Master 2 Recherche
Apprentissage Statistique, Optimisation et Applications

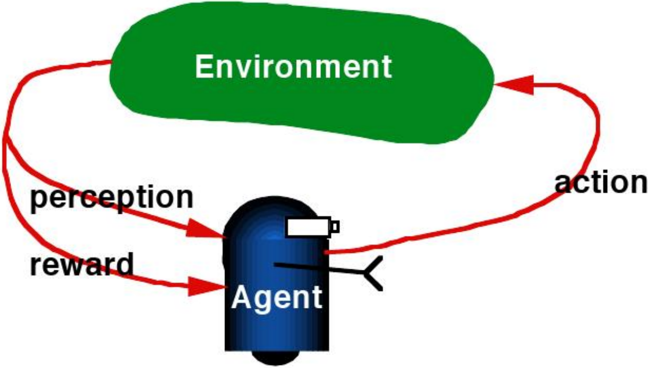
Michèle Sebag – Balazs Kégl – Anne Auger

TAO: Theme Apprentissage & Optimisation

http://tao.lri.fr/tiki-index.php

18 janvier 2012

Apprentissage par Renforcement



Cas général

- Un agent est dans le temps et dans l'espace
- L'environnement est stochastique et incertain
- Le but est d'agir sur l'environnement
- de façon à maximiser une fonction de satisfaction (reward)

Qu'est-ce qu'on apprend ?

Une politique = une stratégie = (état $\rightarrow$ action)

Apprentissage par Renforcement

Plan du cours

1. Contexte
2. Algorithmes
3. Exemple : jouer au Go MoGo
4. de MoGo à la sélection de variables.

Apprentissage par Renforcement: Plan du cours

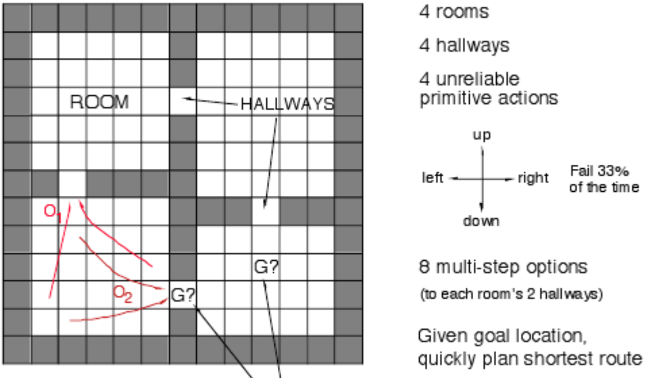
Apprentissage par Renforcement

Contexte

Le monde est inconnu.

Certaines actions, dans certains états, portent des fruits (*rewards*) avec un certain retard [avec une certaine probabilité].

Le but : trouver la politique (état → action) maximisant l'espérance de reward



Apprentissage par Renforcement, exemple

World You are in state 34.
Your immediate reward is 3. You have 3 actions

Robot I'll take action 2

World You are in state 77
Your immediate reward is -7. You have 2 actions

Robot I'll take action 1

World You are in state 34 (again)

Markov Decision Property: actions/rewards only depend on the current state.

Apprentissage par renforcement

Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will – others things being equal – be more firmly connected with the situation, so that when it recurs, they will more likely to recur;

those which are accompanied or closely followed by discomfort to the animal will – others things being equal – have their connection with the situation weakened, so that when it recurs, they will less likely to recur;

the greater the satisfaction or discomfort, the greater the strengthening or weakening of the link.
Thorndike, 1911.

Formalisation

Formalisation

- Espace d'états $\mathcal{S}$
- Espace d'actions $\mathcal{A}$
- Fonction de transition $p(s, a, s') \mapsto [0, 1]$
- Reward $r(s)$

But

- Trouver politique $\pi : \mathcal{S} \mapsto \mathcal{A}$

$$\text{Maximiser } E[\pi] = \text{ Espérance du reward cumulé}$$

(détails après)

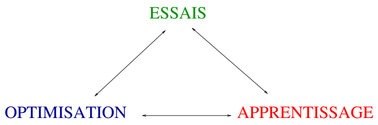
Quelques applications

- Robotique
  Navigation, football, marche, jonglage
- Jeux
  Backgammon, Othello, Tetris, Go, ...
- Contrôle
  Hélicoptère, ascenseurs, telecom, grilles de calcul, gestion de processus industriels, ...
- Recherche opérationnelle
  Transport, scheduling, ...
- Autres
  Computer Human Interfaces, ...

Position du problème

Trois problèmes

- Apprendre le monde (p, r)
- Décider
- Faire des essais



Sources

- Sutton & Barto, Reinforcement Learning, MIT Press, 1998
- http://www.eecs.umich.edu/∼baveja/NIPS05RLTutorial/

Cas particulier

Quand on connait la fonction de transition

Reinforcement learning $\rightarrow$ Optimal control

Défis

Malédiction de la dimensionalité

- état : décrit par *taille*, *apparence*, *couleur*, ...
  $|\mathcal{S}|$ exponentiel en fonction du nombre d'attributs
- Mais tous les attributs ne sont pas toujours pertinents

Exemple:

voir	cygne	blanc	—
	cygne	noir	prendre une photo
	ours	—	fuir

Défis

Malédiction de la dimensionalité

- état : décrit par *taille*, *apparence*, *couleur*, ...
  $|\mathcal{S}|$ exponentiel en fonction du nombre d'attributs
- Mais tous les attributs ne sont pas toujours pertinents

Exemple:

voir	cygne	blanc	—
	cygne	noir	prendre une photo
	ours	—	fuir

Horizon – Rationalité limitée

- Horizon infini : on a l'éternité devant soi. JAMAIS
- Horizon fini inconnu : on veut une politique qui trouve le but aussi vite que possible
- Horizon fini : on veut une politique qui trouve le but après T pas de temps
- Rationalité limitée : on veut trouver rapidement une politique raisonnable (qui trouve une approximation du but)

Reinforcement learning

Formalisation

Notations

- State space $\mathcal{S}$
- Action space $\mathcal{A}$
- Transition model
  - deterministic: $s' = t(s, a)$
  - probabilistic: $P^a_{s,s'} = p(s, a, s') \in [0, 1]$.
- Reward $r(s)$ bounded
- Time horizon H (finite or infinite)

Goal

- Find policy (strategy) $\pi : \mathcal{S} \mapsto \mathcal{A}$
- which maximizes cumulative reward from now to timestep H

Markov Decision Process

But can we define $P^a_{ss'}$ and $r(s)$?

- YES, if all necessary information is in s
- NO, otherwise
  - If state is partially observable

Goal: arrive in the third branch

  - If environment (reward and transition distribution) is changing
    Reward for *first* photo of an object by the satellite

The Markov assumption

$$P(s_{h+1}|s_0\ a_0\ s_1\ a_1 \ldots s_h\ a_h) = P(s_{h+1}|s_h\ a_h)$$

Everything you need to know is the current (state, action).

Approaches

- Value function
  - Value iteration
  - Policy iteration
- Temporal differences
- Q-learning
- Direct policy search
  optimization in the π space

Stochastic optimization

Policy and value function 1/3

Finite horizon, deterministic transition

$$V_\pi(s_0) = r(s_0) + \sum_{h=1}^{H} r(s_h)$$

where $s_{h+1} = t(s_h, a_h = \pi(s_h))$

Policy and value function 1/3

Finite horizon, deterministic transition

$$V_\pi(s_0) = r(s_0) + \sum_{h=1}^{H} r(s_h)$$

where $s_{h+1} = t(s_h, a_h = \pi(s_h))$

Finite horizon, stochastic transition

$$V_\pi(s_0) = r(s_0) + \sum_{h=1}^{H} p(s_{h-1}, a_{h-1} = \pi(s_{h-1}), s_h) r(s_h)$$

where $s_{h+1} = s$ with proba $p(s_h, a_h = \pi(s_h), s)$

Policy and value function, 2/3

Finite horizon, stochastic transition

$$V_\pi(s_0) = r(s_0) + \sum_{h=1}^{H} p(s_{h-1}, a_{h-1} = \pi(s_{h-1}), s_h) r(s_h)$$

where $s_{h+1} = s$ with proba $p(s_h, a_h = \pi(s_h), s)$

Infinite horizon, stochastic transition

$$V_\pi(s_0) = r(s_0) + \sum_{h=1}^{H} \gamma^h p(s_{h-1}, a_{h-1} = \pi(s_{h-1}), s_h) r(s_h)$$

with discount factor γ, $0 < \gamma < 1$

Remark

$\gamma < 1 \rightarrow V < \infty$

γ small $\rightarrow$ myopic agent.

Value function and Q-value function

Value function

$$V : S \mapsto \mathbb{R}$$

$V_\pi(s)$: utility of state s when following policy π

Improving π by using V_π requires to know the transition model:

$$\pi(s) \rightarrow \text{ arg max } P^a_{ss'} V_\pi(s')$$

Q function

$$Q : (S \times A) \mapsto \mathbb{R}$$

$Q_\pi(s, a)$: utility of selecting action a in state s when following policy π

Improving π by using Q_π is straightforward:

$$\pi(s) \rightarrow \text{ arg max } Q_\pi(s, a)$$

Optimal policies

From value function to a better policy

$$\pi(s) = argmax_a\{P^a_{ss'} V_\pi(s')\}$$

From policies to optimal value function

$$V^*(s) = max_\pi V_\pi(s)$$

From value function to optimal policy

$$\pi^*(s) = argmax_a\{P^a_{ss'} V^*(s')\}$$

Linear and dynamic programming

If transition model and reward function are known

Step 1

$$\pi(s) := \arg\max_a \left\{ \sum_{s'} P^a_{s,s'} \left(r(s') + \gamma V(s') \right) \right\}$$

Step 2

$$V(s) := \sum_{s'} P^{a=\pi(s)}_{s,s'} \left(r(s') + \gamma V(s') \right)$$

Properties

Converges eventually toward the optimum if all states, actions are considered.

Value iteration

Bellman equation

Iterate

$$V_{k+1}(s) := \max_a \left\{ \sum_{s'} P^a_{s,s'} \left(r(s') + \gamma V_k(s') \right) \right\}$$

Stop when

$$max_s |V_{k+1}(s) - V_k(s)| < \epsilon$$

Initialisation

- arbitrary
- educated is better — see Inverse Reinforcement Learning

Policy iteration

Principle

- Modify π step 1
- Update V until convergence step 2

Getting faster

- Don't wait until V has converged before modifying π.

Discussion

Policy and value iteration

- Must wait until the end of the episode
- Episodes might be long

Can we update V on the fly ?

- I have estimates of how long it takes to go to RER, to catch the train, to arrive at Cité-U
- Something happens on the way (bump into a friend, chat, delay, miss the train,...)
- I can update my estimates of when I'll be home...

TD(0)

1. Initialize V and π
2. Loop on episode
   2.1 Initialize s
   2.2 Repeat

$$
\begin{aligned}
&\text{Select action } a = \pi(s) \\
&\text{Observe } s' \text{ and reward } r \\
&V(s) \leftarrow V(s) + \alpha(\underbrace{r + \gamma V(s')}_{R} - V(s)) \\
&s \leftarrow s'
\end{aligned}
$$

   2.3 Until s' terminal state

Discussion

Update on the spot ?

- Might be brittle
- Instead one can consider several steps

$$R = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2})$$

Find an intermediate between

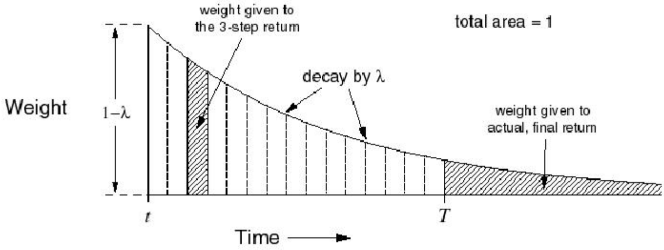
- Policy iteration

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots$$

- TD(0)

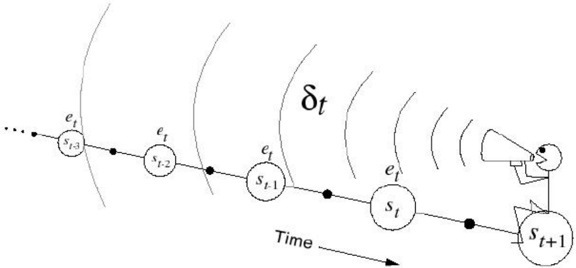
$$R_t = r_{t+1} + \gamma V_t(s_{t+1})$$

TD(λ), intuition



$$R_t^\lambda = (1-\lambda)\underbrace{\sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)}}_{} + \underbrace{\lambda^{T-t-1} R_t}_{}$$

TD(λ), intuition, followed



$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)$$

TD(λ)

1. Initialize V and π
2. Loop on episode
   2.1 Initialize s
   2.2 Repeat

$$
\begin{array}{l}
a = \pi(s) \\
\text{Observe } s' \text{ and reward } r \\
\delta \leftarrow r + V(s') - V(s) \\
e(s) \leftarrow e(s) + 1 \\
\qquad\qquad \text{For all } s“ \\
\qquad\qquad V(s'') \leftarrow V(s“) + \alpha\delta e(s'') \\
\qquad\qquad e(s'') \leftarrow \gamma\lambda e(s'') \\
s \leftarrow s'
\end{array}
$$

   2.3 Until s' terminal state

Q-learning

Principle: Iterate

- During an episode (from initial state until reaching a final state)
- At some point explore and choose another action;
- If it improves, update $Q(s, a)$:

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \times \left[\overbrace{\underbrace{r(s_{t+1})}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})}_{\text{max future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right]$$

Equivalent to

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t)(1 - \alpha) + \alpha[r(s_{t+1}) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})]$$