Decision Transformer with GoFAR Can Learn In-Context Reinforcement Learning

Fred Zhang[1] **Jiaxin Ye**[2]
[1]Yale University, [2]UC San Diego

Abstract

# TODO

1 Introduction

# TODO

2 Related Work

Meta-learning. # TODO Algorithmically, in-context learning falls under the meta-learning framework [1, 2].

Autoregressive transformers for decision-making. # TODO

Value and policy-based offline RL. # TODO

3 Preliminary

Basic decision models. The basic decision model of our study is the finite-horizon Markov decision process (MDP). An MDP is specified by the tuple $\tau = \langle \mathcal{S}, \mathcal{A}, T, R, H, \rho \rangle$ to be solved, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ is the reward function, $H \in \mathbb{N}$ is the horizon, and $\rho \in \Delta(\mathcal{S})$ is the initial state distribution. A learner interacts with the environment through the following protocol: (1) an initial state s_1 is sampled from ρ; (2) at time step h, the learner chooses an action a_h and transitions to state $s_{h+1} \sim T(\cdot|s_h, a_h)$, and receives a reward $r_h \sim R(\cdot|s_h, a_h)$. The episode ends after H steps. A policy π maps states to distributions over actions and can be used to interact with the MDP. We denote the optimal policy as $\pi^\star$, which maximizes the value function $V(\pi^\star) = \max_\pi V(\pi) := \max_\pi \mathbb{E}_\pi \sum_h r_h$. When necessary, we use the subscript τ to distinguish V_τ and $\pi^\star_\tau$ for the specific MDP τ. We assume the state space is partitioned by $h \in [H]$ so that $\pi^\star$ is notationally independent of h. Note this framework encompasses multi-armed bandit settings where the state space is a single point, e.g. $\mathcal{S} = \{1\}$, $H = 1$, and the optimal policy is $a^\star \in \operatorname{argmax}_{a \in \mathcal{A}} \mathbb{E}[r_1 | a_1 = a]$.

Pretraining. Let $\mathcal{T}_{\text{pre}}$ be a distribution over tasks at the time of pretraining. A task $\tau \sim \mathcal{T}_{\text{pre}}$ can be viewed as a specification of an MDP, $\tau = \langle \mathcal{S}, \mathcal{A}, T, R, H, \rho \rangle$. The distribution $\mathcal{T}_{\text{pre}}$ can span different reward and transition functions and even different state and action spaces. We then sample a context (or a prompt) which consists of a dataset $D \sim \mathcal{D}_{\text{pre}}(\cdot; \tau)$ of interactions between the learner and the MDP specified by τ. $D = \{s_j, a_j, s'_j, r_j\}_{j \in [n]}$ is a collection of transition tuples taken in τ. We refer to D as the *in-context dataset* because it provides the contextual information about τ. D could be generated through variety of means, such as: (1) random interactions within τ, (2) demonstrations from an expert, and (3) rollouts of an algorithm. Additionally, we independently sample a query state s_{query} from the distribution $\mathcal{D}_{\text{query}}$ over states $\mathcal{S}$ and a label $a^\star$ is sampled from the optimal policy $\pi^\star_\tau(\cdot|s_{\text{query}})$ for task τ (see Section **??** for how to implement this in common practical scenarios). We denote the joint pretraining distribution over tasks, in-context datasets, query states, and action labels as P_{pre}:

$$P_{pre}(\tau, D, s_{\text{query}}, a^\star) = \mathcal{T}_{\text{pre}}(\tau)\mathcal{D}_{\text{pre}}(D;\tau)\mathcal{D}_{\text{query}}(s_{\text{query}})\pi^\star_\tau(a^\star|s_{\text{query}}) \tag{1}$$

Given the in-context dataset D and a query state s_{query}, we can train a model to predict the optimal action $a^\star$ in response simply via supervised learning. Let $D_j = \{(s_1, a_1, s'_1, r_1), \ldots, (s_j, a_j, s'_j, r_j)\}$ denote the partial dataset up to j samples. Formally, we aim to train a causal GPT-2 transformer model M parameterized by θ, which outputs a distribution over actions $\mathcal{A}$, to minimize the expected loss over samples from the pretraining distribution:

$$\min_\theta \mathbb{E}_{P_{pre}} \textstyle\sum_{j\in[n]} \ell\left(M_\theta(\cdot \mid s_{\text{query}}, D_j), a^\star\right) \tag{2}$$

Generally, we set the loss to be the negative log-likelihood with $\ell(M_\theta(\cdot\ |s_{\text{query}}, D_j), a^\star) := -\log M_\theta(a^\star \mid s_{\text{query}}, D_j)$. This framework can work for both discrete and continuous $\mathcal{A}$. For our experiments with discrete $\mathcal{A}$, we use a softmax parameterization for the distribution of M_θ, essentially treating this as a classification problem. The resulting output model M_θ can be viewed as an algorithm that takes in a dataset of interactions D and can be queried with a forward pass for predictions of the optimal action via inputting a query state s_{query}. We refer to the trained model M_θ as a Decision-Pretrained Transformer (DPT).

Testing. After pretraining, a new task (MDP) τ is sampled from a test-task distribution $\mathcal{T}_{\text{test}}$. If the DPT is to be tested *offline*, then a dataset (prompt) is a sampled $D \sim \mathcal{D}_{\text{test}}(\,\cdot\,;\tau)$ and the policy that the model in-context learns is given conditionally as $M_\theta(\cdot \mid \cdot, D)$. Namely, we evaluate the policy by selecting action $a_h \in \operatorname{argmax}_a M_\theta(a|s_h, D)$ when the learner visits state s_h. If the model is to be tested *online* through multiple episodes of interaction, then the dataset is initialized as empty $D = \{\}$. At each episode, $M_\theta(\cdot \mid \cdot, D)$ is deployed where the model samples $a_h \sim M_\theta(\cdot|s_h, D)$ upon observing state s_h. Throughout a full episode, it collects interactions $\{s_1, a_1, r_1, \ldots, s_H, a_H, r_H\}$ which are subsequently appended to D. The model then repeats the process with another episode, and so on until a specified number of episodes has been reached.

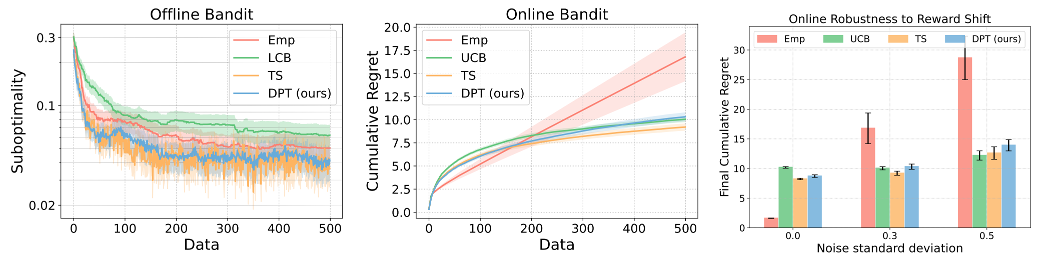
A key distinction of the testing phase is that there are no updates to the parameters of M_θ. This is in contrast to hand-designed RL algorithms that would perform parameter updates or maintain statistics using D to learn from scratch. Instead, the model M_θ performs a computation through its forward pass to generate a distribution over actions conditioned on the in-context D and query state s_h.

Sources of distribution mismatch. Inherent to pretraining, like nearly all foundation models, is distribution mismatch on downstream test-time tasks. DPT pretrained on sufficiently diverse data should ideally be robust (to some extent) to these mismatches. (1) When deployed, M_θ will execute its learned policy which invariably induces a distribution over states different from $\mathcal{D}_{\text{query}}$. (2) Pretraining $\mathcal{T}_{\text{pre}}$ likely differs from the downstream $\mathcal{T}_{\text{test}}$. (3) Similarly, the test-time datasets prompts can also differ, especially online where they are collected by M_θ itself.

4 Learning in Bandits

We begin with an empirical investigation of DPT in a multi-armed bandit, a well-studied special case of the MDP where the state space $\mathcal{S}$ is a singleton and the horizon $H = 1$ is a single step. We will examine the performance of DPT both when aiming to select a good action from offline historical data and for online learning where the goal is to maximize cumulative reward from scratch. Offline, it is critical to account for uncertainty due to noise as certain actions may not be sampled well enough. Online, it is critical to judiciously balance exploration and exploitation to minimize overall regret. For detailed descriptions of the experiment setups, see Appendix A.

Pretraining distribution. For the pretraining task distribution $\mathcal{T}_{\text{pre}}$, we sample 5-armed bandits ($|\mathcal{A}| = 5$). The reward function for arm a is a normal distribution $R(\cdot|s, a) = \mathcal{N}(\mu_a, \sigma^2)$ where $\mu_a \sim \text{Unif}[0, 1]$ independently and $\sigma = 0.3$. To generate in-context datasets $\mathcal{D}_{\text{pre}}$, we randomly generate action frequencies by sampling probabilities from a Dirichlet distribution and mixing them with a point-mass distribution on one random arm (see details in Appendix A.3). Then we sample the actions accordingly from this distribution. This encourages diversity of the in-context datasets. The optimal policy $\pi^\star_\tau$ for bandit τ is $\operatorname{argmax}_a \mu_a$, which we can easily compute during pretraining. We pretrain the model M_θ to predict $a^\star$ from D as described in Section 3 for datasets up to size $n = 500$.



(a) (b) (c)

Figure 1: (a) Offline performance on in-distribution bandits, given random in-context datasets. (b) Online cumulative regret on bandits. (c) Final (after 500 steps) cumulative regret on out-of-distribution bandits with different Gaussian noise standard deviations. The mean and standard error are computed over 200 test tasks.

Comparisons. We compare to several well-known algorithms for bandits[1]. All of the algorithms are designed to reason in a particular way about uncertainty based on their observations.

- Empirical mean algorithm (Emp) selects the action with the highest empirical mean reward naively.
- Upper Confidence Bound (UCB) selects the action with the highest upper confidence bound.
- Lower Confidence Bound (LCB) selects the action with the highest lower confidence bound.
- Thompson Sampling (TS) selects the action with the highest sampled mean from a posterior distribution over reward models. The prior and likelihood functions are Gaussian.

Emp and TS [3, 4] can both be used for offline or online learning; UCB [5] is known to be provably optimal online by ensuring exploration through optimism under uncertainty; and LCB [6, 7] is used to minimize suboptimality given an offline dataset by selecting actions pessimistically. It is the opposite of UCB. We evaluate algorithms with standard bandit metrics. Offline, we use the suboptimality $\mu_{a^\star} - \mu_{\hat{a}}$ where $\hat{a}$ is the chosen action. Online, we use cumulative regret: $\sum_k \mu_{a^\star} - \mu_{\hat{a}_k}$ where $\hat{a}_k$ is the kth action chosen.

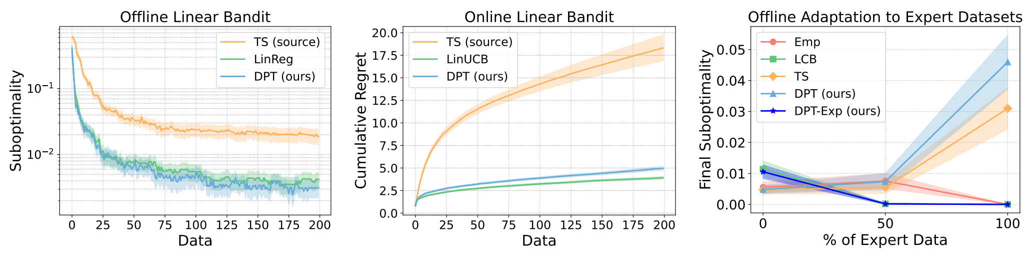
DPT learns to reason through uncertainty. As shown in Figure 1a, in the offline setting, DPT significantly exceeds the performance of Emp and LCB while matching the performance of TS, when the in-context datasets are sampled from the same distribution as during pretraining. The results suggest that the transformer is capable of reasoning through uncertainty caused by the noisy rewards in the dataset. Unlike Emp which can be fooled by noisy, undersampled actions, the transformer has learned to *hedge* to a degree. However, it also suggests that this hedging is fundamentally different from what LCB does, at least on this specific distribution[2].

Interestingly, the same transformer produces an extremely effective online bandit algorithm when sampling actions instead of taking an argmax. As shown in Figure 1b, DPT matches the performance of classical optimal algorithms, UCB and TS, which are specifically designed for exploration. This is notable because DPT was not explicitly trained to explore, but its emergent strategy is on par with some of the best. In Figure 1c, we show this property is robust to noise in the rewards not seen during pretraining by varying the standard deviation. In Appendix B, we show this generalization happens offline too and even with unseen Bernoulli rewards.

Leveraging structure from suboptimal data. We now investigate whether DPT can learn to leverage the inherent structure of a problem class, even without prior knowledge of this structure and even when learning from in-context datasets that do not explicitly utilize it. More precisely, we consider $\mathcal{T}_{\text{pre}}$ to be a distribution over *linear* bandits, where the reward function is given by $\mathbb{E}\left[r \mid a, \tau\right] = \langle \theta_\tau, \phi(a) \rangle$ and $\theta_\tau \in \mathbb{R}^d$ is a task-specific parameter vector and $\phi : \mathcal{A} \to \mathbb{R}^d$ is fixed feature vector that is the same for all tasks. Given the feature representation ϕ, LinUCB [8], a UCB-style algorithm that leverages ϕ, should achieve regret $\widetilde{\mathcal{O}}(d\sqrt{K})$ over K steps, a substantial gain over UCB and TS when $d \ll |\mathcal{A}|$. Here, we pretrain a DPT model with in-context datasets gathered by TS, which does not leverage the linear structure. Figures 2a and 2b show that DPT can exploit the unknown linear structure, essentially learning a surrogate for ϕ, allowing to do more informed exploration online and decision-making offline. It is nearly on par with LinUCB (which

[1]See Appendix A.2 for additional details such as hyperparameters.

[2]Note our randomly generated environments are equally likely to have expert-biased datasets and adversarial datasets, so LCB is not expected to outperform here [6].



(a) (b) (c)

Figure 2: (a) Offline performance of DPT trained on linear bandits from TS source data. LinReg does linear regression and outputs the greedy action. (b) Online cumulative regret of the same model. The mean and standard error are computed over 200 test tasks. (c) Offline performance on expert-biased datasets. DPT pretrained on a different prior continues to match TS, but DPT-Exp trained from a more representative prior excels.

is given ϕ) and significantly outperforms the dataset source, TS, which does not know or use the structure. These results present evidence that (1) DPT can automatically leverage structure, and (2) supervised learning-based approaches to RL *can* learn novel explorations that transcend the quality of their pretraining data.

Adapting to expert-biased datasets. A common assumption in offline RL is that datasets tend to be a mixture between optimal data (e.g. expert demonstrations) and suboptimal data (e.g. random interactions) [9]. Hence, LCB is generally effective in practice and the pretraining and testing distributions should be biased towards this setting. Motivated by this, we pretrain a second DPT model where $\mathcal{D}_{\text{pre}}$ is generated by mixing the in-context datasets with varying fractions of expert data, biasing $\mathcal{D}_{\text{pre}}$ towards datasets that contain more examples of the optimal action. We denote this model by DPT-Exp. In Figure 2c, we plot the test-time performance of both pretrained models when evaluated on new offline datasets with varying percentages of expert data[3]. Our results suggest that when the pretraining distribution is also biased towards expert-suboptimal data, DPT-Exp behaves similarly to LCB, while DPT continues to resemble TS. This is quite interesting as for other methods, such as TS, it is less clear how to automatically incorporate the right amount of expert bias to yield the same effect, but DPT can leverage this from pretraining.

5 Learning in Markov Decision Processes

5.1 Experimental Setup

Environments.

Comparisons.

5.2 Main Results

Generalizing to new offline datasets and tasks.

Learning from image-based observations.

Stitching novel trajectories from in-context subsequences.

5.3 Learning from Algorithm-Generated Policies and Rollouts

6 Theory

[3]That is, 0% is fully random while 100% has only optimal actions in the in-context dataset.

We now shed light on the observations of the previous empirical results through a theoretical analysis. Our main result shows that DPT (under a slight modification to pretraining) essentially performs in-context posterior sampling (PS). PS is a generalization of Thompson Sampling for RL in MDPs. It maintains and samples from a posterior over tasks τ given historical data D and executes optimal policies $\pi_\tau^\star$ from the pre-trained model. It is provably sample-efficient with online Bayesian regret guarantees, but maintaining posteriors is generally computationally intractable. The ability for DPT to perform PS in-context suggests a path towards computation- and provably sample-efficient RL with priors learned from the data.

6.1 Goal-Conditioned Reinforcement Learning

The basic model we consider is the infinite-horizon Markov decision process (MDP) $\mathcal{M} = (S, A, R, T, \mu_0, \gamma)$ with state space S, action space A, deterministic rewards $r(s, a)$, stochastict ransitions $s' \sim T(s, a)$, initial stated tribution $\mu_0(s)$, and discount factor $\gamma \in (0, 1]$. A policy $\pi : S \to \Delta(A)$ outputs a distribution over actions to use in a given state. In goal-conditioned RL, the MDP additionally assumes a goal space $G := \{\phi(s) \mid s \in S\}$, where the state-to-goal mapping $\phi : S \to G$ is known. Now, the reward function $r(s; g)$ as well as the policy $\pi(a \mid s, g)$ depend on the commanded goal $g \in G$. Given a distribution over desired goals $p(g)$, the objective of goal-conditioned RL is to find a policy π that maximizes the discounted return:

$$J(\pi) := \mathbb{E}_{g\sim p(g),s_0\sim\mu_0,a_t\sim\pi(|s_t,g),s_{t+1}\sim T(|s_t,a_t)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t; g)\right] \tag{3}$$

The goal-conditioned state-action occupancy distribution $d^\pi(s, a; g) : S \times A \times G \to [0, 1]$ of π is

$$d^\pi(s, a; g) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr(s_t = s, a_t = a \mid s_0 \sim \mu_0, a_t \sim \pi(s_t; g), s_{t+1} \sim T(s_t, a_t)) \tag{4}$$

which captures the relative frequency of state-action visitations for a policy π conditioned on goal g. The state-occupancy distribution then marginalizes over actions: $d^\pi(s; g) = \sum_a d^\pi(s, a; g)$. Then, it follows that $\pi(a \mid s, g) = \frac{d^\pi(s,a;g)}{d^\pi(s;g)}$. A state-action occupancy distribution must satisfy the *Bellman flow constraint* in order for it to be an occupancy distribution for some stationary policy π:

$$\sum_a d(s, a; g) = (1-\gamma)\mu_0(s) + \gamma\sum_{\tilde{s},\tilde{a}} T(s \mid \tilde{s}, \tilde{a})d(\tilde{s}, \tilde{a}; g), \qquad \forall s \in S, g \in G \tag{5}$$

We write $d^\pi(s, g) = p(g)d^\pi(s; g)$ as the joint goal-state density induced by $p(g)$ and the policy π. Finally, given d^π, we can express the objective function (**??**) as $J(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{(s,g)\sim d^\pi(s,g)}[r(s; g)]$.

6.2 Offline GCRL

In offline GCRL, the agent cannot interact with the environment $\mathcal{M}$; instead, it is equipped with a static dataset of logged transitions $\mathcal{D} := \{\tau_i\}_{i=1}^N$, where each trajectory $\tau^{(i)} = (s_0^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, ...; g^{(i)})$ with $s_0^{(i)} \sim \mu_0$ and $g^{(i)}$ is the commanded goal of the trajectory. In our case, we represent goal as the discounted trajectory history, i.e. $g_t = s_t + a_t + r_t + \gamma^t g_{t-1}$. We denote the empirical goal-conditioned state-action occupancies of $\mathcal{D}^O$ as $d^O(s, a; g)$. The goal of offline RL is to learn a policy $\hat{\pi}$ using the offline dataset so as to minimize the sub-optimality compared to the optimal policy π^*, i.e., $J(\pi^*) - J(\hat{\pi})$, with high probability.

7 Goal-Conditioned f-Advantage Regression

7.1 Algorithm

We first show that goal-conditioned state-occupancy matching is a mathematically principled approach for solving general GCRL problems, formalizing the teleportation intuition in the introduction.

Proposition 7.1. *Given any $r(s; g)$, for each g in the support of $p(g)$, define $p(s; g) = \frac{e^{r(s;g)}}{Z(g)}$, where $Z(g) := \int e^{r(s;g)}ds$ is the normalizing constant. Then, the following equality holds:*

$$-D_{KL}(d^\pi(s; g)\|p(s; g)) + C = (1-\gamma)J(\pi) + \mathcal{H}(d^\pi(s; g)) \tag{6}$$

*where $J(\pi)$ is the GCRL objective (Eq. (**??**)) with reward $r(s; g)$ and $C := \mathbb{E}_{g\sim p(g)}[\log Z(g)]$.*

This proposition states that, for any choice of reward $r(s;g)$, solving the GCRL problem with a maximum state-entropy regularization is equivalent to optimizing for the goal-conditioned state-occupancy matching objective with target distribution $p(s;g) := \frac{e^{r(s;g)}}{Z(g)}$. However, in the offline setting, we cannot optimize this objective by sampling from $d^\pi(s;g)$ directly. Thus, we first derive an offline lower bound involving an f-divergence regularization term, which subsequently enables solving this optimization problem via its dual using purely offline data:

Proposition 7.2. *Assume for all g in support of* $p(g)$, $\forall s, d^O(s;g) > 0$ *if* $p(s;g) > 0$. *Then, for any* f*-divergence that upper bounds the KL-divergence,*

$$-D_{KL}(d^\pi(s;g)\|p(s;g)) \geq \mathbb{E}_{(s,g)\sim d^\pi(s,g)}\left[\log \frac{p(s;g)}{d^O(s;g)}\right] - D_f(d^\pi(s,a;g)\|d^O(s,a;g)) \tag{7}$$

The RHS of (**??**) can be understood as an f-divergence regularized GCRL objective with reward function $R(s;g) = \log \frac{p(s;g)}{d^O(s;g)}$ (we use capital R to differentiate user-chosen reward R from the environment reward r). Intuitively, this reward encourages visiting states that occur more often in the "expert" state distribution $p(s;g)$ than in the offline dataset, and the f-divergence regularization then ensures that the learned policy is supported by the offline dataset.

In this way, the optimization problem now becomes

$$\begin{aligned} &\max_{d(s,a;g)\geq 0} && \mathbb{E}_{(s,g)\sim d(s,g)}\left[r(s;g)\right] -_f \left(d(s,a;g)\|d^O(s,a;g)\right) \\ &\text{(P)} \qquad \text{s.t.} && \sum_a d(s,a;g) = (1-\gamma)\mu_0(s) + \gamma \sum_{\tilde{s},\tilde{a}} T(s \mid \tilde{s},\tilde{a}) d(\tilde{s},\tilde{a};g), \forall s \in S, g \in G \end{aligned} \tag{8}$$

Further, its dual problem can be reduced to an *unconstrained* minimization problem over the dual variables which serve the role of a value function; importantly, the optimal solution to the dual problem can be used to directly retrieve the optimal primal solution

Proposition 7.3. *The dual problem to* (**??**) *is*

$$\text{(D)} \quad \min_{V(s,g)\geq 0} (1-\gamma)\mathbb{E}_{(s,g)\sim\mu_0,p(g)}[V(s;g)] + \mathbb{E}_{(s,a,g)\sim d^O}\left[f_\star\left(R(s;g) + \gamma\mathcal{T}V(s,a;g) - V(s;g)\right)\right], \tag{9}$$

where $f_\star$ *denotes the convex conjugate function of* f, $V(s;g)$ *is the Lagrangian vector, and* $\mathcal{T}V(s,a;g) = \mathbb{E}_{s'\sim T(\cdot|s,a)}[V(s';g)]$. *Given the optimal* V^**, the primal optimal* d^* *satisfies:*

$$d^*(s,a;g) = d^O(s,a;g) f'_\star\left(R(s;g) + \gamma\mathcal{T}V^*(s,a;g) - V^*(s;g)\right), \forall s \in S, a \in A, g \in G \tag{10}$$

Importantly, since we only use the data from d^O, this objective can be estimated entirely using offline data, making it suitable for offline GCRL.

Then, once we have obtained the optimal (resp. converged) V^*, we could learn the policy via the following supervised regression update:

$$\max_\pi \mathbb{E}_{g\sim p(g)}\mathbb{E}_{(s,a)\sim d^O(s,a;g)}\left[(f'_\star(R(s;g) + \gamma\mathcal{T}V^*(s,a;g) - V^*(s;g))\log\pi(a \mid s,g)\right] \tag{11}$$

7.2 In-Context Learning

Given the in-context dataset $\mathcal{D}$ and a query state s_{query}, we can train a model to predict the optimal action a^* in response via supervised learning. Formally, we aim to train a causal GPT-2 transformer model M parameterized by θ, which outputs a distribution over actions $\mathcal{A}$, to minimize the expected loss over samples from the pretraining distribution:

$$\min_\theta \mathbb{E}_{P_{pre}} \sum_{j\in[n]} l(M_\theta(\cdot|s_{\text{query}}, D), a^*) \tag{12}$$

8 Discussion

Limitations and future work.

Acknowledgments and Disclosure of Funding

We thank

References

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

[2] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.

[3] 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.

[4] 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.

[5] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[6] Chenjun Xiao, Yifan Wu, Jincheng Mei, Bo Dai, Tor Lattimore, Lihong Li, Csaba Szepesvari, and Dale Schuurmans. On the optimality of batch policy optimization algorithms. In *International Conference on Machine Learning*, pages 11362–11371. PMLR, 2021.

[7] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

[8] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

[9] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, 34:11702–11716, 2021.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

[12] Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*, 2022.

[13] Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*, 2022.

[14] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[15] Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

[16] Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, et al. On the effect of pretraining corpora on in-context learning by a large-scale language model. *arXiv preprint arXiv:2204.13509*, 2022.

[17] Yingcong Li, M Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and implicit model selection in in-context learning. *arXiv preprint arXiv:2301.07067*, 2023.

[18] Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. *arXiv preprint arXiv:2303.07895*, 2023.

[19] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

[20] Jacob Abernethy, Alekh Agarwal, Teodor V Marinov, and Manfred K Warmuth. A mechanism for sample-efficient in-context learning for sparse retrieval tasks. *arXiv preprint arXiv:2305.17040*, 2023.

[21] Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022.

[22] 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.

[23] Shipra Agrawal and Navin Goyal. Near-optimal regret bounds for thompson sampling. *Journal of the ACM (JACM)*, 64(5):1–24, 2017.

[24] Malcolm Strens. A bayesian framework for reinforcement learning. In *ICML*, volume 2000, pages 943–950, 2000.

[25] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26, 2013.

[26] Shipra Agrawal and Randy Jia. Optimistic posterior sampling for reinforcement learning: worst-case regret bounds. *Advances in Neural Information Processing Systems*, 30, 2017.

[27] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[28] Xiuyuan Lu and Benjamin Van Roy. Ensemble sampling. *Advances in neural information processing systems*, 30, 2017.

[29] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[30] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[32] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[33] 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.

[34] 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.

[35] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1):12348–12355, 2021.

[36] 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. In *International Conference on Learning Representation (ICLR)*, 2020.

[37] Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in neural information processing systems*, 33:20095–20107, 2020.

Additional Related Work

In-context learning. Beyond decision-making and reinforcement learning, our approach takes inspiration from general in-context learning, a phenomenon observed most prominently in large language models in which large-scale autoregressive modelling can surprisingly lead to a model that exhibits meta-learning capabilities [10]. Recently, there has been great interest in understanding the capabilities and properties of in-context learning [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]. While a common hypothesis suggests that this phenomenon is due to properties of the data used to train large language models [21], our work suggests that this phenomenon can also be encouraged in general settings via adjustments to the pre-training objective. In fact, DPT could be interpreted as explicitly encouraging the ability to perform Bayesian inference, which is a popular explanation for the mechanism behind in-context learning for large language models [22].

Posterior Sampling. Posterior sampling originates from the seminal work of [4], and has been popularized and thoroughly investigated in recent years by a number of authors [3, 23, 24, 25, 26, 27]. For bandits, it is often referred to as Thompson Sampling, but the framework is easily generalizable to RL. The principle is as follows: begin with a prior over possible models (i.e. reward and transition functions), and maintain a posterior distribution over models by updating as new interactions are made. At decision-time, sample a model from the posterior and execute its optimal policy. The aforementioned prior works have developed strong theoretical guarantees on Bayesian and frequentist regret for posterior sampling. Despite its desirable theoretical characteristics, a major limitation is that computing the posterior is often computationally intractable, leading practitioners to rely on approximation-based solutions [28, 29, 30]. In Section **??**, we show that a version of the DPT model learned from pretraining can be viewed as implementing posterior sampling as it should be without resorting to approximations or deriving complicated posterior updates. Instead, the posterior update is implicitly learned through pretraining to predict the optimal action. This suggests that in-context learning (or meta-learning more generally) could be a key in unlocking practically applicable posterior sampling for RL.

A Implementation and Experiment Details

Algorithm 1 Decision-Pretrained Transformer (detailed)

```
1: // Collecting pretraining dataset
2: Initialize empty dataset $\mathcal{B}$
3: for $i$ in $[N]$ do
4:     Sample task $\tau \sim \mathcal{T}_{\text{pre}}$
5:     Sample interaction dataset $D \sim \mathcal{D}_{\text{pre}}(\cdot; \tau)$ of length $n$
6:     Sample $s_{\text{query}} \sim \mathcal{D}_{\text{query}}$ and $a^\star \sim \pi^\star_\tau(\cdot | s_{\text{query}})$
7:     Add $(s_{\text{query}}, D, a^\star)$ to $\mathcal{B}$
8: end for
9: // Training model on dataset
10: Initialize model $M_\theta$ with parameters $\theta$
11: while not converged do
12:     Sample $(s_{\text{query}}, D, a^\star)$ from $\mathcal{B}$
13:     Predict $\hat{p}_j(\cdot) = M_\theta(\cdot | s_{\text{query}}, D_j)$ for all $j \in [n]$.
14:     Compute loss in (5) with respect to $a^\star$ and backpropagate to update $\theta$.
15: end while
```

A.1 DPT Architecture: Formal Description

In this section, we provide a detailed description of the architecture alluded to in Section 3 and Figure **??**. See hyperparameter details for models in their respective sections. The model is implemented in Python with PyTorch [31]. The backbone of the transformer architecture we use is an autoregressive GPT-2 model from the HuggingFace `transformers` library.

For the sake of exposition, we suppose that $\mathcal{S}$ and $\mathcal{A}$ are subsets of $\mathbb{R}^{d_S}$ and $\mathbb{R}^{d_A}$ respectively. We handle discrete state and action spaces with one-hot encoding. Consider a single training datapoint derived from an (potentially unknown) task τ: we have a dataset D of interactions within τ, a query

Algorithm 2 Offline test-time deployment (detailed)

```
1: // Task and offline dataset are generated without learner's control
2: Sample unknown task $\tau \sim \mathcal{T}_{\text{test}}$
3: Sample dataset $D \sim \mathcal{D}_{\text{test}}(\cdot; \tau)$
4: // Deploying offline policy $M_\theta(\cdot|\cdot, D)$
5: $s_1 = \texttt{reset}(\tau)$
6: for $h$ in $[H]$ do
7:     $a_h = \text{argmax}_{a \in \mathcal{A}} M_\theta(\cdot|s_h, D)$ // Most likely action
8:     $s_{h+1}, r_h = \texttt{step}(\tau, a_h)$
9: end for
```

Algorithm 3 Online test-time deployment (detailed)

```
1: // Online, dataset is empty as learning is from scratch
2: Initialize $D = \{\}$
3: Sample unknown task $\tau \sim \mathcal{T}_{\text{test}}$
4: for ep in max_eps do
5:     $s_1 = \texttt{reset}(\tau)$
6:     for $h$ in $[H]$ do
7:         $a_h \sim M_\theta(\cdot|s_h, D)$ // Sample action from predicted distribution
8:         $s_{h+1}, r_h = \texttt{step}(\tau, a_h)$
9:     end for
10:    // Experience from previous episode added to dataset
11:    Add $(s_1, a_1, r_1, \ldots)$ to $D$
12: end for
```

state s_{query}, and its corresponding optimal action $a^\star = \pi^\star_\tau(s_{\text{query}})$. We construct the embeddings to be passed to the GPT-2 backbone in the following way. From the dataset $D = \{(s_j, a_j, s'_j, r_j)\}_{j \in [n]}$, we construct vectors $\xi_j = (s_j, a_j, s'_j, r_j)$ by stacking the elements of the transition tuple into dimension $d_\xi := 2d_S + d_A + 1$ for each j in the sequence. This sequence of n elements is concatenated with another vector $v := (s_{\text{query}}, \mathbf{0})$ where the $\mathbf{0}$ vector is a vector of zeros of sufficient length to make the entire element dimension d_ξ. The $(n+1)$-length sequence is given by $X = (v, \xi_1, \ldots, \xi_n)$. As order does not often matter for the dataset D[4], we do not use positional encoding in order to take advantage of this invariance. We first apply a linear layer $\texttt{Linear}(X)$ and pass the result to the transformer, which outputs the sequence $Y = (\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_n)$. In the continuous action case, these can be used as is for predictions of $a^\star$. For the discrete action case, we use them as logits to be converted to either a distribution over actions in $\mathcal{A}$ or one-hot vector predictions of $a^\star$. Here, we compute action probabilities

$$\hat{p}_j = \texttt{softmax}(\hat{y}_j) \in \Delta(\mathcal{A}) \tag{13}$$

Because of the GPT-2 causal architecture (we defer details to the original papers [32, 10]), we note that $\hat{p}_j$ depends only on s_{query} and the partial dataset $D_j = \{(s_k, a_k, s'_k, r_k)\}_{k \in [j]}$, which is why we write the model notation,

$$M_\theta(\cdot|s_{\text{query}}, D_j) = \hat{p}_j(\cdot), \tag{14}$$

to denote that the predicted probabilities of the jth element only depend on D_j and not the entire D for the model M with parameters $\theta \in \Theta$. For example, with $j = 0$, the prediction of $a^\star$ is made without any contextual information about the task τ except for s_{query}, which can be interpreted as the prior over $a^\star$. We measure loss of this training example via the cross entropy for each $j \in [n]$:

$$-\sum_{j \in [n]} \log \hat{p}_j(a^\star) \tag{15}$$

Intuition. Elements of the inputs sequence X represent transitions in the environment. When passed through the GPT-2 transformer, the model learns to associate elements of the sequence via the standard query-key-value mechanism of the attention model. The query state s_{query} is demarcated

[4]This is not always true such as when data comes from an algorithm such as PPO or Thompson Sampling.

by its zeros vector (which also acts as padding). Unlike other examples of transformers used for decision-making such as the Decision Transformer [33] and Algorithm Distillation [34], DPT does not separate the individual (s, a, s', r) into their own embeddings to be made into one long sequence. This is because we view the transition tuples in the dataset as their own singletons, to be related with other singletons in the dataset through the attention mechanism. We note that there are various other implementation variations one could take, but we found success and robustness with this one.

A.2 Implementation Details

A.2.1 Bandit algorithms

First, we describe the comparisons from the bandit experiments with hyperparameters.

Empirical Mean (Emp). Emp has no hyperparameters, but we give it some mechanism to avoid degenerate scenarios. In the offline setting, Emp will only choose from actions that have at least one example in the dataset. This gives Emp and LCB-style effect when actions are missing. Similarly, online, Emp will sample each action at least once before defaulting to its real strategy. These changes only improve Emp.

Upper Confidence Bound (UCB). According to the Hoeffding bound, we choose actions as $\hat{a} \in \operatorname{argmax}_{a \in \mathcal{A}} \left\{ \hat{\mu}_a + \sqrt{1/n_a} \right\}$ where $\hat{\mu}_a$ is the empirical mean so far for action a and n_a is the number of times a has been chosen so far. To arrive at this constant for the bonus, we coarsely tried a set of plausible values given the noise and found this to perform the best.

Lower Confidence Bound (LCB). We choose actions as $\hat{a} \in \operatorname{argmax}_{a \in \mathcal{A}} \left\{ \hat{\mu}_a - \sqrt{1/n_a} \right\}$ where $\hat{\mu}_a$ is the empirical mean so far for action a and n_a is the number of times a has been chosen so far.

Thompson Sampling (TS). Since the means are sampled uniformly from $[0, 1]$, Gaussian TS is partially misspecified; however, we set prior mean and variance to $\frac{1}{2}$ and $\frac{1}{12}$ to match the true ones. The noise model was well-specified with the correct variance. In the linear experiments of Figure 2a and Figure 2b, we set the prior mean and variance to 0 and 1 to fit the true ones better.

LinUCB. We choose $\hat{a}_t \in \operatorname{argmax}_{a \in \mathcal{A}} \langle \hat{\theta}_t, \phi(a) \rangle + \beta \|\phi(a)\|_{\hat{\Sigma}_t^{-1}}$ where $\beta = 1$ and $\hat{\Sigma}_t = I + \sum_{s \in [t-1]} \phi(a_s)\phi(a_s)^\top$ and $\hat{\theta}_t = \hat{\Sigma}_t^{-1} \sum_{s \in [t-1]} r_s \phi(a_s)$. Here, r_s and a_s are the reward and action observed at time s.

LinReg. LinReg (offline) is the same as LinUCB except we set $\beta = 0$ to greedily choose actions.

DPT. The transformer for DPT has an embedding size of 32, context length of 500 for basic bandits and 200 for linear bandits, 4 hidden layers, and 4 attention heads per attention layer for all bandits. We use the AdamW optimizer with weight decay 1e-4, learning rate 1e-4, and batch-size 64. For all experiments, we shuffle the in-context dataset D since order does not matter except in the linear bandit.

A.2.2 RL Algorithms

Below, we describe the comparisons from the MDP experiments and their hyperparameters.

Proximal Policy Optimization (PPO). The reported results for PPO use the Stable Baselines3 implementation [35] with the default hyperparameters, which successfully learns each task given 100K environment steps in Dark Room and 125K environment steps in Miniworld. In Dark Room, the policy is implemented as a multi-layer perceptron with two hidden layers of 64 units each. In Miniworld, the policy is a convolutional neural network with two convolutional layers with 16 3×3 kernels each, followed by a linear layer with output dimension of 8.

Algorithm Distillation (AD). We first collect learning histories with PPO for each of the training tasks. Then, given a cross-episodic context of length H, where H is the task horizon, the model is trained to predict the actions taken K episodes later (given the states visited in that episode). This was shown to lead to faster algorithms in [34]. We evaluated AD across different values of K. Between $K = 10, 50, 100$, we found $K = 100$ to be most performant in the Dark Room environment. In Miniworld, we also subsampled with $K = 100$. In Dark Room, the transformer has similar hyperparameters as DPT: an embedding size of 32, context length of 100 steps, 4 hidden layers, and 4 attention heads per attention layer. In Miniworld, as with DPT, we first encode the image with a convolutional network with two convolutional layers with 16 3×3 kernels each, followed by a linear layer with output dimension of 8.

RL^2. The reported results for RL^2 use an open-sourced implementation from [36]. The implementation uses PPO as the RL algorithm and defines a single trial as four consecutive episodes. The policy is implemented with one hidden layer of 32 units in Dark Room. In Miniworld, the policy is parameterized with a convolutional neural network with two convolutional layers with 16 3×3 kernels each, followed by a linear layer with output dimension of 8.

DPT. The transformer for DPT has an embedding size of 32, context length of 100 steps, 4 hidden layers, and 4 attention heads per attention layer in Dark Room. In Miniworld, the image is first passed through a convolutional network with two convolutional layers 16 3×3 kernels each, followed by a linear layer with output dimension of 8. The transformer model that processes these image embeddings otherwise has the same hyperparameters as in Dark Room. We use the AdamW optimizer with weight decay 1e-4, learning rate 1e-3, and batch-size 128.

A.3 Bandit Pretraining and Testing

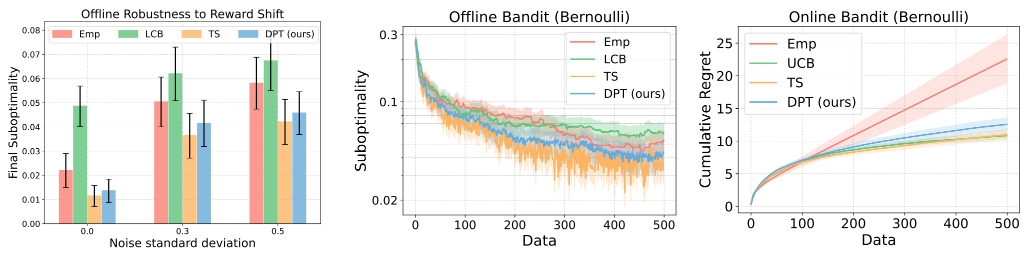
Basic Bandit. Offline, to generate the in-context datasets for pretraining, we used a Dirichlet distribution to sample action frequencies in order to generate datasets with diverse compositions (i.e. some more uniform, some that only choose a few actions, etc.): $p_1 \sim \text{Dir}(\mathbb{1})$ where $p_1 \in \Delta(\mathcal{A})$ and $\mathbb{1} \in \mathbb{R}^{|\mathcal{A}|}$. We also mixed this with a distribution that has all mass on one action: $\hat{a} \sim \text{Unif}(\mathcal{A})$ and $p_2(\hat{a}) = 1$ and $p_2(a) = 0$ for all $a \neq \hat{a}$. The final action distribution is $p = (1 - \omega)p_1 + \omega p_2$ where $\omega \sim \text{Unif}(0.1[10])$. We train on 100,000 pretraining samples for 300 epochs with an 80/20 train/validation split. In Figure 1a, $\mathcal{D}_{\text{test}}$ is generated in the same way.

Expert-Biased Bandit. To generate expert-biased datasets for pretraining, we compute the action frequencies to bias the dataset towards the optimal action. Let $a^\star$ be the optimal one. As before, we take $p_1 \sim \text{Dir}(\mathbb{1})$. Then, $p_2(a^\star) = 1$ and $p_2(a) = 0$ for all $a \neq a^\star$. For of bias of ω, we take $p = (1 - \omega)p_1 + \omega p_2$ with $\omega \sim \text{Unif}(0.1[10])$. We use the same pretraining sample size and epochs as before. For testing, $\mathcal{D}_{\text{test}}$ is generated the same way except we fix a particular $\omega \in \{0, 0.5, 1\}$ to test on.

Linear Bandit. We consider the case where $|\mathcal{A}| = 10$ and $d = 2$. To generate environments from $\mathcal{T}_{\text{pre}}$, we first sampled a fixed set of actions from $\mathcal{N}(\mathbf{0}, I_d/d)$ in $\mathbb{R}^d$ to represent the features. Then, for each τ, we sampled $\theta_\tau \sim \mathcal{N}(\mathbf{0}, I_d/d)$ to produce the means $\mu_a = \langle \theta_\tau, \phi(a) \rangle$ for $a \in \mathcal{A}$. To generate the in-context dataset, we ran Gaussian TS (which does not leverage ϕ) over $n = 200$ steps (see hyperparameters in previous section). Because order matters, we did not shuffle and used $1,000,000$ pretraining samples over 200 epochs with an 80/20 train/validation split. At test time, we set $\mathcal{T}_{\text{test}} = \mathcal{T}_{\text{pre}}$ and $\mathcal{D}_{\text{test}} = \mathcal{D}_{\text{pre}}$. Note that ϕ is fixed over all τ, as is standard for a linear bandit.

A.4 MDP Environment Details

Dark Room. The agent must navigate a 10×10 grid to find the goal within $H = 100$ steps. The agent's observation is its xy-position, the allowed actions are left, right, up, down, and stay, and the reward is only $r = 1$ when the agent is at the goal, and $r = 0$ otherwise. At test time, the agent begins at the $(0, 0)$ position. We randomly designate 80 of the 100 grid squares to be goals for the training tasks, and hold out the remaining 20 for evaluation.



(a) (b) (c)

Figure 3: (a) Final (after 500 steps) offline suboptimality on out-of-distribution bandits with different Gaussian noise standard deviations. (b) Offline performance on out-of-distribution Bernoulli bandits, given random in-context datasets. (c) Online cumulative regret on Bernoulli bandits. The mean and standard error are computed over 200 test tasks.

Miniworld. The agent must navigate to the correct box, which is initially unknown, from 25×25 RGB image observations. The agent is additionally conditioned on its own direction vector. In each episode, the environment is initialized with four boxes of different colors, one in each corner of the square room. The agent can turn left, turn right, or move forward. The reward is only $r = 1$ when the agent is near the correct box and $r = 0$ otherwise, and each episode is 50 time-steps long. At test time, the agent begins in the middle of the room.

A.5 MDP Pretraining Datasets

Dark Room. In Dark Room, we collect 100K in-context datasets, each of length $H = 100$ steps, with a uniform-random policy. The 100K datasets are evenly collected across the 100 goals. The query states are uniformly sampled from the state space, and the optimal actions are computed as follows: move up/down until the agent is on the same y-position as the goal, then move left/right until the agent is on the x-position as the goal. Of the 100K collections of datasets, query states, and optimal actions, we use the first 80K (corresponding to the first 80 goals) for training and the remaining 20K for validation.

Miniworld. While this task is solved from image-based observations, we also note that there are only four distinct tasks (one for each colored box), and the agent does not need to handle new tasks at test time. Hence, the number of in-context datasets required in pretraining is fewer – we use 40K datasets each of length $H = 50$ steps. So as to reduce computation, the in-context datasets only have only (s, a, r) tuples. The query states, which consist of image and direction are sampled uniformly from the entire state space, i.e., the agent is place uniformly at random in the environment, pointing in a random direction. The optimal actions are computed as follows: turn towards the correct box if the agent is not yet facing it (within ± 15 degrees), otherwise move forward. Of the 40K collections of datasets, query states, and optimal actions, we use 32K for training and the remaining 8K for validation.

B Additional Experimental Results

B.1 Bandits

This section reports additional experimental results in bandit environments.

Out-of-distribution reward variances. In Figures 1c and 3a, we demonstrate the robustness of the basic pretrained model under shifts in the reward distribution at test time by varying the amount of noise observed in the rewards. DPT maintains robustness to these shifts similar to TS.

Bernoulli rewards. We test the out-of-distribution ability of DPT further by completely changing the reward distribution from Gaussian to Bernoulli bandits. Despite being trained only on Gaussian tasks during pretraining, DPT maintains strong performance both offline and online in Figures 3b and 3c.

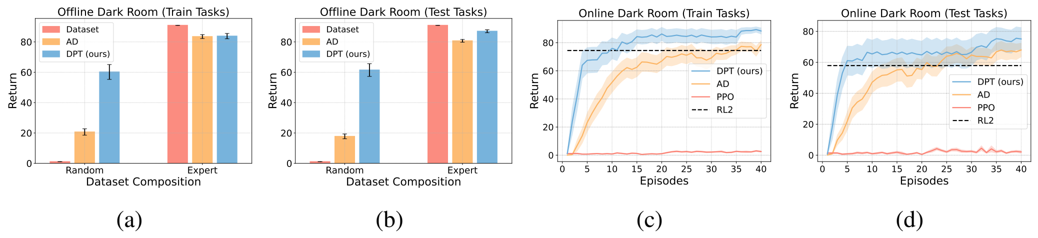


Figure 5: All comparisons in Dark Room evaluated on the tasks that were seen during pretraining, displayed next to their evaluations on test task counterparts from the main text.

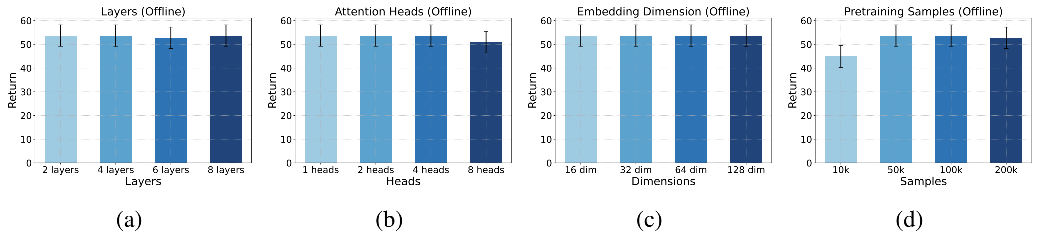


Figure 6: Sensitivity analysis of the offline Dark Rook task over the GPT-2 transformer's hyperparameters: (a) layers (b) attention heads (c) embedding dimensions (d) pretraining samples.

B.2 Markov Decision Processes

This section reports additional experimental results in the Dark Room and Miniworld environments.

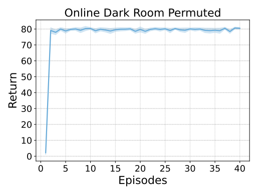


Figure 4: Online evaluation of DPT on Dark Room when tested on novel actions set permutations.

Performance on training tasks. In Fig. 5, we show the performance of each method on the training tasks in Dark Room. Offline, DPT and AD demonstrate comparable performance as on the training tasks, indicating a minimal generalization gap to new goals. Online, DPT, AD, and RL^2 also achieve performance on the training tasks similar to that on the test tasks.

Generalization to new dynamics. In this experiment, we study generalization to variations in a different aspect of the MDP, namely the dynamics. We design *Dark Room (Permuted)*, a variant of Dark Room in which the goal is fixed to a corner but the action space is randomly permuted. Hence, the agent must leverage its historical context to infer the effect of each action. On a held-out set of 20 permutations, DPT infers the optimal policy correctly every time offline, given only 100 offline samples, matching the optimal policy at 83 return. Similarly, the online performance immediately snaps to a near optimal policy in one episode once it identifies the novel permutation in Figure 4.

B.3 Sensitivity Analysis

We next seek to understand the sensitivity of DPT to different hyperparameters, including the model size and size of the pretraining dataset. These experiments are performed in the Dark Room environment. As shown in Fig. 6, the performance of DPT is robust to the model size; it is the same across different embedding sizes, number of layers, and number of attention heads. Notably, the performance is slightly worse with 8 attention heads, which may be attributed to slight overfitting. We do see that when the pretraining dataset is reduced to 10% of its original size (10000 samples) the performance degrades, but otherwise has similar performance with larger pretraining datasets.

C Additional Theory and Omitted Proofs

We start with a well-known concentration inequality for the maximum-likelihood estimate (MLE) to provide some more justification for the approximation made in Assumption **??**. We state a version from [37]. Let $\mathcal{F}$ be a finite function class used to model a conditional distribution $p_{Y|X}(y|x)$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Assume there is $f^\star \in \mathcal{F}$ such that $p(y|x) = f^\star(y|x)$ (realizable), and $f(\cdot|x) \in \Delta(\mathcal{Y})$ for all $x \in \mathcal{X}$ and $f \in \mathcal{F}$ (proper). Let $D = \{x_i, y_i\}_{i\in[N]}$ denote a dataset of i.i.d samples where $x_i \sim p_X$ and $y_i \sim p_{Y|X}(\cdot|x_i)$. Let

$$\hat{f} = \operatorname*{argmax}_{f\in\mathcal{F}} \sum_{i\in[N]} \log f(y_i|x_i) \tag{16}$$

Proposition C.1 (Theorem 21 of [37])**.** *Let D and $\hat{f}$ be given as above under the aforementioned conditions. Then, with probability at least* $1-\delta$*,*

$$\mathbb{E}_{x\sim p_X} \|\hat{f}(\cdot|x) - p_{Y|X}(\cdot|x)\|_1^2 \leq \frac{8\log\left(|\mathcal{F}|/\delta\right)}{N} \tag{17}$$

The finiteness of $\mathcal{F}$ is done for simplicity, but we can see that this yields dependence on the log-cardinality, a common measure of complexity. Extensions to infinite $\mathcal{F}$ of bounded statistical complexity can be readily made to replace this. For our setting, the bound suggests that $\mathbb{E}_{P_{pre}} \|P_{pre}(\cdot|s_{\text{query}}, D, \xi_h) - M_\theta(\cdot|s_{\text{query}}, D, \xi_h)\|_1^2 \to 0$ as $N \to \infty$ with high probability, provided the function class of M_θ has bounded statistical complexity.

C.1 Posterior Sampling

Posterior sampling is most generally described with the following procedure [25]. Initialize a prior distribution $\mathcal{T}_1 = \mathcal{T}_{\text{pre}}$ and dataset $D = \{\}$. For $k \in [K]$

1. Sample $\tau_k \sim \mathcal{T}_k$ and compute $\hat{\pi}_{\tau_k}$
2. Execute $\pi^\star_{\tau_k}$ and add interactions to D
3. Update posterior distribution $\mathcal{T}_{k+1}(\tau) = P(\tau|D)$.

The prior and posteriors are typically over models such as reward functions in bandits or transition dynamics in MDPs.

C.2 Proof of Theorem ??

*

Proof. Without loss of generality, for a task τ, we take $\pi^\star_\tau(\cdot|s)$ to be deterministic and denote the optimal action in state s as $\pi^\star_\tau(s)$. Recall that we consider a fixed current task τ_c and a fixed in-context dataset D. Define $\xi_h = (s_1, a_1, \ldots, s_h, a_h)$.

We now formally state the variant of the full joint distribution from which we sample during pretraining. Let τ and D' be an arbitrary task and dataset and let $a^\star \in \mathcal{A}$, $s_{\text{query}} \in \mathcal{S}$, $\xi_{H-1} \in (\mathcal{S} \times \mathcal{A})^{H-1}$, and $h \in [0, H-1]$ be arbitrary.

$$P_{pre}(\tau, a^\star, s_{\text{query}}, D', \xi_{H-1}, h) = \mathcal{T}_{\text{pre}}(\tau)\mathcal{D}_{\text{pre}}(D'; \tau)\mathcal{D}_{\text{query}}(s_{\text{query}})\mathfrak{S}_H(s_{1:H})\pi^\star_\tau(a^\star|s_{\text{query}}) \tag{18}$$

$$\times \text{Unif}[0, H-1] \prod_{i\in[H]} \pi^\star_\tau(a_i|s_i) \tag{19}$$

The Unif$[0, H-1]$ is due to the fact that we sample $h \sim$ Unif$[0, H-1]$ and then truncate ξ_h from ξ_{H-1} (or, equivalently, sample $\xi_h \sim \mathfrak{S}_h$ directly), marginalizing out the other variables. For $h' \leq h-1$, recall that we also use the notation $\mathfrak{S}_{h'}(s_{1:h'})$ to denote the marginalization of the full joint $\mathfrak{S}_H$. We will eventually work with the posterior of this distribution given the data D and history ξ_h:

$$P_{pre}(\tau|D, \xi_h) \propto \mathcal{T}_{\text{pre}}(\tau)\mathcal{D}_{\text{pre}}(D; \tau) \prod_{i\in[h]} \pi^\star_\tau(a_i|s_i) \tag{20}$$

$$\propto P_{pre}(\tau|D) \prod_{i\in[h]} \pi^\star_\tau(a_i|s_i) \tag{21}$$

We define the following random sequences and subsequences:

$$\Xi_{ps}(h; D) = (S_1^{ps}, A_1^{ps}, \dots, S_h^{ps}, A_h^{ps}) \tag{22}$$

where the variables are generated according to the following conditional process: $\tau_{ps} \sim P(\cdot|D)$, $S_1^{ps} \sim \rho_{\tau_c}$, $A_h^{ps} \sim \pi^\star_{\tau_{ps}}(\cdot|S_h^{ps})$, and $S_{h+1}^{ps} \sim T_{\tau_c}(\cdot|S_h^{ps}, A_h^{ps})$. We also define $\Xi_{ps}(h' : h; D)$ to be the last $h - h'$ elements of $\Xi_{ps}(h; D)$. Analogously, we define

$$\Xi_{pre}(h; D) = (S_1^{pre}, A_1^{pre}, \dots, S_h^{pre}, A_h^{pre}) \tag{23}$$

where the variables are from the process: $S_1^{pre} \sim \rho_{\tau_c}$, $A_h^{pre} \sim P_{pre}(\cdot|S_h^{pre}, D, \Xi_{pre}(h-1; D))$, and $S_{h+1}^{pre} \sim T_{\tau_c}(\cdot|S_h^{pre}, A_h^{pre})$. Note that A_h^{pre} is sampled conditioned on the sequence $\Xi_{pre}(h; D)$ so far.

We will show that $\Xi_{ps}(h; D)$ and $\Xi_{pre}(h; D)$ follow the same distribution for all $h \in [H]$. For convenience, we will drop notational dependence on D, except where it resolves ambiguity. Also, because of Assumption **??**, we have that $P_{pre}(\cdot|S_h^{pre}, D, \Xi_{pre}(h-1)) = M_\theta(\cdot|S_h^{pre}, D, \Xi_{pre}(h-1))$, so we will just work with P_{pre} for the remainder of the proof. We will also make use of the following lemma.

Lemma C.2. *If $\mathcal{D}_{pre}$ is complaint, then $P_{pre}(\tau|D) = P(\tau_{ps} = \tau|D)$.*

Proof. From the definition of posterior sampling (using the same prior, $\mathcal{T}_{\text{pre}}$), we have that

$$\begin{aligned}
P(\tau_{ps} = \tau|D) &\propto P(D|\tau)\mathcal{T}_{\text{pre}}(\tau) && (24)\\
&\propto \mathcal{T}_{\text{pre}}(\tau) \prod_{j\in[n]} T_\tau(s_j'|s_j, a_j) R_\tau(r_j|s_j, a_j) && (25)\\
&\propto \mathcal{T}_{\text{pre}}(\tau) \prod_{j\in[n]} T_\tau(s_j'|s_j, a_j) R_\tau(r_j|s_j, a_j) \mathcal{D}_{\text{pre}}(a_j|s_j, D_{j-1}) && (26)\\
&= \mathcal{T}_{\text{pre}}(\tau)\mathcal{D}_{\text{pre}}(D; \tau) && (27)\\
&= P_{pre}(\tau|D) && (28)
\end{aligned}$$

where the second line crucially uses the fact that posterior sampling chooses actions based only on the prior and history so far. Similarly, the third line uses the fact that $\mathcal{D}_{\text{pre}}$ is compliant. Since the two sides are proportional in τ, they are equivalent. □

We will prove Theorem **??** via induction for each $h \in [H]$. First, consider the base case for a sequence of length $h = 1$. Recall that ρ_{τ_c} denotes the initial state distribution of τ_c. We have that the densities can be written as

$$\begin{aligned}
P(\Xi_{ps}(1) = \xi_1) &= P(S_1^{ps} = s_1, A_1^{ps} = a_1) && (29)\\
&= \rho_{\tau_c}(s_1) P(A_1^{ps} = a_1 | S_1^{ps} = s_1) && (30)\\
&= \rho_{\tau_c}(s_1) \int_\tau P(A_1^{ps} = a_1, \tau_{ps} = \tau | S_1^{ps} = s_1) d\tau && (31)\\
&= \rho_{\tau_c}(s_1) \int_\tau \pi^\star_\tau(a_1|s_1) P_{ps}(\tau_{ps} = \tau | D, S_1^{ps} = s_1) d\tau && (32)\\
&= \rho_{\tau_c}(s_1) \int_\tau \pi^\star_\tau(a_1|s_1) P_{ps}(\tau_{ps} = \tau | D) d\tau && (33)\\
&= \rho_{\tau_c}(s_1) P_{pre}(A_1^{pre} = a_1 | s_1, D) && (34)\\
&= P(\Xi_{pre}(1) = \xi_1) && (35)
\end{aligned}$$

where the second line uses the sampling process of S_1^{pre}; the third marginalizes over τ_{ps}, which is the task that posterior sampling samples to find the optimal policy; the fourth decomposes this into the optimal policy and the posterior over τ_{ps} given D and S_1^{ps}. Since S_1^{ps} is independent of sampling of τ_{ps} this dependence goes away in the next line. The sixth line applies Lemma C.2 and then, for $h = 1$, there is no history to condition on.

Now, we leverage the inductive hypothesis to prove the full statement. Suppose that the hypothesis holds for $h-1$. Then,

$$P(\Xi_{ps}(h)=\xi_h) = P(\Xi_{ps}(h-1)=\xi_{h-1})P(S_h^{ps}=s_h, A_h^{ps}=a_h|\Xi_{ps}(h-1)=\xi_{h-1}) \quad (36)$$
$$(37)$$

By the hypothesis, we have that $P(\Xi_{ps}(h-1)=\xi_{h-1}) = P(\Xi_{pre}(h-1)=\xi_{h-1})$. For the second factor,

$$P(S_h^{ps}=s_h, A_h^{ps}=a_h|\Xi_{ps}(h-1)=\xi_{h-1}) \quad (38)$$
$$= T_{\tau_c}(s_h|s_{h-1},a_{h-1}) \cdot P(A_h^{ps}=a_h|S_h^{ps}=s_h, \Xi_{ps}(h-1)=\xi_{h-1}) \quad (39)$$
$$= T_{\tau_c}(s_h|s_{h-1},a_{h-1}) \cdot \int_\tau P(A_h^{ps}=a_h, \tau_{ps}=\tau|S_h^{ps}=s_h, \Xi_{ps}(h-1)=\xi_{h-1})d\tau \quad (40)$$

As before, we can further rewrite the last factor as

$$P(A_h^{ps}=a_h, \tau_{ps}=\tau|S_h^{ps}=s_h, \Xi_{ps}(h-1)=\xi_{h-1}) \quad (41)$$
$$= \pi_\tau^\star(a_h|s_h) \cdot P(\tau_{ps}=\tau|S_h^{ps}=s_h, \Xi_{ps}(h-1)=\xi_{h-1}) \quad (42)$$

where

$$P(\tau_{ps}=\tau|S_h^{ps}=s_h, \Xi_{ps}(h-1)=\xi_{h-1}) = \frac{P(S_h^{ps}=s_h, \Xi_{ps}(h-1)=\xi_{h-1}|\tau_{ps}=\tau)P(\tau_{ps}=\tau|D)}{P(S_h^{ps}=s_h, \Xi_{ps}(h-1)=\xi_{h-1})} \quad (43)$$
$$\propto P_{pre}(\tau|D) \prod_{i\in[h-1]} T_{\tau_c}(s_{i+1}|s_i,a_i)\pi_\tau^\star(a_i|s_i) \quad (44)$$
$$\propto P_{pre}(\tau|D) \prod_{i\in[h-1]} \pi_\tau^\star(a_i|s_i) \quad (45)$$
$$\propto P_{pre}(\tau|D)\mathcal{D}_{\text{query}}(s_h)\mathfrak{S}_{h-1}(s_{1:h-1}) \prod_{i\in[h-1]} \pi_\tau^\star(a_i|s_i) \quad (46)$$
$$\propto P_{pre}(\tau|s_h, D, \xi_{h-1}) \quad (47)$$
$$(48)$$

where $\propto$ denotes that the two sides are equal up to multiplicative factors independent of τ. In the first line, we used Bayes rule. In the second line, given that $\tau_{ps}=\tau$ (i.e. posterior sampling selected τ to deploy), we decompose the probability of observing that sequence of states of actions. We also used Lemma C.2. The denominator does not depend on τ. Similarly, for the third and fourth lines, T_{τ_c} and $\mathfrak{S}$ do not depend on τ. The final line follows from the definition of the joint pretraining distribution in this regime.

Therefore, we conclude that the posterior over the value of τ_{ps} is the same as the posterior over the task in the pretraining distribution, given s_h, D, ξ_{h-1}. Substituting back through all the previous equations, we have

$$P(\Xi_{ps}(h)=\xi_h) \quad (49)$$
$$= P(\Xi_{pre}(h-1)=\xi_{h-1}) \cdot T_{\tau_c}(s_h|s_{h-1},a_{h-1}) \int_\tau \pi_\tau^\star(a_h|s_h)P_{pre}(\tau|s_h,D,\xi_{h-1})d\tau \quad (50)$$
$$= P(\Xi_{pre}(h-1)=\xi_{h-1}) \cdot T_{\tau_c}(s_h|s_{h-1},a_{h-1})P_{pre}(a_h|s_h,D,\xi_{h-1}) \quad (51)$$
$$= P(\Xi_{pre}(h)=\xi_h) \quad (52)$$

This concludes the proof.

□

C.3 Proof of Corollary ??

*

Proof. Note that $\mathcal{D}_{\text{pre}}$ is clearly compliant since it is generated by random sampling. We use the equivalence between M_θ and posterior sampling established in Theorem **??**. The proof then follows immediately from Theorem 1 of [25] to guarantee that

$$\mathbb{E}_{\mathcal{T}_{\text{pre}}}\left[\text{Reg}_\tau(M_\theta)\right] \le \widetilde{\mathcal{O}}\left(H^{3/2}S\sqrt{AK}\right) \tag{53}$$

where the notation $\widetilde{\mathcal{O}}$ omits polylogarithmic dependence. The bound on the test task distribution follows from the assumed bound on the likelihood ratio under the priors:

$$\int \mathcal{T}_{\text{test}}(\tau)\text{Reg}_\tau(M_\theta)d\tau \le \mathcal{C}\int \mathcal{T}_{\text{pre}}(\tau)\text{Reg}_\tau(M_\theta)d\tau. \tag{54}$$

□

C.4 Proof of Corollary ??

*

Proof. The distribution $\mathcal{D}_{\text{pre}}$ satisfies compliance by definition because it is generated by an adaptive algorithm TS. The proof once again follows by immediately deferring to the established result of [27] (Proposition 3) for linear bandits by the posterior sampling equivalence of Theorem **??**. This ensures that posterior sampling achieves regret $\widetilde{\mathcal{O}}(d\sqrt{K})$. It remains, however, to justify that $P_{pre}(\cdot|D_k)$ will be covered by Gaussian Thompson Sampling for all D_k with $k \in [K]$. This is verified by noting that $P_{ps}(a|D_k) > 0$ for non-degenerate Gaussian Thompson Sampling (positive variances of the prior and likelihood functions) and finite K. This guarantees that any D_k will have support. □

C.5 Proof of Proposition ??

*

Proof. The proof follows by direct inspection of the pretraining distributions. For P^1_{pre}, we have

$$P^1_{pre}(a^\star|s_{\text{query}}, D, \xi) = \int_\tau \pi^\star_\tau(a^\star|s_{\text{query}})P^1_{pre}(\tau|s_{\text{query}}, D, \xi)d\tau \tag{55}$$

The posterior distribution over tasks is simply

$$\begin{aligned}
P^1_{pre}(\tau|s_{\text{query}}, D, \xi) &= \frac{P^1_{pre}(\tau, s_{\text{query}}, D, \xi)}{P^1_{pre}(s_{\text{query}}, D, \xi)} && (56)\\
&\propto P^1_{pre}(\tau)P^1_{pre}(\xi|\tau)\mathcal{D}_{\text{query}}(s_{\text{query}})\mathcal{D}^1_{\text{pre}}(D;\tau) && (57)\\
&= P^2_{pre}(\tau)P^2_{pre}(\xi|\tau)\mathcal{D}_{\text{query}}(s_{\text{query}})\mathcal{D}^1_{\text{pre}}(D;\tau) && (58)
\end{aligned}$$

Then, the distribution over the in-context dataset can be decomposed as

$$\begin{aligned}
\mathcal{D}^1_{pre}(D;\tau) &= \prod_{i\in[n]} R_\tau(r_i|s_i,a_i)T_\tau(s'_i|s_i,a_i)\mathcal{D}^1_{\text{pre}}(a_i|s_i,D_{i-1};\tau) && (59)\\
&= \prod_{i\in[n]} R_\tau(r_i|s_i,a_i)T_\tau(s'_i|s_i,a_i)\mathcal{D}^1_{\text{pre}}(a_i|s_i,D_{i-1}) && (60)\\
&\propto \prod_{i\in[n]} R_\tau(r_i|s_i,a_i)T_\tau(s'_i|s_i,a_i)\mathcal{D}^2_{\text{pre}}(a_i|s_i,D_{i-1}) && (61)\\
&= \mathcal{D}^2_{pre}(D;\tau), && (62)
\end{aligned}$$

where the second equality holds because $\mathcal{D}^1_{\text{pre}}(a_j|s_j,D_j;\tau)$ is assumed to be invariant to τ by compliance, and the fifth equality holds because $\mathcal{D}^2_{\text{pre}}(a_j|s_j,D_j;\tau)$ is assumed to be invariant to τ. Therefore, we conclude that, for any s, D, ξ,

$$\begin{aligned}
P^1_{pre}(\tau|s,D,\xi) &\propto P^2_{pre}(\tau)P^2_{pre}(\xi|\tau)\mathcal{D}_{\text{query}}(s)\mathcal{D}^2_{\text{pre}}(D;\tau) && (63)\\
&\propto P^2_{pre}(\tau|s,D,\xi). && (64)
\end{aligned}$$

Since also $\int_\tau P^1_{pre}(\tau|s,D,\xi) = 1 = \int_\tau P^2_{pre}(\tau|s,D,\xi)$, then

$$P^1_{pre}(\tau|s,D,\xi) = P^2_{pre}(\tau|s,D,\xi). \tag{65}$$

Substituting this back into Equation45 yields $P^1_{pre}(a^\star|s,D,\xi) = P^1_{pre}(a^\star|s,D,\xi)$. $\square$