Learning with Limited Samples

Shouyuan Chen
The Chinese University of Hong Kong
October 29, 2014
samples are sometimes very expensive.

- decision making / prediction using a limited number of samples.
samples are sometimes very expensive.

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

clinical trials

...
samples are sometimes very expensive.

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

clinical trials
samples are sometimes very expensive.

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

**clinical trials**

<table>
<thead>
<tr>
<th>Users</th>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

**movie recommendation**

<table>
<thead>
<tr>
<th>Users</th>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>
samples are sometimes very expensive.

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

### Clinical Trials

- Users: 1, 2, 3, 4
- Movies: 1, 2, 3, 4, 5

### Movie Recommendation

- Users: 1, 2, 3, 4, 5
- Movies: 1, 2, 3, 4, 5

Users 1, 2, 3, 4 like movies 1, 2, 3, 4, respectively. User 5's preferences are unknown.
multi-armed bandits

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

tensor completion

compressed sensing  matrix completion  tensor completion
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

arm
Single-armed bandit
Single-armed bandit

 sampled independently from an unknown distribution (reward distribution)
Multi-armed bandit

$n$ arms
Multi-armed bandit

$n$ arms

play

reward

MAB player

goal: maximize the cumulative reward

take all rewards

in the end...

exploitation v.s. exploration

play

reward

n  arms

pure exploration 
player

pure exploration

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

(1) forfeit all rewards

in the end...

(2) output 1 arm
Multi-armed bandit

$n$ arms

MAB player

reward

MAB player

goal: maximize the cumulative reward

in the end...

exploitation v.s. exploration

pure exploration

player

find the single best arm (largest expected reward)

(1) forfeit all rewards

(2) output 1 arm

7
**Multi-armed bandit**

A Multi-armed bandit (MAB) setup involves a player choosing among multiple options (alias 'arms') with the goal of maximizing cumulative reward. The scenario typically involves:

- **$n$ arms**: Represented as a set of options or treatments.
- **Play Reward**: Interactions with each arm return a reward.
- **Objective**: Maximize cumulative reward.

### Pure Exploration
- **Goal**: Find the single best arm (biggest expected reward).
  - **(1) Forfeit all rewards in the end**.
  - **(2) Output 1 arm**.

### Exploration vs. Exploitation
- **Balance** between exploring new options and exploiting known high-reward options.

[Diagram showing a set of arms and play symbols, illustrating the concept of MAB player.]
Multi-armed bandit

$n$ arms

MAB player

Goal: maximize the cumulative reward
Take all rewards in the end...

Exploitation vs. exploration

Pure exploration

Goal: find the single best arm (largest expected reward)
(1) forfeit all rewards in the end...
(2) output 1 arm
Multi-armed bandit

$n$ arms

MAB player

play → reward

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

goal: find the single best arm (largest expected reward)
Multi-armed bandit

$n$ arms

![Diagram of MAB player with n arms, play and reward cycles]

**goal**: maximize the cumulative reward
	take all rewards

in the end...
	output 1 arm

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

exploitation v.s. exploration
Multi-armed bandit

$n$ arms

play \rightarrow reward

MAB player

in the end...

take all rewards

goal: maximize the cumulative reward

exploitation v.s. exploration

$n$ arms

pure exploration

play \rightarrow reward

pure exploration player

(1) forfeit all rewards

(2) output 1 arm

goal: find the single best arm (largest expected reward)
Multi-armed bandit

\( n \) arms

play \( \Rightarrow \) reward

MAB player

in the end...

take all rewards

goal: maximize the cumulative reward

exploitation v.s. exploration

\( n \) arms

pure exploration

play \( \Rightarrow \) reward

pure exploration player

in the end...

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

goal: find the single best arm (largest expected reward)
Combinatorial Pure Exploration (CPE)

- play one arm at each round
- find the optimal set of arms $M_\ast \in \mathcal{M}$
  - 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 (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.

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

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

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

- **matching**

  ![Diagram](attachment:image.png)

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

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

- matching

  ![Matching Diagram]

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

- spanning trees and paths

  ![Spanning Trees Diagram]

  **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.

- 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.
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 $t$

$\bar{w}_t(i)$

0

1

$\text{rad}_t(i)$

empirical mean: $w_t(i)$

confidence interval: $\text{rad}_t(i)$ (proportional to $1/\sqrt{n_t(i)}$)

maximization oracle: $\text{Oracle}(v) = \max_{M \geq M} \sum_{i=1}^{M} v(i)$ for any $n$-dimensional vector $v$
CLUCB: Fixed confidence algorithm

notations

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

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

notations
CLUCB: Fixed confidence algorithm

all arms

Step 1

Oracle(\(\bar{\omega}_t\))

\(\bar{M}_t\)

maintain: for all \(i\) and \(t\)

\[\text{rad}_t(i)\]

\(0\)

\(\bar{\omega}_t(i)\)

\(1\)
CLUCB: Fixed confidence algorithm

**Step 1**
- \( \bar{M}_t \)
- Oracle(\( \bar{w}_t \))

**Step 2**
- \(-rad_t(i)\) and \(+rad_t(i)\)
- Oracle(\( \tilde{w}_t \))

\( \tilde{M}_t \)

**Maintain:** for all \( i \) and \( t \)

\[ \tilde{w}_t(i) = \bar{w}_t(i) \pm rad_t(i) \]
CLUCB: Fixed confidence algorithm

Maintain: for all $i$ and $t$

$\bar{w}_t(i)$

Step 1

Oracle($\bar{w}_t$)

$\bar{M}_t$

Step 2

$-\text{rad}_t(i)$ $+$ $\text{rad}_t(i)$

Oracle($\tilde{w}_t$)

$\tilde{M}_t$

$\tilde{w}_t(i) = \bar{w}_t(i) \pm \text{rad}_t(i)$

If:

$\bar{M}_t = \tilde{M}_t$

Then:

Stop and output $\bar{M}_t$
CLUCB: Fixed confidence algorithm

Step 1

Oracle($\bar{w}_t$)

$M_t$

Step 3

$\text{rad}_t$

1 2 3 4

maintain: for all $i$ and $t$

$\bar{w}_t(i)$

$0 \leq \bar{w}_t(i) \leq 1$

all arms
CLUCB: Fixed confidence algorithm

Step 1

Oracle($\tilde{w}_t$)

$\overline{M}_t$

Step 3

rad$_t$

Play arm 3!

Go to t+1 round.

maintain: for all i and t

rad$_t$(i)

$\tilde{w}_t$(i)

Q-learning

all arms
Our sample complexity bound depends on two quantities.

- \( H \): only depends on expected rewards
- \( \text{width}(M) \): only depends on the structure of \( M \)
Our sample complexity bound depends on two quantities.

- \( H \): only depends on expected rewards
- \( \text{width}(M) \): only depends on the structure of \( M \)

**Theorem**

*With probability at least \( 1 - \delta \), CLUCB algorithm:*

1. *outputs the optimal set* \( M_\ast \triangleq \arg\max_{M \in M} w(M) \).
2. *uses at most* \( O(\text{width}(M)^2 H \log(nH/\delta)) \) *rounds.*
Sample complexity (1): $H$

- $\Delta_e$: gap of arm $e \in [n]$

$$\Delta_e = \begin{cases} 
  w(M_*) - \max_{M \in \mathcal{M} : e \in M} w(M) & \text{if } e \not\in M_*, \\
  w(M_*) - \max_{M \in \mathcal{M} : e \not\in M} w(M) & \text{if } e \in M_* 
\end{cases}$$

- stability of the optimality of $M_*$ regarding to arm $e$.

- $H = \sum_{e \in [n]} \Delta_e^{-2}$
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.
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:

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

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 $e \in (M \setminus M')$, there exists an exchange set $(b_+, b_-) \in \mathcal{B}$ which satisfies five constraints: (a) $e \in b_-$, (b) $b_+ \subseteq M' \setminus M$, (c) $b_- \subseteq M \setminus M'$, (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 \( \text{Exchange}(\mathcal{M}) \) is the family of all possible exchange classes for \( \mathcal{M} \).

Width: examples

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

**Theorem**

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

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

**Theorem**

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

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

Plug in the widths of examples

**Corollary (Sample Complexity of Examples)**

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

An algorithm $\mathcal{A}$ is a $\delta$-correct algorithm, if $\mathcal{A}$’s probability of error is at most $\delta$ for any expected rewards.
Lower bound

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

Theorem (Problem dependent lower bound)

*Given any expected rewards, any $\delta$-correct algorithm must use at least $\Omega(H \log(1/\delta))$ rounds.*
An algorithm $\mathbb{A}$ is a $\delta$-correct algorithm, if $\mathbb{A}$’s probability of error is at most $\delta$ for any expected rewards.

**Theorem (Problem dependent lower bound)**

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

**Remarks:**

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

phase 1

In each phase (n phases in total):

1. Arm is accepted or rejected.
2. Active arms are sampled for a same number of times.

accepted: include in the output
rejected: exclude from the output
active: neither accepted nor rejected.
require more samples
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
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.

**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.

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

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

problem: which arm to accept or reject?

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

\[
\tilde{\Delta}_e = \begin{cases} 
\bar{w}_t(\bar{M}_t) - \max_{M \in \mathcal{M}_t : e \in M} \bar{w}_t(M) & \text{if } e \not\in \bar{M}_t, \\
\bar{w}_t(\bar{M}_t) - \max_{M \in \mathcal{M}_t : e \not\in M} \bar{w}_t(M) & \text{if } e \in \bar{M}_t
\end{cases}
\]

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

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

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

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

\[
\Delta_e = \begin{cases} 
 w(M_*) - \max_{M \in \mathcal{M}: e \in M} w(M) & \text{if } e \notin M_*, \\
 w(M_*) - \max_{M \in \mathcal{M}: e \notin M} w(M) & \text{if } e \in M_*
\end{cases}
\]
Theorem (Probability of error of CSAR)

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

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

and uses at most $T$ rounds.
Theorem (Probability of error of CSAR)

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

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

and uses at most $T$ rounds.

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

The number of arms (movies) 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 $v_i \in \mathbb{R}^d$

An unknown vector $u \in \mathbb{R}^d$

Playing arm $i$ gives a random reward $r_i = u^T v_i + \epsilon$

$\epsilon$ is a zero-mean r.v.

Algorithms with $\tilde{O}(pT)$ regret [APS11].
Linear bandits

The number of arms (movies) 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 $v_i \in \mathbb{R}^d$.

An unknown vector $u \in \mathbb{R}^d$.

Playing arm $i$ gives a random reward $r_i = u^T v_i + \epsilon$.

$\epsilon$ is a zero-mean r.v.

Algorithms with $\tilde{O}(p^T)$ regret [APS11].
Linear bandits

The number of arms (movies) 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 $v_i \in \mathbb{R}^d$
- An unknown vector $u \in \mathbb{R}^d$
- Playing arm $i$ gives a random reward $r_i = u^T v_i + \epsilon$
- $\epsilon$ is a zero-mean r.v.
- Algorithms with $\tilde{O}(pT)$ regret \cite{APS11}.
Linear bandits

A recommender system aims to recommend items to a user. The challenge is selecting items that the user will like when the number of items is very large. One solution is to make more assumptions about the rewards (ratings) and use linear bandits. Each arm $i$ has a feature vector $v_i \in \mathbb{R}^d$. Playing arm $i$ gives a random reward $r_i = u^T v_i + \epsilon$, where $\epsilon$ is a zero-mean r.v. Algorithms with $\tilde{O}(\sqrt{pT})$ regret are possible, as shown in [APS11].
Linear bandits

User plays n arms, get rewards, and rate them. The recommender system learns from the ratings and updates its predictions.

The number of arms (movies) is very large. Many arms will never be played.

Solution: More assumptions on the rewards (ratings).

Linear bandits: Each arm $i$ has a feature vector $v_i \in \mathbb{R}^d$.

An unknown vector $u \in \mathbb{R}^d$.

Playing arm $i$ gives a random reward $r_i = u^T v_i + \epsilon$.

$\epsilon$ is a zero-mean random variable.

Algorithms with $\sim O(p^T)$ regret [APS11].
Linear bandits

The number of arms (movies) is very large.

\[ \text{Challenge:} \text{ many arms will never be played.} \]

\[ \text{Solution:} \text{ more assumptions on the rewards (ratings)} \]

\[ \text{Linear bandits} \]

Each arm \( i \) has a feature vector \( v_i \in \mathbb{R}^d \).

\[ \text{Playing arm } i \text{ gives a random reward} \ r_i = u^T v_i + \epsilon \]

\( \epsilon \) is a zero-mean r.v.

Algorithms with \( \sim O(pT) \) regret [APS11].
Linear bandits

The number of arms (movies) 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 $v_i \in \mathbb{R}^d$.
- An unknown vector $u \in \mathbb{R}^d$.
- Playing arm $i$ gives a random reward $r_i = u^T v_i + \epsilon$.
- $\epsilon$ is a zero-mean r.v.
- Algorithms with $\tilde{O}(pT)$ regret [APS11].
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 $r_i = \mathbf{u}^T \mathbf{v}_i + \epsilon$
    - $\epsilon$ is a zero-mean r.v.
  - algorithms with $\tilde{O}(\sqrt{T})$ regret \cite{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

- 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 $f_{\mathbf{r}^{(t)}, \mathbf{v}}(S_t)$. 

linear combinatorial bandits
Reward function

we allow a broad class of $f_r, v(S_t)$ that satisfy

- **monotone** and **Lipschitz continuous** in terms of $r$.
- an $\alpha$-maximization oracle
  - approximation ratio $\alpha \in (0, 1]$.
we allow a broad class of $f_r, V(S_t)$ that satisfy

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

a reward function for movie recommendation

$$f_r, V(S) = \sum_{i \in S} r_i + \lambda g(\{v_i | i \in S\}).$$

- sum of ratings
- 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 $\alpha$-regret over $T$ rounds is $\tilde{O}(\sqrt{T})$.

- $\alpha$-regret: $\alpha \text{OPT}(T) - \sum_{i=1}^{T} f_{r^{(t)}}(S_t)$
- $\text{OPT}(T)$: the largest possible reward from $T$ rounds
Ridge regression

Ridge regression problem

\[
\min_x ||Ax - b||_2^2 + \lambda ||x||_2^2.
\]

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

optimal solution

\[
x_* = A^T(AA^T + \lambda I_n)^{-1}b.
\]

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

challenge

\[ n \gg p \gg 1 \]
Fast relative-error approximation

oblivious subspace embedding (OSE)

\[ \tilde{x} = A S^T (A S^T) y + A S^T y b \]

Theorem

Given \( \epsilon > 0 \), there exists a way to construct \( S \) such that, with high probability, \( \|\tilde{x} - x\|_2 \leq \epsilon \|x\|_2 \) and the algorithm runs in \( O(\text{nnz}(A) + n^3/\epsilon) \) time.
Fast relative-error approximation

oblivious subspace embedding (OSE)

our OSE based solution

\[ \tilde{x} = A^T(AS^T)^\dagger T (\lambda (AS^T)^\dagger T + AS^T)^\dagger b. \]
Fast relative-error approximation

oblivious subspace embedding (OSE)

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

Theorem

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

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

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

baselines

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

---

**relative error**

![Relative Error Graph](image)

**speedup factors**

![Speedup Factor Graph](image)
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

<table>
<thead>
<tr>
<th>users</th>
<th>movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 2 5 8</td>
<td></td>
</tr>
<tr>
<td>4 2 3 5 5</td>
<td></td>
</tr>
<tr>
<td>2 1 3 4 2</td>
<td></td>
</tr>
<tr>
<td>5 5 6 4 2</td>
<td></td>
</tr>
<tr>
<td>7 7 4 2 3</td>
<td></td>
</tr>
</tbody>
</table>
Matrix completion

<table>
<thead>
<tr>
<th>movies</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

- **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: 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!
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

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

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

definition

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

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

definition

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

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

**Definition**

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

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

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

Recovery via trace-norm minimization

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

minimize $\sqrt{n_3} \| \hat{A} \|_* + \sqrt{n_1} \| \hat{B} \|_* + \sqrt{n_2} \| \hat{C} \|_*$

subject to $T_{ijk} = \hat{A}_{ij} + \hat{B}_{jk} + \hat{C}_{ki}$ $\forall (i, j, k) \in \Omega$
Exact recovery

\[
\text{minimize} \quad \sqrt{n_3} \| \hat{A} \|_* + \sqrt{n_1} \| \hat{B} \|_* + \sqrt{n_2} \| \hat{C} \|_* \\
\text{subject to} \quad T_{ijk} = \hat{A}_{ij} + \hat{B}_{jk} + \hat{C}_{ki} \quad \forall (i, j, k) \in \Omega
\]

Theorem

- **A, B, 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**:

\[\hat{A} = A, \hat{B} = B, \hat{C} = C.\]
With noise

$Z$: stochastic perturbation

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

\(Z\): stochastic perturbation

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

minimize  \(\sqrt{n_3} \| \hat{A} \|_* + \sqrt{n_1} \| \hat{B} \|_* + \sqrt{n_2} \| \hat{C} \|_*\)

subject to  \(\sum_{(i,j,k) \in \Omega} (\hat{T}_{ijk} - \hat{A}_{ij} - \hat{B}_{jk} - \hat{C}_{ki})^2 \leq \delta^2\).

when noiseless recovery occurs  \(\implies\) noisy variant is stable.
With noise

$\mathbf{Z}$: stochastic perturbation

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

minimize $$\sqrt{n_3} \left\| \hat{\mathbf{A}} \right\|_* + \sqrt{n_1} \left\| \hat{\mathbf{B}} \right\|_* + \sqrt{n_2} \left\| \hat{\mathbf{C}} \right\|_*$$

subject to $$\sum_{(i,j,k) \in \Omega} \left( (\hat{T}_{ijk} - \hat{A}_{ij} - \hat{B}_{jk} - \hat{C}_{ki})^2 \right) \leq \delta^2.$$  

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

Theorem

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

Then, with high probability, the recovery is stable

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

\[ M = \begin{bmatrix} A & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & C \end{bmatrix} \]

Observations: sums of three entries of \( M \).

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.
• 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.
• 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.
Analyze matrix $M$.

- **Observations**: sums of three entries of $M$.
- **Challenge**: matrix completion with non-orthogonal observation operators. 
  - [Gross 2009] resolved the case with orthogonal observation operators.
  - Ours is the first result on non-orthogonal observation operators.

\[
T_{132} = A_{13} + B_{32} + C_{21}
\]
\[
T_{231} = A_{23} + B_{31} + C_{21}
\]

\[\ldots\]
optimization algorithms

- one can use SDP to solve trace-norm minimization problems.
  - too slow for matrices larger than $100 \times 100$. 
Algorithms and Experiments

**optimization algorithms**

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

![Graph](image-url)
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.
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

![Graphs showing precision over rounds for different k values (k=5, k=10, k=15, k=20).]