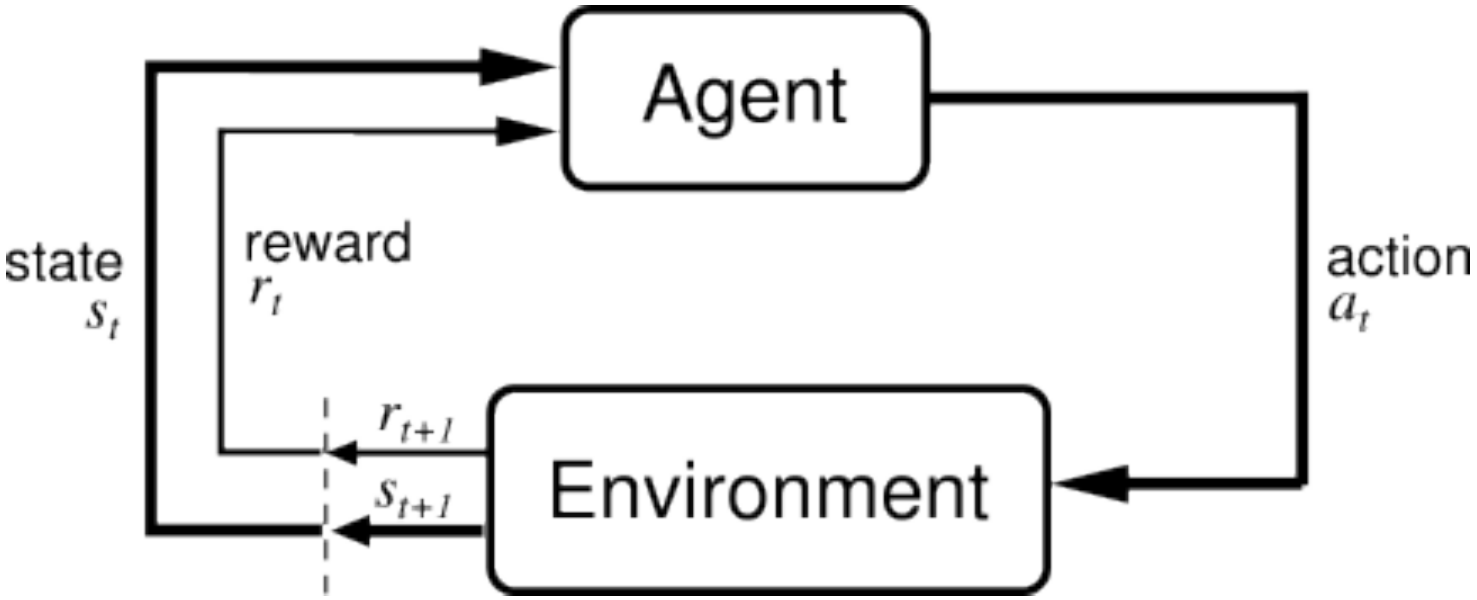
Planning and Control:

Markov Decision Processes

Markov Decision Processes



Examples: cleaning robot, walking robot, pole balancing, shortest path problems

Sutton, Barto: Reinforcement Learning, 1998

Markov decision processes

- Framework for representation complex multi-stage decision problems in the presence of uncertainty
- Efficient solutions
- Outcomes of actions are uncertain – probabilistic model
- Markov assumptions : next state depends in the previous state, and action not the past

Markov Decision Process

- Formal definition
- 4-tuple (X, U, T, R)
- Set of states X - finite
- Set of actions U - finite
- Transition model

$$p(x' \mid x,u) \qquad T : X \times U \times X \rightarrow [0,1]$$

Transition probability for each action, state

- Reward model

$$X \times U \times X \rightarrow R \quad \text{or} \quad X \rightarrow R$$

- Utility of a policy – expected sum of discounted rewards

$$U^{\pi}(x) = E[\sum_{t=0}^{\infty} \gamma^{t} R_{t}(x_{t}) \mid \pi]$$

- Policy π mapping from states to actions

Markov decision processes

- Infinite horizon – since there is infinite time budget the optimal action depends only at the current state
- Discounted rewards are preferable
- Goal how to chose among policies
- Note that given policy, MDP generates not one state sequences but whole set of them each with some probability determined by the transition model
- Hence value of a policy is expected sum of (discounted rewards)

$$E[\sum_{t=0}^{\infty} \gamma^t R_t(x_t) \mid \pi]$$

Types of rewards

- Reward structure: additive rewards

$$U(x_0, x_1, ..., x_n): \quad R(x_0) + R(x_1) + ... + R(x_n)$$

- Discounted rewards

$$U(x_0, x_1, ..., x_n): \quad R(x_0) + \gamma R(x_1) + ... + \gamma^n R(x_n)$$

- Preference for current rewards over future rewards (good model for human and animal preferences over time)
- How to deal with the infinite rewards ? Make sure that the utility of the infinite sequence is finite
- Design proper policies which are guaranteed to reach the final state
- Compare policies based on average reward per step

Utility of the state

- The goodness of the state is defined in terms of utility
- Utility of the state is expected utility of sequences which may follow that state

$$U^{\pi}(x) = E[\sum_{t=0}^{\infty} \gamma^{t} R\ (x_{t}) \,|\, \pi, x_{0} = x]$$

- Distinction between reward and utility
- Goal: Find the best policy (which will maximize expected utility)

$$\pi^{*}: \quad X \rightarrow U$$

- Previous example, deterministic case (no transition probabilities)

Example (Russel&Norvig AI book)

- Robot navigating on the grid
- 4 actions – up, down, left, right
- Effects of moves are stochastic, we may end up in other state then indented with non-zero probability
- Reward +1 for reaching the goal, -1 close to ditch, -0.04 for other states
- Goal: find the policy sequence of actions
- First compute the utility of each state using value iteration

$$\pi: \quad x_t \rightarrow \quad u_t$$

0.81	0.86	0.91	+1
0.76		0.66	-1
0.70	0.66	0.61	0.38

Utility of the states

Transition model:

T(x, u, x')

Up = 0.8 up 0.1 left 0.1 right

Left = …

Right = …

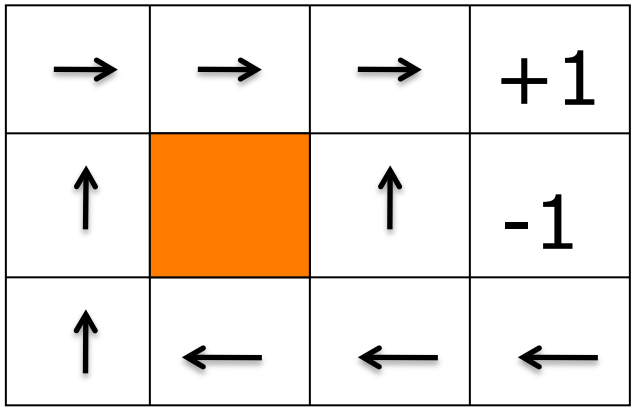
Down = …

Example

- Robot navigating on the grid - up, down, left, right
- Reward +1 for reaching the goal, -1 for going to (4,2)
- R(s) = -0.04 small negative reward for visiting non-goal states (penalize wandering around0
- Goal: find the policy sequence of actions
- Solution

$$\pi: \quad x_t \rightarrow u_t$$

0.81	0.86	0.91	+1
0.76		0.66	-1
0.70	0.66	0.61	0.38



- Idea: calculate utilities of a state, select optimal action in each state – one that maximizes utility

Utility of the state

- How good the state is – defined in terms of sequence
-
- Utility of the state is expected utility of sequences which may follow that state

$$U^{\pi}(x) = E[\sum_{t=0}^{\infty} \gamma^{t} R\ (x_t) \mid \pi, x_0 = x]$$

- Distinction between reward and utility
- Goal: Find the best policy

$$\pi^{*}: \quad X \rightarrow \quad U$$

Value Iteration

- How to compute optimal policies.
- Main idea calculate utility of the each state and then use the state utilities to select optimal action for each state
- The utility of the state is related to the utility of its neighbors (called Bellman equation)

$$U^{\pi}(x) = R(x) + \gamma \max_u \sum_{s'} T(x,u,x')U(x')$$

- We want to solve for utility of each state – going to do it

Iteratively

- Start with arbitrary initial values of utilities and for each state calculate RHS of the equation

$$U_n^{\pi}(x) = R(x) + \gamma \max_u \sum_{s'} T(x,u,x')U_{n-1}(x')$$

- Repeat until you reach equlibrium

Value Iteration

- Bellman equation

$$U_n^{\pi}(x) = R(x) + \gamma \max_u \sum_{s'} T(x,u,x')U_{n-1}(x')$$

- Recursive computation
- Iterate while

$$(U_n^{\pi}(x) - U_{n-1}(x)) > \varepsilon$$

- If the consecutive iterations differ little, fix point is reached
- Value iteration converges

Value iteration

- Compute the optimal value function first, then the policy
- N states – N Bellman equations, start with initial values, iteratively update until you reach equilibrium
- 1. Initialize V; For each state x

$$U_n(x) = R(x) + \gamma \max_a \sum_{x'} T(x,a,x') U_{n-1}(x')$$

- If $\left| U_n(x) - U_{n-1}(x) \right| > \delta$ then $\delta \leftarrow \left| U_n(x) - U_{n-1}(x) \right|$
- until

$$\delta < \varepsilon(1-\gamma)/\gamma$$

- Return U
- Optimal policy can be obtained before convergence of value iteration

Optimal Payoff

- Bellman equation: set of linear constraints, given a policy
- We can compute the utility of each state (value function)under policy

$$U^{\pi}(x) = R(x) + \gamma \sum_{s'} T(x,a,x')U(x')$$

- One equation per state, n states n equations, solve for U
- Find such policy which maximizes the payoff

$$U^{*}(x) = \max_{\pi} U^{\pi}(x)$$

- We know how to compute values function (solve linear eq.)
- How to compute optimal policy – there are exponentially many sequences of actions

Example

- 4 actions – up, down, left, right
- Reward +1 for reaching the goal, -1 close to ditch, -0.04 for other states

$$U(1,1) = -0.04 + \gamma \max($$
$$0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), \quad \text{best action is up}$$
$$0.9U(1,1) + 0.1U(1,2),$$
$$0.9U(1,1) + 0.1U(2,1),$$
$$0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)]$$

3	0.81	0.86	0.91	+1
2	0.76		0.66	-1
1	0.70	0.66	0.61	0.38
	1	2	3	4

Utility of the states

$$\pi: \quad x_t \rightarrow \quad u_t$$

Transition model:

T(x, u, x')

Up = 0.8 up 0.1 left 0.1 right

Left = …

Right = …

Down = …

Policy Iteration

- Alternative Algorithm for finding optimal policies
- Takes policy and computes its value
- Iteratively improved policy, until it cannot be further improved

1. Policy evaluation – calculate the utility of each state under particular policy π_i

$$U^{\pi}(x) = R(x) + \gamma \sum_{s'} T(x, a, x')U(x')$$

Policy Iteration

- Policy improvement – Calculate new MEU policy, using one-step look-ahead based on
  1. Initialize policy π_i
  2. Evaluate policy get U; For each state do if

$$\max_a \sum_{x'} T(x,a,x')U(x') > \sum_{x'} T(x,\pi(x),x')U(x')$$

- Until unchanged π_{i+1}

$$\pi(x) \leftarrow \arg\max_a \sum_{x'} T(x,u,x')U(x')$$

- Above algorithms require updating policy or utility for all states at once – we can do it for a subset of state – asynchronous policy iteration