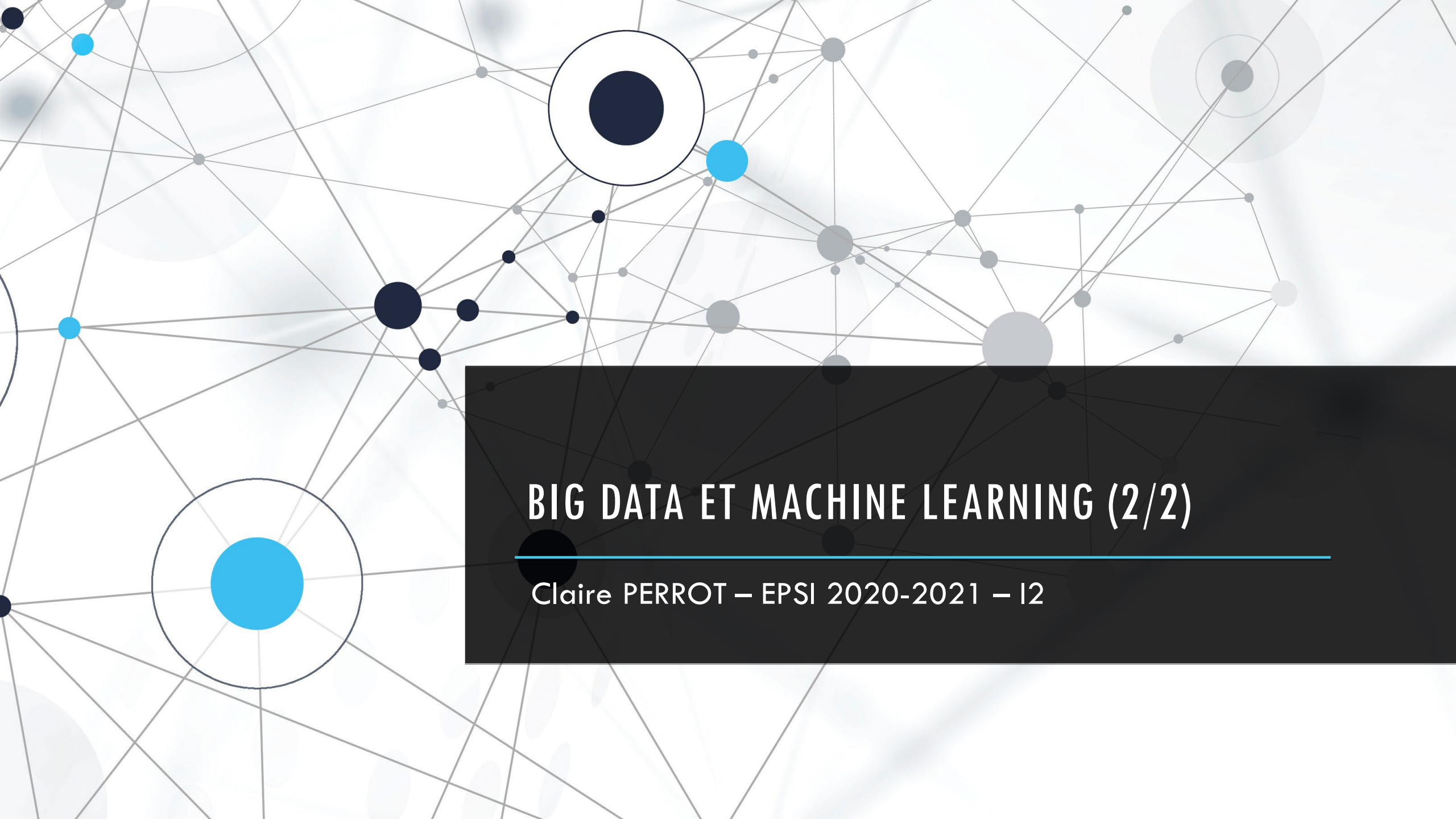


BIG DATA ET MACHINE LEARNING (2/2)

Claire PERROT – EPSI 2020-2021 – I2

APPRENTISSAGE PAR RENFORCEMENT

Agent dans un environnement -> apprend un comportement (= politique/policy)

Récompense (positive ou négative)

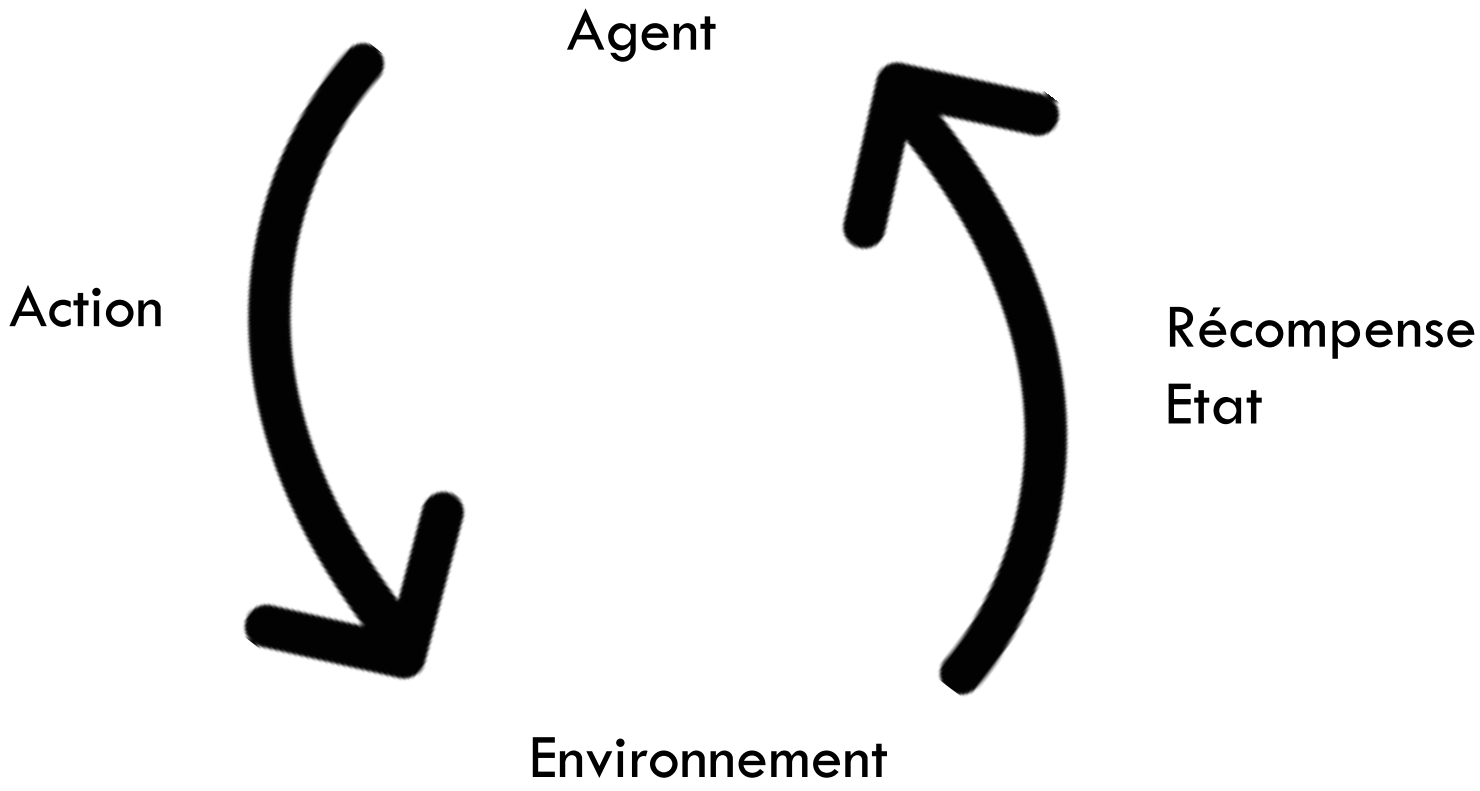
Inspirations : comportement humain / animal (conditionnement de Pavlov, apprentissage global de la sociabilisation)

But : **maximiser la récompense totale** (vs apprentissage supervisé = minimiser une fonction d'erreur)

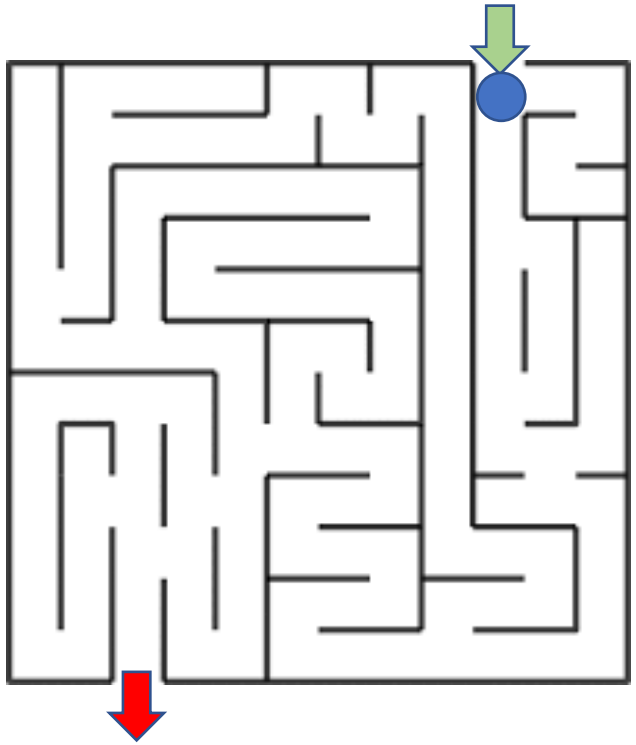
Etat + Action -> Récompense + nouvel état

APPRENTISSAGE PAR RENFORCEMENT – EXEMPLES

APPRENTISSAGE PAR RENFORCEMENT

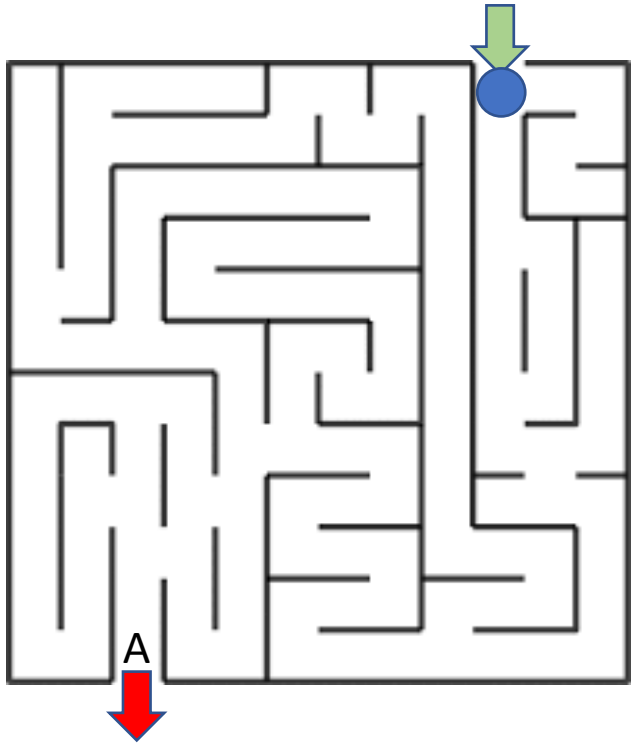


APPRENTISSAGE PAR RENFORCEMENT



Agent ?
Etats ?
Actions ?
Récompenses ?
Environnement ?
Pénalités ?

APPRENTISSAGE PAR RENFORCEMENT



Agent = ●

Etats = déterminés par la position dans le labyrinthe

Actions = déplacement gauche/droite/haut/bas

Récompenses = 10 points si placé en A + action « Bas »

Environnement = labyrinthe

Pénalités = -1 si déplacement vers un mur, -0.1 par déplacement (pour trouver le trajet le plus court)

(1,1) + ⇦ → -1 + (1,1)

(A) + ⇩ → 10 + « etat final »

(9,0) + ⇩ → -0.1 + (9,1)

APPRENTISSAGE PAR RENFORCEMENT

Définir les états (dont l'état initial et l'état final)

Définir les actions

Définir les récompenses pour chaque couple (état,action)

Définir les couples (état,action) non autorisés

Différents types d'environnements : Grille discrète avec des positions, environnement « naturel » (ex : robot qui apprend à marcher), espace de solution d'un jeu (ex: go, échecs…)

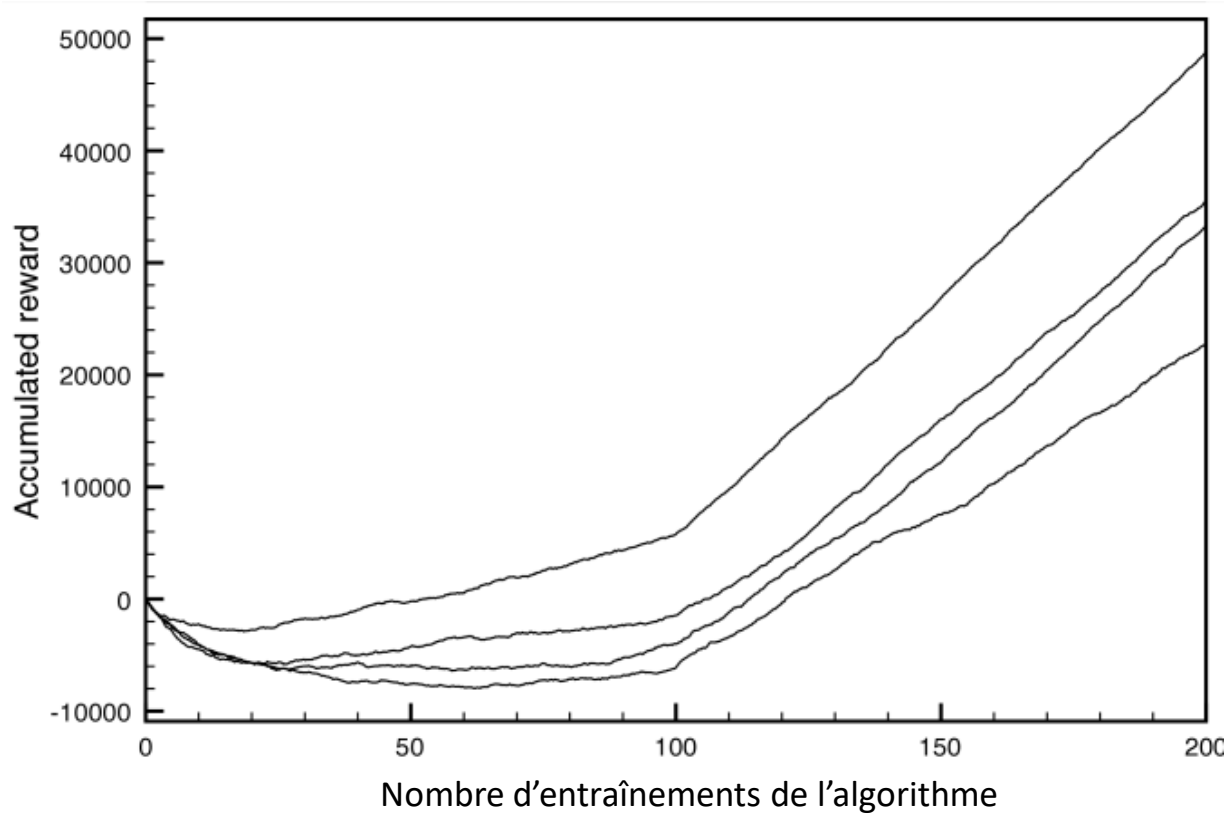
APPRENTISSAGE PAR RENFORCEMENT

Evaluation

Comparer les récompenses accumulées par différents algorithmes (ou le même avec des paramètres différents)

Limites

L'agent ne connait que l'environnement donné
Difficultés à généraliser d'un environnement à un autre
Coût en mémoire peut être important



Q-LEARNING

Q-table : Q(s, a) = récompense espérée dans le futur en étant dans l'état s et en faisant l'action a

On cherche à apprendre la valeur de Q(s,a) pour tous les couples (s,a)

Au départ : Q(s,a) = 0 partout

Q(s, a)

State (s) \ Action (a)	0	1	2	…
0	0.21772	0.09723	0.03119	0.04508
1	0.00168	0.01841	0.09021	0.03007
2	0.09021	0.03901	0.08233	0.06260
3	0.07745	0.06769	0.09672	0.09747
4	0.01272	0.05147	0.09482	0.08170
…	0.02254	0.02488	0.08215	0.03041

Q-LEARNING

Entrainement : l'agent se déplace dans l'environnement et met à jour la Q-table

Q-table se remplit -> c'est la politique de l'algorithme qui se dessine

Quand la table est remplie, on a les actions qui maximisent la récompense à chaque étape.

Q(s, a)

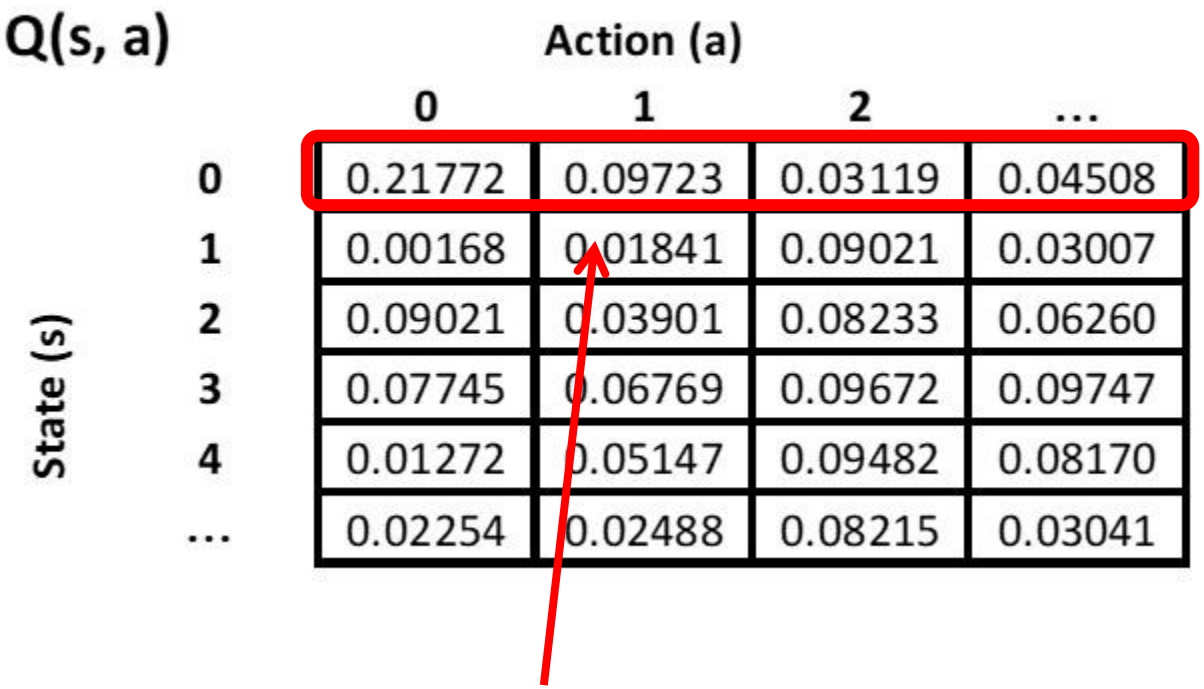
State (s) \ Action (a)	0	1	2	…
0	0.21772	0.09723	0.03119	0.04508
1	0.00168	0.01841	0.09021	0.03007
2	0.09021	0.03901	0.08233	0.06260
3	0.07745	0.06769	0.09672	0.09747
4	0.01272	0.05147	0.09482	0.08170
…	0.02254	0.02488	0.08215	0.03041

Q-LEARNING

Entrainement : l'agent se déplace dans l'environnement et met à jour la Q-table

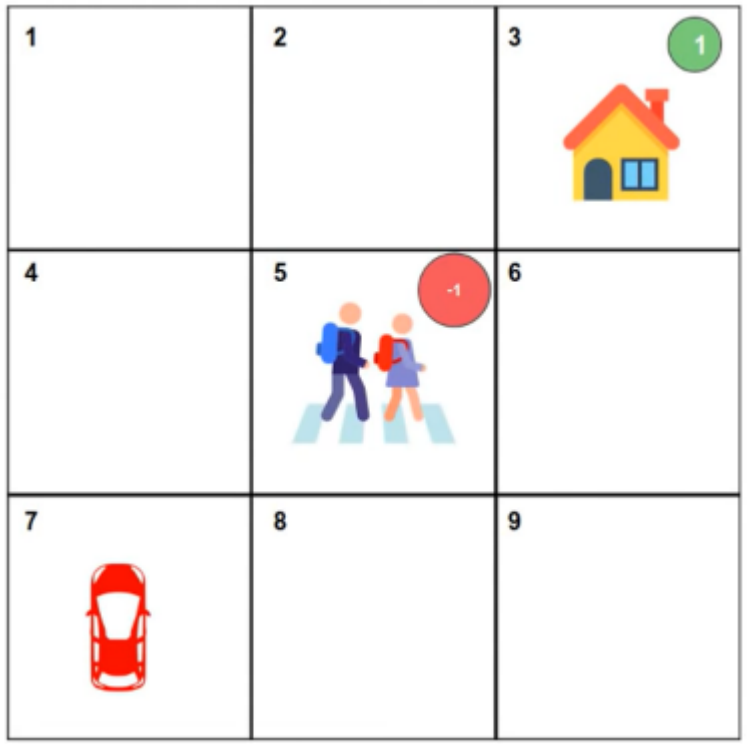
Q-table se remplit -> c'est la politique de l'algorithme qui se dessine

Quand la table est remplie, on a les actions qui maximisent la récompense à chaque étape.



La Q-table nous dit que l'action 0 est préférable dans l'état 0

Q-LEARNING : VOITURE



Etats : de 1 à 9 (calculés à partir de x et y), état 7 = initial

Récompense : 1 si arrive à l'état 3, -1 si arrive à l'état 5, 0 sinon

Actions disponibles : ↓ → ← ↑

Choix d'action :

- X % de chances de faire une action au hasard
- 1-X % de chances de faire la « meilleure » action (selon la Q-table)

> Programme *test_voiture.py* (sur MLB)

Q-LEARNING

Pseudo-code :

Initialiser la Q-table

Pour chaque épisode d'apprentissage :

- placer l'agent dans l'état de départ
- tant que l'état actuel n'est pas l'état final :

    - choisir une action a suivant la politique prévue
    - exécuter l'action a
    - observer la récompense obtenue et l'état d'arrivée s
    - mettre à jour la valeur (s,a) de la Q-table
    - mettre à jour l'état actuel à s

Q-LEARNING

Au final (quand la table est remplie de manière optimale),

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

En fait, $Q(s,a) = r + \gamma \max_{a'} Q(s',a')$ (γ = facteur d'actualisation)

Mise à jour des valeurs de la Q-table :

$Q(s,a) := (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$

> la valeur de Q(s,a) est mise à jour avec les nouvelles informations avec une proportion α

Q-LEARNING : PARAMÈTRES

Gamma γ (entre 0 et 1) = facteur d'actualisation = importance des récompenses futures (0 : l'agent ne regarde que la récompense directe). En général : $\gamma = 0.9$

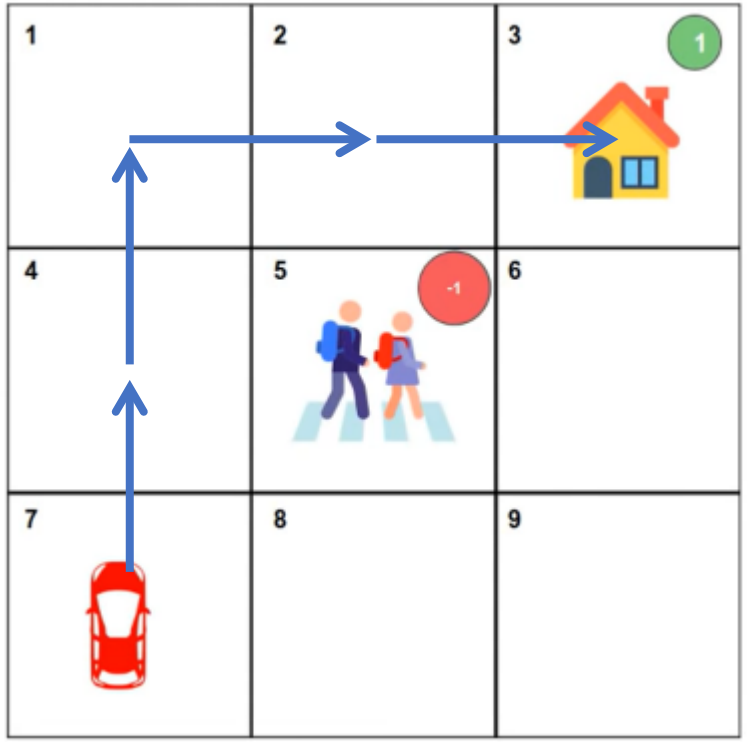
Alpha α (entre 0 et 1) = facteur d'apprentissage : à quelle point la nouvelle information calculée efface l'ancienne ($\alpha = 0$: pas d'apprentissage, $\alpha = 1$ la dernière info écrase toujours les précédentes). En général : $\alpha = 0.1$

Politique générale : exploration vs exploitation

- l'agent doit suivre les valeurs hautes de récompenses déjà apprises (exploitation)

- l'agent doit pouvoir découvrir de nouvelles voies (exploration)

Q-LEARNING : VOITURE



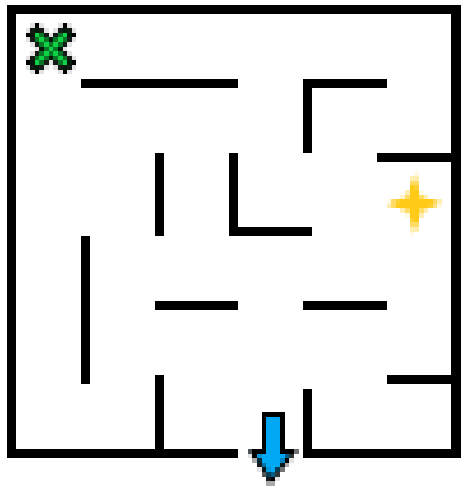
Q-table finale (un exemple) :

	up	down	left	right
1	'0.65'	'0.46'	'0.42'	'0.90'
2	'0.75'	'-0.37'	'0.72'	'1.00'
3	'0.00'	'0.00'	'0.00'	'0.00'
4	'0.81'	'0.42'	'0.54'	'-0.37'
5	'0.81'	'0.06'	'0.11'	'0.07'
6	'0.47'	'0.00'	'0.00'	'0.10'
7	'0.72'	'0.42'	'0.33'	'0.15'
8	'-0.17'	'0.04'	'0.42'	'0.00'
9	'0.04'	'0.00'	'0.00'	'0.00'

Exercice : modifiez le code de *test_voiture.py* pour :
- pénaliser les chemins longs
- interdire les mouvements sortant de la grille

Q-LEARNING : PROJET (EN BINÔME)

Programmer un algorithme de Q-learning qui apprend à sortir d'un **labyrinthe** tout en récupérant un trésor.



Version 2 : L'agent change de case de départ à chaque essai (mais le trésor et la case d'arrivée restent les mêmes !)

Version 3 : visualiser les chemins pris par l'agent à chaque essai

Rendu : un fichier .py + un fichier PDF décrivant rapidement votre projet et les choix faits

> sur MyLearningBox avant le 14 mars 23h59

POUR ALLER PLUS LOIN…

Vidéo sur le dilemme exploration/exploitation : ici (chaîne Fouloscopie)

Q-learning aussi utilisé en combinaison avec le deep learning

Apprentissage par renforcement hors-ligne (sans pouvoir interagir avec l'environnement, on a juste des enregistrements d'interactions précédentes)

Livre de référence : Reinforcement Learning, an Introduction (2e edition), Richard S. Sutton et Andrew Barto (en anglais), MIT Press

LEXIQUE FRANÇAIS – ANGLAIS

Apprentissage par renforcement – *reinforcement learning*
Politique – *policy*
Récompense – *reward*
Etat (initial / final) – *(starting/terminal) state*
Facteur d'actualisation (γ) – *discount factor*
Facteur d'apprentissage (α) – *learning rate*