RL 6: Markovian Decision Processes

Michael Herrmann

University of Edinburgh, School of Informatics

30/01/2015

Last time: Reinforcement Learning Algorithms

Determine (and maximise) expected (discounted|relative) reward

R-learning (off-policy): $a_t = \arg\max_a \mathcal{Q}(s_t, a)$ (plus exploration)

$$\mathcal{Q}_{t+1}(s_t, a_t) = (1-\eta)\,\mathcal{Q}_t(s_t, a_t) + \eta\Big(r_{t+1} - \rho_t + \max_a \mathcal{Q}_t(s_{t+1}, a)\Big)$$

$$\rho_{t+1} = (1-\alpha)\,\rho_t + \alpha\Big(r_{t+1} + \max_a \mathcal{Q}_t(s_{t+1}, a_{t+1}) - \max_a \mathcal{Q}_{t+1}(s_t, a)\Big)$$

$\mathcal{Q}$-learning (off-policy): $a_t = \arg\max_a \mathcal{Q}(s_t, a)$ (plus exploration)

$$\mathcal{Q}_{t+1}(s_t, a_t) = (1-\eta)\,\mathcal{Q}_t(s_t, a_t) + \eta\Big(r_{t+1} + \gamma \max_a \mathcal{Q}_t(s_{t+1}, a)\Big)$$

SARSA (on-policy): $a \sim \pi(s_t, a_t)$

$$\mathcal{Q}_{t+1}(s_t, a_t) = (1-\eta)\,\mathcal{Q}_t(s_t, a_t) + \eta\left(r_{t+1} + \gamma \mathcal{Q}_t(s_{t+1}, a_{t+1})\right)$$

TD(0) (on-policy): (value estimation)

$$\hat{V}_{t+1}(s_t) = \hat{V}_t(s_t) + \eta\delta_{t+1} = (1-\eta)\,\hat{V}_t(s_t) + \eta\Big(r_t + \gamma \hat{V}_t(s_{t+1})\Big)$$

MDPs: Main points

- Markov Chains will be used as model of the state dynamics in a RL problem
- The Markov property implies that predictions for the future based on the present state are just as good as predictions based on the full history
- Not all state dynamices are Markov (why not?), but often Markovianity is a reasonable approximation
- If the intrinsic state dynamics is Markov, Markovianity is conserved if actions follow a state-dependent policy
- If for a fixed policy the state dynamics is a Markov chain, then for a different fixed policy it is a (generally) different Markov chain
- On-policy reinforcement learning algorithms are often based MDPs in a strict sense ($\mathcal{Q}$-learning is off-policy and therefore not a very good example)

Stochastic Processes

- A stochastic process is an indexed collection of random variables $\{X_t\}$
  - e.g. time series of weekly demands for a product
- Discrete case: At a particular time t, labelled by integers, system is found in exactly one of a finite number of mutually exclusive and exhaustive categories or states, labelled by integers, too
- Process could be *embedded*, i.e. time points correspond to occurrence of specific events
- Random variables may depend on others, e.g.,

$$X_{t+1} = \begin{cases} Y_{t+1} & \text{if } X_t < 0 \\ X_t & \text{if } X_t \geq 0 \end{cases}$$

or $$X_{t+1} = \sum_{k=0}^{K} \alpha_k X_{t-k} + \xi_t \text{ with } \xi_t \sim \mathcal{N}\left(\mu, \sigma^2\right)$$

An Example of a Markov Chain Model

- Consider the following application: machine maintenance
- A factory has a machine that deteriorates rapidly in quality and output and is inspected periodically, e.g., monthly
- Inspection declares the machine to be in four possible states:
  - 0: Good as new
  - 1: Operable, minor deterioration
  - 2: Operable, major deterioration
  - 3: Inoperable
- Let X_t denote this observed state
  - evolves according to some "law of motion", so it is a stochastic process
  - Furthermore, assume it is a finite state Markov chain

Towards an Markov Decision Model

- Transition matrix is based on the following:

states	0	1	2	3
0	0	7/8	1/16	1/16
1	0	3/4	1/8	1/8
2	0	0	1/2	1/2
3	0	0	0	1

- Once the machine goes inoperable, it stays there
  - If no repairs, eventually, it reaches this state which is absorbing!
- Repair is an *action* –- a very simple maintenance *policy*
  - e.g., move machine from state 3 to state 0
  - This changes the transition probabilities and leaves us with a different Markov chain (see below)

Let's first look at Markov Chains

The stochastic process is said to have a Markovian property if

$$P(X_{t+1}=j|X_t=i, X_{t-1}=k_{t-1}, \ldots, X_1=k_1, X_0=k_0)=P(X_{t+1}=j|X_t=i)$$

for $t = 0, 1, \ldots$ and every sequence $i,\ j,\ k_0, \ldots, k_{t-1}$

Markovian probability means that the conditional probability of a future event given any past events and current state, is independent of past states and depends only on present

The conditional probabilities are transition probabilities,

$$P\left(X_{t+1} = j|X_t = i\right)$$

These are *stationary* if time invariant. Then we can write

$$p_{ij} = P\left(X_{t+1} = j|X_t = i\right) = P\left(X_1 = j|X_0 = i\right)$$

Markov Chains

- A stochastic process is a finite-state Markov chain if it has
  - a finite number of states $s \in \mathcal{S}$
  - the Markovian property
  - stationary transition probabilities p_{ij} for all i, j
  - a set of initial probabilities $\pi_i^0 = P\{X_0 = i\}$ for all i
- n-step transition probabilities (looking forward in time)

$$p_{ij}^{(n)} = P(X_{t+n} = j | X_t = i) = P(X_n = j | X_0 = i)$$

- One can write a transition matrix

$$\mathsf{P}^{(n)} = \begin{pmatrix} p_{00}^{(n)} & \cdots & p_{0M}^{(n)} \\ \vdots & \ddots & \vdots \\ p_{M0}^{(n)} & \cdots & p_{MM}^{(n)} \end{pmatrix}$$

Andrey Markov

Markov Chains

- 2-step transition probabilities can be obtained from 1-step *transition probabilities*

$$p_{ij}^{(2)} = \sum_{k=1}^{M} p_{ik} p_{kj}, \ \forall i, j$$

- n-step transition probabilities can be obtained from 1-step *transition probabilities* recursively (Chapman-Kolmogorov)*

$$p_{ij}^{(n)} = \sum_{k=1}^{M} p_{ik}^{(v)} p_{kj}^{(n-v)}, \ \forall i, j, n; \ 0 \leq v \leq n$$

- We can get this via the matrix, too

$$P^{(n)} = \underbrace{P \cdots P}_{n \text{ times}} = P^n = P^{n-v} P^v, \quad 0 \leq v \leq n$$

---

* For consistency we define $p^{(0)} = I$ (unit matrix).

Markov Chains: First Passage Times

- Number of transitions to go from i to j for the first time
  - First Passage Times are random variables $\Rightarrow$ mean FPT etc.
  - If $i = j$, this is the *recurrence time*
- n-step recursive relationship for first passage probability

$$
\begin{aligned}
f_{ij}^{(1)} &= p_{ij}^{(1)} = p_{ij} \\
f_{ij}^{(2)} &= p_{ij}^{(2)} - f_{ij}^{(1)} p_{jj} \\
&\vdots \\
f_{ij}^{(n)} &= p_{ij}^{(n)} - f_{ij}^{(1)} p_{jj}^{(n-1)} - f_{ij}^{(2)} p_{jj}^{(n-2)} - \cdots - f_{ij}^{(n-1)} p_{jj}
\end{aligned}
$$

- For fixed i and j, $f_{ij}^{(n)} \geq 0$ and it holds that $\sum_{n=1}^{\infty} f_{ij}^{(n)} \leq 1$
- $\sum_{n=1}^{\infty} f_{ii}^{(n)} = 1$ implies a *recurrent* state; *absorbing* if $f_{ii}^{(1)} = 1$
- mean FPT: $\sum_{n=1}^{\infty} n\, f_{ij}^{(n)}$
- Positive recurrence: State is recurrent and has a mFPT$< \infty$

Markov Chains: Classification of States

- State j is *accessible* from i if $p_{ij}^{(n)} > 0$ (for some n)
- If state j is accessible from i and vice versa, the two states are said to *communicate*
- As a result of communication, one may partition the general Markov chain into states in disjoint classes
  - MC is *irreducible* if there is only one class
- If the MC can only visit the state at integer multiples of t, we call it *periodic*
- Positive recurrent states that are aperiodic are called *ergodic* states

Markov Chains: Long-Run Properties

Remember the inventory example? Interestingly, probability of being in state j (after, e.g., 8 weeks) appears to be independent of the *initial* level π^0 of inventory.

For an irreducible ergodic Markov chain, one has limiting probability

$$\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j^*$$

i.e. the limit for each element p_{ij} does not depend on i.

$$\pi_j^* = \sum_{i=1}^{M} \pi_i^* p_{ij} \;\forall j = 1, \ldots, M$$

$\pi^* = (\pi_1^*, \ldots, \pi_M^*)$ is an eigenvector of the matrix $P = (p_{ij})$.

Perron-Frobenius theorem: Matrices with positive entries have a unique largest eigenvalue. For a probability matrix this EV is 1.

Reciprocal of π_j^* gives the recurrence time m_{jj}

Markov Chains: Expected Average Cost

Sometimes aperiodic chain is a strong assumption. If we relax it, the limiting probability needs a slightly different definition:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} p_{ij}^{(n)} = \pi_j^*$$

Suppose you incur a (time-independent) cost $C(X_t)$, using above you can derive the long-run expected average over unit time as

$$\lim_{n\to\infty} \left\{ E\left[\frac{1}{n} \sum_{k=1}^{n} C\left(x_t\right) \right] \right\} = \sum_{j=1}^{M} C_j \pi_j^*$$

Can be more elaborate in general, depending on cost function

Markov Decision Model

- States in the machine maintenance example
  - 0: Good as new
  - 1: Operable, minor deterioration
  - 2: Operable, major deterioration
  - 3: Inoperable
- Transition matrix was the following:

states	0	1	2	3
0	0	7/8	1/16	1/16
1	0	3/4	1/8	1/8
2	0	0	1/2	1/2
3	0	0	0	1

Markov Decision Model

- There are costs as system evolves:
  - State 0: cost 0
  - State 1: cost 1000
  - State 2: cost 3000
- Replacement cost, taking state 3 to 0, is 4000 (and lost production of 2000), so cost = 6000
- The modified transition probabilities are:

states	0	1	2	3
0	0	7/8	1/16	1/16
1	0	3/4	1/8	1/8
2	0	0	1/2	1/2
3	1	0	0	0

Markov Decision Model

- What is the average cost of this maintenance *policy*?
- Compute the steady state probabilities:

$$\pi_0^* = \frac{2}{13},\ \pi_1^* = \frac{7}{13},\ \pi_2^* = \frac{2}{13},\ \pi_3^* = \frac{2}{13}$$

- (Long run) expected average cost per unit of time

$$0\pi_0^* + 1000\pi_1^* + 3000\pi_2^* + 6000\pi_3^* = \frac{25000}{13} = 1923$$

- Cost for the model without any actions? Could be 2000, but system is not ergodic.

Markov Decision Model

- Consider a slightly more elaborate policy:
  - Repair when inoperable or replace when needing major repairs
- Permit one more thing: overhaul
  - Go back to minor repairs state (1) for the next time step
  - Not possible if truly inoperable, but can go from major to minor
- Transition matrix now changes a little bit
- Key point about the system behaviour. It evolves according to
  - "Laws of motion"
  - Sequence of decisions made (actions from {1:none, 2:overhaul, 3:replace})
- Stochastic process is now defined in terms of X_t and Δ_t
  - Policy R is a rule for making decisions
  - Could use all history, although popular choice is (current) state-based

Markov Decision Model

- Many policies are possible how to use the actions (1: none, 2: overhaul, 3: replace) in the states 0, 1, 2, 3:

Policies	$d_R(0)$	$d_R(1)$	$d_R(2)$	$d_R(3)$
R_a	1	1	1	3
R_b	1	1	2	3
R_c	1	1	3	3
R_d	1	3	3	3

- Each policy defines a transition matrix, e.g., for R_b

states	0	1	2	3
0	0	7/8	1/16	1/16
1	0	3/4	1/8	1/8
2	0	1	0	0
3	1	0	0	0

- Which policy is best? Need costs. . . .

Markov Decision Model

- C_{ik}= expected cost incurred during next transition if system is in state i and decision k is made

state|decision	1	2	3
0	0	4000	6000
1	1000	4000	6000
2	3000	4000	6000
3	∞	∞	6000

- The long run average expected cost for each policy may be computed using

$$E(C) = \sum_{i=0}^{M} C_{ik}\pi_i^*$$

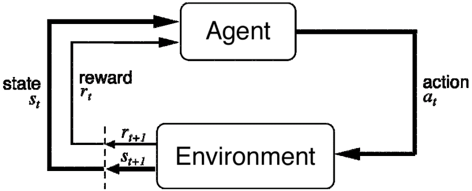
- R_b is best (homework: check!)

Extensions

- Heuristic Search
  - Dynamic Programming: AO*, LAO*, RTDP, ...
- Factored MDPs
  - add planning graph style heuristics
  - use goal regression to generalise better
- Hierarchical MDPs
  - hierarchy of sub-tasks and/or actions to scale better
- Reinforcement Learning
- Partially Observable Markov Decision Processes
  - noisy sensors, partially observable environment, popular in robotics

adapted from Mausam: Markov decision problems, Ch. 17

RL and Markov Decision Processes ... as used in Sutton & Barto



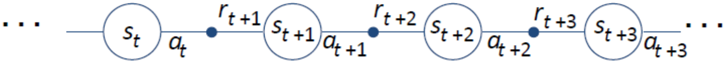
Agent and environment interact at discrete time steps: $t = 0, 1, 2,$

Agent observes state at step t: $s_t \in \mathcal{S}$

Produces action at step t : $a_t \in \mathcal{A}(s_t)$

gets resulting reward: $r_{t+1} \in \mathbb{R}$

and resulting next state s_{t+1}



On the Degree of Abstraction

- Actions can be low level (e.g., voltages to motors), or high level (e.g., accept a job offer), "mental" (e.g., shift in focus of attention), etc.
- States can be low-level "sensations", or they can be abstract, symbolic, based on memory, or subjective (e.g., the state of being "surprised" or "lost").
- An RL agent is not like a *whole* animal or robot.
- Reward computation is in the agent's environment because the agent cannot change it arbitrarily.
- The environment is not necessarily unknown to the agent, only incompletely controllable

The Agent Learns a Policy

- Policy at step t, π_t:
  - a mapping from states to action probabilities $\pi_t(s, a)$: probability that $a_t = a$ when $s_t = s$
- If the policy does not change, the problem can be seen as an MDP.
- Roughly, the agent's goal is to get as much reward as it can over the long run
- Reinforcement learning methods specify how the agent changes its policy as a result of experience, i.e. RL aims at changing the transition probabilities such that a better MPD (with lower cost/higher expected reward) is reached.
- Problem: We only know how to solve fixed MDP in the asymptotic case

Goals and Rewards

- Is a scalar reward signal an adequate notion of a goal? — maybe not, but it is surprisingly flexible
- A goal should specify what we want to achieve, not how we want to achieve it
- A goal must be outside the agent's direct control — thus outside the agent
- The agent must be able to measure success:
    - explicitly;
    - frequently during its lifespan

The reward hypothesis (Sutton)

- That all of what we mean by goals and purposes can be well thought of as the maximisation of the cumulative sum of a received scalar signal (reward)
- A sort of null hypothesis, time scales, stopping criteria needed
- Probably ultimately wrong, but so simple we have to disprove it before considering anything more complicated
- MD problems are solved once optimal values are known

Estimation problems

- In the machine management example, costs where deterministic. In general we have to estimate the stochastic cost in order to find for each state their averages (and variance)
- For a fixed policy, the agent has to visit all states (irreducibility and egodicity!) in order to sample the stationary probability of the MDP
- The agent has to change its policy such that the MDP improves towards an optimal MDP

Where now?

- A fixed policy in a MDP transforms a Markov Chain into a Markov Chain with a (generally) different transition matrix
- If the MDP is known, solve the EV-problem, calculate cost for the eigenvector
- Dijkstra's algorithm: visit all states, keep track of distance to starting state
- Assumptions were
    - MDP with known transition probabilities (deterministic for Dijkstra)
    - (random) immediate cost/reward (lengths for Dijkstra)
    - global information is available
- If these assumptions do not hold, RL is an option.
- On-policy reinforcement learning algorithms are often based on MDPs in a strict sense ($\mathcal{Q}$-learning is not the best example for MDPs)

Next time: Optimality

Bellman's Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. (1957)