Return Decomposition for Delayed Rewards

Une intuition sur *RUDDER*

Visionner sur *YouTube*

GLO-7030 Apprentissage par réseaux de neurones profonds [revision 3.4]

Francois-Alexandre Tremblay et **Luc Coupal**

{francois-alexandre.tremblay.1, luc.coupal.1}@ulaval.ca

F-A Luc

Université Laval

2021-04-09

RUDDER : Return Decomposition for Delayed Rewards

Arjona-Medina *et al.*, 2018

NeurIPS 2019

« We propose RUDDER, a novel reinforcement learning approach for delayed rewards in finite Markov decision processes (MDPs). RUDDER is based on two main ideas : (i) a backward view approach and (ii) the concept of return-equivalent MDPs. Forward view approaches, like deep Q-networks (DQNs) or Monte Carlo tree search (MCTS), have to average over a large number of probabilistic future state-action paths that increases exponentially with the delay of the reward. Backward view approaches, in contrast, identify actions and states that cause a delayed reward by analyzing already chosen paths. RUDDER's backward view transforms tasks of estimating future returns into regression tasks at which deep learning excels. RUDDER decomposes the return into new, non-delayed rewards by redistributing the original reward across the episode, thereby creating a new MDP that is return-equivalent to the original MDP. "Return-equivalent MDPs" is a new concept ensuring that both MDPs have the same optimal policies. If the return decomposition is optimal, then the new MDP will be stripped of any delayed rewards. In this case, action-value estimates are unbiased and the future expected return is always zero. On several artificial tasks with delayed rewards RUDDER significantly outperforms Monte Carlo, MCTS, temporal difference, TD(λ), and reward shaping approaches. RUDDER is even exponentially faster than the last three. RUDDER on top of a Proximal Policy Optimization (PPO) baseline improves the scores on Atari games and excels for delayed rewards. For long delayed rewards, as in Bowling, Frostbite, PrivateEye, and Venture, RUDDER yields exceptional results. »

Deep Reinforcement Learning (*DRL*) en soixante secondes

AlphaStar

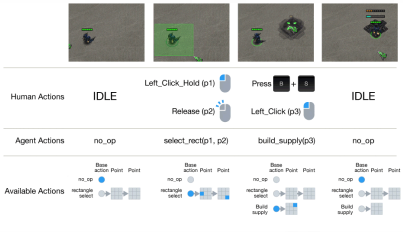


Image : DeepMind

- *DeepMind and Blizzard open StarCraft II as an AI research environment* ▸ Publication
- *StarCraft II : A New Challenge for Reinforcement Learning* [2] ▸ Publication
- *AlphaStar : Mastering the Real-Time Strategy Game StarCraft II* [3] ▸ Publication
- *AlphaStar : The inside story* ▸ Vidéo (5 min)

AlphaGo

Image : AlphaGo – The Movie

- *Mastering the game of Go with deep neural networks and tree search* [4] ▸ Publication
- *AlphaGo – The Movie* ▸ Vidéo (90 min)

Stanford University autonomous helicopter

Image : Stanford Autonomous Helicopter

- *An Application of Reinforcement Learning to Aerobatic Helicopter Flight (LQR, 2006)* [5] ▸ Publication
- *Stanford University autonomous helicopter* ▸ Projet ▸ Vidéo

Composable Deep Reinforcement Learning for Robotic Manipulation

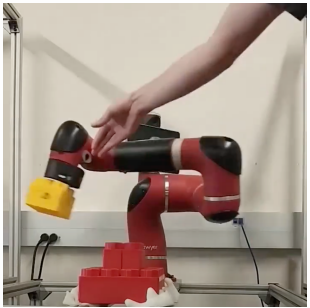
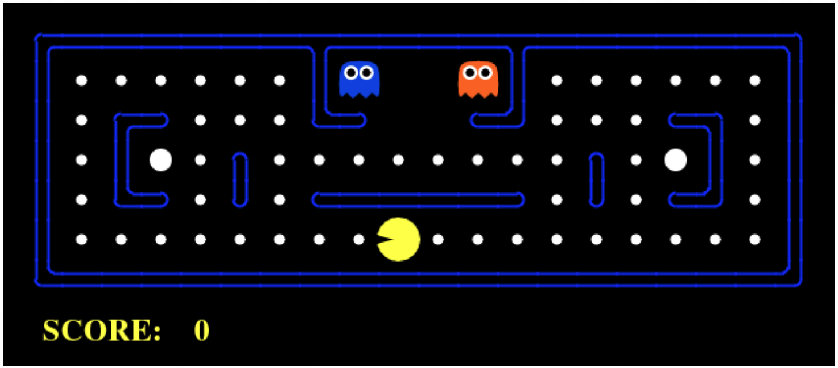


Image : Haarnoja et al.

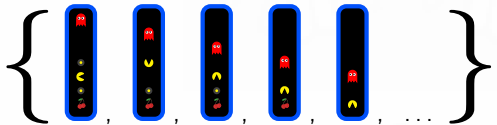
- *Composable Deep Reinforcement Learning for Robotic Manipulation* [6] ▸ Publication ▸ Vidéo
- *Reinforcement learning with deep energy-based policies* (SQL) [7] ▸ Publication



Vidéo : *UC Berkeley CS188 Intro to AI – The Pac-Man Projects*

www

Composante clés (environnement *Pac-Man*)

L'état $\mathbf{s}_t \in$ 

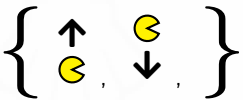
L'action $\mathbf{a}_t \in$

La récompense $r_t \in$

L'agent $:=$ « *I want more* »

Composante clés (environnement *Pac-Man*)

L'état $\mathbf{s}_t \in$

L'action $\mathbf{a}_t \in$ 

La récompense $r_t \in$

L'agent $:=$ « *I want* ***more*** »

Composante clés (environnement *Pac-Man*)

L'état $\mathbf{s}_t \in$

L'action $\mathbf{a}_t \in$

La récompense $r_t \in$ 

L'agent := « *I want more* »

Composante clés (environnement *Pac-Man*)

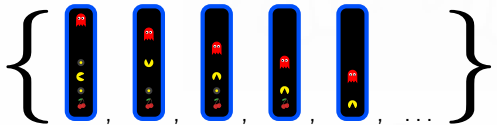
L'état $\mathbf{s}_t \in$

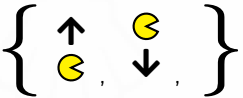
L'action $\mathbf{a}_t \in$

La récompense $r_t \in$

L'agent  $:=$  « *I want* ***more***  »

Composante clés (environnement *Pac-Man*)

L'état $\mathbf{s}_t \in$ 

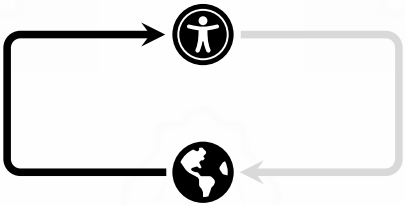
L'action $\mathbf{a}_t \in$ 

La récompense $r_t \in$ 

L'agent  $:=$  *« I want **more** »*

Temps $t = 1$

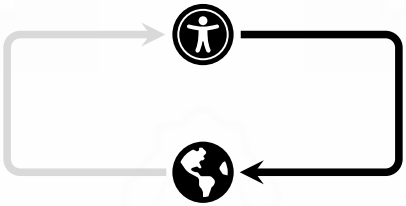
$= \mathbf{s}_1$



$$\text{trajectoire} = \left(\quad , \quad \right)$$

Temps $t = 1$

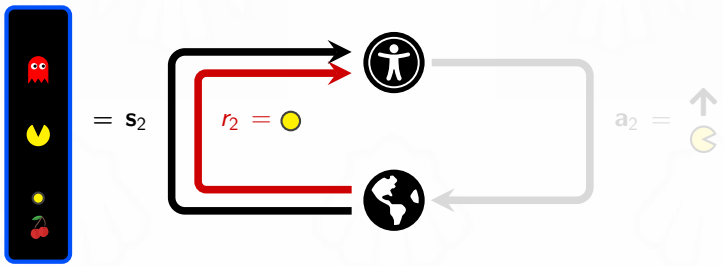
$= \mathbf{s}_1$



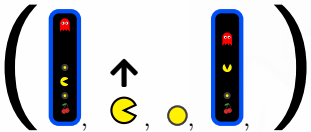
$\mathbf{a}_1 =$ 

$$\text{trajectoire} = \left(\text{  }, \text{  }, \right)$$

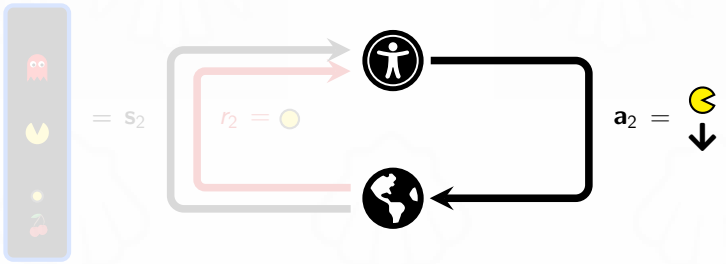
Temps $t = 2$



trajectoire =

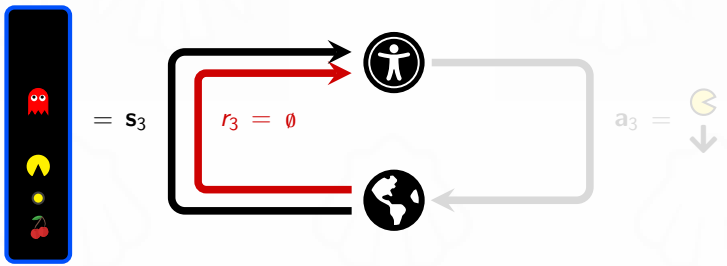


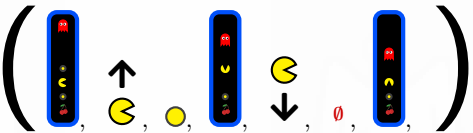
Temps $t = 2$



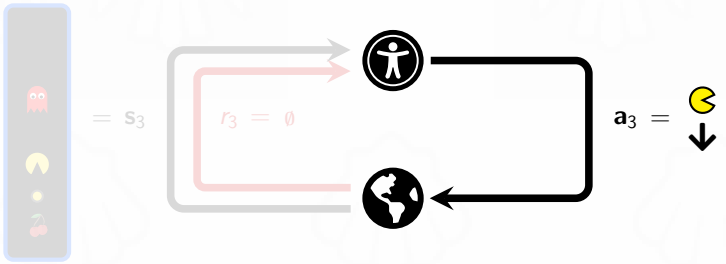
$$\text{trajectoire} = \left(\quad , \quad , \quad , \quad , \quad \right)$$

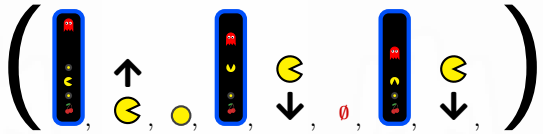
Temps $t = 3$



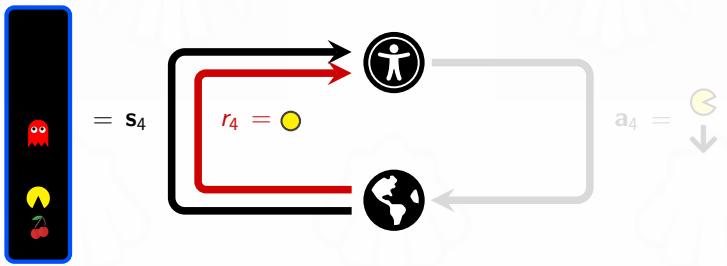
trajectoire $=$ 

Temps $t = 3$



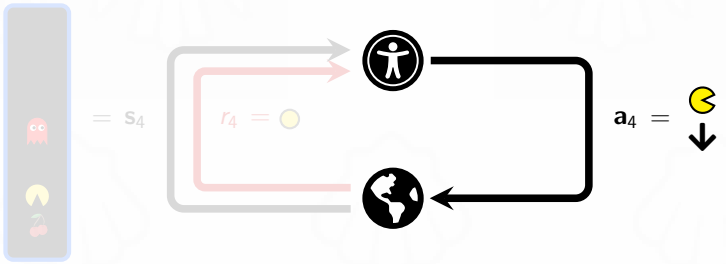
trajectoire $=$ 

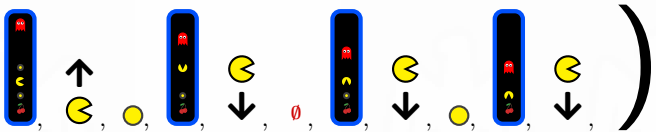
Temps $t = 4$



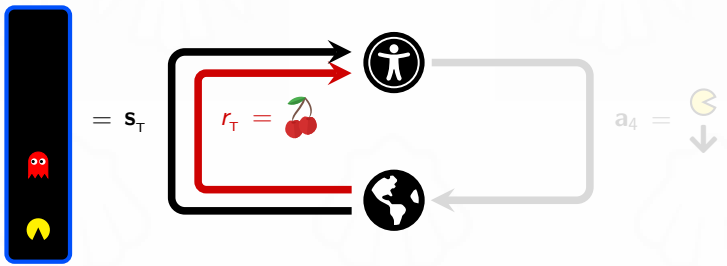
$$\text{trajectoire} = \Big(\; , \uparrow, \; , \; , \downarrow, \; , \emptyset, \; , \downarrow, \; , \; , \; \Big)$$

Temps $t = 4$

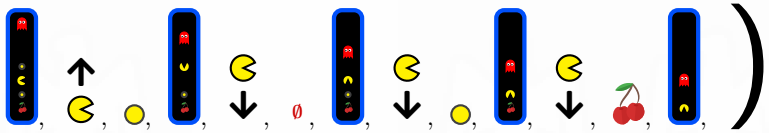


trajectoire $=$ 

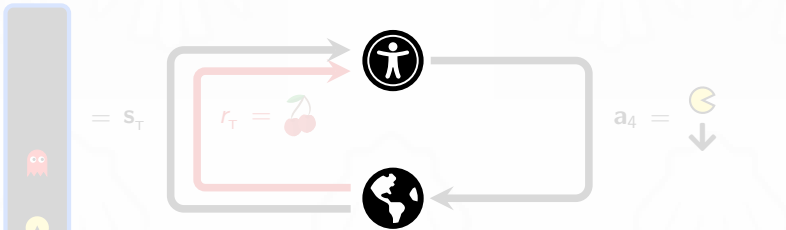
Temps $t = 5$

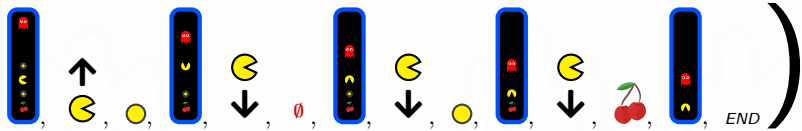


trajectoire $=$



Temps $t = 5$



trajectoire $=$ 

Natural Language Processing (NLP)

$$\left[\texttt{"I"}, \texttt{"love"}, \texttt{"whisky"}, \texttt{"a"}, \texttt{"lot"} \right]$$

Reinforcement Learning (RL)

$$\mathbf{s}_1, \ \mathbf{a}_1, \ r_2, \ \mathbf{s}_2, \ \mathbf{a}_2, \ r_3, \ \ldots \ , \mathbf{s}_T, \ \mathbf{a}_T, \ r_{T+1}$$

Natural Language Processing (NLP)

$$f\Big(\big[\texttt{"I"},\ \texttt{"love"},\ \texttt{"whisky"},\ \texttt{"a"},\ \texttt{"lot"}\big]\ ;\ \theta\Big) \longmapsto \texttt{"Good for you"}$$

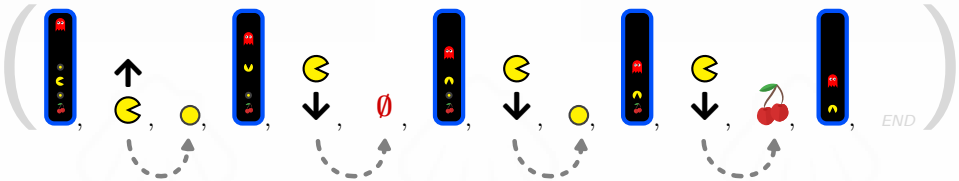
Reinforcement Learning (RL)

$$f\Big(\mathbf{s}_1,\ \mathbf{a}_1,\ r_2,\ \mathbf{s}_2,\ \mathbf{a}_2,\ r_3,\ \ldots\ ,\mathbf{s}_T,\ \mathbf{a}_T,\ r_{T+1}\ ;\ \theta\Big) \longmapsto \widehat{\pi}_\theta(\mathbf{a}_t|\mathbf{s}_t)$$

Problématique

Le problème des récompenses non immédiates

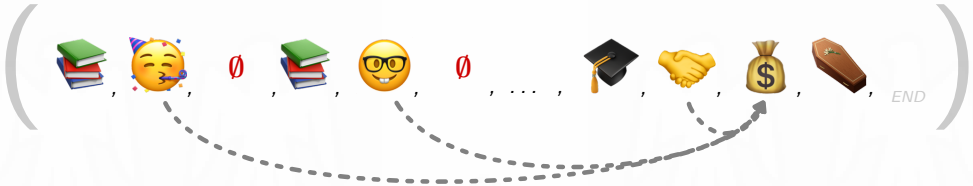
Trajectoire *Pac-Man*



Le problème des récompenses non immédiates

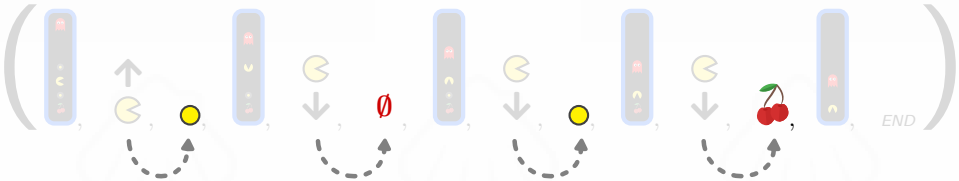
Trajectoire *Pac-Man*

Trajectoire universitaire

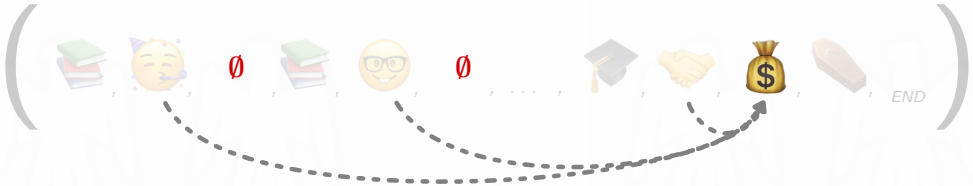


Le problème des récompenses non immédiates

Trajectoire *Pac-Man*



Trajectoire universitaire



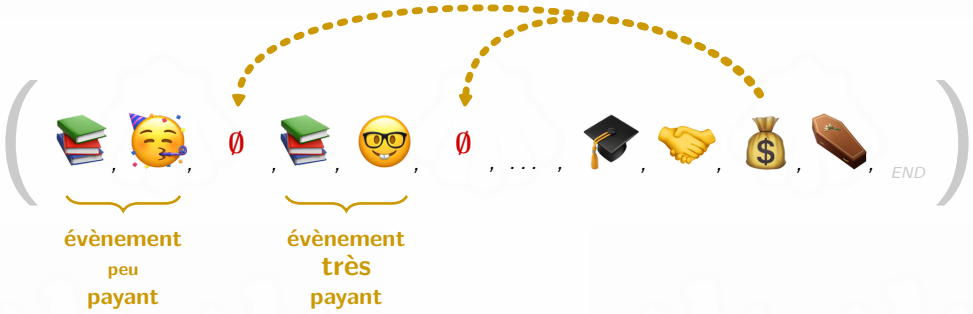
Redistribution des récompenses

Redistribution des récompenses

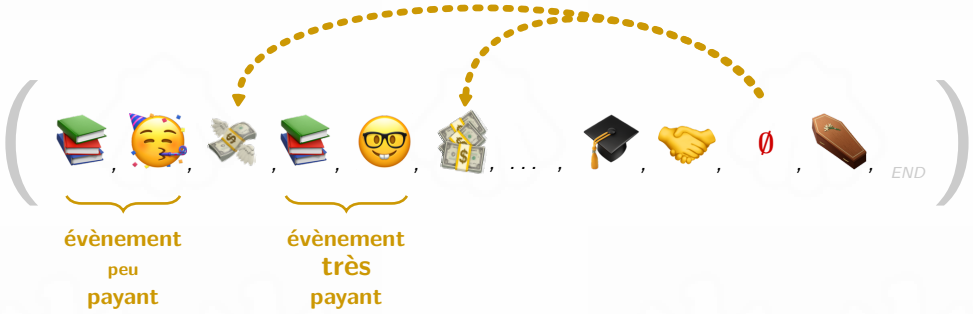
Redistribution des récompenses

Redistribution des récompenses

Redistribution des récompenses



Redistribution des récompenses

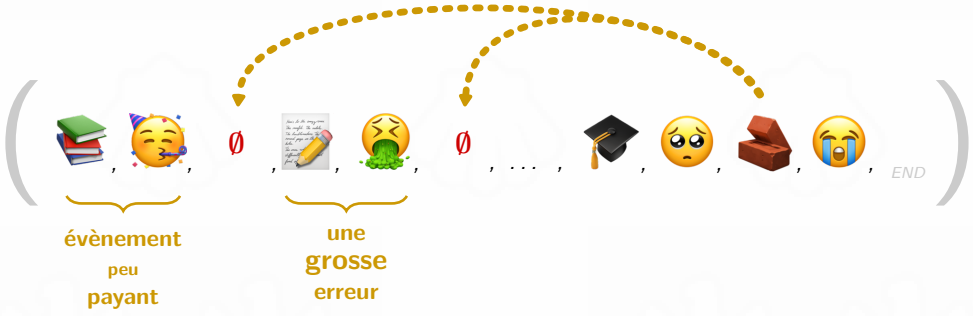


Redistribution des récompenses

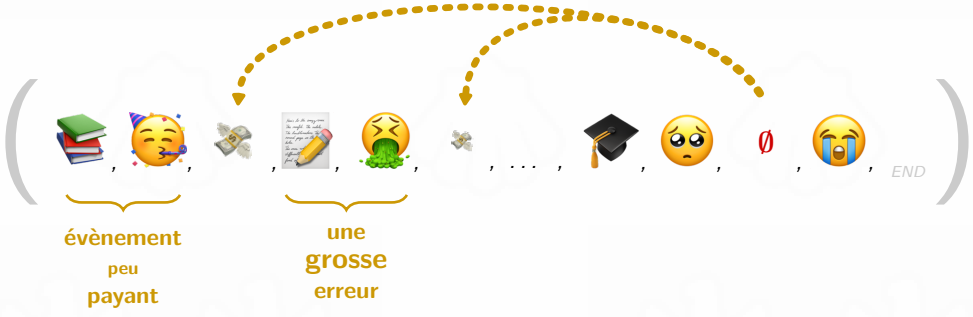
Redistribution des récompenses

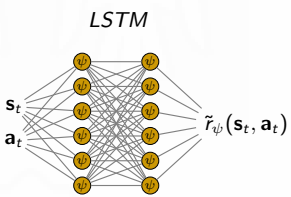
Redistribution des récompenses

Redistribution des récompenses



Redistribution des récompenses

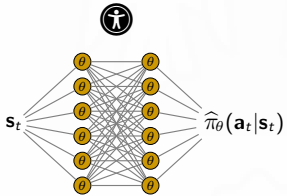




RUDDER

$$f\Big(\ \mathbf{s}_1,\ \mathbf{a}_1,\ r_2,\ \mathbf{s}_2,\ \mathbf{a}_2,\ r_3,\ \mathbf{s}_3,\ \ldots\ ,\mathbf{s}_T,\ \mathbf{a}_T,\ r_{T+1}\ ;\ \psi\ \Big)\ \longmapsto\ \tilde{r}_\psi(\mathbf{s}_t, \mathbf{a}_t)$$

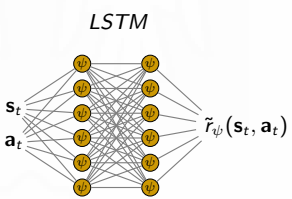
$$f\Big(\ \mathbf{s}_1,\ \mathbf{a}_1,\ r_2,\ \mathbf{s}_2,\ \mathbf{a}_2,\ r_3,\ \mathbf{s}_3,\ \ldots\ ,\mathbf{s}_T,\ \mathbf{a}_T,\ r_{T+1}\ ;\ \theta\ \Big)\ \longmapsto\ \widehat{\pi}_\theta(\mathbf{a}_t|\mathbf{s}_t)$$

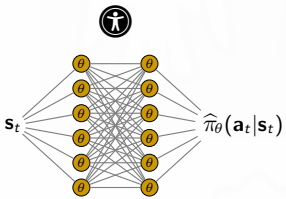


Reinforcement Learning (RL) *policy gradient version*

$$f\Big(\ s_1,\ a_1,\ r_2,\ s_2,\ a_2,\ r_3,\ s_3,\ \ldots\ ,s_T,\ a_T,\ r_{T+1}\ ;\ \psi\ \Big)\ \longmapsto\ \tilde{r}_\psi(s_t, a_t)$$

$$f\Big(\ s_1,\ a_1,\ r_2,\ s_2,\ a_2,\ r_3,\ s_3,\ \ldots\ ,s_T,\ a_T,\ r_{T+1}\ ;\ \theta\ \Big)\ \longmapsto\ \widehat{\pi}_\theta(a_t|s_t)$$



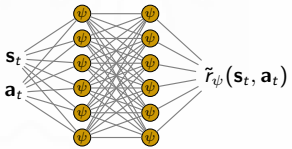


Reinforcement Learning* (*RL*) + *RUDDER

$$f\Big(\ \mathbf{s}_1,\ \mathbf{a}_1,\ r_2,\ \mathbf{s}_2,\ \mathbf{a}_2,\ r_3,\ \mathbf{s}_3,\ \ldots\ ,\mathbf{s}_T,\ \mathbf{a}_T,\ r_{T+1}\ ;\ \psi\ \Big)\ \longmapsto\ \tilde{r}_\psi(\mathbf{s}_t, \mathbf{a}_t)$$

$$f\Big(\ \mathbf{s}_1,\ \mathbf{a}_1,\ \tilde{r}_\psi(\mathbf{s}_1, \mathbf{a}_1),\ \mathbf{s}_2,\ \ldots\ ,\mathbf{s}_T,\ \mathbf{a}_T,\ \tilde{r}_\psi(\mathbf{s}_T, \mathbf{a}_T)\ ;\ \theta\ \Big)\ \longmapsto\ \widehat{\pi}_\theta(\mathbf{a}_t|\mathbf{s}_t)$$

LSTM



Reinforcement Learning* (*RL*) + *RUDDER

$$f\Big(\ \mathbf{s}_1,\ \mathbf{a}_1,\ r_2,\ \mathbf{s}_2,\ \mathbf{a}_2,\ r_3,\ \mathbf{s}_3,\ \ldots\ ,\mathbf{s}_T,\ \mathbf{a}_T,\ r_{T+1}\ ;\ \psi\ \Big) \longmapsto \tilde{r}_\psi(\mathbf{s}_t, \mathbf{a}_t)$$

$$f\Big(\ \mathbf{s}_1,\ \mathbf{a}_1,\ \tilde{r}_\psi(\mathbf{s}_1, \mathbf{a}_1),\ \mathbf{s}_2,\ \ldots\ ,\mathbf{s}_T,\ \mathbf{a}_T,\ \tilde{r}_\psi(\mathbf{s}_T, \mathbf{a}_T)\ ;\ \theta\ \Big) \longmapsto \widehat{\pi}_\theta(\mathbf{a}_t|\mathbf{s}_t)$$

Résultats expérimentaux

Tâche de neutralisation d'explosif

-2	-3	-4	-5	-5	-5
-2	-3	-4	-5	1000	-5
-2	-3	-4	-5	-5	-5
-2	-3	-4	-4	-4	-4
-2	-3	-3	-3	-3	-3
	-2	-2	-2	-2	-2

Exemple d'environnement *GridWorld*

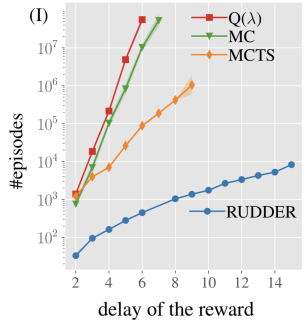


Image : ***RUDDER*** *- Reinforcement Learning with Delayed Rewards,* Arjona-Medina *et al.* [1]

Idées clés

⚠ Les problèmes de *RL* avec **récompenses à retardement** sont **difficiles** à solutionner.

RL + RUDDER

- Trajectoires $\longmapsto$ ***LSTM*** $\longmapsto$ Redistribution de récompenses
- 👍 **Réussit** là où les autres algorithmes de *RL* échouent
- 👍 **Performance** accrue

⚠ Les problèmes de *RL* avec **récompenses à retardement** sont **difficiles** à solutionner.

RL + *RUDDER*

- Trajectoires $\longmapsto$ ***LSTM*** $\longmapsto$ Redistribution de récompenses
- 👍 **Réussit** là où les autres algorithmes de *RL* échouent
- 👍 **Performance** accrue

Limites du *LSTM*

- 👎 Requiert un jeu de données de trajectoires a priori
- 👎 Performance diminue plus les trajectoires sont longues
- 👎 Temps d'entraînement supplémentaire

Références I

1. Arjona-Medina, J. A. *et al.* Rudder: Return decomposition for delayed rewards. *arXiv.* ISSN : 23318422. arXiv : 1806.07857 (2018).
2. Vinyals, O. *et al.* StarCraft II: A New Challenge for Reinforcement Learning. arXiv : 1708.04782. http://arxiv.org/abs/1708.04782 (2017).
3. DeepMind. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. *DeepMind.* https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/ (2019).
4. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529,** 484-489. ISSN : 14764687. http://dx.doi.org/10.1038/nature16961 (2016).
5. Abbeel, P., Coates, A., Quigley, M. & Ng, A. Y. An Application of Reinforcement Learning to Aerobatic Helicopter Flight. *Advances in Neural Information Processing Systems* (2006).
6. Haarnoja, T. *et al.* *Composable Deep Reinforcement Learning for Robotic Manipulation.* in *Proceedings - IEEE International Conference on Robotics and Automation* (2018). ISBN : 9781538630815. arXiv : 1803.06773.

Références II

7. Haarnoja, T., Tang, H., Abbeel, P. & Levine, S. Reinforcement learning with deep energy-based policies. *34th International Conference on Machine Learning, ICML 2017* **3,** 2171-2186. arXiv : 1702.08165 (2017).

8. Arjona-Medina, J. A. *et al.* SUPPLEMENTS to the manuscript " RUDDER : Return Decomposition for Delayed Rewards ".