

Introduction au Machine Learning

Apprentissage par renforcement

Emmanuel Dellandréa

emmanuel.dellandrea@ec-lyon.fr

Version du 13/02/2025



Apprentissage par renforcement ?

- Apprentissage par renforcement (AR) : acquisition automatisée de compétences pour la prise de décisions (actions ou contrôle) en milieu complexe et incertain.
 - Un agent a la capacité d'agir
 - Chaque action influence l'état futur de l'agent
 - Le succès est mesuré par une valeur de récompense
 - Objectif : sélectionner les actions pour maximiser la récompense future
 - Apprentissage par essais/erreurs

Caractéristiques de l'AR

- Différences entre l'apprentissage par renforcement et les autres paradigmes d'apprentissage automatique :
 - Pas de superviseur, uniquement une récompense
 - Le retour d'information est différé, non instantané
 - Processus séquentiel
 - Les actions de l'agent influent sur les données qu'il recevra par la suite

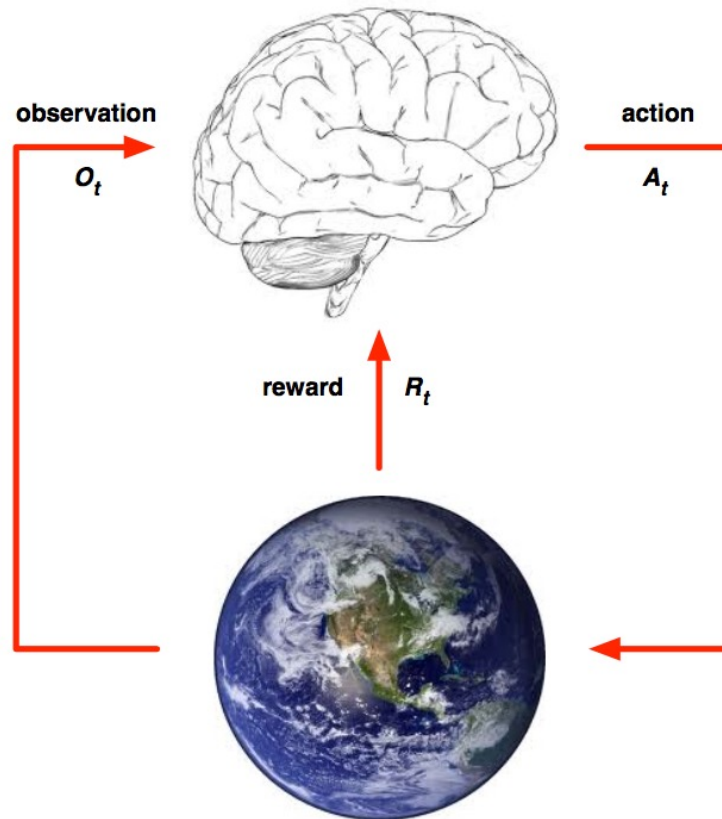
Récompense

- Une récompense R_t est une valeur scalaire d'un signal de retour d'information
- Elle indique si l'agent agit correctement ou non à l'instant t
- Le but de l'agent est de maximiser les récompenses futures cumulées

Exemples de récompenses

- **Pilotage automatique d'un hélicoptère**
 - récompense + si la trajectoire désirée est suivie
 - récompense - si crash
- **Jouer à un jeu vidéo**
 - récompense + si augmentation du score
 - récompense - si diminution du score
- **Contrôle d'une centrale électrique**
 - récompense + si production électrique correcte
 - récompense - si production dépasse les seuils de sécurité
- **Marche d'un robot humanoïde**
 - récompense + si marche
 - récompense - si chute

L'agent et son environnement



- A chaque instant t , l'agent :
 - Reçoit une observation O_t
 - Exécute une action A_t
 - Reçoit une récompense R_t
- L'environnement :
 - Reçoit l'action A_t
 - Emet la récompense R_t
 - Emet l'observation O_{t+1}
- Incrémentation de t

Etat

- Une expérience (ou historique) est une séquence d'observations, d'actions et de récompenses :
 - $o_1, a_1, r_1, \dots, o_t, a_t, r_t$
- Ce qui se passera par la suite dépend de l'expérience :
 - L'agent sélectionne des actions
 - L'environnement sélectionne des observations/récompenses
- Un état est l'information utilisée pour déterminer ce qui se passera par la suite
- Un état est un résumé d'une expérience :
 - $s_t = f(o_1, a_1, r_1, \dots, o_t, a_t, r_t)$
- Dans un environnement complètement observable
 - $s_t = f(o_t)$

Environnement complètement observable

- L'agent observe directement l'état de l'environnement

$$O_t = S_t^a = S_t^e$$

- Etat agent = Etat environnement = Etat d'information
- Il s'agit d'un Processus de Décision Markovien (MDP)

Processus de décision Markovien

- Un processus de décision Markovien (MDP) est un tuple (S, A, P, R, γ)
 - S est un ensemble fini d'états
 - A un est ensemble fini d'actions
 - P est une matrice de probabilités de transitions

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

- R est une fonction de récompense

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

- γ est un facteur d'actualisation

Composants d'un agent

- Un agent peut inclure un ou plusieurs de ces composants :
 - Une politique d'action : la fonction de comportement de l'agent
 - Une fonction valeur : qualité d'un état et/ou d'une action
 - Un modèle : représentation qu'a l'agent de son environnement

Politique

- Une politique correspond au comportement de l'agent
- Fait correspondre un état à une action
 - Politique déterministe : $a = \pi(s)$
 - Politique stochastique : $\pi(a|s) = \mathbb{P}[a|s]$

Fonction valeur

- Une fonction valeur est la prédiction de la récompense future :
 - « Quelle récompense vais-je obtenir de l'action a à partir de l'état s ? »
- Est utilisée pour évaluer la qualité des états
- Et pour sélectionner les actions :

$$v_{\pi}(s) = \mathbb{E}_{\pi} [R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s]$$

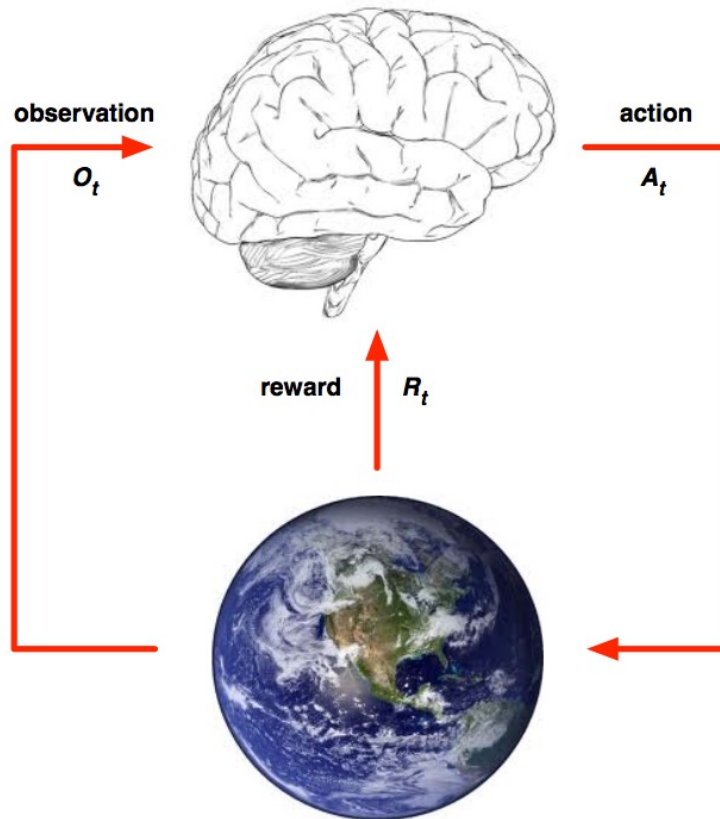
Le modèle

- Un modèle prédit ce que l'environnement fera par la suite
- \mathcal{P} prédit l'état suivant
- \mathcal{R} prédit la récompense immédiate suivante :

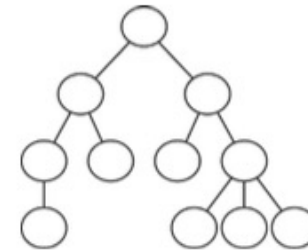
$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

Le modèle



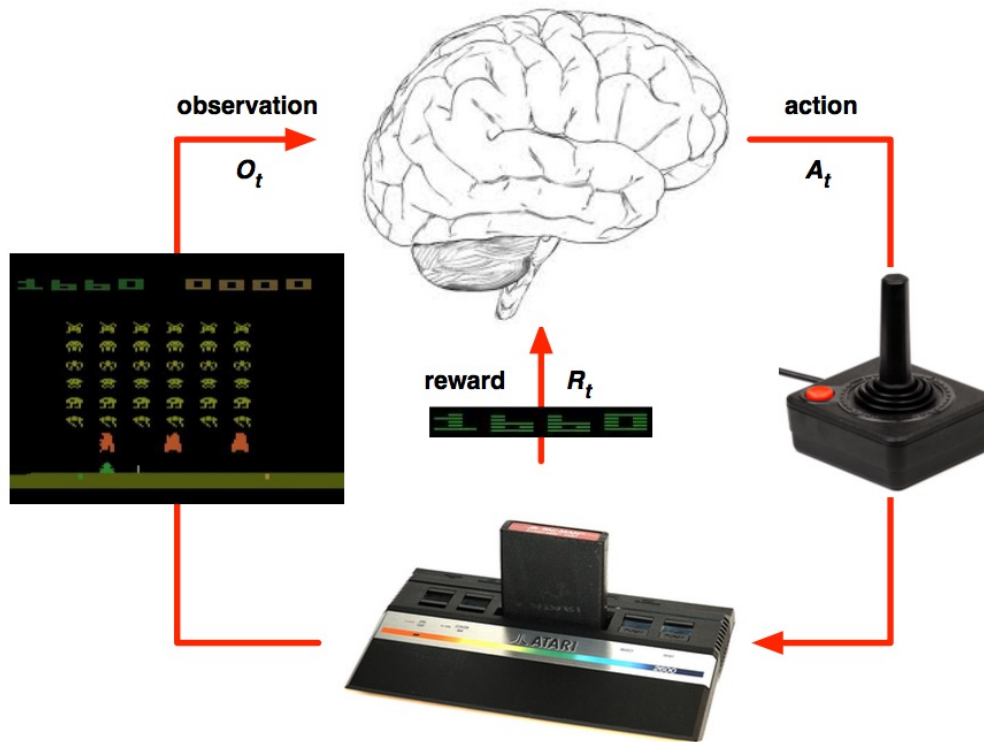
- Le modèle est appris à partir de l'expérience
- Est utilisé pour la planification des actions



Apprentissage et planification

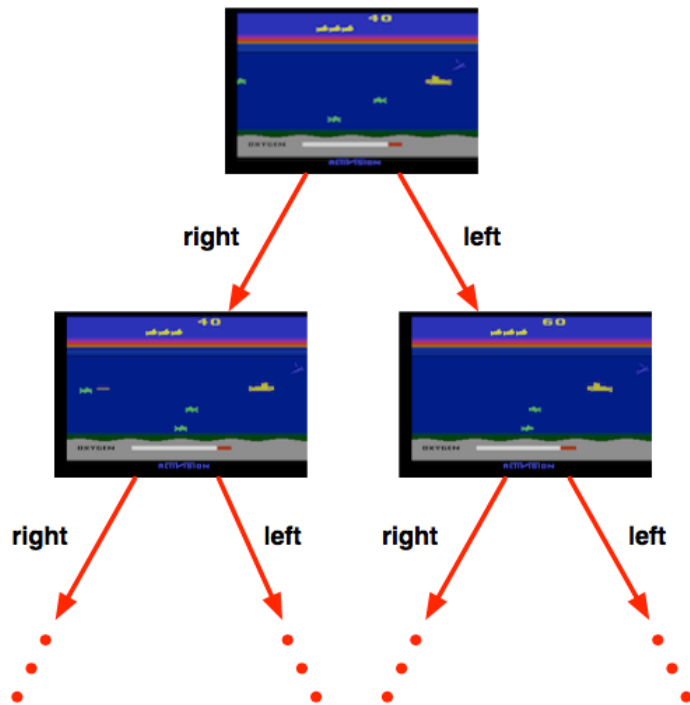
- Deux problèmes fondamentaux pour la prise de décision séquentielle
 - Apprentissage par renforcement :
 - L'environnement est initialement inconnu
 - L'agent interagit avec l'environnement
 - L'agent améliore sa politique
 - Planification :
 - Un modèle de l'environnement est connu
 - L'agent réalise des calculs à partir de ce modèle
 - L'agent améliore sa politique

Exemple ATARI : Apprentissage par Renforcement



- Les règles du jeu ne sont pas connues
- Apprentissage directement à partir du gameplay interactif
- Sélection d'actions sur le joystick, visualisation des pixels et du score

Exemple ATARI : Planification



- Les règles du jeu sont connues
- Peut interroger l'émulateur (l'agent possède un modèle parfait)
- Si l'action a est prise à partir de l'état s :
 - quel sera le prochain état ?
 - quel sera le score ?
- Planification pour trouver la politique optimale (tree search)

Exploration et exploitation

- L'apprentissage par renforcement est un apprentissage par essai/erreur
- L'agent doit découvrir une bonne politique
 - à partir d'expériences avec l'environnement
 - sans perdre trop de récompense durant le processus

Exploration et exploitation

- L'exploration permet de trouver plus d'information sur l'environnement
- L'exploitation utilise l'information connue pour maximiser la récompense
- Il est généralement important d'à la fois explorer et exploiter

Exemples

- Sélection d'un restaurant
 - Exploitation : aller à son restaurant préféré
 - Exploration : essayer un nouveau restaurant
- Bannière de publicité en ligne
 - Exploitation : montrer la publicité ayant le plus de succès
 - Exploration : montrer une publicité différente
- Jeu vidéo
 - Exploitation : réaliser le mouvement supposé le meilleur
 - Exploration : réaliser un mouvement expérimental

Approches pour l'AR

- AR basé sur la fonction valeur :
 - Estimation de l'optimal de la fonction valeur
 - Il s'agit de la valeur maximale atteignable quelle que soit la politique
- AR basé sur la politique :
 - Recherche directe de la politique optimale
 - Il s'agit de la politique permettant d'obtenir la récompense future maximale
- AR basé sur un modèle :
 - Construction d'un modèle de l'environnement
 - Planification des actions en utilisant ce modèle

AR basé sur la fonction valeur

- Rappel : la fonction valeur est une prédiction de la récompense future

$$v_{\pi}(s) = \mathbb{E}_{\pi} [R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s]$$

- On utilise plus souvent la fonction action-valeur (Q-value) qui fournit la récompense future totale attendue :
 - à partir d'un état s et d'une action a
 - sous une politique
 - avec un facteur d'actualisation

$$Q^{\pi}(s, a) = \mathbb{E} [r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \mid s, a]$$

Fonction Q-value

- La fonction action-valeur peut se décomposer selon l'équation de Bellman :

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

Fonction Q-value optimale

- Une fonction valeur optimale est le maximum atteignable

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a) = Q^{\pi^*}(s, a)$$

- Une fois cette fonction connue, on peut agir de manière optimale

$$\pi^*(s) = \operatorname{argmax}_a Q^*(s, a)$$

Fonction Q-value optimale

- Les valeurs optimales sont maximisées en considérant toutes les décisions

$$\begin{aligned} Q^*(s, a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \dots \\ &= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned}$$

- Les valeurs optimales peuvent se décomposer selon l'équation de Bellman

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

Q-learning

- Q-learning : famille d'algorithmes d'apprentissage par renforcement dont l'objectif est d'approximer l'optimal de la fonction action-valeur Q
- Une fois cette fonction calculée, l'agent pourra choisir les meilleures actions
- Deux exemples de Q-learning :
 - Approximation de la fonction Q par une méthode incrémentale
 - Approximation de la fonction Q par un réseau de neurones profond : « deep reinforcement learning »

Q-learning

■ Principe :

- à chaque instant t , l'agent reçoit l'observation o_t
- il choisit une action a_t
- l'environnement fournit alors à l'agent une observation o_{t+1} et une récompense R_t
- La fonction Q est mise à jour, l'objectif de l'apprentissage étant de trouver la séquence d'actions la maximisant

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{actuel}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{vitesse d'apprentissage}} \cdot \left(\underbrace{R_{t+1}}_{\text{recompense}} + \underbrace{\gamma}_{\text{facteur d'actualisation}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\substack{\text{valeur apprise} \\ \text{valeur optimale estimée}}} - \underbrace{Q_t(s_t, a_t)}_{\text{actuel}} \right)$$

Q-learning

- Vitesse d'apprentissage :
 - Détermine l'importance de la nouvelle information calculée par rapport à la précédente
 - Une valeur de 0 ne fera rien apprendre à l'agent
 - Une valeur de 1 ne fera considérer à l'agent que la nouvelle information
 - En pratique, la vitesse est souvent proche de 0,1 pour toute la durée du processus

Q-learning

- Facteur d'actualisation :
 - Détermine l'importance de récompenses futures
 - Un facteur de 0 rend l'agent myope en ne considérant que les récompenses courantes
 - Un facteur de 1 ferait aussi intervenir les récompenses plus lointaines

Algorithme de Q-learning

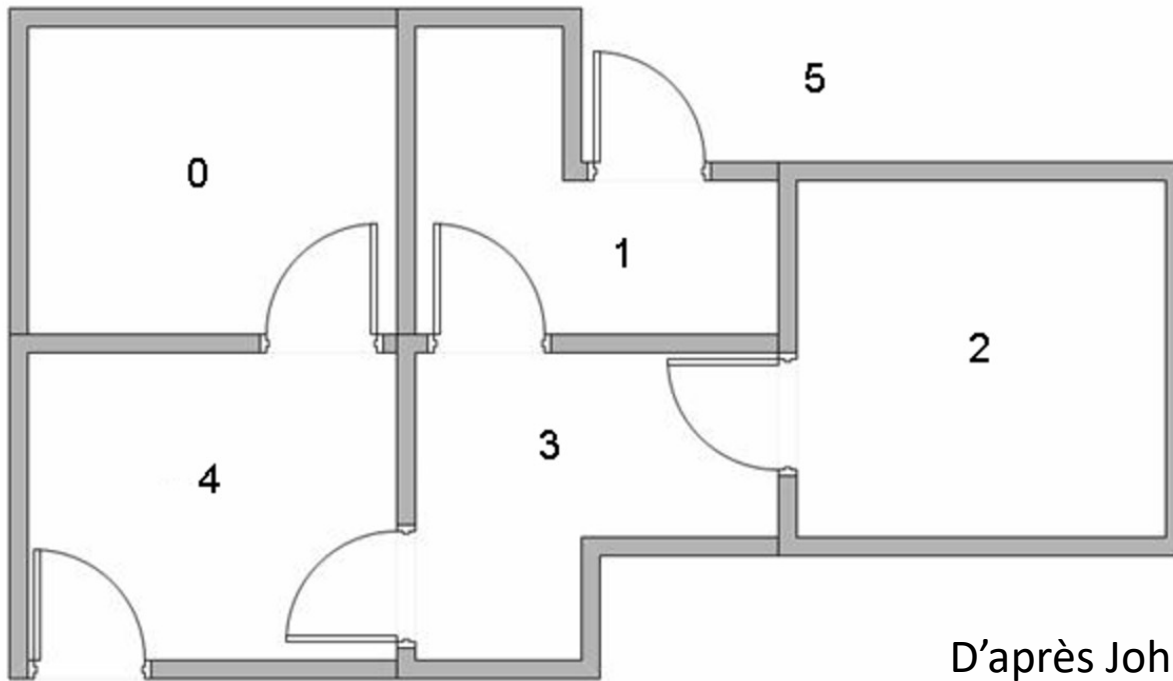
- Fixer les valeurs de gamma, alpha et la matrice de récompenses R
- Initialiser la matrice Q à zéro
- Pour chaque épisode :
 - Sélectionner un état initial aléatoirement
 - Faire :
 - Sélectionner une action parmi les actions possibles à partir de cet état
 - Considérer l'état suivant défini par cette action
 - Déterminer la valeur maximale de Q pour l'état suivant en considérant toutes les actions possibles
 - Mettre à jour Q selon

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{actuel}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{vitesse d'apprentissage}} \cdot \left(\underbrace{R_{t+1}}_{\text{recompense}} + \underbrace{\gamma}_{\text{facteur d'actualisation}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{valeur optimale estimée}} - \underbrace{Q_t(s_t, a_t)}_{\text{actuel}} \right)$$

- Mettre à jour l'état courant $s_{t+1} \leftarrow s_t$
- Tant que l'état final n'est pas atteint

Q-learning : un exemple

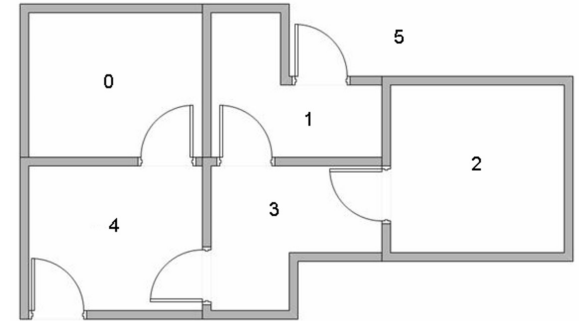
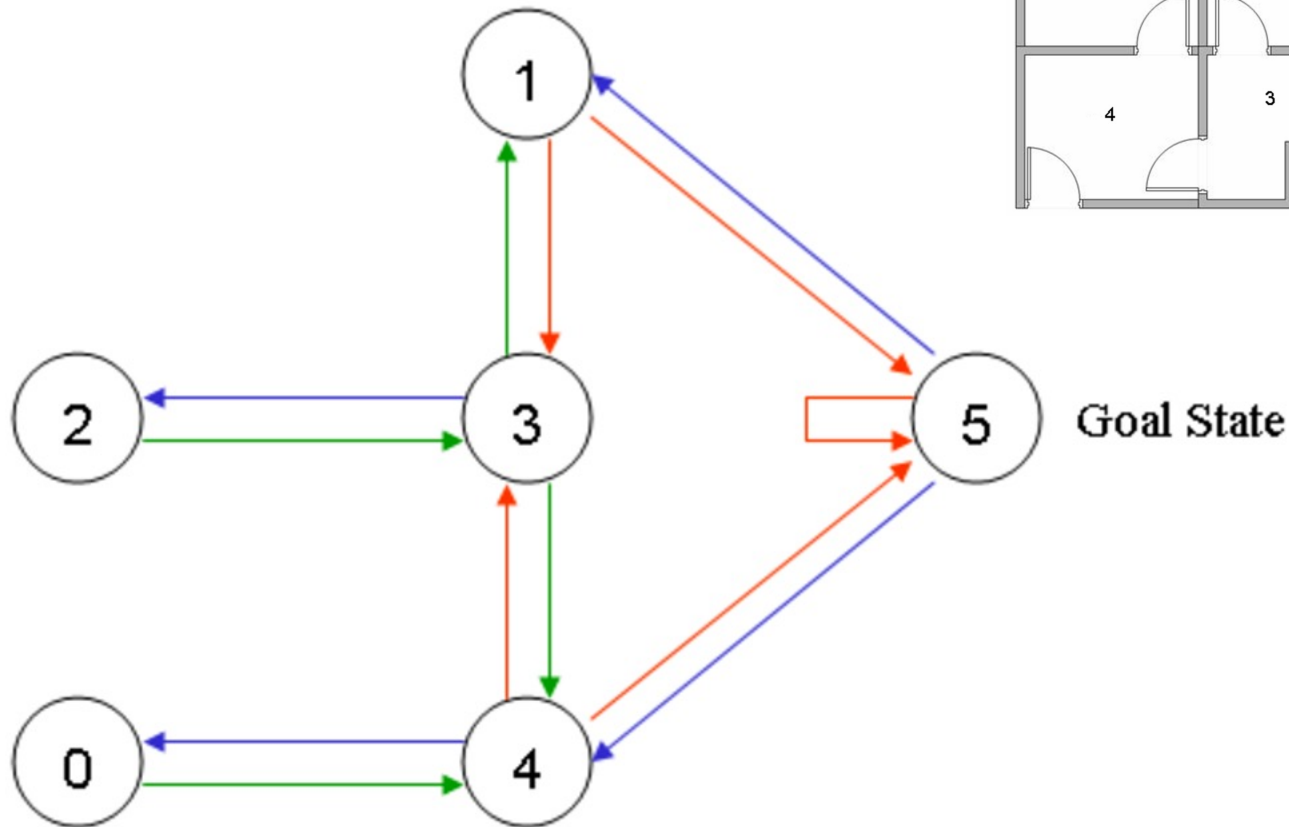
- On considère une maison avec 5 pièces + extérieur (pièce 5)



D'après John McCulloch

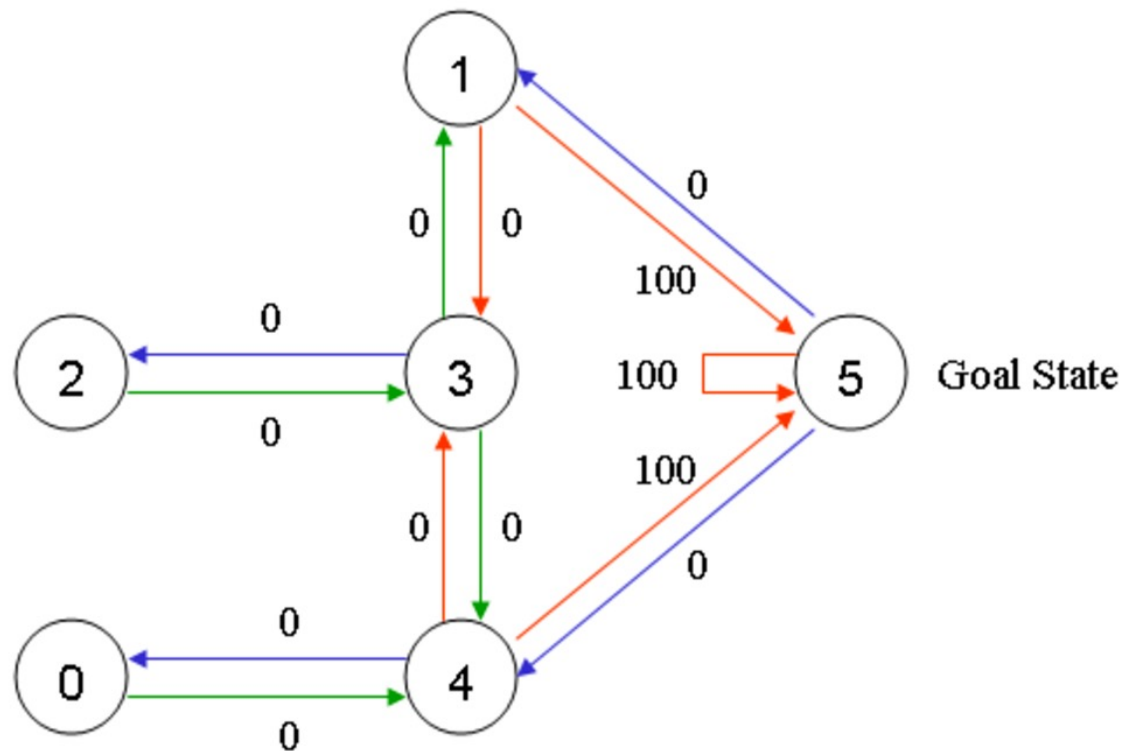
Q-learning : un exemple

- Représentation sous forme de graphe



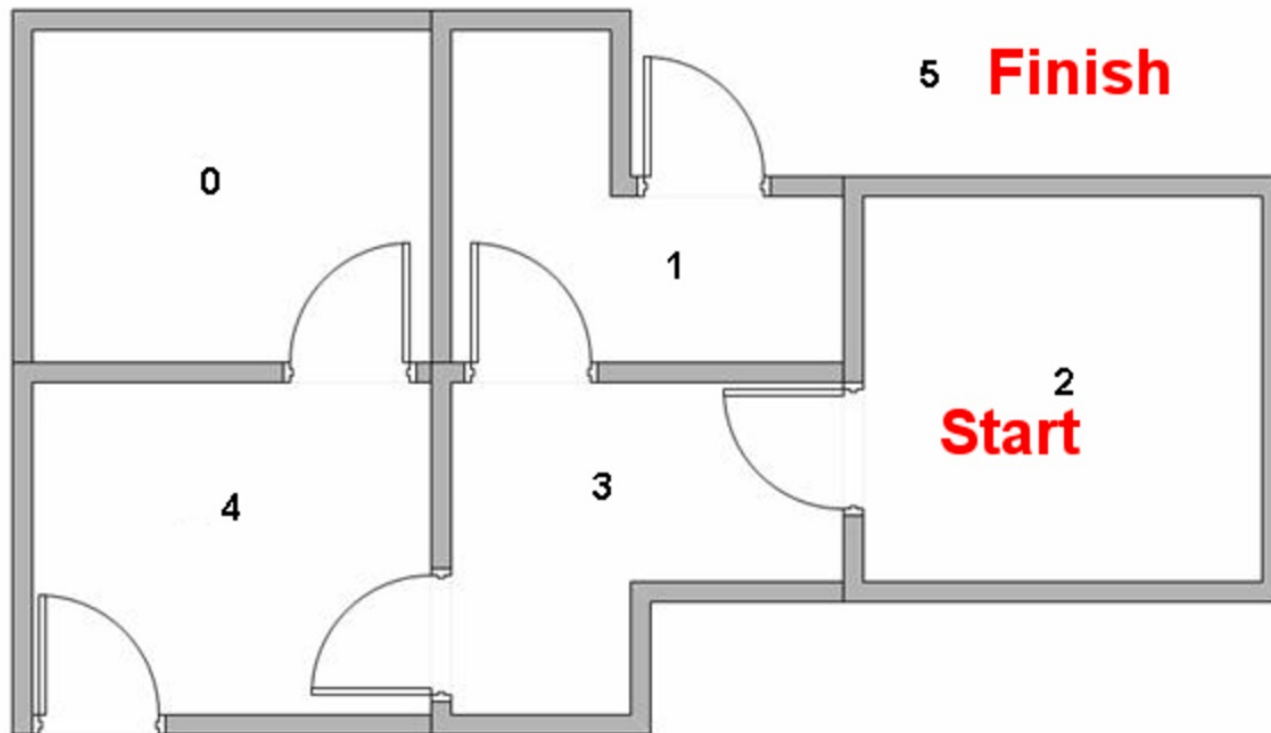
Q-learning : un exemple

- Pour atteindre le but (pièce 5), on associe une récompense immédiate à chaque porte



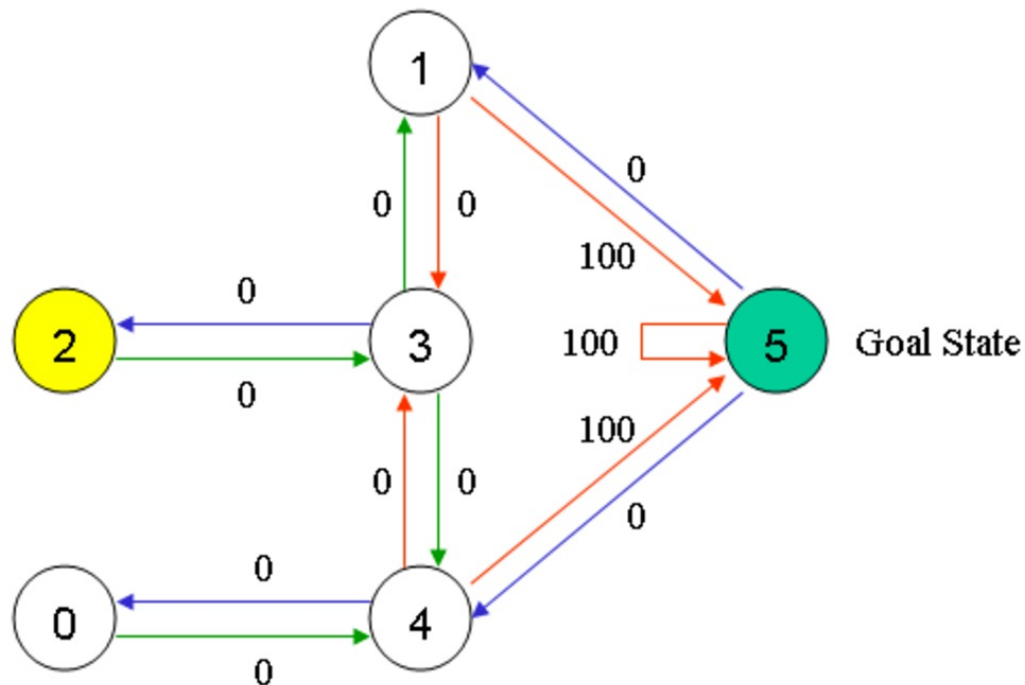
Q-learning : un exemple

- On suppose un agent dans la pièce 2 qui doit apprendre à atteindre l'extérieur



Q-learning : un exemple

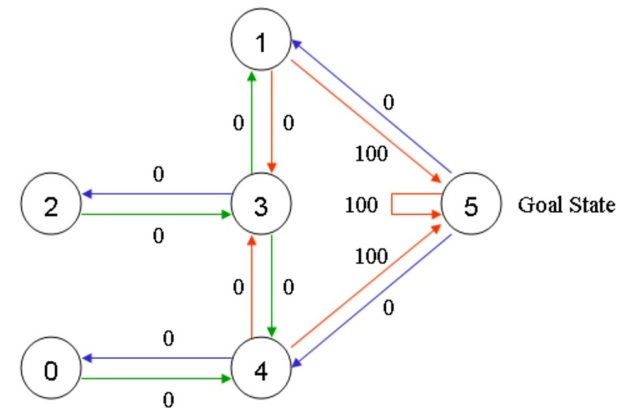
- Cela correspond au graphe suivant :



Q-learning : un exemple

- On considère la matrice de récompense R correspondante :

		Action					
	State	0	1	2	3	4	5
$R =$	0	-1	-1	-1	-1	0	-1
	1	-1	-1	-1	0	-1	100
	2	-1	-1	-1	0	-1	-1
	3	-1	0	0	-1	0	-1
	4	0	-1	-1	0	-1	100
	5	-1	0	-1	-1	0	100



Q-learning : un exemple

- On fixe α à 1 et γ à 0,8. Q est initialisée à 0. L'agent se trouve initialement dans la pièce 1

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Q-learning : un exemple

- Depuis la pièce 1, 2 actions possibles (ligne 1)
 - aller à la pièce 3 ou aller à la pièce 5
 - sélection aléatoire -> pièce 5

		Action					
State		0	1	2	3	4	5
$R=$	0	-1	-1	-1	-1	0	-1
	1	-1	-1	-1	0	-1	100
	2	-1	-1	-1	0	-1	-1
	3	-1	0	0	-1	0	-1
	4	0	-1	-1	0	-1	100
	5	-1	0	-1	-1	0	100

Q-learning : un exemple

- Depuis la pièce 5, 3 actions possibles (ligne 5)

- aller à la pièce 1 ou aller à la pièce 4 ou aller à la pièce 5

$$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q(\text{next state}, \text{all actions})]$$

$$Q(1, 5) = R(1, 5) + 0.8 * \text{Max}[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

$$R = \begin{matrix} & \begin{matrix} \text{Action} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{matrix}$$

Q-learning : un exemple

- L'état suivant (pièce 5) devient l'état courant
- Comme l'état 5 est l'état terminal, cet épisode s'arrête
- Un nouvel épisode commence. L'état 3 est sélectionné aléatoirement comme état initial

Q-learning : un exemple

- Depuis la pièce 3, 3 actions possibles (ligne 3)
 - aller à la pièce 1 ou aller à la pièce 2 ou aller à la pièce 4
 - sélection aléatoire → pièce 1

		Action					
State		0	1	2	3	4	5
$R=$	0	-1	-1	-1	-1	0	-1
	1	-1	-1	-1	0	-1	100
	2	-1	-1	-1	0	-1	-1
	3	-1	0	0	-1	0	-1
	4	0	-1	-1	0	-1	100
	5	-1	0	-1	-1	0	100

Q-learning : un exemple

- Depuis la pièce 1, 2 actions possibles (ligne 1)

- aller à la pièce 3 ou aller à la pièce 5

$$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q(\text{next state}, \text{all actions})]$$

$$Q(3, 1) = R(3, 1) + 0.8 * \text{Max}[Q(1, 3), Q(1, 5)] = 0 + 0.8 * \text{Max}(0, 100) = 80$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

$$R = \begin{matrix} & \begin{matrix} \text{Action} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{matrix}$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Q-learning : un exemple

- L'état suivant (pièce 1) devient l'état courant
- Depuis la pièce 1, 2 actions possibles (ligne 1)
 - aller à la pièce 3 ou aller à la pièce 5
 - sélection aléatoire -> pièce 5

		Action					
State		0	1	2	3	4	5
$R=$	0	-1	-1	-1	-1	0	-1
	1	-1	-1	-1	0	-1	100
	2	-1	-1	-1	0	-1	-1
	3	-1	0	0	-1	0	-1
	4	0	-1	-1	0	-1	100
	5	-1	0	-1	-1	0	100

Q-learning : un exemple

- Depuis la pièce 5, 3 actions possibles (ligne 5)

- aller à la pièce 1 ou aller à la pièce 4 ou aller à la pièce 5

$$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q(\text{next state}, \text{all actions})]$$

$$Q(1, 5) = R(1, 5) + 0.8 * \text{Max}[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

$$R = \begin{matrix} & \begin{matrix} \text{Action} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{matrix}$$

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Q-learning : un exemple

- L'état suivant (pièce 5) devient l'état courant
- Comme l'état 5 est l'état terminal, cet épisode s'arrête
- Un nouvel épisode commence.
- Après plusieurs épisodes, la matrice Q converge

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{matrix}$$

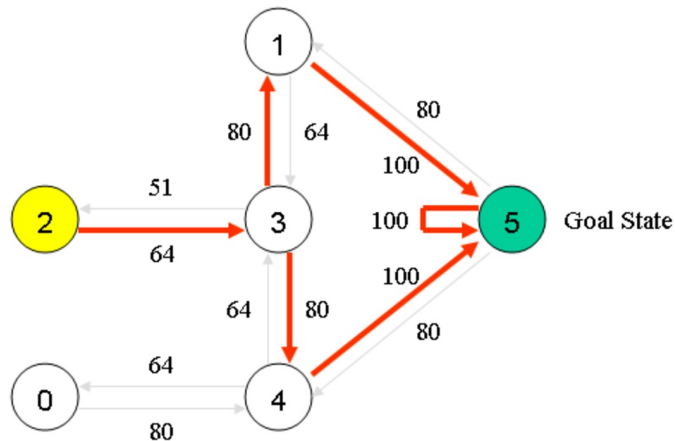
Q-learning : un exemple

- La matrice Q peut être normalisée en (pourcentage en divisant par la valeur maximale)

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{matrix}$$

Q-learning : un exemple

- L'agent a ainsi appris les chemins optimaux vers la sortie
- Par exemple, à partir de la pièce 2, la récompense maximale est obtenue pour aller à la pièce 3
- A partir de la pièce 3, 2 alternatives (valeurs maximales égales). On choisit arbitrairement 1
- A partir de 1, la récompense maximale est obtenue pour aller à la pièce 5



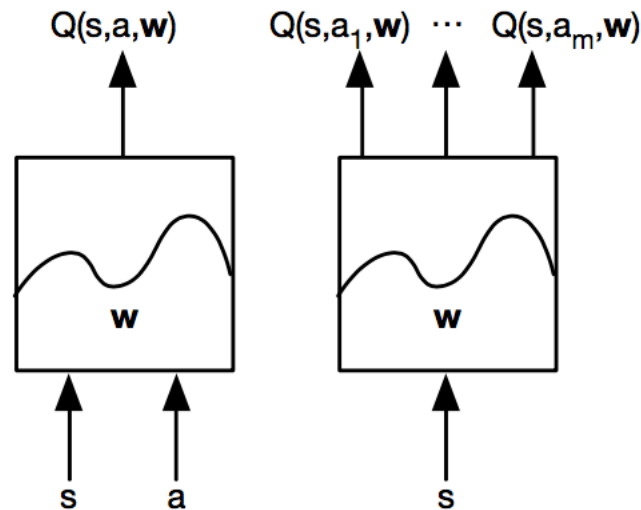
-> séquence : 2-3-1-5

Q-learning et réseaux de neurones

- Q-networks

- Représentation de la fonction action-valeur par un Q-network avec des poids w

$$Q(s, a, \mathbf{w}) \approx Q^*(s, a)$$



Q-networks

- Les valeurs optimales de la fonction Q se décomposent selon l'équation de Bellman

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

- La partie droite est considérée comme la cible

$$r + \gamma \max_{a'} Q(s', a', \mathbf{w})$$

- La fonction de perte MSE est minimisée par la descente du gradient

$$l = \left(r + \gamma \max_{a'} Q(s', a', \mathbf{w}) - Q(s, a, \mathbf{w}) \right)^2$$

Deep Q-networks (DQN)

- Méthode proposée par l'équipe DeepMind de Google (2013)
- Publication dans la revue Nature : « Human-level control through deep reinforcement learning », Nature, 2015.
- ➔ Utilisation d'un réseau convolutif profond (CNN) pour approximer l'optimal de la fonction action-valeur Q
- Expérimentations sur des jeux Atari

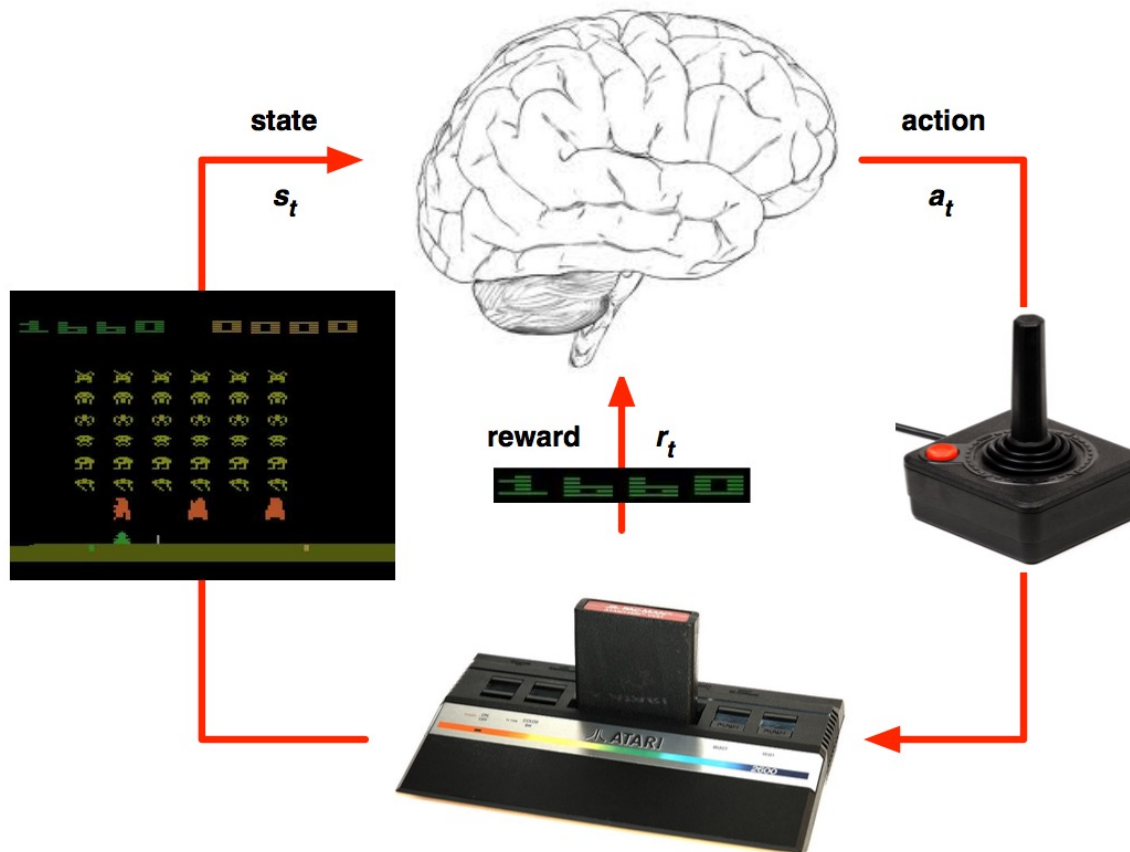


- Code disponible : <https://www.deepmind.com/open-source/dqn>

Deep Q-networks (DQN)

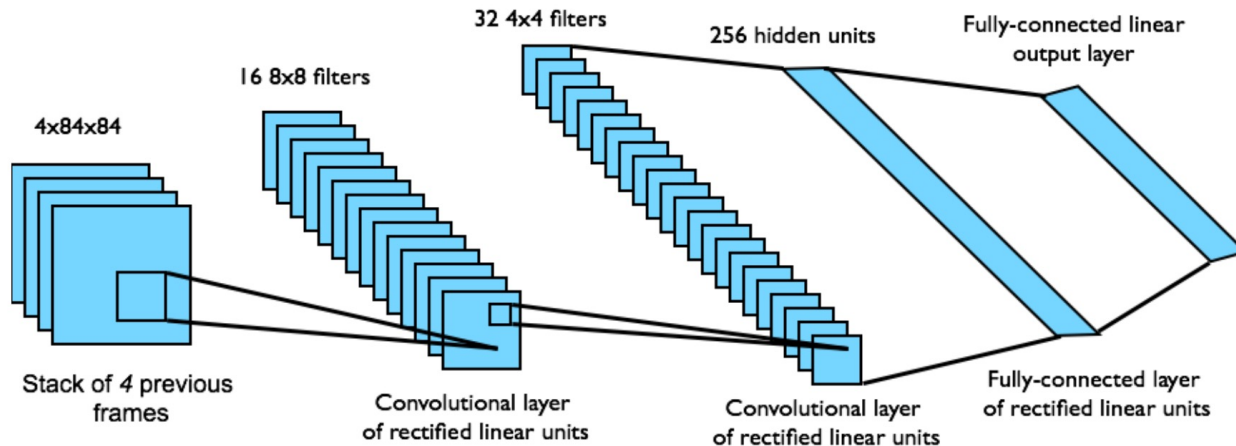
- Modèle de deep learning pour apprendre par renforcement les politiques de contrôle directement à partir données du capteur
- Le modèle est un CNN entraîné par une variante de l'algorithme de Q-learning

DQN pour Atari

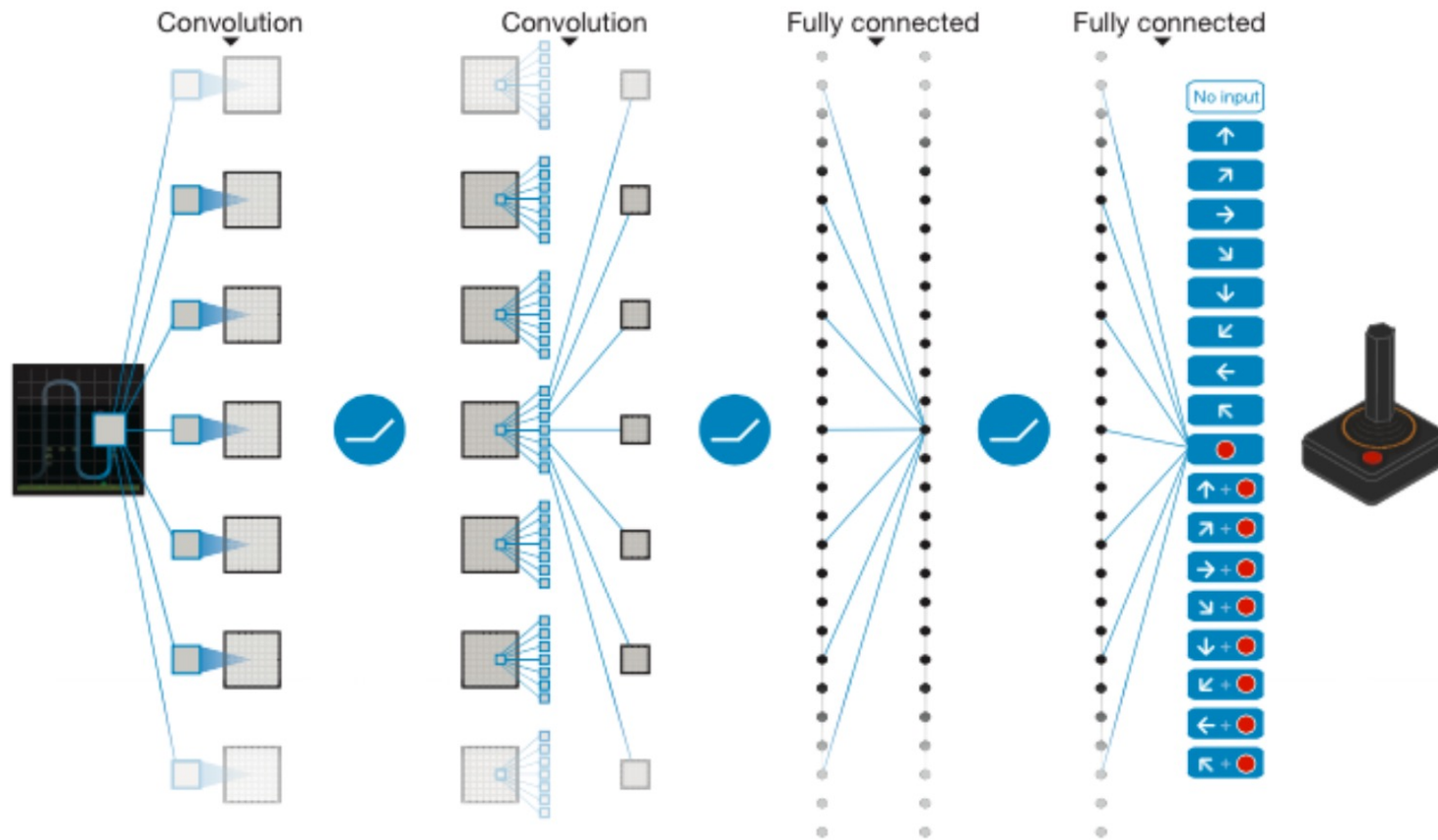


DQN pour Atari

- Apprentissage des valeurs de la fonction Q directement à partir des pixels
- Etat d'entrée s : pixels de 4 frames successives
- Sortie $Q(s,a)$ pour les 18 positions joystick/bouton
- Récompense : changement du score à chaque étape



DQN pour Atari



DQN pour Atari

- Principe général :

- On considère la fonction action-valeur optimale

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

- Le réseau peut être appris en minimisant une séquence de fonction de perte à chaque itération i

$$L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot)} \left[(y_i - Q(s, a; \theta_i))^2 \right]$$

- où $y_i = \mathbb{E}_{s' \sim \mathcal{E}} [r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a]$

DQN pour Atari

- Principe général :

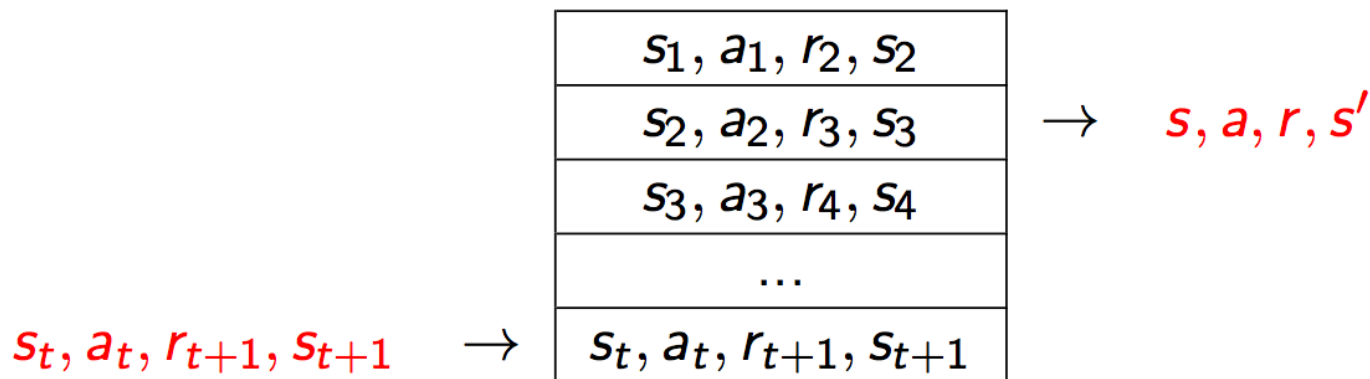
- La fonction de perte peut être dérivée partiellement par rapport aux poids pour obtenir le gradient

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right) \nabla_{\theta_i} Q(s, a; \theta_i) \right]$$

- Les poids sont alors mis à jour par descente du gradient

DQN pour Atari

- Pour éviter les corrélations, utilisation de « Experience Replay »
- Les expériences de l'agent sont stockées dans une mémoire et une sélection de ces expériences est utilisée à chaque étape pour la mise à jour de la fonction Q



DQN pour Atari

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

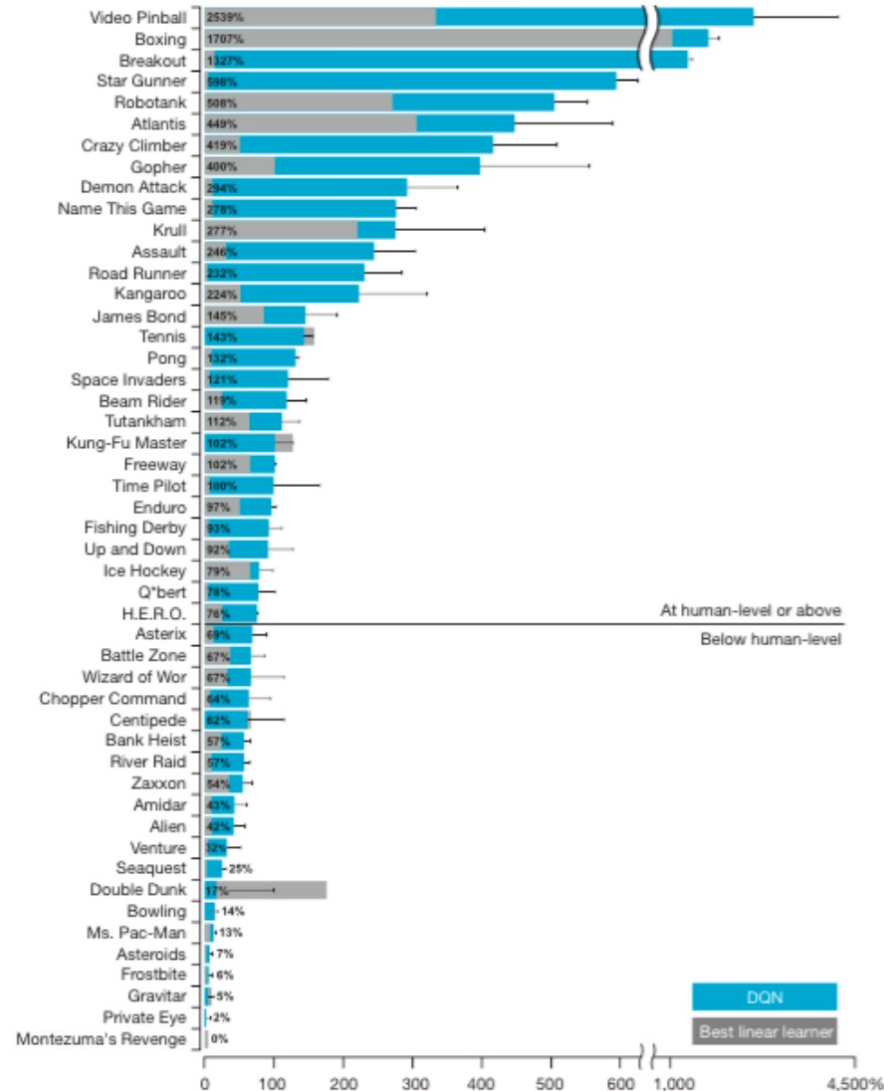
 Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3

end for

end for

DQN pour Atari

■ Résultats



DQN pour Atari

- Démo avec le jeu Breakout :



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

DQN pour Atari

- Démo avec le jeu Space Invaders :



- <https://www.youtube.com/watch?v=W2CAghUiofY>

Evolutions de DQN

- DeepMind AlphaGo (2015), AlphaZero (2017), MuZero (2019)





ÉCOLE
CENTRALE LYON

36, avenue Guy de Collongue 69130 Écully - France
+33 (0)4 72 18 60 00

www.ec-lyon.fr