Learning Distance-to-Goal Functions for Goal-Conditioned RL in Sparse Environments

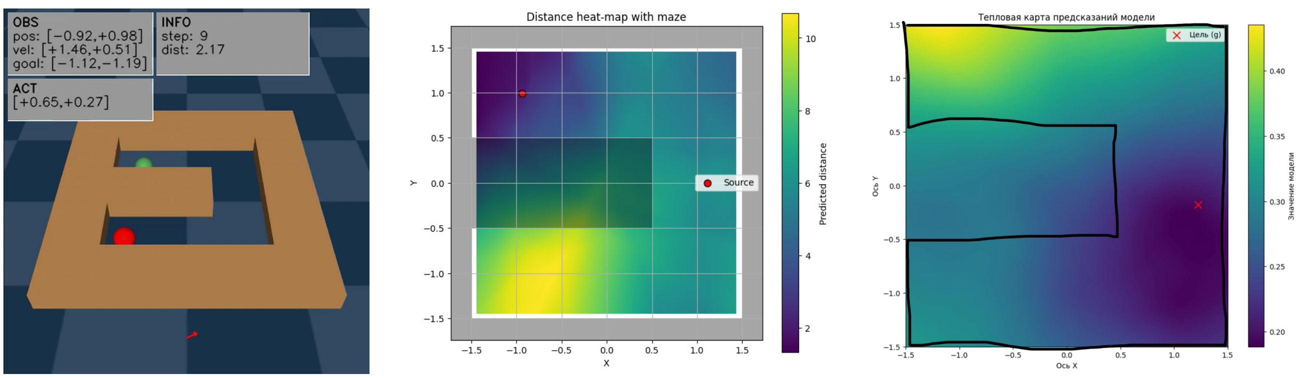
Artem Voronov, Vladislav Ulendeev, Sofia Gamershmidt, Petr Kuderov

July 9, 2025

1 Problem Statement

In goal-conditioned reinforcement learning (RL) an agent must move from its current state s to a desired goal g. If the reward is *sparse*—zero everywhere and +1 only at the goal—the agent rarely sees a positive signal, so learning stalls. Classic fixes such as Hindsight Experience Replay help but can still fail in mazes with dead ends.

A simple idea is to give the agent a dense hint: the **distance-to-goal** $D(s, g)$. If the agent knows (or is rewarded for reducing) this distance it can discover good actions sooner. We ask: *Can a learned distance function speed up PPO on the `PointMaze_UMaze-v3` task* (Fig. 1)? We compare seven variants: plain PPO (baseline) and six versions that use a distance model either for Reward shaping, as an Observed feature, or Both—all trained with either supervised labels (**Sup**) or temporal-difference updates (**TD**) (Table 1).



(a) Environment (b) Sup. heat-map (c) TD heat-map

Figure 1: Task and learned distance fields (light = farther).

2 Method Summary

We learn the PPO agent and the distance module *together* in an iterative loop. Each **stage** has four steps:

Step 0 — Cold start. Run the current PPO agent for N episodes with the sparse reward and record initial transition (x_t, y_t).

Step 1 — Build dataset. For every trajectory enumerate all state–goal pairs (s, g) that occur on the path and compute their trajectory-path distance $d^\star$. Store the resulting triples $(s, g, d^\star)$; the dataset flushed each stage.

Step 2 — Train distance model f_θ. Train distance estimator model:

- **Sup - Supervised**: minimise $\|f_\theta(s, g) - d^\star\|^2$.
- **TD - Bootstrapped**: enforce $f_\theta(s, g) \approx 1 + f_\theta(s', g)$ for each transition $s \to s'$ (and 0 at the goal).

Step 3 — Use f_θ while retraining PPO Train PPO for M updates while using the distance estimate in one of three ways:

- R - **Reward shaping:** $r_t \leftarrow r_t^{\text{sparse}} + \gamma \left[\Phi(s_{t+1}) - \Phi(s_t)\right]$, where $\Phi(s) = -f_\theta(s, g)$.
- O - **Observation feature:** augment the state with $d_t = f_\theta(s_t, g)$.

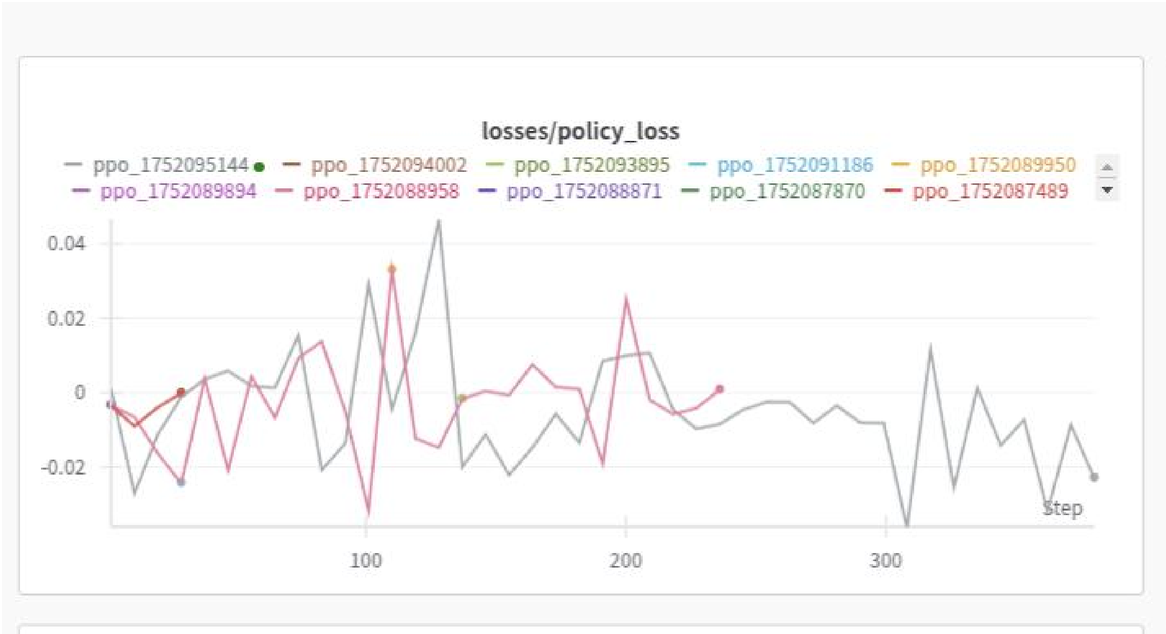


Figure 2: Learning curves (mean ± std. over 10 seeds).

B - **Both**: apply R and O simultaneously.

The policy's exploration enlarges the dataset, the distance model becomes more accurate, and the improved guidance accelerates the next stage.

Variants compared: Baseline, Sup-R, Sup-O, Sup-B, TD-R, TD-O, TD-B.

3 Experimental Setup

Each variant is trained for 3×10^6 steps, repeated over 10 seeds. The metric is mean episodic return (success rate). PPO hyper-parameters and network size (2×64) are shared. Distance nets are trained on 10^5 pairs; TD training runs for 5 epochs over collected roll-outs.

4 Results

Figure 2 shows rapid progress once distance information is introduced. Key observations:

- **Reward shaping** (Sup-R, TD-R) yields the earliest gains.
- **Observation only** helps but lags shaping.
- **Both** signals (Sup-B) achieve near-perfect success fastest.
- Sup models outperform TD owing to more accurate initial distance estimates.

Table 1: Final success rate (mean ± SD, last 10^4 steps).

Variant	Success	Seeds solved
PPO Baseline	0.11 ± 0.14	2/10
Sup-R	0.83 ± 0.09	10/10
Sup-O	0.64 ± 0.18	8/10
Sup-B	$\mathbf{0.95 \pm 0.04}$	10/10
TD-R	0.75 ± 0.14	10/10
TD-O	0.52 ± 0.23	6/10
TD-B	0.88 ± 0.10	10/10

5 Future Work (Brief)

1. **Harder tasks & generalisation.** Scale to deeper mazes and unseen goal layouts to test how well the distance signal transfers.
2. **Distance-guided exploration.** Use f_θ directly to propose exploratory actions (e.g. follow the steepest descent in predicted distance or plan short local detours), reducing the need for random exploration.
3. **Online distance refinement.** Update f_θ on fresh roll-outs while PPO is training, so the shaping signal stays accurate as the agent discovers new parts of the state space.

Conclusion

A learned distance-to-goal function—especially when used for potential-based shaping—turns a nearly unsolvable sparse-reward maze into a reliably solved task, cutting both data and seed variance.