04.01.2020

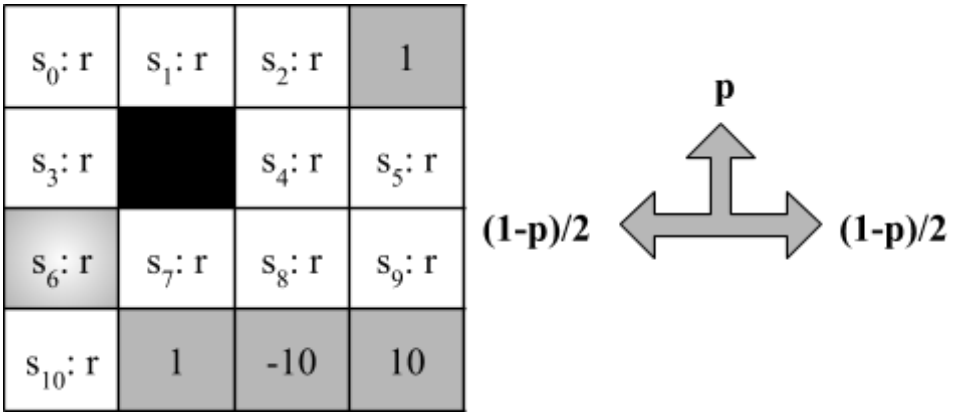
CENG 461: Artificial Intelligence
Homework #2
Value Iteration, Policy Iteration and Q-Learning

DUE DATE: 12.01.2020 - 23:55

1. Implementation

You are expected to implement Value Iteration (VI) and Policy Iteration (PI) algorithms for a Markov Decision Process (MDP) and the Q-learning algorithm for Reinforcement Learning assuming the same process but without the knowledge of state transition probabilities for available actions.

The problem is as the following. An agent is going to explore the environment with the transition properties given below:



"s_i" indicates each state, "r" stands for the state rewards and "p" stands for the probability of the agent actually going in the chosen direction.

Other parameters are:

- d: discount factor
- e (epsilon): exploration probability
- a (alpha): learning rate
- N: number of experiments for Q-learning

The agent starts exploring from the cell painted with radial gradient. It stays in the same cell if it tries to move into a wall (outer edges of the table) or a block (the cell painted in black). The cells painted in gray are terminal states, where the agent has no available actions and the experiment finishes.

Your implementation should be flexible to run experiments with different parameter sets. You should print utility values in each VI/PI iteration for each state and to display the final values and policies found as a result for all algorithms. Additionally for Q-learning, you should record and plot the utility and policy errors every 100 experiments, assuming that the VI/PI result is optimal.

2. Reporting

VI and PI algorithms should converge to the same utility values and hence policies. For PI, initialize the policy all with **up** actions, and in case of equality, the preference order is **up, down, right, left**. There is no need to use random number generators for VI and PI.

For Q-learning, set the seed to 62 for numpy's random sampling module so that the experiments can be repeated the same way (`numpy.random.seed(62)`). You need to generate a random number in the following order:

- Random float(np.random.rand()) to explore if smaller than epsilon
- Random integer(np.random.randint(0,4)) to decide exploration direction (0, 1, 2, 3 for up, down, right, left, respectively)
- Random float(np.random.rand()) to decide the result of an action (**straight** if between 0 and 0.8, **left** if between 0.8 and 0.9 or **right** if between 0.9 and 1.0)

Run VI and PI algorithms with **r:0**, **d:1** and **p:1**. You should obtain the following tables for utility values at each iteration. (upper:VI, lower:PI)

	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
it0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
it1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
it2	0.000	0.000	1.000	0.000	0.000	1.000	0.000	1.000	0.000	10.000	1.000
it3	0.000	1.000	1.000	0.000	1.000	10.000	1.000	1.000	10.000	10.000	1.000
it4	1.000	1.000	1.000	1.000	10.000	10.000	1.000	10.000	10.000	10.000	1.000
it5	1.000	1.000	10.000	1.000	10.000	10.000	10.000	10.000	10.000	10.000	1.000
it6	1.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000
it7	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000
it8	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000
	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
it0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
it1	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000
it2	0.000	0.000	1.000	0.000	1.000	1.000	0.000	1.000	10.000	10.000	1.000
it3	0.000	1.000	1.000	0.000	10.000	10.000	1.000	10.000	10.000	10.000	1.000
it4	1.000	1.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000
it5	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000
it6	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000

Comment on the resulting utilities. What is the optimal policy? With **e:0**, **a:0.1** and **N:1000**, after Q-learning what are the resulting q values and the optimal policy? Does it help to increase N? Comment on your findings.

a. Now, run the same experiments this time with **r:-0.01**. Report your findings. Comment on what changed for VI and PI? What about Q-learning? What is your diagnosis for the problem?

b. Update the discount factor as **d:0.2**. Comment on the changes in optimal policies found by VI and PI.

c. Change the discount factor back to **d:1**. Update rewards as **r:5**. Again comment on the results of VI and PI.

d. Try VI and PI with **d:1**, **r:-0.01** and **p:0.5**. What happens?

e. Run you experiments with **d:0.9**, **r:-0.01**, **p:0.8** for VI and PI. Report the results for utility values and the final policy. You'll use these values to evaluate the Q-learning algorithm in the next steps. Inaddition to your answers to the questions, please also include the plots and resulting q value matrix and the final policy of each experiment in your report:

  i. For Q-learning, in the beginning, continue with **e:0**, **a:0.1** and **N:1000**. Now the agent has a stochastic transition, what can you say about the results?

  ii. What happens if you increase **N:10000**?

  iii. Now let's introduce some exploration possibility for the agent, **e:0.1**. What happens? Comment on the plot.

  iv. Now let's examine the learning rate. How does the performance change if we update it as **a:1**?

  v. Implement a counter for each state-action pair and increment it everytime that pair is experienced. Set the learning rate to be `1/count` for each update. Comment on the result.

  vi. Finally let's increase the number of experiments **N:100000** (with **e:0.1** and **decaying** learning rate in the previous step). Could you reach the optimal policy?

  vii. What is the best parameter set you can come up with, that reach the optimal policy in least number of iterations?