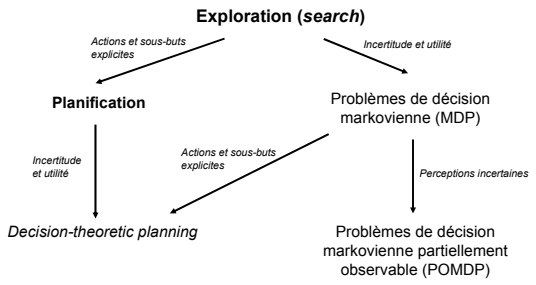
Processus de décisions complexes

IFT-17587
Concepts avancés pour systèmes intelligents
Luc Lamontagne

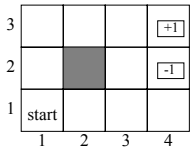
Plan

- Problèmes de décision séquentielle
- *Value iteration*
- *Policy iteration*

Problèmes de décision séquentielle



Exemple



- Imaginer un robot avec des perceptions locales seulement
  - Il peut se déplacer de A vers B
  - Mais les actions ont des résultats incertains
    - N'est pas certain de se diriger dans la direction voulue → Probabilité de 0.8
    - Déplacement à angle droit possible avec une probabilité de 0.1
  - On veut un robot qui décide comment naviguer dans la pièce
- Exemple de problème de décision séquentielle

Processus de décision markovien

- Habituellement désigné par l'acronyme MDP
  - *Markov decision process*
- État initial S_0
- Modèle de transition *T(s, a, s')*
  - Comment est-ce que Markov s'applique ici ?
    - La probabilité d'atteindre *s'* à partir de *s* dépend seulement de *s* et d'aucun autre état de l'historique
  - L'incertitude est possible
    - Transition → probabilité
- Fonction de récompense R(s) (*Reward Function*)
  - Définie pour chaque état

Construire une solution : politique

- Spécifier une solution pour chaque état
  - Indique ce que l'agent doit faire pour chaque état qu'il peut atteindre
  - Politique (π)
    - La meilleure action recommandée pour chaque état
    - Politique dans l'état *s* → $\pi(s)$
  - Politique complète
    - Recouvre tous les états potentiels
  - Une politique optimale π^*
    - Celle qui a la plus grande utilité espérée
      - Les transitions sont stochastiques

Utiliser une politique

- Un agent dans l'état *s*
  - *s* est la seule perception disponible à l'agent
  - $\pi^*(s)$ propose une action qui maximise l'utilité espérée
- La politique représente la fonction de l'agent
  - Elle est une description d'un simple réflexe

Exemples de politique

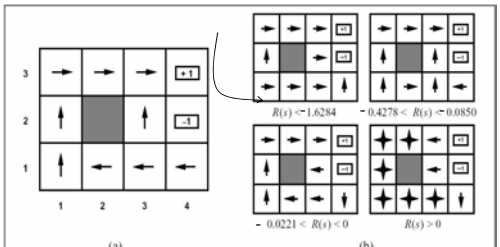


Figure 17.2 (a) An optimal policy for the stochastic environment with $R(s) = -0.04$ in the nonterminal states. (b) Optimal policies for four different ranges of $R(s)$.

Maintenir un équilibre

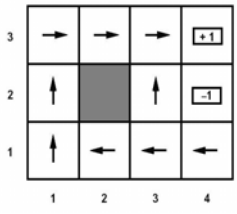
- Différentes politiques établissent un équilibre entre le risque et la récompense
  - Seulement intéressant pour les environnements non-déterministes
- Construire une politique optimale est cependant un problème difficile

Attributs d'optimalité

- On veut trouver une politique qui maximise l'utilité d'un agent durant sa durée de vie
  - Maximiser $U([s_0, s_1, s_2, \ldots, s_n)]$
- Mais est-ce que la durée de vie est connue?
  - Horizon fini
    - Le nombre de transition d'état est connu
    - Après un certain nombre d'étape, ça ne fait plus de différence
    - $U([s_0, s_1, s_2, \ldots, s_n)] = U([s_0, s_1, s_2, \ldots, s_n, s_{n+1}, s_{n+k})]$ pour tout k>0
  - Horizon infini
    - Toujours la possibilité pour plus de transitions d'état

Horizon temporel

- Considérons la case (3, 1)
  - Soit horizon = 3
  - Soit horizon = 8
  - Soit horizon = 20
  - Soit horizon = infini
  - Est-ce π^* change?



- Politique optimale non stationnaire

Évaluer une séquence d'états

- Hypothèse de stationnarité
  - Si j'affirme que demain je préfèrerai *a* à *b*
  - Alors je dois aussi affirmer qu'aujourd'hui je préfère *a* à *b*
  - Les préférences des états sont stationnaires
- Récompenses additives
  - $U[(a, b, c, \ldots)] = R(a) + R(b) + R(c) + \ldots$
- Récompenses escomptées
  - $U[(a, b, c, \ldots)] = R(a) + \gamma R(b) + \gamma^2 R(c) + \ldots$
  - γ est le facteur d'escompte entre 0 et 1
    - Quelle est sa signification ?

Évaluer une politique

- Chaque politique, π, génère plusieurs séquences d'états possibles
  - L'incertitude des transitions selon *T(s, a, s')*
- La valeur de la politique
  - Une somme pondérée des récompenses escomptées
  - Observée pour toutes les séquences d'états

$$\pi^* = \underset{\pi}{\operatorname{argmax}} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]$$

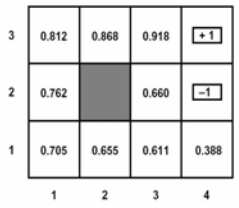
L'utilité des états

- Utilité d'un état *s*
  - L'utilité espérée des séquences d'états qui peuvent être suivies à partir de *s*
    - La séquence d'état subséquente est une fonction de π(s)
- L'utilité pour une politique particulière π est

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

Exemple

- Soit γ = 1 et R(s) = -0.04
  - Les utilités sont plus élevés près du but
  - Comporte moins d'étape à -0.04 dans le calcul de la somme



Reformulation de la politique

- Aller dans l'état avec le maximum d'utilité espérée
  - État atteignable avec utilité élevé mais qui peut avoir une faible probabilité d'être atteint
  - Est fonction des actions disponibles, de la fonction de transition et des états résultants

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

Reconversion du problème

- L'utilité d'un état est

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

- La politique est l'utilité espérée maximum

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

- Ainsi, l'utilité d'un état est

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

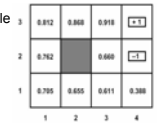
  - Proposé par Richard Bellman (Équations de Bellman - 1957)
  - Moins coûteux à évaluer

Exemple d'équations de Bellman

- Utilité de la cellule (1, 1)

$$U(1,1) = -0.04 + \gamma \max\{\ 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), \quad (Up)$$
$$0.9U(1,1) + 0.1U(1,2), \quad (Left)$$
$$0.9U(1,1) + 0.1U(2,1), \quad (Down)$$
$$0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ \} \quad (Right)$$

- Résolution
  - On se retrouve avec un système d'équations
    - Dans notre cas – 9 :une pour chaque cellule non terminale
  - Difficile à résoudre à cause de la non linéarité
    - Opérateur *max*
    - Pas possible d'utiliser l'algèbre linéaire
  - On doit utiliser une approche itérative



Résolution des équations de Bellman :

Value iteration

- On commence avec des valeurs arbitraires pour l'utilité des états

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

- On met à jour l'utilité de chaque état en fonction de ses voisins

$$U(1,1) = -0.04 + \gamma \max\{\ 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), \quad (Up)$$
$$0.9U(1,1) + 0.1U(1,2), \quad (Left)$$
$$0.9U(1,1) + 0.1U(2,1), \quad (Down)$$
$$0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ \} \quad (Right)$$

- On répète ce processus jusqu'à qu'un équilibre soit atteint

Résolution des équations de Bellman :

Value iteration

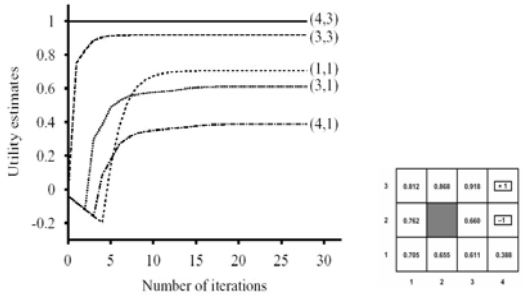
function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
inputs: mdp, an MDP with states S, transition model T, reward function R, discount γ
ϵ, the maximum error allowed in the utility of any state
local variables: U, U', vectors of utilities for states in S, initially zero
δ, the maximum change in the utility of any state in an iteration

repeat
$U \leftarrow U'$; $\delta \leftarrow 0$
for each state s **in** S **do**
$U'[s] \leftarrow R[s] + \gamma \max_a \sum_{s'} T(s,a,s')\, U[s']$
if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1-\gamma)/\gamma$
return U

Figure 17.5

Résolution des équations de Bellman :

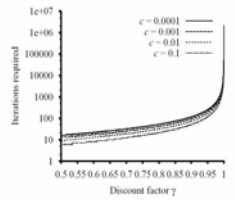
Value iteration - convergence



Résolution des équations de Bellman :

Value iteration - convergence

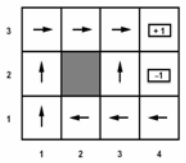
- Optimalité
  - Après *k* mise à jour de Bellman
  - Plus de détail dans le livre sur le calcul de l'erreur au temps i
    - Fonction de l'erreur au temps i-1 et du facteur d'escompte γ



Résolution des équations de Bellman :

Policy Iteration

- Supposons qu'une action est nettement meilleure que toutes les autres
  - Est-ce important de déterminer exactement les utilités ?
- Autrement formulé :
  - Supposons qu'on vous donne une politique
  - Est-ce qu'elle est bonne ?
    - On présume qu'on connaît γ et *R*
    - Comment s'y prendre ?
- On exécute itérativement :
  - L'évaluation de la politique
  - L'amélioration de la politique



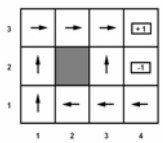
Résolution des équations de Bellman :

Policy iteration

- Évaluation d'une politique
  - Calcul de l'utilité pour chaque état selon l'équation de Bellman

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

  - Pas de *max* !
    - La politique fixe l'action à prendre.
    - Donc plus simple à résoudre
      - Techniques d'algèbre linéaire
      - *Value iteration*



Résolution des équations de Bellman :

Policy iteration

- Amélioration d'une politique
  - Lorsqu'on connait les *U(s)*
  - Pour chaque s, on compare

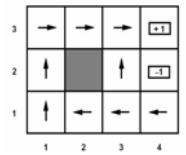
$$U_i(s) = R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s') U_i(s')$$

avec l'utilité obtenu par les autres actions.

  - Si une autre action donne de meilleures résultats

$$\max_a \sum_{s'} T(s, a, s')\ U[s']$$

    - On met à jour la politique $\pi(s) \leftarrow argmax_a$



Résolution des équations de Bellman :

Policy iteration

function POLICY-ITERATION(*mdp*) **returns** a policy
  inputs: *mdp*, an MDP with states $\mathcal{S}$, transition model T
  local variables: U, U', vectors of utilities for states in $\mathcal{S}$, initially zero
    π, a policy vector indexed by state, initially random

repeat
  $U \leftarrow$ POLICY-EVALUATION(π, U, *mdp*)
  unchanged? $\leftarrow$ true
  for each state s **in** $\mathcal{S}$ **do**
    if $\max_a \sum_{s'} T(s,a,s')\ U[s'] > \sum_{s'} T(s,\pi[s],s')\ U[s']$ **then**
      $\pi[s] \leftarrow \text{argmax}_a \sum_{s'} T(s,a,s')\ U[s']$
      unchanged? $\leftarrow$ false
until *unchanged?*
return P

Figure 17.9

Résolution des équations de Bellman :

Policy iteration

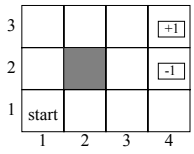
- Souvent l'approche la plus efficace
  - Pour un petit espace d'état → $O(n^3)$
  - Des approximations sont possibles
    - Au lieu de résoudre *U* exactement, on peut l'approximer rapidement avec des techniques itératives
    - Explorer et mettre à jour la politique seulement pour un sous-ensemble de l'espace d'états
      - Ne pas perdre de temps à mettre à jour les portions qu'on pense qui sont mauvaises

MDP et applications réelles

- Les hypothèses des MDP : on connaît
  - L'état *s* dans lequel est l'agent
  - La récompense de *s*
  - Les actions possible *a*
  - La fonction de transition *t (s, a, s')*
- Plusieurs applications → partiellement observable
  - On ne connaît pas toujours l'état dans lequel on est;
  - Peut être difficile de calculer les utilités ou de déterminer les actions possibles
- POMDP
  - *Partially Observable* MDP
  - Prend en compte un modèle de l'incertitude sur les états

Notre exemple en POMDP

- Aucune connaissance de l'état (ou connaissance partielle)
- Le robot n'a pas une idée exacte de l'état dans lequel il est.
- Quelle serait une bonne politique ?



POMDP :

Idée générale

- Pour modéliser l'incertitude sur les états
  - Modèle d'observation *O(s, o)*
    - Spécifie la probabilité de percevoir l'observation *o* lorsque dans l'état *s*
      - Dans notre exemple, *O()* ne retourne rien avec probabilité 1
  - Mise à jour des croyances

$$b'(s') = \alpha\, O(s', o) \sum_{s} T(s, a, s') b(s)$$

  - Déterminer une politique
    - Filtrage sur les croyances d'états
    - Optimisation des *argmax*

Conclusion

- Les problèmes de décision séquentielle sont également appelés MDP
- Ils sont définis par des modèles de transition *T* et une fonction de récompense *R*
- La solution d'un MDP est une politique qui associe la décision d'un agent à chaque état
- Un politique optimale est celle qui maximise l'utilité des séquences d'états possibles
- Les algorithmes *value iteration* et *policy iteration* permettent de résoudre les MDPs
- Les POMDPs prennent en compte l'incertitude sur les états et sont plus difficiles à résoudre.