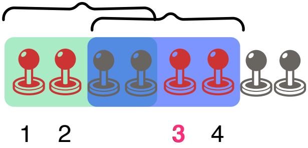
maintain: for all i and t

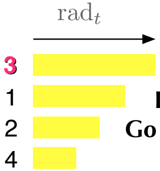


Step 1



Step 3





Play arm 3!
Go to t+1 round.

CLUCB: Sample Complexity

Our sample complexity bound depends on two quantities.

- $\mathbf{H}$: only depends on expected rewards
- $\mathrm{width}(\mathcal{M})$: only depends on the structure of $\mathcal{M}$

CLUCB: Sample Complexity

Our sample complexity bound depends on two quantities.

- $\mathbf{H}$: only depends on expected rewards
- $\mathrm{width}(\mathcal{M})$: only depends on the structure of $\mathcal{M}$

Theorem

With probability at least $1-\delta$, CLUCB algorithm:

1. *outputs the optimal set $M_* \triangleq \arg\max_{M\in\mathcal{M}} w(M)$.*
2. *uses at most $O(\mathrm{width}(\mathcal{M})^2 \mathbf{H} \log(n\mathbf{H}/\delta))$ rounds.*

Sample complexity (1): $\mathbf{H}$

- Δ_e: gap of arm $e \in [n]$

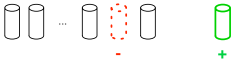
$$\Delta_e = \begin{cases} w(M_*) - \max_{M \in \mathcal{M}: e \in M} w(M) & \text{if } e \notin M_*, \\ w(M_*) - \max_{M \in \mathcal{M}: e \notin M} w(M) & \text{if } e \in M_* \end{cases}$$

  - stability of the optimality of M_* regarding to arm e.
- $\mathbf{H} = \sum_{e \in [n]} \Delta_e^{-2}$

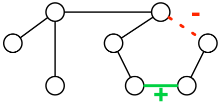
Exchange class: Overview

Intuitions

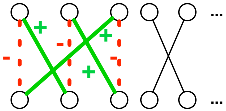
- An exchange class is a "proxy" for the structure of $\mathcal{M}$ in the analysis.
- An exchange class is a collection of "patches" that are used to interpolate between subsets.



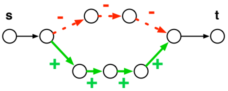
Top-K



Spanning tree



Matching



Path

Exchange class: Formal definition

Exchange set

An **exchange set** b is an ordered pair of disjoint sets $b = (b_+, b_-)$ where $b_+ \cap b_- = \emptyset$ and $b_+, b_- \subseteq [n]$.
Let M be any set. We also define two operators:

- $M \oplus b \triangleq M \backslash b_- \cup b_+$.
- $M \ominus b \triangleq M \backslash b_+ \cup b_-$.

Exchange class

We call a collection of exchange sets $\mathcal{B}$ an **exchange class for** $\mathcal{M}$ if $\mathcal{B}$ satisfies the following property. For any $M, M' \in \mathcal{M}$ such that $M \neq M'$ and for any $e \in (M \backslash M')$, there exists an exchange set $(b_+, b_-) \in \mathcal{B}$ which satisfies five constraints: **(a)** $e \in b_-$, **(b)** $b_+ \subseteq M' \backslash M$, **(c)** $b_- \subseteq M \backslash M'$, **(d)** $(M \oplus b) \in \mathcal{M}$ and **(e)** $(M' \ominus b) \in \mathcal{M}$.

Exchange class: Width

Width: definition

$$\text{width}(\mathcal{B}) = \max_{(b_+,b_-)\in\mathcal{B}} |b_+| + |b_-|.$$

$$\text{width}(\mathcal{M}) = \min_{\mathcal{B}\in\text{Exchange}(\mathcal{M})} \text{width}(\mathcal{B}),$$

where Exchange($\mathcal{M}$) is the family of all possible exchange classes for $\mathcal{M}$.

Width: examples

- *k*-**sets, spanning tree, matroids:** width($\mathcal{M}$) $= 2$.
- **matchings, paths (in DAG)** width($\mathcal{M}$) $= O(|V|)$.

Sample complexity of examples

Recall that

Theorem

With probability at least $1-\delta$, CLUCB algorithm:

1. *outputs the optimal set M_*.*
2. *uses at most $\tilde{O}(\mathrm{width}(\mathcal{M})^2\mathbf{H})$ rounds.*

Sample complexity of examples

Recall that

Theorem

With probability at least $1 - \delta$, CLUCB algorithm:

1. *outputs the optimal set M_*.*
2. *uses at most $\tilde{O}(\mathrm{width}(\mathcal{M})^2 \mathbf{H})$ rounds.*

Plug in the widths of examples

Corollary (Sample Complexity of Examples)

- *k-sets, spanning trees, bases of a matroid*: $\tilde{O}(\mathbf{H})$.
- *matchings, paths (in DAG)*: $\tilde{O}(|V|^2 \mathbf{H})$.

Lower bound

An algorithm $\mathbb{A}$ is a δ-correct algorithm, if $\mathbb{A}$'s probability of error is at most δ for any expected rewards.

Lower bound

An algorithm $\mathbb{A}$ is a δ-correct algorithm, if $\mathbb{A}$'s probability of error is at most δ for any expected rewards.

Theorem (Problem dependent lower bound)

Given any expected rewards, any δ-correct algorithm must use at least $\Omega(\mathbf{H}\log(1/\delta))$ rounds.

Lower bound

An algorithm $\mathbb{A}$ is a δ-correct algorithm, if $\mathbb{A}$'s probability of error is at most δ for any expected rewards.

Theorem (Problem dependent lower bound)

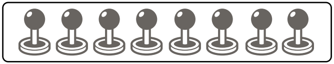
Given any expected rewards, any δ-correct algorithm must use at least $\Omega(\mathbf{H}\log(1/\delta))$ rounds.

Remarks:

- *k*-sets, spanning trees, bases of a matroid: CLUCB's sample complexity $\tilde{O}(\mathbf{H})$ is optimal (up to log factors).
- other $\mathcal{M}$ in general: a gap of $\tilde{O}(\mathrm{width}(\mathcal{M})^2) = \tilde{O}(n^2)$.

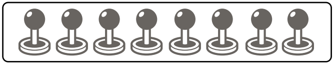
CSAR: Fixed budget algorithm

phase 1



CSAR: Fixed budget algorithm

phase 1



in each phase (n phases in total):

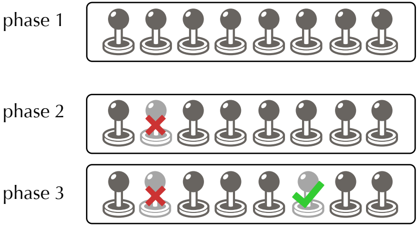
- 1 arm is accepted or rejected.
- active arms are sampled for a same number of times.

active: neither accepted nor rejected. require more samples

accepted: include in the output

rejected: exclude from the output

CSAR: Fixed budget algorithm



in each phase (n phases in total):

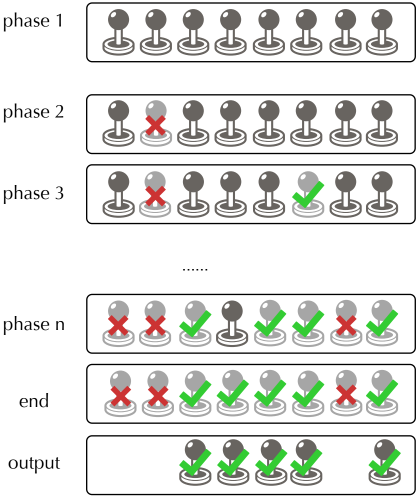
- 1 arm is accepted or rejected.
- active arms are sampled for a same number of times.


active: neither accepted nor rejected. require more samples


accepted: include in the output


rejected: exclude from the output

CSAR: Fixed budget algorithm



in each phase (n phases in total):

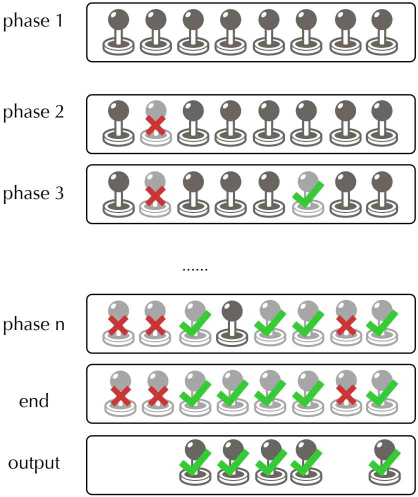
- 1 arm is accepted or rejected.
- active arms are sampled for a same number of times.

active: neither accepted nor rejected. require more samples

accepted: include in the output

rejected: exclude from the output

CSAR: Fixed budget algorithm



in each phase (*n* phases in total):

- 1 arm is accepted or rejected.
- active arms are sampled for a same number of times.

active: neither accepted nor rejected. require more samples

accepted: include in the output

rejected: exclude from the output

problem: which arm to accept or reject?

CSAR: Fixed budget algorithm

problem: which arm to accept or reject?

- accept/reject the arm with the largest **empirical gap**.

$$\bar{\Delta}_e = \begin{cases} \bar{w}_t(\bar{M}_t) - \max\limits_{M \in \mathcal{M}_t : e \in M} \bar{w}_t(M) & \text{if } e \notin \bar{M}_t, \\ \bar{w}_t(\bar{M}_t) - \max\limits_{M \in \mathcal{M}_t : e \notin M} \bar{w}_t(M) & \text{if } e \in \bar{M}_t \end{cases}$$

  - $\mathcal{M}_t = \{M : M \in \mathcal{M}, A_t \subseteq M, B_t \cap M = \emptyset\}$.
  - A_t: accepted arms, B_t: rejected arms (up to phase t).

CSAR: Fixed budget algorithm

problem: which arm to accept or reject?

- accept/reject the arm with the largest **empirical gap**.

$$\bar{\Delta}_e = \begin{cases} \bar{w}_t(\bar{M}_t) - \max_{M \in \mathcal{M}_t : e \in M} \bar{w}_t(M) & \text{if } e \notin \bar{M}_t, \\ \bar{w}_t(\bar{M}_t) - \max_{M \in \mathcal{M}_t : e \notin M} \bar{w}_t(M) & \text{if } e \in \bar{M}_t \end{cases}$$

  - $\mathcal{M}_t = \{M : M \in \mathcal{M}, A_t \subseteq M, B_t \cap M = \emptyset\}$.
  - A_t: accepted arms, B_t: rejected arms (up to phase t).
  - $\bar{\Delta}_e$ can be computed using a maximization oracle.

CSAR: Fixed budget algorithm

problem: which arm to accept or reject?

- accept/reject the arm with the largest **empirical gap**.

$$\bar{\Delta}_e = \begin{cases} \bar{w}_t(\bar{M}_t) - \max\limits_{M \in \mathcal{M}_t : e \in M} \bar{w}_t(M) & \text{if } e \notin \bar{M}_t, \\ \bar{w}_t(\bar{M}_t) - \max\limits_{M \in \mathcal{M}_t : e \notin M} \bar{w}_t(M) & \text{if } e \in \bar{M}_t \end{cases}$$

  - $\mathcal{M}_t = \{M : M \in \mathcal{M}, A_t \subseteq M, B_t \cap M = \emptyset\}$.
  - A_t: accepted arms, B_t: rejected arms (up to phase t).
  - $\bar{\Delta}_e$ can be computed using a maximization oracle.

- recall the (unknown) **gap** of arm e:

$$\Delta_e = \begin{cases} w(M_*) - \max_{M \in \mathcal{M} : e \in M} w(M) & \text{if } e \notin M_*, \\ w(M_*) - \max_{M \in \mathcal{M} : e \notin M} w(M) & \text{if } e \in M_* \end{cases}$$

CSAR: Probability of error

Theorem (Probability of error of CSAR)

Given any budget $T > n$, CSAR correctly outputs the optimal set M_* with probability at least

$$1 - 2^{\tilde{O}\left(-\frac{T}{\text{width}(\mathcal{M})^2 \mathbf{H}}\right)}$$

and uses at most T rounds.

CSAR: Probability of error

Theorem (Probability of error of CSAR)

Given any budget $T > n$, CSAR correctly outputs the optimal set M_* with probability at least

$$1 - 2^{\tilde{O}\left(-\frac{T}{\mathrm{width}(\mathcal{M})^2 \mathbf{H}}\right)}$$

and uses at most T rounds.

Remark: To guarantee a constant probability of error of δ, both CSAR and CLUCB need $T = \tilde{O}(\mathrm{width}(\mathcal{M})^2 \mathbf{H})$ rounds.

Summary

- combinatorial pure exploration: a general framework that covers many pure exploration problems in MAB.
  - find top-k arms, optimal spanning trees, matchings or paths.
- two general algorithms for the problem
  - only need a maximization oracle for $\mathcal{M}$.
  - comparable performance guarantees.
- our algorithm is optimal (up to log factors) for matroids.
  - including k-sets and spanning trees.

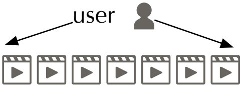
Part III

Linear Combinatorial Bandits

&

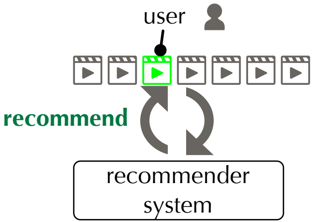
Fast relative-error approximation for ridge regression

Linear bandits

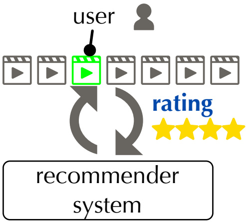


recommender
system

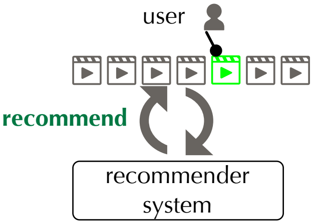
Linear bandits



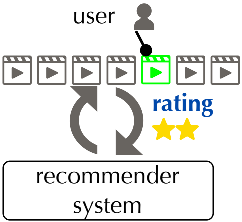
Linear bandits



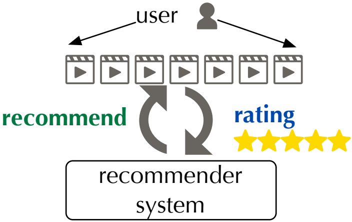
Linear bandits



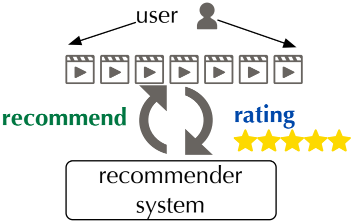
Linear bandits

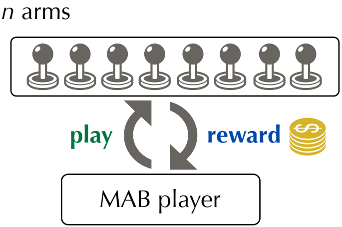


Linear bandits

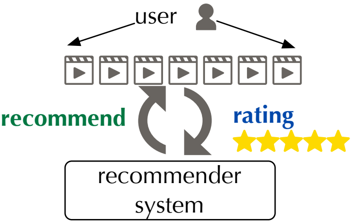


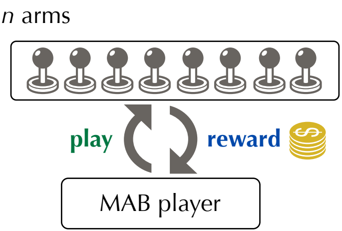
Linear bandits





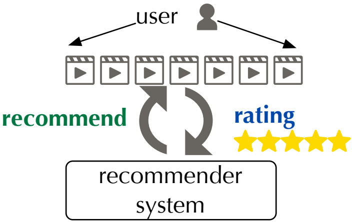
Linear bandits

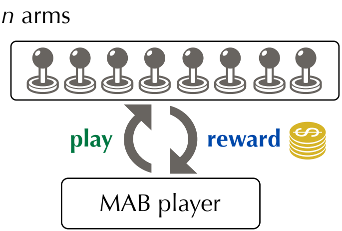




- the number of arms (movies) n is very large
  - challenge: many arms will never be played.
  - solution: more assumptions on the rewards (ratings)

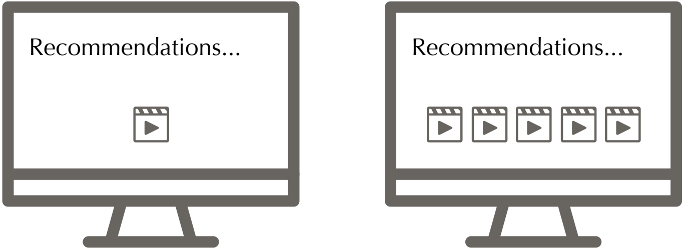
Linear bandits





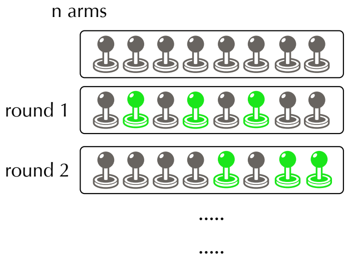
- the number of arms (movies) n is very large
  - challenge: many arms will never be played.
  - solution: more assumptions on the rewards (ratings)
- **linear bandits**
  - each arm i has a feature vector $\mathbf{v}_i \in \mathbb{R}^d$
  - an unknown vector $\mathbf{u} \in \mathbb{R}^d$
  - playing arm i gives a random reward $r_i = \mathbf{u}^T \mathbf{v}_i + \epsilon$
    - ϵ is a zero-mean r.v.
  - algorithms with $\tilde{O}(\sqrt{T})$ regret [APS11].

- rarely recommend a single movie.



- rarely recommend a single movie.
- better recommend a set of movies.
  - a set of movies that are favorable and diverse.

Linear Combinatorial Bandits



linear combinatorial bandits

- a set of arms $S_t \in \mathcal{M}$ are played on each round t.
- observation: rewards $\{r_i^{(t)} \mid i \in S_t\}$
  - $r_i^{(t)} = \mathbf{u}^T \mathbf{v}_i + \epsilon_i^{(t)}$
- reward function: the player earns a reward $f_{\mathbf{r}^{(t)}, \mathbf{V}}(S_t)$.

Reward function

we allow a broad class of $f_{\mathbf{r},\mathbf{V}}(S_t)$ that satisfy

- **monotone** and **Lipschitz continuous** in terms of $\mathbf{r}$.
- an α-maximization oracle
    - approximation ratio $\alpha \in (0, 1]$.

Reward function

we allow a broad class of $f_{\mathbf{r},\mathbf{V}}(S_t)$ that satisfy

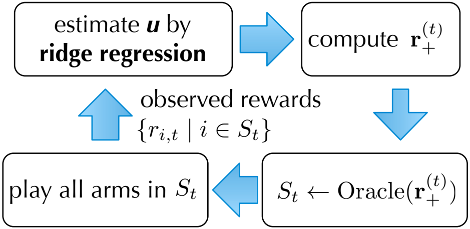
- **monotone** and **Lipschitz continuous** in terms of $\mathbf{r}$.
- an α-maximization oracle
  - approximation ratio $\alpha \in (0, 1]$.

a reward function for movie recommendation

$$f_{\mathbf{r},\mathbf{V}}(S) = \underbrace{\sum_{i \in S} r_i}_{\text{sum of ratings}} + \lambda \underbrace{g(\{\mathbf{v}_i \mid i \in S\})}_{\text{diversity of movies}}.$$

- QZ13 proposed such a $g(X)$ using log-determinants
  - maximal when vectors in X are orthogonal
  - submodular and monotone
    - greedy algorithm has approximation ratio $1 - 1/e$
    - $\implies$ a $(1 - 1/e)$-maximization oracle

Algorithm and Analysis



Theorem

The algorithm's α-regret over T rounds is $\tilde{O}(\sqrt{T})$.

- *α-regret: $\alpha OPT(T) - \sum_{i=1}^{T} f_{\mathbf{r}^{(t)},\mathbf{V}}(S_t)$*
- *$OPT(T)$: the largest possible reward from T rounds*

Ridge regression

ridge regression problem

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_2^2 .$$

- design matrix: $\mathbf{A} \in \mathbb{R}^{n \times p}$ and response vector: $\mathbf{b} \in \mathbb{R}^p$

optimal solution

$$\mathbf{x}_* = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda \mathbf{I}_n)^{-1}\mathbf{b}.$$

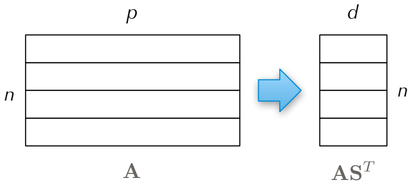
- time complexity: $O(n^2 p)$
- no known algorithms are asymptotically faster.

challenge

$$n \gg p \gg 1$$

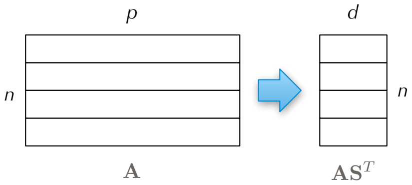
Fast relative-error approximation

oblivious subspace embedding (OSE)



Fast relative-error approximation

oblivious subspace embedding (OSE)

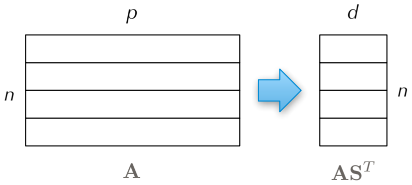


our OSE based solution

$$\tilde{\mathbf{x}} = \mathbf{A}^T(\mathbf{AS}^T)^{\dagger T}(\lambda(\mathbf{AS}^T)^{\dagger T} + \mathbf{AS}^T)^{\dagger}\mathbf{b}.$$

Fast relative-error approximation

oblivious subspace embedding (OSE)



our OSE based solution

$$\tilde{\mathbf{x}} = \mathbf{A}^T(\mathbf{AS}^T)^{\dagger T}(\lambda(\mathbf{AS}^T)^{\dagger T} + \mathbf{AS}^T)^{\dagger}\mathbf{b}.$$

Theorem

Given $\epsilon > 0$, there exists a way to construct $\mathbf{S}$ such that, with high probability,

$$\|\tilde{\mathbf{x}} - \mathbf{x}_*\|_2 \leq \epsilon \|\mathbf{x}_*\|_2$$

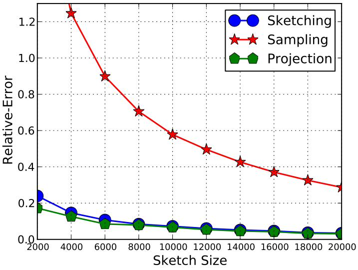
and the algorithm runs in $O(\mathrm{nnz}(\mathbf{A}) + n^3/\epsilon)$ time.

Experiments

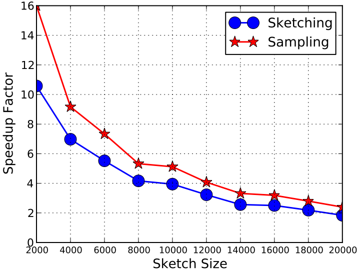
baselines

- sample: randomly select features
- project: compress $\mathbf{A}$ using random projection.

relative error



speedup factors



Summary

- linear combinatorial bandits
  - a generalization of linear bandits to allow multiple plays
  - allow complicated reward functions
  - an algorithm with asymptotically no-regret
    - use ridge regression to estimate the unknown
  - application: diversified movie sets recommendation
- fast ridge regression
  - the first algorithm in $o(n^2p)$ time with relative-error guarantee

Part IV
Recovery for Pairwise Interaction Tensors

Matrix completion

1	2	2	5	8
4	2	3	5	5
2	1	3	4	2
5	5	6	4	2
7	7	4	2	3

movies

users

Matrix completion

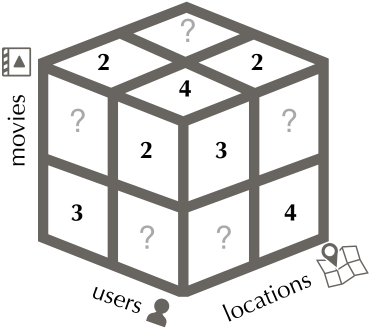
movies

?	**2**	?	?	**8**
4	?	?	**5**	?
?	**1**	**3**	?	?
?	?	?	?	**2**
?	?	**4**	?	?

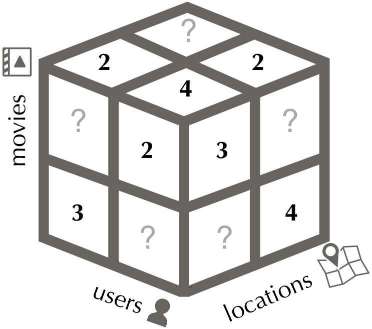
users

- matrix completion: recover the missing entries.
- **exact recovery** for *low rank* matrices!
  - via convex programming.
  - need $\tilde{O}(nr)$ samples (observed entries).

Tensor completion

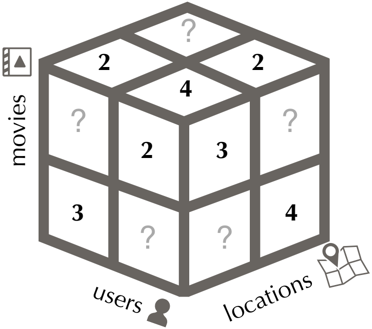


Tensor completion



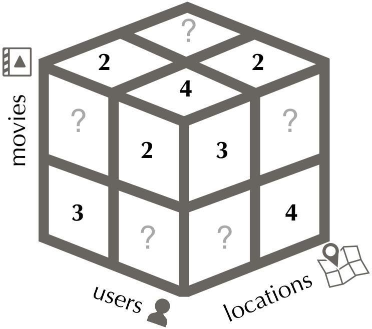
- tensor completion: recover the missing entries.

Tensor completion



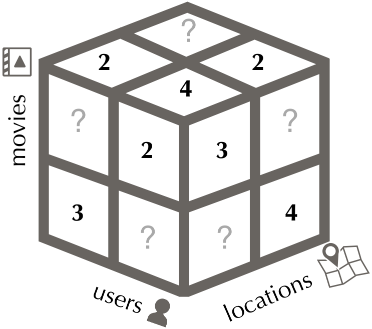
- tensor completion: recover the missing entries.
- bad news: much harder than matrix completion!
  - low rank tensors?
  - even computing the rank is NP-hard.

Tensor completion



- tensor completion: recover the missing entries.
- bad news: much harder than matrix completion!
  - low rank tensors?
  - even computing the rank is NP-hard.
- special tensors?

Tensor completion



- tensor completion: recover the missing entries.
- bad news: much harder than matrix completion!
  - low rank tensors?
  - even computing the rank is NP-hard.
- special tensors?
- pairwise interaction tensors!

Pairwise Interaction Tensor

definition

$$T_{ijk} = A_{ij} + B_{jk} + C_{ki} \quad \forall (i, j, k) \in [n_1] \times [n_2] \times [n_3]$$

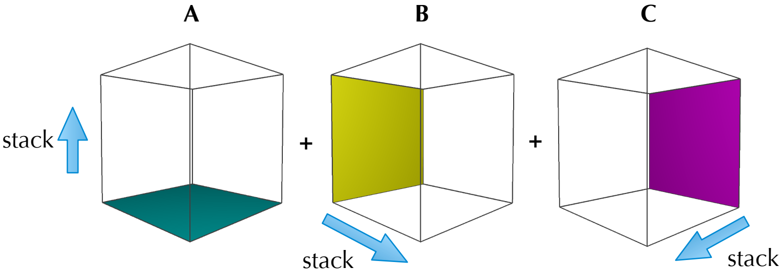
- $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}, \mathbf{B} \in \mathbb{R}^{n_2 \times n_3}, \mathbf{C} \in \mathbb{R}^{n_3 \times n_1}$.
- denote $\mathbf{T} = \mathsf{Pair}(\mathbf{A}, \mathbf{B}, \mathbf{C})$

Pairwise Interaction Tensor

definition

$$T_{ijk} = A_{ij} + B_{jk} + C_{ki} \quad \forall (i,j,k) \in [n_1] \times [n_2] \times [n_3]$$

- $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}, \mathbf{B} \in \mathbb{R}^{n_2 \times n_3}, \mathbf{C} \in \mathbb{R}^{n_3 \times n_1}$.
- denote $\mathbf{T} = \mathrm{Pair}(\mathbf{A}, \mathbf{B}, \mathbf{C})$



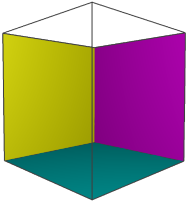
Pairwise Interaction Tensor

definition

$$T_{ijk} = A_{ij} + B_{jk} + C_{ki} \quad \forall (i, j, k) \in [n_1] \times [n_2] \times [n_3]$$

- $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}, \mathbf{B} \in \mathbb{R}^{n_2 \times n_3}, \mathbf{C} \in \mathbb{R}^{n_3 \times n_1}$.
- denote $\mathbf{T} = \mathrm{Pair}(\mathbf{A}, \mathbf{B}, \mathbf{C})$

Pair(A,B,C)



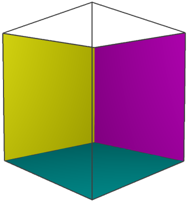
Pairwise Interaction Tensor

definition

$$T_{ijk} = A_{ij} + B_{jk} + C_{ki} \quad \forall (i,j,k) \in [n_1] \times [n_2] \times [n_3]$$

- $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}, \mathbf{B} \in \mathbb{R}^{n_2 \times n_3}, \mathbf{C} \in \mathbb{R}^{n_3 \times n_1}$.
- denote $\mathbf{T} = \text{Pair}(\mathbf{A}, \mathbf{B}, \mathbf{C})$

Pair(A,B,C)



- good model for tag/item recommendations [RT10, RFS10].

Recovery via convex programming

$$T_{ijk} = A_{ij} + B_{jk} + C_{ki} \quad \forall (i, j, k) \in [n_1] \times [n_2] \times [n_3]$$

- observed entries: $\Omega = \{(i_1, j_1, k_1), \ldots, (i_m, j_m, k_m)\}$.
  - $\mathbf{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ unknown outside of Ω
- $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}, \mathbf{B} \in \mathbb{R}^{n_2 \times n_3}, \mathbf{C} \in \mathbb{R}^{n_3 \times n_1}$: unknown
  - goal: recover $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Recovery via convex programming

$$T_{ijk} = A_{ij} + B_{jk} + C_{ki} \quad \forall (i,j,k) \in [n_1] \times [n_2] \times [n_3]$$

- observed entries: $\Omega = \{(i_1, j_1, k_1), \ldots, (i_m, j_m, k_m)\}$.
  - $\mathbf{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ unknown outside of Ω
- $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}, \mathbf{B} \in \mathbb{R}^{n_2 \times n_3}, \mathbf{C} \in \mathbb{R}^{n_3 \times n_1}$: unknown
  - goal: recover $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Recovery via trace-norm minimization

$$\begin{array}{ll} \text{minimize} & \sqrt{n_3}\left\|\hat{\mathbf{A}}\right\|_* + \sqrt{n_1}\left\|\hat{\mathbf{B}}\right\|_* + \sqrt{n_2}\left\|\hat{\mathbf{C}}\right\|_* \\ \text{subject to} & T_{ijk} = \hat{A}_{ij} + \hat{B}_{jk} + \hat{C}_{ki} \quad \forall (i,j,k) \in \Omega \end{array}$$

Exact recovery

$$\begin{array}{ll} \text{minimize} & \sqrt{n_3}\left\|\hat{\mathbf{A}}\right\|_* + \sqrt{n_1}\left\|\hat{\mathbf{B}}\right\|_* + \sqrt{n_2}\left\|\hat{\mathbf{C}}\right\|_* \\ \text{subject to} & T_{ijk} = \hat{A}_{ij} + \hat{B}_{jk} + \hat{C}_{ki} \quad \forall (i,j,k) \in \Omega \end{array}$$

Exact recovery

$$\begin{array}{ll} \text{minimize} & \sqrt{n_3}\left\|\hat{\mathbf{A}}\right\|_* + \sqrt{n_1}\left\|\hat{\mathbf{B}}\right\|_* + \sqrt{n_2}\left\|\hat{\mathbf{C}}\right\|_* \\ \text{subject to} & T_{ijk} = \hat{A}_{ij} + \hat{B}_{jk} + \hat{C}_{ki} \quad \forall (i,j,k) \in \Omega \end{array}$$

Theorem

- $\mathbf{A}$,$\mathbf{B}$,$\mathbf{C}$ *are* incoherent.
- *number of samples* $|\Omega| > \tilde{O}(nr)$.
- *the locations of samples are drawn i.i.d. from* $[n_1] \times [n_2] \times [n_3]$.

Then, with high probability, the recovery is exact:

$$\hat{\mathbf{A}} = \mathbf{A}, \hat{\mathbf{B}} = \mathbf{B}, \hat{\mathbf{C}} = \mathbf{C}.$$

With noise

$\mathbf{Z}$: stochastic perturbation

$$\hat{T}_{ijk} = T_{ijk} + Z_{ijk} \quad \forall (i, j, k) \in \Omega$$

With noise

$\mathbf{Z}$: stochastic perturbation

$$\hat{T}_{ijk} = T_{ijk} + Z_{ijk} \quad \forall (i,j,k) \in \Omega$$

$$\begin{array}{rl} \text{minimize} & \sqrt{n_3}\left\|\hat{\mathbf{A}}\right\|_* + \sqrt{n_1}\left\|\hat{\mathbf{B}}\right\|_* + \sqrt{n_2}\left\|\hat{\mathbf{C}}\right\|_* \\ \text{subject to} & \displaystyle\sum_{(i,j,k)\in\Omega} \left(\hat{T}_{ijk} - \hat{A}_{ij} - \hat{B}_{jk} - \hat{C}_{ki}\right)^2 \leq \delta^2. \end{array}$$

when noiseless recovery occurs $\implies$ noisy variant is stable.

With noise

$\mathbf{Z}$: stochastic perturbation

$$\hat{T}_{ijk} = T_{ijk} + Z_{ijk} \quad \forall (i,j,k) \in \Omega$$

$$\begin{array}{ll} \text{minimize} & \sqrt{n_3}\left\|\hat{\mathbf{A}}\right\|_* + \sqrt{n_1}\left\|\hat{\mathbf{B}}\right\|_* + \sqrt{n_2}\left\|\hat{\mathbf{C}}\right\|_* \\ \text{subject to} & \sum_{(i,j,k)\in\Omega} \left(\hat{T}_{ijk} - \hat{A}_{ij} - \hat{B}_{jk} - \hat{C}_{ki}\right)^2 \le \delta^2. \end{array}$$

when noiseless recovery occurs $\implies$ noisy variant is stable.

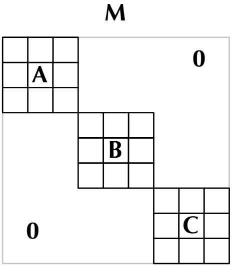
Theorem

- $\|\mathbf{Z}\|_F \le \epsilon$ *(and other conditions for exact recovery)*

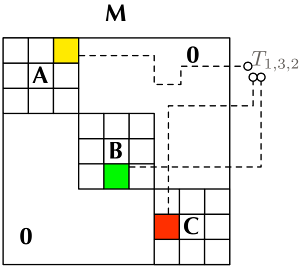
Then, with high probability, the recovery is stable

$$\left\| Pair(\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}) - \mathbf{T} \right\|_F \le \tilde{O}(rn^{3/2}(\delta + \epsilon)).$$

Analysis

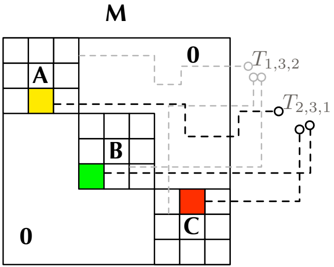


Analysis



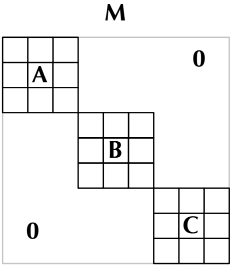
- recover matrix $\mathbf{M}$.
- observations: sums of three entries of $\mathbf{M}$.
- **challenge:** matrix completion with non-orthogonal obs. operators.
  - [Gross 2009] resolved the case with orthogonal obs. operators.
  - ours is the first result on non-orthogonal obs. operators.

Analysis



- recover matrix $\mathbf{M}$.
- observations: sums of three entries of $\mathbf{M}$.
- **challenge:** matrix completion with non-orthogonal obs. operators.
  - [Gross 2009] resolved the case with orthogonal obs. operators.
  - ours is the first result on non-orthogonal obs. operators.

Analysis



$$T_{132} = A_{13} + B_{32} + C_{21}$$

$$T_{231} = A_{23} + B_{31} + C_{21}$$

.....

- recover matrix $\mathbf{M}$.
- observations: sums of three entries of $\mathbf{M}$.
- **challenge:** matrix completion with non-orthogonal obs. operators.
  - [Gross 2009] resolved the case with orthogonal obs. operators.
  - ours is the first result on non-orthogonal obs. operators.

Algorithms and Experiments

optimization algorithms

- one can use SDP to solve trace-norm minimization problems.
  - too slow for matrices larger than 100×100.

Algorithms and Experiments

optimization algorithms

- one can use SDP to solve trace-norm minimization problems.
  - too slow for matrices larger than 100×100.
- we use singular value thresholding (SVT) method to solve a relaxed version.
  - much faster and still accurate.

Algorithms and Experiments

optimization algorithms

- one can use SDP to solve trace-norm minimization problems.
  - too slow for matrices larger than 100×100.
- we use singular value thresholding (SVT) method to solve a relaxed version.
  - much faster and still accurate.

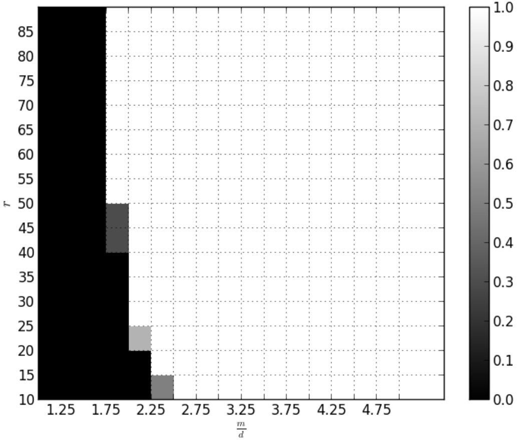
experiments

- exact recovery experiments on synthetic data
- movie recommendation with time information

Experiments: Exact Recovery

empirical recovery probability

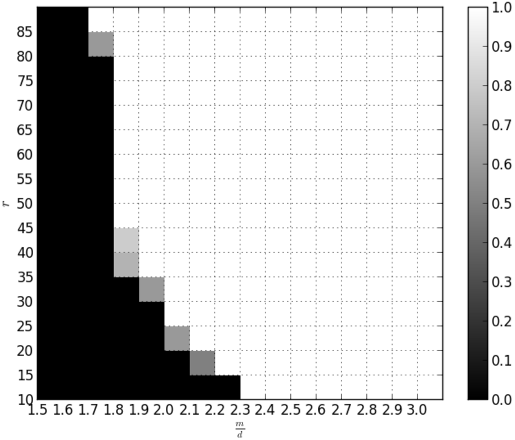
x-axis: number of samples / degree of freedom



Experiments: Exact Recovery

empirical recovery probability (high resolution)

x-axis: number of samples / degree of freedom

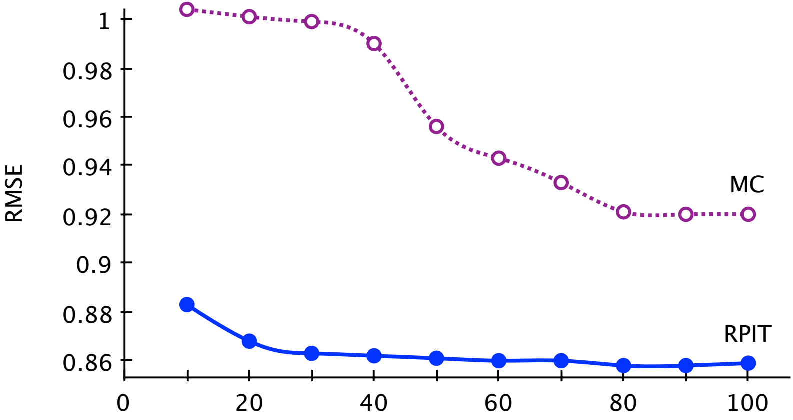


Experiments: Movie Recommendations

- datasets: movielens
  - 1,000,209 timestamped movie ratings
  - 6040 users, 3706 movies, 36 months (0.104% observed)
- baseline: matrix completion
  - ignore time information

Experiments: Movie Recommendations

- datasets: movielens
  - 1,000,209 timestamped movie ratings
  - 6040 users, 3706 movies, 36 months (0.104% observed)
- baseline: matrix completion
  - ignore time information



Summary

- tensor completion
  - recover the missing entries of a tensor.
  - difficult for general tensors.
- pairwise interaction tensor
  - a simpler replacement for general tensors.
- exact recovery for pairwise interaction tensor
  - and stable for noisy observations.
  - via convex programming.

Publications

Combinatorial pure exploration of multi-armed bandits
Shouyuan Chen, Tian Lin, Irwin King, Michael R. Lyu and Wei Chen
To appear in **NIPS 2014**, **Oral presentation**

Contextual combinatorial bandit and its application on diversified recommendation
Lijing Qin, Shouyuan Chen and Xiaoyan Zhu
In **SDM 2014**, **Best Student Paper Award Runner-Up**

Fast relative-error approximation for ridge regression
Shouyuan Chen, Yang Liu, Michael R. Lyu, Irwin King and Shengyu Zhang
Technical report 2014

Exact and stable recovery of pairwise interaction tensors
Shouyuan Chen, Michael R. Lyu, Irwin King and Zenglin Xu
In **NIPS 2013**, **Spotlight**

Thank you!

Experiments of Linear Combinatorial Bandits