TD-Learning, Q-Learning et Équations de Bellman

Rhouma Haythem

Avril 2025

1 Introduction à l'apprentissage par renforcement

L'apprentissage par renforcement (Reinforcement Learning, ou RL) est un domaine de l'intelligence artificielle dans lequel un agent apprend à interagir avec un environnement dans le but de maximiser des récompenses cumulées.

Cadre général

Le processus de prise de décision dans un environnement est souvent modélisé comme un processus de décision de Markov (MDP), défini par :

- Un ensemble d'états possibles S
- Un ensemble d'actions A
- Une fonction de transition $P(s'|s, a)$ donnant la probabilité d'atteindre l'état s' depuis s après action a
- Une fonction de récompense $R(s, a)$
- Un facteur d'actualisation $\gamma \in [0, 1]$

Boucle d'apprentissage

À chaque instant t, l'agent :

1. Observe un état S_t
2. Choisit une action A_t
3. Reçoit une récompense R_{t+1}
4. Passe à un nouvel état S_{t+1}

L'agent répète ce processus en essayant d'apprendre une stratégie (ou *politique*) optimale π qui maximise les récompenses à long terme.

Objectif du RL

L'objectif fondamental est de maximiser la somme des récompenses futures actualisées :

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Pour cela, on définit deux fonctions importantes :

- La fonction de valeur d'un état $V^\pi(s)$: espérance du retour total en suivant π à partir de s
- La fonction de valeur d'une paire état-action $Q^\pi(s, a)$

Ce cours explore les méthodes principales permettant à l'agent d'estimer ces fonctions, notamment :

- Les méthodes TD(n)
- Le Q-Learning
- Les équations de Bellman

Nous commençons par la plus simple : TD(0).

2 TD(0) : Mise à jour immédiate après chaque étape

La méthode TD(0), ou *Temporal Difference Learning de degré 0*, est l'une des méthodes les plus simples et les plus fondamentales en apprentissage par renforcement. Elle combine deux idées clés :

- L'apprentissage à partir de l'expérience directe (comme dans Monte Carlo)
- La mise à jour basée sur une estimation actuelle (comme dans la programmation dynamique)

Formule de mise à jour

$$V(S_t) \leftarrow (1 - \alpha)V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1})\right] \tag{1}$$

Cette équation indique que la nouvelle valeur de l'état S_t est un compromis entre :

- L'ancienne valeur de $V(S_t)$, pondérée par $(1 - \alpha)$
- Une nouvelle estimation : la récompense immédiate R_{t+1} plus la valeur estimée de l'état suivant, pondérée par α

Interprétation des termes

- α : taux d'apprentissage (entre 0 et 1). Plus α est grand, plus l'agent donne du poids à la nouvelle information.
- γ : facteur d'actualisation. Permet de donner plus ou moins d'importance aux récompenses futures.
- $V(S_t)$: valeur de l'état actuel à mettre à jour.
- R_{t+1} : récompense reçue immédiatement après l'action.
- $V(S_{t+1})$: valeur estimée de l'état futur.

Intuition

L'idée est de corriger la prédiction $V(S_t)$ en la rapprochant de ce qu'on observe réellement : une récompense immédiate + la valeur estimée du futur. On appelle cela l'**erreur temporelle (TD error)** :

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

Ainsi, la mise à jour devient simplement :

$$V(S_t) \leftarrow V(S_t) + \alpha \cdot \delta_t$$

Avantages

- Mise à jour simple, rapide, et peu coûteuse
- Fonctionne en ligne, sans attendre la fin d'un épisode
- Converge vers des estimations correctes sous certaines conditions (valeurs moyennes sous une politique fixe)

Limites

- Ne prend en compte qu'une seule étape future (vision très locale)
- Estimation sensible au choix de α et γ
- Ne convient pas toujours à des environnements avec des récompenses différées à long terme

Exemple illustratif

Supposons que l'agent soit dans l'état s_1, avec une valeur initiale $V(s_1) = 10$, qu'il reçoive une récompense $R = 5$, et que la valeur estimée de l'état suivant s_2 soit $V(s_2) = 20$, avec $\alpha = 0.1$ et $\gamma = 0.9$.

$$\delta_t = 5 + 0.9 \times 20 - 10 = 13$$
$$V(s_1) \leftarrow 10 + 0.1 \times 13 = 11.3$$

L'agent augmente sa prédiction $V(s_1)$, car le retour observé est meilleur que prévu.

Résumé

La méthode TD(0) constitue une base essentielle pour comprendre les variantes plus avancées comme TD(n), Q-Learning et SARSA. Elle repose sur un principe de correction incrémentale fondé sur une estimation partielle du futur.

3 TD(n) : Mise à jour avec plusieurs étapes futures

La méthode TD(n) est une généralisation de TD(0). Alors que TD(0) met à jour la valeur d'un état en utilisant uniquement la récompense immédiate et la valeur du prochain état, TD(n) prend en compte plusieurs étapes futures. On parle alors de méthode à n-pas (ou n-steps).

Principe général

L'idée est d'utiliser non plus uniquement la première récompense R_{t+1}, mais aussi $R_{t+2}, R_{t+3}, \ldots, R_{t+n}$, ainsi que la valeur estimée de l'état S_{t+n}. Cela permet une mise à jour plus précise, au prix d'une complexité accrue.

La formule générale de TD(n) est :

$$V(S_t) \leftarrow (1-\alpha)V(S_t) + \alpha \left[\sum_{k=1}^{n} \gamma^{k-1} R_{t+k} + \gamma^n V(S_{t+n}) \right] \tag{2}$$

Chaque récompense est pondérée par un facteur γ^{k-1}, et la valeur du dernier état est actualisée avec γ^n.

Exemples concrets

TD(1)

$$V(S_t) \leftarrow (1-\alpha)V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) \right] \tag{3}$$

C'est identique à TD(0), car on regarde une seule étape (récompense immédiate + 1 état estimé).

TD(2)

$$V(S_t) \leftarrow (1-\alpha)V(S_t) + \alpha\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})\right] \tag{4}$$

TD(3)

$$V(S_t) \leftarrow (1-\alpha)V(S_t) + \alpha\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3})\right] \tag{5}$$

TD(4)

$$V(S_t) \leftarrow (1-\alpha)V(S_t) + \alpha\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \gamma^4 V(S_{t+4})\right] \tag{6}$$

Interprétation pédagogique

TD(n) est un **compromis entre TD(0) et Monte Carlo** :

- TD(0) : rapide, mais peu précis (regarde une seule étape)
- Monte Carlo : très précis mais nécessite d'attendre la fin de l'épisode
- TD(n) : intermédiaire, permet un contrôle plus fin sur la profondeur de l'estimation

Avantages

- Améliore la précision de la mise à jour
- Offre plus de souplesse qu'un apprentissage strictement pas-à-pas
- Peut réduire la variance par rapport à Monte Carlo

Inconvénients

- Plus coûteux en mémoire et en calcul (il faut stocker plusieurs récompenses)
- Plus sensible aux erreurs d'estimation sur les états futurs
- Difficile à choisir n de manière optimale

Conclusion intermédiaire

La méthode TD(n) généralise les approches par différences temporelles en tenant compte de plusieurs étapes dans le futur. Elle constitue une avancée importante pour apprendre dans des environnements où les récompenses ne sont pas immédiates mais différées.

4 Q-Learning : Apprentissage des paires état-action (Off-policy)

Q-Learning est une méthode d'apprentissage par renforcement sans modèle, qui permet d'estimer la valeur optimale de chaque paire (s, a), c'est-à-dire le gain attendu en prenant l'action a dans l'état s, puis en suivant la meilleure politique possible.

Contrairement à TD(0) ou TD(n), Q-Learning ne met pas à jour la valeur d'un état, mais celle d'une **paire état-action**. C'est donc une méthode basée sur la fonction $Q(s, a)$.

Formule de mise à jour

$$Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a)\right] \tag{7}$$

Cette équation repose sur une idée fondamentale : pour estimer la valeur d'un couple (S_t, A_t), on regarde la récompense immédiate R_{t+1} et la meilleure action possible dans l'état suivant S_{t+1}, en prenant le maximum des valeurs $Q(S_{t+1}, a)$.

Composants de l'équation

- $Q(S_t, A_t)$: valeur estimée de la paire état-action courante
- α : taux d'apprentissage
- R_{t+1} : récompense obtenue en passant de S_t à S_{t+1}
- γ : facteur de remise (ou d'actualisation)
- $\max_a Q(S_{t+1}, a)$: meilleure valeur d'action estimée dans l'état suivant

Pourquoi Q-Learning est-il *off-policy* ?

On dit que Q-Learning est **off-policy** car :

- L'agent peut suivre une politique (par exemple ε-greedy pour explorer),
- Mais la mise à jour utilise toujours la valeur de la **meilleure action possible**, qu'on l'ait suivie ou non.

Cela signifie que l'agent apprend en se basant sur une politique **qu'il n'applique pas forcément** pendant son exploration. Il évalue les meilleures options, même s'il ne les choisit pas systématiquement.

Exemple illustratif

Supposons que :

- $Q(S_t, A_t) = 10$
- $R_{t+1} = 4$
- $\max_a Q(S_{t+1}, a) = 20$
- $\alpha = 0.1$, $\gamma = 0.9$

Alors :

$$Q(S_t, A_t) \leftarrow (1 - 0.1) \cdot 10 + 0.1 \cdot (4 + 0.9 \cdot 20) = 9 + 0.1 \cdot (4 + 18) = 9 + 2.2 = 11.2$$

Avantages

- Converge vers une politique optimale même si l'environnement est inconnu
- Fonctionne sans avoir besoin d'un modèle de l'environnement
- S'adapte à des environnements complexes et changeants

Inconvénients

- Plus coûteux en mémoire (il faut stocker $Q(s, a)$ pour toutes les paires)
- Nécessite un bon compromis entre exploration et exploitation (par exemple via ε-greedy)
- Peut diverger si les paramètres α et γ sont mal choisis

Remarque finale

Q-Learning est l'un des algorithmes les plus utilisés en pratique pour résoudre des problèmes de type MDP. Il constitue une étape essentielle vers des méthodes plus avancées comme Deep Q-Learning.

5 Pourquoi reformuler les équations avec $(1 - \alpha)$?

De nombreuses versions des équations d'apprentissage par renforcement sont présentées sous la forme :

$$Q(s, a) \leftarrow Q(s, a) + \alpha \cdot \delta \tag{8}$$

où δ est l'erreur temporelle (ou TD error). Cependant, il est souvent pédagogique de reformuler cette mise à jour comme une combinaison pondérée entre l'ancienne valeur et la nouvelle estimation :

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \cdot \text{estimation} \tag{9}$$

Cette reformulation est mathématiquement équivalente, mais elle **met en évidence le rôle de chaque composante** dans la mise à jour.

1. Visualiser l'impact de l'apprentissage

En écrivant :

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left[R + \gamma \max_{a'} Q(s',a')\right]$$

on comprend immédiatement que la mise à jour est une **moyenne pondérée** entre :

- La valeur actuelle ($Q(s,a)$), pondérée par $(1-\alpha)$
- Une nouvelle estimation basée sur la récompense et l'état futur, pondérée par α

Cela clarifie le mécanisme d'apprentissage : **on ajuste la valeur actuelle sans l'écraser complètement**.

2. Interpréter α comme un facteur de confiance

- Si α est petit (par exemple 0.1), on conserve 90% de l'ancienne valeur, et on ajuste doucement.
- Si α est grand (par exemple 0.9), on oublie l'ancienne valeur et on fait confiance à la nouvelle estimation.

Cela donne une **intuitivité importante** pour le réglage du taux d'apprentissage : plus α est élevé, plus l'apprentissage est rapide mais instable.

3. Contrôle de la stabilité de l'apprentissage

En séparant explicitement les deux poids $(1-\alpha)$ et α, on **comprend mieux comment assurer une convergence progressive** :

- Si α est trop grand, l'agent risque de surcorriger ses valeurs à chaque instant.
- Si α est trop petit, l'apprentissage devient trop lent.

4. Approche pédagogique pour les débutants

Cette écriture permet aux étudiants :

- De voir que l'ancienne information n'est pas supprimée, mais intégrée.
- De comprendre que la valeur estimée est une moyenne dynamique.
- D'anticiper le comportement de l'algorithme sans avoir besoin de l'expérimenter d'abord.

Conclusion

La forme $(1 - \alpha) \cdot$ ancienne valeur $+ \alpha \cdot$ nouvelle estimation rend explicite le mécanisme de compromis entre **mémoire** (information passée) et **apprentissage** (information nouvelle). C'est une aide précieuse pour comprendre comment les agents apprennent progressivement, de façon stable et contrôlée.

6 On-policy, Off-policy et N-step : Concepts fondamentaux

Pour bien comprendre les méthodes d'apprentissage par renforcement comme TD, Q-Learning ou Monte Carlo, il est essentiel de saisir trois notions fondamentales :

- On-policy : apprendre en suivant sa propre politique
- Off-policy : apprendre en observant une autre politique
- N-step : apprendre en regardant plusieurs étapes dans le futur

1. Méthode On-policy

Définition : Une méthode est dite **on-policy** si elle apprend en suivant la même politique qu'elle cherche à améliorer.

- L'agent suit une politique π pour choisir ses actions.
- Les estimations V^π ou Q^π sont mises à jour en fonction de cette même politique.

Exemple : Si l'agent choisit ses actions en utilisant une stratégie ε-greedy, alors les mises à jour sont basées sur les récompenses obtenues en suivant cette stratégie.

Avantage : Alignement entre les choix faits et les apprentissages.

Inconvénient : Moins efficace si la politique explorée n'est pas optimale.

2. Méthode Off-policy

Définition : Une méthode est dite **off-policy** si elle apprend à partir d'une politique différente de celle qu'elle utilise pour agir.

- L'agent peut explorer en suivant une politique π_b (par exemple : exploration aléatoire),
- mais il met à jour ses valeurs en estimant ce qu'il se passerait avec une autre politique π^* (par exemple : la politique optimale).

Exemple : Le Q-Learning est off-policy car, même si l'agent explore de manière non optimale, il met à jour $Q(s, a)$ en utilisant $\max_{a'} Q(s', a')$, ce qui suppose une politique optimale.

Avantage : Permet d'apprendre des stratégies optimales tout en explorant.

Inconvénient : Plus difficile à stabiliser.

3. Méthode N-step

Définition : Une méthode est dite **N-step** si elle met à jour ses estimations à partir de plusieurs étapes futures.

- L'agent attend n étapes avant de mettre à jour la valeur d'un état.
- Cela permet de capturer des informations plus complètes que TD(0).

Exemple : TD(2) utilise les deux prochaines récompenses et la valeur estimée de l'état à $t + 2$.

Avantage : Meilleur compromis entre rapidité et précision.

Inconvénient : Plus de mémoire nécessaire et calcul plus complexe.

4. Tableau récapitulatif

Méthode	**Type**	**Principe**	**Exemple**
On-policy	Même politique utilisée et apprise	Apprendre en suivant π	SARSA, TD(0), TD(n)
Off-policy	Politique d'exploration $\neq$ politique apprise	Apprendre en suivant π^* tout en explorant	Q-Learning
N-step	Basé sur plusieurs étapes dans le futur	Intègre n récompenses + valeur estimée	TD(2), TD(3), TD(n)

5. Analogie simplifiée

On-policy : Tu joues à un jeu avec ta propre stratégie. Tu apprends de ce que tu fais toi-même.

Off-policy : Tu observes un autre joueur (meilleur) et tu apprends de ce qu'il aurait fait à ta place.

N-step : Avant de décider si une action était bonne, tu attends plusieurs coups et observes les résultats avant d'apprendre.

Conclusion

Ces concepts structurent la façon dont un agent interagit avec l'environnement et apprend. La compréhension de ces approches est cruciale pour choisir la bonne méthode selon les contraintes : données disponibles, besoins d'exploration, précision souhaitée, mémoire disponible, etc.

7 Équation de Bellman : Fondement théorique de l'apprentissage par renforcement

L'équation de Bellman est l'un des piliers de l'apprentissage par renforcement. Elle permet de **décomposer un problème de prise de décision à long terme** en sous-problèmes plus simples, basés sur des relations récursives.

1. Idée principale

Elle exprime la **valeur d'un état** (ou d'une action) comme étant la **récompense immédiate** plus la **valeur attendue de l'état futur**. En d'autres termes, elle définit la valeur d'un état à partir des valeurs des états qui suivent.

2. Équation de Bellman pour la valeur d'un état sous une politique π

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma V^{\pi}(S_{t+1}) \mid S_t = s\right] \tag{10}$$

Cette équation dit : "la valeur d'un état s sous la politique π est la moyenne de la récompense reçue plus la valeur actualisée de l'état suivant, si l'on suit π."

3. Équation de Bellman pour la valeur d'une paire état-action

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma Q^{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right] \tag{11}$$

Elle est similaire à l'équation sur V^{π}, mais prend en compte l'action a exécutée dans l'état s. Elle est utilisée pour les méthodes de type Q-Learning ou SARSA.

4. Forme optimale : équation de Bellman pour $V^*(s)$

Lorsqu'on veut apprendre la **politique optimale**, on introduit la **valeur optimale d'un état**, notée $V^*(s)$. Elle est définie comme la meilleure valeur atteignable en choisissant toujours la meilleure action à chaque étape :

$$V^*(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma V^*(S_{t+1}) \mid S_t = s, A_t = a\right] \tag{12}$$

Cela signifie : parmi toutes les actions possibles, on choisit celle qui maximise la récompense attendue plus la valeur future.

5. Forme optimale pour les paires état-action

$$Q^*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') \mid S_t = s, A_t = a\right] \tag{13}$$

Ici, on estime la valeur optimale d'une paire état-action. C'est cette version qui est utilisée par l'algorithme de Q-Learning.

6. Nécessité d'un modèle

L'équation de Bellman, dans sa forme exacte, suppose que l'on connaît :

- Les probabilités de transition $P(s'|s,a)$
- Les récompenses attendues $R(s,a)$

C'est pourquoi on dit qu'elle nécessite un **modèle de l'environnement**. Elle est souvent utilisée pour construire des algorithmes de programmation dynamique.

7. Usage pratique

Même si l'équation de Bellman est rarement utilisée directement en pratique (car on ne connaît pas toujours les transitions), elle est la **base théorique** de nombreuses méthodes :

- TD-Learning cherche à approximer $V^\pi(s)$ selon Bellman
- Q-Learning cherche à approximer $Q^*(s,a)$ en se basant sur l'estimation empirique de cette équation
- Les méthodes comme Value Iteration ou Policy Iteration en dépendent directement

8. Interprétation

Bellman dit simplement :

> "Ce que vaut un état aujourd'hui dépend de ce que je peux en tirer maintenant (récompense immédiate) + ce qu'il m'apportera demain (valeur de l'état suivant)."

C'est une vision récursive du raisonnement, similaire à :

> "Un bon investissement est un placement qui rapporte un peu maintenant et encore plus à l'avenir."

Conclusion

L'équation de Bellman structure mathématiquement le raisonnement d'un agent intelligent qui pense à long terme. Elle est fondamentale dans l'analyse, la conception et l'implémentation des algorithmes d'apprentissage par renforcement.

8 Comparaison approfondie : TD-Learning, Q-Learning, Monte Carlo et Bellman

Il est essentiel de bien distinguer les principales familles de méthodes utilisées en apprentissage par renforcement. Elles partagent des objectifs similaires (estimer des fonctions de valeur), mais diffèrent profondément sur plusieurs aspects fondamentaux, tels que :

- Leur dépendance à un modèle
- Leur stratégie d'estimation
- Le moment où la mise à jour a lieu
- Le type de politique suivie

1. Tableau comparatif global

Méthode	Politique	Modèle requis	Moment de la mise à jour	Forme de la mise à jour
Bellman (exact)	On-policy	Oui	Théorique / analytique	Recursion exacte
Monte Carlo	On-policy	Non	Fin d'épisode	Moyenne sur tous les retours
TD-Learning (TD(0), TD(n))	On-policy	Non	En ligne (à chaque pas)	Estimation + bootstrap
Q-Learning	Off-policy	Non	En ligne (à chaque pas)	Maximisation + bootstrap

2. Détails et explications par méthode

Bellman (formule exacte)

- Utilisé en programmation dynamique
- Requiert la connaissance exacte de l'environnement (transitions et récompenses)
- Utilisé pour calculer les valeurs optimales $V^*(s)$ ou $Q^*(s, a)$

Monte Carlo

- Ne nécessite aucun modèle
- Attend la fin de l'épisode pour mettre à jour la valeur d'un état
- Fournit une estimation exacte du retour total
- Fonctionne bien si les épisodes sont courts et terminent toujours

TD-Learning

- Utilise le principe du bootstrap : mise à jour avec estimation intermédiaire
- Plus rapide et plus efficace que Monte Carlo dans des environnements avec épisodes longs ou infinis
- Ne nécessite pas d'attendre la fin de l'épisode
- Inclut des variantes comme TD(0), TD(n), SARSA

Q-Learning

- Ne suit pas forcément la politique utilisée pour apprendre (off-policy)
- Permet de converger vers une politique optimale tout en explorant
- Très utilisé dans des environnements complexes sans modèle
- Peut être instable si mal paramétré (exploration, α, γ)

3. Représentation visuelle simplifiée

Approche	**Exemple de stratégie**
Théorique	Bellman : exact, modèle requis
Empirique + retardée	Monte Carlo : attend la fin d'un épisode
Empirique + immédiate	TD(0), TD(n) : mise à jour à chaque étape
Empirique + optimisante	Q-Learning : mise à jour vers la meilleure action future

Conclusion

Chaque méthode a ses avantages :

- Bellman est une base théorique puissante, mais peu utilisable sans modèle.
- Monte Carlo est simple à implémenter mais lent.
- TD est efficace et plus pratique.
- Q-Learning est très puissant pour l'apprentissage de politiques optimales.

Le choix de la méthode dépend du problème à résoudre, de la disponibilité d'un modèle, du besoin de précision, de la vitesse requise, et des ressources disponibles.

9 Méthodes avec ou sans modèle : Quelle différence fondamentale ?

En apprentissage par renforcement, on distingue deux grandes familles d'approches selon qu'elles nécessitent ou non un modèle de l'environnement :

- **Méthodes avec modèle (model-based)**
- **Méthodes sans modèle (model-free)**

Comprendre cette différence est essentiel pour choisir la bonne méthode selon les contraintes du problème réel.

1. Qu'est-ce qu'un modèle ?

Un modèle de l'environnement est la connaissance explicite de :

- La fonction de transition $P(s'|s, a)$: donne la probabilité d'atteindre l'état s' après action a dans s
- La fonction de récompense $R(s, a)$: donne la récompense attendue pour chaque action

Avec un modèle, on peut simuler l'environnement à volonté, sans avoir besoin d'y interagir réellement.

2. Méthodes avec modèle (Model-based)

- Utilisent directement $P(s'|s, a)$ et $R(s, a)$
- Permettent de faire des *simulations internes* sans agir dans le vrai monde
- Exemples : Value Iteration, Policy Iteration, Dyna-Q

Avantages :

- Convergence plus rapide
- Possibilité de planification et de recherche de stratégies avant d'agir

Inconvénients :

- Requiert une connaissance complète et précise des dynamiques de l'environnement
- Pas toujours applicable dans des environnements réels inconnus ou trop complexes

3. Méthodes sans modèle (Model-free)

- Ne nécessitent pas de connaître les transitions ni les récompenses à l'avance
- L'agent apprend uniquement à partir de ses propres expériences
- Exemples : TD(0), TD(n), Q-Learning, Monte Carlo

Avantages :

- Très flexibles : s'adaptent à n'importe quel environnement, même inconnu
- Applicables directement sur des systèmes réels (robot, jeu, interface)

Inconvénients :

- Apprentissage plus lent (car empirique)
- Moins de possibilités de planification

4. Comparaison synthétique

Critère	**Avec modèle**	**Sans modèle**
Connaissance des transitions	Requise	Non
Connaissance des récompenses	Requise	Apprise par expérience
Vitesse de convergence	Rapide	Plus lente
Applicabilité en pratique	Limitée aux systèmes connus	Très large
Possibilité de simulation mentale	Oui	Non

5. Illustration simplifiée

Avec modèle : Tu joues aux échecs en ayant une carte qui te dit exactement où chaque pièce peut aller, et tu simules mentalement plusieurs coups.

Sans modèle : Tu joues aux échecs sans connaître les règles exactes, tu bouges les pièces, observes les conséquences, et apprends avec le temps ce qui fonctionne.

Conclusion

Les méthodes sans modèle (comme Q-Learning ou TD) sont aujourd'hui les plus utilisées dans les environnements complexes et inconnus (ex. : robotique, jeux vidéo, simulations dynamiques). Les méthodes avec modèle restent puissantes mais nécessitent des connaissances structurelles qui ne sont pas toujours disponibles en pratique.

10 Conclusion générale

Ce cours vous a permis d'explorer les fondements de l'apprentissage par renforcement, à travers les concepts clés suivants :

- TD-Learning (TD(0), TD(n)) : apprentissage par estimation incrémentale, sans attendre la fin de l'épisode
- Q-Learning : méthode off-policy puissante pour trouver des stratégies optimales
- Équation de Bellman : base théorique pour comprendre la valeur des états et des actions
- Méthodes avec et sans modèle : deux approches fondamentales selon la connaissance de l'environnement

1. Synthèse des méthodes étudiées

Méthode	**On/Off-policy**	**Modèle requis**	**Fréquence de mise à jour**	**Spécificité**
TD(0)	On-policy	Non	À chaque pas	Rapide, peu coûteux, court terme
TD(n)	On-policy	Non	Toutes les n étapes	Plus précis, compromis temporel
Monte Carlo	On-policy	Non	Fin d'épisode	Estimation complète
Q-Learning	Off-policy	Non	À chaque pas	Apprend une stratégie optimale
Bellman (exact)	On-policy	Oui	Théorique / planification	Base mathématique

2. Conseils pédagogiques pour retenir les différences

- **TD(0)** : pense à une personne qui apprend au fur et à mesure, avec peu de mémoire.
- **TD(n)** : même personne, mais qui garde quelques souvenirs pour mieux juger.
- **Monte Carlo** : observe l'ensemble de l'expérience avant d'apprendre quoi que ce soit.
- **Q-Learning** : veut apprendre la meilleure façon de faire, même s'il expérimente autre chose.
- **Bellman** : connaît parfaitement l'environnement et calcule de manière analytique.

3. Choisir la bonne méthode

Le choix dépend de :

- La disponibilité d'un modèle (connaissances complètes ?)
- La longueur des épisodes (courts ou infinis ?)
- Le besoin d'optimisation à long terme ou non
- Les ressources de calcul disponibles (temps réel ou offline ?)

4. Pour aller plus loin

Ce cours est une fondation. Les prochaines étapes sont :

- SARSA : une alternative on-policy à Q-Learning
- Deep Q-Learning : combiner Q-Learning avec les réseaux de neurones
- Actor-Critic : séparer politique et valeur
- Méthodes de planification : Dyna-Q, Prioritized Sweeping

5. Message final au lecteur

L'apprentissage par renforcement n'est pas seulement un domaine mathématique. C'est une manière de comprendre comment des agents – humains, animaux ou machines – prennent des décisions intelligentes dans un monde incertain. Ce que vous venez d'apprendre vous servira à bâtir non seulement des algorithmes, mais aussi des stratégies d'exploration, d'adaptation et de raisonnement à long terme.