# Introduction à *l'apprentissage par renforcement* (RL)

Léo Gagnon

July 6, 2021

Université de Montréal

# Plan

1. Contexte
2. Cadre formel
3. Comment apprendre de ses expériences
4. Algorithmes efficaces
5. Conclusion

# Contexte

Le RL s'intéresse à comment un agent peux apprendre un comportement en intéragissant avec l'environnement.

RL = Info ∩ Math ∩ Neuro ∩ Psyco ∩ Éco

# Exemples et importance de l'apprentissage

On pourrais vouloir apprendre à :

- jouer aux échecs
- jouer à Dota II
- marcher
- conduire

**L'histoire a montré que sans apprentissage, bien résoudre ce genre de tâche est extrêmement difficile.**

# Cadre formel

# Cadre formel

De façon générale, notre agent sera dans la situation suivante

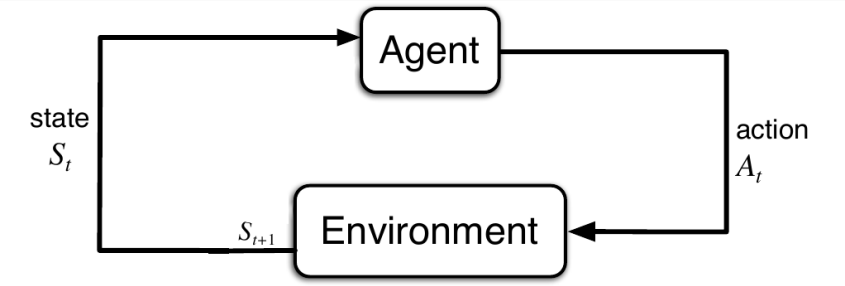


**Figure 1:** Boucle perception-action

Où la boucle commence par un état initial $S_0$ et finit *potentiellement* par un état $S_T$.

# Cadre formel

De façon générale, notre agent sera dans la situation suivante

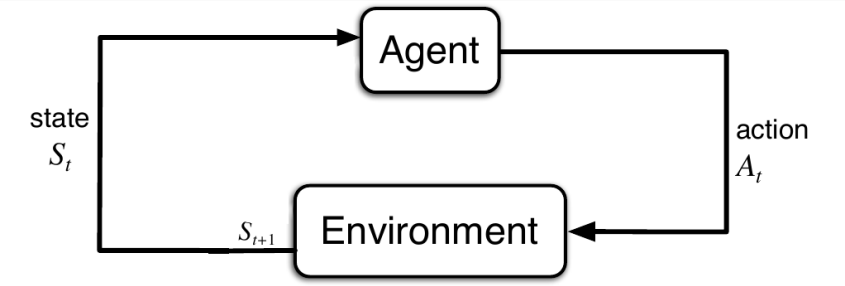


**Figure 1:** Boucle perception-action

Où la boucle commence par un état initial $S_0$ et finit *potentiellement* par un état $S_T$.

# Cadre formel

De façon générale, notre agent sera dans la situation suivante

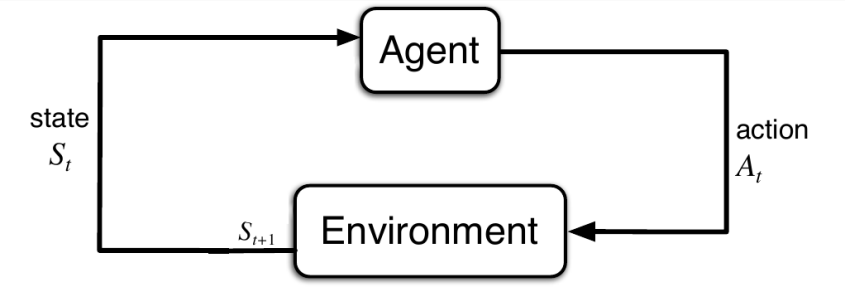


**Figure 1:** Boucle perception-action

Où la boucle commence par un état initial $S_0$ et finit *potentiellement* par un état $S_T$.

**Il manque quelque chose : comment équipe-t-on l'agent de motivation vers un but?**

# Cadre formel

De façon générale, notre agent sera dans la situation suivante

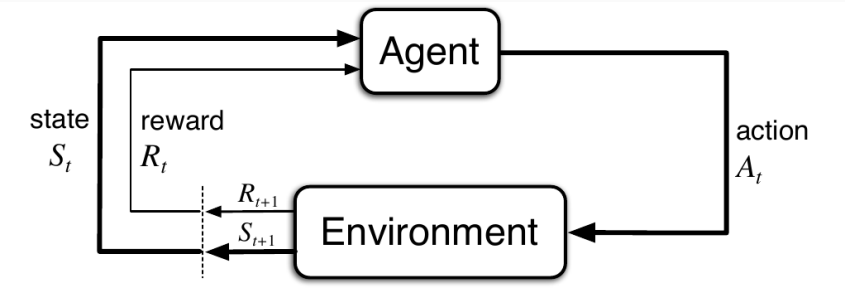


**Figure 2:** Boucle perception-action avec une notion de récompense

**La façon généralement acceptée est de le concevoir de façon à maximiser une récompense fournie par l'environnement.**

# Cadre formel

De façon générale, notre agent sera dans la situation suivante

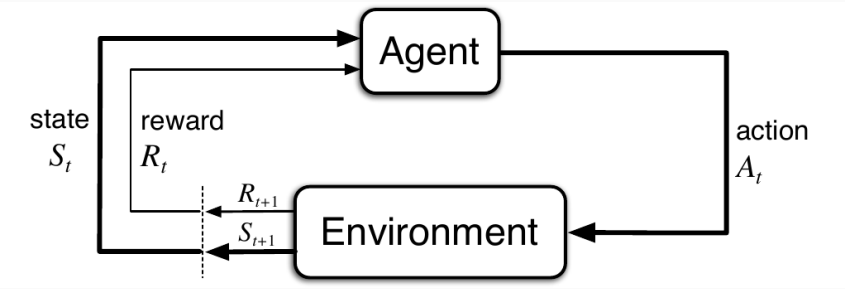


**Figure 2:** Boucle perception-action avec une notion de récompense

**La façon généralement acceptée est de le concevoir de façon à maximiser une récompense fournie par l'environnement.**

Matière à réflexion : étant dans la position de cet agent, comment sommes-nous équipé d'un but? Qu'est-ce qui constitue notre l'environnement?

# Cadre formel

De façon générale, notre agent sera dans la situation suivante. Ce genre de système est naturellement modélisé par un *processus stochastique*.

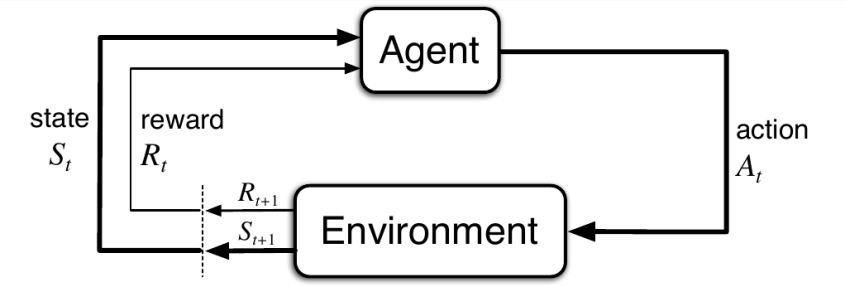


**Figure 3:** Boucle perception-action avec une notion de récompense

# Cadre formel

Un processus stochastique est une suite de variables aléatoire.
Dans le cas qui nous intéresse, à chaque temps $t$ l'agent

- reçoit une représentation de l'état de l'environnement $S_t \in \mathcal{S}$
- choisi une action $A_t \in \mathcal{A}$
- reçoit une récompense $R_t \in \mathcal{R}$ pour l'action au temps $t-1$

# Cadre formel

Un processus stochastique est une suite de variables aléatoire.
Dans le cas qui nous intéresse, à chaque temps $t$ l'agent

- reçoit une représentation de l'état de l'environnement $S_t \in \mathcal{S}$
- choisi une action $A_t \in \mathcal{A}$
- reçoit une récompense $R_t \in \mathcal{R}$ pour l'action au temps $t-1$

Ceci produit une suite (souvent appelé *trajectoire*) de variables aléatoire chronologiquement dépendantes

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots S_{T-1}, A_{T-1}, S_T$$

# Cadre formel

Un processus stochastique est une suite de variables aléatoire.
Dans le cas qui nous intéresse, à chaque temps $t$ l'agent

- reçoit une représentation de l'état de l'environnement $S_t \in \mathcal{S}$
- choisi une action $A_t \in \mathcal{A}$
- reçoit une récompense $R_t \in \mathcal{R}$ pour l'action au temps $t-1$

Ceci produit une suite (souvent appelé *trajectoire*) de variables aléatoire chronologiquement dépendantes

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

# Cadre formel

Pour simplifier notre modélisation, on suppose généralement que le processus stochastique est ***Markovien*** (MDP). Cela signifie que les V.A au temps $t$ peuvent seulement dépendre de celles au temps $t-1$. Dans ce cas, il existe une fonction

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

définissant complètement la *dynamique* de l'environnement

# Cadre formel

L'hypothèse de Markov permet à l'agent de pouvoir agir uniquement en fonction de l'état précédant. Son comportement peut alors être complètement décrit par une distribution conditionnelle qu'on appelle sa *stratégie*

$$\pi(a|s) : \mathcal{S} \to \mathcal{P}(\mathcal{A})$$

# Cadre formel : résumé

L'apprentissage par renforcement est formalisé par un MDP défini par une distribution $p(s', r|s, a)$.

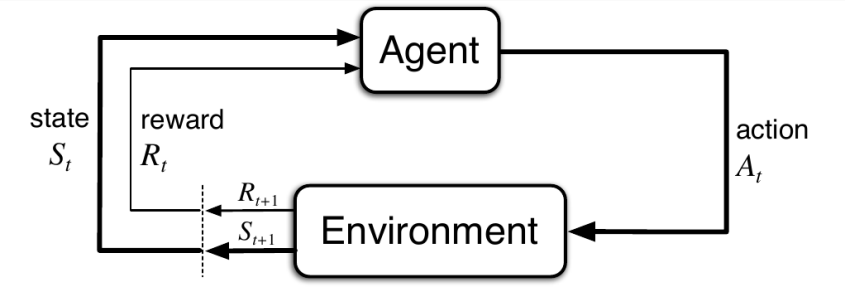


**Figure 4:** Boucle perception-action avec une notion de récompense

Le problème maintenant : créer un agent capable d'apprendre une stratégie $\pi(a|s)$ qui obtient *beaucoup de récompenses*.

# Beaucoup de récompenses?

Que signifie "obtenir beaucoup de récompenses"?

- Naïvement, on pourrait toujours vouloir maximiser $\mathbb{E}[R_{t+1}]$.

# Beaucoup de récompenses?

Que signifie "obtenir beaucoup de récompenses"?

- Naïvement, on pourrait toujours vouloir maximiser $\mathbb{E}[R_{t+1}]$.
  - Mauvaise idée

# Beaucoup de récompenses?

Que signifie "obtenir beaucoup de récompenses"?

- Naïvement, on pourrait toujours vouloir maximiser $\mathbb{E}[R_{t+1}]$.
  - Mauvaise idée
- Un choix plus sensé pour la quantité d'intéret serait $\mathbb{E}[\sum_{k=0}^{\infty} R_{t+k+1}]$ (*expected return*)

# Beaucoup de récompenses?

Que signifie "obtenir beaucoup de récompenses"?

- Naïvement, on pourrait toujours vouloir maximiser $\mathbb{E}[R_{t+1}]$.
  - Mauvaise idée
- Un choix plus sensé pour la quantité d'intéret serait $\mathbb{E}[\sum_{k=0}^{\infty} R_{t+k+1}]$ (*expected return*)
  - Pas réalisable

# Beaucoup de récompenses?

Que signifie "obtenir beaucoup de récompenses"?

- Naïvement, on pourrait toujours vouloir maximiser $\mathbb{E}[R_{t+1}]$.
  - Mauvaise idée
- Un choix plus sensé pour la quantité d'intéret serait $\mathbb{E}[\sum_{k=0}^{\infty} R_{t+k+1}]$ (*expected return*)
  - Pas réalisable
- La quantité généralement utilisée est une hybride appelée le *expected discounted return*

$$\mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}] \text{ pour } \gamma \in [0, 1]$$

# Beaucoup de récompenses?

Que signifie "obtenir beaucoup de récompenses"?

- Naïvement, on pourrait toujours vouloir maximiser $\mathbb{E}[R_{t+1}]$.
  - Mauvaise idée
- Un choix plus sensé pour la quantité d'intéret serait $\mathbb{E}[\sum_{k=0}^{\infty} R_{t+k+1}]$ (*expected return*)
  - Pas réalisable
- La quantité généralement utilisée est une hybride appelée le *expected discounted return*

$$\mathbb{E}[\sum_{k=0}^{\infty} G_t] \text{ pour } \gamma \in [0, 1]$$

# Cadre formel : résumé

L'apprentissage par renforcement est formalisé par un MDP défini par une distribution $p(s', r|s, a)$.

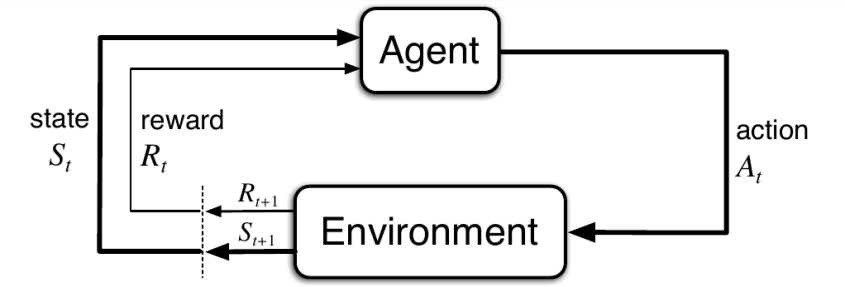


**Figure 5:** Boucle perception-action avec une notion de récompense

Le problème maintenant : créer un agent capable d'apprendre une stratégie $\pi(a|s)$ qui maximise

$$\mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}]$$

pour tout les états $s \in \mathcal{S}$

# Itération de stratégie

# Évaluation de stratégie

Si on veux parler d'amélioration de stratégie, on doit être capable d'évaluer et de comparer différentes stratégies. Les notions suivantes sont très utiles pour ça.

**Définition**

La *value function* $v_\pi : \mathcal{S} \to \mathbb{R}$ d'une stratégie $\pi$ est une fonction qui associe à chaque état $s \in \mathcal{S}$ le expected discounted return si on suit la stratégie en partant de $s$:

$$v_\pi(s) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right]$$

# Évaluation de stratégie

Si on veux parler d'amélioration de stratégie, on doit être capable d'évaluer et de comparer différentes stratégies. Les notions suivantes sont très utiles pour ça.

**Définition**

La *q-function* $q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ d'une stratégie $\pi$ est une fonction qui associe à chaque état+action le expected discounted return si on suit la stratégie par la suite:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right]$$

# Évaluation de stratégie

Si on veux parler d'amélioration de stratégie, on doit être capable d'évaluer et de comparer différentes stratégies. Les notions suivantes sont très utiles pour ça.

**Définition**

La *q-function* $q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ d'une stratégie $\pi$ est une fonction qui associe à chaque état+action le expected discounted return si on suit la stratégie par la suite:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[ R_{t+1} + \gamma v_\pi(S_{t+1}) \middle| S_t = s, A_t = a \right]$$

# Évaluation de stratégie

Les value functions définissent un ordre partiel sur les stratégies :

$$\pi_1 \geq \pi_2 \iff v_{\pi_1}(s) \geq v_{\pi_2}(s) \text{ pour tout } s \in \mathcal{S}$$

# Évaluation de stratégie

Les value functions définissent un ordre partiel sur les stratégies :

$$\pi_1 \geq \pi_2 \iff v_{\pi_1}(s) \geq v_{\pi_2}(s) \text{ pour tout } s \in \mathcal{S}$$

Grace au facteur $\gamma$, il existe toujours une value function maximale

$$v_* = \max_\pi v_\pi$$

associée à une stratégie optimale $\pi_*$ (pas nécéssairement unique)

# Évaluation de stratégie

Comment calculer $v_\pi$? On y revient bientôt.

# Amélioration de stratégie

La value function $v_\pi$ nous donne toute l'information qu'on a besoin pour améliorer la stratégie $\pi$. En effet, $v_\pi$ nous dit quelle est la meilleure action à prendre pour chaque état si on continue à appliquer $\pi$ par la suite. On peut donc prendre comme nouvelle stratégie la **stratégie greedy/gloutone**

$$\pi'(s) = \arg\max_a q_\pi(s, a)$$

et

$$\pi' \geq \pi$$

# Amélioration de stratégie

La value function $v_\pi$ nous donne toute l'information qu'on a besoin pour améliorer la stratégie $\pi$. En effet, $v_\pi$ nous dit quelle est la meilleure action à prendre pour chaque état si on continue à appliquer $\pi$ par la suite. On peut donc prendre comme nouvelle stratégie la **stratégie greedy/gloutone**

$$\pi'(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

et

$$\pi' \geq \pi$$

# Itération de stratégies

On répéter successivement les opération d'évaluation (E) et amélioration (A) pour obtenir une séquence de stratégies de plus en plus efficaces. Puisqu'il y a un nombre fini de stratégies on converge en un nombre fini d'opérations.

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

# Évaluation de stratégie (retour)

Il existe différentes façon d'évaluer une stratégie. On peux généralement les classer dans deux familles : avec ou sans modèle de l'environnement (Model-based ou Model-free)

# Évaluation Model-based

**Model-based :**

**Figure 6:** Évalution model-based

# Évaluation Model-based

**Model-based :** Si la dynamique $p(s', r|s, a)$ est connue, on peux simplement résoudre le système de $|\mathcal{S}|$ équations linéaires à $|\mathcal{S}|$ inconnues

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned}$$

# Évaluation Model-based

**Model-based :** Si la dynamique $p(s', r|s, a)$ est connue, on peux simplement résoudre le système de $|\mathcal{S}|$ équations linéaires à $|\mathcal{S}|$ inconnues

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned}$$

Pas pratique du tout : $p(s', r|s, a)$ est rarement connue et $\mathcal{S}$ est souvent grand

# Évaluation Model-free

Si on ne connais pas $p(s', r|s, a)$, on va approximer $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ en échantillonant des discounted returns $G_t$ (en intéragissant avec l'environnement).

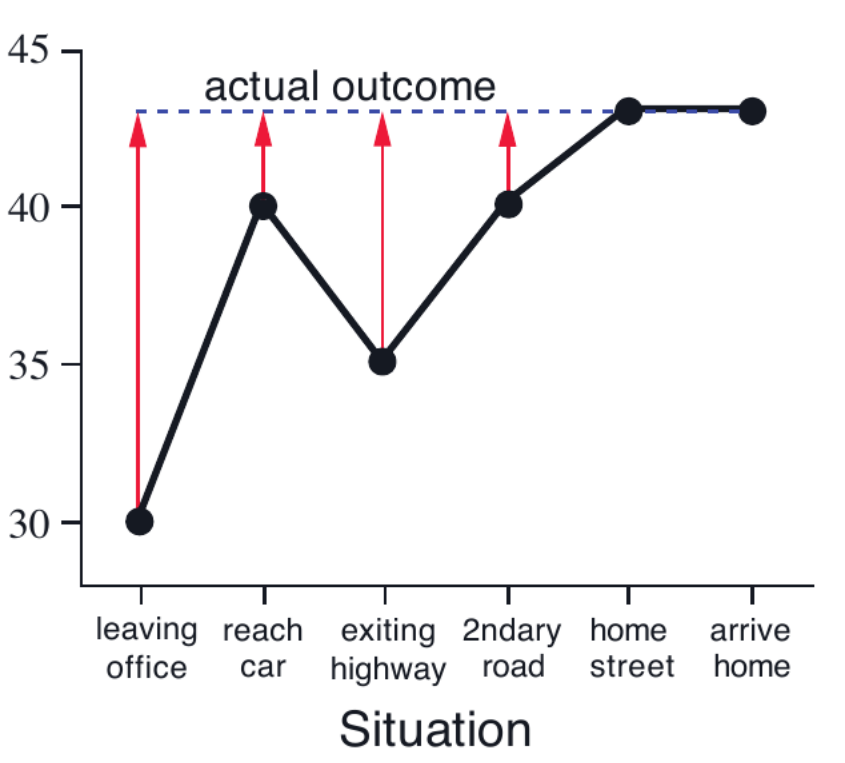


**Figure 7:** Évalution Model-free de type Monte-Carlo

# Évaluation Model-free

Si on ne connais pas $p(s', r|s, a)$, on va approximer $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ en échantillonant des discounted returns $G_t$ (en intéragissant avec l'environnement).

## Évaluation de $v_\pi$ Monte-Carlo

1. Commencer avec une approximation $V(s)$ de $v_\pi(s)$
2. Répéter pour toujours
   - 2.1 Générer une épiosode en suivant la stratégie $\pi$ avec probabilité $(1 - \epsilon)$ et en agissant aléatoire sinon.
   - 2.2 À la fin de l'épisode, mettre à jour $V(s)$ pour tout les états visités dans l'épisode avec la formule suivante

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

# Évaluation Model-free

Si on ne connais pas $p(s', r|s, a)$, on va approximer $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ en échantillonant des discounted returns $G_t$ (en intéragissant avec l'environnement).

## Évaluation de $v_\pi$ Monte-Carlo

1. Commencer avec une approximation $V(s)$ de $v_\pi(s)$
2. Répéter pour toujours
   - 2.1 Générer une épiosode en suivant la stratégie $\pi$ avec probabilité $(1 - \epsilon)$ et en agissant aléatoire sinon.
   - 2.2 À la fin de l'épisode, mettre à jour $V(s)$ pour tout les états visités dans l'épisode avec la formule suivante

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

Problème : Avec seulement la value-function et sans $p(s', r|s, a)$ on ne peux pas faire l'amélioration de stratégie.

# Évaluation Model-free

Si on ne connais pas $p(s', r|s, a)$, on va approximer $q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$ en échantillonant des discounted returns $G_t$ (en intéragissant avec l'environnement).

## Évaluation de $q_\pi$ Monte-Carlo

1. Commencer avec une approximation $Q(s, a)$ de $q_\pi(s, a)$
2. Répéter pour toujours
   - 2.1 Générer une épiosode en suivant la stratégie $\pi$
   - 2.2 À la fin de l'épisode, mettre à jour $Q(s, a)$ pour tout les paires d'état-action visitées dans l'épisode avec la formule suivante

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t - Q(S_t, A_t)]$$

# Évaluation Model-free

Si on ne connais pas $p(s', r|s, a)$, on va approximer $q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$ en échantillonant des discounted returns $G_t$ (en intéragissant avec l'environnement).

## Évaluation de $q_\pi$ Monte-Carlo

1. Commencer avec une approximation $Q(s, a)$ de $q_\pi(s, a)$
2. Répéter pour toujours
   - 2.1 Générer une épiosode en suivant la stratégie $\pi$
   - 2.2 À la fin de l'épisode, mettre à jour $Q(s, a)$ pour tout les paires d'état-action visitées dans l'épisode avec la formule suivante

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t - Q(S_t, A_t)]$$

Problème : Cela fonctionne seulement lorsqu'il y a des épisodes et la variance des échantillons est très haute (effet papillon).

# Évaluation Model-free

Si on ne connais pas $p(s', r|s, a)$, on va approximer $q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a]$ en échantillonant des $R_{t+1} + \gamma V(S_{t+1})$ ("*Temporal Difference target*").

## Évaluation de $q_\pi$ TD

1. Commencer avec une approximation $Q(s, a)$ de $q_\pi(s, a)$
2. Répéter pour toujours
   - 2.1 Choisir une action en suivant la stratégie $\pi$. Recevoir une récompense $R_{t+1}$ et un état $S_{t+1}$.
   - 2.2 Mettre à jour $Q(s, a)$ avec la formule suivante

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - Q(S_t, A_t)]$$

# Évaluation Model-free

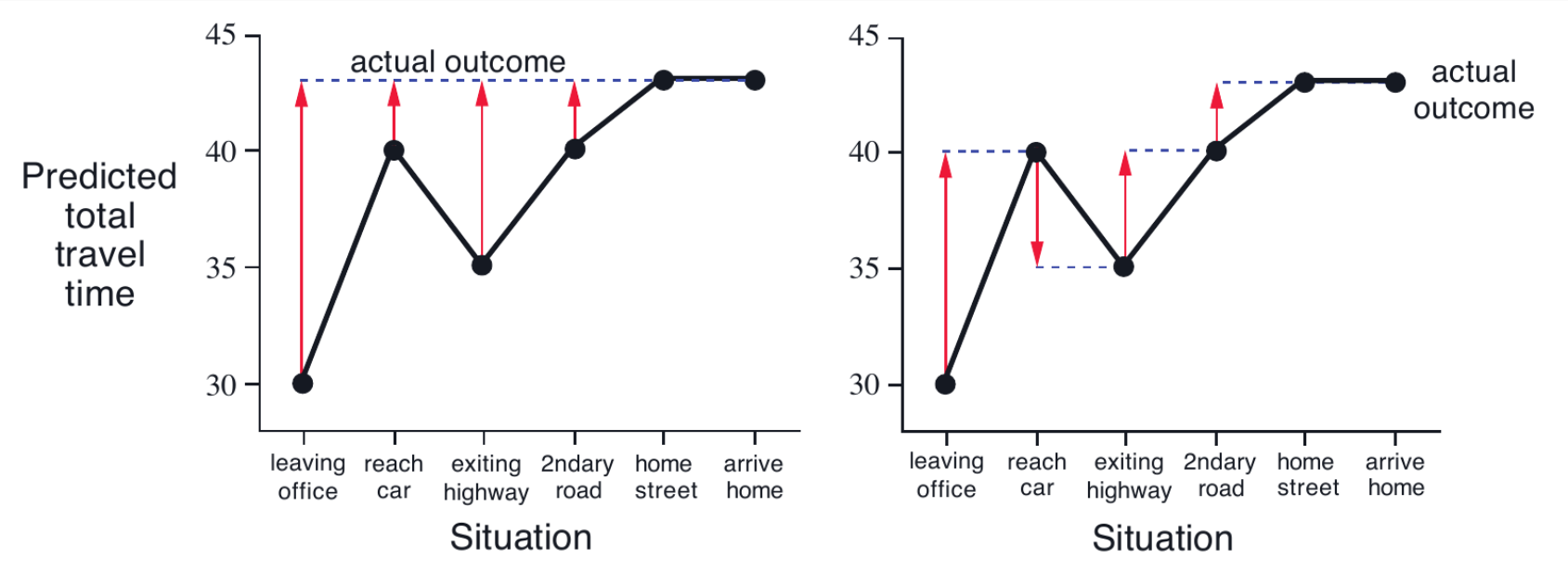


**Figure 8:** Différence entre échantillioner $G_t$ et $R_{t+1} + v_\pi(S_{t+1})$

# Relation entre TD-leaning et Monte-Carlo

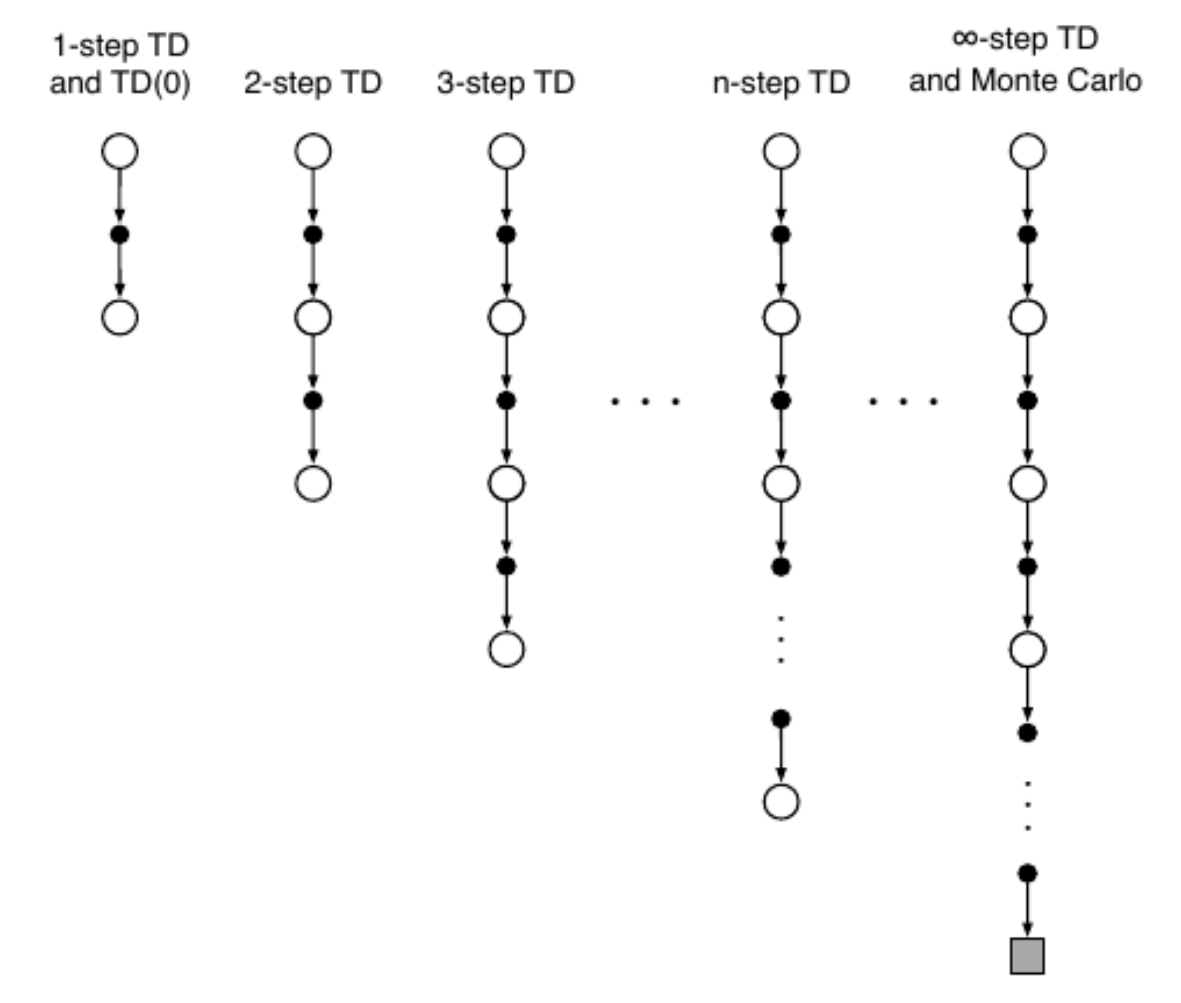


**Figure 9:** Relation entre TD-leaning et Monte-Carlo

# Évalutation de stratégie

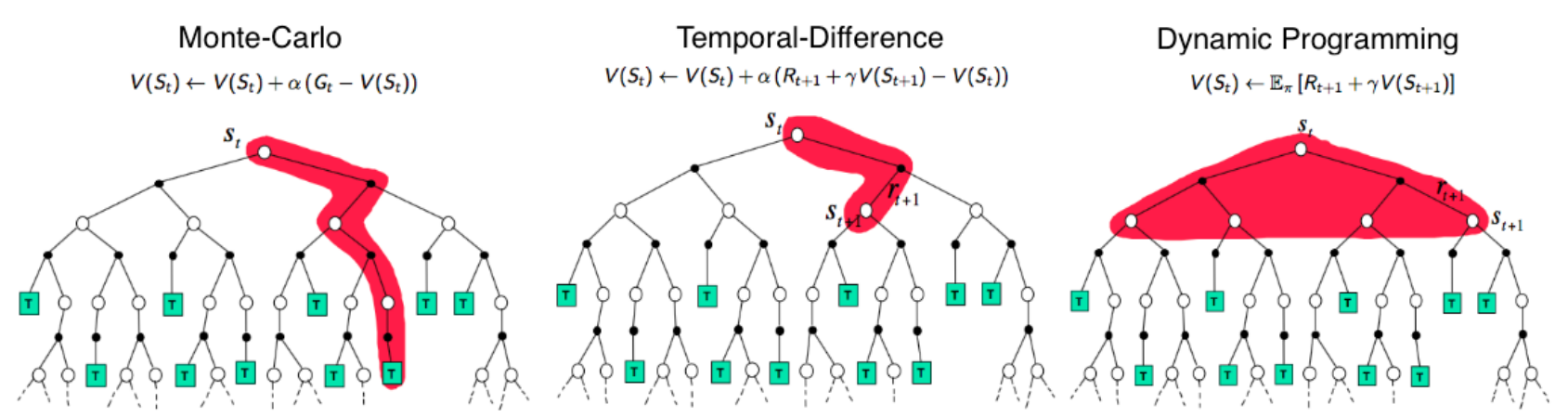


**Figure 10:** Monte-Carlo vs Temporal-Difference vs Dynamic Programming

# Itération de stratégies

On répéter successivement les opération d'évaluation (E) et amélioration (A) pour obtenir une séquence de stratégies de plus en plus efficaces. Puisqu'il y a un nombre fini de stratégies on converge en un nombre fini d'opérations.

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} q_*$$

# Itération de stratégies

On répéter successivement les opération d'évaluation (E) et amélioration (A) pour obtenir une séquence de stratégies de plus en plus efficaces. Puisqu'il y a un nombre fini de stratégies on converge en un nombre fini d'opérations.

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} q_*$$

Pas super efficace.

# Algorithmes efficaces

# Itération de stratégie généralisée

L'itération de stratégie généralisée est une famille d'algorithmes qui combine l'évaluation et l'amélioration en un processus itératif efficace.

## Itération de stratégie généralisée

1. Commencer avec une approximation $Q(s, a)$ de $q_\pi(s, a)$
2. Répéter pour toujours
   - 2.1 Choisir une action en suivant la stratégie $\pi'(s, a) = \arg\max_a Q(S_t, a)$. Recevoir une récompense $R_{t+1}$ et un état $S_{t+1}$.
   - 2.2 Mettre à jour $Q(s, a)$ avec la formule suivante

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - Q(S_t, A_t)]$$

# Itération de stratégie généralisée

L'itération de stratégie généralisée est une famille d'algorithmes qui combine l'évaluation et l'amélioration en un processus itératif efficace.

## Itération de stratégie généralisée (SARSA)

1. Commencer avec $Q(s, a)$ aléatoire
2. Répéter pour toujours
   - 2.1 Choisir l'action $A_t = \arg\max_a Q(S_t, a)$. Recevoir une récompense $R_{t+1}$ et un état $S_{t+1}$.
   - 2.2 Mettre à jour $Q(s, a)$ avec la formule suivante

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - Q(S_t, A_t)]$$

# Q-learning

Un autre algorithme très semblable (et un peu plus populaire) est le Q-learning

## Q-learning

1. Commencer avec $Q(s, a)$ aléatoire
2. Répéter pour toujours
   - 2.1 Choisir l'action $A_t = \arg\max_a Q(S_t, a)$. Recevoir une récompense $R_{t+1}$ et un état $S_{t+1}$.
   - 2.2 Mettre à jour $Q(s, a)$ avec la formule suivante

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

# Deep Q-learning

Le GROS problème avec les algorithmes qu'on a vu dans cette présentation est que nous estimons $q_\pi$ pour chaque élément de $|\mathcal{S}| \times |\mathcal{A}|$.

- La plupart des problèmes sont complètement hors de portée car $|\mathcal{S}|$ est beaucoup trop grand
  - Tic-tac-toe : $765$
  - Échecs : $\approx 10^{43}$
  - Go : $\approx 10^{170}$
  - Nimporte quel environnement continu : $\infty$
- L'agent doit apprendre indépendamment $q_\pi$ pour chaque état même si certains se ressemblent beaucoup.

# Deep Q-learning

**Solution : Deep learning**

Au lieu de définir $q_\pi$ en chaque points on utilise un réseau de neuronnes pour apprendre la fonction. Fun begins!