Active Learning for Reward Estimation in Inverse Reinforcement Learning

Manuel Lopes[1], Francisco Melo[2], and Luis Montesano[3]

[1] Instituto de Sistemas e Robótica - Instituto Superior Técnico
Lisboa, Portugal
macl@isr.ist.utl.pt
[2] Carnegie Mellon University
Pittsburgh, PA, USA
fmelo@cs.cmu.edu
[3] Universidad de Zaragoza
Zaragoza, Spain
lmontesa@unizar.es

Contents

Introduction

- **Inverse reinforcement learning** addresses the general problem of recovering a reward function from samples of a policy provided by an expert/demonstrator.
- We introduce **active learning** for **inverse reinforcement learning** to query the demonstrator for samples at specific states, instead of relying only on samples provided at “arbitrary” states.
- Similar accuracy
- Less queries from the expert

Preliminaries - Bayesian Inverse Reinforcement Learning

- State-action pairs:

$$\mathcal{D} = \{(x_1, a_1), (x_2, a_2), \ldots, (x_n, a_n)\}$$

- The likelihood of a pair (x, a) and D:

$$L_r(x, a) = \mathbb{P}\left[(x, a) \mid r\right] = \frac{e^{\eta Q_r^*(x,a)}}{\sum_{b \in A} e^{\eta Q_r^*(x,b)}} \qquad L_r(\mathcal{D}) = \prod_{(x_i, a_i) \in \mathcal{D}} L_r(x_i, a_i)$$

- Get reward function:

$$\mathbb{P}\left[r \mid \mathcal{D}\right] \propto L_r(\mathcal{D}) \mathbb{P}\left[r\right] \qquad r^* = \max_r \mathbb{P}\left[r \mid \mathcal{D}\right]$$

Preliminaries - Two Methods for Bayesian IRL – Gradient-based IRL

- The log-likelihood of the r:

$$\Lambda_r(\mathcal{D}) = \sum_{(x_i,a_i)\in\mathcal{D}} \log(L_r(x_i,a_i))$$

- Compute gradient:

$$[\nabla_r \Lambda_r(\mathcal{D})]_{xa} = \sum_{(x_i,a_i)\in\mathcal{D}} \frac{1}{L_r(x_i,a_i)} \frac{\partial L_r(x_i,a_i)}{\partial r_{xa}}$$

- Update r:

$$\mathbf{r}_{t+1} = \mathbf{r}_t + \alpha_t \nabla_r \Lambda_{r_t}(\mathcal{D})$$

Algorithm - Active Learning for Reward Estimation

- Define :
  - $\mathcal{R}_{xa}(p)$ as the set of reward functions r, s.t. $\pi_r(x,a) = p$
  - Density: $\bar{\mu}_{xa}(p) = \mathbb{P}\left[\pi(x,a) = p \mid \mathcal{D}\right] = \mathbb{P}\left[r \in \mathcal{R}_{xa}(p) \mid \mathcal{D}\right]$
  - Differential entropy: $-\int \bar{\mu}_{xa}(p) \log \bar{\mu}_{xa}(p)\ dp$

- Define a new discrete probability distribution:
  - $I = [0,1] \qquad I_k = \left(\frac{k}{K}, \frac{k+1}{K}\right] \qquad I_0 = \left[0, \frac{1}{k}\right]$
  - $\mu_{xa}(k) = \mathbb{P}\left[\pi(x,a) \in I_k \mid \mathcal{D}\right] = \int_{I_k} \bar{\mu}_{xa}(p)dp, \quad k = 1, \ldots, K$

Algorithm

$$\bar{H}(x) = \frac{1}{|\mathcal{A}|} \sum_{a} H(\mu_{xa}) = -\frac{1}{|\mathcal{A}|} \sum_{a,k} \mu_{xa}(k) \log \mu_{xa}(k)$$

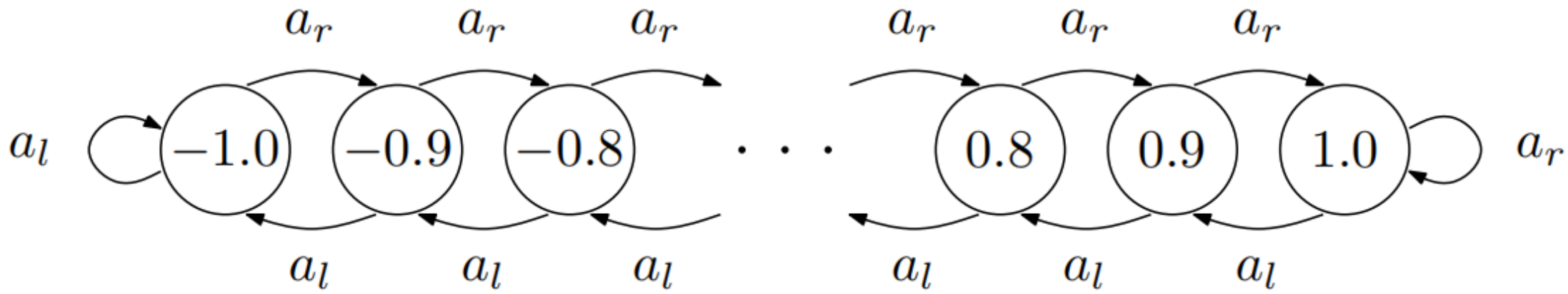
$$x^* = \arg\max_{x \in \mathcal{X}} \bar{H}(x)$$

Algorithm 1. General active IRL algorithm

Require: Initial demo $\mathcal{D}$

1: Estimate $\mathbb{P}[r \mid \mathcal{D}]$ using general MC algorithm
2: **for all** $x \in \mathcal{X}$ **do**
3: Compute $\bar{H}(x)$
4: **end for**
5: Query action for $x^* = \arg\max_x \bar{H}(x)$
6: Add new sample to $\mathcal{D}$
7: Return to 1

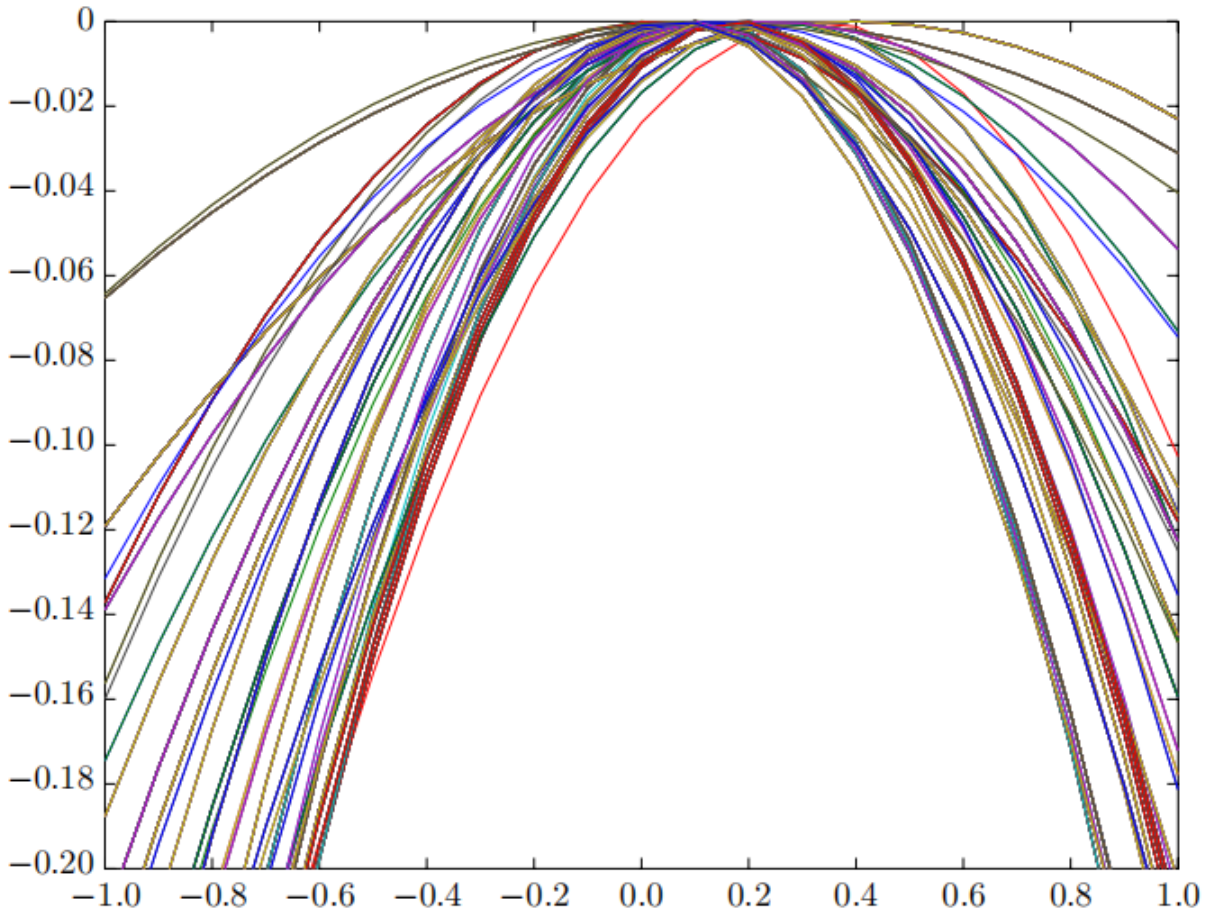
Experiments - Finding the Maximum of a Quadratic Function

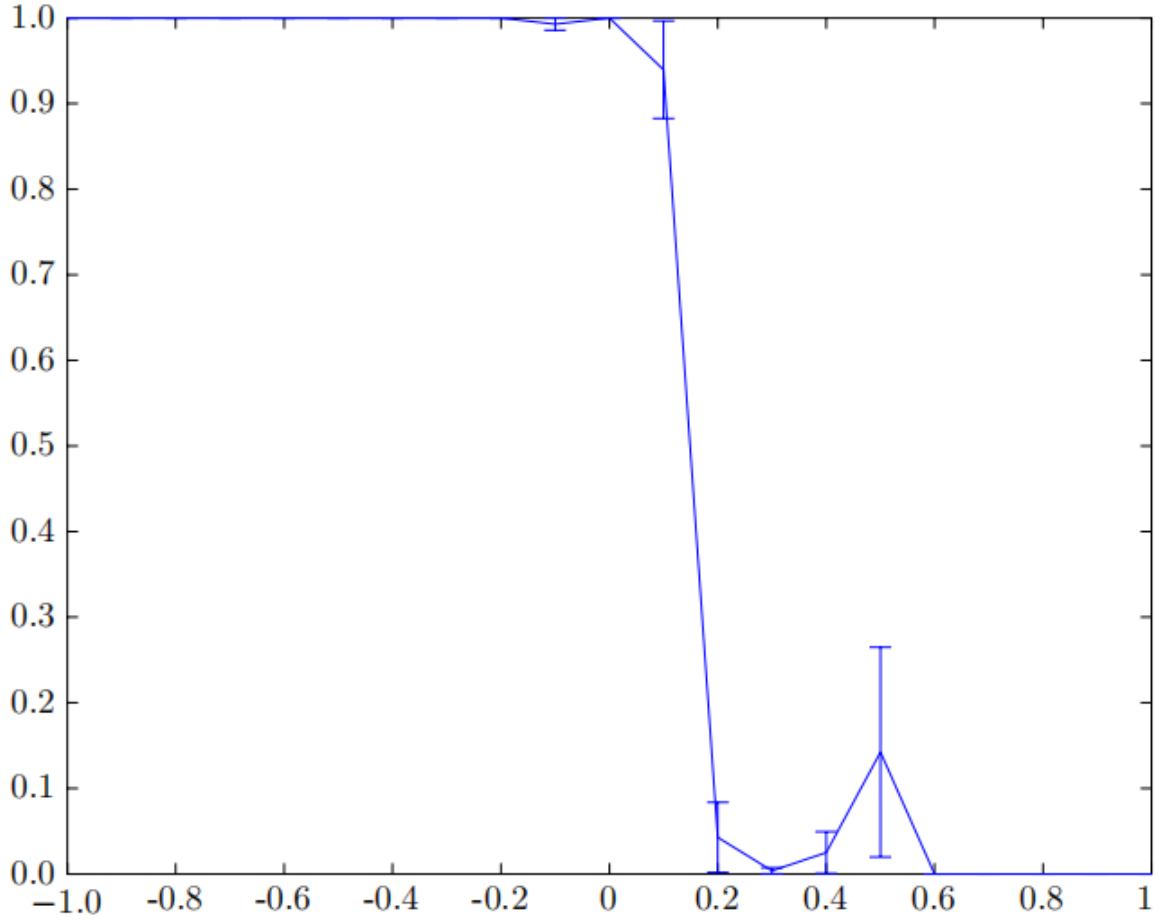


$$r(x) = \theta_1(x - \theta_2)^2 \qquad r(x) = -(x - 0.15)^2$$

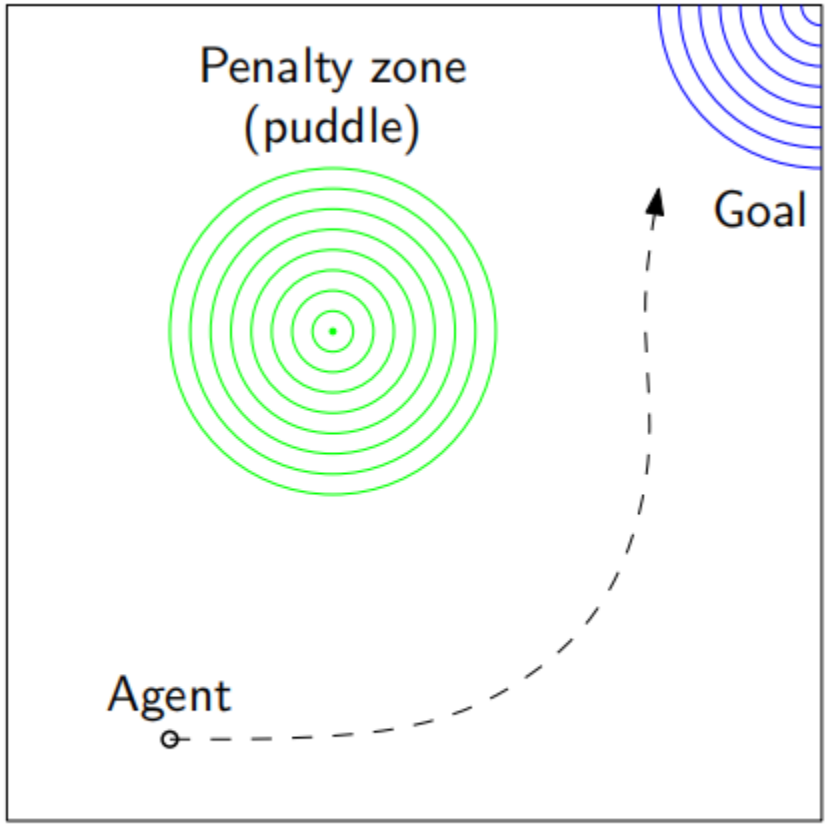
$$\mathcal{D} = \{(-1.0, a_r), (-0.9, a_r), (-0.8, a_r), (0.8, a_l), (0.9, a_l), (1.0, a_l)\}$$

Experiments - Finding the Maximum of a Quadratic Function

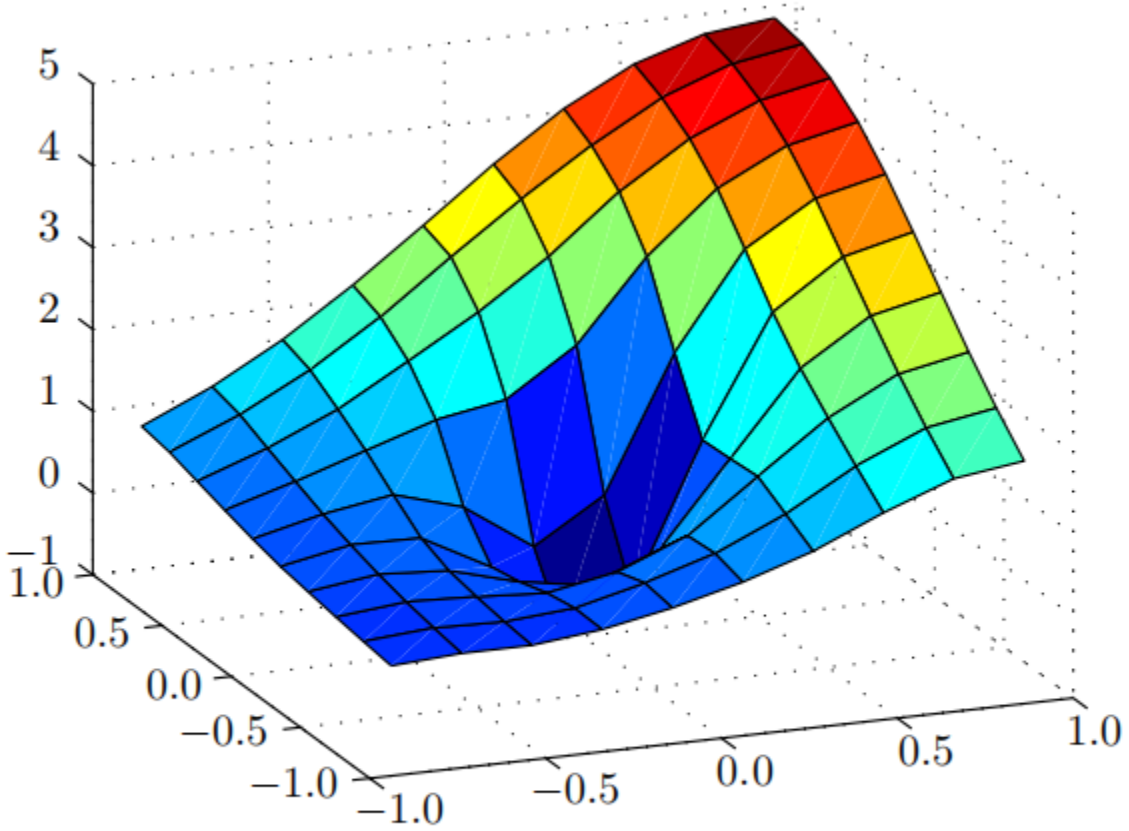




Experiments - Puddle World



(a)



(b)

Experiments - Active sampling vs. Random sampling – Grid-World

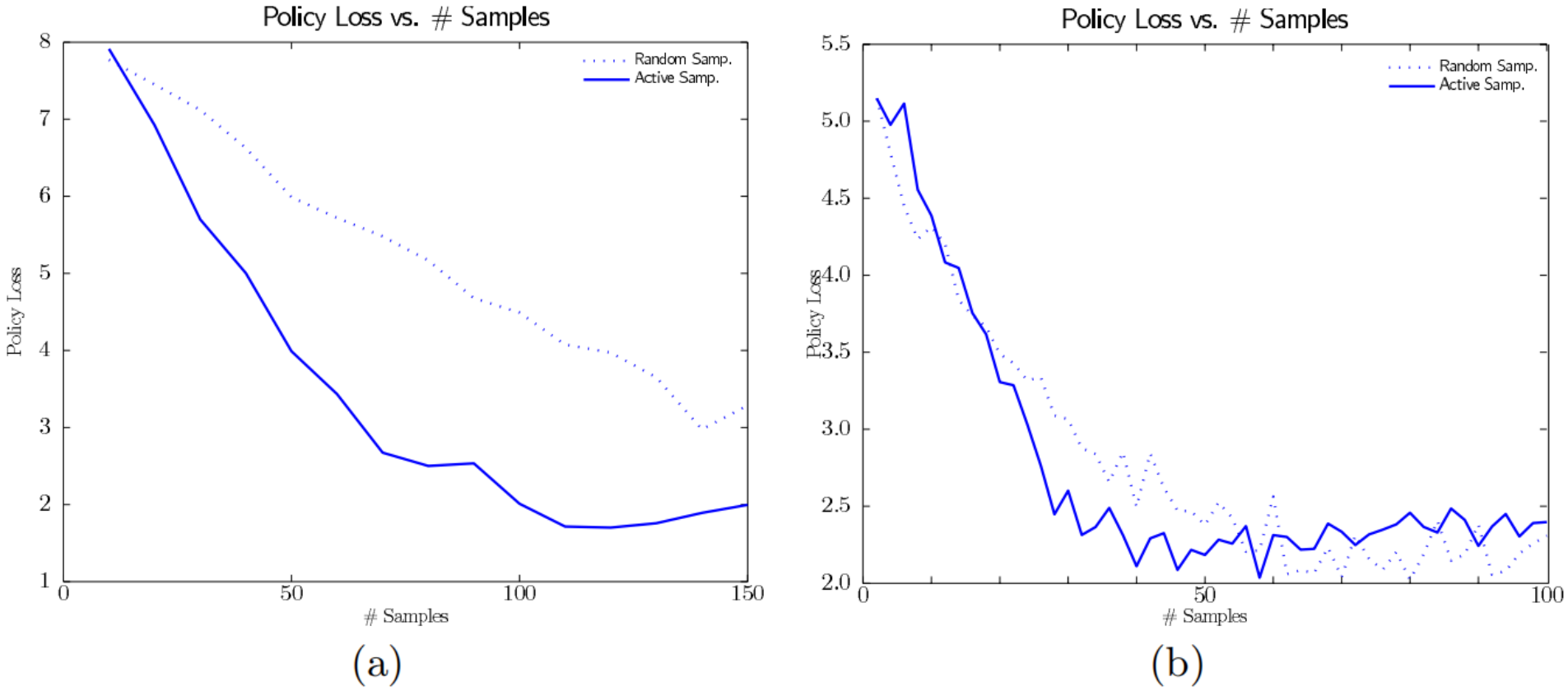


Fig. 6. Performance of Algorithm 2 comparing active sampling vs. random sampling as a function of the demonstration size. (a) Results with parameterized rewards in a 15×15 grid-world. (b) Results with general (non-parameterized) rewards in a 10×10 grid-world.

Conclusions

- We introduced the first **Active Learning** algorithm explicitly designed to reward estimation in **Inverse Reinforcement Learning**.

- Successful application in **Bayesian Inverse reinforcement learning.**