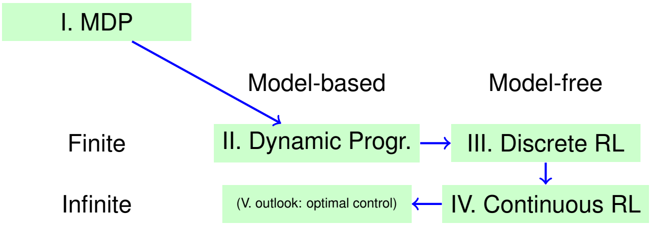
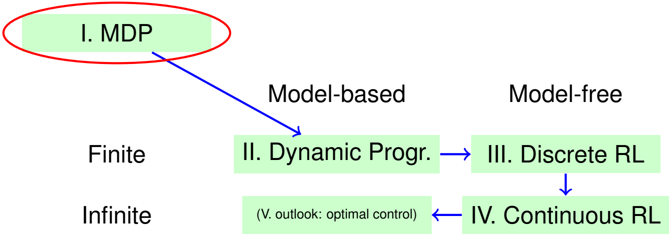
AIC/RL – Course Summary

Freek Stulp and Michèle Sebag

Université Paris-Saclay


I. MDP
Model-based
Model-free
Finite
II. Dynamic Progr.
III. Discrete RL
Infinite
(V. outlook: optimal control)
IV. Continuous RL



Problem definition

$$\text{MDP} = \{S, A, \mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}\} \tag{1}$$

$$\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \quad \text{Transition function} \tag{2}$$

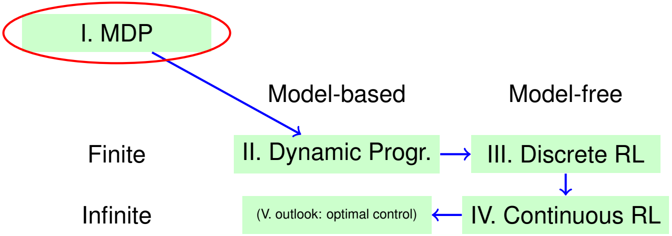
$$\mathcal{R}^a_{ss'} = \mathrm{E}\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \quad \text{Reward function} \tag{3}$$

$$\pi(s, a) = Pr\{a_t = a | s_t = s\} \quad \text{Policy} \tag{4}$$

Aim of the agent

$$\pi^* = argmax_\pi \, \mathrm{E}\{R|\pi\} \quad \text{Optimal policy} \tag{5}$$

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \quad \text{Return (discounted)} \tag{6}$$



Values

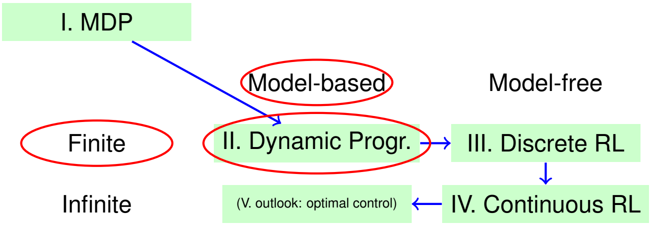
- $V^\pi(s)$ is expected return when starting in s and following π

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \qquad \text{Return (discounted)} \tag{1}$$

$$V^\pi(s) = \mathrm{E}_\pi \left\{ R_t | s_t = s \right\} \qquad \text{Value (SuBa3.7)} \tag{2}$$

$$= \mathrm{E}_\pi \left\{ \sum_{k=0}^{T} \gamma^k r_{t+k+1} | s_t = s \right\} \tag{3}$$

- Return R_t is an <u>actual observation</u>, $V^\pi(s)$ is an <u>expectation</u>.
- Value $V^\pi(s)$ depends on future actions, i.e. which the policy will decide.



Recursive Bellman Equation

- Values are recursively defined in terms of other values!

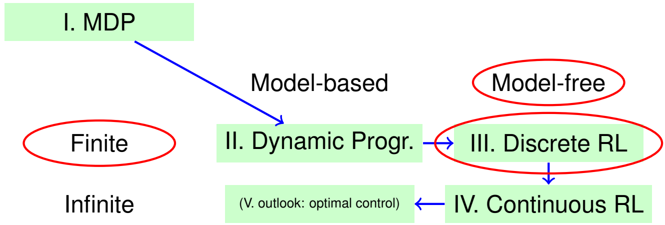
$$V^{\pi}(s) = \mathrm{E}_{\pi}\left\{R_t | s_t = s\right\} \tag{1}$$

$$= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right] \tag{2}$$

Dynamic Programming

- Policy Evaluation: $V_{k+1}(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V_k(s')\right]$
- Policy Improvement: $\pi'(s) = \mathrm{argmax}_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
- Value Iteration: Policy Evaluation + Policy Improvement

DP requires knowledge about $\mathcal{P}^a_{ss'}$ and $\mathcal{R}^a_{ss'}$: model-based!



Without a model?

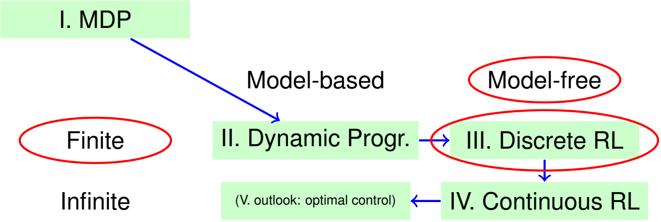
- Use actual observations to estimate values $V^\pi(s)$

Monte-Carlo methods

- Wait until end of the episode to update estimates
- Batch method: average of list of returns $V^\pi(s) = mean(Returns(s))$
- Incremental method: $V^\pi(s) = V^\pi(s) + \alpha\,[R - V^\pi(s)]$

Temporal Difference learning

- Update estimates after each immediate reward
- $TD(0)$: $V^\pi(s_t) = V^\pi(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$

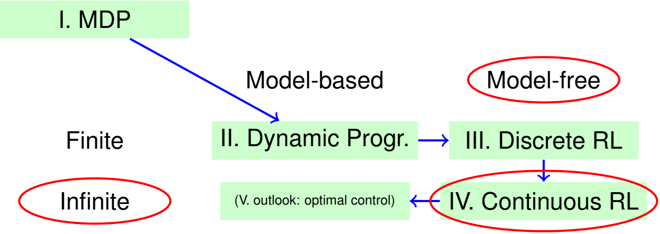


State values $V^\pi(s)$ vs. state/action values $Q^\pi(s,a)$

- Policy improvement: $\operatorname{argmax}_a \sum_{s'} \mathcal{P}^a_{ss'} V_k(s')$
  - Doesn't work in model-free case: we do not know $\mathcal{P}^a_{ss'}$
- Solution, use state/action values $Q^\pi(s,a)$
  - $Q^\pi(s,a) = \mathrm{E}_\pi \{R_t | s_t = s, a_t = a\}$
  - Policy improvement simply becomes: $\operatorname{argmax}_a Q(s,a)$
- How to estimate $Q^\pi(s,a)$ from observations?
  - MC and TD udpate rules for $Q^\pi(s,a)$ essentially same as for $V^\pi(s)$

Exploration/exploitation trade-off

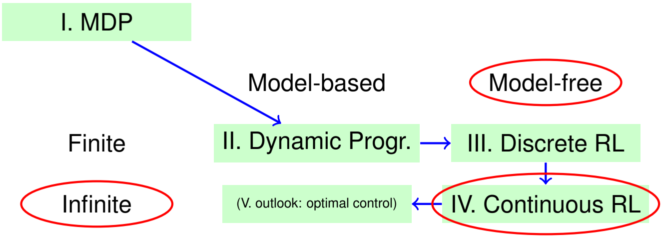
- Only exploration or exploitation would be bad strategy
- Explore first, exploit later: ϵ-greedy exploration, with decaying ϵ



Continuous, infinite MDPs

- S State space $S \subseteq \mathbb{R}^{D_S}$ (D_S-dimensional vector)
- A Action space $A \subseteq \mathbb{R}^{D_A}$ (D_A-dimensional vector)
- f Transition rate function $f : S \times A \to \Delta S$
- r Reward function $r : S \times A \to \mathbb{R}$

- Bad news: infinite number of states and actions...
- Good news: smoothness, i.e. ΔS usually not so big



Function Approximation

- $V_\theta(s) = f_\theta(s)$ estimate V with parameterized function
  - radial basis function network, neural network, decision tree
- $\theta_{t+1} = \theta_t + \frac{1}{2}\alpha\nabla_{\theta_t}\left[V^\pi(s_t) - V_t^\pi(s_t)\right]^2$ gradient-descent

Direct policy search

- Value function not explicitly represented (!)
- Define parameterized policy $\pi_\theta(s)$
- Search directly in space of θ using optimization
  - gradient based, evolution strategies

Case Study 1: Deep Reinforcement Learning

- Use a deep neural network to represent the function that approximates $Q^{\pi}(s, a)$
- Provide raw images as input the the function approximator
- Same algorithm (and its parameters) applied to many different Atari games

References

- Playing Atari with Deep Reinforcement Learning

  http://arxiv.org/abs/1312.5602
- Human-level control through deep reinforcement learning. Nature, 2015.

  http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

Case Study 2: Learning to Manipulate

- Use direct policy search for continuous high-dimensional action spaces
- Applied to real robotic manipulation problems
- Such problems are very difficult to model

References

- Learning Motion Primitive Goals for Robust Manipulation
  http://freekstulp.net/publications/b2hd-stulp11learningmotion.html
- Reinforcement Learning with Sequences of Motion Primitives for Robust Manipulation
  http://freekstulp.net/publications/b2hd-stulp12reinforcement.html