Markov decision processes: Model and basic algorithms

Matthijs Spaan

Institute for Systems and Robotics

Instituto Superior Técnico

Lisbon, Portugal

Reading group meeting, January 22, 2007

Overview

- Introduction to Markov decision processes (MDPs):
  - Model
  - Model-based algorithms
  - Reinforcement-learning techniques
- Discrete state, discrete time case.
- For further reference: (Puterman, 1994; Sutton and Barto, 1998; Bertsekas, 2000).
- In this talk algorithms are taken from (Sutton and Barto, 1998).

Discrete MDP model

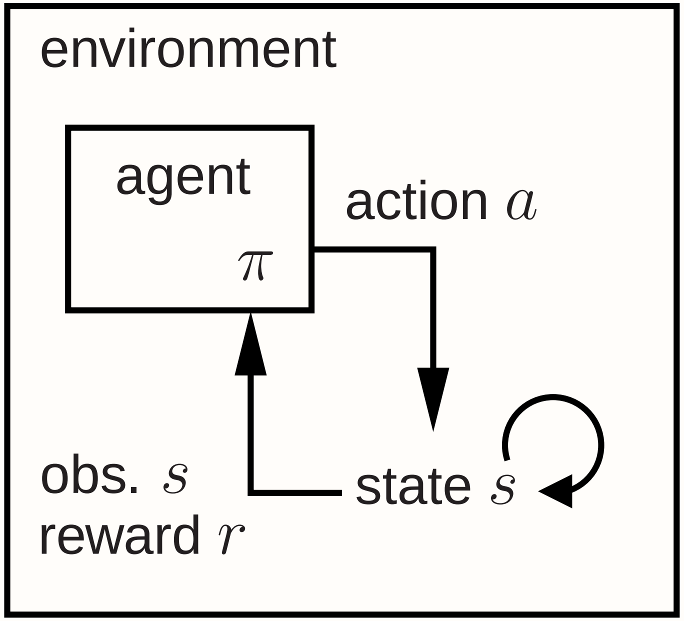
Discrete MDP model:

- Time t is discrete.
- State space S.
- Set of actions A.
- Reward function $R : S \times A \to \mathbb{R}$.
- Transition model $p(s'|s, a)$, $S \times A \to \Delta(S)$.
- Initial state s_0 is drawn from $\Delta(S)$.

The Markov property entails that the next state s_{t+1} only depends on the previous state s_t and action a_t:

$$p(s_{t+1}|s_t, s_{t-1}, \dots, s_0, a_t, a_{t-1}, \dots, a_0) = p(s_{t+1}|s_t, a_t). \quad (1)$$

MDP Agent



Optimality criterion

Agent should maximize

$$E\left[\sum_{t=0}^{h} \gamma^t R_t\right], \qquad (2)$$

where

- h is the planning horizon, can be finite or ∞
- γ is a discount rate, $0 \leq \gamma < 1$

Reward hypothesis (Sutton and Barto, 1998):
All goals and purposes can be formulated as the maximization of the cumulative sum of a received scalar signal (reward).

Policies and value

An agent acts according to its policy

$$\pi : S \rightarrow A. \tag{3}$$

A common way to characterize a policy is by its value function:

$$V^{\pi}(s) = R(s, \pi(s)) + E\Big[\sum_{t=1}^{\infty} \gamma^t R(s_t, \pi(s_t))\Big]. \tag{4}$$

The expectation operator averages over the stochastic transition model, which leads to the following recursion:

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V^{\pi}(s'). \tag{5}$$

Policies and value (1)

Extracting a policy π from a value function V is easy:

$$\pi(s) = \arg\max_{a \in A} \Big[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V(s') \Big]. \qquad (6)$$

Bellman (1957) equation:

$$V^*(s) = \max_{a \in A} \Big[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^*(s') \Big], \qquad (7)$$

Solving this (nonlinear) system of equations for each state s yields the optimal value function, and an optimal policy π^*. However, due to the nonlinear $\max$ operator solving the system for each state simultaneously is not efficient for large MDPs (Puterman, 1994, Sec. 6.9).

Value iteration

Value iteration: successive approximation technique.
The optimal value function V_0^*

$$V_0^*(s) = \max_{a \in A} R(s, a). \tag{8}$$

In order to consider one step deeper into the future, i.e., to compute V_{n+1}^* from V_n^* we can turn (7) into an update:

$$V_{n+1}^*(s) = \max_{a \in A} \Big[R(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V_n^*(s') \Big], \tag{9}$$

which is known as a Bellman backup H, allowing us to write (9) as

$$V_{n+1}^* = H V_n^*. \tag{10}$$

Value iteration (1)

Initialize V arbitrarily, e.g., $V(s) = 0, \forall s \in S$
repeat
  $\delta \leftarrow 0$
  for all $s \in S$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V(s') \right]$
    $\delta \leftarrow \max(\delta, |v - V(s)|)$
  end for
until $\delta < \epsilon$
Return V

Value iteration (2)

Value iteration discussion:

- As $n \to \infty$, value iteration converges.
- Value iteration has converged when the largest update δ in an iteration is below a certain threshold ϵ.
- Exhaustive sweeps are not required for convergence: arbitrary states can be backed up in arbitrary order, provided that in the limit all states are visited infinitely often (Bertsekas and Tsitsiklis, 1989).
- This can be exploited by backing up the most promising states first, known as prioritized sweeping (Moore and Atkeson, 1993; Peng and Williams, 1993).

Policy iteration (Sutton and Barto, 1998)

{1. Initialize} $V(s) \in \mathbb{R}$ and $\pi(s) \in A$
{2. Policy evaluation}
repeat
    $\delta \leftarrow 0$
    for all $s \in S$ **do**
        $v \leftarrow V(s)$
        $V(s) \leftarrow R(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V(s')$
        $\delta \leftarrow \max(\delta, |v - V(s)|)$
    end for
until $\delta < \epsilon$
{3. Policy improvement}
policy-stable $\leftarrow$ true
for all $s \in S$ **do**
    $b \leftarrow \pi(s)$
    $\pi(s) \leftarrow \arg\max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V(s') \right]$
    if $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ false
end for
if *policy-stable*, then stop; else go to 2

Policy iteration (1)

Policy iteration discussion:

- Converges in finite number of steps (only finite number of stationary policies), when using exact policy evaluation.
- Policy evaluation can be done iteratively (as shown before) or by solving the system of linear equations.
- When state space is large, this can be expensive.

Q-learning

- Reinforcement-learning techniques learn from experience, no knowledge of the model is required.
- Policy is often represented as state-action value function:

$$Q : S \times A \rightarrow \mathbb{R} \tag{11}$$

and the policy as

$$\pi(s) = \arg\max_{a \in A} Q(s, a) \tag{12}$$

- Q-learning update (Watkins, 1989):

$$Q(s, a) = (1 - \beta)\, Q(s, a) + \beta \Big[R(s, a) + \gamma \max_{a' \in A} Q(s', a') \Big], \tag{13}$$

where $0 < \beta \leq 1$ is a learning rate.

Q-learning (1)

Initialize $Q(s, a)$ arbitrarily
repeat
  Initialize s
  repeat
    Choose a from s using policy derived from Q (e.g., ϵ-greedy)
    Take action a, observe r, s'
    $Q(s, a) = (1 - \beta)\, Q(s, a) + \beta \left[r + \gamma \max_{a' \in A} Q(s', a') \right]$
    $s \leftarrow s'$
  until s is terminal
until

Q-learning (2)

Q-learning discussion:

- Q-learning is guaranteed to converge to the optimal Q-values if all $Q(s, a)$ values are updated infinitely often Watkins and Dayan (1992).
- In order to make sure all actions will eventually be tried in all states exploration is necessary.
- A common exploration method is to execute a random action with small probability ϵ, which is known as ϵ-greedy exploration Sutton and Barto (1998).

Potential future topics

- Dynamic programming or reinforcement learning in continuous state spaces.
- Advanced reinforcement-learning topics such as generalization.
- Adding partial observability.
- …

References

R. Bellman. *Dynamic programming*. Princeton University Press, 1957.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 2nd edition, 2000.

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.

J. Peng and R. J. Williams. Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 1(4):437–454, 1993.

M. L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, Cambridge University, 1989.

C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3-4): 279–292, 1992.