Learning to control Markov Decision Processes

CS7032: AI & Agents for IET

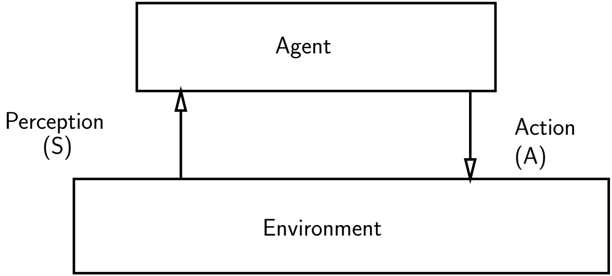
November 24, 2015

Outline

- Reinforcement Learning problem as a Markov Decision Process (MDP)
- Rewards and returns
- Examples
- The Bellman Equations
- Optimal state- and action-value functions and Optimal Policies
- Computational considerations

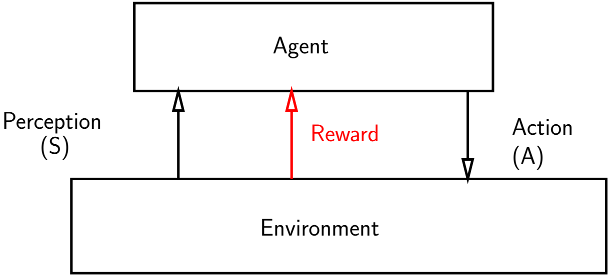
The Abstract architecture revisited (yet again)

- Add the ability to evaluate feedback:



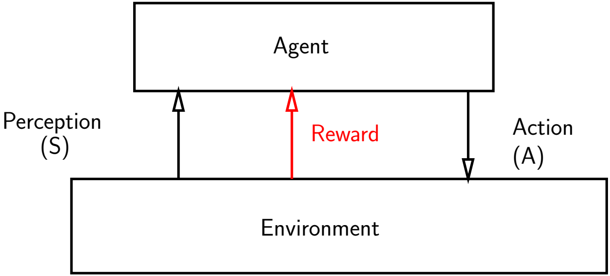
The Abstract architecture revisited (yet again)

- Add the ability to evaluate feedback:



The Abstract architecture revisited (yet again)

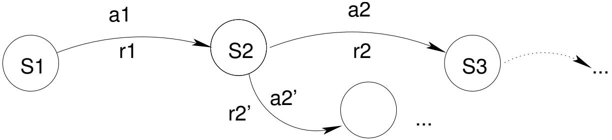
- Add the ability to evaluate feedback:



- How to represent goals?

Interaction as a Markov decision process

- We start by simplifying *action* (as in purely reactive agents):
  - $action : S \rightarrow A$ (*New notation: $action \stackrel{def}{=} \pi$)
  - $env : S \times A \rightarrow S$ (New notation: $env \stackrel{def}{=} \delta$)
- at each discrete time agent observes state $s_t \in S$ and chooses action $a_t \in A$
- then receives immediate reward r_t
- and state changes to s_{t+1} (deterministic case)



Levels of abstraction

- Time steps need not be fixed real-time intervals.
- Actions can be low level (e.g., voltages to motors), or high level (e.g., accept a job offer), mental (e.g., shift in focus of attention), etc.
- States can be low-level sensations, or they can be abstract, symbolic, based on memory, or subjective (e.g., the state of being surprised or lost).
- An RL agent is not like a whole animal or robot.
  - The environment encompasses everything the agent cannot change arbitrarily.
- The environment is not necessarily unknown to the agent, only incompletely controllable.

Specifying goals through rewards

- The reward hypothesis [Sutton and Barto, 1998, see]:

  > All of what we mean by goals and purposes can be well thought of as the maximization of the cumulative sum of a received scalar signal (reward).

- Is this correct?

Specifying goals through rewards

- The reward hypothesis [Sutton and Barto, 1998, see]:

  > All of what we mean by goals and purposes can be well thought of as the maximization of the cumulative sum of a received scalar signal (reward).

- Is this correct?
- Probably not: but simple, surprisingly flexible and easily disprovable, so it makes scientific sense to explore it before trying anything more complex.

Some examples

- Learning to play a game (e.g. draughts):

Some examples

- Learning to play a game (e.g. draughts):
  - $+1$ for winning, -1 for losing, 0 for drawing
  - (similar to the approach presented previously.)

Some examples

- Learning to play a game (e.g. draughts):
  - $+1$ for winning, -1 for losing, 0 for drawing
  - (similar to the approach presented previously.)
- Learning how to escape from a maze:

Some examples

- Learning to play a game (e.g. draughts):
  - $+1$ for winning, -1 for losing, 0 for drawing
  - (similar to the approach presented previously.)
- Learning how to escape from a maze:
  - set the reward to zero until it escapes
  - and $+1$ when it does.
- Recycling robot:

Some examples

- Learning to play a game (e.g. draughts):
  - $+1$ for winning, -1 for losing, 0 for drawing
  - (similar to the approach presented previously.)
- Learning how to escape from a maze:
  - set the reward to zero until it escapes
  - and $+1$ when it does.
- Recycling robot: $+1$ for each recyclable container collected, -1 if container isn't recyclable, 0 for wandering, -1 for bumping into obstacles etc.

Important points about specifying a reward scheme

- the reward signal is the place to specify what the agent's goals are (given that the agent's high-level goal is always to maximise its rewards)
- the reward signal is not the place to specify how to achieve such goals
- Where are rewards computed in our agent/environment diagram?
- Rewards and goals are outside the agent's direct control, so they it makes sense to assume they are computed by the environment!

From rewards to returns

- We define (expected) returns (R_t) to formalise the notion of rewards received in the long run.
- The simplest case:

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T \quad (1)$$

where $r_{t+1}, \ldots$ is the sequence of rewards received after time t, and T is the final time step.

- What sort of agent/environment is this definition most appropriate for?

From rewards to returns

- We define (expected) returns (R_t) to formalise the notion of rewards received in the long run.
- The simplest case:

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T \tag{1}$$

where $r_{t+1}, \ldots$ is the sequence of rewards received after time t, and T is the final time step.
- What sort of agent/environment is this definition most appropriate for?
- Answer: episodic interactions (which break naturally into subsequences; e.g. a game of chess, trips through a maze, etc).

Non-episodic tasks

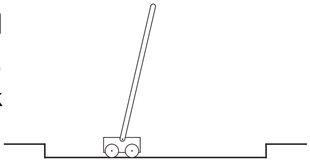
- Returns should be defined differently for continuing (aka non-episodic) tasks (i.e. $T = \infty$).
- In such cases, the idea of discounting comes in handy:

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

  where $0 \leq \gamma \leq 1$ is the discount rate
- Is this sum well defined?
- One can thus specify far-sighted or myopic agents by varying the discount rate γ.

The pole-balancing example

- Task: keep the pole balanced (beyond a critical angle) as long as possible, without hitting the ends of the track [Michie and Chambers, 1968]



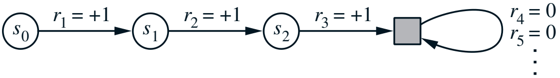
- Modelled as an episodic task:
  - reward of $+1$ for each step before failure $\Rightarrow$ $R_t =$ number of steps before failure
- Can alternatively be modelled as a continuing task:
  - "reward" of -1 for failure and 0 for other steps $\Rightarrow$ $R_t = -\gamma^k$ for k steps before failure

Episodic and continuing tasks as MDPs

- Extra formal requirements for describing episodic and continuing tasks:
  - need to distinguish episodes as well as time steps when referring to states: $\Rightarrow s_{t,i}$ for time step t of episode i (we often omit the episode index, though)
  - need to be able to represent interaction dynamics so that R_t can be defined as sums over finite or infinite numbers of terms [equations (1) and (2)]

Episodic and continuing tasks as MDPs

- Extra formal requirements for describing episodic and continuing tasks:
  - need to distinguish episodes as well as time steps when referring to states: $\Rightarrow s_{t,i}$ for time step t of episode i (we often omit the episode index, though)
  - need to be able to represent interaction dynamics so that R_t can be defined as sums over finite or infinite numbers of terms [equations (1) and (2)]
- Solution: represent termination as an absorbing state:



- and making $R_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$ (where we could have $T = \infty$ or $\gamma = 1$, but not both)

MDPs, other ingredients

- We assume that a reinforcement learning task has the Markov Property:

$$P(s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, \dots r_1, s_0, a_0) = P(s_{t+1} = s', r_{t+1} = r | s_t, a_t) \quad (3)$$

  for all states, rewards and histories.
- So, to specify a RL task as an MDP we need:
  - to specify S and A
  - and $\forall s, s' \in S, a \in A$:
    - transition probabilities:

$$\mathcal{P}^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$$

    - and rewards $\mathcal{R}^a_{ss'}$, Where a reward could be specified as an average over transitions from s to s' when the agent performs action a

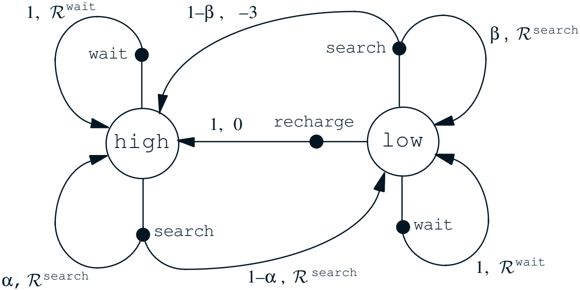
$$\mathcal{R}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$$

The recycling robot revisited

- At each step, robot has to decide whether it should (1) actively search for a can, (2) wait for someone to bring it a can, or (3) go to home base and recharge.
- Searching is better but runs down the battery; if it runs out of power while searching, has to be rescued (which is bad).
- Decisions made on basis of current energy level: high,low.
- Rewards = number of cans collected (or -3 if robot needs to be rescued for a battery recharge and 0 while recharging)

As a state-transition graph

- $S = \{high, low\}$, $A = \{search, wait, recharge\}$
- $\mathcal{R}^{search}$ = expected no. of cans collected while searching
- $\mathcal{R}^{wait}$ = expected no. of cans collected while waiting ($\mathcal{R}^{search} > \mathcal{R}^{wait}$)



Value functions

- RL is (almost always) based on estimating value functions for states, i.e. how much return an agent can expect to obtain from a given state.
- We can the define the state-value function under policy π as the the expected return when starting in s and following π thereafter:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} \tag{4}$$

- Note that this implies averaging over probabilities of reaching future states, that is, $P(s_{t+1} = s' | s_t = s, a_t = a)$ over all t.
- We can also generalise the action function (policy) to $\pi(s, a)$, returning the probability of taking action a while in state s, which implies also averaging over actions.

The action-value function

- we can also define an action-value function to give the value of taking action a in state s under a policy π:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} \tag{5}$$

where $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$.

- Both v^{π} and Q^{π} can be estimated, for instance, through simulation (Monte Carlo methods):
  - for each state s visited by following π, keep an average $\hat{V}^{\pi}$ of returns received from that point on.
  - $\hat{V}^{\pi}$ approaches V^{π} as the number of times s is visited approaches ∞
  - Q^{π} can be estimated similarly.

The Bellman equation

- Value functions satisfy particular recursive relationships.
- For any policy π and any state s, the following consistency condition holds:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} \tag{6}$$

The Bellman equation

- Value functions satisfy particular recursive relationships.
- For any policy π and any state s, the following consistency condition holds:

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}\{R_t | s_t = s\} \\
&= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}
\end{aligned}
\tag{6}
$$

The Bellman equation

- Value functions satisfy particular recursive relationships.
- For any policy π and any state s, the following consistency condition holds:

$$\begin{aligned}
V^{\pi}(s) &= E_{\pi}\{R_t | s_t = s\} \\
&= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \\
&= E_{\pi}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\}
\end{aligned} \tag{6}$$

The Bellman equation

- Value functions satisfy particular recursive relationships.
- For any policy π and any state s, the following consistency condition holds:

$$\begin{aligned}
V^{\pi}(s) &= E_{\pi}\{R_t | s_t = s\} \\
&= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \\
&= E_{\pi}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\} \\
&= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}_{ss'}^{a} [\mathcal{R}_{ss'}^{a} + \gamma E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\}]
\end{aligned} \tag{6}$$

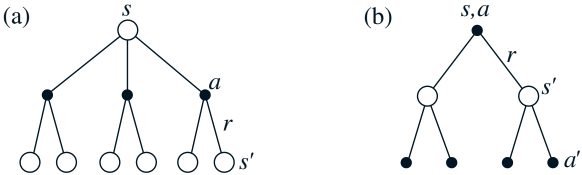
The Bellman equation

- Value functions satisfy particular recursive relationships.
- For any policy π and any state s, the following consistency condition holds:

$$\begin{aligned}
V^{\pi}(s) &= E_{\pi}\{R_t | s_t = s\} \\
&= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \\
&= E_{\pi}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\} \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\}] \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')] \qquad (6)
\end{aligned}$$

Backup diagrams

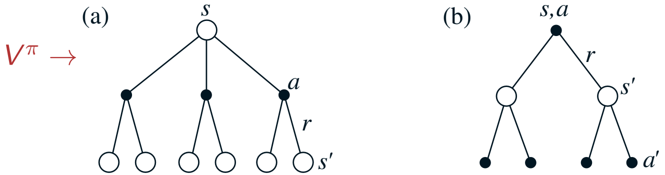
- The Bellman equation for V^{π} (6) expresses a relationship between the value of a state and the value of its successors.
- This can be depicted through backup diagrams



- representing transfers of value information back to a state (or a state-action pair) from its successor states (or state-action pairs).

Backup diagrams

- The Bellman equation for V^π (6) expresses a relationship between the value of a state and the value of its successors.
- This can be depicted through backup diagrams

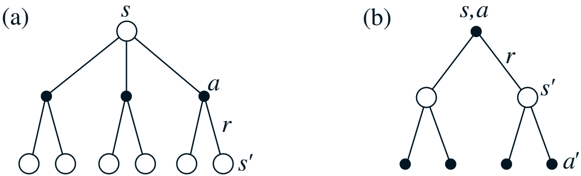


- representing transfers of value information back to a state (or a state-action pair) from its successor states (or state-action pairs).

Backup diagrams

- The Bellman equation for V^π (6) expresses a relationship between the value of a state and the value of its successors.
- This can be depicted through backup diagrams

$V^\pi \rightarrow$

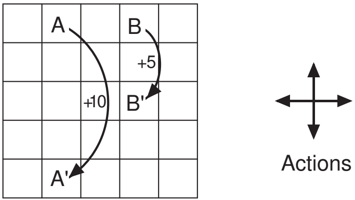


$\leftarrow \; Q^\pi$

- representing transfers of value information back to a state (or a state-action pair) from its successor states (or state-action pairs).

An illustration: The GridWorld

- Deterministic actions (i.e. $\mathcal{P}^a_{ss'} = 1$ for all s, s', a such that s' is reachable from s through a; or 0 otherwise);
- Rewards: $\mathcal{R}^a = -1$ if a would move agent off the grid, otherwise $\mathcal{R}^a = 0$, except for actions from states A and B.



(a)

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

- Diagram (b) shows the solution of the set of equations (6), for equiprobable (i.e. $\pi(s,\uparrow) = \pi(s,\downarrow) = \pi(s,\leftarrow) = \pi(s,\rightarrow) = .25$, for all s) random policy and $\gamma = 0.9$

Optimal Value functions

- For finite MDPs, policies can be partially ordered:
  $\pi \geq \pi'$ iff $V^{\pi}(s) \geq V^{\pi'}(s), \quad \forall s \in S$
- There are always one or more policies that are better than or equal to all the others. These are the optimal policies, denoted π^*.
- The Optimal policies share the same
  - optimal state-value function: $V^*(s) = max_{\pi} V^{\pi}(s), \quad \forall s \in S$
    and
  - optimal action-value function:
    $Q^*(s,a) = max_{\pi} Q^{\pi}(s,a), \quad \forall s \in S$ and $a \in A$

Bellman optimality equation for V*

- The value of a state under an optimal policy must equal the expected return for the best action from that state:

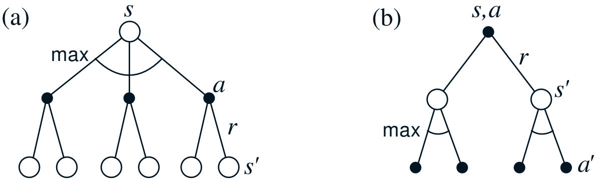
$$
\begin{aligned}
V^*(s) &= \max_{a \in \mathcal{A}(s)} Q^*(s,a) \\
&= \max_a E_{\pi^*}\{R_t | s_t = s, a_t = a\} \\
&= \max_a E_{\pi^*}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a\} \\
&= \max_a E_{\pi^*}\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\} && (7) \\
&= \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^*(s')] && (8)
\end{aligned}
$$

Bellman optimality equation for Q*

- Analogously to V^*, we have:

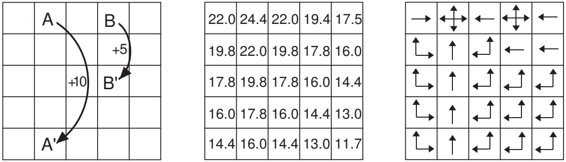
$$
\begin{aligned}
Q^*(s,a) &= E\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\} && (9)\\
&= \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')] && (10)
\end{aligned}
$$

- V^* and Q^* are the unique solutions of these systems of equations.



From optimal value functions to policies

- Any policy that is greedy with respect to V^* is an optimal policy.
- Therefore, a one-step-ahead search yields the long-term optimal actions.
- Given Q^*, all the agent needs to do is set $\pi^*(s) = \arg\max_a Q^*(s, a)$.



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

a) gridworld b) V^* c) π^*

Knowledge and Computational requirements

- Finding an optimal policy by solving the Bellman Optimality Equation requires:
  - accurate knowledge of environment dynamics,
  - the Markov Property.
- Tractability:
  - polynomial in number of states (via dynamic programming)...
  - ...but number of states is often very large (e.g., backgammon has about 10^{20} states).
  - So approximation algorithms have a role to play
- Many RL methods can be understood as approximately solving the Bellman Optimality Equation.

References

These notes are based on [Sutton and Barto, 1998]. For a comprehensive formal treatment of MDPs and RL (under the name of "Neuro-dynamic programming" see [Bertsekas and Tsitsiklis, 1996].

Bertsekas, D. P. and Tsitsiklis, J. N. (1996).
Neuro-Dynamic Programming.
Athena Scientific, Belmont.

Michie, D. and Chambers, R. A. (1968).
BOXES: An experiment in adaptive control.
In Dale, E. and Michie, D., editors, *Machine Intelligence 2*, pages 137–152. Oliver and Boyd, Edinburgh.

Sutton, R. S. and Barto, A. G. (1998).
Reinforcement Learning: An Introduction.
MIT Press, Cambridge, MA.