Reinforcement Learning Assignment 2

Levente Foldesi (s3980456), Samuele Milanese (s3725294)

December 23, 2021

Question 1

The equation for calculating the policy evaluation is the following:

$$v^{k+1}(s) = \sum_a \pi(a|s) \sum_{s_{t+1}} \sum_r p(s_{t+1}|s,a)[r + \gamma V^k(s_{t+1})]$$

It is a recursive method to calculate the optimal policy. Here $\pi(a|s)$ is the policy based on the action, $p(s_{t+1}|s,a)$ is the next state based on the current action and state, r is the reward and γ is a parameter that influences how far in the future the agent can see. We decided to take $\gamma = 1$ for simplicity (taking as reference the example shown in the lecture).
The first iteration then looks like this:

$$V^1(s=1) = 0.2[3+0] + 0.8[2+0] = 2.2$$
$$V^1(s=2) = 0.5[-3+0] + 0.5[4+0] = 0.5$$
$$V^1(s=3) = 0.2[-3+0] + 0.8[10+0] = 7.4$$
$$V^1(s=4) = 0.4[-1+0] + 0.6[20+0] = 11.6$$
$$V^1(s=5) = 0 (\text{This is by default})$$

Then the second iteration is like the following:

$$V^2(s=1) = 0.2[3+7.4] + 0.8[2+0.5] = 4.08$$
$$V^2(s=2) = 0.5[-3+2.2] + 0.5[4+11.6] = 7.4$$
$$V^2(s=3) = 0.2[-3+2.2] + 0.8[10+0] = 7.84$$
$$V^2(s=4) = 0.4[-1+0.5] + 0.6[20+0] = 11.8$$
$$V^2(s=5) = 0 (\text{This is by default})$$

The third:

$$V^3(s=1) = 0.2[3+7.84] + 0.8[2+7.4] = 9.688$$
$$V^3(s=2) = 0.5[-3+4.08] + 0.5[4+11.8] = 8.44$$
$$V^3(s=3) = 0.2[-3+4.08] + 0.8[10+0] = 8.216$$
$$V^3(s=4) = 0.4[-1+7.4] + 0.6[20+0] = 14.56$$
$$V^3(s=5) = 0 (\text{This is by default})$$

As it can be seen the results tend to converge: State 4 is very high compared to the others, and state 3 seems to be the least desirable. So after 7 iterations, we got the following results:

$$V^7(s=1) = 16.8$$

$$V^7(s=2) = 16.1$$

$$V^7(s=3) = 10.2$$

$$V^7(s=4) = 17.8$$

Then the conclusion is that the route 1-2-4-5 seems to be the most rewarding.

Question 2

1

We use TD(0) update rule to compute the value after trajectories:

$$V(s_t) := V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]$$

τ_1: ⟨play,run,play⟩

$$V_{\text{Play}} = V_{\text{Play}} + 0.5[2 + 0.9 * V_{\text{Play}} - V_{\text{Play}}]$$

$$V_{\text{Play}} = 0 + 0.5[2 + 0.9 * 0 - 0] = 1$$

$$V_{\text{Play}} := 1$$

τ_2: ⟨play,shoot,goal⟩

$$V_{\text{Play}} = V_{\text{Play}} + 0.5[2 + 0.9 * V_{\text{Play}} - V_{\text{Play}}]$$

$$V_{\text{Play}} = 1 + 0.5[20 + 0.9 * 1 - 1] = 10.95$$

$$V_{\text{Play}} := 10.95$$

2

This time around we use the Q-Learning rule to update the values of the states after the episodes:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma * \max_{a \in A} Q(s_{t+1}, a_t) - Q(s_t, a_t)]$$

episode ⟨play,run,play⟩:
Initially: $Q(\text{play}, \text{run}) = 0$ as well as any other state-action pair, hence $\max_{a \in A} Q(\text{play}, a_t) = 0$

$$Q(\text{play}, \text{run}) := 0 + 0.5[2 + 0 - 0]$$

$$Q(\text{play}, \text{run}) := 1$$

episode ⟨play,pass,play⟩:
$Q(\text{play}, \text{pass}) = 0$, however, now $\max_{a \in A} Q(\text{play}, a_t) = 1$, because $Q(\text{play}, \text{run}) = 1$

$$Q(\text{play}, \text{pass}) := 0 + 0.5[3 + 0.9 * 1 - 0]$$

$$Q(\text{play}, \text{pass}) := 1.95$$

episode ⟨play,pass,goal⟩:
Initially $Q(\text{play}, \text{pass}) = 1.95$ and $\max_{a \in A} Q(\text{goal}, a_t) = 0$, since no goal-state was updated

$$Q(\text{play}, \text{pass}) := 1.95 + 0.5[8 + 0.9 * 0 - 1.95]$$

$$Q(\text{play}, \text{pass}) := 4.975$$

3

This time we need to take the next action into account as well so the main equation will look like the following:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma * Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

The two episodes made from the new table:
(Play,Run,Play,Shoot)
(Play,Run,Play,Run)
first:s_t second:a_t third:s_{t+1} fourth:$a_t + 1$
(Reward is 2 for both cases)
We know that every $Q(s_t, a_t)$ is zero at the first iteration, hence:

$$(\text{first episode}) = 0 + 0.5[2 + 0.9 * 0 - 0]$$

$$(\text{first episode}) = 1$$

Then, using the value which we obtained before in the second episode's $Q(s_{t+1}, a_{t+1})$ part we get the following equation:

$$(\text{second episode}) = 0 + 0.5[2 + 0.9 * 1 - 0]$$

$$(\text{second episode}) = 1.45$$

Question 3

In order to update the weights of the linear model after the trajectory ⟨play,run,play⟩ we use this formula (according with the Sutton and Barto (2018) book):

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \cdot \delta_t \cdot \mathbf{e}_t$$

Firstly, we take the estimate $\hat{Q}(\text{play, run;}\mathbf{w})$

$$\hat{Q}(\text{play, run;}\mathbf{w}) = w_1 + w_2 \cdot f(s, a)$$

Knowing $w_1 = 0$, $w_2 = 0$ initially and $f(s, a) = -1$ for action run

$$\hat{Q}(\text{play, run;}\mathbf{w}) = 0 + 0 \cdot (-1) = 0$$

Now we can compute δ_t:

$$\delta_t = R_t + (\gamma \max_{a \in A} \hat{Q}(s_{t+1}, a; \mathbf{w}) - \hat{Q}(s_t, a_t; \mathbf{w}))$$

with $R = 2$, $\gamma = 1$. Furthermore, we know that $\hat{Q}(s_{t+1}, a; \mathbf{w}) = 0$ for every action since all weights are initially 0

$$\delta_0 = 2 + (1 \cdot 0 - 0) = 2$$

The gradient vector is:

$$\nabla\hat{Q} = (\frac{\partial\hat{Q}(s_t, a_t; \mathbf{w})}{\partial w_1}, \frac{\partial\hat{Q}(s_t, a_t; \mathbf{w})}{\partial w_2})$$

$$\nabla\hat{Q} = (1, f(s, a))$$

Since it is the first time-step:

$$\mathbf{e} = \nabla\hat{Q}$$

Finally we can update the weights:

$$w_1 := 0 + 0.5 \cdot 2 \cdot 1 = 1$$

$$w_2 := 0 + 0.5 \cdot 2 \cdot f(\text{play, run}) = 1 \cdot (-1) = -1$$

References

https://www.tu-chemnitz.de/informatik/KI/scripts/ws0910/ml09_6.pdf
Sutton, R. S., Barto, A. G. (2018). Reinforcement learning: An introduction (Second edition). The MIT Press.