COMP3667: Reinforcement learning practical 3

Markov decision processes

robert.lieck@durham.ac.uk

1 Overview

Welcome to the third reinforcement learning practical. In this practical, we will try to better understand the concepts introduced in the last lecture by implementing and solving some of them using basic Python/NumPy. We will:

- Define and sample from a simple Markov chain.
- Extend to a Markov reward process and estimate approximate and compute exact state values.
- Extend to a Markov decision process and explore the influence of the policy.

2 Markov Chain

- Using Python and NumPy, define a Markov chain with $n_S = 10$ states $\mathcal{S} = \{0, \ldots, n_S - 1\}$ and a transition matrix $\mathcal{P}$ such that there is a probability of 0.9 for transitioning up ($s \to s+1$) and 0.1 for transitioning down ($s \to s+1$), wrapping around for the highest/lowest state. Only for $s = 0$ (the lowest state) there should be a probability of 0.8 for staying there and 0.1 of transitioning to the next higher/lower state. Check the normalisation of $\mathcal{P}$.
- *Technical side remark:* Why can we use exact comparison (`==`) here, while we should normally use something like `np.isclose(x, y)` to compare floating point numbers?
- Write a function to sample n_E episodes of length l_E (always start in state $s = 0$). Generate episodes for $n_E = 10$ and $l_E = 100$ and plot them over time (i.e. each episode as a separate line with time on the horizontal axis and states on the vertical axis). What would you expect? Does the Markov chain behave as expected?
- Let $p(s_t)$ be the marginal distribution over the states at time t. Remember, a marginal distribution is when you do not know the other (potentially very informative/relevant) variables. So here, it is just a distribution over states, where we do not know what the previous state was (the previous state is marginalised out).
  - What is the marginal distribution over states at time $t + 1$?
  - Sample another 100 episodes of length 100 and plot the marginal distribution over states
    - for the first 50 time steps
    - for the second 50 time steps.

    You can do this using the `plt.hist` function after cutting all episodes appropriately and flattening them (two histogram plots can be done side by side by providing a tuple of data). Do you see a difference? Can you explain it?
  - **Advanced** *[please, skip this question and come back to it later if you want, unless the solution is totally obvious to you and you love linear algebra]*: What is the *stationary* distribution $\bar{p}(s)$ over states? This is defined by the following property: When you assume the stationary distribution as the marginal distribution at time t and compute the distribution at time $t + 1$, it is again the stationary distribution (it's stationary, it does not change). Mathematically speaking, it is an eigenvector of the transition operator $\mathcal{P}$ with eigenvalue 1. Solve this by rewriting the condition as a matrix/vector equation and use `np.linalg.eig` to compute the eigenvalue and eigenvectors (note that `np.linalg.eig` computes right eigenvectors). Compare the result to your estimates of the marginal distribution above. Which one is more similar to the stationary distribution?

3 Markov Reward Process

- Define a reward function that returns a reward of 1 when the chain enters state $s = 0$. Adapt your sampling function from above to also return rewards (along with the state at every time step in all episodes). Sample 1000 episodes of length 100 and plot the average reward for each time step. Does it converge? To what value? Compare to the probability of being in state 0 in the stationary distribution (0.47).
- Assuming a discout of $\gamma = 0.5$, what is the value of being in states $s \in \{n_S - 1, 0, 1\}$? Estimate by sampling episodes from these states (adapt you sampling function to take an initial state).
- In the lecture we had the state value defined as

  $$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s') \ , \tag{1}$$

  where $\mathcal{R}_s$ is the reward for *being* in state s. Here we get a reward for *transitioning* to a state ($s = 0$). How would you need to adapt the equation for the state value to account for that?
- Manually check the estimated state value for state $s = 0$. Is the Bellman quality approximately fulfilled?
- The exact state values can be computed by rewriting the Bellman equation in matrix form and solving it (as in the lecture but again adapted because we get a reward not for *being in* but for *transitioning to* a state):

  $$v = \gamma \mathcal{P}(\mathcal{R} + v) \tag{2}$$
  $$\Leftrightarrow \quad (\mathrm{I} - \gamma \mathcal{P})v = \gamma \mathcal{P}\mathcal{R} \tag{3}$$

  Use `np.linalg.solve` to compute the exact state values.
- The value for state $s = 0$ that you (a) estimated by sampling and (b) computed for checking the estimate both did not exactly match the true state value. Was the computed value (b) better or worse than the estimate from sampling (a)?

4 Markov Decision Process (MDP)

- Add two actions “up” and “down” to your Markov reward process to obtain an MDP. Define the transition matrix such that for action $a = 0$ (“up”) there is a 0.9 probability of transitioning to the next higher state and a 0.1 probability of transitioning to the next lower state (wrapping around in both directions). For action $a = 1$ (“down”) it should be the other way around.
- Extend your function for sampling episodes accordingly by passing the policy as an additional parameter and include the actions along with the state and reward.
- Define a uniform policy, which takes both actions with probability 0.5. Estimate the state value for states $s \in \{n_S - 1, 0, 1\}$ by sampling episodes under the uniform policy. Do the resulting values make sense? *Note:* If you defined your sampling function above in a generic way, you can reuse it unchanged.
- Change the policy to always go one step up and re-estimate the state values. Why is $v(s = n_S - 1)$ less than 0.5?
- How would you change the policy to make it better? Try out and re-estimate, try to achieve the highest values for states $s \in \{n_S - 1, 0, 1\}$.
