Apprenticeship Learning via Inverse Reinforcement Learning

April 25, 2019

CONTACT

Kihwan Kim

0. Abstract

[문제]
Reward function을 정의하기 어려운 상황의 MDP 문제

[해결법]
전문가의 시연을 보고 학습하는 것은 효과적인 접근법
전문가가 **선형 조합**으로 표현한 **reward function**을 최대화 하려는 행동을 했다고 가정하고,
이 **reward function을 복구**하고자 inverse reinforcement learning 을 사용하는 알고리즘을 제안합니다.
=> Weights를 추정

[결과]
이 알고리즘이 적은 횟수로도 학습이 가능하며,
전문가 시연과 비슷한 성능을 얻을 수 있음을 실험을 통해 보이고자 합니다.

1. Introduction

- Apprenticeship learning
  어떤 task를 배울 때 전문가의 시연을 보고 배우는 것
  (Learning by watching, Imitation learning, Learning from demonstration)
  - Behavior cloning
    전문가의 행동을 외우는 방식, 조그만 상황이 바뀌어도 대처 불가능
  - Inverse reinforcement learning
    전문가의 행동을 단순히 모방하는 것이 아닌, 그 행동의 의도를 학습

    이 논문에서는 reward function을 feature들의 선형조합으로 표현한 다음,
    이를 Inverse reinforcement learning으로 학습하는 방법을 제안

2. Preliminaries

- (finite state) MDP is tuple (S,A,T, Γ,D,R)
  - S is finite set of states
  - A is set of actions
  - T is $\{P_{sa}\}$ is set of state transition probabilities
  - Γ ∈ [0,1] is discount factor
  - D is start state가 S_0인 initial-state distribution
  - R : S↦A is 크기가 1이하인 reward function

이 논문은 전문가의 시연으로부터 reward function을 찾고자 하므로,
reward가 없는 MDP인 MDP∖R를 다룹니다.

2. Preliminaries

- Φ : S→$[0,1]^k$, vector of features (task를 수행할 때 고려해야할 요소들)
  - 몇 차선을 달리고 있는지
  - 앞 차와의 거리
  - 다른 차와 충돌 여부

- $R^*(s) = \omega^* \cdot \phi(s)$, "true" reward function ($\|\omega^*\|_1 \leq 1$)
  - 결과적으로 (unkown) vector ω*는 task에 대한 각 고려 요소들의 상대적 weighting이라고 볼 수 있음

- Policy π는 action에 대해서 states를 확률 분포와 mapping하는 역할
  - 따라서 policy π의 value는 expectation으로 표현

$$
\begin{aligned}
E_{s_0 \sim D}[V^{\pi}(s_0)] &= E[\textstyle\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] && (1)\\
&= E[\textstyle\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t)|\pi] && (2)\\
&= w \cdot \boxed{E[\textstyle\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi]} && (3)
\end{aligned}
$$

$$\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \in \mathbb{R}^k$$

feature expectation

2. Preliminaries

$$\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \in \mathbb{R}^k$$

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

Estimation of expert's feature expectation

$$\begin{aligned} E_{s_0 \sim D}[V^{\pi}(s_0)] &= E[\textstyle\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] && (1) \\ &= E[\textstyle\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t)|\pi] && (2) \\ &= w \cdot E[\textstyle\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] && (3) \end{aligned}$$

$$\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \in \mathbb{R}^k$$

feature expectation

2. Preliminaries

$$\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \in \mathbb{R}^k$$

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

Estimation of expert's feature expectation

$$\begin{aligned} E_{s_0 \sim D}[V^{\pi}(s_0)] &= E[\textstyle\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] && (1) \\ &= E[\textstyle\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t)|\pi] && (2) \\ &= w \cdot E[\textstyle\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] && (3) \end{aligned}$$

$\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \in \mathbb{R}^k$

feature expectation

3. Algorithm

$$
\begin{aligned}
& |E[\textstyle\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi_E] - E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\tilde{\pi}]| && (6) \\
&= |w^T \mu(\tilde{\pi}) - w^T \mu_E| && (7) \\
&\leq \|w\|_2 \|\mu(\tilde{\pi}) - \mu_E\|_2 && (8) \\
&\leq 1 \cdot \epsilon = \epsilon && (9)
\end{aligned}
$$

$$\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \in \mathbb{R}^k$$

$$|x^T y| \leq ||x||_2 ||y||_2$$

3. Algorithm

1. Randomly pick some policy $\pi^{(0)}$, compute (or approximate via Monte Carlo) $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$.
2. Compute $t^{(i)} = \max_{w:\|w\|_2 \leq 1} \min_{j \in \{0..(i-1)\}} w^T(\mu_E - \mu^{(j)})$, and let $w^{(i)}$ be the value of w that attains this maximum.
3. If $t^{(i)} \leq \epsilon$, then terminate.
4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (w^{(i)})^T \phi$.
5. Compute (or estimate) $\mu^{(i)} = \mu(\pi^{(i)})$.
6. Set $i = i + 1$, and go back to step 2.

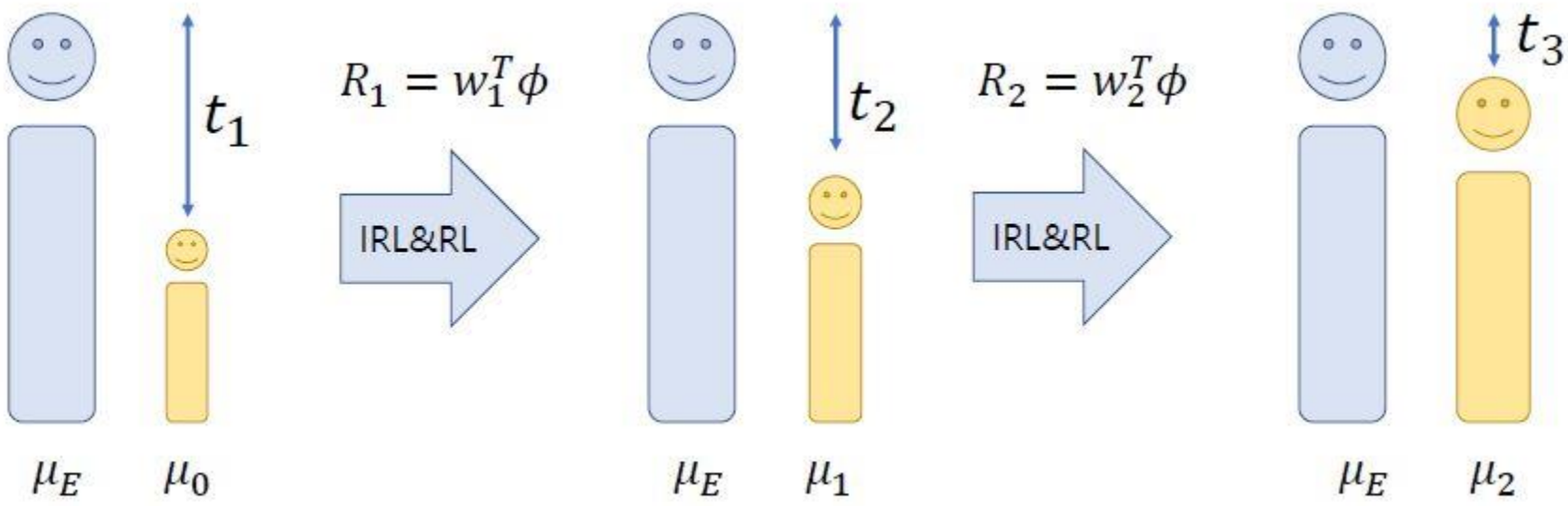
a. Expert feature expectation과 feature expectation set로 부터 계산한 expert와 learner의 performance 차이를 t 로 정의하고, t를 최대화하는 weight를 찾는 과정. 다시 말해 reward를 찾는 IRL step.

b. IRL step 에서 얻은 reward function에 대한 optimal policy를 찾는 RL step.

c. RL step에서 구한 policy로부터 Monte Carlo 시행을 통해 새로운 feature expectation을 구하고, 이를 feature expectation set에 추가

d. a와 b의 IRL step ⇔ RL step 반복하다 t가 ϵ 이하일 때, 즉 feature expectation이 충분히 가까워 졌을 때 학습 종료.

3. Algorithm



Step a. 에서 t를 최대화 하는 것은, learner에 비해서 expert의 performance를 더 잘 설명하는 reward function을 만들고자 하는 것이며, 마치 틀린 시험문제에 대한 더 자세한 오답노트를 만드는것과 같습니다. 이렇게 expert와의 차이가 커야 learner가 RL step에서 이 reward function으로 policy를 다시 학습 했을 때 더 발전을 하기 때문입니다.

3. Algorithm

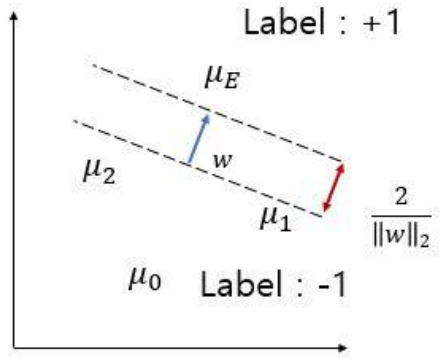
2. Compute $t^{(i)} = \max_{w:\|w\|_2 \leq 1} \min_{j \in \{0..(i-1)\}} w^T(\mu_E - \mu^{(j)})$, and let $w^{(i)}$ be the value of w that attains this maximum.

$$\max_{t,w} \quad t \tag{10}$$

$$\text{s.t.} \quad w^T\mu_E \geq w^T\mu^{(j)} + t,\ j = 0, \ldots, i-1 \tag{11}$$

$$\|w\|_2 \leq 1 \tag{12}$$

$$\forall w \text{ with } \|w\|_2 \leq 1\ \exists i \text{ s.t. } w^T\mu^{(i)} \geq w^T\mu_E - \epsilon. \tag{13}$$



$$\max\left(t = \frac{2}{\|w\|_2}\right) \Rightarrow \min\|w\|_2^2$$

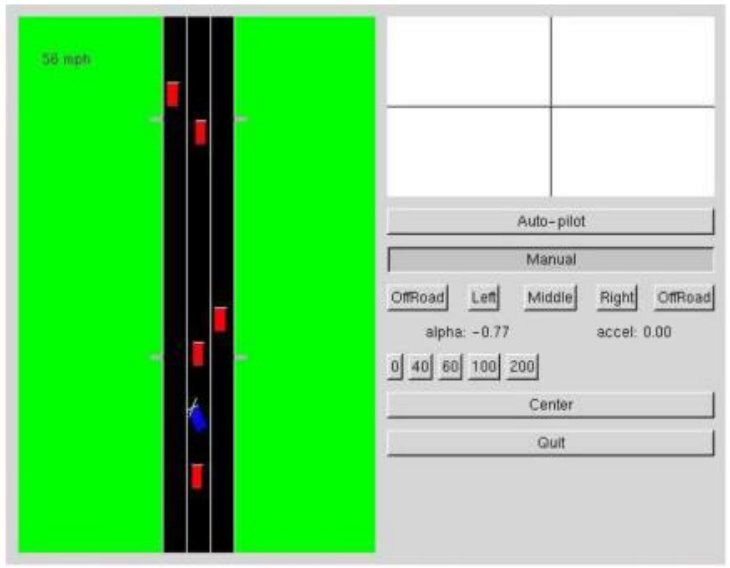
Quadratic loss

$$\min\|w\|_2^2$$

$$s.t\ w^T\mu_E \geq w^T\mu_i + 2$$

이는 Linear IRL(Ng & Russell, 2000) 에서 사용한 Linear programming (LP) 최적화 문제와 유사해 보이지만, ω 에 대한 2-norm (L2)이 constraint 인 차이점이 있습니다. APP 논문을 작성할 당시엔 L2 norm constraint를 포함한 LP를 풀 수 있는 Convex optimization solver가 없었기 때문에, 저자는 논문에서 Quadratic programming (QP) 의 일종인 Support vector machine (SVM)을 사용하는 최적화 방법을 제안합니다.

4. Experiment

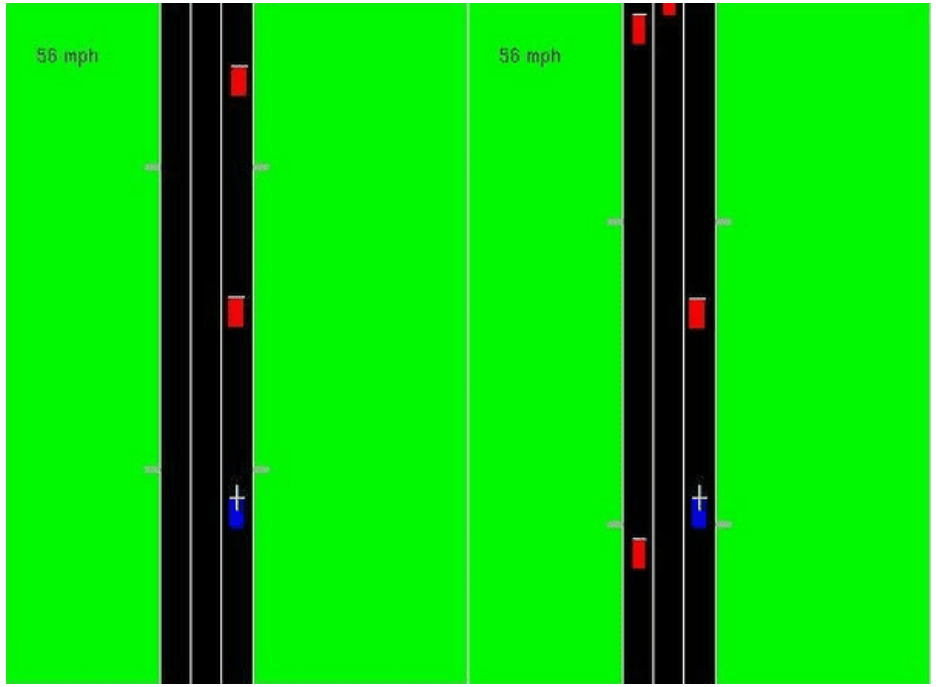


주변의 빨간색 자동차들보다 빠른 25 m/s의 고정된 속도로 움직이는 파란색 자동차를 **좌우**로 움직일 수 있습니다.

선택할 수 있는 **action은 총 5가지로**, 왼쪽/중앙/오른쪽 레인으로 자동차를 이동시키는 action 3가지와 왼쪽/오른쪽의 초록색 비포장도로로 자동차를 이동시키는 2가지입니다.

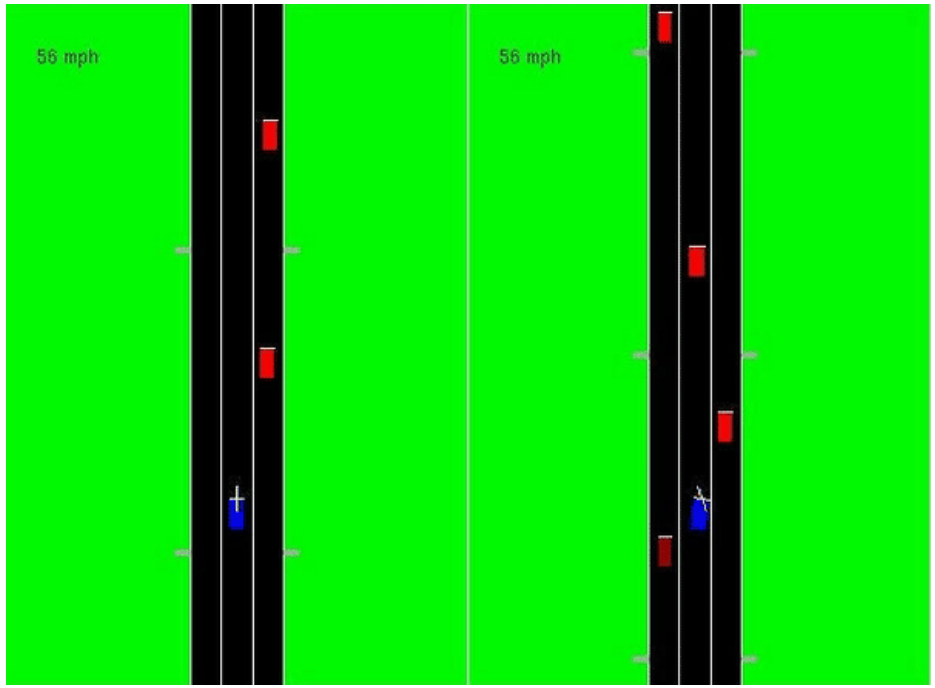
4. Experiment

- Nice: 충돌을 피하는 것을 최우선적으로 함. 또한 레인의 선호도 차이가 있음. (오른쪽 > 중앙 > 왼쪽 > 비포장도로).
- Nasty: 가능한 많은 충돌을 일으킴.
- Right lane nice: 오른쪽 레인으로 달리되 충돌을 피하기 위해 오른쪽 비포장 도로를 사용함.
- Right lane nasty: 오른쪽 비포장 도로를 달리되 충돌하기 위해 오른쪽 레인으로 들어옴.
- Middle lane: 충돌에 상관없이 중앙으로만 달림.



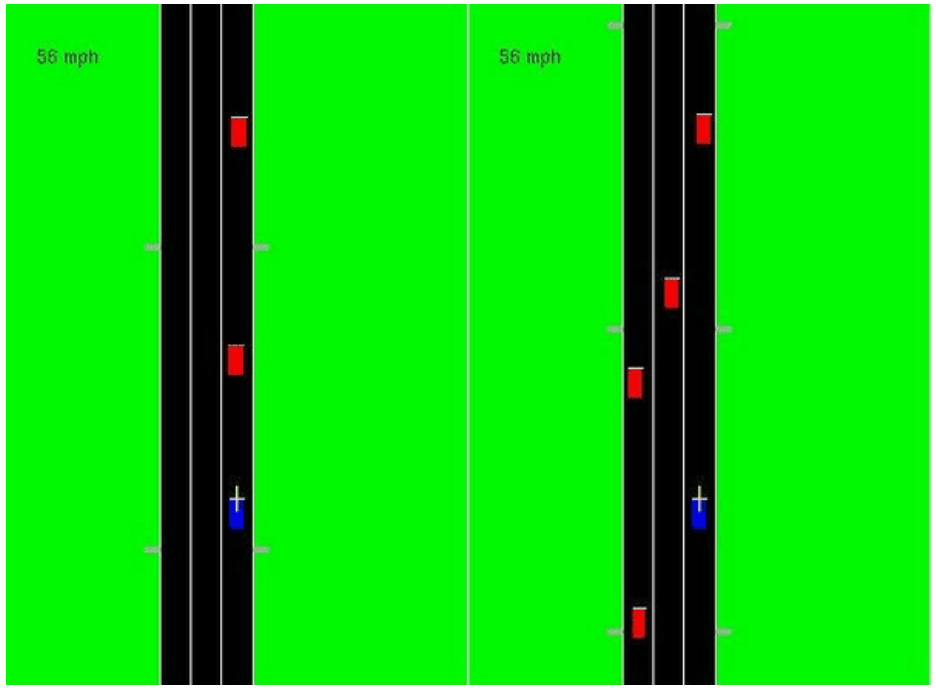
4. Experiment

- Nice: 충돌을 피하는 것을 최우선적으로 함. 또한 레인의 선호도 차이가 있음. (오른쪽 > 중앙 > 왼쪽 > 비포장도로).
- **Nasty: 가능한 많은 충돌을 일으킴.**
- Right lane nice: 오른쪽 레인으로 달리되 충돌을 피하기 위해 오른쪽 비포장 도로를 사용함.
- Right lane nasty: 오른쪽 비포장 도로를 달리되 충돌하기 위해 오른쪽 레인으로 들어옴.
- Middle lane: 충돌에 상관없이 중앙으로만 달림.



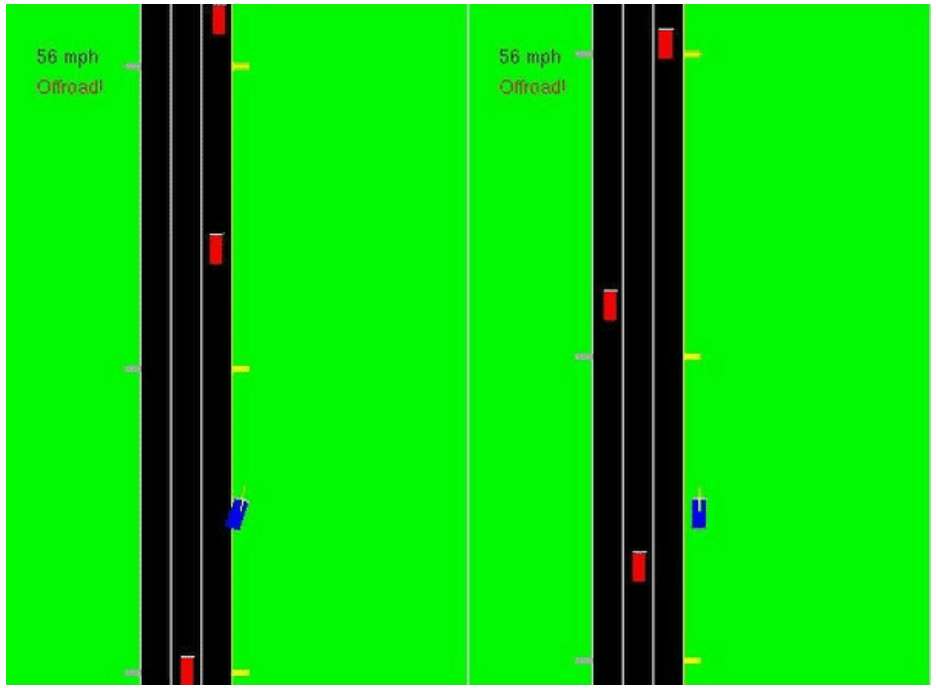
4. Experiment

- Nice: 충돌을 피하는 것을 최우선적으로 함. 또한 레인의 선호도 차이가 있음. (오른쪽 > 중앙 > 왼쪽 > 비포장도로).
- Nasty: 가능한 많은 충돌을 일으킴.
- Right lane nice: 오른쪽 레인으로 달리되 충돌을 피하기 위해 오른쪽 비포장 도로를 사용함.
- Right lane nasty: 오른쪽 비포장 도로를 달리되 충돌하기 위해 오른쪽 레인으로 들어옴.
- Middle lane: 충돌에 상관없이 중앙으로만 달림.



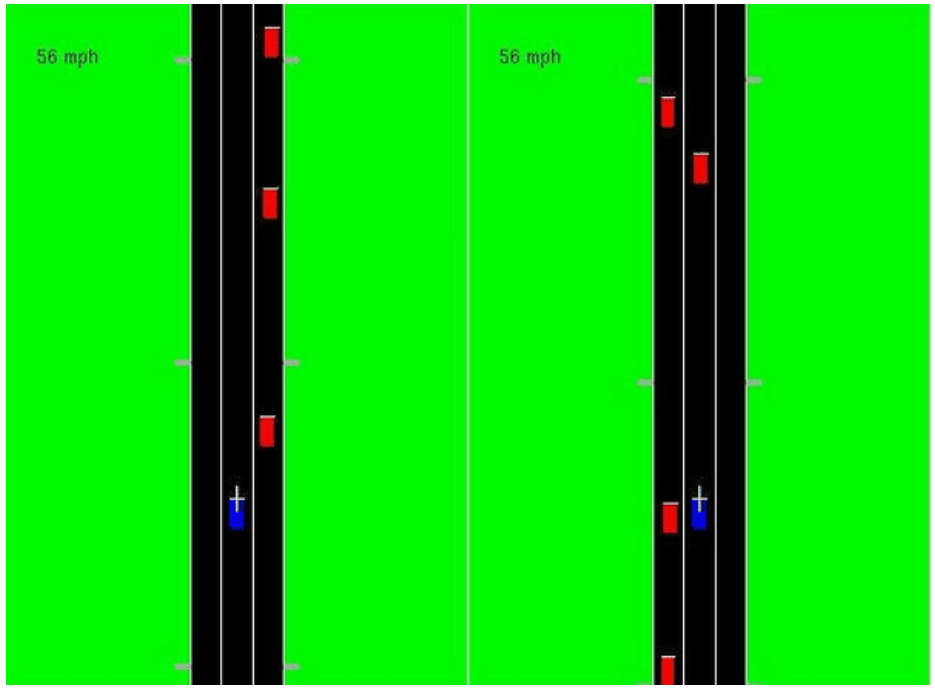
4. Experiment

- Nice: 충돌을 피하는 것을 최우선적으로 함. 또한 레인의 선호도 차이가 있음. (오른쪽 > 중앙 > 왼쪽 > 비포장도로).
- Nasty: 가능한 많은 충돌을 일으킴.
- Right lane nice: 오른쪽 레인으로 달리되 충돌을 피하기 위해 오른쪽 비포장 도로를 사용함.
- Right lane nasty: 오른쪽 비포장 도로를 달리되 충돌하기 위해 오른쪽 레인으로 들어옴.
- Middle lane: 충돌에 상관없이 중앙으로만 달림.



4. Experiment

- Nice: 충돌을 피하는 것을 최우선적으로 함. 또한 레인의 선호도 차이가 있음. (오른쪽 > 중앙 > 왼쪽 > 비포장도로).
- Nasty: 가능한 많은 충돌을 일으킴.
- Right lane nice: 오른쪽 레인으로 달리되 충돌을 피하기 위해 오른쪽 비포장 도로를 사용함.
- Right lane nasty: 오른쪽 비포장 도로를 달리되 충돌하기 위해 오른쪽 레인으로 들어옴.
- Middle lane: 충돌에 상관없이 중앙으로만 달림.



4. Experiment

1. Nice: 충돌을 피하는 것을 최우선적으로 함. 또한 레인의 선호도 차이가 있음. (오른쪽 > 중앙 > 왼쪽 > 비포장도로).

		Collision	Offroad Left	LeftLane	MiddleLane	RightLane	Offroad Right
1	$\hat{\mu}_E$	0.0000	0.0000	0.1325	0.2033	0.5983	0.0658
	$\mu(\tilde{\pi})$	0.0001	0.0004	0.0904	0.2287	0.6041	0.0764
	$\tilde{w}$	-0.0767	-0.0439	0.0077	0.0078	0.0318	-0.0035
2	$\hat{\mu}_E$	0.1167	0.0000	0.0633	0.4667	0.4700	0.0000
	$\mu(\tilde{\pi})$	0.1332	0.0000	0.1045	0.3196	0.5759	0.0000
	$\tilde{w}$	0.2340	-0.1098	0.0092	0.0487	0.0576	-0.0056
3	$\hat{\mu}_E$	0.0000	0.0000	0.0000	0.0033	0.7058	0.2908
	$\mu(\tilde{\pi})$	0.0000	0.0000	0.0000	0.0000	0.7447	0.2554
	$\tilde{w}$	-0.1056	-0.0051	-0.0573	-0.0386	0.0929	0.0081
4	$\hat{\mu}_E$	0.0600	0.0000	0.0000	0.0033	0.2908	0.7058
	$\mu(\tilde{\pi})$	0.0569	0.0000	0.0000	0.0000	0.2666	0.7334
	$\tilde{w}$	0.1079	-0.0001	-0.0487	-0.0666	0.0590	0.0564
5	$\hat{\mu}_E$	0.0600	0.0000	0.0000	1.0000	0.0000	0.0000
	$\mu(\tilde{\pi})$	0.0542	0.0000	0.0000	1.0000	0.0000	0.0000
	$\tilde{w}$	0.0094	-0.0108	-0.2765	0.8126	-0.5099	-0.0154

expert는 단지 시연을 한 것 뿐이지 일일이 보상을 주는 등의 true reward function을 따로 정하지 않았기 때문에 agent가 얼만큼의 보상을 받았는 지로는 알고리즘의 성능을 판단할 수 없습니다.
대신 driving style을 얼마나 잘 모방했는 지의 성능을 분석하는 것은 feature expectation의 비교로 가능합니다. 5가지 style에 따라 순서대로 expert와 알고리즘의 결과를 정리한 아래의 표를 보겠습니다.

5. Conclusions and Future work

이 논문은, 전문가가 선형 조합으로 표현한 reward function을 최대화 하려는 행동을 했다고 가정하고 이 reward function을 복구하고자 inverse reinforcement learning 을 사용하는 알고리즘을 제안하였습니다.

결과적으로 실험을 통해 제시한 알고리즘이 작은 횟수로도 학습이 가능하며, 전문가 시연과 비슷하거나 더 나은 성능을 얻을 수도 있음을 확인하였습니다.

하지만 demonstration을 설명할 feature 수가 많아지면 reward function이 feature들의 선형조합으로 나타낼 수 있다는 초기 가정을 보장할 수 없게 됩니다. feature들에 대해서 비선형으로 reward를 나타내거나 자동으로 feature를 설계하거나 선택하는 것은 매우 중요하며, 이에 대한 연구가 많이 필요합니다.

끝