Lecture 10: Assistance Games

From CIRL to Assistance POMDPs, and Scaling with AssistanceZero

1 Where We Are

A couple of lectures ago we covered reward misspecification and a spectrum of remedies, ending with CIRL (Hadfield-Menell et al., 2016): rather than learn a reward *then* optimize it (RLHF), formalize the problem itself as a cooperative game between a human who knows the objective and a robot who does not. Today we zoom into that framework, give it its modern name (**assistance games**), show that it reduces to a POMDP with a very familiar structure, connect it explicitly to the BAMDPs from our exploration lectures, and then look at how recent work (AssistanceZero, Laidlaw et al. 2025).

After that we'll look at a few other topics as a whirlwind tour so you are aware of them and can think about your future research directions.

Discussion

What are some of the areas that you're most excited about in terms of the future of RL research and RL in deployment? What do you think are the hardest unsolved challenges?

2 From RLHF to Assistance Games

Recall the RLHF pipeline. Humans rank responses, we fit a reward model $\hat{r}$, and then we train a policy π to maximize $\hat{r}$.

In an assistsance game, a human and the robot act in the *same* environment, share a reward function $R(s, a^H, a^R; \theta)$, and the reward parameters θ start out visible only to the human.

An assistance game is a tuple

$$\mathcal{M} = \Big\langle \underbrace{\mathcal{S}}_{\text{states}}, \underbrace{\mathcal{A}^H, \mathcal{A}^R}_{\text{actions}}, \underbrace{\Theta}_{\text{reward params}}, \gamma, \underbrace{p_0(s_1, \theta)}_{\text{initial dist.}}, \underbrace{P(s_{t+1} \mid s_t, a_t^H, a_t^R)}_{\text{transition}}, \underbrace{R(s_t, a_t^H, a_t^R; \theta)}_{\text{shared reward (hidden } \theta)} \Big\rangle$$

consisting of

- a shared state space $\mathcal{S}$ and action spaces $\mathcal{A}^H$, $\mathcal{A}^R$ for the human and robot;
- a reward-parameter space Θ
- an initial start state distribution $p_0(s_1, \theta)$;
- a transition function $P(s_{t+1} \mid s_t, a_t^H, a_t^R)$;
- a shared reward $R(s_t, a_t^H, a_t^R; \theta)$; and

- a discount $\gamma \in [0, 1]$.

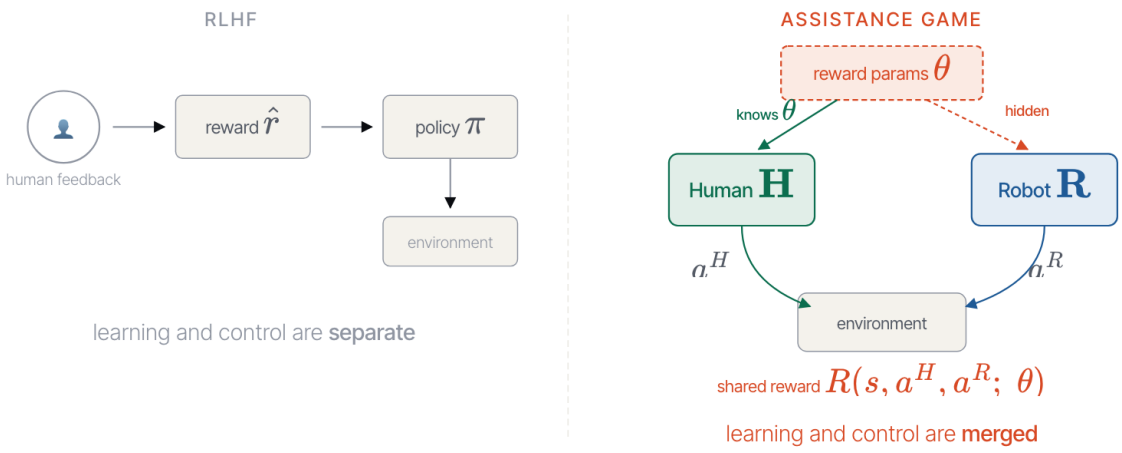


Figure 1: Figure adapted from Laidlaw et al. (2025).

The human policy $\pi_H(a^H \mid s, \theta)$ conditions on θ; the robot policy $\pi_R(a_t^R \mid h_t)$ conditions only on the history $h_t = (s_1, a_1^H, a_1^R, \ldots, s_t)$. The joint return is

$$J(\pi_H, \pi_R) \;=\; \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1}\, R(s_t, a_t^H, a_t^R; \theta)\right].$$

If we assume the human policy $\pi_H(a^H \mid s, \theta)$ is fixed, Shah et al. (2020) show that, from the robot's perspective, computing a best response to π_H reduces to solving a single-agent POMDP. They call this the **assistance POMDP**.

Given an assistance game and a fixed human policy π_H, define a POMDP

$$\bar{M}_{\pi_H} \;=\; (\bar{\mathcal{S}},\, \mathcal{A}^R,\, \bar{P},\, \bar{R},\, \Omega,\, O,\, \gamma).$$

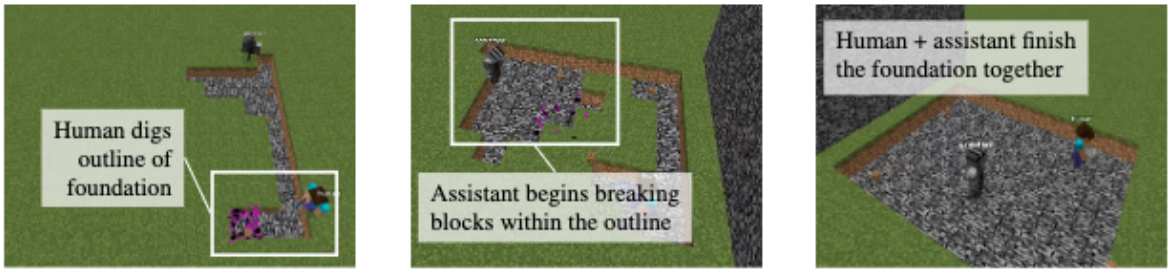
Note the **hidden state** $\bar{s} = (s, \theta) \in \mathcal{S} \times \Theta$.

But solving the *full* assistance game — optimizing jointly over (π_H, π_R) — is harder than solving the induced POMDP because a strategic human will shape a^H to *teach* the robot (Hadfield-Menell et al., 2016) and the policy will evolve over time.

This looks familiar, doesn't it? Recall that in in Lecture 7 we defined the **Bayes-Adaptive MDP** (BAMDP) for exploration: a hyperstate $x = \langle s, p \rangle \in \mathcal{S} \times \Delta(\Theta)$ pairs the physical state with an epistemic belief over unknown MDP parameters θ, and the Bayes-optimal policy maps hyperstates to actions.

2.1 AssistanceZero

Recently Laidlaw et al. (2025) tried to scale assistance games to larger open-ended games and tasks. They used minecraft as the setting.



Digging a foundation: the assistant watches the human outline the house's foundation and then digs it out.

Building a roof: the assistant infers the structure of the roof from human actions and completes it while the human builds another part of the house.

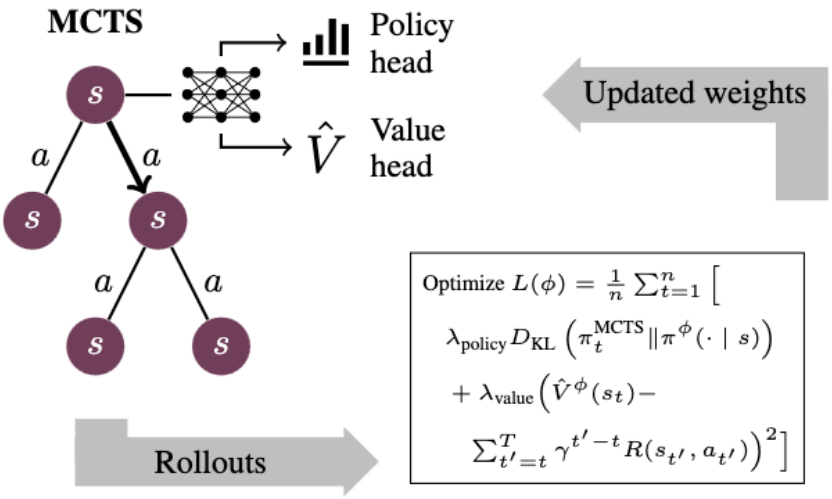
Learning from corrections: the assistant builds the walls too tall, but when the human breaks one of the blocks it learns the correct height and breaks the others.

They assume that the reward has structure. The reward parameters θ consist of a goal grid of blocks. To assign rewards for human and assistant actions, they use the edit distance $d(s, \theta)$ between the current state s and the goal θ, i.e., the minimum number of place and break block actions necessary to transform s into the goal. The reward function

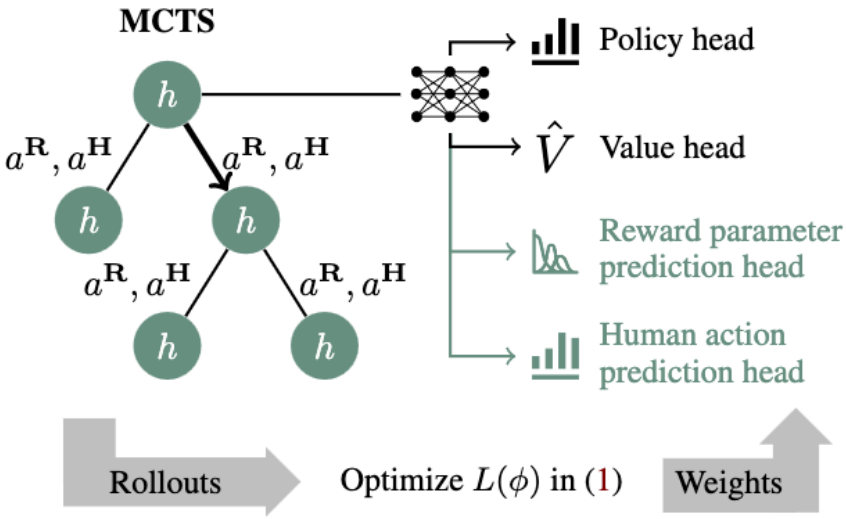
$$R(s, a^H, a^R; \theta) \;=\; d(s', \theta) - d(s, \theta)$$

is the difference in edit distance before and after the assistant and human actions. This means that correct (incorrect) place or break actions give a reward of $+1$ (-1).

To solve this, they introduced AssistanceZero (Laidlaw et al., 2025), extending AlphaZero (Silver et al., 2017) to assistance POMDPs. AlphaZero assumes a known transition model and a known reward; neither holds here (the next state depends on a^H, and the reward depends on θ, both latent). AssistanceZero learns both from rollouts and hands them to MCTS.



(a) AlphaZero



(b) AssistanceZero

Figure 2: Figure adapted from Laidlaw et al. (2025).

This results in loss:

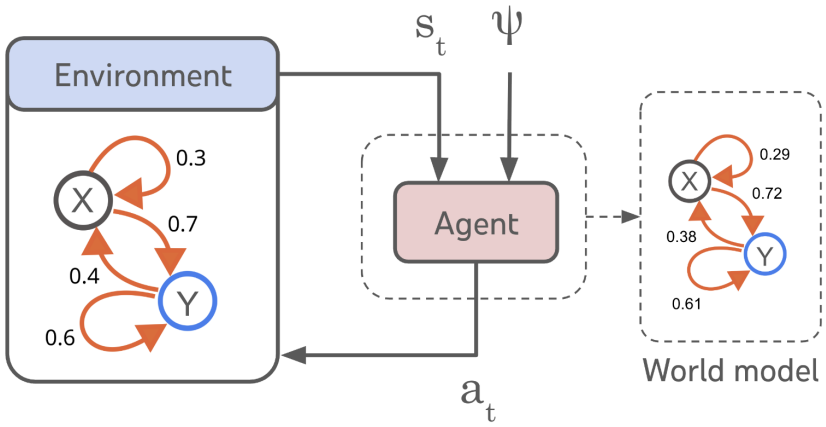


Figure 3: From Richens et al. (2025).

$$L(\phi) = \frac{1}{n}\sum_{t=1}^{n}\Big[\lambda_{\text{policy}}\,\mathrm{KL}\big(\pi_t^{\text{MCTS}} \,\|\, \pi^{\phi}(\cdot \mid h_t)\big) \;+\; \lambda_{\text{value}}\,\big(\hat{V}^{\phi}(h_t) - G_t\big)^2 \\ -\; \lambda_{\text{reward}}\,\log \hat{p}^{\phi}(\theta \mid h_t) \;+\; \lambda_{\text{prev}}\,\mathrm{KL}\big(\hat{p}^{\phi}(\theta \mid h_t) \,\|\, \hat{p}_t(\theta)\big) \\ -\; \lambda_{\text{action}}\,\log \hat{p}^{\phi}(a_t^H \mid h_t)\Big]. \tag{1}$$

with $G_t = \sum_{t' \geq t} \gamma^{t'-t} R(s_{t'}, a_{t'}^H, a_{t'}^R; \theta)$. The λ_{prev} term anchors the current belief to the one that produced the rollout, preventing the reward-parameter head from collapsing to the most recent batch of goals.

Discussion

The authors claim that PPO doesn't really work well for this, which is why they had to do MCTS. Why do you think that is? How could we implement assistance games for language models?

3 From modeling people to modeling the world

If we assume that the person is part of the world and instead the reward is known. We get into model-based RL, that is, you can learn a model of the world and use that for planning and optimization.

In **model-based RL** we fit $\hat{P}(s_{t+1} \mid s_t, a_t)$ and $\hat{R}(s_t, a_t)$ from experience — this is sometimes now called a **world model** — and use it in place of real samples.

Basically, you just train a model as you learn from experience in the real world, so that you can also simulate the environment offline.

Three potential ways (among others) to leverage world models:

- **Planning (MPC).** At each step, roll the model forward H steps, pick the action sequence with

Figure 4: https://worldmodels.github.io/

highest predicted return, execute the first action, replan:

$$a_t^\star = \arg\max_{a_{t:t+H-1}} \mathbb{E}_{\hat{P}}\left[\sum_{k=0}^{H-1} \gamma^k \hat{R}(s_{t+k}, a_{t+k})\right].$$

- **Dyna.** Use imagined transitions as extra data to train a model-free policy (Sutton, 1991).
- **Differentiable imagination.** Backprop policy gradients through the model itself (the Dreamer recipe below).

Ha and Schmidhuber (2018) popularized the term world model and showed that even small RNNs can capture dynamics fairly well. Take a second to play around with their demo live in your browser, from 2018!

You can then play in simulation and learn in simulation, without ever touching the real world.

3.1 Dreamer

The Dreamer series (Hafner et al., 2019, 2020, 2021, 2023) scales this up and shows that you can actually backprop efficiently directly through the world model to improve efficiency.

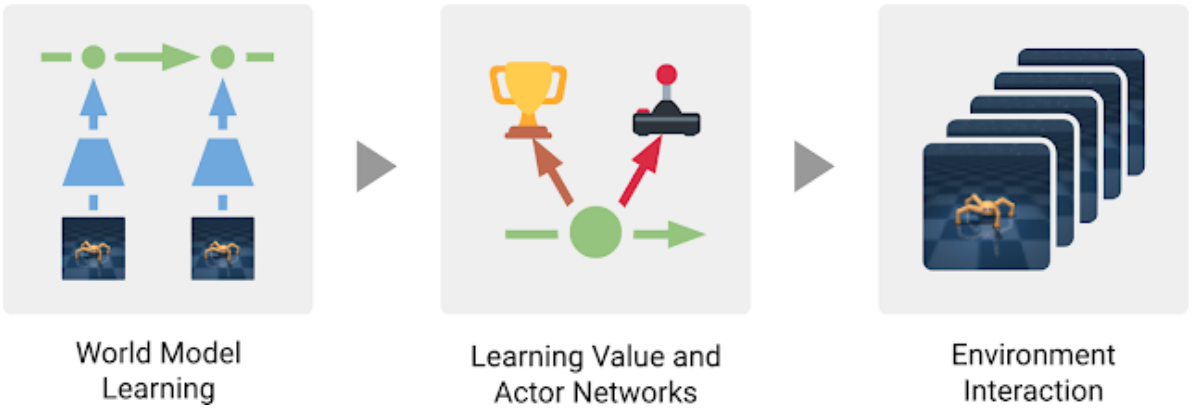


Figure 5: Dreamer's three interleaved loops: fit the world model on replay, learn actor/value networks in imagination, collect more real data. Figure from Hafner et al. (2020).

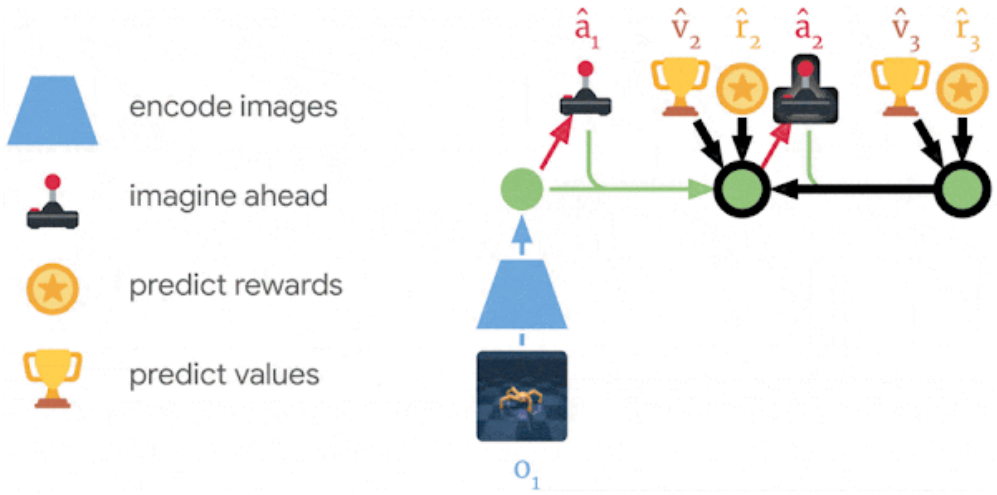


Figure 6: Imagination rollout in latent space: encode o_1 to a state, then roll forward with the learned dynamics to predict actions $\hat{a}_t$, rewards $\hat{r}_t$, and values $\hat{v}_t$ — all without touching the real environment. Figure from Hafner et al. (2020).

Discussion

What can go wrong if you use learned world models for training? On the other hand what are the benefits (hint: think safety)?

General agents contain world models

Jonathan Richens [1] David Abel [1] Alexis Bellot [1] Tom Everitt [1]

Abstract

Are world models a necessary ingredient for flexible, goal-directed behaviour, or is model-free learning sufficient? We provide a formal answer to this question, showing that any agent capable of generalizing to multi-step goal-directed tasks must have learned a predictive model of its environment. We show that this model can be extracted from the agent's policy, and that increasing the agent's performance or the complexity of the goals it can achieve requires learning increasingly accurate world models. This has a number of consequences: from developing safe and general agents, to bounding agent capabilities in complex environments, and providing new algorithms for learning an agent's world model.

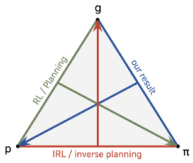


Figure 1: Our result complements previous insights from planning and inverse RL. While planning uses a world model and a goal to determine a policy, and IRL and inverse planning use an agent's policy and a world model to identify its goal, our result uses an agent's policy and its goal to identify a world model

1. Introduction

A hallmark of human intelligence is the ability to perform novel tasks with minimal supervision, formalised by few-shot and zero-shot learning (Lake et al., 2017). With the emergence of these capabilities in language models (Brown et al., 2020), focus has shifted to developing general agents—systems capable of performing long horizon goal-oriented tasks in complex, real-world environments (Yao et al., 2022; Hao et al., 2023). In humans this kind of flexible goal-directed behaviour relies heavily on rich mental representations of the world, i.e. world models (Johnson-Laird, 1983; Ha & Schmidhuber, 2018), which are used to set abstract goals beyond immediate sensory inputs (Locke & Latham, 2013), and to deliberatively and proactively plan actions (Bratman, 1987). Whether world models are necessary for achieving human-level AI has long been debated, pitting the challenges of learning models against the potential benefits they confer (Huang, 2020).

Explicitly model-based agents have achieved impressive performance across many tasks and domains (Hafner et al., 2023; Wang et al., 2023; LeCun, 2022; Raad et al., 2024; Schrittwieser et al., 2020), and having direct access to the agent's world model has benefits like being able to apply formal planning methods (Sutton, 2018), predicting the agent's behaviour in safety-critical domains (Amodei et al., 2016; Dalrymple et al., 2024), reducing sample complexity (Hafner et al., 2019) and supporting transfer learning (Chua et al., 2018; Zhu et al., 2023). However, learning accurate models of real-world systems can be extremely challenging (Dulac-Arnold et al., 2019), and the performance of model-based agents is fundamentally limited by their model's fidelity.

In "Intelligence without representation", Brooks famously proposed that *the world is its own best model*, and that all intelligent behaviours can emerge in model-free agents interacting through action-perception loops, without needing to learn explicit representations of the world (Brooks, 1991). This view has largely been borne out by the development of model-free agents capable of generalizing across a wide range of tasks and environments (Reed et al., 2022; Brohan et al., 2023; Driess et al., 2023; Black et al., 2024). This model-free paradigm aims to achieve truly general agents while side-stepping the challenges inherent in learning a world model. However, there is mounting evidence that model-free agents may in fact learn *implicit* world models

[1]Google DeepMind. Correspondence to: Jonathan Richens <jonrichens@google.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

Food for thought:

> any agent that satisfies a regret bound for a suf- ficiently diverse set of simple goal-directed tasks must have learned an accurate predictive model of its environment

4 Successor Representations

A world model is one answer to "what do we need to evaluate a policy?". A **successor representation** (SR; Dayan 1993) is another, cheaper answer. Starting from the value function and pulling the reward out of the expectation,

$$V^\pi(s_t) \;=\; \sum_{t'\geq t} \gamma^{t'-t}\,\mathbb{E}[r(s_{t'})] \;=\; \sum_{s} \underbrace{\Big(\sum_{t'\geq t} \gamma^{t'-t}\, p(s_{t'}=s \mid s_t)\Big)}_{\propto \text{ future-state occupancy } \mu^\pi(s_t)} r(s) \;=\; \tfrac{1}{1-\gamma}\,\mu^\pi(s_t)^\top \vec{r}.$$

Value factors into *where you go* (policy + dynamics) times *what it pays* (reward). The SR $\mu^\pi(s_t)$ stores the "where you go" part and is reward-agnostic: change the reward vector and you re-evaluate any π for free.

μ^π satisfies a Bellman recursion, so it can be learned with TD — the same algorithmic machinery as

V^π, but vectorized over target states i:

$$\mu_i^\pi(s_t) \;=\; (1-\gamma)\,\delta(s_t = i) \;+\; \gamma\,\mathbb{E}_{a_t\sim\pi,\, s_{t+1}\sim P}\big[\mu_i^\pi(s_{t+1})\big].$$

Successor features. If the reward is (approximately) linear in features,

$$r(s) \;\approx\; \phi(s)^\top \mathbf{w},$$

then we only need the **successor features** $\psi_j^\pi(s_t) = \sum_s \mu_s^\pi(s_t)\,\phi_j(s)$, whose dimension is the number of features rather than the number of states (Barreto et al., 2017). This allows you to recover Q (if you know the reward function):

$$\psi^\pi(s_t, a_t) \;=\; \phi(s_t) + \gamma\,\mathbb{E}\big[\psi^\pi(s_{t+1}, a_{t+1})\big], \qquad Q^\pi(s_t, a_t) \;\approx\; \psi^\pi(s_t, a_t)^\top \mathbf{w}.$$

Why would you want this? Some additional work on exploration that might help you figure out how to visit future high-reward states (or underexplored states). But also, you may want to quickly re-evaluate policies under different reward functions. And more!

5 Hierarchical RL

Moving beyond modeling, you may want to try to simplify the problem by working at different levels of abstraction. Long-horizon tasks have natural *temporal abstractions*. So go to class might be: open the door, walk to class, etc. Each of these can be learned separately and chained together. **Hierarchical RL** (HRL) tackles this.

5.1 The options framework

The classical formalization is the **options framework** (Sutton et al., 1999). An option is a "mini-policy" or skill, a triple

$$o \;=\; \Big(\; \underbrace{I_o}_{\text{initiation set}} \;,\; \underbrace{\pi_o(a \mid s)}_{\text{option policy}} \;,\; \underbrace{\beta_o(s)}_{\text{termination prob.}} \;\Big).$$

You can invoke option o from any state $s \in I_o$; you then roll out π_o until the termination condition β_o fires. This turns the MDP into a **semi-MDP** over the augmented action space $\mathcal{O} \cup \mathcal{A}$ (options plus primitive actions). You can then use all of the same machinery just operating at the higher level action space (over options), as well as learning within an option.

Why hierarchies might help.

- **Shorter effective horizon.** If options run for h steps on average, the high-level Q-function only has to back up values over T/h decisions instead of T, which dramatically shrinks credit assignment.

- **Structured exploration.** A single option can move the agent across a large chunk of state space; ϵ-greedy over options covers more ground than ϵ-greedy over primitives.
- **Reuse.** Good options transfer across tasks that share subgoals.

You can also leverage all the same ideas, like actor-critic. For example via Option-Critic (Bacon, Harb, and Precup, 2016).

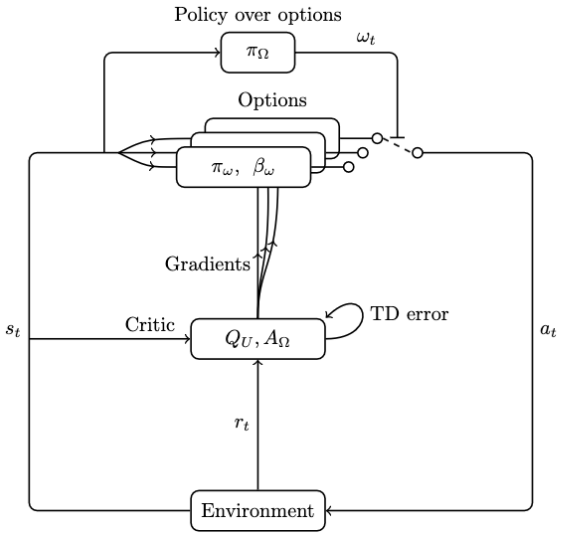


Figure 1: Diagram of the option-critic architecture. The option execution model is depicted by a *switch* $\perp$ over the *contacts* $-\circ$. A new option is selected according to π_Ω only when the current option terminates.

5.2 Hierarchy for LLM agents: ArCHer

The same idea shows up in contemporary work on training LLM agents over multi-turn interactions (Zhou et al., 2024). A conversation has two natural time scales: *utterances* (one message) and *tokens* (one subword). If you want to do GRPO or PPO on dialogues or agent traces, it's not clear how you should do credit assignment at the turn or token level.

Zhou et al. (2024) run a two-level actor–critic, not unlike options (though using a different mechanism):

- **High level (utterance MDP).** A value/advantage head $V(h_t)$ and $A(h_t, u_t)$ is trained over utterance histories. The "action" is a full utterance u_t; the environment transitions one full turn at a time.
- **Low level (token MDP).** The LLM itself is the token-level policy $\pi_\theta(\text{token}_i \mid \text{prefix})$. It is updated with a token-level policy gradient *inside* each utterance, using the high-level advantage $A(h_t, u_t)$ as a per-utterance reward signal (so every token in u_t receives the same utterance-level credit).

Discussion

Do you think we need explicit hierarchy? Maybe neural networks and scale will just handle all of this for free? That's one of the ongoing debates.

6 Multi-agent RL

So far we've treated the world as a single-agent MDP (plus, in assistance games, one cooperating human). When multiple learning agents share the environment, the right formalism is a **Markov game** (or stochastic game) generalizing the MDP to N agents each with their own policy π_i and reward R_i:

$$\mathcal{G} = \Big\langle \underbrace{\mathcal{S}}_{\text{states}}, \underbrace{\mathcal{A}_1, \ldots, \mathcal{A}_N}_{\text{per-agent actions}}, \underbrace{P(s_{t+1} \mid s_t, a_t^1, \ldots, a_t^N)}_{\text{joint transition}}, \underbrace{R_1, \ldots, R_N}_{\text{per-agent rewards}}, \gamma \Big\rangle.$$

There are different regimes depending on how R_i relate: *cooperative* ($R_1 = \cdots = R_N$), *zero-sum competitive* ($\sum_i R_i = 0$), and *general-sum/mixed* (everything else — social dilemmas, negotiation).

If you don't account for agent interactions, this can lead to a bunch of downstream challenges and open research questions.

- **Non-stationarity.** From one agent's perspective the environment dynamics are shifting becuase other agents are also adapting. This creates instability in learning.
- **Credit assignment.** In cooperative settings only the team reward is observed; attributing it to individual actions is hard, and exploratory noise from *any* teammate can mislead an agent's gradient. Specialized methods (COMA (Foerster et al., 2018), value-decomposition networks, QMIX (Rashid et al., 2018)) try to address this.
- **Partial observability.** Agents rarely see the full state or each others' internal state/intentions, meaning we're operating in a POMDP setting most of the time.
- **Equilibrium selection.** Even if everyone converges to a Nash equilibrium, there are typically many, and agents who independently pick different ones coordinate badly. Or they might not try to play for a Nash Equilbrium and then everyone is worse off.
- **Self-play pathologies.** In competitive settings, policies trained against themselves can overfit to idiosyncratic opponents, cycle (rock-paper-scissors dynamics), or exploit brittleness that no real opponent would exhibit.

Open research questions.

- **Social dilemmas.** When individual and collective incentives diverge (iterated prisoner's dilemma, tragedy-of-the-commons variants), does RL find cooperation or defection? Under what mechanism-design or reward-shaping interventions do populations *learn* to cooperate (Leibo et al., 2017)?
- **Emergent communication.** Can agents invent a language to coordinate when given a cheap-talk channel, and does the resulting protocol generalize or compose?

- **Opponent modeling / theory of mind.** How much should each agent model the others' beliefs and learning dynamics, and how does this interact with non-stationarity? This also is tied to assistance games!
- **Robustness and safety.** A policy optimal against a training population can fail against held-out opponents, humans, or adversarially-crafted partners. How much influence should you absorb from external agents? How much should you trust agents?

7 Wrap Up

It's clear (at least to me) that RL—the idea of learning from experience—is a big part of the future for AI. There are many challenges that will continue to pervade this, from reward specification to preventing reward hacking to efficient exploration. But it is also clear how things can go wrong. Encourage all of you to think about the trajectory of where we're going with large scaling of RL and its impacts on society.

References

Hadfield-Menell, D., Dragan, A., Abbeel, P., and Russell, S. (2016). Cooperative inverse reinforcement learning. In *NeurIPS*, volume 29.

Laidlaw, C., Bronstein, E., Guo, T., Feng, D., Berglund, L., Svegliato, J., Russell, S., and Dragan, A. (2025). AssistanceZero: Scalably solving assistance games. In *ICML*.

Shah, R., Freire, P., Alex, N., Freedman, R., Krasheninnikov, D., Chan, L., Dennis, M. D., Abbeel, P., Dragan, A., and Russell, S. (2020). Benefits of assistance over reward learning. In *NeurIPS Workshop on Cooperative AI*.

Silver, D., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint*.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018). Counterfactual multi-agent policy gradients. In *AAAI*.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. (2017). Multi-agent reinforcement learning in sequential social dilemmas. In *AAMAS*.

Bacon, P.-L., Harb, J., and Precup, D. (2017). The option-critic architecture. In *AAAI*.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211.

Zhou, Y., Zanette, A., Pan, J., Levine, S., and Kumar, A. (2024). ArCHer: Training language model agents via hierarchical multi-turn RL. In *ICML*.

Dayan, P. (1993). Improving generalisation for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. (2017). Successor features for transfer in reinforcement learning. In *NeurIPS*.

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2(4):160–163.

Ha, D. and Schmidhuber, J. (2018). World models. In *NeurIPS*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning latent dynamics for planning from pixels. In *ICML*.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2020). Dream to control: Learning behaviors by latent imagination. In *ICLR*.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. (2021). Mastering Atari with discrete world models. In *ICLR*.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. (2023). Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.