Apprentissage par renforcement

Emmanuel Franck avec l'aide de : L. Navoret, V. Vigon, L. Bois, Y. Privat et C. Courtès

Introduction

Apprentissage automatique

Branche de l'informatique et des mathématiques qui porte sur la construction de modèles paramétriques à partir de données. Modèles: déterministe ou aléatoire:

$$y = f_\theta(x), \quad \mathbb{P}_\theta(y \mid x)$$

- Type d'apprentissage:
  - **Supervisé**: on connaît un certain nombre de couple entrée/sortie $((x,y)_1, ..., (x,y)_n)$ et on les utilise pour ajuster le modèle.
  - **Non supervisé**: On ne connaît pas d'exemples des sorties y. L'apprentissage est fait à partir de l'analyse des données $(x_1, .., x_n)$.
  - **Par renforcement**: Apprentissage par un agent autonome à travers un processus: essai-erreur + récompense.

Apprentissage par renforcement

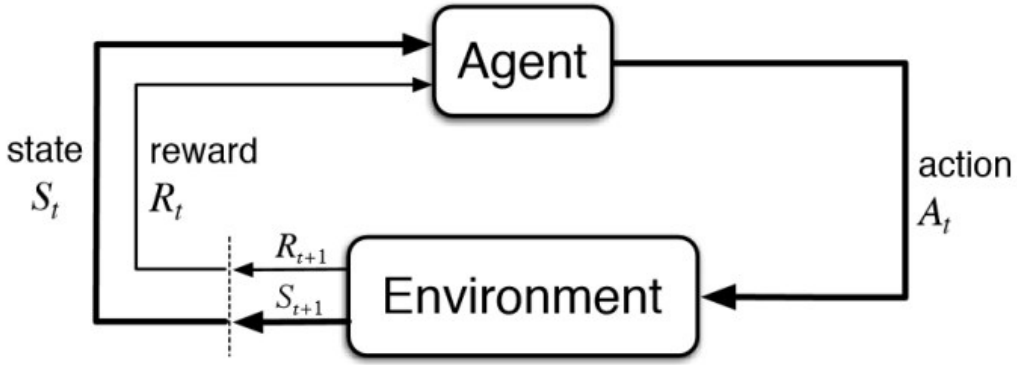
Il s'agit de contrôle optimal stochastique couplé avec de l'apprentissage.

Applications

- Robotique, véhicules autonomes, optimisatio de production, de traitement médical,
- IA pour les jeux: échec, Go et récemment Starcraft.
- EDP/physique ? Nouveau modèle de turbulence: paper (Nature machine intelligence)

Cadre théorique

Principe



- **Etat**: s_t, état courant de l'agent (ex: un échiquier),
- **Action**: a_t, action qui, en intéraction avec l'environnement modifie l'état de l'agent (ex: un mouvement de pièce).
- **Environnement**: il décrit comme une action génère l'état suivant et la recompense (mouvement de l'adversaire aux échecs),
- **Récompense**: r_{t+1}, évalue l'action (aux échecs: pas de récompense pendant la partie, une récompense en cas de victoire).

Plan

- **Partie I**: problèmes de petites tailles, espace des états/actions finis et pas trop grand.
- **Partie II**: problèmes de grandes tailles, espace des états/actions finis et grand où infinis.

Cadre théorique: processus de décision Markovien I

Processus de décision

Un Processus de décision est définit par le quadruplet $(S, A, P(.), r(.))$:

- S est l'espace d'états dans lequel évolue le processus ;
- A est l'espace des actions qui contrôlent la dynamique de l'état ;
- $P(. \mid ., .) : S \times A \times S \rightarrow [0, 1]$ sont les probabilités de transition entre les états;
- $r(., .) : S \times A \rightarrow \mathbb{R}$ est la fonction de récompense sur les transitions entre les états.

- On parle de **modèle parfait** si $P(., . \mid .)$ et $r(., .)$ sont connus.
- **Enjeu central** (en robotique): Trouver des méthodes qui ne connaissance pas le modèle mais interargissent avec.

Processus de décision Markovien

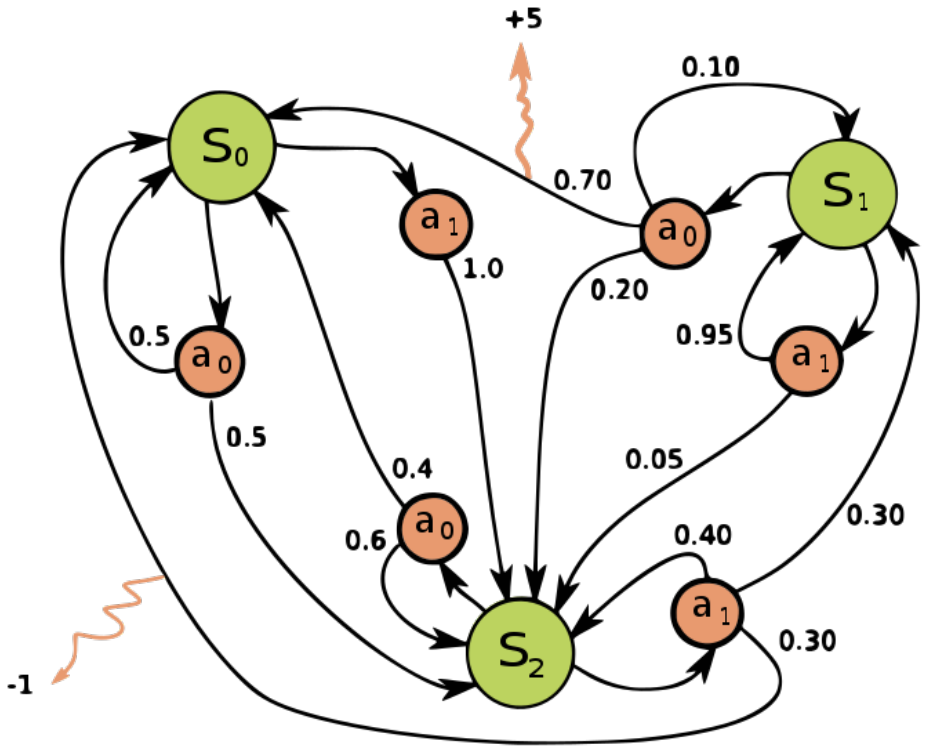
Un processus de décision est dit Markovien si

$$\mathbb{P}(s_{t+1} \mid a_0, s_0, ..., a_t, s_t) = \mathbb{P}(s_{t+1} \mid a_t, s_t)$$

- En pratique P est donc une loi de probabilité des états s' atteignable en partant de s avec l'action a.

Cadre théorique: processus de décision Markovien II

- Exemple de PDM stochastique: pas un seul état associé a un couple état/action.



- Ici: 3 états, 2 actions, 2 récompenses possibles pour certaines transitions.

Processus de décision Markovien: exemples

- **Maintenance de stock**: Le responsable d'un entrepôt dispose d'un stock s_t de marchandise. Il doit satisfaire la demande D_t des clients (Processus de Poisson). Action: quantité a_t supplémentaire commandée à son fournisseur. L'entrepôt a une capacité limite M donc $A = \{0, 1, ...M - s_t\}$.
  La dynamique est donnée par

$$s_{t+1} = [s_t + a_t - D_t]^+$$

  L'objectif est de maximiser le profit. On obtient les récompenses:

$$r_t = -C(a_t) - h(s_t + a_t) + f([s_t + a_t - x_{t+1}]^+)$$

  avec coût de stock $h(s)$, un coût de commande $C(a)$ et une fonction de revenu $f(q)$ qui dépend du stock vendu. Critère:

$$\mathbb{E}\left[\sum_{i=1}^{T-1} r_t + g(s_T)\right]$$

- **Contrôle d'EDO/EDS et de chaine de Markov** (modèle épidémique stochastique etc).

Processus de décision Markovien: exemples II

- **Optimisation de traitement du cancer** Traitement multi-modalités (chirurgie, radiothérapie ou chimiothérapie). L'articulation de ces modalités très dépendante du praticien.

  But: automatiser la construction de ces traitements? Ici, il est considéré trois types de traitements:
  - Modalité 1 (M1): fort risque d'effets secondaires sur les tissus sains, efficacité importante (fréquence d'utilisation limitée)
  - Modalité 2 (M2): faible risque d'effets sécondaires sur les tissus sains, efficacité modéré,
  - Modalité 3 (M3) :pas de traitement, baisse de la probabilité d'effets secondaires, progression de la tumeur plus importante.

- Etat: $s_t = (h_t, \phi_t, \tau_t)$ avec:
  - $h_t \in \{0, 1\}$ qui définit l'utilisation de la modalité de TYPE 1 ($h_t = 1$ la modalité a déjà été utilisée).
  - $\phi_t = \{0, .., m\}$ représente l'effet sur les tissus sains (0 pas d'effet, m beaucoup d'effet),
  - $\tau_t = \{0, .., n\}$ représente la progression de la tumeur (0 rémission, n décès du patient).
  - L'espace des action est donné par $a_t \in \{M_1, M_2, M_3\}$.
- **Applications importantes**: IA de jeux (Echec, Starcraft..), Robotique, vehicules autonomes.

Cadre théorique: politique I

Règles Markovienne

On nomme une régle de décision déterministe Markovienne une fonction

$$\mu_t(s_t) : S \to A$$

qui connaissant un état renvoie une action. On nomme une régle de décision stochastique Markovienne une loi de probabilité:

$$\pi_t(a_t \mid s_t) : A \times S \to [0, 1]$$

tel que $\sum_{a \in A} \pi_t(a \mid s_t) = 1$ qui connaissant un état renvoie les probabilités des actions.

Règles Histoire dépendante

On nomme une régle de décision déterministe histoire dépendante une fonction

$$\mu_t(h_t) : S \times A \times ... \times S \times A \to A$$

qui connaissant l'historique h_t des états/actions renvoie une action. On nomme une régle de décision stochastique Histoire-dépendante une loi de probabilité:

$$\pi_t(a_t \mid h_t) : A \times S \times A \times ... \times S \times A \to [0, 1]$$

tel que $\sum_{a \in A} \pi_t(a \mid h_t) = 1$ qui connaissant l'historique renvoie les probabilités des actions.

Cadre théorique: politique II

Politique

On nomme **politique Markovienne/histoire dépendante** une séquence de régle Markovienne/Histoire dépendante:

$$\pi = (\pi_0, \pi_1, ...\pi_t)$$

et une politique stationnaire:

$$\pi = (\pi, \pi, ..., \pi).$$

- Connaissant une politique on peut définir les probabilités de transition:

$$\mathbb{P}^{\pi}(s_{t+1} = s^{'} \mid s_t = s) = \sum_{a \in A} \pi(a \mid s) P(s^{'} \mid a, s)$$

- et la récompense moyenne associée à la politique π:

$$r^{\pi}(s) = \sum_{a \in A} \pi(a \mid s) r(s, a)$$

- **Proposition**: Si π est Markovienne, le triplet $(S, \mathbb{P}^{\pi}, r^{\pi})$ forme une chaine de Markov valuée.
- **Proposition**: Si la politique est stationnaire la chaine de Markov est homogène.

Processus de décision Markovien: cumul espéré

- On est dans un problème de contrôle optimal: trouver la trajectoire (suite d'action) maximisant la récompense.
- **Problème Markovien**: le problème ne depend que de l'état courant.
- Quel critère maximiser: le cumul espéré à partir du temps t (défini à partir de l'état s_t):
  - le cumul espéré fini:

$$G_t = \sum_{k=0}^{T} r_{t+k+1},$$

  - le cumul espéré amorti:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

    avec $\gamma < 1$.
  - le cumul espéré moyen:

$$G_t = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n} r_{t+k+1}.$$

- On définit le cumul espéré à partir de s_t. On pourrait partir d'un état zéro à cause du caractére Markovien.
- En général et pour la suite on utilise le cumul espéré amorti.

Fonction valeur et Q fonction

- **But**: trouver la meilleur action/meilleur politique pour l'agent.
- On doit donc définir des objets pour évaluer une politique/action

La fonction valeur

La fonction valeur associée à une politique π est une fonction $S \to \mathbb{R}$ qui estime le cumul espéré suivant la politique π:

$$V^{\pi}(s) = \mathbb{E}[G_t \mid s_t = s; \pi]$$

La Q fonction

La Q fonction associée à une politique π est une fonction $S \times A \to \mathbb{R}$ qui estime le cumul espéré avec a comme première action puis suivant la politique π:

$$Q^{\pi}(s, a) = \mathbb{E}[G_t \mid s_t = s, a_t = a; \pi]$$

- Lien entre V et Q: $V(s) = \sum_a Q(s, a)\pi(a \mid s)$ (cas déterministe $Q(s, \mu(s)) = V(s)$).
- Les fonctions V et Q permettent d'évaluer les politiques. Elles permettent donc de construire la notion de politique optimale.

Politique optimale I

- Soit Π^{AH}/Π^{AM} l'espace des politiques aléatoires histoire-dépendante/Markovienne. Soit D^{AM}/D^{M} l'espace des politiques aléatoires/déterministes stationnaires Markoviennes.
- Afin de définir une politique optimale. On définit une relation d'ordre: soit $\pi, \pi^{'} \in \Pi^{AM}$ on dit que

$$\pi^{'} < \pi \quad \text{si} \quad \forall s \in S, \quad V^{\pi^{'}}(s) < V^{\pi}(s)$$

Politique optimale

Une politique π^* est optimale si la fonction valeur associée est optimale au sens

$$V^{\pi^*}(s) = V^*(s) = \max_{\pi \in \Pi^{AH}} V^{\pi}(s) \quad \forall s \in S$$

- Remarques:
  - On peut définir la Q fonction optimale aussi:

$$Q^*(s,a) = \max_{\pi \in \Pi^{AH}} Q^{\pi}(s,a) \quad \forall s \in S, a \in A.$$

- **Définition**: On appelle une politique gloutonne la politique donnée par $\mu(s) = \text{argmax}_a Q(s,a)$.

Politique optimale II

Équivalence entre $\Pi^{AM} = \Pi^{AH}$

Soit $\pi^1 \in \Pi^{AH}$. Pour chaque état initial $s \in S$ il existe une politique $\pi_2 \in \Pi^{AM}$ tel que

$$V_{\pi^1}(s) = V_{\pi^2}(s)$$

- Début preuve Soit π^1 la politique histoire-dépendante. On peut définir une politique Markovienne associée:

  $$\pi^2(a_{t+k} = a \mid s_{t+k} = s) = \mathbb{P}^{\pi^1}(a_{t+k} = a \mid s_{t+k} = s, s_t = x), \quad \forall k \geq t, \forall a \in A, s \in S$$

  Par récurrence (on ne détaille pas), on obtient:

  $$\mathbb{P}^{\pi^1}(s_{t+k} = s, a_{t+k} = a \mid s_t = x) = \mathbb{P}^{\pi^2}(s_{t+k} = s, a_{t+k} = a \mid s_t = x)$$

  Or pour une politique π:

  $$V_\pi(x) = \mathbb{E}[\sum_k^\infty \gamma^t r_{t+k+1} \mid s_t = s; \pi] = \sum_k^\infty \gamma^t \mathbb{E}[r_{t+k+1} \mid s_t = s; \pi]$$

  avec

  $$\mathbb{E}[r_{t+k+1} \mid s_t = s; \pi] = \sum_{s \in S} \sum_{a \in A} r(s, a) \mathbb{P}^\pi(s_{t+k} = s, a_{t+k} = a \mid s_t = x)$$

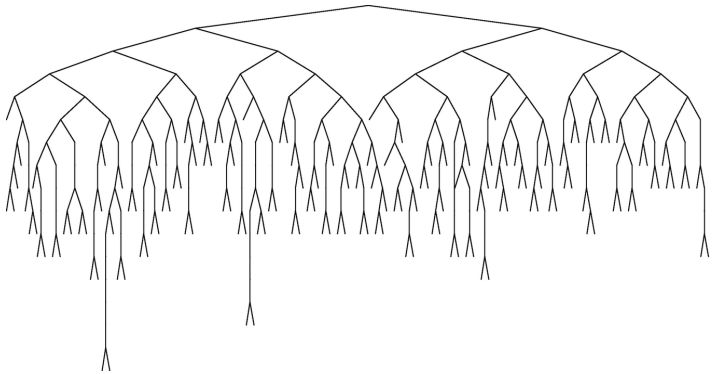
- L'égalité des probabilités permet de conclure associées π_1/π_2. Fin preuve
- **Remarque**: On peut se restreindre à l'ensemble des politiques Markovienne.

Recherche de la meilleure action

- A ce stade deux questions possibles:
  - calculer une trajectoire optimale. Contrôle optimal stochastique en boucle ouverte.
  - calculer une politique optimale (meilleure politique pour l'ensemble des trajectoires/états). Contrôle optimal stochastique en boucle fermée.
- **Meilleure action** à partir d'un état s: action qui maximise l'espérance des récompenses cumulées sur une trajectoire:

$$a^* \in \text{argmax}_a \, Q^*(s, a).$$

- Cela revient à déterminer la meilleure trajectoire (la meilleure action =1ère action).
- Comment la déterminer? on peut voir le problème comme un arbre (stochastique ou déterministe) avec comme poids les récompenses.
- **Meilleure action**: recherche du chemin à poids maximum (algorithme de Dijkstra, algorithme par recherche avant avec horizon).



- **Problème**: l'arbre peut rapidement devenir très très large. **Coût**: $O((S_{max} \mid A \mid)^T)$ avec S_{max} le nombre max de successeur et T le nombre de temps.

Programmation dynamique

Caractérisation des fonctions valeur

Définition

On se place dans le cadre horizon infini amorti. On définit l'opérateur L_π sur les fonctions valeurs $V \in \mathbb{R}^{|S|}$ définies par

$$L_\pi V = r_\pi + \gamma \mathbb{P}^\pi V$$

avec

$$\mathbb{P}^\pi(s_{t+1} = s^{'} \mid s_t = s) = \sum_{a \in A} \pi(a \mid s) P(s^{'} \mid a, s), \quad r^\pi(s) = \sum_{a \in A} \pi(a \mid s) r(s, a)$$

dans le cas stochastique et

$$\mathbb{P}^\pi(s_{t+1} = s^{'} \mid s_t = s) = P(s^{'} \mid \mu(s), s), \quad r_\pi = (s, \mu(s))$$

dans le cas déterministe.

Théorème: Caractérisation

Soient $\gamma < 1$ et $\pi \in D^{AM}$ une politique stationnaire aléatoire Markovienne. Alors la fonction valeur V^π est l'unique solution de l'équation $V^\pi = L_\pi V^\pi$, ce qui équivaut à

$$V^\pi(s) = \mathbb{E}[r_t + \gamma V^\pi(s_{t+1}) \mid s_t = s] = r^\pi(s) + \gamma \sum_{s^{'} \in S} \mathbb{P}^\pi(s^{'} \mid s) V^\pi(s^{'})$$

$$Q^\pi(s, a) = \mathbb{E}[r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a] = r(s, a) + \gamma \sum_{s^{'} \in S} P(s^{'} \mid s, a) V^\pi(s^{'})$$

Preuve de la caractérisation

- Cas stochastique. On considère la fonction valeur au temps initial:

$$V^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^{k} r_{t+1} \mid s_0 = s\right] = \mathbb{E}^{\pi}\left[r_1 + \gamma r_2 + \gamma_2 r_3 + \ldots \mid s_0 = s\right]$$

$$V^{\pi}(s) = \mathbb{E}^{\pi}\left[r_1 \mid s_0 = s\right] + \gamma \mathbb{E}^{\pi}\left[r_2 + \gamma_1 r_3 + \ldots \mid s_0 = s\right]$$

Pour une politique stochastique:

$$\mathbb{E}^{\pi}\left[r_1 \mid s_0 = s\right] = \sum_{a} \pi(a \mid s) r(s, a) = r^{\pi}(s)$$

car la politique est une variable aléatoire sur les actions. On estime $\mathbb{E}^{\pi}\left[r_2 \ldots \mid s_0 = s\right]$ (espérance des récompenses partant de s). Caractère Markovien implique:

$$\mathbb{E}^{\pi}\left[r_2 + \gamma r_3 + \ldots \mid s_0 = s\right] = \sum_{s'} \mathbb{P}^{\pi}(s' \mid s) \mathbb{E}^{\pi}\left[r_2 + \gamma r_3 + \ldots \mid s_1 = s'\right]$$

avec $\mathbb{P}^{\pi}$ définit au-dessus. On note que $\mathbb{E}^{\pi}\left[r_2 + \gamma r_3 + \ldots \mid s_1 = s'\right] = V^{\pi}(s')$, ce qui permet d'obtenir: $V^{\pi} = L_{\pi} V^{\pi}$.

- Unicité: on réécrit l'opérateur sous la forme suivante:

$$\left(I_d - \gamma \mathbb{P}^{\pi}\right) V^{\pi} = r^{\pi}$$

Puisque $\mathbb{P}^{\pi}$ est une matrice de probabilité, toute les valeurs propres de module inférieur ou égal à 1. Si $\gamma < 1$ idem. L'opérateur est donc inversible.

Fonction valeur et politique optimale I

Equation d'optimalité de Bellman

On suppose le critère d'horizon infini amorti. Pour $\gamma < 1$, la fonction valeur optimale $V^*(.) \in \mathcal{V}$ est l'unique solution de **l'équation de Bellman**:

$$V^*(s) = LV^*(s), \quad \forall s \in S$$

avec l'opérateur

$$LV(s) = \max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right)$$

Notation vectorielle:

$$\max_{\mu \in D^M} (r_\mu + \gamma \mathbb{P}_\mu V), \quad \forall V$$

On a aussi

$$Q^*(s,a) = L_Q Q^*(s,a), \quad \forall s \in S, \forall a \in A$$

avec l'opérateur

$$L_Q Q(s,a) = \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a') \right)$$

Preuve Bellman I

Equivalence déterministe/aléatoire

$\forall V$ on a

$$LV = \max_{\mu \in D^M}(r_\mu + \gamma \mathbb{P}_\mu V) = \max_{\pi \in D^{MA}}(r_\pi + \gamma \mathbb{P}_\pi V)$$

- Début preuve
  - Premier sens immédiat: $D^M \subset D^{AM}$
  - Autre sens: Soit $\pi \in D^{AM}$. On considère

$$L_\pi V(s) = \sum_a \pi(s,a) \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right), \quad \forall s \in S$$

$$L_\pi V(s) \leq \sum_a \pi(s,a) \max_{a'} \left(r(s,a') + \gamma \sum_{s'} P(s'|s,a') V(s') \right) = LV(s), \quad \forall s \in S$$

donc

$$L_\pi V = (r_\pi + \gamma \mathbb{P}_\pi V) \leq LV = \max_{\mu \in D^M}(r_\mu + \gamma \mathbb{P}_\mu V)$$

ce qui donne

$$\max_{\pi \in D^{MA}} (r_\pi + \gamma \mathbb{P}_\pi V) \leq \max_{\mu \in D^M}(r_\mu + \gamma \mathbb{P}_\mu V)$$

- fin preuve.
- On peut donc considérer l'ensemble des politiques déterministes.

Preuve Bellman II

- Caractérisation:
  - Si μ^* existe par définition de la politique optimale déterministe, on a:

$$V^* = \max_{\mu \in \Pi} V^\mu = \max_\mu \left(r_\mu + \gamma \mathbb{P}^\mu V^\mu \right).$$

  - Toutes politiques s'écrit $\mu = (\mu_0, \mu_1, ...\mu_n, ...) = (\mu_0, \mu')$. Dans ce cas on obtient :

$$V^* = \max_\mu V^\mu = \max_{(\mu_0, \mu')} V^\mu = \max_{(a, \mu')} V^\mu$$

  - On a donc

$$\max_{(a,\mu')} V^\mu = \max_{(a,\mu')} \mathbb{E}[r(s_0, a) + \gamma r(s_1, \mu_1(s_1)) + ... \mid s_0 = s; \mu]$$

$$\max_{(a,\mu')} V^\mu = \max_{(a,\mu')} \left(r(s, a) + \gamma P(s' \mid s, a) V^{\mu'}(s') \right)$$

$$\max_{(a,\mu')} V^\mu = \max_a \left(r(s, a) + \gamma P(s' \mid s, a) \max_{\mu'} V^{\mu'}(s') \right)$$

$$\max_{(a,\mu')} V^\mu = \max_a \left(r(s, a) + \gamma P(s' \mid s, a) V^*(s') \right)$$

Preuve Bellman III

- Justification de la dernière égalité:
  - 1ère inégalité

$$\max_{\mu'} \sum_{s'} P(s' \mid s,a) V_\gamma^{\mu'}(s') \leq \sum_{s'} P(s' \mid s,a) \max_{\mu'} V^{\mu'}(s')$$

  - 2ème inégalité: On part de

$$(*) = \sum_{s'} P(s' \mid s,a) \max_{\mu'} V^{\mu'}(s')$$

Soit $\bar{\mu} = \text{argmax}_{\mu'} V^{\mu'}$ donc

$$(*) = \sum_{s'} P(s' \mid s,a) V_\gamma^{\bar{\mu}}(s') \leq \max_{\mu'} \sum_{s'} P(s' \mid s,a) V^{\mu'}(s')$$

  - Les deux inégalités permettent de conclure.
- **Remarque**: l'équation de Bellman optimale caractérise les fonctions valeurs optimales. Cette fonction valeur peut être atteinte sur l'ensemble des politiques stationnaires Markovienne déterministes.

Preuve Bellman IV

- Unicité:
  - Problème de **point fixe** dans un espace de Banach, donc théorème de point-fixe. Unicité $=$ L est contractante pour la norme L^{∞}.
  - On va calculer $A = \mid LV(s) - LU(s) \mid$
  - Puisque le maximum des valeurs absolues défini une norme on peut donc utiliser la propriété $\parallel x \parallel - \parallel y \parallel < \parallel x - y \parallel$. On a donc

$$A \leq \max_a \left| \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right) - \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) U(s') \right) \right|$$

$$\mid LV(s) - LU(s) \mid \leq \gamma \max_a \left| \sum_{s'} P(s'|s,a) \left(V(s') - U(s') \right) \right|$$

  - Puisque la fonction valeur ne dépend pas de l'action on a:

$$\mid V(s) - U(s) \mid \leq \gamma \max_a \sum_{s'} P(s'|s,a) \mid V(s') - U(s') \mid \leq \gamma \parallel V - U \parallel_{\infty}$$

  - Précédente relation vrai pour tous les états donc:

$$\parallel LV - LU \parallel_{\infty} = \max_s \mid LV(s) - LU(s) \mid \leq \gamma \parallel V - U \parallel_{\infty}$$

Caractérisation de la politique optimale

Politique optimale

Une politique stationnaire est dit optimale si et ssi sa fonction valeur satisfait l'équation de Bellman ce qui équivaut à

$$\pi \in \text{argmax}_a \left(r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) V^*(s') \right) \tag{1}$$

- Début Preuve.
- 1) (1) vers optimalité:
  - L'équation (1) équivaut à $LV^* = L_\pi V^*$ car L correspond à L_π quand $\pi = \text{argmax}_\pi V^*$.
  - On suppose que π satisfait (1) donc

$$V^* = LV^* = L_\pi V^* = V^*$$

    et par unicité de l'équation $V = L_\pi V$ on a $V_\pi = V^*$. Donc si π dans (1) alors $V_\pi = V^*$.
- 2) Optimalité vers (1) :
  - On a donc $V_\pi = V^*$. Puisque $V_\pi = L_\pi V_\pi$ on a $V^* = L_\pi V^*$. Puisqu'elle est optimale $L_\pi = L$ donc $V^* = LV^*$.

Algorithme basé sur les valeurs

- **1er Idée**: L'équation de Bellman est un est un problème de point fixe. On veut donc faire un Picard.
- Algorithme:
  - Initialiser V_0 la fonction valeur initiale,
  - n=0
  - Tant que $\mid V_{n+1} - V_n \mid > \varepsilon$:
    - Pour tout $s \in S$: $V_{n+1} = \max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s') \right)$
  - Pour tout $s \in S$: $\mu(s) = \text{argmax}_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s') \right)$
- **Remarque**: l'algorithme construit la politique gloutonne.
- Complexité par itération: $O(\mid S \mid^2 \mid A \mid)$.
- **Question**: A chaque itération on améliore V mais est ce qu'on améliore μ ? En effet $V^{\mu_n} \neq V_n$

Amélioration de la politique

$$\| V^* - V^{\mu_n} \|_\infty \leq \frac{2\gamma}{1-\gamma} \| V^* - V_n \|_\infty$$

- **Convergence asymptotique**: $O\left(\frac{log\epsilon^{-1}}{log\gamma^{-1}}\right)$ itérations pour avoir une erreur $\approx \epsilon$.

Algorithme basé sur les valeurs II

- Preuve:
  - On part donc de $\| V^* - V^{\mu^n} \|_\infty$ et on développe:

$$\begin{aligned}\| V^* - V^{\mu^n} \|_\infty =& \| V^* - L_{\mu^n} V_n + L_{\mu^n} V_n - V^{\mu^n} \|_\infty \\ \leq& \| V^* - L_{\mu^n} V_n \| + \| L_{\mu^n} V_n - V^{\mu^n} \|_\infty \\ \leq& \| LV^* - L_{\mu^n} V_n \| + \| L_{\mu^n} V_n - L_{\mu^n} V^{\mu^n} \|_\infty\end{aligned}$$

  - On a utilisé que $LV^* = V^*$ par définition de la fonction valeur optimale,
  - On a utilisé que $L_\mu V^{\mu^n} = V^{\mu^n}$ (caractérisation des politiques).
  - Puisque μ_n est gloutonne (obtenu par argmax) $L_{\mu^n} V = LV$. On a donc

$$\begin{aligned}\| V^* - V^{\mu^n} \|_\infty \leq& \| LV^* - LV_n \|_\infty + \| LV_n - LV^{\mu^n} \|_\infty \\ \leq& \gamma \| V^* - V_n \|_\infty + \gamma \| V_n - V^{\mu^n} \|_\infty \\ \leq& \gamma \| V^* - V_n \|_\infty + \gamma \left(\| V_n - V^* \|_\infty + \| V^* - V^{\mu^n} \|_\infty \right)\end{aligned}$$

  - on a donc

$$(1-\gamma) \| V^* - V^{\mu^n} \|_\infty \leq 2\gamma \| V_n - V^* \|_\infty$$

Algorithme basé sur la politique

- **Algorithme précédent**: politique calculée à la fin.
- **Idée**: processus d'évaluation/amélioration:

$$\pi_0 \xrightarrow{e} V_0 \xrightarrow{a} \pi_1 \xrightarrow{e} v_1 \xrightarrow{a} \pi_2 .... \xrightarrow{a} \pi_* \xrightarrow{e} v_*$$

- Algorithme:
  - Initialiser V_0 la fonction valeur initiale,
  - n=0
  - Tant que $\mid \mu_{n+1} - \mu_n \mid > \varepsilon$:
    - On résoud

$$V_n = \left(r(s, \mu_n(s)) + \gamma \sum_{s'} P(s' \mid s, \mu_n(s)) V_n(s') \right), \quad \forall s \in S$$

    - Pour tout $s \in S$: $\mu_{n+1}(s) = \text{argmax}_a \left(r(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V_n(s') \right)$

- Complexité par itération: $O(\mid S \mid^2 \mid A \mid) + O(\mid S \mid^3)$.
- Ici on inverse un système linéaire. Il existe des variantes ou on résout le système de façon itérative avec une faible précision.
- **Convergence au pire en**: $O\left(\frac{|S||A|}{1-\gamma} log \frac{1}{1-\gamma} \right)$

Méthodes avec/sans modèles

Avec ou sans modèle ?

- Un algorithme est dit avec modèle s' il utilise les lois de probabilité $P(. \mid ., .)$ et $r(., .)$.
- Il est dit sans modèle si il utilise des transitions (s_t, a_t, r_t, s_{t+1}) sans connaître les lois de probabilités.

- Dans certaines applications on ne connaît pas ou on ne peut pas calculer le modèle.
- **Suite**: On va introduire des algorithmes sans modèles basés sur des approches stochastiques.
- **Définition**: On parle de planification lorsque le modèle est connu, d' apprentissage par renforcement si il est inconnu.
- Exemple: robot aspirateur. **Idéal**: il connaît le plan de la pièce. **1ère modèle**: politique aléatoire. **Modèles récents**: RL lien.

Intérêt pour les EDP

- Le modèle est en gros connu. Intérêt de la planification/renforcement pour les EDP ?
  - Beaucoup de méthodes pour construire des contrôles en boucle fermée,
  - **Contrôle en boucle fermée**: contrôle en temps réel. Utile en médecine, pour contrôler des méthodes numériques.
  - Cadre plus flexible que le contrôle optimal classique ?
  - **Possibilité**: codes qui optimisent au gré des simulations.

Méthodes Stochastiques I: Monte-Carlo

Rappel sur Monte-carlo

Estimateur

Soit $(X_n)_{n\geq 0}$ une suite de variable aléatoire idd de loi P_θ. Un estimateur de θ est une variable aléatoire $\hat{\theta}_n$ telle qu'il existe une fonction $F_n : E^n \to \Theta$, $\hat{\theta}_n = F_n(X_1, ..., X_n)$. Il est dit, consistant si $\hat{\theta}_n \xrightarrow{p.s.} \theta$ par rapport à P_θ quand $n \to +\infty$

Méthode de Monte Carlo

Soit X une variable aléatoire et $(X_1, ..., X_n)$ un échantillon de X. La moyenne empirique

$$E_n = \frac{1}{n}\sum_{i=1}^{n} g(X_i)$$

est un estimateur sans biais consistant de $\mathbb{E}[g(X)]$.

Théorème

Sous hypothèses, l'algorithme converge:

$$\bar{E}_{n+1} = (1 - \alpha_n)\bar{E}_n + \alpha_n g(X_{n+1})$$

vers $\mathbb{E}[g(X)]$. $\alpha_n = \frac{1}{n+1}$ donne exactement Monte-Carlo.

Échantillonnage préférentiel

- L'erreur commise à Monte-Carlo dépend de la variance de X.
- **Modification de la variance**: Échantillonnage préférentiel.
- Cas discret:

$$\mathbb{E}[g(X)] = \sum_{i=1}^{m} g(x_i)p(x_i) = \sum_{i=1}^{m} \frac{g(x_i)p(x_i)}{\tilde{p}(x_i)}\tilde{p}(x_i) = \mathbb{E}\left[\frac{g(Y)p(Y)}{\tilde{p}(Y)}\right]$$

- avec Y une variable aléatoire suivant une loi de probabilité $\tilde{p}$.
- Deux possibilités: appliquer la méthode de MC sur la 1ère espérance (loi X) ou sur la dernière (loi Y).
- **But**: trouver la probabilité $\tilde{p}$ tel que

$$\mathbb{V}\left[\frac{g(Y)p(Y)}{\tilde{p}(Y)}\right] \leq \mathbb{V}[g(X)]$$

et dans ce cas on estime l'espérance dépendante de Y.

- **Intérêt**: Permet de générer des échantillons avec une autre loi que celle dont on calcul l'espérance.

Application aux MDP: 1er algorithme I

- On rappelle les fonctions valeurs [Rlin2020]:

$$V(s) = \mathbb{E}[G_t \mid s_t = s], \quad Q(s,a) = \mathbb{E}[G_t \mid s_t = s, a_t = a]$$

- Les méthodes de recherche consiste à calculer l'espérance complète (arbre) et à trouver le plus court chemin.
- Les méthodes de programmation dynamique utilisent une récurrence et calcul l'espérance complète sur une transition.
- **Monte-Carlo**: calculer un estimateur de l'espérance par Monte-Carlo (moyenne empirique) + processus évaluation/amélioration.

$$\pi_0 \xrightarrow{e} V_0 \xrightarrow{a} \pi_1 \xrightarrow{e} v_1 \xrightarrow{a} \pi_2 .... \xrightarrow{a} \pi_* \xrightarrow{e} v_*$$

- Calcul de Q (évaluation):

$$Q(s,a) = \mathbb{E}\left[\sum_k^T r_\pi(s_k) \mid s_0 = s, a_0 = a\right] \approx \frac{1}{n}\sum_{i=1}^n \sum_k^T r(s_k, a_k) \approx \frac{1}{N_s}\sum_{i=1}^n \sum_k^T r(s_k, a_k)$$

avec n un certains nombre de trajectoires générées et N_s le nombre de fois que s est croisé.

- Une fois une politique évaluée, on l'améliore en la calculant avec Q.

Application aux MDP: 1er algorithme II

- On introduit un 1er algorithme de MC (chaque visite).
- Initialisation de $Q_0(.,.)$ et $\pi_0(.)$ arbitraire,
- Pour tout episode $k \leq n$:
  - initialiser $G(.,.) = 0$ et $N(.,.) = 0$
  - choisir aléatoirement un état s_0 et une action a_0
  - calculer une trajectoire $(s_0, a_0, r_1, ...s_{T-1}, a_{T-1}, r_T)$ selon la politique π_k
  - tout couple (s_t, a_t) de la trajectoire:
    - calculer $G_{loc} = \sum_{i=0}^{T} \gamma^i r_{r+t+1}$,
    - $G(s_t, a_t) += Gloc$ et $N(s_t, a_t) += 1$,
    - $Q_k(s_t, a_t) = \frac{1}{N(s_t, a_t)} G(s_t, a_t)$
    - Politique gloutonne: $\pi_{k+1}(s_t) = \text{argmax}_a \, Q_k(s_t, a)$
- Version "1ère visite de l'algorithme": on ne compte un état qu'une fois par transition.
- **Avantage MC vs DP**: Méthodes MC utilisent des estimations indépendantes les unes des autres. DP: utilise les fonctions valeurs associées aux autres états. MC plus adaptées au sous ensemble d'état.

Application aus MDP: MC soft

- **Défaut**: Convergence globale = explorer toutes les paires (s, a). Exploration aléatoire avec la 1er action, pas suffisant. Ex : environnement comme les jeux.

Solution : politique stochastique

Générer les trajectoires avec une politique stochastique $\pi(a \mid s)$. Cette politique stochastique sera la politique évaluée par V (méthode "on-policy")

- Monte-Carlo soft (chaque visite).
- Pour tout episode $k \leq n$:
  - initialiser $G(.,.) = 0$ et $N(.,.) = 0$
  - choisir aléatoirement un état s_0 et une action a_0
  - calculer une trajectoire $(s_0, a_0, r_1, ...s_{T-1}, a_{T-1}, r_T)$ selon la politique π_k
  - tout couple (s_t, a_t) de la trajectoire:
    - calculer $G_{loc} = \sum_{i=0}^{T} \gamma^i r_{r+t+1}$,
    - $G(s_t, a_t) += Gloc$ et $N(s_t, a_t) += 1$,
    - $Q_k(s_t, a_t) = \frac{1}{N(s_t, a_t)} G(s_t, a_t)$
    - la politique est donnée par

$$\pi(a \mid s_t) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s_t)|} & \text{si } a = \text{argmax}_a \, Q_k(s_t, a) \\ \frac{\epsilon}{|A(s_t)|} & \text{si } a \neq \text{argmax}_a \, Q_k(s_t, a) \end{cases}$$

- En général ϵ décroît au fur et à mesure. Convergence vers la politique gloutonne.

Application aux MDP: MC soft II

- On peut démontrer que chaque itération améliore la politique.
- MC soft (cas standard $\epsilon = 0$):

$$V_{\pi_{k+1}}(s) = Q_{\pi_k}(s, \pi_{k+1}(s)) = \sum_a \pi_{k+1}(a \mid s) Q_{\pi_k}(s, a)$$

puisqu'on utilise une politique stochastique avec $\sum_a \pi_{k+1}(a \mid s) = 1$.

$$V_{\pi_{l+1}}(s) = \frac{\epsilon}{\mid A(s) \mid} \sum_a Q_{\pi_k}(s, a) + (1 - \epsilon) \max_a Q_{\pi_k}(s, a)$$

on utilise l'inégalité

$$\max_a Q_{\pi_k}(s, a) \geq \sum_a \frac{\pi_k(a \mid s) - \frac{\epsilon}{|A(s)|}}{1 - \epsilon} Q_{\pi_{lk}}(s, a)$$

Pour obtenir cette inégalité on utilise: $\sum_a \frac{\pi_k(a|s) - \frac{\epsilon}{|A(s)|}}{1-\epsilon} = 1$ et que les coefficients ≥ 0. On a donc une combinaison convexe. En utilisant l'inégalité précédente on obtient:

$$V_{\pi_{k+1}}(s) \geq \frac{\epsilon}{\mid A(s) \mid} \sum_a Q_{\pi_l}(s, a) - \frac{\epsilon}{\mid A(s) \mid} \sum_a Q_{\pi_k}(s, a) + \sum_a \pi_k(a \mid s) Q_{\pi_k}(s, a)$$

$$V_{\pi_{k+1}}(s) \geq V_{\pi_k}(s)$$

Méthode "Off-policy", échantillonnage préférentiel I

- Pour explorer on utilise une politique stochastique. Si on veut construire la politique gloutonne on fait tendre $\pi(a \mid s)$ vers la politique gloutonne. Pas toujours évident.

Idée: méthodes "off-policy"

Séparer la politique d'exploration $b(a \mid s)$ utilisée pour générer la trajectoire de celle évaluée (politique cible) par $Q(s, a)$ notée $\pi(a \mid s)$.

- **Avantages**: plus souple pour l'exploration. On peut brasser les trajectoires (on y reviendra). Etc
- Construction de la méthode Monte-Carlo "off-policy":
  - Condition de couverture: pour évaluer π avec les trajectoires générées par b il faut:

$$\pi(a \mid s) > 0 \Rightarrow b(a \mid s) > 0$$

  - **Outil naturel**: échantillonnage préférentiel.
  - **Idée**: on pondère les **retours** en fonction de la probabilité que leurs trajectoires soit données par les politiques cible et d'exploration.
  - Ratio pour $G_t = \sum_k r_{t+k+1}$:

$$\rho_{t:T-1} = \frac{\mathbb{P}(a_t, s_{t+1}, a_{t+1}, \ldots, s_T \mid s_t, \pi)}{\mathbb{P}(a_t, s_{t+1}, a_{t+1}, \ldots, s_T \mid s_t, b)}$$

Méthode "Off-policy", échantillonnage préférentiel II

- On remarque rapidement que

$$\mathbb{P}(a_t, s_{t+1}, a_{t+1}, ..., s_T \mid s_t, \pi) = \pi(a_t \mid s_t)P(s_{t+1} \mid s_t, a_t)\pi(a_{t+1} \mid s_{t+1})...P(s_T \mid s_{T-1}, a_{T-1})$$

$$\mathbb{P}(a_t, s_{t+1}, a_{t+1}, ..., s_T \mid s_t, \pi) = \Pi_{k=t}^{T-1}\pi(a_k \mid s_k)P(s_{k+1} \mid s_k, a_k)$$

ce qui donne

$$\rho_{t:T-1} = \frac{\Pi_{k=t}^{T-1}\pi(a_k \mid s_k)P(s_{k+1} \mid s_k, a_k)}{\Pi_{k=t}^{T-1}b(a_k \mid s_k)P(s_{k+1} \mid s_k, a_k)} = \Pi_{k=t}^{T-1}\frac{\pi(a_k \mid s_k)}{b(a_k \mid s_k)}$$

- Lorsqu'on génère les trajectoires avec la politique b et qu'on estime $\mathbb{E}[G_t \mid s_t = s]$ en pratique on calcul $\mathbb{E}[G_t \mid s_t = s; b] = V_b(s)$. Par contre si on estime, avec les trajectoires générées par b, l'espérance:

$$\mathbb{E}[\rho_{t:T-1}G_t \mid s_t = s] \approx \frac{\sum_{t\in\tau(s)}\rho_{t:T-1}G_t}{\mid \tau(s) \mid}$$

estime bien $V_\pi(s)$ (avec $\tau(s)$ l'ensemble des temps ou on passe par l'état s).

- En pratique: soft politique pour l'exploration $b(a \mid s)$ et une politique cible gloutonne.

Méthodes Stochastiques II: Différences temporelles

Principe des différences temporelles et TD(0)

Résumé

On doit calculer $V(s) = \mathbb{E}[G_t \mid s_0 = s, a_0 = a] \; \forall s \in S$. Solutions:

- **Recherche exhaustive**: calcul total de l'espérance.
- **Programmation dynamique**: relation de récurrence entre $V(s)$ et $V^{'}(s)$ et calcul total de l'espérance.
- **Monte-Carlo**: estimation empirique de l'espérance.

Différence temporelle

Combiner DP et MC: relation de récurrence et estimation empirique de l'espérance.

- On rappelle la caractérisation de la fonction valeur:

$$V^{\pi}(s) = \mathbb{E}[r_{\pi}(s_t) + \gamma V_{\pi}(s_{t+1}) \mid s_t = s; \pi]$$

- Monte Carlo sur la caractérisation:

$$V^{\pi}(s_t) = \frac{1}{N(s_t)} \sum_{t} r_{\pi}(s_t) + \gamma V_{\pi}(s_{t+1})$$

- Version incrémentale générique:

$$V^{\pi}_{n+1}(s_t) = V^{\pi}_{n}(s_t) + \alpha_n \left(r_{\pi}(s_t) + \gamma V^{\pi}_{n}(s_{t+1}) - V_n(s_t)\right)$$

Algorithme Sarsa

- 1er algorithme basé sur la Q-fonction:

$$Q(s,a) = \mathbb{E}[G_t \mid s_t = s, a_t = a; \pi(,. \mid .)]$$

- Caractérisation par récurrence:

$$Q(s,a) = \mathbb{E}[r(s_t,a_t) + \gamma Q(s_{t+1},a_{t+1}) \mid s_t = s, a_t = a; \pi]$$

- Même principe qu'avant: Monte-Carlo sur la caractérisation. Version incrémentale:

$$Q^{\pi}_{n+1}(s_t,a_t) = Q^{\pi}_{n+1}(s_t,a_t) + \alpha_n(r(s_t,a_t) + \gamma Q^{\pi}_n(s_{t+1},a_{t+1}) - Q^{\pi}_n(s_t,a_t))$$

Algorithme (SARSA):

- Initialisation de $Q(.,.)$ arbitraire avec $Q(s_{terminal},.) = 0$
- pour tout épisode $k \leq n$:
  - initialiser s_0 de façon aléatoire
  - choisir a_0 selon la politique π (dépendante de Q) à partir de s_0
  - pour tout $t \in \{0,...,T\}$
    - calculer s_{t+1} et r_{t+1} en utilisant l'environnement et (s_t,a_t)
    - choisir a_{t+1} selon la politique π à partir de s_{t+1}
    - $Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$
    - on met à jour la politique π (si dépend de Q)
    - $s_t = s_{t+1}$, $a_t = a_{t+1}$

Algorithme Q Learning

- SARSA est un algorithme "on-policy".
- Version "off-policy" appélée Q-learning (algo star). Preuve de convergence.
- On applique MC (incrémentale) sur la caractérisation du Q optimal:

$$Q(s,a) = \mathbb{E}[r(s_t,a_t) + \gamma \max_{a'} Q(s_{t+1},a') \mid s_t = s, a_t = a; b]$$

- On définit la politique gloutonne $\mu(s) = \text{argmax}\, Q(s,a)$:

$$Q(s,a) = \mathbb{E}[r(s_t,a_t) + \gamma Q(s_{t+1},\mu(s_{t+1})) \mid s_t = s, a_t = a, b]$$

- On génère les trajectoires du MC avec une politique d'exploration stochastique $b(. \mid .)$.

Algorithme (Q-learning):

- Initialisation de $Q(.,.)$ arbitraire avec $Q(s_{terminal},.) = 0$
- pour tout épisode $k \leq n$:
  - initialiser s_0 de façon aléatoire;
  - pour tout $t \in \{0,...,T\}$:
    - calculer s_{t+1} et r_{t+1} en utilisant l'environnement et (s_t,a_t):
    - $Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_{t+1} + \gamma max_{a'} Q(s_{t+1},a') - Q(s_t,a_t))$:
    - on met à jour la politique $b(. \mid .)$ (si dépend de Q):
    - $s_t = s_{t+1}$.

Exploration vs exploitation I

Dilemme

Les algorithmes précédents cherchent un optimum global. Pour pouvoir converger vers cet optimum il faut passer par l'ensemble des transitions possibles. Conséquences:

$$\forall (s, a) \in S \times A, \quad b(s, a) > 0$$

avec b la politique d'exploration (identique à la politique cible dans le cas "on-policy").

- **Exploration**: visiter un maximum transitions,
- **Exploitation**: utiliser les meilleures transitions (afin de converger vite).
- **Dilemme**: on veux exploiter au maximum les meilleures transitions tout en explorant suffisamment.
-
- Beaucoup de solutions pour ajouter de l'exploration.

Exploration Gloutonne

On utilise $b(a \mid s) = \text{argmax}_{a'} Q(s, a)$. On risque de rater l'optimum global.

Exploration stochastique

- **Politique aléatoire**: loi uniforme sur les actions possibles. Convergence très lente.

Exploration vs exploitation II

Exploration Stochastique

- ϵ-**gloutonne**: politique gloutonne avec une probabilité ϵ et aléatoire avec une probabilité $1-\epsilon$. Très utilisée surtout avec $\varepsilon_0 = 1$ et ε_t décroissante.
- **Politique de Boltzmann (soft)**:

$$b(a \mid s) = \frac{\exp\left(\frac{1}{\beta} Q(s,a)\right)}{\sum_{a'} \exp\left(\frac{1}{\beta} Q(s,a')\right)}.$$

$\beta \approx 0$ (politique Gloutonne) et $\beta \approx \infty$ (politique aléatoire). β_t variable en temps.

Exploration par motivation intrinsèque

- lien entre l'apprentissage machine et Humain. **Idée**: motivation extérieur = maximiser les récompenses. Ajout une motivation intrinsèque (l'attrait de la nouveauté).
- Il existe deux possibilités
  - Par modification des récompenses: on remplace $r(s_t, a_t)$ par $r(s_t, a_t) + r^+(s_t, a_t)$.
  - Par modification de la sélection d'action: $a_t = \text{argmax}_a(Q(s_t, a) + r^+(s_t, a))$
- **Exemple**: borne supérieur l'intervalle de confiance:

$$r^+(s,a) = \sqrt{p \frac{\ln(\sum_{(s',a')} N(s',a'))}{N(s,a)}}$$

Variantes du Q-learning

- Il existe un certain nombre de variantes du Q-learning. Deux exemples:
- Soft-Q learning
  - permet de gérer l'exploration. On ajoute dans la maximisation un terme d'entropie afin de maximiser l'aspect stochastique de la politique. On ajoute: $\mathcal{H}(\pi(. \mid s)) = \mathbb{E}[-\ln \pi(a \mid s) \mid \pi]$
  - A la fin, on à l'incrément:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma\beta \ln \left(\sum_{a'} \frac{1}{\beta} \exp(Q(s_{t+1}, a')) \right) - Q(s_t, a_t) \right)$$

avec

$$\pi(a \mid s) = \frac{\exp\left(\frac{1}{\beta} Q(s, a)\right)}{\sum_{a'} \exp\left(\frac{1}{\beta} Q(s, a')\right)}$$

- Double Q learning
  - Soit $Q(s_{t+1}, a') = Q^*(s_{t+1}, a') + e_{a'}$ avec $e > 0$. Si Q estimateur sans biais de Q^* et $\max_{a'} Q(s, a')$ estimateur biaisé de $\max_{a'} Q^*(s, a')$.
  - **Conséquence**: on surestime Q dans les zones très stochastiques même si Q petit.
  - DQL: 2 Q-fonctions, politique moyenne. Incrément du DQL:

$$Q_{1,2}(s_t, a_t) = Q_{1,2}(s_t, a_t) + \alpha(r_{t+1} + \gamma Q_{1,2}(s_{t+1}, \text{argmax}_{a'} Q_{2,1}(s_{t+1}, a')) - Q_{1,2}(s_t, a_t)$$

Méthodes TD(n)

- TD(0)/MC: utilise une/toutes les transitions de l'épisode pour construire Q. On peut construire des méthodes intermédiaires.
- Monte Carlo calcule l'espérance empirique associée à:

$$Q^{\pi}(s,a) = \mathbb{E}[G_t, s_t = s, a_t = a, \pi]$$

et SARSA calcule l'espérance empirique de

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}), s_t = s, a_t = a, \pi]$$

- Remarque: on peut faire apparaître la récurrence plus tard dans l'épisode:

$$G_t = r_{t+1} + \gamma r_{t+2} + ... + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$$

- "On/Off policy" pour Monte Carlo: on calcule l'espérance empirique de

$$Q(s,a) = \mathbb{E}[G_t \mid s_t = s, a_t = a, \pi], \quad \text{vs } Q(s,a) = \mathbb{E}[\rho_{t:T-1} G_t \mid s_t = s, a_t = a, b]$$

avec $\rho_{t:h} = \Pi_{k=t}^{T-1} \frac{\mu(a_k|s_k)}{\pi(a_k|s_k)}$.

- "On/Off policy" pour incrément de SARSA-TD(n): on calcule l'espérance empirique de

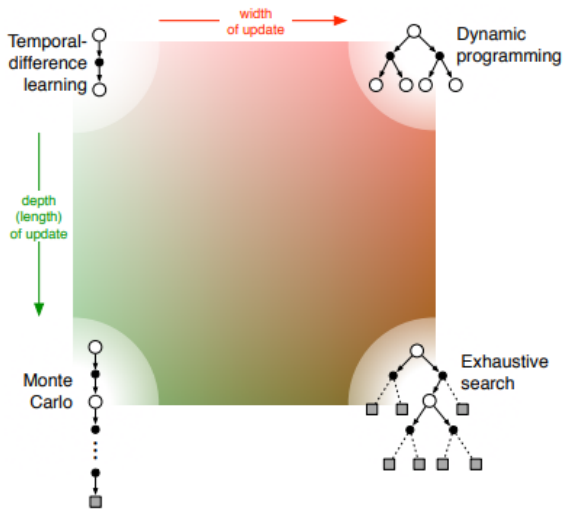
$$Q(s,a) = \mathbb{E}[r_{t+1} + ... + \gamma^{n-1} r_{t+n} + \gamma^n Q(s_{t+n}, a_{t+n}) \mid s_t = s, a_t = a, \pi]$$

vs

$$Q(s,a) = \mathbb{E}[\rho_{t:T-1}(r_{t+1} + ... + \gamma^{n-1} r_{t+n} + \gamma^n Q(s_{t+n}, a_{t+n})) \mid s_t = s, a_t = a, b]$$

Unification des approches

- Introduction de nombreuses méthodes. Critères de classification:
  - **Profondeur** (P): nombre de transitions en temps utilisées pour évaluer la Q fonction.
  - **Largeur** (L): nombre de transitions possibles entre deux temps utilisées pour évaluer la Q fonction.
  - **Avec ou sans modèle**



- **De Largeur** $= 1$: les méthodes TD(n) (SARSA, Q-learning etc) avec $P = n$. Les méthodes MC avec $P = \infty$. SARSA/Q-learning et variantes avec $P = 1$.
- **De Largeur** > 1: les méthodes de programmation dynamique avec $P = 1$. Les méthodes de recherche exhaustive avec $P = \infty$.
- Les méthodes avec $L > 1$ sont des méthodes avec modèles.

Grand problèmes: méthodes avec approximation

Modèles paramétriques et gradient stochastique

Modèles paramétriques

- Problèmes ou S est grand/indénombrable. On ne peut plus construire $V(s)$ un vecteur.
- **Idée**: construire les fonctions $V(s)/Q(s,a)$. On prendra des représentations paramétriques:

$$Q_\theta(s,a) = \sum_{i=1}^{m} \theta_i w_i(s,a)$$

avec $w_i(.,.)$ des fonctions de bases (affines, polynomiales, ondelettes etc).

- On peut utiliser des petits ANN (plus compliqués pour les réseaux profonds).

- Comment ajuster ces modèles sachant que les $(s,a)_i$ sont générés aléatoirement.
- Gradient stochastique (GS):
  - On pose la fonctionnelle:

$$J(\theta) = \mathbb{E}[j(\theta, W)]$$

avec W une variable aléatoire.
  - On écrit le gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta j(\theta, W)]$$

  - Une descente de gradient s'écrit :

$$\theta_{k+1} = \theta_k - \mathbb{E}[\nabla_\theta j(\theta_k, W)]$$

  - GS: $\mathbb{E}[\nabla_\theta j(\theta_k, W)] \approx \nabla_\theta j(\theta_k, W_i)$ avec W_i tirée de façon aléatoire.

Programmation dynamique avec approximation I

- Étendre la programmation dynamique à des problèmes de grandes tailles [PDMIA08].
- Algorithme d'itération sur les valeurs:

$$V^{n+1} = LV^n$$

- On le remplace par (IVA)

$$V^{n+1} = \Pi_p LV^n$$

avec Π_p le projecteur sur un espace $\mathcal{F}$ de fonction représentable.

- Exemple:

$$\mathcal{F} = \left\{ V_\theta = \sum_{i=1}^{d} \theta_i \phi_i(s), \theta \in \mathbb{R}^d \right\}$$

- La projection se fait sur un ensemble d'état $(s_1, ...s_n)$ tirés aléatoirement à chaque itération. La projection s'écrit donc

$$\Pi_p LV^n = \text{argmin}_{\theta \in \mathbb{R}^d} \sum_{k=1}^{n} \| \sum_{i=1}^{d} \theta_i \phi_i(s_k) - LV(s_k) \|_p^p$$

- Est ce que ça marche ? But : démontrer ?

$$\lim_{n\to\infty} \| V^* - V^{\pi^n} \|_p < C \lim_{n\to\infty} \| \Pi_p LV_n - LV_n \|_p$$

- On souhaite majorer l'erreur par rapport à V^* en fonction de l'erreur d'approximation

Programmation dynamique avec approximation II

Résultat dans L^∞

Si $\gamma < 1$

$$\lim_{n\to\infty} \| V - V^{\mu^n} \|_\infty < \frac{2\gamma}{(1-\gamma)^2} \lim_{n\to\infty} \| \Pi_p L V_n - L V_n \|_\infty$$

- Preuve: On pose $\epsilon = max_{0<n<N} \| \Pi_p L V_n - L V_n \|_p$. On a

$$\| V^* - V_{n+1} \|_\infty = \| L V^* - V^{n+1} \|_\infty \leq \| L V^* - L V_n \|_\infty + \| L V_n - V_{n+1} \|_\infty$$

donc puisque $V_{n+1} = \Pi_\infty L V_n$ et $\| L V_a - L V_b \| \leq \gamma \| V_a - V_b \|_\infty$

$$\| V^* - V_{n+1} \|_\infty \leq \gamma \| V^* - V_n \|_\infty + \epsilon$$

- On applique récursivement ce calcul:

$$\| V^* - V_N \|_\infty \leq \gamma^n \| V^* - V_0 \|_\infty + (1 + \gamma + ..\gamma^N)\epsilon$$

- On utilise $(1 + \gamma + ..\gamma^N) < \frac{1}{1-\gamma}$ et $\| V^* - V^{\mu^n} \|_\infty \leq \frac{2\gamma}{(1-\gamma)} \| V^* - V_n \|_\infty$ (slide 22). On a

$$\| V^* - V^{\mu^N} \|_\infty \leq \frac{2\gamma}{(1-\gamma)^2}\epsilon + \frac{2\gamma^{N+1}}{(1-\gamma)^2} \| V^* - V_0 \|_\infty$$

- On conclut en prenant la limite.

Programmation dynamique avec approximation III

- **Lemme précédent**: si on maîtrise l'erreur d'approximation on maitrise l'erreur de l'algorithme IVA.
- erreur d'approximation:

$$\epsilon = O\left(\| f - \Pi_\infty f \|_\infty + \frac{1}{\sqrt{n}} \right)$$

- **Itération sur la politique**: on peut construire des méthodes IPA et obtenir le même genre de résultat (plus technique).

En pratique

Le cas $p = \infty$ n'est pas utilisé en pratique.
On utilise $p = 1$ ou $p = 2$ pas $p = \infty$. Dans ce cas on perd le caractère contractant de π_p. Il existe des résultats (plus complexe) dans ce cadre. Voir [PDMIA08]

- Avec des approches similaires on peut étendre les algorithmes de renforcement aux fonctions paramétriques.

MC/TD avec approximation I

- **Rappel**: Incrément de Monte Carlo/TD(n)/TD(0) [RLC18] unifié :

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(G_t^a - Q(s_t, a_t))$$

avec $G_t^a = G_t$ (MC), $G_t = r_{t+1} + \gamma V(s_{t+1})$ (TD(0) etc).

- Quand le modèle est paramétrique on veut donc minimiser:

$$\text{argmin}_\theta \, \mathbb{E}_\pi[(Q_\pi^{ref} - Q_\theta)^2]$$

- La descente de gradient stochastique donne:

$$\theta_{k+1} = \theta_{k+1} - \alpha(Q_\pi^{ref}(s_t, a_t) - Q_\theta(s_t, a_t))\nabla_\theta Q_\theta(s_t, a_t)$$

- On connait pas Q_π^{ref}. Que faire ?
- Pour MC: G_t un estimateur sans biais de Q_π^{ref}. Pour TD(0) $r_{t+1} + \gamma V_\theta(s_{t+1})$ un estimateur biaisé de Q_π^{ref}.
- On peut aussi utiliser une approche moindre carré (solution dans le cadre linéaire).

MC/TD avec approximation II

L'algorithme (Q-learning) avec approximation:

- Initialisation de $Q(.,.)$ arbitraire avec $Q(s_{terminal},.) = 0$
- pour tout épisode $k \leq n$:
  - initialiser s_0 de façon aléatoire
  - pour tout $t \in \{0, ..., T\}$
    - calculer s_{t+1} et r_{t+1} en utilisant l'environnement et (s_t, a_t)
    - choisir a_{t+1} selon la politique b à partir de s_{t+1}
    - calcul de l'incrément

$$\Delta\theta = (r_{t+1} + \gamma max_{a'} Q_\theta(s_{t+1}, a') - Q_\theta(s_t, a_t))\nabla_\theta Q_\theta(s_t, a_t)$$

    - $\theta = \theta - \alpha\Delta\theta$
    - on met à jour la politique $b(. \mid .)$ (si dépend de Q)
    - $s_t = s_{t+1}$
- Même principe pour MC/SARSA etc.

Cas des réseaux profonds

- Cette approche marche mal. Les ANN préfèrent apprendre avec des données vraiment différentes/peu corrélées.
- Ici: on apprend avec $(s_i, a_i, r_{i+1}, s_{i+1})$ puis avec $(s_{i+1}, a_{i+1}, r_{i+2}, s_{i+2})$ etc. C'est trop proche.
- Il est important de mélanger les trajectoires.

Deep Q learning

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode $= 1, M$ **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For

- Deux ingrédients: "experience replay" (mélange des transitions dans un buffer) et "target policy" (réseaux différents pour la politique target et la politique d'exploration).
- **Important**: avoir un algorithme "off-policy" est nécessaire pour "experience replay".

Grand problèmes: gradients de politiques

Principe

Construire des méthodes principalement basées sur une politique paramétrique et non sur les fonctions valeurs. On souhaite minimiser une fonction. Pas de optimum global.

- Soit $\pi_\theta(a \mid s)$ une politique de paramètres θ (réseaux de neurones).
- Problème de minimisation:

$$\max_{\theta \in \mathbb{R}^d} \mathcal{J}(\theta)$$

avec

$$\mathcal{J}(\theta) = \mathbb{E}[G_t \mid \pi_\theta] = \int_S p_0(s) V_{\pi_\theta}(s) ds$$

et $p_0(s)$ la distribution des états initiaux.

- **But**: maximiser la moyenne des fonctions valeurs sur l'ensemble des états.
- **Principe des méthodes**: méthode de gradient. **Question**: comment calculer le gradient ?
- Type de méthodes:
  - Acteur pur: utilise uniquement la politique,
  - Critique pur: utilise uniquement les fonctions valeurs (DQN etc),
  - Acteur-critique: utilise les deux.

Théorème de Gradient de politique I

lemme

La fonctionnelle à maximiser peut se réécrire sous la forme:

$$\mathcal{J}(\theta) = \int_s \rho^{\pi_\theta}(s) \left(\sum_a \pi_\theta(a \mid s) r(s,a) \right) ds$$

avec la distribution des états amortis:

$$\rho^{\pi}(s) = \int_S \sum_{t=0}^{\infty} \gamma^t p(s_0) P(s_0 \to s, t, \pi_\theta) ds_0$$

- preuve:

On commence par écrire la fonctionnelle avec le retour amorti:

$$\mathcal{J}(\theta) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[r(s_t, a_t) \mid \pi_\theta \right] = \sum_{t=0}^{\infty} \gamma^t \int_S d(s,t) \left(\sum_a \pi(a \mid s) r(s,a) \right) ds$$

avec d une distribution des états au temps t. La distribution peut être réécrite comme

$$d(s,t) = \int_S p(s_0) P(s_0 \to s, t, \pi_\theta) ds_0$$

P la probabilité d'atteindre s à partir de s_0 en suivant la politique π_θ au temps t.

Théorème de Gradient de politique II

On utilise que:

$$\mathcal{J}(\theta) = \sum_{t=0}^{\infty} \gamma^t \int_S \left(\left(\int_S p(s_0) P(s_0 \to s, t, \pi_\theta) ds_0 \right) \left(\sum_a \pi(a \mid s) r(s, a) \right) \right) ds$$

$$\mathcal{J}(\theta) = \int_S \left(\left(\int_S \sum_{t=0}^{\infty} \gamma^t p(s_0) P(s_0 \to s, t, \pi_\theta) ds_0 \right) \left(\sum_a \pi(a \mid s) r(s, a) \right) \right) ds$$

On définit la distribution des états amortis: $\rho^\pi(s)$ pour conclure la preuve.

- fin preuve.

Théorème du Gradient de politique

Le gradient de la fonctionnelle $\mathcal{J}$ est donné par:

$$\nabla \mathcal{J}(\theta) = \int_S \rho^\pi(s) \left(\sum_a Q^{\pi_\theta}(s, a) \nabla_\theta \pi_\theta(a \mid s) \right) ds$$

$$= \mathbb{E}_{\pi_\theta} \left[\sum_a Q^{\pi_\theta}(s, a) \nabla_\theta \pi_\theta(a \mid s) \right]$$

$\mathbb{E}_{\pi_\theta}$ l'espérance ou: les états sont distribués selon ρ^{π_θ} les actions générées par $\pi_\theta(. \mid .)$.

Théorème de Gradient de politique III

- preuve:

On considère la fonctionnelle (on note $p_0(s_0) = p_0$ et $Q_{\pi_\theta} = Q$:

$$\mathcal{J}(\theta) = \int_S p_0 V_{\pi_\theta}(s_0) ds_0 = \int_S p_0 \left(\sum_a \pi_\theta(a_0 \mid s_0) Q_{\pi_\theta}(s_0, a_0) \right) ds_0$$

Maintenant on calcule le gradient nommé A. On a

$$A = \underbrace{\int_S p_0 \left(\sum_a \nabla_\theta \pi_\theta(a_0 \mid s_0) Q(s_0, a_0) \right) ds_0}_{A_1} + \underbrace{\int_S p_0 \left(\sum_a \pi_\theta(a_0 \mid s_0) \nabla_\theta Q(s_0, a_0) \right) ds_0}_{A_2}$$

Maintenant on utilise la caractérisation de la Q fonction (cas continue), ce qui donne:

$$Q_{\pi_\theta}(s_0, a_0) = \mathbf{r}(s_0, a_0) + \gamma \int_S P(s_1 \mid s_0, a_0) V(s_1) ds_1$$

On a donc

$$A_2 = \int_S p_0 \left(\sum_a \pi_\theta(a_0 \mid s_0) \nabla_\theta \left(r(s_0, a_0) + \gamma \int_S P(s_1 \mid s_0, a_0) V(\mathbf{s}_1) ds_1 \right) \right) ds_0$$

$$A_2 = \int_S p_0 \left(\sum_a \pi_\theta(a_0 \mid s_0) \nabla_\theta \left(\gamma \int_S P(s_1 \mid s_0, a_0) V(s_1) ds_1 \right) \right) ds_0$$

puisque une récompense sur une étape en temps ne dépend pas de la politique.

Théorème de Gradient de politique IV

C'est pareil pour les probabilités donc on a:

$$A_2 = \int_S p_0(s_0) \left(\sum_a \pi_\theta(a_0 \mid s_0) \nabla_\theta \left(\gamma \int_S P(s_1 \mid s_0, a_0) \nabla_\theta V(s_1) ds_1 \right) \right) ds_0$$

On utilise maintenant le théorème de Fubini pour obtenir et on utilise la définition $P(s_0 \to s_1, 1, \pi_\theta) = \sum_a \pi_\theta(a_0 \mid s_0) P(s_1 \mid s_0, a_0)$. On obtient que

$$A_2 = \int_S \left(\int_S \gamma p_0(s_0) P(s_0 \to s_1, 1, \pi_\theta) ds_0 \right) \nabla_\theta V_{\pi_\theta}(s_1) ds_1$$

Maintenant on utilise que $V_{\pi_\theta}(s_1) = \sum_a \pi_\theta(a_1 \mid s_1) Q(s_1, a_1)$ pour obtenir

$$A_2 = \int_S \left(\int_S \gamma p_0(s_0) P(s_0 \to s_1, 1, \pi_\theta) ds_0 \right) \nabla_\theta \left(\sum_a \pi_\theta(a \mid s_1) Q(s_1, a) \right) ds_1$$

- Gradient du produit de fonction,
- Pour le terme avec le gradient de Q_1 on utilise la caractérisation de la fonction Q.
- On utilise Fubini et la probabilité de transition en s_1 et s_2.

On applique itérativement cela sur A_2 le long d'une trajectoire et on combine avec A_1:

$$\nabla_\theta \mathcal{J}(\theta) = \int_S \left(\int_S \sum_t^\infty \gamma^t p_0(s_0) P(s_0 \to s, t, \pi_\theta) ds_0 \right) \left(\sum_a \nabla_\theta \pi_\theta(a \mid s) Q_{\pi_\theta}(s, a) \right) ds$$

π et Q ne dépendent pas du temps ce qui permet de conclure.

Algorithme Reinforce I

Corolaire

Le résultat précédent peut être réécrit sous la forme suivante;

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi^\theta} \left[Q^{\pi_\theta}(s,a) \nabla_\theta \ln \pi_\theta(a \mid s) \right]$$

- Preuve:

On considère le résultat du théorème de gradient de politique:

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_a Q(s,a) \nabla_\theta \pi_\theta(a \mid s) \right] = \mathbb{E}_{\pi_\theta} \left[\sum_a \pi_\theta(a \mid s) Q^{\pi_\theta}(s,a) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \right]$$

Puisque la trajectoire est générée par la politique, l'action a est donnée par la politique π_θ. Par définition $\pi(a^{'} \mid s) = 1$ avec $a^{'} = a$ et $\pi(a^{'} \mid s) = 0$ avec $a^{'} \neq a$. On a donc

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta} \left[Q^{\pi_\theta}(s,a) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \right]$$

En utilisant la dérivée du logarithme on obtient le résultat.

- fin preuve
- Rappel: $Q(s,a) = \mathbb{E}[G_t \mid s_t = s, a_t = a]$. Estimateur sans biais:

$$Q(s,a) \approx \frac{1}{M} \sum_{k=1}^{m} G_k$$

avec le retour G_k est calculer à partir de trajectoire commençant par le couple (s,a)

Algorithme Reinforce II

- On remplace pour obtenir:

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}\left[\left(\frac{1}{M}\sum_{k=1}^{m} G_k\right)\frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)}\right]$$

- On applique un Monte-Carlo à l'espérance globale:

$$\nabla \mathcal{J}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{M}\sum_{k=1}^{m} G_k\right)\nabla_\theta \ln \pi_\theta(a_i \mid s_i)$$

avec G_k calculé avec comme état initial (s_i, a_i).

- Au final cela se réécrit sous la forme:

$$\nabla \mathcal{J}(\theta) \approx \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T} G_t \nabla_\theta \ln \pi_\theta(a_t \mid s_t)$$

Algorithme Reinforce

- Pour tout épisode $e \leq N$
  - choisir aléatoirement $\mathbf{s}_0$,
  - calculer (avec la politique) et stocker la trajectoire $(\mathbf{s}_0, \mathbf{a}_0, \mathbf{r}_1, \mathbf{s}_1, ..., \mathbf{r}_T)$,
  - pour tout $t < T$:
    - calculer le retour amorti G_t,
    - $\nabla \mathcal{J} = \nabla \mathcal{J} + G_t \nabla \ln \pi_\theta(a_t \mid s_t)$,
  - $\theta = \theta + \eta \nabla \mathcal{J}$.

Q-acteur critique I

- **Défaut de Reinforce**: grosse variance. Amélioration possible: algorithmes acteur-critique (AC):
- Il existent beaucoup d'algo de type AC. On ne détaille que deux approches.
- On repart de:

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}\left[Q^{\pi_\theta}(s,a)\frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)}\right]$$

- **Proposition**: forme générale:

$$\nabla_\theta \hat{\mathcal{J}}(\theta) = \mathbb{E}_{\pi_\theta}\left[f^{\omega}(s,a)\frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)}\right]$$

- Sous certaines conditions (on détails pas) sur f_ω on a

$$\nabla_\theta \hat{\mathcal{J}}(\theta) = \nabla_\theta \mathcal{J}(\theta)$$

- **Acteur-critique "Vanilla"**: On choisit pour f_ω une approximation de Q par réseaux de neurones.
- Comment construire Q_ω? On rappelle par Bellman que la Q-fonction satisfait:

$$Q^{\pi_\theta}(s_t,a_t) = r_{t+1} + \gamma Q^{\pi_\theta}(s_{t+1},a_{t+1})$$

- Donc ici, on propose de résoudre

$$\min \frac{1}{2} \parallel (r_{t+1} + \gamma Q^{\omega}(s_{t+1},a_{t+1}) - Q^{\omega}(s_t,a_t))) \parallel$$

- Cela permet d'obtenir un algorithme proche de SARSA avec en plus une descente de gradient.

Q-acteur critique II

Algorithme:

- Pour tout épisode $e \leq N$
  - choisir aléatoirement $\mathbf{s}_0$. choisir a_0 avec $\pi(a \mid s_0)$
  - pour tout $t < T$:
    - calculer (s_{t+1}, r_{t+1}) avec l'environnement,
    - calculer la prochaine action a_{t+1} avec la politique π_θ,
    - calcul de la cible:
$$G_t = r_{t+1} + Q^\omega(s_{t+1}, a_{t+1})$$
    - calcul du gradient
$$\mathcal{J}_\omega = \mathcal{J}_\omega + G_t \nabla Q^\omega(s_t, a_t)$$
    - calcul du gradient
$$\mathcal{J}_\theta = \mathcal{J}_\theta + Q^\omega(s_t, a_t) \nabla \ln \pi_\theta(a_t \mid s_t)$$
  - $\theta = \theta + \eta \mathcal{J}_\theta$
  - $\omega = \omega + \alpha \mathcal{J}_\omega$
- "Reinforce" et" Q acteur-critique" sont "on policy". Variante "off policy" par échantillonnage préférentielle:
$$\nabla \mathcal{J}_{off}(\theta) = \mathbb{E}_{\pi^\theta}\left[\frac{\pi(a \mid s)}{b(a \mid s)} Q(s, a) \nabla_\theta \ln \pi_\theta(a \mid s)\right]$$
- Théorie un peu différente: ce gradient est une approximation. L'ensemble des minimum locaux du vrai gradient inclus dans l'ensemble associé a $\nabla \mathcal{J}_{off}(\theta)$.
- **Utile**: pour utiliser "experience replay". Utile pour les réseaux de neurones.

Q-acteur critique III

Algorithme:

- Fonction avantage sur une transition:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

- Sur une transition, en utilisant la caractérisation des Q fonctions (méthodes TD(n)):

$$A(s_t, a_t) = r_{t+1} + \gamma r_{t+2} + ... + \gamma^{n-1} r_{t+n} + \gamma^n Q(s_{t+n}, a_{t+n}) - V(s_t)$$

  ce qui est égal à

$$A(s_t, a_t) = r_{t+1} + \gamma r_{t+2} + ... + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}) - V(s_t)$$

- **Idée**: Utiliser une approximation de la fonction avantage comme fonction f^ω. Pour cela réseaux de neurone V^ω approchant V.
- Plusieurs possibilités:
  - Cas $n = \infty$: on a approché $Q(s, a) = \mathbb{E}[G_t \mid s_t = s, a_t = a]$ par G_t. Approximation sans biais mais à grande variance.
  - Cas $n = 1$: Approximation avec petite variance mais biais potentiellement important (car on approche Q par un ANN)
  - il est conseillé d'utiliser $n > 1$.

Cas déterministe: théorème de gradient

Espace des actions

Pour le moment on a considéré un espace d'action A discret. Comment étendre au cas continu ?
Il est difficile de construire une loi de probabilité continue $\pi(a \mid s)$ avec des réseaux profonds (possible).
Possibilités:

- **Politique Gaussienne**: $\pi(a \mid s) = \mathcal{G}(\mu_\theta(s), \Sigma_\theta(s))$ une gaussienne avec μ et Σ des modèles paramétriques qui servent de moyenne et écart-type.
- **Politique déterministe**: $\mu_\theta(s)$ modèle paramétrique type ANN. Dans ce cas Th. gradient de politique ?

- Fonctionnelle à minimiser:

$$\mathcal{J}(\theta) = \int_S p_0(s) V^\mu(s) = \int_S p_0(s) Q^\mu(s, \mu(s))$$

- Théorème du gradient de politique (preuve similaire) :

$$\nabla_\theta \mathcal{J}(\theta) = \int_S \rho^\mu(s) \nabla_a Q^\mu(s, a)_{a=\mu(s)} \nabla_\theta \mu_\theta(s)$$

- On peut donc construire des algorithmes type Acteur-Critique. Le plus connu (code de V. Vigon à Strasbourg) est le DDPG.

CAs déterministe: DDPG

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Fig 3. DDPG Algorithm. (Image source: Lillicrap, et al., 2015)

- Beaucoup d'autres variantes de ces algorithmes. Quelques approches différentes: PPO/TRPO.
- Autres: Lien vers liste d'algo
- **Autre chose intéressante**: Inverse RL (on apprend les récompenses).

Méthodes basées sur les modèles

Différences avec l'approche précédente

Renforcement

Jusqu'à présent on apprend à contrôler en interaction avec l'environnement. Implicitement on un modèle et un contrôle.

Renforcement avec modèle

On sépare l'apprentissage du modèle et le problème de contrôle. Une fois le modèle connu, on peut:

- Utiliser des algos type planification globale (programmation dynamique),
- utiliser des algos type planification immédiate/locale (contrôle optimal).

- Les deux approches peuvent aussi être couplées.
- On peut apprendre un modèle sur: les états, des observations partielles, des représentations réduites etc.
- **Apprentissage du modèle**: Processus/mélange Gaussien, Réseaux de neurones/ modèles localement linéaires.

Remarque

approches plus supervisé. On apprend le modèle avec les transitions. On peut apprendre la politique avec des trajectoires optimales.

Méthode hybride: Dyna Q

Il s'agit d'un algorithme de renforcement qui apprend le modèle en même temps que la politique et génère des exemples par le modèle. **Utile** quand la réponse de l'environnement est lourde à évaluer.

Détails de l'algorithme pour tout épisode $k \leq n$:

- Initialisation de $Q(.,.)$ arbitraire avec $Q(s_{terminal},.) = 0$
- pour tout $t \in \{0, ..., T\}$
  - choisir a_t selon la politique π (dépendante de Q) à partir de s_t
  - calculer s_{t+1} et r_{t+1} en utilisant l'environnement et (s_t, a_t)
  - $Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$ (apprentissage direct)
  - Ajuster le modèle avec les données de $(s_t, a_t, s_{t+1}, r_{t+1})$ (apprentissage du modèle).
  - $l < n$ (planification)
    - choisir aléatoirement s dans les états visités et une action associée a,
    - Utiliser le modèle pour obtenir (s', r) à partir de (s, a).
    - $Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$.
  - $s_t = s_{t+1}$

MPC

- **Planification immédiate**: calcul de la trajectoire (détaillé après)
- Apprentissage naïf du modèle + planification:
  - Avec un politique aléatoire $\pi(. \mid .)$ on génère une série de trajectoires et on stocke $D = \left\{(s, a, s^{'})_i\right\}$ et $R = \{r_i\}$,
  - Ajuster le modèle avec les données de D et R. Cas déterministe: minimiser

$$\| s_i^{'} - f(s_i, a_i) \|_2$$

  - On applique une planification immédiate avec $f(s, a)$ (ou $p(s \mid s, a)$) pour déterminer une trajectoire optimale.
- **Défaut immédiat**: Pour des gros problèmes, le modèle peut passer, lors du calcul de la trajectoire optimale, par une zone mal apprise et générer un contrôle faux.

Nouvel algorithme (MPC):

- Avec $\pi(. \mid .)$ aléatoire on générer une série de trajectoires stockées
- pour tout $k < N$
  - Ajuster le modèle avec les données de D et R.
  - pour tout $l < M$
    - Planification immédiate pour déterminer une trajectoire de référence: $(a_0^*, ..a_T^*)$
    - on exécute a_0^* à partir de s et on récupère $s^{'}$ et r (environnement),
    - on ajoute $(s, a, s^{'})$ dans D et r dans R.

Méthode de politique guidée

- La méthode MPC construit des contrôles en boucle ouverte (politique locales/trajectoires optimales). Depuis le début on cherche plutôt des contrôles en boucle fermée (politique globale)
- **Idée**: reprendre le principe du MPC mais utiliser les trajectoires locales pour construire une politique globale.

Nouvel algorithme (Politique guidée):

- Avec $\pi(. \mid .)$ aléatoire on génère une série de trajectoires stockées.
- pour tout $k < N$
  - Ajuster le modèle avec les données de D et R.
  - pour tout $l < M$
    - Planification immédiate pour déterminer une trajectoire de référence: $(a_0^*, ..a_T^*)$
    - Mise à jour de la politique en utilisant la trajectoire de référence.
    - on utilise la politique $\pi_\theta(a \mid s)$ pour générer des trajectoires et on ajoute $(s, a, s^{'})$ dans D et r dans R.
- Comment on modifie la politique ? Quel modèle ? Quelles méthodes de planification ?

Méthode de politique guidée: calcul de la politique

- Type de politique:
  - $\pi_\theta(a_t \mid s_t) = \mathcal{G}(\mu_\theta(s_t), \Sigma)$ avec $\mathcal{G}$ une loi Gaussienne mD de moyenne $\mu_\theta(s_t)$ (ANN) et Σ une variance fixée.
  - $\pi_\theta(a_t \mid s_t)$ un réseau de neurones de type softmax.
- Trajectoires issues de la politique globale/locale:

$$\pi_\theta(\tau), \quad \pi_{loc}(\tau) = (s_0, a_0^*, s_1 \ldots, a_{T-1}^*, s_T)$$

- **Problème**: il peut arriver que la trajectoire de référence (politique locale) soit fausse car elle s'éloigne de la région de validité du modèle.
- Modèle étant généré par les transitions de $\pi_\theta(a_t \mid s_t)$ cela équivaut à: les trajectoires locales ne doivent pas trop s'éloigner de celle générées par la politique globale.
- Solution:
  - On cherche $\pi_{loc}(\tau)$ solution de

$$\min_{\pi_{loc}} \mathbb{E}[\sum_{t=1}^{T} f_c(s_t, a_r)], \qquad \text{sous contrainte } d(\pi_{loc}(\tau), \pi_\theta(\tau)) \leq \epsilon$$

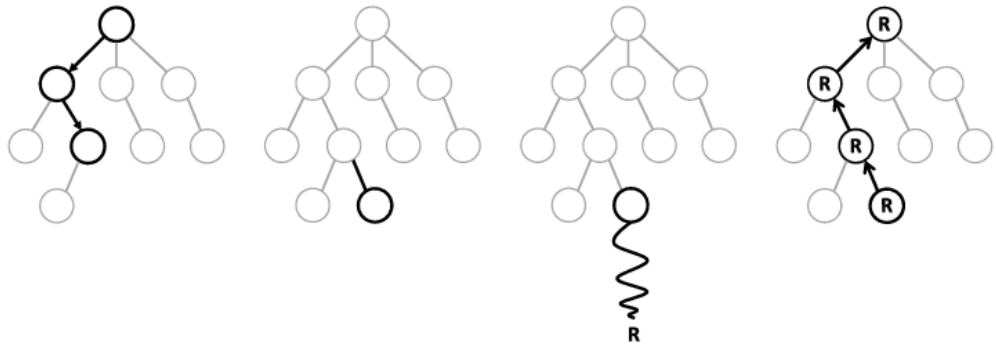
  - On optimise les poids de la politique globale en minimisant:

$$\min_\theta d(\pi_{loc}(\tau), \pi_\theta(\tau))$$

avec d une distance / pseudo-distance. Cas 1 (ou cas déterministe): norme L^2. Cas 2: Divergence de Kullback-Leibler.

Calcul de Trajectoire I: Monte-Carlo Tree Search

- **MCTS est un algorithme** de recherche de la meilleure action dans le cas discret.
- Création d'une approximation de l'arbre des futures transitions possibles de plus en plus étoffé là où sont générées les récompenses les plus importantes.
- 4 étapes:
  - **Sélection**: à l'aide d'une stratégie faisant un compromis entre exploration (branches les moins bien connues) et exploitation (branches les plus prometteuses), on génère un trajectoire jusqu'à un état qui n'avait jamais été visité,
  - **Expansion**: on ajoute ce noeud dans l'arbre,
  - **Simulation/evaluation**: on génère la suite de la trajectoire jusqu'à un état terminal avec une politique aléatoire,
  - **Rétro-propagation**: On remonte la trajectoire pour mettre à jour les estimations de la fonction valeur et des compteurs.



(a) Selection (b) Expansion (c) Simulation (d) Backpropagation

- Comparaison avec MC, TD(0), TD(n), programmation dynamique etc: largeur et profondeur > 1.

Calcul de Trajectoire II: Contrôle optimal LQR/ILQR

- Cas continue et déterministe: on a un modèle de transition T_θ et de récompense R_θ.
- On cherche:

$$\max_{a_0,..a_T} \sum_{t=0}^{T} R_\theta(s_t, a_t)$$

- On a les modèles donc:

$$\sum_{t=0}^{T} R_\theta(s_t, a_t) = R_\theta(s_0, a_0) + R_\theta(T_\theta(s_0, a_0), a_1) + ...R_\theta(T_\theta(T_\theta(..)), a_T)$$

- Si on a des modèles différentiables (ANN par exemple). On peut calculer le gradient (composition de fonctions) et résoudre le problème.
- **Défaut**:
  - important d'explosion/disparition de gradient: risque pour chaque gradient d'un ANN encore plus pour des grandes compositions.
  - Mauvais conditionnement par rapport à a_0.

-

- **Cas particulier**: T_θ est linéaire et $C_\theta = -R_\theta$ le coût est quadratique (LQR). On a des formules explicites (on résout dans le sens du temps et dans le sens inverse).
- **Cas non linéaire**: On utilise la méthode ILQR: on applique LQR sur le linéarisé dans le sens inverse du temps et la dynamique non linéaire dans le sens du temps et on itère.

Modèles locaux ou globaux

- Naturellement pour décrire les transitions on souhaite utiliser un modèle dit global. Il s'agit de représenter l'ensemble des transitions par un modèle paramétrique:
  - Processus Gaussien/ Mélange Gaussien.
  - Réseaux de neurones (MDP, RNN, Réseaux à densité de mélange etc).
- **Défaut**: si le problème est complexe il sera difficile de construire un modèle valide pour l'ensemble des couples (s, a). Un défaut de modèle = défauts de contrôle.

Modèles Locaux

L'idée est d'utiliser des modèles locaux:

$$P(s_{t+1} \mid s_t, a_t) = \mathcal{G}(f_t(s_t, a_t), \Sigma)$$

avec f_t une fonction valide que dans le voisinage de (s_t, a_t).

Approche populaire

Modèles locaux linéaire/quadratique + planification immédiate par la méthode LQR ou une variante stochastique:

$$\pi_{loc}(a_t \mid s_t) = \mathcal{G}(a_{LQR/ILQR}, \Sigma)$$

Apprentissage par imitation et distillation

- On voit que les approches basées sur le modèle on utilise des solutions obtenues par contrôle optimal du modèle appris pour construire la politique.
- Peut t'on coupler cette approche avec du renforcement ? Oui, apprentissage par imitation.
- On a des données $(\hat{s}_1, \hat{a}_1).....(\hat{s}_n, \hat{a}_n)$ issue du contrôle optimal ou autre. Comment les utiliser ?
- Algo 1:
  - Initialiser π_θ tel on résolve $\max_\theta \sum_i \pi_\theta(\hat{a}_i \mid \hat{s}_i)$
  - On utilise un algorithme de renforcement.
- On peut oublier les données de référence.
- Algo 2:
  - On fait du renforcement avec

$$\hat{\mathcal{J}}_\theta = J_\theta + \lambda \sum_i \pi_\theta(\hat{a}_i \mid \hat{s}_i)$$

- Algo 3: distillation
  - On fait du renforcement avec

$$\hat{\mathcal{J}}_\theta, \quad \text{sous contrainte} \sum_i d(\pi_\theta(\hat{a}_i \mid \hat{s}_i), \pi(\hat{a}_i \mid \hat{s}_i)) \leq \epsilon$$

Modèles du monde

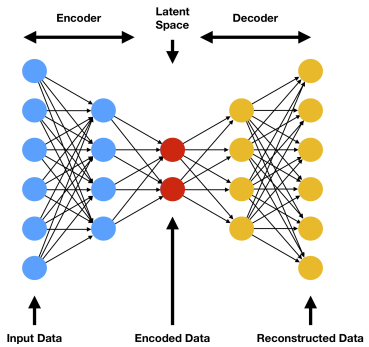
- **Idée**: parfois pas besoin de l'ensemble de l'état pour le contrôle. Renforcement avec modèles + réduction de dimension [WM18]. "Dreamer" de google.
  - Représentation en petite dimension de l'état s_t appelée z_t. Auto Encodeur: deux ANN $\phi(s_t)$ et $\phi^+(z_t)$ tel qu'on minimise:

$$\min \parallel s - \phi^+ \circ \phi(s) \parallel$$

  - Apprentissage du modèle de transition réduit à l'aide d'un réseau récurrent:

$$P(z_{t+1} \mid z_t, a_t, h_t)$$

    avec h_t des variables cachées associées au réseau récurrent.
  - Politique avec petit ANN et un algorithme de planification immédiate (MCTS, LQR/ILQR, algos génétiques)



- `https://worldmodels.github.io/`
- Analogie entre le "modèle du monde" et "EDP+base réduite + contrôle sur modèle réduit".

Conclusion

Papier EDP + RL

- "Reinforcement Learning with Function-Valued Action Spaces". Contrôle de l'équation de la chaleur. Méthode DDPG et variante.
- "Reinforcement Learning-based Model Reduction for Partial Differential Equations". Construction du fermeture pour une méthode de réduction POD. Application à Burgers.
- "Automating turbulence modelling by multi-agent reinforcement learning". Construction d'une fermeture de Turbulence en comparant à des solutions DNS. Version multi-agent du renforcement (un agent par maille). Méthode: "Off-Policy" gradient de politique.
- "learning to discretize: solving 1D scalar conservation laws via deep reinforcement learning". Construction d'un flux WENO pour Burgers en renforcement. Méthode DDPG.
- "Towards model discovery with reinforcement learning". Trouver les coefficients d'une edp à partir de simulation. Méthode DDPG.
- "Personalized Cancer Chemotherapy Schedule: a numerical comparison of performance and robustness in model-based and model-free scheduling méthodologies". Contrôle optimal d'une EDO pour la chimiothérapie. Comparaison du PMP, DDPG et DQN.
- "A review on Deep Reinforcement Learning for Fluid Mechanics". Liste des applications du RL en mécaflu. Principalement du contrôle optimal de fluide.

Applications dans MOCO

- Projets en cours dans MOCO:
  - Stabilisation d'oscillations de schémas numériques pour les EDP:

$$\partial_t \rho(t,x) + \partial_x^{DG} f(\rho) = \epsilon \partial_x^{DG}(g(\rho)\partial_x^{DG}\rho)$$

    But: Trouver $g(.)$ supprimant les oscillations numériques (Thèse Léo).
  - Schéma de Volumes finis pour Euler:

$$\Delta x \partial_t \mathbf{U}_j + G(\mathbf{U}_j, \mathbf{U}_{j+1}) - G(\mathbf{U}_{j-1}, \mathbf{U}_j) = 0$$

    But: Trouver G le meilleur possible. Contrairement au supervisé, le renforcement peut apprendre naturellement la stabilité. On fait un mixte entre RL et Supervisé.
  - Accélération de Newton :

$$\partial_t \rho + \partial_x f(\rho) = \partial_x(g(\rho)\partial_x \rho), \rightarrow H(\rho^{n+1}) = b(\rho^n)$$

    But: Condition initiale Newton $I(\rho^n)$ minimisant le nombre d'itération.
- Autres possibilités:
  - Contrôle en boucle fermée (rapidement évaluable) à partir de contrôle en boucle ouverte. Thèse de Mickaël ? Mimesis ?
  - Contrôle d'EDP avec terme stochastique (Clémentine ?)
  - Code à IMRA: DDPG + Apprentissage génétique (Pop-Up) par V. Vigon avec l'aide L. Bois.

Références

DMIA08 "Processus Décisionnels de Markov en Intelligence Artificielle" (grosse référence), Groupe PDMIA, 2008,

Rlin2020 "Reinforcement learning, an introduction" (grosse référence), R. Sutton. A. Barto, réédition 2020.

DPG14 "Deterministic Policy Gradient Algorithms", D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, 2014

RLC18 "Reinforcement Learning for Control": Performance, Stability, and Deep Approximators", L. Busoniua, T. de Bruinb, D. Tolic, J. Koberb, I. Palunkod, 2018.

GP16 " Policy Search via Approximate Mirror Descent", W. Montgomery, S. Levine, 2016.

WM18 " World models", D. Ha, J. Schmidhuber, 2018

DQL10 "Double Q-learning", H. Hasselt, 2010,

MCTS12 "A Survey of Monte Carlo Tree Search Methods", 2012.

DQN13 "Playing Atari with Deep Reinforcement Learning", V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, 2013.

Ex17 "Exploration in deep reinforcement learning", M. Semmler, 2017.

MBP18 "Model-Based Planning with Discrete and Continuous Actions", M. Henaff, W. Whitney, Y. LeCun, 2018.

Pmed17 "A Markov decision process approach to optimizing cancer therapy using multiple modalities", K. Maass, M. Kimy, 2018