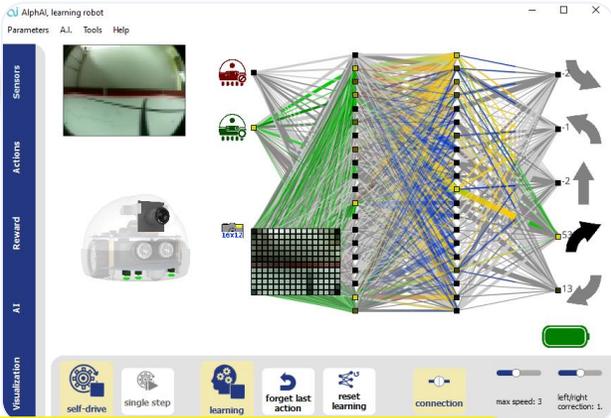




Entraîner une IA robot



Visualiser « Comment ça marche »



Activités ludiques & ambitieuses



De l'Élémentaire ...



... à l'Université



À l'international



En Entreprise

Cours L1: "Découverte de l'Intelligence Artificielle"



Cours d'option ouvert à tous les étudiants de L1

12 * 2h

- **Concepts fondamentaux** du Machine Learning
- **Découvrir puis coder** 3 algorithmes (KNN, Réseaux de neurones, Q-learning)

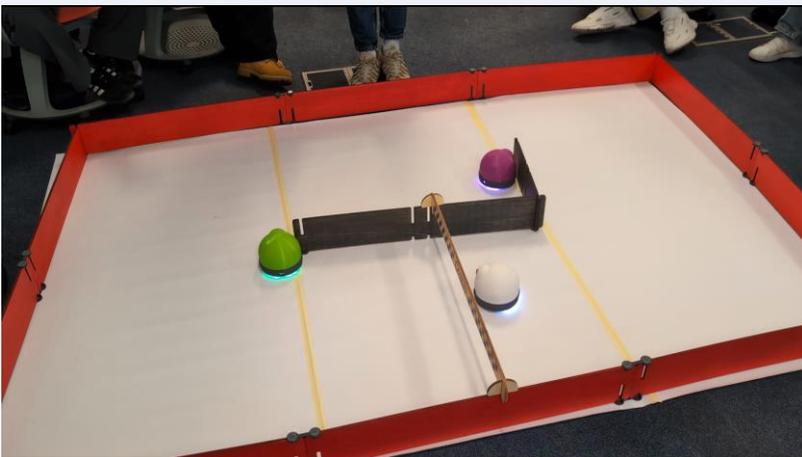
2022 : 1 classe de 28 étudiants

2023, 2024, 2025 : 4 classes (~90 étudiants) chaque année

TP1	Course de robots	<i>découvrir l'apprentissage supervisé en entraînant un robot dans une "course de robots" (on entraîne en pilotant le robot, c'est à dire en faisant l'acquisition du training set)</i>
TP2	Programmation du robot en python	<i>programmation très basique pour apprendre les bases de python</i>
TP3	Perfectionnement en python	<i>programmation de la décision du robot en fonction de la couleur d'un des pixels de la caméra</i>
TP4	Algorithme des K plus proches voisins	<i>découverte de l'algorithme KNN dans notre interface graphique</i>
TP5	Programmation des K plus proches voisins	<i>programmation de KNN en python (le code de l'élève est ensuite chargé dans notre logiciel, qui permet de l'utiliser pour entraîner le robot)</i>
TP6	TP évalué	
TP7	Détection d'intrus	<i>Edition manuelle dans le logiciel des poids d'un mini-réseau de neurones devant actionner une alarme lorsque une distance mesurée est trop faible ou trop forte (la fonction modélisée étant non linéaire, on découvre la nécessité d'utiliser une couche d'au moins 2 neurones intermédiaires)</i>
TP8	Réseaux de neurones avec scikit-learn	<i>Reproduction de l'activité avec ANN construit avec scikit-learn dans le code élève</i>
TP9	Q-learning avec AlphaI	<i>Découverte du Q-learning dans le logiciel (le robot apprend en apprentissage par renforcement à avancer par défaut, mais tourner en présence d'obstacle ; table Q-learning 2 états obstacle / pas d'obstacle x 5 actions)</i>
TP10	Programmation Q-learning	<i>Programmation de l'algo dans un cas particulièrement simplifié</i>
TP11	Récapitulatif	<i>Cours avec slides récapitulatives sur les notions apprises et slides</i>
TP12	TP évalué	
Examen		

① Découvrir par la manipulation

Séance 1 : Course de robots autonomes !!



(1) Acquisition de données

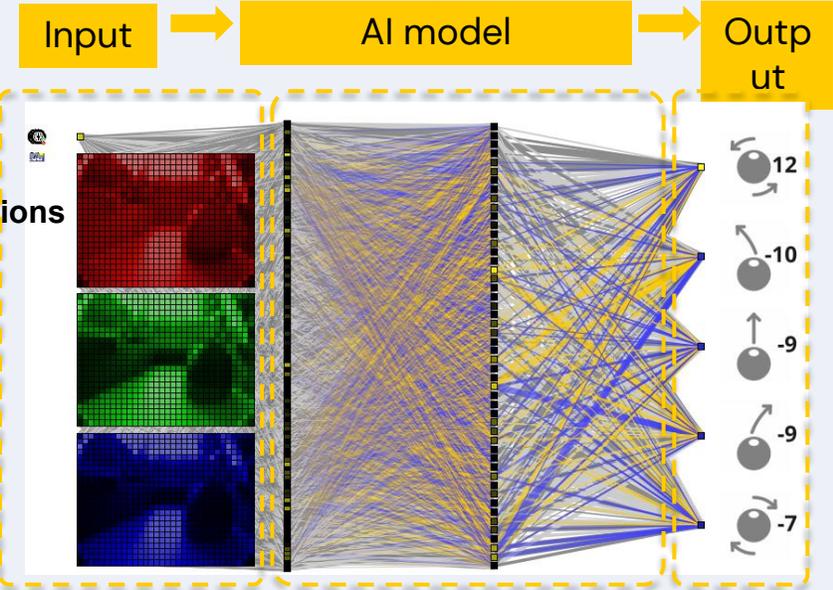


Data

(2) Entraînement du modèle

Learning

(3) Test et Améliorations

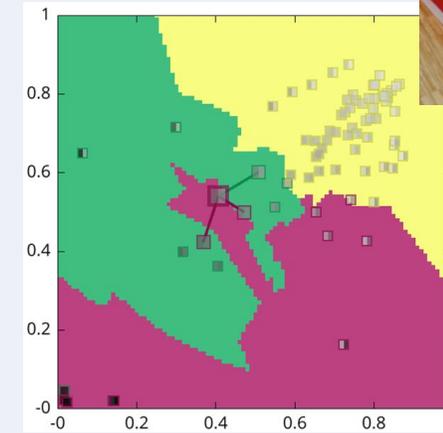


(4) Exploitation

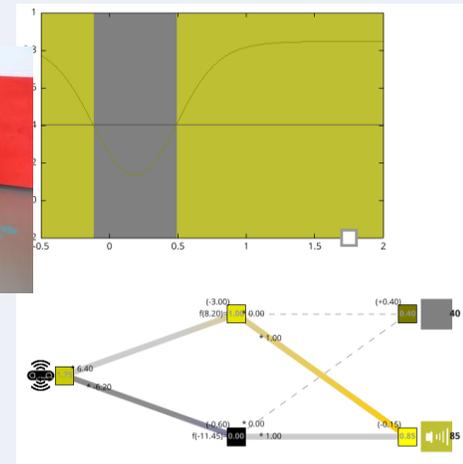
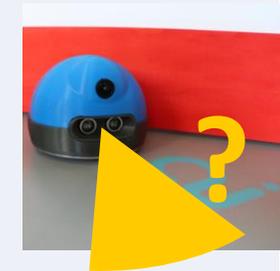
② Visualiser & Comprendre



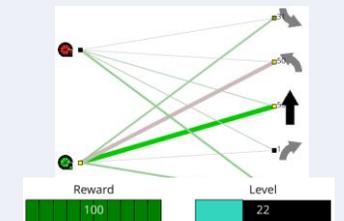
K plus proches voisins



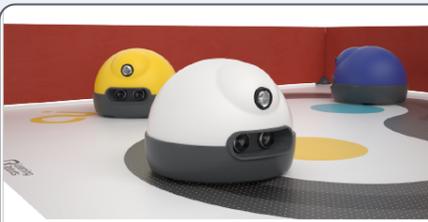
Réseaux de neurones



Apprentissage par Renforcement : Q-learning



③ Programmation Python des algorithmes



alpha*i*
L'ÉDUCATION À L'INTELLIGENCE ARTIFICIELLE

Programmation de l'algorithme Q-learning

effectue l'action a à partir de l'état s .

Pour cela, l'algorithme va mettre à jour une valeur de la table après chaque action à l'aide des formules suivantes :

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

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha target$$

où :

- $Q(s, a)$ représente la nouvelle valeur de la cellule.
- α est un paramètre compris entre 0 et 1, appelé vitesse d'apprentissage.
- γ est un second paramètre compris entre 0 et 1, appelé facteur d'actualisation.
- $\max_{a'} Q(s', a')$ est la récompense maximale que l'on peut obtenir à partir de l'état s' , selon les évaluations actuelles de la Q-table.

L'objectif de ce TP est de programmer en python cet algorithme Q-learning et de l'appliquer à l'apprentissage de l'environnement d'obstacle des robots Alpha*i*.

1.2 CONFIGURATION DU LOGICIEL

- Charger la configuration Apprentissage par renforcement - Elipé en Menu

1 Q-LEARNING

1.1 INTRODUCTION

L'algorithme de Q-learning est un algorithme d'apprentissage par renforcement. Cela signifie que l'IA va explorer en autonome différentes stratégies, et recevoir des récompenses ou pénalités selon des modalités déterminées à l'avance. L'objectif de l'algorithme d'apprentissage est alors d'exploiter les données ainsi générées afin d'améliorer les prises de décision de l'IA, dans le but de maximiser les récompenses reçues à plus ou moins long terme.

Voici un résumé du vocabulaire et des notations du renforcement :

- Le robot est un agent apprenant.
- L'état dans lequel se trouve l'agent est une donnée, comme sa position, sa batterie, etc. Dans le cas de Alpha*i*, on s'appuie sur les données fournies par les capteurs actifs du robot (à l'exception de ceux qui sont désactivés).
- L'action choisie par l'agent est notée a . Il s'agit d'un des mouvements disponibles pour le robot.
- Après avoir effectué une action, l'agent reçoit une valeur numérique positive ou négative.
- Le nouvel état dans lequel se trouve l'agent est noté s' .

L'algorithme d'apprentissage par renforcement apprend à des situations où le nombre d'états possibles est infini et de taille raisonnable. Les valeurs afin de prendre des décisions. Ce tableau qui donne son nom à l'algorithme. Chaque ligne s et chaque colonne à une action a . La valeur de $Q(s, a)$ est donc notée $Q(s, a)$. Voir figure 1.

État	Action		
	a_1	a_2	a_3
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$Q(s_1, a_3)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$Q(s_2, a_3)$
s_n	$Q(s_n, a_1)$	$Q(s_n, a_2)$	$Q(s_n, a_3)$

FIGURE 1 - Représentation de la table de Q-learning. L'objectif de l'apprentissage va être de faire en sorte que le robot apprenne à prendre la meilleure décision possible à l'aide de la table de Q-learning.

1.4 FONCTION TAKE_DECISION

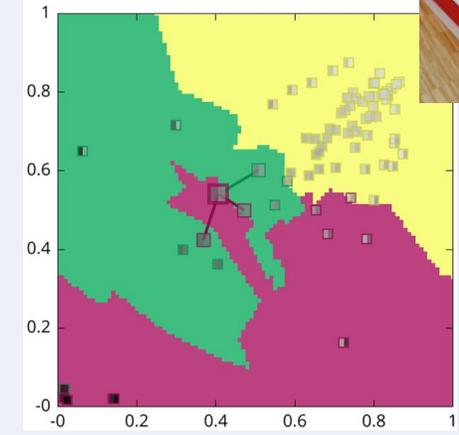
- Définir une fonction auxiliaire argmax qui renvoie l'indice du maximum de cette liste.
- Pour prendre une décision, l'algorithme doit connaître l'état de l'environnement. On peut obtenir cet état à l'aide de la fonction `get_state`.
- Dans le logiciel Alpha*i*, chaque robot possède une table de Q-learning. Cette table contient des valeurs aléatoires entre -1 et 1.

```

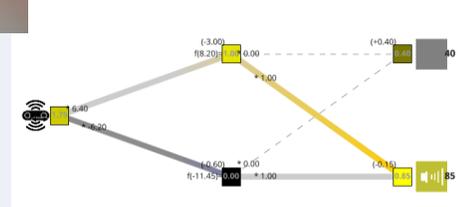
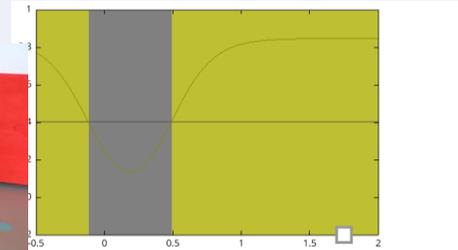
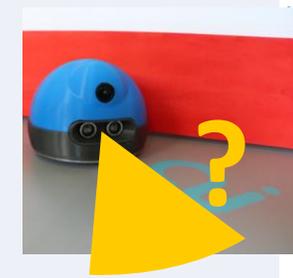
1 from random import uniform
2
3 def print_table(table):
4     for line in table:
5         for value in line:
6             print(round(value, 2), end=' ')
7         print() # Nouvelle ligne
8
9 def init(n_state: int, n_action: int):
10    global q_table
11    q_table = [[uniform(-1,1) for j in range(n_action)] for i in range(n_state)]
12    print_table(q_table)
13
14 def init(n_state: int, n_action: int):
15    global q_table
16    q_table = []
17    for i in range(n_state):
18        line = []
19        for j in range(n_action):
20            line.append(uniform(-1,1))
21        q_table.append(line)
22    print_table(q_table)
23
24 def learn(old_state: int, action: int, reward: float, new_state: int):
25    alpha = 0.2
26    gamma = 0.8
27    target = (1 - gamma) * reward + gamma * max(q_table[new_state])
28    q_table[old_state][action] = (1 - alpha) * q_table[old_state][action] + alpha * target
29    print_table(q_table)
30
31 def take_decision(state: int) -> int:
32    """
33    Determine the appropriate action knowing the current state (or perform
34    exploration!).
35    @param state: an integer coding for the current state.
36    @return: the index of the chosen action (starting from 0 at the top).
37    """
38    action = 0
39    return action
40
41 def get_reward(sensors: list, speed: float) -> float:
42    on_line = sum(sensors) > 0 # Booléen indiquant si le robot est sur la ligne ou non.
43    if on_line:
44        # Le capteur central rapporte plus de points, les capteurs latéraux moins.
45        reward = sensors[2] + 0.6 * (sensors[1] + sensors[3]) + 0.3 * (sensors[0] + sensors[4])
46        # La récompense est proportionnelle à la vitesse.
47        reward *= speed
48    else:
49        # La récompense est négative, mais encourage la marche arrière.
50        reward = -speed - max_speed
51    return reward

```

K plus proches voisins



Réseaux de neurones



Apprentissage par Renforcement : Q-learning

