Machine Learning Worksheet 6

Reinforcement Learning

1 Value Functions and Rewards

Read chapters 3.3 and 3.7 in Sutton & Barto's Book "Reinforcement Learning". If you don't have the book, go buy it (it is worth the money), get it from the library or look up the HTML online version here:

`http://www.cs.ualberta.ca/~sutton/book/the-book.html`.

Exercise 1. Consider an episodic task of length τ, where the return R_t at time step t is defined as the sum of future rewards

$$R_t = (r_{t+1} + r_{t+2} + \cdots + r_\tau) = \sum_{k=0}^{\tau-(t+1)} r_{t+k+1}$$

In this task the reward for reaching the goal is $+1$, the reward for running into the edge of our world is -1 and zero the rest of the time. Are the signs of these rewards important, or only the intervals between them? Prove, using the return definition above and the definition of the value function V^π, that adding a constant C to all the rewards adds a constant, K, to the values of all states, and thus does not affect the relative values of any states under any policies. Give an example.

Exercise 2. Now consider a continuous discounted task, where the return R_t at time step t is the future discounted sum of future rewards

$$R_t = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k\, r_{t+k+1}$$

What happens now, if we add a constant C to the rewards? Would this have any effect, or would it leave the task unchanged as in the episodic task above? What is K in terms of C and γ? Could we use undiscounted sums of rewards for continuous tasks?

Exercise 3. Imagine that you are designing a robot to escape from a maze. You decide to give it a reward of $+1$ for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes—the successive runs through the maze—so you decide to treat it as an episodic task, where the goal is to maximize expected total reward. After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve?

2 Gridworld and SARSA

Read chapter 6.4 and 7.5 in Sutton & Barto's book "Reinforcement Learning" (if you need a reminder of how Q-learning works, also read 6.5).

The following gridworld example is a very small toy problem to test some algorithms. Your agent can move up, down, left, or right to adjacent cells, but not diagonally. It always starts at the starting point **s** and needs to find the goal point **g**. In the middle of the world is a dangerous pitfall **p**. If the agent falls into the pitfall, it receives a reward of -5 and has to go back to **s**. Upon reaching the goal state, it receives $+5$. When the agent leaves the gridworld at the sides, it receives -1 and is placed back on the field it came from. Every other state gives zero reward.

x_1	x_2	x_3
s	p	g
x_4	x_5	x_6

Table 1: Gridworld with start **s**, goal **g** and pitfall **p**.

Exercise 1. Why is the SARSA-Algorithm called that way? How does it work and what is the difference to Q-Learning? What does *on-policy* and *off-policy* mean?

Exercise 2. What is an ϵ-greedy policy and why do we need it? Why can't we always greedily follow the policy? How would you select ϵ at the beginning of a learning task and how does it change over time?

Exercise 3. Assume all state values are initialized to zero and the agent randomly walks on the following path: s $\rightarrow x_4 \rightarrow x_5 \rightarrow$ p. Which $Q(s,a)$ values will be updated with Sarsa(0) and which $Q(s,a)$ values would Sarsa(λ) update during this walk? Calculate the new values for Sarsa(0) and Sarsa(λ). Does that mean it is not a good idea to go from x_4 to x_5 after all? (use $\alpha = 0.1, \gamma = 0.9, \lambda = 0.5$ for your calculations)

$\star$ **Exercise 4.** (This exercise is optional and will give you bonus points.) How does the Q-table look like when it is converged learning with Sarsa(0)? Write a little computer program in a language of your choice and let the agent learn to find the goal. Hand in your source code **including comments** and the resulting Q-values for each state-action pair in a table.