Prise de décisions complexes

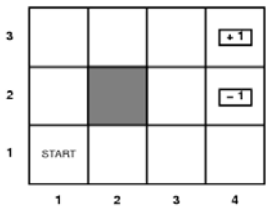
1

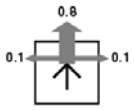
Plan

- Problèmes de décisions séquentielles
- Algorithme « value iteration »
- Algorithme « policy iteration »
- POMDP
- Réseau de décision dynamique

2

Exemple



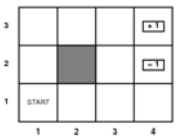


- Tous les états non terminaux ont une récompense de -0.04

3

Exemple

- Si l'environnement était déterministe, la solution serait: [Haut, Haut, Droite, Droite, Droite]
- Dû à l'incertitude, ce chemin n'a que $0.8^5 + (0.1^4 * 0.8) = 0.32776$ chance de réussite.



4

Modèle de transition

- Le modèle de transition spécifie la probabilité de chaque résultat possible d'une action.
- *T*(*s*, *a*, *s*'): la probabilité d'atteindre l'état *s*' en effectuant l'action *a* dans l'état *s*.
- On suppose des transitions Markoviennes
  - La probabilité d'atteindre *s*' ne dépend que de *s* et non des états précédents.

5

Fonction d'utilité

- Comme le problème de décision est séquentiel, la fonction d'utilité doit dépendre d'une séquence d'états.
- L'agent reçoit une récompense *R*(*s*) qui peut être positive ou négative.
- L'utilité d'une suite d'état est (pour le moment) la somme des récompenses reçues.

6

Processus de décision de Markov

- Problème de décision séquentielle dans un environnement complètement observable avec un modèle de transition Markovien et des récompenses additives.
- Un MDP est défini par:
  - État initial: S_0
  - Modèle de transition: $T(s, a, s')$
  - Fonction de récompense: $R(s)$

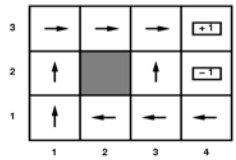
7

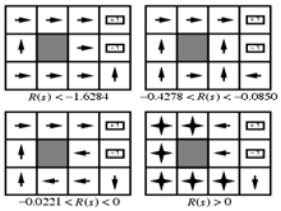
Politique

- Une solution au problème ne peut pas être une suite d'action fixe.
- Il faut pouvoir spécifier l'action à faire dans chaque état.
- Ceci est appelé une politique (π)
- $\pi(s)$ est l'action recommandée par la politique π pour l'état s.
- La politique optimale π^* est celle qui a l'utilité espérée maximale.

8

Politique optimale (Risk vs Reward)





HGD : pénible, désagréable,
BGD : morne, agréable

9

Fonction d'utilité

- On veut définir $U_h([s_0, s_1, \ldots, s_n])$
- Pour un horizon fini où après N étapes, the ``game is over'':
  - $U_h([s_0, s_1, \ldots, s_{N+k}]) = U_h([s_0, s_1, \ldots, s_N])$
  - La politique optimale est non stationnaire
    - La politique optimale peut changer dans le temps.
- Pour un horizon infini, la politique optimale est stationnaire.

10

Assignation d'utilité

- Récompenses additives:

$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots$$

- Récompenses escomptées (discounted):

$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$

  - où le facteur d'escompte γ est un nombre entre 0 et 1.
  - L'agent préfère les récompenses immédiates.

11

« Value Iteration »

- Algorithme utilisé pour calculer une politique optimale.
- Idée: Calculer l'utilité de chaque état et utiliser ces utilités pour choisir l'action optimale dans chaque état.

12

Utilité d'un état

- L'utilité doit être définie sur une séquence d'états.
- On donne comme utilité à un état l'utilité espérée des séquences d'états qui peuvent le suivre.
- Les séquences d'états dépendent de la politique exécutée.

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s\right]$$

13

Utilité d'un état

- La véritable utilité d'un état $U(s)$ est $U^{\pi^*}(s)$.
- C'est-à-dire la somme espérée des récompenses escomptées si l'agent exécute la politique optimale.
- $R(s)$: La récompense à court terme d'être dans l'état s.
- $U(s)$: La récompense à long terme à partir de l'état s.

14

Utilité maximale espérée

- Intuitivement et selon le principe de la rationalité, l'agent devrait choisir l'action qui maximise l'utilité espérée, c'est-à-dire:

$$\pi^*(s) = \text{argmax}_a \sum_{s'} T(s, a, s') U(s')$$

15

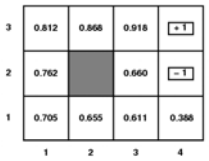
Équation de Bellman

- L'utilité d'un état est la récompense immédiate plus l'utilité espérée escomptée du prochain état, en supposant que l'agent choisi l'action optimale.

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U(s')$$

16

Exemple



$$U(1,1) = -0.04 + \gamma \max\{ \begin{array}{ll} 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), & (Up) \\ 0.9U(1,1) + 0.1U(1,2), & (Left) \\ 0.9U(1,1) + 0.1U(2,1), & (Down) \\ 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1) \} & (Right) \end{array}$$

U(1,1) = -0.04 + 1 * max{0.7456, 0.5732, 0.7, 0.6707} = 0.7056

17

Algorithme « value iteration »

- S'il y a *n* états, il y a n équations de Bellman.
- On veut résoudre toutes ces équations simultanément pour trouver les utilités.
- Le problème c'est que l'équation n'est pas linéaire (opérateur max)
- Donc, on utilise une technique par itération et on arrête quand on obtient un équilibre.

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U_i(s')$$

18

« Policy iteration »

- Commence avec une politique initiale
- Alterne entre:
  - Évaluation de politique
    - Sachant une politique π_i, calculer l'utilité de chaque état si π_i était utilisée ($U_i = U^{\pi i}$).
  - Amélioration de la politique
    - Calculer π_{i+1} en se basant sur U_i à l'aide de:

$$\pi^*(s) = \text{argmax}_a \sum_{s'} T(s,a,s')U(s')$$

- L'algorithme arrête lorsque $\pi_{i+1} = \pi_i$.

19

Évaluation de la politique

- Plus simple que les équations de Bellman parce qu'on connaît l'action de la politique.
- Après *i* itérations, l'utilité d'un état est:

$$U_i(s) = R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s')U(s')$$

- Ce qui donne une suite d'équations linéaires:

$$U_i(1,1) = 0.8U_i(1,2) + 0.1U_i(1,1) + 0.1U_i(2,1)$$
$$U_i(1,2) = 0.8U_i(1,3) + 0.2U_i(1,2)$$

- Peut être résolu exactement en $O(n^3)$ (n états → n éq avec n inconnues) par des algorithmes d'algèbre linéaire.

20

« Modified policy iteration »

- Pour de grands espaces d'états, on ne peut pas résoudre les équations exactement.
- On utilise donc une approche itérative avec une version simplifié de la mise à jour de Bellman:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s')U_i(s')$$

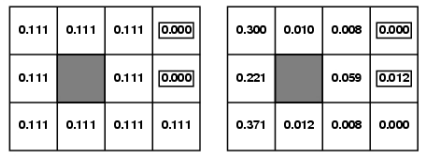
- Généralement plus performant que l'algorithme standard de « policy iteration » ou que l'algorithme « value iteration ».

21

POMDP

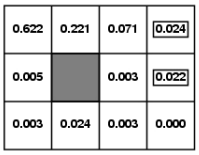
- POMDP: Processus de décision de Markov partiellement observable.
- L'agent ne sait pas nécessairement dans quel état il est, il ne peut donc pas exécuter l'action recommandée par $\pi(s)$.
- L'utilité de l'état *s* et l'action optimale en *s* ne dépendent pas uniquement de l'état *s*, mais aussi du niveau de connaissance de l'agent lorsqu'il est en *s*.
- Les POMDP sont plus difficiles que les MDP, mais le monde réel est un POMDP.

22

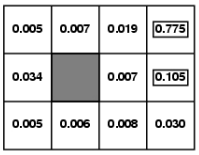


Départ

5 déplacements vers la gauche



5 déplacements vers le haut



5 déplacements vers la droite

23

Définition d'un POMDP

- POMDP:
  - Modèle de transition: $T(s, a, s')$
  - Fonction de récompense: $R(s)$
  - Modèle d'observations: $O(s, o)$
- Le modèle d'observation spécifie la probabilité de percevoir l'observation *o* dans l'état *s*.

24

« Belief state »

- Belief state: Une distribution de probabilités sur tous les états possibles.
  - Le belief state initial dans l'exemple est:
    - ⟨1/9, 1/9, 1/9, 1/9, 1/9, 1/9, 1/9, 1/9, 1/9, 0, 0⟩
- *b*(*s*): la probabilité assignée à l'état *s* par le belief state *b*.

25

« Belief state » courant

- Belief state courant: distribution de probabilités conditionnelles sur les états réels sachant la séquence d'observations et d'actions jusqu'à maintenant.
- Très proche de la tâche de filtrage.
- La fonction de mise à jour du belief state est:

$$b'(s') = \alpha\, O(s', o) \sum_s T(s, a, s') b(s)$$

b(s) est l'état de croyance précédent et l'agent vient de faire l'action *a* et a perçu l'observation *o*:

26

Action optimale

- Dans un POMDP, l'action optimale ne dépend que du belief state courant de l'agent.
- La politique optimale est donc: π*(*b*).
- Elle ne dépend pas de l'état actuel où est l'agent.

27

Décisions dans un POMDP

- Répéter
  - Sachant le belief state courant b, exécuter l'action a = π*(*b*).
  - Percevoir l'observation o.
  - Mettre à jour le belief state avec:

$$b'(s') = \alpha O(s', o) \sum_s T(s, a, s') b(s)$$

28

POMDP comme un MDP

- Modèle de transition sur l'espace des belief state:

$$\tau(b, a, b') = \sum_o P(b'|o, a, b) \sum_{s'} O(s', o) \sum_s T(s, a, s') b(s)$$

- Fonction de récompense sur les belief states:

$$\rho(b) = \sum_s b(s) R(s)$$

- Donc, on a un MDP observable sur l'espace de belief states.
- Une politique optimale pour ce MDP est une politique optimale pour le POMDP.

29

POMDP comme un MDP

- Toutefois, la résolution du MDP sur l'espace des belief states n'est pas facile, car l'espace est maintenant continu (distribution de probabilités).
- Les algorithmes vus pour les MDP ne fonctionnent pas directement.
- Quelques modifications ont été apportées aux algorithmes pour gérer le cas continu.
- Toutefois, la complexité est très élevée.
  - Avec quelques douzaines d'états, les problèmes deviennent impossibles à résoudre.

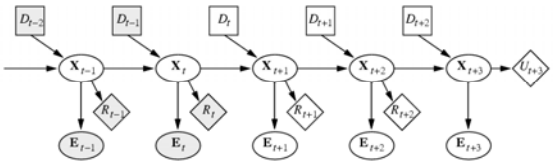
30

Réseau de décision dynamique

- Les modèles de transition et d'observation sont représentés par un réseau bayésien dynamique.
- Par la suite, on ajoute des nœuds de décision et d'utilité comme dans les réseau de décision
- Un algorithme de filtrage est utilisé pour incorporer les nouvelles observations et les nouvelles actions et pour mettre à jour le belief state.
- Les décisions sont prises en projetant les séquences d'actions possibles et en choisissant la meilleure.

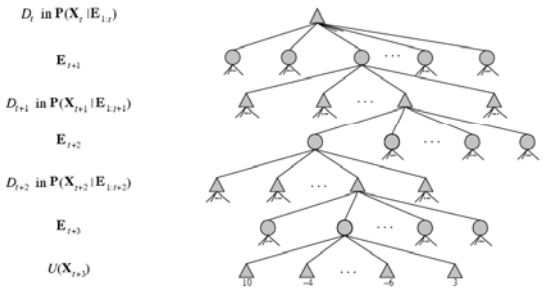
31

Réseau de décision dynamique



32

Réseau de décision dynamique



33