Reinforcement Learning (INF11010)

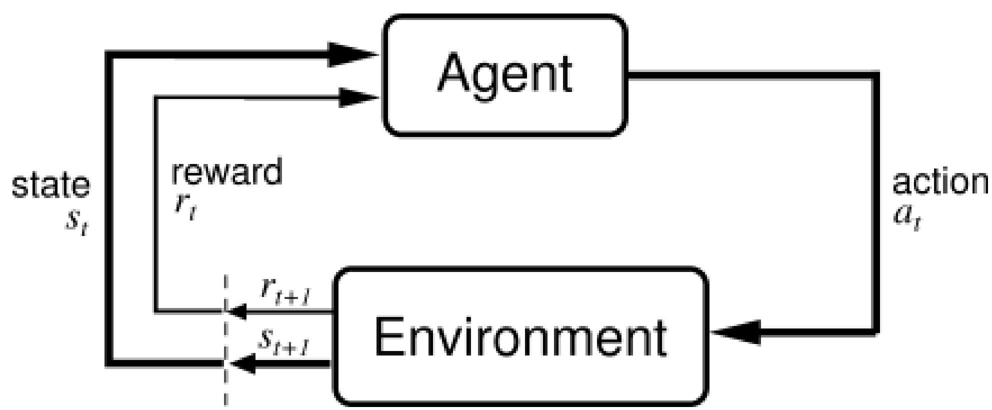
Lecture 7: Monte Carlo for RL

Pavlos Andreadis, February 9th 2018
with slides by Subramanian Ramamoorthy, 2017

Markov Decision Processes

- A finite Markov Decision Process (MDP) is a tuple (S, A, P, R, γ) where:

- S is a finite set of states
- A is a finite set of actions
- P is a state transition probability function
- R is a reward function
- γ is a discount factor



Methods Overview

- Dynamic Programming Methods:
  - *require a model*
  - *bootstrap*
- Monte Carlo Methods:
  - *do not* require a model
  - *do not* bootstrap
- Temporal-Difference Learning Methods:
  - *do not* require a model
  - *bootstrap*

Today’s Content

- Coursework 1 (questions/discussion)
- Monte Carlo (MC) Policy Evaluation
  - State-value and Action-value functions
- MC Control
- MC Exploring Starts
- Problems with MC Assumptions
- On-Policy MC Control

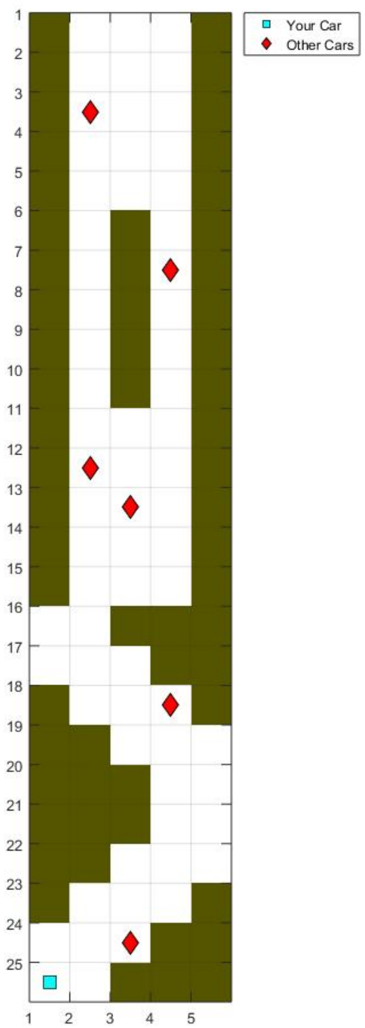
Coursework 1

- Actions:

  UP_LEFT

  UP

  UP_RIGHT



- Episodic task
  - Top row (row 1) = terminal states
- Transition Function:

  MDP_1.getTransitions
- Reward Function:

  MDP_1.getReward
- Policy to evaluate:

  pi_test1 / pi_test1_stateNumbers

[row, column] coordinates

numbered left→right and top->bottom

Monte Carlo Methods

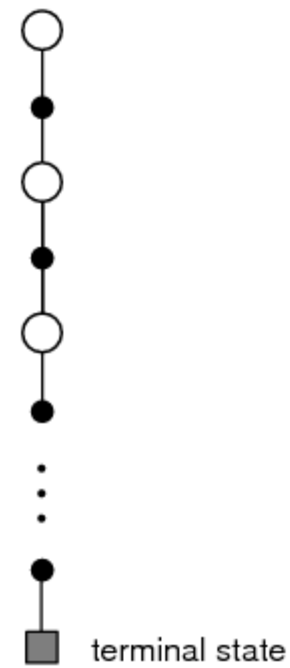
- Learn value functions
- Discover optimal policies
- Do not assume knowledge of model as in DP, i.e., $\mathcal{P}_{ss'}^{a}, \mathcal{R}_{ss'}^{a}$

- Learn from experience: Sample sequences of states, actions and rewards (s, a, r)
  - In simulated or real (e.g., physical robotic) worlds
  - Clearly, simulator *is* a model but not a *full* one as in a prob. distribution

- Eventually attain optimal behaviour (same as with DP)

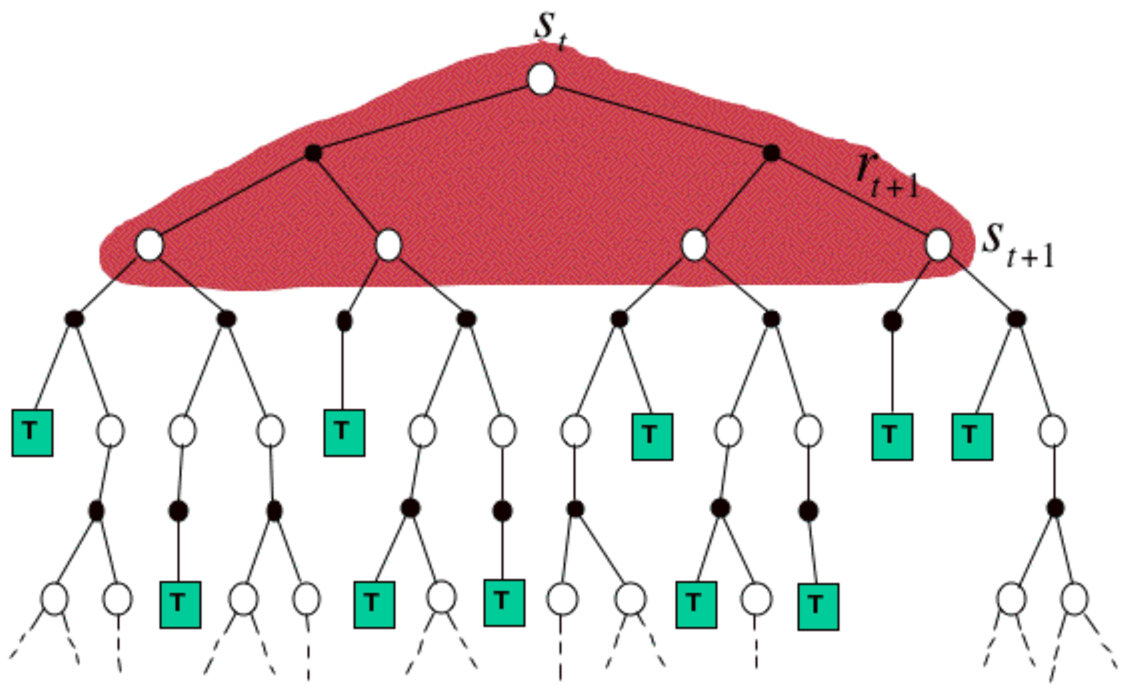
Backup in MC

- Does the concept of backup diagram make sense for MC methods?
- As in figure, MC does not sample all transitions
  - Root node to be updated as before
  - Transitions are dictated by policy
  - Acquire samples along a sample path
  - Clear path from eventual reward to states along the way (credit assignment easier)
- Estimates are different states are independent
  - Computational complexity **not** a function of state dimensionality



Pictorial: What does DP Do?

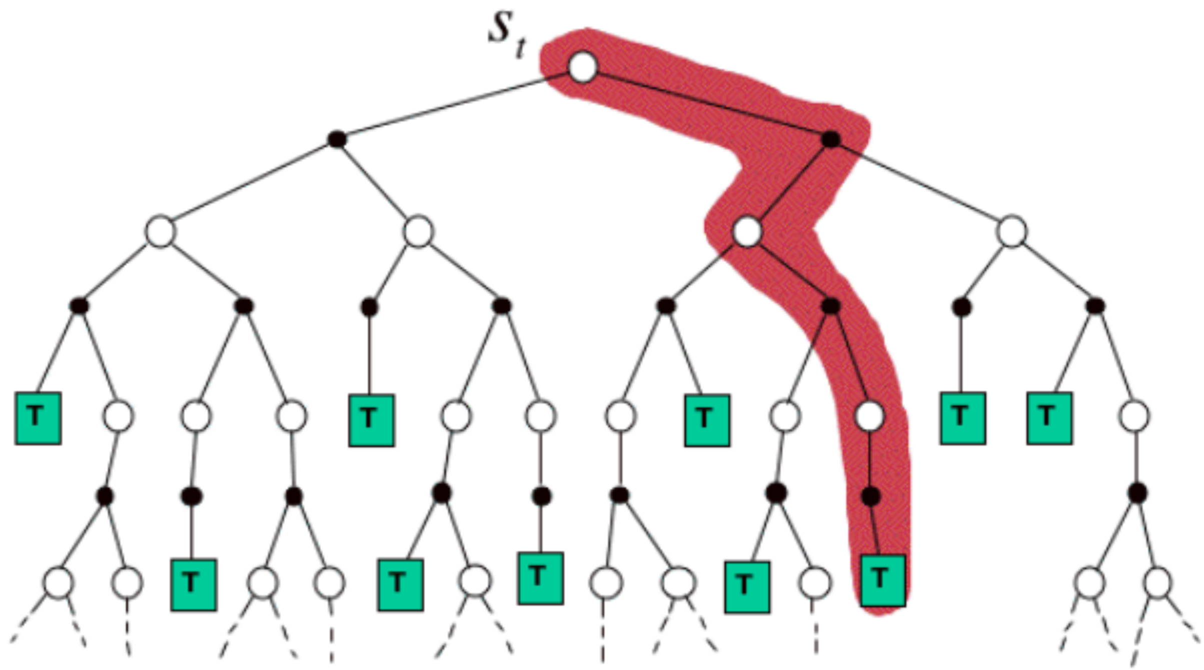
$$V(s_t) \leftarrow E_\pi\left\{r_{t+1} + \gamma\, V(s_t)\right\}$$



Pictorial: What does Simple MC Do?

$$V(s_t) \leftarrow V(s_t) + \alpha\left[R_t - V(s_t)\right]$$

where R_t is the actual return following state s_t.



Monte Carlo Policy Evaluation

- Goal: Approximate a value function $V^{\pi}(s)$
- Given: Some number of episodes under π which contain s
- Maintain average returns after visits to s

What is the effect of π?
What if it is deterministic?

- First visit vs. Every visit MC:
  - Consider a reward process $R(t) = r_t + \gamma r_{t+1} + \ldots$ and define the first visit time, $\tau = \min\{t | x = x_i\}$ and a set, $\Gamma = \{t | x = x_i\}$
  - First visit MC averages $\{R^i(\tau)\}, i = 1, \ldots, n$
    whereas every visit MC averages over $\{R^i(t_j)\}, i = 1, \ldots, n, t_j \in \Gamma$

First-visit Monte Carlo Policy Evaluation

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

(a) Generate an episode using π

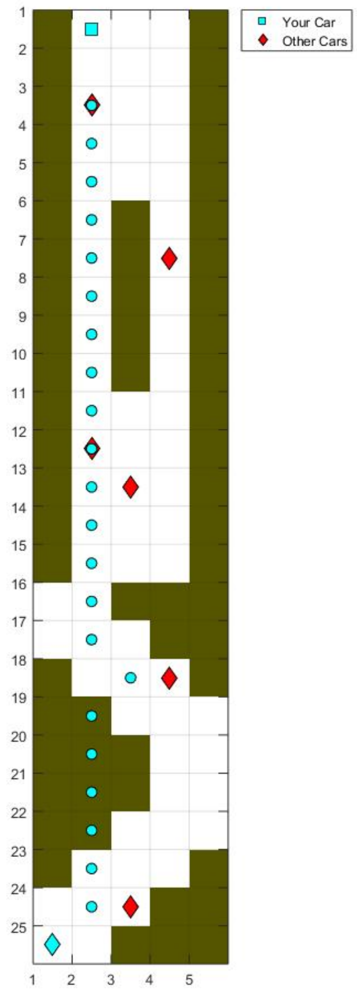
(b) For each state s appearing in the episode:

$R \leftarrow$ return following the first occurrence of s

Append R to $Returns(s)$

$V(s) \leftarrow \text{average}(Returns(s))$

Example: Road Fighter



- So, at every state, we know what actions are available…
- but we don't know anything of where we might transition, and with what probability…
- or what reward signals we might receive.
- Given a policy, we compute the average return starting from a state, across episodes.
- Obviously, the episodes need to terminate.
- Difference between first-time and any-time visit MC here?

Monte Carlo Estimation of Action Values

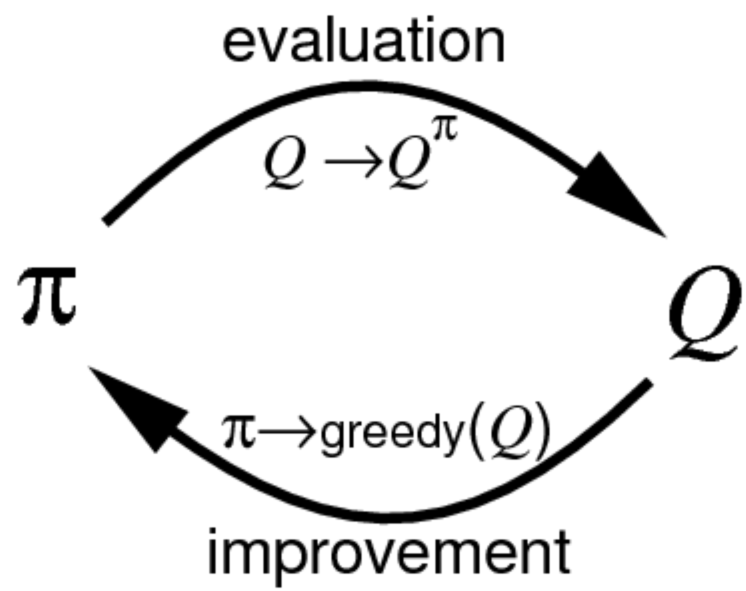
- Model is not available, so we do not know how states and actions interact
  - We want Q^*

- We can try to approximate $Q^\pi(s,a)$ using Monte Carlo method
  - Asymptotic convergence if every state-action pair is visited

- Explore many different starting state-action pairs: Equal chance of starting from any given state
  - Not entirely practical, but simple to understand

Monte Carlo Control

- Policy Evaluation:
  Monte Carlo method
- Policy Improvement:
  Greedify with respect to state-value of action-value function



Convergence of MC Control

- Policy improvement still works if evaluation is done with MC:

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \arg\max_a Q^{\pi_k}(s,a)) \\ &= \max_a Q^{\pi_k}(s,a) \\ &\geq Q^{\pi_k}(s, \pi_k(s)) \\ &= V^{\pi_k}(s). \end{aligned}$$

- $\pi_{k+1} \geq \pi_k$ by the policy improvement theorem
- Assumption: exploring starts and infinite number of episodes for MC policy evaluation (i.e., value function has stabilized)
- Things to do (as in DP):
  - update only to given tolerance
  - interleave evaluation/improvement

Monte Carlo Exploring Starts

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \leftarrow$ arbitrary
    $\pi(s) \leftarrow$ arbitrary
    $Returns(s,a) \leftarrow$ empty list

Fixed point is optimal policy π^*

Repeat forever:
    (a) Generate an episode using exploring starts and π
    (b) For each pair s, a appearing in the episode:
        $R \leftarrow$ return following the first occurrence of s, a
        Append R to $Returns(s,a)$
        $Q(s,a) \leftarrow \text{average}(Returns(s,a))$
    (c) For each s in the episode:
        $\pi(s) \leftarrow \arg\max_a Q(s,a)$

Can We Avoid Thorny Assumptions?

- Two major MC assumptions (infinite sampling and exploring all states) are unrealistic. How to circumvent the issue?

- Need to continually explore, ε-soft policies:

  - **On-policy** method: Explore in an ε-greedy manner

  - **Off-policy** method: Use a behaviour policy that is good at exploring, then infer optimal policy from that

On-Policy Monte Carlo Control

- Overall idea is still that of Generalized Policy Iteration (move *towards* greedy policy), but throw in continual exploration
- In order to always explore, we want to keep policy **ε-soft**:

$$\pi(s,a) > 0, \forall s, \forall a$$

- Moreover, one may really wish to adopt an **ε-greedy** policy:

$$\begin{aligned}\pi(s,a) &= \frac{\epsilon}{|\mathcal{A}|}, \text{if } a \textit{ is not} \text{ the greedy choice} \\ &= 1-\epsilon+\frac{\epsilon}{|\mathcal{A}|}, \text{if } a \textit{ is} \text{ the greedy choice}\end{aligned}$$

- In this case, we have $\pi(s,a) > \frac{\epsilon}{|\mathcal{A}|}, \forall s, \forall a$

The Policy Improvement Step

- Any ε-greedy policy w.r.t. Q^π is an improvement over any ε-soft policy π (Policy Improvement Theorem)

ε - greedy policy

$$
\begin{aligned}
Q^\pi(s,\pi'(s,a)) &= \sum_a \pi'(s,a) Q^\pi(s,a) \\
&= \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^\pi(s,a) + (1-\epsilon)\max_a Q^\pi(s,a) \\
&\geq \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^\pi(s,a) + (1-\epsilon)\sum_a \frac{\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|}}{1-\epsilon} Q^\pi(s,a) \\
& \text{This is bounded above by,} \\
&= \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^\pi(s,a) + \sum_a \pi(s,a) Q^\pi(s,a) \\
&= V^\pi(s)
\end{aligned}
$$

On-Policy MC Control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $Returns(s,a) \leftarrow$ empty list
  $\pi \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
  (a) Generate an episode using π
  (b) For each pair s, a appearing in the episode:
    $R \leftarrow$ return following the first occurrence of s, a
    Append R to $Returns(s,a)$
    $Q(s,a) \leftarrow$ average$(Returns(s,a))$
  (c) For each s in the episode:
    $a^* \leftarrow \arg\max_a Q(s,a)$
    For all $a \in \mathcal{A}(s)$:

$$\pi(s,a) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$$

Evaluate as before

Improve towards ε-greedy, not the max

Reading +

- Sections 5.1 to 5.4 of Sutton and Barto (1st Edition)
  http://incompleteideas.net/book/ebook/the-book.html

Optional (will take you away from course material):

- *Section 3.2 of* Ng, A. et al. (2004)
  Autonomous inverted helicopter flight via reinforcement learning