Apprentissage par renforcement (Reinforcement Learning (RL)

Chapitre 4: Programmation Dynamique (Dynamic Programming)

T. AL-ANI
A²SI-ESIEE-PARIS

Chapitre 4 : Programmation Dynamique (Dynamic Programming) - Sommaire

- Évaluation de la politique (Policy evaluation)
- Amélioration de la politique (Policy improvement)
- Itération de la politique (Policy iteration)
- Itération de la valeur (Value iteration)
- Retours complets (Full backups)
- Itération Généralisée de la Politique (Generalized Policy Iteration (GPI))
- DP Asynchrone (Asynchronous DP)
- Mise à jour (Bootstrapping)

Programmation Dynamique (Dynamic Programming)

Objectifs de ce chapitre :

- Une vue générale sur un ensemble de méthodes, appelé **Programmation Dynamique** (**Dynamic Programming (DP)**), apportant des solutions classiques pour les **Processus de Décision de Markov** (**Markov Decision Process** (**MDP**)).
- Montrer comment DP peut être utilisé pour calculer les **fonctions de la valeur** (**value functions**), et ainsi, les **politiques optimales** (**optimal policies**).
- Montrer l'efficacité et l'utilité de DP

Programmation Dynamique

- Il existe trois classes fondamentales de méthodes pour résoudre le problème d'**Apprentissage par Renforcement (Reinforcement Learning (RL))** :
  - **Programmation Dynamique (DP)**
  - méthodes de **Monte Carlo (MC)**
  - Apprentissage par la **Différence-Temporelle (Temporel-difference) (TD))**

    Toutes ces méthodes permettent de résoudre le problème complet de RL incluant les récompenses retardées).

Programmation Dynamique

Chacune des trois classes de méthodes a ses puissances et ses faiblesses :

- **DP** : Bien développées mathématiquement, mais nécessitent un modèle complet et précis de l'environnement ;
- **MC** : ne nécessitent pas un modèle et sont conceptuellement très simples mais elles ne sont pas situées dans le cadre des calculs incrémentales pas à pas;
- **TD** : ne nécessitent pas un modèle et sont complètement incrémentales, mais sont plus complexes à analyser.

Ces méthodes se diffèrent aussi de différentes façons par rapport à leurs efficacités et la vitesse de convergence.

La troisième partie du livre de R. S. Sutton et A. G. Barto explore la combinaison de ces méthodes pour obtenir les meilleurs propriétés de chacune entre elles.

Programmation Dynamique

Le terme « Programmation Dynamique » se rapport à une collection d'algorithmes qui peuvent être utilisés pour calculer des politiques optimales étant donné un modèle parfait de l'environnement tel qu'un Processus de Décision de Markov (Markov Decision Process (MDP)).

Programmation Dynamique

Les algorithmes classiques de DP ont une utilité limitée en RL à la fois à cause de

- leur hypothèse d'un modèle parfait et
- à cause leur coût élevé du calcul. Cependant, ils restent théoriquement très importants. Les autres techniques permettent d'achever les mêmes performances de DP sans les deux limitations ci-dessus.

Programmation Dynamique

Nous supposons un MDP fini (voir chapitre 3) donné par

Un ensemble de probabilités de transition

$$P_{ss'}^{a} = \Pr\left\{s_{t+1} = s' \middle| s_t = s, a_t = a\right\}$$

Un ensemble de récompenses espérées immédiates

$$r_{ss'}^{a} = E\left\{r_{t+1} \middle| s_t = s, a_t = a, s_{t+1} = s'\right\}$$

$\forall\, s \in S, a \in A(s)$ et $\forall\, s^{'} \in S^{+}$,

où S^{+} est S plus un état terminal si le problème est épisodique.

Programmation Dynamique

Remarque : Malgré la DP peut être appliquée aux espaces continues d'état et d'action, une solution exacte est possible mais uniquement dans certains cas.

Un moyen pour obtenir une solution approchée est de quantifier les espaces d'état et d'action, ensuite appliquer la DP à espace d'état fini. Plusieurs méthodes seront développées au chapitre 8.

Programmation Dynamique

L'idée principale de DP et de RL en général est de calculer une **fonction de valeur** (**value function**) pour organiser et structurer la recherche pour des politiques optimales. Dans ce chapitre nous montrons comment la DP est utilisée pour calculer la fonction de valeur décrite au chapitre 3. Une politique optimale peut être trouvée si nous calculons des valeurs optimales V^* ou Q^* dans l'équation de Bellman.

Evalutaion de la politique (Policy Evaluation)

Evalutaion de la politique (Policy Evaluation) :

Pour une politique donnée π, calculer la **fonction de la valeur d'état** (state-value function) V^{π}

Appelons : **fonction de la valeur d'état (State - value function) pour une politique donnée** π **:**

$$V^{\pi}(s) = E_{\pi}\left\{R_t \middle| s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\}$$

Equation de Bellman pour V^{π} :

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \qquad V^{\pi}(s')\right], \text{ pour tout s} \in \text{S.}$$

C'est un system de $|S|$ équations linéaires simultanées.

Méthodes itératives (Iterative Methods)

$$V_0 \rightarrow V_1 \rightarrow \cdots \rightarrow V_k \rightarrow V_{k+1} \rightarrow \cdots \rightarrow V^\pi$$

balayage “sweep”

Un balayage (sweep) consist à appliquer une **opération de retour (backup operation)** à chaque état.

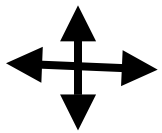
Une politique d’évaluation itérative :

$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right]$$

Evaluation itérative de la politique (Iterative Policy Evaluation)

Input π, the policy to be evaluated
Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx V^\pi$

A Small Gridworld



actions

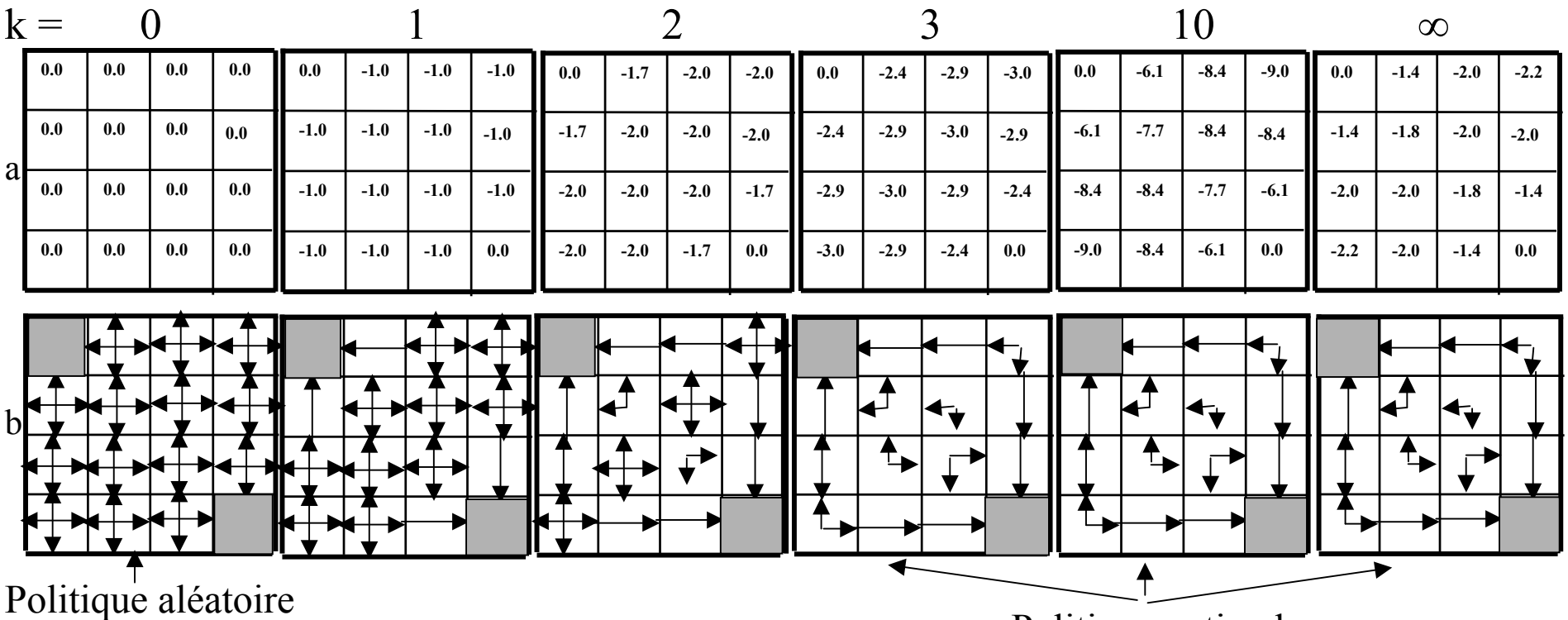
	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

R = -1 pour toutes les transitions

- Une tâche épisodique non actualisée (undiscounted episodic task)
- États non terminaux (Nonterminal states) : 1, 2, . . ., 14;
- Un état terminal (deux carrés gris)
- Des actions amenant l'agent en dehors de grille laissent l'état inchangé.
- Récompense = –1 jusqu'à ce que l'état terminal soit arrivé.

Adapté de (R. S. Sutton and A. G. Barto:
Reinforcement Learning: An Introduction http://www-anw.cs.umass.edu/~rich/book/the-book.html) par Tarik AL ANI, A²SI-ESIEE-PARIS

Iterative Policy Eval for the Small Gridworld



a : V_k pour la politique aléatoire

b : π Politique gourmande (greedy policy) par rapport à V_k

π : choix des actions aléatoires (uniformes)

Adapté de (R. S. Sutton and A. G. Barto:
Reinforcement Learning: An Introduction http://www-anw.cs.umass.edu/~rich/book/the-book.html) par Tarik AL ANI, A²SI-ESIEE-PARIS

Amélioration de la politique

Supposons que nous avons calculer V^{π} pour une politique déterministe π.
Pour un état donné s,
serait il mieux de choisir une action $a \neq \pi(s)$?
La valeur d'effectuer une action a dans l'état s est :

$$Q^{\pi}(s,a) = E_{\pi}\left\{ r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s, a_t = a \right\}$$

$$= \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma V^{\pi}(s') \right]$$

Il est mieux de basculer à l'action a pour l'état s si et seulement si $Q^{\pi}(s,a) > V^{\pi}(s)$

Amélioration de la politique Cont.

Effectuer cette opération pour tous les états pour obtenir une nouvelle politique π' qui est gourmande par rapport à V^{π} :

$$\pi'(s) = \arg\max_a Q^{\pi}(s,a)$$

$$= \arg\max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^{\pi}(s') \right]$$

Alors $V_{\pi'} \geq V_{\pi}$

Amélioration de la politique Cont.

Que ce passe t-il si $V^{\pi'} = V^{\pi}$?
i.e., pour tout $s \in S$,

$$V^{\pi'}(s) = \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^{\pi}(s') \right] \ ?$$

Mais ceci est l'équation d'optimalité de Bellman.
Ainsi $V^{\pi'} = V^{\pi}$ et les deux politiques sont optimales.

Itération de la politique

$$\pi_0 \rightarrow V^{\pi_0} \rightarrow \pi_1 \rightarrow V^{\pi_1} \rightarrow \cdots \pi^* \rightarrow V^* \rightarrow \pi^*$$

évaluation de la politique

Amélioration de la politique "greedification"

Itération de la politique

1. Initialization

   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

   Repeat

   $\Delta \leftarrow 0$

   For each $s \in \mathcal{S}$:

   $v \leftarrow V(s)$

   $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$

   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement

   policy-stable $\leftarrow$ true

   For each $s \in \mathcal{S}$:

   $b \leftarrow \pi(s)$

   $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$

   If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ false

   If *policy-stable*, then stop; else go to 2

Adapté de (R. S. Sutton and .
Reinforcement Learning: An Introduction http://www-anw.cs.umass.edu/~rich/book/the-book.html) par Tarik AL ANI, A²SI-ESIEE-PARIS

Itération de la Valeur (Value Iteration)

Rappelons l'itération complète de l'évaluation de la politique **(full policy evaluation backup)** :

$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right]$$

Voici l'itération complète de la valeur **(full value iteration backup)** :

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right]$$

Itération de la Valeur Cont.

Initialize V arbitrarily, e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat

$\Delta \leftarrow 0$

For each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that

$\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Adapté de (R. S. Sutton and A. G. Barto:
Reinforcement Learning: An Introduction http://www-anw.cs.umass.edu/~rich/book/the-book.html) par Tarik AL ANI, A²SI-ESIEE-PARIS

DP Asynchrone (Asynchronous DP)

- Toutes les méthodes décrites plus loin nécessitent une balayage (sweeps) exhaustive de l'ensemble entier d'état.
- La DP Asynchrone n'utilise pas une balayage. A la place, elle fonctionne comme suit :
  - Repéter jusqu'à ce que le critère de convergence soit satisfait:
    - Choisir aléatoirement un état et appliquer la propriété de retour (backup)
- La DP Asynchrone a toujours besoin de beaucoup de calcul, mais elle ne reste pas bloquée dans une balayage longue désespérante
- Pouvez-vous choisir des états pour effectuer intelligemment un retour? OUI : l'expérience de l'agent peut fonctionner comme un guide.

Itération Généralisée de la politique (Generalized Policy Iteration)

Itération Généralisée de la politique (Generalized Policy Iteration (GPI)) : n'importe quel interaction entre l'évaluation de la politique et l'amélioration de la politique, indépendamment de leur granularité et autres détails. L'interaction continue jusqu'à ce que V et π soient optimales.

Un métaphore géométrique
pour la convergence de GPI:

Mise à jour (Bootstrapping)

La **mise à jour** (**Bootstrapping**) est une propriété spéciale des méthodes de DP. Tous ces méthodes mettent à jour les valeurs estimées des états en se basant sur les valeurs estimées des états successeurs.

Efficacité de DP

- Le processus de trouver une politique optimale est polynomial en nombre d'états…
- MAIS, le nombre d'états est souvent astronomique, e.g., souvent augmente exponentiellement avec le nombre de variables d'état (ceci est appelé par Bellman la malédiction de la dimensionnalité "the curse of dimensionality").
- En practice, la DP classique peut être appliquée au problèmes avec quelques millions d'états.
- La DP asynchrone peut être appliquée aux problèmes plus vastes, et elle est appropriée au calcul parallèle.
- Il est étonnamment facile d'utiliser les MDPS pour lesquels les méthodes de la DP ne sont pas pratiques.