LLM-Empowered State Representation for Reinforcement Learning

Boyuan Wang*, Yun Qu*, Yuhang Jiang, Jianzhun Shao, Chang Liu, Wenming Yang, Xiangyang Ji,

Tsinghua University

ICML International Conference On Machine Learning

More Task-Related State Representaions

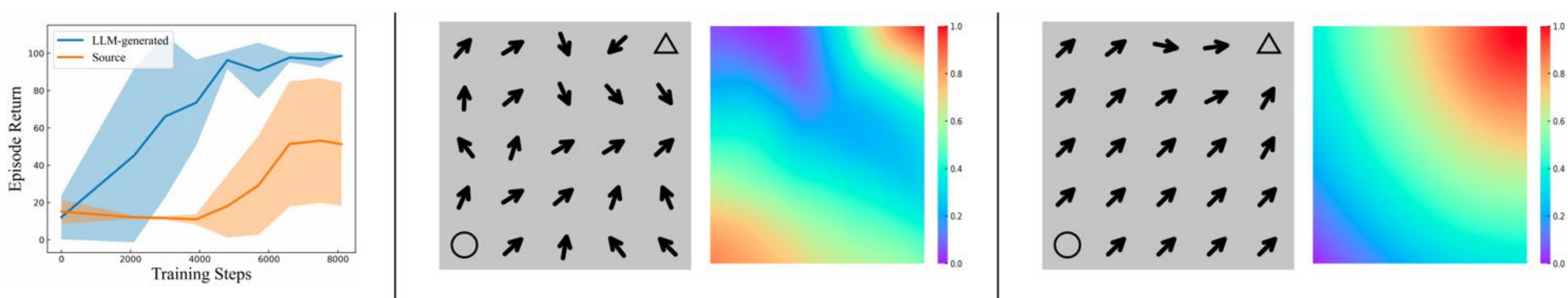
- **Establishing Mappings from States to Task Rewards**

Conventional state representations in reinforcement learning **often omit critical task-related details**, presenting **challenge for value networks** in establishing accurate mappings from states to task rewards.

- **Lipschitz continuity and Training Efficiency**

Utilizes LLM to autonomously **generate task-related state representation and intrinsic reward python functions** which help to enhance the Lipschitz continuity of value network mappings and facilitate efficient training

Origin of Inspiration – A Toy Example

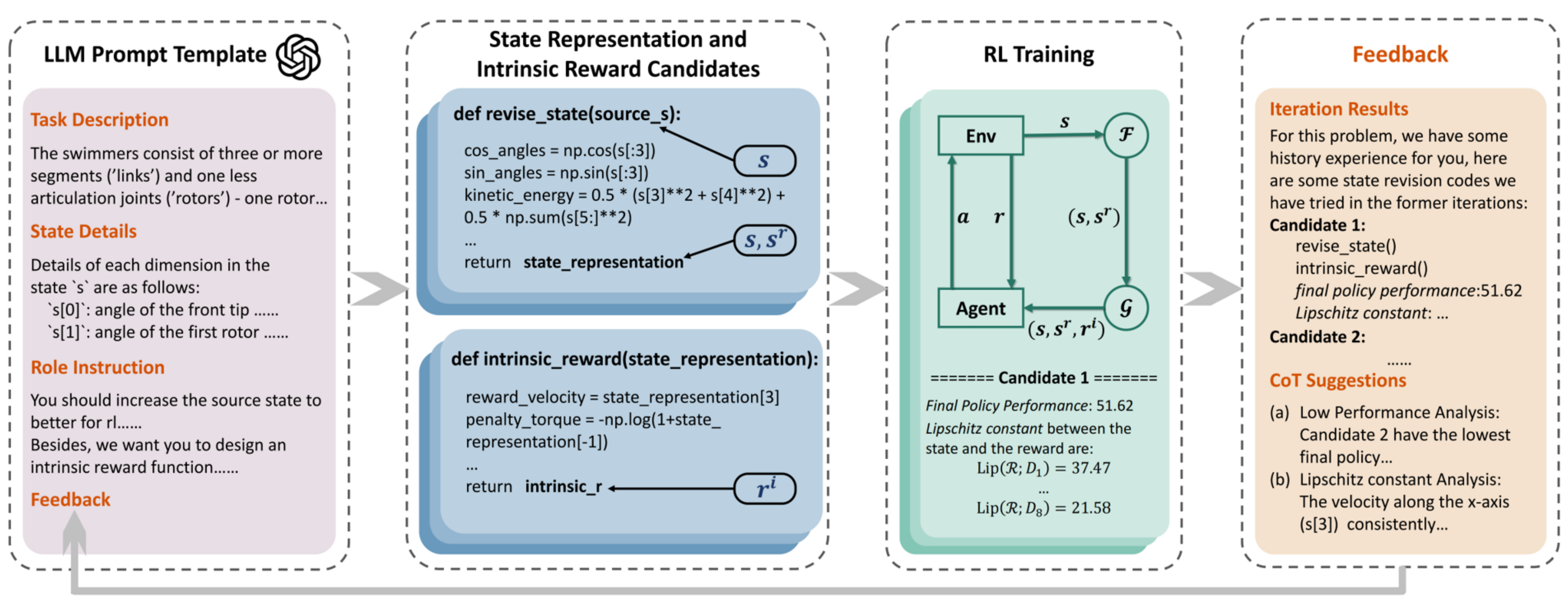


(a) Training Curves (b) Source, $\mathrm{Lip}(Q) = 37.47$ (c) LLM-generated, $\mathrm{Lip}(Q) = 3.15$

Navigate from the bottom left corner (circle) to the top right corner (triangle)

- **Figure (b):** The original state representations (x1, y1, x2, y2)
- **Figure (c):** LLM-Generated state representations (x1, y1, x2, y2, **d**). **d** denotes the distance between the agent and the target. This modification facilitates easier learning, enhancing Lipschitz continuity and convergence.

Algorithm Framework



- **Step 1:** Utilize LLM to generate the **state representation and intrinsic reward python functions**
- **Step 2:** For each python function candidates, initiate RL training to validate their effectiveness
- **Step 3:** Provide the **training performance and Lipschitz constant between state representations and rewards as feedback**, enabling LLM to generate improved functions in the next iteration

Continuous and Discontinuous Scenarios

- **Continuous Scenarios**

Feedback LLM with the Lipschitz constant between each dimension of the LLM-generated state representations and the extrinsic reward. This assessment is crucial for guiding the LLM in identifying and eliminating undesired dimensions within the state representations.

- **Discontinuous Scenarios**

LESR with Discounted Return The Lipschitz constant between the LLM-generated state representations and the discounted return $\sum_t \gamma^t r$

LESR with Spectral Norm Use the spectral norm to estimate $Lip(V;\mathcal{S})$ as feedback to LLM. Calculating the spectral norm of the N weight matrices $W_1, \ldots, W_N$ of the value networks, $Lip(V;\mathcal{S})$ is bounded by $\prod_{i=1}^{N} ||W_i||_2$, which is then presented to the LLM as feedback.

Theoretical Analysis

Why smoother net work mappings?

- **Theorem 3.4:** The mapping exhibiting a lower Lipschitz constant is easier to learn and can attain superior convergence.

Why Lipschitz constant is crucial for RL?

- **Theorem B.4:** Reducing $Lip(r;\mathcal{S})$ lowers the upper bound of $Lip(V;\mathcal{S})$
- **Theorem B.8:** Reducing $Lip(V;\mathcal{S})$ enhance the convergence of V

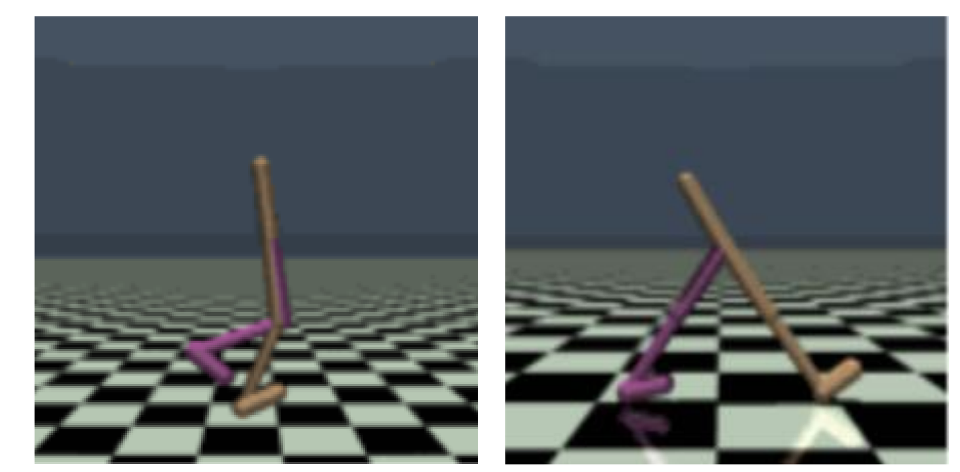
Facilitate Efficient Training

Environments \ Algorithm	TD3	LESR(Ours)	RPI
HalfCheetah	7288.3±842.4	**7639.4±464.1**	**5%**
Hopper	921.9±646.8	**2705.0±623.2**	**193%**
Walker2d	1354.5±734.1	**1874.8±718.2**	**38%**
Ant	1665.2±895.5	**1915.3±885.5**	**15%**
Swimmer	49.9±5.2	**150.9±9.5**	**203%**
Mujoco Improve Mean	-	-	**91%**

Robustness and Stability

Environments \ Algorithm	TD3	Directly Intrinsic	LESR w/o ER	LESR w/o LC	LESR(Ours)
HalfCheetah	9680.2±1555.8	8919.9±1761.9	10252.8±277.1	10442.9±304.4	**10614.2±510.8**
Hopper	3193.9±507.8	3358.4±76.6	**3408.4±144.0**	3362.3±109.4	**3424.8±143.7**
Walker2d	3952.1±445.7	1924.3±969.3	3865.8±142.5	**4356.2±335.3**	**4433.0±435.3**
Ant	3532.6±1265.3	3518.0±147.6	**4779.1±30.4**	3242.3±459.9	4343.4±1171.4
Swimmer-v3	84.9±34.0	25.2±4.1	51.9±0.1	116.9±6.3	**164.2±7.6**

Generalization and Adaptability



Testing on two novel scenarios

- Walker Jump
- Walker Split Legs

Benchmark Results and Ablation Study

Environments \ Algorithm	TD3	EUREKA	RPI	Ours w/o IR	RPI	Ours w/o SR	RPI	Ours w/o FB	RPI	LESR(Ours)	RPI
HalfCheetah	9680.2±1555.8	10400.6±289.2	7%	9969.8±1767.5	3%	9463.2±796.3	-2%	9770.4±1531.3	1%	**10614.2±510.8**	**10%**
Hopper	3193.9±507.8	3346.4±423.2	5%	3324.7±191.7	4%	3159.0±466.7	-1%	2851.3±748.1	-11%	**3424.8±143.7**	**7%**
Walker2d	3952.1±445.7	3606.2±1010.0	-9%	4148.2±352.9	5%	3977.4±434.3	1%	4204.9±590.3	6%	**4433.0±435.3**	**12%**
Ant	3532.6±1265.3	2577.7±1085.6	-27%	**5359.0±336.4**	**52%**	3962.2±1332.0	12%	4244.9±1227.9	20%	4343.4±1171.4	23%
Swimmer	84.9±34.0	98.1±31.1	16%	132.0±6.0	55%	160.2±10.2	89%	85.4±29.8	1%	**164.2±7.6**	**93%**
Mujoco Relative Improve Mean	-	-	-2%	-	24%	-	21%	-	3%	-	**29%**
-	-	-	**PI**	-	**PI**	-	**PI**	-	**PI**	-	**PI**
AntMaze_Open	0.0±0.0	0.13±0.04	13%	0.0±0.0	0%	0.15±0.06	15%	0.16±0.07	16%	**0.17±0.06**	**17%**
AntMaze_Medium	0.0±0.0	0.01±0.01	1%	0.0±0.0	0%	0.07±0.05	7%	0.0±0.0	0%	**0.1±0.05**	**10%**
AntMaze_Large	0.0±0.0	0.0±0.0	0%	0.0±0.0	0%	0.06±0.03	6%	0.04±0.04	4%	0.07±0.03	7%
FetchPush	0.07±0.02	0.07±0.03	0%	0.78±0.16	71%	0.06±0.05	-1%	0.77±0.14	70%	**0.91±0.08**	**84%**
AdroitHandDoor	0.53±0.44	0.83±0.08	30%	0.46±0.46	-7%	0.63±0.39	10%	0.23±0.38	-30%	**0.88±0.12**	**34%**
AdroitHandHammer	0.28±0.32	0.32±0.45	4%	0.22±0.21	-6%	0.29±0.28	1%	0.41±0.33	13%	**0.53±0.38**	**25%**
Gym-Robotics Improve Mean	-	-	8%	-	8%	-	6%	-	10%	-	**30%**

Code: https://github.com/thu-rllab/LESR

Lab: https://github.com/thu-rllab

My Github: https://github.com/BoyuanWang-hub