Part 2: Markov Decision Processes

EL 2805 – Reinforcement Learning

Alexandre Proutiere

KTH, The Royal Institute of Technology

Objectives of this part

Optimal control when the model is known

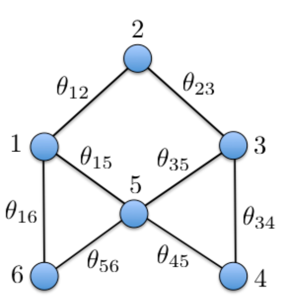
- An example to introduce dynamic programming: the "longest path" problem (or the hot potato problem)
- Markov Decision Processes: A model for sequential decision selection problem under uncertainty
- 3 main classes of MDP
    1. Finite horizon MDP
    2. Infinite horizon MDP: the discounted reward case
    3. Infinite horizon MDP: the average reward case
- For each class of MDP:
    1. Evaluate the average reward of a given policy
    2. Solve Bellman's equations to find the value function and the best policy

Part 2: Outline

1. Dynamic Programming for the Hot Potato Problem
2. Markov Decision Processes
3. Finite-time horizon MDPs
    a. Policy evaluation
    b. Value function and optimal policy through Dynamic Programming
4. Discounted Infinite-Horizon MDPs
    a. Policy evaluation
    b. Value function and optimal policy through Value Iteration and Policy Iteration algorithms
    c. Complexity issues

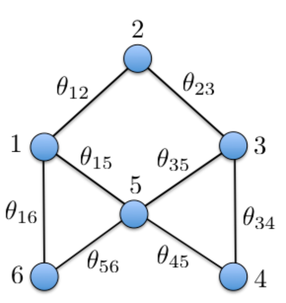
1. The Hot Potato Problem

A hot potato navigates in a graph. When the potato is at a node, the decision maker selects a neighbouring node, and the potato is sent to this node. On a pair of nodes (i, j), the probability that the transmission is successful is θ_{ij} (if not, the potato remains at node i). In T decisions, we aim at maximizing the number of successful transmissions. By definition, for any pair i, j, ($\theta_{ij} = \theta_{ji} > 0$) iff ($i \in \mathcal{N}(j)$). The θ_{ij}'s are **known**.



The Hot Potato Problem

What should we do when T is very large?
Move towards the pair of nodes $(i^\star, j^\star) \in \arg\max_{(i,j)\in G} \theta_{ij}$, and keep sending the potato back and forth from i to j ...
Now what if T is not that large?



The Hot Potato Problem

Model: collect a unit reward when moving from one node to another
Key observation: at any intermediate step, the optimal future decisions only depend on the current state (the position of the potato) and the remaining time before the horizon expires – the past does not matter!

$\underline{T = 1.}$ Starting at node i, the optimal average reward and the corresponding decision are:

$$\begin{cases} V_1(i) = \max_{j \in \mathcal{N}(i)} \theta_{ij} \\ i^\star \in \arg\max_{j \in \mathcal{N}(i)} \theta_{ij} \end{cases}$$

The Hot Potato Problem

$\underline{T = 2.}$ Starting at node i, if node $j \in \mathcal{N}(i)$ is selected, then:

- either the potato moves to j (w.p. θ_{ij}), and we collect an average reward of $1 + V_1(j)$
- or the potato does not move (w.p. $1 - \theta_{ij}$), and we collect a reward of $V_1(i)$

Hence the optimal average reward and the corresponding first decision are:

$$\begin{cases} V_2(i) = \max_{j \in \mathcal{N}(i)} \theta_{ij}(1 + V_1(j)) + (1 - \theta_{ij})V_1(i) \\ i^\star \in \arg\max_{j \in \mathcal{N}(i)} \theta_{ij}(1 + V_1(j)) + (1 - \theta_{ij})V_1(i) \end{cases}$$

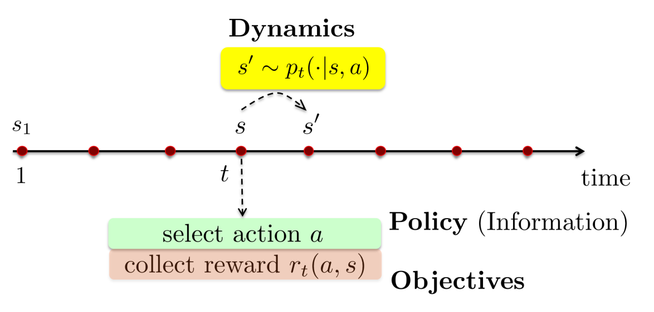
The Hot Potato Problem

$\underline{T = n.}$ Starting at node i, the optimal average reward and the corresponding first decision are:

$$\begin{cases} V_n(i) = \max_{j \in \mathcal{N}(i)} \theta_{ij}(1 + V_{n-1}(j)) + (1 - \theta_{ij})V_{n-1}(i) \\ i^\star \in \arg\max_{j \in \mathcal{N}(i)} \theta_{ij}(1 + V_{n-1}(j)) + (1 - \theta_{ij})V_{n-1}(i) \end{cases}$$

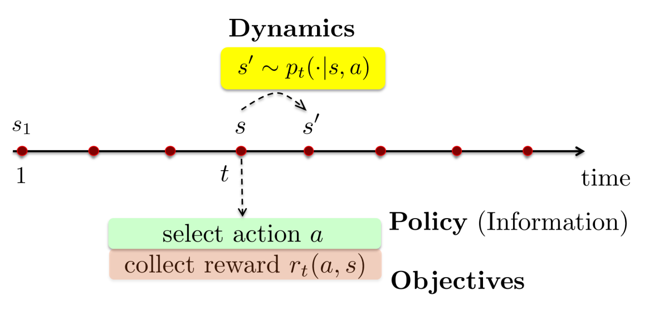
The optimal policy is **Markovian**, and can be computed along with its average reward by solving **Bellman's equation** using **Dynamic Programming**

2. Markov Decision Processes



- Fully observable state and reward
- Known reward distribution and transition probabilities
- a_t function of $h_t = (s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$
- Markovian dynamics: $\mathbb{P}[s_{t+1}|h_t, a_t] = p_t(s_{t+1}|s_t, a_t)$
- Reward at time t: $r_t(s_t, a_t)$ (can be extended to random rewards – see notes)

Assumptions



- State space S: finite, countably infinite, or a compact set of $\mathbb{R}^d$. Finite unless otherwise specified
- Finite action space A: for any $s \in S$, the set of available actions is A_s. $A = \cup_{s \in S} A_s$

Finite Horizon

- Initial state s_1
- Finite time horizon T
  - Objective: find a sequential decision policy π maximizing the expected reward up to time T:

$$R(s_1, a_1^\pi, s_1^\pi, \ldots, s_T^\pi, a_T^\pi) = \sum_{t=1}^{T} r_t(s_t^\pi, a_t^\pi)$$

$$\text{maximize over } \pi : \quad \mathbb{E}[R(s_1, a_1^\pi, s_1^\pi, \ldots, s_T^\pi, a_T^\pi)]$$

Infinite Horizon

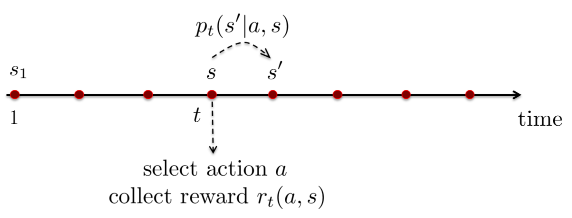
- Initial state s_1
- Infinite time horizon $T = \infty$
- Stationary transitions and rewards: $p(s'|s, a)$ and $r(s, a)$
  - Objective 1: maximize the discounted expected reward ($\lambda \in (0, 1)$)

$$\lim \inf_{T \to \infty} \mathbb{E}[\sum_{t=1}^{T} \lambda^{t-1} r(s_t^{\pi}, a_t^{\pi})]$$

  - Objective 2: maximize the ergodic expected reward

$$\lim \inf_{T \to \infty} \mathbb{E}[\frac{1}{T} \sum_{t=1}^{T} r(s_t^{\pi}, a_t^{\pi})]$$

MDPs – Summary



- A Markov Decision Process is defined through:

$$\{T, S, (A_s, p_t(\cdot|s, a), r_t(s, a), 1 \leq t \leq T, s \in S, a \in A_s)\}$$

- Three types of objectives:
  1. T finite – expected total reward
  2. $T = \infty$ – expected discounted reward
  3. $T = \infty$ – expected ergodic reward

Decision Rules or Policies

- History up to time t: $h_t = (s_1, a_1, \ldots, s_{t-1}, a_{t-1}, s_t) \in (S \times A)^t \times S$
- A priori, the decision selected at time t could depend on the entire history
- The action selected could be random!
- We distinguish different types of policies
    - History-dependent Randomised: HR
    - History-dependent Deterministic: HD
    - Markov Randomised: MR
    - Markov Deterministic: MD

Decision Rules or Policies

$\pi = (\pi_t, 1 \leq t \leq T)$

- History-dependent Randomised: $\pi_t : (S \times A)^t \times S \to \mathcal{P}(A_{s_t})$
  $q_{\pi_t(h_t)}(a)$: probability to select action a at time t
- History-dependent Deterministic: $\pi_t : (S \times A)^t \times S \to A_{s_t}$
  $\pi_t(h_t)$: action selected at time t
- Markov Randomised: $\pi_t : S \to \mathcal{P}(A_{s_t})$
  $q_{\pi_t(s_t)}(a)$: probability to select action a at time t
- Markov Deterministic: $\pi_t : S \to A_{s_t}$
  $\pi_t(s_t)$: action selected at time t

Observe that:

$$MD \subset MR \subset HR$$

$$MD \subset HD \subset HR$$

Markovian deterministic policies are *most often* **optimal – forget about more complicated history-based policies.**

MDP with Discounted Expected Reward

$$\max_{\pi} \lim_{T\to\infty} \mathbb{E}[\sum_{t=1}^{T} \lambda^{t-1} r_t(s_t^\pi, a_t^\pi)]$$

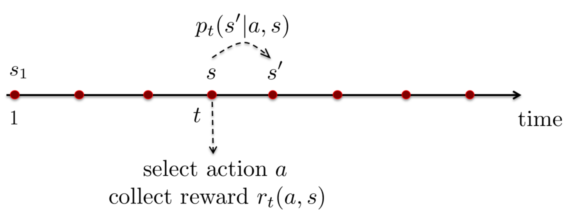
Two interpretations:

- **Interest rate.** The value of a unit reward decreases with time at geometric rate λ
- **Random time horizon.** the decision maker has a time horizon T geometrically distributed $\mathbb{P}[T = k] = (1-\lambda)\lambda^k$; $\mathbb{E}[T] = 1/(1-\lambda)$

Why such an objective? How should we choose λ?

- Life is short!
- Non-stationary environments. Select λ such that $1 \ll 1/(1-\lambda)$ and $1/(1-\lambda) \ll$ coherence time

3. Finite-horizon MDP



- State space: S, actions available in state $s \in S$, A_s $(A \cup_{s \in S} A_s)$
- Transition probabilities at time t: $p_t(s'|s,a)$
- Reward at time t: $r_t(a,s)$
- Objective: find a policy $\pi \in MD$ maximising (over all possible policies)

$$\mathbb{E}\left[\sum_{t=1}^{T} r_t(s_t^\pi, a_t^\pi)\right]$$

The Value Function

- The value function is the maximal expected reward depending on the time horizon T and the initial state s:

$$V_T^\star(s) = \sup_{\pi \in MD} V_T^\pi(s)$$

where $V_T^\pi(s)$ is the average reward achieved under π with initial state s, i.e.,

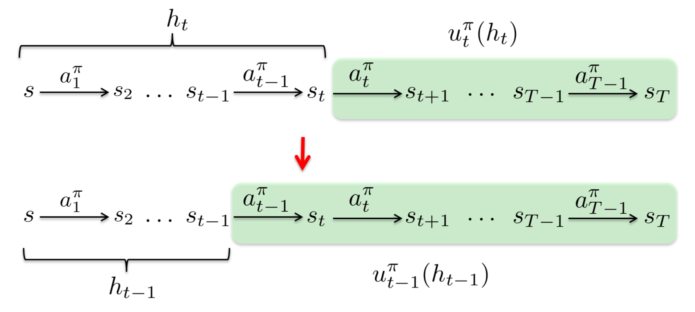
$$\begin{aligned} V_T^\pi(s) &= \mathbb{E}\left[\sum_{t=1}^{T} r_t(s_t^\pi, a_t^\pi) | s_1^\pi = s\right] \\ &= \mathbb{E}_s\left[\sum_{t=1}^{T} r_t(s_t^\pi, a_t^\pi)\right] \end{aligned}$$

- The "sup" is achieved – finite action space.

We wish to compute $\forall s \in S$: $V_T^\pi(s) = \mathbb{E}_s\left[\sum_{t=1}^{T} r_(s_t^\pi, a_t^\pi)\right]$
Remaining average reward starting at time t given some given current state s:

$$u_t^\pi(s) = \mathbb{E}\left[\sum_{u=t}^{T} r_u(s_u^\pi, a_u^\pi) \middle| s_t^\pi = s\right]$$

- Start with: $u_T^\pi(s_T) = r_T(s_T, \pi(s_T))$ for all s_T
- Backward recursion to compute u_{t-1}^π from u_t^π



Average reward under $\pi \in MD$

- At time $t-1$, for all s_{t-1}
  - a is chosen
  - the reward $r_{t-1}(s_{t-1}, a)$ is collected
  - the state becomes $s_t = j$ with probability $p_{t-1}(j|s_{t-1}, a)$
  - the average reward until T is $u_t^\pi(s_t)$

  Hence:

$$u_{t-1}^\pi(s_{t-1}) = r_{t-1}(s_{t-1}, a) + \sum_{j \in S} p_{t-1}(j|s_{t-1}, a) u_t^\pi(j)$$

- We obtain: $V_T^\pi(s) = u_1^\pi(s)$ for any s

3.b Bellman's Equation – Dynamic Programming

Bellman's equation provides a recursive way of computing the value function and the optimal policy. Maximal average reward starting at time t: $u_t^\star(s_t) = \sup_{\pi \in MD} u_t^\pi(s_t)$, estimated by $u_t^B(s_t)$ (B stands for 'Bellman')

1. For all s_T, $u_T^B(s_T) = \max_a r_T(s_T, a)$
2. For all $t \in \{T, T-1, \dots, 2\}$, for all s_{t-1},

$$u_{t-1}^B(s_{t-1}) = \max_{a \in A_{s_{t-1}}} \left[r_{t-1}(s_{t-1}, a) + \sum_{j \in S} p_{t-1}(j|s_{t-1}, a) u_t^B(j) \right]$$

Theorem. $u^B = u^\star$

Finite-horizon MDP: Summary

Bellman's equations: For all s_T, $u_T^\star(s_T) = \max_a r_T(s_T, a)$
For all $t = T-1, T-2, \ldots, 1$

$$u_t^\star(s_t) = \max_{a \in A_{s_t}} \underbrace{\left[r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^\star(s_t, a, j) \right]}_{Q_t(s_t, a) \text{ optimal reward from } t \text{ if } a \text{ selected}}$$

An optimal policy π is obtained by selecting $\pi_t(s_t)$ at time t such that

$$Q_t(s_t, \pi_t(s_t)) = \max_{a \in A_{s_t}} Q_t(s_t, a)$$

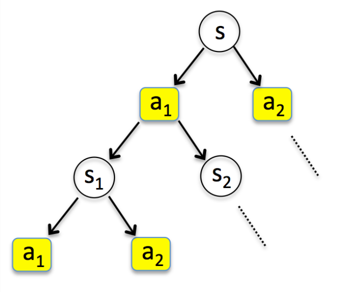
Solving Bellman's equation requires $\Theta(S^2 AT)$ operations

Richard Bellman

1920 - 1984
American applied mathematician

Introduced **Dynamic Programming** (DP) as a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions.

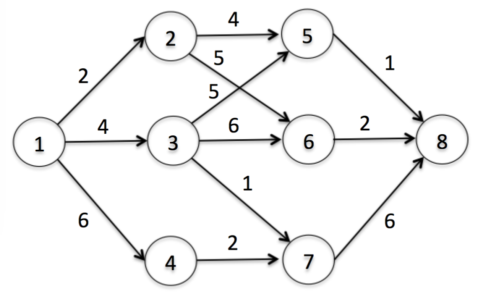
Bellman's breakthrough



Decision tree with depth T: it has $A^T S^{T+1}$ leaves (complexity of optimising over history-dependent policies)

Solving Bellman's equation for optimal MD policies requires $S^2 AT$ operations!

Example: Max-weight routing



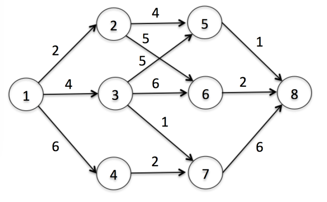
Find the max-weight path from the source 1 to the destination 8

Example: Max-weight routing, DP formulation

- States: positions 1, 2, 3, 4, 5, 6, 7, 8
- Actions: the possible next state, e.g. $A_3 = \{5, 7\}$
- Rewards: edge weigths, e.g. if edge $(3, 5)$ selected, reward $w_{35} = 5$
- Transitions: deterministic, e.g. $p(5|5, 3) = 1$
- Time horizon: T greater that the maximum path length, e.g. $T = 3$
- Max path weight starting at state s: $u^\star(s)$
- Bellman equations: $u^\star(8) = 0$, $A_8 = \emptyset$, and for $s \neq 8$,

$$u^\star(s) = \max_{j \in A_s} \left[w_{sj} + u^\star(j) \right]$$

Example: Max-weight routing, solution



$u^\star(8) = 0$

$u^\star(5) = 1$
$\pi^\star(5) = 8$

$u^\star(6) = 2$
$\pi^\star(6) = 8$

$u^\star(7) = 6$
$\pi^\star(7) = 8$

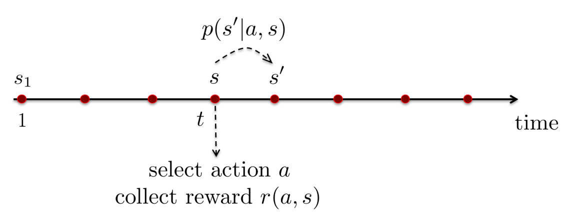
$u^\star(2) = 7$
$\pi^\star(2) = 6$

$u^\star(3) = 8$
$\pi^\star(3) = 6$

$u^\star(4) = 8$
$\pi^\star(4) = 7$

$u^\star(1) = 12$
$\pi^\star(1) = 4$

4. Infinite-horizon discounted MDP



- State space: S finite or countably infinite
- Actions available in state $s \in S$, A_s ($A = \cup_{s \in S} A_s$)
- Stationary transition probabilities $p(s'|s, a)$ and rewards $r(a, s)$, uniformly bounded: $\forall a, s, \ |r(s, a)| \leq 1$
- Objective: for a given discount factor $\lambda \in [0, 1)$, find a policy $\pi \in MD$ maximising (over all possible policies)

$$\lim_{T \to \infty} \mathbb{E} \left[\sum_{t=1}^{T} \lambda^{t-1} r(s_t^\pi, a_t^\pi) \right]$$

The Value Function

- The value function is the maximal expected reward depending on the discount factor λ and the initial state s:

$$V_\lambda^\star(s) = \sup_{\pi \in MD} V_\lambda^\pi(s)$$

  where $V_\lambda^\pi(s)$ is the average reward achieved under π with initial state s, i.e.,

$$V_\lambda^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \lambda^{t-1} r(s_t^\pi, a_t^\pi) | s_1^\pi = s\right] = \mathbb{E}_s\left[\sum_{t=1}^{\infty} \lambda^{t-1} r(s_t^\pi, a_t^\pi)\right]$$

- The "sup" is achieved – finite action space

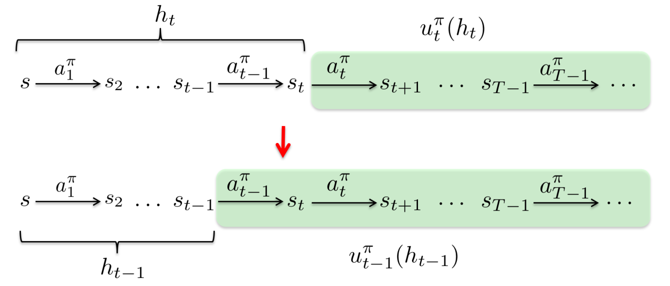
4.a Policy evaluation

Can we compute the average discounted reward $V_\lambda^\pi(s)$ under π?
Through recursive arguments like in the finite horizon case?

Let $\pi = (\pi_1, \pi_2, \ldots) \in MD$. Average reward starting at time t in state s:

$$u_t^\pi(s) = \mathbb{E}\left[\sum_{u=t}^{\infty} \lambda^{u-t} r_u(s_u^\pi, a_u^\pi) | s_t^\pi = s\right]$$

Backward recursion to compute u_{t-1}^π from u_t^π



Average reward under $\pi \in MD$

- At time $t-1$
  - $a = \pi_{t-1}(s_{t-1})$ is chosen
  - the reward $r(s_{t-1}, a)$ is collected
  - the state becomes $s_t = j$ with probability $p(j|s_{t-1}, a)$
  - the average reward from t is $\lambda u_t^\pi(s_t)$

  Hence:

$$u_{t-1}^\pi(s_{t-1}) = r(s_{t-1}, a) + \lambda \sum_{j \in S} p(j|s_{t-1}, a) u_t^\pi(j)$$

- Fine ... but we can not initialise the backward induction!

Notations

- $\mathcal{V}$ set of bounded functions from S to $\mathbb{R}$, with the norm defined as: for $V \in \mathcal{V}$, $\|V\| = \sup_{s \in S} |V(s)| < \infty$
- Let $MD_1 := \{\pi_1 : S \to A\}$ denote the set of one-step deterministic decision policies
- Define for any $\pi_1 \in MD_1$

$$r_{\pi_1}(s) := r(s, \pi_1(s)),\ p_{\pi_1}(j|s) := p(j|s, \pi_1(s))$$

$$P_{\pi_1}\text{: the matrix with entries } p_{\pi_1}(j|s)$$

Notations

With these notations, we have for all $V \in \mathcal{V}$ and $\pi_1 : S \to A$, $r_{\pi_1} + \lambda P_{\pi_1} V \in \mathcal{V}$ with

$$(r_{\pi_1} + \lambda P_{\pi_1} V)(s) = r(s, \pi_1(s)) + \lambda \sum_j p(j|s, \pi_1(s)) V(j)$$

We can also express the average reward of a policy $\pi = (\pi_1, \pi_2, \ldots)$ in MD in a compact form as an element of $\mathcal{V}$:

$$\begin{aligned} V^\pi &= r_{\pi_1} + \lambda P_{\pi_1} r_{\pi_2} + \lambda^2 P_{\pi_1} P_{\pi_2} r_{\pi_3} + \ldots \\ &= r_{\pi_1} + \sum_{t=1}^{\infty} \lambda^t P_\pi^t r_{\pi_{t+1}} \end{aligned}$$

where $P_\pi^t := P_{\pi_1} \ldots P_{\pi_t}$

Note: we drop the subscript λ from now on

Stationary policies

A stationary policy $\pi = (\pi_1, \pi_2, \ldots)$ is a policy in MD applying the **same one-step decision** every step, i.e., $\pi_t = \pi_1$ for all t

Under such a policy, the average reward satisfies:

$$V^\pi = r_{\pi_1} + \lambda P_{\pi_1} V^\pi$$

Indeed since $\pi_1 = \pi_2 = \pi_3 = \ldots$,

$$\begin{aligned} V^\pi &= r_{\pi_1} + \lambda P_{\pi_1}(r_{\pi_2} + \lambda P_{\pi_2} r_{\pi_3} + \ldots) \\ &= r_{\pi_1} + \lambda P_{\pi_1} \underbrace{(r_{\pi_1} + \lambda P_{\pi_1} r_{\pi_1} + \ldots)}_{=V^\pi} \end{aligned}$$

P_{π_1} is a stochastic matrix, and hence the linear operator $I - \lambda P_{\pi_1}$ is a contraction[1]: $\|I - \lambda P_{\pi_1}\| < 1$. Thus $V^\pi = (I - \lambda P_{\pi_1})^{-1} r_{\pi_1}$.

[1] $P : \mathcal{V} \to \mathcal{V}$ has norm $\|P\| = \sup_{V \in \mathcal{V}} \|H(V)\| / \|V\|$

4.b Bellman's equation

- For a deterministic stationary policy π:

$$V^\pi(s) = r(s, \pi_1(s)) + \lambda \sum_j p(j|s, \pi_1(s)) V^\pi(j)$$

- Bellman's equation obtained by selecting the *best* action:

$$\forall s \in S, V^B(s) = \max_{a \in A_s} \left[r(s, a) + \lambda \sum_j p(j|s, a) V^B(j) \right]$$

- (Non-linear) **Bellman operator** $\mathcal{L} : \mathcal{V} \to \mathcal{V}$ defined by: for all $V \in \mathcal{V}$, $\mathcal{L}(V) = \sup_{\pi_1 \in MD_1}(r_{\pi_1} + \lambda P_{\pi_1} V)$ or equivalently by

$$\forall s \in S,\ \mathcal{L}(V)(s) = \max_{a \in A_s} \left[r(s, a) + \lambda \sum_j p(j|s, a) V(j) \right]$$

Bellman's equation

V^B is a fixed point of $\mathcal{L}$, i.e., $\mathcal{L}(V^B) = V^B$

$$\iff \forall s \in S, V^B(s) = \sup_{a \in A_s} \left[r(s,a) + \lambda \sum_j p(j|s,a) V^B(j) \right]$$

Theorem. The operator $\mathcal{L}$ is a contraction mapping of $\mathcal{V}$. Thus it has a unique fixed point V^B, solution of Bellman's equation. Furthermore:

$$V^B = V^\star = \sup_{\pi \in MD} V^\pi$$

Infinite-horizon discounted MDP: Summary

Bellman's equations: For all s,

$$V^\star(s) = \max_{a \in A_s} \underbrace{\left[r(s,a) + \lambda \sum_{j \in S} p(j|s,a) V^\star(j) \right]}_{Q(s,a) \text{ optimal reward from state } s \text{ if } a \text{ selected}}$$

or equivalently $V^\star = \mathcal{L}(V^\star)$.

An optimal policy π is stationary $\pi = (\pi_1, \pi_1, \ldots)$ where $\pi_1 \in MD_1$ is defined by: for any s,

$$\pi_1(s) = \arg\max_{a \in A_s} Q(s,a)$$

Q is referred to as the Q-function.

It remains to solve Bellman's equations ...

Algorithms

To find the optimal policy, we need to solve Bellman's equations

- A fixed point iteration problem
    1. Value iteration
    2. Policy iteration
- Other methods, e.g. Linear Programming

The Value Iteration (VI) algorithm

Parameter. Precision ϵ

1. **Initialization.** Select a value function $V_0 \in \mathcal{V}$, $n = 0$, $\delta \gg 1$
2. **Value improvement.** While $(\delta > \frac{\epsilon(1-\lambda)}{\lambda})$ do
   (a) $V_{n+1} = \mathcal{L}(V_n)$, i.e., for all $s \in S$
   $V_{n+1}(s) = \sup_{a \in A_s}(r(s,a) + \lambda \sum_j p(j|s,a)V_n(j))$
   (b) $\delta = \|V_{n+1} - V_n\|$, $n \leftarrow n + 1$
3. **Output.** $\pi = (\pi_1, \pi_1, \ldots)$ with

$$\forall s \in S,\ \pi_1(s) \in \arg\max_{a \in A_s}(r(s,a) + \lambda \sum_j p(j|s,a)V_n(j))$$

The VI algorithm: Properties

- VI converges since $\mathcal{L}$ is a contraction mapping
- When it stops, VI returns an ϵ-optimal policy
- Complexity
  - The VI algorithm requires $\Theta(S^2 A)$ (floating) operations per iteration
  - Number of iterations?

The Howard's Policy Iteration (PI) algorithm

1. **Initialization.** Select a one-step policy π_0, $n = 0$
2. **Policy evaluation.** Evaluate the value V_n^π of $\pi = (\pi_n, \pi_n, \ldots)$ by solving:

$$\forall s \in S,\ V_n^\pi(s) = r(s, \pi_n(s)) + \lambda \sum_j p(j|s, \pi_n(s)) V_n^\pi(j)$$

3. **Policy improvement.** Update the one-step policy:

$$\forall s \in S,\ \pi_{n+1}(s) = \arg\max_{a \in A_s} (r(s,a) + \lambda \sum_j p(j|s,a) V_n^\pi(j))$$

4. **Stopping criterion.** If $\pi_{n+1} = \pi_n$, return π_n. Otherwise $n := n + 1$, and go to 2.

The Simplex-PI Algorithm

1. **Initialization.** Select a one-step policy π_0, $n = 0$
2. **Policy evaluation.** Evaluate the value V_n^π of $\pi = (\pi_n, \pi_n, \ldots)$ by solving: $V_n^\pi = r_{\pi_n} + \lambda P_{\pi_n} V_n^\pi$

$$\forall s \in S,\ V(s) = \max_{a \in A_s}(r(s,a) + \lambda \sum_j p(j|s,a) V_n^\pi(j))$$

$$s_0 \in \arg\max_{s \in S}(V(s) - V_n^\pi(s))$$

3. **Policy improvement.** Update the one-step policy: $\forall s \neq s_0,\ \pi_{n+1}(s) = \pi_n(s)$ and

$$\pi_{n+1}(s_0) = \arg\max_{a \in A_{s_0}}(r(s_0,a) + \lambda \sum_j p(j|s_0,a) V_n^\pi(j))$$

4. **Stopping criterion.** If $\pi_{n+1} = \pi_n$, return π_n. Otherwise $n := n + 1$, and go to 2.

The PI algorithm: Properties

- Under the PI algorithm, V_n^π is increasing in n
- When S and A are finite, PI terminates with an optimal policy
- Complexity
  - In each iteration, the policy evaluation can be done in $\Theta(S^\omega)$ (floating) operations, and the policy improvement requires $\Theta(S^2 A)$ (floating) operations
    $\Theta(S^\omega)$ is the complexity of inverting a $S \times S$ matrix
  - Number of iterations?

4.c Complexity issues

How many iterations do we need to compute an optimal policy under the VI or the PI algorithm?
How many arithmetic operations do we need?

- Examples
- Complexity results

Numerical Experiments

Four examples:

(i) The VI and PI algorithms are fast for randomly generated MDPs

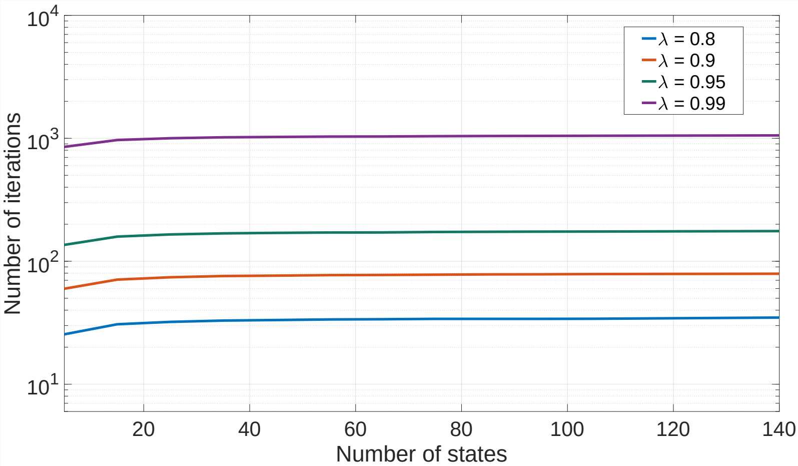
(ii) PI: the number of iterations could grow linearly with S

(iii) VI: the number of iterations could grow exponentially with A

(iv) VI: the number of iterations could scale as $\log(\frac{1}{1-\lambda})\frac{1}{1-\lambda}$

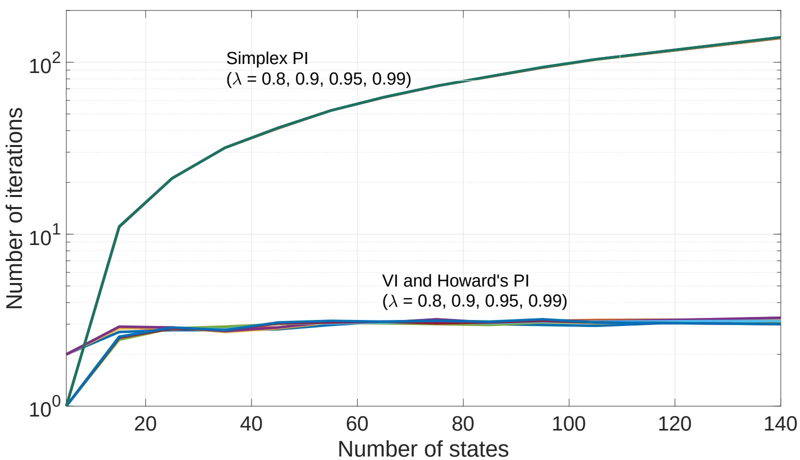
(i) Randomly generated MDPs

Convergence time of values for VI ($\epsilon = 0.01$), for randomly generated MDPs and various discount factors



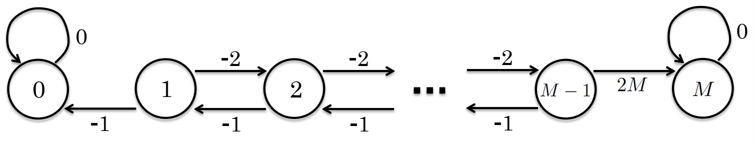
(i) Randomly generated MDPs

Convergence time of policies for VI and PI variants, for randomly generated MDPs and various discount factors



(ii) The PI Algorithm

$$S = \{0, \dots, M\},\ A_s = \{0, 1\}, \quad \forall s$$
$$p(s-1|s,0) = 1,\ p(s+1|s,1) = 1$$
$$r(s,0) = -1,\ r(s,1) = -2, \quad \forall s = 1, \dots, M-2$$
$$r(M-1,0) = -1,\ r(M-1,1) = 2M$$
$$p(0|0,\cdot) = 1 = p(M|M,\cdot), \quad r(0,\cdot) = 0 = r(M,\cdot)$$



Optimal policy: $\pi^\star(s) = 1,\ \forall s \neq 0, M,\ \pi^\star(0) = 0 = \pi^\star(M)$.

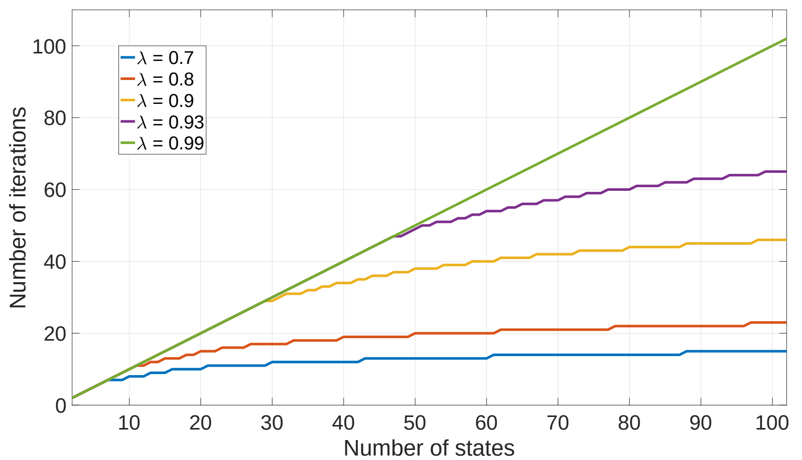
(ii) The PI Algorithm

Policy Iteration with $\pi_0(s) = 0, \ \forall s \neq M-1, \quad \pi_0(M-1) = 1$

At iteration n, π_n differs from π_{n-1} in state $s = M - n - 1$, flipping the optimal action from left to right. Thus, it takes $M-1$ steps so that in all states $\pi_n(s) = 1$.

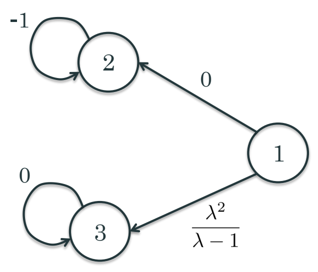
If λ is very close to 1, PI could take $M-1$ steps to compute the optimal policy.

(ii) The PI Algorithm



(iii) The VI Algorithm

$S = \{1,2,3\}$, $A_1 = \{0,1\}$, $A_2 = \{0\} = A_3$
$p(2|1,0) = 1$, $p(3|1,1) = 1$, $\quad p(2|2,0) = 1 = p(3|3,0)$
$r(1,0) = 0$, $r(1,1) = \frac{\lambda^2}{\lambda-1}$, $r(2,0) = -1$, $r(3,0) = 0$



The expected reward of action 1 from state 0 is $\frac{\lambda}{\lambda-1}$, which is smaller that $\frac{\lambda^2}{\lambda-1}$. Hence the optimal policy chooses action 2 in state 0.

(iii) The VI Algorithm

VI equations with $V_0(s) = 0$ for all s:

$$V_n(0) = \max\left[\lambda V_{n-1}(1), \frac{\lambda^2}{\lambda - 1} + \lambda V_{n-1}(2)\right]$$
$$V_n(1) = -1 + \lambda V_{n-1}(1)$$
$$V_n(2) = 0 + \lambda V_{n-1}(2)$$

so that

$$V_n(1) = \frac{1-\lambda^n}{1-\lambda}, \;\; V_n(2) = 0$$

Hence, it takes N iterations for VI to identify the optimal action at state 1, where N satisfies

$$\frac{\lambda(1-\lambda^{N-1})}{\lambda - 1} < \frac{\lambda^2}{\lambda - 1}$$

hence $N > \frac{\log(1-\lambda)}{\log\lambda} + 1$.

(iv) The VI Algorithm (bis)

$S = \{1, 2, 3\}$, $A_1 = \{0, 1, \ldots, k\}$, $A_2 = \{0\} = A_3$
$p(2|1, i) = 1, \quad \forall i = 1, \ldots, k$, $p(3|1, 0) = 1$
$p(2|2, 0) = 1 = p(3|3, 0)$
$r(1, 0) = r(2, 0) = 0$, $r(3, 0) = 1$, $r(1, i) = \frac{\lambda}{1-\lambda}(1 - \exp(-M_i))$
where $0 < M_1 < \ldots < M_k$

If in state 1, choosing action $i \geq 1$ leads to 2 and provides a total reward $r(1, i)$
If in state 0, choosing action 0 leads to 3 and provides a total reward $\frac{\lambda}{1-\lambda}$
Hence the optimal policy consists in selecting 0 in state 1.

(iv) The VI Algorithm (bis)

Value Iteration with $V_0 = 0$:
For all $n \geq 1$, we have:

$$V_n(2) = 0, \quad V_n(3) = \frac{1 - \lambda^n}{1 - \lambda}$$

$$V_n(1) = \max \left[\frac{\lambda}{1 - \lambda}(1 - \exp(-M_k)), \frac{\lambda}{1 - \lambda}(1 - \lambda^{n-1}) \right]$$

Hence the policy computed from V_n is optimal if and only if:

$$n \geq 1 + \frac{M_k}{-\log(\lambda)}$$

Choose $M_i = 2^i$ for all i. $k + 3$ actions, and required number of iterations $1 + \frac{2^k}{-\log(\lambda)}$

Computational Complexity

The number of arithmetic operations needed to compute an optimal policy as a function λ, S, A, and B, where B denotes the number of bits required to encode each entry of the components of the MDP ($r(s,a)$, $p(j|s,a)$, λ)

- An algorithm for computing an optimal policy is **polynomial** if for all MDP instances, the required number of arithmetic operations for computing an optimal policy is bounded by a polynomial in S, A, and B.
- An algorithm for computing an optimal policy is **strongly polynomial** if for all MDP instances, the required number of arithmetic operations for computing an optimal policy is bounded by a polynomial in S and A.

Value Iteration

Assumptions: Rational transition probabilities and discount factor. Integer rewards. Encoding each of these values with $B \sim \log(\delta)$ bits (e.g. $\delta\lambda$, $\delta p(j|s,a)$ are integers, and $|r(s,a)| \leq \delta$)

Theorem. *The number of iterations n required to get an optimal policy under the VI algorithm, i.e.,*

$$\forall s,\ \pi_0^\star(s) \in \arg\max_{a \in A_s}(r(s,a) + \lambda \sum_j p(j|s,a)V_n(j))$$

satisfies:

$$n \leq \left((2S+3)B + S\log(S) + \log(\frac{1}{1-\lambda}) + 2\right)\frac{1}{-\log(\lambda)}$$

Howard's Policy Iteration

Theorem. *The number of iterations n required to get an optimal policy under Howard's PI satisfies:*

$$n \leq (A - S)\lceil \frac{1}{1-\lambda} \log(\frac{1}{1-\lambda}) \rceil = \mathcal{O}(\frac{A}{1-\lambda} \log(\frac{1}{1-\lambda}))$$

Proof. Assume that π_0 is not optimal. For all n, such that $n \geq \lceil \frac{1}{1-\lambda} \log(\frac{1}{1-\lambda}) \rceil$, one of the sub-optimal action of π_0 is eliminated in π_n.

Simplex-Policy Iteration

Theorem. *The number of iterations n required to get an optimal policy under Simplex-PI satisfies:*

$$n \leq S(A-S)\left(1 + \frac{2}{1-\lambda}\log(\frac{1}{1-\lambda})\right) = \mathcal{O}(\frac{AS}{1-\lambda}\log(\frac{1}{1-\lambda}))$$

Theorem. *For determinstic MDPs, the Simplex-PI terminates in $\mathcal{O}(S^3 A^2 \log^2(S))$ iterations.*

Policy Iteration

- Each iteration of the PI algorithm requires a polynomial number of operations, i.e., $\mathcal{O}(S^\omega)$
- For fixed λ, the Howard's and Simplex PI algorithms are strongly polynomial
- The best known λ-independent upper bound on the number of required operations for Howard's PI is $\Theta(A_{\max}^S/S)$ where $A_{\max} = \max_s A_s$ (not very far from enumerating all possible policies!)

Summary

Optimal control of systems with known dynamics and rewards

- MDPs: a generic model for controlled Markovian systems
  An MDP is defined through:

$$\{T, S, (A_s, p_t(\cdot|s,a), r_t(s,a), 1 \le t \le T, s \in S, a \in A_s)\}$$

- Finite-time horizon MDPs
  - Policy evaluation: computing the average reward of a policy $\pi = (\pi_1, \ldots, \pi_T)$ starting at s can be done using DP: $u_T(s) = r_T(s, \pi_T(s))$, and for $t = T-1, \ldots, 1$,

$$u_{t-1}^{\pi}(s_{t-1}) = r_{t-1}(s_{t-1}, a) + \sum_{j \in S} p_{t-1}(j|s_{t-1}, a) u_t^{\pi}(j)$$

    We obtain: $V_T^{\pi}(s) = u_1^{\pi}(s)$

Summary

- Value function and optimal policy: $V_T^\star(s) = \sup_{\pi \in MD} V_T^\pi(s)$ obtained by solving Bellman's equations with DP:
  For all s_T, $u_T^\star(s_T) = \max_a r_T(s_T, a)$
  For all $t = T-1, T-2, \dots, 1$

$$u_t^\star(s_t) = \max_{a \in A_{s_t}} \underbrace{\left[r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^\star(s_t, a, j) \right]}_{Q_t(s_t,a) \text{ optimal reward from } t \text{ if } a \text{ selected}}$$

  An optimal policy π is obtained by selecting $\pi_t(s_t)$ at time t such that

$$Q_t(s_t, \pi_t(s_t)) = \max_{a \in A_{s_t}} Q_t(s_t, a)$$

Summary

- Discounted intinite-horizon MDPs
  - Policy evaluation: computing the average reward of a stationary policy $\pi = (\pi_1, \pi_1, \ldots)$ starting at s can be done solving the linear system:

$$\forall s, \quad V^\pi(s) = r(s, \pi_1(s)) + \lambda \sum_j p(j|s, \pi_1(s))V^\pi(j)$$

  - Value function and optimal policy: $V^\star(s) = \sup_{\pi \in MD} V^\pi(s)$ obtained by solving Bellman's equations through VI or PI algorithm:

$$\forall s, \quad V^\star(s) = \max_{a \in A_s} \underbrace{\left[r(s,a) + \lambda \sum_{j \in S} p(j|s,a)V^\star(j) \right]}_{Q(s,a) \text{ optimal reward from state } s \text{ if } a \text{ selected}}$$

  An optimal policy π is stationary $\pi = (\pi_1, \pi_1, \ldots)$ where $\pi_1 \in MD_1$ is defined by: for any s,

$$\pi_1(s) = \arg\max_{a \in A_s} Q(s,a)$$

  Q is referred to as the Q-function.

References

Chapters 3 and 4 in Sutton-Barto's book.

Main reference on MDPs. All precise statements and proofs (and much more) can be found in:

- M. L. Puterman. "Markov Decision Processes: Discrete Stochastic Dynamic Programming", *Wiley*, 1994.

References

Complexity of solving MDPs

- C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of operations research*, 1987.
- M. Littman et al., "On the complexity of solving Markov decision problems," *Proc. of UAI*, 1995.
- P. Tseng, "Solving H-horizon, stationary Markov decision problems in time proportional to $\log(H)$," *Operations Research Letters*, 1990.
- Y. Ye, "The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate," *Mathematics of Operations Research*, 2011.
- B. Scherrer, "Improved and generalized upper bounds on the complexity of policy iteration," *Proc. of NIPS*, 2013.