HIERARCHICAL BAYESIAN FOR METADATA-BASED MULTI-TASK BANDITS

AAYUSH GAUTAM, ANDREW PORTER, JHANVI GARG, AND FATEMEH DOUDI

Abstract. Recommendation systems are an increasingly important part of online marketing. Motivated by recommendation systems and their usefulness in improving user experiences, we propose a Thompson Sampling algorithm called Meta Hierarchical Thompson Sampling (MetaHierTS) designed to learn appropriate arms for each task and minimize cumulative regret. The MetaHierTS algorithm utilizes task metadata and similarities in order to make the best selection. The agent learns how to solve various bandit tasks by using the information gained from each task while weighting similar tasks to accelerate the learning process. Throughout the paper, we demonstrate our method using Gaussian bandits and show through extensive experiments that this algorithm outperforms the Hierarchical Thompson Sampling Model proposed by [1].

1. Introduction

In an increasingly digital world, the importance of online advertising cannot be ignored. It is the way that many online sites find clients to buy products. The main objective then becomes how to best display relevant advertisements for unknown users in an online environment. Within this system are numerous users, each with individual preferences that the agent must comprehend to choose the best advertisement. When dealing with individual users, one intuitive approach for a recommender agent is to initially observe the preferences of each user and then, after some time, use the information that has been gathered. However, with the presence of metadata and other users, valuable information can be used to learn about preferences much quicker.

If there exist tasks that are similar to a given task, an effective strategy is to leverage the knowledge gained from the prior task and commence the exploitation phase earlier which is called information sharing. The presence of metadata enables us to differentiate between similar tasks and to exchange information among them.

As an example, in an advertisement recommendation system, the agent can utilize information such as a user's location, age, gender, etc., to estimate their preferences and display advertisements that are popular among users with similar attributes. This approach helps to avoid the need for a full exploration phase.

In this project, we created a new algorithm that uses Thompson sampling to exploit the similarities between tasks. Thompson sampling is a decision-making algorithm that begins with an initial assumption about the reward distribution, and at each time step, it updates its reward function based on the agent's interactions with the environment. However, Thompson sampling can be sensitive to the initial assumption, as stated in [3]. Therefore, we aim to improve the initial assumption for the reward function of new tasks by leveraging the experiences gained from other tasks.

This report begins by reviewing the relevant literature that served as the basis for our project. We then provide a detailed explanation of the settings we utilized, followed by a theoretical analysis of the regret bound. Additionally, we evaluated our algorithm's performance in a simulated environment.

2. Related Works

Our project is mainly built upon two research works that leverage Bayesian hierarchical modeling in meta-data multi-armed bandit (MAB). In [3], the authors suggest an approach that leverages metadata to capture task-specific details and a bandit framework to select the most suitable task to learn from in each iteration. The approach incorporates a Bayesian hierarchical model to capture inter-task dependencies and share knowledge across tasks. Additionally, the authors propose a multi-task regret metric to evaluate regret in such situations and show that their algorithm (MTTS) can decrease the mean multi-task regret in a sub-linear manner as the number of tasks increases.

In [1], Each task is defined by a set of parameters, which are randomly sampled from a distribution parameterized by a hyper-parameter. This hierarchical structure allows the agent to learn from each task and use this information to improve performance across all tasks. Then the article proposes a natural Thompson Hierarchical algorithm for the problem. However, the metadata is not utilized and this article did not take into account the similarity between tasks. Moreover, their structure is only capable of dealing with situations where tasks originate from a single distribution.

3. Setting

In our scenario, there are m tasks available, and the agent needs to interact with them exactly n times. At each time step t, the agent selects one task s from the set of available tasks S, and based on the information it has about the rewards for each action $a_{s,t}$ in the action set A_s, it chooses an action. For simplicity, we assume that all tasks have the same action space $A \subset \mathbb{R}^d$. After the agent takes action, it observes the reward and updates its belief about the reward distribution for that task. Our objective is to estimate the reward distribution for each task accurately, by leveraging the experiences the agent has gained from other tasks. To achieve this, we define a similarity matrix B that quantifies the similarity between pairs of tasks. We use a weighted Gaussian kernel to measure the similarity between tasks, which is given by the formula:

$$B_{s,s'} \propto e^{-\frac{1}{2}\|w^T(f_s - f_{s'})\|}$$

Here, w is a weight vector, f_s and $f_{s'}$ represent the feature vectors for tasks s and s' respectively, and $\|\cdot\|$ is the Euclidean distance between the two feature vectors. We use a weighted Gaussian kernel to account for the fact that some features may be on different scales. For example, if there are age and distance features for two users, distance is typically a larger scale than age, so neglecting the age difference may lead to incorrect similarity measurements. It should be emphasized that it is necessary to normalize each row of the similarity matrix to ensure that the sum of each row equals a constant.

The environment is generated as follows:

$$\begin{gathered} Y_{s,t} \mid A_{s,t}, \theta_{s,*} \sim N\left(A_{s,t}^{\top}\theta_{s,*}, \sigma^2\right), \quad \forall t \geqslant 1, s \in \mathcal{S} \\ \theta_{s,*} \mid \mu_{s*}, \gamma^* \sim N\left(\lambda\mu_{s*} + (1-\lambda)\gamma^*, \Sigma_0\right), \quad \forall s \in \mathcal{S} \\ \mu_{s*} \sim N\left(MB_s, \Sigma_a\right), \quad \forall s \in \mathcal{S} \\ \gamma_* \sim N\left(\mu_q, \Sigma_q\right) \end{gathered}$$

In this context, the matrices Σ, Σ_0, Σ_a, and Σ_q are known $d \times d$ covariance matrices that are positive semi-definite. Additionally, there are d-dimensional vectors $\theta_{s,*}$, $\mu_{s,*}$, and γ_*, where s is a task index. Moreover, $A_{s,t}$ is the action vector for task s at time t and $A_{s,t} \in A$. B_s is a similarity vector that reflects the similarity between task s and every other task. Finally, M is a known $= d \times |\mathcal{S}|$ matrix that demonstrates our prior belief about each state and action.

The main contribution of MetaHierTS is in the second layer, where the distribution for hyper-parameter $\theta_{s,*}$ is specified. Unlike the approach presented in [1], which assigns the same weight (γ_*) to the experience of all tasks, a weighted combination of information($\mu_{s,*}$) is considered from all tasks based on their similarity as well as an unweighted combination of information(γ_*) from all tasks. This is because in certain scenarios, there may be tasks with similar feature vectors but different preferences compared to their similar tasks. This could be due to a lack of feature data or incorrect metadata. By incorporating a weighted combination of information based on task similarity, we prevent these tasks from being biased solely by their similar tasks. In essence, this approach ensures that each task's experience is considered in a balanced manner, accounting for both similarities and differences between tasks.

4. Theoretical Analysis

In this section, we conducted an analysis of the regret bound, building upon the proof presented in [1]. Since our proof is only slightly different, we will only present the final results. To simplify our analysis of the regret bound, we focused on the Gaussian bandit, which is a special version of the linear bandit. The procedure in this section follows a three-step process: first, we derived the posterior

distribution for our hyper-parameters; then, we explained the algorithm; and finally, we computed the regret bound.

4.1. **Calculating Posterior.** To begin the computation of the posterior distributions, it is necessary to define certain parameters:

- Consider a $d-$dimensional linear bandit problem.
- $B(s, s')$ is the s'th element in B_s similarity vector.
- $G_{s,t} = \sigma^{-2} \sum_{l<t} \mathbb{1}_{S_l}(s) A_{s,l} A_{s,l}^T$ and $G_{s,t}$ is the outer product of the features of taken actions in task s up to round t
- $R_{s,t} = \sigma^{-2} \sum_{l<t} \mathbb{1}_{S_l}(s) A_{s,l} Y_{s,l}$ where $R_{s,t}$ is the weighed reward of state s until time t, weighted by features of taken actions

The update rules for posterior distributions can be derived in a similar way as [2]

4.1.1. *Posterior Distribution of $\gamma_* \sim Q_t$.* If we denote $\gamma_* \mid \mathcal{H}_t \sim N\left(\bar{\mu}_t, \bar{\Sigma}_t\right)$

$$\bar{\mu}_t = \bar{\Sigma}_t \left(\Sigma_q^{-1} \mu_q + \sum_{s \in [m]} (\Sigma_0 + G_{s,t}^{-1})^{-1} G_{s,t}^{-1} R_{s,t} \right)$$
$$\bar{\Sigma}_t^{-1} = \Sigma_q^{-1} + (\Sigma_0 + G_{s,t}^{-1})^{-1}$$

4.1.2. *Update of μ_{s*} : $P_{s,t}$.*

$$\mu_{s*} \mid \mathcal{H}_t \sim N\left(\hat{\mu}_{s,t} \hat{\Sigma}_{s,t}\right)$$

where

$$\hat{\mu}_{s,t} = M_{s,t} B_s$$
$$M_{s,t} = \hat{\Sigma}_{s,t} (\Sigma_0^{-1} M + X)$$

where X is a d $\times |S|$matrix and column i^{th} of X is defined as follows:

$$X_{:,i} = (\Sigma_0 + G_{i,t}^{-1})^{-1} G_{i,t}^{-1} R_{i,t}$$
$$\hat{\Sigma}_{s,t}^{-1} = \Sigma_a^{-1} + \sum_{s' \in [m]} (\Sigma_0 + G_{s',t}^{-1})^{-1} B(s, s')^{-2}$$

4.1.3. *Posterior for θ_{s*}.*

$$\theta_{s,*} | \mathcal{H}_t, \gamma_t, \mu_{s,t} \sim N\left(\tilde{\mu}_{s,t}, \tilde{\Sigma}_{s,t}\right)$$

Define

$$\mu_{s,t}^{'} = \lambda \gamma_t + \mu_{s,t}(1 - \lambda)$$
$$\tilde{\mu}_{s,t} = \tilde{\Sigma}_{s,t} (\Sigma_0^{-1} \mu_{s,t}^{'} + R_{s,t})$$
$$\tilde{\Sigma}_{s,t}^{-1} = \Sigma_0^{-1} + G_{s,t}$$

4.2. **Algorithm.** To summarize, our algorithm is a variation of Thompson sampling that updates the reward distribution based on the experience of each task and its similarity to other tasks.

The algorithm proceeds as follows:

Algorithm 1: MetaHierTS - Meta Hierarchical Thompson Sampling

Data: Task set S where $|S| = m$ and Round length nm
Result: Task recommendation for each round
Initialize the prior for Q_1 and $P_{s,1}$ for all s $\in \mathcal{S}$, set $\mathcal{H}_0 = \{\}$ and $S' = S$
for *each round* t **do**
- Choose task s at random from the set of tasks $\mathcal{S}'$;
- Sample γ_t from the posterior Q_t and $\mu_{s,t}$ from posterior $P_{s,t}$;
- Sample $\theta_{s,t} \sim \theta_{s,*} \mid \mathcal{H}_t, \gamma_t, \mu_{s,t}$;
- Sample the reward $Y_{s,t,a} \mid \theta_{s,t}, a$ for all action $a \in \mathcal{A}$;
- $a_m = \underset{a \in \mathcal{A}}{\text{argmax}} Y_{s,t,a}$
- Observe the true reward corresponding to the action a_m and call it $\tilde{Y}_{s,t,a_m}$
- $\mathcal{H}_{t+1} = \mathcal{H}_t \cup \left(a_m, \tilde{Y}_{s,t,a_m}\right)$
- if task s is taken n times: $S' = S' \backslash$s

4.3. **Regret Bound.** Taking inspiration from the proof presented in [1], we examine a scenario where a K-armed Gaussian bandit is considered, with $\Sigma_0 = \sigma_0^2 I_d$, $\Sigma_a = \sigma_a^2 I_d$, and $\Sigma_q = \sigma_q^2 I_d$. We make the assumption that the agent is making decisions for only one task per round, which is called sequential decision-making, and that the agent interacts with each task for precisely n rounds. Within this framework, we can establish an upper bound on the Regret.

Theorem (Sequential regret) - Let $|\mathcal{S}_t| = 1$ for all rounds t and action space is finite ($|\mathcal{A}| = K$). The Bayes regret of MetaHierTS environment is given by

$$\mathcal{BR}(m,n) \leq K\sqrt{\frac{2}{\pi}}\sigma_{max} + K\sqrt{2log(mn)mn[c_1 m + c_2]}$$

where

$$\sigma_{max} = \lambda^2\sigma_q^2 + (1-\lambda)^2\sigma_a^2 + \sigma_0^2$$
$$c_1 = \frac{\sigma_0^2}{\log(1+\sigma_0^2\sigma^{-2})}\log\left(1+\frac{n\sigma_0^2}{\sigma^2 K}\right)$$
$$c_2 = \frac{c_q c}{\log(1+c_q\sigma^{-2})}\log\left(1+\frac{m\sigma_q^2}{\sigma_0^2}\right)$$

here

$$c_q = \lambda^2\sigma_q^2 + (1-\lambda)^2\sigma_a^2$$

and

$$c = 1 + \frac{\sigma_0^2}{\sigma^2}$$

In the proof, we make an implicit assumption that $\sigma_a^2 \leq \sigma_q^2$

4.4. **Results.**

- The regret bound scales linearly with the number of tasks and sub-linearly with respect to n. This means that as the number of tasks increases, the regret bound also increases at a similar rate, but as the amount of time spent on the tasks increases, the increase in the regret bound is relatively smaller.

- The linear scaling of the regret bound with respect to the number of tasks is intuitively reasonable because during the generation of the environment, the task parameters, denoted as $\theta_{s,*}$, are generated independently from their distribution, given the parameters $\mu_{s,*}$ and $\gamma*$. This means that each task is generated separately, and there is no interaction or dependence between the tasks. Therefore, as the number of tasks increases, the regret bound also increases linearly because each additional task contributes to the overall regret bound independently.

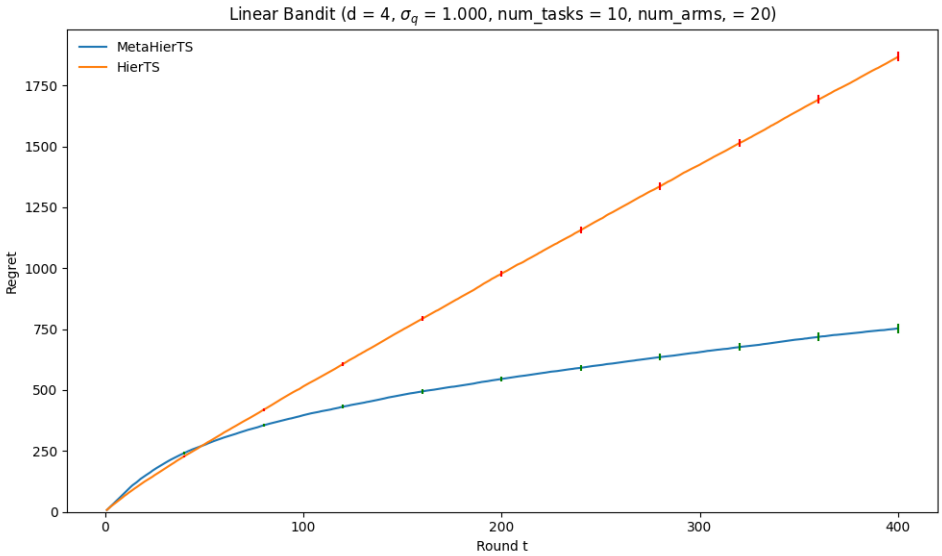
Even though $\mu_{s,*}$ and $\gamma*$ were known, still the regret would be linear because of the above reason.

5. Simulation

5.1. **Simulation Environment in concurrent setting.** In a concurrent setting, multiple tasks will be selected. Our algorithm uses the metadata on the tasks to make informed choices when selecting the actions. In order to simulate an environment with metadata we follow the procedure outlined below:

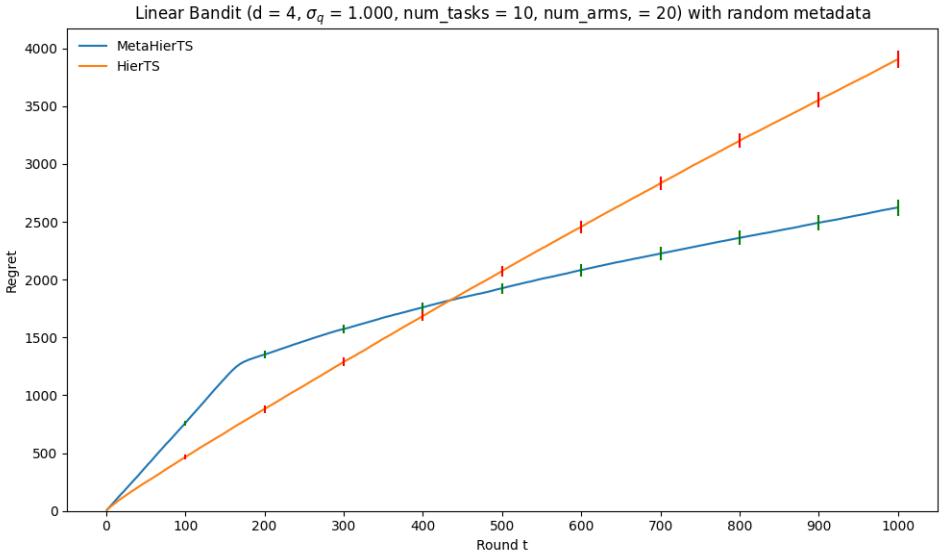
(1) Define the hyperparameters:
   (a) `num_tasks`: Number of tasks, $|S|$ or m in the model
   (b) `K`: Number of actions per task, $|A|$
   (c) `N`: total number of rounds
   (d) `num_tasks_per_round`: Number of tasks to be sampled per round
   (e) `lamda`: Weight given to metadata information during updates, λ
(2) Define the environment parameters
   (a) `num_clusters`: Number of clusters from which μ_* is sampled
   (b) sigma: Reward standard deviation, σ
   (c) mu_q: Mean of the Gaussian hyperprior, known to the algorithm and from which `num_clusters` number of μ_*s are sampled
   (d) `sigma_q` standard deviation of the Gaussian hyperprior, σ_q
   (e) `sigma_0` standard deviation of the Gaussian prior (with mean μ_*) from which θ_* is sampled.
(3) Sample `num_clusters` number of μ_*s from a Gaussian hyperprior with mean μ_q and standard deviation σ_q
(4) For each task, $s \in S$
   (a) Randomly choose a μ_*
   (b) Sample parameters $\theta_{s,*}$ from a Gaussian using μ_* as the mean and σ_0 as the standard deviation
   (c) Generate metadata by adding Gaussian noise to $\theta_{s,*}$
   (d) Sample parameters $A_{s,*}$ for each arm by sampling from a unit ball

5.2. **Results.** In this section, we present the results of our algorithm in various scenarios and compare them to the results obtained in [1].



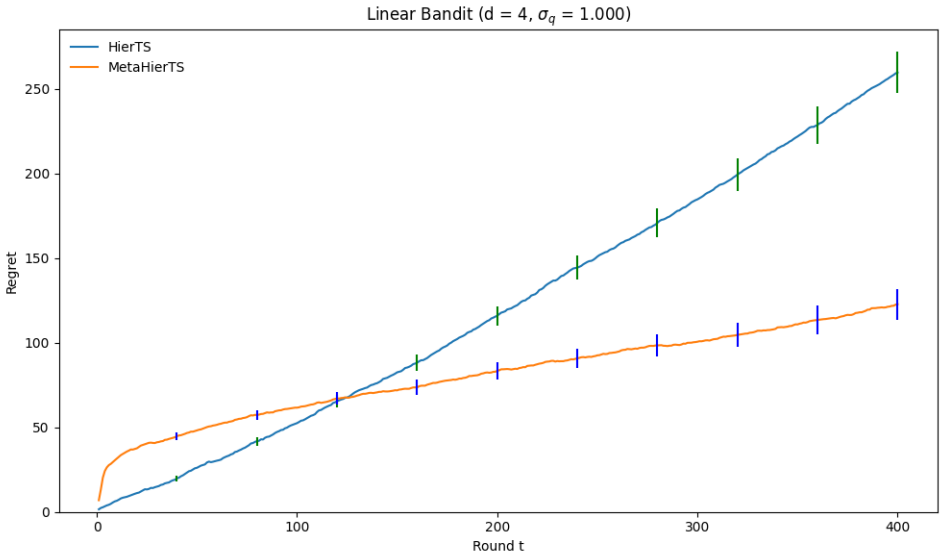
In the above figure, we can see that for a linear bandit whose arms are represented by a 4 dimensional vector, our algorithm achieves a much better sub-linear regret than HierTS [1].

In order to assess the impact of metadata in our results, we ran our algorithm again by giving our algorithm random metadata, which is a Gaussian with mean 0 and variance 1. The results are shown in the figure below.



As we can see from the results, when the metadata contains no information about the underlying similarity of tasks, our algorithm obtains worse regret than the HierTS algorithm for the first 150 rounds. However, after 150 rounds, the accumulated regret grows much slower and it eventually beats HierTS in cumulative regret after 400 rounds.

Furthermore, we also evaluate our algorithm in the same setting as proposed in the paper by Hong et. al. [1]. Here we do not have any clusters for μ_* i.e. there is only one μ_* from which all task parameters ($\theta_{s,*}$) are sampled. The result is shown in the figure below. In this scenario as well, we can see that our algorithm outperforms HierTS in terms of cumulative regret.



6. Conclusion and Future Directions

- We considered the problem of metadata multi-task bandit as in [1]
- To have better utilization of metadata, we defined and used a similarity matrix in our framework
- We use the aforementioned framework as a special case and generalized their setup

- Simulation results show that our MetaHierTS algorithm achieves better regret bounds than the algorithm presented in [1]
- In the future, we aim to improve the theoretical regret bound by including additional assumptions in the environment generation process.
- We aim to test our algorithm using a real-world dataset in the future.
- We plan to broaden our formulation to include non-Gaussian scenarios.
- The way the problem is structured into different levels does not seem to enhance the algorithm's ability to share information effectively. To achieve a sublinear regret with respect to the number of tasks, the tasks should be inherently correlated with each other. This means that the tasks should be related in some way such that the knowledge or insights gained in solving one task can be effectively applied to solving other tasks, thereby reducing the overall regret. When tasks are correlated, information sharing can be more effective, and the algorithm can take advantage of the shared information to improve performance across tasks. This may be possible if we have the following assumptions on the environment

  - **Clustering** - Multiple tasks share the same task parameter.
  - **Adding correlation between tasks parameter**- For every task s in task space $\mathcal{S}$, the task parameters $\theta_{s,a}$ jointly follows a multivariate normal distribution where the covariance matrix is not diagonal for every action a in $\mathcal{A}$.
  - **Sequential Tasks** - Here we have a series of related tasks that must be completed in a specific order, and the task parameter of each task influence the parameter of subsequent tasks.

7. Acknowledgement

We gratefully thank Nilson Chapagain for sharing his insights with us during the process of formulating the problem. Additionally, we would like to thank Prof. Dileep for referring us to relevant literature to guide our research. The authors' responsibilities were as follows - The problem formulation was done by all of us. Jhanvi and Fatemeh were responsible for writing the algorithm and analyzing the algorithm by calculating regret bounds. Aayush and Andrew were primarily responsible for generating data sets and testing the algorithm on the generated data sets. Throughout the project, We have shared equal responsibilities and helped each other in carrying out the above tasks.

References

[1] J. Hong, B. Kveton, M. Zaheer, and M. Ghavamzadeh. Hierarchical bayesian bandits, 2022.

[2] B. Kveton, M. Konobeev, M. Zaheer, C.-W. Hsu, M. Mladenov, C. Boutilier, and C. Szepesvari. Meta-thompson sampling. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5884–5893. PMLR, 18–24 Jul 2021.

[3] R. Wan, L. Ge, and R. Song. Metadata-based multi-task bandits with bayesian hierarchical models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29655–29668. Curran Associates, Inc., 2021.