

Réseaux de neurones

IFT 780

Apprentissage par renforcement

Par

Antoine Théberge

Jusqu'à présent : apprentissage supervisé

Classification

Données: tuples (x,y)

x : images (p.e.)

y : classe

But: maximiser $p(y|x)$



Jusqu'à présent : apprentissage auto-supervisé

Génération

Données: seulement x

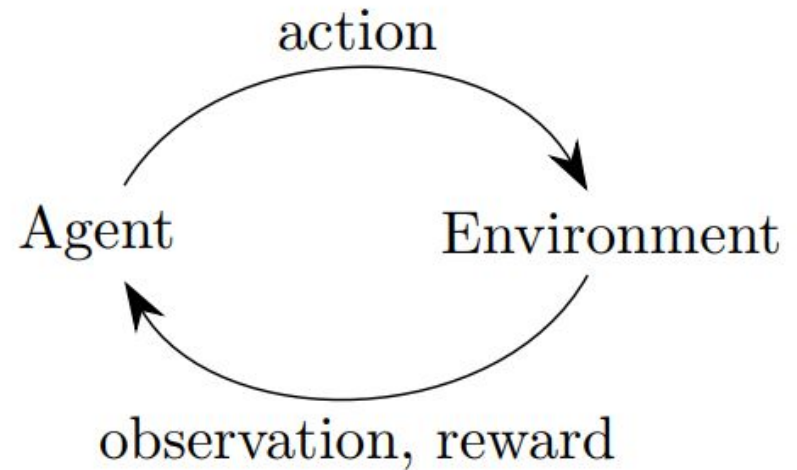
But: apprendre des caractéristiques utiles de x



Maintenant: apprentissage par renforcement (AR)

Données: tuples (s,a,r)

But: apprendre une politique qui maximisera r



Apprentissage par renforcement

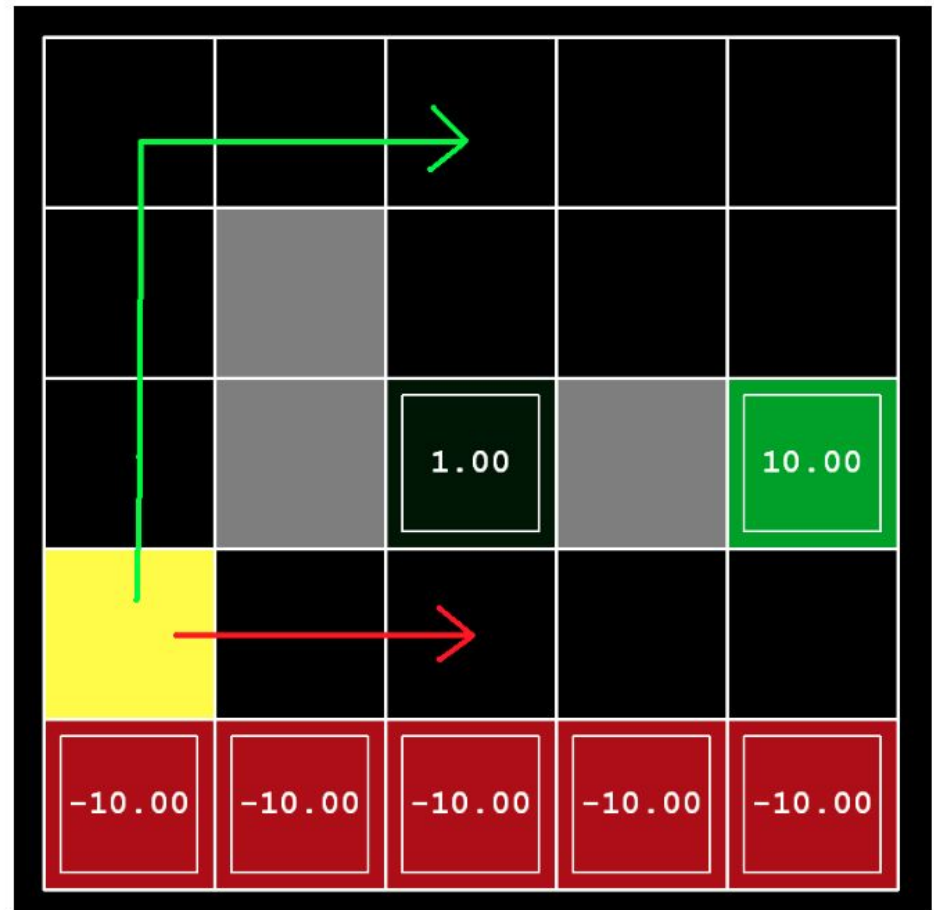
Exemple: “gridworld”

États: chaque position

Actions:

Haut/Bas/Gauche/Droite

Récompense: Indiqué à chaque case, 0 ailleurs



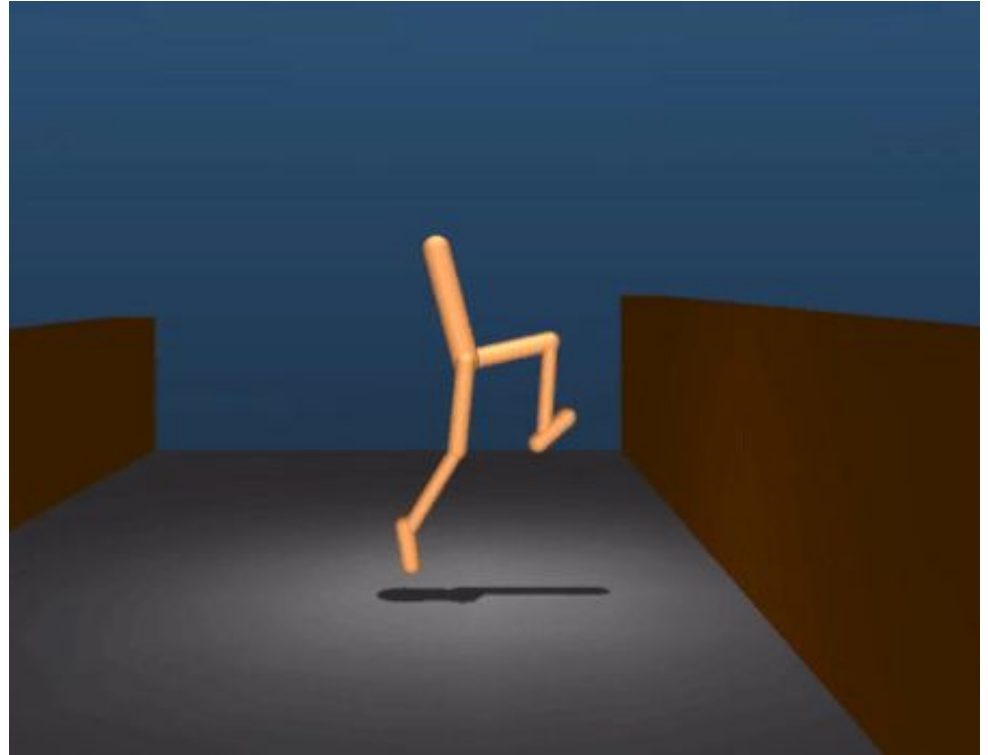
Apprentissage par renforcement

Exemple: Contrôle
robotique

États: Position des joints

Actions: Couple appliqué à
chaque joints

Récompense: +1 à chaque
“timestep” debout +
vélocité frontale



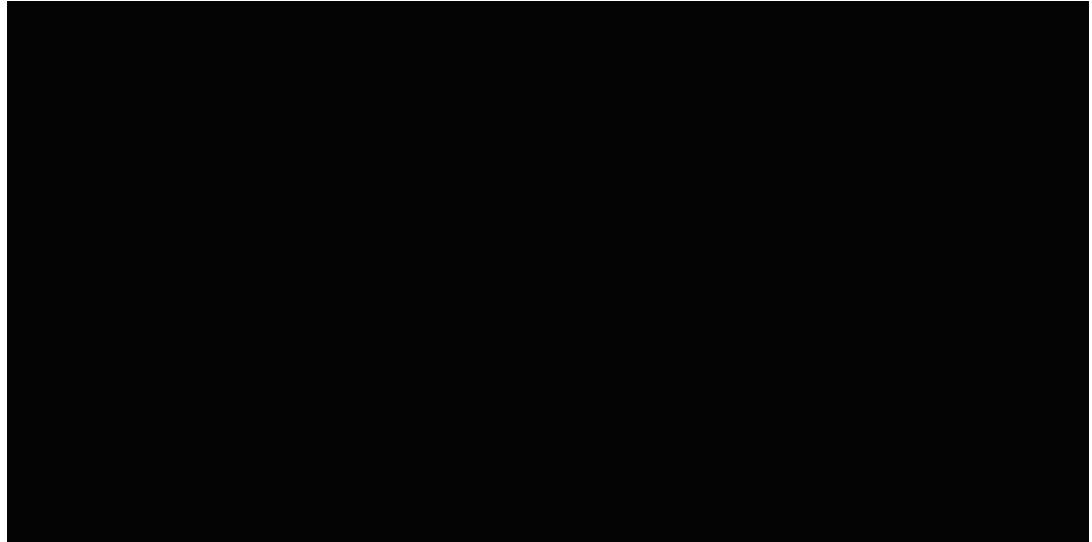
Apprentissage par renforcement

Exemple: Manipulation
robotique

États: Pixels de la caméra

Actions: Couple appliqué à
chaque joints

Récompense: Hauteur de la
pile



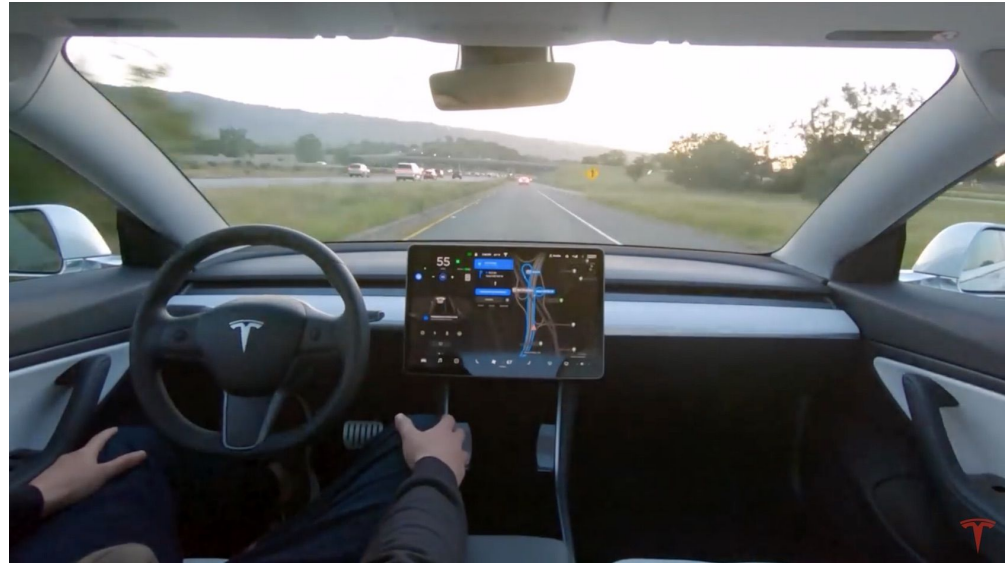
Apprentissage par renforcement

Exemple: Conduite autonome

États: Entrée de senseurs/caméras

Actions: Contrôles de la voiture

Récompense: Respecter le code de la route, arriver à destination



Apprentissage par renforcement

Exemple: Jeux vidéos

États: Pixels à l'écran

Actions: Boutons du
contrôleur

Récompense: Fluctuations
du score



Apprentissage par renforcement

Exemple: Jeux vidéos

États: Pixels à l'écran

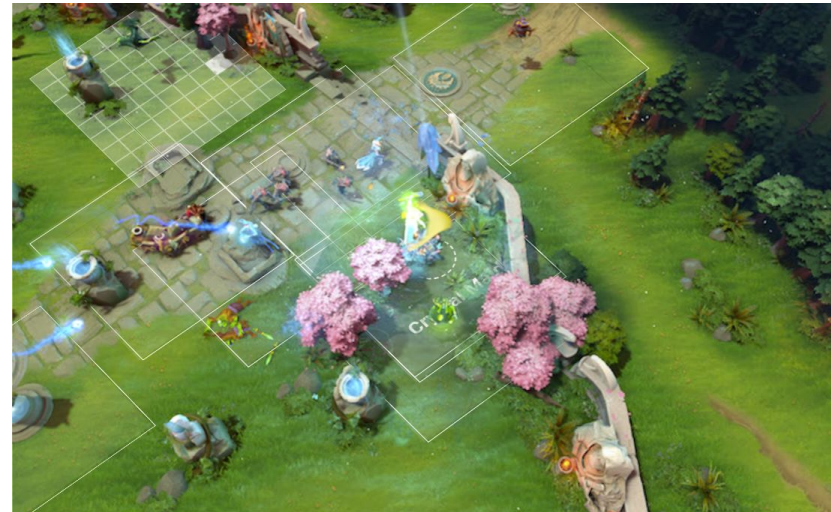
Actions: Clavier + souris

Récompense:

Gagner/Perdre la partie



<https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii>



<https://openai.com/blog/openai-five/>

Apprentissage par renforcement

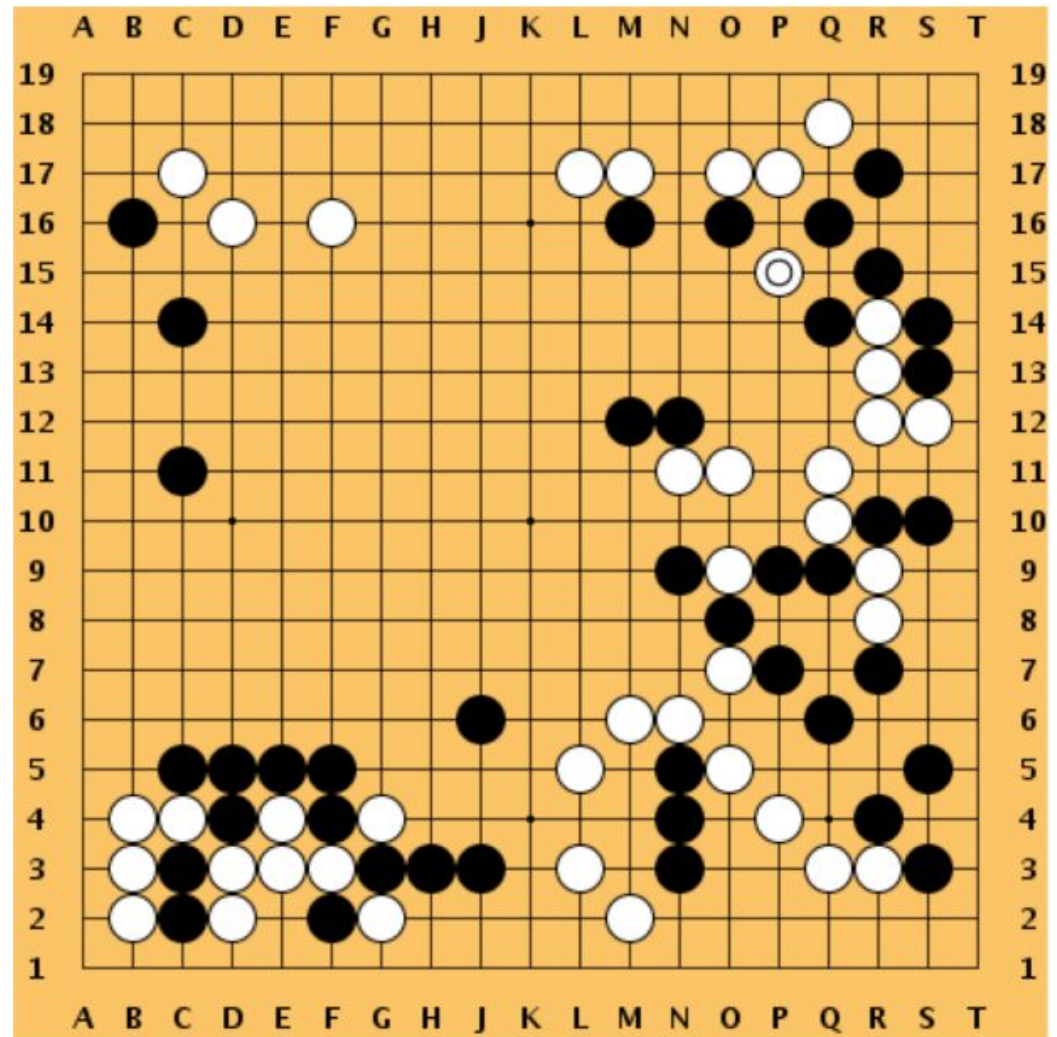
Exemple: Jeux de table

États: Planche de jeu

Actions: Placer une pierre

Récompense:

Gagner/Perdre la partie



Markov decision process (MDP)

Extension des chaînes de markov (S, A, R, P, γ)

S: Ensemble d'états (*states*) $s \in S$

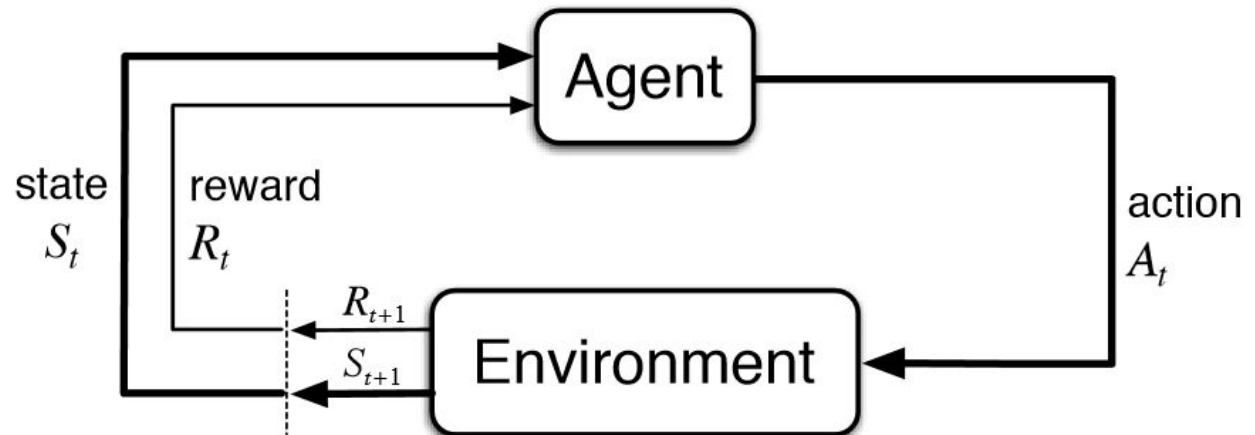
A: Ensemble d'actions $a \in A$

R: Fonction de récompense $r : S \times A \rightarrow \mathbb{R}; r(s_t, a_t)$

P: Fonction de transition $p : S \times A \rightarrow S; p(s_{t+1} | s_t, a_t)$

γ : Facteur de récompense $0 < \gamma < 1$

π : Politique $S \rightarrow A; a \leftarrow \pi(s_t)$



Markov decision process (MDP)

Extension des chaînes de markov (S, A, R, P, γ)

S: Ensemble d'états (*states*) $s \in S$

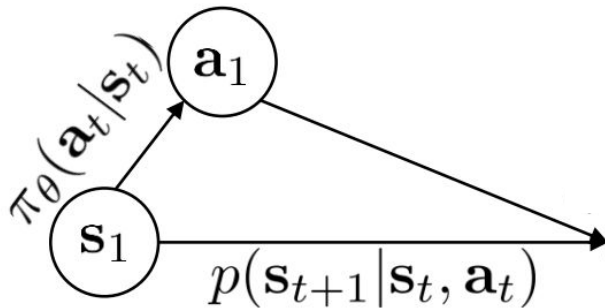
A: Ensemble d'actions $a \in A$

R: Fonction de récompense $r : S \times A \rightarrow \mathbb{R}; r(s_t, a_t)$

P: Fonction de transition $p : S \times A \rightarrow S; p(s_{t+1} | s_t, a_t)$

γ : Facteur de récompense $0 < \gamma < 1$

π : Politique $S \rightarrow A; a \leftarrow \pi(s_t)$



Markov decision process (MDP)

Extension des chaînes de markov (S, A, R, P, γ)

S: Ensemble d'états (*states*) $s \in S$

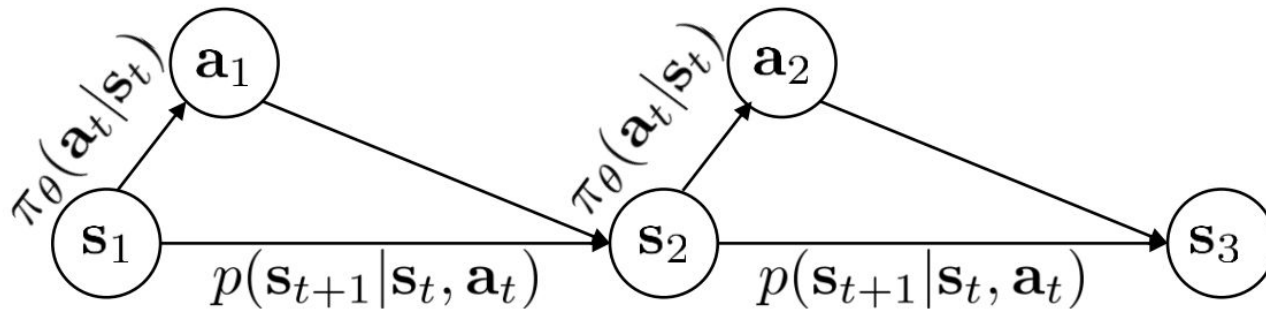
A: Ensemble d'actions $a \in A$

R: Fonction de récompense $r : S \times A \rightarrow \mathbb{R}; r(s_t, a_t)$

P: Fonction de transition $p : S \times A \rightarrow S; p(s_{t+1} | s_t, a_t)$

γ : Facteur de récompense $0 < \gamma < 1$

π : Politique $S \rightarrow A; a \leftarrow \pi(s_t)$



Markov decision process (MDP)

Extension des chaînes de markov (S, A, R, P, γ)

S: Ensemble d'états (*states*) $s \in S$

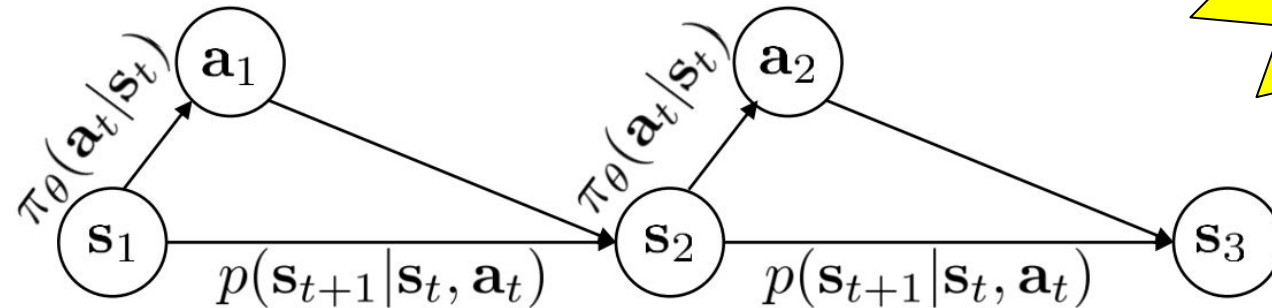
A: Ensemble d'actions $a \in A$

R: Fonction de récompense $r : S \times A \rightarrow \mathbb{R}; r(s_t, a_t)$

P: Fonction de transition $p : S \times A \rightarrow S; p(s_{t+1} | s_t, a_t)$

γ : Facteur de récompense $0 < \gamma < 1$

π : Politique $S \rightarrow A; a \leftarrow \pi(s_t)$



Propriété de
Markov

Markov decision process (MDP)

Extension des chaînes de markov (S, A, R, P, γ)

S: Ensemble d'états (*states*) $s \in S$

A: Ensemble d'actions $a \in A$

R: Fonction de récompense $r : S \times A \rightarrow \mathbb{R}; r(s_t, a_t)$

P: Fonction de transition $p : S \times A \rightarrow S; p(s_{t+1} | s_t, a_t)$

γ : Facteur de récompense $0 < \gamma < 1$

π : Politique $S \rightarrow A; a \leftarrow \pi(s_t)$

Objectif: Trouver la politique maximisant le *retour* espéré

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{s, a \sim \pi} \left[\sum_t^T \gamma^t r(s_t, a_t) \right]$$

Exemple: Gridworld

Actions:

- Haut, bas, gauche, droite

Récompense:

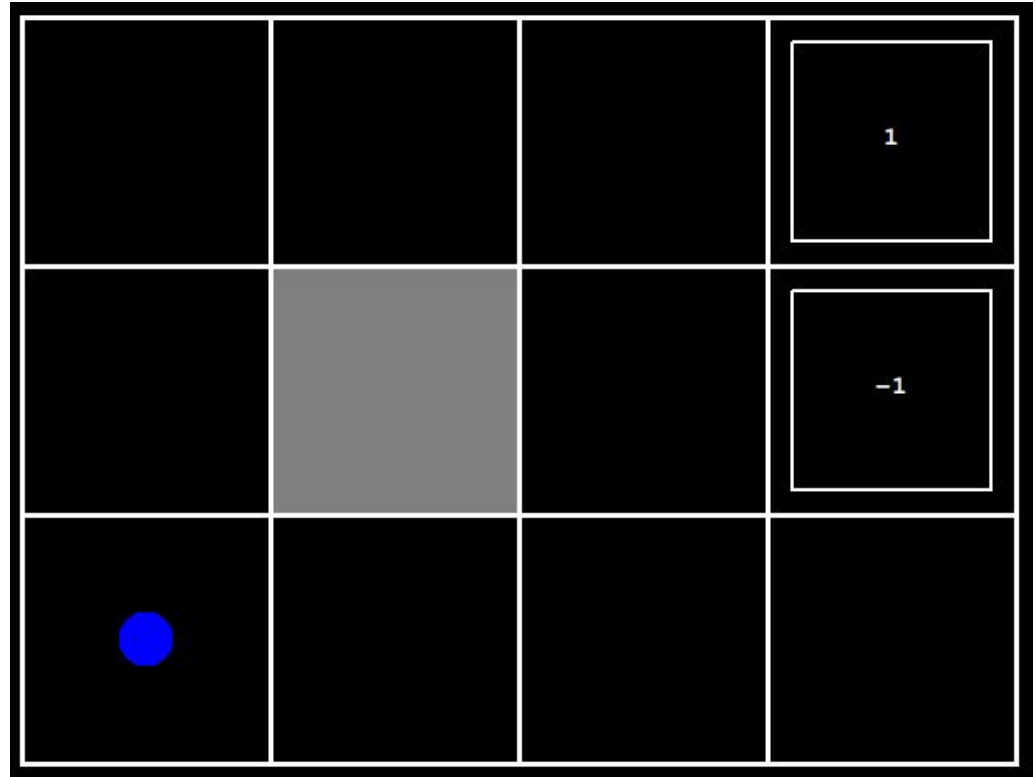
- +1, -1, 0 ailleurs

États:

- Position actuelle

Transitions:

- 0.5 selon l'action
- 0.5 aléatoire



Exemple: Gridworld

Actions:

- Haut, bas, gauche, droite

Récompense:

- +1, -1, 0 ailleurs

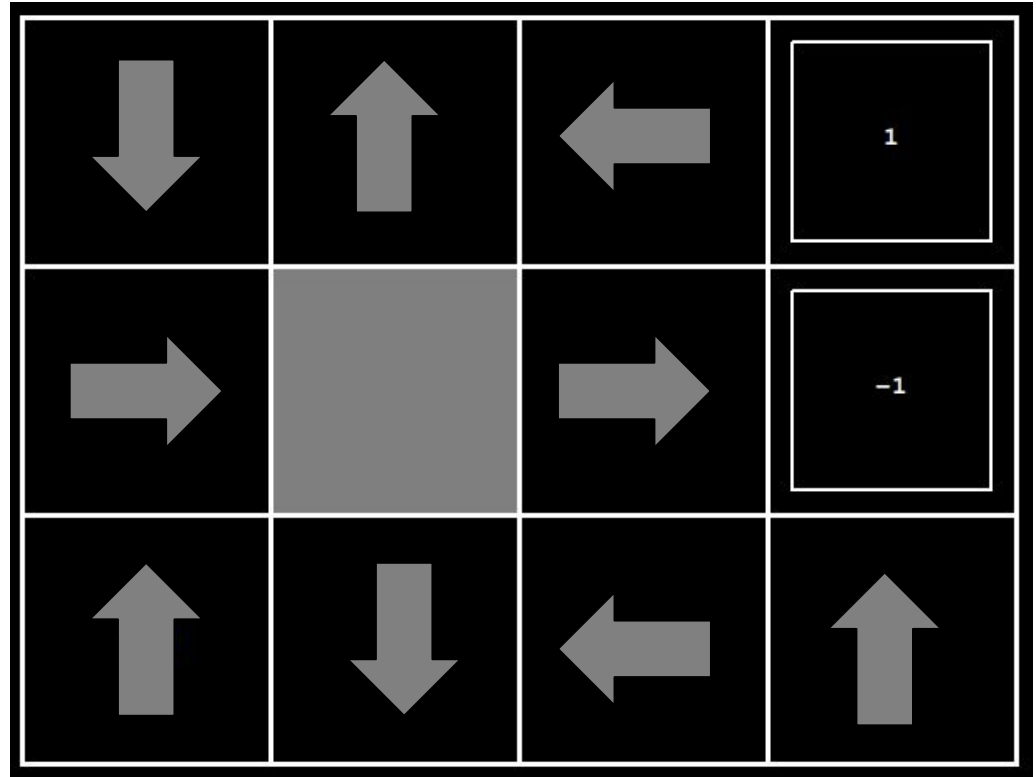
États:

- Position actuelle

Transitions:

- 0.5 selon l'action
- 0.5 aléatoire

Politique: aléatoire



Exemple: Gridworld

Actions:

- Haut, bas, gauche, droite

Récompense:

- +1, -1, 0 ailleurs

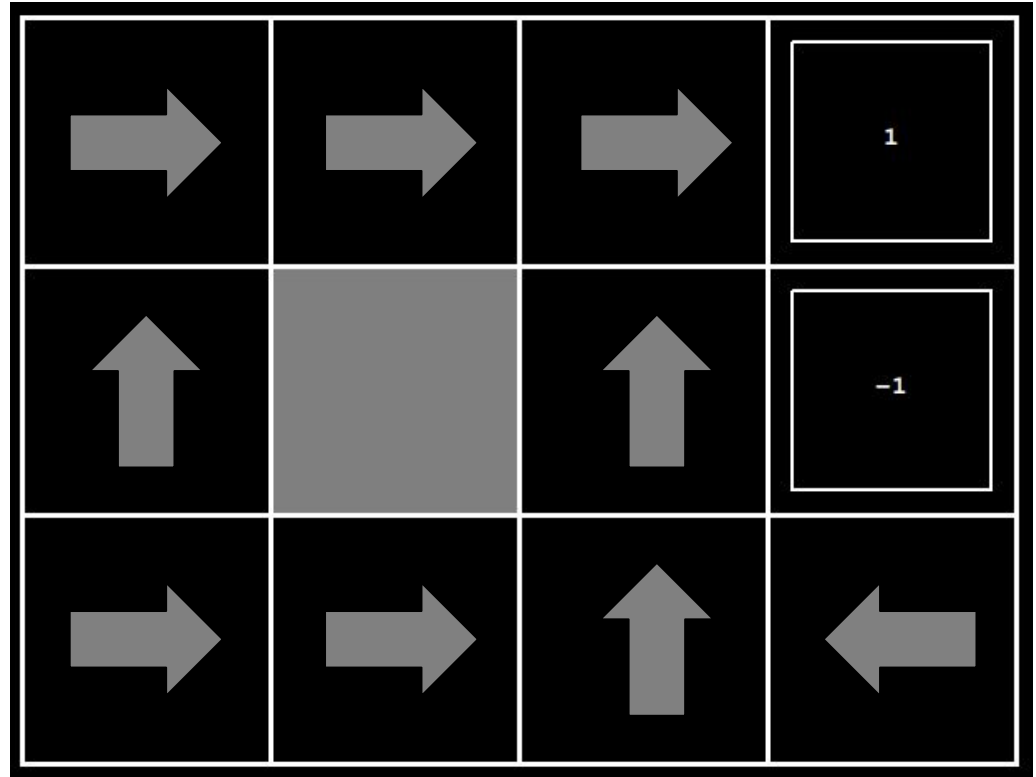
États:

- Position actuelle

Transitions:

- 0.5 selon l'action
- 0.5 aléatoire

Politique: un peu meilleure



Exemple: Gridworld

Actions:

- Haut, bas, gauche, droite

Récompense:

- +1, -1, 0 ailleurs

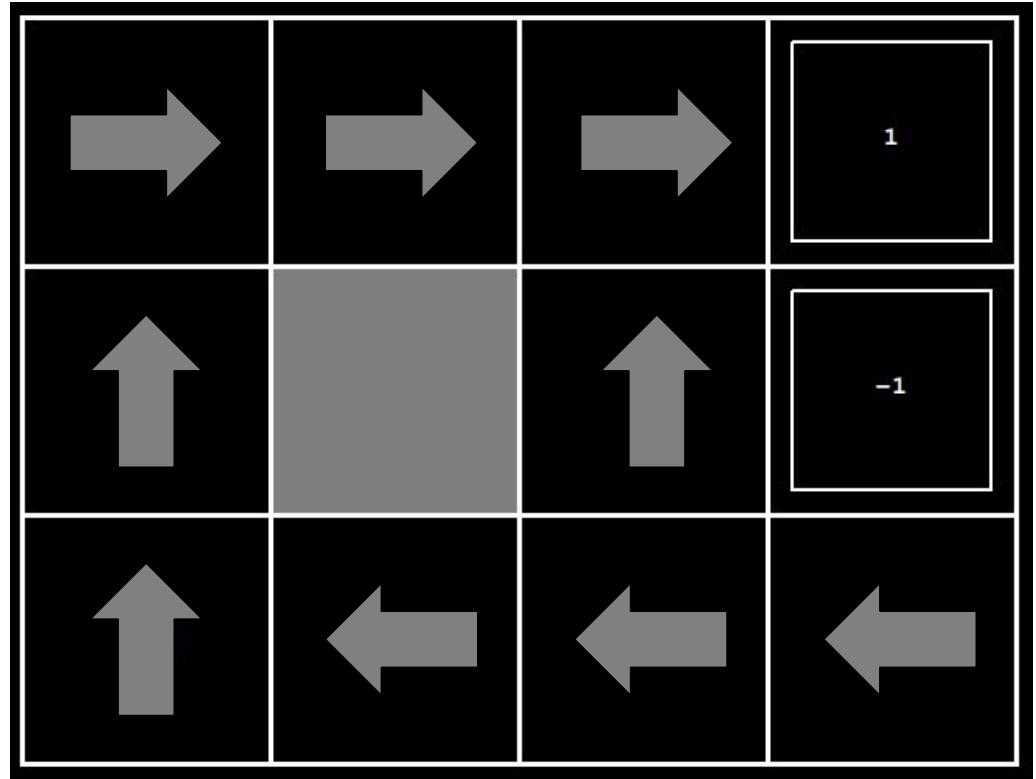
États:

- Position actuelle

Transitions:

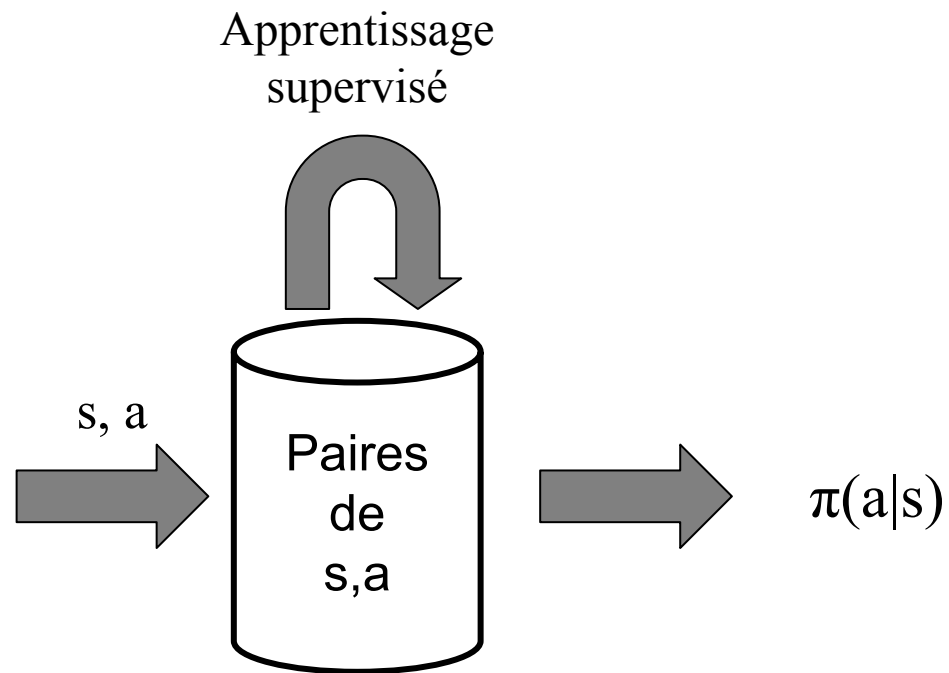
- 0.5 selon l'action
- 0.5 aléatoire

Politique: optimale



Comment trouver la politique optimale ?

Solution (trop) simple: utiliser l'apprentissage supervisé



Comment trouver la politique optimale ?

Solution (trop) simple: utiliser l'apprentissage supervisé

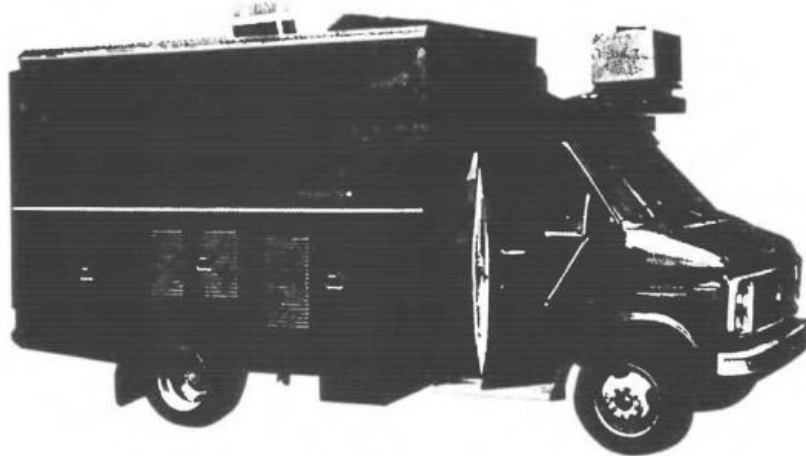


Figure 3: NAVLAB, the CMU autonomous navigation test vehicle.

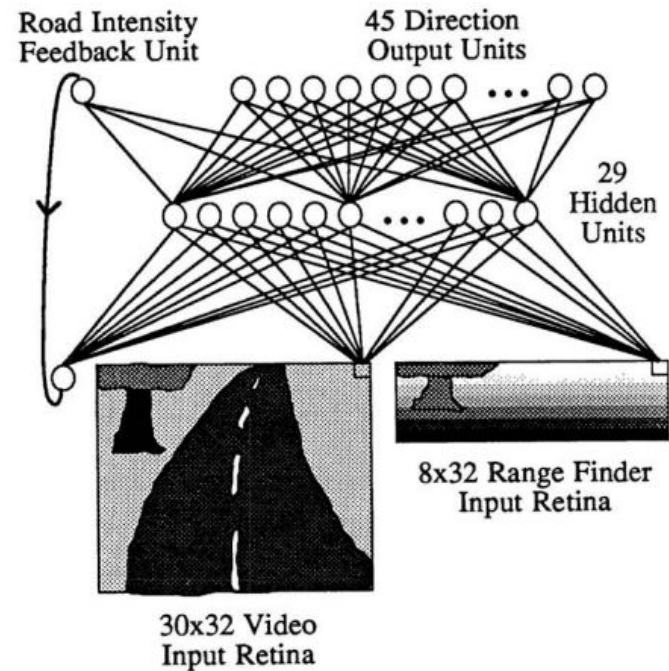


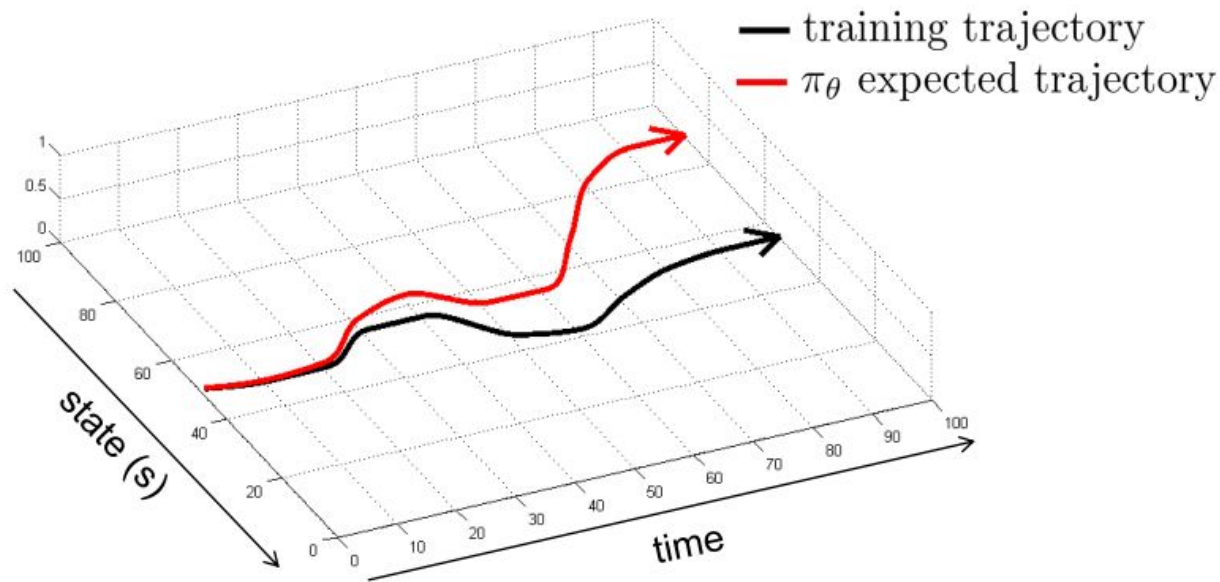
Figure 1: ALVINN Architecture

<https://www.youtube.com/watch?v=ntlczNQKfjQ>

Pomerleau, D. A. (1988). Alvin: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1.

Comment trouver la politique optimale ?

Solution (trop) simple: utiliser l'apprentissage supervisé
... ne fonctionne pas

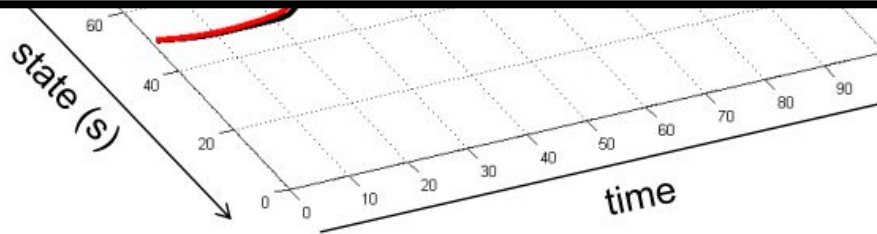


source: Sergey Levine, CS285

Comment trouver la politique optimale ?

Solution (trop) simple: utiliser l'apprentissage supervisé
... ne fonctionne pas

There are difficulties involved with training “on-the-fly” with real images. If the network is not presented with sufficient variability in its training exemplars to cover the conditions it is likely to encounter when it takes over driving from the human operator, it will not develop a sufficiently robust representation and will perform poorly. In addition, the network must not solely be shown examples of accurate driving, but also how to recover (i.e. return to the road center) once a mistake has been made. Partial initial training on a variety of simulated road images should help eliminate these difficulties and facilitate better performance.

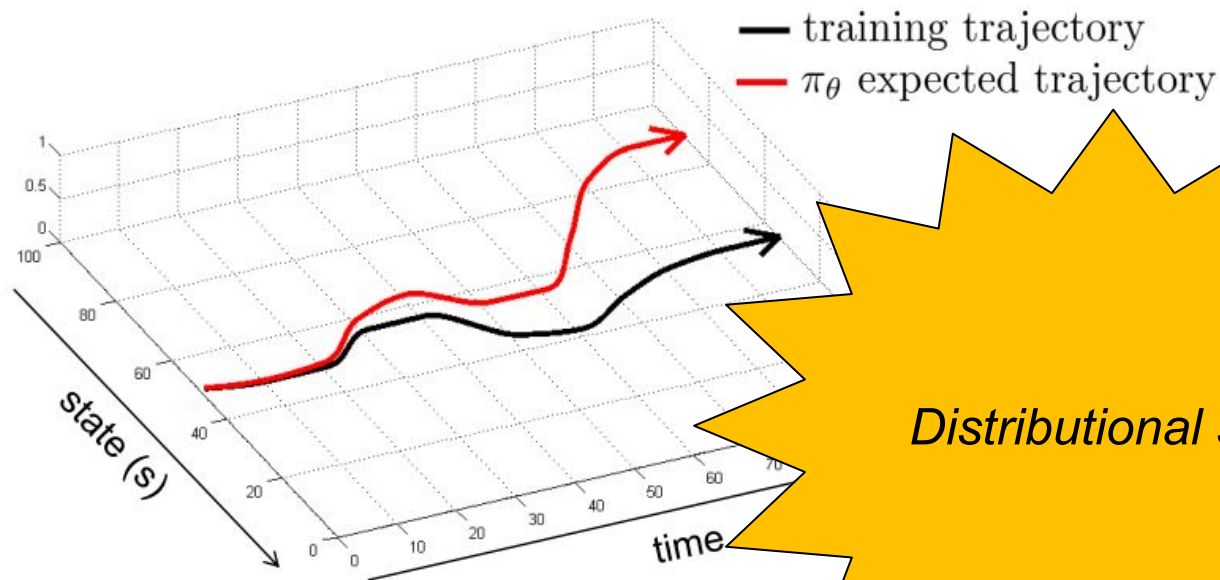


Pomerleau, D. A. (1988). Alvin: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1.

source: Sergey Levine, CS285

Comment trouver la politique optimale ?

Solution (trop) simple: utiliser l'apprentissage supervisé
... ne fonctionne pas



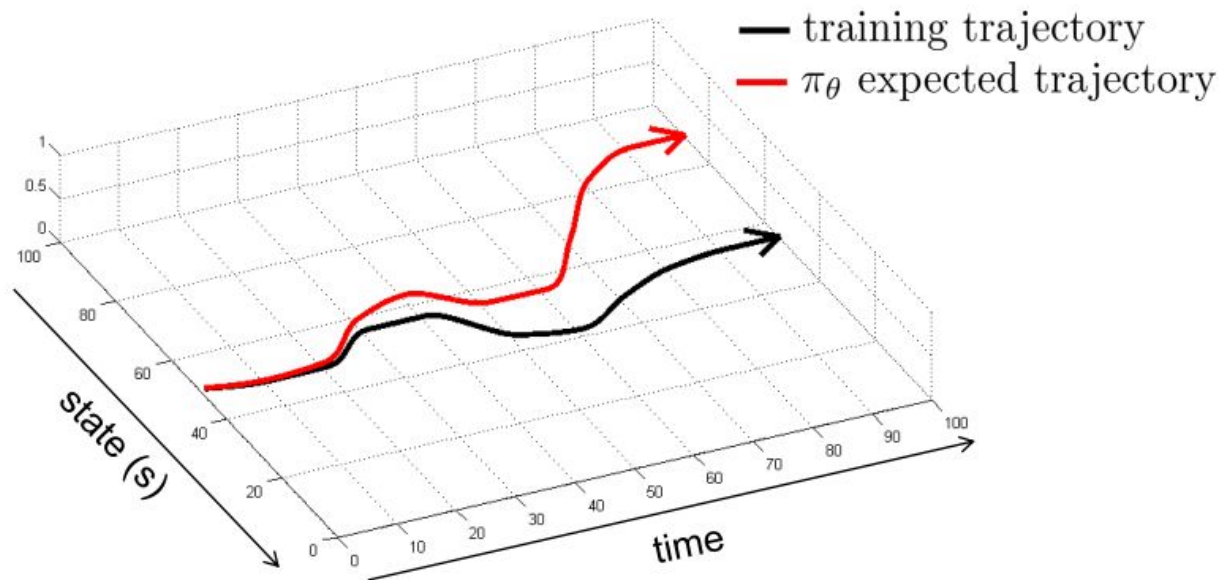
Distributional shift

Comment trouver la politique optimale ?

Solution (trop) simple: utiliser l'apprentissage supervisé

... ne fonctionne pas. Est-ce qu'on peut le faire fonctionner ?

Il faut que $p_{\text{données}}(s) = p_{\pi}(s)$



DAGGER: Dataset Aggregation

```
Initialize  $\mathcal{D} \leftarrow \emptyset$ .  
Initialize  $\hat{\pi}_1$  to any policy in  $\Pi$ .  
for  $i = 1$  to  $N$  do  
  Let  $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$ .  
  Sample  $T$ -step trajectories using  $\pi_i$ .  
  Get dataset  $\mathcal{D}_i = \{(s, \pi^*(s))\}$  of visited states by  $\pi_i$   
  and actions given by expert.  
  Aggregate datasets:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ .  
  Train classifier  $\hat{\pi}_{i+1}$  on  $\mathcal{D}$ .  
end for  
Return best  $\hat{\pi}_i$  on validation.
```

Algorithm 3.1: DAGGER Algorithm.

DAGGER: Dataset Aggregation

Problème ?

```
Initialize  $\mathcal{D} \leftarrow \emptyset$ .
Initialize  $\hat{\pi}_1$  to any policy in  $\Pi$ .
for  $i = 1$  to  $N$  do
  Let  $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$ .
  Sample  $T$ -step trajectories using  $\pi_i$ .
  Get dataset  $\mathcal{D}_i = \{(s, \pi^*(s))\}$  of visited states by  $\pi_i$ 
  and actions given by expert.
  Aggregate datasets:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ .
  Train classifier  $\hat{\pi}_{i+1}$  on  $\mathcal{D}$ .
end for
Return best  $\hat{\pi}_i$  on validation.
```

Algorithm 3.1: DAGGER Algorithm.

Ross, S., Gordon, G., & Bagnell, D. (2011, June). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 627-635). JMLR Workshop and Conference Proceedings.

DAGGER: Dataset Aggregation

Problème ?

Pas toujours possible ou facile

Initialize $\mathcal{D} \leftarrow \emptyset$.

Initialize $\hat{\pi}_1$ to any policy in Π .

for $i = 1$ **to** N **do**

Let $\pi_i = \beta \pi^* + (1 - \beta_i) \hat{\pi}_i$.

Sample T -step trajectories using π_i .

Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.

Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$.

Train classifier $\hat{\pi}_{i+1}$ on \mathcal{D} .

end for

Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

Comment trouver la politique optimale ?

À partir d'une politique π qui produit des trajectoires $s_1, a_1, r_1, s_2, a_2, r_2, \dots$

On aimerait pouvoir évaluer la “valeur” d'un état s arbitraire:

Comment trouver la politique optimale ?

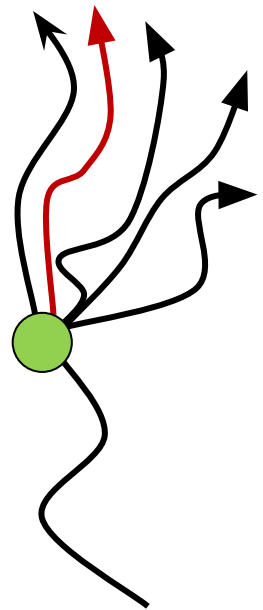
À partir d'une politique π qui produit des trajectoires $s_1, a_1, r_1, s_2, a_2, r_2, \dots$

On aimerait pouvoir évaluer la “valeur” d’un état s arbitraire:

Value function

$$\text{Value function} \\ V^\pi(s_t) = \mathbb{E}_{s,a \sim \pi} \left[\sum_{i=0}^{T-t} \gamma^i r_t \right]$$

“Quelle est le retour espéré si on suit la politique jusqu’à la fin ?”



Comment trouver la politique optimale ?

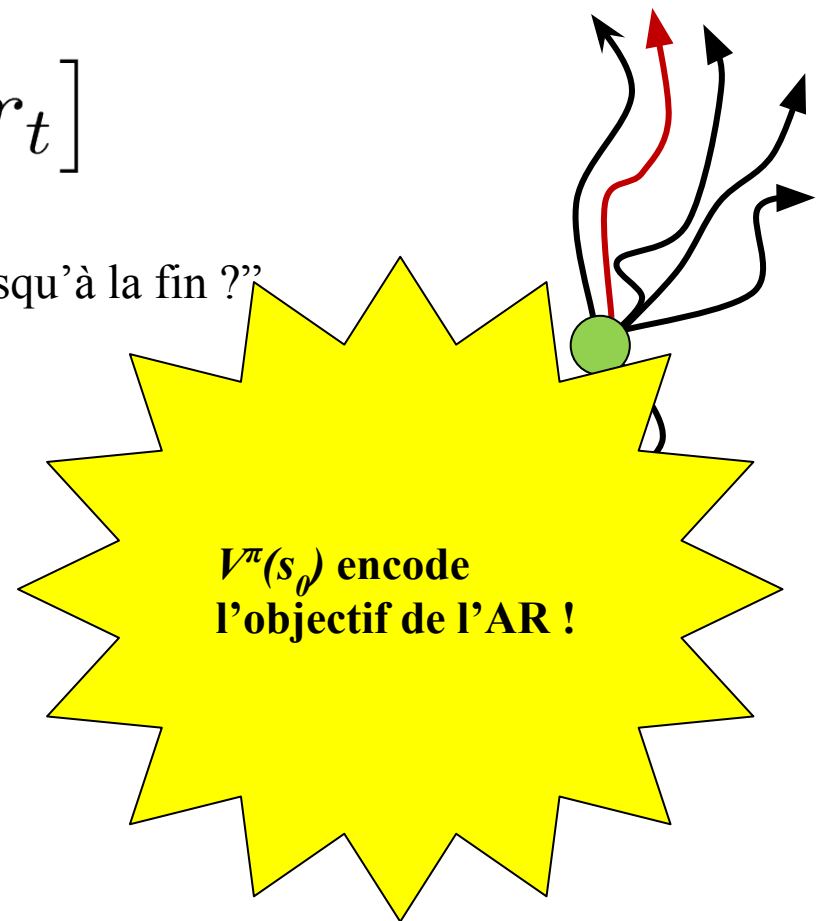
À partir d'une politique π qui produit des trajectoires $s_1, a_1, r_1, s_2, a_2, r_2, \dots$

On aimerait pouvoir évaluer la “valeur” d'un état s arbitraire:

Value function

$$V^\pi(s_t) = \mathbb{E}_{s, a \sim \pi} \left[\sum_{i=0}^{T-t} \gamma^i r_t \right]$$

“Quelle est le retour espéré si on suit la politique jusqu'à la fin ?”



Comment trouver la politique optimale ?

À partir d'une politique π qui produit des trajectoires $s_1, a_1, r_1, s_2, a_2, r_2, \dots$

On aimerait pouvoir évaluer la “valeur” d'un état s arbitraire:

Value function

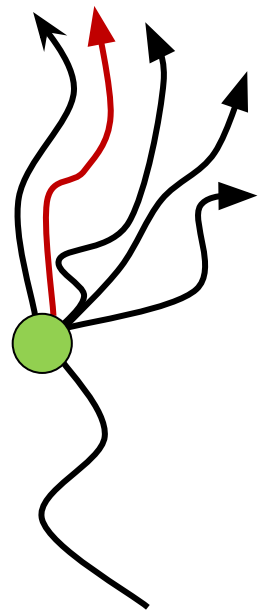
$$V^\pi(s_t) = \mathbb{E}_{s, a \sim \pi} \left[\sum_{i=0}^{T-t} \gamma^i r_{t+i} \right]$$

“Quelle est le retour espéré si on suit la politique jusqu'à la fin ?”

Q-function

$$Q^\pi(s_t, a_t) = r_t(s_t, a_t) + \mathbb{E}_{s, a \sim \pi} \left[\sum_{i=1}^{T-t+1} \gamma^i r_{t+i} \right]$$

“Quelle est le retour espéré si on effectue l'action a à l'état s puis on suit la politique jusqu'à la fin ?”



Comment trouver la politique optimale ?

À partir d'une politique π qui produit des trajectoires $s_1, a_1, r_1, s_2, a_2, r_2, \dots$

On aimerait pouvoir évaluer la “valeur” d'un état s arbitraire:

Value function

$$V^{\pi}(s_t) = \mathbb{E}_{s, a \sim \pi} \left[\sum_{i=0}^{T-t} \gamma^i r_{t+i} \right]$$

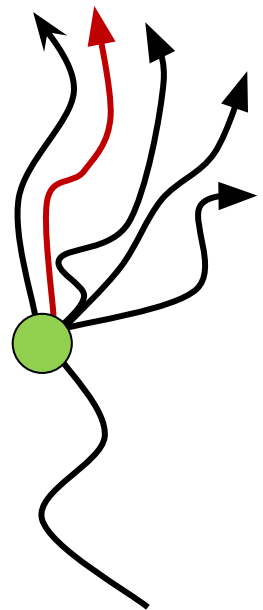
“Quelle est le retour espéré si on suit la politique jusqu'à la fin ?”

Q-function

$$Q^{\pi}(s_t, a_t) = r_t(s_t, a_t) + \mathbb{E}_{s, a \sim \pi} \left[\sum_{i=1}^{T-t+1} \gamma^i r_{t+i} \right]$$

“Quelle est le retour espéré si on effectue l'action a à l'état s puis on suit la politique jusqu'à la fin ?”

$$Q^{\pi}(s_t, a_t) = r_t(s_t, a_t) + V^{\pi}(s_{t+1})$$



Comment trouver la politique optimale ?

Q^* est la *Q-function* optimale pour la politique optimale π^*

$$Q^*(s, a) = \max_{\pi} \mathbb{E}_{s \sim \pi} \left[\sum_t^T \gamma^t r_t \right]$$

Maximise le retour espéré après avoir fait l'action a à l'état s en suivant la politique optimale. **Encode la politique optimale !**

Équation de Bellman:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \pi} \left[r + \gamma \max_{a'} Q^*(s', a') \right]$$

Comment trouver la politique optimale ?

Idée 1: Si on trouve une Q-function qui respecte l'équation de Bellman, on a une Q-function optimale

$$Q^*(s, a) = \mathbb{E}_{s' \sim \pi} [r + \gamma \max_{a'} Q^*(s', a')]$$

Comment trouver la politique optimale ?

Idée 1: Si on trouve une Q-function qui respecte l'équation de Bellman, on a une Q-function optimale

$$Q^*(s, a) = \mathbb{E}_{s' \sim \pi} [r + \gamma \max_{a'} Q^*(s', a')]$$

Idée 2: On peut toujours améliorer une politique en maximisant sa Q-function

$$\pi' := \arg \max_a Q^\pi(s, a) \quad \forall s, a \implies \pi' \geq \pi$$

Comment trouver la politique optimale ?

Idée 1: Si on trouve une Q-function qui respecte l'équation de Bellman, on a une Q-function optimale

$$Q^*(s, a) = \mathbb{E}_{s' \sim \pi} [r + \gamma \max_{a'} Q^*(s', a')]$$

Idée 2: On peut toujours améliorer une politique en maximisant sa Q-function

$$\pi' := \arg \max_a Q^\pi(s, a) \quad \forall s, a \implies \pi' \geq \pi$$

Idée 3: On peut représenter la Q-function par la Value-function

$$V(s) = \mathbb{E}_{s, a \sim \pi} Q(s, a)$$

Policy/Value-iteration

Solution: Utiliser l'équation de Bellman pour apprendre Q

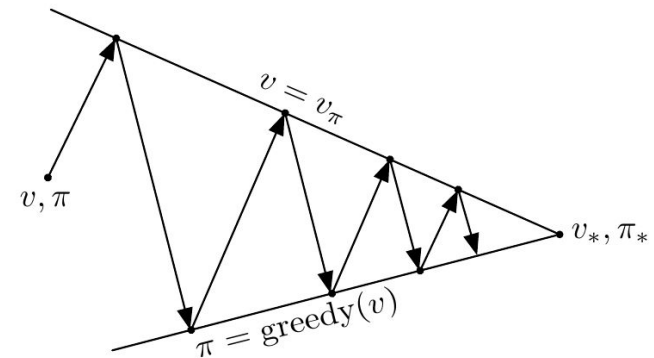
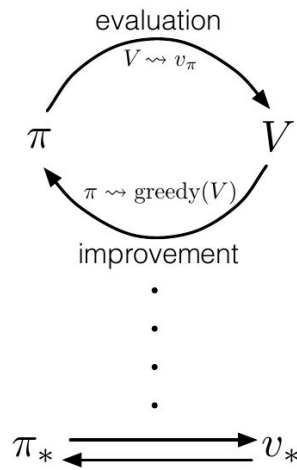
1. Initialize Q randomly.
2. Initialize $\pi(s)$ to $\arg \max_a Q(s, a) \forall s, a$.
3. Repeat until π converges:
 - (a) For each $s \in S$:
 - i. For each $a \in A$:
 - A. $Q(s, a) := r(s, a) + V(s')$
 - ii. $\pi(s) := \arg \max_a Q(s, a)$.

Q, π convergent vers Q^*, π^* !

Policy/Value-iteration

Solution: Utiliser l'équation de Bellman pour apprendre Q

1. Initialize Q randomly.
2. Initialize $\pi(s)$ to $\arg \max_a Q(s, a) \forall s, a$.
3. Repeat until π converges:
 - (a) For each $s \in S$:
 - i. For each $a \in A$:
 - A. $Q(s, a) := r(s, a) + V(s')$
 - ii. $\pi(s) := \arg \max_a Q(s, a)$.

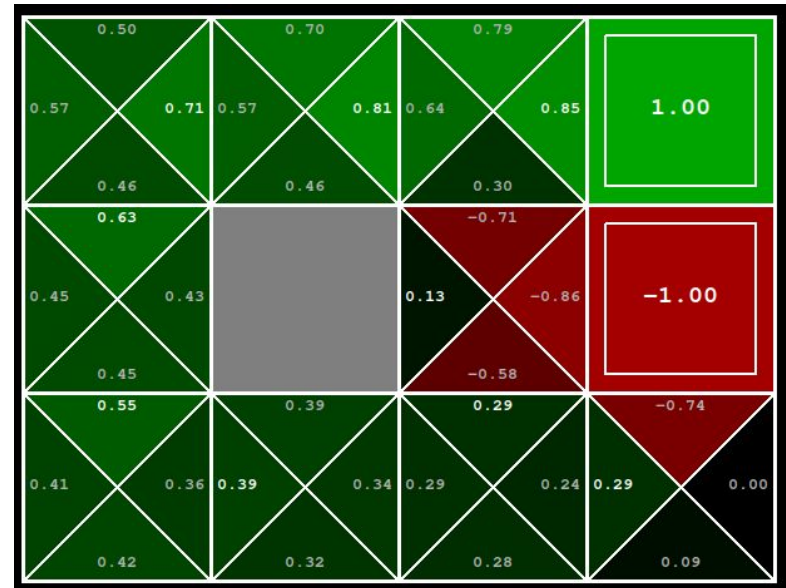
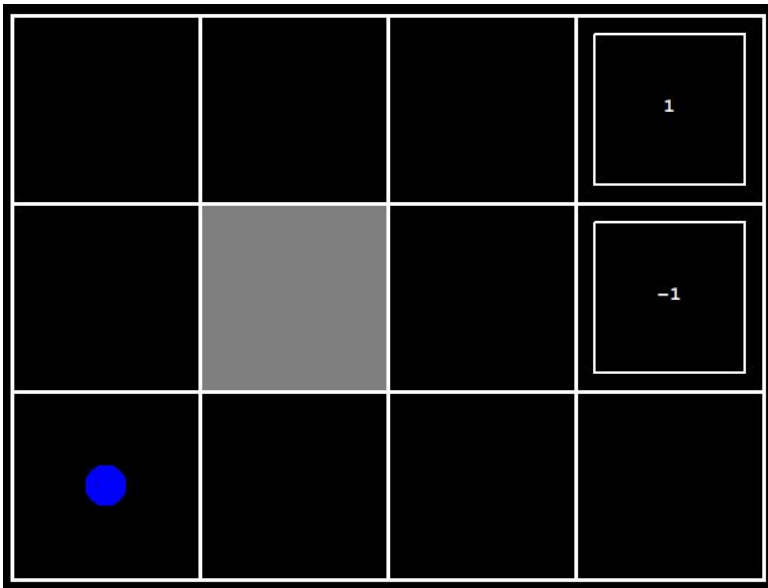


(version simplifiée de Policy/Value iteration)

Q, π convergent vers Q^*, π^* !

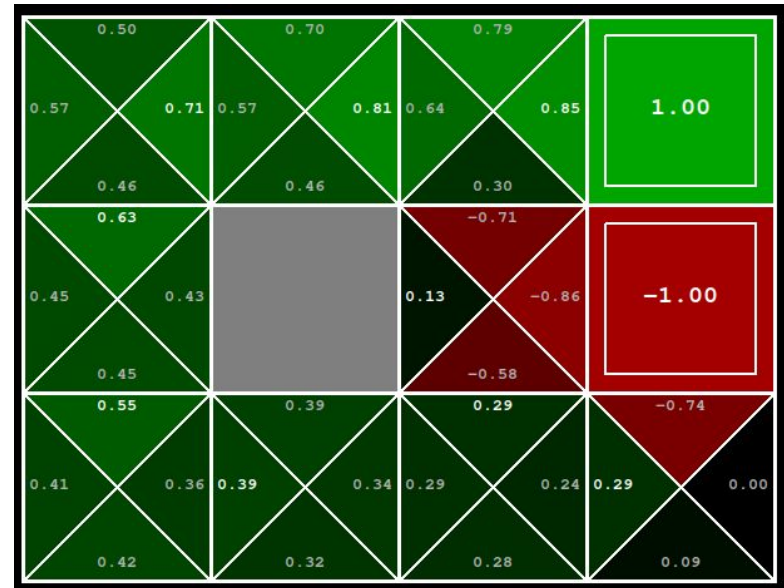
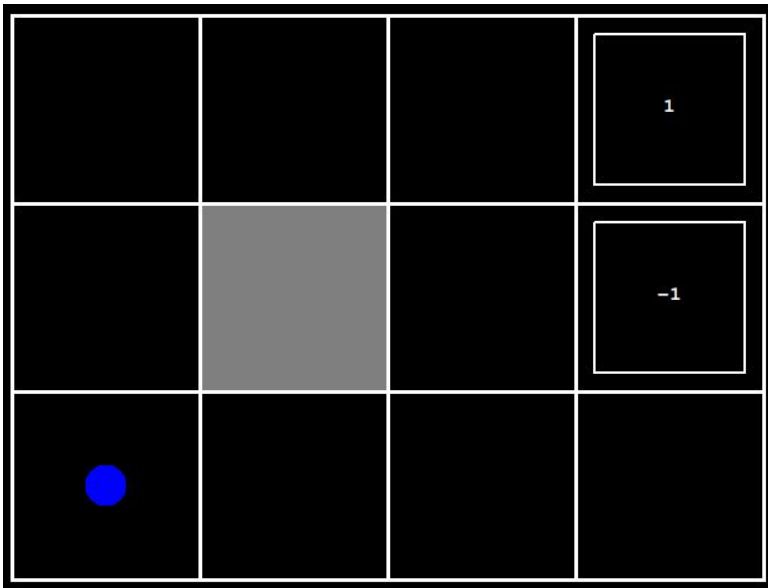
Policy/Value-iteration

Q, π convergent vers Q^*, π^* !



Policy/Value-iteration

Q, π convergent vers Q^*, π^* !



Problème: On doit itérer sur et garder en mémoire tous les s, a

Temporal-difference/Q-learning

Équation de Bellman:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \pi} \left[r + \gamma \max_{a'} Q^*(s', a') \right]$$

Nous allons définir l'équation de Bellman comme cible à apprendre !

$$y = r(s, a) + \gamma \max_{a'} Q(s', a')$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[\underbrace{r_t + \gamma Q(s_{t+1}, a_{t+1})}_{\text{cible}} - \underbrace{Q(s_t, a_t)}_{\text{prédiction}} \right]$$

Temporal-difference/Q-learning


Équation de Bellman:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \pi} \left[r + \gamma \max_{a'} Q^*(s', a') \right]$$

Nous allons définir l'équation de Bellman comme cible à apprendre !

$$y = r(s, a) + \gamma \max_{a'} Q(s', a')$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[\underbrace{r_t + \gamma Q(s_{t+1}, a_{t+1})}_{\text{cible}} \right]$$



Transforme le
problème en
régression !

Temporal-difference/Q-learning


Équation de Bellman:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \pi} \left[r + \gamma \max_{a'} Q^*(s', a') \right]$$

Nous allons définir l'équation de Bellman comme cible à apprendre !

$$y = r(s, a) + \gamma \max_{a'} Q(s', a')$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[\underbrace{r_t + \gamma Q(s_{t+1}, a_{t+1})}_{\text{cible}} \right]$$



Apprentissage
“semi/auto”-
supervisé

Temporal-difference/Q-learning

Nous allons définir l'équation de Bellman comme cible à apprendre !

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \underbrace{[r_t + \gamma Q(s_{t+1}, a_{t+1})]}_{\text{cible}} - \underbrace{Q(s_t, a_t)}_{\text{prédiction}}$$

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory \mathcal{D} to capacity N

Initialize action-value function Q with random weights

for episode = 1, M **do**

 Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$

for $t = 1, T$ **do**

 With probability ϵ select a random action a_t

 otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$

 Execute action a_t in emulator and observe reward r_t and image x_{t+1}

 Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

 Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in \mathcal{D}

 Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from \mathcal{D}

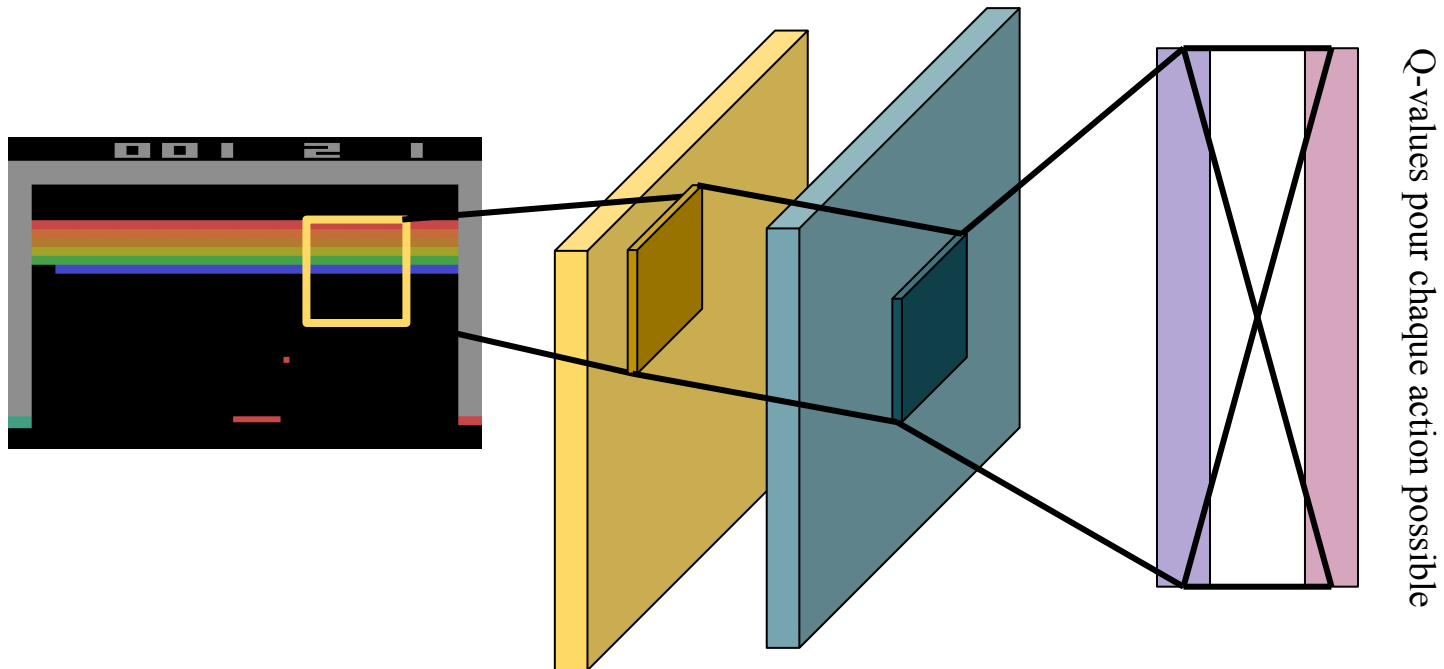
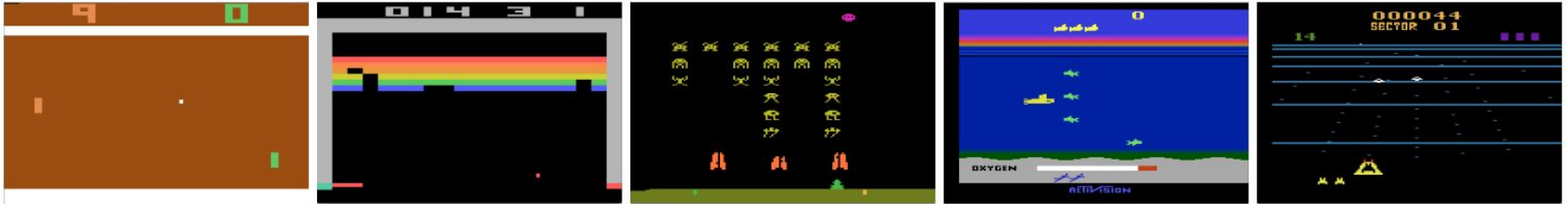
 Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$

 Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3

end for

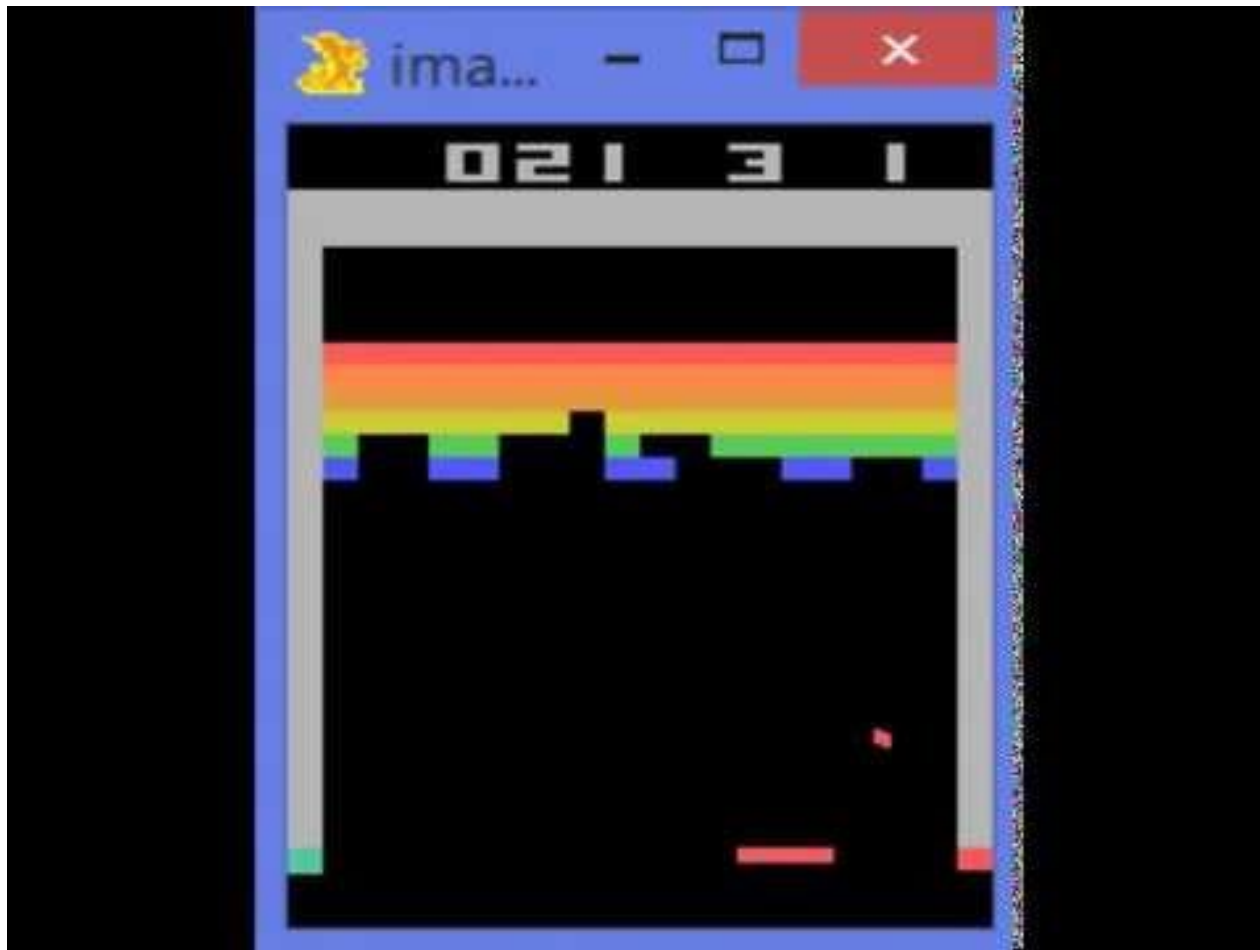
end for

Atari



Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Atari



<https://www.youtube.com/watch?v=V1eYniJ0Rnk>

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Deep Q-Learning

DQN est devenu une famille d'algorithmes en soi !

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 30, No. 1).

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* (pp. 1995-2003). PMLR.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., ... & Legg, S. (2017). Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., ... & Silver, D. (2018, April). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., & Blundell, C. (2020, November). Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning* (pp. 507-517). PMLR.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Deep Q-Learning

DQN est devenu une famille

Mnih, V., Kavukcuoglu, K., Silver, D., Gra...
reinforcement learning. *arXiv preprint arXiv:1312.5680*

Van Hasselt, H., Guez, A., & Silver, D. (2015).
conference on artificial intelligence (Vol. 30). AAAI Press.

Wang, Z., Schaul, T., Hessel, M., Hasselt, P., Lanctot, M., & Sutton, R. S. (2016).
reinforcement learning. In *International conference on machine learning* (pp. 1993-2000). PMLR.

Fortunato, M., Azar, M. G., Piot, B., Menikoff, J., & Sutton, R. S. (2017).
arXiv:1706.10295.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., & Sutton, R. S. (2017).
improvements in deep reinforcement learning with distributional temporal difference learning.

Badia, A. P., Piot, B., Kapturowski, S., Sprechendorfer, G., & Sutton, R. S. (2017).
Outperforming the atari human benchmark with a simple distributional td(0) algorithm.

Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Playing atari with deep reinforcement learning.

Games	Average Human	Random	Agent57
alien	7127.70	227.80	297638.17 ± 37054.55
amidar	1719.50	5.80	29660.08 ± 880.39
assault	742.00	222.40	67212.67 ± 6150.59
asterix	8503.30	210.00	991384.42 ± 9493.32
asteroids	47388.70	719.10	150854.61 ± 16116.72
atlantis	29028.10	12850.00	1528841.76 ± 28282.53
bank heist	753.10	14.20	23071.50 ± 15834.73
battle zone	37187.50	2360.00	934134.88 ± 38916.03
beam rider	16926.50	363.90	300509.80 ± 13075.35
berzerk	2630.40	123.70	61507.83 ± 26539.54
bowling	160.70	23.10	251.18 ± 13.22
boxing	12.10	0.10	100.00 ± 0.00
breakout	30.50	1.70	790.40 ± 60.05
centipede	12017.00	2090.90	412847.86 ± 26087.14
chopper command	7387.80	811.00	999900.00 ± 0.00
crazy climber	35829.40	10780.50	565909.85 ± 89183.85
defender	18688.90	2874.50	677642.78 ± 16858.59
demon attack	1971.00	152.10	143161.44 ± 220.32
double dunk	-16.40	-18.60	23.93 ± 0.06
enduro	860.50	0.00	2367.71 ± 8.69
fishing derby	-38.70	-91.70	86.97 ± 3.25
freeway	29.60	0.00	32.59 ± 0.71
frostbite	4334.70	65.20	541280.88 ± 17485.76
gopher	2412.50	257.60	117777.08 ± 3108.06
gravitar	3351.40	173.00	19213.96 ± 348.25
hero	30826.40	1027.00	114736.26 ± 49116.60
ice hockey	0.90	-11.20	63.64 ± 6.48
jamesbond	302.80	29.00	135784.96 ± 9132.28
kangaroo	3035.00	52.00	24034.16 ± 12565.88
krull	2665.50	1598.00	251997.31 ± 20274.39
kung fu master	22736.30	258.50	206845.82 ± 11112.10
montezuma revenge	4753.30	0.00	9352.01 ± 2939.78
ms pacman	6951.60	307.30	63994.44 ± 6652.16
name this game	8049.00	2292.30	54386.77 ± 6148.50
phoenix	7242.60	761.40	908264.15 ± 28978.92
pitfall	6463.70	-229.40	18756.01 ± 9783.91
pong	14.60	-20.70	20.67 ± 0.47
private eye	69571.30	24.90	79716.46 ± 29515.48
qbert	13455.00	163.90	580328.14 ± 151251.66
riverraid	17118.00	1338.50	63318.67 ± 5659.55
road runner	7845.00	11.50	243025.80 ± 79555.98
robotank	11.90	2.20	127.32 ± 12.50
seaquest	42054.70	68.40	999997.63 ± 1.42
skiing	-4336.90	-17098.10	-4202.60 ± 607.85
solaris	12326.70	1236.30	44199.93 ± 8055.50
space invaders	1668.70	148.00	48680.86 ± 5894.01
star gunner	10250.00	664.00	839573.53 ± 67132.17
surround	6.50	-10.00	9.50 ± 0.19
tennis	-8.30	-23.80	23.84 ± 0.10
time pilot	5229.20	3568.00	405425.31 ± 17044.45
tutankham	167.60	11.40	2354.91 ± 3421.43
up n down	11693.20	533.40	623805.73 ± 23493.75
venture	1187.50	0.00	2623.71 ± 442.13
video pinball	17667.90	0.00	992340.74 ± 12867.87
wizard of wor	4756.50	563.50	157306.41 ± 16000.00
yars revenge	54576.90	3092.90	998532.37 ± 375.82
zaxxon	9173.30	32.50	249808.90 ± 58261.59

Playing atari with deep

ing. In *Proceedings of the AAAI*

k architectures for deep

etworks for exploration. *arXiv preprint*

April). Rainbow: Combining
nce.

, November). Agent57:
7-517). PMLR.

ra, D., & Riedmiller, M. (2013). Playing atari with

Deep Q-Learning

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory \mathcal{D} to capacity N

Initialize action-value function Q with random weights

for episode = 1, M **do**

 Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$

for $t = 1, T$ **do**

 With probability ϵ select a random action a_t

 otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$

 Execute action a_t in emulator and observe reward r_t and image x_{t+1}

 Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

 Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in \mathcal{D}

 Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from \mathcal{D}

 Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$

 Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3

end for

end for

Problème ?

Deep Q-Learning

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory \mathcal{D} to capacity N

Initialize action-value function Q with random weights

for episode = 1, M **do**

 Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$

for $t = 1, T$ **do**

 With probability ϵ select a random action a_t
 otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$

 Execute action a_t in emulator and observe reward r_t and image x_{t+1}

 Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

 Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in \mathcal{D}

 Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from \mathcal{D}

 Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$

 Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3

end for

end for

Problème ?

Demande d'évaluer toutes les actions.

Deep Q-Learning

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory \mathcal{D} to capacity N

Initialize action-value function Q with random weights

for episode = 1, M **do**

 Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$

for $t = 1, T$ **do**

 With probability ϵ select a random action a_t
 otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$

 Execute action a_t in emulator and observe reward r_t and image x_{t+1}

 Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

 Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in \mathcal{D}

 Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from \mathcal{D}

 Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$

 Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3

end for

end for

Problème ?

Demande d'évaluer toutes les actions.

Qu'arrive-t'il si le domaine d'action est continu ?