Apprentissage par renforcement

Plan

- Tâche d'apprentissage
- Apprentissage par renforcement passif
- Q-Learning
- Fonction d'approximation

Apprentissage par Renforcement (RL)

- Imaginons que vous jouez un nouveau jeu dans lequel votre partenaire vous guide en vous disant :
  - Que c'est bon ou mauvais pour chaque coup
  - Que vous avez perdu ou gagné à la fin de chaque jeu
- La question est de savoir si en répétant le jeu vous seriez capable de la battre sans qu'il vous aide cette fois-ci : la réponse est oui

Apprentissage par renforcement

- L'agent apprend à l'aide de ses expériences dans son environnement.
- Pour chaque action, l'agent reçoit une récompense ou une pénalité.
- Le but de l'agent est d'apprendre la suite d'actions qui lui procure la plus grande somme (espérée) de récompenses.

4

Tâche d'apprentissage

- L'agent peut percevoir dans son environnement un ensemble S d'états distincts (donc MDP).
- Il a un ensemble d'actions A qu'il peut exécuter.
- À chaque étapes t,
  - L'agent perçoit l'état courant s_t,
  - Il choisit l'action courante a_t et l'exécute.
  - L'environnement répond en donnant à l'agent une récompense $r_t = r(s_t, a_t)$ et en produisant l'état suivant : $s_{t+1} = \delta(s_t, a_t)$
- Les fonctions r et δ font parti de l'environnement et elles ne sont pas nécessairement connues de l'agent.

5

Tâche d'apprentissage

- La tâche de l'agent est d'apprendre une politique pour sélectionner la prochaine action a_t en se basant sur l'état courant s_t. $\pi : S \rightarrow A$
- La politique que nous voulons que l'agent apprenne est la politique qui va donner la plus grande somme (espérée) de récompenses pour l'agent.
- Définissons la fonction suivante qui donne la somme des récompenses pour une certaine politique et pour un certain état de départ.

$$
\begin{aligned}
V^{\pi}(s_t) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \\
&\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}
\end{aligned}
$$

6

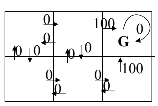
Tâche d'apprentissage (suite)

- Le but de l'agent est d'apprendre la politique optimale, celle qui maximise la somme des récompenses pour tous les états *s*.

$$\pi^* \equiv \max_{\pi} V^{\pi}(s), (\forall s)$$

7

Exemple



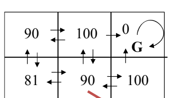
r(s,a) (récompense immédiate)

Attention : la détermination de la politique passe par *T(s',a,s)*

→	→	G
→	→	↑

Une politique optimale

V(s)*



$$0 + \gamma 100 + \gamma^2 0 + \gamma^3 0 + ... = 0.9 \times 100 = 90$$

8

Apprentissage par renforcement passif

- En apprentissage passif, la politique de l'agent est fixe, c'est-à-dire que dans l'état s, il effectue toujours l'action $\pi(s)$.
- Son but est d'apprendre à quel point la politique est utile: $U^{\pi}(s)$ ou $V^{\pi}(s)$.
- L'agent ne connaît pas le modèle de transition $T(s, a, s')$ et la fonction de récompense $R(s)$.

9

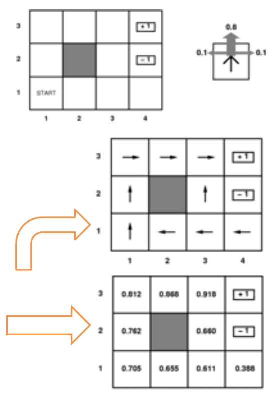
Essais

- Dans cet environnement, le but de l'agent est d'apprendre l'utilité espérée de chacun des états non terminaux.
- Pour ce faire, il effectue des essais:

$$(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$$
$$(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$$
$$(1,1)_{-.04} \rightarrow (2,1)_{-.04} \rightarrow (3,1)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (4,2)_{-1}\ .$$

Exemple

- L'agent reçoit un renforcement de -0.04 dans chaque état, sauf les états finaux.
- Les deux dernières figures représentent la politique de l'agent et les utilités qu'il a apprises.



Essais

- Dans cet environnement, le but de l'agent est d'apprendre l'utilité espérée de chacun des états non terminaux.
- Pour ce faire, il effectue des essais:

$$(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$$
$$(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$$
$$(1,1)_{-.04} \rightarrow (2,1)_{-.04} \rightarrow (3,1)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (4,2)_{-1}\ .$$

Estimation direct de l'utilité

- On conserve une moyenne des utilités pour chacun des états.
- Par exemple, avec le premier essai, on avait un exemple pour l'état (1,1) avec une utilité de 0.72, deux exemples de 0.76 et 0.84 pour l'état (1,2), etc.
- Cette méthode a tendance à converger très lentement, car elle ne considère pas l'interaction entre les états, comme le fait.

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

13

Programmation dynamique adaptive

- Apprentissage du modèle de transition.
- Apprend la probabilité d'arriver dans l'état s' étant donné une action a dans un état s : $T(s,\pi(s),s)$
- Dans l'exemple, l'action Droite a été effectuée trois fois en *(1,3)* et dans deux cas l'état résultant était (2,3), donc *T((1,3),Droite,(2,3)) est estimée à 2/3.*

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

14

Apprentissage par différence temporelle (TD)

- Le but est d'ajuster la valeur des états en fonction des transitions effectuées.
  - Par exemple, supposons que dans un essai, on obtient $U^\pi(1,3) = 0.84$ *et* $U^\pi(2,3) = 0.92$.
  - Si cette transition survient tout le temps, on voudrait: $U^\pi(1,3) = -0.04 + U^\pi(2,3)$, donc $U^\pi(1,3)$ serait 0.88.
  - Son estimation de 0.84 est un peu basse et il faudrait donc l'augmenter.
- Lorsqu'une transition survient de l'état s à l'état s', on met à jour l'utilité avec:

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha(R(s) + \gamma U^\pi(s') - U^\pi(s))$$

15

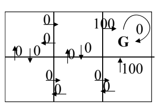
TD (suite)

- Mise à jour selon la règle « delta »: différence entre la sortie désirée et la sortie actuelle + $\boldsymbol{\alpha}$, le taux d'apprentissage
  - Sortie désirée : $\boldsymbol{R(s) + \gamma U^{\pi}(s')}$
  - Sortie actuelle : $\boldsymbol{U^{\pi}(s)}$

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

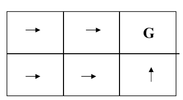
$$U^{\pi}(s) \leftarrow (1-\alpha)U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s'))$$

Exemple



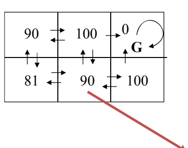
r(s,a) (récompense immédiate)

Attention : la détermination de la politique passe par *T(s',a,s)*



Une politique optimale

V(s)*



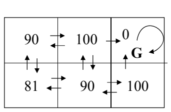
$$0 + \gamma 100 + \gamma^2 0 + \gamma^3 0 + \ldots = 0.9 \times 100 = 90$$

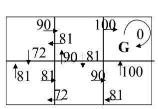
La fonction *Q*

- La fonction que l'agent veut apprendre, est la fonction Q qui représente la récompense immédiate obtenue en exécutant l'action a à l'état s, plus la valeur qui serait obtenue en suivant la politique optimale par la suite.

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a)) = r(s,a) + \gamma V^*(s')$$

V(s)*





Q(s,a)

$$V^*(s) = \max_{a'} Q(s,a')$$

Algorithme

Algorithme d'apprentissage de la fonction Q

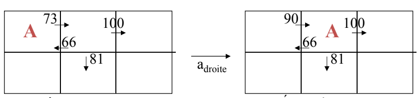
- Pour toutes les paires (s, a), initialiser l'entrée de la table $\hat{Q}(s, a)$ à zéro.
- Observer l'état courant s
- Faire pour toujours :
  - Sélectionner une action a et l'exécuter
  - Recevoir la récompense immédiate r
  - Observer le nouvel état s'
  - Mettre à jour l'entrée de la table $\hat{Q}(s, a)$ de la façon suivante :

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

  - $s \leftarrow s'$

19

Exemple



État initial: s_1 État suivant: s_2

$$\begin{aligned} \hat{Q}(s_1, a_{droite}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a') \\ &\leftarrow 0 + 0.9 \times \max\{66, 81, 100\} \\ &\leftarrow 90 \end{aligned}$$

20

Q-learning

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

Q-Learning avec TD

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s', a') - Q(s,a))$$

Re-Q-Learning avec TD

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

21

Convergence

- À chaque déplacement, l'agent met à jour son estimation de la valeur T pour la transition qu'il vient de prendre.
- Après un certain nombre d'itérations, la valeur de l'estimation sera proche de la valeur réelle de T.
  - La valeur de l'estimation ne peut pas diminuer
    $$(\forall s, a, n)\hat{Q}_{n+1}(s,a) \geq \hat{Q}_n(s,a)$$
  - La valeur de l'estimation de dépassera pas la valeur réelle de T
    $$(\forall s, a, n) 0 \leq \hat{Q}_n(s,a) \leq Q(s,a)$$

22

Résumé

- Apprentissage passif (´ est donnée):

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

- Apprentissage actif:

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

- Est-ce suffisant?

23

Stratégie d'exploration

- Si l'agent choisit toujours l'action qui maximise T :
  - Il va avoir tendance à toujours prendre le même chemin.
  - Il n'explorera pas les autres possibilités qui sont peut-être meilleures.

24

Stratégie d'exploration

- Pour favoriser l'exploration, on choisit aléatoirement une action, mais en donnant plus de chance aux actions ayant une grande valeur de T .

$$P(a_i|s) = \frac{k^{\hat{Q}(s,a_i)}}{\sum_j k^{\hat{Q}(s,a_j)}}$$

- n#A #B#est une constante. Plus n est grand, plus on favorise les valeurs de T élevées.
- En général, on commence avec une petite valeur pour n et on l'augmente graduellement. Donc, il y a plus d'exploration au début.

25

Fonction d'approximation

- Jusqu'à maintenant, on a considéré que les utilités étaient emmagasinées dans une table avec une entrée pour chaque état possible.
- Ça peut fonctionner avec 10 000 états, mais pas pour des problèmes plus complexes comme les échecs (10^{120}) ou le backgammon (10^{50}).
- Il serait absurde de vouloir visiter tous ces états pour pouvoir apprendre à jouer.

26

Fonction d'approximation

- Une idée est d'utiliser une fonction d'approximation, c'est-à-dire n'importe quelle autre représentation qu'un tableau.
- C'est une estimation parce qu'on ne sait pas si la vraie fonction d'utilité peut être représentée par la forme choisie.
- Exemple: Une fonction linéaire pondérée d'un ensemble de caractéristiques.

27

Fonction linéaire pondérée

- Supposons que l'on a ҁ caractéristiques: $(f_1,\ldots,f_n)$, alors la fonction d'utilité sera estimée par:

$$\hat{U}_\theta(s) = \theta_1\, f_1(s) + \theta_2\, f_2(s) + \cdots + \theta_n\, f_n(s)$$

- L'algorithme d'apprentissage par renforcement apprend les valeurs des paramètres „$_i$ pour estimer le mieux possible la vraie utilité.
- Par exemple, au lieu d'avoir 10^{120} valeurs dans un tableau, on a 20 valeurs de paramètres.
- C'est une compression énorme qui peut tout de même donner de très bon résultats.

28

Avantage

- Permet de représenter des fonctions d'utilité pour de très grands espaces d'états.
- Mais le plus grand avantage est la capacité de généralisation d'une fonction d'approximation.
- Par exemple, en examinant seulement 1 état tous les 10^{44} états possibles pour le jeu de backgammon, il est possible d'apprendre une fonction d'utilité permettant de battre les meilleurs joueurs humains au monde.

29

Règle delta

- Exemple: $\hat{U}_\theta(x,y) = \theta_0 + \theta_1 x + \theta_2 y$
- "delta": différence entre la sortie désirée et la sortie actuelle.
- En apprentissage par renforcement, l'agent met à jour les valeurs des paramètres à chaque expérience en minimisant l'erreur (la moitié) entre ce qui est prédit et ce qui est actuel: $Err_j = \frac{(\hat{U}_\theta(s) - u_j(s))^2}{2}$
- Où $u_j(s)$ est la valeur d'utilité observée (actuelle) pour l'expérience *j*.

30

Règle delta

- La vitesse de changement de l'erreur relativement à chacun des paramètres θ_i est alors :

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial Err_j(s)}{\partial \theta_i} = \theta_i + \alpha(u_j - \hat{U}_\theta(s)) \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i}$$

$$\frac{\partial \hat{U}_\theta(s)}{\partial \theta_i} \qquad \frac{\partial \hat{Q}_\theta(s,a)}{\partial \theta_i}$$

31

Règle delta (suite)

- On voudrait alors bouger chacun des paramètres dans la direction des moindres erreurs, par conséquent .

$$\theta_i \leftarrow \theta_i + \alpha[R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)] \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i}$$

$$\theta_i \leftarrow \theta_i + \alpha[R(s) + \gamma \max_{a'} \hat{Q}_\theta(s,a') - \hat{Q}_\theta(s,a)] \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i}$$

32

Règle delta

- Exemple: $\hat{U}_\theta(x,y) = \theta_0 + \theta_1 x + \theta_2 y$
- Dès lors, l'agent met à jour les valeurs des paramètres à chaque expérience selon la règle suivante:

$$\begin{aligned}
\theta_0 &\leftarrow \theta_0 + \alpha\,(u_j(s) - \hat{U}_\theta(s))\,, \\
\theta_1 &\leftarrow \theta_1 + \alpha\,(u_j(s) - \hat{U}_\theta(s))x\,, \\
\theta_2 &\leftarrow \theta_2 + \alpha\,(u_j(s) - \hat{U}_\theta(s))y\,.
\end{aligned}$$

- Où $u_j(s)$ est la valeur d'utilité observée (actuelle) pour l'expérience *j*.

33

Recherche de politique

- L'idée ici consiste à continuer à utiliser une politique tant que sa performance s'améliore, puis à s'arrêter.
- Nous nous intéressons aux représentations paramétrées de ´ qui ont beaucoup moins de paramètres qu'il n'y a d'états dans V.

Ex : $\pi(s) = \max_a \hat{Q}_\theta(s, a)$

Plus de détails pour IFT--7025 en p. 896 du livre.