Apprentissage par renforcement (Reinforcement Learning (RL)) Approche : Monte Carlo

T. AL-ANI
A²SI-ESIEE-PARIS

Plan :

- Partie I
  L'apprentissage par renforcement
- Partie II
  On-Policy Monte Carlo Control
- Partie III
  Off-Policy Monte Carlo Control

Partie I
L'apprentissage par renforcement

- Principe :

  Utiliser les erreurs et les réussites déjà produite pour améliorer sa politique.

- But :

  Choisir l'action optimal pour une situation donnée

Partie II :
On-Policy Monte Carlo Control

1. Algorithme de Monte Carlo
   Calcul de la fonction Politique

2. Application au jeux du Blackjack

3. Problèmes d'implémentation sous Scilab

1. Algorithme de Monte Carlo
Calcul de la fonction Politique

Initialisation :

Q(s,a) fonction d'évaluation
Returns(s,a) fonction de gratification
$\pi(s,a)$ politique : (soft policy) $0 \leq \pi(s,a) \leq 1$
'a*' actions qui maximisent la fonction d'évaluation
'a' autres actions
ϵ pondération des actions 'a' $0 \leq \epsilon \leq 1$ (fixé au départ)

1. Algorithme de Monte Carlo
Calcul de la fonction Politique

Répéter :
  Générer une partie (épisode) utilisant π
  Pour chaque paire s,a faire :
    R = Première occurrence de s,a
    Ajouter R à la mémoire Returns(s,a)
    Q(s,a) = average(Returns(s,a))
 Pour chaque s faire :
    $a^* = \text{argmax}_a Q(s,a)$
    Pour tout a Є A(s)
      $\pi(s,a) = 1 - \epsilon + \epsilon/|A(s)|$ si $a = a^*$
      $\pi(s,a) = \epsilon/|A(s)|$ si $a \neq a^*$

2. Application au jeux du Blackjack

- Fonction de Gratification (reward fonction):

1 si la partie est gagné
0 si la partie continue
-1 si la partie est perdue

- Tirage :

8 et 9 à la banque
5 et 6 pour le joueur
Etat de la partie s = score du joueur
Action a_1 tirer une carte, a_2 s'arrêter.

- La partie générée est:

1) Tirer une carte.
2) S'arrêter. Total 13 points -> partie perdue

- Apprentissage de l'algorithme :

Etat s_{11} Action a_1 Gratification 0 -> $Returns(s_{11}, a_1) = R_0 + 0 = 0$ -> $\pi(s_{11}, a_1)$ inchangé
Etat s_{13} Action a_2 Gratification -1 -> $Returns(s_{13}, a_2) = R_0 - 1 = -1$ -> $\pi(s_{13}, a_2)$ diminue

3. Problèmes d'implémentation sous Scilab

- Causes :

Les actions 'a' peuvent être des couples d'actions à x dimensions
Les états 's' a peuvent être des couples d'états à y dimensions
La fonction mémoire est un tableau à x + y + 1 dimentions

- Conséquences :

La dimension mémoire est de taille inconnue -> limitation de l'algorithme.

Utilisation des fonctions hypermatrices de Scilab pour la version 2 de Rltoolbox.

Partie III :
Off-Policy Monte Carlo Control

1. Evaluer une politique tout en suivant une autre

2. Algorithme

3. Problèmes de mise en application

1. Evaluer une politique tout en suivant une autre

- π = politique d'évaluation (à améliorer)
- π' = politique de comportement

- Condition : **π(s,a)>0 implique π'(s,a)>0**

$$V(s) = \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)} \cdot R_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)}}$$

2. Algorithme Initialisation

- Pour tout s Є S, a Є A(s):
- Q(s,a) ← sans importance
- N(s,a) ← 0 (numérateur de Q)
- D(s,a) ← 0 (dénominateur de Q)
- π ← politique déterministe arbitraire

2. Algorithme
Déroulement

- Sélectionner une politique π′ et générer un épisode:

$$s_0, a_0, R_1; s_1, a_1, R_2; (\ldots); s_{T-1}, a_{T-1}, R_T, s_T$$

- l ← dernier état de l'épisode pour lequel

$$a_l \neq \pi(s_l)$$

2. Algorithme Déroulement (Suite)

- Pour chaque paire s,a de l'épisode après l:

  t ← première occurrence après l de s,a

  $$w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$$

  N(s,a) ← N(s,a) + wRt

  D(s,a) ← D(s,a) + w

  Q(s,a) ← N(s,a) / D(s,a)

- Pour tout s Є S:

  $\pi(s) \leftarrow \text{argmax}_a Q(s,a)$

3. Problèmes de mise en application

- Implémentation facile sous scilab
- Importance du choix de la politique de comportement (soft)
- Apprend uniquement sur la fin d'un épisode
- Convergence extrêmement lente