Introduction of

Inverse Reinforcement Learning

곽동현

서울대학교 바이오지능 연구실

Learning from Demonstration

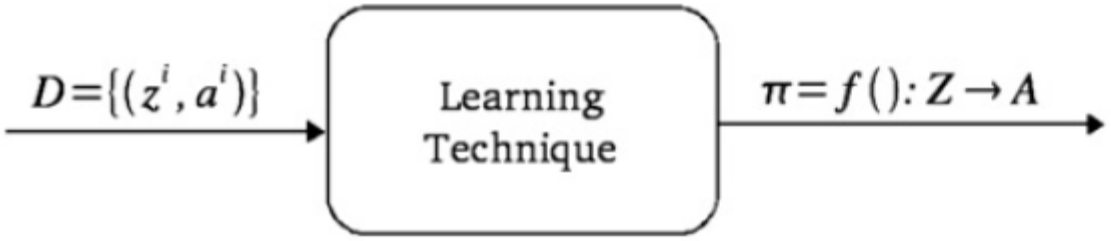
- The goal of LfD is to reproduce the expert's demonstrations with generalized intentions.

- And the reward function is most succinct, robust, transferable to describe the intention/goal of the task.

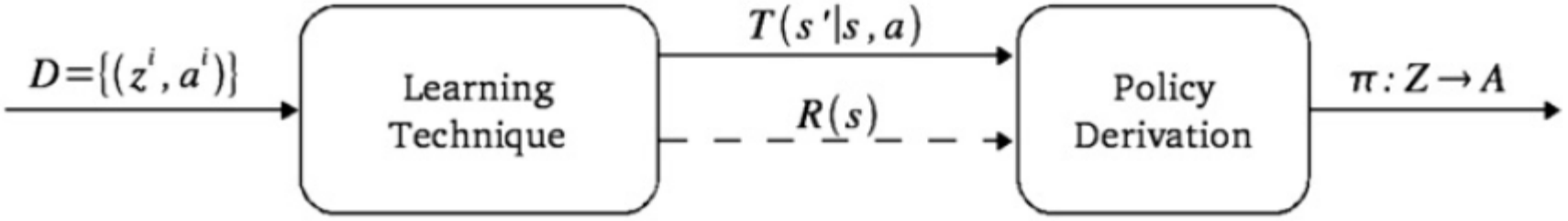
  - Learning from Demonstration: (object의 위치가 계속해서 바뀜)
    https://www.youtube.com/watch?v=lIdioJ4XXsA
  - Model-based Apprenticeship Learning: (1개의 demo로부터 학습)
    https://www.youtube.com/watch?v=1UGQlu9tEvI
  - High-Level Learning from Demonstration: (보다 고수준의 의도 파악)
    https://youtu.be/3nSn1DGek4c?t=2m33s

SL vs IRL

- **Supervised Learning** Approach
  - Directly find the policy with function approximation



- **Inverse Reinforcement Learning** Approach
  - Indirectly find the policy with reward function & RL



Argall, Brenna D., et al. "A survey of robot learning from demonstration."*Robotics and autonomous systems* 57.5 (2009): 469-483.

The limitations of SL

- Generally speaking, the strategy of SL is to simply penalize the behaviors that deviate from the target trajectories, without considerations of the underlying system dynamics. (i.e SL normally minimize the deviation between expert's policy and agent's policy.)

- So too many samples are needed to generalize the intention/goal of demonstrations (or impossible).
- For highway driving, blindly following the expert's trajectory would not work, because the pattern of traffic encountered is different each time.

Argall, Brenna D., et al. "A survey of robot learning from demonstration."*Robotics and autonomous systems* 57.5 (2009): 469-483.

The merits of IRL

- Direct policy representation of expert's intention is rather redundant and usually restricted to certain scenarios, but the robot may be required to do a slightly different task (Atkeson and Schaal, 1997).

- The reward function can succinctly represent the expert's knowledge, and this knowledge is transferable to other scenarios.

Zhifei, Shao, and Er Meng Joo. "A survey of inverse reinforcement learning techniques." *International Journal of Intelligent Computing and Cybernetics* 5.3 (2012): 293-311.

RL vs IRL

- **Reinforcement Learning :** (R → S, A)
  - RL well suits the problems which have predefined reward function. E.g.) Atari game / 바둑

- **Inverse Reinforcement Learning :** (S, A → R)
  - But many real world problems are hard to define the reward function. E.g.) Car driving / Maneuver of drones
  - If we find the proper reward function, we can use RL.

Scalable Reward Learning from Demonstration:
https://www.youtube.com/watch?v=csmr-qpqH00
Autonomous Helicopters Teach Themselves to Fly Stunts:
https://www.youtube.com/watch?v=M-QUkgk3HyE

Inverse Reinforcement Learning

Indirect Problem Solving.

1. Find the optimal Reward Function.
2. Find the optimal Value Function.
3. Find the optimal Policy.

- Original RL can be used in step 2, 3.
- But 2, 3 steps also can be easily solvable with the calculations in step 1.

Big Categories of IRL

- **Max margin based**
  - Maximum Margin Planning(MMP)

- **Probabilistic model based**
  - Bayesian IRL
  - Maximum entropy IRL

Zhifei, Shao, and Er Meng Joo. "A survey of inverse reinforcement learning techniques." *International Journal of Intelligent Computing and Cybernetics* 5.3 (2012): 293-311.

Principles of IRL

1) IRL is fundamentally optimization problem.
2) So derive maximum condition on R(s), but reward function is never unique. (i.e. ill-posed problem)

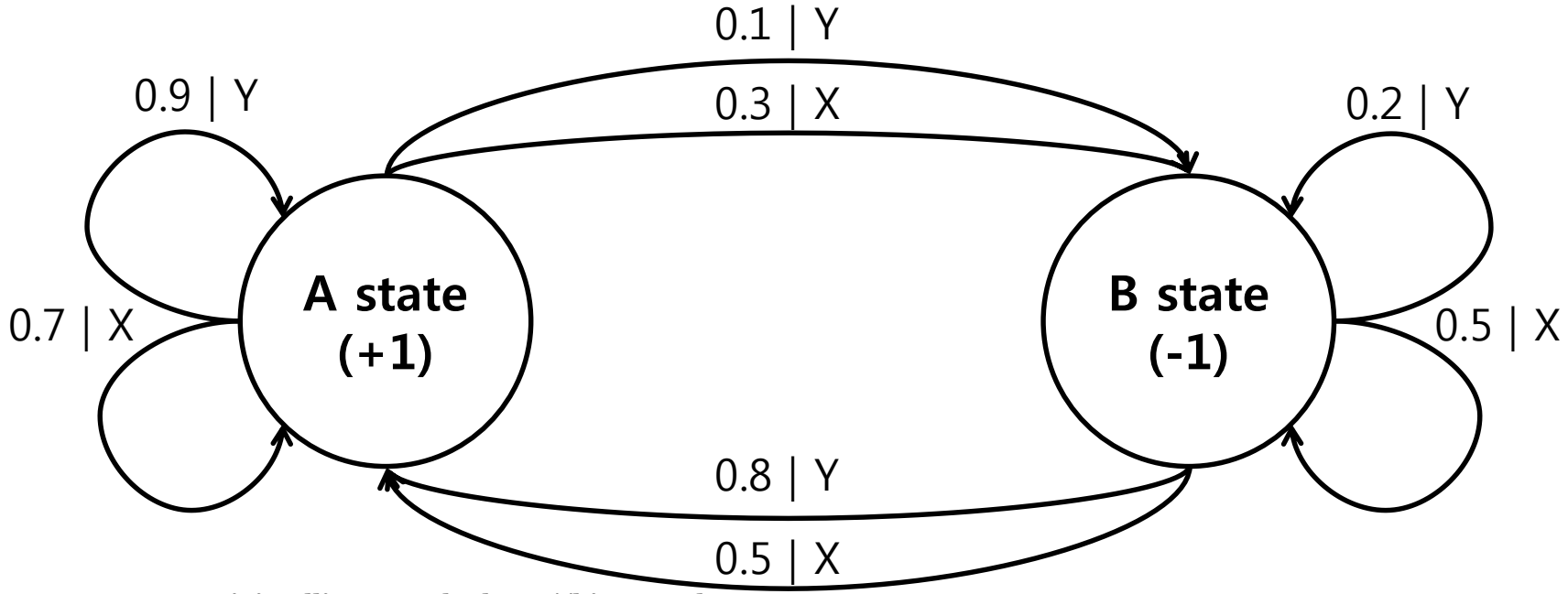
> **Well-posed problem**
>
> From Wikipedia, the free encyclopedia
>
> The mathematical term **well-posed problem** stems from a definition given by J
>
> 1. A solution exists
> 2. The solution is unique
> 3. The solution's behavior changes continuously with the initial conditions.

3) We need more constraints/regularizations/priors to determine the optimal R(s).

Preliminaries

MDP-R

- State space : S = { A , B }
- Action conditional state transition probability : P(S`|S , A)
- ~~Reward function : R(S) = +1 if (S=A), -1 if (S=B)~~
- Action space : A = { X, Y }
- **Expert's Observation = {(s_0, a_0) , (s_1, a_1) , … , (s_n, a_n)}**
- Purpose : find optimal reward function and policy



Notation

- S is a finite set of N **states.**
- $A = \{a_1, \ldots, a_k\}$ is a set of k **actions.**
- $P_{sa}(\cdot)$ are the state **transition probabilities** upon taking action a in state s.
- $\gamma \in [0, 1)$ is the **discount factor.**
- $R : S \mapsto \mathbb{R}$ is the **reinforcement function,** bounded in absolute value by $R_{\max}$.

1. Max margin based : (1) Finite-state MDP with known optimal policy

$$Q^{\pi}(s,a) = R(s) + \gamma \sum_{s'} P_{sa}(s')V^{\pi}(s')$$ → (1) Bellman Equation on Q

$$\pi(s) \in \arg\max_{a \in A} Q^{\pi}(s,a)$$ → (2) Optimal policy

$$\pi(s) \equiv a_1 \; is \; optimal \text{ action}$$

$$a_1 \equiv \pi(s) \in \arg\max_{a \in A} \sum_{s'} P_{sa}(s')V^{\pi}(s') \quad \forall s \in S$$ → (2)에 (1)을 대입

$$\Leftrightarrow \sum_{s'} P_{sa_1}(s')V^{\pi}(s')$$ → 동치.Optimal action을 도입

$$\geq \sum_{s'} P_{sa}(s')V^{\pi}(s') \quad \forall s \in S, a \in A$$ → 모든 action보다 좋음

$$\Leftrightarrow \boldsymbol{P}_{a_1}\boldsymbol{V}^{\pi} \succeq \boldsymbol{P}_{a}\boldsymbol{V}^{\pi} \quad \forall a \in A \setminus a_1$$ → Matrix notation

Ng, Andrew Y., and Stuart J. Russell. "Algorithms for inverse reinforcement learning." *Icml*. 2000.

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)}(s') V^{\pi}(s')$$

→ (3) Bellman Equation on V

$\pi(s) \equiv a_1$ *is optimal* action

$$\boldsymbol{V}^{\pi} = \boldsymbol{R} + \gamma \boldsymbol{P}_{a_1} \boldsymbol{V}^{\pi}$$

→ a_1을 대입, Matrix notation

$$\boldsymbol{V}^{\pi} = (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R}$$

→ V에 대한 closed form

$$\boldsymbol{P}_{a_1} \boldsymbol{V}^{\pi} \succeq \boldsymbol{P}_{a} \boldsymbol{V}^{\pi} \qquad \forall a \in A \setminus a_1$$

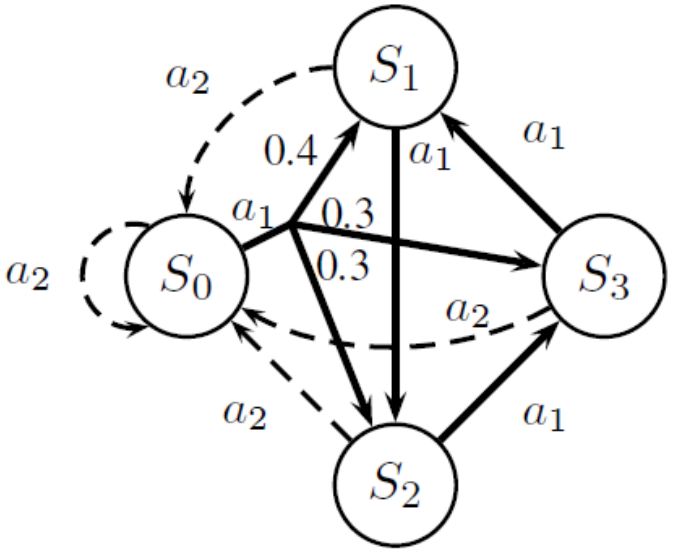
→ 이 부등식에 V 대입

$$\boldsymbol{P}_{a_1} (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \succeq \boldsymbol{P}_{a} (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \qquad \forall a \in A \setminus a_1$$

$$(\boldsymbol{P}_{a_1} - \boldsymbol{P}_{a}) (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \succeq 0$$

→ 한쪽으로 이항하면, R이 갖는 첫 번째 조건식 완성

Multiple Reward Functions



limit

S1 → S2 → S3 → S1 …

Figure 1: An example IRL problem. Bold lines represent the optimal action a_1 for each state and broken lines represent some other action a_2. Action a_1 in s_1 has probabilities 0.4,0.3

- At least three reward functions are possible.
- $R_1(s_1) = 1, R_1(s_2) = 0, R_1(s_3) = 0$
  $R_2(s_1) = 0, R_2(s_2) = 1, R_2(s_3) = 0$
  $R_3(s_1) = 0, R_3(s_2) = 0, R_3(s_3) = 1$

Ramachandran, Deepak, and Eyal Amir. "Bayesian inverse reinforcement learning." *Urbana* 51.61801 (2007): 1-4.

Multiple Reward Functions

$$(\boldsymbol{P}_{a_1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1}\boldsymbol{R} \succeq 0$$

- IRL problem is ill-posed. For example, R=0 will always be a solution
- And there usually exist multiple optimal policies under the same reward function, and multiple reward functions can provide the same optimal policy (Lopes et al., 2009).

- Therefore, additional criteria are introduced to differentiate a good choice of R from the other solutions:

Zhifei, Shao, and Er Meng Joo. "A survey of inverse reinforcement learning techniques." *International Journal of Intelligent Computing and Cybernetics* 5.3 (2012): 293-311.

Max margin formulation

(1) Maximize the difference between the best and second best solutions.

$$\max\left\{\sum_{i=1}^{N}\min_{a\in A/a_1}\{(\mathbf{P}_{a_1}(i)-\mathbf{P}_a(i))(I-\gamma\mathbf{P}_{a_1}(i))^{-1}\mathbf{R}\}\right\}$$

Typo인듯

(2) Simple solutions are preferred.

$$\max\left\{\sum_{i=1}^{N}\min_{a\in A/a_1}\{(\mathbf{P}_{a_1}(i)-\mathbf{P}_a(i))(I-\gamma\mathbf{P}_{a_1})^{-1}\mathbf{P}\}-\lambda\|\mathbf{R}\|_1\right\}$$

이것도 P가아니라 R아니야?

So the reward function optimization problem can be formulated as this

$$\max\left\{\sum_{i=1}^{N}\min_{a\in A/a_1}\{(\mathbf{P}_{a_1}(i)-\mathbf{P}_a(i))(I-\gamma\mathbf{P}_{a_1})^{-1}\mathbf{P}\}-\lambda\|\mathbf{R}\|_1\right\}$$

$$s.t.\quad (\mathbf{P}_{a_1}-\mathbf{P}_a)(I-\gamma\mathbf{P}_{a_1})^{-1}\mathbf{R}\geq 0\quad\text{and}\quad |\mathbf{R}_i|\leq R_{\max}$$

(2) Infinite-state MDP with known optimal policy

- For infinite state space problems, searching through all reward functions is impractical.
- **Instead, the reward function is approximated using a linear combination of all useful features.**

$$R(s) = \alpha_1 \phi_1(s) + \alpha_2 \phi_2(s) + \cdots + \alpha_d \phi_d(s) = \boldsymbol{\alpha}\boldsymbol{\phi}(s)$$

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

$$= E\left[\sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^{d} \alpha_i \phi_i(s_t) | \pi\right]$$

$$= \sum_{i=1}^{d} \alpha_i E\left[\sum_{t=0}^{\infty} \gamma^t \phi_i(s_t) | \pi\right]$$

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

$$V(\pi) = V(\pi) = \sum_{i=1}^{d} \alpha_i \mu_i(\pi) = \boldsymbol{\alpha}\boldsymbol{\mu}(\pi)$$

$$V^{\pi^*}(s_0) \geq V^{\pi_i}(s_0) \quad \Longrightarrow \quad \boldsymbol{\alpha}\boldsymbol{\mu}(\pi^E) \geq \boldsymbol{\alpha}\boldsymbol{\mu}(\pi^i)$$

- (3) Infinite-state MDP with unknown optimal policy, but demonstrations are given ➔ **Apprenticeship Learning**

$$\hat{\boldsymbol{\mu}}(\pi^E) = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\mu}(\pi^i)$$

Apprenticeship Learning Algorithm

(1) Randomly generalize a policy π_0, and i starts from 1.

(2) Perform the following optimization:

$$\max \left\{ t_i = \min_{j \in \{0,1,\ldots i-1\}} \boldsymbol{\alpha}(\boldsymbol{\mu}(\pi^E) - \boldsymbol{\mu}(\pi^j)) \right\} \quad (19)$$
$$s.t. \quad \|\boldsymbol{\alpha}\|_2 = 1$$

Value function의 margin을 maximize 해서 더 좋은 value function을 찾음

where π^j is one of the policy from the generated policy reservoir. This optimization is consistent with previous IRL algorithms that the reward function should greatly distinguish the best and second best policy.

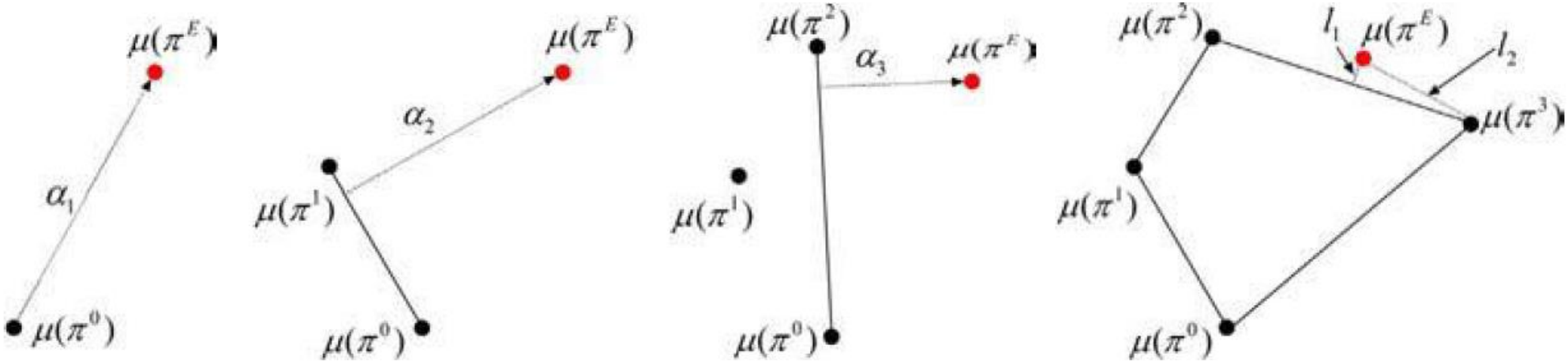
(3) If $t_j \leq \varepsilon$, then terminate, where ε is a predefined threshold.

(4) Find a new policy π^j that maximize V^{π^j} under the new reward function $R = \alpha\phi(s)$ using RL algorithms, and add policy π^j into the policy reservoir.

(5) Set $i = i + 1$ and go to step (2).

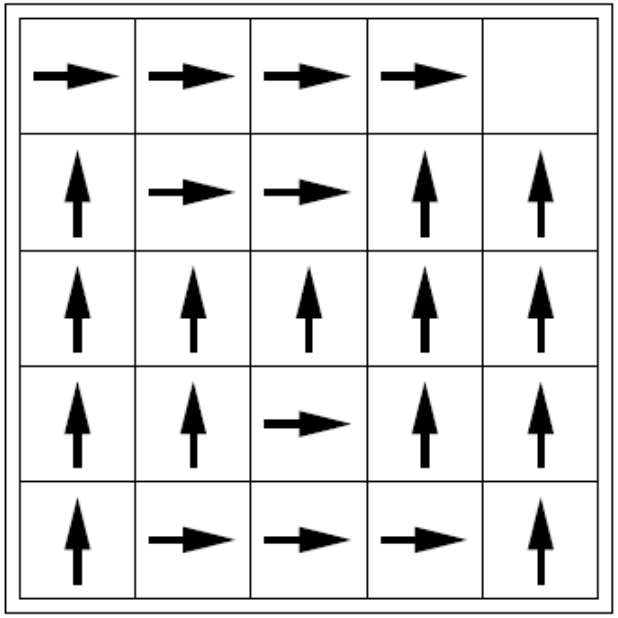
Policy Reservoir

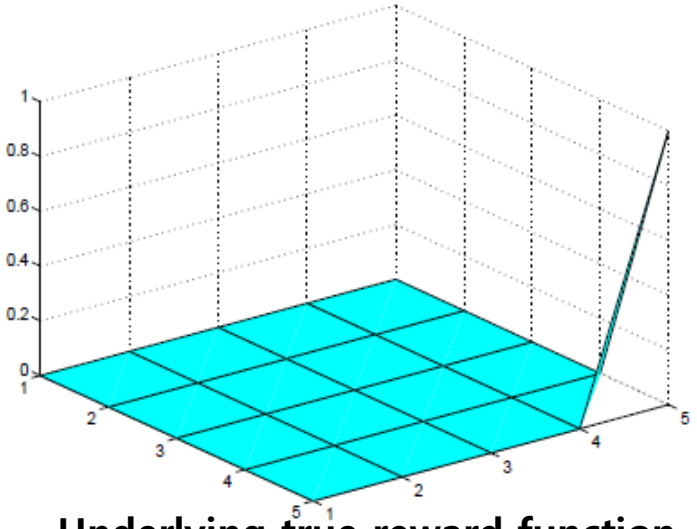
- The optimization problem in equation (19) to be solved by support vector machines (SVM), and α can be viewed as the unit vector orthogonal to the maximum margin that separates $\mu(\pi^E)$ and $\{\boldsymbol{\mu}(\boldsymbol{\pi}^j) | j \in (0, 1, \ldots, i-1)\}$



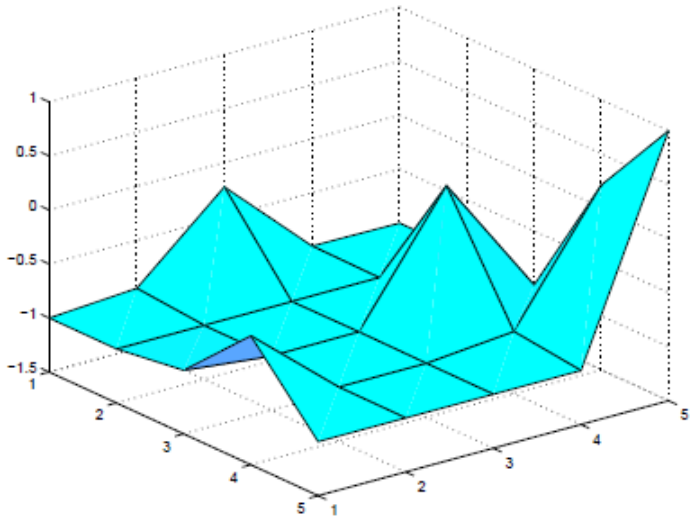
- Then the near optimal policy can be solved using a linear combination of generated policies.

Learning Discrete Reward Function





- **Underlying true reward function**



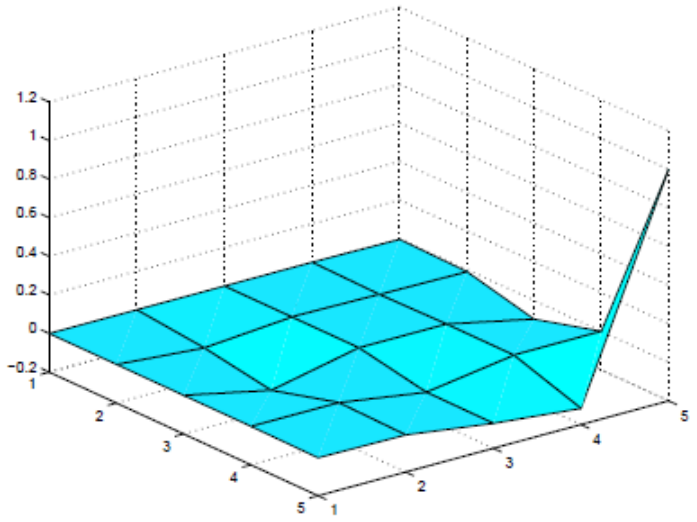


Figure 2. Inverse RL on the 5×5 grid. *Top*: $\lambda = 0$. *Bottom*: $\lambda = 1.05$.

Learning Continuous Reward Function

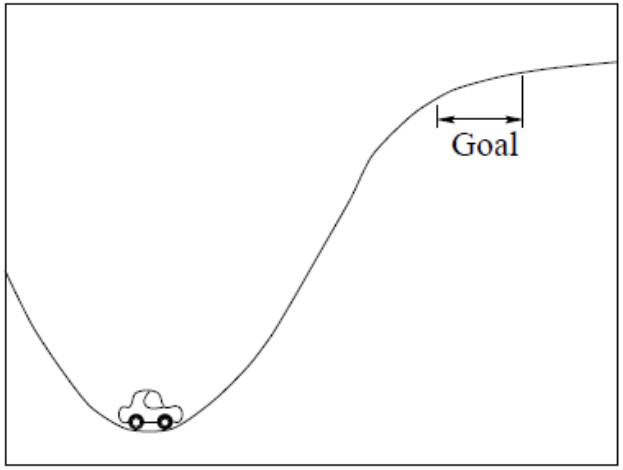


Figure 3. Cartoon of the mountain–car problem (not shown to scale).

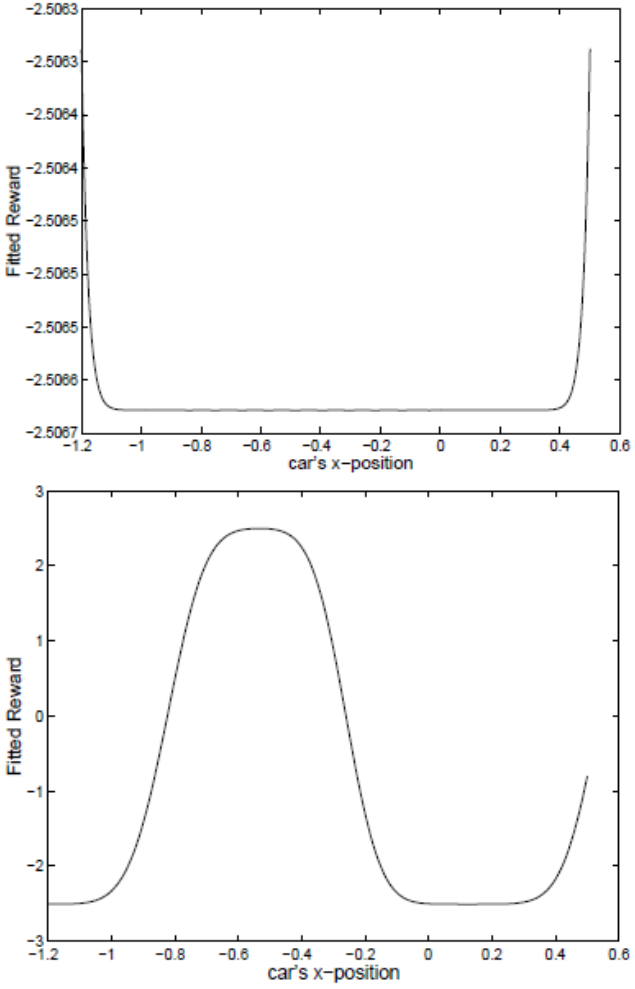


Figure 4. Typical solutions found by IRL for the mountain–car. *Top*: Original problem (note scale on y axis). *Bottom*: Problem of parking at bottom of hill.

Apprenticeship Learning

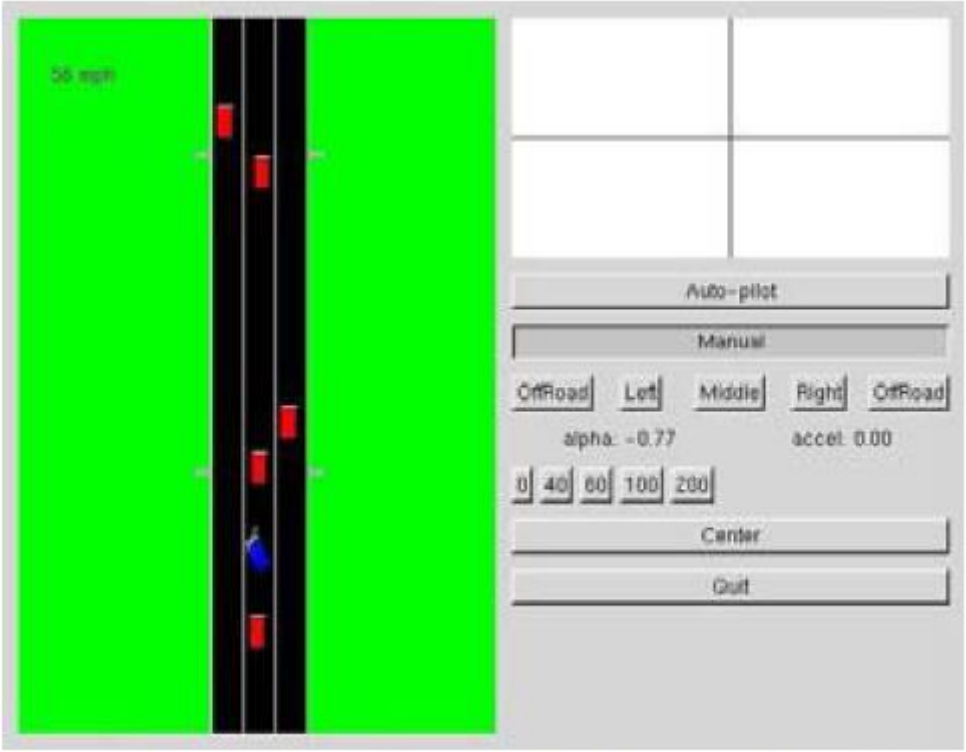


Figure 5. Screenshot of driving simulator.

Inverse Reinforcement Learning based on Critical State : https://www.youtube.com/watch?v=cMaOdoTt4Hw

2. Bayesian IRL

- IRL problem is ill-posed since there is uncertainty existed in the obtained reward function. Therefore, it is natural to use probability distribution to model this uncertainty.

  - Assumptions

1. $\mathcal{X}$ is attempting to maximize the total accumulated reward according to $\boldsymbol{R}$. For example, $\mathcal{X}$ is not using an epsilon greedy policy to explore his environment.
2. $\mathcal{X}$ executes a stationary policy, i.e. it is invariant w.r.t. time and does not change depending on the actions and observations made in previous time steps.

Derivation

$$P_\chi(O_\chi|R) = P_\chi((s_1, a_1)|R)P_\chi((s_2, a_2)|R)\ldots P_\chi((s_k, a_k)|R)$$

- agent $\mathcal{X}$ (the expert)
- expert's behaviour $O_\mathcal{X} = \{(s_1, a_1), (s_2, a_2) \ldots (s_k, a_k)\}$

- **Action-based distribution** : The larger Q*(s, a) is, the more likely it is that X would choose action a at state s.

$$P_\chi((s_i, a_i)|R) = \frac{1}{Z_i} e^{\alpha_\chi Q^*(s_i, a_i, R)}$$

$$P_\chi(O_\chi|R) = \frac{1}{Z} e^{\alpha_\chi E(O_\chi, R)} \qquad E(O_\chi, R) = \sum_i Q^*(s_i, a_i, R)$$

MAP estimation

$$P_\chi(R|O_\chi) = \frac{P_\chi(O_\chi|R)P_R(R)}{P(O_\chi)}$$

$$= \frac{1}{Z \cdot P(O_\chi)} e^{\alpha_\chi E(O_\chi,R)} P_R(R)$$

The normalizing factor $Z \cdot P(O_\chi)$ can be solved through sampling algorithms.

prior information can be implemented through $P_R(R)$

$$\begin{cases} P_{Gaussian}(\boldsymbol{R}(s) = r) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{r^2}{2\sigma^2}}, \forall s \in S \\ P_{Laplace}(\boldsymbol{R}(s) = r) = \frac{1}{2\sigma} e^{-\frac{|r|}{2\sigma}}, \forall s \in S \\ P_{Beta}(\boldsymbol{R}(s) = r) = \frac{1}{(\frac{r}{R_{max}})^{\frac{1}{2}}(1 - \frac{r}{R_{max}})^{\frac{1}{2}}}, \forall s \in S \end{cases}$$

Reward Learning

$$L_{SE}(\boldsymbol{R}, \hat{\boldsymbol{R}}) \quad = \quad \| \boldsymbol{R} - \hat{\boldsymbol{R}} \|_2$$

- **R** is underlying rewards and ˆ**R is** estimated rewards.
- **R** is drawn from the posterior distribution (3), it can be shown that the expected value of L_{SE}(**R**, ˆ**R**) is minimized by setting ´ $\frac{1}{Z'} e^{\alpha_X E(O_X, \boldsymbol{R})} P_R(\boldsymbol{R})$ of the posterior distribution.

Theorem 2. *When the expert's policy is optimal and fully specified, the IRL algorithm of [Ng and Russell, 2000] is equivalent to returning the MAP estimator for the model of (3) with a Laplacian prior.*

Apprenticeship Learning

- Original IRL과 달리, 기존의 RL알고리즘이 필요 없음

$$L^p_{policy}(\boldsymbol{R}, \pi) = \| \boldsymbol{V}^*(\boldsymbol{R}) - \boldsymbol{V}^{\pi}(\boldsymbol{R}) \|_p$$ ➡ *policy loss functions*

(**V** ∗(**R**) is the vector of optimal values for each state achieved by the optimal policy for **R** and p is some norm.)

$$E[L^p_{policy}(\pi)] = E(\| \boldsymbol{V}^*(\boldsymbol{R}) - \boldsymbol{V}^{\pi}(\boldsymbol{R}) \|_p)$$

(We wish to find the **π** that **minimizes the expected policy loss** over the posterior distribution for R.)

Sampling Algorithm for Rapid Convergence

- We have seen that both reward learning and apprenticeship learning require computing the mean of the posterior distribution.

Algorithm `PolicyWalk`(Distribution P, MDP M, Step Size δ)

1. Pick a random reward vector $\boldsymbol{R} \in \mathbb{R}^{|S|}/\delta$.
2. $\pi := \texttt{PolicyIteration}(M, \boldsymbol{R})$
3. Repeat
   (a) Pick a reward vector $\tilde{\boldsymbol{R}}$ uniformly at random from the neighbours of $\boldsymbol{R}$ in $\mathbb{R}^{|S|}/\delta$.
   (b) Compute $Q^{\pi}(s, a, \tilde{\boldsymbol{R}})$ for all $(s, a) \in S, A$.
   (c) If $\exists (s, a) \in (S, A), Q^{\pi}(s, \pi(s), \tilde{\boldsymbol{R}}) < Q^{\pi}(s, a, \tilde{\boldsymbol{R}})$
      i. $\tilde{\pi} := \texttt{PolicyIteration}(M, \tilde{\boldsymbol{R}}, \pi)$
      ii. Set $\boldsymbol{R} := \tilde{\boldsymbol{R}}$ and $\pi := \tilde{\pi}$ with probability $\min\{1, \frac{P(\tilde{\boldsymbol{R}}, \tilde{\pi})}{P(\boldsymbol{R}, \pi)}\}$
      Else
      i. Set $\boldsymbol{R} := \tilde{\boldsymbol{R}}$ with probability $\min\{1, \frac{P(\tilde{\boldsymbol{R}}, \pi)}{P(\boldsymbol{R}, \pi)}\}$
4. Return $\boldsymbol{R}$

Figure 3: PolicyWalk Sampling Algorithm

3. Maximum Entropy IRL

- Very similar with Bayesian IRL. But there are important distinctions.

1) Principle of maximum entropy.
2) Maximum Likelihood Estimation.
3) **Path-based distribution. → Label bias 해결**
4) Gradient ascent method.

Label Bias Problem

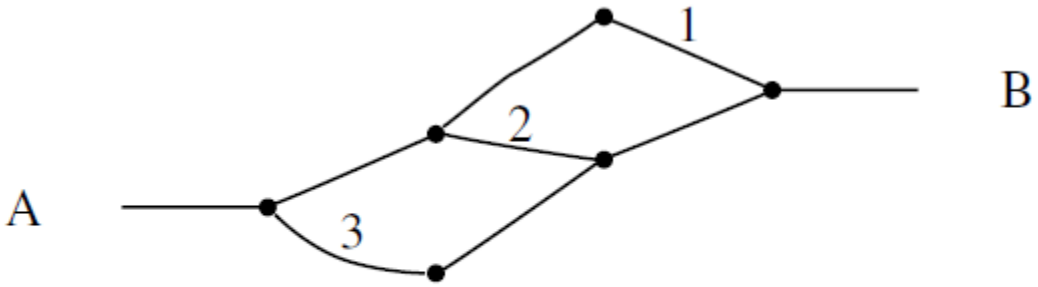


Figure 2: Example of probability distributions over paths.

- There are three obvious paths from **A** to **B** in Figure 2. Assuming each path provides the same reward, in the maximum entropy model, each path will have equal probability.
- In the action-based model, path 3 will have 50% probability while paths 1 and 2 have 25% probability. The distribution will be different for the return trip from **B** to **A**.
- This problem is known as label bias in the Conditional Random Field literature (Lafferty, Mc-Callum, & Pereira 2001).

Label Bias Problem

- Previous IRL methods focus on actions, where the trajectories after the actions are compared, instead of the trajectories before[2], and this causes the problem of label bias (Lafferty et al., 2011).

- As a consequence, policy with the highest reward may not be the one with the highest probability (Ziebart et al., 2008). Maximum entropy IRL avoids this problem by focusing on the distribution over trajectories rather than actions

3. Maximum Entropy IRL

- Principle of maximum entropy

$$P_\chi(O_\chi|\theta) = \frac{1}{Z} e^{\alpha_\chi E(O_\chi,\theta)}$$

- Maximum Likelihood Estimation

$$\theta^* = \arg\max_\theta L(\theta) = \arg\max_\theta \sum_{examples} \log P_\chi(O_\chi|\theta)$$

$$\nabla L(\theta) = \tilde{\mathbf{f}} - \sum_\zeta P(\zeta|\theta,T)\mathbf{f}_\zeta = \tilde{\mathbf{f}} - \sum_{s_i} D_{s_i}\mathbf{f}_{s_i}$$

Dynamic Programming for D

Algorithm 1 Expected Edge Frequency Calculation

Backward pass

1. Set $Z_{s_i,0} = 1$
2. Recursively compute for N iterations

$$Z_{a_{i,j}} = \sum_k P(s_k|s_i, a_{i,j}) e^{\text{reward}(s_i|\theta)} Z_{s_k}$$

$$Z_{s_i} = \sum_{a_{i,j}} Z_{a_{i,j}}$$

Local action probability computation

3. $P(a_{i,j}|s_i) = \dfrac{Z_{a_{i,j}}}{Z_{s_i}}$

Forward pass

4. Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$
5. Recursively compute for $t = 1$ to N

$$D_{s_i,t+1} = \sum_{a_{i,j}} \sum_k D_{s_k,t} P(a_{i,j}|s_i) P(s_k|a_{i,j}, s_i)$$

Summing frequencies

6. $D_{s_i} = \sum_t D_{s_{i,t}}$

Experiments

- We collected GPS trace data from 25 Yellow Cab taxi drivers over a 12 week duration at all times of day

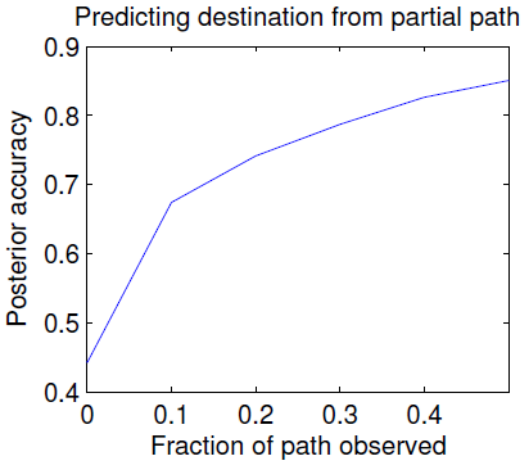


Figure 5: Posterior prediction accuracy over five destinations given partial path.

4. Deep IRL

- Started from **MaxEnt IRL** and improved these.

1) Non-linear reward function with DNN using Back-propagation.
2) Computationally Efficient than **GP IRL**

   (a significant drawback of this GPIRL approach is a computational complexity cubic in the size of the active set for every query posed)
3) Potentially possible to learn the feature representation. (not yet implemented.)

4. Deep IRL

$$
\begin{aligned}
r &\approx f(\mathbf{x}, \theta_1, \theta_2, ..., \theta_n) \\
&= f_1(f_2(...(f_n(\mathbf{x}, \theta_n), ...), \theta_2), \theta_1)
\end{aligned}
$$

$$
\log P(\mathcal{D}, \theta | \mathbf{x}) = \underbrace{\log P(\mathcal{D}|\mathbf{r})}_{\mathcal{L}_\mathcal{D}} + \underbrace{\log P(\theta)}_{\mathcal{L}_\theta}.
$$

$$
\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}_\mathcal{D}}{\partial \theta} + \frac{\partial \mathcal{L}_\theta}{\partial \theta}.
$$

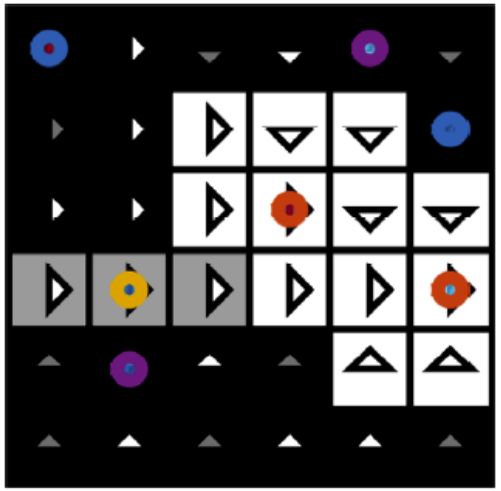
L2 regularization

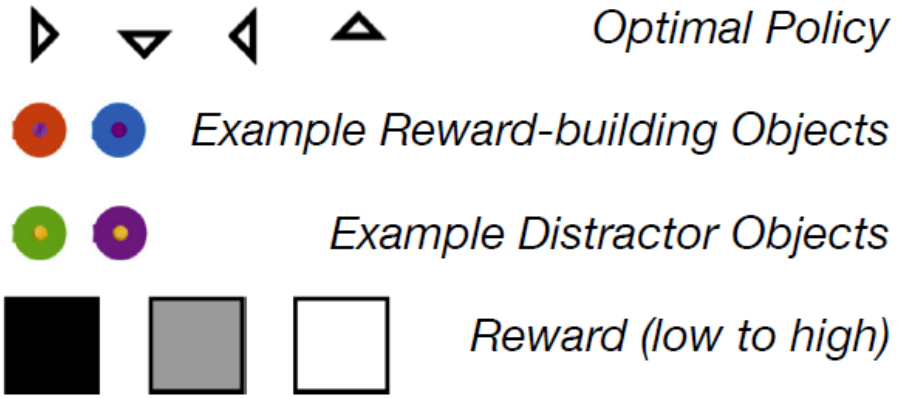
$$
\frac{\partial \mathcal{L}_\theta}{\partial \theta} = \lambda \theta.
$$

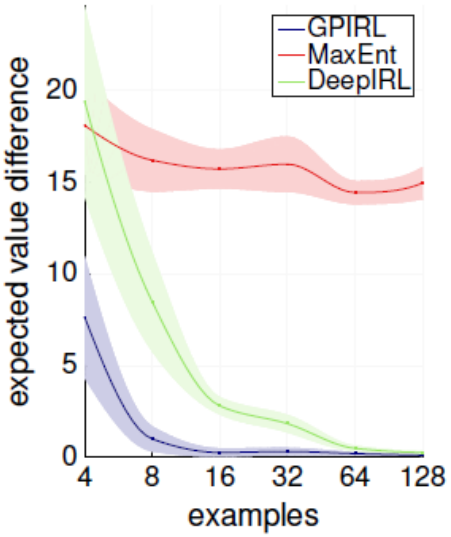
$$
\begin{aligned}
\frac{\partial \mathcal{L}_\mathcal{D}}{\partial \theta} &= \frac{\partial \mathcal{L}_\mathcal{D}}{\partial \mathbf{r}} \cdot \frac{\partial \mathbf{r}}{\partial \theta} \\
&= (\mu_\mathcal{D} - \mathbb{E}[\mu]) \cdot \frac{\partial}{\partial \theta} f(\mathbf{x}, \theta),
\end{aligned}
$$

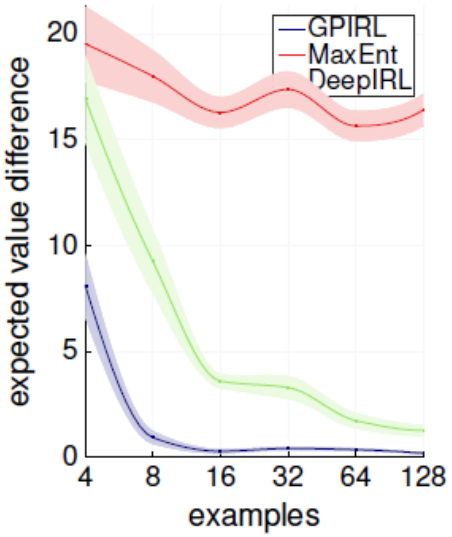
$$
\mathbb{E}[\mu] = \sum_{\varsigma : \{s,a\} \in \varsigma} P(\varsigma | \mathbf{r}).
$$

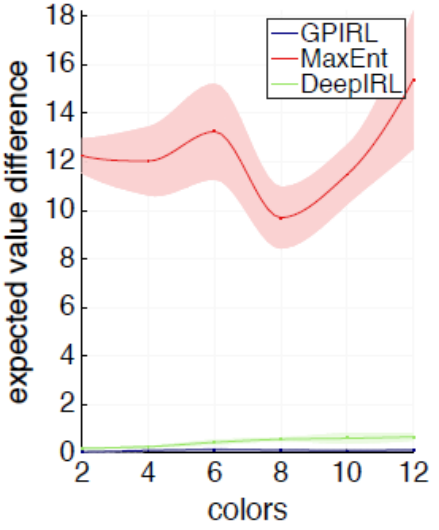
Experiments

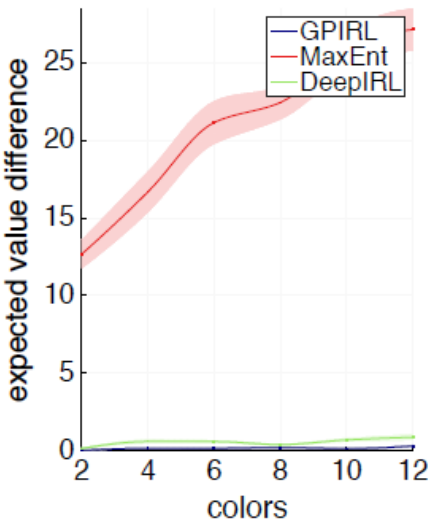












Experiments

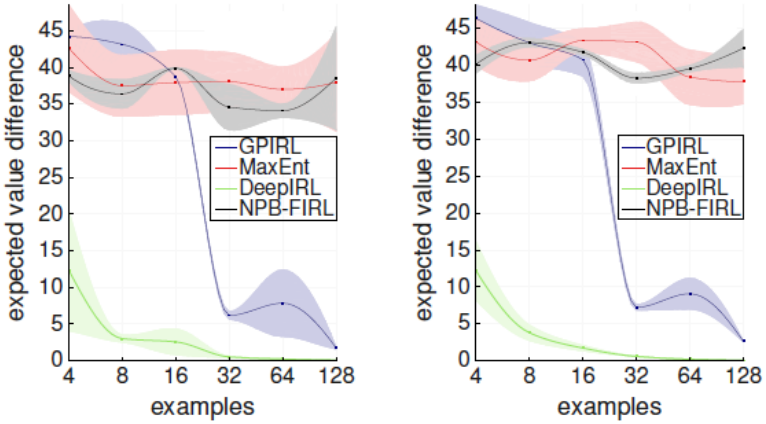


Figure 6: Value differences observed in the Binaryworld benchmark for GPIRL, MaxEnt and DeepIRL for the training scenario (left) and the transfer task (right).

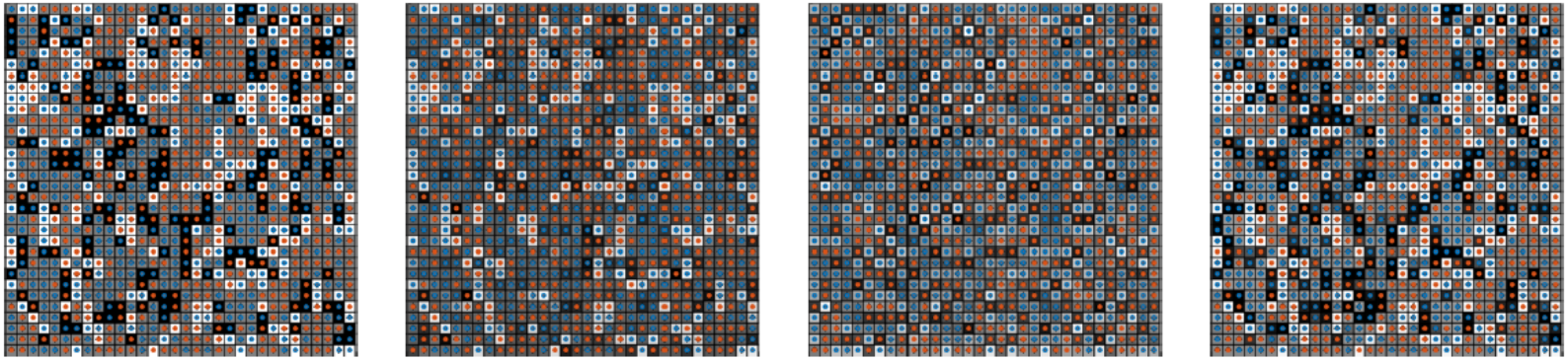


Figure 7: Reward reconstruction for the Binaryworld benchmark using GPIRL, MaxEnt and DeepIRL provided $N = 128$ demonstrations.

THANK YOU