IFT 608 / IFT 702
Planification en intelligence artificielle

Deep Q-Learning

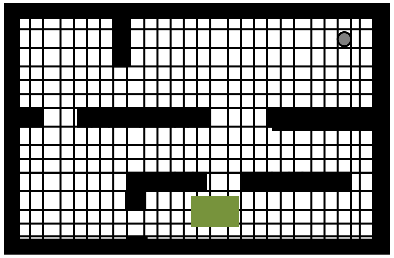
Professeur: Froduald Kabanza

Assistants: D’Jeff Nkashama & Jordan Félicien Masakuna

Université de Sherbrooke

Sujet couvert

- Deep Q-Learning



Cadre général

Maximiser $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$, avec $0 \leq \gamma \leq 1$

Agent

Politique

$\pi(s,a)$

Action a

$a_1 \rightarrow a_2 \rightarrow \ldots a_t$

$$U^{\pi}(s) = \Sigma_{s' \in S} P(s'|s, \pi(s)) [R(s, \pi(s), s') + \gamma U^{\pi}(s')]$$

$$U(s) = \max_{a \in A(s)} \sum_{s' \in S} P(s'|s,a) [R(s,a,s') + \gamma U(s')]$$

$$Q(s,a) = \sum_{s' \in S} P(s'|s,a) [R(s,a,s') + \gamma \max Q(s',a')]$$

$$\pi^*(s) = \operatorname*{argmax}_{a \in A(s)} Q(s,a) = \operatorname*{argmax}_{a \in A(s)} \sum_{s' \in S} P(s'|s,a) [R(s,a,s') + \gamma U(s')]$$

Récompense r

$r_1 \rightarrow r_2 \rightarrow \ldots r_t$

Environnement

État s

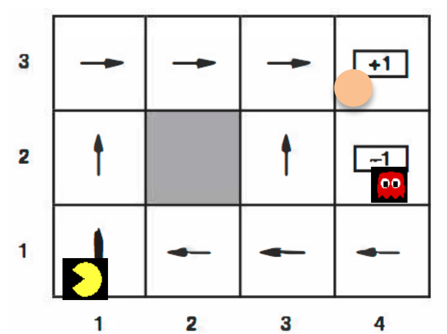
$s_1 \rightarrow s_2 \rightarrow \ldots s_t$

Approche par estimation directe

- Approche la plus simple: **calculer la moyenne de ce qui est observé** dans les essais
- **Essais**

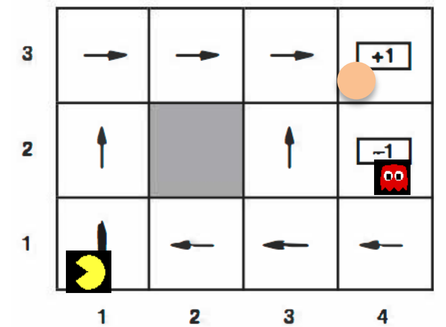
1. $(1,1) \xrightarrow[\text{Up}]{-.04} (1,2) \xrightarrow[\text{Up}]{-.04} (1,3) \xrightarrow[\text{Right}]{-0.4} (1,2) \xrightarrow[\text{Up}]{-.04} (1,3) \xrightarrow[\text{Right}]{-.04} (2,3) \xrightarrow[\text{Right}]{-.04} (3,3) \xrightarrow[\text{Right}]{+1} (4,3)$
2. $(1,1) \xrightarrow[\text{Up}]{-.04} (1,2) \xrightarrow[\text{Up}]{-.04} (1,3) \xrightarrow[\text{Right}]{-0.4} (2,3) \xrightarrow[\text{Right}]{-.04} (3,3) \xrightarrow[\text{Right}]{-.04} (3,2) \xrightarrow[\text{Up}]{-.04} (3,3) \xrightarrow[\text{Right}]{+1} (4,3)$
3. $(1,1) \xrightarrow[\text{Up}]{-.04} (2,1) \xrightarrow[\text{Left}]{-.04} (3,1) \xrightarrow[\text{Up}]{-.04} (3,2) \xrightarrow[\text{Up}]{-1} (4,2)$

- Estimation de $U^{\pi}((1,1))$
  - dans l'essai 1, la somme des récompenses à partir de (1,1) est 0.76
  - dans l'essai 2, on obtient également 0.76
  - dans l'essai 3, on obtient plutôt -1.12
  - l'estimation directe de $U((1,1))$ est donc $(0.76+0.76-1.12)/3 \approx 0.133$



Q-learning

1. $(1,1) \xrightarrow[\text{Up}]{-.04} (1,2) \xrightarrow[\text{Up}]{-.04} (1,3) \xrightarrow[\text{Right}]{-0.4} (1,2) \xrightarrow[\text{Up}]{-.04} (1,3) \xrightarrow[\text{Right}]{-.04} (2,3) \xrightarrow[\text{Right}]{-.04} (3,3) \xrightarrow[\text{Right}]{+1} (4,3)$
2. $(1,1) \xrightarrow[\text{Up}]{-.04} (1,2) \xrightarrow[\text{Up}]{-.04} (1,3) \xrightarrow[\text{Right}]{-0.4} (2,3) \xrightarrow[\text{Right}]{-.04} (3,3) \xrightarrow[\text{Right}]{-.04} (3,2) \xrightarrow[\text{Up}]{-.04} (3,3) \xrightarrow[\text{Right}]{+1} (4,3)$
3. $(1,1) \xrightarrow[\text{Up}]{-.04} (2,1) \xrightarrow[\text{Left}]{-.04} (3,1) \xrightarrow[\text{Up}]{-.04} (3,2) \xrightarrow[\text{Up}]{-1} (4,2)$



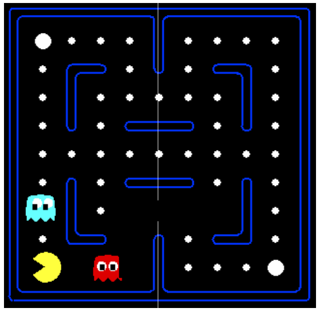
$$Q(s,a) \leftarrow Q(s,a) + \alpha\ (\ R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)\ )$$

function Q-LEARNING-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r
  persistent: Q, a table of action values indexed by state and action, initially zero
    N_{sa}, a table of frequencies for state–action pairs, initially zero
    s, a, the previous state and action, initially null

  if s is not null **then**
    increment $N_{sa}[s,a]$
    $Q[s,a] \leftarrow Q[s,a] + \alpha(N_{sa}[s,a])(r + \gamma \max_{a'} Q[s',a'] - Q[s,a])$
  $s, a \leftarrow s', \text{argmax}_{a'}\ f(Q[s',a'], N_{sa}[s',a'])$
  return a

Approximation de fonction

- Représenter un état en utilisant un vecteur de caractéristiques (*features*)
  - Les caractéristiques sont des fonctions réelles (souvent 0/1) qui capturent les propriétés saillants des états
  - La taille de l’espace d’états représentés par leurs caractéristiques est beaucoup plus petit que l’espace d’états d’origine – c’est une approximation
  - Exemples de caractéristiques (*features*):
    - » Distance au fantôme le plus proche
    - » Distance à la nourriture la plus proche
    - » Nombre de fantômes
    - » 1 / (distance à la nourriture)^2
    - » Pacman proche d’un fantôme ? (0/1)
    - » etc.



  - On peut aussi décrire la fonction de qualité Q(s, a) avec des caractéristiques (ex. l’action permettant à Pacman d’être proche de la nourriture)

Approximation de fonction

- Étant donné un vecteur $f = [f_1, \ldots, f_n]$ de *features* d'état et un vecteur de poids correspondant $w = [w_1, \ldots, w_n]$, on peut approximer U par une fonction linéaire

  $$\widehat{U}_w = \sum_{i=1}^{n} w_i f_i(s)$$

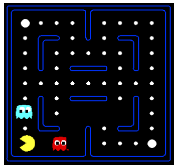
- De façon similaire, avec un vecteur de *features* Q et des poids correspondants:

  $$\widehat{Q}_w(s,a) = \sum_{i=1}^{n} w_i f_i(s,a)$$

- On peut alors utiliser l'apprentissage supervisée pour apprendre $\widehat{U}_w$ *et* $\widehat{Q}_w$

- On peut aussi utiliser des fonctions d'approximation non linéaire

  $$\widehat{U}_w = f(s)$$
  $$\widehat{Q}_w(s,a) = f(s)$$



- En particulier, on pourrait utiliser un réseau de neurones – ce qu'on appelle apprentissage par renforcement profond.

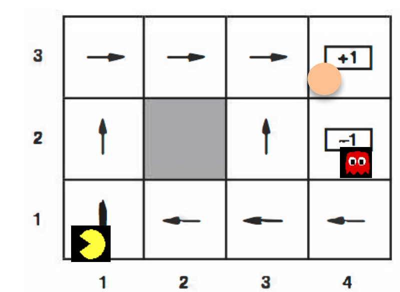
- Voyons cela plus en détail.

Approximer l'estimation directe de l'utilité

- La méthode de prédiction Monte-Carlo (apprentissage passif par estimation directe de l'utilité) simule des trajectoires d'exécution dans l'espace d'états x(s)≡(x_1,x_2) et calcule, pour chaque état, la somme des récompenses cumulées jusqu'à l'état terminal, U(s) approximé par $U(x_1,x_2)$.

- Chaque occurrence d'état (x,y) est une donnée, étiquetée par son utilité correspondante U(x,y), et l'ensemble de toutes les occurrence constitue un ensemble de données d'entraînement pour un algorithme **d'apprentissage supervisé** qui peut apprendre U. On pourrait utiliser n'importe quel algorithme d'apprentissage supervisé.

- Par exemple, avec le même exemple utilisé pour la méthode d'apprentissage passif par estimation directe, approximons l'utilité par une fonction linéaire de *features* – les *features* étant simplement les positions x_1 et x_2.



$$\widehat{U}_w(s) = w_0 + w_1 x_1 + w_2 x_2$$

Approximer l'apprentissage par différence temporelle

- La règle d'apprentissage des poids de la fonction d'approximation linéaire pour l'apprentissage par différence temporelle est

$$w_i \leftarrow w_i + \alpha \left[R(s,a,s') + \gamma \widehat{U_w}(s') - \widehat{U_w}(s)\right] \frac{\partial \widehat{U_w}(s)}{\partial w_i}$$

C.-à-d.

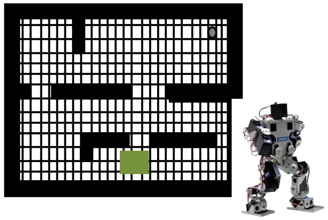
$$w_i \leftarrow w_i + \alpha \left[R(s,a,s') + \gamma \widehat{U_w}(s') - \widehat{U_w}(s)\right] X_i(s)$$

- Celle pour Q-Learning est

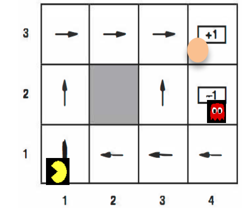
$$w_i \leftarrow w_i + \alpha \left[R(s,a,s') + \gamma \max_{a'} \widehat{Q_w}(s',a') - \widehat{Q_w}(s,a)\right] \frac{\partial \widehat{U_w}(s)}{\partial w_i}$$

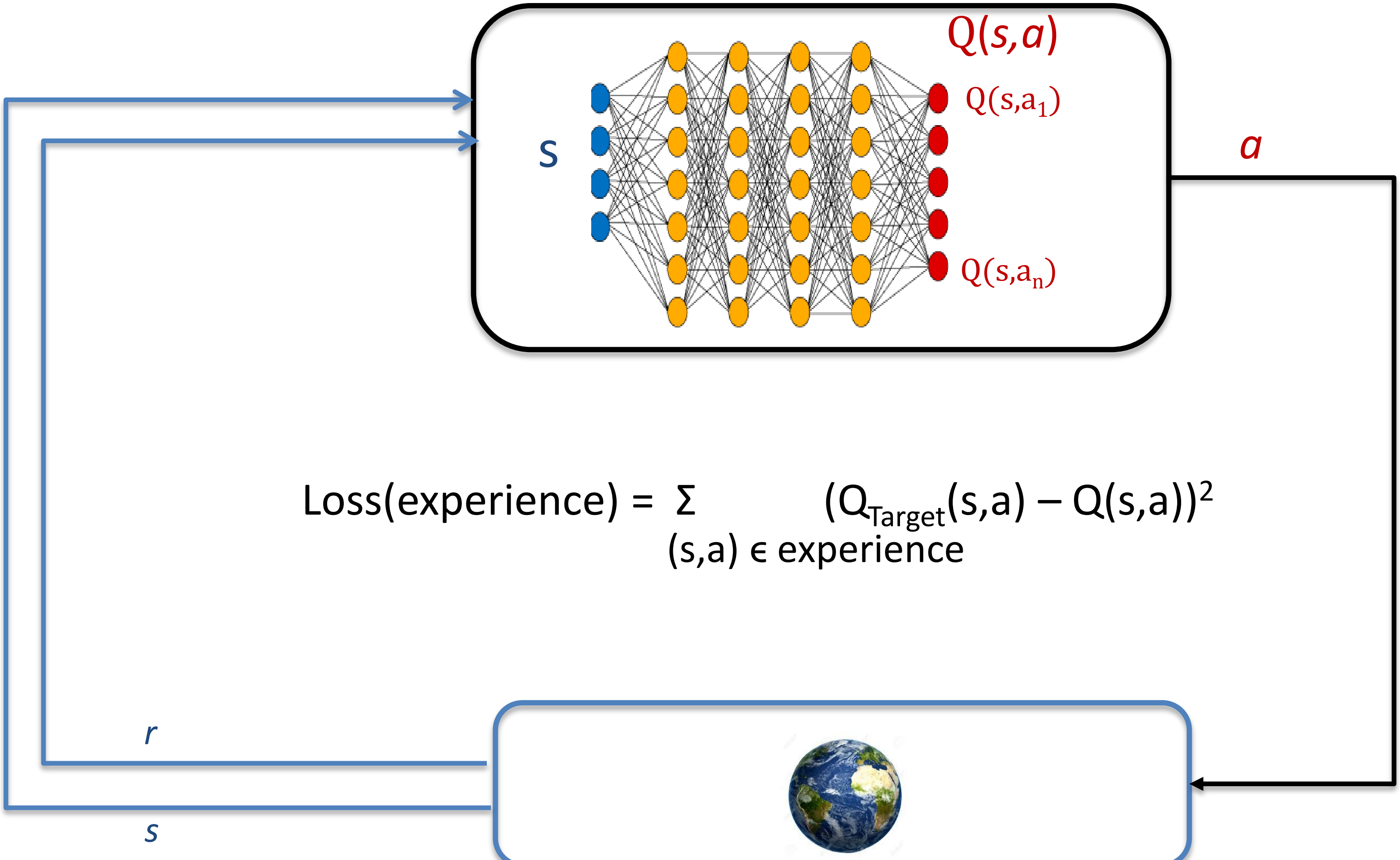
C.-à-d.

$$w_i \leftarrow w_i + \alpha \left[R(s,a,s') + \gamma \max_{a'} \widehat{Q_w}(s',a') - \widehat{Q_w}(s,a)\right] X_i(s)$$

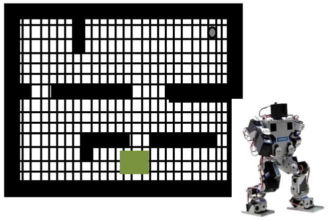


Idée derrière Deep Q-Learning

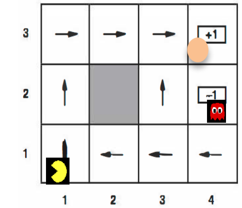


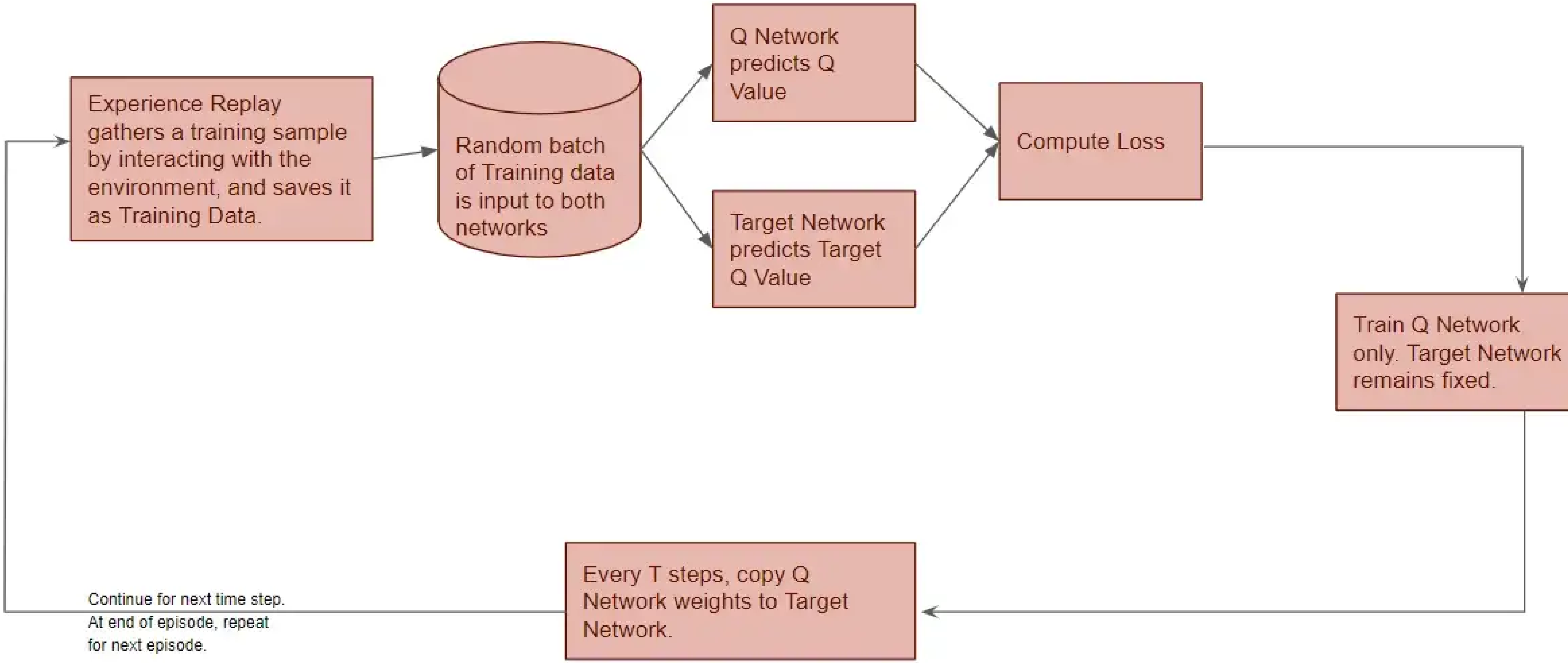


$$\text{Loss(experience)} = \sum_{(s,a)\,\in\,\text{experience}} \left(Q_{Target}(s,a) - Q(s,a)\right)^2$$

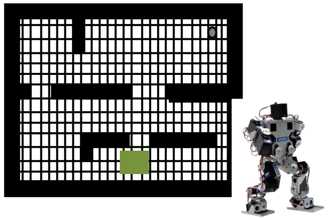


DQN (Deep Q-Network)

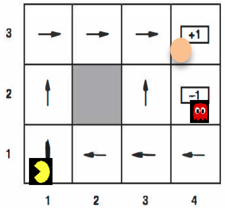


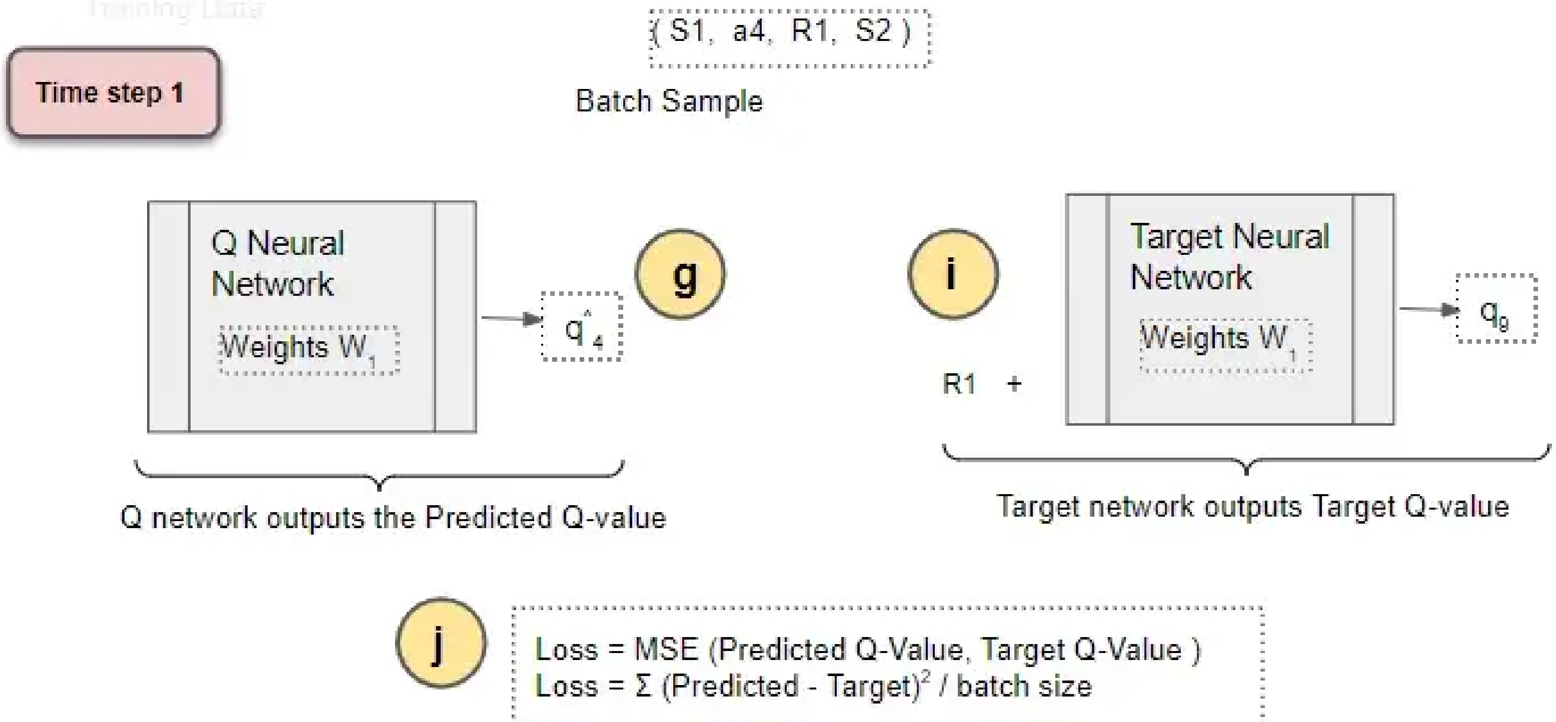


https://towardsdatascience.com/reinforcement-learning-explained-visually-part-5-deep-q-networks-step-by-step-5a5317197f4b



DQN (Deep Q-Network)





https://towardsdatascience.com/reinforcement-learning-explained-visually-part-5-deep-q-networks-step-by-step-5a5317197f4b

Conclusion

- DQN est approxime la Q-function par un réseau de neurone
- L’architecture comprend deux réseaux de neurones (Q-Network et Q-Target)

Vous devriez être capable de...

- Expliquer comment DQN fonctionne
- Implémenter DQN
- L’appliquer à un problème de votre choix