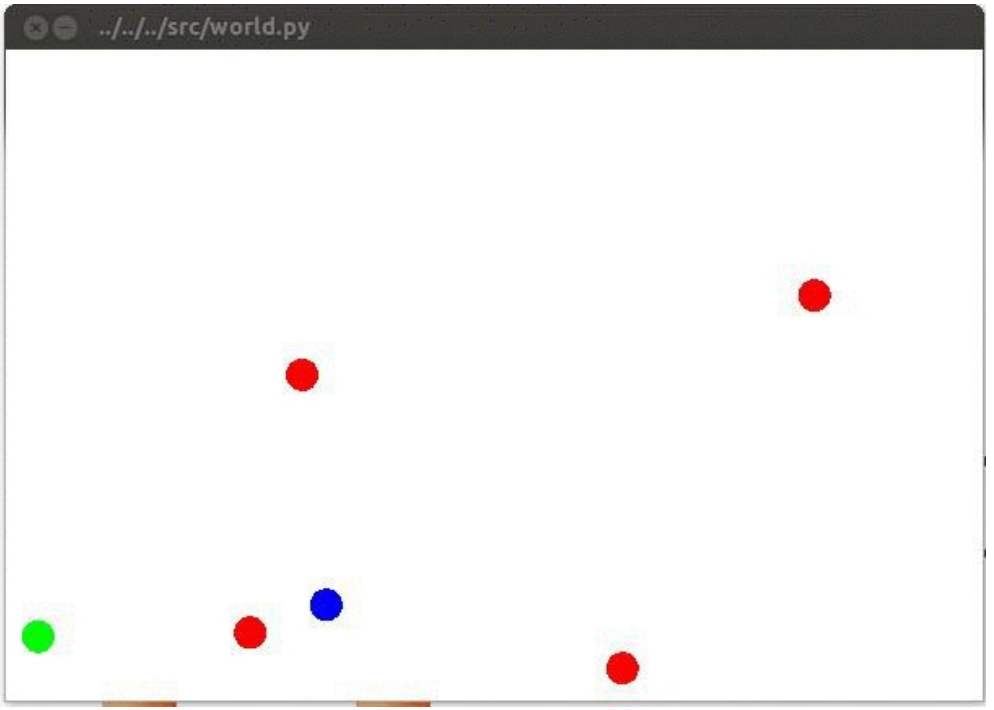
Étude sur l’apprentissage par renforcement dans les jeux

Projet de Bachelor 2018 - hepia

Étudiant : Federico Pfeiffer Professeur : Guido Bologna



h e p i a
Haute école du paysage, d’ingénierie
et d’architecture de Genève

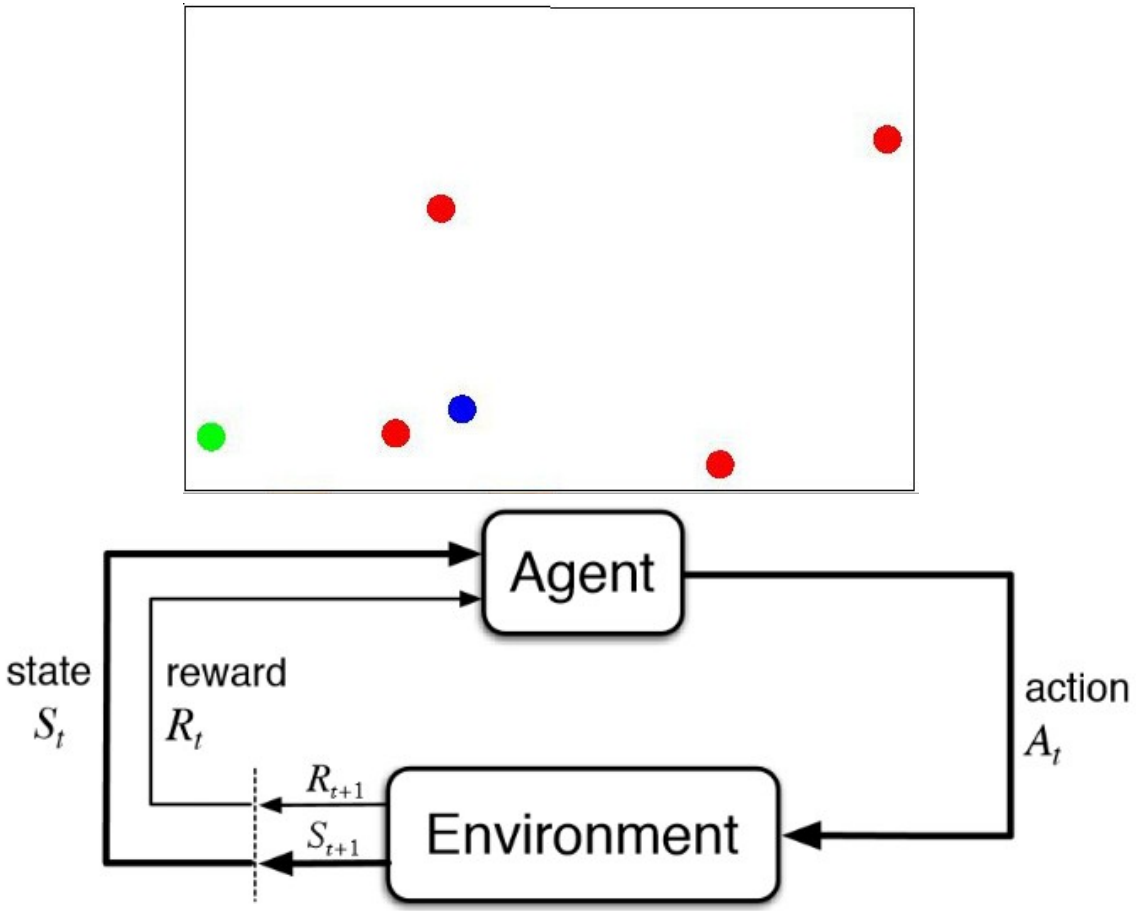
Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Cahier des charges

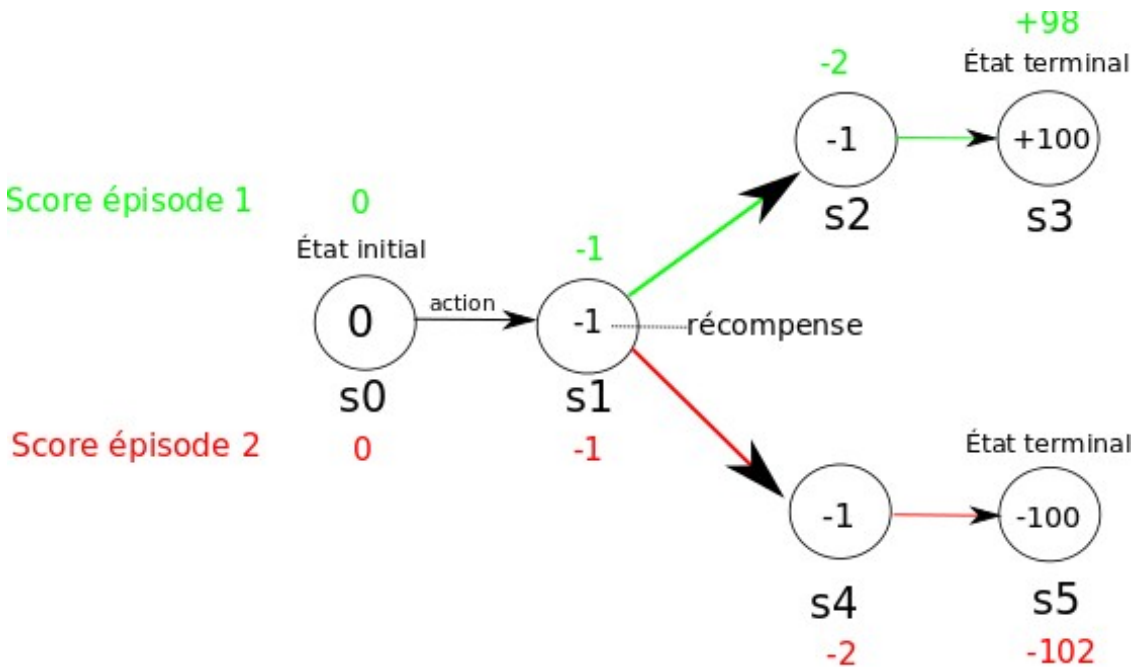
- Comprendre les modèles de réseaux de neurones artificiels standards
- Comprendre les modèles de réseaux de neurones artificiels profonds
- Comprendre les concepts de l'apprentissage par renforcement
- Construire un jeu avec un agent qui apprend par renforcement
- Analyse des résultats
- Rédaction du rapport

Apprentissage par renforcement

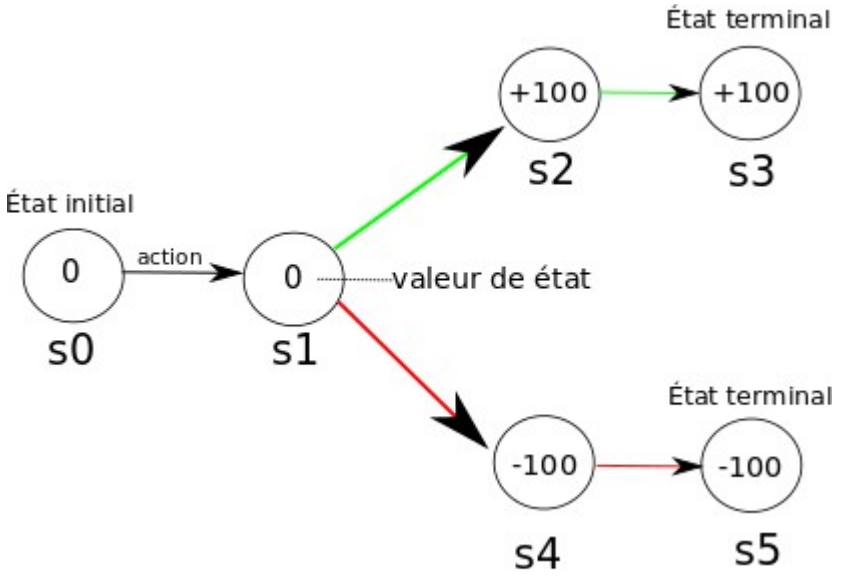
Agent ↔ Environnement



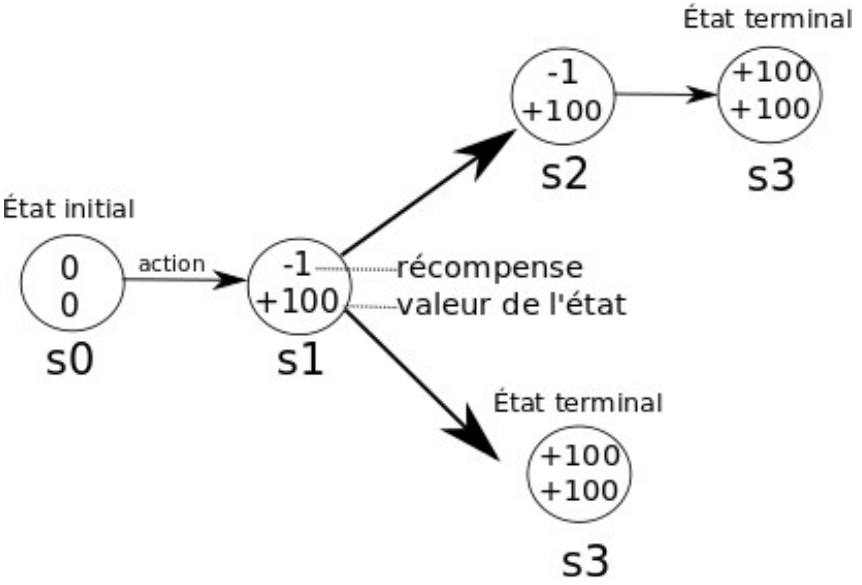
Choix d'une action: valeur d'un état (1)



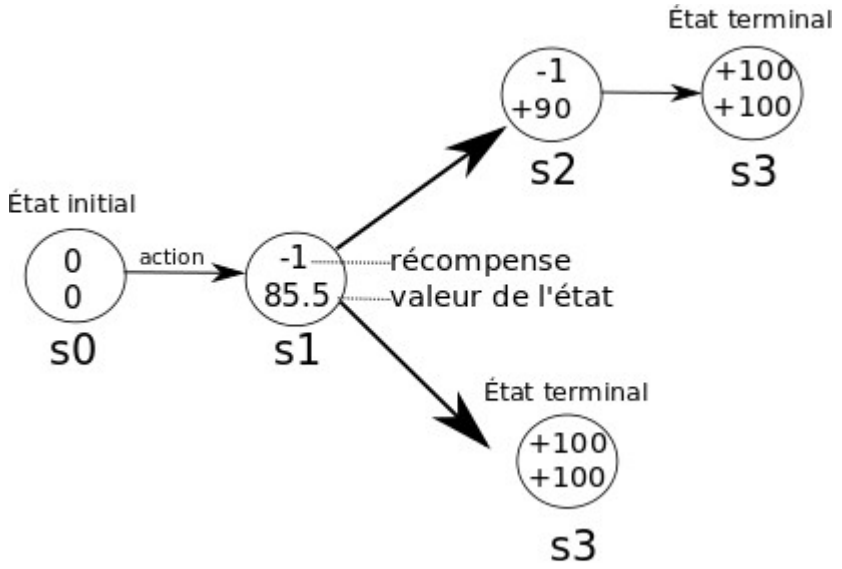
Choix d'une action: valeur d'un état (2)



Choix d'une action: valeur d'un état (3)

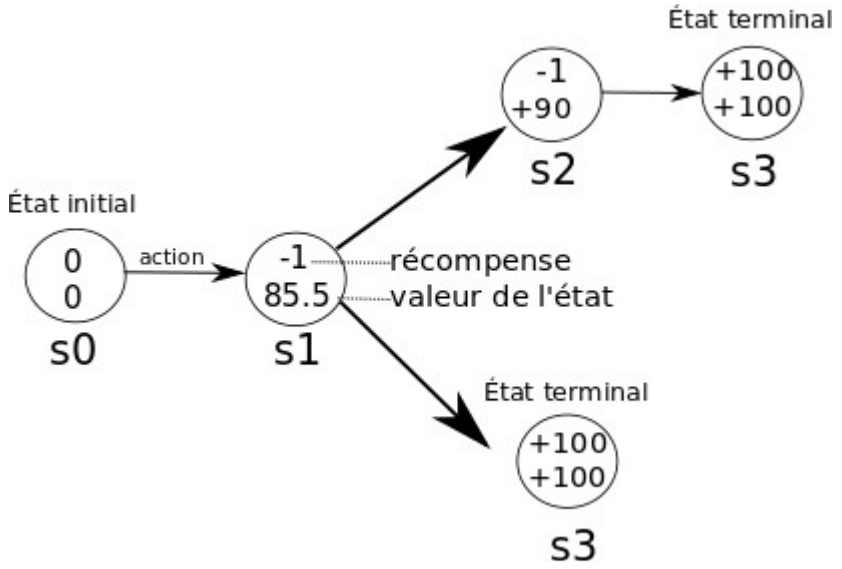


Choix d'une action: valeur d'un état (3)
discount factor (γ)



$$V(s_t) = \gamma V(s_{t+1}) \text{ avec } \gamma \in [0, 1]$$

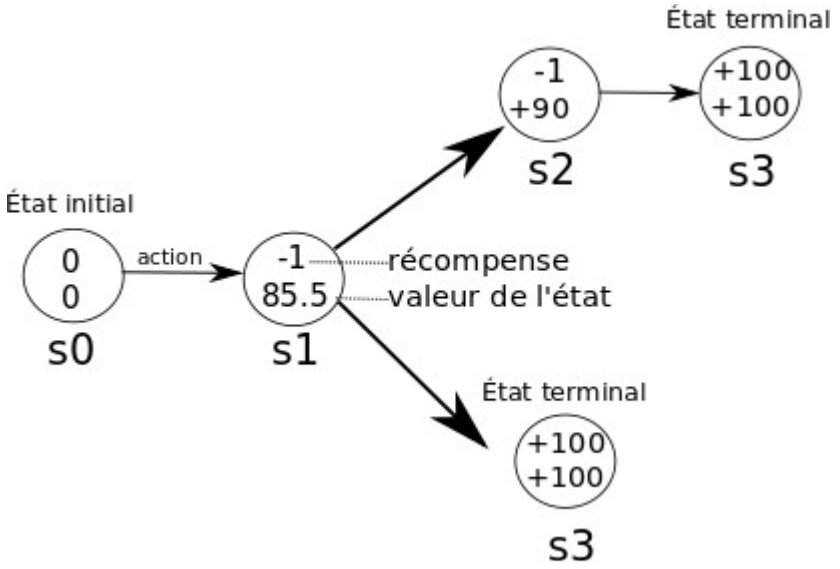
Choix d'une action: explore vs exploit proba aléatoire (ε)



$$V(s_t) = \gamma V(s_{t+1}) \text{ avec } \gamma \in [0, 1]$$

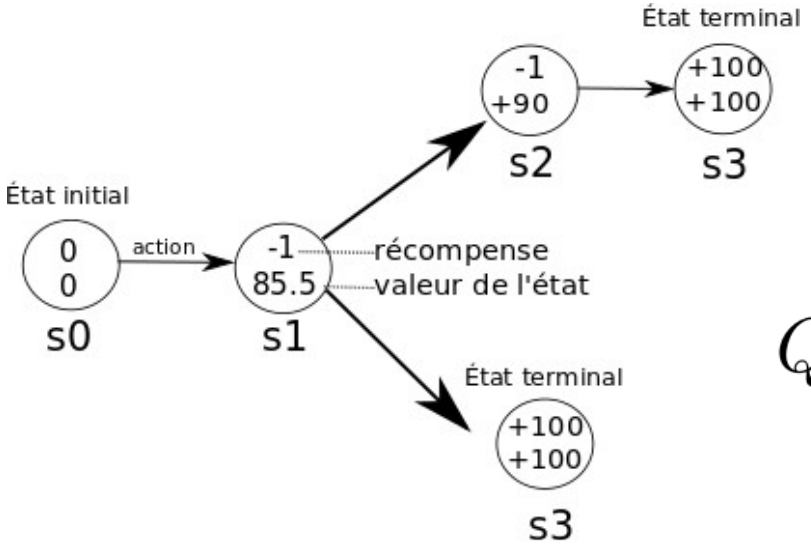
action aléatoire avec p = ε ∈ [0,1]

Valeur d'un état: équation de Bellman (simplifiée)



$$V_\pi(s) = \sum_a \pi(a|s) \left\{ r + \gamma V_\pi(s') \right\}$$

Valeur d'un état: Q-Learning



$$V_\pi(s) = \sum_a \pi(a|s) \left\{ r + \gamma V_\pi(s') \right\}$$

$$Q_\pi(s,a) = \left\{ r + \gamma Q_\pi(s',a') \right\}$$

$$Q(s,a) = Q(s,a) + \alpha(\left\{ r + \gamma Q(s',a') \right\} - Q(s,a))$$

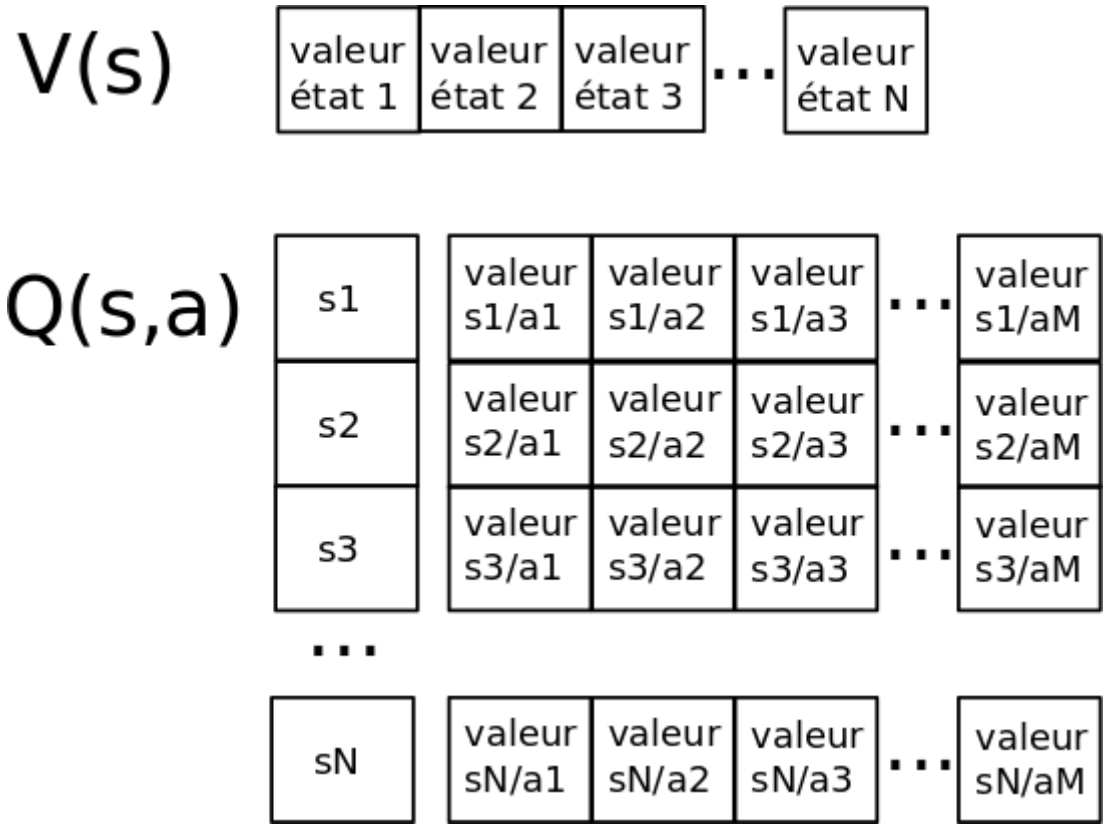
Apprentissage par renforcement

Q-Learning : pseudo-code

```
# phase d'apprentissage
world = new World()
agent = new Agent()
max_episodes = 100000
gamma = 0.9
epsilon = 0.1
alpha = 0.1

for i in range(max_episodes) :
  state = world.reset()
  while not world.game_over() :
    action = agent.choose_action(state, epsilon)
    reward, next_state = world.do_action(state, action)
    a2 = agent.choose_action(next_state, epsilon)

    if world.game_over() :
      agent.Q[state][action] += alpha*(reward)
    else :
      agent.Q[state][action] += alpha*(reward+gamma*agent.Q[next_state][a2])

    state = next_state


  # phase une fois l'apprentissage effectué
  state = world.reset()
  while not world.game_over() :
    action = agent.choose_action(state, epsilon)
    state, _ = world.do_action(action)
```

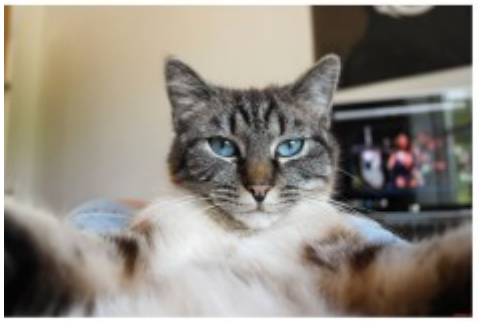
$$Q(s,a) = Q(s,a) + \alpha(\{r + \gamma Q(s',a')\} - Q(s,a))$$

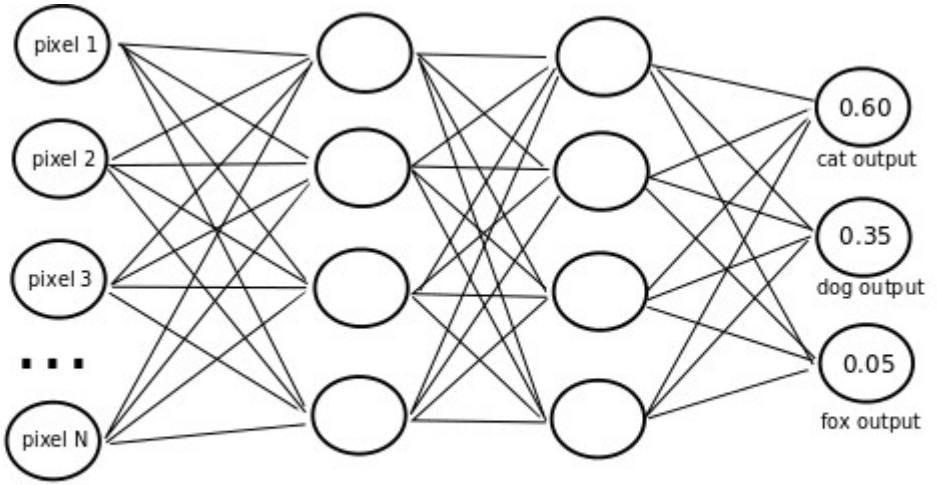
Q-Learning
Nécessité de réseaux neuronaux



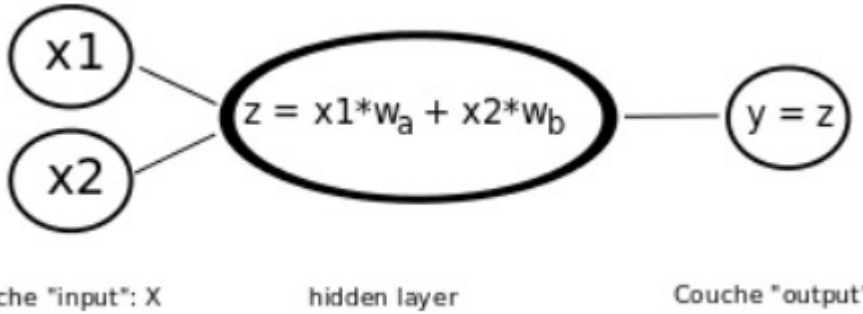
Réseaux Neuronaux

Utilité : estimer la valeur d'un état *Q(s,a)*

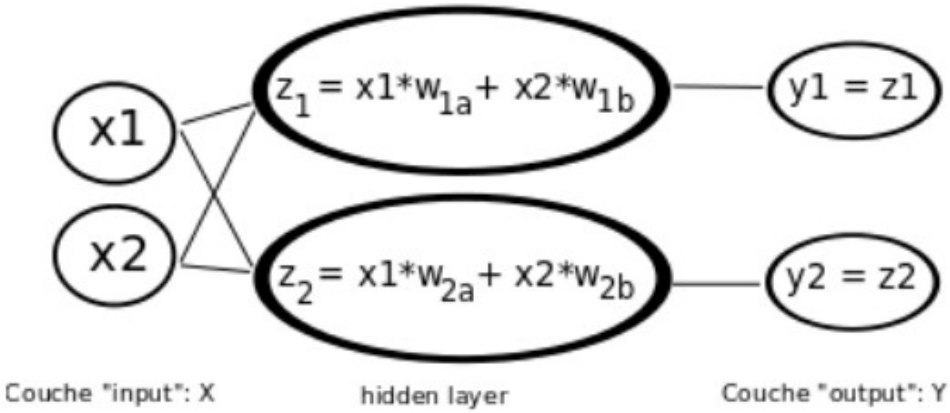




Réseau de neurones « standard »
Prédiction

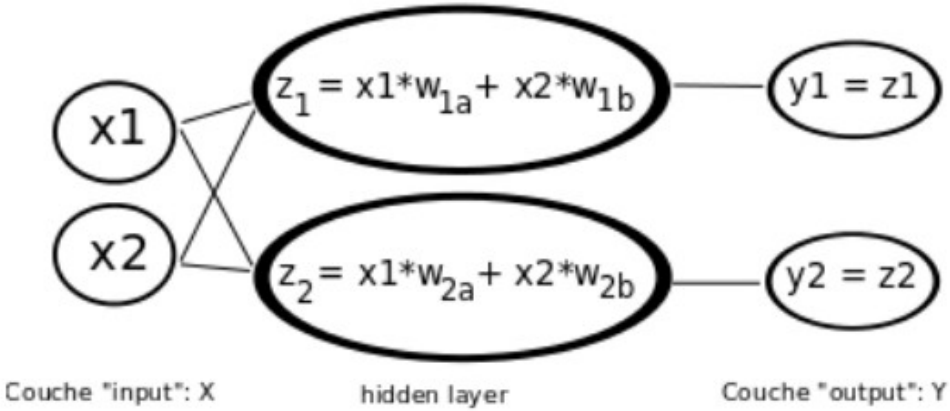


$$
\begin{aligned}
X &= \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
W &= (w_a, w_b) \\
Z &= WX = (w_a, w_b) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = (x_1 w_a + x_2 w_b) \\
Y &= Z = (x_1 w_a + x_2 w_b)
\end{aligned}
$$



$$
\begin{aligned}
X &= \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
W &= \begin{pmatrix} w_{1a}, w_{1b} \\ w_{2a}, w_{2b} \end{pmatrix} \\
Z &= WX = \begin{pmatrix} w_{1a}, w_{1b} \\ w_{2a}, w_{2b} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} w_{1a}x_1 + w_{1b}x_2 \\ w_{2a}x_1 + w_{2b}x_2 \end{pmatrix} \\
Y &= Z = \begin{pmatrix} w_{1a}x_1 + w_{1b}x_2 \\ w_{2a}x_1 + w_{2b}x_2 \end{pmatrix}
\end{aligned}
$$

Réseau de neurones « standard »
Apprentissage (back-propagation)



$$X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$W = \begin{pmatrix} w_{1a}, w_{1b} \\ w_{2a}, w_{2b} \end{pmatrix}$$

$$Z = WX = \begin{pmatrix} w_{1a}, w_{1b} \\ w_{2a}, w_{2b} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} w_{1a}x_1 + w_{1b}x_2 \\ w_{2a}x_1 + w_{2b}x_2 \end{pmatrix}$$

$$Y = Z = \begin{pmatrix} w_{1a}x_1 + w_{1b}x_2 \\ w_{2a}x_1 + w_{2b}x_2 \end{pmatrix}$$

$$W = W + \alpha \frac{dJ(Y,T)}{dW}$$

Réseau de neurones profonds

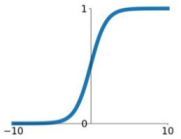
- Rendre les réseaux non-linéaires

$$Z = f(WX + b)$$

Activation Functions

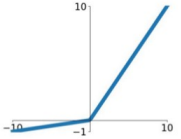
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



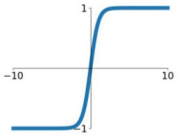
Leaky ReLU

$\max(0.1x, x)$



tanh

$\tanh(x)$

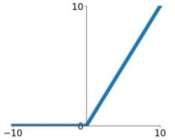


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

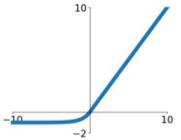
ReLU

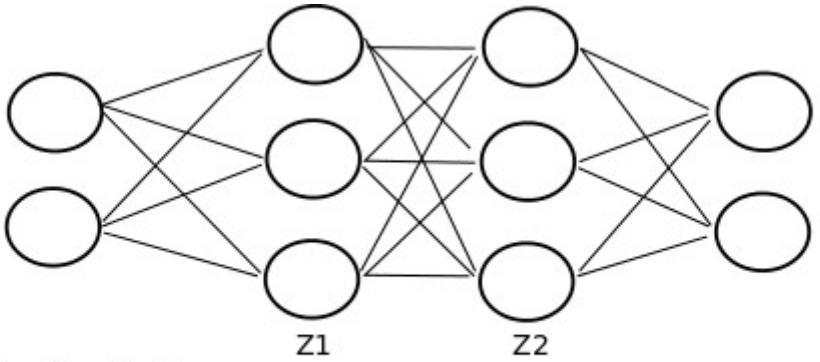
$\max(0, x)$



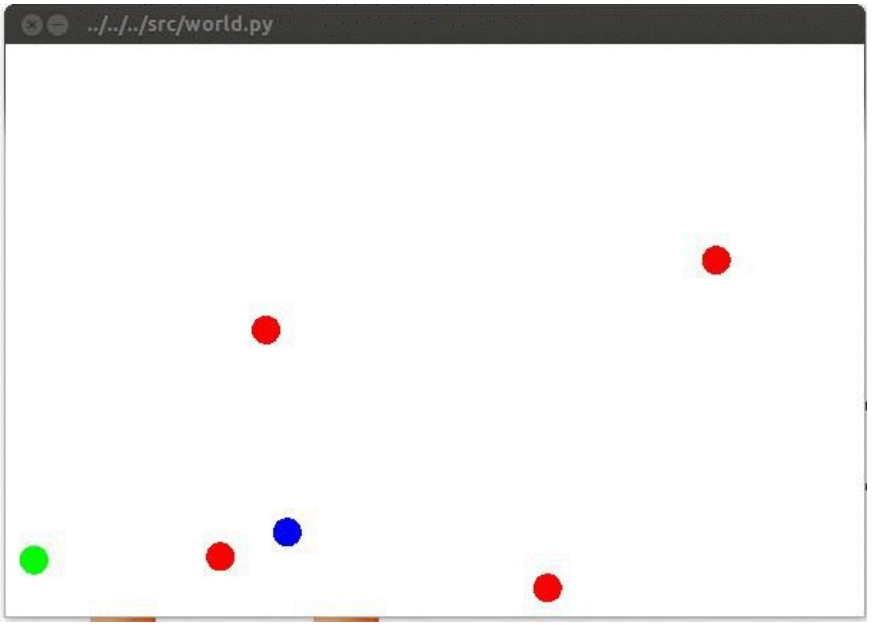
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





Implémentation :
Rappel environnement

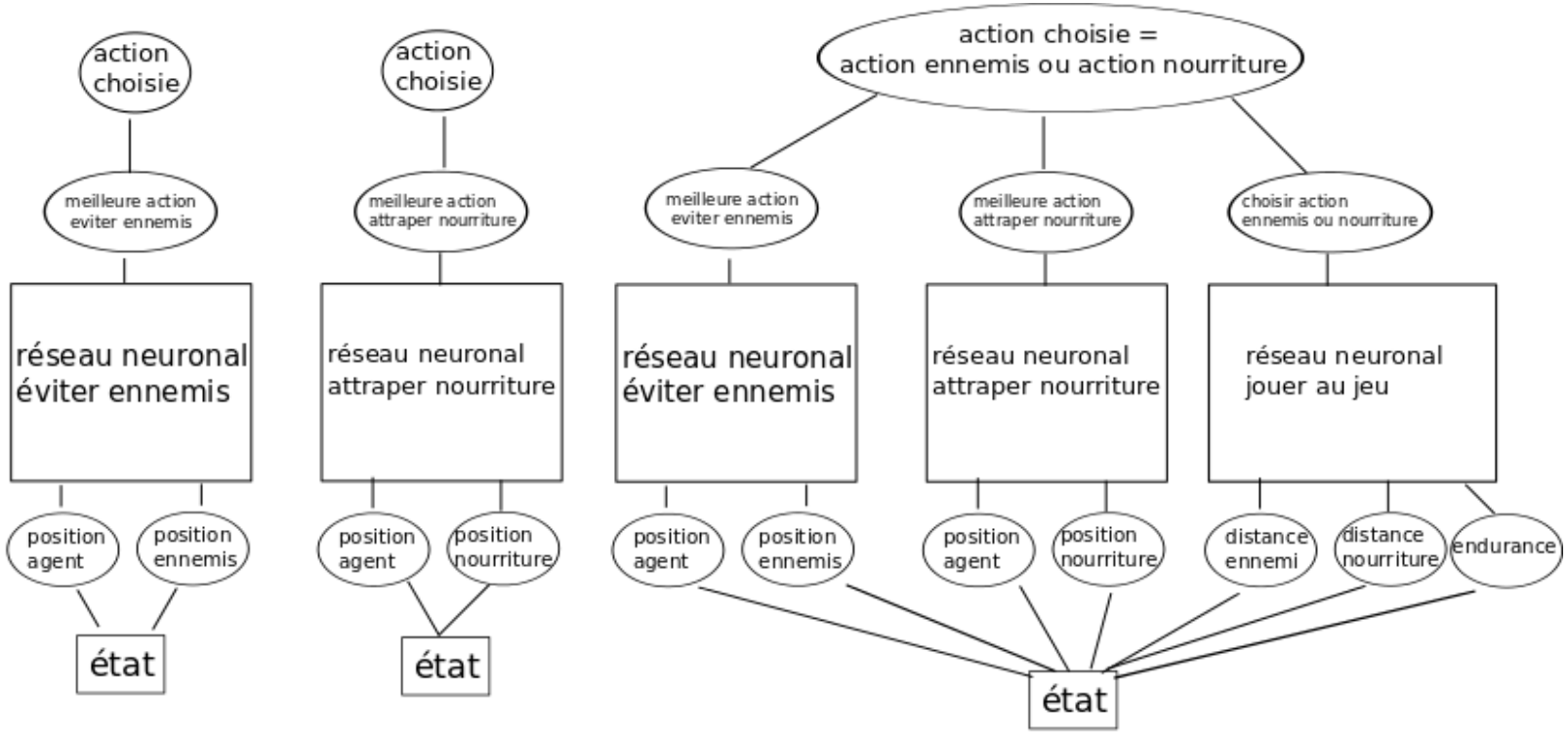


Implémentation (2):
rappel pseudo-code

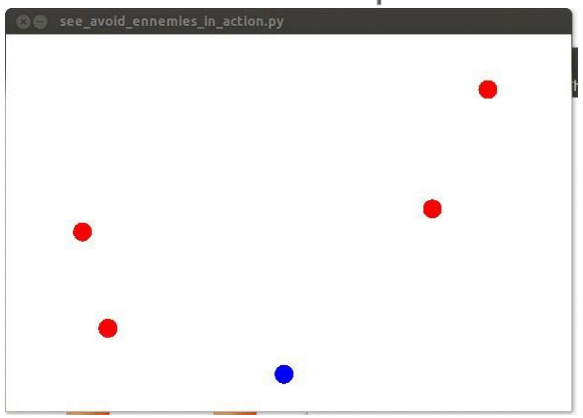
```
# phase d'apprentissage
world = new World()
agent = new Agent()
max_episodes = 100000
gamma = 0.9
epsilon = 0.1
alpha = 0.1

for i in range(max_episodes) :
  state = world.reset()
  while not world.game_over() :
    action = agent.choose_action(state, epsilon)
    reward, next_state = world.do_action(state, action)
    a2 = agent.choose_action(next_state, epsilon)

    if world.game_over() :
      agent.Q[state][action] += alpha*(reward)
    else :
      agent.Q[state][action] += alpha*(reward+gamma*agent.Q[next_state][a2])

    state = next_state


  # phase une fois l'apprentissage effectué
  state = world.reset()
  while not world.game_over() :
    action = agent.choose_action(state, epsilon)
    state, _ = world.do_action(action)
```

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

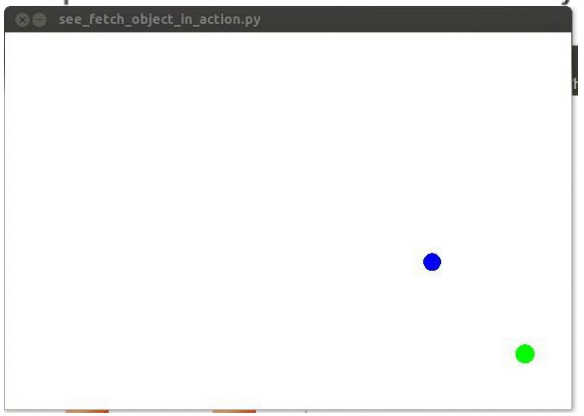
Implémentation (3): phases d'apprentissage



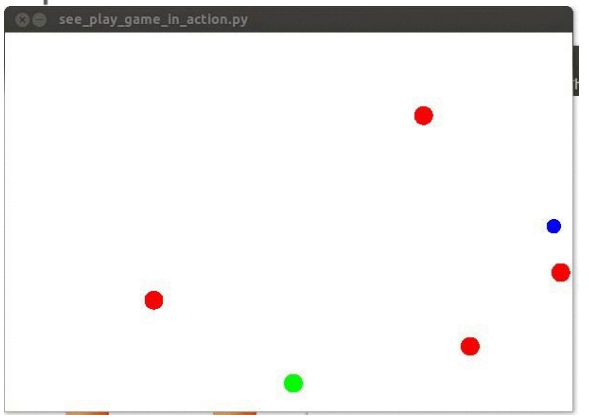
Ennemis uniquement



Nourriture uniquement

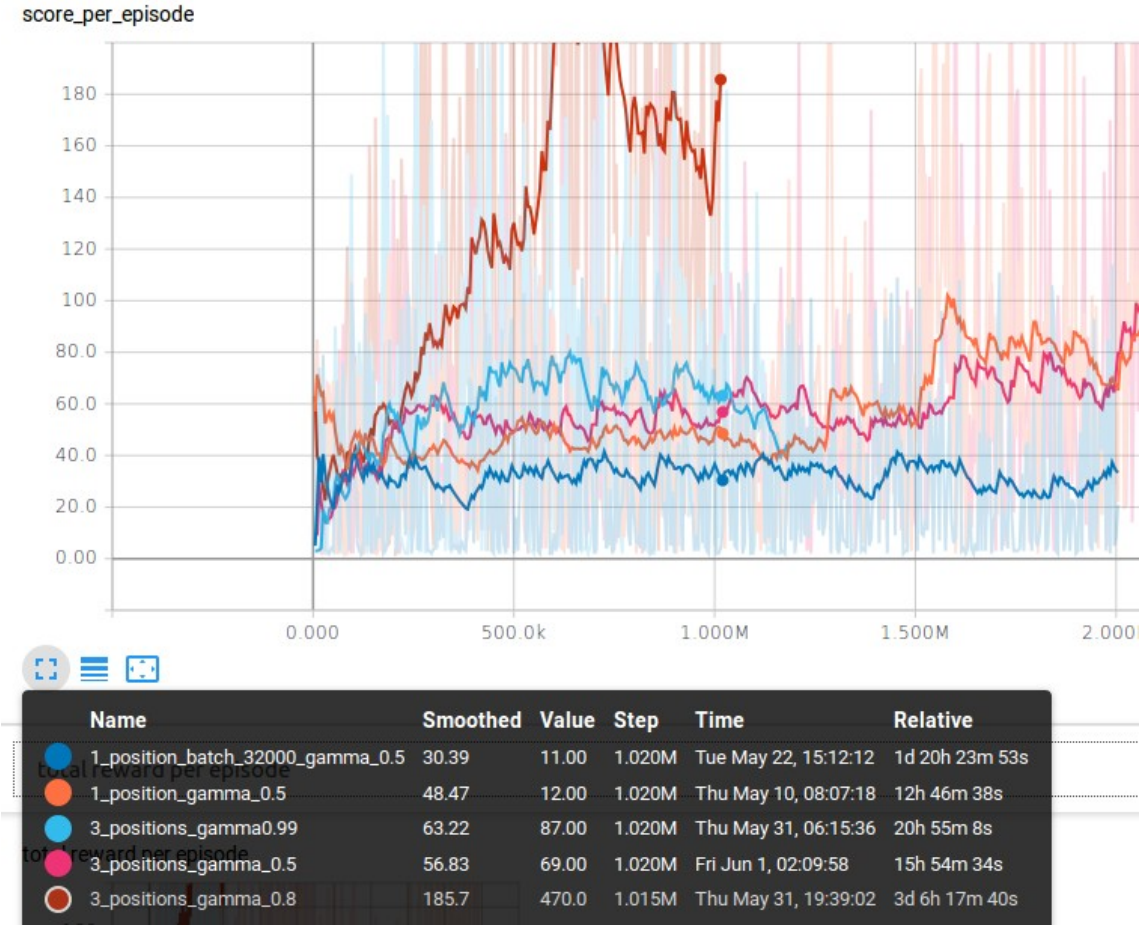


Jeu complet



Résultats

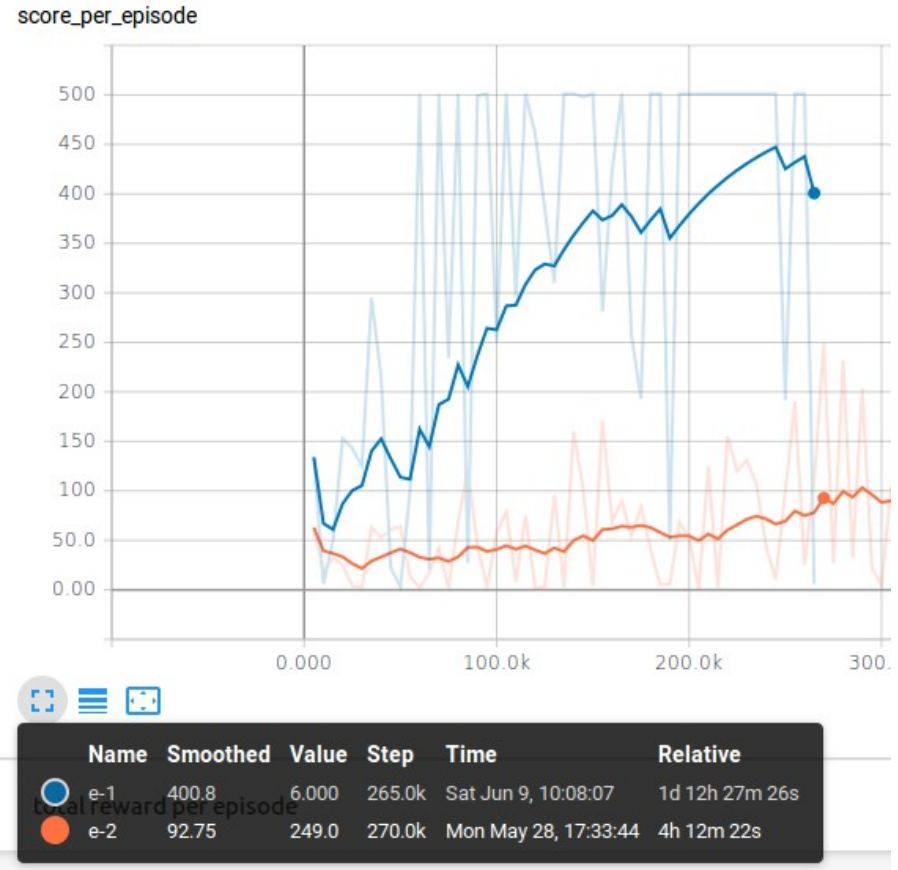
→ Choisir adéquatement la manière de représenter l'environnement
→ Choix du discount factor



Name	Smoothed	Value	Step	Time	Relative
1_position_batch_32000_gamma_0.5	30.39	11.00	1.020M	Tue May 22, 15:12:12	1d 20h 23m 53s
1_position_gamma_0.5	48.47	12.00	1.020M	Thu May 10, 08:07:18	12h 46m 38s
3_positions_gamma0.99	63.22	87.00	1.020M	Thu May 31, 06:15:36	20h 55m 8s
3_positions_gamma_0.5	56.83	69.00	1.020M	Fri Jun 1, 02:09:58	15h 54m 34s
3_positions_gamma_0.8	185.7	470.0	1.015M	Thu May 31, 19:39:02	3d 6h 17m 40s

Résultats (2)

→ Learning rate (α) $$Q(s,a) = Q(s,a) + \alpha(\{r + \gamma Q(s',a')\} - Q(s,a))$$



Conclusion

→ Domaine pas enseigné à l'hepia

→ Théorie éloignée de la pratique

→ Domaine évolue sans cesse

→ Architecture pas adaptée au jeu construit