Invariance in Policy Optimisation and Partial Identifiability in Reward Learning

Joar Skalse[=,*,1,2] Matthew Farrugia-Roberts[=,*,3] Stuart Russell[4] Alessandro Abate[1] Adam Gleave[*,5]

[=]equal contribution [*]work completed at CHAI, UC Berkeley [1]CS, Oxford University [2]FHI, Oxford University [3]CIS, University of Melbourne [4]CHAI, UC Berkeley [5]FAR AI, Inc.

Background: Reward Learning

Reinforcement learning (RL) can learn effective behavioural policies given:

- an **environment** (states $\mathcal{S}$, actions $\mathcal{A}$, transition dynamics δ, …), and
- a **reward function** $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$.

It's difficult to manually specify a good reward functions for complex tasks.

Reward learning: *why not learn reward functions from data?*

- Consider a *reward space* $\mathcal{R} = \{ R \mid R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R} \}$, and
- a *data-generating process* $f : \mathcal{R} \to \mathcal{X}$ representing a *data source*.

What data source? Any data sources containing reward-relevant information. Many such data sources have been proposed. Examples:

- **Inverse RL:** Optimal/noisy policies or trajectories.
- **Human feedback:** Optimal/noisy preferences or ratings of trajectories.

After reward learning, you can use your learnt reward function to do RL (among other potential applications of a learnt reward function).

Partial Identifiability

Partial identifiability: For some data sources, *many reward functions generate indistinguishable data*.

That is, for a data generating process $f : \mathcal{R} \to \mathcal{X}$ producing infinite data sets from reward functions:

$$\exists R, R' \in \mathcal{R} \text{ such that } f(R) = f(R').$$

Reward ambiguity: No reward learning algorithm can resolve this fundamental ambiguity (without additional assumptions).

Characterising ambiguity/partial identifiability: We study the extent of this phenomenon for various data sources.

- We formalise a data source's ambiguity by partitioning the reward space into sets of functions that lead to indistinguishable outputs.
- **Invariance partition** of a data-generating process f:

$$\left\{ \{ R' \in \mathcal{R} \mid f(R) = f(R') \} \,\middle|\, R \in \mathcal{R} \right\}.$$

- We describe the partition by enumerating the *invariances of* f.

Tolerating Partial Identifiability

Invariance in applications: For some applications of learnt reward functions, *many reward functions generate the same outcomes*.

That is, for an application computing a function $g : \mathcal{R} \to \mathcal{Y}$ from reward functions to outcomes:

$$\exists R, R' \in \mathcal{R} \text{ such that } g(R) = g(R').$$

Reward ambiguity tolerance: it doesn't matter for this application if a reward learning algorithm can't distinguish such reward functions.

Characterising ambiguity tolerance: We study the extent of this tolerance for various applications.

- We formalise an application's ambiguity tolerance by partitioning the reward space into sets of functions that lead to indistinguishable outcomes.
- **Invariance partition** of an application function g:

$$\left\{ \{ R' \in \mathcal{R} \mid g(R) = g(R') \} \,\middle|\, R \in \mathcal{R} \right\}.$$

- We describe the partition by enumerating the *invariances of* g.

Reward Learning Lattice: Data Sources and Applications in a Fixed Environment

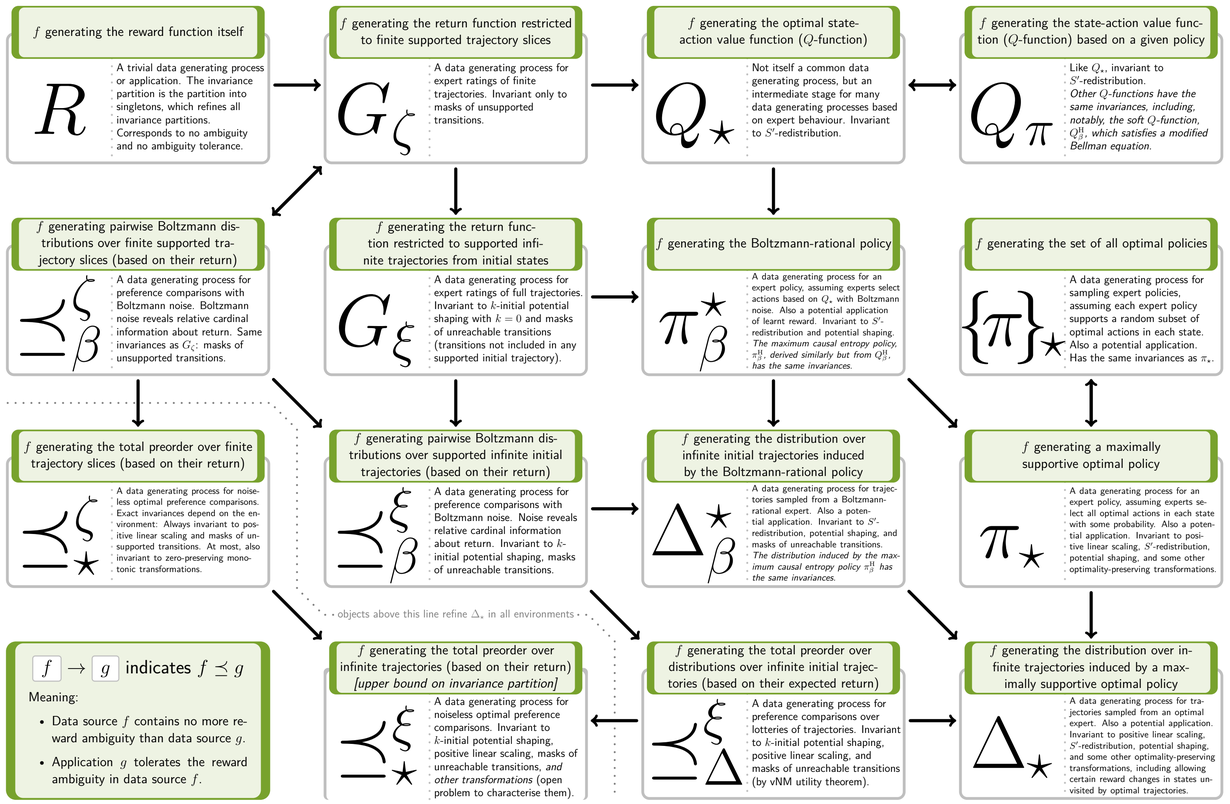


Figure: *Invariance partition refinement* induces a lattice structure on functions from the reward space, based on their invariance partitions (see box below). In this figure, we display the lattice structure for various objects derived from reward functions, given a fixed environment. We describe each invariance partition in terms of several basic sets of invariances (see top right box). In particular, the invariance partition of the object is generated by transformations from the listed sets, and combinations thereof. For full definitions of the derivations and precise statements of the invariances, see the full paper.

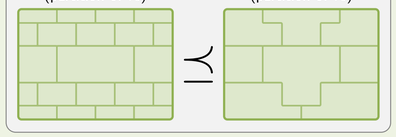
Unified Framework for Comparing Data Sources and Applications

Invariance partition refinement: Let f and g be functions from $\mathcal{R}$, with invariance partitions Π_f and Π_g. Then f *refines* g, written $f \preceq g$, if and only if:

$$\forall P \in \Pi_f, \exists Q \in \Pi_g; P \subseteq Q.$$

In other words, f refines g if all of the partition cells in Π_f are subsets of some partition cell in Π_g.

Example of invariance partition refinement



Comparing two data sources: If f and f' are two data generating processes, then:

$f \preceq f' \iff$ f contains no more reward ambiguity than f'.

Evaluating data sources for applications: If f is a data generating process and g is an application function, then:

$f \preceq g \iff$ g tolerates the reward ambiguity of data source f.

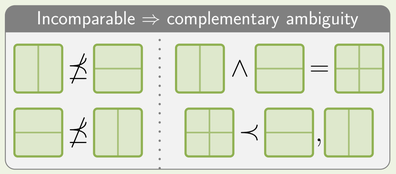
See **Figure** for refinements between common data sources and applications in a fixed environment.

Incomparable ambiguity: Invariance partition refinement is a partial order: Some objects are *incomparable* ($f \not\preceq f'$ and $f' \not\preceq f$).

Incomparable data sources each conflate different reward functions—neither is 'less ambiguous'.

Complementary ambiguity: Incomparable data sources also *distinguish* different reward functions. They contain *complementary reward information*. Combining such sources *always reduces ambiguity!*

Incomparable ⇒ complementary ambiguity



Basic Invariances: Beyond Potential Shaping

Invariance: Given a function from the reward space $f : \mathcal{R} \to \mathcal{X}$, an *invariance* is a reward transformation $t : \mathcal{R} \to \mathcal{R}$ that preserves f, that is, such that

$$\forall R \in \mathcal{R}, f(t(R)) = f(R).$$

We describe the *invariance partition* of f as a set of transformations T such that $f(R) = f(R')$ if and only if $\exists t \in T; R' = t(R)$.

Basic invariances: These sets of transformations generate the invariance partitions of the objects we study (see **Figure**).

1. **Potential shaping:** shifting reward between states along a trajectory.
2. k**-Initial potential shaping:** a special case of potential shaping.
3. S'**-Redistribution:** shifting reward between destination states.
4. **Positive linear scaling:** scaling all rewards by a positive constant.
5. **Zero-preserving monotonic transformations:** preserving the order and sign of transition rewards.
6. **Optimality-preserving transformations:** preserving optimal actions.
7. **Masking transitions:** transforming certain transition rewards freely.

Key Takeaways

Fixed environment: Is ambiguity a problem if we want to learn rewards and then conduct RL in the same environment?

- **Mostly, no:** most data sources refine the outcome of RL (optimal policy trajectory distribution), meaning their ambiguity is tolerable.
- **Exception:** Optimal trajectory comparisons have additional zero-preserving monotonic transformation ambiguity in some environments.

Transfer learning: What if we learn rewards in one environment, and then conduct RL in another?

- Now, ambiguity is a problem! In situations where dynamics differ, reward is essentially unconstrained (assuming destination-dependent rewards).

Caveat: Some important limitations / directions for future work:

- We consider environments with finite state and action spaces.
- Learning using these models assumes that the data-generating process is well specified. However, human agents often aren't (Boltzmann) rational.

More Information

We are poster **24879**

`https://icml.cc/virtual/2023/poster/24879`
(or scan the QR code over there →)

Correspondence to Joar Skalse ⟨joar.skalse@cs.ox.ac.uk⟩ + Matthew Farrugia-Roberts ⟨matt.farrugia@unimelb.edu.au⟩.

Center for Human-Compatible Artificial Intelligence