SCIENCE CHINA
Information Sciences

• Supplementary File •

Prompt tuning with preference ranking for few-shot pre-trained decision transformer

Shengchao Hu[1,2], Li Shen[3*], Ya Zhang[1,2] & Dacheng Tao[4]

[1]*Shanghai Jiao Tong University, Shanghai* 200240*, China*
[2]*Shanghai AI Laboratory, Shanghai* 200233*, China*
[3]*Shenzhen Campus of Sun Yat-sen University, Shenzhen* 518107*, China*
[4]*Nanyang Technological University, Singapore* 639798*, Singapore.*

Appendix A More analysis

Appendix A.1 Main results

In the context of the PTDT-online method, there is an intriguing observation: the experimental outcomes in the Ant-dir environment, when initialized randomly, surpass those achieved through expert initialization. This counterintuitive finding warrants attention and further analysis. We attribute this phenomenon to the presence of local optima when initialized with expert prompts. Specifically, when the prompt is randomly initialized, our algorithm correctly estimates the convergence direction, allowing for more effective convergence toward a global optimum at a larger learning rate. Conversely, when the prompt is initialized with expert guidance, it is already situated within a local optima region. Therefore, if learning continues with a larger learning rate, its performance gradually deteriorates, moving away from the local convergence region. Conversely, if learning continues with a smaller learning rate, it remains constrained within the local convergence domain. Hence, it is possible that the performance of PTDT-online under random initialization conditions might surpass its performance under expert initialization conditions.

However, this pattern is not observed for PTDT-offline. This discrepancy can be attributed to the fact that PTDT-online can directly interact with the simulator, and the cumulative rewards it acquires provide unbiased estimates of the current policy, facilitating the accurate estimation of the global convergence direction. In contrast, PTDT-offline is constrained by the limitations of the offline dataset, making it unable to achieve unbiased estimation of the global convergence direction.

Appendix A.2 Limitation and possible solutions

Our method relies on the initial distribution of the prompt, which serves as the starting point for optimization. While this approach proves effective in many scenarios, it may encounter challenges in complex environments such as Meta-World-10-Reward. In such cases, the desired optimal prompt may lie outside the original distribution, making it difficult for the current framework to effectively optimize a randomly initialized prompt. This limitation underscores the need for exploring methods that can go beyond the constraints of the initial prompt distribution.

To overcome this limitation, we propose exploring alternative frameworks that do not rely solely on optimizing a single prompt within the given distribution. Instead, these approaches could focus on generating prompts that are better aligned with the demands of complex environments. Below, we outline one promising direction:

• **Incorporating Generative Models for Prompt Creation:** Instead of starting with a single prompt and refining it through optimization, we could leverage generative models to directly generate new prompts. Such generative models could be pre-trained on a diverse set of tasks, allowing them to capture a broader range of potential prompt distributions. These generative models would enable the creation of prompts that incorporate richer prior knowledge and are better suited for complex environments, where the optimal prompt may not be easily reachable through local optimization of a random initialization.

• **Hybrid Approach Combining Generation and Optimization:** A hybrid approach could be considered, where a generative model first produces a pool of candidate prompts, which are then fine-tuned using our optimization framework. This two-stage process combines the strengths of generation and refinement, ensuring that the prompts are both diverse and well-suited for the specific task.

• **Meta-Learning for Prompt Initialization:** Another avenue is to incorporate meta-learning techniques to train a meta-model capable of generating task-specific initial prompts. By leveraging knowledge from prior tasks, the meta-model could provide a strong starting point for optimization in complex environments.

* Corresponding author (email: mathshenli@gmail.com)

Appendix B Detailed environment

We evaluate our approach on a variety of tasks, including meta-RL control tasks. These tasks can be described as follows:

• **Cheetah-dir**: The task comprises two directions: forward and backward, in which the cheetah agent is incentivized to attain high velocity along the designated direction. Both the training and testing sets encompass these two tasks, providing comprehensive coverage of the agent's performance.

• **Cheetah-vel**: In this task, a total of 40 distinct tasks are defined, each characterized by a different goal velocity. The target velocities are uniformly sampled from the range of 0 to 3. The agent is subjected to a penalty based on the l_2 error between its achieved velocity and the target velocity. We reserve 5 tasks for testing purposes and allocate the remaining 35 tasks for training.

• **Ant-dir**: There are 50 tasks in Ant-dir, where the goal directions are uniformly sampled in a 2D space. The 8-joint ant agent is rewarded for achieving high velocity along the specified goal direction. We select 5 tasks for testing and use the remaining tasks for training.

• **Meta-World reach-v2**: This task involves controlling a Sawyer robot's end-effector to reach a target position in 3D space. The agent directly controls the XYZ location of the end-effector, and each task has a different goal position. We train on 15 tasks and test on 5 tasks.

• **Meta-World-10-Reward**: This benchmark comprises 10 distinct tasks, each with different reward and transition functions, but sharing the same state and action spaces. It is designed to evaluate the agents' adaptability to more complex meta-environments.

By evaluating our approach on these diverse tasks, we can assess its performance and generalization capabilities across different control scenarios. The experimental setup in Section 5 adheres to the training and testing division specified in Table B1. This ensures consistency and allows for a comprehensive assessment of the approach's performance across tasks.

Table B1 Training and testing task indexes when testing the generalization ability in meta-RL tasks

Cheetah-dir	
Training set of size 2	[0,1]
Testing set of size 2	[0.1]
Cheetah-vel	
Training set of size 35	[0-1,3-6,8-14,16-22,24-25,27-39]
Testing set of size 5	[2,7,15,23,26]
Ant-dir	
Training set of size 45	[0-5,7-16,18-22,24-29,31-40,42-49]
Testing set of size 5	[6,17,23,30,41]
Meta-World reach-v2	
Training set of size 15	[1-5,7,8,10-14,17-19]
Testing set of size 5	[6,9,15,16,20]
Meta-World-10-Reward	
Training set of size 7	[coffee-pull, door-close, door-lock, door-unlock, handle-press-side, handle-press, handle-pull-side]
Testing set of size 3	[coffee-push, door-open, handle-pull]

Appendix C Hyperparameters

We show the hyperparameters of Prompt-Tuning DT in Table C1.

Appendix D Normalization

Table D1 presents the normalized scores used for normalization. The values are selected from the collected datasets, with "Random" corresponding to the minimal value and "Gamer" corresponding to the maximal value.

Appendix E Detailed resource

We have provided detailed comparisons of the time and resource requirements for our method and the baseline approaches during a single step of the tuning process (Table E1). The results demonstrate that our method achieves the best performance while utilizing the least GPU memory and requiring comparable computational time. This highlights both the efficiency and effectiveness of our approach.

Table C1 Common Hyperparameters of Prompt-Tuning DT.

Hyperparameters	Value
K (length of context τ)	20
training batch size for each task	32
training number of steps per iteration	10
training max iterations	5000
fine-tuning batch size for each task	32
fine-tuning number of steps per iteration (offline)	50
fine-tuning number of steps per iteration (online)	10
fine-tuning max iterations	20
number of evaluation episodes for each task	50
learning rate	[1e-3, 1e-6, 1e-8]
learning rate decay weight	1e-4
ranking algorithm m	[10, 20, 30, 50]
ranking algorithm k	same as m
ranking algorithm μ	[0.5, 1.0, 5.0, 10.0]
Return-to-go conditioning	1500 Cheetah-dir
	0 Cheetah-vel
	500 Ant-dir
	650 Meta-World reach-v2
	4500 Meta-World-10-Reward
Prompt length K^*	5
number of layers	3
number of attention heads	1
embedding dimension	128
activation	ReLU

Table D1 Baseline scores used for normalization.

Game	**Random**	**Gamer**
Cheetah-dir	−10.1	994.73
Cheetah-vel	−200.018	−18.864
Ant-dir	−31.046	521.31
Meta-World Reach-v2	0	555.01
Coffee-push-v2	3.2	1000.48
Door-open-v2	170.47	1436.25
Handle-pull-v2	1.29	4695.51

Table E1 Ablation study examining the Time and GPU Memory with one update process. These experiments are performed in the Ant-dir environment with random prompt initialization using a single NVIDIA 4090 GPU.

Method	Soft Prompt	Adaptor	PTDT-ffline
Time	6min46s	7min22s	6min55s
GPU Memory	562MiB	628MiB	512MiB
Performance	66.5 ± 0.4	75.3 ± 6.6	75.3 ± 3.9