Markov Decision Processes

- A finite Markov Decision Process (MDP) is a tuple \((S, A, P, R, \gamma)\)
  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
  - \(\gamma\) is a discount factor
Today’s and Friday’s Content

- Dynamic Programming (DP) solutions to the RL problem
- Policy Evaluation + Policy Improvement $\rightarrow$
  - Policy Iteration $\parallel$ Value Iteration
- Backup diagrams and the Bellman Equation
- Generalised Policy Iteration
- Asynchronous Dynamic Programming
- Dynamic Programming methods in relation to other approaches
Dynamic Programming

- Algorithms for optimal policies given a *perfect model* of the environment as a Markov decision process (MDP)

- … but of theoretical importance.

- Applicable for exact solutions with discrete state & action model:

\[
P_{s,s'}^a = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}
\]

\[
R_{s,s'}^a = E\{r_{t+1} | a_t = a, s_t = s, s_{t+1} = s'\}
\]

- … and provide approximate solutions for continuous problems.
Bellman Optimality Equations

\[ V^*(s) = \max_a E\{r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a\} \]
\[ = \max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^*(s')] \]

\[ Q^*(s, a) = E\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a')|s_t = s, a_t = a\} \]
\[ = \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma \max_{a'} Q^*(s', a')] \]

for all \( s \in S, a \in A(s), \) and \( s' \in S. \)
Policy Evaluation

- There exists a unique solution as long as $\gamma < 1$ or termination is guaranteed:

$$V^\pi(s) = E_\pi \{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s_t = s \}$$

$$= E_\pi \{ r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s \}$$

$$= \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma V^\pi(s') \right]$$

- … which is a system of $|S|$ linear equations with $|S|$ unknowns
Iterative Policy Evaluation

- An iterative solution, starting from an arbitrary $V_0$ (but with terminal states having a value of 0) and computing...

\[ V_{k+1}(s) = E_\pi \{ r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s \} = \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma V_k(s') \right] \]

- … which converges to $V^\pi$ as $k \to \infty$
- At every iteration, every state is backed up
- For DP, this is a full backup, since we don’t sample next states
Backup Diagrams

\[ S_t \]

- State value function \( V \)
Backup Diagrams

- State value function $V$
Backup Diagrams

- State value function $V$
Backup Diagrams

\[ s_t \quad a_t \quad s_{t+1} \]

- State value function \( V \)
Backup Diagrams

- State value function $V$

Diagram showing transitions from $S_t$ to $S_{t+1}$ via actions $A_t$ with rewards $r$.
Backup Diagrams

- Action value function $Q$
Policy Improvement

• Consider a given policy $\pi$
  
  - … can we improve it by changing the action taken at a specific state $s$?
  
  - … yes if $Q^\pi(s, a) > V^\pi(s)$

• [Policy Improvement Theorem] Generally, for deterministic policies $\pi$, $\pi'$, if

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s), \ \forall s \in S$$

then

$$V^{\pi'}(s) \geq V^\pi(s), \ \forall s \in S$$
**greedy Policy Improvement**

- A policy improvement step would then be:

\[
\pi'(s) = \arg\max_a Q^\pi(s, a) \\
= \arg\max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')] 
\]

- Of course, this does not evaluate the value function for the new policy \( \pi' \), but if we put Policy Improvement and Policy Evaluation together, we get...
Policy Iteration

1. initialise $V$ and $\pi_0$ (arbitrarily)

2. perform Policy Evaluation

3. perform Policy Improvement

4. if the policy has changed go to 2.
Value Iteration

- ... is like Policy Iteration but with only a single backup of each state in the Policy Evaluation step.
- This still converges to an optimal policy.
- Policy Evaluation and Policy Improvement can be joined into a single update:

\[
V_{k+1}(s) = \max_a \mathbb{E}\{r_{t+1} + \gamma V_k(s_{t+1})|s_t = s, a_t = a\}
\]

\[
= \max_a \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma V_k(s') \right]
\]

- Need only compute the policy in the end.
Policy Iteration (concept)

Policy evaluation
- Estimate $v_\pi$
- Iterative policy evaluation

Policy improvement
- Generate $\pi' \geq \pi$
- Greedy policy improvement
Generalised Policy Iteration

Policy evaluation  Estimate $v_\pi$

Policy improvement  Generate $\pi' \geq \pi$
Asynchronous DP

- So far, all solutions have considered full sweeps of the state space.
- An Asynchronous DP procedure performs evaluation & improvement computations ...
  - without going through all the states or in any specific ordering
  - with any state or state-action values currently available
- To converge correctly, it needs to visit all states in expectation.
- These are not necessarily faster, but depending on the problem might help us improve convergence by e.g. avoiding states that do not appear in optimal trajectories.
- Examples:
  - Value iteration with only one state updated per iteration.
  - Real-time dynamic programming
DP Efficiency

- Value Iteration for $V$ (1 iteration) has complexity $O(|S|^2 |A|)$.
- Value Iteration for $Q$ (1 iteration) has complexity $O(|S|^2 |A|^2)$.

- Policy Iteration…
  - takes time polynomial in the problems size (state and action space)
  - converges much slower the closer $\gamma$ is to 1

- 1 iteration of Policy Iteration is slower than 1 iteration of Value Iteration, but Policy Iteration will generally require fewer iterations till convergence.
DP in Comparison to Other Methods

• 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
Reading +

- Chapter 4 of Sutton and Barto (1st Edition)

- Please join Piazza for announcements and support:
  [https://piazza.com/ed.ac.uk/spring2018/infr11010](https://piazza.com/ed.ac.uk/spring2018/infr11010)

Optional:

  *On the complexity of solving Markov decision problems.*

- Pashenкова, E. and Rish, I. and Dechter, R. (1996)
  *Value iteration and policy iteration algorithms for Markov decision problems.*