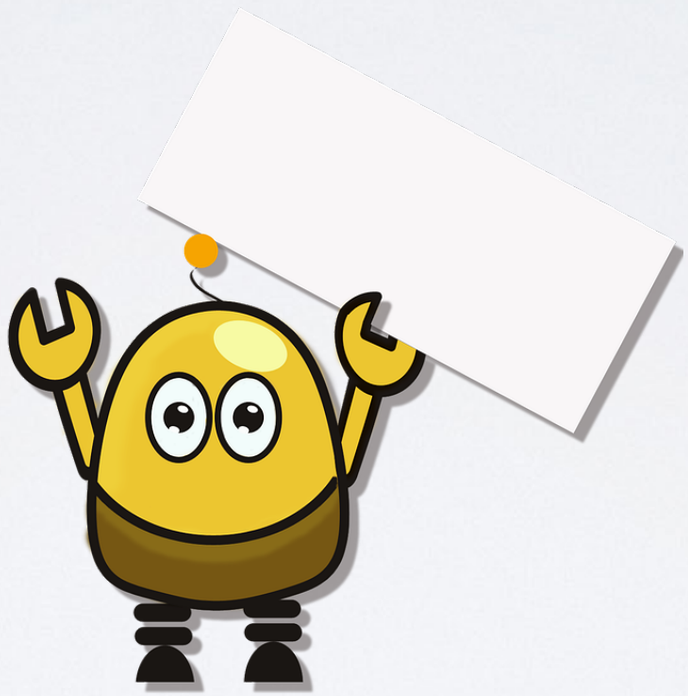
INVERSE REWARD DESIGN

Dylan Hadfield-Menell, Smith Milli, Pieter Abbeel, Stuart Russell, Anca Dragan
University of California, Berkeley



Slides by Anthony Chen

Inverse Reinforcement Learning (Review)

Inverse Reward Design (Intuition)

Inverse Reward Design (Formalism)

INVERSE REINFORCEMENT LEARNING

- Given a Markov Decision Process (MDP) with a reward function, number of ways to obtain a near-optimal or optimal policy.

- However, reward functions can be hard to specify.

- Instead, we can learn a policy by learning from the trajectories (data) generated by an expert.

    - Called inverse reinforcement learning (IRL).

DRIVING

- Many things that go into a reward: safety, comfort, etc.
- Humans learn by watching experts drive.

- In machines, we can learn from data generated by an expert.
- Don't want to exactly copy the expert; doesn't generalize.
- Instead, learn a policy that implicitly maximizes reward from expert data.

NOTATION

- MDP generally defined as a tuple: $M = \langle S, A, T, \gamma, r \rangle$
  - *S*: set of states
  - *A*: set of actions
  - *T*: transition probabilities
  - γ : discount factor
  - *r* : reward function, bounded by 1.
- A MDP without a reward function is called a **world model**, denoted by $\widetilde{M}$

- Assume a feature vector for states: $\phi(s) \in \mathbb{R}^k, \phi : S \to [0, 1]^k$
- Assume an optimal reward function: $r^* = w^* \phi(s), w^* \in \mathbb{R}^k, ||w^*||_1 \leq 1$

- A policy, π, is a mapping from states to a probability over actions.
- The value of a policy can be written as follows:

$$E[V^{\pi}(s_0)] = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi\right]$$

$$= E\left[\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t)|\pi\right]$$

$$= w \cdot E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi\right]$$

- We can then define the **feature expectations** to be

$$\mu(\pi) = E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi\right]$$

and the policy value

$$E[V^{\pi}(s_0)] = w \cdot \mu(\pi)$$

IRL FORMALISM

- Given a world model, $\widetilde{M}$, a predefined feature mapping, ϕ, and a set of trajectories $\{s_0^{(i)}, s_1^{(i)}, \dots\}_{i=1}^{m}$ generated from an expert policy, π_E.
  - Think of this expert policy as returning the optimal value.

- Let μ_E be the expert feature expectation and $\hat{\mu}_E$ be the estimated expert feature expectation.

$$\hat{\mu}_E = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{\infty} \gamma^t \phi\left(s_t^{(i)}\right)$$

- We can find a policy that implicitly maximizes the expert's ***unknown*** reward function via the following optimization:

$$\hat{\pi} = \min_{\pi} ||\mu(\pi) - \hat{\mu}_E||_2$$

Inverse Reinforcement Learning
(Review)

Inverse Reward Design
(Intuition)

Inverse Reward Design
(Formalism)

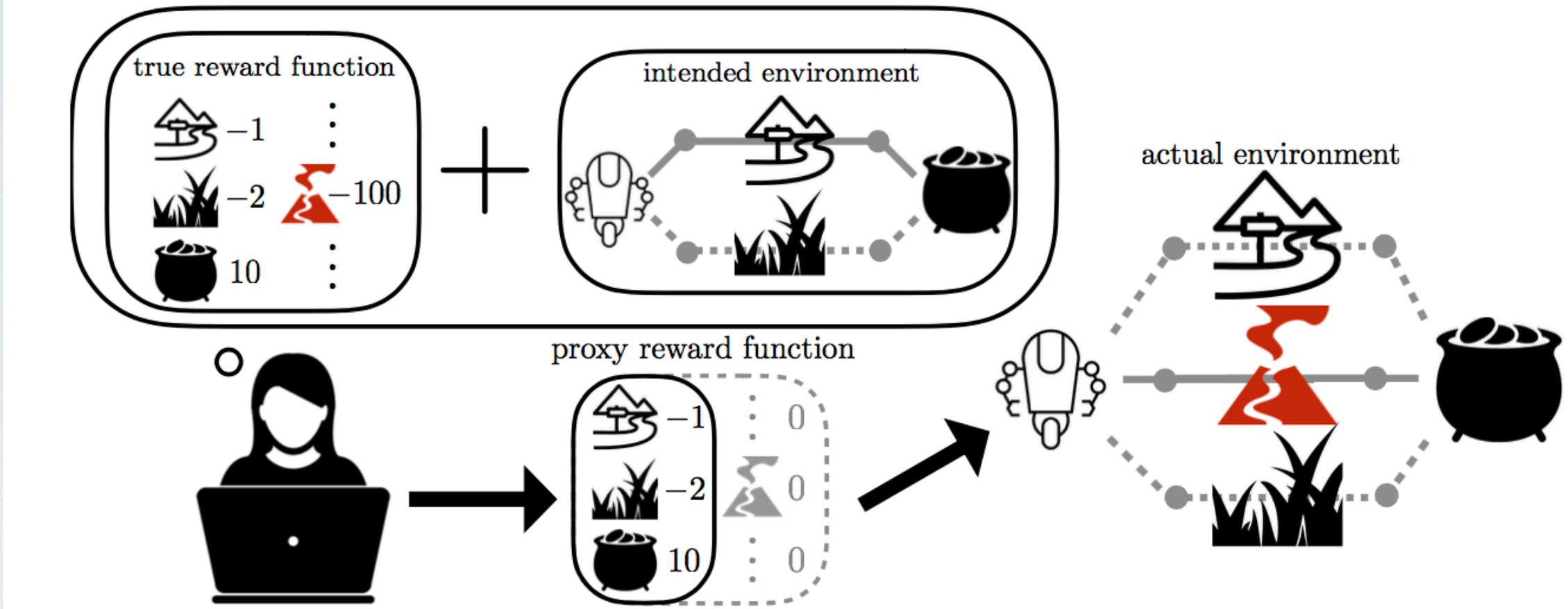
INTUITION

- When a human designs a reward function (known as a proxy reward function), they try and capture as much about the world as possible.

- However, we can't possibly capture all scenarios, leading to **negative side effects** when encountering new states.

- Machines should thus take our reward function as an **guess** at what the true reward function is and assign **uncertainty estimates** to the rewards generated by our reward function.

Image taken from paper



KEY IDEAS

- Proxy reward functions are likely to be true in the context they were defined. Assign high confidence to rewards generated in this case.

- Proxy reward functions can be very wrong in contexts not defined in. Assign high uncertainty to rewards generated in this case.

- Our agent should try and avoid high uncertainty scenarios.

- Formally speaking, inverse reward design tries to capture the true reward function given a proxy reward function.

INVERSE RL V.S. RD

- IRL and IRD both tackle the problem of **value alignment**: the problem of communicating to an agent an good reward.

- Inverse Reinforcement Learning: Hard to design a reward function. Instead given trajectories from expert.

- Inverse Reward Design: More tractable to design reward function. Reward function designed by "expert" but not necessarily complete.

Inverse Reinforcement Learning (Review)

Inverse Reward Design (Intuition)

Inverse Reward Design (Formalism)

BAYESIAN MODELING OF IRD

- In the IRD formalism, we are given a world model, $\widetilde{M}$, and a proxy reward function, $\widetilde{r} = \widetilde{w} \cdot \phi(\xi)$
  - ξ represents a trajectory

- Our goal is to recover the optimal reward function $r^* = w^* \cdot \phi(\xi)$ by recovering the optimal set of weights.

- Let $\pi(\xi|\widetilde{r}, \widetilde{M})$ be the probability of a trajectory under a world model and a proxy reward function.

- We can represent a posterior distribution over the optimal reward function via:

$$P(w^*|\widetilde{w}, \widetilde{M}) = P(\widetilde{w}|w^*, \widetilde{M}) \cdot P(w^*)$$

- Remember, we want to assume that our proxy reward function results in near optimal in the situations in which it was defined.

- We can model the likelihood as follows:

$$P(\widetilde{w}|w^*, \widetilde{M}) \propto exp\Big(\beta \cdot \mathbb{E}\Big[{w^*}^T \phi(\xi)|\xi \sim \pi(\xi|\widetilde{w}, \widetilde{M})\Big]\Big)$$

- This says that in the trajectories generated by our proxy reward function, the reward yielded is optimal with respect to the optimal reward function.

- The posterior can then be written as

$$P(w = w^*|\widetilde{w}, \widetilde{M}) \propto \frac{\exp\left(\beta \cdot w^T \widetilde{\mu}\right)}{\widetilde{Z}(w)} P(w)$$

where $\widetilde{Z}(w) = \int_{\widetilde{w}} \exp\left(\beta w^T \widetilde{\mu}\right) d\widetilde{w}$ is a normalizing constant. Need to approximate.

RISK-AVERSE POLICY

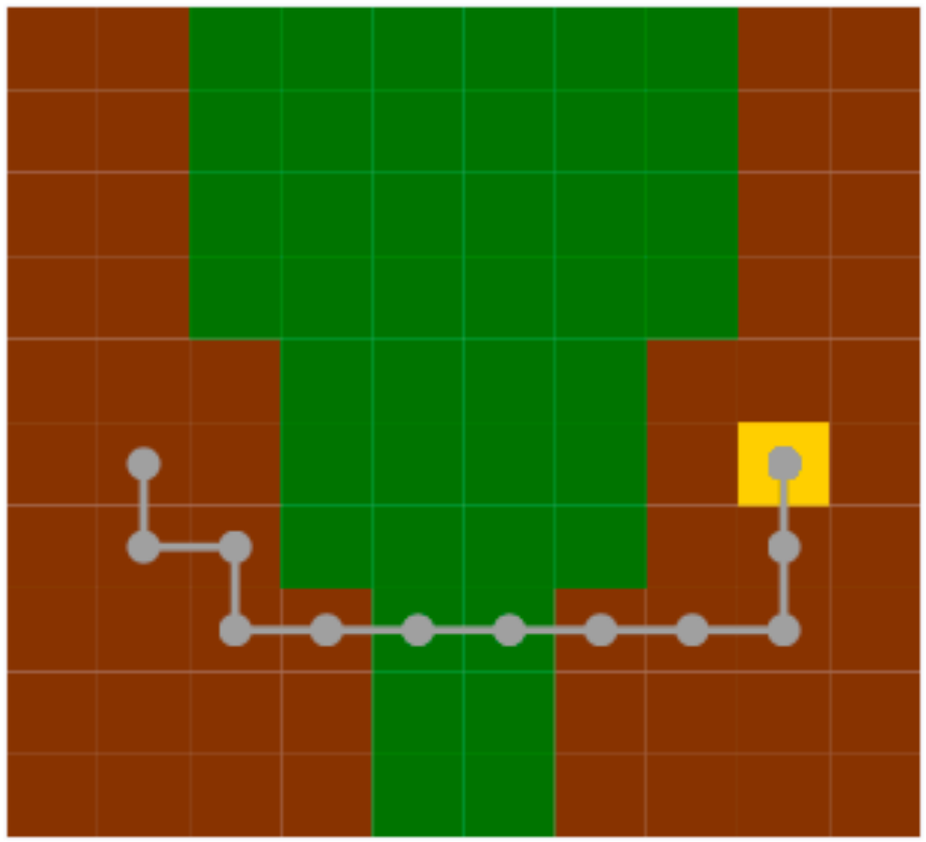
- We can obtain a risk averse policy in the following way:

1. Sample a set of weights $\{\widetilde{w}_i\}$ from our posterior distribution

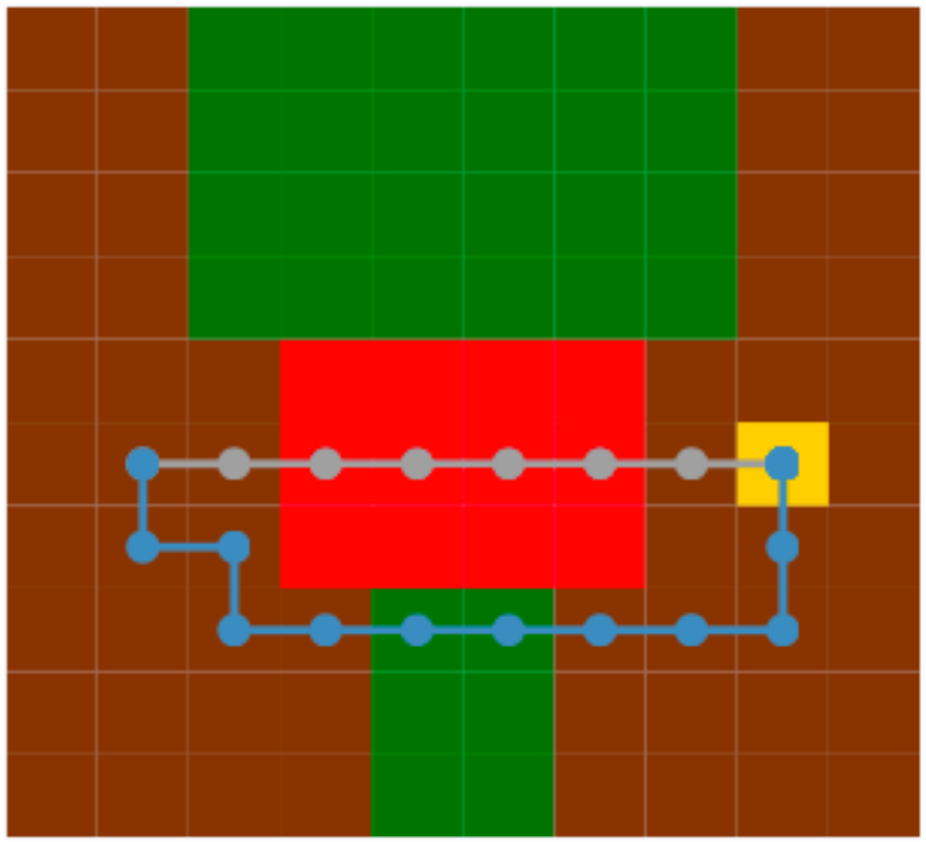
2. Given those weights, pick a trajectory that maximizes the reward in the worst case:

$$\xi^* = \arg\max_{\xi} \min_{w \in w_i} w^T \phi(\xi)$$

EVALUATION: LAVALAND

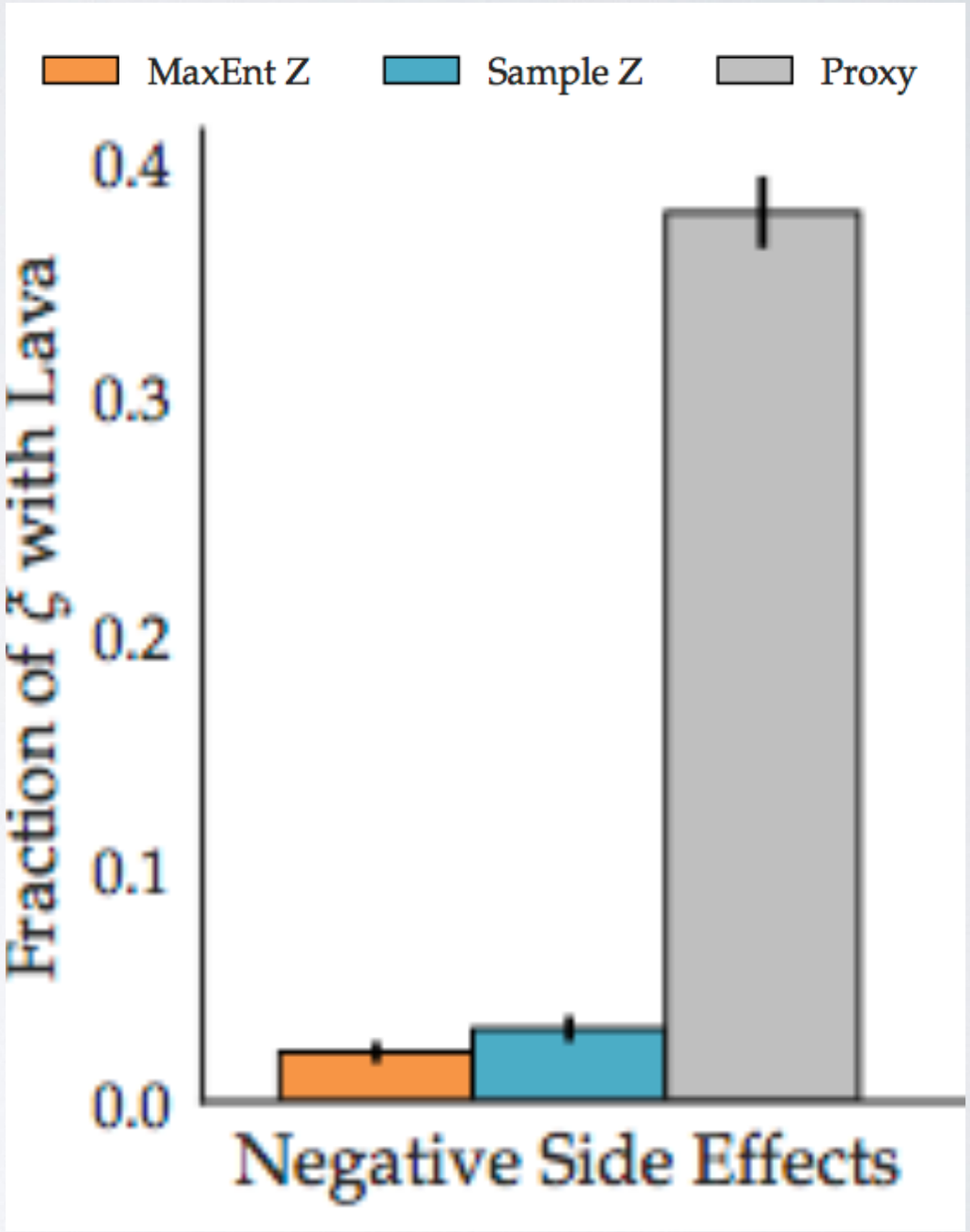


Training



Testing

- Maps initialized at random.
- **Testing**: 5% lava, 66.5% dirt, 28.5% grass
- **Features**: 3 dimensional
- **Proxy reward**: generated random uniformly.
- **Metric**: % of trajectories during test time with lava.



CONCLUSIONS

- In this work, the authors present a method that accounts for uncertainty in the reward function.

- They propose a Bayesian framework to model this uncertainty

- They show empirically on toy examples that their framework can avoid catastrophic events in the face of high uncertainty.