LLMs Can Implement Policy Iteration

Ethan Brooks[1] , Logan Walls[2] , Richard L. Lewis[2] , Satinder Singh[1]
[1]Computer Science and Engineering, University of Michigan
[2]Department of Psychology, University of Michigan

LLMs Can Implement Policy Iteration

1. Feed the MDP into the LLM.
2. Use the LLM to *estimate value.*
3. Use these estimates in Policy Iteration.

Results Overview

LLM goes from random to near-optimal performance in **100s** of time-steps.

- Domains are toy / text-based

Only large models learn:

- **GPT-J (6B params):** doesn't learn
- **InCoder (6.7B params):** doesn't learn
- **OPT (30B params):** doesn't learn
- **code-cushman-001:** learns inconsistently
- **code-cushman-002:** *learns consistently*

Interacting with the environment

During episode:

- Observes state
- For each action in action space:
  - Compute value given state and action
- Choose action with highest value
- Receive reward and next state.
- Add interaction to replay buffer
- **We use the LLM to compute value.**

Estimating $Q^{\pi}(s_t, a)$ Values

- Generate rollout sampled from current policy π starting with action, a
- Use LLM to alternately model
  - transition (next-state, reward, termination)
  - current policy
- Use rollout to estimate value:
  - $Q^{\pi}(s_t, a) = \sum_{u=t}^{T} \gamma^{T-u} r_u$
  - Result is unbiased Monte-Carlo estimate

Chain Environment

Navigate to goal state and “try” it.

- Initial state
  - Agent spawns randomly
- Actions
  - Left
  - Right
  - “Try goal”
- Reward
  - 1 for “try goal” on state 4
  - 0 otherwise
- Termination
  - On “try goal” (any state)
  - After fixed time limit

State	1	2	3	4	5	6	7	8
Reward (on try-goal)	0	0	0	1	0	0	0	0

LLM as Next-State Model

Random transitions

state == 2
right
state == 3

state == 6
right
state == 7

state == 0
right
state == 1

Prefix demonstrates that **right** increments state

state == 4
right
state == 5

LLM generalizes to suffix state

State	1	2	3	4	5	6	7	8
Reward (on try-goal)	0	0	0	1	0	0	0	0

LLM as Reward Model

Random transitions

state == 2
right
reward == 0

state == 4
right
reward == 1

state == 0
right
reward == 0

Prefix demonstrates that **state 4 is the goal**

state == 4
right
reward == 1

LLM infers reward for suffix state

State	1	2	3	4	5	6	7	8
Reward (on try-goal)	0	0	0	1	0	0	0	0

LLM as Policy

Trajectories sampled from recent episodes

state == 2
right
reward == 0

Policy moves right when state < 4

state == 5
left
reward == 0
state == 4
try goal
reward == 1

Policy moves left when state > 4

state == 1
right

LLM generalizes to suffix state

State	1	2	3	4	5	6	7	8
Reward (on try-goal)	0	0	0	1	0	0	0	0

Policy Improvement

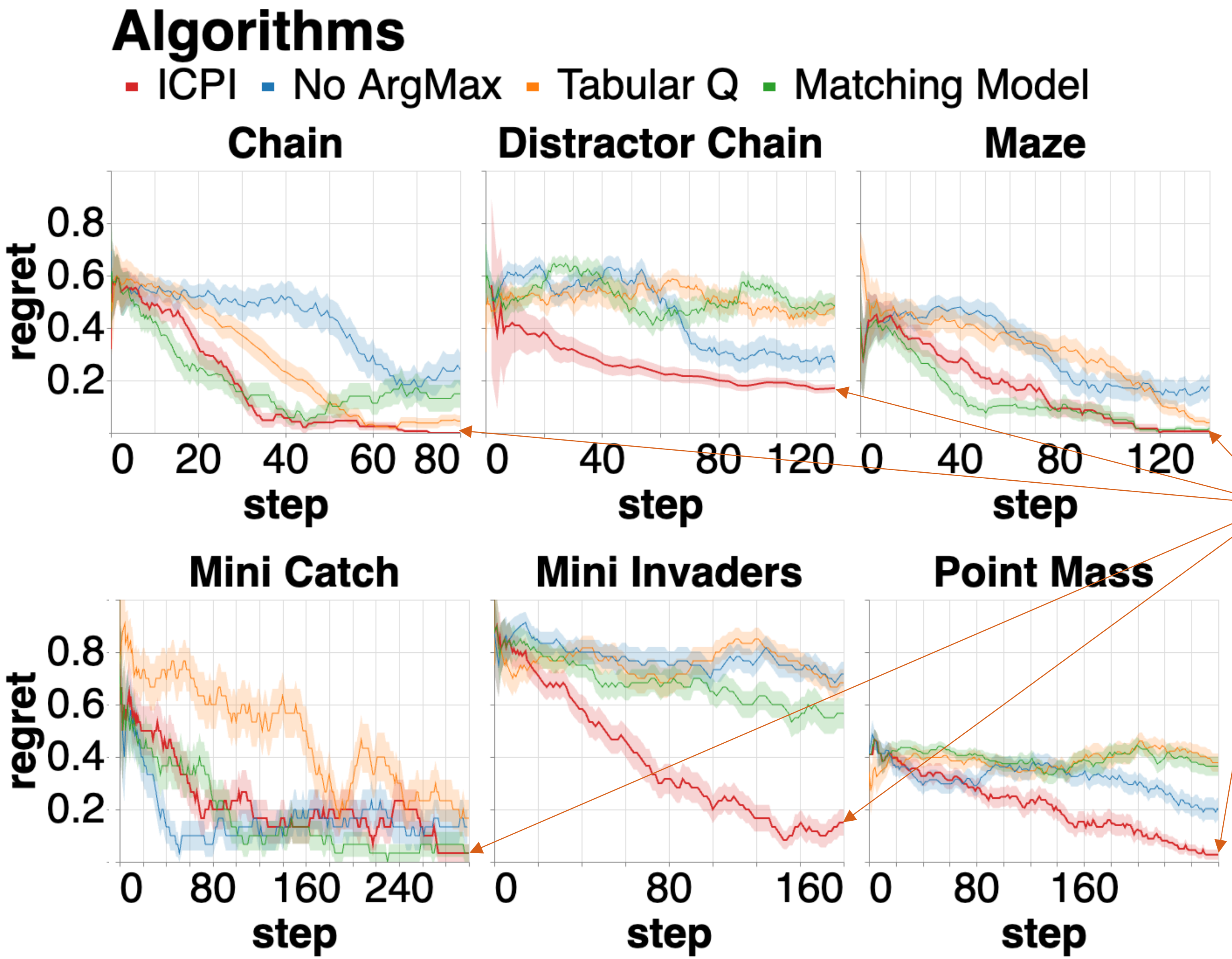
Behavior Policy	Prompt Policy	Rollout Policy	Behavior Policy

$$\pi \xrightarrow{\approx} \pi_{\text{prompt}} \xrightarrow{\approx} \pi_{\text{rollout}} \xrightarrow{\operatorname*{argmax}_a \tilde{Q}^{\pi_{\text{rollout}}}(h_{t-1}, a)} \pi'$$

$$\pi' \xrightarrow{\approx} \pi'_{\text{prompt}} \xrightarrow{\approx} \pi'_{\text{rollout}} \xrightarrow{\operatorname*{argmax}_a \tilde{Q}^{\pi'_{\text{rollout}}}(h_{t-1}, a)} \pi''$$

…

$$V^{\pi} \approx V^{\pi_{\text{prompt}}} \approx V^{\pi_{\text{rollout}}} \leq V^{\pi'} \approx V^{\pi'_{\text{prompt}}} \approx V^{\pi'_{\text{rollout}}} \leq V^{\pi''} \ldots$$



Our method in red
(lower is better)

Why RL + LLMs is a happy marriage

- RL can leverage knowledge distilled in LLMs to learn rapidly.
- LLMs can use RL to improve without further (gradient-based) training.