Regular Decision Processes

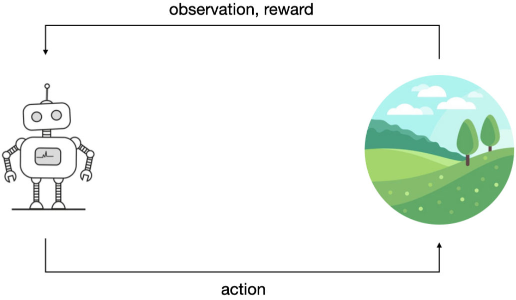
Alessandro Ronca

Sapienza University of Rome

Motivation: Artificial Intelligence

- Optimal agents
  - “Acting rationally” in Russell and Norvig’s AI book
  - Optimal agent for the Turing test

An agent and the environment



A Markov agent and the environment

- Is the state of affairs readily available?
- If not, who is computing states for the agent?
- It is intelligence given an oracle to compute such states
    - Computing states could be the most challenging part

Outline

1. Non-Markov Decision Processes (NMDPs)
2. Regular Decision Processes (RDPs)
3. LDL_f specification of RDPs
4. Learning RDPs via Probabilistic Automata
5. References

Outline

1. Non-Markov Decision Processes (NMDPs)
2. Regular Decision Processes (RDPs)
3. LDL$_f$ specification of RDPs
4. Learning RDPs via Probabilistic Automata
5. References

Motivation

Optimal agents in stochastic domains.

Non-Markov Decision Processes (NMDPs)

$$\mathcal{P} = \langle A, O, R, \mathbf{T}, \mathbf{R}, \gamma \rangle$$

- Actions $A = \{a_1, a_2, \ldots, a_n\}$
- Observations $O = \{o_1, o_2, \ldots, o_m\}$
  - An element h of O^* is called a history
- Rewards $R = \{r_1, r_2, \ldots, r_k\} \subseteq \mathbb{R}$
  - An element t of $(AOR)^*$ is called a trace
- Transition function $\mathbf{T} : O^* \times A \times O \to [0, 1]$
  - $\mathbf{T}(\cdot|h, a)$ is a probability distribution on O
- Reward function $\mathbf{R} : O^* \times A \times R \to [0, 1]$
  - $\mathbf{R}(\cdot|h, a)$ is a probability distribution on R
- $\gamma \in (0, 1)$ is the discount factor (can be replaced with a finite horizon)

Dynamics function

- The transition and reward functions can be combined into the dynamics function $\mathbf{D} : O^* \times A \times O \times R \to [0, 1]$
  - Definition: $\mathbf{D}(o, r|h, a) = \mathbf{T}(o|h, a) \cdot \mathbf{R}(r|h, a)$
  - Note that $\mathbf{D}(\cdot|h, a)$ is a probability distribution on $O \times R$

Policy functions

- A policy is a function $\pi : O^* \times A \to [0, 1]$
  - with $\pi(\cdot|h)$ a probability distribution on A
  - It is deterministic if, on every history h, it assigns probability one to an action a (we can write $\pi(h) = a$)

Value functions

$$\mathbf{v}_\pi(h) = \sum_{aor} \pi(a|h) \cdot \mathbf{D}(o, r|h, a) \cdot (r + \gamma \cdot \mathbf{v}_\pi(ho))$$

$$\mathbf{v}_*(h) = \max_\pi \mathbf{v}_\pi(h) = \max_a \left(\sum_{or} \mathbf{D}(o, r|h, a) \cdot (r + \gamma \cdot \mathbf{v}_*(ho)) \right)$$

$$\mathbf{q}_\pi(h, a) = \sum_{or} \mathbf{D}(o, r|h, a) \cdot (r + \gamma \cdot \mathbf{v}_\pi(ho))$$

$$\mathbf{q}_*(h, a) = \max_\pi \mathbf{q}_\pi(h, a) = \sum_{or} \mathbf{D}(o, r|h, a) \cdot (r + \gamma \cdot \mathbf{v}_*(ho))$$

Optimality and Near-optimality

- A policy π is optimal if $\mathbf{v}_\pi(h) = \mathbf{v}_*(h)$ on every history h.
- A policy π is ϵ-optimal if $|\mathbf{v}_\pi(h) - \mathbf{v}_*(h)| \leq \epsilon$ on every history h

- Optimal agent $\rightsquigarrow$ an agent that follows an optimal policy
- Near-optimal agent $\rightsquigarrow$ an agent that follows an ϵ-optimal policy for some ϵ

Comments on NMDPs

- They are very general
  - maybe too general...
- The transition and reward functions are not even required to be computable!
- Optimal policies may not be computable
  - We could require from an agent to be able to solve the halting problem (prove it!)
- There is no hope for computationally-feasible approaches for NMDPs

Markov Decision Processes (MDPs)

$$\mathcal{M} = \langle A, S, R, \mathbf{T}, \mathbf{R}, \gamma \rangle$$

- All history-dependent functions depend on the last observation only
  - Since the last observation captures the current state of affairs, it is referred to as a state s, coming from a finite set S (which takes the place of the set of observations).
- The functions become as follows:
  - transition function $\mathbf{T}(s'|s, a)$
  - reward function $\mathbf{R}(r|s, a)$
  - dynamics function $\mathbf{D}(s', r|s, a)$
  - policy function $\pi(a|s)$
  - value functions $\mathbf{v}_\pi(s)$, $\mathbf{q}_\pi(s)$, $\mathbf{v}_*(s)$.

Outline

1. Non-Markov Decision Processes (NMDPs)
2. Regular Decision Processes (RDPs)
3. LDL_f specification of RDPs
4. Learning RDPs via Probabilistic Automata
5. References

Motivation

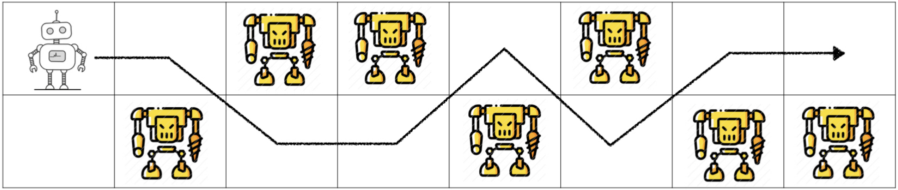
The motivation of NMDPs + Favourable computational properties

Regular Decision Processes (RDPs)

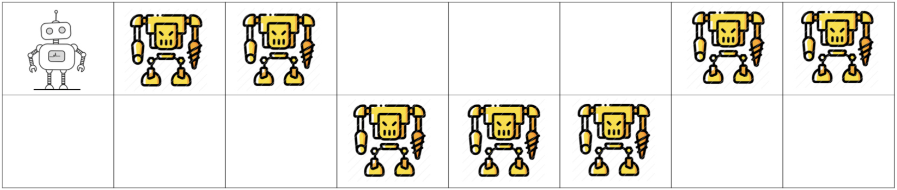
$$\mathcal{R} = \langle A, O, R, \mathbf{T}, \mathbf{R}, \gamma \rangle$$

- $\mathbf{T}$ and $\mathbf{R}$ are 'regular' functions of the history
- They can be characterised/specified in a number of formalisms:
  1. finite-state transducers,
  2. LDL$_f$ formulas,
  3. probabilistic automata (suitable for learning),
  4. other formalisms (e.g., regular expressions)

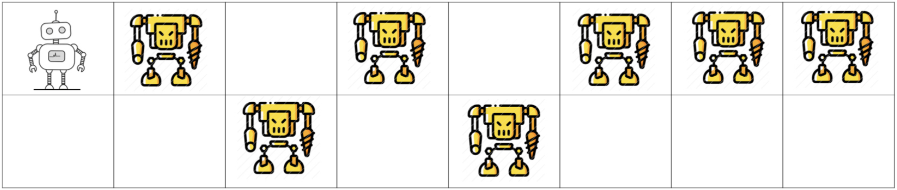
Example



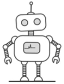
Example



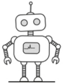
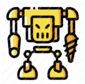
Example



Example

	???	???	???	???	???	???	???
	???	???	???	???	???	???	???

Example

		???	???	???	???	???	???
		???	???	???	???	???	???

Example

Finite Transducers (Moore's variant)

$$T = \langle S, s_0, \Sigma, \tau, \Gamma, \theta \rangle$$

- S is the finite set of states;
- $s_0 \in S$ is the initial state;
- Σ is the finite input alphabet;
- $\tau : S \times \Sigma \to S$ is the deterministic transition function;
- Γ is the finite output alphabet;
- $\theta : S \to \Gamma$ is the output function.

Finite Transducers (Moore's variant) (2)

$$T = \langle S, s_0, \Sigma, \tau, \Gamma, \theta \rangle$$

- The transition function τ and output function θ are extended to strings as follows:

$$\tau(s, \sigma_1\sigma_2 \ldots \sigma_n) = \tau(\tau(s, \sigma_1), \sigma_2 \ldots \sigma_n) \text{ with } \tau(s, \varepsilon) = s$$

$$\theta(s, \sigma_1 \ldots \sigma_n) = \theta(s)\, \theta(\tau(s, \sigma_1), \sigma_2 \ldots \sigma_n) \text{ with } \theta(s, \varepsilon) = \theta(s)$$

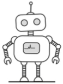
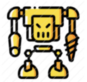
- A transducer T can be seen as a device that maps (or transduces) every string $w \in \Sigma^*$ to the string $\theta(s_0, w)$
- A transducer T can be seen as a representation of the function $T : \Sigma^* \to \Gamma^*$ such that $T(w) = \theta(s_0, w)$

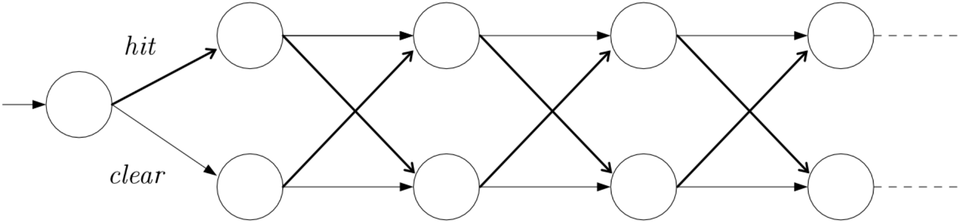
Transducer characterisation of RDPs

$$\mathcal{R} = \langle A, O, R, \mathbf{T}, \mathbf{R}, \gamma \rangle$$

- $\mathcal{R}$ is an RDP if there exist two transducers $T_{\mathbf{T}}$ and $T_{\mathbf{R}}$ such that:
  1. transducer $T_{\mathbf{T}}$ maps every history h to the function $\mathbf{T}_h : A \times O \to [0,1]$ induced by $\mathbf{T}$ when its first argument is h, i.e.,
     - $\mathbf{T}_h(o|a) = \mathbf{T}(o|h,a) \quad \forall a \in A, \forall o \in O.$
  2. transducer $T_{\mathbf{R}}$ maps every history h to the function $\mathbf{R}_h : A \times R \to [0,1]$ induced by $\mathbf{R}$ when its first argument is h, i.e.,
     - $\mathbf{R}_h(r|a) = \mathbf{R}(r|h,a) \quad \forall a \in A, \forall r \in R.$
- Note that $T_{\mathbf{T}}$ and $T_{\mathbf{R}}$ can be combined into a transducer $T_{\mathbf{D}}$ for the dynamics function $\mathbf{D}$.

Example: Transitions

		???	???	???	???	???	???
		???	???	???	???	???	???



Example: Full specification

- $\mathcal{R} = \langle A, O, R, \mathbf{T}, \mathbf{R}, \gamma \rangle$
  - $A = \{a_1, a_2\}$
  - $O = [0, m-1] \times \{hit, clear\}$
  - p_i^j is the j-th probability of enemy i of being in the upper cell
- $T = \langle S, s_0, O, \tau, \Gamma, \theta \rangle$
  - $S = [0, m-1] \times \{0, 1\}$
  - $s_0 = \langle m-1, 0 \rangle$,
  - the transition function is:
    - $\tau(\langle i, b \rangle, \langle i+1 \bmod m, hit \rangle) = \langle i+1 \bmod m, b+1 \bmod 2 \rangle$,
    - $\tau(\langle i, b \rangle, \langle i+1 \bmod m, clear \rangle) = \langle i+1 \bmod m, b \rangle$,
  - the output function is:
    - $\theta(\langle i, b \rangle)(a_k, \langle j, enemy \rangle, 0) = p_j^b$ where $j = i+1 \bmod m$,
    - $\theta(\langle i, b \rangle)(a_k, \langle j, clear \rangle, 1) = 1 - p_j^b$ where $j = i+1 \bmod m$.

Equivalent MDP

Consider an RDP $\mathcal{R} = \langle A, O, R, \mathbf{D}^{\mathcal{R}}, \gamma \rangle$

Consider a transducer $T = \langle S, s_0, O, \tau, \Gamma, \theta \rangle$ for the dynamics of $\mathcal{R}$.
Namely, $T(h) = \mathbf{D}_h$ and $\mathbf{D}_h(o, r|a) = \mathbf{D}(o, r|h, a)$,
or simply $T(h)(a, o, r) = \mathbf{D}(o, r|h, a)$.

We define the equivalent MDP $\mathcal{M} = \langle A, S, R, \mathbf{D}^{\mathcal{M}}, \gamma \rangle$ where the dynamics function is defined as follows:

$$\mathbf{D}^{\mathcal{M}}(s_2, r|s_1, a) = \sum_{o:\tau(s_1,o)=s_2} \theta(s_1)(a, o, r).$$

It is the MDP perceived by an agent that interacts with $\mathcal{R}$ but reads state $\tau(h)$ instead of history h.

Equivalent MDP: Properties

Consider a history h generated by an agent interacting with $\mathcal{R}$. Say that the agent reads state $s = \tau(h)$ instead of history h, and acts following a Markov policy π on S, hence picks actions according to $\pi(\cdot|s)$. Thus, the agent acts according the the composition $\pi\tau$ of π with the transition function τ of the transducer.

Theorem

For every history h, $\mathbf{v}_{\pi}^{\mathcal{M}}(\tau(h)) = \mathbf{v}_{\pi\tau}^{\mathcal{R}}(h)$.

Therefore, among all policies π on S, it is best to choose a policy π_* that has maximum value in $\mathcal{M}$, since $\mathbf{v}_{\pi_*}^{\mathcal{M}}(\tau(h)) = \mathbf{v}_{\pi_*\tau}^{\mathcal{R}}(h)$.

Is $\pi_*\tau$ optimal for $\mathcal{R}$? Or there are better policies that cannot be expressed as the composition of a Markov policy with the transition function?

Equivalent MDP: Properties and Usage

Theorem

Every RDP $\mathcal{R}$ admits an optimal policy of the form $\pi\tau$ for τ the transition function of its dynamics transducer and π a Markov policy on the states of the transducer.

Corollary

If π_* is an optimal policy for the equivalent MDP, then $\pi_*\tau$ is optimal for the original RDP.

Application: solve the equivalent MDP $\mathcal{M}$ to obtain an optimal Markov policy π_*, and compose it with τ, to obtain an optimal policy $\pi_*\tau$ for $\mathcal{R}$.

Remark: all claims above about optimality hold for ϵ-optimality as well.

Equivalent MDP: Implications

- Planning, i.e., computing an optimal policy:
  - Given the dynamics transducer as input, computing an optimal policy for an RDP amount to computing an optimal policy for an MDP (value iteration, etc.)

Equivalent MDP: Implications (2)

- Reinforcement Learning:
  - A Markov reinforcement learning agent can achieve optimality in an RDP if it is provided with the transducer states in place of observations.
    - This requires an external helper, who has knowledge of the transducer.
    - It is what has been done (more or less explictly) in reinforcement learning so far, since classic algorithms cannot operate directly on observations.
  - Fully-fledged reinforcement learning in RDPs requires to learn the transducer's transition function τ as part of the overall learning process
    - Challenges: to learn a good transition function you need good data, but to have good data you need a good transition function
    - The claim holds regardless of the specific approach.

The importance of τ

- The transition function τ allows an agent to establish that two different histories can be considered equivalent for the purpose of predicting their respective futures.
- Thus, policies can be based on the states returned by τ, since there is no reason to behave differently on two histories h_1, h_2 when $\tau(h_1) = \tau(h_2) = s$.
- For an agent, it is sufficient to see the world through τ.

Outline

1. Non-Markov Decision Processes (NMDPs)
2. Regular Decision Processes (RDPs)
3. LDL_f specification of RDPs
4. Learning RDPs via Probabilistic Automata
5. References

Motivation

- High-level specification of RDPs
  - How does a human (e.g., engineer) write down an RDP?
    - Writing the transducer may be inconvenient
  - It could be a convenient language for agents to exchange information about the domain

LDL_f specification of RDPs (syntax)

- Set of propositions (fluents) $P = \{p_1, \dots, p_n\}$.
- Triples specifying transition function:

  $\{\langle \varphi, a, \delta \rangle \mid \varphi$ is an LDL_f formula on P, $a \in A$, δ is a distribution on $O\}$

- Triples specifying reward function:

  $\{\langle \psi, a, \rho \rangle \mid \psi$ is an LDL_f formula on P, $a \in A$, ρ is a distribution on $R\}$

LDL$_f$ specification of RDPs (semantics)

$$\mathcal{R} = \langle A, O, R, \mathbf{T}, \mathbf{R}, \gamma \rangle$$

- $O = 2^P$
- Note that histories O^* are LDL$_f$ interpretations.
- Transition function:
  1. (mutual exclusivity) for every history h and every action a there do not exist two distinct triples $\langle \varphi, a, \delta \rangle$ and $\langle \varphi', a, \delta' \rangle$ such that $h \models \varphi$ and $h \models \varphi'$;
  2. $\mathbf{T}(\cdot|h, a) = \delta(\cdot)$ for some triple $\langle \varphi, a, \delta \rangle$ with $h \models \varphi$.
- Reward function:
  1. (mutual exclusivity) for every history h and every action a there do not exist two distinct triples $\langle \psi, a, \rho \rangle$ and $\langle \psi', a, \rho' \rangle$ such that $h \models \psi$ and $h \models \psi'$;
  2. $\mathbf{R}(\cdot|h, a) = \rho(\cdot)$ for some triple $\langle \psi, a, \rho \rangle$ with $h \models \psi$.

From LDL$_f$ to transducer

- We can compute the transducer version of the specified RDP.
- The transition transducer $T_{\mathbf{T}} = \langle S, s_0, \Sigma, \tau, \Gamma, \theta \rangle$ is built as follows:
  - Consider the formulas $\varphi_1, \ldots, \varphi_n$ for the transition specifications
  - Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be the automata for $\varphi_1, \ldots, \varphi_n$
  - Let $\mathcal{A}$ be the cross product of such automata
  - S, s_0, Σ, τ are the ones of $\mathcal{A}$
  - For each state $s = \langle s_1, \ldots, s_n \rangle$ of $\mathcal{A}$, there is at most one s_i that is a final state in $\mathcal{A}_i$, due to mutual exclusivity.
    - The output $\theta(s)$ of state s is the function that maps each action a to the distribution δ as specified by some triple $\langle \varphi_i, a, \delta \rangle$.
- The reward transducer $T_{\mathbf{R}}$ is built similarly.

Outline

1 Non-Markov Decision Processes (NMDPs)

2 Regular Decision Processes (RDPs)

3 LDL$_f$ specification of RDPs

4 Learning RDPs via Probabilistic Automata

5 References

Motivation

- Recall the importance of the transition function τ
- Learning τ means learning the states S and how to transitions among them
- The semantics of states is probabilistic: a state means the probability it determines over its possible futures
- Learning τ as the transition function of a Probabilistic-Deterministic Finite Automaton (PDFA)

Probabilistic-Deterministic Finite Automata (PDFA)

$$\mathcal{A} = \langle S, \Sigma, \tau, \lambda, \zeta, s_0 \rangle$$

- S is a finite set of states;
- Σ is a finite input alphabet;
- $\tau : S \times \Sigma \to S$ is the deterministic transition function;
- $\lambda : S \times (\Sigma \cup \{\zeta\}) \to [0, 1]$ is the probability of emitting the next symbol or stopping;

Extension of λ to strings:

$$\lambda(s, \sigma_1\sigma_2 \dots \sigma_n) = \lambda(s, \sigma_1) \cdot \lambda(\tau(s, \sigma_1), \sigma_2 \dots \sigma_n) \quad \text{with} \quad \lambda(s, \varepsilon) = 1$$

Automaton $\mathcal{A}$ represents the following probability distribution on Σ^*:

$$\mathcal{A}(w) = \lambda(s_0, w\zeta)$$

RDPs seen as PDFA

$$\mathcal{R} = \langle A, O, R, \mathbf{T}, \mathbf{R}, \gamma \rangle$$
$$T = \langle S, s_0, O, \tau, \Gamma, \theta \rangle$$
$$\mathcal{A} = \langle S, \Sigma, \tau', \lambda, \zeta, s_0 \rangle$$

- Agent explores stopping with probability $p > 0$ and choosing an action uniformly at random otherwise.
- alphabet $\Sigma = AOR$,
- transitions $\tau'(s, aor) = \tau(s, o)$,
- probability function:
  - $\lambda(s, aor) = \frac{1-p}{|A|} \cdot \theta(s)(a, o, r)$,
  - $\lambda(s, \zeta) = p$.

Learning RDPs

- PDFA can be learned
- Thus, we can learn the PDFA encoding the RDP and extract the transition function (which is what we are interested in)
- In particular, the transducer's transition function τ can be obtained from the PDFA's transition function τ' by dropping actions and rewards, which have no effect on transitions.
- Application: reinforcement learning

Outline

1 Non-Markov Decision Processes (NMDPs)

2 Regular Decision Processes (RDPs)

3 LDL$_f$ specification of RDPs

4 Learning RDPs via Probabilistic Automata

5 References

References

Ronen I. Brafman,Giuseppe De Giacomo.

Regular Decision Processes: A Model for Non-Markovian Domains.

IJCAI 2019.

Eden Abadi,Ronen I. Brafman

Learning and Solving Regular Decision Processes.

IJCAI 2020.

Alessandro Ronca, Giuseppe De Giacomo.

Efficient PAC Reinforcement Learning in Regular Decision Processes.

IJCAI 2021 (to appear, write to me for a copy, ronca@diag.uniroma1.it).