Training Agent using Upside-Down Reinforcement Learning (ꓤꓶ)

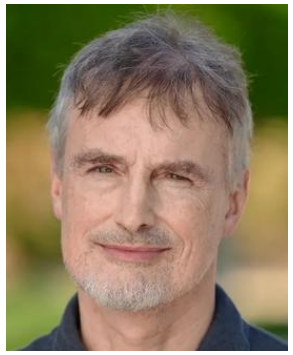
Rupesh Kumar Srivastava[1], Pranav Shyam[2], Filipe Mutz[3], Wojciech Jaśkowski[4], Jürgen Schmidhuber[1, 5]
[1]NNAISENSE, [2]Google DeepMind
[3]Tederal University of Espírito Santo (UFES), Brazil
[4]Snowflake, [5]The Swiss AI Lab IDSIA
NeurIPS Deep Reinforcement Learning Workshop 2019

2025.05.22.
김동민

Training Agents using Upside-Down Reinforcement Learning https://arxiv.org/abs/1912.02877
Reinforcement Learning Upside Down: Don't Predict Rewards -- Just Map Them to Actions https://arxiv.org/abs/1912.02875

Lifetime credit assignment problem solver

- Jürgen Schmidhuber
  - LSTM (Long Short-Term Memory)
  - Convolutional Neural Network(?)
  - Generative Adversarial Networks (?)
  - Transformer(?)

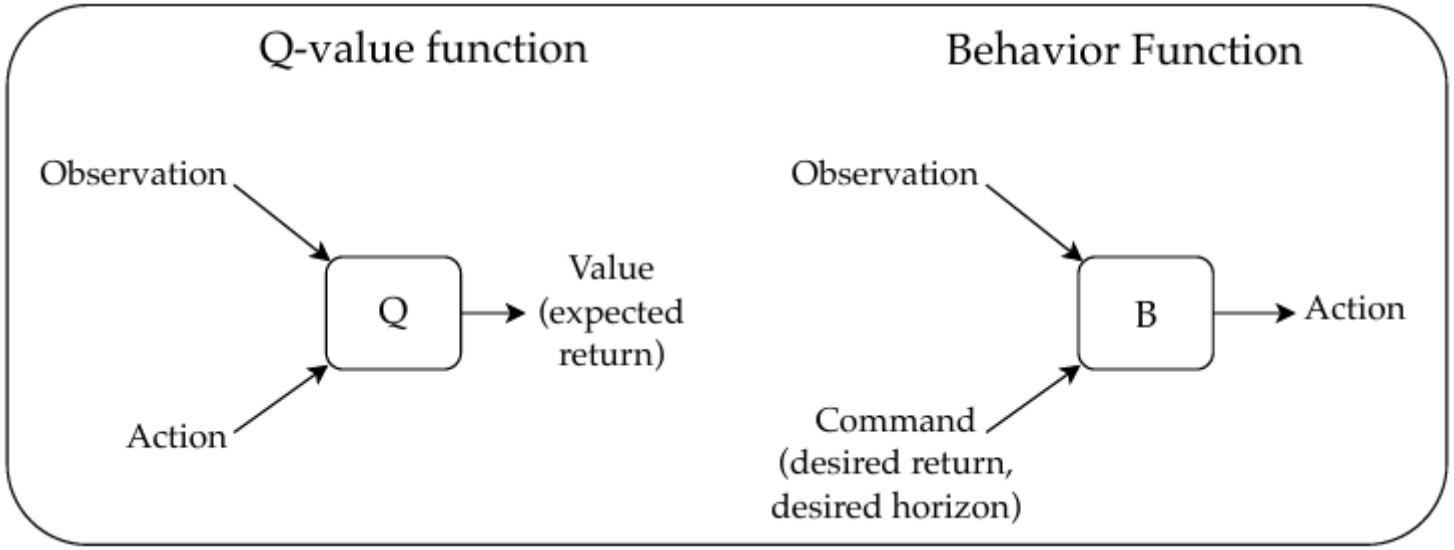


  - 이 논문에서도 CNN을 사용하는데 LeCun을 인용하지 않았음 (NN 관련 Hinton의 연구도 1개만 인용)

https://x.com/SchmidhuberAI/status/1728081107373916162
https://people.idsia.ch/~juergen/most-cited-neural-nets.html
https://people.idsia.ch/~juergen/physics-nobel-2024-plagiarism.html

In Short

- 전통적인 RL문제들을 Supervised Learning (SL) 방법**만** 사용해서 풀어보자
  - To bring the simplicity, robustness and scalability of SL algorithms to traditional RL algorithms
- Key Idea: Use Upside-down function (Behavior function) instead of Q-value function



SL Problem for behavior function $B_{\mathcal{T}}$

$$B_{\mathcal{T}} = \arg\min_{B} \sum_{(\tau,t_1,t_2)} L\left(B\left(a_{t_1}, s_{t_1}, d^r, d^h\right), a_{t_1}\right)$$

$$\text{where } 0 \le t_1 \le t_2 \le \text{len}(\tau) \text{ for } \tau \in \mathcal{T},\ d^r = \sum_{t=t_1}^{t_2} r_t \text{ and } d^h = t_2 - t_1$$

Upside-Down Reinforcement Learning (UDRL)

- 다음과 같은 질문에 답하기 위해 UDRL을 고안해냄
  - Is it possible to learn to act in high-dimensional environments efficiently using only SL, avoiding the issues arising from non-stationary learning objectives common in traditional RL algorithms?
- UDRL
  - a new approach to RL where instead of trying to predict rewards from states like value functions and Q-learning, the agents directly predict actions based on the state and the command
  - The goal of learning is no longer to maximize returns in expectation, but to learn to follow commands
- Command ([desired return, desired time horizon])
  - "achieve total reward R in next T time steps“
  - "reach state S in fewer than T time steps"
  - desired return $d^r = \sum_{t=t_1}^{t_2} r_t$
  - desired horizon $d^h = t_2 - t_1$

https://medium.com/@jscriptcoder/demystifying-upside-down-reinforcement-learning-a-k-a-%EA%93%A4-b7bd4214b33f

Toy Example

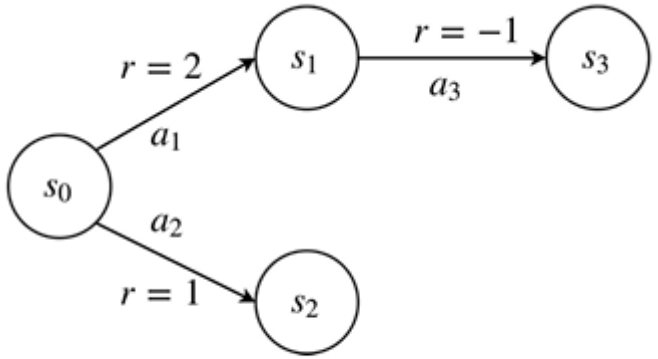
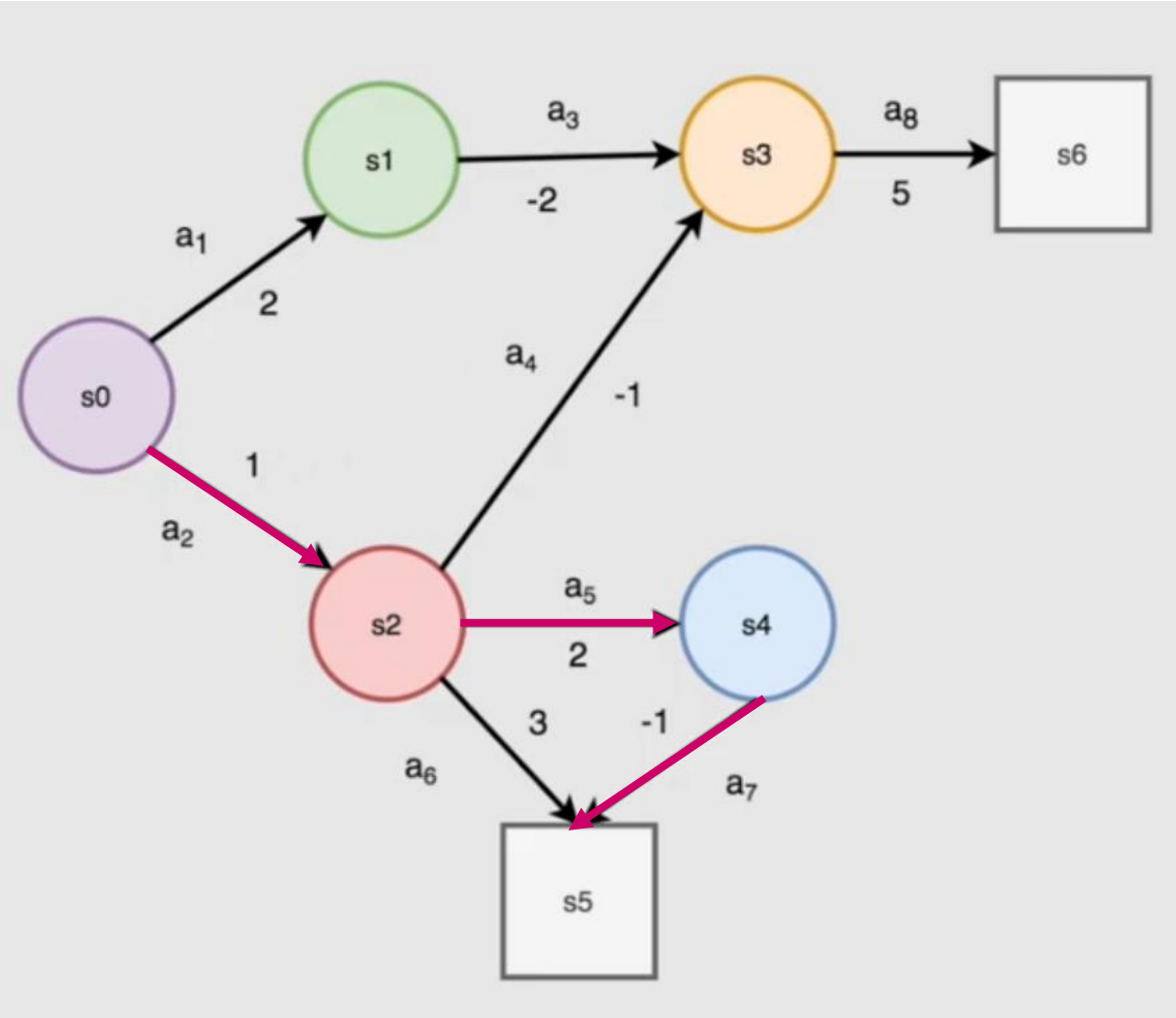


Figure 2: A toy environment with four states.

Table 1. A behavior function based on all unique trajectories for the toy environment (left image).

State	Desired Return	Desired Horizon	Action
s_0	2	1	a_1
s_0	1	1	a_2
s_0	1	2	a_1
s_1	−1	1	a_3

Example 2



trajectory of traditional RL

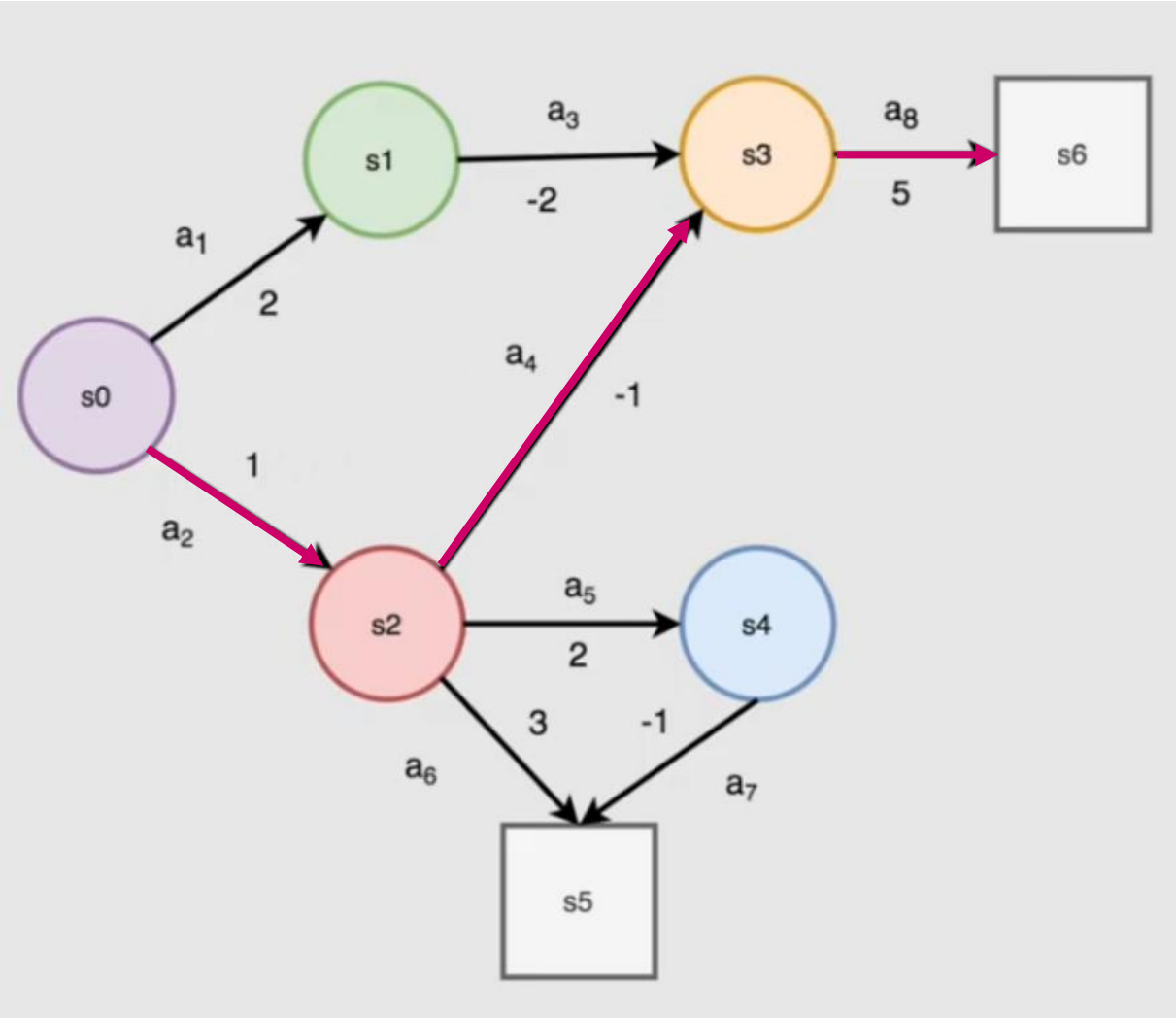
(s0, a2, 1, s2)
(s2, a5, 2, s4)
(s4, a7, -1, s5)

trajectory and SL of UDRL

L(B(s0, 2, 3), a2)
L(B(s2, 1, 2), a5)
L(B(s4, -1, 1), a7)

https://www.youtube.com/watch?v=ed7QQMG24MM

Example 2



trajectory of traditional RL

(s0, a2, 1, s2)
(s2, a4, -1, s3)
(s3, a8, 5, s6)

trajectory and SL of UDRL

L(B(s0, 5, 3), a2)
L(B(s2, 4, 2), a4)
L(B(s3, 5, 1), a8)

https://www.youtube.com/watch?v=ed7QQMG24MM

UDRL Algorithm 1

- Behavior function은 NN을 function approximator로 활용

Algorithm 1 Upside-Down Reinforcement Learning: High-level Description.

```
Initialize replay buffer with warm-up episodes using random actions        // Section 2.2.1
Initialize a behavior function                                             // Section 2.2.2
while stopping criteria is not reached do
    Improve the behavior function by training on replay buffer             // Section 2.2.3
    Sample exploratory initial commands based on replay buffer             // Section 2.2.4
    Explore using sampled commands in Algorithm 2 and add to replay buffer // Section 2.2.5
    If evaluation is required, evaluate using evaluation commands in Algorithm 2 // Section 2.2.6
end while
```

- Sampling exploratory commands

1. A number of episodes with the highest returns are selected from the replay buffer.
2. The exploratory desired horizon d_0^h is set to the mean of the lengths of the selected episodes.
3. The exploratory desired returns d_0^r are sampled from the uniform distribution $\mathcal{U}[M, M + S]$ where M is the mean and S is the standard deviation of the selected episodic returns.

UDRL Algorithm 2

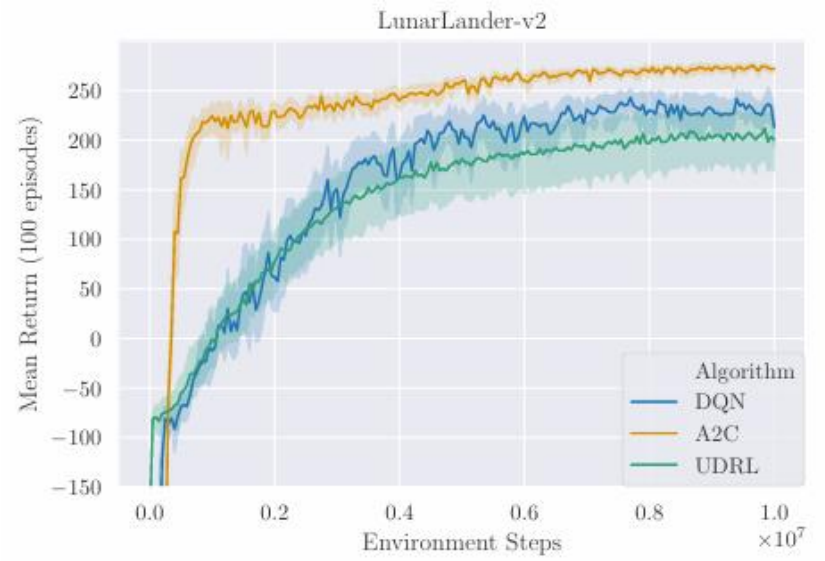
Algorithm 2 Generates an Episode using the Behavior Function.

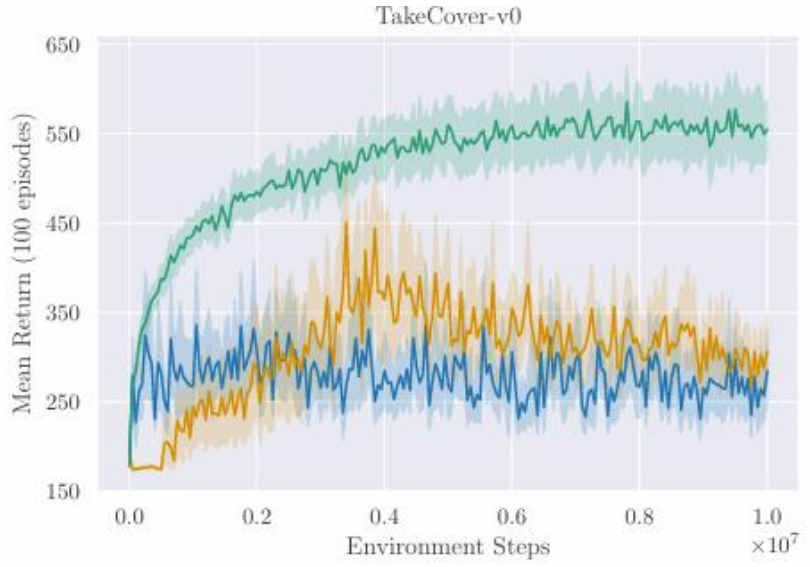
Input: Initial command $c_0 = (d_0^r, d_0^h)$, Initial state s_0, Behavior function B parameterized by θ
Output: Episode data E

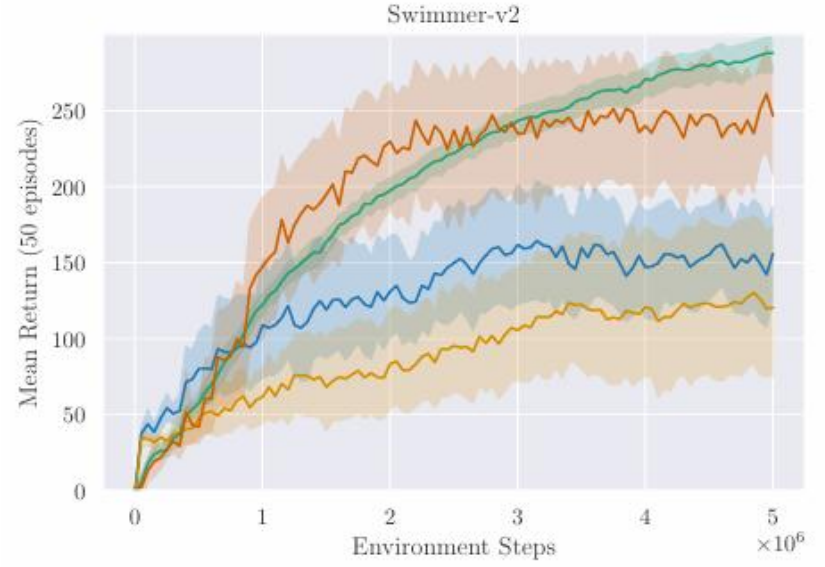
$E \leftarrow \varnothing$
$t \leftarrow 0$
while episode is not done **do**
    Compute $P(a_t \mid s_t, c_t) = B(s_t, c_t; \theta)$
    Execute $a_t \sim P(a_t \mid s_t, c_t)$ to obtain reward r_t and next state s_{t+1} from the environment
    Append (s_t, a_t, r_t) to E
    $t \leftarrow t + 1$
    $d_t^r \leftarrow d_{t-1}^r - r_{t-1}$ // Update desired return
    $d_t^h \leftarrow d_{t-1}^h - 1$ // Update desired horizon
    $c_t \leftarrow (d_t^r, d_t^h)$
end while

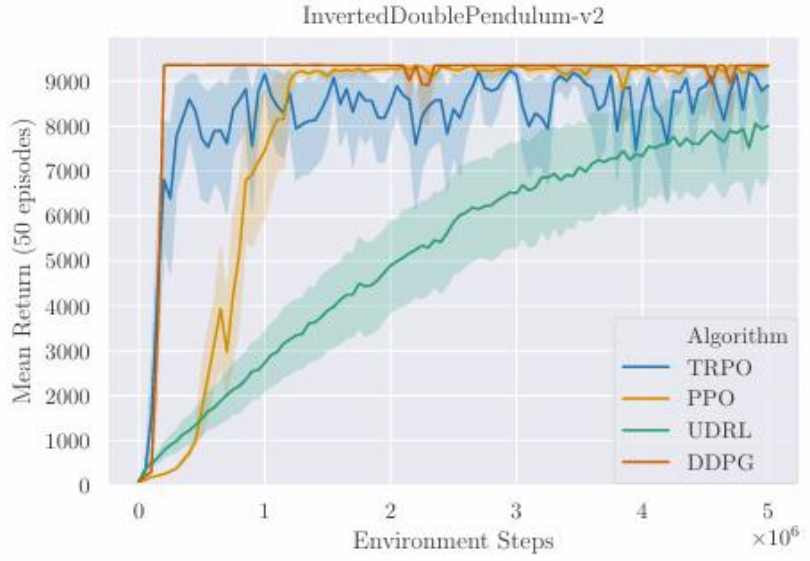
Experiments (1)

- Discrete action env
  - UDRL vs DQN vs A2C
- Continuous action env
  - UDRL vs TRPO vs PPO vs DDPG



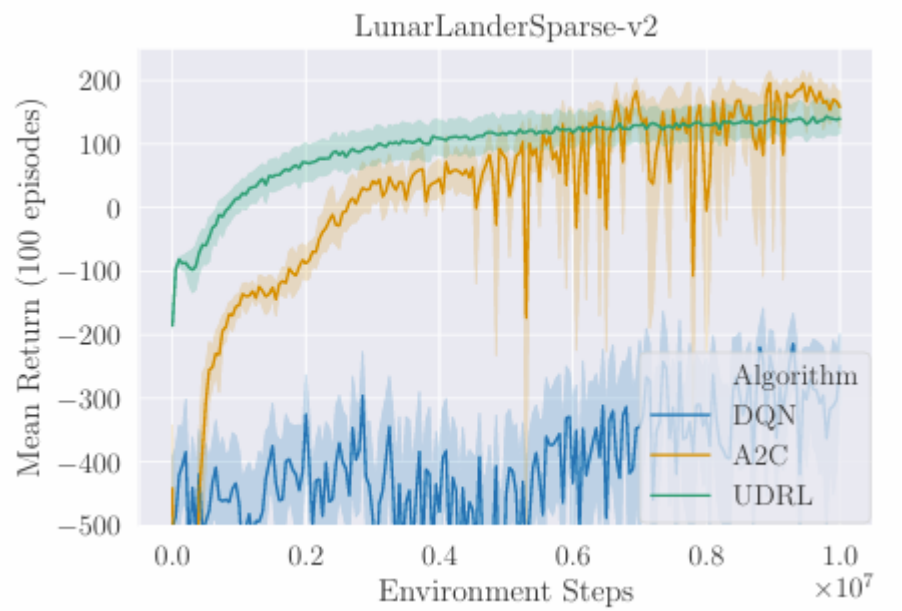


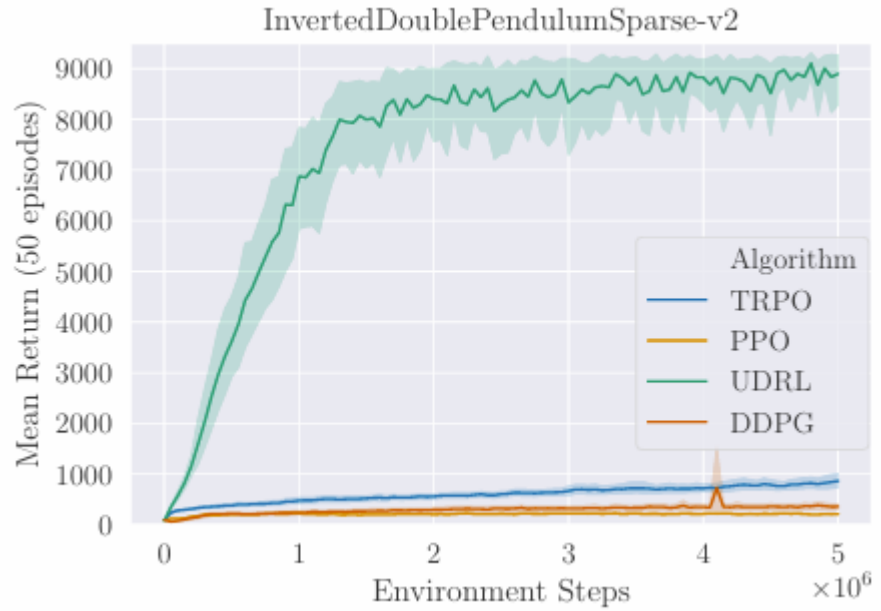




Experiments (2)

- Results on sparse delayed reward versions
  - Sparse delayed reward 환경에서도 UDRL은 별다른 수정 없이 안정적인 성능을 유지, 반면 기존 RL 알고리즘들은 불안정하거나 실패
  - UDRL은 설계상 장기적인 보상 할당(credit assignment)에 강건하여, POMDP 상황에서도 Dense reward에서의 PPO 수준의 성능에 근접





후일담

- RL문제를 SL을 활용해 기존 RL의 문제를 풀 수 있는 가능성을 보여줌
  - 물론 관련된 더 오래된 연구들이 있으며 논문에서도 자세하게 언급함
  - 그러나 SL을 보조적 수단으로 사용하는 연구들이 많은 것에 비해 이 연구는 SL을 본격적으로 RL문제에 적용하고자 하였음
- Decision transformer, Q-transformer, 더 나아가 VLA 연구에 영감을 줬다고 할 수 있음
- 강화학습연구는 앞으로 어떤 방향으로 나아갈까?
  - 脫벨만 연구들이 더 많이 나오지 않을까?
    - 벨만 기반 RL은 reward signal 없이 학습이 어렵고, sample efficiency가 낮음
  - SL을 활용하면 대규모 데이터 활용에 더 적합
    - logged dataset 기반으로 pretraining → downstream fine-tuning 가능
  - 학습 안정성과 재현성
    - 벨만 기반은 non-stationarity, off-policy instability 등의 문제가 있을 수 있음
    - supervised-style 학습은 훈련 안정성과 재현성이 높음 (특히 산업계에 중요)
  - 멀티모달 입력 통합
    - VLA(Vision-Language-Action) 모델이나 world model 기반 학습은 복잡한 high-dimensional 입력에서 벨만 방정식 사용이 어렵거나 비효율적
    - transformer 또는 imitation learning이 더 실용적
- 편견이 좀 있었지만 Schmidhuber는 훌륭한 연구자라는 것을 다시 한 번 느낌