Multi-Armed Bandit and Its Application in **Recommender Systems**

Team: P21

Fan Jue	A0221578B	E0559110
Nadia Victoria Aritonang	A0314698N	E1505949
Reiner Anggriawan Jasin	A0314502W	E1503344
Tian Xiao	A0220592L	E0555784

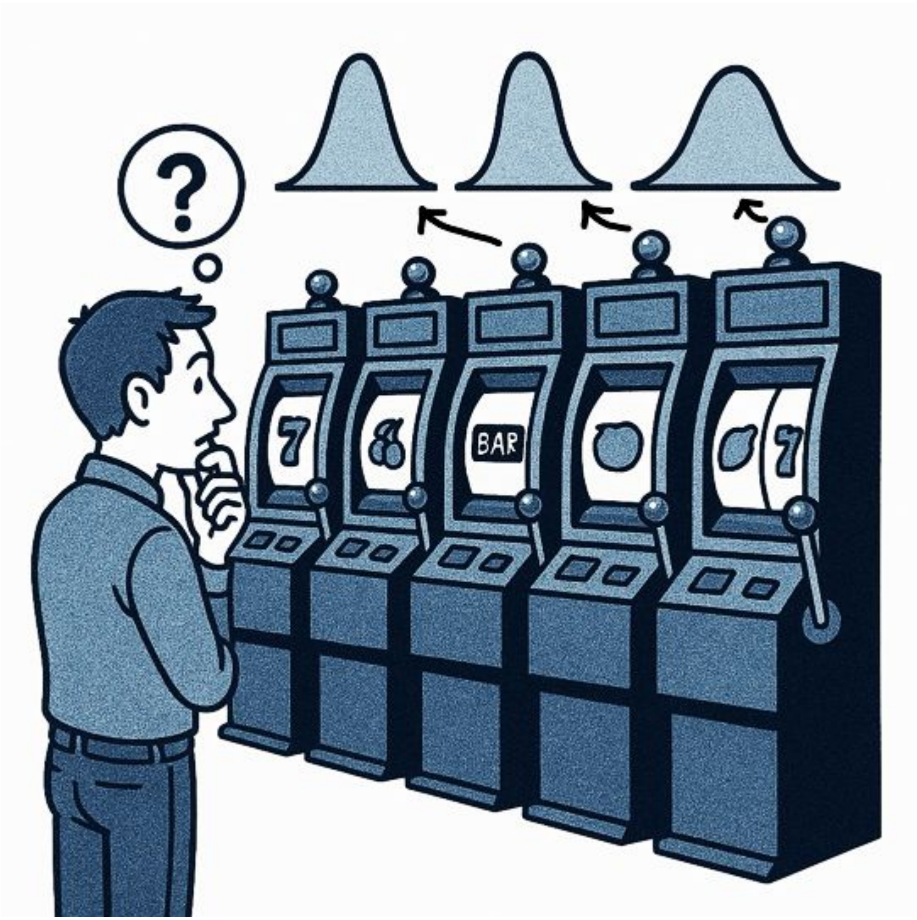
AY2024/25 Sem2
CS4246/CS5446

NUS | Computing
National University of Singapore

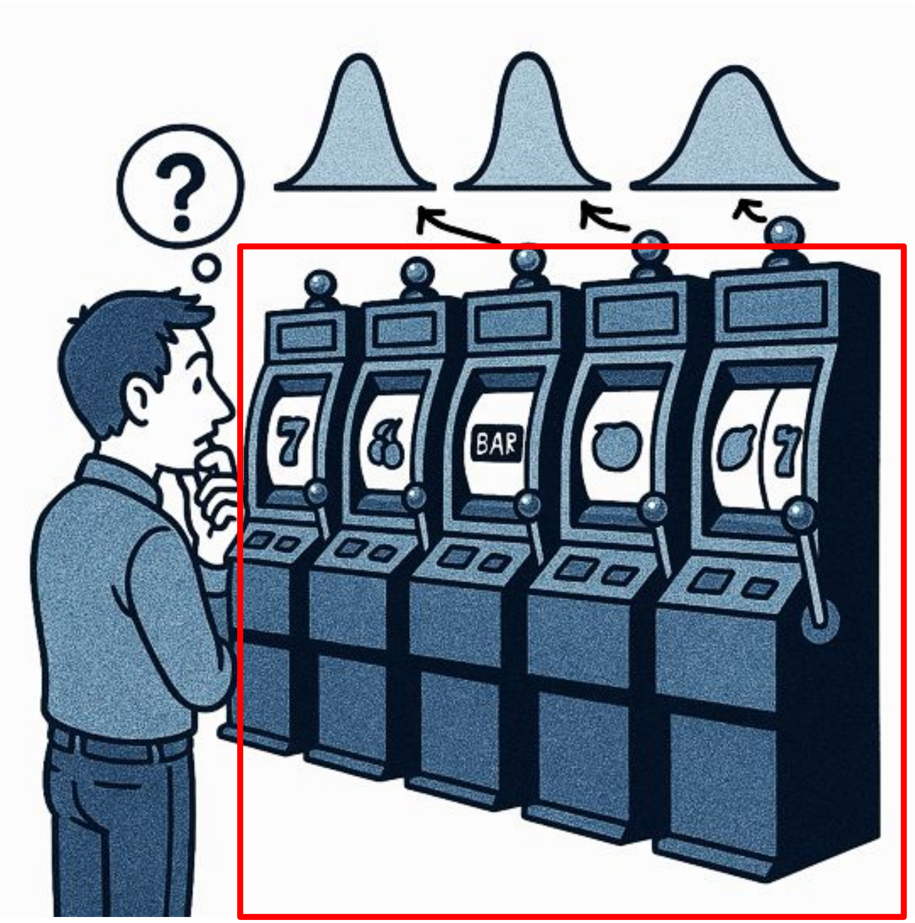
Overview

- Stochastic Bandits
- Contextual Bandits
- Implementation
- Evaluation

Motivation

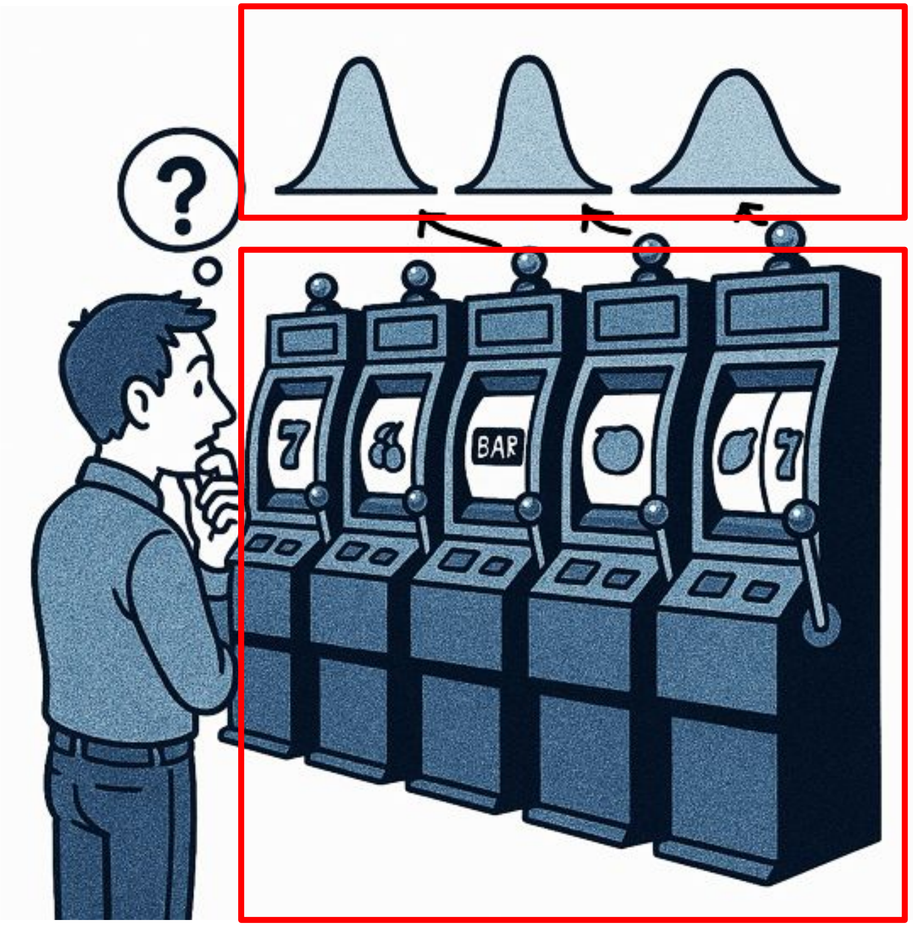


Motivation



A total of *K* slot machines.

Motivation

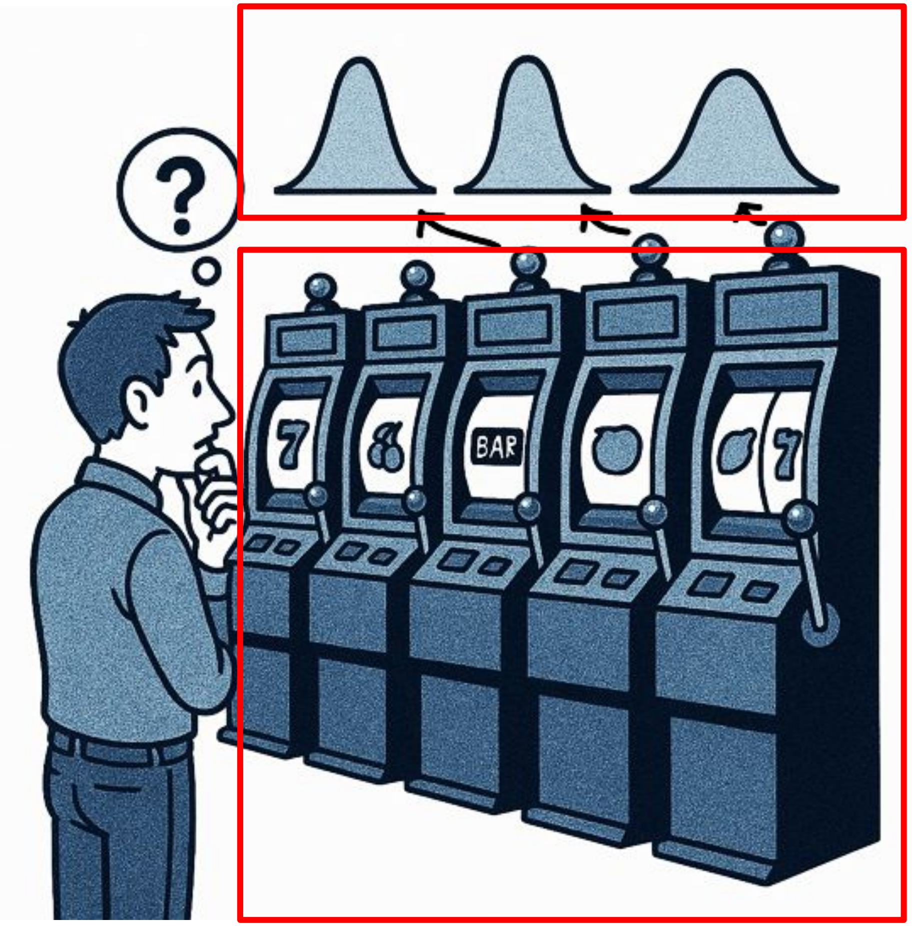


Each machine gives unknown, random rewards.

A total of *K* slot machines.

Motivation

I have *T* tokens.
How can I maximize
my total reward?

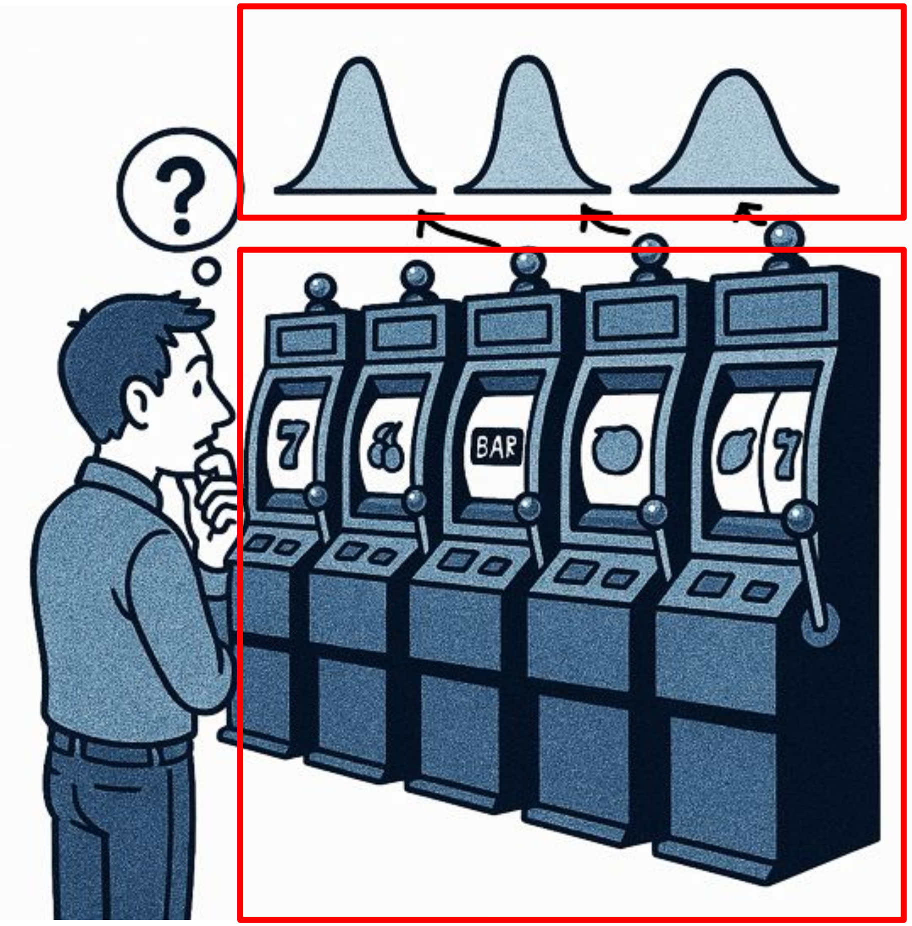


Each machine
gives unknown,
random rewards.

A total of *K*
slot machines.

Stochastic Bandit: $B = (A, R)$

I have *T* tokens.
How can I maximize
my total reward?

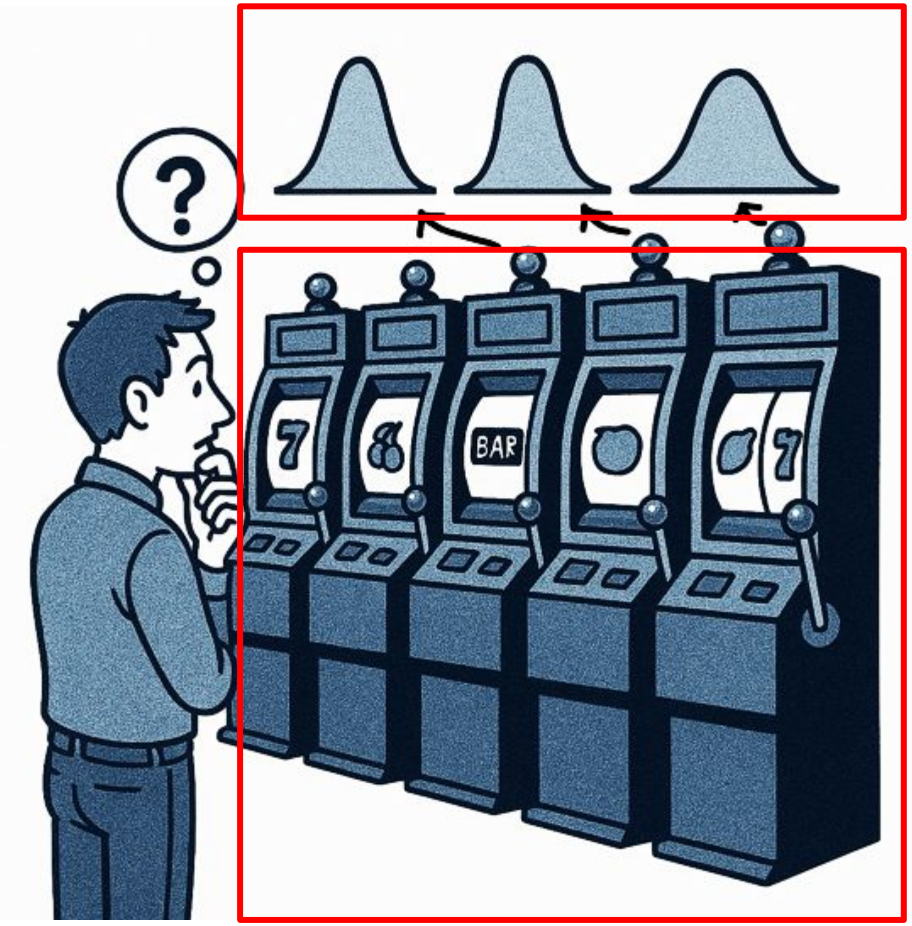


Each machine gives unknown, random rewards.

A total of *K* ~~slot machines~~ actions $A = (a_1, \cdots, a_K)$.

Stochastic Bandit: $B = (A, R)$

I have *T* tokens.
How can I maximize
my total reward?



action a_k
Each ~~machine~~
gives unknown,
random rewards.

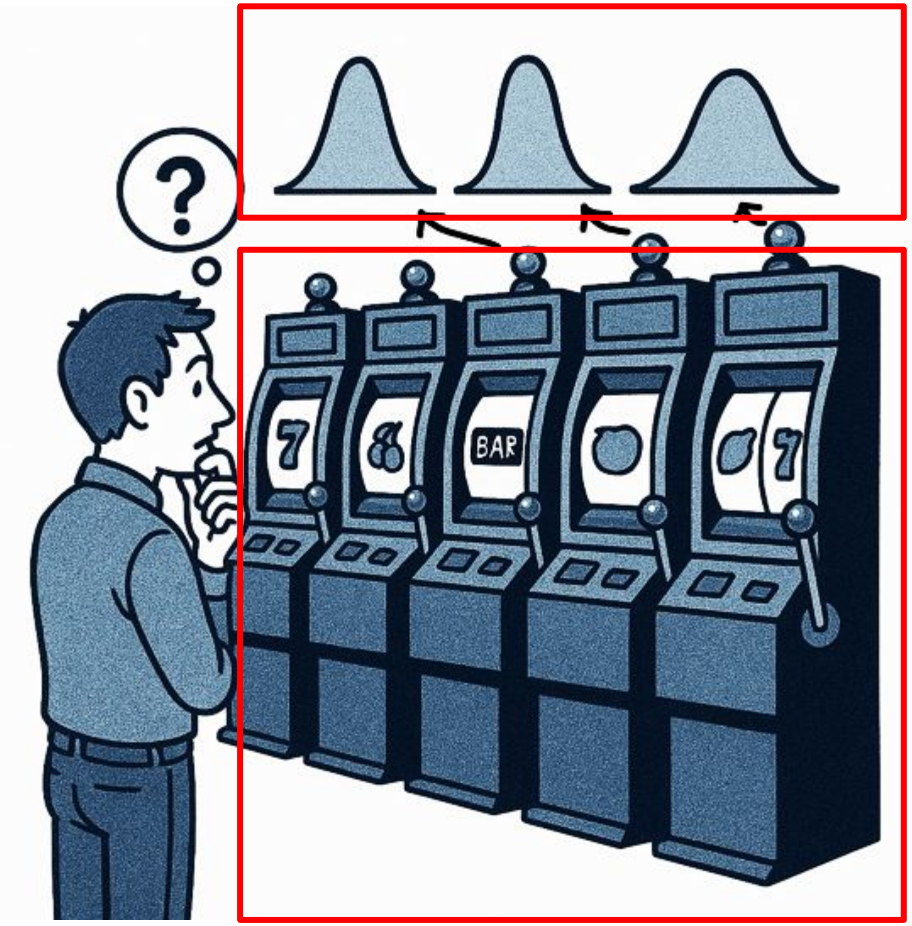
$$R_k \sim p_{R_k}(\cdot)$$

A total of *K*
~~slot machines~~
actions $A =$
$(a_1, \cdots, a_K)$.

Stochastic Bandit: $B = (A, R)$

rounds
I have *T* ~~tokens~~.
How can I maximize
my <u>total reward</u>?

$$\sum_{t=1}^{T} r_{a_t}$$



action a_k
Each ~~machine~~
gives <u>unknown</u>,
<u>random</u> rewards.

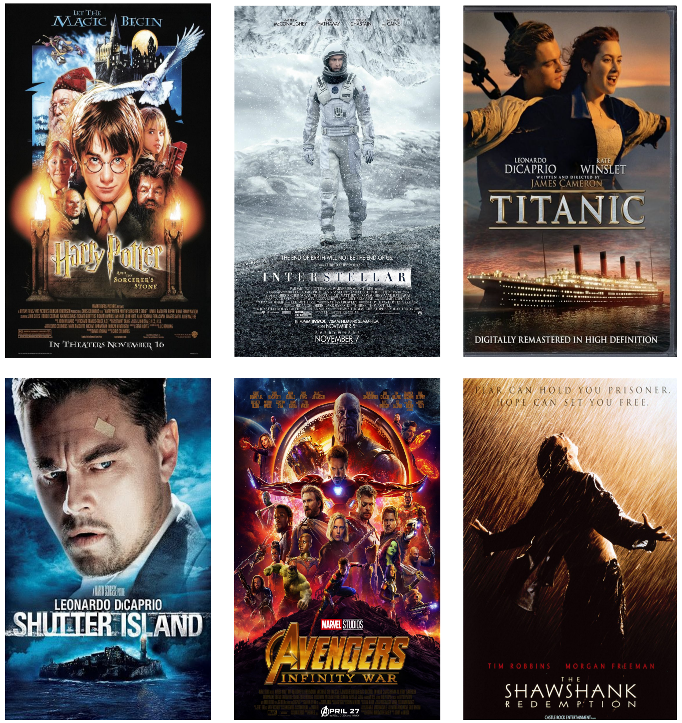
$$R_k \sim p_{R_k}(\cdot)$$

A total of *K*
~~slot machines~~
actions $A =$
$(a_1, \cdots, a_K)$.

Application: Movie Recommendation

Movie recommender

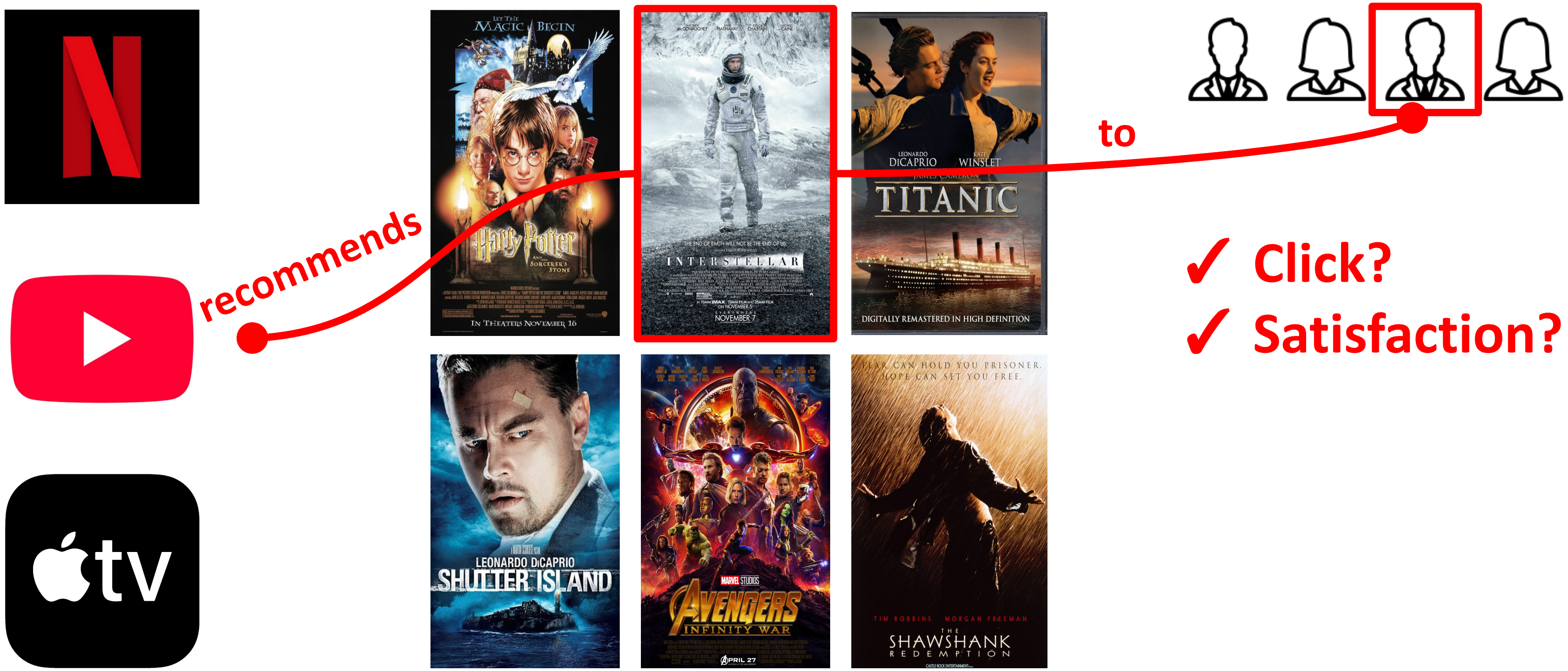
Actions $a_1, \cdots, a_K$



Users

Application: Movie Recommendation

Movie recommender **Actions $a_1, \cdots, a_K$** **Users**



recommends

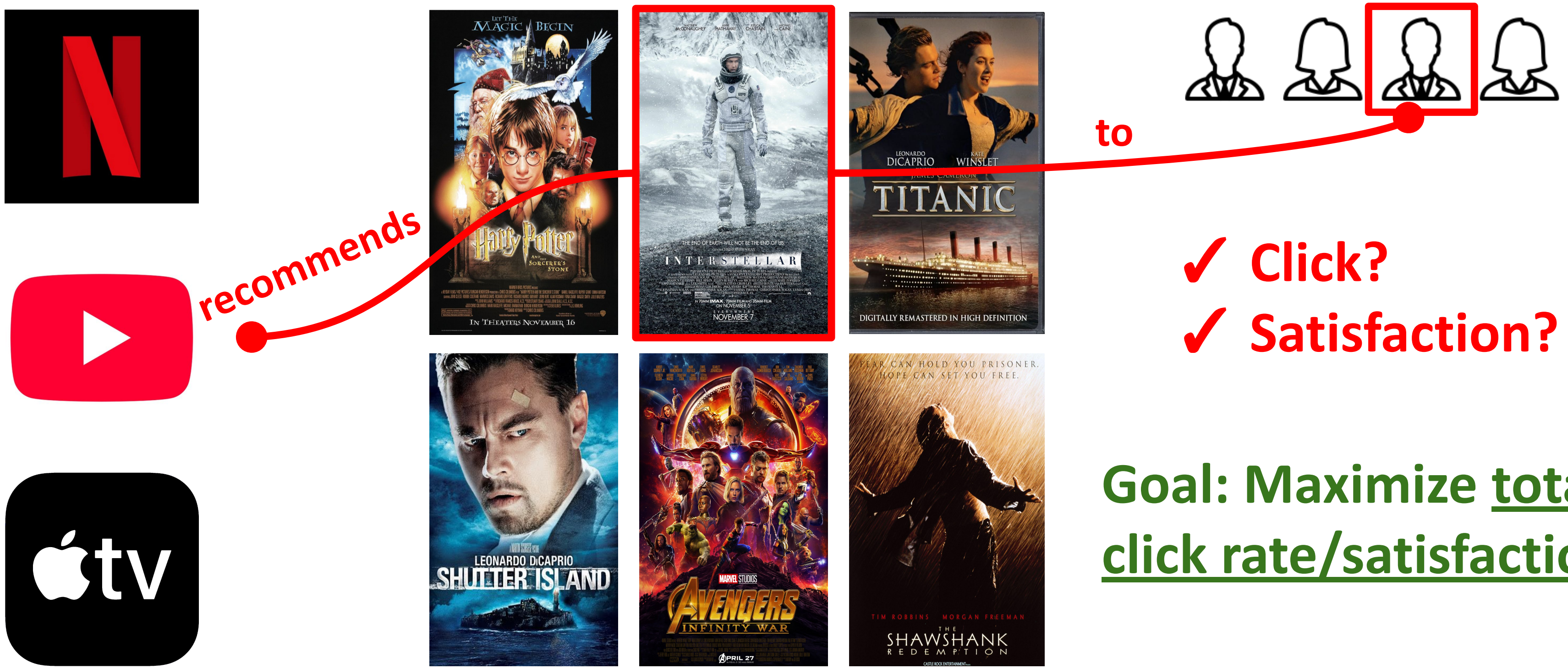
to

✓ **Click?**

✓ **Satisfaction?**

Application: Movie Recommendation

Movie recommender **Actions $a_1, \cdots, a_K$** **Users**



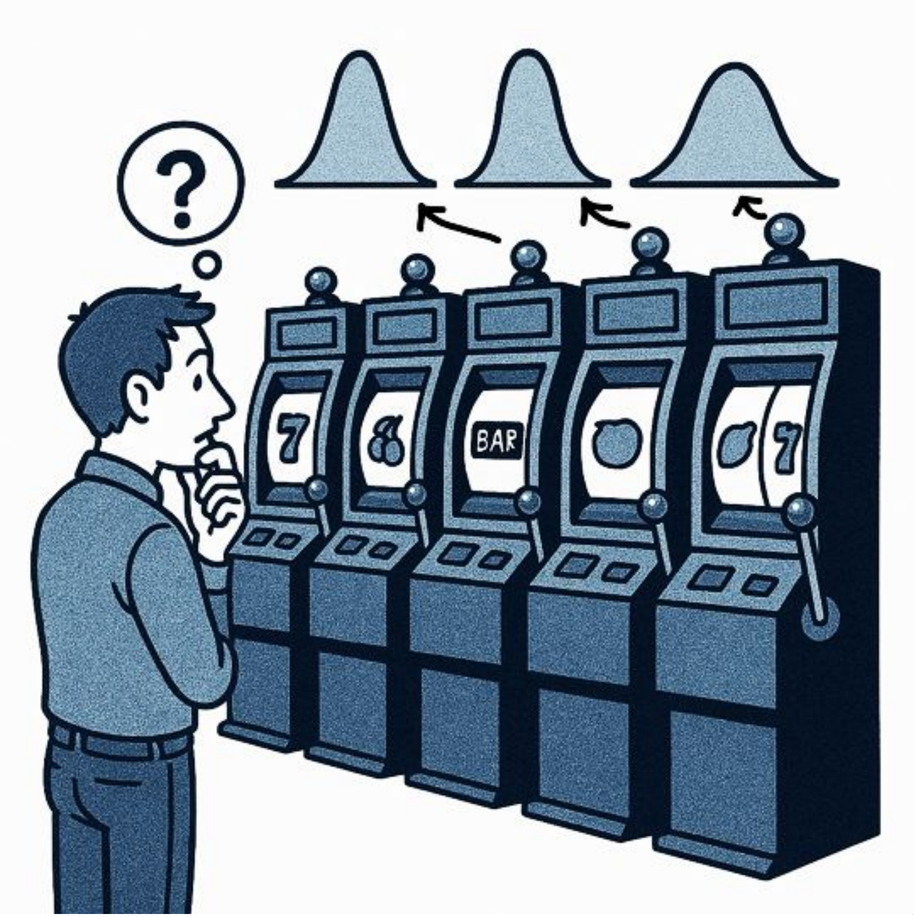
✓ Click?

✓ Satisfaction?

Goal: Maximize <u>total click rate/satisfaction</u>.

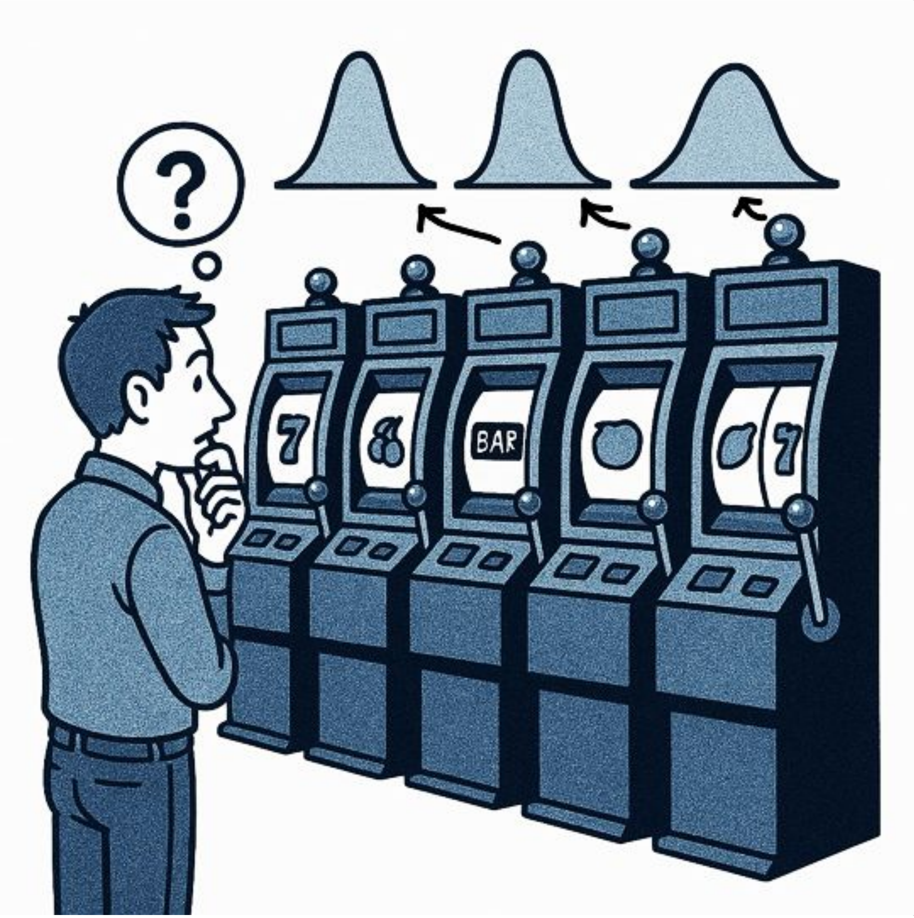
Algorithm

- Assume I know the expected reward $\bar{r}_k$ given by each action, then the best strategy is **to always choose the best action a^* with the highest $\bar{r}^*$.**



Algorithm

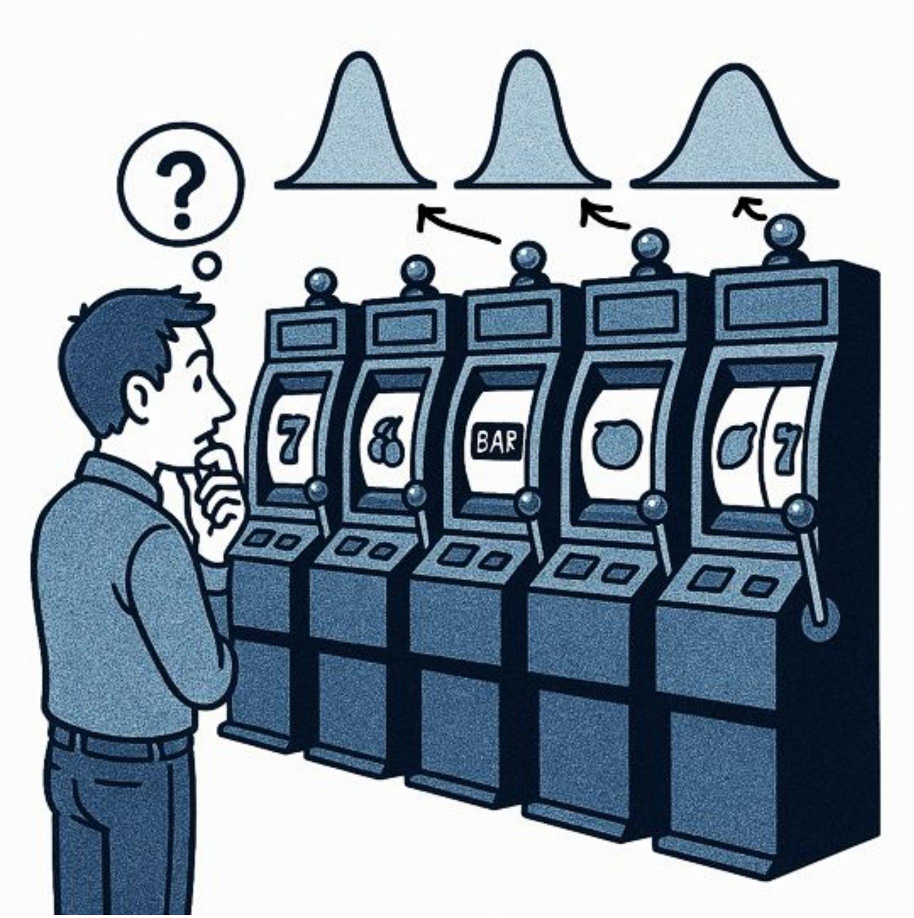
- Assume I know the expected reward $\bar{r}_k$ given by each action, then the best strategy is **to always choose the best action a^* with the highest $\bar{r}^*$.**
  - But I don’t know…



Algorithm

- Assume I know the expected reward $\bar{r}_k$ given by each action, then the best strategy is **to always choose the best action a^* with the highest $\bar{r}^*$.**
    - But I don't know…
- We use ***(cumulative) regret*** to measure how good a bandit algorithm is:

$$\rho_T = \mathbb{E}\left[\sum_{t=1}^{T} R_{a^*} - \sum_{t=1}^{T} R_{a_t}\right]$$
$$= \sum_{t=1}^{T} \left(\bar{r}^* - \bar{r}_{a_t}\right).$$

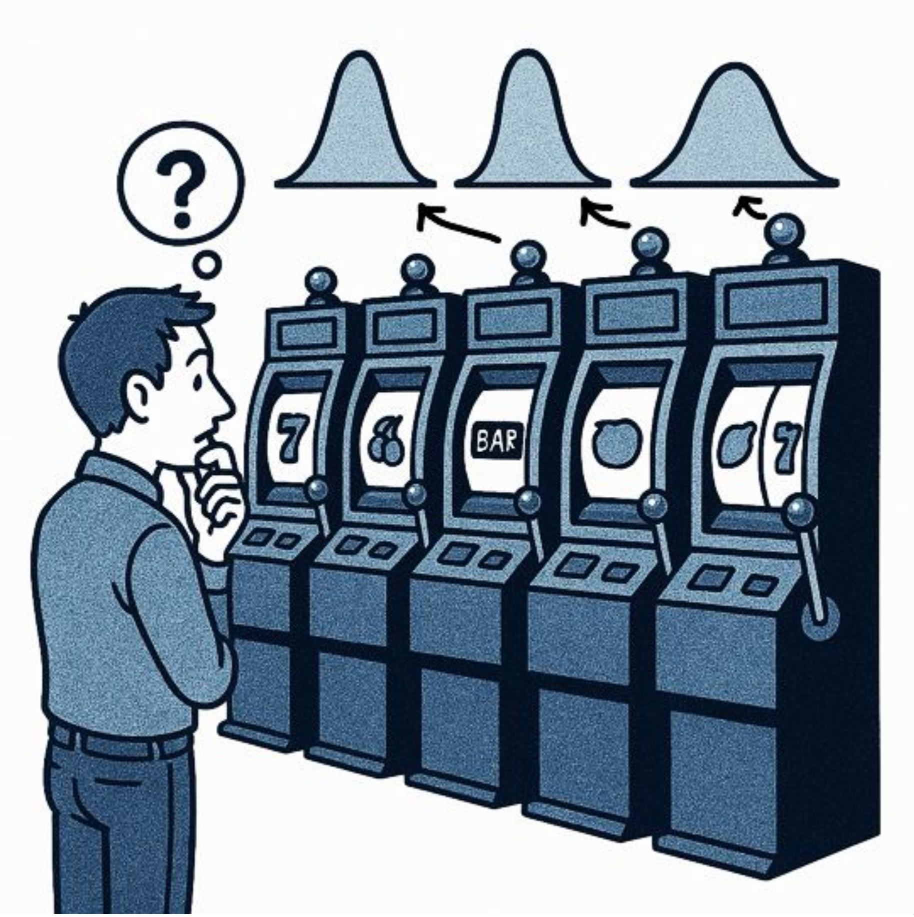


Algorithm

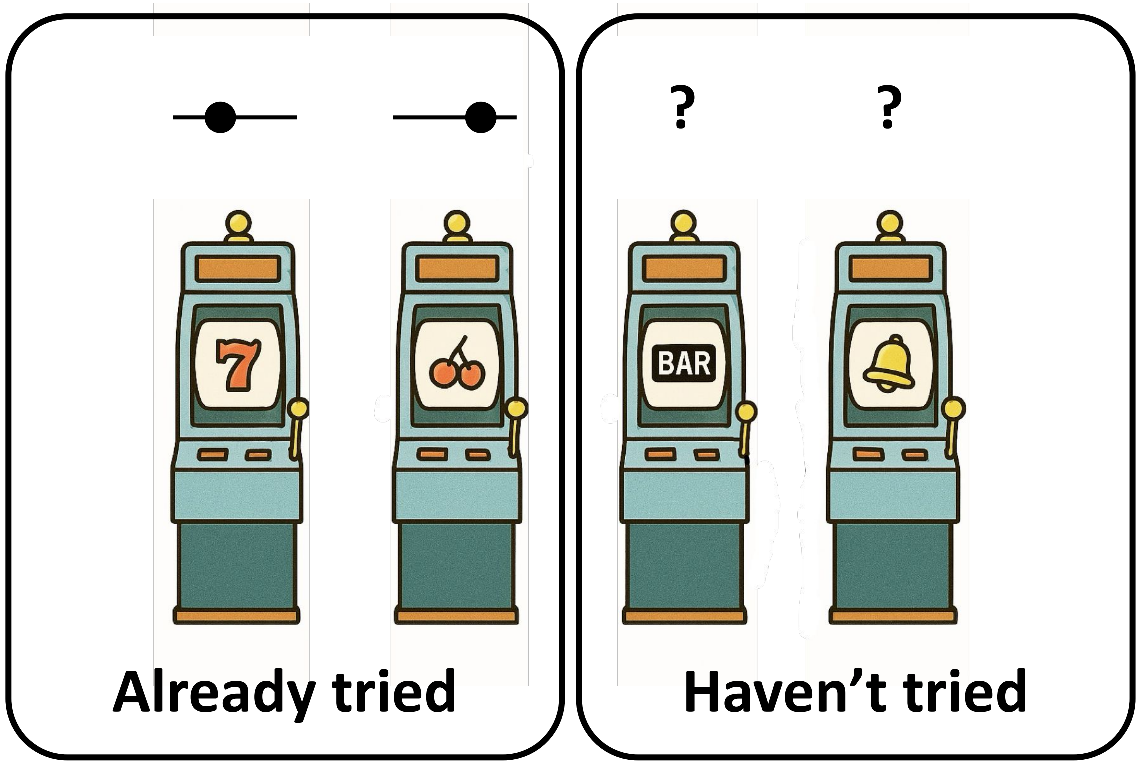
Good if we can bound the regret!

- Assume I know the expected reward $\bar{r}_k$ given by each action, then the best strategy is **to always choose the best action a^* with the highest $\bar{r}^*$.**
  - But I don't know…
- We use ***(cumulative) regret*** to measure how good a bandit algorithm is:

$$\rho_T = \mathbb{E}\left[\sum_{t=1}^{T} R_{a^*} - \sum_{t=1}^{T} R_{a_t}\right]$$
$$= \sum_{t=1}^{T} \left(\bar{r}^* - \bar{r}_{a_t}\right).$$



Algorithm: ϵ-Greedy

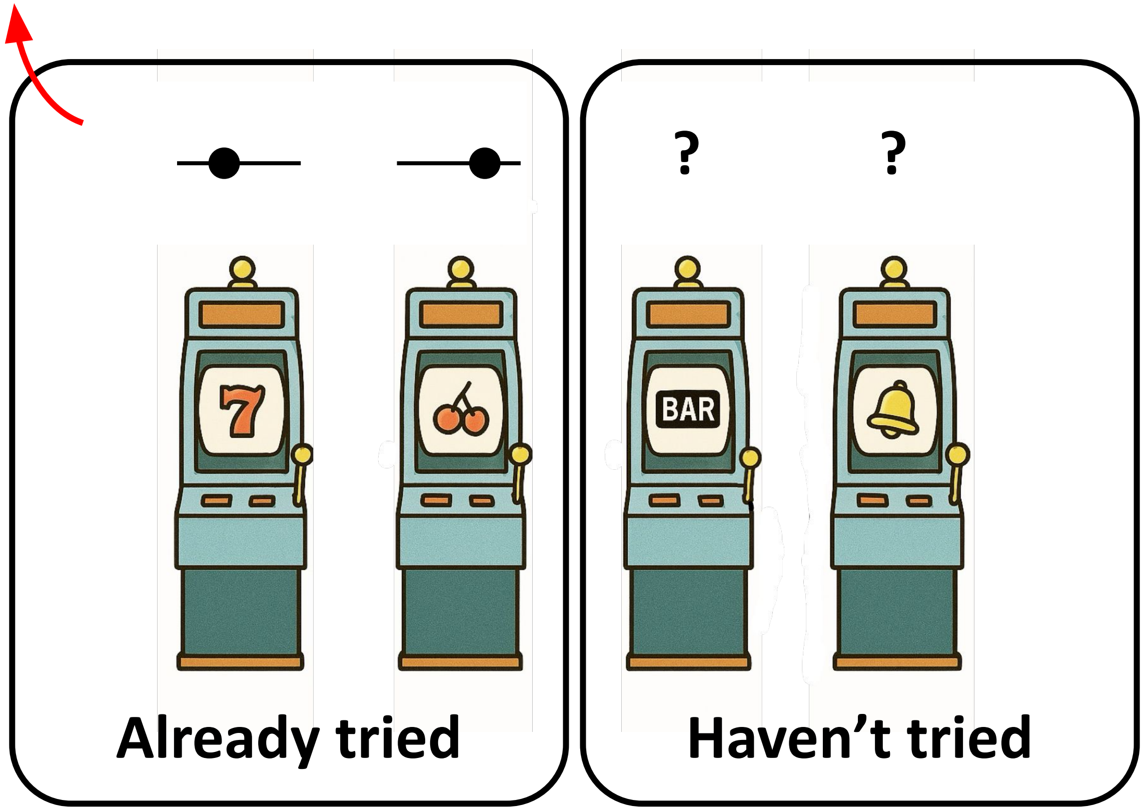


Already tried

Haven't tried

Algorithm: ϵ-Greedy

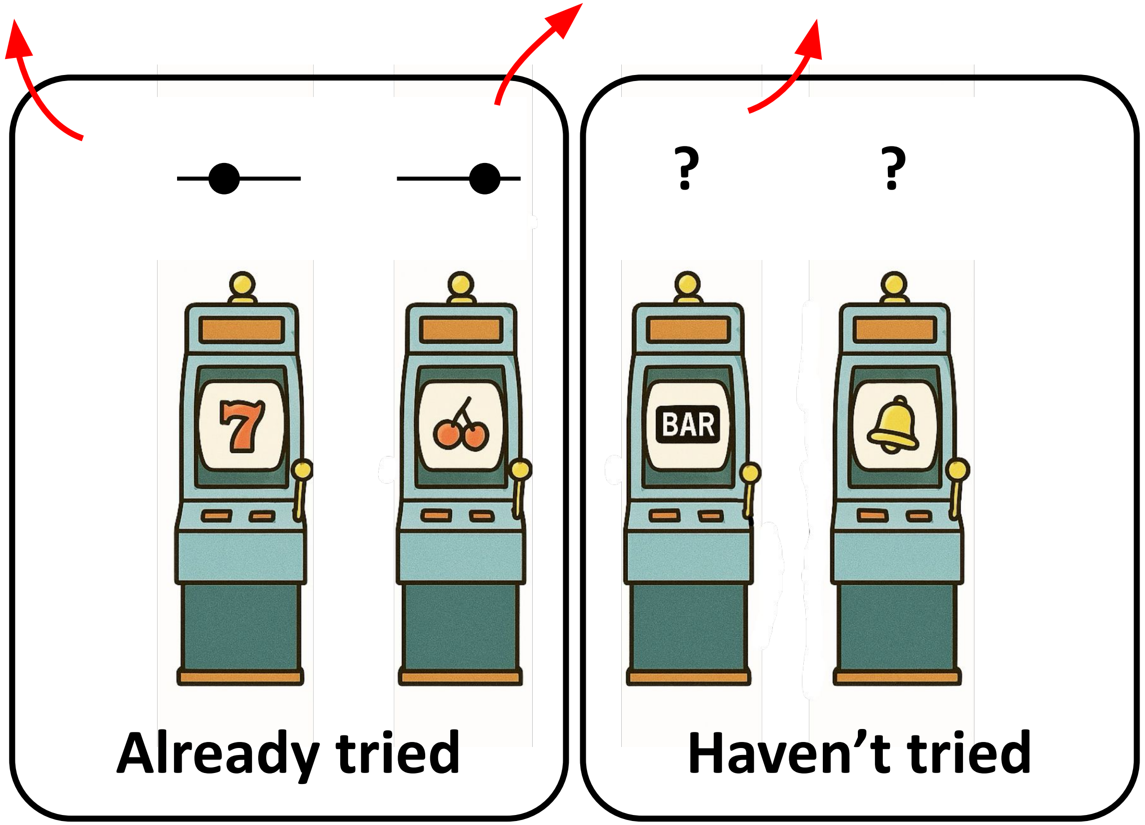
W.p. 1 - ϵ, choose the best (exploit)



Algorithm: ϵ-Greedy

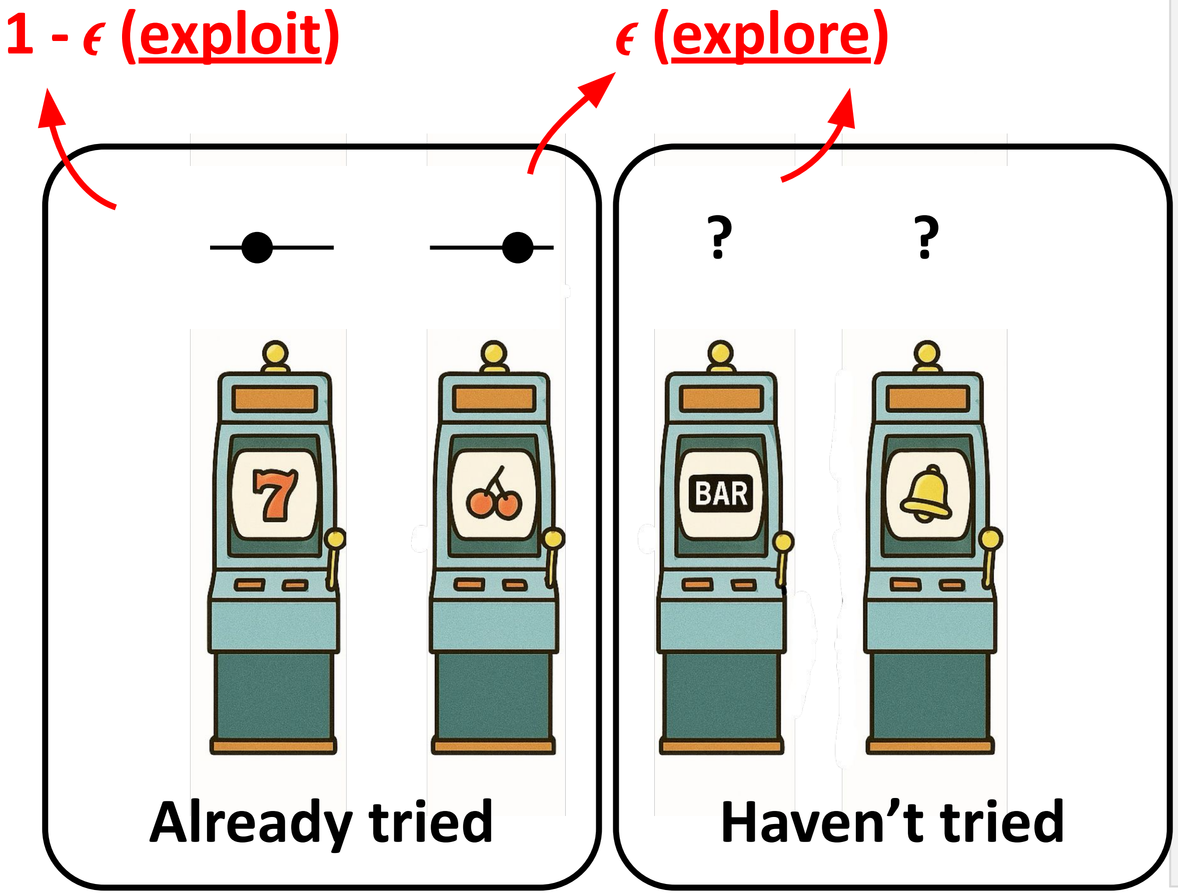
W.p. 1 - ϵ, choose the best (exploit)

W.p. ϵ, choose one randomly (explore)



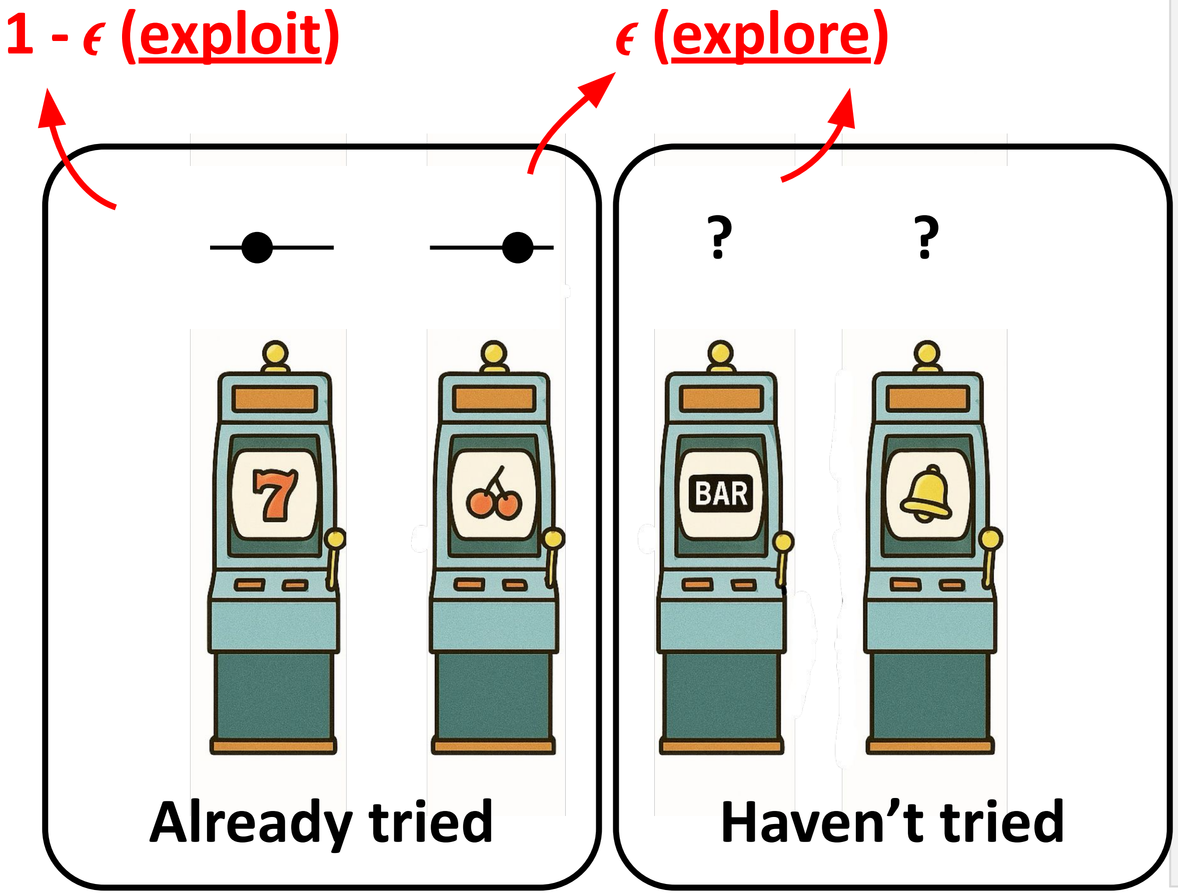
Algorithm: ϵ-Greedy

- At later rounds, the **very bad** actions can still be selected.



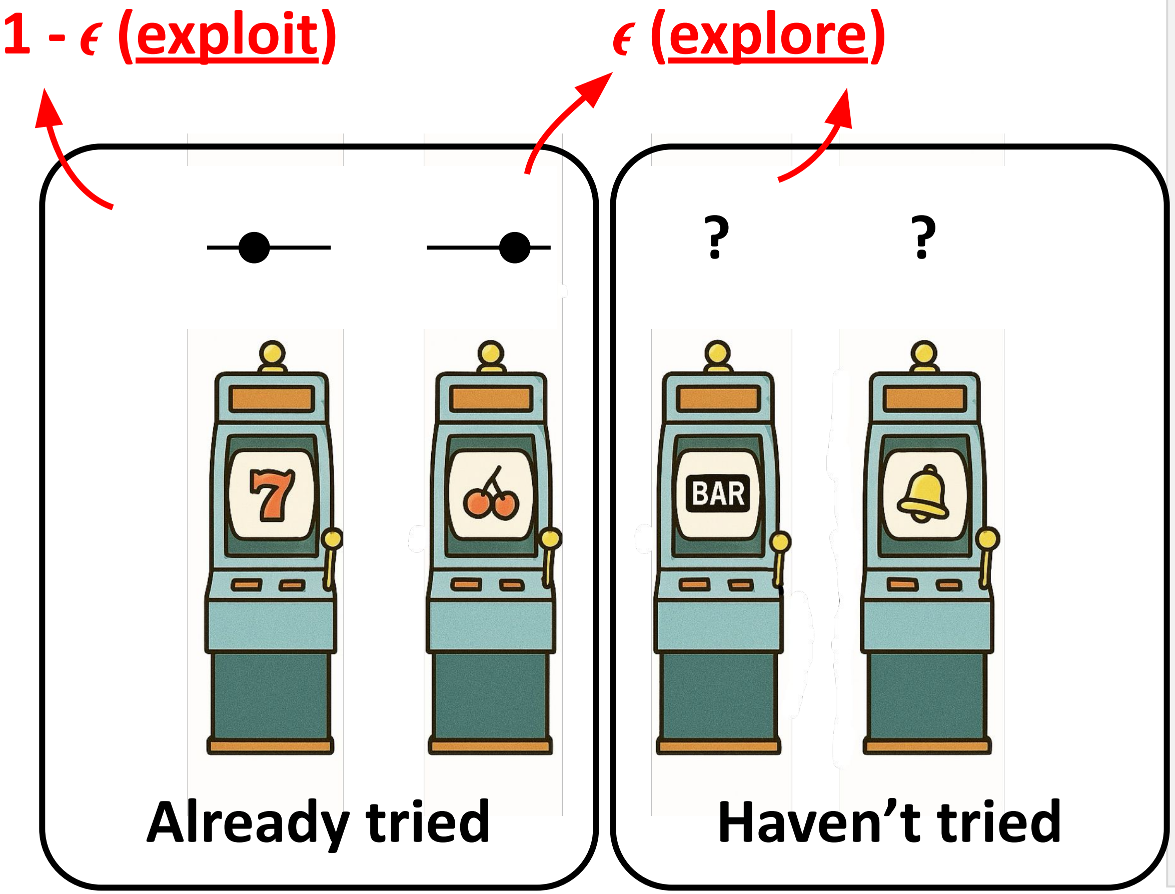
Algorithm: ϵ-Greedy

- At later rounds, the **very bad** actions can still be selected.
  - Use a decaying ϵ.



Algorithm: ϵ-Greedy

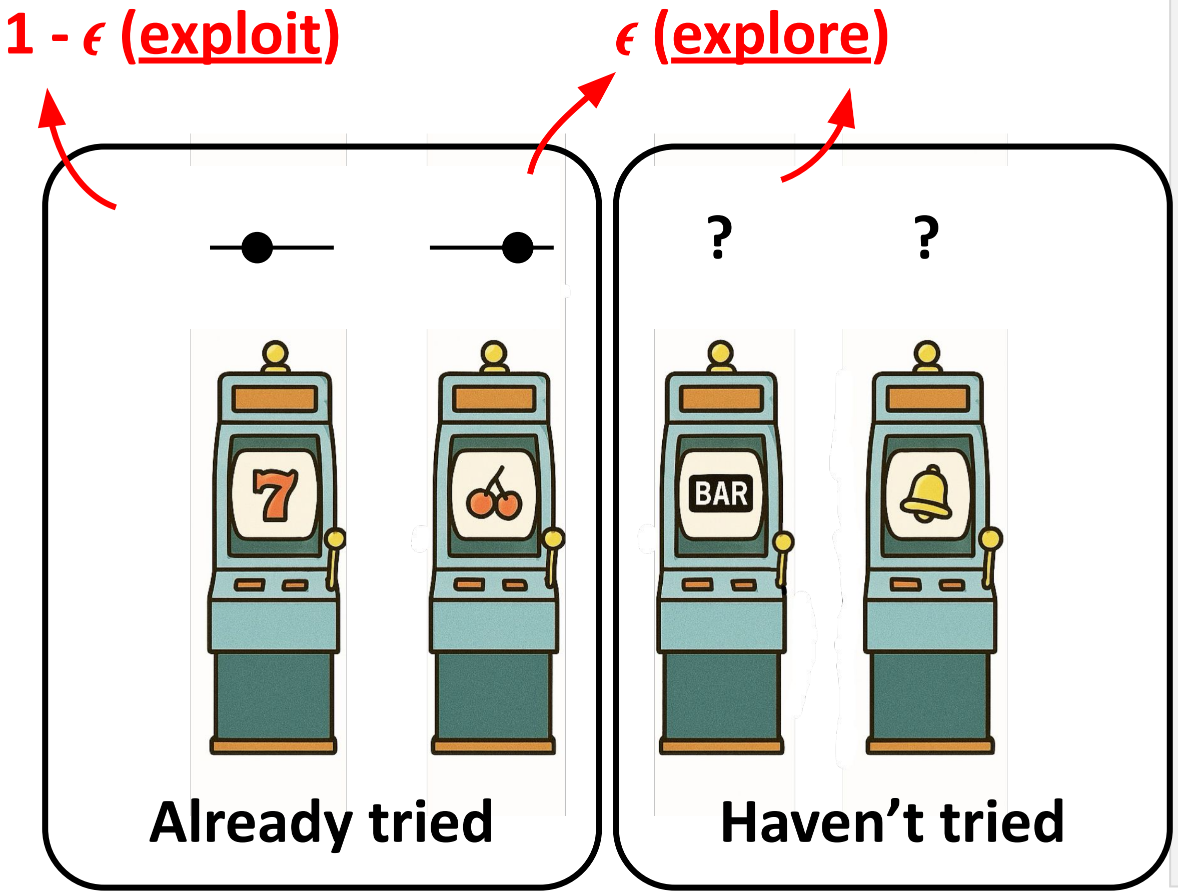
- At later rounds, the **very bad** actions can still be selected.
  - Use a decaying ϵ.
- No uncertainty quantification.



AY2024/25 Sem2
CS4246/CS5446

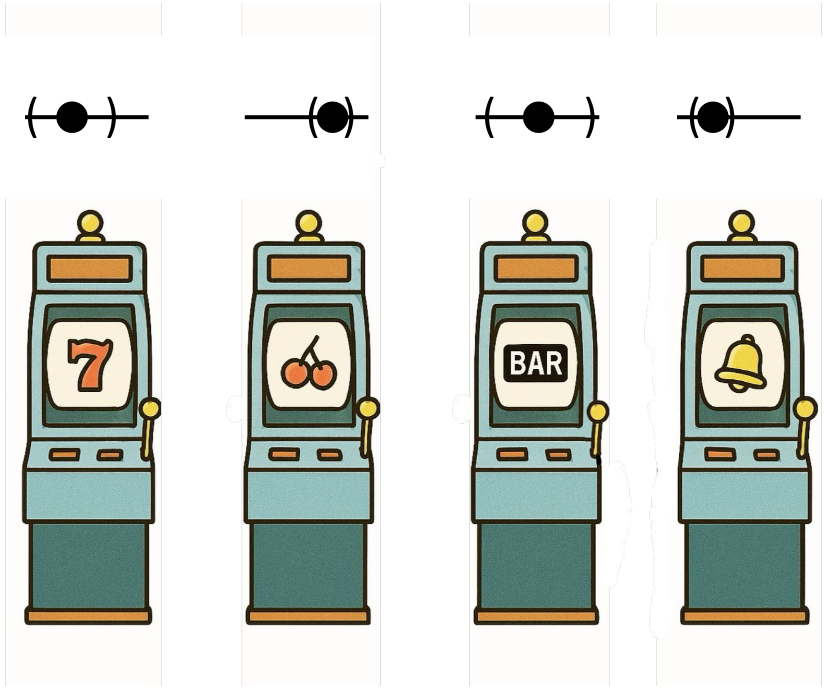
Algorithm: ϵ-Greedy

- At later rounds, the **very bad** actions can still be selected.
  - Use a decaying ϵ.
- No uncertainty quantification.
  - Involve a confidence term.



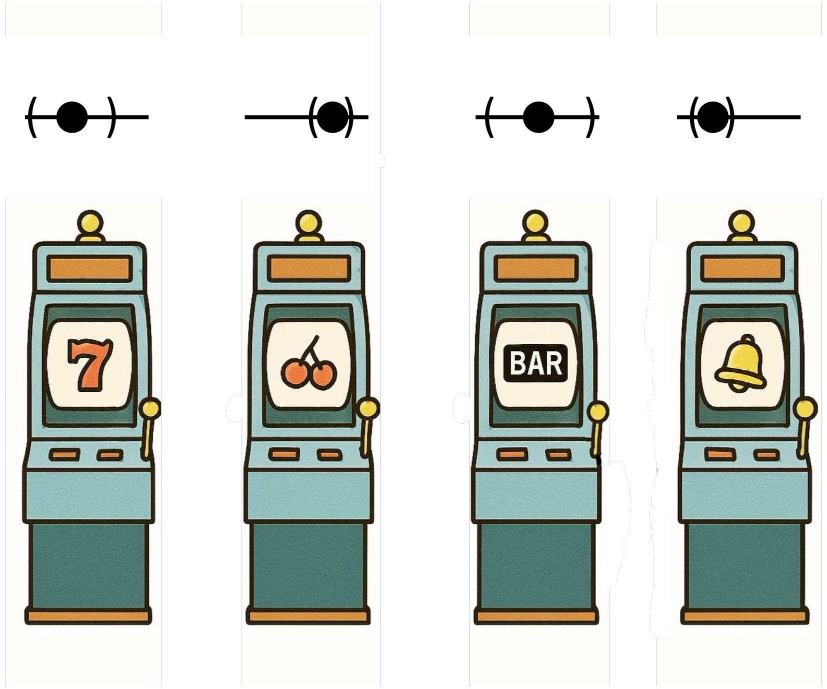
Algorithm: Upper Confidence Bound (UCB)

- Each action is associated with a **mean** and a **confidence term**.



Algorithm: Upper Confidence Bound (UCB)

- Each action is associated with a **mean** and a **confidence term**.



More confident if we try more!

Algorithm: Upper Confidence Bound (UCB)

- Each action is associated with a **mean** and a **confidence term**.
- Hoeffding bound:

$$\Pr\left[\left|\frac{1}{n}\sum_{\tau=1}^{\mathcal{T}} X_\tau - \mathbb{E}\left[X\right]\right| \geq (b-a)\sqrt{\frac{\log(2/\delta)}{2\mathcal{T}}}\right] \leq \delta.$$

  - Each independent X_t is bounded between $[a,\, b]$.

Algorithm: Upper Confidence Bound (UCB)

- Each action is associated with a **mean** and a **confidence term**.
- Hoeffding bound:

$$\Pr\left[\left|\frac{1}{n}\sum_{\tau=1}^{\mathcal{T}} X_\tau - \mathbb{E}\left[X\right]\right| \geq (b-a)\sqrt{\frac{\log(2/\delta)}{2\mathcal{T}}}\right] \leq \delta.$$

  - Each independent X_t is bounded between $[a, b]$.
  - For a fixed δ, the bound gets smaller when $\mathcal{T}$ is larger.

More confident if we try more!

Algorithm: Upper Confidence Bound (UCB)

- Each action is associated with a **mean** and a **confidence term**.
- Hoeffding bound:

$$\Pr\left[\left|\underset{\hat{r}^{t}_{a_k}}{\boxed{\frac{1}{n}\sum_{\tau=1}^{\mathcal{T}} X_\tau}} - \mathbb{E}[X]\right| \geq (b-a)\underset{\hat{u}^{t}_{a_k}}{\boxed{\sqrt{\frac{\log(2/\delta)}{2\mathcal{T}}}}}\right] \leq \delta.$$

  - Each independent X_t is bounded between $[a, b]$.
  - For a fixed δ, the bound gets smaller when $\mathcal{T}$ is larger.

Algorithm: Upper Confidence Bound (UCB)

- Each action is associated with a **mean** and a **confidence term**.
- Hoeffding bound:

$$\Pr\left[\left|\underbrace{\boxed{\frac{1}{n}\sum_{\tau=1}^{\mathcal{T}} X_\tau}}_{\hat{r}^t_{a_k}} - \mathbb{E}[X]\right| \geq (b-a)\underbrace{\boxed{\sqrt{\frac{\log(2/\delta)}{2\mathcal{T}}}}}_{\hat{u}^t_{a_k}}\right] \leq \delta.$$

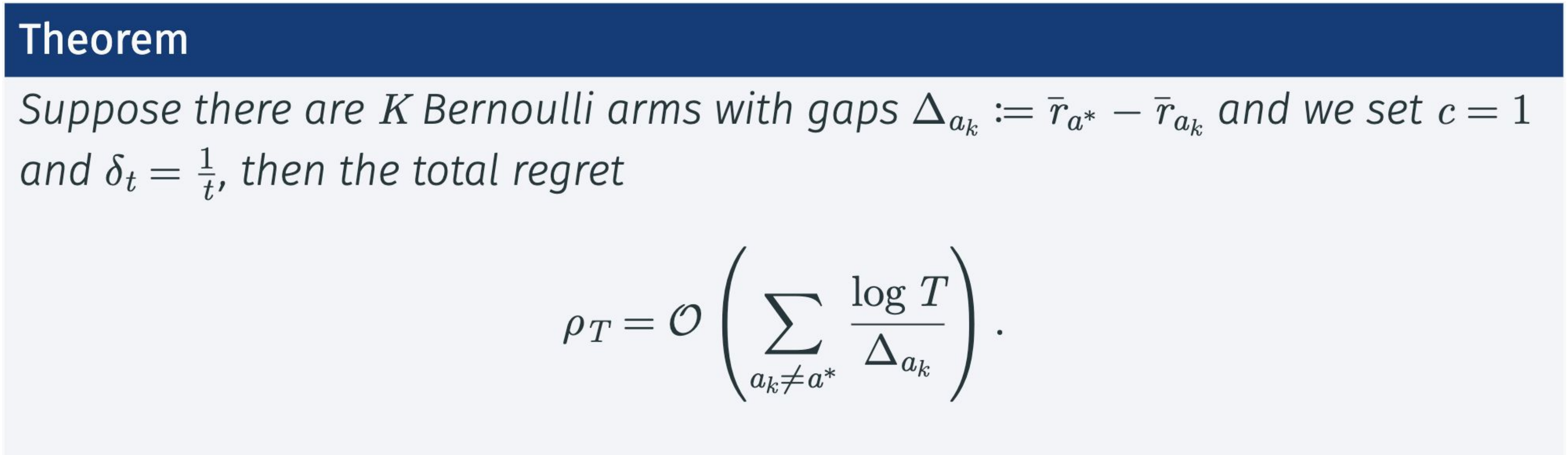
  - Each independent X_t is bounded between $[a, b]$.
  - For a fixed δ, the bound gets smaller when $\mathcal{T}$ is larger.
- At each round *t*, choose the action that maximizes

$$\mathrm{UCB}^t_{a_k} := \hat{\mu}^t_{a_k} + c \cdot \hat{u}^t_{a_k}.$$

Algorithm: Upper Confidence Bound (UCB)

- At each round *t*, choose the action that maximizes

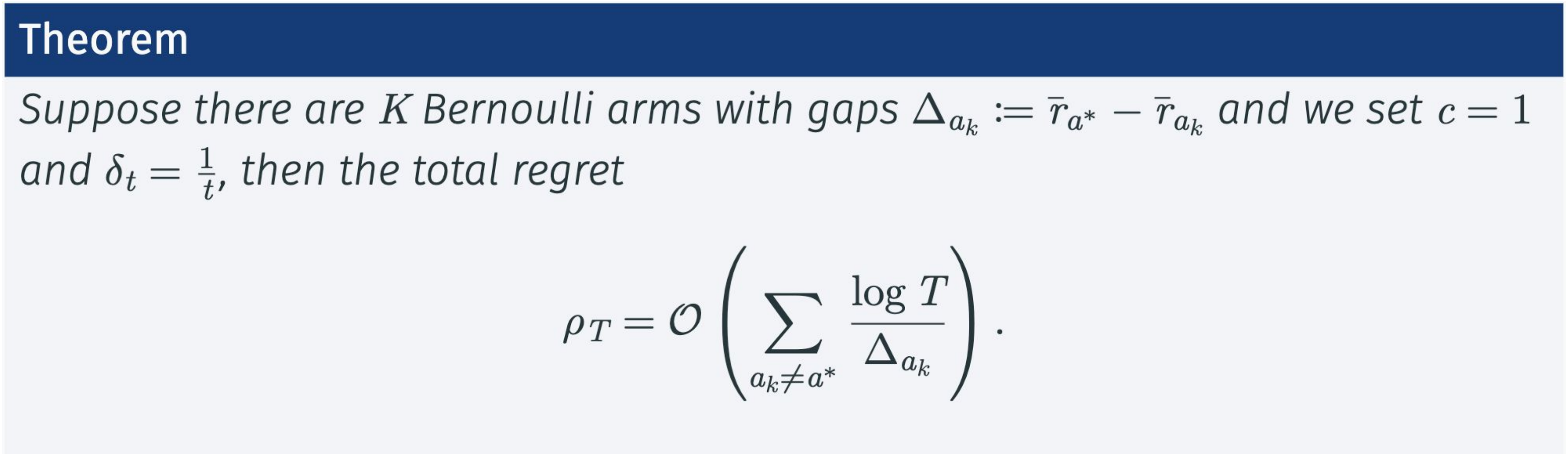
$$\mathrm{UCB}_{a_k}^t := \hat{\mu}_{a_k}^t + c \cdot \hat{u}_{a_k}^t.$$



Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right).$$

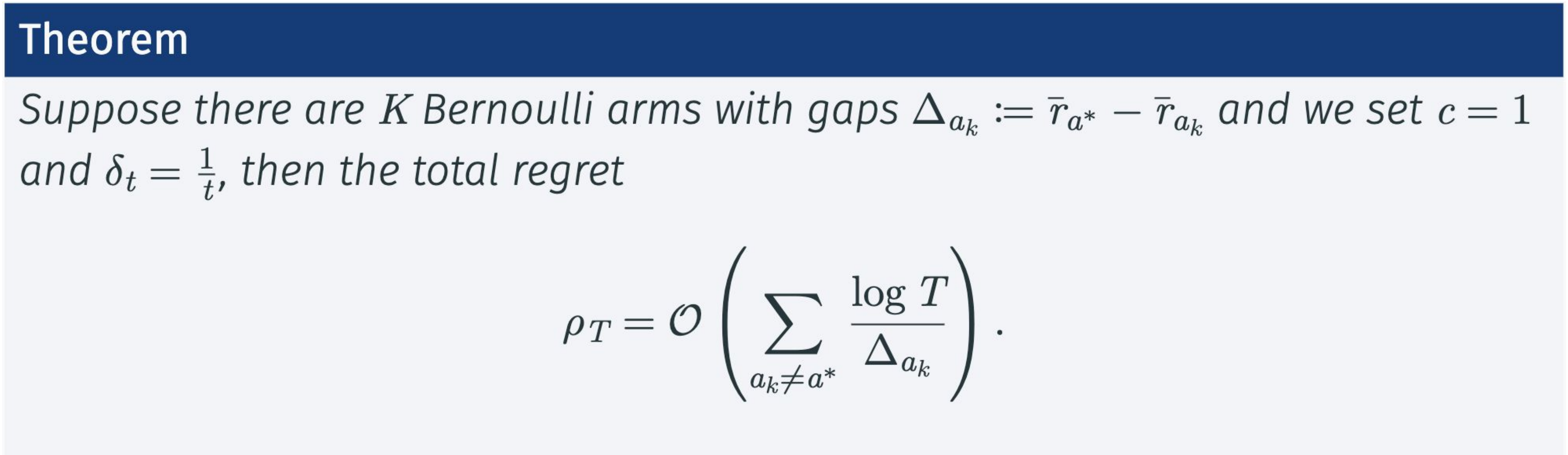


Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right).$$

- Consider the high-probability event $\forall a_k, t\ [|\hat{\mu}_{a_k}^t - \mu_{a_k}| \leq \hat{u}_{a_k}^t]$.

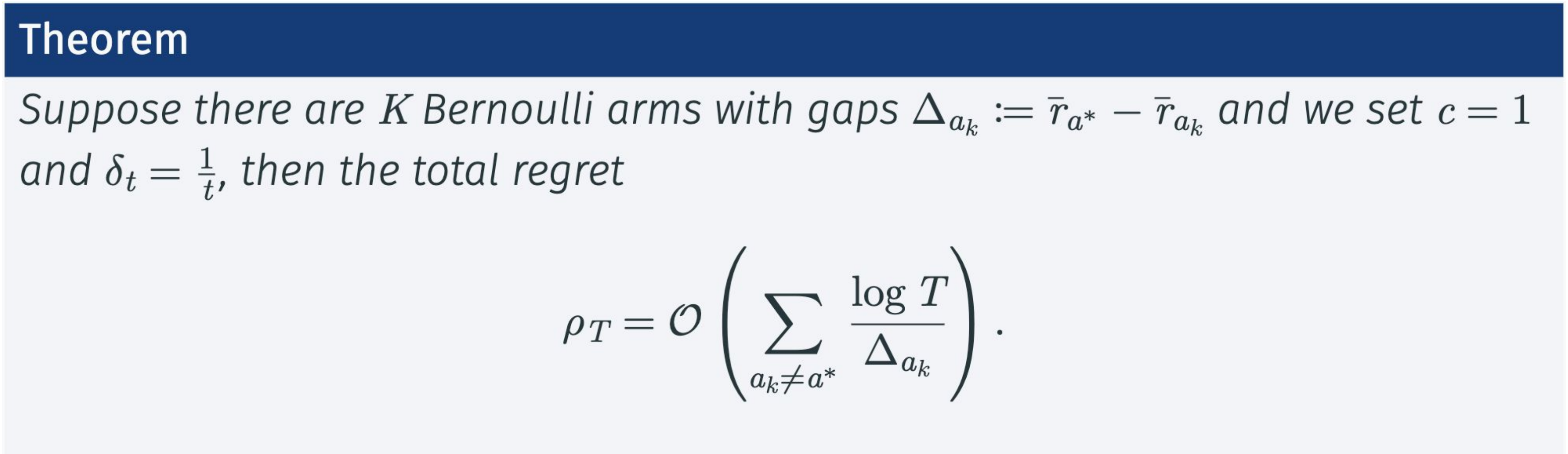


Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right).$$

- Consider the high-probability event $\forall a_k, t\ [|\hat{\mu}^t_{a_k} - \mu_{a_k}| \leq \hat{u}^t_{a_k}]$.
    - Distance between the true mean and the empirical mean is at most our confidence.

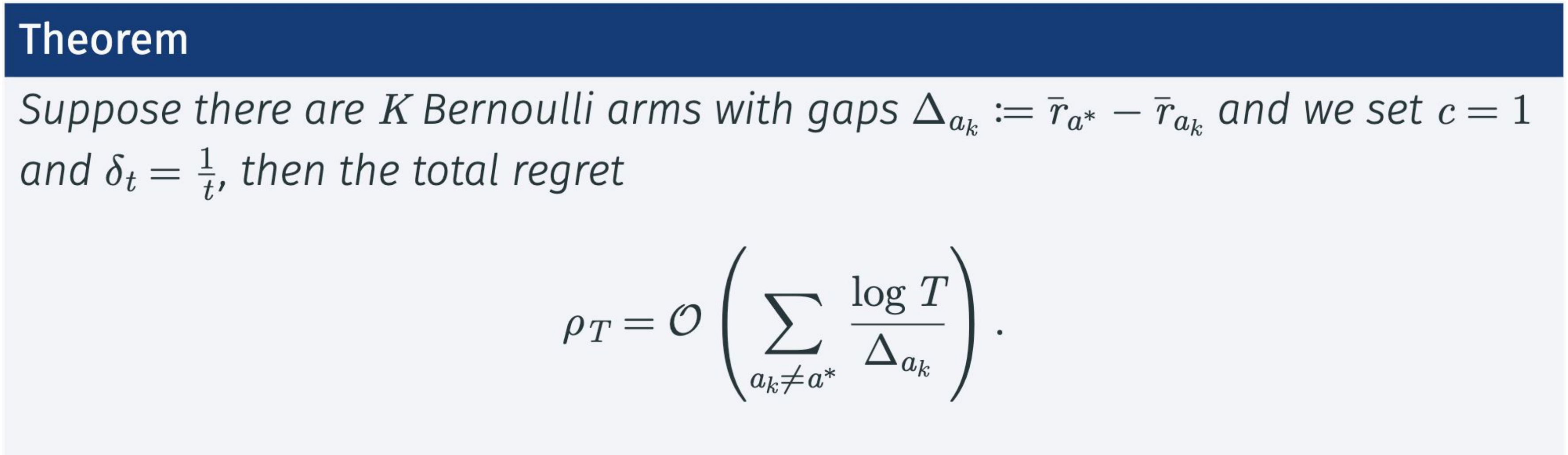


Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right).$$

- Consider the high-probability event $\forall a_k, t \; [|\hat{\mu}_{a_k}^t - \mu_{a_k}| \leq \hat{u}_{a_k}^t]$.
  - Distance between the true mean and the empirical mean is at most our confidence.
- An action $a_k \neq a^*$ is selected over a^* because $\mathrm{UCB}_{a_k}^t \geq \mathrm{UCB}_{a^*}^t$.

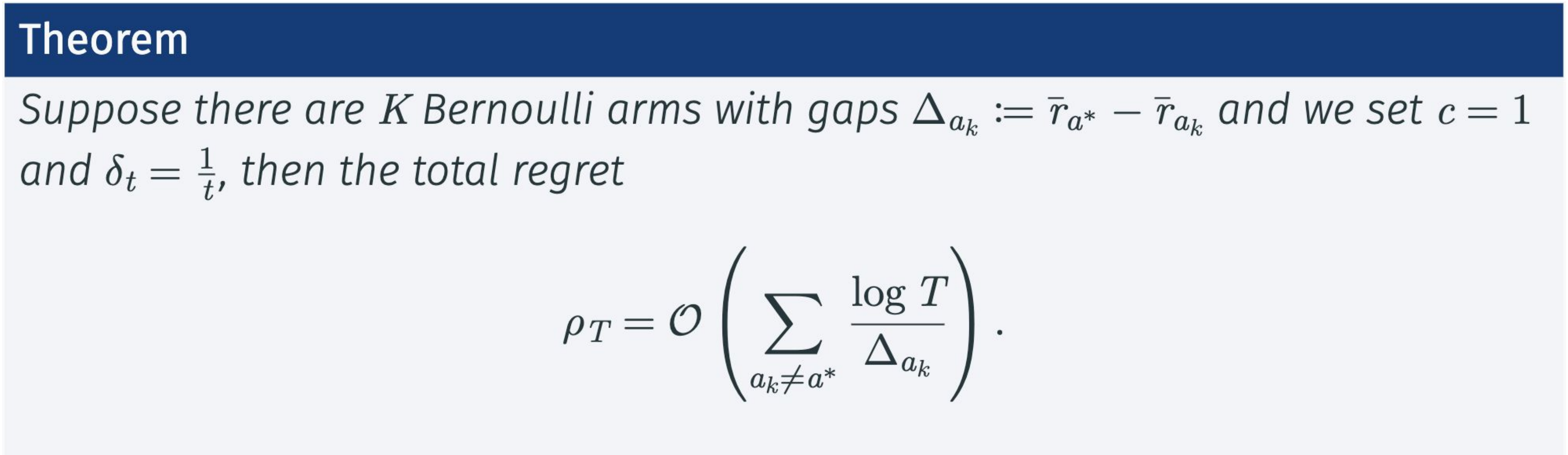


Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right).$$

- Consider the high-probability event $\forall a_k, t \; [|\hat{\mu}^t_{a_k} - \mu_{a_k}| \leq \hat{u}^t_{a_k}]$.
    - Distance between the true mean and the empirical mean is at most our confidence.
- An action $a_k \neq a^*$ is selected over a^* because $\mathrm{UCB}^t_{a_k} \geq \mathrm{UCB}^t_{a^*}$.
    - LHS = empirical mean + confidence ≤ true mean + confidence + confidence (of a_k).

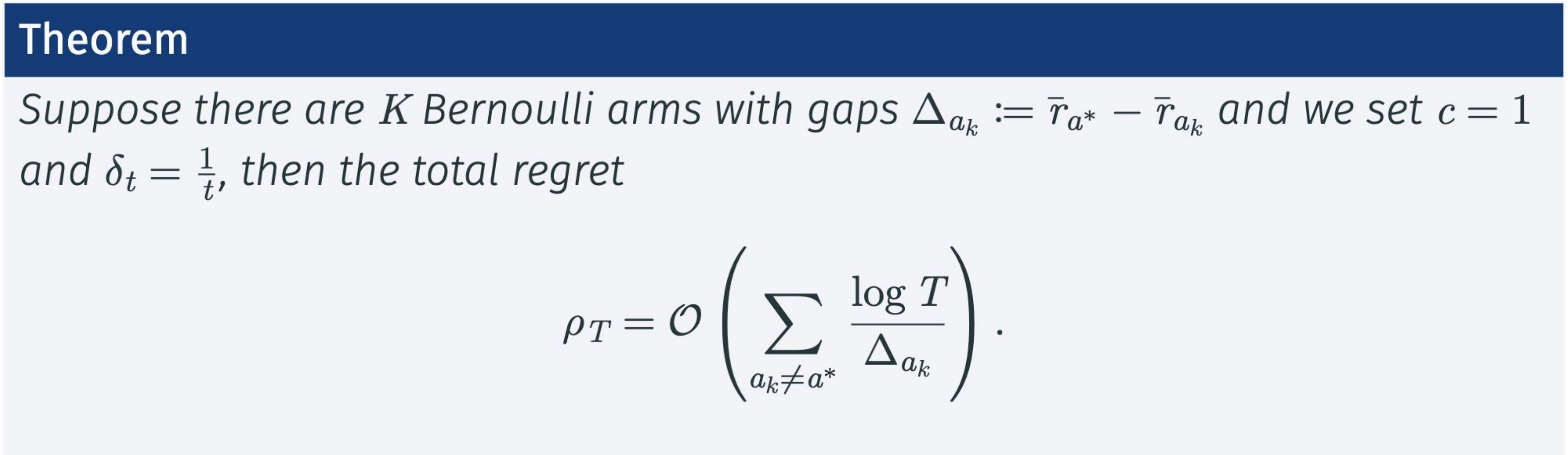


Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right).$$

- Consider the high-probability event $\forall a_k, t \; [|\hat{\mu}^t_{a_k} - \mu_{a_k}| \leq \hat{u}^t_{a_k}]$.
  - Distance between the true mean and the empirical mean is at most our confidence.
- An action $a_k \neq a^*$ is selected over a^* because $\mathrm{UCB}^t_{a_k} \geq \mathrm{UCB}^t_{a^*}$.
  - LHS = empirical mean + confidence ≤ true mean + confidence + confidence (of a_k).
  - RHS ≥ true mean (of a^*).

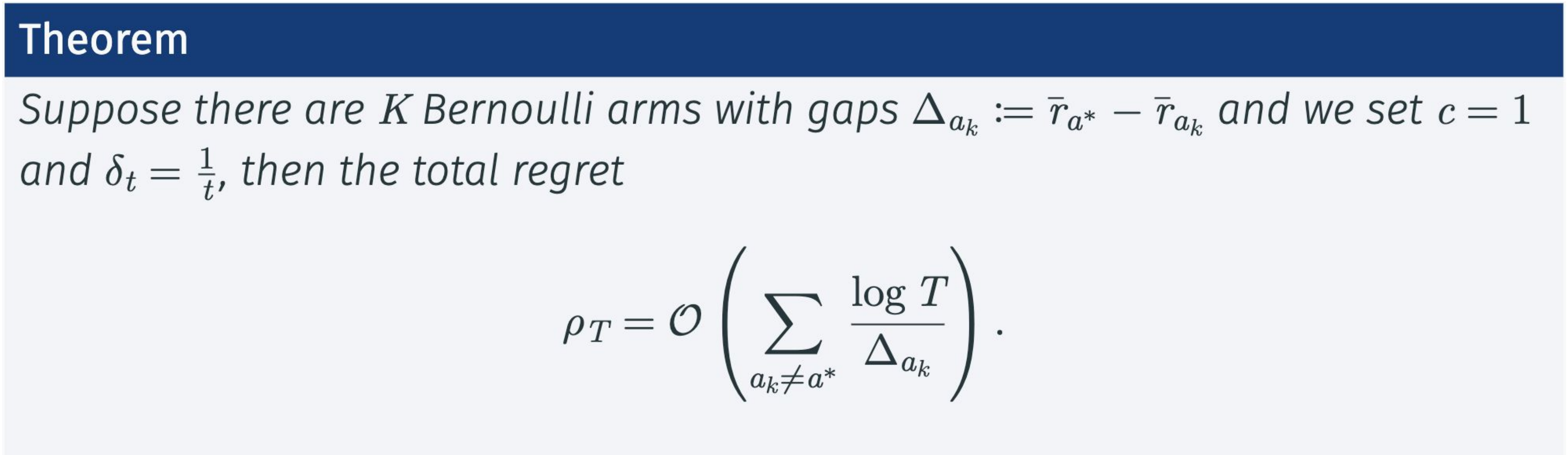


Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right).$$

- Consider the high-probability event $\forall a_k, t \; [|\hat{\mu}^t_{a_k} - \mu_{a_k}| \leq \hat{u}^t_{a_k}]$.
  - Distance between the true mean and the empirical mean is at most our confidence.
- An action $a_k \neq a^*$ is selected over a^* because $\mathrm{UCB}^t_{a_k} \geq \mathrm{UCB}^t_{a^*}$.
  - LHS = empirical mean + confidence ≤ true mean + confidence + confidence (of a_k).
  - RHS ≥ true mean (of a^*).
  - If Δ_{a_k} is large, for LHS to be larger than RHS, confidence of a_k cannot be too small!

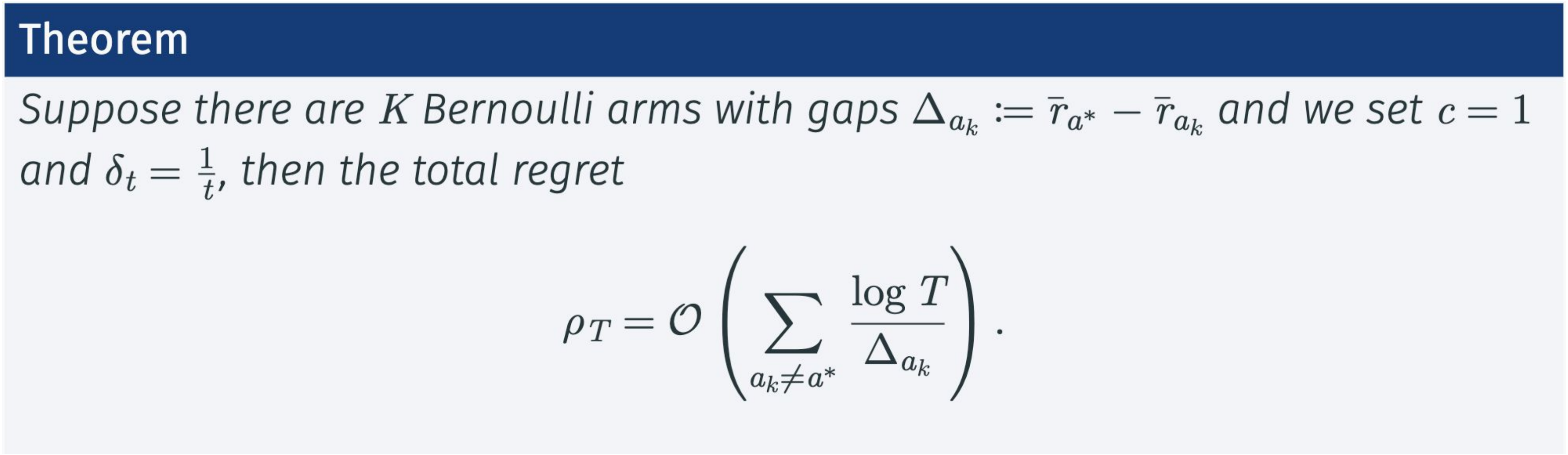


Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right).$$

- Consider the high-probability event $\forall a_k, t \; [|\hat{\mu}^t_{a_k} - \mu_{a_k}| \leq \hat{u}^t_{a_k}]$.
  - Distance between the true mean and the empirical mean is at most our confidence.
- An action $a_k \neq a^*$ is selected over a^* because $\mathrm{UCB}^t_{a_k} \geq \mathrm{UCB}^t_{a^*}$.
  - LHS = empirical mean + confidence ≤ true mean + confidence + confidence (of a_k).
  - RHS ≥ true mean (of a^*).
  - If Δ_{a_k} is large, for LHS to be larger than RHS, confidence of a_k cannot be too small!
    - Used to bound the number of each action $a_k \neq a^*$ being selected.

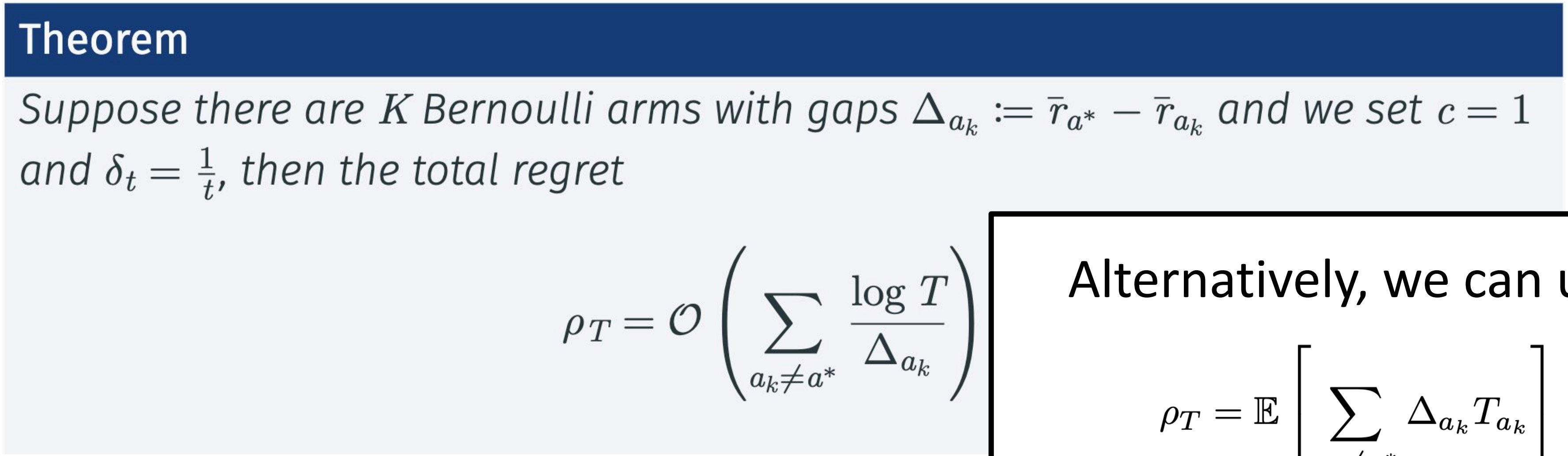


Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right).$$

- Consider the high-probability event $\forall a_k, t\ [|\hat{\mu}^t_{a_k} - \mu_{a_k}| \leq \hat{u}^t_{a_k}]$.
  - Distance between the true mean and the empirical mean is at most our confidence.
- An action $a_k \neq a^*$ is selected over a^* because $\mathrm{UCB}^t_{a_k} \geq \mathrm{UCB}^t_{a^*}$.
  - LHS = empirical mean + confidence ≤ true mean + confidence + confidence (of a_k).
  - RHS ≥ true mean (of a^*).
  - If Δ_{a_k} is large, for LHS to be larger than RHS, confidence of a_k cannot be too small!
    - Used to bound the number of each action $a_k \neq a^*$ being selected.
- Factor in the probability the event does not happen and sum up everything.



Theorem

Suppose there are K Bernoulli arms with gaps $\Delta_{a_k} := \bar{r}_{a^*} - \bar{r}_{a_k}$ and we set $c = 1$ and $\delta_t = \frac{1}{t}$, then the total regret

$$\rho_T = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right)$$

Alternatively, we can use

$$\rho_T = \mathbb{E}\left[\sum_{a_k \neq a^*} \Delta_{a_k} T_{a_k}\right]$$

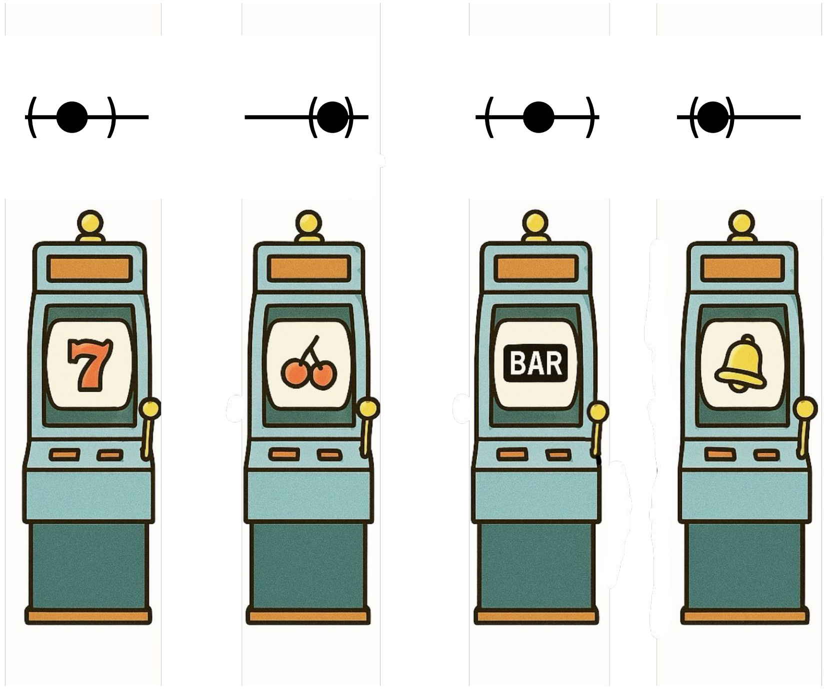
directly to achieve another bound:

$$\rho_T = \mathcal{O}\left(\sqrt{KT \log T}\right).$$

- Consider the high-probability event $\forall a_k, t\ [|\hat{\mu}^t_{a}$
  - Distance between the true mean and the empirical mea
- An action $a_k \neq a^*$ is selected over a^* because
  - LHS = empirical mean + confidence ≤ true mean + confi
  - RHS ≥ true mean (of a^*).
  - If Δ_{a_k} is large, for LHS to be larger than RHS, confidence of a_k cannot be too small!
    - Used to bound the number of each action $a_k \neq a^*$ being selected.
- Factor in the probability the event does not happen and sum up everything.

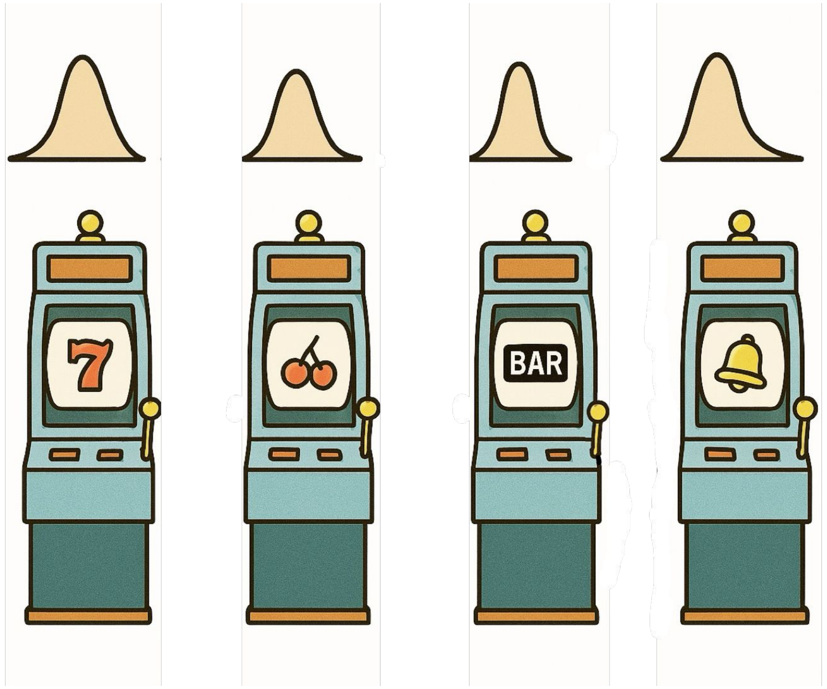
Algorithm: Upper Confidence Bound (UCB)

- Each action is associated with a **mean** and a **confidence term**.
- We use a quantity that needs a bound $[a, b]$ to quantify uncertainty.



A Bayesian View: Bayesian Bandit

- Each action is associated with a **distribution** (i.e., our belief).



A Bayesian View: Bayesian Bandit

- Each action is associated with a **distribution** (i.e., our belief).
- Whenever we try a new action, our belief is updated using Bayes’ rule:

$$p(\theta_{a_t}|r_{a_t}) = \frac{p(\theta_{a_t})p(r_{a_t}|\theta_{a_t})}{p(\theta_{a_t})}.$$

A Bayesian View: Bayesian Bandit

- Each action is associated with a **distribution** (i.e., our belief).
- Whenever we try a new action, our belief is updated using Bayes’ rule:

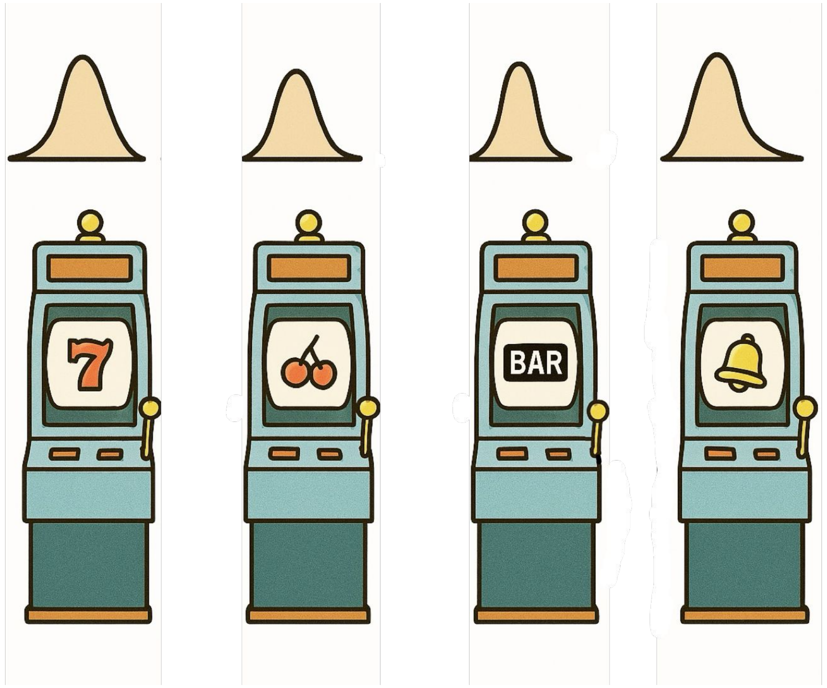
$$p(\theta_{a_t}|r_{a_t}) = \frac{p(\theta_{a_t})p(r_{a_t}|\theta_{a_t})}{p(\theta_{a_t})}.$$

- Goal: Minimize ***Bayesian regret***:

$$\mathbb{E}_{\text{prior}}[\rho_T].$$

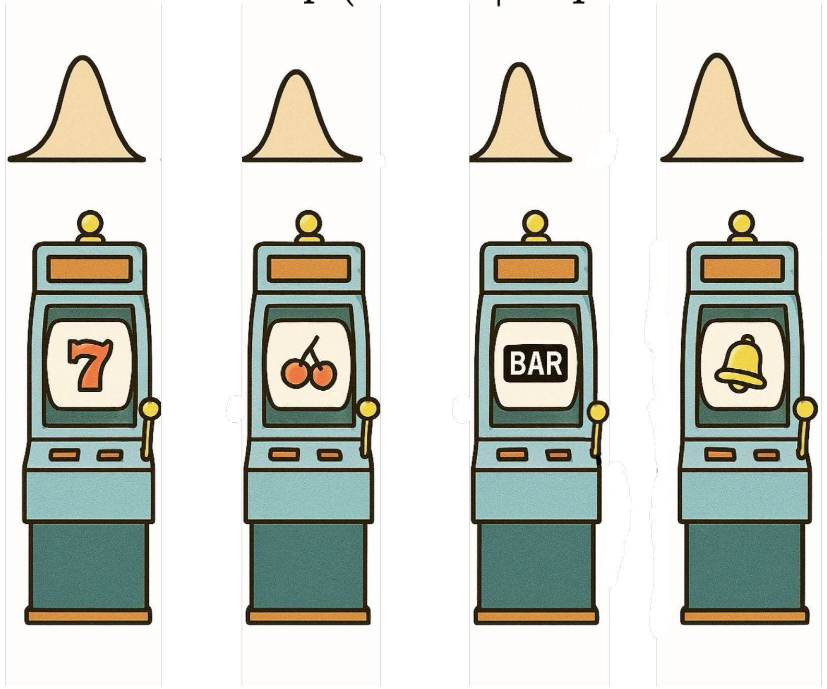
Algorithm: Thompson Sampling

- Each action is associated with a **distribution**.
- At each round t, we randomly sample an (estimated) reward for each action and choose the action that maximizes it.



Algorithm: Thompson Sampling

- Each action is associated with a **distribution**.
- At each round t, we randomly sample an (estimated) reward for each action and choose the action that maximizes it.
  - Equivalently, we are sampling from $p\left(a^* = a \middle| \text{all past observations}\right)$.



Algorithm: Thompson Sampling

- Each action is associated with a **distribution**.
- At each round t, we randomly sample an (estimated) reward for each action and choose the action that maximizes it.
  - Equivalently, we are sampling from $p\left(a^* = a \middle| \text{all past observations}\right)$.
- Bound on Bayesian regret:

$$\mathbb{E}_{\text{prior}}\left[\rho_T\right] = \mathcal{O}\left(\sum_{a_k \neq a^*} \frac{\log T}{\Delta_{a_k}}\right)$$

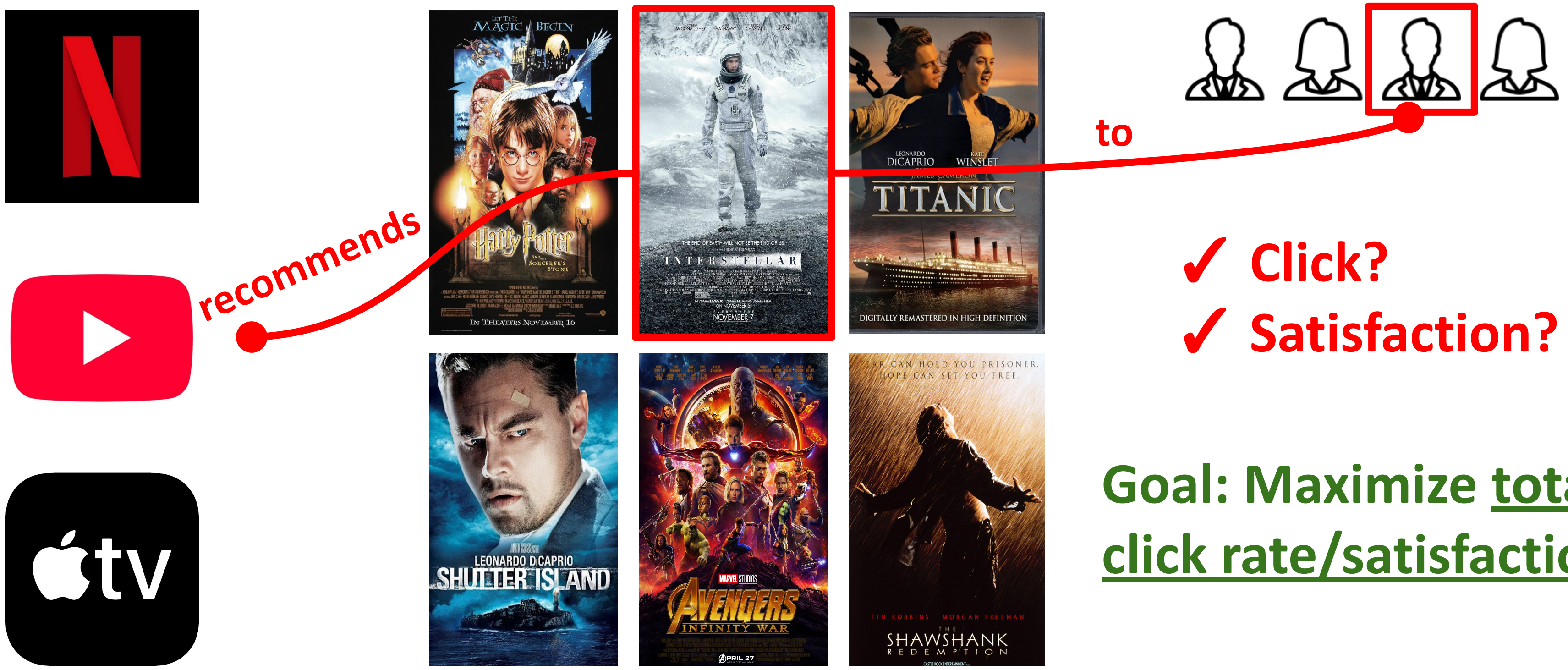
$$\mathbb{E}_{\text{prior}}[\rho_T] = \mathcal{O}\left(\sqrt{KT\log T}\right).$$

Application: Movie Recommendation

Movie recommender

Actions $a_1, \cdots, a_K$

Users



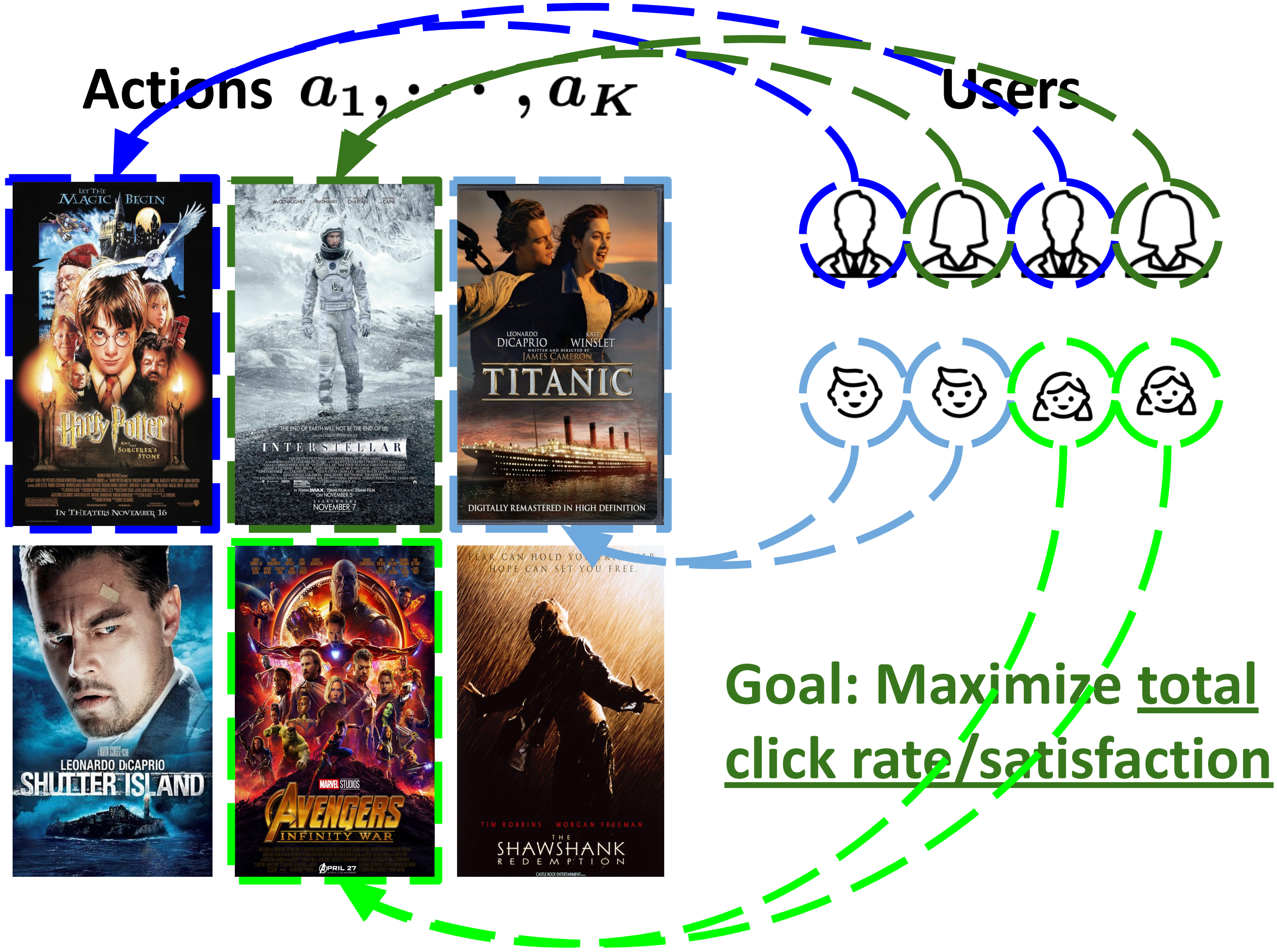
✓ **Click?**

✓ **Satisfaction?**

Goal: Maximize <u>total click rate/satisfaction</u>.

Application: Movie Recommendation

Different groups have different preferences.



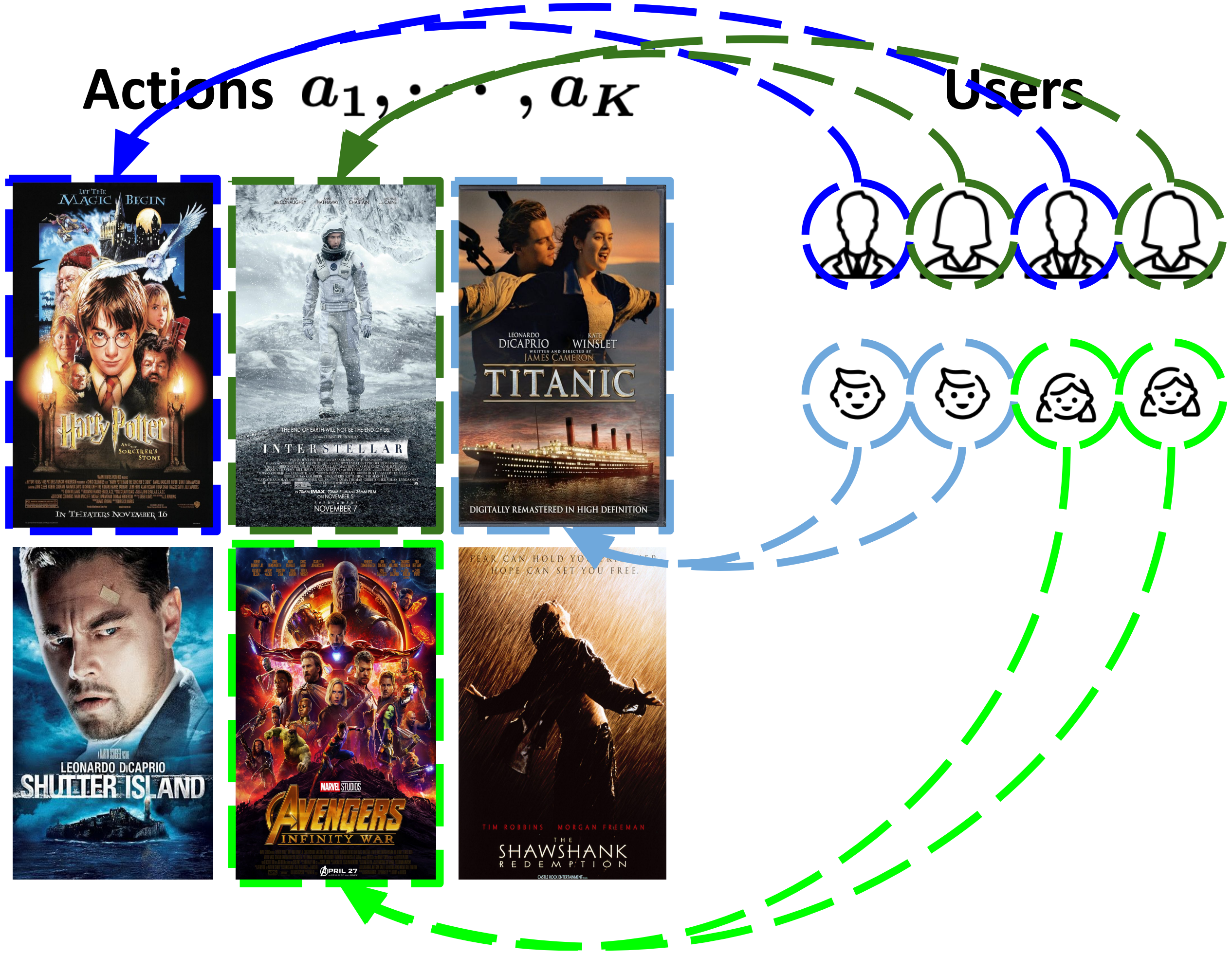
Goal: Maximize total click rate/satisfaction.

Application: Movie Recommendation

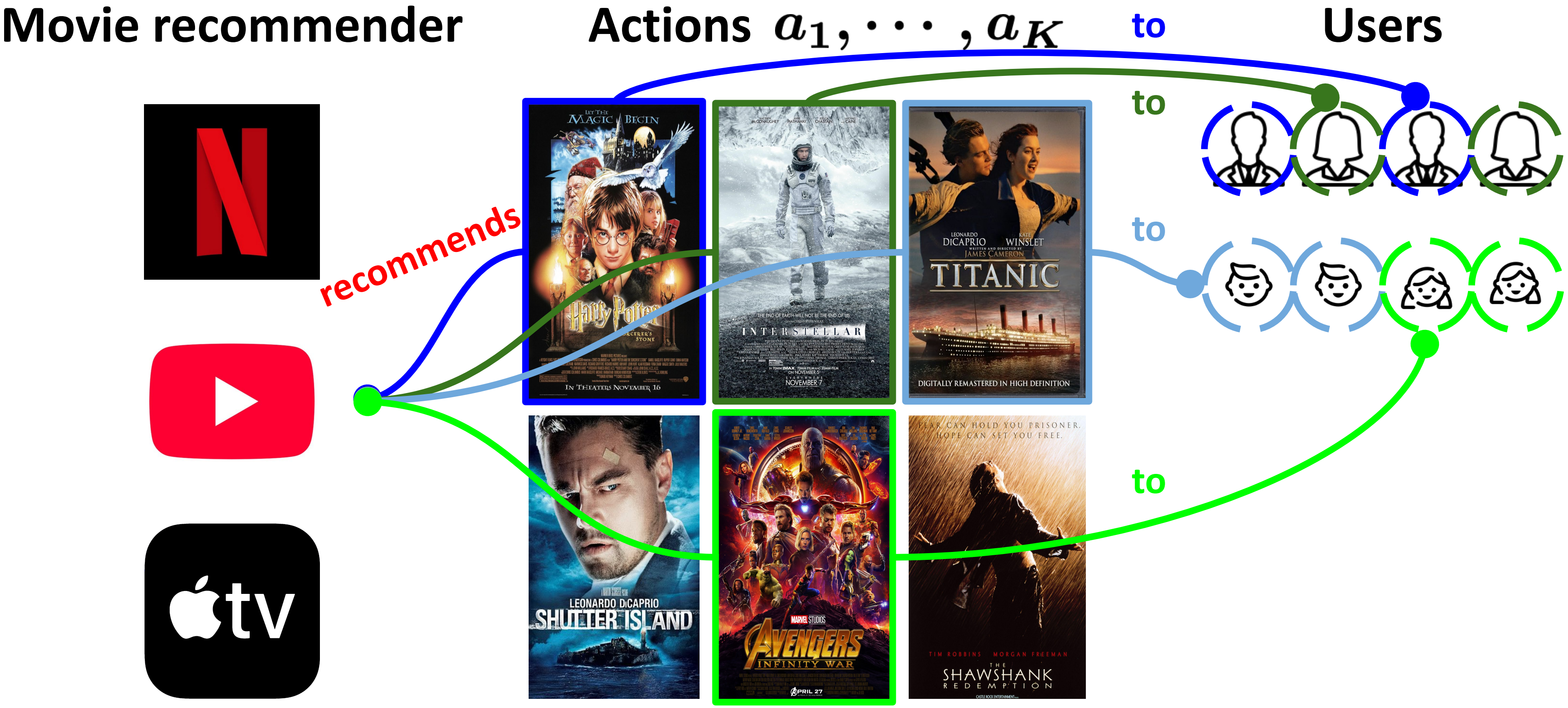
Different groups have different preferences.

↓

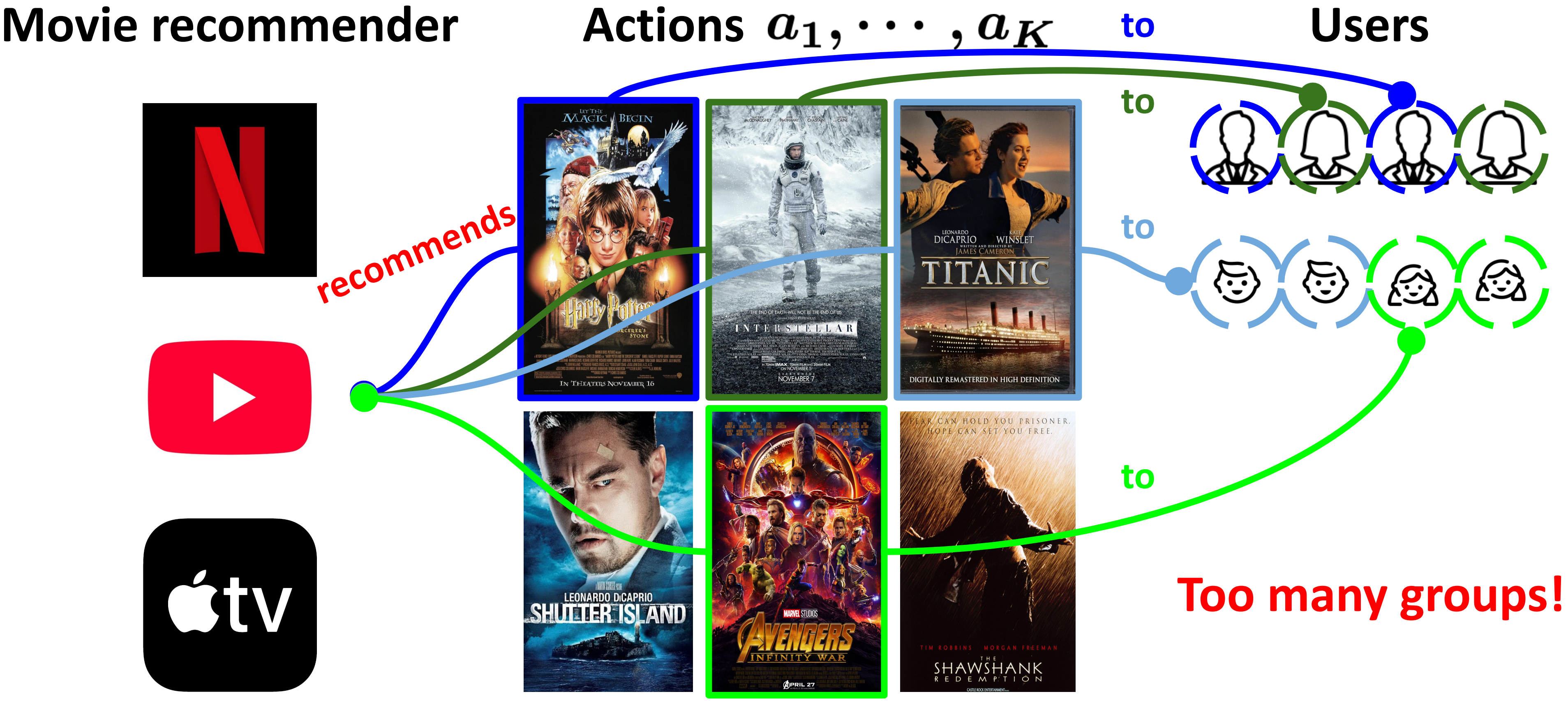
A <u>one-size-fit-all</u> solution does not work well!



Application: Movie Recommendation

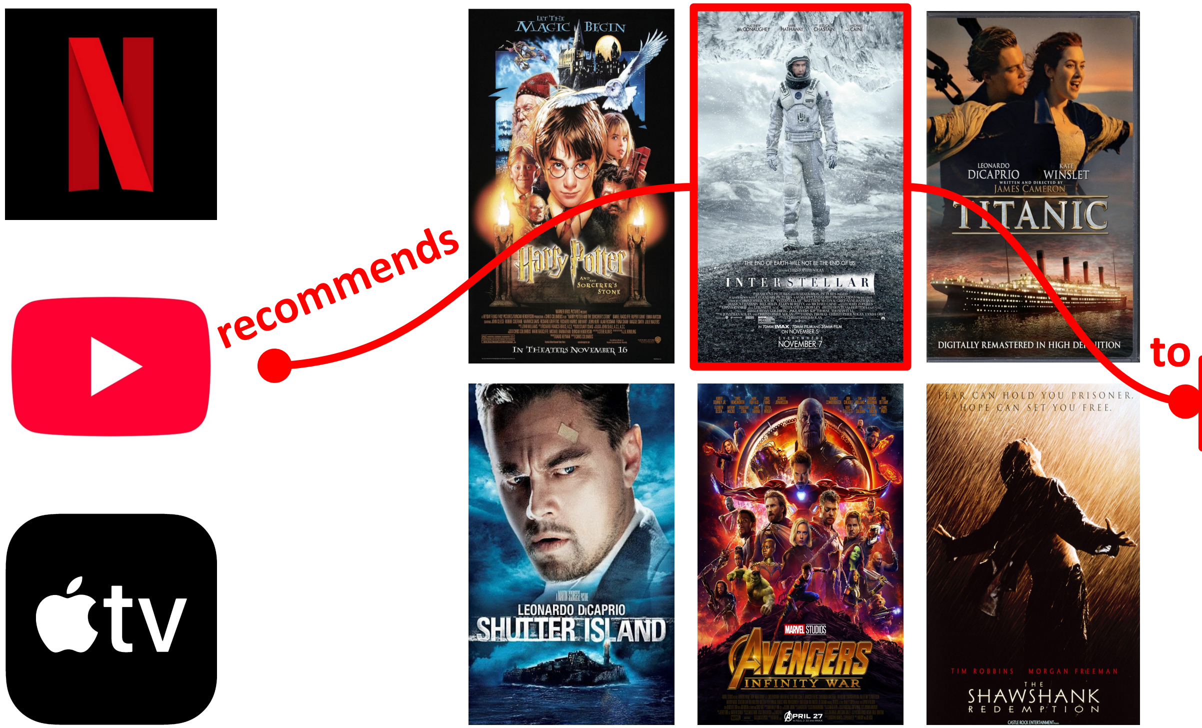


Application: Movie Recommendation



Application: Movie Recommendation

Movie recommender **Actions $a_1, \cdots, a_K$** **Users**



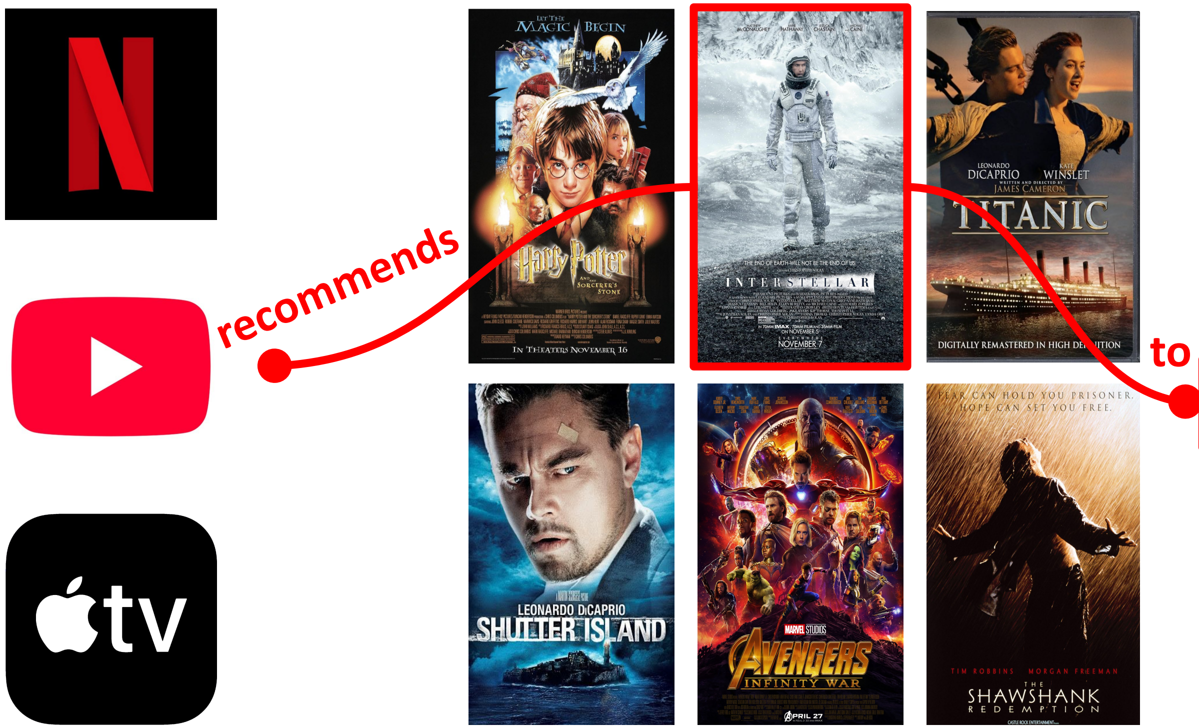
	Features			Rewards R
	Sex	**Age**	**…**	
	M	9	…	…
	M	28	…	…
	F	6	…	…
	F	23	…	…
	…	…	…	…

Contextual Bandit: $B = (A, X, R)$

Movie recommender

Actions A

Users



Contexts X

~~**Features**~~

Rewards R

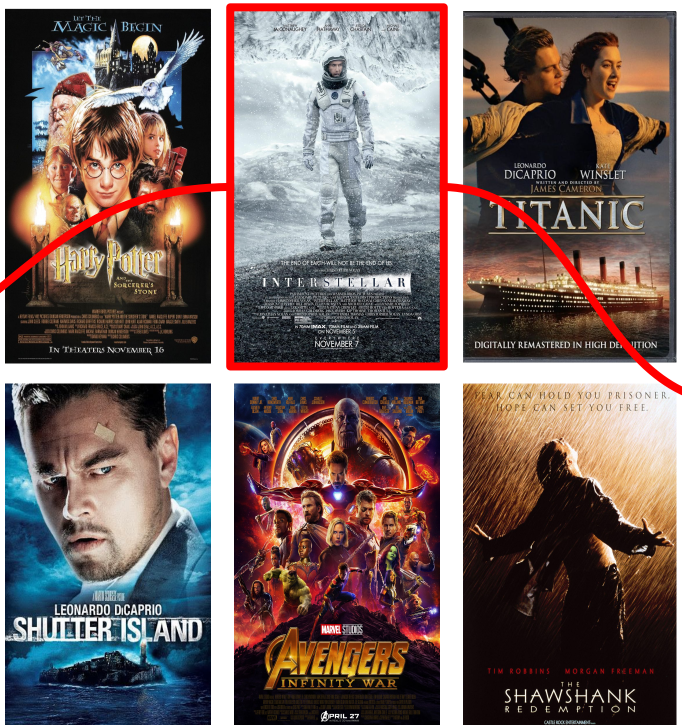
	Sex	Age	…	
	M	9	…	…
	M	28	…	…
	F	6	…	…
	F	23	…	…
	…	…	…	…

Contextual Bandit: $B = (A, X, R)$

Movie recommender

Actions A

Users



recommends

to

Contexts X
~~Features~~

Rewards R

	Sex	Age	…		
	M	9	…		…
	M	28	…	➡	…
	F				
	F	23	…		…
	…	…	…		…

$p(R|A, X)$

Contextual Bandit: $B = (A, X, R)$

- Modelling assumption:

$$R_{\mathbf{x}} = f(\mathbf{x}) + \xi_{\mathbf{x}}$$
$$\mathbb{E}[R_{\mathbf{x}}] = f(\mathbf{x}).$$

  - Each context $\mathbf{x} \in A \times X$ contains both action and features.
  - $\xi_{\mathbf{x}}$ is a zero-mean noise conditioned on $\mathbf{x}$.

Contextual Bandit: $B = (A, X, R)$

- Modelling assumption:

$$R_{\mathbf{x}} = f(\mathbf{x}) + \xi_{\mathbf{x}}$$
$$\mathbb{E}[R_{\mathbf{x}}] = f(\mathbf{x}).$$

  - Each context $\mathbf{x} \in A \times X$ contains both action and features.
  - $\xi_{\mathbf{x}}$ is a zero-mean noise conditioned on $\mathbf{x}$.

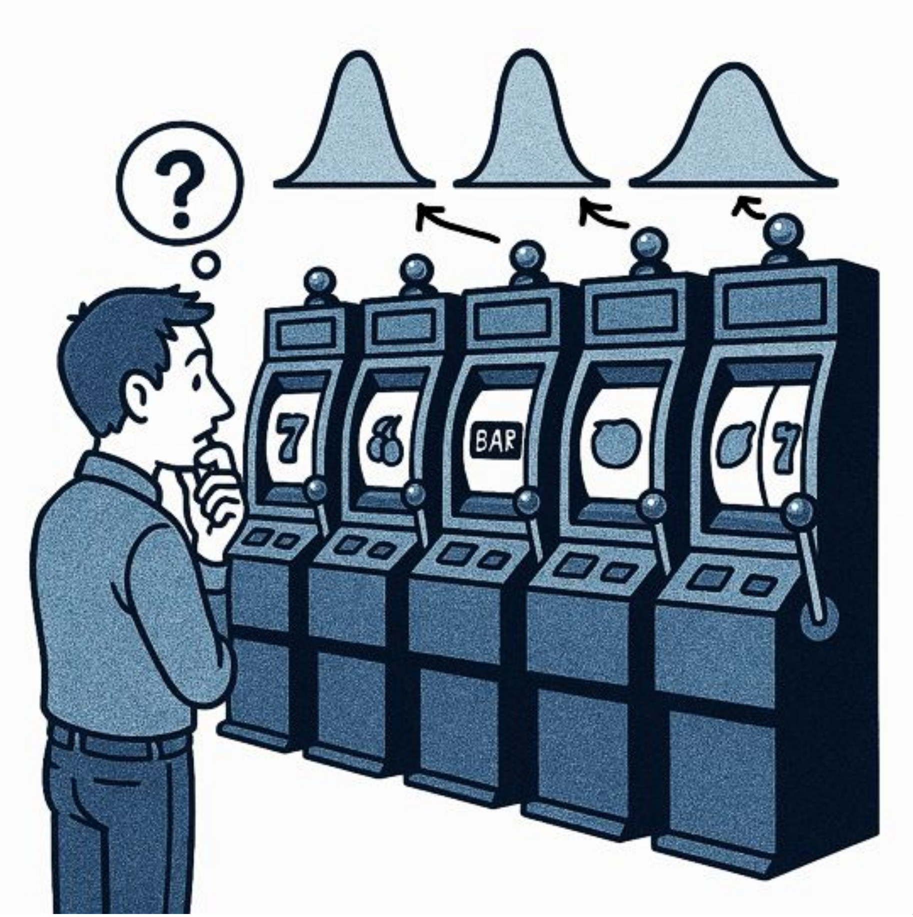
- Examples:
  - Linear bandit: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$.
  - Generalized linear bandit: $f(\mathbf{x}) = g(\mathbf{w}^\top \mathbf{x})$.
  - Gaussian process bandit: $f(\mathbf{x}) = \mathrm{GP}(\mathbf{x})$.
  - Neural bandit: $f(\mathbf{x}) = \mathrm{NN}(\mathbf{x})$.

Algorithm

Good if we can bound the regret!

- Assume I know the expected reward $\bar{r}_k$ given by each action, then the best strategy is **to always choose the best action a^* with the highest $\bar{r}^*$.**
  - But I don't know…
- We use ***(cumulative) regret*** to measure how good a bandit algorithm is:

$$\rho_T = \mathbb{E}\left[\sum_{t=1}^{T} R_{a^*} - \sum_{t=1}^{T} R_{a_t}\right]$$
$$= \sum_{t=1}^{T} \left(\bar{r}^* - \bar{r}_{a_t}\right).$$

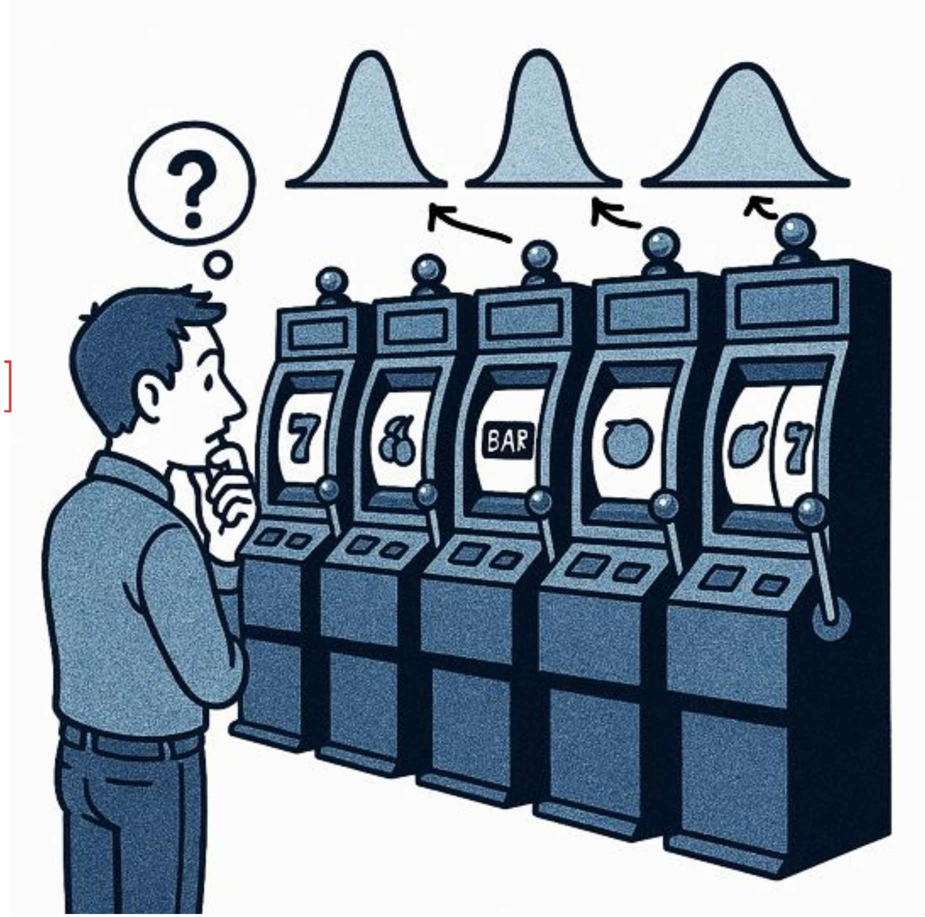


Algorithm

Good if we can bound the regret!

- Assume I know the expected reward $\bar{r}_k$ given by each action, then the best strategy is **to ~~always choose the best action a^* with the highest $\bar{r}^*$~~**.
    - But I don't know… $a_t^* = \arg\max_a \mathbb{E}[R_{\mathbf{x}}]$
- We use ***(cumulative) regret*** to measure how good a bandit algorithm is:

$$\rho_T = \mathbb{E}\left[\sum_{t=1}^{T} R_{a^*} - \sum_{t=1}^{T} R_{a_t}\right]$$
$$= \sum_{t=1}^{T} \left(\bar{r}^* - \bar{r}_{a_t}\right).$$



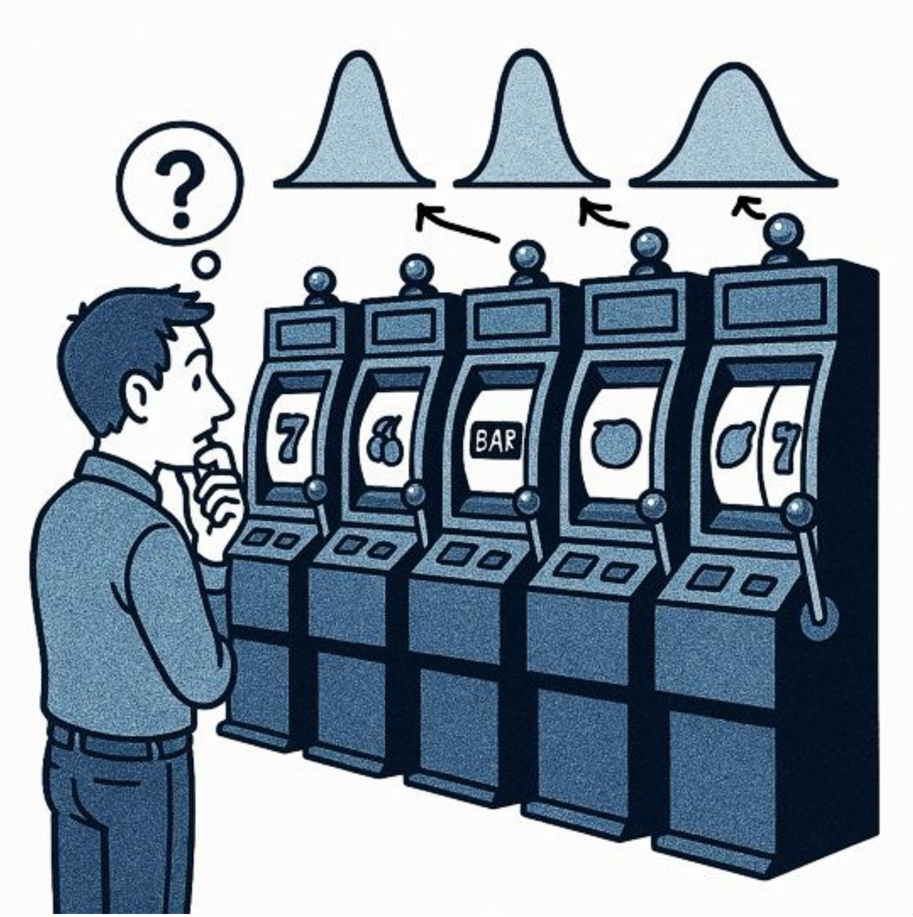
AY2024/25 Sem2
CS4246/CS5446

Algorithm

Good if we can bound the regret!

- Assume I know the expected reward $\bar{r}_k$ given by each action, then the best strategy is **to** ~~**always choose the best action** a^* **with the highest** $\bar{r}^*$~~.
    - But I don't know… $a_t^* = \arg\max_a \mathbb{E}[R_{\mathbf{x}}]$
- We use ***(cumulative) regret*** to measure how good a bandit algorithm is:

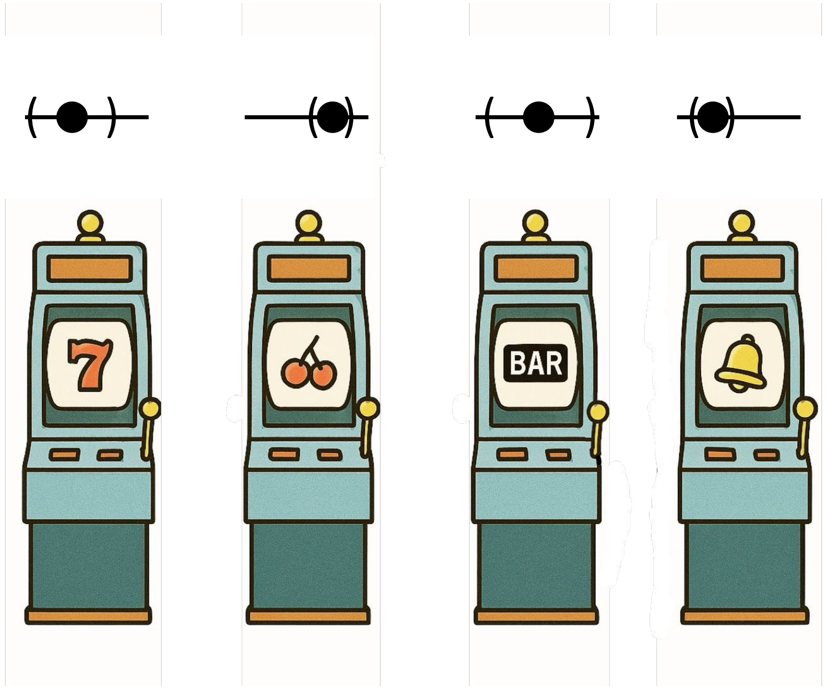
$$\rho_T = \mathbb{E}\left[\sum_{t=1}^{T} R_{\underbrace{a_t^*, \check{\mathbf{x}}_t}_{\mathbf{X}}} - \sum_{t=1}^{T} R_{a_t, \check{\mathbf{x}}_t}\right].$$



AY2024/25 Sem2
CS4246/CS5446

Algorithm: LinearUCB

- Each **weight** is associated with a **mean** and a **confidence term**.



More confident if we try more!

Algorithm: LinearUCB

- Each **weight** is associated with a **mean** and a **confidence term**.
- Confidence ellipsoid bound:

$$\Pr\left[\exists t, \|\hat{\mathbf{w}}_t - \mathbf{w}^*\|_{\mathrm{M}} \geq \nu\sqrt{d \log \frac{1 + tL/\lambda}{\delta}} + \sqrt{\lambda}\|\mathbf{w}^*\|\right] \leq \delta.$$

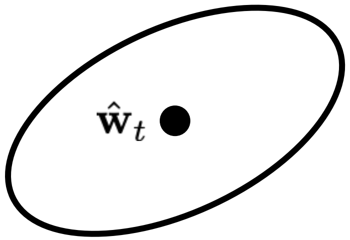
$$\Pr\left[\exists t, \left|\hat{\mathbf{w}}_t^\top \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix} - \mathbf{w}^{*\top} \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}\right| \geq \left(\nu\sqrt{d \log \frac{1 + tL/\lambda}{\delta}} + \sqrt{\lambda}\|\mathbf{w}^*\|\right) \cdot \sqrt{\begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}^\top \mathbf{G}_t^{-1} \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}}\right] \leq \delta.$$

  - v, L, λ are parameters specified in the assumptions.

Algorithm: LinearUCB

- Each **weight** is associated with a **mean** and a **confidence term**.
- Confidence ellipsoid bound:

$$\Pr\left[\exists t, \boxed{\|\hat{\mathbf{w}}_t - \mathbf{w}^*\|_{\mathrm{M}}} \geq \boxed{\nu\sqrt{d\log\frac{1+tL/\lambda}{\delta}} + \sqrt{\lambda}\|\mathbf{w}^*\|}\right] \leq \delta.$$



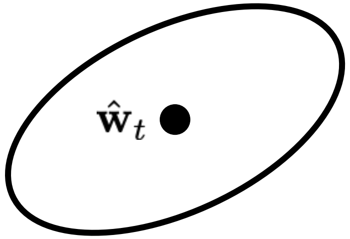
$$\Pr\left[\exists t, \left|\hat{\mathbf{w}}_t^\top \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix} - \mathbf{w}^{*\top} \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}\right| \geq \left(\nu\sqrt{d\log\frac{1+tL/\lambda}{\delta}} + \sqrt{\lambda}\|\mathbf{w}^*\|\right) \cdot \sqrt{\begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}^\top \mathbf{G}_t^{-1} \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}}\right] \leq \delta.$$

- v, L, λ are parameters specified in the assumptions.

Algorithm: LinearUCB

- Each **weight** is associated with a **mean** and a **confidence term**.
- Confidence ellipsoid bound:

$$\Pr\left[\exists t, \boxed{\|\hat{\mathbf{w}}_t - \mathbf{w}^*\|_{\mathrm{M}}} \geq \boxed{\nu\sqrt{d\log\frac{1+tL/\lambda}{\delta}} + \sqrt{\lambda}\|\mathbf{w}^*\|}\,\right] \leq \delta.$$

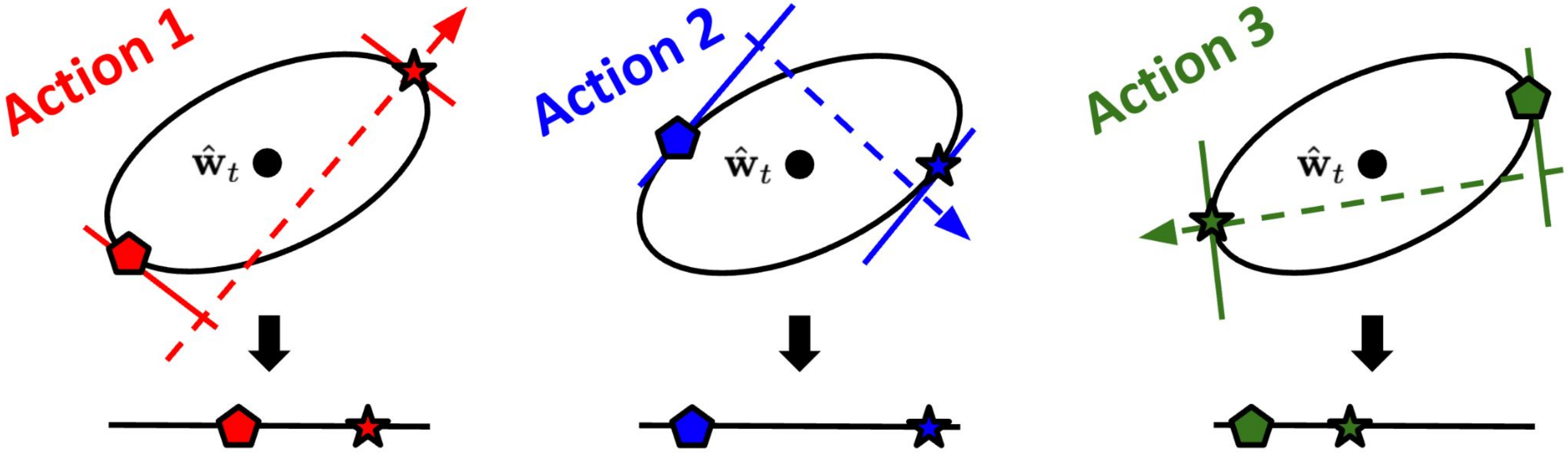


$$\Pr\left[\exists t, \left|\hat{\mathbf{w}}_t^\top \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix} - \mathbf{w}^{*\top} \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}\right| \geq \left(\nu\sqrt{d\log\frac{1+tL/\lambda}{\delta}} + \sqrt{\lambda}\|\mathbf{w}^*\|\right) \cdot \sqrt{\begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}^\top \mathbf{G}_t^{-1} \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}}\,\right] \leq \delta.$$

  - v, L, λ are parameters specified in the assumptions.
- At each round t, choose weight $\mathbf{w}$ from ellipsoid and action a_k that maximize

$$\mathrm{UCB}_{a_k}^t = \mathbf{w}^\top \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}.$$

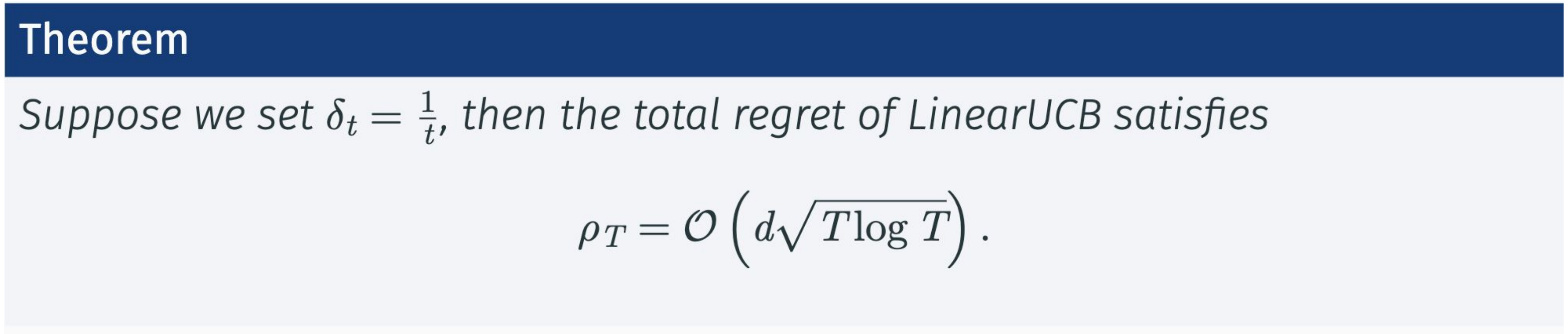
Algorithm: LinearUCB



- At each round t, choose weight $\mathbf{w}$ from ellipsoid and action a_k that maximize

$$\mathrm{UCB}_{a_k}^t = \mathbf{w}^\top \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}.$$

Algorithm: LinearUCB

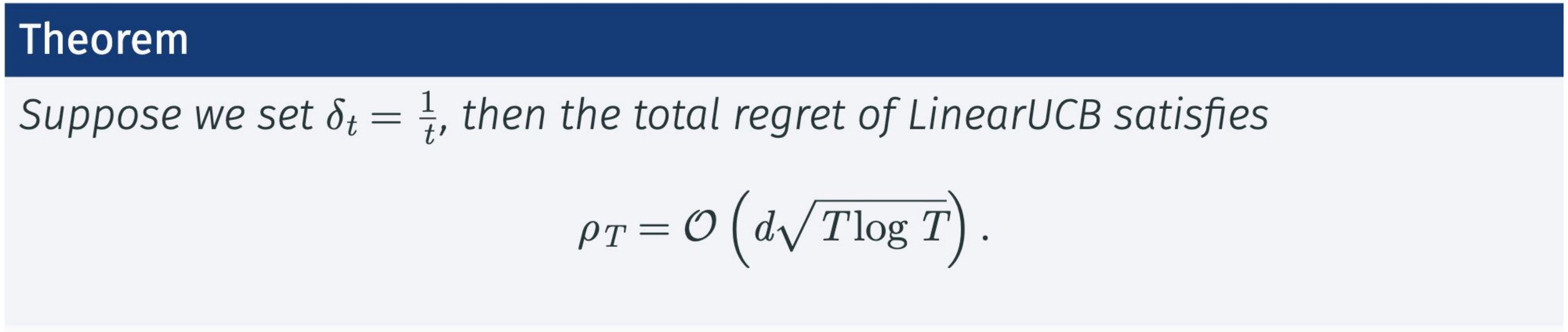


Theorem

Suppose we set $\delta_t = \frac{1}{t}$, then the total regret of LinearUCB satisfies

$$\rho_T = \mathcal{O}\left(d\sqrt{T\log T}\right).$$

Algorithm: LinearUCB



Theorem

Suppose we set $\delta_t = \frac{1}{t}$, then the total regret of LinearUCB satisfies

$$\rho_T = \mathcal{O}\left(d\sqrt{T \log T}\right).$$

- Suppose at some round we choose $\mathbf{w}_t$ and a_t. Then our estimated upper bound of reward is better than the **optimal** reward:

$$\mathrm{UCB}_{a_k}^{t}{}^{*} = \mathbf{w}_t^{\top} \begin{bmatrix} a_t \\ \breve{\mathbf{x}}_t \end{bmatrix} \geq \mathbf{w}^{*\top} \begin{bmatrix} a_t^{*} \\ \breve{\mathbf{x}}_t \end{bmatrix}.$$

Algorithm: LinearUCB

Theorem

Suppose we set $\delta_t = \frac{1}{t}$, then the total regret of LinearUCB satisfies

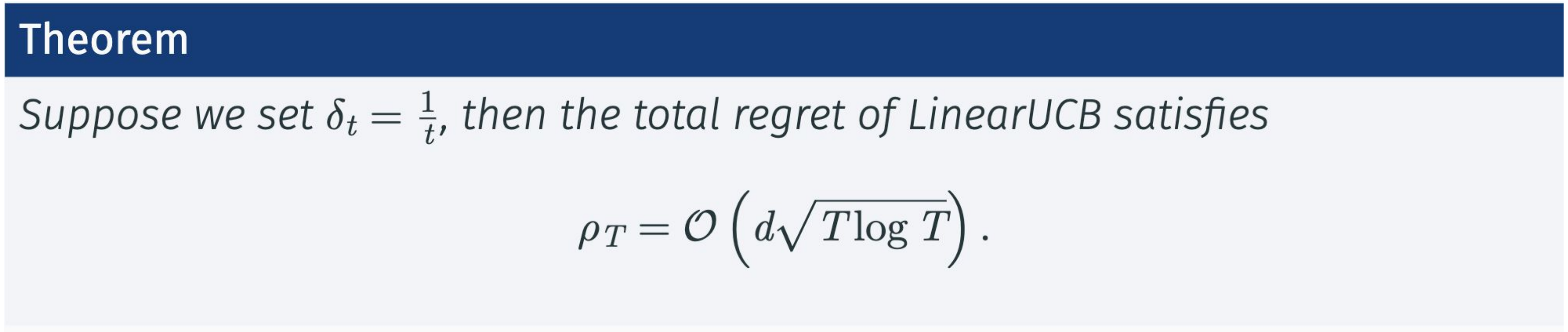
$$\rho_T = \mathcal{O}\left(d\sqrt{T \log T}\right).$$

- Suppose at some round we choose $\mathbf{w}_t$ and a_t. Then our estimated upper bound of reward is better than the **optimal** reward:

$$\mathrm{UCB}_{a_k}^{t}{}^{*} = \mathbf{w}_t^\top \begin{bmatrix} a_t \\ \breve{\mathbf{x}}_t \end{bmatrix} \geq \mathbf{w}^{*\top} \begin{bmatrix} a_t^* \\ \breve{\mathbf{x}}_t \end{bmatrix}.$$

- From the confidence ellipsoid bound, distance between $\mathrm{UCB}_{a_k}^{t}{}^{*}$ and our **actual** reward $\mathbf{w}^{*\top} \begin{bmatrix} a_t \\ \breve{\mathbf{x}}_t \end{bmatrix}$ is bounded.

Algorithm: LinearUCB



Theorem

Suppose we set $\delta_t = \frac{1}{t}$, then the total regret of LinearUCB satisfies

$$\rho_T = \mathcal{O}\left(d\sqrt{T \log T}\right).$$

- Suppose at some round we choose $\mathbf{w}_t$ and a_t. Then our estimated upper bound of reward is better than the **optimal** reward:

$$\mathrm{UCB}_{a_k}^{t}{}^{*} = \mathbf{w}_t^\top \begin{bmatrix} a_t \\ \breve{\mathbf{x}}_t \end{bmatrix} \geq \mathbf{w}^{*\top} \begin{bmatrix} a_t^* \\ \breve{\mathbf{x}}_t \end{bmatrix}.$$

- From the confidence ellipsoid bound, distance between $\mathrm{UCB}_{a_k}^{t}{}^{*}$ and our **actual** reward $\mathbf{w}^{*\top} \begin{bmatrix} a_t \\ \breve{\mathbf{x}}_t \end{bmatrix}$ is bounded.

So regret is bounded!

A Bayesian View: Bayesian Contextual Bandit

- Model parameters now follow a **distribution** (i.e., our belief).
- Whenever we try a new action, our belief is updated using Bayes’ rule:

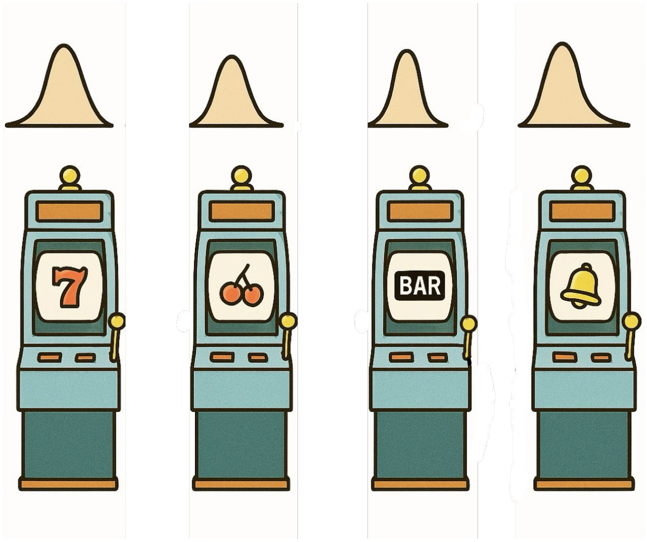
$$p(\mathbf{w}|r_{a_t,\breve{\mathbf{x}}_t}) = \frac{p(\mathbf{w})p(r_{a_t,\breve{\mathbf{x}}_t}|\mathbf{w})}{p(r_{a_t,\breve{\mathbf{x}}_t})}.$$

- Goal: Minimize ***Bayesian regret***:

$$\mathbb{E}_{\text{prior}}[\rho_T].$$

Algorithm: LinearTS

- Model parameters now follow a **distribution** (i.e., our belief).
- At each round t, we randomly sample a weight $\mathbf{w}$ from its distribution and choose the action that maximizes the estimated reward: $\mathbf{w}^\top \begin{bmatrix} a_k \\ \breve{\mathbf{x}}_t \end{bmatrix}$.

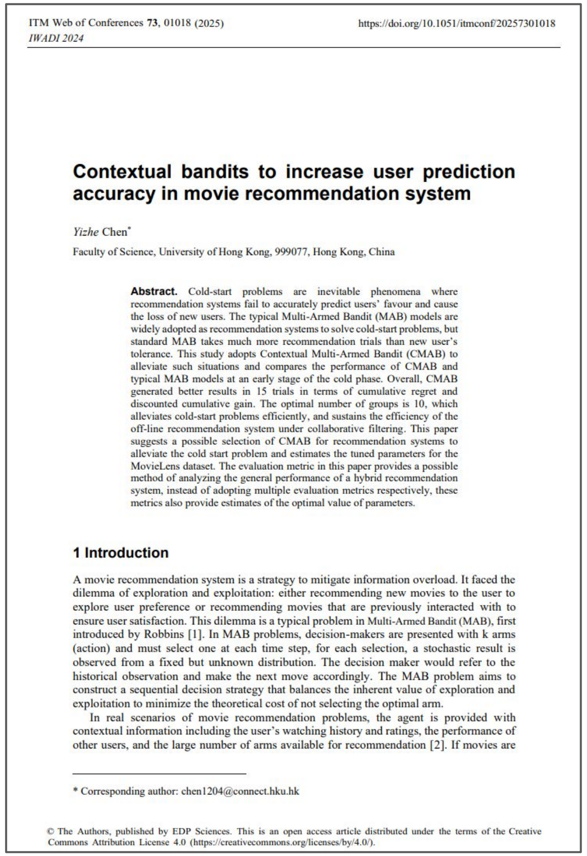


AY2024/25 Sem2
CS4246/CS5446

Implementation based on Paper

Contextual bandits to increase user prediction accuracy in movie recommendation system. Yizhe Chen (2025)

- Utilizes **Contextual Bandit** to make movie recommendation
- makes distinction between ***online*** and ***offline*** recommendations to mitigate cold-start problem which is usually encountered by conventional recommendation system.
- The *offline* recommendation uses **collaborative filtering** which leverages knowledge about the user based on similarity with other users to create recommendations.
- This *offline* recommendations does encounter the ***cold-start* problem**, as we might expect.



ITM Web of Conferences **73**, 01018 (2025)
IWADI 2024
https://doi.org/10.1051/itmconf/20257301018

Contextual bandits to increase user prediction accuracy in movie recommendation system

Yizhe Chen*

Faculty of Science, University of Hong Kong, 999077, Hong Kong, China

Abstract. Cold-start problems are inevitable phenomena where recommendation systems fail to accurately predict users' favour and cause the loss of new users. The typical Multi-Armed Bandit (MAB) models are widely adopted as recommendation systems to solve cold-start problems, but standard MAB takes much more recommendation trials than new user's tolerance. This study adopts Contextual Multi-Armed Bandit (CMAB) to alleviate such situations and compares the performance of CMAB and typical MAB models at an early stage of the cold phase. Overall, CMAB generated better results in 15 trials in terms of cumulative regret and discounted cumulative gain. The optimal number of groups is 10, which alleviates cold-start problems efficiently, and sustains the efficiency of the off-line recommendation system under collaborative filtering. This paper suggests a possible selection of CMAB for recommendation systems to alleviate the cold start problem and estimates the tuned parameters for the MovieLens dataset. The evaluation metric in this paper provides a possible method of analyzing the general performance of a hybrid recommendation system, instead of adopting multiple evaluation metrics respectively, these metrics also provide estimates of the optimal value of parameters.

1 Introduction

A movie recommendation system is a strategy to mitigate information overload. It faced the dilemma of exploration and exploitation: either recommending new movies to the user to explore user preference or recommending movies that are previously interacted with to ensure user satisfaction. This dilemma is a typical problem in Multi-Armed Bandit (MAB), first introduced by Robbins [1]. In MAB problems, decision-makers are presented with k arms (action) and must select one at each time step, for each selection, a stochastic result is observed from a fixed but unknown distribution. The decision maker would refer to the historical observation and make the next move accordingly. The MAB problem aims to construct a sequential decision strategy that balances the inherent value of exploration and exploitation to minimize the theoretical cost of not selecting the optimal arm.

In real scenarios of movie recommendation problems, the agent is provided with contextual information including the user's watching history and ratings, the performance of other users, and the large number of arms available for recommendation [2]. If movies are

* Corresponding author: chen1204@connect.hku.hk

source: https://www.itm-conferences.org/articles/itmconf/pdf/2025/04/itmconf_iwadi2024_01018.pdf

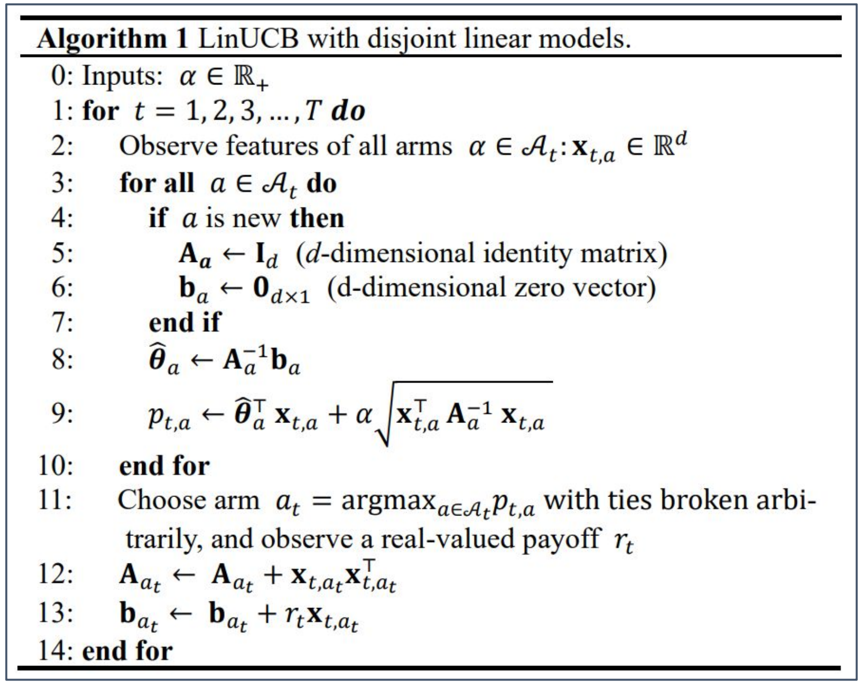
Online Recommendation

- The ***online*** recommendation uses Contextual Bandit to provide the system with context about the user with minimum data (cold users).

- The online recommendation is intended to **replace the early stage of collaborative filtering** until users have enough data which **patches the cold-start** problem.

- Utilizes **LinUCB (linear disjoint models)** to make movie recommendation.

- In the paper, Chen also compared the performance between the LinUCB contextual bandit and other multi-armed strategies.



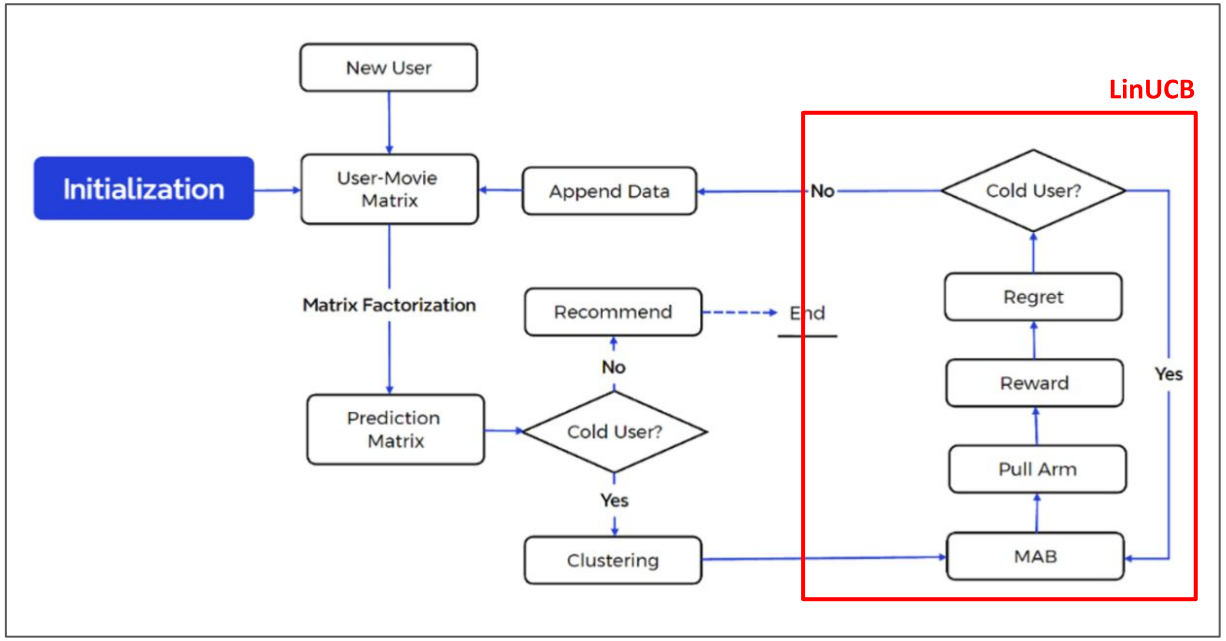
Algorithm 1 LinUCB with disjoint linear models.

```
Inputs: $\alpha \in \mathbb{R}_+$
for $t = 1, 2, 3, \ldots, T$ do
    Observe features of all arms $\alpha \in \mathcal{A}_t: \mathbf{x}_{t,a} \in \mathbb{R}^d$
    for all $a \in \mathcal{A}_t$ do
        if $a$ is new then
            $\mathbf{A}_a \leftarrow \mathbf{I}_d$ ($d$-dimensional identity matrix)
            $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
        end if
        $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
        $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}$
    end for
    Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff $r_t$
    $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
    $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
end for
```

source: https://www.itm-conferences.org/articles/itmconf/pdf/2025/04/itmconf_iwadi2024_01018.pdf

Cold Start Problem in Recommendation System

- Chen's proposed solution is to predict whether the user is "Cold" or not.
- The prediction results will decide whether the user will receive an ***online*** or ***offline*** recommendation.
- The process of ***online*** recommendations with ***LinUCB*** Contextual Bandit will run repetitively as long as the user is still "Cold".



source: https://www.itm-conferences.org/articles/itmconf/pdf/2025/04/itmconf_iwadi2024_01018.pdf

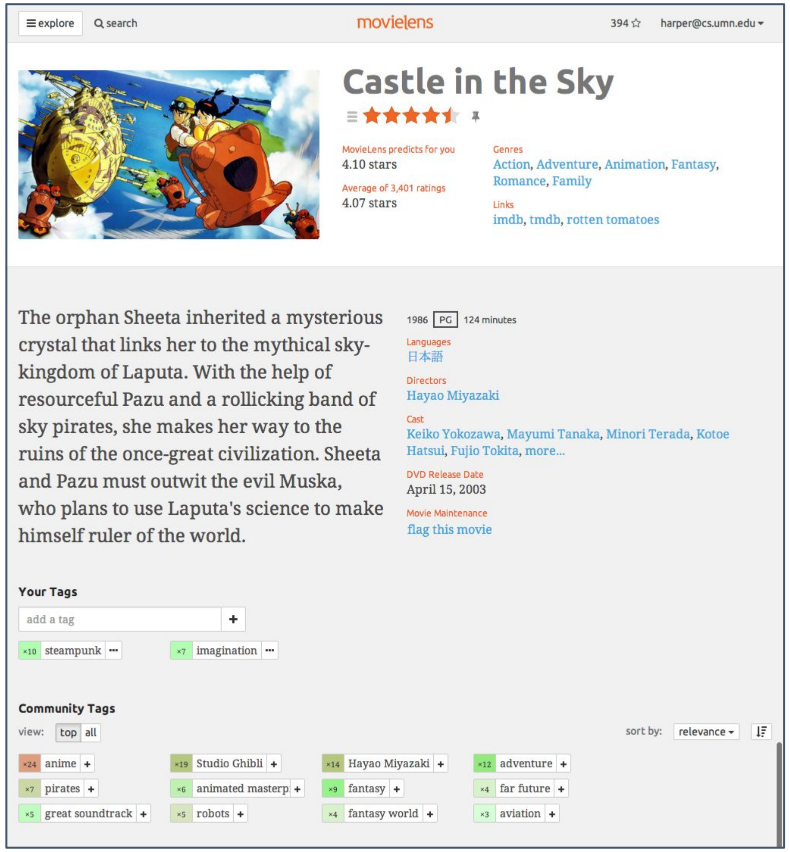
Dataset Description

- As in contextual bandit, the agent is allowed to have partial knowledge about the environment in order to reduce the needs for exploration.

- Dataset: MovieLens (Non-commercial, personalized movie recommendations).

- Chen utilizes 79 context observed from the dataset:
  - User Age, Gender, Occupation
  - Movie Genre, Tag, Average Rating
  - etc.

- Vectorized as feature vector used for the ***LinUCB***.



Castle in the Sky

MovieLens predicts for you
4.10 stars

Average of 3,401 ratings
4.07 stars

Genres
Action, Adventure, Animation, Fantasy, Romance, Family

Links
imdb, tmdb, rotten tomatoes

The orphan Sheeta inherited a mysterious crystal that links her to the mythical sky-kingdom of Laputa. With the help of resourceful Pazu and a rollicking band of sky pirates, she makes her way to the ruins of the once-great civilization. Sheeta and Pazu must outwit the evil Muska, who plans to use Laputa's science to make himself ruler of the world.

1986 PG 124 minutes

Languages
日本語

Directors
Hayao Miyazaki

Cast
Keiko Yokozawa, Mayumi Tanaka, Minori Terada, Kotoe Hatsui, Fujio Tokita, more...

DVD Release Date
April 15, 2003

Movie Maintenance
flag this movie

Your Tags

add a tag +

×10 steampunk ···
×7 imagination ···

Community Tags

view: top all
sort by: relevance

×24 anime +
×7 pirates +
×5 great soundtrack +
×19 Studio Ghibli +
×6 animated masterp +
×5 robots +
×14 Hayao Miyazaki +
×9 fantasy +
×4 fantasy world +
×12 adventure +
×4 far future +
×3 aviation +

source: https://movielens.org/

Our Methodology

- For this project, we limited our research scope to focus on the implementation of LinUCB contextual bandits and compare it with contextual epsilon-greedy bandits.

- Initially, we tried to replicate Chen's approach which uses the user-movie-rating pairs clustering as the contextual vector.

- However, this approach includes user-movie-rating data into clustering. This approach feeds information about how users will rate certain movies which leaks future predictions. Therefore, it causes the problem to not purely be a cold-start problem.

- After further discussion and consideration, we decided to use the user's demographic information and the movie's genre as the context vector.

Our Methodology

- We suspect that Chen’s NDCG matrix score is heavily influenced by the Collaborative Filtering as the number shows an outstanding score with minimum variance rate.

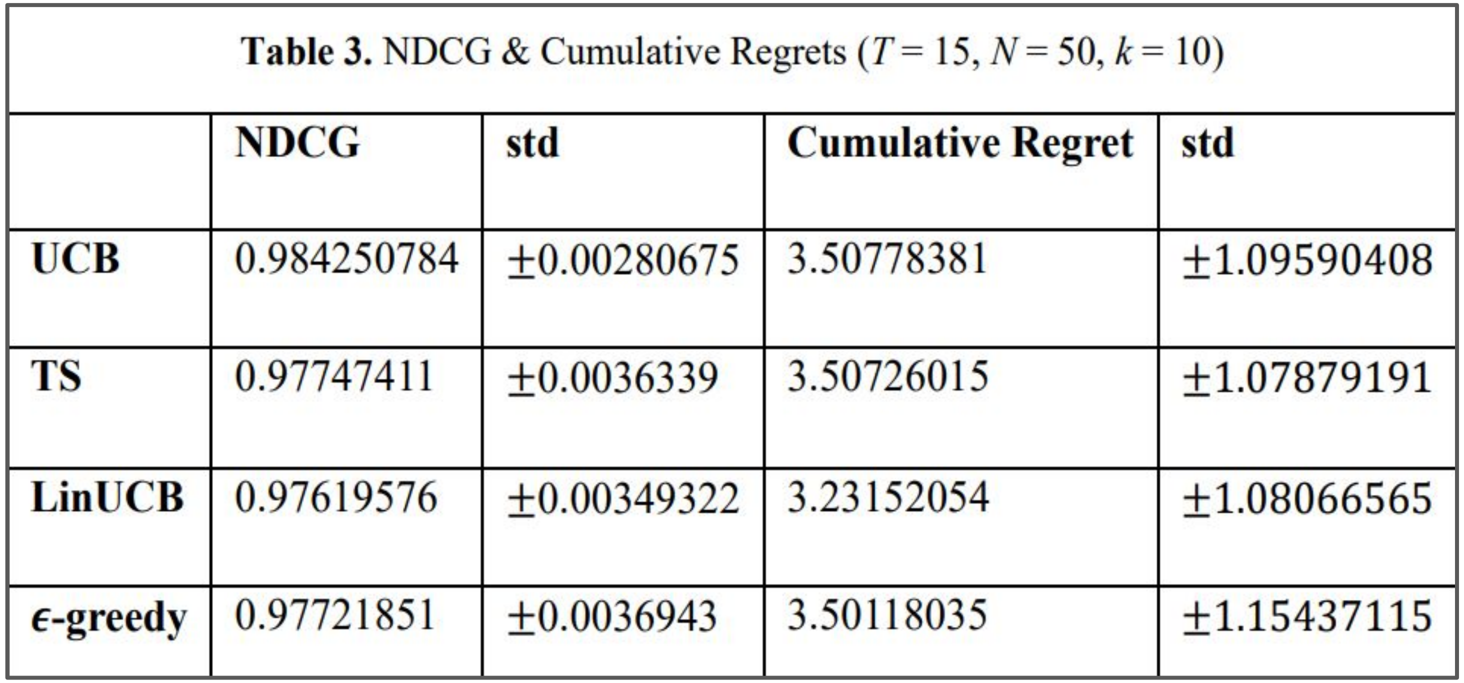


Table 3. NDCG & Cumulative Regrets ($T = 15$, $N = 50$, $k = 10$)

	NDCG	**std**	**Cumulative Regret**	**std**
UCB	0.984250784	±0.00280675	3.50778381	±1.09590408
TS	0.97747411	±0.0036339	3.50726015	±1.07879191
LinUCB	0.97619576	±0.00349322	3.23152054	±1.08066565
ϵ-greedy	0.97721851	±0.0036943	3.50118035	±1.15437115

source: https://www.itm-conferences.org/articles/itmconf/pdf/2025/04/itmconf_iwadi2024_01018.pdf

Results: Cumulative Regret

N: Number of movie clusters -> Number of arms

	N = 3	N = 5	N = 10	N = 20	N = 50
LinUCB					
α = 0.001	87.80	174.80	281.20	135.80	177.80
α = 0.5	96.20	218.00	373.00	342.80	711.00
α = 1	107.60	267.60	535.40	691.80	1695.40
Contextual ε-greedy					
ε = 0.001	94.20	179.20	284.00	137.80	185.20
ε = 0.5	2409.00	3225.00	3575.40	3576.60	3738.00
ε = 0.1	4784.00	6248.40	6804.60	7003.80	7133.00

Results: Cumulative Regret

N: Number of movie clusters -> Number of arms

	N = 3	N = 5	N = 10	N = 20	N = 50
LinUCB					
α = 0.001	87.80	174.80	281.20	135.80	177.80
α = 0.5	96.20	218.00	373.00	342.80	711.00
α = 1	107.60	267.60	535.40	691.80	1695.40
Contextual ε-greedy					
ε = 0.001	94.20	179.20	284.00	137.80	185.20
ε = 0.5	2409.00	3225.00	3575.40	3576.60	3738.00
ε = 0.1	4784.00	6248.40	6804.60	7003.80	7133.00

Results: Cumulative Regret

N: Number of movie clusters -> Number of arms

	N = 3	N = 5	N = 10	N = 20	N = 50
LinUCB					
α = 0.001	87.80	174.80	281.20	135.80	177.80
α = 0.5	96.20	218.00	373.00	342.80	711.00
α = 1	107.60	267.60	535.40	691.80	1695.40
Contextual ε-greedy					
ε = 0.001	94.20	179.20	284.00	137.80	185.20
ε = 0.5	2409.00	3225.00	3575.40	3576.60	3738.00
ε = 0.1	4784.00	6248.40	6804.60	7003.80	7133.00

Results: Cumulative Regret

N: Number of movie clusters -> Number of arms

	N = 3	N = 5	N = 10	N = 20	N = 50
LinUCB					
α = 0.001	87.80	174.80	281.20	135.80	177.80
α = 0.5	96.20	218.00	373.00	342.80	711.00
α = 1	107.60	267.60	535.40	691.80	1695.40
Contextual ε-greedy					
ε = 0.001	94.20	179.20	284.00	137.80	185.20
ε = 0.5	2409.00	3225.00	3575.40	3576.60	3738.00
ε = 0.1	4784.00	6248.40	6804.60	7003.80	7133.00

Results: Cumulative Regret

N: Number of movie clusters -> Number of arms

	N = 3	N = 5	N = 10	N = 20	N = 50
LinUCB					
α = 0.001	87.80	174.80	281.20	135.80	177.80
α = 0.5	96.20	218.00	373.00	342.80	711.00
α = 1	107.60	267.60	535.40	691.80	1695.40
Contextual ε-greedy					
ε = 0.001	94.20	179.20	284.00	137.80	185.20
ε = 0.5	2409.00	3225.00	3575.40	3576.60	3738.00
ε = 0.1	4784.00	6248.40	6804.60	7003.80	7133.00

Results: Cumulative Regret

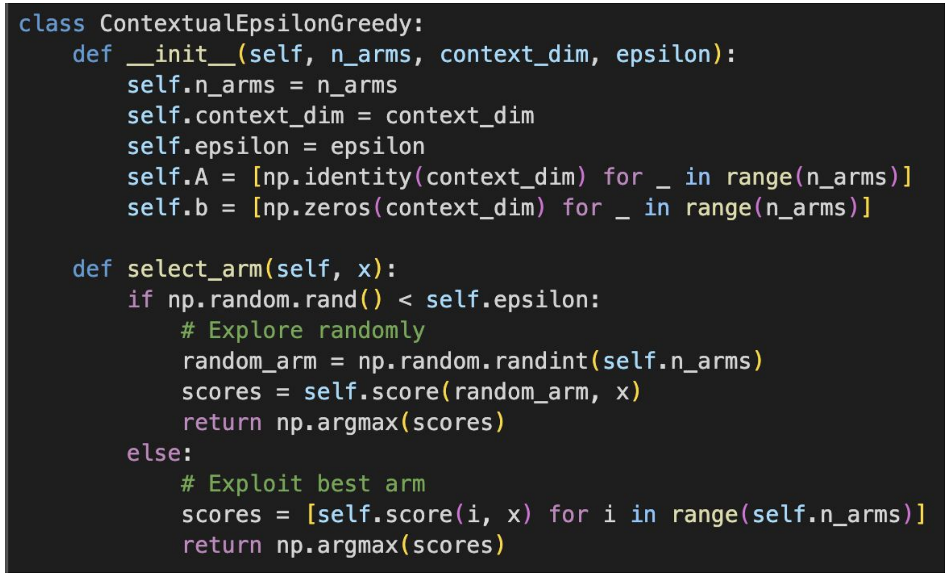
N: Number of movie clusters -> Number of arms

	N = 3	N = 5	N = 10	N = 20	N = 50
LinUCB					
α = 0.001	87.80	174.80	281.20	135.80	177.80
α = 0.5	96.20	218.00	373.00	342.80	711.00
α = 1	107.60	267.60	535.40	691.80	1695.40
Contextual ε-greedy					
ε = 0.001	94.20	179.20	284.00	137.80	185.20
ε = 0.5	2409.00	3225.00	3575.40	3576.60	3738.00
ε = 0.1	4784.00	6248.40	6804.60	7003.80	7133.00

Results: Cumulative Regret

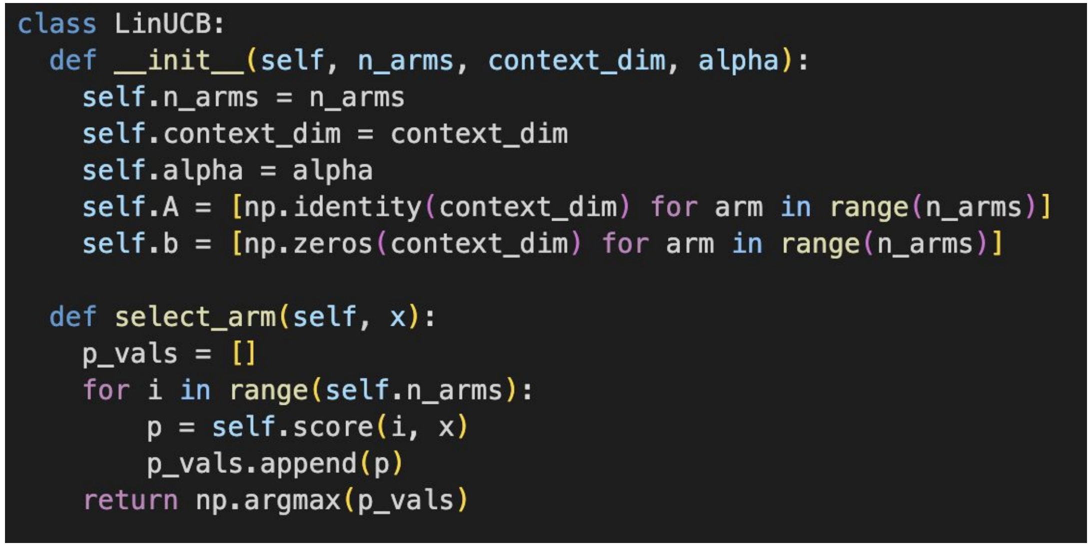
LinUCB achieved a lower cumulative regret compared to contextual epsilon greedy.

- LinUCB selects an arm based on the highest Upper Confidence Bound (UCB)
- Contextual epsilon greedy selects arms at random



```python
class ContextualEpsilonGreedy:
    def __init__(self, n_arms, context_dim, epsilon):
        self.n_arms = n_arms
        self.context_dim = context_dim
        self.epsilon = epsilon
        self.A = [np.identity(context_dim) for _ in range(n_arms)]
        self.b = [np.zeros(context_dim) for _ in range(n_arms)]

    def select_arm(self, x):
        if np.random.rand() < self.epsilon:
            # Explore randomly
            random_arm = np.random.randint(self.n_arms)
            scores = self.score(random_arm, x)
            return np.argmax(scores)
        else:
            # Exploit best arm
            scores = [self.score(i, x) for i in range(self.n_arms)]
            return np.argmax(scores)
```



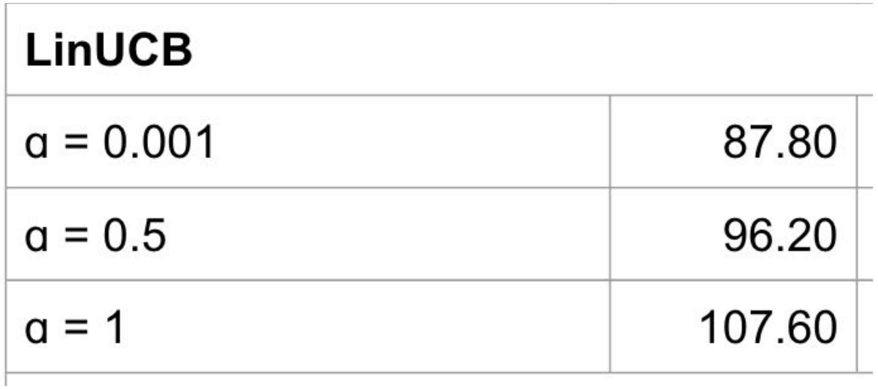
```python
class LinUCB:
  def __init__(self, n_arms, context_dim, alpha):
    self.n_arms = n_arms
    self.context_dim = context_dim
    self.alpha = alpha
    self.A = [np.identity(context_dim) for arm in range(n_arms)]
    self.b = [np.zeros(context_dim) for arm in range(n_arms)]

  def select_arm(self, x):
    p_vals = []
    for i in range(self.n_arms):
        p = self.score(i, x)
        p_vals.append(p)
    return np.argmax(p_vals)
```

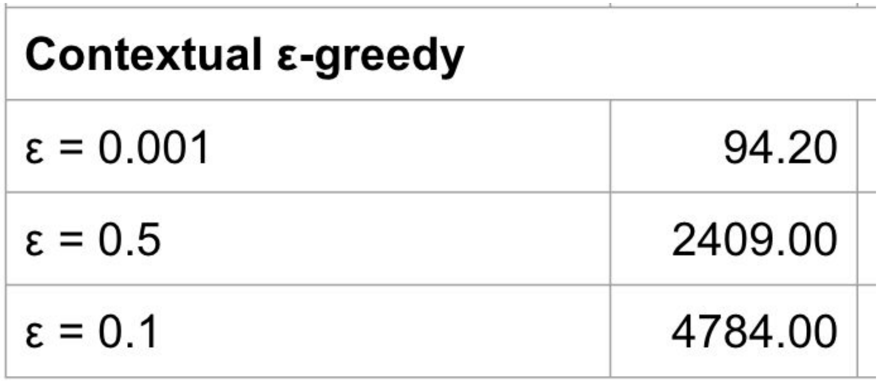
Results Analysis

LinUCB had a lower difference of cumulative regrets when switching from a low exploration rate to a higher exploration rate, whereas **contextual epsilon greedy had a more drastic difference**.

- LinUCB selects an arm based on the highest Upper Confidence Bound (UCB)
- Contextual epsilon greedy selects arms at random

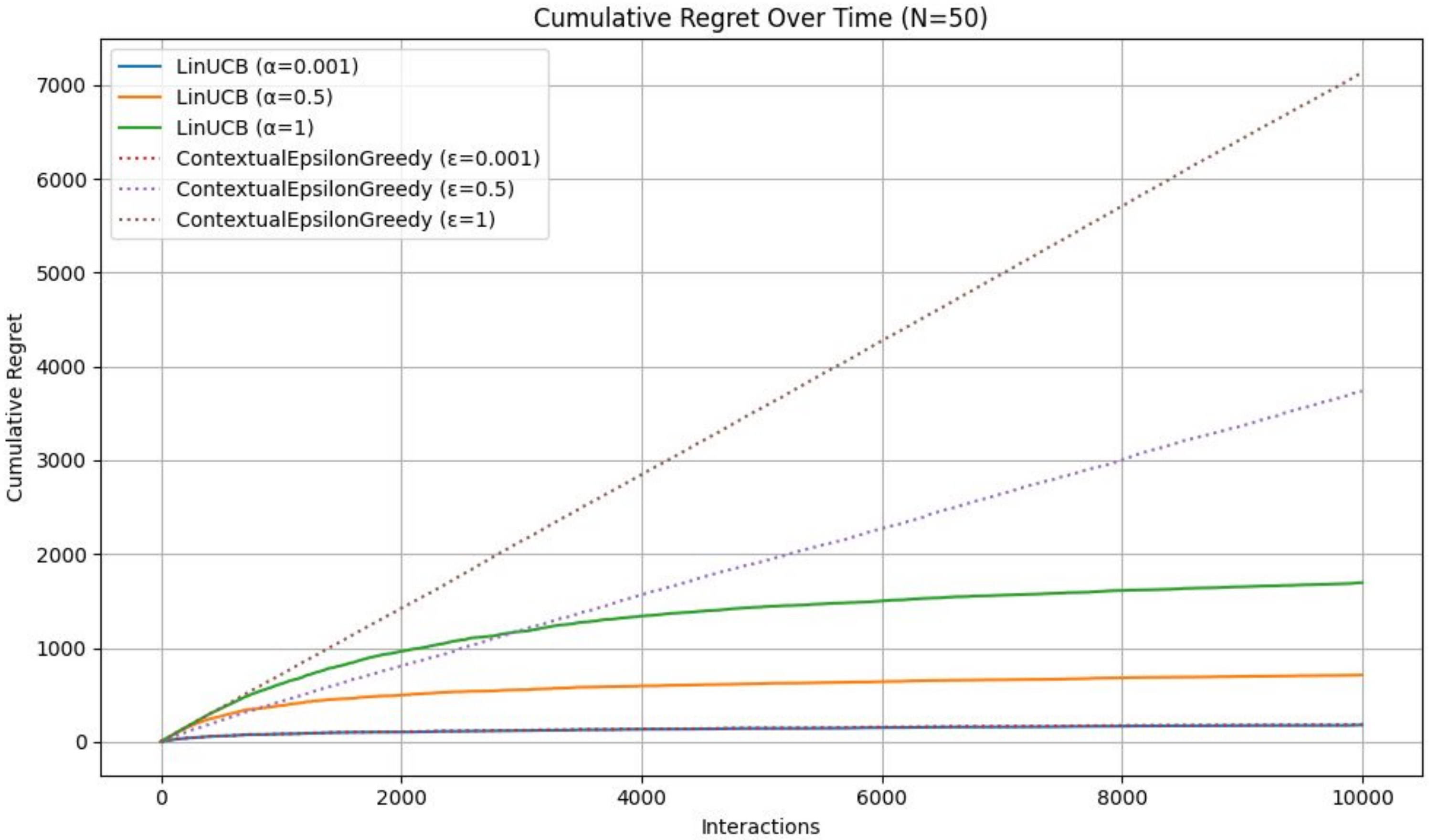


LinUCB	
α = 0.001	87.80
α = 0.5	96.20
α = 1	107.60



Contextual ε-greedy	
ε = 0.001	94.20
ε = 0.5	2409.00
ε = 0.1	4784.00

Results: Cumulative Regret Over Time



LinUCB makes more calculated predictions and adapts over time
Contextual epsilon greedy uses randomized predictions

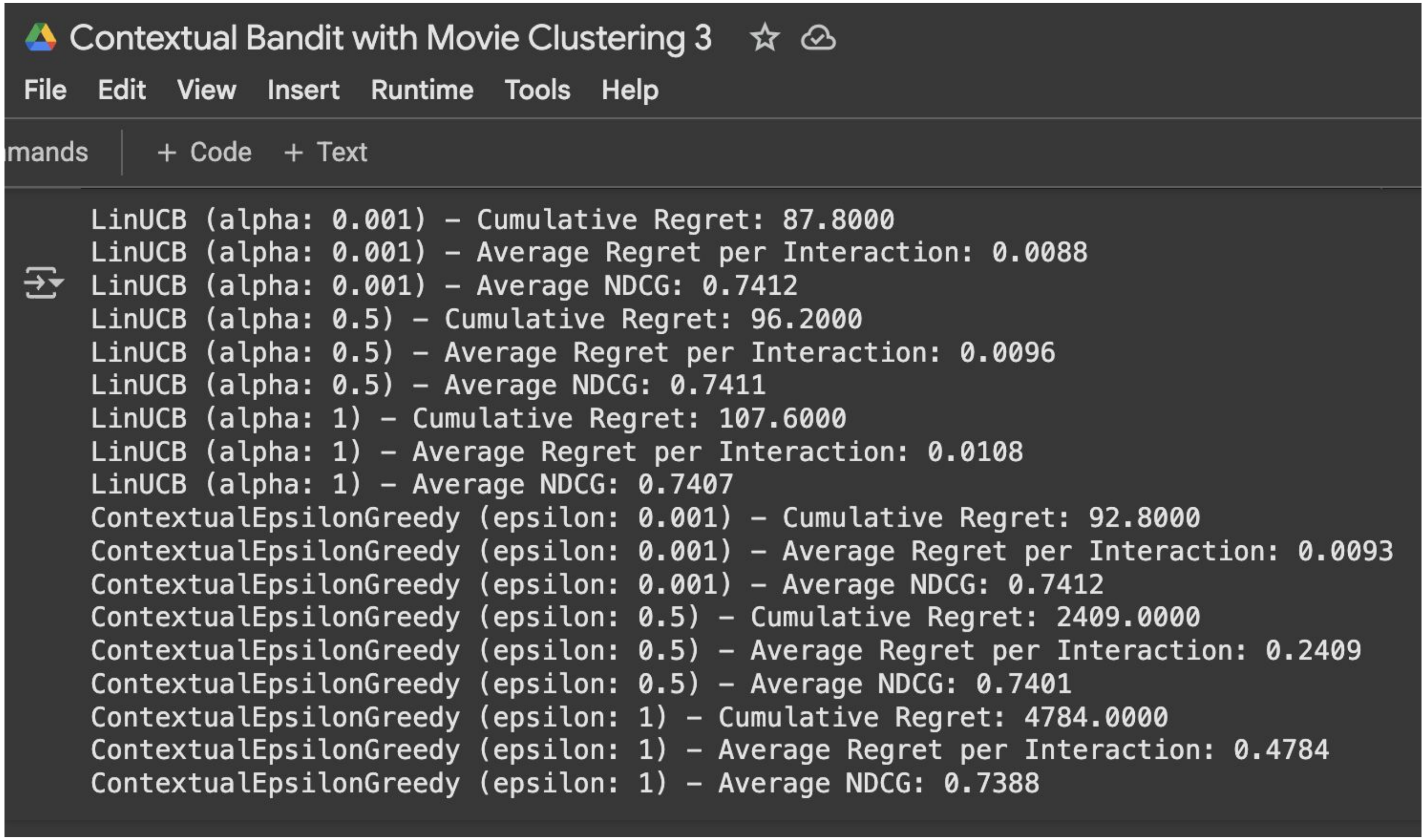
AY2024/25 Sem2
CS4246/CS5446

Thank you

References

- Slivkins, A. (2019). Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, *12*(1-2), 1-286.
- Lecture slides on Multi-armed bandits by Cathy Wu.
- Abbasi-Yadkori, Y., Pál, D., & Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, *24*.
- Agrawal, S., & Goyal, N. (2013, May). Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning* (pp. 127-135). PMLR.
- Sutton, R. S., Barto, A. G., et al. (1998). Reinforcement learning: An introduction, volume 1. MIT press Cambridge.
- Auer, P. (2000). Using upper confidence bounds for online learning. In Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pages 270–279. IEEE.
- Scott, S. L. (2010). A modern Bayesian look at the multi-armed bandit. Applied Stochastic Models in Business and Industry, 26(6):639–658.
- Chen, Y. (2025). Contextual bandits to increase user prediction accuracy in movie recommendation system. ITM Web of Conferences, 73, 01018. https://doi.org/10.1051/itmconf/20257301018
- Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets. ACM Transactions on Interactive Intelligent Systems (TiiS), 5(4), 19. https://doi.org/10.1145/2827872

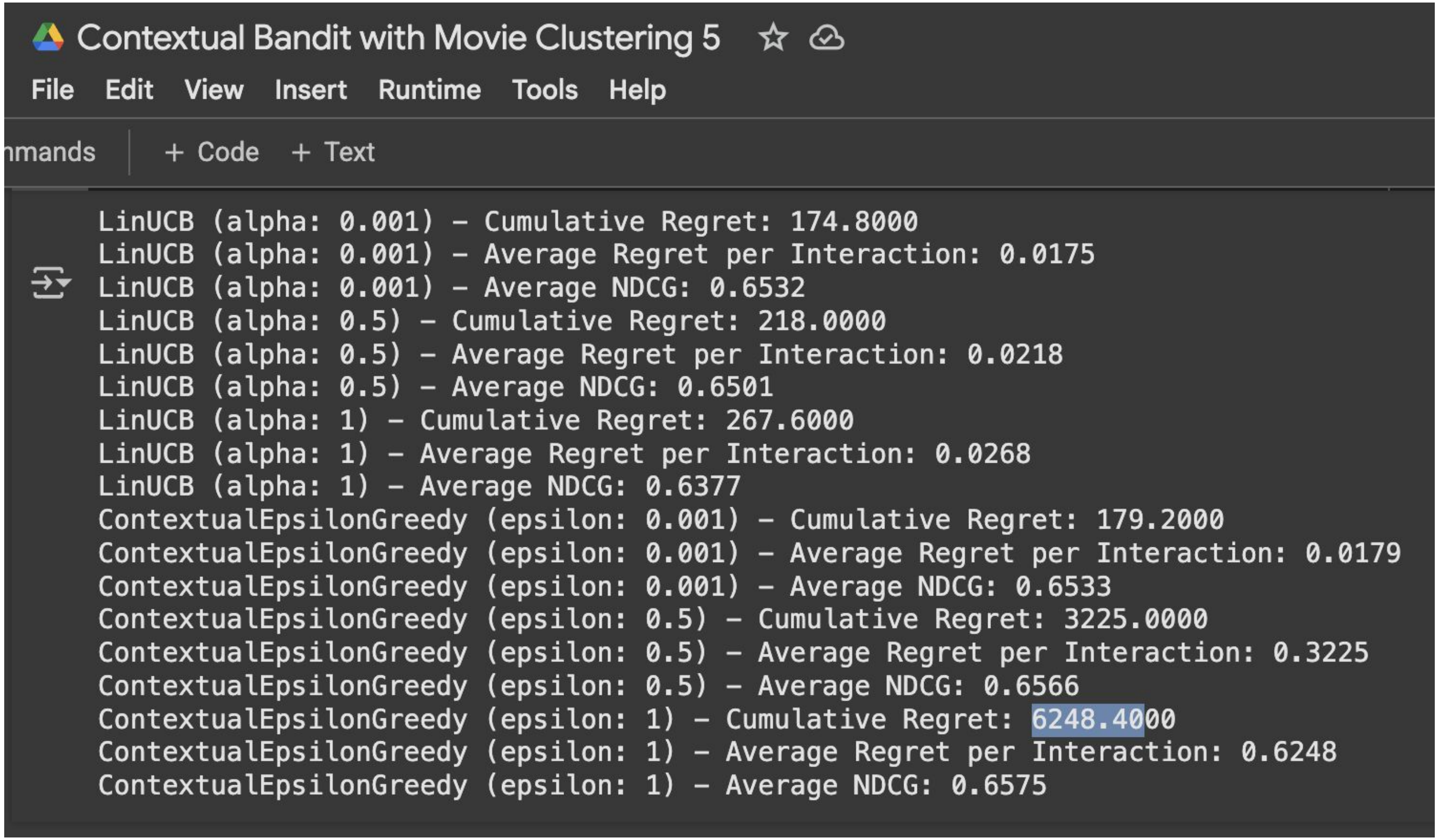
Appendix of results (in case needed)



Contextual Bandit with Movie Clustering 3
File Edit View Insert Runtime Tools Help
mands + Code + Text

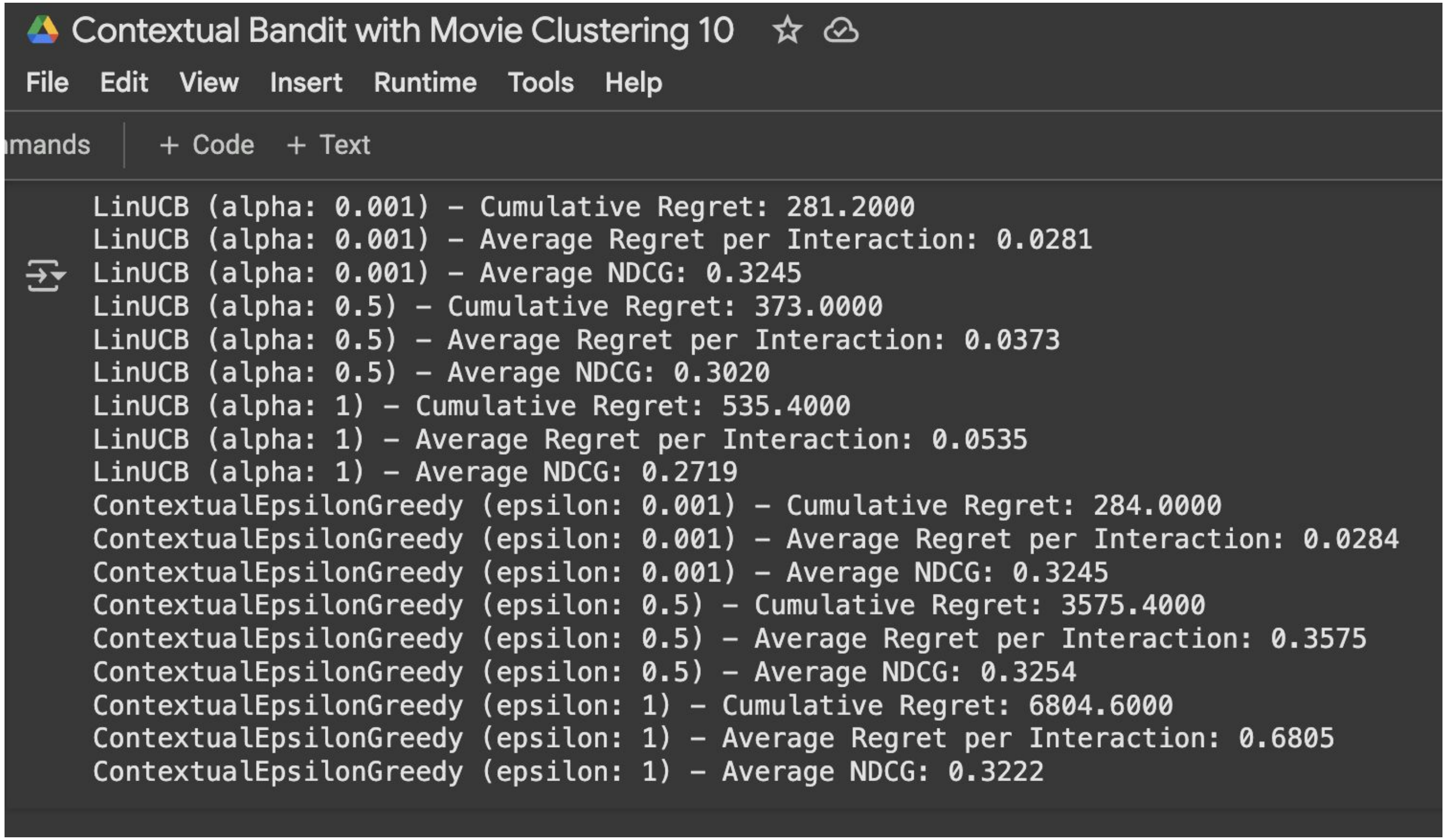
```
LinUCB (alpha: 0.001) – Cumulative Regret: 87.8000
LinUCB (alpha: 0.001) – Average Regret per Interaction: 0.0088
LinUCB (alpha: 0.001) – Average NDCG: 0.7412
LinUCB (alpha: 0.5) – Cumulative Regret: 96.2000
LinUCB (alpha: 0.5) – Average Regret per Interaction: 0.0096
LinUCB (alpha: 0.5) – Average NDCG: 0.7411
LinUCB (alpha: 1) – Cumulative Regret: 107.6000
LinUCB (alpha: 1) – Average Regret per Interaction: 0.0108
LinUCB (alpha: 1) – Average NDCG: 0.7407
ContextualEpsilonGreedy (epsilon: 0.001) – Cumulative Regret: 92.8000
ContextualEpsilonGreedy (epsilon: 0.001) – Average Regret per Interaction: 0.0093
ContextualEpsilonGreedy (epsilon: 0.001) – Average NDCG: 0.7412
ContextualEpsilonGreedy (epsilon: 0.5) – Cumulative Regret: 2409.0000
ContextualEpsilonGreedy (epsilon: 0.5) – Average Regret per Interaction: 0.2409
ContextualEpsilonGreedy (epsilon: 0.5) – Average NDCG: 0.7401
ContextualEpsilonGreedy (epsilon: 1) – Cumulative Regret: 4784.0000
ContextualEpsilonGreedy (epsilon: 1) – Average Regret per Interaction: 0.4784
ContextualEpsilonGreedy (epsilon: 1) – Average NDCG: 0.7388
```

Appendix of results (in case needed)


Contextual Bandit with Movie Clustering 5
File Edit View Insert Runtime Tools Help
nmands + Code + Text

```
LinUCB (alpha: 0.001) - Cumulative Regret: 174.8000
LinUCB (alpha: 0.001) - Average Regret per Interaction: 0.0175
LinUCB (alpha: 0.001) - Average NDCG: 0.6532
LinUCB (alpha: 0.5) - Cumulative Regret: 218.0000
LinUCB (alpha: 0.5) - Average Regret per Interaction: 0.0218
LinUCB (alpha: 0.5) - Average NDCG: 0.6501
LinUCB (alpha: 1) - Cumulative Regret: 267.6000
LinUCB (alpha: 1) - Average Regret per Interaction: 0.0268
LinUCB (alpha: 1) - Average NDCG: 0.6377
ContextualEpsilonGreedy (epsilon: 0.001) - Cumulative Regret: 179.2000
ContextualEpsilonGreedy (epsilon: 0.001) - Average Regret per Interaction: 0.0179
ContextualEpsilonGreedy (epsilon: 0.001) - Average NDCG: 0.6533
ContextualEpsilonGreedy (epsilon: 0.5) - Cumulative Regret: 3225.0000
ContextualEpsilonGreedy (epsilon: 0.5) - Average Regret per Interaction: 0.3225
ContextualEpsilonGreedy (epsilon: 0.5) - Average NDCG: 0.6566
ContextualEpsilonGreedy (epsilon: 1) - Cumulative Regret: 6248.4000
ContextualEpsilonGreedy (epsilon: 1) - Average Regret per Interaction: 0.6248
ContextualEpsilonGreedy (epsilon: 1) - Average NDCG: 0.6575
```

Appendix of results (in case needed)

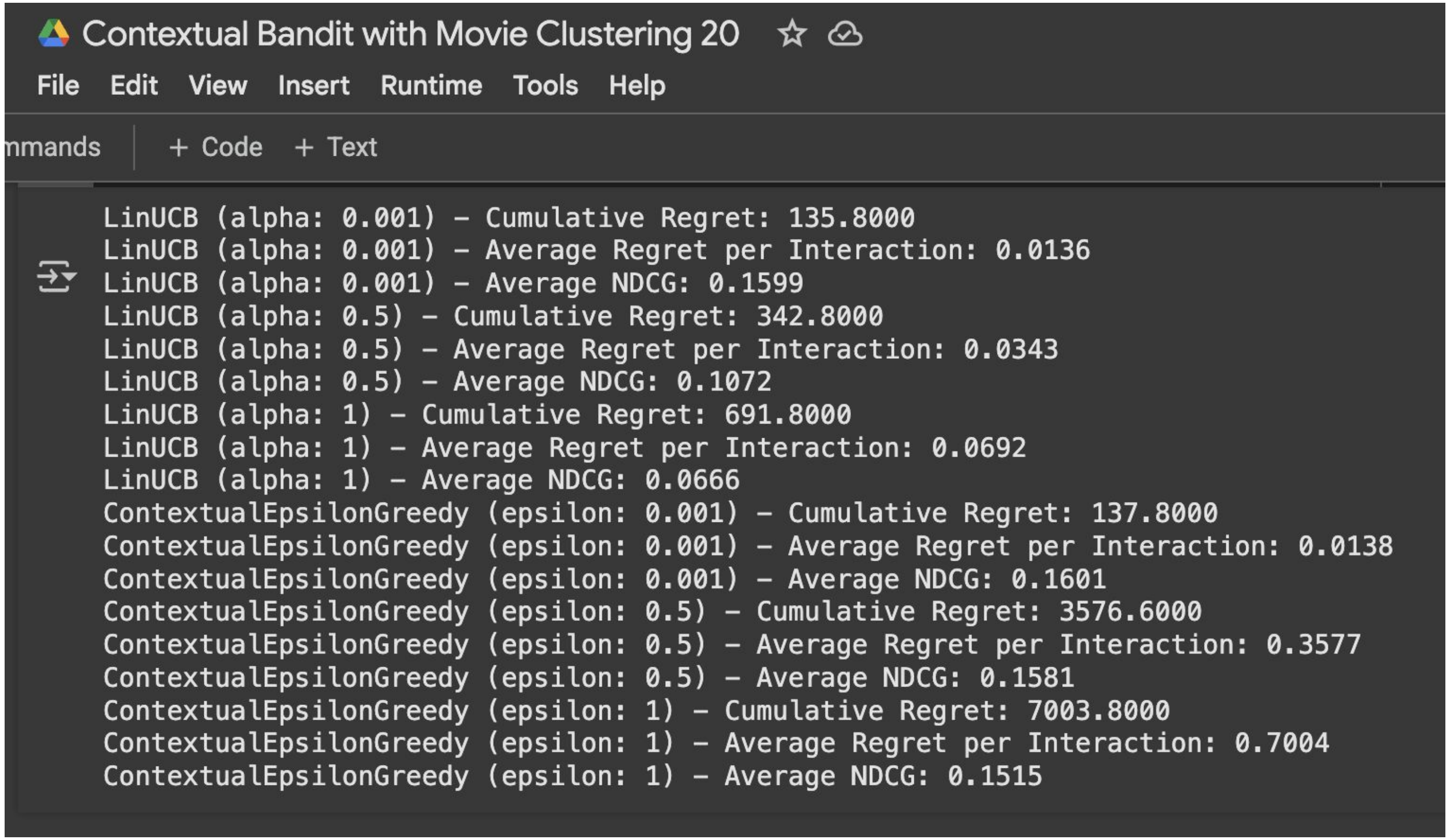


Contextual Bandit with Movie Clustering 10
File Edit View Insert Runtime Tools Help
mands + Code + Text

```
LinUCB (alpha: 0.001) – Cumulative Regret: 281.2000
LinUCB (alpha: 0.001) – Average Regret per Interaction: 0.0281
LinUCB (alpha: 0.001) – Average NDCG: 0.3245
LinUCB (alpha: 0.5) – Cumulative Regret: 373.0000
LinUCB (alpha: 0.5) – Average Regret per Interaction: 0.0373
LinUCB (alpha: 0.5) – Average NDCG: 0.3020
LinUCB (alpha: 1) – Cumulative Regret: 535.4000
LinUCB (alpha: 1) – Average Regret per Interaction: 0.0535
LinUCB (alpha: 1) – Average NDCG: 0.2719
ContextualEpsilonGreedy (epsilon: 0.001) – Cumulative Regret: 284.0000
ContextualEpsilonGreedy (epsilon: 0.001) – Average Regret per Interaction: 0.0284
ContextualEpsilonGreedy (epsilon: 0.001) – Average NDCG: 0.3245
ContextualEpsilonGreedy (epsilon: 0.5) – Cumulative Regret: 3575.4000
ContextualEpsilonGreedy (epsilon: 0.5) – Average Regret per Interaction: 0.3575
ContextualEpsilonGreedy (epsilon: 0.5) – Average NDCG: 0.3254
ContextualEpsilonGreedy (epsilon: 1) – Cumulative Regret: 6804.6000
ContextualEpsilonGreedy (epsilon: 1) – Average Regret per Interaction: 0.6805
ContextualEpsilonGreedy (epsilon: 1) – Average NDCG: 0.3222
```

AY2024/25 Sem2
CS4246/CS5446

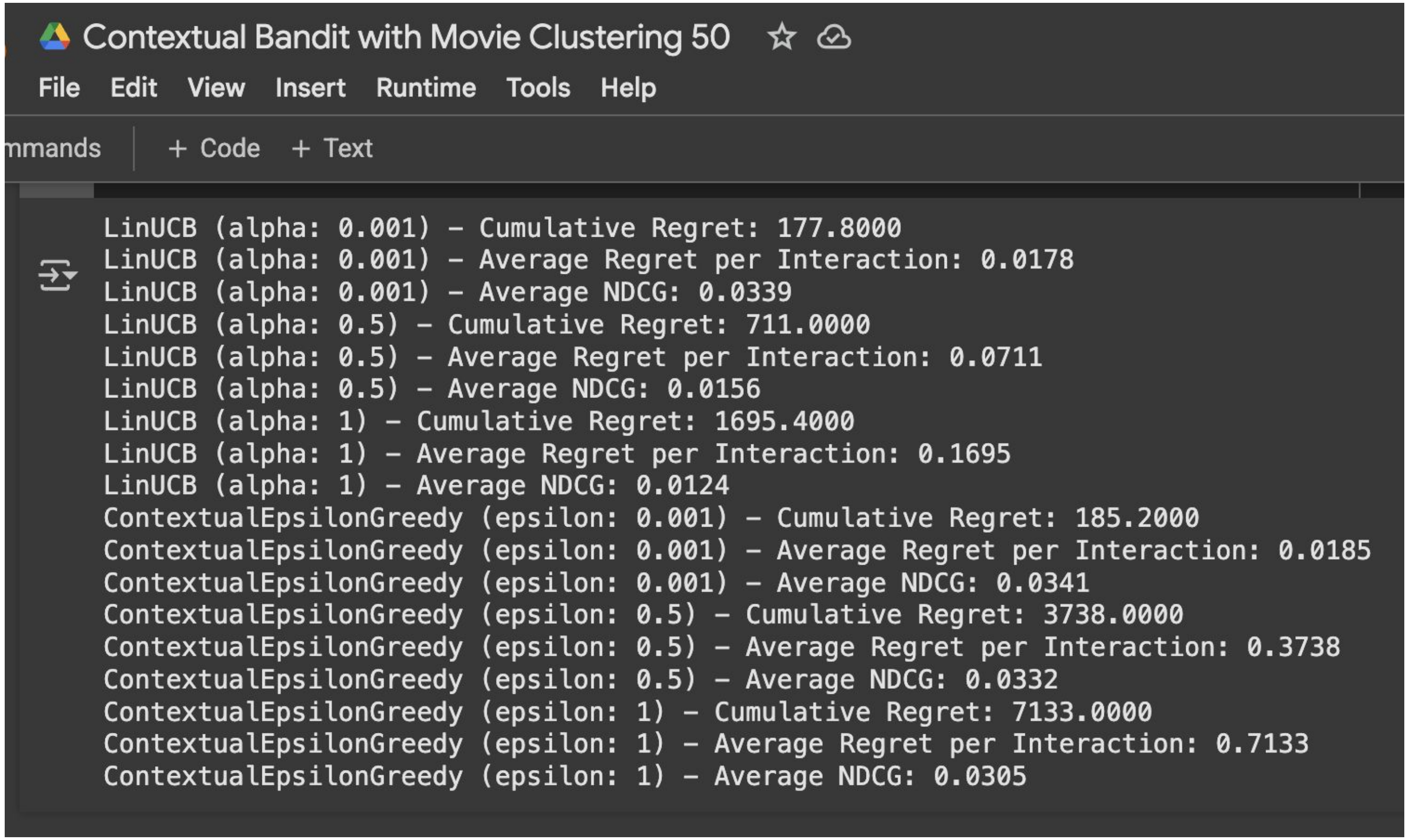
Appendix of results (in case needed)



Contextual Bandit with Movie Clustering 20
File Edit View Insert Runtime Tools Help
mmands + Code + Text

```
LinUCB (alpha: 0.001) - Cumulative Regret: 135.8000
LinUCB (alpha: 0.001) - Average Regret per Interaction: 0.0136
LinUCB (alpha: 0.001) - Average NDCG: 0.1599
LinUCB (alpha: 0.5) - Cumulative Regret: 342.8000
LinUCB (alpha: 0.5) - Average Regret per Interaction: 0.0343
LinUCB (alpha: 0.5) - Average NDCG: 0.1072
LinUCB (alpha: 1) - Cumulative Regret: 691.8000
LinUCB (alpha: 1) - Average Regret per Interaction: 0.0692
LinUCB (alpha: 1) - Average NDCG: 0.0666
ContextualEpsilonGreedy (epsilon: 0.001) - Cumulative Regret: 137.8000
ContextualEpsilonGreedy (epsilon: 0.001) - Average Regret per Interaction: 0.0138
ContextualEpsilonGreedy (epsilon: 0.001) - Average NDCG: 0.1601
ContextualEpsilonGreedy (epsilon: 0.5) - Cumulative Regret: 3576.6000
ContextualEpsilonGreedy (epsilon: 0.5) - Average Regret per Interaction: 0.3577
ContextualEpsilonGreedy (epsilon: 0.5) - Average NDCG: 0.1581
ContextualEpsilonGreedy (epsilon: 1) - Cumulative Regret: 7003.8000
ContextualEpsilonGreedy (epsilon: 1) - Average Regret per Interaction: 0.7004
ContextualEpsilonGreedy (epsilon: 1) - Average NDCG: 0.1515
```

Appendix of results (in case needed)



Contextual Bandit with Movie Clustering 50
File Edit View Insert Runtime Tools Help
mmands + Code + Text

```
LinUCB (alpha: 0.001) - Cumulative Regret: 177.8000
LinUCB (alpha: 0.001) - Average Regret per Interaction: 0.0178
LinUCB (alpha: 0.001) - Average NDCG: 0.0339
LinUCB (alpha: 0.5) - Cumulative Regret: 711.0000
LinUCB (alpha: 0.5) - Average Regret per Interaction: 0.0711
LinUCB (alpha: 0.5) - Average NDCG: 0.0156
LinUCB (alpha: 1) - Cumulative Regret: 1695.4000
LinUCB (alpha: 1) - Average Regret per Interaction: 0.1695
LinUCB (alpha: 1) - Average NDCG: 0.0124
ContextualEpsilonGreedy (epsilon: 0.001) - Cumulative Regret: 185.2000
ContextualEpsilonGreedy (epsilon: 0.001) - Average Regret per Interaction: 0.0185
ContextualEpsilonGreedy (epsilon: 0.001) - Average NDCG: 0.0341
ContextualEpsilonGreedy (epsilon: 0.5) - Cumulative Regret: 3738.0000
ContextualEpsilonGreedy (epsilon: 0.5) - Average Regret per Interaction: 0.3738
ContextualEpsilonGreedy (epsilon: 0.5) - Average NDCG: 0.0332
ContextualEpsilonGreedy (epsilon: 1) - Cumulative Regret: 7133.0000
ContextualEpsilonGreedy (epsilon: 1) - Average Regret per Interaction: 0.7133
ContextualEpsilonGreedy (epsilon: 1) - Average NDCG: 0.0305
```

AY2024/25 Sem2
CS4246/CS5446