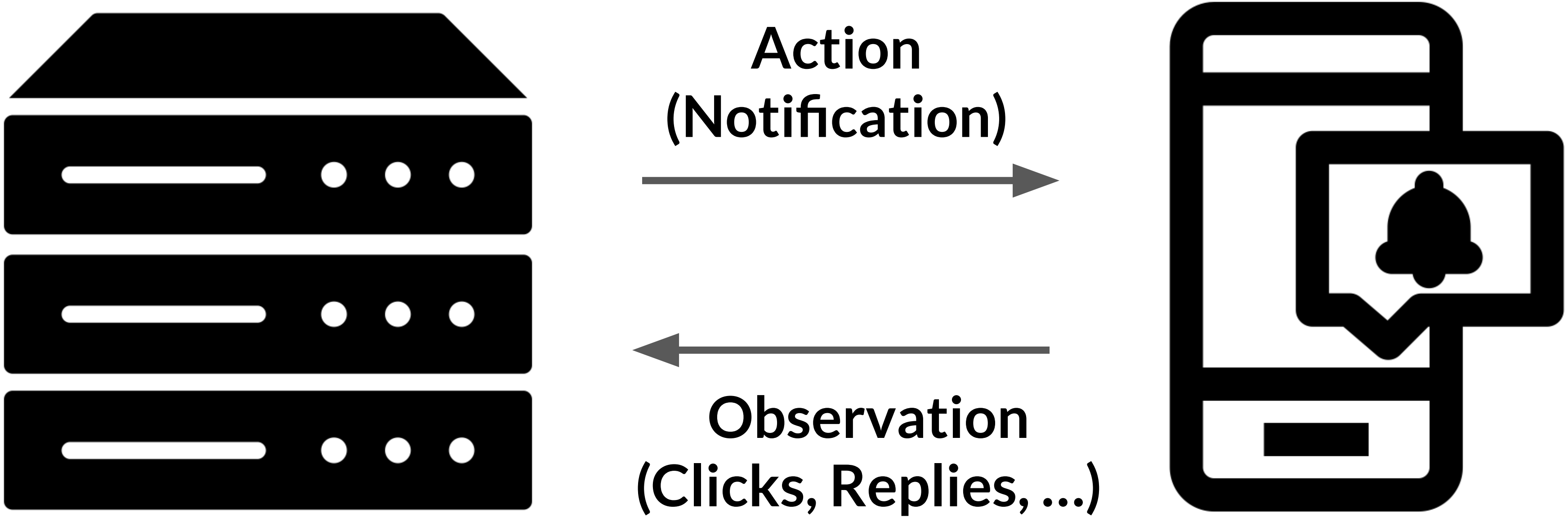
Quantifying Differences in Reward Functions

Adam Gleave, Michael Dennis, Shane Legg, Stuart Russell, Jan Leike

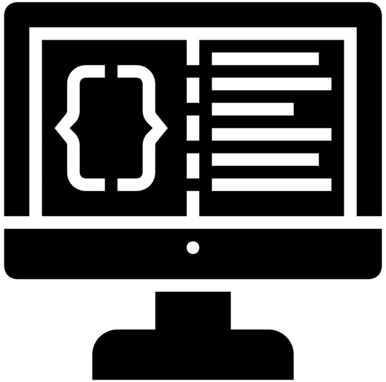
Center for Human-Compatible Artificial Intelligence | DeepMind | BAIR Berkeley Artificial Intelligence Research

Example: Push Notifications

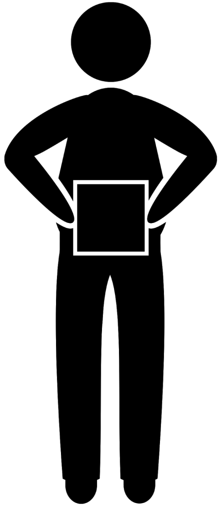


Gauci et al (2018). *Horizon: Facebook's Open Source Applied Reinforcement Learning Platform.* (Section 9.1)

Reward Function Specification

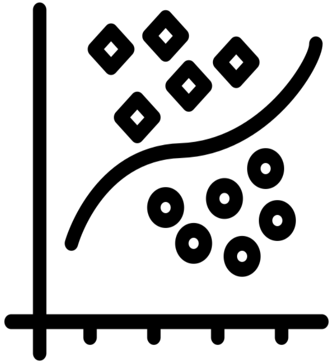


Hand Designed



User Demonstrations

Preference Comparisons

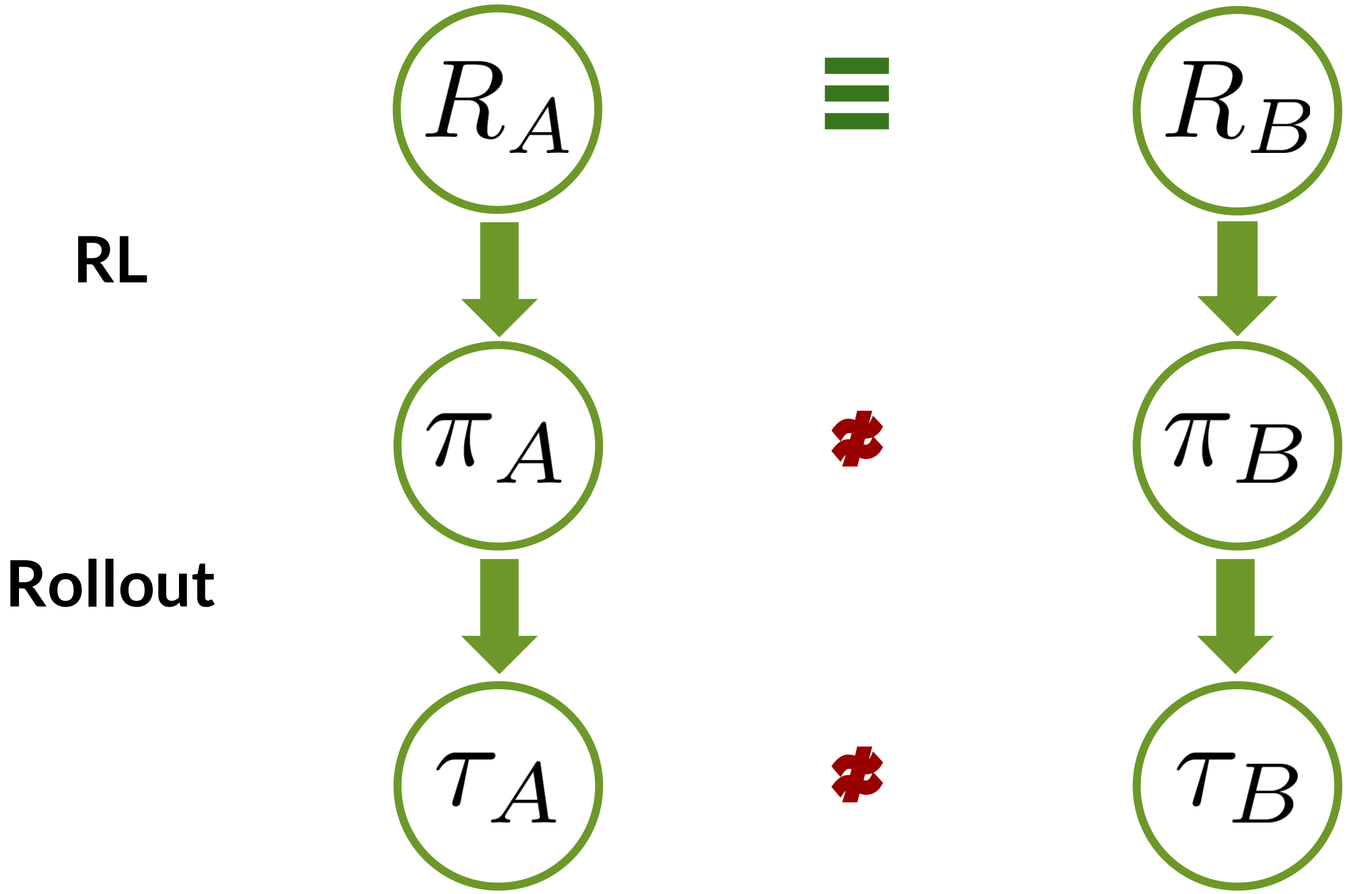


Reward Labels

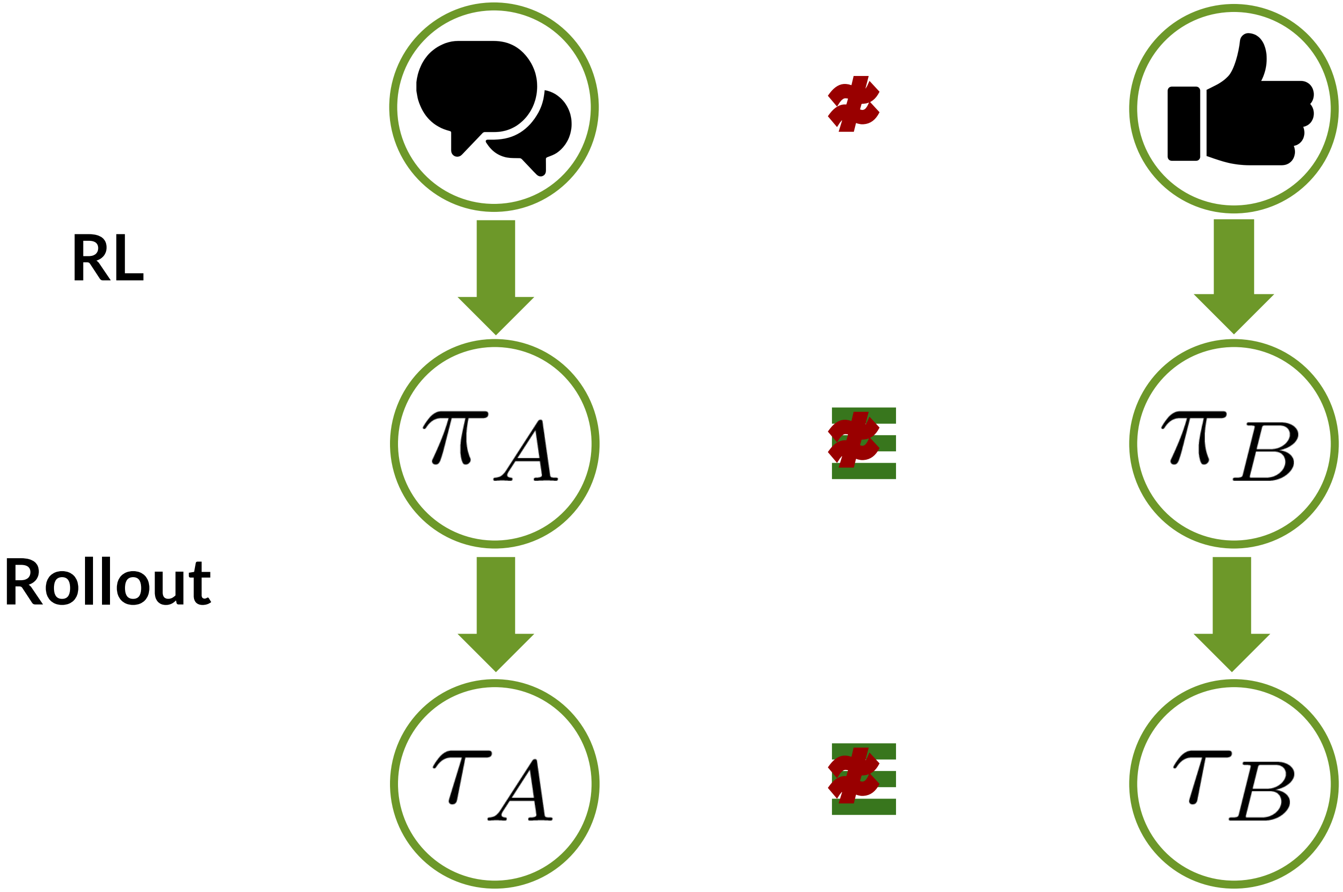
Comparing Reward Functions

R_A **v.s.** R_B

Prior Work: Rollout Evaluation



Prior Work: Rollout Evaluation



We learn rewards,
not policies.

So let's evaluate
rewards, not policies.

Equivalent Rewards Should Be Treated as Equal

Requirement: if two reward functions incentivize the same behaviour, then the comparison should treat them as equal.

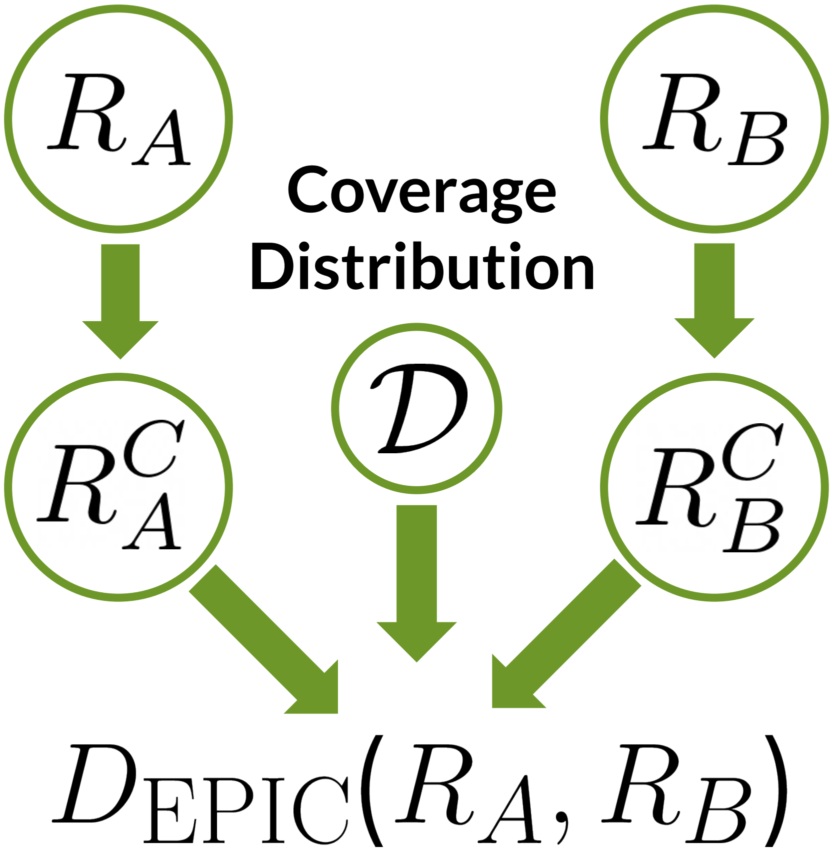
Optimal policy preserving transformations:

- Positive affine: $\lambda R + c \equiv R$ for $\lambda > 0$ and $c \in \mathbb{R}$.
- Potential shaping: moving reward in time.

Equivalent-Policy Invariant Comparison (EPIC)

Canonicalize
(shaping invariant)

Pearson distance
(affine invariant)



Canonicalization

The canonical reward R^C is a shaped and shifted version of R such that the mean reward leaving any state s is zero:

$$\mathbb{E}_{A,S'}\left[R^C(s, A, S')\right] = 0$$

where action A and next state S' are random variables.

R^C can be expressed in terms of expectations on R.

Pearson distance

Pearson correlation coefficient: affine-invariant similarity.

Coverage distribution: $\mathcal{D}$ over transitions.

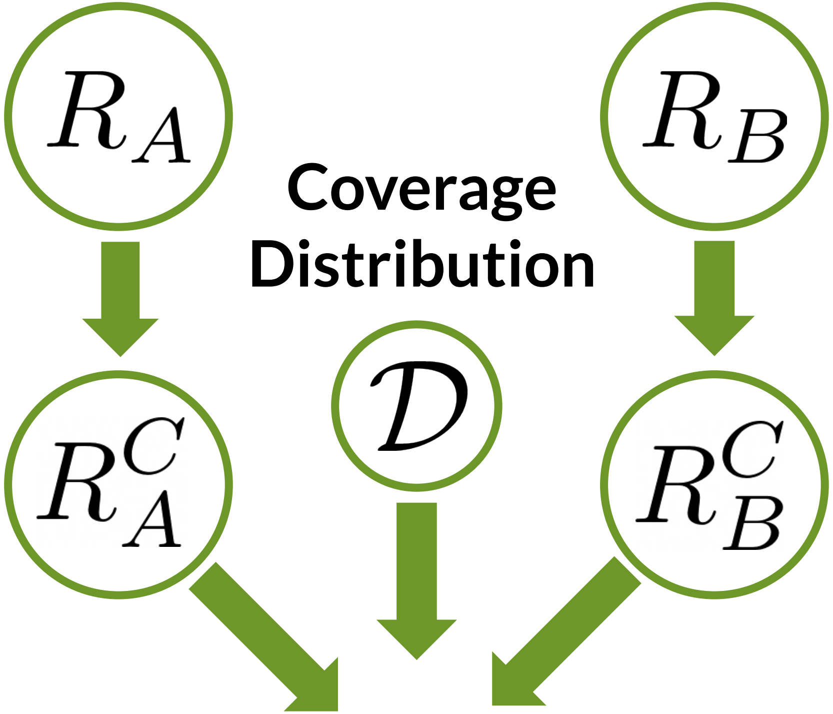
Pearson distance:

$$\frac{1}{\sqrt{2}}\sqrt{1 - \rho(R_A, R_B)}.$$

Equivalent-Policy Invariant Comparison (EPIC)

Canonicalize (shaping invariant)

Pearson distance (affine invariant)



$$D_{\mathrm{EPIC}}(R_A, R_B)$$

EPIC is a distance

Standard properties:

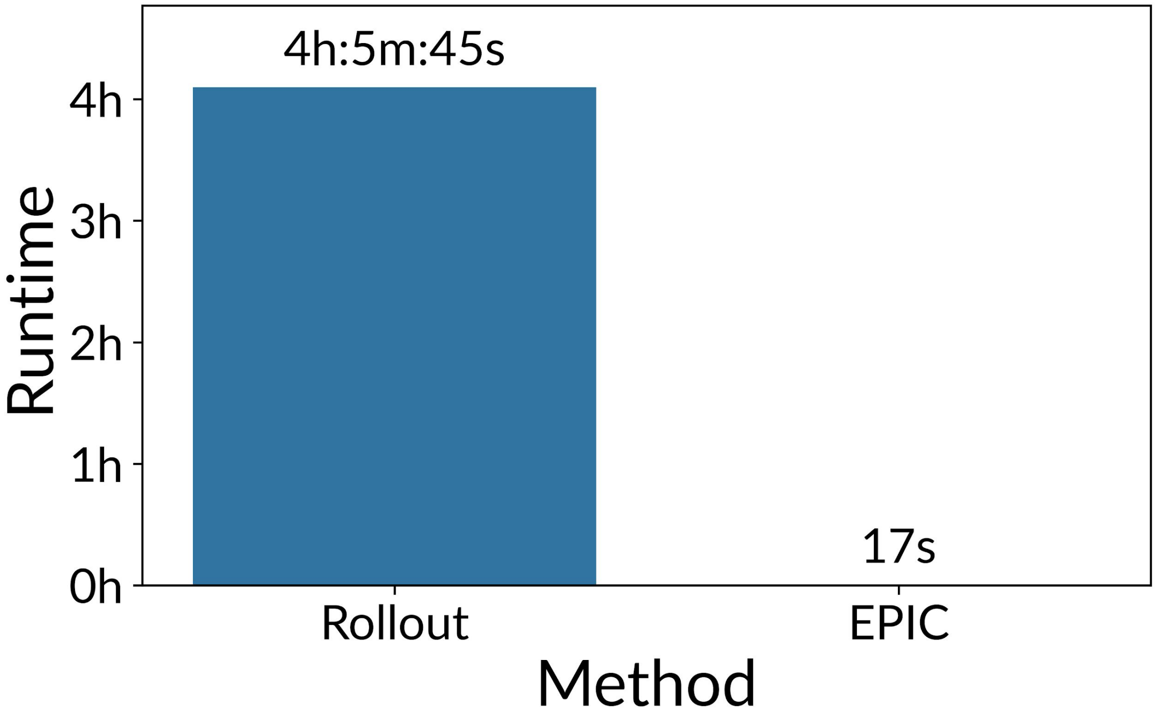
- Symmetry.
- Triangle inequality.

Other properties:

- Bounded on [0,1].
- Zero distance between equivalent rewards.

EPIC is fast

Runtime for reward comparisons in a simple control task:



EPIC is fast

Method	Seeds	Runtime
Rollout	3	4h:5m:45s
EPIC (8192 samples)	3	0h:0m:17s
EPIC (65,536 samples)	30	1h:52m:18s

EPIC is easy to use

EPIC hyperparameters:

- Coverage distribution.
- Number of samples.

Rollout hyperparameters:

- RL algorithm.
- Number of timesteps.
- Batch size.
- Learning rate.
- Entropy coefficient.
- ... and many more!

EPIC is easy to use: choosing number of samples

More samples: higher accuracy, slower speed.

Rule of thumb: increase samples until CI is small enough.

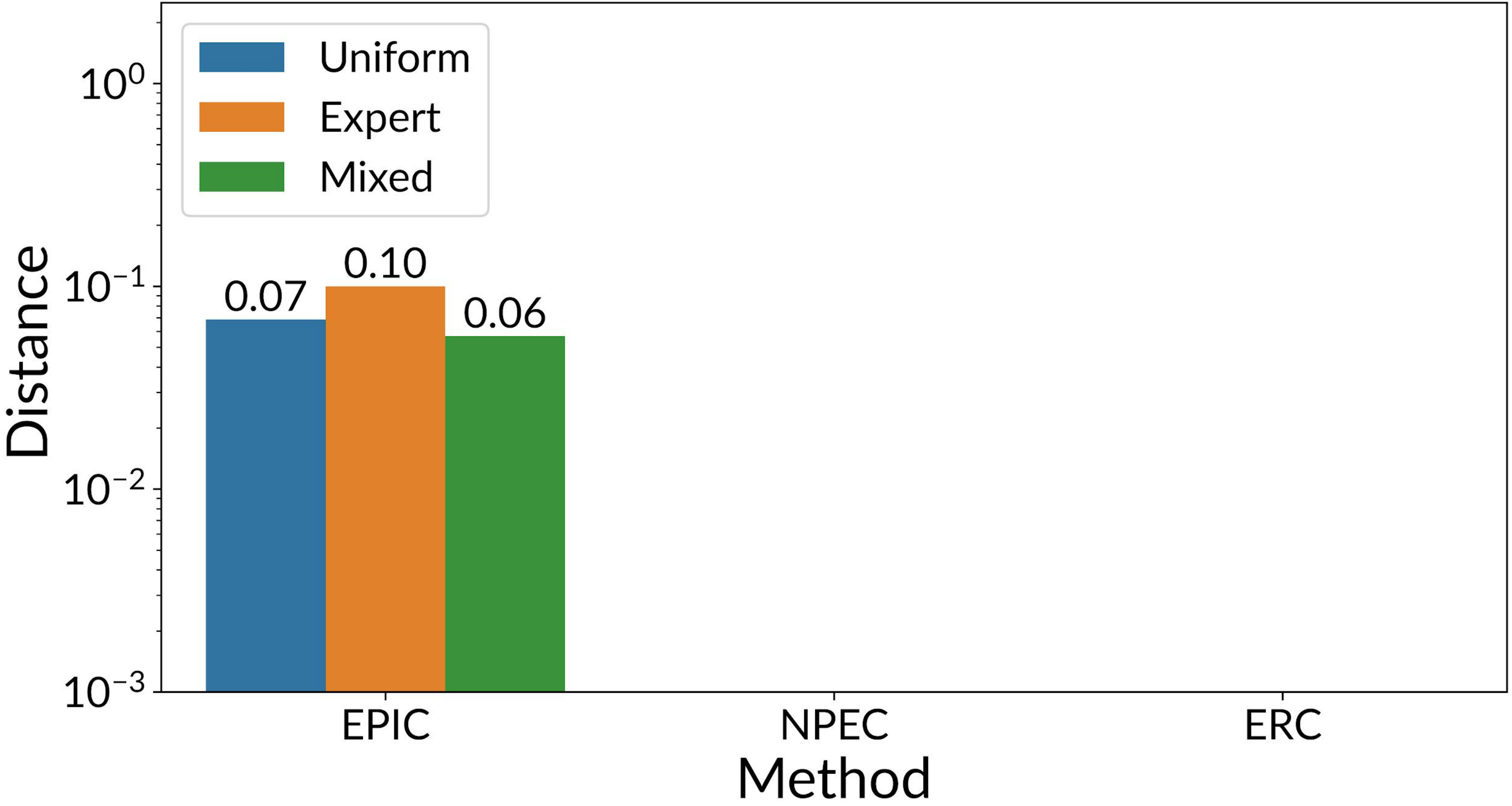
EPIC is easy to use: coverage distribution

Coverage distribution: $\mathcal{D}$ over transitions.

Narrow coverage: overestimates similarity.

Broad coverage: underestimates similarity.

EPIC is easy to use: coverage distribution



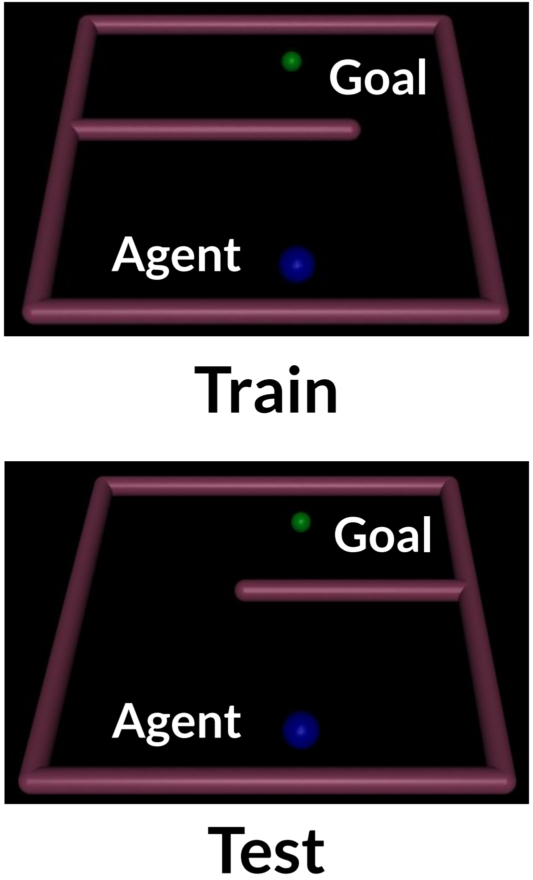
EPIC predicts policy return

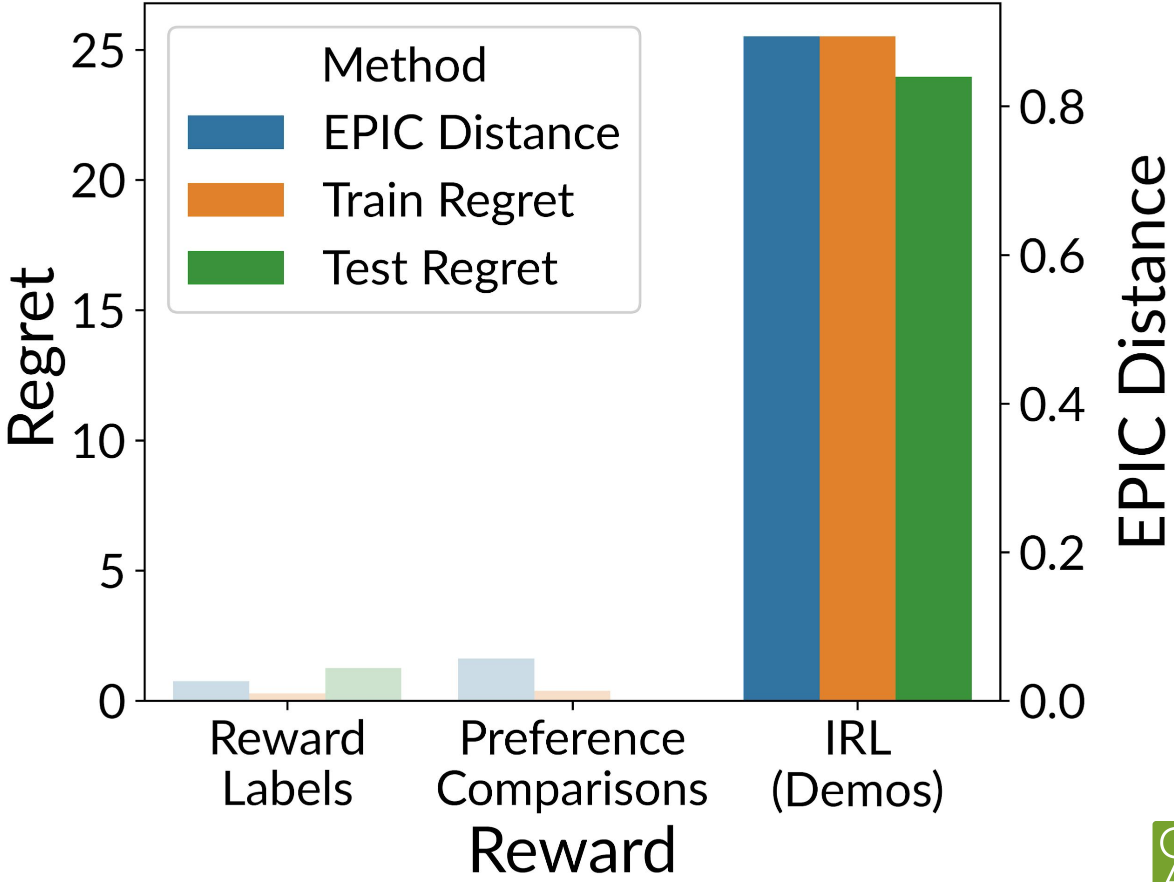
The difference in return G of optimal policies π^*_A and π^*_B for rewards R_A and R_B is bounded by the EPIC distance:

$$G_{R_A}(\pi^*_A) - G_{R_A}(\pi^*_B) \leq K(\mathcal{D})\|R_A\|_2 D_{\text{EPIC}}(R_A, R_B)$$

where $K(\mathcal{D})$ is a constant that depends on the support of the EPIC coverage distribution.

EPIC predicts policy return





Thanks!

Blog: gleave.me/epicblog

Paper: gleave.me/epicpaper

GitHub: gleave.me/epicsrc

Adam Gleave

Michael Dennis

Shane Legg

Jan Leike

Stuart Russell