

Apprentissage par renforcement

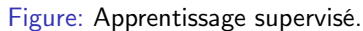
Arthur Aubret

Université Clermont Auvergne, Institut Pascal
ajp.aubret@gmail.com

10/02/2022

Sommaire

- 1 Apprentissage par renforcement
 - Introduction
 - Modélisation
 - Programmation dynamique
 - Apprentissage par renforcement
- 2 Apprentissage profond par renforcement
 - DQN
 - Policy-based methods
 - Actor-Critic
- 3 Motivation intrinsèque
 - Problèmes du RL



- Données + labels en entrée.

Apprentissage non supervisé

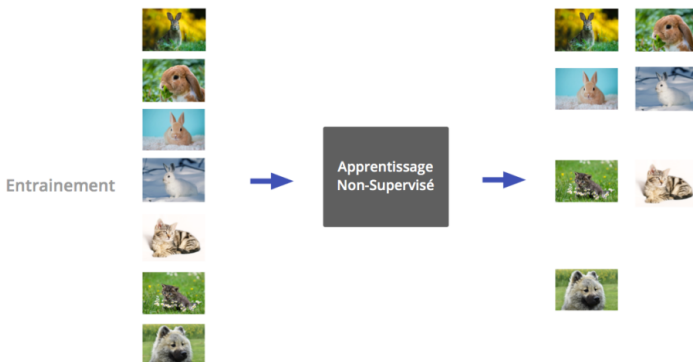
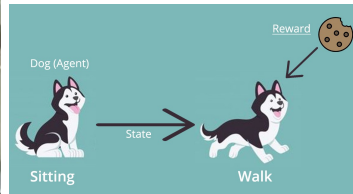


Figure: Apprentissage non supervisé.

- Données en entrée.

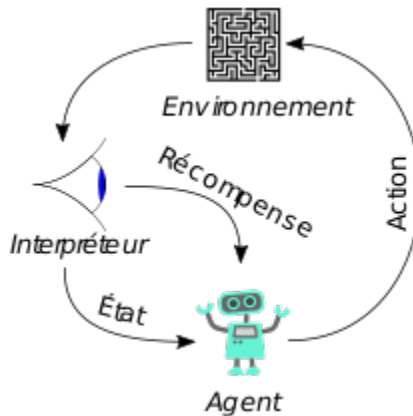
Apprentissage par renforcement



→ Conditionnement opérant de Pavlov.

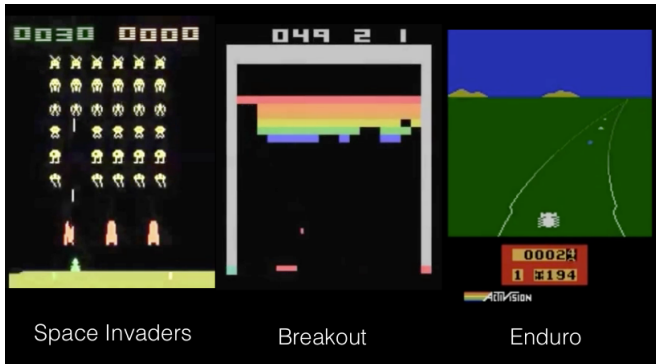
- Apprentissage par essai-erreur;
- Récompenser les *bonnes* actions: punition ou friandise.

Apprentissage par renforcement (2)



- Faire des actions qui donnent des récompenses.

Succès du RL: Atari



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

Succès du RL: Go



- Quelques nouveautés comme l'auto-affrontement.

Possibles applications



- Finances, médecine, smart grids...
- Robotique industrielle, voiture autonome, gestion du trafic...

Processus de décision markovien (MDP)

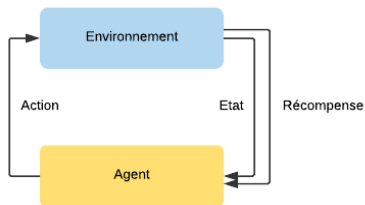


Figure: MDP et exemple d'environnement

Processus de décision markovien (MDP)

- S ensemble d'états (cases du robot);

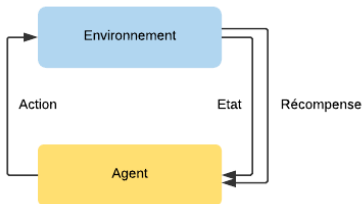


Figure: MDP et exemple d'environnement

Processus de décision markovien (MDP)

- S ensemble d'états (cases du robot);
- A ensemble d'actions (droite, gauche, haut, bas);

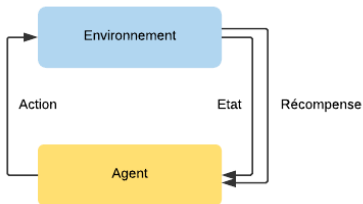


Figure: MDP et exemple d'environnement

Processus de décision markovien (MDP)

- S ensemble d'états (cases du robot);
- A ensemble d'actions (droite, gauche, haut, bas);
- $R : S \times A \times S \rightarrow \mathbb{R}$ fonction de récompense;

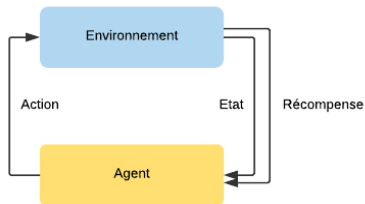


Figure: MDP et exemple d'environnement

Processus de décision markovien (MDP)

- S ensemble d'états (cases du robot);
- A ensemble d'actions (droite, gauche, haut, bas);
- $R : S \times A \times S \rightarrow \mathbb{R}$ fonction de récompense;
- $T : S \times A \times S \rightarrow \mathbb{R}$ fonction de transition, $T(s, a, s') = p(s'|a, s)$;
- ρ_0 distribution initiale d'états.

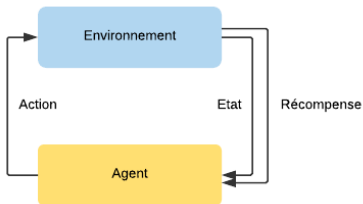


Figure: MDP et exemple d'environnement

Boucle d'interaction

- ① Obtient un état initial $s_0 \sim \rho_0(\cdot)$;
- ② L'agent choisit une action $a \sim \pi(\cdot|s)$, $\pi : S \times A \rightarrow \mathbb{R}$;
- ③ Recoit un nouvel état $s' \sim p(s'|s, a)$ et une récompense $r = R(s, a, s')$.
- ④ Rechoisit une action, etc...
- ⑤ Jusqu'à état absorbant.

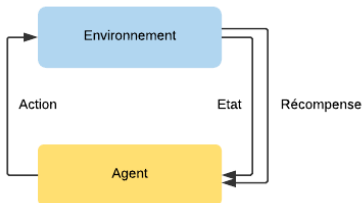


Figure: MDP et exemple d'environnement

Fonction d'évaluation

Objectif: trouver la politique optimale π^* maximisant:

Récompenses cumulées atténuées, $\gamma \in [0, 1]$ facteur d'atténuation:

(2)

Récompenses cumulées, tâche épisodique: $\mathbb{E}_{\pi} \left[\sum_{t=0}^T r_t \right]$.

Fonction d'évaluation

Objectif: trouver la politique optimale π^* maximisant:

Récompenses cumulées atténuées, $\gamma \in [0, 1]$ facteur d'atténuation:

$$\pi^* = \arg \max_{\pi} R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \dots \quad (1)$$

(2)

Récompenses cumulées, tâche épisodique: $\mathbb{E}_{\pi} \left[\sum_{t=0}^T r_t \right]$.

Fonction d'évaluation

Objectif: trouver la politique optimale π^* maximisant:

Récompenses cumulées atténuées, $\gamma \in [0, 1]$ facteur d'atténuation:

$$\pi^* = \arg \max_{\pi} R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \dots \quad (1)$$

$$= \arg \max_{\pi} \mathbb{E}_{\substack{a_t \sim \pi(\cdot | s_t) \\ s_{t+1} \sim p(\cdot | s_t, a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 \sim d_0(\cdot) \right]$$

(2)

Récompenses cumulées, tâche épisodique: $\mathbb{E}_{\pi} \left[\sum_{t=0}^T r_t \right]$.

Fonction d'évaluation

Objectif: trouver la politique optimale π^* maximisant:

Récompenses cumulées atténuées, $\gamma \in [0, 1]$ facteur d'atténuation:

$$\pi^* = \arg \max_{\pi} R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \dots \quad (1)$$

$$= \arg \max_{\pi} \mathbb{E}_{\substack{a_t \sim \pi(\cdot | s_t) \\ s_{t+1} \sim p(\cdot | s_t, a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 \sim d_0(\cdot) \right]$$

$$= \arg \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

(2)

Récompenses cumulées, tâche épisodique: $\mathbb{E}_{\pi} \left[\sum_{t=0}^T r_t \right]$.

Comment résoudre ?

- Essayer toutes les politiques π^* jusqu'à trouver la meilleure.

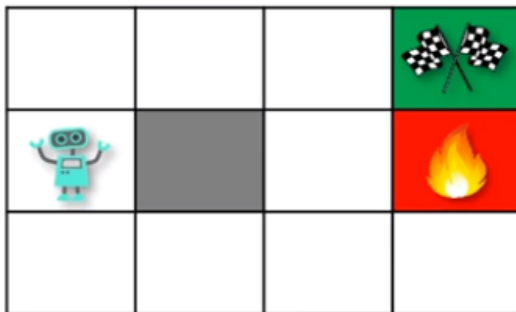


Figure: Environnement

Comment résoudre ?

- Essayer toutes les politiques π^* jusqu'à trouver la meilleure.
- Nombre de politiques déterministes possibles: $|A|^{|S|} = 262144$.

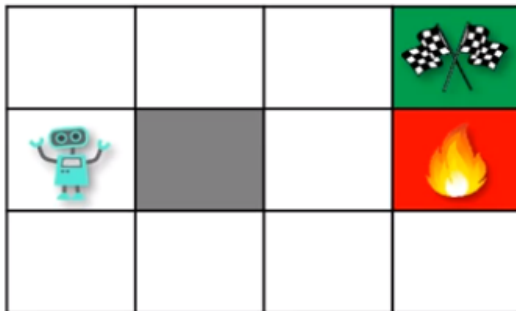


Figure: Environnement

Comment résoudre ?

- Essayer toutes les politiques π^* jusqu'à trouver la meilleure.
- Nombre de politiques déterministes possibles: $|A|^{|S|} = 262144$.
- Nombre de politiques stochastiques infini.

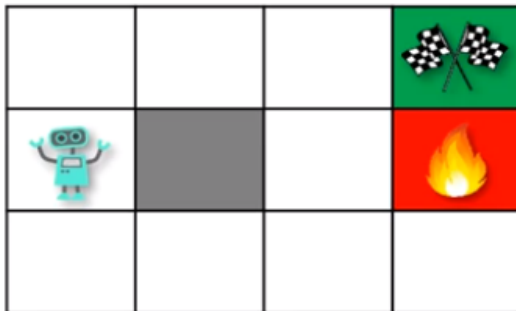



Figure: Environnement

Calculer les valeurs optimales d'états

Quel est le maximum de récompenses futures possible depuis un état ?

- $Q^*(s, a) = \mathbb{E}_{\pi^*} \sum_{t=0}^{\infty} [\gamma^t R(s_t, a_t, s_{t+1})]_{s_0=s, a_0=a};$
- $V^*(s) = \mathbb{E}_{\pi^*} \sum_{t=0}^{\infty} [\gamma^t R(s_t, a_t, s_{t+1})]_{s_0=s} = \arg \max_{a \in A} Q^*(s, a);$


0.81	0.9	1	+1
		0.9	-1
0.9	0.72	0.81	0.72

Calculer les valeurs optimales d'états

Quel est le maximum de récompenses futures possible depuis un état ?

- $Q^*(s, a) = \mathbb{E}_{\pi^*} \sum_{t=0}^{\infty} [\gamma^t R(s_t, a_t, s_{t+1})]_{s_0=s, a_0=a};$
- $V^*(s) = \mathbb{E}_{\pi^*} \sum_{t=0}^{\infty} [\gamma^t R(s_t, a_t, s_{t+1})]_{s_0=s} = \arg \max_{a \in A} Q^*(s, a);$
- Meilleure action en s :

$$\arg \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V^*(s')).$$


0.81	0.9	1	+1
		0.9	-1
0.9	0.72	0.81	0.72

Calculer les valeurs optimales d'états

Quel est le maximum de récompenses futures possible depuis un état ?

- $Q^*(s, a) = \mathbb{E}_{\pi^*} \sum_{t=0}^{\infty} [\gamma^t R(s_t, a_t, s_{t+1})]_{s_0=s, a_0=a}$;
- $V^*(s) = \mathbb{E}_{\pi^*} \sum_{t=0}^{\infty} [\gamma^t R(s_t, a_t, s_{t+1})]_{s_0=s} = \arg \max_{a \in A} Q^*(s, a)$;
- Meilleure action en s :

$$\arg \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V^*(s')).$$
- Définitions similaires de $V_{\pi}(s)$ et $Q_{\pi}(s, a)$.

0.81	0.9	1	+1
		0.9	-1
0.9	0.72	0.81	0.72

Equation d'optimalité de Bellman

On a $V^*(s) = \mathbb{E}_{\pi^*} \sum_{t=0}^{\infty} [\gamma^t R(s_t, a_t, s_{t+1})]_{s_0=s}$.


Equation d'optimalité de Bellman permet l'écriture sous forme récursive:

$$V^*(s) = \max_a \sum_{s' \in S} p(s'|s, a) (R(s, a, s') + \gamma V^*(s')) \quad (3)$$

Algorithme de Value Iteration

Trouver V^* , avec états absorbants, actions déterministes, $\gamma = 0.9$:



- 1 Initialise aléatoirement toutes les valeurs, et 0 aux états absorbants;



0.5	0	0.1	+1
		0.5	-1
1	0	0.8	1

Algorithme de Value Iteration

Trouver V^* , avec états absorbants, actions déterministes, $\gamma = 0.9$:

- ① Initialise aléatoirement toutes les valeurs, et 0 aux états absorbants;
- ② $\forall s \in S: V(s) \leftarrow \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'))$;
- ③ Recommencer jusqu'à convergence des valeurs.


↑ 0.5 ← 0	0.1 → +1
 	↓ 0.5 -1
← 1 ← 0	↓ 0.8 → 1 →


0.45	0.45	1	+1
 	0.72	-1	
0.9	0.9	0.9	0.9


Algorithme de Value Iteration

Trouver V^* , avec états absorbants, actions déterministes, $\gamma = 0.9$:

- 1 Initialise aléatoirement toutes les valeurs, et 0 aux états absorbants;
- 2 $\forall s \in S: V(s) \leftarrow \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'))$;
- 3 Recommencer jusqu'à convergence des valeurs.

0.5	← 0	0.1	→ +1
		0.5	-1
← 1	← 0	0.8	→ 1 →


0.45	0.45	1	+1
		0.72	-1
0.9	0.9	0.9	0.9


0.405	0.9	1	+1
		0.9	-1
0.81	0.81	0.81	0.81


Algorithme de Value Iteration


Trouver V^* , avec états absorbants, actions déterministes, $\gamma = 0.9$:

- 1 Initialise aléatoirement toutes les valeurs, et 0 aux états absorbants;
- 2 $\forall s \in S: V(s) \leftarrow \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'))$;
- 3 Recommencer jusqu'à convergence des valeurs.

0.5	0	0.1	+1
		0.5	-1
1	0	0.8	1

0.45	0.45	1	+1
		0.72	-1
0.9	0.9	0.9	0.9


0.405	0.9	1	+1
		0.9	-1
0.81	0.81	0.81	0.81


0.81	0.9	1	+1
		0.9	-1
0.81	0.72	0.81	0.72

Algorithme de Value Iteration


Trouver V^* , avec états absorbants, actions déterministes, $\gamma = 0.9$:


- 1 Initialise aléatoirement toutes les valeurs, et 0 aux états absorbants;
- 2 $\forall s \in S: V(s) \leftarrow \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'))$;
- 3 Recommencer jusqu'à convergence des valeurs.

0.5	← 0	0.1	→ +1
		0.5	-1
← 1	← 0	0.8	→ 1 →

0.45	0.45	1	+1
		0.72	-1
0.9	0.9	0.9	0.9

0.405	0.9	1	+1
		0.9	-1
0.81	0.81	0.81	0.81

0.81	0.9	1	+1
		0.9	-1
0.81	0.72	0.81	0.72


0.81	0.9	1	+1
		0.9	-1
0.63	0.72	0.81	0.72

Algorithme de Value Iteration (2)

Trouver V^* sans états absorbants, actions déterministes, $\gamma = 0.9$:

- 1 Initialise aléatoirement toutes les valeurs, et 0 aux états absorbants;
- 2 Pour chaque $s \in S$:

$$V(s) = \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'));$$
- 3 Recommencer jusqu'à convergence des valeurs.


0.5	1	0.1	0.1 ¹
		-0.5	0.5 ¹
0.3	0	1	1


Algorithme de Value Iteration (2)

Trouver V^* sans états absorbants, actions déterministes, $\gamma = 0.9$:

- 1 Initialise aléatoirement toutes les valeurs, et 0 aux états absorbants;
- 2 Pour chaque $s \in S$:

$$V(s) = \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'));$$
- 3 Recommencer jusqu'à convergence des valeurs.

0.5	1	0.1	0.1^1
		-0.5	0.5^1
0.3	0	1	1


0.9	0.9	1.09	1.09^1
		0.9	1.09^1
0.27	0.9	0.9	0.9


Algorithme de Value Iteration (2)


Trouver V^* sans états absorbants, actions déterministes, $\gamma = 0.9$:

- 1 Initialise aléatoirement toutes les valeurs, et 0 aux états absorbants;
- 2 Pour chaque $s \in S$:

$$V(s) = \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'));$$
- 3 Recommencer jusqu'à convergence des valeurs.

0.5	1	0.1	0.1^+
		-0.5	0.5^+
0.3	0	1	1

0.9	0.9	1.09	1.09^+
		0.9	1.09^+
0.27	0.9	0.9	0.9


0.81	0.981	1.981	1.981^+
		0.981	1.981^+
0.81	0.81	0.81	0.981


Algorithme de Value Iteration (2)


Trouver V^* sans états absorbants, actions déterministes, $\gamma = 0.9$:


- 1 Initialise aléatoirement toutes les valeurs, et 0 aux états absorbants;
- 2 Pour chaque $s \in S$:

$$V(s) = \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'));$$
- 3 Recommencer jusqu'à convergence des valeurs.

0.5	1	0.1	0.1^1
		-0.5	0.5^1
0.3	0	1	1

0.9	0.9	1.09	1.09^1
		0.9	1.09^1
0.27	0.9	0.9	0.9

0.81	0.981	1.981	1.981^1
		0.981	1.981^1
0.81	0.81	0.81	0.981


7.2	8.1	9	10^1
		8.1	9^1
0.81	6.3	7.2	8.1

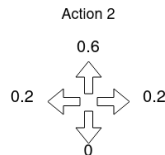
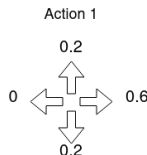
Algorithme de Value iteration (3)

Trouver V^* états absorbants, transitions **stochastiques**:

① Appliquez $\forall s \in S$:

$$V(s) = \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'));$$

0.5	1	0.5	+1
		1	-10
1	0.2	0.5	1



...

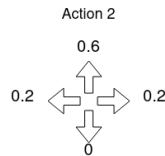
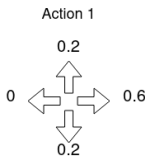
Algorithme de Value iteration (3)

Trouver V^* états absorbants, transitions **stochastiques**:

① Appliquez $\forall s \in S$:

$$V(s) = \max_a \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'));$$

0.5	1	0.5	+1
0		1	-10
1	0.2	0.5	1



...

0.7	0.8	1	+1
0.6		0.8	-10
0.84	0.68	0.9	0.9

Apprentissage par renforcement (RL)

Récompenses $R(s, a, s')$ et probabilités de transition $p(s'|s, a)$
inconnues:

- Impossible de calculer $\max_{a \in A} \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'))$.
- L'agent peut expérimenter transitions et récompenses (échantillonnage).

Apprentissage par renforcement (RL)

Récompenses $R(s, a, s')$ et probabilités de transition $p(s'|s, a)$
inconnues:

- Impossible de calculer $\max_{a \in A} \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V(s'))$.
- L'agent peut expérimenter transitions et récompenses (échantillonnage).

Solutions:

Model-based RL : Apprendre $p(s'|s, a)$ et $R(s, a, s')$ en échantillonnant.

Model-free RL : Approximer directement $V(s)$ en échantillonnant.

Model-free RL

Appliquer simultanément ou consécutivement:

- Générer des transitions avec $\pi(a|s)$ et apprendre $Q_\pi(s, a)$ et/ou $V_\pi(s)$. (*Policy evaluation*)
- Définir $\pi'(a|s)$ selon $\max_a Q_\pi(s, a)$ (Policy improvement).

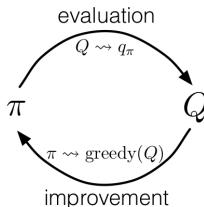
Model-free RL

Appliquer simultanément ou consécutivement:

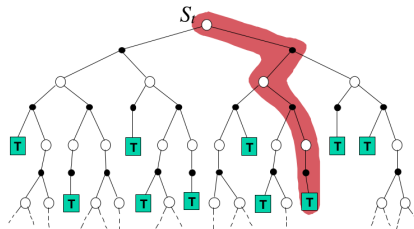
- Générer des transitions avec $\pi(a|s)$ et apprendre $Q_\pi(s, a)$ et/ou $V_\pi(s)$. (*Policy evaluation*)
- Définir $\pi'(a|s)$ selon $\max_a Q_\pi(s, a)$ (Policy improvement).

Policy improvement theorem:

If $\forall s \in S, V_\pi(s) \leq Q_\pi(s, \pi'(s))$, then: $\forall s \in S, V_\pi(s) \leq V_{\pi'}(s)$.



Policy evaluation: méthode Monte-Carlo

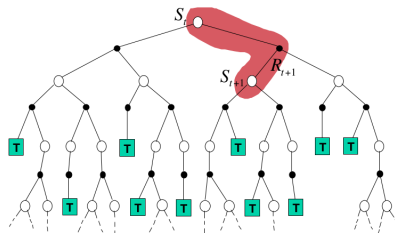


$$V(s_t) \leftarrow V(s_t) + \alpha \left[\sum_{t'=t}^{\infty} (\gamma^{t'-t} R(s_{t'}, a_{t'}, s_{t'+1})) - V(s_{t'}) \right] \quad (4)$$

où α est le taux d'apprentissage.

→ Approximation non biaisée, avec forte variance.

Policy evaluation: TD prediction

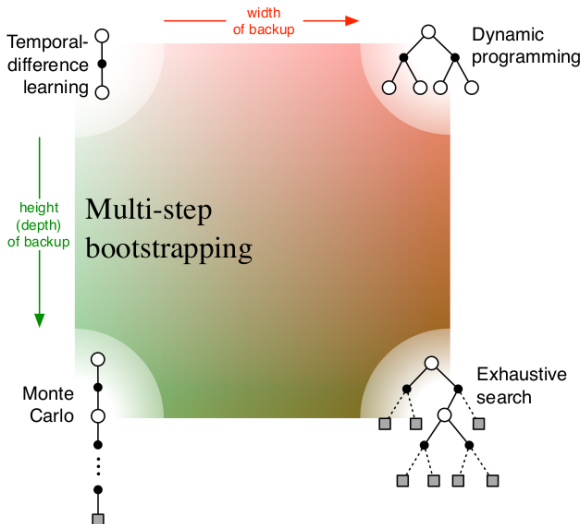


$$V(s_t) \leftarrow V(s_t) + \alpha [R(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) - V(s_t)] \quad (5)$$

où α est le taux d'apprentissage.

→ Approximation biaisée, avec faible variance.

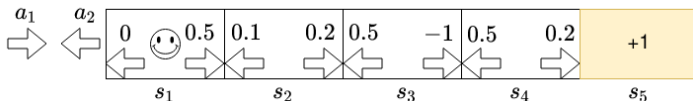
Vue unifiée



Q-learning

Choisir un état initial et recommencer, Q converge vers Q^* :

- 1 Choisir $a \sim \pi(\cdot|s) = \text{Cat}(Q(s, A))$
- 2 Observer r et s' .
- 3 $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \underbrace{\gamma \max_{a' \in A} Q(s', a')}_{\text{Policy improvement}} - Q(s, a)]$.

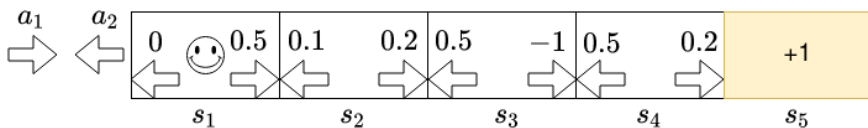


A, S	s_1	s_2	s_3	s_4
a_1	0.5	0.2	-1	0.2
a_2	0	0.1	0.5	0.5

Exemple Q-learning

Updates: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \underbrace{\gamma \max_{a' \in A} Q(s', a')}_{\text{Policy improvement}} - Q(s, a)].$

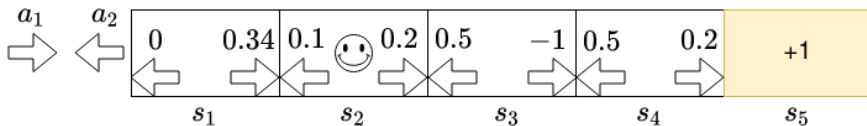
Exemple: $\alpha = 0.5$, $\gamma = 0.9$, politique déterministe:



Exemple Q-learning

Updates: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \underbrace{\gamma \max_{a' \in A} Q(s', a')}_{\text{Policy improvement}} - Q(s, a)].$

Exemple: $\alpha = 0.5$, $\gamma = 0.9$, politique déterministe:

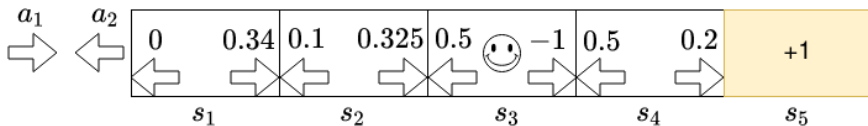


Exemple Q-learning

$$\text{Updates: } Q(s, a) \leftarrow Q(s, a) + \alpha[r + \underbrace{\gamma \max_{a' \in A} Q(s', a')}_{\text{Policy improvement}} - Q(s, a)].$$

Policy improvement

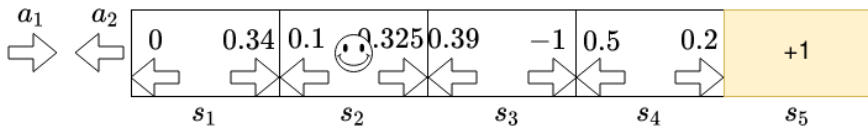
Exemple: $\alpha = 0.5$, $\gamma = 0.9$, politique déterministe:



Exemple Q-learning

Updates: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \underbrace{\gamma \max_{a' \in A} Q(s', a')}_{\text{Policy improvement}} - Q(s, a)].$

Exemple: $\alpha = 0.5$, $\gamma = 0.9$, politique déterministe:

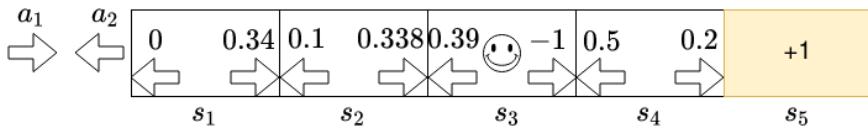


Exemple Q-learning

$$\text{Updates: } Q(s, a) \leftarrow Q(s, a) + \alpha[r + \underbrace{\gamma \max_{a' \in A} Q(s', a')}_{\text{Policy improvement}} - Q(s, a)].$$

Policy improvement

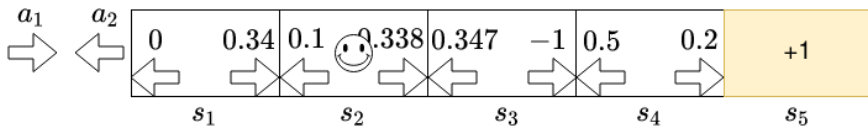
Exemple: $\alpha = 0.5$, $\gamma = 0.9$, politique déterministe:



Exemple Q-learning

Updates: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \underbrace{\gamma \max_{a' \in A} Q(s', a')}_{\text{Policy improvement}} - Q(s, a)].$

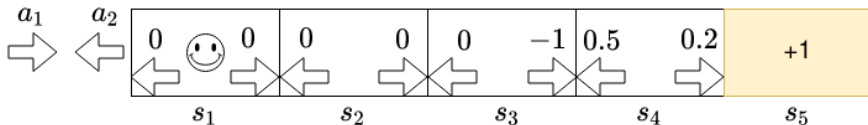
Exemple: $\alpha = 0.5$, $\gamma = 0.9$, politique déterministe:



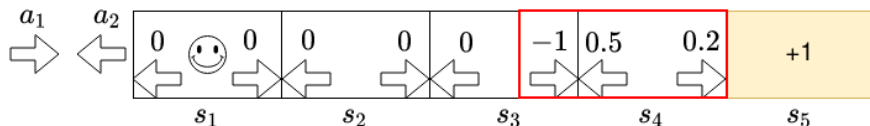
Exemple Q-learning

Updates: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \underbrace{\gamma \max_{a' \in A} Q(s', a')}_{\text{Policy improvement}} - Q(s, a)].$

Exemple: $\alpha = 0.5$, $\gamma = 0.9$, politique déterministe:



Compromis exploration vs exploitation



Exploration ϵ greedy: faire une action aléatoire avec probabilité ϵ .

- $\epsilon = 0.8$, l'agent ne converge pas et l'agent n'explore pas les bons chemins.
- $\epsilon = 0.001$, l'agent n'explore pas suffisamment les mauvais chemins.

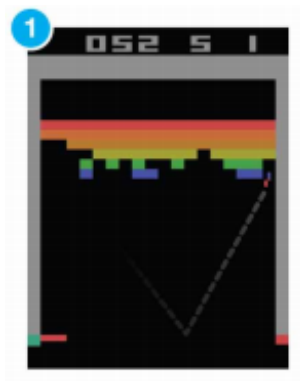
Off-policy vs On-policy

Off-policy : Converge peu importe l'exploration utilisée (Q-learning).

On-policy : Utilise l'exploration pour calculer l'opérateur de Bellman (SARSA).

$$\text{SARSA: } Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)].$$

Exemple d'espace d'état trop grand



Taille de l'espace d'état: $|S| = 255^{84 \times 84 \times 3 \times 2} = 255^{42336}$.

Peut-on tirer parti du deep learning pour généraliser ?

Approches

- 1 Méthodes "Value-based" (DQN): Choix des actions en fonction des valeurs $Q(s, a)$;

Approches

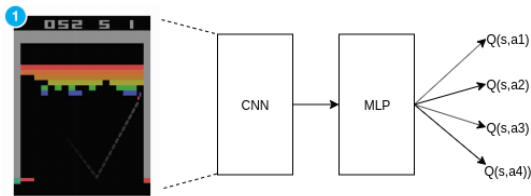
- ① Méthodes "Value-based" (DQN): Choix des actions en fonction des valeurs $Q(s, a)$;
- ② Méthodes "Policy-based" (REINFORCE): Paramétrage direct de la politique π_{θ} ;

Approches

- ➊ Méthodes "Value-based" (DQN): Choix des actions en fonction des valeurs $Q(s, a)$;
- ➋ Méthodes "Policy-based" (REINFORCE): Paramétrage direct de la politique π_{θ} ;
- ➌ Méthodes "Actor-critic" (A2C): Modification de π_{θ} en fonction de $Q(s, a)$.

DQN

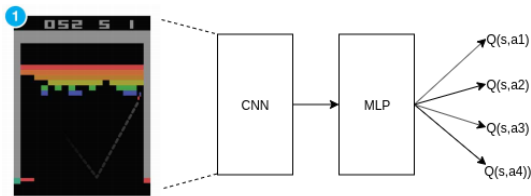
Deep Q-network (DQN)



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

DQN

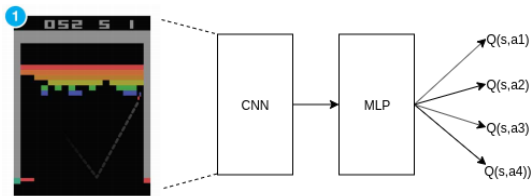
Deep Q-network (DQN)



- <https://www.youtube.com/watch?v=V1eYniJ0Rnk>
- $G_{\theta} = [Q_{\theta}(s_t, a_t) - (R(s_t, a_t) + \gamma \max_a Q_{\theta}(s_{t+1}, a))]^2$

DQN

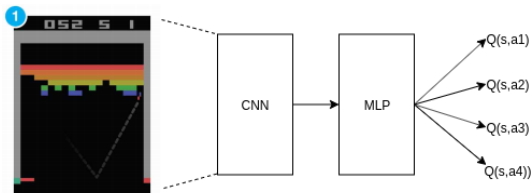
Deep Q-network (DQN)



- <https://www.youtube.com/watch?v=V1eYniJ0Rnk>
- $G_{\theta} = [Q_{\theta}(s_t, a_t) - (R(s_t, a_t) + \gamma \max_a Q_{\theta}(s_{t+1}, a))]^2$
- Minimisation de G_{θ} via backpropagation du gradient.

DQN

Deep Q-network (DQN)



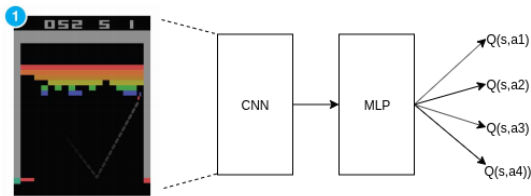
- <https://www.youtube.com/watch?v=V1eYniJ0Rnk>
- $G_{\theta} = [Q_{\theta}(s_t, a_t) - (R(s_t, a_t) + \gamma \max_a Q_{\theta}(s_{t+1}, a))]^2$
- Minimisation de G_{θ} via backpropagation du gradient.

Problèmes:

- 1 Q_{θ} (right) dépend aussi de θ , pouvant faire diverger l'algorithme.

DQN

Deep Q-network (DQN)



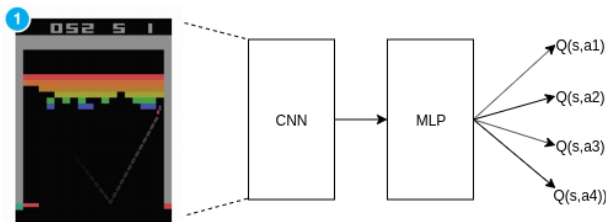
- <https://www.youtube.com/watch?v=V1eYniJ0Rnk>
- $G_{\theta} = [Q_{\theta}(s_t, a_t) - (R(s_t, a_t) + \gamma \max_a Q_{\theta}(s_{t+1}, a))]^2$
- Minimisation de G_{θ} via backpropagation du gradient.

Problèmes:

- 1 Q_{θ} (right) dépend aussi de θ , pouvant faire diverger l'algorithme.
- 2 Les exemples sont corrélés, rendant l'apprentissage instable.

DQN

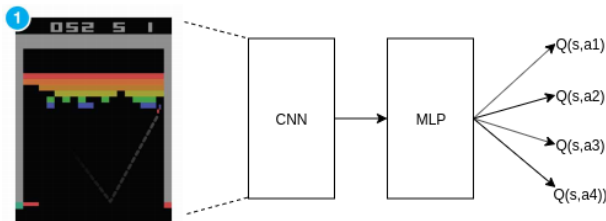
Deep Q-network (DQN) - Astuces



- $G_{\theta} = \left[Q_{\theta}(s_t, a_t) - (R(s_t, a_t) + \gamma \max_a \hat{Q}_{\theta'}(s_{t+1}, a)) \right]^2$
- Utilisation du target network: $\hat{Q}_{\theta'}$ est modifié graduellement vers Q_{θ} .

DQN

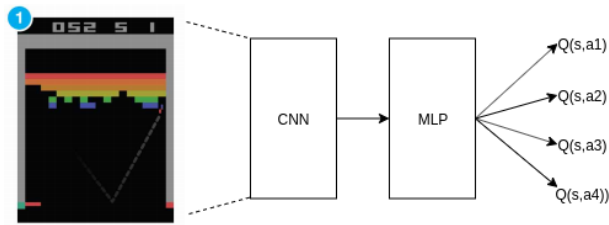
Deep Q-network (DQN) - Astuces



- $G_{\theta} = \left[Q_{\theta}(s_t, a_t) - (R(s_t, a_t) + \gamma \max_a \hat{Q}_{\theta'}(s_{t+1}, a)) \right]^2$
- Utilisation du target network: $\hat{Q}_{\theta'}$ est modifié graduellement vers Q_{θ} .
- Utilisation de l'experience replay : stockages des 100 000 (par exemple) dernières interactions.

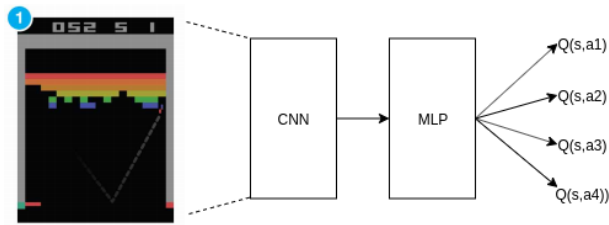
DQN

Deep Q-network (DQN) - Problème



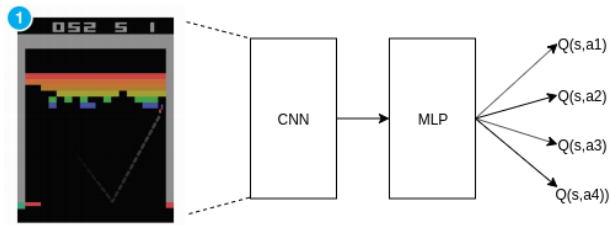
DQN

Deep Q-network (DQN) - Problème

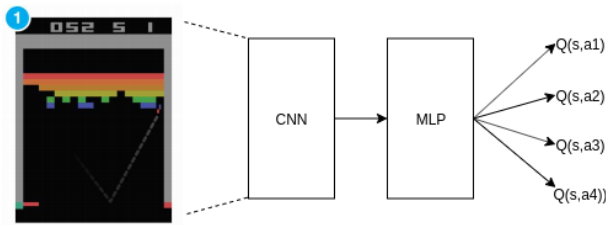


DQN

Deep Q-network (DQN) - Problème

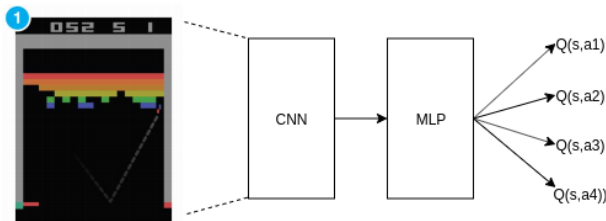


Deep Q-network (DQN) - Problème



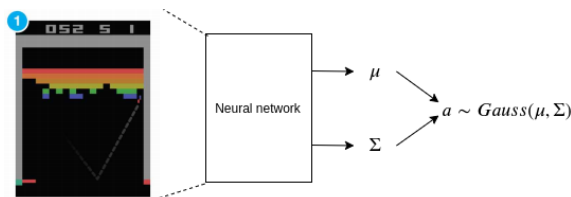
- Le réseau calcule $Q(s, a)$ pour chaque action.

Deep Q-network (DQN) - Problème



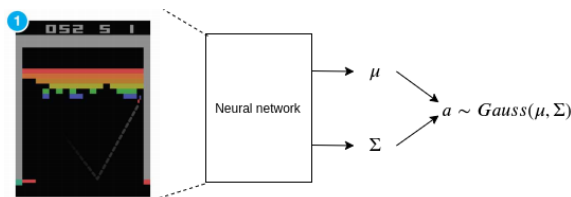
- Le réseau calcule $Q(s, a)$ pour chaque action.
- Mais comment faire si les actions sont continues ?

Méthodes policy-based



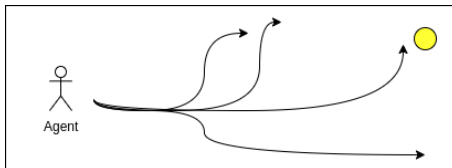
- Policy π génère une distribution de probabilité d'actions.

Méthodes policy-based



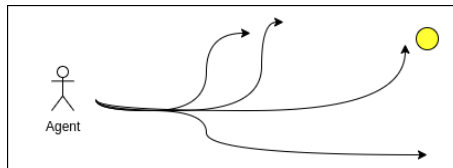
- Policy π génère une distribution de probabilité d'actions.
- On augmente la probabilité des actions générant de fortes récompenses.

Policy gradient théorème



Objectif: π_{θ} maximise $J(\theta) = \mathbb{E}_{\pi_{\theta}} R(\tau)$.

Policy gradient théorème



Objectif: π_{θ} maximise $J(\theta) = \mathbb{E}_{\pi_{\theta}} R(\tau)$.

Trouvons le gradient: $\nabla_{\theta} J(\theta)$.

$$\nabla_{\theta} J(\theta) = \nabla \mathbb{E}_{\pi_{\theta}} R(\tau)$$

(6)

$$\begin{aligned}
 \nabla_{\theta} J(\theta) &= \nabla \mathbb{E}_{\pi_{\theta}} R(\tau) \\
 &= \nabla \int \pi_{\theta}(\tau) R(\tau) d\tau
 \end{aligned}$$

(6)

$$\begin{aligned}
 \nabla_{\theta} J(\theta) &= \nabla \mathbb{E}_{\pi_{\theta}} R(\tau) \\
 &= \nabla \int \pi_{\theta}(\tau) R(\tau) d\tau \\
 &= \int \nabla \pi_{\theta}(\tau) R(\tau) d\tau
 \end{aligned}$$

(6)

$$\begin{aligned}
 \nabla_{\theta} J(\theta) &= \nabla \mathbb{E}_{\pi_{\theta}} R(\tau) \\
 &= \nabla \int \pi_{\theta}(\tau) R(\tau) d\tau \\
 &= \int \nabla \pi_{\theta}(\tau) R(\tau) d\tau \\
 &= \int \pi_{\theta}(\tau) \nabla \log \pi_{\theta}(\tau) R(\tau) d\tau.
 \end{aligned}
 \tag{6}$$

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \nabla \mathbb{E}_{\pi_{\theta}} R(\tau) \\ &= \nabla \int \pi_{\theta}(\tau) R(\tau) d\tau \\ &= \int \nabla \pi_{\theta}(\tau) R(\tau) d\tau \\ &= \int \pi_{\theta}(\tau) \nabla \log \pi_{\theta}(\tau) R(\tau) d\tau.\end{aligned}\tag{6}$$

$$\pi_{\theta}(\tau) = \rho(s_0) \prod_1^T \pi_{\theta}(a_t | s_t) p(s_{t+1}, r_{t+1} | s_t, a_t).$$

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \nabla \mathbb{E}_{\pi_{\theta}} R(\tau) \\ &= \nabla \int \pi_{\theta}(\tau) R(\tau) d\tau \\ &= \int \nabla \pi_{\theta}(\tau) R(\tau) d\tau \\ &= \int \pi_{\theta}(\tau) \nabla \log \pi_{\theta}(\tau) R(\tau) d\tau.\end{aligned}\tag{6}$$

$$\begin{aligned}\pi_{\theta}(\tau) &= \rho(s_0) \prod_1^T \pi_{\theta}(a_t | s_t) p(s_{t+1}, r_{t+1} | s_t, a_t). \\ \log \pi_{\theta}(\tau) &= C_{\theta} + \sum_1^T \log \pi_{\theta}(a_t | s_t).\end{aligned}$$

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \nabla \mathbb{E}_{\pi_{\theta}} R(\tau) \\ &= \nabla \int \pi_{\theta}(\tau) R(\tau) d\tau \\ &= \int \nabla \pi_{\theta}(\tau) R(\tau) d\tau \\ &= \int \pi_{\theta}(\tau) \nabla \log \pi_{\theta}(\tau) R(\tau) d\tau.\end{aligned}$$

(6)

$$\pi_{\theta}(\tau) = \rho(s_0) \prod_1^T \pi_{\theta}(a_t | s_t) p(s_{t+1}, r_{t+1} | s_t, a_t).$$

$$\log \pi_{\theta}(\tau) = C_{\theta} + \sum_1^T \log \pi_{\theta}(a_t | s_t).$$

$$\text{D'où } \nabla \log \pi_{\theta}(\tau) = \sum_1^T \nabla \log \pi_{\theta}(a_t | s_t).$$

Policy-based methods

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \nabla \mathbb{E}_{\pi_{\theta}} R(\tau) \\ &= \nabla \int \pi_{\theta}(\tau) R(\tau) d\tau \\ &= \int \nabla \pi_{\theta}(\tau) R(\tau) d\tau \\ &= \int \pi_{\theta}(\tau) \nabla \log \pi_{\theta}(\tau) R(\tau) d\tau.\end{aligned}$$

(6)

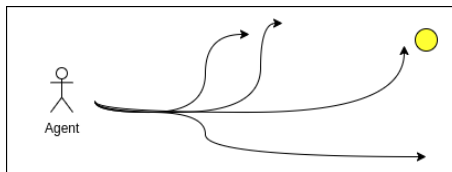
$$\pi_{\theta}(\tau) = \rho(s_0) \prod_1^T \pi_{\theta}(a_t | s_t) p(s_{t+1}, r_{t+1} | s_t, a_t).$$

$$\log \pi_{\theta}(\tau) = C_{\theta} + \sum_1^T \log \pi_{\theta}(a_t | s_t).$$

$$\text{D'où } \nabla \log \pi_{\theta}(\tau) = \sum_1^T \nabla \log \pi_{\theta}(a_t | s_t).$$

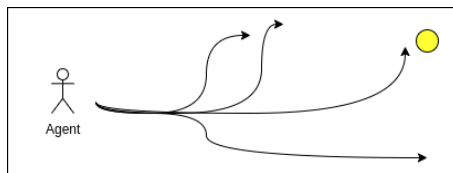
$$\text{On obtient: } \nabla \mathbb{E}_{\pi_{\theta}} R(\tau) = \mathbb{E}_{\pi_{\theta}} R(\tau) \sum_1^T \nabla \log \pi_{\theta}(a_t | s_t).$$

Interprétation du théorème



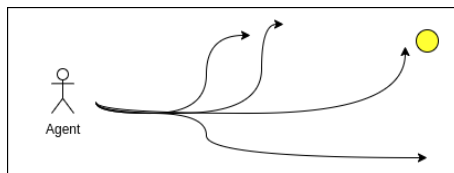
- $$\nabla_{\theta} J(\theta) = \mathbb{E}_{s_t \sim \pi_{\theta}} \left[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^T R(s_{t'}, a_{t'}, s_{t'+1}) \right].$$

Interprétation du théorème



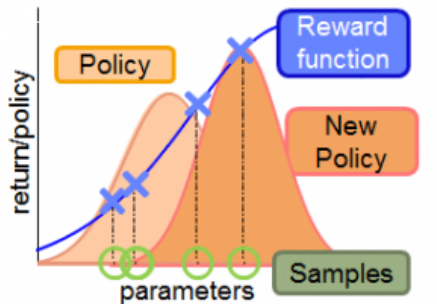
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{s_t \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^T R(s_{t'}, a_{t'}, s_{t'+1})]$.
- Seulement besoin des trajectoires et du gradient de la politique.

Interprétation du théorème



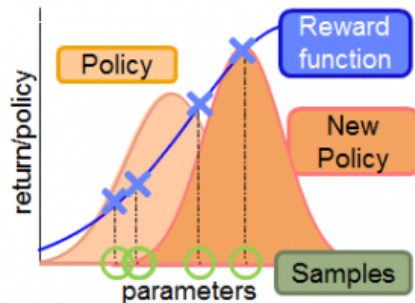
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{s_t \sim \pi_{\theta}} \left[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^T R(s_{t'}, a_{t'}, s_{t'+1}) \right].$
- Seulement besoin des trajectoires et du gradient de la politique.
- Estimateur non biaisé !

REINFORCE



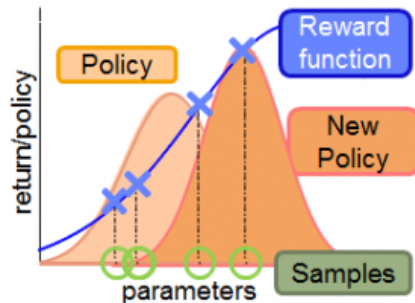
- $$\nabla_{\theta} J(\theta) = \mathbb{E}_{s_t \sim \pi_{\theta}} \left[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^T R(s_{t'}, a_{t'}, s_{t'+1}) \right].$$

REINFORCE



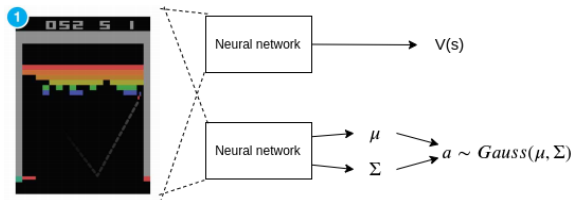
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{s_t \sim \pi_{\theta}} \left[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^T R(s_{t'}, a_{t'}, s_{t'+1}) \right]$.
- $\sum_{t'=t}^T R(s_{t'}, a_{t'}, s_{t'+1})$ "pondère" la probabilité de l'action.

REINFORCE



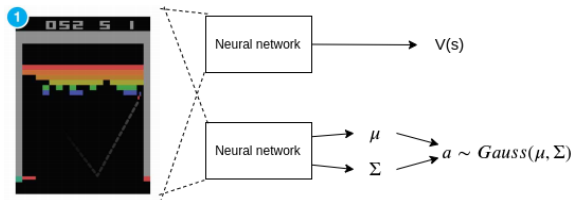
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{s_t \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^T R(s_{t'}, a_{t'}, s_{t'+1})]$.
- $\sum_{t'=t}^T R(s_{t'}, a_{t'}, s_{t'+1})$ "pondère" la probabilité de l'action.
- Mais pour de longs épisodes, la variance est très importante.

Architectures Actor-Critic



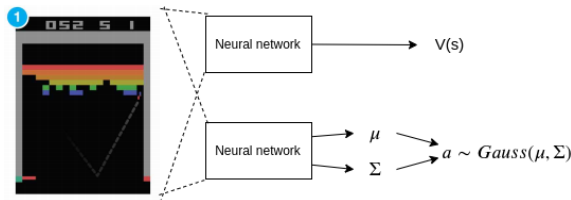
- $$\sum_{t'=t}^T \gamma^{t'-t} R(s_{t'}, a_{t'}, s_{t'+1}) = R(s_t, a_t, s_{t+1}) + \gamma V_{\theta'}(s_{t+1}) = Q(s_t, a_t).$$

Architectures Actor-Critic



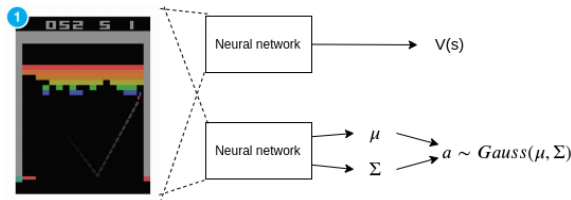
- $\sum_{t'=t}^T \gamma^{t'-t} R(s_{t'}, a_{t'}, s_{t'+1}) = R(s_t, a_t, s_{t+1}) + \gamma V_{\theta'}(s_{t+1}) = Q(s_t, a_t).$
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{a, s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) Q(s, a)].$

Architectures Actor-Critic



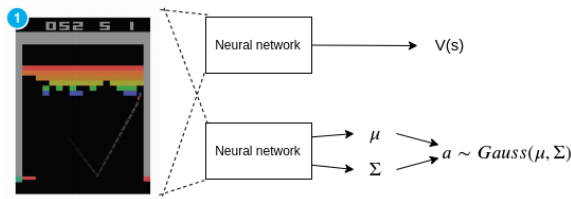
- $\sum_{t'=t}^T \gamma^{t'-t} R(s_{t'}, a_{t'}, s_{t'+1}) = R(s_t, a_t, s_{t+1}) + \gamma V_{\theta'}(s_{t+1}) = Q(s_t, a_t).$
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{a, s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) Q(s, a)].$
- Actor : Calculer $\log \pi_{\theta}(a|s).$

Architectures Actor-Critic



- $\sum_{t'=t}^T \gamma^{t'-t} R(s_{t'}, a_{t'}, s_{t'+1}) = R(s_t, a_t, s_{t+1}) + \gamma V_{\theta'}(s_{t+1}) = Q(s_t, a_t).$
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{a, s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) Q(s, a)].$
- Actor : Calculer $\log \pi_{\theta}(a|s).$
- Critic : Calculer $V_{\theta'}(s')$ ou $Q_{\theta'}(s, a).$

Réduction de la variance



$$\nabla_{\theta} J(\theta) = \mathbb{E}_{a,s,s' \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) (Q(s, a) - b(s))]$$

- Baseline $b(s)$ indépendante de l'action pour ne pas biaiser le gradient.

Dérivation

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{a,s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s)(Q(s, a) - b(s))]$$

Dérivation

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{a,s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s)(Q(s,a) - b(s))] \\ &= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} \nabla_{\theta} \log \pi_{\theta}(a|s) b(s)\end{aligned}$$

Dérivation

$$\begin{aligned}
 \nabla_{\theta} J(\theta) &= \mathbb{E}_{a,s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s)(Q(s,a) - b(s))] \\
 &= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} \nabla_{\theta} \log \pi_{\theta}(a|s) b(s) \\
 &= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} b(s) \nabla_{\theta} \log \pi_{\theta}(a|s)
 \end{aligned}$$

Dérivation

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{a,s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) (Q(s, a) - b(s))] \\ &= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} \nabla_{\theta} \log \pi_{\theta}(a|s) b(s) \\ &= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} b(s) \nabla_{\theta} \log \pi_{\theta}(a|s) \\ &= \nabla_{\theta} J_{prev}(\theta) - \sum_s \mu(s) b(s) \sum_a \pi_{\theta}(a|s) \nabla_{\theta} \log \pi_{\theta}(a|s)\end{aligned}$$

Dérivation

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{a,s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s)(Q(s,a) - b(s))] \\&= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} \nabla_{\theta} \log \pi_{\theta}(a|s) b(s) \\&= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} b(s) \nabla_{\theta} \log \pi_{\theta}(a|s) \\&= \nabla_{\theta} J_{prev}(\theta) - \sum_s \mu(s) b(s) \sum_a \pi_{\theta}(a|s) \nabla_{\theta} \log \pi_{\theta}(a|s) \\&= \nabla_{\theta} J_{prev}(\theta) - \sum_s \mu(s) b(s) \sum_a \pi_{\theta}(a|s) \frac{\nabla_{\theta} \pi_{\theta}(a|s)}{\pi_{\theta}(a|s)}\end{aligned}$$

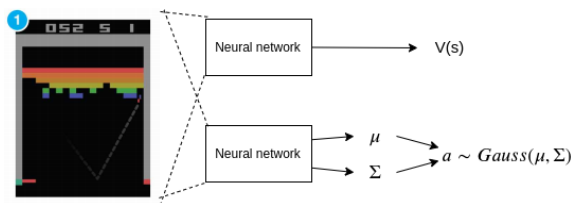
Dérivation

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{a,s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) (Q(s, a) - b(s))] \\&= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} \nabla_{\theta} \log \pi_{\theta}(a|s) b(s) \\&= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} b(s) \nabla_{\theta} \log \pi_{\theta}(a|s) \\&= \nabla_{\theta} J_{prev}(\theta) - \sum_s \mu(s) b(s) \sum_a \pi_{\theta}(a|s) \nabla_{\theta} \log \pi_{\theta}(a|s) \\&= \nabla_{\theta} J_{prev}(\theta) - \sum_s \mu(s) b(s) \sum_a \pi_{\theta}(a|s) \frac{\nabla_{\theta} \pi_{\theta}(a|s)}{\pi_{\theta}(a|s)} \\&= \nabla_{\theta} J_{prev}(\theta) - \sum_s \mu(s) b(s) \nabla_{\theta} 1\end{aligned}$$

Dérivation

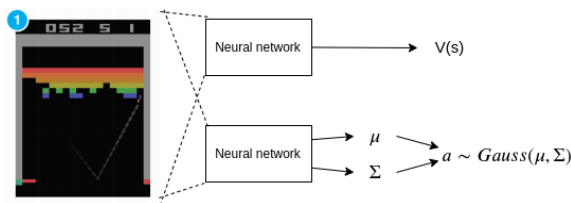
$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{a,s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s)(Q(s, a) - b(s))] \\&= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} \nabla_{\theta} \log \pi_{\theta}(a|s) b(s) \\&= \nabla_{\theta} J_{prev}(\theta) - \mathbb{E}_{a,s \sim \pi_{\theta}} b(s) \nabla_{\theta} \log \pi_{\theta}(a|s) \\&= \nabla_{\theta} J_{prev}(\theta) - \sum_s \mu(s) b(s) \sum_a \pi_{\theta}(a|s) \nabla_{\theta} \log \pi_{\theta}(a|s) \\&= \nabla_{\theta} J_{prev}(\theta) - \sum_s \mu(s) b(s) \sum_a \pi_{\theta}(a|s) \frac{\nabla_{\theta} \pi_{\theta}(a|s)}{\pi_{\theta}(a|s)} \\&= \nabla_{\theta} J_{prev}(\theta) - \sum_s \mu(s) b(s) \nabla_{\theta} 1 \\&= \nabla_{\theta} J_{prev}(\theta)\end{aligned}$$

A2C: réduire la variance



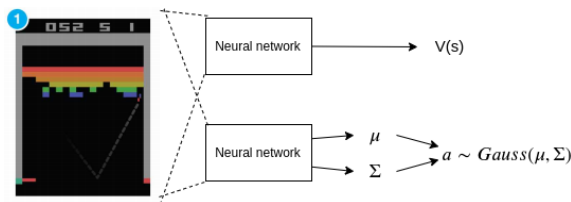
- $$\nabla_{\theta} J(\theta) = \mathbb{E}_{a, s \sim \pi_{\theta}} \left[\nabla_{\theta} \log \pi_{\theta}(a|s) (Q_{\theta'}(s, a) - V_{\theta'}(s)) \right].$$

A2C: réduire la variance



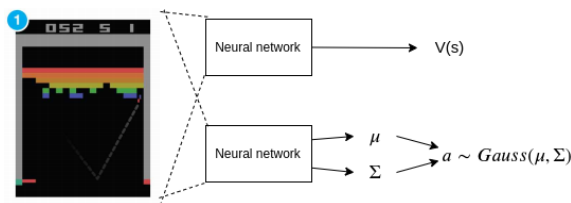
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{a, s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) (Q_{\theta'}(s, a) - V_{\theta'}(s))]$.
- $A(s, a) = Q_{\theta'}(s, a) - V_{\theta'}(s)$ fonction avantage.

A2C: réduire la variance



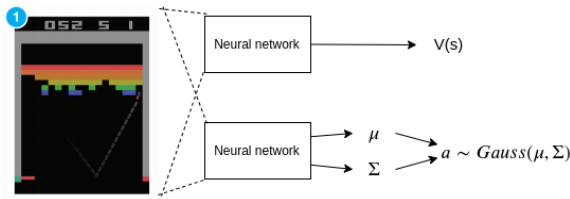
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{a, s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) (Q_{\theta'}(s, a) - V_{\theta'}(s))]$.
- $A(s, a) = Q_{\theta'}(s, a) - V_{\theta'}(s)$ fonction avantage.
- $A(s, a) > 0$ si a est meilleure que la politique actuelle.

A2C: réduire la variance



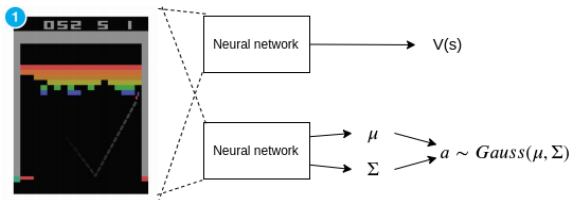
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{a,s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) (Q_{\theta'}(s, a) - V_{\theta'}(s))]$.
- $A(s, a) = Q_{\theta'}(s, a) - V_{\theta'}(s)$ fonction avantage.
- $A(s, a) > 0$ si a est meilleure que la politique actuelle.
- $A(s, a) < 0$ si a est moins bonne que la politique actuelle.

A2C: Explorer



- Convergence prématurée.

A2C: Explorer



- Convergence prématurée.
- Ajout d'un terme d'entropie forçant une forte variance.
- $\nabla_{\theta} J(\theta) = \mathbb{E}_{a, s \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) A(s, a) + \nabla_{\theta} H(\pi_{\theta}(a|s))].$

Autres modèles...

Améliorations du DQN:

- Double DQN [Van Hasselt et al., 2016].

Autres modèles...

Améliorations du DQN:

- Double DQN [Van Hasselt et al., 2016].
- Prioritized Experience replay [Schaul et al., 2015].

Autres modèles...

Améliorations du DQN:

- Double DQN [Van Hasselt et al., 2016].
- Prioritized Experience replay [Schaul et al., 2015].
- Dueling DQN [Wang et al., 2015].

Autres modèles...

Améliorations du DQN:

- Double DQN [Van Hasselt et al., 2016].
- Prioritized Experience replay [Schaul et al., 2015].
- Dueling DQN [Wang et al., 2015].
- Distributional DQN [Dabney et al., 2018].

Autres modèles...

Améliorations du DQN:

- Double DQN [Van Hasselt et al., 2016].
- Prioritized Experience replay [Schaul et al., 2015].
- Dueling DQN [Wang et al., 2015].
- Distributional DQN [Dabney et al., 2018].
- Rainbow DQN [Hessel et al., 2018].

Autres modèles...

Améliorations du DQN:

- Double DQN [Van Hasselt et al., 2016].
- Prioritized Experience replay [Schaul et al., 2015].
- Dueling DQN [Wang et al., 2015].
- Distributional DQN [Dabney et al., 2018].
- Rainbow DQN [Hessel et al., 2018].

Modèles Actor-Critic:

- Trust region policy optimization [Schulman et al., 2015].

Autres modèles...

Améliorations du DQN:

- Double DQN [Van Hasselt et al., 2016].
- Prioritized Experience replay [Schaul et al., 2015].
- Dueling DQN [Wang et al., 2015].
- Distributional DQN [Dabney et al., 2018].
- Rainbow DQN [Hessel et al., 2018].

Modèles Actor-Critic:

- Trust region policy optimization [Schulman et al., 2015].
- Proximal policy optimization [Schulman et al., 2017].

Autres modèles...

Améliorations du DQN:

- Double DQN [Van Hasselt et al., 2016].
- Prioritized Experience replay [Schaul et al., 2015].
- Dueling DQN [Wang et al., 2015].
- Distributional DQN [Dabney et al., 2018].
- Rainbow DQN [Hessel et al., 2018].

Modèles Actor-Critic:

- Trust region policy optimization [Schulman et al., 2015].
- Proximal policy optimization [Schulman et al., 2017].
- Deep deterministic policy gradient [Lillicrap et al., 2015].

Autres modèles...

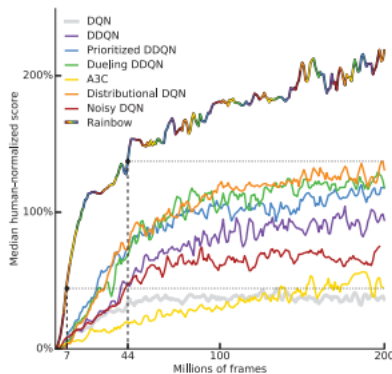
Améliorations du DQN:

- Double DQN [Van Hasselt et al., 2016].
- Prioritized Experience replay [Schaul et al., 2015].
- Dueling DQN [Wang et al., 2015].
- Distributional DQN [Dabney et al., 2018].
- Rainbow DQN [Hessel et al., 2018].

Modèles Actor-Critic:

- Trust region policy optimization [Schulman et al., 2015].
- Proximal policy optimization [Schulman et al., 2017].
- Deep deterministic policy gradient [Lillicrap et al., 2015].
- Soft actor-critic [Haarnoja et al., 2018].

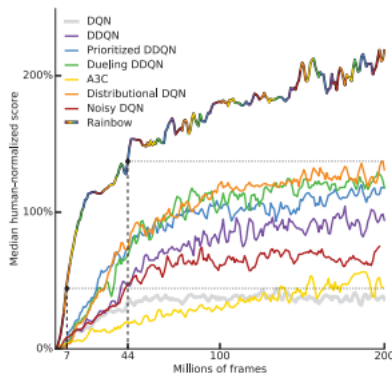
Efficacité computationnelle



- En réalité : pas de simulateurs.

Problèmes du RL

Efficacité computationnelle



- En réalité : pas de simulateurs.
- 200k frames = 1h humaine; 44M = 220h humaines.

Exploration

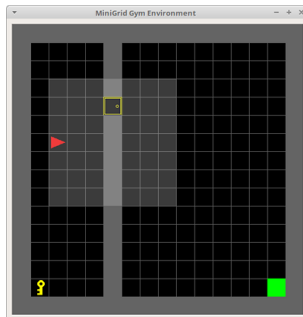
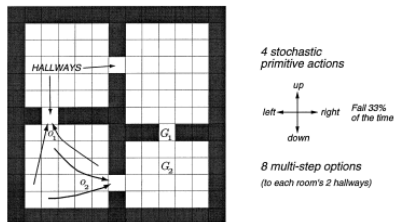


Figure: Environnement très simple avec des récompenses éparses.

- L'agent n'apprend rien car il ne trouve jamais la récompense.

Abstraction des décisions



- Nous sommes capable de prendre des décisions de haut niveau.
- Plus facile d'apprendre sur 10 actions haut-niveau que sur 1000 actions bas-niveau [Sutton et al., 1999].

Motivation intrinsèque

- Issu de la psychologie pour décrire la tendance des bébés à explorer;
- Faire quelque chose pour son inhérente satisfaction;



Motivation intrinsèque

- Issu de la psychologie pour décrire la tendance des bébés à explorer;
- Faire quelque chose pour son inhérente satisfaction;
- RL: intrinsic reward vs extrinsic reward.



Curiosité

- Récompenser l'erreur de prédiction des états suivants [Pathak et al., 2017].

<https://pathak22.github.io/noreward-rl/>

- $R(s, a, s') = ||forward(e(s), a) - e(s')||^2$.

Curiosité

- Récompenser l'erreur de prédiction des états suivants [Pathak et al., 2017].
<https://pathak22.github.io/noreward-rl/>
- $R(s, a, s') = ||forward(e(s), a) - e(s')||^2$.
- Récompenser des états loins de ceux en mémoire [Savinov et al., 2018] <https://ai.googleblog.com/2018/10/curiosity-and-procrastination-in.html>.

Curiosité

- Récompenser l'erreur de prédiction des états suivants [Pathak et al., 2017].
<https://pathak22.github.io/noreward-rl/>
- $R(s, a, s') = ||forward(e(s), a) - e(s')||^2$.
- Récompenser des états loins de ceux en mémoire [Savinov et al., 2018] <https://ai.googleblog.com/2018/10/curiosity-and-procrastination-in.html>.
- Récompenser l'agent selon la nouveauté des états [Bellemare et al., 2016][Ostrovski et al., 2017].
- $R(s, a, s') = \frac{1}{count(s')}.$


Ressources et références I

Ressources (images) :

- <https://machinelearnia.com/apprentissage-supervise-4-etapes/>
- <https://needemand.com/quest-ce-que-le-machine-learning/>
- <http://karpathy.github.io/2016/05/31/rl/>
- <https://blog.goodaudience.com/deep-q-learning-a-reinforcement-learning-algorithm-d1a93b754535>
- Machine Learning Summer Schools 2020

Références:

- Excellent livre de Sutton gratuit :
<http://incompleteideas.net/book/the-book.html>
- Cours en ligne de David Silver
- Cours de Berkeley

 Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016).

Unifying count-based exploration and intrinsic motivation.

In *Advances in Neural Information Processing Systems*, pages 1471–1479.

 Dabney, W., Rowland, M., Bellemare, M., and Munos, R. (2018).


Distributional reinforcement learning with quantile regression.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

 Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018).

Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

In *International Conference on Machine Learning*, pages 1861–1870. PMLR.

 Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018).