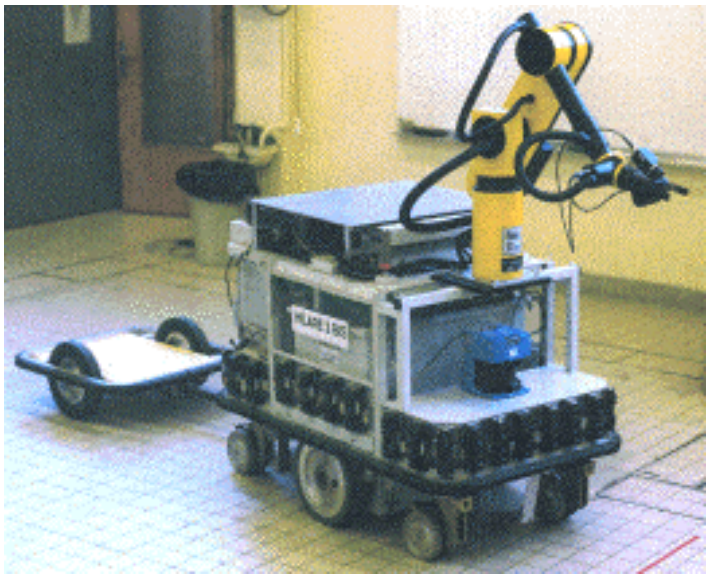
Apprentissage par renforcement
(Reinforcement Learning (RL))

Approche : Programmation Dynamique
(Dynamic Programming)

T. AL-ANI
A²SI-ESIEE-PARIS



SOMMAIRE

INTRODUCTION

Le but du projet est la réalisation d'une boite à outils « toolbox » d'apprentissage par renforcement sous SCILAB. Ceci consistait tout d'abord en l'implantation de différents algorithmes résolvant le problème d'apprentissage par renforcement, puis en le développement d'exemples utilisant ces algorithmes. Il fallait enfin rendre cette « toolbox » la plus conviviale possible par l'ajout de fichiers d'aide, de contrôle des erreurs, et de fenêtres facilitant cette utilisation.

Notre principale base documentaire pour l'aspect théorique du travail a été Reinforcement Learning: An Introduction, de Barto et Sutton [1], [2].

La principale base documentaire de la partie « technique » (codage des algorithmes, fichiers d'aide, vérification des paramètres, fenêtres) est la page Scilab du site de l'INRIA [3].

Première partie: APPRENTISSAGE PAR RENFORCEMENT

Dans cette partie nous présenterons tout d'abord les caractéristiques et les éléments de l'apprentissage par renforcement, puis le problème de l'apprentissage par renforcement.

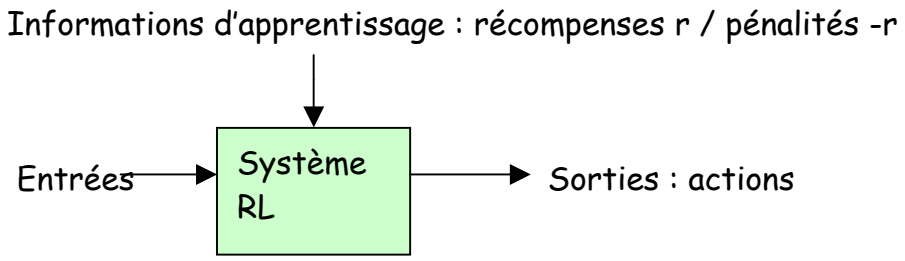
I - Caractéristiques et éléments d'apprentissage par renforcement

I-A - Caractéristiques de l'apprentissage par renforcement

I-A-1 - Qu'est-ce que l'apprentissage par renforcement ?

L'apprentissage par renforcement (en anglais reinforcement learning : RL) est un apprentissage :
- par interaction avec l'environnement : l'agent (celui ou ce qui apprend), par les actions qu'il effectue, modifie son environnement, qui l'informe de ces modifications, modifications qui vont influencer le comportement futur du système
- qui cherche à atteindre un but : on dit que l'apprentissage par renforcement est de type orienté objectif.
- de ce qu'il faut faire, en associant aux situations des actions dans le but de maximiser un signal numérique appelé récompense.

On peut donc voir un système RL de la façon suivante : il reçoit des entrées et des informations d'apprentissage en fonction desquelles il envoie des sorties, les actions :



I-A-2 - Principales caractéristiques de l'apprentissage par renforcement

L'agent apprend de lui-même quelle(s) action(s) effectuer pour atteindre son objectif. Cela distingue l'apprentissage par renforcement de l'apprentissage supervisé où l'apprentissage se fait par des exemples fournis par un superviseur extérieur.

Cette recherche des actions à effectuer est basée sur des essais et des erreurs. L'agent teste différentes actions et rejette celles qui ne sont pas bonnes (nous verrons comment distinguer ce qui est bien de ce qui est mal ultérieurement).

La récompense peut être retardée (c'est la valeur V), en sacrifiant les gains à court terme pour avoir des gains à longs termes plus importants. La récompense r peut ainsi être assimilée à la joie (si elle est grande) ou à la douleur (si elle est petite) immédiate, alors que les valeurs V correspondent à un jugement plus raffiné et à une vue sur le long terme du niveau de bonheur ou de malheur lorsqu l'environnement sera dans un état particulier.

L'agent doit explorer et exploiter, le problème étant d'équilibrer exploration et exploitation. En effet, pour augmenter la récompense, l'agent doit utiliser des actions qu'il a déjà essayé (donc dont il connaît la récompense) : c'est l'exploitation. Mais pour connaître ces actions, il doit tester des actions qu'il n'a pas encore sélectionner : c'est l'exploration. L'agent doit donc exploiter ce qu'il connaît déjà pour obtenir une récompense, mais il doit aussi explorer afin de choisir les meilleurs actions plus tard. Le problème est dû au fait que si l'agent ne fait qu'exploiter sans jamais explorer ou vice-versa, il échouera dans la tâche qui lui a été donné. Ajoutons que d'un point de vue stochastique, chaque action doit être testée plusieurs fois pour estimer correctement ses récompenses espérées.

L'apprentissage par renforcement considère le problème global d'un agent orienté - objectif qui agit avec un environnement qui n'est pas totalement connu. Cela le distingue d'autres approches (comme par exemple l'apprentissage

supervisé) qui s'occupent de sous - problèmes sans s'interroger sur leur correspondance avec un problème plus large.

I-A-3 - Exemples

- Contrôleur d'une raffinerie de pétrole : c'est un contrôleur adaptatif régulant en temps réel les paramètres d'une raffinerie. Il optimise le compromis produit/coût/qualité en se basant sur des coûts spécifiques secondaires, sans utiliser la consigne donnée au départ par l'ingénieur.

- Enfant qui apprend à marcher

- Robot mobile : c'est un robot décidant s'il doit entrer dans une autre pièce à la recherche d'un nouveau casier pour collecter des canettes ou tenter de trouver le chemin jusqu'à sa station pour recharger ses batteries. Il prend sa décision en se basant sur son expérience passée pour suivre le chemin le plus simple et le plus rapide afin de rejoindre cette station.

I-B - <u>Eléments d'apprentissage par renforcement</u>

I-B-1 - Généralités

Avec l'agent et l'environnement, un système d'apprentissage par renforcement comprend quatre sous - éléments :
- la politique P : ce que l'on doit faire
- la récompense r / pénalité -r : ce qui est bien / mal
- la valeur V : ce qui est bien car elle prédit la récompense
- le modèle de l'environnement M : quoi suit quoi

I-B-2 - La politique P

C'est le comportement de l'agent à un instant donné, c'est à dire une association entre un état donné de l'environnement (détecté par l'agent) et l'action à effectuer dans cet état, ceci pour chaque état de l'environnement.

En psychologie, la politique serait un ensemble de règles ou d'associations stimulus-réponse.

I-B-3 - La récompense r / pénalité -r

La récompense r est une fonction définissant une association entre les états de l'environnement mesurés par l'agent ou les couples état – action, et un unique nombre, récompense, appréciant intrinsèquement l'état. C'est une valeur immédiate qui définit les caractéristiques du problème vues par l'agent. Tel quelle, r est nécessairement fixée. Elle peut, malgré tout, être utilisée comme une base pour modifier la politique.

La pénalité -r est une valeur négative immédiate définissant elle aussi les caractéristiques du problème vues par l'agent.

I-B-4 - La valeur V

Contrairement à une fonction r qui indique ce qui est bien dans l'immédiat, une fonction valeur spécifie ce qui est bien dans l'avenir. La Valeur d'un état est la quantité totale des récompenses accumulées par l'agent dans l'avenir à partir de cet état.

Toutefois, V est plus difficile à déterminer que r : r est, normalement, donnée directement par l'environnement, contrairement à V qui doit être estimée et ré - estimée à partir des séquences d'observations (séquences d'états de l'environnement) que l'agent fait sur la totalité de sa vie (ou du processus).

I-B-5 - Le modèle de l'environnement M

Il imite l'environnement. Il peut, par exemple, étant donné un couple état – action prédire le prochain couple état – action.

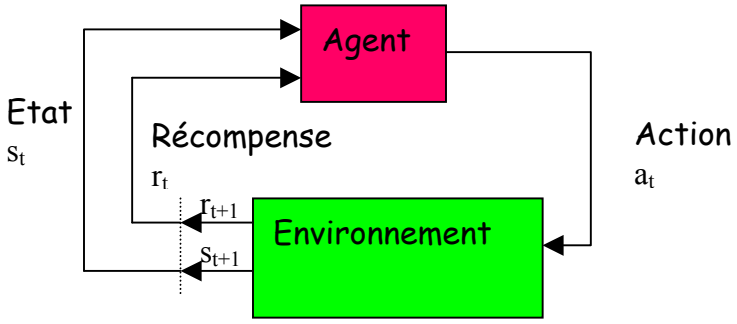
II - Le problème d'apprentissage par renforcement

II-A - L'interaction agent – environnement

L'apprenant et preneur de décision est appelé l'agent. La chose avec laquelle il interagit, qui comprend tout ce qui est hors de l'agent, est appelée environnement. Ils interagissent continuellement, l'agent en choisissant des actions et l'environnement en réagissant à ces actions et en présentant à l'agent de nouvelles situations. L'environnement renvoie également un signal appelé récompense, que l'agent cherche à maximiser.

Plus précisément, l'agent et l'environnement interagissent à des pas de temps discret,
t = 0, 1, 2, 3, … A chaque pas de temps t, l'agent reçoit une représentation de l'état de l'environnement $s_t \in S$, où S est l'ensemble des états possibles, et à partir de cela choisit une action $a_t \in A(s_t)$, où $A(s_t)$ est l'ensemble des actions

disponibles sous l'état s_t. Au pas suivant, en partie à cause de son action, l'agent reçoit une récompense numérique, $r_{t+1} \in \Re$, et se retrouve dans un nouvel état s_{t+1} :



II-B - Buts et récompenses

On définit la politique au pas t, π_t : elle associe aux états des probabilités d'action :
$\pi_t(s,a) = Pr(a_t = a / s_t = s)$.

Les méthodes d'apprentissage par renforcement précisent comment l'agent doit modifier sa politique en fonction de son expérience. L'objectif de l'agent est de maximiser la récompense à long terme.

Si la récompense r n'est peut-être pas la notion la plus adaptée à l'objectif, elle est malgré tout très flexible : c'est un moyen de communiquer à l'agent ce qu'il doit faire, mais pas comment le faire. Par exemple, pour apprendre à un robot comment échapper d'un labyrinthe, la récompense r est souvent 0 jusqu'à ce qu'il s'échappe, alors r devient +1.

Un objectif doit être hors du contrôle direct de l'agent, c'est-à-dire à l'extérieur de l'agent.

II-C - Le revenu R_t

Supposons que nous disposions de la séquence de récompense après le pas t suivante :
$r_{t+1}, r_{t+2}, r_{t+3}, \ldots$

En général, on souhaite maximiser le revenu espéré $E(R_t)$, pour chaque pas t.

II-C-1 - Tâche épisodique

C'est le cas d'une interaction agent - environnement qui peut être naturellement découpée en séquences ou épisodes, comme par exemple le fait de jouer à un jeu où de se promener dans un labyrinthe. On a alors : $R_t = r_{t+1} + r_{t+2} + \ldots + r_T$ (1)
où T est le dernier pas auquel un état terminal a abouti, à la fin de l'épisode.

II-C-2 - Tâche continue

C'est le cas où l'interaction n'a pas d'épisodes naturels. De ce fait T et R peuvent être infinis, rendant l'équation (1) inapplicable. On a donc :

$$R_t = r_{t+1} + \gamma \times r_{t+2} + \gamma^2 \times r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \qquad (2)$$

où γ, $0 \leq \gamma \leq 1$, est le taux de réduction : il détermine la valeur courante des revenus dans l'avenir : une récompense reçue au pas de temps t + k + 1 dans le futur aura une valeur égale à γ^k fois sa valeur immédiate.

Si $\gamma < 1$, la somme de l'équation (2) converge si la séquence des récompenses est bornée.

Si $\gamma \rightarrow 0$, l 'agent est « myope », dans le sens où il est concerné uniquement par la maximisation des récompenses immédiates : son objectif est, dans ce cas, d 'apprendre comment choisir a_t pour maximiser uniquement r_{t+1}.

Si $\gamma \rightarrow 1$, l 'agent est « hypermétrope », dans le sens où il est très fortement concerné par la maximisation des récompenses dans l'avenir.

II-D - La propriété de Markov

II-D-1 - Définitions

Si un état rassemble les observations ou sensations antérieures pour ne retenir que l'information essentielle , il a la Propriété de Markov. C'est le cas idéal. Cette définition peut se résumer par :
$Pr(s_{t+1} = s', r_{t+1} = r / s_t, a_t, r_t, \ldots, r_1, s_0, a_0) = Pr(s_{t+1} = s', r_{t+1} = r / s_t, a_t)$, pour tout s', r ainsi que l'historique $s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0$. (3)

Une tâche d'apprentissage par renforcement qui a la propriété de Markov est essentiellement un processus de décision de Markov (en anglais Markov Decision Process - MDP).

Si les ensembles d'états et des actions sont finis, alors le processus est fini.

II-D-2 - Définition d'un processus de décision de Markov

Pour définir un processus de décision de Markov, il faut préciser :
- l'ensemble des états et celui des actions
- une dynamique dite à un pas définie par les probabilités de transition :

$$P_{ss'}^{a} = \Pr\left(s_{t+1} = s' / s_t = s, a_t = a\right), \forall (s,s') \in S^2, \forall a \in A(s) \qquad (4)$$

- la récompense immédiate espérée :

$$R_{ss'}^{a} = E\left(r_{t+1} / s_t = s, a_t = a, s_{t+1} = s'\right), \forall (s,s') \in S^2, \forall a \in A(s) \qquad (5)$$

II-E - Fonction valeur et équation de Bellman

II-E-1 - Fonction valeur

La valeur d'un état est le revenu espéré en partant de cet état, elle est liée la politique de l'agent :

$$V^{\pi}(s) = E_{\pi}\left(R_t / s_t = s\right) = E_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} / s_t = s\right) \qquad (6)$$

La valeur du choix d'une action a dans un état s sous une politique π est le revenu espéré en partant de l'état s, en choisissant l'action a, et en poursuivant la politique π :

$$Q^{\pi}(s,a) = E_{\pi}\left(R_t / s_t = s, a_t = a\right) = E_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} / s_t = s, a_t = a\right) \qquad (7)$$

II-E-2 - L'équation de Bellman pour la politique π

Elle permet de déterminer $V^{\pi}(s), \forall s \in S$. On montre que :

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma \times V^{\pi}(s')\right] \qquad (8)$$

On a alors un système d'équations linéaires (une inconnue par état) qui a une unique solution : V^{π}. Cette équation effectue la moyenne sur toutes les possibilités, en pondérant chaque possibilité par la probabilité d'occurrence. Elle implique que la valeur de l'état de départ s doit être égal à la valeur réduite espérée de l'état futur, plus la récompense espérée sur le chemin.

II-F - Politique optimale et équations d'optimalité de Bellman

II-F-1 - Politique optimale

On cherche à trouver la politique optimale π*, c'est-à-dire la politique meilleure que toutes les autres. Mais il faut tout d'abord définir comment une politique est

meilleure qu'une autre, ce qui consiste à munir l'ensemble des politiques d'une relation d'ordre partielle :
$\pi' \geq \pi$ si $V^{\pi'}(s) \geq V^{\pi}(s)$, $\forall s \in S$ (9)
Cela signifie que π' est meilleure que π.

Il y a donc au moins une politique meilleure que toutes les autres, ou égale. C'est une politique optimale π^*.

Les politiques optimales ont toutes une même fonction de valeur optimale de l'état :
$V^*(s) = \max_{\pi} V^{\pi}(s)$, pour tout $s \in S$ (10)
Les politiques optimales ont aussi la même fonction de valeur optimale de l'action:
$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$, $\forall s \in S, \forall a \in A(s)$ (11)

II-F-2 - Equations d'optimalité de Bellman
Elles permettent de déterminer V* et Q* :

- Pour V*, l'équation d'optimalité de Bellman est :
$$V^*(s) = \max_{a \in A(s)} \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma \times V^*(s) \right] \quad (12)$$
V* est l'unique solution de ce système d'équation non linéaire.

- Pour Q*, l'équation d'optimalité de Bellman est :
$$Q^*(s,a) = \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma \times \max_{a'} Q^*(s',a') \right] \quad (13)$$
Q* est l'unique solution de ce système d'équation non linéaire.

II-F-3 - Résolutions des équations d'optimalité de Bellman

Déterminer une politique optimale en résolvant l'équation d'optimalité de Bellman nécessite :
- de connaître précisément la dynamique de l'environnement
- de prévoir assez d'espace et de temps pour le calcul
- de respecter la propriété de Markov
On doit donc faire des approximations.

La plupart des méthodes d'apprentissage par renforcement peuvent être considérés comme des résolutions approchées des équations d'optimalité de Bellman.

Le problème d'apprentissage par renforcement étant maintenant défini, nous allons voir comment la programmation dynamique peut le résoudre.

Deuxième partie : PROGRAMMATION DYNAMIQUE

I- Présentation et définition

Le principe général de la méthode de programmation dynamique est simple. C'est la méthode diviser pou régner. En effet, on résout le problème en combinant les solutions des sous problèmes. Ainsi, on fait référence à une méthode tabulaire, et non à l'écriture du code informatique.

Les algorithmes de la programmation dynamique partitionnent le problème en sous-problèmes indépendants, qu'ils résolvent récursivement ou bien itérativement, puis ils combinent les solutions pour résoudre le problème initial.
En revanche, elle est applicable lorsque les sous-problèmes ne sont pas indépendants c'est-à-dire ayant des sous-sous-problèmes en commun. Dans ce cas, l'algorithme diviser pour régner fait plus de travail que nécessaire en résolvant plusieurs fois le sous-sous-problème en commun.

Un algorithme de programmation dynamique résout chacun de sous-sous-problème une seule fois, et mémorise sa solution dans un tableau, épargnant ainsi le recalcule à chaque fois que le sous problème est rencontré.

La programmation dynamique est utilisée en général dans les problèmes d'optimisation comme c'est le cas ici pour l'apprentissage par renforcement. Dans ce type de problèmes, il peut y avoir plusieurs solutions possibles. Chaque

solution est affectée d'une valeur qu'on souhaite optimiser en cherchant soit son maximum soit son minimum. Une telle solution est *une* solution optimale et non *la* solution optimale, puisqu'il peut y avoir plusieurs solutions ayant la valeur optimale.

Le développement d'un algorithme de programmation dynamique peut être planifié par une séquence de quatre étapes :

1- Caractériser la structure de la solution optimale.
2- Définir récursivement ou itérativement la valeur d'une solution optimale.
3- Calculer la valeur d'une solution optimale en remontant progressivement jusqu'à l'énoncé du problème initial.
4- Construire une solution optimale pour les informations calculées.

II- Programmation dynamique et apprentissage par renforcement :

Dans la méthode de l'apprentissage par renforcement le terme programmation dynamique fait référence à un nombre d'algorithmes qu'on peut utiliser pour calculer les meilleures politiques étant donné un modèle parfait de l'environnement selon le processus de décision de Markov. Tous les algorithmes qui vont suivre convergent vers une politique optimale sous l'hypothèse d'un processus de décision de Markov fini et discontinu.

L'idée principale de la programmation dynamique et l'apprentissage par renforcement est l'utilisation des fonctions de valeurs d'états pour organiser et structurer la recherche des politiques adéquates. Comme on l'a vu précédemment il est simple d'obtenir les politiques optimales une fois les fonctions de valeurs optimales à savoir V^* et Q^* qui satisfassent aux équations de Bellman.

$$V^*(s) = \max_a E\left\{r_{t+1} + \gamma V^*(s_{t+1}) \middle| s_t = s, a_t = a\right\}$$

$$V^*(s) = \max_a \sum_{s:} P^a_{ss:}\left[R^a_{ss} + \gamma V^*(s)\right]$$

et

$$Q^*(s,a) = E\left\{r_{t+1} + \gamma \max_a Q^*(s_{t+1}, a) \middle| s_t = s, a_t = a\right\}$$

$$Q*(s,a) = \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma \max_{a'} Q^*(s', a')\right]$$

III- Les algorithmes :

III-A-Policy Evaluation :

L'algorithme a pour but de rechercher la fonction de valeur V^π pour une politique choisie arbitrairement π et c'est ce qu'on appelle l'évaluation de la politique. On le référencer comme un problème de prédiction.

On a déjà que pour $s \in S$

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s') \right]$$

avec $\pi(s,a)$ la probabilité de prendre l'action a dans l'état s sous la politique π et les résultats sont inscrits sous π pour indiquer qu'il sont relatifs à cette politique.

Existence et unicité de V^π sont garantis tant que $\gamma < 1$.

Si les dynamiques de l'environnement sont connus alors on un système de S équations linéaires de S inconnues. La solution est en principe directement acquise par le calcul c'est pour cela que les méthodes itératives sont convenables. On choisi V_0 arbitrairement et les valeurs ultérieures sont obtenues en calculant les équations de Bellman avec :

$$V_{k+1}(s) = E_\pi \left\{ r_{t+1} + \gamma V(s_{t+1}) \middle| s_t = s \right\}$$
$$V_{k+1}(s) = \sum_{s'} \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right]$$

La séquence $\{V_k\}$ converge sous la condition $\gamma < 1$ quand k tend vers l'infini.

On appelle l'algorithme « *Iterative policy evaluation* »

Pour produire chaque approximation successive V_{k+1} de V_k, l'algorithme applique la même opération à chaque état s : il remplace l'ancienne valeur de s par une nouvelle obtenue des anciennes valeurs des états successeurs à s, et de la récompense immédiate attendue, tout le long du des « one-step transitions » sous la politique en cours d'évaluation. C'est ce qu'on appelle le *full « backup »,* il est appelé ainsi car il est basé sur les tous états suivants possibles plutôt que sur un seul.

Chaque itération effectue un retour sur la valeur de chaque état une fois pour produire la nouvelle fonction de valeur V_{k+1}.

Pour implémenter l'algorithme de programmation dynamique on doit utiliser deux variables V_k et V_{k+1}. Un pour les anciennes valeurs l'autre pour les nouvelles. Par se procédé les nouvelles valeurs sont générées directement des anciennes sans changer ces dernières. Ce serait plus simple d'utiliser une seule variable en mettant à chaque fois les valeurs à jour, néanmoins et selon l'ordre du « backup » les nouvelles valeurs peuvent être utilisées à la place des anciennes. Ce dernier algorithme légèrement modifié diffère du premier par la vitesse de la convergence.

Les retours d'états sont effectués sous forme de « *sweep* » le long de l'espace des états.

L'ordre dans lequel les états sont évalués influe considérablement la vitesse de la convergence.

Un autre point concernant l'implémentation est celui de la fin de l'algorithme. Formellement « Iterative policy evaluation » converge uniquement vers la limite, mais en pratique on fixe une condition d'arrêt quand la différence entre deux valeurs consécutives gravite autour d'un très petit réel fixé au préalable.

Imput π, the policy to be evaluated
Initialise $V(s) = 0$, for all $s \in S^+$

Repeat
$\Delta \leftarrow 0$
for each $s \in S$
$v \leftarrow V(s)$
$V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s')]$
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta \leq \theta$ (a small positive number)
output $V \approx V^\pi$

ITERATIVE POLICY EVALUATION

III-C- « Policy improvement » et « Iterative policy » :

On dispose maintenant de la fonction de valeur d'état V^π déterminée pour une politique donnée π. Pour un état s on voudrait savoir s'il est possible de changer

la politique pour choisir une action $a \neq \pi(s)$. Maintenant qu'on connaît la rentabilité de la politique courante pour s, serait il plus ou moins bénéfique si on opte pour un e nouvelle politique ?

Un manière de répondre à la question est de considérer une sélection de a en et reprendre la politique π ensuite. Ceci se traduit mathématiquement par la formule suivante :

$$Q^{\pi}(s,a) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s, a_t = a\right\} = \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

Le critère est la comparaison avec $V^{\pi}(s)$. Si c'est plus grand alors il serait mieux de sélectionner a en s et reprendre π ensuite plutôt que de suivre π tout le temps. Choisir a chaque fois que s est rencontré engendre une nouvelle politique qui sera meilleure que π .
C'est un cas particulier du théorème « *Policy improvement theorem* » . Soit π et π' deux politiques telles que pour tout $s \in S$ on a :

$$V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s)) \qquad (*)$$

alors la politique π' est meilleure que π et donc les récompenses attendues seront supérieures ou égales à celles de π, c'est à dire que pour tout $s \in S$ on a

$$V^{\pi'}(s) \geq V^{\pi}(s)$$

L'idée du théorème est la suivante. En partant de la relation (*) on continue à étendre Q^{π} et ré-appliquer (*) jusqu'à aboutir à $V^{\pi'}(s)$.

Un fois qu'on a vu comment évaluer un changement de politique pour un seul état et une action particulière, il est naturel d'étendre le processus à tous les états et toutes les actions possibles. On choisi à chaque état l'action qui s'accorde le mieux avec $Q^{\pi}(s,a)$. En d'autres termes on considère la nouvelle « *greedy policy* » π', donnée par :

$$\pi(s) = \arg\max_{a} Q^{\pi}(s,a) = \arg\max_{a} \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi'}(s')\right]$$

Etant donné une politique π améliorée en une politique π', on peut toujours calculer $V^{\pi'}$ et l'améliorer encore une fois pour aboutir à une politique π''

meilleure que π' et re-itérer la procédure jusqu'à conclure par une politique qui soit optimale.

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} .... \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

Ici $\xrightarrow{E}$ signifie « policy evaluation » et $\xrightarrow{I}$ indique la « policy improvement ». Chaque politique est garantie d'être la meilleure de toutes celles qui la précèdent car un processus de décision de Markov fini a un nombre fini de politiques. Le processus doit converger vers une politique optimale et une fonction de valeurs optimales en un nombre fini d'itérations.

Ce procédé d'optimisation est dit l'itération de la politique « policy iteration ». On note que chaque évaluation de la politique commence par la fonction de valeur de la politique précédente ce qui induit un accroissement dans la vitesse de convergence de l'évaluation de la politique (car la fonction de valeur ne change que de peu d'une politique à celle qui la suit).

Voici L'algorithme de l'évaluation et l'optimisation de la politique.

1. initialization

$V(s) \in R$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$

2. Policy Evaluation

Repeat

$\Delta \leftarrow 0$

For each $s \in S$

$v \leftarrow V(s)$

$$V(s) \leftarrow \sum_{s'} P_{ss'}^{\pi(s)}[R_{ss'}^{\pi(s)} + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

Until $\Delta \leq 0$ (a small positive number)

3. Policy Improvement

$policy - stable \leftarrow true$

For each $s \in S$

$b \leftarrow \pi(s)$

$$\pi(s) \leftarrow \arg\max_{a} \sum_{s'} P_{ss'}^{a}[R_{ss'}^{a} + \gamma V(s')]$$

if $b \neq \pi(s)$ then $policy - stable \leftarrow false$

If policy-stable, then stop; else go to 2

POLICY ITERATION

Une autre manière de développer cet algorithme est d'effectuer la même procédure mais en itérant sur les $Q^{\pi}(s,a)$ (le fait de sélectionner une action a sous un état s) ce qui donne l'algorithme suivant :

1. initialization

   $\pi \leftarrow$ An arbitrary deterministic policy

   $Q \leftarrow$ An arbitrary function: $s \times A(s) \rightarrow R$

   $\theta \leftarrow$ Small positive number

2. Policy Evaluation

   Repeat

   $\Delta \leftarrow 0$

   For each $s \in S$ and $a \in A(s)$:

   $q \leftarrow Q(s,a)$

   $Q(s,a) \leftarrow \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma Q(s', \pi(s'))]$

   $\Delta \leftarrow \max(\Delta, |q - Q(s,a)|)$

   Until $\Delta \leq 0$

3. Policy Improvement

   $policy - stable \leftarrow true$

   For each $s \in S$:

   $b \leftarrow \pi(s)$

   $\pi(s) \leftarrow \arg\max_a Q(s,a)$

   If $b \neq \pi(s)$ then $policy - stable \leftarrow false$

   If $policy - stable$, then stop; else go to 2

POLICY ITERATION

III-C- Value Iteration :

Un inconvénient de l'itération de la politique est que chaque itération comprend une évaluation de la politique. Si l'évaluation de la politique est faite itérativement alors la convergence vers V^π n'a lieu que vers la limite.

Il est inutile d'attendre jusqu'à la fin. En effet l'itération de la politique peut être tronquée par différents moyens sans perdre la garantie de la convergence de l'itération de la politique. Un cas important est quand l'évaluation de la politique est stoppée après un seul retour d'état « *backup* » sur chaque état. C'est l'algorithme de la « Value iteration ».

On l'interpréter comme un simple *backup* combinant une amélioration tronquée et une amélioration de la politique dans chaque pas :

$$V_{k+1}(s) = \max_a E\{r_{t+1} + \gamma V(s_{t+1}) | s_t = s, a_t = a\} = \max_a \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma V_k(s')\right]$$

pour tous les $s \in S$.

Pour un V_0 donné la séquence $\{V_k\}$ converge vers V^* sous les mêmes conditions qui garantissent l'existence de ce dernier.

Cet algorithme combine dans chacun de ses « *sweeps* » un *sweep* d'évaluation de la politique et un « *sweep* » d'amélioration de la politique.

Initialize v arbitrarily, e.g., $V(s) = 0$ for all $s \in S^+$

Repeat

$\Delta \leftarrow 0$

for each $s \in S$

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta \le \theta$ (a small positive number)

Output a deterministic policy, π, such that:

$$\pi(s) = \arg\max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s')]$$

VALUE ITERATION

Références

[1]Richard S. Sutton, Andrew G. Barto. « Reinforcement Learning (Adaptive Computation and Machine Learning). Ed. MIT Press, Cambidge, MA, 1998.

[2] http://www-anw.cs.umass.edu/~rich/book/the-book.html

[3] http://www-rocq.inria.fr/scilab/