Multi-Armed Bandit and Its Applications in Recommender Systems

Fan Jue	e0559110	A0221578B
Nadia Victoria Aritonang	e1505949	A0314698N
Reiner Anggriawan Jasin	e1503344	A0314502W
Tian Xiao	e0555784	A0220592L

Abstract

The *Multi-Armed Bandit* (MAB) problem studies how the agent can maximize the total reward when each action yields a random reward drawn from an unknown distribution. In this report, we provide a structured overview of *stochastic MAB* formulations and fundamental algorithms (ϵ-Greedy, UCB and ThompsonSampling) with provable performance guarantees. We then extend the discussion to *contextual MAB* where auxiliary information is available and introduce advanced algorithms under different modelling assumptions, particularly those with near-optimal guarantees. Lastly, we present a practical implementation of (contextual) MAB in movie recommender systems and demonstrate how it can help address real-world challenges such as the cold-start problem.

Acknowledgements

We used ChatGPT to check grammar and help implement the LinearUCB code in the implementation.

1 Introduction

The *Multi-Armed Bandit* (MAB) problem is a classic *reinforcement learning* (RL) problem that has wide applications. It studies how an agent repeatedly selects one among several actions (e.g., pull the arms of slot machines/*bandits*) to maximize its reward after a number of iterations, given that it does not know the reward (distribution) each action leads to *a priori*. During the sequential trials and observations of rewards, the agent gradually learns the reward (distribution) each action yields to make better decisions. Similar to other RL problems, the core challenge for MAB algorithms is the *exploitation-exploration tradeoff*: should one stick to the already tried and learnt actions that give the highest reward, or try new actions that can possibly give even higher rewards? By strategically balancing exploitation and exploration, MAB algorithms can work effectively and have broad use cases: *recommender systems* sequentially select items (e.g., movies on Netflix) to users, in order to maximize satisfaction and subscriptions (Zhou et al., 2017); advertisers may use MAB algorithms to choose target users to maximize the influence/exposure of their product advertisements in social networks (Vaswani et al., 2017).

In this report, we first introduce *stochastic* and *Bayesian* formulations of the MAB problem and present classic algorithms such as ϵ-Greedy (Sutton et al., 1998), UCB (Auer, 2000) and ThompsonSampling (Scott, 2010), with a focus on their provable theoretical guarantees. We further explore *contextual bandits* (Langford and Zhang, 2007) and explain different types of contextual bandits depending on the modelling assumptions and analyze the performance guarantee of their corresponding algorithms. In particular, we focus on those with **near-optimal guarantees** on the regret. Finally, we selectively implement these algorithms to build a movie recommender system, where we demonstrate how contextual bandits can address practical challenges such as the *cold start* problem (Silva et al., 2023)).

2 Stochastic Bandit and Algorithms

2.1 Formulation

Suppose at each round $t \in [T]$, the agent can choose an action $a \in A = \{a_1, a_2, \cdots, a_K\}$. The (random) reward R_a associated with action a follows a fixed distribution at each round. Let $\mu_a = \mathbb{E}[R_a]$ denote the

expected reward given by action a. Clearly, if the agent knows the distribution of every R_a in advance, it will consistently choose the single action a^* that maximizes μ_{a^*} and obtain an expected total reward of $T\mu_{a^*}$. The *stochastic bandit* (Lai and Robbins, 1985) problem studies **how the agent can maximize its (expected) reward given that it does not know the distributions in advance**. Hence, the goal of a stochastic bandit algorithm that chooses action a_t and receives reward r_t at round t is to minimize its (expected) *cumulative regret* ρ_T (i.e., loss in total rewards compared with the ideal strategy):

$$\rho_T = \mathbb{E}\left[T\mu_{a^*} - \sum_{t=1}^{T} r_t\right]. \tag{1}$$

An algorithm with a provable upper bound on the regret is favorable because it means the actual reward is at least a certain amount close to the ideal reward (i.e., it cannot be arbitrarily bad). The upper bound is *near-optimal* if we can additionally find a lower bound on the regret (such that for some stochastic bandit algorithms, no algorithm can do better than the lower bound) that is close to the upper bound, which means that the upper bound is really good. Below we give the lower bound of the stochastic bandit problem and in the later sections we introduce 3 classical algorithms that achieve near-optimal guarantees.

The $\Omega(\sqrt{KT})$ Lower Bound Auer et al. (2002) proves the following lower bound[1]:

Theorem 1 (Lower Bound of Stochastic Bandit)**.** *For any stochastic bandit algorithm, there exists a problem instance such that the algorithm achieves a regret of at least $\Omega(\sqrt{KT})$.*

Proof sketch. Consider a problem instance $\mathcal{I}$ where all R_a follows $\mathcal{N}(0,1)$. Consider another instance $\mathcal{I}'$ where only the reward given by action a' differs and $R_{a'} \sim \mathcal{N}(\Delta, 1)$ ($\Delta > 0$). For an algorithm, suppose action a' is chosen for n' rounds. The total regret is thus $\mathbb{E}[\Delta \cdot (T - n')]$. For any algorithm that can reliably distinguish $\mathcal{I}'$ from $\mathcal{I}$, n' should depend on Δ: when Δ is small, n' must be large enough for the agent to be certain that action a' is better (using *Pinsker's inequality* (Csiszár and Körner, 1981), $n' \geq \Theta(1/\Delta^2)$). Yet, the agent does not know which action a' is, so it has to try every action at least $\Theta(1/\Delta^2)$ times. Thus, we can choose $\Delta = \sqrt{K/T}$ such that $1/\Delta^2 = T/K$ and the agent has to make a uniform exploration. Thus, there exists an instance $\mathcal{I}'$ such that the regret is $\Omega\left(\mathbb{E}[\Delta \cdot (T - n')]\right) = \Omega(\sqrt{KT})$. □

2.2 ϵ-Greedy

The ϵ-Greedy algorithm (Sutton et al., 1998) is a simple yet powerful algorithm that balances exploitation and exploration. At round t, the algorithm specifies a parameter ϵ_t and

- **exploitation**: chooses the action that gives the highest mean reward so far with probability $1 - \epsilon_t$:

$$\arg\max_a \frac{1}{n_a(t)} \sum_{s \leq t; a_s = a} r_s, \tag{2}$$

  where $n_a(t)$ represents the number of choosing action a up to round t;
- **exploration**: chooses a random action with probability ϵ_t.

The vanilla version of ϵ-Greedy algorithm uses a constant $\epsilon_t \equiv \epsilon$. In expectation this would achieve a regret of at least $\Omega(\epsilon T)$: In ϵT out of T rounds, the algorithm is expected to choose a random action. In expectation, a random action gives a (constant) regret of $(1/K)\sum_{a \in A}(\mu_a - \mu_{a^*})$ at each of the ϵT rounds, thus $\Omega(\epsilon T)$. Therefore, ϵ should be small enough for the cumulative regret to be small.

However, a large ϵ is preferable at the earlier rounds to explore more actions to avoid being trapped in locally optimal actions. This motivates using an ϵ_t that decays over rounds t. Indeed, it is provable that the ϵ-Greedy algorithm can achieve a better regret when ϵ_t is chosen properly:

Theorem 2 (Regret Bound of ϵ-Greedy (Slivkins, 2024))**.** *Let $\epsilon_t = t^{-1/3}(K \log t)^{1/3}$. The ϵ-Greedy algorithm, when running for T iterations, achieves an expected regret of $\mathcal{O}(T^{2/3}(K \log T)^{1/3}) = \tilde{\mathcal{O}}(T^{2/3}K^{1/3})$.*

Proof sketch. At each round t, we consider exploitation and exploration separately:

- exploitation: If a suboptimal $a_t \neq a^*$ is chosen, its expected reward μ_{a_t} would not be too much lower than μ_{a^*} **(†)**, at least at the later rounds when $n_{a_t}(t)$ and $n_{a^*}(t)$ are large. This is because the empirical mean gradually converges to the expectation as there are more samples. With high probability, $\Delta_{a_t} \coloneqq \mu_{a^*} - \mu_{a_t}$ is bounded by $2\sqrt{2K \log t / t\epsilon_t}$.[2] Note that **(†)** is a crucial statement, an idea that forms the backbone of the proofs for most algorithms discussed in this report.

[1] The bounds discussed in this report do not depend on the actual *gap* between each action a and the optimal action a^*: $\Delta_a \coloneqq \mu_{a^*} - \mu_a$ (i.e., we discuss the **worst-case** (minimax) bounds) due to page limit. There are also a group of **gap-dependent** bounds that are worth noting.

[2] In fact, this is derived using Hoeffding bound (Eq. 3) which we only introduce in Sec. 2.3. The reasoning is also similar to **(‡)** there. We direct interested readers there since the proof there can be applied to contextual bandits later.

• exploration: The regret at round t is $\mathcal{O}(\epsilon_t)$ as in the discussion about vanilla ϵ-GREEDY algorithm. The specified ϵ_t in the theorem balances the tradeoff between the two regrets and thus gives the cumulative regret bound $\mathcal{O}(t^{2/3}(K\log t)^{1/3})$ for every round $t \in [T]$. □

2.3 Upper Confidence Bound

One limitation of the ϵ-GREEDY algorithm is that it only compares the empirical mean reward (Eq. (2)) without considering the confidence/uncertainty. For example, the optimal action a^* can accidentally give a low reward when it is first explored so it will not be exploited in the near future, causing a large regret. This motivates that **an action should not be denied due to its low empirical reward if it is not sufficiently tried/the agent is not confident that it is sub-optimal**. Based on this, Auer (2000) proposes the *upper confidence bound* (UCB) algorithm whose general framework is as follows[3]:

1. At each round, the agent specifies a *confidence region* Π_a^t of the reward associated with each action a such that with high probability, the true expected reward $\mu_a \in \Pi_a^t$ for all $t \in [T]$;
2. The agent chooses the action a_t with the highest reward within its confidence region (i.e., its upper confidence bound $\texttt{UCB}_{a_t}^t$). Clearly, $\texttt{UCB}_{a_t}^t \geq \texttt{UCB}_{a_*}^t$.

At the earlier rounds when confidence is low, the confidence region Π_a^t for each action a is large, giving the agent more chances to explore; at the later rounds when confidence is high, the confidence region Π_a^t for each action a is small. Even if a suboptimal $a_t \neq a^*$ is chosen, its expected reward μ_{a_t} would not be too much lower than μ_{a^*} because $\texttt{UCB}_{a_t}^t \geq \texttt{UCB}_{a_*}^t$, so the regret is small. Therefore, it remains how the agent should set the confidence regions to achieve provable near-optimal guarantees.

In the stochastic bandit setting, the agent's confidence on the reward given by action a gets higher when the number of choosing action a, n_a, increases. The empirical mean reward will converge to μ_a when $n_a \to \infty$. This can be captured by the Hoeffding bound (Hoeffding, 1994):

Theorem 3 (Hoeffding Bound)**.** *Let $R_{a,\tau}$ be independent samples of R_a. With probability $\geq 1-\delta$,*

$$\left|\frac{1}{n_a}\sum_{\tau=1}^{n_a} R_{a,\tau} - \mu_a\right| \leq (h-\ell)\sqrt{\frac{\log\frac{2}{\delta}}{2n_a}}, \tag{3}$$

where h and ℓ are upper and lower bounds of rewards.

Let $\hat{\mu}_a^t$ denote the empirical mean reward given by action a up to round t. Define $u_a^t := \sqrt{\log(2/\delta)/(2n_a)}$. By Eq. (3), we have with high probability the true expectation $\mu_a \in \Pi_a^t := [\hat{\mu}_a^t \pm c \cdot u_a^t]$ for any round t if the hyperparameter c is chosen properly (e.g., close to $h-\ell$). By *union bound* over T rounds, we have with high probability the above holds for all rounds $t \in [T]$. Therefore, the choice of Π_a^t is valid and thus $\texttt{UCB}_a^t := \hat{\mu}_a^t + c \cdot u_a^t$. It leads to the following regret guarantee:

Theorem 4 (Regret Bound of UCB (Bubeck and Cesa-Bianchi, 2012))**.** *The UCB algorithm, when running for T iterations, achieves an expected regret of $\mathcal{O}(\sqrt{KT\log T}) = \tilde{\mathcal{O}}(\sqrt{KT})$.*

Proof sketch. We give a general idea of proof that is applicable to not only the above theorem, but also applicable to all UCB-based algorithms to be discussed later. At round t, suppose an action $a_t \neq a^*$ is selected. Then $\texttt{UCB}_{a_t}^t \geq \texttt{UCB}_{a^*}^t$, that is, $\hat{\mu}_{a_t}^t + c \cdot u_{a_t}^t \geq \hat{\mu}_{a^*}^t + c \cdot u_{a^*}^t$. Since we require with high probability $\mu_a \in \Pi_a^t$, we have LHS $\leq \mu_{a_t} + c \cdot u_{a_t}^t + c \cdot u_{a_t}^t$ and RHS $\geq \mu_{a^*}$. Hence, $\Delta_{a_t} = \mu_{a^*} - \mu_{a_t} \leq 2c \cdot u_{a_t}^t$ (†). We know that the uncertainty $u_{a_t}^t$ is smaller if the number of choosing a_t, $n_{a_t}(t)$, is larger. Thus, if an action a is bad such that $\Delta_a = \mu_{a^*} - \mu_a$ is large, for (†) to hold, n_a must be small. In fact, (†) can be simplified to $n_a(T) \leq \mathcal{O}(\log T/\Delta_a + 1)$. By choosing Δ_a similarly to the proof of Thm. 1 and summing up $\Delta_a \cdot n_a(T)$ over all $a \in A$, we obtain the stated bound, which is close to the lower bound in Thm. 1. □

Remark. It is easy to see that the UCB algorithm can be improved if a concentration bound tighter than Hoeffding's bound is used to construct the confidence region Π_a^t, e.g., *empirical Berstein bound* (Maurer and Pontil, 2009), *Kullback-Leibler divergence* (Kullback and Leibler, 1951).

2.4 Bayesian Bandit and Thompson Sampling

Instead of constructing a confidence region using concentration bounds, it is natural to directly represent uncertainty using *Bayesian beliefs* (Gelman et al., 2013). We assume the reward given by action a follows a *likelihood* model $p(r_a|\theta_a)$ (e.g., binomial distribution) and the latent parameter θ_a follows a *prior*

[3]Later we will discuss how UCB can be adapted to contextual bandits, where the framework and thus the proof outline of regret bound remain the same.

distribution $p_a(\theta_a)$ (e.g., beta distribution). When an action a_t is chosen and reward r_{a_t} is observed, the agent updates its *posterior* belief using *Bayes' rule*: $p_{a_t}(\theta_{a_t}|r_{a_t}) \propto p_{a_t}(\theta_{a_t})p(r_{a_t}|\theta_{a_t})$. When the same action is chosen again, the current posterior becomes prior in Bayes' rule and a new posterior is computed.

For fixed latent parameters θ_a's, the problem reduces to a standard stochastic bandit and the goal is to minimize the regret in Eq. (1). Therefore, the goal of a *Bayesian bandit* is to minimize the **expected** cumulative regret over all possible stochastic bandit instances (i.e., *Bayesian regret*), that is,

$$\mathbb{E}_{\theta_a \sim p_a(\cdot)}[\rho_T | \{\theta_a\}_{a \in A}]. \tag{4}$$

How does the agent properly use its Bayesian belief to balance exploitation and exploration in Bayesian bandits? Scott (2010) proposes the ThompsonSampling algorithm, which works as follows:

1. At round t, for each action $a \in A$, the agent randomly samples a reward from its current belief which is updated using the process described in the first paragraph and is denoted as $p_a(r_a|\{r_{a_s}\}_{s=1}^{t-1})$.
2. The agent chooses the action that gives the highest sampled reward.

Let $\mathfrak{a}^*$ be the random variable representing the agent's belief on the optimal action. It is easy to see that the above algorithm is effectively sampling from $p(\mathfrak{a}^* = a|\{r_{a_s}\}_{s=1}^{t-1})$, that is, the probability of each action a being the optimal action given all the observed rewards so far.

ThompsonSampling achieves a similar near-optimal bound (on Bayesian regret) to UCB:

Theorem 5 (Regret Bound of ThompsonSampling (Agrawal and Goyal, 2013a))**.** *The ThompsonSampling algorithm, when running for T iterations, achieves a Bayesian regret of $\mathcal{O}(\sqrt{KT\log T}) = \tilde{\mathcal{O}}(\sqrt{KT})$.*

Proof sketch. Let $I(\mathfrak{a}^*; r_{a_t}|\{r_{a_s}\}_{s=1}^{t-1})$ denote the information gain about the optimal action $\mathfrak{a}^*$ given by the observed reward in round t conditioned on past rewards. The total information gain about $\mathfrak{a}^*$ over T rounds is thus $I_T \coloneqq \sum_{t=1}^{T} \mathbb{E}[I(a^*; r_{a_t}|\{r_{a_s}\}_{s=1}^{t-1})]$. At iteration t, the instantaneous regret is $\rho_t \coloneqq r_{a^*} - \mathbb{E}[r_{a_t}|\{r_{a_s}\}_{s=1}^{t-1}]$. The core idea here is that if the regret at round t is large, then the information gain about $\mathfrak{a}^*$ at this round will also be large (using Pinsker's inequality, $\rho_t^2 \le 2 \cdot I(a^*; r_{a_t}|\{r_{a_s}\}_{s=1}^{t-1})$), so that the agent can identify the optimal action faster. Summing over T iterations, we have the total regret $\sum_{t=1}^{T} \mathbb{E}[\rho_t] \le \sqrt{2T \cdot I_T}$ by *Cauchy-Schwarz inequality*. Since each of the K actions has a posterior that can be specified to $\log T$ bits of precision after T rounds, the total information gain is at most $I_T \le \mathcal{O}(K \log T)$. Therefore, $\rho_T \le \mathcal{O}(\sqrt{2T \cdot K \log T}) = \mathcal{O}(\sqrt{KT \log T}) = \tilde{\mathcal{O}}\sqrt{KT}$. □

3 Contextual Bandit and Algorithms

3.1 Formulation

Suppose at each round t, the agent observes additional information associated with each action that may be related to the reward. For example, when a movie recommender chooses a movie to recommend to a user, it may have access to the user's gender, age, watching history etc., which can affect whether the user will click/appreciate each movie. However, a stochastic bandit would not use such information and even in the ideal case it will keep recommending the single movie that most users like. Therefore, to utilize such information and obtain even higher rewards, *contextual bandits* (Langford and Zhang, 2007) are proposed. The additional information at each round is formulated as a set of vectors $\{\mathbf{x}_{t,a_k}\}_{k=1}^{K}$ where the *contextualized action* x_{t,a_k} represents the *context* vector associated with action a_k at round t. Since there might be a large number of different contexts, it remains computationally intractable to learn the reward associated with each context separately. Hence, a functional relationship is often assumed between the contexts and their corresponding rewards:

$$R_{t,a_k} \coloneqq f(\mathbf{x}_{t,a_k}) + \xi_{t,a_k} \tag{5}$$

for any $t \in T$ and $k \in K$, where $f(\cdot)$ is a function (e.g., a linear model) and ξ_{t,a_k} is a zero-mean random noise whose distribution is to be specified later. Clearly, the expected reward at round t, $\mathbb{E}[R_{t,a_k}] = f(\mathbf{x}_{t,a_k})$ and there still exists an optimal action a_t^* at each round such that the expected reward is maximized. The goal of a contextual bandit algorithm is still to minimize the regret between the rewards given by the optimal actions and the actual rewards, that is,

$$\rho_T \coloneqq \mathbb{E}\left[\sum_{t=1}^{T} R_{t,a_t^*} - R_{t,a_t}\right], \tag{6}$$

where a_t is the action chosen by the agent at round t. Note that the reward given by the optimal action is also assumed to follow the modelling assumption, i.e., $R_{t,a_t}^* = f(\mathbf{x}_{t,a_t^*}) + \xi_{t,a_t^*}$, so it is important to use a suitable model, such that a good algorithm that gives a low regret can work well in practice.

The lower bound of contextual bandit depends on the specific modelling assumption and will be introduced thereafter. Most contextual bandit algorithms are directly adapted from the UCB and ThompsonSampling algorithms and operate under a similar mechanism. The proof of near-optimality therefore follows the same core idea, except that the confidence region/Bayesian belief is defined differently. Therefore, we will only state the upper bounds and highlight the necessary modifications.

3.2 (Generalized) Linear Bandit and Algorithms

The (generalized) linear bandit assumes the function f to be a (generalized) linear function, that is,

$$R_{t,a_k} \coloneqq g\left(\mathbf{w}^\top \mathbf{x}_{t,a_k}\right) + \xi_{t,a_k}, \tag{7}$$

where $g : \mathbb{R} \to \mathbb{R}$ is a possibly non-linear function and $\mathbf{w}$ is a vector served as the weights of the linear model. When g is the *identity function* (i.e., $\forall r\ [g(r) = r]$), then it is the linear bandit; when g is the *sigmoid function*, then it is a *logistic model*. We assume the linear bandit when we discuss the specific algorithms for simplicity. Although (generalized) linear bandits have a simple modelling assumption, they are found to work surprisingly well in practice (Kang and Kim, 2023).

LinearUCB Similar to the vanilla UCB algorithm (Sec. 2.3), the LinearUCB algorithm (Li et al., 2010) works by constructing a high-probability confidence region Π^t on the **linear weights** $\mathbf{w}$ instead of on the reward associated with each action, then choosing the best contextualized action a_t that maximizes the expected reward under its most favorable weight inside Π^t, that is,

$$a_t \coloneqq \arg\max_{a \in A} \max_{\mathbf{w} \in \Pi} \mathbf{w}^\top \mathbf{x}_{t,a}. \tag{8}$$

The weight vector $\mathbf{w}$, however, does not become more accurate with more samples of $\mathbf{w}$ being drawn (as in vanilla UCB). Thus, the *confidence ellipsoid bound* (Abbasi-yadkori et al., 2011) is used instead:

Theorem 6 (Confidence Ellipsoid Bound)**.** *Let $\{\mathbf{x}_{t,\tilde{a}_t}\}$ be any sequence of contextualized actions adapted to filtration $\{F_t\}_{t=1}^T$.*[4] *Assume the feature $\mathbf{x}_{t,a}$ has a bounded norm $\|\mathbf{x}_{t,a}\| \le L$ and the noise $\xi_{t,a}$ is conditionally ν-sub-Gaussian (i.e, its moment-generating function at any λ is bounded by that of a Gaussian with variance ν^2), then for any a and $\delta \in (0,1)$, we have with probability $1-\delta$, for **all** round t,*

$$\|\hat{\mathbf{w}}_t - \mathbf{w}^*\|_{\{a_\tau\}_{\tau=1}^t} \le \left(\nu\sqrt{d \log \frac{1+tL/\lambda}{\delta}} + \sqrt{\lambda}\|\mathbf{w}^*\|\right), \tag{9}$$

where $\hat{\mathbf{w}}_t$ is the fitted linear weights at round t, $\|\cdot\|_{\{a_\tau\}_{\tau=1}^t}$ measures the Mahalanobis norm (Mahalanobis, 1936) of $\cdot$ (i.e., a confidence-weighted Euclidean norm where dimensions with higher confidence based on past contextualized actions are weighted higher[5]*).*

From its name, the above bound gives an ellipsoidal confidence region Π^t. Thus, for each action a, finding the most favorable weight inside Π^t is optimizing a (generalized) linear objective constrained on a convex set, which is well established.

Following a similar minimax argument as Thm. 1, Dani et al. (2008) shows that the lower bound of linear contextual bandit is $\Omega(d\sqrt{T})$. Correspondingly, LinearUCB is near-optimal:

Theorem 7 (Regret Bound of LinearUCB (Abbasi-yadkori et al., 2011))**.** *The LinearUCB algorithm, when running for T iterations, achieves an expected regret of $\tilde{\mathcal{O}}(d\sqrt{T})$.*

LinearTS Similar to the ThompsonSampling algorithm, the LinearTS algorithm (Agrawal and Goyal, 2013b) aims to represent the agent's uncertainty on the **linear weights** $\mathbf{w}$ using a distribution (i.e., similar to *Bayesian linear regression* (Bishop, 2006)). In each round t,

1. the agent randomly samples a **weight** $\tilde{\mathbf{w}}$ from its current belief;
2. the agent chooses the contextualized action that gives the highest reward according to the sampled weight (i.e., $\arg\max_a \tilde{\mathbf{w}}^\top \mathbf{x}_{t,a}$), plays the action and observes the new reward.

More specifically, Agrawal and Goyal (2013b) uses a Gaussian likelihood $p(r_{t,a}|\mathbf{w}) = \mathcal{N}(\mathbf{w}^\top \mathbf{x}_{t,a}, v^2)$ where v is assumed to be a known constant. The prior distribution $p(\mathbf{w})$ follows the likelihood's conjugate prior distribution (also a Gaussian distribution) such that the posterior $p(\mathbf{w}|r_{t,a})$ has a closed-form expression. In this way, the belief update can be done efficiently at each round.

The LinearTS algorithm achieves the following guarantee:

[4] Roughly, this means that actions at round t depends only on the information up to round t instead of the future.
[5] This also explains why the confidence region Π is an ellipsoid.

Theorem 8 (Regret Bound of LinearTS (Abeille and Lazaric, 2017)). *The LinearTS algorithm, when running for T iterations, achieves an expected regret of $\tilde{\mathcal{O}}(d^{3/2}\sqrt{T})$.*

3.3 Gaussian Process Bandit and Algorithms

The Gaussian process bandit (Srinivas et al., 2010) assumes the function f to be a *Gaussian process* (GP) model (Rasmussen and Williams, 2006). In particular, a GP model $\mathcal{GP}(\mu, \mathfrak{k})$ consists of a *mean function* $\mu(\cdot)$ and a *kernel function* $\mathfrak{k}(\cdot, \cdot)$ such that at input $\mathbf{x}$, $f(\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}), \mathfrak{k}(\mathbf{x}, \mathbf{x}))$. Moreover, for input $\mathbf{x}$ and $\mathbf{x}'$, the covariance between $f(\mathbf{x})$ and $\mathbf{x}'$ is $k(\mathbf{x}, \mathbf{x}')$. In other words, the functional values are jointly Gaussian and each functional value alone follows a Gaussian distribution. Therefore, if the agent specifies a prior $\mathcal{GP}(\mu, \mathfrak{k})$, it can update its belief based on the observed rewards using Bayes' rule too.

For each contextualized action $\mathbf{x}_{t,a}$, we know that $f(\mathbf{x}_{t,a}) \sim \mathcal{N}(\mu(\mathbf{x}_{t,a}), \mathfrak{k}(\mathbf{x}_{t,a}, \mathbf{x}_{t,a}))$. Therefore, it is natural to set the confidence region based on the mean $\mu(\mathbf{x}_{t,a})$ and the standard deviation $\sqrt{\mathfrak{k}(\mathbf{x}_{t,a}, \mathbf{x}_{t,a})}$ using UCB, or directly sample an estimated reward from each $f(\mathbf{x}_{t,a})$ and choose the action that gives the highest sampled reward using ThompsonSampling.

GaussianProcessUCB Let $\mu_{t-1}(\mathbf{x}_{t,a})$ and $\sigma^2_{t-1}(\mathbf{x}_{t,a})$ denote the posterior mean and variance of $f(\mathbf{x}_{t,a})$ just before the start of round t. Similar to the standard UCB discussed in Sec. 2.3, the confidence region is chosen as $\Pi^t_a := [\mu_{t-1}(\mathbf{x}_{t,a}) \pm c \cdot \sigma_{t-1}(\mathbf{x}_{t,a})]$ and the upper confidence bound $\texttt{UCB}^t_a := \mu_{t-1}(\mathbf{x}_{t,a}) + c \cdot \sigma_{t-1}(\mathbf{x}_{t,a})$. The GaussianProcessUCB algorithm thus chooses the action a_t that maximizes $\texttt{UCB}^t_a$ at each round t.

The original version of GaussianProcessUCB (Srinivas et al., 2010) uses $c_t = \sqrt{2B^2 + 300\iota_{t-1}\ln^3(t/\delta)}$, where B is an upper bound of the norm of f, ι_{t-1} is the *maximum information gain* (i.e., reduction in uncertainty) at round $t-1$ which depends on the actual contextual bandit problem and δ is the high-probability parameter same as in Thm. 3; an improved version (Chowdhury and Gopalan, 2017) uses $c_t = B + \nu \cdot \sqrt{2(\iota_{t-1}) + 1 + \ln(1/\delta)}$, where ν is the sub-Gaussian parameter same as in Thm. 6. GaussianProcessUCB achieves the following bound[6]:

Theorem 9 (Regret Bound of GaussianProcessUCB (Valko et al., 2013)). *The GaussianProcessUCB algorithm, when running for T iterations, achieves an expected regret of $\tilde{\mathcal{O}}(\sqrt{\tilde{d}T})$, where $\tilde{d}$ is the number of effective dimensions (i.e., principal components related to the kernel $k(\cdot, \cdot)$).*

GaussianProcessTS At round t, instead of drawing a reward for each input $\mathbf{x}_{t,a}$ separately, the GaussianProcessTS algorithm (Chowdhury and Gopalan, 2017) directly draws a **deterministic** function $\tilde{f}_t$ from $\mathcal{GP}(\mu_t, \mathfrak{k}_t)$ and chooses the action a_t that maximizes the function value $\tilde{f}_t(\mathbf{x}_{t,a})$. It achieves the following regret bound:

Theorem 10 (Regret Bound of GaussianProcessTS (Chowdhury and Gopalan (2017))). *The GaussianProcessTS algorithm, when running for T iterations, achieves an expected regret of $\tilde{\mathcal{O}}(d^{3/2}\sqrt{T})$.*

3.4 Neural Bandit and Algorithms

The neural bandit (Zhou et al., 2020) assumes the function f to be a neural network with parameters $\mathbf{w}$ and ReLU activations, denoted as $f_{\mathbf{w}}$. The expected reward at round t can still be estimated as $f_{\mathbf{w}_{t-1}}(\mathbf{x}_{t,a})$. Regarding the confidence/uncertainty, however, since neural networks are often over-parameterized, it is computationally challenging to maintain a belief on the parameters $\mathbf{w}$ and do a full Bayesian update whenever a new reward is observed, as what linear bandits do. Fortunately, the *neural tangent kernel* (NTK) theory (Jacot et al., 2018) has shown that neural networks behave approximately like linear models in functional space near the initialized parameters $\mathbf{w}_0$. Therefore, the agent can represent its uncertainty by

1. restricting the model parameters' distance from their initialization using some regularization parameter λ (i.e., adding the term $\lambda \cdot \|\mathbf{w} - \mathbf{w}_0\|$ to the loss function $\mathcal{L}_{\mathbf{w}}$;
2. approximating the model as a **linear model** $\mathbf{w}^\top(\nabla_{\mathbf{w}}\mathcal{L}_{\mathbf{w}}(\mathbf{x}_{t,a}))$ where $\nabla_{\mathbf{w}}\mathcal{L}_{\mathbf{w}}(\mathbf{x}_{t,a})$ is the gradient at $\mathbf{x}_{t,a}$. The approximation is valid because when differentiating with regards to $\mathbf{w}$, the same derivative will be obtained.

[6]The bound assumes finite arms. An alternative bound developed by Chowdhury and Gopalan (2017) is applicable to infinite/continuous actions. The same applies to the GaussianProcessTS algorithm below.

NeuralUCB Following the above approximation, Zhou et al. (2020) adapts the LinearUCB algorithm by treating the gradient $\nabla_{\mathbf{w}}\mathcal{L}_{\mathbf{w}_{t-1}}(\mathbf{x}_{t,a})$ as the input at each round t and uses it to compute the confidence ellipsoid same as in Thm. 6. In particular, the upper confidence bound on the reward has the following closed-form expression: $\texttt{UCB}_a^t \coloneqq f_{\mathbf{w}_{t-1}}(\mathbf{x}_{t,a}) + c_t \cdot \sqrt{\nabla_{\mathbf{w}}\mathcal{L}_{\mathbf{w}_{t-1}}(\mathbf{x}_{t,a})^\top \mathbf{Z}_{t-1}^{-1} \nabla_{\mathbf{w}}\mathcal{L}_{\mathbf{w}_{t-1}}(\mathbf{x}_{t,a})/m}$, where $\mathbf{Z}_t$ is the precision matrix capturing the shape of the confidence ellipsoid defined as $\mathbf{Z}_t = \lambda\mathbf{I} + \sum_{s=1}^{t-1} \nabla_{\mathbf{w}}\mathcal{L}_{\mathbf{w}_{s-1}}(\mathbf{x}_{s,a}) \nabla_{\mathbf{w}}\mathcal{L}_{\mathbf{w}_{s-1}}(\mathbf{x}_{s,a})^\top/m$ and m is the network width (assumed to be fixed across layers). When the exploration parameter c_t is properly chosen, the NeuralUCB satisfies the following guarantee:

Theorem 11 (Regret Bound of NeuralUCB (Zhou et al., 2020))**.** *The NeuralUCB algorithm, when running for T iterations, achieves an expected regret of $\tilde{\mathcal{O}}(\sqrt{\tilde{d}T})$, where $\tilde{d}$ is the number of effective dimensions (i.e., principal components related to the NTK, similar to Thm. 9)*

NeuralTS Zhang et al. (2021) adapts the ThompsonSampling algorithm to neural bandits by converting the confidence region in NeuralTS to a Gaussian distribution of mean $f_{\mathbf{w}_{t-1}}(\mathbf{x}_{t,a})$ and variance $c_t^2 \nabla_{\mathbf{w}}\mathcal{L}_{\mathbf{w}_{t-1}}(\mathbf{x}_{t,a})^\top \mathbf{Z}_{t-1}^{-1} \nabla_{\mathbf{w}}\mathcal{L}_{\mathbf{w}_{t-1}}(\mathbf{x}_{t,a})/m$. The rest of the algorithm is the same as standard ThompsonSampling: sample a reward for each action a from its distribution and choose the action a_t that gives the highest sampled reward. The NeuralTS achieves the following guarantee:

Theorem 12 (Regret Bound of NeuralTS (Zhang et al., 2021))**.** *The NeuralTS algorithm, when running for T iterations, achieves an expected regret of $\tilde{\mathcal{O}}(\sqrt{\tilde{d}T})$.*

Remark. Unlike in linear and Gaussian bandits, the bounds for NeuralUCB and NeuralTS are similar. This is because NeuralTS does not sample a function (as compared with sampling a linear weight in LinearTS and sampling a function in GaussianProcessTS). Instead, it considers the uncertainty on the reward given by each action directly, which reduces the need for discretization in functional space.

4 Implementation: Contextual Bandit in a Recommender System

In this section, we investigate the performance of contextual bandit algorithms in a movie recommendation setting, focusing on improving the cold-start problem, a common challenge in real-world recommender systems where new users lack sufficient interaction history and receive bad recommendations.

4.1 Methodology

Our implementation is based on Chen (2025) which focuses on using contextual bandits to increase prediction accuracy in a movie recommender system, particularly to mitigate the cold-start problem. Specifically, this paper uses LinearUCB to replace collaborative filtering to provide recommendations to new users based on contextual features until users have enough data, using the MovieLens dataset (Harper and Konstan, 2015). Specifically, a user-movie-rating clustering is created using *K-Means*, where each of the K clusters serves as an arm. Missing entries in the user-movie-rating matrix are replaced by latent factors using *FunkSVD* (Koren et al., 2009).

We focus on the implementation of the LinearUCB algorithm and compare it with ϵ-Greedy. When replicating Chen (2025)'s approach of creating the contextual vector, we find that this approach includes a factorization of user-movie-rating data into the context vector, which feeds information about how users will rate certain movies and leaks future predictions, causing the problem not purely to be a cold start problem. After further discussion and consideration, we use only the user's demographic information (e.g., Gender, Occupation) and the movie genre as the context vector. Furthermore, we use a K-means clustering of the movies based on its genre as the arms of our contextual bandit. The source code of our implementation can be found here.

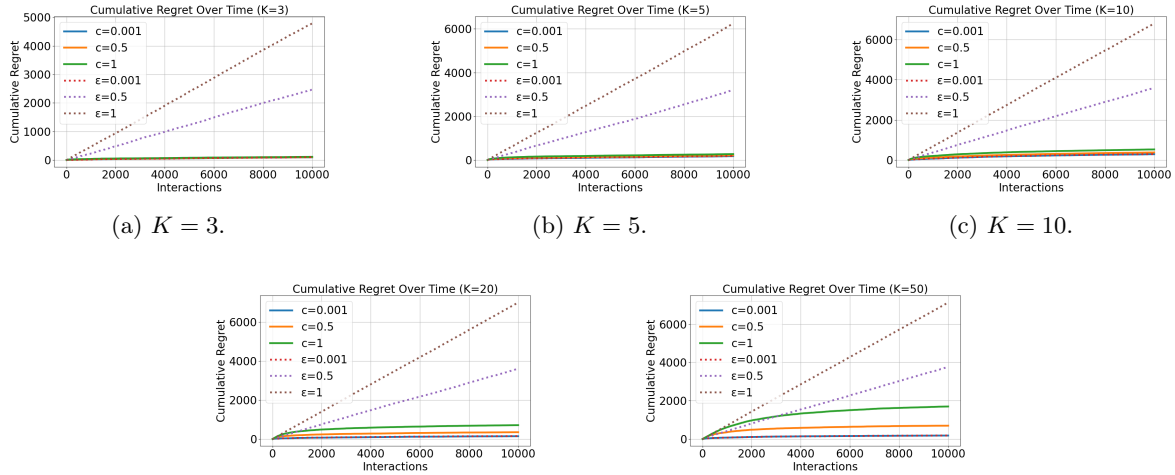
4.2 Results

In the practical implementation, we experimented with different hyperparameters, such as adjusting the c and ϵ for the contextual bandits and the number of movie clusters (i.e, arms of the agent) K. Our results are shown in Tab. 1 and Fig. 1. First, we observed **the impact of hyperparameters that control the exploitation-exploration trade-off** for LinearUCB (c) and ϵ-Greedy (ϵ). The higher the value of such hyperparameters, the greater the chance of exploration tendency. For both LinearUCB and

Table 1: Cumulative regrets of contextual bandits with different number of arms K and algorithms.

	$K = 3$	$K = 5$	$K = 10$	$K = 20$	$K = 50$
LinearUCB with exploration parameter c					
$c = 0.001$	93.40	176.20	287.80	142.40	180.80
$c = 0.5$	99.60	213.60	377.00	348.80	697.40
$c = 1$	111.00	272.00	527.80	710.20	1700.20
Contextual ϵ-Greedy					
$\epsilon = 0.001$	101.20	180.60	294.60	149.00	184.00
$\epsilon = 0.5$	2464.80	3205.00	3595.40	3602.80	3766.40
$\epsilon = 1$	4784.00	6248.40	6804.60	7003.80	7133.00

ϵ-Greedy, cumulative regret increases along with increasing hyperparameter values. An implication is that by allowing more exploration to happen, the contextual bandit agents potentially deviate from the optimal arm (over-exploring) since it increases the number of random arm selection for ϵ-greedy and LinUCB is prone to choosing arms with high uncertainty. Second, in the case of a **small** c and ϵ value, **the cumulative regret of LinearUCB and ϵ-Greedy do not differ too much** (notice when c and ϵ are 0.001). On the other hand, when we set c and ϵ with a **larger** value, **LinearUCB performs better than ϵ-Greedy** (notice when c and ϵ are 1). This is a result of the different arm selection mechanisms of the agents. ϵ-Greedy selects arms at random, whereas LinearUCB selects arms with the highest expected reward. We also observe that even with an increase in K, LinearUCB still outperforms ϵ-Greedy.



(a) $K = 3$. (b) $K = 5$. (c) $K = 10$.

(d) $K = 20$. (e) $K = 50$.

Figure 1: Plot of cumulative regrets. We can see that LinearUCB becomes flatter over time. This is most visible in $K = 20$ and $K = 50$. On the other hand, the line for ϵ-Greedy stays linear. This implies that the LinearUCB starts to learn the better arms from contexts, whereas ϵ-Greedy does not.

5 Conclusion

The MAB problem provides a fundamental framework for balancing exploration and exploitation in uncertain environments. While simple algorithms in stochastic setting are powerful, contextual bandits leverage additional information through linear models and neural networks to further enhance decision quality. Our implementation confirms that contextual bandits can be used in recommendation systems, with LinearUCB generally performing better than contextual ϵ-Greedy. Future research directions include adapting contextual bandits in other areas of artificial intelligence such as *multi-agent reinforcement learning* (Hsu et al., 2025), *Bayesian optimization* (Dai et al., 2023; Lin et al., 2023) and *active learning* (Wang et al., 2021; Ban et al., 2022).

References

Abbasi-yadkori, Y., Pál, D., and Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. *Proc. NeurIPS*, 24.

Abeille, M. and Lazaric, A. (2017). Linear Thompson sampling revisited. In *Proc. AISTATS*, pages 176–184. PMLR.

Agrawal, S. and Goyal, N. (2013a). Further optimal regret bounds for Thompson sampling. In *Proc. AISTATS*, pages 99–107. PMLR.

Agrawal, S. and Goyal, N. (2013b). Thompson sampling for contextual bandits with linear payoffs. In *Proc. ICML*, pages 127–135. PMLR.

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.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002). The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77.

Ban, Y., Zhang, Y., Tong, H., Banerjee, A., and He, J. (2022). Improved algorithms for neural active learning. In *Proc. NeurIPS*, volume 35, pages 27497–27509.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and non-stochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.

Chen, Y. (2025). Contextual bandits to increase user prediction accuracy in movie recommendation system. In *ITM Web of Conferences*, volume 73, page 01018. EDP Sciences.

Chowdhury, S. R. and Gopalan, A. (2017). On kernelized multi-armed bandits. In *Proc. ICML*, pages 844–853. PMLR.

Csiszár, I. and Körner, J. (1981). *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press.

Dai, Z., Lau, G. K. R., Verma, A., Shu, Y., Low, B. K. H., and Jaillet, P. (2023). Quantum Bayesian optimization. In *Proc. NeurIPS*, volume 36, pages 20179–20207.

Dani, V., Hayes, T. P., and Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback. In *Proc. COLT*, number 101, pages 355–366.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. CRC Press, 3rd edition.

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19.

Hoeffding, W. (1994). Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426.

Hsu, H.-L., Wang, W., Pajic, M., and Xu, P. (2025). Randomized exploration in cooperative multi-agent reinforcement learning. In *Proc. NeurIPS*, volume 37, pages 74617–74689.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Proc. NeurIPS*, volume 31.

Kang, M. and Kim, G.-S. (2023). Heavy-tailed linear bandit with Huber regression. In *Uncertainty in Artificial Intelligence*, pages 1027–1036. PMLR.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22.

Langford, J. and Zhang, T. (2007). The epoch-greedy algorithm for contextual multi-armed bandits. In *Proc. NeurIPS*, pages 817–824.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670.

Lin, X., Wu, Z., Dai, Z., Hu, W., Shu, Y., Ng, S.-K., Jaillet, P., and Low, B. K. H. (2023). Use your INSTINCT: INSTruction optimization usIng Neural bandits Coupled with Transformers. In *Proc. NeurIPS Workshop on Instruction Tuning and Instruction Following*.

Mahalanobis, P. C. (1936). On the generalised distance in statistics. *National Institute of Science of India*, 2:49 – 55.

Maurer, A. and Pontil, M. (2009). Empirical bernstein bounds and sample variance penalization. In *Proc. COLT*, pages 115–124.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Scott, S. L. (2010). A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658.

Silva, N., Silva, T., Werneck, H., Rocha, L., and Pereira, A. (2023). User cold-start problem in multi-armed bandits: when the first recommendations guide the user's experience. *ACM Transactions on Recommender Systems*, 1(1):1–24.

Slivkins, A. (2024). Introduction to multi-armed bandits.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. (2010). Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proc. ICML*, pages 1015–1022.

Sutton, R. S., Barto, A. G., et al. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Valko, M., Korda, N., Munos, R., Flaounas, I., and Cristianini, N. (2013). Finite-time analysis of kernelised contextual bandits. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, pages 654–663.

Vaswani, S., Kveton, B., Wen, Z., Ghavamzadeh, M., Lakshmanan, L. V., and Schmidt, M. (2017). Model-independent online learning for influence maximization. In *Proc. ICML*, pages 3530–3539. PMLR.

Wang, Z., Awasthi, P., Dann, C., Sekhari, A., and Gentile, C. (2021). Neural active learning with performance guarantees. In *Proc. NeurIPS*, volume 34, pages 7510–7521.

Zhang, W., Zhou, D., Li, L., and Gu, Q. (2021). Neural Thompson sampling. In *Proc. ICLR*.

Zhou, D., Li, L., and Gu, Q. (2020). Neural contextual bandits with UCB-based exploration. In *Proc. ICML*, pages 11492–11502. PMLR.

Zhou, Q., Zhang, X., Xu, J., and Liang, B. (2017). Large-scale bandit approaches for recommender systems. In *Proc. NeurIPS*, pages 811–821. Springer.

Appendix

A Project Administration

A.1 Division of Labor

All team members contribute equally to this project. Fan Jue and Tian Xiao focus primarily on the theoretical part, while Nadia Victoria Aritonang and Reiner Anggriawan Jasin focus primarily on the implementation part.

A.2 Use of AI Tools

These are the ChatGPT sessions we used when implementing the LinearUCB algorithm, which contain the prompts we used and the responses from ChatGPT:

- Implementing LinearUCB part 1;
- Implementing LinearUCB part 2;
- Troubleshooting;
- Plotting cumulative regrets.