Pretraining Decision Transformers with Reward Prediction for In-Context Multi-task Structured Bandit Learning

Anonymous authors
Paper under double-blind review

Keywords: Structured Bandit, Multi-task Learning, Decision Transformer

Summary

We study learning to learn for the multi-task structured bandit problem where the goal is to learn a near-optimal algorithm that minimizes cumulative regret. The tasks share a common structure and an algorithm should exploit the shared structure to minimize the cumulative regret for an unseen but related test task. We use a transformer as a decision-making algorithm to learn this shared structure from data collected by a demonstrator on a set of training task instances. Our objective is to devise a training procedure such that the transformer will learn to outperform the demonstrator's learning algorithm on unseen test task instances. Prior work on pretraining decision transformers either requires privileged information like access to optimal arms or cannot outperform the demonstrator. Going beyond these approaches, we introduce a pre-training approach that trains a transformer network to learn a near-optimal policy in-context. This approach leverages the shared structure across tasks, does not require access to optimal actions, and can outperform the demonstrator. We validate these claims over a wide variety of structured bandit problems to show that our proposed solution is general and can quickly identify expected rewards on unseen test tasks to support effective exploration.

Contribution(s)

1. We introduce a new pre-training and test time decision-making procedure that in-context learns the underlying reward structure for structured bandit settings, resulting in a near-optimal policy without access to privileged information even when training data comes from a sub-optimal demonstrator.
   Context: Previous works like DPT (Lee et al., 2023) required access to the optimal action per task, Algorithmic Distillation (AD) could not outperform the demonstrator, other works need to know the structure to perform optimally.
2. We show that our approach enables successful in-context learning across a diverse set of structured bandit settings where it matches the performance of existing algorithms that were developed with knowledge of the structure.
   Context: We evaluate our approach in linear,non-linear, bilinear, and latent bandit settings as well as bandit experiments based on real-life datasets and show that it lowers regret compared to DPT and AD while matching the near-optimal performance of specialized algorithms.
3. We show that our algorithm leverages the latent structure and conducts a two-phase exploration to minimize regret.
   Context: We analyze the exploration of the pretrained decision transformer in the simplified linear bandit setting where the optimal policy is well-understood. Previous works like DPT do not study the exploration conducted by such transformer algorithms. We introduce new actions both at train and test time. Since new actions are not shared across tasks now, the transformer algorithm fails to learn the latent structure as we scale up the number of new actions, thus indicating that it is relying on a discovered underlying structure. We observed in our experiments that our proposed algorithm implicitly conducts two-phase exploration, following the distribution of optimal action across training tasks and then switching to the most rewarding action for the task after observing a few in-context examples.

Pretraining Decision Transformers with Reward Prediction for In-Context Multi-task Structured Bandit Learning

Anonymous authors
Paper under double-blind review

Abstract

In this paper, we study the multi-task structured bandit problem where the goal is to learn a near-optimal algorithm that minimizes cumulative regret. The tasks share a common structure and any optimal algorithm should exploit the shared structure to minimize the cumulative regret for an unseen but related test task. We use a transformer as a decision-making algorithm to learn this shared structure so as to generalize to the unseen test task. The prior work of pretrained decision transformers like DPT requires access to the optimal action during training which may be hard in several scenarios. Diverging from these works, our learning algorithm does not need the knowledge of optimal action per task during training but predicts a reward vector for each of the actions using only the observed offline data from the diverse training tasks. Finally, during inference time, it selects action using the reward predictions employing various exploration strategies in-context for an unseen test task. We show that our model outperforms other methods like DPT, and Algorithmic Distillation (AD) and matches the performance of algorithms that requires privileged information on the structure of the problem. Interestingly, we show that our algorithm, without the knowledge of the underlying problem structure, can learn a near-optimal policy in-context by leveraging the shared structure across diverse tasks. We show that when the shared structure breaks down with the introduction of new actions both during training and test time, our proposed algorithm fails to learn the underlying latent structure. We further show that our algorithm conducts an implicit two-phase exploration and validate all of these findings over several experiments spanning linear, non-linear, real-life datasets, bilinear, and latent bandit settings. Finally, we theoretically analyze the performance of our algorithm and obtain generalization bounds in the in-context multi-task learning setting.

1 Introduction

In this paper, we study multi-task bandit learning with the goal of learning an algorithm that discovers and exploits structure in a family of related tasks. In multi-task bandit learning, we have multiple distinct bandit tasks for which we want to learn a policy. Though distinct, the tasks share some structure, which we hope to leverage to speed up learning on new instances in this task family. Traditionally, the study of such structured bandit problems has relied on knowledge of the problem structure like linear bandits (Li et al., 2010; Abbasi-Yadkori et al., 2011; Degenne et al., 2020), bilinear bandits (Jun et al., 2019), hierarchical bandits (Hong et al., 2022a;b), Lipschitz bandits (Bubeck et al., 2008; 2011; Magureanu et al., 2014), other structured bandits settings (Riquelme et al., 2018; Lattimore & Szepesvári, 2019; Dong et al., 2021) and even linear and bilinear multi-task bandit settings (Yang et al., 2022a; Du et al., 2023; Mukherjee et al., 2023). When structure is unknown an alternative is to adopt sophisticated model classes, such as kernel machines or neural networks, exemplified by kernel or neural bandits (Valko et al., 2013; Chowdhury & Gopalan, 2017; Zhou et al., 2020; Dai et al., 2022). However, these approaches are also costly as they learn complex, nonlinear models from the ground up without any prior data (Justus et al., 2018; Zambaldi et al., 2018).

In this paper, we consider an alternative approach of synthesizing a bandit algorithm from historical data where the data comes from recorded bandit interactions with past instances of our target task family. Concretely, we are given a set of state-action-reward tuples obtained by running some bandit algorithm in various instances from the task family. We then aim to train a transformer (Vaswani et al., 2017) from this data such that it can learn in-context to solve new task instances. Laskin et al. (2022) consider a similar goal and introduce the Algorithm Distillation (AD) method, however, AD aims to copy the algorithm used in the historical data and thus is limited by the ability of the data collection algorithm. Lee et al. (2023) develop an approach, DPT, that enables learning a transformer that obtains lower regret in-context bandit learning compared to the algorithm used to produce the historical data. However, this approach requires knowledge of the optimal action at each stage of the decision process. In real problems, this assumption is hard to satisfy and we will show that DPT performs poorly when the optimal action is only approximately known. With this past work in mind, the goal of this paper is to answer the question:

> *Can we learn an in-context bandit learning algorithm that obtains lower regret than the algorithm used to produce the training data without knowledge of the optimal action in each training task?*

To answer this question, we introduce a new pre-training methodology, called **Pre**-trained **De**cision **T**ransf**o**rmer with **R**eward Estimation (PreDeToR) that obviates the need for knowledge of the optimal action in the in-context data — a piece of information that is often inaccessible. Our key observation is that while the mean rewards of each action change from task to task, certain probabilistic dependencies are persistent across all tasks with a given structure (Yang et al., 2020; 2022a; Mukherjee et al., 2023). These probabilistic dependencies can be learned from the pretraining data and exploited to better estimate mean rewards and improve performance in a new unknown test task. The nature of the probabilistic dependencies depends on the specific structure of the bandit and can be complex (i.e., higher-order dependencies beyond simple correlations). We propose to use transformer models as a general-purpose architecture to capture the unknown dependencies by training transformers to predict the mean rewards in each of the given trajectories (Mirchandani et al., 2023; Zhao et al., 2023). The key idea is that transformers have the capacity to discover and exploit complex dependencies in order to predict the rewards of all possible actions in each task from a *small* history of action-reward pairs in a new task. This paper demonstrates how such an approach can achieve lower regret by outperforming state-of-the-art baselines, relying solely on historical data, without the need for any supplementary information like the action features or knowledge of the complex reward models. We also show that the shared actions across the tasks are vital for PreDeToR to exploit the latent structure. We show that PreDeToR learns to adapt, in-context, to novel actions and new tasks as long as the number of new actions is small compared to shared actions across the tasks.

Contributions

1. We introduce a new pre-training procedure of learning the underlying reward structure and a decision algorithm. Moreover, PreDeToR by predicting the next reward for all arms circumvents the issue of requiring access to the optimal (or approximately optimal) action during training time.
2. We demonstrate empirically that this training procedure results in lower regret in a wide series of tasks (such as linear, nonlinear, bilinear, and latent bandits) compared to prior in-context learning algorithms and bandit algorithms with privileged knowledge of the common structure.
3. We also show that our training procedure leverages the shared latent structure. We systematically show that when the shared structure breaks down no reward structure or exploration is learned.
4. Finally, we theoretically analyze the generalization ability of PreDeToR through the lens of algorithmic stability and new results for the transformer setting.

2 Background

In this section, we first introduce our notation and the multi-task, structured bandit setting. We then formalize the in-context bandit learning model studied in Laskin et al. (2022); Lee et al. (2023); Sinii et al. (2023); Lin et al. (2023); Ma et al. (2023); Liu et al. (2023c;a).

2.1 Preliminaries

In this paper, we consider the multi-task linear bandit setting (Du et al., 2023; Yang et al., 2020; 2022a). In the multi-task setting, we have a family of related bandit problems that share an action set $\mathcal{A}$ and also a common action feature space $\mathcal{X}$. The actions in $\mathcal{A}$ are indexed by $a = 1, 2, \ldots, A$. The feature of each action is denoted by $\mathbf{x}(a) \in \mathbb{R}^d$ and $d \ll A$. A policy, π, is a probability distribution over the actions.

Define $[n] = \{1, 2, \ldots, n\}$. In a multi-task structured bandit setting the expected reward for each action in each task is assumed to be an unknown function of the hidden parameter and action features (Lattimore & Szepesvári, 2020; Gupta et al., 2020). The interaction proceeds iteratively over n rounds for each task $m \in [M]$. At each round $t \in [n]$ for each task $m \in [M]$, the learner selects an action $I_{m,t} \in \mathcal{A}$ and observes the reward $r_{m,t} = f(\mathbf{x}(I_{m,t}), \boldsymbol{\theta}_{m,*}) + \eta_{m,t}$, where $\boldsymbol{\theta}_{m,*} \in \mathbb{R}^d$ is the hidden parameter specific to the task m to be learned by the learner. The function $f(\cdot, \cdot)$ is the unknown reward structure. This can be $f(\mathbf{x}(I_{m,t}), \boldsymbol{\theta}_{m,*}) = \mathbf{x}(I_{m,t})^\top \boldsymbol{\theta}_{m,*}$ for the linear setting or even more complex correlation between features and $\boldsymbol{\theta}_{m,*}$ (Filippi et al., 2010; Abbasi-Yadkori et al., 2011; Riquelme et al., 2018; Lattimore & Szepesvári, 2019; Dong et al., 2021).

In our paper, we assume that there exist weak demonstrators denoted by π^w. These weak demonstrators are stochastic A-armed bandit algorithms like Upper Confidence Bound (UCB) (Auer et al., 2002; Auer & Ortner, 2010) or Thompson Sampling (Thompson, 1933; Agrawal & Goyal, 2012; Russo et al., 2018; Zhu & Tan, 2020). We refer to these algorithms as weak demonstrators because they do not use knowledge of task structure or arm feature vectors to plan their sampling policy. In contrast to a weak demonstrator, a strong demonstrator, like LinUCB, uses feature vectors and knowledge of task structure to conduct informative exploration. Whereas weak demonstrators always exist, there are many real-world settings with no known strong demonstrator algorithm or where the feature vectors are unobserved and the learner can only use the history of rewards and actions.

2.2 In-Context Learning Model

Similar to Lee et al. (2023); Sinii et al. (2023); Lin et al. (2023); Ma et al. (2023); Liu et al. (2023c;a) we assume the in-context learning model. We first discuss the pretraining procedure.

Pretraining: Let $\mathcal{T}_{\text{pre}}$ denote the distribution over tasks m at the time of pretraining. Let $\mathcal{D}_{\text{pre}}$ be the distribution over all possible interactions that the π^w can generate. We first sample a task $m \sim \mathcal{T}_{\text{pre}}$ and then a context $\mathcal{H}_m$ which is a sequence of interactions for n rounds conditioned on the task m such that $\mathcal{H}_m \sim \mathcal{D}_{\text{pre}}(\cdot|m)$. So $\mathcal{H}_m = \{I_{m,t}, r_{m,t}\}_{t=1}^n$. We call this dataset $\mathcal{H}_m$ an in-context dataset as it contains the contextual information about the task m. We denote the samples in $\mathcal{H}_m$ till round t as $\mathcal{H}_m^t = \{I_{m,s}, r_{m,s}\}_{s=1}^{t-1}$. This dataset $\mathcal{H}_m$ can be collected in several ways: (1) random interactions within m, (2) demonstrations from an expert, and (3) rollouts of an algorithm. Finally, we train a causal GPT-2 transformer model TF parameterized by $\boldsymbol{\Theta}$ on this dataset $\mathcal{D}_{\text{pre}}$. Specifically, we define $\mathrm{TF}_{\boldsymbol{\Theta}}(\cdot \mid \mathcal{H}_m^t)$ as the transformer model that observes the dataset $\mathcal{H}_m^t$ till round t and then produces a distribution over the actions. Our primary novelty lies in our training procedure which we explain in detail in Section 3.1.

Testing: We now discuss the testing procedure for our setting. Let $\mathcal{T}_{\text{test}}$ denote the distribution over test tasks $m \in [M_{\text{test}}]$ at the time of testing. Let $\mathcal{D}_{\text{test}}$ denote a distribution over all possible interactions that can be generated by π^w during test time. At deployment time, the dataset $\mathcal{H}_m^0 \leftarrow \{\emptyset\}$ is initialized empty. At each round t, an action is sampled from the trained transformer model $I_t \sim \mathrm{TF}_{\boldsymbol{\Theta}}(\cdot \mid \mathcal{H}_m^t)$. The sampled action and resulting reward, r_t, are then added to $\mathcal{H}_m^t$ to form $\mathcal{H}_m^{t+1}$ and the process repeats for n total rounds. Finally, note that in this testing phase, the model parameter $\boldsymbol{\Theta}$ is not updated. Finally, the goal of the learner is to minimize cumulative regret for all task $m \in [M_{\text{test}}]$ defined as follows: $\mathbb{E}[R_n] = \frac{1}{M_{\text{test}}} \sum_{m=1}^{M_{\text{test}}} \sum_{t=1}^{n} \max_{a \in \mathcal{A}} f\left(\mathbf{x}(a), \boldsymbol{\theta}_{m,*}\right) - f\left(\mathbf{x}(I_t), \boldsymbol{\theta}_{m,*}\right)$.

2.3 Related In-context Learning Algorithms

In this section, we discuss related algorithms for in-context decision-making. For completeness, we describe the DPT and AD training procedure and algorithm now. During training, DPT first samples $m \sim \mathcal{T}_{\text{pre}}$ and then an in-context dataset $\mathcal{H}_m \sim \mathcal{D}_{\text{pre}}(\cdot|, m)$. It adds this $\mathcal{H}_m$ to the training dataset $\mathcal{H}_{\text{train}}$, and repeats to collect M_{pre} such training tasks. For each task m, DPT requires the optimal action $a_{m,*} = \arg\max_a f(\mathbf{x}(m,a), \boldsymbol{\theta}_{m,*})$ where $f(\mathbf{x}(m,a), \boldsymbol{\theta}_{m,*})$ is the expected reward for the action a in task m. Since the optimal action is usually not known in advance, in Section 4 we introduce a practical variant of DPT that approximates the optimal action with the best action identified during task interaction. During training DPT minimizes the cross-entropy loss:

$$\mathcal{L}_t^{\text{DPT}} = \text{cross-entropy}(\text{TF}_{\boldsymbol{\Theta}}(\cdot|\mathcal{H}_m^t), p(a_{m,*})) \quad (1)$$

where $p(a_{m,*}) \in \triangle^{\mathcal{A}}$ is a one-hot vector such that $p(j) = 1$ when $j = a_{m,*}$ and 0 otherwise. This loss is then back-propagated and used to update the model parameter $\boldsymbol{\Theta}$.

During test time evaluation for online setting the DPT selects $I_t \sim \text{softmax}_a^\tau(\text{TF}_{\boldsymbol{\Theta}}(\cdot|\mathcal{H}_m^t))$ where we define the $\text{softmax}_a^\tau(\mathbf{v})$ over a A dimensional vector $\mathbf{v} \in \mathbb{R}^A$ as $\text{softmax}_a^\tau(\mathbf{v}(a)) = \exp(\mathbf{v}(a)/\tau)/\sum_{a'=1}^{A} \exp(\mathbf{v}(a')/\tau)$ which produces a distribution over actions weighted by the temperature parameter $\tau > 0$. Therefore this sampling procedure has a high probability of choosing the predicted optimal action as well as induce sufficient exploration. In the online setting, the DPT observes the reward $r_t(I_t)$ which is added to $\mathcal{H}_m^t$. So the $\mathcal{H}_m$ during online testing consists of $\{I_t, r_t\}_{t=1}^n$ collected during testing. This interaction procedure is conducted for each test task $m \in [M_{\text{test}}]$. In the testing phase, the model parameter $\boldsymbol{\Theta}$ is not updated.

An alternative to DPT that does *not* require knowledge of the optimal action is the AD approach (Laskin et al., 2022; Lu et al., 2023). In AD, the learner aims to predict the next action of the demonstrator. So it minimizes the cross-entropy loss as follows:

$$\mathcal{L}_t^{\text{AD}} = \text{cross-entropy}(\text{TF}_{\boldsymbol{\Theta}}(\cdot|\mathcal{H}_m^t), p(I_{m,t})) \quad (2)$$

where $p(I_{m,t})$ is a one-hot vector such that $p(j) = 1$ when $j = I_{m,t}$ (the true action taken by the demonstrator) and 0 otherwise. At deployment time, AD selects $I_t \sim \text{softmax}_a^\tau(\text{TF}_{\boldsymbol{\Theta}}(\cdot|\mathcal{H}_m^t))$. The objective of AD is to match the performance of the demonstrator. In the next section, we introduce a new method that can improve upon the demonstrator without knowledge of the optimal action.

3 Proposed Algorithm PreDeToR

We now introduce our main algorithmic contribution, PreDeToR (which stands for **Pre**-trained **De**cision **T**ransf**o**rmer with **R**eward Estimation).

3.1 Pre-training Next Reward Prediction

The key idea behind PreDeToR is to leverage the in-context learning ability of transformers to infer the reward of each arm in a given test task. By training this in-context ability on a set of training tasks, the transformer can implicitly learn structure in the task family and exploit this structure to infer rewards without trying every single arm. Thus, in contrast to DPT and AD that output actions directly, PreDeToR outputs a scalar value reward prediction for each arm. To this effect, we append a linear layer of dimension A on top of a causal GPT2 model, denoted by $\text{TF}^{\mathbf{r}}{}_{\boldsymbol{\Theta}}(\cdot|\mathcal{H}_m)$, and use a least-squares loss to train the transformer to predict the reward for each action with these outputs. Note that we use $\text{TF}^{\mathbf{r}}{}_{\boldsymbol{\Theta}}(\cdot|\mathcal{H}_m)$ to denote a reward prediction transformer and $\text{TF}_{\boldsymbol{\Theta}}(\cdot|\mathcal{H}_m)$ as the transformer that predicts a distribution over actions (as in DPT and AD). At every round t the transformer predicts the *next reward* for each of the actions $a \in \mathcal{A}$ for the task m based on $\mathcal{H}_m^t = \{I_{m,s}, r_{m,s}\}_{s=1}^{t-1}$. This predicted reward is denoted by $\widehat{r}_{m,t+1}(a)$ for each $a \in \mathcal{A}$.

Loss calculation: For each training task, m, we calculate the loss at each round, t, using the transformer's prediction $\hat{r}_{m,t}(I_{m,t})$ and the actual observed reward $r_{m,t}$ that followed action $I_{m,t}$. We use a least-squares loss function:

$$\mathcal{L}_t = \left(\widehat{r}_{m,t}(I_{m,t}) - r_{m,t}\right)^2 \tag{3}$$

and hence minimizing this loss will minimize the mean squared-error of the transformer's predictions. The loss is calculated using (3) and is backpropagated to update the model parameter $\mathbf{\Theta}$.

Exploratory Demonstrator: Observe from the loss definition in (3) that it is calculated from the observed true reward and action from the dataset $\mathcal{H}_m$. In order for the transformer to learn accurate reward predictions during training, we require that the weak demonstrator is sufficiently exploratory such that it collects $\mathcal{H}_m$ such that $\mathcal{H}_m$ contains some reward $r_{m,t}$ for each action a. We discuss in detail the impact of the demonstrator on PreDeToR (-τ) training in Section 7.

3.2 Deploying PreDeToR

At deployment time, PreDeToR learns in-context to predict the mean reward of each arm on an unseen task and acts greedily with respect to this prediction. That is, at deployment time, a new task is sampled, $m \sim \mathcal{T}_{\text{test}}$, and the dataset $\mathcal{H}_m^0$ is initialized empty. Then at every round t, PreDeToR chooses $I_t = \arg\max_{a\in\mathcal{A}} \mathrm{TF}^{\mathbf{r}}_{\mathbf{\Theta}} \left(\widehat{r}_{m,t}(a) \mid \mathcal{H}_m^t\right)$ which is the action with the highest predicted reward and $\widehat{r}_{m,t}(a)$ is the predicted reward of action a. Note that PreDeToR is a greedy policy and thus may fail to conduct sufficient exploration. To remedy this potential limitation, we also introduce a soft variant, PreDeToR-τ that chooses $I_t \sim \mathrm{softmax}_a^\tau \left(\mathrm{TF}^{\mathbf{r}}_{\mathbf{\Theta}} \left(\widehat{\mathbf{r}}_{m,t}(a) \mid \mathcal{H}_m^t\right)\right)$. For both PreDeToR and PreDeToR-τ, the observed reward $r_t(I_t)$ is added to the dataset $\mathcal{H}_m$ and then used to predict the reward at the next round $t+1$. The full pseudocode of using PreDeToR for online interaction is shown in Algorithm 1. In Appendix A.14, we discuss how PreDeToR (-τ) can be deployed for offline learning.

Algorithm 1 Pre-trained **De**cision **T**ransf**o**rmer with **R**eward Estimation (PreDeToR)

Collecting Pretraining Dataset
Initialize empty pretraining dataset $\mathcal{H}_{\text{train}}$
for i in $[M_{\text{pre}}]$ **do**
  Sample task $m \sim \mathcal{T}_{\text{pre}}$, in-context dataset $\mathcal{H}_m \sim \mathcal{D}_{\text{pre}}(\cdot|m)$ and add this to $\mathcal{H}_{\text{train}}$.
end for
Pretraining model on dataset
Initialize model $\mathrm{TF}^{\mathbf{r}}_{\mathbf{\Theta}}$ with parameters $\mathbf{\Theta}$
while not converged **do**
  Sample $\mathcal{H}_m$ from $\mathcal{H}_{\text{train}}$ and predict $\widehat{r}_{m,t}$ for action $(I_{m,t})$ for all $t \in [n]$
  Compute loss in (3) with respect to $r_{m,t}$ and backpropagate to update model parameter $\mathbf{\Theta}$.
end while
Online test-time deployment
Sample unknown task $m \sim \mathcal{T}_{\text{test}}$ and initialize empty $\mathcal{H}_m^0 = \{\emptyset\}$
for $t = 1, 2, \ldots, n$ **do**
  Use $\mathrm{TF}^{\mathbf{r}}_{\mathbf{\Theta}}$ on m at round t to choose

$$I_t \begin{cases} = \arg\max_{a\in\mathcal{A}} \mathrm{TF}^{\mathbf{r}}_{\mathbf{\Theta}} \left(\widehat{r}_{m,t}(a) \mid \mathcal{H}_m^t\right), & \textbf{PreDeToR} \\ \sim \mathrm{softmax}_a^\tau \mathrm{TF}^{\mathbf{r}}_{\mathbf{\Theta}} \left(\widehat{r}_{m,t}(a) \mid \mathcal{H}_m^t\right), & \textbf{PreDeToR-}\tau \end{cases}$$

  Add $\{I_t, r_t\}$ to $\mathcal{H}_m^t$ to form $\mathcal{H}_m^{t+1}$.
end for

4 Empirical Study: Non-Linear Structure

Having introduced PreDeToR, we now investigate its performance in diverse bandit settings compared to other in-context learning algorithms. In our first set of experiments, we use a bandit setting with a common non-linear structure across tasks. Ideally, a good learner would leverage the structure, however, we choose the structure such that no existing algorithms are well-suited to the non-linear structure. This setting is thus a good testbed for establishing that in-context learning can discover and exploit common structure. Moreover, each task only consists of a few rounds of interactions. This setting is quite common in recommender settings where user interaction with the system lasts only for a few rounds and has an underlying non-linear structure (Kwon et al., 2022; Tomkins et al., 2020). We show that PreDeToR achieves lower regret than other in-context algorithms for the non-linear structured bandit setting. We study the performance of PreDeToR in the large horizon setting in Appendix A.8.

Baselines: We first discuss the baselines used in this setting.

(1) PreDeToR: This is our proposed method shown in Algorithm 1.

(2) PreDeToR-τ: This is the proposed exploratory method shown in Algorithm 1 and we fix $\tau = 0.05$.

(3) DPT-greedy: This baseline is the greedy approximation of the DPT algorithm from Lee et al. (2023) which is discussed in Section 2.3. Note that we choose DPT-greedy as a representative example of similar in-context decision-making algorithms studied in Lee et al. (2023); Sinii et al. (2023); Lin et al. (2023); Ma et al. (2023); Liu et al. (2023c;a) all of which require the optimal action (or its greedy approximation). DPT-greedy estimates the optimal arm using the reward estimates for each arm during each task.

(4) AD: This is the Algorithmic Distillation method (Laskin et al., 2022; Lu et al., 2021) discussed in Section 2.3.

(5) Thomp: This baseline is the celebrated stochastic A-action bandit Thompson Sampling algorithm from Thompson (1933); Agrawal & Goyal (2012); Russo et al. (2018); Zhu & Tan (2020). We choose Thomp as the weak demonstrator π^w as it does not make use of arm features. Thomp is also a stochastic algorithm that induces more exploration in the demonstrations.

(6) LinUCB: (Linear Upper Confidence Bound): This baseline is the Upper Confidence Bound algorithm for the linear bandit setting that leverages the linear structure and feature of the arms to select the most promising action as well as conducting exploration. We choose LinUCB as a baseline for each test task to show the limitations of algorithms that use linear feedback structure as an underlying assumption to select actions. Note that LinUCB requires oracle access to features to select actions per task.

(7) MLinGreedy: This is the multi-task linear regression bandit algorithm proposed by Yang et al. (2021). This algorithm assumes that there is a common low-dimensional feature extractor shared between the tasks and the reward of each task linearly depends on this feature extractor. We choose MLinGreedy as a baseline to show the limitations of algorithms that use linear feedback structure *across tasks* as an underlying assumption to select actions. Note that MLinGreedy requires oracle access to the action features to select actions as opposed to DPT, AD, and PreDeToR.

We describe in detail the baselines Thomp, LinUCB, and MLinGreedy for interested readers in Appendix A.2.2.

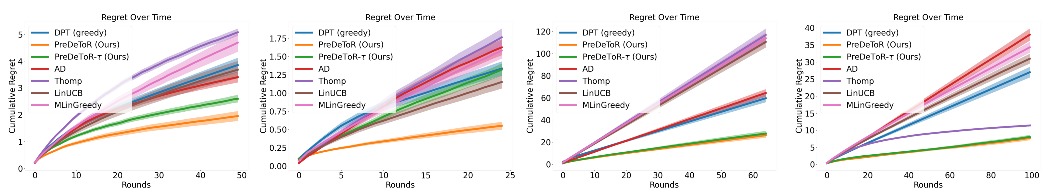
Outcomes: Before presenting the result we discuss the main outcomes from our experimental results in this section:

> **Finding 1:** PreDeToR (-τ) lowers regret compared to other baselines under unknown, non-linear structure. It learns to exploit the latent structure of the underlying tasks from in-context data even when it is trained without the optimal action $a_{m,*}$ (or its approximation) and without action features $\mathcal{X}$.

Experimental Result: These findings are reported in Figure 1. In Figure 1a we show the non-linear bandit setting for horizon $n = 50$, $M_{\text{pre}} = 100000$, $M_{\text{test}} = 200$, $A = 6$, and $d = 2$. The demonstrator π^w is the Thomp algorithm. We observe that PreDeToR (-τ) has lower cumulative regret than DPT-greedy. Note that for this low data regime (short horizon) the DPT-greedy does not have a good estimation of $\widehat{a}_{m,*}$ which results in a poor prediction of optimal action $\widehat{a}_{m,t,*}$. This results in higher regret. The PreDeToR (-τ) has lower regret than LinUCB, and MLinGreedy, which fail to perform well in this non-linear setting due to their algorithmic design and linear feedback assumption. Finally, PreDeToR-τ performs slightly better than PreDeToR in both settings as it conducts more exploration.

In Figure 1b we show the non-linear bandit setting for horizon $n = 25$, $M_{\text{pre}} = 100000$, $M_{\text{test}} = 200$, $A = 6$, and $d = 2$ where the norm of the $\boldsymbol{\theta}_{m,*}$ determines the reward of the actions which also is a non-linear function $\boldsymbol{\theta}_{m,*}$ and action features. This setting is similar to the wheel bandit setting of Riquelme et al. (2018). Again, we observe that PreDeToR has lower cumulative regret than all the other baselines.

Finally in Figure 1c and Figure 1d we show the performance of PreDeToR against other baselines in real-world datasets Movielens and Yelp. The Movielens dataset consists of more than 32 million ratings of 200,000 users and 80,000 movies (Harper & Konstan, 2015) where each entry consists of user-id, movie-id, rating, and timestamp. The Yelp dataset (Asghar, 2016) consists of ratings of 1300 business categories by 150,000 users. Each entry is summarized as user-id, business-id, rating, and timestamp. Previously structured bandit works (Deshpande & Montanari, 2012; Hong et al., 2023) directly fit a linear structure or low-rank factorization to estimate the $\boldsymbol{\theta}_{m,*}$ and simulate the ratings. However, we directly use the user-ids and movie-ids (or business-ids) to build a histogram of ratings per user and calculate the mean rating per movie (or business-id) per task. Define this as the $\{\mu_{m,a}\}_{a=1}^{A}$. This is then used to simulate the rating for n horizon per movie per task where the data collection algorithm is uniform sampling. Note that this does not require estimation of user or movie features, and PreDeToR (-τ) learns to exploit the latent structure of user-movie (or business) rating correlations directly from the data. From Figure 1c and Figure 1d we see that PreDeToR, and PreDeToR-τ outperform all the other baselines in these settings.



(a) Non-linear bandit (b) Feature bandit (c) Movielens (d) Yelp

Figure 1: Non-linear regime. The horizontal axis is the number of rounds. Confidence bars show one standard error.

5 Empirical Study: Linear Structure and Understanding the Exploration of PreDeToR

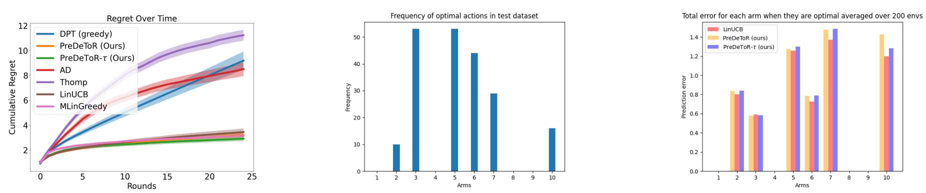
The previous experiments were conducted in a non-linear structured setting where we are unaware of a provably near-optimal algorithm. To assess how close PreDeToR's regret is to optimal, in this section, we consider a *linear* setting for which there exist well-understood algorithms (Abbasi-Yadkori et al., 2011; Lattimore & Szepesvári, 2020). Such algorithms provide a strong upper bound for PreDeToR. We summarize the key finding below:

Finding 2: PreDeToR (-τ) matches the performance of the optimal algorithm LinUCB in linear bandit setting as it learns to exploit the latent structure across tasks from in-context data and without access to features.

In Figure 2 we first show the linear bandit setting for horizon $n = 25$, $M_{\text{pre}} = 200000$, $M_{\text{test}} = 200$, $A = 10$, and $d = 2$. Note that the length of the context (the number of rounds) is an artifact of the transformer architecture and computational complexity. This is because the self-attention takes in as input a length-n sequence of tokens of size d, and requires $O\left(dn^2\right)$ time to compute the output (Keles et al., 2023). Further empirical setting details are stated in Appendix A.2.

We observe from Figure 2 that PreDeToR (-τ) has lower cumulative regret than DPT-greedy, and AD. Note that for this low data (short horizon) regime, the DPT-greedy does not have a good estimation of $\widehat{a}_{m,*}$ which results in a poor prediction of optimal action $\widehat{a}_{m,t,*}$. This results in higher regret. Observe that PreDeToR (-τ) performs quite similarly to LinUCB and lowers regret compared to Thomp which also shows that PreDeToR is able to exploit the latent linear structure and reward correlation of the underlying tasks. Note that LinUCB is close to the optimal algorithm for this linear bandit setting. PreDeToR outperforms AD as the main objective of AD is to match the performance of its demonstrator. In this short horizon, we see that MLinGreedy performs similarly to LinUCB.

We also show how the prediction error of the optimal action by PreDeToR is small compared to LinUCB in the linear bandit setting. In Figure 2b we first show how the 10 actions are distributed in the $M_{\text{test}} = 200$ test tasks. In Figure 2b for each bar, the frequency indicates the number of tasks where the action (shown in the x-axis) is the optimal action. Then, in Figure 2c, we show the prediction error of PreDeToR (-τ) for each task $m \in [M_{\text{test}}]$. The prediction error is calculated as $(\widehat{\mu}_{m,n,*}(a) - \mu_{m,*}(a))^2$ where $\widehat{\mu}_{m,n,*}(a) = \max_a \widehat{\boldsymbol{\theta}}_{m,n}^\top \mathbf{x}_m(a)$ is the empirical mean at the end of round n, and $\mu_{*,m}(a) = \max_a \boldsymbol{\theta}_{m,*}^\top \mathbf{x}_m(a)$ is the true mean of the optimal action in task m. Then we average the prediction error for the action $a \in [A]$ by the number of times the action a is the optimal action in some task m. From the Figure 2c, we see that for actions $\{2, 3, 5, 6, 7, 10\}$, the prediction error of PreDeToR is either close or smaller than LinUCB. Note that LinUCB estimates the empirical mean directly from the test task, whereas PreDeToR has a strong prior based on the training data. So PreDeToR is able to estimate the reward of the optimal action quite well from the training dataset $\mathcal{D}_{\text{pre}}$. This shows the power of PreDeToR to go beyond the in-context decision-making setting studied in Lee et al. (2023); Lin et al. (2023); Ma et al. (2023); Sinii et al. (2023); Liu et al. (2023c) which require long horizons/trajectories and optimal action during training to learn a near-optimal policy.



(a) Linear Bandit setting (b) Test action distribution (c) Test Prediction Error

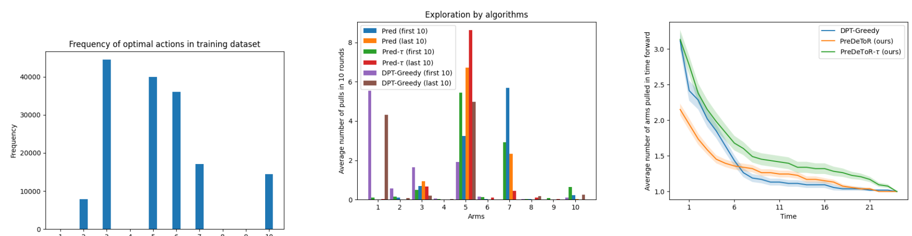
Figure 2: Linear Expt. The horizontal axis is the number of rounds. Confidence bars show one standard error.

We now state the main finding of our analysis of exploration in the linear bandit setting:

> **Finding 3:** The PreDeToR (-τ) has an implicit two-phase exploration. In the first phase, it explores with a strong prior over the in-context training data. In the second phase, once the task data has been observed for a few rounds (in-context) it switches to task-based exploration.

We first show in Figure 3a the training distribution of the optimal actions. For each bar, the frequency indicates the number of tasks where the action (shown in the x-axis) is the optimal action. Then in Figure 3b we show how the sampling distribution of DPT-greedy, PreDeToR and PreDeToR-τ change in the first 10 and last 10 rounds for all the tasks where action 5 is optimal. To plot this graph we first sum over the individual pulls of the action taken by each algorithm over the first 10 and last 10 rounds. Then we average these counts over all test tasks where action 5 is optimal. From the figure Figure 3b we see that PreDeToR(-τ) consistently pulls the action 5 more than DPT-greedy. It also explores other optimal actions like $\{2, 3, 6, 7, 10\}$ but discards them quickly in favor of the optimal action 5 in these tasks. This shows that PreDeToR (-τ) only considers the optimal actions seen from the training data. Once sufficient observation have been observed for the task it switches to task-based exploration and samples the optimal action more than DPT-greedy.

Finally, we plot the feasible action set considered by DPT-greedy, PreDeToR, and PreDeToR-τ in Figure 3c. To plot this graph again we consider the test tasks where the optimal action is 5. Then we count the number of distinct actions that are taken from round t up until horizon n. Finally we average this over all the considered tasks where the optimal action is 5. We call this the candidate action set considered by the algorithm. From the Figure 3c we see that DPT-greedy explores the least and gets stuck with few actions quickly (by round 10). Note that the actions DPT-greedy samples are sub-optimal and so it suffers a high cumulative regret (see Figure 2). PreDeToR explore slightly more than DPT-greedy, but PreDeToR-τ explores the most.



(a) Train Optimal Action Distribution

(b) Distribution of action sampling in all test tasks where action 5 is optimal

(c) Candidate Action Set in Time averaged over all tasks where action 5 is optimal

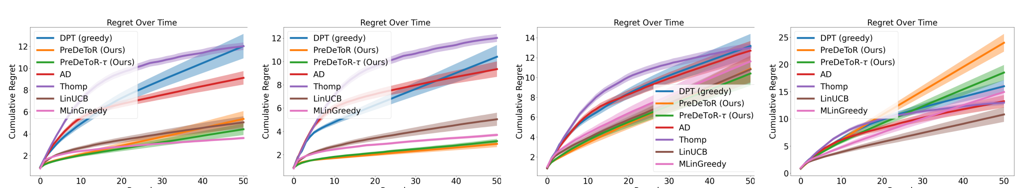
Figure 3: Exploration Analysis of PreDeToR(-τ)

6 Empirical Study: Importance of Shared Structure and Introducing New Actions

One of our central claims is that PreDeToR (-τ) internally learns and leverages the shared structure across the training and testing tasks. To validate this claim, in this section, we consider the introduction of new actions at test time that do *not* follow the structure of training time. These experiments are particularly important as they show the extent to which PreDeToR(-τ) is leveraging the latent structure and the shared correlation between the actions and rewards.

Invariant actions: We denote the set of actions fixed across the different tasks in the pretraining in-context dataset as $\mathcal{A}^{\text{inv}}$. Therefore these action features $\mathbf{x}(a) \in \mathbb{R}^d$ for $a \in \mathcal{A}^{\text{inv}}$ are fixed across the different tasks m. Note that these invariant actions help the transformer $\text{TF}_{\mathbf{w}}$ to learn the latent structure and the reward correlation across the different tasks. Therefore, as the structure breaks down, PreDeToR starts performing worse than other baselines.

New actions: We also want to test whether PreDeToR (-τ) exploits shared structure when new actions are introduced that are not seen during training time. To this effect, for each task $m \in [M_{\text{pre}}]$ and $m \in [M_{\text{test}}]$ we introduce $A - |\mathcal{A}^{\text{inv}}|$ new actions. *That is both for train and test tasks, we introduce new actions.* For each of these new actions $a \in [A - |\mathcal{A}^{\text{inv}}|]$ we choose the features $\mathbf{x}(m, a)$ randomly from $\mathcal{X} \subseteq \mathbb{R}^d$. Note the transformer now trains on a dataset $\mathcal{H}_m \subseteq \mathcal{D}_{\text{pre}} \neq \mathcal{D}_{\text{test}}$.

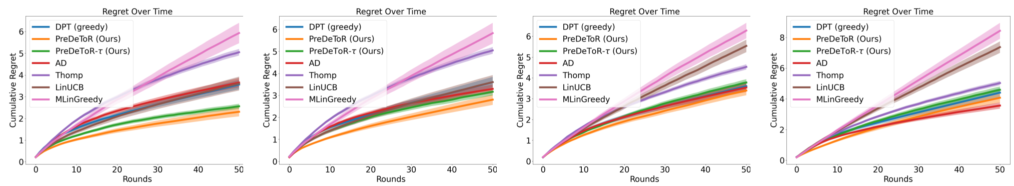


(a) 0 new action (b) 1 new action (c) 5 new actions (d) 10 new actions

Figure 4: Linear new action experiments. The horizontal axis is the number of rounds. Confidence bars show one standard error.

Baselines: We implement the same baselines discussed in Section 4.

Outcomes: Again before presenting the result we discuss the main outcomes from our experimental results of introducing new actions during data collection and evaluation:



(a) 0 new action (b) 1 new action (c) 5 new actions (d) 10 new actions

Figure 5: Non-linear new action experiments with non-linear setting.

Finding 4: Shared structure across the tasks is important to learn the reward structure.

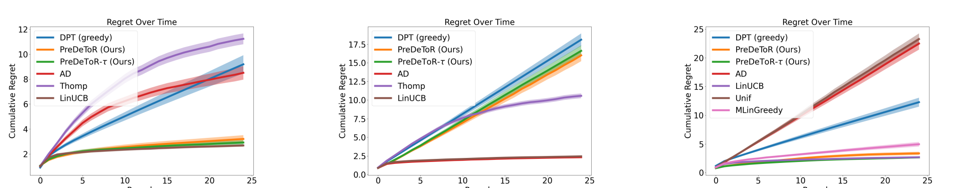
Experimental Result: We observe these outcomes in Figure 4 and Figure 5. We consider the linear and non-linear bandit setting of horizon $n = 50$, $M_{\text{pre}} = 100000$, $M_{\text{test}} = 200$, $A = 10$, and $d = 2$. Here during data collection and during collecting the test data, we randomly select between $0, 1, 5$, and 10 new actions from $\mathbb{R}^d$ for each task m. So the number of invariant actions is $|\mathcal{A}^{\text{inv}}| \in \{10, 5, 1, 0\}$. Again, the demonstrator π^w is the Thomp algorithm. From Figure 4a, 4b, 4c, and 4d, we observe that when the number of invariant actions is less than PreDeToR (-τ) has lower cumulative regret than DPT-greedy, and AD. Observe that PreDeToR (-τ) matches LinUCB and has lower regret than DPT-greedy, and AD when $\mathcal{A}^{\text{inv}}| \in \{10, 5, 1\}$. This shows that PreDeToR (-τ) is able to exploit the latent linear structure of the underlying tasks. However, as the number of invariant actions decreases we see that PreDeToR(-τ) performance drops and becomes similar to the unstructured bandits Thomp. We also show in Appendix A.3 that in K-armed bandit setting when there is no structure across arms PreDeToR (-τ) matches the performance of the demonstrator.

Similarly in Figure 5a, 5b, 5c, and 5d we show the performance of PreDeToR in the non-linear bandit setting. Observe that LinUCB, MLinGreedy fails to perform well in this non-linear setting due to their assumption of linear rewards. Again note that PreDeToR (-τ) has lower regret than DPT-greedy, and AD when $\mathcal{A}^{\text{inv}}| \in \{10, 1\}$. This shows that PreDeToR (-τ) is able to exploit the latent linear structure of the underlying tasks. However, as the number of invariant actions decreases we see that PreDeToR(-τ) performance drops and becomes similar to AD.

7 Data Collection Analysis

In this section, we analyze the performance of PreDeToR, PreDeToR-τ, DPT-greedy, AD, Thomp, and LinUCB when the weak demonstrator π^w is Thomp, LinUCB, or Uniform. We again consider the linear bandit setting discussed in Section 4. We show the cumulative regret by the above baselines in Figure 6a, 6b, and 6b when data is collected through Thomp, LinUCB, and Uniform respectively. We first state the main finding below:

Finding 5: The PreDeToR (-τ) excels in exploiting the underlying latent structure and reward correlation from in-context data when the data diversity is high.



(a) Thomp data collection (b) LinUCB data collection (c) Uniform data collection

Figure 6: Data Collection with various algorithms and Performance analysis

Experimental Result: We observe these outcomes in Figure 6. In Figure 6a we see that the A-actioned Thomp is explorative enough as it does not explore with the knowledge of feature representation. So it pulls the sub-optimal actions sufficiently high number of times before discarding them in favor of the optimal action. Therefore the training data is diverse enough so that PreDeToR ($-\tau$) can predict the reward vectors for actions sufficiently well. Consequently, PreDeToR ($-\tau$) almost matches the LinUCB algorithm. Both DPT-greedy and ADperform poorly in this setting.

In Figure 6b we see that the LinUCB algorithm is not explorative enough as it explores with the knowledge of feature representation and quickly discards the sub-optimal actions in favor of the optimal action. Therefore the training data is not diverse enough so that PreDeToR ($-\tau$) is not able to correctly predict the reward vectors for actions. Note that DPT-greedy also performs poorly in this setting when it is not provided with the optimal action information during training. The AD matches the performance of its demonstrator LinUCB because of its training procedure of predicting the next action of the demonstrator.

Finally, in Figure 6c we see that the A-armed Uniform is fully explorative as it does not intend to minimize regret (as opposed to Thomp) and does not explore with the knowledge of feature representation. Therefore the training data is very diverse which results in PreDeToR ($-\tau$) being able to predict the reward vectors for actions very well. Consequently, PreDeToR ($-\tau$) perfectly matches the LinUCB algorithm. Note that AD performs the worst as it matches the performance of its demonstrator whereas the performance of DPT-greedy suffers due to the lack of information on the optimal action during training.

We also empirically study the test performance of PreDeToR ($-\tau$) in K-armed bandit setting when there is no structure across arms in Appendix A.3, against the original DPT in Appendix A.3, in other *non-linear* bandit settings such as bilinear bandits (Appendix A.4), latent bandits (Appendix A.5), draw a connection between PreDeToR and Bayesian estimators (Appendix A.6), and perform sensitivity and ablation studies in Appendix A.7, A.9, A.10, A.11. Due to space constraints, we refer the interested reader to the relevant section in the appendices.

8 Theoretical Analysis of Generalization

In this section, we present a theoretical analysis of how PreDeToR-τ generalizes to an unknown target task given a set of source tasks. We observe that PreDeToR-τ's performance hinges on a low excess error on the predicted reward of the actions of the unknown target task based on the in-context data. Thus, in our analysis, we show that, in low-data regimes, PreDeToR-τ has a low expected excess risk for the unknown target task as the number of source tasks increases. This is summarized as follows:

Finding 6: PreDeToR ($-\tau$) has a low expected excess risk for the unknown target task as the number of source tasks increases. Moreover, the transfer learning risk of PreDeToR-τ (once trained on the M source tasks) scales with $\widetilde{O}(1/\sqrt{M})$.

To show this, we proceed as follows: Suppose we have the training data set $\mathcal{H}_{\text{all}} = \{\mathcal{H}_m\}_{m=1}^{M_{\text{pre}}}$, where the task $m \sim \mathcal{T}$ with a distribution $\mathcal{T}$ and the task data $\mathcal{H}_m$ is generated from a distribution $\mathcal{D}_{\text{pre}}(\cdot|m)$. For illustration purposes, here we consider the training data distribution $\mathcal{D}_{\text{pre}}(\cdot|m)$ where the actions are sampled following soft-LinUCB (a stochastic variant of LinUCB) (Chu et al., 2011). Given the loss function in Equation (3), we can define the task m training loss of PreDeToR-τ as $\widehat{\mathcal{L}}_m(\mathrm{TF}^{\mathbf{r}}{}_{\mathbf{\Theta}}) = \frac{1}{n}\sum_{t=1}^{n} \ell(r_{m,t}, \mathrm{TF}^{\mathbf{r}}{}_{\mathbf{\Theta}}(\widehat{r}_{m,t}(I_{m,t})|\mathcal{H}_m^t)) = \frac{1}{n}\sum_{t=1}^{n}(\mathrm{TF}^{\mathbf{r}}{}_{\mathbf{\Theta}}(\widehat{r}_{m,t}(I_{m,t})|\mathcal{H}_m^t) - r_{m,t})^2$. We drop the notation $\mathbf{\Theta}, \mathbf{r}$ from $\mathrm{TF}^{\mathbf{r}}{}_{\mathbf{\Theta}}$ for simplicity and let $M = M_{\text{pre}}$. We define

$$\widehat{\mathrm{TF}} = \underset{\mathrm{TF}\in\mathrm{Alg}}{\arg\min}\, \widehat{\mathcal{L}}_{\mathcal{H}_{\text{all}}}(\mathrm{TF}) := \frac{1}{M}\sum_{m=1}^{M} \widehat{\mathcal{L}}_m(\mathrm{TF}), \quad \text{(ERM)} \tag{4}$$

where Alg denotes the space of algorithms induced by the TF. Let $\mathcal{L}_m(\mathrm{TF}) = \mathbb{E}_{\mathcal{H}_m}\left[\widehat{\mathcal{L}}_m(\mathrm{TF})\right]$ and $\mathcal{L}_{\mathrm{MTL}}(\mathrm{TF}) = \mathbb{E}\left[\widehat{\mathcal{L}}_{\mathcal{H}_{\text{all}}}(\mathrm{TF})\right] = \frac{1}{M}\sum_{m=1}^{M}\mathcal{L}_m(\mathrm{TF})$ be the corresponding population risks. For the ERM in (4), we want to bound the following excess Multi-Task Learning (MTL) risk of PreDeToR-τ

$$\mathcal{R}_{\mathrm{MTL}}(\widehat{\mathrm{TF}}) = \mathcal{L}_{\mathrm{MTL}}(\widehat{\mathrm{TF}}) - \min_{\mathrm{TF}\in\mathrm{Alg}} \mathcal{L}_{\mathrm{MTL}}(\mathrm{TF}). \tag{5}$$

Note that for in-context learning, a training sample (I_t, r_t) impacts all future decisions of the algorithm from time step $t+1$ to n. Therefore, we need to control the stability of the input perturbation of the learning algorithm learned by the transformer. We introduce the following stability condition.

Assumption 8.1. (Error stability (Bousquet & Elisseeff, 2002; Li et al., 2023)). Let $\mathcal{H} = (I_t, r_t)_{t=1}^n$ be a sequence in $[A] \times [0,1]$ with $n \geq 1$ and $\mathcal{H}'$ be the sequence where the t'th sample of $\mathcal{H}$ is replaced by (I'_t, r'_t). Error stability holds for a distribution $(I, r) \sim \mathcal{D}$ if there exists a $K > 0$ such that for any $\mathcal{H}, (I'_t, r'_t) \in ([A] \times [0,1]), t \leq n$, and $\mathrm{TF} \in \mathrm{Alg}$, we have

$$\left|\mathbb{E}_{(I,r)}\left[\ell(r, \mathrm{TF}(\widehat{r}(I)|\mathcal{H})) - \ell\left(r, \mathrm{TF}(\widehat{r}(I)|\mathcal{H}')\right)\right]\right| \leq \frac{K}{n}.$$

Let ρ be a distance metric on Alg. Pairwise error stability holds if for all $\mathrm{TF}, \mathrm{TF}' \in \mathrm{Alg}$ we have

$$\left|\mathbb{E}_{(x,y)}\left[\ell(r, \mathrm{TF}(\widehat{r}(I)|\mathcal{H})) - \ell\left(r, \mathrm{TF}'(\widehat{r}(I)|\mathcal{H})\right) - \ell(r, \mathrm{TF}(\widehat{r}(I)|\mathcal{H}')) + \ell\left(r, \mathrm{TF}'(\widehat{r}(I)|\mathcal{H}')\right)\right]\right| \leq \frac{K\rho(\mathrm{TF}, \mathrm{TF}')}{n}.$$

Now we present the Multi-task learning (MTL) risk of PreDeToR-τ.

Theorem 8.2. ***(PreDeToR risk)*** *Suppose error stability Assumption 8.1 holds and assume loss function $\ell(\cdot,\cdot)$ is C-Lipschitz for all $r_t \in [0, B]$ and horizon $n \geq 1$. Let $\widehat{\mathrm{TF}}$ be the empirical solution of (ERM) and $\mathcal{N}(\mathcal{A}, \rho, \epsilon)$ be the covering number of the algorithm space* Alg *following Definition C.2 and C.3. Then with probability at least $1-2\delta$, the excess MTL risk of PreDeToR-τ is bounded by*

$$\mathcal{R}_{\mathrm{MTL}}(\widehat{\mathrm{TF}}) \leq 4\frac{C}{\sqrt{nM}} + 2(B + K\log n)\sqrt{\frac{\log(\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)/\delta)}{cnM}},$$

where $\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)$ is the covering number of transformer $\widehat{\mathrm{TF}}$ and $\epsilon = 1/\sqrt{nM}$.

The proof of Theorem 8.2 is provided in Appendix C.1. From Theorem 8.2 we see that in low-data regime with a small horizon n, as the number of tasks M increases the MTL risk decreases. We further discuss the stability factor K and covering number $\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)$ in Remark C.4, and C.5.

We now present the transfer learning risk of PreDeToR-τ for an unknown target task $g \sim \mathcal{T}$ with the test dataset $\mathcal{H}_g \sim \mathcal{D}_{\text{test}}(\cdot|g)$. Note that the test data distribution $\mathcal{D}_{\text{test}}(\cdot|g)$ is such that the actions are sampled following soft-LinUCB.

Theorem 8.3. ***(Transfer risk)*** *Consider the setting of Theorem 8.2 and assume the training source tasks are independently drawn from task distribution $\mathcal{T}$. Let $\widehat{TF}$ be the empirical solution of (ERM) and $g \sim \mathcal{T}$. Define the expected excess transfer learning risk $\mathbb{E}_g[\mathcal{R}_g] = \mathbb{E}_g\left[\mathcal{L}_g(\widehat{\mathrm{TF}})\right] - \arg\min_{\mathrm{TF}\in\mathrm{Alg}} \mathbb{E}_g\left[\mathcal{L}_g(\mathrm{TF})\right]$. Then with probability at least $1-2\delta$, the $\mathbb{E}_g\left[\mathcal{R}_g\right] \leq 4\frac{C}{\sqrt{M}} + 2B\sqrt{\frac{\log(\mathcal{N}(\mathrm{Alg},\rho,\varepsilon)/\delta)}{M}}$, where $\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)$ is the covering number of $\widehat{\mathrm{TF}}$ and $\epsilon = \frac{1}{\sqrt{M}}$.*

The proof is given in Appendix C.2. This shows that for the transfer learning risk of PreDeToR-τ (once trained on the M source tasks) scales with $\widetilde{O}(1/\sqrt{M})$. This is because the unseen target task $g \sim \mathcal{T}$ induces a distribution shift, which, typically, cannot be mitigated with more samples n per task. A similar observation is provided in Lin et al. (2023). We further discuss this in Remark C.7. We also observe a similar phenomenon empirically; see the discussion in Appendix A.13.

9 Conclusions, Limitations and Future Works

In this paper, we studied the supervised pretraining of decision transformers in the multi-task structured bandit setting when the knowledge of the optimal action is unavailable. Our proposed methods PreDeToR (-τ) do not need to know the action representations or the reward structure and learn these with the help of offline data. PreDeToR (-τ) predict the reward for the next action of each action during pretraining and can generalize well in-context in several regimes spanning low-data, new actions, and structured bandit settings like linear, non-linear, bilinear, latent bandits. The PreDeToR (-τ) outperforms other in-context algorithms like AD, DPT-greedy in most of the experiments. Finally, we theoretically analyze PreDeToR-τ and show that pretraining it in M source tasks leads to a low expected excess error on a target task drawn from the same task distribution $\mathcal{T}$. In the future, we want to extend our PreDeToR (-τ) to the MDP setting (Sutton & Barto, 2018; Agarwal et al., 2019), and constrained MDP setting (Efroni et al., 2020; Gu et al., 2022).

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32, 2019.

Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pp. 39–1. JMLR Workshop and Conference Proceedings, 2012.

Nabiha Asghar. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362*, 2016.

Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2):235–256, May 2002. ISSN 1573-0565. DOI: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche ..., 1990.

C Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:531–537, 2006.

Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

George EP Box and George C Tiao. *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.

David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? *Advances in Neural Information Processing Systems*, 35:1542–1553, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Sébastien Bubeck, Gilles Stoltz, Csaba Szepesvári, and Rémi Munos. Online optimization in x-armed bandits. *Advances in Neural Information Processing Systems*, 21, 2008.

Sébastien Bubeck, Gilles Stoltz, and Jia Yuan Yu. Lipschitz bandits without the lipschitz constant. In *Algorithmic Learning Theory: 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011. Proceedings 22*, pp. 144–158. Springer, 2011.

Bradley P Carlin and Thomas A Louis. *Bayesian methods for data analysis*. CRC press, 2008.

Kamalika Chaudhuri, Prateek Jain, and Nagarajan Natarajan. Active heteroscedastic regression. In *International Conference on Machine Learning*, pp. 694–702. PMLR, 2017.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pp. 844–853. PMLR, 2017.

Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 208–214. JMLR Workshop and Conference Proceedings, 2011.

Zhongxiang Dai, Yao Shu, Arun Verma, Flint Xiaofeng Fan, Bryan Kian Hsiang Low, and Patrick Jaillet. Federated neural bandit. *arXiv preprint arXiv:2205.14309*, 2022.

Rémy Degenne, Pierre Ménard, Xuedong Shang, and Michal Valko. Gamification of pure exploration for linear bandits. In *International Conference on Machine Learning*, pp. 2432–2442. PMLR, 2020.

Yash Deshpande and Andrea Montanari. Linear bandits in high dimension and recommendation systems. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1750–1754. IEEE, 2012.

Kefan Dong, Jiaqi Yang, and Tengyu Ma. Provable model-based nonlinear bandit and reinforcement learning: Shelve optimism, embrace virtual curvature. *Advances in neural information processing systems*, 34:26168–26182, 2021.

Yihan Du, Longbo Huang, and Wen Sun. Multi-task representation learning for pure exploration in linear bandits. *arXiv preprint arXiv:2302.04441*, 2023.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl 2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Yonathan Efroni, Shie Mannor, and Matteo Pirotta. Exploration-exploitation in constrained mdps. *arXiv preprint arXiv:2003.02189*, 2020.

Valerii Vadimovich Fedorov. *Theory of optimal experiments*. Elsevier, 2013.

Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. Parametric bandits: The generalized linear case. *Advances in neural information processing systems*, 23, 2010.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11 (3-4):219–354, 2018.

Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4019–4026. IEEE, 2016.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Yao Ge, Yuting Guo, Yuan-Chi Yang, Mohammed Ali Al-Garadi, and Abeed Sarker. Few-shot learning for medical text: A systematic review. *arXiv preprint arXiv:2204.14081*, 2022.

Kamyar Ghasemipour, Shixiang Shane Gu, and Ofir Nachum. Why so pessimistic? estimating uncertainties for offline rl through ensembles, and why their independence matters. *Advances in Neural Information Processing Systems*, 35:18267–18281, 2022.

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in neural information processing systems*, 31, 2018.

Samarth Gupta, Shreyas Chaudhari, Subhojyoti Mukherjee, Gauri Joshi, and Osman Yağan. A unified approach to translate classical bandit algorithms to the structured bandit setting. *IEEE Journal on Selected Areas in Information Theory*, 1(3):840–853, 2020.

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

Joey Hong, Branislav Kveton, Manzil Zaheer, Yinlam Chow, Amr Ahmed, and Craig Boutilier. Latent bandits revisited. *Advances in Neural Information Processing Systems*, 33:13423–13433, 2020.

Joey Hong, Branislav Kveton, Sumeet Katariya, Manzil Zaheer, and Mohammad Ghavamzadeh. Deep hierarchy in bandits. In *International Conference on Machine Learning*, pp. 8833–8851. PMLR, 2022a.

Joey Hong, Branislav Kveton, Manzil Zaheer, and Mohammad Ghavamzadeh. Hierarchical bayesian bandits. In *International Conference on Artificial Intelligence and Statistics*, pp. 7724–7741. PMLR, 2022b.

Joey Hong, Branislav Kveton, Manzil Zaheer, Sumeet Katariya, and Mohammad Ghavamzadeh. Multi-task off-policy learning from bandit feedback. In *International Conference on Machine Learning*, pp. 13157–13173. PMLR, 2023.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Yiding Jiang, Evan Liu, Benjamin Eysenbach, J Zico Kolter, and Chelsea Finn. Learning options via compression. *Advances in Neural Information Processing Systems*, 35:21184–21199, 2022.

Richard Arnold Johnson, Dean W Wichern, et al. Applied multivariate statistical analysis. 2002.

Kwang-Sung Jun, Rebecca Willett, Stephen Wright, and Robert Nowak. Bilinear bandits with low-rank structure. In *International Conference on Machine Learning*, pp. 3163–3172. PMLR, 2019.

Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. Predicting the computational cost of deep learning models. In *2018 IEEE international conference on big data (Big Data)*, pp. 3873–3882. IEEE, 2018.

Yue Kang, Cho-Jui Hsieh, and Thomas Chun Man Lee. Efficient frameworks for generalized low-rank matrix bandit problems. *Advances in Neural Information Processing Systems*, 35:19971–19983, 2022.

Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention. In *International Conference on Algorithmic Learning Theory*, pp. 597–619. PMLR, 2023.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Branislav Kveton, Csaba Szepesvári, Anup Rao, Zheng Wen, Yasin Abbasi-Yadkori, and S Muthukrishnan. Stochastic low-rank bandits. *arXiv preprint arXiv:1712.04644*, 2017.

Jeongyeol Kwon, Yonathan Efroni, Constantine Caramanis, and Shie Mannor. Tractable optimality in episodic latent mabs. *Advances in Neural Information Processing Systems*, 35:23634–23645, 2022.

Nicholas C Landolfi, Garrett Thomas, and Tengyu Ma. A model-based approach for sample-efficient multi-task reinforcement learning. *arXiv preprint arXiv:1907.04964*, 2019.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Tor Lattimore and Csaba Szepesvári. An information-theoretic approach to minimax regret in partial monitoring. In *Conference on Learning Theory*, pp. 2111–2139. PMLR, 2019.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Jonathan N Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*, 2023.

Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936, 2022.

Lanqing Li, Rui Yang, and Dijun Luo. Focal: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. *arXiv preprint arXiv:2010.01112*, 2020.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

Lihong Li, Yu Lu, and Dengyong Zhou. Provably optimal algorithms for generalized linear contextual bandits. In *International Conference on Machine Learning*, pp. 2071–2080. PMLR, 2017.

Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pp. 19565–19594. PMLR, 2023.

Licong Lin, Yu Bai, and Song Mei. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. *arXiv preprint arXiv:2310.08566*, 2023.

Evan Z Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International conference on machine learning*, pp. 6925–6935. PMLR, 2021.

Xiaoqian Liu, Jianbin Jiao, and Junge Zhang. Self-supervised pretraining for decision foundation model: Formulation, pipeline and challenges. *arXiv preprint arXiv:2401.00031*, 2023a.

Xin Liu, Daniel McDuff, Geza Kovacs, Isaac Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, and Shwetak Patel. Large language models are few-shot health learners. *arXiv preprint arXiv:2305.15525*, 2023b.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch off-policy reinforcement learning without great exploration. *Advances in neural information processing systems*, 33:1264–1274, 2020.

Zhihan Liu, Hao Hu, Shenao Zhang, Hongyi Guo, Shuqi Ke, Boyi Liu, and Zhaoran Wang. Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency. *arXiv preprint arXiv:2309.17382*, 2023c.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *arXiv preprint arXiv:2303.03982*, 2023.

Yangyi Lu, Amirhossein Meisami, and Ambuj Tewari. Low-rank generalized linear bandit problems. In *International Conference on Artificial Intelligence and Statistics*, pp. 460–468. PMLR, 2021.

Yi Ma, Chenjun Xiao, Hebin Liang, and Jianye Hao. Rethinking decision transformer via hierarchical reinforcement learning. *arXiv preprint arXiv:2311.00267*, 2023.

Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. Few-shot bot: Prompt-based learning for dialogue systems. *arXiv preprint arXiv:2110.08118*, 2021.

Stefan Magureanu, Richard Combes, and Alexandre Proutiere. Lipschitz bandits: Regret lower bound and optimal algorithms. In *Conference on Learning Theory*, pp. 975–999. PMLR, 2014.

Odalric-Ambrym Maillard and Shie Mannor. Latent bandits. In *International Conference on Machine Learning*, pp. 136–144. PMLR, 2014.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pp. 7780–7791. PMLR, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Subhojyoti Mukherjee, Qiaomin Xie, Josiah P Hanna, and Robert Nowak. Multi-task representation learning for pure exploration in bilinear bandits. *arXiv preprint arXiv:2311.00327*, 2023.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071*, 2017.

Soumyabrata Pal, Arun Sai Suggala, Karthikeyan Shanmugam, and Prateek Jain. Optimal algorithms for latent bandits with cluster structure. In *International Conference on Artificial Intelligence and Statistics*, pp. 7540–7577. PMLR, 2023.

Theodore J Perkins and Doina Precup. Using options for knowledge transfer in reinforcement learning title2, 1999.

Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. In *International Conference on Machine Learning*, pp. 17811–17829. PMLR, 2022.

Friedrich Pukelsheim. *Optimal design of experiments*. SIAM, 2006.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

Adam Roberts, Colin Raffel, Katherine Lee, Michael Matena, Noam Shazeer, Peter J Liu, Sharan Narang, Wei Li, and Yanqi Zhou. Exploring the limits of transfer learning with a unified text-to-text transformer. 2019.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

Tom Schaul and Jürgen Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, 2010.

Sina Semnani, Violet Yao, Heidi Zhang, and Monica Lam. Wikichat: Stopping the hallucination of large language model chatbots by few-shot grounding on wikipedia. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2387–2413, 2023.

Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Viacheslav Sinii, Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, and Sergey Kolesnikov. In-context reinforcement learning for variable action spaces. *arXiv preprint arXiv:2312.13327*, 2023.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Sabina Tomkins, Peng Liao, Predrag Klasnja, Serena Yeung, and Susan Murphy. Rapidly personalizing mobile health treatment policies with limited data. *arXiv preprint arXiv:2002.09971*, 2020.

Michal Valko, Nathaniel Korda, Rémi Munos, Ilias Flaounas, and Nelo Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Adam X Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. Bayesian low-rank adaptation for large language models. *arXiv preprint arXiv:2308.13111*, 2023.

Jiaqi Yang, Wei Hu, Jason D Lee, and Simon S Du. Impact of representation learning in linear bandits. *arXiv preprint arXiv:2010.06531*, 2020.

Jiaqi Yang, Wei Hu, Jason D Lee, and Simon Shaolei Du. Impact of representation learning in linear bandits. In *International Conference on Learning Representations*, 2021.

Jiaqi Yang, Qi Lei, Jason D Lee, and Simon S Du. Nearly minimax algorithms for linear bandits with shared representation. *arXiv preprint arXiv:2203.15664*, 2022a.

Lin Yang and Mengdi Wang. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning*, pp. 10746–10756. PMLR, 2020.

Mengjiao Yang, Dale Schuurmans, Pieter Abbeel, and Ofir Nachum. Dichotomy of control: Separating what you can control from what you cannot. *arXiv preprint arXiv:2210.13435*, 2022b.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023.

Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pp. 11492–11502. PMLR, 2020.

Qiuyu Zhu and Vincent Tan. Thompson sampling algorithms for mean-variance bandits. In *International Conference on Machine Learning*, pp. 11599–11608. PMLR, 2020.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

A Appendix

A.1 Related Works

In this section, we briefly discuss related works.

In-context decision making (Laskin et al., 2022; Lee et al., 2023) has emerged as an attractive alternative in Reinforcement Learning (RL) compared to updating the model parameters after collection of new data (Mnih et al., 2013; François-Lavet et al., 2018). In RL the contextual data takes the form of state-action-reward tuples representing a dataset of interactions with an unknown environment (task). In this paper, we will refer to this as the in-context data. Recall that in many real-world settings, the underlying task can be structured with correlated features, and the reward can be highly non-linear. So specialized bandit algorithms fail to learn in these tasks. To circumvent this issue, a learner can first collect in-context data consisting of just action indices I_t and rewards r_t. Then it can leverage the representation learning capability of deep neural networks to learn a pattern across the in-context data and subsequently derive a near-optimal policy (Lee et al., 2023; Mirchandani et al., 2023). We refer to this learning framework as an in-context decision-making setting.

The in-context decision-making setting of Sinii et al. (2023) also allows changing the action space by learning an embedding over the action space yet also requires the optimal action during training. In contrast we do not require the optimal action as well as show that we can generalize to new actions without learning an embedding over them. Similarly, Lin et al. (2023) study the in-context decision-making setting of Laskin et al. (2022); Lee et al. (2023), but they also require a greedy approximation of the optimal action. The Ma et al. (2023) also studies a similar setting for hierarchical RL where they stitch together sub-optimal trajectories and predict the next action during test time. Similarly, Liu et al. (2023c) studies the in-context decision-making setting to predict action instead of learning a reward correlation from a short horizon setting. In contrast we do not require a greedy approximation of the optimal action, deal with short horizon setting and changing action sets during training and testing, and predict the estimated means of the actions instead of predicting the optimal action. A survey of the in-context decision-making approaches can be found in Liu et al. (2023a).

In the in-context decision-making setting, the learning model is first trained on supervised input-output examples with the in-context data during training. Then during test time, the model is asked to complete a new input (related to the context provided) without any update to the model parameters (Xie et al., 2021; Min et al., 2022). Motivated by this, Lee et al. (2023) recently proposed the Decision Pretrained Transformers (DPT) that exhibit the following properties: **(1)** During supervised pretraining of DPT, predicting optimal actions alone gives rise to near-optimal decision-making algorithms for unforeseen task during test time. Note that DPT does not update model parameters during test time and, therefore, conducts in-context learning on the unforeseen task. **(2)** DPT improves over the in-context data used to pretrain it by exploiting latent structure. However, DPT either requires the optimal action during training or if it needs to approximate the optimal action. For approximating the optimal action, it requires a large amount of data from the underlying task.

At the same time, learning the underlying data pattern from a few examples during training is becoming more relevant in many domains like chatbot interaction (Madotto et al., 2021; Semnani et al., 2023), recommendation systems, healthcare (Ge et al., 2022; Liu et al., 2023b), etc. This is referred to as few-shot learning. However, most current RL decision-making systems (including in-context learners like DPT) require an enormous amount of data to learn a good policy.

The in-context learning framework is related to the meta-learning framework (Bengio et al., 1990; Schaul & Schmidhuber, 2010). Broadly, these techniques aim to learn the underlying latent shared structure within the training distribution of tasks, facilitating faster learning of novel tasks during test time. In the context of decision-making and reinforcement learning (RL), there exists a frequent choice regarding the specific 'structure' to be learned, be it the task dynamics (Fu et al., 2016; Nagabandi et al., 2018; Landolfi et al., 2019), a task context identifier (Rakelly et al., 2019; Zintgraf et al., 2019; Liu et al., 2021), or temporally extended skills and options (Perkins & Precup, 1999; Gupta et al., 2018; Jiang et al., 2022).

However, as we noted in the Section 1, one can do a greedy approximation of the optimal action from the historical data using a weak demonstrator and a neural network policy (Finn et al., 2017; Rothfuss et al., 2018). Moreover, the in-context framework generally is more agnostic where it learns the policy of the demonstrator (Duan et al., 2016; Wang et al., 2016; Mishra et al., 2017). Note that both DPT-greedy and PreDeToR are different than algorithmic distillation (Laskin et al., 2022; Lu et al., 2023) as they do not distill an existing RL algorithm. moreover, in contrast to DPT-greedy which is trained to predict the optimal action, the PreDeToR is trained to predict the reward for each of the actions. This enables the PreDeToR (similar to DPT-greedy) to show to potentially emergent online and offline strategies at test time that automatically align with the task structure, resembling posterior sampling.

As we discussed in the Section 1, in decision-making, RL, and imitation learning the transformer models are trained using autoregressive action prediction (Yang et al., 2023). Similar methods have also been used in Large language models (Vaswani et al., 2017; Roberts et al., 2019). One of the more notable examples is the Decision Transformers (abbreviated as DT) which utilizes a transformer to autoregressively model sequences of actions from offline experience data, conditioned on the achieved return (Chen et al., 2021; Janner et al., 2021). This approach has also been shown to be effective for multi-task settings (Lee et al., 2022), and multi-task imitation learning with transformers (Reed et al., 2022; Brohan et al., 2022; Shafiullah et al., 2022). However, the DT methods are not known to improve upon their in-context data, which is the main thrust of this paper (Brandfonbrener et al., 2022; Yang et al., 2022b).

Our work is also closely related to the offline RL setting. In offline RL, the algorithms can formulate a policy from existing data sets of state, action, reward, and next-state interactions. Recently, the idea of pessimism has also been introduced in an offline setting to address the challenge of distribution shift (Kumar et al., 2020; Yu et al., 2021; Liu et al., 2020; Ghasemipour et al., 2022). Another approach to solve this issue is policy regularization (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019; Siegel et al., 2020; Liu et al., 2019), or reuse data for related task (Li et al., 2020; Mitchell et al., 2021), or additional collection of data along with offline data (Pong et al., 2022). However, all of these approaches still have to take into account the issue of distributional shifts. In contrast PreDeToR and DPT-greedy leverages the decision transformers to avoid these issues. Both of these methods can also be linked to posterior sampling. Such connections between sequence modeling with transformers and posterior sampling have also been made in Chen et al. (2021); Müller et al. (2021); Lee et al. (2023); Yang et al. (2023).

A.2 Experimental Setting Information and Details of Baselines

In this section, we describe in detail the experimental settings and some baselines.

A.2.1 Experimental Details

Linear Bandit: We consider the setting when $f(\mathbf{x}, \boldsymbol{\theta}_*) = \mathbf{x}^\top \boldsymbol{\theta}_*$. Here $\mathbf{x} \in \mathbb{R}^d$ is the action feature and $\boldsymbol{\theta}_* \in \mathbb{R}^d$ is the hidden parameter. For every experiment, we first generate tasks from $\mathcal{T}_{\text{pre}}$. Then we sample a fixed set of actions from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d/d)$ in $\mathbb{R}^d$ and this constitutes the features. Then for each task $m \in [M]$ we sample $\boldsymbol{\theta}_{m,*} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d/d)$ to produce the means $\mu(m, a) = \langle \boldsymbol{\theta}_{m,*}, \mathbf{x}(m, a) \rangle$ for $a \in \mathcal{A}$ and $m \in [M]$. Finally, note that we do not shuffle the data as the order matters. Also in this setting $\mathbf{x}(m, a)$ for each $a \in \mathcal{A}$ is fixed for all tasks m.

Non-Linear Bandit: We now consider the setting when $f(\mathbf{x}, \boldsymbol{\theta}_*) = 1/(1 + 0.5 \cdot \exp(2 \cdot \exp(-\mathbf{x}^\top \boldsymbol{\theta}_*)))$. Again, here $\mathbf{x} \in \mathbb{R}^d$ is the action feature, and $\boldsymbol{\theta}_* \in \mathbb{R}^d$ is the hidden parameter. Note that this is different than the generalized linear bandit setting (Filippi et al., 2010; Li et al., 2017). Again for every experiment, we first generate tasks from $\mathcal{T}_{\text{pre}}$. Then we sample a fixed set of actions from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d/d)$ in $\mathbb{R}^d$ and this constitutes the features. Then for each task $m \in [M]$ we sample $\boldsymbol{\theta}_{m,*} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d/d)$ to produce the means $\mu(m, a) = 1/(1+0.5\cdot\exp(2\cdot\exp(-\mathbf{x}(m, a)^\top \boldsymbol{\theta}_{m,*})))$ for $a \in \mathcal{A}$ and $m \in [M]$. Again note that in this setting $\mathbf{x}(m, a)$ for each $a \in \mathcal{A}$ is fixed for all tasks m.

We use NVIDIA GeForce RTX 3090 GPU with 24GB RAM to load the GPT 2 Large Language Model. This requires less than 2GB RAM without data, and with large context may require as much as 20GB RAM.

A.2.2 Details of Baselines

(1) Thomp: This baseline is the stochastic A-action bandit Thompson Sampling algorithm from Thompson (1933); Agrawal & Goyal (2012); Russo et al. (2018); Zhu & Tan (2020). We briefly describe the algorithm below: At every round t and each action a, Thomp samples $\gamma_{m,t}(a) \sim \mathcal{N}(\widehat{\mu}_{m,t-1}(a), \sigma^2/N_{m,t-1}(a))$, where $N_{m,t-1}(a)$ is the number of times the action a has been selected till $t-1$, and $\widehat{\mu}_{m,t-1}(a) = \frac{\sum_{s=1}^{t-1} \widehat{r}_{m,s} \mathbf{1}(I_s=a)}{N_{m,t-1}(a)}$ is the empirical mean. Then the action selected at round t is $I_t = \arg\max_a \gamma_{m,t}(a)$. Observe that Thomp is not a deterministic algorithm like UCB (Auer et al., 2002). So we choose Thomp as the weak demonstrator π^w because it is more exploratory than UCB and also chooses the optimal action, $a_{m,*}$, a sufficiently large number of times. Thomp is a weak demonstrator as it does not have access to the feature set $\mathcal{X}$ for any task m.

(2) LinUCB: (Linear Upper Confidence Bound): This baseline is the Upper Confidence Bound algorithm for the linear bandit setting that selects the action I_t at round t for task m that is most optimistic and reduces the uncertainty of the task unknown parameter $\boldsymbol{\theta}_{m,*}$. To balance exploitation and exploration between choosing different items the LinUCB computes an upper confidence value to the estimated mean of each action $\mathbf{x}_{m,a} \in \mathcal{X}$. This is done as follows: At every round t for task m, it calculates the ucb value $B_{m,a,t}$ for each action $\mathbf{x}_{m,a} \in \mathcal{X}$ such that $B_{m,a,t} = \mathbf{x}_{m,a}^\top \widehat{\boldsymbol{\theta}}_{m,t-1} + \alpha \|\mathbf{x}_{m,a}\|_{\boldsymbol{\Sigma}_{m,t-1}^{-1}}$ where $\alpha > 0$ is a constant and $\widehat{\boldsymbol{\theta}}_{m,t}$ is the estimate of the model parameter $\boldsymbol{\theta}_{m,*}$ at round t. Here, $\boldsymbol{\Sigma}_{m,t-1} = \sum_{s=1}^{t-1} \mathbf{x}_{m,s}\mathbf{x}_{m,s}^\top + \lambda \mathbf{I}_d$ is the data covariance matrix or the arms already tried. Then it chooses $I_t = \arg\max_a B_{m,a,t}$. Note that LinUCB is a *strong* demonstrator that we give oracle access to the features of each action; other algorithms do not observe the features. Hence, in linear bandits, LinUCB provides an approximate upper bound on the performance of all algorithms.

(3) MLinGreedy: This is the multi-task linear regression bandit algorithm proposed by Yang et al. (2021). This algorithm assumes that there is a common low dimensional feature extractor $\mathbf{B} \in \mathbb{R}^{k\times d}$, $k \leq d$ shared between the tasks and the rewards per task m are linearly dependent on a hidden parameter $\boldsymbol{\theta}_{m,*}$. Under a diversity assumption (which may not be satisfied in real data) and $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_M]$ they assume $\boldsymbol{\Theta} = [\boldsymbol{\theta}_{1,*}, \ldots, \boldsymbol{\theta}_{M,*}] = \mathbf{BW}$. During evaluation MLinGreedy estimates the $\widehat{\mathbf{B}}$ and $\widehat{\mathbf{W}}$ from training data and fit $\widehat{\boldsymbol{\theta}}_m = \widehat{\mathbf{B}}\widehat{\mathbf{w}}_m$ per task and selects action greedily based on $I_{m,t} = \arg\max_a \mathbf{x}_{m,a}^\top \widehat{\boldsymbol{\theta}}_{m,*}$. Finally, note that MLinGreedy requires access to the action features to estimate $\widehat{\boldsymbol{\theta}}_m$ and select actions as opposed to DPT, AD, and PreDeToR.

A.3 Empirical Study: Comparison against K-armed bandits and DPT

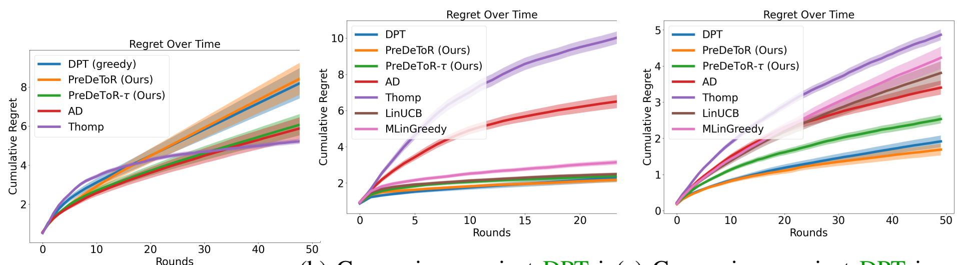
In this section, we discuss the performance of PreDeToR (-τ) when there is no latent structure in the data, that is the K-armed bandits. Then we compare the performance of PreDeToR (-τ) against DPT.

Baselines: In the K-armed bandits We implement the same baselines discussed in Section 4. The baselines are PreDeToR, PreDeToR-τ, DPT-greedy, AD, Thomp, and LinUCB. In the linear and non-linear setting, we compare against DPT instead of DPT-greedy.

Settings: In the K-armed bandit setting we consider $d = 6$, and the arms as canonical vectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_6$. For each task m, we choose the hidden parameter $\boldsymbol{\theta}_{m,*}$ similar to the linear bandit setting discussed in Section 5. Note that this results in a K-armed bandit setting. For the linear and non-linear setting comparison, we use the same setting as Section 5, and 4.

Outcomes: We first discuss the main outcomes from our experimental results in K-armed bandits and then in comparison against DPT in linear and non-linear settings.

> **Finding 7:** PreDeToR ($-\tau$) matches the performance of the demonstrator when there is no structure (K-armed bandits). PreDeToR ($-\tau$) performs close to DPT in the linear and non-linear setting showing the usefulness of learning the reward structure.



(a) K-armed Bandit (b) Comparison against DPT in linear setting (c) Comparison against DPT in non-linear setting

Figure 7: Experiment with k-armed bandits and DPT (original). The y-axis shows the cumulative regret.

Experimental Result: We observe these outcomes in Figure 7. In Figure 7a the demonstrator π^w is the Thomp algorithm. *Note that there is no structure across arms now, and sampling one arm gives no information about other arms in a task.* We observe that PreDeToR-τ performs similarly to the demonstrator Thomp, and also shows that incorporating exploration is a sound technique. Also, AD performs similarly to the demonstrator Thomp. Both DPT-greedy and PreDeToR fail to learn the latent structure across the tasks and therefore do not learn any exploration strategy.

In Figure 7b we show the linear bandit setting discussed in Appendix A.2. We observe that PreDeToR ($-\tau$) matches the performance of DPT, and LinUCB. Note that DPT has access to the optimal action per task, and LinUCB is the optimal oracle algorithm that leverages the structure information.

In Figure 7c we show the non-linear bandit setting discussed in Appendix A.2. Again we observe that PreDeToR ($-\tau$) matches the performance of DPT and has lower cumulative regret than AD and LinUCB which fails to perform well in this non-linear setting due to its algorithmic design.

A.4 Empirical Study: Bilinear Bandits

In this section, we discuss the performance of PreDeToR against the other baselines in the bilinear setting. Again note that the number of tasks $M_{\text{pre}} \gg A \geq n$. Through this experiment, we want to evaluate the performance of PreDeToR to exploit the underlying latent structure and reward correlation when the horizon is small, the number of tasks is large, and understand its performance in the bilinear bandit setting (Jun et al., 2019; Lu et al., 2021; Kang et al., 2022; Mukherjee et al., 2023). Note that this setting also goes beyond the linear feedback model (Abbasi-Yadkori et al., 2011; Lattimore & Szepesvári, 2020) and is related to matrix bandits (Yang & Wang, 2020).

Bilinear bandit setting: In the bilinear bandits the learner is provided with two sets of action sets, $\mathcal{X} \subseteq \mathbb{R}^{d_1}$ and $\mathcal{Z} \subseteq \mathbb{R}^{d_2}$ which are referred to as the left and right action sets. At every round t the learner chooses a pair of actions $\mathbf{x}_t \in \mathcal{X}$ and $\mathbf{z}_t \in \mathcal{Z}$ and observes a reward

$$r_t = \mathbf{x}_t^\top \mathbf{\Theta}_* \mathbf{z}_t + \eta_t$$

where $\mathbf{\Theta}_* \in \mathbb{R}^{d_1 \times d_2}$ is the unknown hidden matrix which is also low-rank. The η_t is a σ^2 sub-Gaussian noise. In the multi-task bilinear bandit setting we now have a set of M tasks where the reward for the m-th task at round t is given by

$$r_{m,t} = \mathbf{x}_{m,t}^\top \mathbf{\Theta}_{m,*} \mathbf{z}_{m,t} + \eta_{m,t}.$$

Here $\mathbf{\Theta}_{m,*} \in \mathbb{R}^{d_1 \times d_2}$ is the unknown hidden matrix for each task m, which is also low-rank. The $\eta_{m,t}$ is a σ^2 sub-Gaussian noise. Let κ be the rank of each of these matrices $\mathbf{\Theta}_{m,*}$.

A special case is the rank 1 structure where $\boldsymbol{\Theta}_{m,*} = \boldsymbol{\theta}_{m,*}\boldsymbol{\theta}_{m,*}^\top$ where $\boldsymbol{\Theta}_{m,*} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{\theta}_{m,*} \in \mathbb{R}^d$ for each task m. Let the left and right action sets be also same such that $\mathbf{x}_{m,t} \in \mathcal{X} \subseteq \mathbb{R}^d$. Observe then that the reward for the m-th task at round t is given by

$$r_{m,t} = \mathbf{x}_{m,t}^\top \boldsymbol{\Theta}_{m,*}\mathbf{x}_{m,t} + \eta_{m,t} = (\mathbf{x}_{m,t}^\top \boldsymbol{\theta}_{m,*})^2 + \eta_{m,t}.$$

This special case is studied in Chaudhuri et al. (2017).

Baselines: We again implement the same baselines discussed in Section 4. The baselines are PreDeToR, PreDeToR-τ, DPT-greedy, and Thomp. Note that we do not implement the LinUCB and MLinGreedy for the bilinear bandit setting. However, we now implement the LowOFUL (Jun et al., 2019) which is optimal in the bilinear bandit setting.

LowOFUL: The LowOFUL algorithm first estimates the unknown parameter $\boldsymbol{\Theta}_{m,*}$ for each task m using E-optimal design (Pukelsheim, 2006; Fedorov, 2013; Jun et al., 2019) for n_1 rounds. Let $\widehat{\boldsymbol{\Theta}}_{m,n_1}$ be the estimate of $\boldsymbol{\Theta}_{m,*}$ at the end of n_1 rounds. Let the SVD of $\widehat{\boldsymbol{\Theta}}_{m,n_1}$ be given by $\mathrm{SVD}(\widehat{\boldsymbol{\Theta}}_{m,n_1}) = \widehat{\mathbf{U}}_{m,n_1}\widehat{\mathbf{S}}_{m,n_1}\widehat{\mathbf{V}}_{m,n_1}$. Then LowOFUL rotates the actions as follows:

$$\mathcal{X}'_m = \left\{\left[\widehat{\mathbf{U}}_{m,n_1}\widehat{\mathbf{U}}_{m,n_1}^\perp\right]^\top \mathbf{x}_m : \mathbf{x}_m \in \mathcal{X}\right\} \textbf{ and } \mathcal{Z}' = \left\{\left[\widehat{\mathbf{V}}_{m,n_1}\widehat{\mathbf{V}}_{m,n_1}^\perp\right]^\top \mathbf{z}_m : \mathbf{z}_m \in \mathcal{Z}\right\}.$$

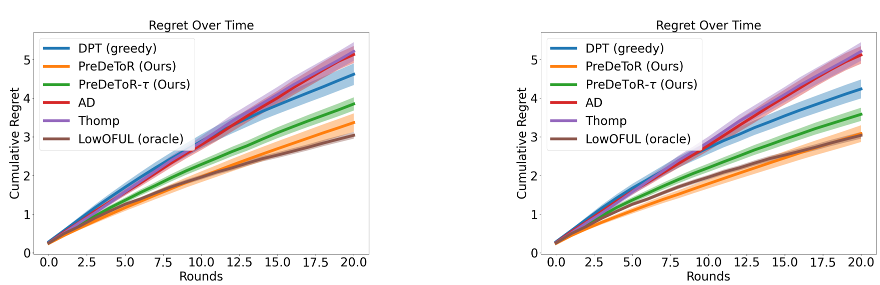
Then defines a vectorized action set for each task m so that the last $(d_1 - \kappa)\cdot(d_2 - \kappa)$ components are from the complementary subspaces:

$$\begin{aligned}\widetilde{\mathcal{A}}_m = \Big\{ \big[&\mathrm{vec}\left(\mathbf{x}_{m,1:\kappa}\mathbf{z}_{m,1:\kappa}^\top\right); \mathrm{vec}\left(\mathbf{x}_{m,\kappa+1:d_1}\mathbf{z}_{m,1:\kappa}^\top\right); \mathrm{vec}\left(\mathbf{x}_{m,1:\kappa}\mathbf{z}_{m,\kappa+1:d_2}^\top\right);\\ &\mathrm{vec}\left(\mathbf{x}_{m,\kappa+1:d_1}\mathbf{z}_{m,\kappa+1:d_2}^\top\right)\big] \in \mathbb{R}^{d_1 d_2} : \mathbf{x}_m \in \mathcal{X}'_m, \mathbf{z}_m \in \mathcal{Z}'_m \Big\}.\end{aligned}$$

Finally for $n_2 = n - n_1$ rounds, LowOFUL invokes the specialized OFUL algorithm (Abbasi-Yadkori et al., 2011) for the rotated action set $\widetilde{\mathcal{A}}_m$ with the low dimension $k = (d_1 + d_2)\,\kappa - \kappa^2$. Note that the LowOFUL runs the per-task low dimensional OFUL algorithm rather than learning the underlying structure across the tasks (Mukherjee et al., 2023).

Outcomes: We first discuss the main outcomes of our experimental results for increasing the horizon:

> **Finding 8:** PreDeToR (-τ) outperforms DPT-greedy, AD, and matches the performance of LowOFUL in bilinear bandit setting.



(a) Rank 1 $\boldsymbol{\Theta}_{m,*}$ (b) Rank 2 $\boldsymbol{\Theta}_{m,*}$

Figure 8: Experiment with bilinear bandits. The y-axis shows the cumulative regret.

Experimental Result: We observe these outcomes in Figure 8. In Figure 8a we experiment with rank 1 hidden parameter $\boldsymbol{\Theta}_{m,*}$ and set horizon $n = 20$, $M_{\text{pre}} = 200000$, $M_{\text{test}} = 200$, $A = 30$, and $d = 5$. In Figure 8b we experiment with rank 2 hidden parameter $\boldsymbol{\Theta}_{m,*}$ and set horizon $n = 20$, $M_{\text{pre}} = 250000$, $M_{\text{test}} = 200$, $A = 25$, and $d = 5$. Again, the demonstrator π^w is the Thomp algorithm. We observe that PreDeToR has lower cumulative regret than DPT-greedy, AD and Thomp. Note that for any task m for the horizon 20 the Thomp will be able to sample all the actions at most once. Note that for this small horizon setting the DPT-greedy does not have a good estimation of $\widehat{a}_{m,*}$ which results in a poor prediction of optimal action $\widehat{a}_{m,t,*}$. In contrast PreDeToR learns the correlation of rewards across tasks and can perform well. Observe from Figure 8a, and 8b that PreDeToR has lower regret than Thomp and matches LowOFUL. Also, in this low-data regime it is not enough for LowOFUL to learn the underlying $\mathbf{\Theta}_{m,*}$ with high precision. Hence, PreDeToR also has slightly lower regret than LowOFUL. Note that the main objective of AD is to match the performance of its demonstrator. Most importantly it shows that PreDeToR can exploit the underlying latent structure and reward correlation better than DPT-greedy, and AD.

A.5 Empirical Study: Latent Bandits

In this section, we discuss the performance of PreDeToR (-τ) against the other baselines in the latent bandit setting and create a generalized bilinear bandit setting. Note that the number of tasks $M_{\text{pre}} \gg A \geq n$. Using this experiment, we want to evaluate the ability of PreDeToR (-τ) to exploit the underlying reward correlation when the horizon is small, the number of tasks is large, and understand its performance in the latent bandit setting (Hong et al., 2020; Maillard & Mannor, 2014; Pal et al., 2023; Kveton et al., 2017). We create a latent bandit setting which generalizes the bilinear bandit setting (Jun et al., 2019; Lu et al., 2021; Kang et al., 2022; Mukherjee et al., 2023). Again note that this setting also goes beyond the linear feedback model (Abbasi-Yadkori et al., 2011; Lattimore & Szepesvári, 2020) and is related to matrix bandits (Yang & Wang, 2020).

Latent bandit setting: In this special multi-task latent bandits the learner is again provided with two sets of action sets, $\mathcal{X} \subseteq \mathbb{R}^{d_1}$ and $\mathcal{Z} \subseteq \mathbb{R}^{d_2}$ which are referred to as the left and right action sets. The reward for the m-th task at round t is given by

$$r_{m,t} = \mathbf{x}_{m,t}^\top \underbrace{(\mathbf{\Theta}_{m,*} + \mathbf{U}\mathbf{V}^\top)}_{\mathbf{Z}_{m*}} \mathbf{z}_{m,t} + \eta_{m,t}.$$

Here $\mathbf{\Theta}_{m,*} \in \mathbb{R}^{d_1 \times d_2}$ is the unknown hidden matrix for each task m, which is also low-rank. Additionally, all the tasks share a *common latent parameter matrix* $\mathbf{U}\mathbf{V}^\top \in \mathbb{R}^{d_1 \times d_2}$ which is also low rank. Hence the learner needs to learn the latent parameter across the tasks hence the name latent bandits. Finally, the $\eta_{m,t}$ is a σ^2 sub-Gaussian noise. Let κ be the rank of each of these matrices $\mathbf{\Theta}_{m,*}$ and $\mathbf{U}\mathbf{V}^\top$. Again special case is the rank 1 structure where the reward for the m-th task at round t is given by

$$r_{m,t} = \mathbf{x}_{m,t}^\top \underbrace{(\boldsymbol{\theta}_{m,*}\boldsymbol{\theta}_{m,*}^\top + \mathbf{u}\mathbf{v}^\top)}_{\mathbf{Z}_{m,*}} \mathbf{x}_{m,t} + \eta_{m,t}.$$

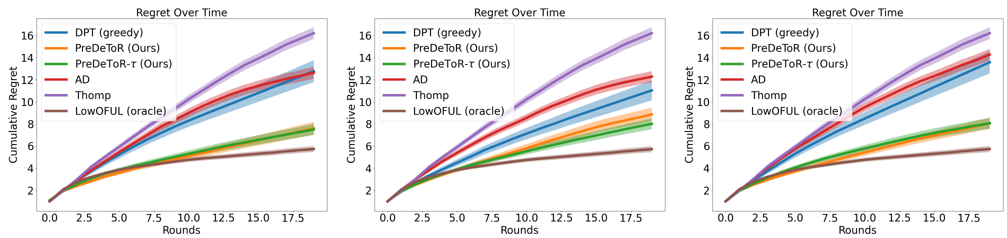
where $\boldsymbol{\theta}_{m,*} \in \mathbb{R}^d$ for each task m and $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$. Note that the left and right action sets are the same such that $\mathbf{x}_{m,t} \in \mathcal{X} \subseteq \mathbb{R}^d$.

Baselines: We again implement the same baselines discussed in Section 4. The baselines are PreDeToR, PreDeToR-τ, DPT-greedy, AD, Thomp, and LowOFUL. However, we now implement a special LowOFUL (stated in Appendix A.4) which has knowledge of the shared latent parameters $\mathbf{U}$, and $\mathbf{V}$. We call this the LowOFUL (oracle) algorithm. Therefore LowOFUL (oracle) has knowledge of the problem parameters in the latent bandit setting and hence the name. Again note that we do not implement the LinUCB and MLinGreedy for the latent bandit setting.

Outcomes: We first discuss the main outcomes of our experimental results for increasing the horizon:

> **Finding 9:** PreDeToR (-τ) outperforms DPT-greedy, AD, and matches the performance of LowOFUL (oracle) in latent bandit setting.

Experimental Result: We observe these outcomes in Figure 9. In Figure 9a we experiment with rank 1 hidden parameter $\boldsymbol{\theta}_{m,*}\boldsymbol{\theta}_{m,*}^\top$ and latent parameters $\mathbf{u}\mathbf{v}^\top$ shared across the tasks and set horizon



(a) Rank 1 $\mathbf{Z}_{m,*}$ (b) Rank 2 $\mathbf{Z}_{m,*}$ (c) Rank 3 $\mathbf{Z}_{m,*}$

Figure 9: Experiment with latent bandits. The y-axis shows the cumulative regret.

$n = 20$, $M_{\text{pre}} = 200000$, $M_{\text{test}} = 200$, $A = 30$, and $d = 5$. In Figure 9b we experiment with rank 2 hidden parameter $\mathbf{\Theta}_{m,*}$, and latent parameters $\mathbf{U}\mathbf{V}^\top$ and set horizon $n = 20$, $M_{\text{pre}} = 250000$, $M_{\text{test}} = 200$, $A = 25$, and $d = 5$. In Figure 9c we experiment with rank 3 hidden parameter $\mathbf{\Theta}_{m,*}$, and latent parameters $\mathbf{U}\mathbf{V}^\top$ and set horizon $n = 20$, $M_{\text{pre}} = 300000$, $M_{\text{test}} = 200$, $A = 25$, and $d = 5$. Again, the demonstrator π^w is the Thomp algorithm. We observe that PreDeToR (-τ) has lower cumulative regret than DPT-greedy, AD and Thomp. Note that for any task m for the horizon 20 the Thomp will be able to sample all the actions at most once. Note that for this small horizon setting the DPT-greedy does not have a good estimation of $\widehat{a}_{m,*}$ which results in a poor prediction of optimal action $\widehat{a}_{m,t,*}$. In contrast PreDeToR (-τ) learns the correlation of rewards across tasks and is able to perform well. Observe from Figure 9a, 9b, and 9c that PreDeToR has lower regret than Thomp and has regret closer to LowOFUL (oracle)which has access to the problem-dependent parameters. Hence. LowOFUL (oracle) outperforms PreDeToR (-τ) in this setting. This shows that PreDeToR is able to exploit the underlying latent structure and reward correlation better than DPT-greedy, and AD.

A.6 Connection between PreDeToR and Linear Multivariate Gaussian Model

In this section, we try to understand the behavior of PreDeToR and its ability to exploit the reward correlation across tasks under a *linear multivariate Gaussian model*. In this model, the hidden task parameter, $\boldsymbol{\theta}_*$, is a random variable drawn from a multi-variate Gaussian distribution (Bishop, 2006) and the feedback follows a linear model. We study this setting since we can estimate the Linear Minimum Mean Square Estimator (LMMSE) in this setting (Carlin & Louis, 2008; Box & Tiao, 2011). This yields a posterior prediction for the mean of each action over all tasks on average, by leveraging the linear structure when $\boldsymbol{\theta}_*$ is drawn from a multi-variate Gaussian distribution. So we can compare the performance of PreDeToR against such an LMMSE and evaluate whether it is exploiting the underlying linear structure and the reward correlation across tasks. We summarize this as follows:

> **Finding 10:** PreDeToR learns the reward correlation covariance matrix from the in-context training data $\mathcal{H}_{\text{train}}$ and acts greedily on it.

Consider the linear feedback setting consisting of A actions and the hidden task parameter $\boldsymbol{\theta}_* \sim \mathcal{N}(0, \sigma^2_{\boldsymbol{\theta}} \mathbf{I}_d)$. The reward of the action $\mathbf{x}_t$ at round t is given by $r_t = \mathbf{x}_t^\top \boldsymbol{\theta}_* + \eta_t$, where η_t is σ^2 sub-Gaussian. Let π^w collect n rounds of pretraining in-context data and observe $\{I_t, r_t\}_{t=1}^n$. Let $N_n(a)$ denote the total number of times the action a is sampled for n rounds. Note that we drop the task index m in these notations as the random variable $\boldsymbol{\theta}_*$ corresponds to the task. Define the matrix $\mathbf{H}_n \in \mathbb{R}^{n \times A}$ where the t-th row represents the action I_t for $t \in [n]$. The t-th row of $\mathbf{H}_n$ is a one-hot vector with the I_t-th component being 1. We represent each action by one hot vector because we assume that this LMMSE does not have access to the feature vectors of the actions similar to the PreDeToR for fair comparison. Then define the reward vector $\mathbf{Y}_n \in \mathbb{R}^n$ where the t-th component is the reward r_t observed for the action I_t for $t \in [n]$ in the pretraining data. Define the diagonal matrix

$\mathbf{D}_A \in \mathbb{R}^{A\times A}$ estimated from pretraining data as follows

$$\mathbf{D}_A(i,i) = \begin{cases} \frac{\sigma^2}{N_n(a)}, \text{ if } N_n(a) > 0 \\ = 0, \text{ if } N_n(a) = 0 \end{cases} \tag{6}$$

where the reward noise being σ^2 sub-Gaussian is known. Finally define the estimated reward covariance matrix $\mathbf{S}_A \in \mathbb{R}^{A\times A}$ as $\mathbf{S}_A(a,a') = \widehat{\mu}_n(a)\widehat{\mu}_n(a')$, where $\widehat{\mu}_n(a)$ is the empirical mean of action a estimated from the pretraining data. This matrix captures the reward correlation between the pairs of actions $a, a' \in [A]$. Then the posterior average mean estimator $\widehat{\mu} \in \mathbb{R}^A$ over all tasks is given by the following lemma. The proof is given in Appendix B.1.

Lemma 1. *Let $\mathbf{H}_n$ be the action matrix, $\mathbf{Y}_n$ be the reward vector and $\mathbf{S}_A$ be the estimated reward covariance matrix. Then the posterior prediction of the average mean reward vector $\widehat{\mu}$ over all tasks is given by*

$$\widehat{\mu} = \sigma^2_{\boldsymbol{\theta}}\mathbf{S}_A\mathbf{H}_n^\top \left(\sigma^2_{\boldsymbol{\theta}}\mathbf{H}_n(\mathbf{S}_A + \mathbf{D}_A)\mathbf{H}_n^\top\right)^{-1}\mathbf{Y}_n. \tag{7}$$

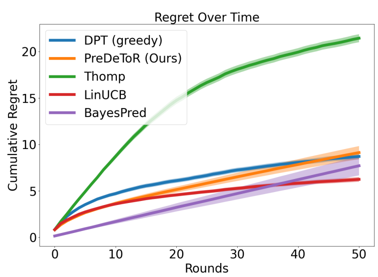


Figure 10: BayesPred Performance

The $\widehat{\mu}$ in (7) represents the posterior mean vector averaged on all tasks. So if some action $a \in [A]$ consistently yields high rewards in the pretraining data then $\widehat{\mu}(a)$ has high value. Since the test distribution is the same as pretraining, this action on average will yield a high reward during test time.

We hypothesize that the PreDeToR is learning the reward correlation covariance matrix from the training data $\mathcal{H}_{\text{train}}$ and acting greedily on it. To test this hypothesis, we consider the greedy BayesPred algorithm that first estimates $\mathbf{S}_A$ from the pretraining data. It then uses the LMMSE estimator in Lemma 1 to calculate the posterior mean vector $\widehat{\mu}$, and then selects $I_t = \arg\max_a \widehat{\mu}(a)$ at each round t. Note that BayesPred is a greedy algorithm that always selects the most rewarding action (exploitation) without any exploration of sub-optimal actions. Also the BayesPred is an LMMSE estimator that leverages the linear reward structure and estimates the reward covariance matrix, and therefore can be interpreted as a lower bound to the regret of PreDeToR. The hypothesis that BayesPred is a lower bound to PreDeToR is supported by Figure 10. In Figure 10 the reward covariance matrix for BayesPred is estimated from the $\mathcal{H}_{\text{train}}$ by first running the Thomp (π^w). Observe that the BayesPred has a lower cumulative regret than PreDeToR and almost matches the regret of PreDeToR towards the end of the horizon. Also note that LinUCB has lower cumulative regret towards the end of horizon as it leverages the linear structure and the feature of the actions in selecting the next action.

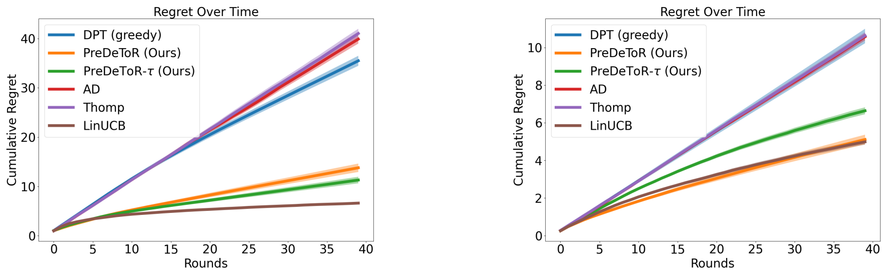
A.7 Empirical Study: Increasing number of Actions

In this section, we discuss the performance of PreDeToR when the number of actions is very high so that the weak demonstrator π^w does not have sufficient samples for each action. However, the number of tasks $M_{\text{pre}} \gg A > n$.

Baselines: We again implement the same baselines discussed in Section 4. The baselines are PreDeToR, PreDeToR-τ, DPT-greedy, AD, Thomp, and LinUCB.

Outcomes: We first discuss the main outcomes from our experimental results of introducing more actions than the horizon (or more dimensions than actions) during data collection and evaluation:

> **Finding 11:** PreDeToR (-τ) outperforms DPT-greedy, and AD, even when $A > n$ but $M_{\text{pre}} \gg A$.



(a) Linear Bandit

(b) Non-linear Bandit

Figure 11: Testing the limit experiments. The horizontal axis is the number of rounds. Confidence bars show one standard error.

Experimental Result: We observe these outcomes in Figure 11. In Figure 11a we show the linear bandit setting for $M_{\text{pre}} = 250000$, $M_{\text{test}} = 200$, $A = 100$, $n = 50$ and $d = 5$. Again, the demonstrator π^w is the Thomp algorithm. We observe that PreDeToR (-τ) has lower cumulative regret than DPT-greedy and AD. Note that for any task m the Thomp will not be able to sample all the actions even once. The weak performance of DPT-greedy can be attributed to both short horizons and the inability to estimate the optimal action for such a short horizon $n < A$. The AD performs similar to the demonstrator Thomp because of its training. Observe that PreDeToR (-τ) has similar regret to LinUCB and lower regret than Thomp which also shows that PreDeToR is exploiting the latent linear structure of the underlying tasks. In Figure 11b we show the non-linear bandit setting for horizon $n = 40$, $M_{\text{pre}} = 200000$, $A = 60$, $d = 2$, and $|\mathcal{A}^{\text{inv}}| = 5$. The demonstrator π^w is the Thomp algorithm. Again we observe that PreDeToR (-τ) has lower cumulative regret than DPT-greedy, AD and LinUCB which fails to perform well in this non-linear setting due to its algorithmic design.

A.8 Empirical Study: Increasing Horizon

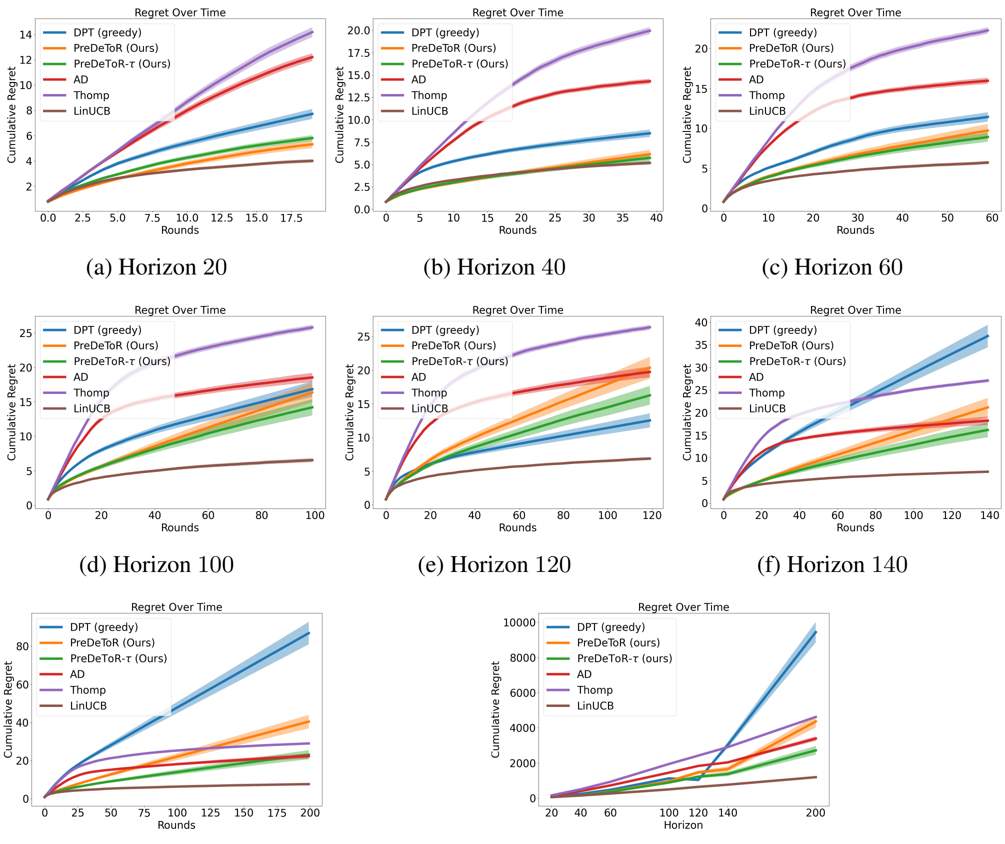
In this section, we discuss the performance of PreDeToR with respect to an increasing horizon for each task $m \in [M]$. However, note that the number of tasks $M_{\text{pre}} \geq n$. Note that Lee et al. (2023) studied linear bandit setting for $n = 200$. We study the setting up to a similar horizon scale.

Baselines: We again implement the same baselines discussed in Section 4. The baselines are PreDeToR, PreDeToR-τ, DPT-greedy, AD, Thomp, and LinUCB.

Outcomes: We first discuss the main outcomes of our experimental results for increasing the horizon:

> **Finding 12:** PreDeToR (-τ) outperforms DPT-greedy, and AD with increasing horizon.

Experimental Result: We observe these outcomes in Figure 12. In Figure 12 we show the linear bandit setting for $M_{\text{pre}} = 150000$, $M_{\text{test}} = 200$, $A = 20$, $n = \{20, 40, 60, 100, 120, 140, 200\}$ and $d = 5$. Again, the demonstrator π^w is the Thomp algorithm. We observe that PreDeToR (-τ) has lower cumulative regret than DPT-greedy, and AD. Note that for any task m for the horizon 20 the Thomp will be able to sample all the actions at most once. Observe from Figure 12a, 12b, 12c, Figure 12d, 12e, 12f and 12g that PreDeToR (-τ) is closer to LinUCB and outperforms Thomp which also shows that PreDeToR (-τ) is learning the latent linear structure of the underlying tasks. In Figure 12h we plot the regret of all the baselines with respect to the increasing horizon. Again we see that PreDeToR (-τ) is closer to LinUCB and outperforms DPT-greedy, AD and Thomp. This shows that PreDeToR (-τ) is able to exploit the latent structure and reward correlation across the tasks for varying horizon length.



(a) Horizon 20 (b) Horizon 40 (c) Horizon 60

(d) Horizon 100 (e) Horizon 120 (f) Horizon 140

(g) Horizon 200 (h) Increasing Horizon

Figure 12: Experiment with increasing horizon. The y-axis shows the cumulative regret.

A.9 Empirical Study: Increasing Dimension

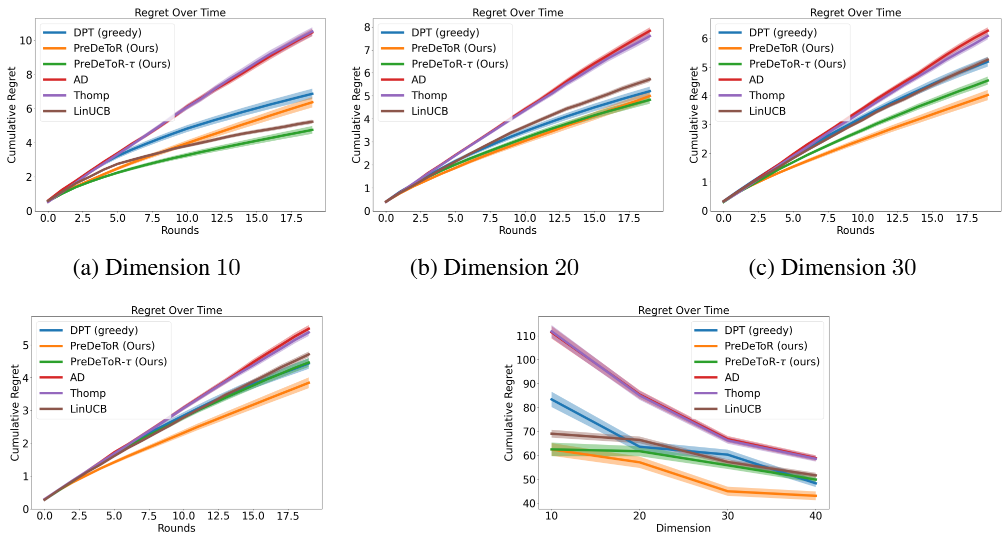
In this section, we discuss the performance of PreDeToR with respect to an increasing dimension for each task $m \in [M]$. Again note that the number of tasks $M_{\text{pre}} \gg A \geq n$. Through this experiment, we want to evaluate the performance of PreDeToR and see how it exploits the underlying reward correlation when the horizon is small as well as for increasing dimensions.

Baselines: We again implement the same baselines discussed in Section 4. The baselines are PreDeToR, PreDeToR-τ DPT-greedy, AD, Thomp, and LinUCB.

Outcomes: We first discuss the main outcomes of our experimental results for increasing the horizon:

> **Finding 13:** PreDeToR (-τ) outperforms DPT-greedy, AD with increasing dimension and has lower regret than LinUCB for larger dimension.

Experimental Result: We observe these outcomes in Figure 12. In Figure 12 we show the linear bandit setting for horizon $n = 20$, $M_{\text{pre}} = 160000$, $M_{\text{test}} = 200$, $A = 20$, and $d = \{10, 20, 30, 40\}$. Again, the demonstrator π^w is the Thomp algorithm. We observe that PreDeToR (-τ) has lower cumulative regret than DPT-greedy, AD. Note that for any task m for the horizon 20 the Thomp will be able to sample all the actions at most once. Observe from Figure 13a, 13b, 13c, and 13d that PreDeToR (-τ) is closer to LinUCB and has lower regret than Thomp which also shows that PreDeToR (-τ) is exploiting the latent linear structure of the underlying tasks. In Figure 13e we plot the regret of all the baselines with respect to the increasing dimension. Again we see that PreDeToR (-τ) has lower regret than DPT-greedy, AD and Thomp. Observe that with increasing dimension



(a) Dimension 10 (b) Dimension 20 (c) Dimension 30

(d) Dimension 40 (e) Increasing Dimension

Figure 13: Experiment with increasing dimension. The y-axis shows the cumulative regret.

PreDeToR is able to outperform LinUCB. This shows that the PreDeToR (-τ) is able to exploit reward correlation across tasks for varying dimensions.

A.10 Empirical Study: Increasing Attention Heads

In this section, we discuss the performance of PreDeToR with respect to an increasing attention heads for the transformer model for the non-linear feedback model. Again note that the number of tasks $M_{\text{pre}} \gg A \geq n$. Through this experiment, we want to evaluate the performance of PreDeToR to exploit the underlying reward correlation when the horizon is small and understand the representative power of the transformer by increasing the attention heads. Note that we choose the non-linear feedback model and low data regime to leverage the representative power of the transformer.

Baselines: We again implement the same baselines discussed in Section 4. The baselines are PreDeToR, PreDeToR-τ, DPT-greedy, AD, Thomp, and LinUCB.

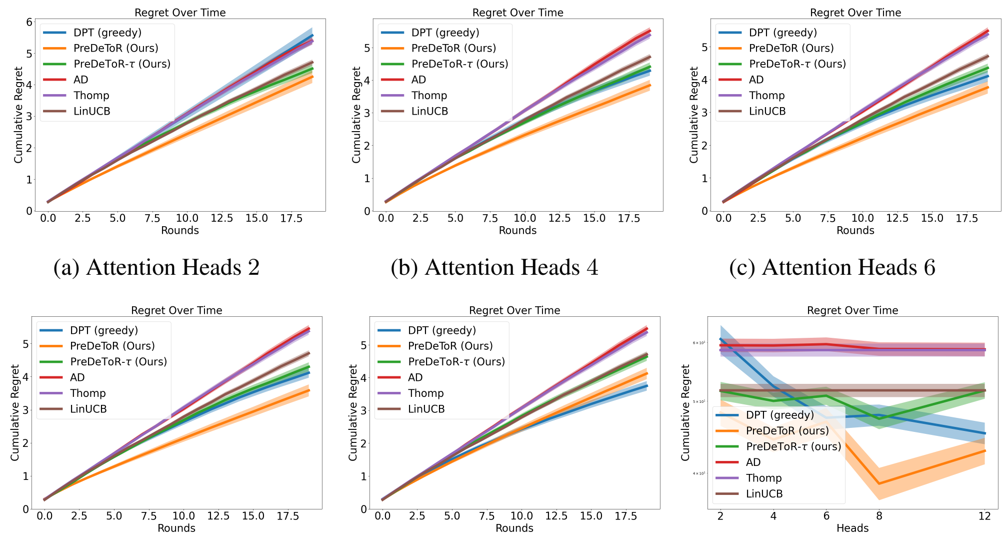
Outcomes: We first discuss the main outcomes of our experimental results for increasing the horizon:

> **Finding 14:** PreDeToR (-τ) outperforms DPT-greedy, and AD with increasing attention heads.

Experimental Result: We observe these outcomes in Figure 14. In Figure 14 we show the non-linear bandit setting for horizon $n = 20$, $M_{\text{pre}} = 160000$, $M_{\text{test}} = 200$, $A = 20$, heads $= \{2, 4, 6, 8\}$ and $d = 5$. Again, the demonstrator π^w is the Thomp algorithm. We observe that PreDeToR (-τ) has lower cumulative regret than DPT-greedy, AD. Note that for any task m for the horizon 20 the Thomp will be able to sample all the actions atmost once. Observe from Figure 14a, 14b, 14c, and 14d that PreDeToR (-τ) has lower regret than AD, Thomp and LinUCB which also shows that PreDeToR (-τ) is exploiting the latent linear structure of the underlying tasks for the non-linear setting. In Figure 14f we plot the regret of all the baselines with respect to the increasing attention heads. Again we see that PreDeToR (-τ) regret decreases as we increase the attention heads.

A.11 Empirical Study: Increasing Number of Tasks

In this section, we discuss the performance of PreDeToR with respect to the increasing number of tasks for the linear bandit setting. Again note that the number of tasks $M_{\text{pre}} \gg A \geq n$. Through



(a) Attention Heads 2 (b) Attention Heads 4 (c) Attention Heads 6

(d) Attention Heads 8 (e) Attention Heads 12 (f) Increasing Attention Heads

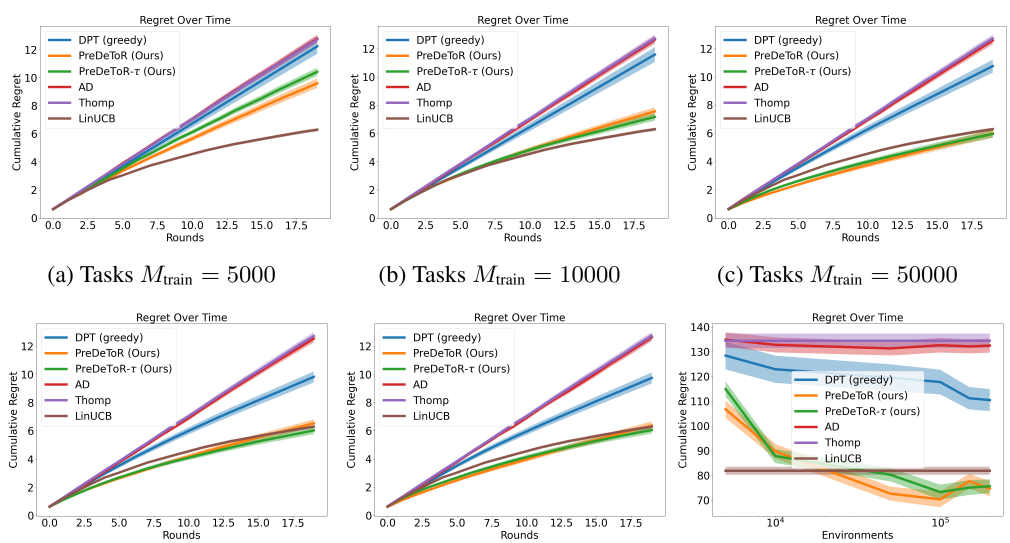
Figure 14: Experiment with increasing attention heads. The y-axis shows the cumulative regret.

this experiment, we want to evaluate the performance of PreDeToR to exploit the underlying reward correlation when the horizon is small and the number of tasks is changing. Finally, recall that when the horizon is small the weak demonstrator π^w does not have sufficient samples for each action. This leads to a poor approximation of the greedy action.

Baselines: We again implement the same baselines discussed in Section 4. The baselines are PreDeToR, PreDeToR-τ, DPT-greedy, AD, Thomp, and LinUCB.

Outcomes: We first discuss the main outcomes of our experimental results for increasing the horizon:

> **Finding 15:** PreDeToR (-τ) fails to exploit the underlying latent structure and reward correlation from in-context data when the number of tasks is small.



(a) Tasks $M_{\text{train}} = 5000$ (b) Tasks $M_{\text{train}} = 10000$ (c) Tasks $M_{\text{train}} = 50000$

(d) Tasks $M_{\text{train}} = 100000$ (e) Tasks $M_{\text{train}} = 150000$ (f) Increasing tasks

Figure 15: Experiment with an increasing number of tasks. The y-axis shows the cumulative regret.

Experimental Result: We observe these outcomes in Figure 15. In Figure 15 we show the linear bandit setting for horizon $n = 20$, $M_{\text{pre}} \in \{5000, 10000, 50000, 100000, 150000\}$, $M_{\text{test}} = 200$, $A = 20$, and $d = 40$. Again, the demonstrator π^w is the Thomp algorithm. We observe that PreDeToR (-τ), AD and DPT-greedy suffer more regret than the LinUCB when the number of tasks is small ($M_{\text{train}} \in \{5000, 10000\}$ in Figure 15a, and 15b. However in Figure 15c, 15d, 15e, and 15f we show that PreDeToR has lower regret than Thomp and matches LinUCB. This shows that PreDeToR (-τ) is exploiting the latent linear structure of the underlying tasks for the non-linear setting. Moreover, observe that as M_{train} increases the PreDeToR has lower cumulative regret than DPT-greedy, AD. Note that for any task m for the horizon 20 the Thomp will be able to sample all the actions at most once. Therefore DPT-greedy does not perform as well as PreDeToR. Finally, note that the result shows that PreDeToR (-τ) is able to exploit the reward correlation across the tasks better as the number of tasks increases.

A.12 Exploration of PreDeToR(-τ) in New Arms Setting

In this section, we discuss the exploration of PreDeToR (-τ) in the linear and non-linear new arms bandit setting discussed in Section 6. Recall that we consider the linear bandit setting of horizon $n = 50$, $M_{\text{pre}} = 200000$, $M_{\text{test}} = 200$, $A = 20$, and $d = 5$. Here during data collection and during collecting the test data, we randomly select one new action from $\mathbb{R}^d$ for each task m. So the number of invariant actions is $|\mathcal{A}^{\text{inv}}| = 19$.

Outcomes: We first discuss the main outcomes of our analysis of exploration in the low-data regime:

> **Finding 16:** The PreDeToR (-τ) is robust to changes when the number of in-variant actions is large. PreDeToR (-τ) performance drops as shared structure breaks down.

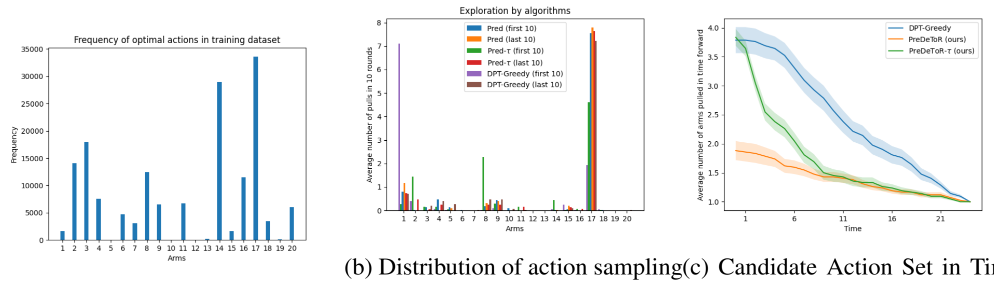
We first show in Figure 16a the training distribution of the optimal actions. For each bar, the frequency indicates the number of tasks where the action (shown in the x-axis) is the optimal action.

Then in Figure 16b we show how the sampling distribution of DPT-greedy, PreDeToR and PreDeToR-τ change in the first 10 and last 10 rounds for all the tasks where action 17 is optimal. We plot this graph the same way as discussed in Section 5. From the figure Figure 16b we see that PreDeToR(-τ) consistently pulls the action 17 more than DPT-greedy. It also explores other optimal actions like $\{1, 2, 3, 8, 9, 15\}$ but discards them quickly in favor of the optimal action 17 in these tasks.

Finally, we plot the feasible action set considered by DPT-greedy, PreDeToR, and PreDeToR-τ in Figure 16c. To plot this graph again we consider the test tasks where the optimal action is 17. Then we count the number of distinct actions that are taken from round t up until horizon n. Finally we average this over all the considered tasks where the optimal action is 17. We call this the candidate action set considered by the algorithm. From the Figure 16c we see that PreDeToR-τ explores more than PreDeToR in this setting.

We also show how the prediction error of the optimal action by PreDeToR compared to LinUCB in this 1 new arm linear bandit setting. In Figure 17a we first show how the 20 actions are distributed in the $M_{\text{test}} = 200$ test tasks. In Figure 17a for each bar, the frequency indicates the number of tasks where the action (shown in the x-axis) is the optimal action. Then in Figure 17b we show the prediction error of PreDeToR (-τ) for each task $m \in [M_{\text{test}}]$. The prediction error is calculated the same way as stated in Section 6 From the Figure 17b we see that for most actions the prediction error of PreDeToR (-τ) is closer to LinUCB showing that the introduction of 1 new action does not alter the prediction error much. Note that LinUCB estimates the empirical mean directly from the test task, whereas PreDeToR has a strong prior based on the training data. Therefore we see that PreDeToR is able to estimate the reward of the optimal action quite well from the training dataset $\mathcal{D}_{\text{pre}}$.

We now consider the setting where the number of invariant actions is $|\mathcal{A}^{\text{inv}}| = 15$. We again show in Figure 18a the training distribution of the optimal actions. For each bar, the frequency indicates the number of tasks where the action (shown in the x-axis) is the optimal action. Then in Figure 18b we

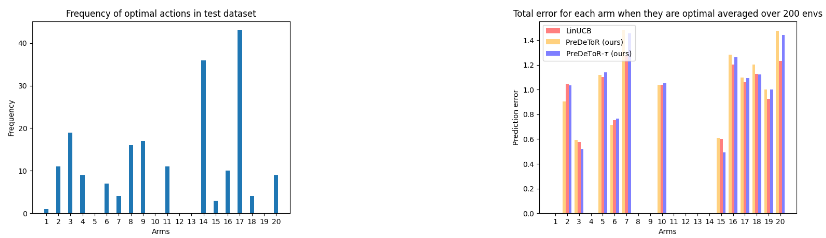


(a) Train Optimal Action Distribution

(b) Distribution of action sampling in all tasks where action 17 is optimal

(c) Candidate Action Set in Time averaged over all tasks where action 17 is optimal

Figure 16: Exploration Analysis of PreDeToR(-τ) in linear 1 new arm setting



(a) Test action distribution

(b) Test Prediction Error

Figure 17: Prediction error of PreDeToR(-τ) in linear 1 new arm setting

show how the sampling distribution of DPT-greedy, PreDeToR and PreDeToR-τ change in the first 10 and last 10 rounds for all the tasks where action 17 is optimal. We plot this graph the same way as discussed in Section 5. From the figure Figure 18b we see that none of the algorithms PreDeToR, PreDeToR-τ, DPT-greedy consistently pulls the action 17 more than other actions. This shows that the common underlying actions across the tasks matter for learning the epxloration.

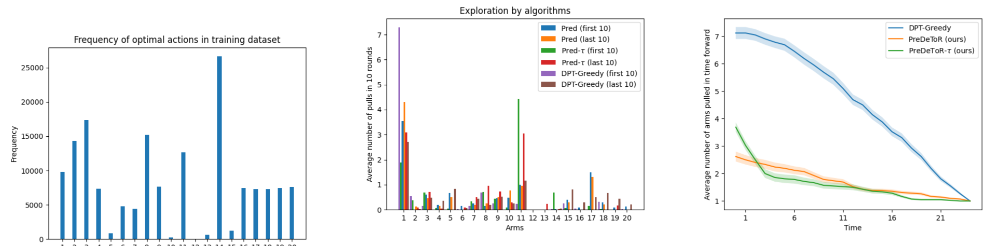
Finally, we plot the feasible action set considered by DPT-greedy, PreDeToR, and PreDeToR-τ in Figure 18c. To plot this graph again we consider the test tasks where the optimal action is 17. We build the candidate set the same way as before. From the Figure 18c we see that none of the three algorithms DPT-greedy, PreDeToR, PreDeToR-τ, is able to sample the optimal action 17 sufficiently high number of times.

We also show how the prediction error of the optimal action by PreDeToR compared to LinUCB in this 1 new arm linear bandit setting. In Figure 19a we first show how the 20 actions are distributed in the $M_{\text{test}} = 200$ test tasks. In Figure 19a for each bar, the frequency indicates the number of tasks where the action (shown in the x-axis) is the optimal action. Then in Figure 19b we show the prediction error of PreDeToR (-τ) for each task $m \in [M_{\text{test}}]$. The prediction error is calculated the same way as stated in Section 6. From the Figure 19b we see that for most actions the prediction error is higher than LinUCB showing that the introduction of 5 new actions (and thereby decreasing the invariant action set) significantly alters the prediction error.

A.13 Empirical Validation of Theoretical Result

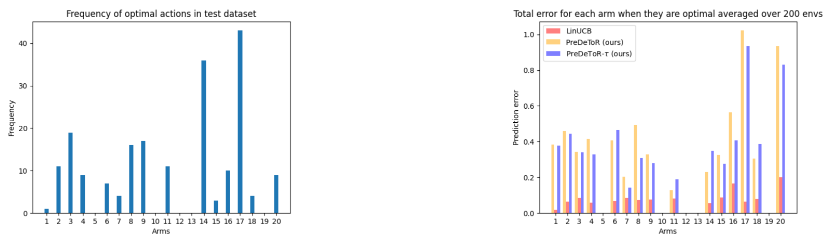
In this section, we empirically validate the theoretical result proved in Section 8. We again consider the linear bandit setting discussed in Section 4. Recall that the linear bandit setting consist of horizon $n = 25$, $M_{\text{pre}} = \{100000, 200000\}$, $M_{\text{test}} = 200$, $A = 10$, and $d = 2$. The demonstrator π^w is the Thomp algorithm and we observe that PreDeToR (-τ) has lower cumulative regret than DPT-greedy, AD and matches the performance of LinUCB.

Baseline (LinUCB-τ): We define soft LinUCB (LinUCB-τ) as follows: At every round t for task m, it calculates the ucb value $B_{m,a,t}$ for each action $\mathbf{x}_{m,a} \in \mathcal{X}$ such that $B_{m,a,t} = \mathbf{x}_{m,a}^\top \widehat{\boldsymbol{\theta}}_{m,t-1} +$



(a) Train Optimal Action Distribution (b) Distribution of action sampling in all tasks where action 17 is optimal (c) Candidate Action Set in Time averaged all tasks where action 17 is optimal

Figure 18: Exploration Analysis of PreDeToR(-τ) in linear 5 new arm setting



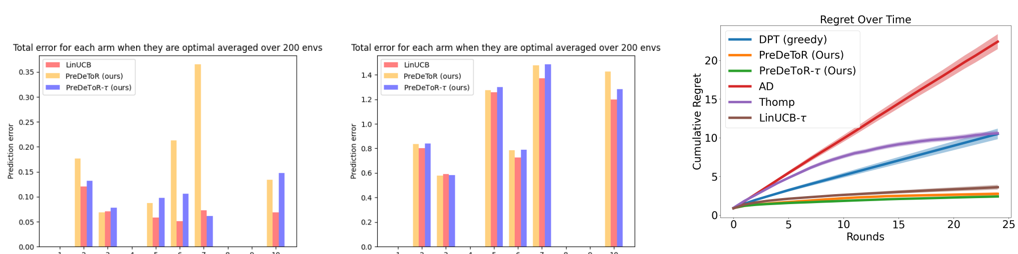
(a) Test action distribution (b) Test Prediction Error

Figure 19: Prediction error of PreDeToR(-τ) in linear 1 new arm setting

$\alpha \|\mathbf{x}_{m,a}\|_{\mathbf{\Sigma}^{-1}_{m,t-1}}$ where $\alpha > 0$ is a constant and $\widehat{\boldsymbol{\theta}}_{m,t}$ is the estimate of the model parameter $\boldsymbol{\theta}_{m,*}$ at round t. Here, $\mathbf{\Sigma}_{m,t-1} = \sum_{s=1}^{t-1} \mathbf{x}_{m,s}\mathbf{x}_{m,s}^\top + \lambda \mathbf{I}_d$ is the data covariance matrix or the arms already tried. Then it chooses $I_t \sim \text{softmax}_a^\tau(B_{m,a,t})$, where $\text{softmax}_a^\tau(\cdot) \in \triangle^A$ denotes a softmax distribution over the actions and τ is a temperature parameter (See Section 4 for definition of $\text{softmax}_a^\tau(\cdot)$).

Outcomes: We first discuss the main outcomes of our experimental results:

> **Finding 17:** PreDeToR (-τ) excels in predicting the rewards for test tasks when the number of training (source) tasks is large.



(a) Prediction Error for 10^5 tasks (b) Prediction Error for 2×10^5 tasks (c) Cumulative Regret of PreDeToR(-τ) compared against LinUCB-τ

Figure 20: Empirical validation of theoretical analysis

Experimental Result: These findings are reported in Figure 20. In Figure 20a we show the prediction error of PreDeToR (-τ) for each task $m \in [M_{\text{test}}]$. The prediction error is calculated as $(\widehat{\mu}_{m,n,*}(a) - \mu_{m,*}(a))^2$ where $\widehat{\mu}_{m,n,*}(a) = \max_a \widehat{\boldsymbol{\theta}}_{m,n}^\top \mathbf{x}_m(a)$ is the empirical mean at the end of round n, and $\mu_{*,m}(a) = \max_a \boldsymbol{\theta}_{m,*}^\top \mathbf{x}_m(a)$ is the true mean of the optimal action in task m. Then we average the prediction error for the action $a \in [A]$ by the number of times the action a is the optimal action in some task m. We see that when the source tasks are 100000 the reward prediction falls short of LinUCB prediction for all actions except action 2.

In Figure 20b we again show the prediction error of PreDeToR (-τ) for each task $m \in [M_{\text{test}}]$ when the source tasks are 200000. Note that in both these settings, we kept the horizon $n = 25$, and the same set of actions. We now observe that the reward prediction almost matches LinUCB prediction in almost all the optimal actions.

In Figure 20c we compare PreDeToR (-τ) against LinUCB-τ and show that they almost match in the linear bandit setting discussed in Section 4 when the source tasks are 100000.

A.14 Empirical Study: Offline Performance

In this section, we discuss the offline performance of PreDeToR when the number of tasks $M_{\text{pre}} \gg A \geq n$.

We first discuss how PreDeToR (-τ) is modified for the offline setting. In the offline setting, the PreDeToR first samples a task $m \sim \mathcal{T}_{\text{test}}$, then the test dataset $\mathcal{H}_m \sim \mathcal{D}_{\text{test}}(\cdot|m)$. Then PreDeToR and PreDeToR-τ act similarly to the online setting, but based on the entire offline dataset $\mathcal{H}_m$. The full pseudocode of PreDeToR is in Algorithm 2.

Algorithm 2 Pre-trained **D**ecision **T**ransf**o**rmer with **R**eward Estimation (PreDeToR)

Collecting Pretraining Dataset
Initialize empty pretraining dataset $\mathcal{H}_{\text{train}}$
for i in $[M_{\text{pre}}]$ **do**
  Sample task $m \sim \mathcal{T}_{\text{pre}}$, in-context dataset $\mathcal{H}_m \sim \mathcal{D}_{\text{pre}}(\cdot|m)$ and add this to $\mathcal{H}_{\text{train}}$.
end for
Pretraining model on dataset
Initialize model $\text{TF}_{\mathbf{\Theta}}$ with parameters $\mathbf{\Theta}$
while not converged **do**
  Sample $\mathcal{H}_m$ from $\mathcal{H}_{\text{train}}$ and predict $\widehat{r}_{m,t}$ for action $(I_{m,t})$ for all $t \in [n]$
  Compute loss in (3) with respect to $r_{m,t}$ and backpropagate to update model parameter $\mathbf{\Theta}$.
end while
Offline test-time deployment
Sample unknown task $m \sim \mathcal{T}_{\text{test}}$, sample dataset $\mathcal{H}_m \sim \mathcal{D}_{\text{test}}(\cdot|m)$
Use $\text{TF}_{\mathbf{\Theta}}$ on m at round t to choose

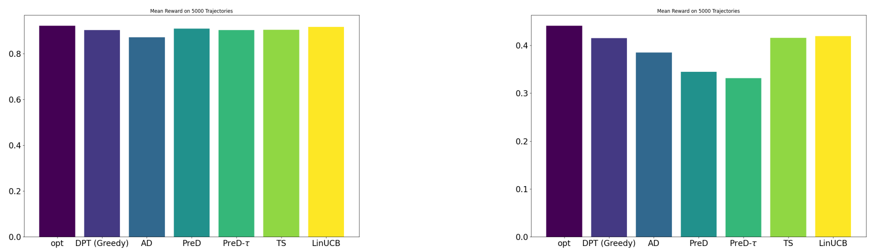
$$I_t \begin{cases} = \arg\max_{a \in \mathcal{A}} \text{TF}_{\mathbf{\Theta}} \left(\widehat{r}_{m,t}(a) \mid \mathcal{H}_m\right), & \textbf{PreDeToR} \\ \sim \text{softmax}_a^{\tau} \text{TF}_{\mathbf{\Theta}} \left(\widehat{r}_{m,t}(a) \mid \mathcal{H}_m\right), & \textbf{PreDeToR-}\tau \end{cases}$$

Recall that $\mathcal{D}_{\text{test}}$ denote a distribution over all possible interactions that can be generated by π^w during test time. For offline testing, first, a test task $m \sim \mathcal{T}_{\text{test}}$, and then an in-context test dataset $\mathcal{H}_m$ is collected such that $\mathcal{H}_m \sim \mathcal{D}_{\text{test}}(\cdot|m)$. Observe from Algorithm 2 that in the offline setting, PreDeToR first samples a task $m \sim \mathcal{T}_{\text{test}}$, and then a test dataset $\mathcal{H}_m \sim \mathcal{D}_{\text{test}}(\cdot|m)$ and acts greedily. Crucially in the offline setting the PreDeToR does not add the observed reward r_t at round t to the dataset. Through this experiment, we want to evaluate the performance of PreDeToR to learn the underlying latent structure and reward correlation when the horizon is small. Finally, recall that when the horizon is small the weak demonstrator π^w does not have sufficient samples for each action. This leads to a poor approximation of the greedy action.

Baselines: We again implement the same baselines discussed in Section 4. The baselines are PreDeToR, PreDeToR-τ, DPT-greedy, AD, Thomp, and LinUCB. During test time evaluation for offline setting the DPT selects $I_t = \widehat{a}_{m,t,*}$ where $\widehat{a}_{m,t,*} = \arg\max_a \text{TF}_{\mathbf{\Theta}}(a|\mathcal{H}^t_m)$ is the predicted optimal action.

Outcomes: We first discuss the main outcomes of our experimental results for increasing the horizon:

> **Finding 18:** PreDeToR ($-\tau$) performs comparably to DPT-greedy and AD in the offline setting.



(a) Offline for Linear setting (b) Offline for Non-linear setting

Figure 21: Offline experiment. The y-axis shows the cumulative reward.

Experimental Result: We observe these outcomes in Figure 21. In Figure 21 we show the linear bandit setting for horizon $n = 20$, $M_{\text{pre}} = 200000$, $M_{\text{test}} = 5000$, $A = 20$, and $d = 5$ for the low data regime. Again, the demonstrator π^w is the Thomp algorithm. We observe that PreDeToR ($-\tau$) has comparable cumulative regret to DPT-greedy. Note that for any task m for the horizon $n = 20$ the Thomp will be able to sample all the actions at most once. In the non-linear setting of Figure 21b the $n = 40$, $M_{\text{pre}} = 100000$, $A = 6$, $d = 2$. Observe that in all of these results, the performance of PreDeToR ($-\tau$) is comparable with respect to cumulative regret against DPT-greedy.

B Theoretical Analysis

B.1 Proof of Lemma 1

Proof. The learner collects n rounds of data following π^w. The weak demonstrator π^w only observes the $\{I_t, r_t\}_{t=1}^n$. Recall that $N_n(a)$ denotes the total number of times the action a is sampled for n rounds. Define the matrix $\mathbf{H}_n \in \mathbb{R}^{n \times A}$ where the t-th row represents the action sampled at round $t \in [n]$. The t-th row is a one-hot vector with 1 as the a-th component in the vector for $a \in [A]$. Then define the reward vector $\mathbf{Y}_n \in \mathbb{R}^n$ as the reward vector where the t-th component is the observed reward for the action I_t for $t \in [n]$. Finally define the diagonal matrix $\mathbf{D}_A \in \mathbb{R}^{A \times A}$ as in (6) and the estimated reward covariance matrix as $\mathbf{S}_A \in \mathbb{R}^{A \times A}$ such that $\mathbf{S}_A(a, a') = \widehat{\mu}_n(a)\widehat{\mu}_n(a')$. This matrix captures the reward correlation between the pairs of actions $a, a' \in [A]$.

Assume $\mu \sim \mathcal{N}(0, \mathbf{S}_*)$ where $\mathbf{S}_* \in \mathbb{R}^{A \times A}$. Then the observed mean vector $\mathbf{Y}_n$ is

$$\mathbf{Y}_n = \mathbf{H}_n \mu + \mathbf{H}_n \mathbf{D}_A^{1/2} \eta_n$$

where, η_n is the noise vector over the $[n]$ training data. Then the posterior mean of $\widehat{\mu}$ by Gauss Markov Theorem (Johnson et al., 2002) is given by

$$\widehat{\mu} = \mathbf{S}_* \mathbf{H}_n^\top \left(\mathbf{H}_n (\mathbf{S}_* + \mathbf{D}_A) \mathbf{H}_n^\top\right)^{-1} \mathbf{Y}_n. \tag{8}$$

However, the learner does not know the true reward co-variance matrix. Hence it needs to estimate the $\mathbf{S}_*$ from the observed data. Let the estimate be denoted by $\mathbf{S}_A$.

Assumption B.1. We assume that π^w is sufficiently exploratory so that each action is sampled at least once.

The Assumption B.1 ensures that the matrix $\left(\sigma_{\boldsymbol{\theta}}^2 \mathbf{H}_n (\mathbf{S}_A + \mathbf{D}_A) \mathbf{H}_n^\top\right)^{-1}$ is invertible. Under Assumption B.1, plugging the estimate $\mathbf{S}_A$ back in (8) shows that the average posterior mean over all the tasks is

$$\widehat{\mu} = \mathbf{S}_A \mathbf{H}_n^\top \left(\mathbf{H}_n (\mathbf{S}_A + \mathbf{D}_A) \mathbf{H}_n^\top \right)^{-1} \mathbf{Y}_n. \tag{9}$$

The claim of the lemma follows. □

C Generalization and Transfer Learning Proof for PreDeToR

C.1 Generalization Proof

Alg is the space of algorithms induced by the transformer $\mathrm{TF}_{\mathbf{\Theta}}$.

Theorem C.1. ***(PreDeToR risk)*** *Suppose error stability Assumption 8.1 holds and assume loss function $\ell(\cdot,\cdot)$ is C-Lipschitz for all $r_t \in [0, B]$ and horizon $n \geq 1$. Let $\widehat{\mathrm{TF}}$ be the empirical solution of (ERM) and $\mathcal{N}(\mathcal{A}, \rho, \epsilon)$ be the covering number of the algorithm space Alg following Definition C.2 and C.3. Then with probability at least $1 - 2\delta$, the excess Multi-task learning (MTL) risk of PreDeToR-τ is bounded by*

$$\mathcal{R}_{\mathrm{MTL}}(\widehat{\mathrm{TF}}) \leq 4\frac{C}{\sqrt{nM}} + 2(B + K \log n)\sqrt{\frac{\log(\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)/\delta)}{cnM}}$$

where, $\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)$ is the covering number of transformer $\widehat{\mathrm{TF}}$.

Proof. We consider a meta-learning setting. Let M source tasks are i.i.d. sampled from a task distribution $\mathcal{T}$, and let $\widehat{\mathrm{TF}}$ be the empirical Multitask (MTL) solution. Define $\mathcal{H}_{\text{all}} = \bigcup_{m=1}^M \mathcal{H}_m$. We drop the $\mathbf{\Theta}, \mathbf{r}$ from transformer notation $\mathrm{TF}^{\mathbf{r}}_{\mathbf{\Theta}}$ as we keep the architecture fixed as in Lin et al. (2023). Note that this transformer predicts a reward vector over the actions. To be more precise we denote the reward predicted by the transformer at round t after observing history $\mathcal{H}_m^{t-1}$ and then sampling the action a_{mt} as $\mathrm{TF}\left(\widehat{r}_{mt}(a_{mt})|\mathcal{H}_m^{t-1}, a_{mt}\right)$. Define the training risk

$$\widehat{\mathcal{L}}_{\mathcal{H}_{\text{all}}}(\mathrm{TF}) = \frac{1}{nM} \sum_{m=1}^{M} \sum_{t=1}^{n} \ell\left(r_{mt}(a_{mt}), \mathrm{TF}\left(\widehat{r}_{mt}(a_{mt})|\mathcal{H}_m^{t-1}, a_{mt}\right)\right)$$

and the test risk

$$\mathcal{L}_{\mathrm{MTL}}(\mathrm{TF}) = \mathbb{E}\left[\widehat{\mathcal{L}}_{\mathcal{H}_{\text{all}}}(\mathrm{TF})\right].$$

Define empirical risk minima $\widehat{\mathrm{TF}} = \arg\min_{\mathrm{TF} \in \mathrm{Alg}} \widehat{\mathcal{L}}_{\mathcal{H}_{\text{all}}}(\mathrm{TF})$ and population minima

$$\mathrm{TF}^* = \arg\min_{\mathrm{TF} \in \mathrm{Alg}} \mathcal{L}_{\mathrm{MTL}}(\mathrm{TF})$$

In the following discussion, we drop the subscripts MTL and $\mathcal{H}_{\text{all}}$. The excess MTL risk is decomposed as follows:

$$\begin{aligned}
\mathcal{R}_{\mathrm{MTL}}(\widehat{\mathrm{TF}}) &= \mathcal{L}(\widehat{\mathrm{TF}}) - \mathcal{L}(\mathrm{TF}^*) \\
&= \underbrace{\mathcal{L}(\widehat{\mathrm{TF}}) - \widehat{\mathcal{L}}(\widehat{\mathrm{TF}})}_{a} + \underbrace{\widehat{\mathcal{L}}(\widehat{\mathrm{TF}}) - \widehat{\mathcal{L}}(\mathrm{TF}^*)}_{b} + \underbrace{\widehat{\mathcal{L}}(\mathrm{TF}^*) - \mathcal{L}(\mathrm{TF}^*)}_{c}.
\end{aligned}$$

Since $\widehat{\mathrm{TF}}$ is the minimizer of empirical risk, we have $b \leq 0$.

Step 1: (Concentration bound $|\mathcal{L}(\mathrm{TF}) - \widehat{\mathcal{L}}(\mathrm{TF})|$ for a fixed $\mathrm{TF} \in \mathrm{Alg}$) Define the test/train risks of each task as follows:

$$\widehat{\mathcal{L}}_m(\mathrm{TF}) := \frac{1}{n} \sum_{t=1}^{n} \ell\left(r_{mt}(a_{mt}), \mathrm{TF}\left(\widehat{r}_{mt}(a_{mt})|\mathcal{H}_m^{t-1}, a_{mt}\right)\right), \quad \text{and}$$

$$\mathcal{L}_m(\mathrm{TF}) := \mathbb{E}_{\mathcal{H}_m}\left[\widehat{\mathcal{L}}_m(\mathrm{TF})\right] = \mathbb{E}_{\mathcal{H}_m}\left[\frac{1}{n} \sum_{t=1}^{n} \ell\left(r_{mt}(a_{mt}), \mathrm{TF}\left(\widehat{r}_{mt}(a_{mt})|\mathcal{H}_m^{t-1}, a_{mt}\right)\right)\right], \quad \forall m \in [M].$$

Define the random variables $X_{m,t} = \mathbb{E}\left[\widehat{\mathcal{L}}_t(\mathrm{TF}) \mid \mathcal{H}_m^t\right]$ for $t \in [n]$ and $m \in [M]$, that is, $X_{m,t}$ is the expectation over $\widehat{\mathcal{L}}_t(\mathrm{TF})$ given training sequence $\mathcal{H}_m^t = \{(a_{mt'}, r_{mt'})\}_{t'=1}^t$ (which are the filtrations). With this, we have that $X_{m,n} = \mathbb{E}\left[\widehat{\mathcal{L}}_m(\mathrm{TF}) \mid \mathcal{H}_m^n\right] = \widehat{\mathcal{L}}_m(\mathrm{TF})$ and $X_{m,0} = \mathbb{E}\left[\widehat{\mathcal{L}}_m(\mathrm{TF})\right] = \mathcal{L}_m(\mathrm{TF})$. More generally, $(X_{m,0}, X_{m,1}, \ldots, X_{m,n})$ is a martingale sequence since, for every $m \in [M]$, we have that $\mathbb{E}\left[X_{m,i} \mid \mathcal{H}_m^{t-1}\right] = X_{m,t-1}$. For notational simplicity, in the following discussion, we omit the subscript m from a, r and $\mathcal{H}$ as they will be clear from the left-hand-side variable $X_{m,t}$. We have that

$$\begin{aligned} X_{m,t} &= \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \ell\left(r_{t'}, \mathrm{TF}\left(\widehat{r}_{t'}|\mathcal{H}^{t'-1}, a_{t'}\right)\right)\middle|\, \mathcal{H}^t\right] \\ &= \frac{1}{n}\sum_{t'=1}^{t} \ell\left(r_{t'}, \mathrm{TF}\left(\widehat{r}_{t'}|\mathcal{H}^{t'-1}, a_{t'}\right)\right) + \frac{1}{n}\sum_{t'=t+1}^{n} \mathbb{E}\left[\ell\left(r_{t'}, \mathrm{TF}\left(\widehat{r}_{t'}|\mathcal{H}^{t'-1}, a_{t'}\right)\right) \mid \mathcal{H}^t\right] \end{aligned}$$

Using the similar steps as in Li et al. (2023) we can show that

$$|X_{m,t} - X_{m,t-1}| \overset{(a)}{\leq} \frac{B}{n} + \sum_{t'=t+1}^{n} \frac{K}{t'n} \leq \frac{B + K\log n}{n}.$$

where, (a) follows by using the fact that loss function $\ell(\cdot,\cdot)$ is bounded by B, and error stability assumption.

Recall that $\left|\mathcal{L}_m(\mathrm{TF}) - \widehat{\mathcal{L}}_m(\mathrm{TF})\right| = |X_{m,0} - X_{m,n}|$ and for every $m \in [M]$, we have $\sum_{t=1}^n |X_{m,t} - X_{m,t-1}|^2 \leq \frac{(B+K\log n)^2}{n}$. As a result, applying Azuma-Hoeffding's inequality, we obtain

$$\mathbb{P}\left(\left|\mathcal{L}_m(\mathrm{TF}) - \widehat{\mathcal{L}}_m(\mathrm{TF})\right| \geq \tau\right) \leq 2e^{-\frac{n\tau^2}{2(B+K\log n)^2}}, \quad \forall m \in [M] \tag{10}$$

Let us consider $Y_m := \mathcal{L}_m(\mathrm{TF}) - \widehat{\mathcal{L}}_m(\mathrm{TF})$ for $m \in [M]$. Then, $(Y_m)_{m=1}^M$ are i.i.d. zero mean sub-Gaussian random variables. There exists an absolute constant $c_1 > 0$ such that, the subgaussian norm, denoted by $\|\cdot\|_{\psi_2}$, obeys $\|Y_m\|_{\psi_2}^2 < \frac{c_1(B+K\log n)^2}{n}$ via Proposition 2.5.2 of (Vershynin, 2018). Applying Hoeffding's inequality, we derive

$$\mathbb{P}\left(\left|\frac{1}{M}\sum_{m=1}^{M} Y_t\right| \geq \tau\right) \leq 2e^{-\frac{cnM\tau^2}{(B+K\log n)^2}} \Longrightarrow \mathbb{P}(|\widehat{\mathcal{L}}(\mathrm{TF}) - \mathcal{L}(\mathrm{TF})| \geq \tau) \leq 2e^{-\frac{cnM\tau^2}{(B+K\log n)^2}}$$

where $c > 0$ is an absolute constant. Therefore, we have that for any $\mathrm{TF} \in \mathrm{Alg}$, with probability at least $1 - 2\delta$,

$$|\widehat{\mathcal{L}}(\mathrm{TF}) - \mathcal{L}(\mathrm{TF})| \leq (B + K\log n)\sqrt{\frac{\log(1/\delta)}{cnM}} \tag{11}$$

Step 2: (Bound $\sup_{\mathrm{TF}\in\mathrm{Alg}} |\mathcal{L}(\mathrm{TF}) - \widehat{\mathcal{L}}(\mathrm{TF})|$ **where** Alg **is assumed to be a continuous search space)**. Let

$$h(\mathrm{TF}) := \mathcal{L}(\mathrm{TF}) - \widehat{\mathcal{L}}(\mathrm{TF})$$

and we aim to bound $\sup_{\mathrm{TF}\in\mathrm{Alg}} |h(\mathrm{TF})|$. Following Definition C.3, for $\varepsilon > 0$, let Alg_ε be a minimal ε-cover of Alg in terms of distance metric ρ. Therefore, Alg_ε is a discrete set with cardinality $|\mathrm{Alg}_\varepsilon| := \mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)$. Then, we have

$$\sup_{\mathrm{TF}\in\mathrm{Alg}} |\mathcal{L}(\mathrm{TF}) - \widehat{\mathcal{L}}(\mathrm{TF})| \leq \sup_{\mathrm{TF}\in\mathrm{Alg}'} \min_{\mathrm{TF}\in\mathrm{Alg}_\varepsilon} |h(\mathrm{TF}) - h(\mathrm{TF}')| + \max_{\mathrm{TF}\in\mathrm{Alg}_\varepsilon} |h(\mathrm{TF})|.$$

We will first bound the quantity $\sup_{\mathrm{TF}\in\mathrm{Alg}'}\min_{\mathrm{TF}\in\mathrm{Alg}_\varepsilon}|h(\mathrm{TF})-h(\mathrm{TF}')|$. We will utilize that loss function $\ell(\cdot,\cdot)$ is C-Lipschitz. For any $\mathrm{TF}\in\mathrm{Alg}$, let $\mathrm{TF}\in\mathrm{Alg}_\varepsilon$ be its neighbor following Definition C.3. Then we can show that

$$
\begin{aligned}
&\left|\widehat{\mathcal{L}}(\mathrm{TF})-\widehat{\mathcal{L}}\left(\mathrm{TF}'\right)\right| \\
&=\left|\frac{1}{nM}\sum_{m=1}^{M}\sum_{t=1}^{n}\left(\ell\left(r_{mt}(a_{mt}),\mathrm{TF}\left(\widehat{r}_{mt}(a_{mt})|\mathcal{H}_m^{t-1},a_{mt}\right)\right)-\ell\left(r_{mt}(a_{mt}),\mathrm{TF}'\left(\widehat{r}_{mt}(a_{mt})|\mathcal{H}_m^{t-1},a_{mt}\right)\right)\right)\right| \\
&\leq\frac{L}{nM}\sum_{m=1}^{M}\sum_{t=1}^{n}\left\|\mathrm{TF}\left(\widehat{r}_{mt}(a_{mt})|\mathcal{H}_m^{t-1},a_{mt}\right)-\mathrm{TF}'\left(\widehat{r}_{mt}(a_{mt})|\mathcal{H}_m^{t-1},a_{mt}\right)\right\|_{\ell_2} \\
&\leq L\varepsilon.
\end{aligned}
$$

Note that the above bound applies to all data-sequences, we also obtain that for any $\mathrm{TF}\in\mathrm{Alg}$,

$$\left|\mathcal{L}(\mathrm{TF})-\mathcal{L}\left(\mathrm{TF}'\right)\right|\leq L\varepsilon.$$

Therefore we can show that,

$$
\begin{aligned}
&\sup_{\mathrm{TF}\in\mathrm{Alg}}\min_{\mathrm{TF}}\in\mathrm{Alg}_\varepsilon\left|h(\mathrm{TF})-h\left(\mathrm{TF}F'\right)\right| \\
&\qquad\leq\sup_{\mathrm{TF}\in\mathrm{Alg}}\min_{\mathrm{TF}}\in\mathrm{Alg}_\varepsilon\left|\widehat{\mathcal{L}}(\mathrm{TF})-\widehat{\mathcal{L}}\left(\mathrm{TF}'\right)\right|+\left|\mathcal{L}(\mathrm{TF})-\mathcal{L}\left(\mathrm{TF}'\right)\right|\leq 2L\varepsilon. \qquad (12)
\end{aligned}
$$

Next we bound the second term $\max_{\mathrm{TF}\in\mathrm{Alg}_\varepsilon}|h(\mathrm{TF})|$. Applying union bound directly on Alg_ε and combining it with (11), then we will have that with probability at least $1-2\delta$,

$$\max_{\mathrm{TF}\in\mathrm{Alg}_\varepsilon}|h(\mathrm{TF})|\leq(B+K\log n)\sqrt{\frac{\log(\mathcal{N}(\mathrm{Alg},\rho,\varepsilon)/\delta)}{cnM}}$$

Combining the upper bound above with the perturbation bound (12), we obtain that

$$\max_{\mathrm{TF}\in\mathrm{Alg}}|h(\mathrm{TF})|\leq 2C\varepsilon+(B+K\log n)\sqrt{\frac{\log(\mathcal{N}(\mathrm{Alg},\rho,\varepsilon)/\delta)}{cnM}}.$$

It follows then that

$$\mathcal{R}_{\mathrm{MTL}}(\widehat{\mathrm{TF}})\leq 2\sup_{\mathrm{TF}\in\mathrm{Alg}}|\mathcal{L}(\mathrm{TF})-\widehat{\mathcal{L}}(\mathrm{TF})|\leq 4C\varepsilon+2(B+K\log n)\sqrt{\frac{\log(\mathcal{N}(\mathrm{Alg},\rho,\varepsilon)/\delta)}{cnM}}$$

Again by setting $\varepsilon=1/\sqrt{nM}$

$$\mathcal{L}(\widehat{\mathrm{TF}})-\mathcal{L}\left(\mathrm{TF}^*\right)\leq\frac{4C}{\sqrt{nM}}+2(B+K\log n)\sqrt{\frac{\log(\mathcal{N}(\mathrm{Alg},\rho,\varepsilon)/\delta)}{cnM}}$$

The claim of the theorem follows. □

Definition C.2. (Covering number) Let Q be any hypothesis set and $d\left(q,q'\right)\geq 0$ be a distance metric over $q,q'\in\mathcal{Q}$. Then, $\bar{Q}=\{q_1,\ldots,q_N\}$ is an ε-cover of Q with respect to $d(\cdot,\cdot)$ if for any $q\in\mathcal{Q}$, there exists $q_i\in\bar{Q}$ such that $d\left(q,q_i\right)\leq\varepsilon$. The ε-covering number $\mathcal{N}(Q,d,\varepsilon)$ is the cardinality of the minimal ε-cover.

Definition C.3. (Algorithm distance). Let Alg be an algorithm hypothesis set and $\mathcal{H}=(a_t,r_t)_{t=1}^n$ be a sequence that is admissible for some task $m\in[M]$. For any pair $\mathrm{TF},\mathrm{TF}'\in\mathrm{Alg}$, define the distance metric $\rho\left(\mathrm{TF},\mathrm{TF}'\right):=\sup_{\mathcal{H}}\frac{1}{n}\sum_{t=1}^{n}\left\|\mathrm{TF}\left(\widehat{r}_t|\mathcal{H}^{t-1},a_t\right)-\mathrm{TF}'\left(\widehat{r}_t|\mathcal{H}^{t-1},a_t\right)\right\|_{\ell_2}$.

Remark C.4. **(Stability Factor)** The work of Li et al. (2023) also characterizes the stability factor K in Assumption 8.1 with respect to the transformer architecture. Assuming loss $\ell(\cdot,\cdot)$ is C-Lipschitz, the algorithm induced by $\mathrm{TF}(\cdot)$ obeys the stability assumption with $K = 2C\left((1+\Gamma)e^{\Gamma}\right)^{L}$, where the norm of the transformer weights are upper bounded by $O(\Gamma)$ and there are L-layers of the transformer.

Remark C.5. **(Covering Number)** From Lemma 16 of Lin et al. (2023) we have the following upper bound on the covering number of the transformer class $\mathrm{TF}_{\boldsymbol{\Theta}}$ as

$$\log(\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)) \leq O(L^2 D^2 J)$$

where L is the total number of layers of the transformer and J and, D denote the upper bound to the number of heads and hidden neurons in all the layers respectively. Note that this covering number holds for the specific class of transformer architecture discussed in section 2 of (Lin et al., 2023).

C.2 Generalization Error to New Task

Theorem C.6. ***(Transfer Risk)*** *Consider the setting of Theorem 8.2 and assume the source tasks are independently drawn from task distribution $\mathcal{T}$. Let $\widehat{TF}$ be the empirical solution of (ERM) and $g \sim \mathcal{T}$. Then with probability at least $1 - 2\delta$, the expected excess transfer learning risk is bounded by*

$$\mathbb{E}_g\left[\mathcal{R}_g(\widehat{\mathrm{TF}})\right] \leq 4\frac{C}{\sqrt{M}} + B\sqrt{\frac{2\log(\mathcal{N}(\mathrm{Alg},\rho,\varepsilon)/\delta)}{M}}$$

where, $\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)$ is the covering number of transformer $\widehat{\mathrm{TF}}$.

Proof. Let the target task g be sampled from $\mathcal{T}$, and the test set $\mathcal{H}_g = \{a_t, r_t\}_{t=1}^n$. Define empirical and population risks on g as $\widehat{\mathcal{L}}_g(\mathrm{TF}) = \frac{1}{n}\sum_{t=1}^n \ell\left(r_t(a_{mt}), \mathrm{TF}\left(\widehat{r}_t(a_{mt})|\mathcal{H}_g^{t-1}, a_t\right)\right)$ and $\mathcal{L}_g(\mathrm{TF}) = \mathbb{E}_{\mathcal{H}_g}\left[\widehat{\mathcal{L}}_g(\mathrm{TF})\right]$. Again we drop $\boldsymbol{\Theta}$ from the transformer notation. Then the expected excess transfer risk following (ERM) is defined as

$$\mathbb{E}_g\left[\mathcal{R}_g(\widehat{\mathrm{TF}})\right] = \mathbb{E}_{\mathcal{H}_g}\left[\mathcal{L}_g(\widehat{\mathrm{TF}})\right] - \arg\min_{\mathrm{TF}\in\mathrm{Alg}} \mathbb{E}_{\mathcal{H}_g}\left[\mathcal{L}_g(\mathrm{TF})\right]. \quad (13)$$

where $\mathcal{A}$ is the set of all algorithms. The goal is to show a bound like this

$$\mathbb{E}_g\left[\mathcal{R}_g(\widehat{\mathrm{TF}})\right] \leq \min_{\varepsilon\geq 0}\left\{4C\varepsilon + B\sqrt{\frac{2\log(\mathcal{N}(\mathrm{Alg},\rho,\varepsilon)/\delta)}{T}}\right\}$$

where $\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)$ is the covering number.

Step 1 ((Decomposition): Let $\mathrm{TF}^* = \arg\min_{\mathrm{TF}\in\mathrm{Alg}} \mathbb{E}_g\left[\mathcal{L}_g(\mathrm{TF})\right]$. The expected transfer learning excess test risk of given algorithm $\widehat{\mathrm{TF}} \in \mathrm{Alg}$ is formulated as

$$\widehat{\mathcal{L}}_m(\mathrm{TF}) := \frac{1}{n}\sum_{t=1}^{n} \ell\left(r_{mt}(a_{mt}), \mathrm{TF}\left(\widehat{r}_{mt}(a_{mt})|\mathcal{D}_m^{t-1}, a_{mt}\right)\right), \quad \text{and}$$

$$\mathcal{L}_m(\mathrm{TF}) := \mathbb{E}_{\mathcal{H}_m}\left[\widehat{\mathcal{L}}_t(\mathrm{TF})\right] = \mathbb{E}_{\mathcal{H}_m}\left[\frac{1}{n}\sum_{t=1}^{n} \ell\left(r_{mt}(a_{mt}), \mathrm{TF}\left(\widehat{r}_{mt}(a_{mt})|\mathcal{D}_m^{t-1}, a_{mt}\right)\right)\right], \quad \forall m \in [M].$$

Then we can decompose the risk as

$$\begin{aligned}\mathbb{E}_g\left[\mathcal{R}_g(\widehat{\mathrm{TF}})\right] &= \mathbb{E}_g\left[\mathcal{L}_g(\widehat{\mathrm{TF}})\right] - \mathbb{E}_g\left[\mathcal{L}_g\left(\mathrm{TF}^*\right)\right] \\ &= \underbrace{\mathbb{E}_g\left[\mathcal{L}_g(\widehat{\mathrm{TF}})\right] - \widehat{\mathcal{L}}_{\mathcal{H}_{\text{all}}}(\widehat{\mathrm{TF}})}_{a} + \underbrace{\widehat{\mathcal{L}}_{\mathcal{H}_{\text{all}}}(\widehat{\mathrm{TF}}) - \widehat{\mathcal{L}}_{\mathcal{H}_{\text{all}}}\left(\mathrm{TF}^*\right)}_{b} + \underbrace{\widehat{\mathcal{L}}_{\mathcal{H}_{\text{all}}}\left(\mathrm{TF}^*\right) - \mathbb{E}_g\left[\mathcal{L}_g\left(\mathrm{TF}^*\right)\right]}_{c}.\end{aligned}$$

Here since $\widehat{\mathrm{TF}}$ is the minimizer of training risk, $b < 0$. Then we obtain

$$\mathbb{E}_g\left[\mathcal{R}_g(\widehat{\mathrm{TF}})\right] \leq 2 \sup_{\mathrm{TF}\in\mathrm{Alg}} \left|\mathbb{E}_g\left[\mathcal{L}_g(\mathrm{TF})\right] - \frac{1}{M}\sum_{m=1}^{M}\widehat{\mathcal{L}}_m(\mathrm{TF})\right|. \tag{14}$$

Step 2 (Bounding (14)) For any $\mathrm{TF} \in \mathrm{Alg}$, let $X_t = \widehat{\mathcal{L}}_t(\mathrm{TF})$ and we observe that

$$\mathbb{E}_{m\sim\mathcal{T}}\left[X_t\right] = \mathbb{E}_{m\sim\mathcal{T}}\left[\widehat{\mathcal{L}}_m(\mathrm{TF})\right] = \mathbb{E}_{m\sim\mathcal{T}}\left[\mathcal{L}_m(\mathrm{TF})\right] = \mathbb{E}_g\left[\mathcal{L}_g(\mathrm{TF})\right]$$

Since $X_m, m \in [M]$ are independent, and $0 \leq X_m \leq B$, applying Hoeffding's inequality obeys

$$\mathbb{P}\left(\left|\mathbb{E}_g\left[\mathcal{L}_g(\mathrm{TF})\right] - \frac{1}{M}\sum_{m=1}^{M}\widehat{\mathcal{L}}_m(\mathrm{TF})\right| \geq \tau\right) \leq 2e^{-\frac{2M\tau^2}{B^2}}.$$

Then with probability at least $1 - 2\delta$, we have that for any $\mathrm{TF} \in \mathrm{Alg}$,

$$\left|\mathbb{E}_g\left[\mathcal{L}_g(\mathrm{TF})\right] - \frac{1}{M}\sum_{m=1}^{M}\widehat{\mathcal{L}}_m(\mathrm{TF})\right| \leq B\sqrt{\frac{\log(1/\delta)}{2M}}. \tag{15}$$

Next, let Alg_ε be the minimal ε-cover of Alg following Definition C.2, which implies that for any task $g \sim \mathcal{T}$, and any $\mathrm{TF} \in \mathrm{Alg}$, there exists $\mathrm{TF}' \in \mathrm{Alg}_\varepsilon$

$$\left|\mathcal{L}_g(\mathrm{TF}) - \mathcal{L}_g\left(\mathrm{TF}'\right)\right|, \left|\widehat{\mathcal{L}}_g(\mathrm{TF}) - \widehat{\mathcal{L}}_g\left(\mathrm{TF}'\right)\right| \leq C\varepsilon. \tag{16}$$

Since the distance metric following Definition 3.4 is defined by the worst-case datasets, then there exists $\mathrm{TF}' \in \mathrm{Alg}_\varepsilon$ such that

$$\left|\mathbb{E}_g\left[\mathcal{L}_g(\mathrm{TF})\right] - \frac{1}{M}\sum_{m=1}^{M}\widehat{\mathcal{L}}_m(\mathrm{TF})\right| \leq 2C\varepsilon.$$

Let $\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon) = |\mathrm{Alg}_\varepsilon|$ be the ε-covering number. Combining the above inequalities ((14), (15), and (16)), and applying union bound, we have that with probability at least $1 - 2\delta$,

$$\mathbb{E}_g\left[\mathcal{R}_g(\widehat{\mathrm{TF}})\right] \leq \min_{\varepsilon\geq 0}\left\{4C\varepsilon + B\sqrt{\frac{2\log(\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)/\delta)}{M}}\right\}$$

Again by setting $\varepsilon = 1/\sqrt{M}$

$$\mathcal{L}(\widehat{\mathrm{TF}}) - \mathcal{L}\left(\mathrm{TF}^*\right) \leq \frac{4C}{\sqrt{M}} + 2B\sqrt{\frac{\log(\mathcal{N}(\mathrm{Alg}, \rho, \varepsilon)/\delta)}{cM}}$$

The claim of the theorem follows. ☐

Remark C.7. (Dependence on n) In this remark, we briefly discuss why the expected excess risk for target task $\mathcal{T}$ does not depend on samples n. The work of Li et al. (2023) pointed out that the MTL pretraining process identifies a favorable algorithm that lies in the span of the M source tasks. This is termed as inductive bias (see section 4 of Li et al. (2023)) (Soudry et al., 2018; Neyshabur et al., 2017). Such bias would explain the lack of dependence of the expected excess transfer risk on n during transfer learning. This is because given a target task $g \sim \mathcal{T}$, the TF can use the learnt favorable algorithm to conduct a discrete search over span of the M source tasks and return the source task that best fits the new target task. Due to the discrete search space over the span of M source tasks, it is not hard to see that, we need $n \propto \log(M)$ samples (which is guaranteed by the M source tasks) rather than $n \propto d$ (for the linear setting).

C.3 Table of Notations

Notations	**Definition**
M	Total number of tasks
d	Dimension of the feature
$\mathcal{A}_m$	Action set of the m-th task
$\mathcal{X}_m$	Feature space of m-th task
M_{test}	Tasks for testing
M_{pre}	Total Tasks for pretraining
$\mathbf{x}(m, a)$	Feature of action a in task m
$\boldsymbol{\theta}_{m,*}$	Hidden parameter for the task m
$\mathcal{T}_{\text{pre}}$	Pretraning distribution on tasks
$\mathcal{T}_{\text{test}}$	Testing distribution on tasks
n	Total horizon for each task m
$\mathcal{H}_m = \{I_t, r_t\}_{t=1}^n$	Dataset sampled for the m-th task containing n samples
$\mathcal{H}_m^t = \{I_s, r_s\}_{s=1}^t$	Dataset sampled for the m-th task containing samples from round $s = 1$ to t
$\mathbf{w}$	Transformer model parameter
$\mathrm{TF}_{\mathbf{w}}$	Transformer with model parameter $\mathbf{w}$
$\mathcal{D}_{\text{pre}}$	Pretraining in-context distribution
$\mathcal{H}_{\text{train}}$	Training in-context dataset
$\mathcal{D}_{\text{test}}$	Testing in-context distribution

Table 1: Table of Notations