Apprentissage par renforcement

Stéphane Airiau

Université Paris Dauphine

Plan de cette leçon

1. Approximation des fonctions V et Q
2. Trouver les paramètres de la fonction d'approximation de manière supervisée
3. Q-learning avec fonction Q approximée
4. Policy Gradient (algorithme `Reinforce`)

Motivations

- Dans l'algorithme de Q-learning, on stocke $Q(s,a)$ dans un tableau
  Dans le tp `Qsa` était une matrice 256×4
- Pour certains problèmes,
  - le nombre d'états est très grand (ex : jeu d'échec, go)
    ⇨ impossible de stocker le tableau
  - l'état est continu
    ⇨ discrétisation : méthode un peu violente
  - l'état est une image
- ⇨ besoin d'approximer Q

Approximation

Idée :

- Utiliser une fonction avec un petit nombre de paramètres θ pour approximer V ou Q

$$V(s) \approx V_\theta(s) = f_\theta(s)$$

$$Q(s,a) \approx Q_\theta(s,a) = g_\theta(s,a)$$

- par exemple, f_θ peut être
  - un réseau de neurones (par exemple dans αGo)
    Le nombre de poids à apprendre est surement (beaucoup) plus petit que la taille du tableau
  - un arbre de décision (les points de split et les valeurs des feuilles)
  - une fonction simple (linéaire, polynome, bases de fourier)

Approximation

- la mise à jour $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max'_a Q(s',a') - Q(s,a))$ est un calcul simple

⇨ on va utiliser des méthodes plus sophistiquées pour la mise à jour. la mise à jour pour s va aussi affecter les autres états (et peut être généraliser quelque chose d'appris)

- Voir la mise à jour comme un problème supervisé $Q(s,a)$ est notre donnée, r est l'étiquette et on nourris un algorithme d'apprentissage supervisé à chaque mise à jour

⇨ il faut utiliser des méthodes qui peuvent apprendre online ou de manière incrémentale.

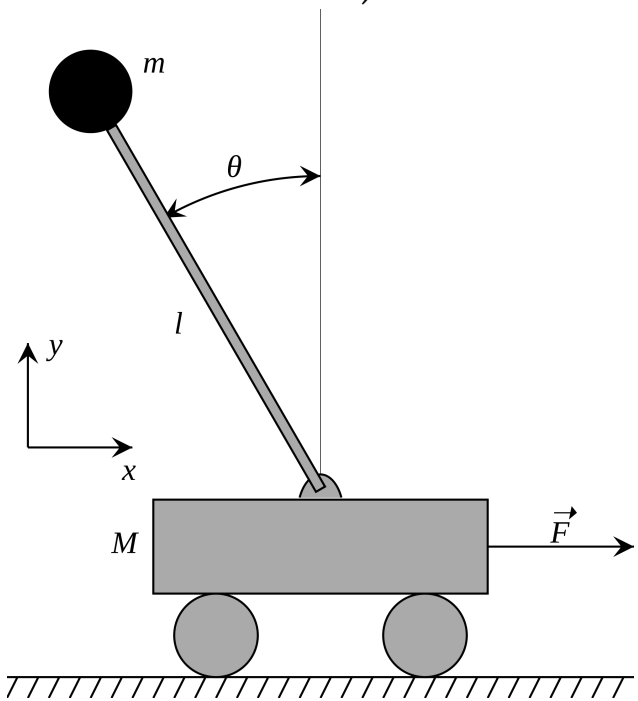
Pour cette leçon, on va se concentrer sur l'approximation linéaire

mais on peut utiliser les mêmes idées pour des approximations plus sophistiquées.

Le cas du cart-pole

Maintenir un pendule inversé en haut en déplaçant une caisse roulante.

➪ problème classique pour l'apprentissage par renforcement
(nombreuses vidéos sur internet)



- l'état a 4 parameters : position, velocity, angle, velocity angular.
- deux action : aller à gauche ou à droite
- pour simplifier, on considère que deux paramètres : vélocity v et angle α. L'état est donc le couple $s = (v, \alpha)$.

Le cas du cart-pole

On cherche à approximer la fonction V,

- *approche naïve*
  - $\theta = (\theta_1, \theta_2)$. $V_\theta(s) = \theta_1 \times v + \theta_2 \times \alpha$.
    Est-ce réaliste ?

- *Autre modélisation*
  - supposons que $v \in [-1,1]$ et $\alpha \in [-\pi, \pi]$.
    On crée trois variables pour chaque paramètre :
    - $v_{petit} = \frac{1}{1+(v+1)^2}$, $v_{moyen} = \frac{1}{1+v^2}$, $v_{grand} = \frac{1}{1+(v-1)^2}$
    - $\alpha_{petit}, \alpha_{moyen}, \alpha_{grand}$
  - On nomme la fonction $\phi(s) = \phi(v, \alpha) = (v_{petit}, v_{moyen}, v_{grand}, \alpha_{petit} \dots)$
  - $V_\theta(s) = \phi(s) \cdot \theta = v_{petit} \times \theta_1 + \dots$ avec $\theta = (\theta_1 \dots \theta_6)$.
  - On appelle parfois ca des fonctions d'appartenance floue, ou des noyaux...

Approximation de la fonction $Q(s,a)$

- L'action est de choisir gauche ou droite $a \in \{0,1\}$
- $\phi(s,a) = (v_{petit}, v_{moyen}, v_{grand}, \alpha_{petit} \ldots, \alpha_{grand}, a)$
- $Q_\theta(s,a) = \phi(s,a) \cdot \theta$ avec $\theta = (\theta_1 \ldots \theta_7)$.
- Qu'en pensez-vous ?
  Pensez-vous que le Q^* soit modélisable ainsi ?

- Meilleure solution : $\theta = (\theta_1 \ldots \theta_{12})$ avec
  $\phi(s,a) = (v_{petit} \times a, v_{petit} \times (1-a), v_{moyen} \times a, v_{moyen} \times (1-a), \ldots)$.

Approximation de la fonction $Q(s,a)$

- L'action est de choisir gauche ou droite $a \in \{0,1\}$
- $\phi(s,a) = (v_{petit}, v_{moyen}, v_{grand}, \alpha_{petit} \ldots, \alpha_{grand}, a)$
- $Q_\theta(s,a) = \phi(s,a) \cdot \theta$ avec $\theta = (\theta_1 \ldots \theta_7)$.
- Qu'en pensez-vous ?
  Pensez-vous que le Q^* soit modélisable ainsi ?
- Réponse : si on fait cela, alors $Q(s,1) = Q(s,0) + \theta_7$,
  ⇨ l'action 1 est toujours jugée meilleure que 0 si $\theta_7 > 0$, et l'inverse sinon !!!
  Ca ne va pas !
- Meilleure solution : $\theta = (\theta_1 \ldots \theta_{12})$ avec
  $\phi(s,a) = (v_{petit} \times a, v_{petit} \times (1-a), v_{moyen} \times a, v_{moyen} \times (1-a), \ldots)$.

Objectif

- L'objectif de notre prédiction est de minimiser une erreur
- on utilise généralement l'erreur moyenne quadratique (*" Mean Squared Value Error"*)

$$\sum_{s \in S} [v(s) - v_\theta(s)]^2$$

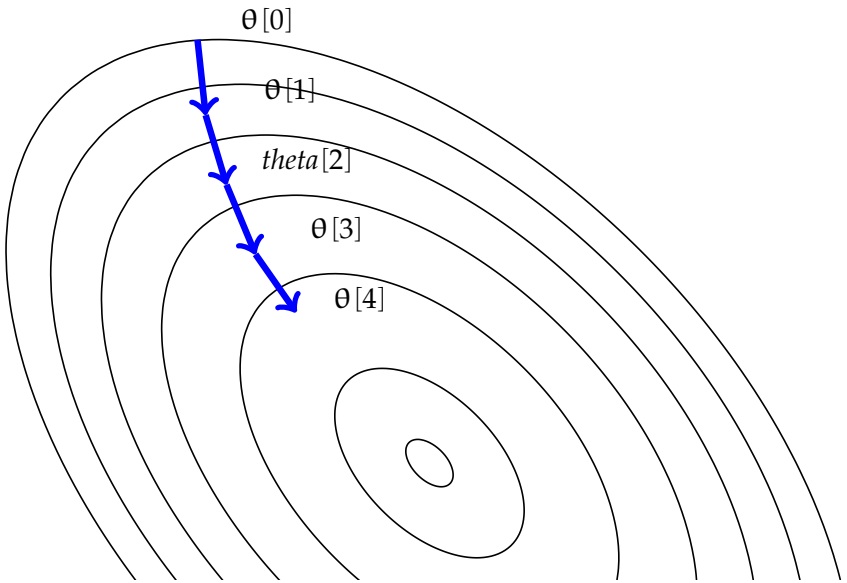
- sauf qu'on ne connaîtra jamais $v(s)$ exactement !

Comment trouver les paramètres θ de façon supervisée

- Supposons, de façon *irréaliste*, qu'on possède une base de triplets (s_t, a_t, q_t) où $q_t = Q^*(s_t, a_t)$.
  ⇨ un superviseur nous a donné des valeurs de q_t.
- Cas plus vraissemblable :
  on observe un agent, et on récolte de nombreuses traces $(s_1, a_1, r_1, s_2, a_2, r_2 \ldots)$.
  ⇨ pour chaque s_i, on peut estimer $\hat{Q}^\pi(s_i, a_i) = \sum_{t=i} \gamma^{t-i} r_{t-i}$
  ⇨ une estimation de q_t.
- On veut trouver une politique qui fasse au moins aussi bien
  amélioration de la politique ⇨ $\pi' = argmax Q^\pi(s, a)$

Pour simplifier, supposons que $\theta = (\theta_1, \theta_2)$.
Idéalement, on voudrait θ tel que $Q_\theta(s_t, a_t) = \phi(s_t, a_t) \cdot \theta = q_t \ \forall t$.
impossible ! ⇨ trouver θ tel que $\theta = \arg\min_\theta \sum_t (Q_\theta(s_t, a_t) - q_t)^2$ soit minimal. c'est un problème de régression.

Descente de gradient de l'erreur



le gradient indique la direction de la pente la plus forte
La règle de mise à jour sera donc :

$$\vec{\theta} \leftarrow \vec{\theta} + \Delta\vec{\theta} \qquad \theta_i \leftarrow \theta_i + \Delta\theta_i$$

$$\text{où } \Delta\vec{\theta} = -\eta\nabla E(\vec{\theta}) \qquad \Delta\theta_i = -\eta\frac{\partial E}{\partial\theta_i}$$

- η est le taux d'apprentissage qui determine la taille du pas que l'on fait en descendant
- le signe négatif indique que l'on veut descendre

Gradient

$\nabla (Q_\theta(s_t,a_t) - q_t)^2 = 2\,(Q_\theta(s_t,a_t) - q_t)\,\phi(s_t,a_t)$.
Donc

- $\theta \leftarrow 0$
- répéter N fois
  - prendre un triplet (s_t,a_t,q_t) au hasard dans la base
  - Calcule $d = (Q_\theta(s_t,a_t) - q_t)$
  - $\theta \leftarrow \theta - \alpha.d.\phi(s_t,a_t)$

  (dans le cas où $Q_\theta = \phi(s,t)\cdot\theta$), sinon il faut prendre la dérivée partielle.

On peut attendre d'avoir vu tous les états pour faire la mise à jour ou bien, on peut le faire à chaque nouvel exemple :
⇨ on parle alors de gradient stochastique

Q-Learning avec fonction Q approximée

- $$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$
- Cet Q-learning step rapproche un peu $Q(s_t,a_t)$ de $r_{t+1} + \gamma \max_{a'} Q(s_{t+1},a')$
- Idée : On veut rapprocher "un peu" $Q_\theta(s_t,a_t)$ de $(r_{t+1} + \gamma \max_{a'} Q_\theta(s_{t+1},a'))$
- Donc on fait un "gradient step" pour minimiser $(Q_\theta(s_t,a_t) - r_{t+1} - \gamma \max_{a'} Q_\theta(s_{t+1},a'))^2$.
- Pour simplifier, on pose $y_t = r_{t+1} - \gamma \max_{a'} Q_\theta(s_{t+1},a')$ et on fait un gradient pour minimiser $(Q_\theta(s_t,a_t) - y_t)^2$

Quelques problèmes

- Rappel $\nabla \left(Q_\theta(s_t,a_t) - y_t\right)^2 = 2\left(Q_\theta(s_t,a_t) - y_t\right)\phi(s_t,a_t)$
-

1 Prendre une action a et observe (s,a,s',r)
2 $y = r(s,a) + \gamma \max_{a'} Q_\phi(s',a')$
3 $\phi \leftarrow \phi - \alpha \frac{\partial Q_\phi}{\partial \phi}(s,a)\left(Q_\phi(s,a) - y\right)$

- Problèmes avec cet algorithme :
  1. les échantillons dépendent les uns des autres. Pour que le gradient stochastique converge, il faudrait qu'ils soient tirés aléatoirement et indépendamment.
  2. Minimiser $\left(Q_\theta(s_t,a_t) - y_t\right)^2$ n'est pas équivalent à minimiser $\left(Q_\theta(s_t,a_t) - r_{t+1} - \gamma \max_{a'} Q_\theta(s_{t+1},a')\right)^2$, puisque Q est fixé dans le terme de droite.
  3. $\max_{a'} Q(s',a')$ n'est pas un estimateur non biaisé.

Quelques solutions

1. Experience Replay
2. Double Q-learning.

Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Double q-learning

- Use two Q-functions: Q^A and Q^B

$$Q^A_{t+1}(s,a) \leftarrow Q^A_t(s,a) + \alpha\Big(r + \gamma Q^B_t(s', a^*) - Q^A_t(s,a)\Big) \quad \text{or}$$
$$Q^B_{t+1}(s,a) \leftarrow Q^B_t(s,a) + \alpha\Big(r + \gamma Q^A_t(s', b^*) - Q^B_t(s,a)\Big)$$

where $a^* = \arg\max_a Q^A(s', a)$

$b^* = \arg\max_a Q^B(s', a)$

Policy Gradient (Reinforce)

- On veut apprendre directement la politique.
- Elle doit être stochastique pour que son espérance soit dérivable $\pi_\theta(s,a)$.
- Comme pour la value function, on cherche à utiliser $\phi(s,a)^T\theta$.
- Contrairement à la value function, on doit avoir $\sum_a \pi_\theta(s,a) = 1$.
  On pourrait prendre $\pi_\theta(s,a) = \frac{\phi(s,a)^T\theta}{\sum_{a'}\phi(s,a')^T\theta}$.
  Dans la pratique, on utilise plutot $\pi_\theta(s,a) = \frac{e^{\phi(s,a)^T\theta}}{\sum_{a'}e^{\phi(s,a')^T\theta}}$ (c'est un softmax).
- On cherche la politique $\pi_\theta(s,a)$ qui optimise $J(\theta) = V^{\pi_\theta}(s_0) = \mathbb{E}_{\pi_\theta}\left[r_0 + \gamma r_1 + \gamma^2 r_2 \ldots\right]$
- Une solution serait de faire des petites perturbations sur θ :

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

  et d'en déduire un gradient et d'aller vers ce gradient. Pas très efficace.

Policy Gradient (Reinforce)

- Tentons de calculer le gradient de $J(\theta)$
- Pour simplifier énormément, on calcule la récompense au bout d'un seul pas de temps ! ! !
- $J(\theta) = \mathbb{E}_{\pi_\theta}[r_0]$
- Alors on a $J(\theta) = \sum_a \pi_\theta(s_0,a) r(s_0,a)$
- Et donc $\nabla_\theta J(\theta) = \sum_a \nabla_\theta \pi_\theta(s_0,a) r(s_0,a)$. Cette forme est inexploitable.
- On utilise le truc suivant :

$$\begin{aligned} \nabla_\theta \pi_\theta(s,a) &= \pi_\theta(s,a) \frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} \\ &= \pi_\theta(s,a) \nabla_\theta log \pi_\theta(s,a) \end{aligned}$$

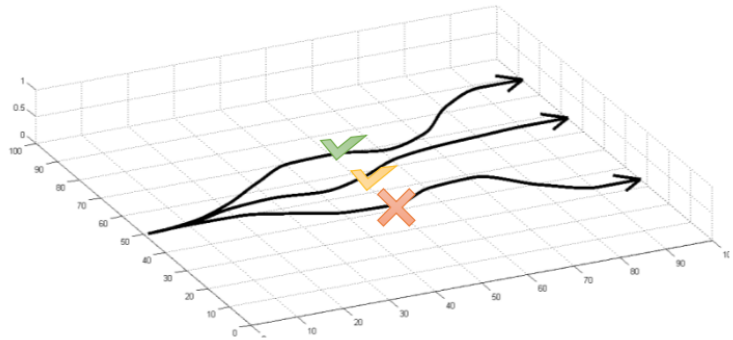
, ce qui donne :

$$\nabla_\theta J(\theta) = \sum_a \pi_\theta(s_0,a) \nabla_\theta \log \pi_\theta(s_0,a) r(s_0,a)$$

Policy Gradient (Reinforce)

Fonctionnement :

- on sample des trajectoires,
- celles qui donne un haut retour auront du bonus



Policy Gradient (Reinforce)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return v_t as an unbiased sample of $Q^{\pi_\theta}(s_t, a_t)$

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

function REINFORCE
    Initialise θ arbitrarily
    for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
        for $t = 1$ to $T - 1$ **do**
            $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
        end for
    end for
    return θ
end function

Policy Gradient (Reinforce)

Pour le gradient de π_θ, on a :

$$\nabla_\theta \log \pi_\theta(s,a) = \phi(s,a) - \frac{\nabla_\theta \sum_{a'} e^{\phi(s,a')^T \theta}}{\sum_{a'} e^{\phi(s,a')^T \theta}} = \phi(s,a) - \frac{\sum_{a'} \phi(s,a') e^{\phi(s,a')^T \theta}}{\sum_{a'} e^{\phi(s,a')^T \theta}}$$
$$= \phi(s,a) - \sum_{a'} \phi(s,a') \pi_\theta(s,a') = \phi(s,a) - \mathbb{E}_{a' \sim \pi(s,.)} \left[\phi(s,a') \right]$$

Problème de Reinforce :

- l'exploration n'est pas gérée.
- la variance est énorme