Markovian Decision Processes

Ionut Moraru

October 22, 2019

Definitions:

- Planning under uncertainty - Computing *sequences of actions* to achieve occasional rewards in a *known*, stochastic environment;
- Reinforcement Learning - *learning to act* from periodic rewards in an *unknown*, stochastic environment.

Observations, state and actions:

- **Observation set O** - formed by perceptions e.g. Distance from car to edge of the road;
- **State set S** - at any point in time, our system is in some state e.g. Actual distance to edge of the road from the car.
- **Actions set A** - not all the actions need to be under the control of the agent. They can happen because of other agents (alternating or concurrent turns) or because of exogenous events due to nature e.g. random failure of equipment.

Observation Function (relationship between states and observations) can be:

- Not observable: $O = \emptyset$ (Heaven vs Hell);
- Fully observable: $S \iff O$ (Go, Chess);
- Partially observable - all remaining cases.

Transition Function - how actions take us between states.

- T(s,a,s') encodes P(s'|s,a);
- There are two properties:
  - **Stationary** - T does not change over time;
  - **Markovian** - T only depends on the previous state/action.
- if T is not either by default, we can augment it to respect one of them

Goals and rewards - In our problems we usually have *Goal-oriented rewards*. Assign any reward value so that $R(success) > R(fail)$ (it can have negative costs $C(a)$ for action a). In the case that we have multiple or no goals, we specify preference by having $R(s, a)$ assigning utilities to each state s and action a (this will maximize expected rewards/utility). To have a trade off for immediate vs future rewards, we use a discount factor γ.

Sequential-decision making objective:

- **Horizon**
  - *Finite* - only care about h-steps into future;
  - *Infinite* - will act same today as tomorrow.

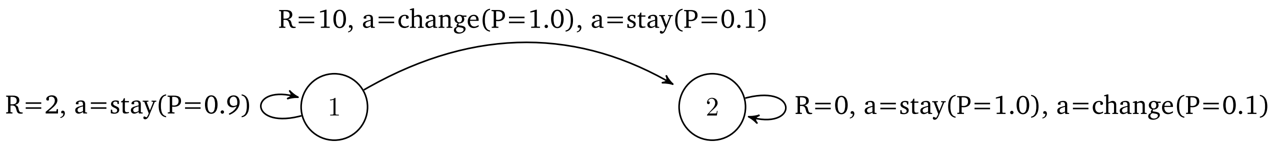
- How to trade off reward over time? Either with an *expected average cumulative return* or *by having expected discounted cumulative* by using a discount factor γ.

Knowledge of Environment:

- **Model-known** - we know the observations, transition and reward functions. It is called planning (under uncertainty). Planning in general is assumed to be goal-oriented;
- **Model-free** - ≥ 1 unknown: observations, transition or reward function. It's called Reinforcement learning and we have to interact with environment to obtain samples;
- **Model-based: approximate model in model free case** - permits hybrid planning and learning.

Fomal Model:

- Markovian Decision Process (MDP): $\langle S, A, T, R \rangle$ - fully observable. What we need to do hear is to define a policy $\pi : S \to A$;


R=10, a=change(P=1.0), a=stay(P=0.1)
R=2, a=stay(P=0.9)
1
2
R=0, a=stay(P=1.0), a=change(P=0.1)

  - S = {1,2};
  - A = {stay, change};
  - R(s=1,a=stay) = 2;
  - T(s=1,a=stay,s'=1) = P(s'=1 | s=1, a=stay) = .9

- Partially Observable Markovian Decision Process (POMDP): $\langle S, A, O, Z, T, R \rangle$

MDP policy, value and Solution: first we define a *value of a policy* π:

$$V_\pi(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot r_t | s = s_0]$$

This tells us how much value you expect to get by following π starting from state s. It also allows us to find optimal solution π^* that maximizes value: $V_{\pi^*}(s) \geq V_\pi(s)$

Value Iteration - from finite to ∞ decisions. Given optimal (t-1)-stage-to-go value functions, how to act optimally with t decisions? Take action a then act so as to achieve V^{t-1} thereafter:

$$Q^t(s,a) := R(s,a) + \gamma \cdot \sum_{s' \in S} T(s,a,s') \cdot V^{t-1}(s')$$

What is the expected value of the best action a at decision stage t?

$$V^t(s) := \max_{a \in A} \{Q^t(s,a)\}$$

At ∞ horizon, converges to V^*

$$\lim_{t \to \infty} \max_s |V^t(s) - V^{t-1}(s)| = 0$$

Bellman Fixed Point - define *Bellman backup* operator b:

$$(BV)(s) = \max_a \{R(s,a) + \gamma \sum_{s'} T(s,a,s') T(s')\}$$

there exists an optima value function V^* and an optima deterministic greedy policy $\pi^* = \pi_{V^*}$ satisfying:

$$\forall s, V^*(s) = (BV^*)(s)$$

Define *Bellman error BE*:

$$(BEV) = \max_s |(BV)(s) - V(s)|$$

From here we can clearly see that $(BEV^*) = 0$ and we can prove that B is a contraction operator for BE:

$$(BE(BV)) \leq \gamma(BEV)$$

We now perform Value Iteration in the search of a fixed-point. We start with an arbitrary value function V^0. We iteratively apply Bellman backups $V^t(s) = (BV^{t-1})(s)$. The Bellman error decreases on each iteration and it terminates when

$$\max_s |V^t(s) - V^{t-1}(s)| < \frac{\epsilon(1-\gamma)}{2\gamma}$$

Policy Evaluation - given π, how to derive V_π?

- Matrix Inversion - set ip linear equality ()no max for each state:

$$\forall s, V_\pi(s) = \{R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s')V_\pi(s')\}$$

- Successive approximation - essentially value iteration with fixed policy. Initialize V_π^0 arbitrarily and

$$V_\pi^t(s) := \{R(s, \pi(s)) + \sum_{s'} T(s, \pi(s), s')V_\pi^{t-1}(s')\}$$

  It's guaranteed to converge to V_π

Policy Iteration:

1. *Initialization:* pick an arbitrary initial decision policy $\pi_0 \in \Pi$ and set $i = 0$.
2. *Policy Evaluation:* solve for B_{π_i}.
3. *Policy Improvement:* find a new policy π_{i+1} that is a greedy policy w.r.t. V_{π_i} (i.e., $\pi_{i+1} \in \arg\max_{\pi\in\Pi}\{R_\pi + \gamma T_\pi V_{\pi_i}$ with ties resolved via a total precedence order over actions).
4. *Termination Check:* If $\pi_{i+1} \neq \pi_i$ then increment i and go to stept 2 else return π_{i+1}.

Comparing Value and Policy Iteration

- Value iteration:
  - Each iteration seen as doing 1-step of policy evaluation for current greedy policy'
  - Bootstrap with value estimate of previous policy.
- Policy iteration:
  - Each iteration is full evaluation of V_π for current policy π;
  - Then do greedy policy update.
- Modified policy iteration:
  - Like policy iteration, but $V_{\pi i}$ need only be closer to V^* than $V_{\pi i-1}$ (fixed number of steps of successive approximation for $V_{\pi i}$ suffices when bootstrapped with $V_{\pi i-1}$)
  - Typically faster than VI and PI in practice