Techniques d'apprentissage

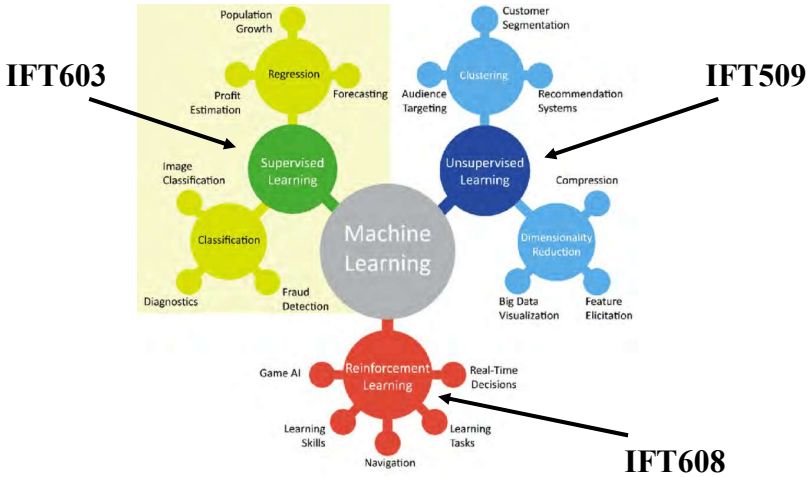
IFT 603 / IFT 712

Introduction aux méthodes par renforcement

Par
Pierre-Marc Jodoin

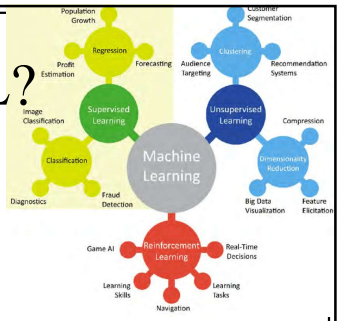
1

Pourquoi une intro au RL?



2

Pourquoi une intro au RL?



Bien que les méthodes de RL s'appuient sur un formalisme différent des techniques d'apprentissage, certaines méthodes de RL utilisent [presque] directement des méthodes de techniques d'apprentissage vues en ift603/712 (en particulier les réseaux de neurones).

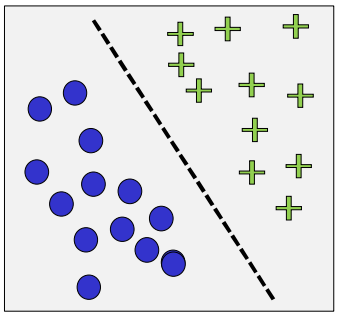
Ce chapitre est une **courte introduction au RL** afin de démontrer **l'étendu des applications de techniques d'apprentissage**

3

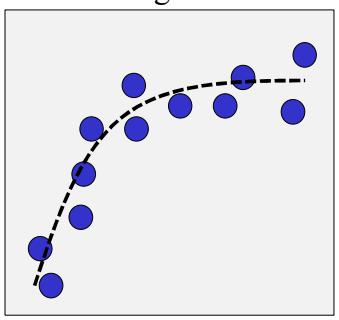
3

Jusqu'à présent : apprentissage supervisé

Classification



Régression



4

4

Supervisé vs non supervisé

Apprentissage supervisé : il y a une cible

$$D = \{(\vec{x}_1, t_1), (\vec{x}_2, t_2), \ldots, (\vec{x}_N, t_N)\}$$

Apprentissage non-supervisé : la cible n'est pas fournie

$$D = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$$

5

5

Apprentissage par RL

La différence entre un apprentissage par RL vs les deux autres familles sont:

- A priori, pas de base de données (annotée ou non)
- C'est remplacé par
  - Interaction avec un **environnement**
  - La notion de « **récompense** »
  - La notion « **d'agent** » et de « **politique** » au lieu de « **modèle** »

6

6

Qu'est-ce qu'un agent?

Un **modèle** en apprentissage supervisé, est un système entraîné avec des **données annotées** et une **fonction de perte** et dont l'objectif est de produire un **résultat** pour **chaque donnée en entrée**.

Un **agent** d'apprentissage par renforcement (RL) est un système qui **évolue dans un environnement** et apprend à prendre de **nombreuses décisions** en interagissant avec son environnement afin de maximiser une **récompense cumulative**.

7

7

Apprentissage par renforcement profond

NOTE: lorsque les actions de l'agent découlent d'un **réseau de neurones**, on parle de ***« deep reinforcement learning »***

8

8

Au menu du jour…

- Les bases du formalisme *deep* RL
- *Imitation learning* et *Dagger*
- *Policy gradient* et l'algorithme REINFORCE

9

9

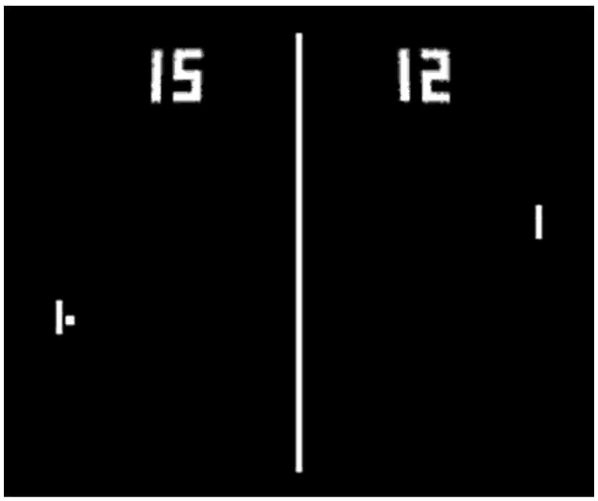
Agent et applications RL

10

10

Exemple 1

Jeu pong

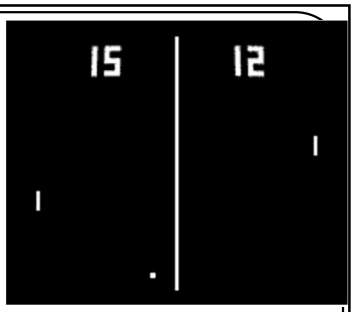


11

11

Exemple 1

Jeu pong



Agent : raquette

Actions de l'agent : {monter, descendre, arrêt}

Environnement (état) : balle, les raquettes

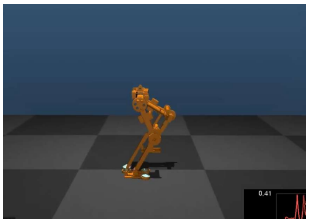
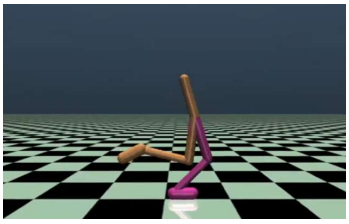
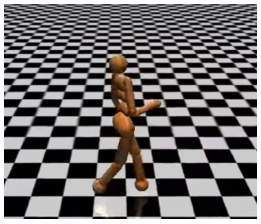
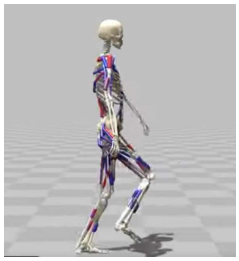
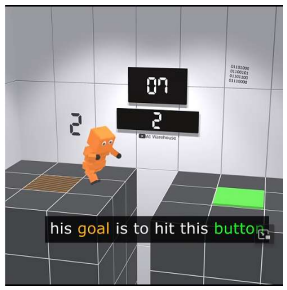
Récompense : ex. toucher à la balle

12

12

Exemple 2

Robot marcheur

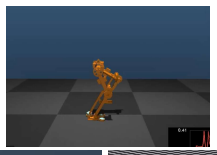
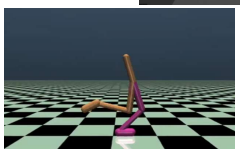






13

13

Exemple 2

Robot marcheur


Agent : robot

Actions de l'agent : pivoter les X joints de son corps

Environnement (état) : plancher, gravité, marcheur

Récompense : ex.: marcher sans tomber

14

14

Exemple 3

Conduite autonome

15

15

Exemple 3

Conduite autonome

Agent : auto

Actions de l'agent : tourner le volant entre –X, 0 et +X degrés

Environnement (état) : route, autres autos, piétons, panneaux, etc.

Récompense : ex. aller du point A au point B sans accident.

16

16

Agent vs politique

Agent : l'entité qui interagit avec l'environnement.

Politique : C'est le "cerveau" de l'agent qui dicte quelle action prendre en fonction d'un état.

Ex: conduite autonome

Agent : voiture (roues, moteur, transmission, carrosserie, radio, etc.)

Politique : processus d'ajustement de la vitesse + direction de l'auto en fonction de l'état observé

17

17

Notions de base en RL

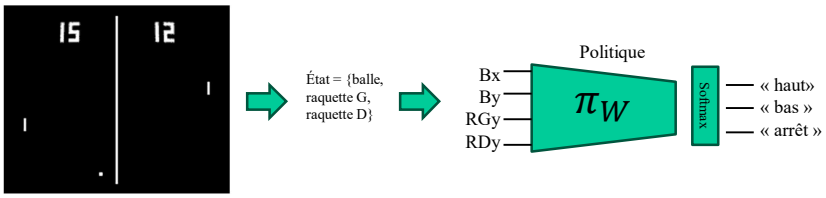
18

18

Notions de base

Nous verrons ici le cas pour lequel **la politique est un réseau de Neurones** qui prend en **entrée un état** et retourne une **action**

Ex. Pong, l'agent est la raquette gauche



19

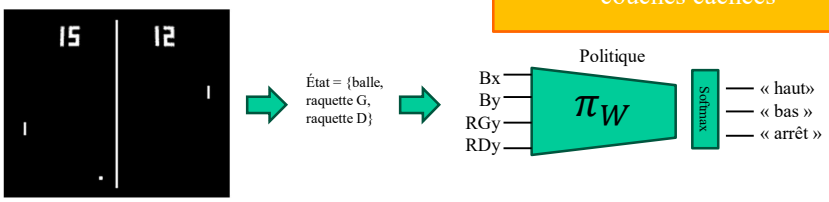
19

Notions de base

Nous verrons ici le cas pour lequel **l'agent est un réseau de neurones** qui prend en **entrée un état** et retourne une **action**

Ex. Pong, l'agent est la raquette gauche

Dans ce cas, le réseau pourrait être un MLP de **classification** à X couches cachées



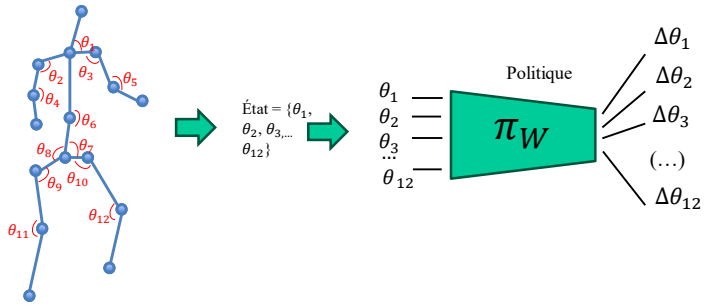
20

20

Notions de base

Nous verrons ici le cas pour lequel **l'agent est un réseau de neurones** qui prend en **entrée un état** et retourne une **action**

Ex. Agent marcheur,



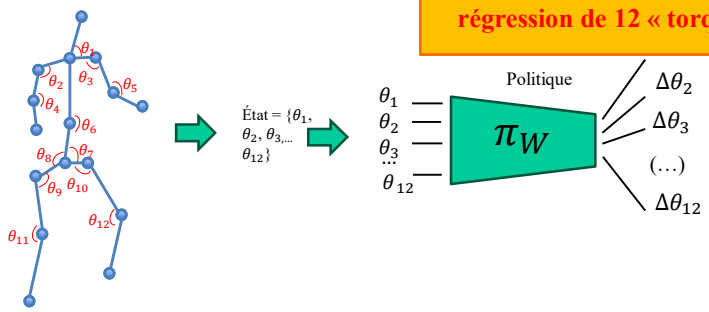
21

Notions de base

Nous verrons ici le cas pour lequel **l'agent est un réseau de neurones** qui prend en **entrée un état** et retourne une **action**

Ex. Agent marcheur,

Dans ce cas, le réseau pourrait être un MLP à X couches cachées de **régression de 12 « torques »**



22

Notions de base

Nous verrons ici le cas pour lequel **l'agent est un réseau de neurones** qui prend en **entrée un état** et retourne une **action**

Ex. Voiture autonome,

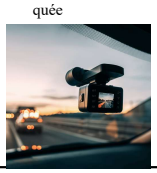
23

Notions de base

Nous verrons ici le cas pour lequel **l'agent est un réseau de neurones** qui prend en **entrée un état** et retourne une **action**

Ex. Voiture autonome,

Ça pourrait être **un réseau à convolution** de **classification** qui prédit un delta de l'angle du volant



24

Formalisme RL

Dans tous les cas, on a

- Un environnement $e \in E$
- Un état $s \in S$
- Une action $a \in A$
- Une récompense $r \in R$
- La politique d'un agent π_W (réseau de neurones) $\pi_W(s) \rightarrow a$

25

Formalisme RL

Dans tous les cas, on a

- Un environnement $e \in E$
- Un état $s \in S$
- Une action $a \in A$
-
-

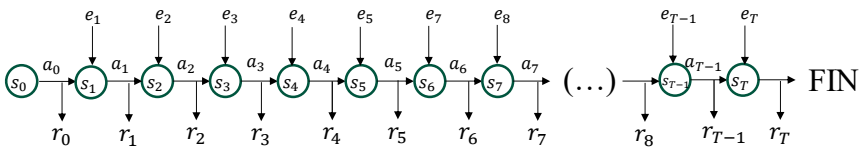
Puisqu'un réseau de neurone prédit souvent des probabilités conditionnelles, on écrit en RL $\boldsymbol{\pi_W(a|s)}$ au lieu de $\boldsymbol{\pi_W(s) \rightarrow a}$

26

Formalisme RL

Et tout cela s'exprime dans une **séquence/épisode** temporel

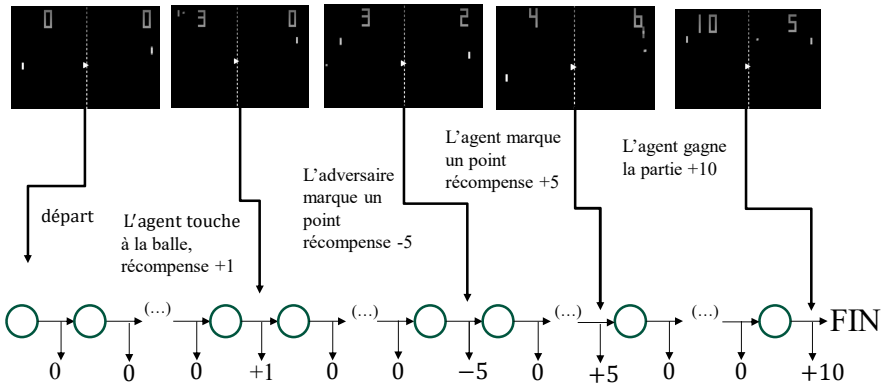
$$\pi_W(a|s)$$



27

Formalisme RL

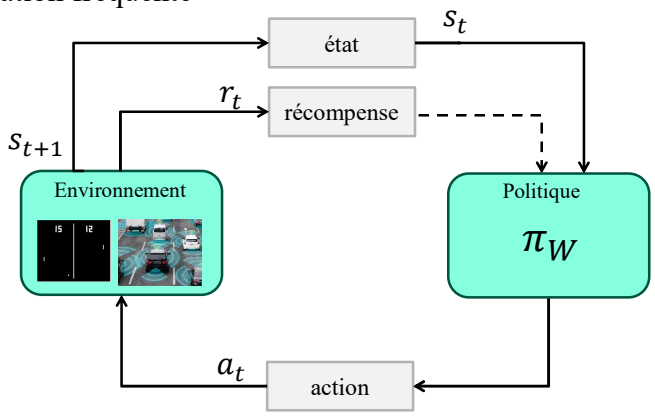
Exemple avec Pong



28

Formalisme RL

Illustration fréquente



29

La grande question en RL est … comment entraîner un agent ?

(i.e. le réseau de neurones)

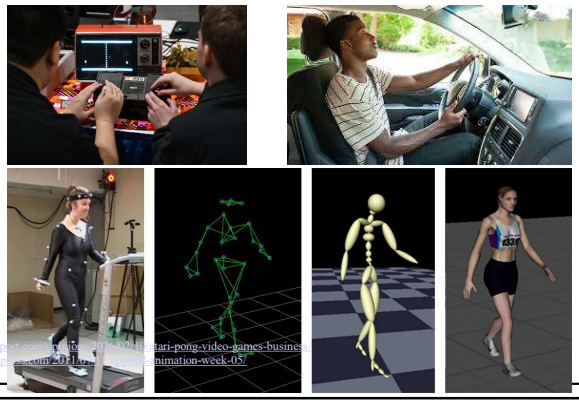
30

30

L'approche la plus simple :
Imitation learning

L'idée

1. On élimine la notion de récompense
2. On demande à des humains de créer un ensemble d'entraînement



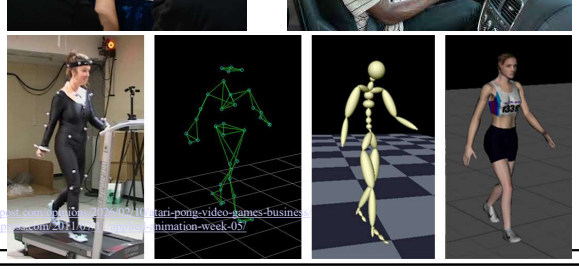
https://www.washingtonpost.com/[illegible]/atari-pong-video-games-busines
https://casfxblock2.word[illegible]nimation-week-05/

L'approche la plus simple :
Imitation learning

L'idée

1. On élimine la notion de récompense
2. On demande à des humains de créer un ensemble d'entraînement

Apprentissage supervisé!!



https://www.washingtonpost.com/[illegible]/atari-pong-video-games-busines
https://casfxblock2.word[illegible]nimation-week-05/

Imitation learning

Approche 1 : ***Behaviour cloning***

1. On demande à des humains de performer la tâche (conduire, jouer au jeu, etc.)
2. Pour chaque état s_t on enregistre l'action a_t qu'a fait l'humain
3. On crée un ensemble d'entraînement

$$D_{train} = \{(s_0, a_0), (s_1, a_1), \dots, (s_T, a_T)\}$$

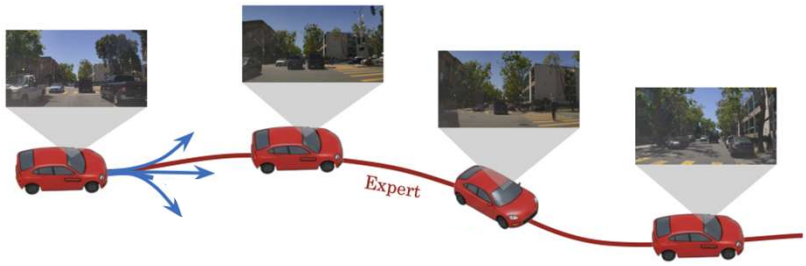
4. On entraîne la politique $\pi_W(s)$ comme un réseau de neurones de base (ex. TP3, TP4)

33

33

Imitation learning

Approche 1 : ***Behaviour cloning***



https://research.nvidia.com/publication/2025-12_beyond-behavior-cloning-autonomous-driving-survey-closed-loop-training

34

34

Imitation learning

Approche 1 : ***Behaviour cloning***

Problème 1

hypothèse que les données utilisées à l'entraînement (issues d'un expert) ont la même distribution que celles lors du déploiement de l'agent. C'est faux!

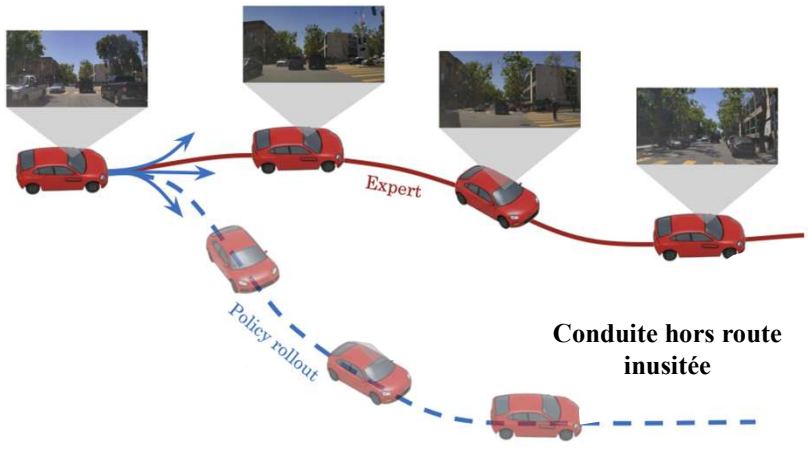
- une petite erreur peut amener l'agent dans un état jamais observé.
- L'agent ne sait pas comment réagir, ce qui entraîne de nouvelles erreurs qui s'accumulent progressivement

https://research.nvidia.com/publication/2025-12_beyond-behavior-cloning-autonomous-driving-survey-closed-loop-training 35

35

Imitation learning

Approche 1 : ***Behaviour cloning***



https://research.nvidia.com/publication/2025-12_beyond-behavior-cloning-autonomous-driving-survey-closed-loop-training 36

36

Imitation learning

Approche 1 : ***Behaviour cloning***

Problème 2

L'agent se contente d'imiter l'expert plutôt que de maximiser une récompense. L'agent ne peut donc pas dépasser la performance de l'expert et reproduit ses erreurs si celui-ci est sous-optimal.

37

37

Imitation learning

Approche 1 : ***Behaviour cloning***

Apprentissage d'une politique peu performante

Apprentissage d'une politique très performante

38

38

Imitation learning

Dagger (Dataset Aggregation)

(une solution pour améliorer le *behaviour cloning)*

But: créer D_{train} à partir d'épisodes générer par $\pi_W(s)$ au lieu d'un humain

1. Entraîner π_W avec l'algo de *behaviour cloning*
2. Exécuter π_W et collecter les états $D = \{s_0, s_1, s_2, \dots, s_T\}$
3. Demander à un humain d'annoter D avec les bonnes actions a_t
4. $D_{train} = D_{train} \cup D$
5. Entraîner π_W avec D_{train}

39

39

Imitation learning

Dataset Aggregation (DAgger)

Est-ce que ça fonctionne? Oui!

Kendall et al. *Learning to Drive in a Day IEEE International Conference on Robotics and Automation (ICRA)*, 2019

40

40

Imitation learning

Dataset Aggregation (DAgger)

Autre exemple

Giusti et al. *A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots,* IEEE Robotics and Automation Letters, 2015

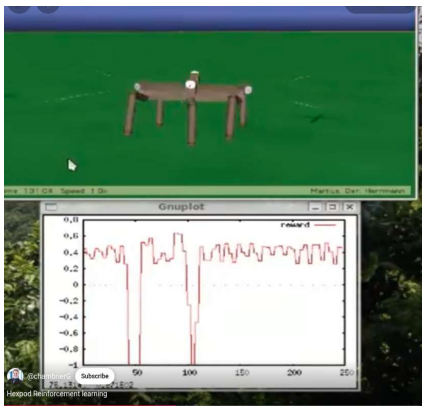
41

41

DAgger

Malheureusement, l'annotation de données par des humains est souvent pénible, coûteuse et **parfois impossible.**

Exemple, faire marcher un robot araignée ayant 24 degrés de liberté.



42

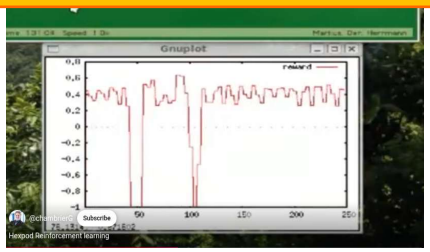
42

DAgger

Malheureusement, l'annotation de données par des humains est souvent pénible, coûteuse et **parfois impossible.**

Exe de liberté.

Comment un humain peut savoir quel angle donné aux joints à tous les instants?



43

43

Solution!

1. Abandonner l'idée de construire une BD d'entrainement annotée par des humains
2. Optimiser la recompense.

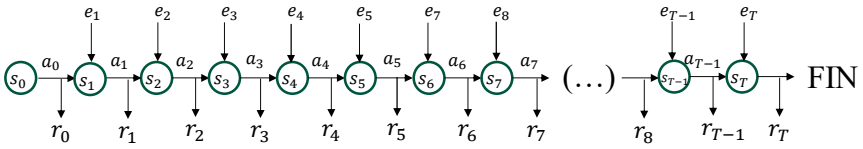
44

44

Formalisme RL

$\pi_W(a|s)$

Soit un **épisode** temporel τ

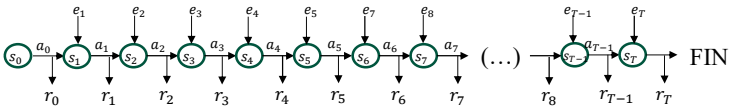


La récompense totale pour cet épisode τ est

$$\mathrm{R}(\tau) = \sum_{t=0}^{T} r(s_t, a_t)$$

45

Formalisme RL

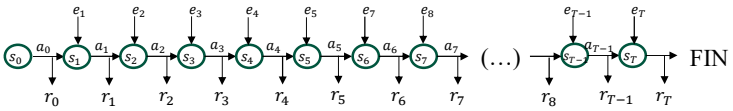


L'objectif est d'avoir une politique dont la **récomprense totale** est **maximale**

$$\mathrm{R}(\tau) = \sum_{t=0}^{T} r(s_t, a_t)$$

46

Formalisme RL



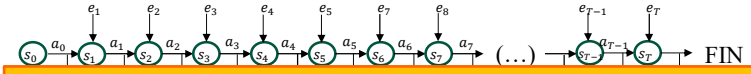
Deux observations :

1. Il n'y a pas qu'un seul episode τ, il y en a **une infinité**
2. La recompense (ex. +1 si l'agent touche à la balle dans pong) n'est **pas differentiable.**

$$\mathrm{R}(\tau) = \sum_{t=0}^{T} r(s_t, a_t)$$

47

Formalisme RL



Aille! Comment optimiser une fonction cible qu'on ne peut pas différencier?

té

ns

pong) n'est **pas differentiable.**

$$\mathrm{R}(\tau) = \sum_{t=0}^{T} r(s_t, a_t)$$

48

Formalisme RL

Une infinité d'épisodes

La suite d'état et d'actions dans un épisode est déterminée par la politique et l'environnement. Même en partant d'un même état initial (s_t) le contenu de l'épisode est déterminé par la politique et l'environnement.

ex: Conduite autonome : différents niveaux de trafic, présence inattendue de cyclistes, etc.
Ex: Pong : Meilleur adversaire, différentes stratégies de jeu, etc.
Ex: Marcheur: différents obstacles, différents terrains, neige, glace, etc.

49

Formalisme RL

Une infinité d'épisodes

Ce qu'on veut, c'est une politique qui performe bien en **moyenne** sur tous les épisodes pouvant être générés par la politique

Espérance mathématique

$$W = \text{argmax}_W \underbrace{E_{\tau \sim \pi_W}[R(\tau)]}_{L(W):\, loss \text{ à maximiser}}$$

50

Formalisme RL

La récomprense n'est pas différentiale

Puisque la politique est un réseau de neurones, il nous faut **calculer le gradient de la loss** par rapport à ses poids. Mais comment faire si la récompense n'est pas différentiable?

Solution : le gradient de la politique π_W (*policy gradient*)

51

Formalisme RL

Le gradient de la politique π_W (*policy gradient*)

$$L(W) = \mathrm{E}_{\tau \sim \pi_W}[R(\tau)]$$
$$= \int_\tau \pi_W(\tau) R(\tau) d\tau$$

$$\nabla_W L(W) = \int_\tau \nabla_W \pi_W(\tau) R(\tau) d\tau$$

52

Formalisme RL

Le gradient de la politique π_W (*policy gradient*)

$$\nabla_W L(W) = \int_\tau \overbrace{\nabla_W \pi_W(\tau) R(\tau) d\tau}$$

Nous avons toujours pris le gradient de la *loss* et jamais le gradient du modèle

53

Formalisme RL

Le gradient de la politique π_W (*policy gradient*)

$$\nabla_W L(W) = \int_\tau \overbrace{\nabla_W \pi_W(\tau) R(\tau) d\tau}$$

Que signifie le gradient du modèle? À quoi ça sert?

54

Note : dérivée de la fonction log

Propriété intéressante. Sachant que

$$\frac{d\ln(x)}{dx} = \frac{1}{x}$$

Par la propriété de la dérivée en chaîne

$$\frac{d\ln(f(x))}{dx} = \frac{d\ln(f(x))}{df(x)}\frac{df(x)}{dx}$$

$$= \frac{1}{f(x)}\frac{df(x)}{dx}$$

On obtient que

$$\frac{df(x)}{dx} = f(x)\frac{d\ln(f(x))}{dx}$$

55

55

Note : dérivée de la fonction log

Propriété intéressante. Lorsque x est un vecteur

$$\nabla f(\vec{x}) = f(\vec{x})\ln(f(\vec{x}))$$

56

56

Note : dérivée de la fonction log

Propriété intéressante. Appliquée au gradient de la politique $\nabla_W \pi_W(\tau)$

$$\nabla_W \pi_W(\tau) = \pi_W(\tau) \ln(\pi_W(\tau))$$

57

57

Formalisme RL

Le gradient de la politique π_W (*policy gradient*)

$$\nabla_W L(W) = \int_\tau \nabla_W \pi_W(\tau) R(\tau) d\tau$$

$$= \int_\tau \pi_W(\tau) \nabla_W \ln(\pi_W(\tau)) R(\tau) d\tau$$

Puisque $\pi_W(\tau)$ peut être vu comme comme une probabilité (softmax)

$$= \mathrm{E}_{\tau \sim \pi_W}[\underbrace{\nabla_W \ln(\pi_W(\tau))}_{\text{Équation du gradient de l'entropie croisée (tp3 et tp4)}} R(\tau)]$$

58

58

Gradient de la politique *(Policy gradient)*

Algorithme REINFORCE

POUR *N* itérations FAIRE

$\tau \leftarrow$ Exécuter la politique π_W *1* fois

Calculer $R(\tau)$

$\nabla_W L(W) \leftarrow \emptyset$

POUR s_1 à s_T FAIRE /* chaque état */

$\nabla_W L(W) \mathrel{+}= \left(\nabla_W \ln(\pi_W(a_t|s_t)\right) R(\tau)$

$W = W + \eta \nabla_W L(W)$

59

59

Très similaire à un entraînement supervisé

Algorithme REINFORCE

POUR *N* itérations FAIRE

$\tau \leftarrow$ Exécuter la politique π_W une fois

Calculer $R(\tau)$

$\nabla_W L(W) \leftarrow \emptyset$

POUR s_1 à s_T FAIRE /* chaque état */

$\nabla_W L(W) \mathrel{+}= \left(\nabla_W \ln(\pi_W(a_t|s_t)\right) R(\tau)$

$W = W + \eta \nabla_W L(W)$

Algorithme d'entraînement supervisé d'un réseau de neurones

$D_{train} = \{(x_1, t_1), (x_2, t_2), \ldots, (x_T, t_T)\}$

POUR *N* epochs FAIRE /* chaque Epoch */

$\nabla_W L(W) \leftarrow \emptyset$

POUR les *M* éléments de la batch FAIRE

$\nabla_W L(W) \mathrel{+}= \left(\nabla_W \ln(\pi_W(t_m|x_m)\right)$

$W = W - \eta \nabla_W L(W)$

60

60

Gradient de la politique *(Policy gradient)*

(Autre illustration du même algo, N=1 pour algo précédent)

Algorithm 1 REINFORCE Algorithm

1: Initialize a policy π_θ
2: **while** NOT stopping criterion **do**
3: Collect N trajectories $\{\tau^i\}$ from the environment using π_θ
4: Calculate $\nabla_\theta \mathcal{J}(\pi_\theta)$ using Equation (16)
5: Update $\theta = \theta + \alpha \nabla_\theta \mathcal{J}(\pi_\theta)$
6: **end while**

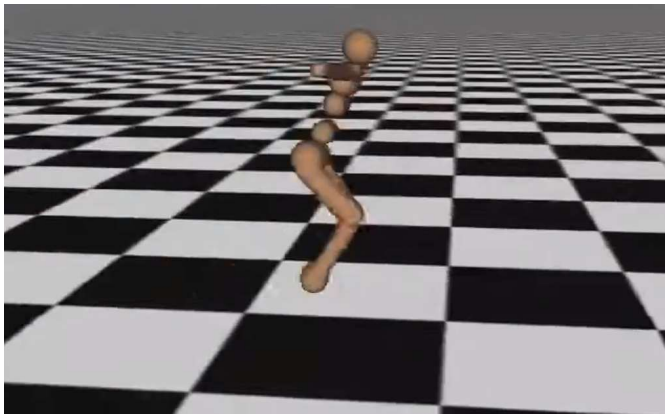
from Sutton Barto book: Introduction to Reinforcement Learning

61

61

Illustration du gradient de la politique avec un marcheur

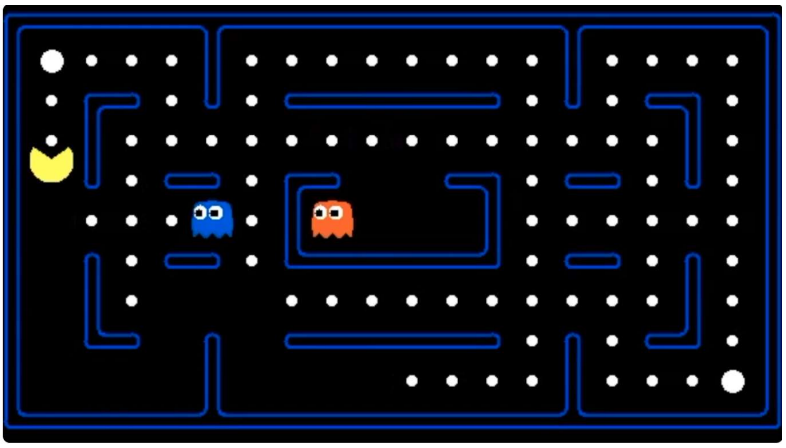
Ici, la politique est améliorée par ascension du gradient à chaque 10 épisodes



62

62

Illustration de RL avec PacMan



63

63

Forces du RL

- Pas besoin de données annotées
- Pas besoin d'une fonction différentiable (reward)
- Peut obtenir des performances supérieures à celles des humains. Ex, AlphaGo

64

64

Inconvénients du RL

- Essaie – erreur, difficile/dangereux dans la réalité (besoin d'un simulateur)
- Pas évident de trouver la bonne fonction de récompenses
- *Reward hacking*

65

65

Reward hacking

La grande peur des scientifiques.

Phénomène où un agent exploite des failles ou des bogues dans sa fonction de récompense et/ou de l'environnement afin de **maximiser son score** sans **réellement atteindre l'objectif visé.**

Ex:
Reward d'un marcheur = avancer le plus vite possible
Hack: le marcheur apprend à sauter au lieu de marcher

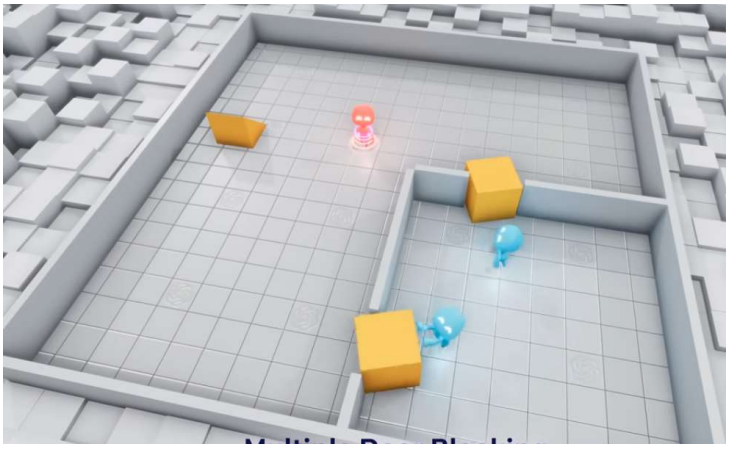
Reward d'un agent transactionnel = acheter un objet X sur internet au prix le plus bas
Hack: hacker Amazon, changer le prix, et acheter à rabais.

66

66

Illustration de *reward hacking*

Exploitation d'un bug dans un jeu



67

67

Illustration de *reward hacking*

Maximise les points bonus sans jamais terminer la course

68

68

Ressources

- https://lilianweng.github.io/posts/2018-02-19-rl-overview
- https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c
- https://web.stanford.edu/class/cs237b/pdfs/lecture/lecture_10111213.pdf
- https://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_2_behavior_cloning.pdf
- https://jonathan-hui.medium.com/rl-policy-gradients-explained-9b13b688b146

69