CS4200/CS5200, On-line Machine Learning

Class 9: Reinforcement Learning

Yuri Kalnishkan

Department of Computer Science
Royal Holloway, University of London

2018/19

Class Outline

1. Environment and Policy

2. Value of a State

3. Bellman Equation and Dynamic Programming

References

[TM] T. M. Mitchell, “Machine Learning”, McGraw-Hill, 1998, Chapter 13.

[SB] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction”, 2nd edition, The MIT Press, 2018

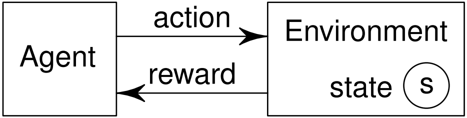
[CS] C. Szepesvári “Algorithms for Reinforcement Learning”, Morgan & Claypool, 2010

[JT] J. N. Tsitsiklis, On the Convergence of Optimistic Policy Iteration, JMLR 3 (2002) 59-72

[WD] C. J. C. H. Watkins and P. Dayan, Technical Note: Q-Learning, Machine Learning, 8, 279-292 (1992)

Principal Setup

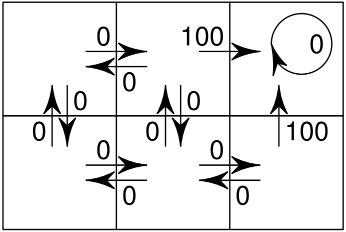
- a learning agent interacts with an environment
  — the agent takes an action
  — the environment gives a reward and changes its state



- we make the Markov assumption: all activity (choice of the action, reward, state to go to) depends on the current state rather then the whole history
  — given the current state, the future is independent of the past
- the current state is visible to the agent

Gridworlds

- gridworlds are toy examples used to illustrate principles of reinforcement learning



(after [TM], Fig. 13.2)

- the agent can move from a square to a neighbouring square; the rewards are shown on the diagram
- here the top right corner is a goal state (or terminal state); further moves from it are neither possible nor needed

Deterministic Environment

- this example is a deterministic environment
  — the reward and the state we move into are functions of the current state and action taken
- let S be the set of all states and A be the set of all actions; if the environment is deterministic, then
  — reward $r = \text{Reward}(s, a)$, where $\text{Reward} : S \times A \to \mathbb{R}$ is a function
  — state we move into $s = \text{State}(s, a)$, where $\text{State} : S \times A \to S$ is a function
- the environment can be described by two functions

Stochastic Environment

- suppose we are controlling a robot in a real-life situation
- there is uncertainty as to what happens after an action — the reward we get and the state we move into after taking an action a in a state s can be modelled by random variables $\mathrm{Reward}_{s,a}$ and $\mathrm{State}_{s,a}$
- the environment may be described by a collection of distributions on $S \times \mathbb{R}$ — there is one distribution for each pair (s, a)
- this is called a Markov Decision Process (MDP)
- we assume the MDP is stationary, i.e., if we return to state s and choose the same action a, we are faced with the same probabilities

Finite MDP

- we will be dealing with the case of finite sets of states S and actions A
- an MDP specifies:
  — the probabilities that upon choosing an action a in a state s we move to a state s': $\Pr(s' \mid s, a)$
  — the random variable $R_{s,a}$ giving us the reward if we choose an action a in a state s
- we assume that all rewards are bounded
  — more technical assumption: the expectation and variance of $R_{s,a}$ exist

Policy

- a deterministic policy is a function from states to actions $\pi : S \to A$
  — given a state, it tells us what action we should take
- a stochastic policy can toss a coin before it decides what action to take
  — in other words, a stochastic policy is a set of distributions on the set of actions, one for each state
- in the case of a finite MDP, a stochastic policy π specifies probability $\pi(a \mid s)$ of taking an action a in a state s

Cumulative Reward

- consider an MDA and a policy π
- suppose that we start in a state s_0 and follow a policy π
  — we take an action A_0 (a random variable), get reward R_1 and go to the state S_1 (both random variables)
  — we then take an action A_1, get reward R_2 and go to the state S_2 etc
  — we get a sequence $s_0, A_0, R_1, S_1, A_1, R_2, S_2, A_3, \ldots$
- we want a policy to bring high reward
  — quickly
- pick a discounting factor $\gamma \in [0, 1]$
- the discounted cumulative reward is

$$G_t = R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \ldots$$

Discounting

- the choice of γ reflects our preferences to immediate rewards vs future rewards
  — the value $\gamma = 0$ implies that only the immediate reward matters
  — the value $\gamma = 1$ implies that the moment of time when the reward arrives does not matter at all
- mathematically the values $\gamma < 1$ make sure the series converges
  — the sequence of rewards $r_1, r_2, r_3, \ldots$ may be finite (if we run into a terminal state, all rewards are zeros from some point) or infinite (even if the set of states is finite)
  — but if the rewards are bounded and $\gamma < 1$, the series $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$ always converges

Value of a State

- the value

$$V_\pi(s_0) = \mathbf{E} G_t = \mathbf{E}(R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots)$$

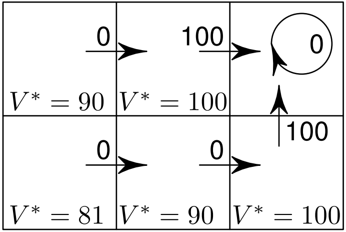
shows how much reward V earns on average if we start from state s_0

Optimal Value Function

- we want to find an optimal policy maximising rewards
- define $V^*(s) = \sup_\pi(s)$
  — this is the value of the discounted cumulative reward if we follow an optimal policy from s

Gridworld Example

- assume $\gamma = 0.9$
- clearly, the best we can do in the gridworld example is to run towards the exit asap



Optimal Policy

- we have actually constructed an optimal policy π^* such that
$$V^*(s) = V_{\pi^*}(s)$$
— our optimal policy achieves the optimal values for all s
- in a general case, is there such a policy?
— are the values $V^*(s)$ actually achieved by the same π for different states s?
- we can define an optimal policy as π^* such that for any other policy π and any state s we have $V_{\pi^*}(s) \geq V_\pi(s)$
— does an optimal policy exist?
— does it achieve $V^*(s)$?
- the answers to all these questions are positive (at least for finite MDPs) but this requires further study

1. Environment and Policy

2. Value of a State

3. Bellman Equation and Dynamic Programming