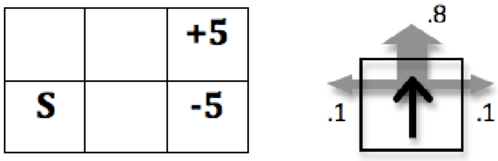
THE HONG KONG UNIVERSITY OF SCIENCE & TECHNOLOGY

Machine Learning
Homework 4 Solutions

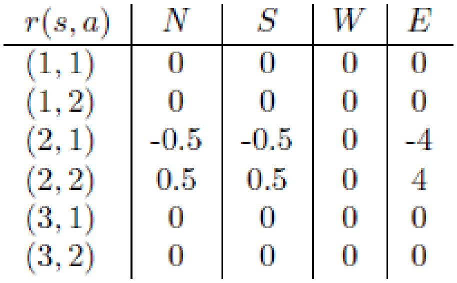
For self-practice. No need to submit your work.
Everyone gets 4 points nonetheless.

Solutions will be provided later.

Question 1: Consider an agent that acts in the gridworld shown below. The agent always starts in state $(1, 1)$, marked with the letter S. There are two terminal goal states, $(3, 2)$ with reward $+5$ and $(3, 1)$ with reward -5. Rewards are 0 in non-terminal states. (The reward for a state is received as the agent moves into the state.) The transition function is such that the intended agent movement (North, South, West, or East) happens with probability 0.8. With probability 0.1 each, the agent ends up in one of the states perpendicular to the intended direction. If a collision with a wall happens, the agent stays in the same state.



The expected immediate reward function $r(s, a) = \sum_{s'} r(s, a, s')P(s'|s, a)$ is as follows:



$r(s,a)$	N	S	W	E
$(1,1)$	0	0	0	0
$(1,2)$	0	0	0	0
$(2,1)$	-0.5	-0.5	0	-4
$(2,2)$	0.5	0.5	0	4
$(3,1)$	0	0	0	0
$(3,2)$	0	0	0	0

(a) Assume the initial value function $Q_0(s, a) = 0$ for all states s and actions a. Let $\gamma = 0.9$. The Q-function Q_1 after the first value iteration is the same as $r(s, a)$. What is the Q-function Q_2 after the second value iteration? What is the greedy policy π_2 based on Q_2. In case of ties, list all tied actions.

(b) Suppose the agent does not know the transition probabilities and the reward function, and it tries to learn by interacting with the environment. Assume the Q-learning algorithm is used with $Q(s, a) = 0$ initially. Let $\alpha = 0.1$ and $\gamma = 0.9$. Update the Q-function using the following experience tuples. Show the function after each update.

s	a	r	s'
(2, 2)	E	5	(3, 2)
(2, 1)	N	0	(2, 2)
(1, 2)	E	0	(2, 2)
(1, 1)	N	0	(1, 2)

Give the greedy policy based on the latest Q function. In case of ties, list all tied actions.

Solution: (a) First, note that $V_1(s') = \max_{a'} Q_1(s', a')$ is as follows: Q_2 can be obtained from V_1 as follows:

s'	$V_1(s')$
$(1,1)$	0
$(1,2)$	0
$(2,1)$	0
$(2,2)$	4
$(3,1)$	0
$(3,2)$	0

$$Q_2(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) V_1(s').$$

Hence, we have:
$Q_2((1,1),N) = 0 + \gamma * (0.8 * V_1((1,2)) + 0.1 * V_1((1,1)) + 0.1 * V_1((2,1))) = 0,$
$Q_2((1,1),S) = 0 + \gamma * (0.8 * V_1((1,1)) + 0.1 * V_1((1,1)) + 0.1 * V_1((2,1))) = 0,$
$Q_2((1,1),W) = 0 + \gamma * (0.8 * V_1((1,1)) + 0.1 * V_1((1,1)) + 0.1 * V_1((1,2))) = 0,$
$Q_2((1,1),E) = 0 + \gamma * (0.8 * *V_1((2,1)) + 0.1 * V_1((1,2)) + 0.1 * V_1((1,1))) = 0,$
$Q_2((1,2),N) = 0 + \gamma * (0.8 * V_1((1,2)) + 0.1 * V_1((1,2)) + 0.1 * V_1((2,2))) = 0.36,$
$Q_2((1,2),S) = 0 + \gamma * (0.8 * V_1((1,1)) + 0.1 * V_1((1,2)) + 0.1 * V_1((2,2))) = 0.36,$
$Q_2((1,2),W) = 0 + \gamma * (0.8 * V_1((1,2)) + 0.1 * V_1((1,1)) + 0.1 * V_1((1,2))) = 0,$
$Q_2((1,2),E) = 0 + \gamma * (0.8 * V_1((2,2)) + 0.1 * V_1((1,1)) + 0.1 * V_1((1,2))) = 2.88,$
$Q_2((2,1),N) = -0.5 + \gamma * (0.8 * V_1((2,2)) + 0.1 * V_1((1,1)) + 0.1 * V_1((3,1)) = 2.38,$
$Q_2((2,1),S) = -0.5 + \gamma * (0.8 * V_1((2,1)) + 0.1 * V_1((1,1)) + 0.1 * V_1((3,1))) = -0.5$
$Q_2((2,1),W) = 0 + \gamma * (0.8 * V_1((1,1)) + 0.1 * V_1((2,1)) + 0.1 * V_1((2,2))) = 0.36$
$Q_2((2,1),E) = -4 + \gamma * (0.8 * V_1((3,1)) + 0.1 * V_1((2,1)) + 0.1 * V_1((2,2))) = -3.64,$
$Q_2((2,2),N) = 0.5 + \gamma * (0.8 * V_1((2,2)) + 0.1 * V_1((1,2)) + 0.1 * V_1((3,2))) = 3.38,$
$Q_2((2,2),S) = 0.5 + \gamma * (0.8 * V_1((2,1)) + 0.1 * V_1((1,2)) + 0.1 * V_1((3,2))) = 0.5,$
$Q_2((2,2),W) = 0 + \gamma * (0.8 * V_1((1,2)) + 0.1 * V_1((2,2)) + 0.1 * V_1((2,1))) = 0.36,$
$Q_2((2,2),E) = 4 + \gamma * (0.8 * V_1((3,2)) + 0.1 * V_1((2,2)) + 0.1 * V_1((2,1))) = 4.36.$
In summary, we have

$Q_2(s,a)$	N	S	W	E
$(1,1)$	0	0	0	0
$(1,2)$	0.36	0.36	0	2.88
$(2,1)$	2.38	-0.5	0.36	-3.64
$(2,2)$	3.38	0.5	0.36	4.36
$(3,1)$	0	0	0	0
$(3,2)$	0	0	0	0

The greedy policy $\pi(s) = \arg\max_a Q_2(s,a)$ is as follows: $\pi_2((1,1)) = \{N,S,W,E\}$, $\pi_2((1,2)) = E$, $\pi_2((2,1)) = N$, $\pi_2((2,2)) = E$.

(b) The Q-learning update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)].$$

For $s = (2,2), a = E, r = 5, s' = (3,2)$, $Q((2,2),E) = 0 + 0.1 * (5 + 0.9 * 0 - 0) = 0.5$.
For $s = (2,1), a = N, r = 0, s' = (2,2)$, $Q((2,1),N) = 0 + 0.1 * (0 + 0.9 * 0.5 - 0) = 0.045$.
For $s = (1,2), a = E, r = 0, s' = (2,2)$, $Q((1,2),E) = 0 + 0.1 * (0 + 0.9 * 0.5 - 0) = 0.045$.
For $s = (1,1), a = N, r = 0, s' = (1,2)$, $Q((1,1),N) = 0 + 0.1 * (0 + 0.9 * 0.045 - 0) = 0.00405$.
The greedy policy $\pi(s) = \arg\max_a Q(s,a)$ is as follows: $\pi((1,1)) = N$, $\pi((1,2)) = E$, $\pi((2,1)) = N$, $\pi_2((2,2)) = E$.

$Q(s,a)$	N	S	W	E
$(1,1)$	0	0	0	0
$(1,2)$	0	0	0	0
$(2,1)$	0	0	0	0
$(2,2)$	0	0	0	0.5
$(3,1)$	0	0	0	0
$(3,2)$	0	0	0	0

$Q(s,a)$	N	S	W	E
$(1,1)$	0	0	0	0
$(1,2)$	0	0	0	0
$(2,1)$	0.045	0	0	0
$(2,2)$	0	0	0	0.5
$(3,1)$	0	0	0	0
$(3,2)$	0	0	0	0

$Q(s,a)$	N	S	W	E
$(1,1)$	0	0	0	0
$(1,2)$	0	0	0	0.045
$(2,1)$	0.045	0	0	0
$(2,2)$	0	0	0	0.5
$(3,1)$	0	0	0	0
$(3,2)$	0	0	0	0

$Q(s,a)$	N	S	W	E
$(1,1)$	0.00405	0	0	0
$(1,2)$	0	0	0	0.045
$(2,1)$	0.045	0	0	0
$(2,2)$	0	0	0	0.5
$(3,1)$	0	0	0	0
$(3,2)$	0	0	0	0

Question 2: Here is the parameter update rule for Deep Q-Networks:

$$\theta \leftarrow \theta - \alpha \nabla_\theta ([r(s,a) + \gamma \max_{a'} Q(s', a'; \theta^-)] - Q(s,a;\theta))^2$$

What do s, a, $r(s,a)$ and s' stand for? What about θ^-? What is the objective that the update rule is intended to achieve?

Solution: The tuple (s, a, $r(s,a)$, s') is an experience of the agent has with its environment. θ^- denotes the parameters of a target network, which is updated once in a while. The objective of the rule to change the parameters of the Q-network such that it better approximates the TD-target $r(s,a) + \gamma \max_{a'} Q(s', a'; \theta^-)$, which is an improved estimate of the value for the pair (s,a) given the experience tuple.

Question 3: Here is the update rule for the actor in the Actor-Critic algorithm:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a|s) \hat{A}^\pi(s,a),$$

where $\hat{A}^\pi(s,a) \leftarrow r + \gamma \hat{V}^\pi_\phi(s') - \hat{V}^\pi_\phi(s)$.

What do s, a, $r(s,a)$ and s' stand for? What about $\hat{V}^\pi_\phi(s)$? Intuitively, what does the update rule try to achieve?

Solution: The tuple (s, a, r, s') is an experience of the agent has with its environment. $\hat{V}^\pi_\phi(s)$ is an estimate of the value of the state s given by the current value network. The update rules tries to increase the probabilities of the actions that lead to high rewards and decrease the probabilities of the actions that lead to low rewards.

Question 4: (a) In the context of deep image classification, what is adversarial attack?

(b) The CW attack finds an adversarial example $\mathbf{x}'$ for a benign example $\mathbf{x}$ by solving the following optimization problem:

$$\min_{\mathbf{x}'} c||\mathbf{x} - \mathbf{x}'||_2^2 + l(\mathbf{x}')$$
$$\text{s.t. } \mathbf{x}' \in [0, 1]^n$$
$$\text{where } l(\mathbf{x}') = \max\{\max_{i \neq t} Z_i(\mathbf{x}') - Z_t(\mathbf{x}'), -\kappa\}.$$

What do the terms $Z_t(\mathbf{x}')$, $Z_i(\mathbf{x}')$ stand for? What are the key ideas behind this attack?

Solution: (a) In the context of deep image classification, adversarial attack means to add small perturbations to an input image that are (almost) imperceptible to human vision system, and that cause the classifier to misclassify the input with high confidence.

(b) $Z_i(\mathbf{x}')$ is the logit of class i and t stands for the target class. The CW attack aims to maximize the logit $Z_t(\mathbf{x}')$ of the target class and to minize the logits of all other classes $\max_{i \neq t} Z_i(\mathbf{x}')$. When then difference $Z_t(\mathbf{x}') - \max_{i \neq t} Z_i(\mathbf{x}')$ becomes large enough $\geq \kappa$, it turns its attention to minimize the pertubations $||\mathbf{x} - \mathbf{x}'||_2^2$.

Question 5: In some pixel-level explanations, we need to compute $\frac{\partial z_c(\mathbf{x})}{\partial x_i}$. The backpropagation algorithm for the task is given in Lecture 16. Suppose the last layer of the model is a Softamx layer. How would you change to algorithm if we are to compute $\frac{\partial \log P(y=c|\mathbf{x})}{\partial x_i}$?

Solution: Let $Z = \sum_c e^{z_c(\mathbf{x})}$. The first step of backprop should be changed as follows:

$$\begin{aligned}
\frac{\partial \log P(y = c|\mathbf{x})}{\partial u_j} &= \frac{\partial \log \frac{e^{z_c(\mathbf{x})}}{Z}}{\partial u_j} \\
&= \frac{\partial z_c(\mathbf{x})}{\partial u_j} - \frac{1}{Z}\frac{\partial \sum_{c'} e^{z_{c'}(\mathbf{x})}}{\partial u_j} \\
&= w_{cj} - \sum_{c'} P(y = c'|\mathbf{x}) w_{c'j}.
\end{aligned}$$

There is no need to change the rest of the algorithm.